

Advances in Computational Intelligence with Applications in Finance and Molecule Design

Von der Fakultät für Informatik,
Wirtschafts- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels

Doktor der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von Herrn Lars Elend

geboren am 14. Dezember 1992 in Norden

Erstgutachter:

Prof. Dr. Oliver Kramer

Department für Informatik

Carl von Ossietzky Universität Oldenburg

Externer Zweitgutachter:

Prof. Dr. Günter Rudolph

Fakultät für Informatik

Technische Universität Dortmund

Tag der Disputation: 20. Dezember 2023

Zusammenfassung

Verfahren aus dem Bereich **Computational Intelligence (CI)** können sehr vielseitig zur Lösung unterschiedlichster Probleme eingesetzt werden. In den letzten 15 Jahren haben insbesondere neue Entwicklungen im Bereich der Deep Neural Networks und die stark gestiegene Rechenleistung von Grafikprozessoren zu Fortschritten geführt. So konnten häufig bessere Ergebnisse zu bekannten Problemen gefunden werden als mit bisherigen Verfahren und bestimmte Probleme konnten überhaupt erst in akzeptabler Zeit gelöst werden. In dieser Arbeit werden **CI**-Verfahren für drei unterschiedliche Bereiche eingesetzt.

Im Finanzbereich ist eine wesentliche Aufgabe den Wert von Unternehmen genau zu beurteilen. Finanzinstitute basieren ihr Handeln in der Regel auf aufwendige Prognosen, die von Finanzanalysten erstellt wurden. Eine entscheidende Rolle spielen hierbei die erwarteten zukünftigen Gewinne eines Unternehmens, welche sich in den Gewinnen pro Aktie widerspiegeln. Für börsengehandelte US-Unternehmen gibt es regelmäßige Quartalsberichte, deren Informationen auch für eine datenbasierte Zeitreihenprognose genutzt werden können. Zu diesem Zweck werden in dieser Arbeit verschiedene Deep Neural Networks für die Prognose der Gewinne pro Aktie verglichen. Bei den Experimenten werden zudem unterschiedliche Gruppen von Unternehmen betrachtet. Hierbei konnte festgestellt werden, dass unsere Modelle insbesondere für die Unternehmen, die nicht im Finanzbereich tätig sind, gute Prognosen erstellen und sogar die Prognosen von Analysten übertreffen können. Ebenso wurde eine Unterteilung in mehrere Industriegruppen untersucht.

Auch in der Medizin können **CI**-Verfahren sinnvoll eingesetzt werden. Ein Anwendungsbereich ist die Entwicklung von Medikamenten gegen Viren. Ein Ansatz zur Vermeidung der Virusausbreitung im menschlichen Körper, ist die Blockade der Virusprotease, die für die Replikation desselben notwendig ist. Zu diesem Zweck wird ein geeigneter Proteaseinhibitor gesucht, der an die Protease bindet und zusätzlich bestimmte Eigenschaften erfüllt (u. a. Synthetisierbarkeit). In dieser Arbeit werden drei evolutionäre Verfahren zum Generieren von Proteaseinhibitoren vorgestellt: *Weighted Sum Evolutionary Molecule Search*, *Pareto Ranking Evolutionary Molecule Search* und *Evolutionary Molecule Generation Algorithm*. Letzterer nutzt intern ein Language Model für die Mutation. Für die Experimente werden zwei unterschiedliche String-Repräsentationen für Moleküle verwendet: Simplified Molecular-Input Line-Entry System (SMILES) und Self-Referencing Embedded Strings (SELFIES). Für einige Moleküle wurde eine weitergehende Molecular-Dynamics-Analyse durchgeführt, bei der einige Kandidaten verworfen und andere als vielversprechend bestätigt werden konnten. Der **CI**-Ansatz bietet im Vergleich zur herkömmlichen manuellen Suche nach geeigneten Molekülen, zwei zusammenhängende Vorteile: Die **CI**-Verfahren sind um mehrere Größenordnungen schneller und ermöglichen dadurch eine breitere Erkundung des riesigen Suchraums der möglichen Moleküle. Somit eignen sich diese Ansätze insbesondere für eine erste Auswahl von Molekülen, die anschließend von aufwendigeren, aber auch genaueren Verfahren genutzt werden können.

Als Drittes wird die neu entwickelte **Convolutional Self-Organizing Map (ConvSOM)** Architektur zur Visualisierung von Daten unter der Berücksichtigung höherwertiger Features vorgestellt. Die **ConvSOM** nutzt interne Informationen aus einem Convolutional Neural Network als Eingabe für eine Self-Organizing Map. Für die Untersuchung der Architektur wurden Experimente mit Bilddaten

durchgeführt. Hierbei ist das Ziel, nicht nur die rohen Pixelinformationen, sondern auch höherwertige Features zu nutzen, um semantische Zusammenhänge herzustellen. Zur Bewertung dieser die semantischen Zusammenhänge der resultierenden Karten, wurden mehrere Metriken entwickelt. Davon nutzen einige Metriken die Label der Samples. Es wird allerdings auch eine auf dem Deep Visual-Semantic Embedding Model aufbauende semantische Metrik beschrieben, die ohne Label auskommt. Für die gewählten Zielmetriken liefert die **ConvSOM** bessere Ergebnisse als die normale Self-Organizing Map.

Abstract

Computational intelligence (CI) methods can be used in various ways to solve a wide range of problems. In the last 15 years, the large increase in computing power of graphics processing units and multiple new developments in deep neural networks have led to advancements in the field. As a result, it has often been possible to achieve better results on known problems than with previous methods, and certain problems can now be solved with acceptable effort for the first time. In this thesis, I applied **CI** methods to three different areas.

In finance, an essential task is to accurately assess the value of companies. Financial institutions usually base their actions on elaborate forecasts prepared by financial analysts. A crucial role is played by the expected future earnings of a company, which are reflected in earnings per share. There are regular quarterly reports for publicly traded U.S. companies whose information can also be used for data-based time series prediction. This thesis compares different deep neural networks regarding their ability to forecast the earnings per share. The experiments also consider different groups of companies. I found that the proposed models can produce good forecasts, especially for the non-financial companies, and even outperform the analysts' forecasts. Likewise, a subdivision into several industry groups was examined.

CI methods can also be used to support drug development against viruses. One approach to preventing the spread of viruses in the human body is to block the viral protease, which is necessary for virus replication. Therefore, a suitable protease inhibitor is sought, which binds to the protease and additionally fulfills certain properties (e.g., synthesizability). In this thesis, three evolutionary methods for the generation of protease inhibitors are presented: weighted sum evolutionary molecule search, Pareto ranking evolutionary molecule search, and evolutionary molecule generation algorithm. The latter internally uses a language model for mutation. The experiments use two different string representations for molecules: simplified molecular-input line-entry system (SMILES) and self-referencing embedded strings (SELFIES). More extensive molecular dynamics analysis was performed for several selected molecules. The analysis allowed some molecules to be discarded and others to be confirmed as promising. The **CI** approaches offer two advantages over the traditional, mostly manual search for suitable molecules: The **CI** methods are several orders of magnitude faster and allow a broader exploration of the large search space of possible molecules. Thus, these approaches are particularly suitable for initial selection of molecules that can later be used for subsequent more elaborate, but also more accurate, methods.

Third, I present the newly developed **convolutional self-organizing map (ConvSOM)** architecture for visualizing data while taking into account higher-level features. The **ConvSOM** uses internal information from a convolutional neural network as input to a self-organizing map. Image data was used for the experiments. The goal is to use not only the raw pixel information, but also higher-level information to establish semantic relationships. In order to evaluate the semantic relationships of the resulting maps, several metrics have been developed. These metrics mostly use labels, whereas one metric based on the deep visual-semantic embedding model that does not use labels is also described. For the chosen target metrics, the **ConvSOM** returns better results than the regular self-organizing maps.

Contents

I	Introduction and Foundations	1
1	Introduction	3
1.1	Thesis Structure	4
1.2	Contributions	5
2	Computational Intelligence	7
2.1	Machine Learning	7
2.1.1	Supervised Learning	8
2.1.2	Unsupervised Learning	9
2.1.3	Reinforcement Learning	10
2.2	Artificial Neural Networks	11
2.2.1	Artificial Neuron	11
2.2.2	Activation Functions	12
2.2.3	Feed Forward Network	13
2.2.4	Backpropagation	13
2.3	Conclusion	14
II	Prediction	15
3	Earnings Prediction	17
3.1	Related Work	17
3.2	Time Series Prediction	18
3.3	Long Short-Term Memory	19
3.4	Temporal Convolutional Network	20
3.5	Data Preprocessing	21
3.6	Quality Measures	22
3.7	Experimental Analysis	22
3.7.1	Architecture and Meta Parameters	24
3.7.2	Financial Firms	25
3.7.3	Fama French Industries	25
3.7.4	Test of Best Model	26
3.8	Conclusion	26
III	Molecule Design	29
4	Foundations of Molecule Design	31
4.1	Motivation	31
4.2	Organic Chemistry	32
4.2.1	Atoms and Elements	32

4.2.2	Bonds	35
4.2.2.1	Intramolecular Bonds	35
4.2.2.2	Intermolecular Bonds	35
4.2.3	Molecules	37
4.2.3.1	1D / String Representations	37
4.2.3.2	2D Representations	39
4.2.3.3	3D Representations	40
4.2.3.4	Molecule characteristics	42
4.2.3.5	Representation characteristics	43
4.3	Evolutionary Algorithms	45
4.3.1	Parameters	46
4.3.2	Recombination	46
4.3.3	Mutation	47
4.3.4	Selection	47
4.4	Conclusion	48
5	Evolutionary Multi-Objective Approach	49
5.1	Related Work	49
5.2	Molecule Design Metrics	50
5.2.1	Binding Affinity	51
5.2.2	Synthetic Accessibility	51
5.2.3	Quantitative Estimate of Drug-Likeness	52
5.2.4	Natural Product-Likeness	53
5.2.5	Toxicity Filters	53
5.3	Evolutionary Molecule Search	53
5.3.1	Representation	54
5.3.2	Mutation	54
5.3.3	Fitness Evaluation	55
5.3.4	Weighted Sum Evolutionary Molecule Search	56
5.3.5	Pareto Ranking Evolutionary Molecule Search	56
5.4	Experiments	57
5.4.1	Metric Development	58
5.4.2	Candidate Comparison	60
5.5	Conclusion	62
6	Language Model-based Evolutionary Approach	63
6.1	Evolutionary Molecule Generation Algorithm	64
6.1.1	Representation	64
6.1.2	Neural Language Model	65
6.1.3	Evolutionary Algorithm with Language Model	67
6.2	Molecular Dynamics	68
6.3	Results and Discussion	70
6.3.1	Evolutionary Molecule Generation Algorithm	71
6.3.2	Molecular Dynamics	74
6.4	Conclusion	78
IV	Visualization	79
7	Convolutional Self-Organizing Map	81

7.1	Related Work	82
7.2	Self-Organizing Map	82
7.3	Convolutional Neural Network	84
7.4	Convolutional Self-Organizing Map	85
7.5	Quality Metrics	86
7.5.1	Kruskal Shepard Error	87
7.5.2	Cross Entropy	88
7.5.3	Minor Class Occurrence	88
7.5.4	Class Scatter Index	88
7.6	Visualization	89
7.7	Experimental Analysis	90
7.7.1	Experimental Settings	90
7.7.2	Quality Measure Results	91
7.7.3	Visualization Results	92
7.8	DeViSE as Semantic Metric	93
7.9	Conclusion	93
V	Closing	95
8	Conclusion	97
8.1	Earnings Prediction	97
8.2	Molecule Design	98
8.3	Convolutional Self-Organizing Map	100
VI	Appendices	103
A	Data Sets	105
B	Earnings Prediction	107
B.1	Data Sets	107
B.2	Workflow	108
C	Molecule Design	111
C.1	Workflow	111
C.2	Ligands Overview	111
	List of Figures	113
	List of Tables	115
	Literature	117
	Index of Parameters	137
	Acronyms	141
	Glossary	145
	Index	147

Part I

Introduction and Foundations

1 Introduction

Computational intelligence (CI) methods can be applied to many types of problems and applications: For forecasting wind energy [WOK17] or financial data [Ele+20], for object detection [RF18], for speech recognition [Cha+16] and speech synthesis [Oor+16], or in medicine for detecting diseases like cancer [PK20] or for developing new drugs [Ele+22]. Furthermore, the detection of anomalies [Pan+21] or manipulations [Wor20] is possible. In particular, advances in deep learning have led to many new and better solutions over the past 15 years. First, the computing power of graphics processing units (GPUs) has greatly increased, allowing the computation of ever larger deep neural network (DNN). Second, new architectures, some of them specialized, have been developed, such as AlexNet [KSH12], ResNet [He+16], or Transformers [Vas+17]. This allows modern methods to outperform earlier methods. And in some areas they can assist humans [HT17], if not outperform them altogether [Sil+17b]. In this thesis, I developed and studied CI methods to assist humans in three different areas.

In the field of finance it is important to assess companies and especially to analyze their future performance at the stock market. Many factors can be considered to make a prediction, such as the company's profit for the next quarter. Financial institutions employ well-paid analysts to make good predictions. Since publicly listed companies are required by law to publish financial information on a regular basis (quarterly in the U.S.), a great collection of historical data is already publicly available. This raises the question whether an CI method can be trained to produce similar or even better forecasts than analysts. As a result, such a model could either support a human analyst or even compete with them in some fields.

Another area in which modern CI methods can support humans is the development of drugs in medicine. In this thesis, the development of drugs against viruses is considered using severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) as an example. There exist several approaches to inhibit the spread of a virus in a human body. One of these approaches comprises to stop the virus from replicating inside a cell by preventing the process of replication. Specifically, the protease enzyme responsible for cleaving viral proteins can be blocked by a protease inhibitor. Therefore, the search for a molecule that can act as a protease inhibitor is crucial. Such a molecule must fulfill certain properties. First, it must be able to permanently bind to the protease in order to block it. In addition, the protease inhibitor must be non-toxic to the human body. Before a molecule can be used as a drug, it must be possible to synthesize it. Traditionally, the development of candidate molecules has been done by trial and error and with assistance of experts. Testing single molecules, especially *in vitro* but also in precise physical simulations, is very time-consuming. Therefore, compared to the space of over 10^{60} possible molecules with masses below 500 g/mol [Rey+10], only very few can be considered in this way. The use of CI methods can drastically reduce the computational time for individual molecules.

As a result, many more molecules can be considered to find a suitable one. This raises the question of which **CI** methods can be used for this application, and whether they yield good candidate molecules that can be used to pre-select for further steps.

The third area we look at is the visualisation of data. Deep learning typically uses very large data sets. The data is often highly dimensional and not directly interpretable by humans. To get an overview of the data, there are several dimensionality reduction methods that provide a visual representation that can be interpreted by humans. One such method is the **self-organizing map (SOM)**. It represents the given data on a two-dimensional map. Ideally, the relationships between paired data points are preserved, i.e., data points that are far apart in high-dimensional space should also be far apart on the two-dimensional map, and vice versa. For certain data, it may be of interest to not only visualize the original data, but also to consider higher-level features. In this thesis, this will be investigated using images as an example. Lines and contours of objects are examples of higher-level features within images. Such information can be found in the hidden layers of a **convolutional neural network (CNN)** trained for object recognition. This combination results in the newly developed **convolutional self-organizing map (ConvSOM)**. We want to investigate whether the consideration of higher order features leads to better results.

1.1 Thesis Structure

This document is divided into five parts. After the general introduction in Part **I**, the following three parts cover the three major topics of this thesis: Prediction in a financial context in Part **II**, molecule design for drug development in Part **III** and visualization of data using a newly developed method in Part **IV**. At the end, Part **V** concludes this thesis.

The following is a more detailed overview of the individual chapters. After this first introductory chapter, Chapter 2 gives a general introduction to **machine learning (ML)** and **artificial neural networks (ANNs)**. More specific algorithms will be presented later in the respective parts where they are used. In Chapter 3, we look at earnings prediction for US equity companies using time series data and various types of **ANNs**. Then, in Chapter 4, we motivate and introduce the basics for the molecule design part. This begins with a small introduction to organic chemistry and **evolutionary algorithms (EAs)**. Chapter 5 describes an evolutionary approach to molecule generation using **multi-objective optimization (MOO)**. In addition, the metrics used for molecule evaluation are presented. These are likewise used for the language model based procedure in the following Chapter 6. In this second approach to molecule generation, essential components of **EA** are handled by a language model. Next, a molecular dynamics analysis is performed on the most promising molecules. Chapter 7 introduces the newly developed **ConvSOM**, which is a combination of a **CNN** and a **SOM**. Dedicated quality metrics are introduced to evaluate this new approach. Finally, Chapter 8 summarizes this work and also provides an outlook on potential enhancements and future work.

1.2 Contributions

There are four main contributions to this thesis. These have been published as peer-reviewed articles. A brief overview of these publications:

- The results presented in Chapter 3 are based on a collaboration with the economist Sebastian Tideman, who was interested in the predictive power of modern deep learning methods for finance. He provided two large data sets, which are well-known in his field. With his expertise he helped in selecting the relevant parameters of the data sets I ended up using and advised me regarding the financial application domain. All the implementation and methodology of the realization was done by me: The data pre-processing, the implementation of the deep learning techniques, the quality measures and the experimental analysis.

Lars Elend, Sebastian A. Tideman, Kerstin Lopatta, and Oliver Kramer. “Earnings Prediction with Deep Learning”. In: *KI 2020: Advances in Artificial Intelligence - 43rd German Conference on AI, Bamberg, Germany, September 21-25, 2020, Proceedings*. Ed. by Ute Schmid, Franziska Klügl, and Diedrich Wolter. Vol. 12325. Lecture Notes in Computer Science. Springer, 2020, pp. 267–274. DOI: [10.1007/978-3-030-58285-2_22](https://doi.org/10.1007/978-3-030-58285-2_22)

- As of March 2020, my research group was working on the topic of molecule design for drug discovery. This was motivated by [coronavirus disease 2019 \(COVID-19\)](#), which was beginning to spread at the time, but our methods are not limited to this and could be used against any virus. Chapter 5 is based on the results of this first collaboration. The implementation, the behavior of the specific EA and the execution of the experiments were essentially shared equally between Tim Cofala and me. However, I would like to highlight the chemist Thomas Teusch, whose expertise we particularly sought for the chemical evaluation of the resulting molecules.

Tim Cofala, Lars Elend, Philip Mirbach, Jonas Prellberg, Thomas Teusch, and Oliver Kramer. “Evolutionary Multi-objective Design of SARS-CoV-2 Protease Inhibitor Candidates”. In: *Parallel Problem Solving from Nature – PPSN XVI*. ed. by Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 357–371. ISBN: 978-3-030-58115-2. DOI: [10.1007/978-3-030-58115-2_25](https://doi.org/10.1007/978-3-030-58115-2_25)

- Building on the previous work, another approach to molecule design is presented in Chapter 6. It is based on the collaboration with several physicists and was subsequently published in a journal article. I carried out the realization, implementation and experimental investigation of the [evolutionary molecule generation algorithm \(EMGA\)](#) together with Tim Cofala, and we built on preliminary work by Jonas Prellberg to implement the neural language model. The manual selection of the most promising candidates was again done by Thomas Teusch. The molecular dynamics analysis of the selected molecules – a more precise physical simulation and calculation – was essentially carried out by Luise Jacobsen.

Lars Elend, Luise Jacobsen, Tim Cofala, Jonas Prellberg, Thomas Teusch, Oliver Kramer, and Ilia A. Solov'yov. “Design of SARS-CoV-2 Main Protease Inhibitors Using Artificial

Intelligence and Molecular Dynamic Simulations”. In: *Molecules* 27.13 (13 Jan. 2022), p. 4020. ISSN: 1420-3049. DOI: [10.3390/molecules27134020](https://doi.org/10.3390/molecules27134020)

- Chapter 7 introduces the **ConvSOM** architecture I invented. The implementation and further development was conducted by me, and published as a conference paper. The chapter introduces this new architecture as well as the specially developed quality metrics for experimental analysis. In addition, another metric is presented, which was developed in the course of supervising a Master’s thesis.

Lars Elend and Oliver Kramer. “Self-Organizing Maps with Convolutional Layers”. In: *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization - Proceedings of the 13th International Workshop, WSOM+ 2019, Barcelona, Spain, June 26-28, 2019*. Ed. by Alfredo Vellido, Karina Gibert, Cecilio Angulo, and José David Martín-Guerrero. Vol. 976. *Advances in Intelligent Systems and Computing*. Springer, 2019, pp. 23–32. DOI: [10.1007/978-3-030-19642-4_3](https://doi.org/10.1007/978-3-030-19642-4_3)

The remainder of this thesis will be written in a scientific style with the use of *we* rather than *I*. At this point, we would like to mention the indexes in the appendix, which provide an overview of the parameters and acronyms used and are also accessible via hyperlinks in the digital version.

2 Computational Intelligence

Computational intelligence (CI) is a field of computer science, but it is also applied in many other areas. It is seen as a central part of *artificial intelligence (AI)* [And90]. One important subset of *CI* is *machine learning (ML)*. The idea of *ML* is to solve specific problems with algorithms that improve automatically through experience [Mit97], usually gained from data. In this chapter, we will look at the different types of learning used to describe and group *ML* algorithms. Furthermore, we will introduce *ANNs*. The specific *CI* algorithms used in this thesis are presented directly in the respective parts. Figure 2.1 shows an overview of these algorithms and how they are grouped.

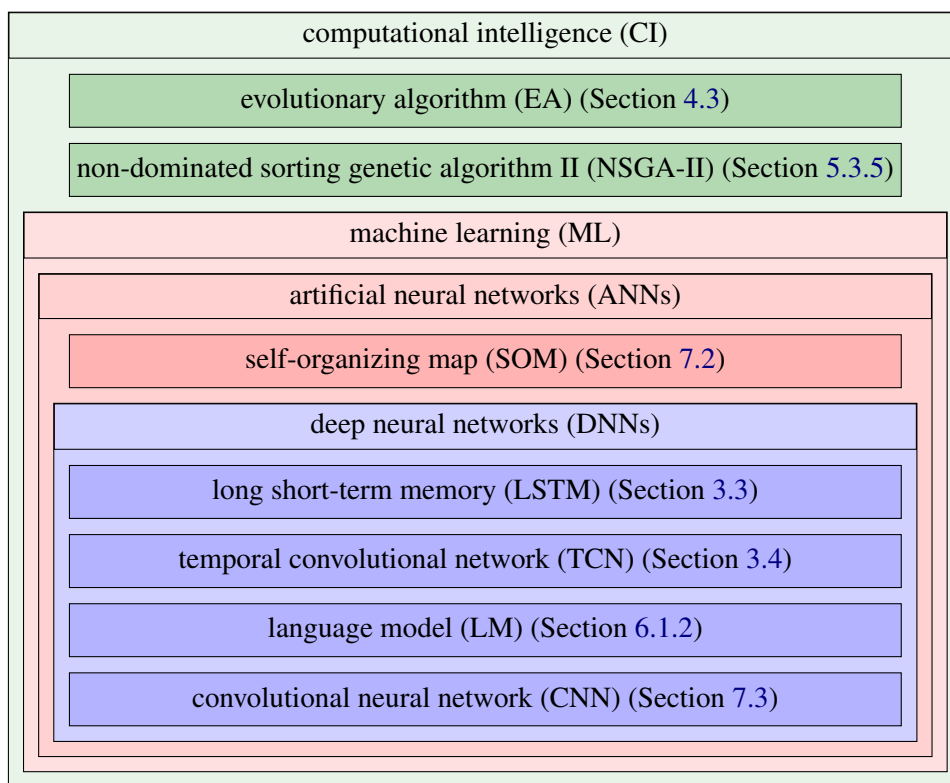


Figure 2.1: Overview of the *CI* algorithms that are used in this thesis and explained in the later parts.

2.1 Machine Learning

There are different types of learning that are used depending on the type of problem and the given information. Algorithms can be grouped by three main types of learning: supervised, unsupervised, and reinforcement learning.

2.1.1 Supervised Learning

Supervised learning uses a number N of samples \mathbf{x} (e.g., images) and their corresponding labels \mathbf{y} (e.g., objects on the image). These samples and labels can be defined by tensors¹ $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$, where $\forall i, 1 \leq i \leq N : \mathbf{y}_i \in \bar{\mathcal{Y}}$ is the label of the sample $\mathbf{x}_i \in \bar{\mathcal{X}}$. A relation² ϕ between the sample space $\bar{\mathcal{X}}$ and the label space $\bar{\mathcal{Y}}$ can be defined as:

$$(\mathbf{x}, \mathbf{y}) \in \phi = \{(\mathbf{x}_i, \mathbf{y}_i) \mid \forall i \in \mathbb{N}, 1 \leq i \leq N\} \subseteq \bar{\mathcal{X}} \times \bar{\mathcal{Y}}. \quad (2.1)$$

The ordered pair (\mathbf{x}, \mathbf{y}) will be used as representative for an arbitrary pair $(\mathbf{x}_i, \mathbf{y}_i)$. The goal of supervised learning is to find a function $f : \bar{\mathcal{X}} \rightarrow \bar{\mathcal{Y}}$, that approximates the relation ϕ .

Depending on the type of the value \mathbf{y} , problems are separated into classification and regression. With *classification* problems \mathbf{y} is a discrete value. A common problem is classification of images, i.e., the prediction of the main visible object on the image, which is than the class of the image. Therefore we need a function:

$$f : \mathbb{R}^{\bar{h} \times \bar{w} \times \bar{c}} \rightarrow \{1, \dots, C\}, \quad (2.2)$$

with image height \bar{h} , image width \bar{w} , color channels \bar{c} , and number of classes C . For standard color images with channels red (R), green (G), and blue (B) $\bar{c} = 3$, while for gray-scale images $\bar{c} = 1$. Figure 2.2 shows an example of a simple image classification problem with images from the CIFAR-10 data set.

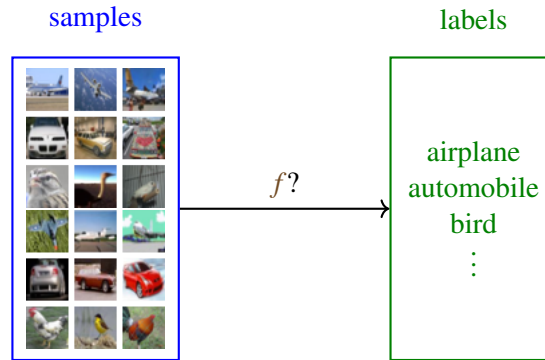


Figure 2.2: An example of an image classification problem, where the goal is to find a function $f : \bar{\mathcal{X}} \rightarrow \bar{\mathcal{Y}}$ (see Eq. (2.2)) that gives the object that is visible on a given image.

In *regression* problems \mathbf{y} is a not discrete value or vector, but a continuous one, often $\mathbf{y} \in \mathbb{R}^n$. Examples of regression problems are weather prediction (wind, temperature, barometric pressure etc.) or the prediction of financial properties like earnings per share of companies, as presented in Chapter 3.

¹While a vector is 1-dimensional (e.g., \mathbb{R}^n) and a matrix is 2-dimensional (e.g., $\mathbb{R}^{n \times m}$), a *tensor* is used as a generic term and can have any dimension. A tensor is especially used for dimensions ≥ 3 (e.g., $\mathbb{R}^{n \times m \times k}$). Since we also want to cover the case where the samples consist of 2-dimensional data structures, such as images, we use a tensor.

² ϕ is not a function, since it is usually not left total, as not all elements of the sample space are given as samples. Besides, it is possible that it is not functional (right-unique), e.g., if some real-world measurements are used to form the data set and two samples with the same measurements have different labels.

To train a ML model, the data set, containing \mathbf{X} and \mathbf{Y} , is first split into training and test data. The training data set is used during the learning of the model, while the test data set is used afterwards as an independent evaluation of the model, i.e., testing the quality of the model with before unseen data. It is common to use also a validation data set. This is similar to the test data set also independent of the training data and can be used for tuning hyperparameters of the model (e.g., number of neurons or layers) or for early stopping. This subdivision of the data is useful to detect and avoid *overfitting*. What is meant by this is overfitting the model to the specific training data and results in the model not generalizing and performing poorly for unknown data. Sometimes data sets are already published with a specific division into training and test data (see Table A.1). In this case, the validation set is taken from the original training set. Figure 2.3 shows an overview of the data sets and the training steps. At first $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ is used for training of the model. Then the hyperparameters of the model can be tuned using $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$. Finally, the model is tested using $(\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}})$.

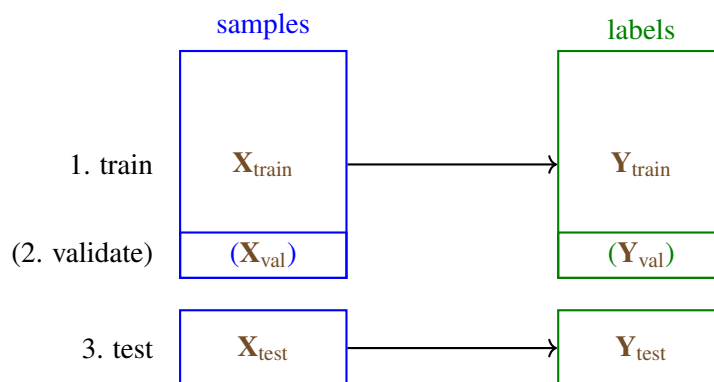


Figure 2.3: Overview of training, validation, and test sets. The samples \mathbf{X} and labels \mathbf{Y} are at first both partitioned into subsets: train and test. The train set can be partitioned again, to create a validation set.

If the data sets are not dependent on each other, the partial data sets are usually initially selected once at random. However, it is also possible not to use a fixed allocation by using cross-validation. In this case, the data set is divided into a certain number of parts, one of which is always used for validation, while the rest can be used for training (see Fig. 2.4). With cross-validation, a more accurate assessment of the model quality is possible with respect to the training data compared to the validation data due to the different iterations.

In case of dependent data, e.g., time series data as it is used in Chapter 3, the dependencies must be taken into account, when selecting validation and test sets. That means, if the goal is to predict some future values or trends, these future data points should not be used within the training set. In that case, the model would have information about the future that it cannot have in reality.

2.1.2 Unsupervised Learning

In *unsupervised learning* the given samples \mathbf{x} are without labels. The goal is to find hidden structures or *patterns*. This can be used for example for clustering, i.e., building groups of similar samples, with respect to some automatically identified pattern. Some well-known algorithms for clustering are: k-means, mixture model, density-based spatial clustering of applications with noise (DBSCAN)

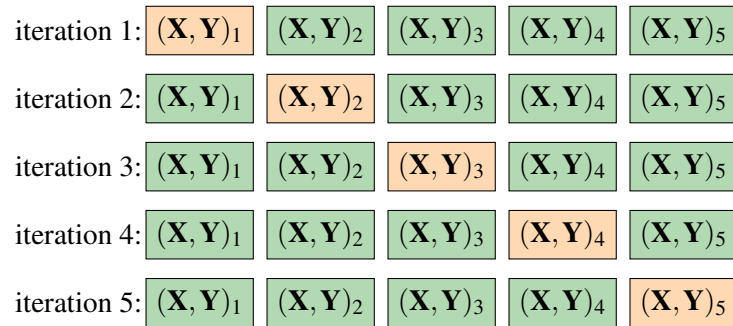


Figure 2.4: Example partition of a data set using cross-validation. In this example the data set (\mathbf{X}, \mathbf{Y}) is split into 5 segments, of which one is used for validation (orange), while the remaining are used for training (green). Thus, there are different constellations for 5 iterations.

and hierarchical clustering. In the field of neural networks exist algorithms like: self-organizing map (SOM) (see Section 7.2), neural gas (NG), deep belief network (DBN), autoencoders (AE), generative adversarial network (GAN). If only a few samples are labeled but most are unlabeled we speak of semi-supervised learning.

2.1.3 Reinforcement Learning

In *reinforcement learning*, a so-called agent is supposed to learn a meaningful behavior or strategy in a given world or environment. The agent takes actions based on its observation of the world and its current strategy, which in turn can change the state of the environment. Then the strategy is iteratively adjusted to maximize the reward. The general structure is shown in Fig. 2.5.

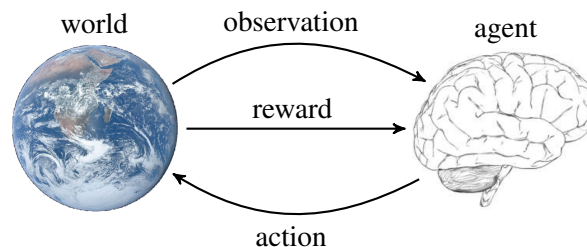


Figure 2.5: General setup of reinforcement learning. An agent observes the world and takes actions. In addition, the agent receives a reward on the basis of which it can adapt – i.e., learn – its behavior.

In reinforcement learning, there is no controller to monitor the training process, but only the reward signal. Thus, the reward function significantly influences the learned behavior. It can be positive or negative. The presented scheme takes place iteratively within a period of time. It is important to note that delayed feedback (reward) can complicate the learning process. For example, a move at the beginning of a game could be decisive for victory or defeat. If a simple reward function is used, the reward would not be received until the end of the game. In addition, the time course is important, since we are dealing with sequential and not *independent and identically distributed* (i.i.d.) data. That means a certain observation can have a different meaning at different times, depending on what happened before. This is due to the fact that the agent's actions affect the world and thus future observations.

Because of the good environment description, board or computer games of different complexity are often used for reinforcement learning, like Chess [Sil+17a] and Go [Sil+16; Sil+17b] or Atari [CEK20] and Star Craft 2 [Vin+17; Vin+19]. In many games it is now possible to reach a superhuman playing strength. In fact, RL agents often find entirely new strategies in the process.

2.2 Artificial Neural Networks

An *artificial neural network (ANN)* is an algorithm inspired by certain aspects of the natural behaviour of a mammalian brain. The individual neuron forms its basic building block. The neurons can be linked depending on the topology chosen, and are usually arranged in several successive layers. ANNs with many layers are called *deep neural networks (DNNs)*. Although there is no general definition for this classification based on the number of layers, a minimum of 3 layers can be assumed as a guide. A DNN comes closer to the brain, which can have many layers, especially in the visual system [Ben09]. Due to the greatly increased computing power of GPUs and its good parallelization, the intensive use of DNNs has been enabled over the last 15 years. In this chapter, however, we first focus on the basic structure and functioning of ANNs, using a simple feed forward network. The specific ANNs used for individual applications within this thesis are presented directly in the respective parts, as mentioned at the beginning of Chapter 2.

2.2.1 Artificial Neuron

An artificial neuron n_i produces an output y_i which is calculated on the basis of its inputs x_1, \dots, x_n . The sum of the inputs weighted by $w_{i,j}$ is first calculated. The bias b_i is added. Then the result is put into an activation function φ . An overview of some commonly used activation functions is given in the following section. Figure 2.6 visualises the structure of an artificial neuron. The generated output of the neuron can be described as:

$$y_i = \varphi \left(\sum_{j=1}^n w_{i,j} \cdot x_j + b_i \right). \quad (2.3)$$

Instead of the bias, a threshold value is sometimes used which is equal to the negative bias.

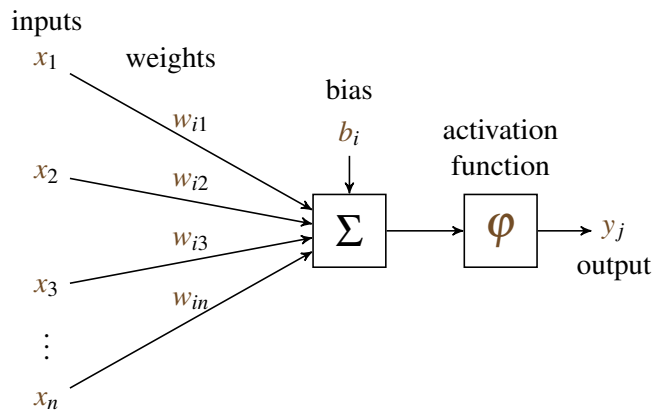


Figure 2.6: Structure of an artificial neuron n_i . The sum of the weighted inputs and the bias is given to an activation function that calculates the output.

2.2.2 Activation Functions

A suitable function can be selected for activation depending on the desired behaviour. Some typical activation functions are shown in Fig. 2.7. With the step or threshold function (s. Fig. 2.7a), a binary

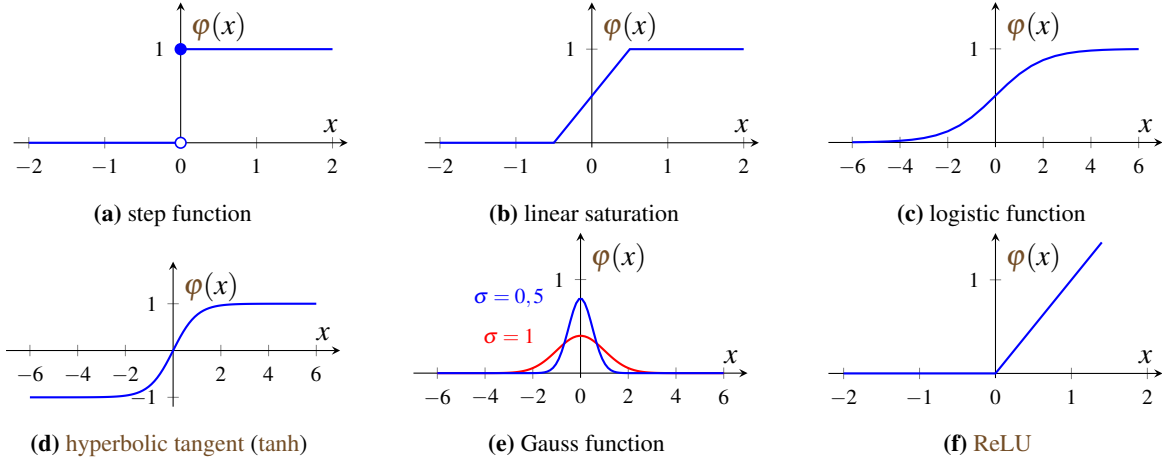


Figure 2.7: Activation functions

output is generated, which means there is a jump discontinuity at the threshold. This leads to a behaviour that is the closest to the natural neuron, in which according to the so-called *all-or-nothing law* a reaction is triggered if the threshold potential is exceeded [STL00, p. 24]. Continuous output can be achieved using different functions, one can choose from among a linear (s. Fig. 2.7b), a sigmoid transition, as in the logistic function (s. Fig. 2.7c) or the *hyperbolic tangent (tanh)* (s. Fig. 2.7d), a Gaussian function (s. Fig. 2.7e), and a *rectified linear unit (ReLU)* (s. Fig. 2.7f). ReLU is particularly suitable as an activation function for DNNs [GBB11]. The functions shown in Fig. 2.7 are defined in Eqs. (2.4) to (2.9):

$$\text{step}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{else} \end{cases} \quad (2.4)$$

$$\text{linear_saturation}(x) = \begin{cases} 0 & \text{if } x < -0.5 \\ 1 & \text{if } x > 0.5 \\ x + 0.5 & \text{else} \end{cases} \quad (2.5)$$

$$\text{logistic}(x) = \frac{1}{1 + e^{-x}} \quad (2.6)$$

$$\text{tanh}(x) = \tanh(x) \quad (2.7)$$

$$\text{gauss}(x, \sigma_G) = \frac{1}{\sigma_G \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x}{\sigma_G}\right)^2} \quad (2.8)$$

$$\text{ReLU}(x) = \max\{0, x\} \quad (2.9)$$

When evaluating the Gaussian function, the standard deviation σ_G can also be set. The mean value μ_G is not needed here, as this can already be expressed via the bias. By composing non-linear functions, functions of greater complexity can be created [McC00].

2.2.3 Feed Forward Network

While individual neurons can only describe very simple functions, their interconnection within an ANN also enables much more complex functions, depending on its size. This is illustrated by an example of a two-dimensional classification problem in Fig. 2.8. The individual neurons can initially only make a linear subdivision. By combining them, a more complex 2-dimensional area can also be described. Multilayer feedforward networks can be considered universal approximators, provided they have a sufficient number of hidden neurons [HSW89].

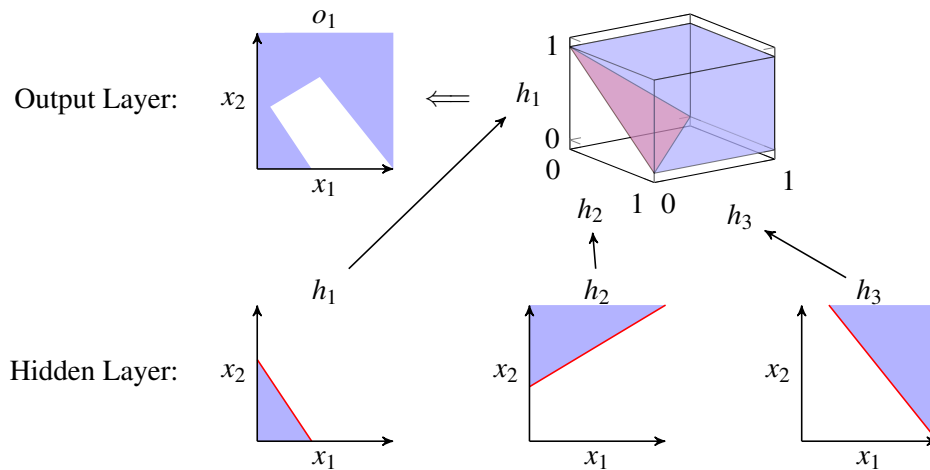


Figure 2.8: Combination of several neurons to solve a classification problem with two dimensions. Each individual neuron can describe a hyperplane (red) in the space spanned by its inputs on the basis of its weights and its bias. In this case, a step function is considered and therefore this hyperplane forms the boundary between the output values 0 (white) and 1 (blue). By connecting several neurons in series, it is also possible to classify more complex problems. The result in relation to the inputs x_1 and x_2 is shown in the top left for the output neuron.

The structure of a neural network can be represented as a directed graph. Neurons are represented as nodes and the connections between neurons are represented as directed edges. The nodes can be divided into three types or layers: input, hidden and output layer. The input layer only consists of the input data of the ANN. The output layer consists of neurons whose output is the result of the ANN. The other layers in between are called hidden layers. Typically, such a representation is built so the input and output layers are opposite each other and the general flow of information is in one direction. Figure 2.9 shows an example of the structure of two ANNs. Since the edges here all run from left to right and there are no cycles in the graph, it is also called a *feed forward network*. Similarly, the previous Fig. 2.8 is a feed forward network with 2 input neurons, 3 hidden neurons, and one output neuron. There are also networks that contain feedbacks (i.e., cycles in the graph). These are called *recurrent neural networks (RNNs)*.

2.2.4 Backpropagation

To get ANNs to behave in a certain way, the internal weights and bias values can be adjusted. This weight adjustment is done automatically via *Backpropagation* (of error). The use of backpropagation

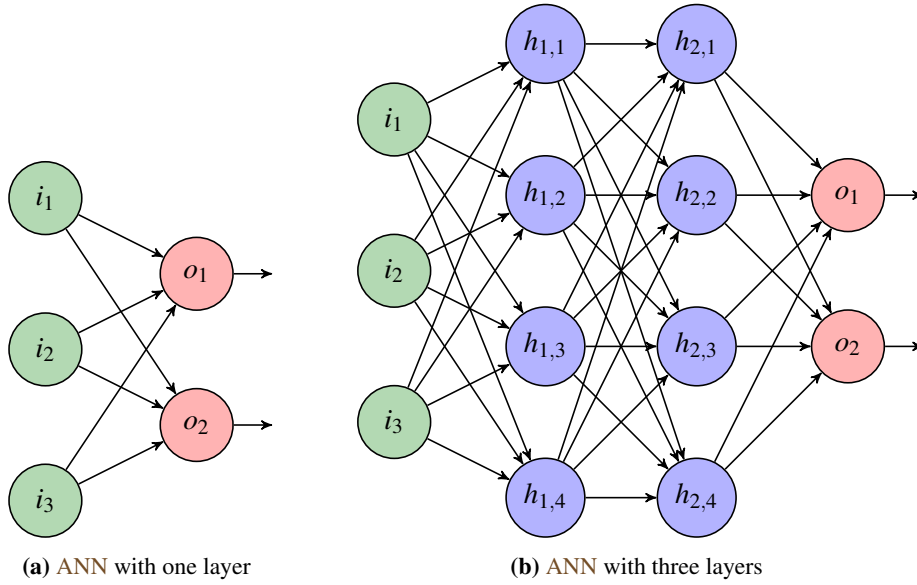


Figure 2.9: Feed forwards networks consists of one or multiple layers of neurons, that are only connected in the direction from input to output. The input layer (green) just contains the actual input values and is therefore not counted as real layer. After that, any number of hidden layers (blue) can follow. The last layer is the output layer (red) and returns the result of the ANN.

for ANNs was first mentioned by Werbos in 1974 [Wer74]. In 1985 it was used to train ANNs with hidden neurons and different connectivity [RHW85].

In backpropagation, the deviation of the computed result $\hat{\mathbf{y}}$ from the expected result \mathbf{y} is calculated using the *loss function* or *error function* for each output neuron j . The squared error can be used for this:

$$\hat{e} = (\mathbf{y}_j - \hat{\mathbf{y}}_j)^2. \quad (2.10)$$

Subsequently, the *gradient* is calculated. The gradient is a vector pointing in the direction of the largest slope of the function value. The result $\hat{\mathbf{y}}_j$ is calculated using the prior neurons and their weights, and the error is regressed based on these weights in the network. For the regression, it is necessary that the activation function φ is derivable, since for each weight $w_{i,j}$ a partial derivative is calculated using the chain rule. The weights are adjusted for the next step to the extent that they contributed to the erroneous result and depending on the learning rate α [Hag+14; Wer90]:

$$\Delta w_{ij} = -\alpha \frac{\partial \hat{e}}{\partial w_{ij}}. \quad (2.11)$$

2.3 Conclusion

This chapter gave an introduction to ML and the different learning categories supervised, unsupervised and reinforcement learning. Furthermore, the basics of ANNs were discussed, on which some of the algorithms presented later are based. For example, in the following chapter, the networks *long short-term memory (LSTM)* and *temporal convolutional network (TCN)*, which are specialized for time series data, are presented and used for supervised learning in the context of financial data.

Part II

Prediction

3 Earnings Prediction

Investors rely primarily on earnings predictions when making investment decisions, such as whether to buy, hold or sell shares of a company. In addition to their own projections, they base their forecasts to a large extent on the earnings forecasts of financial analysts. As a result, forecasting earnings is one of the main tasks of financial analysts working at large financial institutions, such as brokerage firms, and they spend significant time and resources to make accurate forecasts. However, prediction is a difficult endeavor because numerous factors affect the prediction performance. In this chapter, we predict the quarterly *earnings per share (EPS)* of publicly listed U.S. firms using state-of-the-art deep neural network techniques based on time series data of the companies.

This chapter is structured as follows. In Section 3.1, the related work on prediction of financial data is presented. The base time series model is introduced in Section 3.2. Two types of ANNs suitable for handling time series were used: LSTMs (Section 3.3) and TCNs (Section 3.4). We describe the data preprocessing process in Section 3.5. After that the quality measures are defined in Section 3.6. Section 3.7 presents the experimental analysis, and Section 3.8 draws a conclusion.

Parts of this chapter are based on the following published paper:

Lars Elend, Sebastian A. Tideman, Kerstin Lopatta, and Oliver Kramer. “Earnings Prediction with Deep Learning”. In: *KI 2020: Advances in Artificial Intelligence - 43rd German Conference on AI, Bamberg, Germany, September 21-25, 2020, Proceedings*. Ed. by Ute Schmid, Franziska Klügl, and Diedrich Wolter. Vol. 12325. Lecture Notes in Computer Science. Springer, 2020, pp. 267–274. DOI: [10.1007/978-3-030-58285-2_22](https://doi.org/10.1007/978-3-030-58285-2_22)

3.1 Related Work

Analyst forecasts are frequently utilized as a benchmark to assess the accuracy of earnings predictions derived from models. However, the implementation of recent regulations governing the working conditions of financial analysts, such as restricted private access to management, has resulted in a decline in analyst coverage [AZ11]. This reduction in coverage can potentially be addressed by employing automated earnings prediction models supported by artificial intelligence. Although there has been considerable research on predicting stock market prices and returns using neural networks, which encompass various factors at the company, industry, and country levels [dSD17], it remains unclear whether artificial intelligence can effectively generate meaningful earnings forecasts as a direct measure of business success.

Some studies indicate that machine learning techniques can predict instances of fraud, such as illegal earnings manipulation [Bao+20]. Bao et al. discovered that ensemble learning, utilizing raw accounting numbers, possesses predictive power for identifying future cases of fraud. Their approach

outperformed logistic regression models based on commonly used financial ratios in previous research [Dec+11], as well as a support-vector-machine model [Cec+10], in which a financial kernel maps raw accounting numbers into a set of financial ratios. However, predicting restatements is comparatively less challenging as it involves a binary decision tree (future restatement vs. no future restatement). Forecasting future earnings is more complex due to the possibility of various values and the need to consider information from multiple sources, such as financial statements and stock market data.

To the best of our knowledge, as of early 2020, no study has yet utilized artificial intelligence to predict future earnings. The study by Ball and Ghysels [BG17] comes closest to this research. They employed a mixed data sampling regression method (without neural networks) to predict future earnings and found that their predictions outperformed analysts' forecasts in specific cases, particularly when the company size was smaller and there was high forecast dispersion among analysts.

3.2 Time Series Prediction

In real world problems time has often a direct or indirect meaning. This can be a state depending on the history of events that happened before or on its own historic states. Time series data is gathered by observing some property over time, e.g., by using any sensor. Some examples are: weather forecast, physical movement prediction and game move prediction. If we consider a jumping ball and measure its spacial location over time, we can calculate its speed, acceleration and if we have some basic knowledge about its environment, we could also calculate some of its properties like weight, density and air resistance. By calculating this information we can even predict its future behavior and thus its location. Therefore, the historical positions of the ball are only indirectly relevant to the prediction, as they can be used to calculate other information that is not directly measured, and perhaps not even easily measured.

The more complex a system is, the more parameters are important for making a prediction. If we can not measure all necessary parameters and also can't calculate them, the historical data can also be used to learn some behavior in a function approximator, such as a neural network. For example, the movement of clouds could be analyzed as part of a weather forecast. Relevant geographical information such as mountains or bodies of water would automatically be taken into account indirectly, provided that enough training data is available.

Now we define the data series prediction more formally as a foundation for the later application. Its goal is, to find a function ϕ that yields a future value \mathbf{y} based on the data of the past β time steps $\mathbf{x} = (u_{t-\beta+1}, \dots, u_t)$. In the simple case the time steps have a fixed size, e.g., 15 minutes, 1 hour or 1 day. In this case, β is also called *window size*, as it moves a window of fixed size over the time series. If a large time span and also a near past is relevant, it would be also possible to use time dependent time resolution, i.e., use a high resolution for the near past and a low one for the distant past, e.g., by using some exponential function. This was done, for example, with the *exponential piecewise approximate aggregation (exPAA)* introduced in [Oeh18]. The time series prediction is illustrated in Fig. 3.1.

An overview on this topic is given by Montgomery, Jennings, and Kulahci [MJK15] and Yaffee and McGee [YM00]. In the following we will introduce two widely used kinds of artificial neural networks, which are successfully used for time series prediction [WOK17].

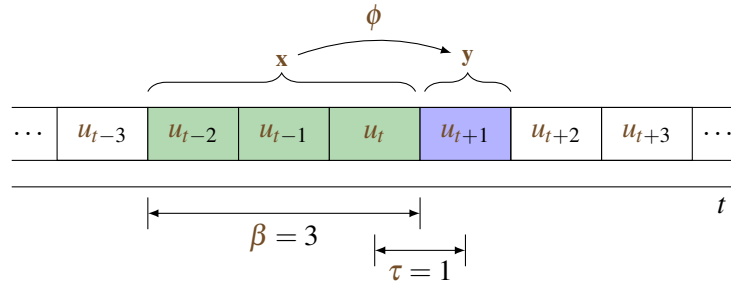


Figure 3.1: Illustration of time series model for prediction of the data of the next time step u_{t+1} . We seek a mapping ϕ from sample x of the past to label y for the future $t = t + \tau$. The window size β describes the time span of considered steps and τ is the prediction horizon.

3.3 Long Short-Term Memory

The *long short-term memory (LSTM)* [HS97] is a special kind of *recurrent neural network (RNN)*, i.e., a network with backward edges that allow a kind of memory. An LSTM cell internally contains three gates: forget, input and output gate. It has an internal state that is adapted by these three gates. Some of the old knowledge may be forgotten, some new inputs should be taken into account and be memorized, and some information should be extracted from the internal state as an output. Figure 3.2 shows the exact internal structure of a LSTM cell.

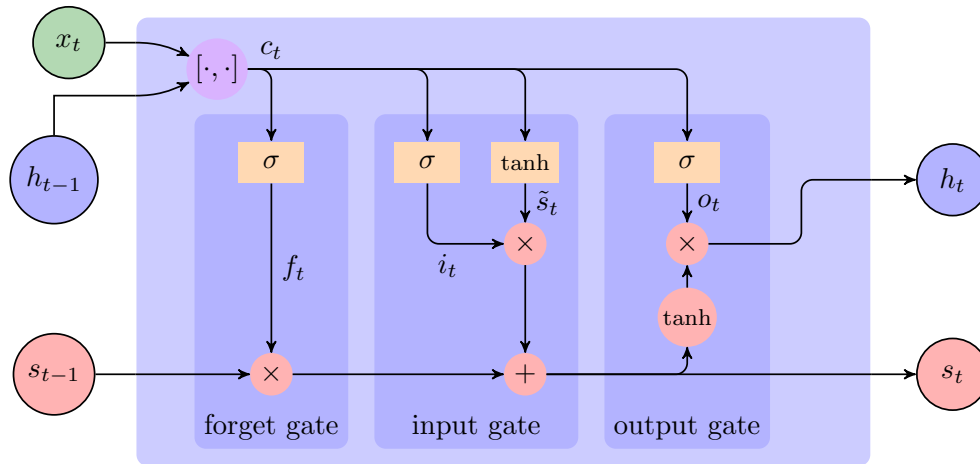


Figure 3.2: The LSTM cell has two internal states h and s , which are saved for the next time step. At first the input x_t is concatenated with h_{t-1} . The result goes into the three gates: forget, input and output. The orange boxes represent ANN layers with the given activation function. The red circles represent element wise operations. After an arbitrary number of time steps h_t will be used as output.

The same cell is used iteratively at each time step with the current input x_t . Thus, the internal weights of the ANN layers are the same for all time steps. Therefore, the LSTM needs fewer weights compared to feedforward networks. Due to the internal memory it is well suited for temporal data, e.g., time series prediction. Therefore, a LSTM can for example be used for Wind Power Prediction [WOK17], but also for other tasks, where knowledge about the past is important, like text generation [Gra13], speech recognition [GJ14], or video to text [Ven+15].

3.4 Temporal Convolutional Network

The *temporal convolutional network (TCN)* [BKK18] is a special kind of CNN [LeC+89]. While CNNs are mainly used for classification tasks in images, text or language, TCNs can be used for time series data. The architecture of the standard CNNs has been adapted for this purpose by using two concepts: causal convolutions, dilated convolutions. Casual convolutions means that for any output at time step t only input elements from a time up to that point ($\leq t$) are used. With dilated convolutions it is possible to exponentially increase the receptive field, i.e., the range of data considered. The dilation d is the distance between two input data elements which are used for a convolution. Both of these concepts are visualized in Fig. 3.3. The dilated causal convolution is the main element of the TCN, but it contains also of some other layers (WeightNorm, ReLU, Dropout) and concepts, which can be read in the original paper [BKK18].

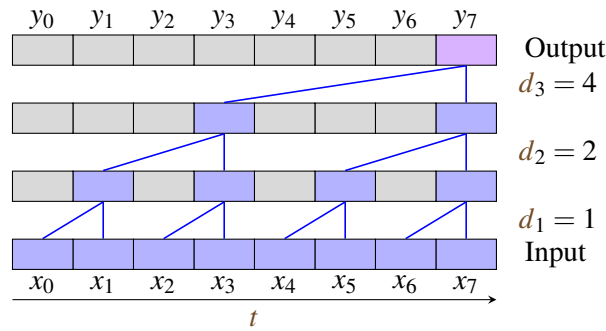


Figure 3.3: A dilated causal convolutional network with $v = 3$ layers. The kernel size $k = 2$ indicates how many inputs each neuron has (see blue lines leading into a neuron). The dilation d gives the (temporal) distance between each 2 inputs a neuron has. In this example, in the second layer $d_2 = 2$, i.e., the inputs have a distance of 2. Highlighted in blue here are all neurons and connections that affect the output y_7 (purple).

The length L of data considered by the TCN can be calculated by

$$L = 1 + s \cdot (k - 1) \cdot 2 \cdot \sum_{i=1}^v d_i, \quad (3.1)$$

with stack size s , kernel size k , number of layers v and dilation factor d_i in layer i . The stack size indicates how often the block consisting of dilated causal convolutional layer and the other layers is repeated. As in the paper by Bai, Kolter, and Koltun [BKK18] we increase d exponentially with depth of the network, i.e., $d_i = 2^{i-1}$. Therefore, we select v in a way that the given data length L' is covered:

$$v = \left\lceil \log_2 \left(\frac{L' - 1}{s \cdot (k - 1) \cdot 2} + 1 \right) \right\rceil. \quad (3.2)$$

TCNs can be used in various applications with a time context, e.g., for weather forecasting [Hew+20], for classification of satellite image time series [PWP19] or for lip-reading [KTF20].

3.5 Data Preprocessing

As input data, we use accounting data from quarterly data (COMPUTSTAT QUARTERLY) as well as daily stock market returns (DAILY SHARES). Both data sets come from *Center for Research in Security Prices (CRSP)* [CRS], whose primary mission is to maintain and expand a database that includes all securities listed on the U.S. exchanges: New York Stock Exchange (NYSE), American Stock Exchange (AMEX), and National Association of Securities Dealers Automated Quotation (NASDAQ) [Eis18]. COMPUTSTAT QUARTERLY contains a lot of data based on the quarterly reports of the companies. DAILY SHARES contains entries with daily information (for stock exchange days).

At first both data sets COMPUTSTAT QUARTERLY and DAILY SHARES are reduced to the most important parameters per time step and firm. A description of the used parameters can be found in Tables B.1 and B.2 in Appendix B.1. The outliers and gaps in the data are then handled. Since there are some very low forecast values from analysts, these are ignored if they are smaller than the minimum EPS value (-24.57). The data is “normalized”, as it consists of very different number ranges, which would be difficult as ANN input. All features \hat{z}_i that are listed after total assets atq in Table B.1 are divided by the atq to allow a relative comparison between companies of different sizes:

$$z_i = \frac{\hat{z}_i}{\max\{1, \text{atq}\}}. \quad (3.3)$$

Afterwards atq is scaled logarithmically. Values lower than 1 are ignored again:

$$\text{atq}' = \log(\max\{1, \text{atq}\}). \quad (3.4)$$

Finally, all values z_i (including atq) are studentized:

$$z'_{i,j} = \frac{z_{i,j} - \bar{z}_i}{\sqrt{\frac{1}{n} \sum_k (z_{i,k} - \bar{z}_i)^2}}, \text{ where } \bar{z}_i = \frac{1}{n} \sum_k z_{i,k}. \quad (3.5)$$

Outliers of the EPS values are removed by using the 1 % percentile as minimum and the 99 % percentile as maximum, as some of them can be considered false data. Afterwards the company samples for a given window size, i.e., a number of quarters, are build. Linear interpolation is used as imputation method to fill in missing values in the sample data. However, the label value EPS must exist for the prediction time and the time step before it (for the persistent model). It is also checked if whole time steps (quarter) are missing in between. Typically, two quarterly reports should be approximately 90 days apart. Therefore, the sample is discarded, if there is a gap of more than 100 days between two time steps. The corresponding daily stock data DAILY SHARES are stored for the selected samples. The data is also studentized. Missing values are filled with zeros. In addition to the parameters already specified, the day of the week is one-hot encoded. The samples thus created are used by the ANN predictor, whose settings will be described later under Section 3.7. An overview of the used workflow is given in Fig. B.1 in Appendix B.2.

3.6 Quality Measures

To evaluate the experimental results we use different forecast models. As a simple baseline we use the persistent model, which assumes that the future value will be the same as the last known value. As time series tend to strongly correlate between successive data points, this is a good starting point. In addition, we make a comparison with the analysts' forecasts.

We use different metrics to make the comparison. At first we calculate the *mean squared error (MSE)* by comparing the predicted with the actual values:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_{\text{true}} - y_{\text{pred}})^2. \quad (3.6)$$

This is done for all forecast models. The *MSE* punishes larger deviations more than smaller ones. Errors like *mean absolute percentage error (MAPE)* can't be used because they don't work when the true values are zero or close to zero, since they will divide by the true value.

The error value alone is not very meaningful, since it always depends on the difficulty of the given data, which can vary greatly over time and between different companies. Therefore, we use quality measures which evaluate the results relative to the other prediction methods, persistent model and analyst forecast.

Therefore, we use the *skill score (SS)*, which is able to compare metrics of two different models [Roe98]. It is defined as:

$$\text{SS} = \frac{\text{metric}_x - \text{metric}_{\text{base}}}{\text{metric}_{\text{opt}} - \text{metric}_{\text{base}}}, \quad (3.7)$$

where metric_x is the metric under consideration, $\text{metric}_{\text{base}}$ is the comparison metric (here: persistent model or analyst forecast), and $\text{metric}_{\text{opt}}$ would be the optimal metric. Since the optimum of the error metric *MSE* is zero, this can be simplified to:

$$\text{SS} = 1 - \frac{\text{metric}_x}{\text{metric}_{\text{base}}}. \quad (3.8)$$

SS is in the range $(-\infty, 1]$, where values less than 0 mean that the model under consideration is worse than the reference model, while values greater than 0 mean that it is better [Roe98]. *SS* is used to compare our models m (*ANN*, *LSTM*, or *TCN*) with the persistent model¹ pa and the analyst prediction a .

3.7 Experimental Analysis

During training, the last 10 % of the training set is used for validation only. The test set is in the time period after the training, so it is independent and has no unfair knowledge (Fig. 3.4). Otherwise overfitting could occur, i.e., the model performs well for given data, but generally not for new, unknown data. Similarly, we use two different data sets A and B, where test set of B is in the future of A. While A is used to find a good neural network architecture, B is used to evaluate the selected best architecture.

¹For the comparison with the persistent model only those data points are used for which analyst forecasts exist.

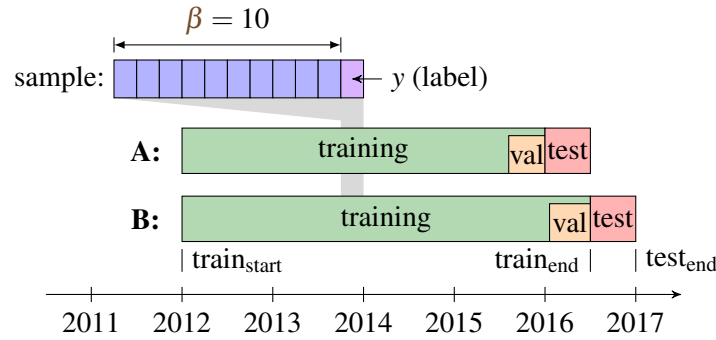


Figure 3.4: Data set horizons of training, validation, and test set. The given sample falls into both training sets as its value to be predicted y lies in their periods. Therefore, the period of used data starts β quarters before $\text{train}_{\text{start}}$. **A** displays the time horizons that were used for the following experiments to find a good architecture (training from 2012 to 2016 with last 10 % validation, half a year of test data). **B** shows the time horizons for the final experiment with the selected best architecture.

Unless otherwise specified, each model is trained with a batch size of 1024 for 1000 epochs. A dropout of 0.3 is applied after each intermediate layer and also for the recurrent edges of an LSTM layer, to prevent overfitting. For dense layers \tanh is used as activation function, except for the last layer, where a linear activation function is used. The window size of COMPUTSTAT QUARTERLY and DAILY SHARES is set to 20 each, i.e., the last 20 quarters and the last 20 trading days. These parameters were selected in the context of general recommendations from the literature and own preliminary experiments. The model is optimized with Adam [KB17] using MSE as loss. After training, the model of the epoch with the best validation error is selected and used for testing.

The experiments were each carried out 5 times. The metrics are therefore given with mean and standard deviation. In the following tables, the results of our model are shown in green if they are clearly better than those of the comparison model, i.e., lower limit of standard deviation is greater than zero, red if they are clearly worse, and black if they are very similar. For each column and group separated by lines, the best result is printed in bold whereas the worst is underlined.

For the following experiments different types (feed forward/dense, LSTM and TCN) and architectures of ANNs are used. For the description of the used architecture we introduce a short notation. In a comma separated list each layer is represented by an optional char and a number. The character gives the type of the layer: L=LSTM, T=TCN, M=merge, else dense. The merge layer is used to merge the branch of the shares-layers into the net. The number indicates the number of neurons in relation to the size of the input dimension, i.e., the number of features. For example, let us consider the meaning of the short notation $L2, M, 1$ for the input dimension 20. The input is first given in an LSTM layer with $2 \cdot 20 = 40$ neurons. Then there is a merge layer that combines the output of the previous layer with the output of the shares layers, followed by a dense layer of size $1 \cdot 20 = 20$. Finally, there is implicitly a dense output layer of size 1 with linear activation function, to get a single value as result. For TCNs the short notation is $T\langle\# \text{ filters}\rangle@ \langle\text{kernel size}\rangle$.

3.7.1 Architecture and Meta Parameters

In Table 3.1 different network architectures are compared. Therefore some representative layer structures were selected for feed forward nets, **LSTMs** and **TCNs**. In this experiment we are also testing if the usage of DAILY SHARES improves the result. If the daily shares values are not taken into account there is a dash in column sh_lay. In Table 3.1 it can be seen that there is always a shared layer architecture, which is better than ignoring DAILY SHARES. It can be seen that feed forward nets are clearly outperformed by **LSTMs** and **TCNs**. The **LSTM** achieves the best result here, but the **TCNs** are not far from this. Therefore, the best representatives (marked in bold) of these two network types will be used in the following. For simplicity, these selected architectures are referred to only as **LSTM** and **TCN** in the following tables.

Table 3.1: Overview of different network architectures with and without daily shares layers (sh_lay).

type	layers	sh_lay	SS _{MSE}			
			(m, pa)	(m, a)		
Dense	3,2,1,M,,5,.2	-	0.132±0.052	-0.433±0.086		
		3,2,1	0.240±0.040	-0.255±0.066		
		L2,L1,1	0.220±0.054	-0.288±0.088		
		L4,L2,1	0.179±0.091	-0.356±0.150		
		T32@2,2,1	0.213±0.075	-0.300±0.124		
	4,3,2,1,M,,5,.2	-	0.260±0.083	-0.221±0.138		
		3,2,1	0.322 ±0.050	-0.120 ±0.082		
		L2,L1,1	0.233±0.044	-0.266±0.072		
		L4,L2,1	0.245±0.069	-0.247±0.115		
		T32@2,2,1	0.211±0.067	-0.303±0.110		
		LSTM	L2,L1,M,0.5	-	0.399±0.036	0.007±0.060
				3,2,1	0.438±0.066	0.073±0.109
				L2,L1,1	0.450±0.015	0.091±0.025
				L4,L2,1	0.445±0.023	0.084±0.039
T32@2,2,1	0.431±0.037			0.060±0.062		
L4,L2,M,2,1	-		0.401±0.081	0.011±0.134		
	3,2,1		0.473 ±0.048	0.130 ±0.079		
	L2,L1,1		0.461±0.019	0.111±0.032		
	L4,L2,1		0.393±0.043	-0.003±0.071		
T32@2,2,1	0.374±0.063	-0.033±0.104				
TCN	T32@2,2,M,1,.4	-	0.414±0.045	0.033±0.075		
		3,2,1	0.429±0.027	0.056±0.044		
		L2,L1,1	0.405±0.053	0.017±0.087		
		L4,L2,1	0.343±0.125	-0.085±0.207		
		T32@2,2,1	0.348±0.093	-0.076±0.154		
	T32@3,2,M,1,.4	-	0.390±0.042	-0.007±0.070		
		3,2,1	0.436 ±0.021	0.069 ±0.035		
		L2,L1,1	0.324±0.135	-0.116±0.223		
		L4,L2,1	0.368±0.042	-0.043±0.069		
		T32@2,2,1	0.391±0.054	-0.006±0.089		

The two selected architectures are visualized in Fig. 3.5. The prepared information from COMPUT-STAT QUARTERLY and DAILY SHARES (see Section 3.5) are used as input (green). The dimensions

are indicated in parentheses. As the shares data are given into a dense layer (D), the temporal data with dimension 20×11 are flattened to 220. After some layers, the two inputs are joined by a merge layer. For the **TCN**, 32 filters and a kernel size of 3 were used. The last dense layer, consisting of a single neuron, outputs the predicted **EPS** value.

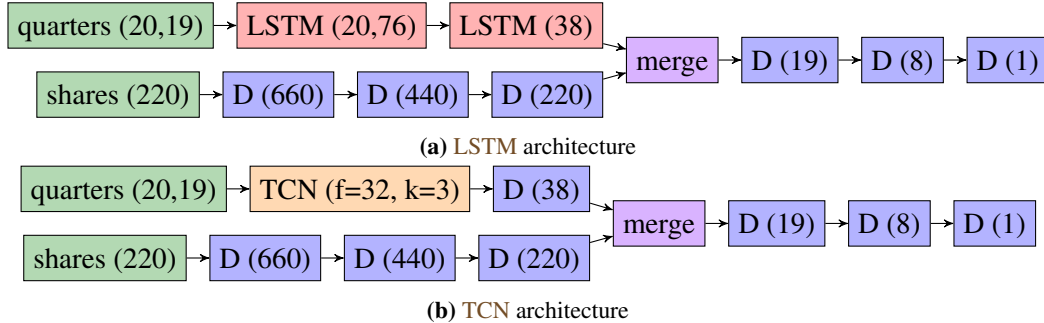


Figure 3.5: Visualization of selected **LSTM** and **TCN** architectures.

3.7.2 Financial Firms

Financial companies, such as banks and insurance companies, behave differently in many ways than companies in other industries. Therefore, we make an experiment with 3 different sets of companies: all companies (all), no financial companies (nofin), only financial companies (onlyfin), which are given in the column comp of the following tables.

Table 3.2 compares the data used depending on whether they belong to the financial sector. There the selected network architectures are used. The data sets without financial firms usually give the best results. Using only financial companies gives normally the worst result.

Table 3.2: Selected architectures (s. Fig. 3.5) and parameters for the three groups of companies: financial (onlyfin), non-financial (nofin) and all.

type	comp	SS _{MSE}	
		(<i>m, pa</i>)	(<i>m, a</i>)
LSTM	all	0.466±0.030	0.119±0.050
	nofin	0.543 ±0.013	0.146 ±0.024
	onlyfin	<u>0.378</u> ±0.025	<u>0.100</u> ±0.036
TCN	all	0.355±0.058	-0.065±0.096
	nofin	0.547 ±0.015	0.154 ±0.028
	onlyfin	<u>-0.001</u> ±0.155	<u>-0.450</u> ±0.225

3.7.3 Fama French Industries

The *Fama-French industries (FFI)* [FF93] uses the *standard industrial classification (SIC)* to define larger groups of companies. In the following we use the *Fama-French industries with 5 groups (FFI5)*. Table 3.3 gives an overview of the **FFI5** classes.

Table 3.3: Overview of the FFI5 classes [Fre22].

FFI5	Name	Description
1	Cnsmr	Consumer Durables, NonDurables, Wholesale, Retail, and Some Services (Laundries, Repair Shops)
2	Manuf	Manufacturing, Energy, and Utilities
3	HiTec	Business Equipment, Telephone and Television Transmission
4	Hlth	Healthcare, Medical Equipment, and Drugs
5	Other	Other – Mines, Constr, BldMt, Trans, Hotels, Bus Serv, Entertainment, Finance

Table 3.4 compares the results for the 5 different industry groups. Both architectures have the best result on FFI5 group 3 (HiTec) compared to the analysts, with skill scores of 0.64 and 0.56, i.e., over 50 % better. The results on group 4 (Hlth) are also quite good, with skill scores around 0.3.

Table 3.4: Results of the selected architectures and parameters for the FFI5 groups.

type	ffi5	SS _{MSE}	
		(<i>m, pa</i>)	(<i>m, a</i>)
LSTM	1	0.340±0.049	-0.338±0.100
	2	0.628 ±0.014	-0.282±0.048
	3	0.452±0.032	0.603 ±0.023
	4	-0.027±0.021	0.320±0.014
	5	0.381±0.030	0.044±0.047
	all	0.460±0.051	0.108±0.084
TCN	1	0.235±0.256	-0.550±0.519
	2	0.613 ±0.069	-0.333±0.237
	3	0.436±0.018	0.591 ±0.013
	4	0.039±0.022	0.364±0.015
	5	0.264±0.086	-0.136±0.133
	all	0.384±0.057	-0.018±0.094

3.7.4 Test of Best Model

For the final test the best model is selected and used for data set B, i.e., data with an independent test set (cf. Fig. 3.4). Table 3.5 shows the results of the best configurations of Section 3.7.2. The results for the non-financial companies are similarly to before: We get a MSE that is 12 % to 13 % better than the analysts. The predictions for all companies are now only slightly better, but worse than on data set A. In Table 3.6 the same is done for the best configurations of Section 3.7.3. The results are similarly to before. For FFI5 group 3 we get MSE above 58 % of the analysts.

3.8 Conclusion

The results of our experimental analysis indicate that LSTM networks and TCNs are robust models for predicting earnings. Our prediction models rely on quarterly accounting data, encompassing variables

Table 3.5: Results on the data set B of selected architectures and parameters for the previous best settings of non-financial companies.

type	comp	SS _{MSE}	
		(m, pa)	(m, a)
LSTM	nofin	0.300 ±0.012	0.122 ±0.015
TCN	nofin	0.308 ±0.014	0.132 ±0.018

Table 3.6: Results on the data set B of selected architectures and parameters for the previous best settings of companies of FFI5 group 3 (HiTec).

type	ffi5	SS _{MSE}	
		(m, pa)	(m, a)
LSTM	3	0.387 ±0.014	0.632 ±0.009
TCN	3	0.426 ±0.020	0.655 ±0.012

such as cost of goods sold, total assets, as well as stock market price and return data. Notably, these widely accessible time series data outperform the persistent model by a significant margin. Among the evaluated models using the same set of variables, the LSTM networks display slightly superior performance.

To enhance our financial predictions, we plan to extend our experimental analysis to encompass additional data sets and incorporate further domain knowledge. These findings hold importance for both broker firms and investors. Broker firms may consider developing LSTM networks and TCNs to complement their analysts' forecasts. Meanwhile, investors can leverage artificial intelligence to build their own forecast models, particularly when financial analysts' forecasts are unavailable. This has become an increasingly urgent concern due to the regulatory-induced drop in analyst coverage.

Part III

Molecule Design

4 Foundations of Molecule Design

This first chapter of the *molecule design* part starts with the motivation for this topic in Section 4.1. Then, a general introduction to the relevant parts of organic chemistry, with special emphasis on different representations of molecules, is given in Section 4.2. Section 4.3 introduces genetic algorithms in general, which serve as an important component in the following approaches. In the two subsequent Chapters 5 and 6, specific procedures for molecule generation are presented.

4.1 Motivation

The development of drugs that help the human body defend itself against viral diseases is of great interest in the medical and pharmaceutical fields. To prevent the spread of the virus in the body, it is crucial to interrupt the replication of the virus. The normal procedure of a virus is to enter a cell and reproduce until the cell bursts, and then continue to spread. One approach to prevent the virus replication is to disrupt the reproductive process within the cell. An important step of this process is protein cleavage, which is performed by the virus protease enzyme (see Figs. 4.1a and 4.1b). This protein cleavage can be prevented with a suitable *protease inhibitor*. The protease inhibitor works by docking to a specific site on the protease enzyme, preventing it from functioning (see Fig. 4.1c).

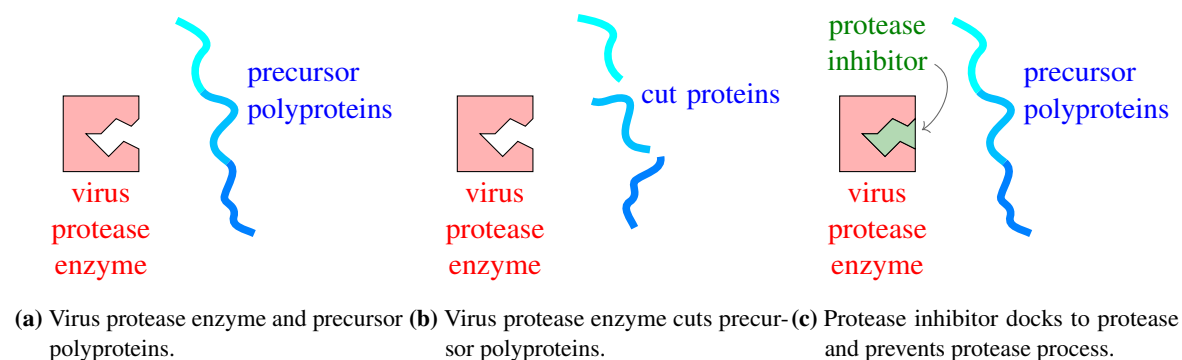


Figure 4.1: Protease Process and Protease Inhibition. The initial situation is shown in (a): The viral protease enzyme is in the vicinity of a precursor polyprotein that it intends to cleave. (b) depicts the result of the protease process: The polyproteins are cut in certain places. In (c) a protease inhibitor has docked to the protease enzyme and prevents it from cleaving.

Finding and evaluating molecules that could be suitable protease inhibitors falls into the areas of organic chemistry and physics. In addition to the binding affinity between the inhibitor and the protease, other properties are also important (e.g., *Can the molecule be synthesized?* or *Is the molecule non-toxic?*). These and other properties are considered in the form of metrics, which are introduced in more detail in Section 5.2.

One challenge in finding suitable molecules is, that the search space for potential drug molecules is vast: There are at least 10^{60} molecules with masses below 500 g/mol in the universe [Rey+10]. Consequently, an exhaustive search of the entire space encompassing all potentially potent drug molecules is simply not feasible. AI methods enable targeted navigation through the vast search space. For example, EAs presented in Section 4.3 can be used to select molecules based on certain evaluation criteria. Such criteria can be calculated precisely with physically exact simulations or even *in vitro* experiments. However, those approaches take days and weeks for a single molecule. Therefore, we use functions or simulations, that estimate the considered metrics several orders of magnitude faster, i.e., in few minutes per molecule.

Although the methods presented in the Chapters 5 and 6 could be applied generally to any arbitrary ligand-receptor complex, as an example, due to its actuality, we will deal specifically with the ribonucleic acid (RNA) based *severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2)*, which causes the *coronavirus disease 2019 (COVID-19)*. Here, we are trying to find a suitable inhibitor for the *main protease (M^{pro})*, also known as *3C-like protease (3CL^{pro})*, of SARS-CoV-2. M^{pro} is responsible for cleaving the viral polyprotein and is thus vital for the SARS-CoV-2 life cycle [Pil+16]. At the same time, because M^{pro} has little similarity to related human homologies, it is a potential drug target for the treatment of coronaviruses [Jin+20; Pil+16; Pan+20]. Otherwise, a ligand could bind to a molecule that belongs to the body instead of the viral protease, which could cause problems. Furthermore, M^{pro} is particularly promising because it is conserved across different variants within the Coronaviridae [Ana+03]. This property also makes M^{pro} an interesting drug target for mutations of the virus, since any alteration in the function of this protein could be lethal to the virus [Str+20].

4.2 Organic Chemistry

The classification into *organic* and *inorganic* compounds was proposed by Jöns Jakob Berzelius in 1807. Historically, the organic chemistry included all compounds that came from living organisms, with the assumption that these contain an unmeasurable living force *vis vitalis*. But 1828 Friedrich Wöhler succeeded to produce urea (a substance in the urine of mammals) in the laboratory. After this discovery the definition of *organic compounds* had to be changed, and today they are defined as compounds that contain carbon. *Organic chemistry* deals with the properties, structure, reactivity and synthesis of these organic compounds. [Bru11, p. 4; 15]

4.2.1 Atoms and Elements

In chemistry atoms can be considered as basic building blocks. They consist of positive charged *protons*, negative charged *electrons* and neutrally charged *neutrons*¹. While the protons and neutrons build the atomic nucleus, the electrons orbit that nucleus. The number of protons in the nucleus of an atom defines its *atomic number*. This number is used to classify atoms into different *elements* and to arrange them in the periodic table [OR14a, p. 1]. If the atom is not charged, also the number of electrons are equal to the atomic number. In organic chemistry, only a small subset of elements is of

¹Further subdivisions of these particles would be part of particle physics and will not be considered in this thesis.

interest. The most common are hydrogen (H) and carbon (C), less common are nitrogen (N), oxygen (O), and fluorine (F), and very rare are phosphorus (P) and sulfur (S). Figure 4.3 shows the periodic table and highlights these organic elements.

To represent a single atom, there are different simplifying models focusing on various aspects. One part of that simplification is often the representation in 2D instead of 3D. The Bohr model, for example, shows the distribution of electrons on different shells around the atomic nucleus (see Fig. 4.2). For the bonding of atoms especially the number of electrons in the outer shells are of interest, as we will see in Section 4.2.2.

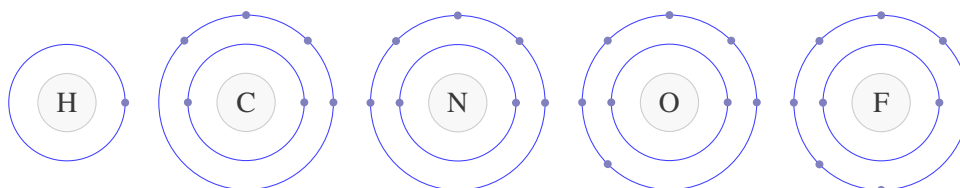


Figure 4.2: Bohr models of elements H, C, N, O, and F. The Bohr model is a simplified atom model that focuses on the distribution of the electrons on different shells around the atomic nucleus.

Periodic Table

Period

Legend

atomic number atomic mass

symbol name

electronegativity density

Symbol:

black = solid
blue = liquid
red = gas
gray = unknown
underlined = radioactive

Density:

red = kg/m³
black = kg/dm³

Series (fill color):

- Alkali Metal
- Alkaline Earth Metal
- Transition Metal
- Lanthanid
- Actinide
- Transactinide

Group

- Metal
- Metalloid
- Nonmetal
- Halogen
- Noble Gas

		Group										Group																
		1		2		3-12										13		14		15		16		17		18		
1	1	1.007																							4.002			
		H																							He			
		Hydrogen																							Helium			
		2.2	0.09																						0.179			
2	3	6.941	4	9.012																								
		Li	Be																									
		Lithium	Beryllium																									
		0.98	0.534		1.57		1.85																					
3	11	22.99	12	24.305																								
		Na	Mg																									
		Sodium	Magnesium																									
		0.93	0.971		1.31		1.74																					
4	19	39.098	20	40.078																								
		K	Ca																									
		Potassium	Calcium																									
		0.82	0.862		1.0		1.54																					
5	37	85.468	38	87.62																								
		Rb	Sr																									
		Rubidium	Strontium																									
		0.82	1.53		0.95		2.64																					
6	55	132.905	56	137.327																								
		Cs	Ba																									
		Cesium	Barium																									
		0.79	1.87		0.89		3.59																					
7	87	223.0	88	226.0																								
		Fr	Ra																									
		Francium	Radium																									
		0.7	1.87		0.9		5.5																					

57	138.905	58	140.116	59	140.908	60	144.242	61	145.0	62	150.36	63	151.964	64	157.25	65	158.925	66	162.5	67	164.93	68	167.259	69	168.934	70	173.054
La	Ce	Pr	Nd	Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb														
Lanthanum	Cerium	Praseodymium	Neodymium	Promethium	Samarium	Eurpium	Gadolinium	Terbium	Dysprosium	Holmium	Erbium	Thulium	Ytterbium														
1.1	6.15	1.12	6.77	1.13	6.77	1.14	7.01	1.13	7.26	1.17	7.52	1.2	5.24	1.2	7.9	1.22	8.23	1.23	8.8	1.24	9.07	1.25	9.32	1.1	1.25	6.97	
89	227.0	90	232.038	91	231.036	92	238.029	93	237.0	94	244.0	95	243.0	96	247.0	97	247.0	98	251.0	99	252.0	100	257.0	101	258.0	102	259.0
Ac	Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No														
Actinium	Thorium	Protactinium	Uranium	Neptunium	Plutonium	Americium	Curium	Berkelium	Californium	Einsteinium	Fermium	Mendelevium	Nobelium														
1.1	10.1	1.3	11.7	1.5	15.4	1.38	19	1.36	20.5	1.28	19.8	1.3	13.7	1.3	13.5	1.3	14.8	1.3	15.1	1.3	13.5	1.3	1.3	1.3	1.3	1.3	

Figure 4.3: In this periodic table the important organic elements are highlighted: hydrogen (H), carbon (C), oxygen (O), fluorine (F), phosphorus (P), and sulfur (S).

4.2.2 Bonds

Atoms are held together by chemical bonds and can form larger complexes – *molecules*. First, a distinction must be made between *strong* or *intramolecular bonds* and *weak* or *intermolecular bonds*. While the former are important for the cohesion of atoms within a single molecule, the latter are relevant for the attraction between different molecules.

4.2.2.1 Intramolecular Bonds

The strong bond is based on the electrostatic attraction between protons and electrons. In order to form a bond, the electrons are shared (*covalent bond*), moved (*ionic bond*), or delocalized (*metallic bond*) between the atoms.

Covalent bonds are the most common type of bond found in organic chemistry [OR14a, p. 5]. An example for a molecule that is based on covalent bonds is methane (CH₄). It consists of one carbon and four hydrogen atoms. As mentioned before, the electrons in the outer shell – the *valence electrons* – of these atoms are of importance for covalent bonds. While hydrogen has one electron in its outer shell, carbon has four (see Fig. 4.2). Figure 4.4 shows how these atoms are arranged to get full valence shells.

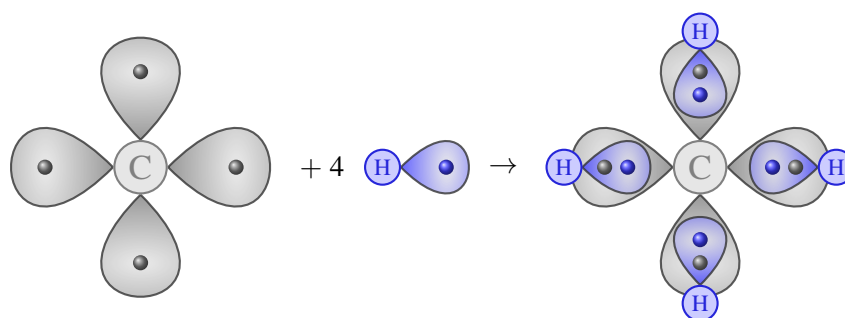


Figure 4.4: Visualization of C and H atoms using the *valence shell electron pair repulsion* (VSEPR) model. Only the orbitals of the outer shells are shown. Inside the orbitals you can see the number electrons. On the right side is the bonding of carbon with 4 hydrogen, i.e., the molecule methane.

In an *ionic bond*, the electrons are not shared as before, but are transferred from one atom to another, so that the (new) outer shells are subsequently filled [OR14a, p. 4]. An example of this is the molecule NaCl. Here, the single electron of the outermost shell of Na is transferred to Cl: $\text{Na} + \text{Cl} \longrightarrow \text{Na}^+ + \text{Cl}^- \longrightarrow \text{NaCl}$ (Fig. 4.5).

In a *metallic bond*, the valence electrons are delocalized and are free to move between the positively charged metal ions (see Fig. 4.6). They are therefore also referred to as an electron cloud. This results in various properties of metals, such as electrical and thermal conductivity.

4.2.2.2 Intermolecular Bonds

Two or more molecules can be held together by *intermolecular bonds*. Because the intermolecular forces (IMFs) are weak relative to the intramolecular forces, they are also called *weak bonds*.

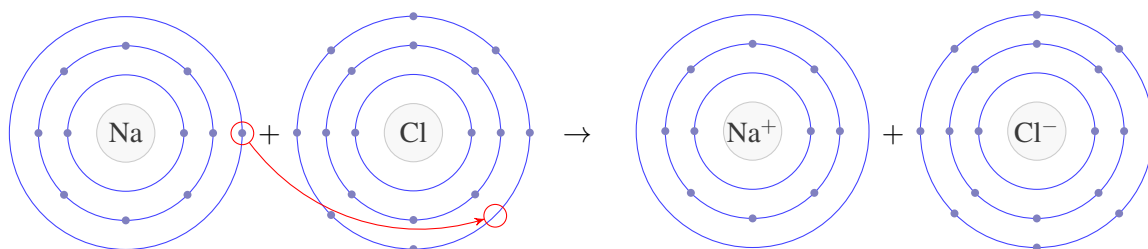


Figure 4.5: The ionic bond of NaCl represented by Bohr models. The valence electron of Na is transferred to Cl (as marked by the red arrow). This results in the charged elements Na^+ and Cl^- .

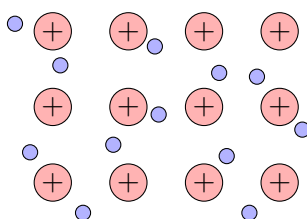


Figure 4.6: Schematic of a positively charged metal ion grid (red) with freely movable electrons (blue).

The electrons in nonpolar molecules are distributed more or less uniformly throughout the molecule. When several such molecules come close to each other, as happens in a liquid, they can interact with each other. The electrons of one molecule can temporarily polarize those of an adjacent one, resulting in an uneven electron distribution. This leads to the temporary formation of dipoles in the molecules. These dipoles then interact through the *van der Waals forces*. [OR14c, p. 142]

Another weak bond that has a major influence on the properties of naturally occurring macromolecules, such as proteins, is the *hydrogen bond* [OR14b, p. 995]. It is an attractive interaction between a hydrogen atom and an electronegative atom Y . The other atom may belong to a different molecule or to the same molecule (in which case it would be an intramolecular force). The hydrogen itself is attached to a second, relatively electronegative atom X . This written as $X\text{--H}\cdots Y$, where the dots represent the hydrogen bond (see Fig. 4.7a). Typical electronegative atoms for X and Y are N, O, and F. A simple natural example of such a bond is water (see Fig. 4.7b). [Aru+11; IUP]



(a) This is a general schematic description of a hydrogen bond with the local charges (δ^+ and δ^-). $X\text{--H}$ represents the hydrogen donor, while Y is the acceptor.

(b) A hydrogen bond that is formed between two H_2O molecules within water. The locally positively charged H and the locally negatively charged O attract each other.

Figure 4.7: Hydrogen bonds in general (a) and in water (b).

4.2.3 Molecules

Molecules consist of multiple atoms (see Section 4.2.1) connected by intramolecular bonds (see Section 4.2.2.1). Standard descriptions of molecules are the chemical formula and the structural formula. We will look at these and other representations below. Here we start with one-dimensional representations with few details and end with detailed 3D ones (Sections 4.2.3.1 to 4.2.3.3). For comparison, we use the molecules propane and α -D-glucopyranose. Then, some properties of molecules (Section 4.2.3.4) and their representations (Section 4.2.3.5) are presented.

4.2.3.1 1D / String Representations

Propane is a molecule consisting of 3 carbon (C) atoms and 8 hydrogen (H) atoms and therefore has the *chemical formula* C_3H_8 . This could be seen as a first simple 1D representation. However, since only the number of different atoms is given here, but not how they are connected, this representation is mostly ambiguous.

A more specific textual description is the *IUPAC nomenclature of organic chemistry*, developed by the *International Union of Pure and Applied Chemistry (IUPAC)* [FPoP14]. Here, the parent hydride in particular is identified. This is usually the longest molecular chain of hydrocarbons. In this way, the atoms are placed in an unambiguous order. In relation to the parent hydride, branches, bonds, etc. can then be specified. For example, the IUPAC nomenclature for α -D-glucopyranose is: *(3R,4S,5S,6R)-6-(Hydroxymethyl)oxan-2,3,4,5-tetrol*. For a more detailed derivation of this nomenclature, please refer to the literature [FPoP14].

A description that is also more specific, but shorter and easier to interpret compared to the IUPAC nomenclature, is the *simplified molecular-input line-entry system (SMILES)* [Wei88]. Due to its simplicity and already long use, the SMILES format is also often supported by molecular editors. However, it must be noted that it is still a simplified and ambiguous representation for a 3D molecule. In SMILES, the atoms are given in the order in which they are connected. Bonds between two atoms can be specified with: - for single (optional), = for double, and # for triple. Branches are indicated by parentheses (); they can also be nested. To close a ring, two identical numbers (e.g., 1) are written behind each of the atoms to be connected. The specification of hydrogen atoms is optional and usually omitted, so they are automatically added based on the free valences. The previously mentioned molecule propane could be described in detail as [CH3]-[CH2]-[CH3], but also briefly as CCC. Since this is very short and simple, we look at the slightly larger molecule α -D-glucopyranose ($C_6H_{12}O_6$) in Fig. 4.8.

A common structure in organic compounds is *aromatic rings*. These are cyclic and usually planar structures with delocalized electrons. This makes them stable compared to simple bonds. A simple example of this is benzene (C_6H_6) (see Fig. 4.9). To describe this easily with SMILES, besides $C1=CC=CC=C1$, there are other notations: $C:1:C:C:C:C1$, where : represents an aromatic bond, or even shorter by using lowercase letters for the atoms: $c1ccccc1$.

Furthermore, it is possible to describe stereoisomers (using /, \, and @). These are molecules that are structurally the same but twisted at certain bonds and therefore different (see Section 4.2.3.4). Isotopes such as deuterium – hydrogen with 2 neutrons in the nucleus – can be described as [2H].

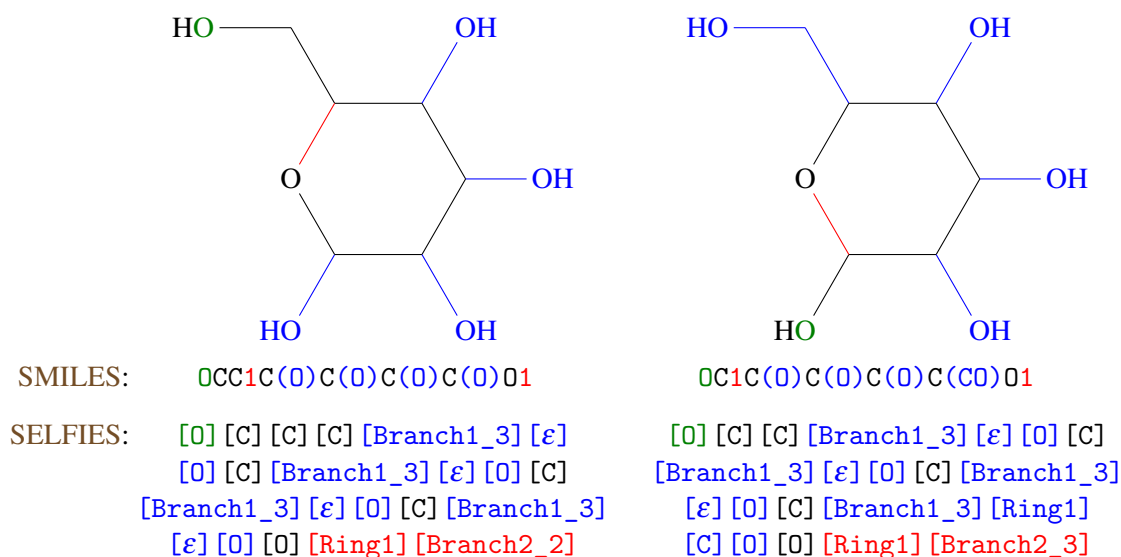


Figure 4.8: Molecular structure formula, SMILES, and self-referencing embedded strings (SELFIES) representation of α -D-glucopyranose ($C_6H_{12}O_6$). Note that there is implicitly a C atom at the nodes without a letter. Since there are many ways to describe the same molecule, two different ones are shown here as examples. The starting atom is colored green in each case. On the left side, the description starts at the top left and goes through the molecule in the clockwise direction. On the right side, the description starts at the bottom left and then goes through the molecule against the clockwise direction. The branches indicated by brackets are highlighted blue. For closing a ring with SMILES two corresponding numbers are used, behind the atoms that should be bonded (marked in red). Hydrogen is implicitly encoded. In the SELFIES the token behind [Branch1_3] is interpreted as number, that gives the size of the branch. In this case [ε] is size 1, so only the next token [O] belongs to the branch. Likewise, the token after [Ring1] is interpreted as a number, but here it indicates how far back the token to be connected is.

The same molecule can be described in many different ways with SMILES. It can be started with any atom and also the order of the further atoms can be varied by the different use of branches. Thus, for example OCC1C(O)C(O)C(O)C(O)O1 and OC1C(O)C(O)C(O)C(CO)O1 would describe the same molecule as shown in Fig. 4.8. Here, instead of starting with the OH group from the top left, we start with the one from the bottom left. In addition, the part at the top left is described here as a branch instead of a main path. In order to resolve this ambiguity and thus make it possible to compare different molecules with each other *canonical SMILES* exist. There are several algorithms that can calculate canonical SMILES strings, e.g., the Python API of RDKit offers a possibility for this.

SMILES can be used to describe any molecules, even those that are impossible in reality, e.g., in the case of C#C#C the middle C atom would contain 6 bonds when only 4 are possible. An extension which allows only valid molecules with respect to the valences is described below.

A more recent string-based encoding of molecules is *self-referencing embedded strings (SELFIES)* [Kre+20]. The particularity here is that with SELFIES only valid molecules can be described. This is achieved by interpreting each token based on the tokens read previously. In particular, only valid bonds are formed based on the valence electrons present. For example, the token [#C], which according to the SMILES interpretation represents a triple bond to a carbon atom, is interpreted as a single bond

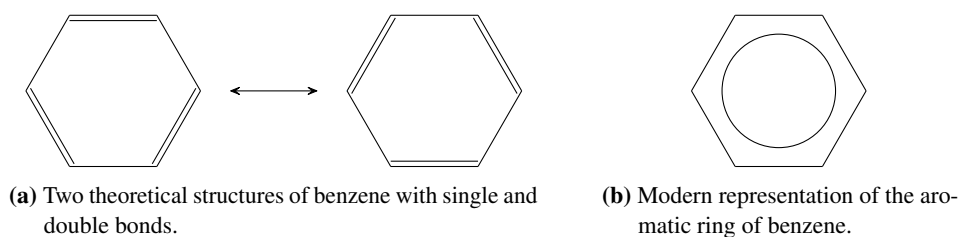


Figure 4.9: Different representations of an aromatic benzene molecule C_6H_6 . It consists of a ring of 6 carbon atoms, each of which is bonded to a hydrogen atom. Theoretically, one could imagine two variants in which there are alternating single and double bonds between the C atoms (a), but a mixture of the two, in which the electrons can move freely in the ring, is more realistic (b).

if there is only one free valence electron, or as a double bond if there are two. Only when there are at least three free valence electrons will it be interpreted as a triple bond. Hydrogen atoms cannot be specified explicitly here, but they are automatically added at the free valences. To describe branches and rings, the following token is interpreted as a number. This number indicates how long the branch is or with which token how far back a ring should be closed. In Fig. 4.8, the **SELFIES** for the depicted α -D-glucopyranose is also given. The substring for the branches of the OH groups (marked in blue) is: [Branch1_3] [ϵ] [0]. The token [ϵ] is interpreted as 1, i.e., the branch has length 1 and thus only the following token [0] is in the branch.

Another string-based encoding for molecules is *International Chemical Identifier (InChI)*, developed by the **IUPAC**. This format contains several layers for different levels of information and abstraction. Unlike the previously described representations, this ensures that each structure can be assigned a unique **InChI** string, which is important for use in databases. However, **SMILES** strings are often used because of their simpler structure, which makes them easier for humans to read.

Fingerprints represent another special type of representation, encoding structural and possibly other application-specific information within a bit vector. Although they usually do not allow a clear assignment to a chemical formula, they do allow a direct comparison of the stored molecular properties. Since fingerprints are not used in this thesis, we only mention them here for completeness and refer to the literature presented by Riniker and Landrum [RL13] for an overview and benchmark comparison of different fingerprints.

4.2.3.2 2D Representations

Figure 4.10 shows different 2D-representations of propane. In the *Lewis structure* (see Fig. 4.10a) all atoms and their bonds are displayed. Shared pairs of electrons are represented by lines between the atoms. The Lewis structure is planar and does not give any information about the real spatial arrangement. The *natta projection* (see Fig. 4.10b) adds some spatial information by visualizing connections that run forward or backward with filled or dashed triangular connections, respectively. Different angles between the atoms are also considered. The *semi-structural formula* contains the same information as the Lewis structure, but with simpler notation (see Fig. 4.10c). Here, atoms and atom groups that are connected to carbon are written directly behind it. For example, the 3 hydrogen atoms connected to one carbon atom on the edge of the molecule are written here as chemical formula CH_3 .

A further simplification of the notation is achieved by the *skeletal formula* (see Fig. 4.10d), in which carbon atoms are implicitly placed at each vertex of a compound unless another atom is explicitly specified. In addition, hydrogen atoms bonded to carbon are implied and omitted. Therefore, propane can be simply visualized as two lines with an 120° angle between them.

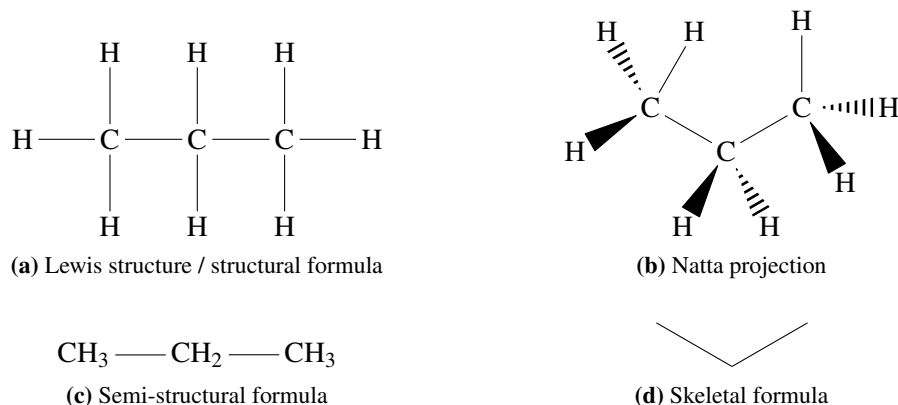


Figure 4.10: Different 2D-representations of the molecule propane.

For molecules containing a six-membered ring of carbon atoms with single bonds, there are special visualizations that should show the spatial arrangement. In contrast to the aromatic rings mentioned above (see Fig. 4.9), the rings with single bonds do not form a planar surface. The simplest molecule of this type is cyclohexane (C_6H_{12}). In Fig. 4.11, the slightly larger molecule α -D-glucose ($C_6H_{12}O_6$) is illustrated. Figure 4.11a shows the *chair representation*, which is the most common conformation. This illustrates in particular that it is not a planar hexagon, but that there can be different spatial conformations (stereochemistry). The *Haworth projection* (see Fig. 4.11b) is also used for this purpose. The *Fischer projection* (see Fig. 4.11c) maps a tetrahedral structure onto a planar surface. Here, the carbon atoms are arranged vertically on top of each other in a line. To represent a ring, a rounded line is inserted on the outside. The Fischer projection can differentiate between chiral molecules².

A molecule can also be described simplified as graph $G = (N, E)$, where $N = \{a_0, \dots, a_{n-1}\}$ is the set of atoms and $E = \{e_{i,j}\}$ is the set of bonds between atoms. In a single atom a_i would then be stored which element it is. It would also be conceivable to store further atomic properties. An edge $e_{i,j}$ indicates how the atoms a_i and a_j are connected. So a 1 could indicate that it is a simple covalent bond. For example, a water molecule H_2O could be described as a graph as follows: $a_0 = O, a_1 = H, a_2 = H$ and $e_{0,1} = 1, e_{0,2} = 1$. Graphs offer the advantage of being well established and well researched in computer science. However, depending on the choice of information to be stored, difficulties may arise when converting to a 3D model.

4.2.3.3 3D Representations

To create a 3D representation, the 3D coordinates of all the atoms must be given. A *protein data bank (PDB)* file can be used for this purpose. Among other things, it contains a list of the exact positions of each atom. This file format is used to store molecules in the database of the same name.

²Mirrored but different molecules are *chiral*. This will be explained in more detail in Section 4.2.3.4.

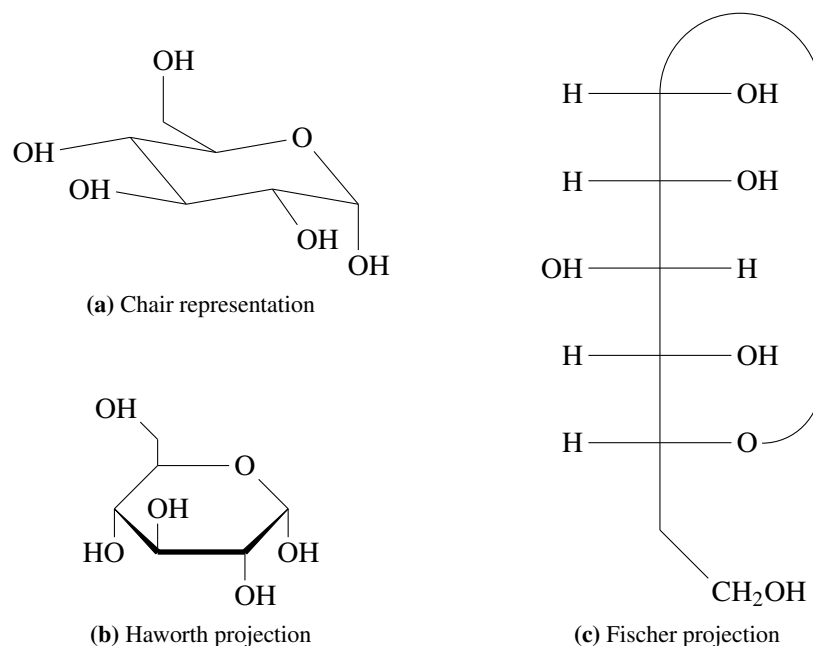


Figure 4.11: Different representations of the α -D-glucopyranose molecule, which contains a ring. Note that in (a) and (b), some hydrogen atoms are implicit around the carbon atoms.

Figure 4.12 shows propane with three commonly used 3D models. In all models the atoms are colored according to the CPK scheme (named after its inventors Corey, Pauling and Koltun) depending on their type (see Table 4.1). The *sticks model* shows the structural composition and the connections between the atoms, while the atoms themselves form the corners and can only be distinguished by the coloring (see Fig. 4.12a). In the *ball-and-stick model* (see Fig. 4.12b), as the name indicates, the atoms are represented as balls and the compounds as sticks. Here, the atoms are displayed in different sizes based on the proportions of their actual size differences. Finally, Fig. 4.12c shows the *space-filling model*, where the sphere containing the electrons is represented. The covalent bond between the C and H atoms causes their electron spheres to overlap. This type of representation makes the shape more clearly visible. Therefore, it is also used for very large molecules, such as proteins, when visualizing how they interact with smaller molecules. We will see this in the following chapters.

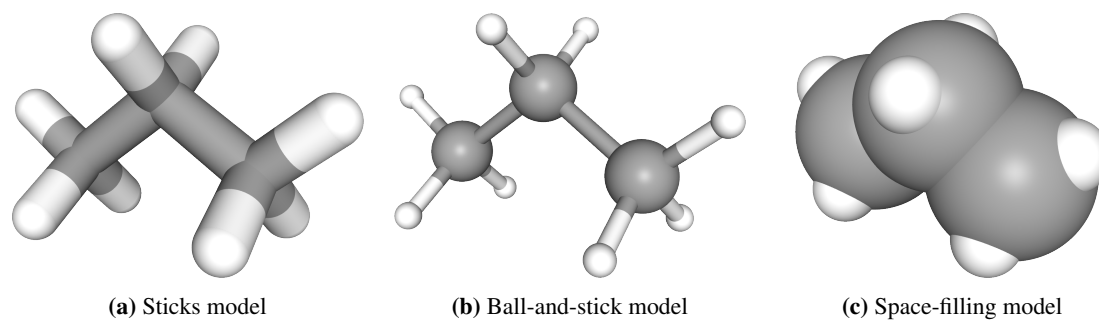









Figure 4.12: 3D models of propane. The meaning of the atoms colors is defined in Table 4.1. (image source: [Pub])

Table 4.1: CPK coloring scheme for atoms by Corey, Pauling and Koltun.

Symbol	Name	Color	
	H	hydrogen	white
	C	carbon	black or gray
	N	nitrogen	blue
	O	oxygen	red
	F	fluorine	green
	P	phosphorus	purple
	S	sulfur	yellow

4.2.3.4 Molecule characteristics

In this section, we will focus in particular on some structural properties of molecules that need to be considered, both for their unique description and for the calculation of the 3D coordinates of atoms. This is necessary, for example, when a 1D or 2D representation is to be transformed into a 3D model.

The angle between two atoms and also their distance can vary depending on the element and the type of bond. Figure 4.13 compares the distance between the carbon atoms of ethane and ethene and the angle between C=C–H for ethene and C=C–C for propene. [OR14d, pp. 164–165]

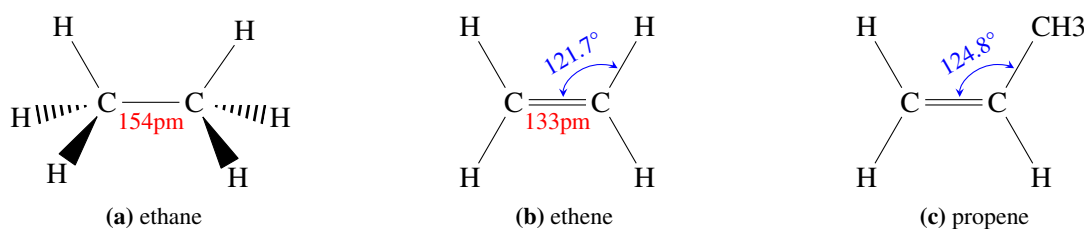


Figure 4.13: Here are some similar molecules, but they have different atomic distances and angles. For example, the spacing of a single bond C–C in ethane, 154 pm, is larger than that of a double bond C=C in ethene, 133 pm, cf. (a) and (b). Likewise, the angle between the atoms varies depending on the structure, cf. (b) and (c).

Molecules with the same chemical formula contain exactly the same atoms, but are often not unique. Molecules made up of the same parts (Greek: *isos meros*) that have different 3-dimensional structures are called *isomers*. Isomers whose atoms are connected in different ways are called *constitutional* or *structural* isomers. An example is butane C₄H₁₀, which exists both in a chain as *n*-butane CH₃–CH₂–CH₂–CH₃ and with a central C atom as isobutane CH(CH₃)₃. When molecules that represent the same structural isomer are different from each other, they are called *stereoisomers*. They can be divided into conformers, enantiomers, and diastereomers.

Two atoms with a single bond between them can rotate axially around it. If these two atoms are each bonded to other atoms within a molecule, the rotation can result in different *conformations* of this molecule. Ethane has two special conformations: *eclipsed*, in which the hydrogen atoms are exactly opposite each other on both sides (see Fig. 4.13a), and *staggered*, in which the hydrogen atoms are rotated 60° around the C–C axis and are thus maximally distant from each other. Since the energy difference between these two conformations is relatively small, a rapid switch between these states occurs at room temperature. [OR14c, pp. 119–121]

Enantiomers are mirror images of each other, which are non-superposable, i.e., they cannot be formed by a combination of rotations, translations and conformational changes. An analogous example of such an object is a hand (Greek: *chiron*). The left and the right hand are mirror images of each other, but not superposable. Such objects are therefore also called *chiral*. An example for a chiral molecule is shown in Fig. 4.14. [OR14e, pp. 242–243]

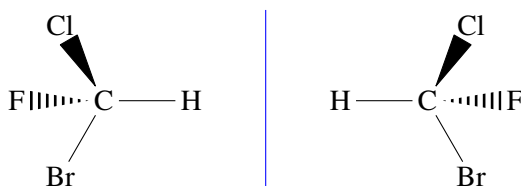


Figure 4.14: Bromochlorofluoromethane has no plane of symmetry and is therefore chiral. The mirror image along the blue line creates two molecules that cannot be superimposed.

One type of *diastereomers* are *cis-trans isomers*. Such molecules usually contain double bonds and are therefore, in contrast to conformational isomers, not arbitrarily rotatable. However, two different variants of the structure can exist in principle, provided that the atoms on both sides of the double bond are distinguishable. Figure 4.15 shows the general structure of these isomers. In the *cis* isomer, the same groups are on the same side, while in the *trans* isomer they are on opposite sides. An example of this is butene $\text{CH}_3\text{-CH}=\text{CH-CH}_3$. A counterexample is propene, where the atoms on one side are identical (see left side of Fig. 4.13c). [OR14d, pp. 168–169]



Figure 4.15: This is an example of a molecule with one *cis* and one *trans* isomer. Because of the double bond between the carbon atoms, they cannot rotate freely and all six atoms are coplanar, meaning they are in the same plane. *X* and *Y* can represent different single atoms or groups of atoms, e.g., for butene $X = \text{CH}_3$ and $Y = \text{H}$. For the *cis* isomer, the groups are on the same side (a) and for *trans* on opposite sides (b).

4.2.3.5 Representation characteristics

As we have seen, there are different ways of representing molecules with varying degrees of detail and emphasis, e.g., some attempt to show three-dimensionality while others show only a simplified structure of interconnected atoms. On the one hand, it is possible to investigate which properties of the molecules are uniquely described and, on the other hand, whether there is a canonical description for each molecule. Therefore, we can describe different properties of the representations:

Unique Arrangement A molecule can be described simplified as graph $G = (N, E)$, where N is the set of atoms and E is the set of bonds between atoms. If such a graph can be generated from

a representation, it is considered to fulfill the *unique arrangement* property. That means it is unambiguous which atoms are connected together.

Unique Diastereomers Indicates whether the representation can distinguish between different diastereomers (see Section 4.2.3.4).

Unique Enantiomers Indicates whether the representation can distinguish between different enantiomers.

Unique Conformations Indicates whether the representation can distinguish between different conformations.

Unique Position If a 3-dimensional structure is completely described or can be uniquely derived, this property is given.

Canonical A representation is canonical, if there is no other representation of the same molecule, i.e., the mapping from representation to molecule is injective.

Which of these properties are fulfilled for all presented representations is specified in Table 4.2.

Table 4.2: Overview of molecule representations and their satisfied (✓) or unsatisfied (✗) properties. The ○ means, it is possible to satisfy the property but not required by the representation. (*Note:* In the digital version, the representations and properties are hyperlinked to the corresponding examples or text descriptions from earlier.)

Representation	Uniqueness					
	Arrangement	Diast.	Enant.	Conf.	Position	Canonical
chemical formula	✗	✗	✗	✗	✗	✗
IUPAC nomenclature	✓	✓	✓	✗	✗	✓
SMILES	✓	✗	✗	✗	✗	✗
canonical SMILES	✓	○	✗	✗	✗	✓
SELFIES	✓	○	✗	✗	✗	✗
InChI	✓	✓	✓	✗	✗	✓
Fingerprints	✗	✗	✗	✗	✗	✓
Lewis structure	✓	✓	✗	✗	✗	✗
Natta projection	✓	✓	✓	✗	✗	✗
Semi-structural formula	✓	✓	✗	✗	✗	✗
Skeletal formula	✓	✓	✗	✗	✗	✗
Chair representation	✓	✓	✓	✗	✗	✗
Haworth projection	✓	✓	✓	✗	✗	✗
Fischer projection	✓	✓	✓	✗	✗	✓
Graph	✓	✗	✗	✗	✗	✗
Sticks model	✓	✓	✓	✓	✓	✗
Ball-and-stick model	✓	✓	✓	✓	✓	✗
Space-filling model	✓	✓	✓	✓	✓	✗
PDB file	✓	✓	✓	✓	✓	✗

4.3 Evolutionary Algorithms

An *evolutionary algorithm (EA)* is a metaheuristic based on a biological model. It attempts to mimic the natural evolution of a population, which produces increasingly well-adapted individuals over time according to the Darwinian principle of *survival of the fittest*. In evolution, a new individual acquires certain characteristics (genes) through inheritance and mutation. The better an individual is at certain traits, which are measured as fitness, the greater the likelihood that it will survive or pass on its genes.

To formulate the natural evolution as an algorithm, it is necessary to encode the problem to be solved in the form of an individual. For example, binary coding can be used as the encoding, i.e., each bit represents a certain trait and the binary value indicates whether this trait is available for the specific individual or not. In order to obtain better results, all the relevant properties can be identified and stored in a meaningful format so that the operations described below can be carried out. It is also a prerequisite that each individual can be assigned a fitness with a function that describes the goodness of the solution found, the so-called fitness function. An EA contains operators with two different functions: genetic variation operators, which are used to explore the solution space, and selection operators, which make selections based on fitness and give direction to the optimization procedure.

In this section, we mainly focus on *evolution strategies (ES)*. Other basic forms of EAs are the *genetic algorithm*, *evolutionary programming* and *genetic programming* [Kra03]. They are briefly explained below. Today, however, some of these terminologies are used similarly.

In the *genetic algorithm (GA)*, the individuals are encoded as bit strings and varied using n-point crossover and bit flip mutation. In most cases, genotype-phenotype mapping [Hol75] is used to transform the binary coded individuals (genotypes) for evaluation. The phenotype describes the actual appearance, analogous to nature, where there is the distinction between DNA and RNA.

Evolutionary programming (EP) was originally developed to find deterministic finite automata (DFA) that give the correct outputs for certain input words [FAM66]. It is similar to the ES of Schwefel and Rechenberg, but was designed for this more specific problem [Fog94]. In this method, only mutation was used to vary and recombination was omitted. After each parent has produced an offspring, the better half of the parents and the offspring is selected by competitive selection.

Genetic programming (GP) attempts to automatically generate computer programs that can solve specific problems. The programs are represented by trees. Their leaves represent variables or constants and the inner nodes represent functions. As programming language, e.g., Lisp can be used, since it supports these tree structures. [Koz92]

We now consider all the required steps of the *evolution strategies (ES)* [BS02] in overview (see Fig. 4.16) and define some variables. Later, some of the steps will be discussed in more detail.

First, a starting population \mathcal{P} is needed, consisting of μ individuals. These individuals are usually generated randomly. Their quality is calculated with the fitness function and stored. Next, λ children are created, with the following procedure for each child. First, ρ parents are selected to serve as the basis. The selection can be done randomly or depending on the fitness. In particular, $\rho = 1$ is also possible, in which case the following step of recombination can be omitted. During the recombination step, the genes of the parents are combined in a specific way to form a new individual. Then a mutation of the individual takes place. This is usually a minor change that can be made to only a single parameter

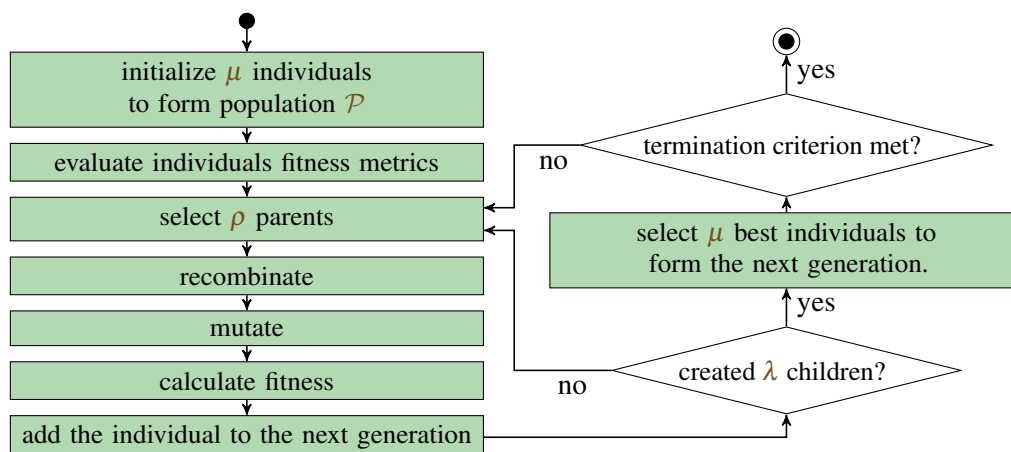


Figure 4.16: Activity diagram of an evolutionary algorithm

if necessary. Now the new individual is ready and is evaluated with a fitness function f . Finally, it is added to the successor generation \mathcal{P}' . After all λ offspring have been created, μ parents must be selected for the new generation based on fitness. This repetition takes place until a termination criterion is met. This is often done by specifying a maximum number of generations or fitness function calls, or a maximum runtime. It is also possible, depending on the extent of improvement over the last generations, to terminate the algorithm. These termination criteria can of course also be combined.

4.3.1 Parameters

To specify the **ES**, *exogenous* and *endogenous* parameters are used. The exogenous parameters are specified from outside the **ES** and remain unchanged during execution. These include the size of the parent population μ , the size of the descendant population λ , and the number of parents per individual ρ . To briefly notate an **ES** with certain exogenous parameters, the notation $(\mu/\rho \ddagger \lambda)$ -**ES** is used. The plus or comma here stands for the type of selection (see Section 4.3.4). The $/\rho$ is often omitted.

Endogenous parameters can be adjusted at runtime based on the circumstances at hand. For example, the mutation strength σ can be varied to become smaller over time, depending on the generation. Alternatively, it can be increased or decreased depending on the success rate, which is determined by counting how many offspring are better than the previous best individual. While an **ES** with a larger population only considers the successor generation to calculate the success rate, the (1+1)-**ES** considers the next n generations. This is especially used in Rechenberg's 1/5 success rule [Rec73], which states that a success rate of 1/5 should be aimed at. To do this, the mutation rate is increased, if the success rate is greater than 1/5 and otherwise it is decreased.

4.3.2 Recombination

Recombination, along with mutation, is one of the variation operators used to create a new individual. This is done based on ρ parents. A special case is $\rho = 1$, i.e., there is only one parent and accordingly it is only cloned. If $\rho = 2$, it is called local recombination and if $\rho > 2$, it is called multirecombination. Object variables and endogenous strategy parameters are recombined. In intermediate recombination,

the arithmetic mean is calculated for the individual parameters of the individuals, while in discrete recombination, individual parameters are adopted exactly.

In the evolutionary procedures for molecule generation in the following chapters, recombination is not used, because it would lead to very strong changes and often to defective molecules in the used 1D molecule representations (**SELFIES** and **SMILES**). Therefore, the reader is referred to other literature (e.g., [BS02]) for a more detailed description of this operator.

4.3.3 Mutation

The mutation operator has the greatest impact on genetic variation. For mutation there are three desirable properties [BS02]:

1. **Reachability:** Each point in the solution space must be reachable within a finite number of mutations from any starting point. This property is a necessary criterion for the reachability of a global optimum.
2. **Unbiasedness:** Mutation is used to explore the search space and therefore should not specify a particular direction (no bias). Only selection, with the help of fitness information, can guide evolution in a particular direction. However, in the case of limited search spaces, a bias can be beneficial [KS06].
3. **Scalability:** The mutation strength σ must be adaptable. In continuous search spaces, σ is also called step size. Scalability is particularly necessary for parameter tuning.

For example, if a real number is to be mutated, Gaussian mutation can be applied, which corresponds to adding a random value using the Gaussian distribution. Here, all three properties are respected: (1) the value can be changed in \mathbb{R} without restrictions, (2) the Gaussian distribution is symmetric and thus unbiased, and (3) the standard deviation can be used for scaling.

Since the mutation operator must be chosen for the specific problem and in particular according to the structure of the genotype, an exact implementation for the use case of molecule design is described later (see Section 5.3.2).

4.3.4 Selection

While the genetic operators described earlier are used to explore the search space, selection gives direction to evolution based on fitness. Thus, optimization takes place. There are several variants in the selection of individuals for the next generation. Evolutionary strategies are distinguished between plus and comma selection. Both selection operators have in common, that they select the μ best individuals for the next generation. However, they differ in how they select the parents from which the new individuals are created.

The *comma selection* selects parents only from the λ individuals of the successor generation \mathcal{P}' , omitting previous individuals, as it is also common in nature. In this way, good solutions may be lost, but their retention could also lead to a local optimum that is never left. The comma selection is notated

as (μ, λ) -ES. In order to have selection, $\mu < \lambda$ must hold. Comma selection is particularly suitable for unbounded search spaces, such as the \mathbb{R}^N [Sch87].

In contrast, with *plus selection*, the parents are selected from the new and the old generation, i.e., from $\mathcal{P} \cup \mathcal{P}'$. This way it is possible that excellent individuals remain in the population and can be used further and further. Plus selection is particularly suitable for discrete finite search spaces, such as combinatorial optimization problems [BS02; Her90; Bey92].

4.4 Conclusion

Molecule design helps to find protease inhibitors that can be used to prevent virus replications. Therefore, molecular design is an important step in developing a drug against viral diseases. This chapter gave an introduction to organic chemistry and EAs. Besides the different types of bonding, we have also provided an overview of different molecule representations. The approaches presented in the following two chapters, are both based on an EA and use the string representation for molecules.

5 Evolutionary Multi-Objective Approach

This chapter presents methods for protease inhibitor design. Here, the coronavirus SARS-CoV-2, which causes the disease COVID-19, is considered as a use case due to its timeliness during the research period. To stop virus replication, a biomolecule is sought that binds to the virus protease enzyme, as described in Section 4.1. Such a molecule is also called a *ligand* or *protease inhibitor*. In the case of SARS-CoV-2, the crystal structure of the main protease M^{pro} is already known [Jin+20]. The search for a suitable protease inhibitor can be regarded as an optimization problem. Here, not only the binding of the ligand is important, but also other properties, such as synthesizability and toxicity. Therefore, the molecule search problem is a MOO problem, which is solved in this chapter with the help of evolutionary algorithms.

This chapter is organized as follows. First, related work is presented in Section 5.1. Then, Section 5.2 introduces the five metrics, which are used to measure the suitability of a molecule. Section 5.3 presents our two evolutionary search methods specialized for molecules and the two strategies we use for MOO: *weighted sum evolutionary molecule search (WSEMS)* and *Pareto ranking evolutionary molecule search (PREMS)*. Section 5.4 gives insight into the experiments performed and is supported by different visualizations. Finally, a conclusion is drawn in Section 5.5.

Parts of this chapter are based on the following published paper:

Tim Cofala, Lars Elend, Philip Mirbach, Jonas Prellberg, Thomas Teusch, and Oliver Kramer. “Evolutionary Multi-objective Design of SARS-CoV-2 Protease Inhibitor Candidates”. In: *Parallel Problem Solving from Nature – PPSN XVI*. ed. by Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 357–371. ISBN: 978-3-030-58115-2. DOI: [10.1007/978-3-030-58115-2_25](https://doi.org/10.1007/978-3-030-58115-2_25)

Section 5.2 is additionally also based on:

Lars Elend, Luise Jacobsen, Tim Cofala, Jonas Prellberg, Thomas Teusch, Oliver Kramer, and Ilia A. Solov'yov. “Design of SARS-CoV-2 Main Protease Inhibitors Using Artificial Intelligence and Molecular Dynamic Simulations”. In: *Molecules* 27.13 (13 Jan. 2022), p. 4020. ISSN: 1420-3049. DOI: [10.3390/molecules27134020](https://doi.org/10.3390/molecules27134020)

5.1 Related Work

The categorization of methods for *de novo* drug design can vary [DSC15; Bro+19]. Some studies involve the direct construction of molecules from atoms [DTG00; Nig+19], while others utilize chemical fragments as the smallest building blocks [PHK01]. The objectives of these publications also differ. In certain cases, the goal is to identify drugs that specifically bind to a particular protein

binding site, as demonstrated in our work or [PHK01; YPL20]. Some other studies focus on generating drug-like molecules in general, as explored in [DTG00; Pol+20].

ADAPT [PHK01] is a fragment-based approach that uses an EA on an acyclic graph-representation comprising chemical fragments to optimize molecules' binding to a specific binding site. The resulting compounds' fitness is assessed through docking simulations with the target protein binding site and common drug-likeness indicators. Single-target EAs have been successfully used to optimize peptide ligands [RBF12; Kra+18], where the fitness of individuals was determined experimentally in-vitro. In contrast, Douguet, Thoreau, and Grassy [DTG00] work at the atom level using the SMILES representation instead of fragments. Their genetic algorithm focuses solely on optimizing drug-likeness rather than binding to a specific ligand. Moreover, their single-objective algorithm assigns constant coefficients to different properties within the fitness function. Similarly, Nigam et al. [Nig+19] present a genetic algorithm based on the SMILES representation for general molecule design. To enhance diversity, they employ a deep neural network as an adaptive fitness function, penalizing long-surviving molecules. Unlike approaches such as ours, which tend to focus on the distribution of drug-like molecules, their genetic algorithm can explore the entire chemical space without restriction.

LigBuilder, as described in the study by Yuan, Pei, and Lai [YPL20], is a software tool designed for drug design, based on a genetic algorithm. It allows optimization of binding to multiple targets, offering the advantage of treating complex diseases with a single drug without the risk of drug-drug interactions associated with therapies with a combination of drugs. In a separate study, Lameijer et al. [Lam+06] developed Molecule Evaluator, a program that employs an atomic-based evolutionary approach. In this method the fitness is evaluated by the user. A subset of molecules selected based on the evaluation is subsequently subjected to experimental investigations.

Brown et al. [Bro+04] employed a graph-based representation of molecules to perform MOO. Their approach utilized a multi-objective evolutionary algorithm with a Pareto ranking scheme to guide the optimization process. In contrast, Wager et al. [Wag+16] developed a tool for identifying potential central nervous system (CNS) drugs through MOO. Their method optimizes molecules based on six physical properties. Unlike our approach, their tool relies on medical knowledge rather than evolutionary algorithms. Van der Horst et al. [vdHor+12] developed a multi-objective evolutionary algorithm for designing adenosine receptor ligands using a pharmacophore model and three support vector machines. The results of their approach were experimentally validated. Nicolaou and Brown [NB13] provided a concise review that focuses on the MOO of drugs. They summarize different problem definitions and various methods for MOO in the context of drug design.

Luukkonen et al. [Luu+23] give a good overview of the latest AI methods in the area of multi-objective drug design. They provide a comparison table with information about the architecture, representation, MOO strategy, and objectives of each method.

5.2 Molecule Design Metrics

In computer-aided molecular design, metrics are needed to evaluate the suitability of a molecule. This chapter provides an overview of the five metrics used to assess how likely a molecule is to be

considered as a drug candidate and inhibitor (for example in the case of M^{pro}). Table 5.1 shows an overview of value ranges and the optima of these metrics, which are described in detail below.

Table 5.1: Value ranges and optima of the used metrics. The toxicity filters, can either be fulfilled or not and therefore only has the values 0 or 1.

	metric	value range	optimum
BA	Binding Affinity	\mathbb{R}	$-\infty$
SA	Synthetic Accessibility	[1, 10]	1
QED	Quantitative Estimate of Drug-likeness	[0, 1]	1
NP	Natural Product-Likeness	[-5, 5]	5
TF	Toxicity Filters	{0, 1}	1

5.2.1 Binding Affinity

The *binding affinity* (*BA*) estimates the binding free energy between the receptor and a potential ligand. Sometimes it is also referred to as *docking score*. It is crucial to evaluate whether the ligand binds to the receptor. The automated docking tool *AutoDock* [Mor+09] is a widely used tool to calculate this metric. AutoDock uses grid-based look-up tables to speed up calculations. The binding energy of all atoms of the ligand are determined for all positions of the grid using semi-empirical force field methods. A Lamarckian genetic algorithm is then used to find the best position and binding energy for a ligand. A further development of this is *AutoDock Vina* (Vina) [TO10], that uses a hybrid scoring function based on empirical and knowledge-based data [TO10]. *QuickVina 2* [Alh+15] improves the search algorithm by performing the most complex part of the optimization only for very promising ligand positions. Gaillard [Gai18] showed that Vina outperforms other docking software. We use *QuickVina 2* because it computes faster than Vina while producing comparable results [Alh+15].

The lower the *BA*, the stronger the potential ligand is expected to bind to the receptor. It should be noted that because of its extensive use of heuristics, *QuickVina 2* provides only an estimate of the true *BA*. However, since the evolutionary design of inhibitors requires many *BA* calculations, a balance between accuracy and computational time is important. Therefore, one binding affinity calculation per molecule is made. While it is conceivable to perform multiple runs and average the results to obtain a more accurate estimate, this would dramatically increase the computational time per molecule and result in fewer molecules being generated. Promising molecule candidates can be validated with more accurate methods (see Section 6.2) to obtain a better estimate of their true performance.

5.2.2 Synthetic Accessibility

The *synthetic accessibility* (*SA*) introduced by Ertl and Schuffenhauer [ES09] evaluates the synthesizability of molecules. For this purpose, on the one hand a fragment analysis of selected molecules from the PubChem database [Kim+16; Kim+21] is performed and on the other hand atypical chemical structures are evaluated with a complexity score. The difference between the fragment score and the complexity score gives the final *SA*, which ranges from 1 to 10. The lower the *SA*, the easier the

molecule is to synthesize. A typical example of two molecules with a high and low SA is shown in Fig. 5.1A.

Synthesis is the next step in typical drug development procedure, that follows the computer-assisted determination of the promising molecular candidates. Therefore, the SA is an important parameter to justify how complex the production of a potent drug molecule is in reality.

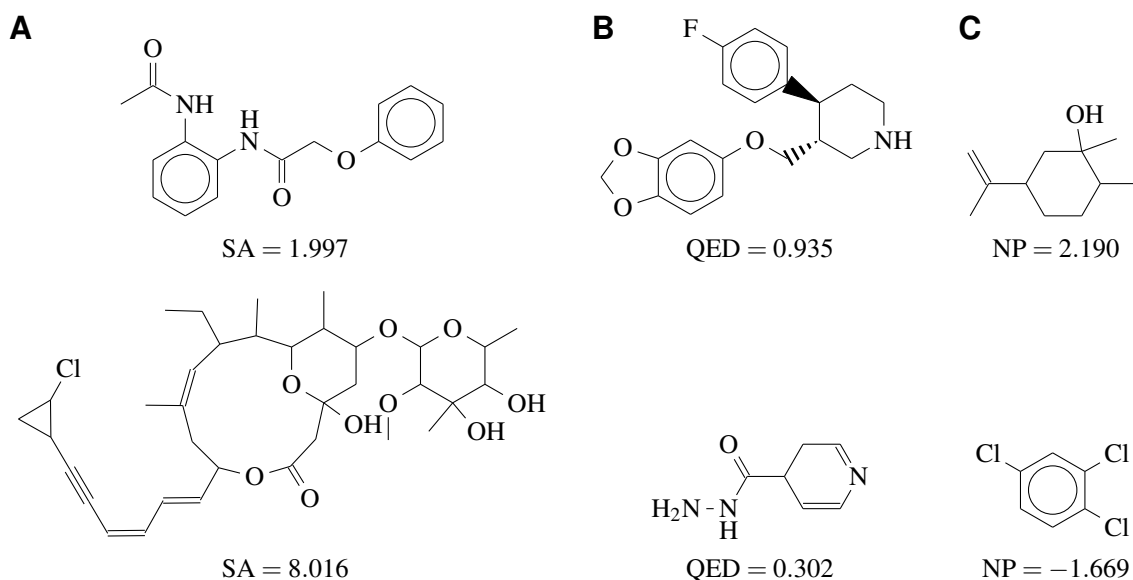


Figure 5.1: Exemplary molecules for high and low values of **A: SA**, **B: quantitative estimate of drug-likeness (QED)**, and **C: natural product-likeness (NP)**. The top row shows molecules with an optimal value compared to molecules in the bottom row. A low SA indicates that a molecule is easy to access synthetically [ES09]. A high QED or NP indicates that a molecule has a high similarity with drug-like molecules or natural products, respectively [Bic+12].

5.2.3 Quantitative Estimate of Drug-Likeness

Since many chemical properties for drug molecules are not randomly distributed, but share certain similarities, a comparison with already existing drug molecules is useful. A simple rule of thumb that evaluates oral bioavailability is Lipinski's rule of five [Lip+97]. Since this is a yes or no decision that is not always true, a more accurate metric with a continuous range of values would be more appropriate.

Therefore, we use the *quantitative estimate of drug-likeness (QED)* from Bickerton et al. [Bic+12] in the following. It estimates the similarity to existing drug molecules by taking into account the following properties, among others: the molecular weight, octanol-water partition coefficient, number of hydrogen bonds, and number of aromatic rings. The scores of the individual properties are finally combined to the QED with the value range from 0 to 1, where high values represent a more drug-like molecule. Figure 5.1B shows example molecules with a low and a high QED. A molecule with a high QED and thus a high similarity to existing drugs has a good chance of also possessing important properties of a drug.

5.2.4 Natural Product-Likeness

In addition to the previously described similarity to existing drug molecules, similarity to naturally occurring molecules is an important metric as well. Such molecules have been created and validated by nature in an evolutionary process. The *natural product-likeness (NP)* by Ertl, Roggo, and Schuffenhauer [ERS08] is used as a similarity measure for this purpose. It is based on the structural properties of the molecule, such as the distribution of nitrogen and oxygen atoms. NP has a range of values from -5 to 5 , with high values representing greater similarity to naturally occurring molecules. Figure 5.1 shows molecules with a high and low NP, respectively.

5.2.5 Toxicity Filters

Molecules that may be toxic due to their structure, e.g., because they contain isocyanate fragments or can lead to toxic metabolites, must be excluded or filtered out. Similarly, charged molecules should be avoided. This is the purpose of the *toxicity filters (TF)*, which combine several filters and output a value of 0 or 1, where 1 means that a molecule passes the filter and is therefore probably not toxic. TF combines the *pan assay interference compounds (PAINS)* filters by Baell and Holloway [BH10] and the *medical chemical filters (MCFs)* described by Polykovskiy et al. [Pol+20].

A good candidate molecule should have values close to the optimum for as many metrics as possible. To make it easier to compare the metrics with each other and to merge them into one overall metric, they can all be normalized to the same range of values, as described in more detail later in Section 5.3.3. Apart from BA, for which QuickVina 2 is used as mentioned above, for all other metrics (QED, NP, SA, and TF), the MOSES framework [Pol+20] is used. A more detailed overview of the workflow for calculating the metrics is given in the appendix in Appendix C.1.

5.3 Evolutionary Molecule Search

In this section, our evolutionary approach to design suitable protease inhibitor molecules is described. To explore the search space of biomolecules, an ES oriented ($\mu + \lambda$) population model is used (see Section 4.3). Here, the molecules are the individuals of a population and are encoded using the SELFIES representation (see Section 5.3.1). Based on this representation, specific mutation operators are defined (see Section 5.3.2). Finally, the molecules are evaluated by a fitness function (see Section 5.3.3) based on the previously described molecule design metrics (see Section 5.2). An overview of this process is given in Fig. 5.2.

First, the starting population is generated by concatenating a fixed number of random SELFIES symbols for each individual to describe a random molecule. This string is converted to a canonical SELFIES after a simple validity check and added to the set of molecules that have already been considered. If it already exists in this set, it is discarded to avoid duplication. The same procedure is used to generate new offspring. This creates a population of unique individuals.

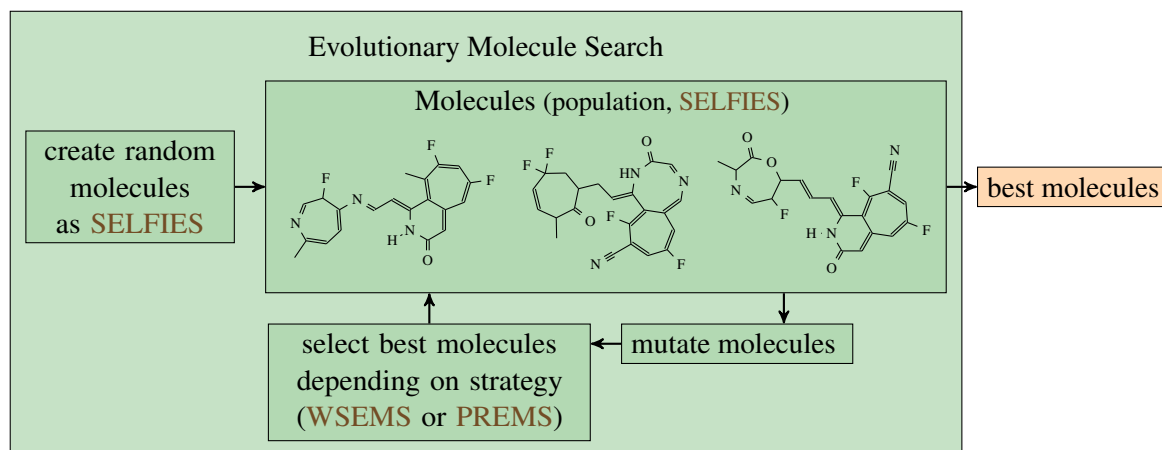


Figure 5.2: Overview of the evolutionary molecule search. After the creation of an initial population it's molecules are iteratively mutated and selected for the next generation. As a result, the best molecules under consideration are finally available in the last generation.

5.3.1 Representation

The molecules are described by **SELFIES** (see Section 4.2.3.1). The default symbols of **SELFIES** are [epsilon], [Ring1], [Branch1_1], [Branch1_2], [Branch1_3], [F], [O], [=O], [N], [=N], [#N], [C], [=C], [#C], [S], and [=S]. Benzene is a very common substructure, but as it consists of 8 **SELFIES** symbols ([C] [=C] [C] [=C] [C] [=C] [Ring1] [Branch1_1]), it is laborious to get there by chance. Therefore, this substructure is also considered as one symbol for the initial generation and the insertion mutation, that is described in Section 5.3.2. Each symbol has its own probability with which it is selected to account for the different abundance of atoms. This way, the weighing can be used to select more common symbols like [C] with a higher probability than e.g., branches or ring structures. By ensuring that the **SELFIES** representation can always be transformed into valid molecules, it is particularly suitable for random string creation and adaptations.

5.3.2 Mutation

The design space of molecules is explored by applying specific mutation operations with some randomness. An offspring is created by mutating a randomly chosen parent individual. The mutation consists of three different operations that are applied sequentially. Figure 5.3 shows an example molecule that is mutated by all three operations.

Deletion is applied with probability p_d and deletes a randomly chosen **SELFIES** symbol.

Replacement is applied independently for every symbol with a probability of p_r . The symbol is replaced by a random **SELFIES** symbol from the extended list described before weighted by a probability (see Section 5.3.1).

Insertion is applied with probability p_i . A random symbol is inserted at a random position in the individual's representation.

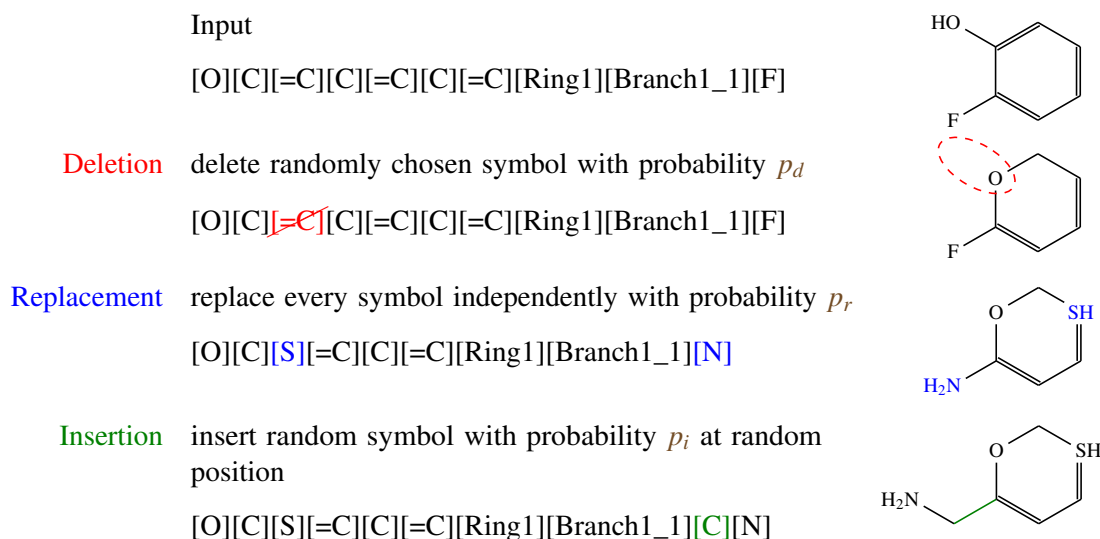


Figure 5.3: Example of the application of all three mutation operations in the specified order: first deletion, then replacement, and at last insertion. The changes in each step are highlighted. Note, that this is just an example to demonstrate all mutation operations at once and usually not all of them are executed, as their execution depends on the probabilities p_d, p_r, p_i .

5.3.3 Fitness Evaluation

The fitness value needed for the selection operator is determined by the fitness function $f(\mathbf{x})$, which takes into account the molecule's metrics (see Section 5.2). To allow easy comparison between the metrics, they are all scaled to the range $[0, 1]$, where 0 is the best value and 1 is the worst value. As the BA value is in \mathbb{R} it needs a special treatment. The BA is scaled to the experimentally chosen minimum of -15 kcal/mol and maximum of 1 kcal/mol and clipped to the range $[0, 1]$ using the soft clipping function [KP20] with $p = 30$ (see Eq. (5.1)). The soft clipping function and its application to the BA is visualized in Fig. 5.4.

$$SC_p(x) = \frac{1}{p} \log \left(\frac{1 + e^{px}}{1 + e^{p(x-1)}} \right) \quad (5.1)$$

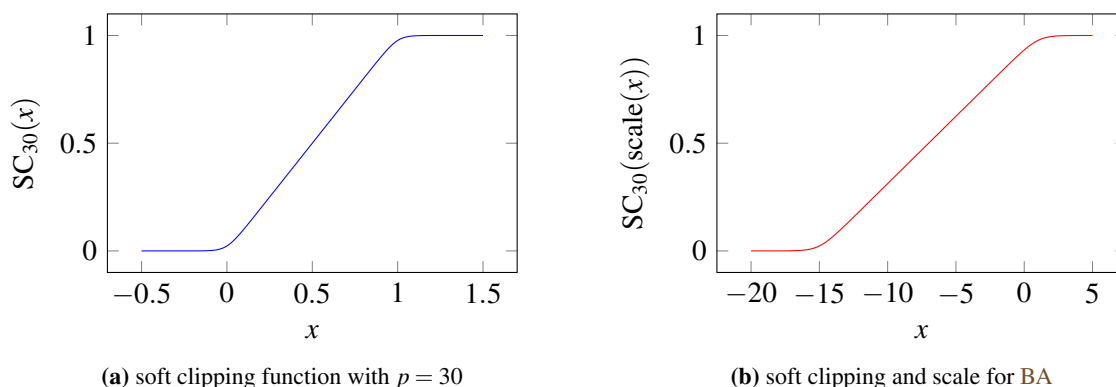


Figure 5.4: The soft clipping function (a) and its application to BA in combination with scale (b).

All individuals of one generation are evaluated concurrently, to speed up the run time of the GA. For the calculation of the metrics SA, QED, NP, and TF we use the framework MOSES [Pol+20]. For this purpose, the representation of SELFIES is converted to SMILES. The BA for each molecule is determined by QuickVina 2. Therefore, the SMILES representation is translated by RDKit¹ and MGLTools² to PDB and PDBQT files. As a specific case study we examine M^{pro} (PDB ID: 6LU7 [Jin+20])³ of SARS-CoV-2. For the calculation of BA, we use a search space of size⁴ 22 Å × 24 Å × 22 Å centered around the expected binding site (−12 Å, 15.6 Å, 69 Å). The exhaustiveness parameter of QuickVina 2 was kept at the default value of 8, resulting in several minutes of execution time per molecule.

5.3.4 Weighted Sum Evolutionary Molecule Search

Our first approach – *weighted sum evolutionary molecule search (WSEMS)* – uses a weighted sum to compose all $n = 5$ metrics into one value:

$$f(\mathbf{x}) = \sum_{i=1}^n w_i f_i(\mathbf{x}) \quad (5.2)$$

with weights $\mathbf{w} = (0.4, 0.15, 0.15, 0.15, 0.15)$ with i corresponding to 1: BA, 2: SA, 3: QED, 4: NP, and 5: TF. The weights were chosen based on a preliminary experiment with the aim of paying the most attention to the BA while considering the other properties as well.

5.3.5 Pareto Ranking Evolutionary Molecule Search

The previously described metrics (see Section 5.2) can behave in opposite ways. For example, a molecule may show good binding – i.e., have a good BA – but at the same time be dissimilar to other drug molecules – i.e., have a poor QED. Since it is difficult to choose pre-defined weights for the objectives, a Pareto ranking approach may be preferable in practice. Therefore, we introduce *Pareto ranking evolutionary molecule search (PREMS)* as our second approach.

A *multi-objective evolutionary algorithm (MOEA)* is a special kind of EAs that can be used to solve MOO problems [Smo13]. The *non-dominated sorting genetic algorithm II (NSGA-II)* [Deb+02] is a frequently used algorithm of this class in research and applications [Zho+11]. Our goal is to find the best molecules in the molecule space \mathcal{M} that minimize the fitness functions f_1, \dots, f_n . The NSGA-II is used to approximate the Pareto set $\{\mathbf{x}^* \mid \nexists \mathbf{x} \in \mathcal{M} : \mathbf{x} \prec \mathbf{x}^*\}$ of non-dominated solutions, where $\mathbf{x} \prec \mathbf{x}^*$ means \mathbf{x} dominates \mathbf{x}^* . To *dominate* in this context means to be at least as good in all properties (fitness functions) and to be better in at least one property: $\forall i \in \{1, \dots, n\} : f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$, while $\exists i \in \{1, \dots, n\} : f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$. Figure 5.5 visualizes the solution areas that dominate x , are dominated by x , and are incomparable to x for a two-dimensional minimization problem.

¹<https://www.rdkit.org>

²<http://mgltools.scripps.edu>

³PDB: protein data base, <https://www.rcsb.org>

⁴The angstrom (or ångström) (Å) is a metric unit of length, which is defined as 1 Å = 0.1 nm = 10^{−10} m. It is often used to express sizes of atoms and molecules.

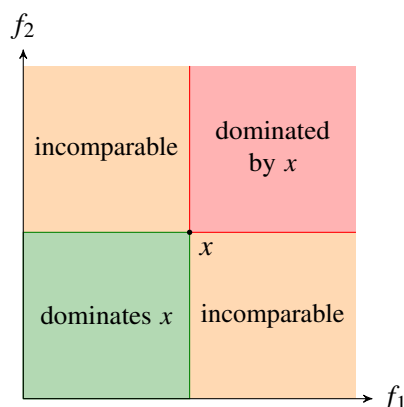


Figure 5.5: NSGA-II domination exemplified by a two-dimensional minimization problem. Given a solution x , it dominates the red area and is dominated by the green area. The orange areas are incomparable to x .

NSGA-II approximates the Pareto set with a broad distribution of solutions in objective space, i.e., of the Pareto front. After the non-dominated sorting, the μ non-dominated solutions maximizing the crowding distance are selected. This metric encourages individuals to spread out across the Pareto front, ensuring a diverse set of high-quality solutions. To compare various NSGA-II runs, we utilize the hypervolume indicator (S-metric) as a means to measure the dominated hypervolume in the objective space relative to a dominated reference point [ZT98].

The objective space consists of the five metrics outlined in Section 5.2. Although the fulfillment of TF is a binary criterion, it is treated as an objective. Since all objectives are determined computationally, they serve as approximations of the actual molecule properties. Molecules that perform poorly in one of the objectives may still exhibit potential as potent drug candidates or guide the algorithm towards unexplored regions of the search space.

5.4 Experiments

In this section, we conduct experimental analyses to evaluate the effectiveness of WSEMS and PREMS for the search of protease inhibitor candidates. We use the following settings for our experimental analyses. For WSEMS, we employ a (10 + 100)-EA, where in each generation, from 10 parents 100 offspring candidate molecules are generated by mutation. The mutation operators introduced in Section 5.3.2 are applied, with mutation probabilities set to $p_r = 0.05$, $p_i = 0.1$, and $p_d = 0.1$. Plus selection is used to get the best candidates from the parents and offspring. In the PREMS approach, we increase the number of parents to 20 in order to obtain a broader distribution of solutions in the objective space⁵. Crossover is not applied in this approach. To ensure consistency and comparability, individuals in both approaches are limited to a maximum length of 80 SELFIES tokens. This constraint is oriented to the approach proposed by Krenn et al. [Kre+20]. In terms of termination criteria, all runs are conducted for a fixed number of 200 generations, and the experiments are repeated a total of 20 times to obtain reliable statistical results.

⁵The total number of fitness function calls is almost unaffected by this. Only for the generation of the initial population, 10 more molecules are evaluated.

5.4.1 Metric Development

The development of the normalized metrics, as explained earlier, is depicted in Fig. 5.6. Both **WSEMS** and **PREMS** runs are analyzed in this figure.

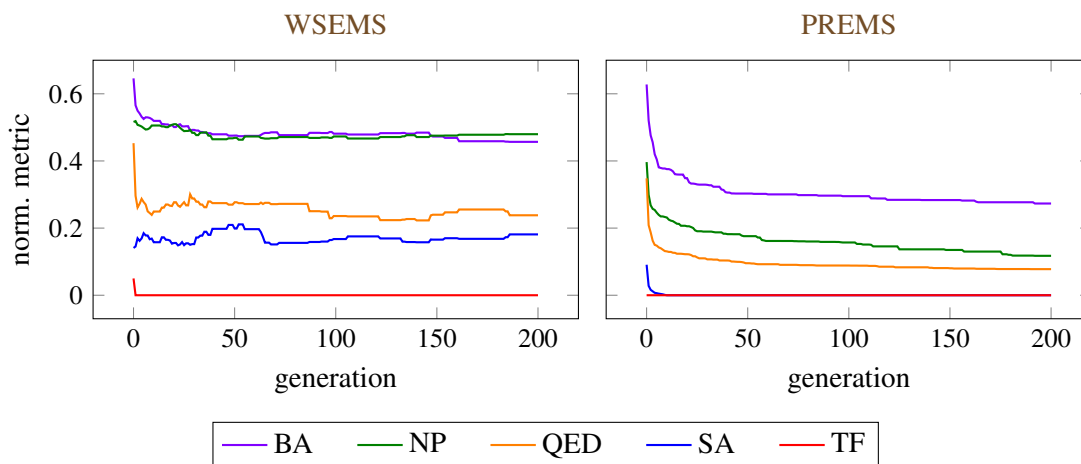


Figure 5.6: Development of all Metrics. On the left side the normalized values of the five metrics is shown for **WSEMS** and on the right side for **PREMS**. The results are averaged over 20 runs.

In the **WSEMS** runs, the best individuals based on fitness are selected in each generation, and their corresponding metrics are averaged across all runs. The optimization process primarily focuses on enhancing **BA**, **QED**, and **NP**. It is worth noting that the improvement of one metric may lead to a deterioration of another metric. For instance, starting from generation 140, there is a trade-off observed as **QED**, **NP**, and **SA** deteriorate in favor of **BA**.

In the case of **PREMS** runs, the best individuals for each metric are chosen in each generation, and their metrics are subsequently averaged across all runs. A consistent improvement is observed for all objectives in these runs. However, it is important to acknowledge that achieving this improvement in certain objectives may come at the expense of deteriorations in other objectives, which are not explicitly shown in the figure.

Figure 5.7 exhibits three distinct two-dimensional sections of the Pareto front, where the **BA** is compared to **QED**, **NP**, and **SA**. For each tenth generation, a Pareto front is displayed, with colors ranging from low contrast for the initial generation to high contrast for the final generation. The plots effectively demonstrate how **PREMS** generates solutions that vary in their balance between the **BA** and the metric being plotted. Throughout the optimization process, the front of non-dominated solutions consistently tends to shift towards the lower left, as expected. This tendency is also reflected in the hypervolume indicator, which, on average across all runs, improves from 0.10 ± 0.03 in the first generation to 0.20 ± 0.05 in the last generation. In the slice plots, it is possible to observe deteriorations due to advancements made in the remaining three objectives.

Table 5.2 presents a comparison between the final experimental results of the **WSEMS** optimization runs and the **PREMS** runs. In the case of the **PREMS**, the table showcases the best values achieved for each objective (in the original value range), which correspond to the corner points of the Pareto front approximation. To provide a reference, metric values are also displayed for N3, a ligand proposed

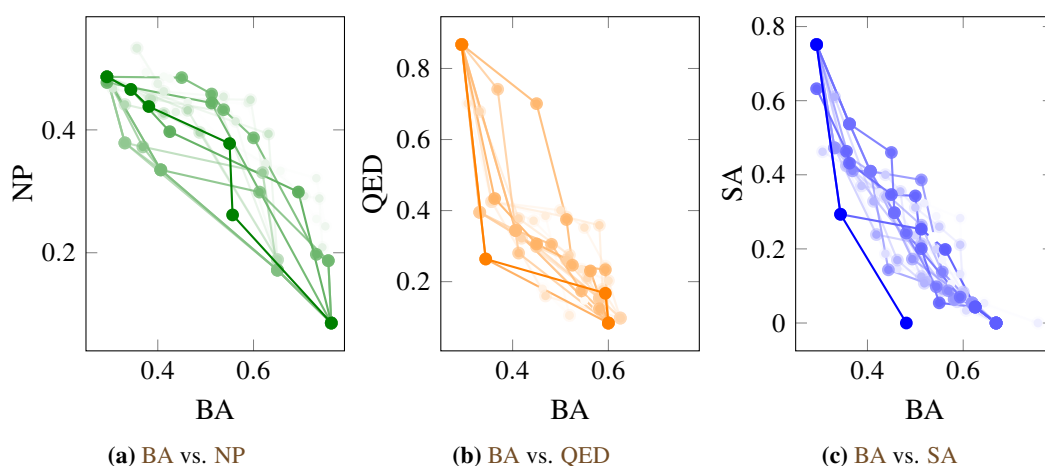


Figure 5.7: Visualization of Pareto fronts over the generations a single **PREMS** run. For every 10th generation, a Pareto front is displayed, with the colors starting from low contrast for the first generation and gradually transitioning to high contrast for the final generation.

in the **PDB**, and Lopinavir, an inhibitor of the **human immunodeficiency virus (HIV)** main protease [KH05]. The **BA** attained through the **WSEMS** optimization process demonstrate that the best values even surpass those of N3 and Lopinavir. Both Lopinavir and N3 exhibit similar strong binding to **M^{pro}**. **PREMS** achieves promising values for all metrics, offering a broad coverage of objective function values and presenting practitioners with a wide range of intriguing candidates. However, it is important to note that some of the extreme metric values may occasionally be impractical. For instance, the exceptional **BA** of the top **PREMS** molecule (-13.3 kcal/mol) has been attained by a chemically unrealistic candidate.

Table 5.2: The experimental results include the outcomes of **WSEMS**, as well as the best values per objective achieved by **PREMS**. Additionally, the N3 ligand (from **PDB 6LU7**) and Lopinavir, a prominent drug candidate, are included in the analysis. For the **PREMS** method, statistical evaluation is conducted based on the best 20 individuals per objective. In the results, a ▼ symbol represents a minimization objective, while a ▲ symbol indicates a maximization objective.

objective	WSEMS		PREMS		N3	Lopinavir
	best	avg±std	best	avg±std	value	value
fitness ▼	0.30	0.32±0.01	0.31	0.39±0.06	0.43	0.41
BA ▼	-9.30	-7.68±0.90	-13.30	-10.63±1.18	-8.40	-8.40
SA ▼	3.04	2.63±0.59	1.00	1.00±0.00	4.29	3.90
QED ▲	0.66	0.76±0.10	0.94	0.92±0.01	0.12	0.20
NP ▲	0.33	0.20±0.54	4.27	3.82±0.24	-0.18	-0.04
TF ▲	1.00	1.00±0.00	1.00	1.00±0.00	1.00	1.00

Based on our observations, we can conclude that utilizing the **SELFIES** representation along with our mutation operators enables the generation of molecules with a consistent level of quality. Nevertheless, we anticipate that incorporating mechanisms that facilitate the creation of larger molecules would lead to further improvements in the quality of the results by overcoming fitness plateaus and local optima.

In Fig. 5.8, we present a comparison of the populations from the last generation of a typical **WSEMS** run and a **PREMS** run. It is evident that the solutions in the **WSEMS** population exhibit a high degree of similarity to each other. However, the solutions in the final population of the **PREMS** run maintain a higher diversity in terms of molecule properties.

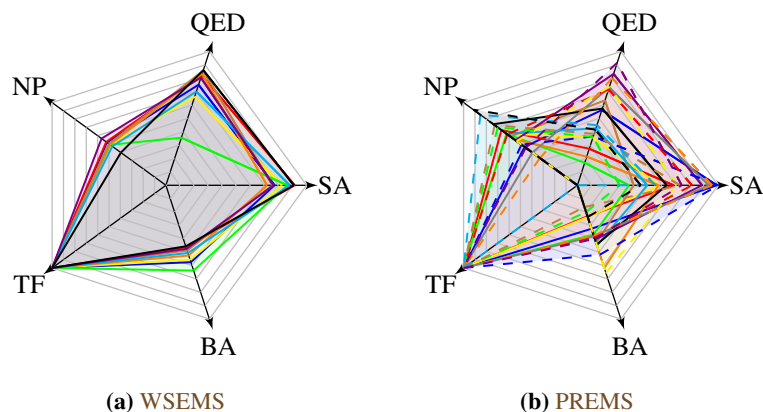


Figure 5.8: Comparison of the population in the final generation between exemplary **WSEMS** runs (10 molecules) and **PREMS** runs (20 molecules). Each line represents a molecule candidate. Note that good values are at the edge of the spider chart.

5.4.2 Candidate Comparison

In the following, we present a selection of interesting protein inhibitor candidates that were evolved using both **WSEMS** and **PREMS** approaches. We made three notable observations through our experiments:

1. The generated molecules tend to incorporate aromatic ring structures prominently.
2. Candidates with favorable drug-likeness properties are typically shorter in size.
3. Molecules with high **BA** often exhibit unrealistic geometries.

Figure 5.9 showcases a compilation of six promising **protease inhibitor (PI)** candidates, displaying radar plots, structural formulas, and chemical names. PI-I (a) to PI-III (c) are the results obtained from **WSEMS** runs, while PI-IV (c) to PI-VI (f) represent candidates generated by **PREMS**. In the radar plots, points positioned closer to the plot's border indicate superior values, with zero values located on the edge.

All candidates fulfill the filter condition, indicating their viability. PI-1 achieves a high **SA** value alongside a reasonable **BA**. PI-II attains an excellent **BA** of -9.7 kcal/mol. PI-III, PI-IV, and PI-VI demonstrate excellent drug-likeness as measured by **QED**, accompanied by solid docking results around -7.0 kcal/mol. An intriguing candidate that strikes a balance across all objectives is PI-V, which achieves a **BA** of -7.7 kcal/mol and a **QED** value of 0.75.

Lastly, we provide a visualization depicting how the ligand candidates are situated within the optimized M^{pro} protein pocket, performed using **QuickVina 2**. Figure 5.10 illustrates the placement of candidates (a) PI-I and (b) PI-V within their respective M^{pro} pockets.

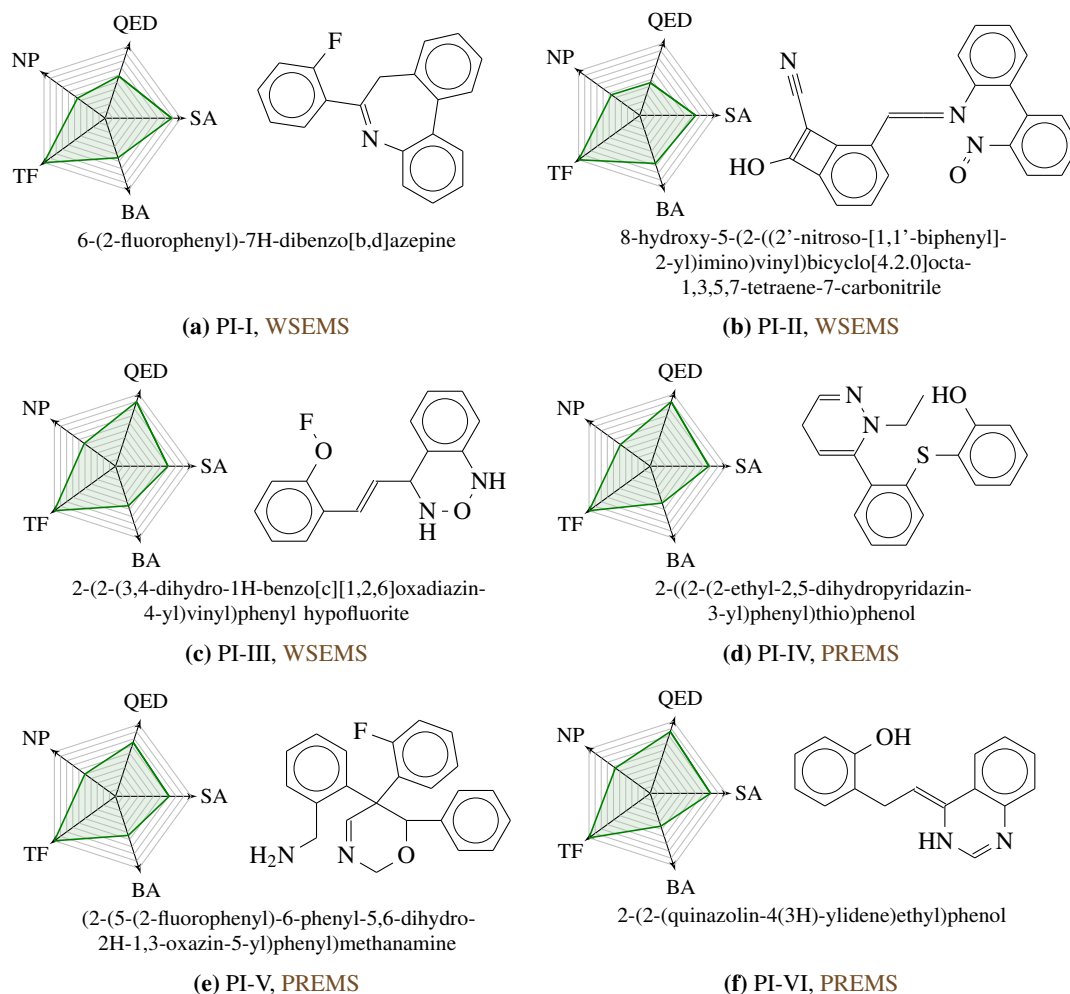


Figure 5.9: Spider charts of exemplary protease inhibitors, along with their structural formula, and chemical name. The results are from runs of WSEMS (a–c) and PREMS (d–f).

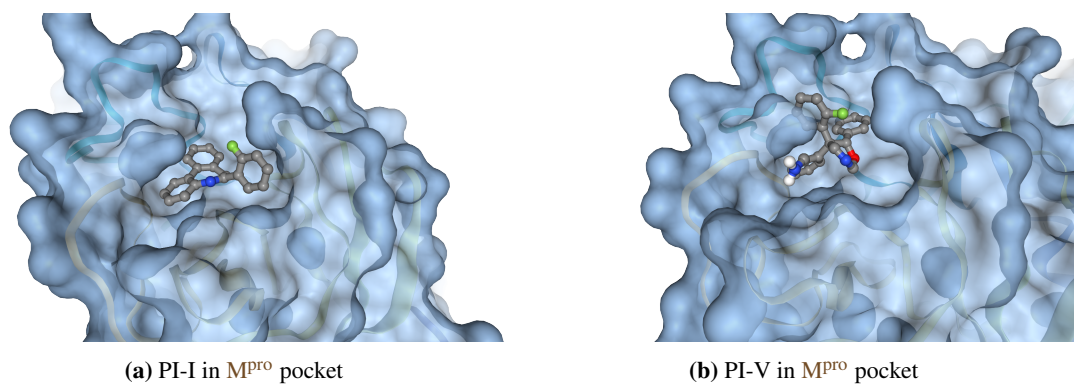


Figure 5.10: The docking of PI-I and PI-V to the pocket of SARS-CoV-2's M^{pro} [Cof+20].

5.5 Conclusion

In this chapter, an evolutionary multi-objective approach was introduced to evolve protein inhibitor candidates for the M^{pro} of SARS-CoV-2. We applied two strategies for the fitness evaluation and selection process (WSEMS and PREMS). Our approach serves as a starting point for drug design efforts, with the goal of optimizing the QuickVina 2-based protein-ligand binding score (BA), as well as other significant objectives like QED and TF properties. The experimental results demonstrated the capability of the evolutionary processes to generate intriguing inhibitor candidates. While many of these candidates achieved promising metrics using conventional structures, unconventional candidates also emerged that require deeper analysis. Due to potential limitations in the practical utility of QuickVina 2 BA and other metrics, the approach is understood as an AI-assisted virtual screening technique for exploring the chemical biomolecule space. The comparatively fast evaluation of the molecules results in a significant time saving compared to traditional methods and, moreover, allows far more molecules to be considered in the first place.

Future research will be directed towards enhancing the protein-ligand models to enable more detailed and efficient binding affinity predictions. Additionally, there is potential for improvement in the SELFIES representation, particularly addressing bloated strings that represent relatively small molecules, and developing mechanisms to ensure their validity. Furthermore, the utilization of additional multi-objective evolutionary algorithms is desired.

6 Language Model–based Evolutionary Approach

Expanding on the previous Chapter 5, a further and extended procedure for protease inhibitor design – as motivated in Section 4.1 – is presented here. An evolutionary algorithm is used here that employs a neural language model for initialization and mutation. This is referred to as *evolutionary molecule generation algorithm (EMGA)*. The process is complemented by subsequent manual selection and molecular dynamics (MD) simulation, which can more precisely evaluate the binding between the ligand and the protease enzyme to determine whether the ligand remains in the desired position or moves away over time. An overview of this method is provided in Fig. 6.1. In summary, our workflow consists of EMGA – a combination of EA and language model (LM) – as well as MD simulation and analysis of the resulting molecular candidates.

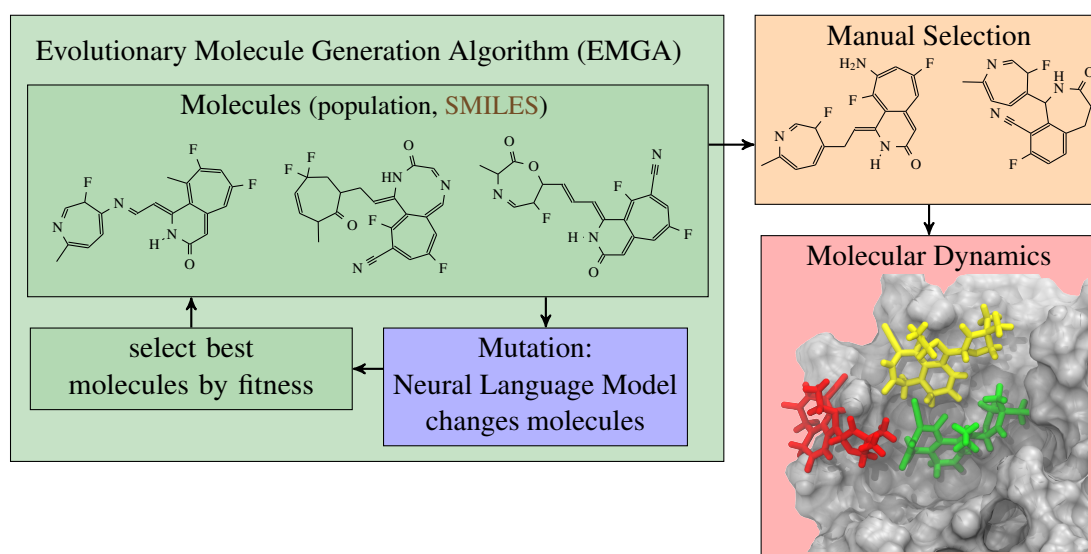


Figure 6.1: Overview of the language model–based evolutionary approach. The EMGA (green) iteratively mutates a population of molecules with a neural language model (blue). After each generation the best molecules are selected based on a fitness metric. By a manual selection (orange) the best overall molecules are selected for further analysis with molecular dynamics simulations (red).

The proposed setup is intended to combine the advantages of the two sub-methods: Large-scale exploration through fast AI methods and high simulation accuracy through expensive MD simulations. Depending on which molecules are considered, computer-aided drug design (CADD) approaches generally fall into the following two categories:

1. Only existing molecules from large databases, such as ZINC database [SI15] or the DrugBank database [Wis+18], are used.
2. New molecules are generated based on existing molecular data sets.

A comprehensive overview of CADD for the discovery of protein inhibitors against SARS-CoV-2 is provided by Liu, Wan, and Wang [LWW22], which provides numerous examples of both of the aforementioned approaches. Numerous research papers adopt known molecules to evaluate their viability as inhibitors of SARS-CoV-2 using techniques such as docking and MD simulations [Bha+21; Sha+21; Sin+21]. Arshia et al. [Ars+21] explored an approach that combines AI with subsequent MD simulations to design candidate protease inhibitors against SARS-CoV-2. Their study involved the use of an LSTM neural network to generate novel potential drug molecules, with a primary focus on optimizing binding affinity as a key metric. The approach in this chapter, just like the one in the previous one, falls into category 2. Note that further related work on drug design using AI methods has already been discussed in Section 5.1.

This chapter is structured as follows. First, EMGA is introduced in Section 6.1, with particular reference to the Molecular Representation, the Neural Language Model, and its integration into an Evolutionary Algorithm described. Then, the foundations for the MD calculations are defined. In Section 6.3, the results of EMGA and the MD analysis are discussed. Finally, a summary is given in Section 6.4.

Parts of this chapter are based on the following published paper:

Lars Elend, Luise Jacobsen, Tim Cofala, Jonas Prellberg, Thomas Teusch, Oliver Kramer, and Iliia A. Solov'yov. "Design of SARS-CoV-2 Main Protease Inhibitors Using Artificial Intelligence and Molecular Dynamic Simulations". In: *Molecules* 27.13 (13 Jan. 2022), p. 4020. ISSN: 1420-3049. DOI: [10.3390/molecules27134020](https://doi.org/10.3390/molecules27134020)

Note that the MD analysis results in Section 6.3.2 are mainly based on the work of Luise Jacobsen in the context of the publication mentioned above.

6.1 Evolutionary Molecule Generation Algorithm

The core component of EMGA is again an ES oriented ($\mu + \lambda$) population model as introduced in Section 4.3. Here the individuals – the molecules – are encoded as SMILES strings (see Section 6.1.1). A LM is trained to fill gaps in the SMILES string in a meaningful way (see Section 6.1.2). The LM is then combined into the ES as a mutation operation (see Section 6.1.3). For the selection the same fitness function Eq. (5.2) is used as before (see Section 5.3.3). It is based on the five molecule design metrics that were introduced in Section 5.2.

6.1.1 Representation

In contrast to previous approach (see Chapter 5) EMGA is based on the SMILES string representation (see Section 4.2.3.1). This decision is based on two related reasons:

1. While the **SELFIES** representation ensures that only valid molecules are ever described, this often involves ignoring the trailing part of the string, from the point that would lead to a faulty molecule. Thus, in such cases, mutations of the posterior substring no longer affect the molecule, which is not ideal.
2. The **SMILES** representation has no such limitation – but also offers no guarantee of valid molecules. In the previous approach within the **EA**, random mutations were made to the molecule, which could easily result in defective molecules. In the **EMGA**, on the other hand, a **LM** is used to make context-based changes to the string. Thus, with an **LM** trained on valid molecules, lower error rates are expected.

Figure 6.2 shows as an example the structural formula of the α -D-glucopyranose and the corresponding **SMILES** string. Each letter of this string can be considered as a token by the neural language model that will be introduced later.

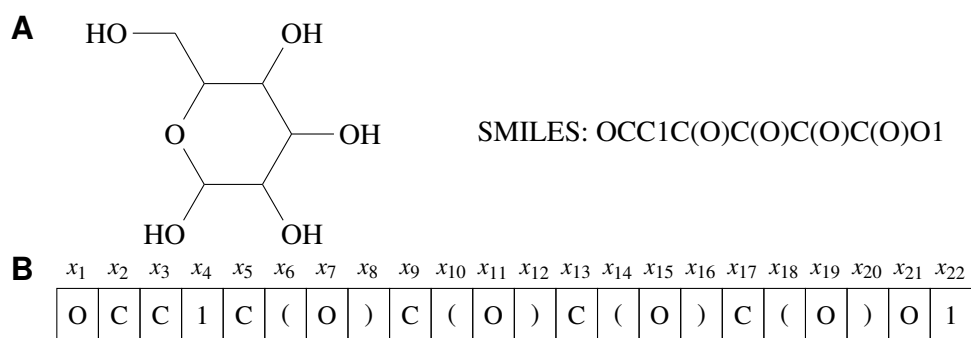


Figure 6.2: **A:** Structural formula and **SMILES** string of α -D-glucopyranose. **B:** **SMILES** string of α -D-glucopyranose split into a sequence of tokens $\mathbf{x} = (x_1, \dots, x_t)$.

6.1.2 Neural Language Model

The process of generating new and realistic drug molecules can be facilitated by **AI**-based molecular generation models [Sch18]. Consequently, in order to discover more drug-like molecules as anticipated, a molecular generation model was incorporated into **EMGA**. The molecular generation model utilized in this implementation was built upon the Transformer architecture [Vas+17]. The network architecture was specifically crafted to handle sequential data and incorporates a distinctive and inherent attention mechanism. The model underwent training using a set of known molecules, aiming to generate molecules with comparable properties using this data set.

Given that the molecules in this study are initially represented in a textual representation, the implementation of a molecular structure generation model draws inspiration from concepts in language processing. A *language model (LM)* processes a sequence of tokens $\mathbf{x} = (x_1, \dots, x_t)$. For each token position t , the model can predict a probability distribution over the possible tokens in the sequence, considering the other positions in the sequence.

An example of such an approach is provided by Segler et al. [Seg+18], who demonstrated the use of a recurrent neural network to generate molecules in the **SMILES** representation. In our study, a token refers to the smallest constituent of a **SMILES** string, such as a letter, bracket, number, or equal sign.

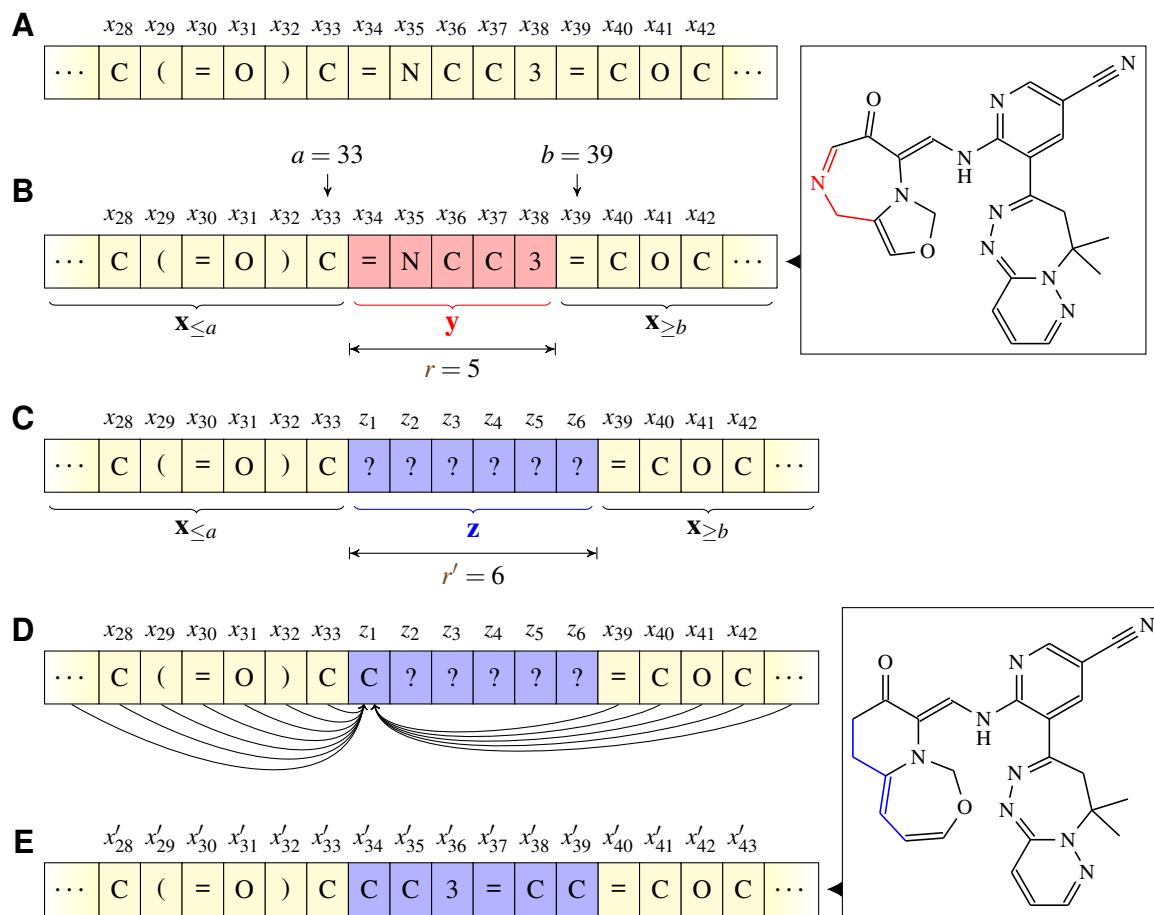


Figure 6.3: The language model as a mutation operator does the following steps: **A:** A SMILES string to be mutated. **B:** A random range \mathbf{y} (highlighted in red) of size r is selected for replacement. The molecular structure on the top right corresponds to the SMILES string with \mathbf{y} highlighted. **C:** The language model generates a new sequence \mathbf{z} (highlighted in blue) of length r' . It is important to note that r' does not necessarily have to be equal to r . **D:** The language model iteratively calculates the values z_i . For each z_i , the input includes all $\mathbf{x}_{\leq a}$, $\mathbf{z}_{<i}$, and $\mathbf{x}_{\geq b}$ values. **E:** After the language model processing, the resulting SMILES string becomes $\mathbf{x}' = (\mathbf{x}_{\leq a})\mathbf{z}(\mathbf{x}_{\geq b})$. The molecular structure on the bottom right corresponds to the mutated SMILES string, with the mutated part highlighted in blue.

The SMILES string itself represents the sequence, as shown in Fig. 6.2B. The language model was trained to predict new molecules by considering the tokens. The training process involved iterative observation of a molecule set, updating the model parameters to predict corresponding probability distributions. To enable iterative sampling of new molecules, the generation model was trained with an autoregressive objective, meaning that the probability of the next token (e.g., letter, bracket) depends on the previous tokens. More formally, given a sequence of tokens describing a molecule, the likelihood function for the molecule can be factorized into conditional probabilities as

$$p(\mathbf{x}) = \prod_{i=1}^t p(x_i | x_{<i}). \quad (6.1)$$

In this context, \mathbf{x} represents a sequence of tokens, where t denotes the maximum number of tokens in \mathbf{x} . The notation $x_{<i}$ refers to all tokens in the sequence that appear before the index i .

In **EMGA**, the neural language model served as a mutation operator, allowing the modification of existing molecules. To achieve this, the training objective of the language model was adjusted to enable the completion of contiguous parts at any position within a **SMILES** string. Given a sequence of tokens \mathbf{x} with a prefix $\mathbf{x}_{\leq a}$ and a suffix $\mathbf{x}_{\geq b}$, where $a < b$, a new sequence $\mathbf{z} = (z_1, \dots, z_d)$ of length d could be sampled. This sampling aimed to ensure that $(\mathbf{x}_{\leq a})\mathbf{z}(\mathbf{x}_{\geq b})$ formed a valid **SMILES** string according to the modeled distribution. This process is illustrated in Fig. 6.3. To maximize the likelihood of generating realistic molecules, the special Transformer architecture called XLNet [Yan+19] was used. This architecture allows the model to consider all permutations of the factorization order, unlike training solely on a left-to-right factorization order.

The neural language model was trained on a subset of the **ZINC** database [SI15], which contains a collection of purchasable molecules. The molecules in the subset adhered to the definition of a drug-like molecule outlined by Polykovskiy et al. in their molecular generation benchmark paper **MOSES** [Pol+20]. This resulted in a data set comprising 1.9 million molecules.

6.1.3 Evolutionary Algorithm with Language Model

Figure 6.4 provides an activity diagram illustrating the workflow of **EMGA**. The process begins with the neural language model generating a population of molecules by sampling new **SMILES** strings. These strings are generated character by character, ensuring a diverse set of starting molecules. It is also possible to start with parts of already known structures to guide the evolution in a specific direction. The language model is trained on the **ZINC** database, so the generated molecules should resemble **ZINC** molecules and be chemically reasonable. After the generation phase, each individual in the population is evaluated using the fitness function to assess its quality. Next, λ individuals are

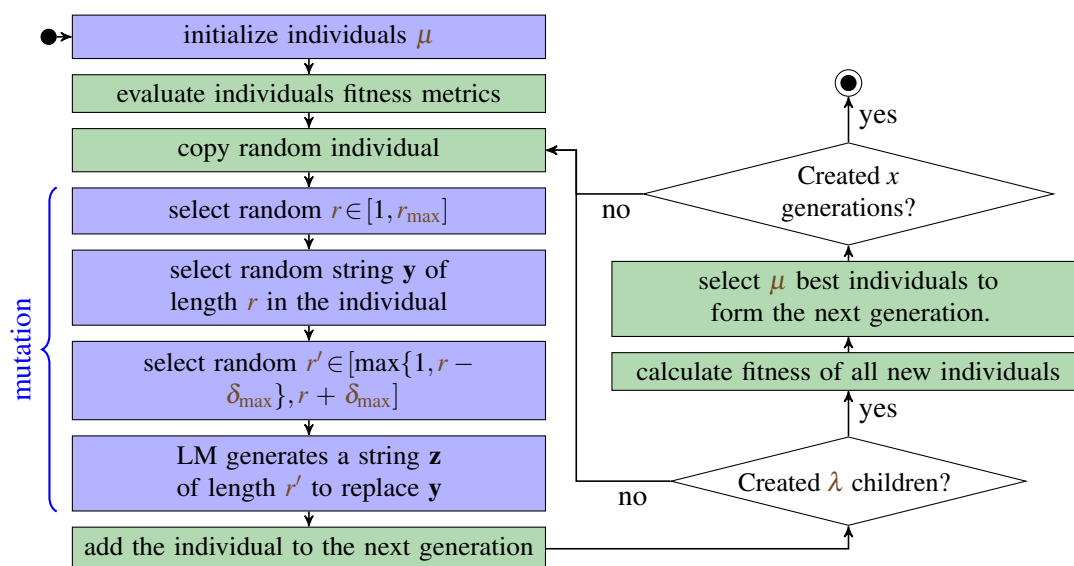


Figure 6.4: The flow of **EMGA** as an activity diagram. As in the overview in Fig. 6.1, the actions of **EMGA** are generally shown in green and those of the Language Model in blue.

created by mutating random individuals from the initial population (parents). The mutation process involves replacing a random part of the SMILES string of a molecule with a new string generated using the neural language model (see Fig. 6.3). The maximum length of the replaced string is determined by the parameter r_{\max} . The length of the new string may vary compared to the original length r , but it cannot exceed $r + \delta_{\max}$, where δ_{\max} is an offset parameter. The balance between exploring the search space and exploiting well-performing molecules is controlled by the values of r_{\max} and δ_{\max} . Higher values of r_{\max} and δ_{\max} can lead to more diverse molecules, but may also result in individuals that are significantly different from their parents. Conversely, smaller values of r_{\max} and δ_{\max} allow for fine adjustments of well-performing individuals but increase the risk of getting stuck in local minima. In the presented study, r_{\max} and δ_{\max} were set to 8 and 5, respectively, to strike a balance between exploration and exploitation. Algorithm 1 shows a pseudocode for EMGA.

Algorithm 1 EMGA pseudocode

```

1:  $Pop \leftarrow \text{LM.sample\_autoregressive}()$ 
2: repeat
3:   for all  $I \in Pop$  do
4:      $Fit_I \leftarrow \text{calculate\_metrics}(I)$ 
5:   end for
6:   select  $\mu$  parents from  $Pop$  based on  $Fit$ 
7:    $Pop_{\text{new}} = \emptyset$ 
8:   repeat ▷ Mutation
9:     select random parent  $P$ 
10:     $r \leftarrow \text{random}[1, r_{\max}]$ 
11:     $r' \leftarrow \text{random}[\max\{1, r - \delta_{\max}\}, r + \delta_{\max}]$ 
12:     $I_{\text{new}} \leftarrow \text{LM.replace}(P, r, r')$ 
13:    add  $I_{\text{new}}$  to  $Pop_{\text{new}}$ 
14:   until  $\text{size}(Pop_{\text{new}}) = \lambda$ 
15:    $Pop \leftarrow \text{parents} \cup Pop_{\text{new}}$ 
16: until termination condition

```

6.2 Molecular Dynamics

After EMGA first evaluates the protease inhibitor candidates based on the relatively quickly calculated metrics (see Section 5.2), a much more detailed simulation of the binding between the ligand and the protease is performed as part of the *molecular dynamics (MD)* analysis. The various physical forces that affect the positioning of the ligand are calculated. The physical simulation is then run for a small time period of 50 ns. In this way it can be investigated whether the ligand maintains the desired position within the ligand or moves away from it and thus does not bind. The physical energies and formulas relevant to this simulation are presented in the following.

After generating potential drug molecules using EMGA, it is possible to further assess them by evaluating the inhibitor binding free energy, which can be determined using the *molecular mechanics/generalized Born surface area (MM/GBSA)* method. In this approach, the binding free energies G^0 were computed as:

$$\Delta G^0 = \langle G_C \rangle_C - \langle G_R \rangle_R - \langle G_L \rangle_L, \quad (6.2)$$

where G_L , G_R , and G_C represent the free energies of the ligand (L), receptor (R), and ligand-receptor complex (C), respectively. The symbol $\langle \cdot \rangle$ denotes an average over a corresponding MD simulation trajectory carried out specifically for the L, R, or C, as described in [Kol+00]. These MD simulations should be performed at the atomistic level after identifying suitable drug candidates from the EMGA calculations. The individual free energies in Eq. (6.2) can be calculated as:

$$G_i = (E_{MM} + G_p + G_{np} - TS)_i, \quad (6.3)$$

where for a selected subsystem $i \in \{L, R, C\}$, E_{MM} represents the non-bonding molecular mechanics energies, G_p and G_{np} are the polar and non-polar solvation free energies of the i th subsystem, respectively, and TS accounts for the free energy associated with the entropy S of the subsystem at temperature T . The dependency of G_{np} on the solvent-accessible surface area, A , of the subsystem and the surface tension parameter $\gamma = 6 \times 10^{-4}$ kcal/(mol Å²) can be described as [Bri+17; GC04]:

$$G_{np} = \gamma A. \quad (6.4)$$

In the calculation of G_p contributions in Eq. (6.3), the generalized Born (GB) model, based on a modified version of Still et al.'s method [Sti+90], was employed to account for the ionization of the solvent [Sri+99; OC19; Ber+20]:

$$G_p = -k_e \sum_{i=1}^N \sum_{j=i}^N \frac{F_{ij} q_i q_j}{g_{ij}}, \quad (6.5)$$

where k_e is the Coulomb constant and the summations are performed over the N atoms in the corresponding subsystem (L, R, or C). The dielectric term is defined as $F_{ij} = \left(1 - \frac{\exp(-\kappa g_{ij})}{\epsilon_s}\right)$, where the dielectric constant of the solvent is $\epsilon_s = 74$. The Debye screening length is $\kappa^{-1} = \sqrt{\frac{\epsilon_0 k_B T}{2 N_A e_c^2 I}}$, with Boltzmann constant k_B , Avogadro number N_A , elementary charge e_c , ion concentration $I = 0.15$ M, and vacuum permittivity ϵ_0 [Ber+20; OBC00]. The function g_{ij} appearing in Eq. (6.5) was suggested by Still et al. [Sti+90] and describes the effective distance between the atoms i and j :

$$g_{ij} = \sqrt{r_{ij}^2 + \alpha_i \alpha_j \exp\left(\frac{-r_{ij}^2}{4\alpha_i \alpha_j}\right)}, \quad (6.6)$$

where r_{ij} is the distance between two particles i and j . The effective Born radius α_i indicates how deep an atom is buried inside a molecule or a protein [OBC04; OBC00], and can be computed following Onufriev, Bashford, and Case [OBC00; OBC04; Ber+20]. The GB method treats the solvent as a continuum, which can result in lower accuracy compared to simulation models that incorporate explicit solvent molecules. Additionally, the GB method's performance may vary depending on the system being studied, such as underestimating α_i for atoms deeply embedded within macro-molecules [OBC04]. However, since the free energies of binding are calculated for the same receptor in the present problem, a qualitatively accurate relative comparison can be expected.

The entropy term in Eq. (6.3) was estimated using Schlitter's quasi-harmonic approach [Sch93], which provides an upper bound to the entropy as

$$S \lesssim \frac{1}{2} k_B \ln \det \left[\mathbf{I} + \frac{k_B T e_c^2}{\hbar^2} \mathbf{M} \boldsymbol{\sigma} \right], \quad (6.7)$$

where \mathbf{I} is the identity matrix and \hbar represents the reduced Planck's constant. The matrix \mathbf{M} contains the atomic masses of the subsystem on its diagonal and is zero elsewhere. The covariance matrix $\boldsymbol{\sigma}$ is obtained from the MD trajectory, encompassing the 3N Cartesian coordinates describing the atoms in the given subsystem:

$$\sigma_{ij} = \langle (\xi_i - \langle \xi_i \rangle) (\xi_j - \langle \xi_j \rangle) \rangle, \quad (6.8)$$

where ξ_i represents the x -, y -, or z -coordinate of an atom. For practical entropy calculations of the receptor, it is convenient to consider approximately 100 non-hydrogen atoms surrounding the ligand, as the inclusion of further atoms makes the calculation computationally too expensive.

The AI-MD approach, although applicable to various systems, was demonstrated using M^{pro} from SARS-CoV-2 as a case study. The specific details of the conducted MD simulations are as follows. The MD simulations were initiated using the ligands with the highest fitness scores designed from EMGA. Hydrogen atoms were incorporated into the ligands using the Open Babel package [OB+11] at a pH value of 7.4. The addition was applied to the poses generated by QuickVina 2. Subsequently, the ligand structures underwent minimization using the conjugate gradient algorithm until a convergence criterion of 10^{-6} was met. For the simulations involving the protein-ligand complex, the minimized ligand structure was reintegrated into the receptor in the pose obtained through docking. The M^{pro} protein was modeled using the Amber ff14SB force field [Tia+20], while the ligands were modeled using the general Amber force field [Wan+04]. The force fields were prepared using AmberTools [Cas+]. NAMD 2.14 [Phi+05; Phi+20], along with its generalized Born implicit solvent (GBIS) functionality, which provides solvation free energy and electrostatic energy output, was employed to conduct the simulations. The simulation analysis was carried out using the MDAnalysis python library [Mic+11].

In each simulation for L, R, and C, a total of 10000 minimization steps were performed, followed by a 50 ns simulation in implicit solvent. The time step used was 1 fs. For the calculation of van der Waals (vdW) and short-range electrostatic interactions in the presence of GBIS, a cutoff distance of 16 Å with a switching distance of 15 Å was employed, as recommended in the NAMD user guide [Ber+20]. To maintain a constant temperature of 310 K throughout the simulations, the Langevin thermostat [Brü92] was utilized. A damping coefficient of 5/ps was applied in the Langevin thermostat scheme.

6.3 Results and Discussion

The AI-MD algorithm described above, was used to generate and select potent drug molecules. The method was applied as an illustrative case study to M^{pro} of SARS-CoV-2. In the following the results of EMGA are presented. The 21 most promising molecules were selected and further analyzed with a MD simulation.

6.3.1 Evolutionary Molecule Generation Algorithm

Figure 6.5 shows fitness score of the best performing individual in each generation. It is calculated over 15 runs of **EMGA** and therefore shows the mean value and the standard deviation. A low fitness score indicates for a more suitable inhibitor of M^{pro} . The plot demonstrates the optimization process of **EMGA**, which led to the improvement of the initial individuals. However, this optimization progress reached a point of stagnation after approximately 70 generations, with the best performing molecule achieving a fitness score of 0.225. In the later generations, notable advancements were observed in terms of metrics, specifically for **BA** (-11.8 kcal/mol), **QED** (0.954), **NP** (0.372), and **SA** (1.0). Throughout the course of the 15 independent runs of **EMGA**, a total of 120300 molecules were generated and analyzed.

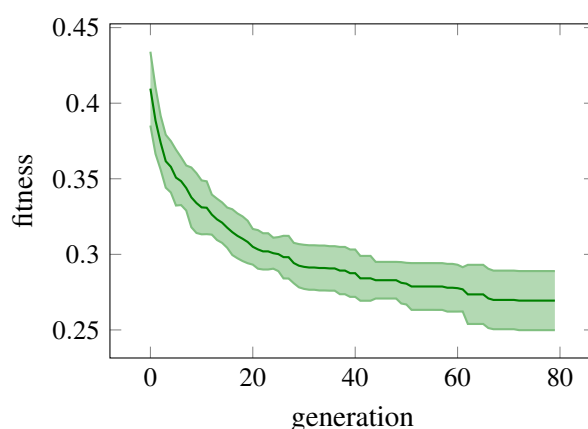


Figure 6.5: Mean values and standard deviations of the fitness scores for the best performing individual in each generation were calculated over 15 runs of **EMGA**. Note that lower fitness scores are indicative of more suitable inhibitors of M^{pro} .

To increase the number of available molecules for subsequent **MD** simulations, a final run of **EMGA** was executed with an increase in both μ and λ values to 50 and 300, respectively. For a comprehensive record of all 144350 generated molecules and their corresponding metrics, please refer to the supporting information of [Ele+22].

Among the 144350 generated molecules, a subset of the top 200 molecules was selected based on their fitness scores. From this subset, 21 molecules were hand-picked based on the validity of their molecular structures. Figure 6.6 showcases these molecules selection along with their respective spider charts, which visualize the five metrics. Additionally, Table C.1 in the appendix provides the associated **SMILES** strings and metric values of these molecules.

The best performing molecules generated by **EMGA** exhibit notable structural patterns. These patterns include ring-based structures, particularly nitrogen-based heterocycles like the six-membered pyridine and pyridazine, as well as the seven-membered azepine and diazepine rings. These ring structures seem to contribute to ligand stabilization and are advantageous for protease inhibition. Furthermore, the generated ligands feature functional groups such as fluoride and cyanide, as well as oxygen-based groups including carbonyl, carbonamide, and hydroxyl groups. However, carboxylate ester groups were found infrequently. These groups are known to act as electron donors, facilitating the

formation of hydrogen bonds that enhance the BA between the ligand and the M^{pro} . Similar structural patterns have been observed and discussed in previous studies [MDJ21; Sha+20].

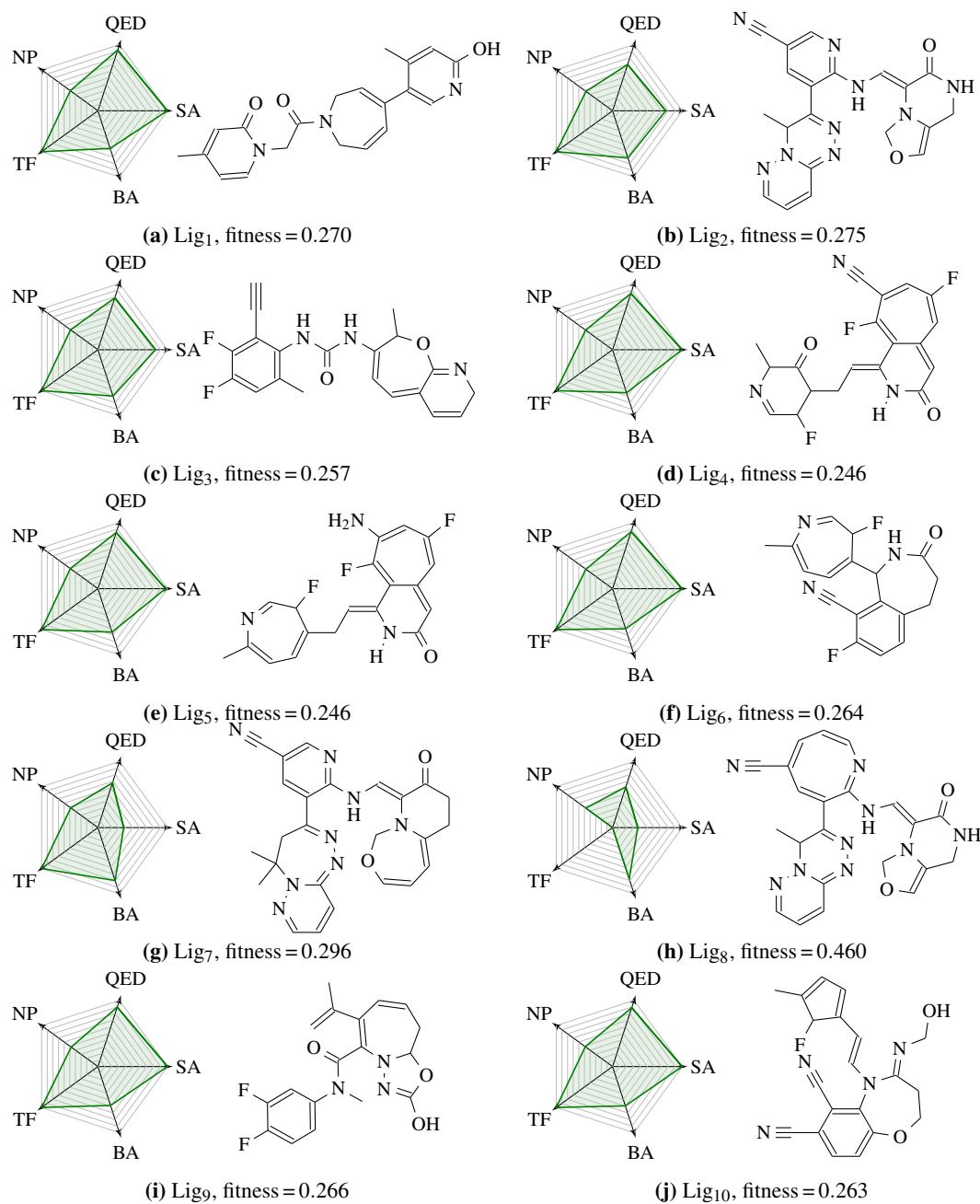


Figure 6.6: Selection of ligands created by EMGA. The spider charts show how well the molecules perform with respect to the five metrics. The best values are on the edge of the spider chart and the worst values are in the center. (to be continued)

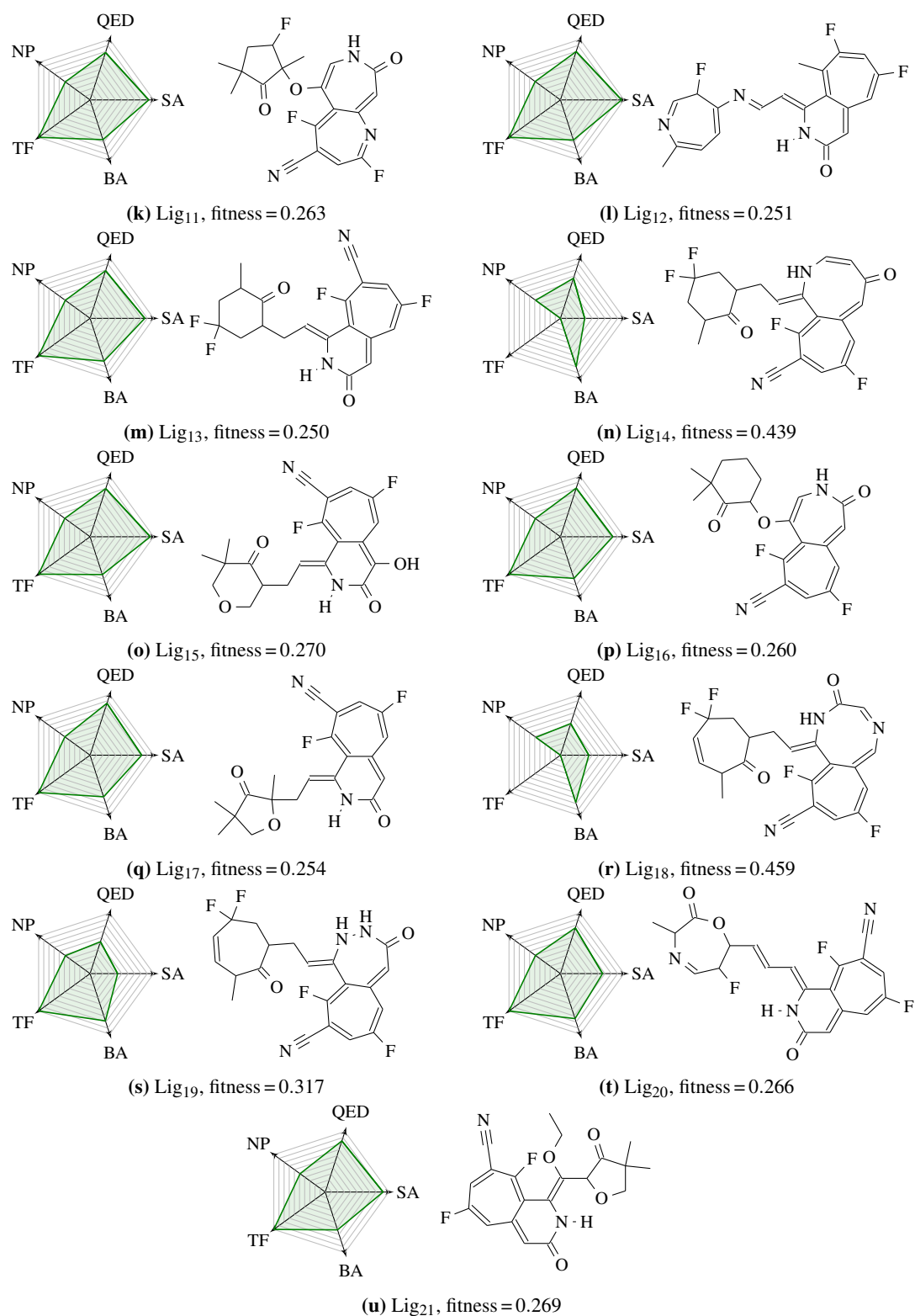


Figure 6.6: Selection of ligands created by EMGA. The spider charts show how well the molecules perform with respect to the five metrics. The best values are on the edge of the spider chart and the worst values are in the center. (cont.)

6.3.2 Molecular Dynamics

The 21 ligands previously selected (see Table C.1) on the basis of EMGA are now evaluated in more detail with respect to their binding to the protease enzyme. To do this, a series of simulations are carried out. Finally, the formulae described in Section 6.2 to calculate the binding free energies. In this way, it can be determined whether the ligand would hold the desired position or move away from it over time.

First, a simulation is performed for the empty receptor without a ligand. In addition, one simulation is run for each ligand and each ligand-receptor complex. This results in 43 simulation runs. Although multiple replicates of the simulation would be useful for more specific biophysical applications based on the proposed methodology, only the methodology will be presented here, and therefore replicates were not performed. In addition, it should be noted that the MD simulation is very computationally time-consuming. It takes 2 to 3 weeks on the high performance cluster (HPC) of the university to run a simulation.

To check if a ligand stays at the desired position – the M^{pro} binding site – the center of mass (COM) distance between the ligand and the binding site is calculated during the complex simulation. The M^{pro} binding site is shown in Fig. 6.7.

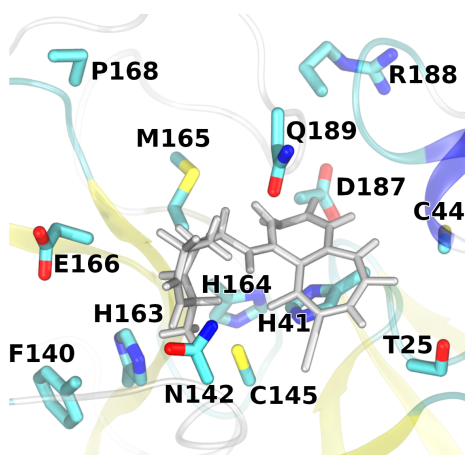


Figure 6.7: Binding site of M^{pro} defined by the labeled residues [Dai+20] with Lig₁₉ in its initial bound pose illustrated in gray. [Ele+22]

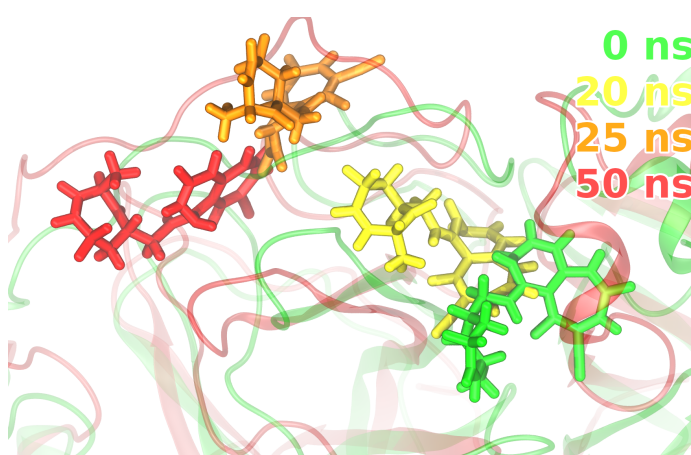


Figure 6.8: Position of Lig₁₉ in M^{pro} after 0 ns (green), 20 ns (yellow), 25 ns (orange), and 50 ns (red) of simulation. [Ele+22]

The average COM distances during the last 10 ns of the 50 ns simulations are listed, along with other values that will be described later, in Table 6.1. The ligands with an average COM distance above 7 Å are discarded, as it indicates that these ligands leave the initial binding site. The COM distances of the affected ligands (Lig₃, Lig₄, Lig₁₆, Lig₁₉, Lig₂₀, and Lig₂₁) that drift away during the simulation are shown in Fig. 6.9 along the time course. To illustrate the analysis of the MD simulations, the position of one of these ligands, Lig₁₉, is shown at different time instances in Fig. 6.8. The remaining ligands have a COM distance value in the range 2.5 Å to 7 Å and are therefore candidates for the time being, but will need to be investigated below using further criteria.

Table 6.1: The averaged values of center of mass (COM), root mean square displacement (RMSD), and root means square fluctuations (RMSF) for the 21 ligands are calculated for different time intervals within the complex simulations. During the last 10 ns of the simulations the COM distances are measured. The average RMSD values of the ligands are calculated based the last 30 ns the simulations with the protein backbone aligned with itself. The avg₅₀ RMSF is averaged for all atoms in the ligands throughout the 50 ns simulations. Ligands that exhibit high COM and RMSD values are identified and marked in red, indicating that these ligands are discarded from further analysis. [Ele+22]

Ligand	avg ₁₀ COM (Å)	avg ₃₀ RMSD (Å)	avg ₅₀ RMSF (Å)
Lig ₁	4.82	7.66	1.37
Lig ₂	3.83	7.80	1.05
Lig ₃	12.43	14.05	0.99
Lig ₄	9.08	6.40	1.03
Lig ₅	6.15	6.90	1.15
Lig ₆	4.70	6.46	0.64
Lig ₇	4.75	4.85	0.51
Lig ₈	5.00	9.13	1.07
Lig ₉	5.87	7.17	0.39
Lig ₁₀	6.20	8.59	0.63
Lig ₁₁	4.23	5.76	1.32
Lig ₁₂	6.82	10.44	1.34
Lig ₁₃	5.04	3.68	0.85
Lig ₁₄	2.62	7.67	1.18
Lig ₁₅	4.85	6.57	1.10
Lig ₁₆	7.33	9.95	0.83
Lig ₁₇	5.24	7.56	0.45
Lig ₁₈	5.55	7.19	1.11
Lig ₁₉	13.28	13.42	1.15
Lig ₂₀	7.27	9.50	1.54
Lig ₂₁	7.93	7.15	0.71

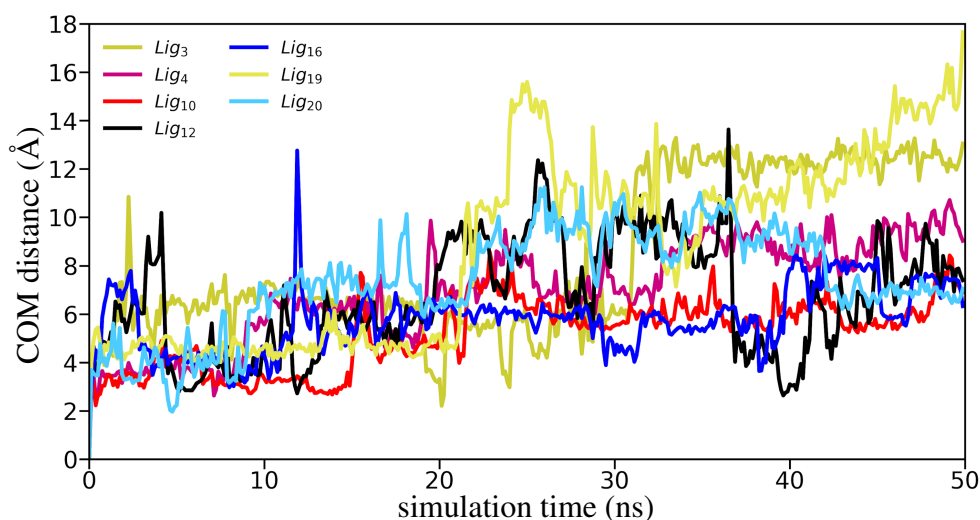


Figure 6.9: Time evolution of the center of mass (COM) distance between the M^{Pro} binding site and the ligands that drifted away from the binding site during the ligand-receptor complex simulations. Each data point was averaged over a time window of 2.5 ps. [Ele+22]

So far, only the distance between the center of mass of the ligand and the binding site was taken into account. However, for a stable bond it is also important to consider whether the ligand itself is moving, e.g., by rotation. To measure the stability of the molecule within the binding pocket, the **RMSD** is calculated. This considers the positional changes of all atoms of the molecule and is defined as:

$$\text{RMSD}(t) = \sqrt{\frac{1}{N_a} \sum_{i=1}^{N_a} \|\mathbf{a}_{i,0} - \mathbf{a}_{i,t}\|^2}, \quad (6.9)$$

where N_a is the number of atoms in a ligand and $\mathbf{a}_{i,t}$ is the position of the i th atom at time t . For this purpose, the positions of the backbone atoms in the course of time are each compared with their initial position at time $t = 0$ on the basis of their Euclidean distance. The **RMSD** aggregates this for a given time point t to a single value. Figure 6.10 shows the **RMSD** of the 21 analyzed ligands over the simulation time.

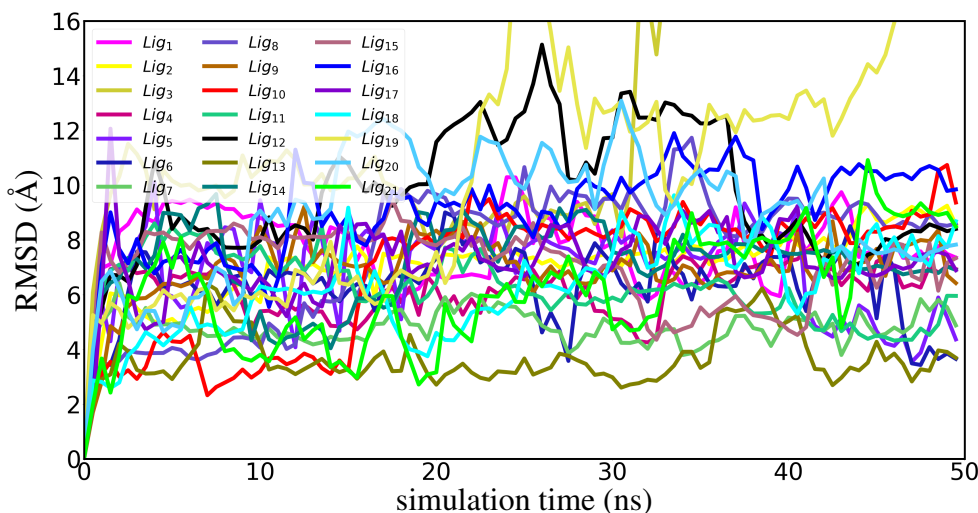


Figure 6.10: Root mean square displacement (RMSD) of the simulated ligands. The RMSD is calculated based on a trajectory in which the protein backbone was aligned with itself. [Ele+22]

Table 6.1 shows the averaged **RMSD** over the last 30 ns of simulation Y. Ligands with a value greater than 7 \AA have not bound to a specific site in the binding pocket and are therefore marked in red in the table (Lig₁, Lig₂, Lig₈, Lig₉, Lig₁₀, Lig₁₂, Lig₁₄, Lig₁₇, and Lig₁₈). Exemplary of these ligands with a high RMSD value, the motion of Lig₈ is shown in Fig. 6.11B. Here it can be seen that although Lig₈ moves only slightly away from the pocket overall, resulting in acceptable COM value, it makes a rotational movement, resulting in the increased RMSD value. Due to the stronger motion, it can be assumed that this ligand will not bind well to the pocket and thus is not a good drug candidate. Therefore, for such ligands, the Eqs. (6.2) and (6.3) cannot be used as an estimate of the binding free energy. Therefore, the ligands with high RMSD value that have already been listed will not be further considered in the following analysis. Lig₁₃ with a **RMSD** value below 4 \AA serves as an example of a good ligand that is likely to bind stable to the pocket (see Fig. 6.11A).

While the **RMSD** calculates the relative position change to the reference point at $t = 0$, it is also of interest to see how much the individual atoms of the ligand fluctuate over the course of the simulation.

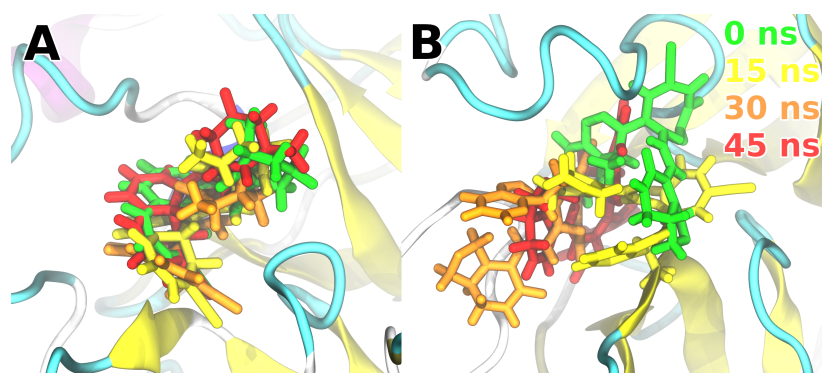


Figure 6.11: Position and orientation of Lig₁₃ (A) and Lig₈ (B) in M^{pro} after 0 ns (green), 15 ns (yellow), 30 ns (orange), and 45 ns (red) of simulation. Based on the stronger motion of Lig₈, which is also reflected in its large averaged RMSD value of 9.13, it can be assumed that this ligand does not reach a proper binding state. [Ele+22]

For this, the position of an atom averaged over time $\langle \mathbf{a}_i \rangle$ is used as the reference point. Then the squared distance of the atom to its reference position, also averaged over time, is determined. The root of this then gives the *root means square fluctuations* (RMSF) for atom i :

$$\text{RMSF}(i) = \sqrt{\langle \|\mathbf{a}_i - \langle \mathbf{a}_i \rangle\|^2 \rangle}. \quad (6.10)$$

In Table 6.1, RMSF averaged over the atoms of each ligand is given. To average over time, the total simulation time of 50 ns is used. Of the ligands not already excluded, Lig₅, Lig₁₁, and Lig₁₅ have in the overall comparison relatively high values ranging from 1.10 Å to 1.32 Å, while the remaining Lig₆, Lig₇, and Lig₁₃ have a low mean RMSD in the range 0.51 Å to 0.85 Å.

Binding free energy estimates were calculated for the remaining ligands Lig₅, Lig₆, Lig₇, Lig₁₁, Lig₁₃, and Lig₁₅ that have COM distances and RMSD values below 7 Å. The calculation uses Eqs. (6.2) and (6.3) based on the last 30 ns of the simulation. From this period 800 frames were extracted and used for the entropy calculation with Eq. (6.7). Pre-experiments with different numbers of frames have shown that 800 frames are sufficient to obtain a converged entropy contribution. The resulting binding free energy estimates are listed in Table 6.2.

Ligands Lig₁₅ and Lig₅ exhibit notably higher binding free energy estimates of -23.0 kcal/mol and -20.8 kcal/mol, respectively, which surpass the third-best ligand, Lig₆, by more than twice the energy. The superior binding free energy values of Lig₁₅ and Lig₅ primarily stem from a significant disparity in the van der Waals (vdW) interactions (a component of E_{MM} in Eq. (6.3)) between the systems with bound and unbound ligands, amounting to approximately -45 kcal/mol. Since almost no hydrogen bonds were observed between ligands and receptor, it is clear that the ligand-receptor interactions are primarily caused by vdW interactions. On the contrary, ligands Lig₇ and Lig₁₁ have positive binding free energy values, indicating that these ligands are unlikely to spontaneously bind to M^{pro} and would likely move away from the binding site should the simulations be extended.

The MD simulations have enabled a twofold refinement of the ligand list generated by EMGA, achieved through dynamic and energetic assessments. As a result of this refined selection process, two highly promising drug candidates, Lig₁₅ and Lig₅, emerged. The logical progression would

Table 6.2: Binding free energy estimates, ΔG^0 , calculated using Eqs. (6.2) and (6.3) and based on the last 30 ns of the simulations. [Ele+22]

Ligand	ΔG^0 (kcal/mol)
Lig ₁₅	-23.0
Lig ₅	-20.8
Lig ₆	-9.5
Lig ₁₃	-4.0
Lig ₇	5.1
Lig ₁₁	11.4

involve validating the potential of the identified drugs through wet lab experiments. Nevertheless, it is important to point out that conducting such experiments is not within the scope of this work.

6.4 Conclusion

In this chapter, a novel drug design workflow was presented. The workflow consists of **EMGA** – a combination of **EA** and **LM** – as well as **MD** simulation and analysis of the resulting molecular candidates. **EMGA** generates drug candidates similar to the **ZINC** database and optimizes them using the five relatively fast to compute metrics **BA**, **SA**, **QED**, **NP**, and **TF**. Thus, the much more time-consuming **MD** analysis only needs to be performed on a few preselected candidates to evaluate them with higher accuracy. After **EMGA** considered several hundred thousand molecules during the course of the calculations, 21 chemically promising molecules were finally hand-selected from the top 200 and used for **MD** analysis. These were further investigated using **COM**, **RMSD**, and binding free energies between the ligands and the binding site, and narrowed down to a few good candidates. Lig₅ and Lig₁₅ are the most promising drug candidates. With the help of the presented method, it is possible to consider a very large search space and to gradually reduce the set of considered molecules in order to perform the time-consuming more exact calculations only for a small preselection. The method has been used here to find a suitable ligand for the **M^{pro}** of **SARS-CoV-2**. However, it could also be applied to other proteases and viruses. As in the previous chapter, the next logical step would be an *in vitro* analysis of the best candidates found.

Although the workflow was demonstrated to generate inhibitors of **M^{pro}**, it can be extended to address various drug discovery challenges. On a methodological level, it may be interesting to dynamically adjust the r_{\max} and δ_{\max} parameters during the evolutionary process. Larger values could provide the evolutionary algorithm with an additional mechanism for exploring the molecular search space, while smaller values could help refine molecules that already exhibit promising properties. While our approach is primarily focused on the early stages of drug discovery, in the future, the promising candidates discovered could be subjected to *in vitro* analysis.

Part IV

Visualization

7 Convolutional Self-Organizing Map

Data Science often comprises work with very large data sets, which in turn often consist of high-dimensional **samples**. Looking back to Chapter 3, for example, a sample of 19 parameters was taken for the quarterly company data. A sample that considers the last 10 quarters thus consists of $19 \cdot 10 = 190$ **features**. If someone would like to view and compare a multitude of these 190-dimensional samples, this should be in a human-interpretable 2- or 3-dimensional representation. With a good dimensionality reduction, the similarities of individual samples to each other should be preserved, i.e., samples that are similar in high dimensional space should also be similar in low dimensional space and vice versa. Another simpler example of high-dimensional data, is image data. An RGB color image has three color values for each pixel. Even at low resolutions, a single image already consists of thousands of features.

The **SOMs** introduced by Teuvo Kohonen [Koh82] offer an option for visualization. They represent high-dimensional data as a 2- or 3-dimensional map. However, the results depend strongly on the features of the input data. High-dimensional data from raw images given into a **SOM** does not necessarily lead to good results for certain use cases. Two images of different objects that happen to both have the same color would have high similarity when looking at the raw data, while same objects of different color would be dissimilar. Therefore, in this chapter a method will be developed that takes into account semantic information in addition to raw color information.

In the field of image recognition, **CNNs** have been successfully used for quite some time [LeC+89; KSH17]. The convolutional layer, which is the main component, is able to recognize patterns in the data and output higher-order features. This fact is exploited by the **ConvSOM** introduced in this chapter by using the output of convolutional layers of pre-trained **CNNs** to serve as higher-order features for the SOM training process. This is expected to yield semantically better SOM results.

This chapter is organized as follows. Section 7.1 provides an overview of related work. Afterwards, the basic **SOM** (Section 7.2) and the **CNN** (Section 7.3) are presented. Section 7.4 introduces the **ConvSOM** while Section 7.5 presents relevant quality metrics for dimensionality reduction. The experimental analysis of the **ConvSOM** is conducted in Section 7.7. Finally, the outcomes are discussed in Section 7.9.

Parts of this chapter are based on the following published paper:

Lars Elend and Oliver Kramer. “Self-Organizing Maps with Convolutional Layers”. In: *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization - Proceedings of the 13th International Workshop, WSOM+ 2019, Barcelona, Spain, June 26-28, 2019*. Ed. by Alfredo Vellido, Karina Gibert, Cecilio Angulo, and José David Martín-Guerrero. Vol. 976. *Advances in Intelligent Systems and Computing*. Springer, 2019, pp. 23–32. DOI: [10.1007/978-3-030-19642-4_3](https://doi.org/10.1007/978-3-030-19642-4_3)

7.1 Related Work

The concept of **SOMs** was developed by Kohonen [Koh82] in the 1980s and is presented in detail in the following Section 7.2. Normally, the input samples of the **SOM** are normalized. Feature transformations can be utilized, such as the kernel **SOM**'s feature transformation, which employs kernel functions to map patterns into a feature space [MF00; And02]. A demonstration of the effectiveness of kernel **SOMs** is in its application in the detection of interturn short-circuit faults in a three-phase converter-fed induction motor [CBM17].

Kutics, O'Connell, and Nakagawa [KON13] present the layered **SOM** for segment-based image classification. They choose a pertinent segment, feed it through diverse image descriptors, and integrate outcomes within interconnected layers. Schleif [Sch17] blends transformed Fourier features with kernelized matrix learning vector quantization. Miikkulainen [Mii90] employs hierarchical feature maps for script recognition.

Villmann et al. [Vil+17] explore the combination of multilayer feedforward networks and learning vector quantizers, utilizing the networks as adaptive filters. This approach does not use convolutional layers and **SOMs**. Dozono, Niina, and Araki [DNA16] examine the mix of convolutional layers and **SOMs**, though with vague description and limited experimental backing. In contrast to our approach, they route the **SOM**'s output to a convolutional layer. A related method by Platon, Zehraoui, and Tahy uses a **SOM** as preprocessing for supervised learning, facilitating the discovery of new classes [PZT17]. Wang et al. [Wan+17] use a combination of **CNN** and **SOM** for monitored quantization to solve the approximate nearest neighbor search issue. Incorporation with a denoising autoencoder is proposed by Ferles, Papanikolaou, and Naidoo [FPN18].

Recent work also presents architectural combinations of **CNNs** and **SOMs**, although these also differ from the one presented here. Aly and Almotairi [AA20] describe deep convolutional self-organizing maps (DCSOM). Here, 3-dimensional **SOMs** are integrated as layers into a larger architecture. Sakkari and Zaied [SZ20] presents the unsupervised deep self-organizing map (UDSOM) algorithm for feature extraction.

7.2 Self-Organizing Map

The *self-organizing map (SOM)* is a biologically inspired neural model using unsupervised learning. It is suitable for visualization of high-dimensional data sets on a 2- or 3-dimensional map. This map is formed by neurons n_i , each occupying a fixed position $\mathbf{p}_i \in \mathbb{R}^k$ on it. Usually a 2-dimensional lattice arrangement is chosen, i.e., $k = 2$. As input, the **SOM** receives data from a training set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, consisting of N samples $\mathbf{x}_i \in \mathbb{R}^m$. Each neuron has an associated weight vector $\mathbf{w}_i \in \mathbb{R}^m$ with the same dimensionality m as the input data. Initially, all these weights \mathbf{w}_i are set to random values of a uniform distribution.

During the learning process, all samples are presented to the **SOM** several times in succession. The **SOM** is then supposed to recognize **patterns** that the **samples** have in common and use this for their subsequent arrangement. To accomplish this, for each given \mathbf{x} , initially the *best matching unit (BMU)* or *winner neuron* must be found. This is the neuron whose weight is most similar to \mathbf{x} in terms of

Euclidean distance. The weight and position of **BMU** for a given sample \mathbf{x} are each marked with an asterisk and are defined as follows:

$$\mathbf{w}^*(\mathbf{x}) := \mathbf{w}_\eta, \quad (7.1)$$

$$\mathbf{p}^*(\mathbf{x}) := \mathbf{p}_\eta, \quad (7.2)$$

where

$$\eta = \arg \min_j \|\mathbf{x} - \mathbf{w}_j\|_2. \quad (7.3)$$

In the update step, the weight of the winning neuron and, in a weakened form, the weights of the neighboring neurons are pulled in the direction of the sample. The strength of the adjustment is determined by the neighborhood function $h(\mathbf{p}_i, \mathbf{p}_j, \sigma) \in [0, 1]$ which measures the connection between two neurons n_i and n_j . For low distances it should give high values and values close to zero for distances outside the radius σ , i.e., $\|\mathbf{p}_i - \mathbf{p}_j\|_2 > \sigma$. Therefore, define the neighborhood function as:

$$h(\mathbf{p}_i, \mathbf{p}_j, \sigma) := e^{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2}{\sigma}}. \quad (7.4)$$

On this basis, each weight \mathbf{w}_j is adjusted to the **SOM**:

$$\mathbf{w}'_j := \mathbf{w}_j + \alpha \cdot h(\mathbf{p}^*(\mathbf{x}), \mathbf{p}_j, \sigma) \cdot (\mathbf{x} - \mathbf{w}_j), \quad (7.5)$$

with learning rate $\alpha \in \mathbb{R}^+$. The **BMU** selection and the update step are iteratively repeated for each sample shown. The two endogenous parameters α and σ , which influence the strength and the adjustments, are usually reduced during the learning process.

In Fig. 7.1 this update step is illustrated by an example. Here, the winning neuron n_8 was determined. The strength of the adaptation depending on the neighborhood is shown in color (from red to blue).

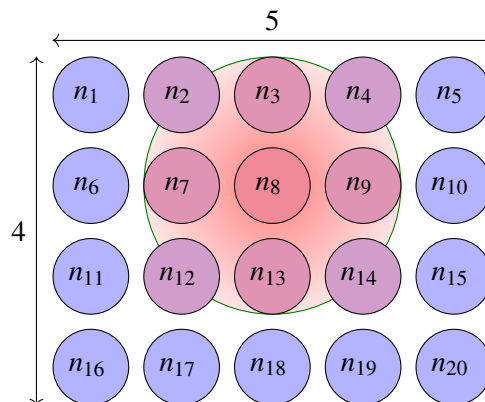


Figure 7.1: Update step of an example **SOM** of size 4×5 . The **BMU**, i.e., the neuron whose weight \mathbf{w}_j is closest to the input vector \mathbf{x} (7.1), is selected – n_8 in this case. The **BMU** and its neighborhood is pulled into the direction of the input, depending on the neighborhood function h (7.4) and the learning rate α (7.5).

Usually the **SOM** receives only single samples for learning one by one. In order to accelerate this process, it is possible to consider batch-wise samples. The runtime is accelerated by the parallel calculations. This is done by adapting the Eq. (7.5) so that a subset of samples $\mathcal{X}' \subseteq \mathcal{X}$ is considered:

$$\mathbf{w}'_j := \mathbf{w}_j + \alpha \cdot \frac{\sum_{\mathbf{x} \in \mathcal{X}'} h(\mathbf{p}^*(\mathbf{x}), \mathbf{p}_j, \sigma) \cdot (\mathbf{x} - \mathbf{w}_j)}{\sum_{\mathbf{x} \in \mathcal{X}'} h(\mathbf{p}^*(\mathbf{x}), \mathbf{p}_j, \sigma)}. \quad (7.6)$$

The sum of the differences weighted by the neighborhood function, is divided by the total influence of the neighborhood function for the respective neuron n_j . This product is then scaled by the learning rate α and added to the previous weight \mathbf{w}_j .

The divisor of Eq. (7.6) must contain the sum of the neighborhood influences and not $|\mathcal{X}'|$, so that the result resembles an iterative execution and so that samples with a distant **BMU** have a correspondingly small influence on the weight adjustment. Note that this variant is slightly different from already existing batch **SOM** variants, because instead of subsets, the whole set of samples \mathcal{X} is considered at the same time [Koh95; Wit+17].

7.3 Convolutional Neural Network

The *convolutional neural network (CNN)* is nature inspired by the receptive fields in the visual cortex and especially used in the field of image recognition. It works with 2-dimensional input data. In contrast to the simple fully connected **ANN**, only partial areas of the data are considered with so-called filters. However, these filters are applied in parallel to the entire data by shifting. Thus here the same weights are used several times. This is particularly advantageous for large images in order to reduce the number of weights required. These filters are part of a convolutional layer, which is the name-giving main component of the **CNN**. Based on the filters being shifted over the data, the Frobenius scalar product is calculated, i.e., the sum of the element-wise products:

$$\tilde{y}_{i,j} = \sum_{k=0}^{m-1} \sum_{\ell=0}^{m-1} \tilde{w}_{k,\ell} \cdot \tilde{x}_{i+k,j+\ell}, \quad (7.7)$$

where m is the size of the filter matrix $\tilde{\mathbf{W}} = (\tilde{w}_{i,j})$, which is usually quadratic, $\tilde{\mathbf{X}} = (\tilde{x}_{i,j})$ is the input, and $\tilde{\mathbf{Y}} = (\tilde{y}_{i,j})$ is the result of the convolution.

Figure 7.2 shows a minimal example of a **CNN** and shows the result of the operation described above. There, e.g., the 1 at the top right of *Res. Filter 1* is obtained by applying the formula to the four elements at the top right of the input and filter 1: $0 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 = 1$.

In this example, the dimensionality of the matrix changes. The resulting matrix size is influenced by filter size, stride and padding. The filter size is usually chosen as $i \times i$ where i is odd. The *stride* specifies the step size with which the filters are moved over the input. In the example, $\text{stride} = 1$ was chosen. Padding allows keeping an unchanged dimensionality by adding zeros around the input (zero padding). The padding size is usually $\frac{\text{filter size} - 1}{2}$. Thus, for a 5×5 filter, zero padding of 2 could be used. This way, the edges and corners of the input are equally considered by the filter as the middle elements.

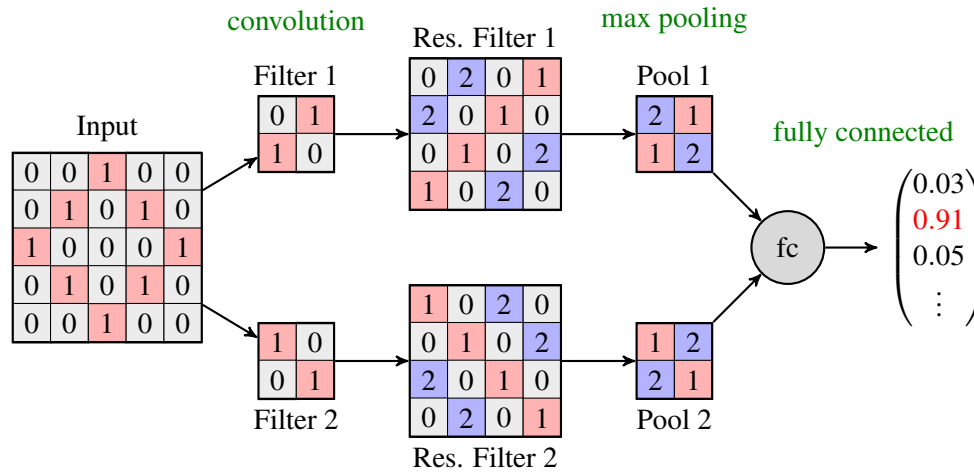


Figure 7.2: CNN example. This minimal example illustrates the main components of a CNN. The input data is first examined in the context of the convolutional layer with different filters by performing an element-wise matrix multiplication. The result can be seen under Res. filter. Afterwards a max pooling is performed. Usually a CNN contains a multiple repetition of these two layers. Finally the values are put into a fully connected layer to get a result vector.

A convolutional layer is usually followed by a pooling layer to reduce the amount of data. In pooling, a certain subrange is considered and then the pooling operation is applied to it, e.g., determine the maximum or the average. In Fig. 7.2, max pooling is applied with size 2×2 , i.e., the maximum of those four values is selected. By connecting convolutional and pooling layers in series multiple times, higher-level features can be detected. Here usually **ReLU** is used as an activation function and dropout to avoid overfitting. Dropout means that random neurons are ignored during each training step. At the end, a fully connected layer is usually used, where its output with a *softmax* function leads to a one-hot coding of the different classes. The softmax function f_s is a normalized exponential function that ensures that the sum of the output values equals 1, and is defined as follows:

$$f_s : \mathbb{R}^C \rightarrow \{\mathbf{z} \in \mathbb{R}^C \mid z_i \geq 0, \sum_{i=1}^C z_i = 1\}, \quad (7.8)$$

$$f_s(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{i=1}^C e^{z_i}}. \quad (7.9)$$

Numerous CNN variants have been proposed in the last decade, e.g., ResNet [He+16] or the Squeeze-and-excitation network [HSS18].

7.4 Convolutional Self-Organizing Map

The *convolutional self-organizing map* (*ConvSOM*) is a combination of the two previously presented methods: CNN and SOM. Since the C in the abbreviation CSOM already has many different meanings: concurrent [NR02; Nea+14], continuous [HD05], complex-valued [OAH14], cognition-based

[Sun+11], contextual [Voe00], and community [CBA15], the approach presented here is more uniquely labeled as **ConvSOM**.

The approach requires a **CNN** pre-trained on the same or similar data with (semi-)supervised learning, whose weights are not further adjusted in the following. Figure 7.3 shows the structure of the **ConvSOM** architecture.

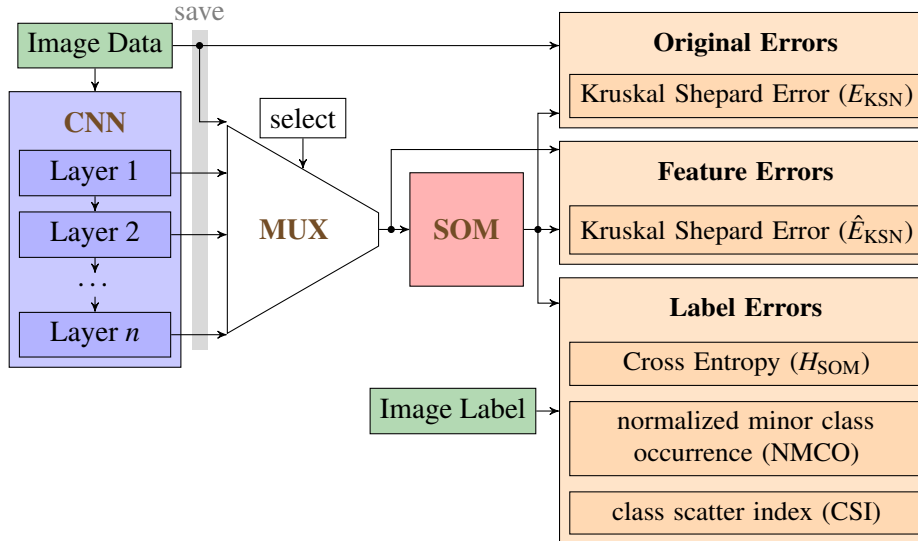


Figure 7.3: Structure of the **ConvSOM** and the associated quality measures. The intermediate output of a particular layer of the **CNN** can serve as input of the **SOM**. For comparison purposes, the original image data can also be used as input. The arrows indicate the information flow and show in particular which data are considered for the different error metrics (original, feature and label). Note that the image labels are only used for the label errors, but the **ConvSOM** procedure itself works unsupervised.

The **ConvSOM** is only trained with an independent data set that was not used for training the **CNN**. After training the **CNN** and freezing its weights, the raw input samples $\mathbf{x} \in \mathbb{R}^{\bar{h} \times \bar{w} \times \bar{c}}$, images of size $\bar{h} \times \bar{w}$ and \bar{c} color channels, are given into the **CNN**. For the **SOM**, however, the input data is flattened to $\mathbf{x} \in \mathbb{R}^m$. Patterns $\hat{\mathbf{x}}_i \in \mathbb{R}^{\hat{m}}$ resulting from the intermediate layers of the **CNN** can serve as input to the **SOM**. The *multiplexer* (**MUX**) allows the selection of any input for the **SOM** and thus different training modes for the **ConvSOM**. The **MUX** and quality measures are not used for the training process itself. The error metrics seen on the right side of Fig. 7.3, are used to evaluate the results and are presented in more detail in the following section.

In order to perform the calculations quickly and efficiently, the **ConvSOM** including metrics was fully developed with **TensorFlow** for the **GPU**. The metrics presented here were also later implemented in a survey paper by Forest et al. on **SOM** metrics [For+20].

7.5 Quality Metrics

To measure the quality of the **SOM** output, we use the Kruskal Shepard error in two functions on the one hand and three newly developed quality metrics on the other hand: cross entropy, minor class occurrence, and class scatter index. Traditional **SOM** error metrics such as topographic and

quantization errors are not suitable for assessing feature-transformation-based SOMs because they rely on comparing neighborhoods using standard metrics like Euclidean distances. However, the assertion that preserving distances and neighborhoods from the original data space is universally desirable is not valid. In contrast, meaningful maps should position samples with similar labels close to each other and keep samples with distinct semantic content at a distance from each other on the map.

In previous Fig. 7.3, it can be seen that these quality metrics are divided into three categories and which information is considered for them. The *original errors* compare the output to the original input data, which is the raw image data. The *feature errors* evaluate the SOM on its own, using the SOM input as a reference. The *label errors* use the label information of the samples to evaluate the result. Although this label information is not available for the unsupervised training process of the SOM, it can be used in the context of labeled benchmark problems to evaluate the outcome of the ConvSOM. Thus, the ConvSOM would only train with the samples $\mathbf{x}_1, \dots, \mathbf{x}_N$, but for the evaluation, the associated labels $\mathbf{y}_1, \dots, \mathbf{y}_N$ are also used at the end.

7.5.1 Kruskal Shepard Error

The Kruskal Shepard error E_{KS} is used in multidimensional scaling as an error measure [Kru64]. It measures how well the distance information of the data is preserved in the low-dimensional space and is defined as:

$$E_{KS} := \|\mathbf{D}_{\text{data}} - \mathbf{D}_{\text{SOM}}\|_F^2, \quad (7.10)$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm, \mathbf{D}_{data} is the normalized distance matrix in data space, and \mathbf{D}_{SOM} is the normalized distance matrix on the map. The distance matrices contain the pairwise distances between the data points or the positions on the SOM. The normalized matrices are scaled to the range $[0, 1]$ by dividing by the largest value [MK17].

The E_{KS} does not consider semantic information, but it allows evaluating how well the SOM is able to generate the low dimensional representation based on the convolutional layer features $\hat{\mathbf{x}}$. Since the size of the distance matrices depends on the number of samples $\mathbf{D} \in \mathbb{R}^{N \times N}$ and thus the E_{KS} would tend to become larger as the number of samples increases, it makes sense to normalize this to the range $[0, 1]$ as well. To do this, we determine the theoretical maximum of E_{KS} , which occurs when all pairwise distance values are maximum (i.e., 1). The result is $\max(E_{KS}) = N^2 - N$, since there must be zeros on the diagonal. Therefore, we define the normalized version E_{KSN} as a function of samples as:

$$E_{KSN} := \frac{E_{KS}}{N^2 - N}. \quad (7.11)$$

Now $E_{KSN} = 0$ would mean that all distances were perfectly transferred into the low-dimensional space. As mentioned before, the Kruskal Shepard error is used twice for different inputs (see Fig. 7.3). Besides the output of the SOM, the original samples \mathbf{x} are considered for \hat{E}_{KSN} and the transformed features $\hat{\mathbf{x}}$ of the convolutional layer are considered for \hat{E}_{KSN} .

7.5.2 Cross Entropy

Cross entropy, which is otherwise also used to evaluate a classification, is utilized as the first label-based metric for evaluating the **ConvSOM** results. However, the cross entropy calculation needs to be somewhat adapted for **ConvSOM**, since there are no fixed target values for the individual neurons. The reason for this is the random starting distribution and the random order of the images considered. Therefore the maps are very different, but may nevertheless be equivalent. Thus, no fixed expected distribution can be given. Instead, we use the most frequent assigned class of a neuron as the target class. For this purpose, we count for each label the samples for which the considered neuron n_i is determined as **BMU**: To do so, we count the samples of each label for which the neuron n_i is determined to be **BMU**:

$$\mathbf{c}_i = (c_{i,1}, \dots, c_{i,C}), \quad (7.12)$$

$$c_{i,j} = |\{\mathbf{x}_k \mid \forall k \in \mathbb{N}, 1 \leq k \leq N : \mathbf{y}_k = j \wedge \eta = i\}| \quad \forall i, j \in \mathbb{N}, 1 \leq j \leq C, \quad (7.13)$$

where C is the number of classes and η as defined in Eq. (7.3).

The cross entropy of the SOM is calculated by summing the relative frequencies the expected most common classes calculated as follows:

$$H_{\text{SOM}} := \sum_i -\log \frac{\max_{j \in \mathcal{C}}(c_{i,j})}{\sum_{j \in \mathcal{C}}(c_{i,j})}, \quad (7.14)$$

where $\mathcal{C} = \{1, \dots, C\}$ is the set of classes. If there are two classes, both of which occur most often, any one of them can be chosen without affecting the result. Ideally, if only samples of the same class are assigned for a neuron at a time, $H_{\text{SOM}} = 0$. As for all quality metrics presented here, low is better than high. This metric would tend to benefit when there are many neurons and few samples.

7.5.3 Minor Class Occurrence

We assume that ideally there should be as little overlap as possible of different classes in individual neurons. We therefore introduce *minor class occurrence* (**MCO**) as another new metric to evaluate this. For each neuron, we count the number of patterns whose labels do not belong to the main class:

$$\text{MCO} := \sum_i \left(\sum_{j \in \mathcal{C}} (c_{i,j}) - \max_{k \in \mathcal{C}} (c_{i,k}) \right). \quad (7.15)$$

This metric closely resembles cross-entropy but does not differentiate based on error distribution. To facilitate comparisons across varying numbers of samples N , normalization results in the *normalized minor class occurrence* (**NMCO**):

$$\text{NMCO} := \frac{\text{MCO}}{N}. \quad (7.16)$$

7.5.4 Class Scatter Index

The last metric we introduce is *class scatter index* (**CSI**). It is intended to evaluate the goal that labels occur in as few clusters as possible and are not scattered across the map. This is done by determining

the number of clusters for each class. Here, a cluster is defined as follows: If a class $k \in \mathcal{C}$ is considered, two directly adjacent neurons n_i and n_j belong to the same cluster if and only if they have been assigned samples with the same class k , i.e., $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq 1 \wedge c_{i,k} > 0 \wedge c_{j,k} > 0$. For each class k , the number of clusters s_k is now calculated. The **CSI** is then the mean number of clusters on the map for all classes:

$$\text{CSI} := \frac{1}{C} \sum_{k \in \mathcal{C}} s_k. \quad (7.17)$$

7.6 Visualization

Three different visualization types of **SOM** results were used for the experiments. They provide a more direct and simple view of the map and can also be used to compare the progression of training across epochs to see if positive evolution is occurring.

It is often easiest to directly output the weight vectors of the neurons as images in a *weightage plot*. In Fig. 7.4a this was done for a 20×20 **SOM** which was trained with **MNIST**. The digits visible there are usually not to be found in the same way in the data set. Here, for example, the transition from the upright 1 to the oblique 1 and to the 7 is clearly visible from the bottom left to the middle right.

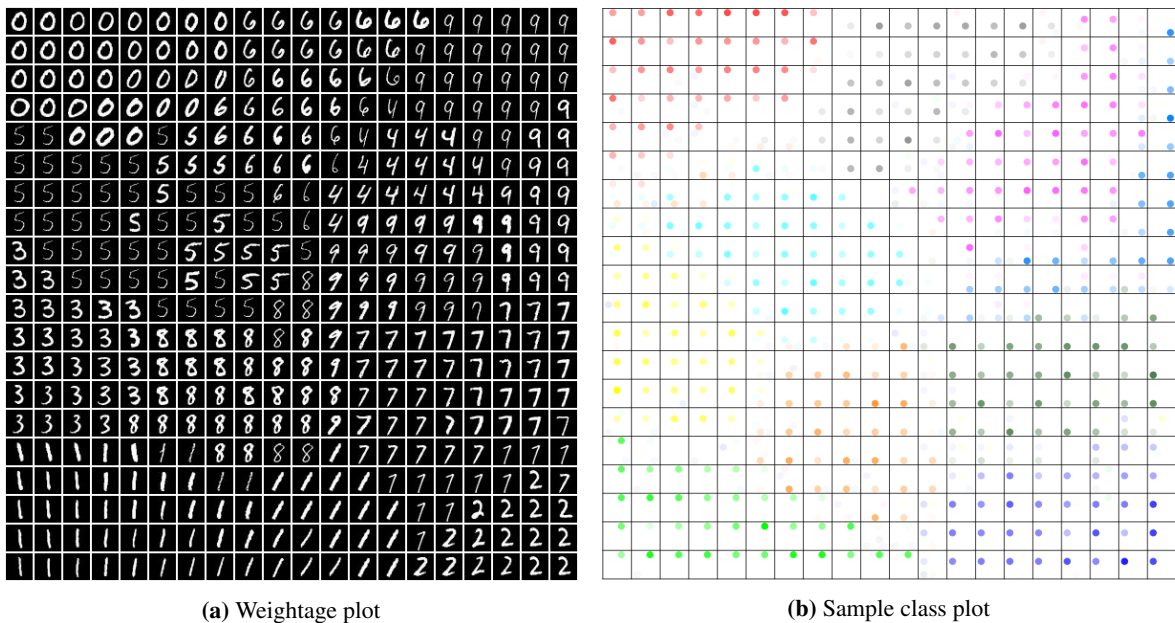


Figure 7.4: Visualization plots for **SOM** results. Both plots show the same 20×20 **SOM** trained on a subset of **MNIST**. A legend for the sample class plot can be found in Fig. 7.5.

However, if the input data and thus the weight vectors are not directly interpretable as a meaningful image, as is the case with **ConvSOM**, the weight plot is not as suitable. Instead, the closest sample from the data set is then shown as the representative for each neuron. This representation method is used in the following section in Fig. 7.7.

To provide an overview of the distribution of all used samples and their classes on the map, we developed the *sample class plot*. Within each neuron, each class is assigned a color and position, as shown in Fig. 7.5. In this way, it is possible to show which classes of samples have been assigned to that neuron. The number of assigned samples is indicated by the color intensity. If the position for

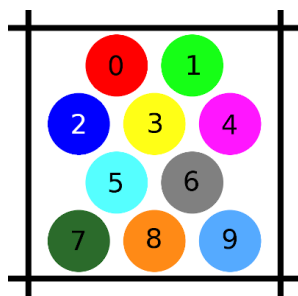


Figure 7.5: Legend of the sample class plot. Each class has its own color and position. The color intensity indicates, how many samples of the class are assigned to a neuron.

class 0 is white, no sample of this class has been assigned to the neuron; for light red, few, and for red, many. The color scaling is adjusted relative to the number of samples. The result can be seen in Fig. 7.4b. Especially at the transition points between the classes, several points per neuron can often be seen. A completely white field here means that the corresponding neuron was not the **BMU** for any sample.

7.7 Experimental Analysis

In this section, **ConvSOM** is studied experimentally using the two well-known image data sets **MNIST** and **CIFAR-10**. In the following, we first describe the experimental settings (Section 7.7.1). Then, the results of the quality measures (Section 7.7.2) and visualization (Section 7.7.3) are presented.

7.7.1 Experimental Settings

The **MNIST** experiments use a **CNN** with two layers of 16 and 32 kernels, each with a size of 3×3 , a stride of 1, **ReLU** activation, and non-overlapping 2×2 max pooling. The second layer incorporates dropout with a rate of 0.25. Subsequently, a dense layer with 128 neurons and **ReLU** activation is followed by an output layer utilizing SoftMax with 10 neurons. This network is trained for 100 epochs using AdaDelta as the optimizer, specifically on the first 10000 samples, and achieves an accuracy rate of 98.5 % on an independent test set.

In the case of **CIFAR-10**, a **CNN** with four convolutional layers is employed, with the first two layers having 32 filters and the subsequent two layers having 64 filters, all with a 3×3 shape, **ReLU** activation, non-overlapping 2×2 max pooling, and dropout rates of 0.25 for the second and third layers and 0.5 for the fourth layer. Following the convolutional layers, a dense layer with 512 neurons and **ReLU** activation is used, and the output layer employs SoftMax with 10 neurons. This network achieves an accuracy of 80.5 % when trained for 1000 epochs using RMSProp on the complete data set.

The 10×10 -**SOM** is trained in batch mode with 100 epochs, a batch size of 100, a learning rate of $\alpha = 0.5$ linearly reduced to 0.05, and a neighborhood radius σ gradually reduced from 5.0 to 0.5 during the training process. The **ConvSOM** is implemented in **TensorFlow**, while the **CNNs** are based on **Keras**.

7.7.2 Quality Measure Results

We will now compare the **ConvSOM** with the normal **SOM** based on the quality metrics introduced in Section 7.5. Table 7.1 shows the experimental results for the **ConvSOM** variants (layer 1 to 3) and the original **SOM** (orig) for the **MNIST** data set. Here, different sizes N for the training set were examined, and 20 replicates were performed for each. It can be observed that the values of \hat{E}_{KSN} and E_{KSN} are especially for $N \in \{1000, 2000\}$ very similar to each other and indicate that the **ConvSOM** achieves reasonable results in terms of distance preservation. In the three label metrics: H_{SOM} , **NMCO**, and **CSI**, the **ConvSOM** achieves much better results. For comparability, note that H_{SOM} increases with N because, unlike **NMCO**, this metric is not related to N . N is normalized. The best – i.e., the smallest – values are bold in the table for each different N .

Table 7.1: **ConvSOM** on **MNIST** with 20 repetitions and different training set sizes N . The mean value is given for each metric. If the standard deviation (SD) is not smaller than 10^{-2} , it is given in the form mean \pm SD.

N	input	\hat{E}_{KSN}	E_{KSN}	H_{SOM}	NMCO	CSI
500	orig	0.094	0.094	26.002 \pm 1.935	0.180	4.435 \pm 0.365
	layer 1	0.090	0.093	22.659 \pm 1.658	0.154	4.175 \pm 0.340
	layer 2	0.085	0.095	12.137 \pm 1.653	0.077	3.110 \pm 0.377
	layer 3	0.031	0.097	4.645\pm1.116	0.023	1.620\pm0.194
1000	orig	0.230	0.230	27.862 \pm 1.650	0.180	4.535 \pm 0.292
	layer 1	0.229	0.229	25.808 \pm 2.023	0.158	4.660 \pm 0.244
	layer 2	0.234	0.234	15.715 \pm 1.383	0.089	3.760 \pm 0.235
	layer 3	0.237	0.238	7.484\pm1.198	0.029	1.705\pm0.190
2000	orig	0.225	0.225	31.106 \pm 2.359	0.204	5.380 \pm 0.424
	layer 1	0.222	0.222	26.240 \pm 1.717	0.166	6.045 \pm 0.606
	layer 2	0.226	0.226	16.534 \pm 1.555	0.089	4.885 \pm 0.318
	layer 3	0.232	0.232	8.735\pm0.996	0.034	2.360\pm0.223

The observations described also apply to the **CIFAR-10** data set, whose associated experimental results are presented in Table 7.2. In comparison to **MNIST**, it can be noted here that all quality measures perform worse, which is likely due to the higher complexity of the data. However, unlike **MNIST**, the values for H_{SOM} , **NMCO**, and **CSI** are similar for layer 2 than for orig, while they are better for layers 1 and 3.

To analyze the **CSI** a bit more closely, we consider the number of individual classes s_k . Figure 7.6 shows box plots for the original **SOM** and for **ConvSOM** layer 3 for the two data sets with 500 samples. The data are also based on the 20 times repetitions of the experiments. The comparison shows that the **CSIs** of the **ConvSOM**, significantly reduces the number of all clusters. This confirms the assumption that the **ConvSOM** can produce better maps concerning high-level information. One can also see that especially in the original **SOM** some classes are more distributed than others, e.g., in **MNIST** the digit 8. This indicates that this class in the original representation has many similarities with other classes.

Table 7.2: ConvSOM on CIFAR-10 with 20 repetitions and different training set sizes N . The mean value is given for each metric. If the SD is not smaller than 10^{-2} , it is given in the form $\text{mean} \pm \text{SD}$.

N	input	\hat{E}_{KSN}	E_{KSN}	H_{SOM}	NMCO	CSI
500	orig	0.025	0.025	88.079 ± 2.922	0.571	11.250 ± 0.477
	layer 1	0.063	0.039	75.229 ± 2.699	0.514	8.530 ± 0.682
	layer 2	0.030	0.050	88.285 ± 2.524	0.578	11.505 ± 0.672
	layer 3	0.035	0.046	52.571 ± 3.040	0.381	4.865 ± 0.545
1000	orig	0.216	0.216	102.758 ± 2.059	0.625	6.450 ± 0.556
	layer 1	0.214	0.213	84.712 ± 2.798	0.549	7.085 ± 0.488
	layer 2	0.213	0.213	106.634 ± 2.280	0.645	6.735 ± 0.621
	layer 3	0.201	0.201	56.452 ± 1.578	0.401	3.655 ± 0.488
2000	orig	0.206	0.206	111.509 ± 2.503	0.658	3.540 ± 0.393
	layer 1	0.207	0.207	88.246 ± 1.705	0.563	5.500 ± 0.521
	layer 2	0.209	0.209	118.471 ± 1.960	0.679	3.315 ± 0.575
	layer 3	0.192	0.1925	57.969 ± 1.353	0.413	3.280 ± 0.284

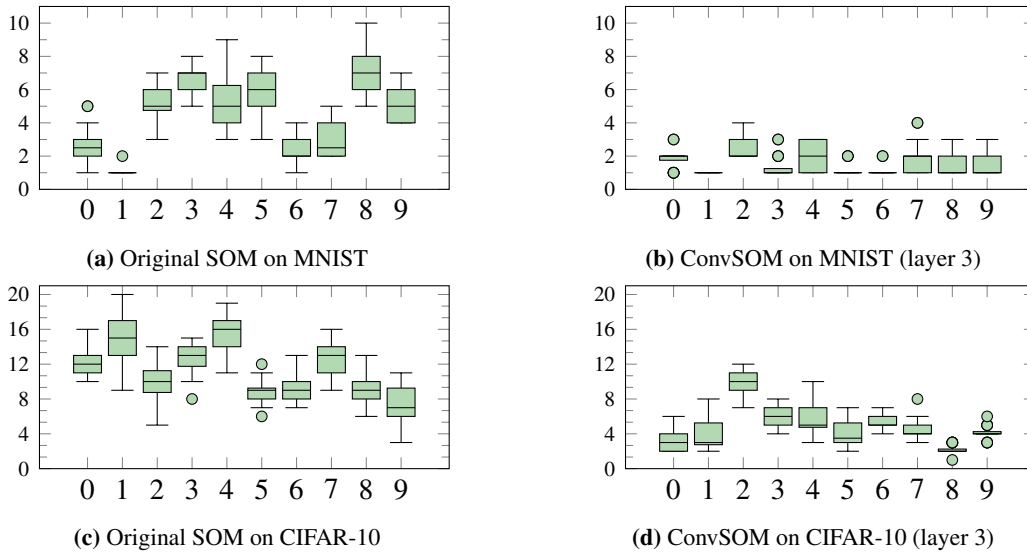


Figure 7.6: Box-plots of class-wise CSI (s_k) of original SOM and ConvSOM on MNIST and CIFAR-10 with $N = 500$. Outliers are displayed as circles.

7.7.3 Visualization Results

A visualization of the results for the original SOM and the ConvSOM is shown in Fig. 7.7. After training was completed, for each neuron n_i , the image from the previously unused test data set closest to the neurons weight was shown: $\arg \min_{\mathbf{x} \in \mathcal{X}_{\text{test}}} \|\mathbf{x} - \mathbf{w}_i\|_2$. The weights of the SOM cannot be visualized directly in the ConvSOM because these data are in a different format, which depends on the layers of the CNN. Figure 7.7 shows that the ConvSOM maps are a broader representation of the different classes. The ConvSOM tends to have large clusters for each class, instead of several smaller clusters. This is also consistent with the smaller CSI values that we observed in the previous section.

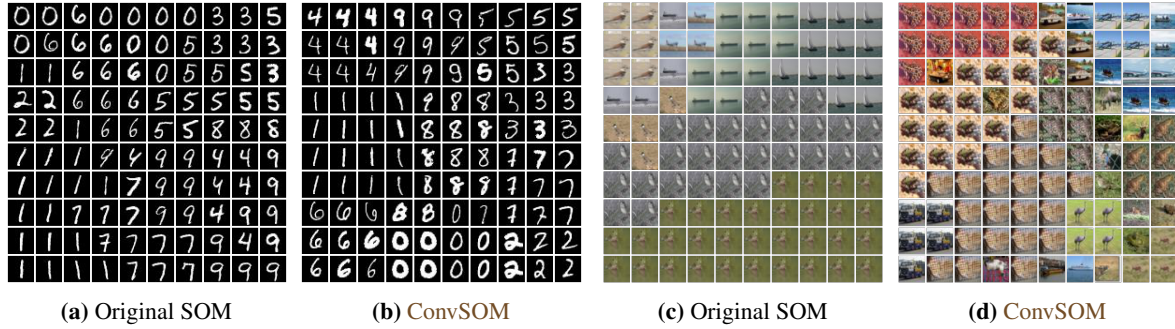


Figure 7.7: Comparison of exemplary original SOM and ConvSOM training results on MNIST (left) and CIFAR-10 (right) with 500 samples. At each neuron location corresponding to a weight \mathbf{w} , the closest sample \mathbf{x} (image) from the test set is shown.

7.8 DeViSE as Semantic Metric

In some of the metrics presented so far, the image label was used to evaluate the results. Since the ConvSOM is otherwise an unsupervised method, the question arose whether a suitable metric could be found that evaluates the semantic coherence of the map without requiring the label. Thus, the idea was born to develop a semantic metric based on the *deep visual-semantic embedding model* (DeViSE) [Fro+13]. DeViSE is able to transform images into a semantically meaningful word vector representation. To create the word vectors, a LM was trained on a corpus of 5.7 million Wikipedia articles.

The idea was further explored during the supervision of the master thesis “DeViSE as Semantic Measure for Convolutional Self-Organizing Maps” by Rina Ferdinand. To use the word vectors for a metric, we used a formula similar to E_{KS} (Eq. (7.10)):

$$E_{DeViSE} := \|\mathbf{D}_{\text{word_vector}} - \mathbf{D}_{\text{SOM}}\|_F^2, \quad (7.18)$$

where $\mathbf{D}_{\text{word_vector}}$ is the distance matrix of the word vectors.

Figure 7.8 shows how the ConvSOM architecture was extended with DeViSE and the resulting semantic error. For the experiments, the two additional larger data sets CIFAR-100 and ImageNet were used. In the experiments performed, E_{DeViSE} often returned similar values to \hat{E}_{KSN} and E_{KSN} . It is possible that adjustments to the E_{DeViSE} formula or the other parameters could lead to more meaningful results.

7.9 Conclusion

In this chapter, the combination of convolutional layers with a SOM – ConvSOM – was presented. New specialized metrics were developed to evaluate the quality of the results. Experimental analysis has shown that the high-level features used in ConvSOM lead to better dimension reduction results, especially with respect to the label-oriented metrics. For example, the reduced CSI shows that the training process benefits from high-level features. Thus, it is also visible in the visualization that there

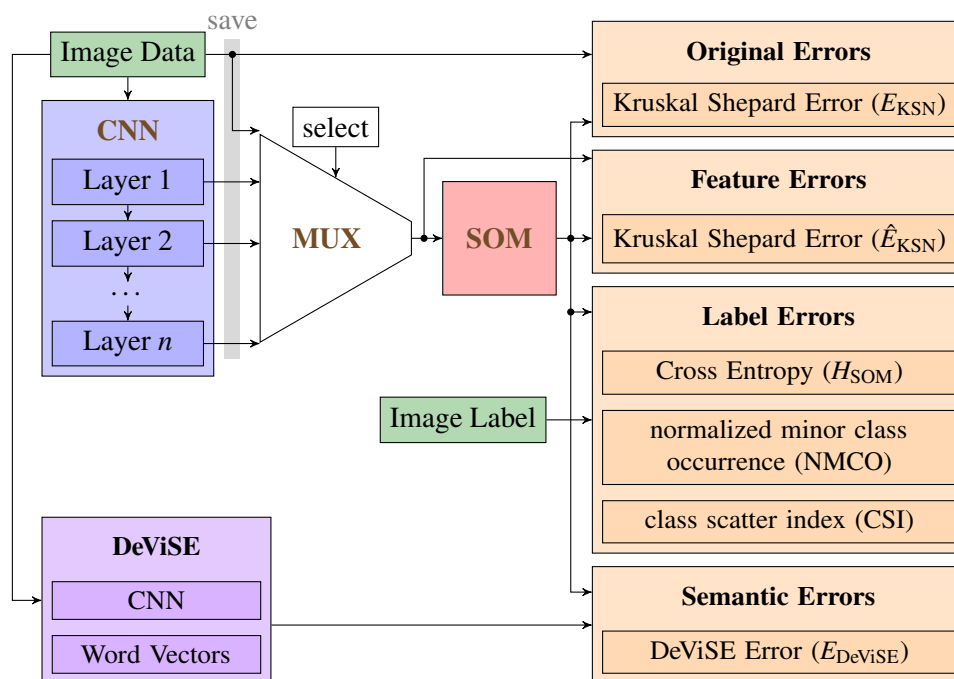


Figure 7.8: Structure of the ConvSOM (see Fig. 7.3) extended by DeVISE (purple) and semantic DeVISE error E_{DeViSE} (lower right).

are fewer distributed clusters of the same class in the ConvSOM. The semantic metric E_{DeViSE} allows to evaluate the results without the label information. Therefore, unsupervised evaluation is also possible.

More data sets could be investigated in the future. In particular, it would be interesting to see how the ConvSOM can handle larger images with multiple objects per image. Instead of image data, it would also be possible to visualize the time series data of the financial forecast (see Chapter 3) with ConvSOM or a slightly adapted variant of it. This could be used, for example, to group similarly behaving companies.

Part V

Closing

8 Conclusion

Computational intelligence (CI) methods are very versatile and can be applied in almost any field. They can be used for automation or as a supporting tool. In this thesis, methods for more application-oriented problems have been developed, as well as new, more fundamental concepts. In the following, we will summarize the individual parts and present their respective results. Furthermore, for each part, we propose on possible extensions and interesting questions for the future.

8.1 Earnings Prediction

In Part II, we looked at predicting the earnings per share (EPS) of U.S. companies. We based the model on detailed information based from quarterly reports, as well as some daily indicators such as stock price and trading volume. In particular, we performed normalization to better compare companies of different sizes, as part of the data preprocessing. We trained and compared different types and settings of deep neural networks (DNNs) for forecasting. Training uses historical data by trying to predict an already known data point using data from a specific period before. This allows the model to learn in a supervised manner. We set this period to the last 20 quarters and the last 20 trading days according to previous experiments. Besides dense layers, we also used long short-term memorys (LSTMs) and temporal convolutional networks (TCNs), which are specialized for time series. We compared our results to the simple persistent model and to the average analyst forecast. As an error metric, we used mean squared error (MSE), which compares the predicted value to the actual value and penalizes larger deviations more severely. Then we used skill score (SS) to compare the different forecasts.

Our results show that the forecast based on the quarterly data can be further improved by including the daily data. Therefore, we have chosen network architectures that first process the quarterly data and the daily data separately for a few layers before combining them with a merge layer and providing the output value after a few dense layers. Regarding the quarterly data, the LSTMs and TCN clearly outperformed the dense layers, while for the daily data the dense layers performed slightly better. This could be due to the fact that the DAILY SHARES consist of fewer features. We used the best LSTM and TCN architectures for the rest of the experiments. We have found that the models work very well, especially for the group of non-financial companies. Such a subdivision is useful because financial companies behave very differently in many aspects (e.g., leverage ratio). We also used the selected architectures for different industries. In the Fama-French industries with 5 groups (FFI5) classification, the high-tech group performed best compared to the analysts. The quality of the solution was also confirmed in the independent test data set.

As a result, we can conclude that CI procedures are quite capable of outperforming analysts in certain fields. Moreover, in other fields they could be used as an additional indicator by analysts.

Future Work

Financial forecasting using deep learning techniques could be explored further in several ways. The method could be applied to other data sets to further investigate the robustness of the approach. It is also conceivable that other relevant data sets could complement the previous input data and thus further improve the result. For example, comparative indices, seasonal influences, or sentiment data could be taken into account. A recent paper by Hajek and Munk [HM23] also uses LSTMs to process quarterly financial data, but extends the inputs to include sentiment and emotion indicators. In particular, they use speech emotion recognition to take into account the emotional state of managers.

Other DNN architectures such as transformers could be investigated. Furthermore, the exact procedure could also be optimized. So far the networks were trained on the whole data set. When the companies were divided into different groups, either according to whether they were financial companies or according to Fama-French industries (FFI) industry groups, differences in the prediction performance became apparent. The next step might be to train industry-specific networks. This would result in a hierarchical structure that first determines to which category or industry the company in question belongs before passing it to the appropriate network. This classification could be done either using the existing FFI classifications or again using CI procedures. In this case, a separate model would in the first step classify the companies based on their data. Another option would be to subdivide by company size, as small and large companies may differ more in their behavior. Therefore, it is foreseeable that CI procedures can be further improved in this area and thus serve as a helpful support or even as a substitute in case of missing analyst forecasts.

8.2 Molecule Design

In Part III, we looked at molecule design in a medical context. The goal was to inhibit viral replication inside a cell. To do this, we needed to find suitable molecules that could act as protease inhibitors for the viral protease. Specifically, we considered main protease (M^{pro}) from severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2). After an introduction to organic chemistry and evolutionary algorithms (EAs), we presented two evolutionary methods to generate molecules. To evaluate the molecules, we used the same five metrics for both methods: binding affinity (BA), synthetic accessibility (SA), quantitative estimate of drug-likeness (QED), natural product-likeness (NP), and toxicity filters (TF). The metrics were calculated using MOSES and QuickVina 2. However, the two methods differ in the exact procedure and the molecule representation used.

The evolutionary molecule search in Chapter 5 uses the self-referencing embedded strings (SELFIES) representation. In the EA, three mutation operations are objected to: deletion, replacement, and insertion. In the weighted sum evolutionary molecule search (WSEMS), the fitness value is the weighted sum of the five metrics. For our second approach (Pareto ranking evolutionary molecule search (PREMS)) we use non-dominated sorting genetic algorithm II (NSGA-II) that tries to optimize each of the five metrics.

In Chapter 6 we introduced evolutionary molecule generation algorithm (EMGA). Here, the simplified molecular-input line-entry system (SMILES) string was used as the molecule representation. The EA was complemented by a language model (LM) which handles the initialization and mutation of

individuals. The **LM** was trained on a subset of the **ZINC** database to generate valid **SMILES** strings and to fill gaps within a **SMILES** string. A **molecular dynamics (MD)** analysis was then performed on the 21 most promising molecules to allow more accurate simulation and evaluation of ligand-protease binding.

Let us first look at the results of the evolutionary molecule search with **SELFIES**. For each of the experiments, we performed 20 runs for 200 generations. As expected, the **PREMS** approach is able to find molecules with better results for individual metrics. This does not mean that it will find molecules that are better in all metrics. For example, a molecule may have an excellent **BA**, but due to a poor **SA** value, it probably cannot be synthesized. Nevertheless, it can be advantageous to cover a more diverse distribution of the individual metrics through **PREMS**. This way, a desired weighting can still be applied during the manual selection. The results of **WSEMS** runs, on the other hand, are more similar with respect to the metrics. However, both methods can outperform each other in terms of metrics for known ligands such as N3 and Lopanivir. Regarding the generated molecules, we made three general observations: aromatic rings are often included, molecules with a good **QED** value tend to have a smaller size, and molecules with an extremely high **BA** often have an unrealistic structure in return. In summary, the molecules generated by our method are a promising basis for further studies.

Next, we look at the results of the **EMGA** and corresponding **MD** analysis. First, we found that the continuous improvement that takes place during the evolutionary process stagnates at a fitness value of 0.225 around the 70th generation. Therefore, we ran the **genetic algorithm (GA)** for 80 generations. **EMGA** generated 144 350 molecules during the 16 runs and calculated their metrics. From the best 200 molecules, 21 were manually selected for further study. We studied these 21 ligands using **MD** simulations. A ligand was placed in the **M^{pro}** pocket and then the motions of the individual atoms and the whole ligand were calculated for a period of 50 ns. As part of the simulation, we determined the values of **center of mass (COM)**, **root mean square displacement (RMSD)**, and **root means square fluctuations (RMSF)**, which allowed us to discard some molecules because they were unlikely to bond. For the remaining 6 ligands, we calculated the binding free energy estimates and identified two particularly good drug candidates. By estimating the docking more accurately with the **MD** analysis, we were not just able to discard some molecules, but also to find some promising candidates. It can be concluded that the very fast **BA** calculation (a few minutes) with **QuickVina 2** sometimes leads to imprecise results, but can be used for a quick pre-selection before the tedious **MD** analysis (which would take 2 to 3 weeks on the **high performance cluster (HPC)**).

A direct comparison of the two approaches is only partially possible, since we used different molecule representations as well as different **EAs**. However, we can compare the fitness values of the best molecules since we used the same metrics for this purpose. While the fitness values of **WSEMS** and **PREMS** are both around 0.3, better values around 0.25 are achieved with **EMGA**. We can also compare the two string representations used. **SELFIES** seems to be well suited for a **GA**, since only valid molecules can be described. But this is often done by simply stopping the string interpretation at the position where an invalid token is located. This often results in short molecules, and changes at the end of the string have no effect. The **SMILES** representation does not have this problem, but can lead to invalid molecules instead. However, we were able to show that this problem can be well solved using a **LM**.

In conclusion, we have developed two methods that are able to improve molecules step by step and to study numerous molecules relatively quickly. In Chapter 6 we have found that our method serves well as a basis for the subsequent MD analysis and allows a rapid pre-selection for this purpose.

Future Work

In the field of molecule design, many extensions and variants are conceivable. EMGA could be extended with other multi-objective variants. For this purpose either a NSGA-II or another method like a *S* metric selection evolutionary multiobjective optimisation algorithm (SMS-EMOA) [BNR08] could be used. The internal parameters of EAs and LM could also be explored further. The LM parameters r_{\max} and δ_{\max} , which affect the length of the substring to be changed and thus the strength of the change, could be adaptively adjusted. This could improve both exploration and exploitation. We used molecular design to find a protease inhibitor for M^{pro} from SARS-CoV-2. However, the same procedure could be used for other viruses. All that would be needed is to know the virus protease and the pocket – i.e., the area where the protease inhibitor should dock – then fit both of them into our model. By adapting the metrics, the method could be used to search for molecules in other fields, such as biology or materials research. We have covered the first steps of drug development. The further steps, which could be carried out by chemists, would be the synthesis of the selected molecules and an *in vitro* analysis. This is the only way to experimentally determine how the molecules actually behave in reality. The use of artificial intelligence (AI) methods can also be useful for these further steps, e.g., to find a synthesis pathway.

8.3 Convolutional Self-Organizing Map

In Part IV, we introduced the convolutional self-organizing map (ConvSOM) – a combination of convolutional neural network (CNN) and self-organizing map (SOM). The ConvSOM is used to create a two-dimensional visualization of data, taking into account higher-level features. The ConvSOM uses the internal intermediate results of a pre-trained CNN, which are kept unchanged for further processing. We used image data as input, but the ConvSOM could be applied to other data as well. To evaluate the results, we used the existing Kruskal Shepard Error (\hat{E}_{KSN}) and developed new metrics that incorporate label information: Cross Entropy (H_{SOM}), normalized minor class occurrence (NMCO), and class scatter index (CSI). These new metrics examine in different ways how samples of the same class are arranged on the map. For example, CSI examines how many clusters samples of the same class form on average. Furthermore, a semantic metric based on deep visual-semantic embedding model (DeViSE) has been developed. It is able to evaluate the semantic distribution of the images without needing the label information. As an additional visualization of the distribution of the classes associated with the samples on the resulting map, we developed the *sample class plot*. Here, for each neuron of the SOM, we count how many samples of a given class were assigned to it as best matching unit (BMU). The distribution is then plotted using different colors and color intensities.

In the experimental analysis, we used small subsets of each of the well-known MNIST and CIFAR-10 data sets as inputs. Regarding the new metrics that take the label into account, the inputs from higher CNN layers lead to better results. This is especially true for the class-wise CSI. This is to be

expected, since the CNN is trained to classify images and ultimately tries to determine a label. Overall, we can state that the use of higher-order features leads to an improvement of the results in terms of labeling errors.

Future Work

In the future, the ConvSOM could be applied to more data sets. For image data, those with multiple objects per image could be examined. In the previous experiments, a CNN was pre-trained on a part of the same data set with data that was not used later. In order to be able to apply the procedure as a whole to unsupervised data, the CNN could be pre-trained on an extraneous, more general data set. However, the approach could also be extended to non-image data. For example, the time series of company data from Part II could be used as input to provide an overview of companies and their similarity.

Another application comes from the molecule design part. A selection of molecules could be ordered based on their higher order features. Higher-order features could be the number of certain structures contained, combined with the results of the molecule metrics. For this case, instead of a CNN a graph neural network (GNN), like the message passing neural network (MPNN) [Gil+17] could be used. The MPNN can be used to create fix sized fingerprints of molecules [Duv+15]. These fingerprints contain information on the structure of the molecules. The fingerprints information could be enriched by the molecule metric results and given to a SOM. Additional application-specific metrics may be defined to evaluate the molecular map result. In this way, the selection of candidate molecules could be further simplified.

Part VI

Appendices

A Data Sets

Table A.1: Image recognition data sets

Name	Size	Color	Train	Test	Classes	Notes
MNIST ¹	28 × 28	grayscale	60.000	10.000	10	
Fashion MNIST ²	28 × 28	grayscale	60.000	10.000	10	
notMNIST ³	28 × 28	grayscale	60.000	10.000	10	not hand-cleaned, 6.5 % label error
CIFAR-10 ⁴	32 × 32	RGB	50.000	10.000	10	
CIFAR-100 ⁵	32 × 32	RGB	50.000	10.000	100	
STL-10 ⁶	96 × 96	RGB	5.000	8.000	10	100000 unlabeled images for unsupervised learning
SVHN ⁷	32 × 32	RGB	73.257	26.032	10	
Dogs vs. Cats ⁸	varies	RGB	25.000	–	2	size: (300x401), (500x377), (239x360), ...

¹website: <http://yann.lecun.com/exdb/mnist/>

²github <https://github.com/zalando-research/fashion-mnist>, kaggle: <https://www.kaggle.com/zalando-research/fashionmnist>

³kaggle: <https://www.kaggle.com/jwjohnson314/notmnist/data>

⁴website: <https://www.cs.toronto.edu/~kriz/cifar.html>

⁵website: <https://www.cs.toronto.edu/~kriz/cifar.html>

⁶website: <https://cs.stanford.edu/~acoates/stl10/>

⁷website: <http://ufldl.stanford.edu/housenumbers/>

⁸kaggle: <https://www.kaggle.com/c/dogs-vs-cats/data>, microsoft: <https://www.microsoft.com/en-us/download/details.aspx?id=54765>

B Earnings Prediction

B.1 Data Sets

Table B.1: Used parameters of data set COMPUTSTAT QUARTERLY. The parameters (in blue) above the dividing line are only used for the assignment and selection of the samples. Below are the 19 features, that are given for each quarter of a COMPUTSTAT QUARTERLY sample.

Parameter	Description
<code>cusip</code>	unique identifier of the company
<code>fpedats</code>	date of quarter
<code>ffi5</code>	group number in FFI with 5 subdivisions
<code>ffi10</code>	group number in FFI with 10 subdivisions
<code>ffi12</code>	group number in FFI with 12 subdivisions
<code>ffi48</code>	group number in FFI with 48 subdivisions
<code>financialfirm</code>	Boolean, is financial firm
<code>EPS_Mean_Analyst</code>	mean of all available analyst estimates
<code>rdq</code>	Report Date of Quarterly Earnings
<code>epsfig</code>	Earning per Share
<code>atq</code>	Total Assets
<code>revtq</code>	Revenue
<code>nopiq</code>	Nonoperating Income
<code>xoprq</code>	Operating Expenses – Total
<code>apq</code>	Accounts Payable – Trade
<code>gdwlq</code>	Goodwill Net
<code>rectq</code>	Receivables – Total
<code>xrdq</code>	Research and Development Expense
<code>cogsq</code>	Cost of Goods Sold
<code>rcpq</code>	Restructuring Costs Pretax
<code>ceqq</code>	Common Equity Total
<code>niq</code>	Net Income (Loss)
<code>oiadpq</code>	Operating Income After Depreciation
<code>oibdpq</code>	Operating Income Before Depreciation
<code>dpq</code>	Depreciation and Amortization Total
<code>ppentq</code>	Property Plant and Equipment
<code>piq</code>	Pretax Income
<code>txtq</code>	Income Taxes Total

Table B.2: Used parameters of data set DAILY SHARES. The parameters (in blue) above the dividing line are only used for the assignment and selection of the samples. The parameters marked in orange are derived from the above ones and are also used for the sample. Thus, one sample of DAILY SHARES consists of 11 features per day.

Parameter	Description
cusip	unique identifier of the company
date	date to which the following variables refer
ret	daily return (change in share price) in percent
prc	scaled share price
vol	number of shares traded on that day
shrout	number of tradable shares (in thousands)
wvret	market return of all major companies (S&P 500) (as macro variable)
rel_vol	relative trade volume: $rel_vol = \frac{vol}{1000 \cdot shrout}$
Monday	} current day one hot encoded
Tuesday	
Wednesday	
Thursday	
Friday	

B.2 Workflow

Figure B.1 shows the workflow of the finance experiments. Given are the data sets COMPUTSTAT YEARLY, COMPUTSTAT QUARTERLY and DAILY SHARES. The `data_preprocessor.py` reduces these data sets to the most important parameters and handles outliers and gaps as described in Section 3.5, resulting in a small version of the data sets. The `shares_separator.py` sorts and separates the data to one file per month, so the needed data can be accessed more efficiently.

The `company_sample_builder.py` creates samples for the companies, depending on a number of parameters (e.g., quarterly or yearly, time horizon, imputation method) in a python readable pickle format (`samples/*_YYYY.pkl`). The `shares_sample_builder.py` creates samples of daily shares analogous to the `company_sample_builder.py`. The created shares samples contain exactly the data for the samples from the associated sample quarter or year (`samples/*_YYYY.pkl`). Thus, the numpy array contained in the pkl file contains exactly the same number of rows. Missing data is filled with zeros.

Now that the data has been preprocessed and samples have been created, they can be given to the `ann_predictor.py` for forecasting. Besides the main result (`predictions.tsv`), there is also saved some information about the course of training of the **artificial neural network (ANN)** (e.g., loss and validation loss per epoch) in `history.tsv` and `history_rv.tsv` (rounded values). In addition, the structure of the ANN is saved as an image in `model.png`.

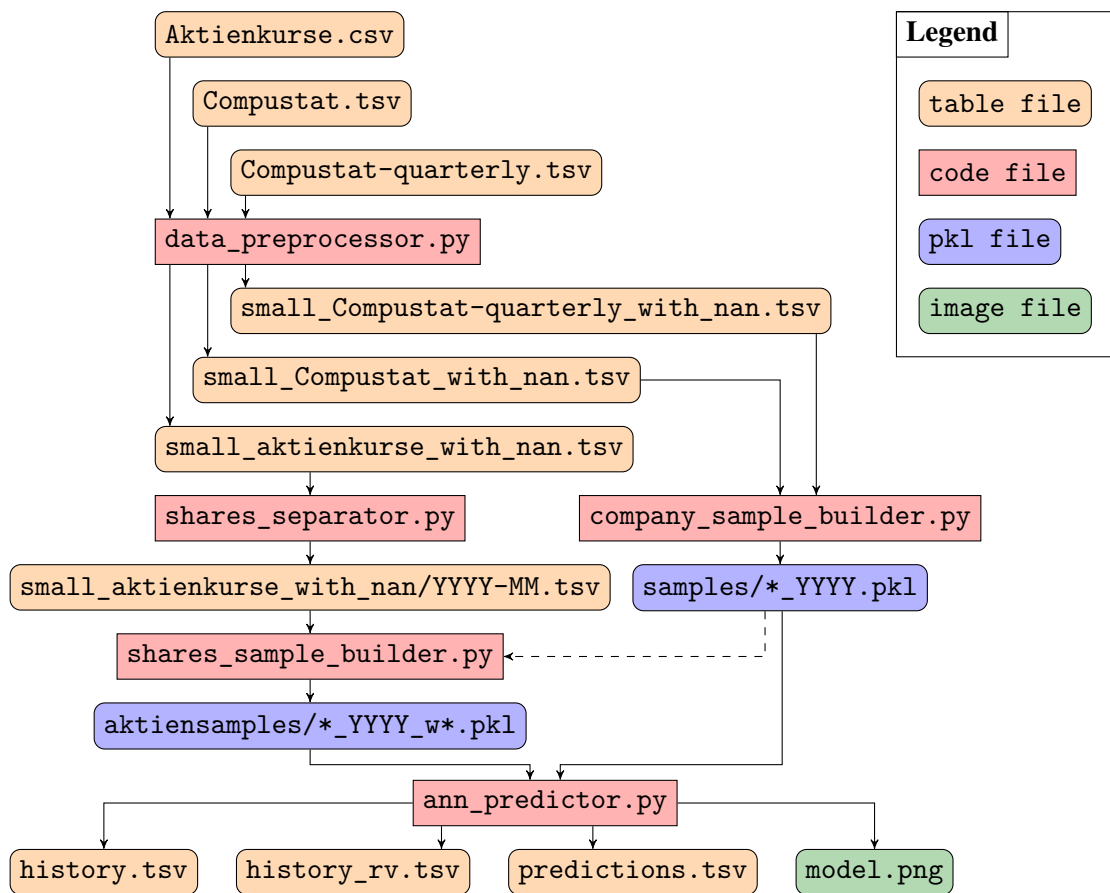


Figure B.1: Workflow of the finance experiments. The legend in the upper right defines the color coding of the nodes.

C Molecule Design

C.1 Workflow

Figure C.1 shows the workflow used to calculate the molecule metrics. This workflow is used by Chapter 5 and Chapter 6, with the latter omitting the conversion of **SELFIES** to **SMILES**, since the individuals of the **EA** are already present there as **SMILES**. Various existing packages and tools are used to convert the file formats. The processing steps for the ligand generated by the **GA** are shown on the left. The protein to which it will be docked is shown on the right. Both are passed to QuickVina2 in PDBQT format to determine the **BA**. The other 4 metrics are calculated using **MOSES**.

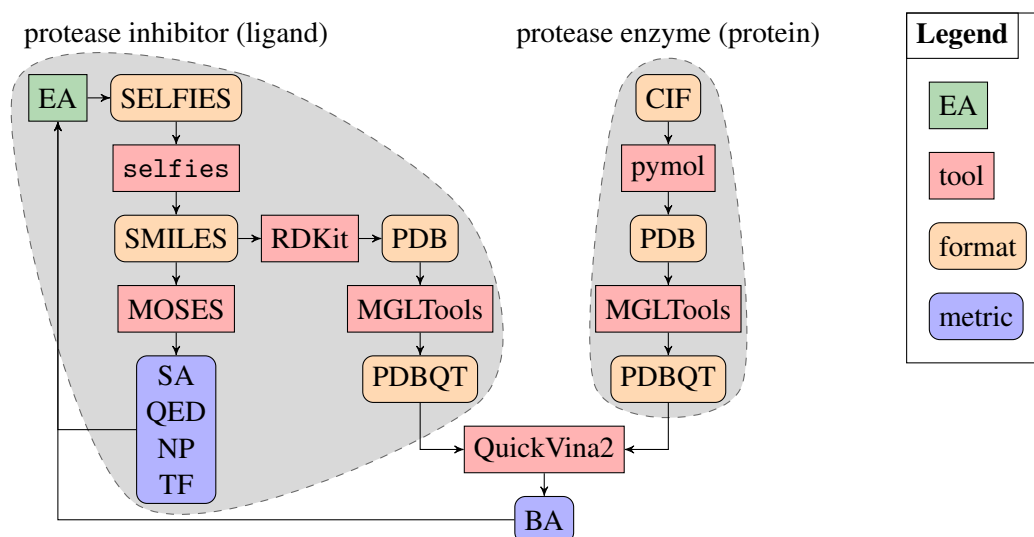


Figure C.1: Workflow that is used to calculate the molecule metrics, described in Section 5.2. The legend in the upper right defines the color coding of the nodes.

C.2 Ligands Overview

Table C.1 shows an overview of the selected ligands. The **SMILES** representation and the results for the five metrics are listed.

Table C.1: Overview of the SMILES representation and molecule design metrics of the 21 ligands, that were selected by EMGA in Section 6.3.1. The metrics are described in Section 5.2.

Ligand	BA	SA	QED	NP	TF	SMILES
Lig ₁	-8.1	1.019	0.918	-0.089	1	<chem>Cc1ccn(CC(=O)N2CC=CC(c3cnc(O)cc3C)=CC2)c(=O)c1</chem>
Lig ₂	-10.4	3.119	0.699	-0.272	1	<chem>CC1C(c2cc(C#N)enc2NC=C2C(=O)NCC3=COCN32)=NN=C2C=CC=NN21</chem>
Lig ₃	-10.2	2.543	0.783	-0.133	1	<chem>C#Cc1c(F)c(F)cc(C)c1NC(=O)NC1=CC=C2C=CCN=C2OC1C</chem>
Lig ₄	-9.4	1.000	0.850	-0.212	1	<chem>CC1N=CC(F)C(CC=c2[nH]c(=O)cc3c2=C(F)C(C#N)=CC(F)=C3)C1=O</chem>
Lig ₅	-9.5	1.254	0.848	-0.094	1	<chem>CC1=CC=C(CC=c2[nH]c(=O)cc3c2=C(F)C(N)=CC(F)=C3)C(F)C=N1</chem>
Lig ₆	-8.7	1.000	0.861	-0.256	1	<chem>CC1=CC=C(C2NC(=O)CCc3ccc(F)c(C#N)c32)C(F)C=N1</chem>
Lig ₇	-11.8	6.664	0.676	-0.174	1	<chem>CC1(C)CC(c2cc(C#N)enc2NC=C2C(=O)CCC3=CC=COCN32)=NN=C2C=CC=NN21</chem>
Lig ₈	-11.4	6.745	0.615	-0.238	0	<chem>CC1C(C2=CC(C#N)=CC=CN=C2NC=C2C(=O)NCC3=COCN32)=NN=C2C=CC=NN21</chem>
Lig ₉	-8.4	1.000	0.898	-0.248	1	<chem>C=C(C)C1=C(C(=O)N(C)c2ccc(F)c(F)c2)N2N=C(O)OC2CC=C1</chem>
Lig ₁₀	-8.5	1.000	0.890	-0.149	1	<chem>CC1=CC=C(C=CN2C(=NCO)CCOc3ccc(C#N)c(C#N)c32)C1F</chem>
Lig ₁₁	-9.5	1.695	0.783	-0.183	1	<chem>CC1(C)CC(F)C(C)(OC2=CNC(=O)C=C3N=C(F)C=C(C#N)C(F)=C32)C1=O</chem>
Lig ₁₂	-9.6	1.342	0.799	-0.082	1	<chem>CC1=CC=C(N=CC=c2[nH]c(=O)cc3c2=C(C)C(F)=CC(F)=C3)C(F)C=N1</chem>
Lig ₁₃	-10.3	2.252	0.787	-0.183	1	<chem>CC1CC(F)(F)CC(CC=C2[nH]c(=O)cc3c2=C(F)C(C#N)=CC(F)=C3)C1=O</chem>
Lig ₁₄	-11.8	6.567	0.662	-0.174	0	<chem>CC1CC(F)(F)CC(CC=C2NC=CC(=O)C=C3C=C(F)C=C(C#N)C(F)=C32)C1=O</chem>
Lig ₁₅	-9.0	1.477	0.796	-0.083	1	<chem>CC1(C)COCC(CC=c2[nH]c(=O)c(O)c3c2=C(F)C(C#N)=CC(F)=C3)C1=O</chem>
Lig ₁₆	-10.0	2.586	0.803	-0.118	1	<chem>CC1(C)CCCC(OC2=CNC(=O)C=C3C=C(F)C=C(C#N)C(F)=C32)C1=O</chem>
Lig ₁₇	-10.0	2.692	0.858	-0.097	1	<chem>CC1(C)COC(C)(CC=c2[nH]c(=O)cc3c2=C(F)C(C#N)=CC(F)=C3)C1=O</chem>
Lig ₁₈	-11.5	5.985	0.523	-0.235	0	<chem>CC1C=CC(F)(F)CC(CC=C2NC(=O)C=NC=C3C=C(F)C=C(C#N)C(F)=C32)C1=O</chem>
Lig ₁₉	-11.5	6.102	0.527	-0.209	1	<chem>CC1C=CC(F)(F)CC(CC=C2NNC(=O)C=C3C=C(F)C=C(C#N)C(F)=C32)C1=O</chem>
Lig ₂₀	-10.9	4.036	0.756	-0.149	1	<chem>CC1N=CC(F)C(C=CC=c2[nH]c(=O)cc3c2=C(F)C(C#N)=CC(F)=C3)OC1=O</chem>
Lig ₂₁	-9.0	1.813	0.844	-0.103	1	<chem>CCOC(=c1[nH]c(=O)cc2c1=C(F)C(C#N)=CC(F)=C2)C1OCC(C)(C)C1=O</chem>

List of Figures

2.1	Overview of the CI algorithms that are used in this thesis	7
2.2	An example of an image classification problem	8
2.3	Overview of training, validation, and test sets	9
2.4	Example partition of a data set using cross-validation	10
2.5	General setup of reinforcement learning	10
2.6	Structure of an artificial neuron n_i	11
2.7	Activation functions	12
2.8	Combination of several neurons to solve a classification problem with two dimensions	13
2.9	Feed forwards networks	14
3.1	Time series model for prediction of the data of the next time step	19
3.2	LSTM cell	19
3.3	A dilated causal convolutional network with 3 layers	20
3.4	Data set horizons of training, validation, and test set	23
3.5	Visualization of selected LSTM and TCN architectures	25
4.1	Protease Process and Protease Inhibition	31
4.2	Bohr models of elements H, C, N, O, and F	33
4.3	Periodic table with the important organic elements	34
4.4	Visualization of C and H atoms using the VSEPR model	35
4.5	The ionic bond of NaCl represented by Bohr models	36
4.6	Schematic of a positively charged metal ion grid with freely movable electrons	36
4.7	Hydrogen bonds in general and in water	36
4.8	Visualizations of SMILES and SELFIES representations	38
4.9	Benzene molecule	39
4.10	Different 2D-representations of the molecule propane	40
4.11	Different representations of the α -D-glucopyranose molecule, which contains a ring .	41
4.12	3D models of propane	41
4.13	Similar molecules with different atomic distances and angles	42
4.14	A chiral molecule	43
4.15	A chiral molecule with one <i>cis</i> and one <i>trans</i> isomer	43
4.16	Activity diagram of an evolutionary algorithm	46
5.1	Exemplary molecules for high and low values of SA, QED, and NP	52
5.2	Overview of the evolutionary molecule search	54

5.3	Application example of the mutation operations	55
5.4	The soft clipping function and its application to BA	55
5.5	NSGA-II domination	57
5.6	Development of all Metrics	58
5.7	Visualization of Pareto fronts over the generations a single PREMS run	59
5.8	Comparison of final generation molecules from WSEMS and PREMS runs	60
5.9	Spider charts, structural formula, and chemical names of exemplary protease inhibitors	61
5.10	The docking of PI-I and PI-V to the pocket of SARS-CoV-2's M ^{pro}	61
6.1	Overview of the language model-based evolutionary approach	63
6.2	Structural formula, SMILES string, and token representation of α -D-glucopyranose	65
6.3	Language model as mutation operator	66
6.4	Activity diagram of EMGA	67
6.5	Evolutionary Process of EMGA	71
6.6	Selection of ligands created by EMGA	72
6.6	Selection of ligands created by EMGA	73
6.7	Binding site of M ^{pro} defined by the labeled residues with Lig ₁₉ in its initial bound pose	74
6.8	Position of Lig ₁₉ in M ^{pro} at different simulation times	74
6.9	COM distance over time between M ^{pro} binding site and drifting ligands	75
6.10	RMSD of the simulated ligands	76
6.11	Position and orientation of Lig ₁₃ and Lig ₈ in M ^{pro} at different simulation times	77
7.1	Update step of an example SOM	83
7.2	CNN example	85
7.3	Structure of the ConvSOM and the associated quality measures	86
7.4	Visualization plots for SOM results	89
7.5	Sample class plot legend	90
7.6	Box-plots of class-wise CSI of SOM and ConvSOM on MNIST and CIFAR-10	92
7.7	SOM and ConvSOM training results on MNIST and CIFAR-10 with 500 samples . . .	93
7.8	Structure of the ConvSOM extended by the DeVISE error	94
B.1	Workflow of the finance experiments	109
C.1	Workflow of the molecule metrics calculation	111

List of Tables

3.1	Overview of different network architectures with and without daily shares layers . . .	24
3.2	Selected architectures and parameters for financial, non-financial and all companies .	25
3.3	Overview of the FFI5 classes	26
3.4	Results of the selected architectures and parameters for the FFI5 groups	26
3.5	Results data set B for non-financials with preselected architectures and parameters .	27
3.6	Results data set B for FFI5 group 3 with preselected architectures and parameters . .	27
4.1	CPK coloring scheme for atoms by Corey, Pauling and Koltun	42
4.2	Overview of molecule representations and their properties	44
5.1	Value ranges and optima of the used metrics	51
5.2	Experimental results of WSEMS and PREMS runs	59
6.1	Averaged values of COM, RMSD, and RMSF	75
6.2	Binding free energy estimates, ΔG^0	78
7.1	Experimental results of ConvSOM on MNIST	91
7.2	Experimental results of ConvSOM on CIFAR-10	92
A.1	Image recognition data sets	105
B.1	Used parameters of data set COMPUTSTAT QUARTERLY	107
B.2	Used parameters of data set DAILY SHARES	108
C.1	SMILES representation and molecule metrics of the 21 ligands	112

Bibliography

- [15] *Organic Chemistry*. Chemistry LibreTexts. Feb. 1, 2015. URL: https://chem.libretexts.org/Bookshelves/Organic_Chemistry (visited on 01/21/2022).
- [AA20] Saleh Aly and Sultan Almotairi. “Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition”. In: *IEEE Access* 8 (2020), pp. 107035–107045. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2020.3000829](https://doi.org/10.1109/ACCESS.2020.3000829).
- [Alh+15] Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. “Fast, Accurate, and Reliable Molecular Docking with QuickVina 2”. In: *Bioinformatics* 31.13 (July 1, 2015), pp. 2214–2216. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btv082](https://doi.org/10.1093/bioinformatics/btv082).
- [Ana+03] Kanchan Anand, John Ziebuhr, Parvesh Wadhvani, Jeroen R. Mesters, and Rolf Hilgenfeld. “Coronavirus Main Proteinase (3CLpro) Structure: Basis for Design of Anti-SARS Drugs”. In: *Science* 300.5626 (June 13, 2003), pp. 1763–1767. DOI: [10.1126/science.1085658](https://doi.org/10.1126/science.1085658).
- [And02] Péter András. “Kernel-Kohonen Networks”. In: *International Journal of Neural Systems* 12.02 (Apr. 1, 2002), pp. 117–135. ISSN: 0129-0657. DOI: [10.1142/S0129065702001084](https://doi.org/10.1142/S0129065702001084).
- [And90] John Robert Anderson. *Machine Learning: An Artificial Intelligence Approach*. Vol. 3. Morgan Kaufmann, 1990.
- [Ars+21] Amir Hossein Arshia, Shayan Shadravan, Aida Solhjo, Amirhossein Sakhteman, and Ashkan Sami. “De Novo Design of Novel Protease Inhibitor Candidates in the Treatment of SARS-CoV-2 Using Deep Learning, Docking, and Molecular Dynamic Simulations”. In: *Computers in Biology and Medicine* 139 (Dec. 1, 2021), p. 104967. ISSN: 0010-4825. DOI: [10.1016/j.combiomed.2021.104967](https://doi.org/10.1016/j.combiomed.2021.104967).
- [Aru+11] Elangannan Arunan, Gautam R. Desiraju, Roger A. Klein, Joanna Sadlej, Steve Scheiner, Ibon Alkorta, David C. Clary, Robert H. Crabtree, Joseph J. Dannenberg, Pavel Hobza, Henrik G. Kjaergaard, Anthony C. Legon, Benedetta Mennucci, and David J. Nesbitt. “Definition of the hydrogen bond (IUPAC Recommendations 2011)”. In: *Pure and Applied Chemistry* 83.8 (July 8, 2011), pp. 1637–1641. ISSN: 1365-3075. DOI: [10.1351/PAC-REC-10-01-02](https://doi.org/10.1351/PAC-REC-10-01-02).
- [AZ11] Divya Anantharaman and Yuan Zhang. “Cover Me: Managers’ Responses to Changes in Analyst Coverage in the Post-Regulation FD Period”. In: *The Accounting Review* 86.6 (Nov. 1, 2011), pp. 1851–1885. ISSN: 0001-4826. DOI: [10.2308/accr-10126](https://doi.org/10.2308/accr-10126).

- [Bao+20] Yang Bao, Bin Ke, Bin Li, Y. Julia Yu, and Jie Zhang. “Detecting Accounting Fraud in Publicly Traded U.S. Firms Using a Machine Learning Approach”. In: *Journal of Accounting Research* 58.1 (2020), pp. 199–235. ISSN: 1475-679X. DOI: [10.1111/1475-679X.12292](https://doi.org/10.1111/1475-679X.12292).
- [Ben09] Yoshua Bengio. “Learning Deep Architectures for AI”. In: *Foundations and Trends® in Machine Learning* 2.1 (Nov. 14, 2009), pp. 1–127. ISSN: 1935-8237, 1935-8245. DOI: [10.1561/2200000006](https://doi.org/10.1561/2200000006).
- [Ber+20] R. Bernardi, M. Bhandarkar, A. Bhatele, E. Bohm, R. Brunner, R. Buch, F. Buelens, H. Chen, C. Chipot, A. Dalke, S. Dixit, G. Fiorin, P. Freddolino, H. Fu, P. Grayson, J. Gullingsrud, A. Gursoy, D. Hardy, C. Harrison, J. Hénin, W. Humphrey, D. Hurwitz, A. Hynninen, N. Jain, W. Jiang, N. Krawetz, S. Kumar, D. Kunzman, J. Lai, C. Lee, J. Maia, R. McGreevy, C. Mei, M. Melo, M. Nelson, J. Phillips, B. Radak, J. Ribeiro, T. Rudack, O. Sarood, A. Shinozaki, D. Tanner, P. Wang, D. Wells, G. Zheng, and F. Zhu. *NAMD 2.14 User’s Guide*. Aug. 5, 2020. URL: <https://www.ks.uiuc.edu/Research/namd/2.14/ug/> (visited on 03/26/2022).
- [Bey92] Hans-Georg Beyer. “Some Aspects of the ‘Evolution Strategy’ for Solving TSP-like Optimization Problems”. In: *Parallel problem solving from nature 2* (1992), pp. 361–370.
- [BG17] Ryan T. Ball and Eric Ghysels. “Automated Earnings Forecasts: Beat Analysts or Combine and Conquer?” In: *Management Science* 64.10 (Oct. 27, 2017), pp. 4936–4952. ISSN: 0025-1909. DOI: [10.1287/mnsc.2017.2864](https://doi.org/10.1287/mnsc.2017.2864).
- [BH10] Jonathan B. Baell and Georgina A. Holloway. “New Substructure Filters for Removal of Pan Assay Interference Compounds (PAINS) from Screening Libraries and for Their Exclusion in Bioassays”. In: *Journal of Medicinal Chemistry* 53.7 (Apr. 8, 2010), pp. 2719–2740. ISSN: 0022-2623. DOI: [10.1021/jm901137j](https://doi.org/10.1021/jm901137j).
- [Bha+21] Vijay Kumar Bhardwaj, Rahul Singh, Pralay Das, and Rituraj Purohit. “Evaluation of Acridinedione Analogs as Potential SARS-CoV-2 Main Protease Inhibitors and Their Comparison with Repurposed Anti-Viral Drugs”. In: *Computers in Biology and Medicine* 128 (Jan. 1, 2021), p. 104117. ISSN: 0010-4825. DOI: [10.1016/j.combiomed.2020.104117](https://doi.org/10.1016/j.combiomed.2020.104117).
- [Bic+12] G. Richard Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. “Quantifying the Chemical Beauty of Drugs”. In: *Nature Chemistry* 4.2 (2 Feb. 2012), pp. 90–98. ISSN: 1755-4349. DOI: [10.1038/nchem.1243](https://doi.org/10.1038/nchem.1243).
- [BKK18] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”. In: *CoRR* abs/1803.01271 (2018). arXiv: [1803.01271](https://arxiv.org/abs/1803.01271). URL: <http://arxiv.org/abs/1803.01271>.

- [BNR08] Nicola Beume, Boris Naujoks, and Günter Rudolph. “SMS-EMOA – Effektive evolutionäre Mehrzieloptimierung (SMS-EMOA – Effective Evolutionary Multiobjective Optimization)”. In: 56.7 (July 1, 2008), pp. 357–364. ISSN: 2196-677X. DOI: [10.1524/auto.2008.0715](https://doi.org/10.1524/auto.2008.0715).
- [Bri+17] Martin Brieg, Julia Setzler, Steffen Albert, and Wolfgang Wenzel. “Generalized Born Implicit Solvent Models for Small Molecule Hydration Free Energies”. In: *Physical Chemistry Chemical Physics* 19.2 (Jan. 4, 2017), pp. 1677–1685. ISSN: 1463-9084. DOI: [10.1039/C6CP07347F](https://doi.org/10.1039/C6CP07347F).
- [Bro+04] Nathan Brown, Ben McKay, François Gilardoni, and Johann Gasteiger. “A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules”. In: *Journal of Chemical Information and Computer Sciences* 44.3 (May 1, 2004), pp. 1079–1087. ISSN: 0095-2338. DOI: [10.1021/ci034290p](https://doi.org/10.1021/ci034290p).
- [Bro+19] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. “GuacaMol: Benchmarking Models for de Novo Molecular Design”. In: *Journal of Chemical Information and Modeling* 59.3 (Mar. 25, 2019), pp. 1096–1108. ISSN: 1549-9596. DOI: [10.1021/acs.jcim.8b00839](https://doi.org/10.1021/acs.jcim.8b00839).
- [Bru11] Paula Yurkanis Bruice. *Organische Chemie : Studieren kompakt. 5., aktualisierte Auflage. Always learning.* 2011. ISBN: 978-3-86326-607-3.
- [Brü92] Axel T. Brünger. *X-PLOR: Version 3.1 : A System for X-ray Crystallography and NMR.* Yale University Press, Jan. 1, 1992. 404 pp. ISBN: 978-0-300-05402-6. Google Books: [nFK5_F5hhJ0C](https://books.google.com/books?id=nFK5_F5hhJ0C).
- [BS02] Hans-Georg Beyer and Hans-Paul Schwefel. “Evolution Strategies – A Comprehensive Introduction”. In: *Natural Computing* 1.1 (Mar. 1, 2002), pp. 3–52. ISSN: 1572-9796. DOI: [10.1023/A:1015059928466](https://doi.org/10.1023/A:1015059928466).
- [Cas+] D.A. Case, H.M. Aktulga, K. Belfon, I.Y. Ben-Shalom, S.R. Brozell, D.S. Cerutti, T.E. Cheatham, G.A. Cisneros, V.W.D. Cruzeiro, T.A. Darden, R.E. Duke, G. Giambasu, M.K. Gilson, H. Gohlke, A.W. Goetz, R. Harris, S. Izadi, S.A. Izmailov, C. Jin, K. Kasavajhala, M.C. Kaymak, E. King, A. Kovalenko, T. Kurtzman, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, M. Machado, V. Man, M. Manathunga, K.M. Merz, Y. Miao, O. Mikhailovskii, G. Monard, H. Nguyen, K.A. O’Hearn, A. Onufriev, F. Pan, S. Pantano, R. Qi, A. Rahnamoun, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C.L. Simmerling, N.R. Skrynnikov, J. Smith, J. Swails, R.C. Walker, J. Wang, H. Wei, R.M. Wolf, X. Wu, Y. Xue, D.M. York, S. Zhao, and P.A. Kollman. *AmberTools21*. URL: <https://ambermd.org/AmberTools.php> (visited on 03/26/2022).
- [CBA15] Vikas Chaudhary, R. S. Bhatia, and Anil K. Ahlawat. “Community SOM (CSOM): An Improved Self-Organizing Map Learning Technique”. In: *International Journal of Fuzzy Systems* 17.2 (June 1, 2015), pp. 129–132. ISSN: 2199-3211. DOI: [10.1007/s40815-015-0022-7](https://doi.org/10.1007/s40815-015-0022-7).

- [CBM17] D. N. Coelho, G. A. Barreto, and C. M. S. Medeiros. “Detection of Short Circuit Faults in 3-Phase Converter-Fed Induction Motors Using Kernel SOMs”. In: *WSOM. 2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*. June 2017, pp. 1–7. DOI: [10.1109/WSOM.2017.8020016](https://doi.org/10.1109/WSOM.2017.8020016).
- [Cec+10] Mark Cecchini, Haldun Aytug, Gary J. Koehler, and Praveen Pathak. “Detecting Management Fraud in Public Companies”. In: *Management Science* 56.7 (May 17, 2010), pp. 1146–1160. ISSN: 0025-1909. DOI: [10.1287/mnsc.1100.1174](https://doi.org/10.1287/mnsc.1100.1174).
- [CEK20] Tim Cofala, Lars Elend, and Oliver Kramer. “Tournament Selection Improves Cartesian Genetic Programming for Atari Games”. In: *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*. 2020, pp. 345–350. URL: <https://www.esann.org/sites/default/files/proceedings/2020/ES2020-204.pdf> (visited on 10/25/2021).
- [Cha+16] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. “Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Mar. 2016, pp. 4960–4964. DOI: [10.1109/ICASSP.2016.7472621](https://doi.org/10.1109/ICASSP.2016.7472621).
- [Cof+20] Tim Cofala, Lars Elend, Philip Mirbach, Jonas Prellberg, Thomas Teusch, and Oliver Kramer. “Evolutionary Multi-objective Design of SARS-CoV-2 Protease Inhibitor Candidates”. In: *Parallel Problem Solving from Nature – PPSN XVI*. Ed. by Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 357–371. ISBN: 978-3-030-58115-2. DOI: [10.1007/978-3-030-58115-2_25](https://doi.org/10.1007/978-3-030-58115-2_25).
- [CRS] CRSP. *Research Data | CRSP - The Center for Research in Security Prices*. URL: <https://www.crsp.org/products/research-products> (visited on 08/28/2023).
- [Dai+20] Wenhao Dai, Bing Zhang, Haixia Su, Jian Li, Yao Zhao, Xiong Xie, Zhenming Jin, Fengjiang Liu, Chunpu Li, You Li, Fang Bai, Haofeng Wang, Xi Cheng, Xiaobo Cen, Shulei Hu, Xiuna Yang, Jiang Wang, Xiang Liu, Gengfu Xiao, Hualiang Jiang, Zihe Rao, Lei-Ke Zhang, Yechun Xu, Haitao Yang, and Hong Liu. “Structure-Based Design of Antiviral Drug Candidates Targeting the SARS-CoV-2 Main Protease”. In: *Science* (Apr. 22, 2020). ISSN: 0036-8075, 1095-9203. DOI: [10.1126/science.abb4489](https://doi.org/10.1126/science.abb4489). pmid: 32321856.
- [Deb+02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197. ISSN: 1941-0026. DOI: [10.1109/4235.996017](https://doi.org/10.1109/4235.996017).

- [Dec+11] Patricia M. Dechow, Weili Ge, Chad R. Larson, and Richard G. Sloan. “Predicting Material Accounting Misstatements”. In: *Contemporary Accounting Research* 28.1 (2011), pp. 17–82. ISSN: 1911-3846. DOI: [10.1111/j.1911-3846.2010.01041.x](https://doi.org/10.1111/j.1911-3846.2010.01041.x).
- [DNA16] H. Dozono, G. Niina, and S. Araki. “Convolutional Self Organizing Map”. In: *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. Dec. 2016, pp. 767–771. DOI: [10.1109/CSCI.2016.0149](https://doi.org/10.1109/CSCI.2016.0149).
- [DSC15] R. Vasundhara Devi, S. Siva Sathya, and Mohane Selvaraj Coumar. “Evolutionary Algorithms for de Novo Drug Design – A Survey”. In: *Applied Soft Computing* 27 (Feb. 1, 2015), pp. 543–552. ISSN: 1568-4946. DOI: [10.1016/j.asoc.2014.09.042](https://doi.org/10.1016/j.asoc.2014.09.042).
- [dSD17] Leonardo dos Santos Pinheiro and Mark Dras. “Stock Market Prediction with Deep Learning: A Character-Based Neural Language Model for Event-Based Trading”. In: *Proceedings of the Australasian Language Technology Association Workshop 2017*. 2017, pp. 6–15.
- [DTG00] Dominique Douguet, Etienne Thoreau, and Gérard Grassy. “A Genetic Algorithm for the Automated Generation of Small Organic Molecules: Drug Design Using an Evolutionary Algorithm”. In: *Journal of Computer-Aided Molecular Design* 14.5 (July 1, 2000), pp. 449–466. ISSN: 1573-4951. DOI: [10.1023/A:1008108423895](https://doi.org/10.1023/A:1008108423895).
- [Duv+15] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. “Convolutional Networks on Graphs for Learning Molecular Fingerprints”. In: *Advances in Neural Information Processing Systems* 28 (2015), pp. 2224–2232. URL: <https://proceedings.neurips.cc/paper/2015/hash/f9be311e65d81a9ad8150a60844bb94c-Abstract.html> (visited on 11/16/2020).
- [Eis18] Dr Alexis Eisenhofer. *Definition: Center for Research in Security Prices (CRSP)*. <https://www.gabler-banklexikon.de/definition/center-research-security-prices-crsp-56614>. Oct. 30, 2018. URL: <https://www.gabler-banklexikon.de/definition/center-research-security-prices-crsp-56614/version-341290> (visited on 08/28/2023).
- [EK19] Lars Elend and Oliver Kramer. “Self-Organizing Maps with Convolutional Layers”. In: *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization - Proceedings of the 13th International Workshop, WSOM+ 2019, Barcelona, Spain, June 26-28, 2019*. Ed. by Alfredo Vellido, Karina Gibert, Cecilio Angulo, and José David Martín-Guerrero. Vol. 976. Advances in Intelligent Systems and Computing. Springer, 2019, pp. 23–32. DOI: [10.1007/978-3-030-19642-4_3](https://doi.org/10.1007/978-3-030-19642-4_3).
- [Ele+20] Lars Elend, Sebastian A. Tideman, Kerstin Lopatta, and Oliver Kramer. “Earnings Prediction with Deep Learning”. In: *KI 2020: Advances in Artificial Intelligence - 43rd German Conference on AI, Bamberg, Germany, September 21-25, 2020, Proceedings*. Ed. by Ute Schmid, Franziska Klügl, and Diedrich Wolter. Vol. 12325. Lecture Notes in

- Computer Science. Springer, 2020, pp. 267–274. DOI: [10.1007/978-3-030-58285-2_22](https://doi.org/10.1007/978-3-030-58285-2_22).
- [Ele+22] Lars Elend, Luise Jacobsen, Tim Cofala, Jonas Prellberg, Thomas Teusch, Oliver Kramer, and Ilia A. Solov'yov. “Design of SARS-CoV-2 Main Protease Inhibitors Using Artificial Intelligence and Molecular Dynamic Simulations”. In: *Molecules* 27.13 (13 Jan. 2022), p. 4020. ISSN: 1420-3049. DOI: [10.3390/molecules27134020](https://doi.org/10.3390/molecules27134020).
- [ERS08] Peter Ertl, Silvio Roggo, and Ansgar Schuffenhauer. “Natural Product-likeness Score and Its Application for Prioritization of Compound Libraries”. In: *Journal of Chemical Information and Modeling* 48.1 (Jan. 1, 2008), pp. 68–74. ISSN: 1549-9596. DOI: [10.1021/ci700286x](https://doi.org/10.1021/ci700286x).
- [ES09] Peter Ertl and Ansgar Schuffenhauer. “Estimation of Synthetic Accessibility Score of Drug-like Molecules Based on Molecular Complexity and Fragment Contributions”. In: *Journal of Cheminformatics* 1.1 (June 10, 2009), p. 8. ISSN: 1758-2946. DOI: [10.1186/1758-2946-1-8](https://doi.org/10.1186/1758-2946-1-8).
- [FAM66] Lawrence J Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, 1966.
- [FF93] Eugene F. Fama and Kenneth R. French. “Common Risk Factors in the Returns on Stocks and Bonds”. In: *Journal of Financial Economics* 33.1 (Feb. 1, 1993), pp. 3–56. ISSN: 0304-405X. DOI: [10.1016/0304-405X\(93\)90023-5](https://doi.org/10.1016/0304-405X(93)90023-5).
- [Fog94] David B. Fogel. “An Introduction to Simulated Evolutionary Optimization”. In: *IEEE transactions on neural networks* 5.1 (1994), pp. 3–14.
- [For+20] Florent Forest, Mustapha Lebbah, Hanane Azzag, and Jérôme Lacaille. *A Survey and Implementation of Performance Metrics for Self-Organized Maps*. Nov. 11, 2020. arXiv: [2011.05847](https://arxiv.org/abs/2011.05847) [cs]. URL: <http://arxiv.org/abs/2011.05847> (visited on 09/18/2023). preprint.
- [FPN18] Christos Ferles, Yannis Papanikolaou, and Kevin J. Naidoo. “Denoising Autoencoder Self-Organizing Map (DASOM)”. In: *Neural Networks* 105 (Sept. 2018), pp. 112–131. ISSN: 08936080. DOI: [10.1016/j.neunet.2018.04.016](https://doi.org/10.1016/j.neunet.2018.04.016).
- [FPoP14] Henri A. Favre, Warren H. Powell, and International Union of Pure and Applied Chemistry, eds. *Nomenclature of Organic Chemistry: IUPAC Recommendations and Preferred Names 2013*. Cambridge: Royal Soc. of Chemistry [u.a.], 2014. 1568 pp. ISBN: 978-0-85404-182-4.
- [Fre22] Kenneth R. French. *Detail for 5 Industry Portfolios*. 2022. URL: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_5_ind_port.html (visited on 08/23/2022).
- [Fro+13] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc’Aurelio Ranzato, and Tomas Mikolov. “DeViSE: A Deep Visual-Semantic Embedding Model”. In: (2013), p. 9.

- [Gai18] Thomas Gaillard. “Evaluation of AutoDock and AutoDock Vina on the CASF-2013 Benchmark”. In: *Journal of Chemical Information and Modeling* 58.8 (Aug. 27, 2018), pp. 1697–1706. ISSN: 1549-9596. DOI: [10.1021/acs.jcim.8b00312](https://doi.org/10.1021/acs.jcim.8b00312).
- [GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, June 14, 2011, pp. 315–323. URL: <https://proceedings.mlr.press/v15/glorot11a.html> (visited on 10/11/2023).
- [GC04] Holger Gohlke and David A. Case. “Converging free energy estimates: MM-PB(GB)SA studies on the protein–protein complex Ras–Raf”. In: *Journal of Computational Chemistry* 25.2 (2004), pp. 238–250. ISSN: 1096-987X. DOI: [10.1002/jcc.10379](https://doi.org/10.1002/jcc.10379).
- [Gil+17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. “Neural Message Passing for Quantum Chemistry”. June 12, 2017. arXiv: [1704.01212](https://arxiv.org/abs/1704.01212) [cs]. URL: <http://arxiv.org/abs/1704.01212> (visited on 11/16/2020).
- [GJ14] Alex Graves and Navdeep Jaitly. “Towards End-to-End Speech Recognition with Recurrent Neural Networks”. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML’14. Beijing, China: JMLR.org, 2014, pp. II-1764–II-1772.
- [Gra13] Alex Graves. “Generating Sequences With Recurrent Neural Networks”. Aug. 4, 2013. arXiv: [1308.0850](https://arxiv.org/abs/1308.0850) [cs]. URL: <http://arxiv.org/abs/1308.0850> (visited on 07/30/2019).
- [Hag+14] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, and Orlando De Jesus. *Neural Network Design*. Wrocław Amazon, 2014.
- [HD05] F. Hadzic and T.S. Dillon. “CSOM: Self-Organizing Map for Continuous Data”. In: *INDIN ’05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005*. INDIN ’05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005. Aug. 2005, pp. 740–745. DOI: [10.1109/INDIN.2005.1560466](https://doi.org/10.1109/INDIN.2005.1560466).
- [He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (visited on 02/13/2019).
- [Her90] Michael Herdy. “Application of the Evolutionsstrategie to Discrete Optimization Problems”. In: *Parallel Problem Solving from Nature*. Ed. by Hans-Paul Schwefel and Reinhard Männer. Vol. 496. Berlin/Heidelberg: Springer-Verlag, 1990, pp. 187–192. ISBN: 978-3-540-54148-6. DOI: [10.1007/BFb0029751](https://doi.org/10.1007/BFb0029751).

- [Hew+20] Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. “Temporal Convolutional Neural (TCN) Network for an Effective Weather Forecasting Using Time-Series Data from the Local Weather Station”. In: *Soft Computing* 24.21 (Nov. 1, 2020), pp. 16453–16482. ISSN: 1433-7479. DOI: [10.1007/s00500-020-04954-0](https://doi.org/10.1007/s00500-020-04954-0).
- [HM23] Petr Hajek and Michal Munk. “Speech Emotion Recognition and Text Sentiment Analysis for Financial Distress Prediction”. In: *Neural Computing and Applications* 35.29 (Oct. 1, 2023), pp. 21463–21477. ISSN: 1433-3058. DOI: [10.1007/s00521-023-08470-8](https://doi.org/10.1007/s00521-023-08470-8).
- [Hol75] John H Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. U Michigan Press, 1975.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1, 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [HSS18] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-Excitation Networks”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 7132–7141. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html (visited on 02/13/2019).
- [HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural Networks* 2.5 (Jan. 1, 1989), pp. 359–366. ISSN: 0893-6080. DOI: [10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).
- [HT17] Pavel Hamet and Johanne Tremblay. “Artificial Intelligence in Medicine”. In: *Metabolism. Insights Into the Future of Medicine: Technologies, Concepts, and Integration* 69 (Apr. 1, 2017), S36–S40. ISSN: 0026-0495. DOI: [10.1016/j.metabol.2017.01.011](https://doi.org/10.1016/j.metabol.2017.01.011).
- [IUP] IUPAC (The International Union of Pure and Applied Chemistry). *IUPAC - Hydrogen Bond (H02899)*. DOI: [10.1351/goldbook.H02899](https://doi.org/10.1351/goldbook.H02899).
- [Jin+20] Zhenming Jin, Xiaoyu Du, Yechun Xu, Yongqiang Deng, Meiqin Liu, Yao Zhao, Bing Zhang, Xiaofeng Li, Leike Zhang, Chao Peng, Yinkai Duan, Jing Yu, Lin Wang, Kailin Yang, Fengjiang Liu, Rendi Jiang, Xinglou Yang, Tian You, Xiaoce Liu, Xiuna Yang, Fang Bai, Hong Liu, Xiang Liu, Luke W. Guddat, Wenqing Xu, Gengfu Xiao, Chengfeng Qin, Zhengli Shi, Hualiang Jiang, Zihe Rao, and Haitao Yang. “Structure of M pro from COVID-19 Virus and Discovery of Its Inhibitors”. In: *Nature* (Apr. 9, 2020), pp. 1–9. ISSN: 1476-4687. DOI: [10.1038/s41586-020-2223-y](https://doi.org/10.1038/s41586-020-2223-y).
- [KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980). arXiv: [1412.6980](https://arxiv.org/abs/1412.6980) [cs]. preprint.

- [KH05] Susan S. Kaplan and Charles B. Hicks. “Safety and Antiviral Activity of Lopinavir/Ritonavir-Based Therapy in Human Immunodeficiency Virus Type 1 (HIV-1) Infection”. In: *Journal of Antimicrobial Chemotherapy* 56.2 (Aug. 1, 2005), pp. 273–276. ISSN: 0305-7453. DOI: [10.1093/jac/dki209](https://doi.org/10.1093/jac/dki209).
- [Kim+16] Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, Jiyao Wang, Bo Yu, Jian Zhang, and Stephen H. Bryant. “PubChem Substance and Compound Databases”. In: *Nucleic Acids Research* 44.D1 (Jan. 4, 2016), pp. D1202–D1213. ISSN: 0305-1048. DOI: [10.1093/nar/gkv951](https://doi.org/10.1093/nar/gkv951).
- [Kim+21] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. “PubChem in 2021: New Data Content and Improved Web Interfaces”. In: *Nucleic Acids Research* 49.D1 (Jan. 8, 2021), pp. D1388–D1395. ISSN: 0305-1048. DOI: [10.1093/nar/gkaa971](https://doi.org/10.1093/nar/gkaa971).
- [Koh82] Teuvo Kohonen. “Self-Organized Formation of Topologically Correct Feature Maps”. In: *Biological Cybernetics* 43.1 (Jan. 1, 1982), pp. 59–69. ISSN: 1432-0770. DOI: [10.1007/BF00337288](https://doi.org/10.1007/BF00337288).
- [Koh95] Teuvo Kohonen. “The Basic SOM”. In: *Self-Organizing Maps*. Ed. by Teuvo Kohonen. Springer Series in Information Sciences. Berlin, Heidelberg: Springer, 1995, pp. 77–130. ISBN: 978-3-642-97610-0. DOI: [10.1007/978-3-642-97610-0_3](https://doi.org/10.1007/978-3-642-97610-0_3).
- [Kol+00] Peter A. Kollman, Irina Massova, Carolina Reyes, Bernd Kuhn, Shuanghong Huo, Lillian Chong, Matthew Lee, Taisung Lee, Yong Duan, Wei Wang, Oreola Donini, Piotr Cieplak, Jayshree Srinivasan, David A. Case, and Thomas E. Cheatham. “Calculating Structures and Free Energies of Complex Molecules: Combining Molecular Mechanics and Continuum Models”. In: *Accounts of Chemical Research* 33.12 (Dec. 1, 2000), pp. 889–897. ISSN: 0001-4842. DOI: [10.1021/ar000033j](https://doi.org/10.1021/ar000033j).
- [KON13] A. Kutics, C. O’Connell, and A. Nakagawa. “Segment-Based Image Classification Using Layered-SOM”. In: *2013 IEEE International Conference on Image Processing*. Sept. 2013, pp. 2430–2434. DOI: [10.1109/ICIP.2013.6738501](https://doi.org/10.1109/ICIP.2013.6738501).
- [Koz92] John R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Vol. 1. MIT press, 1992.
- [KP20] Matthew Klimek and Maxim Perelstein. “Neural Network-Based Approach to Phase Space Integration”. In: *SciPost Physics* 9.4 (Oct. 19, 2020), p. 053. ISSN: 2542-4653. DOI: [10.21468/SciPostPhys.9.4.053](https://doi.org/10.21468/SciPostPhys.9.4.053).
- [Kra+18] Thorsten Krause, Niels Röckendorf, Nail El-Sourani, Katrin Ramaker, Maik Henkel, Sascha Hauke, Markus Borschbach, and Andreas Frey. “Breeding Cell Penetrating Peptides: Optimization of Cellular Uptake by a Function-Driven Evolutionary Process”. In: *Bioconjugate Chemistry* 29.12 (Dec. 19, 2018), pp. 4020–4029. ISSN: 1043-1802. DOI: [10.1021/acs.bioconjchem.8b00583](https://doi.org/10.1021/acs.bioconjchem.8b00583).

- [Kra03] Kramer, Oliver. “Restriktionsbehandlung bei Evolutionsstrategien mit Geschlechtern”. Diplomarbeit. Dortmund: Universität Dortmund, 2003.
- [Kre+20] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. “Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation”. In: *Machine Learning: Science and Technology* 1.4 (Oct. 2020), p. 045024. ISSN: 2632-2153. DOI: [10.1088/2632-2153/aba947](https://doi.org/10.1088/2632-2153/aba947).
- [Kru64] J. B. Kruskal. “Nonmetric Multidimensional Scaling: A Numerical Method”. In: *Psychometrika* 29.2 (June 1964), pp. 115–129. ISSN: 1860-0980. DOI: [10.1007/BF02289694](https://doi.org/10.1007/BF02289694).
- [KS06] O. Kramer and H.-P. Schwefel. “On Three New Approaches to Handle Constraints within Evolution Strategies”. In: *Natural Computing* 5.4 (Nov. 1, 2006), pp. 363–385. ISSN: 1572-9796. DOI: [10.1007/s11047-006-0001-x](https://doi.org/10.1007/s11047-006-0001-x).
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012. URL: <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html> (visited on 10/17/2023).
- [KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. ISSN: 00010782. DOI: [10.1145/3065386](https://doi.org/10.1145/3065386).
- [KTF20] Dimitris Kastaniotis, Dimitrios Tsourounis, and Spiros Fotopoulos. “Lip Reading Modeling with Temporal Convolutional Networks for Medical Support Applications”. In: *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI). Oct. 2020, pp. 366–371. DOI: [10.1109/CISP-BMEI51763.2020.9263634](https://doi.org/10.1109/CISP-BMEI51763.2020.9263634).
- [Lam+06] Eric-Wubbo Lameijer, Joost N. Kok, Thomas Bäck, and Ad P. IJzerman. “The Molecule Evaluator. An Interactive Evolutionary Algorithm for the Design of Drug-Like Molecules”. In: *Journal of Chemical Information and Modeling* 46.2 (Mar. 1, 2006), pp. 545–552. ISSN: 1549-9596. DOI: [10.1021/ci050369d](https://doi.org/10.1021/ci050369d).
- [LeC+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (Dec. 1, 1989), pp. 541–551. ISSN: 0899-7667. DOI: [10.1162/neco.1989.1.4.541](https://doi.org/10.1162/neco.1989.1.4.541).
- [Lip+97] Christopher A. Lipinski, Franco Lombardo, Beryl W. Dominy, and Paul J. Feeney. “Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings”. In: *Advanced Drug Delivery Reviews*. In Vitro Models for Selection of Development Candidates 23.1 (Jan. 15, 1997), pp. 3–25. ISSN: 0169-409X. DOI: [10.1016/S0169-409X\(96\)00423-1](https://doi.org/10.1016/S0169-409X(96)00423-1).

- [Luu+23] Sohvi Luukkonen, Helle W. van den Maagdenberg, Michael T. M. Emmerich, and Gerard J. P. van Westen. “Artificial Intelligence in Multi-Objective Drug Design”. In: *Current Opinion in Structural Biology* 79 (Apr. 1, 2023), p. 102537. ISSN: 0959-440X. DOI: [10.1016/j.sbi.2023.102537](https://doi.org/10.1016/j.sbi.2023.102537).
- [LWW22] Qiaoming Liu, Jun Wan, and Guohua Wang. “A Survey on Computational Methods in Discovering Protein Inhibitors of SARS-CoV-2”. In: *Briefings in Bioinformatics* 23.1 (Jan. 1, 2022), bbab416. ISSN: 1477-4054. DOI: [10.1093/bib/bbab416](https://doi.org/10.1093/bib/bbab416).
- [McC00] Charles E. McCulloch. “Generalized Linear Models”. In: *Journal of the American Statistical Association* 95.452 (Dec. 1, 2000), pp. 1320–1324. ISSN: 0162-1459. DOI: [10.1080/01621459.2000.10474340](https://doi.org/10.1080/01621459.2000.10474340).
- [MDJ21] Hylemariam Mihiretie Mengist, Tebelay Dilnessa, and Tengchuan Jin. “Structural Basis of Potential Inhibitors Targeting SARS-CoV-2 Main Protease”. In: *Frontiers in Chemistry* 9 (2021). ISSN: 2296-2646. URL: <https://www.frontiersin.org/article/10.3389/fchem.2021.622898> (visited on 03/26/2022).
- [MF00] D. MacDonald and C. Fyfe. “The Kernel Self-Organising Map”. In: *KES’2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*. KES’2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516). Vol. 1. Aug. 2000, 317–320 vol.1. DOI: [10.1109/KES.2000.885820](https://doi.org/10.1109/KES.2000.885820).
- [Mic+11] Naveen Michaud-Agrawal, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein. “MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations”. In: *Journal of Computational Chemistry* 32.10 (2011), pp. 2319–2327. ISSN: 1096-987X. DOI: [10.1002/jcc.21787](https://doi.org/10.1002/jcc.21787).
- [Mii90] Risto Miikkulainen. “Script Recognition with Hierarchical Feature Maps”. In: *Connection Science* 2.1-2 (Jan. 1, 1990), pp. 83–101. ISSN: 0954-0091. DOI: [10.1080/09540099008915664](https://doi.org/10.1080/09540099008915664).
- [Mit97] Tom M. Mitchell. “Machine Learning”. In: (1997).
- [MJK15] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. *Introduction to Time Series Analysis and Forecasting*. John Wiley & Sons, 2015.
- [MK17] Almuth Meier and Oliver Kramer. “An Experimental Study of Dimensionality Reduction Methods”. In: *KI 2017: Advances in Artificial Intelligence* (Cham). Ed. by Gabriele Kern-Isberner, Johannes Fürnkranz, and Matthias Thimm. Springer International Publishing, 2017, pp. 178–192. ISBN: 978-3-319-67190-1. DOI: [10.1007/978-3-319-67190-1_14](https://doi.org/10.1007/978-3-319-67190-1_14).
- [Mor+09] Garrett M. Morris, Ruth Huey, William Lindstrom, Michel F. Sanner, Richard K. Belew, David S. Goodsell, and Arthur J. Olson. “AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility”. In: *Journal of Computational Chemistry* 30.16 (2009), pp. 2785–2791. ISSN: 1096-987X. DOI: [10.1002/jcc.21256](https://doi.org/10.1002/jcc.21256).

- [NB13] Christos A. Nicolaou and Nathan Brown. “Multi-Objective Optimization Methods in Drug Design”. In: *Drug Discovery Today: Technologies* 10.3 (Sept. 1, 2013), e427–e435. ISSN: 1740-6749. DOI: [10.1016/j.ddtec.2013.02.001](https://doi.org/10.1016/j.ddtec.2013.02.001).
- [Nea+14] Victor-Emil Neagoe, Radu-Mihai Stoica, Alexandru-Ioan Ciurea, Lorenzo Bruzzone, and Francesca Bovolo. “Concurrent Self-Organizing Maps for Supervised/Unsupervised Change Detection in Remote Sensing Images”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.8 (Aug. 2014), pp. 3525–3533. ISSN: 2151-1535. DOI: [10.1109/JSTARS.2014.2330808](https://doi.org/10.1109/JSTARS.2014.2330808).
- [Nig+19] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alan Aspuru-Guzik. “Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space”. In: International Conference on Learning Representations. Sept. 25, 2019. URL: <https://openreview.net/forum?id=H1lmyRNFvr> (visited on 12/21/2021).
- [NR02] V.-E. Neagoe and A.-D. Ropot. “Concurrent Self-Organizing Maps for Pattern Classification”. In: *Proceedings First IEEE International Conference on Cognitive Informatics*. Proceedings First IEEE International Conference on Cognitive Informatics. Aug. 2002, pp. 304–312. DOI: [10.1109/COGINF.2002.1039311](https://doi.org/10.1109/COGINF.2002.1039311).
- [OAH14] Shogo Onojima, Yuya Arima, and Akira Hirose. “Millimeter-Wave Security Imaging Using Complex-Valued Self-Organizing Map for Visualization of Moving Targets”. In: *Neurocomputing*. Special Issue on the 2011 Sino-foreign-interchange Workshop on Intelligence Science and Intelligent Data Engineering (IScIDE 2011) 134 (June 25, 2014), pp. 247–253. ISSN: 0925-2312. DOI: [10.1016/j.neucom.2013.06.048](https://doi.org/10.1016/j.neucom.2013.06.048).
- [OBC00] Alexey Onufriev, Donald Bashford, and David A. Case. “Modification of the Generalized Born Model Suitable for Macromolecules”. In: *The Journal of Physical Chemistry B* 104.15 (Apr. 1, 2000), pp. 3712–3720. ISSN: 1520-6106. DOI: [10.1021/jp994072s](https://doi.org/10.1021/jp994072s).
- [OBC04] Alexey Onufriev, Donald Bashford, and David A. Case. “Exploring Protein Native States and Large-Scale Conformational Changes with a Modified Generalized Born Model”. In: *Proteins: Structure, Function, and Bioinformatics* 55.2 (2004), pp. 383–394. ISSN: 1097-0134. DOI: [10.1002/prot.20033](https://doi.org/10.1002/prot.20033).
- [OB0+11] Noel M. O’Boyle, Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. “Open Babel: An Open Chemical Toolbox”. In: *Journal of Cheminformatics* 3.1 (Oct. 7, 2011), p. 33. ISSN: 1758-2946. DOI: [10.1186/1758-2946-3-33](https://doi.org/10.1186/1758-2946-3-33).
- [OC19] Alexey V. Onufriev and David A. Case. “Generalized Born Implicit Solvent Models for Biomolecules”. In: *Annual review of biophysics* 48 (2019), pp. 275–296.
- [Oeh18] Stefan Oehmcke. “Deep Learning of Virtual Marine Sensors”. PhD thesis. University of Oldenburg, Germany, 2018. URL: <http://oops.uni-oldenburg.de/3613>.

- [Oor+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio”. Sept. 19, 2016. arXiv: [1609.03499](https://arxiv.org/abs/1609.03499) [cs]. URL: <http://arxiv.org/abs/1609.03499> (visited on 01/07/2020).
- [OR14a] Robert J. Ouellette and J. David Rawn. “1 - Structure and Bonding in Organic Compounds”. In: *Organic Chemistry*. Ed. by Robert J. Ouellette and J. David Rawn. Boston: Elsevier, Jan. 1, 2014, pp. 1–39. ISBN: 978-0-12-800780-8. DOI: [10.1016/B978-0-12-800780-8.00001-2](https://doi.org/10.1016/B978-0-12-800780-8.00001-2).
- [OR14b] Robert J. Ouellette and J. David Rawn. “28 - Synthetic Polymers”. In: *Organic Chemistry*. Ed. by Robert J. Ouellette and J. David Rawn. Boston: Elsevier, Jan. 1, 2014, pp. 993–1020. ISBN: 978-0-12-800780-8. DOI: [10.1016/B978-0-12-800780-8.00028-0](https://doi.org/10.1016/B978-0-12-800780-8.00028-0).
- [OR14c] Robert J. Ouellette and J. David Rawn. “4 - Alkanes and Cycloalkanes: Structures and Reactions”. In: *Organic Chemistry*. Ed. by Robert J. Ouellette and J. David Rawn. Boston: Elsevier, Jan. 1, 2014, pp. 111–161. ISBN: 978-0-12-800780-8. DOI: [10.1016/B978-0-12-800780-8.00004-8](https://doi.org/10.1016/B978-0-12-800780-8.00004-8).
- [OR14d] Robert J. Ouellette and J. David Rawn. “5 - Alkenes Structures and Properties”. In: *Organic Chemistry*. Ed. by Robert J. Ouellette and J. David Rawn. Boston: Elsevier, Jan. 1, 2014, pp. 163–193. ISBN: 978-0-12-800780-8. DOI: [10.1016/B978-0-12-800780-8.00005-X](https://doi.org/10.1016/B978-0-12-800780-8.00005-X).
- [OR14e] Robert J. Ouellette and J. David Rawn. “8 - Stereochemistry”. In: *Organic Chemistry*. Ed. by Robert J. Ouellette and J. David Rawn. Boston: Elsevier, Jan. 1, 2014, pp. 241–286. ISBN: 978-0-12-800780-8. DOI: [10.1016/B978-0-12-800780-8.00008-5](https://doi.org/10.1016/B978-0-12-800780-8.00008-5).
- [Pan+20] Pritam Kumar Panda, Murugan Natarajan Arul, Paritosh Patel, Suresh K. Verma, Wei Luo, Horst-Günter Rubahn, Yogendra Kumar Mishra, Mrutyunjay Suar, and Rajeev Ahuja. “Structure-Based Drug Designing and Immunoinformatics Approach for SARS-CoV-2”. In: *Science Advances* 6.28 (July 10, 2020), eabb8097. DOI: [10.1126/sciadv.abb8097](https://doi.org/10.1126/sciadv.abb8097).
- [Pan+21] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. “Deep Learning for Anomaly Detection: A Review”. In: *ACM Computing Surveys* 54.2 (Mar. 5, 2021), 38:1–38:38. ISSN: 0360-0300. DOI: [10.1145/3439950](https://doi.org/10.1145/3439950).
- [Phi+05] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. “Scalable Molecular Dynamics with NAMD”. In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1781–1802. ISSN: 1096-987X. DOI: [10.1002/jcc.20289](https://doi.org/10.1002/jcc.20289).
- [Phi+20] James C. Phillips, David J. Hardy, Julio D. C. Maia, John E. Stone, João V. Ribeiro, Rafael C. Bernardi, Ronak Buch, Giacomo Fiorin, Jérôme Hémin, Wei Jiang, Ryan McGreevy, Marcelo C. R. Melo, Brian K. Radak, Robert D. Skeel, Abhishek Singharoy, Yi Wang, Benoît Roux, Aleksei Aksimentiev, Zaida Luthey-Schulten, Laxmikant V.

- Kalé, Klaus Schulten, Christophe Chipot, and Emad Tajkhorshid. “Scalable Molecular Dynamics on CPU and GPU Architectures with NAMD”. In: *The Journal of Chemical Physics* 153.4 (July 28, 2020), p. 044130. ISSN: 0021-9606. DOI: [10.1063/5.0014475](https://doi.org/10.1063/5.0014475).
- [PHK01] Scott C.-H. Pegg, Jose J. Haresco, and Irwin D. Kuntz. “A Genetic Algorithm for Structure-Based de Novo Design”. In: *Journal of Computer-Aided Molecular Design* 15.10 (Oct. 1, 2001), pp. 911–933. ISSN: 1573-4951. DOI: [10.1023/A:1014389729000](https://doi.org/10.1023/A:1014389729000).
- [Pil+16] Thanigaimalai Pillaiyar, Manoj Manickam, Vigneshwaran Namasivayam, Yoshio Hayashi, and Sang-Hun Jung. “An Overview of Severe Acute Respiratory Syndrome–Coronavirus (SARS-CoV) 3CL Protease Inhibitors: Peptidomimetics and Small Molecule Chemotherapy”. In: *Journal of Medicinal Chemistry* 59.14 (July 28, 2016), pp. 6595–6628. ISSN: 0022-2623. DOI: [10.1021/acs.jmedchem.5b01461](https://doi.org/10.1021/acs.jmedchem.5b01461).
- [PK20] Jonas Prellberg and Oliver Kramer. *Acute Lymphoblastic Leukemia Classification from Microscopic Images Using Convolutional Neural Networks*. Apr. 1, 2020. arXiv: [1906.09020](https://arxiv.org/abs/1906.09020) [cs]. URL: <http://arxiv.org/abs/1906.09020> (visited on 10/17/2023). preprint.
- [Pol+20] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. “Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models”. In: *Frontiers in Pharmacology* 11 (2020), p. 1931. ISSN: 1663-9812. DOI: [10.3389/fphar.2020.565644](https://doi.org/10.3389/fphar.2020.565644).
- [Pub] PubChem. *Propane (3D Conformer)*. URL: <https://pubchem.ncbi.nlm.nih.gov/compound/6334> (visited on 11/23/2021).
- [PWP19] Charlotte Pelletier, Geoffrey I. Webb, and François Petitjean. “Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series”. In: *Remote Sensing* 11.5 (5 Jan. 2019), p. 523. DOI: [10.3390/rs11050523](https://doi.org/10.3390/rs11050523).
- [PZT17] L. Platon, F. Zehraoui, and F. Tahi. “Self-Organizing Maps with Supervised Layer”. In: *WSOM. 2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*. June 2017, pp. 1–8. DOI: [10.1109/WSOM.2017.8020022](https://doi.org/10.1109/WSOM.2017.8020022).
- [RBF12] Niels Röckendorf, Markus Borschbach, and Andreas Frey. “Molecular Evolution of Peptide Ligands with Custom-Tailored Characteristics for Targeting of Glycostructures”. In: *PLOS Computational Biology* 8.12 (Dec. 13, 2012), e1002800. ISSN: 1553-7358. DOI: [10.1371/journal.pcbi.1002800](https://doi.org/10.1371/journal.pcbi.1002800).
- [Rec73] Ingo Rechenberg. “Evolutionsstrategie”. In: *Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* (1973).
- [Rey+10] Jean-Louis Reymond, Ruud van Deursen, Lorenz C. Blum, and Lars Ruddigkeit. “Chemical Space as a Source for New Drugs”. In: *MedChemComm* 1.1 (July 1, 2010), pp. 30–38. ISSN: 2040-2511. DOI: [10.1039/C0MD00020E](https://doi.org/10.1039/C0MD00020E).

- [RF18] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. Apr. 8, 2018. arXiv: 1804.02767 [cs]. URL: <http://arxiv.org/abs/1804.02767> (visited on 02/25/2020).
- [RHW85] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Internal Representations by Error Propagation*. Institute for Cognitive Science, University of California, San Diego La . . . , 1985. URL: <https://apps.dtic.mil/sti/citations/ADA164453> (visited on 10/12/2023).
- [RL13] Sereina Riniker and Gregory A. Landrum. “Open-Source Platform to Benchmark Fingerprints for Ligand-Based Virtual Screening”. In: *Journal of Cheminformatics* 5.1 (1 Dec. 2013), pp. 1–17. ISSN: 1758-2946. DOI: [10.1186/1758-2946-5-26](https://doi.org/10.1186/1758-2946-5-26).
- [Roe98] Paul J. Roebber. “The Regime Dependence of Degree Day Forecast Technique, Skill, and Value”. In: *Weather and Forecasting* 13.3 (Sept. 1, 1998), pp. 783–794. ISSN: 0882-8156. DOI: [10.1175/1520-0434\(1998\)013<0783:TRDODD>2.0.CO;2](https://doi.org/10.1175/1520-0434(1998)013<0783:TRDODD>2.0.CO;2).
- [Sch17] F. Schleif. “Small Sets of Random Fourier Features by Kernelized Matrix LVQ”. In: *WSOM*. 2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM). June 2017, pp. 1–5. DOI: [10.1109/WSOM.2017.8020026](https://doi.org/10.1109/WSOM.2017.8020026).
- [Sch18] Gisbert Schneider. “Automating Drug Discovery”. In: *Nature Reviews Drug Discovery* 17.2 (2 Feb. 2018), pp. 97–113. ISSN: 1474-1784. DOI: [10.1038/nrd.2017.232](https://doi.org/10.1038/nrd.2017.232).
- [Sch87] Hans-Paul Schwefel. “Collective Phenomena in Evolutionary Systems”. In: (1987).
- [Sch93] Jürgen Schlitter. “Estimation of Absolute and Relative Entropies of Macromolecules Using the Covariance Matrix”. In: *Chemical Physics Letters* 215.6 (Dec. 17, 1993), pp. 617–621. ISSN: 0009-2614. DOI: [10.1016/0009-2614\(93\)89366-P](https://doi.org/10.1016/0009-2614(93)89366-P).
- [Seg+18] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. “Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks”. In: *ACS Central Science* 4.1 (Jan. 24, 2018), pp. 120–131. ISSN: 2374-7943. DOI: [10.1021/acscentsci.7b00512](https://doi.org/10.1021/acscentsci.7b00512).
- [Sha+20] Mousmee Sharma, Parteek Prasher, Meenu Mehta, Flavia C. Zacconi, Yogendra Singh, Deepak N. Kapoor, Harish Dureja, Dinesh M. Pardhi, Murtaza M. Tambuwala, Gaurav Gupta, Dinesh K. Chellappan, Kamal Dua, and Saurabh Satija. “Probing 3CL Protease: Rationally Designed Chemical Moieties for COVID-19”. In: *Drug Development Research* 81.8 (2020), pp. 911–918. ISSN: 1098-2299. DOI: [10.1002/ddr.21724](https://doi.org/10.1002/ddr.21724).
- [Sha+21] Jatin Sharma, Vijay Kumar Bhardwaj, Rahul Singh, Vidya Rajendran, Rituraj Purohit, and Sanjay Kumar. “An In-Silico Evaluation of Different Bioactive Molecules of Tea for Their Inhibition Potency against Non Structural Protein-15 of SARS-CoV-2”. In: *Food Chemistry* 346 (June 1, 2021), p. 128933. ISSN: 1873-7072. DOI: [10.1016/j.foodchem.2020.128933](https://doi.org/10.1016/j.foodchem.2020.128933). pmid: 33418408.

- [SI15] Teague Sterling and John J. Irwin. “ZINC 15 – Ligand Discovery for Everyone”. In: *Journal of Chemical Information and Modeling* 55.11 (Nov. 23, 2015), pp. 2324–2337. ISSN: 1549-9596. DOI: [10.1021/acs.jcim.5b00559](https://doi.org/10.1021/acs.jcim.5b00559).
- [Sil+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (7587 Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: [10.1038/nature16961](https://doi.org/10.1038/nature16961).
- [Sil+17a] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. Dec. 5, 2017. DOI: [10.48550/arXiv.1712.01815](https://doi.org/10.48550/arXiv.1712.01815). arXiv: [1712.01815 \[cs\]](https://arxiv.org/abs/1712.01815). preprint.
- [Sil+17b] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. “Mastering the Game of Go without Human Knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359.
- [Sin+21] Rahul Singh, Vijay Kumar Bhardwaj, Pralay Das, and Rituraj Purohit. “A Computational Approach for Rational Discovery of Inhibitors for Non-Structural Protein 1 of SARS-CoV-2”. In: *Computers in Biology and Medicine* 135 (Aug. 2021), p. 104555. ISSN: 1879-0534. DOI: [10.1016/j.compbiomed.2021.104555](https://doi.org/10.1016/j.compbiomed.2021.104555). pmid: [34144270](https://pubmed.ncbi.nlm.nih.gov/34144270/).
- [Smo13] Tomasz G. Smolinski. “Multi-Objective Evolutionary Algorithms”. In: *Encyclopedia of Computational Neuroscience*. Ed. by Dieter Jaeger and Ranu Jung. New York, NY: Springer, 2013, pp. 1–4. ISBN: 978-1-4614-7320-6. DOI: [10.1007/978-1-4614-7320-6_16-2](https://doi.org/10.1007/978-1-4614-7320-6_16-2).
- [Sri+99] Jayashree Srinivasan, Megan W. Trevathan, Paul Beroza, and David A. Case. “Application of a Pairwise Generalized Born Model to Proteins and Nucleic Acids: Inclusion of Salt Effects”. In: *Theoretical Chemistry Accounts* 101.6 (May 1, 1999), pp. 426–434. ISSN: 1432-2234. DOI: [10.1007/s002140050460](https://doi.org/10.1007/s002140050460).
- [Sti+90] W. Clark Still, Anna Tempczyk, Ronald C. Hawley, and Thomas Hendrickson. “Semianalytical Treatment of Solvation for Molecular Mechanics and Dynamics”. In: *Journal of the American Chemical Society* 112.16 (Aug. 1, 1990), pp. 6127–6129. ISSN: 0002-7863. DOI: [10.1021/ja00172a038](https://doi.org/10.1021/ja00172a038).
- [STL00] Robert F. Schmidt, Gerhard Thews, and Florian Lang, eds. *Physiologie Des Menschen*. Springer-Lehrbuch. Berlin, Heidelberg: Springer, 2000. ISBN: 978-3-662-09346-7. DOI: [10.1007/978-3-662-09346-7](https://doi.org/10.1007/978-3-662-09346-7).

- [Str+20] Birgit Strodel, Olujide Olubiyi, Maryam Olagunju, Monika Keutmann, and Jennifer Loschwitz. “High Throughput Virtual Screening to Discover Inhibitors of the Main Protease of the Coronavirus SARS-CoV-2”. In: (Apr. 9, 2020). DOI: [10.20944/preprints202004.0161.v1](https://doi.org/10.20944/preprints202004.0161.v1).
- [Sun+11] G. Sunilkumar, J. Thriveni, K. R. Venugopal, and L. M. Patnaik. “Cognition Based Self-Organizing Maps (CSOM) for Intrusion Detection in Wireless Networks”. In: *2011 Annual IEEE India Conference*. 2011 Annual IEEE India Conference. Dec. 2011, pp. 1–6. DOI: [10.1109/INDCON.2011.6139377](https://doi.org/10.1109/INDCON.2011.6139377).
- [SZ20] Mohamed Sakkari and Mourad Zaied. “A Convolutional Deep Self-Organizing Map Feature Extraction for Machine Learning”. In: *Multimedia Tools and Applications* 79.27 (July 1, 2020), pp. 19451–19470. ISSN: 1573-7721. DOI: [10.1007/s11042-020-08822-9](https://doi.org/10.1007/s11042-020-08822-9).
- [Tia+20] Chuan Tian, Koushik Kasavajhala, Kellon A. A. Belfon, Lauren Raguette, He Huang, Angela N. Miguez, John Bickel, Yuzhang Wang, Jorge Pincay, Qin Wu, and Carlos Simmerling. “ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution”. In: *Journal of Chemical Theory and Computation* 16.1 (Jan. 14, 2020), pp. 528–552. ISSN: 1549-9618. DOI: [10.1021/acs.jctc.9b00591](https://doi.org/10.1021/acs.jctc.9b00591).
- [TO10] Oleg Trott and Arthur J. Olson. “AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading”. In: *Journal of Computational Chemistry* 31.2 (2010), pp. 455–461. ISSN: 1096-987X. DOI: [10.1002/jcc.21334](https://doi.org/10.1002/jcc.21334).
- [Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 10/17/2023).
- [vdHor+12] Eelke van der Horst, Patricia Marqués-Gallego, Thea Mulder-Krieger, Jacobus van Veldhoven, Johannes Kruisselbrink, Alexander Aleman, Michael T. M. Emmerich, Johannes Brussee, Andreas Bender, and Adriaan P. IJzerman. “Multi-Objective Evolutionary Design of Adenosine Receptor Ligands”. In: *Journal of Chemical Information and Modeling* 52.7 (July 23, 2012), pp. 1713–1721. ISSN: 1549-9596. DOI: [10.1021/ci2005115](https://doi.org/10.1021/ci2005115).
- [Ven+15] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. “Sequence to Sequence - Video to Text”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4534–4542. URL: http://openaccess.thecvf.com/content_iccv_2015/html/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.html (visited on 07/30/2019).

- [Vil+17] T. Villmann, M. Biehl, A. Villmann, and S. Saralajew. “Fusion of Deep Learning Architectures, Multilayer Feedforward Networks and Learning Vector Quantizers for Deep Classification Learning”. In: *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*. 2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM). June 2017, pp. 1–8. DOI: [10.1109/WSOM.2017.8020009](https://doi.org/10.1109/WSOM.2017.8020009).
- [Vin+17] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhn-evets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekeremo, Jacob Repp, and Rodney Tsing. “StarCraft II: A New Challenge for Reinforcement Learning”. Aug. 16, 2017. arXiv: [1708.04782](https://arxiv.org/abs/1708.04782) [cs]. URL: <http://arxiv.org/abs/1708.04782> (visited on 02/06/2019).
- [Vin+19] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhn-evets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. “Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning”. In: *Nature* 575.7782 (7782 Nov. 2019), pp. 350–354. ISSN: 1476-4687. DOI: [10.1038/s41586-019-1724-z](https://doi.org/10.1038/s41586-019-1724-z).
- [Voe00] T. Voegtlin. “Context Quantization and Contextual Self-Organizing Maps”. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. Vol. 6. July 2000, 20–25 vol.6. DOI: [10.1109/IJCNN.2000.859367](https://doi.org/10.1109/IJCNN.2000.859367).
- [Wag+16] Travis T. Wager, Xinjun Hou, Patrick R. Verhoest, and Anabella Villalobos. “Central Nervous System Multiparameter Optimization Desirability: Application in Drug Discovery”. In: *ACS Chemical Neuroscience* 7.6 (Mar. 18, 2016), pp. 767–775. DOI: [10.1021/acchemneuro.6b00029](https://doi.org/10.1021/acchemneuro.6b00029).
- [Wan+04] Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. “Development and Testing of a General Amber Force Field”. In: *Journal of Computational Chemistry* 25.9 (2004), pp. 1157–1174. ISSN: 1096-987X. DOI: [10.1002/jcc.20035](https://doi.org/10.1002/jcc.20035).

- [Wan+17] Min Wang, Wengang Zhou, Qi Tian, Junfu Pu, and Houqiang Li. “Deep Supervised Quantization by Self-Organizing Map”. In: *Proceedings of the 25th ACM International Conference on Multimedia* (Mountain View, California, USA). MM ’17. New York, NY, USA: ACM, 2017, pp. 1707–1715. ISBN: 978-1-4503-4906-2. DOI: [10.1145/3123266.3123415](https://doi.org/10.1145/3123266.3123415).
- [Wei88] David Weininger. “SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules”. In: *Journal of Chemical Information and Computer Sciences* 28.1 (Feb. 1, 1988), pp. 31–36. ISSN: 0095-2338. DOI: [10.1021/ci00057a005](https://doi.org/10.1021/ci00057a005).
- [Wer74] Paul J. Werbos. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences” PhD Diss., Harvard University. Werbos, Paul J. 1988”. In: *Generalization of back propagation with application to a recurrent gas market method,* *Neural Networks* 1.4 (1974), pp. 339–356.
- [Wer90] Paul J. Werbos. “Backpropagation through Time: What It Does and How to Do It”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: [10.1109/5.58337](https://doi.org/10.1109/5.58337).
- [Wis+18] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. “DrugBank 5.0: A Major Update to the DrugBank Database for 2018”. In: *Nucleic Acids Research* 46.D1 (Jan. 4, 2018), pp. D1074–D1082. ISSN: 0305-1048. DOI: [10.1093/nar/gkx1037](https://doi.org/10.1093/nar/gkx1037).
- [Wit+17] Peter Wittek, Shi Chao Gao, Ik Soo Lim, and Li Zhao. “Somoclu: An Efficient Parallel Library for Self-Organizing Maps”. In: *Journal of Statistical Software* 78.9 (2017). ISSN: 1548-7660. DOI: [10.18637/jss.v078.i09](https://doi.org/10.18637/jss.v078.i09). arXiv: [1305.1422 \[cs\]](https://arxiv.org/abs/1305.1422).
- [WOK17] Wei Lee Woon, Stefan Oehmcke, and Oliver Kramer. “Spatio-Temporal Wind Power Prediction Using Recurrent Neural Networks”. In: *Neural Information Processing*. Ed. by Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy. Lecture Notes in Computer Science. Springer International Publishing, 2017, pp. 556–563. ISBN: 978-3-319-70139-4. DOI: [10.1007/978-3-319-70139-4_56](https://doi.org/10.1007/978-3-319-70139-4_56).
- [Wor20] Nils Steffen Worzyk. “Adversarials⁻¹: Detecting Adversarial Inputs with Internal Attacks”. PhD thesis. Universität Oldenburg, May 13, 2020. URL: <https://oops.uni-oldenburg.de/4616> (visited on 10/17/2023).
- [Yan+19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html> (visited on 12/21/2021).
- [YM00] Robert Yaffee and Monnie McGee. *Time Series Analysis and Forecasting with Applications of SAS and SPSS*. San Diego: Academic Press, Inc, 2000.

- [YPL20] Yaxia Yuan, Jianfeng Pei, and Luhua Lai. “LigBuilder V3: A Multi-Target de Novo Drug Design Approach”. In: *Frontiers in Chemistry* 8 (2020). ISSN: 2296-2646. DOI: [10.3389/fchem.2020.00142](https://doi.org/10.3389/fchem.2020.00142).
- [Zho+11] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. “Multiobjective Evolutionary Algorithms: A Survey of the State of the Art”. In: *Swarm and Evolutionary Computation* 1.1 (Mar. 1, 2011), pp. 32–49. ISSN: 2210-6502. DOI: [10.1016/j.swevo.2011.03.001](https://doi.org/10.1016/j.swevo.2011.03.001).
- [ZT98] Eckart Zitzler and Lothar Thiele. “Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Case Study”. In: *Parallel Problem Solving from Nature — PPSN V*. Ed. by Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1998, pp. 292–301. ISBN: 978-3-540-49672-4. DOI: [10.1007/BFb0056872](https://doi.org/10.1007/BFb0056872).

Index of Parameters

Vectors are printed in bold type. Matrices and tensors are written as capital roman letters in bold type. Sets are written as capital letters in calligraphic font. The parameters are sorted alphabetically, first Greek and then Roman. All pages are referenced on which a variable is used.

Greek letters

- δ_{\max} The value δ_{\max} specifies the maximum difference between the length of the new string r' and the length of the old string r . 67, 68, 78, 100, 137, 138
- α learning rate of a ANN or a SOM 14, 83, 84, 90,
- α_i effective Born radius of atom i 69,
- β window size 18, 19, 23,
- γ surface tension 69,
- ϵ_0 vacuum permittivity 69,
- ϵ_s dielectric constant of a solvent 69,
- η index of the BMU of a SOM 83, 88,
- κ Debye parameter 69,
- κ^{-1} Debye length 69,
- λ Number of individuals in the child generation of an evolution strategies (ES) 45–48, 64, 67, 68,
- μ Number of individuals in the parent generation of an ES 45–48, 64, 67, 68,
- μ_G mean of a Gaussian distribution 12,
- π the number $\pi \approx 3.14159$ 12,
- ρ Number of parents per child in an ES 45, 46,
- σ covariance matrix 70,
- σ Mutation strength in an ES 46, 47,
- σ range / radius of SOM 83, 84, 90,
- σ_G standard deviation of a Gaussian distribution 12,
- τ prediction horizon 19,
- ϕ a binary relation between \mathcal{X} and \mathcal{Y} . $\phi \subseteq \mathcal{X} \times \mathcal{Y}$. It contains ordered pairs of samples and labels (\mathbf{x}, \mathbf{y}) . 8, 18, 19, 137
- φ activation function of an ANN 11, 12, 14,

Roman letters

- A solvent-accessible surface area 69,
- \mathbf{a} atom position 76, 77,
- a atom 40,

- b bias of an artificial neuron) 11,
- C number of classes (of a classification problem) 8, 85, 88, 89, 138
- \mathcal{C} set of classes (of a classification problem), $\mathcal{C} = \{1, \dots, C\}$ 88, 89, 138
- $c_{i,j}$ class counter of neuron i for class j 88, 89,
- \bar{c} image color channels (e.g., $\bar{c} = 3$ for RGB images) 8, 86, 138
- D** distance matrix) 87, 93,
- d dilation factor of a TCN 20,
- r' Specifies the length of the new string generated by LM to replace the old string of length r . The value r' is randomly drawn from the interval $[\max\{1, r - \delta_{\max}\}, r + \delta_{\max}]$. 66–68, 137, 138
- e Euler's number $e \approx 2.71828$ 12, 83,
- e_c elementary charge 69, 70,
- \hat{e} result of the error function of an ANN 14,
- E_{DeViSE} semantic error based on DeViSE 93, 94,
- E_{KS} error Kruskal Shepard 87, 93,
- H_{SOM} cross entropy of the SOM 88, 91, 92, 100
- E_{KSN} original normalized error Kruskal Shepard 87, 91–93,
- \hat{E}_{KSN} normalized error Kruskal Shepard 87, 91–93, 100
- E_{MM} non-bonding molecular mechanics energies 69,
- F_{ij} dielectric term 69,
- f multi purpose function 8, 46,
- G multi purpose graph
- G^0 binding free energies 68,
- G_{C} binding free energies of the ligand-receptor complex 68, 69,
- G_{L} binding free energies of the ligand 68, 69,
- G_{np} non-polar solvation free energies 69,
- G_{p} polar solvation free energies 69,
- G_{R} binding free energies of the receptor 68, 69,
- h neighborhood function of SOM 83, 84,
- \bar{h} image height in pixel 8, 86,
- \hbar reduced Planck's constant 70,
- I** identity matrix: a square matrix with ones on the main diagonal and zeros elsewhere 70,
- I ion concentration 69,
- i multi purpose variable (counter, index etc.)
- j multi purpose variable (counter, index etc.)
- k multi purpose variable (counter, index etc.)
- ℓ multi purpose variable (counter, index etc.)
- k kernel size of a TCN 20,
- k_{B} Boltzmann constant 69, 70,
- k_e Coulomb constant, $k_e = 8.988 \times 10^9 \text{ N m}^2/\text{C}^2$ 69, 138
- L length of data considered by a TCN 20,

L'	given data length that should be covered by a TCN 20,
\mathcal{M}	molecule space / set of all molecules 56,
m	dimension of a SOM sample \mathbf{x} and weight vector \mathbf{w} 82, 86,
\hat{m}	dimension of a ConvSOM sample \mathbf{x} 86,
N	number of samples (and labels) 8, 82, 87, 88, 91, 92,
n_i	neuron i (of ANN or SOM) 11, 82–84, 88, 89, 92, 139
N_A	Avogadro number 69,
N_a	number of atoms in a ligand 76,
\mathcal{P}'	the next generation of individuals in an ES 46–48,
\mathcal{P}	a population of individuals in an ES 45, 46, 48,
\mathbf{p}_i	position of neuron n_i on the SOM 82–84, 89,
p_d	propability to delete a symbol during the mutation (see Section 5.3.2 54, 55, 57,
p_i	propability to insert a symbol during the mutation (see Section 5.3.2 54, 55, 57,
p_r	propability to replace a symbol during the mutation (see Section 5.3.2 54, 55, 57,
q_i	electrostatic charge of particle i 69,
r	The length $r \in [1, r_{\max}]$ of a substring which should be replaced by the LM model. 66–68, 137–139
r_{ij}	distance between particles i and j 69,
r_{\max}	The maximum value of r . 67, 68, 78, 100, 139
S	entropy 69, 70,
s_i	number of clusters of class i 89, 91, 92,
s	stack size of a TCN 20,
T	temperature 69, 70,
t	time or time step 18–20, 76,
u	prediction data 18, 19,
v	number of layers of a TCN 20,
$w_{i,j}$	a single weight j of neuron n_i 11, 14,
\mathbf{w}_i	weight vector of neuron n_i 82–84, 92, 93, 139
\bar{w}	image width in pixel 8, 86,
\mathcal{X}'	a batch of samples, $\mathcal{X}' \subseteq \mathcal{X}$ 84, 139
X	placeholder for an electronegative atom 36,
$\bar{\mathcal{X}}$	sample space 8,
\mathbf{X}	tensor of samples 8–10, 139
\mathcal{X}	set of samples \mathbf{x} 82, 84, 92, 137, 139
$\hat{\mathbf{x}}$	sample from a convolutional layer of the ConvSOM 86, 87,
x	multi purpose input (e.g., input of an artificial neuron) 11,
\mathbf{x}	sample 8, 9, 18, 19, 82–84, 86–88, 92, 93, 137, 139, 145
\mathbf{X}_{test}	A tensor of samples that are used for testing. It contains a part of the tensor of samples: $\forall i \in \mathbb{N} \exists j \in \mathbb{N} : \mathbf{X}_{\text{test}i} = \mathbf{X}_j$. 9, 139
$\mathbf{X}_{\text{train}}$	A tensor of samples that are used for training. It contains a part of the tensor of samples: $\forall i \in \mathbb{N} \exists j \in \mathbb{N} : \mathbf{X}_{\text{train}i} = \mathbf{X}_j$. 9, 139

- \mathbf{X}_{val} A tensor of samples that are used for validation. It contains a part of the tensor of samples:
 $\forall i \in \mathbb{N} \exists j \in \mathbb{N} : \mathbf{X}_{\text{val}i} = \mathbf{X}_j$. 9, 139
- Y placeholder for an electronegative atom 36,
- $\bar{\mathcal{Y}}$ label space 8,
- \mathbf{Y} tensor of labels 8–10, 140
- \mathcal{Y} set of labels \mathbf{y} 137
- y label 8, 14, 18, 19, 87, 88, 137, 140, 145
- y multi purpose output (e.g., output of an artificial neuron) 11,
- $\hat{\mathbf{y}}$ Result of an ANN that tries to approximate the real label \mathbf{y} 14,
- \mathbf{Y}_{test} A tensor of labels that are used for testing. It contains a part of the tensor of labels:
 $\forall i \in \mathbb{N} \exists j \in \mathbb{N} : \mathbf{Y}_{\text{test}i} = \mathbf{Y}_j$. 9, 140
- $\mathbf{Y}_{\text{train}}$ A tensor of labels that are used for training. It contains a part of the tensor of labels:
 $\forall i \in \mathbb{N} \exists j \in \mathbb{N} : \mathbf{Y}_{\text{train}i} = \mathbf{Y}_j$. 9, 140
- \mathbf{Y}_{val} A tensor of labels that are used for validation. It contains a part of the tensor of labels:
 $\forall i \in \mathbb{N} \exists j \in \mathbb{N} : \mathbf{Y}_{\text{val}i} = \mathbf{Y}_j$. 9, 140
- \hat{z}_i \hat{z}_i is the i th feature of COMPUTSTAT QUARTERLY (see Table B.1) 21, 140
- \bar{z}_i mean of z_i 21,
- z_i z_i is \hat{z}_i (the i th feature of COMPUTSTAT QUARTERLY) divided by the total assets atq 21, 140
- z'_i z'_i is z_i studentized 21, 140

Acronyms

If there is a page with the definition or description of the acronym, it is in bold type.

3CL ^{pro}	3C-like protease 32 ,
AE	autoencoders 10 ,
AI	artificial intelligence 7, 32, 50, 62–65, 70, 100
AMEX	American Stock Exchange 21 ,
ANN	artificial neural network 4, 7, 11, 13, 14, 17, 19, 21–23, 84, 108, 137–140
API	application programming interface 38, 145 ,
BA	binding affinity 51, 53, 55, 56, 58–60, 62, 71, 72, 78, 98, 99, 111, 112, 145
BMU	best matching unit 82, 83, 84, 88, 90, 100, 137
C	ligand-receptor complex 68–70 ,
CADD	computer-aided drug design 63, 64 ,
CGP	cartesian genetic programming
CI	computational intelligence III, V, 3, 4, 7, 97, 98
CIFAR-10	Canadian Institute For Advanced Research 90–93, 100 , <i>Glossary</i> : CIFAR-10
CNN	convolutional neural network 4, 20, 81, 82, 84, 85, 86, 90, 92, 94, 100, 101
COM	center of mass 74, 75, 77, 78, 99
ConvSOM	convolutional self-organizing map III–V, 4, 6, 81, 85, 86–94, 100, 101, 139
COVID-19	coronavirus disease 2019 5, 32, 49 ,
CRSP	Center for Research in Security Prices 21 ,
CSI	class scatter index 88, 89, 91–93, 100
DBN	deep belief network 10 ,
DBSCAN	density-based spatial clustering of applications with noise 9 ,
DeViSE	deep visual-semantic embedding model 93, 94, 100, 138
DNA	deoxyribonucleic acid 45 ,
DNN	deep neural network 3, 11, 12, 97, 98

EA	evolutionary algorithm 4, 5, 32, 45 , 48, 50, 56, 57, 63, 65, 78, 98–100, 111
EMGA	evolutionary molecule generation algorithm 5, 63 , 64, 65, 67–74, 77, 78, 98–100, 112
EP	evolutionary programming 45 ,
EPS	earnings per share 17 , 21, 25, 97
ES	evolution strategies 45 , 46, 53, 64, 137, 139
FFI	Fama-French industries 25 , 98, 107
FFI5	Fama-French industries with 5 groups 25 , 26, 27, 97
GA	genetic algorithm 45 , 56, 99, 111
GAN	generative adversarial network 10,
GB	generalized Born 69,
GBIS	generalized Born implicit solvent 70,
GNN	graph neural network 101
GP	genetic programming 45 ,
GPU	graphics processing unit 3, 11, 86,
HIV	human immunodeficiency virus 59,
HPC	high performance cluster 74, 99
i.i.d.	independent and identically distributed 10,
InChI	International Chemical Identifier 39 ,
IUPAC	International Union of Pure and Applied Chemistry 37 , 39,
L	ligand 68–70,
LM	language model 63, 64, 65 , 78, 93, 98–100, 138, 139
LSTM	long short-term memory 14, 17, 19 , 22–27, 64, 97, 98
M ^{pro}	main protease 32 , 49, 51, 56, 59–62, 70–72, 74, 75, 77, 78, 98–100
MAPE	mean absolute percentage error 22 ,
MCF	medical chemical filter 53,
MCO	minor class occurrence 88 ,
MD	molecular dynamics 63, 64, 68 , 69–71, 74, 77, 78, 99, 100
ML	machine learning 4, 7 , 9, 14,
MM/GBSA	molecular mechanics / generalized Born surface area 68,
MNIST	Modified National Institute of Standards and Technology 89–93, 100, <i>Glossary</i> : MNIST

MOEA	multi-objective evolutionary algorithm 56 ,
MOO	multi-objective optimization 4 , 49 , 50 , 56 ,
MOSES	Molecular Sets 56 , 67 , 98 , 111 , <i>Glossary</i> : MOSES
MPNN	message passing neural network 101
MSE	mean squared error 22 , 23 , 26 , 97
MUX	multiplexer 86 , 94 ,
NASDAQ	National Association of Securities Dealers Automated Quotation 21 ,
NG	neural gas 10 ,
NMCO	normalized minor class occurrence 88 , 91 , 92 , 100
NP	natural product-likeness 52 , 53 , 56 , 58 , 59 , 71 , 78 , 98 , 112
NSGA-II	non-dominated sorting genetic algorithm II 56 , 57 , 98 , 100
NYSE	New York Stock Exchange 21 ,
PAINS	pan assay interference compounds 53 ,
PDB	protein data bank 40 , 59 ,
PI	protease inhibitor 60 ,
PREMS	Pareto ranking evolutionary molecule search 49 , 54 , 56 , 57 – 62 , 98 , 99
QED	quantitative estimate of drug-likeness 52 , 53 , 56 , 58 – 60 , 62 , 71 , 78 , 98 , 99 , 112
R	receptor 68 – 70 ,
ReLU	rectified linear unit 12 , 20 , 85 , 90 ,
RMSD	root mean square displacement 75 – 78 , 99
RMSF	root means square fluctuations 75 , 77 , 99
RNA	ribonucleic acid 32 , 45 ,
RNN	recurrent neural network 13 , 19 ,
SA	synthetic accessibility 51 , 52 , 53 , 56 , 58 – 60 , 71 , 78 , 98 , 99 , 112
SARS-CoV-2	severe acute respiratory syndrome coronavirus-2 3 , 32 , 49 , 56 , 61 , 62 , 64 , 70 , 78 , 98 , 100
SD	standard deviation 91 , 92 ,
SELFIES	self-referencing embedded strings 38 , 39 , 47 , 53 , 54 , 56 , 57 , 59 , 62 , 65 , 98 , 99 , 111
SIC	standard industrial classification 25 ,
SMILES	simplified molecular-input line-entry system 37 , 38 , 39 , 47 , 50 , 56 , 63 – 68 , 71 , 98 , 99 , 111 , 112

SMS-EMOA	<i>S</i> metric selection evolutionary multiobjective optimisation algorithm 100
SOM	self-organizing map 4, 10, 81, 82 , 83–87, 89–94, 100, 101, 137–139
SS	skill score 22 , 97
tanh	hyperbolic tangent 12 , 23,
TCN	temporal convolutional network 14, 17, 20 , 22–27, 97, 138, 139
TF	toxicity filters 51, 53 , 56, 57, 59, 62, 78, 98, 112
vdW	van der Waals 70, 77,
VSEPR	valence shell electron pair repulsion 35,
WSEMS	weighted sum evolutionary molecule search 49, 54, 56 , 57–62, 98, 99
ZINC	is not commercial 64, 67, 78, 99, 144, <i>Glossary</i> : ZINC

Glossary

CIFAR-10 The CIFAR-10 database contains 32×32 color images of 10 different classes. The data set consists of 50000 samples for training and 10000 for testing. 90, 100

feature A **sample** can have multiple features, e.g., features of a car would be: color, brand, maximum speed etc. 81, 107, 145

Keras Keras is an open-source high-level deep learning library for Python. It can especially be used as interface for **TensorFlow**. 90,

label A label **y** corresponding to a **sample**, e.g., in object recognition the image is the sample while the object on the image is the label. 140

MNIST The MNIST database contains 28×28 grayscale images of handwritten digits. The data set consists of 60000 samples for training and 10000 for testing, which are divided into 10 classes. 89, 100

MOSES Molecular Sets (MOSES) is a benchmarking platform for molecular generation models (see [Pol+20]). 56, 98

pattern 1. A pattern can exist between different **samples**, that have similar **features**, e.g., multiple cars with the same color. 2. A pattern can exist from different features within a single sample, e.g., sub-elements of an object in object recognition, such as eyes in face recognition. 9, 82,

QuickVina 2 QuickVina 2 [Alh+15] is an improved version of AutoDock Vina and is used to estimate the **BA** between two molecules. 51, 53, 56, 60, 62, 98, 99

RDKit RDKit is a open-source software for cheminformatics and machine learning with a python **API** (<https://www.rdkit.org>). 38, 56,

sample A sample **x** is a single input element, e.g., a specific car out of a set of all cars. 81, 82, 107, 139, 145

TensorFlow TensorFlow is an open-source deep learning library for Python, JavaScript, C++, and Java. 86, 90, 145

ZINC ZINC is a large public access molecule database and tool set (see [SI15]). 64, 99

Index

Symbols

$(\mu + \lambda)$	48
(μ, λ)	47
3CL ^{pro}	32

A

ANN	11
aromatic rings	37
atom	32
atomic number	32
AutoDock	51
AutoDock Vina	51

B

BA	51
ball-and-stick model	41
BMU	82
Bohr model	33
bond	35
covalent	35
intermolecular	35
intramolecular	35
ionic	35
metallic	35
strong	35
weak	35

C

canonical SMILES	38
chair representation	40
chiral	43

CI	7
cis	43
cis-trans isomers	43
classification	8
CNN	84
comma selection	47
constitutional isomers	42
ConvSOM	85
covalent bond	35
COVID-19	32
cross entropy	88
CRSP	21
CSI	88

D

DeViSE	93
DNN	11
docking score	51
dropout	85

E

EA	45
electron	32
element	32
EMGA	63
enantiomers	43
endogenous parameters	46
EP	45
EPS	17
ES	45
exogenous parameters	46

- F**
- feed forward network 13
 - FFI 25
 - FFI5 25
 - fingerprints 39
 - Fischer projection 40
- G**
- GA 45
 - GP 45
 - graph 40
- H**
- Haworth projection 40
 - hydrogen bond 36
- I**
- InChI 39
 - intermolecular bond 35
 - intramolecular bond 35
 - ionic bond 35
 - isomer 42
 - IUPAC 37
 - IUPAC nomenclature of organic chemistry 37
- K**
- Kruskal Shepard error 87
- L**
- learning
 - reinforcement 10
 - supervised 8
 - unsupervised 9
 - Lewis structure 39
 - ligand 49
 - LM 65
 - LSTM 19
- M**
- MAPE 22
 - MCO 88
 - MD 68
 - metallic bond 35
 - ML 7
 - MOEA 56
 - molecule 37
 - 1D / String representation 37
 - 2D representation 39
 - 3D representation 40
 - M^{pro} 32
 - MSE 22
 - MUX 86
- N**
- natta projection 39
 - neutron 32
 - NMCO 88
 - NP 53
 - NSGA-II 56
- O**
- organic chemistry 32
 - overfitting 9
- P**
- padding 84
 - PDB 40
 - plus selection 48
 - pooling layer 85
 - PREMS 56
 - protease enzyme 31
 - protease inhibitor 31, 49
 - protein cleavage 31
 - proton 32
- Q**
- QED 52
 - QuickVina 2 51
- R**
- reachability 47

regression	8	van der Waals forces	35
reinforcement learning	10	W	
ReLU	12	weak bond	35
RMSF	77	weightage plot	89
RNN	19	winner neuron	82
S		WSEMS	56
SA	51	Z	
sample class plot	89	zero padding	84
SARS-CoV-2	32		
scalability	47		
SELFIES	38		
SIC	25		
skeletal formula	40		
SMILES			
canonical	38		
SMILES	37		
softmax function	85		
SOM	82		
space-filling model	41		
SS	22		
stereoisomers	42		
sticks model	41		
stride	84		
strong bond	35		
structural isomer	42		
supervised learning	8		
T			
tanh	12		
TCN	20		
TF	53		
time series prediction	18		
trans	43		
U			
unbiasedness	47		
unsupervised learning	9		
V			
valence electron	35		

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 7. Februar 2024

Lars Elend