Otto-von-Guericke Universität

Bachelor's Thesis

Deep Learning approach for Peptide Identification using big public databases

Author:

Daniel Micheel

August 17, 2018

Supervisors:

Prof. Dr. rer. nat. habil. Gunter Saake

Institute for Technical and Business Information Systems

M.Sc. Roman Zoun

Workgroup Databases & Software Engineering

Dipl.-Ing. Kay Schallert

Institute of Process Engineering

Magdeburg, Sachsen-Anhalt, Deutschland

2018

Micheel, Daniel:

Deep Learning approach for Peptide Identification using big public databases

Bachelor’s Thesis, Otto-von-Guericke Universität Magdeburg, 2018.

Abstract

Proteomics is the study of proteins by mass spectrometry and involves the identification via protein database searches. Unfortunately only known proteins can be identified through this method. De novo sequencing of proteins constitutes an alternative method, but previous imperative algorithms suffered from high computation load and low accuracy. All proteomics data is collected in publicly available databases such as the PRIDE Archive. Advances in deep learning have lead to learning-based solutions for recognition and prediction problems. For that reason, this thesis proposes a new system called DeepPSM that solves the task of de novo sequencing using a deep learning approach. DeepPSM continuously integrates new training data into the deep learning model, which is automatically retrieved from the PRIDE archive. Therefore, a sequence to sequence model, a convolutional sequence to sequence model and the complex DeepNovo model were implemented and automatically trained with real-world data. Preliminary tests already demonstrate that the best model can achieve a recall of up to 24.3%, outperforming previous imperative approaches. Extending the training data to the entire PRIDE archive should increase recall even more. In conclusion, the DeepPSM software is suitable for the task of peptide identification from mass spectrometry data.

Contents

Abstract **iii**

1 Introduction **1**
1.1 Motivation 1
1.2 Goal 3
1.3 Related Work 3

2 Background **5**
2.1 Proteomics 5
2.2 Identification of Proteins 6
2.3 PRIDE database 7
2.4 Machine Learning 8
2.4.1 Neural Networks 8
2.4.2 TensorFlow 14
2.4.3 Keras 14

3 Concept **15**
3.1 DeepPSM architecture 15
3.2 Data 16
3.2.1 Binning 17
3.3 Deep learning architectures 18
3.3.1 Sequence to Sequence RNN 18
3.3.2 Convolutional Sequence to Sequence RNN 20
3.3.3 DeepNovo 21

4 Implementation **23**
4.1 Hardware 23
4.2 PRIDE archive API 24
4.3 MGF Parsing 25
4.4 mzIdentML Parsing 26

4.5 Creation of training data sets 27
4.6 Deep learning models 27
4.7 Training 28
4.7.1 DeepNovo 28
4.8 Overview of implementation 29

5 Evaluation **30**
5.1 Data retrieval component 30
5.1.1 Setup 30
5.1.2 Evaluation concept 30
5.1.3 Results 31
5.1.4 Discussion 32
5.2 Deep learning models 33
5.2.1 Setup 33
5.2.2 Evaluation concept 33
5.2.3 Results 34
5.2.4 Discussion 37

6 Conclusion & Future Work **40**

Appendix **43**

List of abbreviations **46**

List of figures **47**

Bibliography **48**

CHAPTER 1

Introduction

1.1 Motivation

Proteomics is the study of proteins by mass spectrometry, which has many practical uses with industrial, medical and analytical applications. Enzymes and some hormones are proteins that are not only important for living organisms as they are carriers of biological functions, but are also used for industrial applications [1]. Enzymes make biological processes possible as they act as catalysts for chemical reactions. Biological systems are regulated by hormones. The four different blood types of humans are differentiated based on the proteins present in it. The research of proteins is also important for the development of medical treatments for diseases. Certain proteins have been identified as biomarkers of cancer [2] and the identification of host proteins that are exploited by HIV-1 is a topic of ongoing research [3]. The identification of the unknown or modified proteins in biological samples has become a wide field of practical research. Proteomics researchers are combining their efforts for the identification of proteins by utilizing the proteomics databases that contain and provide metadata, protein sequences and mass spectra by sharing their data publicly. This not only leads to the ability of better recognizing known proteins but also leads to the discovery of novel proteins. Not only can a discovered new protein give a better understanding of the biological functioning of an organism it can also give clues of the condition of the organism that formed the protein in its cells. Molecular diagnostics including the identification and recognition of proteins has been proposed as a new component to the healthcare value chain [4].

In the field of proteomics mass spectrometry technology has become more powerful and the size of proteomics data sets has increased with the introduction of high resolution mass spectrometers. Modern high performance mass spectrometers reach a mass resolution[1] of up to one million at Full Width at Half Maximum (FWHM)[2] and

[1] Measurement for the ability of separating two narrow mass spectral peaks.

[2] Width of a spectrum curve measured between those points on the x-axis, which are half the maximum amplitude.

the data size of public proteomics archives reaches hundreds of TBs. For the identification of proteins from the resulting mass spectra, three approaches are commonly used: protein database searches, mass spectral libraries and de novo peptide sequencing [5]. These approaches utilize search, optimization and matching algorithms, which are computationally intensive and time consuming, to identify the proteins or peptides in the sample. An example of a publicly available data repositories is the Proteomics Identifications (PRIDE) archive, which contains the research data of many proteomics projects. The individual projects in the archive contain identified proteins, peptides and their mass spectra.

Machine learning as a field of computer science has gained popularity. In recent years as the improving performance of computing hardware, an increase in available data and the maturity of the field made the widespread utilization of deep learning techniques feasible. Deep learning frameworks like Keras[3] and Tensorflow[4] have now reached maturity after years of collaborative development. Machine learning shows promising results in object recognition [6], image analysis and time series prediction [7] among other topics. These results have led to real-world applications as machine learning is utilized in new mobile phones with Google [8] and Apple [9] introducing mobile machine learning frameworks and engineers at Qualcomm creating dedicated mobile hardware [10] for the development of applications that utilize deep learning components. This allows developers to implement applications that utilize machine learning and reach a mainstream audience. This bachelor thesis proposes a new system, DeepPSM, for the identification of peptides using deep learning approaches that incorporate the PRIDE archive of the European Bioinformatics Institute as background knowledge. The DeepPSM system is able to download, process and prepare training data provided by the PRIDE archive for the deep learning component. The DeepPSM system is capable to apply different machine learning techniques and systems to the problem of amino acid identification in peptides. The fitting of the newly downloaded data to the machine learning component and the following testing is done autonomous.

In the following I will define the goal of this thesis. Afterwards the related work in deep-learning based peptide identification are introduced. In Chapter 2 the theoretical background of proteomics research and machine learning is explained, before giving an overview of the implementation in Chapter 4 and the concept in Chapter 3. Afterwards the evaluation concept and results are presented in Chapter 5 followed by the conclusion and proposals for future improvements of the DeepPSM in Chapter 6.

[3] *Keras*. Retrieved August 6, 2018, `https://github.com/keras-team/keras`

[4] *Tensorflow*. Retrieved August 6, 2018, `https://github.com/tensorflow/tensorflow`

1.2 Goal

The goal of this work is to explain the DeepPSM system and evaluate the prediction performance of different deep learning models for the task of peptide identification. The evaluation will introduce the necessary evaluation criteria, which will measure the performance of the data retrieval and the quality of the predictions provided by the different deep learning approaches that have been explored. I applied a sequence to sequence model, a convolutional sequence to sequence model and the complex DeepNovo model to the problem of de novo peptide sequencing. The results show that it is possible to integrate the PRIDE archive as background knowledge and create training, validation and testing data sets with minimal manual intervention.

1.3 Related Work

The solutions to the problem of protein identification can be divided into two approaches: bottom-up and top-down proteomics [11]. The top-down approach analyzes proteins intact while the bottom-up approach starts with a fragmentation of proteins into peptides. This makes the identification of individual peptides necessary. In the field of machine learning the problem of peptide identification has not been explored by many researchers. DeepNovo is a learning-based tool that addresses the problem of peptide identification from tandem mass spectrometry data using a deep learning approach. The current version of DeepNovo was introduced by Ngoc Hieu Tran et al. in 2017 [12]. DeepNovo combines multiple convolutional neural networks, recurrent LSTM networks and a dynamic programming algorithm to create a protein identification tool that shows promising results in comparison to conventional de novo peptide sequencing tools. DeepNovo is not only able to perform de novo peptide sequencing, but also offers a hybrid approach, which incorporates database searches.

The deep learning component of the DeepNovo system computes a probability distribution over 27 classes that are representing the proteinogenic amino acids, their modifications and start and stop indicators. This probability distribution is used to compute the probability of a sequence for a given mass spectrum. It utilizes a Convolutional Neural Network (CNN) for feature extraction and a Recurrent Neural Network (RNN) for sequence to sequence generation.

If the protein database search feature of DeepNovo is utilized the tool behaves similar to classic protein database search algorithms. The matching score is calculated by the equation that utilizes the probability result of the deep learning component. The de novo peptide sequencing component of DeepNovo performs a beam search that explores a fixed number of the best candidates resulting from the probability equation. The DeepNovo de novo model was measured by the author to be able to reach a recall

of 74.3% for amino acids and 61.7% for peptides. This recall already outperforms previously used imperative de novo sequencing algorithms. The recall is the percentage of retrieved relevant items. The peptide recall being the percentage of correct sequences of all decoded sequences and amino acid recall being the percentage of exact amino acids of all decoded amino acids. However the DeepNovo tool was only trained with a preselected and fixed-size training set from the Saccharomyces cerevisiae proteome [12]. While DeepNovo offers a deep learning approach to the de novo peptide sequencing problem the tool itself does not include a data retrieval and processing component for the available data in the public proteomics databases such as PRIDE. The evaluation in the paper presented in 2017 did not evaluate DeepNovo with a proteomics data set containing different species. The data utilized by the DeepPSM system automatically incorporates the data available in the PRIDE archive.

CHAPTER 2

Background

The following chapter introduces the necessary background knowledge for this thesis. Hence, proteomics will be introduced in the next section. Afterwards I will explain basic machine learning concepts and neural networks.

2.1 Proteomics

The introduction of high resolution mass spectrometry and developments in bioinformatics have led to the rise of large-scale proteomics research. Proteomics is the branch of molecular biology that focuses on the analysis, identification and quantification of proteins. The goal of proteomics research is to analyze the entire proteome of different species and identify the proteins in biological samples [5, 13, 14]. The proteome is the set of expressed proteins, the proteins that are present in an organism at any given time. In contrast, the genome encodes all proteins of a species, including those, which are not expressed. Therefore, proteomics can reveal information about the current state of an organism. A protein is a macromolecule that can perform a specific function in the body of an organism. Proteins are constructed inside cells from the 20 proteinogenic amino acids and can be represented as a sequence of letters, where each letter represents an amino acid. Deoxyribonucleic acid (DNA) is transcribed into Ribonucleic acid (RNA), which is then translated into the protein sequences. The transcription and translation can also be replicated theoretically, which allows to predict the accurate amino acid sequences that are stored in protein databases. Proteins perform various functions in an organism and their study is important for the understanding of biological processes, diseases and natural phenomena. Proteomics plays an important role in the development of cancer treatments and analyzed and studied proteins have many different industrial, medical and analytical uses. There are two different approaches to the problem of protein identification: the top-down approach and the bottom-up approach. In this work I selected the bottom-up approach as the basis of the continued research.

2.2 Identification of Proteins

For the bottom-up identification of proteins a sample of proteins has to be collected. The proteins in the sample are fragmented using the enzyme trypsin, which breaks the proteins into peptides. A peptide sample can be analyzed by a mass spectrometer, an instrument that can separate and quantify molecules by their mass to charge ratio. Tandem mass spectrometry combines two mass spectrometry steps with a fragmentation step to only measure the ions of a single peptide. The ions are sorted based on their mass-to-charge ratio. As a result the ion signal can be plotted as a function of the mass-to-charge ratio forming a mass spectrum (figure 2.1). Based on mass spectra one can draw conclusions for the structure and the composition of the measured samples. Mass spectra are the basis for all algorithms that want to identify the amino acid sequences of peptides or proteins [15].

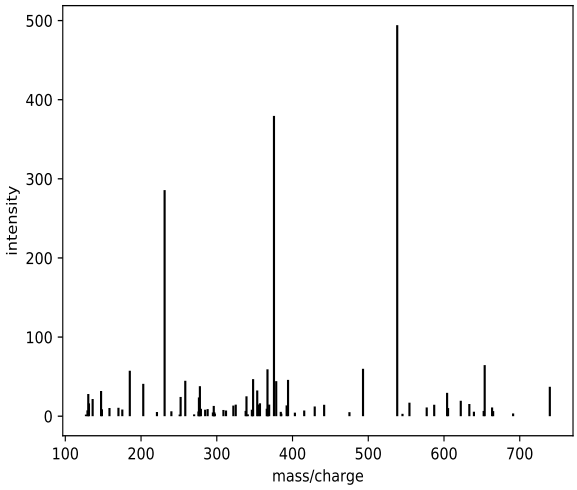


FIGURE 2.1 Mass spectrum of the peptide ”DDYVEK” retrieved from the PRIDE archive.

Mass spectra can be identified with protein database searches and with de novo peptide sequencing. There are many large protein databases like PDBe or UniProt. A typical database search algorithm compares the experimental mass spectra with the theoretical mass spectra derived from a protein database. However the experimental mass spectra acquired by mass spectrometry deviate from the theoretical spectra. These deviations come from noise peaks and missing peaks and also from the modifications that occur during the formation of a protein. Protein database search algorithms calculate a matching score between experimental mass spectra and the theoretical mass spectra. The peptide with the highest matching score is chosen as the peptide in the sample. A validation is required to identify the likely candidates among the peptide matches. Proprietary tools and implementations for protein identification with database searches

like MASCOT and SEQUEST are commercially available. However this approach can not identify those peptides in the sample that are not included in the protein database.

The issue can be tackled by peptide identification with de novo peptide sequencing. De novo peptide sequencing, which is often only refereed to as de novo sequencing, is a global optimization problem where a peptide sequence has to be found such that the total mass of the sequence is approximately equal to the experimental peptide mass.

Instead of searching a large peptide database the de novo peptide sequencing method creates a peptide sequence by interpreting each peak of the underlying mass spectrum based on the mass-to-charge ratio and the peptide mass. De novo peptide sequencing does not require knowledge about possible sequences *a priori*. One by one, the algorithm creates a sequence of amino acids for the peptide. De novo peptide sequencing is ideal for the identification of novel peptides. On already identified proteins de novo peptide sequencing will not reach the same performance as a database search algorithms. Hybrid systems of de novo peptide sequencing and database searches have also been the topic of ongoing research [16]. A proprietary and commercial implementation is available in the PEAKS software [17, 18]. This work focuses on the de novo peptide sequencing problem for peptide identification.

2.3 PRIDE database

Proteomics **Ide**ntifications database is a file-based public data repository created by the European Bioinformatics Institute for the results of MS/MS-based proteomics. The database stores mass spectra, identified proteins, identified peptides and associated meta data. By September 1, 2015 the archive contained more than 298 million identified peptides and over 690 million spectra. 25% of the submitted data sets were marked as 'Complete', which means that mass spectra as well as their peptide identifications are included. The total data size of the 3336 individual projects reported to the PRIDE archive was approximately 140 Terabyte (TB) [19]. In the last years the number of projects has almost doubled as 5579[1] individual proteomics data sets were reported by August 8, 2018. The increase in proteomics data sets shows the continuously growing importance of proteomics research, resulting in a massively increased data volume. The centralized archive provides a standardized way for sharing proteomics data and the database is relying on researchers to submit their proteomics data. The database provides a RESTful API allowing applications to access the results stored in the database without requiring a specific programming language [19, 20].

The PRIDE archive offers researchers to store the peak lists of mass spectra in the widely used and proprietary Mascot Generic Format (.mgf) developed by Matrix

[1] *PRIDE Archive*. Retrieved August 8, 2018, https://www.ebi.ac.uk/pride/archive/simpleSearch?q=&show=10&page=0&sort=publication_date&order=desc

Science (London, UK). However PRIDE encourages the use of the standard formats mzML and mzIdentML. [21]

The mass spectra are matched to the peptide and protein in an mzIdentML (.mzid) file. The mzIdentML file format is one of the standards developed by the Proteomics Informatics working group. The format is based on Extensible Markup Language (XML) with special tags for the proteomics use case.

2.4 Machine Learning

Advances in computer hardware, especially Graphics Processing Unit (GPU)s, have allowed machine learning on large data sets to become feasible for a multitude of classification and recognition problems. As a result techniques, like RNNs or CNNs, can be applied to many data categories. This also lead to the creation of frameworks and libraries focused on machine learning and them becoming available and stable. Libraries, like TensorFlow, have made it relatively simple to apply machine learning to a specific problem without having to focus on the implementation of these basic techniques [22]. Machine learning as a field of computer science, utilizes statistical techniques to allow computers to learn features of a data set without having to program these features explicitly. Deep learning is a part of machine learning and includes specific techniques, like deep neural networks, recurrent neural networks or deep belief networks. In this work Deep Neural Network (DNN) and especially Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM) are utilized.

2.4.1 Neural Networks

A deep neural network is a multi-layered artificial neural network (figure 2.2), which is inspired by the biological connections between neurons in biological brains. A neural network consists of artificial neurons, activation functions, biases and weighted connections between the neurons. The bias of a neuron is a weighted connection where the input is one. These neurons form layers where neurons are connected to the next layer. A solution in a feed-forward neural network is formed by passing the input from the input layer through the hidden layers and forming at the output layer. Each connection between neurons is weighted. The input of a neuron is the weighted output of the previous neuron. The neuron applies its activation function to the input and passes the result as output to the next neuron. Input layers are special as they are connected directly to the input values with a connection that is weighted with one. Their output is connected to the first hidden layer. An output layers' input is the output of the last hidden layer. The output layer computes the values of the neural network output.

RNNs and CNNs are special variants of classic neural networks [23, p. 9-13]. The data set for a machine learning solution is the most important component besides the machine learning architecture. The data set has to contain good samples for each class that the machine learning solution should recognize. Commonly the data set will be divided into multiple subsets for validation, training and testing. The split between the sets prevents over-fitting and can prove that the learned features of the training set can also be applied to a testing set that has not been trained to the machine learning architecture [15].

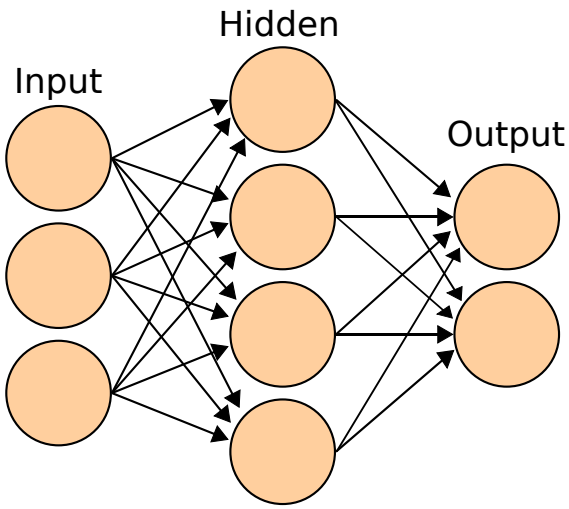


FIGURE 2.2 Feed-forward neural network with connected input layer, hidden layer and output layer composed of simple artificial neurons.[3]

Forward Propagation

Forward propagation is the transfer of information of a neural network through the neural network architecture without cyclic information transfer. The information passes forward through the input nodes, hidden nodes to the output nodes.

Activation Function

The activation function is applied to the input of a neuron. The result of the function is the output of the neuron. In simple neural networks the sigmoid function $S(x)$ (equation 2.1, figure 2.3) is commonly used because result of the derivative (equation 2.2) can be calculated easily. Sigmoid functions have domain of all real numbers with the return value increasing monotonically.

[3]Colin M.L. Burnett, *ANN*. Retrieved August 6, 2018, from `https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg`

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

Sigmoid function

$$S'(x) = f(x)(1 - f(x)) \tag{2.2}$$

Derivative of sigmoid function

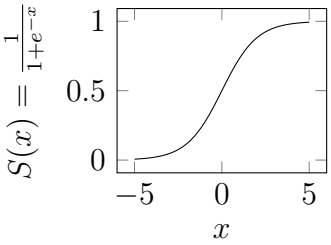


FIGURE 2.3 Sigmoid Function Plot

Error function

The error function E is utilized in a neural network to calculate the error between the output of a neuron and the desired output. An example for an error function is the squared euclidean error (equation 2.3).

$$E = \sum \frac{1}{2}(target - output)^2 \tag{2.3}$$

Squared Euclidean Error

Training

During the training phase of neural networks the training set is used as the input. The training set contains known inputs and their corresponding outputs. In neural networks that are only containing one hidden layer a gradient descend method such as the delta rule would reduce the error of the neural network by updating the weights. The delta rule is applied to the weight of the neuron j with the ith weight w_{ji} of the hidden layer. The error of the output is calculated by subtracting the target output t_j with the actual output y_j. The error times the learning rate η multiplied with the derivative of the activation function $g'(h_j)$ and the input x_i results in the delta. Where h_j is the weighted input of neuron j. The resulting delta is added to weight w_{ji} (equation 2.4).
However neural networks containing more than one hidden layer require a different

$$\Delta w_{ji} = \eta(t_j - y_j)g'(h_j)x_i \tag{2.4}$$

Delta Rule

method as gradient decent with the delta rule is not possible. This problem is solved by the backward propagation of errors.

Backpropagation can be divided into two distinct phases. During the forward propagation the input is propagated through the neural network to generate the output values. The error between the output and the targeted output of each output neuron of the neural network is calculated with the previously selected error function. The sum of the individual errors of all neurons is the total error of the neural network. In the second phase the partial derivative of the total error with respect to the individual weights of each neuron is calculated. The partial derivative can be calculated by applying the chain rule from the output neuron backward to the individual weights. The individual weights are updated by subtracting the partial derivative times the learning rate. This process is repeated for each weight and bias in the neural network [24, pp. 203– 212].

Epochs

During the training of a neural network an epoch describes propagating all training examples forward and backwards once. An epoch of training is followed by a forward propagation of the validation data set to verify if the neural network was successfully trained.

CNN

CNNs (figure 2.4) are modeled after the human eye with neurons only connected to a subset of the neurons of the previous layer. Convolutional neural networks have been successfully applied to image recognition and feature selection tasks with minimal prepossessing. The neural network can be trained to learn the image filters that would have been hand-engineered in conventional image recognition systems [25]. During the training of a CNN the functions for the different sampling or pooling layers are treated as activation functions.

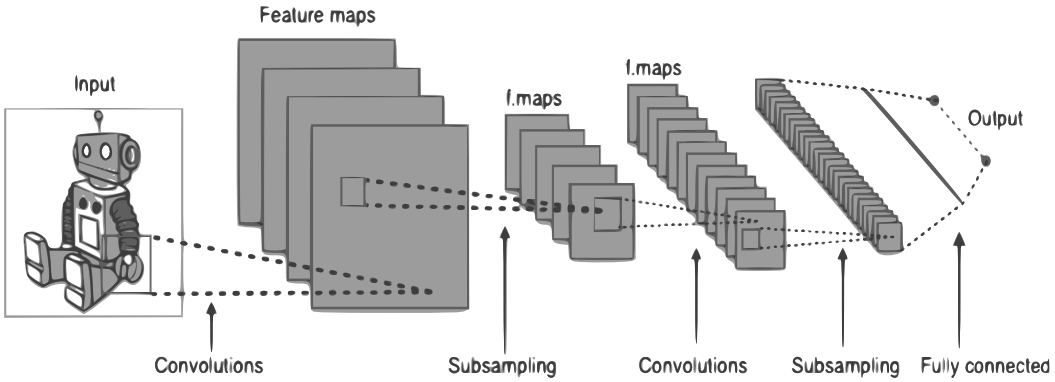


FIGURE 2.4 Typical CNN structure containing convolutional and sampling layers.[4]

RNN

Recurrent neural networks are special neural networks whose neurons possess a connection to themselves allowing the retention of previously computed solutions inside the neuron (figure 2.5). Simple RNNs contain a network of artificial neurons. Each neuron has an activation function and has a connection to the next neuron with a real-valued weight for the connection. This adds the benefit of retaining the information of the previous output for the generation of the current output. The past output influences the future output, which is a helpful feature for learning data sequences that depend on previous output. RNNs are successfully utilized to solve a variety of problems like speech recognition, time series prediction or handwriting recognition among other usages [26].

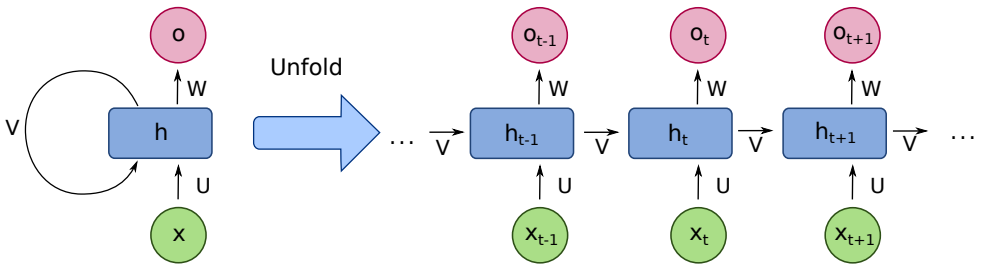


FIGURE 2.5 Unfolded recurrent artificial neuron over its time steps.[5]

[4] Aphex34, *CNN*. Retrieved August 6, 2018, from `https://commons.wikimedia.org/wiki/File:Typical_cnn.png`

[5] François Deloche, *RNN*. Retrieved August 6, 2018, from `https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`

Long short term memory

Long-short-term-memory cells (figure 2.6) are used in RNNs that are called LSTM networks. A LSTM cell is composed of an input gate, output gate and forget gate, which are neural network neurons. The gates are simple neurons with a sigmoid activation function with an output range of $[0, 1]$. The output of a gate is multiplied with the input. The multiplication with a value between zero and one modifies how much information of the input is let through. The forget gate modifies the ”memory” while the input and output gate modify the input and output. A cell is able to retain and improve its short term state for long times. The internal memory or state of a LSTM cell can be extracted and implanted into another LSTM cell allowing the processing of different inputs with the same internal state. LSTM networks have been successfully applied to the problem of neural machine translation, where the implanting of the internal state creates an encoder-decoder LSTM network. The encoder trains the internal state with an input (e.g. translated french language sentences) and the decoder uses the same internal state to generate an output with different input (e.g. German language sentences) [27].

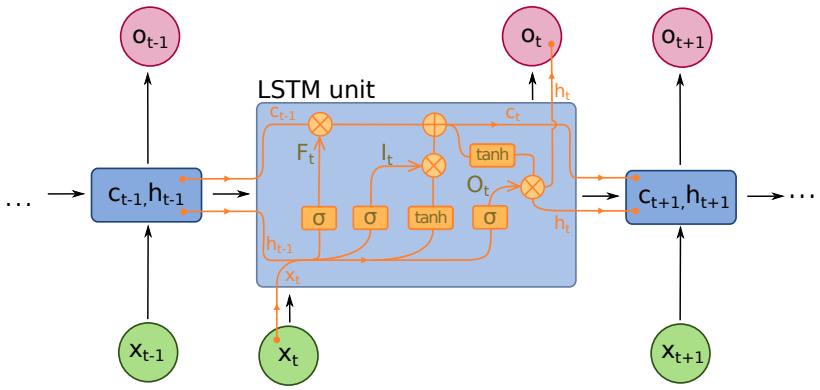


FIGURE 2.6 LSTM cell containing *tanh* and *sigmoid* neurons as gates.[6]

Encoder-decoder architecture

An encoder-decoder neural network architecture is composed of two neural networks that are combined to form a compressed representation of the input data. The encoder neural network reduces the size of its output layer to create a reduced representation of the input. The reduced representation is a vector with the size of the output-layer of the encoder. The input of the decoder network decompresses the reduced representation to the initial input of the encoder. The compressed vector representation can be used in other machine learning algorithms instead of the uncompressed input, because key features are reliably stored in the vector representation if the error between

[6] François Deloche, *LSTM*. Retrieved August 6, 2018, from `https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg`

decompression output and compression input is minimal. This architecture has been successfully applied to the task of image compression. During the compression with an encoder-decoder architecture data is lost. However this loss is intended because if the decoded data has only small deviations from the original it can be concluded that only insignificant features are lost during encoding.

Sequence to sequence

A sequence to sequence architecture is an encoder-decoder neural network applied to problems that require different input and output lengths. The encoder accepts inputs with a maximum length. The maximum length is predefined by the maximum length of sequence in the training data set. The decoder generates a single symbol as output. The combined generated symbols create the sequence of variable length as output. This feature allowed sequence to sequence models to be successfully applied to the problem of neural machine translation [12, 28].

2.4.2 TensorFlow

TensorFlow is an interface for expressing machine learning algorithms and executing such algorithms. The interface provides heterogeneous systems with the ability to execute machine learning algorithms. It is the result of the Google Brain project and is still being expanded and refined. The machine learning algorithms are formed into a directed graph that allows dataflow programming. The TensorFlow Application Programming Interface (API) supports Python and C++.

2.4.3 Keras

Implementations of the aforementioned machine learning components are available with Keras, a high-level framework for neural networks, written in Python. The API allows switching between different machine learning back ends without requiring changes to the source code. Tensorflow is available as a back end [29].

CHAPTER 3

Concept

As I stated in the first chapter this thesis focuses on the de novo optimization problem for the identification of peptide sequences. To solve problems with the current methods of peptide identification the DeepPSM model was created. This chapter will explain the DeepPSM architecture, the continuous integration of training data and the different deep learning models.

3.1 DeepPSM architecture

Figure 3.1 shows the architecture of the DeppPSM software tool. The DeepPSM application combines the public proteomics data archive PRIDE as a resource for training data with a deep learning approach for peptide identification. Since the data stored in the PRIDE archive is created from real-world samples and a large data size is available it is ideal for the usage with a deep learning solution. The PRIDE archive is based on file based projects that contain the mass spectra and the sequences of proteomics samples. These files can be automatically retrieved, processed and prepared for the training and testing of a deep learning model. DeepPSM is a tool that was created with the end goal of being usable by proteomics researchers that plan to utilize the PRIDE archive as background knowledge for peptide identification with deep learning. The trained deep learning model will be saved and is utilizable for peptide identification by the user or to continue the training with a modified or different data set. The identification with a deep learning model offers advantages as classic imperative de novo sequencing methods utilize time consuming searches and optimization algorithms. The time neccessary for a prediction by a deep learning solution is done in milliseconds, while other methods take much longer to iterate through the possible data. The PRIDE archive data is also gathered from the proteome of organisms of many species, which makes the DeepPSM system able to train the deep-learning models for the task of proteomics.

The DeepPSM system is combining a deep learning model with PRIDE archive data access and data preprocessing. The deep learning model can be trained, saved and retrained. The goal of the trained deep learning model is being able to predict peptide

sequences. For each mass spectrum received as an input the deep learning model will create a peptide sequence as an output. The quality of the output peptide sequence is influenced by two factors, the model architecture and composition and the training data that is used for training. While the output of the DeepPSM system is a peptide sequence the intermediate step depends on the deep learning model and implementation. The PRIDE data retrieval component will automatically create the training, validation and testing data sets. The composition of the data set depends on the attributes that can be chosen by the user. Each data sets contains mass spectra with their identified sequence and meta data.

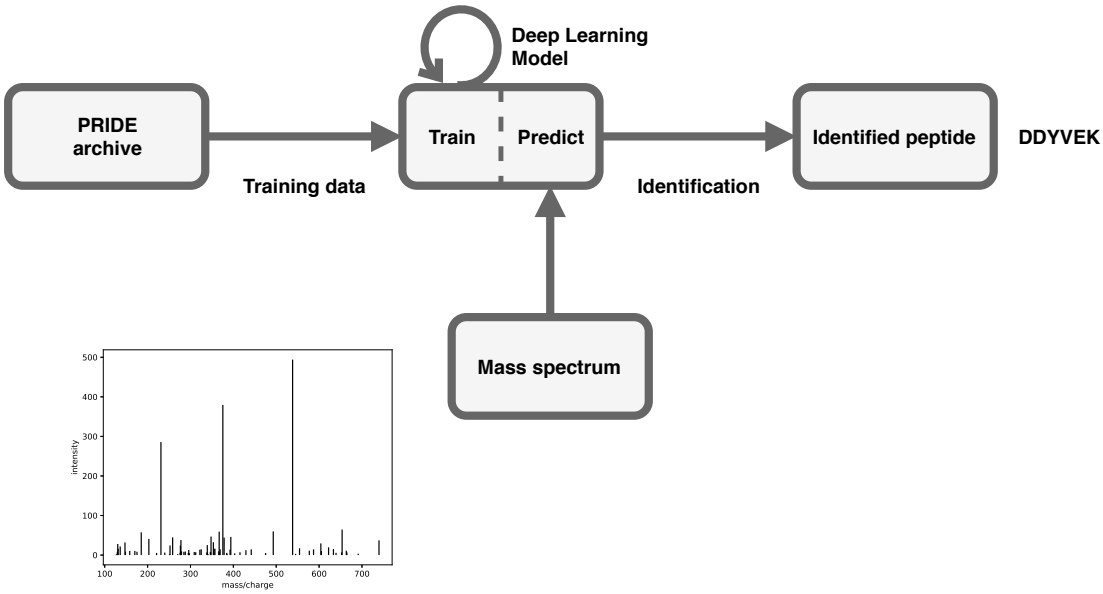


FIGURE 3.1 Simplified overview of the DeepPSM system.

3.2 Data

For training and evaluating a neural network real world data is necessary. Proteomics data can be retrieved from proteomics databases and in the case of DeepPSM data is retrieved from the PRIDE archive. The data comes in the form of structured but different file formats that require preprocessing and merging. PRIDE utilizes the ProteomeXchange standard, which defines a set of workflows, access patterns and allowed file formats [30]. The PRIDE archive provides an open REST API, which allows to download the data. PRIDE projects (figure 3.2) are accessed by their project accession id and contain Mascot Generic Format (MGF) files with mass spectra and mzIdentML files with peptide sequences. DeepPSM uses the REST access to download the relevant files of a project and combine the result files to a training file. For the training file format a MGF structure with an additional attribute for the sequence in the header of each spectrum is defined.

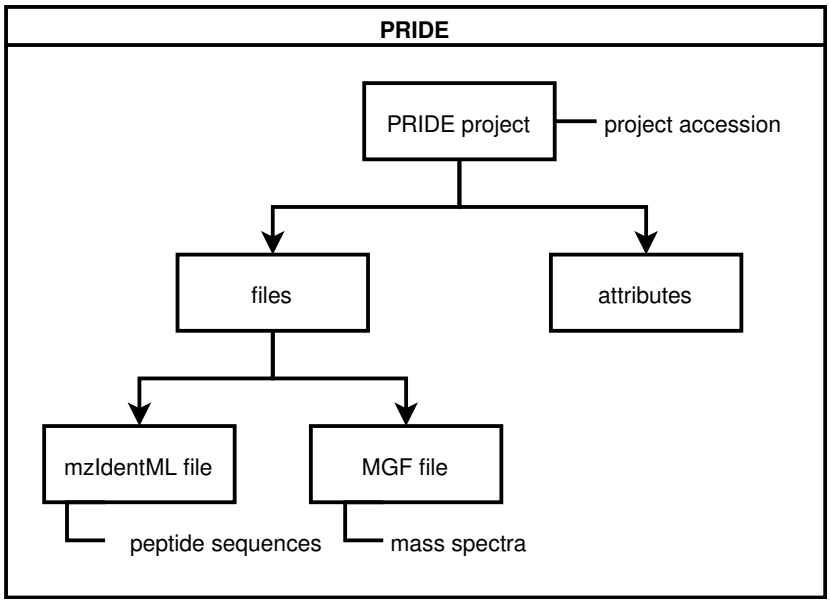


FIGURE 3.2 Overview of a PRIDE project.

3.2.1 Binning

The deep learning architectures require a binning of the mass spectra because of different lengths, maxima and minima of the peak lists that can be generated from the mass spectra. The binning is used to create a feature vector representation of a spectrum with the first dimension being the mass-to-charge dimension and the second dimension representing the intensity dimension of the mass spectrum. During the binning step mass spectrum peaks are assigned to a bin in the feature vector (figure 3.3). As this binning is applied to each mass spectrum the mass spectra are transformed into fixed dimensions that allow the creation of the fixed neural network input layer. The input layer requires a fixed size because neural networks necessitate defined number of weights that are being trained. Changing the size of the layers during training would require retraining of the layer because of the new or lost interconnections between the cells of the layer. Since the accuracy and the resolution of mass spectrometer increases in the last years, a minimum size of 10 000 bins is needed for the spectra data.

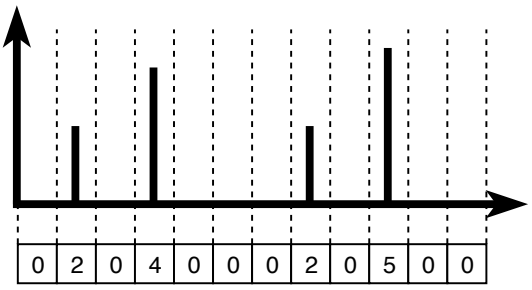


FIGURE 3.3 Binning of a mass spectrum.

3.3 Deep learning architectures

For the deep learning system multiple implementations are evaluated and considered: a sequence to sequence RNN, a convolutional sequence to sequence RNN and the DeepNovo deep learning approach to peptide identification introduced by Ngoc Hieu Tran et al. in 2017 [12].

3.3.1 Sequence to Sequence RNN

The first neural network model implemented for the evaluation of the DeepPSM architecture utilizes a sequence to sequence structure. The input for the sequence to sequence model is the binned feature vector of a mass spectrum. Sequence to sequence architectures have been successfully applied to the task of neural machine translation [12, 28]. A sequence to sequence structure is an encoder-decoder RNN that is able to decode a fixed size input into a variable length sequence output. The variable length output of the DeepPSM sequence to sequence model is a peptide sequence. The encoder and decoder are each represented by a LSTM layer of fixed size. The encoder LSTM layer transfers its internal state to the decoder LSTM layer. The internal state is created during the feeding of input data. As more data is put into the encoder the internal state becomes more refined and the actual output of the encoder layer is discarded. The internal state is implanted into the decoder layer. The decoder layer input is a one-hot encoded vector structured after its output vector. A one-hot vector is an array of bits with only a single high (1) bit. The element of the vector that represents the previously predicted class is marked and the output is with an element for each amino acid symbol. The last layer of the encoder-decoder architecture is a fully connected layer with a softmax activation function that outputs the distribution over the possible amino acids, their modified versions and the start and stop symbol. A peptide in a sample can be affected by different modifications from the preparation and handling of the sample. Since modifications of can change the mass of the underlying amino acid, each has to be represented as a separate symbol. An increase of possible symbols for peptide sequences is the consequence.

The softmax function's (equation 3.1) output tells the probabilities for each amino acid and produces values in the range $(0, 1)$. An activation of the jth neuron is applied to its jth input. The vector z contains K inputs. It highlights the largest values and tries to suppress values which are below the maximum value, with the sum of its resulting values always being one [31, p. 22-7].

$$o(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.1}$$

Softmax Activation function

The symbol with the highest score in the distribution is concatenated with previously output symbols to create the final sequence. This output is transformed into a one-hot vector and utilized as the decoder input for the prediction of the next symbol. The first input to the decoder is always the start symbol. For the training of the sequence to sequence model a RMSProp optimizer (equation 3.2) is utilized, which is an unpublished adaptive learning rate method that is a very robust optimizer [24, pp. 307–308] [32]. Root Mean Square Propagation adapts the learning rate for each weight w of the neural network. An optimizer minimizes or maximizes the parameters of an objective function (error function) E used in the neural network. In this case the optimizer minimizes the loss function by updating the weights and biases of the neural network.

$$\begin{aligned} MeanSquare_t &= \gamma MeanSquare_{t-1} + (1-\gamma)(\nabla E_i(w))^2 \\ w_{t+1} &= w_t - \frac{\eta}{\sqrt{MeanSquare_t + \epsilon}} \nabla E_i(w) \\ \gamma &= 0.9 \\ \eta &= 0.001 \end{aligned} \tag{3.2}$$

RMSProp Optimizer

For the loss function categorical cross entropy (equation 3.3) is utilized. This loss function works well for data sets were each data point can be represented by a class. For this function the target vector for the training has to be a one-hot vector with length of valid symbols and a one in the index of the symbol that is represented by the vector. Where y_i is the predicted probability value for class i and y'_i is the target probability for that class.

$$E(y', y) := -\sum_i y'_i \log(y_i) \tag{3.3}$$

Categorical Cross Entropy

This function was chosen because the amino acids are mutually exclusive as one element of a peptide sequence can only be represented by one amino acid. The usage of the previous timestep's output as the input of the next timestep is called teacher forcing [24, pp. 381–382].

In figure 3.4 the combined sequence to sequence model and binning step (section 3.2.1) are visualized as they create the amino acid sequence. The mass spectra are binned and the binned feature vector is feed to the encoder LSTM network. The encoder output is not considered for future calculations. Afterwards, the internal state of the encoder is transferred into the decoder LSTM network. The decoder creates a variable length sequence by emitting a single amino acid symbol after receiving the start symbol or the symbol of the previous iteration.

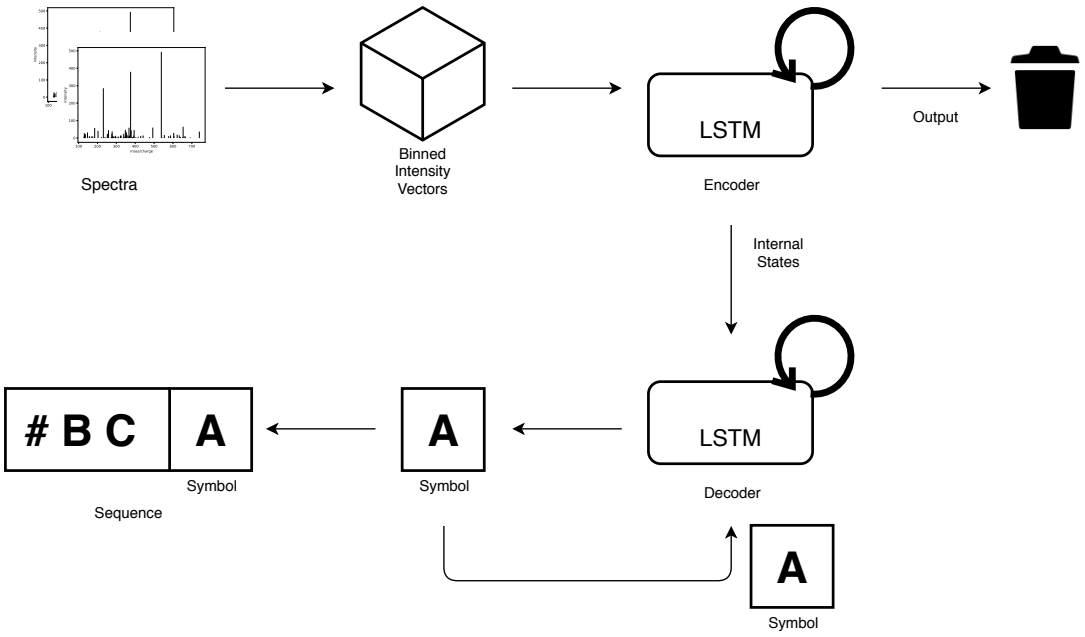


FIGURE 3.4 Visualized sequence to sequence model architecture in DeepPSM.

3.3.2 Convolutional Sequence to Sequence RNN

The convolutional sequence to sequence model utilizes the sequence to sequence structure that was introduced above in combination with a convolutional neural network. As neural networks have been successfully applied to the problem of feature extraction from two dimensional data. The combination of feature extraction from the binned input data with the benefits of a sequence to sequence architecture was deemed of interest for evaluation. The CNN that was applied combined two convolutional layers, a max pooling layer and a dropout layer. Convolutional layers are special neural network layers where each neuron is only connected to a subset of the input. The size of the subset is defined by the kernel size. If the kernel size is bigger than the size of the input the overlapping undefined inputs have to be padded to create a full kernel. The activation function utilized for the convolutional layers is the ReLu. The Rectified Linear Unit (equation 3.4, figure 3.5) is the most popular activation function that is used for convolutional neural networks and deep learning [24, p. 16]. The range of the function is $[0, \infty)$ as it simply thresholds the input at zero. ReLUs result in much faster training for large networks (e.g. factor of 6 in the work of Krizhevsky et al. [33]).

$$f(x) = \max(0, x) \tag{3.4}$$

Rectified Linear Unit

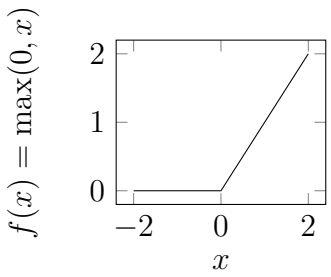


FIGURE 3.5 Rectified Linear Unit Plot

A max pooling layer is another layer that is commonly applied in CNNs. The layer down samples the input by applying a max filter on the subregion of the input. This layer also reduces the dimensionality of the input. The dropout layer disables the units of the previous layer with a probability of p. Dropout layers have been proven to be able to prevent overfitting [34].

3.3.3 DeepNovo

The DeepNovo tool is a deep learning tool for peptide identification [35, 36]. It supports de novo sequencing for peptide identification. Instead of utilizing data sets of fixed size, DeepNovo in the DeepPSM tool will be trained with the automatically generated real-world data retrieved from the PRIDE archive. The DeepNovo model is automatically configured, trained and utilized in peptide prediction. The utilization of the DeepNovo model in DeepPSM will require changes to the source code of DeepNovo. As DeepNovo is the only available implementation of a deep learning model for peptide identification it also provides a benchmark for the aforementioned models. However the DeepNovo model is more complex then the aforementioned deep learning models and is composed out of two convolutional neural networks, a recurrent LSTM network and a fully connected neural network. Figure 3.6 shows DeepNovo and its components. The model utilizes a CNN called *spectrum-CNN* for feature extraction from the mass spectra. The extracted features are the input for the *ion-CNN* that learns the peaks of the spectrum with the mass of the previous output and summarizes the overall information. The model looks up the embedding vector of the previous output that is utilized as a prefix and uses them as input for a LSTM network. The output of LSTM and *ion-CNN* are combined by fully connected neural layers to form the probability distribution for the amino acid. This process is repeated for each amino acid of a sequence [35].

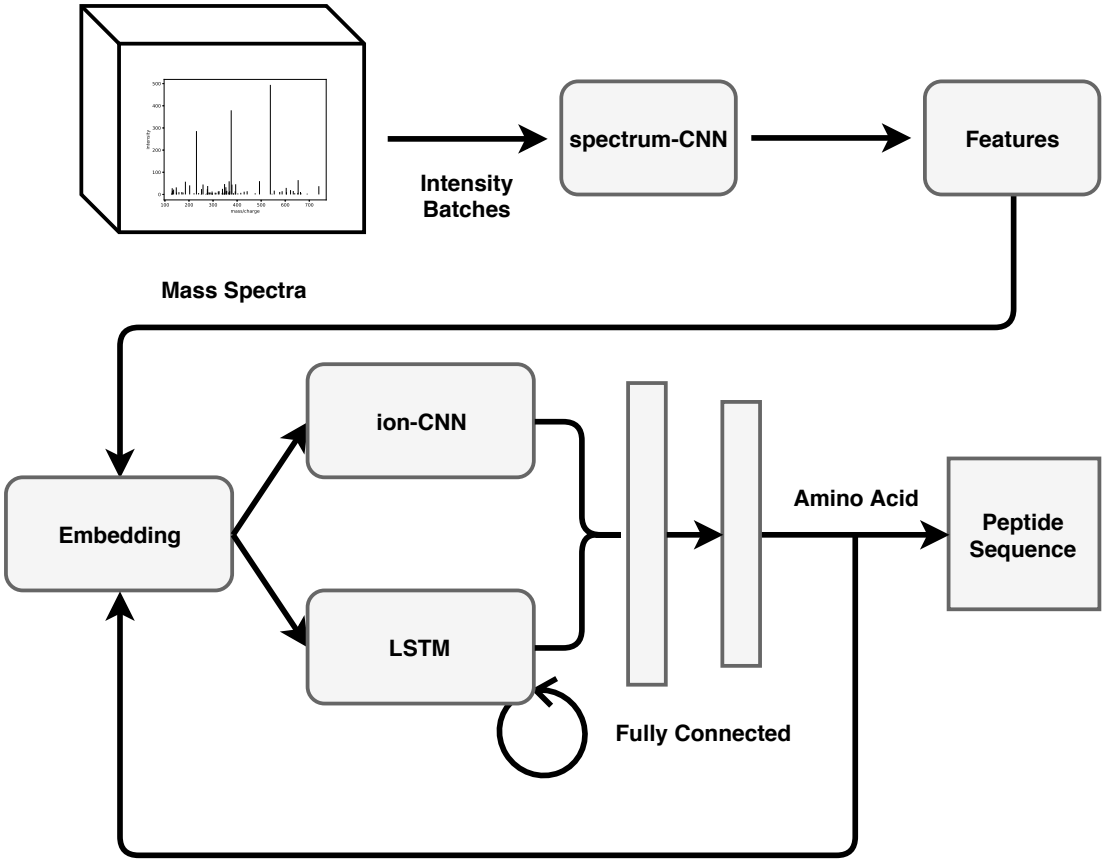


FIGURE 3.6 Overview of DeepNovo

CHAPTER 4

Implementation

In this chapter I will explain the sample implementation of the DeepPSM system and the challenges that were encountered during the development.
After the model architectures were designed, development for a sample implementation was started. Since I have previously used the programming language Python and many frameworks, modules and samples are freely available for it, Python 3.5 was chosen as the programming language for the implementation sample. Tensorflow, Keras and the proteomics framework pyteomics can be integrated into a Python 3.5 application. However a program implemented in Python will under certain conditions not reach the performance of a dedicated C++ program. This drawback was considered but the benefits of using Python to allow a fast development process outweigh the aforementioned drawback.

4.1 Hardware

The hardware that was utilized for the testing and creation of the results was a cloud server that was configured to allow fast training and testing of deep learning models and a desktop PC. However the hardware abstraction of the utilized deep learning frameworks achieves the same prediction performance for each deep learning model regardless of utilized hardware. The hardware specification of the cloud server are shown in table 4.1 and the specifications of the desktop computer are specified in the table 4.2.

Hardware	
Component	Specification
CPU	24 virtual cores from Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
RAM	252 GBs
GPUs	2 x Nvidia GTX1080ti with 11GB GDDR5X
ROM	1.8 TBs

TABLE 4.1 Cloud server hardware

Hardware	
Component	Specification
CPU	4 Cores Intel 6600k @ 3.9GHz
RAM	8 GBs
GPUs	Nvidia GTX1070 with 8GB GDDR5
ROM	250GBs

TABLE 4.2 Desktop PC hardware

4.2 PRIDE archive API

Before the development of the machine learning component could be started, the data sets for training, validation and testing had to be created. As specified in the previous chapter it was decided to utilize the data stored in the PRIDE archive because of the mentioned advantages. The PRIDE archive is accessed with the provided RESTful API[1]. REST[2]ful APIs provide a set of HTTP requests on the client side to trigger HTTP responses on the server side that provide interoperability. In the case of PRIDE project data access is possible with a GET HTTP request to an URL[3]. HTTP requests in Python can be realized by the *requests* module. The response is a file in JavaScript Object Notation (JSON)[4] format containing n projects with their attributes and the id that uniquely identifies the project. Python 3.5 is able to directly access JSON similar to standard dictionaries. The accession ids of the requested projects matching the

[1] *PRIDE RESTful web service*. Retrieved August 6, 2018, `https://www.ebi.ac.uk/pride/help/archive/access/webservice`

[2] Representational State Transfer

[3] *PRIDE archive*. Retrieved August 6, 2018, `http://www.ebi.ac.uk/pride/ws/archive/project/list/?show=1000`

[4] *JSON* format is a standardized human readable text format for the transfer of name-value pairs and serialized objects. `https://json.org/`, Retrieved August 6, 2018

user-defined attributes will be saved by the program. The DeepPSM software uses the REST API functions to retrieve the mass spectrometer files in the MGF format and the identified peptide sequences in mzIdentML format for the project with the project id. The spectrum data from the MGF file and the peptide sequence of the identification file are used to create a peptide spectrum match. A collection of such peptide spectrum matches constitutes the training data. The training data uses an extended MGF format. An overview of the utilized functions of the RESTful API is provided in table 4.3.

PRIDE access		
Functions	RESTful API request	Response
Retrieval of n project ids	http://www.ebi.ac.uk/pride/ws/archive/project/list/?show=n	JSON containing id and project attributes
Retrieval of project id	http://www.ebi.ac.uk/pride/ws/archive/file/list/project/id	JSON containing MGF and mzIdentML links

TABLE 4.3 Utilized RESTful API functions.

4.3 MGF Parsing

After having retrieved the projects with the matching attributes the required mzIdentML and MGF files will be downloaded. These files are gzip[5] files, which have to be decompressed with the *gzip* module of Python 3.5 and are saved by the program. The proprietary MGF format contains a header and a peak list, which are delimited by the statement pair *BEGIN IONS* and *END IONS*. The header can be filled with meta data about the attached peak list. Most importantly the header can contain unique identifier marked with *TITLE* and the decoded amino acid sequence introduced by the *SEQ* statement. The *SEQ* header property is added by the DeepPSM tool. Listing 4.1 displays a header from the training data set of DeepPSM, which contains the *SEQ* and *TITLE* attributes. The peptide mass and charge of the measured peptide can also be provided by the header. The peak list is space separated with a column for the mass-to-charge ratio and another column for the measured intensity. The MGF file is accessed with the pyteomics module, which is available through the Python Package Index PyPi[6]. Pyteomics contains implementations of tools to handle proteomics data. If a MGF file

[5] *gzip* is a file format and application used for lossless compression and decompression based on the DEFLATE data compression algorithm. `https://www.gnu.org/software/gzip/`, Retrieved August 6, 2018

[6] *Python Package Index*. Retrieved August 6, 2018, `https://pypi.org/`

is passed to the MGF class then the pyteomics library instantiates a reader class that can iterate through the saved spectra inside the MGF file. The spectra are returned as a dictionary, which also contains the header data of the MGF file including the peptide sequence.

```
1 TITLE=id=PXD009549;MEM_MIA_3_LN_150702155601.
      mzid_MEM_MIA_3_LN_150702155601.MGF;index=6140
2 PEPMASS=754.4324336666667
3 CHARGE=3+
4 SEQ=LKLVGTLPASLGVEIPANSEPR
```

LISTING 4.1 MGF header with *TITLE* and *SEQ* attribute.

4.4 mzIdentML Parsing

The mzIdentML 1.2.0 format is a standardized file format based on the XML format with special XML elements for the proteomics use case. The standard was defined by the Human Proteome Organization (HPO) Proteomics Standards Initiative (PSI). Tags are standardized for identified protein, matched spectrum and peptide. In this work the .mzid files provided by PRIDE were only utilized to attach the peptide sequence to identified mass spectra for the creation of testing, training and validation data sets. The elements of interest for this task are *SpectrumIdentificationResult* if it contains a valid *SpectrumID* attribute. The enclosed *SpectrumIdentificationItem* is accessed to retrieve the *peptide_ref* attribute (figure 4.1). If the *peptide_ref* can be matched to the *id* attribute of a *Peptide* element the *PeptideSequence* (figure 4.2) element of the Peptide is accessed and the *text* attribute of the *PeptideSequence* is saved with the matched *spectrumID* attribute by the program for further usage. [37].

```
<SpectrumIdentificationItem chargeState="2" experimentalMassToCharge="451.2285767" calculatedMassToCharge=
"451.2289155" calculatedPI="4.05" peptide_ref="8359" rank="1" passThreshold="true" massTable_ref="8" sample_ref=
"SAM_0" id="SII_8359_54_SEQ">
        <PeptideEvidenceRef peptideEvidence_ref="PEV_SEQ_8359_5158"></PeptideEvidenceRef>
        <PeptideEvidenceRef peptideEvidence_ref="PEV_SEQ_8359_5159"></PeptideEvidenceRef>
        <PeptideEvidenceRef peptideEvidence_ref="PEV_SEQ_8359_59314"></PeptideEvidenceRef>
        <cvParam cvRef="MS" accession="MS:1001155" name="SEQUEST:xcorr" value="1.893254"></cvParam>
        <cvParam cvRef="MS" accession="MS:1001215" name="SEQUEST:PeptideSp" value="390.8432312"></cvParam>
</SpectrumIdentificationItem>
```

FIGURE 4.1 *SpectrumIdentificationItem* with enclosed *peptide_ref* from a mzIdentML file.

```
<Peptide id="8359">
    <PeptideSequence>QVEELER</PeptideSequence>
</Peptide>
```

FIGURE 4.2 *Peptide* element with *PeptideSequence* from a mzIdentML file.

4.5 Creation of training data sets

With the saved *spectrumID* from the mzIdentML file the spectrum has to be found in the MGF file. Furthermore, the matching peptide sequence are added as a header property *SEQ* to the spectrum from the MGF file. The modified MGF is saved inside the specified project folder. After the program created multiple MGF files for the projects the files are merged and shuffled to create a randomized data set. The randomization utilized the *random* module of Python 3.5. The *sample* function returns a unique list of k randomized elements from a population of samples. This large data set is then divided into the training, validation and testing set with the help of a user-defined split parameter. After the data sets are created training can be started using the modified MGF file containing randomized identified spectra.

4.6 Deep learning models

The deep learning component was implemented with Keras and the Tensorflow GPU back end. Keras offers a high-level API for neural network model creation. A model can be defined with the Sequential model API (listing 4.2), which allows to simply add layers upon the neural network model with the *add* function. The API contains functions for training, compiling and saving the neural network model. It is possible to add a variety of predefined layers, activation functions, loss functions or optimization functions to a model without the need to manage the low-level implementations. However the ability to define custom layers, model types or functions is also offered. A designed model is compiled by Keras for the specified back-end. In the case of Tensorflow a computation graph is automatically generated [29].

```
# instantiate Sequential model
model = Sequential()
# adding layers to neural network model
model.add(Conv2D(...))
model.add(Conv2D(...))
model.add(MaxPooling2D(...))
model.add(Dropout(...))))
model.add(Flatten())
model.add(Dense(...))
model.add(Dropout(...))
model.add(Dense(...))
# compiling neural network model
model.compile(...)
# training neural network model
model.fit(...)
```

LISTING 4.2 Sample of the sequential model API implementing a simple CNN.

Different layers were utilized for the two neural network architectures. The encoder-decoder LSTM model utilizes two *CuDNNLSTM* recurrent layers, which form the encoder-decoder model. The CuDNN implementations were chosen because of the available Nvidia GPUs as these implementations offer a speedup on Nvidia GPUs when compared to the standard LSTM layers. The LSTM layers can exchange their internal state to form the encoder-decoder structure. The encoder-decoder neural network with added convolutional layers utilizes two *Conv1D* layers, a *MaxPooling1d* and a *Dropout* layer as well as the encoder-decoder structure. Because of the large data sizes found in the PRIDE archive the ability of Keras to use a generator for the training process was utilized. The generator reads the user-defined size of training data in batches from the MGF file with the help of the pyteomics module into the system memory and passes it to the Keras training routine [29].

4.7 Training

For the training of the sequence to sequence and convolutional sequence to sequence model the training data had to be preprocessed because the data was not in a uniform shape. Since the data that is used for prediction and for training is created by experimental research the shape and range of different spectra is not the same for all. A MGF file contains the spectrum as a list of mass-to-charge and intensity pairs. The file format does not contain restrictions on value maxima or numbers of data entries. Binning is utilized to create uniform data out of the unsorted data. The user selects the resolution of the data, which causes direct modification to the input layer of the neural networks. The binning utilizes the *numpy* Python module. A two dimensional array is created with the first dimension for the mass-to-charge ratio and the second dimension for the intensity. The *digitize* function of *numpy* is applied to create the list of indices required to match each peak to its correct bin. The binned mass spectra are used to create a three dimensional training data array for each batch of an epoch. During training the generator passes the training data to the Keras training functions. Keras transforms the model into a Tensorflow computation graph and also incorporates implementations for the selected activation, loss and optimizer functions. The Tensorflow back-end of Keras executes the created computation graph.

4.7.1 DeepNovo

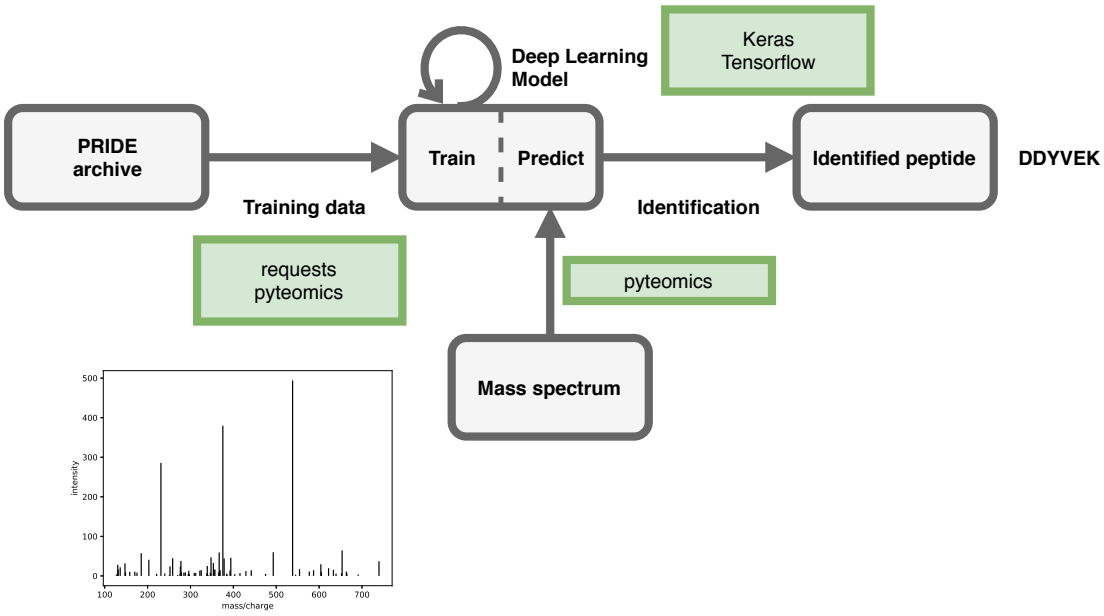
To use the DeepNovo[7] model in the DeepPSM system the DeepNovo code had to be adapted to Tensorflow 1.8 and Python 3.5. The MGF reader inside DeepNovo had to be improved to read MGF files with different numbers of header attributes and to

[7] *DeepNovo*. Retrieved August 6, 2018, from `https://github.com/nh2tran/DeepNovo`

accept unsorted header attributes. The configuration module of DeepNovo had to be integrated into the program to allow continuous training and updating of PRIDE data in conjunction with DeepNovo.

4.8 Overview of implementation

The conceptual architecture of DeepPSM (figure 3.1) presented in Chapter 3 can now be amended by the technologies chosen for the sample implementation (figure 4.3). The implemented data retrieval component for the PRIDE archive combines the modules *pyteomics* 3.5.1 and the *requests* 2.9.1 to access the file-based data archive. The MGF and MZidentML files are parsed and the data is used for the creation of training, validation and testing data sets for the deep learning models. The data sets are randomized with the *sample* function of the Python *random* module. Each data set is saved in MGF format. During identification and training the data sets are parsed by the *pyteomics* module and binned using the *numpy* module. The deep learning component and the models for the evaluation were implemented utilizing the Python modules *Keras* 2.1.6 and *Tensorflow* 1.8.



Module	Version
Keras	2.1.6
pyteomics	3.5.1
numpy	1.14.2
requests	2.9.1
Tensorflow-gpu	1.8.0

FIGURE 4.3 Overview of DeepPSM with utilized modules and versions.

CHAPTER 5

Evaluation

This chapter is divided into two parts, first the performance of the data retrieval component incorporated into the DeepPSM system will be evaluated. In the second section the prediction performance of the deep learning models in the DeepPSM system is evaluated. I evaluated the predictions with the PRIDE data set that was retrieved for the first section. Each section will be followed by a discussion of the results. Additionally, I locate advantages and disadvantages in the concept presented in Chapter 3.

5.1 Data retrieval component

In this section first the evaluation setup and the evaluation criteria for the implemented data retrieval component will be introduced and the data retrieval component will be evaluated.

5.1.1 Setup

In the beginning, I created a data set from the PRIDE archive that is composed out of 40 proteomics projects (appendix table 1, 2 and 3). The data was automatically prepared using the data retrieval component of the DeepPSM system. The data retrieval was configured as a greedy retrieval with the most recent 40 projects being retrieved. The projects had to be marked as *COMPLETE* by the researchers on PRIDE. The retrieved projects used for data preparation are containing peptides from the proteome of different species and spectra that were collected with different instruments. The real world data was created by a variety of different researchers.

5.1.2 Evaluation concept

For the evaluation of the data retrieval component that creates the necessary training, validation and testing data from the PRIDE archive. The recall of the mass spectra was chosen as an approximation for quality of the PRIDE data retrieval. The PRIDE

archive offers the attribute *numIdentifiedSpectra* for the project details of a PRIDE project. Since this attribute is optional and it is not verified by the PRIDE archive the criteria might not accurately reflect the quality of the data retrieval.

$$recall = \frac{|retrieved\ mass\ spectra \cap identified\ mass\ spectra|}{|identified\ mass\ spectra|} \tag{5.1}$$

Data Retrieval Recall

5.1.3 Results

While the raw project data contains 2,123,245 identified spectra the data prepared for training, testing and validation contained a combined number of 807,488 spectra with their sequences. The data retrieval component achieved a recall of *0.38*. The three data sets (table 5.1) reached a combined size of 3.4GB, thus only representing a small fraction of the total data volume stored in the PRIDE archive.

The splitting of the data results into three data sets, where the testing and validation data sets represents 10% each and the training data set represents 80%. The splitting parameter is defined by the user. The randomization results (table 5.2, 5.3) in data sets with similar characteristics the average peaks per spectra, average length per mass spectra and average peaks per amino acid are comparable between the data sets.

Data sets		
Data set	Number of spectra	Percentage
Training	645990	80%
Testing	80749	10%
Validation	80749	10%

TABLE 5.1 Retrieved data sets

Peaks in spectra		
Data set	Avg. peaks per spectra	Avg. peaks per amino acid
Training	262.43	16.07
Testing	262.79	16.05
Validation	262.33	16.04

TABLE 5.2 Properties of mass spectra

Length of sequences			
Data set	Maximum	Minimum	Avg. length per spectra
Training	92	3	16.33
Testing	87	3	16.38
Validation	89	3	16.35

TABLE 5.3 Properties of peptide sequences

5.1.4 Discussion

With only 38% of the identified spectra also present in the constructed data set, the data retrieval component performance offers room for improvements. While it cannot be ruled out that 62% of mass spectra are not present in the data set, the underlying *numIdentifiedSpectra* attribute may be an unreliable, since certain projects reported *numIdentifiedSpectra = 0*, while the project contained identifications. Another possibility for the discrepancy between retrieved spectra and *numIdentifiedSpectra* is the underlying one-to-many relationship between spectra and peptides for which I only retrieved one entry each. Therefore, the recall criteria for the data retrieval component should not be considered a real benchmark for the performance of the data retrieval.

After the division into the individual data sets the training set would contain 30.4% of the initial data, the validation and testing set would contain 3.8% of initial data each. The data retrieval component only supports spectra encoded in the MGF format with their sequence present in an mzIdentML file. If mass spectra in a data set are present in an unsupported file format they can not be extracted by the data retrieval component. The PRIDE archive does not verify the integrity of the files and compliance of a file to the file format standards. The author was able to observe files that have been uploaded containing non-standard alterations to the matching of sequences to spectra. This leads to the conclusion that the resulting recall of the identified mass spectra is also influenced by the quality of the data in the PRIDE archive. The analysis of the splitting and randomization of the data sets shows that the resulting three data sets are similar to each other. The data retrieval component of DeepPSM allows access to the data inside the PRIDE archive and the automated creation of training, validation and testing data sets. This offers the ability to utilize the data in the proteomics database as background knowledge. The data retrieval component does however require user configuration to create a data set with attributes like species, mass spectrometry instruments or sample treatments.

5.2 Deep learning models

In this section the evaluation setup and the evaluation criteria for the implemented deep learning models will be introduced. Furthermore, the deep learning models will be evaluated.

5.2.1 Setup

For the evaluation of the deep learning components the data sets generated for the evaluation of the data retrieval component are utilized as background knowledge, for training, validation and testing. The different deep learning models were trained for 50 epochs on the server (see Chapter 4) and on the desktop PC. After automated training was completed the different models were tested on the testing data set.

5.2.2 Evaluation concept

Before the evaluation of the deep learning models, it has to be verified if the adapted DeepNovo model is still capable of reproducing the results reported by Ngoc Hieu Tran et al. in 2017 [12]. This is achieved by testing DeepNovo on the same data utilized by Ngoc Hieu Tran et al. and comparing the results. For the evaluation of the different deep learning models criteria that allow a measurement of the quality of the predictions had to be found. To measure the quality of the deep learning prediction the criteria introduced by the authors of DeepNovo were also utilized for this evaluation [36]. The selected criteria are the peptide recall, amino acid recall and length recall (equations 5.2, 5.3, 5.4).

$$peptide\_recall = \frac{|peptides \cap predicted\ peptides|}{|peptides|} \tag{5.2}$$

Peptide Recall

$$aminoacid\_recall = \frac{|amino\ acids \cap predicted\ amino\ acids|}{|amino\ acids|} \tag{5.3}$$

Amino Acid Recall

$$length\_recall = \frac{|peptides\ lengths \cap predicted\ peptide\ lengths|}{|peptides|} \tag{5.4}$$

Length Recall

The peptide recall measures the share of correctly predicted sequences from all predicted sequences by the deep learning model. While this criterion provides an effective measurement for the quality of the prediction it offers no measurement for peptides that are only slightly deviating from the correct peptide sequence. These slight deviations are expected and frequently observed in imperative de novo sequencing algorithms. A predicted peptide that only deviates by one amino acid from the correct sequence would still be a good prediction if it is coupled with a value for the confidence of the deep learning solution. The amino acid recall measures the share of correctly predicted amino acids of all predicted amino acids. This removes the drawback of the first measurement, but a high amino acid is still possible with a low peptide recall. The length recall measures the share of correctly identified sequence lengths of all sequences. Because a neural network always produces an output, the number of identified sequences is equal to the number of inputs. While not as important as the quality of the predictions, the performance of the different models also has to be measured. To measure the performance of the deep learning model, the time required for the prediction of a single spectrum will be measured in seconds.

5.2.3 Results

In the work of Ngoc Hieu Tran et al. [35], DeepNovo was shown to outperform the conventional de novo peptide sequencing tools PepNovo, PEAKS and Novor. The area under the precision-recall curve[1] shows superior prediction performance for the Homo sapiens (figure 5.1) and the Saccharomyces cerevisiae (figure 5.2) proteome. However the changes to the implementation of DeepNovo require a verification that the prediction performance was not influenced.

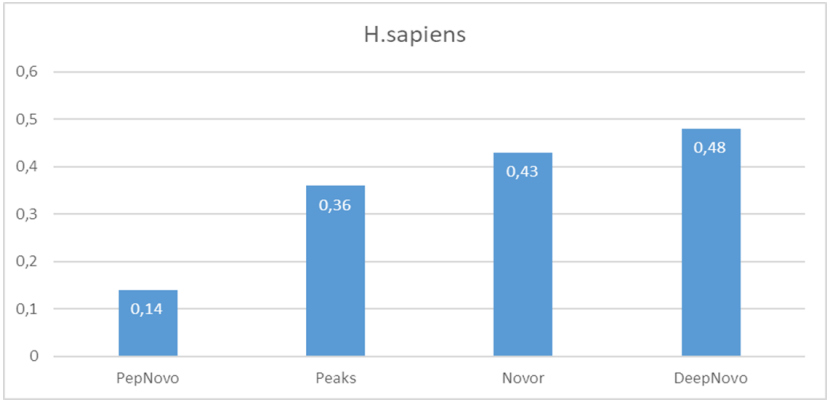


FIGURE 5.1 Comparison of de novo sequencing methods PepNovo, Novor, PEAKS and DeepNovo by recall precision for human sample data set.

[1]High precision and high recall is represented by a high area under the precision-recall curve.

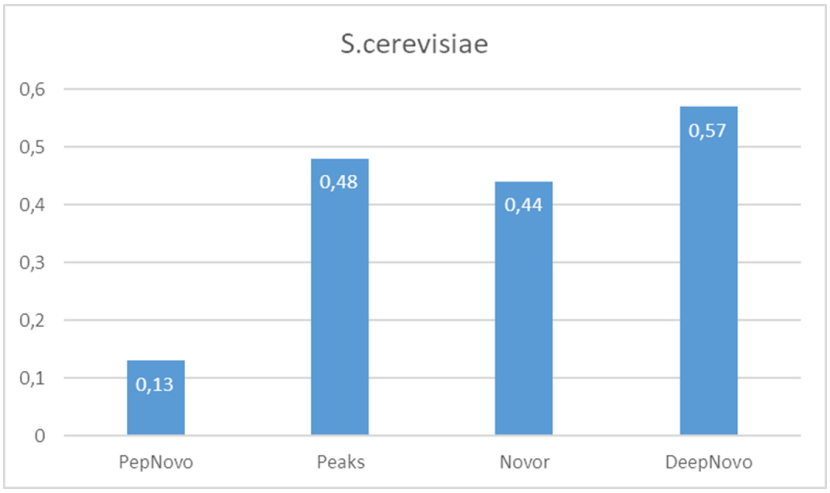


FIGURE 5.2 Comparison of de novo sequencing methods PepNovo, Novor, PEAKS and DeepNovo by recall precision for yeast sample data set.

To verify, that the improvements to the implementation of the DeepNovo model did not influence the performance of the model, DeepNovo was tested with the same fixed size data set from the Saccharomyces cerevisiae proteome presented by Ngoc Hieu Tran et al. in 2017 [12]. The testing data set contains 1812 mass spectra. As seen in table 5.4, the adapted implementation reaches identical quality measures as the original implementation, proving both implementations produce identical results. Therefore, we can assume that the adapted implementation produces high quality results compared to other de novo sequencing implementations.

DeepNovo			
	AA recall	peptide recall	length recall
adapted implementation	0.7433	0.6165	0.8183
Ngoc Hieu Tran et al.	0.7433	0.6165	0.8183

TABLE 5.4 DeepNovo prediction performance

The different deep learning models were trained for 50 epochs with the PRIDE data set and the criteria were measured (table 5.5). In my measurement the DeepNovo model reached an amino acid recall of 26.9%, a peptide recall of 24.3% and a length recall of 35.6%. The sequence to sequence model reached an amino acid recall of 14%, a peptide recall of 0.2 % and a length recall of 0.3%. The output predicted by the sequence to sequence model and convolutional sequence to sequence model was the same sequence regardless of the mass spectrum used as input. The convolutional sequence to sequence model failed in every prediction, which is reflected by the amino acid recall, peptide and length recall of 0 %. DeepNovo showed the best results for each measurement as clearly shown in the visualized results in figure 5.3.

Deep learning models			
Model	AA recall	peptide recall	length recall
DeepNovo	0.269	0.243	0.356
Sequence to sequence	0.14	0.002	0.003
CNN sequence to sequence	0	0	0

TABLE 5.5 Deep learning model prediction performance

Deep learning models	
Model	Index
DeepNovo	1
Sequence to sequence	2
CNN sequence to sequence	3

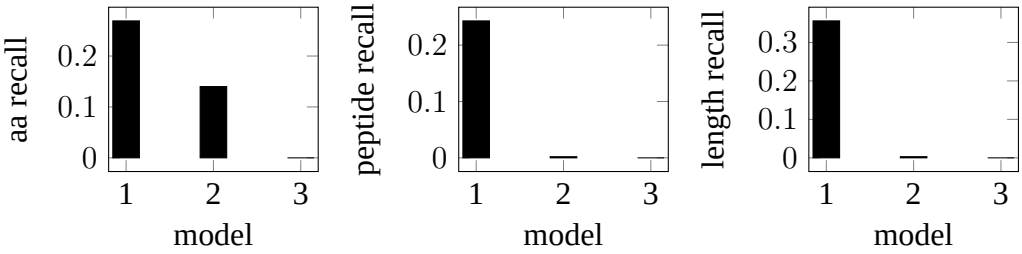


FIGURE 5.3 Deep learning model prediction performance

The mean prediction time for a single mass spectrum was measured for each deep learning model. The goal of this evaluation was to reach a minimal prediction time. The measurement was performed with the testing data set. The results (table 5.6) show that the DeepNovo model performed with an average prediction time of 47 milliseconds for a single mass spectrum. The sequence to sequence model performed with 199 milliseconds prediction time for a single mass spectrum. The convolutional sequence to sequence model reaches a prediction result for a single mass spectrum after 242 milliseconds.

As DeepNovo showed the best results in the evaluation of the deep learning models the quality of the predictions was also measured for a different number of training epochs. The DeepNovo model was trained for 5, 14, 27 and 50 epochs and amino acid recall, peptide recall and length recall were measured. The results (table 5.7, figure 5.4) show that the increase in training time only results in marginally better prediction results. Between a training time of five epochs and a training time of 50 epochs the amino acid recall only increases by 0.032 (13%), the peptide recall increased by 0.04 (19%) and the length recall increased by 0.024 (7.3%).

Deep learning models	
Model	mean prediction time (ms)
DeepNovo	47
Sequence to sequence	199
CNN sequence to sequence	242

TABLE 5.6 Deep learning model prediction time

DeepNovo training			
epochs	AA recall	peptide recall	length recall
5	0.237	0.203	0.332
14	0.231	0.194	0.317
36	0.265	0.238	0.353
50	0.269	0.243	0.356

TABLE 5.7 DeepNovo prediction performance for different epochs

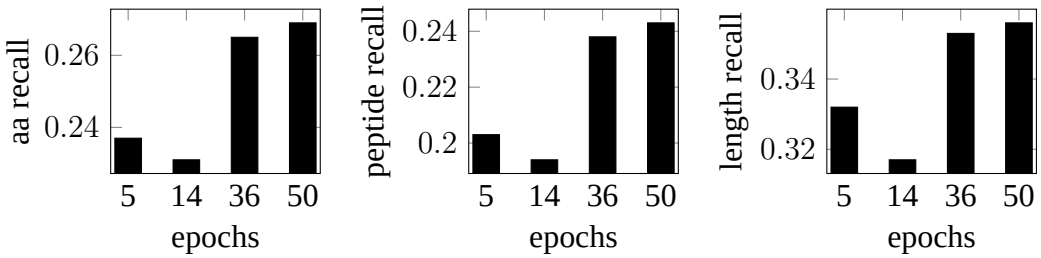


FIGURE 5.4 DeepNovo prediction performance for different epochs

5.2.4 Discussion

In the verification of the prediction performance of DeepNovo the model reached the same results as presented by Ngoc Hieu Tran et al. in 2017 [12]. This indicates that the adapted model implementation and the model utilized by Ngoc Hieu Tran et al. are equivalent.

In comparison between the different deep learning models DeepNovo showed the best results in each category. The sequence to sequence and the convolutional sequence to sequence model performed significantly worse during testing than the DeepNovo model. As both networks return the same sequence regardless of input the models are not usable for the prediction of amino acid sequences without identifying the cause of these bad results. Causes for the bad results of a neural network could be that the

network is overfitting, has reached a local minimum, is extrapolating or does not have enough neurons to fit the data. Overfitting can be identified during training by a minimal training error and a high validation error as the neural network becomes too fitted to the training data. If the error of validation, training and testing are large then the neural network might not contain enough neurons to fit the data. When small training and validation error and a large testing error can be observed the network might be extrapolating [31, p. 22-27].

The sequence to sequence and the convolutional sequence to sequence model could be improved by tuning the neural network and binning parameters. While the learning rate is adapted by the RMSProp optimizer the size of the neural network layers, the binning size, epochs and the neural network composition are changed by the user directly. Changes in the size of the neural network, binning parameters and structure of the neural network require the time consuming retraining of the neural network. As both sequence to sequence models only create the same output for each input the models likely found a local minimum that is represented by the output sequence. However it can also be assumed that a simple deep learning approach as exercised by the sequence to sequence and convolutional sequence to sequence model to the problem of de novo peptide sequencing does not contain the necessary capacity to learn the features of peptides encoded in mass spectra.

The complex DeepNovo system that combines multiple CNNs and RNNs with a dynamic programming algorithm for the knapsack problem has shown that it delivers the capability to learn these features even with minimal tuning of the parameters. The prediction results of DeepNovo for the PRIDE data set are not as good as the results for data set from the Saccharomyces cerevisiae proteome with a difference between the peptide recall rates of 0.414. The loss of peptide recall can be explained by the different data sources. The PRIDE data set contains 80749 real-world mass spectra of peptides from the proteome of different species and different mass spectrometry instruments. The data set utilized in [12] contains only 1812 peptides from the proteome of a single species captured by a single instrument. The generalization of the deep learning approach to mass spectrometry data from many different sources, such as contained in the PRIDE database, would constitute a major breakthrough in proteomics and warrants future research. Tuning the parameters of DeepNovo model like binning size, neural network layout and training variables could also improve the performance of DeepNovo on real-world data.

DeepNovo also shows the best result in reducing the time necessary for a prediction of a spectrum: the prediction for a single spectrum takes 47 milliseconds. The prediction by the sequence to sequence model takes 199 milliseconds and the prediction by the convolutional sequence to sequence model takes 242 milliseconds. Exemplary database search times are taken from the work of Hsieh et al. [38]. The database

searches on a four core AMD 64-bit 2.5Ghz processor took on average 74.94 minutes for 14,774 mass spectra. As a consequence, the average prediction time for a single mass spectrum is 304 milliseconds. The average database searches are dependent on different components like hardware, search algorithm, search parameters and the protein database. Furthermore, the implemented deep learning models utilize GPUs for their calculations, while database searches are algorithms that calculate their results on CPUs. While the values reported here show that the predictions by the deep learning solutions are faster than database searches, a fair comparison is not possible for the aforementioned reasons.

The difference in prediction time between the different deep learning models could have multiple causes. The sequence to sequence model and the convolutional sequence to sequence model are implemented with the high-level framework Keras using Tensorflow as back-end, while DeepNovo is directly implemented in Tensorflow. The data-retrieval and binning for the sequence to sequence model and the convolutional sequence to sequence model is implemented with Python, which is known to have a lower performance than the compiled Cython[2] modules for DeepNovo.

The training and evaluation of the DeepNovo model for different training time intervals revealed that even a limited training time of 5 epochs results in comparable results to a training time of 50 epochs. The prediction results of the DeepNovo model increased slightly with the increase in training time. The amino acid recall increased by 0.032, the peptide recall increased by 0.04 and the length recall increased by 0.024.

It can be concluded that the deep learning approaches for de novo peptide sequencing require future research, but already outperform other de novo sequencing algorithms, while getting closer to protein database search in terms of accuracy.

[2]Cython translates Python code into C, which results in a speedup.

CHAPTER 6

Conclusion & Future Work

In this thesis the DeepPSM tool for de novo peptide sequencing was proposed and evaluated with three different deep learning approaches. I developed the DeepPSM system, which contains a data retrieval component for automated data set construction from real world data provided by the public proteomics database PRIDE and a deep learning model for the prediction of peptide identifications in mass spectra. Three deep learning approaches were developed and evaluated: a sequence to sequence model, a convolutional sequence to sequence model and the complex DeepNovo model. Evaluation results show that the DeepNovo model outperforms other tested models and conventional de novo sequencing algorithms. The DeepPSM system is able to automatically generate the training, validation and testing data set from the real-world data in the file based PRIDE archive. The time required for the predictions of peptide sequences by a trained deep learning model depend on the implementation and the available hardware. Database search implementations are currently more mature than deep learning solutions and still outperform them in terms of peptide recall, but only for known peptide sequences. Ultimately, it can be concluded, that a deep learning approach for de novo peptide sequencing with real-world data as background knowledge, shows promising results for future research. Furthermore, future optimization of deep learning approaches for de novo peptide sequencing can be expected to further speed up the prediction and training performance.

The evaluation is able to achieve the goal of this thesis. However the evaluation has shown that the DeepPSM system is not ready for real world application. Continued research could lead to a deep-learning solution capable of identifying peptides in mass spectra more reliably, which would make a real world application of the DeepPSM system as the main identification solution in proteomics research feasible.

Future Work

In addition to the deep learning approaches in this thesis, there are multiple ways of approaching the problem of de novo peptide sequencing. In the following, a number

of points that have not been addressed in this work are explored.

Training deep learning models for specific species: The prediction performance of the DeepNovo model could be improved by creating a library of trained models for the different species present in the PRIDE archive. A model would be trained for each species where enough proteomics data is available for training. The peptides in a sample would be identified by the deep learning model that is trained for the identification of peptides for the species of the sample. If this approach results in a better peptide recall it may be the topic of future research.

Attention mechanism for sequence to sequence neural network: The sequence to sequence models could be increased in size and also implemented with an attention mechanism to evaluate if the recall rates could be improved.

Mass spectra feature extraction by compression: An encoder-decoder RNN could be designed to compress the mass spectrum and utilize the compressed expression instead of the binned mass spectrum for training.

Inclusion of other proteomics databases: The data retrieval component could be extended by adding more proteomics databases to the retrieval process and also include the theoretical mass spectra and sequences that can be generated by studying the genome of an organism.

Automated parametrization component: The DeepPSM system could be improved by creating an automated parametrization system based on the data provided by the PRIDE archive. In fixed size data sets the neural network is tuned by the researcher to specifically fit the utilized data set. A system like DeepPSM that automatically trains a neural network with real-world data could be improved by a component that configures the neural network based on data set statistics and selects the pretrained model that is the best fit to the input data set.

Appendix

Retrieved PRIDE projects		
Project	Species	Instrument
PXD009295	Homo sapiens (Human)	Orbitrap Fusion Lumos
PXD009003	Enterococcus faecalis (Streptococcus faecalis)	LTQ Orbitrap, TripleTOF 5600
PXD009096	environmental samples, uncultured environmental isolates, environmental samples <Archaea>, Eukaryota (eucaryotes)	Q Exactive
PXD009919	Homo sapiens (Human), Mus musculus (Mouse)	Orbitrap Fusion Lumos
PXD009480	Serpula lacrymans, Postia placenta, Gloeophyllum trabeum (Brown rot fungus), Trametes versicolor	LTQ Orbitrap
PXD004004	Mus musculus (Mouse)	Q Exactive
PXD009950	Homo sapiens (Human)	Q Exactive
PXD009079	Homo sapiens (Human)	LTQ Orbitrap Elite
PXD006938	Homo sapiens (Human)	LTQ Orbitrap
PXD006928	Homo sapiens (Human)	6340 Ion Trap LC/MS
PXD009705	Pseudomonas aeruginosa	Q Exactive
PXD008592	Peptoclostridium difficile (strain R20291) (Clostridium difficile)	Synapt MS, Q Exactive
PXD007601	Homo sapiens (Human)	LTQ Orbitrap

TABLE 1 Retrieved PRIDE projects 1-13

Retrieved PRIDE projects		
PXD006615	Homo sapiens (Human)	Orbitrap Fusion
PXD006618	Homo sapiens (Human)	Orbitrap Fusion
PXD002488	Mus musculus (Mouse)	LTQ Orbitrap Velos
PXD005434	Mus musculus (Mouse)	LTQ Orbitrap Velos
PXD005698	Homo sapiens (Human)	LTQ Orbitrap Elite
PXD002147	Zymoseptoria tritici (strain CBS 115943 / IPO323) (Speckled leaf blotch fungus) (Septoria tritici)	Q Exactive
PXD003129	Bursaphelenchus xylophilus (Pinewood nematode worm) (Aphelenchoides xylophilus)	TripleTOF 5600
PXD005290	Salmonella enterica subsp. enterica serovar Untypable str. 83332, Mycobacterium tuberculosis H37Rv	LTQ
PXD004453	Equus caballus (Horse)	Q Exactive
PXD004164	Mus musculus (Mouse)	LTQ Orbitrap Elite
PXD008786	Mycobacterium tuberculosis K	Orbitrap Fusion ETD
PXD009713	Mycoplasma sp. enrichment culture	Q Exactive
PXD009387	Mycoplasma sp. enrichment culture	Q Exactive
PXD009483	Rattus norvegicus (Rat)	LTQ Orbitrap
PXD006466	Homo sapiens (Human)	LTQ Orbitrap Elite
PXD008844	Mus musculus (Mouse)	Orbitrap Fusion Lumos
PXD009810	Cucurbita maxima	LTQ Orbitrap Velos

TABLE 2 Retrieved PRIDE projects 14-30

Retrieved PRIDE projects		
PXD008667	unclassified Bacteria	Q Exactive
PXD009738	Homo sapiens (Human)	Orbitrap Fusion Lumos
PXD009194	Sus scrofa domesticus (domestic pig)	Synapt MS
PXD009948	Homo sapiens (Human)	Orbitrap Fusion, Q Exactive HF
PXD005923	Opsanus beta	LTQ Orbitrap Velos
PXD009549	Mycobacterium tuberculosis	LTQ Orbitrap Velos
PXD008263	Bradyrhizobium japonicum USDA 110	Q Exactive
PXD008773	Homo sapiens (Human)	TripleTOF 5600
PXD006205	Bacillus cereus (strain ATCC 14579 / DSM 31)	LTQ Orbitrap XL
PXD006169	Bacillus cereus (strain ATCC 14579 / DSM 31)	LTQ Orbitrap XL

TABLE 3 Retrieved PRIDE projects 31-40

List of abbreviations

API Application Programming Interface

CNN Convolutional Neural Network

DNA Deoxyribonucleic acid

DNN Deep Neural Network

FWHM Full Width at Half Maximum

GPU Graphics Processing Unit

HPO Human Proteome Organization

JSON JavaScript Object Notation

MGF Mascot Generic Format

PRIDE Proteomics Identifications

PSI Proteomics Standards Initiative

LSTM Long Short Term Memory

RNA Ribonucleic acid

RNN Recurrent Neural Network

TB Terabyte

XML Extensible Markup Language

List of Figures

2.1 Mass spectrum of the peptide ”DDYVEK” retrieved from the PRIDE archive. 6
2.2 Feed-forward neural network with connected input layer, hidden layer and output layer composed of simple artificial neurons.[1] 9
2.3 Sigmoid Function Plot 10
2.4 Typical CNN structure containing convolutional and sampling layers.[2] 12
2.5 Unfolded recurrent artificial neuron over its time steps.[3] 12
2.6 LSTM cell containing *tanh* and *sigmoid* neurons as gates.[4] 13

3.1 Simplified overview of the DeepPSM system. 16
3.2 Overview of a PRIDE project. 17
3.3 Binning of a mass spectrum. 17
3.4 Visualized sequence to sequence model architecture in DeepPSM. . . 20
3.5 Rectified Linear Unit Plot 21
3.6 Overview of DeepNovo 22

4.1 *SpectrumIdentificationItem* with enclosed *peptide_ref* from a mzIdentML file. 26
4.2 *Peptide* element with *PeptideSequence* from a mzIdentML file. 26
4.3 Overview of DeepPSM with utilized modules and versions. 29

5.1 Comparison of de novo sequencing methods PepNovo, Novor, PEAKS and DeepNovo by recall precision for human sample data set. 34
5.2 Comparison of de novo sequencing methods PepNovo, Novor, PEAKS and DeepNovo by recall precision for yeast sample data set. 35
5.3 Deep learning model prediction performance 36
5.4 DeepNovo prediction performance for different epochs 37

Bibliography

[1] O. Kirk, T. V. Borchert, and C. C. Fuglsang, “Industrial enzyme applications,” *Current Opinion in Biotechnology*, vol. 13, no. 4, pp. 345 – 351, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0958166902003282

[2] P. Kuppusamy, N. Govindan, M. M. Yusoff, and S. J. Ichwan, “Proteins are potent biomarkers to detect colon cancer progression,” *Saudi Journal of Biological Sciences*, vol. 24, no. 6, pp. 1212 – 1221, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319562X14001247

[3] A. L. Brass, D. M. Dykxhoorn, Y. Benita, N. Yan, A. Engelman, R. J. Xavier, J. Lieberman, and S. J. Elledge, “Identification of Host Proteins Required for HIV Infection Through a Functional Genomic Screen,” *Science*, vol. 319, p. 921, Feb. 2008.

[4] G. Poste, “Molecular diagnostics: A powerful new component of the healthcare value chain,” *Expert Review of Anticancer Therapy*, vol. 1, no. 1, pp. 1–5, 2001.

[5] J. Cox and M. Mann, “Quantitative, high-resolution proteomics for data-driven systems biology,” *Annual review of biochemistry*, vol. 80, pp. 273–99, 06 2011.

[6] X. Zhou, W. Gong, W. Fu, and F. Du, “Application of deep learning in object detection,” in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, May 2017, pp. 631–634.

[7] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *CoRR*, vol. abs/1609.03499, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[8] Google, “Android neural networks api,” retrieved August 6, 2018. [Online]. Available: https://developer.android.com/ndk/guides/neuralnetworks/

[9] Apple, “Core ml,” retrieved August 6, 2018. [Online]. Available: https://developer.apple.com/documentation/coreml

[10] Qualcomm, “Snapdragon artificial intelligence,” retrieved August 6, 2018. [Online]. Available: https://www.qualcomm.com/snapdragon/artificial-intelligence

[11] B. T. Chait, “CHEMISTRY: Mass spectrometry: Bottom-up or top-down?” *Science*, vol. 314, no. 5796, pp. 65–66, oct 2006. [Online]. Available: https://doi.org/10.1126/science.1133987

[12] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[13] D. Liebler, *Introduction to Proteomics: Tools for the New Biology*, ser. Introduction to Proteomics: Tools for the New Biology. Humana Press, 2001.

[14] A. Bensimon, A. Heck, and R. Aebersold, “Mass spectrometry–based proteomics and network biology,” *Annual review of biochemistry*, vol. 81, pp. 379–405, 03 2012.

[15] J. S. Cottrell, “Protein identification using ms/ms data,” *Journal of Proteomics*, vol. 74, no. 10, pp. 1842 – 1851, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874391911002053

[16] M. Bern, Y. Cai, and D. Goldberg, “Lookup peaks: a hybrid of de novo sequencing and database search for protein identification by tandem mass spectrometry,” *Analytical Chemistry*, vol. 79, no. 4, pp. 1393–1400, 2007, pMID: 17243770. [Online]. Available: https://doi.org/10.1021/ac0617013

[17] M. T. Olson, J. A. Epstein, and A. L. Yergey, “Chapter 9 - de novo sequencing of peptides,” in *Medical Applications of Mass Spectrometry*, K. Vékey, A. Telekes, and A. Vertes, Eds. Amsterdam: Elsevier, 2008, pp. 195 – 201. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444519801500112

[18] T. Muth, E. Rapp, F. S. Berven, H. Barsnes, and M. Vaudel, *Tandem Mass Spectrum Sequencing: An Alternative to Database Search Engines in Shotgun Proteomics*. Cham: Springer International Publishing, 2016, pp. 217–226. [Online]. Available: https://doi.org/10.1007/978-3-319-41448-5_10

[19] J. Vizcaíno, A. Csordas, N. del Toro, J. A. Dianes, J. Griss, I. Lavidas, G. Mayer, Y. Perez-Riverol, F. Reisinger, T. Ternent, Q.-W. Xu, R. Wang,

and H. Hermjakob, “2016 update of the pride database and its related tools,” *Nucleic Acids Research*, vol. 44, no. 22, p. 11033, 2016. [Online]. Available: http://dx.doi.org/10.1093/nar/gkw880

[20] J. A. Vizcaíno, R. Côté, F. Reisinger, H. Barsnes, J. M. Foster, J. Rameseder, H. Hermjakob, and L. Martens, “The proteomics identifications database: 2010 update,” *Nucleic Acids Research*, vol. 38, no. suppl_1, pp. D736–D742, 2010. [Online]. Available: http://dx.doi.org/10.1093/nar/gkp964

[21] E. W. Deutsch, “File formats commonly used in mass spectrometry proteomics,” *Molecular cellular proteomics : MCP*, vol. 11, no. 12, p. 1612—1621, December 2012. [Online]. Available: http://europepmc.org/articles/PMC3518119

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[23] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, and P. Held, *Computational Intelligence: A Methodological Introduction*. Springer Publishing Company, Incorporated, 2013.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[25] D. Claudiu Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, “Flexible, high performance convolutional neural networks for image classification.” pp. 1237–1242, 07 2011.

[26] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, May 2009.

[27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[28] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[29] F. Chollet *et al.*, “Keras,” 2015. [Online]. Available: https://keras.io

[30] E. W. Deutsch, A. Csordas, Z. Sun, A. Jarnuczak, Y. Perez-Riverol, T. Ternent, D. S. Campbell, M. Bernal-Llinares, S. Okuda, S. Kawano, R. L. Moritz, J. J. Carver, M. Wang, Y. Ishihama, N. Bandeira, H. Hermjakob, and J. A. Vizcaíno, “The proteomexchange consortium in 2017: supporting the cultural change in proteomics public data deposition,” *Nucleic Acids Research*, vol. 45, no. D1, pp. D1100–D1106, 2017. [Online]. Available: http://dx.doi.org/10.1093/nar/gkw936

[31] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*, 2nd ed. USA: Martin Hagan, 2014.

[32] G. H. with Nitish Srivastava and K. Swersky, “Lecture 6a: Overview of mini-batch gradient descent.” [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, and Y. Bengio, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, 1929.

[35] N. H. Tran, X. Zhang, L. Xin, B. Shan, and M. Li, “De novo peptide sequencing by deep learning,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 31, pp. 8247–8252, 2017. [Online]. Available: http://www.pnas.org/content/114/31/8247

[36] N. H. Tran, Z. Levine, L. Xin, B. Shan, and M. Li, “Protein identification with deep learning: from abc to xyz,” *CoRR*, vol. abs/1710.02765, 2017. [Online]. Available: http://arxiv.org/abs/1710.02765

[37] J. A. Vizcaino, G. Mayer, S. R. Perkins, H. Barnsnes, M. Vaudel, Y. Perez-Riverol, T. Ternent, J. Uszkoreit, M. Eisenacher, L. Fischer, J. Rappsilber, E. Netz, M. Walzer, O. Kohlbacher, A. Leitner, R. J. Chalkley, F. Ghali, S. Martínez-Bartolomé, E. W. Deutsch, and A. R. Jones, “The mzidentml data standard version 1.2, supporting advances in proteome informatics,” *Mol. Cell. Prot.*, vol. 16, no. 7, pp. 1275–1285, 2017.

[38] E. J. Hsieh, M. R. Hoopmann, B. MacLean, and M. J. MacCoss, “Comparison of database search strategies for high precursor mass accuracy ms/ms data,” *Journal of proteome research*, vol. 9, no. 2, p. 1138—1143, February 2010. [Online]. Available: http://europepmc.org/articles/PMC2818881

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.
Magdeburg, den *17. August 2018*