DEGREE PROJECT IN ENGINEERING PHYSICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Prediction of Ranking of Chromatographic Retention Times using a Convolutional Network

KRUCZEK DANIEL

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Prediction of Ranking of Chromatographic Retention Times using a Convolutional Network

Daniel Philippe Kruczek
kruczek@kth.se

Master's thesis in Machine Learning (30 ECTS credits)
as part of M. Sc. degree in Engineering Physics

Supervisor was Lukas Käll
Supervisor at CSC was Jens Lagergren
Examiner was Erik Fransén

School of Electrical Engineering and Computer Science
Royal Institute of Technology
Sweden
24 Mar, 2018

Abstract

In shotgun proteomics, the liquid chromatography step is used to separate peptides in order to analyze as few as possible at the same time in the mass spectrometry step. Each peptide has a retention time, that is how long it takes to pass through the chromatography column. Prediction of the retention time can be used to gain increased identification of peptides or in order to create targeted proteomics experiments. Using machine learning methods such as support vector machines has given a high prediction accuracy, but such methods require known features that the retention time depends on. In this thesis we let a convolutional network, learn to rank the retention times instead of predicting the retention times themselves. We also tested how the prediction accuracy depends on the size of the training set. We found that pairwise ranking of peptides outperforms pointwise ranking and that adding more training data increased accuracy until the end without an increase in training time. This implies that accuracy can be further increased by training on even greater training sets.

Sammanfattning

Inom shotgun proteomics används vätskekromatigrafisteget för att separera peptiderna så att så få som möjligt kan analyseras i masspektrometristeget. Varje peptid har en retentionstid, vilket är tiden som det tar för peptiden att komma igenom vätskekromatografikolumnen. Att förutsäga retentionstiden kan användas för att identifiera fler peptider eller för att skapa riktade experiment. Genom att använda maskininlärningsmetoder som stödvektormaskiner har hög förutsägningsnoggrannhet uppnåtts, men sådana metoder kräver att det är känt vad retentionstiden beror på. I denna rapport låter vi ett faltningsnätverk lära sig att rangordna retentionstiden hos ett par peptider istället för att förutsäga retentionstiden för den enskilda peptiden. Vi har även testat hur förutsägningsnoggrannheten beror på storleken av träningssetet. Vi fann att träning med parvis rangordning av peptider ger en högre noggrannhet än rankning med hjälp av förutsagd retentionstid och att mer träningsdata gav högre noggrannhet utan att förlänga träningstiden. Detta implicerar att noggrannheten kan öka ytterligare med hjälp av ännu större träningsset.

Contents

1 Introduction

1.1 Background

Shotgun Proteomics is a popular method for protein identification. First the proteins are digested into peptides. Those peptides are then separated through Liquid Chromatography. Possible peptides are then identified by use of mass spectrometry and lastly the information about the peptide content is used in order to deduce which proteins were present in the original sample. In the Liquid Chromatography step each peptide takes an approximate time in order to go through a column depending on its chemical and structural properties. This time is called the peptide's Retention Time. Knowing the peptide's Retention Time has many uses, which will be described later in this thesis. Because of the usefulness of Retention Time prediction there have been many attempts to predict the Retention Times of peptides through various approaches. One of the currently most successful approaches uses the machine learning method Support Vector Regression. This method requires that we know what features of the peptides the Retention Time depends on, which is a hard problem in itself. As an example some of the 60 features chosen for ELUDE did not have a big impact on the retention time prediction [12]. An alternative solution is to choose a machine learning algorithm that learns useful features directly from a representation of the peptide and uses these features to decide the retention time. This has been tested by S. Valjamets that found Convolutional Neural Networks predict retention time better than ELUDE when trained on a sufficiently large data set [20]. An alternative to predicting the peptides' retention times is to learn to sort them in the order of their retention times. There are some results suggesting that pairwise ranking outperforms pointwise ranking [10]. For our problem this suggests that it is easier to sort a list of peptides by their retention time by having pairs of peptides as input and output than by predicting just the retention time of the peptides. Training on ranked data rather than predicting retention time also has the benefit of both being able to train on combined data sets with different run times and be used for experiments independent of their run time.

1.2 Objective

Peptide retention time tools such as ELUDE [12] have achieved a high accuracy predicting retention times. Recently even higher accuracies have been achieved using Convolutional Neural Networks, as they benefit from large data sets and can learn important features from a 2D representation of the peptide [20]. In this thesis we will examine how well a Convolutional Neural Network can learn the problem of ranking peptides. Additionally we will examine how our ranking prediction model can learn from training on multiple data sets from different experimental runs and if we can predict the ranking for data sets with other experimental setups. Lastly we examine if a pairwise ranking of peptides outperforms pointwise ranking.

2 Relevant Theory

2.1 Shotgun proteomics overview

Shotgun Proteomics is a technique used to identify proteins from a sample with a high number of proteins. As handling peptides instead of proteins gives several advantages for protein identification [13], the proteins are divided into peptides using an enzyme. In order to analyze as few peptides as possible at the same time, the peptides are separated using Liquid Chromatography. In the Liquid Chromatography step the peptides go through a column packed with porous silica beads and an aqueous solvent. The silica beads' surfaces are covered in alkyl chains to which the peptides bind along their way through the column. Another solvent is gradually added so that the concentration follows a, typically linear, gradient in the column. Each different peptide will stay at a specific concentration throughout the column and elude when the concentration at the exit is high enough. Because of this each peptide will elude the column at different times, which is referred to as the peptide's retention time. At the end of this column, the peptides are analyzed using mass spectrometry. In the mass spectrometry step the peptides are ionized and their mass-to-charge ratio is recorded along with the respective signal intensity in a so called MS1 spectrum. As the liquid chromatography step doesn't fully separate the peptides, they are further separated by performing another mass spectrometry step. In this step a mass-to-charge ratio range of varying size is chosen in order to be analyzed. By choosing a small range the number of peptide species analyzed can be minimized (Data Dependent Acquisition) and by choosing a larger range the number of peptides analyzed is increased, but gives a more complicated spectra (Data Independent Acquisition). The chosen peptides are now fragmented, which gives additional information about the original peptides by providing information about their fragments. This spectra is referred to as the MS2 spectra. From here the peptides and proteins have to be inferred using computational methods as many peptides can fit the given mass-to-charge ratio. The most common method is Database Searching where spectra are compared to either a database of theoretical spectra or a library of experimental spectra. Each comparison gives a score according to some scoring function and a list of highest scoring matches is given. Methods that do not make use of databases are collectively referred to as de novo sequencing. [14] [19]

2.2 Applications of retention time prediction

As the methods used to match spectra to peptides usually yield a list of candidate peptides that all match the given mass-to-charge ratios, additional information in form of retention time can be used in order to discriminate between candidates.

One simple implementation is to remove the candidates that differ too much between predicted and observed retention time. This was successfully implemented by Klammer et al. where the authors manually searched for an actual and predicted retention time difference cutoff that yielded the most true positives for peptide identifications among different initial false discovery rates [5].

Another way to use prediction of retention time is to better choose which peptides from the MS1 spectra should be fragmented. If a protein has already

been identified, retention time prediction can identify peptides from the already identified protein from the MS1 spectrum, and the peptide does not have to be further analyzed. This has been successfully implemented by Krokhin et al [7].

Retention time prediction has also been used in order to design better liquid chromatography setups. Moruz et al. discovered that the retention times predicted for peptides obtained from an in silico digest of the human proteome are not uniformly distributed when the second solvent is increased linearly. Predicting the retention time distribution can thus be used in order to obtain a more uniform retention time distribution and thus a better separation of peptides [11].

2.3 Retention Time prediction approaches

Retention time prediction is mainly done using four approaches, that is through look-up tables, index based methods, modeling and by use of machine learning. Look-up tables make use of previously noted retention times for some experimental setup. While they can't predict retention time for peptides not in the database, they have several other uses. Index based methods estimate the effect of each amino acid in the peptide on the retention time and predicts the retention time as the sum of all contributions. Modeling based approaches use information about the peptide's structure and chemical properties in order to model how a given peptide will elude through the chromatography column. [11]

Machine Learning approaches make use of retention time data from previous experiments in order to train some mathematical model. Once the model is sufficiently trained it can be used in order to predict the retention time or the order of the peptides that were not a part of the training set. The input to the model can be given in various forms, which may have an impact on how well the model is trained. The input can be a representation of the peptide in one or more dimensions, it's chemical properties or a mixture of those.

2.4 Deep Learning

In Machine Learning, a program can learn to solve a task such as classification or regression by itself, usually given some training data. The learning process is generally supervised or unsupervised. In unsupervised learning the data is not labeled and the model can thus not learn by comparing its solution to some correct answer. Still unsupervised learning can be successfully used in problems such as clustering. In supervised learning a model can be trained by comparing the output to some training data. In the retention time problem this output can be a retention time with a peptide as input or a ranking of two peptides with two peptides as input. The input data for training the model will most of the time have some incorrectly labeled data point and noise. As an example in the retention time problem all of the peptides used as training data are not correctly classified. The model thus has to generalize well enough so that the few erroneous examples will worsen the predictions as little as possible. But even a model trained on a true training set can give bad or incorrect predictions. Since a model will be trained on a subset of all possible data, it will receive a bias from that training set, risking to learn quirks from the training set instead of more general patterns helpful in prediction of unseen data. Generally a model that can represent a pattern with a high resolution can learn the general concepts needed to predict unseen data well, but will also learn the particularities of the

training set such as noise. Models that can only represent a low resolution of the pattern in the data won't learn peculiarities of the training set, but risk missing patterns that are important for generalizing to beyond the training set. This problem is known as the Bias-Variance dilemma.

A common problem in machine learning is that it is not always obvious what parameters that are important in a problem. In the previous retention time prediction work by Moruz et al. with Elude a support vector machine model with 60 features was trained. The features included retention coefficients for the different amino acids, length of the peptide, number of occurrences of each amino acid and chemical properties of the peptide. It was discovered that some of the features have very little influence on the retention time. To better predict the retention time using support vector machines the properties that dictate the retention time of a peptide would need to be known, which is a very difficult problem. However, many machine learning methods also learn the relevant features when training.

Deep Learning methods in machine learning consist of multiple nonlinear layers which lets the model represent the problem with a higher level of abstraction between each level. Deep learning methods also learn important features by themselves, bypassing the problem of choosing features present when using support vector machines for retention time prediction. An example of a deep learning method are convolutional neural networks, commonly used to classify images [8].

2.5 Convolutional Neural Networks

In Machine Learning a frequently used method is Artificial Neural Networks (ANN). An ANN is a network of interconnected nodes that receive and send signals. Each of the nodes has some activation function, that takes the sum of the incoming signals as an input and sends the function value as output. Each of the node connections also has a weight that is multiplied with a nodes output before becoming the next node's input. It is this weight that is what can be changed in the network and is where the learning happens. In the learning process the output of the network is compared to the correct answer and the value of a cost function is calculated. The cost function consists of a function of the error, but can also include regularization terms, that can punish behaviors such as too big weights. The regularization terms prevent overfitting. That is when our model is too complicated, which causes the model to give low errors on the training data, but does not generalize well on other data. When training the model, the cost function is minimized using gradient descent towards some minimum. A basic neural network consists of a number of layers, where the first layer is an input layer, the last layer is the output layer and one or more hidden layers in between. Each layer is fully connected to its neighboring layers.

Another type of neural network is the Convolutional Neural Network (CNN). The input of a CNN is an image, or in other words a matrix with values. CNNs are made out of two layers in addition to the fully connected layer from the basic neural network. These are the convolutional layer and the pooling layer. The convolutional layer consists of filters or convolutional kernels, which are small matrices. In the convolution step, these matrices are convoluted with the input matrix, which gives one matrix as output for each filter. If the input to the convolutional layer is a volume, as in for example an image with three colors

or the output of a previous convolutional layer, the results of the convolutions of each filter are added into one matrix. Pooling layers are used in order to downsample the output of a convolutional layer, which reduces the computation required for the subsequent layers. A common example is maxpooling where the layer is divided into equal rectangles. Each rectangle is then replaced with the largest value in that rectangle, giving a smaller matrix as output. After the convolutional and pooling layers one or more fully connected layers can be added.

2.6 Motivation for using Convolutional Neural Networks for predicting retention time

One of the currently best retention time predictors is ELUDE, which uses a Support Vector Machine with 60 features [12]. Unfortunately it is not well known what properties chromatographic retention time depends on, which makes it hard to choose good enough features. Convolutional neural networks are successful in classifying images, because the filters learn features that describe the objects in the image well. The motivation for using convolutional neural networks is therefore that the algorithm might learn useful features from a suitable representation of the peptide and outperform a support vector machine approach. With a suitable representation of the peptides the Convolutional Network can also learn from the order of the amino acids in the peptide sequence, as models that don't take this into account are currently outperformed by models that do [18]. A convolutional neural network model has been shown to outperform ELUDE when using a sufficiently large training set. An additional advantage of Convolutional Networks over Support Vector Regression is how well they handle large data sets. A Support Vector Regression model will quickly take too long time to train if the training data set becomes too large, while a Convolutional Network can simply train the same amount of time, but on more diverse data. The network can also be easily modified in order to learn to rank two peptides by only changing the input and output layers.

3 Related work

Prediction of chromatographic retention time has been attempted since about 1980 and have since then been greatly improved. Existing retention time prediction approaches can be divided into index based, modeling based, and machine learning based. None of the approaches is currently the best option in all cases and different methods do well on different datasets and with a different amount of data. A recent overview of existing retention time prediction tools, that also tests a few selected tools on different datasets, has been written by Tarasova et al. [18]

3.1 Prior retention time prediction approaches

Index based approaches have been one of the first attempts at predicting retention time. One of the first was made by Meek assigning amino acid coefficients that described the individual contribution to each amino acid. The score from each peptide was then added to a total score which predicted the retention

time [9]. A more successful approach called SSRCalc was presented by Krokhin et al., where in addition to individual amino acid scoring, sequence specific correction factors were introduced [6].

Modeling approaches aim to predict a peptide's retention time by using information about the peptide's structure and the chromatography settings. One such predictor is BioLCCC that predicts the retention time of a peptide by using a random walk taking into account entropy and energy for interactions within the chromatography column. The interaction parameters have to be obtained for a specific chromatography setup. [4]

Three machine learning methods has been successfully used for retention time prediction. Petritis et al. was first with using neural networks in 2003 [15]. The neural network was a single layer perceptron, originally using 20 input nodes, which was later changed to 1052. These nodes encoded the amino acid for each position as well as other peptide specific properties [16]. Several groups have also predicted retention time using support vector regression, the most successful tool being by Moruz et al. The SVR is based upon 60 features containing information about the peptide. Elude also has pretrained models, that can be used in order to lower the number of peptides required for training [12]. Recently Afkham et al. has tried predicting retention time using Gaussian Processes on the same peptide representation as Elude. The prediction power of this approach is similar to that of Elude, but provides a measurement of prediction uncertainty for each peptide [1].

4 Model Selection and Experiments

All models take as input two peptides of variable lengths, where each peptide is made of a combination of 20 different amino acids. The output of each model consists of two nodes, where the node with the largest output indicates that the corresponding peptide has the lowest retention time. The output is normalized to 1. The training data consists of two peptides with the output 1 for the node with the lower retention time and 0 for the node with the higher retention time. All models were created in Lasagne.

Training and test data

Single file input - The training and test data for the single file input experiments came from the same shotgun proteomics experiment on yeast, where 20000 of the peptides were chosen for the training set and 4000 of the peptides were chosen for the test set so that both sets did not have any overlapping peptides. Both the training and test set had a false discovery rate of 1%, which could affect the validation error. However there is a bias in the incorrectly chosen peptides, because they are likely to have similar mass to the correct peptides, as they are identified from the mass to charge ratio.

Multiple file input - The training and test data comes from 2201 experiments run on human tissue. The experiments were made on several tissues, with multiple runs on each tissue. In total the data contains 237210 unique peptides. However for each epoch only one file was used, so small files had to be excluded or training on a small number of peptides would cause overfitting.

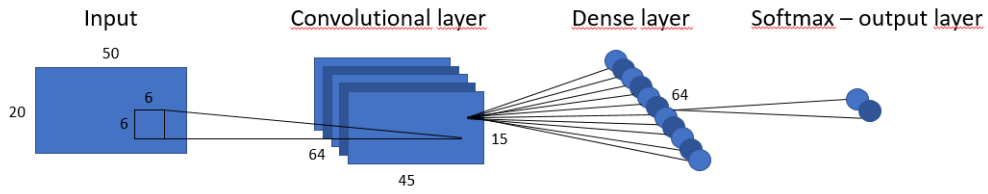

Figure 1: The basic network consists of 64 6×6 convolutional kernels followed by a single dense layer with 64 nodes and an output layer using the softmax function.

To prevent this all files with less than 3000 discovered peptides were excluded, which still provided 369 files containing 171977 unique peptides. The file containing the largest number of peptides (6887) was chosen to be the test data. Before each epoch the peptides that were found in both the training file and the test file were removed from the training file. The data had a false discovery rate of 0.1%. All runs had the same experimental setup. **Elude:** Elude was trained on the yeast training set.

Training

The training for each experiment was done for 1500 epochs, where for each epoch 20000 pairs of peptides from the training set was chosen randomly. The number of epochs was chosen based on that the test accuracy was still rising until then and the the training time was short enough to complete the training in one day.

Testing

The testing was done by randomly choosing 100000 peptide pairs from the testing dataset and for each pair checking if the model would predict the retention time order correctly. The same test set was used for all single file experiments and another test set for all the multiple file experiments. The peptide pairs were each time randomly selected from these respective sets. No cross-validation was performed. The accuracy of the model was determined as the percentage of correct predictions. As the accuracy varied considerably between epochs, the final accuracy was the average accuracy of the last 20 epochs.

4.1 Basic Convolutional Network

The network described below is the most basic network tested. All other models will use this network as a basis to make changes to and for comparison.

4.1.1 Input

In order to feed a Convolutional Neural Network (CNN) a peptide as input, the peptide needs to be expressed by a two dimensional representation. For this we chose a 20 times 50 size matrix where the horizontal position expressed the position of an amino acid in the peptide and the vertical position expressed the

$$
\begin{array}{c} \\ A \\ C \\ D \\ E \\ \\ Y \end{array}
\begin{array}{c}
\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & & 50 \end{array} \\
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0
\end{pmatrix}
\end{array}
$$

Figure 2: The peptide ACCAD expressed in the input representation. The vertical position expresses amino acid type and the horizontal position expresses the position of the amino acid in the peptide.

type. The input size of a CNN has to be constant, which is why the horizontal size was chosen so that all peptides could safely fit. The part of the matrix that is not used is padded with zeros.

4.1.2 Model architecture and parameters

As ranking requires two peptides to be fed into the network at the same time, they were first fed into two separate pipelines that contained the convolutional part of the network. The 64 convolutional kernels used in each pipeline had the dimensions 6×6 with stride 1. The output from both pipelines was fed into a single dense layer with 64 nodes. The ranking was finally decided with a final softmax function layer with two nodes.

The loss function used was nesterov momentum, with a learning rate of 0.015 and a momentum of 0.9. The loss function chosen was cross entropy over squared error as the cross entropy error is bigger while near a locally optimal solution [3]. The weight initialization used in the dense layers was GlorotUniform, as it is designed to make the intial values such that the propagated signal in a deep neural network will not become too large or too small. [2] The transfer function used was the rectifier function.

The parameters used were based on the Lasagne MNIST tutorial where Lasagne is used to discriminate on handwritten digits using convolutional neural networks. Stride, momentum, transfer function and weight initialization were kept the same as in the Lasagne tutorial. The initial kernel size was changed from 5×5 to 6×6 and the learning rate was changed from 0.01 to 0.015 after a manual search. The network layer sizes were chosen to be as large as possible within a reasonable time frame.

4.2 Convolutional kernels covering all amino acids

In our basic model quadratic convolutional kernels are used as is often done in image recognition. One of the advantages of using convolutional networks on images is that similar patterns can emerge in different parts of the image. While this can also happen in our case, different vertical position in the matrix representing the peptide means different amino acids are present, and the similarity is only in the alphabetical order of the amino acids. In order to get around this

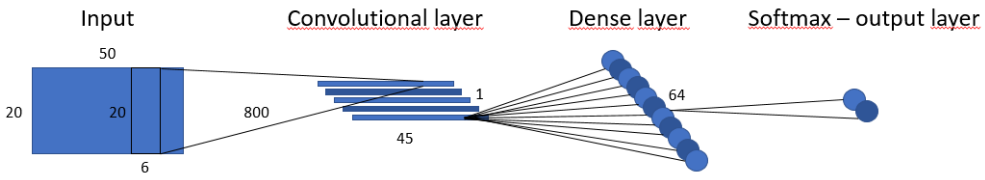

Figure 3: The network with extended convolutional kernels with additional kernels consists of 800 20×6 convolutional kernels followed by a single dense layer with 64 nodes and an output layer using the softmax function.

problem, 20×6 convolutional kernels were used instead of the 6×6 kernels in order to cover all amino acids at once. Since making the kernel size bigger also reduced the training time significantly, we also tested a increasing the number of kernels from 64 to 800 which increased the training time to match the basic model.

4.3 Extended peptide representation

One of the limitations of convolutional networks is the need of all input being the same size. It is possible that the ends of the peptide are more important in relation to the retention time than the middle part. The input to the basic network aligns the left side of the peptide to the left side of the matrix, which puts the right ending of the peptide in a different position depending on the length of the peptide. Therefore the input matrix was doubled in size, to a 20×100 matrix, with the peptide expressed twice, once aligned to the left side of the matrix and once to the right side, as seen in figure 4.

4.4 Additional dense layers

When adding an additional hidden layer to an artificial neural network one can always gain an equally good output by setting the weights to the identity, so that the next layer will get the same input as the previous layer. Additionally one could possibly modify the weights in a way that gives better results. However, adding too many hidden layers decreases performance. To determine the optimal number of hidden layers the basic model was extended with 5, 10 and 20 layers.

4.5 Testing on different datasets

In the previous experiments testing has been done on test sets with the same experimental setups as the training sets. It is thus not clear how well the predictions work on datasets with a different experimental setup. To test this a model trained on human data was tested on the yeast test set and a model trained on yeast data was tested on the human test set. The model chosen had 800 20×6 kernels with 10 dense layers after the convolutional layer.

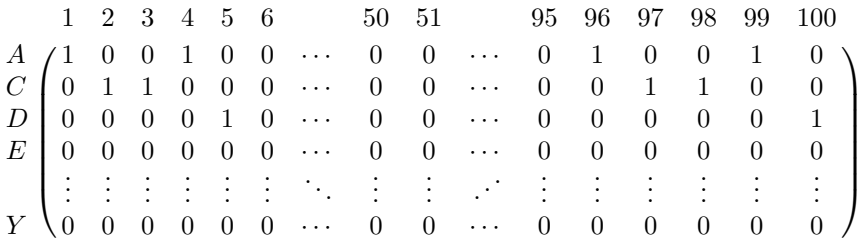

Figure 4: The peptide ACCAD expressed in the extended input representation. As in the normal representation the vertical position expresses the amino acid, however the amino acid is expressed twice. Once in its regular position and once aligned to the right side of the matrix.

4.6 Testing with a large dataset from multiple experiments

Generally the larger the training set is the better the model can be trained. However it can be hard to generate a large dataset from a shotgun proteomics experiment. We have thus tested if multiple datasets can be used in order to get better results. The training data was from multiple experimental runs on human tissue data. Only the files for the runs that found sufficiently many peptides were chosen. The model chosen had 800 20×6 kernels with 10 dense layers after the convolutional layer. To see how much impact a bigger dataset had on the results a subset of the files were excluded in order to see how the model prediction worked as a function of the number of unique peptides in the training set.

4.7 Ranking by regression

In order to compare how the model learns to predict ranking directly compared to ranking by regression we let the model predict the retention times of the test set and then evaluate how well it ranks pairs of peptides by comparing the predicted retention times.

The model compared used 800 20×6 kernels with 10 dense layers after the convolutional layer. The model was trained and tested on the yeast training set. The model was trained for 700 epochs, as the training error started to increase with further training.

Comparison with earlier studies

Neural networks have previously been used for predicting retention time by Petritis et al. [15] [16]. However these were single layer perceptrons that do not utilize relative input position the same way as convolutional neural networks. By using convolutional neural networks the network could utilize input where the relative positions of the amino acids were expressed, which has not been done before. In all previous studies the retention time of the peptides is predicted, however in this model we rank the retention times of two peptides.

5 Results

As results differ between epochs, all results presented will be an average of the 20 last epochs.

As can be seen in figure 5 the retention time ranking accuracy of the neural network is somewhat unstable, even after many epochs.

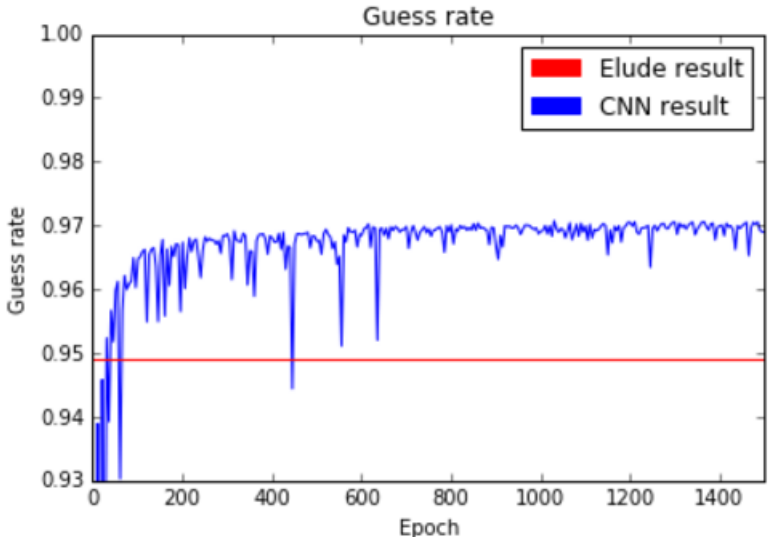

Figure 5: The test accuracy of the model prediction for a network with 800 20×6 convolutional kernels followed by a single dense layer with 64 nodes and an output layer using the softmax function compared to Elude's result on the same test set.

5.1 Accuracy of different models

All models were tested on the single file input test set in order to compare the accuracy with Elude, which is presented in figure 6. All except one tested network designs beat Elude in ranking prediction, although all except one were trained to rank directly instead of predicting regression like Elude. This network was trained for regression which was then used to predict the ranking and had a somewhat lower score than the same model but that predicted ranking directly, however the error was still significantly lower than Elude's. The highest scoring network had a 36% lower error rate than Elude on the same training and test set. The prediction accuracy of the neural network models was significantly increased when the convolutional kernels were extended to cover all of the amino acids simultaneously. Adding an extended peptide representation which presented the other end of the peptide in a fixed position lowered the accuracy of the model. The addition of additional dense layers after the convolutional layer did not bring a significant increase in performance.

Model setup	Accuracy [%]
Basic model	95.01
Extended convolutional kernels	96.61
Extended convolutional kernels with more kernels	96.75
Extended peptide representation	94.71
Basic model with 5 dense layers	95.23
Basic model with 10 dense layers	95.27
Basic model with 20 dense layers	95.17
Elude	94.90
Extended kernels with more kernels regression	96.35

Figure 6: Results for the the different models when trained on a single input file. The accuracy is the ratio of correct predictions.

5.2 Result of having training and test set from different experimental setup

In figure 7 we can see the comparison of training and testing with different data sets using the same model. The data comes from a smaller set from experiments on yeast peptides and a much larger set from human tissue experiments. In both cases when the training and test sets are from different experiments the accuracy is significantly lower than when both sets are from the same experiment, however the accuracy drop is larger when training on the smaller yeast training set. The increased accuracy from training on the larger human tissue set is so large that the test accuracy on the yeast test set is nearly identical independently of which data set the model was trained on.

Training and test data	Accuracy [%]
Human data with human test	97.27
Human data with yeast test	96.75
Yeast data with human test	95.56
Yeast data with yeast test	96.78

Figure 7: Accuracy of a network with 800 20×6 convolutional kernels followed by a single dense layer with 64 nodes and an output layer using the softmax function, given different different training and testing sets.

5.3 Accuracy depending on training set size

In figure 8 we can see the accuracy of the model, with extended convolutional kernels with additional kernels, depending on the number of peptides used during training. The same data is shown in figure 9, but instead of accuracy the error is shown as a function of the number of unique peptides.

Number of files	Number of unique peptides	Accuracy [%]
369	171977	97.39
333	166552	97.36
296	159601	97.31
259	152558	97.34
222	144975	97.27
185	135549	97.23
148	125880	97.30
111	115186	97.14
74	94874	97.27
56	83020	97.24
37	63558	97.07
30	56551	97.14
23	48413	96.91
15	36856	96.72
8	24282	96.47

Figure 8: The accuracy of the model prediction for a network with 800 20×6 convolutional kernels followed by a single dense layer with 64 nodes and an output layer using the softmax function, based on the number of peptides used in training.

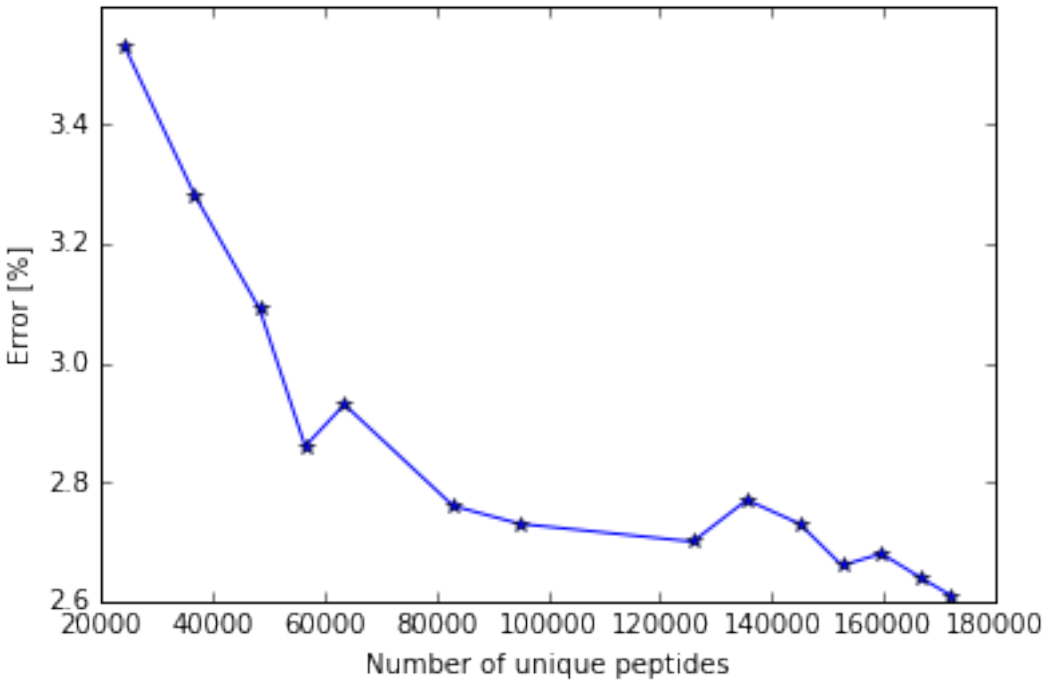

Figure 9: The ratio of wrong answers, when tested on 100000 peptide pairs from the test set, as a function of the number of unique peptides used in training.

6 Discussion

6.1 Comparison of model based on a Convolutional Neural Network with Elude

The model based on Convolutional Neural Networks achieved a significantly lower error rate ranking peptides' retention times, when trained on the same data set, than Elude. Additionally the Convolutional Neural Network model was able to learn from a much larger dataset with additional increase in accuracy without the training taking more time. With larger datasets becoming available this can be an increasingly important factor for the choice of machine learning algorithm for prediction.

6.2 Choice of hyperparameters

The choice of hyperparameters was based on the tutorial on a similar problem of the tool used to create the neural networks. Out of the numerical parameters only kernel size and learning rate were changed, while stride and momentum was left as is. The kernel size and learning rate were decided using a limited manual search. While a more extensive parameter search would help make the model more accurate, it is unfortunately out of the scope of this project. However such a search could be performed by using Bayesian Optimization. It's an automatic process in which the next model parameters are chosen from the parameter space based on the results from the previous trials. This optimization algorithm has been shown to reach or surpass human expert level optimization on convolutional neural networks. [17]

6.3 Choice of test data and test method

For the multiple file training the largest file was chosen because retention times for the same peptide from different runs can vary and thus a pair of peptides can only be formed using peptides from the same pair. It was assumed that the test data in this file was the most representative subset for the whole set of data, as it contained the most peptides. However this cannot be easily verified. The problem to find suitable features for different peptides is very non-trivial, which is why convolutional neural networks were chosen. Testing if the retention times of the peptides came from the same distribution could potentially give insight on if the test data was representative. However the speed between the experiments varies and it is not possible to scale the retention times, as the details of the experiments are unknown.

The testing for each model was performed using a single predetermined dataset for both the single and multi file experiments. Because of this it is unknown how much the choice of training data influenced the results of the experiments. In order to better determine the real accuracy of the models a k-fold cross-validation could be done. However, this would prolong the already long training time k-fold. As mixing data from different files for the multi-file experiment was inappropriate, a k-fold cross-validation could not be performed. Instead, the test could be done on k different files, which would not be as exhaustive as k-fold cross-validation, but still give a better insight than using a single test set.

6.4 Comparison of different model architectures

When using convolutional kernels covering all amino acids at once the accuracy was significantly increased. One of the reasons for this increase could be that a 6×6 kernel is not able to see the pattern between two neighboring amino acids if there are more than 4 amino acids between the in the amino acid alphabet and thus our representation. Extending the convolutional kernels to cover all the amino acids also significantly reduces the training time as it decreases the size of the convolution layer.

The extended peptide representation had a lower prediction accuracy, even though potentially providing additional information as both ends of the peptides were represented in a fixed position. This extension of the peptide representation increases the size of the model and the number of weights to be trained. This could lead to a model that is harder to train.

6.5 Performance increase from additional data and prediction on data from other experimental setups

An important aspect of choosing a neural network to predict retention times is that the amount of training data can be increased without increasing the training time. As seen in figure 9 more data continues to increase the performance of our model, suggesting that more data could further improve the performance.

As seen in figure 7 the same model trained on a larger data set on human tissue performed as good on the yeast test set as when the model was trained on yeast data with a different experimental setup. This suggests that a high accuracy can be achieved independently of the experimental setup of the training and test set.

6.6 Comparison of pointwise versus pairwise ranking

The ranking accuracy achieved by predicting and comparing retention times was slightly lower than the ranking accuracy achieved when the model learned to rank peptides by retention time directly. This result is in line with the results of the experiments by Melenikov et al. [10] where pairwise ranking outperformed pointwise ranking. This suggests that ranking peptides is a more suitable problem for retention ranking prediction.

6.7 Comparison of predicting retention time and ranking

Unlike the direct prediction of retention time, training a model to directly predict the ranking of peptides can be done using data sets with different run times without any modification. In the same way the results of the prediction can be used for any experiment independently on runtime. Additionally when using the predicted retention time it is assumed that the concentration of solvents changes linearly. However it has been shown that the peptides are not spread out uniformly throughout the liquid chromatography phase and it can be beneficial to change the solvent concentration depending on how many the relative concentration of different peptides [12]. In this experimental setting using retention time directly would be very complicated.

We have shown that Convolutional Neural Networks can benefit from very large data sets and it is likely that by using even more training data will further improve the results. We did also show that training on a large data set with a different experimental setup gave as good results as training on a smaller data set with data from the same experiment as the test data. This combined suggests that training data for a ranking model can be composed of data sets from different experiments, potentially greatly increasing the amount of training data and prediction accuracy.

6.8 Further improvements and testing

A popular method when using Convolutional Neural Networks for image recognition is to pretrain the model on a large data set as many patterns learned by the convolutional kernels are found frequently in many types of images. The importance of an amino acid pattern learned by a convolutional kernel should not alter significantly based on experimental setup. This, together with the fact that neural networks can benefit from training on large data sets without increased training time suggests that the model can be successfully pretrained on a large data set and can then be retrained on only a small data set for good results.

The field of machine learning is rapidly advancing as can be seen on the results of yearly image recognition competitions. Many of the various algorithms can be used on our peptide representation. As our model is very simple compared to the currently used in image recognition and already a few years old the performance could be greatly increased with new algorithms.

6.9 Ethical Considerations

This project uses existing methods in order to solve a new problem. Thus all potential advancement is in regards to the studied problem and not the method itself. All uses for retention time prediction have the goal of increasing the number of identified peptides and thus identified proteins from a sample. Improving shotgun proteomics in general can help efforts in regards to curing or preventing diseases, understanding of the human body and the biochemistry of other organisms. However such advances could also potentially be used to cause harm to humans more efficiently.

The data used in this project originated from both a yeast sample and multiple samples of human tissue. It is likely, but not known, whether the human samples were obtained with consent, and full understanding, of the person of whom the tissue samples were taken from.

7 References

[1] H. M. Afkham, X. Qiu, L. Käll, et al., *Uncertainty estimation of predictions of peptides' chromatographic retention times in shotgun proteomics*, Bioinformatics, (2016), p. btw619.

[2] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks.*, in Aistats, vol. 9, 2010, pp. 249–256.

[3] P. Golik, P. Doetsch, and H. Ney, *Cross-entropy vs. squared error training: a theoretical and experimental comparison.*, in INTERSPEECH, 2013, pp. 1756–1760.

[4] A. V. Gorshkov, I. A. Tarasova, V. V. Evreinov, M. M. Savitski, M. L. Nielsen, R. A. Zubarev, and M. V. Gorshkov, *Liquid chromatography at critical conditions: comprehensive approach to sequence-dependent retention time prediction*, Analytical chemistry, 78 (2006), pp. 7770–7777.

[5] A. A. Klammer, X. Yi, M. J. MacCoss, and W. S. Noble, *Improving tandem mass spectrum identification using peptide retention time prediction across diverse chromatography conditions*, Analytical Chemistry, 79 (2007), pp. 6111–6118.

[6] O. V. Krokhin, *Sequence-specific retention calculator. algorithm for peptide retention prediction in ion-pair rp-hplc: application to 300-and 100-å pore size c18 sorbents*, Analytical chemistry, 78 (2006), pp. 7785–7795.

[7] O. V. Krokhin, S. Ying, J. P. Cortens, D. Ghosh, V. Spicer, W. Ens, K. G. Standing, R. C. Beavis, and J. A. Wilkins, *Use of peptide retention time prediction for protein identification by off-line reversed-phase hplc-maldi ms/ms*, Analytical chemistry, 78 (2006), pp. 6265–6269.

[8] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.

[9] J. L. Meek, *Prediction of peptide retention times in high-pressure liquid chromatography on the basis of amino acid composition*, Proceedings of the National Academy of Sciences, 77 (1980), pp. 1632–1636.

[10] V. Melnikov, E. Hüllermeier, D. Kaimann, B. Frick, P. Gupta, et al., *Pairwise versus pointwise ranking: A case study*, Schedae Informaticae, 2016 (2017), p. 7383.

[11] L. Moruz and L. Käll, *Peptide retention time prediction*, Mass spectrometry reviews, (2016).

[12] L. Moruz, D. Tomazela, and L. Käll, *Training, selection, and robust calibration of retention time models for targeted proteomics*, Journal of proteome research, 9 (2010), pp. 5209–5216.

[13] A. I. Nesvizhskii and R. Aebersold, *Interpretation of shotgun proteomic data the protein inference problem*, Molecular & Cellular Proteomics, 4 (2005), pp. 1419–1440.

[14] A. I. Nesvizhskii, O. Vitek, and R. Aebersold, *Analysis and validation of proteomic data generated by tandem mass spectrometry*, Nature methods, 4 (2007), pp. 787–797.

[15] K. Petritis, L. J. Kangas, P. L. Ferguson, G. A. Anderson, L. Paša-Tolic, M. S. Lipton, K. J. Auberry, E. F. Strittmatter, Y. Shen, R. Zhao, et al., *Use of artificial neural networks for the accurate prediction of peptide liquid chromatography elution times in proteome analyses*, Analytical chemistry, 75 (2003), pp. 1039–1048.

[16] K. Petritis, L. J. Kangas, B. Yan, M. E. Monroe, E. F. Strittmatter, W.-J. Qian, J. N. Adkins, R. J. Moore, Y. Xu, M. S. Lipton, et al., *Improved peptide elution time prediction for reversed-phase liquid chromatography-ms by incorporating peptide sequence information*, Analytical chemistry, 78 (2006), pp. 5026–5039.

[17] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, in Advances in neural information processing systems, 2012, pp. 2951–2959.

[18] I. A. Tarasova, C. D. Masselon, A. V. Gorshkov, and M. V. Gorshkov, *Predictive chromatography of peptides and proteins as a complementary tool for proteomics*, Analyst, 141 (2016), pp. 4816–4832.

[19] M. The, *Statistical and machine learning methods to analyze large-scale mass spectrometry data.* KTH, Gene Technology, TRITA-BIO-Report 2016:3, 2016. QC 20160412.

[20] S. Väljamets, *Peptide retention time prediction using artificial neural networks*, 2016.