Full-Spectrum Prediction of Peptides Tandem Mass Spectra using Machine Learning and Deep Neural Networks

Anmol Srivastava[1], Haoran Qiu[2], Searidang Pa[3]

srivastava.anm@northeastern.edu[1], qiu.haor@northeastern.edu[2], pa.s@northeastern.edu[3]

Khoury Collge of Computer Sciences[1,3]
College of Engineering[2]

Northeastern University, Boston, MA, US

1 Objective and Significance

Peptide Identification using Tandem Mass Spectra (MS) is one of the important and widely recognized concept in the field of proteomics. The intractable and convoluted nature of data makes it one of the difficult problem which elicit the need for automatizing the spectrum prediction, thereby augmenting the understanding of researchers for peptide fragmentation process. Traditional rule based processes for spectra prediction relies heavily focuses on backbone ions generated from peptide backbone collisions which makes then ignorant towards ions of low intensities.

In this paper we validate and recreate the work of researchers from *School of Informatics, Indiana University, Bloomington, USA* [1] with certain modifications and additional experiments. Their work involves developing deep learning model to predict the complete HCD spectra from peptide sequences without any prior fragmentation knowledge. In addition, multi-task learning approach has also been incorporated which facilitates the training of HCD spectra with insufficient training examples.

2 Background

To identify proteins is one of the main topics in proteomics. And the most common way is to use tandem mass spectrometry (MS) paired with advanced liquid chromatography.

First, the proteins of interest are digested into peptides using trypsin. Then, the peptides are separated by High Performance Liquid Chromatography (HPLC), ionized and sent to a mass spectrometer to measure the mass/charge ratio of each peptide [2]. In detail, precursors (which refers to the actual unique molecular units) are analyzed on the mass spectrometer [3]. Each precursor contains information about the sequence, the modification status, and the charge state [3]. Next, the peptides of interest are selected and sent to a collision cell for further fragmentation to produce tandem (MS/MS) mass spectra, which consists of a sequence of peaks, each characterizing the mass/charge ratio and intensity of an ion [2]. Finally, computer software is used to identify the peptide sequence associated with each MS/MS spectrum.

The problem in this process is that in the last step, when identifying peptide sequence with each MS/MS spectrum, either high quality spectra with nearly complete ladders of b/y ions is required to learn the peptide fragmentation pattern, or a complete sequence database including theoretical spectra determined by the in silico digestion and fragmentation of known proteins is needed to compare with experimental spectrum[2]. These two ways are not sufficient enough sometimes.

In the first way, we mentioned b/y ions: in the fragmentation process of peptides, when the backbone bonds (which means the primary structure of the peptides) cleave, the fragment peaks appearing to extend from the amino terminus are called “b ions”, and the fragment peaks appearing to extend from the C-terminus are called “y ions” [4]. For b icons, if it contains only the amino terminal amino acid, it is called b1 ion. If it contains the first two amino terminal amino acids, it is called b2 ion, and so forth [4]. The idea is the same for y ions.

However, peptides do not fragment sequentially, which means the first fragmentation event does not start from either amino terminus or C-terminus [4]; it could start at any point, therefore for some fragment peaks, it

is hard to tell whether it is b ions or y ions, which makes it hard to get a high quality spectra with nearly complete ladders of b/y ions.

The second way tends to be more useful than the first way, but there are still lots of spectra not included in the existing sequence database, so there is usually part of each experiential spectrum which is not included in the database and can not be used to identify a peptide sequence.

Therefore, deep learning has emerged to predict the complete spectra from peptide sequences without the requirements of high quality spectra with nearly complete ladders of b/y ions or a complete sequence database including theoretical spectra.

2.1 Previous Work

Before the advance of deep learning, previous methods used to tackle this spectra prediction problem can be divided into two main paradigms: kinetic model-based methods and pure machine learning based methods [5]. One example of the kinetic model-based methods is the MassAnalyzer algorithm, which explicitly models the chemical process of peptide fragmentation based on the mobile proton hypothesis. The key parameters of the model are tuned to fit annotated MS/MS spectra [4].

On the other hand, PeptideART, a purely machine learning based model, aims to learn peptide fragmentation rules in the form of posterior probabilities and then use the trained model for peptide identification [6]. First, it converts the problem of predicting spectral peak intensities into a multi-class classification problem. The posterior probability of the occurrence of each peak given a precursor sequence and its charge is learned by using just a shallow neural network, which is then used to make peptide identification.

It is important to note that in these two paradigms, the goal is to predict only the spectra from the backbone ions and its derivative. Because of the lack of mechanical explanation of non-backbone fragment ions, it is not plausible to have an annotated MS/MS training set for them. It is also difficult to provide fragmentation rules to guide machine learning algorithms to learn the intensities of the non-backbone ions. However, in practice, backbone ions and its derivatives only account for at most 70% of the total intensities in the HCD spectra [7].

Recently, due to the advance of deep learning and the increase of abundance of high quality MS/MS data, several deep learning approaches have been proposed. A well-designed and well-trained deep neural network would be able to learn how to craft the fragmentation rules directly from the spectrum. Not only would this help us tackle the prediction of the spectra of the non-backbone ions, but the fact that one does not need to have annotation on the spectrum in the data set also allows the training set to be sufficiently large enough for the deep network model. Previous deep learning-based spectral prediction tools prior to this paper (PredFull) such as pDeep [5], DeepMass [8], and Prosit[9]only aim to predict the intensity of expected fragment ions such as b/y ions, which are derived based on rational fragmentation rules.

Moreover, previous work prior to PredFull only aimed to predict the spectra for fragment ions of charge +2 and +3, because the number of fragment ions that have +1 and more charge than +3 charges is too small for a machine learning-based method [5]. In addition to predicting the full spectrum, PredFull also tackles this problem by implementing a multitask approach .

2.2 What makes our work distinct or particularly interesting.

By replicating PredFull, we hope to learn and explore the deep learning techniques that the paper uses. First, we want to understand how the architecture of the network enables such good performance. Then, we also would like to see if there is any improvement that can be done through tweaking the model in any way. We want to experiment with using different loss functions. We also would like to see if how the result will vary when we use a shallower network.

3 Methods

3.1 Dataset

For the purpose of this paper, the training data has been acquired from various spectra libraries such as the NIST HCD library [10], the NIST Synthetic HCD library [10], the Human HCD library from Proteome Tools [11], and MassIVE [12]. The test dataset has been obtained from the PredFull website [7] under the name hcd_testingset.mgf. The test set has 19478 data points with charge 2 that has peptide sequence length under or equal to 25.

Due to the limitation of time and computational resources, we decide to focus only on peptides with $+2$ charges obtained by the HCD method. In total, our training data set contains 1, 759, 639 data points. Below is the breakdown of the number of valid data points with charge +2.

- Proteome: 142,609
- NIST: 648,968
- NIST Synthetic: 392,217
- MassIVE: 575,845

We will split the training data further into cross validation set when we train our shallow network as we will discuss later, while for the deep network, we will just follow the paper's architecture.

3.2 Data cleaning and processing

Our input to the model is a 21x25 matrix, where row 1 to row 20 contains the one hot encoding for each of the 20 amino acids and the last row contain the monoisoptopic mass of the amino acid that is present in the sequence. The monoisotopic mass of the amino acids is obtained from [13]. We can disregard the relatively rare spectra sequences whose peptide length is greater than 25, and we also disregard the cases where the peptide sequence contain mutation or variations from the 20 amino acids.

Following the paper, we represent an MS/MS spectrum as a sparse one-dimensional vector by binning the m/z range between 180 and 2000 with bin width of 0.1, since they report that a more granulated bin width did not result in improved prediction. We limit the range to 0-2000 because there are very few MS/MS spectra contain peaks with m/z above 2000. Therefore, the target is a sparse vector encoding of 1 x 20000 dimensions.

Since the high quality of data is one of the crucial factor for the successful convergence of deep learning/machine learning models, we remove the suspicious data points using two conditions. If the spectra is under-fragmented or over-fragmented; i.e, containing fewer than 20 peaks or more than 500 peaks, we remove it. Another condition is that we check if the precursor mass difference is more than 200pm. A precursor mass difference higher than 200pm would imply that the instrument or the experiment is too unreliable. We use the following formula obtained from [14] to calculate the precursor mass difference.

$$\text{ppm} = \frac{\text{Experimental mass - exact mass}}{\text{exact mass} \times 106}$$

where the experimental mass is obtained from the dataset and the exact mass is obtained by adding the monoisotopic mass of each amino acid in the peptide.

Finally, because the absolute intensities in the MS/MS spectra are irrelevant to the peptide identification task, we normalize all spectra in training and testing sets by dividing the maximum peak intensity in each spectrum. Moreover, for the data loading process of the model training to be feasible and fast for our limited computational resources, we divide the data into chunks of size 100 data points.

```
BEGIN IONS
PEPMASS=705.6675415039
CHARGE=3
MSLEVEL=2
COLLISION_ENERGY=0.0
FILENAME=filtered_library_mgf_files/e6ed062958694bf49b9b3842b33462ed.mgf
SEQ=AAAAASSSSPSCTPATSQGHLR
PROTEIN=sp|Q6ZU67|BEND4_HUMAN
SCANS=1
SCAN=1
SCORE=21.4134303487
FDR=0.0
115.08665466308594  173094.53125
143.08157348632812  551418.5625
169.09716796875 242847.8125
175.11915588378906  88141.09375
185.09156799316406  43390.2890625
214.1184844970703   1169041.75
215.1217498779297   111555.9140625
228.09707641601562  32184.994140625
230.1133575439453   42946.86328125
240.1343231201172   166376.421875
242.11338806152344  52422.07421875
257.1607971191406   330362.28125
258.163818359375    33996.15625
283.1405944824219   83780.5234375
```

Figure 1. An example of a data point from ProteomeTools

To give a clearer picture of the raw data, as seen as figure 3.2, each spectra begins with the string "BEGIN IONS". We would need the following information from the header information: the charge, the experimental mass (PEPMASS), and the sequence (SEQ). The spectra is encoded by the two columns where the first column contains the m/z values and the second column contains the peak intensity value.

3.3 +2 HCD Spectra Prediction

An implementation of Convolutional Neural Network using Keras framework with TensorFlow backend is being used to predict the doubly (+2) charged HCD spectra.

Figure 2 depicts the architecture of the neural network which is similar to a generalized Seq2Seq model based on Residual Neural Networks.

The input feature space of one-hot encoding matrix is passed to the 8 separate 1-dimensional convolutional layer of multifarious kernel size ranging from 2 to 9. These separate 8 convolutional 1 dimensional layer is responsible for capturing the correlations among subsequences of the input peptide.

Afterwards, the output from all these 8 CNN layers is merged together and passed through 10 residual blocks. This will also ensure that there is no problem of gradi-

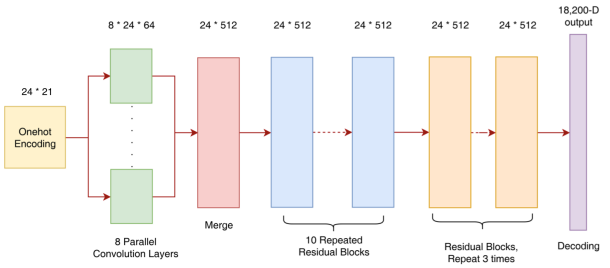

Figure 2. *Architecture of Residual CNN for Spectra Prediction* [1]

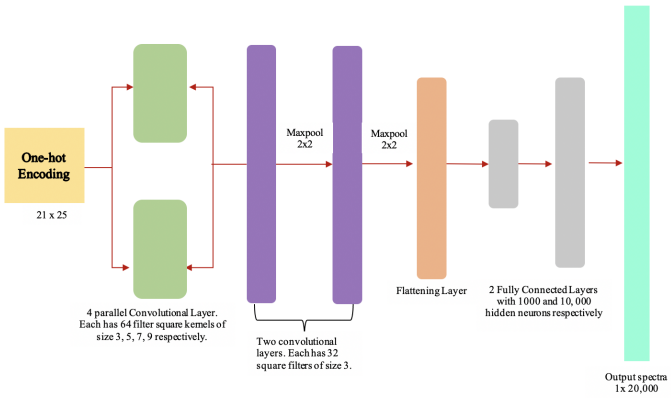

Figure 3. *Architecture of Shallow CNN for Spectra Prediction (Model 4)*

ent vanishing during training phase. The last 4 layers of the propose architecture takes a resulting tensor as an input ad decodes it into MS/MS peptide sequence of 20,000 length.

In addition to the proposed architecture in **Figure 1**, we have also tried the similar approach with shallow neural network, where the same input matrix is passed to 4 separate parallel convolution layers. However, in this shallow network the resulting merged tensor will be passed to two different 2-dimensional convolution layers each with the kernel of dimensions 3 by 3. The next layer of the network flattens the resulting tensor into 1-dimensional which is then passed through two fully connected layers of 1000 and 10,000 neurons and outputs the spectra sub-sequence as shown in **Figure 3.3**. The purpose of the fully connected layer is to run the linear regression on CNN output vector with linear weights.

3.3.1 PeptideNet Implementation

PeptideNet is our implementation of *predfull* model with certain modifications like the input one-hot encoding matrix of a peptide sequence is of (21×25) shape where row 1 to row 20 are the one hot encoding of peptide sequence and the last row encompasses mono-isotopic amino acid mass.

The network architecture is being trained using **Adam** optimizer and **Mean Squared Error** loss function for over 50 epochs with a learning rate of $\mathbf{0.001}$. Along with these certain parameters *Callback* features of Keras framework is being used for optimizing and boosting training of neural networks:

- **Model Checkpoint** ensures that only the best weights of neural network is being saved over course of epochs by monitoring the loss.
- **ReduceLROnPlateau** is one of the most important callback feature provided by tensorflow framework which make sure that our optimizer function does not get stuck on plateau/local minima by adjusting the learning rate after wait time of specified epochs. Our general approach is to start with reasonably high value of learning rate to kick start the training in beginning and reducing it accordingly at later stage of training phase by factor of 0.1 on the basis of convergence of loss function.
- **EarlyStopping** abrupt the training in between in order to save the training time when performance stops improving for large count of epochs.

Due to the availability of limited resources and time, the PeptideNet is being trained to predict the mass spectra of +2 charge peptide sequence which could be easily extended to other various charged sequences.

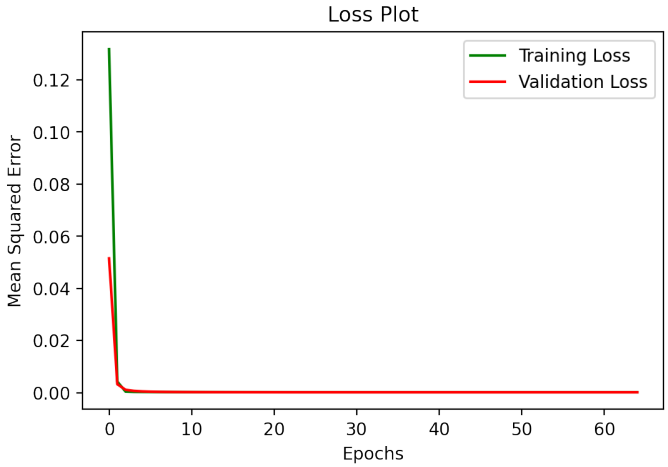

Figure 4: Mean Squared Error Loss with Early Stopping around 65^{th} epoch

Figure 4 depicts the convergence of training and validation loss over 50 epochs. A smooth monotonic decrements in mean square error could be easily observed

which further illuminate the absence of over-fitting as both training loss and validation loss tends to be approximately close to each other which bolster the generalization ability of predfull model as asserted in the original paper [1].

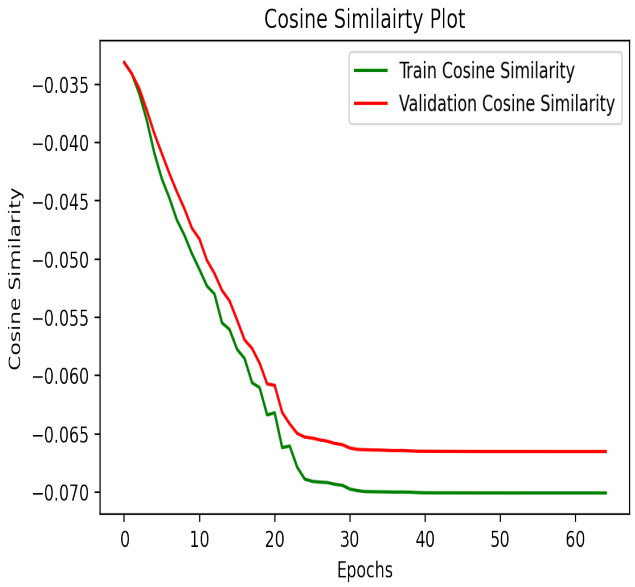

Figure 5: Cosine Similarity Loss with Early Stopping around 65^{th} epoch

	PeptideNet
Train MSE Loss	$1.7763e^{-4}$
Test MSE Loss	$1.6667e^{-4}$
Train Cosine Similarity Loss	-0.0701
Test Cosine Similarity Loss	-0.0665

In addition to mean squared error, we have also documented the changes in cosine similarity loss over period of epochs. According to the [15], the range of this loss function is [-1,1]. When the number is between -1 and 0, the value closer to -1 symbolizes the similarity between the two vectors whereas on other hand values closer to 1 indicate greater dissimilarity. Therefore, this loss could be used in scenarios where our goal is to reduce the distance between the target and the predicted vector. An regular decrease in the value of this loss function further stipulates the credibility of performance of PeptideNet neural network along with minimum difference between values on training set and validation set (*as depicted in* ***Figure 5***).

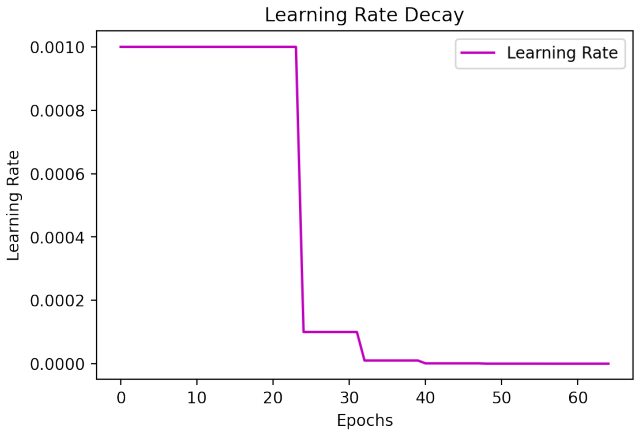

Figure 6: Learning Rate Decay during Training Phase

During the whole training phase, it could be observed from ***Figure 6*** that the learning rate get reduced three times, firstly around 25^{th} epoch from $\mathbf{0.001}$ to $\mathbf{0.0001}$, secondly at 32^{nd} epoch from $\mathbf{1.0e^{-3}}$ to $\mathbf{1.0e^{-5}}$ and lastly at 40^{th} epoch to $\mathbf{1.0e^{-6}}$. These decay trend depicts the convergence of our model to the vicinity of global minima which forces **Learning Rate** to reduce in order to slow down the rate of convergence as compared to the initial epochs.

The model is being trained on training set and tested on validation set which was generated using the splitting ratio of 0.25. The hardware resources needed for training is provided by *Northeastern University's* Discovery Cluster.

3.3.2 Shallow CNN implementation

We perform a model selection where we set aside 25% of the training set for cross validation. We trained 5 models with slightly different architecture using the mean squared error loss function (MSE) and Adam optimizer with learning rate = 0.0001 and weight decay = 0.0001.

Due to the limitation of our computational resources and our unexpected technical incident with discovery, we could not train on GPU for the shallow network. Therefore, we resort to sampling the dataset 100 data points randomly without replacement each time for back-propagation. All models were trained for roughly 8 hours, with the stopping criterion being that the difference between the MSE in the previous round and the current round less than 1e-6. The MSE of each model starts with around 0.2 and decreases to fluctuate around 0.00010 - 0.0002.

The descriptions of the 5 models are listed below. Because our model are only slight variation from each other, for the sake of simplicity, we will describe it in relative

Table 1. MSE on cross validation set for the 5 models

	Cross Validation MSE
Model 1	1.76e-4
Model 2	1.63e-4
Model 3	1.67e-4
Model 4	1.62e-4
Model 5	2.12e-4

terms to model 4 in Figure 3.3.

- Model 1: The same architecture except that there is only one fully connected layer with 2000 neurons after the convolution layer
- Model 2: The two fully connected layers have 1000 and 1000 neurons.
- Model 3: Max pool 2 x 2 between the two middle convolution layers and there is only one fully connected layer with 10,000 neurons.
- Model 5: Max pool 2 x 2 between the two middle convolution layers and there are three fully connected layer with 1000, 4000, and 10,000 neurons.

For model 1, 2, 3, and 5, the activation function in all layers except the last layer is ReLu. The activation function of the last layer is Sigmoid because our output range is between 0 and 1. For model 4, we experimented with the new activation function PReLu. Because our output is a sparse vector and because ReLu suffers from the dead neuron problem, it could be the case that a lot of the neurons just output zero all the time. Hence, we thought that LeakyReLu might be a better option. PReLu is the same, in principle, as LeakyReLu, except that it also learns the slope in the negative penalty region [16].

We note that the MSE of model 2 and model 4 are quite close. However, we decide to pick model 4 as our winner, because max pool has the advantage of shrinking the number of trainable parameters down significantly, which would speed up the training process. We also thought that

the architecture of model 4 is somewhat similar to that of a decoder model, where the size of the model grows bigger and bigger as you go in layer.

Finally, we trained two versions of model 4 on the whole training set, each for 24 hours. One uses MSE loss, while another uses cosine similarity loss.

3.4 Performance Evaluation

The predicted and the original spectra could be evaluated using some of the popular metrics prominent for measuring the similarities between two vectors. In addition to cosine similarity as suggested in the original paper, we will be going to experiment with Euclidean Distance.

3.4.1 Cosine

Cosine similarity is the measure obtained from the multiplication of cosine angle of two different vectors using the following mathematical formula [17]:

$$Cosine = \frac{A \cdot B}{||A|| \cdot ||B||}$$

where A and B are two different vectors that need to be compared. in our case it could be the 1 dimensional predicted and original spectra.

One important thing to note is that there is a difference in the intepretation of the similarity based on the output of the cosine loss function between Pytorch implementation and Keras. We used Pytorch to implement shallow network and Keras to implement deep network. As noted above, for Keras, when the number of the loss function is between -1 and 0, the value closer to -1 symbolizes the similarity between the two vectors whereas on other hand values closer to 1 indicate greater dissimilarity.

However, for shallow network, we used the CosineEmbeddingLoss function from Pytorch [18]. The loss function takes in input tensor (x_1, x_2), which is our model prediction and the truth spectra and compute the following loss for the target label $y = 1$.

$$\text{loss}(x, y) = 1 - \cos(x_1, x_2)$$

We always normalize our perform a normalization of the prediction vector and the truth spectra vector. Our target is always 1, because we want the angle between the two vectors x_1 and x_2 to be zero, which would mean the same vector. The range of the loss function is $(0, 2)$. Therefore, when the loss is closer to zero, our vector is getting closer to each other. When the loss is closer to 2, the cosine of the angle of the two vectors is closer to -1, which indicates greater degree of dissimilarity.

3.4.2 Euclidean Distance

The euclidean distance is provided by the following formula which measures the distance between two points in the euclidean space.

$$D(p,q) = \sqrt{\sum_{i}^{n}(p_i - q_i)^2}$$

3.5 Result

In this section, we have evaluated the predictions of various versions of Shallow CNN network and PeptideNet via means of plots between the predicted spectra and the true m/z spectrum of given peptide sequence. In each graph, observed spectrum is plotted via blue color and the predictions are shown using green color.

3.5.1 Model 4 with Euclidean Distance and Cosine Similarity Loss Function - Shallow CNN

Table 2. MSE and Average Cosine Similarity evaluation on the Test set for our selected model

	MSE loss	Cosine Similarity
Model 4	0.236	0.132

Table 2 shows the MSE loss evaluation on the test set by model 4 trained with MSE loss function and shows the cosine similarity evaluation on the test set by model 4 trained with cosine similarity embedded loss function. A cosine similarity close to 1 means that the predicted spectra vector is very similar to the target spectra. It is clear that our model trained with cosine similarity has very poor performance. On the other hand, it is difficult to interpret what the MSE loss means in this case, because our output is a sparse vector in a very high dimensional space and therefore any two vectors would have very small euclidean distance

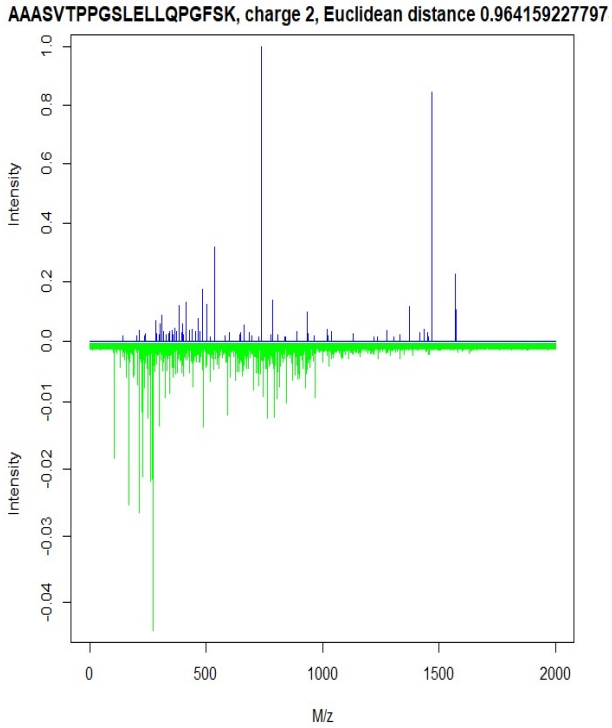

Figure 7: An example of spectra prediction using Model 4 with MSE loss function

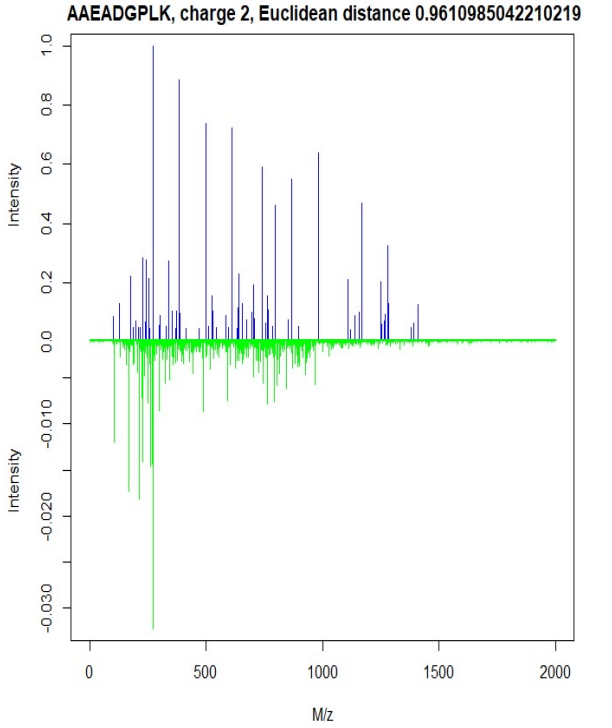

Figure 8: An example of spectra prediction using Model 4 with MSE loss function

For the two different peptide sequence, the spectrum prediction generated by **Model 4** with respective **0.964** and **0.961** euclidean distance is depicted in **Figure 7** and

Figure 8 respectively. It could be inferred that there are only 2 major ion peaks present in the original spectrum. However, predicted spectrum hosts only 1 major ion peak with most of the predictions skewed towards the left side of the graph. In addition, length of bins differs a lot which clearly stipulates the model incompetence of predicting mass spectra from peptide sequence. Similarly for the shorter sequence left skewness in prediction with relatively less ion peaks as compared to original spectrum could be observed.

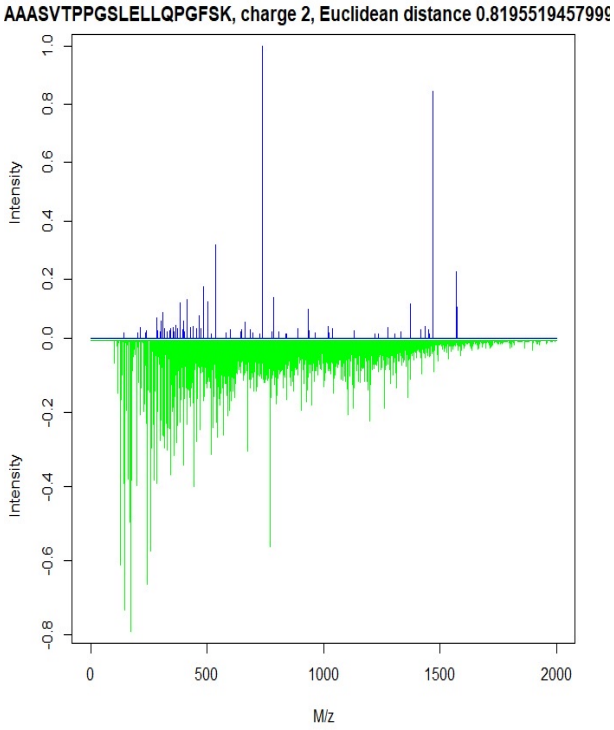

Figure 9: An example of spectra prediction using Model 4 with cosine similarity loss function. Similarity closer to 1 indicates greater similarity while similarity closer to -1 indicates greater dissimilarity.

From Figure 8 and Figure 10, it looks like the model has at least learned something as the predicted spectra has peak fragments at similar m/z bin place as the truth spectra. However, it is clear that our shallow network has poor performance. We pass the same peptide sequence for Figure 8 and 10 through model 4 trained with ED and with cosine similarity respectively and similarly for figure 7 and figure 9. From figure 7 and 9, it seems that model 4 trained with cosine similarity produce a much more noisy spectra prediction than model 4 trained with ED.

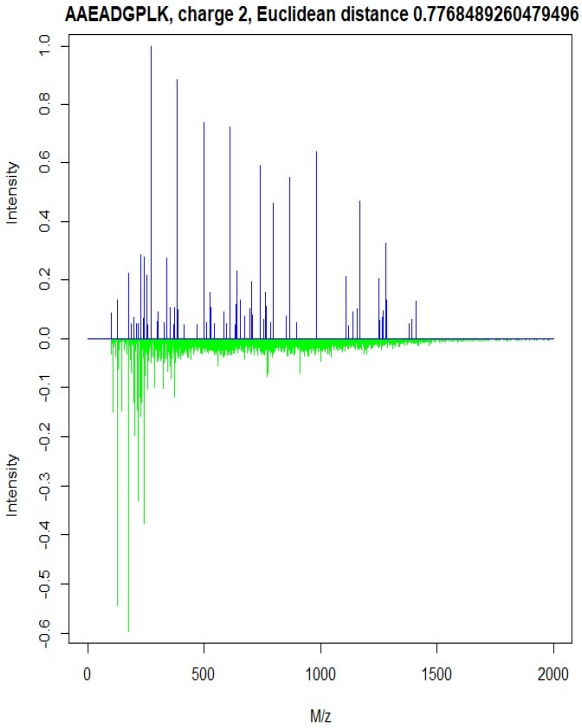

Figure 10: An example of spectra prediction using Model 4 architecture with cosine similarity loss function

3.5.2 PetideNet

As discussed above peptide net is the similar architecture as original predfull model with certain modifications. It has been strongly asserted in the *predfull paper* that combination of squeeze and excitation blocks and residual block network is far more proficient in predicting mass spectrum of protein sequences because of its architecture designed to capture the correlations among the subsequences of the input peptide.

However, the graph shown in ***Figure 11*** depicts the contrasting case where, peptideNet is abortive is generating accurate spectrum predictions for peptide sequence. This further opens up the need of analysis which suggests that the shallow cnn converges before towards the global minima as compared to the complex peptideNet. This could be credited to the slow training of network with almost 19 million total trainable parameters. It coudl be observed from the MSE plot (**Figure 4**) that after initial 11 epochs the change in loss value is significantly small with each successive value being differed with difference of 0.001 on every epoch.

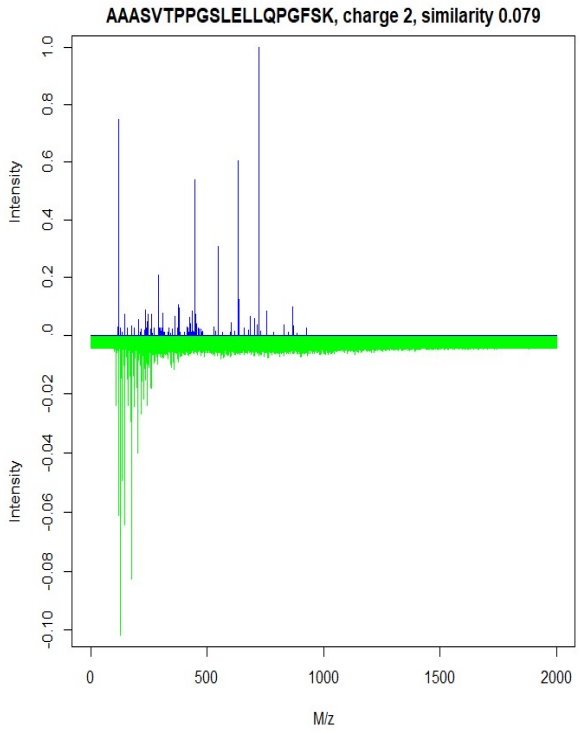

Figure 11: An example of spectra prediction using PeptideNet. Note here that the similarity value closer to -1 indicates greater similarity while value closer to 1 indicates greater dissimilarity.

3.6 Conclusion

As we are recreating the paper [1], we expect to see the similar results with the first residual CNN network on +2 HCD spectra prediction. In addition, in order to verify the assertion made in the original paper that *"deep neural networks are optimal for spectra predictions"*, we have implemented the shallow neural network for the specific purpose and expect to see the poor performance with the new devised architecture.

We have also done a comparison between two evaluation metric in this project. However, because both of our models do not have good enough performance for us to see visually which generated spectra is better, we cannot provide a conclusive argument for which evaluation metric is better. We had conjectured that cosine similarity would outperform Euclidean distance, because our output is a sparse vector in a very high dimensional space and therefore any two vectors would have very small euclidean distance. However, we see in one of figure above that Euclidean distance seem to cleaner and clear-cut peaks.

We did everything we could to match how the paper trained their model, but we did not succeed. We have thought of a few reasons why this might be the case. One is the inherent difficulty in training a deep neural network as initialization can greatly impact the outcome of the model. Another reason is that we might just need more computing power and time to train our model. One approach to take in the future, if one has enough computational power and time, would be to train an ensemble of models and take an average of each model, because as shown in the lecture, a committee of machines always lead to better result in the long run.

4 Individual Tasks

I wrote the entire data cleaning and processing for the training set for all four datasets. Each dataset contains slightly different headers. I also wrote a separate script for the test set as the format is slightly different. I spent a lot of effort and time trying to understand the paper's rationale behind their choice of representation of the input and output. The paper actually contained some mistakes. I found out what the precursor mass difference means and how to calculate it in ppm. It was also difficult to understand the format of the dataset and what the meaning of the headers are, given that I have no background in biology. I also wrote the shallow models, trained them, and did cross validation. I also wrote the evaluation script on the test set.

Hoaran implemented the visualization graphs that compare between the prediction spectra and the ground truth spectra of a sampled data point. She helped running the training script and the evaluation on the test set script of the shallow network. Because I was in quarantine and had limited internet, Hoaran helped me download, upload and the dataset and ran the script on discovery cluster

Anmol implemented, trained, and evaluated the deep neural network.

References

[1] L. W. Y. Y. H. T. Kaiyuan Liu, Sujun Li, "Full-spectrum prediction of peptides tandem mass spectra using deep neural network,"

[2] B. A. W. B. J. J. M. Y. C. M. C. B. B. C. E. H. . K. R. E. Liu, Jian, "Methods for peptide identification by spectral comparison. proteome science," *Proteome Science*, vol. 5(1), 2007.

[3] SEZ, "What is the difference between peptide, modified peptide and precursor."

[4] Z. Zhang, "Prediction of low-energy collision-induced dissociation spectra of peptides," *Analytical Chemistry*, p. 6364–6373, 2005.

[5] A. D. T. H. R. P. Arnold RJ, Jayasankar N, "pDeep: Predicting MS/MS Spectra of Peptides with Deep Learning," *Anal Chem*, vol. 219, no. 30, 2017.

[6] A. D. T. H. R. P. Arnold RJ, Jayasankar N, "A machine learning approach to predicting peptide fragmentation spectra," *Pac Symp Biocomput*, vol. 219, no. 30, 2006.

[7] L. W. Y. Y. Kaiyuan Liu, Sujun Li and H. Tang, " Full-Spectrum Prediction of Peptides Tandem Mass Spectra using Deep Neural Network," *Anal. Chem*, vol. 92, no. 6, p. 4275–4283, 2020.

[8] L. R. G. P. e. a. Tiwary, S., "High-quality MS/MS spectrum prediction for data-dependent and data-independent acquisition data analysis. ," *Nat Methods*, vol. 16, p. 519–525, 2019.

[9] S. T. Z. D. e. a. Gessulat, S., " Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning.," *Nat Methods*, vol. 16, p. 509–518, 2019.

[10] S. E. S. Xiaoyu Yang, Pedatsur Neta, "Extending a tandem mass spectral library to include ms 2 spectra of fragment ions produced in-source and ms n spectra.," *Journal of The American Society for Mass Spectrometry*, vol. 28, pp. 2280–2287, 2017.

[11] S. K. Z. J. K. T. D. B. B. D. G. S. E. H. W. M. Y. P. S. J. K. K. S. T. K. U. D. E. A. R. M. R. W. H. M. T. A. S. H. A. R. U. K. B. Zolg DP, Wilhelm M, "Building proteometools based on a complete synthetic human proteome," *Nature Methods*, vol. 14, p. 259, 2017.

[12] J. C. B. S. P. S. W. C. N. B. Mingxun Wang, Jian Wang, "Assembling the community-scale discov- erable human proteome," *Cell Systems*, vol. 7, pp. 412–421, 2018.

[13] Expasy, "The amino acid masses."

[14] Waters, "Mass accuracy and resolution."

[15] TensorFlow, "Cosine similarity loss."

[16] SG, "Prelu activation."

[17] M. L. G. Katty X. Wan, Ilan Vidavsky, "Comparing similar spectra: from similarity index to spectral contrast angle," *Journal of the American Society for Mass Spectrometry*, vol. 13, pp. 85–88, 2007.

[18] Pytorch, "Cosineembeddingloss."