Deep Learning for Peptide Feature Detection from Liquid Chromatography - Mass Spectrometry Data

by

Fatema Tuz Zohora

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Michael Hallett Professor Department of Biochemistry University of Western Ontario
Supervisor:	Ming Li Professor David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Lila Kari Professor David R. Cheriton School of Computer Science University of Waterloo
	Bin Ma Professor David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Brendan J. McConkey Associate Professor Department of Biology University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Proteins are the main workhorses of biological functions and activities, such as catalyzing metabolic reactions, DNA replication, providing structure to cells and organisms, etc. Comparative analysis of protein samples from a healthy person and disease afflicted person can discover disease biomarkers, which can be diagnostic or prognostic of the respective disease. Liquid chromatography with tandem mass spectrometry (LC-MS/MS) is the cutting-edge technology for protein identification and quantification. In this thesis, we target the first step in the LC-MS/MS analysis: peptide feature detection from LC-MS map, which is promising for disease biomarker discovery and protein quantification. LC-MS map is usually a three-dimensional plot where peptide features form multi-isotopic patterns. Each map may contain hundreds of thousands of peptide features, which frequently overlap, are tiny with respect to the background, and are often blended with feature-like noisy signals. All of these characteristics make peptide feature detection very challenging. However, deep learning is bringing groundbreaking results in various pattern recognition contexts. Therefore, in this thesis, we investigate deep learning models to address the peptide feature detection problem.

Existing tools for peptide feature detection are designed with domain-specific parameters whose different settings bring very different outcomes and, thus, prone to human error. Moreover, they are hardly updated despite a vast amount of newly coming proteomics data. As a solution, we develop a foundation for applying deep learning in automating peptide feature detection for the first time. The main strength of our approach is that it provides higher sensitivity than other existing tools by learning necessary parameters through training on the appropriate dataset, and newly available information can be easily integrated through fine-tuning the model. We first propose DeepIso, combining convolutional neural network (CNN) and recurrent neural network (RNN), providing higher sensitivity for peptide feature detection than other existing models. Then we offer PointIso, a point cloud based (set of data points in space) deep learning model with attention-based segmentation, which is three times faster than DeepIso and improves the feature detection as well. PointIso's sensitivity for detecting identified spiked peptides on a benchmark dataset is about 98%, which is 5% higher than other existing models. Then we perform a quality assessment of the peptide features generated by PointIso, showing its potential for biomarker discovery. We also apply PointIso to relative peptide abundance calculation among multiple samples, demonstrating its utility in label-free quantification. Finally, we adapt our 3D PointIso model to handle 4D data, achieving 4-6% higher sensitivity than other algorithms on the human proteome dataset. Therefore, our model is transferable to various contexts. We believe our research makes a notable contribution to accelerating the progress of deep learning in the proteomics area, as well as general pattern recognition study.

Acknowledgements

First of all, I thank Allah (God) for bringing me into existence through such lovely parents, providing me the privilege of a comfortable life, opportunities to pursue higher studies at the best institutions, and giving me the stamina to make this thesis possible.

I wholeheartedly thank my supervisor, Professor Ming Li, for introducing me to significant life science problems and making me brave to explore diverse, innovative machine learning ideas. His thoughtful advice has made me an independent thinker and taught me the value of quality research, ethics, and professionalism. I hope to be able to turn to him for his guidance in the years to come. He has not only been a mentor to me in academic research but provided mental support during my saddest incident in life so far, which let me realize what a wonderful person he is! I will always be grateful to him for that.

I highly appreciate our collaborators from Bioinformatics Solutions Inc. Ngoc Hieu Tran contributed by suggesting various deep learning ideas. Rui Xiao helped by illustrating the scope of point cloud based models. M. Ziaur Rahman assisted in data generation and having a better insight into the proteomics domain. Lei Xin and Baozhen Shan helped in conducting the quality assessment of our research. All of them helped us to finish this thesis successfully. I also thank Johra Muhammad Moosa and other friends for being there for me during the hard times, for making me feel supported, loved, and cared for.

I am thankful to my dissertation committee members: Professor Lila Kari, Professor Bin Ma, Professor Brendan J. McConkey, and Professor Michael Hallett, for their time in reviewing and providing valuable comments on this dissertation. Their constructive feedback and critical questions have helped me further nourish this thesis. I thoroughly enjoyed the conversation with them during my PhD seminars and oral defense.

I would be indebted to my husband for his amazing support and encouragement throughout this long period of PhD program. I am grateful to my four years old son for making me strong, steady, and patient; to my father for being remarkably forbearing to me and making me ambitious about life; to my mother for showing me the glory of empathy and affection; and to my siblings for making my life joyful. I highly appreciate my in-laws who always feel proud about my study & career. All of them made me the person who I am today.

Dedication

This is dedicated to my *angel*: my mother *Monowara Begum*, and my two *superheroes*: my son *Saifan Chowdhury* & my father *Sirajul Hoque*.

Table of Contents

List of Figures xii

List of Tables xxiv

1 Introduction 1

1.1 LC-MS/MS Analysis Workflow . . . 3

1.1.1 Peptide Feature Detection . . . 3

1.1.2 Peptide Identification . . . 6

1.1.3 Peptide Quantification . . . 7

1.2 Existing Methods of Peptide Feature Detection . . . 8

1.3 Motivation for Peptide Feature Detection . . . 10

1.3.1 Lable-Free Quantification (LFQ) . . . 10

1.3.2 Biomarker Discovery . . . 13

1.3.3 Identifying Chimeric Spectra in DDA or DIA . . . 13

1.4 Deep Learning . . . 14

1.4.1 Convolutional Neural Network (CNN) . . . 15

1.4.2 Recurrent Neural Network (RNN) . . . 15

1.4.3 Combination of CNN, RNN and Attention Mechanism . . . 17

1.4.4 3D Point Cloud Based Models . . . 20

1.4.5 Deep Learning in Proteomics . . . 20

1.5 Overview on Research Contribution 21
1.5.1 List of Developed Models and Experimental Analysis 22
1.5.2 Model Training Criteria 23
1.5.3 Model Evaluation Criteria 24
1.6 Thesis Organization 25

2 Naive Convolutional Neural Network For Peptide Feature Detection 26
2.1 Workflow 26
2.2 Dataset 27
2.3 Result 29
2.3.1 Model Sensitivity 30
2.3.2 Model Specificity 31
2.3.3 Verification of Peptide Intensity 33
2.4 Methods 33
2.4.1 Training Data Generation 34
2.4.2 Model Training Parameters 34
2.4.3 Heuristics Steps 35
2.4.4 An Intuitive Example 39
2.4.5 Discussion 39

3 DeepIso: A Deep Learning Model for Peptide Feature Detection from LC-MS Map 41
3.1 Workflow of DeepIso 42
3.2 Results 43
3.2.1 Dataset 43
3.2.2 Training of DeepIso Model 44
3.2.3 Testing of DeepIso Model 48
3.3 Architectural Details and Methods for Reproducing DeepIso 51

3.3.1 Step 1: Scanning of LC-MS map by IsoDetecting module to detect isotopes . . . 51
3.3.2 Intermediate Step to Make a Sequence of Isotopes . . . 54
3.3.3 Step 2: Scanning of detected isotopes by IsoGrouping module to report peptide feature . . . 55
3.3.4 Ensemble of Multiple IsoGrouping Modules . . . 59
3.3.5 Fine-tuning DeepIso with Misclassified Features . . . 60
3.4 Discussion on the Design Strategy & Performance . . . 62
3.5 Data & Code Availability . . . 68

4 PointIso: Point Cloud Based Deep Learning Model with Attention Based Segmentation 70
4.1 Workflow of PointIso . . . 71
4.2 Results . . . 73
4.2.1 Dataset . . . 73
4.2.2 Training of PointIso . . . 73
4.2.3 Performance Evaluation of PointIso . . . 74
4.2.4 Peptide Feature Intensity Calculation by PointIso . . . 77
4.2.5 Time Requirement of PointIso . . . 77
4.3 Discussion on the Design Strategy & Performance . . . 78
4.3.1 IsoDetecting Module Changes from Image Based Model To Point Cloud Based Model . . . 79
4.3.2 Weighted-Cross Entropy Loss for IsoDetecting Module . . . 80
4.3.3 Attention Mechanism in IsoDetecting Module . . . 80
4.3.4 Upgrading IsoGrouping Module . . . 82
4.3.5 Fine Tuning . . . 84
4.3.6 Impact of Secondary Signals on Total Number of Features . . . 84
4.4 Architectural Details and Methods for Reproducing PointIso . . . 85

4.4.1 Step 1: Scanning of LC-MS map by IsoDetecting module to detect isotopes . . . 85
4.4.2 Step 2: Scanning of LC-MS map by IsoGrouping module to report peptide feature . . . 90
4.4.3 Fine Tuning Using Misclassified Features . . . 96
4.5 Data & Code Availability . . . 100

5 Assessment of PointIso for the Practical Application and Extension for 4D Peptide Feature 101
5.1 Disease Biomarker Detection . . . 102
5.1.1 Feature Quality Assessment . . . 102
5.2 Label-Free Quantification (LFQ) . . . 105
5.2.1 Dataset . . . 107
5.2.2 LFQ steps . . . 107
5.2.3 Evaluation of PointIso for LFQ . . . 109
5.3 PointIso Extension for Higher Dimensional Data . . . 111
5.3.1 Adaptation Strategy . . . 113
5.3.2 Dataset . . . 114
5.3.3 Evaluation of PointIso for 4D dataset . . . 115
5.3.4 Data & Code Availability . . . 115

6 Conclusion and Future Work 117
6.1 Main Research Contribution . . . 117
6.2 Future Works . . . 119
6.2.1 Chimeric Spectra Identification . . . 119
6.2.2 Model Improvement through Semi-supervised Learning . . . 120
6.3 Author Contribution and Acknowledgement . . . 122

References 124

APPENDICES **134**

A Supplementary Notes **135**

A.1 Scanning Window Dimension 135

A.2 Cross-Validation Technique 135

A.3 Class Weight Assignment Procedure 137

A.4 Candidate solutions for boundary region point segmentation in PointIso . . 137

B Supplementary Methods **141**

B.1 Augmented Data Generation for 'IsoDetect' Module 141

B.2 Attention Calculation Flowchart for PointIso 141

B.3 Resolution Degradation in IsoGrouping Module 143

B.4 Merge Secondary Peaks in PointIso 143

C Supplementary Tables **144**

C.1 Comparative Analysis of 3D Peptide Feature Detection Tools 144

C.2 Comparative Analysis of 4D Peptide Feature Detection Tools 149

D Supplementary Figures **152**

List of Figures

1.1 The m/z vs Retention Time (RT) plane of a LC-MS map is shown in the leftmost image. A small area is zoomed in next, where we can see many groups of vertical lines/traces. Each of them is called peptide feature. We mark a peptide feature by rectangle and see the detailed 3D view of this peptide feature in the next image. 5

1.2 Shape of isotopic signal and feature intensity distribution. (a) A peptide feature having four isotopes is shown in the left. The intensity signal of each isotope forms a beta distribution shaped curve (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) if watched at $[RT \times I]$ plane. Again, if looked at $[m/z \times I]$ plane, it forms a bell shaped curve. 5

1.3 Peptide sequencing from an MS/MS spectrum. For a given peptide sequence, the B ions are the product when the charge is retained on the N-Terminus (i.e., at the beginning of the sequence) and the Y ions the product when the charge is retained at the C-Terminus (i.e., at the end of the sequence). Each amino acid is identified by the mass difference between neighbouring peaks (B or Y) in the spectrum. The peptide sequence is predicted as the optimum one that best fits fragment ions in the spectrum. The figure was generated from the tool PEAKS Studio. 7

1.4 Workflow of Dinosaurs for peptide feature detection. It converts the raw LC-MS maps to mzML data format before starting the analysis. 9

1.5 Label-Free Quantification: Peptide features are mapped across multiple replicates. For instance, the connected marked rectangles shown in this figure. 11

1.6 MNIST digit recognition network. Here a $[32 \times 32]$ input image is accepted as input. Then it is passed through convolution and pooling (subsampling) layers for feature extraction. The model learns basic shapes like lines and edges in the beginning layers and gradually learns more complex shapes as it gets closer to the output layer. There are fully connected feed forward layers to classify different digits before the end. The final output layer is a Softmax layer with 10 neurons because it wants to classify digits: 0 to 9. . 16

1.7 Long-term recurrent convolutional network (LRCN) for video clip activity description. LRCN processes the (possibly) variable-length visual in-put (left) with a CNN (middle-left), whose outputs are fed into a stack of recurrent sequence models (LSTMs, middle-right), which finally produce a variable-length prediction (right). 16

1.8 Comparison of standard RNN and FCRNN. The main difference lies in the way of deciding the next state. The variables in red correspond to the parameters that need to be trained from scratch. In RNN, weight matrix associated with previous state, current state and bias at current state are to be learned through training. In FCRNN, only the weight matrix associated with previous state is to be learned from scratch since it works on a pretrained model. 17

1.9 Workflow of 'Show, Attend and Tell' system for neural image caption generation 18

1.10 TAGM first employs an attention module to extract the salient frames from the noisy raw input sequences, and then learns an effective hidden representation for the top classifier. The wider the arrow is, the more the information is incorporated into the hidden representation. The dashed line represents no transfer of information. 18

1.11 Performance of dual attention network (DANet). It appends two types of attention modules: spatial dimension (green) and channel dimension (blue), in order to model semantic interdependencies in those two dimensions. In our work, we use the attention calculation technique as shown in the attention modules. 19

1.12 Pointnet is a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks. 20

2.1 Block diagram of our proposed method to detect peptide features from LC-MS map of protein sample. First a CNN is trained as a 10-category classification problem. Then that is used to scan the LC-MS map through sliding window. While scanning, CNN outputs $z = 0$ if no feature is seen in input, and outputs $z = 1$ to 9 if a feature having corresponding charge is seen. After those outputs are processed though some heuristic methods, we get the feature table as shown in the right. Here, Id is assigned to each feature. For each feature, there is a list of isotopes. For each isotope, we show the m/z location, and RT time range. We also show the charge z of the feature and its total intensity as AUC. 27

2.2 Architecture of our proposed Convolutional Neural Network. It takes input a $[15 \times 211]$ dimension image (scanning window). Then it is passed through four convolutional layers (without pooling), one fully connected layer, and a Softmax output layer having 10 neurons, since this is a 10-category classification problem. 28

2.3 Calculation of specificity 33

2.4 Generation of training data: (a) Generate positive samples by placing a [15 x 211] window over the feature, such that, first isotope of the feature is centered at [0,6] pixel of the window; (b) Generate negative samples by translating the window around the features, such that, NO feature starts within the $[0, 0]$ to $[0, 6]$ pixels of the window 33

2.5 CNN detections recorded in hash table 36

2.6 Break within a peptide feature 36

2.7 Merging of RT Extents 37

2.8 Combine adjacent traces who are overlapped along RT axis 37

2.9 Selection of single m/z value for each isotope 38

2.10 The left most three shapes represent peptide feature but the last shape is probably noise, therefore ignored in our method 38

2.11 Condition between consecutive isotopes in a feature 39

2.12 List of detected peptide features 39

2.13 Visualization of Step 2 and Step 3: (A) Sample LC-MS map, (B) CNN detection in Step 2 by scanning over sample LC-MS map, (C) Corresponding records in hash table after the scanning, (D) $m/z \times RT$ plot for the detected multi isotope pattern, (E) In Step 3, the pattern is listed as a peptide feature in the feature table 40

3.1 Workflow of DeepIso to detect peptide features from LC-MS map of protein sample. In the first step, IsoDetecting module takes input a sliding window as $[M \times N]$ image and outputs the charge of the feature detected in the input. If the output is 0, it means no feature is seen. The scanning results are saved in hash tables and later used to generate a sequence of isotopes. That sequence is sent to the second module, IsoGrouping, to separate the adjacent features and discard noisy traces. The final results are saved in a feature table showing detailed information on the detected features. 43

3.2 Learning curve for DeepIso. The cross-entropy loss for training and validation data is shown for 130 epochs. We see on the left that the validation loss does not change anymore after epoch 100, although training loss keeps on decreasing. We see the validation sensitivity also does not improve anymore after epoch 100. That means the model converges at about epoch 100. In this plot we show the average sensitivity of features from charge 1 to 5 because we have a very low amount of feature for charge above 5. 45

3.3 Venn Diagram of feature-matched MS/MS identification by different tools. The blue area shows that DeepIso is capable of finding some peptide features not detected by other tools. 49

3.4 Network of IsoDetecting module. In the left we see some scanning windows going bottom to up. Then we show how a particular frame or window is passed through three convolution layers, 2 fully connected layers, one FC-RNN layer, and finally the Softmax output layer. The final output should be 0 to 9, indicating if the input frame has noise or feature having respective charge. 53

3.5 Training data generation for the 'IsoDetect' module 54

3.6 Intuition of scanning by IsoGrouping module on the sequences of features. 56

3.7 Network of IsoGrouping module. It shows how a frame in the input sequence is passed through 4 convolution layers, 2 fully connected layers, 1 RNN layer with attention gate, and finally Softmax output layer. The output decides about the feature boundary and also discards noisy frames in the input sequence. 57

3.8 Pseudocode of IsoGrouping module. The actual script is uploaded at Github repository. 58

3.9 'Adjacent Feature' problem 61

3.10 (a) A peptide feature with broken signals. RNN along the RT axis in the IsoDetecting module helps in detecting such features with broken signals; (b) Proper detection of overlapping peptide features by DeepIso model; (c) Adjacent feature case. Such features are used for fine tuning DeepIso. . . . 63

3.11 The effect of pooling layers in IsoDetecting module is shown. A peptide feature with charge 1 is shown in LC-MS map. When we use pooling layer to detect features, the isotope detections are wider, as shown in the right most image. But if we avoid using pooling layer, then the detections are thin and precise, as presented in middle image. 64

3.12 Matching vs epochs plot for validation LC-MS map 9_01 66

3.13 Intuitive image showing the effect of resolution along the m/z axis of LC-MS map: (a) Lower resolution merges the closely residing peptide features. For example, the 1st isotope of feature A and B are merged together. As a result, the monoisotope of feature A is missed by the model; (b) Higher resolution separates the first isotope of feature A and B. Therefore, IsoDetecting module can perform correct detection. 69

4.1 The workflow of our proposed model PointIso to detect peptide features from LC-MS map of protein sample. In 3D LC-MS plot we show a random scanning window in bold black boundary, enclosing two features. This region is further shown in the next image, labeled as 'Zoomed in Simplified View'. Here, two features A and B are shown using orange and green boundary, each having multiple isotopes (although smooth beta distributions of the isotopic signals are shown for simplicity, practically they are distorted due to noise). The corresponding point cloud version of this window is shown in the next image, labeled as 'Point Cloud Input'. Here the blue and white points correspond to the features and background points respectively. The 'Visualize Output' shows the PointIso predicted labels for the datapoints in that window. The datapoints labeled as '1' belong to feature A having charge 1. And the datapoints labeled as '2' belong to feature B having charge 2. The background or noisy datapoints are labeled '0'. 72

4.2 Detection percentage of identified peptide features by different tools for 12 samples is shown. We see that PointIso gives about 2% higher detection than all other software. Both settings $k = 2$ and $k = 6$ give almost similar performance. Maybe the reason is, smaller training set (as in $k = 2$) is good enough for achieving model convergence. 74

4.3 Observations on identified peptide features detected by only PointIso. (a) Venn diagram of identified peptide features detected by different algorithms from replicate 4 of sample 3. We show four algorithms to keep the Venn diagram simple. Here the the percentages are over MS/MS identified peptides of amount 10,000 approximately. The comparison with DeepIso is shown in Appendix D.2. (b) Most of the tools merge the low intensity feature marked with orange line with the bigger one marked with black line during pre-processing steps. (c) In this image, A and C are the actual features. Only PointIso detects these precisely. But other tools detect A, and instead of C, they report B by mistake because of missing the monoisotope (merging it with A). (d) Closely residing and overlapping features, like feature B in blue and C in orange rectangles are sometimes missed by other tools as well, although detected by PointIso. (e) Feature with broken signals are detected by our model, but discarded by other algorithms. 78

4.4 Need for attention mechanism for improving the sensitivity with *non-overlapping* sliding window. (a) Two non-overlapping sliding windows are shown in the top, and the corresponding output is shown in the bottom. It is applying a segmentation network without any surrounding knowledge, therefore, it causes missing of the feature isotopes, as shown by cross marks. (b) Surrounding regions of a target window. Among the eight regions, only four regions seem important according to our experiments: r_1, r_2, r_3, r_4. (c) 2D bi-directional RNN to flow the surrounding information towards the target window in center. (d) Attention coming from surrounding regions over the marked datapoints of the target window. There are partially seen features in those marked regions, and for segmenting those data points, IsoDetecting needs to consider the influence or attention coming from the surrounding regions. 81

4.5 Detection performance comparison between two candidate solutions and illustration of samples used for fine-tuning. Illustrations in (a) and (b) show the comparison between attention-based mechanism and bi-directional 2D RNN. The orange rectangle is showing the target window. For each target window, detections by attention mechanism and bi-directional two-dimensional RNN are shown next to it, pointed by arrow signs. In the target window, we see some vertical traces in each of the circle markers. In (a), we see that those separate traces are detected separately by attention mechanism, but merged by bi-directional 2D RNN. In (b), only blue features are detected by bi-directional 2D RNN (which is wrong), but both blue and green features (partially seen in the target window) are detected by attention mechanism. (c) When isotope lists are passed to the IsoGrouping module with wrong frames (dotted rectangles) because of the wrong charge ($z = 4$) detected by the IsoDetecting step, it results in discarding this whole group of frames as noise due to the inconsistency (blank frames) observed. (d)Two features having same charge z are adjacent (not overlapping) such that the distance between last isotope of feature 1 and first isotope of feature 2 is equal to $\frac{1}{z}$, i.e., the inter isotope distance of the features. We name such cases as 'adjacent feature' problem. (e) Feature like noisy signals just beside the actual isotopic signals. 83

4.6 The network of IsoDetecting module. This network goes through three steps, finding the local features (please see the original PointNet paper for T-Net and other technical details), global features, and point features respectively of the given target window. The number of layers and neurons in the Multiple Parceptron Layers (MLP) and Fully Connected Layers (FCL) is determined by experiments and mentioned in the figure. Point features of the target window are then diffused with features of surrounding regions based on their attention or influence over the target window (calculation of $Attention\_left$ and others are shown in the next figure). Finally, the diffused features are passed through four Multi-Layer Perceptron (MLP), and the Softmax layer at the output provides the final segmentation result. . . 86

4.7 Flowchart of attention calculation in the IsoDetecting module. Here, 'T' and 'L' means target window and left window respectively. This particular flowchart is intended to find out the attention or impact of the left region over the datapoints of the target window. Exactly similar approach is followed for other surrounding regions as well and finally, all are diffused with the $Point\_Feature_{target}$ by addition. 87

4.8 (a) An area in LC-MS map is shown. A target window in bold black rectangle containing a feature in blue color is shown. This target window and its four surrounding regions (as shown in Figure 4.4(b)) forms a positive training sample for IsoDetecting module. We also slide the target window within the region showed by arrow signs to generate training samples holding the peptide feature in different locations of the target widow (some are shown by dotted rectangles). (b) One negative training sample containing feature like noises is shown the topmost rectangle. In this region of LC-MS map the traces look like a feature having three isotopes. However, those are actually noisy signals as shown in the middle rectangle. And if we do not provide such training samples, then PointIso label the respective datapoints as positive class and report it as a feature, as shown in bottom rectangle. That is why we should provide such samples during training. (c) Some region in LC-MS map containing only arbitrary noises is selected for generating negative training samples. (d) Some region in LC-MS map containing blank area is also selected for generating negative training samples. 88

4.9 The network of IsoGrouping module. It starts with two convolution layers to fetch the graphical features from the input frame. Then we concatenate the intensity of the isotopic signal (area under the beta distributed (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) isotopic signal) with it through an embedding layer of neurons (frame context). Then this is passed through two fully connected layers having sizes 16 and 8. This gives us the 'frame feature' of the input frame. We perform the same for five consecutive frames and then concatenate the 'frame feature' of those altogether. Then one layer of convolution is applied to detect the combined feature from all the frames. The resultant features are passed through two fully connected layers (size 128 and 64) to decide whether this is a noise or potential feature. This probability is also used to activate a scaling neuron, that feeds the charge into the network through proper scaling. The scaled charge is concatenated with the latest layer output (size 64) and passed through two fully connected layers. Finally, the Softmax output layer at the end classifies the sequence. We include pooling layers after the first and second convolution layers. We apply the ReLu activation function for the neurons. The dropout layers are included after each fully connected layer with a dropout probability of 0.5. The other network parameters are mentioned in the figure. 92

4.10 (a) A sequence of five frames, first three holding three isotopes. Each frame has dimension $[15x3]$, covering 15 scans along the RT axis and 0.03 m/z along the m/z axis (each pixel represents 0.01 m/z). We filter out isotopic signals from the background by taking the intensity within the range of 2 ppm before and after the peak intensity m/z value, and 7 scans before and after the peak intensity RT value. The signal is left aligned with the frame. (b) A sequence of five frames which is generated from noisy areas in LC-MS map. (c) A sequence of five frames which is generated from blank areas in LC-MS map. (d) A sequence of five frames where the initial frames are holding noisy traces. 93

4.11 (a) A peptide feature with three isotopes and charge 2 is shown. But if this is passed to the IsoGrouping module with wrong frames (dotted rectangles) because of the wrong charge ($z = 4$) predicted by the IsoDetecting step, it results in discarding this whole group of frames (by predicting it as noise) due to the inconsistency (blank frames) observed. (b) We see the adjacent feature problem. (c) We see one training sample prepared for IsoGrouping module to solve the adjacent feature problem. (d) In the topmost rectangle we see isotopes of a peptide feature in black traces and some secondary signals as well. The middle rectangle is showing the true labeling of the datapoints using two colors: grey means negative class and black means positive class. The primary signals are detected by PointIso as shown in the bottom rectangle. However, the seconadary signals are also predicted as positive class by PointIso. But this is wrong. Datapoints in those regions should be predicted as negative class by IsoDetecting module. So we select such peptide features for fine tuning. 97

5.1 How peptide feature detection helps in biomarker detection and feature intensity distribution. (a) Disease biomarker discovery. (b) Intensity distribution of identified (orange) and detected (blue) peptide features. 103

5.2 Comparison of mass, m/z, and RT distribution of detected features (blue) and identified features (orange) for different tools. Good alignment between the blue and orange distribution indicates a better probability of detected features being the true feature. 105

5.3 How to find protein quantity. After a peptide is identified from MS2 data (fragment ion spectra), that peptide is mapped to its corresponding peptide feature in the MS1 data to get its total intensity, which is used for quantity calculation of that peptide. 106

5.4 (a) Theoretical injected amount of spiked peptides (the figure is adapted from the paper by Chawade et al. [9]); (b) MASCOT identification of spiked peptides. It shows that sample 1 to sample 7, mostly potato peptides are identified by MASCOT due to very low concentration of human peptides in those samples. Similarly, mostly human peptides are identified in sample 6 to sample 12, due to very low concentration of potato peptides in those samples. 106

5.5 (a) Alignment of peptide features over multiple replicates. (b) Same peptide sequence ANLYGIGEHTK is mapping to different peptide features, having charge 2 and 3. We may take the sum or maximum of those peptide features' intensity to get the quantity/abundance of this particular peptide. (c) Peptide abundance list for a sample LC-MS map. 108

5.6 Abundance comparison for a particular peptide (shown in black box) among multiple samples. Since its abundance increases, therefore, slope in the top is positive. 109

5.7 (a) Distribution of potato, human, and background peptide concentration slopes over 12 samples by PointIso only; (b) Distribution of human and potato peptide concentration slopes are compared among multiple software. It indicates that PointIso, like all other software, is potential for LFQ. . . . 110

5.8 Illustration of 4D peptide features. For simplicity, we are not showing the Intensity axis. (a) In the left, we see a usual peptide feature in $[m/z \times RT]$ plane. This same feature can get separated into two different features if we consider the additional $\frac{1}{k0}$ dimension, as shown right to it. (b) Scanning windows of PointIso cover full range of $1/k0$ while scanning the MS1 TimsTOF data. 111

5.9 Workflow of MaxQuant for processing 4D peptide features. The space in $(RT \times \frac{m}{z} \times \frac{1}{k_0})$ is sliced into multiple $(RT \times \frac{m}{z})$ planes. Then for each plane, it applies the conventional MaxQuant algorithm. After that, the overlapping detection areas are clustered across slices to obtain a feature in $(RT \times \frac{m}{z} \times \frac{1}{k_0})$ space again. 112

5.10 Detection Percentage of identified peptide features for different tools. We see that Pointiso is giving about 86% sensitivity, about 4% to 6% higher than MaxQuant and PEAKS. 116

6.1 Unsupervised pretraining of the transformer model 121

6.2 Supervised refining of the transformer model 122

B.1 Flowchart of attention calculation in IsoDetecting module. This particular flowchart is intended to find out the attention or impact of left region over the datapoints of target window. Exactly similar approach is followed for other surrounding regions as well and finally all are diffused with the $Point\_Feature_{target}$ by addition. 142

D.1 Comparison of mass, m/z, and RT distribution of detected features (blue) and identified features (orange) for different tools. 153

D.2 Venn diagram of identified peptide features detected by four algorithms (PointIso, DeepIso, Dinosaur, and MaxQuant) for replicate 1 of sample 10. 154

List of Tables

2.1 Class sensitivity (%) for six LC-MS maps (A to F) for type X data. Type X represents the features detected by PEAKS which may or may not map to a De Novo or database sequence. For charge 7 to 9, there were very low amount of training samples. Therefore, those features are not learned well, showing comparatively poor sensitivity. 30

2.2 Class sensitivity (%) for six LC-MS maps (A to F) for type Y data. Type Y represents the features detected by PEAKS which also map to a De Novo or database sequence. There were no type Y feature belonging to charge 7 to 9. 31

2.3 The class sensitivity (%) of our model under different AUC range for type X and Y. Here, AUC is the total intensity of peptide features or area under the curve of isotopic signals in peptide features. 32

2.4 Average class sensitivity of training and validation data. Please note that, for charge z=1 and z=6 to 9, the sensitivity is poor because the model cannot learn those features due to low amount of training samples. 35

3.1 Class distribution of samples in our dataset consisting of 57 LC-MS maps. Amount of samples from charge 1, 2, 3, and 4 is 10.96%, 58.04%, 28.84%, and 1.96% respectively. Samples from the rest of the charges have less than 1% amount. 45

3.2 Class sensitivity and precision of IsoDetecting module and amount of samples for training and validation. The training set for class 5 has some duplicated samples. Training sets for classes 6 to 9 have augmented (oversampling) and duplicated samples. However, very low amount of original features from these classes (5 to 9) results in lower sensitivity for these classes due to over-fitting. The amount of samples from class 0 depends on our choice, and we keep this higher than other classes because the LC-MS map is very sparse. The validation set does not contain any duplicated data, and there is no overlapping between the validation dataset and the training dataset. 46

3.3 Class sensitivity of IsoGrouping module on training set and validation set. Please note that it does not relate to the charge/class $z = 0$ to 9. It shows how well the IsoGrouping module is able to group the isotopes to form the feature (whatever the charge is), and recognize the noises. 47

3.4 Confusion matrix produced by IsoGrouping module on validation dataset. The diagonal values, e.g. [C, C] represent the sensitivity for class C. We say a feature is misclassified as class A when the monoisotope (first isotope) or all of the isotopes are missed, i.e., the feature is thought to be noise by mistake. The value of [C, A] indicates what percentage of features with three isotopes are either misclassified as noise, or monoisotope is missed. [C, B] indicates the percentage of features which actually have three isotopes but the third one is missed, and only first two are combined together. Similarly [C, D] shows for what percentage of three isotope features, IsoGrouping module finds ONE additional isotope at the end. 47

3.5 Percentage of high confidence MS/MS identifications matched by feature list produced by different algorithms. 48

3.6 Pearson correlation coefficient of the peptide feature intensity between DeepIso and other tools. A coefficient value of close to 1 indicates that the models in comparison have a good linear correlation with each other. That means both of them may give similar label-free quantification results. 50

3.7 Approximated running time of different algorithms. Here the platform used for OpenMS and DeepIso did not have support for running Windows application of MaxQuant and Dinosaur. So we used different machine for running those. 51

3.8 Models used for Ensemble 59

3.9 IsoDetecting module give better validation sensitivity with FC-RNN network than attention-gated RNN. 65

3.10 Performance of IsoGrouping module in different stages of the development (based on validation dataset). Here, the initial model needs about 430,000 parameters to learn, whereas the more effective version shown in the third row needs only 167,000 parameters to learn. This is because we use max-pooling with stride $[2 \times 2]$. This pooling lets the model focus on the important features. Achieving higher sensitivity with a smaller model also demonstrates that simple and concise model works better than an unnecessarily big model. 65

3.11 Improvement of class sensitivity of IsoDetecting module for charge states 6 to 9 with increasing amount of training samples. We do not bother for further improvement (by including more data from different but similar dataset) since most of the peptide features generally appear in LC-MS map with charge states < 6. Here the validation set does not contain duplicated data and there is no overlapping among the training set and validation set. . . 67

4.1 Detection Percentage of MS/MS identified peptide features by different methods. PointIso gives about 98% sensitivity (2% higher than other tools) when we match the detection to the high confident identifications in terms of m/z and RT, as shown in the first row. If we also match in terms of same charge z, then the sensitivity goes down for all the software by about 1%, but PointIso is still giving better sensitivity, as shown in the second row. In the third row, we show the model sensitivity taking into account all the MS/MS identified peptide features irrespective of their score, and PointIso is consistantly showing better performance than other tools. 75

4.2 Detection Percentage of MS/MS identified spiked peptides by different methods. Matching is performed by comparing the monoisotopic peak (m/z, RT) and charge z of detected features with the identified spiked peptides. PointIso is giving 3%-4% higher detection of human peptides and 5%-6% higher detection of potato peptides. 76

4.3 Approximated running time of different algorithms. Here the platform used for OpenMS, DeepIso and PointIso did not have support for running Windows application of PEAKS, MaxQuant, and Dinosaur. So we used different machine for running those. 79

4.4 Performance of PointIso in different developmental stages (based on validation dataset). 79

4.5 Class distribution of peptide features in our dataset consisting of 57 LC-MS maps. 87

4.6 Amount of samples for training and validation of IsoDetecting module in PointIso model. Because of inadequate training data for features with charge states 5 to 9 as mentioned in Table 4.5, we had to apply data oversampling and augmentation to increase training samples from these classes. The amount of samples from class 0 depends on our choice. We chose the amount so that the total number of datapoints from this class is higher than others because the LC-MS map is very sparse. The validation set does not contain any duplicated data, and there is no overlapping between the validation dataset and the training dataset. 90

4.7 Class sensitivity of the IsoDetecting module for fold 1 in the 2-fold cross validation experiment. That is, dilution sample 10, 11, 12 are used for training, sample 9 is used for validation, and the rest are used for testing. We show two cases, average and best case in terms of feature detection difficulty. The best case occurs when the feature is left aligned with the target window boundary, e.g., feature 'A' in Figure 4.4(a). The average case means the target window may have the features at any location of the window, may contain any number of features and the features may be partially or fully seen and may be overlapping as well. Due to the lack of variance in training data for charge states 6 to 9, the model's validation sensitivity does not go up high for these classes. However, since most of the peptide features appear with charge states < 6, lower sensitivity for them does not impact the overall performance. The validation set does not contain any duplicated/over-sampled data and there is no overlapping between validation samples and training samples. 91

4.8 Class sensitivity of the IsoGrouping module on the training set and validation set. The output is $i = 0$ to 4, where $i = 0$ means that no feature starts in the first frame, so skip it. Output $i = 1$ to 4 means, there is a feature starting in the first frame, and it ends at $(i + 1)^{th}$ frame. When output $i = 4$, it means there might be more isotopes left. So we run another round of processing over the rest of the isotopes of the same cluster or sequence. Therefore, although our network process five frames at a time, if the feature has more than five isotopes, those can be found by overlapping rounds. . . 94

4.9 Confusion matrix produced by IsoGrouping module on validation dataset. The diagonal values, e.g. [C, C] represent the sensitivity for class C. We say a feature is misclassified as class A when the monoisotope (first isotope) or all of the isotopes are missed, i.e., the feature is thought to be noise by mistake. The value of [C, A] indicates what percentage of features with three isotopes are either misclassified as noise, or monoisotope is missed. [C, B] indicates the percentage of features which actually have three isotopes but the third one is missed, and only first two are combined together. Similarly [C, D] shows for what percentage of three isotope features, IsoGrouping module finds ONE additional isotope at the end. 95

A.1 Class sensitivity for different weighting mechanisms. We compare the candidate weighting mechanisms based on the sensitivity for the best case scenario, i.e., when feature is aligned with the left boundary of the scanning window (e.g., feature A in Figure 4(a)), with high abundant features, i.e., features having charge, z = 1, 2, 3, 4. Please note that, although the sensitivity of negative class (z=0) is comparatively lower for our chosen criteria, however, it does not imply that it reports many false positives. Although the datapoints which are very close or adjacent to the real signal, are sometimes predicted as positive points, but in general the negative class has higher class sensitivity than all others as presented in the main manuscript. 138

A.2 Different techniques of absorbing surrounding information and corresponding class sensitivity of IsoDetecting module. We define the class sensitivity of a scanning window as the number of datapoints from class z (0 to 9) detected correctly out of total number of datapoints in a scanning window. To evaluate candidate solutions we use the class sensitivity of high abundant features (charge z = 1,2,3, and 4) in a *average case* scenario. Average case means the scanning window might contain any number of features, they may appear at any location of the window, they might be partially or fully seen, and might be overlapping as well. We see that the DANet inspired attention based mechanism works better than other techniques. 139

A.3 Better learning (higher class sensitivity) by IsoDetecting module with upgraded architecture. 140

Chapter 1

Introduction

Proteins are the main workhorses responsible for biological functions and activities in cells, tissues, or organisms. Accurate protein profiling brings us closer to the cell phenotype to understand different cell types and developmental stages. The proteins present in readily accessible biofluids that are diagnostic or prognostic of a disease are called 'biomarkers'. Liquid chromatography with tandem mass spectrometry (LC-MS/MS) based proteomics provides the relative differential protein abundance between healthy and disease-afflicted patients for comparative analysis. In addition, it provides information for molecular interactions, signaling pathways, and biomarker identification to serve drug discovery and clinical research. The latest advanced LC-MS technologies generate massive amounts of data with a very high scan speed and resolution, which is almost impossible to interpret manually. Therefore, computational models for speeding up the data analysis and automating the feature extraction from the massive dataset are highly advantageous.

Deep Learning [38], an approach in artificial intelligence, has brought a new era with its groundbreaking results in computer vision, natural language processing, as well as many multidisciplinary research, such as clinical data analysis and multi-omics study. Deep learning involves computational models containing multiple processing layers to learn data representations with multiple levels of abstraction. Deep learning discovers complex structures in large data sets using the backpropagation algorithm to learn how a machine should change its internal parameters that are used to compute the representation in each layer using the representation in the previous layer. Convolutional Neural Networks (CNN) have brought breakthroughs in image and video processing, whereas Recurrent Neural Networks (RNN) have significant contributions to sequential data processing such as text and speech. Auto-Encoders, Graph Neural Networks (GNN), and Generative Adversarial

Networks (GAN) are also gaining popularity for unsupervised and semi-supervised learning.

The outstanding performance of deep learning on object recognition opens a new frontier in the domain of bioinformatics. As a continuation, we investigated the power of deep learning in analyzing proteomics data. **To be specific, our target problem is to develop the first deep learning based model to address the crucial step of the LC-MS/MS analysis workflow: peptide feature detection along with charge state and intensity from a three-dimensional LC-MS map. Our developed model is free from manual input of parameters and provides higher sensitivity than other exisiting tools.** Peptide feature detection is directly applicable in disease biomarker discovery, which can be protein biomarkers or neoantigens. Neoantigens are some short-length peptides generated on the surface of cancerous cells or tumors, and discovering those can help in cancer immunotherapy. Peptide feature detection also serves in label-free quantification (LFQ), a technique for measuring the relative abundance of proteins in multiple samples. To detect peptide features, we first start with a naive CNN, along with some heuristics [89]. After receiving promising results from the initial study, we approached further by combining CNN with RNN and proposed DeepIso [87], the first deep learning model to automate the peptide feature detection. We avoid heuristic steps in DeepIso, and all the necessary parameters were learned through training on an appropriate dataset. Next, we proposed PointIso [88], a point cloud based model along with an attention mechanism, to address the same problem, but three times faster, more robust, and capable of handling arbitrary precision datasets. We also adapt our model for four-dimensional peptide feature detection by bringing few architectural changes [88], which illustrates the generic nature of our model. Then we evaluated the quantitative accuracy of our deep learning model so that it can be applied in the downstream pipeline of label-free quantification. Besides peptide feature detection, we also assess the quality of peptide features through statistical analysis so that our model can be considered reliable in biomarker discovery [88]. We are the first to attempt a deep learning approach to address the peptide feature detection problem as per our knowledge. Our problem of peptide feature detection is very different from general object detection problems and more challenging in various aspects. Therefore, off-the-shelf object detection or classification models do not work in our context, and we had to develop a separate deep learning model for addressing our particular problem. Before further discussion on our research works, we would like to introduce the basic concepts of LC-MS/MS analysis, as well as, some deep learning terminologies in the following sections. We will start with the discussion on the LC-MS/MS workflow of proteomics data analysis in Section 1.1. Then a brief discussion on existing approaches of peptide feature detection in Section 1.2, and its application in

Section 1.3. In Section 1.4, we will provide brief introduction of some deep neural networks which we deem beneficial for our research. We will also discuss some existing deep learning literature in the context of proteomics data analysis. At the end, there will be a summary of our accomplished research works in Section 1.5, which are elaborated in the other chapters.

1.1 LC-MS/MS Analysis Workflow

Protein identification and quantification are the two fundamental tasks in a proteomics study. Liquid chromatography coupled with tandem mass spectrometry (LC-MS/MS) is the current state-of-the-art technology for protein identification and quantification [2]. Proteins are first digested into smaller peptides by various sequence-specific enzymes, e.g., trypsin. Then this protein digest or peptide sample is analyzed by LC-MS/MS instruments. The procedure starts with separating the peptides in the LC phase, then ionized and analyzed in the first MS phase or MS1. The output of MS1 is called **LC-MS map/data** or **MS1 data** that contains the peptide features. The peptide features are some multi-isotopic patterns in 3D space, where the axes are mass over charge (m/z) ratio, Retention time (RT), and Intensity (I) of the isotopic signals. **Peptide features** may also be called **LC-MS peptide features** since they are found from LC-MS map. The high abundant peptide ions or precursor ions are sent to the next MS, i.e., MS2, for fragmentation. Therefore, MS2 generates **fragmentation spectra** or **MS/MS spectra**, also called **MS2 data**. MS2 data allows to identify the peptide sequences [64]. Identified peptides are also called **MS/MS identified peptides**. As a part of label-free protein quantification, we have to trace them back to the corresponding features in MS1. Therefore, a typical analysis workflow of LC-MS/MS data, e.g., MaxQuant [13], OpenMS [65], PEAKS Studio [84], etc., often go through the following steps: peptide feature detection from LC-MS map, peptide identification from MS/MS spectra, peptide quantification by mapping the identified peptides to the corresponding features (which were detected at the beginning of the workflow), and protein profiling by matching the peptides to their parent protein. We briefly explain these key steps in the following sections.

1.1.1 Peptide Feature Detection

We will first explain how to interpret the LC-MS map since this is the first raw data generated by the instrument. Each LC-MS experiment may be repeated multiple times, where each experiment runs for up to 120 minutes. In LC-MS map, the Y-axis shows the

Retention Time (RT), i.e., the time when the peptide ion is eluted (i.e., extracted) from the mixture, and the X-axis shows the mass over charge (m/z) ratio of those peptide ions in sorted order. We can visualize one run as the 2D plot (leftmost image) of Figure 1.1. So the peptide feature is a multi-isotopic pattern formed by different molecular isotopes, e.g., carbon-12 and carbon-13, of the same peptide. For instance, the red box in the zoomed-in view of LC-MS map (middle image) in Figure 1.1 presents a peptide feature with charge 1, and it has four isotopes represented by the vertical traces. The charge of a feature depends on the inter-isotope distance of that feature. If a feature has charge z, its isotopes are $1/z$ distance apart. A typical range of peptide charges is 1 to 3. However, theoretically, a charge over 3 or 10 is possible, although rarely found in nature. In my research, I choose to classify features having charges 1 to 9 based on the discussion with our collaborators at Bioinformatics Solutions Inc. The LC-MS map is essentially a 3D plot where the intensity (I) of the peptide ion signals is the third dimension, as shown in the rightmost image of Figure 1.1. We see that the signals produce a beta distribution shaped curve (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$), which is one of the key features of identifying peptide features (although it is often distorted due to noise). So we can observe that the peptide feature detection is a pattern recognition problem that is traditionally handled using various heuristic steps and simple machine learning techniques, e.g., centroiding, curve fitting, clustering, etc. Detecting multi-isotopic patterns in an LC-MS map is a challenging task due to the overlapping peptides, multiple potential charge states for the same molecule, and intensity variation. Moreover, a single LC-MS map may have a gigapixel size containing thousands to millions of peptide features. Deep learning model for peptide feature detection should learn the following basic properties of peptide feature [7]:

Peptide Feature Properties

1. The isotopes in a peptide feature are equidistant along m/z axis, and the distance is $1/z$ unit. For charge $z = 1$ to 9, the isotopes are respectively 1.00 m/z, 0.500 m/z, 0.333 m/z, 0.250 m/z, 0.200 m/z, 0.167 m/z, 0.143 m/z, 0.125 m/z, and 0.111 m/z distance apart from each other.

2. Intensity signal of an isotope is beta distribution (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) if looked at [RT $\times$ Intensity] plane, but normal distribution if looked at [m/z Intensity] plane, as shown in Figure 1.2(a). This distributions are the characteristics of the features.

3. The isotope having highest intensity in a peptide feature is called precursor ion and that is the first isotope in a feature having charge 1 to 4. For charge 5 and up, the

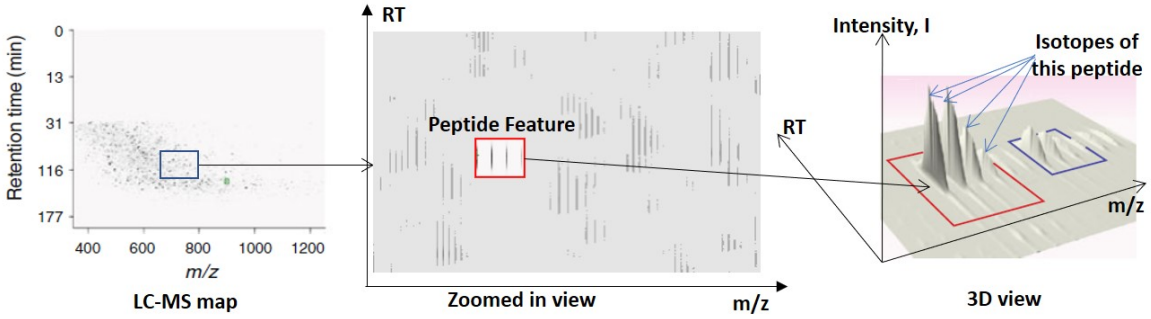
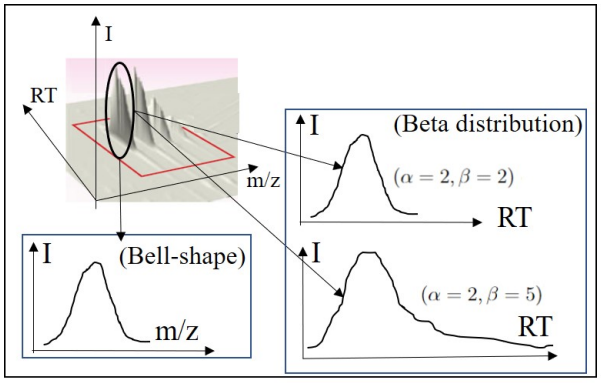
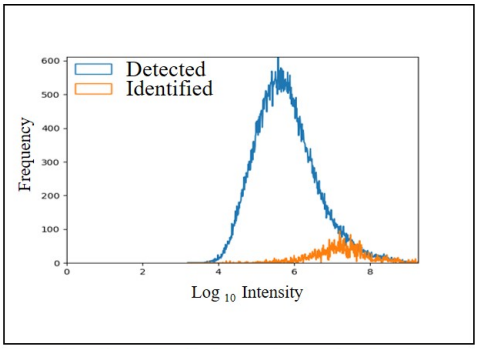


Figure 1.1: The m/z vs Retention Time (RT) plane of a LC-MS map is shown in the leftmost image. A small area is zoomed in next, where we can see many groups of vertical lines/traces. Each of them is called peptide feature. We mark a peptide feature by rectangle and see the detailed 3D view of this peptide feature in the next image.



(a) Shape of the isotopic signal



(b) Intensity distribution of peptide features in an LC-MS map (detected & identified)

Figure 1.2: Shape of isotopic signal and feature intensity distribution. (a) A peptide feature having four isotopes is shown in the left. The intensity signal of each isotope forms a beta distribution shaped curve (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) if watched at $[RT \times I]$ plane. Again, if looked at $[m/z \times I]$ plane, it forms a bell shaped curve.

middle isotope can be the precursor ion. However, we always represent a peptide feature using the m/z and RT of the monoisotope, i.e., the first (or leftmost) isotope of the feature.

4. Peptide features often overlap with each other along m/z and RT axes.

5. Intensity of a feature is calculated by adding up the total intensity of all the isotopes in that feature. Intensity distribution of the features in an LC-MS map forms a normal distribution as shown in Figure 1.2 (b) with blue color. Only high abundant peptide features are sent to MS2 for MS/MS identification. That is why the intensity distribution of MS/MS identified peptide features (orange color) wedge along the high-intensity side of the blue distribution, as shown in the figure.

6. Sometimes there are some noisy signals right besides the actual signals of the peptide features (shown later in Figure 4.5(e)). In our research, we denote those noisy signals as **'Secondary Signals'** and the actual signals as **'Primary Signals'**. The secondary signals look like a shadow of the primary signals and are good to be ignored by the model.

1.1.2 Peptide Identification

The precursor ion is the highest intensity peak isotope of a peptide feature present in the LC-MS map. For example, the left most isotope in the peptide feature shown in Figure 1.1. In Data Dependent Acquisition (DDA), the instrument performs a stochastic MS/MS sampling over the peptide features to select the highly abundant features and the respective precursor ions are sent to MS2 for fragmentation. In case of Data Independent Acquisition (DIA), usually all precursor ions within a particular range of m/z and RT are sent to MS2 for fragmentation. Then MS2 generates fragment ion spectra which is called MS/MS spectra. One such spectrum looks like Figure 1.3. Then the task of peptide identification is to reconstruct the amino acid sequence of a peptide given this MS/MS spectrum and the peptide mass. For a given peptide sequence, the B ions are the product when the charge is retained on the N-Terminus (i.e., at the beginning of the sequence) and the Y ions the product when the charge is retained at the C-Terminus (i.e., at the end of the sequence). If a protein or peptide database is given, this problem becomes a sequence database search where one could first filter candidate sequences based on the peptide mass and then select the optimum sequence that best fits the fragmentation patterns in the spectrum. However, this approach fails to recognize novel peptides since it can only match to existing sequences in the database. So, for finding novel peptides people use de novo peptide sequencing in which a peptide amino acid sequence is determined from MS2 data (MS/MS spectra), e.g., PEAKS software [43]. It involves pattern recognition and global optimization with various forms of dynamic programming that have been developed over the past decade [71].

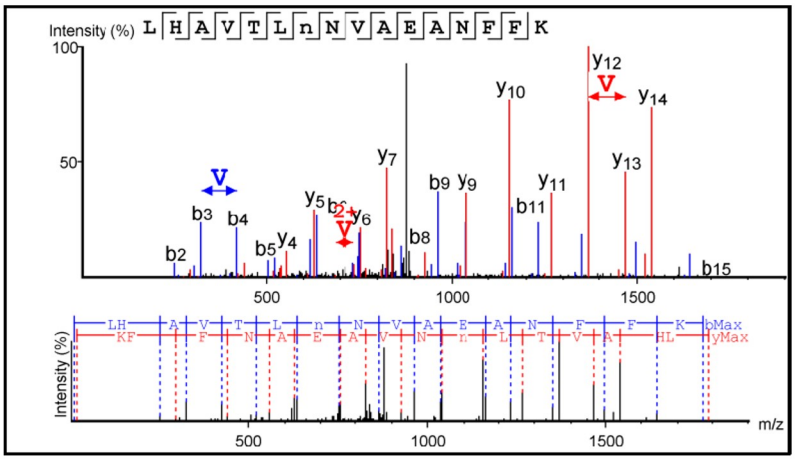


Figure 1.3: Peptide sequencing from an MS/MS spectrum. For a given peptide sequence, the B ions are the product when the charge is retained on the N-Terminus (i.e., at the beginning of the sequence) and the Y ions the product when the charge is retained at the C-Terminus (i.e., at the end of the sequence). Each amino acid is identified by the mass difference between neighbouring peaks (B or Y) in the spectrum. The peptide sequence is predicted as the optimum one that best fits fragment ions in the spectrum. The figure was generated from the tool PEAKS Studio.

1.1.3 Peptide Quantification

Measuring relative protein abundance across multiple patient samples is particularly important in proteomics-based biomarker discovery. The intensity of peptide features is used in peptide quantification. This is estimated by summing up the calculated area under the intensity signals produced by the isotopes of a peptide feature. Currently available MS platforms for quantitative proteomics can be categorized into following three classes [31]:

- Identity-based methods that rely on proteolytic digestion of proteins to peptides with analysis by LC-MS/MS. They find the intensity of only those peptide features which are identified by MS/MS analysis, and that is done by counting the matched MS/MS spectra. Identity based methods might miss lower abundant proteins since MS/MS sampling for identification usually favors high abundance peptides (hence proteins).

- Pattern-only methods focus on production of Mass Spectrometry (MS) derived protein patterns. However, the main job of identifying the peptides and proteins that generate the pattern is often difficult or impossible using these methods.

- Hybrid identity/pattern-based methods use peptide-derived LC-MS map from FTMS or Orbitrap mass spectrometers with very high resolution and mass accuracy. Because of the high resolution it can match the peptide identification result to corresponding LC-MS peptide features successfully, thus resolves the problem associated with only pattern based methods. Therefore the abundance of the identified peaks allows large scale relative quantification of peptides among groups of samples, and statistical analysis for biological characterization or biomarker discovery.

Once the peptides have been identified and their abundances have been estimated, we can assign them to their parent proteins by different string matching techniques, e.g. De Bruijn graph [71]. Finally, various statistical calculations are performed to infer the protein abundance from the peptide abundance measured in previous steps.

1.2 Existing Methods of Peptide Feature Detection

Since we are particularly interested in peptide feature detection problem, we would like to discuss the existing literature and methods for addressing that problem. Traditional methods of detecting peptide features from LC-MS map depend on many parameters for applying different heuristic steps and simple machine learning techniques involving a high level of feature engineering than feature learning. Moreover, none of them relies on deep learning to automatically find the appropriate parameters from the available LC-MS map. MSight [50] generates images from the raw LC-MS map file for adapting the image-based peak detection. CentWave [66] identifies important centroids and then the centroids are collapsed into a one-dimensional chromatogram, and wavelet-based curve fitting is performed to separate closely eluting signals. In MaxQuant [13], peaks (isotopic signals) are detected by fitting a Gaussian peak shape, and then the peptide feature is found by employing a graph theoretical data structure. AB3D [3] first roughly picks all local maxima peaks whose intensity is larger than a given threshold, then applies an iterative algorithm to process neighboring peaks of each to form peptide feature. TracMass [69] and Massifquant [11] use a 2D Kalman Filter (KF) to find peaks in highly complex samples. MSTracer [82] applies machine learning techniques (support vector machine) for peptide feature detection. Dinosaur is proposed by Teleman et al. [67] where the workflow of feature finding involves centroiding on LC-MS map, assembling centroid peaks into single isotope traces (hills), clustering of hills by theoretically possible m/z differences, and finally deconvolution of clusters into charge-state-consistent features. These steps are presented in Figure 1.4. In our experiments, this model performs as the second-best model (our model performs best)

and is publicly available for use. That is why we provide its workflow for the readers to have some context.

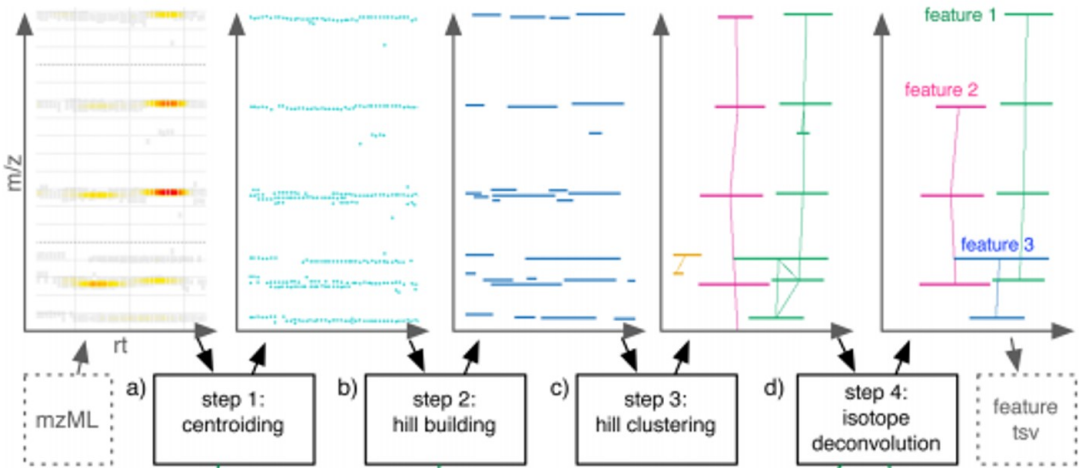


Figure 1.4: Workflow of Dinosaurs for peptide feature detection. It converts the raw LC-MS maps to mzML data format before starting the analysis.

Evaluation Criteria for Peptide Feature Detection Methods

Evaluation of peptide feature detection algorithms is challenging because manual annotation of peptide features is out of scope due to the huge size of the LC-MS maps [66]. As a result, when a software detects some peptide features, we cannot decide whether those are real features or false features just by comparing with some ground truth data on feature list. Some of the literature mentioned above prepare the ground truth data (for evaluating a new peptide feature detection software) by taking a common set of peptide features generated by multiple existing algorithm. They treat those as ground truth data/true positives, and detection outside those ground truth data as false positives, thus report performance in terms of several statistical measures, e.g., sensitivity, specificity, etc. For instance, CentWave [66] provides high sensitivity with high precision. AB3D [3] gives good sensitivity but poor precision. Massifquant [11] provides high sensitivity with high specificity. On the other hand, there are arguments supporting that we cannot label the peptide features as true positives or false positives in an LC-MS map perfectly. Because a multi-isotopic pattern in LC-MS map that is not detected as peptide feature, nor identified later (in MS2 phase) by peptide identification tools, might actually be a peptide feature

or merely a noise. We are not definite about their existence since no peptide feature detection tool, or identification tool is perfect. However, when some peptide features from MS1 phase are sent to the MS2 phase for identification (please recall that there are two mass spectrometers in tandem, MS1 and MS2, in the LC-MS/MS technology), then that identification result can be used for evaluation of peptide feature detection models. That identification result contains a list of peptides that are identified. Therefore their corresponding peptide features must exist in the MS1 phase or LC-MS map. In other words, if we can detect peptide features in LC-MS map (MS1 data) corresponding to those MS2 identified (also called MS/MS identified) peptides, we are detecting true features. Therefore, a higher percentage of detection should imply better performance. So the percentage of MS/MS identified peptides matched with the peptide feature list produced by different algorithms could be used for performance evaluation. For instance, Dinosaur [67] reports higher matching with MS/MS identified peptides than other existing tools. We also follow this evaluation technique in our experiments. Please note that peptide features may be detected that do not map to any identified peptides. We cannot simply mark them as false positives. We can not say anything about them.

1.3 Motivation for Peptide Feature Detection

Peptide feature detection has promising application in label-free quantification and disease biomarker discovery. These are explained in the following sections.

1.3.1 Lable-Free Quantification (LFQ)

Although LFQ is a different problem from feature detection, in the workflow of LFQ, we have to use the results of peptide feature detection. Therefore, our target problem has promising scope in LFQ. First, we will discuss why and when LFQ technique for protein quantification is superior over other quantification techniques. We have introduced some categories of peptide quantification techniques and briefly discussed their comparative performance in Section 1.1.3. All of those approaches can be further classified into stable isotope-based labeling methods or label-free methods. There are a variety of chemical tagging and metabolic labeling methods available for differentially labeling peptides with stable isotopic labels, for example, ICAT, SILAC, and iTRAQ. However, many clinical samples cannot be metabolically labeled. Even if its possible, the additional labeling strategies incur high cost. Therefore the LFQ is a preferred approach for analysing complex samples like cancer tissues. In addition, there is no limit on the number of samples that

can be compared, whereas a limited number of samples are comparable by label-based methods [12]. The use of LFQ has increased significantly in the past 15 years and has shown potential for identification and quantification of differentially expressed proteins in normal and diseased samples [4]. Therefore we will be focusing on LFQ only. Let us think we have two samples, A and B, and we want to know the relative abundance of various proteins in A and B using LFQ. We briefly demonstrate the steps involved in LFQ [47, 31] as follows:

- Clustering of identical peptide features across replicates: Each LC-MS experiment can be run multiple times, and the corresponding generated output or LC-MS maps are called replicates. The same peptide feature usually appears around the similar m/z and RT regions in all the replicates/LC-MS maps. So the first step is to match or align the identical LC-MS peptide features across those multiple replicates, taking into account m/z and retention time variation. Please see Figure 1.5 for an illustration. Although the m/z variation of the same feature across multiple replicates is minimal with the high-performance MS instrumentation, retention time can vary significantly across multiple replicates for a given peptide. Therefore the consistency of the retention time values over different replicates is a crucial factor and has led to the development of various alignment methods to correct chromatographic fluctuations.

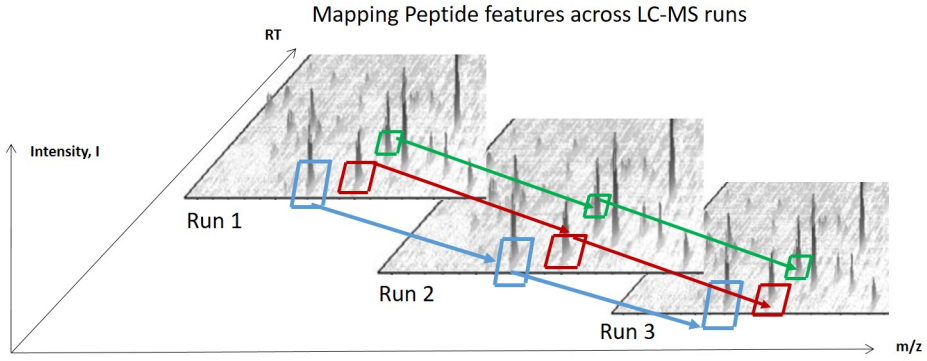


Figure 1.5: Label-Free Quantification: Peptide features are mapped across multiple replicates. For instance, the connected marked rectangles shown in this figure.

- Peptide feature intensity table for each sample: Each matched peak is represented by a $(m/z, RT, z)$ triplet and is constituted by peaks from one or more replicates that cluster together. Then the final result is an intensity table whose rows represent

features that are the matched peaks and whose columns represent replicates/runs. For each row, i.e., each feature, summation of peak intensities of all the replicates is calculated. In this way, we get the intensity of features for each sample.

- Quantitative Data Analysis: Now the detected and quantified peptides (features in the intensity table as described above) are identified by MS/MS peptide assignment [48]. Then these peptide sequences are assigned to their parent proteins by string matching. The matches are then sorted and grouped by parent protein. All abundance values can be normalized to the summed abundance of all the detected peptide features. Then the Log 2 transformation redistributes the abundance values observed into a normal distribution as shown in Figure 1.5(b). Ratios are computed in a log space by subtraction using average peptide abundances. Subsequently protein ratios can be computed by averaging all of the constituent peptide ratios and taking 2 to the power of this average.

The steps mentioned above will be discussed again using a specific protein sample later in Chapter 5, when we apply PointIso reported feature boundary for measuring the peptide intensity.

In addition to the peptide features identified by MS/MS peptide assignment as mentioned above, it is desirable to assess the identity of those remaining peptide features which do not readily match with any MS/MS identification but show interesting quantitative patterns across different samples, e.g., intensity variation (shown later in Figure 5.1). Because MS/MS identification is performed for highly abundant peptide ions, sometimes low abundant peptides are discarded. For example, suppose there is a peptide feature X detected at location (400.34 m/z, 32 min RT) with over 10^5 intensity in Sample A but not detected at all in sample B, and X is also not identified by MS/MS identification by database search. In that case, it is interesting to discover what X is. In that case, post-annotation of these peptide features is done and identified (i.e., protein name and peptide sequence) by targeted MS/MS, or comparing them with a large MS/MS repository, or de novo peptide sequencing, according to the assessment by Mueller et al. [47].

Label-free strategies are mostly applicable to the data acquired on mass spectrometers equipped with the time-of-flight (Tof), Fourier transform-ion cyclotron resonance (FT-LTQ), or Orbitrap mass analyzers. Because measurements on these MS platforms reach very high-resolution power and mass precision in the low parts per million mass unit range, this enables the extraction of peptide signals for specific analytes on the MS1 level and thus uncouples the quantification from the identification process and the acquisition technique. In principle, LFQ allows every peptide feature within the sensitivity range of the MS

analyzer to be used for the quantification process independent of MS/MS acquisition, i.e., Data Independent Acquisition (DIA) or Data Dependent Acquisition (DDA) [47]. This also increases the dynamic range of the detected peptides. Because if we depend on DIA data, then peptide features outside of a sampling window are discarded. Similarly, if we depend on DDA data, lower abundant features are discarded. LFQ lets the quantification process independent of DIA or DDA acquisition technique. LFQ also largely reduces the undersampling problem common to the MS/MS spectral counting based approaches.

1.3.2 Biomarker Discovery

Comparative analysis between the detected peptide features in the protein samples obtained from healthy and disease-afflicted persons can discover novel proteins responsible for that disease. PEPPeR, a platform for experimental proteomic pattern recognition [31], demonstrates the discovery of novel proteins that could be considered biomarkers through targeted MS/MS (details will be explained later in Chapter 5). It is able to detect proteins shown to be changing between two sample groups without prior knowledge of their identities or even their presence in the mixture. PEPPeR takes a set of samples for comparison of protein abundance. Then performs peptide feature detection (using open source tool MapQuant [40]) on those samples' LC-MS maps. Then it aligns the identical peptide features across multiple samples. Record is kept for high abundant peptide features missing in some samples as well. In there experiments they found 232 peaks or peptide features significantly changing among the samples. Those were considered as biomarker peptides. MS/MS spectra were acquired for 171 of those 232 peaks with 119 yielding confident identifications via search by Spectrum Mill software. They obtain the sequence identities of even minor m/z peaks (peptide features with very low intensity) found to change across samples, which result in a large increase in the number of lower abundance proteins identified as differentially regulated in disease versus health. Besides targeted MS/MS, LFQ has also been found to be very effective in the development of disease protein biomarkers [76]. For example, Atrih et al. [4] present state-of-art label-free quantitative proteomics method in resected renal cancer tissue for biomarker discovery and profiling.

1.3.3 Identifying Chimeric Spectra in DDA or DIA

Some regions in LC-MS map are sampled for collecting peptide ions and sending to MS2 for fragmentation. That sampling window (very narrow range of m/z and RT) is also called MS/MS isolation window. Each isolation window ideally tries to select one peptide ion. A

four-hour long LC-MS/MS experiment using a high-resolution Orbitrap mass spectrometer can *identify* over 40,000 peptides and 5000 proteins with the data-dependent acquisition (DDA). It may also contain hundreds of peptides eluting simultaneously. If all the co-eluting peptides have a high density, it causes a high probability of two or more peptide ions overlapping within an MS/MS isolation window. It results in **chimeric** MS/MS spectra, with cofragmenting precursors being naturally multiplexed. Such chimeric MS/MS spectra are generally unwelcome in DDA because the product ions from different precursors interfere with the assignment of MS/MS fragment identities, causing false discoveries in the database search.

In some DDA studies, the MS/MS isolation window width was set very narrow (as narrow as 0.35 m/z) to prevent unwanted ions from being coselected, or fragmented. On the other hand, the Andromeda database search of the MaxQuant workflow [14] implements the "second peptide identification" method that submits the MS/MS spectra to the search engine several times based on the list of chromatographic peptide features (**detected peptide features**), subtracting assigned MS/MS peaks after each identification round. In the DeMix workflow [83], the deconvolution of chimeric MS/MS spectra consists of simply "cloning" an MS/MS spectrum if a potential cofragmented peptide is detected. The list of candidate peptide precursors is generated from chromatographic feature detection using The OpenMS Proteomics Pipeline (TOPP). Teleman et al. [67] proposed a **better peptide feature detection tool** which increased the chimeric identification from 26% to 32% over the standard workflow. We are interested to see **whether our proposed model can improve the chimeric identification further or not**.

In DIA, the MS/MS isolation window is large and therefore it often results in MS/MS spectra containing precursor ion of multiple peptides. Almost similar technique of identifying chimeric spectra mentioned above is used in the pipeline of DeepNovo-DIA [71]. It uses existing peptide feature detection tools, e.g., MaxQuant, to extract the chromatographic profile of precursor ion from LC-MS map and that is later incorporated with MS/MS spectra for the de novo peptide sequencing from DIA dataset using deep learning approach. Therefore, our model can also be applied for chimeric spectra separation in DIA data.

1.4 Deep Learning

Deep learning is a type of machine learning method based on artificial neural networks with the capability of feature learning. That means, it uses a set of techniques to automatically discover the representations needed for feature detection or classification from raw data. As a result, we can avoid feature engineering, i.e., using domain knowledge to extract features

from raw data. The learning is usually supervised (e.g., Convolutional Neural Network, Recurrent Neural Network), but can be semi-supervised or unsupervised as well (e.g., Auto-encoders [33], generative adversarial network [25]). Usually the backpropagation technique is used for training deep neural networks [59]. In the following sections we will discuss some deep learning models that are used in our research, and some proteomics studies which use deep learning models.

1.4.1 Convolutional Neural Network (CNN)

The CNN is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is biologically inspired by the organization of the animal visual cortex [30, 39]. The use of CNN in image processing was pioneered by Yann LeCun et al. [39] in 1998 for hand-written digit recognition. A simplified version of the MNIST digit recognition network is presented in Figure 1.6. The CNN layers introduce the property 'equivariant to translation' required for generalizing the edge, texture, and shape of objects in different locations. Different types of pooling layers (max-pooling, min-pooling, average-pooling) are usually inserted right after the CNN layers for achieving the property 'invariant to translation' that causes the precise location of the detected features to matter less. Therefore the target object is detected irrespective of its position in the image. The final layers consist of fully connected layers that do the job of classification. Although deep convolutional neural networks were invented in the early 1980s, they became popular after the revolutionary breakthrough in the ImageNet object recognition competition in 2012. The model proposed by Hinton and two of his students [35] was almost as good as humans at object recognition.

1.4.2 Recurrent Neural Network (RNN)

Next, we discuss recurrent neural network (RNN) which exhibits a temporal dynamic behavior. It is a class of artificial neural networks that can form a directed or undirected graph along a temporal sequence. Each node keeps internal state (memory) which lets the network process variable length sequences of inputs. As a result, it is applicable to tasks such as time series analysis, natural language processing, and speech recognition. John Hopfield popularized the "Hopfield" network, which is considered as the first recurrent neural network. This was subsequently expanded upon by Jürgen Schmidhuber and Sepp Hochreiter in 1997 with the introduction of the long short-term memory (LSTM). It greatly improved the efficiency and practicality of recurrent neural networks. Eventually deep

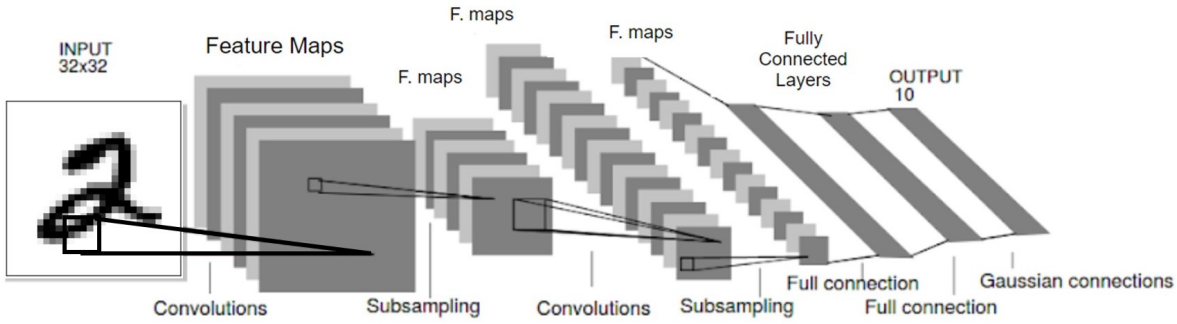


Figure 1.6: MNIST digit recognition network. Here a $[32 \times 32]$ input image is accepted as input. Then it is passed through convolution and pooling (subsampling) layers for feature extraction. The model learns basic shapes like lines and edges in the beginning layers and gradually learns more complex shapes as it gets closer to the output layer. There are fully connected feed forward layers to classify different digits before the end. The final output layer is a Softmax layer with 10 neurons because it wants to classify digits: 0 to 9.

learning with such memory networks started to bring groundbreaking results in many natural language processing tasks, for instance, sentiment analysis [63], learning semantic relations between similar words and sentence [10], reading comprehension [56], Google Neural Machine Translation system [79].

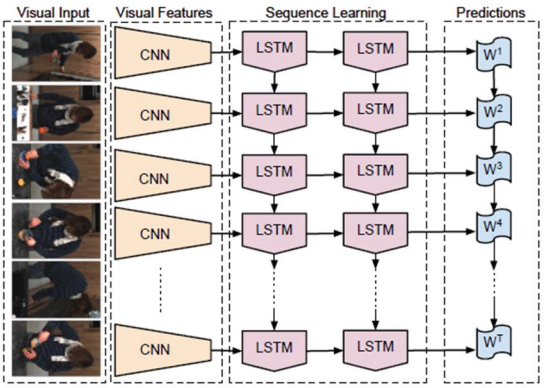


Figure 1.7: Long-term recurrent convolutional network (LRCN) for video clip activity description. LRCN processes the (possibly) variable-length visual in-put (left) with a CNN (middle-left), whose outputs are fed into a stack of recurrent sequence models (LSTMs, middle-right), which finally produce a variable-length prediction (right).

1.4.3 Combination of CNN, RNN and Attention Mechanism

People have developed different strategies for combining CNN and RNN to detect patterns that span over time. For example, long-term recurrent convolutional network for activity recognition in a video clip [16], as presented in Figure 1.7. NVIDIA proposed a more effective approach FC-RNN, to transform a network pre-trained on separate frames or clips to deal with video as a whole sequence. A simple comparison of standard RNN and FCRNN is presented in Figure 1.8. The variables in red correspond to the parameters that need to be trained from scratch. In FC-RNN, only one parameter needs learning because all other parameters are learned during pre-training.

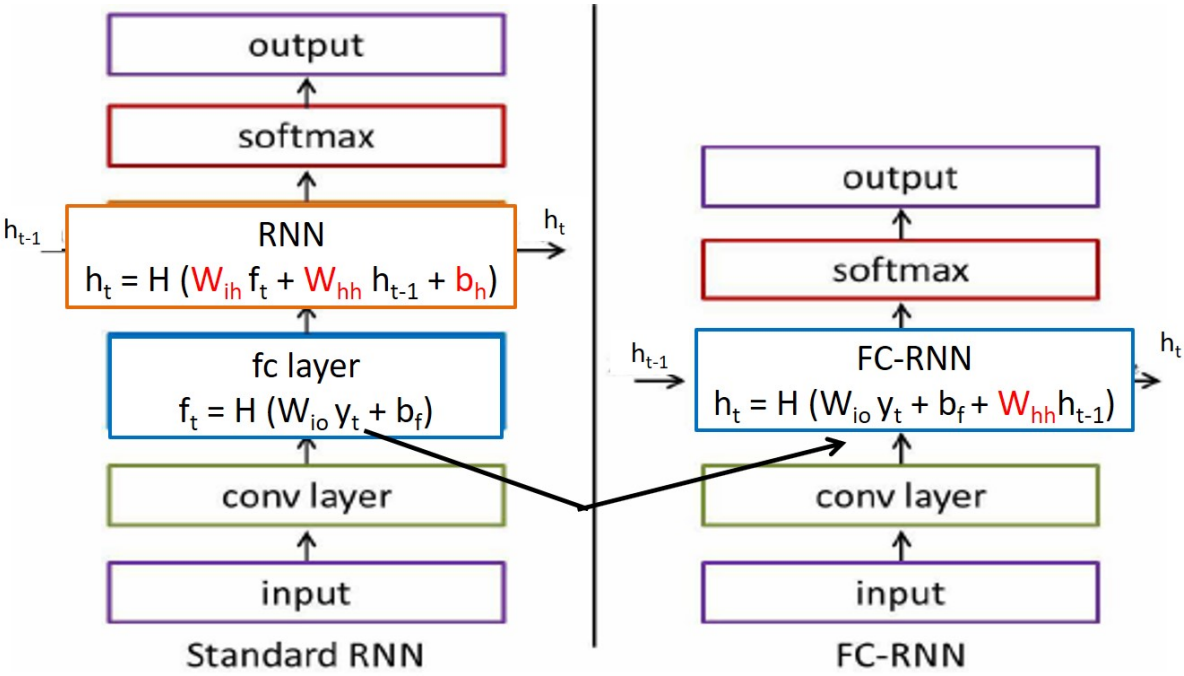


Figure 1.8: Comparison of standard RNN and FCRNN. The main difference lies in the way of deciding the next state. The variables in red correspond to the parameters that need to be trained from scratch. In RNN, weight matrix associated with previous state, current state and bias at current state are to be learned through training. In FCRNN, only the weight matrix associated with previous state is to be learned from scratch since it works on a pretrained model.

Various 'Attention' mechanisms are also proposed in computer vision and natural language processing problems to fetch vital information or features more effectively. For example, the Image Captioning system presented in Figure 1.9 is proposed by Levin et al. [80]. Pei et al. [51] propose the Temporal Attention-Gated Model (TAGM) which inte-

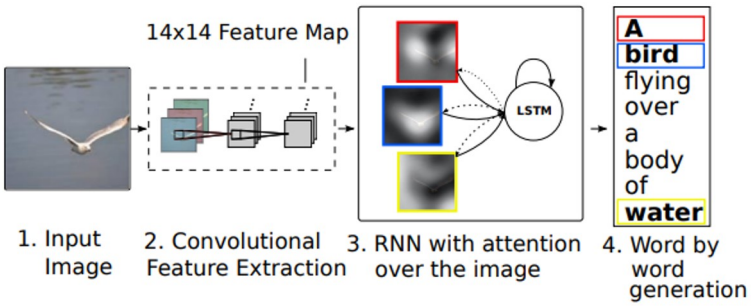


Figure 1.9: Workflow of 'Show, Attend and Tell' system for neural image caption generation

grates ideas from attention models and gated recurrent networks to better deal with noisy or unsegmented sequences, expected in real-world applications. The model is presented in Figure 1.10.

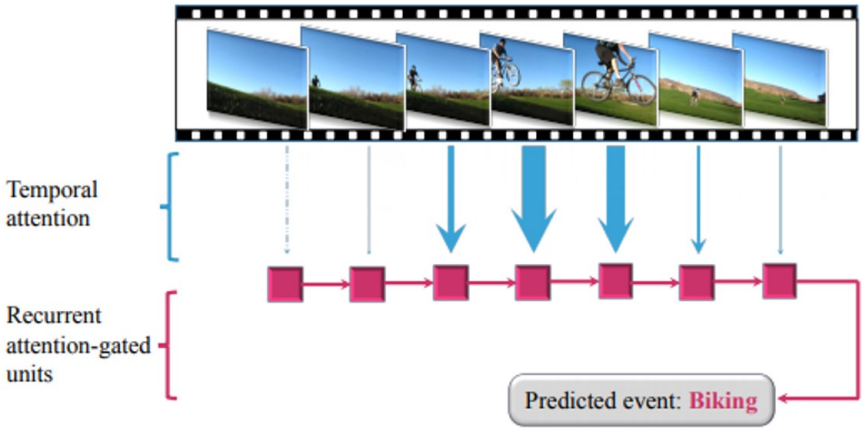


Figure 1.10: TAGM first employs an attention module to extract the salient frames from the noisy raw input sequences, and then learns an effective hidden representation for the top classifier. The wider the arrow is, the more the information is incorporated into the hidden representation. The dashed line represents no transfer of information.

The dual attention network (Figure 1.11) for scene segmentation is proposed by Jun et al. [20] that adaptively integrates local features with their global dependencies. They append two types of attention modules which model the semantic interdependencies in spatial and channel dimensions respectively. They achieve new state-of-the art segmentation performance on three challenging scene segmentation datasets.

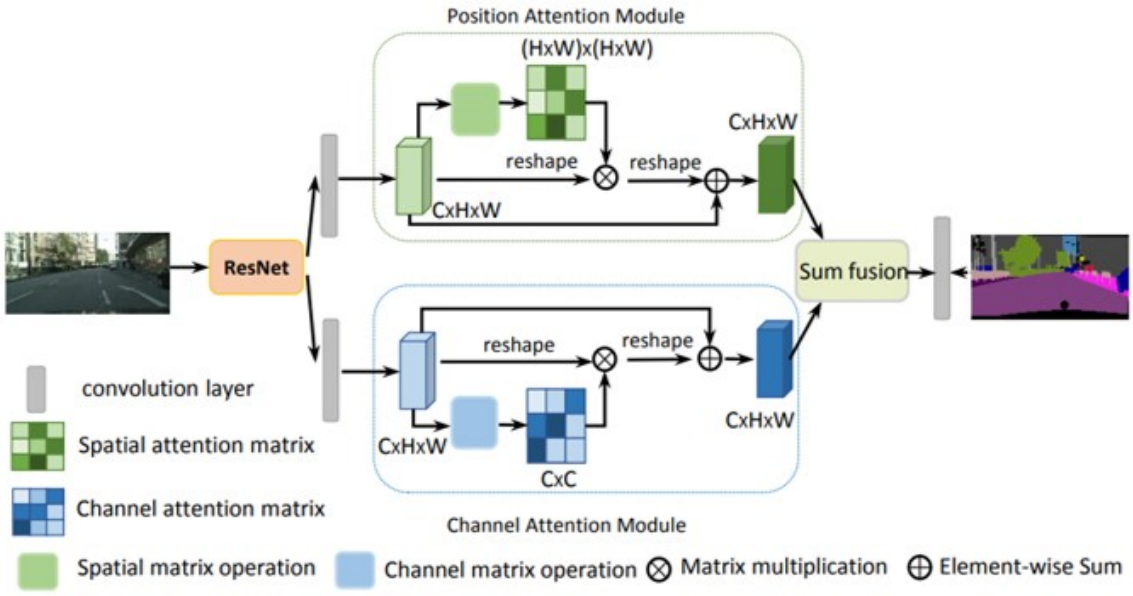


Figure 1.11: Performance of dual attention network (DANet). It appends two types of attention modules: spatial dimension (green) and channel dimension (blue), in order to model semantic interdependencies in those two dimensions. In our work, we use the attention calculation technique as shown in the attention modules.

1.4.4 3D Point Cloud Based Models

A point cloud is a set of data points in space, where each point position has its set of Cartesian coordinates. Therefore, if it is a 3D space, the points have (X, Y, Z) coordinates. For 3D object detection, people may transform the object into regular 3D voxel grids or 2D projected images. However, converting 3D objects into 2D projected images makes the data unnecessarily voluminous and causes issues of running time. Unnecessary pixel representation of blank space in the point cloud may also lead to false positives during prediction. PointNet [54] is a novel neural network that directly processes point clouds instead of converting them to some other representations, and applies to 3D shape classification, shape part segmentation, and scene semantic parsing task, as shown in Figure 1.12. Empirically, it shows strong performance on par or even better than other existing approaches. Later, Dynamic Graph CNN [77] and many other networks are proposed to handle various aspects of point cloud based systems.

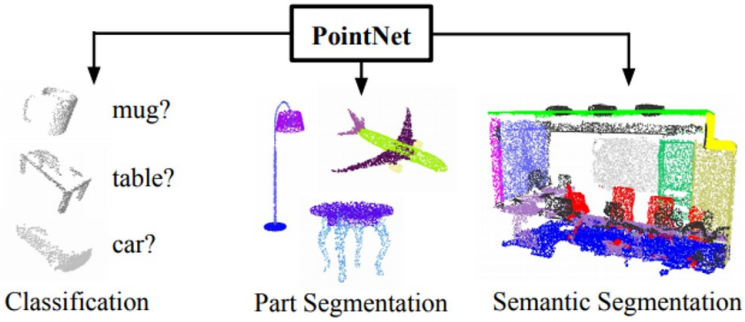


Figure 1.12: Pointnet is a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

1.4.5 Deep Learning in Proteomics

The latest LC-MS technologies generate vast amounts of analytical data with high scan speed and high resolution, which is almost impossible to interpret manually. On the other hand, deep neural networks are very effective and flexible in discovering complex data structures through their many layers of neurons. Thus deep learning has made its way into analyzing LC-MS/MS data as well [38]. For instance, DeepNovo [74, 71] introduces

deep learning to de novo peptide sequencing from tandem MS data. Bulik-Sullivan et al. [6] propose a computational model of antigen presentation for neoantigen prediction using deep learning. DeepSig [61] takes input protein sequence to detect signal peptides and predict cleavage-sites. DeepRT [44] provides improved peptide Retention Time prediction in liquid chromatography. PROSIT [22] offers a proteome-wide prediction of peptide tandem mass spectra by deep learning. Guan et al. [27] propose deep learning strategy for the prediction of LC-MS/MS properties of peptides from sequence. Very recently, AlphaFold, developed by Google's DeepMind, shows how deep learning can be used for predicting the 3D structure of a protein solely based on its genetic sequence. The 3D models of proteins generated by AlphaFold are far more accurate than all other existing methods, making significant progress on one of the core challenges in biology. The application of deep learning in this field is gradually becoming a method of choice.

1.5 Overview on Research Contribution

Every step in LC-MS/MS based analysis of proteomics data is important, and new literature on these steps is published frequently. In this thesis, I present my research on applying deep learning in peptide feature detection from raw LC-MS map, a crucial step in the proteomics analysis of biological samples. In the existing algorithms for peptide feature detection, many parameters are set based on empirical experiments, whose different settings may have a significant impact on the outcomes, therefore, prone to human error. In contrast to these existing works, our research aims to systematically train a deep neural network using an appropriate dataset to automatically learn all data characteristics without human intervention. This automation of peptide feature detection is the main strength of this research. On top of that, there is still room for improvement in the peptide feature detection context. It is a pattern recognition problem, and deep learning networks are bringing groundbreaking results in the computer vision domain. So it is worth investigating our target problem with deep learning. Besides that, although advancing technology frequently generates new information that may help detect peptide features, it is hard to incorporate new information into the existing knowledge-based systems. In contrast, we can easily fine-tune deep learning models to adapt to that information. Last but not least, even if our model makes wrong predictions, we can apply reinforcement learning by putting back the correct results as new training data so that the model can learn from its own mistakes. We are the first to apply deep learning in solving peptide feature detection problem as per our knowledge. This problem is far more challenging than general object detection problems. Because the input LC-MS map is of gigabyte size and each LC-MS

map can provide over 50,000 peptide features. These features are tiny with respect to the background and often overlap. Moreover, there are feature-like noisy signals, and sometimes actual features do not follow the typical characteristics of being a true feature. All these things make this problem more challenging to address. We observe that our deep learning models have superior performance over existing techniques and may become the method of choice soon.

1.5.1 List of Developed Models and Experimental Analysis

A brief overview of our accomplished research works is provided below:

- Development of image-based deep learning model DeepIso, that works on the 2D projected image of 3D peptide features.
  - I initiate the research through a naive Convolutional Neural Network (CNN) and some heuristics. Our model [89] achieves 93.21% sensitivity with 99.44% specificity on an antibody dataset including a heavy and a light chain. This method is explained in Chapter 2. Although this work was not published, it was crucial for understanding the scope of deep learning in this context.
  - After achieving convincing results from the previous step, we approached further and developed a more sophisticated model by combining Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) [81], and some attention mechanism [51]. The peptide feature list reported by our model [87] gives 97% sensitivity for detecting high-quality MS/MS identifications in a benchmark dataset. This sensitivity is higher than the matching produced by several other widely used tools. This model is elaborated in Chapter 3. This work is published in Scientific Reports. (Although the model name is kept same as the naive model mentioned in the previous point, please note that they have totally different architecture.)
- Development of point cloud based deep learning model PointIso, that allows direct input of 3D datapoints and provides higher accuracy with three times faster speed.
  - I utilize three-dimensional point cloud based models [54] and attention mechanisms [20] in order to design a novel feature segmentation technique through a non-overlapping sliding window. PointIso [88] achieves 98% detection of high-quality MS/MS identified peptide features in a benchmark dataset. It provides

3% to 6% higher spiked peptide (target) detection than other existing tools. PointIso is explained in Chapter 4. It is published in Scientific Reports.

- Evaluating the practical utility of PointIso by quality assessment of peptide features, and extending it for 4D peptide feature detection.
  - In order to make PointIso more reliable for biomarker discovery, we performed a quality assessment [88] of the reported peptide features based on the statistical analysis proposed by Michalski et al. [46].
  - Label-Free quantification (LFQ) directly utilizes the intensity of detected peptide features. Therefore, we evaluate the quantitative accuracy of PointIso by measuring relative peptide abundance in a benchmark dataset (produced for LFQ analysis) according to the technique discussed by Chawade et al. [9].
  - I extend our PointIso model for 4D peptide feature detection, where the additional dimension represents the 'ion-mobility'. Our model obtains 4% higher detection than other existing tools on human proteome dataset [88]. This demonstrates that PointIso is generic to be used in a wide range of problems in biological pattern recognition, as well as general computer vision problems.

  These experiments are presented in Chapter 5.

1.5.2 Model Training Criteria

Our models are supervised deep neural networks. Therefore, we need annotated dataset for model training. One LC-MS map can contain hundreds of thousands of peptide features. However, human annotation is out of scope for those features [66]. Therefore, we apply some existing software for peptide feature detection and use the common set of those as the annotated training data for our model. Although it may seem that our training data is biased, however, this type of training approach is already supported by existing literature [66, 65, 57]. We also provide the following statements in support of our training approach:

- The first concern is what happens if the common set (training set) misses some true features. By experiment, we see that our model is able to detect identified peptide features out of the common set. Our model's detection percentage is higher than all other existing software. That means, missing some true features in the common set does not hamper the learning, since deep learning models can learn the true

characteristics of features instead of relying on some fixed heuristic steps for feature detection.

- If the common set has some false positives, that amount will be quite low compared to the set's true features. Therefore, our model should be able to distinguish them itself as noise or outliers. Being able to do that implies higher intelligence or true learning by the model. We can say that our model is good at noise removal. Because our model works on raw (profile mode) data, whereas all other existing software (e.g., OpenMS, MaxQuant, PEAKS, Dinosaur, etc.) perform some external/additional noise removal step on their data before starting the main detection. As mentioned in later chapters, we do not apply external noise removal but still provide a reasonable amount of peptide features. Also, we show that fine-tuning by false positives improves the model's performance in filtering out noises. Therefore probable false positives in the training set can not cause a bad impact.

- Using a common set of other software does not imply that our model learns to mimic their approach. We used the common set merely to replace human annotators to label the training data, or choose some reliable features, and such techniques are common in machine learning context. Our deep learning model learns the appropriate parameters for peptide feature detection by stochastic gradient descent through several layers of neurons and backpropagating the prediction errors, a completely different approach than existing heuristic methods. Therefore the learning outcome of this deep learning model is quite different, and thus, it provides higher detection than other heuristic methods.

Once we train a model on a protein sample dataset, the same model should be applicable to all other protein samples from the same or other close species (processed by similar MS instrument as our training dataset), without further training. Although we are having good results through this supervised training approach, we also propose some semi-supervised learning technique as future work in the end, so that our future models become more independent of existing tools.

1.5.3 Model Evaluation Criteria

From a protein sample we get two types of data through LC-MS/MS analysis: LC-MS maps (MS1 data) and MS/MS spectra (MS2 data). We apply peptide feature detection on LC-MS map and perform database search on MS/MS spectra to get a list of identified peptides. Now, for the identified peptides, we are sure about their existence in the LC-MS

map in a form of peptide feature because they are identified. Therefore, if our model can detect more identified peptide features than other software, it implies better performance by our model. (Please note that peptide identification by database or de novo searching alone is insufficient in the analysis pipeline. Because to quantify the respective protein, we need the abundance of its peptides, which comes from the corresponding peptide features.) Therefore, **our principle evaluation metric is: "What percentage of identified peptides are detected by the model?". So sensitivity of our model is defined as the percentage of identified peptides for which our model can detect a corresponding peptide feature.** Besides that, we also perform a quality assessment of the peptide features generated by our model but not identified by database search. Because they should be reliable features and not false positives, so that our model has the potential for disease biomarker discovery through targeted MS/MS (but please note that, we can never say whether those detected, but unidentified features are true/false with absolute certainty). Finally, we verify the reliability of relative peptide abundance calculation based on peptide feature boundary proposed by our model so that it is applicable in Label-Free Quantification.

1.6 Thesis Organization

We provide a brief discussion on our naive convolutional neural network and some significant findings in Chapter 2. Then in Chapter 3, we propose a more effective deep learning model DeepIso along with experimental results. Next, we present PointIso in Chapter 4, which is significantly different in terms of architecture, three times faster, and more robust than DeepIso. We will elaborate our research works in each of these two chapters by presenting the corresponding model workflow, result, discussion, and model architecture with methods. Then Chapter 5 has three major sections. First, we present a feature quality assessment. Second, we verify our model in Label-Free Quantification (LFQ). Third, we extend our 3D model for 4D peptide feature detection and discuss empirical results. Finally, we will conclude in Chapter 6 by stating some potential future works.

Chapter 2

Naive Convolutional Neural Network For Peptide Feature Detection

Detecting peptide features, i.e., multi-isotopic patterns in LC-MS map is a challenging task due to the overlapping peptides, several charges of the same molecule, and intensity variation. If we represent LC-MS map as a 2D projected image over $[m/z \times \text{RT}]$ plane, then it may have gigapixel size containing hundreds of thousands of peptide features. However, Convolutional Neural Networks (CNN) are very effective in similar pattern recognition problems, e.g., in detecting cancer metastasis on gigapixel pathology images by Google DeepMind [42]. Being inspired by the work, we propose a deep learning model using CNN to scan input LC-MS map using overlapping sliding window to detect peptide features in the map along with their charge, and estimate their abundance.

2.1 Workflow

We explain our method using a block diagram as shown in Figure 2.1. We first train a CNN through supervised training and then use it to scan the LC-MS maps through sliding window. We design our problem as a 10-category classification problem using CNN, where each category is a charge $z = 0$ to 9. The intuition is, CNN takes a $[M \times N]$ dimension sliding window as input image (extracted from LC-MS map) and outputs the charge $z = 0$ to 9. We use $z = 0$ as an indication of 'No' feature is seen in the input image or sliding window. The charge $z = 1$ to 9 means a feature having that charge is seen in the input image.

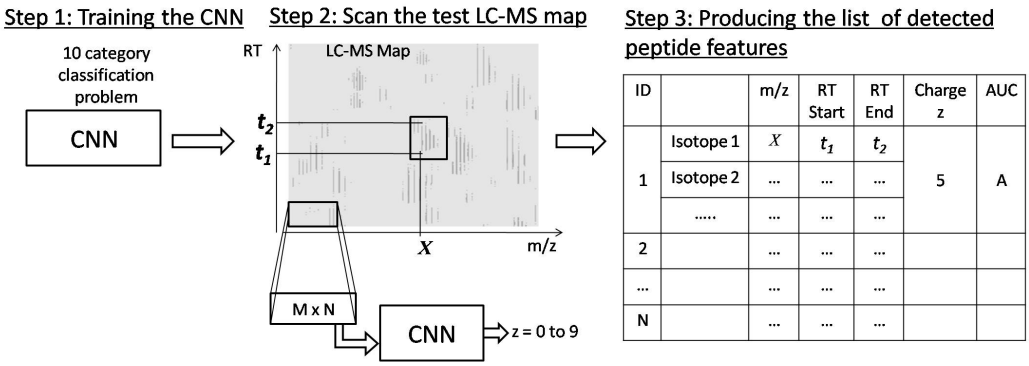


Figure 2.1: Block diagram of our proposed method to detect peptide features from LC-MS map of protein sample. First a CNN is trained as a 10-category classification problem. Then that is used to scan the LC-MS map through sliding window. While scanning, CNN outputs $z = 0$ if no feature is seen in input, and outputs $z = 1$ to 9 if a feature having corresponding charge is seen. After those outputs are processed though some heuristic methods, we get the feature table as shown in the right. Here, Id is assigned to each feature. For each feature, there is a list of isotopes. For each isotope, we show the m/z location, and RT time range. We also show the charge z of the feature and its total intensity as AUC.

The scanning results are recorded in a hash table, where each record holds a detected isotope, along with its position (m/z and RT range), charge of the corresponding feature, and intensity. Next, we use some heuristic steps to group those detected isotopes into peptide features and save the features in a feature table, as shown in 'step 3' of the block diagram. The detailed architecture of our convolution neural network is shown in Figure 2.2. Model hyper-parameters are discussed later in Section 2.4.2.

2.2 Dataset

During the training, CNN is supposed to learn basic properties of peptide feature as mentioned in Section 1.1.1, besides many other hidden characteristics from the training data. We use the dataset WIgG1 of monoclonal antibody sequence from mouse including a light chain and a heavy chain [73] to perform the experiment. It was generated from the LC-MS/MS analysis of the Intact mAb Mass Check Standard purchased from Waters. It is an intact mouse antibody purified by Protein-A with known molecular weights and amino acid sequences of both the light and heavy chains. Two chains result in two dif-

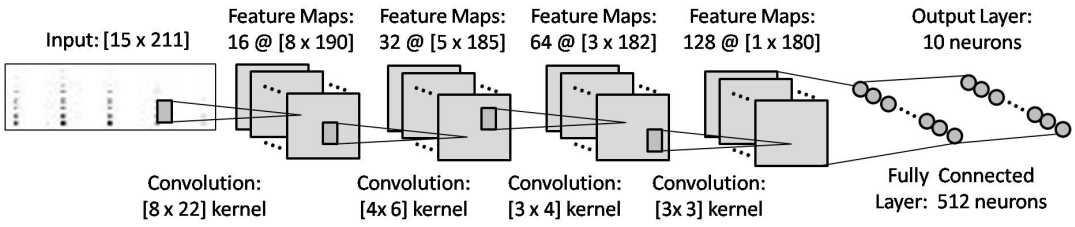


Figure 2.2: Architecture of our proposed Convolutional Neural Network. It takes input a [15×211] dimension image (scanning window). Then it is passed through four convolutional layers (without pooling), one fully connected layer, and a Softmax output layer having 10 neurons, since this is a 10-category classification problem.

ferent types of protein sample. Each sample was digested using three types of enzymes separately: Asp-N, Chymotrypsin, and Trypsin. Therefore, from the sample of heavy chain, we have three separate protein digests resulting three LC-MS maps: WIgG1-Heavy-AspN, WIgG1-Heavy-Chymotrypsin, WIgG1-Heavy-trypsin. The same is performed for the sample containing light chain, and we get three more LC-MS maps: WIgG1-Light-AspN, WIgG1-Light-Chymotrypsin, WIgG1-Light-trypsin. In this way, we have in total six LC-MS maps for doing the experiment. The RAW files of the antibody dataset can be downloaded from the database MassIVE with accession number MSV000079801. The data ranges from 400 m/z to 1,500 m/z, and up to 30 minutes along RT axis. We consider resolution of 0.01 m/z along the horizontal axis, and 0.01 minute along the vertical axis. Based on this resolution, each LC-MS map has dimension of around $[\frac{30}{0.01} \times \frac{1100}{0.01}]$= $[3,000 \times 110,000]$ pixels. We scale the pixel intensities in each map from 0 to 255. For clarification please refer to the LC-MS map shown in Step 2 in Figure 2.1.

Application of PEAKS Studio[1] produces about 20,000 peptide features from each LC-MS map. We use that as our annotated/labeled dataset for the training (in later models we use a common set of multiple software). We cut the features from the training LC-MS maps, considering a block/window size of $[M \times N] = [15 \times 211]$. In Appendix A.1 we discuss why this block size is good enough to cover a feature so that CNN can take decision about the existence and charge of a feature. The cut samples which contain features having charge $z = 1$ to 9, are marked as positive samples. We cut some samples from blank regions and noisy areas which are marked as negative samples. In this way we cut about 55,000 positive samples and about 90,000 negative samples for each fold, giving about 145,000 features for training. The details about training data generation is provided later in the

[1]http://www.bioinfor.com/peaks-studio/

Section 2.4.1.

2.3 Result

We have six LC-MS maps, and therefore, we apply 6-fold cross validation for evaluating our model. Each time we keep one map for testing and the remaining for the training. This way, it keeps about 16% data for testing and the rest for training. We cannot reduce the fold size, because it will give inadequate number of samples for training. During training, we basically cut sample from each peptide feature in training LC-MS maps for the training. During the testing, we scan the sliding window over the whole region of a given LC-MS map and detect peptide features.

For evaluating testing performance, we see what percentage of PEAKS detected features are also detected by our model. The existence of a peptide feature is supposed to be 100% accurate if that peptide feature can be identified by database search or de novo sequencing technique. Please note that, we may not find De Novo or database sequence for all of the peptide features detected from an LC-MS map. We also cannot decide if those other features are true or false since there is no ground truth. Therefore, for the evaluation of testing phase, we arrange the features detected from test LC-MS map by PEAKS into following two types:

- Type X: All features
- Type Y: Only those features for which we can find De Novo or Database sequence.

In testing phase, we would say a feature is detected by our proposed model, if the feature profile reported by our method matches with the feature profile provided by PEAKS. To be specific, we compare following three points to decide whether a feature is detected.

1. If the charge z of a feature is matched.
2. The m/z value of the starting isotope in a feature reported by our naive CNN matches with that of PEAKS result within tolerance level of 0.05 m/z or Da. This tolerance level is good enough for this dataset according to the collaborators from Bioinformatics Solutions Inc. If the first one matches, then the remaining isotopes' m/z match as well, since they are equidistant.

3. The RT range of a feature reported by our method overlaps with that of PEAKS result and the RT value giving the highest peak intensity of a feature matches with a tolerance level of 0.5 minute. This tolerance level is good enough for this dataset according to the collaborators from Bioinformatics Solutions Inc.

2.3.1 Model Sensitivity

Like other existing literature works, we define the true positive rate of our model on the test LC-MS map using the metric sensitivity or recall, i.e., what percentage of type X and Y are detected. We denote the test LC-MS map in six folds as 'A', 'B', 'C', 'D', 'E', and 'F' respectively. We report the sensitivity for each fold and the average sensitivity as well for data type X in Table 2.1, and for data type Y, in Table 2.2.

	A	B	C	D	E	F	Average sensitivity of z detection over all samples (A to F)
charge, z=1	85.65	81.56	87.35	85.07	83.68	85.01	84.72
charge, z=2	90.84	88.68	88.85	90.80	89.17	88.10	89.41
charge, z=3	90.09	88.17	86.29	88.73	87.34	85.81	87.74
charge, z=4	88.53	86.66	84.90	89.66	86.80	83.70	86.71
charge, z=5	91.04	91.74	89.79	90.16	94.40	85.51	90.44
charge, z=6	71.79	78.49	73.08	82.99	78.71	64.89	74.99
charge, z=7	70.97	76.00	74.29	90.00	76.67	75.00	77.15
charge, z=8	25.81	28.57	42.86	37.08	15.38	16.67	27.73
charge, z=9	31.58	50.00	25.00	2.56	16.67	0	20.97
Average sensitivity of feature detection (all charges)							**87.61**

Table 2.1: Class sensitivity (%) for six LC-MS maps (A to F) for type X data. Type X represents the features detected by PEAKS which may or may not map to a De Novo or database sequence. For charge 7 to 9, there were very low amount of training samples. Therefore, those features are not learned well, showing comparatively poor sensitivity.

	A	B	C	D	E	F	Average sensitivity of z detection over all samples (A to F)
charge, z=1	-	50	100	75.00	100	75	80.00
charge, z=2	95.37	95.00	93.21	94.68	94.58	93.80	94.44
charge, z=3	92.13	94.53	88.24	92.59	92.68	92.47	92.11
charge, z=4	91.11	92.31	84.48	91.14	91.88	84.73	89.28
charge, z=5	95.74	100	91.67	92.31	95.96	84.43	93.35
charge, z=6	90.48	100	50	90.57	95.74	87.50	85.71
charge, z= 7	-	-	-	-	-	-	-
charge, z=8	-	-	-	-	-	-	-
charge, z=9	-	-	-	-	-	-	-
Average sensitivity of feature detection (all charges)							**93.21**

Table 2.2: Class sensitivity (%) for six LC-MS maps (A to F) for type Y data. Type Y represents the features detected by PEAKS which also map to a De Novo or database sequence. There were no type Y feature belonging to charge 7 to 9.

Detection of features having higher intensity is important in the workflow of LC-MS/MS analysis [66]. We present the model sensitivity under different intensity range, starting from 0 to 1×10^{10} (maximum intensity in all six LC-MS maps lie within this range) in Table 2.3 for data type X and Y. Our CNN performs well when intensity is higher, as desired.

2.3.2 Model Specificity

In order to evaluate the model in terms of False Positives, we use the metric *specificity*. Usually people define True Negative cases based on their processing strategy according to the existing literature. For example, Conley et al. [11] define True Negative cases in terms of unused centroids, and report their model specificity. Since our model finds the

Intensity Range	Type X	Type Y
$0 \leq AUC < 1 \times 10^3$	73	77.25
$1 \times 10^3 \leq AUC < 1 \times 10^4$	73.61	80.06
$1 \times 10^4 \leq AUC < 1 \times 10^5$	81.63	83.82
$1 \times 10^5 \leq AUC < 1 \times 10^6$	88.52	93.04
$1 \times 10^6 \leq AUC < 1 \times 10^7$	92.24	94.03
$1 \times 10^7 \leq AUC < 1 \times 10^8$	94.02	95.13
$1 \times 10^8 \leq AUC < 1 \times 10^9$	96.04	95.93
$1 \times 10^9 \leq AUC < 1 \times 10^{10}$	99.24	100.00

Table 2.3: The class sensitivity (%) of our model under different AUC range for type X and Y. Here, AUC is the total intensity of peptide features or area under the curve of isotopic signals in peptide features.

features in LC-MS map by scanning it using CNN, therefore we define True Negatives in terms of pixels. We will illustrate it using a counter example. Let us consider CNN detection after scanning over a small area of LC-MS map having dimension [10 x 20] as shown in the Figure 2.3. Here CNN says 'Yes' in the pixels shown in Black and Green. In all other pixels CNN says 'No'. Now, the Black pixels belong to two True Positive features. The Green pixels represent a False Positive feature and the Red pixels represent a False Negative feature. The other White pixels represent True Negative cases. According to the figure, we can calculate the specificity in this region as follows:

$$\begin{aligned} \text{specificity} &= \frac{\text{pixels belong to True Negative (White pixels)}}{\text{pixels belong to True Negative + pixels belong to False Positive (Green pixels)}} \\ &= \frac{160}{168} \\ &= 95.24\%. \end{aligned}$$

In this way the average specificity of 6-fold cross validation of our model is **99.44%**. However, we measure this metric just to observe the model behavior. But in our later experiments, we do not use specificity in this way for model evaluation. Because, we can not really define true negative or false positive cases due to the absence of ground truth data.

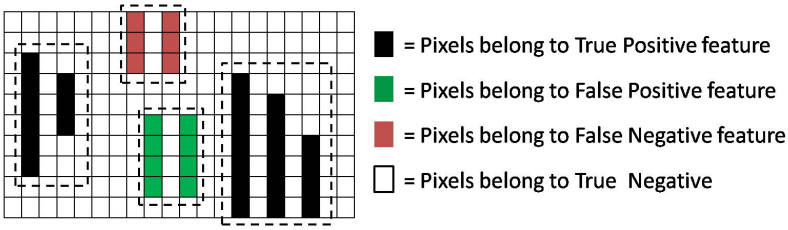


Figure 2.3: Calculation of specificity

2.3.3 Verification of Peptide Intensity

For the statistical analysis of biological experiments the feature intensity is of interest and has to be calculated from the raw data [66]. The technique is to first apply curve fitting over the beta distribution shaped intensities of isotopes in a feature. Then the Area Under Curve (AUC) of all isotopes in a feature are calculated and added to get the intensity or AUC of that feature. The Pearson correlation of feature intensity calculated by our model and PEAKS is 0.78 for type X data, and 0.83 for type Y data (average of 6-fold cross validation). They are good since it means that our naive CNN model should provide similar protein quantification performance as PEAKs.

2.4 Methods

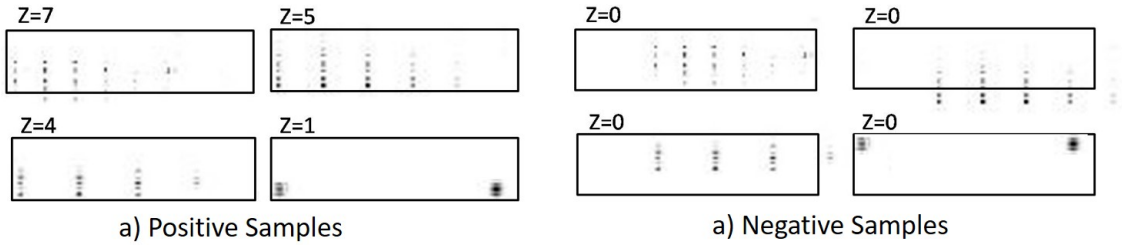


Figure 2.4: Generation of training data: (a) Generate positive samples by placing a [15 x 211] window over the feature, such that, first isotope of the feature is centered at [0,6] pixel of the window; (b) Generate negative samples by translating the window around the features, such that, NO feature starts within the $[0, 0]$ to $[0, 6]$ pixels of the window

2.4.1 Training Data Generation

We cut the positive features so that the feature is placed at pixel $[0, 6]$ of the block (horizontal displacement of 6 pixels) as shown in Figure 2.4 (a), because some features have wider isotopes, for instance, the feature having charge z=1 in the figure. The amount of features having charge $z = 1, 2, 3, 4$ is higher than other positive samples in the dataset. Therefore, to balance the dataset, we perform oversampling for charge 5 to 9. We cut multiple samples from the same feature by translating the sampling window a little bit along the m/z axis (horizontally left to right). The procedure is explained in Appendix B.1. To generate negative samples, we select some features and cut blocks from their surrounding region, satisfying the condition that NO feature starts within the $[0, 0]$ to $[0, 6]$ pixels of the block, as shown in Figure 2.4 (b).

In this way we cut about 55,000 positive samples and about 90,000 negative samples for each fold, giving about 145,000 peptide features for training where the percentage of features having $z = 0$ to 9 is about 62%, 7%, 10%, 9%, 4%, 5%, 2%, 0.5%, 0.2%, 0.1% respectively. We select 20% of them for validation such that validation dataset *does not contain* over-sampled positive samples. We keep the ratio of negative samples higher because the LC-MS map is very sparse, and most of the spaces hold no feature.

2.4.2 Model Training Parameters

We apply a 6-fold cross validation on six LC-MS maps. Each time we keep one map for testing and the remaining are used for training. We keep 20% samples of training data for model validation. We apply stochastic optimization using Tensorflow provided 'AdamOptimizer' with learning rate of 0.001 [32]. We use the rectified linear unit (ReLU) as activation function of the neurons and sparse Softmax cross entropy as error function at the output layer. We add dropout layer after final convolution layer and fully connected layer with a value of 0.50, which increases the validation sensitivity by 1.5%. Minibatch size is considered 128 to ensure enough weight update in each epoch. During training, we see the training loss and validation loss every 10 minibatch. Initially both training and validation loss continue to go down. We run the epochs until we see that training loss goes on decreasing but validation loss stops decreasing and does not change anymore. However, during that time, we see that the validation sensitivity goes on increasing for some epochs, and we keep track of the model having highest sensitivity. After some epochs, the validation sensitivity also stops improving. We mark this point as the convergence point and record the model having highest average sensitivity with lowest possible loss. We perform data

shuffling after each epoch which helps to achieve faster convergence. Our CNN converges within 30 epochs.

Model Training and Validation Sensitivity

We apply a 6-fold cross validation on six LC-MS maps. Each time we keep one map for testing and the remaining are used for training (with 20% kept for validation). In this way, the average training and validation sensitivity is 94.44% and 96.08% respectively. The class sensitivity is presented in Table 2.4.

charge z	Training sensitivity (%)	Validation sensitivity (%)
0	99.87	99.82
1	86.37	81.80
2	93.76	90.73
3	96.99	94.58
4	97.12	94.01
5	56.00	95.27
6	47.87	78.02
7	53.11	87.02
8	46.04	46.07
9	45.02	45.32

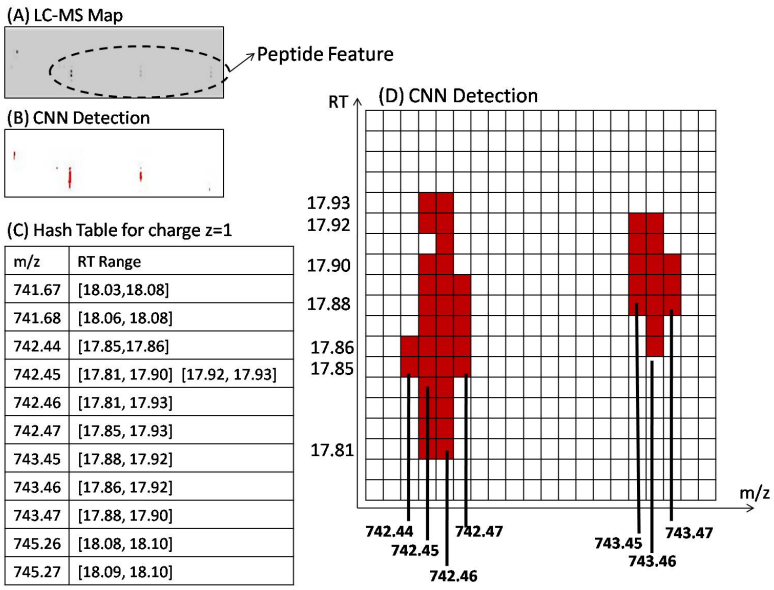
Table 2.4: Average class sensitivity of training and validation data. Please note that, for charge z=1 and z=6 to 9, the sensitivity is poor because the model cannot learn those features due to low amount of training samples.

2.4.3 Heuristics Steps

In this step 3, we process the hash tables (resulting from Step 2 as shown in Figure 2.1) and group the isotopes into features using some heuristics based on typical peptide feature properties. This gives a complete list of peptide features showing the m/z, and RT range of each isotopes and intensity of the feature as shown in Figure 4.1, in Step 3. We explain the detailed procedure of processing the hash tables as follows.

Processing the Hash Table for Naive CNN

Please refer to Figure 2.5 where a small region of LC-MS map holding a feature with charge $z = 1$ is shown (in (A)) and the result of CNN detection after scanning this region (in (B)) and corresponding records in hash table (in (C)) are shown as well. Records presenting the feature is further shown using a RT vs m/z plot (in (D)).



m/z	RT Range
741.67	[18.03,18.08]
741.68	[18.06, 18.08]
742.44	[17.85,17.86]
742.45	[17.81, 17.90] [17.92, 17.93]
742.46	[17.81, 17.93]
742.47	[17.85, 17.93]
743.45	[17.88, 17.92]
743.46	[17.86, 17.92]
743.47	[17.88, 17.90]
745.26	[18.08, 18.10]
745.27	[18.09, 18.10]

Figure 2.5: CNN detections recorded in hash table

Following three steps are performed while processing the hash tables:

1. **Merging the RT extents**: Noise during data record causes some break in a feature as shown in Figure 2.6. Because of this the CNN detection may produce traces with

Figure 2.6: Break within a peptide feature

small gaps as shown by arrow sign in Figure 2.7. We merge such small gaps if the gap <= 5 pixels, that is 0.05 minutes. This value is chosen by experiment.

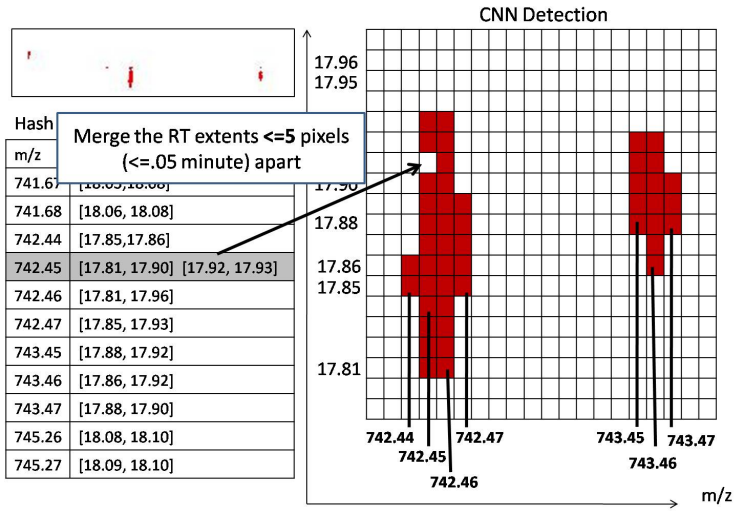


Figure 2.7: Merging of RT Extents

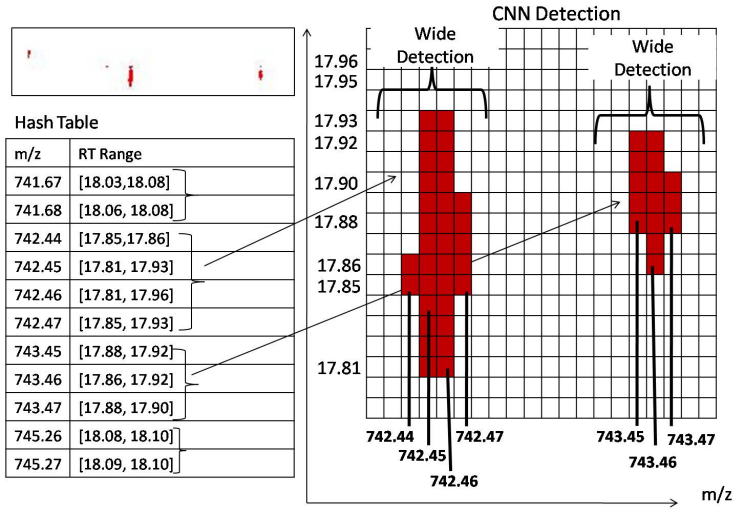


Figure 2.8: Combine adjacent traces who are overlapped along RT axis

2. **Select the center m/z for a wider isotope**: Although PEAKS reports just a single m/z value for an isotope, but each isotope has width. That is, each isotope span over multiple pixels along m/z axis. Therefore, the CNN also produces wide detection as visible in Figure 2.8. However, we have to pick one m/z value for each isotope. For a set of adjacent traces representing one isotope, we calculate the

intensity in terms of Area Under Curve (AUC) for each of them, and select the one that gives highest AUC as shown in Figure 2.9.

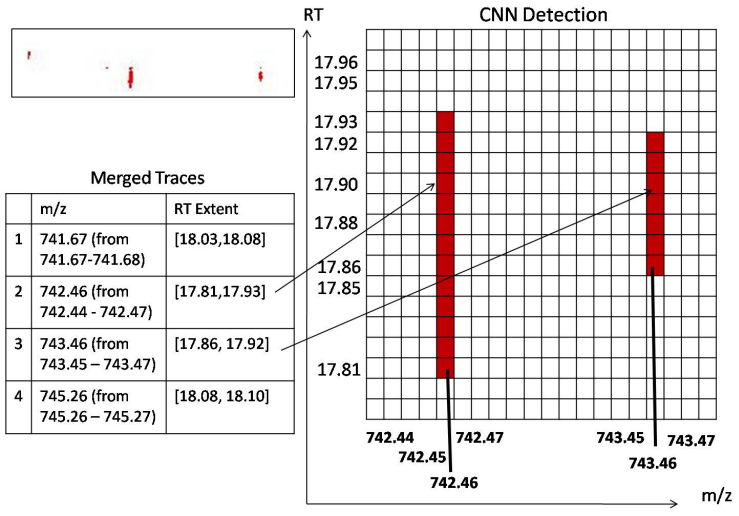


Figure 2.9: Selection of single m/z value for each isotope

3. **Condition checking to combine potential isotopes into one feature**: In this step we apply some heuristics as explained below.

   - We focus on the equidistant isotope property and usual shape of features as shown in Figure 2.10.

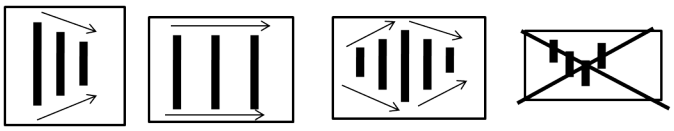


Figure 2.10: The left most three shapes represent peptide feature but the last shape is probably noise, therefore ignored in our method

Please refer to Figure 2.11, where we see a feature having charge $z = 2$, and its first isotope/trace is at x m/z with $(b - a)$ RT extent. It should get it's next isotope at $(x + .50)$ m/z, with overlapping RT extent $(d - c)$, satisfying the condition shown in the figure. It is found by our experiment that in 99% cases, this relation holds between two consecutive isotopes within a same feature. We

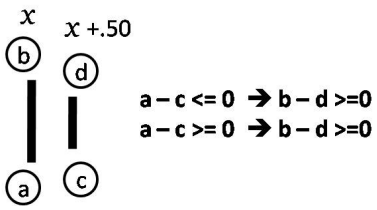


Figure 2.11: Condition between consecutive isotopes in a feature

Peptide Feature Table

ID		m/z	RT Range	z	AUC
1	Isotope 1	742.46	[17.81, 17.93]	1	201231.4122
	Isotope 2	743.46	[17.86, 17.92]		
2	...	...	...	.	...
3	...	...	...	.	...

Figure 2.12: List of detected peptide features

allow +-2 pixels distortion at the RT extreme points. This is also experiment based.

- Another property is that, for two consecutive isotopes A and B in a feature: Intensity of $A >= \frac{(\text{Intensity of B})}{3}$ (or the opposite). Otherwise we consider them as belonging to two different features.

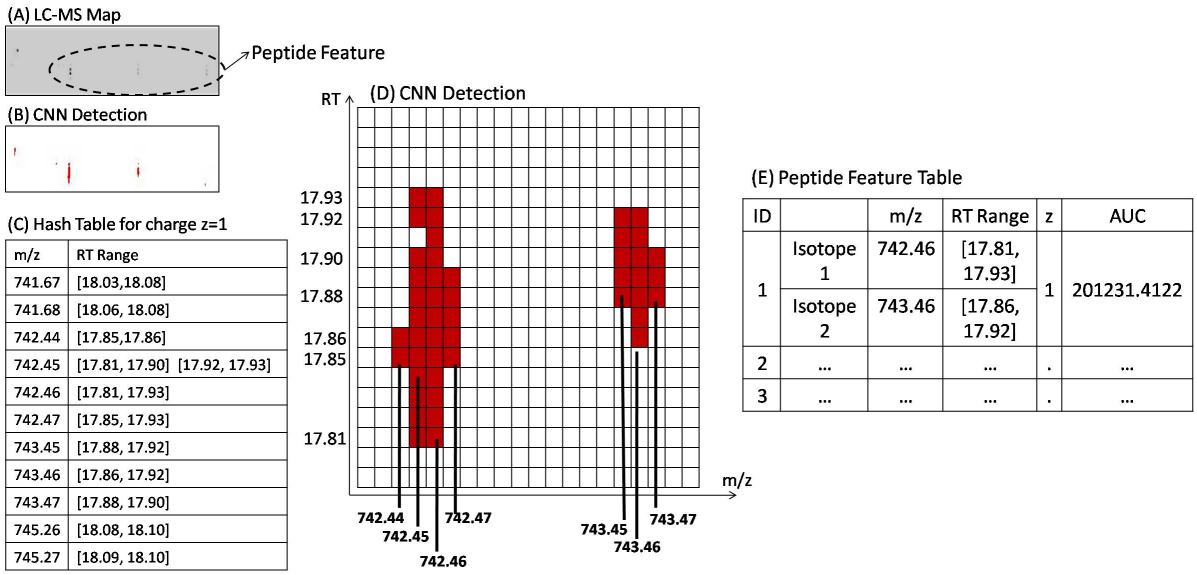
The isotopes holding these conditions are grouped in to one feature and inserted into a final list of detected peptide features. For example, the isotopes shown in Figure 2.9 are grouped in to one feature and listed as shown in Figure 2.12.

2.4.4 An Intuitive Example

To clarify the intuition of scanning phase we show an example in Figure 2.13 where a small region of LC-MS map holding a feature with charge $z = 1$ is shown (in (A)) and the result of CNN detection after scanning this region (in (B)) and corresponding records in hash table (in (C)) are shown as well. Records presenting the feature is further shown using a RT vs m/z plot (in (D)). After applying Step 3, the peptide feature is listed as shown in feature table (in (E)).

2.4.5 Discussion

The performance of our naive convolutional neural network in peptide feature detection reveals that the application of deep learning in this context has the potential of bringing success. Therefore, we became motivated for proceeding further with more effective CNN models and replace the heuristic part (for grouping the isotopes into features) of our model



m/z	RT Range
741.67	[18.03,18.08]
741.68	[18.06, 18.08]
742.44	[17.85,17.86]
742.45	[17.81, 17.90] [17.92, 17.93]
742.46	[17.81, 17.93]
742.47	[17.85, 17.93]
743.45	[17.88, 17.92]
743.46	[17.86, 17.92]
743.47	[17.88, 17.90]
745.26	[18.08, 18.10]
745.27	[18.09, 18.10]

ID		m/z	RT Range	z	AUC
1	Isotope 1	742.46	[17.81, 17.93]	1	201231.4122
	Isotope 2	743.46	[17.86, 17.92]		
2	...	...	...	.	...
3	...	...	...	.	...

Figure 2.13: Visualization of Step 2 and Step 3: (A) Sample LC-MS map, (B) CNN detection in Step 2 by scanning over sample LC-MS map, (C) Corresponding records in hash table after the scanning, (D) $m/z \times RT$ plot for the detected multi isotope pattern, (E) In Step 3, the pattern is listed as a peptide feature in the feature table

with a Recurrent Neural Network (RNN). This allows us to propose a fully automated deep learning model that learns all the parameters itself using appropriate training data. We introduce those models in the following chapters.

Chapter 3

DeepIso: A Deep Learning Model for Peptide Feature Detection from LC-MS Map

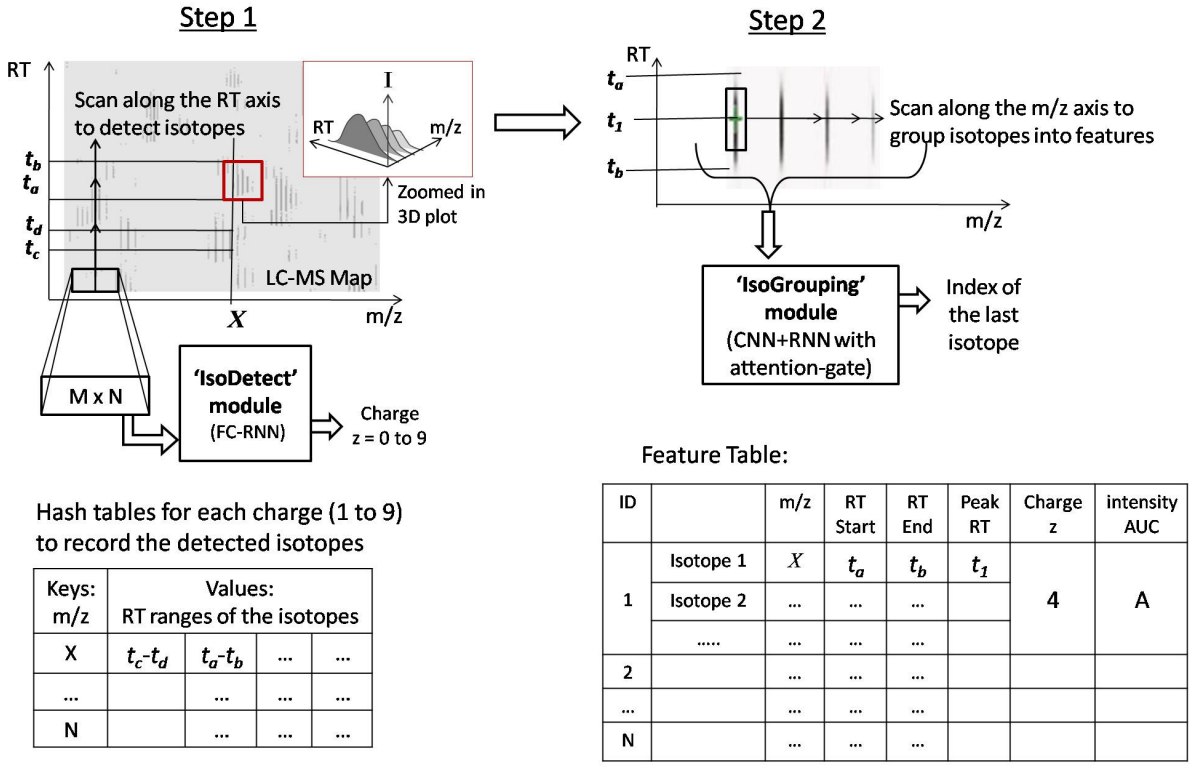
In the workflow of protein identification and quantification through LC-MS/MS technology, the first mass spectrometer (MS1) generates a three-dimensional plot, known as LC-MS map or MS1 data. The peptides in the protein sample are ionized inside the MS1, and a multi-isotopic pattern, i.e., peptide feature is generated in the LC-MS map each time a peptide ion is eluted from the sample. That is how we get over 50,000 peptide features in an LC-MS map. Scanning an LC-MS map through a sliding window can capture those features. In the previous approach using naive CNN, each sliding window independently decides about the existence of feature in a window, without using knowledge of previous time steps. As a result, if there are breaks in peptide feature signals, or the sliding window reaches the trailing region of the feature, it may miss the feature because of not knowing about the continuation of that respective feature from previous time steps. To overcome this drawback, **we propose a deep learning based model DeepIso [87], by adapting and combining FC-RNN [81] and temporal attention-gated [51] model to detect peptide features along with their charge states and estimate their intensities in LC-MS map.** It works in two steps. In the first step, IsoDetecting module spots the multi-isotope patterns and generates a list of detected isotopes along with their charge. In the second step, IsoGrouping module goes around the spotted region of interests, and groups multiple isotopes into a peptide feature. We use FC-RNN in IsoDetecting module because FC-RNN is a video classification model, where RNN is combined with CNN to deal with the patterns spanning over multiple time frames. This is exactly what happens in our

context. We can relate the task of IsoDetecting module with a video clip through which we watch all over the LC-MS map. Secondly, we adapt temporal attention-gated model in IsoGrouping module because originally the model was proposed for selecting frame of interests in a noisy and unsegmented sequence of frames (video clip). It can relate to finding the boundary of peptide features in a highly sparse and noisy LC-MS map. The peptide features generated by our model match with 97.43% of high quality MS/MS identifications in a benchmark dataset, which is higher than the matching produced by several other widely used tools. We will start with a block diagram showing the workflow of DeepIso. Then we will present the dataset used for the experiments and evaluation strategy along with the test results. After that, we will discuss our design strategy, some limitations of our model, and how to overcome those. Then we will provide the detailed architecture of the model and methods for the interested reader to reproduce this research work.

3.1 Workflow of DeepIso

We explain the schematic of our proposed model using the workflow shown in Figure 3.2. It consists of two steps and works on raw LC-MS maps without any preprocessing for noise removal. In the first step, the IsoDetecting module scans the LC-MS map along the RT axis to detect the isotopes having the potential of forming features. The scanning window is large enough to see the pattern of the isotopes and determine their charge states ($z = 1$ to 9) as well. The isotopes are recorded in a hash table. The class $z = 0$ represents noisy traces. In the second step, the IsoGrouping module goes to the region of detected isotopes and slides another scanning window along m/z axis to determine the starting and ending isotopes of a feature. Thus it produces a feature table that reports the detected features along with the m/z of monoisotope (the first isotope of a feature), charge, RT range of each isotope, and intensity.

In the first step, our job of scanning the LC-MS map along the RT axis resembles the video clip classification, where the RT axis is the time horizon. Therefore, we build IsoDetecting module combining CNN and RNN in a FC-RNN fashion proposed by Yang et al. [81], which achieved state-of-the-art results in the context of video classification on two benchmark datasets. In the second step, we develop the IsoGrouping module combining CNN with the attention-gated RNN proposed by Pei et al. [51]. We use attention gate in this module to concentrate more attention on the frame holding monoisotopes (first isotope in a peptide feature) while grouping the isotopes into peptide features. The IsoDetecting and IsoGrouping modules are trained separately using suitable training data.



Keys: m/z	Values: RT ranges of the isotopes			
X	t_c-t_d	t_a-t_b	...	...
...		...	...	...
N		...	...	...

ID		m/z	RT Start	RT End	Peak RT	Charge z	intensity AUC
1	Isotope 1	X	t_a	t_b	t_1	4	A
	Isotope 2	...	...	...			
	.....	...	...	...			
2		...	...	...			
...		...	...	...			
N		...	...	...			

Figure 3.1: Workflow of DeepIso to detect peptide features from LC-MS map of protein sample. In the first step, IsoDetecting module takes input a sliding window as $[M \times N]$ image and outputs the charge of the feature detected in the input. If the output is 0, it means no feature is seen. The scanning results are saved in hash tables and later used to generate a sequence of isotopes. That sequence is sent to the second module, IsoGrouping, to separate the adjacent features and discard noisy traces. The final results are saved in a feature table showing detailed information on the detected features.

3.2 Results

3.2.1 Dataset

We downloaded the benchmark dataset from ProteomeXchange (PXD001091), which was prepared by Chawade et al. [9] through LTQ Orbitrap XL ETD mass spectrometer(Thermo) with collision-induced fragmentation in the linear ion trap using top four data-dependent

acquisition (DDA). The samples consist of a long-range dilution series of synthetic peptides (115 peptides from potato and 158 peptides from human) spiked (injected) in a background of stable and nonvariable peptides, obtained from *Streptococcus pyogenes* strain SF370 [68]. This dataset was prepared to evaluate label-free quantification, i.e., measuring relative protein abundance among multiple samples using different software. Therefore, synthetic peptides were spiked into the background at 12 different concentration points resulting in 12 samples, each having a different concentration of spiked peptides. Again, each experiment was replicated multiple times for better feature coverage and intensity detection. We obtain LC-MS map (profile mode) from each replicate, totaling 57 LC-MS maps for the experiment. We cut peptide features from these maps for model training. We apply $k = 3$ fold cross validation [36] technique to evaluate our proposed model. To test each fold, we used 12 maps for model training and 4 maps for model validation. Model validation step is a part of training that is used to choose the best state of the model. In the following sections, we will first elaborate the training and validation sensitivity of the model. Then we will evaluate the performance of DeepIso by comparing it with existing tools.

The assessment criteria is set differently for training and testing. During training we measure what percentage of z charged features are classified correctly for $z = 0$ to 9. Here, $z = 0$ represents noisy traces. During testing, we assess the model based on what percentage of MS/MS identified features are detected by the model. The reason behind such criteria are already justified in Section 1.5.2 and Section 1.5.3 under the Introduction section. It is also briefly mentioned again in the following sections, whenever necessary.

3.2.2 Training of DeepIso Model

Since CNN and RNN are supervised learning approaches, we need labeled data for training. Human annotation of peptide features is out of scope due to gigapixel size of the LC-MS maps [66] and over 50,000 features in each map. Therefore, we run the feature detection algorithm of MaxQuant 1.6.3.3 and Dinosaur 1.1.3 on the LC-MS maps and then take the common set of feature lists generated by these two algorithms with a tolerance of 10 ppm m/z and 0.03 RT, as labeled samples for training and validation [66, 65, 57]. The total amount of samples collected in this way from each charge state is presented in Table 3.1.

First, we train the IsoDetecting module that tries to maximize the class sensitivity on the validation dataset. Here, class sensitivity is the percentage of samples detected correctly from each class, where classes belong to charge states $z = 0$ to 9. The charge state $z = 0$ represents the absence of features. The sensitivity of this class indicates how well the

Class (charge state)	1	2	3	4	5	6	7	8	9
Amount	163,038	863,050	428,909	29,183	1,503	653	179	236	233

Table 3.1: Class distribution of samples in our dataset consisting of 57 LC-MS maps. Amount of samples from charge 1, 2, 3, and 4 is 10.96%, 58.04%, 28.84%, and 1.96% respectively. Samples from the rest of the charges have less than 1% amount.

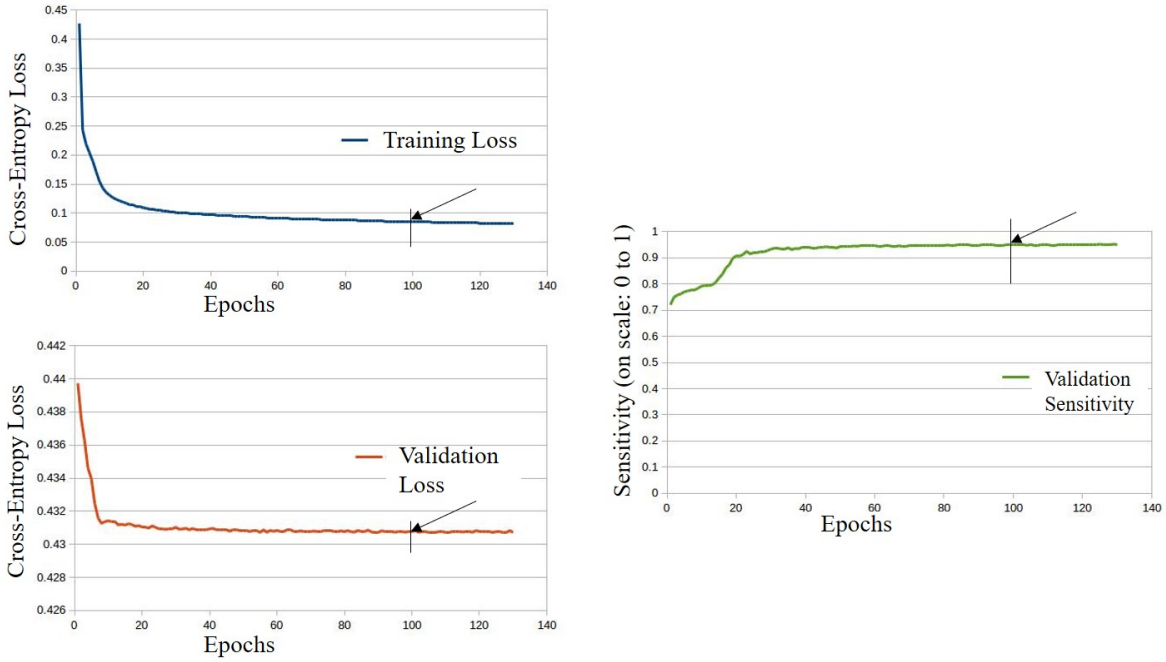


Figure 3.2: Learning curve for DeepIso. The cross-entropy loss for training and validation data is shown for 130 epochs. We see on the left that the validation loss does not change anymore after epoch 100, although training loss keeps on decreasing. We see the validation sensitivity also does not improve anymore after epoch 100. That means the model converges at about epoch 100. In this plot we show the average sensitivity of features from charge 1 to 5 because we have a very low amount of feature for charge above 5.

model distinguishes actual features from noisy traces and separates the closely residing features as well. Because of inadequate training data for features with charge states 5 to 9, as presented in Table 3.1, we had to apply data duplication and augmentation by oversampling (cutting multiple training windows from the same feature by translating the window at multiple surrounding regions) to increase training samples from these classes.

Class (z)	Training			Validation		
	Dataset Size	Sensitivity (%)	Precision (%)	Dataset size	Sensitivity (%)	Precision (%)
0	257,250	98.83	99.73	28,992	97.62	99.21
1	21,345	98.19	96.47	3126	94.53	96.23
2	131,951	98.94	96.94	26,480	98.18	95.91
3	59,045	99.26	95.95	10,903	97.95	94.12
4	6,765	99.38	96.07	646	95.72	92.23
5	4,140	98.28	97.36	20	86.59	82.32
6	8,446	99.91	96.48	30	40.36	18.04
7	3,324	94.28	97.87	10	50	16
8	4,060	99.66	97.44	15	61.79	61.80
9	4,203	99.69	96.58	14	28.21	76.74

Table 3.2: Class sensitivity and precision of IsoDetecting module and amount of samples for training and validation. The training set for class 5 has some duplicated samples. Training sets for classes 6 to 9 have augmented (oversampling) and duplicated samples. However, very low amount of original features from these classes (5 to 9) results in lower sensitivity for these classes due to over-fitting. The amount of samples from class 0 depends on our choice, and we keep this higher than other classes because the LC-MS map is very sparse. The validation set does not contain any duplicated data, and there is no overlapping between the validation dataset and the training dataset.

The average sensitivity and precision of the trained model on the training set and validation set are provided in Table 3.2. Due to the lack of variance in training data for charge states 5 to 9, the model's validation sensitivity for these classes is not close to 90% due to over-fitting. However, since most of the peptide features appear with charge states < 6, lower sensitivity for them does not impact the overall performance.

In the second step, the sensitivity of the IsoGrouping module is defined as the percentage of features reported with the correct number of isotopes. We categorize the training samples into five classes denoted A, B, C, D, and E. Class A associates with noisy traces that do not form any feature. Class B, C, D, and E correspond to features with 2, 3, 4, and 5 isotopes, respectively. We have not kept any 1 isotope category because usually, a peptide feature has at least two isotopes. Since the scanning window slides left to right, it can handle the cases when peptide features have isotopes over five (details are provided later in Section 3.3.3). We see the training and validation sensitivity in Table 3.3. We observe that the sensitivity of most of the classes is below 80%. To have a better perception, we present the confusion matrix in Table 3.4. The column A presents the percentages when monoisotope in a feature is missed. We see the model hardly misses the monoisotopes but confuses about the last isotope of a peptide feature. Please note that reporting the monoisotope along with the

Class	Sensitivity on Training Set (%)	Sensitivity on Validation Set (%)
A (noise)	95.06	94.68
B (2 isotopes)	56.49	57.52
C (3 isotopes)	72.24	72.41
D (4 isotopes)	72.69	74.23
E (5 isotopes or more)	72.41	72.67

Table 3.3: Class sensitivity of IsoGrouping module on training set and validation set. Please note that it does not relate to the charge/class $z = 0$ to 9. It shows how well the IsoGrouping module is able to group the isotopes to form the feature (whatever the charge is), and recognize the noises.

first few isotopes (having higher intensity peaks) of a feature is more important in the workflow (based on the discussion with collaborators from Bioinformatics Solution Inc.). Because they dominate the feature intensity and are used in the next steps of protein quantification and identification. But rest of the low-intensity isotopic peaks in a feature do not contribute much to quantification or identification. Therefore we accept a feature if the monoisotope along with high-intensity isotopes are reported correctly. Then we choose the state of the IsoGrouping module that maximizes the percentage of feature-matched MS/MS identifications on the validation dataset.

Class	A	B	C	D	E
A	94.68%	2.77%	1.73%	0.57%	0.25%
B	3.4%	57.52%	33.86%	4.59%	0.62%
C	0.89%	5.59%	72.41%	20.19%	0.93%
D	0.31%	0.89%	16.18%	74.23%	8.39%
E	0.79%	0.37%	2.70%	23.46%	72.67%

Table 3.4: Confusion matrix produced by IsoGrouping module on validation dataset. The diagonal values, e.g. [C, C] represent the sensitivity for class C. We say a feature is misclassified as class A when the monoisotope (first isotope) or all of the isotopes are missed, i.e., the feature is thought to be noise by mistake. The value of [C, A] indicates what percentage of features with three isotopes are either misclassified as noise, or monoisotope is missed. [C, B] indicates the percentage of features which actually have three isotopes but the third one is missed, and only first two are combined together. Similarly [C, D] shows for what percentage of three isotope features, IsoGrouping module finds ONE additional isotope at the end.

3.2.3 Testing of DeepIso Model

For performance evaluation, we present the percentage of high confidence (a confidence score is assigned to each identified peptide by database search tool) MS/MS peptide identifications matched with the peptide feature list produced by our algorithm. Since the identified peptides from MS2 data of a sample (i.e., the peptides for who we can find the amino acid sequence) must have corresponding peptide features in that MS1 maps, therefore, the more we detect features corresponding to them, the higher the performance [3, 66, 65, 57]. We run MASCOT 2.5.1 to generate the list of MS/MS identified peptides and the identifications with peptide score > 25 (ranges approximately from 0.01 to 150) are considered as high confidence identifications [3]. This list of peptides serves as ground truth data for performance evaluation. We observe the percentage of ground truth peptide features detected by different software.

In the testing phase, the first step is to scan the LC-MS map with the IsoDetecting module. Then we perform another scan with the IsoGrouping module through the potential patterns detected in the first step. It reports the final list of peptide features. To compare the performance of our model with other existing tools, we run the peptide feature detection algorithm of MaxQuant 1.6.3.3, OpenMS 2.4.0, and Dinosaur 1.1.3. We used the published parameter for MaxQuant as reported by Chawade et al. [9]. For Dinosaur, default parameters mentioned at their GitHub repository (https://github.com/fickludd/dinosaur) are used. For OpenMS, we use the python binding pyOpenMS [58, 57] and follow the centroided technique explained in the documentation[1]. For all feature detection algorithms, we set the range of charge states from 1 to 9 (or the maximum charge supported by the tool). Then the produced feature lists are matched with the high confidence MS/MS identifications with a tolerance of 0.01 m/z and 0.2 minute RT.

Algorithms	MaxQuant	OpenMS	Dinosaur	DeepIso
Matching	96.83%	97.14%	97.23%	97.43%

Table 3.5: Percentage of high confidence MS/MS identifications matched by feature list produced by different algorithms.

We have 12 samples where samples 2, 3, and 4 have seven replicates each, and the remaining samples have four replicates each. When we evaluate model sensitivity over a sample, we take the average sensitivity of all the replicates of that sample. Also, when we keep one sample for the training set (or testing set), all the replicates of that sample belong

[1]https://pyopenms.readthedocs.io/en/latest/feature_detection.html

to the same set. We show the average percentage of high confidence MS/MS identifications matched with the detected peptide features for 12 samples in Table 3.5. Although the performances are quite close, DeepIso is still a little bit ahead of all others. Moreover, DeepIso does not need a manual setting of parameters, whereas all other software needs field experts to set many parameters to make those work. Therefore, deep learning tools are desired to advance state-of-the-art techniques. It can report some features not detected by other tools as provided in the Venn diagram of feature-matched MS/MS identification by different tools in Figure 3.3. Later in Section 3.4, we discuss the scenario when our model might miss a feature and propose a potential solution to overcome the problem, thus increasing the sensitivity further as well.

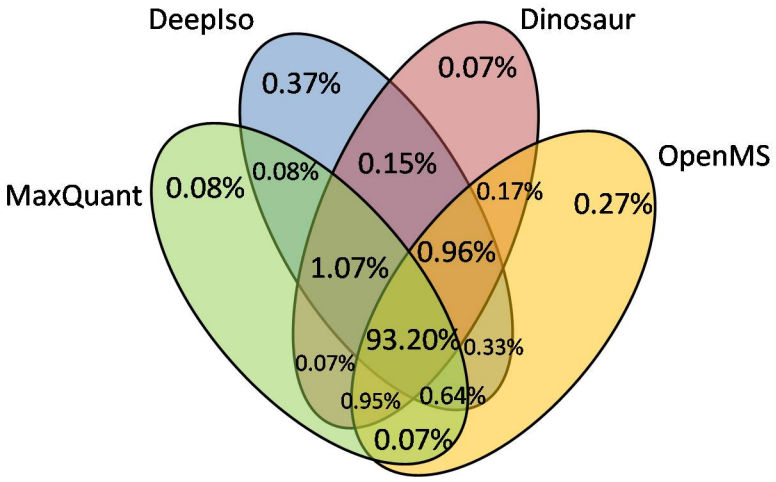


Figure 3.3: Venn Diagram of feature-matched MS/MS identification by different tools. The blue area shows that DeepIso is capable of finding some peptide features not detected by other tools.

We want to mention how fine-tuning with misclassified cases promotes better learning in our model. Please note that it is different from the backpropagation of errors. Fine-tuning in deep learning involves collecting samples (which may not have been provided during initial training) for which model makes mistakes and then retraining the model with those specific cases for a few epochs. This improves the model's ability to give the correct result next time. Obviously, we have to keep a validation set during this retraining or fine-tuning approach so that the model does not overfit those specific cases. We applied this retraining process while building the IsoGrouping module. The module was unable to separate adjacent features, e.g., feature 1 and 2, shown in Figure 3.10(c). We collected such cases and retrained the model which improved the overall matching by about 4%

(details are provided later in Section 3.3.5). Therefore, such model can learn from its own mistakes.

Peptide Feature Intensity Calculation by DeepIso

Next, we would like to verify the correctness of peptide feature intensity calculation by our model. For the statistical analysis of biological experiments the peptide feature intensity is of interest and has to be calculated from LC-MS map or MS1 data [66]. The steps are: apply curve fitting over the intensity signals of isotopes in a feature, calculate the area under those curves, and finally add those up to get the total intensity of that feature. (Intensity signal of an isotope is beta distribution if looked at [$RT\times$ Intensity] plane, but normal distribution if looked at [$m/z\times$ Intensity] plane.) Therefore the correctness of peptide feature intensity depends on whether the isotope boundaries are detected precisely or not. Please note that, peptide feature intensities are scalar values. So we can report the Pearson correlation coefficient of the peptide feature intensity between DeepIso and other existing algorithms in Table 3.6. It appears that our algorithm has a good linear correlation with other existing algorithms, which indicates that the relative protein abundance calculated by DeepIso will be similar to that by other software. Therefore, our model has the potential of being used in the label-free quantification pipeline, a protein quantification technique.

	DeepIso	Dinosaur	MaxQuant	OpenMS
DeepIso	1	0.8773	0.8899	0.9146
Dinosaur	0.8773	1	0.8657	0.9517
MaxQuant	0.8899	0.8657	1	0.8760
OpenMS	0.9146	0.9517	0.8760	1

Table 3.6: Pearson correlation coefficient of the peptide feature intensity between DeepIso and other tools. A coefficient value of close to 1 indicates that the models in comparison have a good linear correlation with each other. That means both of them may give similar label-free quantification results.

Time Requirement of DeepIso

The running time of the DeepIso model is the total time of scanning the LC-MS map by IsoDetecting module and IsoGrouping module. The running time of different algorithms, along with the platforms used in our experiment, is presented in Table 3.7. Our DeepIso

model has a higher running time than Dinosaur and Maxquant. However, we can improve this running time by increasing the available GPU for parallel processing. This does not need any change in our implementation since the number of parallel GPUs can be controlled with a parameter. Besides that, we also discuss some potential methods to speed up the DeepIso model later in Section 3.4.

Platform	Processor: Intel Core i7, 4 cores OS: Windows 10 for running the applications		Processor: Intel(R) Xeon(R) Gold 6134 CPU, NVIDIA Tesla OS: Ubuntu 16.04.5 LTS for running the python scripts	
Algorithms	Dinosaur	MaxQuant	**DeepIso**	OpenMS
Running Time	15 minutes	30 minutes	**1 hour and 40 minutes**	2 hours and 50 minutes

Table 3.7: Approximated running time of different algorithms. Here the platform used for OpenMS and DeepIso did not have support for running Windows application of MaxQuant and Dinosaur. So we used different machine for running those.

3.3 Architectural Details and Methods for Reproducing DeepIso

Our model runs the processing on raw LC-MS map collected in profile mode. We use the ProteoWizerd 3.0.18171 [8] in order to obtain the .ms1 format of the raw LC-MS maps. Then we read the file and convert it to a 2D grey scale image (i.e. RT $\times$ m/z plot) by treating the third dimension 'Intensity' as a grey value scaled between 0 to 255.

3.3.1 Step 1: Scanning of LC-MS map by IsoDetecting module to detect isotopes

This step is a 10-category classification problem according to our design. Please refer to the LC-MS map represented as a $m/z \times RT$ plot shown in Figure 3.4 for the clarification on the scanning process. Our network scans the LC-MS map as a sequence of $[M \times N]$ dimension frames, where each sequence is positioned at a point on the m/z axis (for instance, X), and the time steps range from the first MS-scan to the last MS-scan along the RT axis. We name a scanning through each sequence like this as one round of 'deep scan'. This figure shows a sequence passing over the isotopes of two features, features' a' and 'b' having charge '1' and '2' respectively. At each time step, the network outputs one of the classes in the range 0 to 9, 0 being the class indicating 'No Feature Seen', and 1 to 9 being the classes indicating features seen having corresponding charges. For instance, in the figure,

we use dotted arrows to indicate the network outputs (charge 1) at the corresponding time steps. The network outputs 0 in the blank spaces or noisy traces. Please note that the scanning window has dimension $[M \times N] = [15 \times 211]$ which is large enough to see the second isotope (along the m/z axis) in a potential feature to predict the charges. We do this to avoid using bidirectional RNN. The frames overlap to trace the RT range of the isotopes precisely.

Each unit of the RT axis represents one MS-scan (MS-scans are at least 0.01 minutes apart), and each unit of the m/z axis equals 0.01 m/z. However, each MS-scan does not hold signals from all m/z points. Therefore there are breaks in a sequence of 'deep scan' as shown in the LC-MS map (in Figure 3.4) by a broken line, where we pass the current RNN state to the next available frame. Please note that one 'deep scan' state is passed along the RT axis. Therefore, one 'deep scan' positioned at X m/z is independent of another 'deep scan' positioned at $X + 0.01$ m/z and vice versa. So we can process multiple 'deep scan's in a batch which makes the whole approach time efficient.

We keep nine hash tables for recording the detection coordinates (the points in the $RT \times m/z$ plot) of features from nine classes ($z = 1$ to 9) during the 'deep scan'. We need one hash table for each class because isotopes with the same charge are grouped together to form a feature. The m/z values of the isotopes are used as the key of these hash tables. The RT ranges of the isotopes in a feature are inserted as values under these keys as shown in the block diagram of Figure 4.1. Since the detection of wide isotopes may span over a range of m/z (i.e., multiple pixels along the m/z axis as shown for feature 'C' in Figure 3.13), we take their weighted average to select specific m/z of an isotope.

Network Architecture

The deep learning network is shown in Figure 3.4. The network is taking the frame at time step $t = t_1$ as input. There are three convolution layers, followed by two fully connected layers (denoted as i and o), one FC-RNN layer, and output is generated at each time step. The dropout layer is added after the third convolution layer and the first fully connected layer i with a value of 0.50, which is considered ideal for many cases. We use state size four and the tanh activation function. Since we are dealing with FC-RNN model [73], the state f_t at time step t is defined as below:

$$f_t = H(W_{io}.X_{it} + W_{hh}.f_{t-1} + b_o) \tag{3.1}$$

Wherein, H is the activation function, W_{io} is the weight matrix connecting the neurons of layer i to layer o (as shown in Figure 3.4), X_{it} is the output of the layer i at current

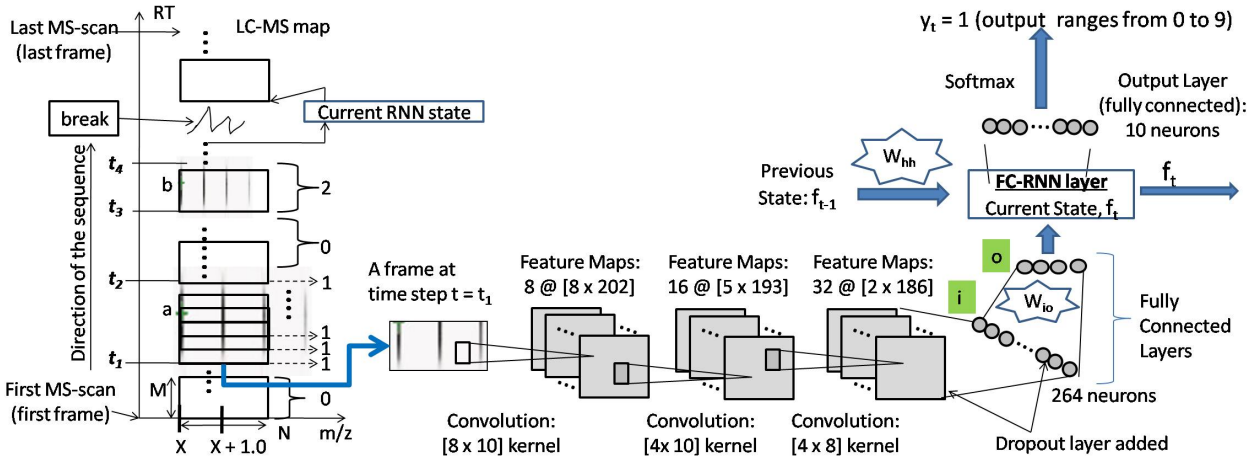


Figure 3.4: Network of IsoDetecting module. In the left we see some scanning windows going bottom to up. Then we show how a particular frame or window is passed through three convolution layers, 2 fully connected layers, one FC-RNN layer, and finally the Softmax output layer. The final output should be 0 to 9, indicating if the input frame has noise or feature having respective charge.

time step t, W_{hh} is the weight matrix of RNN state, b_o is the bias at layer o, and f_{t-1} is the previous state.

Training Procedure of IsoDetecting module

Now we will discuss about the training procedure of the network. It is supposed to learn the basic properties of peptide feature as mentioned in Section 1.1.1 [7], besides many other hidden characteristics from the training data. Training sequences are 20 frames long, i.e., each training sample is consists of 20 frames. Therefore, it covers 20 consecutive scans (who are at least 0.01 minute apart). Positive samples are created by cutting a sequence that is aligned with the monoisotope of the peptide feature as shown in Figure 3.5. The actual feature boundary is shown using dotted rectangle. The sequence starts 2 scans earlier than the actual start of the feature so that the network learns the gamma shape of intensities nicely. Similarly, the frames are positioned 10 ppm earlier than the given m/z of the monoisotope to make it error tolerable and let it see the whole isotope in case of wider isotopes. However, a peptide feature might span over less than 20 scans and that is why we deal with variable length sequences. We cut sequences from blank or noisy areas not holding features and treat them as negative samples. In this way we generate about 200k

positive samples and 200k negative samples. Please note that, our 'IsoDetect' network produces output at each time step. Therefore we label each frame of a sequence with one of the classes ranging from 0 to 9 (as shown 0, 0, 1, ... for the first three frames in the figure). We deal with variable length sequences since the peptide features might not span over 20 frames (scans).

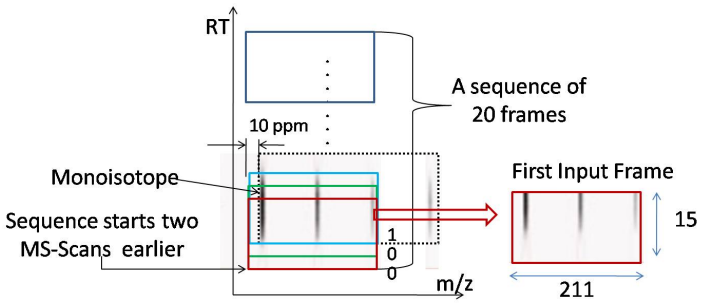


Figure 3.5: Training data generation for the 'IsoDetect' module

For charge states 6 to 9, we did not have enough samples for training. Therefore we applied data augmentation by oversampling. Please note that the 10 ppm tolerance along m/z axis let us cut the features couple of pixels before the exact start, as mentioned above. This number of pixels can vary from 0 to 2 based on the actual m/z. Since the LC-MS maps in our dataset span from 400 m/z to 2000 m/z (approximately), therefore for each sample having these charge states we can **cut multiple sequences** within the tolerance limit. For example, if a peptide feature with charge state 6 lies around 2000 m/z area, then tolerance limit is up to 2 pixels. So we cut sequences starting at exact m/z, 1 pixel (or 0.01 m/z) before, and 2 pixels (0.02 m/z) before. So we get three sequences for it. In this way we generate augmented samples.

We use 'Adam' stochastic optimization [32] with initial learning rate of 0.01. We use sparse softmax cross entropy as error function at the output layer. We run about 100 epochs and the model starts converging after about 90 epochs.

3.3.2 Intermediate Step to Make a Sequence of Isotopes

We use an intermediate step that forms sequences of closely residing isotopes having the same charge, overlapping RT extent but disjoint and equidistant from each other along the m/z axis. In other words, the equidistant isotopes of the same hash table are grouped into one sequence. For instance, we see two sequences, 'P' and 'Q' in Figure 3.6. We also

observe that the same sequence might hold multiple peptide features. This step is designed just to speed up the whole process by allowing batch processing in the second step. Each batch containing about 500 sequences is passed to the IsoGrouping module. Detecting the starting and ending of features in the sequences is handled by this module. Please note that this step is optional, and avoiding this step does not bring any significant change in the result. However, the running time of the IsoGrouping module increases drastically due to not utilizing the power of batch processing. We do not limit the maximum number of isotopes per sequence. Experiment shows that each sequence usually holds at most 16-20 isotopes.

3.3.3 Step 2: Scanning of detected isotopes by IsoGrouping module to report peptide feature

In this step, we place the frames at the isotopes of the sequences. For convenience, please refer to Figure 3.6. There are two significant differences between IsoDetecting and IsoGrouping modules. First, the IsoDetecting module scans the LC-MS map along the RT axis, whereas IsoGrouping module scans left to right, along m/z axis. Therefore, the time steps span along the m/z axis. Second, IsoGrouping module generates one output after seeing through five consecutive frames (after 5 time steps), unlike IsoDetecting module which generates output at each time step. Here, each sequence is processed in multiple rounds. Starting frame of one round depends on the output of previous round and the rounds can be overlapping as well. A step by step explanation of the scanning procedure with figure is provided below.

Step-by-Step Illustration

Let us consider sequence P in the figure. Each isotope is marked with its index, starting from 0. The frames are placed at five successive isotopes of the sequence (marked with a blue box and red-colored arrow). The output is generated at the last step, which can be one of the frame indexes: 0 to 4. In the first round, for instance, it outputs 3. Therefore, we get a feature starting at X m/z, with RT peak t_1, and 4 isotopes. Next, we place the scanning window at the 4^{th} isotope of the sequence, i.e., just the next one. This round sees through 4^{th} to 8^{th} isotopes of the sequence. In this round, the output is 4. Therefore, we should extend the counting of isotopes further to see if more isotopes belong to the current feature. So we start another round, but this time it starts from 8^{th} isotope (instead of 9^{th} isotope). After seeing through 5 subsequent frames, it outputs 1. That means we find the

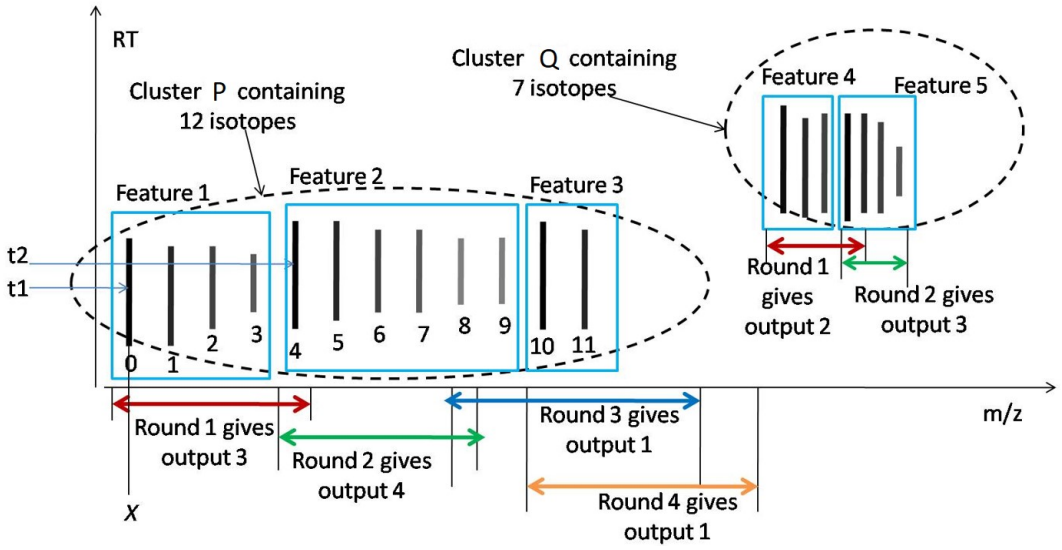


Figure 3.6: Intuition of scanning by IsoGrouping module on the sequences of features.

second feature starting at the 4^{th} isotope and ending at the 9^{th} isotope of the sequence. This scanning continues until all the isotopes of the sequence are seen. When we process multiple sequences in batch, then P and Q sequence are processed in parallel.

Network Architecture

We show the network in Figure 3.8. It has four convolution layers, followed by two fully connected layers. This time we include pooling layers after the first and second convolution layers. After each fully connected layer, the dropout layers are included with a dropout probability of 0.5. As shown in the figure, we input the charge z detected by the IsoDetecting module as a feature at the layer i. We do this by concatenating z with the output X_i of layer i. We use state size 8 and tanh activation function (ReLu and sigmoid activation did not work well according to our experiments). The current state f_t at time step t is calculated using attention gate a_t [42] as follows:

$$f_t = (1 - a_t).f_{t-1} + a_t.f'_t \tag{3.2}$$

Wherein, f_{t-1} is the previous hidden state, f'_t is the current state calculated in the conventional fashion and a_t denotes the importance of current frame to the final decision. The f'_t and a_t are calculated as below:

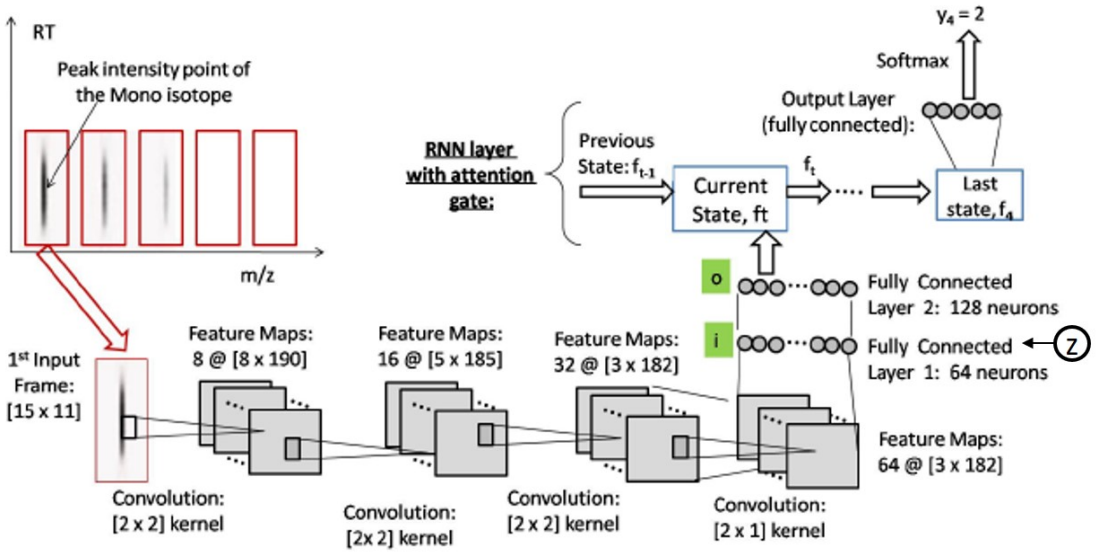


Figure 3.7: Network of IsoGrouping module. It shows how a frame in the input sequence is passed through 4 convolution layers, 2 fully connected layers, 1 RNN layer with attention gate, and finally Softmax output layer. The output decides about the feature boundary and also discards noisy frames in the input sequence.

$$f'_t = H(W_{hh}.f_{t-1} + W_{oh}.X_{ot} + b_h) \tag{3.3}$$
$$a_t = \sigma(W_a.f'_t + b_a) \tag{3.4}$$

In Equation 3, H is the activation function, W_{hh} is the weight matrix connecting the previous hidden state f_{t-1} to the current state, X_{ot} is the output of the layer o, W_{oh} is the weight matrix connecting the X_{ot} to the RNN layer, b_h is the bias at the RNN layer. In Equation 4, σ is the sigmoid activation function (ReLu and tanh activation did not work well according to our experiments), W_a is the weight matrix that learns the attention mechanism and b_a is the corresponding bias.

Training Procedure of IsoGrouping module

The monoisotope's intensity is the highest among the other isotopes in a feature. Therefore, in a sequence of isotopes we can identify the starting of a feature by finding the isotope having higher intensity. That should be the monoisotope (first or left most isotope in a

```
sequence_isotopes ← get a sequence of isotopes from the hash table
isotope_i = 0
start_index = isotope_i
end_index = start_index
While (isotopes left in sequence_isotopes):
        five_isotopes ← sequence holding [isotope_i to isotope_i+4]
        result ← IsoGrouping(five_isotopes)
        if result == 0:
            if end_index>start_index:
                new_feature ← form a feature using isotope at start_index to end_index
            isotope_i = isotope_i + 1
            start_index= isotope_i
            end_index= start_index
        If result > 0 and result <4:
            end_index = start_index+result-1
            new_feature ← form a feature using isotope at start_index to end_index
            isotope_i = isotope_i + result
            start_index= isotope_i
            end_index= start_index
         if result = = 4:
            end_index = start_index+result-1
            isotope_i = isotope_i + result – 1
        if new_feature != null:
             feature_table ← insert the new_feature
  if end_index>start_index:
        new_feature ← form a feature using isotope at start_index to end_index
        feature_table ← insert the new_feature
```

Figure 3.8: Pseudocode of IsoGrouping module. The actual script is uploaded at Github repository.

feature) and the intensity of the rest of the isotopes after that should decrease gradually.

Therefore, if there are multiple high intensity peaks in a sequence, like isotope 0, 4, and 10 in the sequence P in Figure 3.6, then those should be the breaking point of the sequence. The IsoGrouping module should learn this technique. We create the positive samples by producing a sequence of 5 frames for each peptide feature, where the sequence starts at the first isotope of the respective feature. Each frame has dimension [15 x 10], covering 15 scans along the RT axis and 10 units along the m/z axis. The frames are centered on the point associated with the peak intensity of the monoisotope, as shown in Figure 3.8. Each sequence is labeled by the frame index holding the last isotope of the feature (indexing starts from 0). If the feature has more than 5 isotopes, it is labeled as '4'. In this way, we generate about 220,000 positive samples.

We generate negative samples by cutting some sequences from the noisy or blank area. We also create sequences that contain peptide features, but the feature does not start at the first frame of the sequence. Those samples are labeled as '0' and considered negative samples. We do this to handle the cases where noisy traces are classified as isotopes by the IsoDetecting module by mistake and thus grouped with the isotopes of actual features in the intermediate step. We generate about 120,000 negative samples.

We apply 'Adagrad' stochastic optimization [17] with initial learning rate of 0.07. We use Softmax cross-entropy as an error function at the output layer. We validate the training of the IsoGrouping module based on the percentage of MS/MS identified peptide features of the validation LC-MS map reported by the module.

3.3.4 Ensemble of Multiple IsoGrouping Modules

To reduce variance, we use ensemble [86] of multiple IsoGrouping modules to report the peptide features. We generate four instances of the IsoGrouping module, which are different in terms of initial weights, learning rate (0.07, 0.08), state size (6, 8, 10), and size of the second fully connected layer (80, 128). Their outputs are combined using soft voting [21]. The ensemble technique improves the matching with identified peptides by about 0.33%. We ensemble four models presented in Table 3.8. This gives about 95.46% matching.

Models	Learning Rate	State Size	Last fully connected layer size	Individual Matching
1	0.07	8	128	95.43
2	0.08	10	128	95.22
3	0.08	10	80	95.36
4	0.09	8	128	95.19

Table 3.8: Models used for Ensemble

3.3.5 Fine-tuning DeepIso with Misclassified Features

Adjacent Feature case

This case appears when two features having the same charge state (e.g., z=2) reside one after another such that the distance between the last isotope of the first feature, and the monoisotope (first isotope) of the second feature, is equal to the inter isotope gap of those features ($\frac{1}{z}\ m/z$). Please see Figure 3.9(a) for clarification. We show two peptide features, 'p' and 'q', having the same charge and inner isotope gap of 0.50 m/z. This pair of peptide features create an adjacent feature case. Another pair of peptide features holding 'r' and 's' causing the same problem is also shown in this figure. The Adjacency Feature case might involve more than two peptide features in a row.

Misclassification of Adjacent Feature case

We noticed that the isoGrouping module was doing mistakes in separating such adjacent features while sliding the scanning window from left to right over these peptide features. A peptide detection is considered correct if the monoisotope is reported accurately. We redraw the 1st pair, peptide feature 'p' and 'q' as in Figure 3.9(b). Here we see how the isoGrouping module sees them during scanning. It just sees a bunch of isotopes, which we index as 0 to 5 for convenience. Following three types of mistakes were observed:

- Isotope 0 to isotope 3, these four isotopes are grouped as peptide feature 'p'. And then the rest two isotopes 4, and 5 were grouped as peptide feature 'q'. As a result, peptide feature 'q' misses the monoisotope, which is considered as missed feature. Because for a detection to be correct, the monoisotope of the peptide must be accurate.
- Only isotope 0 and isotope 1 are grouped as peptide feature 'p'. And then from isotope 2 to isotope 5, these 4 isotopes were grouped as peptide feature 'q'. This time although isotope 3 exists on peptide 'q', however the isotope 2 is reported as monoisotope of this peptide. This is also considered as missed feature, since monoisotope is wrong.
- Sometimes the isoGrouping module group all of the isotopes into just one big peptide feature 'p'. As a result the second peptide 'q' is again missed.

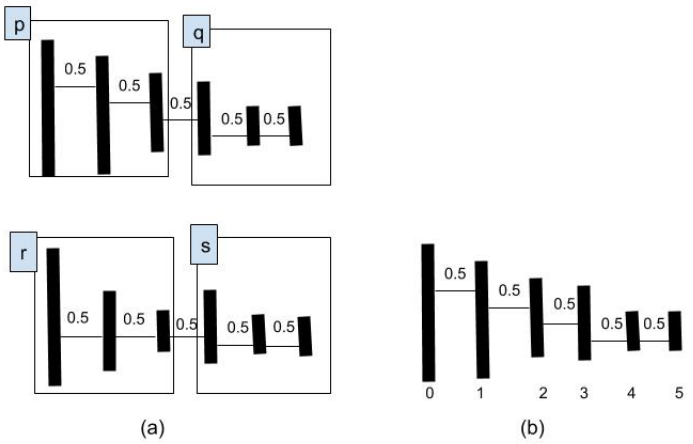


Figure 3.9: 'Adjacent Feature' problem

Retraining/Fine-tuning to teach Adjacent Feature case

For each fold we do the following steps:

- We select three samples, e.g., 10, 11, 12 for training the IsoGrouping module using the training set prepared as mentioned in Method section. Please recall that the percentage of feature matched MS/MS identification was used to verify the performance of IsoGrouping module. Two LC-MS maps, e.g., 9_01, 9_02 are used for validating the isoGrouping module. We first scan these two LC-MS maps using IsoDetecting module. The produced isotope lists are passed to the IsoGrouping module for validating through matching with MS/MS identifications.

- During training IsoGrouping module, after the model reaches saturation we chose the best state of the model based on MS/MS identifications matched with reported features for 9_01, 9_02. Then we find out the MS/MS identifications from 9_01, 9_02 for which we do not get any match due to adjacent feature case problem. We record that amount as N1 and N2 respectively. The N1 and N2 equal to about 8% of the MS/MS identified peptides.

- Then we do the same scanning (first by isoDetecting and then by isoGrouping) on samples 10, 11, 12 (which were used for training). We see that some of the peptide

features are missed as well from these LC-MS maps for adjacent feature case. We compare the feature list generated by DeepIso with the training feature list for these maps (common set of MaxQuant and Dinosaur) and find out about 18,000 adjacent feature cases which were not detected by our model. We cut sequences holding those cases. For example, each sequence would hold two or more peptide features like 'a' and 'b', as explained in above sections, and also can start from middle isotope of features (initial training sequences for IsoGrouping module consists of one peptide feature only and always start with the first isotope of features). Then we add this set of sequences with the previous training set (duplicated 5 times) and run the training again.

- This time we don't run training from scratch. We saved the model state every 10 epochs while doing the initial training. We choose the state saved at epoch 50, and start retraining from that point. By 90 epochs it approaches to saturation. We also keep track of what percentage of N1 is correctly detected this time. That amount comes close to 97% as the model approaches saturation. Then we choose the best state and save it. We use this retrained model on 9_02 again and see that over 95% of 'N2' are detected this time correctly.

Finally, we would like to mention the common strategies followed for implementing and training both of the modules. We implemented our deep learning model using the Google developed Tensorflow library. However, we had to build the RNN network ourselves instead of using their built-in RNN cells, in order to reflect the gating mechanism proposed in FC-RNN [81] and attention gated RNN cells [51]. During the training of both modules, we use minibatch size of 128 to ensure enough weight update in each epoch. We check the accuracy on validation set after training on every 10 minibatches. We perform data shuffling after each epoch which helps to achieve convergence faster. We continue training until no progress is seen on validation set for about 5 epochs. Including dropout layer in our model increases the validation sensitivity by about 1.5%. Although the Rectifier Linear Unit (ReLu) activation function is preferred over tanh in many literature, our model does not learn well with ReLu according to our experiments.

3.4 Discussion on the Design Strategy & Performance

We propose DeepIso, a peptide feature detection algorithm that does not apply human-designed heuristics involving centroiding, curve fitting, clustering, etc. Instead, it uses the power of deep neural networks to automate the learning of peptide feature detection by

revealing the important feature characteristics from LC-MS map. We will first demonstrate the justification of different design strategies followed and the utility of our model in industrial application. Then we will discuss some limitations of the current model and propose potential solutions to overcome the problems.

We would like to explain the significance of using RNN along with CNN for peptide feature detection. In the initial stage of this research, we used naive CNN (one discussed in the previous chapter) in IsoDetecting module, and set the unit along the RT axis as 0.01 minute. Only about 73% of the MS/MS identified features are reported in that technique, whereas, about 97% are reported in current model due to using RNN along RT axis. The RNN cells in the IsoDetecting module helps to detect the features having broken signals as shown in Figure 3.10(a). Another important thing is, RT values found in our data is sometimes over 0.01 minutes apart. For instance, 0.04 min, 0.07 min, 0.09 min, 0.13 min, and so on. Therefore, if we consider fixed 0.01 minute resolution along RT axis, there will be some blank row of pixels in the 2D image representation of LC-MS map. Such blank rows cause confusion for DeepIso during feature detection. That is why, instead of considering fixed resolution of 0.01 minute, we consider MS-Scans (those available RT reads: 0.04 min, 0.07 min, etc.) as units of RT axis. It also makes the whole scanning faster. Please see the table in Appendix A.3 for the experimental details.

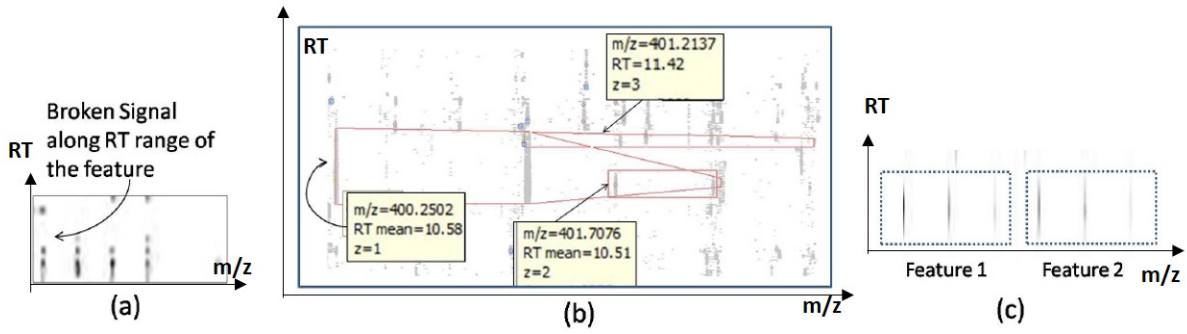


Figure 3.10: (a) A peptide feature with broken signals. RNN along the RT axis in the IsoDetecting module helps in detecting such features with broken signals; (b) Proper detection of overlapping peptide features by DeepIso model; (c) Adjacent feature case. Such features are used for fine tuning DeepIso.

Now we discuss about the reason of using simple RNN cells in IsoDetecting, instead of Long Short-term Memory (LSTM) [29] cells. Although the span of LC-MS map along RT axis is very long, the RNN does not need to look back very far in the past to detect an isotope, since each isotope's RT range is not very long. After start detecting a feature

(charge z with value 1 to 9), it has to remember the states upto the end of the feature ($z = 0$) only. After that it can refresh its memory. This is why we did not use LSTM cells to make the network unnecessarily complicated.

We do not use any pooling layer in the network of IsoDetecting module. In order to detect the sharp boundary and location of the peptide features, we want the network to have the property 'equivariant to translation' (ensured by CNN filters) to generalize edge, texture, shape detection in different locations, but not 'invariant to translation' (ensured by pooling layers) that causes the *precise location* of the detected features to matter *less*, and give unexpectedly wider detection for isotopes as presented in Figure 3.11).

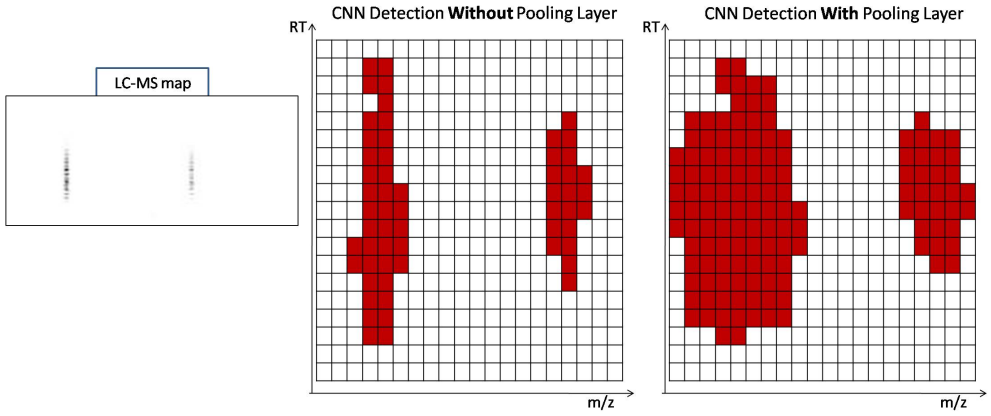


Figure 3.11: The effect of pooling layers in IsoDetecting module is shown. A peptide feature with charge 1 is shown in LC-MS map. When we use pooling layer to detect features, the isotope detections are wider, as shown in the right most image. But if we avoid using pooling layer, then the detections are thin and precise, as presented in middle image.

In the IsoDetecting module, the frame size of $[15 \times 211]$ (covering 15 scans and 2.11 m/z) ensures that it sees a reasonable area of a feature to decide about its existence along with the charge. If we reduce the frame size, we have to use two-dimensional and bi-directional RNN in the IsoDetecting module to look at the surrounding area. It prevents batch processing of multiple regions of the LC-MS map making the whole process time-consuming.

If we use attention-gated RNN in the IsoDetecting module, then it results in low class sensitivity, as apparent from Table 3.9. Therefore we chose the FC-RNN network for designing this module.

So far, we have discussed the methodology for training the IsoDetecting module. Now we will discuss the same for the IsoGrouping module, the second step in our DeepIso model.

Class (z)	0	1	2	3	4	5
FC-RNN network (better)	96.43%	93.80%	96.98%	98.74%	97.94%	85.86%
CNN with attention-gated RNN	96.15%	89.00%	96.04%	96.46%	95.07%	54.29%

Table 3.9: IsoDetecting module give better validation sensitivity with FC-RNN network than attention-gated RNN.

Model	Matching with MS/MS identified peptides
Initial model (about 430,000 parameters to learn)	87.55%
Retraining using Adjacent Feature cases	92.82%
Addition of max-pooling layers, one more fully connected layers and state size raised to 8 (about 167,000 parameters to learn)	94.66%
Network with attention-gated RNN (almost same amount of parameters as the previous one)	95.08 %
Same as above but trained with more data	95.13%
Ensemble of multiple instance of the model	95.46%

Table 3.10: Performance of IsoGrouping module in different stages of the development (based on validation dataset). Here, the initial model needs about 430,000 parameters to learn, whereas the more effective version shown in the third row needs only 167,000 parameters to learn. This is because we use max-pooling with stride $[2 \times 2]$. This pooling lets the model focus on the important features. Achieving higher sensitivity with a smaller model also demonstrates that simple and concise model works better than an unnecessarily big model.

We present different stages of the IsoGrouping module that we have gone through to achieve the current state of the model in Table 3.10. The matching with MS/MS identifications mentioned in this table is based on validation sample 9. We divide the experiments into following stages:

- Stage 1: This is the initial model designed with FC-RNN network with three convolution layers, one fully connected layer, without any pooling layer and state size 4. It gives about 87.55% matching.
- Stage 2: This is the model after retrained on Adjacent feature cases as mentioned previously in Section 3.3.5. It gives about 92.82% matching.
- Stage 3: The initial model was upgraded with max-pooling layer and one more fully connected layer as shown in Figure 3.8. The state size was also raised to 8. This

raised the matching to about 94.66%.

- Stage 4: Instead of using FC-RNN, we changed the gating mechanism as attention-gated RNN as explained in the Section 3.3 section. That is, we use Equation (2), instead of Equation (1) while implementing the RNN cells. This process gives about 0.4% improvement.

- Stage 5: So far we have been using peptide features from 8 LC-MS maps coming from two samples. Now we add one more sample which gives 4 additional LC-MS maps for training. Thus the amount of positive sequence in training set is increased by about 60,000. When Stage 4 was trained with this bigger dataset, we get about 95.13% matching.

- Stage 6: Here we just ensemble multiple trained IsoGrouping modules to get the final result. We ensemble four models presented previously in Table 3.8. This gives about 95.46% matching.

We also show a matching vs epoch plot in Figure 3.12. We run validation step every 10 minibatch after epoch 95 and select the state with maximum matching.

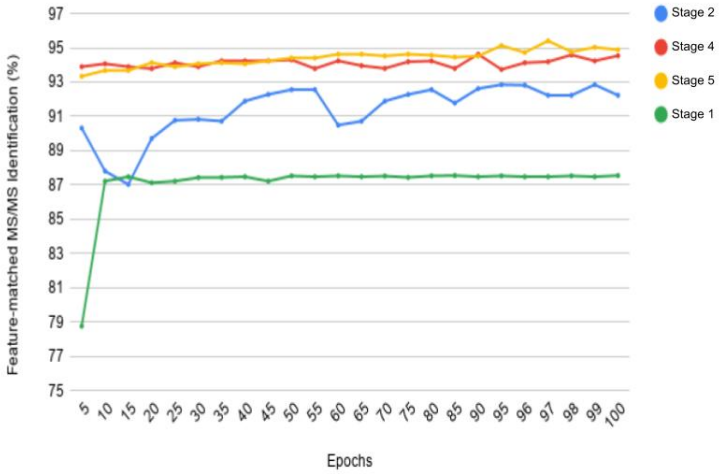


Figure 3.12: Matching vs epochs plot for validation LC-MS map 9_01

Finally, we increase the sensitivity of peptide feature detection for charge states 6 to 9 as presented in Table 3.11. This mostly involves IsoDetecting module. Since the original

amount of samples from these classes were negligible according to Table 3.1, we had to apply data oversampling (equivalent to applying more penalty for misclassifying samples from lower abundant classes) and augmentation by oversampling (see Section 3.3.1 for details) to train the module on these classes. It improves the final matching with MS/MS identification as presented in Result section above.

Class (z)	Initial Dataset		Oversampling was performed by duplicating training samples ($z = 6$ to 9) 10 times		Augmented samples were created from training samples ($z = 6$ to 9) and then duplicated 10 times	
	Validation Sensitivity	Training Sensitivity	Validation Sensitivity	Training Sensitivity	Validation Sensitivity	Training Sensitivity
6	0	0	52.65%	98.32%	40.36%	99.9%
7	0	0	0	96.53%	50%	94.1%
8	0	0	31.67%	99.14%	61.80%	99.6%
9	0	0	38.57%	98.12%	28.20%	99.7%
Comments	Network does not learn anything due to negligible amount of original samples.		Although the network starts learning because of introducing higher penalty for lower abundant samples, however it still does not learn well due to lack of variance in samples. Network also overfits due to lack of data.		Various samples were created using augmentation which improves the validation sensitivity further. However, the network still overfits due to lack of data amount and variation.	

Table 3.11: Improvement of class sensitivity of IsoDetecting module for charge states 6 to 9 with increasing amount of training samples. We do not bother for further improvement (by including more data from different but similar dataset) since most of the peptide features generally appear in LC-MS map with charge states < 6. Here the validation set does not contain duplicated data and there is no overlapping among the training set and validation set.

In our research, it is more important to detect as much true features as we can. In industry, sometimes it is more important to not missing any high intensity features, according to the discussion with Bioinformatics Solutions Inc. To see how well all the software detect high intensity peptide features, we sort the peptide feature list generated by different algorithms based on descending order of peptide feature intensity. Then we select the top 10,000 peptide features (about 20% of the existing peptide features in each LC-MS map, which is about 50,000) from each list and denote them as high confidence feature list. Finally we compare that list with the high confidence MS/MS identifications. DeepIso provides 89.32% matching which is higher than Dinosaur (89.24%), MaxQuant (87.65%), and OpenMS (60.44%). The performance of OpenMS is lower than others, because it produces some high intensity false positives. We believe, the good performance by DeepIso makes it a suitable model for industrial sector as well.

Now, we would like to refer to some scopes of improvement in our proposed DeepIso model. Visual observations at some peptide features on LC-MS map discover that some

features are missed due to the low resolution (2 digits after the decimal point) considered for m/z axis. Although we are able to teach DeepIso to detect overlapping features as shown in Figure 3.10(b), detection of some closely residing peptide features (with close monoisotopic peaks) in the LC-MS map, for instance, feature A and feature B in Figure 3.13(a), are merged together. However, if we increase the resolution as shown in Figure 3.13(b), then the features are separated in LC-MS map and thus isolated by IsoDetecting module as well. Therefore, increasing the resolution will let IsoDetecting module see the LC-MS map better and result in higher sensitivity. However, in that case we will need to sacrifice the time efficiency since increasing resolution by one decimal point, for instance 0.01 to 0.001 will make the input frames 10 times bigger in dimension and eventually resulting in larger feature maps, turning the model slower than before. Therefore we have to find an intelligent architecture that will let us increase the resolution without compromising running time. One potential approach might be using PointNet [54], which avoids 2D image representations of 3D objects, and directly works on the point cloud (set of data points in space). Besides that, time efficiency is also an important factor considering the practical utility. The running time of DeepIso is dominated by the first step, IsoDetecting module. Because it has to scan the whole LC-MS map represented as a 2D image of gigapixel size (about [12,000 x 140,000] pixels considering LC-MS map ranging from 400 m/z - 1800 m/z and up to 120 min along RT axis). Therefore one of our next concern is to make the IsoDetecting module time efficient. Designing the IsoDetecting module as a segmentation network might be helpful in this case. Besides that, whether using 'BERT' [15] technique (semi-supervised learning technique in the context of natural language processing) in implementing IsoGrouping module brings better performance is also left for future research. Application of DeepIso in label-free quantification (LFQ) can be another direction of work. These research scopes are addressed in the next chapters.

3.5 Data & Code Availability

The benchmark dataset is available to download from ProteomeXchange using accession number PXD001091.
The github repository is available here: https://github.com/anne04/deepIso

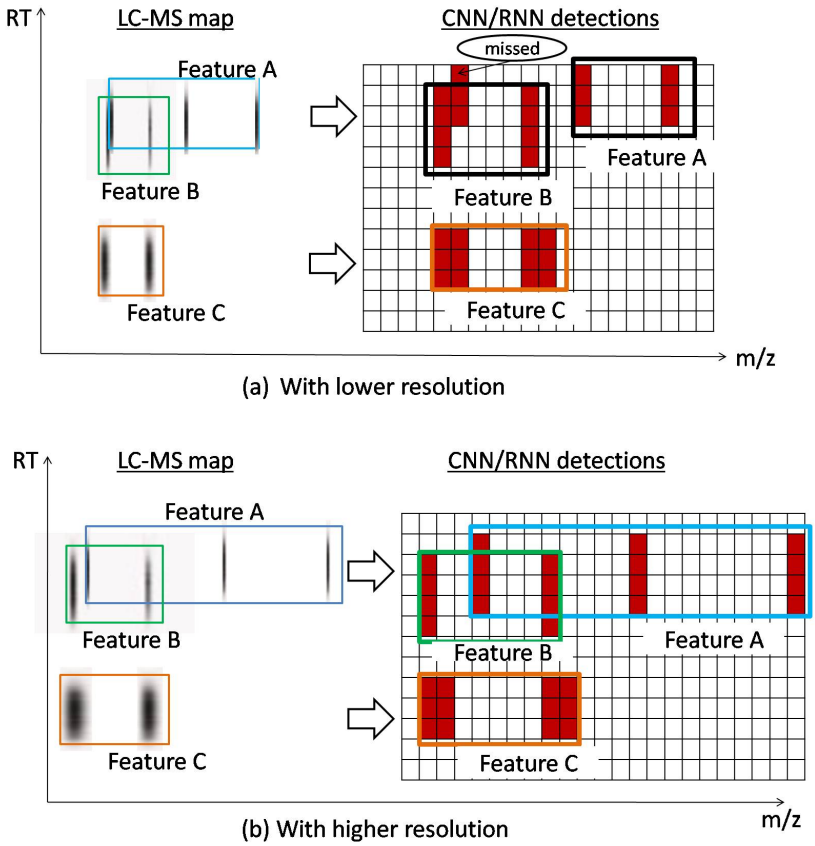


Figure 3.13: Intuitive image showing the effect of resolution along the m/z axis of LC-MS map: (a) Lower resolution merges the closely residing peptide features. For example, the 1st isotope of feature A and B are merged together. As a result, the monoisotope of feature A is missed by the model; (b) Higher resolution separates the first isotope of feature A and B. Therefore, IsoDetecting module can perform correct detection.

Chapter 4

PointIso: Point Cloud Based Deep Learning Model with Attention Based Segmentation

Peptide feature detection from LC-MS map is a crucial step in the downstream workflow of protein quantification and biomarker discovery. We proposed DeepIso [87], the first deep learning model which combines recent advances in CNN and RNN to detect peptide features of various charge states and estimates their intensity. It gives better sensitivity than other existing tools. However, it has two limitations. First, because of using image-based CNN, DeepIso is a fixed precision model (up to 2 decimal places) and is not feasible for high-resolution and higher-dimensional data. But LC-MS data are usually of very high resolution (or arbitrary resolution) and also higher dimensional sometimes (over 3D). Therefore, we need a model that supports high resolution data. Second, DeepIso is also comparatively slower than other competitive tools because of using a classification network in an overlapping sliding window approach for doing the feature segmentation. However, despite some limitations, DeepIso gave us a good insight into the scope of deep learning in this context. **Therefore, we bring significant changes in DeepIso to overcome these problems and offer PointIso [88], that accepts arbitrary resolution data (can be of very high resolution) and gives precise feature boundary information in a time-efficient manner. PointIso achieves a higher percentage of feature detection than other existing tools and is three times faster than DeepIso.** In particular, we change the image based classification network to a *point cloud* based *segmentation* network. The point cloud is a data structure for representing objects using points, e.g., using triplets in a three-dimensional environment. We combine point

cloud based deep neural network PointNet [54] and Dual Attention Network (DANet) [20] to integrate local features with their global dependencies and some context information. Unlike DeepIso where 2D projected images are used for representing 3D peptide features, we adapt PointNet to our context in order to directly process the 3D features. It makes it feasible to accept input data with two or more times higher resolution (arbitrary-precision) than DeepIso and achieves better detection. Also, moving from image-based CNN to point cloud based model prevents the model from getting unnecessary voluminous with the high resolution data and speed up the model. On the other hand, the original DANet is proposed for finding the correlated objects in the input landscape image for the autonomous driving problem using *attention* mechanism. We take the idea and plug it into the PointNet network to solve boundary value problems during scanning the huge LC-MS map through *non-overlapping* sliding windows and improve the running time by avoiding redundant calculation caused by overlapping. This novel concept of *attention* based scanning through a *non-overlapping* sliding window also has the potential to serve the general image processing problems by improving time complexity and segmentation accuracy. Therefore, we believe PointIso makes a notable contribution in accelerating the progress of deep learning in proteomics area, as well as, general pattern recognition study.

4.1 Workflow of PointIso

We explain the intuition of our proposed model using the workflow shown in Figure 4.1. We see the three-dimensional LC-MS map in the upper left corner, and PointIso starts by scanning this map by sliding a window in two directions: m/z axis and RT axis. The third axis tracks the signal intensity, I. A sliding window (or a target window) is essentially a 3D cube of point cloud. The PointIso model works through two modules, IsoDetecting in the first step and IsoGrouping in the second step. So the point cloud input consists of a set of 'N' datapoints which is passed as input to the IsoDetecting module as shown by the arrow sign from the sliding 3D window. IsoDetecting module segments the datapoints as $z = 0$ to 9, where $z = 0$ means the respective datapoint belongs to noise or background, and $z = 1$ to 9 means the respective datapoint belongs to a feature having charge z. We build this module by incorporating the attention mechanism offered by DANet into the PointNet architecture, to support *non-overlapping* sliding windows. The IsoDetecting module produces a list of isotopes of potential features that is recorded in a hash table. Then in the second step, IsoGrouping module takes those sequences of isotopes (each sequence may contain isotopes of multiple adjacent features) and predicts the boundary (first and last isotope) of features. Intuitively, it extracts multiple features

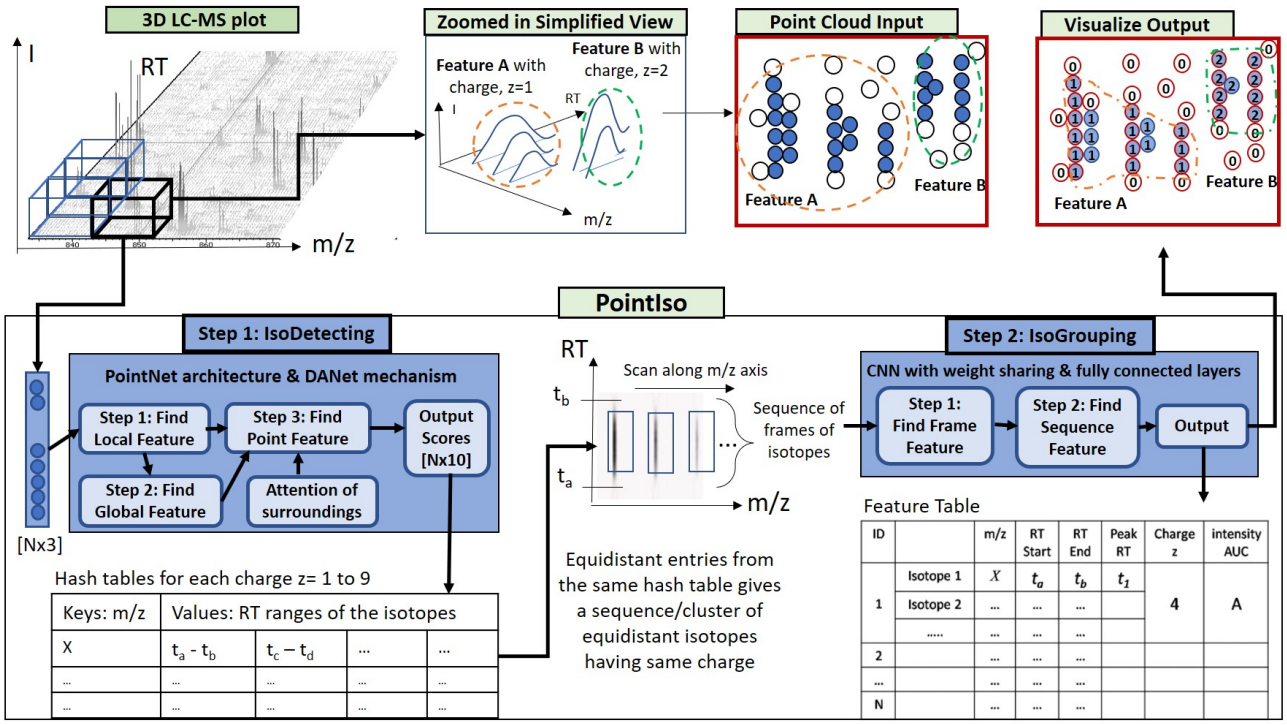


Figure 4.1: The workflow of our proposed model PointIso to detect peptide features from LC-MS map of protein sample. In 3D LC-MS plot we show a random scanning window in bold black boundary, enclosing two features. This region is further shown in the next image, labeled as 'Zoomed in Simplified View'. Here, two features A and B are shown using orange and green boundary, each having multiple isotopes (although smooth beta distributions of the isotopic signals are shown for simplicity, practically they are distorted due to noise). The corresponding point cloud version of this window is shown in the next image, labeled as 'Point Cloud Input'. Here the blue and white points correspond to the features and background points respectively. The 'Visualize Output' shows the PointIso predicted labels for the datapoints in that window. The datapoints labeled as '1' belong to feature A having charge 1. And the datapoints labeled as '2' belong to feature B having charge 2. The background or noisy datapoints are labeled '0'.

from the input sequence or discard the noises. This prediction finally gives us a feature table that reports the detected peptide features along with the monoisotopic m/z (the first isotope of a feature), charge, RT range of each isotope, and intensity. We can also visualize the final result as shown in the image labeled as 'Visualize Output' in Figure 4.1 (upper right corner).

4.2 Results

4.2.1 Dataset

We downloaded the benchmark dataset from ProteomeXchange (PXD001091), which was prepared by Chawade et al. [9] through LTQ Orbitrap XL ETD mass spectrometer(Thermo) with collision-induced fragmentation in the linear ion trap using top four data-dependent acquisition (DDA). The samples consist of a long-range dilution series of synthetic peptides (115 peptides from potato and 158 peptides from human) spiked (injected) in a background of stable and nonvariable peptides, obtained from *Streptococcus pyogenes* strain SF370 [68]. This dataset was prepared to evaluate label-free quantification, i.e., measuring relative protein abundance among multiple samples using different software. Therefore, synthetic peptides were spiked into the background at 12 different concentration points resulting in 12 samples, each having a different concentration of spiked peptides. Again, each experiment was replicated multiple times for better feature coverage and intensity detection. We obtain LC-MS map (profile mode) from each replicate, totaling 57 LC-MS maps for the experiment.

4.2.2 Training of PointIso

Since we use a supervised learning approach, we need labeled data for training. Human annotation of peptide features is out of scope due to the gigapixel size of the LC-MS maps [66]. Therefore, we match the feature lists produced by MaxQuant 1.6.3.3 and Dinosaur 1.1.3 with a tolerance of 10 ppm m/z and 0.03 minute RT and take the intersection set as the training samples. In PointIso, we also need the precise boundary information (i.e., RT time range and m/z value of each isotope of the features), which is not generated for the users in MaxQuant. Therefore we use Dinosaur for that information. The IsoDetecting and IsoGrouping modules are trained separately using suitable training data. To generate training samples for the IsoDetecting module, we place a scanning window over the features and cut the region and the surrounding area. The total number of features available for charge $z = 1$ to 9 are provided later in Table 4.5. The input resolution of our dataset is up to 4 decimal places along m/z axis (whereas DeepIso accepts only 2 digits after the decimal point). For training the IsoGrouping module, we cut a sequence of frames (each frame holding an isotopic trace) from these peptide features. The training data generation technique is further explained in Section 4.4. We apply $k-$fold cross-validation [36] technique to evaluate our proposed model which is elaborated in Appendix A.2. We consider two settings, one with $k = 2$ and another with $k = 6$. In $k = 2$, only 30% data was used

for training and the rest for testing. In $k = 6$, about 80% data was used for training and the rest for testing. We show the consistency of model sensitivity by using these different settings of experiments.

4.2.3 Performance Evaluation of PointIso

We run MASCOT 2.5.1 to generate the list of MS/MS identified peptides, where we consider the identifications with peptide score > 25 (ranges approximately from 0.01 to 150) as high confidence identifications [3]. For performance evaluation, we compare the percentage of high confidence MS/MS peptide identifications matched with the LC-MS peptide feature list produced by our algorithm and some other popular algorithms. Since the identified peptides must exist in LC-MS maps, therefore, the more we detect features corresponding to them, the better the performance [3, 66, 65, 57]. The other tools used for comparison are MaxQuant 1.6.17.0 [13], OpenMS 2.4.0 [57], Dinosaur 1.2.0 [67], and PEAKS Studio X [43]. The creator of our training dataset [9] used MaxQuant for processing the data, and we use the same parameter as them. For Dinosaur, default parameters mentioned at their GitHub repository (https://github.com/fickludd/dinosaur) are used. For OpenMS, we use the python binding pyOpenMS [58, 57] and follow the centroided technique explained in the documentation. For all of the feature detection algorithms, we set the range of charge state 1 to 9 (or the maximum charge supported by the tool).

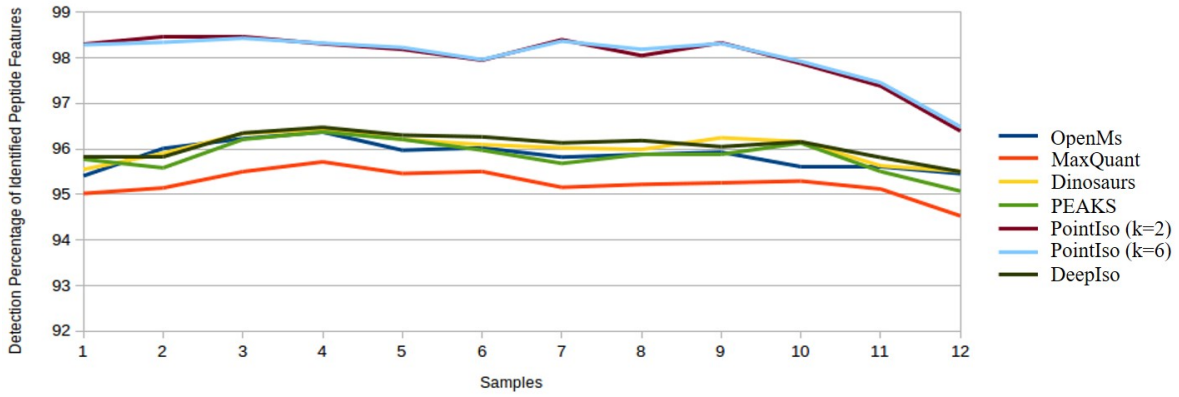


Figure 4.2: Detection percentage of identified peptide features by different tools for 12 samples is shown. We see that PointIso gives about 2% higher detection than all other software. Both settings $k = 2$ and $k = 6$ give almost similar performance. Maybe the reason is, smaller training set (as in $k = 2$) is good enough for achieving model convergence.

Percentage of Identified Peptide Features Detected by PointIso

We show the plot of detection percentage of high confidence MS/MS identifications with error tolerance of 0.01 m/z and 0.2 minute peak RT [87, 67, 9] for 12 samples by different algorithms in Figure 4.2. We see that PointIso (for both $k = 2$ & $k = 6$ folds) has a significantly higher detection rate for all the samples. The average detection rate of PointIso is 98.01% for $k = 2$, and 98.02% for $k = 6$, as presented in the first row of Table 4.1 (the entire result can be found in Appendix C.1). Besides the monoisotopic peak position, we also match the features in terms of charge z in the second row, and with all the MS/MS identifications (any score) in the third row. We see that our algorithm consistently provides a higher detection rate than other tools. Total number of features generated by different software are discussed later in Section 4.3.6. Please note that, during matching the LC-MS peptide features to the identified MS/MS spectra, multiple peptide features may map to the same peptide sequence. In DeepIso, we merge those multiple MS/MS spectra entries into one entry, whereas in PointIso we treat each of those entries as separate entity during calculating sensitivity.

Matching Criteria	MaxQuant	OpenMS	Dinosaur	Peaks	DeepIso	**PointIso (k=2)**	**PointIso (k=6)**
Match (m/z, RT) with identifications having high confidence score	95.2434%	95.8586%	96.0068%	95.8559%	96.0534%	**98.0080**%	**98.0246**%
Match (m/z, RT, z) with identifications having high confidence score	95.0728%	95.4941%	95.6654%	95.6993%	95.7639%	**96.7477**%	**97.0684**%
Match (m/z, RT) with all identifications (any score)	93.7312%	94.0335%	94.9049%	94.8216%	94.1011%	**96.9832**%	**96.9616**%

Table 4.1: Detection Percentage of MS/MS identified peptide features by different methods. PointIso gives about 98% sensitivity (2% higher than other tools) when we match the detection to the high confident identifications in terms of m/z and RT, as shown in the first row. If we also match in terms of same charge z, then the sensitivity goes down for all the software by about 1%, but PointIso is still giving better sensitivity, as shown in the second row. In the third row, we show the model sensitivity taking into account all the MS/MS identified peptide features irrespective of their score, and PointIso is consistantly showing better performance than other tools.

Percentage of Identified Spiked Peptides Detected by PointIso

The previous experiment shows the average of both spiked and background peptides. Next, we show separately, what percentage of identified spiked peptides, i.e., potato peptides and

human peptides are detected by PointIso and other tools, in Table 4.2. It gives a better insight on the performance since detection of spiked peptides is more important. We see that, PointIso detects 3-4% higher human peptides and 5-6% higher potato peptides than other methods.

Peptides	MaxQuant	OpenMS	Dinosaur	Peaks	**PointIso**
Human Peptides	94.16%	93.96%	94.71%	93.84%	**97.25**%
Potato Peptides	92.96%	93.55%	93.17%	92.92%	**98.32**%

Table 4.2: Detection Percentage of MS/MS identified spiked peptides by different methods. Matching is performed by comparing the monoisotopic peak (m/z, RT) and charge z of detected features with the identified spiked peptides. PointIso is giving 3%-4% higher detection of human peptides and 5%-6% higher detection of potato peptides.

During 2-fold cross validation, the model is trained on mostly human peptide features, e.g., using sample 9 to 12, and the rest were used for testing. Although PointIso did not see much of potato peptides during training, it can still detect about 98% potato peptides during testing in sample 1 to 8, i.e., features coming from *different species*. We also test our trained PointIso model on breast cancer proteomic data [24] (file P1_LN_1.RAW of project PXD012431 from ProteomXchange). We see that our model detects 72.6% identified peptide features, whereas, Dinosaur and MaxQuant detect only 63% and 50% identified peptide features respectively. It implies that our model can well generalize the peptide feature properties irrespective of peptide patterns or intensities seen during training time. One important fact is, the dataset should be generated by the same type of instrument that was used for generating our training data: LTD Orbitrap XL ETD. If the instrument changes, then the patterns may also change a bit, and thus, PointIso may not give optimal result according to our experiment (usual behavior for any deep learning model). However, few epochs of fine tuning may solve that problem as well. Therefore, once we train a model on a protein sample, the same model should be applicable to other protein samples coming from different species but similar instrument, making it more appealing in the practical sectors.

Venn Diagram of Peptide Features by Different Tools

Venn diagram of identified peptide features detected by different algorithms (we show four algorithms to keep the Venn diagram simple) is shown in Figure 4.3. We see that there is about 2.58% peptide features which are detected exclusively by PointIso. The comparison

with DeepIso is shown in Appendix D.2. Some illustrative examples of features which are detected by PointIso but missed by other tools are shown in (b), (c), (d), and (e). In (b), peaks connected by the black line are detected as a peptide feature by all algorithms. However, the feature having a lower peak, connected by the orange line, is missed by other tools. Because they have the same m/z as the one with a higher peak. Therefore, most of the tools merge it with the bigger one during pre-processing steps. In (c), we see that, all other tools except PointIso merge the monoisotope (first isotope) of feature C (enclosed in the orange rectangle) with the second isotope of feature A. As a result, all other tools report two features here: A and B. But, B is just a fraction of actual feature C, therefore, it is counted as missing feature C. However, PointIso can detect A and C precisely. Then we see that, very closely residing features like (d), and features with broken signals like (e) are sometimes missed by other tools, but detected by our model.

4.2.4 Peptide Feature Intensity Calculation by PointIso

The correctness of peptide feature intensity depends on whether the isotopic signals are detected precisely or not. The Pearson correlation coefficients of the peptide feature intensity (area under the isotopic signals of peptide feature) between PointIso and OpenMS, MaxQuant, Dinosaur, PEAKS are respectively 93.76%, 95.31%, 89.88%, and 88.93%. Our algorithm has a good linear correlation with other existing algorithms, which validates the correctness of peptide feature boundary or area detection by our model.

4.2.5 Time Requirement of PointIso

The total time of scanning the LC-MS map by IsoDetecting module and IsoGrouping module is the running time of PointIso model. However, IsoDetecting module dominates the running time. We present the running time of different algorithms and the platforms in Table 4.3. PointIso model is about three times faster than DeepIso and has a comparable running time with most of the existing tools. PEAKS is much time-efficient than all other algorithms. However, we believe that the PointIso can be made faster as well, by using multiple powerful GPU machines in parallel. This does not need any change in our implementation since number of parallel GPUs can be controlled with a parameter. For example, with 1 GPU and without any parallel processing, PointIso takes about Two and half an hour. But with 3 GPUs and parallel processing PointIso takes about 30 minutes as shown in the table.

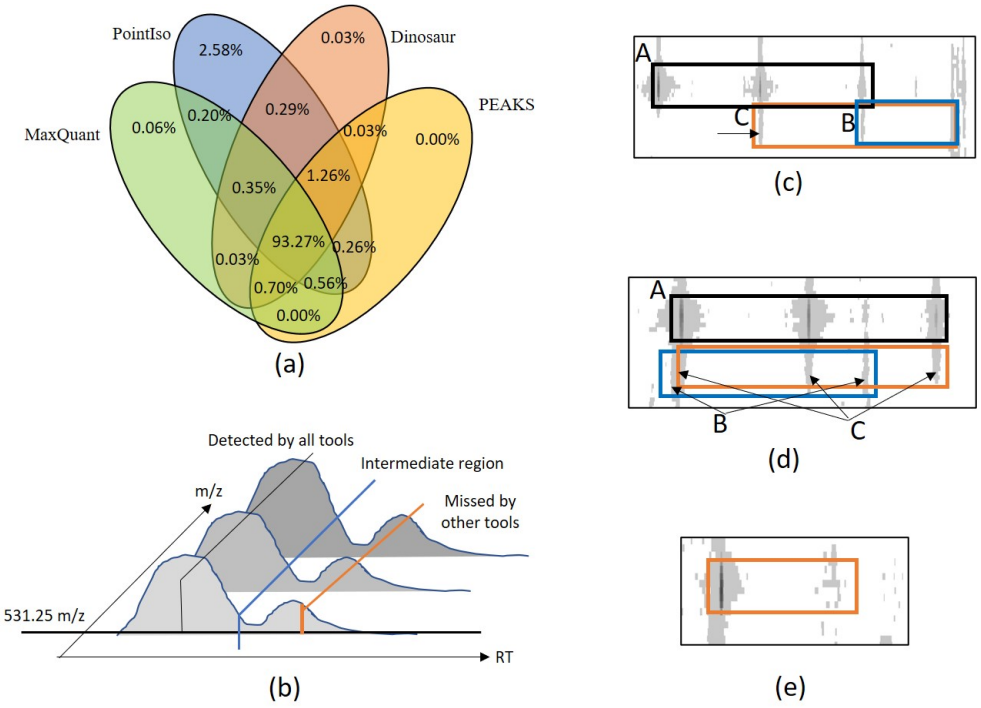


Figure 4.3: Observations on identified peptide features detected by only PointIso. (a) Venn diagram of identified peptide features detected by different algorithms from replicate 4 of sample 3. We show four algorithms to keep the Venn diagram simple. Here the the percentages are over MS/MS identified peptides of amount 10,000 approximately. The comparison with DeepIso is shown in Appendix D.2. (b) Most of the tools merge the low intensity feature marked with orange line with the bigger one marked with black line during pre-processing steps. (c) In this image, A and C are the actual features. Only PointIso detects these precisely. But other tools detect A, and instead of C, they report B by mistake because of missing the monoisotope (merging it with A). (d) Closely residing and overlapping features, like feature B in blue and C in orange rectangles are sometimes missed by other tools as well, although detected by PointIso. (e) Feature with broken signals are detected by our model, but discarded by other algorithms.

4.3 Discussion on the Design Strategy & Performance

We propose PointIso, a deep learning-based model that discovers the important characteristics of peptide features by proper training on a vast amount of available LC-MS data. Other heuristic method based models have to set different parameters, e.g., the number

Platform	Processor: Intel Core i7, 4 cores OS: Windows 10 for running the applications			Processor: Intel(R) Xeon(R) Gold 6134 CPU, NVIDIA Tesla OS: Ubuntu 16.04.5 LTS for running the python scripts		
Algorithms	PEAKS	Dinosaur	MaxQuant	**PointIso**	DeepIso	OpenMS
Running Time	8 minutes	15 minutes	30 minutes	**30 minutes**	1 hour and 40 minutes	2 hours and 50 minutes

Table 4.3: Approximated running time of different algorithms. Here the platform used for OpenMS, DeepIso and PointIso did not have support for running Windows application of PEAKS, MaxQuant, and Dinosaur. So we used different machine for running those.

of scans to be considered as a feature, centroiding parameters, theoretical formulas for grouping together the isotopes, and also data dependant parameters for noise removal and other preprocessing steps. Therefore, field experts have to go through extensive experiments to set those parameters. However, PointIso does not rely on manual input of these parameters anymore, which is the main strength of this model. In this section, we will first demonstrate the justification of different design strategies performed. Then we will refer to some potential research directions.

Model	Matching with MS/MS identified peptides
Initial model with 50% overlapping sliding window	65%
Bi-directional 2D RNN	72%
Dual attention mechanism	94%
Fine tuning of IsoDetecting module with long RT range	95.5 %
Increasing resolution from 0.01 m/z to 0.0001 m/z	97%
Fine tuning using features detected with wrong charge by IsoDetecting module	98.22%
New architecture of IsoGrouping module	99.55%
Fine tuning with feature like noises	98.52%

Table 4.4: Performance of PointIso in different developmental stages (based on validation dataset).

4.3.1 IsoDetecting Module Changes from Image Based Model To Point Cloud Based Model

We will first discuss why our point cloud based system is preferred over the image based algorithm, e.g., DeepIso. The reason is twofold. First, we want to change the classification network (that slides scanning window pixel by pixel and generate prediction for each pixel) to a segmentation network that can predict all the datapoints at a time, making the process quite faster. Second, we want to accept higher resolution with arbitrary precision. Now, a scanning window covers 15 RT and 2.0 m/z. If we consider a resolution of up to four decimal points, 2D image based segmentation network will need to segment 300,000

pixels ($15 \times \frac{2.0}{0.0001} = 300,000$), where most of the points will be blank. So, the 2D image representation of 3D features makes the segmentation problem unnecessarily voluminous, especially with higher resolution. However, with point cloud representation a scanning window has about 5000 points (each having 3 axes information) only. This number is chosen by observing the number of datapoints through out all the scanning windows and taking the 95^{th} percentile of those numbers. Therefore, to support higher resolution compatibility and a faster speed, we move from image based classification network to point cloud based segmentation network. There are also other literature, e.g., PointNovo [55], which switched to point cloud representation for supporting higher resolution data like us.

4.3.2 Weighted-Cross Entropy Loss for IsoDetecting Module

In PointIso, we have to deal with a highly class-imbalanced problem (in terms of $z = 0$ to 9) with this segmentation network of the IsoDetecting module. We use class weights (decided based on the class distribution per sample) while calculating cross-entropy loss so that both the positive ($z = 1$ to 9) and negative ($z = 0$) classes are learned well. More discussion and empirical results regarding this are included in Appendix A.3. For scanning the LC-MS map, we used sliding window with 50% overlapping. This initial model was able to achieve about 65% matching with the peptide identifications as shown in the first row of Table 4.4. Therefore, we had to discover more effective approach to improve the sensitivity, which are discussed next.

4.3.3 Attention Mechanism in IsoDetecting Module

We would like to explain the reason for using the attention mechanism with the segmentation network of the IsoDetecting module. In a random scenario, features can spread over multiple windows. Applying a segmentation network without any surrounding knowledge will cause missing of the features as illustrated in Figure 4.4(a). Here we see two successive and non-overlapping scanning windows, W1 and W2, and six features: A, B, C, D, E, and F are shown at the top. Then the combined output of the two successive scanning windows is presented at the bottom. Feature A and F are fully contained within W1. Therefore, all of its isotopic signals are correctly detected, as shown in the combined output. However, for each of the other four features (B, C, D, and E), W1 sees partial traces shown by small circles in the upper image. Without any background knowledge, those traces are not adequate for deciding whether they belong to real features or merely noisy traces. The isotopes which are not detected are marked by cross signs. The second window, W2, detects

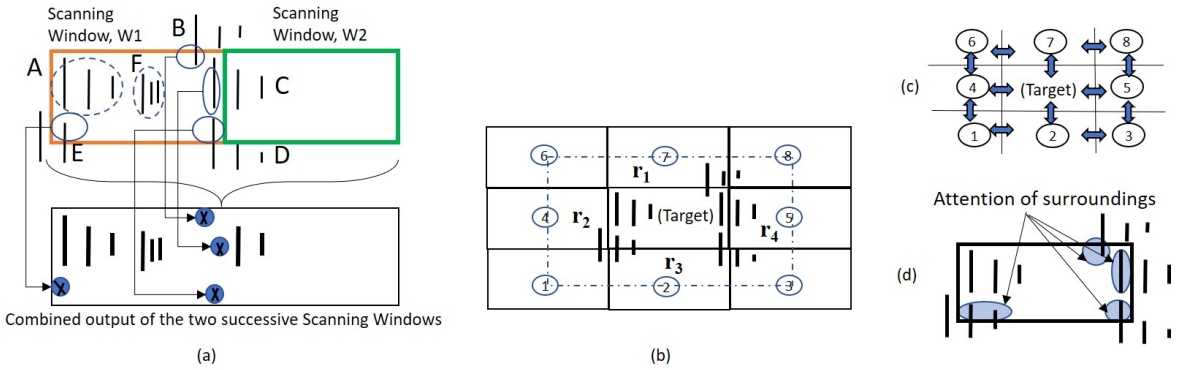


Figure 4.4: Need for attention mechanism for improving the sensitivity with *non-overlapping* sliding window. (a) Two non-overlapping sliding windows are shown in the top, and the corresponding output is shown in the bottom. It is applying a segmentation network without any surrounding knowledge, therefore, it causes missing of the feature isotopes, as shown by cross marks. (b) Surrounding regions of a target window. Among the eight regions, only four regions seem important according to our experiments: r_1, r_2, r_3, r_4. (c) 2D bi-directional RNN to flow the surrounding information towards the target window in center. (d) Attention coming from surrounding regions over the marked datapoints of the target window. There are partially seen features in those marked regions, and for segmenting those data points, IsoDetecting needs to consider the influence or attention coming from the surrounding regions.

two isotopes of feature C. But the system missed the monoisotope of feature C, which is treated as missing the feature as a whole. Because when peptide features are matched with the identified peptides, the matching is performed in terms of monoisotope, not the other isotopes. Besides that, it also fails to compute the total abundance of features properly, since its missing trailing regions of features like B, D, and E. To overcome this problem, we have to incorporate surrounding knowledge while segmenting the datapoints of a target window, i.e., W1 in Figure 4.4(a). According to our experiments, we find that the regions r_1, r_2, r_3, and r_4 in Figure 4.4(b) are actually playing the key role in detecting the traces inside target window. Just using a big window and predicting the smaller center region points does not solve the problem according to our experimental result. The results with other different criteria: 50% overlapping of scanning window, 2D bi-directional RNN (Figure 4.4(c)), and the attention mechanism (Figure 4.4(d)) inspired by DANet are presented in Appendix A.4. Besides that, we also had a visual verification of whether the partially seen peptide features are properly detected or not as presented in Figure 4.5(a) and (b).

Since the PointNet segmentation network combined with DANet works better than other techniques, we choose this strategy to develop our IsoDetecting module.

4.3.4 Upgrading IsoGrouping Module

Next, we discuss how the IsoGrouping module supports the higher resolution output coming from the IsoDetecting module as follows:

- The IsoDetecting module outputs the isotope list with m/z resolution of up to 4 decimal places, but IsoGrouping module does not need that much high resolution to group together the potential isotopes into a feature. Therefore, we use a resolution degrading approach before passing the isotope lists to IsoGrouping module (without resolution degradation, its frame width will become very wide, which would increase the network size without bringing any additional benefit). The isotopes who merge in the lower resolution are kept in separate lists and thus passed to the IsoGrouping module separately. This resolves the problem of missing features merged in lower resolution.

- Besides that, in the Isogrouping module of DeepIso, each frame of the input sequence covers a wide range of m/z value. As a result, each frame holds an isotopic signal along with its background. We filter out the signal area from background while passing the frames (unlike DeepIso), based on the boundary information provided by IsoDetecting module, so that IsoGrouping module can see the beta distributed (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) isotopic signals better than before. As a result, we can keep the module simple by avoiding attention gate (which was required in DeepIso).

- Unlike DeepIso, which uses the RNN layer to process frames of the sequence one at a time, we process five frames at a time using a network consisting of CNN and fully connected layers through weight sharing. It results in better prediction at the output layer.

- Other new additions are: we incorporate area under the isotopic signal as context information through embedding (which reduces the uncertainty during class prediction) and feed the charge into the network through a scaling gate neuron, which helps in a proper grouping.

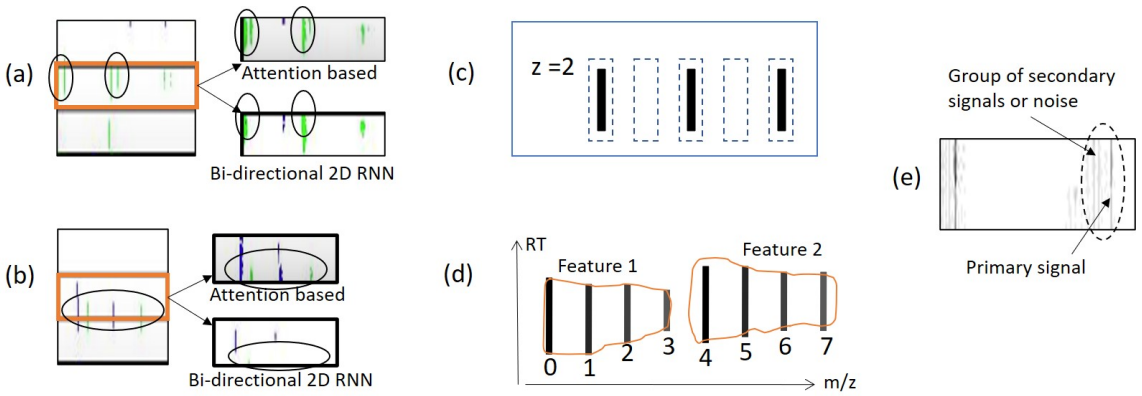


Figure 4.5: Detection performance comparison between two candidate solutions and illustration of samples used for fine-tuning. Illustrations in (a) and (b) show the comparison between attention-based mechanism and bi-directional 2D RNN. The orange rectangle is showing the target window. For each target window, detections by attention mechanism and bi-directional two-dimensional RNN are shown next to it, pointed by arrow signs. In the target window, we see some vertical traces in each of the circle markers. In (a), we see that those separate traces are detected separately by attention mechanism, but merged by bi-directional 2D RNN. In (b), only blue features are detected by bi-directional 2D RNN (which is wrong), but both blue and green features (partially seen in the target window) are detected by attention mechanism. (c) When isotope lists are passed to the IsoGrouping module with wrong frames (dotted rectangles) because of the wrong charge ($z = 4$) detected by the IsoDetecting step, it results in discarding this whole group of frames as noise due to the inconsistency (blank frames) observed. (d)Two features having same charge z are adjacent (not overlapping) such that the distance between last isotope of feature 1 and first isotope of feature 2 is equal to $\frac{1}{z}$, i.e., the inter isotope distance of the features. We name such cases as 'adjacent feature' problem. (e) Feature like noisy signals just beside the actual isotopic signals.

The final architecture of IsoGrouping module is quite different than the one in DeepIso and improves the feature detection by about 1.3%, as presented in the seventh row of Table 4.4. Details are presented later in Section 4.4.2.

4.3.5 Fine Tuning

Fine tuning the primary model by feeding back the misclassified data acted as a reinforcement learning and leads to overall improvement. Please note that it is different from the backpropagation of errors. Fine-tuning in deep learning involves collecting samples (which may not have been provided during initial training) for which model makes mistakes and then retraining the model with those specific cases for a few epochs. This improves the model's ability to give the correct result next time. Some examples include, features detected with wrong charge (Figure 4.5(c)), adjacent features (Figure 4.5(d)), secondary signals (Figure 4.5(e)) etc., which are fed back to the model for further learning. Obviously, we have to keep a validation set during this retraining or fine-tuning approach so that the model does not overfit those specific cases. We avoid discussing the details in this section for brevity and elaborate the technique later in Section 4.4.3.

4.3.6 Impact of Secondary Signals on Total Number of Features

Secondary signals as shown in Figure 4.5(e), and defined in Section 1.1.1, are caused by the instrument, and we can not always avoid having them. These signals cause multiple reports to the same feature and results in about 130,000 peptide features from each LC-MS map, by our model (even after the improvement by fine tuning as explained in Section 4.4.3). We merge some redundant reports by comparing the similarity of final output layer (as described in Section B.4) and it gives us about 100,000 features. This is still much higher than other tools, e.g., PEAKS, MaxQuant, Dinosaur, OpenMS report 41,000, 42,000, 45,000, and 60,000 respectively. Please note that, later in Section 5.0.1, we verify that our 100,000 peptide features do not result from a high amount of false positives. Rather, many of them are redundant reports of the same feature due to secondary signals close to the primary signals. If we use some tighter threshold at the final output layer of PointIso model, we can reduce the total number of peptide features to about 48,000 only. However, this also reduces the model sensitivity for potato peptides: from 97.25% to 96.77%, and for human peptides: 98.32% to 96.26%. Although this sensitivity is still 3-4% higher than other tools (based on Table 4.2), but it can be an important future direction of work to reduce the redundant reports without the reduction in sensitivity.

4.4 Architectural Details and Methods for Reproducing PointIso

We train our model and evaluate it through K-fold cross-validation. We divide the LC-MS maps in the dataset into K groups or K folds. The experiment is repeated K times, where one group is kept for testing, and all other groups $(K-1)$ are used for training. We tried two different settings. In one setting we set $K = 2$, and in another setting $K = 6$. In setting $k = 2$, we have 33% data for training and the rest for testing. In $k = 6$, we have over 80% data for training and about 16% data for testing. We evaluate our model for two different scenarios to see how they influence model performance. We discuss how the LC-MS maps are divided into different folds for cross-validation in Appendix A.2. To save the best model state during training, we keep one LC-MS map from the training set as a validation set. Then we use that model state for testing. Whenever we say train or validation on an LC-MS map, we mean training/validation on the features cut from that LC-MS map (explained later under the subsections regarding training data generation for IsoDetecting and IsoGrouping module). However, when we say testing on an LC-MS map, we actually mean scanning the full LC-MS map as shown in the block diagram of Figure 4.1. That is, during testing (or the real application phase) we will scan the whole LC-MS map in a bottom-up, left to right fashion (in other words, column by column).

Our model runs the processing on a raw LC-MS map which is obtained in .ms1 format (profile mode, no centroiding) using the ProteoWizerd 3.0.18171 [8]. Then we read the file and convert it to point cloud based hash table, where RT scans are used as keys and (m/z, intensity) are inserted in a sorted order under those keys. Therefore we have the datapoints saved as triplets (RT, m/z, intensity) in the hash table. In the following sections, we will discuss IsoDetecting and IsoGrouping modules' scanning procedure and technical details on model training, including training data generation.

4.4.1 Step 1: Scanning of LC-MS map by IsoDetecting module to detect isotopes

Our network scans the 3D LC-MS plot using a non-overlapping sliding window having dimension 2.0 m/z, 15 RT scan, and covering full intensity range, as already presented in Figure 4.1. The intensities are real numbers scaled between 0 to 255. Here the objects, i.e., peptide features are to be separated from the background. The background may contain feature-like noisy signals and peptide features are frequently overlapped with each other. So the target is to label each datapoint represented by a triplet (m/z, RT,intensity) with

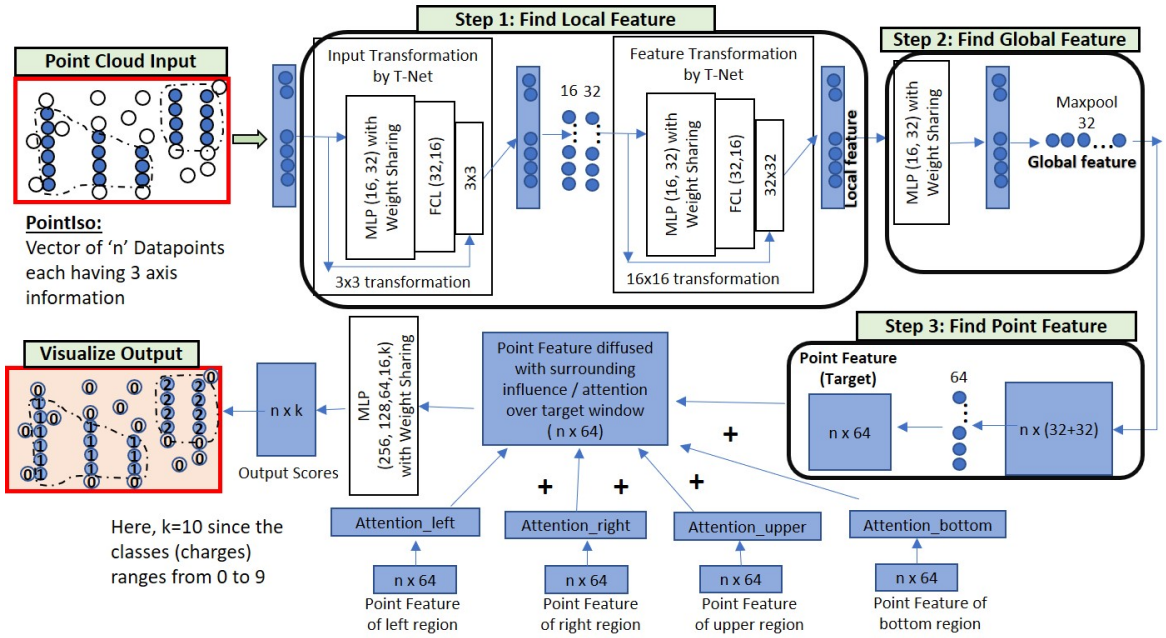


Figure 4.6: The network of IsoDetecting module. This network goes through three steps, finding the local features (please see the original PointNet paper for T-Net and other technical details), global features, and point features respectively of the given target window. The number of layers and neurons in the Multiple Parceptron Layers (MLP) and Fully Connected Layers (FCL) is determined by experiments and mentioned in the figure. Point features of the target window are then diffused with features of surrounding regions based on their attention or influence over the target window (calculation of $Attention\_left$ and others are shown in the next figure). Finally, the diffused features are passed through four Multi-Layer Perceptron (MLP), and the Softmax layer at the output provides the final segmentation result.

its class. The class is either charge $z = 1$ to 9 (positive) if the datapoint belongs to a feature having that charge, or $z = 0$ if that datapoint comes from background or noise. Each window sees a point cloud which is essentially a set of points, or triplets (RT, m/z, intensity). This is passed through a PointNet architecture as shown in Figure 4.6. In order to properly segment peptide features spreading over multiple sliding windows, we adapt the DANet [20] and plug into our model to find the attention or influence of four surrounding regions (Figure 4.4(b)) over the target window datapoints. We present a flowchart in Figure 4.7, showing the calculation of attention coming from the surrounding regions. The detailed explanation of this flowchart is provided in Appendix B.2.

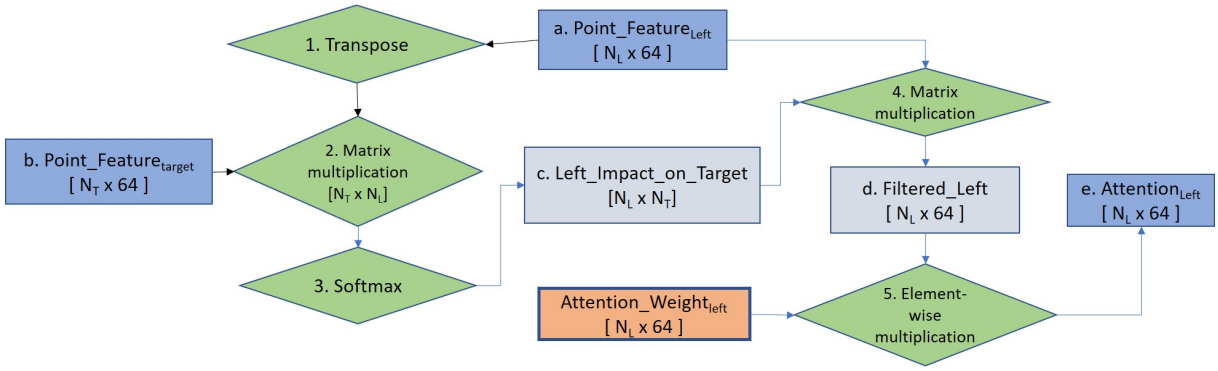


Figure 4.7: Flowchart of attention calculation in the IsoDetecting module. Here, 'T' and 'L' means target window and left window respectively. This particular flowchart is intended to find out the attention or impact of the left region over the datapoints of the target window. Exactly similar approach is followed for other surrounding regions as well and finally, all are diffused with the $Point\_Feature_{target}$ by addition.

We can divide the LC-MS map along m/z axis into sections of equal ranges (e.g., 400-599 m/z, 600-799 m/z, 800-999 m/z, ...) and process multiple sections in parallel to make the process time-efficient. We keep nine hash tables for recording the detection coordinates (RT, m/z) of features from nine classes ($z = 1$ to 9) during the scanning. The m/z values of the isotopes are used as the key of these hash tables, and the RT ranges of the isotopes in a feature are inserted as values under these keys as shown in the block diagram of Figure 4.1. Since the detection of wider isotopes may span over a range of points along m/z axis, we take their weighted average to select specific m/z of an isotope.

Class (charge state)	1	2	3	4	5	6	7	8	9
Peptide Features	163,038	863,050	428,909	29,183	1,503	653	179	236	233

Table 4.5: Class distribution of peptide features in our dataset consisting of 57 LC-MS maps.

Training data generation for IsoDetecting module

IsoDetecting module is supposed to learn some basic properties of peptide features, e.g., beta-distribution shaped signal, equidistant isotopes, etc. (as discussed in Section 1.1.1),

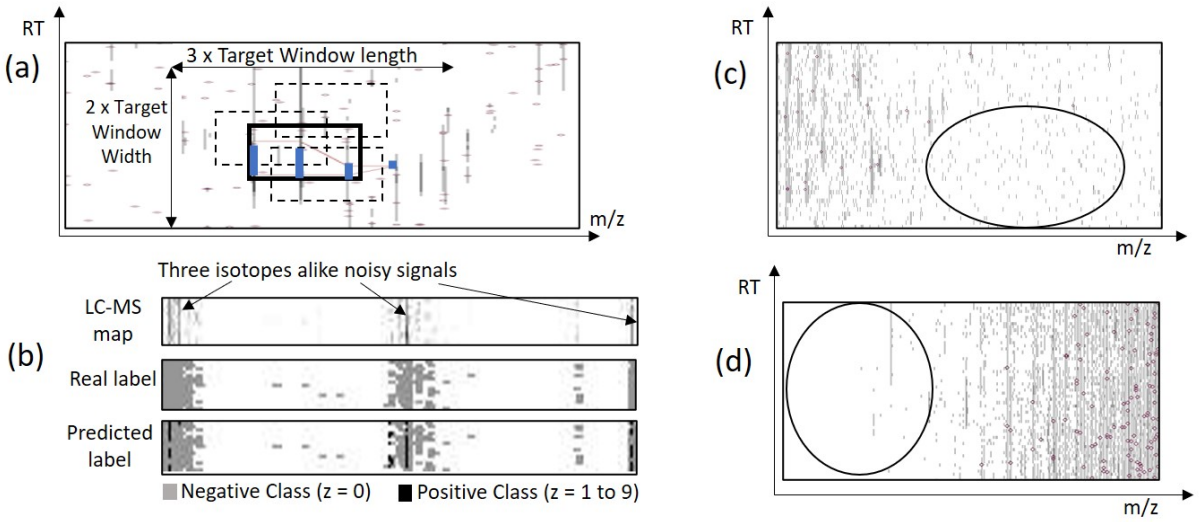


Figure 4.8: (a) An area in LC-MS map is shown. A target window in bold black rectangle containing a feature in blue color is shown. This target window and its four surrounding regions (as shown in Figure 4.4(b)) forms a positive training sample for IsoDetecting module. We also slide the target window within the region showed by arrow signs to generate training samples holding the peptide feature in different locations of the target widow (some are shown by dotted rectangles). (b) One negative training sample containing feature like noises is shown the topmost rectangle. In this region of LC-MS map the traces look like a feature having three isotopes. However, those are actually noisy signals as shown in the middle rectangle. And if we do not provide such training samples, then PointIso label the respective datapoints as positive class and report it as a feature, as shown in bottom rectangle. That is why we should provide such samples during training. (c) Some region in LC-MS map containing only arbitrary noises is selected for generating negative training samples. (d) Some region in LC-MS map containing blank area is also selected for generating negative training samples.

and many other hidden characteristics. From each LC-MS map, we cut out the peptide features for training, which we call *training samples*. Each training sample consists of a target window (resembling the sliding/scanning window in the testing or application phase) and four surrounding regions. The datapoints in the target window are labeled with a value within the range z = 0 to 9. Here, z = 0 means the respective datapoint belongs to noise or background, and z = 1 to 9 means the respective datapoint belongs to a feature having charge z. We slide the target window over the peptide feature and

its surrounding region to mimic the testing scenario so that we can generate training samples holding peptide features in different locations of the target window, as shown in Figure 4.8(a). We select the common list of peptide features provided by MaxQuant and Dinosaur to generate the positive training samples. The total number of such common peptide features available from each charge state is presented in Table 4.5. Next, we see the LC-MS maps in PEAKS Studio, and we visually choose some regions containing noisy signals that look like features (Figure 4.8(b)) (in our case, it was around the retention time range of 10 minutes), only noises of different intensities (Figure 4.8(c)), and completely blank areas as well (Figure 4.8(d)). We cut out some training samples from those regions (one target window and four surrounding regions as before) and call them negative training samples. Please note that all the datapoints in negative samples are labeled '0'. We see the total number of training samples in Table 4.6. Readers may wonder why we have included a column for 'Total Datapoints' in the table. Please note that, unlike DeepIso, the IsoDetecting module is a segmentation network here. As a result, it has to classify each datapoint in the target window of training samples. Suppose a target window in a positive training sample prepared from peptide feature having charge '2' contains features with other charges, e.g., '3' and '4'. In that case, the IsoDetecting module has to classify the datapoints belonging to not only charge '2', but also charge '3' and '4'. Therefore, the class sensitivity is impacted by the total amount of datapoints having that class, taking into account *all* the training samples. So we have included a column labeled as 'Total Datapoints' in Table 4.6. The positive and negative samples are generated from the validation LC-MS map similarly and presented in the table as well.

IsoDetecting module training and validation

We use a minibatch size of 8 during the training which takes about 15 GB GPU Memory due to the sophisticated architecture. We could not increase the batch size due to GPU Memory limitation. We use 'NAdam' stochastic optimization [32] with initial learning rate of 0.001. We check the cross-entropy loss and sensitivity on validation set after training on every 1200 training samples per epoch. If we do not observe any significant reduction in the validation loss for about 5 epochs, we decrease the learning rate by half. The model converges within about 100 epochs. We use Sparse Softmax cross-entropy as the error function at the output layer. Besides that, we use the weighted cross-entropy as already mentioned in Section 4.3.2. The average sensitivity of the trained model on the training samples and validation samples are provided in Table 4.7 for a particular fold.

Class (z)	Training		Validation	
	Total datapoints	Total Samples	Total datapoints	Total Samples
0	544,014,927	40,502	22,100,416	10,138
1	3,648,512	37,924	167,671	1,282
2	28,633,707	152,148	1,731,244	8,902
3	19,021,011	91,542	1,033,477	4,485
4	2,998,905	25,526	77,534	281
5	3,526,031	22,134	3,032	13
6	431,139	7,721	606	5
7	30,145	4,472	319	4
8	18,144	8,706	48	3
9	17,310	3,429	227	2

Table 4.6: Amount of samples for training and validation of IsoDetecting module in PointIso model. Because of inadequate training data for features with charge states 5 to 9 as mentioned in Table 4.5, we had to apply data oversampling and augmentation to increase training samples from these classes. The amount of samples from class 0 depends on our choice. We chose the amount so that the total number of datapoints from this class is higher than others because the LC-MS map is very sparse. The validation set does not contain any duplicated data, and there is no overlapping between the validation dataset and the training dataset.

4.4.2 Step 2: Scanning of LC-MS map by IsoGrouping module to report peptide feature

There are four significant differences between IsoDetecting and IsoGrouping modules. First, the IsoDetecting module scans the LC-MS map along the RT and m/z axis, whereas the IsoGrouping module scans left to right, i.e., only along the m/z axis. Second, the IsoDetecting network is a point cloud based network, whereas the IsoGrouping network is image-based. Third, the IsoGrouping module performs a sequence classification task that generates one output after seeing through 5 consecutive frames, unlike the IsoDetecting module, which segments the datapoints of the input window. Last, the IsoDetecting module accepts very high resolution m/z values (up to 4 decimal places), but the IsoGrouping works on comparatively lower resolution m/z values (up to 2 decimal places to keep the model simple), because it does not need much higher resolution to group the isotopes into features.

The IsoDetecting module provides us a list of isotopes. Equidistant isotopes having

Class (z)	Training		Validation	
	Sensitivity-Average (%)	Sensitivity-Best (%)	Sensitivity-Average (%)	Sensitivity-Best (%)
0	88.48	30.0	88.48	38.69
1	57.59	91.0	61.54	99.28
2	76.90	97.0	82.98	98.40
3	75.08	94.75	79.42	96.05
4	64.78	94.24	52.80	97.33
5	88.57	94.21	58.74	100
6	60.73	82.5	15.68	70.56
7	40.12	60	10.03	70.78
8	4.07	10	3.0	10
9	4.50	8	2.01	15

Table 4.7: Class sensitivity of the IsoDetecting module for fold 1 in the 2-fold cross validation experiment. That is, dilution sample 10, 11, 12 are used for training, sample 9 is used for validation, and the rest are used for testing. We show two cases, average and best case in terms of feature detection difficulty. The best case occurs when the feature is left aligned with the target window boundary, e.g., feature 'A' in Figure 4.4(a). The average case means the target window may have the features at any location of the window, may contain any number of features and the features may be partially or fully seen and may be overlapping as well. Due to the lack of variance in training data for charge states 6 to 9, the model's validation sensitivity does not go up high for these classes. However, since most of the peptide features appear with charge states < 6, lower sensitivity for them does not impact the overall performance. The validation set does not contain any duplicated/over-sampled data and there is no overlapping between validation samples and training samples.

the same charge are grouped into a cluster or sequence. Then those sequences of isotopes are passed to the IsoGrouping module. Unlike DeepIso, we do not pass the isotopes directly to the second module, because here the input resolution is different in two modules. Therefore we apply a resolution degradation technique that filters out the region of isotope (as suggested by IsoDetecting module) from the background, present it in lower resolution, and then pass it to the IsoGrouping module. The detailed procedure is provided in Appendix B.3.

We illustrate our proposed network for the IsoGrouping module in Figure 4.9. A sequence of five frames is shown in top left of the figure (first three frames are holding isotopes and the rest two are blank). It may break each sequence into multiple features, or report one feature consisting of the whole sequence of isotopes, or even report that sequence as mere noise. It works on a sequence of isotopes in multiple rounds. Each round process five

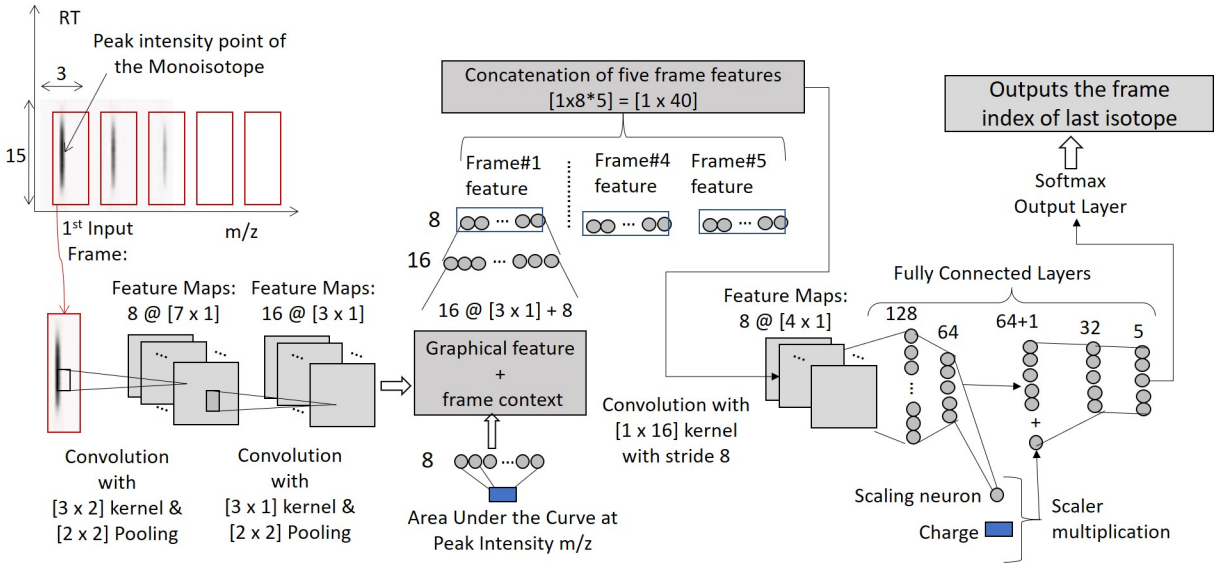


Figure 4.9: The network of IsoGrouping module. It starts with two convolution layers to fetch the graphical features from the input frame. Then we concatenate the intensity of the isotopic signal (area under the beta distributed (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) isotopic signal) with it through an embedding layer of neurons (frame context). Then this is passed through two fully connected layers having sizes 16 and 8. This gives us the 'frame feature' of the input frame. We perform the same for five consecutive frames and then concatenate the 'frame feature' of those altogether. Then one layer of convolution is applied to detect the combined feature from all the frames. The resultant features are passed through two fully connected layers (size 128 and 64) to decide whether this is a noise or potential feature. This probability is also used to activate a scaling neuron, that feeds the charge into the network through proper scaling. The scaled charge is concatenated with the latest layer output (size 64) and passed through two fully connected layers. Finally, the Softmax output layer at the end classifies the sequence. We include pooling layers after the first and second convolution layers. We apply the ReLu activation function for the neurons. The dropout layers are included after each fully connected layer with a dropout probability of 0.5. The other network parameters are mentioned in the figure.

consecutive frames at a time. The output is $i = 0$ to 4, where, $i = 0$ means that no feature starts at the first frame, so skip it. Output $i = 1$ to 4 means, there is a feature starting in the first frame, and it ends at $(i + 1)^{th}$ frame. If output $i = 4$, it means that the feature has at least 5 isotopes. We denote the classes $i = 0$ to 4 as A, B, C, D, and E. That means,

class A associates with noisy traces that do not form any feature. Class B, C, D, and E correspond to features with 2, 3, 4, and 5 isotopes, respectively. We have not kept any 1 isotope category because usually, a peptide feature has at least two isotopes. Since the scanning window slides left to right, it can handle the cases when peptide features have isotopes over five. Although the architecture of IsoGrouping module in DeepIso and PointIso is quite different, the workflow is kept same for the sake of user convenience. Therefore, we refer to see Section 3.3.2 for a step by step explanation of the scanning procedure by IsoGrouping.

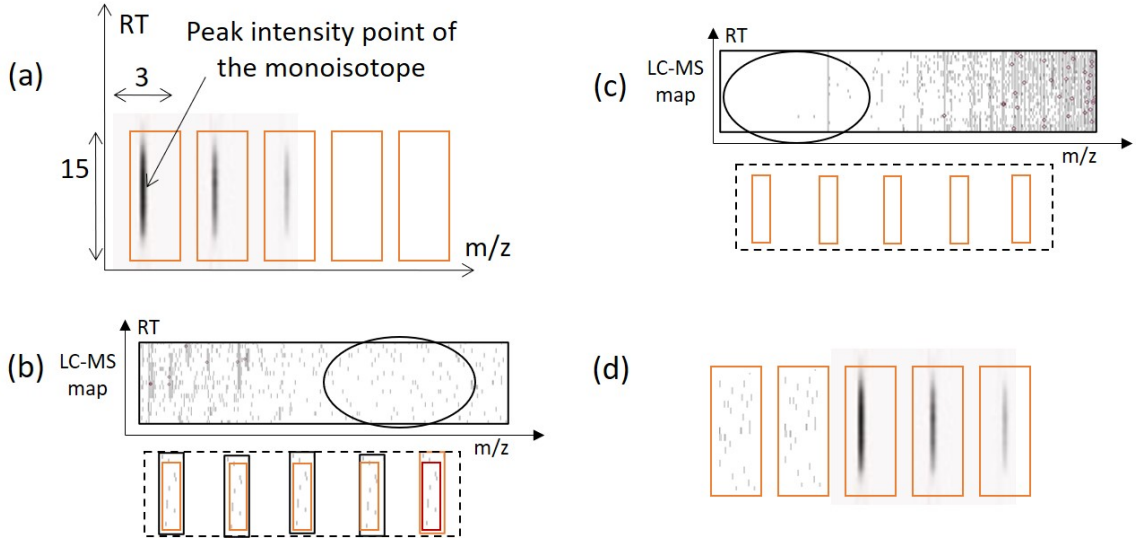


Figure 4.10: (a) A sequence of five frames, first three holding three isotopes. Each frame has dimension $[15x3]$, covering 15 scans along the RT axis and 0.03 m/z along the m/z axis (each pixel represents 0.01 m/z). We filter out isotopic signals from the background by taking the intensity within the range of 2 ppm before and after the peak intensity m/z value, and 7 scans before and after the peak intensity RT value. The signal is left aligned with the frame. (b) A sequence of five frames which is generated from noisy areas in LC-MS map. (c) A sequence of five frames which is generated from blank areas in LC-MS map. (d) A sequence of five frames where the initial frames are holding noisy traces.

Training data generation for IsoGrouping module

Usually, monoisotope's intensity is the highest among the other isotopes in a feature and dominates the total intensity of the feature. This property should be learned by the

IsoGrouping module. We prepare positive training samples by generating a sequence of 5 frames for each peptide feature in the LC-MS maps, where the sequence starts at the first isotope of the respective feature, as shown in Figure 4.10(a). Each sequence is labeled by the frame index holding the last isotope of the feature (indexing starts from 0). So the sample in Figure 4.10(a) has label 2. The minimum number of isotopes in a feature is 2, i.e., the minimum label of a positive sample can be 1. If the feature has equal to or more than 5 isotopes, the label is 4. We generate negative samples by cutting some sequences from the noisy (Figure 4.10(b)) and blank area (Figure 4.10(c)) and labeling them as 0. We generate sequences that contain peptide feature, but the feature does not start at the first frame of the sequence (Figure 4.10(d)). Those samples are labeled as 0 as well, indicating that no feature starts at the first frame. We do this to handle the cases where noisy traces are classified as isotopes by the IsoDetecting module by mistake and thus grouped with the isotopes of actual features in the intermediate step.

IsoGrouping module training and validation

For the training, we set minibatch size 128 and apply 'NAdam' stochastic optimization [32] with initial learning rate of 0.001. We run the training for 10 epochs, with validation step run in every 1200 sample training. We use Softmax cross-entropy as an error function at the output layer. We see the training and validation sensitivity in Table 4.8. The sensitivity of the IsoGrouping module is defined as the percentage of features reported with the correct number of isotopes. We already mentioned that we categorize the training samples into five classes denoted A, B, C, D, and E.

Class	Sensitivity on Training Set (%)	Sensitivity on Validation Set (%)
i=0, or A (noise)	89.42	90.78
i=1, or B (2 isotopes)	57.93	57.50
i=2, or C (3 isotopes)	51.98	43.30
i=3, or D (4 isotopes)	61.90	59.86
i=4, or E (5 isotopes or more)	61.77	64.14

Table 4.8: Class sensitivity of the IsoGrouping module on the training set and validation set. The output is $i = 0$ to 4, where $i = 0$ means that no feature starts in the first frame, so skip it. Output $i = 1$ to 4 means, there is a feature starting in the first frame, and it ends at $(i + 1)^{th}$ frame. When output $i = 4$, it means there might be more isotopes left. So we run another round of processing over the rest of the isotopes of the same cluster or sequence. Therefore, although our network process five frames at a time, if the feature has more than five isotopes, those can be found by overlapping rounds.

We observe that the maximum sensitivity of the classes > 0 (i.e., B, C, D, E) is at most 61% on the training set, and 64% on the validation set. To have a better observation we present the confusion matrix in Table 4.9. The column A presents the percentages when monoisotope in a feature is missed. We see the model hardly misses the monoisotopes but confuses about the last isotope of a peptide feature. Please note that reporting the monoisotope along with the first few isotopes (having higher intensity peaks) of a feature is more important in the workflow (based on the discussion with collaborators from Bioinformatics Solution Inc.). Because they dominate the feature intensity and are used in the next steps of protein quantification and identification. But rest of the low-intensity isotopic peaks in a feature do not contribute much to quantification or identification. Therefore we accept a feature if the monoisotope along with high-intensity isotopes are reported correctly.

Class	A	B	C	D	E
A	89.43%	6.88%	1.72%	0.94%	1.03%
B	17.29%	57.93%	18.18%	5.58%	1.03%
C	5.38%	18.94%	51.98%	21.58%	2.13%
D	3.25%	5.44%	14.29%	61.90%	15.12%
E	5.71%	2.55%	3.18%	26.79%	61.77%

Table 4.9: Confusion matrix produced by IsoGrouping module on validation dataset. The diagonal values, e.g. [C, C] represent the sensitivity for class C. We say a feature is misclassified as class A when the monoisotope (first isotope) or all of the isotopes are missed, i.e., the feature is thought to be noise by mistake. The value of [C, A] indicates what percentage of features with three isotopes are either misclassified as noise, or monoisotope is missed. [C, B] indicates the percentage of features which actually have three isotopes but the third one is missed, and only first two are combined together. Similarly [C, D] shows for what percentage of three isotope features, IsoGrouping module finds ONE additional isotope at the end.

While training the IsoGrouping module, to save a network state we use a validation LC-MS map. We keep track of the model sensitivity of five classes and the average cross-entropy loss for detecting convergence and avoid overfitting. After the convergence is reached, the state that gives highest detection of identified peptide features in the validation LC-MS map is saved for testing. And then we perform the testing using the saved IsoDetecting state (as described in the previous section) and saved IsoGrouping state.

Finally, we would like to mention some common strategies followed for implementing and training both of the modules. We implemented our deep learning model in Python [75,

28] using the Google developed Tensorflow [1] library. We check the accuracy (cross-entropy loss and average sensitivity of the classes) on validation set after training on every 1200 samples per epoch. We perform data shuffling after each epoch which helps to achieve convergence faster. We stop training if no progress is seen on validation set for about 15 epochs. In the initial phase of project development, we used batch normalization layers in PointIso. It made the model converge quickly during training but does not improve the class sensitivity. Moreover, adding the batch normalization layer in IsoDetecting module took more GPU memory. That is why we discard it later to make the testing or application phase more memory efficient. For developing the PointIso we use Intel(R) Xeon(R) Gold 6134 CPU, NVIDIA Tesla GPU, and Ubuntu 16.04.5 LTS operating system.

4.4.3 Fine Tuning Using Misclassified Features

In this section we will discuss the approach of fine tuning. Fine tuning the primary model by feeding back the misclassified data played an essential role in overall improvement. After we train and validate IsoDetecting and Isogrouping module, we run a full scanning on the same LC-MS maps used for training (e.g., in 3D dataset, LC-MS maps of dilution sample 9 to 12 for the first fold). That means, we scan the full LC-MS map by IsoDetecting module in a column by column order, which gives a list of sequence of equidistant isotopes. Then those are passed to IsoGrouping module for generating the final list of peptide features. Then this peptide feature list is matched against the MS/MS identified peptide features for the respective LC-MS map by MASCOT. Based on the matching result, we select the misclassified peptide features. By the term misclassified we mean four particular cases: peptide features not detected due to very long retention time range, peptide features detected with wrong charge, multiple peptide features grouped into one feature, and false positives due to secondary signals. Then we generate training samples from those misclassified cases as mentioned in the training data generation subsections of Method section in the main text and retrain both modules using those. We call this approach fine tuning by misclassified peptide features. After that we run the testing on test set. How do we detect those four types of misclassified features, extract those from LC-MS map, and label them for fine tuning PointIso, are explained below in more detail. While providing examples, we will be using LC-MS maps from fold 1.

- First, some identified peptide features were not detected and when we inspected those visually in the LC-MS map by PEAKS Studio, we discovered that those have comparatively longer retention time range, about 0.30 minute or longer. Please note that, those features were provided during initial training. But those cases are not

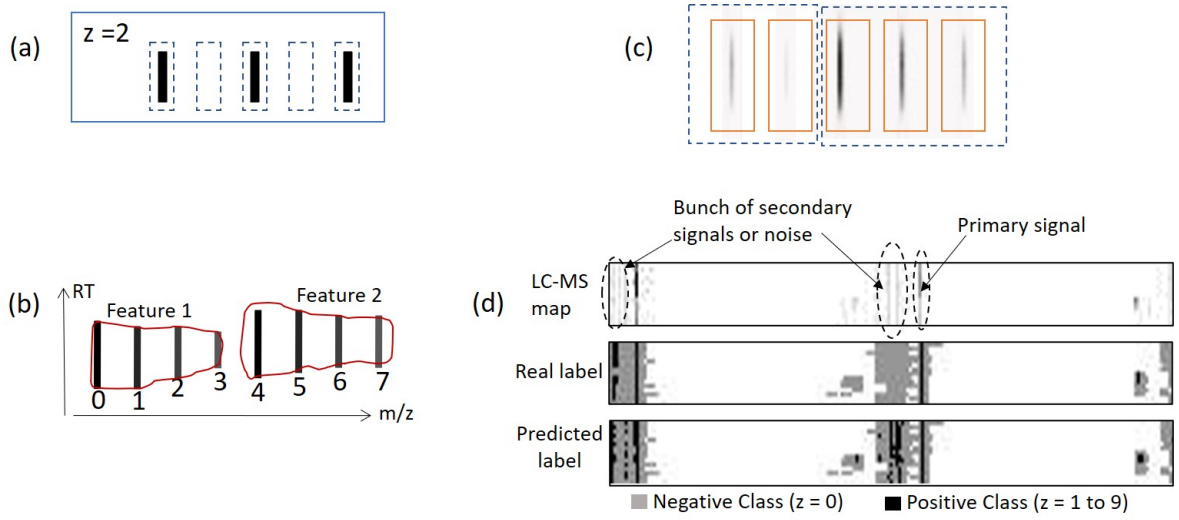


Figure 4.11: (a) A peptide feature with three isotopes and charge 2 is shown. But if this is passed to the IsoGrouping module with wrong frames (dotted rectangles) because of the wrong charge ($z = 4$) predicted by the IsoDetecting step, it results in discarding this whole group of frames (by predicting it as noise) due to the inconsistency (blank frames) observed. (b) We see the adjacent feature problem. (c) We see one training sample prepared for IsoGrouping module to solve the adjacent feature problem. (d) In the topmost rectangle we see isotopes of a peptide feature in black traces and some secondary signals as well. The middle rectangle is showing the true labeling of the datapoints using two colors: grey means negative class and black means positive class. The primary signals are detected by PointIso as shown in the bottom rectangle. However, the seconadary signals are also predicted as positive class by PointIso. But this is wrong. Datapoints in those regions should be predicted as negative class by IsoDetecting module. So we select such peptide features for fine tuning.

learned well. So we provide such samples twice in the dataset (i.e., given more weight or emphasized by duplication) and retrain the model. It improves the detection rate by about 2% (fourth row in Table 3).

- Second, we compare the clusters or sequences of isotopes returned by IsoDetecting module with the MS/MS identification result for the respective LC-MS map and select those sequences which are matched with the identification result in terms of m/z range and retention time range but DOES NOT match in terms of charge and are also rejected later by IsoGrouping module. So such features are selected and

fed back for further learning by IsoDetecting module. These sequences present the features which are detected by IsoDetecting module but with wrong charge. This usually happens when the feature appears in middle region of the target window (like 'F' in Figure 4(a) of the main manuscript). As a result, the wrong set of frames are passed to the IsoGrouping module, as shown in Figure S4.11(a). This causes complete rejection of the feature by the IsoGrouping module, and we miss a feature although it was detected in the first module. Please note that, it is possible that these features went through the usual training before. However, since these selected ones are not learned well, so we are just asking the IsoDetecting module to learn these again with greater emphasis (ensured by duplicating these samples during retraining as done in previous point). Retraining the IsoDetecting module using such samples improves the detection rate by about 1.5% (sixth row in Table 3 of main manuscript).

- Third, adjacent features as shown in Figure S4.11 (b) were not correctly separated by IsoGrouping module. Here, both features have the same charge, same or very close RT value at the peak intensity point, and the distance between the two features along m/z axis is equal to the distance between their own isotopes. We select such features by comparing the sequences of isotopes returned by IsoDetecting module with the MS/MS identified features for the respective LC-MS map and finding out those sequences where the $2^{nd}/3^{rd}/4^{th}/5^{th}$ frame matches with the identification result in terms of m/z, RT, and charge, but not the 1^{st} frame. Because it means that the monoisotope appears on the $2^{nd}/3^{rd}/4^{th}/5^{th}$ frame and the isotopes preceding it are coming from some other feature that is preceding and adjacent to this matched feature. So we select such sequence and label them in a way which will break the feature into two. For example, the sequence shown in Figure S4.11(c) is labeled as 1. So that IsoGrouping will learn to output '1' for this sequence and this will break it into two features as shown by the dotted rectangles.

- Finally, we fine tune the model with false positives which are essentially some feature-like noisy or secondary signals appearing very close to the main isotopic signal (Figure S4.11(d), topmost rectangle). Our initial trained model reports those feature-like signals as peptide features (Figure S4.11(d), bottom rectangle) and as a result we get about 200,000 peptide features in total. We visually traced some of the false positive features and realized that those are actually secondary signals. Although random noise removal is learned easily by PointIso but separation of feature like noisy traces and those secondary peaks removal is difficult for PointIso. This tasks has to be performed by IsoDetecting module since it has access to the whole context. So it has to see through all the signals and decide which ones are to be reported and which are

to be ignored. In order to collect those wrong reports, we match the peptide feature list returned by PointIso (for the 4^{th} replicate of dilution sample 9, in fold 1) with the peptide feature list produced by all other tools (i.e., OpenMS, MaxQuant, Dinosaurs, and PEAKS) with an error tolerance of $0.01 m/z$ and 0.2 min RT. The features from PointIso which neither match with other tools nor with the identification result by MASCOT are selected for fine tuning (about 70,000 features). As mentioned in the Method section, a training sample for the IsoDetecting module consists of a target window and its four surrounding regions. So we cut out those false positive features from the respective LC-MS map keeping the feature in the target window. All the datapoints in the target window which do not belong to primary signal are labeled as '0'. Then these newly generated training samples (about 30,000) are used for retraining the already trained model, by running few epochs with 0.0001 learning rate and keeping the best model state. This same set of features is also used for fine tuning IsoGrouping module. In order to make a training sample for fine tuning IsoGrouping module, we pick a false positive (isotopes are secondary signals) and cut a sequence of 5 frames each holding the respective secondary signals. Then we label the sequence as '0' indicating that no feature exists in that sequence. We can actually balance between the true positive detection rate and false positive detection rate using the amount of such training samples. Without any fine tuning we get 99.50% detection, with over 200,000 total features. After fine tuning with 30,000 samples we have 98% detection with over 120,000 features. If we fine tune with 50,000 samples then total number of features drop to about 80,000 but the detection percentage also falls to 97.53%. So it seems like if we are too strict about discarding the secondary signals, we may also miss some true lower intensity peptide features which are close to other higher intensity peptide features. It should be an interesting research scope to see if we can solve this problem without a reduction in detection percentage. Besides that, we believe the necessity of this step also depends on the dataset and the mass spectrometer used. Because high-resolution mass spectrometers usually produce narrower signals without those secondary peaks, thus we might avoid this step and obtain over 99% detection. In our manuscript we have used fine tuned model of IsoDetecting module. But the other models are also uploaded in the GitHub repository. Users can choose according to their need. They can also fine tune further if necessary.

4.5 Data & Code Availability

The benchmark 3D dataset is available to download from ProteomeXchange using accession number PXD001091. Source code can be found at this address: https://github.com/anne04/PointIso

Chapter 5

Assessment of PointIso for the Practical Application and Extension for 4D Peptide Feature

Given a disease state, proteomics can directly target the involvement of specific proteins, protein complexes, and their modification status. Because most diseases can be characterized by changes in the protein level, e.g., D-dimer for COVID-19, Troponin for heart disease, Creatinine for kidney functionality, and many others. Therefore, proteomics has considerable promise in medical science, e.g., drug discovery [3], cancer immunotherapy through discovering neoantigen [6], and understanding pathogens, e.g., virus and bacteria. In particular, peptide feature detection from the LC-MS map is potential for disease biomarker detection. For instance, PEPPeR [31] demonstrates the discovery of novel proteins by post annotation of LC-MS peptide features without prior knowledge of their identities or even their presence in the mixture. Besides that, it directly relates to the Label-Free Quantification (LFQ), a widely used technique for protein quantification [4]. In the following sections, we will discuss the potential of PointIso model in two crucial application pipelines: disease biomarker detection through targeted MS/MS and relative protein abundance calculation through LFQ. Then we will show the way of adapting our model for 4D data, to show that our model can be made generic to be applicable in a wide range of problems. (Feature quality assessment presented in Section 5.1.1 and model extension in Section 5.3 are published in PointIso publication [88]. But the performance in label-free quantification is not published yet. We intend to publish it in future along with some model improvement through point transformer model.)

5.1 Disease Biomarker Detection

Peptide feature detection methods should detect reliable peptide features so that the model is trustworthy to identify novel proteins responsible for diseases, thus assisting treatment planning. Besides that, it can also help in neoantigen detection for cancer immunotherapy. Neoantigens are some novel peptides having a short length (8/10 amino acids) generated on the surface of malignant tumor cells. If those neoantigens can be identified, then people can engineer T-cells to target those specific cancer cells to kill. However, neoantigens may not show very high abundance in the LC-MS map, and are thus challenging to detect. Fortunately, our PointIso has shown better detection of peptide features than other competitive tools, even for low abundant peptide features. Therefore, it has the potential for the detection of disease biomarker discovery, including neoantigens. In Figure 5.1(a), we show an illustration of how a peptide feature detection tool can be used for disease biomarker discovery through targeted MS/MS. This method consists of multiple rounds of MS/MS identification. Each time, peptide ions are collected from different locations of MS1 or LC-MS map in terms of RT time range and m/z range and sent to the MS2 for fragmentation.

We start with collecting protein samples from healthy and disease-afflicted people for disease biomarker discovery. Then apply peptide feature detection on both samples and compare the found peptide features. Then we mark the peptide features showing significant abundance differences between the healthy and disease samples as candidate biomarkers. That is, we target those specific regions for sampling the precursor ions and send them to the next mass spectrometer (MS2) for generating fragment ion spectra or MS2 data [31]. Finally, we apply database search or De Novo Peptide sequencing on those MS2 data to identify candidate biomarkers. Now, since this method consists of multiple rounds depending on detected peptide features, if the software generates many false positives, then this approach may incur a high cost. So one important aspect is to verify whether PointIso is generating reliable true peptide features or too many false positives. Therefore, we perform some feature quality assessments for that purpose, as shown in the following section.

5.1.1 Feature Quality Assessment

We have seen that PointIso is detecting 98% identified peptide features, but there are many features that are not identified. This happens because the Data Dependent Acquisition (DDA) mode selects the most abundant peptides for fragmentation and identification. As

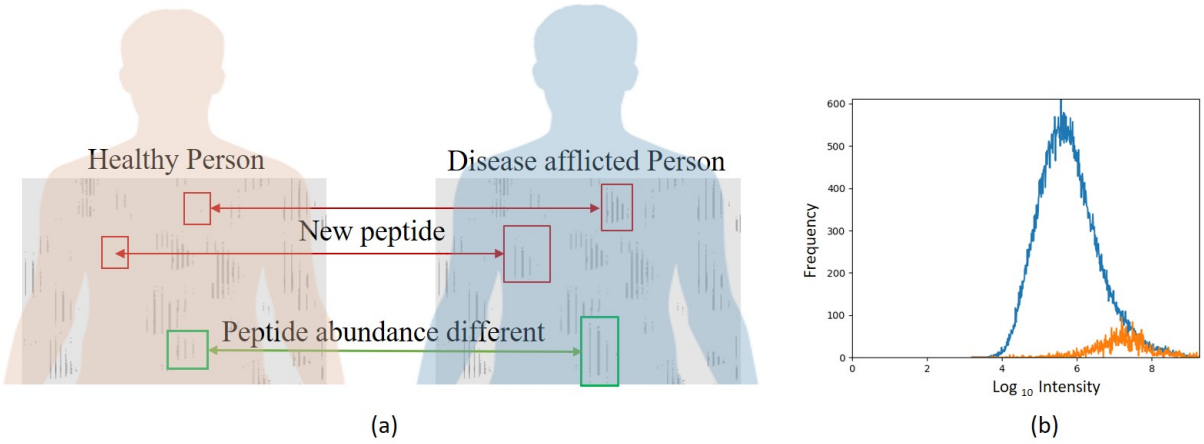


Figure 5.1: How peptide feature detection helps in biomarker detection and feature intensity distribution. (a) Disease biomarker discovery. (b) Intensity distribution of identified (orange) and detected (blue) peptide features.

a result, many actual peptides having lower abundance remain unidentified. The absolute number of peptide species eluted in a single LC run can be over 100,000, as discovered by Michalski et al. [46]. Therefore, during biomarker identification through relative protein quantification, it is desirable to assess the identity of those remaining peptide features which do not readily match with any MS/MS identification but show significant abundance change among multiple samples. This is done by performing post-annotation of the remaining peptide features through targeted MS/MS [31]. That is why this is important to verify whether the PointIso detected, but non-identified features are potential peptide features or not. The typical statistical methods for finding the false positive rate are not applicable here due to the absence of ground truth about the existence of those non-identified peptide features [66, 87]. That is why, although in our initial study with naive CNN, we calculated 'Specificity' in Section 2.3.2, later we realize that such metrics are not well-suited with our context. Instead of that, some reliable techniques are: to observe their intensity distribution and physiochemical properties, which are presented below.

- Intensity distribution of detected peptide feature should be log-normal. If there are many false positives, then there will be multiple peaks in the distribution [46]. We present the intensity distribution of the detected features by PointIso in an LC-MS/MS run in Figure 5.1(b). We see that our distribution is log-normal. We also see, the distribution of identified features (orange) is wedged into the high-intensity tail of the distribution of detected peptide features (blue), as only high-intensity features

are selected for fragmentation. If we saw the orange distribution is left aligned or in the middle of the blue distribution, it would have indicated something wrong in the model. However, that is not happening, so our model is generating accurate distribution.

- We investigated the physicochemical properties of the two populations (blue and orange) as shown in Figure 5.2. The comparison between all the software is shown in Appendix D.1. The columns represent mass, m/z, and RT left to right. The rows represent distributions of peptide features for PointIso, MaxQuant, and PEAKS top to bottom (blue). The orange distributions remain the same along the column since its representing the identified peptides. Plots in the first column show that all algorithms might have some false positives below 1000 Da mass, but the rate is lowest for PointIso. MaxQuant may have some false positives above 2000 Da as well. Then the second column represents the distribution of m/z. Again, MaxQuant might have some false positives in the higher range of m/z. Finally, the third column presents the distribution of RT, and all the tools have a probability of detecting false positives above 100 minutes RT, but that rate is lower for PointIso (and OpenMS in Appendix). Besides that, all other software might report some false positives below 30 minutes RT, but not PointIso.Therefore, we can say that the histograms of detected features (blue) by PointIso show good alignment with the histograms of identified features (orange) and support the correctness of the detected features.

Although we cannot validate whether the detected peptide features that are not matched with identified peptides are true or false, we can have some intuitions about how reliable they are, based on these statistical analyses. Our feature quality assessment demonstrates that the PointIso detected peptide features have a good probability of being true peptides [46]. So, PointIso has the potential to be applied in disease biomarker discovery.

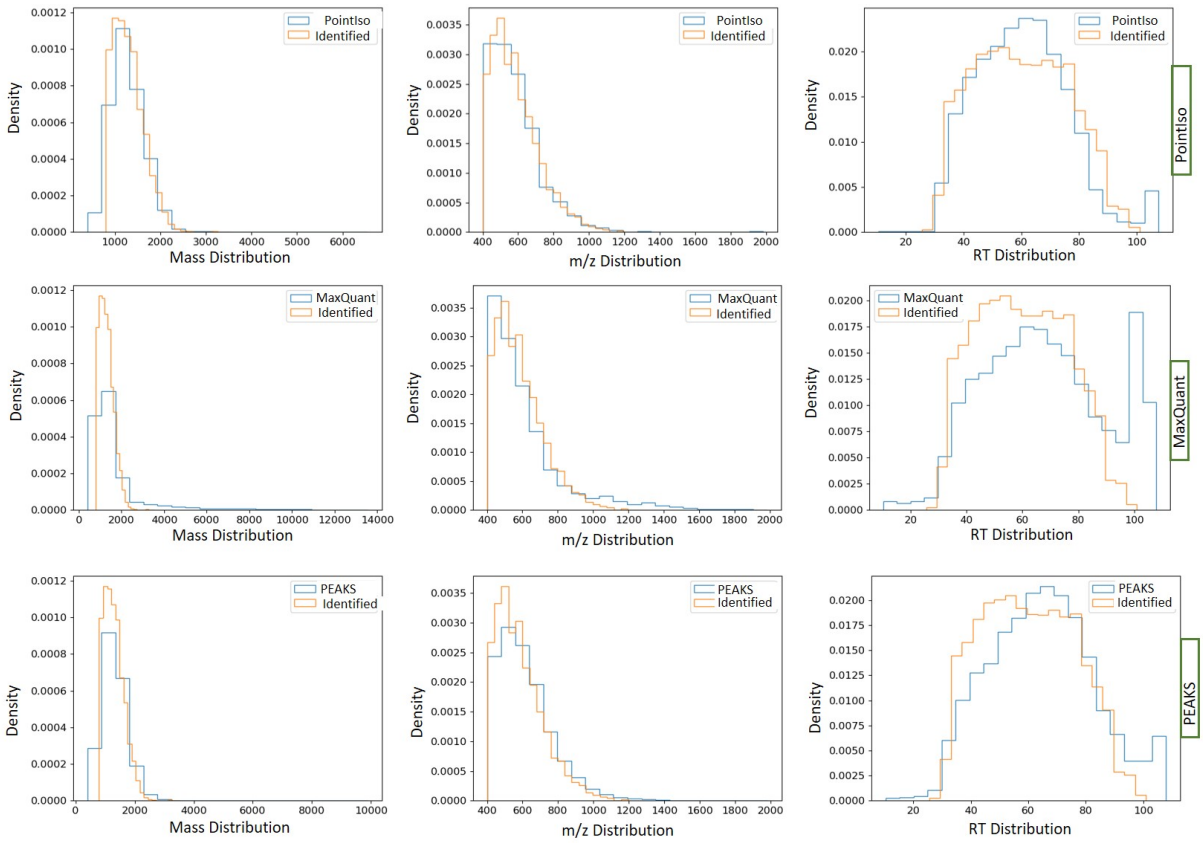


Figure 5.2: Comparison of mass, m/z, and RT distribution of detected features (blue) and identified features (orange) for different tools. Good alignment between the blue and orange distribution indicates a better probability of detected features being the true feature.

5.2 Label-Free Quantification (LFQ)

In a typical workflow of LC-MS/MS analysis, there are two mass spectrometers in tandem, MS1 & MS2. The first mass spectrometer, MS1, generates LC-MS map used for peptide feature detection. Then, the second mass spectrometer, MS2, generates fragment ion spectra of the precursor ion (found from the first step) to identify the peptides. Now, to quantify the identified peptides, we have to map those back to the peptide features found from MS1 data or LC-MS map, as shown in Figure 5.3. That is why we need the good performance of peptide feature detection; otherwise, there will be a wrong quantification of identified peptides.

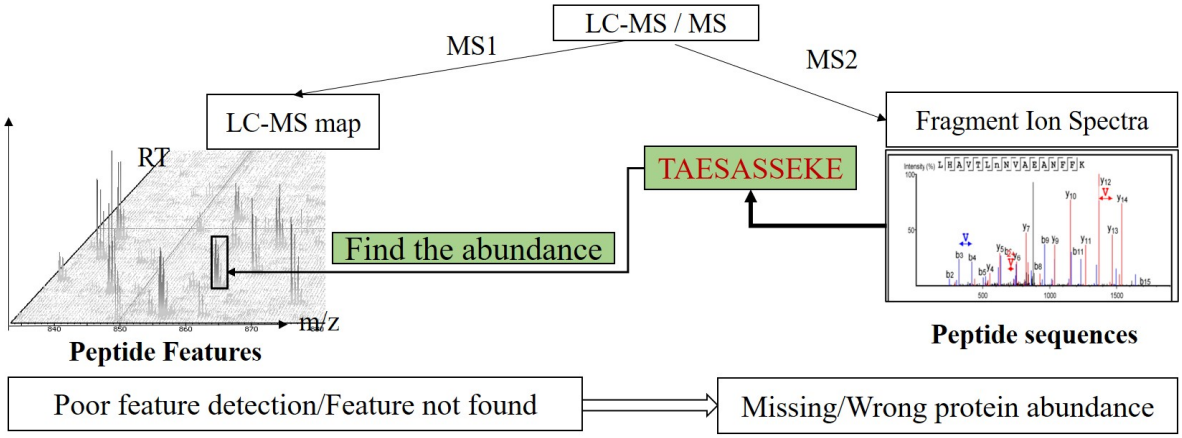


Figure 5.3: How to find protein quantity. After a peptide is identified from MS2 data (fragment ion spectra), that peptide is mapped to its corresponding peptide feature in the MS1 data to get its total intensity, which is used for quantity calculation of that peptide.

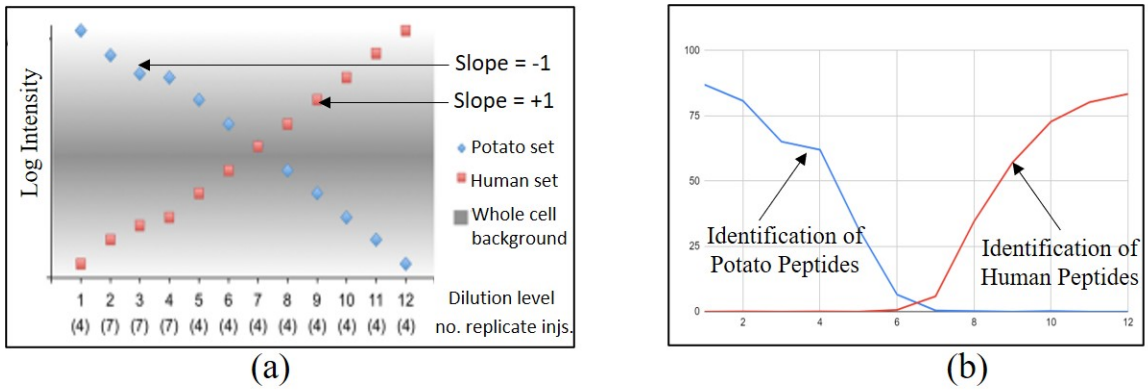


Figure 5.4: (a) Theoretical injected amount of spiked peptides (the figure is adapted from the paper by Chawade et al. [9]); (b) MASCOT identification of spiked peptides. It shows that sample 1 to sample 7, mostly potato peptides are identified by MASCOT due to very low concentration of human peptides in those samples. Similarly, mostly human peptides are identified in sample 6 to sample 12, due to very low concentration of potato peptides in those samples.

Label-free quantification (LFQ) measures the relative protein abundance among multiple samples, directly depending on peptide feature detection. LFQ is also used in disease biomarker discovery. For instance, Atrih et al. [4] applied LFQ in resected renal cancer

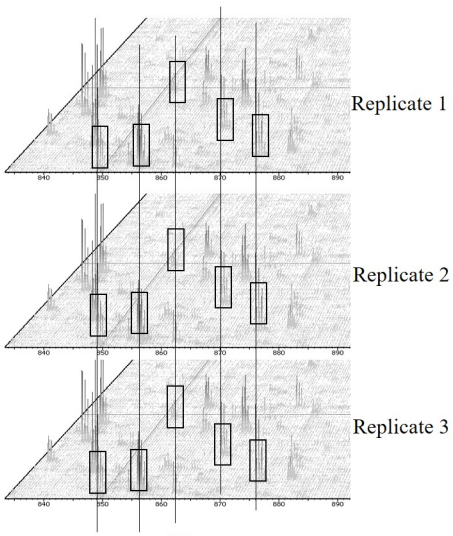
tissue for biomarker discovery and profiling. **LFQ has its own extensive research domain and mainly consists of statistical analysis, thus out of our scope. However, LFQ has some basic steps which are sufficient to verify whether a peptide feature detection model is applicable in the pipeline or not. We apply those essential steps to assess the potential of PointIso in this context.** We will first discuss the dataset we use for the assessment and then elaborate the LFQ steps.

5.2.1 Dataset

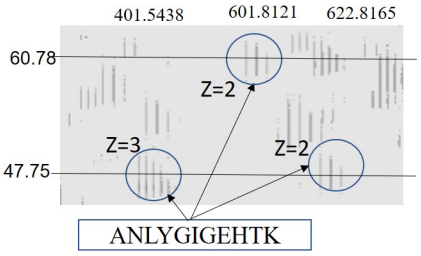
We downloaded the benchmark dataset from ProteomeXchange (PXD001091) that was prepared by Chawade et al. [9] for evaluating LFQ performance by different software. Therefore, we use this dataset for evaluating LFQ by PointIso model. The data is obtained through data-dependent acquisition (DDA). The samples consist of a long-range dilution series of synthetic peptides (115 peptides from potato and 158 peptides from human) spiked in a background of stable and non-variable peptides, obtained from *Streptococcus pyogenes* strain SF370 [68]. This is the same dataset used in the previous chapter for evaluating the PointIso model. It consists of 12 samples with variable concentrations of spiked proteins. From sample 1 to 12, the concentration of human spiked peptides increase, and potato spiked peptides decreases, as shown in Figure 5.4(a). However, background peptides stay stable, as shown in the figure. In Figure 5.4(b), we see the percentage of identified potato and human peptides by MASCOT database search for the different samples. We see that only potato peptides are identified for Sample 1-7. Because in these samples, the concentration of human peptides is very low. On the other hand, mostly human peptides are identified for Sample 6-12, due to the very low concentration of potato peptides in these samples. Now, if we measure the slopes of concentration of potato peptides among 12 samples, we will get a negative one slope due to decreasing concentration. Similarly, we will get a positive one slope for the human peptides since their abundance increases from Sample 1 to 12. Therefore, we will see if PointIso and other tools support this slope direction to verify and compare our model's performance.

5.2.2 LFQ steps

There is extensive amount of literature on LFQ procedure, mostly involving statistical analysis, and out of our scope. However, we perform some basic steps of LFQ which are good enough to verify whether a model is potential in LFQ application.



(a)



(b)

Peptide	Abundance
VCWPVPNK	15631140.5399
VVDAFSHLETGR	59296111.59
EALAFYAR	23249685.825
LSGPIPESFSK	132071693.737
…	…

(c)

Figure 5.5: (a) Alignment of peptide features over multiple replicates. (b) Same peptide sequence ANLYGIGEHTK is mapping to different peptide features, having charge 2 and 3. We may take the sum or maximum of those peptide features' intensity to get the quantity/abundance of this particular peptide. (c) Peptide abundance list for a sample LC-MS map.

First, we find the peptide abundance in a sample. Intensity of a peptide feature is found by adding the area under the beta distributed (e.g., $(\alpha = 2, \beta = 2), (\alpha = 2, \beta = 5), \dots$) isotopic signals in that feature. Each sample has multiple replicates since each experiment is repeated multiple times to improve accuracy in terms of feature coverage and intensity.

1. So the first step is we align the peptide features among the multiple replicates of the same sample in terms of m/z, RT peak, charge z, and peptide sequence, as shown in Figure 5.5(a).

2. Then we take the maximum intensity for each alignment. So we have a peptide feature list along with the maximum intensity of peptides over all replicates. (Some replicate may miss a feature due to noise, that is okay.)

3. It may appear that multiple peptide features having different m/z, RT, and z map to the same peptide sequence, like Figure 5.5(b). In that case, we take the sum or maximum of their intensities to represent the abundance of that particular peptide.

4. In this way, we have a peptide abundance list for each of the 12 samples, as presented in Figure 5.5(c).

Second, we have a peptide list and their abundance for each sample, we compare them to get the relative peptide abundance among the 12 samples. One standard way of doing that in our dataset is observing the slope of peptide intensities from Sample 1 to Sample 12, according to the creator of this dataset [9]. For each peptide sequence, we have a list of concentrations found from Sample 1 to 12. We draw a plot of log intensity vs. samples and take the slope of it, as shown in Figure 5.6. This gives a slope for each peptide.

Finally, since we have slopes for all the peptides, we draw a plot showing the distribution of potato, human, and background peptide slopes separately, as shown in Figure 5.7.

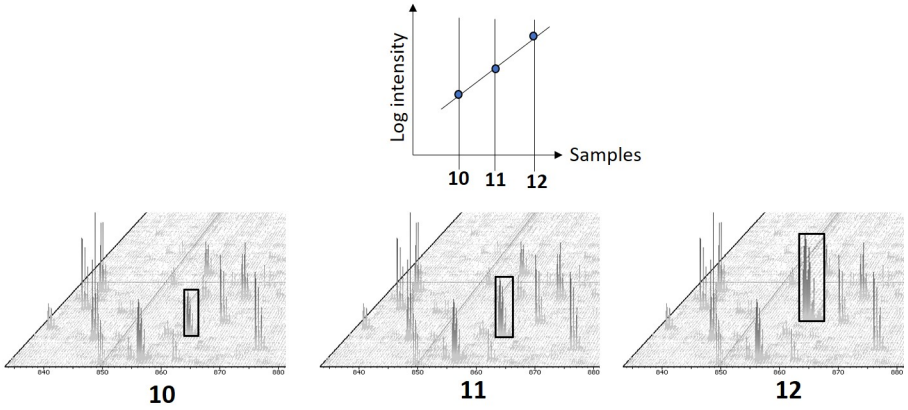


Figure 5.6: Abundance comparison for a particular peptide (shown in black box) among multiple samples. Since its abundance increases, therefore, slope in the top is positive.

5.2.3 Evaluation of PointIso for LFQ

The distribution of potato peptides should have a peak close to -1, and the distribution of human peptides should have a peak close to +1. In Figure 5.7(a), we see that the peaks of the distributions are very close to the ideal one for potato and background peptide and slightly right shifted for human peptide. In Figure 5.7(b), we compare the distributions of human and potato peptides among multiple software. When the distribution peak for the human peptide is over +1, and the distribution peak for potato peptides is below -1, it indicates slope overestimation. Slope overestimation indicates a larger fold change

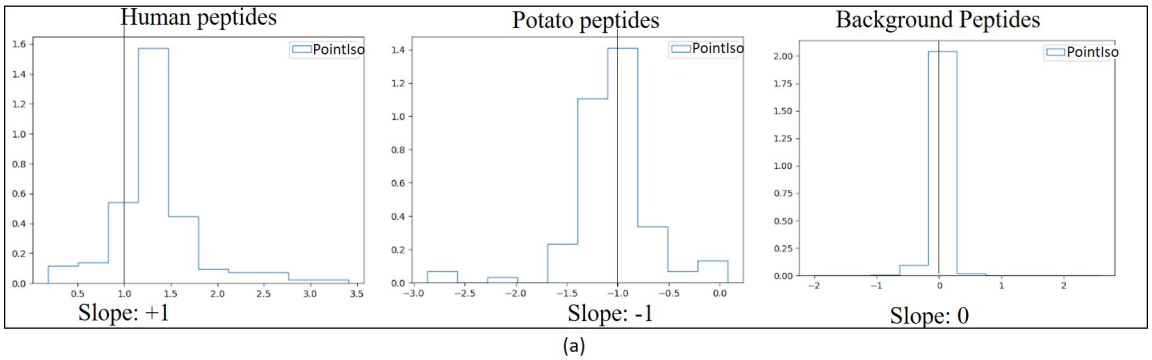


Figure 5.7: (a) Distribution of potato, human, and background peptide concentration slopes over 12 samples by PointIso only; (b) Distribution of human and potato peptide concentration slopes are compared among multiple software. It indicates that PointIso, like all other software, is potential for LFQ.

than the underlying one. However, it is not a problem since usually we need to know if a protein is up or down regulated than to what extent [9]. However, we must ensure that the distribution stays within the correct region, i.e., greater than 0 for human peptides (abundance goes up from sample 1 to 12) and less than 0 for potato peptides (abundance goes down from sample 1 to 12). Otherwise, it will indicate that a model does not detect the features correctly and result in a wrong abundance. In Figure 5.7, we see all of the tools produce distribution within the correct region. Although we see few traces in the wrong region, those are usually removed using outlier detection as a post-processing step to refine the measurements for better quantification. Therefore, we can say that the PointIso model, just like other models, is potential in the label-free quantification pipeline.

5.3 PointIso Extension for Higher Dimensional Data

Next, we discuss another important contribution of our research, adapting the 3D PointIso model to handle the additional dimension introduced by the latest technology, namely the 'trapped ion mobility' spectrometry with Time-of-flight instrument (TimsTOF). It adds ion mobility ($\frac{1}{k0}$) as another dimension of separation [45]. Such instruments offer several desirable properties for the analysis of complex peptide mixtures and becoming popular in shotgun proteomics, e.g., analyzing immune suppression in the early stage of COVID-19 disease [70]. Usually, these features are very close to each other and challenging to detect by existing peptide feature detection algorithms. For example, PEAKS has about 20% MS/MS identified TimsTOF peptides for which it can not detect any corresponding LC-MS peptide features. As a result, those peptides cannot be quantified. Moreover, very few existing 3D peptide feature detection software have extensions for supporting additional dimensions because those depend on many parameters that have to be set by careful experiments. However, we can easily adapt our model through minimal architectural changes and learn the parameters by training on an appropriate dataset. This section shows how to adapt our original 3D model to handle the additional axis information, i.e., 4D LC-MS map. We believe this adaptation capability of our PointIso model with other contexts should make it more appealing in the proteomics society.

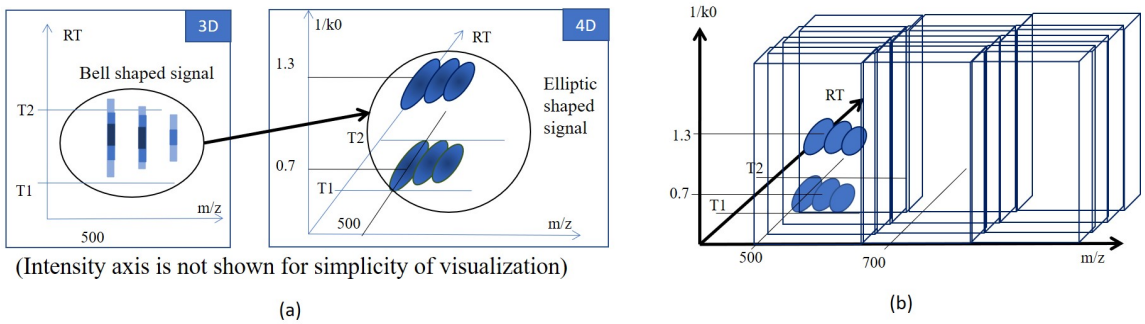


Figure 5.8: Illustration of 4D peptide features. For simplicity, we are not showing the Intensity axis. (a) In the left, we see a usual peptide feature in $[m/z \times RT]$ plane. This same feature can get separated into two different features if we consider the additional $\frac{1}{k0}$ dimension, as shown right to it. (b) Scanning windows of PointIso cover full range of $1/k0$ while scanning the MS1 TimsTOF data.

In Figure 5.8, we show the peptide features in 3D and 4D space, where we avoid the intensity (I) axis for simplicity. In (a), we see how a peptide feature in a three-dimensional

plot can be separated into two different features when the ion mobility dimension is considered. Before stating the changes we bring in our model, we would like to briefly discuss the existing technique of 4D TimsTOF feature detection by MaxQuant in Figure 5.9. We see that the space in $(RT \times \frac{m}{z} \times \frac{1}{k_0})$ is sliced into multiple $(RT \times \frac{m}{z})$ planes. Then for each plane, it applies the conventional MaxQuant algorithm. After that, the overlapping detection areas are clustered across slices to obtain a feature in $(RT \times \frac{m}{z} \times \frac{1}{k_0})$ space again. That is, MaxQuant treats the 4D peptide feature detection problem as a sequence of 3D peptide feature detection problems. On the other hand, PointIso directly takes 4D datapoints as input and detects features in the 4D space. Besides these, another method is named IonQuant, which also works on 4D TimsTOF data. However, it is focused on peptide identification and does not have a peptide feature detection step. Rather, it applies Extracted Ion Chromatogram (XIC) technique only for the identified peptides for their quantification. Therefore, IonQuant is not really suitable for biomarker discovery by the postannotation technique. So this is out of our context, and we do not use this method in our comparative analysis.

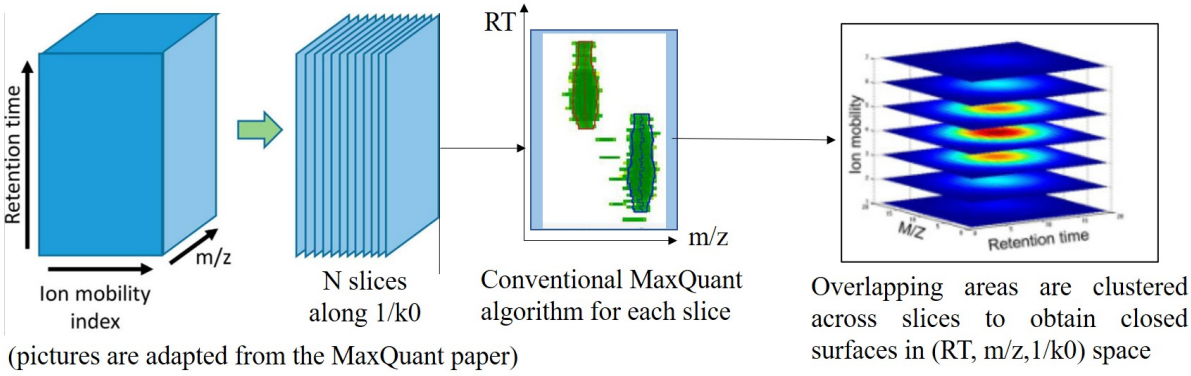


Figure 5.9: Workflow of MaxQuant for processing 4D peptide features. The space in $(RT \times \frac{m}{z} \times \frac{1}{k_0})$ is sliced into multiple $(RT \times \frac{m}{z})$ planes. Then for each plane, it applies the conventional MaxQuant algorithm. After that, the overlapping detection areas are clustered across slices to obtain a feature in $(RT \times \frac{m}{z} \times \frac{1}{k_0})$ space again.

5.3.1 Adaptation Strategy

Extension of IsoDetecting module for 4D data

The significant changes we had to make are in the IsoDetecting module since it has to segment 4D points now. The scanning window dimension covering 2.0 m/z and 15 minute RT now also spans along the new $\frac{1}{k0}$ dimension. To keep the changes minimal, we let each scanning window cover the whole range of $\frac{1}{k0}$ as shown in 5.8(b), (0 to about 1.5 with the resolution of up to 5 decimal points) so that we do not need to translate the scanning window along the new axis. However, this increases the number of datapoints by three times (because each $(m/z, RT, I)$ point coordinate in 3D space can give rise to two or more 4D point coordinates: $(m/z, RT, k1, I1), (m/z, RT, k2, I2), \dots)$, which is not feasible to load in GPU memory. In terms of Tensors, 3D PointIso model needs [batch size, 5000, 3] sized Tensor whereas 4D PointIso model would need [batch size, 20000, 4] sized Tensor. That is why we reduce the scanning window dimension to 1.0 m/z and 10 minute RT. Besides that, we read the $\frac{1}{k0}$ values every third datapoint in the ion mobility dimension. After making this alteration, the number of datapoints within a scanning window is about 6000, which is loadable in our GPU memory (Tensor size: [batch size, datapoints, n]=[batch size, 6000, 4]). Now, the point cloud input consists of a vector of 'n' datapoints each having 4 axis information: m/z, $\frac{1}{k0}$, RT, and I. To make the GPU matrix calculation manageable, we reduce the number of neurons and layers in the IsoDetecting module. In particular, the following modifications are performed on the IsoDetecting network (please refer to Figure 4.6):

- The last layer in the 'Input Transformation by T-Net' has now 4×4 dimension since we are dealing with 4D data.
- There is only one layer having 16 neurons in between the 'input transformation' and 'feature transformation' networks.
- Last layer in 'feature transformation' network has 16×16 dimension.
- During finding the point feature, the concatenation of global feature and local feature gives $n \times (16 + 32)$ dimension since we are doing 16×16 feature transformation now.
- At the end, before the output layer, we keep three fully connected layers (256, 128, 64), and remove the last one having 16 neurons.

Besides bringing these changes to the network, we take the entire region of the left and right windows for the attention calculation network. Because the features having charge

1 have isotopes 1.0 m/z distance apart. But for the upper and bottom window, we take 50% region as it seems enough for feature detection. We should mention that we had to reduce the network size of the IsoDetecting module due to a shortage of GPU memory. Therefore, we encourage users to observe the performance of PointIso with 4D data keeping the original dimension if possible.

Extension of IsoGrouping module for 4D data

After we have the cluster of isotopes found from the IsoDetecting module, we insert an additional $\frac{1}{k0}$ finding step. In this step, each cluster is analyzed along $\frac{1}{k0}$ axis. We know each cluster has an elution time range or retention time (RT) range. At each point of that range along the RT axis, we have a scan $\frac{1}{k0} \times m/z$, that is bounded by the m/z range found from the IsoDetecting module. So within that bound, we look for a group of equidistant vertical traces that presents the ion mobility signal. Then those groups are merged over different RT points within the range, and we get a complete set of clusters having m/z, $\frac{1}{k0}$, and RT values listed. A separate script for this additional $\frac{1}{k0}$ finding step is uploaded in Github repository, mentioned in the Data Availability section.

After we have the final list of clusters, while sending those to the IsoGrouping module, we actually compress the scans along the ion mobility, i.e., $\frac{1}{k0}$ dimension, and two-dimensional frames are passed to the IsoGrouping module. We do this to keep the changes minimal.

Next, in the IsoGrouping module, we avoid weight sharing among the five frames, and different weights are learned for different frames. We believe the reason behind this is the higher complicity associated with the 4D patterns.

5.3.2 Dataset

High throughput and robustness are essential requirements to integrate mass spectrometry-based proteomics into biomedical research, especially in clinical settings. Evosep One instrument, a new high-performance liquid chromatography (HPLC) instrument, employs an embedded gradient and is capable of fast turnaround between analyses [5]. It is promising for rapid yet comprehensive analysis, e.g. analysing protein interactomes or quantification of trace-level host cell proteins (HCPs) in recombinant biotherapeutics. That is why we performed the experiments on a 4D LC-MS map generated by Evosep One instrument. We downloaded this dataset from ProteomeXchange with accession number PXD010012. It contains human cervical cancer cells (HeLa) grown in Dulbecco's modified Eagle's medium

with 10% fetal bovine serum, 20 mm glutamine, and 1% penicillin-streptomycin (all Life Technologies Ltd., Paisley, UK) [45]. This dataset has 16 LC-MS maps or raw files. The common set of feature lists produced by MaxQuant 1.6.3.3 [53] and PEAKS Studio X is used for the training, as before.

5.3.3 Evaluation of PointIso for 4D dataset

In the evaluation step, we used MSFragger-3.2, an ultrafast and comprehensive peptide identification tool in mass spectrometry-based proteomics [34] that is free to use for academic purposes (our licensed MASCOT version does not have support for processing the latest TimsTOF data). The MS/MS spectra were matched to human reference proteome (Uniprot, 2021/06, including isoforms) using the default closed search settings parameters. In the dataset, two mobile phases, A (12 samples) and B (4 samples) were water with 0.1% formic acid (v/v) and 80/20/0.1% ACN/water/formic acid (v/v/vol), respectively. We perform a k-fold cross-validation, where we consider two settings: $k = 2$ and $k = 8$. With 0.01 $\frac{1}{k0}$, 0.01 m/z, and 0.2 minute RT tolerance, peptide feature detection by MaxQuant, PEAKS, PointIso - $k = 2$, and PointIso - $k = 8$ are 80.32%, 82.16%, 86.31%, and 86.27% respectively. The result for 16 samples is provided in Figure 5.10, and in Supplementary Table C.2. The running time of MaxQuant, PEAKS and PointIso for this 4D dataset are about 40 minutes, 25 minutes, and 50 minutes, respectively. Therefore, we can say that PointIso is adaptable to diverse datasets implying greater utility.

We show an adaptation strategy for PointIso that works well with the TimsTOF generated 4D data. We believe users can make simple modifications of the PointIso model in different ways based on the context and available GPU memory, which makes PointIso generalizable to various domains, making it more appealing in the proteomics society.

5.3.4 Data & Code Availability

Dataset can be downloaded from ProteomeXchange with accession number PXD010012. Source code can be found at this address: https://github.com/anne04/PointIso.

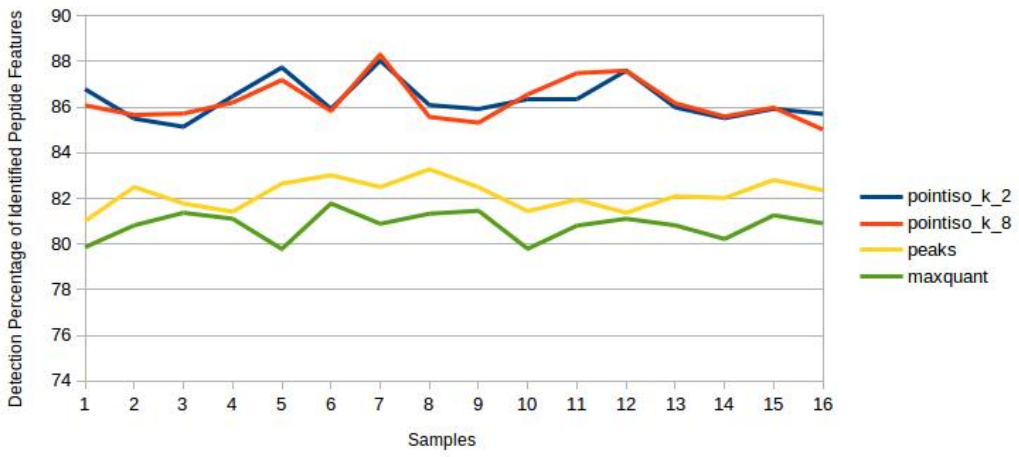


Figure 5.10: Detection Percentage of identified peptide features for different tools. We see that Pointiso is giving about 86% sensitivity, about 4% to 6% higher than MaxQuant and PEAKS.

Chapter 6

Conclusion and Future Work

Liquid chromatography with tandem mass spectrometry (LC-MS/MS) based proteomics is a well-established research field for discovering disease biomarkers, drug target validation, safety marker identification, mode of action (MOA) studies, and many other clinical research [3]. Peptide feature detection from LC-MS map has tremendous promise in disease biomarker detection through targeted MS/MS and Label-Free Quantification (LFQ). The existing deterministic algorithms for peptide feature detection mainly depend on many parameters that are supposed to be set by field experts through rigorous experiments and thus prone to human error. Some simple machine learning methods are available, but they involve a lot of feature engineering instead of automatically learning the underlying feature characteristics from the training data. For a particular pattern recognition problem, when it is very challenging to identify the features and an enormous amount of data are available, deep learning usually comes in very beneficial in solving the particular problem. Besides that, with the advancing technologies, many new information becomes available to us. However, it is hard to integrate those into the existing knowledge-based heuristic software. In contrast, we can fine-tune a deep learning model to adapt to the newly generated data. Therefore, it is worth investigating deep learning in the context of peptide feature detection from LC-MS map.

6.1 Main Research Contribution

We build a foundation for the application of deep learning to automate the peptide feature detection for the first time, where the model itself learns all the necessary parameters

through training on the appropriate dataset. Off-the-shelf machine learning or deep learning models are never well-suited for diverse proteomics problems, and we often need to adapt those models to fit in our context. Technically, we picked different types of deep learning models originally developed for computer vision and/or sequence classification domain. We then modified the model architecture to combine and fit them in our context. There were also some modules in our model which were designed & developed solely by ourselves, without using any external models. We evaluate our models based on the detection percentage of identified peptides, and our models give better detection than other existing methods in multiple datasets. The main contributions of this research work are as follows:

- We first start with a naive convolutional neural network [89] and some heuristic steps (discussed in Chapter 2). It helps us to understand the scope of deep learning in our context.

- To improve the model sensitivity and avoid heuristic steps, we proceed further and propose a more robust and fully deep learning based model DeepIso [87] (discussed in Chapter 3). In DeepIso, we combine FCRNN [81] (video clip classification) and attention-gated RNN [51] (frame selection in noisy video clip) to address our problem. DeepIso provides better feature detection than other software.

- DeepIso is a fixed precision model and comparatively slower than other tools. Therefore, we explore further and propose PointIso [88] (discussed in Chapter 4), which provides better detection with three times faster speed than DeepIso and supports arbitrary precision, thus suitable for very high-resolution data. In PointIso, we adapt PointNet [54] (point cloud based object detection) and DANET [20] (autonomous driving) to support attention based segmentation.

- Then we perform a feature quality assessment which shows that our model generates reliable peptide features and thus, has the potential for biomarker discovery [88] (discussed in Chapter 5.1). We perform this analysis because usually 10-50% of spectra generated from LC-MS/MS are correctly assigned for the identification in the proteomics field [18]. That means about half of the spectra, which are filtered during the identification process, may play important biological roles such as post-translational modifications. That is why we needed feature quality assessment.

- We also verify our model's effectiveness in the pipeline of Label-Free Quantification to make it more appealing in the proteomics society (discussed in Chapter 5.2).

- Finally, we show how our 3D feature detection model can adapt for 4D peptide feature detection as well [88] (discussed in Chapter 5.3). That means our model is generic and transferable to a wide range of problems making it more appealing in the proteomics society.

We also test our trained PointIso model on breast cancer proteomic data [24] (file P1_LN_1.RAW of project PXD012431 from ProteomXchange). We see that our model detects 72.6% identified peptide features, whereas, Dinosaur and MaxQuant detect only 63% and 50% identified peptide features respectively. It implies that our PointIso model can learn the peptide feature properties irrespective of peptide patterns or intensities seen during training time and does not overfit the training data. One important fact is, the dataset should be generated by the same type of instrument that was used for generating our training data: LTD Orbitrap XL ETD. If the instrument changes, then the pattern properties may also change, and thus, PointIso may not give optimal result (usual behavior for any deep learning model). However, few epochs of fine tuning may solve that problem as well. Therefore, once we train a model on a protein sample, the same model should be applicable (without further training) to other protein samples coming from different species but similar instrument, making it more appealing in the practical sectors. We believe our model may also assist in detecting neoantigens, which are essentially short-length novel peptides generated on the surface of cancer cells and usually show lower abundance than other usual peptides in the sample. Since our model is sensitive enough to detect very low-intensity features, it should help discover neoantigen peptides in the early stage, thus helping in cancer immunotherapy. Next, we would like to discuss some potential future scopes of application and model improvement.

6.2 Future Works

6.2.1 Chimeric Spectra Identification

There are two mass spectrometers in LC-MS/MS analysis, MS1, and MS2. First, MS1 captures the peptide features. Then, there are two ways of acquiring peptide ions for fragmentation by MS2. In Data Dependent Acquisition (DDA), highly abundant peptide ions are usually selected for fragmentation. However, co-eluting peptides cause fragment ion spectra from multiple peptides to overlap in MS2 or MS/MS data, called chimeric spectra. Also, in Data Independent Acquisition (DIA) all the peptide ions within a m/z and RT range are selected and sent to MS2 for fragmentation. Therefore, there is a higher

chance of multiple fragment ion spectra overlapping in MS/MS data. While applying database search or De Novo sequencing on MS/MS data for peptide identification, if we do not separate those overlapping fragment ion spectra, i.e., chimeric spectra, it may lead to wrong identification. Therefore, we have to separate those, and for this purpose, we have to map those MS/MS spectra to corresponding peptide features in MS1 data. A better peptide feature detection tool can give better separation, thus resulting in better identification. For instance, Dinosaur [67] increases the chimeric identification on DDA data from 26% to 32%. DeepNovo-DIA [74] uses peptide feature detection for chimeric spectra separation from DIA data. Therefore, it should be an interesting future scope to see if the application of PointIso can improve chimeric identification.

6.2.2 Model Improvement through Semi-supervised Learning

Our current deep learning methods follow a supervised learning approach, and thus, it requires a huge amount of annotated data. Each LC-MS map or MS1 data can have over 50,000 peptide features, and each experiment usually involves multiple LC-MS maps. It results in hundreds of thousands of peptide features. However, those features are unlabelled, and human annotation is out of scope [3]. We also have a few thousand ground truth data (identified peptides), but that is insufficient for deep learning model training. Therefore, we used a common set of peptide features generated by other software as training data. Although this training let us achieve high sensitivity, but our model suffers from duplicate reports of the same peptide features due to secondary peaks, which causes a higher number of peptide features generated by our model. Currently, we can reduce the number by imposing some threshold feature score (based on Softmax layer), however, it also reduces the sensitivity. Therefore, we would like to eliminate those redundant reports to make the LC-MS/MS analysis cost-effective without compromising the performance. We are not certain whether our training data is responsible for this problem of redundant features (since the training data is a common set of existing tools). However, if we could perform unsupervised learning, we would have made our model totally independent of existing tools. It also has the potential of solving redundant feature problem. That is why we propose an idea of 'unsupervised pretraining' method using transformer models [15]. Transformer models are made from auto-encoders and are usually used for natural language processing. However, recently it has been gaining popularity in object segmentation problems as well [85]. It consists of two steps:

1. Step 1 - Unsupervised Pretraining: We can use hundreds of thousands of unlabelled data in this step. In Figure 6.1 we see that the input datapoints are arranged in an

input vector, having a length (rows) of about 5000 (total number of datapoints in the scanning window of current model), and dimension (columns) of three (m/z, RT, and I). Random regions in the input vector can be masked, and then those masked values are predicted in the output. During this process, some weight will be learned for the transformer, which is in the middle.

2. Step 2 - Supervised Refining: In this step, we use our few thousand ground truth data, i.e., identified peptide features. Please refer to Figure 6.2 for clarification. Now, no regions in the input vector are masked, and in the output, we predict the input datapoints' class or charge (z). During this process, the transformer weights learned in the previous step are frozen. Before the output layer, some feed-forward layers of neurons are inserted, and those additional weights are learned in this step. Usually, this refining step needs much fewer epochs to converge than the model from scratch would need. It also does not need a huge amount of annotated data.

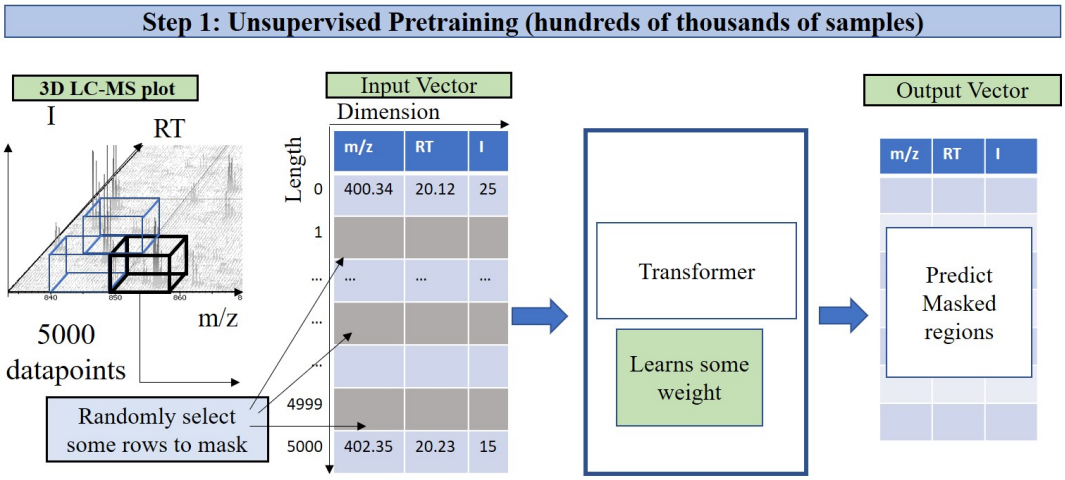


Figure 6.1: Unsupervised pretraining of the transformer model

Besides these, testing the PointIso model on various other datasets, specifically disease datasets, should be an important scope of future assessment of the model. We may also try to apply our model on data obtained in centroid mode instead of profile mode (our current mode), to see if it reduces the false positives or redundant features. Adapting our model to fit another significant proteomics problem: Intact Mass Analysis, can be another future work. Finally, whether we can make our model end-to-end instead of using two separate modules (i.e., IsoDetecting & IsoGrouping) to have better speed (also less chance of a false positive), is an exciting research direction.

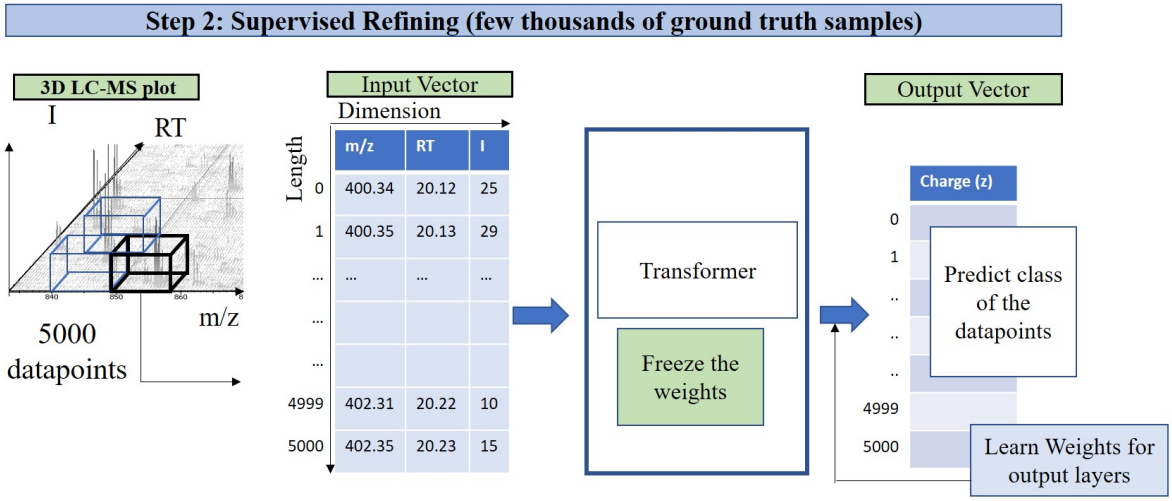


Figure 6.2: Supervised refining of the transformer model

This thesis demonstrates that novel deep learning tools are desirable to advance the state-of-the-art in LC-MS/MS analysis. Besides that, our novel idea of attention-based 3D and 4D peptide feature segmentation technique can also serve the general pattern recognition domain. We believe our research reveals the superior performance of deep learning models than other existing methods in solving peptide feature detection problem and discloses the area of improvements to make it more robust and high performing in the near future.

6.3 Author Contribution and Acknowledgement

Fatema Tuz Zohora[1] designed and developed the model, and performed the experiments as well. M Ziaur Rahman[2] helped with producing database search result by MASCOT and setting parameters of other existing tools. M Ziaur Rahman and Lei Xin[2] helped with studying the characteristics of peptide features and LC-MS map. Ngoc Hieu Tran[1,2] contributed by suggesting various deep learning ideas. Baozhen Shan[2] and Ming Li[1] proposed and supervised the project. All of them reviewed the manuscript. We thank Rui Qiao[2], alumni of Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, ON, Canada, for sharing his thoughtful ideas about PointNet and class imbalance

[1]Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[2]Bioinformatics Solutions Inc., Waterloo, ON N2L 6J2, Canada

problems. This work is partially supported by the Canada Research Chair program, the National Key RD Program of China grants 2016YFB1000902 and 2018YFB1003202, and Bioinformatics Solutions Inc.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198, 2003.

[3] Ken Aoshima, Kentaro Takahashi, Masayuki Ikawa, Takayuki Kimura, Mitsuru Fukuda, Satoshi Tanaka, Howell E. Parry, Yuichiro Fujita, Akiyasu C. Yoshizawa, Shin-ichi Utsunomiya, et al. A simple peak detection and label-free quantitation algorithm for chromatography-mass spectrometry. *BMC Bioinformatics*, 15(1):376, 2014.

[4] A Atrih, MAV Mudaliar, P Zakikhani, DJ Lamont, J TJ Huang, SE Bray, G Barton, Stewart Fleming, and G Nabi. Quantitative proteomics in resected renal cancer tissue for biomarker discovery and profiling. *British Journal of Cancer*, 110(6):1622, 2014.

[5] Nicolai Bache, Philipp E. Geyer, Dorte B. Bekker-Jensen, Ole Hoerning, Lasse Falkenby, Peter V. Treit, Sophia Doll, Igor Paron, Johannes B. Müller, Florian Meier, et al. A novel lc system embeds analytes in pre-formed gradients for rapid, ultra-robust proteomics. *Molecular & Cellular Proteomics*, 17(11):2284–2296, 2018.

[6] Brendan Bulik-Sullivan, Jennifer Busby, Christine D. Palmer, Matthew J. Davis, Tyler Murphy, Andrew Clark, Michele Busby, Fujiko Duke, Aaron Yang, Lauren Young,

et al. Deep learning using tumor hla peptide mass spectrometry datasets improves neoantigen identification. *Nature Biotechnology*, 37(1):55, 2019.

[7] Salvatore Cappadona, Peter R. Baker, Pedro R. Cutillas, Albert JR. Heck, and Bas v. Breukelen. Current challenges in software solutions for mass spectrometry-based quantitative proteomics. *Amino Acids*, 43(3):1087–1108, 2012.

[8] Matthew C. Chambers, Brendan Maclean, Robert Burke, Dario Amodei, Daniel L. Ruderman, Steffen Neumann, Laurent Gatto, Bernd Fischer, Brian Pratt, Jarrett Egertson, et al. A cross-platform toolkit for mass spectrometry and proteomics. *Nature Biotechnology*, 30(10):918, 2012.

[9] Aakash Chawade, Marianne Sandin, Johan Teleman, Johan Malmstrom, and Fredrik Levander. Data processing has major impact on the outcome of quantitative label-free lc-ms analysis. *Journal of Proteome Research*, 14(2):676–687, 2014.

[10] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, 2014. Association for Computational Linguistics.

[11] Christopher J. Conley, Rob Smith, Ralf J. Torgrip, Ryan M. Taylor, Ralf Tautenhahn, and John T. Prince. Massifquant: open-source kalman filter-based xc-ms isotope trace feature detection. *Bioinformatics*, 30(18):2636–2643, 2014.

[12] Jürgen Cox, Marco Y. Hein, Christian A. Luber, Igor Paron, Nagarjuna Nagaraj, and Matthias Mann. Accurate proteome-wide label-free quantification by delayed normalization and maximal peptide ratio extraction, termed maxlfq. *Molecular & Cellular Proteomics*, 13(9):2513–2526, 2014.

[13] Jürgen Cox and Matthias Mann. Maxquant enables high peptide identification rates, individualized ppb-range mass accuracies and proteome-wide protein quantification. *Nature Biotechnology*, 26(12):1367–1372, 2008.

[14] Jurgen Cox, Nadin Neuhauser, Annette Michalski, Richard A. Scheltema, Jesper V. Olsen, and Matthias Mann. Andromeda: a peptide search engine integrated into the maxquant environment. *Journal of Proteome Research*, 10(4):1794–1805, 2011.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[16] Jeffrey Donahue, Lisa A. Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[18] Joshua E. Elias, Wilhelm Haas, Brendan K. Faherty, and Steven P. Gygi. Comparative evaluation of mass spectrometry platforms used in large-scale proteomics investigations. *Nature Methods*, 2(9):667–675, 2005.

[19] Matthew Fitzgibbon, Wendy Law, Damon May, Andrea Detter, and Martin McIntosh. Open-source platform for the analysis of liquid chromatography-mass spectrometry (lc-ms) data. In *Clinical Proteomics*, pages 369–381. Springer, 2008.

[20] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[21] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc., 2017.

[22] Siegfried Gessulat, Tobias Schmidt, Daniel P. Zolg, Patroklos Samaras, Karsten Schnatbaum, Johannes Zerweck, Tobias Knaute, Julia Rechenberger, Bernard Delanghe, Andreas Huhmer, et al. Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning. *Nature Methods*, 16(6):509, 2019.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[24] Talita H. B. Gomig, Iglenir J. Cavalli, Ricardo L. R. de Souza, Aline C. R. Lucena, Michel Batista, Kelly C. Machado, Fabricio K. Marchini, Fabio A. Marchi, Rubens S. Lima, Cícero de Andrade Urban, et al. High-throughput mass spectrometry and

bioinformatics analysis of breast cancer proteomic data. *Data in Brief*, 25:104125, 2019.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[26] Jinwei Gu, Xiaodong Yang, Shalini D. Mello, and Jan Kautz. Dynamic facial analysis: From bayesian filtering to recurrent neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1548–1557, 2017.

[27] Shenheng Guan, Michael F. Moran, and Bin Ma. Prediction of lc-ms/ms properties of peptides from sequence by deep learning. *Molecular & Cellular Proteomics*, 18(10):2099–2107, 2019.

[28] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[30] David H. Hubel and Torsten N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243, 1968.

[31] Jacob D. Jaffe, D R. Mani, Kyriacos C. Leptos, George M. Church, Michael A. Gillette, and Steven A. Carr. Pepper, a platform for experimental proteomic pattern recognition. *Molecular & Cellular Proteomics*, 5(10):1927–1941, 2006.

[32] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[34] Andy T. Kong, Felipe V. Leprevost, Dmitry M. Avtonomov, Dattatreya Mellacheruvu, and Alexey I. Nesvizhskii. Msfragger: ultrafast and comprehensive peptide identification in mass spectrometry–based proteomics. *Nature Methods*, 14(5):513–520, 2017.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[36] Ludmila I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[37] Ludmila I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Kyriacos C Leptos, David A. Sarracino, Jacob D. Jaffe, Bryan Krastins, and George M. Church. Mapquant: open-source software for large-scale protein quantification. *Proteomics*, 6(6):1770–1782, 2006.

[41] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[42] Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E Dahl, Timo Kohlberger, Aleksey Boyko, Subhashini Venugopalan, Aleksei Timofeev, Philip Q Nelson, Greg S Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[43] Bin Ma, Kaizhong Zhang, Christopher Hendrie, Chengzhi Liang, Ming Li, Amanda Doherty-Kirby, and Gilles Lajoie. Peaks: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry*, 17(20):2337–2342, 2003.

[44] Chunwei Ma, Yan Ren, Jiarui Yang, Zhe Ren, Huanming Yang, and Siqi Liu. Improved peptide retention time prediction in liquid chromatography through deep learning. *Analytical Chemistry*, 90(18):10881–10888, 2018.

[45] Florian Meier, Andreas-David Brunner, Scarlet Koch, Heiner Koch, Markus Lubeck, Michael Krause, Niels Goedecke, Jens Decker, Thomas Kosinski, Melvin A Park,

et al. Online parallel accumulation–serial fragmentation (pasef) with a novel trapped ion mobility mass spectrometer. *Molecular & Cellular Proteomics*, 17(12):2534–2545, 2018.

[46] Annette Michalski, Juergen Cox, and Matthias Mann. More than 100,000 detectable peptide species elute in single shotgun proteomics runs but the majority is inaccessible to data-dependent lc- ms/ms. *Journal of Proteome Research*, 10(4):1785–1793, 2011.

[47] Lukas N. Mueller, Mi-Youn Brusniak, DR Mani, and Ruedi Aebersold. An assessment of software solutions for the analysis of mass spectrometry based quantitative proteomics data. *Journal of Proteome Research*, 7(01):51–61, 2008.

[48] Lukas N. Mueller, Oliver Rinner, Alexander Schmidt, Simon Letarte, Bernd Bodenmiller, Mi-Youn Brusniak, Olga Vitek, Ruedi Aebersold, and Markus Müller. Superhirn–a novel tool for high resolution lc-ms-based peptide/protein profiling. *Proteomics*, 7(19):3470–3480, 2007.

[49] Mythreyi Narasimhan, Sadhana Kannan, Aakash Chawade, Atanu Bhattacharjee, and Rukmini Govekar. Clinical biomarker discovery by swath-ms based label-free quantitative proteomics: impact of criteria for identification of differentiators and data normalization method. *Journal of Translational Medicine*, 17(1):184, 2019.

[50] Patricia M. Palagi, Daniel Walther, Manfredo Quadroni, Sébastien Catherinet, Jennifer Burgess, Catherine G. Zimmermann-Ivol, Jean-Charles Sanchez, Pierre-Alain Binz, Denis F. Hochstrasser, and Ron D. Appel. Msight: An image analysis software for liquid chromatography-mass spectrometry. *Proteomics*, 5(9):2381–2384, 2005.

[51] Wenjie Pei, Tadas Baltrusaitis, David M.J. Tax, and Louis-Philippe Morency. Temporal attention-gated model for robust sequence classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6730–6739, 2017.

[52] David N. Perkins, Darryl J.C. Pappin, David M. Creasy, and John S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *ELECTROPHORESIS: An International Journal*, 20(18):3551–3567, 1999.

[53] Nikita Prianichnikov, Heiner Koch, Scarlet Koch, Markus Lubeck, Raphael Heilig, Sven Brehmer, Roman Fischer, and Jürgen Cox. Maxquant software for ion mobility enhanced shotgun proteomics. *Molecular & Cellular Proteomics*, 19(6):1058–1069, 2020.

[54] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[55] Rui Qiao, Ngoc H. Tran, Lei Xin, Xin Chen, Ming Li, Baozhen Shan, and Ali Ghodsi. Computationally instrument-resolution-independent de novo peptide sequencing for high-resolution devices. *Nature Machine Intelligence*, 3(5):420–425, 2021.

[56] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics.

[57] Hannes L. Röst, Timo Sachsenberg, Stephan Aiche, Chris Bielow, Hendrik Weisser, Fabian Aicheler, Sandro Andreotti, Hans-Christian Ehrlich, Petra Gutenbrunner, Erhan Kenar, et al. Openms: a flexible open-source software platform for mass spectrometry data analysis. *Nature Methods*, 13(9):741, 2016.

[58] Hannes L. Röst, Uwe Schmitt, Ruedi Aebersold, and Lars Malmström. pyopenms: a python-based interface to the openms mass-spectrometry algorithm library. *Proteomics*, 14(1):74–77, 2014.

[59] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[60] Marianne Sandin, Ashfaq Ali, Karin Hansson, Olle Månsson, Erik Andreasson, Svante Resjö, and Fredrik Levander. An adaptive alignment algorithm for quality-controlled label-free lc-ms. *Molecular & Cellular Proteomics*, 12(5):1407–1420, 2013.

[61] Castrense Savojardo, Pier L. Martelli, Piero Fariselli, and Rita Casadio. Deepsig: deep learning improves signal peptide detection in proteins. *Bioinformatics*, 34(10):1690–1696, 2017.

[62] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W.R. Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, pages 1–5, 2020.

[63] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[64] Hanno Steen and Matthias Mann. The abc's (and xyz's) of peptide sequencing. *Nature reviews. Molecular Cell Biology*, 5(9):699, 2004.

[65] Marc Sturm, Andreas Bertsch, Clemens Gröpl, Andreas Hildebrandt, Rene Hussong, Eva Lange, Nico Pfeifer, Ole Schulz-Trieglaff, Alexandra Zerck, Knut Reinert, et al. Openms–an open-source software framework for mass spectrometry. *BMC Bioinformatics*, 9(1):163, 2008.

[66] Ralf Tautenhahn, Christoph Boettcher, and Steffen Neumann. Highly sensitive feature detection for high resolution lc/ms. *BMC Bioinformatics*, 9(1):504, 2008.

[67] Johan Teleman, Aakash Chawade, Marianne Sandin, Fredrik Levander, and Johan Malmstrom. Dinosaur: a refined open-source peptide ms feature detector. *Journal of Proteome Research*, 15(7):2143–2151, 2016.

[68] Johan Teleman, Christofer Karlsson, Sofia Waldemarson, Karin Hansson, Peter James, Johan Malmstrom, and Fredrik Levander. Automated selected reaction monitoring software for accurate label-free protein quantification. *Journal of Proteome Research*, 11(7):3766–3773, 2012.

[69] Erik Tengstrand, Johan Lindberg, and K M. Åberg. Tracmass: A modular suite of tools for processing chromatography-full scan mass spectrometry data. *Analytical Chemistry*, 86(7):3435–3442, 2014.

[70] Wenmin Tian, Nan Zhang, Ronghua Jin, Yingmei Feng, Siyuan Wang, Shuaixin Gao, Ruqin Gao, Guizhen Wu, Di Tian, Wenjie Tan, et al. Immune suppression in the early stage of covid-19 disease. *Nature Communications*, 11(1):1–8, 2020.

[71] Ngoc H. Tran, Rui Qiao, Lei Xin, Xin Chen, Chuyi Liu, Xianglilan Zhang, Baozhen Shan, Ali Ghodsi, and Ming Li. Deep learning enables de novo peptide sequencing from data-independent-acquisition mass spectrometry. *Nature Methods*, 16(1):63–66, 2019.

[72] Ngoc H. Tran, Rui Qiao, Lei Xin, Xin Chen, Baozhen Shan, and Ming Li. Personalized deep learning of individual immunopeptidomes to identify neoantigens for cancer vaccines. *Nature Machine Intelligence*, 2(12):764–771, 2020.

[73] Ngoc H. Tran, M Z. Rahman, Lin He, Lei Xin, Baozhen Shan, and Ming Li. Complete de novo assembly of monoclonal antibody sequences. *Scientific Reports*, 6, 2016.

[74] Ngoc H. Tran, Xianglilan Zhang, Lei Xin, Baozhen Shan, and Ming Li. De novo peptide sequencing by deep learning. *Proceedings of the National Academy of Sciences*, 114(31):8247–8252, 2017.

[75] Guido V. Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[76] Lei Wang, Adam J. McShane, Mary J. Castillo, and Xudong Yao. Quantitative proteomics in development of disease protein biomarkers. In *Proteomic and Metabolomic Approaches to Biomarker Discovery*, pages 261–288. Elsevier, 2020.

[77] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019.

[78] Xiu-Shen Wei, Chen-Wei Xie, Jianxin Wu, and Chunhua Shen. Mask-cnn: Localizing parts and selecting descriptors for fine-grained bird species categorization. *Pattern Recognition*, 76:704–714, 2018.

[79] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[80] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[81] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 978–987. ACM, 2016.

[82] Xiangyuan Zeng and Bin Ma. Mstracer: A machine learning software tool for peptide feature detection from liquid chromatography–mass spectrometry data. *Journal of Proteome Research*, 20(7):3455–3462, 2021.

[83] Bo Zhang, Mohammad Pirmoradian, Alexey Chernobrovkin, and Roman A. Zubarev. Demix workflow for efficient identification of cofragmented peptides in high resolution data-dependent tandem mass spectrometry. *Molecular & Cellular Proteomics*, 13(11):3211–3223, 2014.

[84] Jing Zhang, Lei Xin, Baozhen Shan, Weiwu Chen, Mingjie Xie, Denis Yuen, Weiming Zhang, Zefeng Zhang, Gilles A Lajoie, and Bin Ma. Peaks db: de novo sequencing assisted database search for sensitive and accurate peptide identification. *Molecular & Cellular Proteomics*, 11(4):M111–010587, 2012.

[85] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021.

[86] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[87] Fatema T. Zohora, M Z. Rahman, Ngoc H. Tran, Lei Xin, Baozhen Shan, and Ming Li. Deepiso: A deep learning model for peptide feature detection from lc-ms map. *Scientific Reports*, 9(1):1–13, 2019.

[88] Fatema T. Zohora, M Z. Rahman, Ngoc H. Tran, Lei Xin, Baozhen Shan, and Ming Li. Deep neural network for detecting arbitrary precision peptide features through attention based segmentation. *Scientific Reports*, 11(1):1–16, 2021.

[89] Fatema T. Zohora, Ngoc H. Tran, Xianglilan Zhang, Lei Xin, Baozhen Shan, and Ming Li. Deepiso: a deep learning model for peptide feature detection. *arXiv preprint arXiv:1801.01539*, 2017.

APPENDICES

Appendix A

Supplementary Notes

A.1 Scanning Window Dimension

The hight of the block is chosen to be 15 pixels since that seems enough to discover the bell shaped intensity of the isotopes. On the other hand, the width of block is considered 211 pixels = 2.11 m/z, because this is sufficient to detect the equidistant property of all the charges. The isotopes in features with z = 1 are 1.0 m/z (100 pixels) apart from each other. Usually peptide features have more than two isotopes. To look over three consecutive isotopes of a feature having charge z = 1, window width of 211 pixels is enough. In all other charges, the isotopes are closer to each other. Therefore, this block size let the CNN look over sufficient area of peptide features to take decision about it's existence and charge.

A.2 Cross-Validation Technique

Our model is trained and evaluated through k-fold cross-validation. That is, we divide the LC-MS maps in the dataset into k groups or k folds. Then we run k round of experiments, where we keep one group for testing, and train on the other $k - 1$ groups. In order to save the best model state during training, we keep one LC-MS map from the training set as a validation set. Then we use that model state for testing. Whenever we say train or validation on a LC-MS map, we mean training/validation on the features cut from that LC-MS map (explained later under the subsections regarding training data generation for IsoDetecting and IsoGrouping module). However, when we say test on a LC-MS map, we actually mean scanning the full LC-MS map as shown in the block diagram of Figure 1 in

main text. That is, during testing (or the real application phase) we will actually scan the whole LC-MS map in a bottom up, left to right fashion (in other words, column by column). Now, we discuss how the LC-MS maps are divided into different folds for cross-validation as follows:

- In the 3D dataset (downloaded from ProteomeXchange with accession number PXD001091), there are 12 dilution samples for the LC-MS analysis, each sample having a different concentration of spike proteins. Each sample goes through the physical LC-MS instrument and produce a LC-MS map. Usually each experiment is repeated multiple times, and each resultant LC-MS map is called a replicate. Dilution sample 1 and sample 5 to 12 have 4 replicates each (e.g., 130124_dilA_1_01, 130124_dilA_1_02, 130124_dilA_1_03, 130124_dilA_1_04 are the four replicates for sample 1. These names are also mentioned in the final result shown in Supplementary Table S3). Dilution sample 2, 3, and 4 have 7 replicates each. Therefore, we have 57 LC-MS maps in total. For this dataset, we consider two settings: $k = 2$ and $k = 6$.

    1. For $k = 2$: We divide these samples or LC-MS maps into 2 groups, X and Y. Group X contains the LC-MS maps from sample 9, 10, 11, 12. Group Y contains the LC-MS maps from sample 1 to 8. We first train the model on Group X (i.e., sample 10,11,12 are used for training and sample 9 is used as a validation set to save the best state of the model) and test the trained model on Group Y. Then we do the opposite. We train the model on Group Y (sample 5, 6, 7, 8 are used for training and sample 1 is used for validation) and test the trained model on Group X. The result of testing on all the samples is provided in Supplementary Table. Please note that, we report the result separately for each replicate. Detection percentage for a sample is the average detection percentage for all the replicates of that sample, which is already reported in the main body of the thesis.
    2. For $K = 6$: Here we just divide 12 samples into 6 groups, each having 2 samples. Then we treat one group as testing, and use the rest for training. This result is also included in the main body of the thesis and also in Supplementary Table.

- Next, we discuss the distribution of LC-MS maps in our 4D dataset (downloaded from ProteomeXchange with accession number PXD010012) into different folds. For the 4D dataset, we have samples from two mobile phases: A and B. A provides 12 LC-MS maps (denoted as 2042 to 2053) and B provides 4 LC-MS maps (2054 to 2057). In total we have 16 LC-MS maps. Again, we consider two settings: $k = 2$ and $k = 8$.

1. For $k = 2$: We just divide it into two groups having equal size. Group X has 8 LC-MS maps: 2042 to 2049. Group Y has the remaining 8 LC-MS maps: 2050 to 2057. Just like before, we train on Group X (2044 is used for validation and the rest for training) and test on Group Y. Then we train on Group Y (2054 is used for validation and the rest for training) and test on Group X. The test result for each LC-MS map is provided in the main text.
2. For $k = 8$: We divide 16 samples into 8 groups, each having 2 samples. Then we repeat the experiments as usual. The results are included in the main text.

A.3 Class Weight Assignment Procedure

Please refer to Table A.1 for comparison among different candidate weighting mechanisms. We choose the class weight based on distribution per sample. Let us have a sample window which contains 100 datapoints. They come from three different classes (z): 0, 2, and 3. We have 10 points from $z = 3$, 30 points from $z = 2$, and remaining 60 points from $z = 0$. Then points from class 0, 2, and 3 get weights of 0.4, 0.7, and 0.9 respectively. We perform this for every sample. So every training sample has a weight list associated with it, which we pass to the network during cross entropy loss calculation.

A.4 Candidate solutions for boundary region point segmentation in PointIso

We would like to discuss our experiments to correctly segment partially covered peptide features in a target window. We started with considering a bigger scanning window which will produce output for the center region. This caused three problems. First, we can apply back-propagation based on full bigger window during training time (number of input nodes equals to the number of output nodes), but use the center region only during testing time or prediction (output nodes corresponding to center region are used only). In this case model gets confused about the boundary region during training and cannot learn well. Second, during training time we can apply back-propagation based on center region only, therefore, number of input nodes is not equal to the number of output nodes anymore. This actually makes the architecture very complex. Because number of datapoints in the center region, and surrounding r1, r2, r3, r4 regions is never fixed. Although we consider a fixed number and do padding for simplification, but when number of input is not equal to the number

Experiment Category	z=0	z=1	z=2	z=3	z=4
Without any class weight (after 10 epochs)	96%	40%	53%	33%	15%
Class weight based on distribution over whole dataset (after 10 epochs)	91%	48%	68%	51%	27%
Class weight based on distribution over sample (after 10 epochs)	86%	51%	71%	55%	30%
Same as above, after convergence	64%	59%	79%	67%	14%
Same as above, with attention mechanism	50%	79%	95%	91%	85%
Same as above, with attention mechanism and higher resolution	40%	99%	98%	98%	98%

Table A.1: Class sensitivity for different weighting mechanisms. We compare the candidate weighting mechanisms based on the sensitivity for the best case scenario, i.e., when feature is aligned with the left boundary of the scanning window (e.g., feature A in Figure 4(a)), with high abundant features, i.e., features having charge, z = 1, 2, 3, 4. Please note that, although the sensitivity of negative class (z=0) is comparatively lower for our chosen criteria, however, it does not imply that it reports many false positives. Although the datapoints which are very close or adjacent to the real signal, are sometimes predicted as positive points, but in general the negative class has higher class sensitivity than all others as presented in the main manuscript.

of outputs, internal design gets quite complicated. Also the fact that, datapoints in r1, r2, r3, r4 need not to worry about each other, only have to focus on center region, may not be well learned by the model. We would not have any control over that. Finally, we also face technical issues since GPU memory exceeded (more than 16 GB) with bigger region. Because of these reasons we were unable to proceed with the design.

So we applied next simpler technique, sliding windows with 50% overlapping. The resultant class sensitivities are presented in the first row of Table SA.2. After passing the IsoDetecting output through IsoGrouping module, it finally produce only 65% feature detection as presented in the first row of Table 5. Therefore we had to use more sophisticated approach to address this problem.

In Figure 4 of main manuscript, we see that the regions r_1, r_2, r_3, and r_4 are actually playing the key role in detecting the traces inside target window. We empirically tested following three techniques for diffusing the surrounding information into current window, and the results are summarized in Table SA.2.

Experiment Category	z=0	z=1	z=2	z=3	z=4
Sliding window with 50% overlapping	86%	50%	72%	57%	36%
Skiplink inserted in above model	85%	56%	76%	66%	41%
Bi-directional 2D RNN	85%	51%	77%	67%	49%
Attention mechanism	84%	62%	85%	71%	50%
Attention mechanism with higher resolution	90%	64%	85%	81%	61%

Table A.2: Different techniques of absorbing surrounding information and corresponding class sensitivity of IsoDetecting module. We define the class sensitivity of a scanning window as the number of datapoints from class z (0 to 9) detected correctly out of total number of datapoints in a scanning window. To evaluate candidate solutions we use the class sensitivity of high abundant features (charge z = 1,2,3, and 4) in a *average case* scenario. Average case means the scanning window might contain any number of features, they may appear at any location of the window, they might be partially or fully seen, and might be overlapping as well. We see that the DANet inspired attention based mechanism works better than other techniques.

- First, we just calculated the global features of surrounding regions and diffuse them together by addition and concatenation with the global features of target window. Then we repeated the 50% overlapping technique, which did not bring any significant change. Using some skip links along with that brings little improvement as shown in the second row of the table.
- Then we used a bi-directional two dimensional RNN network to flow information from all direction into the target window. The corresponding class sensitivities are provided in the third row.
- Finally, we applied the attention mechanism proposed by DANet which works better than first two techniques, as reported in the fourth and fifth row. So we choose PointNet segmentation network combined with DANet to develop our IsoDetecting module.

Class (z)	0	1	2	3	4	5
Old architecture (only CNN with fixed interval along RT axis)	85.04%	82.08%	90.96%	86.53%	83.87%	62.18%
Upgraded architecture (both CNN and RNN, with MS-Scans given intervals along RT axis)	96.43%	93.80%	96.98%	98.74%	97.94%	85.86%

Table A.3: Better learning (higher class sensitivity) by IsoDetecting module with upgraded architecture.

Appendix B

Supplementary Methods

B.1 Augmented Data Generation for 'IsoDetect' Module

For charge states 6 to 9, we did not have enough samples for training. Therefore we applied data augmentation. Please note that the 10 ppm tolerance along m/z axis let us cut the features couple of pixels before the exact start, as mentioned above. This number of pixels can vary from 0 to 2 based on the actual m/z. Since the LC-MS maps in our dataset span from 400 m/z to 2000 m/z (approximately), therefore for each sample having these charge states we can **cut multiple sequences** within the tolerance limit. For example, if a peptide feature with charge state 6 lies around 2000 m/z area, then tolerance limit is up to 2 pixels. So we cut sequences starting at exact m/z, 1 pixel (or 0.01 m/z) before, and 2 pixels (0.02 m/z) before. So we get three sequences for it. In this way we generate augmented samples.

B.2 Attention Calculation Flowchart for PointIso

Please refer to the flowchart shown in Figure B.1. We have the point features of left region, $Point\_Feature_{left}$ according to the PointNet architecture shown in the main manuscript. Then we multiply $Point\_Feature_{target}$ with the transpose of $Point\_Feature_{left}$ according to the rule of matrix multiplication. Then we take softmax of the product as Equation 1, which gives us a $[N_T \times N_L]$ matrix, $Left\_Impact\_on\_Target$, where, N_T and N_L are the

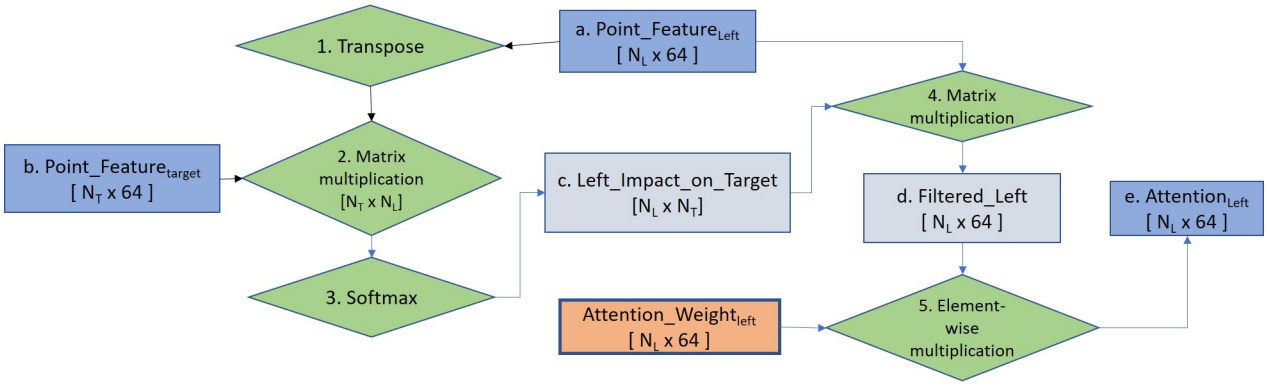


Figure B.1: Flowchart of attention calculation in IsoDetecting module. This particular flowchart is intended to find out the attention or impact of left region over the datapoints of target window. Exactly similar approach is followed for other surrounding regions as well and finally all are diffused with the $Point\_Feature_{target}$ by addition.

total number of datapoints of the target window and left region respectively. So each row presents a datapoint from target window and columns present the datapoints from left region.

$$Left\_Impact\_on\_Target = \frac{exp((Point\_Feature_{target_i}).(Point\_Feature_{left_j}))}{\sum_{i=1 \text{ to } N_L} exp((Point\_Feature_{target_i}).(Point\_Feature_{left_j}))} \tag{B.1}$$

Therefore, $Left\_Impact\_on_{Target_{(j,i)}}$ presents the attention of i^{th} point of left region over the j^{th} point of target window. The higher the value, the higher the correlation (similar feature) between those two points. So the j^{th} row tells us which datapoints from left region have higher attention or highly correlated with the j^{th} datapoint of target window. Next, we want to fetch those *significant point features from left region. So we apply another round of matrix multiplication (4^{th} operation) between $Left\_Impact\_on\_Target$ and $Point\_Feature_{Left}$. We denote the resultant product as $Filtered\_Left$ since it essentially gives us $Point\_Features_{left}$ but scaled/filtered according to the aforementioned correlation or attention. Then again we have to know how much of those filtered features should be incorporated with the $Point\_Feature_{target}$ while segmenting the datapoints of target window. So we use a weight matrices, namely $Attention\_Weight_{left}$, and multiply it with $Filtered\_Left$, producing $Attention_{left}$, which is finally passed forward to be diffused (by addition) with the $Point\_Feature_{target}$. This $Attention\_Weight_{left}$ is learned through training.

B.3 Resolution Degradation in IsoGrouping Module

IsoDetecting module generates sequences of potential isotopes which are sent to the IsoGrouping module for final detection of peptide features. Now, the m/z value of isotopic signals are real numbers having up to 4 decimal places. We degrade each signal into real numbers having up to 2 decimal places. For example, let us have two sequences of isotopes A, and B, where the first isotope of the sequences are denoted as A_1 and B_1 respectively. The m/z values of these are respectively $A_{1\_mz} = 500.2351$ and $B_{1\_mz} = 500.2443$. During resolution degradation we filter out the signal instensity from the background with $+-2$ ppm m/z range. The range is calculated as: $\frac{A_{1\_mz} \times 2.0}{10^6}$. So that we have $A_{1\_mz} = 500.24$ and intensity is set as the maximum of the intensities of datapoints within range 500.2341 to 500.2361. Similarly, $B_{1\_mz} = 500.24$, *but* intensity is set as the maximum of the intensities of datapoints within range 500.2433 to 500.2453. So we see that, although both have the same m/z values in lower resolution, they definitely belong to different sequences and have different intensities (thus might have different pattern as well).

B.4 Merge Secondary Peaks in PointIso

If *ftr* and *ftr_pred* are two peptide features then they are merged if following two conditions are met:

- They lie within the 0.005 m/z and 0.1 min RT tolerance from each other, and have the same charge.
- For both of the features, after we sort the *index* of Softmax output layer (found from IsoGrouping module) in the ascending order, their orders match with each other.

After this merging, we discard all the features having feature score (maximum value of the Softmax output layer) less than 0.3. It gives us about 100,000 features.

After that, for further reduction in total number of features, we discard following features having low probability of being a true feature:

- Only first isotope is detected nicely and has score < 0.50
- The feature has charge 1 and score < 0.80

After that, we have about 48,000 peptide features in total from each LC-MS map.

Appendix C

Supplementary Tables

C.1 Comparative Analysis of 3D Peptide Feature Detection Tools

Supplementary Table C1: Percentage of identified peptide features by different algorithms

LC-MS maps	OpenMS	MaxQuant	Dinosaurs	Peaks	**PointIso (k=2)**	**PointIso (k=6)**	DeepIso
130124_dilA_1_01	94.2266	94.6575	95.4761	95.6053	**98.0181**	**98.3525**	95.8206
130124_dilA_1_02	96.0611	94.8955	95.4582	95.8601	**98.8344**	**98.7606**	95.7123
130124_dilA_1_03	95.9013	95.3994	95.8176	96.0686	**98.2016**	**98.1563**	96.3864
130124_dilA_1_04	95.444	95.1351	95.444	95.5598	**98.1467**	**97.8753**	95.3393
	95.40825	95.021875	95.548975	95.7735	**98.3002**	**98.2862**	95.81465
130124_dilA_2_01	96.3771	95.2163	96.2012	95.7088	**98.593**	**98.5643**	95.9088
130124_dilA_2_02	95.6753	95.5335	96.1361	95.6044	**98.3339**	**98.3094**	96.0128
130124_dilA_2_03	96.107	94.9221	95.9039	95.7346	**99.0521**	**98.7677**	95.8496
130124_dilA_2_04	95.8291	94.9135	95.6256	95.242	**98.4401**	**98.1291**	95.685
130124_dilA_2_05	96.0703	95.0362	95.6222	95.5188	**98.1041**	**98.2706**	95.4794
130124_dilA_2_06	95.7873	94.6133	95.8564	95.442	**98.308**	**98.2689**	96.0538
130124_dilA_2_07	96.2179	95.7668	95.975	95.8362	**98.3692**	**98.0728**	95.83156667
	96.00914286	95.1431	95.90291429	95.58382857	**98.4572**	**98.3404**	96.435
130124_dilA_3_01	95.9933	95.3872	96.2963	95.9933	**98.3838**	**98.279**	96.4134
130124_dilA_3_02	96.3699	95.6849	96.4041	96.3014	**98.6301**	**98.6942**	96.7434
130124_dilA_3_03	96.0927	95.596	96.8212	96.6225	**98.543**	**98.5667**	96.1336
130124_dilA_3_04	96.1678	95.0446	95.9035	95.9035	**99.0089**	**98.9169**	96.0115
130124_dilA_3_05	95.9974	95.7392	96.417	96.3525	**98.5474**	**98.307**	96.4881
130124_dilA_3_06	96.5644	95.497	96.5977	96.0974	**98.0654**	**98.0678**	96.2172
130124_dilA_3_07	96.3929	95.5556	95.942	96.1997	**98.0998**	**98.1818**	96.34888571
	96.22548571	95.50064286	96.34025714	96.21	**98.46834286**	**98.4305**	96.6091
130124_dilA_4_01	96.3514	95.6784	96.3868	96.3868	**98.1226**	**98.5243**	96.4585
130124_dilA_4_02	95.8527	95.3522	95.6382	96.2817	**97.4973**	**97.3616**	96.7043
130124_dilA_4_03	96.286	96.286	96.7025	96.772	**98.2645**	**98.2748**	96.2997
130124_dilA_4_04	96.011	95.4952	96.4237	96.2173	**98.7276**	**98.7273**	96.0813
130124_dilA_4_05	96.3086	95.5437	96.1756	96.0758	**98.9691**	**98.7331**	96.486

130124_dilA_4_06	96.4737	95.5755	96.6401	96.3407	**98.4365**	**98.2733**	96.6774
130124_dilA_4_07	97.3072	96.0771	97.0412	96.5426	**98.1383**	**98.3539**	96.47375714
	96.37008571	95.71544286	96.42972857	96.3738	**98.30798571**	**98.3212**	96.618
130124_dilA_5_01	95.6699	95.6699	96.151	96.225	**97.9645**	**97.9457**	96.4579
130124_dilA_5_02	95.747	95.8899	96.2831	96.1401	**98.0343**	**98.1353**	96.0579
130124_dilA_5_03	96.3406	95.3488	96.3748	96.4774	**97.777**	**97.7231**	96.0755
130124_dilA_5_04	96.1288	94.9298	96.026	95.9918	**98.9723**	**99.1018**	96.302325
	95.971575	95.4596	96.208725	96.2086	**98.187025**	**98.2265**	96.6903
130124_dilA_6_01	96.4139	96.1524	96.6007	96.526	**97.5719**	**97.6038**	96.6123
130124_dilA_6_02	96.0598	95.3667	96.1328	96.1693	**98.5042**	**98.5313**	96.0113
130124_dilA_6_03	96.3015	95.28	95.8084	95.7027	**98.3093**	**98.2115**	95.7466
130124_dilA_6_04	95.321	95.2186	95.8333	95.4918	**97.4044**	**97.497**	96.265125
	96.02405	95.504425	96.0938	95.9724	**97.94745**	**97.9609**	95.8438
130124_dilA_7_01	95.4466	94.7167	95.4814	95.1338	**98.1578**	**97.8763**	96.4602
130124_dilA_7_02	96.1326	95.9254	96.6851	95.9945	**98.9986**	**98.8415**	96.1854
130124_dilA_7_03	95.9586	95.0568	96.0254	95.992	**98.664**	**98.6804**	96.0363
130124_dilA_7_04	95.7404	94.929	95.9094	95.6051	**97.7688**	**98.0653**	96.131425
	95.81955	95.156975	96.025325	95.6814	**98.3973**	**98.3659**	96.025
130124_dilA_8_01	95.9857	95.4122	96.0573	95.6631	**97.957**	**98.0613**	95.9962
130124_dilA_8_02	95.8304	94.7703	95.4417	95.689	**97.7739**	**98.2217**	96.2512
130124_dilA_8_03	95.9361	95.3803	96.3876	96.1445	**98.2633**	**98.0943**	96.4653
130124_dilA_8_04	95.7613	95.3204	96.0665	95.9986	**98.2028**	**98.3691**	96.184425
	95.878375	95.2208	95.988275	95.8738	**98.04925**	**98.1866**	95.8088
130124_dilA_9_01	95.604	94.8282	95.9365	95.4932	**98.7071**	**98.7238**	96.6721
130124_dilA_9_02	96.3676	95.5685	96.6945	96.5492	**98.1475**	**98.0111**	95.5114
130124_dilA_9_03	95.5315	95.1712	96.036	95.6396	**97.9459**	**98.0467**	96.2029
130124_dilA_9_04	96.2108	95.453	96.3142	95.8663	**98.5188**	**98.4768**	96.0488
	95.928475	95.255225	96.2453	95.8871	**98.329825**	**98.3146**	95.8997
130124_dilA_10_01	95.3586	95.2436	96.0491	95.8189	**98.3122**	**98.2379**	96.3651
130124_dilA_10_02	94.8629	95.3418	96.5607	96.3431	**96.8219**	**97.1553**	96.1563

130124_dilA_10_03	95.7868	95.1026	95.7148	95.9309	**97.8394**	**97.8873**	96.2041
130124_dilA_10_04	96.4209	95.4903	96.3135	96.4209	**98.5326**	**98.4056**	96.1563
	95.6073	95.294575	96.159525	96.1284	**97.876525**	**97.9215**	95.6797
130124_dilA_11_01	96.0899	95.3411	95.9651	95.7155	**98.4193**	**98.2508**	96.0016
130124_dilA_11_02	95.2318	95.2759	96.0706	95.4525	**95.9382**	**96.0412**	95.5539
130124_dilA_11_03	95.5888	95.1311	95.4224	95.5472	**97.0037**	**97.2303**	96.0157
130124_dilA_11_04	95.5084	94.7325	95.0592	95.3042	**98.1625**	**98.2924**	95.812725
	95.604725	95.12015	95.629325	95.5049	**97.380925**	**97.4537**	95.1277
130124_dilA_12_01	95.0474	94.3625	95.2582	94.8894	**95.8377**	**95.9792**	95.6958
130124_dilA_12_02	95.7128	94.7481	95.7128	94.8553	**95.9271**	**95.8393**	95.5627
130124_dilA_12_03	95.7011	94.5752	95.6499	95.5476	**96.4176**	**96.6148**	95.6148
130124_dilA_12_04	95.3646	94.4271	95.4167	95	**97.3958**	**97.5136**	95.50025
	95.456475	94.528225	95.5094	95.0731	**96.39455**	**96.4867**	
1	95.40825	95.021875	95.548975	95.7735	**98.3002**	**98.2862**	95.81465
2	96.00914286	95.1431	95.90291429	95.58382857	**98.4572**	**98.3404**	95.83156667
3	96.22548571	95.50064286	96.34025714	96.21	**98.46834286**	**98.4305**	96.34888571
4	96.37008571	95.71544286	96.42972857	96.3738	**98.30798571**	**98.3212**	96.47375714
5	95.971575	95.4596	96.208725	96.2086	**98.187025**	**98.2265**	96.302325
6	96.02405	95.504425	96.0938	95.9724	**97.94745**	**97.9609**	96.265125
7	95.81955	95.156975	96.025325	95.6814	**98.3973**	**98.3659**	96.131425
8	95.878375	95.2208	95.988275	95.8738	**98.04925**	**98.1866**	96.184425
9	95.928475	95.255225	96.2453	95.8871	**98.329825**	**98.3146**	96.0488
10	95.6073	95.294575	96.159525	96.1284	**97.876525**	**97.9215**	96.1563
11	95.604725	95.12015	95.629325	95.5049	**97.380925**	**97.4537**	95.812725
12	95.456475	94.528225	95.5094	95.0731	**96.39455**	**96.4867**	95.50025
average	95.85862411	95.24341964	96.00679583	95.85590238	**98.00804821**	**98.02455833**	96.07251954

z matched	OpenMS	MaxQuant	Dinosaur	PEAKS	PointIso, k=2	PointIso, k=6	DeepIso
1	95.0087	94.7927	95.3102	95.5823	96.7628	97.1064	95.6572
2	95.4936	94.8916	95.467	95.4037	97.0528	97.3897	94.0906
3	95.8882	95.3	96.0355	96.1029	97.1074	97.5097	95.9061
4	96.0141	95.5145	96.1481	96.2651	97.3909	97.6808	96.4644
5	95.5536	95.2989	95.9081	95.9649	97.0397	97.3741	96.3772
6	95.7341	95.3373	95.6742	95.8477	96.9667	97.2028	96.3073
7	95.4077	95.0207	95.5941	95.5405	97.2987	97.5613	95.9608
8	95.4817	95.0427	95.7046	95.8003	97.0583	97.4518	96.1395
9	95.5216	95.0946	95.8272	95.6348	97.1059	97.4343	96.0668
10	95.2142	95.2608	95.8088	95.9075	96.4943	96.7874	96.1326
11	95.379	94.9199	95.3995	95.3646	95.9769	96.1767	95.4982
12	95.2336	94.4008	95.1084	94.9767	94.7185	95.1462	94.5665
average	95.494175	95.072875	95.665475	95.69925	96.74774167	97.06843333	95.76393333

C.2 Comparative Analysis of 4D Peptide Feature Detection Tools

Supplementary Table C2: Percentage of identified 4D peptide features by different algorithms

	Samples	PointIso, k=2	PointIso, k=8	PEAKS	MaxQuant
	1	86.79	86.08	81.01	79.86
	2	85.5	85.66	82.5	80.82
	3	85.14	85.72	81.78	81.37
	4	86.48	86.2	81.42	81.11
	5	87.74	87.19	82.65	79.78
	6	85.93	85.83	83.02	81.78
	7	88.04	88.31	82.5	80.89
m/z,rt,k0	8	86.09	85.57	83.28	81.33
	9	85.92	85.32	82.49	81.46
	10	86.32	86.56	81.44	79.79
	11	86.32	87.49	81.95	80.81
	12	87.6	87.6	81.37	81.11
	13	85.99	86.17	82.1	80.82
	14	85.52	85.59	82.02	80.22
	15	85.93	85.98	82.81	81.26
	16	85.7	85.02	82.36	80.91
	avg	86.313125	86.268125	82.16875	80.8325

	Samples	PointIso, k=2	PointIso, k=8	PEAKS	MaxQuant
	1	84.407	84.6404	80.0908	78.4268
	2	84.7287	84.5967	81.391	78.9261
	3	84.0981	84.1734	80.8468	79.3347
	4	84.646	84.9344	80.391	79.193
	5	86.1258	86.0649	81.6702	78.2146
	6	84.873	84.8053	81.9201	79.9546
	7	86.8624	87.1737	81.2789	78.9684
m/z,rt,k0,z	8	84.4464	84.2944	82.3383	79.5115
	9	84.0422	84.0371	81.6051	79.7811

10	84.9463	85.3131	80.3057	78.2679	
11	86.1217	86.1136	80.7935	78.8484	
12	86.2066	86.3396	80.224	79.4515	
13	84.3422	84.8143	80.9947	78.8329	
14	84.2136	84.5094	80.9989	78.3131	
15	83.6216	83.8755	81.6278	79.3446	
16	83.3035	83.2277	81.0365	79.1775	
avg	84.81156875	84.93209375	81.09458125	79.03416875	

Appendix D

Supplementary Figures

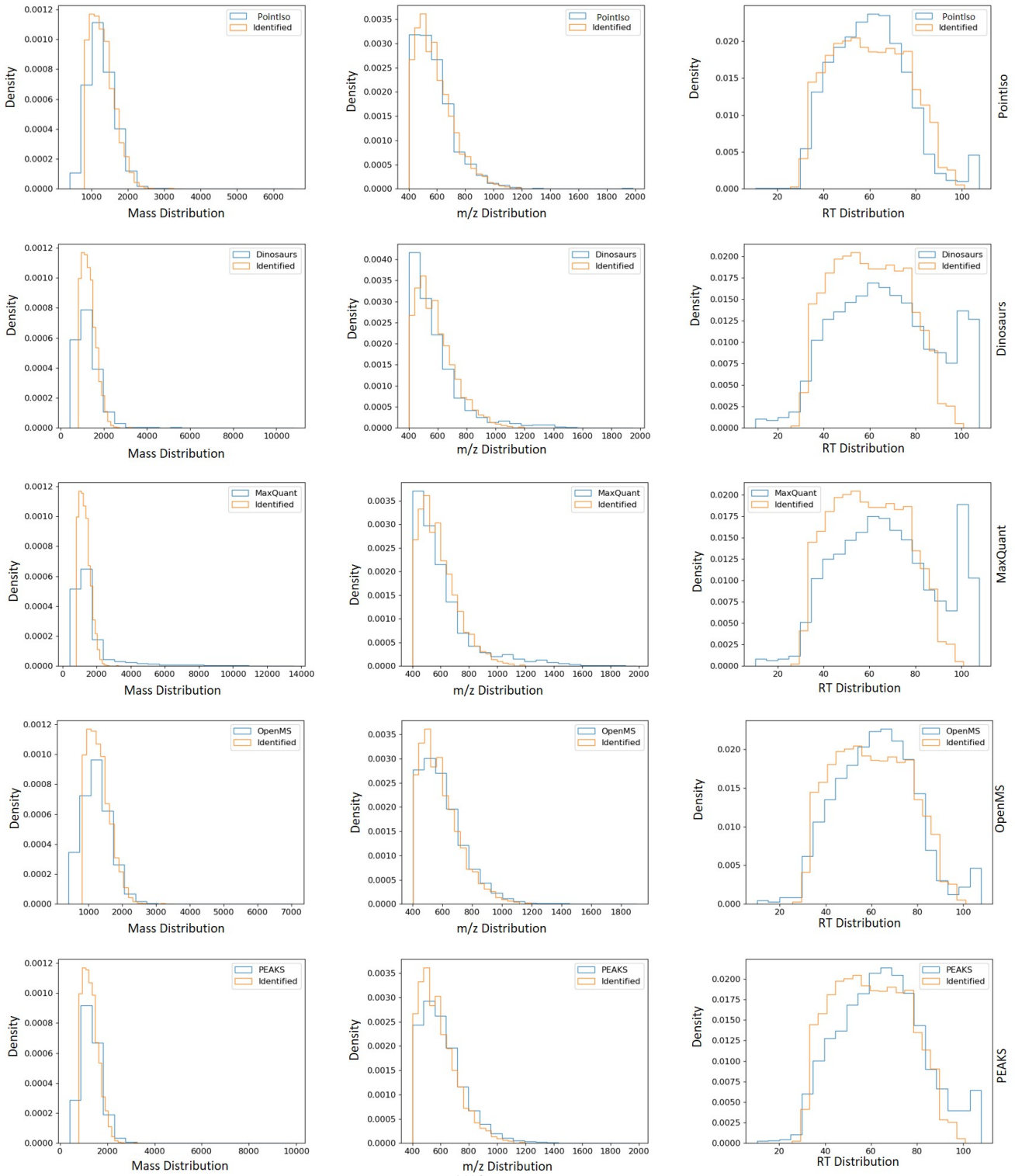


Figure D.1: Comparison of mass, m/z, and RT distribution of detected features (blue) and identified features (orange) for different tools.

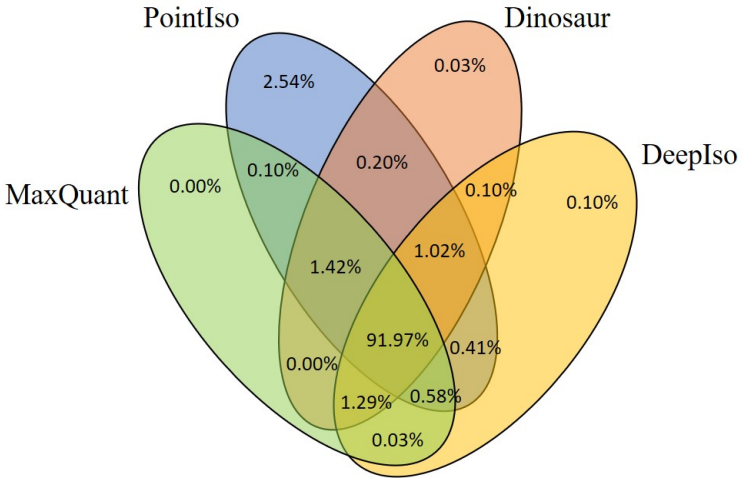


Figure D.2: Venn diagram of identified peptide features detected by four algorithms (PointIso, DeepIso, Dinosaur, and MaxQuant) for replicate 1 of sample 10.