HIGH SCHOOL EDITION
Journal of Student Research

Volume 11 Issue 3 (2022)

Deep Learning for MS2 Feature Detection in Liquid Chromatography-Mass Spectrometry

Jonathan He[1], Olivia Liu[#] and Xuan Guo[#]

[1]Texas Academy of Mathematics and Science, Denton, TX, USA
[#]Advisor

ABSTRACT

Accuracy of peptide identification is crucial for LC-MS analysis to reveal information regarding many different aspects of proteins that aid in the discovery of biomarkers and profiling of complex proteomes. Preprocessing steps such as feature detection are crucial yet challenging; current feature detection tools are not robust enough to detect low-abundance, low-peak fragments of peptides found in MS2 data from tandem mass spectrometry. In this study, we developed a deep learning-based model with an innovative sliding window process that enables high-resolution processing of quantitative MS/MS data to conduct accurate feature detection on MS2 data. Experimental results show that our model is able to produce more accurate values and identifications than existing feature detection tools. Therefore, we believe that our model can realize the full potential of neural networks in the field of bioinformatics and yield long-term benefits in the advancement of proteomic inquiry.

Introduction

Peptides, as the fundamental structures of proteins, are increasingly crucial for aspects of bioinformatics, such as biomarker discovery and drug identification research. Liquid chromatography with tandem mass spectrometry (LC-MS/MS) based proteomics is the most common research field on this subject. Recent advancements in the development of mass spectrometry hardware and the subsequently increased amounts of analytical data make LC-MS maps difficult to interpret manually or through existing commercial feature detection tools. Deep learning neural networks have been identified as an adaptable and compelling structure to tackle the complexity of this new data and hence have become an increasingly popular tool for new LC-MS/MS software. For example, DeepNovo uses sequencing steps and an LSTM network for de novo peptide sequencing [10]. DeepSig uses a DCNN and probabilistic methods to detect signal peptides and locate sequences' cleavage sites [9]. DeepRT combines a CNN and RNN to predict peptide retention times [26]. As seen through the multitudes of recent software, deep learning's outstanding performance on LC-MS/MS analysis is making significant breakthroughs in the field of proteomics and bioinformatics.

Data-independent acquisition (DIA) proteomics is a recently developed mass spectrometry (MS)-based proteomics strategy. In the DIA method [3], enzymatically broken peptides must first be identified and quantified by analysis instruments in order to be further investigated. Through tandem mass spectrometry, an extension of the MS procedure that places two mass spectrometers in tandem to further fragment peptides. This is commonly used for product-ion or precursor-ion scans and high-level analysis of trace components found in complex mixtures. Through the first mass spectrometer one obtains MS1 data for the precursor peptides, and from the second, MS2 data for the peptide fragments or product ions; thus, the peptide fragments found in MS2 data are physically smaller [22]. A feature is a pattern formed by multiple isotopes of a peptide fragment that have distinctly high signals, forming "peaks" in the retention time dimension. Feature detection is especially challenging due to the possibility of overlapped peptide fragments (which create difficulties for object detection methods) and rigid mathematical assumptions for distinguishing between true peaks and noise. Furthermore, due to the aforementioned hardware developments, LC-MS/MS maps can be exceedingly large, making manual quantification nearly impossible and requiring considerable computational power for the efficiency of analysis.

The peptide identification process begins with low-level analysis problems such as feature detection through detection and intensity calculation from the LC-MS map, which can be visualized as an image that plots mass-to-charge ratio (m/z), retention time (RT), and intensity of peaks on a 3D map [5]. Mass to charge ratio is equal to the mass of an ion divided by its charge. The retention time of a peptide is defined as the time spent between the point at which the peptide is injected into the mass spectrometer and the point at which a solute emerges. This endpoint is shown by a peak in the signal created by the peptide. Lastly, the intensity factor indicates the abundance of an ion, i.e., greater repetition of a particular ion results in greater intensity.

Recently, deep learning has been used for feature detection. For instance, DeepIso incorporates both a Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) to predict peptides [2]. However, these existing tools are created and tested primarily for MS1 features. Feature detection at the MS2 level becomes exponentially more challenging due to the weak signals and low abundance of product ions found. Furthermore, the challenge of distinguishing features from noise is augmented by the low intensities of these MS2 peptide fragments combined with the extensive size of the LC-MS/MS map. Our novel deep learning tool targets this issue by using a process of sliding windows to break an LC-MS/MS map into smaller frames, which are used individually to train the model. In our study, the use of Faster-RCNN [1] combined with the optimization of our training dataset allows for the full realization of the potential of deep learning applications for feature detection.

Methods

The workflow of our method is shown in Fig. 1. We rendered LC-MS/MS benchmark data into a 3D map with m/z on the x-axis, retention time on the y-axis, and intensity as a whiteness value. We then developed an automated process of sliding windows that cut through the image in a sequential fashion to generate a series of small subframes from the original image. This step allowed for more efficient training due to the higher resolution of the windows in comparison to the original image as well as the ease of training because the model is able to process smaller frames with higher throughput. We then used the union of MaxQuant [6] and Dinosaur's [7] outputs to annotate the images generated with bounding boxes as the ground truth of the features.

After loading the training images, we incorporated the Faster-RCNN model, which is composed of a Region Proposal Network (RPN), a classifier, and a regressor. By proposing sets of anchor boxes, filtering them through objectiveness score, and updating the weights based on the error in output confidence and bounding box coordinates, the model can identify and localize features accurately. Afterward, we use Soft Non-Maximum Suppression to filter out repeated detections by comparing detection confidence and calculating IoU. To fine-tune the hyperparameters, we take the loss value as an indication of training progress to adjust the number of epochs and the learning rate to achieve a minimal stable loss. Finally, we ran the model on an independent dataset to evaluate its performance in terms of the number of features detected and the bounding box area.

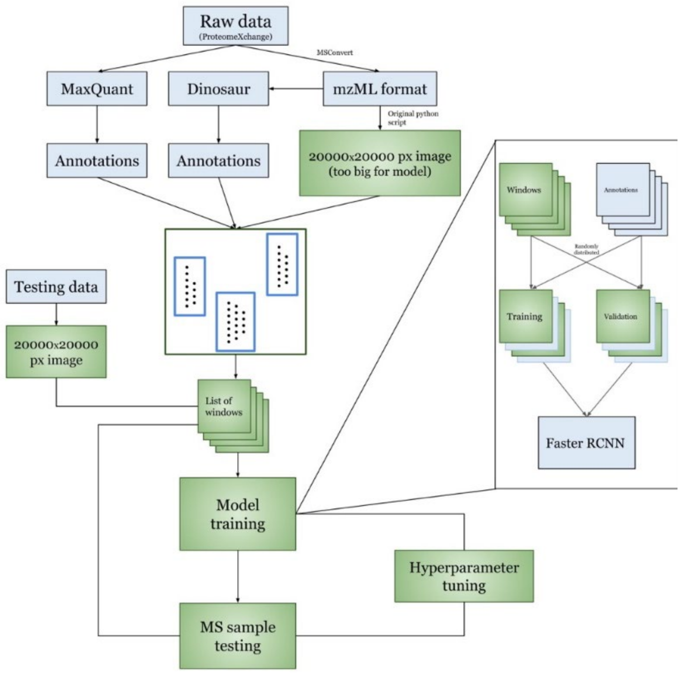

Figure 1. Workflow of our experimental process. The blue denotes existing procedures/tools while the green denotes our original contributions.

Data Acquisition

We extracted data from ProteomeXchange (PXD006096, PXD010357, PXD004684), all of which are either lung, breast, or prostate cancer samples measured in the DIA mode. [13, 17, 18]. In order to prevent overfitting towards a single type of protein and minimize repetition across the training data, we selected data from different sources spanning multiple different projects. We utilized ProteoWizard's [15] tool msConvert [16] to convert raw LC-MS/MS data into mzML format. Raw data was also sent to MaxQuant, while the files after conversion to mzML were fed through Dinosaur. These existing tools provide annotations for start and end retention times, m/z midpoint, and the number of isotopes, the union of which we consider the ground truth of the features. We use the values produced by MaxQuant and Dinosaur to annotate the map with bounding boxes around the features manually.

Image Generation

In tandem mass spectrometry, precursor ions are further broken into product ions before going through the mass detector; thus, these peptide fragments have lower signal intensities and are more difficult to detect. We chose here to generate LC-MS maps as an image for object detection to address this issue. Figure 2 shows this comparison of MS1 and MS2.

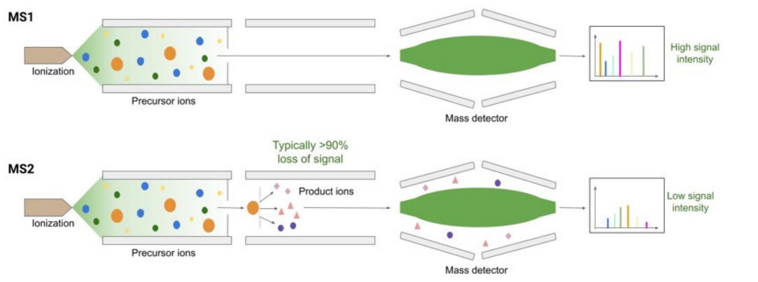

Figure 2. MS2 data differs from MS1 in that MS2 signals are less intense and therefore more difficult to detect.

We wrote an in-house script to generate an LC-MS map and convert it to a grayscale image. Each peak is defined by three dimensions: m/z (mass over charge), retention time (in seconds), and intensity. For each peak, we begin by converting the m/z to integer values scaled along the x-axis, with each integer value representing the index of the pixel along the width of the image. The equation for this process is as follows:

Equation 1. Normalization of m/z values.

$$i_n = \lfloor \frac{(k_n - k_{min})}{k_{max} - k_{min}} \times (w-1) \rfloor$$

Where *w* is the width of the image, *k* is the m/z charge of a given peak, *n* is the desired peak, k_{max} and k_{min} respectively represent the highest and lowest m/z values in the dataset, and i_n is the x-axis index of the peak. Through this process, the minimum normalized m/z value is 0 and the maximum value is (width - 1). The same process is repeated for the retention times to calculate the y-axis indices with respect to the height of the image. Intensity, which is given in the mzML format as a double value, is converted to an integer between 0 and 255 and represents the whiteness of the pixel in a similar process as follows:

Equation 2. Normalization of intensity values.

$$a_n = \lfloor \frac{(k_n - k_{min})}{k_{max} - k_{min}} \times 255 \rfloor$$

Sliding Window Procedure

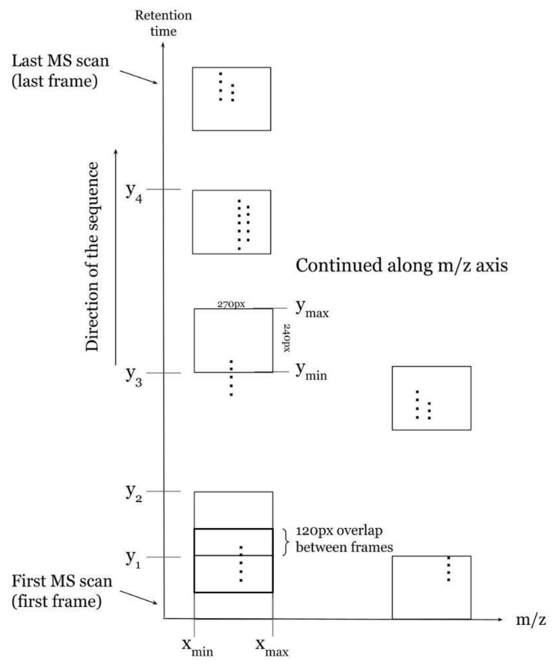

Figure 3. The sliding window process along the retention time axis is used to separate the generated image into smaller frames for model training and prediction.

We developed a sliding window procedure that builds upon DeepIso's IsoDetecting module by using static images instead of dynamic motion from FC-RNN [11]. By doing this, we removed the dependency of the data to optimize the size of the training dataset. With a height of 240px and width of 270px as the base size, we used the following loop to cut each image into separate frames:

Equation 3. Loop to create window frames.

$$window_{i,j} = arr[120i : 120i + 240, 270j : 270j + 270]$$

This created a 120px overlap in the frames. As most features are less than 240px in height, there are virtually no features not encompassed by any singular subwindow. Furthermore, even if a feature is partially cut off by the end of a window, as long as a majority of the feature is included, the model will recognize the pattern as a feature.

Faster-RCNN Training

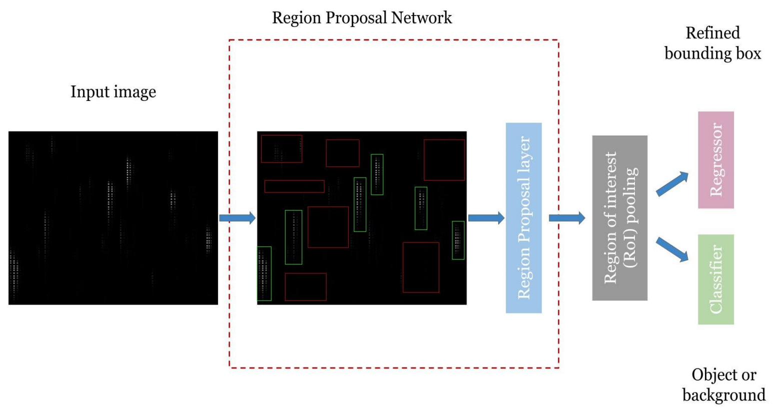

Figure 4. Faster-RCNN's process, including Region Proposal Network, on our image.

We used Faster-RCNN with ResNet-50 to train our model. Faster-RCNN is the state-of-the-art object detection tool that combines a Region Proposal Network and a classification model in order to utilize full-image convolutional features [1]. From the input image, a feature map, which outlines the image's features such as edges and shapes while retaining the original image's structure, is generated from the first few convolutional layers.

Region Proposal Network

The RPN takes the feature map as input and provides a list of about two thousand regions where an object could potentially be located. The RPN proposes regions in the form of anchor boxes around the pixels on the feature map, known as anchors. Nine anchor boxes, which are combinations of three different scales and three different aspect ratios, are proposed for each anchor. The RPN, a convolutional neural network itself in nature, allows it to be combined into a single CNN with the classifier, which makes Faster-RCNN a single-step process [1]. In our case, it is implemented using a ResNet-50 model. The tight integration of Faster-RCNN allows it to be trained end-to-end and reduces the model complexity as well as run time.

Classifier & Regressor

Additionally, RPN contains a classifier and a regressor, which outputs the probability of the region containing an object and which regresses the coordinates of the proposals, respectively [1]. Each anchor box is converted to a feature vector using RoI pooling and fed into the classifier and the regressor, which then filters the anchor boxes to produce a final list of detections. The anchor boxes are filtered through an objectiveness score, which is dependent on the box's IoU (intersection over union) with the ground truth. The box with IoU over 0.7 or the highest IoU among a subset of anchor boxes is assigned a positive objectiveness score, which indicates a higher probability of it containing an object [1]. On the other hand, a box with a low IoU is assigned a negative objectiveness score indicating that it is unlikely to contain an object.

Loss Functions

During the training of Faster-RCNN, the weights of the ResNet-50 CNN are updated in accordance with the loss function, which is defined as follows:

Equation 4. The loss function.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{classifier}} \sum_i L_{classifier}(p_i, p_i^*) + \frac{\lambda}{N_{box}} p_i^* \sum_i L_{box}(t_i, t_i^*)$$

The loss function L takes in the probability, or confidence, of the detection, denoted as p_i, and the bounding box coordinates, denoted as t_i. The final loss is the sum of the losses from the classifier and the bounding box, each with a constant in front. The normalizing terms N are set to the mini-batch size for the classifier and the number of anchor locations for the box. λ serves as a balancing term so that the losses from the classifier and the bounding box are weighted equally. The classifier loss compares the confidence of the output against the ground truth, denoted as p^*_i, and in a binary form (1 indicating that the object of the class exists and vice versa) as follows:

Equation 5. Classifier loss formula.

$$L_{classifier}(p_i, p_i^*) = -p_i^* log(p_i) - (1 - p_i) log(1 - p_i)$$

Additionally, the loss of the bounding box is restricted by the ground-truth term p_i^* in which the entire term is 0 when the object of L_1 loss between the predicted bounding box coordinates and the ground-truth box.

Equation 6. Bounding box loss formula

$$L_{box}(t_i, t_i^*) = L_1^{smooth}(t_i, t_i^*) = \sum_{j \in \{x,y,w,h\}} y$$

$$where\ y = \begin{cases} 0.5(t_{ij} - t_{ij}^*)^2 & if\ |t_{ij} - t_{ij}^*| < 1 \\ |t_{ij} - t_{ij}^*| - 0.5 & if\ |t_{ij} - t_{ij}^*| \geq 1 \end{cases}$$

Soft Non-Maximum Suppression

As we evaluated the output by our model while filtering out detections with low confidence (<0.1), we noticed several repeated detections of the same features. To increase the accuracy of our detections, we utilized Soft Non-Maximum Suppression [24] to filter out the duplicate detections and reduce redundancy. Soft-NMS, instead of removing the low-confidence detections as in its predecessor NMS, simply reduces the confidence of the repeated detection. The detailed algorithm is as follows:

```
Let $S$ = a set of initial detections; $K$ = an empty set to hold the list of detections to keep; $u$ = NMS threshold
While $S \neq \emptyset$:
d = detection with the highest confidence in S
Add $d$ to $K$
For $s_i$ in $S$:
    If $IoU(s_i, d) > u$: Multiply the confidence of $s_i$ by $(1 - IoU(s_i, d))$
```

When encountering two overlapping true positive detections, NMS has a chance of mistakenly removing one of them, but since soft-NMS removes the repeated detections by reducing its confidence to below the threshold (0.1), it leaves a chance of keeping both true positive detections by allowing high confidence detections to stay above the threshold even after the soft-NMS process. Thus, soft-NMS provides a balance between reducing false positives and false negatives.

Experiment and Results

Window Size Optimization

As mentioned in the introduction, the LC-MS map is far too large to be efficiently processed; thus, we use an overlapped cropping mechanism to create smaller frames from the original image. Compared to the dynamic DeepIso IsoDetecting module, our process employs a 240 x 270 pixel window that slides across the retention time axis with 120 pixel overlap between frames, applied on a static image to eliminate dependency from previous frames. Optimization of this process is shown in Figure 5; with larger window sizes, though the runtime is significantly lowered, the number of features the model is able to detect accurately after training is inadequately low. This is because there are fewer frames generated to train with and each frame has a lower resolution. Additionally, with such a large height, many features are captured by multiple windows for each scan, producing duplicate images in the training dataset and increasing the likelihood of overfitting even with a low number of epochs. However, as the window height decreases to 120px, there comes a point where the window is no longer tall enough to encompass a full feature; larger features are no longer accounted for in the model's training, and thus the model is unable to recognize these particularly complex peptides. Furthermore, even with GPU hardware, the runtime for the smallest size is exceedingly high and not offset by an increased number of features. Hence, we choose an intermediate height of 240px to split the windows by, thus creating optimally sized frames to train the model with and producing a sufficient number of features detected after training while minimizing both repetition and runtime.

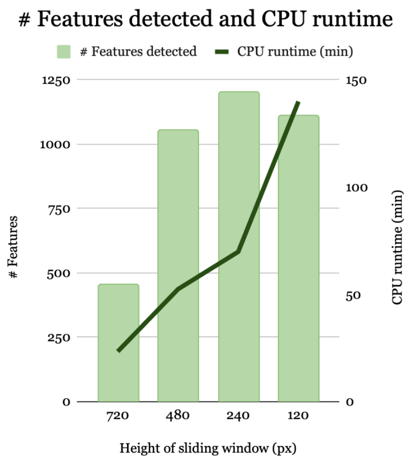

Figure 5. As the size of the sliding window decreases, both the number of features detected and the runtime increase until the window reaches 120px, at which point the number of features decreases.

Training

Since the features in the LC-MS map are relatively sparse, there occur situations where a window is empty and thus useless for training the model. We removed these windows for the efficiency of training. After removing these empty frames, the remaining frames each contain between 1 and 10 features. Table 1 shows the number of windows in each step of our screening process to ensure only images containing features are used for training.

Table 1. Number of windows in each step of the pre-training process

File	Initial windows generated	Empty windows	Final windows used
PXD006096-1	14949	13845	1104
PXD006096-2	16053	13814	2239
PXD006096-3	17188	13994	3194
PXD010357-19	12428	8945	3483
PXD010357-20	12717	8165	4552
PXD004684-H	13867	8142	5725
PXD004684-L	17551	10852	6699
Total	**104753**	**77757**	**26996**

The empty windows made up nearly 75% of the frames generated by our sliding window process. The elimination of these frames from the training data resulted in an estimated 36-hour reduction in training time, contributing to our model's efficiency. About twenty percent of the remaining data is first set aside for testing. We then randomly and independently split the windows into training and validation subsets.

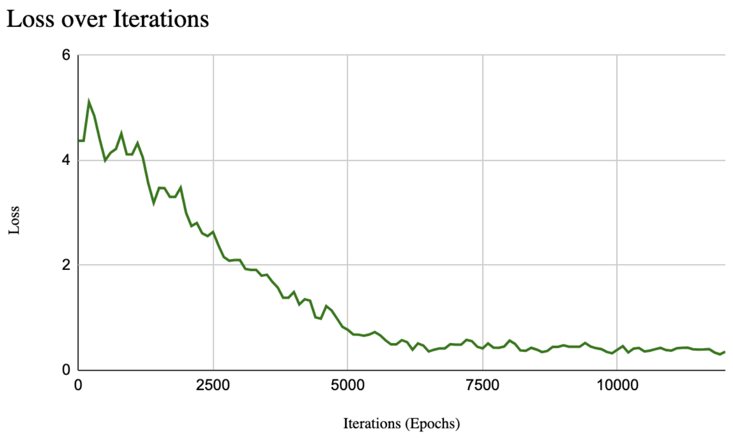

Figure 6. The relative performance of the model after different numbers of epochs are used for training. There occur 1000 iterations for every epoch.

Since the model uses the loss to tune its weights during training, the rate at which the loss decreases indicates the model's rate of improvement. After testing various learning rates, we found that the loss for nearly all of them approached the same satisfactory result, simply at different speeds. Thus, we settled on a learning rate of 0.0001. As we trained on the 12 epochs, the loss is recorded and thus the relative performance of the model is shown. As depicted in Figure 6, the model in epochs 1 and 2 produced largely erroneous results. After beginning to stabilize at around 5 epochs, no improvements in performance were seen in the validation subset after 7 epochs. As we continued training for the next 5 epochs afterward, the loss was relatively unchanged. Thus, we concluded that the model's performance was relatively stable, and no further progress was made. We stopped the model's training at 12 epochs to impede overfitting.

Performance Evaluation

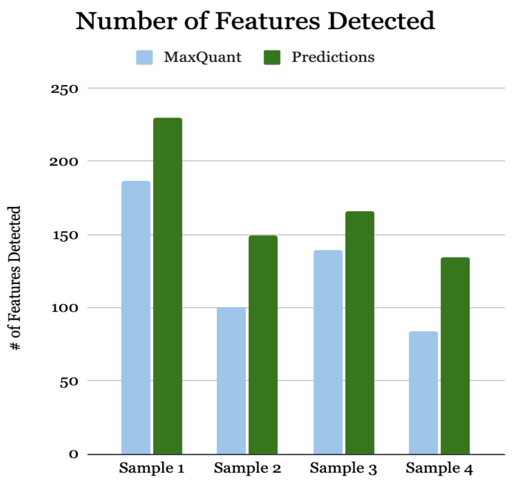

Figure 7. Our model is able to accurately detect more features than MaxQuant. Testing samples are acquired from PXD006096, PXD010357, and PXD004684 [13, 17, 18].

On average, our model is able to detect 35% more high-confidence MS2 features than the conventional feature detection tool MaxQuant. The samples we tested on are small sections of images, again generated with our script and processed with the sliding window procedure. Due to the nature of MS2 spectra, it is difficult to designate high-confidence MS2 features using a 100% accuracy metric. Thus, we cannot evaluate our model based on a percentage accuracy; we instead manually analyze the predictions made by our model in regard to the likelihood of false-positive detection. Furthermore, our model is able to localize the bounding boxes around each detected feature, giving more accurate values for both the m/z ratio and retention times. There is, on average, a 16% decrease in the area of a bounding box for any given feature identified by both our model and MaxQuant. Figure 8 gives a side-by-side comparison and overlaid output as a comparison of our model's performance against that of MaxQuant.

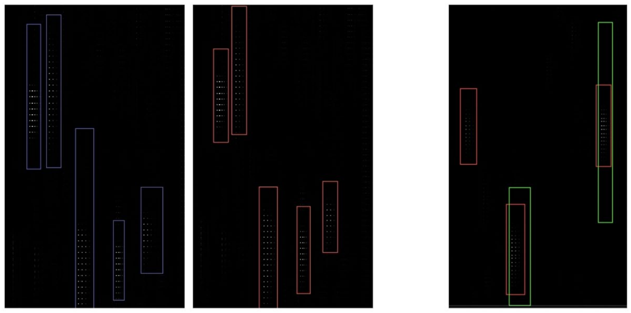

Figure 8. Side-by-side and overlaid comparison of our model against MaxQuant. The red boxes are from our model, while the purple and green are generated by MaxQuant.

In the side-by-side comparison, our bounding boxes are clearly shorter in length and width than those generated by MaxQuant. This is also shown in the overlay, as well as a feature detected by our model that is omitted by MaxQuant's detection. With human observation, it is evident that the start and end retention time of the features shown in the images is closer in value to the lines dictated by our model, thus proving that our model is able to provide more accurate values and provide ease of subsequent analysis in the LC-MS/MS identification process. Our model also outputs a table of values (Table 2) indicating the position of the features on the image, allowing for easy calculations for feature area and m/z and retention time values.

Table 2. Sample output of our model's table of values.

File	x_{min}	y_{min}	x_{max}	y_{max}
15-14	77	130	97	308
35-17+	144	357	170	468
25-17+	50	131	74	249
26-11	214	194	238	302
26-11+	84	284	107	377
27-13	210	162	242	324
19-14	237	278	256	383

The file names, shortened for simplification purposes, indicate the position of the window within the image. For instance, 15-14 refers to the 15th window along the m/z axis and the 14th window along the retention time axis. The files with a + refer to the overlapped windows, i.e., 35-17+ indicates the window found in the overlap between scans 17 and 18. With this output, one can easily calculate the original m/z and retention time endpoints through the reversal of our normalization equation.

Discussion and Conclusion

In this paper, we developed a deep learning-based model that can accurately conduct MS2 feature detection. Among other properties learned by the neural network, the model is trained to recognize the general characteristics of a peptide feature. This includes the general bell shape of a feature, the equidistance on the m/z axis between isotopes, and their overlapping nature. Our model performs at an average 35% increase in the number of features detected in comparison to MaxQuant and a 16% reduction in bounding box area, showing that it is both more sensitive and accurate than conventional feature detection tools and proving its superior capabilities in relative performance and localization despite the complexities of feature detection and extra challenges presented by MS2 data.

We would like to explain the significance of using comparisons against MaxQuant and Dinosaur as a metric of our model's performance. Since a 100% accurate feature detection tool does not exist, it is impossible to know the full truth of the features. However, our model's training and performance evaluations are not intended to mimic the output of MaxQuant and Dinosaur simply because their annotations' union is used as the ground truth for training. Instead, this union replaces manual annotations.

We would also like to note that although different RAW files from cancer samples were used for testing, the scope of our results is not limited to applications related to cancer proteins. Regardless of the protein species used to generate the raw data, the features, once processed by our script, appear nearly identical, so our model should be able to process the MS2 data generated from different proteins without the need for ad-hoc training. Our results suggest that our model will be applicable to a multitude of situations regarding proteomic analysis.

In future work, we would like to perform more testing on larger datasets to measure the model's performance on different types of proteins, as well as increase confidence in its capabilities for detecting cancer proteins. Furthermore, as our current output only gives values for the m/z endpoints and retention time start and end, we plan to incorporate a module to identify the number of isotopes in a feature by using its width. Lastly, we would like to implement the sliding window procedure to scan across the m/z axis in addition to the retention time axis in order to capture any feature that may be cut off by windows. This would also contribute to the isotope detecting module while again increasing the number of features detected.

We also plan on adding several functionalities to the model, such as allowing it to convert back from pixel indices to m/z and retention time, i.e., perform a reversal of the normalization process used to generate the image. In our current normalization process, as the pixel value must be an integer, there is a minor source of error as the equation must round the normalized m/z or retention time value down to the nearest whole number in order to assign a pixel value to the peak. Thus, a reversal of this process would result in m/z and retention times slightly lower than the original input values. However, as the size of the original image increases, this error on either end approaches negligible values. We plan on addressing this issue by improving the model's sensitivity and ability to process larger images in order to make predictions on these increasingly large files to enhance its accuracy. This will also be applicable to the surplus of increasingly complex data generated by the rapid improvements in mass spectrometry hardware.

Acknowledgments

Research reported in this publication was supported by the National Library of Medicine of the National Institutes of Health under award number R15LM013460.

References

[1] Ren, Shaoqing, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6,2016, pp. 1137-1149, doi:10.1109/TPAMI.2016.2577031

[2] Zohora, Fatema Tuz, et al. “DeepIso: A Deep Learning model for Peptide Feature Detection from LC-MS Map.” *Scientific Reports*, vol. 9, no. 17168, 2019. https://doi.org/10.1038/s41598-019-52954-4.

[3] Sellers, K. & Miecznikowski, J. “Feature Detection Techniques for Preprocessing Proteomic Data.” *International Journal of Biomedical Imaging*, vol. 2010, 2010, 896718. doi: 10.1155/2010/896718.

[4] Ma, Chunwei, et al. “Improved Peptide Retention Time Prediction in Liquid Chromatography through Deep Learning.” *Analytical Chemistry*, vol. 90, no. 18, 2018, pp. 10881-10888, https://doi.org/10.1021/acs.analchem.8b02386.

[5] Nicolescu, Teodor Octavian. “Interpretation of Mass Spectra.” *Mass Spectrometry*, edited by Mahmood Aliofkhazraei, IntechOpen, 2017. doi: 10.5772/intechopen.68595.

[6] Cox, J & Mann, M. “MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide quantification.” *Nature Biotechnology*, vol. 26, 2008, pp. 1367-1372. https://doi.org/10.1038/nbt.1511.

[7] Teleman, John, et al. “Dinosaur: A Refined Open-Source Peptide MS Feature Detector.” *Journal of Proteome Research*, vol. 15, no. 7, 2016, pp. 2143-2151. https://doi.org/10.1021/acs.jproteome.6b00016.

[8] Cappadona, S., Baker, P. R., Cutillas, P. R., Heck, A. J. & van Breukelen, B. “Current challenges in software solutions for mass spectrometry-based quantitative proteomics.” *Amino acids*, vol. 43, 2012, pp. 1087–1108. https://doi.org/10.1007/s00726-012-1289-8

[9] Savojardo, Castrense, et al. “DeepSig: deep learning improves signal peptide detection in proteins.” *Bioinformatics*, vol. 34, no. 10, 2018, pp. 1690-1696. https://doi.org/10.1093/bioinformatics/btx818.

[10] Tran, Ngoc Hieu, et al. “De novo peptide sequencing by deep learning.” *Proceedings of the National Academy of Sciences*, vol. 114, no. 31, 2017, pp. 8247-8252. https://doi.org/10.1073/pnas.1705691114.

[11] Yang, X., Mochanov, P., Kautz, J. “Multilayer and Multimodal Fusion of Deep Neural Networks for Video Classification.” *Proceedings of the 24th ACM International Conference on Multimedia*, 2016, pp. 978-987, doi: 10.1145/2964284.2964297.

[12] Rost, H., Sachsenberg, T., Aiche, S. et al. “OpenMS: a flexible open-source software platform for mass spectrometry data analysis.” *Nature Methods*, vol 13, 2016, pp. 741-748. https://doi.org/10.1038/nmeth.3959.

[13] Hoffman, Melissa A et al. “Comparison of Quantitative Mass Spectrometry Platforms for Monitoring Kinase ATP Probe Uptake in Lung Cancer.” *Journal of Proteome Research*, vol. 17, no. 1, 2018, pp. 63-75. doi:10.1021/acs.jproteome.7b00329.

[14] McLafferty, F. W. “Tandem mass spectrometry (MS/MS): a promising new analytical technique for specific component determination in complex mixtures.” *Accounts of Chemical Research*, vol. 13, no. 2, 1980, pp. 33-39. Doi: 10.1021/ar50146a001.

[15] Kessner, D., Chambers, M., Burke, R., Agus, D. & Mallick, P. “ProteoWizard: open source software for rapid proteomics tool development.” *Bioinformatics*, vol. 24, no. 21, 2008, pp. 2534-2536. https://doi.org/10.1093/bioinformatics/btn323

[16] Adusumilli, R. & Mallick, P. “Data Conversion with ProteoWizard msConvert.” *Proteomics, Methods in Molecular Biology*, vol. 1550, 2017. https://doi.org/10.1007/978-1-4939-6747-6_23.

[17] Stewart, P. A., Welsh, E. A., Slebos, R. J. C. *et al*. “Proteogenomic landscape of squamous cell lung cancer.” *Nature Communications*, vol. 10, no. 3578, 2019. https://doi.org/10.1038/s41467-019-11452-x.

[18] Stewart, P. et al. “Relative protein quantification and accessible biology in lung tumor proteomes from four LC-MS/MS discovery platforms.” *Proteomics*, vol. 17, no. 6, 2017. Doi: 10.1002/pmic.201600300.

[19] Hu, A. et al. “Technical Advances in proteomics: new developments in data-independent acquisition.” *F1000Research*, vol. 5, 2016. doi: 10.12688/f1000research.7042.1.

[20] Tada, I. et al. “Correlation-Based Deconvolution (CorrDec) To Generate High-Quality MS2 Spectra from Data-Independent Acquisition in Multisample Studies.” *Analytical Chemistry,* vol. 92, no. 16, 2020, pp. 11310-11317. Doi: 10.1021/acs.analchem.0c01980.

[21] Virreira Winter, S., Meier, F., Wichmann, C. et al. “EASI-tag enables accurate multiplexed and interference-free MS2-based proteome quantification.” *Nature Methods*, vol. 15, 2018, pp. 527–530. https://doi.org/10.1038/s41592-018-0037-8.

[22] Weissert, H. & Choudhary, J. S. “Targeted Feature Detection for Data-Dependent Shotgun Proteomics.” *Journal of Proteome Research*, vol. 16, no. 8, 2017, pp. 2964-2974. doi: 10.1021/acs.jproteome.7b00248.

[23] Wang, X., Shen, S., Rasam, S. S. & Qu, J. “MS1 ion current-based quantitative proteomics: A promising solution for reliable analysis of large biological cohorts.” *Mass Spectrometry Reviews*, vol. 38, no. 6, 2019, pp. 461-482. doi: 10.1002/mas.21595.

[24] Bodla, N., Singh, B., Chellappa, R. & Daivs, L. “Soft-NMS – Improving Object Detection with One Line of Code.” *IEEE International Conference on Computer Vision*, 2017, pp. 5562-5570. https://doi.org/10.48550/arXiv.1704.04503.

[25] Zhang, G. et al. “Overview of peptide and protein analysis by mass spectrometry.” *Current protocols in protein science*, vol. 16, no. 1, 2010. doi:10.1002/0471140864.ps1601s62.

[26] Ma, C. et al. “DeepRT: deep learning for peptide retention time prediction in proteomics.” *arXiv*, 2017. https://doi.org/10.48550/arXiv.1705.05368.