A Novel Phosphosite Localization Approach using Tandem Phosphoprotein Mass Spectra and Temporal Dilated Networks

Molly Arthur

1. Abstract

Dysregulation of protein phosphorylation during cellular development and signal transduction is associated with a myriad of human cancers, Parkinson's Disease, and Alzheimer's Disease. Understanding the exact amino acid binding sites is critical for medical and pharmacological applications but current experimental and computational methods are time-consuming or imprecise.

I developed a novel approach to identifying phosphosites using peptide tandem fragmentation mass spectra, which is more applicable for drug design compared to the current method of sequence input. I created a temporal convolutional neural network (TCN) to identify the exact phosphorylated amino acid from a given spectrum. I processed the mass spectra data to create a two-vector representation for efficient analysis. My model uses a unique architecture that promotes flexibility and interpretability with considerations for long-range associations between spectral peaks critical for phosphosite identification. I accounted for neutral losses, a common phenomenon in proteomics mass spectrometry, through the independent identification of unique spectral patterns. Instead of requiring existing input features that could increase bias or decrease the identification of rare or new phosphosites, my model identifies its own patterns.

My model achieved an average accuracy of 95% compared to the accuracies (76 - 78%) of sequence-based models. Future work includes developing my model to predict neutral losses. My novel approach provides a critical step to the biological understanding of phosphorylation events and the effective development of medical and pharmacological tools to correct biochemical pathways for targeted treatments that reduce risk and discomfort for patients.

2. Introduction

2.1. Post-Translational Modifications (PTMs)

PTMs are chemical changes made to proteins through the addition or removal of functional groups, other proteins, lipids, or sugars to or from amino acid side chains [1]. These events regulate protein activity at all stages of the protein lifecycle, from directly after translation to activating catalytic activity to eventual degradation [2]. With an approximate human genome size of 20,000-25,000 genes, PTMs play a critical role in proteomic diversity with the proteome estimated to include over 1 million individual proteins [3]. Figure 1 depicts the increased complexity of the proteome compared to the genome.

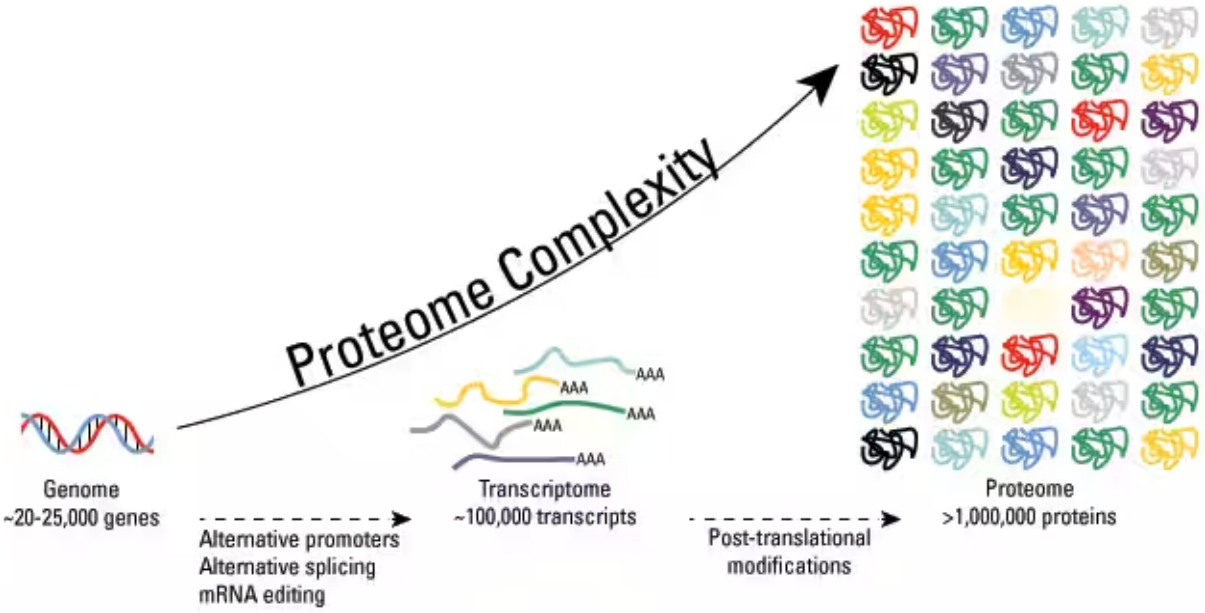

Figure 1: A visual representation of the effect of PTMs on proteome diversity demonstrates how the transcriptome with approximately 100,000 transcripts produces proteins modified to produce the full human proteome [2].

2.2. Phosphorylation

Phosphorylation is one of the most important PTMs and plays a critical role in regulating the cell cycle, growth, apoptosis, and signal transduction pathways [4]. Between 30 and 40% of eukaryotic proteins undergo phosphorylation at some point, and approximately 230,000 and 156,000 individual phosphosites exist in the human and mouse proteomes respectively [5]. Phosphorylation is also present in prokaryotes with approximately 5-10% of proteins undergoing modification [6]. The reversible nature of phosphorylation, where a kinase phosphorylates a specific amino acid side chain and a phosphatase hydrolyzes the phosphate group to remove it, enables cell regulation with an "on/off" switch corresponding to phosphorylation or dephosphorylation for a variety of signaling pathways [7]. Simultaneously, the reversibility of phosphorylation poses a unique challenge in identifying exact phosphosites as any given sample may contain proteins with amino acids that function as phosphosites but are not currently phosphorylated [8].

Phosphosite identification is further complicated by the variety of proteinogenic amino acids that can undergo phosphorylation. In eukaryotes, phosphorylation is primarily limited to serine (S), threonine (T), tyrosine (Y), and histidine (H). At the same time, prokaryotic phosphorylation can also occur on arginine (R), aspartic acid (D), or cysteine (C) [9]. The varying bonds formed during phosphorylation between the amino acid and phosphate can change the function of the phosphosite within the protein and its greater regulatory role. For example, S, T, and Y form

relatively low-energy ester bonds when phosphorylated compared to higher-energy phosphoramidates (P–N) bond from H and R. Phosphorylated aspartic acid forms a mixed anhydride or acyl phosphate, and cysteine forms a phosphorothiolate (P–S) bond [10]. The phosphorylation of tyrosine is in Figure 2 where the nucleophilic group of tyrosine attacks the terminal (γ) phosphate group of ATP [11].

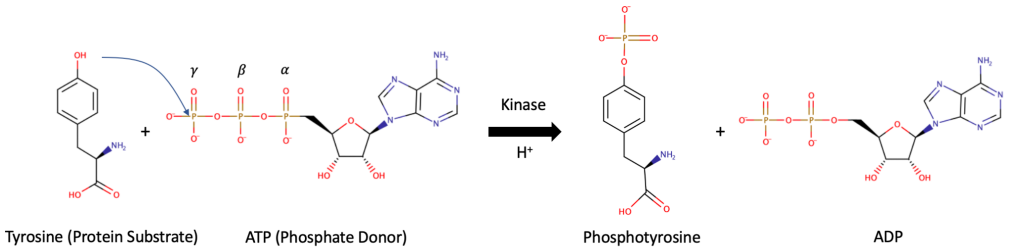

Figure 2: The phosphorylation of tyrosine where a kinase serves as the enzymatic mediator.

The deregulation of phosphorylation has been associated with the onset of multiple diseases including cancer, Alzheimer's Disease, Parkinson's Disease, and other degenerative disorders [12]. Potential treatments for such diseases lie in the ability for precise identification of a phosphosite for pharmacological developments such as kinase or phosphatase inhibitors for specific proteins [13]. The frequency of phosphorylation on many different signaling pathways leads to the potential of a relationship between the onset of disease and abnormal phosphorylation. Sample deregulation for a variety of disease pathways is in Figure 3.

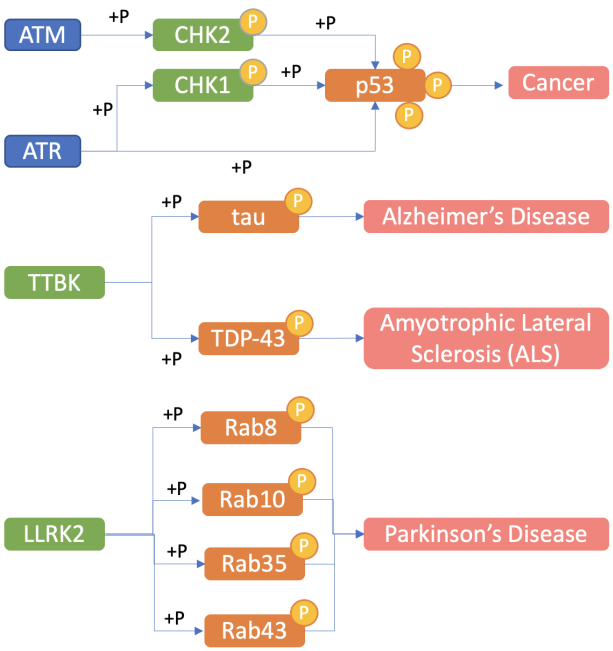

Figure 3: Simplified pathways of protein phosphorylation events that lead to various diseases. Kinases are in blue, primary proteins are in green, critical regulatory proteins are in orange, and diseases caused by the deregulation of the proteins are in pink. PI_3 kinases phosphorylate CHK1 and CHK2. These kinases phosphorylate p53 which can enable cancer when mutated. TTBK phosphorylates tau and TDP-43 which may lead to Alzheimer's Disease and ALS respectively. LLRK2 phosphorylates four Rab proteins which lead to Parkinson's Disease.

2.3. Mass Spectrometry (MS) Proteomics

The important identification and analysis of phosphorylation sites have now become mostly fulfilled by MS implementations. MS-based proteomics was first established by Ficarro et al. in 2002 and has since become one of the most utilized approaches for analyzing proteins at whole-proteome scales in biological fluids or tissues [14,15]. However, typical MS that only produces a mass-to-charge ratio (m/z) is not always specific enough to identify complex biological molecules. Tandem mass spectrometry (MS/MS) resolves this issue by performing two rounds of analysis: first with the intact analyte and then after fragmenting the analyte with inert gas molecule collision analyzing the fragments [16]. MS/MS, therefore, provides insight into a protein's substructures, which is crucial for the identification of protein phosphorylation sites [17].

A common phenomenon during the fragmentation process of MS/MS is a neutral loss which poses a particular challenge to the study of phosphorylation. During high-energy fragmentation, the phosphate group may dissociate from the peptide as a neutral species [18]. The peptide, now dephosphorylated, may not produce the necessary intense fragment ions necessary for phosphopeptide identification. While methods have been developed so that general

phosphopeptide identification is not hindered by neutral loss, confident phosphosite localization is still significantly impacted by the phenomenon [19].

The amount of publicly available MS-based proteomic data has increased exponentially recently due to the large pools of fragmentation spectra, which represent partial protein sequence information [20]. Each spectrum includes characteristic peak patterns based on the specific fragmentation process and peptide [21]. However, spectra are not easily identified because of PTMs or mutations. A variety of computational methods have been implemented to address this issue, most relying on an "open search" strategy that attempts to match a spectrum to an already known sample in a database [22]. The typical workflows for MS-based proteomic analysis are summarized in Figure 4.

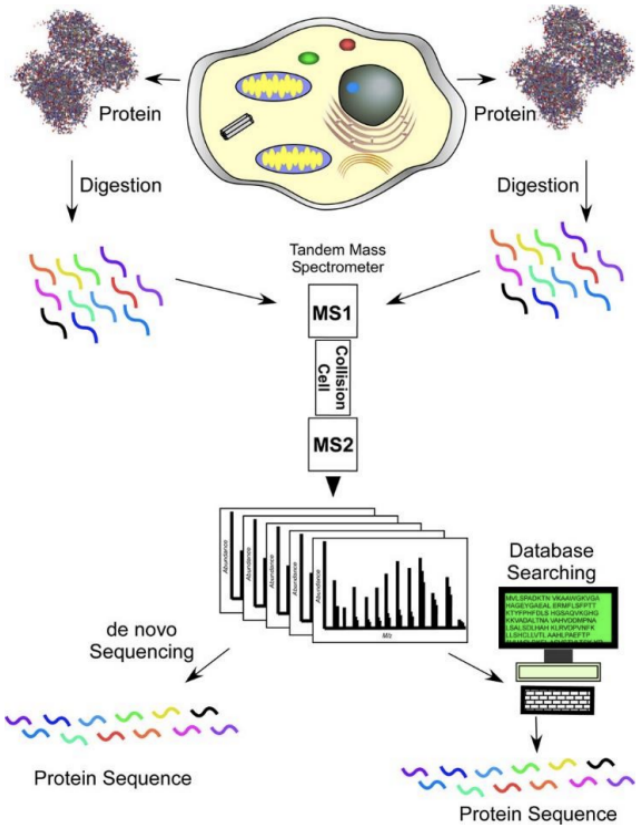

Figure 4 [17]: A high-level outline of typical proteomics analysis. Proteins are first extracted from a biological tissue or other samples of interest and are then digested into peptides, typically by trypsin. The peptides created normally have basic terminal residues to promote ionization. Two layers of mass analysis following MS/MS. The first MS is completed on the whole peptide and the second MS is completed on characteristic fragments that were created in the collision cell. The data are then typically analyzed via two methods: de novo search algorithms or database searching [17].

Currently, computational analysis of MS/MS spectra focuses on a derived amino acid sequence [1]. De novo search algorithms and database searching tools, the two most common spectra analytics tools, focus on generating amino acid sequences from the spectra or searching datasets with preexisting matches between a specific spectrum and sequence respectively [23, 24]. While identifying the sequence of a particular protein is critical to understanding aspects of its structure and function, for phosphosite localization an amino acid sequence provides less specific information such as individual bonds compared to raw spectra. Furthermore, a complete amino acid sequence for a given peptide is necessary for full phosphosite identification since many proteins have multiple phosphosites [15].

Within MS-based proteomics, there have been primarily two avenues of machine learning applications: fragment intensity and de novo sequencing (the novel generation of a sequence without a reference sequence for alignment). However, both of these methods are focused on or require a peptide sequence as an input, output, or intermediary rather than a spectrum alone. Recurrent neural networks have recently been utilized to predict fragment intensities, or strength of specific ion peaks, in pDeep and Prosit [25, 26], and implemented for de novo sequencing in DeepNovo and PointNovo [27, 28, 29]. A significant limitation of these approaches, especially with potential application to phosphosite localization, is the fixed and untrainable nature of the fragmentation patterns characterized by ion types and amino acids; the models are unable to independently characterize spectra patterns [30]. In 2022, Altenburg et. al produced the model AHLF that can independently recognize relevant fragmentation patterns without ever receiving them during the training process which resolves the limitation described above [31].

2.4. Phosphosite Localization Methods

The detection of phosphopeptides has been more rigorously studied both experimentally and computationally compared to the localization of phosphosites. In MS-based and broader proteomics, phosphopeptide detection presents a comparatively easier task with less specificity, precision, and flexibility necessary for detection. The varied potential amino acids that can serve as phosphosites, many with different binding patterns, the reversible nature of phosphorylation, and the phenomenon of neutral loss all pose particularly difficult challenges to phosphosite identification, especially from an MS-based approach.

The few current phosphosite localization methods are generally ambiguous and at times contradictory [32]. Upon writing, there are no known existing methods for phosphosite localization that are derived directly from spectra. Instead, most methods such as MusiteDeep and DeepPhos rely on small 20-amino acid long sequences for phosphosite identification [33, 34]. The small inputs reduce the accuracy of the models since there are not enough phosphosite-specific features, potentially increase model bias by requiring detailed arbitrary feature selection, and create limited whole proteome applicability which is critical for medical and pharmacological applications [35]. Feature selection presents a particularly difficult problem

in the context of phosphorylation. Phosphorylation events are still not well understood so picking all necessary features for adequate localization of unique, or potentially new, phosphosites is a difficult task [36].

An MS-based approach would potentially resolve some of these issues. Spectra for entire peptides provide significantly increased detail compared to small sequences and do not require the second step of acquiring a sequence, which is especially helpful for lab-based settings where peptides from specific tissues are used.

2.5. Temporal Convolutional Networks (TCNs)

To adequately approach phosphosite localization from MS/MS using a machine learning method, a unique model with the ability for long-range associations is critical. Traditional convolutional neural networks generally only make accurate predictions from close kernels.

Lea et al. established TCNs in 2016 for video-based action segmentation [37]. The framework replaces typical two-model architectures of recurrent neural networks such as long short-term memories (LSTMs) and gated recurrent units (GRUs) that are both comparatively inefficient and unable to learn from long-range associations (a necessity for analyzing spectra given the distance between peaks) [38]. TCNs are deep neural networks consisting of dilated convolutions (layers that increase the size of the kernel (input) by inserting holes within the data) and can learn long-range associations. Another critical feature of TCNs is the ability of the architecture to take a sequence of any length and map it to an identical length output [39]. TCNs provide a unique approach to phosphosite localization via MS by accommodating the long-range nature of spectra. A TCN enables ad-hoc learning, a type of deep learning that emphasizes pattern recognition without specific built-in features that provide a model with such patterns beforehand. A generalized TCN architecture is in Figure 5.

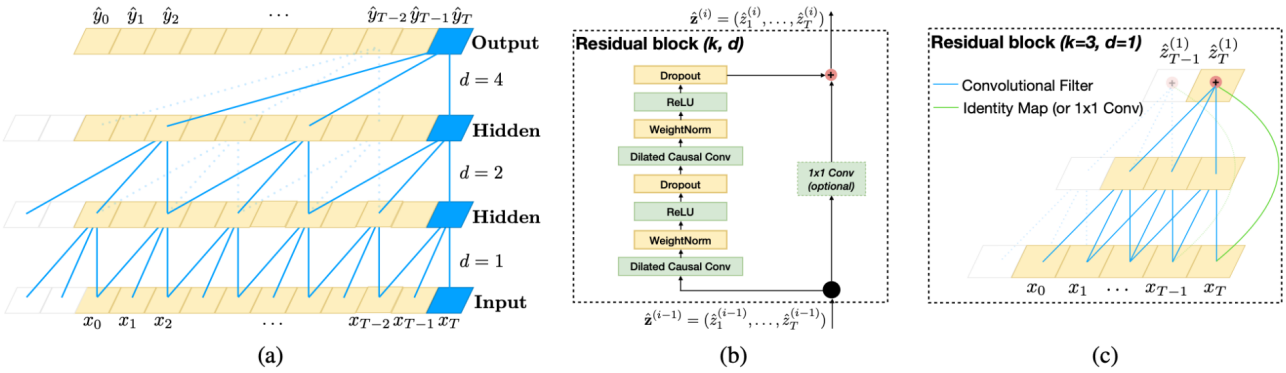

Figure 5 [37]: The traditional architecture of a TCN. (a) A dilated causal convolution where the dilation factors increase through the layers. (b) The TCN residual block implements two layers of dilated causal convolution. Each layer has sublayers with the rectified linear unit (ReLU) applied to prevent exponential growth, weight normalization layers that ensure convergence and speed up training, and dropout layers which temporarily remove unused hidden nodes for efficiency. (c) Residual connection within a TCN allows the model to skip some layers to provide more pathways for data.

2.6. Approach

This research presents a novel approach to the detection and localization of phosphosites using tandem MS and a TCN architecture. By using input data that is widely publicly available and allows for the high specificity necessary for localization and a machine learning method that can perform long-range associations on spectra without detailed feature input, my method provides an efficient and accurate approach to phosphosite localization.

3. Methods

3.1. Dataset

Datasets of the phosphoproteomes of humans (*Homo sapiens*), common liverwort (*Marchantia polymorpha*), thale cress (*Arabidopsis thaliana*), and the house mouse (*Mus musculus*) from the Proteomics Identifications (PRIDE) Archive [40] and the Japan Proteome Standard (jPOST) Repository [41] were utilized for both training and testing of the model. These species' spectra were both the most robust from the archive given their status as model organisms and provided a variety of eukaryotes within the overall data. Specifically, the repositories PXD012174 [42], JPST000685 [43], JPST000703 [44], PXD013868 [45], PXD014865 [46], PXD015050 [47], and PXD000138 [48] were used for both their robustness and the efficient comparison of results between the proposed model and other non-machine learning MS-based phosphosite localization methods such as MusiteDeep [33] and DeepPhos [34]. I used PXD012174 for the training dataset since it is roughly balanced between phosphorylated and unphosphorylated spectra which provided adequate training of phosphosite localization from the large fragments. The other

datasets were used for testing and comparison to other methods. I initially trained and tested my model exclusively on eukaryotic samples for maximum pharmacological and medical applicability. A sample of the raw data is in Table 5 of Appendix A.

3.2. Two-Vector Representation of Spectra

Due to the high accuracy and resolution of MS measurements of m/z, the potential number of features in a dense feature vector would be too large for efficient training. All the spectra were in mgf (Mascot generic format) which preserved the high resolution. With the raw spectra, a simple conversion of peaks to features would create over 100,000 features for the model to analyze which would dramatically slow run time. I used a two-vector representation of the spectra (separating intensity and m/z) to maintain accuracy but reduce the total number of features. This representation exploits the sparsity created by specific m/z with non-zero intensities using Equation 1, where i is an integer that describes a specific index within a spectra segment and s is the size of the segment, to calculate the m/z remainder: m/z' for each peak (a point of m/z and intensity) [31].

$$m/z \; = \; i \cdot s + m/z' \qquad \text{Equation 1}$$

The representation allows convolutional layers to be applied to the spectra and reduces memory consumption without lessening resolution. The workflow for separating the files is in Figure 6 and a portion of the code implementation is in Dataset.py in Appendix A.

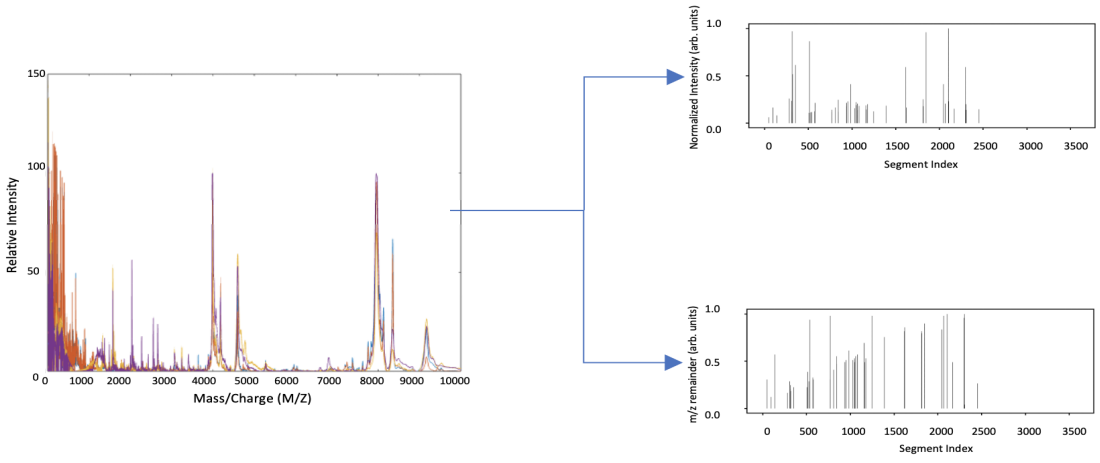

Figure 6: A flowchart demonstrating the preparation of the raw data values. MATLAB msviewer from the bioinformatics toolbox was used to evaluate unedited spectra. I then created a python script to separate the raw spectra into one plot that included reduced segment indexes which were confirmed to not lose peaks while reducing feature size to 3500. The lower m/z remainder plot would allow the true m/z values to be reacquired while saving feature space.

3.3. Model Architecture

To construct my model, I used Python 3.7, Tensorflow 2.0, Pyteomics, and Numpy. I modified the base TCN architecture to accommodate the unique specifications of spectra analysis, especially for phosphosite identification. The full model architecture (Figure 7) can make the critical long-range associations between peaks that are necessary to identify a phosphosite and accommodate the unique two-vector representation. These long-range associations were enabled by the artificially increased receptive field size.

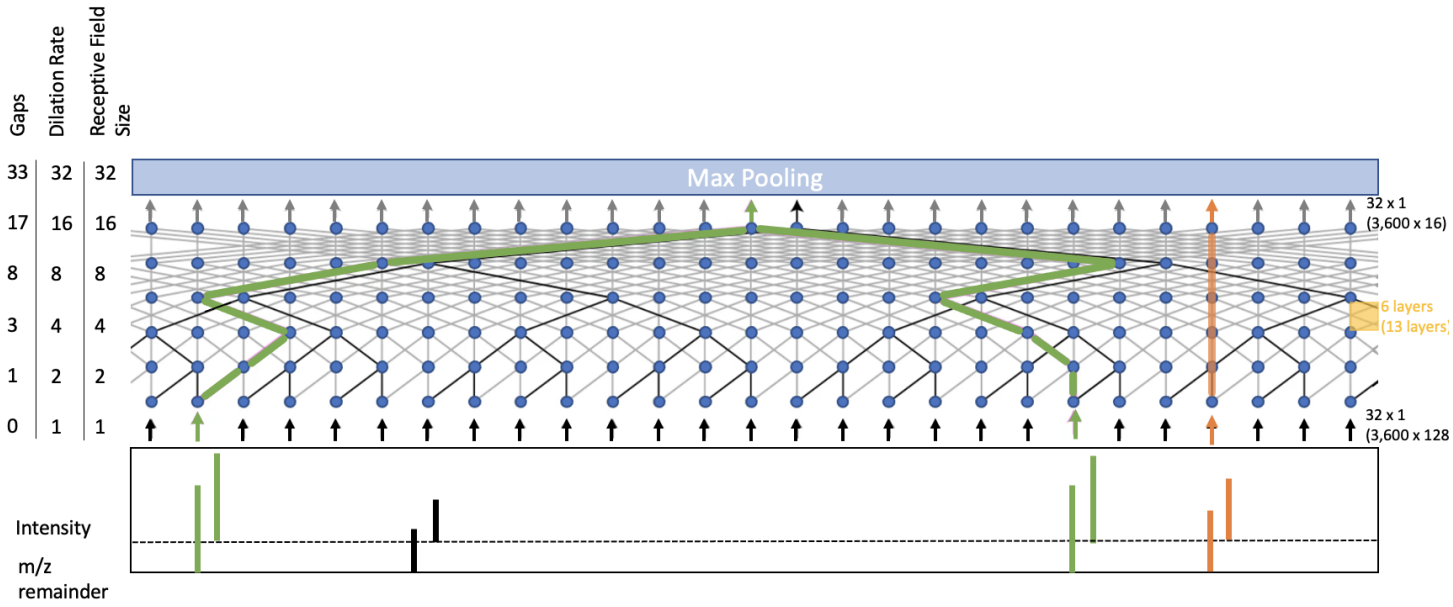

Figure 7: The full model architecture. The two-vector spectra are represented with both intensity and m/z remainder peaks. The long-range association capability of the model is demonstrated through the green peaks. General convolution ability is demonstrated through black peaks and the orange peaks demonstrate the ability of the model to skip layers to maximize efficiency. The figure condenses the layers where the representative size of the figure is accompanied by the true size of the model in parenthesis.

Each layer within the full model architecture is composed of a TCN-based dilation convolution block that promotes long-range associations. Each input feature is dilated to a larger size while undergoing convolution (Figure 8). These dilations are followed by a simple base convolution to create an input for the corresponding layer paired with a ReLU activation.

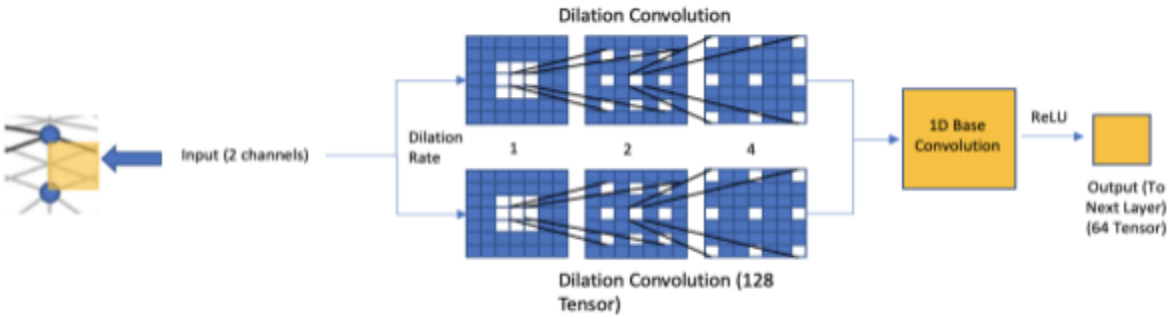

Figure 8: The dilation block of each layer of the full architecture with an artificially increasing dilation with a feature size of 128 tensor. The final output from the block produced a 64 tensor, with each layer systematically reducing the size of the tensor until pooling to produce localization predictions.

While the dilation blocks were critical for long-range associations, not all phosphosites required computationally expensive dilations. Therefore, I also implemented more simple general convolutions which implement basic pattern recognition among spectra. I also utilized skip connections, where layers that produce outputs with a strong enough accuracy could skip dilation layers to improve runtime. These skip connections also enhanced nonlinearity within my model because nonsequential layers could be directly connected. The skip connection process is in Figure 9.

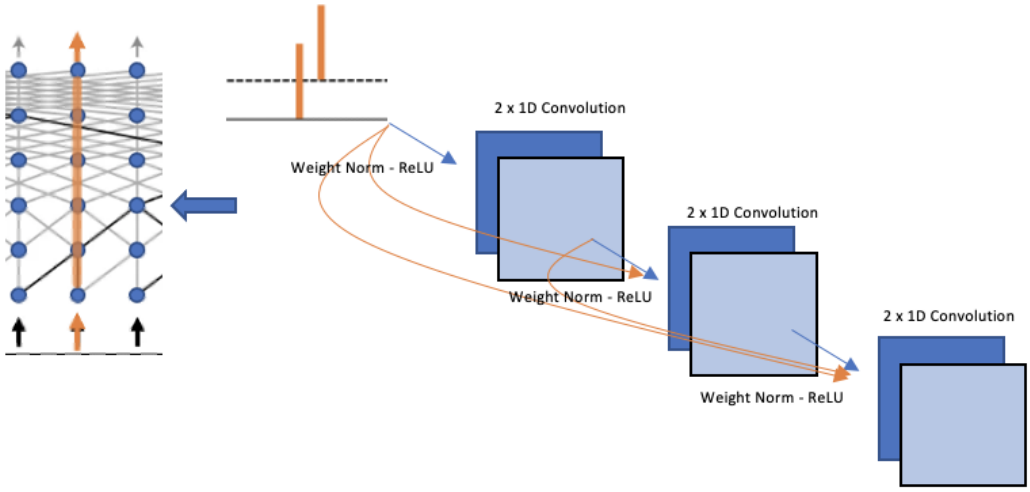

Figure 9: The potential skip connections that were possible within the model. The blue arrows represent the typical feature pathway through all the layers and the orange arrows represent possible skip connections between layers.

After going through the main model architecture, finalized outputs went through phases of max pooling and dropout layers to produce the finalized phosphosite prediction scores (Figure 10). By

identifying specific spectral associations that represented both possible amino acids and phosphosites, the final outputs of the model were predicted fragment intensities and corresponding amino acids that were phosphosites.

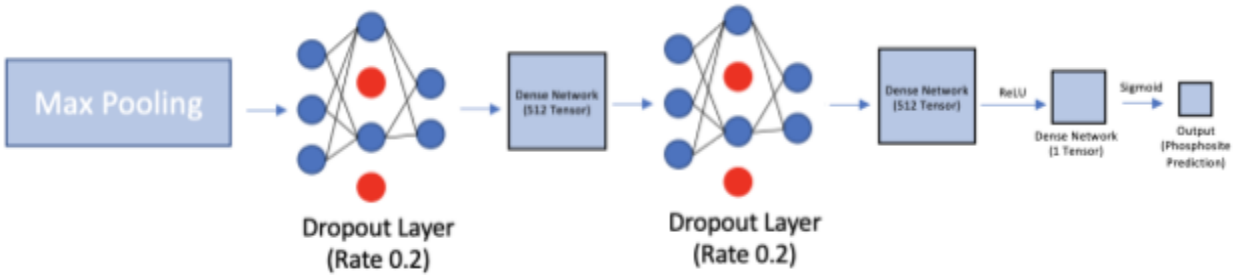

Figure 10: The max pooling from the main architecture went into multiple layers of alternating dropout and dense network layers that removed and condensed extra nodes. The features went through a final ReLU activation and sigmoid function to produce a single output prediction for every identified amino acid.

The model also accounted for neutral losses and phosphorylation reversibility with these same long-range associations and flexible feature identification. Instead of manually inputting features for the identification of neutral losses or potential phosphosites without a phosphate attached to the input spectra, the model developed its own patterns and features of recognition with data that had known neutral losses. A portion of the code of the architecture is in TCN.py in Appendix A.

3.4. Training and Testing Schedule

I performed fourfold cross-validation to produce four independently trained models to ensure the full model could adequately predict new data and combat overfitting, which is especially important given the many unique characteristics of phosphosite localization. I trained the model for 100 epochs with 9,000 steps each to follow the optimal gradient descent algorithm step. Approximately 80% of spectra in PXD012174 were used for training and the rest for testing. The remaining spectra from other databases were solely used for testing, especially to compare against previous phosphosite localization tools. Training.py and Inference.py were portions of the model that carried out the training and testing phases and are in Appendix A.

3.5. Evaluation Metrics

Given the novel nature of my model in terms of the combination of the dataset and model structure utilized, it was important that it performed comparably or better to current phosphosite localization methods that used amino acid sequence input. I intentionally chose specific spectral datasets that had been used in a model that had identified phosphopeptide. Therefore, I would be able to somewhat compare my model's accuracy to that of a model performing a similar, although arguably simpler, task. To compare to true phosphosite localization models, I used typical machine learning accuracy metrics and compared such metrics to those calculated for the

amino acid sequence models. The three evaluation metrics I implemented were a balanced accuracy (Bacc) score, F1 score, and area under the receiver operating characteristic curve (ROC-AUC). The F1 score and Bacc are shown in Equations 2 and 3 respectively where *TP* is a true positive, *TN* is a true negative, *FP* is a false positive, and *FN* is a false negative. The ROC-AUC curve is not a simple equation but rather a calculation of how well the model is at identifying an amino acid as a phosphosite or not.

$$F1 = 2\left(\frac{precision * recall}{precision + recall}\right) \quad Precision = \frac{TP}{TP + FP} \; Recall = \frac{TP}{TP + FN}$$

Equation 2

$$Balanced\ Accuracy = \frac{sensitivity + specificity}{2} = \frac{\frac{TP}{TP + FN} + \frac{TN}{TN + FP}}{2}$$

Equation 3

3.6. Prokaryotic Adaptations

After running the training and testing for my model on exclusively eukaryotic spectra for maximum medical and pharmacological applications, prokaryotic spectra were investigated for a better understanding of phosphorylation in a complete biological context. Given the additional amino acids that can be phosphorylated in prokaryotes, feature selection would become more difficult in a traditional model. Additionally, it was more difficult to identify publicly rigorous MS/MS phosphoproteomes for prokaryotes, and there were even fewer machine learning or computational methods that were implementing these datasets that I could compare my model to in a meaningful way. Therefore, I decided to test my model on prokaryotic phosphosite localization after eukaryotic spectra were tested to not interfere with my primary goal of identifying pharmacologically relevant phosphosites.

I used the PRIDE archive to find relevant spectra datasets for four prokaryotic species: *Klebsiella pneumoniae*, a bacterial pathogen with high antibiotic resistance, *Escherichia coli*, *Staphylococcus aureus*, and *Peptoclostridium difficile*, which all are commonly studied model organisms with important functions or threats to humans. The specific spectral datasets I used were PXD041015 [49], PXD040658 [50] PXD026981 [51], and PXD029827 [52] respectively.

I first tested the new datasets with the model to see if the patterns developed in initial eukaryotic phosphosite localization would be sufficient. I also created another version of the model where the training data came exclusively from prokaryotic data. I took 20% of the spectra from each of the four datasets to serve as the training dataset and then tested the model on the remaining spectra.

3.7. Improved Phosphosite Localization for Thale Cress and Liverwort

During the testing phase of my research, I observed that the localization scores for the datasets of thale cress and liverwort were lower than that of human and mouse datasets. I determined this was most likely due to the exclusive training on a human dataset, so I added training data from all eukaryotic datasets to determine if more variation would prevent a preference toward human phosphosites.

4. Results

4.1. Training Results (Initial Eukaryotic)

This section contains only the training results for the initial eukaryotic datasets. The training was identical for the prokaryotic and more varied training dataset iterations of the model. Furthermore, these iterations were not able to be easily compared to existing results of similar models because of the limited model creation for prokaryotic organisms.

I implemented a standard cross-entropy loss function to monitor the training of all four submodels that were created during cross-validation. The accuracies and losses across all models maintained a stable and expected pattern demonstrating appropriate training despite the unique modifications made to a typical TCN to accommodate spectra specifications. Figure 11 contains the loss plots for each of the models.

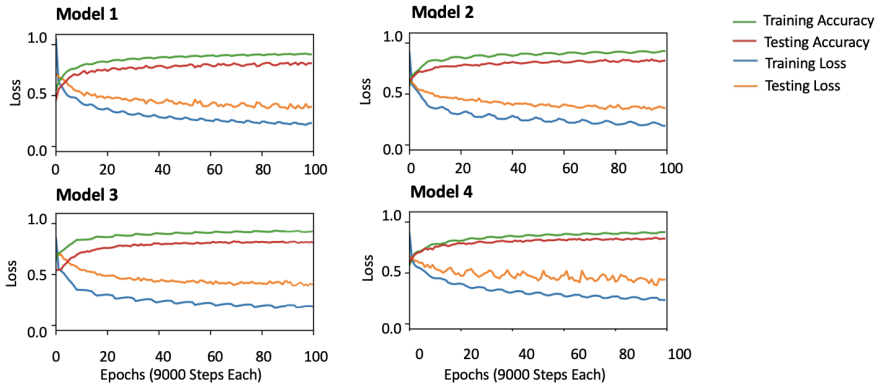

Figure 11: The training and testing accuracies and losses plotted over the epochs for each cross-validation submodel. The consistency of loss function between models demonstrates appropriate training.

In order to assess the validity of my model in performing the localization task, I compared the training results to another model, PhoStar [22], that identifies if a peptide, in general, has been phosphorylated (not the specific phosphosite). Since I used the same training dataset as PhoStar, I was able to compare the Bacc scores of my model and PhoStar. My model performed at an

equal or better rate compared to PhoStar on a variety of subsections of the PXD012174 dataset which is in Table 1.

Table 1: Selected spectra categories from the training dataset. The balanced accuracy scores of my model represent the accuracy of phosphosite localization whereas the balanced accuracy of PhoStar represents the accuracy of identifying a peptide as phosphorylated in general.

Name of Dataset in PXD012174	My Model (Bacc)	PhoStar (Bacc)
Colon	0.85	0.64
Muscle	0.97	0.96
Primary-Melanoma	0.94	0.89
Brain	0.82	0.63
Primary-Liver	0.90	0.68
HeLa	0.94	0.89
Primary-Gastro	0.96	0.77

The results of training provided assurance that my approach was valid for the phosphosite localization task.

4.2. Testing Results and Comparison with Sequence-Based Models

In testing, which included all eukaryotic organism datasets, my model achieved an average Bacc, F-1 Score, and ROC-AUC of 91.84, 94.85, and 94.89 respectively. These results are in Table 2. My model was able to produce effective accuracy across all organisms. Additionally, the datasets used varying fragmentation and MS options which demonstrates the model was not reliant on spectrometer-specific features.

Table 2: My model's testing validation for five additional datasets. The MS settings for the datasets are included. Fragmentation options include collision-induced dissociation (CID) or higher-energy C-Trap dissociation (HCD). Mass analyzer (MA) options are ion trap mass spectrometer (IMTS) or Fourier-transform mass spectrometer (FTMS). The collision energy (volts) is also listed. The average and standard deviation for each evaluation metric is included.

Dataset	Fragmentation/MA/CE	Bacc (SD = 1.16)	F1 Score (SD = 1.16)	ROC-AUC (SD = 2.37)
PXD015050 (Mouse)	CID/ITMS/35	93.23	95.02	96.45
JPST000685 (Liverwort)	CID/ITMS/35, HCD/FTMS/45	89.45	91.53	92.83
PXD039200 (Human)	HCD/FTMS/25	93.45	97.62	97.43
JPST000703 (Thale Cress)	CID/ITMS/35	87.65	92.21	91.32
PXD037724 (Human)	CID/ITMS/35	95.43	97.91	96.43
Average (SD)		**91.84**	**94.85**	**94.89**

While these results are generally high, there was a difference in the accuracy of the model in predicting phosphosites in the common liverwort and thale cress datasets. These differences were addressed in the iteration of the model with a more varied training dataset.

Comparison between my model and existing sequence models was somewhat difficult because the input data used (spectra and small amino acid sequences respectively) were different. I compared the F1 scores of my model and the scores of MusiteDeep and DeepPhos. Both of these models used exclusively human sequencing data so I only used the accuracies of my model on human datasets for a better comparison. The F1 score of my model across two human spectra datasets was 97.76 compared to the average F1 score of 76.16 for MusiteDeep and 78.84 for DeepPhos on 4 specific human kinase-based sequence datasets.

4.3. Neutral Loss and Reversible Phosphorylation

My model was able to accurately identify phosphosites given training and testing datasets with known neutral losses and phosphosites that were not phosphorylated on the specific spectra. The end-to-end nature of my model and its ability to make long-range associations most likely enabled this capability. Without relying on specific input parameters to identify phosphosites without the phosphate group bonded, the model was still able to develop and recognize phosphosites that had undergone neutral loss or were not currently phosphorylated. Furthermore, the model was able to make associations besides the typical β-/γ- ions manual feature typically used. Figure 12 demonstrates the ability of my model to recognize phosphosites from fragment intensities where neutral losses occurred on the peptide. The ability of the model to identify phosphosites with neutral losses presents may be able to be modified to create a model that can identify neutral losses in general.

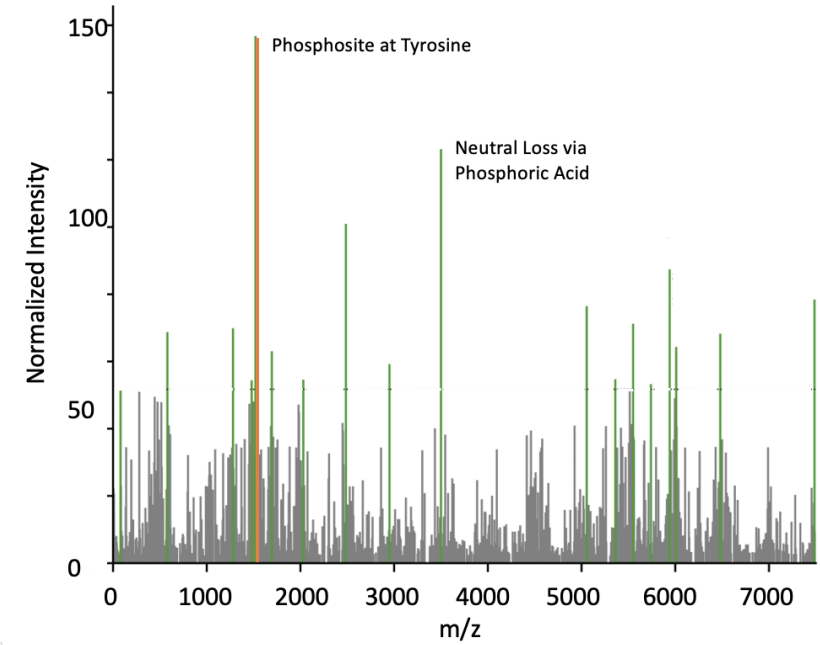

Figure 12: An individual run of the Primary-Gastro set of the PXD012174 dataset where the neutral losses were already annotated by Ochoa et. al shown in green in MSviewer. The orange intensity is an identified phosphosite made by my model. The identification of the phosphosite with a neutral loss demonstrates neutral losses have less of an impact on the model.

4.4. Varied Eukaryotic Training and Prokaryotic Model

After modifying my original model with both a varied eukaryotic training dataset version and a prokaryotic version, I recorded the accuracies. After shifting the training dataset to include spectra from all eukaryotic organisms, I retested my model. The accuracies of the thale cress and liverwort datasets did improve with no negative ramifications for the other eukaryotic organism datasets across all evaluation metrics when compared to the initial model's accuracies (Table 3).

Table 3: A table of the evaluation metrics for the initial model that had an exclusively human training set and the adjusted training set model. All eukaryotic organisms had increased accuracies except for humans which maintained the same scores.

Dataset	Bacc (Initial|Varied) (SD = 1.16)	F1 Score (Initial|Varied) (SD = 1.16)	ROC-AUC (Initial|Varied) (SD = 2.37)
PXD015050 (Mouse)	93.23|94.45	95.02|96.02	96.45|96.89
JPST000685 (Liverwort)	89.45|94.45	91.53|94.35	92.83|96.02
PXD039200 (Human)	93.45|93.45	97.62|97.62	97.43|97.43
JPST000703 (Thale Cress)	87.65|94.32	92.21|95.68	91.32|96.45
PXD037724 (Human)	95.43|95.43	97.91|97.43	96.43|96.43

The prokaryotic version of the model achieved adequate accuracies for phosphosite localization across all studied organisms. The accuracies, in general, are lower than that of eukaryotic organisms (Table 4). This may be due to there being more potential amino acids that may be phosphorylated which would necessitate more complicated pattern development and recognition. The model provides an important initial step given that prokaryotic phosphosite localization is even less rigorously studied than eukaryotic phosphosites.

Table 4: The evaluation metrics for the four prokaryotic organisms of the tested portion of the modified model. The accuracy in general of the prokaryotic model is less than that of both the initial and varied training eukaryotic models.

Dataset	Bacc (SD = 3.86)	F1 Score (SD = 2.60)	ROC-AUC (SD = 3.73)
PXD041015 (*K. pneumoniae*)	76.81	79.43	82.32
PXD040658 (*E. coli*)	68.35	74.32	73.55
PXD026981 (*S. aureus*)	72.83	76.41	77.22
PXD029827 (*C. difficile*)	78.32	81.02	82.43
Average	**74.07**	**77.80**	**78.88**

5. Conclusion

My eukaryotic model with varied training achieved an average accuracy of 95% compared to the accuracies (76 - 78%) of sequence-based models. This high accuracy enables my approach to be used as a critical next step for the effective development of medical and pharmacological tools. With the ability to accurately identify precise phosphosites from mass spectra, whole phosphoproteome patterns can be identified more easily for an improved understanding of general phosphorylation and specific biochemical pathways. Furthermore, by being able to potentially treat patients with diseases associated with abnormal phosphosites, treatment can be implemented at a more fundamental biological stage with the potential to reduce patient discomfort by avoiding only treating symptoms. While the accuracy of the prokaryotic model can be improved, it also provides a next step towards understanding phosphorylation at a full biological level and potentially manipulating phosphorylation patterns of bacteria harmful to humans for new antibiotics. Future steps for this research include improving upon the accuracy of the prokaryotic model and using the model's ability to overcome neutral losses to potentially create a new tool for efficiently identifying and understanding neutral losses in general. Therefore, my model provides an efficient and accurate approach to the identification of exact phosphosites from mass spectra that can be applied to a further understanding of phosphorylation and the fight against both infectious and chronic diseases.

6. References

[1] Geiszler, D. (2022). *Computational Methods for Characterizing Post-translational and Chemical Modifications Found in Open Searches* (Doctoral dissertation).

[2] *Overview of Post-Translational Modification - US*. (n.d.). Www.thermofisher.com. https://www.thermofisher.com/us/en/home/life-science/protein-biology/protein-biology-learning-center/protein-biology-resource-library/pierce-protein-methods/overview-post-translational-modification.html

[3] Olsen, J. V., Blagoev, B., Gnad, F., Macek, B., Kumar, C., Mortensen, P., & Mann, M. (2006). Global, in vivo, and site-specific phosphorylation dynamics in signaling networks. *Cell*, *127*(3), 635–648. https://doi.org/10.1016/j.cell.2006.09.026.

[4] Cohen, P. (2002). The origins of protein phosphorylation. *Nature cell biology*, *4*(5), E127-E130.

[5]Vlastaridis, P., Kyriakidou, P., Chaliotis, A., Van de Peer, Y., Oliver, S. G., & Amoutzias, G. D. (2017). Estimating the total number of phosphoproteins and phosphorylation sites in eukaryotic proteomes. *Gigascience*, *6*(2), giw015.

[6] Schmidl, S. R., Gronau, K., Pietack, N., Hecker, M., Becher, D., & Stülke, J. (2010). The phosphoproteome of the minimal bacterium Mycoplasma pneumoniae: analysis of the complete known Ser/Thr kinome suggests the existence of novel kinases. *Molecular & cellular proteomics: MCP*, 9(6), 1228–1242. https://doi.org/10.1074/mcp.M900267-MCP200

[7] Shahin Ramazi, Javad Zahiri, Post-translational modifications in proteins: resources, tools and prediction methods, *Database*, Volume 2021, 2021, baab012, https://doi.org/10.1093/database/baab012
[8] Johnson, L. N. (2009). The regulation of protein phosphorylation. *Biochemical Society Transactions*, *37*(4), 627-641.
[9]Lai, S. J., Tu, I. F., Wu, W. L., Yang, J. T., Luk, L. Y., Lai, M. C., ... & Wu, S. H. (2017). Site-specific His/Asp phosphoproteomic analysis of prokaryotes reveals putative targets for drug resistance. *BMC microbiology*, *17*(1), 1-10.
[10] Mijakovic, I., Grangeasse, C., & Turgay, K. (2016). Exploring the diversity of protein modifications: special bacterial phosphorylation systems. *FEMS microbiology reviews*, *40*(3), 398-417.
[11] Cohen P (2000) The regulation of protein function by multisite phosphorylation--a 25-year update. *Trends Biochem Sci* 25:596–601.
[12] Plun-Favreau, H., Lewis, P. A., Hardy, J., Martins, L. M., & Wood, N. W. (2010). Cancer and neurodegeneration: between the devil and the deep blue sea. *PLoS genetics*, 6(12), e1001257. https://doi.org/10.1371/journal.pgen.1001257.
[13] Ferguson, F. M., & Gray, N. S. (2018). Kinase inhibitors: the road ahead. *Nature reviews Drug discovery*, *17*(5), 353-377.
[14]Ficarro, S. B., McCleland, M. L., Stukenberg, P. T., Burke, D. J., Ross, M. M., Shabanowitz, J., Hunt, D. F., & White, F. M. (2002). Phosphoproteome analysis by mass spectrometry and its application to Saccharomyces cerevisiae. *Nature biotechnology*, 20(3), 301–305. https://doi.org/10.1038/nbt0302-301
[15] Potel, C. M., Lemeer, S., & Heck, A. J. R. (2019). Phosphopeptide Fragmentation and Site Localization by Mass Spectrometry: An Update. *Analytical chemistry*, 91(1), 126–141. https://doi.org/10.1021/acs.analchem.8b04746.
[16] Hunt, D. F., Yates 3rd, J., Shabanowitz, J., Winston, S. & Hauer, C. R. (1986). Protein sequencing by tandem mass spectrometry. *Proceedings of the National Academy of Sciences* 83, 6233- 6237.
[17]Yates III, J. R. (2013). The revolution and evolution of shotgun proteomics for large-scale proteome analysis. *Journal of the American Chemical Society*, *135*(5), 1629-1640.
[18]Mann, M., Ong, S. E., Grønborg, M., Steen, H., Jensen, O. N., & Pandey, A. (2002). Analysis of protein phosphorylation using mass spectrometry: deciphering the phosphoproteome. *Trends in biotechnology*, *20*(6), 261-268.
[19] Martin, D. B., Eng, J. K., Nesvizhskii, A. I., Gemmill, A., & Aebersold, R. (2005). Investigation of neutral loss during collision-induced dissociation of peptide ions. *Analytical chemistry*, 77(15), 4870–4882. https://doi.org/10.1021/ac050701k.
[20] Vizcaíno, J. A., Walzer, M., Jiménez, R. C., Bittremieux, W., Bouyssié, D., Carapito, C., ... & Kohlbacher, O. (2017). A community proposal to integrate proteomics activities in ELIXIR. *F1000Research*, *6*.

[21] Aebersold, R. & Mann. (2016). M. Mass-spectrometric exploration of proteome structure and function. *Nature* 537, 347–355.
[22] Dorl, S., Winkler, S., Mechtler, K., & Dorfer, V. (2018). PhoStar: identifying tandem mass spectra of phosphorylated peptides before database search. *Journal of proteome research*, *17*(1), 290-295.
[23] Seidler, J., Zinn, N., Boehm, M. E., & Lehmann, W. D. (2010). De novo sequencing of peptides by MS/MS. *Proteomics*, 10(4), 634–649. https://doi.org/10.1002/pmic.200900459
[24] Nesvizhskii A. I. (2007). Protein identification by tandem mass spectrometry and sequence database searching. *Methods in molecular biology* (Clifton, N.J.), 367, 87–119. https://doi.org/10.1385/1-59745-275-0:87.
[25] Zhou, X. X., Zeng, W. F., Chi, H., Luo, C., Liu, C., Zhan, J., ... & Zhang, Z. (2017). pDeep: predicting MS/MS spectra of peptides with deep learning. *Analytical chemistry*, *89*(23), 12690-12697.
[26] Gessulat, S., Schmidt, T., Zolg, D. P., Samaras, P., Schnatbaum, K., Zerweck, J., ... & Wilhelm, M. (2019). Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning. *Nature methods*, *16*(6), 509-518.
[27] Tran, N. H., Zhang, X., Xin, L., Shan, B., & Li, M. (2017). De novo peptide sequencing by deep learning. *Proceedings of the National Academy of Sciences*, *114*(31), 8247-8252.
[28] Tran, N. H., Qiao, R., Xin, L., Chen, X., Shan, B., & Li, M. (2020). Personalized deep learning of individual immunopeptidomes to identify neoantigens for cancer vaccines. *Nature Machine Intelligence*, *2*(12), 764-771..
[29] Qiao, R., Tran, N. H., Xin, L., Chen, X., Li, M., Shan, B., & Ghodsi, A. (2021). Computationally instrument-resolution-independent de novo peptide sequencing for high-resolution devices. *Nature Machine Intelligence*, *3*(5), 420-425.
[30] Xu, L. L., Young, A., Zhou, A., & Röst, H. L. (2020). Machine learning in mass spectrometric analysis of DIA data. *Proteomics*, *20*(21-22), 1900352.
[31] Altenburg, T., Giese, S. H., Wang, S., Muth, T., & Renard, B. Y. (2022). Ad hoc learning of peptide fragmentation from mass spectra enables an interpretable detection of phosphorylated and cross-linked peptides. Nature Machine Intelligence, 4(4), 378-388.
[32] Wen, B., Zeng, W. F., Liao, Y., Shi, Z., Savage, S. R., Jiang, W., & Zhang, B. (2020). Deep Learning in Proteomics. Proteomics, 20(21-22), e1900335. https://doi.org/10.1002/pmic.201900335.
[33] Wang, D., Zeng, S., Xu, C., Qiu, W., Liang, Y., Joshi, T., & Xu, D. (2017). MusiteDeep: a deep-learning framework for general and kinase-specific phosphorylation site prediction. *Bioinformatics*, *33*(24), 3909-3916.
[34] Luo, F., Wang, M., Liu, Y., Zhao, X. M., & Li, A. (2019). DeepPhos: prediction of protein phosphorylation sites with deep learning. Bioinformatics (Oxford, England), 35(16), 2766–2773. https://doi.org/10.1093/bioinformatics/bty1051.

[35] Jamal, S., Ali, W., Nagpal, P., Grover, A., & Grover, S. (2021). Predicting phosphorylation sites using machine learning by integrating the sequence, structure, and functional information of proteins. *Journal of Translational Medicine*, *19*(1), 1-11.
[36] Thapa, N., Chaudhari, M., Iannetta, A. A., White, C., Roy, K., Newman, R. H., ... & Kc, D. B. (2021). A deep learning based approach for prediction of Chlamydomonas reinhardtii phosphorylation sites. *Scientific Reports*, *11*(1), 1-12.
[37] Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016, October). Temporal convolutional networks: A unified approach to action segmentation. In European conference on computer vision (pp. 47-54). Springer, Cham.
[38] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.
[39] Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.
[40] Perez-Riverol Y, Bai J, Bandla C, García-Seisdedos D, Hewapathirana S, Kamatchinathan S, Kundu DJ, Prakash A, Frericks-Zipper A, Eisenacher M, Walzer M, Wang S, Brazma A, Vizcaíno JA. The PRIDE database resources in 2022: a hub for mass spectrometry-based proteomics evidences. Nucleic Acids Res. 2022 Jan 7;50(D1):D543-D552. doi: 10.1093/nar/gkab1038. PMID: 34723319; PMCID: PMC8728295.
[41] Okuda, S., Watanabe, Y., Moriya, Y., Kawano, S., Yamamoto, T., Matsumoto, M., ... & Ishihama, Y. (2017). jPOSTrepo: an international standard data repository for proteomes. *Nucleic acids research*, *45*(D1), D1107-D1111.
[42] Ochoa, D., Jarnuczak, A. F., Viéitez, C., Gehre, M., Soucheray, M., Mateus, A., ... & Beltrao, P. (2020). The functional landscape of the human phosphoproteome. *Nature biotechnology*, *38*(3), 365-373.
[43] Li, X., Sanagi, M., Lu, Y., Nomura, Y., Stolze, S. C., Yasuda, S., ... & Yamaguchi, J. (2020). Protein phosphorylation dynamics under carbon/nitrogen-nutrient stress and identification of a cell death-related receptor-like kinase in Arabidopsis. *Frontiers in plant science*, *11*, 377.
[44] Koide, E., Suetsugu, N., Iwano, M., Gotoh, E., Nomura, Y., Stolze, S. C., ... & Nishihama, R. (2020). Regulation of photosynthetic carbohydrate metabolism by a raf-like kinase in the liverwort Marchantia polymorpha. *Plant and Cell Physiology*, *61*(3), 631-643.
[45] Marx, H., Lemeer, S., Schliep, J. E., Matheron, L., Mohammed, S., Cox, J., ... & Kuster, B. (2013). A large synthetic peptide and phosphopeptide reference library for mass spectrometry–based proteomics. *Nature biotechnology*, *31*(6), 557-564.
[46] Fan, Y., Cheng, Y., Li, Y., Chen, B., Wang, Z., Wei, T., ... & Wang, L. (2020). Phosphoproteomic analysis of neonatal regenerative myocardium revealed important roles of checkpoint kinase 1 via activating mammalian target of rapamycin C1/ribosomal protein S6 kinase b-1 pathway. *Circulation*, *141*(19), 1554-1569.
[47] Raghuram, V., Salhadar, K., Limbutara, K., Park, E., Yang, C. R., & Knepper, M. A. (2020). Protein kinase A catalytic-α and catalytic-β proteins have nonredundant regulatory functions. *American Journal of Physiology-Renal Physiology*, *319*(5), F848-F862.

[48] Mergner, J., Frejno, M., List, M., Papacek, M., Chen, X., Chaudhary, A., ... & Kuster, B. (2020). Mass-spectrometry-based draft of the Arabidopsis proteome. *Nature*, *579*(7799), 409-414.
[49] Muselius, B., Sukumaran, A., Yeung, J., & Geddes-McAlister, J. (2020). Iron limitation in Klebsiella pneumoniae defines new roles for Lon protease in homeostasis and degradation by quantitative proteomics. Frontiers in Microbiology, 11, 546.
[50]Sukumaran, A., Pladwig, S., & Geddes-McAlister, J. (2021). Zinc limitation in Klebsiella pneumoniae profiled by quantitative proteomics influences transcriptional regulation and cation transporter-associated capsule production. *BMC microbiology*, *21*(1), 1-15.
[51] Prust, N., van Breugel, P. C., & Lemeer, S. (2022). Widespread Arginine Phosphorylation in Staphylococcus aureus. Molecular & Cellular Proteomics, 21(5).
[52] Garcia-Garcia, T., Douché, T., Gianetto, Q. G., Poncet, S., El Omrani, N., Smits, W. K., ... & Martin-Verstraete, I. (2022). In-depth characterization of the Clostridioides difficile phosphoproteome to identify Ser/Thr kinase substrates. Molecular & Cellular Proteomics, 21(11).

7. Appendix A

Table 5: A sample of the raw mgf files inputted into my model. The first number listed is the mass of the ion and the second number is the mass-to-charge ratio. The specific mass of the inputted peptide (found from the first round of mass analysis)

```
Unset
BEGIN IONS
PEPMASS=850.7927819146404
101.07134173652747 8116.2739257813
102.47219044333973 3009.556640625
104.0532072896122 21715.228515625
106.18246495595973 3256.3903808594
109.76244474004274 3149.3000488281
110.07161830775757 16287.4169921875
113.25425150481418 2967.927734375
123.34886581572484 2987.9130859375
127.83984119933304 2890.8212890625
129.06591576786647 4147.0751953125
129.10233847584374 77638.609375
130.06854048746015 4469.2143554688
130.08631699135356 68444.5390625
```

```
130.10549729555441 4272.7817382813
131.05059630254888 3106.2216796875
131.08152590932303 4287.8344726563
132.0767959273059 5171.1635742188
132.8649735999317 3571.4035644531
136.22572253599898 3115.4389648438
138.12791415261478 3706.7404785156
139.2822113054276 2892.2731933594
140.96900977486823 2849.830078125
143.0815743375597 5329.6591796875
143.11796654553027 4438.4443359375
145.3930524438169 3165.2937011719
146.6571519206785 3355.5895996094
147.11274892046282 108521.828125
148.11610604021692 4301.2348632813
148.79854028968288 3080.9194335938
155.11780127372006 9202.8505859375
155.72050830572428 3228.5920410156
```

Dataset.py: This file was the main driver code for separating raw mgf files from the PRIDE archive for training into the two-vector representation.

Python

```
from pyteomics import mgf
import tensorflow as tf
import numpy as np
import os,glob
AUTOTUNE = tf.data.experimental.AUTOTUNE

MZ_MIN=100
MZ_MAX=1900
SEGMENT_SIZE=0.5
k = 50

def set_k(new_k):
    global k
    k = new_k
    return k

def moduloparse(mz,intensity):
    dtype=tf.float32
    mz = tf.cast(mz,dtype)
    intensity = tf.cast(intensity,dtype)
    intensity = ion_current_normalize(intensity)

    greater_mask = tf.math.greater(mz,tf.zeros_like(mz)+MZ_MIN)
    # truncate
    smaller_mask = tf.math.less(mz,tf.zeros_like(mz)+MZ_MAX)
    # put into joint mask
    mask = tf.logical_and(greater_mask,smaller_mask)
    mask = tf.ensure_shape(mask,[None])
    # apply mask
    trunc_mz = tf.boolean_mask(mz,mask)
    trunc_intensity = tf.boolean_mask(intensity,mask)

    def segment_argmax(values,indices):
        i = tf.unique(indices)[0]
        zero,one=tf.zeros(1,dtype=dtype),tf.ones(1,dtype=dtype)
```

```
        return tf.vectorized_map(lambda x:
tf.argmax(values*tf.where(indices==x,zero,one)),i)

    mz_mod = tf.math.floormod(trunc_mz,SEGMENT_SIZE)
    mz_div =
tf.cast(tf.math.floordiv(trunc_mz-MZ_MIN,SEGMENT_SIZE),tf.int32)

    uniq_indices, i = tf.unique(mz_div)
    aggr_intensity = tf.math.segment_max(trunc_intensity,i)
    argmax_indices = segment_argmax(trunc_intensity,i)
    aggr_mz_mod = tf.gather(mz_mod,argmax_indices)

    aggr_intensity = aggr_intensity#/tf.reduce_sum(intensity**2)
    aggr_mz_mod = aggr_mz_mod/SEGMENT_SIZE

    shape = tf.constant([int((MZ_MAX-MZ_MIN)/SEGMENT_SIZE)])
    print(uniq_indices,aggr_intensity)
    print(aggr_mz_mod,aggr_intensity)
    aggr_mz_mod = tf.scatter_nd(tf.expand_dims(uniq_indices,1),
aggr_mz_mod, shape)
    aggr_intensity =
tf.scatter_nd(tf.expand_dims(uniq_indices,1), aggr_intensity,
shape)
    x = aggr_intensity
    i = aggr_mz_mod

    x = tf.cast(x,tf.float32)
    i = tf.cast(i,tf.float32)
    output = tf.stack([x,i],axis=1)
    return output

def tf_preprocess_spectrum(mz,intensity):
    #global MZ_MAX, SPECTRUM_RESOLUTION
    SPECTRUM_RESOLUTION=2
    n_spectrum = MZ_MAX * 10**SPECTRUM_RESOLUTION
    mz = mz*10**SPECTRUM_RESOLUTION
```

```
    indices = tf.math.floor(mz)
    indices = tf.cast(indices,tf.int64)

    uniq_indices, i = tf.unique(indices)
    # TODO: check optimum segmentation method (sum, max, mean)
    uniq_values = tf.math.segment_max(intensity,i)
    # make into mask to truncate between min<mz<max
    # eliminate zeros (reduce sparsity)
    lower_bound = 100 * 10**SPECTRUM_RESOLUTION
    notzero_mask =
tf.math.greater(uniq_indices,tf.zeros_like(uniq_indices)+lower_bo
und)
    # truncate :
    trunc_mask =
tf.math.less(uniq_indices,tf.zeros_like(uniq_indices)+n_spectrum)
    # put into joint mask:
    mask = tf.logical_and(notzero_mask,trunc_mask)
    # apply mask:
    uniq_indices = tf.boolean_mask(uniq_indices,mask)
    uniq_indices = uniq_indices - lower_bound
    uniq_values = tf.boolean_mask(uniq_values,mask)


    #### adapation because tf.SparseTensor only works with tuple
indices (stack zeros)
    zeros = tf.zeros_like(uniq_indices)
    uniq_indices_tuples = tf.stack([uniq_indices, zeros],axis =
1)
    sparse = tf.SparseTensor(indices = uniq_indices_tuples,
values = uniq_values,dense_shape = [n_spectrum-lower_bound,1])
    dense = tf.sparse.to_dense(sparse)
    return dense

def tf_maxpool(dense,k):
    shape = dense.shape
    dense = tf.reshape(dense,[1,-1,1,1])
```

```
    n_spectrum = int(shape[0])
    x, i =
tf.compat.v1.nn.max_pool_with_argmax(dense,[1,k,1,1],[1,k,1,1],pa
dding='SAME')
    i0 = tf.constant(np.arange(0,n_spectrum,k))
    i0 = tf.reshape(i0,[1,int(n_spectrum/k),1,1])
    i = i-i0
    x = tf.squeeze(x)
    i = tf.squeeze(i)
    return x,i

def tf_maxpool_with_argmax(dense,k):
    dense = tf.reshape(dense,[-1,k])
    x = tf.reduce_max(dense,axis=-1)
    i = tf.math.argmax(dense,axis=-1)
    return x,i

def ion_current_normalize(intensities):
    total_sum = tf.reduce_sum(intensities**2)
    normalized = intensities/total_sum
    return normalized

def standardize(intensities,global_mean,global_var,noise=False):
    log_intensity = tf.math.log(intensities)
    standardized = (log_intensity-global_mean)/global_var
    #return standardized
    return tf.math.exp(standardized)

def parse(mz,intensity):
    intensity = ion_current_normalize(intensity)
    dense = tf_preprocess_spectrum(mz, intensity)
    x,i = tf_maxpool_with_argmax(dense,k=k)
    x = tf.cast(x,tf.float32)
    i = tf.cast(i,tf.float32)
    # turn into logits (logit (p) = log(p/(1-p)
    i = i/tf.cast(k,tf.float32)
```

```
    output = tf.stack([x,i],axis=1)
    return output

def
get_dataset(dataset='train',maximum_steps=10000,batch_size=16,mod
e='training',weights=None):
    buffer_size=1*10**6 # in steps

    phos_path=[glob.glob('%s/*.phos.mgf'%(x)) for x in dataset]
    phos_path=[i for g in phos_path for i in g] # flatten
    other_path=[glob.glob('%s/*.other.mgf'%(x)) for x in dataset]
    other_path=[i for g in other_path for i in g] # flatten

    if mode=='training' or mode=='test':
        np.random.shuffle(phos_path)
        np.random.shuffle(other_path)

    def generator(label,reader):
        def get_features(entry):
            mz = entry['m/z array']
            intensities = entry['intensity array']
            return label,np.array(mz),np.array(intensities)
        try:
            entry = next(reader)
            yield get_features(entry)
        except:
            return

    with mgf.chain.from_iterable(phos_path) as phos_reader,
mgf.chain.from_iterable(other_path) as other_reader:

    phos_ds = tf.data.Dataset.from_generator(lambda:
generator(label=1.0,reader=phos_reader),output_types=(tf.float32,
tf.float32,tf.float32),output_shapes=((),None,None))
other_ds = tf.data.Dataset.from_generator(lambda:
```

```
generator(label=0.0,reader=other_reader),output_types=(tf.float32
,tf.float32,tf.float32),output_shapes=((),None,None))
        if mode=='training':
            drop_remainder=False
            ds =
tf.compat.v1.data.experimental.sample_from_datasets([phos_ds,othe
r_ds],weights)
            elif mode=='test':
            drop_remainder=False
            ds =
tf.compat.v1.data.experimental.sample_from_datasets([phos_ds,othe
r_ds],weights,seed=42)
            elif mode=='inference':
            drop_remainder=False
            ds = other_ds.concatenate(phos_ds)

        ### MAP & BATCH-REPEAT  ###
        ds = ds.map(lambda label,mz,intensities:
tuple(moduloparse(label,mz,intensities)),num_parallel_calls=AUTOT
UNE)
        if maximum_steps is None:
            ds = ds.repeat()
        else:
            ds = ds.repeat(int(maximum_steps/2))

        ### SHUFFLE ###
        if mode=='training':
            ds =
ds.shuffle(buffer_size=buffer_size,reshuffle_each_iteration=False
)
        elif mode=='test':
            ds =
ds.shuffle(buffer_size=buffer_size,reshuffle_each_iteration=False
,seed=42)

        ### CACHE & EPOCH-REPEAT  ###
```

```
        ds = ds.batch(batch_size,drop_remainder=drop_remainder)
    return ds

def get_dataset_inference(mgf_file='example.mgf',batch_size=16):

    def generator(label,reader):
        def get_features(entry):
            mz = entry['m/z array']
            intensities = entry['intensity array']
            scans = int(entry['params']['scans'])
            return scans,np.array(mz),np.array(intensities)
        try:
            entry = next(reader)
            yield get_features(entry)
        except:
            return

    with mgf.chain.from_iterable([mgf_file]) as mgf_reader:
        ds = tf.data.Dataset.from_generator(lambda:
generator(label=None,reader=mgf_reader),output_types=(tf.float32,
tf.float32,tf.float32),output_shapes=((),None,None))
        ### MAP & BATCH-REPEAT  ###
        ds = ds.map(lambda label,mz,intensities:
tuple(parse(label,mz,intensities)),num_parallel_calls=AUTOTUNE)
        ds = ds.repeat()

        ds = ds.batch(batch_size,drop_remainder=False)
    return ds

if __name__ == "__main__":
    for x,i in
get_dataset(dataset=['training'],maximum_steps=2,batch_size=1,mod
e='training'):
        print(x)
```

TCN.py: This file includes the Temporal Dilated Convolutional Neural Network Architecture that was created for the model.

Python

```
import tensorflow as tf

#establish temporal block class
class TemporalBlock(tf.keras.Model):
    def
__init__(self,filters,kernel_size,padding,dilation_rate,dropout_r
ate=0.0):
        super(TemporalBlock,self).__init__()
        init=tf.initializers.he_normal()
        #first convolution with assigned dilation rate
        self.conv1 =
tf.keras.layers.Conv1D(filters=filters,kernel_size=kernel_size,pa
dding=padding,dilation_rate=dilation_rate,kernel_initializer=init
)
        #ReLU activation
        self.ac1 = tf.keras.layers.Activation('relu')
        #Dropout Layer
        self.drop1 = tf.keras.layers.Dropout(dropout_rate)

        self.conv2 =
tf.keras.layers.Conv1D(filters=filters,kernel_size=kernel_size,pa
dding=padding,dilation_rate=dilation_rate,kernel_initializer=init
)
        self.ac2 = tf.keras.layers.Activation('relu')
        self.drop2 = tf.keras.layers.Dropout(dropout_rate)

        self.conv1x1 =
tf.keras.layers.Conv1D(filters=filters,kernel_size=1,padding='sam
e',kernel_initializer=init)
        self.ac1x1 = tf.keras.layers.Activation('relu')

    def call(self,x,training):
        prev_x = x
```

```
        x=self.conv1(x)
        x=self.ac1(x)
        x=self.drop1(x)

        x=self.conv2(x)
        x=self.ac2(x)
        x=self.drop2(x)

        prev_x = self.conv1x1(prev_x)
        return self.ac1x1(prev_x + x)
#implement temporal block class into full TCN architecture
class TCN(tf.keras.Model):
    def
__init__(self,num_channels,kernel_size=2,padding='causal',dropout
_rate=0.0):
        super(TCN,self).__init__()
        assert isinstance(num_channels, list)

        model = tf.keras.Sequential()

        num_levels = len(num_channels)
        for i in range(num_levels):
            dilation_rate = 2 ** i
            model.add(TemporalBlock(num_channels[i], kernel_size,
                                    padding=padding,
dilation_rate=dilation_rate, dropout_rate=dropout_rate))
        self.network = model

    def call(self, x, training):
        return self.network(x, training=training)
```

Training.py: This file ran the initial training of the complete model on the chosen training dataset.

```python
import tensorflow as tf
import numpy as np
import sys,os
#import simplejson
AUTOTUNE = tf.data.experimental.AUTOTUNE
tf.compat.v1.disable_eager_execution()
from dataset import get_dataset
from network import network

train = True
saving = False

def binary_accuracy(y_true, y_pred):
    return tf.reduce_mean(tf.cast(tf.math.equal(y_true,
tf.math.round(y_pred)),tf.float32))

callbacks = []

ch = 64
net =
network([ch,ch,ch,ch,ch,ch,ch,ch,ch,ch,ch,ch,ch],kernel_size=2,pa
dding='same',dropout=.2)

inp = tf.keras.layers.Input((3600,2))
sigm = net(inp)
model = tf.keras.Model(inputs=and,outputs=sign)
bce=tf.keras.losses.BinaryCrossentropy(from_logits=False)

learning_rate=5.0e-6
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=le
arning_rate,clipnorm=1.0),loss=bce,metrics=['binary_accuracy','Re
call','Precision'])

batch_size=64
```

```
steps=1
maximum_steps=batch_size*steps

train_data =
get_dataset(dataset=['./training'],maximum_steps=maximum_steps,ba
tch_size=batch_size,mode='training').prefetch(buffer_size=AUTOTUN
E)

if train:

model.fit(train_data,steps_per_epoch=1,epochs=1,callbacks=callbac
ks)

if saving:
    model.save_weights('model_weights.hdf5')
```

Inference.py: Testing file that took testing data and outputted prediction scores for phosphosite locations.

Python

```
import tensorflow as tf
import numpy as np
import sys,os
import argparse

parser = argparse.ArgumentParser(description='Read spectra from
an mgf-file and output a determined amino acid and phosphosite
prediction score.')
parser.add_argument('model_weights', type=str, help='trained
model weights')
parser.add_argument('mgf_file', type=str, help='input filename:
mgf-file containing ms/ms spectra.')
parser.add_argument('out_file', type=str, help='output filename')
parser.add_argument('--tsv', dest='tsv', action='store_const',
                    const=True, default=False, help='write a
tsv-file')

args = parser.parse_args()

AUTOTUNE = tf.data.experimental.AUTOTUNE
tf.compat.v1.disable_eager_execution()
from dataset import get_dataset_inference
from network import network

ch = 64
net =
network([ch,ch,ch,ch,ch,ch,ch,ch,ch,ch,ch,ch,ch],kernel_size=2,pa
dding='same',dropout=.2)
inp = tf.keras.layers.Input((3600,2))
sigm = net(inp)
model = tf.keras.Model(inputs=inp,outputs=sigm)
model.compile(optimizer=tf.keras.optimizers.Adam(),loss=tf.keras.
losses.BinaryCrossentropy(from_logits=False))
```

```
batch_size=64

model.load_weights(args.model_weights)

mgf_file = args.mgf_file


ds =
get_dataset_inference(mgf_file=mgf_file,batch_size=batch_size)pri
nt('predicting: ',mgf_file)

some_big_number=10**5
pred_score = model.predict(ds,steps=some_big_number)
score = np.atleast_1d(np.squeeze(pred_score))
pred = np.round(score)

print('Writing results to file: %s'%args.out_file)

if not args.tsv:
    np.savetxt(args.out_file,score)
if args.tsv:
    from pyteomics import mgf
    import pandas as pd
    def get_scans(entry):
        title = str(entry['params']['title'])
        scans = int(entry['params']['scans'])
        return title,scans
    title_,scan_ = zip(*list(map(get_scans,mgf.read(mgf_file))))

    ###
    df =
pd.DataFrame({'title':title_,'scan':scan_,'score':score,'pred':pr
ed})
    df.to_csv(args.out_file,sep='\t')

print('Done.')
```