Phosphorylation Prediction Using Convolutional Neural Networks for Post Translational Modifications

Christopher Chow

Mentor: Dr. Shang-Huan Chiu

New York City College of Technology

Abstract

Post-translational modifications (PTMs) refer to the attachment of molecules (e.g., phosphates, sulfates, glycans, ubiquitins, etc.) to amino acids in proteins. These modifications change the properties of proteins and play a large role in protein function, regulation, and cell signaling. PTMs also play a role in disease (e.g., Alzheimer's disease via Tau hyperphosphorylation, Huntington's disease via hypopalmitoylation of huntingtin protein, and type 2 diabetes mellitus via malonylation of enzymes involved in glucose and lipid metabolism); therefore, understanding PTMs and locating these modifications can provide valuable targets for therapeutics. However, discovery of PTMs and their locations on proteins is time consuming, laborious, and expensive. Computational methods, such as machine learning, are required to alleviate these challenges. Logistic regression, support vector machines, random forests, and k-nearest neighbors are some traditional machine learning methods that were used with good success in predicting PTMs. As machine learning and AI have advanced tremendously in the last few years, advanced machine learning techniques such as Convolutional Neural Networks(CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Transformer Models, and Graph Neural Networks (GNNs) have been used and outperformed traditional machine learning methods. This work will explore the use of a CNN to predict phosphorylation sites using the Pytorch package in Python and data from UniProt. Structural data, attention heads, PTM crosstalk, and bootstrapping will be considered in future work to improve the model.

Introduction

Post-translational modifications (PTMs) are responsible for many functional changes in proteins and cellular activity, such as phosphorylation – where a phosphate group from ATP is added to amino acid side chains (most commonly serine, threonine, and tyrosine) with the help of kinases [1]. PTMs are also involved in various diseases which can be potential targets for therapeutics. For example, the hyperphosphorylation of tau protein is considered to be a contributing factor to the development of Alzheimer's disease and is currently in active research to modulate this as a therapeutic [2]. Therefore, it is valuable to develop a method to predict PTMs. Utilizing computational deep learning techniques, such as CNNs, is a time and cost effective way of predicting PTMs.

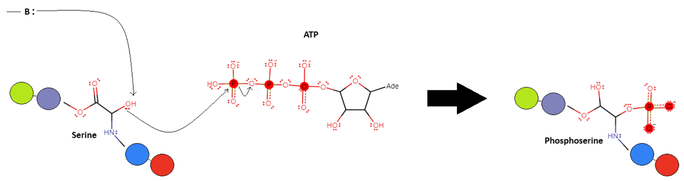


Figure 1: A diagram of a phosphorylation process on serine in a protein. A basic residue (–B:) from a kinase removes a proton from the hydroxyl group on serine to initialize a nucleophilic attack on the outer most phosphate in ATP, resulting in a phosphoserine residue and ADP (not shown).

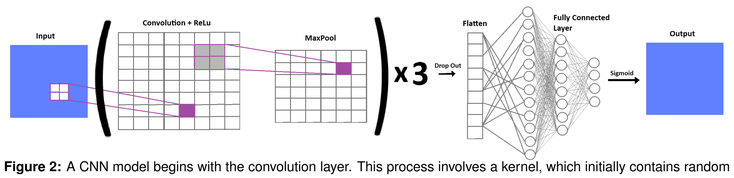


Figure 2: A CNN model begins with the convolution layer. This process involves a kernel, which initially contains random weights, that slides along the entire input. At each step, the dot product between the kernel weights and the corresponding overlapping input values are calculated. A ReLu activation function, which turns every negative number in the matrix to zero, is applied after the convolution then max pooling. Max pooling returns the largest value in the window and results in a smaller matrix that keeps the most important features. This method is repeated three times. Next, a portion of neurons are dropped out to prevent overfitting the model and then flattened to be inputted in the fully connected layer to predict the phosphorylation sites. Loss and gradient are computed to conduct back propagation to adjust the weights in the model to improve prediction performance.

Numerical Method

Phosphorylation and sequence data for human proteins were collected from Uniprot. A sequence of length 33 enveloping a phosphorylation site was randomly obtained for each site in the sequence and one hot encoded resulting in a 33 by 20 matrix. The columns represent 20 different types of amino acids and the rows represent the amino acid position. The corresponding phosphorylation site data was also one hot encoded in a 1 by 33 vector. The model was trained on 18,551 samples and evaluated with 4,415 testing samples. The CNN model was created with Python using the Pytorch library. The initial model was composed of three one- dimensional convolutions, ReLu activation function, and one-dimensional max pooling were used. The kernel size of convolutions were 3, 11, and 3. Next, drop out of 45% was used to help prevent overfitting and the resulting output was flattened and sent through a 3 layer fully connected layer. The result was sent through the BCEWithLogitsLoss function with a class weight of 6 which was chosen because of the ability to change class weights to ameliorate the class bias nature of the data and improve model performance. Accuracy, precision, recall, f1-score, and AUROC were calculated to evaluate model performance.

$$\text{Precision} = \frac{TP}{TP+FP}, \qquad \text{Recall} = \frac{TP}{TP+FN}, \qquad F_1 = \frac{2TP}{2TP+FP+FN}$$

where TP is true positives, FP is false positives, and FN is false negatives. Precision measures the accuracy of the model's prediction, recall measures how many positive predictions made by the model were correct, and a high f1-score (close to 1) indicates that the model is accurately predicting sites with low false positives and false negatives. F1-score will be the main focus in evaluating model performance.

Numerical Results and Discussion

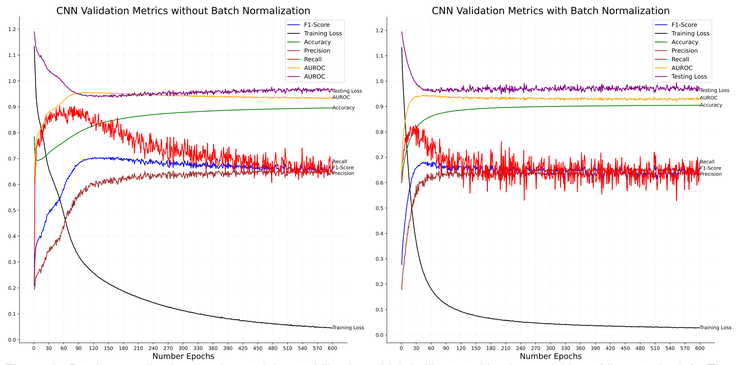


Figure 3: Batch normalization sped up training stabilization which is illustrated by the squishing of lines to the left. The model with and without batch normalization have testing loss of 0.9627 and 0.9602, and training loss of 0.0277 and 0.0452 at the last epoch, respectively. The large difference between testing and training loss suggests that the current model is over fitted. Batch normalization also appears to slightly worsen model performance with f1-score of 0.6561 and 0.6638 for with and without batch normalization, respectively.

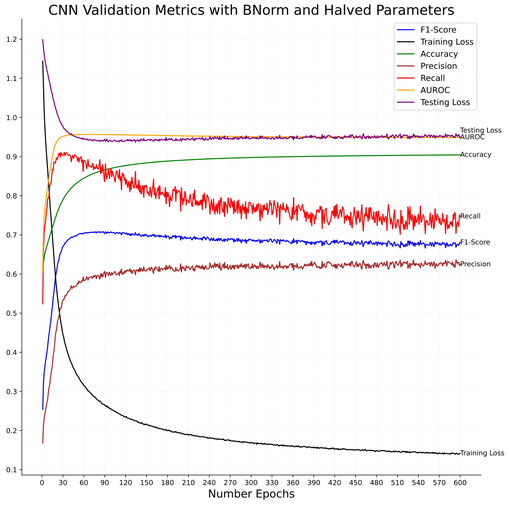
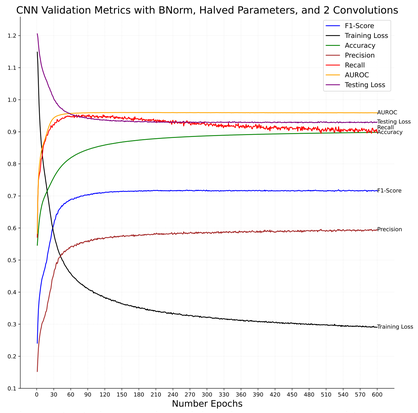


Figure 4: The training and testing loss at the last epoch was 0.1414 and 0.9512, respectively. The lower testing loss after halving the parameters in the model indicates that it is less over fitted, however, the problem still persists. The model has a f1-score of 0.6803 from 0.6561 which indicates an improvement in model performance and suggests that the model was overly complex.



Drop Out	Testing Loss	Training Loss	F1-Score
0.45	0.9512	0.1414	0.6803
0.50	0.9552	0.1544	0.6748
0.55	0.9502	0.1685	0.6843
0.60	0.9486	0.1843	0.6887
0.65	0.9452	0.1977	0.6893
0.70	0.9454	0.2080	0.6896
0.75	0.9470	0.2326	0.6904

Figure 5: Data was obtained after 600 epochs. As drop out rate increased, model performance appeared to improve which was illustrated by an increasing f1-score. Testing loss also decreased which showed that the model was less overfitted. The drop out rate of 0.65 procured the lowest testing loss which will be used for a model with one less convolution to further reduce overfitting. The testing and training loss of the new model were 0.9305 and 0.2910, respectively. The testing loss decreased slightly from 0.9452 along with an increase in training loss from 0.1977. Since the gap between training and testing loss has reduced, overfitting as been reduced. Model performance has also increased with an f1-score of 0.7155 from 0.6896.

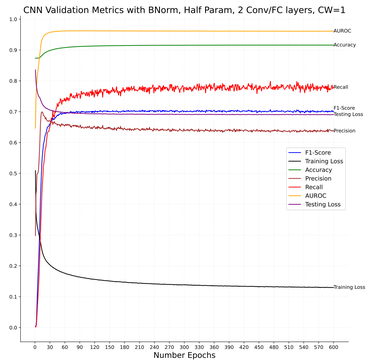


Figure 6: The last fully connected layer was removed, class weight of 1 was used as it procured the best metrics, and data was collected for 600 epochs. Training and testing loss was 0.1300 and 0.6907, respectively. The reduction of class weight resulted in the biggest reduction of testing error compared to removing layers in the model and reduced overfitting. The f1-score was 0.7010 which indicates that this model has similar predictive performance to the model measured in figure 5. The slightly lower f1-score of this model suggests that higher class weights for models trained on class imbalanced data yield better predictive performance for the minority class.

- Due to the large class discrepancy in the data, solving the overfitting problem is difficult, however, drop out, reducing convolution and/or fully connected layers, and optimizing class weights are effective methods of mitigation.
- Batch normalization improves training stabilization which speeds up training but does not help with overfitting.
- Incorporating more data about the proteins (e.g amino acid properties, structural data, evolutionary information, protein function, PTM cross talk) can improve the class bias issue in the data set and overfitting problem which can lead to improved model performance.
- Incorporating a hybrid model approach, such as an CNN+LSTM model, and full sequence data can allow learning of long range interactions between residues that occur in nature.

References

[1] Zhong Q, Xiao X, Qiu Y, Xu Z, Chen C, Chong B, Zhao X, Hai S, Li S, An Z, Dai L. Protein posttranslational modifications in health and diseases: Functions, regulatory mechanisms, and therapeutic implications. *MedComm (2020).*, 2023 May 2;4(3):e261.doi: 10.1002/mco2.261. PMID: 37143582; PMCID: PMC10152985.

[2] Basheer, N., Smolek, T., Hassan, I. et al. Does modulation of tau hyperphosphorylation represent a reasonable therapeutic strategy for Alzheimer's disease? From preclinical studies to the clinical trials. *Mol Psychiatry 28,*, 2197–2214 (2023). https://doi.org/10.1038/s41380-023-02113-z