SUMONET: DEEP SEQUENTIAL PREDICTION OF SUMOYLATION SITES

by
BERKE DILEKOGLU

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
JULY 2022

ABSTRACT

SUMONET: DEEP SEQUENTIAL PREDICTION OF SUMOYLATION SITES

BERKE DILEKOGLU

COMPUTER SCIENCE ENGINEERING MSc. THESIS, FEB 2022

Thesis Supervisor: Dr. Oznur Tastan

Keywords: Deep sequential learning, SUMOylation, Post-translational Modifications, CNNs, Transformers

SUMOylation is a reversible post-translational protein modification in which SUMOs (small ubiquitin-like modifiers) covalently attach to a specific lysine residue of the target protein. This process is vital for many cellular events such as protein binding, subcellular transport, DNA repair, and cellular signaling. Aberrant SUMOylation is linked with several diseases, including Alzheimer's, cancer, and diabetes. Therefore, accurate identification of SUMOylation sites is essential to understanding cellular processes and pathologies that arise with their disruption. In this thesis, we present three deep neural architectures, SUMOnets, that take the peptide sequence centered on the candidate SUMOlylation site as input and predict whether the lysine could be SUMOylated. Each of these models, SUMOnet-1, -2 and -3, relies on different compositions of deep sequential learning architectural units, such as Bidirectional Gated Recurrent Units(biGRUs) and convolutional layers. We evaluate these models on the benchmark dataset with three different input peptide representations of the input sequence. SUMOnet-3 achieves 75.8% AUPR and 87% AUC scores, corresponding to approximately 5% improvement over the closest state-of-the-art SUMOylation predictor. We also create a challenging subset of the test data based on the absence and presence of known SUMOylation motifs. Even though the performances of all methods degrade in these cases, SUMOnet-3 remains the best predictor in these challenging cases, and the current methods' predictive abilities decrease significantly. The SUMOnet-3 framework is available as an open source project and a Python library at `https://github.com/berkedilekoglu/SUMOnet`.

ÖZET

SUMONET:SUMOLANMA BÖLGELERININ DERIN SIRALI ÖĞRENME ILE TAHMINI

BERKE DİLEKOĞLU

BİLGİSAYAR BİLİMİ MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, ŞUBAT 2022

Tez Danışmanı: Dr. Öznur Taştan

Anahtar Kelimeler: Makine Öğrenmesi, SUMOylation, PTM

SUMOlanma, SUMO'ların (küçük ubikuitin benzeri değiştiriciler) hedef proteinin spesifik bir lizin aminoasidine kovalent olarak bağlandığı, tersine çevrilebilir protein çeviri sonrası modifikasyonudur. SUMOlanma, hücre içi taşıma, DNA onarımı ve hücresel sinyalleşme gibi birçok hücresel olay için önemlidir. SUMOlanma sürecindeki bozukluklar, Alzeimer, kanser ve diyabet dahil olmak üzere çeşitli hastalıklarla bağlantılıdır. Bu nedenle, SUMOlanma bölgelerinin doğru tanımlanması, hücresel süreçleri ve onların aksaması sonucu ortaya çıkan patolojileri anlamak için elzemdir. Bu tezde, peptit dizisini girdi olarak alıp, bu bölgenin SUMOlanıp, SUMOlanmayacağını tahmin eden üç derin öğrenme mimari, SUMOnets, sunuyoruz. SUMOnet-1, -2 ve -3 adını verdiğimiz modellerin her biri biGRU'lar ve CNN'ler gibi derin sıralı öğrenme mimari birimlerinin farklı bileşimine dayanır. Girdi peptid dizilerin farklı gösterimleri ile bu modelleri eğitip, kıyaslama verisinde değerlendirdik. SUMOnet-3 %75,8 AUPR ve %87 AUC sonucu ile en iyi tahmin edici oldu ve bu performans değerleri literatürdeki, en iyi SUMOlaşma tahmini araçlarından yaklaşık %5'lik iyileşmeye denk geliyor. Ayrıca bilinen SUMOlanma motiflerinin var olup olmadığına göre oluşturulan, zor sınama kümesinde, ayrıca bir değerlendirme yaptık. Bu kümede tüm yöntemlerin performansı düşerken, SUMOnet-3 hala bu zorlu durumlarda en iyi tahmin edici olarak performans gösterdi ve literatürdeki diğer yöntemlerin performansı ise ciddi olarak düşüş gösterdi. SUMOnet-3 açık kaynak projesi ve bir Python kütüphanesi olarak `https://github.com/berkedilekoglu/SUMOnet.` adresinde mevcuttur.

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my graduate advisor and thesis supervisor, Asst. Prof. Dr. Oznur Taştan Okan for her unlimited support, kind attitude, understanding, and deep knowledge. Her trust in me has been my most motivating factor throughout my thesis process. In this challenging journey, you may encounter unexpected difficulties from time to time. In times like these, you need a light to guide you in order not to lose your way. Thanks to her precious guidance, I am where I am now.

I also would like to thank the members of TastanLAB.

Moreover, I would like to thank Büşra Öz and Rana Kalkan for their friendship. I wouldn't have been able to handle the stress I've been through without them.

Finally, I would like to thank my sister Nehir Dilekoğlu, my father Kaan Dilekoğlu, and my mother Selda Dilekoğlu for their endless love and support throughout my graduate studies. Knowing that your family is behind you during difficult times is priceless.

Dedicated
to those who walk alone through complicated paths of life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Protein post-translational modifications (PTMs) are chemical alterations that occur during or after protein synthesis. These changes are often mediated by enzymes and may involve adding functional chemical groups covalently to one or more amino acids on the protein, proteolytic cleavage of regulatory subunits, or degradation of the whole protein. Most proteins undergo post-translational modification (PTMs) throughout their lifetimes (Walsh, Garneau-Tsodikova & Gatto Jr, 2005). Almost all cellular functions are regulated by PTMs that dynamically respond to extracellular and intracellular stimuli (Bode & Dong, 2004). PTMs modulate the modified proteins' functions through altering their targets' structure, subcellular locations and/or their interactions with other proteins, lipids, or nucleic acids(Walsh et al., 2005). Therefore, understanding PTMs are critical for a fundamental understanding of cell biology.

SUMOylation is one of the most critical PTM, in which small ubiquitin like modifiers (SUMOs) covalently attach to specific lysine (K) residues of target proteins in a reversible manner (Celen & Sahin, 2020). SUMO proteins are ubiquitously expressed in eukaryotes and are highly conserved, indicating their functional importance Geiss-Friedlander & Melchior (2007). SUMO was initially characterized for its role in binding nuclear proteins (Mahajan, Delphin, Guan, Gerace & Melchior, 1997; Matunis, Coutavas & Blobel, 1996), later its wide range of activities are discovered in transcription regulation, chromatin remodeling, DNA repair, and the control of cell cycle progression (Celen & Sahin, 2020; Flotho & Melchior, 2013; Jackson & Durocher, 2013). Due to its critical role in the regulation of cell cycle and cellular responses to stress conditions, alteration in SUMOylation has been also associated with pathogenesis. Relevant examples include neurodegeneration diseases such as Alzheimer's (Lee, Sakurai, Matsuzaki, Arancio & Fraser, 2013), cancer (Seeler & Dejean, 2017), and several autoimmune diseases such as type I diabetes (Zhang, Chen, Zhou, Yang & Wang, 2017).

SUMO covalently attaches to its target through a series of reactions facilitated by SUMO-activating enzymes 1 (E1), SUMO-conjugating enzyme E2 (Ubc9), and var-

ious SUMO E3 ligases. SUMO-specific proteases cleave the bond between SUMO and its substrate to reverse the SUMOlylation process (Geiss-Friedlander & Melchior, 2007). Mass spectrometry (MS)-based proteomics studies allow the identification of SUMOylated proteins (Filosa, Barabino & Bachi, 2013; Golebiowski, Matic, Tatham, Cole, Yin, Nakamura, Cox, Barton, Mann & Hay, 2009; Vertegaal, Andersen, Ogg, Hay, Mann & Lamond, 2006) and SUMOylation sites (Hendriks, D'souza, Yang, Verlaan-de Vries, Mann & Vertegaal, 2014; Tammsalu, Matic, Jaffray, Ibrahim, Tatham & Hay, 2015) in a high-throughput fashion. However, the transient nature of SUMOylation and the small fraction of the SUMOylated protein sets a limitation to uncovering all SUMOylated proteins.

To assist experimental efforts, identifying whether a protein is SUMOylated or not has also been formulated as a computational problem and several methods have been proposed to tackle this problem. Early methods made use of common sequence motifs that are commonly observed around the SUMOylation accepting lysine residue. Later approaches provide more sophisticated techniques, including machine learning approaches, as reviewed in Sec. 2.2.

Although several SUMOylation predictors are widely adapted, they do not yield high prediction performance - especially on challenging cases, for example SUMOylation sites that lack a known SUMO motif or non-SUMOylation sites that include those motifs. Also, the comparisons of many of these methods do not follow machine learning standards; they are not trained on the same data. Finally, many tools fail to provide working open source implementations. In this work, we aim to provide a state-of-the-art SUMOylation predictor that, given a peptide sequence, would predict whether the lysine residue centered on the peptide could be SUMOylated or not. Our method is based on a deep sequential learning approach that shows significant progress in Natural language processing tasks (NLP). We evaluate this tool rigorously with other tools on a recently available benchmark SUMOylation dataset.

The brief summary of contributions of this thesis in bioinformatics are as follows:

- We developed three deep learning architectures, SUMOnet-1, -2 and -3, for SUMOylation prediction, which outperform existing methods on a benchmark dataset. The performance difference is especially pronounced in hard cases. These architectures utilize fundamental sequential data processing units, such as a convolutional layer, GRUs, and LSTMs.
- We experimented with three different representations of the input protein sequences. We evaluate each SUMOnet and compare machine learning methods

with these three different encoding techniques.

- We evaluated the proposed SUMOnets with other state-of-the-art machine learning methods such as gradient boosted trees or logistic regression and other SUMOlyation predictors on a benchmark dataset rigorously.
- We provide the best performing architecture, SUMOnet-3, as an open source tool on GitHub (https://github.com/berkedilekoglu/SUMOnet) and as a Python library that can be easily installed via pypi 'pip install sumonet==0.1'.

The remainder of this thesis is organized as follows:

- Chapter 2 provides a review of related work in the literature. We review computational methods that are used to classify SUMOylation sites.
- Chapter 3 presents the encoding methods and the representation of amino acid sequences in a vector space are described. Secondly, we mention about scaling algorithm that is used in our experiments. Finally, SUMOnet components are briefly reviewed.
- Chapter 4 first presents the dataset used in building and evaluating SUMOnets. Secondly, we present the overview of three sequential models SUMOnet-1, -2, and -3 input. Lastly, we provide the details of hyperparameter tuning and the evaluation setup.
- Chapter 5 presents the predictive performance results of experiments conducted with the novel architecture we develop in this study. We also present the performance of our proposed model on hard test samples, which are based on the presence and absence of the SUMOlylation motifs.
- In Chapter 6, we present the Python library `sumonet`. `sumonet` is an open-source project and a module that includes the pre-trained SUMOnet-3 architecture and necessary pre-processing steps for replicating our experiments. A detailed explanation of the library and each module can be found in this chapter.
- In Chapter 7, we highlight the main findings of our experiments and compare our results with the literature . We finally present possible future direction to expand the work presented herein.

2. LITERATURE REVIEW

This chapter summarizes the related work on SUMOylation prediction. Many SUMOylation site prediction algortihms rely on the known SUMOylation motifs or alignments of experimentally validated SUMOylation sites. A set of approaches relies on the similarity of the candidate sequence to the other SUMOYlated sequences. The final category casts the problem as a classification task. In this chapter we review the available SUMOylation prediction tools.

2.1 Cluster Based Scoring Methods

Early studies reported that the SUMOylated sites follow a sequence motif, which is of the form: ψ–K–X–E (ψ: a hydrophobic amino acid: A, I, L, M, P, F, V or W; X: any amino acid) (Denison, Rudner, Gerber, Bakalarski, Moazed & Gygi, 2005; Sampson, Wang & Matunis, 2001). Initially, SUMOylation sites was determined based on motif match around the candidate sequence. However, classifying all candidates with motifs as SUMOylated yields high false positive results because most of the protein sites which include consensus sequence are not SUMOylated. Similarly, since there exist SUMOylation sites that do not match this motif, these techniques also miss true positives. Thus, relying solely on motifs yields both low recall and precision, which we show in the Chapter 5.

2.1.1 Motif extraction and scoring algorithms

Positive SUMOylation sites may contain different motifs (Beauclair et al., 2015; Ren, Gao, Jin, Zhu, Wang, Shaw, Wen, Yao & Xue, 2009; Xue, Zhou, Fu, Xu & Yao,

2006; Zhao, Xie, Zheng, Jiang, Liu, Mu, Liu, Zhao, Xue & Ren, 2014). Motif based SUMOylation site prediction methods extract sequence motifs in SUMOylation data. Then, a clustering strategy is combined with a scoring algorithm to predict whether the given sequence is SUMOylated or not. Below, we explain the most commonly used motif extraction algorithm *MotifX* (Schwartz & Gygi, 2005) and the scoring algorithm *Group-based phosphorylation site prediction and scoring (GPS)* (Zhou, Xue, Chen & Yao, 2004). Originally, MotifX and GPS algorithms are used to predict phosphorylation sites, due to similarity of the problem, they are adapted for SUMOylation prediction task (Xue et al., 2006).

MotifX (Schwartz & Gygi, 2005) is a an statistical method that identifies sequence motifs from protein phosphorylation datasets. The algorithm first asses the significance of the each residue/position pair and then builds motifs by finding successive significant residue/position pairs. To calculate the significant residue/position pairs, the probability of observing and amino acid more is accessed based on the random background. The method uses two peptide data sets. The first one is the phosphorylated peptide datasets, on this one the observed frequencies of each residues are extracted. The second one is a peptide data set that is used for background probability calculation. In each dataset six residues flanking upstream and downstream of the phosphorylation site are considered. In the ensuing step, position specific weight matrices are calculated on each dataset, where each matrix contains information on the frequency of amino acids at each position around the phosphosite. Using the these two matrices, the binomial probability matrix is calculated as follows:

$$P(m, c_{xj}, p_{xj}) = \sum_{i=c_{xj}}^{m} \binom{m}{i} {p_{xj}}^{i} (1 - p_{xj})^{m-i} \tag{2.1}$$

m represents the number of protein sequences in the dataset, the frequency of residue x at position j in the dataset is represented by c_{xj}, and p_{xj} represents fractional percentage of residue x at position j in the current background matrix. After calculation of binomial probabilities of possible residues, most significant motifs are selected according to the user defined threshold 10^{-16}. The score of each motif is determined by the summation of negative log probabilities for each residue/position of the motif. Finally, the given peptide sequence is predicted as the phosphorylation site, if a significant motif is found in it.

Group-based phosphorylation site prediction and scoring (GPS 1.0) algorithm (Zhou et al., 2004) uses amino acid substitution matrix to calculate similarity between known phosphorylation sites and a given peptide sequence. The algorithm

three upstream and three downstream amino acids around the phosphosite. Their method is based on that if a given peptide sequence differ from a known phosphorylated peptide in one residue and the pair of different amino acids has similar biochemical properties site, there is a high probability that the given sequence is also phosphorylated by the same kinase. To asses the biochemical similarity of the differing amino acid pair, they use BLOSUM62 substitution matrix (Henikoff & Henikoff, 1992).

The similarity between P_1 and P_2 is calculated by Equation 2.2

$$S(P_1, P_2) = \sum_{1 \leq i \leq 7} Score(P_1[i], P_2[i]) \tag{2.2}$$

P_1 and P_2 are the two peptide sequences with length 7. Score(P_1[i],P_2[i]) is calculated with respect to BLOSUM62 substitution matrix and negative values are converted to 0. After calculating the similarity score, they transform it to the distance metric by calculating: $D(P_1, P_2) = 1/S(P_1, P_2)$. Thus, since the similarity between two sequences get closer to 0, the distance between them will near to ∞.

The peptide sequences are then clustered using a graph method. The peptide sequences form the nodes of the undirected weighted graph, the edge weights are the distances between the corresponding peptides in the connecting nodes. The clusters are determined by using Markov Cluster Algorithm (Dongen, 2000). *Group-based phosphorylation scoring algorithm (GPS)* measures the similarity between given peptide sequence and clusters to determine potential phosphorylation sites. The algorithm decides whether the given peptide sequence is a phosphorylated or not with respect to user defined threshold.

GPS 2.0 algorithm (Xue, Ren, Gao, Jin, Wen & Yao, 2008) is an improved version of GPS 1.0. Only 11% of the kinase groups were divided into different clusters by using graph partitioning in GPS 1.0. Therefore, this strategy was removed in GPS 2.0. In addition to that, matrix mutation method is used to improve prediction performance in the new version.

In the MaM algorithm, value from BLOSUM62 matrix is randomly chosen and added +1 or -1 with respect to *Sn* as can be seen in algorithm 2. This process continues until the sensitivity decreases. The *Sn* is calculated for each instance by

Algorithm 1 Group-based phosphorylation scoring algorithm

for Each cluster C_i in C **do**

for Each peptide P_j in C_i **do**

Calculate S(Seq, P_j)

end for

$S_i = [\sum_j S(\text{Seq,P}_j)]/C_i$

end for

$S(Seq) = \max_j S_i$

Return $S(Seq)$

Algorithm 2 Matrix mutation algorithm

Initialization of substitution matrix with BLOSUM62
$time \leftarrow 0$
for $time \leq$10000 **do**
for R **do**andomly pick element from substitution matrix

Add +1 or -1 to that element

Calculate Sn score of leave-one-out with the mutated matrix

if Sn increase **then**
Keep the mutation
else
Continue
end if
Calculate time
end for

Return mutated matrix

leave-one-out cross validation, which takes an instance as test data and all other samples as training data. In GPS 2.0, firstly the peptide sequence is taken with PSP(7,7), 7 upstream and 7 downstream amino acids, and then the similarity score between a given peptide sequence and all known sites are calculated by Equation 2.2 without using clusters. Finally, algorithm makes a prediction with respect to the threshold value. This modifications decrease computational time and improve the robustness of prediction method.

GPS 3.0 algorithm (Xue, Liu, Gao, Jin, Wen, Yao & Ren, 2010) is an enhanced version of GPS 2.0. The fundamental hypothesis in GPS algorithm is that, similar peptide sequences have similar biochemical features. In GPS 3.0, four components - k-means clustering, peptide selection (PS), weight training (WT) - matrix mutation (MaM) are used to improve prediction performance.

K-means clustering is used to cluster training data into several groups. K is selected as 3 because of the high computational time. The clustering strategy is based on similarity measurement by Equation 2.3.

$$s(P_1, P_2) = \frac{\#\text{ Conserved Substitutions}}{\#\text{ All Substitutions}} \tag{2.3}$$

The similarity score $Score(a, b) > 0$, which is calculated by Equation 2.2, is a conserved substitution. PSP(7,7) is used to measure similarity between peptide sequences. Firstly, randomly chosen three positive sites are made as centroids of different clusters. Secondly, the similarity between each positive site, which was not chosen as centroid, and centroids is measured. Therefore, a peptide site can be put into the most similar cluster. Finally, the centroids are updated with respect to highest average similarity. The second and final steps are repeated until the member of the clusters are not changed anymore.

Peptide selection (PS) is a process that concerns to choose the best PSP(m,n) peptide sequences with m upstream and n downstream amino acid neighbors for each cluster. The combination of PSP(m,n) with $m = 1$ to 30 and $n = 1$ to 30 is experimented by using leave-one-out cross validation for each cluster with respect to the highest *Sn*.

Weight training (WT) is a method that gives a weight to the amino acid pairs in the calculation of similarity score between peptide sequences. The initial weight for each position is determined as 1 and *Sn* is calculated by leave-one-out cross validation with 80% *Sp*.

$$S(P_1, P_2) = \sum_{-m \leq i \leq n} w_i Score(P_1[i], P_2[i]) \tag{2.4}$$

In Equation 2.4, w_i represents weights for position i and it is randomly chosen for each position as +1 or -1. *Sn* is re-calculated by leave-one-out cross validation, if the *Sn* is increased, manipulation is applied.

Matrix mutation MaM is applied with the same way as in GPS 2.0 as we previously described in algorithm 2.

2.1.2 Scoring based SUMOylation site prediction methods

In this section, we briefly explain methods that use canonical consensus motifs and scoring algorithms to predict SUMOylation sites. Apart from using the MotifX algorithm to extract the SUMO motifs, some studies used the motifs mentioned in the literature, and some studies decided on the motifs according to the motif frequency in their data.

SUMOsp (Xue et al., 2006) is one of the early studies that uses computational motif discovery method MotifX (Schwartz & Gygi, 2005) and group based scoring (GPS) algorithm (Zhou et al., 2004) for SUMOylation site prediction. They used motif based algorithm MotifX, GPS algorithm as they found using two algorithms together yielded better results. A potential SUMOylation peptide *PSP(n)* is determined as a lysine residue, which consists of n residues upstream and n residues downstream. They used $n = 7$ for their experiments. Secondly, they used GPS algorithm to cluster SUMOylation site dataset and they decided whether the given sequence was SUMOylated or not by using a threshold they determined. They also experiment on motif based prediction strategy. However, SUMOylation prediction based on motif discovery causes missing a high number of true positives. 23% of the sumoylation sites in SUMOsp dataset do not follow the consensus canonical motifs ψ–K–X–E or ψ–K–X–E/D. Therefore, MotifX is used to extract other motifs that can improve prediction performance. MotifX extract several motifs IKXEP, VKXE, IKXE, LKXE and KXE (X can be any amino acid) which have high confidence score for positive SUMOylation sites. SUMOsp tool shows results for both GPS and MotifX algorithms on given peptide sequence. Moreover, they suggested to use combination of two methods. For instance, if a motif is found by MotifX in a given

peptide that the GPS algorithm predicts as positive, the probability of this given peptide being positive increases.

SUMOsp2 (Ren et al., 2009) is an improved version of SUMOsp (Xue et al., 2006), which used GPS and MotifX algorithms to predict SUMOylation sites. SUMOsp2 eliminated MotifX method and used improved version of GPS. GPS algorithm was improved by eliminating a clustering strategy and using matrix mutation (MaM) technique on substitution matrix BLOSUM62.

Clustering strategy is based on the partitioning of similar SUMOylation sites in GPS algorithm (Zhou et al., 2004). In SUMOsp2, firstly known SUMOylation sites are directly clustered in two group consensus and non-consensus. The first cluster consists of peptide sequences that follows ψ–K–X–E (ψ: a hydrophobic amino acid: A, I, L, M, P, F, or V; X: any amino acid) motif, while non-consensus cluster includes other non-canonical sites. Moreover, while consensus group consists of peptides *PSP(3,3)*, non-consensus group contains peptides *PSP(3,5)*. After that step, enhanced version of GPS algorithm is used on both clusters separately.

GPS 2.0 algorithm, which mutate BLOSUM62 matrix (Henikoff & Henikoff, 1992) for measuring optimal similarity between peptide sequences by mutation (MaM) (Xue et al., 2008) is used in SUMOsp2. BLOSUM62 matrix calculates similarity between residues '*' and others as -4. This similarity score is taken into account as 0 after MaM is applied. Also, 2 is used to get a new substitution matrix. The given peptide sequence is predicted as SUMOylated or not by using a threshold and its similarity score, which is calculated by Equation 2.2 in SUMOsp2.

GPS-SUMO (Zhao et al., 2014) predicts both SUMOlyation sites and SUMO interaction motifs (SIMs), which are binding motifs through which SUMO proteins can interact with other proteins non-covalently. It is a developed version of SUMOsp2 (Ren et al., 2009), which used modified GPS 3.0 algorithm for SUMOylation site prediction. GPS 3.0 algorithm is used k-means clustering for finding clusters in training data, peptide selection method to find the best length for peptide sequences in each cluster, matrix mutation to find a more suitable substitution matrix for the problem and weight training for optimizing the measurement of the similarity scores between peptide sequences as we mentioned before. The particle swarm optimization (Eberhart & Kennedy, 1995) method is used in the GPS 4.0, which is the latest version of group based scoring algorithm in the literature, to decrease the computational time in GPS 3.0. Also, more accurate prediction results were taken when particle swarm optimization is used to find weights in weight training part and final substitution matrix after using matrix mutation method.

JASSA (Beauclair et al., 2015) group experimentally validated SUMOylation sites in clusters based on the presence of the consensus motif and its inverted form; then predict, SUMOylation sites using a scoring system based on the position-specific frequencies of amino acids calculated specifically for each of these groups.

	Name	Motif	Nb	%
Consensus direct	Strong consensus	$[\Psi_1]$-[K]-[x]-$[\alpha]$	498	56.8
	Consensus	$[\Psi_2]$-[K]-[x]-$[\alpha]$	591	67.4
	Weak consensus	$[\Psi_3]$-[K]-[x]-$[\alpha]$	598	68.2
	PDSM	$[\Psi_2]$-[K]-[x]-$[\alpha]$-$[x]_2$-[S]-[P]	32	3.6
	NDSM	$[\Psi_2]$-[K]-[x]-$[\alpha]$-[x]-$[\alpha]_{2/6}$	231	26.3
	HCSM	$[\Psi_4]_3$-[K]-[x]-[E]	105	12.0
	SC-SUMO	[P/G]-$[x]_{(0-3)}$-[I/V]-[K]-[x]-[E]-$[x]_{(0-3)}$-[P/G]	110	12.5
	Minimal SC-SUMO	[I/V]-[K]-[x]-[E]-$[x]_{(0-3)}$-[P]	178	20.3
	SUMO-acetyl switch	$[\Psi_2]$-[K]-[x]-$[\alpha]$-[P]	130	24.8
	pSuM	$[\Psi_2]$-[K]-[x]-$[_p$S]-[P]	1	0.1
Consensus inverted	Strong consensus	$[\alpha]$-[x]-[K]-$[\Psi_1]$	30	3.4
	Consensus	$[\alpha]$-[x]-[K]-$[\Psi_2]$	77	8.8
	Weak consensus	$[\alpha]$-[x]-[K]-$[\Psi_3]$	80	9.1
Non-Consensus			229	26.1

Table 2.1 Frequency of motifs on known SUMOylation sites in the JASSA dataset. *Nb* represents number and *%* represents percentage. Ψ_1 = I, L, V; Ψ_2 = A, F, I, L, M, P,V, W; Ψ_3 = A, F, G, I, L, M, P, V, W, Y. α = D, E; $_p$S/T = phosphorylated serine/threonine.

Clustering strategy is based on motif discovery on known SUMOylation sites in dataset. They directly searched motifs from literature in their dataset and frequency of those each motif can be seen on Table 2.1. A motif search in the dataset showed that 68.2 % of the DB follows the direct consensus ψ–K–X–α motif (ψ: a hydrophobic amino acid: A, F, G, I, L, M, P, V or Y; X: any amino acid; α= D or E). Also, since $\Psi_1 \subset \Psi_2 \subset \Psi_3$, strong consensus motifs($Melchior$, 2000; $Rodriguez, Dargemont\&Hay$, 2001) $\subset$ consensus motifs $\subset$ weak consensus motifs. Among known SUMOylation sites, 3.6% are PDSM (Hietakangas, Anckar, Blomster, Fujimoto, Palvimo, Nakai & Sistonen, 2006) and 12.0% are HCSM (Matic, Schimmel, Hendriks, van Santen, van de Rijke, van Dam, Gnad, Mann & Vertegaal, 2010). $[x]_i$ represents there are i subsequent amino acid x in that position in PDSM and HCSM. Moreover, among known SUMOylation sites, 26.3% are NDSM (Yang, Galanis, Witty & Sharrocks, 2006), 12.5% are SC-SUMO (Benson, Li, Kieckhafer, Dudek, Whorton, Sunahara, Iñiguez-Lluhí & Martens, 2007), 20.3% are Minimal SC-SUMO (Subramanian, Benson & Iñiguez-Lluhí, 2003), 24.8% are SUMO-acetyl switch (Stankovic-Valentin, Deltour, Seeler, Pinte, Vergoten, Guérardel, Dejean & Leprince, 2007), 0.1% are pSuM (Picard, Caron, Bilodeau, Sanchez, Mascle, Aubry & Tremblay, 2012). $[x]_{(i-j)}$ represents there are at least i at most j subsequent amino acid x in that position. 80 sites on the dataset follows the inverted consensus motif (Ivanov, Peng, Yurchenko, Yap,

Negorev, Schultz, Psulkowski, Fredericks, White, Maul, Sadofsky, Zhou & Rauscher, 2007; Matic et al., 2010) that is 9.1% of data. The remaining 26.1% does not follow any motif on the Table 2.1.

According to the motif extraction results, the entire dataset is clustered into 3 different group the consensus motif, its inverted form and all (all the sequences in dataset).

Scoring strategy is a method to calculate score for the prediction of a given peptide sequence. Two predictive scores PS_d and PS_i are calculated for a given peptide in JASSA. However, their calculation strategy is changed with respect to the selected cluster approach. Since either All or Directed cluster is chose, Equation 2.5 is used to calculate predictive scores.

$$PS = \begin{cases} PS_d = f_{-1}(aa_{-1}) \text{ x } f_0(K_0) \text{ x } f_{+2}(aa_{+2}) \text{ x } 100 \\ PS_i = f_{-1}(aa_{+1}) \text{ x } f_0(K_0) \text{ x } f_{+2}(aa_{-2}) \text{ x } 100 \end{cases} \tag{2.5}$$

$$PS = \begin{cases} PS_d = f_{+1}(aa_{-1}) \text{ x } f_0(K_0) \text{ x } f_{-2}(aa_{+2}) \text{ x } 100 \\ PS_i = f_{+1}(aa_{+1}) \text{ x } f_0(K_0) \text{ x } f_{-2}(aa_{-2}) \text{ x } 100 \end{cases} \tag{2.6}$$

Equation 2.6 is used when the Inverted cluster is selected. $f_p(aa_q)$ is the frequency at position p of the amino acid on the given sequence at the position q of the selected cluster. Importantly, $f_0(K_0) = 1$ and if a residue cannot be found on cluster its frequency is taken as 0.0001. The highest score among PS_d and PS_i is a prediction score for a given peptide sequence. SUMOylation sites are predicted with respect to the two cut-off values, which were determined by decision tree.

2.2 Machine Learning Methods

In this section, we review SUMOylation predictors that cast the problem as a binary classification task.

pSumo-CD (Jia et al., 2016) is a SUMOylation site prediction method which uses

conditional probability of amino acid pairs for vectorization and covariance discriminant (CD) algorithm (Chou & Maggiora, 1998) for prediction. They used 21-mer peptide sequences with 10 upstream and 10 downstream of the candidate site. Since CD algorithm can handle class imbalance, the entire 755 positive samples, 9944 negative samples were used in their experiments.

$$P(K) = P^{+}(K) - P^{-}(K) \tag{2.7}$$

The feature vectors of each sequence is created with respect to the Equation 2.7. $P^{+}(K)$ indicates a conditional probability matrix of a sequence which is calculated among positive classes. Therefore, 20x1 dimensional matrix is constructed with excluding 'K' lysine amino acid in sequence. For each position, a conditional probability $P_i(A_i|A_{i-1})$ represents that the probability of an amino acid A_i and its closest right neighbor A_{i-1} being together. After the calculation of this matrix for both classes positive and negative, we can find $P(K)$ to encode each sequence.

$$Sgn(\delta) = \arg\min\{\mathbb{F}(\mathbb{P}, \overline{\mathbb{P}^{\delta}})\}, (\delta = + \text{ or - }) \tag{2.8}$$

Hence, a given peptide sequence $\mathbb{P}$ can be predicted as SUMOylation site or not by the calculation of class specific *Sgn* using Equation 2.8. CD algorithm is used for minimization of $\mathbb{F}(\mathbb{P}, \overline{\mathbb{P}^{\delta}})$, which uses Mahalanobis distance (Mahalanobis, 1936) between peptide sequences and the covariance matrix of a class δ.

C-iSUMO (López, Dehzangi, Reddy & Sharma, 2020) uses an Adaboost classifier (Freund & Schapire, 1999) that relies on features derived from structural properties such as accessible surface area of protein site and backbone torsion angles between residues. The data in the experiments was highly imbalanced with 780 SUMOylation sites and 21353 non-SUMOylation sites. In C-iSUMO, the length of the each positive and negative residue was taken as 31 with 15 upstream and 15 downstream amino acids.

Accessible surface area (ASA) is a feature that provides an information for 3D structure of a protein sequence. SPIDER2 algorithm (Yang, Heffernan, Paliwal, Lyons, Dehzangi, Sharma, Wang, Sattar & Zhou, 2017), which is a pre-trained deep learning model with a sequential, physicochemical, and evolutionary features of protein, was used to extract ASA of each amino acid to take structural features of peptide sequences.

Backbone torsion angles provide a local structural information between neighboring amino acids. The backbone torsion angles φ and Ψ are continues features for a given amino acid. SPIDER2 algorithm is also used to find these features. Besides φ and Ψ, SPIDER2 algorithm extract two other angles θ which is formed between calcium atoms and τ which represents the rotation around calcium bond.

NearMiss method (Yen & Lee, 2006) is an under-sampling strategy for the highly imbalanced training data. Samples in majority class are extracted from the training data with respect to the average distance between minority class. Thus, the sensitivity of the C-iSUMO was increased.

AdaBoost (Freund & Schapire, 1999) is a machine learning method that uses an ensemble technique to predict SUMOylation sites by C-iSUMO. AdaBoost trains weak classifiers sequentially by using bootstrap sampling to get strong classifier. Initially, a weak classifier is trained with random weights and error rate is measured. The weights of the incorrectly classified samples (hard examples) are increased to take more attention on them for the new classifier. Thus, each new weak classifier learns to correct the errors of its predecessor.

SUMO-Forest (Qian, Ye, Zhang & Zhang, 2020) extracts dipeptides from protein sequences by using bi-gram and k-skip-bi-gram with k=1,2 approaches. Thus, each protein site is vectorized by finding probability for each dipeptide which is calculated with respect to their frequency. Extraction of bi-grams from the sequences for vectorization is commonly used technique in natural language processing and bioinformatics. For instance bi-grams of a protein sequence VKPEI are VK, KP, PE and EI and 1-skip-bi-gram sequences are VP, KE and PI.

The peptide sequence is taken as 21. The ratio between positive sites and negative sites is 1:9 in their dataset. To handle with the imbalanced dataset, class weights are used in the training phase to give more penalty for mis-classifications of the minority class. They use an ensemble technique, Cascade Forest (Zhou & Feng, 2017), to predict SUMOylation sites. There are two consequtive phase in the Cascade Forest model. The first phase is uses the same strategy with AdaBoost (Freund & Schapire, 1999) classifier. In the second phase, assembling phase, the weighted average of the results of weak classifiers are calculated. Weights are found by using genetic algorithm for each M weak classifier.

Ensemble and Transfer Learning method (He, Wang, Gao, Wang, Yu, Xu & Zhao, 2019) is used to predict SUMOylation sites by using deep learning architectures. To the best of our knowledge, the only deep learning model that was used in SUMOylation prediction was provided by (He et al., 2019). They built an ensemble

architecture by using deep learning models. Each model receives a separate amino acid feature to combine all information learned. In addition to that, they transfer weights of a model which is trained on similar domain to the SUMOylation site prediction problem. The dataset in their experiments is imbalanced with a 1:13 positive/negative ratio and each peptide sequence consists of 21 upstream and 21 downstream amino acids. They divide the imbalanced data into N bins that includes equal number of randomly chosen positive and negative samples. Therefore, they train the network N times (by using each bin). Physico-chemical properties of amino acids and one-hot encoding are used for the representation of peptide sequences in vector space. Physico-chemical properties are extracted by AAindex database, which cluster them into 6 groups α and turn properties, β propensity, composition, hydrophobicity, physicochemical propensities, and other properties.

Their proposed neural network architecture contains 7 sub-networks, each of them is for different encoding representations. The final layer of each sub-network has two neurons with Softmax activation function to predict probability of peptide sequence. Firstly, all networks are trained separately for learning different features of amino acids. Then, sub-networks combined with final 2 neurons to construct a strong classifier. The training strategy for the final network is that they freeze weights of sub-networks and train only last 2 neurons. Thus, their proposed deep architecture is trained with 7 weak neural nets. They also used transfer learning strategy with a similar domain ubiquitination, which is another type of PTM like SUMOylation, sites prediction. Firstly, the model is trained on ubiquitination site dataset and then same model is tuned on SUMOylation site data.

3. BACKGROUND

In this chapter, we present the background information on the pre-processing steps and our model components.

3.0.1 One-hot Encoding

One-hot encoding is a widely used scheme for converting categorical data into numerical vectors, where a new binary columns is created indicating the presence of the categories. We have 21 different categories which consist of 20 amino acids and an additional category 'X' to represent unknown amino acids. As an example, the one-hot representation of lysine is a vector of length 21 where all entries are 0, except the one representing the amino acid type 'K', which is 1.

3.0.2 BLOSUM62 Encoding

BLOSUM62 matrix is a substitution matrix mostly used for aligning protein sequences and represents amino acids replacibility with each other(Henikoff & Henikoff, 1992) BLOSUM62 matrix entries are calculated based on the frequencies of amino acid substitutions observed when a related group of protein are aligned (Henikoff & Henikoff, 1992). The BLOSUM62 value for a particular pair of amino acid is the log-odds ratio that estimates the biological probability of a substitution to occur relative to that substitution being merely by chance. Figure 3.1 represents BLOSUM62 values for each pair of 20 amino acid. The number 62 signifies how divergent the sequences that are used to construct the alignment. The BLOSSUM62 matrix is calculated over protein sequences such that every pair of sequences is at least 62% identical when two proteins were pairwise aligned. In BLOSSUM62 encod-

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
S	-1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
T	-1	1	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
P	-3	-1	-1	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	0	1	0	-1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
G	-3	0	-2	-2	0	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-
N	-3	1	0	-2	-2	0	6	-	-	-	-	-	-	-	-	-	-	-	-	-
D	-3	0	-1	-1	-2	-1	1	6	-	-	-	-	-	-	-	-	-	-	-	-
E	-4	0	-1	-1	-1	-2	0	2	5	-	-	-	-	-	-	-	-	-	-	-
Q	-3	0	-1	-1	-1	-2	0	0	2	5	-	-	-	-	-	-	-	-	-	-
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8	-	-	-	-	-	-	-	-	-
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5	-	-	-	-	-	-	-	-
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5	-	-	-	-	-	-	-
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5	-	-	-	-	-	-
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4	-	-	-	-	-
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4	-	-	-	-
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4	-	-	-
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6	-	-
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	-
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11

Figure 3.1 BLOSUM62 matrix

ing scheme, we represent each type of amino acid with its row in the BLOSSUM62 matrix.

3.0.3 NLF Encoding

NLF encoding technique was provided by (Nanni & Lumini, 2011) to represent the physicochemical properties of amino acids. Firstly, they used the principal component analysis to de-correlate peptide data and then feature factors are determined by using non-linear Fisher's transform. Figure 3.2 represents feature vectors in the NLF matrix.

3.1 Min-Max Scaling

In this section, we present the scaling method that we used on our encoded vectors BLOSUM62 and NLF. We used min-max scaling technique as a pre-processing step to represent features in between 0 and 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A	0.42	-2.07	-0.67	0.01	-1.1	-0.32	-0.2	0.09	-0.2	0.09	-0.11	0.15	0.01	0.06	0.02	0.16	0.07	-0.03
R	1.65	1.4	-0.01	-0.88	-0.08	-0.07	0.6	-0.53	-0.1	0.01	0.09	-0.07	0.09	0.08	0.03	0.09	0.03	-0.02
N	1.68	0.3	-0.49	0.15	0.09	0.59	-0.06	0.02	0.14	0.0	-0.14	-0.09	0.08	-0.14	-0.11	-0.01	0.01	0.01
D	0.81	0.13	1.36	0.63	-0.15	-0.1	-0.45	-0.31	-0.1	0.03	0.15	0.02	0.16	0.12	-0.07	-0.11	-0.01	-0.05
C	-2.7	-0.32	1.19	1.37	0.04	-0.18	0.64	0.21	0.26	0.35	-0.02	-0.11	0.05	-0.04	-0.03	0.1	0.04	-0.04
Q	1.71	1.11	-0.08	0.15	0.11	0.45	0.11	0.08	0.02	0.25	-0.12	0.25	-0.2	0.16	-0.01	-0.07	0.02	0.03
E	1.56	0.48	0.87	-0.02	-0.07	0.13	-0.22	-0.15	-0.09	0.1	0.04	0.05	-0.12	-0.28	0.03	0.09	-0.06	0.02
G	1.32	-2.05	-0.6	0.31	0.61	-0.58	0.0	-0.3	0.44	-0.14	0.18	0.12	-0.12	0.01	0.06	0.0	-0.04	0.01
H	0.13	1.5	-1.22	0.52	-1.14	-0.45	0.13	0.04	0.1	-0.07	0.11	-0.13	-0.06	-0.07	-0.01	-0.16	-0.06	0.03
I	-1.52	-0.45	-0.39	-0.36	-0.01	0.55	0.06	0.1	-0.02	0.08	0.1	0.12	0.18	0.01	0.12	-0.02	-0.2	-0.01
L	-1.29	-1.21	-0.25	-0.96	0.18	0.06	-0.04	0.0	-0.09	0.26	0.18	-0.05	0.0	-0.11	0.01	-0.15	0.14	0.02
K	2.03	0.26	1.22	-0.98	-0.05	-0.32	0.1	0.73	0.11	-0.19	0.14	0.02	0.03	0.02	-0.05	0.01	-0.01	-0.02
M	-1.72	0.85	-0.34	0.44	-0.01	0.8	-0.16	0.05	0.05	-0.3	0.29	0.06	-0.02	0.03	0.03	0.09	0.14	0.0
F	-2.37	0.23	-0.09	-0.37	0.19	-0.04	0.03	-0.06	-0.14	-0.14	-0.1	0.03	-0.21	-0.04	-0.09	-0.03	-0.06	-0.18
P	1.41	0.27	-1.09	0.77	0.87	-0.33	-0.04	0.27	-0.43	0.06	0.1	-0.14	0.03	0.03	-0.01	0.04	-0.03	0.01
S	1.47	-1.11	-0.27	0.13	0.15	0.22	0.09	-0.05	0.05	-0.14	-0.3	0.01	0.16	-0.03	-0.01	-0.06	0.05	-0.06
T	0.3	-0.68	0.88	0.23	-0.1	0.23	0.03	-0.01	-0.14	-0.16	-0.15	-0.2	-0.12	0.05	0.21	-0.07	0.0	0.04
W	-2.83	1.79	0.16	-0.14	0.42	-0.84	-0.13	-0.06	-0.04	-0.18	-0.32	0.26	0.15	-0.08	0.04	-0.01	0.06	0.12
Y	-0.7	0.95	-0.36	-0.6	0.09	-0.06	-0.55	0.01	0.28	0.17	-0.12	-0.23	-0.02	0.13	0.05	0.08	-0.02	-0.02
V	-1.33	-1.39	0.15	-0.4	-0.04	0.27	0.07	-0.12	-0.1	-0.06	-0.01	-0.09	-0.07	0.1	-0.2	0.02	-0.08	0.13

Figure 3.2 NLF matrix

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

$$X_{scaled} = X_{std} * (max - min) + min \tag{3.2}$$

The mathematical formula of the min-max scaling process is represented with Equation 3.1 and Equation 3.2. The standard deviation is calculated for each sample and for each feature with respect to the feature itself X, the maximum X_{max} and the minimum values X_{min} among that feature column. The important point is that, X_{max} and X_{min} should be calculated on the training data to avoid an information leak from the test data.

3.2 Background on SUMOnet Components

In this section, we briefly explain activation functions and neural layers that are used in our architectures.

3.2.1 Activation Functions

Activation functions determines how the weighted sum of the input on a node is transformed into an output node. It enables learning complex features. The activation functions should be differentiable because of the back-propagation algorithm. Below we explain ReLU and softmax that we used in SUMOnets (Szandala, 2020). We also briefly explain Sigmoid and Tanh activation functions, which are used in the neural layers.

3.2.1.1 ReLU

Rectified Linear Unit (ReLU) is the most commonly used activation function especially in CNNs (LeCun, Boser, Denker, Henderson, Howard, Hubbard & Jackel, 1989). As can be seen in the Figure 3.3, ReLU provides value itself if value is greater than or equal to 0 and if a value is less than zero the output is always zero.

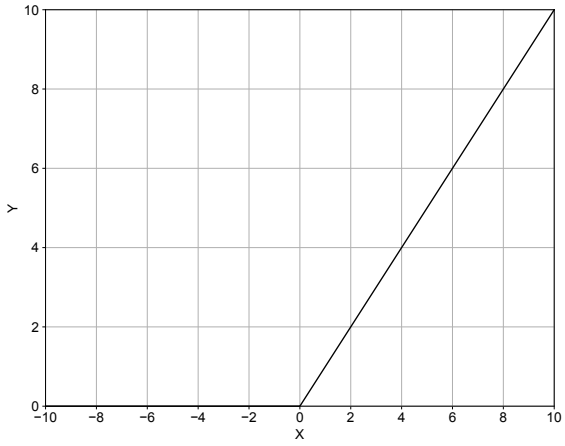

Figure 3.3 Visual representation of ReLU activation function.

$$f(x) = \max(0, x) \tag{3.3}$$

ReLU functions is shown in the Equation 3.3. Thus, the derivative of the ReLU activation function can be calculated as $\max(0, 1)$ for the back-propagation. The derivative is 1 for values above 0, else it assigns 0.

3.2.1.2 Softmax

General application area of the softmax activation function is the calculation of class probabilities in output layer for classification problems. The mathematical formula of the softmax activation function is as follows:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.4}$$

Here, K represents the number of inputs for the softmax function. It can be thought as the number of categories for the classification problem, if the function is on the output layer. Therefore, for the category i, the probability is calculated as on the Equation 3.4 and sum of the output vector of softmax is always equal to 1.

3.2.1.3 Sigmoid

SUMOnets use the sigmoid activation function in gated recurrent units' cells (Cho, van Merrienboer, Gulcehre, Bahdanau, Bougares, Schwenk & Bengio, 2014) . The sigmoid function squeezes input between 0 and 1 as can be seen on the Figure 3.4; thus, it can be thought as a normalization.

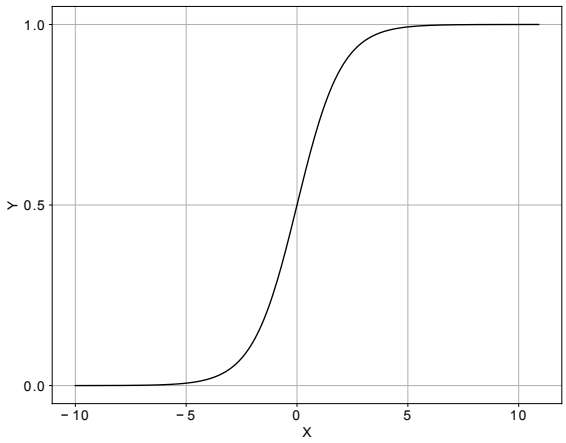

Figure 3.4 Visual representation of Sigmoid activation function

$$f(x) = \frac{1}{1+\epsilon^{-x}} \tag{3.5}$$

Sigmoid function is problematic because its derivative becomes smaller when the output is very high or low and that causes vanishing gradient problem. Also, ex-

ponential calculation of function brings complexity problem (Shatravin, Shashev & Shidlovskiy, 2022).

3.2.1.4 Tanh

We used tanh activation function in the gated recurrent units (Cho et al., 2014). Tanh is a hyperbolic tangent function as can be seen on Figure 3.5 and it is very similar with the sigmoid function. However, when input of the function is large or small, the output is always smooth and also it is zero centring.

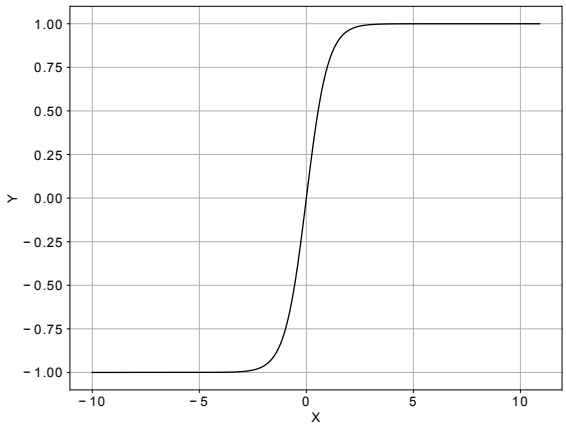

Figure 3.5 Visual representation of Tanh activation function.

$$f(x) = \frac{\epsilon^{z} - \epsilon^{-z}}{\epsilon^{z} + \epsilon^{-z}} \tag{3.6}$$

On the Equation 3.6, the mathematical formulation of tanh function can be also seen. The main advantage of the tanh function is that if an input value is near to zero it is also mapped to near zero values. However, tanh activation function prone to vanishing gradient as a drawback.

3.2.2 Pooling Layers

Pooling layers are used to decrease computational complexity and variance in neural network architectures (Gholamalinezhad & Khosravi, 2020). We used several

pooling methods Average Pooling, Max Pooling, and Global Average Pooling in our experiments. In this section, each pooling method will be explained with examples. Pooling layers are not trainable and reduce the input vector size. Therefore, the number of trainable parameters are decreased and computational complexity of the network is declined. Reducing the number of trainable parameters is also an important characteristic to decrease over-fitting.

3.2.2.1 Average pooling

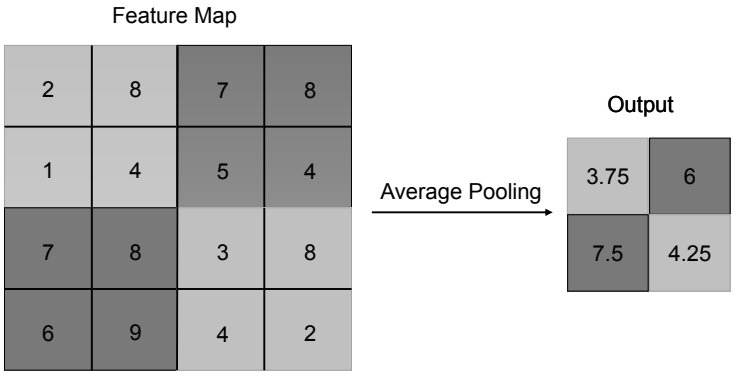

Figure 3.6 The working process of the Average Pooling on a feature map with 2x2 pooling kernel and stride 2.

Average Pooling layer is popular in computer vision area to smooth sharp features on images, however, it is not widely used in NLP problems (Suárez-Paniagua & Segura-Bedmar, 2018). The main disadvantage to use Average Pooling in NLP is that taking average of the features misleads the network when the vector representation of some sequences are padded. We discard this disadvantage because our peptides have same length and thus, padding is not used in our experiments.

The working process of the Average Pooling layer depends on kernel size and stride as can be seen on Figure 3.6. Kernel size and stride are hyper parameters which should be tuned with respect to data and network structure. Kernel moves on the feature vector by using stride parameter and it takes an average of values inside of its kernel.

3.2.2.2 Max pooling

Max Pooling layer uses a max function to take the maximum value and it is commonly used with CNN architecture (Suárez-Paniagua & Segura-Bedmar, 2018). The main reason behind that CNN extract features from its input and maximum feature values can give better understanding about the input vector.

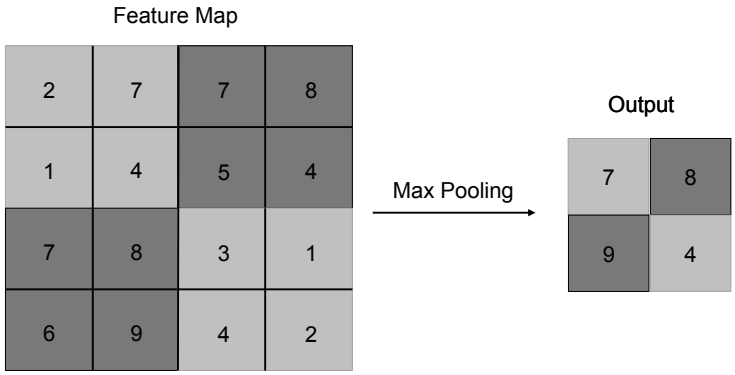

Figure 3.7 The working process of the Max Pooling on a feature map with 2x2 pooling kernel and stride 2.

Max Pooling layer works in similar ways as Average Pooling layer in terms of stride and its kernel size as can be seen on Figure 3.7. Kernel window moves according to the stride and layer takes the maximum value inside that window.

3.2.2.3 Global average pooling

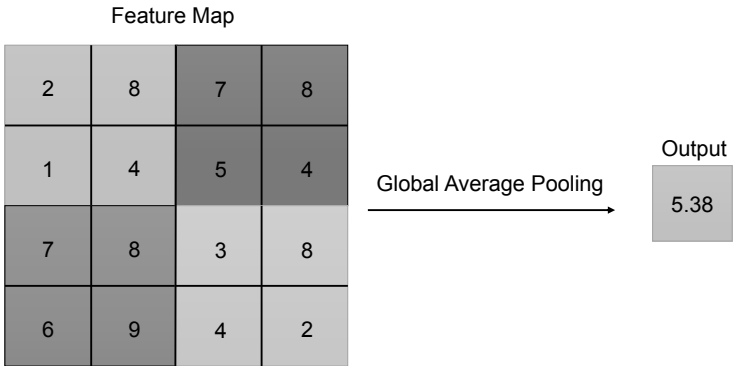

Figure 3.8 The working process of the Global Average Pooling layer on a feature map

Global Average Pooling is a layer that calculates average of feature values inside its input likes Average Pooling layer (Lin, Chen & Yan, 2013). The difference between Global Average Pooling and Average Pooling is that the kernel size of the Global Average Pooling layer is equal to its input vector size, however, the kernel size of the Average Pooling layer is a hyper-parameter and it should be tuned. Therefore,

all values in the feature vector are averaged to give an output as can be seen on the Figure 3.8.

3.2.3 Neural Layers

The layers, Embedding, FFNN, CNN, GRU, BiGRU, and Attention mechanism that we used in each architecture SUMOnet -1, -2, and -3 will be explained in this section.

3.2.3.1 Embedding layer

As we mentioned earlier, one of the encoding techniques that we use is one-hot encoding. Even one-hot encoding is a practical way to vectorize categorical data, it creates 0s and 1s for each amino acid in peptide sequence. Therefore, memory usage is inefficient for long peptide sequences. Also, even amino acid positions are determined, contextual information in peptide sequence cannot be represented well by one-hot encoding.

Encoding techniques that have used amino acid features such as BLOSUM62 (Henikoff & Henikoff, 1992) and NLF (Nanni & Lumini, 2011) boost model performance. Embedding layer in Keras (Chollet & others, 2015) is used to learn feature representation for each amino acids. Thus, network can learn the vector representation of peptide sequence.

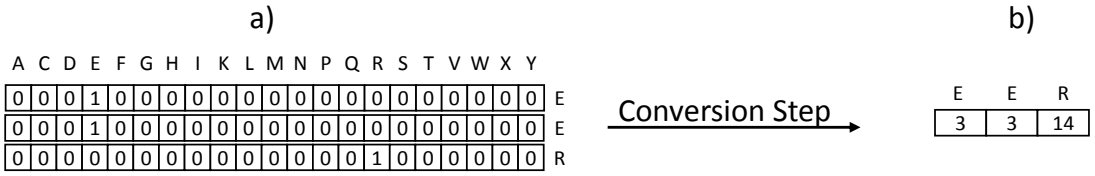

Figure 3.9 **a)** is the one-hot representation of amino acid sequence EER. Embedding layer firstly converts one-hot encoded peptide sequence to a vector like on **b)** that represents positions of each amino acids.

We feed embedding layer with one-hot encoded peptide sequence. Firstly, layer converts one-hot representation of each amino acid to position vector. In Figure 3.9, conversion step for amino acid sequence EER is illustrated. Secondly, each amino

acid is mapped to the new vector representation on embedding layer with respect to the position vector. Initially, embedding layer is randomly initialized and its size is determined by vocabulary size (number of unique amino acids in our case) and vector dimension as a hyper-parameter. The layer is trainable, so, appropriate vector representation can be learned in the training phase.

3.2.3.2 Feed forward neural network

A Feed forward neural networks (FFNNs) are based on single direction data movement from input layer to the output layer (Ojha, Abraham & Snášel, 2017). In other words, output of each hidden layer feeds the next layer in the network. FFNNs consist of one or more hidden layers between input and output layer and each hidden layer contains one or more neurons. Also, an activation function is applied to the each neuron for non-linearity, we use ReLU activation function for hidden layers and softmax for output layer.

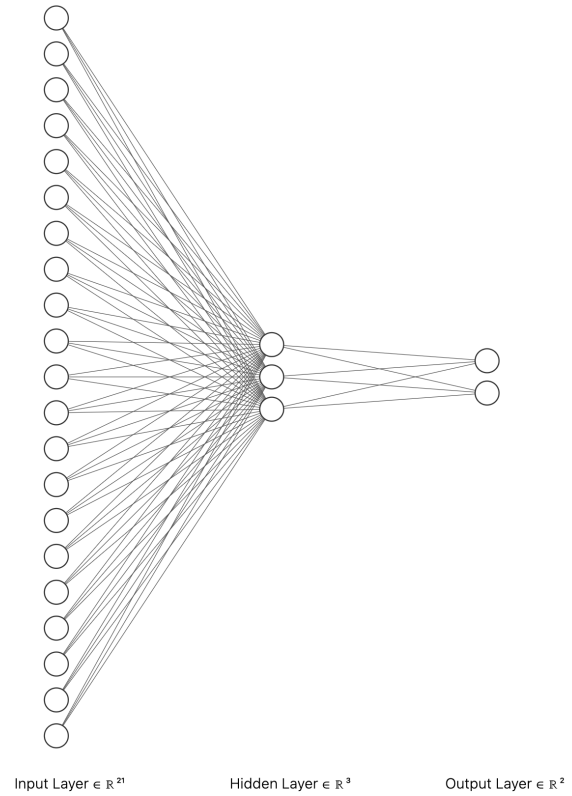

Figure 3.10 Visual representation of FFNN with 1 hidden layer.

The working process of the feed forward neural network is illustrated on the Figure 3.10. There are 21 features, which are represented with $\mathbb{R}^{21}$, of an input sample

x, X is a peptide sequence in our case because each peptide consists of 21 amino acids. These features are multiplied by the weights and biases are added to the result of that multiplication.

$$\mathbb{Y} = \begin{bmatrix} x_1 & x_2 & . & . & x_{20} & x_{21} \end{bmatrix} * \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ . & . & . \\ . & . & . \\ w_{20,1} & w_{20,2} & w_{20,3} \\ w_{21,1} & w_{21,2} & w_{21,3} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \tag{3.7}$$

The number of nodes in the hidden layers are hyper-parameters. Suppose that we determine the number of nodes in hidden layer on Figure 3.10 as 3. Therefore, we can formulate mathematical operations in hidden layer as in Equation 3.7. The first matrix represents a feature vector, in our case we have 21 features as described above. The second matrix represents the weight matrix and each column represents different nodes. As can be seen on the Equation 3.7, the first dimension of the weight matrix is determined with respect to the feature vector or the input matrix of that hidden layer and the second dimension is a hyper parameter that represents the number of nodes in the hidden layer. The third vector represents bias matrix. The output vector of Equation 3.7 is $\mathbb{Y} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$. Last but not least, the activation function f is used on the output $f(\mathbb{Y})$ for the non-linearity.

3.2.3.3 1-D Convolutional neural network (1-D CNN)

Peptide sequences that constitute our data is vectorized by 1-d encoding representations of amino acids. Therefore, each CNN in our architectures consists of 1-d convolutional kernels.

1-D Convolutional Neural Network (1-D CNN) is a modified version of traditional 2-Dimensional CNN. 2-D CNNs are efficient neural models especially when 2-D data is used such as images and videos. 1D CNNs have several advantages when the data is represented in 1-D feature space (Kiranyaz, Avci, Abdeljaber, Ince, Gabbouj & Inman, 2021):

- Since 2-D representation of images is in NxN feature vectors, the computa-

tional complexity of the convolution operation with KxK kernel is $O(N^2 * K^2)$. On the other hand, the computational complexity of 1-D CNN on 1-D data $O(N * K)$ shows the significant complexity advantage of using 1-D CNNs instead of using 2-D CNNs.

- Recent studies show that, 1-D CNNs consist of shallow (1-2 hidden CNN layers) architectures in literature. Thus, the training time of 1-D CNNs is less than the deeper architectures of 2-D CNNs and it is also easy to implement.
- 1-D CNNs can be run into standard computer systems even in CPU. However, the deeper 2-D CNN architectures needs more GPU power.
- Because of the above reasons, 1-D CNNs are easy to use on real time systems such as mobile/web services.

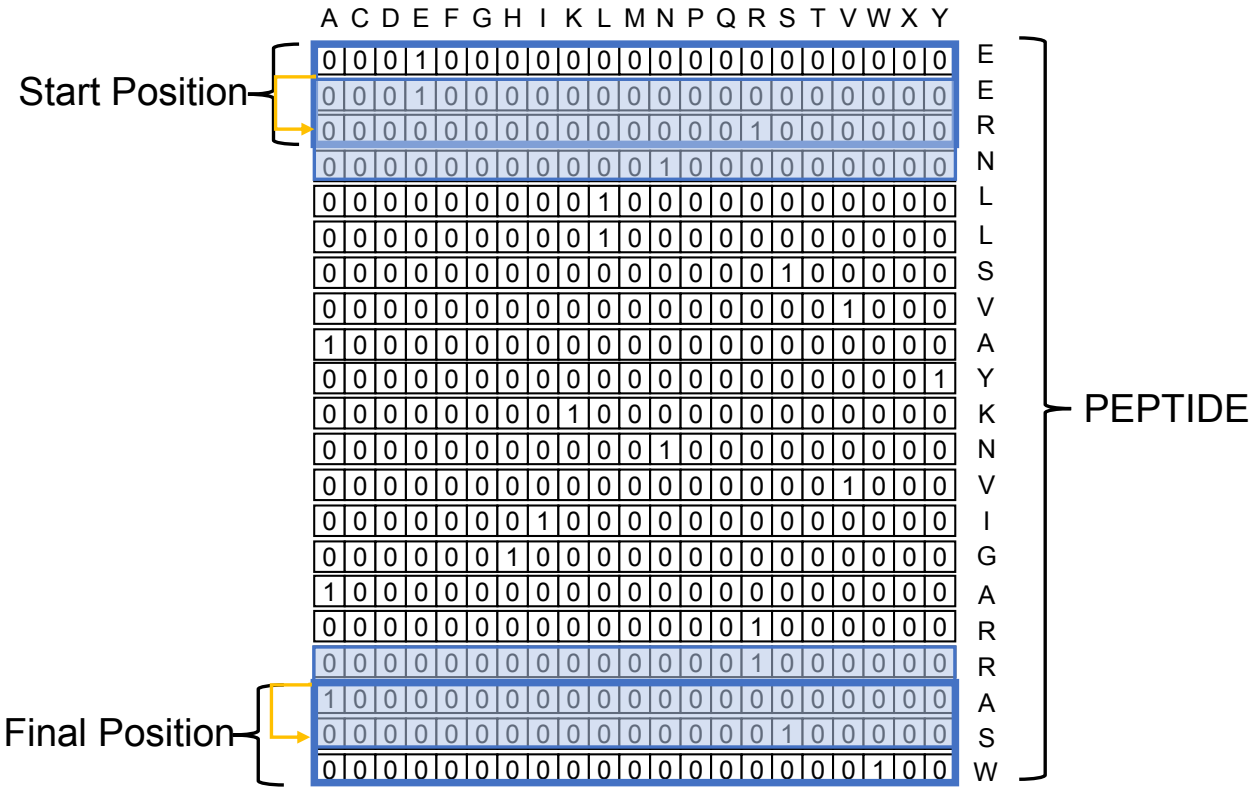

Figure 3.11 The working process of 1-D convolutional neural network. Figure represents the start and final positions of 1-D kernel on a peptide sequence. 1-d kernel has height = 3 and width = 21.

Figure 3.11 illustrates the working process of the 1-D CNNs on our data. The height and the width of the kernel are hyper-parameters for 2-D CNNs, however, the width of the kernel is automatically determined and equal to the encoding length in 1-D CNNs. We show the one-hot vector representation of a peptide sequence in Figure 3.11, which has $1x21$ vectors for each amino acids. Therefore, 1-D convolutional

kernel automatically has $Hx21$ dimensions, H represents height, which is a hyper-parameter, and 21 represents the width of the kernel. Also, in Figure 3.11, the stride, which is a hyper-parameter, controls amount of movement over the amino acids and it can be seen as 1. Hence, we can extract N-gram representations by using N stride in 1-D CNNs.

3.2.3.4 Gated recurrent units

Gated recurrent units (GRUs) are the updated version of the Recurrent Neural Network (RNN) to solve gradient vanishing problem (Cho et al., 2014). Some problems in natural language processing requires understanding of each word to solve the meaning of the entire sentence. Therefore, neural model needs to remember very first word in the sentence. Words are amino acids and sentence is peptide in our case. GRUs can learn to remember specific amino acids or forget them with its **update gate** and **reset gate**.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{3.8}$$

Update gate decides the amount of information that needs to be passed by the help of sigmoid activation function. The calculation of the update state z_t can be seen on Equation 3.8. Sigmoid activation function outputs a value between 0 and 1. The value 1 means that all the information will be passed and the value 0 means that none of the information will be passed to the output. This gate avoids the gradient vanishing problem by remembering all the necessary past information.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{3.9}$$

Forget gate decides the amount of information that needs to be forgotten and the forget state is also calculated like the update state as on Equation 3.9. The value 1 means that all the information will be forgotten and the value 0 means that none of the information will be forgotten for the output. The usage of the output is different than the update state.

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{3.10}$$

The usage of the output of forget gate is shown on the Equation 3.10. The previous cell information is represented with h_{t-1}, hence, element wise multiplication with the forget state determines how much of the past information will be removed. The remaining information is added to the current information x_t and new vector $\tilde{h}_t$ (let's say information state) is created by tanh activation function.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{3.11}$$

As a final step, network needs to calculate output vector h_t for the current cell. Therefore, update state is applied in this phase to give updated information. As can be seen on the Equation 3.11, if update gate is near to 1 that means the information state $\tilde{h}_t$ is important. Thus, $1 - z_t$ is near to zero and helps to forget past information h_{t-1}.

3.2.3.5 Bi-directional gated recurrent units

Bi-directional gated recurrent unites (BiGRUs) is the modified version of the basic RNN structure. Advantages of the GRUs are mentioned early and the bi-directional property brings in another advantage to GRUs. In NLP, some problems need to analyze both future and previous features X_{t-1}, X_{t+1} to understand feature X_t.

Each hidden state h is a GRU cell and the forward layer evaluates the input from the beginning to the end, while the backward layer takes it from end to beginning. Therefore, output y_t is calculated with considering both previous and upcoming features. The output of a hidden state h_t is calculated as we discussed in *Gated recurrent units* Sec. 3.2.3.4.

3.2.3.6 Attention mechanism

The Attention mechanism that we used in our architecture was firstly proposed to increase the performance of the basic RNN encoder-decoder model (Cho, van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk & Bengio, 2014) in machine translation problem. RNN encoder-decoder model used a fixed-length vector for encoding an input sentence, thus, decoder can take a limited information about input vector. This bottleneck is caused by encoding with fixed-length vectors, even if the sentences get longer. The proposed method, which is called Bahdanau Attention mechanism (Bahdanau, Cho & Bengio, 2015), is used between encoder and decoder to assign weights to the encoded hidden states. The Attention mechanism can be used different tasks such as classification, even the mechanism was firstly proposed for machine-translation task.

Bahdanau Attention mechanism is mathematically represented by three different vectors, alignment scores, the weights and the context vector.

$$e_{t,i} = a(S_{t-1}, h_i) \tag{3.12}$$

Alignment score $e_{t,i}$ is calculated by using a feed forward neural network with encoded hidden states h_i and previous decoder output S_{t-1}. Briefly $a(.)$ can be represented by an activation function and the dot product of S_{t-1} and h_i.

$$\alpha_{t,i} = \text{softmax}(e_{t,i}) \tag{3.13}$$

Weights are calculated by softmax function to squeeze alignment scores between 0 and 1 as in Equation 3.13. The closer the weight towards 1 indicates that the state at that position is more important and needs more attention.

$$c_t = \sum_{n=1}^{T} \alpha_{t,i} h_i \tag{3.14}$$

Context vector is basically the summation of weighted states. After weight vector is calculated, importance of the each hidden state can be found by element wise multiplication. Then, the summation of weighted hidden state in each position i $\alpha_{t,i} * h_i$ is given as an output of the Attention mechanism to the decoder.

4. METHODOLOGY

We cast the SUMOylation prediction as a binary classification task in which the input is the 21-mer peptide sequence, x, centered on a lysine residue and the class label $y \in -1, 1$. Here, the label 1 indicates the positive class where the site is SUMOylated and -1 indicates the negative class. We propose three different alternative architectures SUMOnet-1, -2, and -3 and train these models in a supervised learning setup. We evaluate the performance of these models with three different peptide input representations.

Below, we first detail the dataset that we use in our experiments. Secondly, we describe the encoding methods. In the ensuing steps, we present the components of the SUMO-nets and the design process of our deep learning architectures SUMOnet-1, -2, and -3, evaluation metrics and how we performed hyper-parameter tuning on each model.

4.1 Dataset

We obtained the experimentally identified SUMOylation samples from the dbPTM database (Huang, Lee, Kao, Ma, Lee, Lin, Chang & Huang, 2019). We chose this data source for several reasons. It is comprehensive as it culls data from various biological databases. It provides an up-to-date non-homologous benchmark dataset. Although most prediction tools provide predictive performance results, there is a lack of standard, they use different datasets and most of them contain redundant sequences. By using this benchmark data, we hope others will also be able to compare our methods with theirs. We use the non-redundant benchmark dataset provided (version 2, download on 12.01.2021). dbPTM database curates this data with the CD-hit program (Li & Godzik, 2006) such that no sequence pair have similarity more than 40% (Huang et al., 2019). This ensures that the test data do

not include sequences very similar to sequences that the model have trained on and to avoid optimistic test performance estimates.

$$\mathbb{P} = A_{-10}A_{-9}A_{-8}.....A_{-2}A_{-1}KA_{+1}A_{+2}.....A_{+8}A_{+9}A_{+10} \tag{4.1}$$

The peptide sequences with the SUMOylated sites constitute the positive set in our classification task. The 21-mer peptides that are also centered on a lysine residue but is not reported to be a SUMOylation site constitute the negative examples. In Equation 4.1, $\mathbb{P}$ represents a peptide sequence which is centered with lysine 'K' and A_i represents amino acid residues on the position i. The dataset contains 1,432 proteins and 5,191 SUMOylation positive examples and 16,066 negative examples in total. This corresponds to a 1:3 ratio of positive-to negative class labels. We randomly held out 10% of the examples using stratified sampling for class labels and used it as test data to evaluate the models. The numbers for the train and test data are summarized in tab:data.

We hypertuned model parameters on the training data using 5-fold cross-validation. To further evaluate the models, we subset for the most challenging examples. For this, we looked for negatively labeled examples that contains the SUMOylation motifs and positive examples that are not SUMOylated. We also provide evaluation results on this harder test cases.

Dataset	Number of pos.	Number of neg	Pos:neg
ALL	5191	16066	1:3
Train	4672	14459	1:3
Test	519	1607	1:3

Table 4.1 The number of SUMOylation sites in dbPTM, positive (pos) and negative (neg) examples in train and test folds along with the positive-to-negative examples ratio are listed.

4.2 Peptide Encodings

To represent the input peptide sequence in a numerical vector space, we use three different encoding techniques: one-hot, BLOSUM62 (Henikoff & Henikoff, 1992) and NLF (Nanni & Lumini, 2011). Vector representations of lysine 'K' amino acid in

each different encoding illustrated in Figure 4.1.

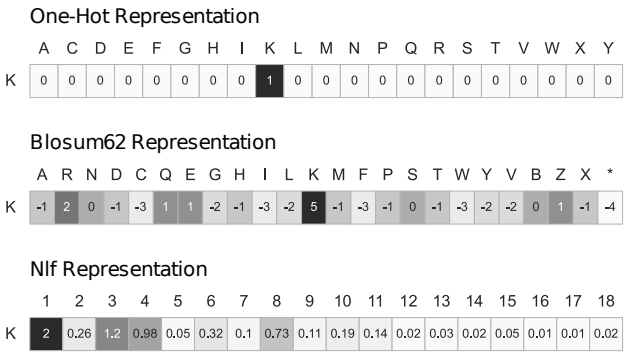

Figure 4.1 The three encodings are shown for the lysine 'K' amino acid.

Assume V is the N dimensional vector representation of an amino acid: Equation 4.2.

$$V = [v_1, v_2, v_3, ..., v_{N-2}, v_{N-1}, v_N] \tag{4.2}$$

Peptide sequences consist of 21 amino acids in our experiments. Each 21-mer peptites $\mathbb{P}$ can be represented with 21xN vectors by using Equation 4.2 as follows:

$$\mathbb{P} = \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ V_{20} \\ V_{21} \end{bmatrix} = \begin{bmatrix} v_{1,1} & v_{1,2} & . & . & . & v_{1,N-1} & v_{1,N} \\ v_{2,1} & v_{2,2} & . & . & . & v_{2,N-1} & v_{2,N} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ v_{20,1} & v_{20,2} & . & . & . & v_{20,N-1} & v_{20,N} \\ v_{21,1} & v_{21,2} & . & . & . & v_{21,N-1} & v_{21,N} \end{bmatrix} \tag{4.3}$$

The dimension N of the vectors is depended on the encoding mechanism. They are set as follows:

- **One-hot encoding:** Each candidate SUMOylation site is represented with a 21-mer peptide sequence centered on the lysine residue; hence, the input sequence is represented by a $21 * 21 = 441$ dimensional vector.
- **BLOSUM62 encoding**: We use the epitopepredict tool developed by (Farrell, 2021) to extract BLOSUM62 matrix for amino acids. The peptide se-

quences are represented by a $21 * 24 = 504$ dimensional vectors as Equation 4.1 for $N = 24$.

- **NLF encoding**: We directly use NLF factors to encode peptide sequences as input to the models. Each amino acid is represented by a vector of length 18. Hence, each protein sequence in our data is mapped to $21 * 18 = 378$ dimensional feature space as $N = 18$. Epitopepredict tool developed by (Farrell, 2021) is used for extracting NLF representation.

We treated the encoding scheme as an hyper-parameter and select the best performing one.

4.3 SUMOnets

We train novel deep learning architectures to classify SUMOylated and non-SUMOylated input peptide sequences. Each SUMOnet artitechure is coupled with the best performing encoding scheme for that specific architecture, which is decided during the hyper parameter tuning process. Our architectures consist of combinations of Convolutional layers (LeCun et al., 1989), Gated Recurrent Units (GRU) (Cho et al., 2014), Bi-Directional Gated Recurrent Units (Graves, Jaitly & Mohamed, 2013), Self-Attention mechanism (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser & Polosukhin, 2017) and Feed Forward Neural Network (Ojha et al., 2017).

All CNN layers are 1-D CNNs in all the architectures designed. Pooling layers, Average-Pooling, Max-Pooling and Global Average-Pooling are used to decrease model complexity. We apply ReLu activation function after each CNN and FFNN layers. GRU layers are activated by tanh and we estimate the target class probabilities using Softmax. We optimize weights of our models with Adam optimizer (Kingma & Ba, 2017) by minimizing cross-entropy loss. The three SUMO-net architectures are visualized in Figure 4.2. We use Keras (Cholletet al. , 2015) to implement our architectures.

4.3.1 SUMOnet-1

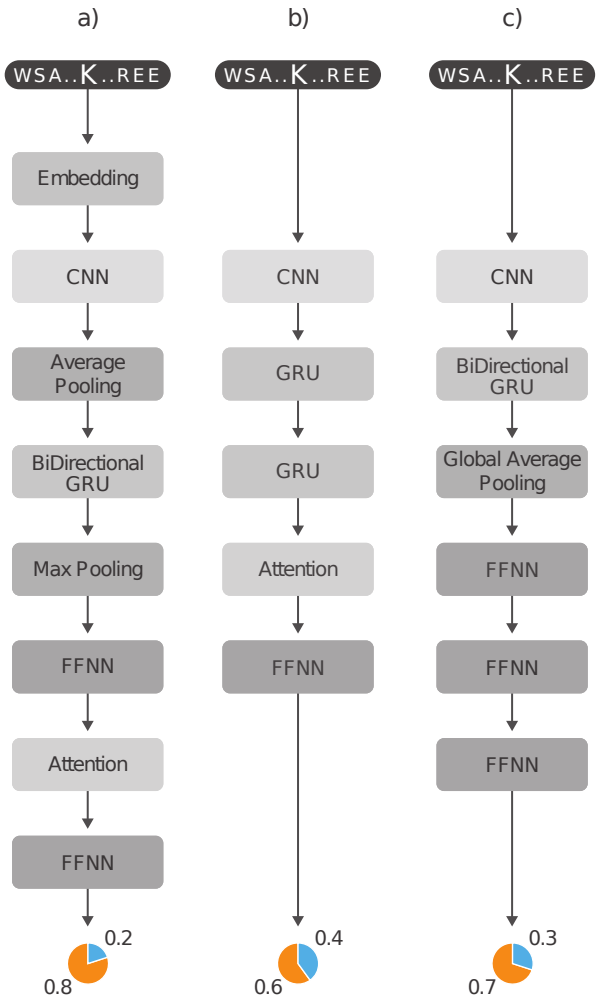

Figure 4.2 Deep learning model architectures with 21-mer peptide sequence centered on the lysine residue as input. **a)** SUMOnet-1 deep model architecture. **b)** SUMOnet-2 deep model architecture. **c)** SUMOnet-3 is the architecture that provides the best prediction results. Each architecture was selected with respect to mean of area under the curve of the receiver operating characteristic curves after 5-fold cross-validation.

SUMOnet-1 is a deep neural architecture that uses one-hot encoded vectors as an input. Each sequence is given into a trainable embedding layer to learn better vector representations for input peptides. Then, CNN layer extract the information among residues. The extracted features are averaged by using Average Pooling layer. The next layer is BiGRU; thus, network processes features in forward and backward directions. Max Pooling, FFNN and Self-Attention layers follows BiGRU for paying attention to the most important features. Finally, FFNN connects extracted features to the output layer.

After the hyper-parameter tuning, the embedding layer maps each amino acid to the 32-dimensional vector space. 128 filter maps are used in CNN layer and size of the each filter is 4. The number of memory units of BiGRU is determined as 32 for both forward and backward layers. FFNNs consist of 64 and 256 hidden units

respectively.

4.3.2 SUMOnet-2

The best peptide encoding scheme that performed best with the SUMOnet-2 artitechture was SUMOnet-2. Each input vector directly feeds to the CNN layer to extract residue features, which is followed by two GRU layer. Therefore, the extracted features by CNN are learned as sequentially and the positional information can be captured. Then, Self-Attention layer is used for attending to important features by giving weights to each output state of GRU. In the end, FFNN follows the Self-Attention as a final layer.

The hyper-parameters are set after the tuning process. CNN layer consists of 64 filters with kernel size of 3. We determined 32 units GRU layers and 128 hidden units for FFNN layer after the tuning process.

4.3.3 SUMOnet-3

SUMOnet-3 is an architecture that uses BLOSUM62 encodings as input vectors. Inputs are given to the CNN layer to capture features, which is followed by BiGRU, thus, network processes the captured features sequentially in forward and backward directions. Then, Global Average Pooling is used to calculate average of feature values and to decrease the number of trainable parameters, hence, over-fitting is reduced. Finally, three consecutive FFNN layers are used.

After the hyper-parameter tuning, the number of filters is determined as 128 with kernel size 2 for CNN layer after the hyper-parameter tuning. BiGRU consists of 32 memory units each in forwards and backwards layers. There are 64, 128 and 128 hidden units in FFNN layers respectively.

4.4 Architecture Design and Hyper-parameter Tuning

We firstly explain the experimental setup for architecture design. Then, the design process of each architecture is presented. Finally, after experimentally decided each layer in SUMOnet-1, -2, and -3, we mention our tuning steps for hyper-parameters of each model and hyper-parameters of training process.

4.4.1 Experimental Setup

We use the experimentally verified sumolyation sites from the dbPTM database (Huang et al., 2019). After separate out 10% of the training data as test data, we use stratified k-fold cross-validation method (Refaeilzadeh, Tang & Liu, 2009) to randomly split our imbalanced training data for tuning and design part. Thus, class distributions of training and validation data remain same. Moreover, the advantage comes from randomness by dividing the training data only once is decreased with k-fold cross-validation for evaluation of the models. We select k as 5 and area under the curve of the receiver operating characteristic curves to evaluate design and tuning results.

4.4.2 Architecture Design

Neural Units	Fixed Units
FFNN	128
1-D CNN	Filter Size = 120, Kernel Size = 2
RNN	64
LSTM	64
GRU	64
BiDirectional RNN/LSTM/GRU	64
Self-Attention	-
Max-Pooling	2
Average-Pooling	2
Global max/average Pooling	-
Dropout	0.3
Optimizer	Adam
Batch Size	32
Epoch Number	1000 + Early stoppings

Table 4.2 The neural units that we experimented to get final architectures and their fixed units. Kernel size and filter size for 1-D CNN. Number of cells for RNN, LSTM, GRU, and their bi-directional versions. Kernel size for Max-Pooling and Average Pooling. - means that there is no parameter to be tuned. Adam optimizer uses 0.001 learning rate.

We design three different architectures. Our experiments focus on several combinations of well known neural networks FFNN (Bebis & Georgiopoulos, 1994), CNN (LeCun et al., 1989), RNN (Quast, 2016), LSTM (Hochreiter & Schmidhuber, 1997), GRU (Chung, Gulcehre, Cho & Bengio, 2014) and Self-Attention (Zhang, Goodfellow, Metaxas & Odena, 2019) with pooling methods Max-Pooling, Average-Pooling and global versions of them from the review of (Gholamalinezhad & Khosravi, 2020). Further, we add dropout layer (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014) on different part of the architecture to prevent over-fitting.

To design our architectures, we left the number of units of each network optimizer, batch and epoch number fixed to be tuned later by considering the computational complexity as can be seen on Table 4.2. We use FFNN with 128 units, CNN with 120 units of filter and kernel size of 2, 64 units for each RNN, LSTM, GRU, pooling size of 2, Adam optimizer (Kingma & Ba, 2017), batch size of 32, 1000 as a number of epoch with early stopping by patience 5 to avoid over-fitting and embedding size of 100 for just architecture for one-hot encoded inputs. The starting point is the most shallow architecture for each encoded input vector. After that, we deepen the network by adding layers that improve the evaluation result one after the other.

4.4.3 Hyper-Parameter Tuning

We create a search space and tuning schedule for hyper-parameter tuning. Tuning schedule consists of searching the best parameter layer by layer instead of searching in all architectures. To do that, all parameters of each layer remain same as in Table 4.2 except the layer that tuned. Thus, computational complexity can be reduced for minimizing time to find best parameters for all architectures. We also tune the hyper-parameters of XGBoost and Logistic Regression to get the best prediction results.

We firstly tune hyper-parameters of XGBoost and Logistic Regression. The search space for the XGBoost is 100, 200, 300, 400, 500, 600, 700, 800, 900, 100 for number of estimators and 3, 4, 5, 6 for depth. The hyper parameter penalty is tuned for Logistic Regression with l1 and l2 penalties. In addition to the penalty, we also tune the parameter c for Logistic Regression starting from 0.0001 and we multiply each c value with 10 in each search to the ending point 1000 as can be seen on the Table 4.3.

Our search space for each SUMOnet architecture is also illustrated on Table 4.3.

	Tuning Object	Search Space
SUMOnet Layers	Dense Layer Unit	64, 128, 256
	CNN Filter Size	32, 64, 128, 256
	CNN Kernel Size	2, 3, 4
	GRU Unit Number	16, 32, 64, 128
	BiGRU Unit Number	16, 32, 64, 128
	Dropout	0.2, 0.25, 0.3, 0.35, 0.4, 0.45
	Embedding Layer	16, 32, 64, 128
Training Parameters	Optimizer	Adam, Nadam, Adadelta
	Batch Size	16, 32, 64, 128, 256
	Epoch Number	15, 25, 35, 45, 55
XGBoost	Number of Estimators	100, 200, ..., 900, 1000
	Depth	3, 4, 5, 6
Logistic Regression	Penalty	l1, l2
	C	0.0001, 0.001, ..., 100, 1000

Table 4.3 The search space for tuning hyper-parameters of each SUMOnet architecture, training components, XGBoost and Logistic Regression. We use default learning rates for each optimizer.

It involves dense layer units 64,128,256 for feed forward neural network, number of filters 32,64,128,256 and kernel size 2,3,4 for 1-D convolutional neural network, number of units 16,32,64,128 for gated recurrent units, 0.2,0.25,0.3,0.35,0.4,0.45 for dropout, and 16,32,64,128 for embedding layer. After tuning the hyper-parameters of architectures we search the best optimizer, batch size and epoch numbers for training of the models. Adam (Kingma & Ba, 2017), Nadam, Adadelta optimizers (Ruder, 2017), batch sizes of 16,32,64,128,256 and epoch numbers of 15,25,35,45,55 are tried to finalize the tuning part.

After the hyper-parameter tuning, the best parameters for XGBoost are 500 estimators and 6 for depth. The best parameters for Logistic Regression are l2 penalty and 1 for parameter c. The final architecture description for each model, SUMOnet-1, SUMOnet-2, and SUMOnet-3 is given in the Sec. 4.3.1, Sec. 4.3.2, Sec. 4.3.3 respectively.

4.5 Evaluation Metrics

We measure the performance of our models by using an independent test dataset that the model had never seen until the evaluation process. Moreover, evaluation metrics are selected with consideration of imbalanced behaviour of the test data to make a more straight forward comparison between different models. Matthews' correlation

coefficient (MCC) is well known technique in bioinformatics to evaluate models on imbalanced data (Boughorbel, Jarray & El-Anbari, 2017) and it is calculated as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4.4}$$

The correctly predicted SUMOylation sites are represented as TP (true positive) and TN (true negatives) are correctly predicted non-SUMOylation sites. The false predictions as SUMOylation sites and non-SUMOylation sites are FP (false positive) and FN (false negative) respectively. In addition to MCC we use F1-score to calculate the harmonic mean of precision and recall as follows:

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{4.5}$$

Further, we use area under the curve for receiver operating characteristics (AUC-ROC) and area under the precision recall curve (AUPR) to evaluate model performance for different thresholds due to the misleading evaluation of accuracy metric (Fawcett, 2006).

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.6}$$

Accuracy of the model would be high when the model favor the dominant class. Because the accuracy score is calculated by looking at how often model classify correctly both positive and negative classes as can be seen on Equation 4.6. Therefore, we investigate ROC and PR curves of the models and consider their auc scores as a summary of the curves.

5. RESULTS AND DISCUSSION

This chapter first describes the evaluation setup we use for comparing the performance of SUMOnets with other predictors. We finally compare SUMOnets with the state-of-the-art methods and with classical machine learning models, which we train and motif-based classifiers.

5.1 Evaluation Setup

5.1.1 Compared Methods

In this section, we present the evaluation setup on our neural architectures SUMOnet-1, -2, and -3. We compare SUMOnets with other models on a held-out test set. We also conduct hyperparameter tuning on the training data. The evaluation metrics, which are used for decisions are described in Sec. 4.5 and hyperparameter tuning steps are discussed in Sec. 4.4.3.

We compare SUMO-nets with three different sets of algorithms:

- **Motif-based rules:** This simple baseline relies solely on the known SUMOylation motifs. To understand what can be achieved by only using the known sequence motifs, we use rule-based classifiers that would return positive and negative labels based on the presence and absence of known SUMOlylation motifs, respectively. In this way, we obtain a representative level of achievement based on known motif-based rules.
- **Shallow learners:** To assess if there is any merit in using deep learning mod-

els for SUMOylation prediction, we compare SUMOnets with two machine learning methods, Logistic Regression (McCullagh & Nelder, 1989) and gradient boosted decision trees that we build and train. For the latter, we use the XGBoost implementation (Chen & Guestrin, 2016).

- **SUMOylation Predictors in Literature:** Finally, we compare SUMOnets with the widely available SUMOylation prediction tools as comprehensively as possible. The tools include GPS-SUMO (Zhao et al., 2014), pSumo-CD (Jia et al., 2016), Jassa (Beauclair et al., 2015) and SUMO-Forest (Qian et al., 2020). We attempted to expand our comparisons with other tools, but we were hindered by two constraints: the lack of readily available implementations and the lack of sufficient details in the descriptions to reproduce these tools independently by ourselves.

Below we describe how we used each of the compared methods.

Rule-based Classifier: The comprehensive summary of different motifs in literature is provided in (Beauclair et al., 2015). Each motif is scanned on the test examples to obtain results. Since these classifiers will return only binary class and no scores, only metrics that do not depend on the scores are calculated. We also use an ensemble classifier that would predict the positive class if only a single motif exists.

Shallow Learners: These models are trained with three different encoding methods on the training data. The models are trained with three different encodings. Hyperparameters of logistic regression and XGBoost models are tuned on training data using 5-fold cross-validation. The details of the hyperparameter tuning steps are provided in Sec. 4.4.3.

SUMO-Forest: We train SUMOForest on the training data using their available implementation at `https://github.com/sandyye666/SUMOForest`. The trained model is run on the test data to obtain the predicted class labels and scores.

GPS-SUMO: The GPS-SUMO (Zhao et al., 2014) source code is not available. Therefore, we use their webserver available at GPS-SUMO web server. The test data is uploaded to the webserver to obtain UMOylation site predictions. The web server provides predicted labels and the associated scores based on a predefined threshold value of the high, medium, all, low, and none. We apply a medium threshold to measure MCC and F1-Score according to the results by determining predicted labels. To compare ROC and AUPR curves, we use the scores obtained with the threshold setting `all`.

Jassa: The source code is not available therefore, we used the Jassa webserver at `http://www.jassa.fr/`. The server takes a single sequence as an input. To overcome this issue, we used a Python script to submit test sequences one by one and retrieved the predictions from the results page. Jassa server provides prediction labels and their scores with respect to thresholds `high` and `low` for three different clustering methods, all, directed and inverted. We used the clustering method all and applied both high and low thresholds separately to measure MCC and F1-Score according to the results by the determination of predicted labels. To be able to compare based on ROC and AUPR curves, we use the scores obtained when applying the clustering method `all`.

pSumo-CD (Jia et al., 2016) is another widely used SUMOylation predictor. Since the implementation is not available, we used `http://www.jci-bioinfo.cn/pSumo-CD` for finding SUMOylation sites. We extracted predicted labels for our test dataset by using their web server. For pSumo-CD, however, the server provides prediction labels but not the prediction scores. Thus, for that method, we could only calculate MCC and F1 metrics but not AUC or AUPR curves.

5.2 Prediction Performances

	Name	Motif	F1-Score	MCC
Consensus direct	Strong consensus	[Ψ_1]-[K]-[x]-[α]	0.308	0.336
	Consensus	[Ψ_2]-[K]-[x]-[α]	0.403	0.374
	Weak consensus	[Ψ_3]-[K]-[x]-[α]	0.418	0.373
	PDSM	[Ψ_2]-[K]-[x]-[α]-[x]$_2$-[S]-[P]	0.015	0.076
	NDSM	[Ψ_2]-[K]-[x]-[α]-[x]-[α]$_{2/6}$	0.158	0.237
	HCSM	[Ψ_4]$_3$-[K]-[x]-[E]	0.071	0.160
	SC-SUMO	[P/G]-[x]$_{(0-3)}$-[I/V]-[K]-[x]-[E]-[x]$_{(0-3)}$-[P/G]	0.063	0.158
	Minimal SC-SUMO	[I/V]-[K]-[x]-[E]-[x]$_{(0-3)}$-[P]	0.098	0.199
	SUMO-acetyl switch	[Ψ_2]-[K]-[x]-[α]-[P]	0.092	0.180
	pSuM	[Ψ_2]-[K]-[x]-[$_p$S]-[P]	0.000	0.000
Consensus inverted	Strong consensus	[α]-[x]-[K]-[Ψ_1]	0.077	0.068
	Consensus	[α]-[x]-[K]-[Ψ_2]	0.151	0.108
	Weak consensus	[α]-[x]-[K]-[Ψ_3]	0.176	0.119
Non-Consensus	Ensemble motif		0.469	0.359

Table 5.1 Evaluation results of motifs in Jassa (Beauclair et al., 2015) via rule-based method on independent test set. The protein sequence is predicted as a SUMOylation site if a corresponding motif exists. For Non-Consensus, the protein site is predicted as SUMOylated if the protein sequence consists of at least one motif. Ψ_1 = I, L, V; Ψ_2 = A, F, I, L, M, P,V, W; Ψ_3 = A, F, G, I, L, M, P, V, W, Y. α = D, E, $_p$S/T = phosphorylated serine/threonine.

Separately for each motif, its presence in a test sample is checked, and the sequence

is predicted as a SUMOylation site if the test sequence bears the motif. In addition to individual comparisons, the ensemble of all motifs was also evaluated by checking whether a particular protein sequence includes any of the motifs. The maximum F1 score that can be achieved with a single motif is 0.418, which the consensus motif classifier yields. The MCC motif score of this motif is also high, 0.373. The consensus motif also achieves 0.403 F1 and 0.374 MCC scores. The prediction performance for ensemble, weak consensus, consensus and strong consensus methods are far better than other consensus direct and consensus inverted motifs. Most methods lead to very small F1 scores and MCC scores (Table 5.1). Even when methods are combined, their F1 scores and MCC scores remain at 0.469 and 0.359, respectively. These results establish that the linear sequence motifs are not sufficient to predict all the SUMOylated sites effectively.

We next evaluated the SUMO-nets with the other two sets of methods, the shallow models and the available SUMOylation predictors.

Predictor	Encoding	F1-Score	MCC	AUC	AUPR
XGBoost	BLOSUM62	0.614	0.536	0.844	0.727
LogisticRegression	BLOSUM62	0.591	0.509	0.827	0.696
GPS-SUMO	-	0.428	0.157	0.707	0.569
Jassa-Low	-	0.403	0.380	0.727	0.560
Jassa-High	-	0.256	0.305	0.727	0.560
pSumo-CD	-	0.332	0.215	-	-
SUMO-Forest	-	0.592	0.502	0.819	0.688
SUMOnet-1	One-Hot	0.640	0.544	0.859	0.742
SUMOnet-2	NLF	0.623	0.547	0.857	0.741
SUMOnet-3	BLOSUM62	**0.658**	**0.569**	**0.870**	**0.758**

Table 5.2 Comparison of the SUMOylation prediction methods on the whole test data. Jassa predicts labels with respect to two different thresholds, low and high. pSumo-CD (Jia et al., 2016) gives the predicted labels without any score; therefore, ROC-AUC and AUPR scores cannot be calculated. Encodings that give the best results are listed for the implemented models. Since the services/codes provided by the models in the literature do the encoding process automatically, we did not specify them.

Table 5.2 shows the evaluation metrics computed on the test data for all the models. SUMO-Forest is the best predictor among the existing models in all evaluation metrics. SUMO-Forest achieves 0.59 F1 score, which is approximately 14% higher than Jassa , 17% higher than GPS-SUMO and 26% higher than pSumo-CD. SUMO-Forest achieves 0.50 MCC score, which corresponds to 25-30% more than other tools in literature. Some of the evaluation metrics for pSumo-CD could not be measured; therefore, our interpretation is limited for this method. The ROC-AUC and AUPR scores of SUMO-Forest are 0.82 and 0.69 respectively, which are approximately 11% higher than that of GPS-SUMO's. Thus, we conclude that SUMO-Forest performs best among the compared available tools in the literature on these datasets.

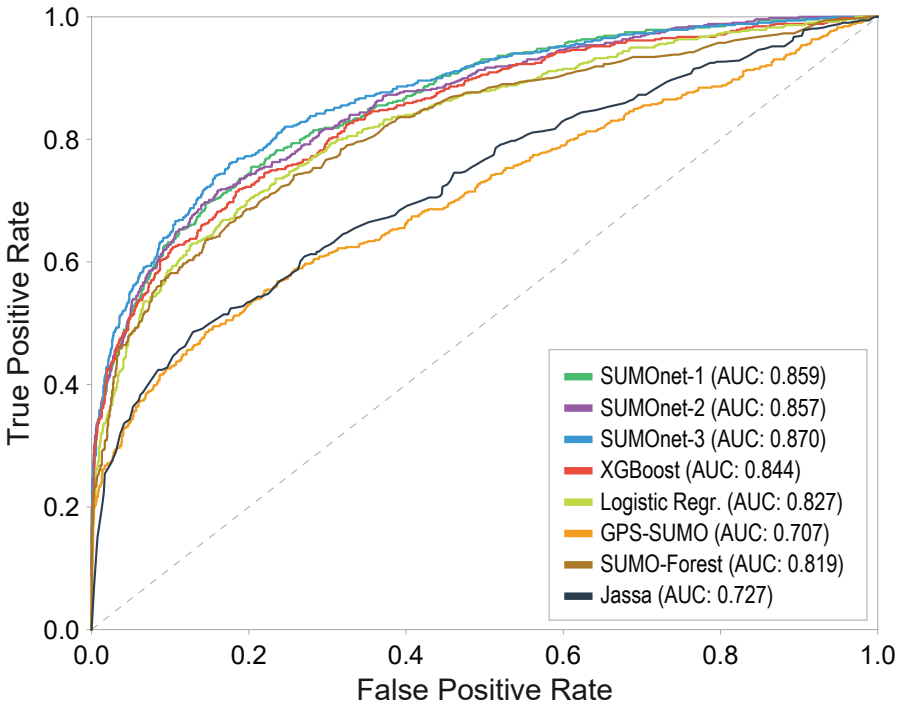

Figure 5.1 Receiver operating characteristic curve. False positive and true positive rates are evaluated on our independent test data.

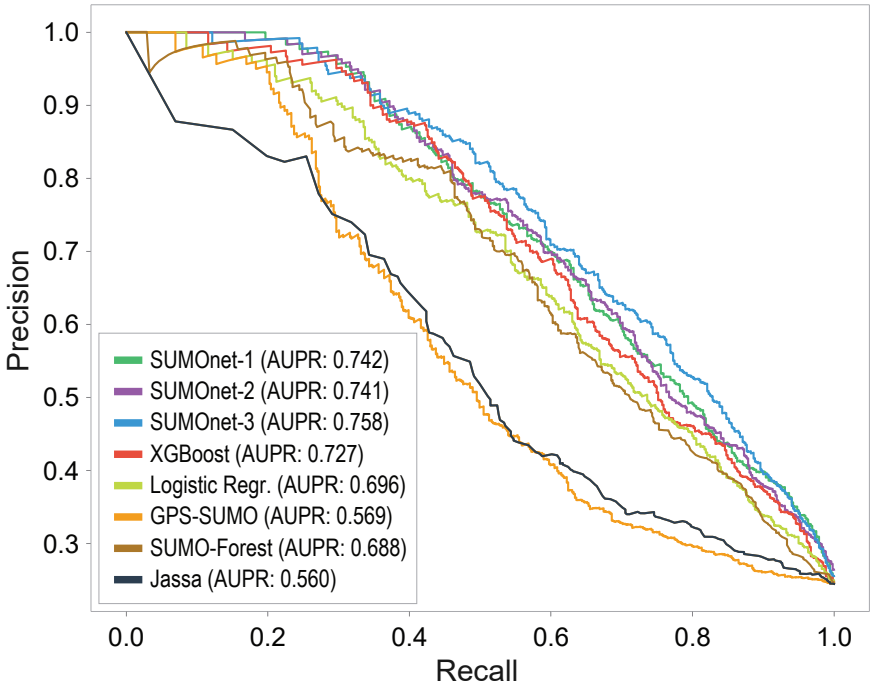

Figure 5.2 Precision recall curve. Precision and recall scores are evaluated on our independent test data.

As we mentioned in the discussion of the evaluation setup Sec. 5.1, we experimented with Logistic Regression and XGBoost to predict SUMOylation sites to create strong baselines in our evaluation. Of the three peptide encodings, both models provide

the best performance with BLOSUM62 encoding. XGBoost produces about 2% improvement in performance for all the evaluation metrics over Logistic Regression. For the XGBoost classifier, the F1-Score and ROC-AUC are found to be 0.61, and 0.84, respectively, which are 2% higher than SUMO-Forest. The MCC and AUPR scores are 0.54 and 0.72, respectively. This corresponds to 3% increase compared to the best tool in the previous Comparison, SUMO-Forest. Thus, we conclude that a SUMOylation predictor better than the existing models can be trained with an XGBoost classifier.

Finally, we examine SUMOnet prediction performance (Table 5.2). All the three SUMOnets yield better prediction performance than the other models with respect ro the used evaluation metrics. Among the three architectures, SUMOnet-3 is the one that achieves the best scores. The F1-Score of SUMOnet-3 is 0.66, which is 5% more than XGBoost classifier (Chen & Guestrin, 2016). MCC, ROC-AUC and AUPR scores of SUMOnet-3 are 0.57, 0.87 and 0.76, respectively. These correspond to approximately 3% improvement over the XGBoost classifier. We conclude that we can attain the best predictor using SUMO-net3.

We also compare models using the Receiver operating characteristic (ROC) and precision-recall (PR) curves. As shown in Figure 5.1 SUMOnet-3 is the best predictor across the different false positive rates. Similarly, it achieves the best precision values across different recall ranges (Figure 5.2)

5.3 Evaluation on Hard Test Examples

Predictor	Encoding	F1-Score	MCC	AUC	AUPR
XGBoost	BLOSUM62	0.438	0.0838	0.523	0.742
LogisticRegression	BLOSUM62	0.410	0.0598	0.507	0.740
GPS-SUMO	-	0.357	-0.491	0.321	0.605
Jassa-Low	-	0.028	-0.364	0.240	0.564
Jassa-High	-	0.000	-0.259	0.240	0.564
pSumo-CD	-	0.194	-0.091	-	-
SUMO-Forest	-	0.460	0.101	0.533	0.762
SUMOnet-1	One-Hot	0.547	0.152	0.657	**0.822**
SUMOnet-2	NLF	0.479	0.118	0.601	0.787
SUMOnet-3	BLOSUM62	**0.565**	**0.168**	**0.667**	0.819

Table 5.3 Comparison of the SUMOylation prediction methods on the hard test examples. Hard test examples are the subset of our test data set. We select positive samples which have no motifs and negative samples which have any motif.

As shown in Table 5.1, it is possible to achieve around a 0.4 F1 score based on motifs. The SUMOylation sites that contain these motifs are easy to capture. Thus, to evaluate different methods, we subset the most challenging examples in the test set based on the presence and absence of the SUMO motifs. The hard test examples set includes sequences that lack any of the SUMO motifs but are positively labeled and negatively labeled sequences that include SUMO motifs. These are the specific examples where we expect the methods to have difficulty in the predictions.

The ROC curve in Figure 5.3 shows the comparison of various methods. As expected, overall performance degrades for all methods; however, compared SUMOnet models perform the best among all methods. Particularly, our final choice, SUMOnet-3 is the best performer.

5.4 Ablation Study on SUMOnet-3

To asses the value that each component bring, we conduct an evaluation of SUMOnet-3 where we add the components gradually and re-evaluate each model. Each model is evaluated using 5-fold cross validation on the training data.

Method	F1-Score	MCC	AUC	AUPR
Dense[2]	0.538	0.492	0.831	0.697
Conv[120,2]_Dense[2]	0.584	0.507	0.837	0.709
Conv[120,2]_BiGRU[32]_Dense[2]	0.633	0.545	0.864	0.745
Conv[120,2]_BiGRU[32]_GlobalAv.Pool_Dense[2]	0.628	0.551	0.868	0.751
Conv[120,2]_BiGRU[32]_GlobalAv.Pool_Dense[128,128,128,2]	**0.640**	**0.557**	**0.873**	**0.756**

Table 5.4 Evaluation results of each model as we add components of SUMOnet-3 in the artitechure. The table reports 5-fold cross validation average scores on the training data. The BLOSUM62 encodings are used for input representation.

The starting point for the architecture construction is a single layer perceptron that contains 2 hidden units. When the the CNN is added, we see an increase in all evaluation metrics. The largest increase is observed when BiGRU is added after the CNN with nearly 3% in ROC-AUC and 4% in AUPR. There is a small raise in all scores after adding Global Average Pooling and three consecutive dense layers.

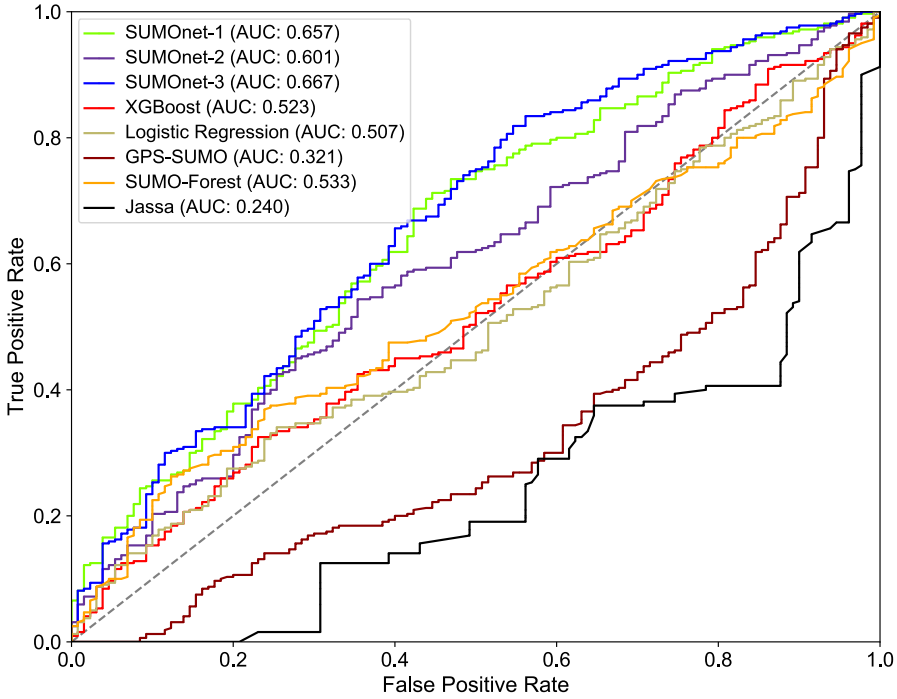

Figure 5.3 Receiver operating characteristic curve on hard test samples, which is a subset of our independent test data. Hard test samples includes sequences that lack any of the SUMO motifs but are positively labeled and negatively labeled sequences that include SUMO motifs.

.5

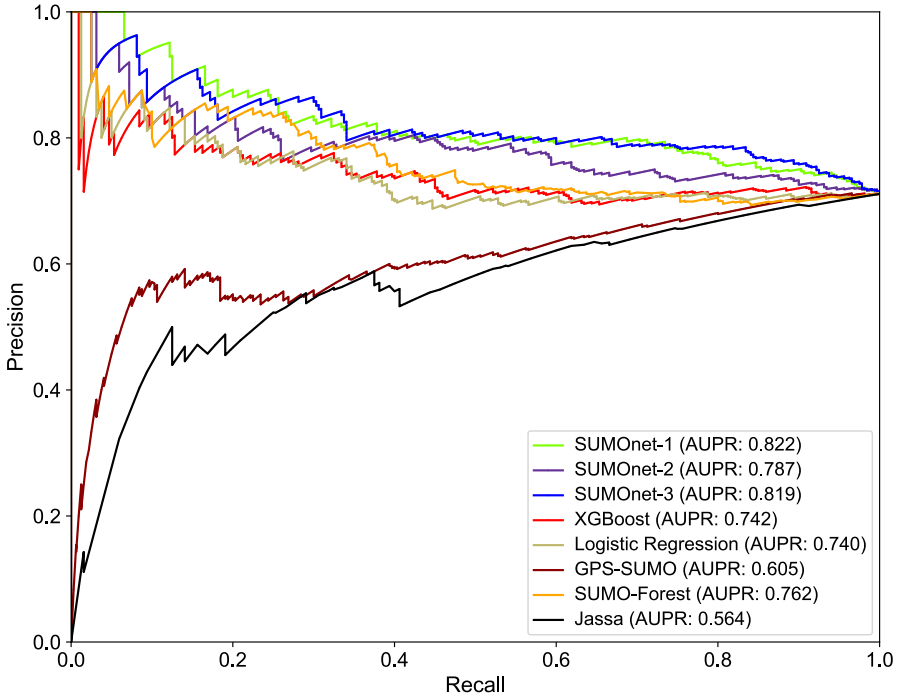

Figure 5.4 Precision recall curve on hard test samples, which is a subset of our independent test data. Hard test samples includes sequences that lack any of the SUMO motifs but are positively labeled and negatively labeled sequences that include SUMO motifs.

6. SUMONET: A PYTHON LIBRARY TO PREDICT SUMOYLATION SITES

In this section, we present `sumonet` Python library for the prediction of SUMOylation sites. The main motivation behind implementing `sumonet` is to support SUMOylation research with a user-friendly and capable computational tool that is freely accessible. We also want to facilitate the reproducibility of the experiments. `sumonet` library supports the following experiments:

- The SUMOylation dataset can be used entirely, or it can be sampled randomly.
- Users can choose from the three encoding methods: one-hot, BLOSUM62, or NLF encoding.
- SUMOnet-3 architecture is ready to use for training with randomly initialized weights to train with a new dataset
- Our pre-trained SUMOnet-3 model is ready for making predictions on protein sequences
- Libray reports F-1 Score, MCC, ROC, and AUPR evaluation metrics
- You can extract our train and test sequences to compare with your data

Using the above features, users can either reproduce our experiments with SUMOnet3 or train new models with their own data.

The library can be used by cloning from GitHub `https://github.com/berkedilekoglu/SUMOnet` or it can be installed by `'pip installsumonet==0.1'`. We also provide a tutorial in the GitHub repository for the training of end users.

6.1 Structure of the Library

`sumonet` library consists of modules utils, model, and evaluation to provide data processing, deep model training, and evaluation capabilities, respectively. This section explains each module and the data directory in detail.

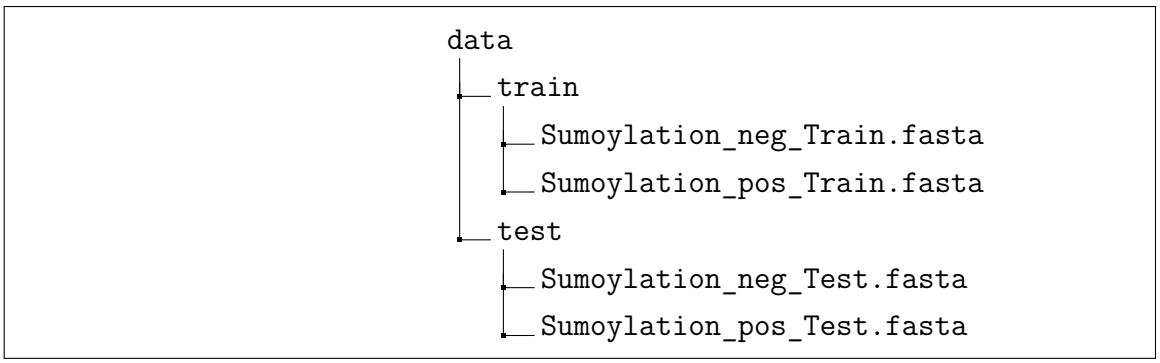

Figure 6.1 Directory tree of the data

The data directory consists of an up-to-date benchmark dataset obtained from dbPTM (Huang et al., 2019). The data folder includes two subfolders which serve as train and test directories, as can be seen on Figure 6.1. We provide positive and negative samples that are used in our experiments for both training and test processes. Thus, users can use these data to train their own models. Moreover, they can use this information to exclude specific test examples that are already in our training data for further experimentation.

6.1.1 Utils Module

The Utils module contains two classes to prepare data for training and testing. These classes enable loading, encoding, and pre-processing of the data. There is also a scaler directory, which consists of pre-trained `min-max` scaler (Figure 6.1.1). Min-max scaler is applied to our training data to avoid information leaks from test data to training.

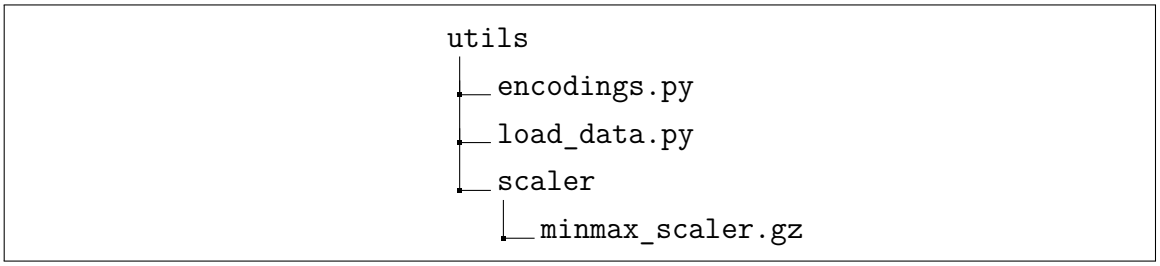

Figure 6.2 Directory tree of the utils module

`Encodings.py` script contains an encoding class for the pre-processing of a given data set. The user can give a 21-mer peptide sequence or a data path in Fasta format to the class object; thus, one-hot, NLF, or BLOSUM62 vector representations are taken as an output.

Load_data.py script is for loading dbPTM (Huang et al., 2019) data-set from the data directory on Figure 6.1. Hence, our experiments can be replicated, or samples can be compared for experimental purposes.

6.1.2 Evaluation Module

Figure 6.3 Directory tree of the evaluation module

```
evaluation
└── metrics.py
```

The evaluation module consists of a python script metrics.py, as can be seen on Figure 6.1.2. The evaluation metrics F1-score, MCC, ROC, and AUPR are defined and combined in a function for easy usage. Hence, the user can either use all or one to evaluate the prediction results.

6.1.3 Model Module

Figure 6.4 Directory tree of the model module

```
model
├── architecture.py
└── pretrained
    └── sumonet3.h5
```

The model module includes `architecture.py` script and a pre-trained directory, which consists of a pre-trained model with the weights of SUMOnet-3. The architecture of SUMOnet-3 is provided by using a superclass of the Keras Model. Hence, all of the functions in Keras models can be used for our architecture class.

The pre-trained model weights can be used to reproduce our results or make predictions on any test data. Also, SUMOnet-3 architecture can be used with randomly initialized weights for training from scratch.

7. CONCLUSION

Post-translational modification (PTM) is a chemical modification critical for the regulation of cellular processes. SUMOylation is one of the major protein modifications in which small ubiquitin-like modifiers (SUMOs) covalently and reversibly attach to specific lysine (K) residues of target proteins. The SUMOylation process is conserved across eukaryotes, underlining its essentiality for the cell. Indeed, SUMOylation has been reported to take several critical processes, such as cell cycle regulation and DNA repairing, and its aberrations in SUMOylation is associated with several diseases. Therefore, accurate identification of SUMOylation sites is critical.

SUMO proteins bind to their targets in a site-specific way; thus, the sequence around the SUMOYlated lysine residue contains the information to predict the sites. Based on this, several SUMOylation predictors are developed that, given a lysine residue and its surrounding sequence, would predict whether the lysine will be SUMOYlated or not. However, most of these methods are not evaluated on a benchmark dataset. In this work, we developed state-of-the-art SUMOylation predictors, SUMOnets, that surpass the closest best SUMOylation predictor.

We present three different deep learning architectures, SUMOnet-1, -2, and -3 to predict SUMOylation sites accurately. Various neural units that are known to perform well in NLP tasks, such as CNN, GRU, and attention, are used to construct these architectures. We experimented with three encoding schemes: one-hot, NLF, and BLOSUM62 for peptide representation. We provide the most accurate deep learning architectures for these encoding techniques.

When compared on an experimental benchmark dataset, the predictors surpass the next best SUMOylation predictor by approximately 5% AUPR. Among the three architectures, SUMO-net3 achieves the best performance across all metrics. When the methods are compared on a challenging subset of the test data, the performance improvement over the existing SUMOylation predictors becomes more evident.

For our evaluation experiments, we also build a rule-based prediction method on

our independent test data. The rule-based classification is a method that uses motifs to decide whether a protein sequence is SUMOylated or not. We show that motifs only are not capable enough to predict SUMOylation sites. We also trained XGBoost and Logistic regression models to form a strong baseline. We show that it is possible to attain SUMOylation predictors better than the existing ones with XGBoost and BLOSSUM62. SUMOnet-3 is the best SUMOylation site prediction tool in all evaluation experiments. We provide SUMOnet-3 as an open-source project in GitHub and a Python library that can be easily installed by pypi. The library also provides means to reproduce our results for further evaluation.

The work can be extended in future directions. One immediate plan is to crawl up-to-date SUMOylation sites from the newly published papers to construct an additional dataset for SUMOnet-3. A second future work is to build a web-server for SUMOnet-3 predictions. The peptide site can be submitted to the server and the predicted label and the prediction score will be output. Thus, users can easily access the predictions seamlessly. In this work, the input sequence is a 21-mer peptide sequence centered on the candidate lysine residue. One future work involves inspecting whether having a longer input peptide could improve the model performance. Our initial experiments with protein sequence embedding methods with ProtVecAsgari & Mofrad (2015) did not outperform the existing embedding methods we used. Therefore, we had stopped exploring those directions. However, there has been progressing in protein sequence embedding methods. It would be interesting to explore whether they can further improve the prediction performance. Lastly, similar architectures could be useful for other PTMs site predictions. Future work would be to assess these architectures for other PTMs, such as ubiquitylation.

The work also has limitations. Currently, the model relies solely on sequence information. Protein structural features, as well as conservation information, could help improve the model. One limitation of current artitechure is that it lacks an explanation module. It will be interesting to understand which sequence positions are critical for the positions. Such interpratations may lead to the discovery of unknown SUMOylation motifs that were not possible to capture with traditional methods.

BIBLIOGRAPHY

Asgari, E. & Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, *10*(11), e0141287.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. & LeCun, Y. (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Beauclair, G., Bridier-Nahmias, A., Zagury, J.-F., Saïb, A., & Zamborlini, A. (2015). Jassa: a comprehensive tool for prediction of sumoylation sites and sims. *Bioinformatics*, *31*(21), 3483–3491.

Bebis, G. & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, *13*(4), 27–31.

Benson, M. D., Li, Q.-J., Kieckhafer, K., Dudek, D., Whorton, M. R., Sunahara, R. K., Iñiguez-Lluhí, J. A., & Martens, J. R. (2007). Sumo modification regulates inactivation of the voltage-gated potassium channel kv1.5. *Proc Natl Acad Sci U S A*, *104*(6), 1805–1810.

Bode, A. M. & Dong, Z. (2004). Post-translational modification of p53 in tumorigenesis. *Nat Rev Cancer*, *4*(10), 793–805.

Boughorbel, S., Jarray, F., & El-Anbari, M. (2017). Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PLOS ONE*, *12*(6), 1–17.

Celen, A. B. & Sahin, U. (2020). Sumoylation on its 25th anniversary: mechanisms, pathology, and emerging concepts. *The FEBS Journal*, *287*(15), 3110–3140.

Chen, T. & Guestrin, C. (2016). Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1724–1734)., Doha, Qatar. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Chollet, F. et al. (2015). Keras.

Chou, K. C. & Maggiora, G. M. (1998). Domain structural class prediction. *Protein Engineering, Design and Selection*, *11*(7), 523–538.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Denison, C., Rudner, A. D., Gerber, S. A., Bakalarski, C. E., Moazed, D., & Gygi, S. P. (2005). A proteomic strategy for gaining insights into protein sumoylation in yeast*s. *Molecular & Cellular Proteomics*, *4*(3), 246–254.

Dongen, S. (2000). Graph clustering by flow simulation. *PhD thesis, Center for Math and Computer Science (CWI)*.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, (pp. 39–43).

Farrell, D. (2021). epitopepredict: A tool for integrated mhc binding prediction. *bioRxiv*.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, *27*(8), 861–874. ROC Analysis in Pattern Recognition.

Filosa, G., Barabino, S. M., & Bachi, A. (2013). Proteomics strategies to identify sumo targets and acceptor sites: a survey of rna-binding proteins sumoylation. *Neuromolecular medicine*, *15*(4), 661–676.

Flotho, A. & Melchior, F. (2013). Sumoylation: a regulatory protein modification in health and disease. *Annual review of biochemistry*, *82*, 357–385.

Freund, Y. & Schapire, R. E. (1999). A short introduction to boosting.

Geiss-Friedlander, R. & Melchior, F. (2007). Concepts in sumoylation: a decade on. *Nature Reviews Molecular Cell Biology*, *8*(12), 947–956.

Gholamalinezhad, H. & Khosravi, H. (2020). Pooling methods in deep neural networks, a review.

Golebiowski, F., Matic, I., Tatham, M. H., Cole, C., Yin, Y., Nakamura, A., Cox, J., Barton, G. J., Mann, M., & Hay, R. T. (2009). System-wide changes to sumo modifications in response to heat shock. *Science signaling*, *2*(72), ra24–ra24.

Graves, A., Jaitly, N., & Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, (pp. 273–278).

He, F., Wang, R., Gao, Y., Wang, D., Yu, Y., Xu, D., & Zhao, X. (2019). Protein ubiquitylation and sumoylation site prediction based on ensemble and transfer learning, 117–123.

Hendriks, I. A., D'souza, R. C., Yang, B., Verlaan-de Vries, M., Mann, M., & Vertegaal, A. C. (2014). Uncovering global sumoylation signaling networks in a site-specific manner. *Nature structural & molecular biology*, *21*(10), 927–936.

Henikoff, S. & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, *89*(22), 10915–10919.

Hietakangas, V., Anckar, J., Blomster, H. A., Fujimoto, M., Palvimo, J. J., Nakai, A., & Sistonen, L. (2006). Pdsm, a motif for phosphorylation-dependent sumo modification. *Proc Natl Acad Sci U S A*, *103*(1), 45–50.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780.

Huang, K.-Y., Lee, T.-Y., Kao, H.-J., Ma, C.-T., Lee, C.-C., Lin, T.-H., Chang, W.-C., & Huang, H.-D. (2019). dbptm in 2019: exploring disease association and cross-talk of post-translational modifications. *Nucleic Acids Res*, *47*(D1), D298–D308.

Ivanov, A. V., Peng, H., Yurchenko, V., Yap, K. L., Negorev, D. G., Schultz, D. C., Psulkowski, E., Fredericks, W. J., White, D. E., Maul, G. G., Sadofsky, M. J., Zhou, M.-M., & Rauscher, F. J. r. (2007). Phd domain-mediated e3 ligase activity directs intramolecular sumoylation of an adjacent bromodomain required for gene silencing. *Mol Cell*, *28*(5), 823–837.

Jackson, S. P. & Durocher, D. (2013). Regulation of dna damage responses by ubiquitin and sumo. *Molecular cell*, *49*(5), 795–807.

Jia, J., Zhang, L., Liu, Z., Xiao, X., & Chou, K.-C. (2016). psumo-cd: predicting sumoylation sites in proteins with covariance discriminant algorithm by incorporating sequence-coupled effects into general pseaac. *Bioinformatics*, *32*(20),

3133–3141.

Kingma, D. P. & Ba, J. (2017). Adam: A method for stochastic optimization.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*, 107398.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.

Lee, L., Sakurai, M., Matsuzaki, S., Arancio, O., & Fraser, P. (2013). Sumo and alzheimer's disease. *Neuromolecular medicine*, *15*(4), 720–736.

Li, W. & Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, *22*(13), 1658–1659.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network.

López, Y., Dehzangi, A., Reddy, H. M., & Sharma, A. (2020). C-isumo: A sumoylation site predictor that incorporates intrinsic characteristics of amino acid sequences. *Computational Biology and Chemistry*, *87*, 107235.

Mahajan, R., Delphin, C., Guan, T., Gerace, L., & Melchior, F. (1997). A small ubiquitin-related polypeptide involved in targeting rangap1 to nuclear pore complex protein ranbp2. *Cell*, *88*(1), 97–107.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, *2*, 49–55.

Matic, I., Schimmel, J., Hendriks, I. A., van Santen, M. A., van de Rijke, F., van Dam, H., Gnad, F., Mann, M., & Vertegaal, A. C. O. (2010). Site-specific identification of sumo-2 targets in cells reveals an inverted sumoylation motif and a hydrophobic cluster sumoylation motif. *Mol Cell*, *39*(4), 641–652.

Matunis, M. J., Coutavas, E., & Blobel, G. (1996). A novel ubiquitin-like modification modulates the partitioning of the ran-gtpase-activating protein rangap1 between the cytosol and the nuclear pore complex. *The Journal of cell biology*, *135*(6), 1457–1470.

McCullagh, P. & Nelder, J. (1989). *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall.

Melchior, F. (2000). Sumo–nonclassical ubiquitin. *Annu Rev Cell Dev Biol*, *16*, 591–626.

Nanni, L. & Lumini, A. (2011). A new encoding technique for peptide classification. *Expert Systems with Applications*, *38*(4), 3185–3191.

Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, *60*, 97–116.

Picard, N., Caron, V., Bilodeau, S., Sanchez, M., Mascle, X., Aubry, M., & Tremblay, A. (2012). Identification of estrogen receptor as a sumo-1 target reveals a novel phosphorylated sumoylation motif and regulation by glycogen synthase kinase 3. *Mol Cell Biol*, *32*(14), 2709–2721.

Qian, Y., Ye, S., Zhang, Y., & Zhang, J. (2020). Sumo-forest: A cascade forest based method for the prediction of sumoylation sites on imbalanced data. *Gene*, *741*, 144536.

Quast, B. (2016). rnn: a recurrent neural network in r. *Working Papers*.

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation, 532–538.

Ren, J., Gao, X., Jin, C., Zhu, M., Wang, X., Shaw, A., Wen, L., Yao, X., & Xue, Y. (2009). Systematic study of protein sumoylation: Development of a site-specific predictor of sumosp 2.0. *PROTEOMICS*, *9*(12), 3409–3412.

Rodriguez, M. S., Dargemont, C., & Hay, R. T. (2001). Sumo-1 conjugation in vivo requires both a consensus modification motif and nuclear targeting. *J Biol Chem*, *276*(16), 12654–12659.

Ruder, S. (2017). An overview of gradient descent optimization algorithms.

Sampson, D. A., Wang, M., & Matunis, M. J. (2001). The small ubiquitin-like modifier-1 (sumo-1) consensus sequence mediates ubc9 binding and is essential for sumo-1 modification. *J Biol Chem*, *276*(24), 21664–21669.

Schwartz, D. & Gygi, S. P. (2005). An iterative statistical approach to the identification of protein phosphorylation motifs from large-scale data sets. *Nature Biotechnology*, *23*(11), 1391–1398.

Seeler, J.-S. & Dejean, A. (2017). Sumo and the robustness of cancer. *Nat Rev Cancer*, *17*(3), 184–197.

Shatravin, V., Shashev, D., & Shidlovskiy, S. (2022). Sigmoid activation implementation for neural networks hardware accelerators based on reconfigurable computing environments for low-power intelligent systems. *Applied Sciences*, *12*(10).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958.

Stankovic-Valentin, N., Deltour, S., Seeler, J., Pinte, S., Vergoten, G., Guérardel, C., Dejean, A., & Leprince, D. (2007). An acetylation/deacetylation-sumoylation switch through a phylogenetically conserved psikxep motif in the tumor suppressor hic1 regulates transcriptional repression activity. *Mol Cell Biol*, *27*(7), 2661–2675.

Suárez-Paniagua, V. & Segura-Bedmar, I. (2018). Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction. *BMC Bioinformatics*, *19*(8), 209.

Subramanian, L., Benson, M. D., & Iñiguez-Lluhí, J. A. (2003). A synergy control motif within the attenuator domain of ccaat/enhancer-binding protein alpha inhibits transcriptional synergy through its piasy-enhanced modification by sumo-1 or sumo-3. *J Biol Chem*, *278*(11), 9134–9141.

Szandala, T. (2020). Review and comparison of commonly used activation functions for deep neural networks. *CoRR*, *abs/2010.09458*.

Tammsalu, T., Matic, I., Jaffray, E. G., Ibrahim, A. F., Tatham, M. H., & Hay, R. T. (2015). Proteome-wide identification of sumo modification sites by mass spectrometry. *Nature protocols*, *10*(9), 1374–1388.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need.

Vertegaal, A. C., Andersen, J. S., Ogg, S. C., Hay, R. T., Mann, M., & Lamond, A. I. (2006). Distinct and overlapping sets of sumo-1 and sumo-2 target proteins revealed by quantitative proteomics* s. *Molecular & Cellular Proteomics*, *5*(12), 2298–2310.

Walsh, C. T., Garneau-Tsodikova, S., & Gatto Jr, G. J. (2005). Protein posttranslational modifications: the chemistry of proteome diversifications. *Angewandte*

Chemie International Edition, *44*(45), 7342–7372.

Xue, Y., Liu, Z., Gao, X., Jin, C., Wen, L., Yao, X., & Ren, J. (2010). Gps-sno: computational prediction of protein s-nitrosylation sites with a modified gps algorithm. *PLoS One*, *5*(6), e11290.

Xue, Y., Ren, J., Gao, X., Jin, C., Wen, L., & Yao, X. (2008). Gps 2.0, a tool to predict kinase-specific phosphorylation sites in hierarchy. *Mol Cell Proteomics*, *7*(9), 1598–1608.

Xue, Y., Zhou, F., Fu, C., Xu, Y., & Yao, X. (2006). Sumosp: a web server for sumoylation site prediction. *Nucleic Acids Res*, *34*(Web Server issue), W254–7.

Yang, S.-H., Galanis, A., Witty, J., & Sharrocks, A. D. (2006). An extended consensus motif enhances the specificity of substrate modification by sumo. *EMBO J*, *25*(21), 5083–5093.

Yang, Y., Heffernan, R., Paliwal, K., Lyons, J., Dehzangi, I. A., Sharma, A., Wang, J., Sattar, A., & Zhou, Y. (2017). *SPIDER2: A Package to Predict Secondary Structure, Accessible Surface Area, and Main-Chain Torsional Angles by Deep Neural Networks*, volume 1484, (pp. 55–63).

Yen, S.-J. & Lee, Y.-S. (2006). *Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset*, (pp. 731–740). Berlin, Heidelberg: Springer Berlin Heidelberg.

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019). Self-attention generative adversarial networks.

Zhang, J., Chen, Z., Zhou, Z., Yang, P., & Wang, C.-Y. (2017). Sumoylation modulates the susceptibility to type 1 diabetes. *SUMO Regulation of Cellular Processes*, 299–322.

Zhao, Q., Xie, Y., Zheng, Y., Jiang, S., Liu, W., Mu, W., Liu, Z., Zhao, Y., Xue, Y., & Ren, J. (2014). Gps-sumo: a tool for the prediction of sumoylation sites and sumo-interaction motifs. *Nucleic acids research*, *42*(Web Server issue), W325–W330.

Zhou, F.-F., Xue, Y., Chen, G.-L., & Yao, X. (2004). Gps: a novel group-based phosphorylation predicting and scoring method. *Biochemical and Biophysical Research Communications*, *325*(4), 1443–1448.

Zhou, Z.-H. & Feng, J. (2017). Deep forest.