Proteasomal cleavage prediction: state-of-the-art and future directions

Ingo Ziegler[1], Bolei Ma[1], Bernd Bischl[2,3], Emilio Dorigatti[2,3,4,*,†], and Benjamin Schubert[4,5,†]

[1]Center for Information and Language Processing, Ludwig-Maximilians-Universität München
[2]Department of Statistics, Ludwig-Maximilians-Universität München
[3]Munich Center for Machine Learning (MCML)
[4]Computational Health Center, Helmholtz Zentrum München – German Research Center for Environmental Health
[5]Department of Mathematics, Technical University of Munich
[*]Corresponding author: emilio.dorigatti@stat.uni-muenchen.de
[†]Share last authorship.

July 17, 2023

Abstract

Epitope vaccines are a promising approach for precision treatment of pathogens, cancer, autoimmune diseases, and allergies. Effectively designing such vaccines requires accurate proteasomal cleavage prediction to ensure that the epitopes included in the vaccine trigger an immune response. The performance of proteasomal cleavage predictors has been steadily improving over the past decades owing to increasing data availability and methodological advances. In this review, we summarize the current proteasomal cleavage prediction landscape and, in light of recent progress in the field of deep learning, develop and compare a wide range of recent architectures and techniques, including long short-term memory (LSTM), transformers, and convolutional neural networks (CNN), as well as four different denoising techniques. All open-source cleavage predictors re-trained on our dataset performed within two AUC percentage points. Our comprehensive deep learning architecture benchmark improved performance by 0.4 AUC percentage points, while closed-source predictors performed considerably worse. We found that a wide range of architectures and training regimes all result in very similar performance, suggesting that the specific modeling approach employed has a limited impact on predictive performance compared to the specifics of the dataset employed. We speculate that the noise and implicit nature of data acquisition techniques used for training proteasomal cleavage prediction models and the complexity of biological processes of the antigen processing pathway are the major limiting factors. While biological complexity can be tackled by more data and, to a lesser extent, better models, noise and randomness inherently limit the maximum achievable predictive performance.

Keywords: Epitope Vaccine, Proteasomal Cleavage Prediction, Deep Learning, Benchmarking

Key points:

- We benchmark a wide range of neural architectures and learning methods for proteasomal cleavage prediction.
- Predictors re-trained on our dataset performed similarly and better than closed source, pre-

trained predictors.

- We show clear performance plateaus with little variation among methods.
- We argue that collecting more data or using different learning methods is unlikely to yield significant performance gains.

1 Introduction

Epitope vaccines (EV) are a promising avenue for precision medical treatment of conditions such as autoimmune diseases and cancer, as they offer the possibility of including highly patient- and disease-specific antigens [1]. EVs are long polypeptides that contain the very specific parts of an antigen that are recognized by the immune system and trigger an immune response, the so-called epitopes. Proteasomal digestion plays a major role in EV efficacy [2] (Fig. 1). By cleaving the EV incorrectly, therapeutic epitopes might get destroyed and will not be presented at the surface by the Major Histocompatibility Complex (MHC) to T cells.

Several strategies have been tested to deliver EVs in the most effective way [3], and it has been experimentally shown that epitope ordering in an EV affects its efficacy by determining the specificity of the immune response to antigen stimulation after vaccination [2]. Epitope ordering affects cleavage efficiency at their terminals and thus the probability that they are correctly separated from the EV construct. Modern EV design frameworks [4, 5], therefore, leverage proteasomal cleavage predictors to improve epitope recovery rates, thereby improving their effectiveness at lower dosages and thus reducing the cost to patients.

In this review, we give an overview of all known proteasomal cleavage site predictors and benchmark all currently accessible ones. In light of the recent developments in the field of deep learning, we additionally benchmark a wide variety of neural architectures, denoising methods, amino acid embedding methods, and training regimes, that have never been applied to proteasomal cleavage prediction before and identify parameters and architecture choices that impact performance. Based on the benchmark results and ablation study, we finally discuss the future direction of the field and give recommendations for the development of novel proteasomal cleavage predictors.

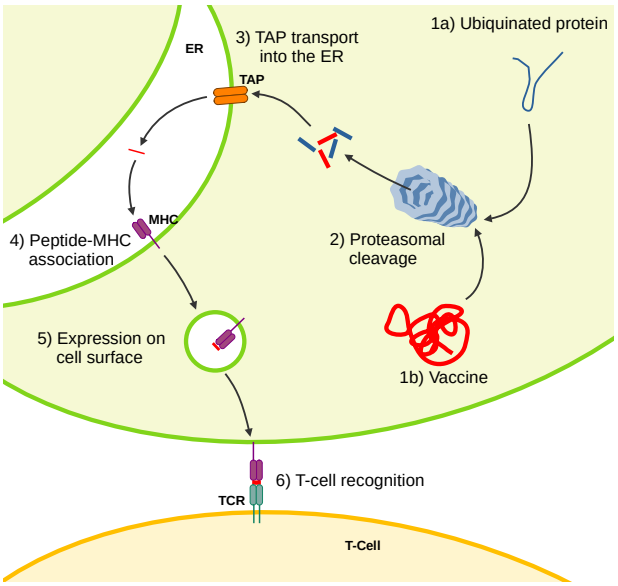

Figure 1: The events in the antigen processing pathway determine what is presented to T cells, thus accurate predictions are essential to enable the design of epitope vaccines. The original purpose of proteasomal cleavage (2) is to digest ubiquinated (old or misfolded) proteins into shorter peptides, but epitope vaccines (1b) undergo the same process. Some of these peptides enter the endoplasmic reticulum (ER) through the Transporter associated with Antigen Processing (TAP, 3), and a fraction of these bind to the Major Histocompatibility Complex (MHC, 4). The resulting complex is then presented on the cell surface (5) enabling immunosurveillance by T cells (6).

2 Evolution of proteasomal cleavage predictors

Proteasomal cleavage prediction efforts (Table 1) were initiated by the Prediction Algorithm for Proteasomal Cleavages (PAProC), predicting proteasomal cleavage sites of human and yeast proteasomes [6], which later has been made publicly available as a web-server [7]. The model encodes hand-picked features such as dis-

tance information from one cut to the next as well as probabilities about neighboring amino acids and the resulting likelihood leading to another cut. While PAProC was only trained on *in vitro* data, [8] also used natural MHC-I ligands and extended their initial multi-layer-perceptron to an ensemble of neural networks to further improve performance [9]. Cleavage prediction was integrated with the two other main processes of the antigen processing pathway, namely TAP transport and MHC binding, almost concurrently by [10, ProteaSMM] and [11, PCM]. Both methods are based on position-specific scoring matrices (PSSM), where a separate coefficient is associated with each amino acid position combination and the final cleavage probability is derived linearly from such scores. [12] proposed Pcleavage, a cleavage predictor based on Support Vector Machines [13, SVM], which has been made available as a web-server. Along Pcleavage, the authors benchmarked alternative classifier systems based on nearest-neighbors, logistic regression, naive Bayes, and decision trees, but found the SVM predictor to perform best, with little performance difference between polynomial and radial basis function (RBF) kernels. [14] approached cleavage prediction from a uniquely different perspective and tried to predict the probability that a given peptide was generated by proteasomal cleavage event, rather than the cleavage probability at each position of an antigen, with a similar approach to that of PCM. While most previous and following cleavage predictors focused on the C-terminal, [15] used the PSSM technique introduced previously to study N-terminal cleavage patterns and used the insights derived from their models to propose a more comprehensive modeling approach of the antigen processing pathway comprising epitope trimming besides proteasomal cleavage, TAP transport, and MHC binding. With PCPS, [16] used a model based on n-gram frequencies due to their success in computational linguistics and showed that their model in conjunction with MHC-I binding prediction greatly increase the discovery rate of MHC-restricted CD8 T cell epitopes. [17] departed from the one-hot encoding of amino acids used in the previous methods and instead used amino acid features summarizing their physio-chemical properties [18, VHSE] via principal-component analysis [19, PCA]. By interpreting the importance of the features used by their proposed SVM-based predictor, the authors identified proteasome specificities in terms of hydrophobicity of the amino acids in the flanking regions. NetCleave [20] extended cleavage prediction to MHC-II and H2 ligands and returned to neural network models, while also using VHSE amino acid descriptors. Pepsickle [21] collected a larger dataset of *in vitro* cleavage sites and showed that using predicted C-terminal cleavage probabilities helped isolating immune-responsive epitopes, thus improving EV design. Finally, [22] proposed to treat decoy samples as unlabeled, rather than negatives, and used specific techniques that do not require labeled negatives to learn a classifier [23].

2.1 Data types for training proteasomal cleavage prediction models

Proteasomal cleavage sites are usually inferred implicitly from MHC ligandomics data [25] collected via mass spectrometry (MS) by matching eluted ligands to their progenitor protein to recover sequence information surrounding the terminals [8]. This procedure, however, does not give an indication of which amino acid sequences *cannot* result in a cleavage event, since missed cleavage sites are not observed in MHC ligands, and presents a biased picture of cleavage patterns as not all proteolytically generated peptides will be presented by the MHC. Alternatively, *in vitro* collected proteasomal digested proteins are used to train cleavage predictors. These data are more comprehensive and less biased, however, low-throughput and are harder to obtain compared to MHC ligandomics. Additionally, such data do not contain any information about alternative antigen-processing pathways that lead to peptide presentation and successive antigen-processing steps. This complicates the use of predictors trained on *in vitro* data alone in pipelines aimed at identifying immunogenic peptides as it creates the necessity of modeling successive pathway events separately, as done for example by NetCTLpan [26]. Therefore, most modern proteasomal cleavage predictors are trained solely on MHC ligandomics data or integrate both *in vitro* and MS.

To bypass MHC liganomdics' inability to identify true non-cleaved sites, decoy negative samples are

Model	Modeling Technique	Epitope Terminals	Window size B	Window size A	Training data *in vitro*	Training data MS	Availability Model	Availability Code	Availability Web
PAProC [6]	MLP	N/C	6	4	✓				
NetChop [8]	MLP	C	9	9	✓	✓			
ProteaSMM [24]	Linear	C	6	4	✓		✓	✓	
PCM [11]	Linear	C	4	2	✓				
NetChop 3.1 [9]	MLP Ens.	C	8	8	✓	✓	✓		✓
Pcleavage [12]	SVM	C	9	9	✓	✓		✓	
[15]	Linear	N/C	5	0		✓			
PCPS [16]	N-grams	C	6	6		✓			✓
[17]	SVM	C	14	14	✓	✓			
NetCleave [20]	MLP	C	4	3		✓	✓	✓	
Pepsickle [21]	MLP	C	8	8	✓	✓	✓	✓	
PUUPL [22]	MLP Ens.	N/C	6	4		✓	✓	✓	
This paper	BiLSTM	N/C	6	4		✓	✓	✓	

Table 1: Chronological overview of existing proteasomal cleavage predictors, the modeling technique they adopted, which epitope terminals were modeled, the window size before (B) and after (A) the cleavage site, whether they used *in vitro* or mass-spectrometry (MS) training data, and the public availability of the predictor. Links to model, code and/or web server functionality, when available, are found in Appendix A.1.

usually generated synthetically either by randomly shuffling the amino acids in a short window around the cleavage site or by considering artificial negative sites located around observed cleavage events [27] (Fig. 2). Even though such negative samples are not entirely reliable, the growing availability of this kind of data [28] spurred continuous development and improvement of proteasomal cleavage predictors [8, 6, 29, 30], which have been recently revised in light of new innovations in the deep learning field [20, 22, 21].

3 Benchmarking of proteasomal cleavage prediction strategies

All existing methods differ by subtle details such as the size and composition of the training data, the dimension of the window surrounding the cleavage site, and the way amino acids are encoded and presented to the model. Therefore, we performed a comprehensive benchmark of publicly available approaches on comparable datasets to compare and identify beneficial modeling approaches. Additionally, we extended this benchmark to neural architectures and modeling techniques that have not yet been used in the context of proteasomal cleavage site prediction. We considered three main axis of variation: the initial embedding of amino acids, the neural architecture of the predictor, and their training regime via noise handling and data augmentations.

3.1 Embedding

The concept of embeddings originally stems from natural language processing (NLP) and language models, but targeted adaptations for amino acid sequence data have already been presented [31]. By associating numerical vectors to *tokens* such as characters or words, embeddings constitute the first encoding step of a model, and the method employed is crucial as it influences what intrinsic information a model can exploit for classification [31]. We, thus, considered various embeddings in our analysis, while keeping the base architecture equal. Specifically, we analyzed the performance of a randomly initialized embedding layer that was optimized in conjunction with the loss function of the whole model, and dedicated and pre-trained large protein language models such as Prot2Vec [32]. Furthermore, we also employed sequence embeddings obtained by concatenating independently-trained forward and backward amino acid representations of each input [33], as well as embeddings of the N most common variable-length substrings identified via byte-level byte pair encoding [34, BBPE] and WordPiece [35, WP]. For BBPE, we try both $N = 1000$ and $N = 50\,000$, while for WP we use $N = 50\,000$.

3.2 Neural architectures

Different neural architectures embody different inductive biases that may or may not help a learning task. Most proteasomal cleavage methods made use of multi-layer perceptrons (Table 2), however, other types of architectures also provided high performance in sequence-based classification tasks. We, therefore, evaluated all modern types of neural architectures, from recurrent to transformer-based, convolutional neural networks, and perceptrons.

Recurrent: Bidirectional long short-term memory (BiLSTM) networks [36] are well suited for a wide range of text classification tasks, thus, as proteasomal cleavage prediction is performed on amino acid sequences in a sequence-to-one classification setting similar to text classification, we based nine of 12 model architectures around BiLSTMs. The fundamental structure for our BiLSTMs was built around the architecture proposed by [37], in which multiple sequential BiLSTMs were followed by a hidden and an output layer. For eight of our nine BiLSTM-related experiments, we chose two sequential BiLSTMs in which sequence dimensionality was reduced by taking the maximum value of the depth-wise per-residue output of the last layer. For the hidden layer, we used the Gaussian Error Linear Units (GELU) [38] activation function. As the attention mechanism has also been applied to non-transformer architectures, we additionally employed a modified version of the BiLSTM with extended residual connections and an attention mechanism in order to improve its performance on protein sequence data [39].

Transformers: Besides RNNs, the attention mechanism introduced by [40] enabled a whole new architec-

ture capable of processing sequences: the Transformer. These complex multi-million and up to multi-billion parameter models feature an encoder and decoder system and are frequently pre-trained on large datasets in a self-supervised manner, for example by predicting randomly masked-out tokens from an input sequence. The encoders and/or decoders of such pre-trained models are then used as the basis of new models that are fine-tuned for a variety of specific downstream tasks. We used ProtTrans' T5-XL encoder-only model [41] featuring 1.2 billion parameters and pre-trained on the UniRef dataset, as well as the ESM2 Transformer [42] in its 150 million parameter version. Additionally, we included a fine-tuning performance of ESM2 by adding a linear layer projection from its vocabulary-sized per-residue RoBERTa Language Model Head [43, 44] to our binary classification target.

Convolutional and Perceptron: While convolutional neural networks [45, CNN] are predominantly used in computer vision problems, the convolution filters can also be seen as a sliding k-mer window over the input sequence [46]. While CNNs have already been used in conjunction with attention for protease-specific substrates FMSand cleavage sites [47], their use for proteasomal cleavage prediction remains unexplored (Table 2). Furthermore, as most of the proposed proteasomal cleavage predictors make use of multi-layer perceptrons (Table 2), we also included a single hidden layer perceptron [48] with Rectified Linear Units [49] as activation function into the analysis.

3.3 Training

The training dataset and procedure also impacts the final performance. As described in the introduction, decoy negative samples may contain some positives. To account for negative label uncertainty, we tested multiple procedures that account for this fact. Data augmentation also proved successful to reduce overfitting and improve generalizability.

Dataset: We used the dataset introduced in [22], which contains 229 163 and 222 181 N- and C-terminals cleavage sites, respectively. Each cleavage site is captured in a window comprising six amino acids to its left and four to its right and is associated with six decoy negative samples obtained by considering the three residues preceding and following it (Fig. 2), resulting in a total of 1 434 989 and 1 419 501 samples after deduplication for N- and C-terminals. As the decoy negatives are situated in close proximity to real cleavage sites and due to the probabilistic nature of proteasomal cleavage, some of the negative samples are likely to be actual, unmeasured cleavage sites, and may influence the performance of predictors trained using such data. However, we prefer this method of generating decoys compared to the popular alternative of randomly shuffling amino acids in the window as the latter results in sequences that are potentially not observed in nature, thus making the prediction problem artificially easier and overestimating the models' performance on natural sequences. For the ablation analyses, we used the above-described dataset to investigate the impact of varying the window size around the cleavage site as well as the number of training samples used to train the predictors.

Noisy labels: To reduce the impact of asymmetric label noise on the performance of our classifiers, we took five recent deep learning-specific denoising approaches into consideration: a noise adaptation layer, which attempts to learn the noise distribution in the data [50], co-teaching, where two models are trained simultaneously by deciding for the respective other network which samples from a mini-batch to use for training [51], and co-teaching-plus [52], which extends co-teaching with the disagreement learning approach of decoupling [53]. We additionally considered a joint training method with co-regularization (JoCoR) [54] that, unlike co-teaching and its variations, aims at reducing the disagreement of two networks by calculating a joint loss under co-regularization and choosing small-loss samples to update the parameters of both networks simultaneously. We also tried DivideMix [55], a holistic approach originally developed for computer vision, which integrates multiple frameworks such as co-teaching and MixMatch [56] into one. Following the co-teaching approach, each network feeds the other network parts of the training dataset. In this case, however, the division of samples happens at the beginning of an epoch by fitting a Gaussian Mixture Model [57, GMM] and splitting the dataset at a chosen threshold. High-loss samples, as

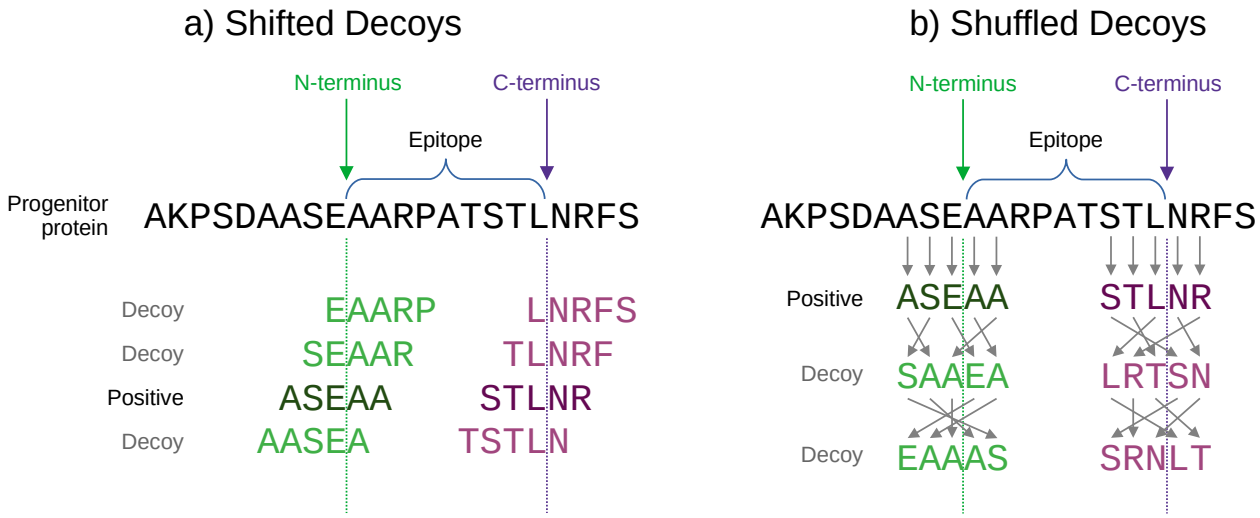

Figure 2: Two alternative ways of creating decoy negative samples from MHC ligandomics data. MHC-I epitopes are matched to their progenitor protein and short windows around its terminals, in the figure composed of three residues before and two after, used to construct decoys. Shifted decoys (a) are substrings of the original sequence starting earlier or after the observed cleavage site, whereas shuffled decoys (b) contain the same amino acids as the positive example but shuffled in random order. In this work, we consider shifted decoys as they are more realistic (see main text).

classified by the GMM, are treated as unlabeled and semi-supervised learning techniques based on MixMatch are followed. MixMatch unifies consistency regularization, where different augmented views of the same example are constrained to lead to the same prediction, with entropy regularization, which forces the classifier's predictions to have low entropy. As MixMatch was developed for image data, we adjust it for sequence data by mixing up the embedded sequence representation [58] instead of the pixel input in the data loading process.

Data augmentation: For all models except external baselines, we applied data augmentation directly on the input sequences to combat overfitting and improve generalizability by randomly masking $\max(1, \lfloor \ell/10 \rfloor)$ amino acids for a sequence of length ℓ as unknown [59]. All predictors except ESM2 fine-tuning used adaptive momentum [60] as their optimization technique, whereas ESM2 fine-tuning used adaptive momentum with decoupled weight decay [61]. All models without denoising techniques used binary cross-entropy loss [62], while all denoising models calculated dedicated losses.

3.4 Baselines

As baseline models, we considered a logistic regression classifier with L2 regularization, which is essentially equivalent to ProteaSMM, tuned via 10-folds cross-validation, and a SVM model fitted using 15 000 samples to computational limitations, optimizing its hyperparameters via three-folds cross-validation. As for published predictors, we benchmarked against NetCleave [20], Pepsickle [21], and PUUPL [22], retraining them on our dataset of both N- and C-terminals, as well as comparing a retrained version of PCM [29] with the published pretrained model. We also considered PCPS (Model 1) [16], NetChop 20S and NetChop C-term [9], however, we could not re-train these due to the unavailability of the source code. We could not use PAProC as the web-server [7] is currently not functional, while [17] never published code nor pretrained models. Both authors did not respond to our inquiries.

3.5 Experimental protocol

The experimental protocol determines how we optimized and evaluated the models.

Hyperparameter optimization: To gather initial broad ranges of well-performing hyperparameters, we performed manual try-out runs in a grid search approach. Due to computational limitations, we divided the hyperparameter search into three priority groups: group one used Ray Tune's [63] implementation of the asynchronous Hyperband algorithm [64] and evaluated each configuration in a ten-folds cross-validation (CV), while for groups two and three we chose hyperparameters according to the initial grid search results and evaluated each configuration with five-folds CV (group two) or a single run on a held-out validation set (group three) (as shown in Table 16 of Appendix A.3). We then used the best hyperparameter combination to train each architecture with all denoising methods, except for DivideMix where we only trained the overall best-performing architecture due to computational limitations. Information on the different architectures and the exact Hyperband ranges and chosen hyperparameters for all models are documented in Appendix A.2.

Evaluation: As previously mentioned, some decoy examplesmay actually result in a proteasomal cleavage event *in vivo* due to the way these negative samples are generated. For this reason, traditional binary classification metrics such as accuracy, precision, recall, etc., are misleading and model evaluation should instead be based on the AUC [65]. We reserved a random 10% of each terminal dataset as test dataset used for the final evaluation of the best hyperparameters.

4 Results

The overall best-performing C-terminal model architecture as measured by AUC was the BiLSTM at 88.55% without any denoising methods, while for the N-term, the BiLSTM+T5 with noise adaptation layer version narrowly outperformed the base BiLSTM version by 0.04 percentage points (p.p.) at a level of 79.54% AUC (Fig. 3 and Appendix A.3 and A.4). If denoising techniques were applied, the noise adaptation layer consistently performed best for both the C- and N-terminal. However, in 11 (10) of 12 models for the C-terminal (N-terminal), no denoising method resulted in superior results. Co-teaching-plus dominated co-teaching along all (11) model architectures in the C-terminal (N-terminal). JoCoR appeared to significantly hinder model performance in all architectures, whereas DivideMix also reduced the BiLSTM AUC score by around 2.4 p.p. in both terminals. While the best-performing BiLSTM consisted of 4.6 million parameters, the MLP with 30 529 parameters only lacked 0.38 p.p. AUC behind and additionally beat the pre-trained Prot2Vec as well as DeepCleave architectures, both featuring around 16 million parameters in the C-terminal version. All transformer architectures in the ranges of 148 million (ESM2-based) and 1.2 billion (T5) parameters ranked behind the BiLSTM architecture but narrowly outperformed the MLP with AUC scores of around 88.32%.

For the C-terminal, models including trainable tokenizers dropped to their worst-performing state compared to their fixed-vocabulary counterpart, especially when increasing the number of amino acid substring combinations that had to be learned. Whereas the BiLSTM with BBPE vocabulary size 1000 dropped 3.3 p.p. to 85.25% AUC, the same model architecture with 50 000 learned sub-string combinations was only able to reach 68.92% AUC. A similar but less severe pattern could be observed with WordPiece encodings, where the version with vocabulary of size 50 000 reached 73.46% AUC. If these models additionally featured denoising methods, the performance loss intensified up to a level of almost random-guessing (52.09% AUC for 50 000 BBPE and JoCoR).

Interestingly, the N-terminal cleavage models showed a different behavior for certain architecture combinations. BiLSTM+Attention and BiLSTM+Prot2Vec had significantly larger performance drop-offs from their best-performing model for JoCoR-denoising (10.7 and 18 p.p., respectively) compared to the C-terminal (6.8 and 12.7 p.p., respectively). On the other hand, the performance loss due to trainable tokenizers paired with JoCoR was less severe for N-terminal cleavage sites (21.5, 26.5, 23.2 p.p., respectively) for BBPE-1000, BBPE-50 000, and WP-50 000 than for C-terminal cleavage sites (29.7, 36.4, 35.1

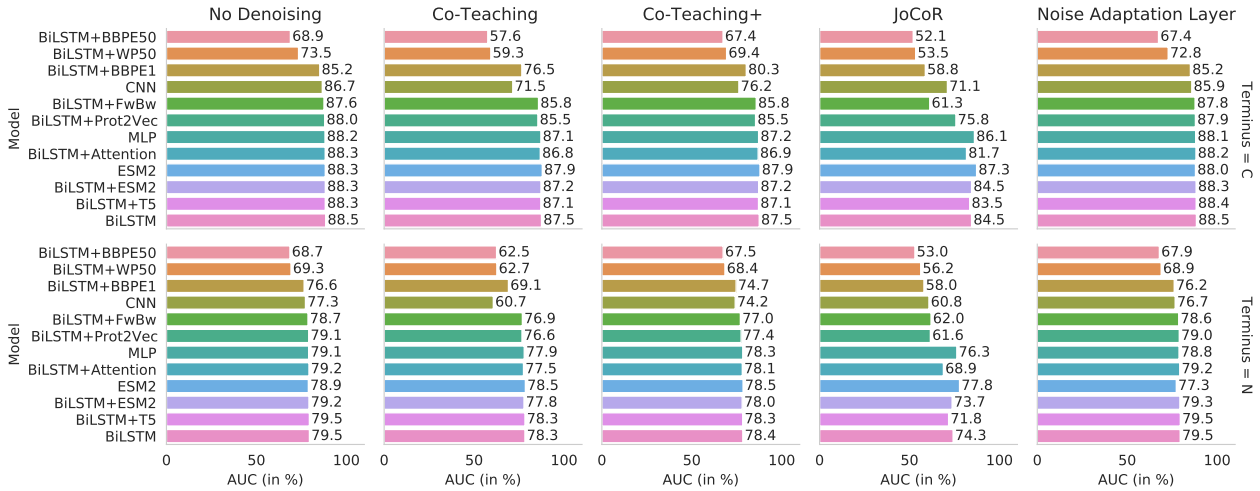

Figure 3: Model performances on C- and N-terminal (rows) for different denoising methods (columns).

p.p., respectively).

Replacing the embedding layer with a forward-backward representation yielded comparable performance to the base BiLSTM architecture. Nonetheless, the base BiLSTM architecture was preferable as the additional forward-backward encoding steps increased training time by a factor of six.

4.1 Ablation results

To study the influence of the dataset on the predictive performance, we trained the best-performing model identified in the comprehensive benchmark, the BiLSTM, with fewer and fewer training examples as well as with different window sizes around cleavage sites. Due to the small variation in performance discussed earlier, it is likely that repeating these experiments with different methods would result in qualitatively identical conclusions.

Training set size: The size of the training dataset had a neglectable impact on performance, with a decrease of only 0.4 and 0.3 p.p. in AUC with half as many training samples for N- and C-terminals respectively (Fig. 4 left, and Table 29 in Appendix A.5). The performance decrease when training with only 10% of the full dataset (about 22 000 cleavage sites and 140 000 decoys) was 1.8 p.p. for N-terminals and 1.2 for C-terminals. We monitored training and validation losses throughout to ensure that no overfitting occurred with the smaller training sets.

Window size: As for window size, performance seemed to plateau when 14 residues were used at each side of the cleavage site, with an improvement of less than 0.2 p.p. when doubling the window size (Fig. 4 right, and Table 30 in Appendix A.5). Starting at a half-window size of 21, the BiLSTM models failed to learn anything, achieving no more than 54% training AUC. Introducing batch normalization layers [66] resolved the issue, and in this way the overall best AUC achieved with a window of 56 residues was 92.3% for C-terminals and 89.3% for N-terminals. Increasing the window size from 6 to 56 improved predictive performance by 12.5 p.p. for N-terminals and only by 5.9 p.p. for C-terminals.

Model size: To ensure that the performance numbers just mentioned were not limited by the capacity of the model, we tried to increase the number of units and layers by 50% and 100%, however, both models started to overfit before reaching the performance level of our base BiLSTM model. As we already applied 50% dropout [67], we progressively increased weight decay [68] and repeated training. In no scenario did any combination of weight decay and increased model size achieve the performance of our base BiL-

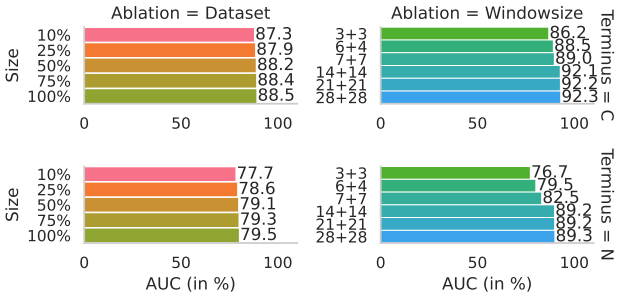

Figure 4: BiLSTM performance with smaller training datasets (left) and different window sizes (right) on C- and N-terminals (top and bottom rows).

STM model. For higher levels of weight decay, e.g., 1×10^{-2}, training was severely limited and learning stopped at 87% AUC. Lower levels of weight decay, e.g., 1×10^{-4} were more competitive but still could not match the base BiLSTM performance, even in a setting with extended training epochs. The learning and generalization curves for the 50% size increase model were mirrored by those with 100% increased size. Exact results for each weight decay setting can be found in Table 31 in Appendix A.5.

4.2 Baseline results

Models re-trained on our dataset (PCM, Logistic regression, SVM, NetCleave, PUUPL, and Pepsickle) outperformed available pre-trained models (both versions of NetChop, PCPS and PCM). In all cases, our BiLSTM models outperform all to-date published methods and thus constitute a new state-of-the-art. Interestingly, both PCM and the logistic regression classifier performed better than NetChop but worse than NetCleave. Both NetChop and NetCleave are based on MLPs, but while NetCleave was re-trained, NetChop was not, suggesting that retraining NetChop on our dataset would obtain higher performance than shown in the table. Importantly, such simple linear models achieved only two or three p.p. fewer than the considerably more complicated BiLSTM. The fact that both linear models performed very competitively compared to the other deep architectures suggests that improvements in the training data were the ma-

Method	Dataset	
	C-terminal	N-terminal
PCPS	51.3	50.0
PCM	64.5	52.4
NetChop 3.1 (20S)	66.1	52.7
NetChop 3.1 (C-term)	81.5	51.0
SVM†	84.8	73.2
PCM†	85.3	75.5
Logistic Regression†	86.2	76.2
NetCleave†	86.9	76.4
PUUPL†	87.2	78.0
Pepsickle†	88.1	78.9
BiLSTM (6+4)	**88.5**	**79.5**
BiLSTM (28+28)	**92.3**	**89.3**

Table 2: AUC (in %) of several cleavage prediction methods on our dataset. We present our BiLSTM results with a window size (in parentheses) comparable to the other methods as well as with the largest window size we could use for training. † SVM, logistic regression, NetCleave, Pepsickle, and PUUPL were re-trained from scratch on our dataset, while we used the published NetChop pre-trained models and the PCPS web-server functionality to obtain their performance on our test set. We present the performance of the original PCM as well as a version re-trained on our dataset.

jor driver of the improved performance observed over the years. The SVM model performed slightly worse than logistic regression even though the hyperparameter optimization preferred a RBF kernel over the linear and polynomial kernels. The number of training samples was the bottleneck, and performance could be increased at great computational cost.

5 Conclusion

We benchmarked various deep learning architectures, including MLPs, CNNs, LSTMs, and Transformers, for the task of proteasomal cleavage prediction, and showed that several embedding techniques in combination with model architectures of vastly different scale and complexity all reach similar performance levels around 88.5% AUC for C-terminals and 79.5% AUC for N-terminals, while a simple linear model only achieves two or three p.p. fewer compared to such complicated models. Denoising techniques as well as trainable tokenizers appeared to offer limited to no, or even negative benefits, possibly because the MHC ligandomics dataset already covers most of the cleavage sites that can result in an MHC ligand. By increasing the size of the window around the cleavage site, we reached up to 92.3% and 89.3% AUC for C- and N-terminals, with evident plateaus in terms of training dataset and window size. Performance on the N-terminals dataset was more sensitive both to window size and the number of training samples, confirming the higher difficulty of predicting this process compared to C-terminus cleavage. When maximum predictive performance is desired, we suggest using our best-performing BiLSTM model with the largest possible window size that is feasible for the application at hand, while single-layer or multi-layer perceptrons can be used when increased efficiency is desired with only a slight penalty in predictive performance. The logistic regression classifier also presents a viable option when efficiency is the most important concern as its predictive performance is very competitive with all other deep neural network models.

Such saturated results suggest that different modeling choices of architectures, embeddings, or training regimes are unlikely to yield significantly better predictive performance in the future, although marginal gains could still be possible with even larger datasets. Based on this insight we speculate that further efforts to improve proteasomal cleavage prediction could focus on a more comprehensive modeling of the antigen pathway. For example, it has been shown that cleavage prediction benefits from additionally predicting the MHC binding affinity of the resulting peptides [8, 16]. Another possibility is that biological processes such as proteasomal cleavage are simply too noisy and random to allow more accurate predictions, in which case we may already be close to the boundary of what is possible to achieve *in silico*.

Declarations

Competing interests: No competing interest is declared.

Author contributions statement:

E.D. conceived and directed the benchmark, developed the experimental design, performed the literature review, implemented and executed the SVM, logistic regression, and PCM baselines. I.Z. and B.M. implemented and executed all experiments involving deep architectures. E.D., I.Z., and B.M. and B.S. wrote the manuscript. B.S. advised in all phases of the project. B.B. advised on the experimental design and interpretation of the results. All authors reviewed and approved the manuscript.

Acknowledgments: The authors thank the anonymous reviewers for their valuable suggestions. E. D. was supported by the Helmholtz Association under the joint research school "Munich School for Data Science - MUDS" (Award Number HIDSS-0006).

Code and data availability: All our datasets and experiments are available at `https://github.com/ziegler-ingo/cleavage_extended`.

References

[1] Fabio Boniolo, Emilio Dorigatti, Alexander J. Ohnmacht, Dieter Saur, Benjamin Schubert, and Michael P. Menden. Artificial intelligence in early drug discovery enabling precision medicine. *Ex-*

pert Opinion on Drug Discovery, 16(9):991–1007, June 2021.

[2] Sébastien Cornet, Isabelle Miconnet, Jeanne Menez, François Lemonnier, and Kostas Kosmatopoulos. Optimal organization of a polypeptide-based candidate cancer vaccine composed of cryptic tumor peptides with enhanced immunogenicity. *Vaccine*, 24(12):2102–2109, 2006.

[3] Alessandro Sette and John Fikes. Epitope-based vaccines: an update on epitope identification, vaccine design and delivery. *Current Opinion in Immunology*, 15(4):461–470, aug 2003.

[4] Emilio Dorigatti and Benjamin Schubert. Graph-theoretical formulation of the generalized epitope-based vaccine design problem. *PLOS Computational Biology*, 16(10):e1008237, October 2020.

[5] Emilio Dorigatti and Benjamin Schubert. Joint epitope selection and spacer design for string-of-beads vaccines. *Bioinformatics*, 36(Supplement_2):i643–i650, December 2020.

[6] Christina Kuttler, Alexander K Nussbaum, Tobias P Dick, Hans-Georg Rammensee, Hansjörg Schild, and Karl-Peter Hadeler. An algorithm for the prediction of proteasomal cleavages. *Journal of Molecular Biology*, 298(3):417–429, May 2000.

[7] A. Nussbaum, C. Kuttler, KP. Hadeler, HG. Rammensee, and H. Schild. Paproc: a prediction algorithm for proteasomal cleavages available on the www. *Immunogenetics*, 53:87–94, 2001.

[8] Can Keşmir, Alexander K. Nussbaum, Hansjörg Schild, Vincent Detours, and Søren Brunak. Prediction of proteasome cleavage motifs by neural networks. *Protein Engineering, Design and Selection*, 15(4):287–296, April 2002.

[9] Morten Nielsen, Claus Lundgegaard, Ole Lund, and Can Keşmir. The role of the proteasome in generating cytotoxic t-cell epitopes: insights obtained from improved predictions of proteasomal cleavage. *Immunogenetics*, 57:33–41, 1-2 2005.

[10] S. Tenzer, B. Peters, S. Bulik, O. Schoor, C. Lemmel, M. M. Schatz, P.-M. Kloetzel, H.-G. Rammensee, H. Schild, and H.-G. Holzh"utter. Modeling the MHC class i pathway by combining predictions of proteasomal cleavage,TAP transport and MHC class i binding. *CMLS, Cell. Mol. Life Sci. Cellular and Molecular Life Sciences*, 62(9):1025–1037, may 2005.

[11] Pierre Dönnes and Oliver Kohlbacher. Integrated modeling of the major events in the MHC class I antigen processing pathway. *Protein Science*, 14(8):2132–2140, August 2005.

[12] M. Bhasin and G. P. S. Raghava. Pcleavage: an SVM based method for prediction of constitutive proteasome and immunoproteasome cleavage sites in antigenic sequences. *Nucleic Acids Research*, 33(Web Server):W202–W207, jul 2005.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] Ido Ginodi, Tal Vider-Shalit, Lea Tsaban, and Yoram Louzoun. Precise score for the prediction of peptides cleaved by the proteasome. *Bioinformatics*, 24(4):477–483, jan 2008.

[15] Mark M. Schatz, Björn Peters, Nadja Akkad, Nina Ullrich, Alejandra Nacarino Martinez, Oliver Carroll, Sascha Bulik, Hans-Georg Rammensee, Peter van Endert, Hermann-Georg Holzhütter, Stefan Tenzer, and Hansjörg Schild. Characterizing the n-terminal processing motif of MHC class i ligands. *The Journal of Immunology*, 180(5):3210–3217, mar 2008.

[16] Carmen M Diez-Rivero, Esther M Lafuente, and Pedro A Reche. Computational analysis and modeling of cleavage by the immunoproteasome and the constitutive proteasome. *BMC Bioinformatics*, 11(1), sep 2010.

[17] Jiangan Xie, Zhiling Xu, Shangbo Zhou, Xianchao Pan, Shaoxi Cai, Li Yang, and Hu Mei. The vhse-based prediction of proteasomal cleavage sites. *PLOS ONE*, 8(9):1–8, 09 2013.

[18] Hu Mei, Zhihua Liao, Yuan Zhou, and Shengshi Z. Li. A new set of amino acid descriptors and its application in peptide qsars. *Biopolymers*, 80 6:775–86, 2005.

[19] Ian T. Jolliffe. Principal component analysis. In *International Encyclopedia of Statistical Science*, 2002.

[20] Pep Amengual-Rigo and Victor Guallar. NetCleave: an open-source algorithm for predicting c-terminal antigen processing for MHC-i and MHC-II. *Scientific Reports*, 11(1), June 2021.

[21] Benjamin R Weeder, Mary A Wood, Ellysia Li, Abhinav Nellore, and Reid F Thompson. pepsickle rapidly and accurately predicts proteasomal cleavage sites for improved neoantigen identification. *Bioinformatics*, 37(21):3723–3733, September 2021.

[22] Emilio Dorigatti, Bernd Bischl, and Benjamin Schubert. Improved proteasomal cleavage prediction with positive-unlabeled learning. *arXiv preprint arXiv:2209.07527*, 2022.

[23] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.

[24] S. Tenzer, B. Peters, S. Bulik, O. Schoor, C. Lemmel, M. M. Schatz, P.-M. Kloetzel, H.-G. Rammensee, H. Schild, and H.-G. Holzh"utter. Modeling the MHC class I pathway by combining predictions of proteasomal cleavage,TAP transport and MHC class I binding. *CMLS Cellular and Molecular Life Sciences*, 62(9):1025–1037, May 2005.

[25] Anthony W. Purcell, Sri H. Ramarathinam, and Nicola Ternette. Mass spectrometry–based identification of MHC-bound peptides for immunopeptidomics. *Nat Protoc*, 14(6):1687–1707, may 2019.

[26] Thomas Stranzl, Mette Voldby Larsen, Claus Lundegaard, and Morten Nielsen. NetCTLpan: pan-specific MHC class i pathway epitope predictions. *Immunogenetics*, 62(6):357–368, apr 2010.

[27] Jorg J. A. Calis, Peter Reinink, Christin Keller, Peter M. Kloetzel, and Can Keşmir. Role of peptide processing predictions in t cell epitope identification: contribution of different prediction programs. *Immunogenetics*, 67(2):85–93, December 2014.

[28] Randi Vita, Swapnil Mahajan, James A Overton, Sandeep Kumar Dhanda, Sheridan Martini, Jason R Cantrell, Daniel K Wheeler, Alessandro Sette, and Bjoern Peters. The Immune Epitope Database (IEDB): 2018 update. *Nucleic Acids Research*, 47(D1):D339–D343, 10 2018.

[29] Pierre Dönnes and Oliver Kohlbacher. Integrated modeling of the major events in the MHC class i antigen processing pathway. *Protein Science*, 14(8):2132–2140, August 2005.

[30] Morten Nielsen, Claus Lundegaard, Ole Lund, and Can Keşmir. The role of the proteasome in generating cytotoxic t-cell epitopes: insights obtained from improved predictions of proteasomal cleavage. *Immunogenetics*, 57(1-2):33–41, mar 2005.

[31] Nabil Ibtehaz and Daisuke Kihara. Application of sequence embedding in protein sequence-based predictions. *arXiv preprint arXiv:2110.07609*, 2021.

[32] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[33] Georg Heigold, Guenter Neumann, and Josef van Genabith. Neural morphological tagging from characters for morphologically rich languages. *CoRR*, abs/1606.06640, 2016.

[34] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics.

[35] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

[36] Alex Graves and J"urgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, July 2005.

[37] Matiss Ozols, Alexander Eckersley, Christopher I. Platt, Callum Stewart-McGuinness, Sarah A. Hibbert, Jerico Revote, Fuyi Li, Christopher E. M. Griffiths, Rachel E. B. Watson, Jiangning Song, Mike Bell, and Michael J. Sherratt. Predicting proteolysis in complex proteomes using deep learning. *International Journal of Molecular Sciences*, 22(6), 2021.

[38] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[39] Jiale Liu and Xinqi Gong. Attention mechanism enhanced lstm with residual architecture and its application for protein-protein interaction residue pairs prediction. *BMC bioinformatics*, 20(1):1–11, 2019.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. arXiv: 1706.03762.

[41] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, 2022.

[42] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[43] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019. arXiv: 1907.11692.

[44] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[45] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proc. of the IEEE*, pages 1–46, 1998.

[46] William B Cavnar and John M Trenkle. N-Gram-Based Text Categorization. *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, May 2021.

[47] Fuyi Li, Jinxiang Chen, André Leier, Tatiana Marquez-Lago, Quanzhong Liu, Yanze Wang, Jerico Revote, A Ian Smith, Tatsuya Akutsu, Geoffrey I Webb, Lukasz Kurgan, and Jiangning Song. DeepCleave: a deep learning predictor for caspase and matrix metalloprotease substrates and cleavage sites. *Bioinformatics*, 36(4):1057–1065, 09 2019.

[48] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by

back-propagating errors. *nature*, 323(6088):533–536, 1986.

[49] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[50] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations*, 2017.

[51] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

[52] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173. PMLR, 2019.

[53] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". *Advances in neural information processing systems*, 30, 2017.

[54] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13726–13735, 2020.

[55] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[56] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

[57] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009.

[58] Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

[59] Hongyu Shen, Layne C. Price, Taha Bahadori, and Franziska Seeger. Improving generalizability of protein sequence models with data augmentations. *bioRxiv*, 2021.

[60] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[61] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[62] D.R. Cox. The Regression Analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 2(2), 1958.

[63] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[64] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020.

[65] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 125–134, Lille, France, 07–09 Jul 2015. PMLR.

[66] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[67] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[68] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

A Appendix

A.1 Availability

As of January 2023, the publicly available proteasomal cleavage predictors listed in Table 2 are found at:

- ProteaSMM: The official website `http://www.mhc-pathway.net/` does not reply. The model is re-implemented in the epytope framework: `https://github.com/KohlbacherLab/epytope`
- PCM is implemented in the epytope framework: `https://github.com/KohlbacherLab/epytope`
- NetChop 3.1: `https://services.healthtech.dtu.dk/service.php?NetChop-3.1`
- Pcleavage: `http://crdd.osdd.net/raghava/pcleavage/`
- PCPS: `http://imed.med.ucm.es/Tools/pcps/`
- NetCleave: `https://github.com/pepamengual/NetCleave`
- Pepsickle: `https://github.com/pdxgx/pepsickle`
- PUUPL: `https://github.com/SchubertLab/proteasomal-cleavage-puupl`
- This paper: `https://github.com/ziegler-ingo/cleavage_extended`

A.2 Hyperparameters

Table 3: BiLSTM

Hyperparameter	**Range** (Uniformly random choice)	**Final value**	
		C-terminal	**N-terminal**
Epochs trained	≤ 25	15	15
Learning rate	$\{5 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$	3×10^{-4}	3×10^{-4}
Dropout rate	$\{0.45, 0.46, \ldots, 0.51, 0.52\}$	0.5	0.5
Linear layer size	$[120, 181)$	164	179
Embedding dimension	$[50, 201)$	91	76
LSTM size 1	$[220, 281)$	228	252
LSTM size 2	$[450, 520)$	506	518

Table 4: BiLSTM+Attention

Hyperparameter	**Range** (Uniformly random choice)	**Final value**	
		C-terminal	**N-terminal**
Epochs trained	≤ 25	20	20
Learning rate	$\{3 \times 10^{-5}, 5 \times 10^{-5}, 8 \times 10^{-5}, 10^{-4}\}$	10^{-4}	10^{-4}
Dropout rate	$\{0.45, 0.46, \ldots, 0.51, 0.52\}$	0.5	0.5
Linear layer size	$[100, 181)$	147	150
Embedding dimension	$\{120, 124, \ldots, 216, 220\}$	216	216
LSTM size	$[64, 131)$	108	111
Attention heads	$\{1, 2, 4\}$	4	1

Table 5: BiLSTM+Prot2Vec

Hyperparameter	**Range** (Uniformly random choice)	**Final value**	
		C-terminal	**N-terminal**
Epochs trained	≤ 60	60	60
Learning rate	$\{8 \times 10^{-5}, , 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$	3×10^{-4}	3×10^{-4}
Dropout rate	$\{0.45, 0.46, \ldots, 0.51, 0.52\}$	0.5	0.5
Linear layer size	$[120, 180)$	145	139
LSTM size	$[480, 531)$	480	531

Table 6: CNN

Hyperparameter	**Range** (Uniformly random choice)	**Final value**	
		C-terminal	**N-terminal**
Epochs trained	≤ 60	60	60
Learning Rate	$\{8 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$	3×10^{-4}	3×10^{-4}
Dropout Rate	$\{0, 0.02, \ldots, 0.08, 0.1\}$	0.04	0.08
Linear layer size 1	$[64, 101)$	79	89
Linear layer size 2	$[15, 33)$	24	15
Attention heads 1	$\{1, 2, 3, 4, 5, 6\}$	3	4
Attention heads 2	$\{1, 2, 3, 4, 5, 6\}$	2	3
Filter size 1	-	1	1
Number filters 1	$[220, 301)$	220	249
Number filters 2	$[220, 301)$	262	229
Filter size 2a	$\{1, 3, 5, 7\}$	3	1
Filter size 2b	$\{15, 17, 19, 21, 23, 25\}$	17	15
Filter size 3c	$\{13, 15, 17, 19, 21, 23\}$	13	13
Number filters 3	$[350, 431)$	398	400
Filter size 3a	$\{11, 13, 15, 17, 19\}$	11	13
Filter size 3b	$\{13, 15, 17, 19, 21, 23\}$	15	21
Filter size 3c	$\{11, 13, 15, 17, 19\}$	19	15

Table 7: MLP

Hyperparameter	**Range** (Uniformly random choice)	**Final value**	
		C-terminal	**N-terminal**
Epochs trained	≤ 60	30	30
Learning rate	$\{10^{-4}, 5 \times 10^{-4}, 8 \times 10^{-4}, 10^{-3}\}$	10^{-3}	10^{-3}
Dropout rate	$\{0.1, 0.12, \ldots, 0.24, 0.26\}$	0.24	0.24
Linear layer size	$[120, 201)$	144	167

Table 8: Pepsickle

	Hyperparameter	Range (Uniformly random choice)	Final value C-terminal	Final value N-terminal
Sequence-Net	Epochs trained	≤ 100	20	10
	Learning rate	10^{-3}	10^{-3}	10^{-3}
	Dropout rate	$\{0.1, 0.12, \ldots, 0.34, 0.36\}$	0.34	0.32
	Linear layer size 1	$[120, 201)$	187	175
	Linear layer size 2	$[60, 141)$	71	96
	Linear layer size 3	$[20, 81)$	20	44
Motif-Net	Epochs trained	≤ 100	20	10
	Learning rate	$\{10^{-3}$	10^{-3}	10^{-3}
	Dropout rate	$\{0.1, 0.12, \ldots, 0.24, 0.26\}$	0.12	0.12
	Linear layer size 1	$[120, 201)$	200	200
	Linear layer size 2	$[20, 101)$	23	23

Table 9: BiLSTM+ESM2

Hyperparameter	C-terminal	N-terminal
Epochs trained	10	10
Learning rate	3×10^{-4}	3×10^{-4}
Dropout rate	0.5	0.5
Linear layer size	128	128
LSTM size	512	512

Table 10: ESM2

Hyperparameter	C-terminal	N-terminal
Epochs trained	3	3
Learning rate	2×10^{-5}	2×10^{-5}
Dropout rate	0.5	0.5

Table 11: BiLSTM+T5

Hyperparameter	C-terminal	N-terminal
Epochs trained	10	10
Learning rate	3×10^{-4}	3×10^{-4}
Dropout rate	0.5	0.5
Linear layer size	128	128
LSTM size	512	512

Table 12: BiLSTM+BBPE1, BiLSTM+BBPE50, BiLSTM+WP50

Hyperparameter	**C-terminal**	**N-terminal**
Epochs trained	15	15
Learning rate	10^{-4}	10^{-4}
Dropout rate	0.5	0.5
Embedding dimension	150	150
Linear layer size	128	128
LSTM size	512	512

Table 13: BiLSTM+FwBw

Hyperparameter	**C-terminal**	**N-terminal**
Epochs trained	15	15
Learning rate	10^{-4}	10^{-4}
Dropout rate	0.5	0.5
Linear layer size 1	128	128
LSTM size 1	128	128
LSTM size 2	512	512
Sequence encoding embedding dimension	100	100
Sequence encoding BiLSTM size	200	200

Table 14: Logistic Regression

Hyperparameter	**C-terminal**	**N-terminal**
Maximum Iteration	100	100
Cs	10	10
Penalty	L2	L2
Fit Intercept	Yes	Yes
Intercept Scaling	1	1
Solver	LBFGS	LBFGS
Tolerance	10^{-4}	10^{-4}

PUUPL: See configuration files at `https://github.com/SchubertLab/proteasomal-cleavage-puupl/tree/main/configs`

Table 15: Denoising methods (Co-Teaching, Co-Teaching+, JoCoR, DivideMix, Noise adaptation layer)

Hyperparameter	**Co-teaching**	**Co-teaching+**	**JoCoR**	**DivideMix**	**Noise adaptation layer**
Number scale-up epochs	10	10	10	5	-
Noise rate	0.2	0.2	0.2	-	-
Forget rate	0.2	0.2	0.1	-	-
Exponent	1	1	1	-	-
Number warm-up epochs	-	-	-	1	1
α	-	-	-	0.5	-
λ_u	-	-	-	0	-
Probability threshold	-	-	-	0.5	-
Temperature	-	-	-	0.5	-
β	-	-	-	-	0.8

A.3 Results without denoising methods

Table 16: Model performances without denoising

Terminal	Prio.	Models	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	1	BiLSTM	88.55 ± 0.12	86.97 ± 0.09	25	15	4 655 984	4 655 984
		BiLSTM+Attention	88.28 ± 0.08	86.86 ± 0.11	20	20	1 632 391	1 632 391
		BiLSTM+Prot2Vec	87.99 ± 0.14	86.57 ± 0.13	20	15	16 009 371	5 830 371
		CNN	86.66 ± 0.17	85.74 ± 0.12	45	60	16 084 198	16 084 198
		MLP	88.17 ± 0.11	86.78 ± 0.09	4	30	30 529	30 529
	2	BiLSTM+ESM2	88.34 ± 0.05	86.87 ± 0.08	330	10	152 998 267	4 858 113
		ESM2	88.32 ± 0.16	86.48 ± 1.51	900	3	148 140 188	148 140 188
		BiLSTM+T5	88.32 ± 0.05	86.89 ± 0.04	780	10	1 214 572 801	6 430 977
	3	BiLSTM+BBPE1	85.25	85.48	13	15	5 319 409	5 319 409
		BiLSTM+BBPE50	68.92	82.46	12	15	12 669 409	12 669 409
		BiLSTM+WP50	73.46	83.04	16	15	12 669 409	12 669 409
		BiLSTM+FwBw	87.59	86.15	120	15	4 315 369	4 315 369
N-terminal	1	BiLSTM	79.50 ± 0.11	83.51 ± 0.11	25	15	5 096 135	5 096 135
		BiLSTM+Attention	79.24 ± 0.11	83.36 ± 0.13	20	20	1 718 233	1 718 233
		BiLSTM+Prot2Vec	79.10 ± 0.11	83.22 ± 0.13	20	15	16 772 049	6 593 049
		CNN	77.30 ± 0.82	82.89 ± 0.22	45	60	15 237 057	15 237 057
		MLP	79.08 ± 0.11	83.33 ± 0.12	4	30	35 405	35 405
	2	BiLSTM+ESM2	79.24 ± 0.10	83.35 ± 0.09	360	10	152 998 267	4 858 113
		ESM2	78.91 ± 0.18	82.63 ± 0.64	900	3	148 140 188	148 140 188
		BiLSTM+T5	79.48 ± 0.11	83.45 ± 0.08	780	10	1 214 572 801	6 430 977
	3	BiLSTM+BBPE1	76.56	82.88	13	15	5 319 409	5 319 409
		BiLSTM+BBPE50	68.67	82.03	12	15	12 669 409	12 669 409
		BiLSTM+WP50	69.28	82.08	15	15	12 669 409	12 669 409
		BiLSTM+FwBw	78.71	83.15	120	15	4 315 369	4 315 369

A.4 Results with denoising methods

Table 17: BiLSTM with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	87.50	86.64	41	15	4 655 984	4 655 984
	Co-Teaching+	87.50	86.64	41	15	4 655 984	4 655 984
	JoCoR	84.53	85.49	41	15	4 655 984	4 655 984
	DivideMix	86.25	84.02	210	15	4 656 149	4 656 149
	Noise Adaptation Layer	88.49	87.02	22	15	4 656 149	4 656 149
N-terminal	Co-Teaching	78.28	83.21	44	15	5 096 135	5 096 135
	Co-Teaching+	78.37	83.17	44	15	5 096 135	5 096 135
	JoCoR	74.26	82.12	44	15	5 096 135	5 096 135
	DivideMix	77.08	81.52	210	15	5 096 315	5 096 315
	Noise Adaptation Layer	79.48	83.43	23	15	5 096 315	5 096 315

Table 18: BiLSTM+Attention with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	86.82	86.20	40	20	1 632 391	1 632 391
	Co-Teaching+	86.91	86.16	40	20	1 632 391	1 632 391
	JoCoR	81.70	85.04	38	20	1 632 391	1 632 391
	Noise Adaptation Layer	88.23	86.90	18	20	1 632 539	1 632 539
N-terminal	Co-Teaching	77.48	83.13	43	20	1 718 233	1 718 233
	Co-Teaching+	78.09	83.13	42	20	1 718 233	1 718 233
	JoCoR	68.86	82.40	41	20	1 718 233	1 718 233
	Noise Adaptation Layer	79.21	83.37	22	20	1 718 384	1 718 384

Table 19: BiLSTM+Prot2Vec with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	85.47	85.78	32	15	16 009 371	5 830 371
	Co-Teaching+	85.47	85.78	31	15	16 009 371	5 830 371
	JoCoR	75.80	82.20	32	15	16 009 371	5 830 371
	Noise Adaptation Layer	87.93	86.60	15	15	16 009 517	5 830 517
N-terminal	Co-Teaching	76.64	83.00	40	15	16 772 049	6 593 049
	Co-Teaching+	77.44	82.90	40	15	16 772 049	6 593 049
	JoCoR	61.62	81.91	40	15	16 772 049	6 593 049
	Noise Adaptation Layer	78.96	83.15	19	15	16 772 189	6 593 189

Table 20: CNN with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	71.51	82.35	102	60	16 084 198	16 084 198
	Co-Teaching+	76.18	82.91	102	60	16 084 198	16 084 198
	JoCoR	71.10	82.92	102	60	16 084 198	16 084 198
	Noise Adaptation Layer	85.91	85.50	49	60	16 084 223	16 084 223
N-terminal	Co-Teaching	60.71	81.88	102	60	15 237 057	15 237 057
	Co-Teaching+	74.15	82.05	102	60	15 237 057	15 237 057
	JoCoR	60.81	81.88	102	60	15 237 057	15 237 057
	Noise Adaptation Layer	76.68	82.78	47	60	15 237 073	15 237 073

Table 21: MLP with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	87.14	86.45	6	30	30 529	30 529
	Co-Teaching+	87.16	85.85	6	30	30 529	30 529
	JoCoR	86.11	85.56	5	30	30 529	30 529
	Noise Adaptation Layer	88.07	86.73	3	30	30 674	30 674
N-terminal	Co-Teaching	77.89	83.21	6	30	35 405	35 405
	Co-Teaching+	78.34	83.11	5	30	35 405	35 405
	JoCoR	76.31	82.30	5	30	35 405	35 405
	Noise Adaptation Layer	78.82	83.32	4	30	35 573	35 573

Table 22: BiLSTM+ESM2 with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	87.19	86.52	660	10	152 998 267	4 858 113
	Co-Teaching+	87.19	86.52	660	10	152 998 267	4 858 113
	JoCoR	84.52	84.93	660	10	152 998 267	4 858 113
	Noise Adaptation Layer	88.32	86.88	330	10	152 998 396	4 858 242
N-terminal	Co-Teaching	77.76	83.25	660	10	152 998 267	4 858 113
	Co-Teaching+	78.03	83.15	660	10	152 998 267	4 858 113
	JoCoR	73.71	81.96	660	10	152 998 267	4 858 113
	Noise Adaptation Layer	79.29	83.37	360	10	152 998 396	4 858 242

Table 23: ESM2 with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	87.93	85.56	2160	3	148 140 188	148 140 188
	Co-Teaching+	87.93	85.56	2100	3	148 140 188	148 140 188
	JoCoR	87.31	86.25	2100	3	148 140 188	148 140 188
	Noise Adaptation Layer	87.97	86.58	960	3	148 140 222	148 140 222
N-terminal	Co-Teaching	78.53	80.54	2220	3	148 140 188	148 140 188
	Co-Teaching+	78.53	80.54	2220	3	148 140 188	148 140 188
	JoCoR	77.83	83.02	2160	3	148 140 188	148 140 188
	Noise Adaptation Layer	77.31	82.13	930	3	148 140 222	148 140 222

Table 24: BiLSTM+T5 with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	87.11	86.40	1500	10	1 214 572 801	6 430 977
	Co-Teaching+	87.11	86.40	1500	10	1 214 572 801	6 430 977
	JoCoR	83.48	83.88	1500	10	1 214 572 801	6 430 977
	Noise Adaptation Layer	88.36	86.85	720	10	1 214 572 930	6 431 106
N-terminal	Co-Teaching	78.30	83.22	1560	10	1 214 572 801	6 430 977
	Co-Teaching+	78.32	83.26	1560	10	1 214 572 801	6 430 977
	JoCoR	71.76	82.28	1500	10	1 214 572 801	6 430 977
	Noise Adaptation Layer	79.54	83.48	780	10	1 214 572 930	6 431 106

Table 25: BiLSTM+BBPE1 with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	76.55	83.84	33	15	5 319 409	5 319 409
	Co-Teaching+	80.31	83.44	34	15	5 319 409	5 319 409
	JoCoR	58.81	82.20	33	15	5 319 409	5 319 409
	Noise Adaptation Layer	85.15	85.45	17	15	5 319 538	5 319 538
N-terminal	Co-Teaching	69.12	82.19	34	15	5 319 409	5 319 409
	Co-Teaching+	74.65	82.37	34	15	5 319 409	5 319 409
	JoCoR	58.03	81.88	33	15	5 319 409	5 319 409
	Noise Adaptation Layer	76.15	82.58	17	15	5 319 538	5 319 538

Table 26: BiLSTM+BBPE50 with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	57.62	82.28	30	15	12 669 409	12 669 409
	Co-Teaching+	67.40	82.35	30	15	12 669 409	12 669 409
	JoCoR	52.09	82.20	30	15	12 669 409	12 669 409
	Noise Adaptation Layer	67.42	82.38	16	15	12 669 538	12 669 538
N-terminal	Co-Teaching	62.52	81.88	31	15	12 669 409	12 669 409
	Co-Teaching+	67.45	81.98	32	15	12 669 409	12 669 409
	JoCoR	53.02	81.88	31	15	12 669 409	12 669 409
	Noise Adaptation Layer	67.90	81.95	15	15	12 669 538	12 669 538

Table 27: BiLSTM+WP50 with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	59.31	82.25	40	15	12 669 409	12 669 409
	Co-Teaching+	69.37	82.57	40	15	12 669 409	12 669 409
	JoCoR	53.45	82.20	39	15	12 669 409	12 669 409
	Noise Adaptation Layer	72.80	82.98	20	15	12 669 538	12 669 538
N-terminal	Co-Teaching	62.66	81.92	37	15	12 669 409	12 669 409
	Co-Teaching+	68.44	81.97	37	15	12 669 409	12 669 409
	JoCoR	56.25	81.88	36	15	12 669 409	12 669 409
	Noise Adaptation Layer	68.91	81.99	20	15	12 669 538	12 669 538

Table 28: BiLSTM+FwBw with denoising

Terminal	Denoising methods	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	Co-Teaching	85.84	84.92	270	15	4 315 369	4 315 369
	Co-Teaching+	85.84	84.92	270	15	4 315 369	4 315 369
	JoCoR	61.26	82.20	270	15	4 315 369	4 315 369
	Noise Adaptation Layer	87.75	86.09	120	15	4 315 498	4 315 498
N-terminal	Co-Teaching	76.86	82.45	270	15	4 315 369	4 315 369
	Co-Teaching+	77.02	82.88	270	15	4 315 369	4 315 369
	JoCoR	62.00	81.88	270	15	4 315 369	4 315 369
	Noise Adaptation Layer	78.64	83.09	120	15	4 315 498	4 315 498

A.5 Ablation results

Table 29: Dataset size ablation of BiLSTM

Terminal	Size	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	10%	87.29 ± 0.38	86.29 ± 0.26	2	15	4 655 984	4 655 984
	25%	87.88 ± 0.18	86.61 ± 0.18	6	15	4 655 984	4 655 984
	50%	88.23 ± 0.24	86.91 ± 0.11	12	15	4 655 984	4 655 984
	75%	88.43 ± 0.14	86.94 ± 0.10	18	15	4 655 984	4 655 984
	100%	88.55 ± 0.12	86.97 ± 0.09	25	15	4 655 984	4 655 984
N-terminal	10%	77.73 ± 0.62	82.93 ±0.26	2	15	5 096 135	5 096 135
	25%	78.60 ± 0.32	83.22 ± 0.22	6	15	5 096 135	5 096 135
	50%	79.12 ± 0.16	83.34 ± 0.09	12	15	5 096 135	5 096 135
	75%	79.32 ± 0.13	83.42 ± 0.13	18	15	5 096 135	5 096 135
	100%	79.50 ± 0.11	83.51 ± 0.11	25	15	5 096 135	5 096 135

Table 30: Window size ablation of BiLSTM

Terminal	Halfwindow Size	AUC	ACC	s/epoch	Epochs	Parameters	Trainable
C-terminal	3	86.17 ± 0.11	85.51 ± 0.09	17	15	4 655 984	4 655 984
	7	89.05 ± 0.12	87.46 ± 0.11	30	15	4 655 984	4 655 984
	14	92.12 ± 0.10	89.77 ± 0.09	57	15	4 655 984	4 655 984
	21	92.20 ± 0.09	89.86 ± 0.06	84	15	4 656 648	4 656 648
	28	92.35 ± 0.18	89.96 ± 0.14	109	15	4 656 648	4 656 648
N-terminal	3	76.68 ± 0.27	82.71 ± 0.06	21	15	5 096 135	5 096 135
	7	82.52 ± 0.23	84.61 ± 0.09	34	15	5 096 135	5 096 135
	14	89.17 ± 0.14	88.28 ± 0.14	64	15	5 096 135	5 096 135
	21	89.20 ± 0.23	88.38 ± 0.15	90	15	5 096 829	5 096 829
	28	89.34 ± 0.15	88.46 ± 0.12	115	15	5 096 829	5 096 829

Table 31: Size variations with weight decay for BiLSTM on C-terminal with a halfwindow size of 28

Modal size	Weight decay	Max AUC	Epoch	s/epoch	Epochs trained	Parameters	Trainable
50%	0	91.89	5	210	15	10 464 925	10 464 925
	1×10^{-2}	87.25	15	210	15	10 464 925	10 464 925
	1×10^{-3}	89.55	15	210	15	10 464 925	10 464 925
	1×10^{-4}	91.68	19	210	20	10 464 925	10 464 925
	1×10^{-5}	91.85	6	210	15	10 464 925	10 464 925
100%	0	91.84	4	330	15	10 464 925	10 464 925
	1×10^{-2}	87.31	15	330	15	10 464 925	10 464 925
	1×10^{-3}	89.72	15	330	15	10 464 925	10 464 925
	1×10^{-4}	91.70	18	330	20	10 464 925	10 464 925
	1×10^{-5}	91.77	5	330	15	10 464 925	10 464 925