Signal Peptide Detection Using Small, Resource-Efficient Language Models

MOL3022 Bioinformatics - Project report

Authored by: Andreas Bentzen Winje
Code contribution: Yauhen Yavorski

Abstract

Background: Signal peptides (SPs) are small sequences at the start of a protein sequence that play a role in the translocation, secretion, and ultimately the function of proteins, which has been a major research focus in the field of bioinformatics. In this project, Microsoft's MiniLM language model was trained for binary classification on a dataset of 29101 protein sequences to predict whether a protein sequence contains a signal peptide or not.
Results: After training for just over 3 hours (50 epochs), the model reached an F1-score of 98.5%, with an equal precision and recall score, showing the efficacy of the model on the problem. The model was also tested across kingdoms, where it was found that Eukarya were harder to predict than other kingdoms.
Conclusion: To conclude, the model performed well given its size. Further work includes looking into how the performance of the model can be optimized, how well the model would perform with more classes, and how recall and precision should be traded off in practical applications. Comparing results with other methods should also be focused on. Ideally, this project provides insight into and inspiration for how data-driven approaches can be further utilized in the field of bioinformatics.

Keywords— Signal peptide · Protein function · Sequence classification · Deep learning · Language model

Department of Clinical and Molecular Medicine
Norwegian University of Science and Technology (NTNU)
Trondheim, April 2024

NTNU

1 Background

The prediction of signal peptides in protein sequences has served as an important research area in the last few years. As more and more proteins are being identified following the application of advanced computational methods, the significance of understanding the function of proteins, and thus also the cellular location of proteins, becomes increasingly apparent.

Signal peptides are short sequences of amino acids located at the beginning of a protein sequence (the N-terminus region) of newly synthesized proteins that are cleaved off after translocation of the protein. They play a crucial role in directing the mature protein to its intended cellular destination, whether it is transported somewhere inside the cell (in the case of non-secretory proteins), or secreted outside the cell. Knowing the cellular destination of proteins can, in turn, give vital insight for predicting the function of these proteins - a considerable and enduring problem in medical and biological research.

In this project, we explore the usage of small language models (LMs) for the automatic detection of signal peptides within protein sequences.

1.1 Related work

As neatly outlined by Nielsen et al. [6], solving the problem of signal peptide prediction necessitates solving two related sub-tasks: (1) discriminating between SP and non-SP proteins, and (2) determining the position of the cleavage site of the SP in proteins that have SPs [6]. Both tasks have been addressed to varying extents by the research community in the past, but it is with the introduction of Artificial Neural Networks (ANNs) and deep learning that we see the most significant promise.

As AI is entering the intersection between computer science and biology, new advances in DNA sequencing and big data are allowing the application of deep learning techniques to perform large-scale predictions computationally. With the application of modern language model architectures, like the transformer architecture [9], the development of language models specifically catered to bioinformatical needs, as exemplified by ProtBERT [3], is possible. ProtBERT is a version of the auto-encoding language model BERT [1] that has been fine-tuned on protein sequences to capture the essence of what they call the '*grammar of the language of life*' [3].

Approaches to specifically signal peptide prediction from the field of AI have traditionally relied on the use of biological properties encoded into Hidden Markov Models (HMMs) in conjunction with deep learning to perform accurate predictions [6] [8]. More recent developments, however, have seen the use of deep learning alone for a fully data-driven approach without the need for biological properties. This is exemplified by the approach of Dumitrescu et al. [2] from 2022 called TSignal, where they modified the ProtBERT model with their own transformer decoder to predict the cleavage sites of all SP types with high accuracy [2].

For an overview of the earlier history of signal peptide prediction (up until 2019), the reader is directed to the paper titled '*A Brief History of Protein Sorting Prediction*' by Nielsen et al. [6].

1.2 Problem definition and scope

This project aims to (1) develop a binary classification model for predicting the presence of signal peptides in protein sequences using a language model, and, ultimately, (2) assess the efficacy of using deep learning as an approach to signal peptide prediction in practical applications.

The scope of the project is limited to using resource-efficient methods as hardware is limited to a single NVIDIA GTX 1060 6GB GPU. This entails that comparative methods are limited to reported results or testing of small models only.

In this report, we will first look at the implementation of the project, like the dataset used, the architecture and training of the classification model, and the metrics used for the evaluation of the model. We then proceed to the calculation and reporting of the metrics from the model's performance. This is followed by a discussion of the results and the model's practical applications based on these results.

2 Implementation

2.1 SP-detection as a binary classification problem

The detection of which protein contains a signal peptide (SP) and not can be formulated as a binary classification problem with two classes, one positive class for SP protein, and one negative class for non-SP proteins. These classes are represented by numerical values where positive is 1 and negative is 0. For a formal overview, see table 1.

Class type	Class label	Description
Positive (1)	`SP`	a signal peptide is present in the protein
Negative (0)	`NO_SP`	no signal peptide is present in the protein

Table 1: Overview of the classes used in classification.

As features, the protein sequence is used together with the corresponding kingdom (also called organism group). This is inspired by Dumitrescu et al. [2], where they use the kingdom as an extra feature to give the classifier more context to work with, as it is not given that the model will learn from the protein sequences alone. How this is done is further explained below.

2.2 Dataset

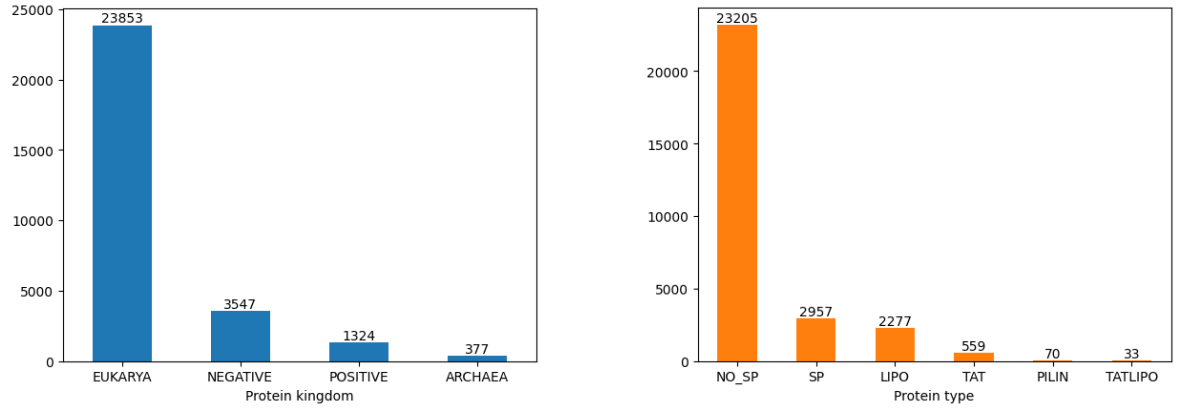
2.2.1 Dataset description

The dataset used for this project is taken from Teufel et al. [8] and contains 29101 samples of protein sequences and their corresponding type (signal peptide), organism group (annotated "kingdom"), Uniprot Access Number (AC), and partition number. The type corresponds to which signal peptide, if any, is present in the protein sequence, and takes one of six values: `NO_SP` (non-SP proteins), `SP` (standard SPs), `LIPO` (lipoprotein SPs), `TAT` (tat SPs), `TATLIPO` (tat-lipoprotein SPs), and `PILIN` (pilin or pilin-like SPs). Additionally, each sample has a sequence of classifications for each amino acid in the protein sequence classifying which signal peptide or mature protein type (intracellular, extracellular, or transmembrane) it corresponds to. The protein sequences are each of one of four kingdoms, eukarya (`EUKARYA`), archaea (`ARCHAEA`), gram-positive bacteria (`POSITIVE`), and gram-negative bacteria (`NEGATIVE`).

To understand the data better, a preliminary analysis of the distributions within the data is conducted. Figure 1a shows the count of data points according to each of the kingdoms. Similarly, fig. 1b shows the distribution of type in the dataset. As it is evident that there are imbalances in the dataset concerning both kingdom and type, measures are taken to deal with this imbalance, which are presented below.

2.2.2 Preprocessing

From the original dataset, only the kingdom, protein sequence, and protein type variables are used in the processed dataset. Here, the protein type is used as a label, where protein types of `NO_SP` is the negative label and protein types of `SP` is the positive label. To reduce the number of labels, and to align the dataset with the scope of binary classification, proteins of other types (other signal peptides) are also classified as `SP`. This reduction of the labels is formally illustrated in fig. 2. Following this, the `SP`-label has a total of 5896 samples, against the 23205 samples of the `NO_SP`-label.



(a) Protein kingdom distribution

(b) Protein type distribution

Figure 1: Distribution of data points with regards to (a) kingdom and (b) protein type.

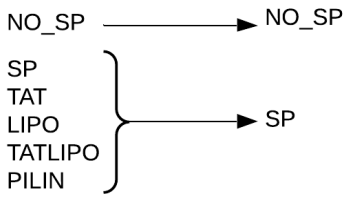


Figure 2: Reduction of labels in the dataset. Here, the left side represents the original dataset with six labels, and the right side is the processed dataset with only two labels.

Further, the kingdom variable of each data point is concatenated to the front of the protein sequence, separated by as `[SEP]` (separation)-token[1], to form one single sequence used as a feature. This is similar to the method of Dumitrescu et al. [2], although not identical. A representation of the final structure of each data point after processing can be seen in fig. 3. Each feature is then tokenized[2] before it is passed to the classification model.

Feature: `"{KINGDOM} [SEP] {PROTEIN SEQUENCE}"`
Label: $y \in \{\texttt{NO\_SP}, \texttt{SP}\}$

Figure 3: General structure of a data point in the processed dataset.

2.2.3 Dataset splits

The 29101 samples of the dataset are split up into a train set and a test set, where the train set contains 20290 samples ($\approx 69.7\%$) and the test set contains 8811 samples ($\approx 30.3\%$). In addition, 1762 samples (20%) of the test set are used for validation while training. A separate validation set is not used as no hyperparameter tuning is performed, and is thus not needed.

2.3 Classification model

The model used for fine-tuning[3] is based on the MiniLM model (specifically MiniLMv1-L12-H384-uncased) by Wang et al. [10], a pre-trained transformer model distilled from the original BERT-base model [1] [10].

[1] A separation token is used in language models to signify a change in textual context.
[2] Tokenization of a text means breaking the text into segments called tokens, e.g. words or subwords.
[3] Fine-tuning a pre-trained machine learning model means to re-train the model for a specific (downstream) task.

While the MiniLM model is not pre-trained specifically on protein sequences, like the ProtBERT model [3] (referenced in section 1.1), it has a significantly smaller resource requirement and has a faster (2.7x, as reported by the authors [10]) inference time. For reference, the MiniLM model has a parameter count of only 33M [10], while the ProtBERT model is significantly larger at 420M parameters [3]. Using a smaller model aligns with the limited scope of the project and allows for testing the efficacy of smaller models for the benefit of future research on efficient signal peptide detectors. Since the MiniLM model is not pre-trained on protein sequences, however, the effectiveness of its tokenizer on protein sequences is not guaranteed to be adequate. For this reason, the tokenizer of the MiniLM model is switched out by the tokenizer of the ProtBERT model for effective tokenization of the input strings.

Further, as the MiniLM model is trained for outputting tokens and not classifications, the output layer is replaced by a newly initialized classification layer (fully-connected layer) with two outputs, one for each class (`NO_SP` and `SP`), which is then trained from scratch on top of the fine-tuning of the MiniLM model. For a full architectural overview, see fig. 4.

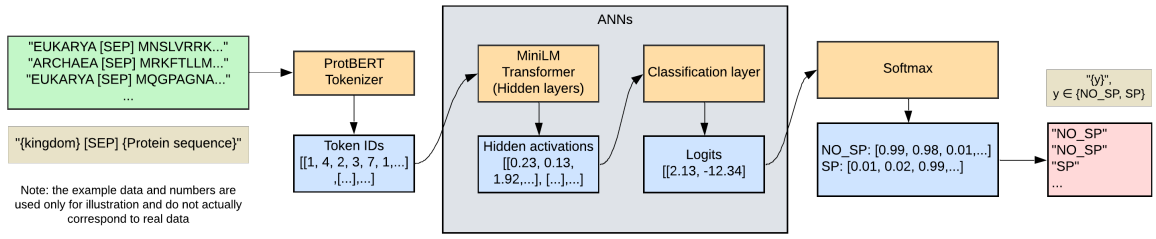


Figure 4: Overview of the architecture. The input (green) is passed to the ProtBERT tokenizer to get the token IDs, these are then passed to the MiniLM model and classification model to get the model output (logits). The model outputs are then passed through a softmax function to get prediction probabilities, giving out the final outputs (red). The MiniLM and classification layer (the artificial neural networks (ANNs), gray box) are trained beforehand on the sample data.

2.4 Training setup

2.4.1 Label balancing and loss function

The labels are balanced to prevent the model from overfitting to one label. This is done by using weighted loss, where the loss of each sample is given a weight according to its label. This is done by calculating the average number of samples per label in the dataset and dividing the average by the number of samples per label ($\frac{n_{\text{NO\_SP}}+n_{\text{SP}}}{2*n_{\text{NO\_SP}}}$ and $\frac{n_{\text{NO\_SP}}+n_{\text{SP}}}{2*n_{\text{SP}}}$ for `NO_SP` and `SP`, respectively). Here, a weight of 0.6493 was used for the `NO_SP`-label and a weight of 2.1747 was used for the `SP`-label.

For the loss function, the Focal Loss function by Lin et al. [4] is used with a gamma of $\gamma = 2$, as it has shown good performance on unbalanced data using label weights [4].

2.4.2 Hyperparameters

The model is trained for 50 epochs with an initial learning rate of 2e-5 and a linear learning rate scheduler, a weight decay of 1e-04, and AdamW [5] as optimizer. The hardware used for training the model was a single NVIDIA GTX 1060 6GB GPU.

2.5 Evaluation metrics

For testing and mid-training validation, the metrics precision, recall, and F1-score are calculated. The metrics are calculated using true and false positives and negatives (TP, FP, TN, and FN, respectively),

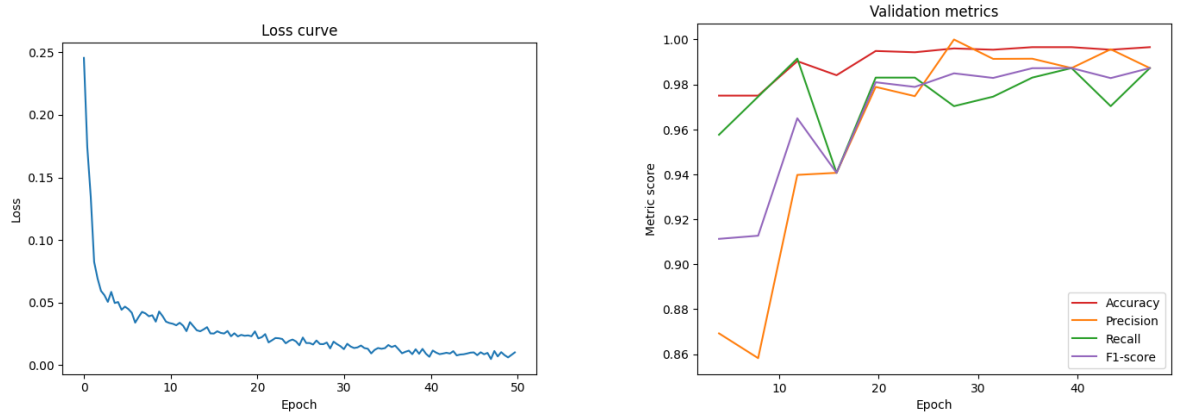
where positives and negatives are defined as the predictions of the model according to the classes outlined in section 2.1, while true and false refer to whether the predictions correspond to the actual label of the data point or not. Precision is calculated as $P = \frac{TP}{TP+FP}$ and recall is calculated as $R = \frac{TP}{TP+FN}$. Here, precision gives an idea of how good the model is at only classifying SP proteins as SP, and recall gives an idea of how good the model is at classifying SP proteins as only SP. The F1 score is the harmonic mean of precision and recall and is calculated as $\frac{2*P*R}{P+R}$, indicating the model's overall performance. All three metrics are practical for determining the efficacy and usefulness of the model.

3 Results

3.1 Training results

The full runtime of training the model was approximately `3,09 hours`. The model's training loss progression can be seen in fig. 5a. Initially, a significant decline in loss can be seen, after which it stabilizes and seems to converge, which indicates effective learning. Note that the use of the learning rate scheduler, which lowers learning rate during training, could also have an effect on the loss graph, and it might not be at its minimum.

Figure 5b shows the evolution of the different metric scores over the training time. This also shows promising results, where all metrics scores are already quite high by the first validation with a minimum of 0.85 points and an increase of up to 0.10 points during training. The metrics also do not seem to increase much by the end, which shows that the model might have reached its optimum given the current hyperparameters.



(a) Loss curve

(b) Metrics from training-validation

Figure 5: Results from training: (a) shows how the model loss decreases during training, (b) shows the metrics on the validation set at various points during training. Accuracy is also reported, although it is not important.

3.2 Testing results

To determine the model's final performance, we will look at the testing results. Table 2 shows the overall final metrics of the model after training. As can be seen, all of precision, recall and F1-score were high at a score of 0.985. To take it into perspective, this means that of all proteins containing a signal peptide, 98.5% of them were successfully detected (recall), and of all proteins that were detected to contain a signal peptide, 98.5% of them actually had a signal peptide (precision).

This can be more easily understood from the confusion matrix in fig. 6, where the individual true and false positives and negatives are reported. It is important to note that the percentages in the confusion

Precision	Recall	F1
0.985	0.985	0.985

Table 2: Overall metric performance.

matrix only tell the distribution of cases between true and false positives and negatives and not the actual performance. As one can see, the true negatives make up most of the cases (85.82%), which is also expected due to the imbalance in the dataset. More interesting is how many false values are reported, which from the confusion matrix can be seen to make up, in total, only 36 (0.4%) of the 8811 test samples.

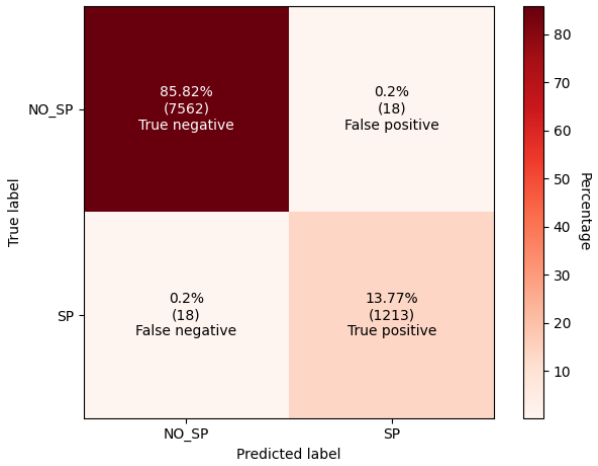


Figure 6: Confusion matrix for final test performance. The confusion matrix shows the actual label (y-axis) against the predicted label (x-axis) (true and false positives and negatives), where the percentages (and numbers in parenthesis) show the distribution of combinations of these.

Further, since there was also an imbalance in the distribution of kingdoms (as discussed in section 2.2.1), where a majority of proteins were from Eukarya, and to disclose if there is any imbalance in how well the model performs across each kingdom, a test metric per kingdom is also calculated. The results from this can be seen in table 3, where the variance of each metric over kingdoms is also reported. As can be seen, the variances of all metrics are relatively low with a variance of at most 5.631e-4 (for precision). Interestingly, despite being the most common kingdom in the dataset, which means that the model has more data on Eukarya to learn with, the metric scores for Eukarya are overall lower than all of the other kingdoms.

	Precision	Recall	F1-score
Eukarya	0.939	0.957	0.948
Archaea	0.989	*1.000	*0.995
Gram-negative	0.995	0.989	0.992
Gram-positive	*0.997	0.993	*0.995
σ^2	5.631e-04	2.714e-04	3.936e-04

Table 3: Performance metrics for all kingdoms and their statistical variance between kingdoms. An asterisk marks the best kingdom for each metric.

4 Discussion

As we can see from the results in section 3.2, the model was indeed able to learn and generalize from the data with a test-F1-score of 98.5%, despite its limited size of only 33M parameters. While this is a high number, the possibility for comparative studies is limited as the recent focus has been on the detection of the cleavage sites or discrimination between signal peptides (SPs) rather than the detection of the presence of SPs, whose performance measures are not directly comparable. This is something that should be taken into account in further work, however. The current performance does anyhow seem to suggest that the model has significant potential, given its size.

The performance of the model is not given to be optimal either, due to the limited training and testing of the model. One way to potentially increase the performance is to perform hyperparameter tuning, which for this project was not performed due to the constraints of the project. It might be that the hyperparameters used for the model, like learning rate and weight decay, are not optimal and would lead to higher performance if tuned correctly. This also includes training the model for a longer time, as, as mentioned in section 3.1, the model might not have reached its minimum loss. One thing to note is the efficacy of using FocalLoss as the loss function, as, despite the dataset's imbalance, the model generalized well for both positive and negative results. This is evident in that false positives and negatives have the same amount of 18 cases each.

It might also be that the model is not able to generalize better given its current size, and a larger model could be necessary to increase the performance. As already said in section 2.3, using a smaller model is desirable as it furthers the research on resource-efficient methods, and this also entails that finding the increase in performance per increase in model size is also desirable, as to find the optimal size given a resource and performance requirement.

For practical applications, however, depending on the application, the current performance is not necessarily inadequate. Although most methods today perform more complex tasks that go beyond merely the detecting of SP proteins (like also discriminating between SPs and detecting their cleavage sites), one practical application of our model could be for pre-filtering SP proteins for use in an integrated system where performance is important. Say, for example, one has an arbitrarily large dataset of both SP and non-SP proteins, where the non-SP proteins make up a majority of the samples. If one would like to perform large-scale data-driven analysis on the SP proteins using a large model like TSignal [2], it would make sense to filter out all non-SP proteins using a smaller model first to avoid spending unnecessary resources on analyzing non-SP proteins with the large model. If the model would also be trained to discriminate between SPs, this could also be used to study specific SPs. These scenarios make even more sense when taking into consideration the near-exponential growth of data generated and used in bioinformatical research, a trend that is likely to continue [7]. The development of resource-efficient models could ultimately allow even low-resource researchers to perform large-scale analyses of their data.

This would of course require a higher recall score, as one would not like to miss out on potentially 1.5% of the SP samples (with a recall of 98.5%, as is the case for our model). The model could be optimized for recall by using a lower threshold for the confidence score[4] of the model, which in turn would potentially impact the model's precision, and thus an acceptable trade-off between precision and recall must be found. This trade-off could be analyzed using metrics like PR curves[5] and AUC-PR[6], which would require further analysis of the model and should be explored in further work.

[4]The Confidence score is a score between 0 and 1 given to each class by a prediction model. It tells how "confident" the model is of the data to pertain to that class.

[5]Precision-Recall curves, used for mapping precision against recall to analyze the precision performance as the recall is increased.

[6]Area Under Precision-Recall Curve, defined as the integral of the Precision-Recall curve and serves as a single number to summarize the performance when increasing recall.

5 Conclusion

In conclusion, the model reached a good performance in detecting signal peptide (SP) and non-SP proteins. Although the domain is limited to binary classification, there are practical applications for such a model, for example for use as a module in an integrated system, like for preliminary filtering of large datasets.

For further work, more analysis of the model should be conducted. This includes how using different hyperparameters and sizes can increase the performance, and looking into how a desirable trade-off between precision and recall can be found using e.g. PR and AUC-PR metrics. The detection of multiple classes for discriminating between the SPs using a similarly-sized model should also be explored.

The work in this project shows how resource-efficient machine learning models can be trained and used in practical applications, and should ideally serve as an inspiration to further the utilization of fully data-driven computational methods and resource-efficient solutions within the field of bioinformatics.

Availability and software

All code, data, and models can be found at the following links:

- **Original dataset** [8]: https://services.healthtech.dtu.dk/services/SignalP-6.0/
- **Backend and training code**: https://github.com/andreas122001/MOL3022-bioinformatics-project
- **Frontend code**: https://github.com/Senja20/mol-3022-front-end-application
- **Model card**: https://huggingface.co/andreas122001/mol3022-signal-peptide-prediction

References

[1] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv* (Oct. 2018). DOI: 10.48550/arXiv.1810.04805. eprint: 1810.04805. URL: https://arxiv.org/abs/1810.04805v2.

[2] Alexandru Dumitrescu et al. "TSignal: a transformer model for signal peptide prediction". In: *Bioinformatics* 39.Supplement_1 (June 2023), pp. i347–i356. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad228. URL: https://doi.org/10.1093/bioinformatics/btad228.

[3] Ahmed Elnaggar et al. "ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning". In: *bioRxiv* (May 2021), p. 2020.07.12.199554. DOI: 10.1109/TPAMI.2021.3095381. eprint: 2020.07.12.199554. URL: https://doi.org/10.1101/2020.07.12.199554.

[4] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *arXiv* (Aug. 2017). DOI: 10.48550/arXiv.1708.02002. eprint: 1708.02002.

[5] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *arXiv* (Nov. 2017). DOI: 10.48550/arXiv.1711.05101. eprint: 1711.05101.

[6] Henrik Nielsen et al. "A Brief History of Protein Sorting Prediction". In: *Protein J.* 38.3 (June 2019), pp. 200–216. ISSN: 1875-8355. DOI: 10.1007/s10930-019-09838-3. URL: https://link.springer.com/article/10.1007/s10930-019-09838-3.

[7] Subhajit Pal et al. "Big data in biology: The hope and present-day challenges in it". In: *Gene Reports* 21 (Dec. 2020), p. 100869. ISSN: 2452-0144. DOI: 10.1016/j.genrep.2020.100869.

[8] Felix Teufel et al. "SignalP 6.0 predicts all five types of signal peptides using protein language models". In: *Nat. Biotechnol.* 40 (July 2022), pp. 1023–1025. ISSN: 1546-1696. DOI: 10.1038/s41587-021-01156-3. URL: https://www.nature.com/articles/s41587-021-01156-3.

[9] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv* (June 2017). DOI: 10.48550/arXiv.1706.03762. eprint: 1706.03762.

[10] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL].