Bioinformatics, 2023, **39**, i347–i356
https://doi.org/10.1093/bioinformatics/btad228
ISMB/ECCB 2023

TSignal: a transformer model for signal peptide prediction

Alexandru Dumitrescu [1,2,*], Emmi Jokinen[1], Anja Paatero[2], Juho Kellosalo[2], Ville O. Paavilainen[2], Harri Lähdesmäki[1,*]

[1]Department of Computer Science, Aalto University, Espoo 02150, Finland
[2]Institute of Biotechnology, HiLIFE, University of Helsinki, Helsinki 00014, Finland

*Corresponding author. Institute of Biotechnology, HiLIFE, University of Helsinki, Helsinki 00014, Finland. E-mail: alexandru.dumitrescu@aalto.fi (A.D.); Department of Computer Science, Aalto University, Espoo 02150, Finland. E-mail: harri.lahdesmaki@aalto.fi (H.L.)

Abstract

Motivation: Signal peptides (SPs) are short amino acid segments present at the N-terminus of newly synthesized proteins that facilitate protein translocation into the lumen of the endoplasmic reticulum, after which they are cleaved off. Specific regions of SPs influence the efficiency of protein translocation, and small changes in their primary structure can abolish protein secretion altogether. The lack of conserved motifs across SPs, sensitivity to mutations, and variability in the length of the peptides make SP prediction a challenging task that has been extensively pursued over the years.

Results: We introduce TSignal, a deep transformer-based neural network architecture that utilizes BERT language models and dot-product attention techniques. TSignal predicts the presence of SPs and the cleavage site between the SP and the translocated mature protein. We use common benchmark datasets and show competitive accuracy in terms of SP presence prediction and state-of-the-art accuracy in terms of cleavage site prediction for most of the SP types and organism groups. We further illustrate that our fully data-driven trained model identifies useful biological information on heterogeneous test sequences.

Availability and implementation: TSignal is available at: https://github.com/Dumitrescu-Alexandru/TSignal.

1 Introduction

Signal peptides (SPs) are short amino acid chains found at the N-terminus of newly synthesized proteins. Their role is to facilitate the translocation of proteins, after which they are cleaved off from the mature protein by signal peptidases (SPases). SPs may direct proteins to the secretory (Sec) pathway (in all organisms) or twin-arginine translocation (TAT) pathway, which is found only in prokaryotes and in plant chloroplasts. Proteins enter the Sec pathway in an unfolded state, while those going through the TAT pathway fold before the translocation.

Almost all SPs overall contain a tripartite structure with, generally positively charged, N-region, H-region (hydrophobic region), and a cleavage site-containing C-terminal region. In SPs cleaved by SPase I the cleavage site is preceded by a generally polar C-region, while SPs cleaved by SPase II have a three residue lipobox instead of the C region (Owji et al. 2018) (see Fig. 1) and also a cysteine residue after the cleavage site (Tokunaga et al. 1982). SPs processed by SPase IV do not have a tripartite structure, but instead contain a translocation-mediating basic region (BR). Each of the aforementioned SP regions, which can vary in length and residue composition, dynamically interact with various components of the Sec or TAT machinery in order to facilitate protein translocation (Owji et al. 2018). While SPs have recognizable regions, they lack clear consensus motifs. Consequently, the exact sequence properties of functional SPs have not been determined. This makes SP prediction challenging, which is evident in the problems of identifying translocation-abolishing point mutations (Rajpar et al. 2002; Liu et al. 2012).

Many of the previous machine learning approaches for SP detection and cleavage site prediction rely on different types of hidden Markov models (HMMs; Käll et al. 2004; Reynolds et al. 2008; Viklund et al. 2008; Tsirigos et al. 2015). Deep learning approaches have also been employed for the feature representations of the sequence residues. However, the final prediction is still carried out by structured prediction algorithms, using the deep residue representations as the inputs for conditional random fields (CRFs; Savojardo et al. 2018; Zhang et al. 2020; Teufel et al. 2022). Non-machine-learning methods have also been employed. Homology-based search algorithms are used to detect the presence of SPs and report putative cleavage sites by aligning the queries to annotated sequences (Frank and Sippl 2008; Wishart et al. 2008). Despite several approaches to design and optimize SP prediction methods, no single approach provides robust SP identification. This difficulty is highlighted by the fact that protein database Uniprot relies on four separate SP prediction programs for SP assignment (UniProt Consortium 2019). An important aspect is that both HMMs and CRFs rely on a matrix of transition probabilities between consecutive amino acids. Using this matrix, prior information about the structure of SPs can be hard-coded into a model by constraining the transitions known to be biologically impossible to have zero probability. Although HMMs and CRFs can also work without hard-coded constraints, to our knowledge, all SP prediction architectures found in the literature have incorporated such constraints. As a concrete example, state transition matrices can be restricted to produce only contiguous SP predictions and ensure they always start at the N-terminus of the sequence (Owji et al. 2018), while cleavage sites for SPase II cleaved proteins can additionally be constrained to always be

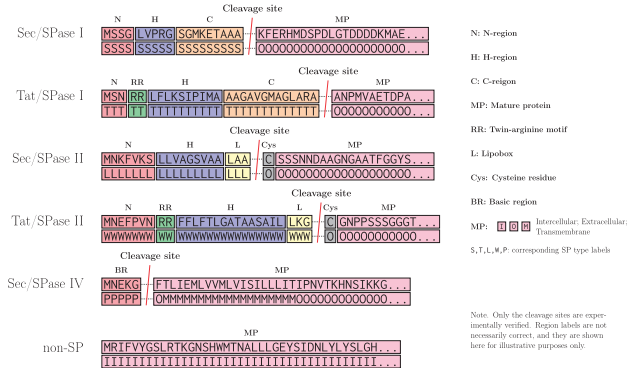

Figure 1. Examples of input sequences and associated labels for TSignal. Secretory path-directed SPs cleaved by SPaseI have all three of the N, H, and C subregions. TAT-directed proteins present the twin-arginine (RR) motif in their SPs and the SP also has its N and H regions delimited by the RR motif. SPase II cleaved proteins do not present a proper C region but instead have three amino acids belonging to a region called lipobox (L), which is always followed by a Cys residue. SPase IV cleaved SPs only have a small BR at the N-terminus. Amino acid sequence of an intracellular protein that does not contain an SP is included as a reference.

followed by a cysteine residue (Tokunaga et al. 1982). While incorporating such inductive bias is generally found useful for classical HMM and CRF models, we do not enforce any such prior information. Here, we developed a new data-driven prediction method, TSignal, that uses transformer-based architectures both for the residue representation, as well as the prediction network. We utilize rich representations of amino acid sequences obtained from a BERT language model (LM), and further train the BERT model together with self-attentive prediction methods.

Our approach aims to solve the SP type and CS prediction tasks using a neural machine translation setting, instead of the typically employed sequence tagging approaches. We note that translation is a generalization of tagging, since tagging is, in fact, the translation of sequences with identical input and output sequence lengths, where inputs and outputs have a one-to-one correspondence. The main advantage of our setting is that labels at all positions in a sequence are predicted based on all input sequence embeddings from the encoder, thanks to the inner workings of the transformer encoder-decoder architecture. On the other hand, a classical tagging setting like the one developed in SignalP version 6.0 (Teufel et al. 2022) has the advantage of a clear one-to-one mapping from input token a_i to its corresponding output label y_i, since the model only ever receives the embedding corresponding to position i when predicting y_i. We show that concatenating the same positional encodings to both the encoder's outputs and the decoder's inputs of a transformer-based model allows it to easily identify such mappings, and this plays a crucial role in our improved CS prediction performance (please refer to Supplementary Section 3 for more details). Lastly, although we have observed this to be highly unlikely in practice, formulating the problem as a translation setting implies that label predictions may have different lengths than the input. Nevertheless, solving this issue can trivially be achieved by trimming the sequences that are too long and by never predicting an "end of sequence" tag (i.e. setting its probability to 0), such that output label sequences have at least the sequence length of the input. Using data from large databases of known SPs we demonstrate that our model achieves state-of-the-art performance compared to previous best approaches, including SignalP version 6.0 (Teufel et al. 2022).

2 Materials and methods

A protein is defined by its amino acid sequence $\mathbf{a} = (a_1 a_2 \dots a_n)$ together with an associated label for each amino acid residue $\mathbf{y} = (y_1 y_2 \dots y_n)$. We consider the following eight labels, $y_i \in \mathcal{Y} = \{\text{Sec/SPase I, Sec/SPase II, Sec/SPase IV, TAT/SPase I, TAT/SPase II, intracellular, transmembrane (TM), extracellular}\}$. The model distinguishes between five different SP types: secretory pathway directed peptides cleaved by SPase I, II, and IV, and TAT pathway-directed peptides cleaved by SPase I and II. Our model solves the cleavage site prediction task by predicting sequences of labels. The model predicts residue a_i to be part of an SP if its predicted label $\hat{y}_i$ corresponds to one of the five SP types we train for. The predicted SP type is inferred from the label $\hat{y}_1$ associated with residue a_1, while the cleavage site is determined by the first residue a_c that is predicted to have one of the three non-SP labels $\hat{y}_c \in \{\text{intracellular, TM, extracellular}\}$ following a sequence of predicted SP labels. This denotes that the cleavage site is located between residues a_{c-1} and a_c. Additionally, each sequence $\mathbf{a}$ originates from one of the four organism groups $\mathbf{g} \in \{\text{eukarya}, \text{gn}-\text{bacteria}, \text{gp}-\text{bacteria}, \text{archaea}\}$, corresponding to eukaryotes, gram-negative bacteria, gram-positive bacteria, and archaea. Thus, each data item can be represented by a triplet $(\mathbf{a}, \mathbf{y}, \mathbf{g})$.

The structures of the SP types predicted by TSignal are shown in Fig. 1. Although we do not explicitly utilize this information, we show that the model can intrinsically learn this type of structural information and generalize on diverse protein sequences.

2.1 Transformer models

Contrary to previous approaches that use HMMs or CRFs, our transformer model does not have any hard-coded knowledge of SP structures. Instead, TSignal builds on the transformer model architecture which was initially developed for sequence translation tasks in the natural language field (Vaswani et al. 2017). We use a contextual protein embedding model trained on 216 million protein sequences called ProtBERT (Elnaggar et al. 2021). ProtBERT is a character-level adaptation of the auto-encoder BERT LM that is trained only on the masked-token prediction task. We use this model to retrieve the 1024 dimensional representations of all residues in each amino acid sequence, resulting in an $\mathbb{R}^{1024 \times N}$ representation for each sequence, with N being the maximum sequence length. To further adjust the ProtBERT model for SP sequences, we integrate the BERT model as part of our model and train it together with the sequence prediction model, using a similar approach to SignalP version 6.0. The $1024 \times N$ sequence representations retrieved by ProtBERT are used as keys and values for the transformer decoder, along with the label queries in the multi-head attention blocks of the decoder. This way, we effectively train a sequence-to-sequence transformer architecture, as described by Vaswani et al. (2017). TSignal model architecture is shown in Fig. 2 and described in more detail below.

2.1.1 Encoder model

Transformer encoders use initial embedding layers to map tokens from one-hot vectors to dense representations. The

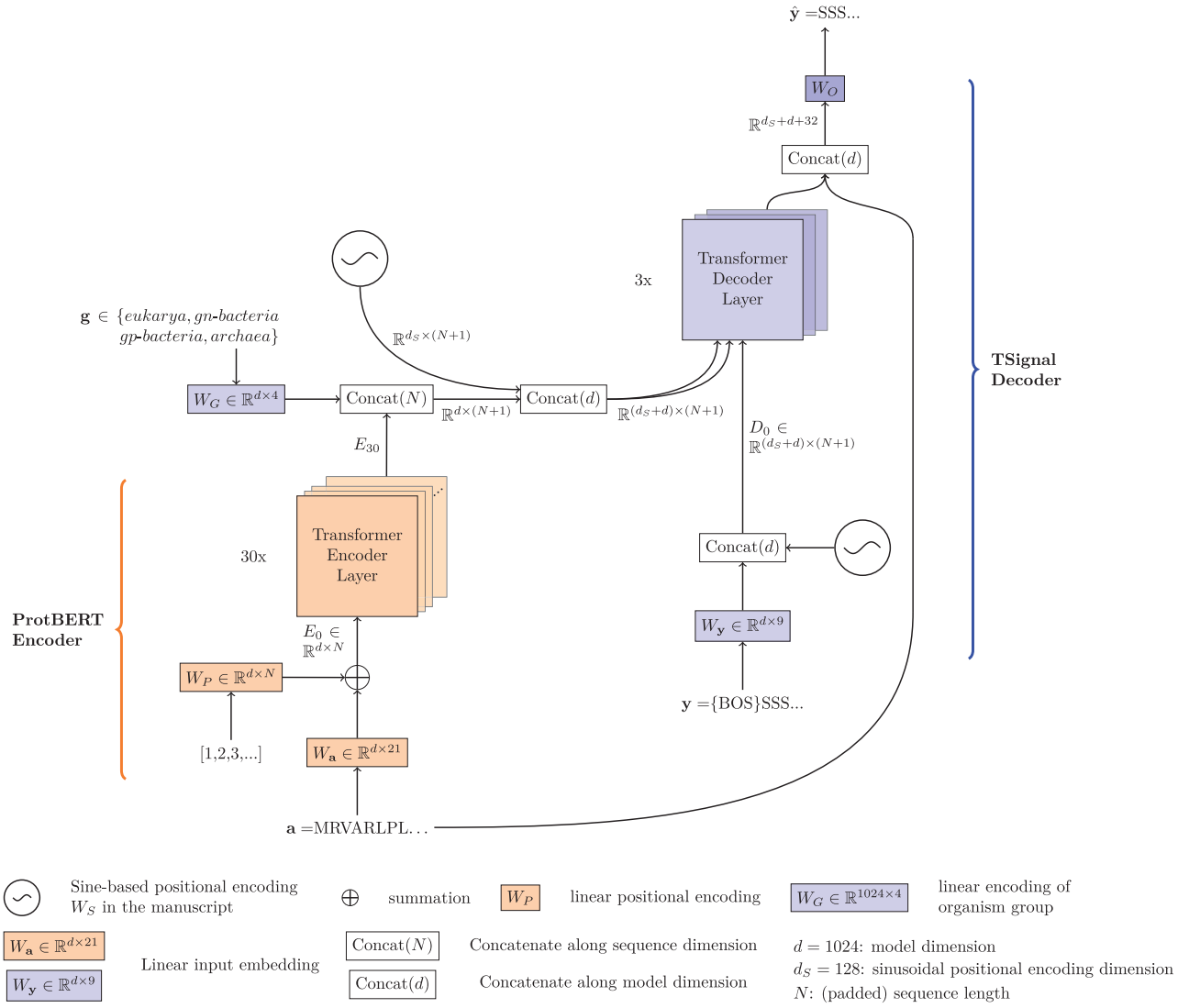

Figure 2. TSignal architecture consists of ProtBERT encoder and multi-head attention-based transformer decoder. Initial embedding involves a dense representation and a linear positional encoding. ProtBERT embeddings are concatenated with organism group and positional representations prior to using them as key and value vectors in the decoder, together with query vector from dense representations of the sequence labels. See Section 2.1 for an in-depth description of the TSignal model.

positional encoding and multi-head attention mechanism are then able to extract contextual vector representations of these input tokens.

Initial embedding: Each amino acid a_i of a sequence $\mathbf{a} = (a_1 a_2 \dots a_n)$ is initially one-hot encoded into 21-dimensional standard unit column vector $\mathbf{c}^{(i)}$ (from 20 unique amino acids and one additional padding token). For the whole sequence $\mathbf{a}$ this results in a binary one-hot encoded matrix $A = (\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(n)})$ of size 21-by-n. If n is smaller than the maximum sequence length N in a mini-batch, then A is padded with N—n one-hot vectors corresponding to an additional padding token, forming a matrix of size 21-by-N. The initial embedding involves a linear transformation that maps each one-hot encoded amino acid to a dense d-dimensional (we use $d = 1024$ but we use symbol d in method description for clarity) representation using a matrix $W_\mathbf{a} \in \mathbb{R}^{d\times 21}$. Collectively for the whole sequence $\mathbf{a}$ this can be written as a matrix multiplication $I = W_\mathbf{a} A \in \mathbb{R}^{d\times N}$.

Positional encoding: The encoder model uses a linear positional encoding (this linear transformation is a very large input embedding layer that can encode positions in much larger sequences, but we preserve linear algebra notation for consistency). Since the amino acid indices will always be ordered from 1 to N (we assume the left-most amino acid is always the first N-terminus residue), we can directly define our positional representation for the amino acids as $W_P \in \mathbb{R}^{d\times N}$. With the above definitions, the initial positionally encoded embedding matrix for a sequence is:

$$E_0 = I + W_P = W_\mathbf{a} A + W_P. \tag{1}$$

Transformer encoder: Representation $E_0^T = (\mathbf{e}_{0,1}, \dots, \mathbf{e}_{0,N})^T \in \mathbb{R}^{N\times d}$ from Equation (1) is then passed to multiple transformer block layers, where $\mathbf{e}_{0,i}$ is the d-dimensional initial representation of amino acid residue at position i. The core idea of transformer blocks is to process sequential information using only attention mechanisms, without any recurrent neural networks. In particular, transformers use dot-product attention. In one attention head h of layer l, attention weights $w_{ij}^{l,h}$ for a query residue i and key residue j are computed using the d-dimensional residue representations from the previous layers $\mathbf{e}_{l-1,i}$ and $\mathbf{e}_{l-1,j}$. This is done by first

mapping the two vectors $\mathbf{e}_{l-1,i}$ and $\mathbf{e}_{l-1,j}$ into a dot-product suitable space and then computing their dot product

$$w_{ij}^{l,h} = (\mathbf{e}_{l-1,i}^T W_{Q,l,h}) \cdot (\mathbf{e}_{l-1,j}^T W_{K,l,h}) \in \mathbb{R}, \tag{2}$$

where $\cdot$ denotes the vector dot-product and $W_{Q,l,h}, W_{K,l,h} \in \mathbb{R}^{d \times d/H}$ are the query and key linear layers of attention head h at layer l, and H is the total number of heads per layer (model dimension d is usually chosen to be exactly divisible by H). For a given query residue i, the attention weights $w_{ij}^{l,h}, j \in \{1, 2, \ldots, N\}$ are normalized with softmax transformation, denoted as $\tilde{w}_{ij}^{l,h}$. These attention weights are then used to compute a weighted average of the vectors formed by a third linear mapping (called value matrix) $W_{V,h,l} \in \mathbb{R}^{d \times d/H}$ of the intermediate vectors that map the previous layer representations $\mathbf{e}_{l-1,j}$ into value vectors

$$\mathbf{e}'_{l,h,i} = \sum_{j=1}^{L} \tilde{w}_{i,j}^{l,h} (\mathbf{e}_{l-1,j}^T W_{V,l,h}). \tag{3}$$

Multiple such attention "heads" are used in each layer, allowing the model to attend, at each step, to various parts of the sequences using different pairs of linear mappings $(W_{Q,l,h}, W_{K,l,h})$, where $h \in \{1, \ldots, H\}$ and $l \in \{1, \ldots, L\}$. Let's define the key, query, and value matrices computed by an attention head at layer l as

$$\begin{aligned} K_{l,h} &= E_{l-1}^T W_{K,l,h} \\ Q_{l,h} &= E_{l-1}^T W_{Q,l,h} \\ V_{l,h} &= E_{l-1}^T W_{V,l,h}. \end{aligned} \tag{4}$$

Note that the product $Q_{l,h} K_{l,h}^T$ gives all the unnormalized weights $w_{ij}^{l,j}$ from Equation (2). The weighted average from Equation (3) for head h can be written compactly, for the whole sequence as

$$E'_{l,h} = \text{Attention}(Q_{l,h}, K_{l,h}, V_{l,h}) = \text{Softmax}\left(\frac{Q_{l,h} K_{l,h}^T}{\sqrt{d/H}}\right) V_{l,h}, \tag{5}$$

and the resulting sequence representations $E'_{l,h} \in \mathbb{R}^{N \times d/H}$ from all heads $h \in \{1, \ldots, H\}$ are stacked $(E'_{l,1}, \ldots, E'_{l,H}) \in \mathbb{R}^{N \times d}$, and then multiplied with an output matrix $W_{O,l} \in \mathbb{R}^{d \times d}$ to obtain the intermediate matrix representation $E'_l \in \mathbb{R}^{N \times d}$.

Finally, the intermediate representation E'_l given by the multi-head attention at layer l is passed to a two-layered feed-forward network of the form $F(E'_l) = \text{ReLU}(E'_l W_1) W_2$. Usually, W_1 is chosen to expand the model's dimension from d to e_d, $W_1 \in \mathbb{R}^{d \times e_d}$, with e_d being some expanding dimension, $e_d > d$, and $W_2 \in \mathbb{R}^{e_d \times d}$ maps the vectors back to dimension d (the network applies this transformation to each residue, individually). Skip connection and layer normalization layers are also added from E_{l-1} to E'_l and from E'_l to the output of the feed-forward network block, giving the layer's output representations E_l. The whole process is repeated for each layer of the network. For notational simplicity, we omitted the bias terms in our notations, but all linear operators except the initial embedding, positional encoding, and the linear transformations in the multi-head attention block use a bias term. We refer to Vaswani et al. (2017) for further details of attention mechanisms and to Elnaggar et al. (2021) for details of the ProtBERT model.

2.1.2 Decoder model

Transformer decoders use similar input embedding and positional encoding layers as the encoder, but their inputs are now the labels.

Initial Embedding: In the first step, decoder processes the output labels $\mathbf{y} = (y_1, \ldots, y_n)$ similarly as the encoder processes the residues in the sequence $\mathbf{a} = (a_1, \ldots, a_n)$. An important difference is that we append $y_0 = \{\text{BOS}\}$ (beginning of sequence) token, which will be used by the model to predict the first label y_1. The matrix $Y \in \mathbb{R}^{9 \times (N+1)}$ consisting of the one-hot label representations as columns is mapped to a dense representation using $I_D = W_\mathbf{y} Y \in \mathbb{R}^{d \times (N+1)}$, where $W_\mathbf{y} \in \mathbb{R}^{d \times 9}$ is the initial label embedding layer for all nine unique labels (eight real labels and {BOS}).

Positional encoding: We use a different type of positional encoding for the decoder part of our model. One alternative to modeling positional vectors using a linear layer, as in Section 2.1.1, is to use a sinusoidal function. The resulting fixed matrix $W_S \in \mathbb{R}^{d_S \times (N+1)}$ (for a sequence of length N with an additional element $y_0 = \{\text{BOS}\}$) is defined element-wise as:

$$\begin{aligned} W_S(2k+1, i) &= \sin(i/10000^{2k/d_S}) \\ W_S(2k+2, i) &= \cos(i/10000^{2k/d_S}), \end{aligned} \tag{6}$$

where $2k+1$ and $2k+2$ refer to the odd and even dimensions in our d_S-dimensional positional encoding vector (here, $d_S \neq d$) with $k \in \{0, \ldots, d_S/2 - 1\}$ and $i \in \{1, \ldots, N+1\}$ is the sequence residue index.

For the decoder, the fixed d_S-dimensional positional information W_S is concatenated with the d-dimensional vector representations of the labels I_D. The initial label sequence representation matrix is given by $D_0 = I_D \oplus W_S \in \mathbb{R}^{(d+d_S) \times (N+1)}$, where $\oplus$ denotes the concatenation operator and the sequence length is increased by one to $N+1$ because of the additional y_0 token. We choose this sinusoidal positional encoding for the decoder as it does not need any further training. Because of the limited amount of data, we hypothesize that another linear positional encoding would be difficult to converge well (note that the linear positional encoding in the encoder is already pre-trained on large amounts of data).

We concatenate an additional d-dimensional vector $\mathbf{g} \in \mathbb{R}^{d \times 1}$ representing the organism group of the input sequence (obtained using a linear layer $W_G \in \mathbb{R}^{d \times 4}$) to the encoder's final layer residue representations E_L. The result is also concatenated with the same sinusoidal positional encoding W_S, in order to have the same type of positional information in the label and residue representations, forming $E = (E_L \oplus \mathbf{g}) \oplus W_S \in \mathbb{R}^{(d+d_S) \times (N+1)}$, where the first concatenation is along the sequence dimension N for $E_L \in \mathbb{R}^{d \times N}$ and $\mathbf{g}^T \in \mathbb{R}^{d \times 1}$ and the second one along the model's dimension d (see Supplementary Section 3 for the predictive performance effect of the additional positional encoding used on the encoder's outputs).

Transformer decoder: Compared to the encoder layers, the decoder contains both a self-attention module, as well as an additional cross-attention layer which "looks" at the input vectors from the encoder. The first step of a decoder layer is a self-multi-head attention, similar to the transformer encoder.

Label representations are contextualized using the matrix from the previous layer D_{l-1}, as described in Equations (4)–(5) (with E_l replaced by D_l). We denote this intermediate matrix of the decoder as $D'_l \in \mathbb{R}^{(d+d_S)\times(N+1)}$. The label vector representations of D'_l are then used as queries in the second multi-head attention layer, together with the representations from the encoder $E \in \mathbb{R}^{(d+d_S)\times(N+1)}$. Concretely, similarly as in Equation (4), values from D'_l mapped into queries are attending to key and value mappings from $E \in \mathbb{R}^{(d+d_S)\times(N+1)}$ to yield an additional intermediate representation D''_l. Lastly, a two-layered feed-forward network retrieves the next layer's representations of the decoder $D_l \in \mathbb{R}^{(d+d_S)\times(N+1)}$, like the ones in the encoder layers.

During training, the self-attention module in the decoder layers computes all contextual values at once (all pairs (y_i, y_j) are considered) and therefore vectors representing y_i have access to $y_j, j \geq i$. This induces undesired behavior, as we wish to extract the amino acid labels based only on the previous labels. This issue is addressed by adding an additional {BOS} (beginning of sequence) token at the start of the label sequences and using appropriate masking. The mask is a matrix filled with zeros and $-\infty$ above the diagonal and it can be added to the unnormalized attention weights $Q_{l,h}K_{l,h}^T$ (see Equation (5)). This ensures that the attention weights of current and future labels will be 0, and y_1 will be predicted based only on the input retrieved by the encoder (as {BOS} does not contain any label information). Padding also uses a similar masking approach, where the attention to {PAD} tokens are zeroed.

The type of SP can only be correctly predicted when considering all residues forming an SP, and this requires the model to capture long range context. In contrast, the exact cleavage site prediction can be hindered by the fact that close residues have similar representations because of the context, and therefore we concatenate a one-hot representation of residue k for the associated label prediction $\hat{y}_k$. We hypothesize that this allows the model more freedom in terms of the optimal amount of context it can add in its representations, and we observed consistent improvements when using this addition, which supports this claim.

Label predictions y_k are finally computed based on the encoder's last layer representations $(\mathbf{e}_{L,1}, \mathbf{e}_{L,2}, \ldots, \mathbf{e}_{L,N})$, the previously predicted labels during inference (and previous true labels during training) $(y_0, \hat{y}_1, \ldots, \hat{y}_{k-1})$, as well as an additional one-hot encoded residue $\mathbf{c}^{(k)}$ at the position where label k is being predicted. During inference, the model predicts $\hat{y}_k$ sequentially based on its own generated label sequence $(y_0, \hat{y}_1, \ldots, \hat{y}_{k-1})$ as well as the encoder representations

$$p(y_k) = p(y_k|\hat{y}_{k-1}, \hat{y}_{k-2}, \ldots, \hat{y}_1, y_0, \mathbf{e}_{L,1}, \ldots, \mathbf{e}_{L,N}, \mathbf{c}^{(k)}), \quad (7)$$

and $\hat{y}_1$ will be predicted based the special token $y_0 = \{\text{BOS}\}$, that we also use during training. Concretely, $p(y_k) = \text{Softmax}\left((D_{3,k} \oplus \mathbf{c}_k)^T W_O\right)$, where $D_{3,k} \in \mathbb{R}^{d+d_S \times 1}$ is the decoder's last layer representation of the previous residue label (since the outputs are shifted to the right by one position due to the {BOS} token), $\mathbf{c}_k \in \mathbb{R}^{21\times 1}$ is the one-hot representation of the k^{th} amino acid and $W_O \in \mathbb{R}^{d+d_S+21\times 8}$ is a linear layer.

2.2 Architecture details

We use a dropout of 0.1 on all transformer decoder weights. The position-wise feed-forward network of our decoder has an almost fourfold expanding dimension, from the original 1152 to 4096 ($d + d_S = 1152$, from the original representation $d = 1024$ and the concatenated sinusoidal positional information $d_S = 128$). The ProtBERT encoder and transformer decoder have 30 and 3 layers of 16 attention heads, respectively. We initialize the decoder parameters with the Xavier Uniform initializer described by Glorot and Bengio (2010).

For better generalization, we used stochastic weight averaging which helps avoid sharp local minima solutions (Izmailov et al. 2018). We chose constant learning rates of 10^{-5} and 10^{-4} for ProtBERT and decoder parameters, respectively, ensuring as much exploration of the local minima as possible, without risking divergence. SWA weights are updated after each training epoch. For further details and insights on the effect of SWA we refer to Supplementary Section 4.

We also experimented with three other model variants that utilize the ProtBERT model in different ways. Based on the F1 scores from a CS prediction comparison, we chose the model setup presented in Section 2.1. See Supplementary Section 7 for details.

A crucial aspect of TSignal's high performance was tuning the LM that computes amino acid embeddings, ProtBERT. Predicting a specific position where a CS occurs should evidently account for the exact location of residues in a SP. However, this information can become blurred due to the heavy contextualization of ProtBERT. Therefore, further tuning the LM significantly increases performance, as shown in the Results Section 3.3.

2.3 Dataset

We use the same dataset $\mathcal{D}$ as Teufel et al. (2022), which contains sequences from Uniprot (UniProt Consortium 2019) and Prosite (Sigrist et al. 2013) for proteins containing SPs as well as UniProt and TOPDB (Dobson et al. 2015) for soluble and TM proteins, where only the expert-reviewed sequences are considered. The dataset contains 19 174 protein sequences grouped into four organism groups: eukaryotes (1995 SPs and 14 095 non-SPs), gram-negative bacteria (1274 SPs and 898 non-SPs), gram-positive bacteria (496 SPs and 223 non-SPs), and archaea (84 SPs and 109 non-SPs). Every residue in each protein sequence has an annotated label that tells whether the residue belongs to the mature protein or the SP which will be cleaved, as well as the type of SP ($y_i \in \mathcal{Y}$). We use the same three-fold homology-based partitioning $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$, with the exception that we further split each partition into $\mathcal{D}_i = (\mathcal{D}_{i,\text{train}}, \mathcal{D}_{i,\text{test}})$, where $\mathcal{D}_{i,\text{train}}$ have 90% of the original $\mathcal{D}_i$ and $\mathcal{D}_{i,\text{test}}$ the remaining 10%. We then train the model on $\mathcal{D}_{i,\text{train}}, \mathcal{D}_{j,\text{train}}$, validate on $\mathcal{D}_{i,\text{test}}, \mathcal{D}_{j,\text{test}}$ and then test on $\mathcal{D}_k = (\mathcal{D}_{k,\text{train}}, \mathcal{D}_{k,\text{test}})$. We, therefore, train and validate on different homology partitions than the test partition, ensuring a fair comparison against SignalP version 6.0.

We do not use all the sequences in $\mathcal{D}$ for the benchmark comparisons. Instead, we compare the predictive performance of TSignal and all other models on a benchmark subset $\mathcal{D}_B \subset (\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3)$. $\mathcal{D}_B$ was created in SignalP version 5 (Almagro Armenteros et al. 2019) such that sequences in $\mathcal{D}_B$ have at most 25% sequence identity to the training dataset used by DeepSig (Savojardo et al. 2018), and therefore all comparisons between TSignal, DeepSig, and SignalP version 6.0 are fair. Moreover, we point out that SignalP version 6.0 method has a similar complexity, as it also tunes the 30-layered transformer encoder ProtBERT. As such, the two methods give an objective comparison between structured

and unstructured computational methods' performance on the SP prediction tasks.

2.4 Evaluation

For cleavage site prediction, we use precision and recall as our main performance evaluation metrics. The precision and recall are computed for each SP type and organism group individually. We report the CS prediction scores at various tolerance levels as this should give a more detailed performance evaluation of the models (e.g. it allows us to distinguish between models predicting a CS within a distance of 1 and 2 from the true CS). Additionally, CS positions can have erroneous or uncertain annotations (Almagro Armenteros et al. 2019). If this happens, we assume it would be likely for the true, biological cleavage sites to be within a small distance of these annotations. We consider a cleavage site prediction to be correct if $i_p \in [i_c - \text{tol}, i_c + \text{tol}]$, where i_p is the predicted index of the cleavage site, i_c is the true (annotated) index and the tolerance $\text{tol} \in \{0, 1, 2, 3\}$. Additionally, a cleavage site is only considered correct if the predicted SP type is correct. For example, in the case where an SP exists in a sequence but the predicted SP type is not correct, the CS prediction is accounted as both a false positive (for the SP type which is wrongly predicted) and a false negative (for the true SP type). To have a single metric for the model's performance, we use F1 score defined as $\text{F1} = \frac{2 \cdot \text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$. To summarize the results across all SP types and organism groups, we report the average F1 score and weighted F1 score (weighted by the number of data points in each group).

To assess the SP presence prediction performance, we use Matthew's correlation coefficient (MCC). We compute two separate metrics, MCC1 which considers only soluble and TM proteins as negative samples, and MCC2 which also counts other SP types as negative samples.

3 Results

In Sections 3.1 and 3.2, we report benchmark comparisons on Sec/SPase I and II and TAT/SPase I sequences (because only these SP types have numeric results reported in Teufel et al. (2022) from $\mathcal{D}_B$ described in Section 2.3. This allows us to directly compare our results to the results of the previous state-of-the-art method, SignalP version 6.0. The CS-F1 performance on the whole data $\mathcal{D}$ is reported in Supplementary Section 8, where we also report the Sec/SPase IV and TAT/SPase II performance.

In addition to SignalP version 6.0 comparisons, we also report the performance of a few other popular HMM-based SP prediction methods: DeepSig (Savojardo et al. 2018), PRED-TAT (Bagos et al. 2010), LipoP (Juncker et al. 2003), and Phobius (Käll et al. 2004) on the $\mathcal{D}_B$ data points. These additional results are obtained using their corresponding web servers. Since their training datasets likely contain similar sequences to those found in $\mathcal{D}_B$, these results may be overestimates when compared to TSignal, SignalP versions 6.0, and DeepSig (please refer to the corresponding publications for further details on the other methods' datasets).

3.1 SP prediction comparisons

We first assess the SP prediction performance on the sequences found in $\mathcal{D}_B$ using the MCC metric. We evaluate MCC1 for TSignal, SignalP version 6.0 and a few other popular models for SP prediction: (Savojardo et al. 2018), PRED-TAT (Bagos et al. 2010), LipoP (Juncker et al. 2003), and Phobius (Käll et al. 2004). For the task of separating various types of SPs (MCC2), we only compare TSignal to SignalP version 6.0 (because other models were not trained to distinguish all SP types considered in this work). Figure 3 shows the MCC values (we report the numeric values achieved by TSignal and SignalP version 6.0 in Supplementary Table S4). We observe small but consistent improvements on Sec and TAT SPase I, and a slight decrease in Sec/SPase II SP type prediction performance compared to most of the previous approaches. As we show later, the model learns to detect the RR motif for TAT predictions, so the increased TAT/SPase I performance suggests our model finds causal features, important for good generalization. The weighted MCC1 and MCC2 scores across all organism groups and SP types for TSignal are 0.8520 ± 0.016 and 0.8312 ± 0.013, respectively, while SignalP version 6.0 has 0.8532 and 0.8263. We, therefore, note similar, or even a slight improvement, on the SP type prediction accuracy compared to the previous state-of-the-art method.

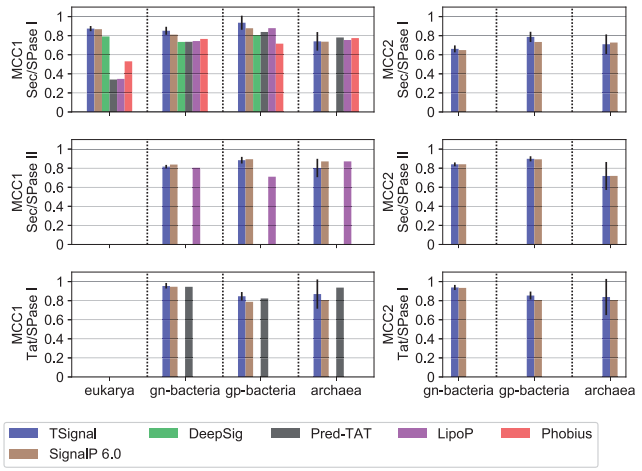

Figure 3. A comparison of SP predictions for TSignal and other models on the benchmark dataset $\mathcal{D}_B$ using the average MCC metrics: MCC1 on left and, MCC2 on right. The height of the bar plots represent the mean MCC result across five different runs, and the approximated 95% confidence interval is shown by the black vertical lines plotted on top of the bars. Organism groups are shown on the *x*-axis, and the SP-type i.e. tested on the *y*-axis. The benchmark dataset $\mathcal{D}_B$ is homology partitioned only between the train and test set of TSignal, SignalP version 6.0, and DeepSig, and results for other models are likely overestimates. Missing bars in the plots correspond to the respective model not being trained on that particular organism group or SP type.

To further test TSignal's ability to recognize difficult-to-predict SPs, we assessed its capability in identifying four SP-containing sequences that were earlier identified in a separate study that will be published separately (J.Kellosalo & V. Paavilainen, Personal communication; see Supplementary Section 2 and Supplementary Fig. S1). These four sequences were identified in a screen for functional SPs that mediate protein secretion in mammalian cells and were not recognized to contain an SP by existing prediction methods. Distinctively, all of these sequences contain basic amino acids dispersed throughout the SP sequence. Basic residues are typically contained at the N terminus of SPs, yet these previously unidentified sequences are sufficient to facilitate protein secretion in human cells (Supplementary Fig. S2). We postulate that in this case imposing specific transitions through structured prediction models may hinder the prediction of unusual sequences. Here, we tested the same models we compare against for

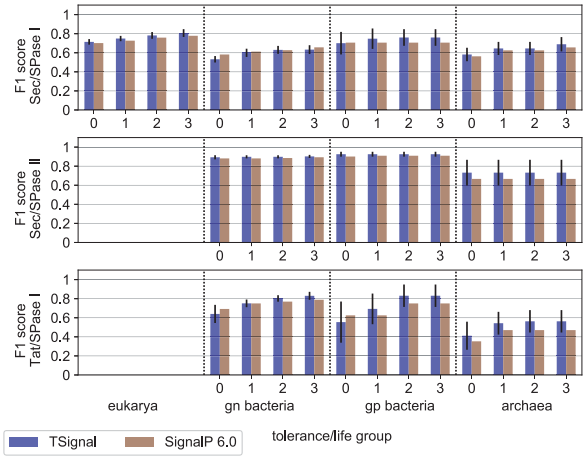

Figure 4. A comparison of CS predictions between TSignal and SignalP version 6.0 on the benchmark dataset $\mathcal{D}_B$ using the average F1 score. The height of the bar plots represents the mean F1 result across 5 different runs, and the approximated 95% confidence interval is shown by the black vertical lines plotted on top of the bars. Organism groups and tolerance levels are shown on the *x*-axis, and the SP-type i.e. tested on the *y*-axis. SignalP version 6.0 results were computed using the precision and recall scores reported in their manuscript.

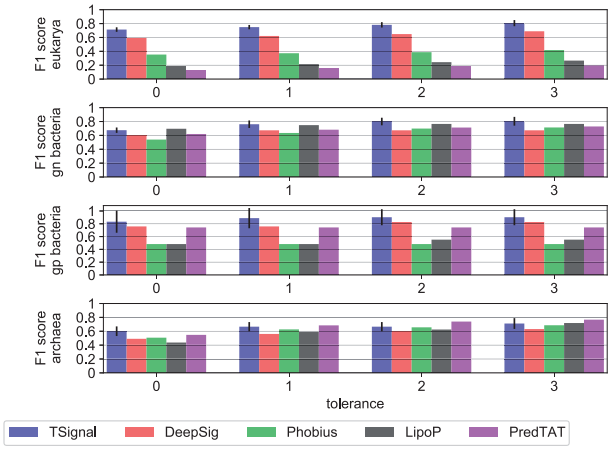

Figure 5. A comparison of CS predictions between TSignal and other popular models. We use the publicly available website tools for each of the tested models. The height of the bar plots represents the mean F1 result across five different runs, and the approximated 95% confidence interval is shown by the black vertical lines plotted on top of the bars. Tolerance levels and organism groups are shown on the *x*-axis and *y*-axis, respectively. The results were computed using only Sec/SPI sequences from the benchmark dataset $\mathcal{D}_B$, since the other models were not trained for all SP types considered here.

the SP type and CS predictive performance using their publicly available webservers: DeepSig, PRED-TAT, LipoP, Phobius, and SignalP version 6.0. DeepSig classifies three of these sequences as containing a TM domain and Phobius reports a TM domain in all four sequences, but none of the methods classifies any of those four sequences as containing an SP. In contrast, TSignal predicts a TM domain in two of the four sequences and correctly determines an SP in the other two. The four sequences and their TSignal predictions are reported in Supplementary Fig. S2.

3.2 Cleavage site prediction comparison

We now test the CS prediction accuracy using the CS-F1 score. We only compare against SignalP version 6.0 as the other methods do not predict all three SP types considered. Figure 4 shows that our model compares favorably to SignalP version 6.0 for most organism groups and SP types on $\mathcal{D}_B$ (we report the numeric values of the F1 score, precision, and recall in Supplementary Tables S1, S2, and S3). Particularly interesting is that the cleavage site of Sec/SPase II SPs is more accurately predicted by our fully data-driven model, although the presence of Cys residues restricts the number of possible cleavage sites, and should help structured prediction models. We additionally note that TAT CS predictions are also better, even though the CRF model SignalP version 6.0 is explicitly trained to detect the twin-arginine motif. In terms of overall performance, TSignal outperforms SignalP version 6.0 on the majority of SP types and organism groups with a weighted F1 score of 0.8127 ± 0.005 compared to SignalP version 6.0 0.7976 (with weights given by the occurrence frequency of each SP type and organism group in the test folds). The final overall CS prediction performance is therefore about three standard deviations higher.

We also compare TSignal to PRED-TAT, LipoP, Phobius, and DeepSig on Sec/SPase I and no-SP sequences from $\mathcal{D}_B$. We only use Sec/SPase I sequences since all these models have been trained (at least) on Sec/SPase I and no-SP sequences. We report these results in Fig. 5. Note that $\mathcal{D}_B$ was homology partitioned to DeepSig's training data in Almagro Armenteros et al. (2019), so comparing TSignal to it is fair, while all other results may be overestimates, due to the lack of homology-based test set partitioning of this experiment.

The predictions of TSignal are carried on diverse sequences, as we predict the sequences on the homology split test set, and from those extract the no-SP and Sec/SPase I sequences present in $\mathcal{D}_B$. Although the models we compare against in Fig. 5 are likely trained on sequences that are similar to those we use to test them, we still notice considerable improvements across most tolerance levels for Sec/SPase I CS predictions, and therefore we can be fairly confident that our model outperforms these previous methods. Note that we could not include SignalP version 6.0 here, as predicting using their publicly available web server would mean testing SignalP version 6.0 with its own training data.

3.3 Model performance analysis

To assess the ability of TSignal model to learn and generalize useful and interpretable information about an SP when predicting its type and CS, we employ a similar approach as Simonyan et al. (2014). As our training procedure is fully data-driven, we investigate the model's ability to learn the information which can be useful for structured prediction models.

We define importance scores as the gradients of the predicted SP type or CS wrt. the input residues. Specifically, we compute the gradients wrt. E_0, defined in Equation (1), which are the last non-contextualized residue representations. These gradient values reflect how much the predicted probability would change for slight changes in the input sequences. Therefore, by comparing the absolute values of these gradients, we are able to tell which residues were most important in the final prediction. For detailed explanations of this experiment, we refer to Supplementary Section 6.

We compute the average input importance scores for each residue in aligned sequences. We investigate 26 TAT/SPase I sequences that have the "RRXFLK" motif and 1682 Sec/SPase II sequences. We align the TAT/SPase I sequences wrt. the RR motif and the Sec/SPase II wrt. the Cys residue i.e.

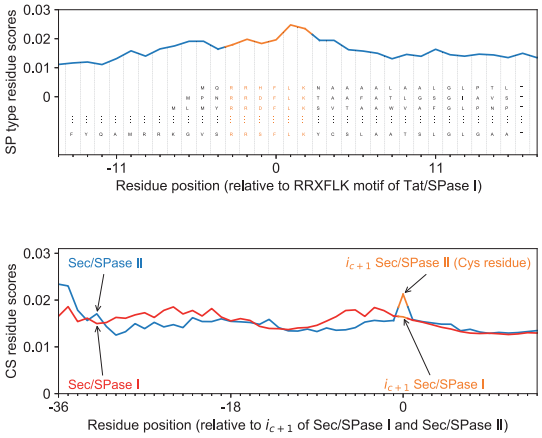

Figure 6. Input importance scores for (top) SP type predictions for TAT/SPase I and (bottom) CS predictions for Sec/SPase I and Sec/SPase II. On the top panel the sequences are aligned by the twin-arginine motif and it is shown how the model distinguishes the input embedding representation (denoted as E_0 in this work) of the RR motif. On the bottom panel the sequences are aligned by the residue i_{c+1} following the CS. For Sec/SPase II residue i_{c+1} is cysteine and it has a high relative importance compared to the Sec/SPase I.

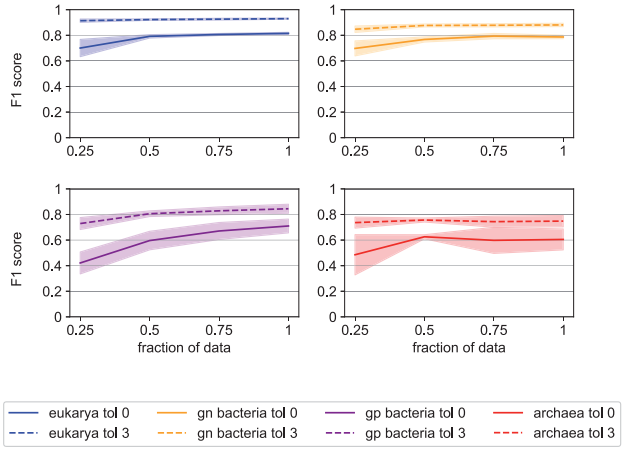

Figure 7. Performance of the TSignal model as a function of the amount of training data. F1 score is evaluated after training the model on 25%, 50%, 75%, and 100% of the full training data while keeping the validation and test data fixed. The test-train procedure is repeated 5 times, and we report the mean and 95% confidence interval (the shaded areas). Results are plotted for tolerances zero and three, for Sec/SPase I SPs from all organism groups.

always present in the first position after the CS. We denote i_{c+1} to be the first residue after the cleavage site. We compute the input importance scores of the predictions for the test set, to investigate whether this information is generalized on sequences that are dissimilar to those used in the training.

In Fig. 6, the top panel shows how the model distinguishes the twin-arginine motif. The twin-arginine motif clearly has a high relative importance in the TAT prediction. We note that the RR motif naturally has some sequence variation (Stanley et al. 2000). Therefore changes in the corresponding residues might give another valid RR motif configuration, which is likely why the relative importance is not as high as the one shown in the panel below. Additionally, the exact position of the motif may also change without rendering the TAT SP non-functional (note that the importance scores also depend on the positional encoding). The bottom panel illustrates the relevance of the cysteine residue (positioned at i_{c+1}) for Sec/SPase II CS predictions. We also align Sec/SPase I sequences on the CS and plot them together. The red curve (Sec/SPase I) shows no higher relative importance of the i_{c+1} residue compared to the other residues surrounding the CS, whereas in Sec/SPase II, there is a very clear spike on the position matching the i_{c+1} (Cys residue).

Next, we evaluated how the performance of TSignal model increases with the amount of training data. We do this by randomly sampling an increasing number of sequences from the datasets $\mathcal{D}_i$ and $\mathcal{D}_j$ as the train-validation set and test on the homology-split $\mathcal{D}_k$. Figure 7 shows that the model performance increases consistently as more training data is used, while the variance of the score estimates decreases. Note that at some point, newly added sequences (e.g. after using >50% of the data) will likely have similar corresponding datapoints (e.g. high sequence identity) already in the training data. The curves, therefore, naturally flatten as less diverse sequences are added to the training data. We assume that as more diverse SP-protein pairs become available, the performance of our architecture will also steadily increase.

We also check the model's probability calibration using the expected calibration error (Guo et al. 2017). To a large degree, the confidence scores of the model reflect the actual probability of the given prediction being correct (see Supplementary Section 5).

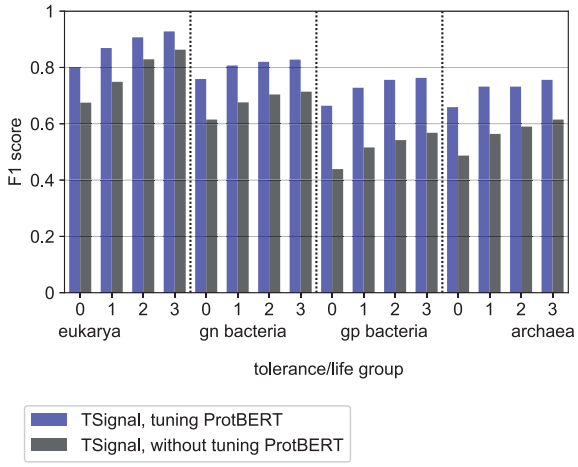

Figure 8. Sec/SPase I CS prediction performance with and without the parameters of ProtBERT being fine-tuned.

A crucial aspect of TSignal's performance was determined by ProtBERT's fine-tuning. In Fig. 8, we show the test CS prediction performance on Sec/SPase I peptides for all organism groups, when the parameters of ProtBERT are tuned or frozen, respectively. Notably, TSignal's performance increases faster with the tolerance when ProtBERT is frozen, which supports the hypothesis that the high amount of context hinders the CS predictions (the model is able to detect only a window where the CS occurs but cannot accurately predict specific locations). A detailed overview of this experiment is described in Supplementary Section 7.

Finally, we briefly analyse the effect of the organism group information $\mathbf{g}$, which we model as a d-dimensional vector, and concatenate it along the residue embedding sequence (see Fig. 2). Similar to SignalP version 6.0 by Teufel et al. (2022), the predictive performance difference is insignificant (please refer to Supplementary Section 9 for the analysis).

4 Conclusion

We introduce, to our knowledge, the first deep learning model for SP and cleavage site prediction, which does not use known biological properties of SPs explicitly (see Fig. 1). Our results show that a transformer-based model provides competitive SP prediction results and improves the accuracy of cleavage site prediction compared to the current state-of-the-art method. Indeed, on several organism groups, our transformer-based model outperforms previous methods. Our analysis also demonstrates that the model performance increases consistently with the amount of data. In other words, as more and more experimentally verified SP sequences will become available, data-driven end-to-end training of expressive deep learning models is likely to further improve the predictive performance. We also note that the amount of variability in TSignal's performance is small, which indicates reliable performance evaluations as well as robust predictions.

TSignal correctly identifies the presence of SPs in two out of four novel SP-protein sequences for which previously developed SP prediction methods could not make correct predictions, showing that unstructured prediction methods have complementary applications in the field.

Model interpretability is generally difficult to obtain for deep learning models and they are usually regarded as "black-boxes." We show that our model generalizes biologically relevant information on homology partitioned data. In addition to the state-of-the-art cleavage site prediction performance, this further illustrates our model's promising generalization potential on diverse sequences. Furthermore, it represents another argument for fully data-driven models, as information that was previously used in structured prediction models is learned by a model using a fully data-driven approach. We also note that hypotheses regarding the importance of various other motifs or specific residues could also be tested using the saliency map approach.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Funding

This work was supported by the Academy of Finland [grant numbers 314445 and 328401 to H.L. and 338836 and 314672 to V.O.P.]; by Sigrid Juselius Foundation [grant to V.O.P]; by the Jane and Aatos Erkko Foundation [grant to V.O.P] and by National Institute of Health [grant number 1R01GM132649 to V.O.P.].

Data availability

Implementation for TSignal is available at https://github.com/Dumitrescu-Alexandru/TSignal. The dataset used (full data $\mathcal{D}$ and its splits, and the $\mathcal{D}_B$ sequences) can be found at https://services.healthtech.dtu.dk/services/SignalP-6.0/.

Acknowledgments

We would like to acknowledge the computational resources provided by the Aalto Science-IT.

References

Almagro Armenteros JJ, Tsirigos KD, Sønderby CK *et al.* SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nat Biotechnol* 2019;**37**:420–3.

Bagos PG, Nikolaou EP, Liakopoulos TD *et al.* Combined prediction of Tat and Sec signal peptides with hidden Markov models. *Bioinformatics* 2010;**26**:2811–7.

Dobson L, Lango T, Reményi I *et al.* Expediting topology data gathering for the topdb database. *Nucleic Acids Res* 2015;**43**:D283–9.

Elnaggar A, Heinzinger M, Dallago C *et al.* ProtTrans: Toward understanding the language of life through self-supervised learning. IEEE Trans Pattern Anal Mach Intell 2022;**44**:7112–27. https://doi.org/10.1109/TPAMI.2021.3095381.

Frank K, Sippl MJ. High-performance signal peptide prediction based on sequence alignment techniques. *Bioinformatics* 2008;**24**:2172–6.

Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Teh YW, Titterington M (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Vol. **9**. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, 249–56.

Guo C, Pleiss G, Sun Y *et al.* On calibration of modern neural networks. In Precup D, Teh YW (eds.), *Proceedings of the 34th International Conference on Machine Learning*, Vol. **70**, Chia Laguna Resort, Sardinia, Italy: PMLR, 2017, 1321–30.

Izmailov P, Podoprikhin D, Garipov T *et al. Averaging weights leads to wider optima and better generalization. In: 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*. In: *Association For Uncertainty in Artificial Intelligence (AUAI), Monterey, CA, USA*. Monterey, CA, USA: AUAI Press for Association for Uncertainty in Artificial Intelligence pp. 876–885 2018.

Juncker AS, Willenbrock H, Von Heijne G *et al.* Prediction of lipoprotein signal peptides in gram-negative bacteria. *Protein Sci* 2003;**12**:1652–62.

Käll L, Krogh A, Sonnhammer EL. A combined transmembrane topology and signal peptide prediction method. *J Mol Biol* 2004;**338**:1027–36.

Liu M, Lara-Lemus R, Shan SO *et al.* Impaired cleavage of preproinsulin signal peptide linked to autosomal-dominant diabetes. *Diabetes* 2012;**61**:828–37.

Owji H, Nezafat N, Negahdaripour M *et al.* A comprehensive review of signal peptides: structure, roles, and applications. *Eur J Cell Biol* 2018;**97**:422–41.

Rajpar M, Koch M, Davies R *et al.* Mutation of the signal peptide region of the bicistronic gene dspp affects translocation to the endoplasmic reticulum and results in defective dentine biomineralization. *Hum Mol Genet* 2002;**11**:2559–65.

Reynolds SM, Käll L, Riffle ME *et al.* Transmembrane topology and signal peptide prediction using dynamic Bayesian networks. *PLoS Comput Biol* 2008;**4**:e1000213.

Savojardo C, Martelli PL, Fariselli P *et al.* Deepsig: deep learning improves signal peptide detection in proteins. *Bioinformatics* 2018;**34**:1690–6.

Sigrist CJ, De Castro E, Cerutti L *et al.* New and continuing developments at PROSITE. *Nucleic Acids Res* 2013;**41**:D344–7.

Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: visualising image classification models and saliency maps. In: *Computer Vision and Pattern Recognition*. 2014.

Stanley NR, Palmer T, Berks BC. The twin arginine consensus motif of tat signal peptides is involved in sec-independent protein targeting in Escherichia coli. *J Biol Chem* 2000;**275**:11591–6.

Teufel F, Almagro Armenteros JJ, Johansen AR *et al.* SignalP 6.0 predicts all five types of signal peptides using protein language models. *Nat Biotechnol* 2022;**40**:1023–5.

Tokunaga M, Tokunaga H, Wu HC. Post-translational modification and processing of Escherichia coli prolipoprotein in vitro. *Proc Natl Acad Sci USA* 1982;**79**:2255–9.

Tsirigos KD, Peters C, Shu N *et al.* The TOPCONS web server for consensus prediction of membrane protein topology and signal peptides. *Nucleic Acids Res* 2015;**43**:W401–7.

UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic Acids Res* 2019;**47**:D506–15.

Vaswani A, Shazeer N, Parmar N *et al.* Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds.), *Advances in Neural Information Processing Systems*, Vol. **30**. Red Hook, NY: Curran Associates, Inc., 2017, (pp. 6000–10). https://dl.acm.org/doi/10.5555/3295222.3295349.

Viklund H, Bernsel A, Skwark M *et al.* SPOCTOPUS: a combined predictor of signal peptides and membrane protein topology. *Bioinformatics* 2008;**24**:2928–9.

Wishart DS, Arndt D, Berjanskii M *et al.* PPT-DB: the protein property prediction and testing database. *Nucleic Acids Res* 2008;**36**: D222–9.

Zhang WX, Pan X, Shen HB. Signal-3L 3.0: improving signal peptide prediction through combining attention deep learning with Window-Based scoring. *J Chem Inf Model* 2020;**60**: 3679–86.