RESEARCH ARTICLE

A Comparison of Mutual Information, Linear Models and Deep Learning Networks for Protein Secondary Structure Prediction

Saida Saad Mohamed Mahmoud[1,2], Beatrice Portelli[1], Giovanni D'Agostino[1], Gianluca Pollastri[3], Giuseppe Serra[*1] and Federico Fogolari[*1]

[1]*Department of Mathematics, Computer Science and Physics, University of Udine, Udine, Italy;*
[2]*Faculty of Science, Cairo University, Cairo, Egypt;*
[3]*School of Computer Science, University College of Dublin, Dublin, Ireland*

ARTICLE HISTORY

Pre-print version of journal paper published at:

https://doi.org/10.2174/1574893618666230417103346

Abstract: ***Background:*** Over the last several decades, predicting protein structures from amino acid sequences has been a core task in bioinformatics. Nowadays the most successful methods employ multiple sequence alignments and can predict the structure with excellent performance. These predictions take advantage of all the amino acids at a given position and their frequencies. However, the effect of single amino acid substitutions in a specific protein tends to be hidden by the alignment profile. For this reason single-sequence-based predictions attract interest even after accurate multiple-alignments methods have become available: the use of single sequences ensures that the effects of a substitution are not confounded by homologous sequences.

Objective: This work aims at understanding how the single-sequence secondary structure prediction of a residue is influenced by the surrounding ones. We aim at understanding how different prediction methods use single-sequence information to predict the structure.

Methods: We compare mutual information, the coefficients of two linear models and three deep learning networks. For the deep learning algorithms we use the DeepLIFT analysis to assess the effect of each residue at each position in the prediction. Mutual information and linear models quantify direct effects, whereas DeepLIFT applied on deep learning networks quantifies both direct and indirect effects.

Conclusion: Our analysis shows how different network architectures use the information of single protein sequences and highlights their differences with respect to linear models. In particular the deep learning implementations take into account context and single position information differently, with the best results obtained using BERT architecture.

Keywords: Secondary structure prediction, single sequence, mutual information, linear model, deep learning, neural network, LSTM, BERT.

1. INTRODUCTION

The three-dimensional structure of a protein is determined by its amino acid sequence and it is key to its functional mechanisms. In addition, its prediction is a major challenge for biologists [1,2]. In recent years the astonishing success of the AlphaFold prediction system based on multiple alignments and deep learning [3] essentially solved the problem.

However, understanding the effect of single amino acid substitutions on the stability of the structure, or on its interactions, still remains a difficult problem. In order to do this it is important to estimate the effect of each amino acid at each position on the secondary structure of the neighboring residues.

Here we address the prediction of secondary structure from single sequences to characterize and compare different approaches. We aim at understanding how the different predictive approaches leverage the information coming from neighboring amino acids to predict the local structure of the protein.

*Address correspondence to this author at the Department of Mathematics, Computer Science and Physics, University of Udine, Via delle Scienze 206, Udine, Italy; Tel/Fax: ++39-0432-558493, ++39-0432-558499

We consider protein secondary structure prediction which is one of the classical problems in bioinformatics and it has been addressed extensively using many methods. We consider predictions which do not use evolutionary information, i.e. using only a single sequence, in order to assess the influence that neighboring positions have on the secondary structure of a given position. As detailed below we compare linear models, different deep learning architectures and mutual information, to highlight the differences in the ways the different algorithms use sequence information.

The interest in predicting secondary structure stemmed from different reasons:

1. identifying secondary structures provides an idea about overall structural categories;
2. secondary structure plays an important role in determining how proteins fold [4,5,6];
3. since the practical experiments that are used to determine protein structure were and are costly and time-consuming, it appears convenient to develop prediction methods. In addition, the number of known sequences is at least 1,000 times bigger than that of resolved structures [7], a fact that calls for predictive methods.

Although various techniques were introduced for predicting protein secondary structure from multiple sequence alignments, there have been relatively few attempts to improve the performance of secondary structure predictions from single sequences. The prediction of secondary structure from single sequence challenge, as reviewed by Rost and Sander [8], started with Pauling and Corey in 1951, when they predicted helical and sheet conformations for protein polypeptide backbone. In addition, they focused on predicting structural properties of proteins like backbone dihedral angles, leveraging this information for secondary structure prediction [9,10]. After these early works, a first generation of prediction methods has been put forward based on local amino acids' propensities to adopt a particular secondary structure, like e.g. the Chou and Fasman method [11]. The second generation of algorithms was based on information theory instead, considering the effects of amino acids and pairs of amino acids within a window of 11 to 21 adjacent residues. Among these methods the Garnier-Osguthorpe-Robson method was among the best available, together with its subsequent refinements [12,13,14].

Later on, evolutionary information and the introduction of novel neural network architectures led to an outstanding increase in the accuracy of the predictions [15,16]. The interested reader will find details and further references in the papers cited in this work and in recent reviews [17].

More recently novel deep learning methods have been applied to the problem of single sequence prediction of secondary structure.

Heffernan et al. [18] proposed LSTM-BRNNs (SPIDER3-Single), a single-sequence-based model using long short-term memory bidirectional recurrent neural networks to solve the task. The model can predict multiple one-dimensional (1D) structural properties with relatively high accuracy, especially for non-homologous sequences.

The ProteinUnet architecture has been introduced as an alternative way to SPIDER3-Single for sequence-based prediction of protein secondary structure. The model is based on the U-NET architecture which consists of blocks placed symmetrically as contracting and expanding paths [19].

The deep neural network architectures described above learn patterns from protein sequences and succeed in giving more accurate predictions than traditional methods. Still, the way this is done is hidden in the connections and parameters of the network, which acts as a black box. It is somehow frustrating that the learning machine uses the patterns in the sequences in the best way, but it does not disclose this information to the user.

In recent years several methods have been developed to relate the predictions of the model architecture to its input features (e.g. DeepLIFT [20]), making it possible to assess the interplay of input variables in determining the output. These input-output relationships may be compared in principle with the mutual information of input variables and/or pairs of input variables and the output, and with linear models, thus providing a glimpse of how the network is using input information.

Here we use different approaches to the task of predicting protein secondary structure from protein single sequences:

1. a linear model solved exactly for coefficients minimizing the mean square prediction error;
2. a linear model implemented as a single-layer feedforward neural network;
3. a neural network based on Bidirectional Long Short-Term Memory (Bi-LSTM) architecture;
4. a neural network based on the Transformer framework with a novel BERT-based model;
5. a neural network based on the Encoder-Decoder framework, utilizing a T5 model.

Using different predictive models, based on really different and recent architectures gives us the opportunity to compare extensively performances and understanding how single position information is linked to the output.

We compare the different predictive approaches in terms of feature importance. This corresponds to the absolute magnitude of the coefficients for the linear models, whereas for neural networks it is assessed by propagating activation differences in the protein single-sequence with respect to a reference activation, using the DeepLIFT (Deep Learning Important FeaTures) [20] method.

The comparison of the models among themselves and with the mutual information may help in rationalizing the way deep neural network architectures use the information coded in protein single-sequences.

For completeness we must note that a large number of different approaches are currently being used in structural predictions concerning proteins, including for example a specific protein language model (AminoBERT) [21], many-objective genetic algorithms [22], Deep Belief Networks [23]. Also a large effort may be made to optimize methods [24]. Natural Language Processing, from which field many of the methods described here are derived, is an extremely active area of research. Although multiple Transformer-based models are introduced every year, most of them are built to solve very specific tasks, with particular sets of assumptions on the input. Since this is one of the first works which investigates the performance of Transformer-based models (BERT and T5) on the task of protein secondary structure prediction, we chose to experiment with their vanilla implementation instead of exploring more recent (but highly specialized) BERT-derived models such as AminoBERT [21], Longformer [25] (developed for long document understanding), Autoformer [26] (developed for long-range forecasting), and Pegasus [27] (developed for summarization). To the best of our knowledge, this is also the first attempt at using text-generation approaches such as T5 for this task.

The networks employed here use the information encoded in the sequence characters in a rather nonspecific manner, allowing for comparisons and an easier interpretation of the differences among different models.

2. MATERIALS AND METHODS

2.1 Dataset and Metrics

The dataset of structures used in this work was obtained by downloading non-redundant sequences (at less than 40% identity) from the culling server Pisces [28] (http://dunbrack.fccc.edu/pisces/) with resolution less than 4 Å and R-factor less than 1.0. The set entails 27,316 sequences. Additionally splitting sequences at breaks in connectivity of the corresponding PDB file, resulted in 35,964 sequences with maximum length of 2,166 amino acids.

This dataset was split into a Training set (TR33964) consisting of 33,964 different chains and a test dataset (TS2000) consisting of 2,000 different chains. Each subsequence with no breaks was divided into subsequences of length 17, centered around the position for which we aim to predict the secondary structure (see Fig. **(1)**). When the central residue is at the beginning (or end) of the original sequence, and therefore lacks preceding (or following) amino acids, the sequences are padded by adding another code that represents the lack of amino acids (see the letter “B” in Fig. (1), blue subsequence). This information will then be sensibly encoded in each of the predictive models.

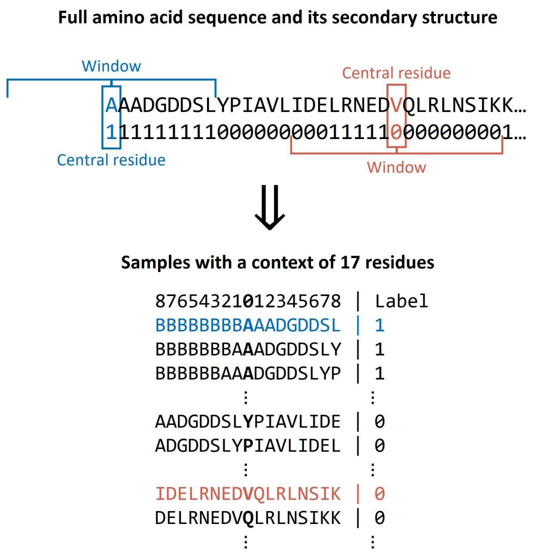

Fig. (1). Example of the process that splits the full protein sequence in subsequences with a context window of 17 residues/amino acids.

This procedure results in 5,672,821 samples for the Training set and 490,612 samples for the Test set, where each sample consists in a subsequence of 17 amino acids (8 preceding and 8 following the central residue) and a label indicating the secondary structure of the protein at the position of the central residue.

The length of the subsequence was first chosen by Garnier, Osguthorpe and Robson based on the analysis of the information content at increasing distances from the central residue [14]. The same window length has been used by other successful methods, including the PHD prediction server [29].

In addition, we did some experiments using different subsequence lengths and beyond this length the gain in performance was marginal.

The rationale for generating the dataset was to have a larger dataset than the ones used in the papers describing SPIDER3-Single and ProteinUnet, but still non-redundant at least at less than 40% identity. The dataset was however found not to influence significantly the overall performance of the models, as preliminary experiments showed that the same model developed on TR33964/TS2000 or on the SPIDER3-Single dataset reached the same classification accuracy. At the same time, a larger dataset enables us to perform a more in-depth analysis of the models' predictions.

The secondary structure of the central residue was assigned using the DSSP program as implemented by Vriend and collaborators [30] which outputs for each residue one of eight possible states [31]. The eight states were further clustered in three main classes as detailed in the following section.

2.2 Features and Model Output

The input for each of the predictive models is one feature vector for each of the 17 amino acids in the subsequence. In the case of the linear models and the LSTM, the feature vector is a one-hot encoding of the amino acid (size 20), while in the case of BERT and T5 it is a word embedding generated by the model itself. As BERT and T5 accept plain text as input, they are given the sequence of 17 amino acids directly as characters.

We did not augment the data with any other single sequence features, such as BLOSUM62 substitution scores [32] and/or physicochemical properties of the amino acid [33,34]. These features did not offer any improvement as shown in [33].

Each model predicts the secondary structure state for the amino acid at the center of the subsequence (see Fig. **(1)**). The secondary structure state is represented by one of the possible eight secondary structure states, grouped as follows:

1. the three helical states: 3_{10}-helix (G), α-helix (H), and π-helix (I);
2. the two extended hydrogen-bonded states: β-bridge (B) and β-strand (E);
3. the three coil types: high curvature loop (S), β-turn (T), and coil (C).

Such classification based on hydrogen bonding patterns is due to Kabsch and Sander [31] who undertook a rigorous classification of secondary structures. However, it is far more common to discuss structural features of proteins using the following grouping of states: states G, H, and I are grouped into a Helix (H) state; states B and E into an Extended (E) state; and states S, T, and C into a Coil (C) state.

We designed our networks and linear models to independently predict this 3-state (labeled H, E or C) secondary structure.

2.3 Mutual Information

Before analyzing the models, we want to provide a reference assessment of the influence of single (or pair) of amino acid positions on the secondary structure state of a central residue in 17-residue subsequences.

In order to do this we consider as a measure of information:

- the entropy of the 17 positions and the secondary structure state;
- the mutual information of each pair of positions and state;
- the three-variable mutual information for each pair of positions and the secondary structure state.

Entropy and mutual information are defined as follows.

Each position in the 17-residue window may assume 21 possible values: 20 amino acids plus the "absence of amino acids", used as padding for the initial and final positions in the sequence. The central position can only assume 20 values, as we require the presence of a central amino acid associated to a secondary structure state. The latter can be one of three possible states.

In the following equations, we reserve index 0 for the secondary structure state variable and indices 1 to 17 for the amino acid position variable.

The entropy of each variable is estimated from, that is the counts of occurrences of its possible values $\{v_l\}$:

$$\hat{S}(x_i) = - \sum_{l=1,\dots,ns_i} \frac{n_{il}}{\sum_{l=1,\dots,ns_i} n_{il}} \log\left(\frac{n_{il}}{\sum_{l=1,\dots,ns_i} n_{il}} \right)$$

where ns_i is the number of possible values (states) of variable x_i.

The joint entropy of two variables is computed as:

$$\hat{S}(x_i, x_j) = - \sum_{l=1,\dots,ns_i\, m=1,\dots,ns_j} \frac{n_{ij;lm}}{\sum_{l=1,\dots,ns_i\, m=1,\dots,ns_j} n_{ij;lm}} \log\left(\frac{n_{ij;lm}}{\sum_{l=1,\dots,ns_i\, m=1,\dots,ns_j} n_{ij;lm}} \right)$$

where $n_{ij;lm}$ is the count of simultaneous occurrences of variable i assuming the value v_l and variable j assuming the value v_m.

The mutual information between two variables is estimated as:

$$\hat{MI}(x_i, x_j) = \hat{S}(x_i) + \hat{S}(x_j) - \hat{S}(x_i, x_j)$$

Note that

$$\hat{MI}(x_i, x_i) = \hat{S}(x_i)$$

In order to quantify also the effect of pairs of positions on the secondary structure state we consider the three-variable joint entropy and mutual information. The joint entropy of three variables is defined similarly as above:

$$\hat{S}(x_i, x_j, x_k) = - \sum_{l=1,\dots,ns_i\, m=1,\dots,ns_j\, n=1,\dots,ns_k} \frac{n_{ijk;lmn}}{\sum_{l=1,\dots,ns_i\, m=1,\dots,ns_j\, n=1,\dots,ns_k} n_{ijk;lmn}} \log\left(\frac{n_{ijk;lmn}}{\sum_{l=1,\dots,ns_i\, m=1,\dots,ns_j\, n=1,\dots,ns_k} n_{ijk;lmn}} \right)$$

The mutual information of three-variable is defined as [35]:

$$\hat{MI}(x_i, x_j, x_k) = \hat{S}(x_i) + \hat{S}(x_j) + \hat{S}(x_k) - \hat{S}(x_i, x_j) - \hat{S}(x_i, x_k) - \hat{S}(x_j, x_k) + \hat{S}(x_i, x_j, x_k)$$

When one of the variables is the secondary structure state, the three-variable mutual information may be used to assess the information which is gained on the secondary structure when the amino acids at two positions are known simultaneously as opposed to when they are known independently. The latter case is implemented in linear models as those discussed here, whereas the former is implemented in machine learning algorithms.

When the three-variable mutual information involving the secondary structure variable is written in terms of two-variable (albeit also composite) mutual information:

$$\hat{MI}(x_i, x_j, x_0) = \hat{MI}(x_i, x_0) + \hat{MI}(x_j, x_0) - \hat{MI}(\{x_i, x_j\}, x_0)$$

It is apparent that any additional effect coming from the combination of x_i and x_j, with respect to the separate effect of x_i and x_j will translate in a negative three-variable mutual information.

2.4 Linear Models

We implemented two linear models: Multilinear Regression (LIN) and a Feedforward Network (FFN). In both cases, the amino acids are encoded as one-hot vectors of size 20. The padding amino acid, which represents the lack of previous/following amino acids in the original sequence, is encoded as a 20-dimensional zero vector.

2.4.1 Multilinear Regression (LIN)

A linear model (LIN) was implemented in an ad hoc program to find the best 340 x 3 linear coefficients (17 one-hot vectors of size 20 to fit the one-hot vector of length 3). A linear system of the form $Ax = y$ was solved by pseudo-inversion, i.e. $\vec{x} = \left(A^t A\right)^{-1} A^t y$, which is the solution that minimizes the sum of square errors in prediction. Due to matrix size, the model was fitted on only 136,000 randomly selected input samples. However, repeated experiments showed that the coefficients found were essentially independent of the set of inputs chosen.

2.4.2 Feedforward Network (FFN)

We implemented a simple feedforward neural network (FFN) model, consisting of one linear layer and no activation function. As for LIN, the input of FFN was the concatenation of the 17 one-hot vectors of the subsequence, and the output was a one-hot vector of size 3. The model was trained on the whole Training set for 50 epochs with stochastic gradient descent, using a batch size of 128 samples, a learning rate of 1e-3 and Mean Square Error as a loss function.

Fig. **(2)** shows a schema of both the LIN and FFN model.

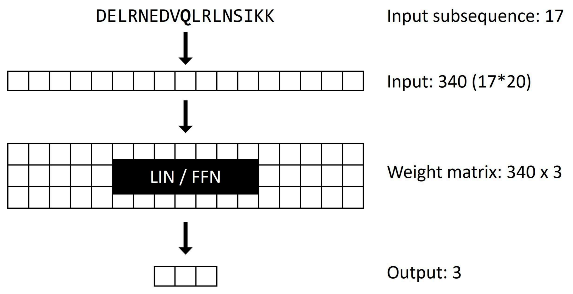

Fig. (2). Schema of the LIN and FNN architectures.

2.5 Bi-LSTM neural network (LSTM)

A Bidirectional LSTM (Bi-LSTM) is a variant of the LSTM model [36] which scans the input sequence in both directions: forward, from the start to the end of the sequence, using a Forward LSTM layer made of forward cells (Fig. **(3)**, light blue layer); and backwards, from the end to the start of the sequence, using a Backward LSTM layer made of backward cells (Fig. **(3)**, red layer). At time t, a Bi-LSTM has two different active cells: a forward cell $\vec{c}$ and a backward cell c'. The forward cell uses the current input $\vec{x}_t$ and the state of the previous cell $\vec{c}_{t-1}$ to calculate its internal cell state $\vec{c}_t$, and hidden state $\vec{h}_t$. In Fig. **(3)**, the cell states flow left-to-right, while the hidden states are output from the bottom of the cells. At the same time, the backward cell uses its current input x_t' and the state of the previous backward cell c_{t-1}' to calculate its internal cell state c_t', and its hidden state h_t'. Cells states of the Backward LSTM flow from right-to-left, while hidden states are output from the bottom of the cells. The choice of using a Bi-LSTM instead of a simple LSTM network to predict the secondary structure is justified by the fact that such typology of neural network has achieved promising results while also beating the performance obtained with simple LSTMs, as shown in [18]. This is given by the ability of the Bi-LSTM to use more contextual information, as is it able to read the sequence both ways.

We implemented a Bi-LSTM model with a hidden state size of 64. The last hidden states of the forward and backward LSTMs are concatenated (size 128) and passed to a small postprocessing network to project them to the output space. The postprocessing network is formed by a linear layer (128x32), a ReLU activation function [37], and another linear layer (32x3). A schema of the network is shown in Fig. **(3)**.

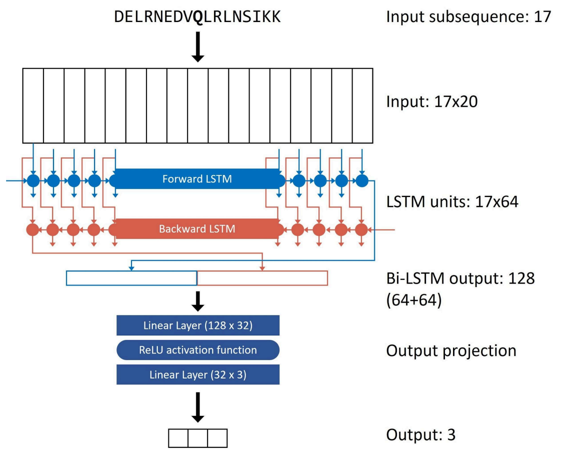

Fig. (3). Schema of the proposed Bi-LSTM architecture.

Similarly to the two linear models, the amino acids are encoded as 20-dimensional one-hot vectors, and the padding amino acid is represented by a 20-dimensional zero vector. The input of the LSTM is a sequence of 17 one-hot vectors representing the amino acids, while the output is a one-hot vector of size 3. Similarly to the FFN model, LSTM was trained on the whole Training dataset for 50 epochs with stochastic gradient descent, using a batch size of 128, learning rate of 1e-1 and Mean Square Error as a loss function.

2.6 BERT-based Neural Network (BERT)

BERT [38] (Bidirectional Encoder Representations from Transformers) is a Transformers-based architecture that has revolutionized many tasks in the Natural Language Understanding field. It is a large language model, composed of a series of stacked encoder blocks, aimed at creating deeply contextualized word embeddings. Each encoder processes the whole input sequence and transforms it into an embedding sequence that can be either used for inference or fed into another encoder block for further processing. This model is commonly used in the following way:

- A domain of interest is chosen, together with a large corpus that can be used to train the model.
- The BERT model is initialized from scratch and pretrained on the large corpus, to learn domain-specific correlations between the words/tokens in the given samples. The pretraining is usually performed using Masked Language Modeling.
- The pretrained model is extended with new untrained layers and finetuned (i.e., trained) on a new specific task in the same domain of interest.

In the last years, a great variety of pretrained BERT-based models have been created and publicly released, such as PubMedBERT [39] for the biomedical domain, LEGAL-BERT [40] for legal documents and CodeBERT [41] for source code.

We implemented a BERT-based model and trained it from scratch on the domain of protein encoding and secondary structure prediction. The basic BERT model is usually composed of 12 encoder blocks and can read sequences up to 512 words. Preliminary experiments showed that such a large network size led to a low performance in our task. For this reason, we created a model with only 4 encoder blocks and a maximum sequence length of 64 words. The embedding size of the model is 768 (standard BERT embeddings).

The model was *pretrained* on 20% of the full protein sequences in the training set with the task of Masked Language Modeling: given an input sequence, a random amino acid was “masked” (i.e., replaced with the special token [MASK]) and the model was asked to guess the correct amino acid to fill the gap. The pretraining was carried out for 50 epochs, with a batch size of 2048. The vocabulary of the model only includes the characters for the 20 amino acids (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y) plus five special tokens needed for the model’s functioning ([CLS], [PAD], [SEP], [UNK], [MASK]). The padding amino acid is encoded using the appropriate [PAD] special token.

The pretrained model was then extended with an untrained linear layer for *finetuning*. The linear layer projects the word embedding of the central amino acid (size 768) to the output space (3 states). Fig. **(4)** is a schema of the final model, which takes as input the string-representation of the amino acids’ subsequence, embeds it, runs it through a stack of encoders and then projects the embedding to the three possible output states. BERT was finetuned on the training set subsequences for 20 epochs, with a batch size of 128 and a learning rate of 1e-6.

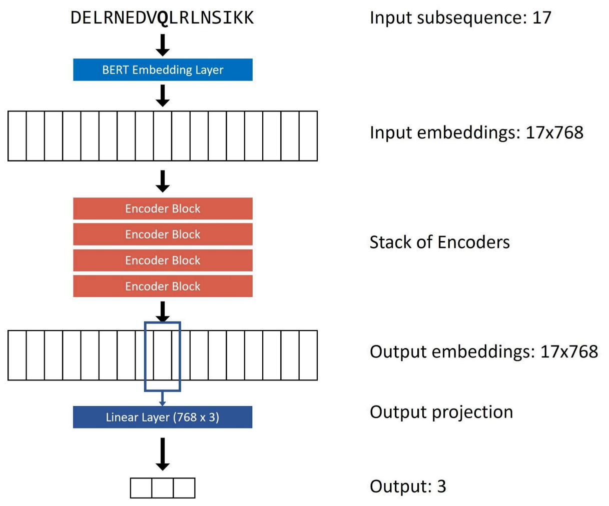

Fig. (4). Schema of the implemented BERT architecture.

2.7 T5 Neural Network (T5)

T5 [42] is a text-to-text encoder-decoder model pretrained on a multi-task mixture of unsupervised and supervised tasks. It is composed of a stack of encoder blocks (similarly to BERT), which output an embedding of the input sequence, and a stack of decoder blocks, which transform the embedding representation back into plain text. Fig. **(5)** shows a schema of the working of T5. Similarly to BERT, the 20 amino acids are encoded directly using their characters (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y), while the padding amino acid is encoded using the appropriate <pad> special token. The dimension of the embeddings is 512, and the stacks of encoders/decoders consists of six blocks each. The output of the model is a string containing one of the three words "Helix", "Coil" or "Extended". T5 was finetuned on the training set subsequences for 9 epochs, with a batch size of 256 and a learning rate of 1e-5.

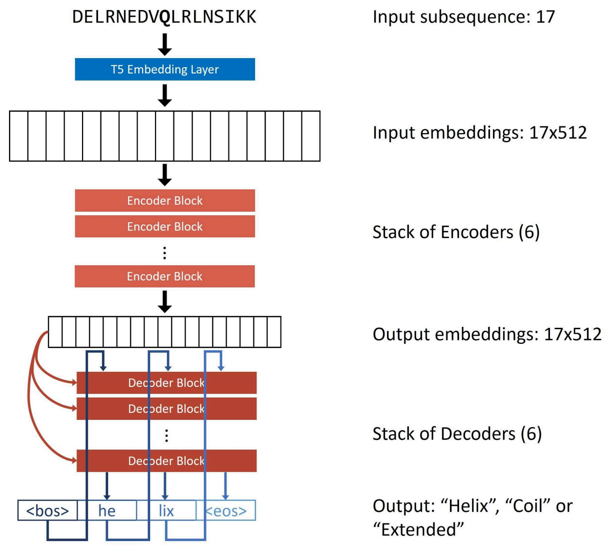

Fig. (5). Schema of the T5 architecture.

2.8 DeepLIFT Analysis

DeepLIFT is a method to that can be used to analyze the inner workings of a neural network, drawing relations between its input features and the resulting output. In particular, its aim is to determine the "contribution" that each input feature has in generating a specific prediction.

The DeepLIFT method requires the user to supply the network with a "reference" input (i.e., an example of an empty sequence or background noise) and a real input (e.g., one of the 17-amino acid subsequences). The algorithm then compares the output produced by the network on the reference input with the output obtained using the real input sequence. It then attributes the change of the output to the previous layer in the neural network, using gradients and network activations to determine how much each neuron contributed to the output change. The algorithm keeps traversing the network backwards (from outputs to inputs), calculating the impact that every neuron had in obtaining the final output. The final values calculated for the input features are called *attributions* or *attribution scores*.

We used the DeepLIFT algorithm on all neural network architectures (both linear and non-linear). In order to observe the patterns that the network learned for the three kinds of protein secondary structure, we calculated the attributions separately for the sets of samples belonging to the three output classes. We used an "empty" sequence as a reference value for the method, that is 17 repetitions of the value used to pad the input sequences (i.e., the "B" character in Fig. **(1)**). More specifically, we used an all-zero vector for the models that employ one-hot encoding, and the "padding" special token for the BERT and T5 models).

The attribution scores can be used to gain insights on the patterns used by the neural networks to determine the protein secondary structure, and can be compared to visualizations such as the absolute magnitude of the coefficients for the linear models, or the values of the Mutual Information for a position-state pair.

2.9 Implementation

All the methods were implemented in the environment containing Python 3.7, Pytorch accelerated [43] by CUDA 10.1 and CUDA 11.2.

3. RESULTS AND DISCUSSION

3.1 Mutual Information

The mutual information was computed as described in Section 2.3.

Entropy. The single variable entropy was almost identical for all of the 17 positions, with the exception of a slight increase towards the ends. The cause of the increased entropy for the tails of the sequence is that the terminal residues can also assume the value “absence of amino acid”. Although the “absence of amino acid” occurs with a low frequency, this additional value is never present at the central position, and increasingly more frequent moving towards the end of the sequence. This leads the entropy to increase in a similar pattern, from 2.89 (central position) to 2.94 at the tails of the distribution, in a logarithmic fashion.

Two-variable Mutual Information (position-position). When considering pairs of positions, the two-variable mutual information is dominated by the correlation of the absence of amino acid for windows entailing terminal residues, e.g. because the absence of amino acids at a position before the central one implies the absence of amino acids at all preceding positions.

Two-variable Mutual Information (position-state). The most interesting result obtained with two-variable mutual information, is the one calculated between each input position and the predicted state. We computed it taking into account all possible states and considering only one state at a time, i.e. categorizing for example Helix and non-Helix states. The position-state mutual information for each position is reported in Fig. **(6)**.

The mutual information highlights the direct influence that each position has on the secondary structure state. As expected, the mutual information is bell-shaped around the central position (the position for which the secondary structure is considered). When the single states, Helix, Coil and Extended, are considered some differences are apparent. For Helices the curve has a larger width than for Coils and Extended structures, consistent with the spread in local interactions. Another interesting feature is that for Coils there is a peak at the central position (as expected), but also at the following one, consistent with the disruptive effect some amino acids have on other secondary structures. For Extended structures the mutual information appears less consistent with the non-local interactions that stabilize this secondary structure element. However, the interpretation of the plot is not straightforward: as stated before, we compute the mutual information comparing Extended and non-Extended samples. Because of this, the mutual information implicitly takes into account the characteristics of Helices and Coils too (grouped together in the non-Extended class), and the plot highlights which positions are more important to distinguish between these two groups.

The position-state mutual information should provide a good reference for linear models which, albeit only additively, take into account the effect that each amino acid at each position has on the secondary structure of the central position.

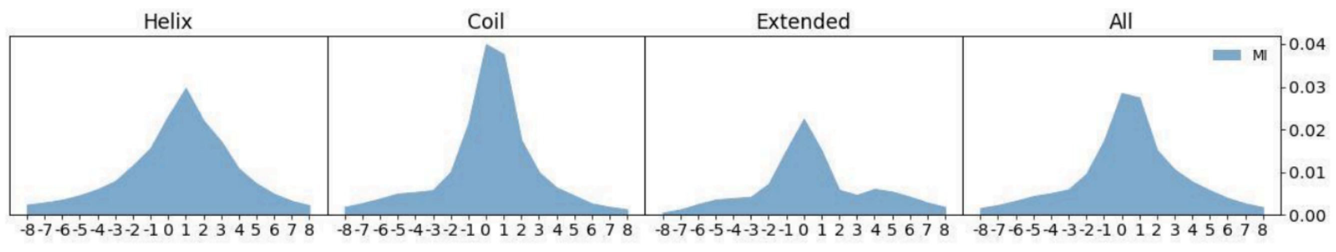

Fig. (6). Two-variable Mutual information (position-state) of each position along the 17-residue window around the position for which secondary structure is considered. From left to right: mutual information obtained categorizing Helix states, Coil states, Extended states, and the overall Mutual Information.

Three-variable Mutual Information (position-position-state). Another piece of information which is not caught by linear models is the additional effect of pairs of positions. This is captured by the three-variable (position, position, secondary structure) mutual information, computed as described in Methods section. This three-variable mutual information is directly related to how much information is gained by knowing the residue at two positions simultaneously with respect to knowing them separately. Contrary to the two-variable mutual information, the three-variable mutual information can be positive (if the pieces of information provided by the two variables on the third one are not independent) or negative (when the two variables are cooperative or anti-cooperative). The corresponding heatmaps for all the states together and for the separate states are reported in Fig. **(7)**.

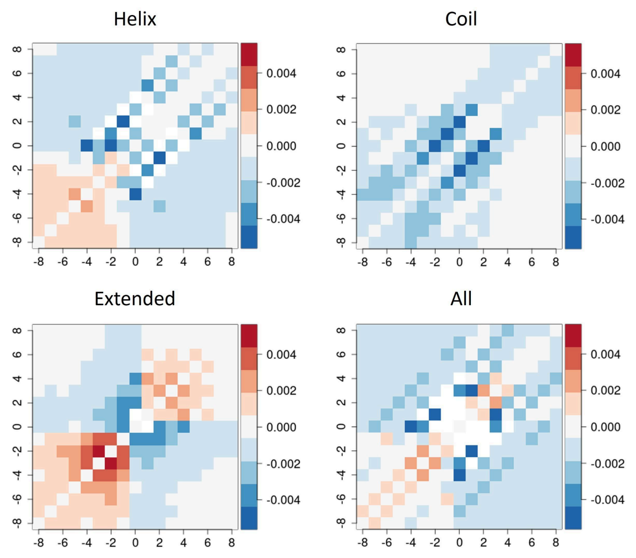

Fig. (7). Three variable mutual information for each pair of positions and the secondary structure state of the central residue. The information along the diagonal corresponds to the information reported in Fig. **(6)** and has been set to zero, for better readability of the maps. Heatmaps refer to: Helices, Coils, Extended structures, and all secondary structures.

As for the two-variable mutual information, the heatmaps for the single states are calculated considering a single state (e.g., Helix vs non-Helix), so they also implicitly contain information from other states. Although interpretation is not obvious, patterns are different among different secondary structure elements, such as the negative peak at (0,-4) for Helices, or the interactions parallel to the diagonal for Coils, showing that there are (albeit small) cooperative effects which involve a large number of pairs of positions.

In order to provide an illustration of this important point we consider only Helix and Extended structures. The three variable mutual information is reported in Fig. **(8)** and displays negative (informative) values along diagonals running parallel to the main diagonal, in particular at distances ±1, ±3, and ±4. The map is thus reporting about features determining both Helix and Extended structures as said above.

It is clear that the information reported in these heatmaps (resulting from the joint effect of pairs of positions) is not modelled by linear models, whereas neural networks should be able to reproduce even complex dependencies of secondary structure on pairs of amino acids. The mutual information computed here offers a convenient way to compare predictive methods.

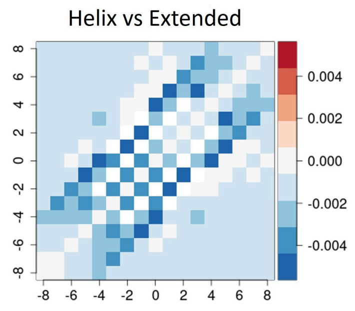

Fig. (8). Three variable mutual information for each pair of positions and the secondary structure state of the central residue when only Helix and Extended structures are considered. The information along the diagonal has been set to zero, for better readability of the maps.

3.2 Linear Models

3.2.1 Multilinear Regression (LIN)

A linear model has been implemented as a set of 136,000 overdetermined equations in 340 variables (17 positions times 20 elements of the 1-hot vector for each position), which has been solved to find the solution leading to minimum sum of square errors by pseudoinversion as detailed in Section 2.4.1.

In order to compare the results with the mutual information results, the 340 coefficients were grouped 20 by 20, and we calculated the average of the absolute value of the 20 coefficients as a measure of the overall importance of that position.

The results are reported in Fig. **(9)** where the different profiles for Helix, Coil and Extended structures are apparent, together with the silhouette of the mutual information, represented by the dashed line.

The plots have a bell shape, which is wider for Helices and Extended structures than for Coil structures. Similarly to what is seen in the mutual information, the bell shape for the Extended structures is less defined and noisier.

The distribution of the linear coefficients broadly parallels that of the mutual information (reported in Fig. **(6)**), although the difference between the central peak and the tails is more accentuated. The linear correlation of $-\overline{c}_j log\left(\overline{c}_j\right)$ of the former quantity with the mutual information of each position with the secondary structure of the central residue is 0.90.

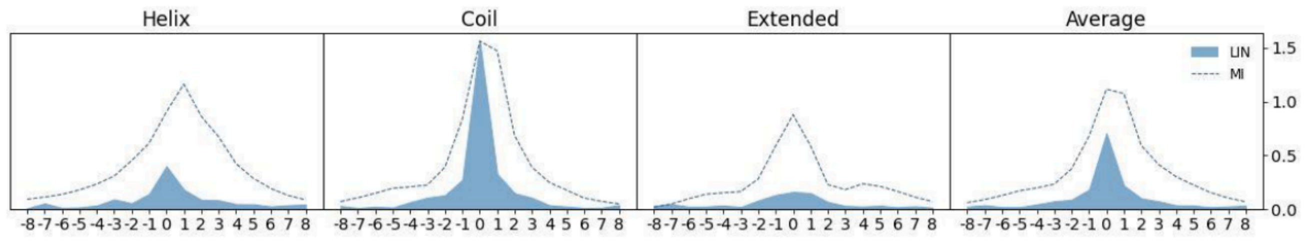

Fig. (9). Weights of LIN for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average. For each position we plot the average of the absolute values of the 20 coefficients that encode the amino acid.

3.2.2 Feedforward Network (FFN)

The linear model implemented using neural networks was analyzed in two ways: considering its weights, similarly to what was done for the LIN model, and using the DeepLIFT algorithm to determine the attribution scores of each input position.

Fig. **(10)** displays both plots for the three output classes. We observe that the weight distribution is more similar to the mutual information than to the other linear model (LIN). It is interesting to observe such a huge difference in the weights of the

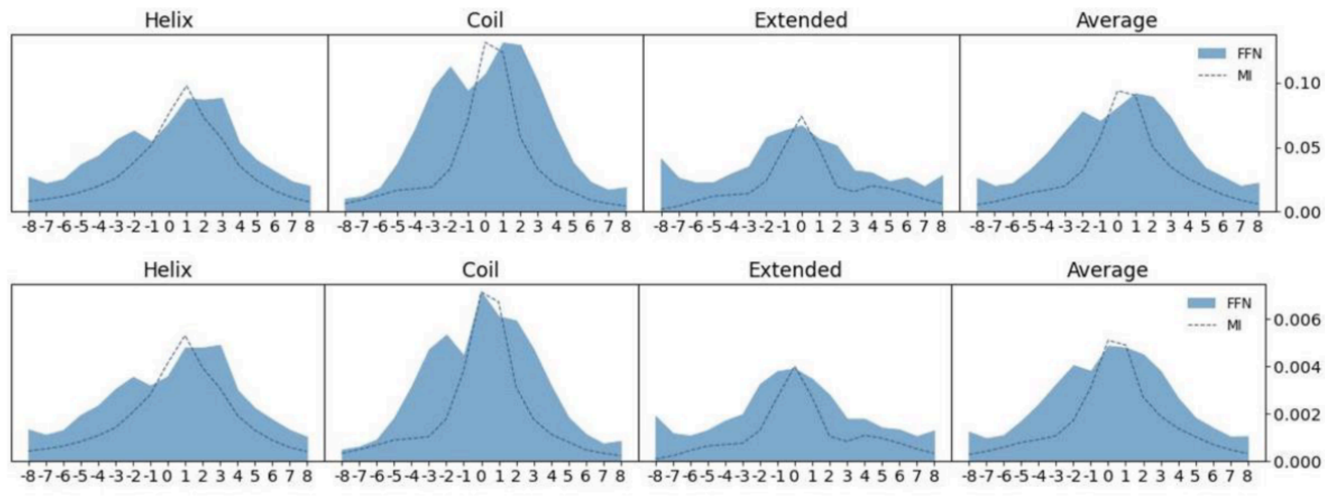

Fig. (10). Weights and DeepLIFT attributions of the FFN model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average. Top: weights of the FFN, for each position we plot the average of the absolute values of the 20 coefficients that encode the amino acid. Bottom: DeepLIFT attributions for each position.

model, since FFN was trained without using any weight normalization. We assume that the weight distribution of FFN is largely due to its weight initialization and some characteristics of the training samples. In particular, the weights of the neural network are first initialized to random values, sampled from a continuous uniform distribution between $\frac{-1}{\sqrt{340}} \approx -0.054$ and $\frac{+1}{\sqrt{340}} \approx +0.054$, and then adjusted during training. We can see that 0.05 (after absolute value) is also the pivotal number to distinguish informative and non-informative positions in Fig. **(10)** top. This is likely because the model's weights are modified only if they contribute to a wrong prediction, while they are left unchanged if they do not impact the prediction negatively. Since most of the values in the tail positions have low impact on the model's output (see mutual information), the model initially lowers their weights during the first epochs, but then only focuses on updating the weights related to the central region, which contain more information and bring greater gains in accuracy. This explains why most of the tail positions have weights close to 0.05: they have low impact on the model's accuracy, so their values remain similar to the ones set during the random initialization.

The bell-shaped curve for the Coil class is wider for the FFN model, and it displays two peaks around position -2 and 1. This behavior loosely imitates the shifted peak of the mutual information, but the presence of a double peak is probably an effect of the stochastic nature of the training algorithm.

We can also note that the weights and DeepLIFT attributions of the FFN model create remarkably similar representations, showing that the DeepLIFT representations computed for the following neural networks can be used to draw similarities between the models.

3.3 Bi-LSTM Neural Network (LSTM)

For the LSTM neural network, we report the attributions calculated with the DeepLIFT algorithm for the samples belonging to the three output classes (Fig. **(11)**).

Comparing the attribution scores with the mutual information in Fig. **(6)**, we see that the bell curves are even more similar than the ones produced by the two linear models. In particular, the LSTM attribution score distribution for the Extended class resembles the mutual information scores very closely, and the distribution for the Helices peaks at position 1 (same as the mutual information).

It is surprising that a model containing non-linearities is better at replicating the mutual information patterns than the two linear models previously considered. This reflects the fact that linear models poorly reproduce the single-position state mutual information, and additionally that the average absolute value of the 20 coefficients (instead of e.g. the root mean square, or its logarithm) at each of the 17 positions might not be the best quantity to compare with mutual information.

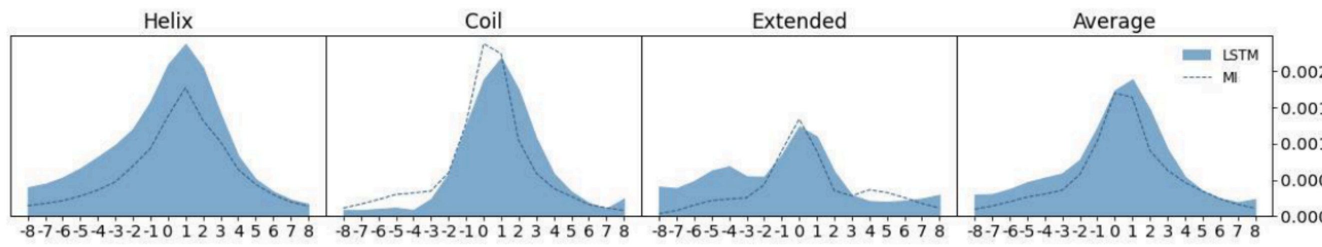

Fig. (11). DeepLIFT attributions of the LSTM model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average.

3.4 BERT-based Neural Network (BERT)

Fig. **(12)** reports the attributions scores of our BERT-based implementation on the three states. In this case, the shapes of the curves for the three output classes are very similar to each other: they all have a strong peak at position 0, and the attribution score gradually decreases on both sides in a mostly symmetrical way.

Compared with the mutual information, the attributions scores are more evenly distributed among the positions. This can be explained by the densely connected structure of the stacked encoder blocks, which encourages the even spread of information along all positions of the sequence. It is also interesting to note that the attributions for Extended structures are generally higher than the ones of Coils, which is significantly different from the behavior of all the previous models. This might indicate a better ability in recognizing Extended patterns, which leads to stronger activations.

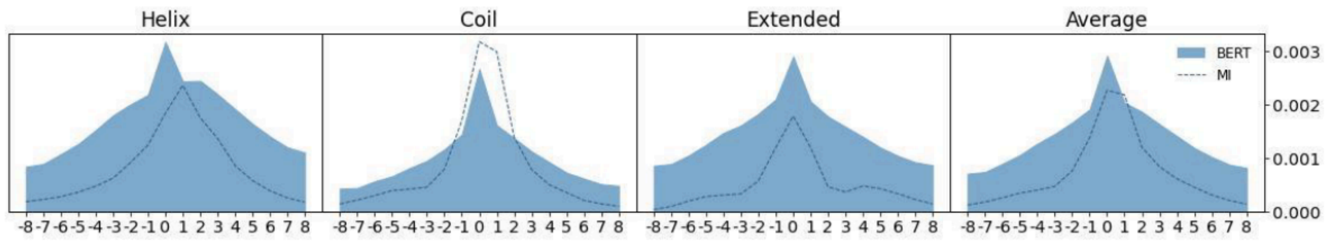

Fig. (12). DeepLIFT attributions of the BERT model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average.

3.5 T5 Neural Network (T5)

Finally, Fig. **(13)** reports the DeepLIFT attributions for the T5 model. The shapes of the attribution plots are very similar among each other, and behave similarly to what was seen for the BERT model, descending symmetrically on both sides. They all have a central peak at position 0-1, which is more pronounced for the coil class. Contrary to BERT, the attributions scores are high for all classes.

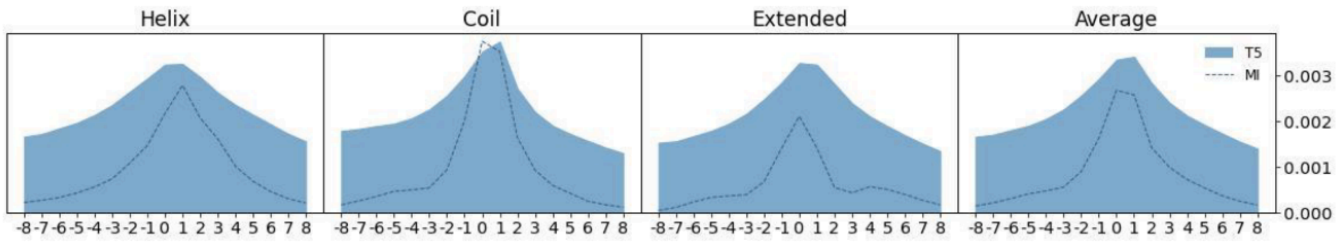

Fig. (13). DeepLIFT attributions of the T5 model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average.

3.6 Comparison of Input Features Effects

After we separately analyzed the way the models use their input features, we performed a quantitative evaluation of their similarity using the Pearson Correlation Coefficient [44], which measures the linear correlation between two sets of data.

We calculated the correlation between the weights/attribution scores of each pair of models and the position-state mutual information. For each pair of models we compare the concatenation of the three sets of 17 values computed for each of the output classes, in order to measure the similarities in the way the models treat each output state.

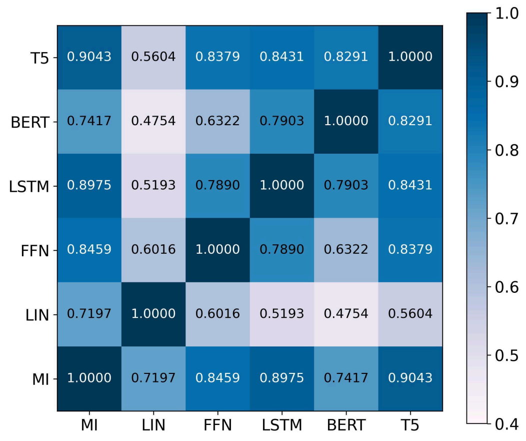

Fig. (14). Pearson Correlation coefficient between the models.

Fig. **(14)** shows the correlation matrix between all the models and the mutual information. The Pearson Correlation Coefficient confirms some of our previous observations: the models that most closely resemble the score distribution of the mutual information are the LSTM and T5, followed by the FFN, BERT and finally LIN.

The linear model LIN is the one which is least similar to the other models, due to the large difference in magnitude between the central peak and the tails, and also due to the high valu e of the peak for the Coil class (see Fig. **(9)**). However, it still has a high correlation with the mutual information (0.72) and the other linear model (0.60 FFN).

In general, the attribution scores of all the neural networks have high correlation (over 0.63) among each other, and T5 has a very high correlation with all the other models.

3.7 Performance and Comparison of the Predictive Models

We compared the performance of all the classification models on both the train and test datasets.

The performance is computed using accuracy (percentage of samples with the correct predicted secondary structure), but we also report the confusion matrices on the test set for a more in-depth analysis.

Looking at Table **1** we can observe that, for each model, train and test accuracy are really close. This indicates great generalization capabilities for all classifiers and no signs of overfitting.

Comparing the linear models (LIN and FFN) with the non-linear ones (LSTM, BERT, T5), we can see that the latter consistently outperform the former. The non-linear models surpass the linear ones of up to 8% in train and test accuracy. In particular, BERT proves to be the most accurate model, with a test accuracy of 70.15%, while the best linear model reaches 62.25%.

Interestingly, the test accuracy of the non-linear models is inversely correlated to their Pearson Coefficient with mutual information (bottom row in Fig. **(14)**). For example, BERT has the highest test accuracy, but its attribution scores were the least correlated ones with the mutual information. On the other hand, T5 has the highest correlation with mutual information, but it has the worst performance on the test set. This is supposedly because deep neural networks gain performance by capturing complex non-linear relationships between pairs of amino acids (or patterns in even higher dimensions), which leads to their attributions scores to look different from the two-variable mutual information, which only captures direct dependencies between a single amino acid and the output. As we observed in Fig. **(7)** with the three-variable mutual information, the information gained from pairs of positions is not negligible, so it is reasonable for non-linear models to exploit them for more accurate predictions.

Table 1. Performance evaluation of the analyzed models for protein secondary structure prediction on the training and test sets.

Model	**Train Accuracy**	**Test Accuracy**
LIN	62.62%	62.09%
FFN	62.72%	62.25%
LSTM	69.88%	69.15%
BERT	70.24%	70.15%
T5	69.04%	68.35%

To summarize:

- we can infer how accurate a linear model is based on how much their attribution score distribution resembles the mutual information. This is because the more they are similar to the mutual information, the more they are exploiting all linear dependencies between input and output.
- we can infer how accurate a non-linear model is based on how much their attribution score distribution differs from the mutual information. This is because the weights of the model need to deviate from that distribution to focus on higher-dimension correlations between different input positions. If the attributions of a non-linear model resemble mutual information, they are probably not extracting all the information out of the data.

Looking at the confusion matrices in Fig. **(15)**, we can see that the main difference in predictive power between linear and non-linear models comes from the classification of Extended structures.

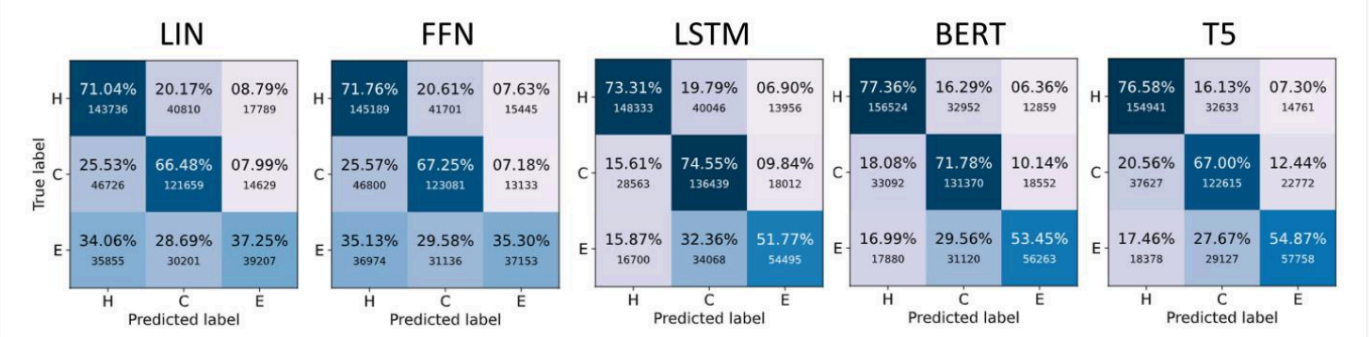

Fig. (15). Confusion matrices of the analyzed models. The first matrices refer to the linear models LIN and FFN, the following ones to the deep learning models LSTM, BERT and T5.

The confusion matrices of LIN and FFN show that only 35-37% of the Extended structures are correctly identified as such (lower right corner). The non-linear models improve their identification by 20% reaching 54.87% with T5. This shows that neural networks containing non-linearities are better-suited to encode the non-local interactions needed to identify Extended structures and distinguish them from Helices.

CONCLUSION

The secondary structure prediction problem is one of the classical problems in structural bioinformatics. In this work we have considered the prediction of secondary structure from a single sequence without using evolutionary information. In particular a stretch of 17 residues was considered to predict the secondary structure of the central residue.

Mutual information analysis shows that single positions and several pairs of positions can help determine the secondary structure of the central residue. The information that can be gained from single positions has a dominant role, however the effect of pairs of positions is often non-negligible. In particular, three-variable mutual information analysis showed that clear information gains can be achieved by considering pairs of residues jointly, instead of separately.

The linear models we tested perform very well as secondary structure predictors, because the information provided by single positions is the major contribution to the secondary structure of the central residue. The average of the absolute values of the linear coefficients ($\overline{c_j}, j \in [-8, +8]$), relative to each position parallels the mutual information.

These observations are in line with the accuracy of the linear model which is 62.09%, ca. 10% less than the best predictors available in the literature. However, linear models cannot exploit the joint effect of pairs (and higher order groups) of positions.

The additional mutual information that can be gained from many pairs of positions appears to be one order of magnitude smaller than the mutual information gained from single positions (e.g. two-variable mutual information peaks at 0.04 in Fig. **(6)**, while three-variable mutual information peaks around ±0.004 in Fig. **(7)**). Roughly speaking, non-linear models have access to 10% more information, which is quantitatively similar to the 10% increase in accuracy compared to the linear models.

Additional smaller gains in accuracy may stem from better network architecture using iteratively other predicted quantities like solvent accessibility and torsional angles as done by the SPIDER3 algorithm. We remark that the networks employed here use the information coded in the subsequence characters in a straightforward scheme.

All deep neural network based methods perform better than the linear model in accuracy. The positional effect on the predicted state is more spread over all positions, which is consistent with their ability to take into account context effects. The spread of the importance of features appears increasingly more pronounced in the LSTM, BERT and T5 networks, respectively. The BERT-based architecture achieves the best accuracy, and appears to maintain a weight on the central position that is apparent over the smoother background distribution. The best performance is likely due to balanced handling of local and context information.

This interpretation is consistent with the accuracy obtained on single secondary structure classes, which are determined mostly by local information (Helix and Coil) and non-local and correlated information (Extended). Indeed, the largest gain in accuracy (ca. 20%) provided by neural networks compared to linear models is for the Extended secondary structure. The accuracy on the latter class increases to 51.77%, 53.45% and 54.87% for LSTM, BERT and T5, respectively. Overall our analysis provides a frame to interpret different prediction models and to understand how mutual information between input and output data is reflected in the performance of such prediction models.

In conclusion, our findings suggest that the linear model might be more suited than non-linear ones to use the single position mutual information with the secondary structure of the central residue. However, non-linear models will always perform better than linear ones in accuracy, because they can exploit more information.

LIST OF ABBREVIATIONS

LSTM	– Long short term memory
FFN	– Feedforward networks
BERT	– Bidirectional Encoder Representations from Transformers
DeepLIFT	– Deep Learning Important FeaTures

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

Not applicable.

HUMAN AND ANIMAL RIGHTS

Research Involving Humans

Not applicable.

Research Involving Animals

Not applicable.

Research Involving Plants

Not applicable.

CONSENT FOR PUBLICATION

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

The data supporting the findings of the article is available as described by the Pisces server at http://dunbrack.fccc.edu/pisces/ [28]. The codebase used to implement and test the models described in the article is available in the GitHub repository ProteinStructure at https://github.com/AilabUdineGit/ProteinStructure.

FUNDING

Not applicable.

Conflict of Interest

Not applicable.

Acknowledgements

Not applicable.

REFERENCES

[1] Anfinsen, C. B. Principles that govern the folding of protein chains. Science 181, 223–230 (1973). doi: 10.1126/science.181.4096.223.

[2] Rost, B, Sander, C and Schneider, R. Redefining the goals of protein secondary structure prediction. J Mol Biol 235, 13–26 (1994). doi: 10.1016/s0022-2836(05)80007-5.

[3] Jumper, J et al. Highly accurate protein structure prediction with AlphaFold. Nature 596, 583–589 (2021). doi: 10.1038/s41586-021-03819-2.

[4] Zhou, Y and Karplus, M. Interpreting the folding kinetics of helical proteins. Nature 401, 400–403 (1999). doi: 10.1038/43937.

[5] Ozkan, S. B et al. Protein folding by zipping and assembly. Proc Natl Acad Sci USA 104, 11987–11992 (2007). doi: 10.1073/pnas.0703700104.

[6] Plaxco, K. W, Simons, K. T and Baker, D. Contact order, transition state placement and the refolding rates of single domain proteins. J Mol Biol 277, 985–994 (1998). doi: 10.1006/jmbi.1998.1645.

[7] Yang, Y et al. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? Brief Bioinformatics 19, 482–494 (2018). doi:10.1093/bib/bbw129

[8] Rost, B and Sander, C. Third generation prediction of secondary structures. In Protein Structure Prediction: Methods and Protocols, 71–95 (Humana Press, Totowa, NJ, 2000).

[9] Pauling, L and Corey, R. B. Configurations of polypeptide chains with favored orientations around single bonds: two new pleated sheets. Proc Natl Acad Sci USA 37, 729 (1951). doi: 10.1073/pnas.37.11.729.

[10] Pauling, L, Corey, R. B and Branson, H. R. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. Proc Natl Acad Sci USA 37, 205–211 (1951). doi: 10.1073/pnas.37.4.205.

[11] Chou, P. Y and Fasman, G. D. Prediction of protein conformation. Biochemistry 13, 222–245 (1974). doi: 10.1021/bi00699a002.

[12] Garnier, J, Osguthorpe, D. J and Robson, B. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. J Mol Biol 120, 97–120 (1978). doi: 10.1016/0022-2836(78)90297-8.

[13] Gibrat, J. F, Garnier, J and Robson, B. Further developments of protein secondary structure prediction using information theory. New parameters and consideration of residue pairs. J Mol Biol 198, 425–443 (1987). doi: 10.1016/0022-2836(87)90292-0.

[14] Garnier, J, Gibrat, J. F and Robson, B. GOR method for predicting protein secondary structure from amino acid sequence. Methods Enzymol 266, 540–553 (1996).doi: 10.1016/s0076-6879(96)66034-0.

[15] Rost, B. Review: Protein secondary structure prediction continues to rise. J Struct Biol 134, 204–218 (2001). doi: 10.1006/jsbi.2001.4336.

[16] Pollastri, G et al. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins: Struct Funct Bioinf 47, 228–235 (2002). doi: 10.1002/prot.10082.

[17] Torrisi, M, Pollastri, G and Le, Q. Deep learning methods in protein structure prediction. Comput Struct Biotechnol J 18, 1301–1310 (2020). doi: 10.1016/j.csbj.2019.12.011.

[18] Heffernan, R et al. Single-sequence-based prediction of protein secondary structures and solvent accessibility by deep whole-sequence learning. J Comput Chem 39, 2210–2216 (2018). doi: 10.1002/jcc.25534.

[19] Kotowski, K et al. Proteinunet—an efficient alternative to spider3-single for sequence-based prediction of protein secondary structures. J Comput Chem 42, 50–59 (2021). doi: 10.1002/jcc.26432.

[20] Shrikumar, A, Greenside, P and Kundaje, A. Learning important features through propagating activation differences. In 34th ICML (2017). URL:http://arxiv.org/abs/1704.02685

[21] Chowdhuri R. et al. Single-sequence protein structure prediction using a language model and deep learning. Nature Biotechnol. 40, 1617-1623 (2022) doi: 10.1038/s41587-022-01432-w

[22] Lei Z. et al. MO4: A many-objective evolutionary algorithm for protein structure prediction. IEEE Trans. Evol. Comput. 26, 417-430 (2022). doi:10.1109/TEVC.2021.3095481

[23] Rashid S., Surndaram, S. and Kwoh C. K. Empirical study of protein feature representation on deep belief networks trained with small data for secondary structure prediction. IEEE/ACM Trans. Comput. Biol. Bioinformatics, in press. doi:0.1109/TCBB.2022.3168676

[24] Hu L. et al. A distributed framework for large scale protein-protein interaction data analysis and prediction using MapReduce. IEEE/CAA J. Autom. Sinica. 9, 160-172 (2022). doi: 10.1109/JAS.2021.1004198.

[25] Beltagy, I, Peters, ME, Cohan, A Longformer: The Long-Document Transformer, arXiv 2004.05150 (2020). doi: 10.48550/arXiv.2004.05150

[26] Wu, H, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Advances in Neural Information Processing Systems 34 (2021): 22419-22430.

[27] Zhang, J, et al. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. International Conference on Machine Learning. PMLR 119, 2020.

[28] Wang, G and Dunbrack, R. L. PISCES: a protein sequence culling server. Bioinformatics 19, 1589–1591 (2003). doi: 10.1093/bioinformatics/btg224

[29] Rost, B PHD: Predicting one-dimensional protein structure by profile-based neural networks, Methods Enzymol, 266, 525-539 (1996). doi: 10.1016/S0076-6879(96)66033-9

[30] Touw, W. G et al. A series of PDB-related databanks for everyday needs. Nucl Acids Res 43, D364–D368 (2014). doi: 10.1093/nar/gku1028

[31] Kabsch, W and Sander, C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22, 2577–2637 (1983). doi: 10.1002/bip.360221211.

[32] Henikoff, S and Henikoff, J. G. Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci USA 89, 10915–10919 (1992). doi: 10.1073/pnas.89.22.10915.

[33] Heffernan, R et al. Improving prediction of secondary structure, local backbone angles and solvent accessible surface area of proteins by iterative deep learning. Sci Rep 5, 1–11 (2015). doi: 10.1038/srep11476.

[34] Heffernan, R et al. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. Bioinformatics 33, 2842–2849 (2017). doi: 10.1093/bioinformatics/btx218.

[35] Matsuda, H. Physical nature of higher-order mutual information: Intrinsic correlations and frustration. Phys. Rev. E 62, 3096–3102 (2000). doi: 10.1103/PhysRevE.62.3096

[36] Hochreiter, S and Schmidhuber, J. Long short-term memory. Neural Comput 9, 1735–1780 (1997). doi: 10.1162/neco.1997.9.8.1735

[37] Sibi, P, Jones, S. A and Siddarth, P. Analysis of different activation functions using back propagation neural networks. J Theor Appl Inf Technol 47, 1264–1268 (2013). URL:https://www.jatit.org/volumes/Vol47No3/61Vol47No3.pdf

[38] Devlin, J et al. Bert: Pre-training of deep bidirectional transformers for language understanding (2019). 1810.04805. doi: 10.18653/v1/n19-1423

[39] Gu, Y et al. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. ACM Trans Comput Healthc 3, 1–23 (2020). doi: 10.1145/3458754.

[40] Chalkidis, I et al. LEGAL-BERT: The muppets straight out of law school. In Findings of the Association for Computational Linguistics: EMNLP 2020, 2898–2904 (Association for Computational Linguistics, Online, 2020). doi: https://doi.org/10.48550/arXiv.2010.02559

[41] Feng, Z et al. CodeBERT: A pre-trained model for programming and natural languages. In Findings of the Association for Computational Linguistics: EMNLP 2020, 1536–1547 (Association for Computational Linguistics, Online, 2020). doi: 10.18653/v1/2020.findings-emnlp.139.

[42] Raffel, C et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res 21, 1–67 (2020). https://doi.org/10.48550/arXiv.1910.10683

[43] Paszke, A et al. Automatic differentiation in pytorch. In NIPS 2017 Workshop on Autodiff (2017). URL https://openreview.net/forum?id=BJJsrmfCZ.

[44] Benesty, J et al. Pearson correlation coefficient. In Noise reduction in speech processing, 1–4 (Springer, 2009).