IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 31, 2021, accepted February 2, 2022, date of publication March 8, 2022, date of current version March 16, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3156888

Solving the Protein Secondary Structure Prediction Problem With the Hessian Free Optimization Algorithm

KONSTANTINOS CHARALAMPOUS[1], MICHALIS AGATHOCLEOUS[1,2], (Member, IEEE), CHRIS CHRISTODOULOU[1], AND VASILIS PROMPONAS[3]

[1]Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
[2]Department of Computer Science, University of Nicosia, 1700 Nicosia, Cyprus
[3]Department of Biological Sciences, University of Cyprus, 1678 Nicosia, Cyprus

Corresponding author: Chris Christodoulou (cchrist@cs.ucy.ac.cy)

ABSTRACT

Trying to extract features from complex sequential data for classification and prediction problems is an extremely difficult task. This task is even more challenging when both the upstream and downstream information of a time-series is important to process the sequence at a specific time-step. One typical problem which falls in this category is Protein Secondary Structure Prediction (PSSP). Recurrent Neural Networks (RNNs) have been successful in handling sequential data. These methods are demanding in terms of time and space efficiency. On the other hand, simple Feed-Forward Neural Networks (FFNNs) can be trained really fast with the Backpropagation algorithm, but in practice they give poor results in this category of problems. The Hessian Free Optimization (HFO) algorithm is one of the latest developments in the field of Artificial Neural Network (ANN) training algorithms which can converge faster and more accurately. In this paper, we present the implementation of simple FFNNs trained with the powerful HFO second-order learning algorithm for the PSSP problem. In our approach, a single FFNN trained with the HFO learning algorithm can achieve an approximately 79.6% per residue (Q_3) accuracy on the PISCES dataset. Despite the simplicity of our method, the results are comparable to some of the state of the art methods which have been designed for this problem. A majority voting ensemble method and filtering with Support Vector Machines have also been applied, which increase our results to 80.4% per residue (Q_3) accuracy. Finally, our method has been tested on the CASP13 independent dataset and achieved 78.14% per residue (Q_3) accuracy. Moreover, the HFO does not require tuning of any parameters which makes training much faster than other state of the art methods, a very important feature with big datasets and facilitates fast training of FFNN ensembles.

INDEX TERMS Hessian free optimization, neural networks, protein secondary structure prediction, second order learning algorithms.

I. INTRODUCTION

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that provides a system the ability to learn from data without being explicitly programmed. Usually, ML methods consist of a parameterized learnable model and a learning algorithm [1], [2]. These methods are mostly used for classification and prediction on static and sequential data.

Even though a number of theoretical ML algorithms have been designed to process and make predictions on sequential data, the mining of such data types is still an open field of research due to its complexity. Analysis and development of optimisation algorithms for specific ML techniques for sequencial data must take into account how to (a) capture and exploit sequential correlations, (b) represent and incorporate loss functions, (c) identify long-distance dependencies, and (d) make the optimisation algorithm fast [3]. Successful ANN models which have been designed to handle sequencial data are the classes of Recurrent Neural Networks (RNNs) [4] and Deep Learning (DL) methods [5]. More specifically, RNNs are universal approximators of dynamical systems [6] that can associate patterns which are located far away from each

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

other on a sequence and create a kind of short and long range dependencies between data. Such models are Long-Short Term Memory (LSTM) blocks [7], [8], Echo State Networks (ESNs) [9]–[11], Conceptors [12] and Clockwork RNNs (CW-RNN) [13] as architectural improvements of RNNs. On the other hand, DL and more specifically Convolutional Neural Networks (CNNs) [5] have shown superior results in various sequential problems [14]–[16]. All these methods are new established models in the ML field which can build high quality feature extraction procedures and achieve high-performance classification of sequential data.

Moreover, predictions on sequential data are particularly challenging when both the upstream and downstream information of a sequence is important for a specific time-step. Application examples include Phoneme Speech Recognition [7], [17] and Bioinformatics problems, such as the Protein Secondary Structure Prediction (PSSP) [18]–[20] and other related problems (e.g., Transmembrane Protein Topology Prediction [21]). In such sequence-based problems the events are dynamic and located downstream and upstream, i.e., left and right in the sequence. To predict these events, researchers have used many ANN models but also utilise specific ANN architectures, such as Bidirectional Recurrent Neural Networks (BRNNs) [18], [19], [22], [23] which are usually trained with an extension of the Back-propagation Through Time (BPTT) algorithm or with second order learning algorithms [24]. In general, a ML model designed for such data has to be in position to extract relevant features, and at the same time reveal any long/short range interdependencies in the sequence of data given.

Knowledge of a protein's secondary structure is important as a step towards elucidation of its 3D structure, which is crucial to understand its function. This problem is even more timely since the rapid growth of the number of available protein sequences has far outpaced the experimental determination of their structures. Thus, there is a growing need for a computational approach to the problem of protein structure prediction. The prediction of Secondary Structure (SS), i.e., the local structure commonly defined by hydrogen bond patterns and local geometry, is a critical first step towards this end and, therefore, it has attracted a large amount of interest over the past 60 years [25]. With respect to their secondary structure, amino acid residues in protein chains are usually assigned into three main classes, namely helix, extended and coil/loop [26]. Feed-Forward Neural Networks (FFNNs) trained with the Gradient Descent (GD) algorithm were firstly used to tackle this problem but the accuracy was less than 70% [27], [28]. Ensembles of BRNNs trained with the GD algorithm have proved to be a very efficient method for the PSSP problem with accuracy of approximately of 76%-79% [18], [23], [29]. Even though many ML algorithms and methodologies have been employed to tackle this problem [18], [25], [30]–[36], only a few of them will be discussed in this paper, which are closely related with the methods used in the present study.

In addition, DL and LSTM-BRNN methods fed with sequence profile inputs have achieved around 80%-84% per residue accuracy [32], [37]–[43]. However, considering the theoretical limit of the three-state prediction which is around 88-90% [25], there is still room for improvement. These results are even higher (approximately 92% per residue accuracy) when the observed secondary structure from homologs in the Protein Data Bank (PDB; http://www.rcsb.org/) is used. The major key point that needs to be considered when trying to solve the PSSP problem is the complex sequence correlations and interactions between the amino acid residues of a protein molecule. In order to maximize the prediction accuracy of a proposed ANN technique for a specific amino acid in a protein molecule, the adjacent amino acids have to be considered by the proposed ANN architecture.

Training RNNs is a demanding task in terms of time and space efficiency because of the compexity of the models where many parameters have to be handled in order to specify the model's architecture and characteristics. Furthermore, training of these models is a difficult task because of the vanishing gradient and exploding gradient problems, where the gradient is getting smaller as the information moving backward through hidden layers is getting very big at the early layers of a model [44]. The most common algorithm to train these models is based on the GD minimization method. Unfortunately, this kind of algorithms have a poor convergence rate [45]. Moreover, they depend on parameters which have to be specified by the user and are usually crucial for the performance of the algorithm.

In order to improve these drawbacks, simple models and more efficient algorithms need to be used. Simple FFNNs can be trained really fast with low demands in terms of time and space. The most common learning algorithm for training FFNNs is the Backpropagation learning algorithm (BP) [46] which is based on the GD method. Unfortunatelly, it has been proven that FFNNs have poor results in this kind of problems [27], [28]. On the other hand, latest developments in the field of ANN training algorithms, such as the Hessian Free Optimization (HFO) [47], [48] second order learning algorithm, can converge faster and more accurately. This algorithm has been found to be superior to the conventional BP algorithm in terms of accuracy, convergence rate and the vanishing-gradient problem [47], [48]. In addition, the original form of the algorithm [47], [48] does not depend on any parameters.

The HFO [47], [48] second order learning algorithm demonstrated promising results on problems such as the noiseless memorization problem, the 3-bit temporal order problem and the random permutation problem [48]. In this algorithm, the finite differences method is applied on the model's gradient vector in order to quickly calculate an approximation of the Hessian Matrix. The system's gradient vector and Hessian Matrix approximation are used to compute the system's Taylor expansion function. Then, the Preconditioned Conjugate Gradient algorithm (PCG) [48], which is a variant of the CG algorithm, is used to optimize

the calculated Taylor expansion function. The PCG algorithm is used to give a new step size and a direction to update the system's parameters. Moreover, it has been proven that this algorithm, which was based on the works of [45], [49], [50] and [51], can manage well to train RNNs. Consequently, the HFO has been applied to deep ANN architectures and it was demonstrated that it outperforms other learning algorithms in specific sequential problems [48].

The need to build efficient methods in terms of accuracy, convergence time and simplicity for sequential data where both the upstream and downstream information of a sequence is important for a specific time-step and more specifically for the PSSP problem, has been the initial motivation for this work. Consequently, in this paper we demonstrate that providing profiles of protein sequences for training simple FFNNs with the HFO algorithm yields results which compare well with much more complicated ANN architectures in the 3-state PSSP problem. Furthermore, we demonstrate that a simple majority vote of 5 HFO-trained FFNNs, coupled with a SVM classifier for filtering purposes, achieves performance directly comparable to the current state-of-the-art methods.

The paper is organized as follows: *Section II* covers the PSSP application domain, the related data and existing methods, while *Section III* presents our methodology based on the HFO learning algorithm. *Section IV* presents results, discussion and comparison of our methodology with other methods for the PSSP problem. Finally, *Section V* gives the conclusions and future directions for research in the field.

II. APPLICATION DOMAINS, EXISTING METHODS, AND DATA

The prediction of a protein's SS from its Primary Structure (PS) can be an important intermediate step to the prediction of a protein's three-dimensional (3D) structure [52], [53]. A protein's PS is a sequence composed of 20 different amino acid types which are connected and interact to create the SS, the local (geometrical) structural patterns defined by hydrogen bonding patterns which may be short, mid- or even long-range. When an experimentally-determined 3D structure is available, each amino acid can be assigned to a SS class, usually under a commonly accepted scheme: helix (H), extended (E) and coil/loops (L) [26].

Knowledge of a protein's three-dimensional (3D) structure can be an important step in studying the functional properties of protein molecules, which are the functional workhorses in all living cells. Experimental biochemical methods for the characterization of the molecular structures of individual proteins in atomic detail are expensive, time consuming and frequently inefficient [18]. Since genomic technologies provide genetic sequences at an ever increasing pace, the gap between our knowledge of protein sequences (primary structures) and the corresponding experimentally determined 3D structures is widening exponentially. Even though it has been reported that a high fraction of residues from proteins encoded in the human genome and other model species can be mapped to a 3D structure (either by experimental or theoretical methods) [54], the same does not necessarily hold for non-model species, especially for some "exotic" pathogens, whose genomes often encode protein molecules with peculiar sequence features. For example, of the >5000 proteins encoded in the malaria-causing parasite *Plasmodium falciparum isolate 3D7* for which no experimental structural information exists in the Protein Databank, less than half (2459) can be mapped onto 3D structures using sequence similarity methods (source UniProt: https://www.uniprot.org/uniprot/accessed May 9 2019).

Over the past 30 years, the accuracy for secondary structure predictive methodologies has improved significantly with machine learning techniques [27] and evolutionary information from multiple sequence alignments [28] having a crucial role. These methods are commonly evaluated based on the per residue accuracy (Q3) and the segment overlap (SOV) metrics [55], [56]. The Q3 metric is defined as the three-state overall percentage of correctly predicted residues. On the other hand, the SOV metric is defined as a measure that is based on the average overlap between the observed and the predicted segments instead of the average per-residue accuracy [55], [56]. A brief review of these methodologies can be found in [57]. Furthermore, a more recent review presents all the latest developments in the field of PSSP which have achieved more than 80% per residue accuracy [25]. These results rely on increasingly larger databases of protein sequences, the use of templates and powerful deep learning models. More specifically, a method based on Deep Convolutional Neural Fields (based on CNNs) [33] and the MUFold-SS method [31] have shown promising results. In addition, the work of [43], which is based on CNNs, has achieved a noticeable accuracy above 80%. Finally, the work of [32], which is based on LSTM [8] BRNNs has produced an 84% Q3 accuracy which is one of the highest scores reported so far [25]. Current state-of-the-art methods for predicting protein SS from PS are based on ML classifiers fed with sequence profile inputs and achieve around 82%-84% Q3 accuracy, whereas the SOV is circa 73-75. Finally, many kernel methods have demonstrated noticeable results for this demanding problem which are approaching 80% accuracy [58], [59].

Pollastri and colleagues [23] have shown results on the SSpro method where they have used an ensemble of 11 BRNNs to achieve a prediction accuracy of 78%. Moreover, the SCRATCH server uses the SSpro 5 method trained on newer datasets to achieve an accuracy of 79% [29], [60]. This method has been trained and validated on a large dataset of approximately 11000 sequence profiles, using an elaborate training process where individual BRNNs were trained on different subsets of the training data. Furthermore, this method uses an ensemble of 100 BRNNs. With SSpro 5 [60] an additional step of using the observed secondary structure from homologs in the PDB to infer the final prediction was introduced, reaching an accuracy of approximately 92%.

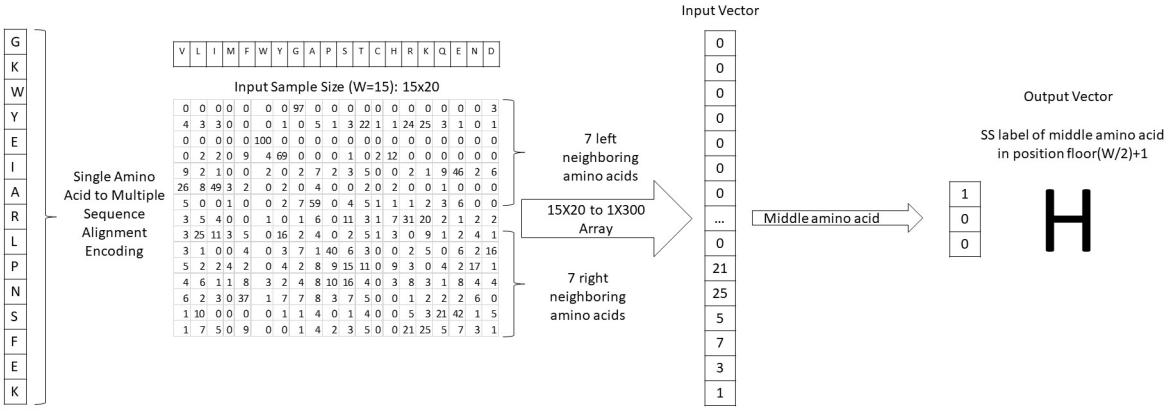

FIGURE 1. Data Representation: Each amino acid is within a window (*W*) centered at the residue of interest, encoded by a 20-dimensional MSA profile vector.

Furthermore, several PSSP prediction methodologies have used a multi-step process with ensemble methods [61]–[63] and filtering techniques [20] to improve the quality of results. Ensembles reduce the mis-predicted residues by averaging the results of multiple classifiers. For example, these methods show improved generalization capabilities that outperform those of single networks [64]. However, for aggregation to be effective, the individual networks must be as accurate and diverse as possible. In addition, filtering techniques remove conformations that are physicochemically unlikely. For instance, helical conformations in proteins usually consist of at least three, four or five residues for 3_{10}-helix, α-helix and π-helix, respectively. Since the different types of helices are usually grouped in a single category by PSSP methods, a predicted helical structure would be expected to have a minimum number of three consecutive residues in order to fulfill geometric and hydrogen-bonding requirements. Hence, predictions of isolated helical residues are physicochemically unrealistic, because one residue cannot form a helix. To tackle this problem, both machine learning algorithms [65] and empirical rules [28], [35], [66] have been used in the past with variable levels of success, highlighted in a comparative study [20].

High quality datasets for training and validation purposes are mandatory when constructing a prediction model. For the purpose of this paper, we have used the non-redundant PISCES dataset [67], as reported previously [68].

III. METHODOLOGY

A. DATA REPRESENTATION

The major obstacle on trying to solve a complex sequential data classification problem with any ANN is the representation of the data, in such a way that the network is able not only to understand the shape of the input volume, but also to track the complex sequence correlations among the input volume. Hence, multiple sequence alignment (MSA) profiles have been used for data preprocessing and PS encoding [69]. MSA profiles have been shown to enhance machine learning-based PSSP, since they incorporate useful evolutionary information for the encoding of each position of a protein. More specifically, each amino acid on a protein PS is replaced by a 20-dimensional vector, which corresponds to the frequencies of 20 different amino acid types aligned to the query sequence using PSI-BLAST (Version 2.4.0) [70] searches (maximum 3 iterations) against the NCBI-NR (NCBI: http://www.ncbi.nlm.nih.gov/) database.

As shown in *Figure* 1, the network's input vector at each time-step t consists of the MSA information contained in a sliding window W on a protein PS. More specifically, we have created an input volume by placing MSA [71] profile vectors of each amino acid one after another to construct an 1D representation of the MSA profiles of a certain number of neighboring amino acid residues. Through this technique, the attention given to any neighboring amino acid correlations is equally weighted across all the input volume, for each W given. This lets the FFNN discover and capture any strong short range correlations among the input records and consider them all equally in terms of the output volume created. The target output is the respective class of the amino acid which is located in the center of a specific input W. The *FRNN* processes simultaneously the residues located on the left and on the right side of the position t to predict the corresponding SS class.

B. THE HFO LEARNING ALGORITHM

HFO [47], [48] is a second order optimization algorithm of real-valued objective functions. It is a variation of the standard Newton's method [72] which uses local quadratic approximations to generate update proposals. In high dimensionality problems, for which large ANN models with many hidden layers are employed, first order optimization algorithms like GD can be extremely slow and ineffective due to the vanishing gradient problem [8]. In GD methods, the updates are proportional to the gradient of the error function which is back-propagated through the layers of the model. Each time the error is back-propagated, the gradient becomes vanishingly small which results in the front layers having

close to zero information on how to update their weights, meaning slow to completely ineffective training [47], [48].

The advantage of using a second order optimization algorithm (i.e., Newton's method or HFO) is that these algorithms consider the curvature of the error surface (Hessian Matrix (HM)) in their optimization process which results in extremely better step-wise performance. More specifically, instead of fitting a plane at an initial solution and then determining the step-wise jump like first order algorithms, second order methods find a tightly fitting quadratic curve at that point and directly find the minimum of that curvature, which is supremely fast and efficient. Computing the HM for a large ANN with thousands to millions of free parameters however is not always possible due to the extremely high memory requirements needed to store it. Concequently, while there have been a number of Newton's variations like Newton-CG, CG-Steihaug, Newton-Lanczos [73] and Truncated Newton [74], none of them has been applied effectively to ANN models and concequently their applications in this domain have been extremely limited [48].

The Hessian Free method proposes solutions to the high memory requirements of second order learning algorithms, which enable it to be effective for ANN training. First of all, it does not compute and store the whole HM. Instead, it computes just the dot product of the HM H with an arbitrary vector u ($H \cdot u$) [48], using mathematical methods like finite differences which cost as much as a single gradient evaluation. This works really well for the HFO algorithm since it does not require the explicit use of the HM, but rather many dot products based on $H \cdot u$. Secondly, the local quadratic objectives, which are approximated with second-order methods, can be efficiently optimized using the linear conjugate gradient (CG) method [75], [76] in order to compensate for the lack of the HM. While the CG method needs N iterations to converge in a quadratic function, where as N is the number of the free parameters of the network, there is a number of stopping criteria, which terminate it at early stages when significant progress in the minimization process has been made. This is extremely important since it is clearly impractical to wait for a complete CG convergence when there is a very low margin of further minimization.

It is important to note that even though in HFO no HM is calculated, there are no approximations done and the $H \cdot u$ product is calculated accurately. The only difference between HFO and Newton's method is that while the standard Newton's method performs a complete optimization of the approximated quadratic information, HFO does not. This is because the CG does not fully converge [47]. However, the efficiency-related benefits of avoiding the full HM calculation and inversion are clearly more beneficiary than the extremely small difference in accuracy by the not fully converged CG.

Finally, although the $H \cdot u$ product can be calculated efficiently and accurately, it is not the one usually used in HFO. Based on the same theory, the $G \cdot u$ product is used, where G is an approximation of the HM which is called the Gauss-Newton matrix [77]. While it seems pointless to use an approximation instead of the correct curvature matrix when there is no problem in efficiency, Gauss-Newton avoids some of the problems that the HM may face and cause the algorithm to be completely ineffective. In fact, comparing to the usage of the HM, the use of the $G \cdot u$ matrix consistently results in better search directions utilizing half the memory and running twice as fast [47].

C. THE PROPOSED METHODOLOGY

In this work, the HFO learning algorithm has been used for training FFNNs to handle the PSSP problem. Our methodology appears in *Figure* 2. As it can be seen, we have 5 levels of processing. In the first level of our methodology, the protein PS appears in MSA encoding (*Figure* 1). Input data is used from 5 individual FFNNs in the second level of our methodology. The small number of FFNNs has been chosen based on the work of [18] and [20]. These FFNN classifiers are trained with the HFO learning algorithm. Each one of the adequate number of 5 FFNN returns three real values in the range (0,1) for the central residue of a sliding local window W, one for each secondary structure state. Subsequently, the corresponding outputs of each FFNN for each state are combined through majority voiting in the ensembles level [61], [62]. Then, the resulting predictions from the ensembles level are used for the filtering [20] procedure. In the case of the PSSP problem, the filtering method which appears to have the best results is based on SVM models [20]. Consequently, we have created a stacked network architecture where an SVM model is used for filtering purposes. More specifically, the predicted SS sequence of the majority voting ensemble method is used as input to the SVM model. Each predicted SS state is within a window (W_{svm}) centered at the position t of the residue of interest. For each position t of the sequence, this window is used as an input vector to the SVM which predicts the corresponding corrected SS state. Finally, the last level of our methodology returns three real values, which represent the predicted SS class for a specific input vector of a local window.

IV. RESULTS AND DISCUSSION

Our proposed methodology has been applied to the PSSP problem and through experimental analysis, various results have been extracted. The models and learning algorithms of our methodology have been built with the Python 3.6 programming language. The final results demonstrate the efficiency and the effectiveness of the method.

A. DATA PREPARATION AND SIMULATION DETAILS

Special care has been taken to retrieve datasets of the highest possible quality for the PSSP problem relying in specialized resources. For the purposes of this work, we have used the PISCES [68] dataset, which consists of 8632 protein chains. More specifically, high resolution protein structural

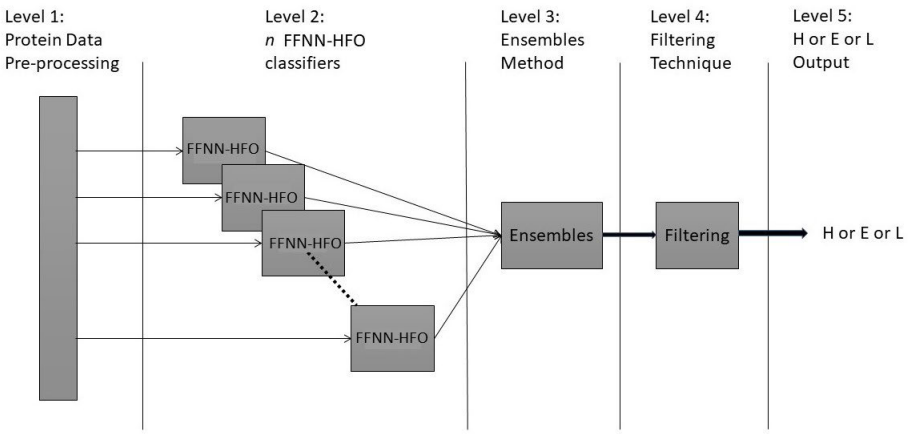

FIGURE 2. Our proposed methodology for the PSSP problem (see text for more details).

data have been obtained from the RCSB Protein Data Bank (PDB; http://www.rcsb.org/). We have used the DSSP program (URL: http://swift.cmbi.ru.nl/gv/dssp/, accessed 01/04/2019), to extract the SS class for each amino acid in each dataset. The DSSP program uses the atomic coordinates and hydrogen bond patterns for assigning each amino acid in one of eight classes: H (α-helix), G (3_{10}-helix), I (π-helix), E (extended β-strand), B (isolated β-bridge), T (turn), S (bend) and C (other/coil). Then, we have reduced the eight classes to the three predefined SS classes as: H, G, and I to the helix state (H), E and B to the extended state (E) and the rest to the loop state (L). Moreover, MSA profiles have been used for data preprocessing and PS encoding. During this procedure, the MSA files of our dataset have been analyzed and cleaned up from data with short or no information. More specifically, the PSI-BLAST has created some MSA files with arrays of zeros or arrays with less amount of elements than expected. The respective protein sequences (342 in total) have been excluded from the training and validation dataset (Table 6 in Appendix). Furthermore, we have followed a strict 5-fold cross-validation approach as described in [68].

As it has already been explained in *Section III*, the network's input vector at each time-step t consists of the MSA information contained in a local sliding window W and the target output is the respective class of the amino acid which is located in the center of that local window. Firstly, a single FFNN has been trained on different single folds of the PISCES dataset. At this stage, we carried out multiple experiments to tune up the architectural parameters of the FFNN trained with the HFO learning algorithm. More specifically, we have concluded that, for the purposes of our methodology, the optimum FFNN architecture is 1 hidden layer FFNN of 75 Sigmoid neurons with Mean Square Error (MSE) function. Furthermore, we have used 3 neurons of this category for the output layer. This architecture has full connectivity between input-hidden and hidden-output layers. The only difference between the multiple FFNNs is the random weight initialization. These models have been evaluated based on the Q3 and SOV metrics, but only the Q3 metric was used for tuning. Then, this specific FFNN architecture has been successfully used for our proposed methodology (*Section III*). For our experiments, a machine with 6 cores and 32GB RAM of a CISCO UCS C240 M5 server (2 x Intel Xeon 6140 Gold Processors) has been used.

In this work, the PISCES dataset has been used as training and validation set for tuning the model's parameters. Finally, the CASP13 (13th Critical Assessment of Protein Structure) dataset has been used as an independent dataset to test the prediction capabilities of the final models. This dataset consists of only 40 proteins with no significant sequence similarity to the proteins in the PISCES dataset.

B. CROSS-VALIDATION AND TESTING SIMULATIONS

In order to validate the robustness of our proposed methodology as well as to prove its efficiency to the exposure of various training and validation data, we had to complete the evaluation of the PSSP problem on the PISCES dataset, using a 5-fold cross-validation test. All the experiments conducted are with the optimal parameters of the model which is described in detail in *Section IV*(*A*). In *Table* 1, we present the results of a single FFNN with the optimal parameters after we manage to get the highest possible Q_3 score. As shown in *Table* 1, the Q_3 accuracy and SOV score results of a single FFNN with a 5-fold cross-validation are 79.57% Q_3 and 0.728 SOV respectively. These results, compared to the results of other methods which are mentioned in *Section II*, are high enough to be considered as a good solution for the PSSP problem. Given the simplicity of the model, we can say that the learning algorithm is powerful enough to optimize the problem and relate the neighboring amino acids to a SS based on the strong information which is coming from local dependencies of a protein PS. Furthermore, as it can be seen from *Table* 1, our method can predict better the H and L classes, whereas some difficulty is shown in the prediction of E class.

TABLE 1. Experimental results for a single FFNN trained with HFO: Q3 and SOV results for each fold of PISCES dataset.

$Fold$	$Q_3(\%)$	$Q_H(\%)$	$Q_E(\%)$	$Q_L(\%)$	SOV	SOV_H	SOV_E	SOV_L
0	79.43	78.32	74.09	79.04	0.7172	0.7365	0.7488	0.6957
1	79.38	77.28	73.51	79.62	0.7226	0.7278	0.7447	0.7055
2	79.57	78.05	72.84	79.45	0.7180	0.7319	0.7360	0.6994
3	79.87	78.63	74.29	79.80	0.7229	0.7398	0.7500	0.7056
4	79.58	79.18	73.64	79.81	0.7205	0.7470	0.7437	0.7027
Average	**79.57**	78.29	73.67	79.54	**0.7202**	0.7366	0.7446	0.7018

TABLE 2. Experimental results for an ensemble of 5 FFNNs trained with HFO: Q3 and SOV results for each fold of PISCES dataset.

$Fold$	$Q_3(\%)$	$Q_H(\%)$	$Q_E(\%)$	$Q_L(\%)$	SOV	SOV_H	SOV_E	SOV_L
0	80.02	77.28	73.58	81.80	0.7396	0.7464	0.7605	0.7214
1	80.11	77.19	73.69	81.44	0.7560	0.7560	0.7624	0.7323
2	80.19	76.83	72.22	82.33	0.7409	0.7395	0.7474	0.7284
3	80.40	77.32	73.62	82.60	0.7468	0.7495	0.7618	0.7317
4	80.12	78.08	73.07	82.58	0.7460	0.7598	0.7563	0.7314
Average	**80.17**	77.35	73.24	82.15	**0.7459**	0.7503	0.7577	0.7291

TABLE 3. Experimental results for an ensemble of 5 FFNNs trained with HFO and filtered with a SVM: Q3 and SOV results for each fold of PISCES dataset.

$Fold$	$Q_3(\%)$	$Q_H(\%)$	$Q_E(\%)$	$Q_L(\%)$	SOV	SOV_H	SOV_E	SOV_L
0	80.24	77.75	73.30	81.78	0.7641	0.7800	0.7642	0.7313
1	80.27	77.05	73.62	81.71	0.7682	0.7684	0.7670	0.7369
2	80.43	77.28	71.80	82.45	0.7653	0.7725	0.7504	0.7374
3	80.55	77.64	73.41	82.35	0.7699	0.7818	0.7659	0.7357
4	80.33	78.51	72.84	82.57	0.7683	0.7912	0.7598	0.7390
Average	**80.37**	77.65	72.99	82.17	**0.7671**	0.7788	0.7615	0.7361

A major improvement in the results of single FFNNs trained with the HFO algorithm has been achieved when the majority voting ensemble method and filtering techniques [20] have been used. In this paper, we employ an ensemble of 5 FFNNs as described in *Section III*(C). The results of the majority voting ensemble method are shown in *Table* 2. Clearly, this method corrects some missclassified SS states, thus increasing Q3 by 0.55% and SOV by 0.0234 overall. Based on these results, the majority voting ensemble method improves the method's accuracy. Moreover, based on the SOV results, it improves significantly the quality of the predicted SS sequence.

The resulting predictions are then used for SVM filtering, as explained in [20]. More specifically, after gathering the predictions from the majority voting ensemble method of FFNNs, we have trained a SVM using a window of SS states predicted by the FFNNs. In this case, we have used a 5-fold cross-validation approach based on the folds which have been used for the training of our FFNNs. After performing several experiments using different kernels, misclassification penalty parameters (C) [78], Gamma values (G) [78] and window sizes (W_{svm}), we have decided on the optimal SVM parameters that lead to the highest Q_3 accuracy and SOV score on the PSSP problem and which are: (a) Kernel: Radial Basis Function, (b) $C = 1$, (c) $G = 0.001$ and (d) $W_{svm} = 7$. The final results are shown in *Table* 3. The final results of our methodology are approximately 80.4% Q_3 accuracy and 0.77 SOV for the PISCES dataset. More specifically, the SVM filtering technique can increase the accuracy of our proposed methodology by only 0.2% but it can improve significantly the quality of results by approximately 0.02 SOV units.

TABLE 4. FFNN ensembles and SVM filtering: Statistical analysis of Q3 accuracy and SOV score for fold 0-4 results presented in *Table* 3.

	Q3(%)	**SOV**
Sample Standard Deviation (s)	0.1268	0.0023
Variance (Sample Standard) (s^2)	0.0160	5.678E-6
Mean (Average)	80.37	0.7671
Standard Error of the Mean ($SE_{\bar{X}}$)	0.0567	0.0010

As it can be seen in *Tables* 1−3, the ensemble methods and filtering techniques which were applied to our methodology have increased the single FFNN accuracy by approximately 0.8% and the SOV metric by 0.05. Furthermore, in *Figure* 3 we can see how the algorithm can manage with each SS class. Obviously, in the case of the PISCES dataset, the algorithm can predict correctly with 84.41%, 70.60% and 83.61% the H, E and L classes, respectively. As it can be observed, the H and L classes are most often predicted as such compared to the E class, a known shortcoming when applying FFNNs in the PSSP problem. Moreover, in *Table* 4, we can see that the values of Standard Deviation and Variance for the case of the Q_3 metric are very small which indicates that the algorithm

TABLE 5. Results for CASP13 dataset after training on PISCES dataset.

	Q3(%)	SOV
Single FFNN trained with HFO	76.9	0.714
Ensemble of 5 FFNNs trained with HFO	77.67	0.732
Ensemble of 5 FFNNs trained with HFO and filtered with a SVM	78.14	0.755

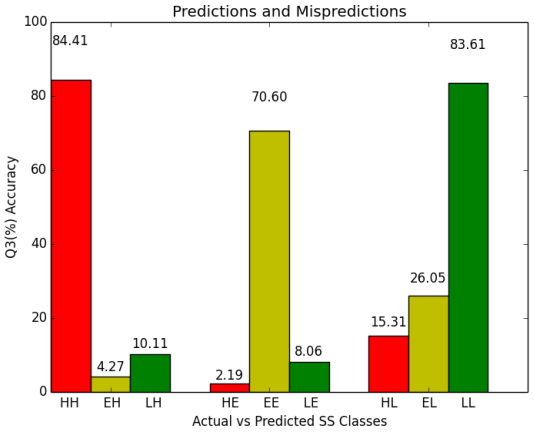

FIGURE 3. Graph of confusion matrix for actual vs predicted SS classes: HH, EE and LL are the True Positive scores of our method for each class after the majority voting ensemble method and SVM filtering technique. EH, LH, HE, LE, HL and EL are the scores for the mis-predicted classes where the first letter is the actual class and the second letter is the predicted class. Based on an average of a 5-fold cross validation evaluation, the method can predict correctly with 84.41%, 70.60% and 83.61% the H, E and L classes, respectively.

behaves similarly between the different protein sequences of several dataset folds.

Finally, we have used the CASP13 independent dataset to test the accuracy of our method. More specifically, we have trained the FFNNs on the whole PISCES dataset and then we have used CASP13 as testing dataset. The results can be seen on *Table* 5. The final accuracy of the methodology on CASP13 dataset is 78.14% and the SOV metric is 0.755.

C. COMPARISON OF OUR METHODOLOGY TO OTHER METHODS

Finally, in order to evaluate the accuracy, quality of results and computational performance of our method, we have implemented other well known methods to compare their results with our method using the same data. More specifically, we have used the same 5-fold cross validation approach with the PISCES dataset. Hence, we have used two different strategies. At first we have used other optimizers to train and compare the ensemble of 5 FFNNs. Then, we have tested (using the same 5-fold cross validation procedure on our dataset) well established ML methods that have been used on the PSSP.

The optimizers which were chosen to train the ensemble of FFNNs on the PISCES dataset were the BP [46] learning algorithm, as the most common benchmark training algorithm in training FFNNs, and the Adam [79] learning algorithm, as one of the latest and most powerful developments in the field of training algorithms for ANNs. Compared to

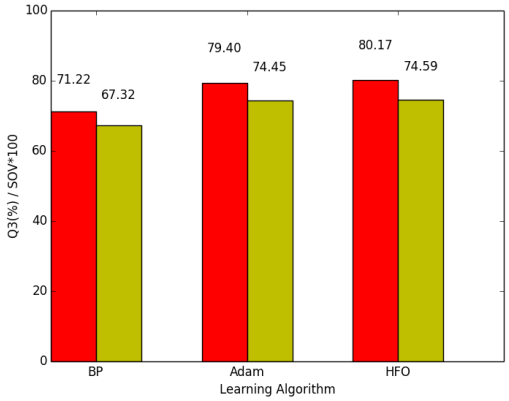

FIGURE 4. Q3 accuracy and SOV score for each FFNN optimizer. The red bar corresponds to Q_3 accuracy and the yellow bar to SOV score. The SOV score has been multiplied by 100 for presentation purposes (see text for more details).

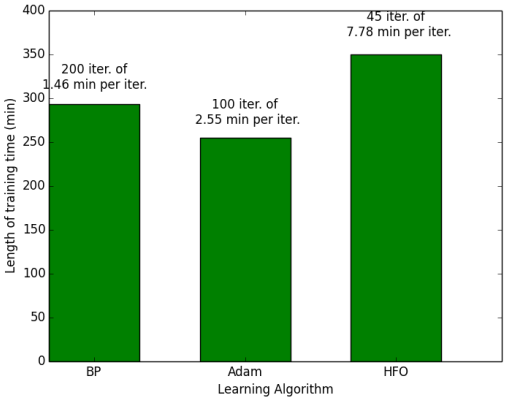

FIGURE 5. Length of training time of the FFNN optimisers: The length of training time in minutes is calculated based on the average 5-fold cross-validation training time of the 5 FFNNs used in each ensemble. The label of each bar corresponds to 'number of iterations / average iteration execution time (min)' (see text for more details).

the HFO learning algorithm, which is a second order learning algorithm, both BP and Adam are first order gradient descent learning algorithms. Although all the training algorithms were applied to the same FFNN architecture, which is described in detail in *Section IV*(*A*), the results are indicative for this problem. Nevertheless, further optimization can be done for the FFNN architectures which are trained with the BP and Adam learning algorithms. The parameters of each one of the training algorithms were tuned through many executions on a single FFNN trained for the PISCES dataset. In the case of BP we have identified as optimal the following set of parameters: learning rate = 0.01 and momentum = 0.0. Similarly, in the case of Adam we have used learning rate = 0.001, beta1 = 0.9, beta2 = 0.999. As it can be seen from *Figure* 4, the results of BP compared to Adam and HFO learning algorithms are very poor. On the other hand, the HFO learning algorithm has achieved 0.8% better Q_3 accuracy and comparable SOV score compared to the Adam learning algorithm. As it can be seen from *Figure* 5, although the HFO algorithm needs only 45 iterations to be trained, each iteration requires an average execution time of 7.78 minutes.

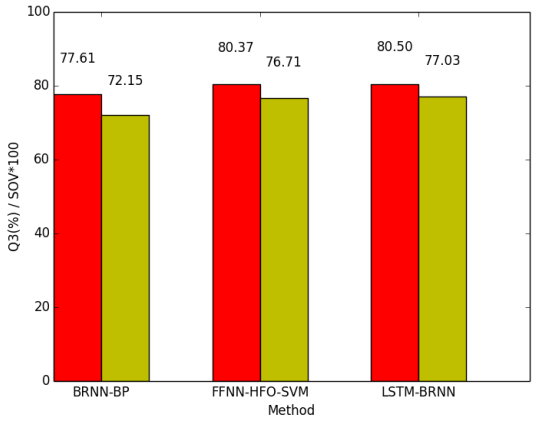

FIGURE 6. **Q3 accuracy and SOV score for ANN models used for the PSSP problem. The red bar corresponds to Q_3 accuracy and the yellow bar to SOV score. The SOV score has been multiplied by 100 for presentation purposes (see text for more details).**

On the other hand, the BP and the Adam algorithms need 200 iterations of 1.46 minutes and 100 iterations of 2.55 minutes each to be trained, respectively. An HFO iteration is executing multiple times the PCG algorithm, a number which cannot be estimated before the completion of an iteration. Therefore, the training time needed for HFO is approximately 350 minutes. The lengh of this execution time is 17% and 27% more than the training time needed for BP and Adam learning algorithms, respectively, despite needing much less iterations. Nevertheless, the entire process of building FFNN classifiers is much faster with the HFO learning algorithm because there are no parameters to be tuned compared to the three parameters of the Adam learning algorithm which need to be optimised by trial and error.

Then, we have chosen to compare our method with an ensemble of 6 BRNNs trained with the BPTT learning algorithm, as one of the most established methods for this problem [18], [23], [29], [60] (see Section II) and also with an LSTM-BRNN, as the method which has been reported as obtaining the highest results for the PSSP problem [32]. The BRNN architecture consisted of 1 hidden layer of 20 hyperbolic tangent neurons for the feed forward subnetwork and 1 layer of 11 hyperbolic tangent neurons for each one of the recurent subnetworks. Furthermore, the BPTT has been tuned to learning rate = 0.1 and momentum = 0.001. The LSTM-BRNN consisted of 25 Bidirectional LSTM neurons and it has been trained with the Adam learning algorithm where the parameters of learning rate = 0.001, beta1 = 0.9, beta2 = 0.999 were used. Both methods methods use the Mean Square Error function. As it can be seen in *Figure* 6, the BRNN trained with BPTT has achieved 77.61% Q_3 accuracy for the PISCES dataset, which is much lower than our results which were approximately 80.4%. Furthermore, although LSTM-BRNNs are reported to capture the long range dependencies of a sequence, they gave comparable overall results to our (more local) method. Definitely, there needs to be more work in order to compare the results of the two methods in such detail (work in progress). Given that our method can capture very well the short range dependencies, the LSTM-BRNNs may face difficulties in short range dependencies which results in the same accuracy to our methodology. Furthermore, our methodology provides an upper limit on what methods may be able to capture when they rely on only local sequence patterns. The biggest advantage of our methodology to these two methods is the simplicity of the models we use. This is very important if we take into account the latest developments in the field which demand very big datasets and network architectures, which consequently increase exponentially the amount of training time. In addition, many of these methods are combined in ensembles, as in [18], where the training amount of time is even more increased. This is extremely important if we take into consideration that an increasing size of ensembles is often used to improve the results of ML methods [61], [62], [64].

V. CONCLUSION

In this work, we present a second order-based methodology for training simple FFNNs for the challenging PSSP problem where both the upstream and downstream information of a sequence is important for a specific time-step. In this methodology, to the best of our knowledge the HFO learning algorithm is applied for the first time to this problem. More specifically, we present the development and implementation of a methodology where ensembles of FFNNs are trained with the HFO learning algorithm for the PSSP problem. In contrast to the conventional GD learning algorithm, the HFO exploits both gradient and curvature information for fast convergence. The results from the ensembles of FFNNs are combined through majority voting ensemble methods which are then fed for filtering purposes to a SVM model for producing the final results.

The efficiency and effectiveness of simple FFNNs trained with the HFO learning algorithm have been tested on the PISCES dataset, achieving approximately 79.6% Q_3 accuracy and approximately 0.72 SOV score which compare well with other state of the art methods. The majority voting ensemble method and SVM filtering techniques have improved even more the single FFNN Q_3 accuracy by approximately 0.8% and the SOV metric by 0.05. The final results of our methodology are approximately 80.4% Q_3 accuracy and approximately 0.77 SOV score. In terms of accuracy, our proposed method outperforms similar methodologies which are trained based on GD or Adam algorithms. Although, the GD and Adam algorithms seem to converge faster than the HFO learning algorithm, the process of building FFNN classifiers is much faster with the HFO because there are no parameters to be tuned. Therefore, our ML method seems to be a particularly good option for complex feature extraction and prediction on sequential data, as it takes advantage of the benefits of these techniques.

At first glance, this method seems to have good chances to outperform in terms of accuracy and convergence time some of the state of the art methods, such as SSpro and

SCRATCH [18], [23], [29], [60] where hundreds of BRNNs are used to achieve an accuracy near to 80%. Hence, to conclude, the predicted sequences from many simulations of our proposed methodology on different PSSP datasets must be carefully extracted and analyzed compared to the results of these state of the art methods. Furthermore, our method has comparable results with the method based on the LSTM-BRNN models which can handle long range dependencies and it has been reported as the method with highest results on this problem. In contrast, our approach provides an upper limit on what methods may be able to capture when they rely on only local sequence patterns. Consequently, this method takes advantage of the strong local dependencies of amino acids. Furthermore, the simplicity of our models is very important if we take into account the latest developments in the field with very big datasets, network architectures and ensembles of networks. Finally, latest developments in the 3D structure prediction of a protein can benefit from our solution on the PSSP problem. More specifically, Alquraishi [80] has presented work, based on deep learning methods, to predict the 3D structure of a protein from its PS. The author has indicated a possible improvement to his methodology if he incorporates PSSP results from other algorithms [80].

The accuracy of 100% will probably never be achieved for the PSSP problem because of the presence of disordered regions, the ambiguities inherent in the definitions of secondary structure, the errors and uncertainties contained in databases and the role of the solvent and other molecules, as well as the inherent protein structural dynamics [29]. Nevertheless, the improvement and systematic combination of sequence profiles, machine learning methods and sequence-based structural similarity methods seem to be the best strategy to improve the results related to the PSSP problem. A contribution on any of these three categories, combined with other data preprocessing and algorithmic methods may play a catalytic role on the general improvement of the PSSP problem results.

As a conclusion to all the results presented in this paper, we can see that the second order HFO learning algorithm can effectively detect and extract features from complex sequential data like the PSSP problem. Furthermore, the combination of multiple ML algorithms seem to be a particularly good option for complex feature extraction and prediction on sequential data, as it takes advantage of the benefits of all techniques. Finally, we demonstrate that a powerful learning algorithm, such as the HFO algorithm, applied on simple ANN models for the PSSP problem can produce comparable results to the most complicated ANN architectures which have been utilized for this problem. Also, this work, gives the initiative to other techniques where powerful learning algorithms can be used for complicated ANN architectures to produce even more accurate results for the PSSP and other related problems. Moreover, these methods may be applied to more problems where the upstream and downstream information is important for a specific time-step of the sequence for more generic conclusions.

APPENDIX A
PROTEINS EXCLUDED FROM PISCES DATASET

TABLE 6. Protein sequences which have been excluded from the PISCES dataset. These protein sequences have been excluded from the datasets because of miscalculations of the PSI-BLAST.

Protein Sequence Name (PISCES)
1BB1A, 1BB1B, 1BB1C, 1C94A, 1DPJB, 1DTDB, 1F8VD, 1GWMA, 1HX6A, 1KD8A, 1KP6A, 1KVEA, 1L2WI, 1M3WA, 1M45B, 1M46B, 1MCTI, 1MQSB, 1MW5A, 1O06A, 1P9IA, 1PJMA, 1PJNA, 1SVFB, 1T01B, 1TQEX, 1TTWB, 1U6HB, 1UVQC, 1ZVZB, 1ZW2B, 2BPA3, 2BPTB, 2C5IP, 2C5KP, 2DS2A, 2ERLA, 2GWWB, 2HZSI, 2MLTA, 2P06A, 2PBDV, 2PLXB, 2QUOA, 2W4YA, 2WBYC, 2WFUA, 2WFVA, 2WWXB, 2X5CA, 2X5RA, 2XF7A, 2XZEQ, 3AJBB, 3C5TB, 3DT5A, 3E8YX, 3FBLA, 3GP2B, 3H0TC, 3HE4B, 3HE5A, 3HE5B, 3L9AX, 3LCNC, 3LJMA, 3M6ZA, 3NK4C, 3OWTC, 3P06A, 3PLVC, 3QKSC, 3R46A, 3R4AA, 3RA3B, 3RF3C, 3RKLA, 3S1BA, 3S6PE, 3SHPA, 3SJHB, 3TQ2A, 3TWEA, 3TZ1B, 3U4VA, 3U4ZA, 3UC7A, 3UKXC, 3UL1A, 3V62C, 3V86A, 3VU5B, 3VU6B, 3VVIA, 3W8VA, 3W92A, 3WKNE, 3WOEB, 3WX4A, 3WY9C, 3ZTAA, 4A94C, 4BFHA, 4BLQA, 4BPLB, 4C1AA, 4CAYC, 4CU4B, 4CXFB, 4DACA, 4EHQG, 4F87A, 4FBWC, 4FTBD, 4FZ0M, 4G1AA, 4GVBB, 4H62V, 4H7RA, 4H8MA, 4H8OA, 4HB1A, 4HBEA, 4HLBA, 4HR1A, 4I7ZE, 4IIKA, 4J4AA, 4JGLA, 4JHKC, 4KVTA, 4KYTB, 4LOOB, 4M1XA, 4M6BC, 4MGPA, 4MI8C, 4N3BB, 4N3CB, 4OGQE, 4OQ9A, 4OYDB, 4OZKA, 4PC0C, 4PN8A, 4PN9A, 4PNAA, 4PNBA, 4PNDA, 4PW1A, 4QMFA, 4R0RA, 4R80A, 4R8TA, 4RIQC, 4TTLA, 4UEBB, 4W6YA, 1M1QA, 1Q2HA, 1Q9UA, 1QV9A, 1SQWA, 1TU9A, 1U9LA, 1UV7A, 1V96A, 1VPRA, 1VR4A, 1WPNA, 1WV3A, 1WWPA, 1YPYA, 1Z4RA, 2BDRA, 2C0NA, 2CVIA, 2D59A, 2D68A, 2D7EA, 2E1FA, 2ERWA, 2FB0A, 2FI1A, 2G7SA, 2HL7A, 2HQLA, 2HX5A, 2IP6A, 2NPTA, 2NS0A, 2O1QA, 2O38A, 2O4AA, 2O99A, 2OL5A, 2OU1A, 2OU5A, 2OX7A, 2P63A, 2P9XA, 2PW9A, 2Q3SA, 2Q3TA, 2R19A, 2R5SA, 2R85A, 2UVPA, 2V73A, 2V9PA, 2VS0A, 2WG8A, 2WVQA, 2Y5PA, 2YF2A, 2ZEXA, 3B0FA, 3BPQA, 3BRVA, 3C0DA, 3C4RA, 3CPTA, 3CRYA, 3CSXA, 3D33A, 3D3OA, 3D55A, 3DFUA, 3DNXA, 3E0RA, 3ESLA, 3ESMA, 3F43A, 3F67A, 3F95A, 3FF5A, 3FH3A, 3G21A, 3GO9A, 3GP6A, 3H0DA, 3H0NA, 3H16A, 3H35A, 3HLSA, 3I00A, 3I4UA, 3I76A, 3IBWA, 3IV4A, 3JQOA, 3K8RA, 3KTOA, 3KUPA, 3KVPA, 3L49A, 3L60A, 3L7HA, 3LQ9A, 3M5QA, 3MD9A, 3N7XA, 3NOQA, 3NR5A, 3NS4A, 3O12A, 3O65A, 3O6QA, 3ONQA, 3OP6A, 3P51A, 3PD7A, 3PL0A, 3Q0HA, 3Q18A, 3Q6CA, 3QH6A, 3RK6A, 3TE8A, 3TS9A, 3TUOA, 3TVQA, 3U5WA, 3U97A, 3UV0A, 3UV1A, 3VS8A, 3W6SA, 3ZC0A, 4AP5A, 4AQOA, 4BOQA, 4BSVA, 4BSXA, 4DHXA, 4E6SA, 4E6WA, 4F27A, 4F2LA, 4F4WA, 4F87A, 4FX7A, 4G97A, 4GOFA, 4GT9A, 4GUCA, 4H41A, 4HHVA, 4I16A, 4I86A, 4IHQA, 4J1VA, 4J5RA, 4J91A, 4JX0A, 4K12A, 4K92A, 4KQIA, 4KTWA, 4L2WA, 4L3UA, 4LKUA, 4LQBA, 4LTBA, 4MO0A, 4MTUA, 4MYVA, 4N74A, 4NUAA, 4P2VA, 4P49A, 4P78A, 4PF3A, 4PSFA, 4Q53A, 4Q7OA, 4QRNA, 4QSGA, 4R7RA, 4U9OA, 4X33A

REFERENCES

[1] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, no. 1, pp. 249–268, 2007.
[2] S. S. Liew, M. Khalil-Hani, and R. Bakhteri, "An optimized second order stochastic learning algorithm for neural network training," *Neurocomputing*, vol. 186, pp. 74–89, Apr. 2016.
[3] T. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition* (Lecture Notes in Computer Science), vol. 2396, T. Caelli, Ed. Springer-Verlag, pp. 15–30, 2002.
[4] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.
[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classication with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, Lake Tahoe, NV, USA, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, pp. 1097–1105, 2012.
[6] K.-I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Netw.*, vol. 6, no. 6, pp. 801–806, 1993.
[7] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5, pp. 602–610, 2005.
[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[9] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks-with an erratum note (technical report gmd-report 148)," German Nat. Res. Center Inf. Technol., Sankt Augustin, Germany, Tech. Rep. 148, 2001.
[10] H. Jaeger, "Discovering multiscale dynamical features with hierarchical echo state networks (technical report, no. 9)," 'School Eng. Sci., Jacobs Univ. Bremen, Bremen, Germany, Tech. Rep. 10, 2007.
[11] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.
[12] H. Jaeger, "Controlling recurrent neural networks by conceptors," 2014, *arXiv:1403.3369*.

[13] J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," 2014, *arXiv:1402.3511*.

[14] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[15] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. S. Kruthiventi, and R. V. Babu, "A taxonomy of deep convolutional neural nets for computer vision," *Frontiers Robot. AI*, vol. 2, p. 36, Jan. 2016.

[16] R. Waseem and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Aug. 2017.

[17] M. Wollmer, F. Eyben, J. Keshet, A. Graves, B. Schuller, and G. Rigoll, "Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Taipei, Taiwan, Apr. 2009, pp. 740–747.

[18] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri, "Exploiting the past and the future in protein secondary structure prediction," *Bioinformatics*, vol. 15, no. 11, pp. 937–946, Nov. 1999.

[19] M. Agathocleous, G. Christodoulou, V. Promponas, C. Christodoulou, V. Vassiliades, and A. Antoniou, "Protein secondary structure prediction with bidirectional recurrent neural nets: Can weight updating for each residue enhance performance?" in *Artificial Intelligence Applications and Innovations* (IFIP Advances in Information and Communication Technology), H. Papadopoulos, A. Andreou, M. Bramer, Ed. Berlin, Germany: Springer-Verlag, vol. 339, pp. 128–137, 2010.

[20] P. Kountouris, M. Agathocleous, V. J. Promponas, G. Christodoulou, S. Hadjicostas, V. Vassiliades, and C. Christodoulou, "A comparative study on filtering protein secondary structure prediction," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 9, no. 3, pp. 731–739, May 2012.

[21] T. Nugent and D. T. Jones, "Transmembrane protein topology prediction using support vector machines," *BMC Bioinf.*, vol. 10, no. 1, p. 159, Dec. 2009.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[23] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi, "Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles," *Proteins, Struct., Function, Genet.*, vol. 47, no. 2, pp. 228–235, May 2002.

[24] M. Agathocleous, C. Christodoulou, V. Promponas, P. Kountouris, and V. Vassiliades, "Training bidirectional recurrent neural network architectures with the scaled conjugate gradient algorithm," *Artificial Neural Networks and Machine Learning* (Lecture Notes in Computer Science), vol. 9886, A. E. P. Villa, P. Masulli A. J. P. Rivero, Springer-Verlag, pp. 123–131, 2016.

[25] Y. Yang, J. Gao, J. Wang, R. Heffernan, J. Hanson, K. Paliwal, and Y. Zhou, "Sixty-five years of the long march in protein secondary structure prediction: The final stretch?" *Briefings Bioinf.*, vol. 19, no. 3, pp. 482–494, 2016.

[26] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers*, vol. 22, no. 12, pp. 2577–2637, Dec. 1983.

[27] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models," *J. Mol. Biol.*, vol. 202, no. 4, pp. 865–884, Aug. 1988.

[28] B. Rost and C. Sander, "Improved prediction of protein secondary structure by use of sequence profiles and neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 90, no. 16, pp. 7558–7562, Aug. 1993.

[29] C. N. Magnan and P. Baldi, "SSpro/ACCpro 5: Almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity," *Bioinformatics*, vol. 30, no. 18, pp. 2592–2597, 2014.

[30] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Zídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[31] C. Fang, "MUFOLD-SS: New deep inception-inside-inception networks for protein secondary structure prediction," *Proteins Struct. Function Bioinf.*, vol. 86, no. 5, pp. 592–598, Sep. 2017.

[32] R. Heffernan, Y. Yang, K. Paliwal, and Y. Zhou, "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*, vol. 33, no. 18, pp. 2842–2849, 2017.

[33] S. Wang, J. Peng, J. Ma, and J. Xu, "Protein secondary structure prediction using deep convolutional neural fields," *Sci. Rep.*, vol. 6, no. 1, p. 18962, May 2016.

[34] J. A. Cuff and G. J. Barton, "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction," *Proteins, Struct., Function, Genet.*, vol. 34, no. 4, pp. 508–519, Mar. 1999.

[35] A. A. Salamov and V. V. Solovyev, "Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments," *J. Mol. Biol.*, vol. 247, no. 1, pp. 11–15, Mar. 1995.

[36] B. Rost and C. Sander, "Combining evolutionary information and neural networks to predict protein secondary structure," *Proteins, Struct., Function, Genet.*, vol. 19, no. 1, pp. 55–72, May 1994.

[37] Z. Guo, J. Hou, and J. Cheng, "DNSS2: Improved *ab initio* protein secondary structure prediction using advanced deep learning architectures," *Proteins*, vol. 89, no. 2, pp. 207–217, 2021.

[38] J. Hou, T. Wu, Z. Guo, F. Quadir, and J. Cheng, "The MULTICOM protein structure prediction server empowered by deep learning and contact distance prediction," *Protein Structure Prediction* (Methods in Molecular Biology), vol. 2165, D. Kihara, Ed. New York, NY, USA: Humana, 2020, pp. 13–26.

[39] K. Kotowski, T. Smolarczyk, I. Roterman-Konieczna, and K. Stapor, "ProteinUnet—An efficient alternative to SPIDER3-single for sequence-based prediction of protein secondary structures," *J. Comput. Chem.*, vol. 42, no. 1, pp. 50–59, Jan. 2021.

[40] M. Shapovalov, R. L. Dunbrack, and S. Vucetic, "Multifaceted analysis of training and testing convolutional neural networks for protein secondary structure prediction," *PLoS ONE*, vol. 15, no. 5, May 2020, Art. no. e0232528.

[41] M. Torrisi and G. Pollastri, "Brewery: Deep learning and deeper profiles for the prediction of 1D protein structure annotations," *Bioinformatics*, vol. 36, no. 12, pp. 3897–3898, Jun. 2020.

[42] G. Xu, Q. Wang, and J. Ma, "OPUS-TASS: A protein backbone torsion angles and secondary structure predictor based on ensemble neural networks," *Bioinformatics*, vol. 36, no. 20, pp. 5021–5026, Dec. 2020.

[43] A. Dionysiou, M. Agathocleous, C. Christodoulou, and V. Promponas, "Convolutional neural networks in combination with support vector machines for complex sequential data classification," *Artificial Neural Networks and Machine Learning*, (Lecture Notes in Computer Science), vol. 11139, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham, Switzerland: Springer-Verlag, 2018, pp. 444–455.

[44] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[45] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, Nov. 1993.

[46] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representation by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, vol. 1. Cambridge, MA, USA: MIT Press, pp. 318–362, 1986.

[47] J. Martens, "Deep learning via hessian-free optimization," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 735–742.

[48] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, L. Getoor and T. Scheffer, Eds., vol. 46. Bellevue, WA, USA: Omnipress, pp. 1033–1040, 2011.

[49] M. F. Møller, "Exact calculation of the product of the hessian matrix of feed-forward network error functions and a vector in $O(N)$ time," Comput. Sci. Dept., Aarhus Univ., Aarhus, Denmark, Tech. Rep. PB-432, 1993.

[50] B. A. Pearlmutter, "Fast exact multiplication by the Hessian," *Neural Comput.*, vol. 6, no. 1, pp. 147–160, 1994.

[51] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *J. Mach. Learn. Res.*, vol. 3, no. 1, pp. 115–143, 2003.

[52] K. Yue and K. Dill, "Constraint-based assembly of tertiary protein structures from secondary structure elements," *Protein Sci.*, vol. 9, no. 10, pp. 1935–1946, 2000.

[53] A. D. Palù, A. Dovier, and F. Fogolari, "Constraint logic programming approach to protein structure prediction," *BMC Bioinf.*, vol. 5, no. 1, p. 186, 2004.

[54] T. Schwede, "Protein modeling: What happened to the 'protein structure gap'?" *Structure*, vol. 21, no. 9, pp. 1531–1540, 2013.
[55] A. Zemla, Č. Venclovas, K. Fidelis, and B. Rost, "A modified definition of Sov, A segment-based measure for protein secondary structure prediction assessment," *Proteins, Struct., Function, Genet.*, vol. 34, no. 2, pp. 220–223, Feb. 1999.
[56] H. Zhang, T. Zhang, K. Chen, K. D. Kedarisetti, M. J. Mizianty, Q. Bao, W. Stach, and L. Kurgan, "Critical assessment of high-throughput standalone methods for secondary structure prediction," *Briefings Bioinf.*, vol. 12, no. 6, pp. 672–688, Nov. 2011.
[57] P. Kountouris and J. D. Hirst, "Prediction of backbone dihedral angles and protein secondary structure using support vector machines," *BMC Bioinf.*, vol. 10, no. 1, p. 437, Dec. 2009.
[58] G. Karypis, "YASSPP: Better kernels and coding schemes lead to improvements in protein secondary structure prediction," *Proteins, Struct., Function, Bioinf.*, vol. 64, no. 3, pp. 575–586, Jun. 2006.
[59] H. Rangwala, K. DeRonne, and G. Karypis, "Protein structure prediction using string kernels," in *Knowledge Discovery in Bioinformatics: Techniques, Methods, and Applications*, X. Hu and Y. Pan, Ed. Hoboken, NJ, USA: Wiley, pp. 145–168, 2007.
[60] J. Cheng, A. Z. Randall, M. J. Sweredoski, and P. Baldi, "SCRATCH: A protein structure and structural feature prediction server," *Nucleic Acids Res.*, vol. 33, pp. W72–W76, Jul. 2005.
[61] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, no. 1, pp. 239–263, 2002.
[62] Z.-H. Zhou, J. Wu, and W. Tang, "Corrigendum to 'ensembling neural networks: Many could be better than all', [Artificial intelligence 137 (1–2) (2002) 239–263]," *Artif. Intell.*, vol. 174, no. 18, p. 1570, Dec. 2010.
[63] H. Li, X. Wang, and S. Ding, "Research and development of neural network ensembles: A survey," *Artif. Intell. Rev.*, vol. 49, no. 4, pp. 455–479, Apr. 2018.
[64] P. M. Granitto, P. F. Verdes, and H. A. Ceccatto, "Neural network ensembles: Evaluation of aggregation algorithms," *Artif. Intell.*, vol. 163, no. 2, pp. 139–162, Apr. 2005.
[65] J. Chen and N. S. Chaudhari, "Cascaded bidirectional recurrent neural networks for protein secondary structure prediction," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 4, no. 4, pp. 572–582, Oct. 2007.
[66] M. J. Wood and J. D. Hirst, "Protein secondary structure prediction with dihedral angles," *Proteins, Struct., Function, Bioinf.*, vol. 59, no. 3, pp. 476–481, Mar. 2005.
[67] G. Wang and R. L. Dunbrack, Jr., "PISCES: A protein sequence culling server," *Bioinformatics*, vol. 19, no. 12, pp. 1589–1591, 2003.
[68] C. A. Kieslich, J. Smadbeck, G. A. Khoury, and C. A. Floudas, "ConSSert: Consensus SVM model for accurate prediction of ordered secondary structure," *J. Chem. Inf. Model.*, vol. 56, no. 3, pp. 455–461, Mar. 2016.
[69] B. Rost, "PHD: Predicting one-dimensional protein structure by profile-based neural networks," *Methods Enzymol.*, vol. 266, pp. 525–539, Jan. 1996.
[70] S. F. Altschul, T. Madden, A. Schäffer, A. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucl. Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997.
[71] I. Wallace, G. Blackshields, and D. Higgins, "Multiple sequence alignment," *Current Opinion Structural Biol.*, vol. 15, no. 3, pp. 261–266, 2005.
[72] C. J. Li and L. Yan, "Mechanical system modelling using recurrent neural networks via quasi-Newton learning methods," *Appl. Math. Model.*, vol. 19, no. 7, pp. 421–428, Jul. 1995.
[73] S. G. Nash, "Newton-type minimization via the Lanczos method," *SIAM J. Numer. Anal.*, vol. 21, no. 4, pp. 770–788, Aug. 1984.
[74] S. G. Nash, "A survey of truncated-Newton methods," *J. Comput. Appl. Math.*, vol. 124, nos. 1–2, pp. 45–59, Dec. 2000.
[75] E. Johansson, E. Dowla, and D. Goodman, "Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method," *Int. J. Neural Syst.*, vol. 2, no. 4, pp. 291–302, 1991.
[76] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *IEE Proc. G Circuits, Devices Syst.*, vol. 139, no. 3, pp. 301–310, Jun. 1992.
[77] N. Schraudolph, "Fast curvature matrix-vector products for second-order gradient descent," *Neural Comput.*, vol. 14, no. 7, pp. 1723–1738, Jul. 2002.
[78] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Jul. 1995.
[79] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, Y. Bengio and Y. LeCun, Eds., 2015, pp. 1–15.
[80] M. AlQuraishi, "End-to-end differentiable learning of protein structure," *Cell Syst.*, vol. 8, no. 4, pp. 292–301, 2019.

KONSTANTINOS CHARALAMPOUS received the B.Sc. degree in computer science from the University of Cyprus, Nicosia, in 2018, and the M.Sc. degree in computer science from the University of Oxford, Oxford, U.K, in 2021. Between his B.Sc. and M.Sc. degrees, he was working as a Data Scientist in a big data and AI company. He is currently working as a Software Engineer in a software company offering data-driven intelligent solutions, Limassol, Cyprus. He is also a part-time Special Scientist at the Department of Computer Science, University of Cyprus.

MICHALIS AGATHOCLEOUS (Member, IEEE) received the B.Sc. degree in computer science from the University of Cyprus, Cyprus, in 2009, the M.Sc. degree in machine learning from the University London College (UCL), U.K., in 2010, and the Ph.D. degree in computer science from the University of Cyprus, in 2019. He is a Senior Research Associate and a Visiting Lecturer at the University of Nicosia and at the University of Cyprus. Since 2010, he has been publishing works related to bioinformatics, recurrent neural networks, deep learning, and general machine learning algorithms. Furthermore, he has applied his machine learning knowledge as a Machine Learning Developer, a Senior Software Engineer, and a Data Scientist in several industrial companies. His main research interests include machine learning, neural networks, deep learning, evolutionary computation, computational intelligence, and bioinformatics.

CHRIS CHRISTODOULOU received the B.Eng. degree in electronic engineering from the Queen Mary and Westfield College, University of London, in 1991, the Ph.D. degree in neural networks/computational neuroscience from the King's College, University of London, in 1997, and the B.A. degree in German from the Birkbeck College, University of London, in 2008. He worked as a Postgraduate Research Assistant (1991–1995) and a Postdoctoral Research Associate (1995–1997) at the Centre for Neural Networks, King's College, University of London. He has taught as a Lecturer (Assistant Professor) at the Birkbeck College, University of London (1997–2005). He has been a Visiting Research Fellow at the King's College (1997–2001). Since 2005, he has been a Visiting Research Fellow at the Birkbeck College. He is currently a Professor at the University of Cyprus after joining in 2005. His current research interests include computational neuroscience, neural networks, and machine learning.

VASILIS PROMPONAS received the B.Sc. degree in physics and the Ph.D. degree in biological sciences from the University of Athens, in 1996 and 2004, respectively. He has received an EMBL-EBI Visitors Program Fellowship, in 1998, and further conducted postdoctoral research at the Department of Cell Biology and Biophysics, University of Athens, from 2004 to 2005. He is currently an Associate Professor and the Head of the Bioinformatics Research Laboratory (BRL), Department of Biological Sciences, University of Cyprus. His research interests include the development of empirical, statistical and machine learning methods/tools and specialized databases for exploiting available types of biological information towards understanding biological systems, ranging from macromolecules and macromolecular complexes, to phenotypes. Key contributions of the BRL are in the fields of sequence analysis and protein structural bioinformatics, computational comparative genomics, and biomedical text mining.