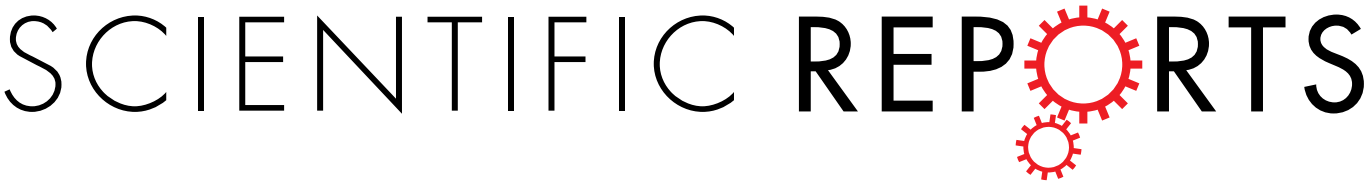

OPEN

Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields

Received: 28 June 2015
Accepted: 26 November 2015
Published: 11 January 2016

Sheng Wang[1,2], Jian Peng[3], Jianzhu Ma[1] & Jinbo Xu[1]

Protein secondary structure (SS) prediction is important for studying protein structure and function. When only the sequence (profile) information is used as input feature, currently the best predictors can obtain ~80% Q3 accuracy, which has not been improved in the past decade. Here we present DeepCNF (Deep Convolutional Neural Fields) for protein SS prediction. DeepCNF is a Deep Learning extension of Conditional Neural Fields (CNF), which is an integration of Conditional Random Fields (CRF) and shallow neural networks. DeepCNF can model not only complex sequence-structure relationship by a deep hierarchical architecture, but also interdependency between adjacent SS labels, so it is much more powerful than CNF. Experimental results show that DeepCNF can obtain ~84% Q3 accuracy, ~85% SOV score, and ~72% Q8 accuracy, respectively, on the CASP and CAMEO test proteins, greatly outperforming currently popular predictors. As a general framework, DeepCNF can be used to predict other protein structure properties such as contact number, disorder regions, and solvent accessibility.

The 3D structure of a protein is determined largely by its amino acid sequence[1]. However it is extremely challenging to predict protein structure from sequence alone[2]. Since protein structure is critical to analysis of its function and many applications like drug and/or enzyme design[3–5], understanding the complex sequence-structure relationship is one of the greatest challenges in computational biology[6–8]. Accurate protein structure and function prediction relies, in part, on the accuracy of secondary structure prediction[9–12].

Protein secondary structure (SS) refers to the local conformation of the polypeptide backbone of proteins. There are two regular SS states: alpha-helix (H) and beta-strand (E), as suggested by Pauling[13] more than 60 years ago, and one irregular SS type: coil region (C). Sander[14] developed a DSSP algorithm to classify SS into 8 fine-grained states. In particular, DSSP assigns 3 types for helix (G for 3_{10} helix, H for alpha-helix, and I for pi-helix), 2 types for strand (E for beta-strand and B for beta-bridge), and 3 types for coil (T for beta-turn, S for high curvature loop, and L for irregular). Overall, protein secondary structure can be regarded as a bridge that links the primary sequence and tertiary structure and thus, is used by many structure and functional analysis tools[15–18].

Protein SS prediction has been extensively studied[10–12,19–35]. Many computational methods have been developed to predict both 3-state SS and a few to predict 8-state SS. Meanwhile, 8-state prediction may provide more detailed local structure information[33,34,36]. Holley & Karplus[19] and Qian & Sejnowski[20] may be the first that used neural networks (NN) to predict SS, which have been followed by a few others[19,21,23,24,37]. The most significant improvement in SS prediction was achieved by Rost & Sander[23] and Zvelebil *et al.*[35] by making use of sequence profile derived from multiple sequence alignment[38–40]. Jones *et al.*[24] developed a 2-stage neural network method PSIPRED, which takes PSI-BLAST sequence profile[41] as input and obtains ~80% accuracy for 3-state SS prediction. Other machine learning methods include bidirectional recurrent neural networks[26,34,37] (which can capture spatial dependency), probabilistic graphical models[25,29,42], support vector machines[27,28,30,43] and hidden Markov models[22,31].

Very recently Baldi *et al.*[34] presented a template-based method for SS prediction, which can yield much better accuracy by making use of solved structures as templates. However, when close templates are not available, Baldi's method performs slightly worse than PSIPRED. Cheng *et al.*[44] proposed a deep learning approach to 3-state SS prediction using a typical deep belief network model, in which each layer is a restricted Boltzmann machine

[1]Toyota Technological Institute at Chicago, Chicago, IL. [2]Department of Human Genetics, University of Chicago, Chicago, IL. [3]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL. Correspondence and requests for materials should be addressed to S.W. (email: wangsheng@uchicago.edu) or J.X. (email: jinboxu@gmail.com)

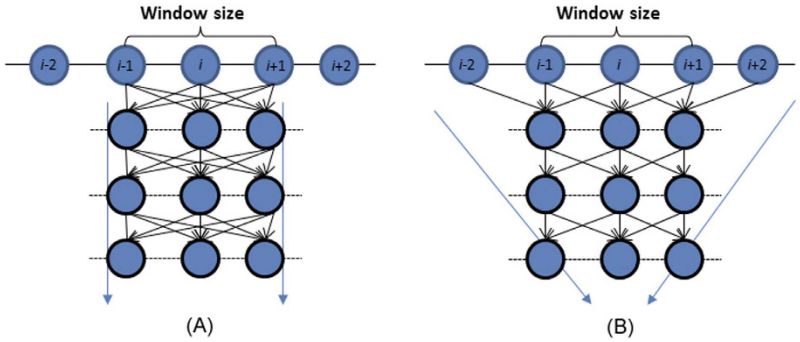

Figure 1. A typical deep neural network (**A**) vs. a convolutional deep neural network (**B**). A convolutional deep neural network can capture longer-range sequence information than a typical deep neural network when both use the same window size.

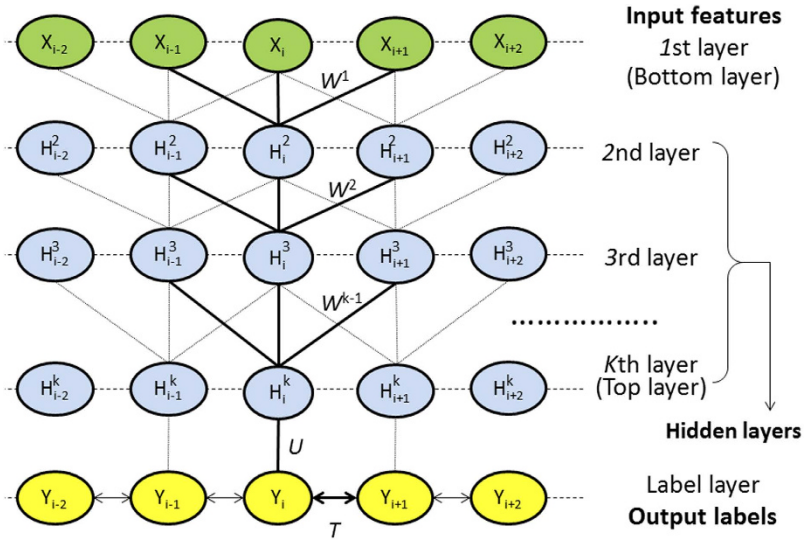

Figure 2. The architecture of DeepCNF, where *i* is the residue index and X_i the associated input features, H^k represents the *k*-th hidden layer, and *Y* is the output label. All the layers from the 1st to the top layer form a deep convolutional neural network (DCNN) with parameter $W^k\{k = 1, 2, \ldots, K\}$. The top layer and the label layer form a conditional random field (CRF) with U and T being the model parameters. U is the parameter used to connect the top layer to the label layer, and T is used to model correlation among adjacent residues.

(RBM)[45] and trained by contrastive divergence[46] in an unsupervised manner. Zhou & Troyanskaya[36] reported another deep learning approach to 8-state SS prediction using a supervised generative stochastic network, which to our best knowledge may be the best 8-state predictor. However, neither Cheng nor Zhou reported a better than 80% accuracy for 3-state prediction.

SS prediction is usually evaluated by Q3 or Q8 accuracy, which measures the percent of residues for which 3-state or 8-state secondary structure is correctly predicted[44]. So far the best Q3 accuracy for ab initio prediction (i.e., templates are not allowed) is ~80% obtained by PSIPRED and a few other state-of-the-art approaches such as JPRED[47,48]. It is very challenging to develop a method that can break this long-lasting record. This may be because the relatively shallow architectures of existing methods cannot model well the complex sequence-structure relationship. Alternatively, 3-state SS prediction could also be measured by segment of overlap (SOV) score, which can be interpreted as SS segment-based accuracy. SOV allows for small wrong predictions at SS segment ends, but penalizes more on wrong predictions in the middle region of a SS segment[49].

In this paper we present a machine learning method DeepCNF (Deep Convolutional Neural Fields) for both 3-state and 8-state SS prediction. DeepCNF combines the advantages of both conditional neural fields (CNF)[50] and deep convolutional neural networks (DCNN)[51], which captures not only complex sequence-structure relationship, but also models SS label correlation among adjacent residues. DeepCNF is similar to conditional random fields (CRF)[52] and CNF[33] in modeling interdependency among adjacent SS labels. However, DeepCNF uses DCNN to replace the shallow neural networks used in CNF so that it can capture very complex relationship between input features and output labels. This DCNN can also include longer-range sequence information (see Figs 1 and 2).

Our DeepCNF method differs from Cheng's method[44] in that the latter uses a typical deep belief network (see Fig. 1A) while we use a deep convolutional network (see Fig. 1B). As such, our method can capture longer-range sequence information than Cheng's method. Our method also differs from Cheng's method in that the latter does not explicitly model SS interdependency among adjacent residues. Our method differs from Zhou's deep learning method (denoted as ICML2014)[36] in the following aspects: (1) our method places only input features in a visible layer and treats the SS labels as hidden states while Zhou's method places both the input features and SS labels in a visible layer; (2) our method explicitly models the SS label interdependency while Zhou's method does not; (3) our method directly calculates the conditional probability of SS labels on input features while Zhou's method uses sampling; (4) our method trains the model parameter simultaneously from the input to output layer while Zhou's method trains the model parameters layer-by-layer; and (5) more importantly, our method demonstrated a significantly improved Q3 accuracy and SOV score while Zhou's method did not.

Our experiments show that our method greatly outperforms the state-of-the-art methods, especially on those structure types which are more challenging to predict, such as high curvature regions (S), beta loop (T), and irregular loop (L).

Results

Dataset. We used five publicly available datasets: (1) CullPDB[53] of 6125 proteins, (2) CB513 of 513 proteins, (3) CASP10[54] and (4) CASP11[55] datasets containing 123 and 105 domain sequences, respectively, and (5) CAMEO (http://www.cameo3d.org/sp/6-months/) test proteins in the past 6 months (from 2014-12-05 to 2015-05-29). Meanwhile, datasets (2–5) are only used for test. The CullPDB dataset was constructed before CASP10 (i.e., May 2012) and any two proteins in this set share less than 25% sequence identity with each other. Following the same procedure in[36], we divided CullPDB into two subsets for training and test, respectively, such that the training proteins share no more than 25% sequence identity with our test sets (2–4). Our training set consists of ~5600 CullPDB proteins and the remaining ~500 PDB proteins are used as the test data. In total there are 403 CAMEO test targets in the past 6 months and 179 proteins are kept for test after removing those sharing more than 25% sequence identity with the training set. The native SS labels of all the training and test proteins are generated by DSSP[14].

An alternative way to select non-redundant proteins for training and test is to pick one representative from each protein superfamily defined by CATH[56] or SCOP[57]. By using test proteins in different superfamilies than the training proteins, we can reduce the bias incurred by the sequence profile similarity between the training and test proteins. To fulfill this, we use the publically available JPRED training and test data[47] (http://www.compbio.dundee.ac.uk/jpred4/about.shtml), which has 1338 training and 149 test proteins, respectively, each of which belongs to a different superfamily.

Programs to compare. We compare our method DeepCNF-SS (abbreviated as DeepCNF) with the following programs: SSpro[34], RaptorX-SS8[33], ICML2014[36] for 8-state SS prediction; and SSpro, RaptorX-SS8, PSIPRED[24], SPINE-X[12], JPRED[47], for 3-state SS prediction. The SSpro package uses two prediction strategies: without template and with template (i.e., using a solved structure in PDB as template). All the other test methods do not make use of template information at all. All the programs are run with their parameters set according to their respective papers. The program derived from the ICML2014 method is not publicly available, so we cannot evaluate its performance on CASP10, CASP11 and CAMEO test sets. We cannot test Cheng's deep learning method either because it is not publicly available. However, only minor improvement in Q3 accuracy over PSIPRED was reported by Cheng's paper[44].

Performance metric. We measure the prediction results in terms of Q3 and Q8 accuracy. The Q3 (Q8) accuracy is defined as the percentage of residues for which the predicted secondary structures are correct[32]. For 3-state SS prediction, we also calculate the SOV (Segment of OVerlap) score[49], which measures how well the observed and the predicted SS segments match. In particular, the SOV measure assigns a lower score to the prediction deviating from observed SS segment length distribution even if it has high Q3 accuracy (i.e., per-residue accuracy)[31]. A wrong prediction in the middle region of a SS segment results in a lower SOV score than a wrong prediction at the terminal regions. A detailed definition of SOV is described in[32], and also in our Supplemental File.

Determining the regularization factor by cross validation. Our DeepCNF has only a hyper-parameter, i.e., the regularization factor, which is used to avoid overfitting. Once it is fixed, we can estimate all the model parameters by solving the optimization problem in Eq. (10). To choose the right regularization factor and examine the stability of our DeepCNF model, we conduct a five-fold cross-validation test. In particular, we randomly divide the training set (containing 5600 CullPDB proteins) into 5 subsets and then use 4 subsets as training and one as validation. Figure 3 shows the Q8 accuracy of our DeepCNF model with respect to the regularization factor. The optimal regularization factor is around 50, which yields 73.2% Q8 accuracy on average. At this point, the Q8 accuracy difference of all the models is less than 1%, consistent with the previous report[33].

Determining the DeepCNF architecture. The architecture of our DeepCNF model is mainly determined by the following 3 factors (see Fig. 2): (i) the number of hidden layers; (ii) the number of different neurons at each layer; and (iii) the window size at each layer. We fix the window size to 11 because the average length of an alpha helix is around eleven residues[58] and that of a beta strand is around six[59]. To show the relationship between the performance and the number of hidden layers, we trained four different DeepCNF models with 1, 3, 5, and 7 layers, respectively. All the models use the same window size (i.e., 11) and the same number (i.e., 100) of different neurons at each layer. In total these four models have ~50 K, ~270 K, ~500 K, and ~700 K parameters, respectively. We trained the models with different regularization factors. As shown in Fig. 4A, when only one hidden layer is used, DeepCNF becomes CNF[50] and its performance is quite similar to RaptorX-SS8 (single model) as shown in

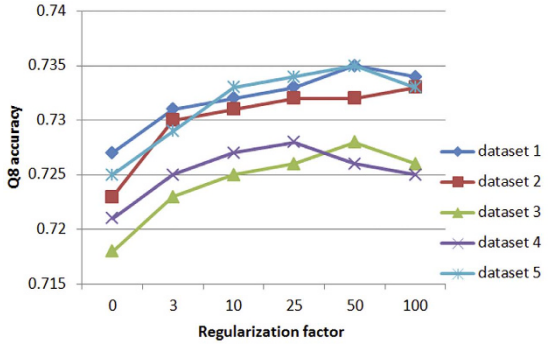

Figure 3. Five-fold cross-validation results of Q8 accuracy on the CullPDB training set with different regularization factors.

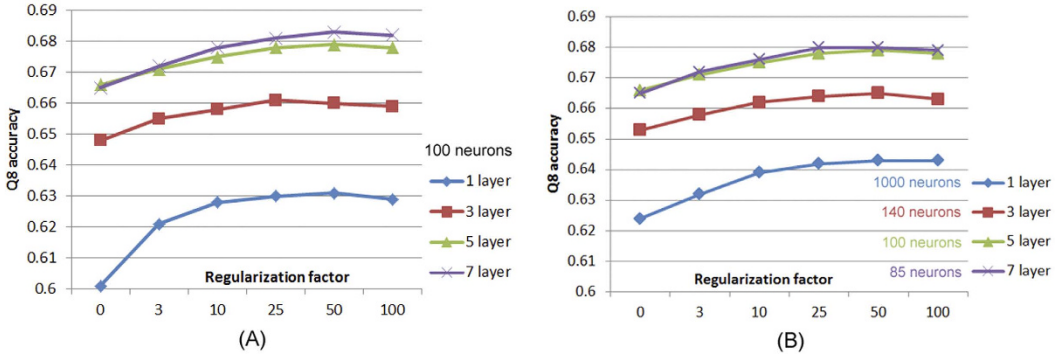

Figure 4. The Q8 accuracy on CB513 by the models of different number of layers of 1, 3, 5, and 7 (the same window size is used). (**A**) Each layer of the 4 models has 100 neurons for a position. The total parameter number of the 4 models is different. (**B**) Each layer of the models has different neurons for a position. The total parameter number of the 4 models is similar.

Methods	Q8 (%)				
	CullPDB	CB513	CASP10	CASP11	CAMEO
SSpro (without template)	66.6	63.5	64.9	65.6	63.5
SSpro (with template)	**85.1**	**89.9**	**75.9**	66.7	65.7
ICML2014	72.1	66.4	—	—	
RaptorX-SS8	69.7	64.9	64.8	65.1	66.2
DeepCNF-SS	75.2	68.3	71.8	**72.3**	**72.1**

Table 1. Q8 accuracy of the tested methods on 5 datasets: CullPDB, CB513, CASP10, CASP11 and CAMEO. The program for ICML2014 is not publicly available. Its result is taken from its paper.

Table 1. When more layers are applied, the Q8 accuracy gradually improves. To balance the model complexity and performance, by default we set window size to 11 and use 5 hidden layers, each with 100 different neurons.

To show that it is the deep convolutional structure but not the number of model parameters that mainly contributes to performance improvement, we trained four models with ~500 K model parameters but 4 different numbers of layers: 1, 3, 5 and 7. We still use the same window size at 11 and for each model all the layers have the same number of neurons. That is, for the 1-, 3-, 5- and 7-layer models, we use 1000, 140, 100, and 85 neurons for each layer, respectively. As shown in Fig. 4B, the models with more layers have better Q8 accuracy, although they have the same number of model parameters. Since the 7-layer model is only slightly better than the 5-layer model, to reduce computational complexity, in the following experimental results we use a DeepCNF model of 5 hidden layers and 100 neurons for each layer. The window size at each layer is set to 11.

Overall performances. Tables 1, 2 and 3 show the Q8, Q3 accuracy, and SOV score of our method DeepCNF and the others on the five datasets. As listed in these tables, when templates are not used, our method significantly outperforms the others, including the popular PSIPRED, our old method RaptorX-SS8[33], SPINE-X and the recent deep learning method ICML2014[36]. SSpro with template obtains very good accuracy on CullPDB,

Methods	Q3 (%)				
	CullPDB	CB513	CASP10	CASP11	CAMEO
SSpro (without template)	79.5	78.5	78.5	77.6	77.5
SSpro (with template)	**88.7**	**90.7**	84.2	78.4	78.9
SPINE-X	81.7	78.9	80.7	79.3	80.0
PSIPRED	82.5	79.2	81.2	80.7	80.1
JPRED	82.9	81.7	81.6	80.4	79.7
RaptorX-SS8	81.2	78.3	78.9	79.1	79.4
DeepCNF-SS	85.4	82.3	**84.4**	**84.7**	**84.5**

Table 2. Q3 accuracy of the tested methods on 5 datasets: CullPDB, CB513, CASP10, CASP11 and CAMEO.

Methods	SOV score (%)				
	CullPDB	CB513	CASP10	CASP11	CAMEO
SSpro (without template)	77.4	77.2	75.9	77.3	75.4
SSpro (with template)	81.3	79.4	80.7	77.4	76.3
SPINE-X	79.1	78.7	78.7	79.3	79.4
PSIPRED	81.8	81.0	80.9	81.4	80.1
JPRED	82.5	83.3	82.4	82.0	80.7
RaptorX-SS8	80.9	79.5	80.2	81.1	78.1
DeepCNF-SS	**86.7**	**84.8**	**85.7**	**86.5**	**85.5**

Table 3. SOV score of the tested methods on 5 datasets: CullPDB, CB513, CASP10, CASP11 and CAMEO.

CB513 and CASP10, but not on CASP11 and CAMEO. This is because SSpro uses a template database built in 2013, covering only the former three sets but not CASP11 or CAMEO. Most CASP11 and CAMEO test proteins share $<25\%$ sequence identity with any template databases created before 2014. In terms of Q3 accuracy on CASP10, DeepCNF obtains 84.4%, even slightly outperforming SSpro with template (84.2%). In terms of both Q3 and Q8 accuracy on CASP11, DeepCNF obtains 83.8% and 71.9%, respectively, significantly outperforming SSpro with template (78.4% and 66.7%, respectively). The same trend is also observed on 179 CAMEO targets, where DeepCNF obtains 84.4% Q3 and 72.1% Q8 accuracy, respectively, much better than SSpro with template (78.9% and 65.7%, respectively).

When only 85 CASP10 and CASP11 hard targets are evaluated, DeepCNF, PSIPRED, SPINE-X, JPRED, RaptorX-SS8, SSpro (with template) and SSpro (without template) have Q3 accuracy 82.2%, 77.9%, 77.1%, 78.5%, 76.2%, 76.9 and 75.4%, respectively, and SOV score 83.0%, 78.1%, 77.2%, 79.8%, 77.9%, 75.3% and 73.6%, respectively. All the methods have lower Q3 accuracy on these hard targets than on the whole datasets since all the test methods use sequence profiles as input features and a hard target usually has sparse sequence profile that carries little evolutionary information. In addition, the hard targets do not have good structure homologs in the training set, so a predictor cannot copy SS labels from the training data as prediction. However, our method even has a slightly larger advantage over the others on the CASP hard targets than on the whole CASP sets. This implies that our method is slightly better than the others in dealing with sparse sequence profiles and learning sequence-structure relationship from the training data. Similar results are observed on the 86 CAMEO hard targets, on which DeepCNF, PSIPRED, SPINE-X, JPRED, RaptorX-SS8, SSpro (with template) and SSpro (without template) have Q3 accuracy 82.1%, 78.0%, 77.6%, 77.7%, 76.8%, 77.0%, and 76.5%, respectively, and SOV score 81.7%, 77.1%, 74.8%, 78.2%, 73.7% , 72.6%, and 72.5%, respectively.

The SOV score tolerates on small wrong predictions at the terminal regions of a segment while penalizes more on the erroneous predictions in the middle region of a segment[49]. As shown in Table 3, in terms of SOV score on all the five datasets, DeepCNF obtains 86.2%, 84.8%, 85.6%, 85.8%, and 84.5%, respectively, significantly outperforming all the other predictors including SSpro with template. These results show that DeepCNF could yield more meaningful SS predictions. This is mainly because our deep convolutional neural networks are better in predicting beta turn (T), high curvature loop (S), and irregular loop (L) states, which appear more often at the boundary of a helix or sheet segment. The other reason is that the conditional random fields in our method models the interdependency among adjacent residues in a SS segment, which helps reduce erroneous predictions in the middle region of a segment.

Recall and precision. Table 4 shows the recall and precision on each of the 8 states obtained by our method DeepCNF and the second best method ICML2014[36] on the CullPDB test set. Both methods fail on state I since it is too rare to even appear in the test set. The result of the ICML 2014 method is taken from its paper. Overall, our method obtains better recall and precision for each state, especially those non-ordinary states such as G, S and T. For the high curvature loop (S), our method has recall and precision 0.323 and 0.543, respectively, while ICML2014 obtains 0.159 and 0.423. For beta turn (T), our method obtains recall and precision 0.594 and 0.613, respectively, while ICML2014 obtains 0.506 and 0.548, respectively. DeepCNF also outperforms ICML2014 for the loop (L) state.

SS8 label	Recall		Precision	
	DeepCNF	ICML2014	DeepCNF	ICML2014
L	**0.707**	0.633	**0.615**	0.541
B	**0.046**	0.001	**0.638**	0.5
E	**0.867**	0.823	**0.814**	0.748
G	**0.302**	0.133	**0.535**	0.496
I	**0.0**	0.0	**0.0**	0.0
H	**0.937**	0.935	**0.878**	0.828
S	**0.323**	0.159	**0.543**	0.423
T	**0.594**	0.506	**0.613**	0.548

Table 4. Recall and precision of DeepCNF and ICML2014 on the CullPDB test set.

SS8 label	Recall		Precision	
	DeepCNF	ICML2014	DeepCNF	ICML2014
L	**0.657**	0.655	**0.571**	0.518
B	**0.026**	0.0	**0.433**	0.0
E	**0.833**	0.797	**0.748**	0.717
G	**0.26**	0.131	**0.49**	0.45
I	**0.0**	0.0	**0.0**	0.0
H	**0.904**	0.9	**0.849**	0.831
S	**0.255**	0.14	**0.487**	0.444
T	**0.528**	0.503	**0.53**	0.496

Table 5. Recall and precision of DeepCNF and ICML2014 on the CB513 dataset.

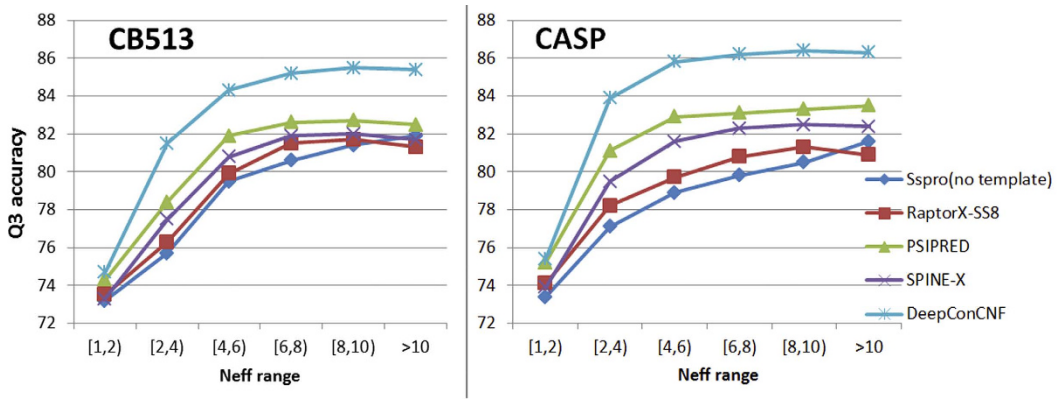

Figure 5. Q3 accuracy on CB513 and two CASP (CASP10-11) test sets with respect to Neff. Each point represents the average Q3 accuracy on those proteins falling into an Neff interval.

This result could be due to the fact that S, T, and L state may be impacted by medium-range information on the protein sequence[36] and our method is better than the others in modeling this kind of information. Our DeepCNF has a similar performance trend on the CB513 test set (see Table 5). DeepCNF outperforms ICML2014, especially on those non-ordinary states, as well as the ordinary beta sheet state. The largest advantage lies in predicting curvature loop (S) and 3_{10} helix (G). We cannot do a detailed comparison between our method and the ICML2014 method on the other test sets since the program derived from the ICML2014 method is not publicly available.

Prediction accuracy with respect to homologous information. We further examine the performance of DeepCNF with respect to the amount of homologous information measured by Neff[33]. The Neff of a protein measures the average number of effective amino acids across all the residues, ranging from 1 to 20 since there are only 20 amino acids in nature. A small Neff indicates that the protein has a sparse sequence profile. By contrast, a large Neff implies that the protein may have a large amount of homologous information. Figure 5 shows the Q3 accuracy of the five tested methods on the CB513 and the two CASP datasets with respect to Neff. When Neff ≤ 2, DeepCNF performs slightly better than the others. However, when Neff > 2, DeepCNF greatly outperforms the others.

Where is the improvement from? We used 25% sequence identity as cutoff to remove redundancy between the training and test sets. Although the training and test proteins may not have similar primary sequences, their sequence profiles may be still similar, so one may wonder if our improvement is due to the sequence profile similarity between the test and training proteins. We conducted one stricter experiment to study this problem. In particular, we retrained our DeepCNF models using the 1338 JPRED training proteins and tested the resultant models on the 149 JPRED test proteins[47]. All the test and training proteins belong to different superfamilies. That is, it is unlikely that one test protein shares similar sequence profile with one training protein. The sequence profiles of these JPRED training and test proteins are generated from an NR database dated in 2012-10-26. We divided the training set into 7 groups according to the JPRED cross-validation sets (available at http://www.compbio.dundee.ac.uk/jpred4/about.shtml) and then trained 7 DeepCNF models separately. Each model is trained by the proteins in 6 groups. Tested on the 149 JPRED test proteins, the resultant 7 DeepCNF models have an average Q3 accuracy of 84.9% (see Supplemental Table 1), far better than what can be obtained by the other methods. For example, on this test set, JPRED, which is one of the best SS predictors, has Q3 accuracy 82.1%.

This experimental result indicates that the improvement mainly comes from the DeepCNF model itself instead of the profile similarity between the test and training proteins. In fact, as shown in previous sections, our method also greatly outperforms the others on the CASP11 and CAMEO hard targets (which do not have similar profiles as our training proteins), which further confirms this conclusion.

Conclusion and Future Work

We have presented a new sequence labeling method, called DeepCNF (Deep Convolutional Neural Fields), for protein secondary structure prediction. This new method can not only model complex sequence-structure relationship by a deep hierarchical architecture, but also exploit interdependency between adjacent SS labels. The overall performance of DeepCNF is significantly better than the state-of-the-art methods, breaking the long-lasting ~80% Q3 accuracy[34]. DeepCNF is even better than the other methods in terms of SOV score. In particular, DeepCNF performs much better on the SS types which are challenging to predict, such as high curvature region (state S), beta loop (state T), and irregular loop (state L). DeepCNF also performs reasonably well on proteins without any good homologs in PDB, better than the other methods. However, DeepCNF has no significant advantage over the others when a protein under prediction has very sparse sequence profile (i.e., Neff ≤ 2). That is, it is still challenging to predict SS structure from primary sequence instead of sequence profile.

In addition to secondary structure prediction, DeepCNF can be directly applied to many sequence labelling problems[50,60–62]. For example, DeepCNF can be used to predict solvent accessibility[34,63,64], contact number[65], structural alphabet[66–69] and order/disorder regions[70–72], which are useful for other purposes such as protein threading, remote homology detection[73–75], and protein model quality assessment[76,77].

Method

DeepCNF model. As shown in Fig. 2, DeepCNF consists of two modules: (a) the Conditional Random Fields (CRF) module consisting of the top layer and the label layer, and (b) the deep convolutional neural network (DCNN) module covering the input to the top layer. When only one hidden layer is used, this DeepCNF becomes Conditional Neural Fields (CNF), a probabilistic graphical model described in[50].

Conditional Random Field (CRF). Given a protein sequence of length L, let $\boldsymbol{Y} = (Y_1, \ldots, Y_L)$ denote its SS where Y_i is the SS type at residue i. Let $\boldsymbol{X} = (X_1, \ldots, X_L)$ denote the input feature where X_i is a column vector representing the input feature for residue i. Using DeepCNF, we calculate the conditional probability of $\boldsymbol{Y}$ on the input $\boldsymbol{X}$ as follows,

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \exp\left(\sum_{i=1}^{L}[\Psi(\boldsymbol{Y}, \boldsymbol{X}, i) + \Phi(\boldsymbol{Y}, \boldsymbol{X}, i)]\right)/Z(\boldsymbol{X}) \quad (1)$$

where $\Psi(\boldsymbol{Y}, \boldsymbol{X}, i)$ is the potential function quantifying correlation among adjacent SS types at around position i, $\Phi(\boldsymbol{Y}, \boldsymbol{X}, i)$ is the potential function modeling relationship between Y_i and input features for position i, and $Z(\mathbf{X})$ is the partition function. Formally, $\Psi()$ and $\Phi()$ are defined as follows,

$$\Psi(\boldsymbol{Y}, \boldsymbol{X}, i) = \sum_{a,b} T_{a,b}\delta(Y_i = a)\delta(Y_{i+1} = b) \quad (2)$$

$$\Phi(\boldsymbol{Y}, \boldsymbol{X}, i) = \sum_a \sum_m U_{a,m} H_m(\boldsymbol{X}, i, W)\delta(Y_i = a) \quad (3)$$

where a and b represent secondary structure states, $\delta()$ is an indicator function, $H_m(\boldsymbol{X}, i, W)$ is a neural network function for the m-th neuron at position i of the top layer, and W, U, and T are the model parameters to be trained. Specifically, W is the parameter for the neural network, U is the parameter connecting the top layer to the label layer, and T is for label correlation. Below see the details of the deep convolutional neural network for $H_m(\boldsymbol{X}, i, W)$.

Deep convolutional neural network (DCNN). Figure 6 shows two adjacent layers. Let M_k be the number of neurons for a single position at the k-th layer. Let $X_i(m)$ be the m-th feature at the input layer for residue i and $H_i^k(m)$ denote the output value of the m-th neuron of position i at layer k. When $k = 1$, $\boldsymbol{H^k}$ is actually the input feature $\boldsymbol{X}$. Otherwise, $\boldsymbol{H^k}$ is a matrix of dimension $L \times M_k$. Let $2N_k + 1$ be the window size at the k-th layer. Mathematically, $H_i^k(m)$ is defined as follows.

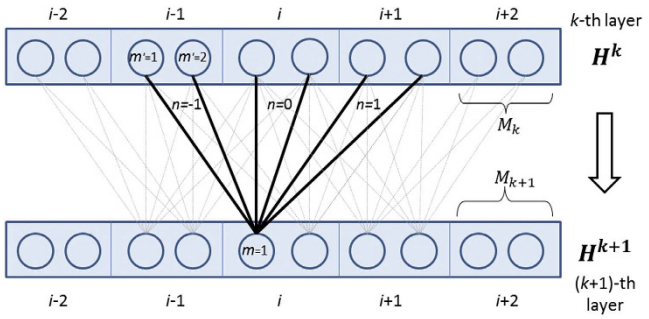

Figure 6. The feed-forward connection between two adjacent layers in the deep convolutional neural network.

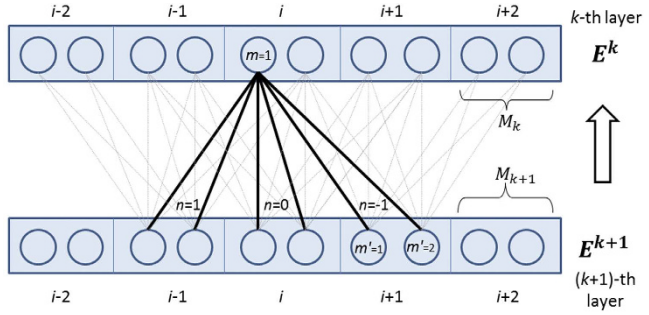

Figure 7. Illustration of calculating the gradient of deep convolutional neural network from layer $k+1$ to layer k.

$$\begin{aligned} H_i^k(m) &= X_i(m), && \text{if } k = 1 \\ H_i^{k+1}(m) &= h\Big(\textstyle\sum_{n=-N_k}^{N_k}\sum_{m'=1}^{M_k}[H_{i+n}^k(m') * W_n^k(m, m')]\Big), && \text{if } k < K \\ H_m(\boldsymbol{X}, i, W) &= H_i^k(m) && \text{if } k = K \end{aligned} \tag{4}$$

Meanwhile, $h()$ is the activation function, either the sigmoid (i.e., $1/(1+\exp(-x))$) or the tanh (i.e., $(1-\exp(-2x))/(1+\exp(-2x))$) function. W_n^k where $(-N_k \le n \le N_k)$ is a 2D weight matrix for the connections between the neurons of position $i+n$ at layer k and the neurons of position i at layer $k+1$. $W_n^k(m, m')$ is shared by all the positions in the same layer, so it is position-independent. Here m' and m index two neurons at the k-th and $(k+1)$-th layers, respectively.

Training method. Similar to CRF[52], we train the model parameters by maximum-likelihood. The log-likelihood is as follows.

$$\log P(\boldsymbol{Y}|\boldsymbol{X}) = \textstyle\sum_{i=1}^{L}[\Psi(\boldsymbol{Y}, \boldsymbol{X}, i) + \Phi(\boldsymbol{Y}, \boldsymbol{X}, i)] - \log Z(\boldsymbol{X}) \tag{5}$$

To train the model parameters, we need to calculate the gradient with respect to each parameter. We calculate the gradient first for CRF and then for DCNN. The gradient of the log-likelihood with respect to the parameters T and U is given by,

$$\nabla_{T_{a,b}} = \Big[\textstyle\sum_{i=1}^{L}\delta(Y_i = a)\delta(Y_{i+1} = b)\Big] - EXP_{P(\tilde{\boldsymbol{Y}}|X,W,U,T)}\Big[\sum_{i=1}^{L}\delta(\tilde{Y}_\iota = a)\delta(\tilde{Y}_{i+1} = b)\Big] \tag{6}$$

$$\nabla_{U_{a,m}} = \Big[\textstyle\sum_{i=1}^{L}\delta(Y_i = a)H_m(\boldsymbol{X}, i, W)\Big] - EXP_{P(\tilde{\boldsymbol{Y}}|\boldsymbol{X},W,U,T)}\Big[\sum_{i=1}^{L}\delta(\tilde{Y}_i = a)H_m(\boldsymbol{X}, i, W)\Big] \tag{7}$$

where EXP is the expectation function and can be calculated efficiently using the forward-backward algorithm[50,52].

As shown in Fig. 7, we can calculate the neuron error values at the k-th layer in a back-propagation mode as follows.

$$\begin{aligned} E_i^k(m) &= g(H_i^k(m)) * \textstyle\sum_a[E_i(a) * U_{a,m}], && \text{if } k = K \\ \text{where } E_i(a) &= [\delta(Y_i = a)] - EXP_{P(\tilde{\boldsymbol{Y}}|\boldsymbol{X},W,U,T)}[\delta(\tilde{Y}_i = a)] \\ E_i^k(m) &= g(H_i^k(m)) * \textstyle\sum_{n=-N_k}^{N_k}\sum_{m'=1}^{M_{k+1}}[E_{i-n}^{k+1}(m') * W_n^k(m', m)], && \text{if } k < K \end{aligned} \tag{8}$$

where $g()$ is the derivative of the activation function; it is $g(x) = (1 - x)x$ and $g(x) = 1 - x^2$ for the sigmoid and tanh function, respectively. $\boldsymbol{E^k}$ is the neuron error value matrix at the k-th layer, with dimension $L \times M_k$. $E_i(a)$ is the error of the log-likelihood function with respect to the label at the i-th position and can be calculated by the forward-backward algorithm. Finally, the gradient of the parameter W at the k-th layer is:

$$\nabla_{W_n^k(m,m')} = \sum_{i=1}^{L} [E_i^{k+1}(m) \ * \ H_{i+n}^k(m')] \tag{9}$$

L2 regularization and L-BFGS. To reduce over-fitting, the log-likelihood objective function is penalized with a L_2-norm of the model parameters. Thus, our final objective function is as follows.

$$\max_\theta \ \log P_\theta(\boldsymbol{Y}|\boldsymbol{X}) \ - \ \lambda\|\theta\|^2 \tag{10}$$

where θ is the set of model parameters and λ is the regularization factor used to avoid overfitting. Although DeepCNF has a large number of model parameters, by setting the regularization factor large enough, we can make the L_2-norm of the model parameters small and thus, restrict the search space of the model parameter and avoid overfitting. However, a very large regularization factor (e.g., infinity) may restrict the model parameter into too small a search space and the resultant model may not learn enough from the training data (i.e., under-fitting). We will determine the regularization factor by cross-validation.

Since the log-likelihood function is not convex, usually we can only solve the objective function to a local instead of global optimum. Although a typical way to train a deep network is to do it layer-by-layer, in our implementation we train all the model parameters simultaneously. We use the L-BFGS[78] to search for the optimal model parameters, which has also been successfully used to train CRF and CNF[50,52]. In addition to learning the model parameters simultaneously, we can also train them layer-by-layer in a supervised mode. Starting from the first layer (i.e., input feature), we train the model parameter $W1$ by removing the third to the K-th layers but keeping the label layer. After $W1$ is trained, we generate the neuron output values for the second layer and use them as input to train the model parameter $W2$ by removing the fourth to the K-th layers but keeping the label layer. We repeat this procedure until all the parameters are trained, and finally we fine-tune these parameters by simultaneous training.

Input Features. We used the input features described in[36]. In particular, for each protein sequence, we ran PSI-BLAST[41] with E-value threshold 0.001 and 3 iterations to search UniRef90[79] to generate the position specific scoring matrix (PSSM). We then transform PSSM by the sigmoid function $1/(1 + \exp(-x))$ where x is a PSSM entry. We also use a binary vector of 21 elements to indicate the amino acid type at position i. We use 21 since there might be some unknown amino acids in a protein sequence. In total there are 42 input features for each residue, 21 from PSSM and the other 21 from the primary sequence. Note that besides using PSSM generated by 3 iterations, we could also add the PSSM generated by 5 iterations into the input features.

Availability. DeepCNF is available at http://raptorx.uchicago.edu/download/.

References

1. Baker, D. & Sali, A. Protein structure prediction and structural genomics. *Science* **294,** 93–96 (2001).
2. Dill, K. A. & MacCallum, J. L. The protein-folding problem, 50 years on. *Science* **338,** 1042–1046 (2012).
3. Petsko, G. A. & Ringe, D. *Protein structure and function*. (New Science Press, 2004).
4. Whittle, P. J. & Blundell, T. L. Protein structure-based drug design. *Annu. Rev. Biophys. Biomol. Struct.* **23,** 349–375 (1994).
5. Schaffhausen, J. Advances in structure-based drug design. *Trends Pharmacol. Sci.* **33,** 223 (2012).
6. Whisstock, J. C. & Lesk, A. M. Prediction of protein function from protein sequence and structure. *Q. Rev. Biophys.* **36,** 307–340 (2003).
7. Lee, D., Redfern, O. & Orengo, C. Predicting protein function from sequence and structure. *Nat. Rev. Mol. Cell Biol.* **8,** 995–1005 (2007).
8. Radivojac, P. *et al.* A large-scale evaluation of computational protein function prediction. *Nat. Methods* **10,** 221–227 (2013).
9. Rost, B., Yachdav, G. & Liu, J. The predictprotein server. *Nucleic Acids Res.* **32,** W321–W326 (2004).
10. Lin, K., Simossis, V. A., Taylor, W. R. & Heringa, J. A simple and fast secondary structure prediction method using hidden neural networks. *Bioinformatics* **21,** 152–159 (2005).
11. Yoo, P. D., Zhou, B. B. & Zomaya, A. Y. Machine learning techniques for protein secondary structure prediction: an overview and evaluation. *Current Bioinformatics* **3,** 74–86 (2008).
12. Faraggi, E., Zhang, T., Yang, Y., Kurgan, L. & Zhou, Y. SPINE X: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *J. Comp. Chem.* **33,** 259–267 (2012).
13. Pauling, L., Corey, R. B. & 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).
14. Kabsch, W. & Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen - bonded and geometrical features. *Biopolymers* **22,** 2577–2637 (1983).
15. Myers, J. K. & Oas, T. G. Preorganized secondary structure as an important determinant of fast protein folding. *Nat. Struct. Mol. Biol.* **8,** 552–558 (2001).
16. Källberg, M. *et al.* Template-based protein structure modeling using the RaptorX web server. *Nature protocols* **7,** 1511–1522 (2012).
17. Zhang, Y. I-TASSER server for protein 3D structure prediction. *BMC Bioinformatics* **9,** 40 (2008).
18. Simons, K. T., Kooperberg, C., Huang, E. & Baker, D. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *J. Mol. Biol.* **268,** 209–225 (1997).
19. Holley, L. H. & Karplus, M. Protein secondary structure prediction with a neural network. *Proc. Natl. Acad. Sci. USA* **86,** 152–156 (1989).
20. Qian, N. & Sejnowski, T. J. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.* **202,** 865–884 (1988).

21. Kneller, D., Cohen, F. & Langridge, R. Improvements in protein secondary structure prediction by an enhanced neural network. *J. Mol. Biol.* **214,** 171–182 (1990).
22. Asai, K., Hayamizu, S. & Handa, K. I. Prediction of protein secondary structure by the hidden Markov model. *Comput. Appl. Biosci.* **9,** 141–146 (1993).
23. Rost, B. & Sander, C. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.* **232,** 584–599 (1993).
24. Jones, D. T. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* **292,** 195–202 (1999).
25. Schmidler, S. C., Liu, J. S. & Brutlag, D. L. Bayesian segmentation of protein secondary structure. *J. Comput. Biol.* **7,** 233–248 (2000).
26. Pollastri, G., Przybylski, D., Rost, B. & Baldi, P. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins: Struct. Funct. Bioinform.* **47,** 228–235 (2002).
27. Kim, H. & Park, H. Protein secondary structure prediction based on an improved support vector machines approach. *Protein Eng.* **16,** 553–560 (2003).
28. Ward, J. J., McGuffin, L. J., Buxton, B. F. & Jones, D. T. Secondary structure prediction with support vector machines. *Bioinformatics* **19,** 1650–1655 (2003).
29. Chu, W., Ghahramani, Z. & Wild, D. L. A graphical model for protein secondary structure prediction. *Proceedings of the Twenty-first International Conference (ICML 2004)*, Banff, Alberta, Canada, July 4-8, 2004. ACM International Conference Proceeding Series **69**, *ACM* 2004, 21 (2004).
30. Guo, J., Chen, H., Sun, Z. & Lin, Y. A novel method for protein secondary structure prediction using dual - layer SVM and profiles. *Proteins: Struct. Funct. Bioinform.* **54,** 738–743 (2004).
31. Aydin, Z., Altunbasak, Y. & Borodovsky, M. Protein secondary structure prediction for a single-sequence using hidden semi-Markov models. *BMC Bioinformatics* **7,** 178 (2006).
32. Im, I. G. *Predicting Protein Secondary Structure Using Markov Chain Monte-Carlo Simulation*. (ProQuest, 2008).
33. Wang, Z., Zhao, F., Peng, J. & Xu, J. Protein 8 - class secondary structure prediction using conditional neural fields. *Proteomics* **11,** 3786–3792 (2011).
34. Magnan, C. N. & Baldi, P. SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics* **30,** 2592–2597 (2014).
35. Zvelebil, M. J., Barton, G. J., Taylor, W. R. & Sternberg, M. J. Prediction of protein secondary structure and active sites using the alignment of homologous sequences. *J. Mol. Biol.* **195,** 957–961 (1987).
36. Zhou, J. & Troyanskaya, O. Deep Supervised and Convolutional Generative Stochastic Network for Protein Secondary Structure Prediction. *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. JMLR Proceedings 32*, 745-753 (2014).
37. Baldi, P., Brunak, S., Frasconi, P., Soda, G. & Pollastri, G. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics* **15,** 937–946 (1999).
38. Higgins, D. G. & Sharp, P. M. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene* **73,** 237–244 (1988).
39. Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32,** 1792–1797 (2004).
40. Thompson, J. D., Higgins, D. G. & Gibson, T. J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22,** 4673–4680 (1994).
41. Altschul, S. F. *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25,** 3389–3402 (1997).
42. Maaten, L., Welling, M. & Saul, L. K. Hidden-unit conditional random fields. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. JMLR Proceedings* **15**, 479-488 (2011).
43. Hua, S. & Sun, Z. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *J. Mol. Biol.* **308,** 397–407 (2001).
44. Spencer, M., Eickholt, J. & Cheng, J. A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **12,** 103–112 (2015).
45. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313,** 504–507 (2006).
46. Hinton, G. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **14,** 1771–1800 (2002).
47. Drozdetskiy, A., Cole, C., Procter, J. & Barton, G. J. JPred4: a protein secondary structure prediction server. *Nucleic Acids Res.*, gkv332 (2015).
48. Cuff, J. A., Clamp, M. E., Siddiqui, A. S., Finlay, M. & Barton, G. J. JPred: a consensus secondary structure prediction server. *Bioinformatics* **14,** 892–893 (1998).
49. Zemla, A., Venclovas, Č., Fidelis, K. & Rost, B. A modified definition of Sov, a segment - based measure for protein secondary structure prediction assessment. *Proteins: Struct. Funct. Bioinform.* **34,** 220–223 (1999).
50. Peng, J., Bo, L. & Xu, J. Conditional neural fields. *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009*. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada. Curran Associates, Inc. 2009, 1419-1427 (2009).
51. Lee, H., Grosse, R., Ranganath, R. & Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009. ACM International Conference Proceeding Series 382, ACM 2009*, 609-616 (2009).
52. Lafferty, J., McCallum, A. & Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001. Morgan Kaufmann 2001* (2001).
53. Wang, G. & Dunbrack, R. L. PISCES: a protein sequence culling server. *Bioinformatics* **19,** 1589–1591 (2003).
54. Kryshtafovych, A. *et al.* Assessment of the assessment: evaluation of the model quality estimates in CASP10. *Proteins: Struct. Funct. Bioinform.* **82,** 112–126 (2014).
55. Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T. & Tramontano, A. Critical assessment of methods of protein structure prediction (CASP)—round x. *Proteins: Struct. Funct. Bioinform.* **82,** 1–6 (2014).
56. Sillitoe, I. *et al.* CATH: comprehensive structural and functional annotations for genome sequences. *Nucleic Acids Res.* **43,** D376–D381 (2015).
57. Andreeva, A., Howorth, D., Chothia, C., Kulesha, E. & Murzin, A. G. SCOP2 prototype: a new approach to protein structure mining. *Nucleic Acids Res.* **42,** D310–D314 (2014).
58. Andersen, C. A., Bohr, H. & Brunak, S. Protein secondary structure: category assignment and predictability. *FEBS Lett.* **507,** 6–10 (2001).
59. Penel, S., Morrison, R. G., Dobson, P. D., Mortishire - Smith, R. J. & Doig, A. J. Length preferences and periodicity in β - strands. Antiparallel edge β - sheets are more likely to finish in non - hydrogen bonded rings. *Protein Eng.* **16,** 957–961 (2003).
60. Sha, F. & Pereira, F. Shallow parsing with conditional random fields. HLT-NAACL 2003, *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, *May* 27 - June 1, *Edmonton, Canada*, 134-141 (2003).

61. Wang, S. B., Quattoni, A., Morency, L., Demirdjian, D. & Darrell, T. Hidden conditional random fields for gesture recognition. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA. IEEE Computer Society 2006* **2**, 1521-1527 (2006).
62. Taskar, B., Guestrin, C. & Koller, D. Max-margin Markov networks. *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]. MIT Press 2004* **16**, 25 (2004).
63. Joo, K., Lee, S. J. & Lee, J. Sann: Solvent accessibility prediction of proteins by nearest neighbor method. *Proteins: Struct. Funct. Bioinform*. **80,** 1791–1797 (2012).
64. Faraggi, E., Xue, B. & Zhou, Y. Improving the prediction accuracy of residue solvent accessibility and real - value backbone torsion angles of proteins by guided - learning through a two - layer neural network. *Proteins: Struct. Funct. Bioinform*. **74,** 847–856 (2009).
65. Kinjo, A. R., Horimoto, K. & Nishikawa, K. Predicting absolute contact numbers of native protein structure from amino acid sequence. *Proteins: Struct. Funct. Bioinform*. **58,** 158–165 (2005).
66. Wang, S. & Zheng, W.-M. CLePAPS: fast pair alignment of protein structures based on conformational letters. *J. Bioinf. Comput. Biol*. **6,** 347–366 (2008).
67. Wang, S. & Zheng, W.-M. Fast multiple alignment of protein structures using conformational letter blocks. *Open Bioinformatics Journal* **3,** 69–83 (2009).
68. Wang, S., Ma, J., Peng, J. & Xu, J. Protein structure alignment beyond spatial proximity. *Scientific reports* **3** (2013).
69. Zheng, W.-M. The use of a conformational alphabet for fast alignment of protein structures. *Bioinformatics Research and Applications*, 331–342 (2008).
70. Cheng, J., Sweredoski, M. J. & Baldi, P. Accurate prediction of protein disordered regions by mining protein structure data. *Data Min. Knowl. Disc*. **11,** 213–222 (2005).
71. Ward, J. J., McGuffin, L. J., Bryson, K., Buxton, B. F. & Jones, D. T. The DISOPRED server for the prediction of protein disorder. *Bioinformatics* **20,** 2138–2139 (2004).
72. Dosztányi, Z., Csizmok, V., Tompa, P. & Simon, I. IUPred: web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content. *Bioinformatics* **21,** 3433–3434 (2005).
73. Ma, J., Peng, J., Wang, S. & Xu, J. A conditional neural fields model for protein threading. *Bioinformatics* **28,** i59–i66 (2012).
74. Ma, J., Wang, S., Zhao, F. & Xu, J. Protein threading using context-specific alignment potential. *Bioinformatics* **29,** i257–i265 (2013).
75. Ma, J., Wang, S., Wang, Z. & Xu, J. MRFalign: protein homology detection through alignment of Markov random fields. *PLoS Comp. Biol*. **10,** e1003500 (2014).
76. Benkert, P., Künzli, M. & Schwede, T. QMEAN server for protein model quality estimation. *Nucleic Acids Res.*, gkp322 (2009).
77. Zhao, F. & Xu, J. A position-specific distance-dependent statistical potential for protein structure and functional study. *Structure* **20,** 1118–1126 (2012).
78. Liu, D. C. & Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical programming* **45,** 503–528 (1989).
79. Consortium, U. The universal protein resource (UniProt). *Nucleic Acids Res.* **36,** D190–D195 (2008).

Acknowledgements

National Institutes of Health R01GM0897532 to J.X. and National Science Foundation DBI-0960390 to J.X. The authors are also grateful to the computing power provided by the UChicago Beagle and RCC allocations.

Author Contributions

S.W. conceived, designed and implemented the algorithm. J.P. and J.M. helped design the algorithm. S.W. and J.X. designed the experiments, analyzed the data and wrote the manuscript. All authors read and approved the final manuscript.

Additional Information

Supplementary information accompanies this paper at http://www.nature.com/srep

Competing financial interests: The authors declare no competing financial interests.

How to cite this article: Wang, S. *et al.* Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Sci. Rep.* **6**, 18962; doi: 10.1038/srep18962 (2016).