Appendix

S1. Three protein sequence labeling problems

We employ three impo==rtant protein sequence labeling problems to test our DeepCNF models trained by three different methods: solvent accessibility (ACC) prediction, disorder (DISO) prediction, and 8-state protein secondary structure (SS8) prediction. A protein sequence consists of a collection of sequentially-linked residues. We want to predict a label for each residue from the sequence information. Below we briefly introduce each problem, especially how to calculate the true label.

ACC. We used DSSP [11] to calculate the absolute accessible surface area for each residue in a protein and then normalize it by the maximum solvent accessibility to obtain the relative solvent accessibility (RSA) [4]. Solvent accessibility of one residue is classified into 3 labels: buried (B) for RSA from 0 to 10), intermediate (I) for RSA from 10 to 40 and exposed (E) for RSA from 40 to 100. The ratio of these three labels is around 1:1:1.

DISO. Following the definition in [15], we label a residue as disordered (label 1) if it is in a segment of more than three residues missing atomic coordinates in the X-ray structure. Otherwise it is labeled as ordered (label 0). The distribution of these two labels (ordered vs. disordered) is 94:6.

SS8. The 8-state protein secondary structure is calculated by DSSP [11]. In particular, DSSP assigns 3 types for helix (G for 310 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). The distribution of these 8 labels (H,E,L,T,S,G,B,I) is 35:22:19:11:8:4:1:1.

Existing work. Quite a few methods have been developed to predict ACC, DISO, and SS8 [13,10,20]. Many of them used networks (NN) [17] or support vector machines (SVM) [9]. Recently, [6] applied a deep belief network (DBN) [8] to DISO prediction, and [21] reported a supervised generative stochastic network (GSN) [2] for SS8 prediction. Besides maximum-AUC training, our work differs from them as follows.

Our method differs from Chengs work on DISO prediction: (a) we use DCNN while Cheng uses DBN. DCNN is better than DBN in capturing a longer-range of sequential information; and (b) our method considers the correlation of the ordered/disordered states of sequentially-adjacent residues while Chengs method does not.

Our method differs from Zhous work on SS8 prediction: (a) our method places only input features at a visible layer and treats the SS labels as hidden states while Zhous method places both the input features and SS labels in a visible layer; (b) our method explicitly models the SS label interdependency while Zhous method does not; (c) our method directly calculates the conditional probability of SS labels on input features while Zhous method uses sampling; and (d) our method trains the model parameter simultaneously from end to end while Zhous method trains the model parameters layer-by-layer.

Input features. Given a protein sequence, we use the same feature set for the prediction of ACC, DISO, and SS8. There are two types of features: residue-related feature and evolution-related feature.

Residue-related features. (a) amino acid identity represented as a binary vector of 20 elements; (b) amino acid physic-chemical properties (7 values from Table 1 in [14]); propensity of being at endpoints of a secondary structure segment (11 values from Table 1 in [5]; (d)correlated contact potential (40 values from Table 3 in [19] and (e) AAindex (5 values from Table 2 in [1]). These features may allow for a richer representation of amino acids [12].

Evolution-related features. We use PSSM (position specific scoring matrix) generated by PSI-BLAST [3] to encode the evolutionary information of the sequence under prediction. We also use the HHM profile generated by HHpred [18], which is complementary to PSSM to some degree.

S2. More details about the DeepCNF model

As shown in Fig.1 in the main text, DeepCNF has three architecture hyper-parameters: (a) the number of neurons at each layer; (b) the window size at each layer; and (c) the number of hidden layers. We train the model parameters (i.e., U, T, W) simultaneously. We first calculate the gradient for parameter U, T and then for parameter W. Below we explain how to calculate the DeepCNF in a feed-forward way and the gradient by back-propagation.

S2.1 Feed-forward function of DCNN (deep convolutional neural network)

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

$$\begin{aligned} H_i^k(h) =& X_i(h) && \text{if } k = 1 \\ H_i^{k+1}(h) =& \pi\Big(\sum_{n=-N_k}^{N_k} \sum_{h'=1}^{M_k} (H_{i+n}^k(h') * W_n^k(h, h')) \Big) && \text{if } k < K \\ A_h(X, i, W) =& H_i^k(h) && \text{if } k = K. \end{aligned}$$

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

S2.2 Approximated AUC

We have introduced the detailed derivation of AUC in related work. We therefore carefully computed the gradient of the approximate AUC with respect

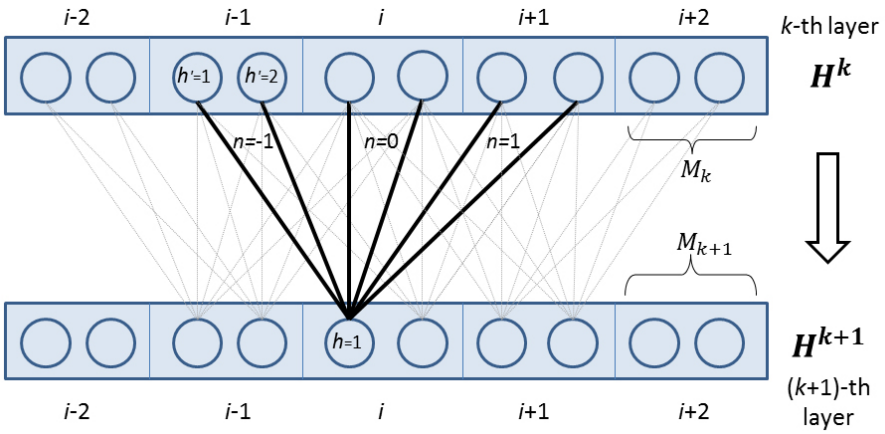

Fig. 1. The feed-forward connection between two adjacent layers of DCNN.

to the parameter θ, which is as follows:

$$\frac{\partial AUC^{WMW}(P_\theta,\tau)}{\partial\theta}=\frac{1}{n_0n_1}\sum_{\mu=0}^{d}\sum_{l=0}^{\mu}\mathcal{Y}_{\mu l}\Big(\frac{\partial s(P_\theta^l,D^\tau)}{\partial\theta}v(P_\theta^{\mu-l},D^{!\tau})==+s(P_\theta^l,D^\tau)\frac{\partial v(P_\theta^{\mu-l},D^{!\tau})}{\partial\theta}\Big).$$

Note that the calculation of $\frac{\partial s(P_\theta^l,D^\tau)}{\partial\theta}$ and $\frac{v(P_\theta^{\mu-l},D^{!\tau})}{\partial\theta}$ is similar, so we only explain one of them, and suppose there is only one training sequence with length L. In particular,

$$\frac{\partial s(P_\theta^l,D^\tau)}{\partial\theta}=\sum_{i\in[L]}\frac{\partial(\delta_i^\tau P_\theta(y_i^\tau|X))^l}{\partial\theta}.$$

Let $Q_i(P_\theta)=(\delta_i^\tau P_\theta(y_i^\tau|X))^l$, then

$$\frac{\partial s(P_\theta^l,D^\tau)}{\partial\theta}=\sum_{i\in[L]}Q_i'\frac{\partial P_\theta(y_i^\tau|X)}{\partial\theta},\tag{1}$$

where Q_i' is the gradient of Q_i with respect to the marginal probability P_θ.

Since

$$P_\theta(y_i^\tau|X)=\frac{1}{Z(X)}\sum_{y_{1:L}}\Big(\delta(y_i=\tau)\exp(F_{1:L}(y,X,\theta))\Big),$$

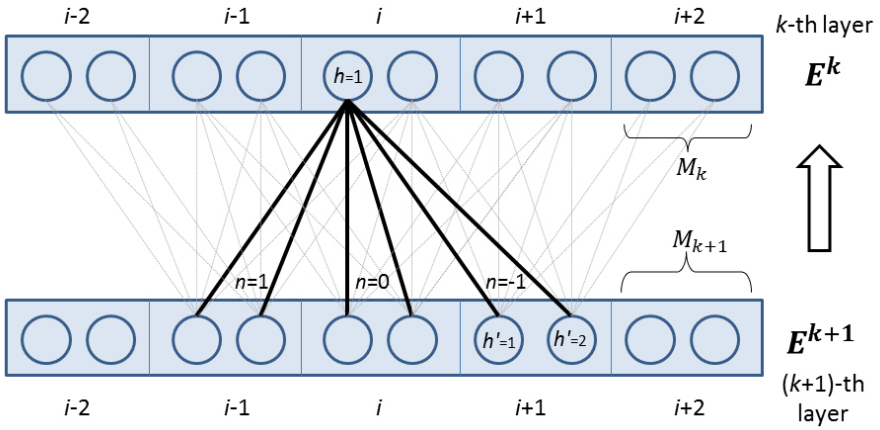

Fig. 2. Illustration of how to calculate the gradient of DCNN from layer $k+1$ to layer k.

applying the quotient rule we can compute the gradient of equation (1) as follows

$$\frac{\partial s(P_\theta^l, D^\tau)}{\partial \theta} = \sum_{i\in[L]} \left(\frac{1}{Z(x)} Q'_i \sum_{y_{1:L}} \left(\delta(y_i = \tau) \frac{\partial F_{1:L}(y, X, \theta)}{\partial \theta} \exp(F_{1:L}(y, X, \theta)) \right) \right) \cdot \frac{-1}{Z(X)} \frac{\partial Z(x)}{\partial \theta} \sum_{i\in[L]} \left(Q'_i P_\theta(y_i^\tau | X) \right). \tag{2}$$

The second term in equation (2) could be calculated efficiently using forward-backward algorithm. For parameter T at position i, the gradient could be calculated as follow:

$$-\mathcal{C} \sum_{u'} \sum_{u} \frac{\alpha(u', i-1)\beta(u, i)}{Z(X)} \exp(f_\theta(u', u, X, i)) \frac{\partial f_\theta(u', u, X, i)}{\partial \theta}.$$

For parameter U at position i, the gradient could be calculated as follows:

$$-\mathcal{C} \sum_{u} \frac{\alpha(u, i)\beta(u, i)}{Z(X)} \frac{\partial g_\theta(u, X, i)}{\partial \theta},$$

where $u = \sum_{i\in[L]} \left(Q'_i P_\theta(y_i^\tau | X) \right)$ denotes one label and

$$\mathcal{C} = \sum_{i\in[L]} \left(Q'_i P_\theta(y_i^\tau | X) \right).$$

The forward function $\alpha(u,i)$ and backward function $\beta(u,i)$ are defined as

$$\alpha(u,i) = \sum_{y_{1:i}} \delta(y_i = u) \exp(F_{1:i}(y, X, \theta))$$
$$\beta(u,i) = \sum_{y_{i:L}} \delta(y_i = u) \exp(F_{i+1:L}(y, X, \theta)).$$

They can be calculated by dynamic programming as follows,

$$\alpha(u,i) = \sum_{u'} \alpha(u', i-1) \exp(f_\theta(u', u, X, i))$$
$$\beta(u,i) = \sum_{u'} \beta(u', i+1) \exp(f_\theta(u, u', X, i+1)).$$

The gradient of the inner summation part of the first term in equation (2) with respect to parameter T at position i could be calculated as follows:

$$\sum_u \sum_{u'} \phi(u', u, i) \exp(f_\theta(u', u, X, i)) \frac{\partial f_\theta(u', u, X, i)}{\partial \theta},$$

where

$$\phi(u', u, i) = Q'_i \delta(y_i = \tau) \frac{\alpha(u', i-1)\beta(u,i)}{Z(X)} + \frac{\alpha^\tau(u', i-1)\beta(u,i)}{Z(X)} + \frac{\alpha(u', i-1)\beta^\tau(u,i)}{Z(X)}.$$

Similarly, the inner summation part of the first term in equation (2) with respect to parameter U at position i could be calculated as

$$\sum_u \Phi(u,i) \frac{\partial g_\theta(u, X, i)}{\partial \theta},$$

where $\Phi(u,i) = \frac{\alpha^\tau(u,i)\beta(u,i)}{Z(X)} + \frac{\alpha(u,i)\beta^\tau(u,i)}{Z(X)}$. Here we define,

$$\alpha^\tau(u,i) = \sum_{t=1}^{i} \sum_{y_{1:i}} \delta(y_t = \tau \wedge y_i = u) Q'_t \exp(F_{1:i}(y, X, \theta))$$
$$\beta^\tau(u,i) = \sum_{t=i+1}^{L} \sum_{y_{i:L}} \delta(y_t = \tau \wedge y_i = u) Q'_t \exp(F_{i+1:L}(y, X, \theta)).$$

Like the forward matrix $\alpha(u,i)$ and backward matrix $\beta(u,i)$, $\alpha^\tau(u,i)$ and $\beta^\tau(u,i)$ may also be calculated by dynamic programming. In particular, given the initial conditions $\alpha^\tau(u,1) = Q'_1 \delta(u = \tau)\alpha(u,1)$ and $\beta^\tau(u,L) = 0$. $\alpha^\tau(u,i)$ and $\beta^\tau(u,i)$ can be computed by the following recurrences:

$$\alpha^\tau(u,i) = \sum_{u'} \Big(\alpha^\tau(u', i-1) + Q'_i \delta(u = \tau)\alpha(u', i-1) \Big) \exp(f_\theta(u', u, X, i))$$
$$\beta^\tau(u,i) = \sum_{u'} \Big(\beta^\tau(u', i+1) + Q'_{i+1} \delta(u' = \tau)\beta(u', i+1) \Big) \exp(f_\theta(u', u, X, i)).$$

Let a and b denote the labels at two adjacent sequence positions, then the gradient of equation (2) with respect to parameter T is

$$\frac{\partial s(P_\theta^l, D^\tau)}{\partial T_{a,b}} = \sum_{i\in[L]} \Big(\tilde{\phi}(a,b,i)\exp(f_\theta(a,b,X,i))\Big)$$

where

$$\tilde{\phi}(a,b,i) = Q_i'\delta(y_i=\tau)\frac{\alpha(a,i-1)\beta(b,i)}{Z(X)} + \frac{\alpha^\tau(a,i-1)\beta(b,i)}{Z(X)} + \frac{\alpha(a,i-1)\beta^\tau(b,i)}{Z(X)} - \frac{\alpha(a,i-1)\beta(b,i)}{Z(X)}\mathcal{C}$$

The gradient of equation (2) with respect to parameter U is:

$$\frac{\partial s(P_\theta^l, D^\tau)}{\partial U_{a,h}} = \sum_{i\in[L]} \Big(\tilde{\Phi}(a,i)A_{a,h}(X,i,W)\Big),$$

where

$$\tilde{\Phi}(a,i) = \frac{\alpha^\tau(a,i)\beta(a,i)}{Z(X)} + \frac{\alpha(a,i)\beta(a,i)^\tau}{Z(X)} - \frac{\alpha(a,i)\beta(a,i)}{Z(X)}\mathcal{C} \tag{3}$$

S2.3 Calculation of gradient by back-propagation

The error function from the CRF part at position i for a certain label u is

$$E_i(u) = \sum_{\mu=0}^{d}\sum_{l=0}^{\mu} \mathcal{Y}_{\mu l}\Big(\tilde{\phi}_s^l(\mu,i)v(P_\theta^{\mu-l}, D^{!\tau}) + s(P_\theta^l, D^\tau)\tilde{\phi}_v^{\mu-l}(\mu,i)\Big),$$

where $\tilde{\phi}_s^l$ and $\tilde{\phi}_v^{u-l}$ are derived according to equation (3) with respect to function $s(P_\theta^l, D^\tau)$ and $v(P_\theta^{\mu-l}, D^{!\tau})$, respectively. As show in Fig. 2, we can calculate the neuron error values as well as the gradients at the k-th layer by back-propagation as follows:

$$E_i^k(h) = \eta(H_i^k(h)) * \sum_u \Big(E_i(u) * U_{a,h}) \qquad \text{if } k = K$$

$$E_i^k(h) = \eta(H_i^k(h)) * \sum_{n=-N_k}^{N_k}\sum_{h'=1}^{M_{k+1}} \Big(E_{i+n}^{k+1}(h') * W_n^k(h',h)\Big) \qquad \text{if } k < K,$$

where η is the derivative of the activation function π. In particular, it is $\eta(x) = (1-x)x$ and $\eta(x) = 1 - x * x$ for the sigmoid and tanh function, respectively. E^k is the neuron error value matrix at the k-th layer, with dimension $L \times M_k$. Finally, the gradient of the parameter W at the k-th layer is

$$\nabla_{W_n^k(h,h')} = \sum_{i=1}^{L} \Big(E_i^{k+1}(h) * H_{i+n}^k(h')\Big)$$

S3. Performance comparison with the state-of-the-art predictors

Programs to compare. Since our method is *ab initio*, we do not compare it with consensus-based or template-based methods. Instead, we compare our method with the following ab initio predictors: (i) for ACC prediction, we compare to SPINE-X [7] and ACCpro5-ab [13]. SPINE-X uses neural networks (NN) while ACCpro5-ab uses bidirectional recurrent neural network (RNN); (ii) for DISO prediction, we compare to DNdisorder [6] and DisoPred3-ab [10]. DNdisorder uses deep belief network (DBN) while DisoPred3-ab uses support vector machine (SVM) and NN for prediction; (iii) for SS8 prediction, we compare our method with SSpro5-ab [13] and RaptorX-SS8 [20]. SSpro5-ab is based on RNN while RaptorX-SS8 uses conditional neural field (CNF) [16]. We cannot evaluate Zhous method [21] since it is not publicly available.

Overall evaluation. Here we only compare our AUC-trained DeepCNF model (trained by the JPRED data) to the other state-of-the-art methods on the CASP and CAMEO datasets. As shown in Tables 1 to 3, our AUC-trained DeepCNF model outperforms the other predictors on all the three sequence labeling problems, in terms of the Qx accuracy, Mcc and AUC. When the label distribution is highly imbalanced, our method greatly exceeds the others in terms of Mcc and AUC. Specifically, for DISO prediction on the CASP data, our method achieves 0.53 Mcc and 0.88 AUC, respectively, greatly outperforming DNdisorder (0.37 Mcc and 0.81 AUC) and DisoPred3_ab (0.47 Mcc and 0.84 AUC). For SS8 prediction on the CAMEO data, our method obtains 0.42 Mcc and 0.83 AUC, respectively, much better than SSpro5_ab (0.37 Mcc and 0.78 AUC) and RaptorX-SS8 (0.38 Mcc and 0.79 AUC).

sensitivity, specificity, and precision. Tables 4 and 5 list the sensitivity, specificity, and precision on each label obtained by our method and the other competing methods evaluated on the merged CASP and CAMEO data. Overall, at a high specificity level, our method obtains compatible or better precision and sensitivity for each label, especially for those rare labels such as G, I, B, S, T for SS8, and disorder state for DISO. Taking SS8 prediction as an example, for pi-helix (I), our method has sensitivity and precision 0.18 and 0.33 respectively, while the second best method obtains 0.03 and 0.12, respectively. For beta-bridge (B), our method obtains sensitivity and precision 0.13 and 0.42, respectively, while the second best method obtains 0.07 and 0.34, respectively.

Table 1. Performance of solvent accessibility (ACC) prediction on the CASP and CAMEO data. Sens, spec, prec, Mcc and AUC are averaged on the 3 labels. The best values are shown in bold.

	CASP						CAMEO					
Method	Q3	Sens	Spec	Prec	Mcc	AUC	Q3	Sens	Spec	Pre	Mcc	AUC
OurMethod	**0.69**	**0.65**	**0.82**	**0.64**	**0.47**	**0.82**	**0.66**	**0.62**	**0.81**	**0.62**	**0.43**	**0.80**
SPINE-X	0.63	0.59	0.80	0.59	0.42	0.78	0.61	0.58	0.78	0.57	0.39	0.75
ACCpro5_ab	0.62	0.58	0.81	0.57	0.41	0.76	0.59	0.55	0.79	0.55	0.36	0.73

Table 2. Performance of order/disorder (DISO) prediction on the CASP and CAMEO data.

Method	CASP						CAMEO					
	Q2	Sens	Spec	Prec	Mcc	AUC	Q2	Sens	Spec	Pre	Mcc	AUC
OurMethod	**0.94**	**0.74**	**0.74**	**0.75**	**0.53**	**0.88**	**0.94**	**0.73**	**0.73**	**0.74**	**0.47**	**0.86**
DisoPred3_ab	0.94	0.67	0.67	0.72	0.47	0.84	0.94	0.71	0.71	0.71	0.42	0.83
DNdisorder	0.94	0.73	0.73	0.70	0.37	0.81	0.94	0.72	0.72	0.68	0.36	0.79

Table 3. Performance of 8-state secondary structure (SS8) prediction on the CASP and CAMEO data.

Method	CASP						CAMEO					
	Q8	Sens	Spec	Prec	Mcc	AUC	Q8	Sens	Spec	Pre	Mcc	AUC
OurMethod	**0.71**	**0.48**	**0.96**	**0.56**	**0.44**	**0.85**	**0.69**	**0.45**	**0.95**	**0.54**	**0.42**	**0.83**
RaptorX-SS8	0.65	0.42	0.95	0.50	0.41	0.81	0.64	0.40	0.94	0.48	0.38	0.79
SSpro5_ab	0.64	0.41	0.95	0.48	0.40	0.79	0.62	0.38	0.94	0.46	0.37	0.78

Table 4. Sensitivity, specificity, and precision of each solvent accessibility (ACC) label, tested on the combined CASP and CAMEO data.

ACC	Sensitivity			Specificity			Precision		
Label	Our	SpX*	Acc5**	Our	SpX	Acc5	Our	SpX	Acc5
B	**0.77**	0.74	0.75	**0.82**	0.81	0.80	**0.67**	0.63	0.62
M	**0.45**	0.36	0.34	**0.80**	0.78	0.79	**0.54**	0.48	0.46
E	**0.71**	0.67	0.63	**0.82**	0.79	0.80	**0.67**	0.62	0.61

* SPINEX, ** ACCpro5_ab

Table 5. Sensitivity, specificity, and precision of each disorder label on the combined CASP and CAMEO data.

DISO	Sensitivity			Specificity			Precision		
Label	Our	Diso*	DN**	Our	Diso	DN	Our	DISO	DN
0	**0.96**	0.96	0.89	0.51	0.41	**0.55**	**0.95**	0.94	0.93
1	0.51	0.41	**0.55**	**0.96**	0.96	0.89	**0.54**	0.51	0.47

* DisoPred3_ab; ** DNdisorder

Table 6. Sensitivity, specificity, and precision of each 8-state secondary structure label on the combined CASP and CAMEO data.

SS8	Sensitivity			Specificity			Precision		
Label	Our	Rapt*	SSp5**	Our	Rapt	SSp5	Our	Rapt	SSp5
H	**0.91**	0.89	0.90	0.92	**0.93**	0.93	**0.85**	0.84	0.84
G	**0.28**	0.21	0.19	**0.99**	0.98	0.97	**0.47**	0.43	0.41
I	**0.18**	0.03	0.02	**0.99**	0.98	0.98	**0.33**	0.12	0.06
E	**0.84**	0.78	0.77	**0.94**	0.91	0.89	**0.73**	0.72	0.69
B	**0.13**	0.05	0.07	**0.99**	0.99	0.99	**0.42**	0.33	0.34
T	**0.56**	0.49	0.51	**0.95**	0.93	0.93	**0.56**	0.50	0.49
S	**0.29**	0.21	0.18	**0.97**	0.96	0.97	**0.51**	0.43	0.45
L	0.61	0.62	**0.63**	0.86	0.86	**0.87**	**0.58**	0.58	0.54

* RaptorX-SS8; ** SSpro5_ab

S4. Complexity Analysis

The gradient of the labelwise accuracy function is derived in [**?**]. While all the three training methods have the same space complexity $O(|\Sigma| \cdot L)$, their time complexity is different. Specifically, the time complexity of calculating log-likelihood, labelwise accuracy, and the polynomial approximation of AUC is $O(|\Sigma|^2 \cdot L)$, $O(|\Sigma|^2 \cdot L)$ and $O(d^2 \cdot |\Sigma|^3 \cdot L)$, respectively. Since DCNN is used in DeepCNF, we may not be able to solve the training problem to global optimum. Instead we use the L-BFGS [**?**] algorithm to find a suboptimal solution.

The running time of maximum-AUC training is approximately linear when the sequence length is much larger than the number of labels and the degree of the polynomial approximation. When the degree d is larger, we can approximate the loss function better, but the approximation itself becomes less smooth and more challenging to optimize. A large d also increases model complexity, which makes it easier to overfit. In our experiments, along with the increase of d, the training AUC always improves, but the test AUC drops after $d = 15$.

References

1. William R Atchley, Jieping Zhao, Andrew D Fernandes, and Tanja Drüke. Solving the protein sequence metric problem. *Proceedings of the National Academy of Sciences of the United States of America*, 102(18):6395–6400, 2005.
2. Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, 2013.
3. Gapped BLAST. Psi-blast: a new generation of protein database search programs altschul. *Stephen F*, pages 3389–3402.
4. Cyrus Chothia. The nature of the accessible and buried surfaces in proteins. *Journal of molecular biology*, 105(1):1–12, 1976.
5. Mojie Duan, Min Huang, Chuang Ma, Lun Li, and Yanhong Zhou. Position-specific residue preference features around the ends of helices and strands and a novel strategy for the prediction of secondary structures. *Protein Science*, 17(9):1505–1512, 2008.
6. Jesse Eickholt and Jianlin Cheng. Dndisorder: predicting protein disorder using boosting and deep networks. *BMC bioinformatics*, 14(1):88, 2013.
7. Eshel Faraggi, Bin Xue, and Yaoqi Zhou. 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: Structure, Function, and Bioinformatics*, 74(4):847–856, 2009.
8. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
9. Shuichi Hirose, Kana Shimizu, Satoru Kanai, Yutaka Kuroda, and Tamotsu Noguchi. Poodle-l: a two-level svm prediction system for reliably predicting long disordered regions. *Bioinformatics*, 23(16):2046–2053, 2007.
10. David T Jones and Domenico Cozzetto. Disopred3: precise disordered region predictions with annotated protein-binding activity. *Bioinformatics*, 31(6):857–863, 2015.
11. Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.
12. Jianzhu Ma and Sheng Wang. Acconpred: Predicting solvent accessibility and contact number simultaneously by a multitask learning framework under the conditional neural fields model. *BioMed Research International*, 2015, 2015.
13. Christophe N Magnan and Pierre Baldi. Sspro/accpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics*, 30(18):2592–2597, 2014.
14. Jens Meiler, Michael Müller, Anita Zeidler, and Felix Schmäschke. Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks. *Molecular modeling annual*, 7(9):360–369, 2001.
15. Bohdan Monastyrskyy, Krzysztof Fidelis, John Moult, Anna Tramontano, and Andriy Kryshtafovych. Evaluation of disorder predictions in casp9. *Proteins: Structure, Function, and Bioinformatics*, 79(S10):107–118, 2011.
16. Jian Peng, Liefeng Bo, and Jinbo Xu. Conditional neural fields. In *Advances in neural information processing systems*, pages 1419–1427, 2009.
17. Ning Qian and Terrence J Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of molecular biology*, 202(4):865–884, 1988.

18. Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.
19. Yen Hock Tan, He Huang, and Daisuke Kihara. Statistical potential-based amino acid similarity matrices for aligning distantly related protein sequences. *Proteins: Structure, Function, and Bioinformatics*, 64(3):587–600, 2006.
20. Zhiyong Wang, Feng Zhao, Jian Peng, and Jinbo Xu. Protein 8-class secondary structure prediction using conditional neural fields. *Proteomics*, 11(19):3786–3792, 2011.
21. Jian Zhou and Olga G Troyanskaya. Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. *arXiv preprint arXiv:1403.1347*, 2014.