Deep Recurrent Conditional Random Field Network for Protein Secondary Prediction

Alexander Rosenberg Johansen
Technical University of Denmark, DTU Compute
alexander@munk.ai

Casper Kaae Sønderby
University of Copenhagen, Bioinformatics Centre
casper.sonderby@bio.ku.dk

Søren Kaae Sønderby
University of Copenhagen, Bioinformatics Centre
soren.sonderby@bio.ku.dk

Ole Winther
Technical University of Denmark, DTU Compute
olwi@dtu.dk

ABSTRACT

Deep learning has become the state-of-the-art method for predicting protein secondary structure from only its amino acid residues and sequence profile. Building upon these results, we propose to combine a bi-directional recurrent neural network (biRNN) with a conditional random field (CRF), which we call the biRNN-CRF. The biRNN-CRF may be seen as an improved alternative to an autoregressive uni-directional RNN where predictions are performed sequentially conditioning on the prediction in the previous time-step. The CRF is instead nearest neighbor-aware and models for the joint distribution of the labels for all time-steps. We condition the CRF on the output of biRNN, which learns a distributed representation based on the entire sequence. The biRNN-CRF is therefore close to ideally suited for the secondary structure task because a high degree of cross-talk between neighboring elements can be expected. We validate the model on several benchmark datasets. For example, on CB513, a model with 1.7 million parameters, achieves a Q8 accuracy of 69.4 for single model and 70.9 for ensemble, which to our knowledge is state-of-the-art. [1]

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Neural networks**; *Batch learning*;

KEYWORDS

Secondary protein structure; Recurrent Neural Network; Conditional Random Field

1 INTRODUCTION

Current approaches to predict secondary protein structures, from amino acid residues and sequence profiles, use machine learning methods [14, 24]. Recently, artificial neural network based methods [18] has emerged and received state of the art performance [6, 19, 26, 29, 31]. These approaches have arisen as the challenge of annotating secondary protein structures is predictable from conformations along the sequence of the amino acid and sequence profiles [29].

Recent methods propose using either; convolutions, which work as fixed-sized sliding windows, combined with variants of autoregressive models [6, 31], convolutions with a conditional random field (CRF) output layer [29], a bi-directional RNN [26] or convolutions combined with a bi-directional RNN and multi-task learning [19].

Thus, previous work has had a tendency to either focus on integrating information over longer distances with a RNN or modeling the sequence likelihood in secondary structures with autoregressive or CRF components. We propose to combine these methods in an end-to-end trainable artificial neural network. A method that learns to model temporal information along the entire protein structure, using a bi-directional recurrent neural network (biRNN), as well as inferring the most likely sequence of proteins, using a conditional random field (CRF). We argue that the biRNN part of the model learns a hidden representation that integrates information from formations along the sequence, and the CRF uses that representation to model the joint distribution of the secondary structure for all labels. Furthermore, by modeling the labels jointly rather than independent conditioned on the hidden states of the biRNN means that it will produce predictions closer to what is observed in actual proteins. E.g. a prediction alpha-turn-alpha-turn will be highly unlikely.

The recurrent neural network (RNN), as well as the convolutional neural network (CNN), are modular architectures for spatial feature extraction. Both these methods have proven the preferred way for modelling sequences with deep learning methods [2, 11]. The RNN subjects the network to reuse its weights by recursively applying them across a sequence, based on a prior that the input domain will have a time dependency, e.g. along a series of words or amino acids. Having only one set of parameters for every time-step along the entire sequence lowers the parameter space significantly, as opposed to a standard feed forward network, which reduces overfitting. As the RNN applies the same weights repeatedly a fixed sequences length is not required. [11] Furthermore, modern RNNs use gating mechanisms, such as the long-short term memory cell (LSTM) [13] or the gated recurrent unit (GRU) [7, 9], to exhibit an internal memory state. An internal memory state makes the RNN capable of modeling dynamical temporal behavior over longer

[1] Codebase available at: github.com/alrojo/biRNN-CRF

ACM-BCB'17, August 20-23, 2017, Boston, MA, USA.

978-1-4503-4722-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3107411.3107489

sequences and enabling the neural network to express long-term dependencies.

When predicting a sequence of labels the probability of a transition between labels may depend not only on the current observation, but also on past and future observations. The Conditional Random Field (CRF) [17] allow us to incorporate transitions between labels by modeling the joint probability of the entire sequence. The CRF uses a transition between neighboring states to achieve such. As each label becomes dependent on the transition of its neighbors, so will the neighbors become dependent on the transition of their neighbors, creating a chain of dependencies that stretches the entire sequence for each label (also known as the Markov assumption).

Despite the attractive qualities of neural network architectures, building state-of-the-art neural networks is prohibitively expensive. Luckily, current graphics processing units (GPUs), paired with optimized implementations of neural network architectures make it convenient to train state-of-the-art neural networks. Furthermore, easy-to-use libraries for wrapping GPU implementations, such as the Python based TensorFlow [1] (our preferred choice) or Keras [8], empowers researchers to quickly iterate and build state-of-the-art neural networks with limited knowledge of GPUs.

The specific contribution of this paper are as follows: we present how a recurrent neural network, with memory cells, and a conditional random field can be successfully combined and trained on secondary protein structure prediction given a sequence of amino acids and sequence profiles. Our results find that using a CRF improves our baseline model and makes it comparable state-of-the-art neural networks with much higher parameter count and complexity.

2 RELATED WORK

The use of deep learning for sequence analysis is not new. Since the break-through of deep learning methods, which came from the increase in available data and computational power [16], new and previously underutilized deep learning methods [3, 4, 13, 23] have achieved significant improvements over state-of-the-art methods within their respective sequential field. Such as Machine Translation [4, 30], Speech Recognition [2] and Audio Generation [28]. Tasks in computational biology that fits well into these deep learning methods have experienced a spill-over, resulting in state-of-the-art improvements as well [25]. More specifically, CRF is a method originally tested on sequential labeling in natural language processing for POS tagging [17], but has later been found useful to the, mathematically, much similar problem of predicting secondary protein structures [29]. In our work, we attempt to accomplish a similar feat. As previous effort in secondary structure prediction has yet to combine the representational power of RNNs with benefits of modeling the joint probability of the entire sequence.

3 MATERIALS AND METHODS

This section describes the datasets used, evaluation metric and the deep learning architecture.

3.1 Dataset

We use four different dataset combinations: 1) the CB6133 with the official train, validation and test split, 2-4) the filtered version of the CB6133 as train and validation set and either the CB513, CASP10 or CASP11 as test set [20, 21, 31] [2].

3.1.1 CB6133. In the CB6133 (non-filtered) dataset [3] we use the amino acid residues (features $[0:21)$) and sequence profiles (features $[35:56)$) giving a total of 42 input features at each time step. For labels, we use the secondary structure notation (features $[22:30)$) at every time step and mask the loss using the `NoSeq` label (feature [30]). We train and evaluate on the official dataset splits (training: $[0:5600)$, validation: $[5877:6133)$, test: $[5605:5877)$).

3.1.2 CB6133 filtered with CB513 & CASP10 & CASP11. For training and validation, we use the filtered-CB6133 dataset [4] with the same features as the non-filtered version. As no official validation set is provided we optimized hyperparameters using 256 random samples from the training set. For testing we use CB513 [5], CASP10 and CASP11 [6].

3.2 Evaluation

All models are evaluated using tagging accuracy reporting the following scores: 3-class accuracy (Q3; α-helix, β-strand and coil) and 8-class accuracy (Q8; α-helix={G for 310-helix, H for α'-helix, I for π-helix}; β-strand={E for β'-strand, B for β-bridge} and coil={T for β-turn, S for high-curvature-loop, L for irregular}) tagging accuracy.

3.3 Our Model

The architecture of the network is defined by three different types of learnable layers and one regularization layer, the fully connected, the recurrent layer with gated recurrent unit (GRU) memory cells, a conditional random field (CRF) layer and a Bernoulli dropout regularization layer. Below, we describe the different layer types used for our network in more detail.

3.4 Fully Connected

The fully connected layer, also known as a dense layer or a multi layer perceptron (MLP) [22], is a non-linear transformation of the previous layer, $\ell - 1$. The first layer, $\ell = 0$, is considered the input layer. In our case the input layer is sequence profiles and a one-hot encoding of the amino acid. The standard MLP is defined in the following linear algebraic operation

$$z_t^\ell = h_t^{\ell-1}\theta^\ell + b^\ell, \tag{1}$$

$$h_t^\ell = a(z_t^\ell) \tag{2}$$

where h^l is the current layer, θ^ℓ is the weight matrix and b^ℓ is the bias used to compute the linear combination of the input; z^ℓ. Notice that we subscript t as this is performed individually for every state at all time-steps. A non-linear activation function, $a(z^\ell)$ is applied element-wise to the linear combination of the input, which results in the next layer; h^ℓ.

Most commonly used functions for the element-wise activation function $a(z)$ includes the Logistic Sigmoid $a(z) = 1/(1 + e^{-z})$ and

[2] http://www.princeton.edu/~jzthree/datasets/ICML2014/
[3] http://www.princeton.edu/~jzthree/datasets/ICML2014/cullpdb+profile_6133.npy.gz
[4] http://www.princeton.edu/~jzthree/datasets/ICML2014/cullpdb+profile_6133_filtered.npy.gz
[5] http://www.princeton.edu/~jzthree/datasets/ICML2014/cb513+profile_split1.npy.gz
[6] Generously supplied by the authors of [29] and [19] and formatted identically to CB6133/CB513

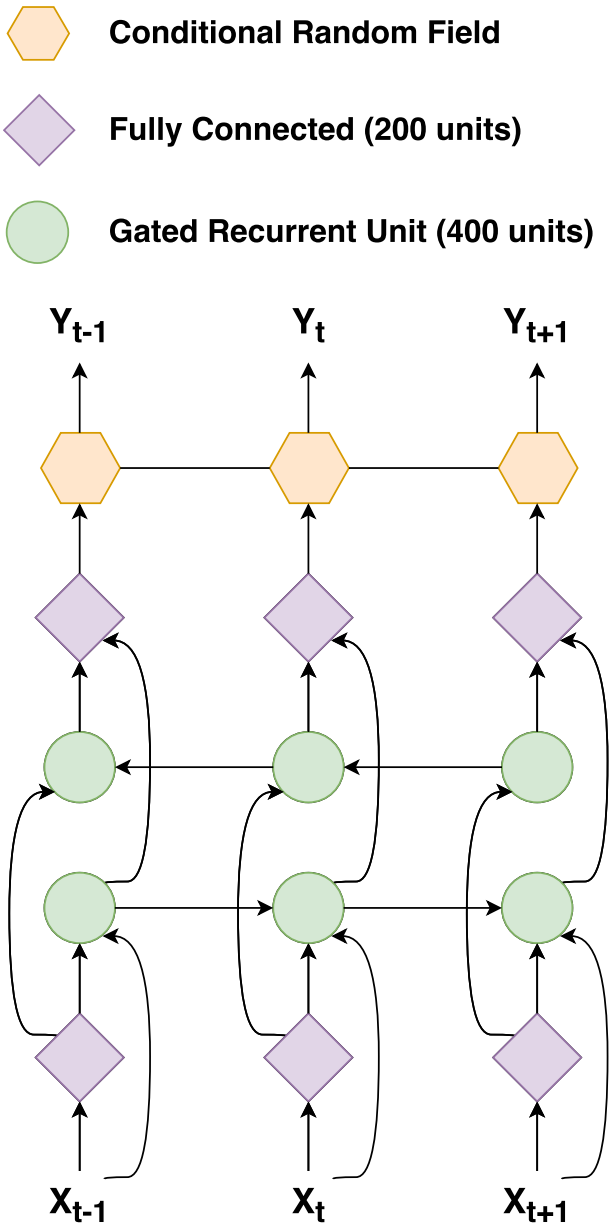

Figure 1: Our proposed model: The biRNN-CRF

the Hyperbolic Tangent $a(z) = (e^z - e^{-z})/(e^z + e^{-z})$. Non-saturating functions such as the Rectifier Linear Unit (ReLU) $a(z) = max(0, z)$ (our preferred choice) has become popular as their gradients do not vanish and it is computationally easier, which makes optimizing the network faster using stochastic gradient descent [10, 16].

As regularization technique, to avoid overfitting, we add Bernoulli dropouts [27], which is a non-trainable layer. Such that

$$h^{\ell} = h^{\ell-1} \odot p \tag{3}$$

where $p_i \in [0, 1]$ is sampled from a Bernoulli distribution with chance k for $p_i = 0$ and $1 - k$ for $p_i = 1$. Most commonly $k = 0.5$, which is also the case in our Bernoulli dropout layer. The purpose of the Bernoulli Dropout is to introduce noise such that the model becomes less dependent on combinations of specific activations.

3.5 Bi-directional Recurrent Neural Network

A recurrent neural network (RNN) is a type of neural network layer that repeatedly uses the same weights along a sequence of data with a prior that the sequence dimension contains a useful signal [11]. The gated recurrent unit (GRU), which we use, is a type of computational unit for calculating the hidden representation at every time step of the RNN. The GRU uses memory cells to better model long-term dependencies and improve convergence speed. The GRU is defined as described in Chung, 2014 [9].

$$r_t^{\ell} = \sigma(h_t^{\ell-1} W_r^{\ell} + h_{t-1}^{\ell} U_r^{\ell} + b_r^{\ell}), \tag{4}$$

$$z_t^{\ell} = \sigma(h_t^{\ell-1} W_z^{\ell} + h_{t-1}^{\ell} U_z^{\ell} + b_z^{\ell}), \tag{5}$$

$$\hat{h}_t^{\ell} = \tanh(h_t^{\ell-1} W_h^{\ell} + (r \odot h_{t-1}^{\ell}) U_h^{\ell} + b_h^{\ell}), \tag{6}$$

$$h_t^{\ell} = (1 - z_t^{\ell}) \odot h_{t-1}^{\ell} z_t^{\ell} \odot \hat{h}_t^{\ell} \tag{7}$$

Where $\sigma(z)$ is the sigmoid function $1/(1 + e^{-z})$, $h_t^{\ell-1}$ is the input at time t and h_{t-1}^{ℓ} the hidden state at time $t - 1$. The RNN with GRU is only defined in one direction, h_t^{ℓ} for $t = [1, 2, ..., T]$ where T is sequence length. To utilize information from both directions we compute the backwards RNN with h_t for $t = [T, T - 1, ..., 1]$ and h_t^{ℓ} being dependent on h_{t+1}^{ℓ} instead of h_{t-1}^{ℓ}. To combined both directions, we concatenate the hidden states from the forward and backward pass, such as described by [23], where the hidden states are aligned with their index, $t = [1, 2, ..., T]$, such that.

$$h_t^{\ell} = \begin{bmatrix} \overrightarrow{h_t^{\ell}} \\ \overleftarrow{h_t^{\ell}} \end{bmatrix} \tag{8}$$

where $\overrightarrow{h_t^{\ell}}$ is the forward pass and $\overleftarrow{h_t^{\ell}}$ the backwards pass. This is known as the bi-directional RNN (biRNN).

3.6 Conditional Random Field (CRF)

The CRF is a joint distribution of the labels in the sequence $y = y_1, \ldots, y_T$ given the input sequence $x = x_1, \ldots, x_T$ on the following restricted form

$$p(y|x) = \frac{1}{Z(h)} \prod_{t=1}^{T} \exp \psi_{y_t}(h_t) \prod_{t=1}^{T-1} \exp \phi_{y_t, y_{t+1}}(h_t, h_{t+1}), \tag{9}$$

where $h = h_1, \ldots, h_T$ is the output of the previous hidden layer, $\psi_t = \psi(h_t)$ is a linear model which takes h_t as input and has the number of classes C ($C = 8$ or $C = 3$ in this paper) outputs, $\phi_t = \phi(h_t, h_{t+1})$ is another linear model with C^2 real-valued outputs

$$\psi_t = W_{\psi} h_t + b_{\psi} \tag{10}$$

$$\phi_t = W_{\phi} \begin{bmatrix} h_t \\ h_{t+1} \end{bmatrix} + b_{\phi} \tag{11}$$

and $Z(h)$ is the normalization constant of the distribution. Due to the chain structure, inference can be carried out exactly using dynamic programming in $O(TC^2)$ [5]. During training where (x, y) is observed we need to compute $Z(h)$ for each training sequence as part of the likelihood $p(y|x)$. During prediction where only x

Table 1: Q8 accuracy on: CullPDB, CB513, CASP10, CASP11.

Methods	Q8(%)			
	CullPDB	**CB513**	**CASP10**	**CASP11**
Single Model				
GSN	72.1	66.4	N/A	N/A
DCRNN	N/A	69.4	N/A	N/A
Deep Multi-Scale CNN	N/A	**70.0**	N/A	N/A
biRNN (ours)**	μ=72.5, σ=0.15	μ=68.5, σ=0.12	μ=72.8, σ=0.23	μ=70.1, σ=0.32
biRNN-CRF (ours)**	μ=**73.4**, σ=0.13	μ=69.4, σ=0.16	μ=**73.5**, σ=0.38	μ=**70.8**, σ=0.33
Ensemble				
DeepCNF	N/A*	68.3	71.8	72.3
DCRNN	73.2	69.7	**76.9**	**73.1**
Deep Multi-Scale CNN	N/A	70.6	N/A	N/A
biRNN-CRF (ours)***	74.6	70.8	74.7	72.2
biRNN-CRF (ours)****	**74.8**	**70.9**	74.9	72.4

Table 2: Q3 accuracy on: CullPDB, CB513, CASP10, CASP11.

Methods	Q3(%)			
	CullPDB	**CB513**	**CASP10**	**CASP11**
Single Model				
biRNN (ours)**	μ=83.6, σ=0.14	μ=81.8, σ=0.13	μ=84.1, σ=0.20	μ=81.3, σ=0.28
biRNN-CRF (ours)**	μ=**84.2**, σ=0.13	μ=**82.2**, σ=0.16	μ=**84.2**, σ=0.29	μ=**81.7**, σ=0.25
Ensemble				
DeepCNF	N/A*	82.3	84.4	84.7
DCRNN	N/A	**84.0**	**87.8**	**85.3**
biRNN-CRF (ours)***	**85.0**	83.2	85.0	83.2
biRNN-CRF (ours)****	**85.0**	83.3	85.2	82.8

is observed we can calculate either the most probably sequence $\arg\max_y p(y|x)$ (using the Viterbi decoding algorithm) or the marginal probabilities $p(y_t|x)$, $t = 1, \ldots, T$. In all our reported results, we use the maximum marginal probability prediction since this gives the smallest expected element-wise error [5]. Viterbi decoding is relevant when the objective is the lowest possible sequence-wise prediction error.

The most straight forward alternative to the CRF is an autoregressive model:

$$p(y|x) = p(y_1|x) \prod_{t=2}^{T} p(y_t|y_1, \ldots, y_{t-1}, x) \tag{12}$$

which can be implemented by replacing the CRF layer in figure figure 1 with T independent C-dimensional softmax units and adding y_{t-1} as an extra dimension to $\overrightarrow{h_t}$. The disadvantage of this model compared to the CRF is two-fold: 1) computing the most probable sequences and marginals have exponential complexity in the sequence length and 2) predictions are slow because we need to recompute a large part of the model at each time step.

3.7 Architecture and details of learning

As depicted in figure 1 the network contains five layers. The first layer is a fully connected layer, the second is a bi-directional RNN with GRU memory cells. The third layer is a dropout layer, the fourth is a fully connected layer and the fifth is a conditional random field layer that provides the conditional probability of the 8-way classification problem. Note that the input has a skip connection to the bidirectional RNN. Both of the fully connected layers have 200 hidden units each and use ReLU as their activation function. Both of the recurrent GRU layers have 400 hidden units each. To train the model we minimize the loss which is the negative log likelihood of the model parameters collectively denoted by θ over a training set (X, Y) of n paired input and output sequences $X = x^{(1)}, \ldots, x^{(n)}$ and $Y = y^{(1)}, \ldots, y^{(n)}$:

$$\mathrm{Loss}_\theta(\mathrm{X}, \mathrm{Y}) = -\sum_{(\mathrm{x},\mathrm{y})\in(\mathrm{X},\mathrm{Y})} \log \mathrm{p}(\mathrm{y}|\mathrm{x}, \theta)\,. \tag{13}$$

We train our model with the first order method: stochastic gradient descent (SGD) with mini batches of size 64. SGD works by utilizing chain ruling to take the partial derivative of the loss function with respect to each weight vector in the network, and use the derivative to update the weights. We use a version of SGD known as Adam [15] with default parameters and a learning rate of $1e-3$. Adam uses historic information to adapt the learning rate for every parameter while training. To avoid exploding gradients [11] we normalize and clip the gradients if its norm exceeds a threshold of 1. We train our neural network on an Nvidia GeForce GTX Titan X GPU using the python built TensorFlow library [1] to compile to CUDA (a GPU interpretable language).

Table 3: Recall and precision of biRNN-CRF**, DeepCNF ensemble and GSN on the CB6133 dataset. SS8 label corresponds to the eight secondary protein structure labels as described in section 3.2**

SS8 label	Recall			Precision		
	biRNN-CRF	DCRNN	GSN	biRNN-CRF	DCRNN	GSN
L	**0.672**	0.662	0.633	**0.606**	0.589	0.541
B	**0.099**	0.049	0.001	**0.614**	0.596	0.500
E	0.853	**0.862**	0.823	**0.803**	0.792	0.748
G	**0.343**	0.311	0.133	**0.530**	0.434	0.496
I	0.000	0.000	0.000	0.000	0.000	0.000
H	**0.936**	0.927	0.935	0.878	0.878	0.828
S	**0.288**	0.275	0.159	**0.537**	0.518	0.423
T	**0.604**	0.572	0.506	**0.589**	0.577	0.548

Table 4: Recall and precision of biRNN-CRF** and DeepCNF ensemble on the CB513 dataset.**

SS8 label	Recall		Precision	
	biRNN-CRF	DeepCNF	biRNN-CRF	DeepCNF
L	0.657	0.657	**0.597**	0.571
B	**0.042**	0.026	**0.515**	0.433
E	**0.835**	0.833	**0.760**	0.748
G	**0.348**	0.260	0.456	**0.490**
I	0.000	0.000	0.000	0.000
H	**0.931**	0.904	0.847	**0.849**
S	**0.265**	0.255	**0.549**	0.487
T	**0.554**	0.528	**0.557**	0.53

4 EXPERIMENTS

We evaluate the Q3 and Q8 performance of our neural network (biRNN-CRF) on the four datasets CB6133 train/val/test, CB6133 filtered with either CB513, CASP10 or CASP11 as test set (elaborated in section 3.1). The Q3 and Q8 accuracy is elaborated in section 3.2.

On the Q8 problem we further supply recall and precision for the CB6133 and the CB6133 filtered + CB513 dataset.

We compute the Q3 for our models by summing over our predictions from the Q8 into their respective Q3 classes: helix (H), strand (E) and coil (C).

The deep learning models we benchmark against are the GSN [31], DeepCNF [29], DCRNN [19] and the Deep Multi-Scale CNN [6].

4.1 Training details

We train two different type of models, the biRNN-CRF as illustrated in figure 1 and the biRNN, which is the exact same setup as the biRNN-CRF, but using the sequence independent cross entropy instead of the CRF layer and loss function.

We train 10 models for both the biRNN and biRNN-CRF to calculate single model mean/standard deviation as well as ensembles. All models are trained with the same setup, only seed for initializing weights differ. We use early stopping, based on the validation set, to pick the optimal set of weights.

4.2 Results

In single model performance, the biRNN-CRF outperforms the baseline biRNN using cross entropy across all datasets with between **0.1-0.9%**, as shown in table 1 and table 2. In our benchmark against state-of-the-art models using deep learning methods, our model achieves a new state-of-the-art performances on the CB6133 Q8: **74.8%(+1.6%)**, CB513 Q8: **70.9%(+0.3%)** and we also achieve **85.0%** on CB6133 Q3. Since we have not found any previously published results using deep learning methods on the CB6133 Q3 dataset we assume that it is also state-of-the-art. However, on the CASP10 and CASP11 datasets the DCRNN model significantly outperforms ours.

Furthermore, precision and recall for the CB6133 and CB513 is illustrated in table 3 and table 4. Here we find that our ensemble model outperforms or equals most previously published deep learning models.

4.3 Notes on models and ensemble

Not all previously published work we use for comparison has provided single model and ensemble results for all datasets. Because of such, we either supply a N/A if one or more results are missing or leave out the method from the table if all results are missing. * The DeepCNF article uses a different test set than the official for their CB6133 results. ** Mean, μ, and standard deviation, σ, are based on our 10 trained models as described in section 4.1. As there is not a single adopted way of ensembling models within the secondary protein structure literature, we employ our own ensembling strategy. Our ensemble is based on averaging predictions from our 10 trained models used for **. For the *** ensemble, we sample the weights from the best performing epoch, based on the validation set. The **** ensemble is the same as ***, but samples the top three weights instead of top one to reduce variance in predictions from each model.

5 DISCUSSION

Our results show that the bi-directional recurrent neural network with a conditional random field (biRNN-CRF) can perform on pair with vastly more advanced architectures. The biRNN model can be seen as a Spartan version of the DCRNN [19] ,which makes the DCRNN interesting for comparison. The biRNN has about a quarter of the parameters of the DCRNN (1.7 million vs. >6 million), no cascading convolutions, only one RNN layer and no multi-task training. Increasing depth and network size has in recent literature proven successful strategy [12], which might point to why our baseline (biRNN) is trailing the DCRNN. However, as we add the CRF layer we close the gap between the models. This result might indicate that the temporal benefits of a recurrent neural network with the conditioning power of a CRF is helpful when tackling sequential problems in computational biology. Moreover, there is no reason to conclude that the benefits obtained from the depth and capacity of the DCRNN are mutually exclusive with the benefits from the CRF layer. However, we will leave combining deep models for secondary protein structure prediction with a CRF layer for future research.

ACKNOWLEDGMENTS

This research was supported by the Novo Nordisk Foundation and the NVIDIA Corporation with the donation of TITAN X and Tesla K40 GPUs.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Gordon Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR* abs/1603.04467 (2016). https://arxiv.org/abs/1603.04467 Software available from tensorflow.org.

[2] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. 2015. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *CoRR* abs/1512.02595 (2015). http://arxiv.org/abs/1512.02595

[3] J. Ba, J. Kiros, and G. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[4] D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). https://arxiv.org/abs/1409.0473

[5] C. Bishop. 2006. *Pattern recognition and machine learning*. Springer.

[6] A. Busia, J. Collins, and N. Jaitly. 2016. Protein Secondary Structure Prediction Using Deep Multi-scale Convolutional Neural Networks and Next-Step Conditioning. *CoRR* abs/1611.01503 (2016). http://arxiv.org/abs/1611.01503

[7] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

[8] F. Chollet. 2015. Keras: Theano-based deep learning library. (2015). https://github.com/fchollet/keras

[9] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). http://arxiv.org/abs/1412.3555

[10] X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep Sparse Rectifier Neural Networks.. In *Aistats*, Vol. 15. 275.

[11] A. Graves. 2012. Supervised sequence labelling with recurrent neural networks. *Springer* (2012).

[12] K. He, X. Zhang, S. Ren, and J. Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385

[13] S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. DOI: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[14] D. Jones. 1999. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology* 292, 2 (1999), 195 – 202. DOI: http://dx.doi.org/10.1006/jmbi.1999.3091

[15] D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. International Conference on Learning Representation.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[17] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289. http://dl.acm.org/citation.cfm?id=645530.655813

[18] Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[19] Z. Li and Y. Yu. 2016. Protein Secondary Structure Prediction Using Cascaded Convolutional and Recurrent Neural Networks. *arXiv preprint arXiv:1604.07176* (2016).

[20] B. Monastyrskyy, A. Kryshtafovych, J. Moult, A. Tramontano, and K. Fidelis. 2014. Assessment of protein disorder region predictions in CASP10. *Proteins: Structure, Function, and Bioinformatics* 82, S2 (2014), 127–137.

[21] J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano. 2014. Critical assessment of methods of protein structure prediction (CASP)âĂŤround x. *Proteins: Structure, Function, and Bioinformatics* 82, S2 (2014), 1–6.

[22] D. Ruck, S. Rogers, M. Kabrisky, M. Oxley, and B. Suter. 1990. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *Neural Networks, IEEE Transactions on* 1, 4 (1990), 296–298.

[23] M. Schuster and K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.* 45, 11 (Nov. 1997), 2673–2681. DOI: http://dx.doi.org/10.1109/78.650093

[24] M. Singh. 2005. Predicting protein secondary and supersecondary structure. *Handbook of Computational Molecular Biology, hapman & Hall CRC Computer and Information Science Series* (2005).

[25] S. Sønderby, C. Sønderby, H. Nielsen, and O. Winther. 2015. Convolutional LSTM networks for subcellular localization of proteins. In *International Conference on Algorithms for Computational Biology*. Springer, 68–80.

[26] S. Sønderby and O. Winther. 2014. Protein secondary structure prediction with long short term memory networks. *arXiv preprint arXiv:1412.7828* (2014).

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* (2014). http://dl.acm.org/citation.cfm?id=2627435.2670313

[28] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *CoRR* abs/1609.03499 (2016). http://arxiv.org/abs/1609.03499

[29] S. Wang, J. Peng, J. Ma, and J. Xu. 2016. Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports* 6 (2016).

[30] Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, and others. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016). https://arxiv.org/abs/1609.08144

[31] J. Zhou and O. Troyanskaya. 2014. Deep Supervised and Convolutional Generative Stochastic Network for Protein Secondary Structure Prediction.. In *ICML*. 745–753.