02456 DEEP LEARNING, DTU COMPUTE, FALL 2017

DEEP LEARNING ARCHITECTURES FOR PROTEIN SECONDARY STRUCTURE PREDICTION

Blaabjerg, Lasse (s13214)

Technical University of Denmark (DTU)

ABSTRACT

In this project we train three different neural network models on a data set of protein primary sequences in order to predict the correct secondary structure of each amino acid in the polypeptide chain. The models trained were i) a sliding window feed-forward neural network ii) a bi-directional recurrent neural network iii) a convolutional layer feeding into a bi-directional recurrent neural network. Several modern techniques such as batch normalization, gradient clipping, Bernouli drop-out, L^2-regularization and leaky rectifier activation functions are implemented in order to improve predictive performance. Surprisingly, it is found that the more advanced models were not able to improve the performance of the simple sliding window feed-forward network. These results are supported by the findings of other researchers and it is therefore concluded that the lack of predictive performance is due to a lack of common higher-order features in the training and the validation data. However, this results is most likely unique to the investigated data and not a general feature of how amino acids interact in nature.

1. INTRODUCTION

Protein secondary structure prediction is one of the classic prediction challenges in computational biology and involves the prediction of how a single amino acid in the polypeptide chain will fold into one of the allowed secondary structures. Traditionally, either eight secondary structure classes (alpha-helix, 3_{10}-helix, π-helix, β-strand, β-bridge, turn, bend and coil) or three secondary structure classes (helix, β-strand and random coil) have been used in predictive modeling.

In recent years, great advances have been made in secondary structure prediction accuracy using increasingly more sophisticated deep learning models and architectures [1]. A variety of different approaches have been developed today using deep learning models both as a stand-alone tool for analysis of the primary amino acid sequence as well as a technique used in combination with other predictive models.

In this paper, we present three different deep learning models with different levels of network complexity in order to illustrate how different deep learning architectures can impact predictive performance. The three deep learning architectures that have been investigated are i) a sliding window feed-forward neural network (FFNN) ii) a bi-directional recurrent neural network (biRNN) iii) a convolutional layer feeding into a bi-directional recurrent neural network (CNNbiRNN).

2. THEORY

2.1. Learning methods

Throughout all three models a common set of learning methods were implemented. The learning methods described below were chosen due to their respective advantages over more traditional methods which was verified experimentally during the development process.

2.1.1. Optimization

The Adam algorithm was used as the preferred optimization method. The Adam algorithm utilizes an adaptive learning rate that is defined by the update rule:

$$\theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \to \theta_t \tag{1}$$

where α is the learning rate and $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first- and second-order momenta which are computed from an exponential moving average of the gradient and the gradient squared using decay-rates β_1 and β_2 [2].

Gradient clipping was implemented on all models in order to avoid the issue of large gradients interfering with optimal search behavior near local minima. The gradients were value clipped to values between -1 and 1.

2.1.2. Loss function

The cross-entropy with L^2-regularization was chosen as the loss function:

$$C = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log h_{nk} + \frac{\lambda}{2N} \sum_{w} w^2 \tag{2}$$

where n is the observation, k is the class, h is the soft-max prediction, y is the class label and λ is the regularization strength [3].

2.1.3. *Activation function*

In terms of activation function, the leaky rectified linear unit was chosen: [4]:

$$f(x) = \max(x, \alpha x) \tag{3}$$

with $\alpha = 0.1$. This activation function was chosen due to its non-vanishing, non-dead gradient properties compared to the classical sigmoid and the standard ReLU activation function respectively.

2.1.4. *Advanced model techniques*

In the two advanced models, two additional techniques were used to increase performance; batch normalization and Bernoulli drop-out. In batch normalization the activations of the previous are normalized during training according to the formula:

$$\hat{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{4}$$

$$\gamma \hat{\mathbf{x}}_i + \beta \rightarrow \mathbf{y}_i \tag{5}$$

where $\mathcal{B}$ is the batch ensemble and γ and β are learned parameters [5]. During validation, the batch statistics are replaced by the populations statistics. Batch normalization attempts to solve the problem of internal covariate shifts in trained models. Traditionally, batch normalization is placed before the non-linearity operation in the layer, however, new research has indicated that performance might be improved in some cases by placing batch normalization after the non-linearity operation [1]. In this project, batch normalization is performed after the non-linearity operation.

Bernoulli drop-outs serve as an additional regularization technique used in combination with the implemented the L^2-regularization. Drop-outs are included in the final hidden dense layer in the advanced models and works by randomly removing hidden nodes during training:

$$\mathbf{h}_{l+1} = \mathbf{h}_l \circ \mathbf{p} \tag{6}$$

where $p \in]0; 1[$ and $\circ$ is the Hadamard product [6]. During validation, all nodes are included with weight magnitudes scaled by $1/p$.

2.2. Feed-forward neural networks

A FFNN is defined by a series of layers interconnected sequentially with each layer containing a number of layer nodes which perform a non-linear transformation of the summed input of activations from the nodes in the previous layer. For a standard FFNN with bias the activation of an arbitrary layer l is defined as:

$$\begin{aligned} \mathbf{z}_{l+1} &= \mathbf{h}_l \theta_{l+1} + \mathbf{b}_{l+1} \\ \mathbf{h}_{l+1} &= a(\mathbf{z}_{l+1}) \end{aligned}$$

where $\mathbf{h}$ is the activation, θ is the weight matrix and a is the activation function [3]. The final prediction is made by normalizing the activation of each output node via the soft-max function [3]:

$$\text{softmax}(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{7}$$

where K is the total the number of output nodes i.e. the number of predicted classes. The FFNN model implemented in this project utilizes a sliding window input such that the input features for a given amino acid contains information about the amino acid itself as well as the neighboring amino acids on both sides.

2.3. Bi-directional recurrent neural networks

A general RNN model consists of a deep learning architecture wherein at least one layer is defined by its ability to feed its own activation value into itself in a series of sequential time-steps. By doing so, the RNN is able to make predictions on each position in the analyzed sequence using information found at earlier times in the sequence. In this project, long short term memory (LSTM) cells are used to store memory about past predictions while mitigating the problem of unstable gradients associated with classical RNN networks [3]. The LSTM cell without peepholes is defined by the following equations:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{8}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{9}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{10}$$
$$\widetilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g) \tag{11}$$
$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \widetilde{\mathbf{c}}_t \tag{12}$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \tag{13}$$

where $\mathbf{i}_t$ is the input gate, $\mathbf{f}_t$ is the forget gate, $\widetilde{\mathbf{c}}_t$ is the proposed new hidden state, $\widetilde{\mathbf{c}}_t$ is the new hidden state and $\mathbf{o}_t$ is the ouput gate [7]. Traditionally, the LSTM unit analyzes the input sequence in a single direction which is relevant to prediction tasks were strong sequential causality is implied [7]. However, the folding of a single amino acid in the polypeptide chain is influenced by the physical properties of neighboring amino acids on both sides and bi-directional sequence analysis is therefore needed in secondary structure prediction. A solution to this problem is the use of a bi-directional RNN. A biRNN consists of two memory cells that analyze the data sequence from opposing directions [7] with the final hidden

[1]See e.g. `https://github.com/ducha-aiki/caffenet-benchmark/blob/master/batchnorm.md`

state being a concatenation of the two separate hidden states from each direction [8]:

$$\mathbf{h}_t = \begin{pmatrix} \overrightarrow{\mathbf{h}_t} \\ \overleftarrow{\mathbf{h}_t} \end{pmatrix} \tag{14}$$

2.4. One-dimensional convolutional neural networks

A one-dimensional CNN applies a fixed weight kernel on a sliding window section of the input sequence (the receptive field) in order to extract characteristic patterns in the input. By learning specific values of the weight kernel, the CNN is able to extract different types of features and by using a number of different filters the CNN is able to search for a number of different patterns across the input sequence. The activation of the i'th hidden node in a single one-dimensional CNN layer with input $\mathbf{x}$ can then be defined as:

$$\mathbf{z}_{l+1,d,i} = \sum_{k=0}^{K} \mathbf{w}_{k,d}\mathbf{x}_{j+k} + \mathbf{b}_d \tag{15}$$

$$\mathbf{h}_{l+1,d,i} = a\left(\mathbf{z}_{l+1,d,i}\right) \tag{16}$$

where K is the kernel size, d is the feature map (the output created by the application of a particular filter) and j is the index of the current position in the input sequence [9].

3. DATA SET

The UC Irvine Machine Learning Repository protein secondary structure data set was used to train and validate all three models. This data set has been made available by Terry Sejnowski and was originally developed in collaboration with Ning Qian [2]. The data set consist of a training and a validation set containing 111 and 17 sequences of non-homologous globular proteins respectively. The data set contained three secondary structure classes with a slightly imbalanced class distribution as illustrated in Table 1 . However, these imbalances were not found to be so severe as to require more specialized techniques (e.g. over/under-sampling). All amino acid and and secondary structure-label information was encoded via standard one-hot-encoding such that each amino acid was represented as a 20-element long binary vector and each secondary structure-label was represented as a 3-element long binary vector.

4. RESULTS

All models were implemented in TensorFlow [10] and trained on a personal laptop using a GeForce GTX 860M GPU. An overview of model architectures and results can be seen in Figure 2 and Table 2 respectively. The codebase for this project is available at: `https://github.com/lblaabjerg/02456-project`.

[2]Available at `https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Protein+Secondary+Structure)`

Data set	Sequences	Freq. C	Freq. E	Freq. H
Training	111	9,866	3,636	4,601
Validation	17	1,923	748	849

Table 1. Class distribution for training and validation set with standard annotations: C (turns/bends/coils), E (beta-strands/beta-bridges) and H (all helices).

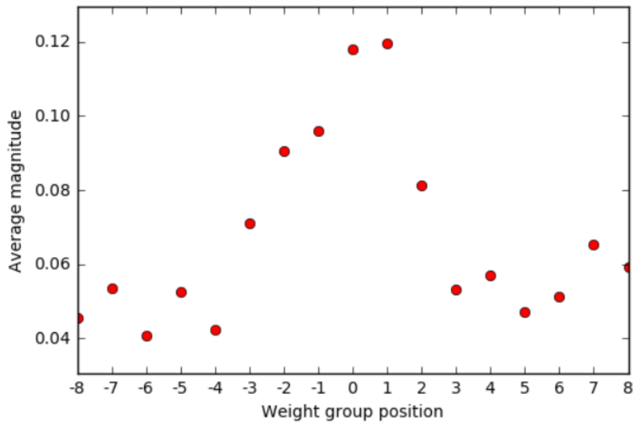

Fig. 1. Average weight magnitude for each weight group position in a 17 amino acid sliding window. It is observed, that the central five amino acids carry the majority of the secondary structure information.

Model	Q3 validation accuracy
FFNN	62.5%
biRNN	54.6%
CNN-biRNN	54.6%
Qian et al. FFNN	61.5%

Table 2. Model performance of the three proposed models on the Sejnowski/Qian data set including benchmark performance by Qian et al. [11].

4.1. First model results

A simple FFNN model was applied to the data set using a 17 amino acid sliding window of the protein primary sequence as input. The overall model consisted of a single hidden layer with 40 hidden nodes feeding into a softmax output layer. The optimal learning rate was determined to be $\alpha = 0.001$ and the optimal L^2 regularization strength was determined to be $\lambda = 0.01$. This yielded a 62.5% validation accuracy on the data set.

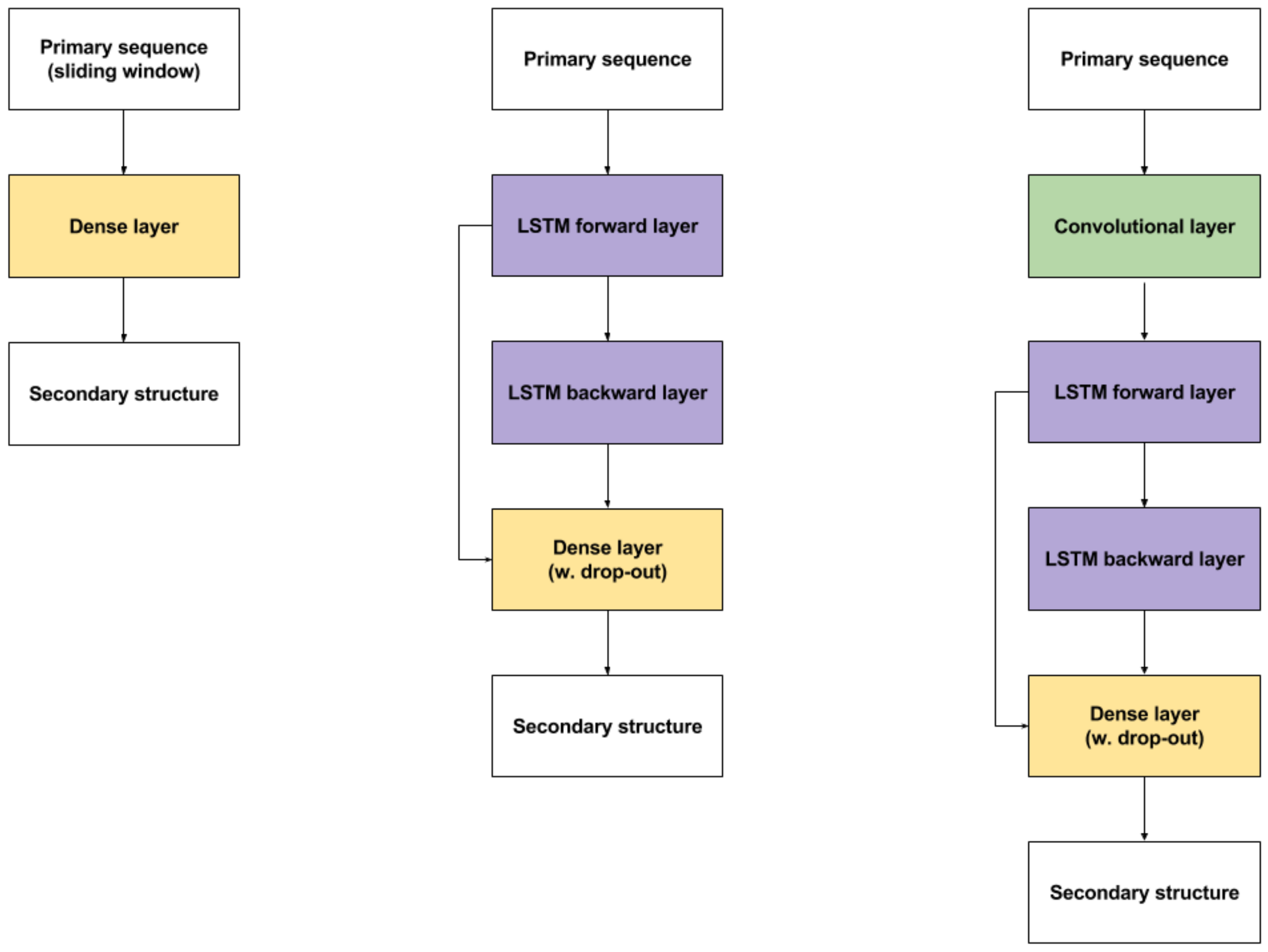

Fig. 2. Overview of neural network architectures for each of the three proposed models. Left: Sliding window FFNN model. Middle: biRNN model. Right: CNN-biRNN model. Batch normalization is included after each predictive layer in the two advanced models (i.e. middle and right model).

Interestingly, it was discovered that the relatively information content of each position in the sliding window differed significantly from each other in a unique pattern. It is known, that each amino acid in a 17 amino acid sliding window contributes with information to the secondary structure of the central amino acid [11]:

$$I(s_i = S|R_{i-8}, ..., R_i, ..., R_{i+8}) \approx \tag{17}$$

$$\sum_{j=-8}^{8} I(s_i = S|R_{i+j}) \tag{18}$$

where i is the index of the central amino acid. This situation is illustrated in Figure 1 which depicts the average weight magnitude for each weight group position in the 17 amino acid sliding window of the final FFNN model. It is observed, that the central five amino acids carry the majority of the secondary structure information and that the information content seems to decrease from the central amino acid in a Gauss-like fashion. These findings are consistent with similar results found in the literature [11].

4.2. Advanced model results

Two different advanced models were implemented. The first model consisted of a biRNN layer followed by a dense layer with drop-out feeding into the softmax output layer. The second model consisted of a one-dimensional convolutional layer followed by a biRNN layer followed by a dense layer with drop-out feeding into the softmax output layer. A hyperparameter search was performed for both models with hyperparameter values in the following ranges: CNN-filters $\in \{8, 10, 16\}$, CNN-kernel size $\in \{5, 7, 13, 17\}$, LSTM units $\in \{20, 40, 80, 100\}$, dense hidden units $\in \{20, 40, 50\}$, $p_{drop} \in \{0.2, 0.5, 0.8\}$, $\alpha_{Adam} \in \{0.0001, 0.001, 0.01, 0.1\}$, $\lambda_{L^2} \in \{0.0001, 0.001, 0.01, 0.1\}$. Surprisingly, none of the models were able to surpass a 54.6% level of validation accuracy.

This level of performance was, however, achieved by several of the investigated models corresponding to a situation wherein the model predicts all the amino acids to belong to the abundant C-class (i.e. random coil). This peculiar behavior on the validation data was observed despite the models being able to accurately distinguish between each of the three classes in the training data.

5. DISCUSSION AND CONCLUSION

From the results in Table 2 it is clear that the advanced models were not able to improve the predictive performance of the sliding window FFNN. This is a surprising result, as combinations of CNN and biRNN architectures have been successfully applied to secondary structure prediction problems by others researchers with great success [1] [12].

In order to test the developed models for bugs, the models were applied to new sets of training and validation data. Specifically the cullpdb+profile data set was used for training and the CB513+profile data set was used for validation [3] both of which have been used extensively for deep learning secondary structure prediction by other researchers [12] [13]. Both data sets contained eight different secondary structure classes and a subset of 1,000 training sequence and 1,000 validation sequences were extracted for debugging purposes. Applied to these alternative data sets, our advanced models were able to distinguish between each of the eight classes in both the training and the validation data and achieve a predictive performance comparable to the state of the art [12] with only little hyperparameter tuning.

The above findings suggests that the cause of the relatively low validation accuracy of our advanced models on the UC Irvine Machine Learning Repository data is not due to the investigated models themselves but rather due to a lack of correlation in the data between the training and the validation sets. The same data set issues were apparently faced by Qian et al. [11] who originally analyzed the UC Irvine Machine Learning Repository data using a simple sliding window FFNN model comparable to our first model. In their analysis, Qian et al. found that the peak performance on the data set was almost independent of the number of hidden units in their model. Even more surprising, the testing success rate of a network with no hidden units was about the same as one with 40 hidden units. Concluding from these results, Qian et al. hypothesized that:

”These results suggest that the common features in the training and testing proteins are all first order features and that all of the first order features learned from the training set that we used were common features. The higher order features (the information due to interactions between 2 or more residues) learned by the network were specific to each individual protein, at least for the proteins that were used.”[11]

Our findings in this project support the above hypothesis and we therefore conclude that the lack of predictive performance for our advanced models is due to a lack of common higher-order features in the training and the validation data sets. However, it should be noted that these findings are most likely unique to the investigated data set and are therefore not a general feature of amino acid interactions in nature.

[3]Both available at `http://www.princeton.edu/~jzthree/datasets/ICML2014/`

6. REFERENCES

[1] W. Sheng, J. Peng, J. Ma, and J. Xu, "Protein secondary structure prediction using deep convolutional neural fields," *Scientific reports*, vol. 6, 2016.

[2] D. P. Kingma and J. L. Ba, Eds., *Adam: A method for stochastic optimization*. International Conference on Learning Representations, 2015.

[3] M. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[4] K He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv preprint: 1502.01852*, 2015.

[5] I. Sergey and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shifts," *arXiv preprint: 1502.03167*, 2015.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[7] A. Graves, *Supervised sequence labelling with reuccurent neural networks*, Springer, 2012.

[8] M. Schuster, K. Paliwal, and A. General, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, 1997.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

[10] M. Abadi and et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, `https://www.tensorflow.org`.

[11] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models," vol. 202, pp. 865–884, 1988.

[12] A. R. Johansen, S. K. Soenderby, and O. Winther, "Protein and secondary structure prediction with convolutions and vertical-bi-directional rnns," 2016, Unpublished. Available at `https://github.com/alrojo/CB513/blob/master/Article/cb513_artikel.pdf`.

[13] S. Soenderby and O. Winther, "Protein secondary structure prediction with long short term memory networks," *arXiv preprint: 1412.7828*, 2014.