Secondary Structure Prediction with Bidirectional LSTMs and Structured Prediction

Ankit Gupta
ankitgupta@college.harvard.edu

Tom Silver
tsilver@college.harvard.edu

May 12, 2016

Code: `https://github.com/ankitvgupta/CS287assignments/tree/master/finalproject`

1 Introduction

It is widely believed that the four levels of protein structure are completely determined by primary structure, which consists of one-dimensional sequences of amino acids (Cheng et al., 2008). Nonetheless, the relationship between this primary structure and the secondary structure — the local three-dimensional configurations of the protein — remains poorly understood. After over a decade of machine learning and bioinformatics research on this problem, state of the art algorithms for predicting secondary structure from primary structure can only do so with 68.4% accuracy (Wang et al., 2016). Given the deep relationship between protein structure and function, and the myriad biomedical applications of protein profiling, secondary structure prediction (SSP) remains a central problem in applied machine learning (Paliwal et al., 2015).

Research on SSP is driven not only by the importance of the applications, but also by the clean formulation of the problem. An input sequence of discrete symbols (amino acids) is mapped to an output sequence of classes (secondary structure tags). This simple problem description invites direct application of methods for multiclass classification and sequence tagging. In particular, methods for the Natural Language Processing (NLP) tasks of Part of Speech tagging (POS) and Named Entity Recognition (NER) can be immediately applied. In this work, we adapt models for POS and NER into the context of SSP. With a focus on Bidirectional Long Short-Term Memory networks (Bi-LSTMs), we are able to achieve an accuracy of 67.7% on the same training and test data used in previous reports. We also introduce a novel structured prediction model that combines Bi-LSTMs and Maximum-Entropy Markov Models (Bi-LSTM-MEMMs). Our work suggests deep structured prediction as a promising path for future work on SSP.

2 Related Work

SSP has enjoyed consistent research attention from the machine learning community over the past two decades (Cheng et al., 2008; Paliwal et al., 2015). The first efforts to predict secondary structure largely ignored the sequential nature of the data and relied on simple window-based methods like SVMs (Ward et al., 2003) and feedforward neural networks. Neural network-based approaches evolved from the single-layer PSIPRED model of McGuffin et al. (2000) into deeper models like SPINE X (Faraggi et al., 2012). Spencer et al. (2015) recently used Deep Belief Networks for SSP, finding these models to be comparable if slightly less performant than the previously introduced neural networks. Zhou and Troyanskaya (2014) improve on these previous results

with a supervised, convolutional adaptation of Bengio's Stochastic Generative Network (Bengio et al., 2013).

Window-based models are unable to learn longterm dependencies, which has proven to be a critical deficiency in the context of SSP. One of the first attempts to capture relationships beyond a fixed window was led by Pollastri et al. (2002), who used a Bidirectional Recurrent Neural Network (Bi-RNN). Though this model is demonstrably more effective than previous approaches, the learning ability of the Bi-RNN is severely limited due to the vanishing gradient problem. LSTMs, which were introduced by Hochreiter and Schmidhuber (1997) to avoid this problem, were a natural next step for SSP. As early as 2003, Daniel considered single-layer LSTMs for SSP with some success, but his results were constrained by computational resources of the time (Daniel, 2003). More recently, Sønderby and Winther (2014) developed a bidirectional three-layer LSTM and achieved the state of the art at the time. Their efforts form the basis of the work described in this project.

Like NER, SSP deals with subsequences of targets; the secondary structures output in SSP are analogous to the named entities output in NER. In this vein, methods for structured prediction used for NER and other related problems may be appropriately applied to SSP. Common structured prediction methods like Hidden Markov Models (HMMs) (Asai et al., 1993), Maximum-Entropy Markov Models (MEMMs) (Ding et al., 2009), and Conditional Random Fields (CRFs) (Lukov et al., 2010) have been considered for SSP. Wang et al. (2011) use the more sophisticated Conditional Neural Fields (CNFs) — a combination feedforward neural network and CRF — with greater success. In 2016, the same group extended CNFs by replacing the feedforward neural network with deep convolutional neural networks, setting the current state of the art on SSP (Wang et al., 2016). Following their lead, we consider a similar deep structured prediction model in this work.

While previous work on SSP is extensive, the specific instance of the problem considered in this project follows a more narrow literature trail. Though protein secondary structures are typically classified according to eight categories, SSP has been historically reduced to a three-class classification problem, with six of the eight classes folded under one heading. This instance of SSP is commonly referred to as Q_3. As methods for Q_3 have improved, there has been increasing interest in the original problem, which is referred to as Q_8. Pollastri et al. (2002) introduced CB513, the standard dataset used for testing in Q_8. While other work varies the features included for SSP, CB513 standardizes the feature set to include 24 dense features in addition to the sparse amino acid tags. Training sets vary between works; to allow for direct comparison, we use the training and test data provided by Zhou and Troyanskaya (2014), which was also used by Sønderby and Winther (2014). We also introduce a new sequence-only training set that is significantly larger than those previously reported.

3 Problem Description

SSP requires a mapping from sequences of amino acid features $(\boldsymbol{x}_{1:n})$ to sequences of secondary structure labels $(c_{1:n})$, one per input. We first interpret the problem as one of multiclass classification. Inputs are comprised of sparse features, corresponding to the 22 amino acids found in the dataset and an additional sequence separator token, and dense features, which include chemical features of the amino acids (see (Zhou and Troyanskaya, 2014) for more details). The output space consists of nine classes — eight for the traditional secondary structure labels, and one sequence

separator class corresponding to the additional input token. Model results are reported in terms of accuracy, i.e. the percentage of correctly predicted classes.

We also consider the problem in the context of structured prediction. Here the objective is to learn a function of the input to class scores that is parameterized by the class of the previous input. Such a function is denoted $\hat{\boldsymbol{y}}(c_{i-1})_{c_i}$, where the $\boldsymbol{x}_i$ is implied. Then we find a sequence of classes that maximizes a sequential scoring function $f(\boldsymbol{x}_{1:n}; c_{1:n}) = \sum_{i=1}^{n} \log \hat{\boldsymbol{y}}(c_{i-1})_{c_i}$. The sequence that maximizes f is returned as the output at test time. For consistency, we also use accuracy to report results in this context.

4 Model and Algorithms

4.1 Long Short-Term Memory

The standard Recurrent Neural Network (RNN) model is given by the hidden state space $\mathbb{R}^{d_{hid}}$ (with default $d_{hid} = 50$), the input state space $\Sigma = \bigcup_{i=1}^{23}\{i\}$, the initial state vector (chosen uniformly at random), and the transition function $R : (\mathbb{R}^{d_{hid}} \times \Sigma) \rightarrow \mathbb{R}^{d_{hid}}$. An output function is added to map the hidden state space to the output space $\mathbb{R}^9$, which represents the log probabilities of an input belonging to each of our 9 classes. Long Short-Term Memory networks (LSTMs) are a popular extension of RNNs that avoid the vanishing gradient problem with the addition of input, forget, and output gates as part of the transition function (Hochreiter and Schmidhuber, 1997). Multiple LSTM layers may be stacked on top of one another to allow for deeper representations of the input; we consider two and three-layer models. As an additional extension, we allow for dropout between LSTM layers, which has been suggested as an effective form of regularization for deep networks (Srivastava et al., 2014). We train our models according to the Cross Entropy Loss using minibatching, which introduces small error for the sake of significant efficiency gains.

4.1.1 Bidirectional LSTMs

RNNs and LSTMs have bidirectional variants in which two submodels are learned: one for the forward sequence, and one for the reverse sequence. These two representations of the input sequence are then concatenated together, forming a vector of length twice the embedding size. Following Sønderby and Winther (2014), we introduce a feedforward neural network layer with a Rectified Linear Unit to map the output of the top Bi-LSTM layer to the output space.

4.2 Structured Prediction with MEMMs

Structured prediction requires learning a function $\hat{\boldsymbol{y}}(c_{i-1})_{c_i}$ as described above. MEMMs attempt to learn this function directly, rather than learning intermediate transition and emission probabilities in the style of HMMs. MEMMs are nearly identical to the standard log linear models except that they take $\boldsymbol{x}_i \oplus [c_{i-1}]$ as input. More formally, $\hat{\boldsymbol{y}}$ is modeled with a linear and a softmax:

$$\log \hat{\boldsymbol{y}}(c_{i-1})_{c_i} = \log softmax(feat(\boldsymbol{x}_i, c_{i-1})\boldsymbol{W} + \boldsymbol{b}) \tag{1}$$

With such a function in hand, we generate a sequence of output labels using the Viterbi dynamic programming algorithm. Viterbi is guaranteed to return the most probable output sequence according to our model.

4.2.1 LSTM-MEMM

For our last model, we introduce a deep structured prediction model that to our knowledge has not been previously considered in the literature. The state of the art SSP method uses deep CNNs to extract representations of the input and CRFs for structured prediction on these representations (Wang et al., 2016). We propose an analogous model in which Bi-LSTMs are used to extract input representations and MEMMs are used for structured prediction, forming a Bi-LSTM-MEMM. This architecture is illustrated in Figure 1. During training, each input is run through the Bi-LSTM, concatenated with the training output, and then passed to the MEMM. Error backpropogates throughout the entire model according to the Cross Entropy loss. At test time, the Bi-LSTM portion of the model is run over the entire test sequence. The output of the Bi-LSTM is used as new input features for the MEMM, which can then proceed in the standard way.

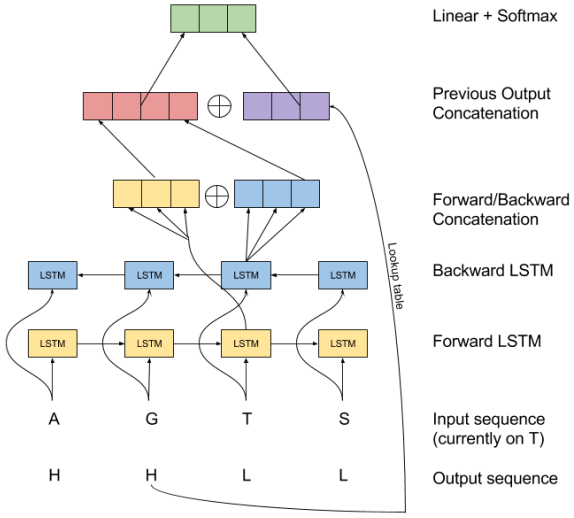

Figure 1: ***Architecture of a Bi-LSTM-MEMM.*** *The input sequence is passed through a bidirectional LSTM. For a given index in the input sequence, the LSTM representation is concatenated with a learned dense representation of the previous output class. A linear and softmax projects this vector into the output space. While the depicted Bi-LSTM-MEMM contains one bidirectional LSTM layer, our models stack two or three of these layers.*

5 Datasets

We use three main datasets in this work. The first, known as CB513, is a widely used dataset for testing SSP models (Pollastri et al., 2002; Zhou and Troyanskaya, 2014; Sønderby and Winther, 2014; Wang et al., 2016). It consists of 513 protein sequences and their corresponding secondary structures in the terms of class labels between 1 and 8. We use the parsed version of the dataset provided by Zhou and Troyanskaya (2014) as our primary test set[1].

[1] `http://princeton.edu/~jzthree/datasets/ICML2014/`

Given the relatively small size of CB513, other datasets must be used for training. Previous work varies in the selection of a training set. To allow for direct comparison with previous methods, we primarily use the training set provided by Zhou and Troyanskaya (2014), which contains roughly 6,000 sequences from the PISCES Cull PDB serverWang and Dunbrack (2003). We refer to this dataset as CullPDB. Both the CB513 and CullPDB datasets contain a number of extra features for each amino acid, which encodes information specific to that location, including information that is shared between all instances of the amino acid, and others that depended on the surrounding amino acids (such as the type of bonds formed) (Zhou and Troyanskaya, 2014).

Our third dataset, extracted from a recent addition to the RCSB Protein Data Bank (Berman et al., 2000), has to our knowledge not yet been used for SSP. This dataset contains approximately 250,000 amino acid sequences (several times bigger than any of the prior datasets) along with their associated structure sequences. This dataset, which we will call HUMAN[2] going forward, needed to be filtered in the same way as the CullPDB dataset to remove the training sequences that are too similar to test sequences. We reached out to the authors from Zhou and Troyanskaya (2014), who graciously supplied us with their sequence filtering python script. Unfortunately, because of the size of our new dataset, we estimated it would take approximately 3 months to run the script on the full dataset with a single CPU. Thus, we parallelized the algorithm by running it on smaller sets of sequences, and ran on the Odyssey Research Cluster with 2200 compute nodes, which completed the processing in a few hours. The resultant filtered dataset has roughly 110,000 protein sequences in it. It is important to note that this HUMAN dataset contains only the raw amino acid sequences; it does not contain any of the additional features present in the CullPDB and CB513 datasets.

To summarize, we consider three datasets:

- CB513 (test dataset)
- CullPDB (small training dataset)
- HUMAN (large training dataset; amino acids only)

5.1 Previous Results on CB513

We begin with a report of previous results on the CB513 dataset to provide context for our subsequent results (see Table 1).

Model	Accuracy	Citation
Bidirectional RNN	0.511	Pollastri et al. (2002)
Conditional Neural Fields	0.649	Wang et al. (2011)
Convolutional Generative Stochastic Networks	0.665	Zhou and Troyanskaya (2014)
Bidirectional LSTMs	0.674	Sønderby and Winther (2014)
Deep CNF	0.684	Wang et al. (2016)

Table 1: Previously reported results on the CB513 dataset. We are able to outperform the previously reported Bidirectional LSTM results, but we do not reach the state of the art set by the deep CNF.

[2]The dataset has a large array of human protein sequences.

6 Experiments

6.1 Experimental Setup

All of our larger experiments were conducted on the Harvard Odyssey Research Cluster. Since LSTMs are quite computationally extensive, and the stacked LSTMs are even more so, we decided to configure our code to work with the CUDA-based GPUs available in Odyssey. Thus, we used the appropriate torch GPU packages, and ran our code on largely on the holyseasgpu partition in Odyssey.

6.2 Amino Acid Sequence Only Experiments

In this first series of experiments, we wished to determine the limits of amino acid sequence-based predictive models. We therefore ignored all additional features between the amino acid sequence in both the training set, which was CullPDB in this case, and the test set, CB513. Prior work has not reported on the performance of sequence-only models for SSP. Given that the majority of available sequence data do not contain consistent additional features beyond the amino acid sequence, we submit that sequence-only predictive ability is actually a critical metric for SSP methods. We use CullPDB to allow for direct comparison with previous work, which does take advantage of additional features.

6.2.1 Stacked LSTMs with Dropout

First we consider simple stacked LSTMs with dropout. We performed hyperparameter optimization via grid search with 433 parameter configurations. The top results are reported in the Table 2. The second set of four is the best configurations that used just one layer.

SeqLength	Embedding	Optimizer	η	RNN Unit 1	RNN Unit 2	Dropout	Accuracy
100	45	sgd	0.1	lstm	lstm	0.75	0.44798
50	25	adagrad	0.001	gru	lstm	0.5	0.44769
50	25	sgd	0.1	lstm	lstm	0.5	0.44757
50	45	sgd	0.1	lstm	lstm	0.75	0.44715
100	25	adagrad	0.001	lstm	none	0.5	0.44500
100	25	adagrad	0.001	lstm	none	0.75	0.44432
50	25	adagrad	0.001	lstm	none	0.5	0.44139
50	25	adagrad	0.001	lstm	none	0.75	0.43995

Table 2: The best results of Hyperparameter Optimization for Single Directional LSTM. The second set of four is the best configurations that used just one layer.

6.2.2 Single-layer Bi-LSTM

Once we had an understanding of how LSTMs would perform, we decided to take this further and use a bidirectional LSTM. This is particularly useful for protein structure, since proteins do not necessarily follow the same left to right structure that language follows. As a result, having information from both sides of the protein sequence appeared to be useful for understanding the

structure of a particular amino acid. We implemented this in Torch using the `nn.BiSequencer` module, which allows us to specify forward, backward, and merge modules. For the forward and backward modules here, we used the `FastLSTM` implementation from the Element-RNN library, and for the merging module, we use a simple `JoinTable` function, which just concatenates the results for the forward and backward sequencers. We then pass the concatenated tensor through a simple feed-forward multilayer perceptron, with a hidden layer size that is specified as a command line argument. We trained using SGD and adagrad, but the best results all used SGD. We first tested with 193 configurations for 200 epochs, and then took the best 20 from there, and ran them for 600 epochs. The top results are reported in Table 3.

SeqLength	Embedding	Hidden	η	Dropout	Accuracy
200	200	100	0.01	0.5	0.53617
200	200	200	0.01	0.5	0.53347
100	100	200	0.01	0.5	0.52410
100	100	100	0.01	0.5	0.51974
100	200	100	0.01	0.5	0.51588

Table 3: The best results of Hyperparameter Optimization for Bi-LSTM with the CullPDB dataset. These were trained using an LSTM in each direction with SGD.

6.2.3 Stacked Bi-LSTM with HUMAN

After running the previous preliminary experiments on the smaller CullPDB dataset and verifying the correctness of our single-layer Bi-LSTM, we were ready to scale up to larger data and deeper models. In this next sequence-only experiment, we consider the performance of stacked Bi-LSTMs on CB513 after training on the large HUMAN dataset. In this case, we added a command line parameter to specify the number of bidirectional layers. Then, in our model constructor, we have a simple loop which repeatedly adds layers to the network. Naturally, we pay for the greater representation power of the additional layers with a greater time cost for backpropogation. The results of two and three layer Bi-LSTMS trained on HUMAN and evaluated on CB513 are displayed in Table 4. Note that the best results are better than those of the bidirectional LSTMS that were not stacked in Table 3.

SeqLength	Embedding	Epochs	Hidden	*eta*	Dropout	NumBidirectionalLayers	Accuracy
100	300	15	100	0.1	0.5	2	0.545534
200	200	30	200	0.1	0.5	2	0.545168
200	200	30	200	0.2	0.5	2	0.543882
50	100	15	200	0.1	0.5	2	0.542855
50	100	30	200	0.1	0.5	2	0.542675

Table 4: The best results of Hyperparameter Optimization for Stacked Bi-LSTM with the filtered HUMAN dataset. These were trained using an LSTM in each direction, and that process repeated several times. Optimization was done with SGD.

6.2.4 Stacked Bi-LSTM with windowed HUMAN

Up until this point, we were only implicitly using information about the amino acids that surround a particular amino acid when predicting its structure class. This is because the LSTMs take into account the surrounding amino acids when the bidirectional LSTM reads over the chain. However, we wanted to also explicitly add these windows. So, as input features, rather than just having the LookupTable embedding for a single amino acid, we have it for 3 or 5 amino acids, where the one whose class is being predicted is the middle one. As a whole, we found that this led to a modest uptick in the prediction accuracy, suggesting that the bidirectional LSTM alone was not able to capture all of the necessary information about the neighboring amino acids, which would have been helpful. See Table 5 for the top results.

d_{win}	SeqLength	Embedding	Hidden	η	Dropout	NumBidirectionalLayers	Accuracy
5	100	50	200	0.1	0.5	2	0.55587
3	50	50	100	0.1	0.5	2	0.55503
3	100	50	100	0.1	0.5	2	0.55495
5	100	20	200	0.1	0.5	2	0.55464
5	100	30	200	0.1	0.5	2	0.55319
3	100	100	200	0.2	0.5	2	0.55278

Table 5: The best results of Hyperparameter Optimization for Stacked Bi-LSTM with the filtered HUMAN dataset with different window sizes. These were trained using an LSTM in each direction, and that process repeated several times. Optimization was done with SGD.

6.3 Experiments with Full Features

After setting sequence-only baselines with our models, we next set out to establish direct comparisons with previous work by incorporating the full set of features present in the CullPDB and CB513 datasets. We found, in fact, that they help substantially, getting us to a point where we could, just marginally, beat the benchmark set by the team that worked with bidirectional LSTMs in the past (Sønderby and Winther, 2014). This is important, as it shows that the LookupTable-based embedding system is not able to capture enough of the information about the amino acids to perform highly here. The full set of experiments is described below.

6.3.1 Stacked Bi-LSTMs

In this set of experiments, we consider the same stacked Bi-LSTM with dropout previously described (with $d_{win} = 1$), but this time with additional features. See Table 6 for the results. We are able to achieve accuracy of 0.677, which outperforms the previously reported accuracy of 0.674 using Bi-LSTMS on the same training and test set (Sønderby and Winther, 2014).

6.4 Simple MEMM

Before considering the more sophisticated Bi-LSTM-MEMM previously described, we wished to establish the ability of a simple MEMM without a transformed input representation. Due to computational constraints, we were only able to train and evaluate the MEMM with window sizes of 1

SeqLength	Embedding	Hidden	η	Dropout	NumBidirectionalLayers	Accuracy
100	50	100	0.05	0.15	3	0.677230
100	50	100	0.05	0.25	3	0.676115
100	50	100	0.05	0.25	2	0.674906
100	50	100	0.05	0.25	3	0.674671
100	50	100	0.05	0.1	3	0.674448
100	50	100	0.05	0.05	3	0.674319

Table 6: The best results of Hyperparameter Optimization for Stacked Bi-LSTM with the CullPDB dataset with full features. These were trained using an LSTM in each direction, and that process repeated several times. Optimization was done with SGD.

and 3 on the dataset with full features. As expected, the MEMM performs relatively poorly, almost always predicting the most probable output class given the input. These models were trained for 50 epochs in minibatches of size 320. Top results are reported in Table 7.

d_{win}	Class Embedding	Hidden	η	Accuracy
3	10	50	0.1	0.3344708
3	10	100	0.1	0.3342363
1	10	100	0.1	0.3235652
1	10	100	0.01	0.3233777

Table 7: The best results of simple MEMM trained on CullPDB with full features and evaluated on CB513.

6.5 Stacked Bi-LSTM-MEMM

In this experiment, we decided to add the MEMM technique that we worked in before to the LSTM system. See the Models section for information how this works. Essentially, the idea here is that we would incorporate the additional informational about the predicted previous classes as we make predictions, and we use the Viterbi algorithm at test time to find the best sequence. Unfortunately, we could not optimize this quite enough to match the LSTM-only performance, but this is definitely an area where further work can improve the success. The two best results here are in Table 8.

SeqLength	Embedding	Hidden	η	Dropout	NumBidirectionalLayers	Accuracy
100	50	100	0.05	0.25	2	0.6514604
50	100	100	0.1	0.5	2	0.6459706

Table 8: The best results of LSTM-MEMM. These were trained using an LSTM in each direction, and that process repeated several times. Optimization was done with SGD.

7 Future Work

There are several directions for future work in this area, and some of these are much more involved than others. First, we can improve the datasets that we used. We found in our tests that when we used the HUMAN dataset rather than the CullPDB dataset without the additional features, meaning when we were using the raw sequence only, we were able to get a noticeable uptick in the accuracy of the algorithms. However, the CullPDB dataset with additional features still performed better, and this is likely because those extra features provide meaningful information. Thus, by doing the same sort of preprocessing done on the smaller dataset to this new, larger one, we may be able to get a performance boost. Furthermore, one of the reasons we believe the sequence alone did not perform as well is because we were not able to get, from learning embeddings alone, all of the interesting features that the processed features contained. Thus, being able to use pretrained embeddings may be able to improve the starting point of this algorithm. Fortunately, there has been work in this area, and there are published embeddings for amino acids available from Asgari and Mofrad (2015). We think that these have the potential to improve the amino acid models, as they did for text, and we definitely recommend incorporating those in the future.

Furthermore, we began work with structured prediction in this project by using the Maximum Entropy Markov Model setup, and using the Viterbi algorithm. However, there are several other successful techniques for structured prediction as well. For example, others have used a Conditional Random Fields, including the state of the art in this space, which uses a convolutional neural network along with a conditional random field. LSTM-CRF networks are known to perform well at certain text-based tasks (Huang et al., 2015), and so it would be interesting to apply them to this space as well, and see their performance.

8 Conclusion

As a whole, we found that deep stacked Bidirectional LSTM networks are an effective way to learn protein secondary structure networks, achieving an accuracy of 67.7% on a widely-used benchmark. Moreover, our early tests with structured prediction were not able to match the results with the bidirectional LSTMs alone, but they did suggest that structured prediction can perform quite well, and open the door for more advanced methods for structured prediction in the future.

In this project, we have demonstrated how methods that were designed for text and language can be taken out of that space and be applied to proteins as well. While at first this may seem surprising, in fact this should make sense, as cells learn to interpret structure in proteins just as humans learn to interpret structure in language. That being said, our intuitions about text do not always directly translate. For one, proteins do not have the same left-to-right structure that English text does, and so bidirectional networks appear to perform disproportionately well in this area. Furthermore, the space of possible inputs is much smaller, as there are only around 20 amino acids, while there are thousands of words in the English language. That being said, enough of an analogy remains between the two domains that we can transfer known algorithms to this space, which makes for interesting and powerful techniques like the ones we explored.

References

Asai, K., Hayamizu, S., and Handa, K. (1993). Prediction of protein secondary structure by the hidden markov model. *Computer applications in the biosciences: CABIOS*, 9(2):141–146.

Asgari, E. and Mofrad, M. R. (2015). Protvec: A continuous distributed representation of biological sequences. *arXiv preprint arXiv:1503.05140*.

Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2013). Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*.

Berman, H., Westbrook, J., Z., F., G., G., T.N., B., H, W., IN, S., and P.E., B. (2000). The protein data bank. *Nucleic Acids Research*, 28:235–242.

Cheng, J., Tegge, A. N., and Baldi, P. (2008). Machine learning methods for protein structure prediction. *Biomedical Engineering, IEEE Reviews in*, 1:41–49.

Daniel, A. (2003). Prediction of protein secondary structure using long short-term memory. Technical report, Citeseer.

Ding, Y.-S., Zhang, T.-L., Gu, Q., Zhao, P.-Y., and Chou, K.-C. (2009). Using maximum entropy model to predict protein secondary structure with single sequence. *Protein and peptide letters*, 16(5):552–560.

Faraggi, E., Zhang, T., Yang, Y., Kurgan, L., and Zhou, Y. (2012). Spine x: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *Journal of computational chemistry*, 33(3):259–267.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Lukov, L., Chawla, S., Liu, W., Church, B., and Pandey, G. (2010). *Protein Secondary Structure Prediction with Conditional Random Fields*. School of Information Technologies, University of Sydney.

McGuffin, L. J., Bryson, K., and Jones, D. T. (2000). The psipred protein structure prediction server. *Bioinformatics*, 16(4):404–405.

Paliwal, K., Lyons, J., and Heffernan, R. (2015). A short review of deep learning neural networks in protein structure prediction problems. *Advanced Techniques in Biology & Medicine*, 2015.

Pollastri, G., Przybylski, D., Rost, B., and Baldi, P. (2002). Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins: Structure, Function, and Bioinformatics*, 47(2):228–235.

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

Spencer, M., Eickholt, J., and Cheng, J. (2015). A deep learning network approach to ab initio protein secondary structure prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 12(1):103–112.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Wang, G. and Dunbrack, R. L. (2003). Pisces: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591.

Wang, S., Peng, J., Ma, J., and Xu, J. (2016). Protein secondary structure prediction using deep convolutional neural fields. *Scientific Reports*, 6:18962.

Wang, Z., Zhao, F., Peng, J., and Xu, J. (2011). Protein 8-class secondary structure prediction using conditional neural fields. *Proteomics*, 11(19):3786–3792.

Ward, J. J., McGuffin, L. J., Buxton, B. F., and Jones, D. T. (2003). Secondary structure prediction with support vector machines. *Bioinformatics*, 19(13):1650–1655.

Zhou, J. and Troyanskaya, O. G. (2014). Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. *arXiv preprint arXiv:1403.1347*.