# DeepDom: Predicting protein domain boundary from sequence alone using stacked bidirectional LSTM

Yuexu Jiang, Duolin Wang, Dong Xu
*Department of Electrical Engineering and Computer Science, Bond Life Sciences Center, University of Missouri, Columbia, Missouri 65211, USA*
*Email: xudong@missouri.edu*

Protein domain boundary prediction is usually an early step to understand protein function and structure. Most of the current computational domain boundary prediction methods suffer from low accuracy and limitation in handling multi-domain types, or even cannot be applied on certain targets such as proteins with discontinuous domain. We developed an *ab-initio* protein domain predictor using a stacked bidirectional LSTM model in deep learning. Our model is trained by a large amount of protein sequences without using feature engineering such as sequence profiles. Hence, the predictions using our method is much faster than others, and the trained model can be applied to any type of target proteins without constraint. We evaluated DeepDom by a 10-fold cross validation and also by applying it on targets in different categories from CASP 8 and CASP 9. The comparison with other methods has shown that DeepDom outperforms most of the current *ab-initio* methods and even achieves better results than the top-level template-based method in certain cases. The code of DeepDom and the test data we used in CASP 8, 9 can be accessed through GitHub at https://github.com/yuexujiang/DeepDom.

*Keywords:* protein domain; domain boundary prediction; deep learning; LSTM.

## 1. Introduction

Protein domains are conserved parts on protein sequences and structures that can evolve, function, and exist independently of the rest of the protein chain. While some proteins have only one domain, many proteins contain more than one domain. Molecular evolution uses domains as building blocks and these may be recombined in different arrangements to create proteins with different functions[1]. Thus, accurate identification of protein domains is crucial to understanding protein function and evolutionary mechanisms. Currently, the most reliable characterization of protein domain is through experimental methods. However, due to the large amount of data being generated by high-throughput technologies nowadays, it is impossible to manually identify domains for these proteins, not to mention that the experimental methods are time consuming and costly. Thus, computational domain prediction methods are in highly demand.

A variety of computational methods for protein domain prediction have been developed, and they can be roughly categorized as either template-based methods or *ab-initio* methods. The principle of most template-based methods is to find homologous sequences that have known domain information by sequence alignments and then map the domain information from these sequences to the query protein sequence. The methods belonging to this category are Pfam[2], CHOP[3], FIEFDOM[4], and ThreaDom[5]. A variation of template-based methods is to use 3D structural models to assist protein domain prediction, e.g. SnapDRAGON[6] and RosettaDom[7]. These methods first construct a tertiary structure model of the target using structural templates.

Domains are then assigned by domain parser tools from the constructed 3D model. The template-based methods can have a high prediction accuracy when close templates are found; however, their prediction performance may drop dramatically if there is no highly similar sequence in domain databases.

*Ab-initio* methods are more widely used than template-based methods, since these template-free methods can be applied to any protein. They are mainly statistical and machine learning algorithms that train models using the known protein domain boundary information stored in databases such as CATH[8] and SCOP[9]. Some of the representative methods in this category are PPRODO[10], DOMPro[11], PRODOM[12], DomCut[13], ADDA[14], DomNet[15], DROP[16], DOBO[17], and EVEREST[18]. Compared with the template-based approaches, the prediction accuracy of the *ab-initio* methods is low. This is mainly because these methods suffer from the weak domain boundary information in sequence, even after a deliberate but tedious process of feature extraction.

Deep learning is currently the most attractive area in machine learning. Among the various architectures of deep learning, Long Short Term Memory (LSTM)[19] has been successfully applied to problems such as speech recognition, language modeling, translation, image captioning[20-22]. Essential to these successes is its chain-like structure that can capture the sequential information, and its repeating module designed to avoid the vanishing gradient problem that the original Recurrent Neural Network (RNN) suffers[23]. Here, we consider protein sequences as strings of information just like language. Thus, in this paper we propose a new *ab-initio* protein domain boundary prediction method using LSTM. We assume that the signal pattern from a domain boundary region is different from the signals generated from other regions. So, we made each LSTM layer in our deep learning architecture bidirectional to capture the sequential information not just from the N-terminal side of the domain boundary region but also from the C-terminal side. Then we stack multiple such layers together to fit a high-order non-linear function in order to predict the complex domain boundary signal pattern. Instead of paying much effort in feature engineering on a small dataset, which is what traditional machine learning methods do, we train our LSTM model on a big dataset to learn data representations automatically. To the best of our knowledge, this is the first deep learning method applied on the protein domain boundary prediction problem.

## 2. METHODS

### *2.1 Data Set Preparation*

We collected 456,128 proteins with domain boundary annotations in the CATH database (version 4.2). All the sequences of corresponding proteins were downloaded from the Uniprot database[24]. Then we used CD-HIT[25] to cluster similar proteins into clusters that meet our pre-defined similarity threshold (40%). The representative sequence in each cluster was extracted to form a non-redundant dataset in which every pair of proteins has sequence identity less than 40%[26]. This threshold instead of a lower number makes sure enough data were remained for deep learning. We further excluded proteins with sequence length less than 40 residues, since it needs at least 40 residues for a domain boundary signal to be significant according to Ref. [17]. The final dataset

contains 57,887 proteins. We used 10-fold cross validation to evaluate our model. In each fold, 90% proteins were used to train a model, the remaining 10% proteins were used for testing.

### *2.2 Input Encoding*

Before using our data to train the model, we need to understand the distribution of the data. Figure 1 shows some statistics of our data, which let us believe that encoding the entire sequence for each protein was probably not a good idea. The first reason is that it introduces bias. When there is only one domain on a protein, the boundaries of the only domain are always near the protein's two termini. As shown in Figure 1(A), proteins with one domain represent the majority of the data, and this would make our model over-memorize this pattern and favor the prediction as one domain, which results in poor performance for multi-domain cases. The second reason is as illustrated in Figure 1(B), that proteins with different number of domains have different length distributions. When encoding the entire protein sequence using a dynamic length, we cannot train the model in batch, which is much faster to handle big data set. So, we decided to use a sliding window strategy independent of the protein length to encode an input sequence into equal-length fragments. And we use symbol "-" for padding when the last fragment is shorter than window size. After experiments, we determined the best combination of window size and stride is 200 residues and 80 residues.

Next, we need to encode each residue in every fragment. According to the work of Venkatarajan and Braun[27], a comprehensive list of 237 physical-chemical properties for each amino acid was compiled from the public databases. Their study showed that the number of properties could be reduced while retaining approximately the same distribution of amino acids in the feature space. Particularly, the correlation coefficient between the original and regenerated distances is more than 99% when using the first five eigenvectors. Thus, we used five numerical descriptors to represent each amino acid for computational efficiency while maintaining almost all the information at the same time. We also added the sixth encoding dimension as the padding indicator. For all the 20 types of amino acids, their sixth code is zero. The symbol "-", as the sixth

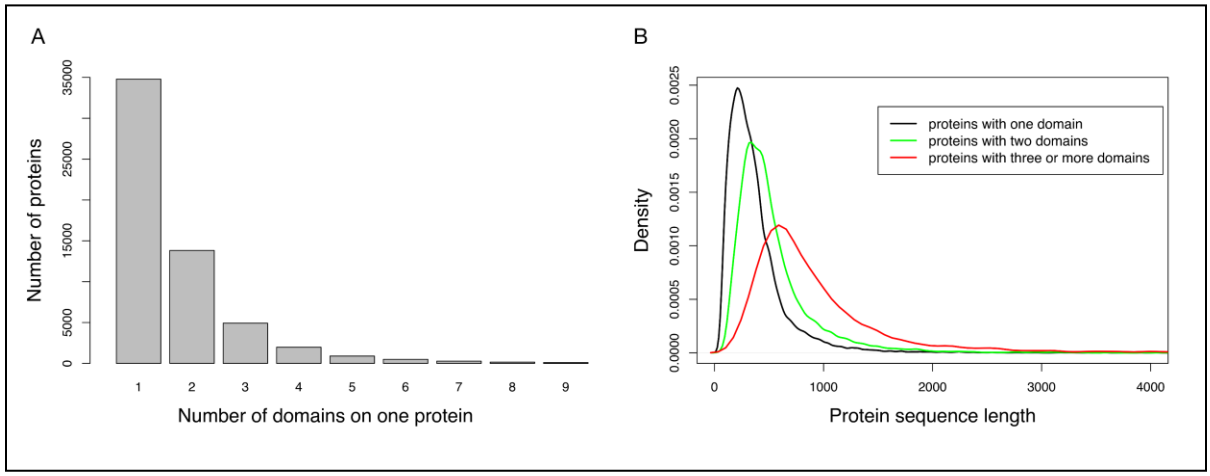


Figure 1. (A) The distribution of proteins with different numbers of domains. (B) The distribution of protein sequence lengths in different categories.

code with value 1, indicates a padding residue, and its first five codes are all zeros. Thus, for each input fragment, its coding dimension is 200 by 6.

For model training, we also need to encode the label for each residue. We derive the protein domain boundary annotation from the CATH database, and follow the convention that considers a residue as positive if it is within $\pm 20$ residues of the true boundary. Thus, the coding dimension for output labels is 200 by 3. The three values represent the probability of a residue being a positive (within the true boundary), negative (outside the true boundary), and padding residue, respectively.

### *2.3 Model Architecture*

Our deep learning architecture is shown in Figure 2. The bidirectional design in each middle layer captures the information from residues before and after a protein domain boundary. We stacked four such layers to capture the high order non-linear features that can detect complex boundary patterns or weak signals. Each neuron in the hidden layers is an LSTM unit.

The key to LSTM is the cell state C that runs through the entire chain. An LSTM unit has the ability to remove or add information to the cell state by a regulation structure called gate. Firstly, an LSTM unit uses its "forget gate" to decide what information to discard from the cell state. It takes the output $h_{t-1}$ from the previous unit and the current input $x_t$ as the input of a sigmoid function to produce a number between 0 and 1 for each number in the cell state. A 1 means completely keeping the value while a 0 means completely removing it. The formulas for the forget gate is shown as Eq. (1).

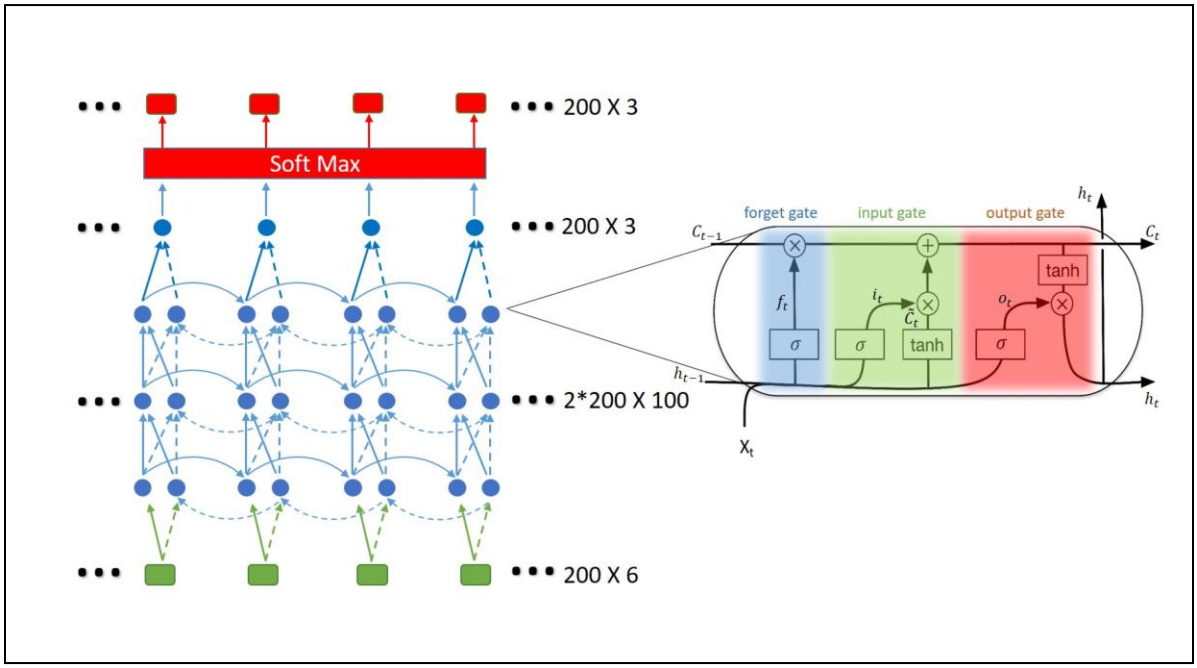


Figure 2. The stacked bidirectional LSTM model. Green boxes represents the input layer. Red boxes represents the output layer. Each box represents a residue. Blue dots form the bi-directional hidden layers. Signals from left to right are represented by solid arcs, while dashed arcs represent signals from the reverse direction. Each dot represents an LSTM unit. A magnified LSTM unit is shown. Its different gates are highlighted with different colors. At the end of the model, a Softmax layer is added to scale the output value with a sum of 1 so that they can be interpreted as probabilities.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

where $W_f$ and $b_f$ are the weight matrix and bias for the forget gate layer. Next, a tanh layer creates a new candidate input vector. It will be performed a pointwise product with a sigmoid layer called the "input gate" to decide which values to add to the cell state. The formula for candidate input creation and the input gate are shown as Eq. (2) and Eq. (3), respectively.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (2)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3)$$

where $W_C$ and $W_i$ are weight matrix for the tanh layer and the input gate layer, respectively. $b_C$ and $b_i$ are bias for the tanh layer and the input gate layer, respectively. Then the LSTM unit can update the old cell state $C_{t-1}$ into the new cell state $C_t$ by Eq. (4).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

Finally, the cell state goes through a tanh layer to scale the values between -1 and 1. The scaled cell state will be filtered by a sigmoid layer called "output gate" to decide which values to output. The formulas for output gate definition and the current output are shown as Eq. (5) and Eq. (6), respectively.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

The ability of avoiding vanishing gradient is mainly owing to the design of forget gate in LSTM. Thus, if a protein domain boundary prediction depends on some signals from remote residues, our model can be trained to set those forget gates' values as 1 on informative positions and let the far, weak but informative signal propagate far without significant loss.

### *2.4 Evaluation criteria*

We used prediction precision, recall and Matthew's correlation coefficient (MCC) to evaluate our method and compare with others'. The definitions of precision, recall, MCC are listed in Eq. (7), Eq. (8) and Eq. (9), respectively:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (7)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (8)$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(FP+TN)(TN+FN)}} \quad (9)$$

where TP, FP, TN, FN are true positive, false positive, true negative and false negative prediction, respectively. When a residue has a predicted probability of being within a domain boundary region higher than a cutoff, we checked its surrounding $\pm 20$ residues to see if there is a recorded domain boundary in the CATH database for the protein. If yes, then we have a true positive, otherwise it is a false positive. On the contrary, when there is a residue our model predicted it being outside of domain boundary regions, we checked its surrounding $\pm 20$ residues to see if there is a recorded

Table 1. Prediction performance in different experiment designs

| Window size | 80 | | | 100 | | | 200 | | |
|---|---|---|---|---|---|---|---|---|---|
| Stride | 20 | 40 | 80 | 20 | 40 | 80 | 20 | 40 | 80 |
| Experiment ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precision_d1 | 0.572 | 0.625 | **0.626** | 0.609 | 0.622 | 0.588 | 0.465 | 0.547 | 0.618 |
| Recall_d1 | 0.493 | 0.498 | 0.447 | 0.486 | 0.513 | 0.529 | **0.602** | 0.582 | 0.584 |
| MCC_d1 | 0.442 | 0.478 | 0.450 | 0.462 | 0.485 | 0.472 | 0.415 | 0.471 | **0.520** |
| Precision_d2 | 0.608 | **0.655** | 0.650 | 0.652 | 0.653 | 0.623 | 0.496 | 0.576 | 0.654 |
| Recall_d2 | 0.361 | 0.338 | 0.291 | 0.346 | 0.366 | 0.365 | **0.473** | 0.443 | 0.426 |
| MCC_d2 | 0.361 | 0.374 | 0.341 | 0.377 | 0.391 | 0.372 | 0.341 | 0.386 | **0.426** |
| Precision_d3+ | 0.639 | 0.670 | 0.661 | 0.675 | 0.668 | 0.629 | 0.543 | 0.598 | **0.669** |
| Recall_d3+ | 0.357 | 0.297 | 0.245 | 0.315 | 0.330 | 0.310 | **0.453** | 0.418 | 0.381 |
| MCC_d3+ | 0.360 | 0.340 | 0.301 | 0.354 | 0.360 | 0.326 | 0.343 | 0.367 | **0.391** |
| Precision_ALL | 0.601 | **0.644** | 0.641 | 0.637 | 0.643 | 0.607 | 0.496 | 0.570 | 0.641 |
| Recall_ALL | 0.409 | 0.382 | 0.332 | 0.386 | 0.407 | 0.406 | **0.513** | 0.486 | 0.468 |
| MCC_ALL | 0.392 | 0.402 | 0.369 | 0.401 | 0.416 | 0.394 | 0.370 | 0.412 | **0.450** |

domain boundary in the CATH database for the protein. If yes, then we have a false negative; otherwise it is a true negative.

## 3. RESULTS AND DISCUSSION

### *3.1 Parameter configuration experiments on test data*

We have done a series of experiments with different window sizes and stride values to determine the best combination of these two parameters. The prediction performance of each experiment design is listed in Table 1. And we presented the results separately based on the number of domains that a protein has. Each value is the result after the 10-fold cross validation. Note that in

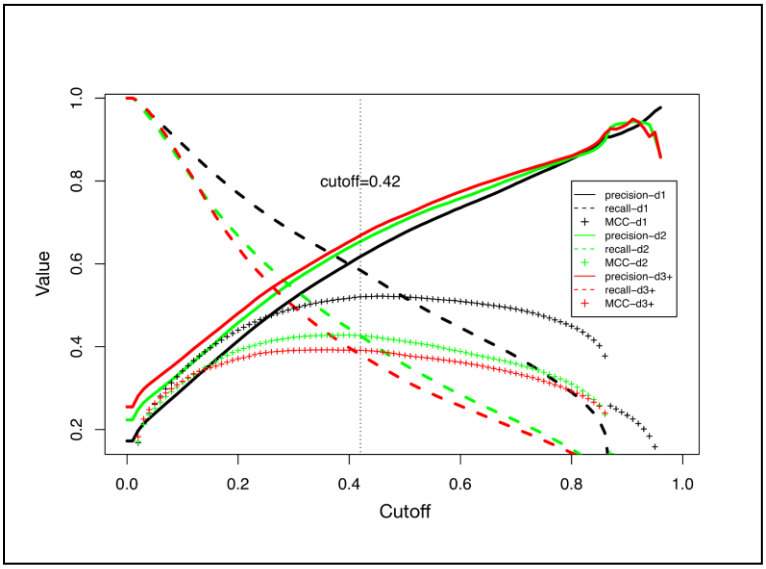


Figure 3. Illustration of the prediction precision, recall and MCC as a function of the decision threshold when the window size=200 and stride=80. The results are based on a 10-fold cross validation.

Experiment 3, we considered the situation that there is no overlap between windows. Under each experiment design (one column) in Table 1, we only presented the result that had the highest MCC-ALL at a certain threshold. We also conducted experiments using sliding window of 300 residues. However, the improvement for MCC-ALL is not significant (around 0.01) compared with cases when window size is 200 residues. So, we believe 200 is enough. As shown in Table 1, the highest MCC-ALL, also the overall best prediction performance is achieved when the sliding window size equals to 200 residues and the stride value equals to 80 residues. Figure 3 illustrates a plot of the precision, recall and MCC as functions of the decision threshold when using the optimum window size and stride value. The threshold at which the highest MCC-ALL reached is 0.42, and hence we used this value as the default threshold.

### *3.2 Comparison with Other Domain Boundary Predictors*

To perform a fair comparison with other methods on a benchmark dataset, we tested our method on the proteins in the Critical Assessment of Techniques for Protein Structure Prediction (CASP). The definitions of domain boundaries on target proteins are provided by the CASP protein domain prediction contest sessions. Based on the categories those target proteins belong to, we conducted several experiments accordingly. In each experiment, the proteins that have a 40% or higher identity with any target protein were excluded from our training dataset.

#### *3.2.1 Free modeling targets from CASP 9*

Free modeling (FM) targets are proteins without any homologous templates. These targets are often regarded as “hard cases”, since their predictions usually had poor performance. We selected all the 22 FM targets in CASP 9 and applied different methods to predict their domain boundaries. By comparing the results in the two categories in Table 2, we found most template-based methods suffered a significant decrease in both precision and recall for FM targets. ThreaDom is currently the top 1 templated-based method using multiple threading alignments to extract protein domain boundary information. For FM targets, ThreaDom identifies multiple alignments or super-secondary structure segments from weakly homologous templates, then a domain conservation score profile extracts consensus information between the domain structure and alignment gaps. This way, ThreaDom maintained a good precision for FM targets. Our *ab-initio* method DeepDom achieved the overall best prediction results for FM targets, with the same precision as ThreaDom but higher recall. All the results by different methods are listed in Table 2, where some of them were generated from the tools provided and others were collected from Ref. [5] and Ref. [17], since they used the same data.

#### *3.2.2 Multi-domain targets from CASP 9*

We also selected all the 14 multi-domain targets from CASP 9 with the constraint that every domain on one protein must be continuous, since most other methods can only handle multi-domain targets of this kind. For this category, template-based methods generally have better results. ThreaDom achieved the overall best prediction performance. But DeepDom is still the best among ab-initio methods and also competitive with the template-based methods, as shown in Table 2.

Table 2. Comparison results from different methods on two category targets in CASP 9 contest

| Category | Predictor | CASP9 protein boundary prediction | |
|---|---|---|---|
| | | Precision | Recall |
| FM | DeepDom | **0.882** | 0.468 |
| | ThreaDom | **0.882** | 0.455 |
| | Pfam | 0.323 | **0.485** |
| | FIEFDom | 0.231 | 0.182 |
| | DomPro | 0.500 | 0.182 |
| | PPRODO | 0.333 | 0.485 |
| | DROP | 0.429 | 0.182 |
| Multi-Domain | DeepDom | 0.689 | 0.441 |
| | ThreaDom | **0.764** | 0.534 |
| | Pfam | 0.500 | **0.548** |
| | FIEFDom | 0.340 | 0.233 |
| | DomPro | 0.500 | 0.140 |
| | PPRODO | 0.500 | 0.520 |
| | DROP | 0.679 | 0.260 |

### *3.2.3 Discontinuous domain target from CASP 8*

Some protein domains consist of several separated segments. The prediction of such discontinuous domain is still an unsolved problem. Most mentioned methods above have been explicitly designed to handle domains without discontinuous segments, despite the fact that discontinuous domain is important in protein structural determination and function annotations.

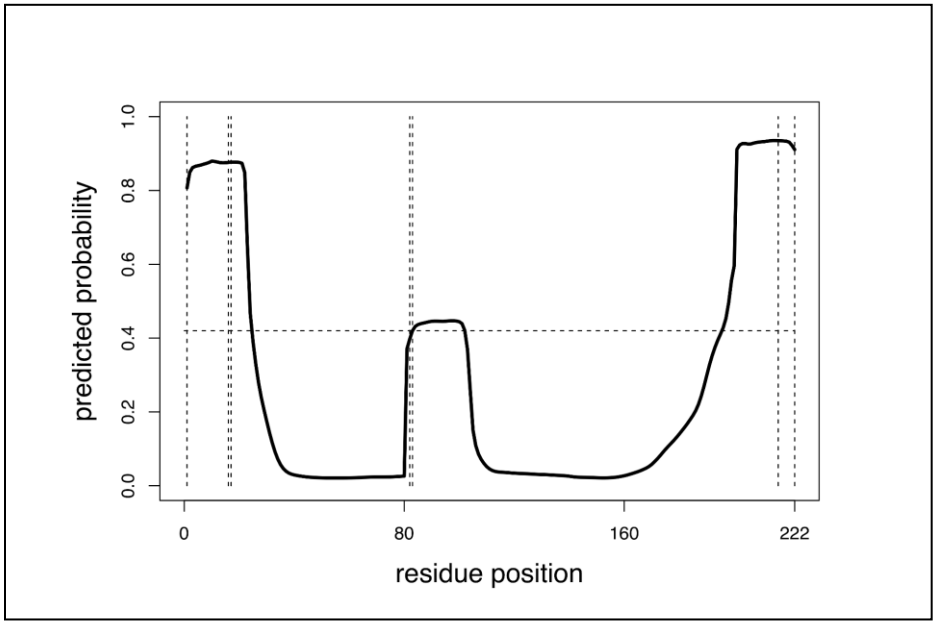


Figure 4. An illustration of discontinuous domain boundary prediction using target T0418 from CASP 8. The domain assignment is (1-16|83-216) (17-82), where the first domain has two segments. The defined domain boundaries are presented by vertical dash lines. The threshold of our model is 0.42.

To evaluate the ability of DeepDom in predicting discontinuous domain, we selected all the 18 targets that contain at least one discontinuous domain from CASP 8. The overall discontinuous domain boundary prediction precision is 81.2%, the recall is 34.8%, and with MCC of 0.38. However, currently we have not found a method to predict whether multiple segments belong to the same domain. Figure 4 gives an illustration of one discontinuous domain protein prediction.

## 4. CONCLUSION

In this paper, we designed a novel computational method called "DeepDom" for protein domain boundary prediction using deep learning. Our model does not need elaborated feature engineering. Instead, it extracts information from a large amount of raw sequence data. The comparison showed that DeepDom achieved better results than other *ab-initio* methods and is competitive with template-based methods. As an *ab-initio* method, DeepDom has the advantage to outperform the most successful template-based method when dealing with free modeling targets. Importantly, it can run much faster than other methods, all of which use sequence profiles that are time consuming to generate.

There is room for improvement of DeepDom. Ideally, a protein sequence should be encoded "globally", since breaking into fragments excludes the potential long distance dependency. By doing several experiments with varying window sizes and strides, an interesting discovery is that protein domain boundary prediction seems to depend on the signals from remote residues. However, this still requires further experiments to prove and develop a new method to use the information. The other limitation is that the prediction performance for template-available targets is lower than the best template-based method. We will develop a hybrid method that can take advantages of existing methods from both approaches (ab-initio and template-based). We also plan to make the hybrid method available as a web server. Most of the existing domain prediction web servers only allow users to submit one protein sequence a time. Since DeepDom avoids the time-consuming sequence profile generation process, the users can predict for a list of proteins in a short time.

## 5. ACKNOWLEDGEMENTS

The authors would like thank Dr. Jianlin Cheng and his student Jie Hou for their help to this work. This work was partially supported by National Institutes of Health grant R35-GM126985.

## REFERENCES

1. Ponting CP, Russell RR, Annual review of biophysics and biomolecular structure. 31, 45-71 (2002).
2. Finn RD, Coggill P, Eberhardt RY, Eddy SR, Mistry J, Mitchell AL, Potter SC, Punta M, Qureshi M, Sangrador-Vegas A et al, Nucleic acids research. 44(D1), D279-285 (2016).
3. Liu J, Rost B, Proteins. 55(3), 678-688 (2004).
4. Bondugula R, Lee MS, Wallqvist A, Nucleic acids research. 37(2), 452-462 (2009).
5. Xue Z, Xu D, Wang Y, Zhang Y, Bioinformatics. 29(13), i247-256 (2013).
6. George RA, Heringa J, Journal of molecular biology. 316(3), 839-851 (2002).
7. Kim DE, Chivian D, Malmstrom L, Baker D, Proteins. 61 Suppl 7, 193-200 (2005).

8. Dawson NL, Lewis TE, Das S, Lees JG, Lee D, Ashford P, Orengo CA, Sillitoe I, Nucleic acids research. 45(D1), D289-D295 (2017).
9. Andreeva A, Howorth D, Chothia C, Kulesha E, Murzin AG, Nucleic acids research. 42(Database issue), D310-314 (2014).
10. Sim J, Kim SY, Lee J, Proteins. 59(3), 627-632 (2005).
11. Cheng J, Sweredoski MJ, Baldi P, Data Mining and Knowledge Discovery. 13(1), 1-10 (2006).
12. Servant F, Bru C, Carrere S, Courcelle E, Gouzy J, Peyruc D, Kahn D, Briefings in bioinformatics. 3(3), 246-251 (2002).
13. Suyama M, Ohara O, Bioinformatics. 19(5), 673-674 (2003).
14. Heger A, Wilton CA, Sivakumar A, Holm L, Nucleic acids research. 33(Database issue), D188-191 (2005).
15. Yoo PD, Sikder AR, Taheri J, Zhou BB, Zomaya AY, IEEE transactions on nanobioscience. 7(2), 172-181 (2008).
16. Ebina T, Toh H, Kuroda Y, Bioinformatics. 27(4), 487-494 (2011).
17. Eickholt J, Deng X, Cheng J, BMC bioinformatics. 12, 43 (2011).
18. Portugaly E, Harel A, Linial N, Linial M, BMC bioinformatics. 7, 277 (2006).
19. Hochreiter S, Schmidhuber J, Neural computation. 9(8), 1735-1780 (1997).
20. Graves A, Mohamed A-r, Hinton G, In: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on: 2013. IEEE, 6645-6649 (Year).
21. Soutner D, Müller L, In: International Conference on Text, Speech and Dialogue: 2013. Springer, 105-112 (Year).
22. Vinyals O, Toshev A, Bengio S, Erhan D, In: Proceedings of the IEEE conference on computer vision and pattern recognition: 2015. 3156-3164 (Year).
23. Bengio Y, Simard P, Frasconi P, IEEE transactions on neural networks. 5(2), 157-166 (1994).
24. UniProt Consortium T, Nucleic acids research. 46(5), 2699 (2018).
25. Fu L, Niu B, Zhu Z, Wu S, Li W, Bioinformatics. 28(23), 3150-3152 (2012).
26. Wang D, Zeng S, Xu C, Qiu W, Liang Y, Joshi T, Xu D, Bioinformatics. 33(24), 3909-3916 (2017).
27. Venkatarajan MS, Braun W, Molecular modeling annual. 7(12), 445-453 (2001).