DeepDBP:A Novel Prediction Method for Identification of DNA-binding Protein Using Deep Neural Networks

Nazia Afrin Neezi (011 143 007)
Shadman Shadab (011 143 043
Md. Tawab Alam Khan (011 143 127)
Department of Computer Science and Engineering
United International University

A thesis submitted for the degree of
BSc in Computer Science & Engineering

April 2019

Abstract

DNA-Binding proteins are associated with many cellular level functions which includes but not limited to body's defense mechanism and oxygen transportation. They bind DNAs and interact with them. In the past DBPs were identified using experimental lab based methods. But nowadays researchers use supervised learning to identify DBPs solely from protein sequences.

At the time of writing this paper, to our knowledge, no approach has been taken to apply deep learning in order to identify DBP. So we decided to try out to different approach using deep learning to identify DNA-Binding proteins; DeepDBP(ANN) and DeepDBP(CNN).

Both of our methods were able to produce state-of-the-art results.DeepDBP(ANN) had a train accuracy of 99.02% and test accuracy of 82.80%.And DeepDBP(CNN) though had train accuracy of 94.32%, it excelled at identifying test instances with 84.31% accuracy. All the source codes and other resources including the datasets of DeepDBPcan be found at `https://github.com/antorkhan/deepdbp`.

This work is dedicated to our respected parents and teachers.

Acknowledgements

This research was supported by United International University. We thank our fellow students who provided insights that greatly assisted the research. We are also in great depth to Mr. Jeremy Howard of University of Melbourne for his online courses.

Finally we would like to thank our respected teachers, particularly Dr. Swakkhar Shatabda and Ms. Sheikh Adilina of United International University for their help and guidelines. This research wouldn't have been possible without their help. We thank them for their comments on the earlier manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

DNA is the blueprint for the cell. It contains all the information and instruction that codes for the development and function of living things. But DNA does not do it by itself. There are thousands of DNA-binding proteins that help modulate DNA's functions. DNA-binding proteins have an indispensable role in major cellular processes. DNA-binding proteins perform two main functions: to organize and compact the chromosomal DNA and to regulate and effect the processes of transcription, DNA replication, and DNA recombination .However identifying the proteins that bind single or double-stranded DNA in the major groove is one of the challenging task. In early research DNA-binding proteins were identified by different types of experiment which includes filter chromatin immune precipitation on microarrays, binding assays, X-ray crystallography, genetic analysis and NMR.

1.2 Motivation

Though these experimental approaches are able to identify DBP's, it takes more time and resources. Due to the development of Next generation sequencing (NGS) also familiar as High-throughput sequencing it is easier than previously to sequence DNA and RNA quickly ,as a result the number of new protein sequence has increased. According to The Universal Protein Resource Knowledgebase ,protein sequence repository is increasing .Therefore to manage the increasing protein sequence data, there should be an efficient and time-saving computational methods to identify DNPs. .

1.3 Objectives of the Thesis

Nowadays machine learning algorithms are very effective as computational method to recognize DNPs.ML based automated methods are very fast and effective to predict whether a protein se-

quence binds to DNA or not. Our objective for this thesis was to build a predictor that can identify if a protein is DNA-Binding or not.

1.4 Organization of the Thesis

The thesis is organised as follows:

Chapter 2 provides related works.

Chapter 3 presents the proposed method.

Chapter 4 discusses the results and experimental analysis.

Chapter 5 presents the conclusions, summaries the thesis contributions, and discusses the future works.

Chapter 2

Background

2.1 Related Work

The robustness of structure-based predictors[1] are highly dependent on the structural information of protein sequence. There are many approaches based on the three-dimensional (3D) structure of a query protein to predict whether it binds DNA. There are some methods that compare the query protein and DBPs to find similarity [2, 3].As the protein sequence data is increasing day by day and the DNA binding domain is novel. Previous data is very important to predict DBPs for different types of methods. Though there are some approaches that do not rely directly on previous data may be helpful. Those methods often rely on electrostatic features of the proteins. It is found that DNA is negatively charged, and the DNA binding region of the protein is often positively charged, this information or knowledge is used for the method of Nimrod et al[4]. the entire approach is divided into several steps. The first step in predicting functional region of the sequence from its evolutionary profile. Next, several features of the predicted functional region and global features of the protein are calculated, among them are cluster-based amino acid conservation patterns and the average surface electrostatic potential. For the last step a RF classifier predicts whether the amino acid sequence is a DBP or not. Another similar approach is of Ahmad et al. here the net charge of protein, electric dipole moment and quadrupole moment tensors , three structural perspectives are considered to represent proteins with 62 structural features .Although these methods give promising result to predict the DBPs, in the absence of enough the structural information the performance of the predictor may vary as the structural information of proteins is not always available as much as necessary.

Recently, sequence based predictors got more attention of researchers to improve the performance of identifying the DNA binding proteins as that do not need the information of protein sequence structure. The sequence based predictors find the suitable feature extractor algorithms to identify the characteristics of DNA binding proteins and non DNA binding proteins. There are many machine learning based algorithms are used to solve the problem as well as to improve the

prediction performance such as Support Vector Machine (SVM),in 2007 Kumar et al. developed a method named DNABinder for predicting DNA-binding proteins using SVM and PSSM profiles .It achieved the maximum accuracy of 79.80% and 86.62% using amino acid composition and PSSM profiles. The only downside is, it was highly dependent on a certain or on a used dataset. A webserver is also available to identify DNA-binding proteins and domains from query amino acid sequences `http://www.imtech.res.in/raghava/dnabinder`.Another similar type of approach is proposed by Fang et al. in 2008[5] where autocross-covariance (ACC) transform, pseudo-amino acid composition and dipeptide composition are used with SVM .This approach is efficient to predict DBPs and non DBPs using the primary sequence. The best result is acquire from PseAA with the overall accuracy of 96.6% and the sensitivity of 90.7%. In 2009 kumar et al. report a random forest algorithm DNA-Prot, to identify DNA binding proteins from protein sequence. To identify sequence similarities between DNA and protein database, BLAST is a useful tool. In 2010, Robert E. Langlois and Hui Lu used this pattern based ML protocol to identify DNA-binding proteins.

Chapter 3

Materials and Methods

For our methodologies we used two distinct approach. The first one includes the 5 standard steps introduced by Chou [6] and summarized by Saifur el al.[7] which follow down below.

1. Acquire a standard test and train dataset.
2. Represent features in form of a feature vector.
3. Develop a classification algorithm.
4. Neutrally evaluate the classification algorithm
5. Creating public access to the classifier.

However this approach has a major drawback, that is it requires various algorithms to extract the features. Those algorithms are often dataset specific. And it needs human interaction to extract the features. So we came up with another approach which doesn't require featurized data but rather takes raw amino acid sequence as input. The second approach isn't dataset specific.So the second approach works as follows.

We take the model that we've discussed in our first section and add a convolutional block to it.Along with the addion on a convolutional block we replace the features that we've used with a set of embedding vectors where each vector represents an L-dimensional point. To generate the features from these embedding vectors we create a 2D matrix from a sequence of proteins of length N. The size of our final 2D matrix is (L x N). On this matrix we apply repeated convoluons and sub-sampling. Each convolutional is done with a window-size of $L * 31$ and on each layer we apply 128 filters to generate 128 unique feature-maps. These feature-maps are then sub-sampled using max-pooling to shrink the size of the feature-maps. These reduced feature-maps are applied to repeated convolutions and sub-sampling unl we are le with a K feature maps with dimensions of 1x1. These feature-maps are treated as K features and fed into the neural-network that we developed in the previous step.

3.1 Datasets

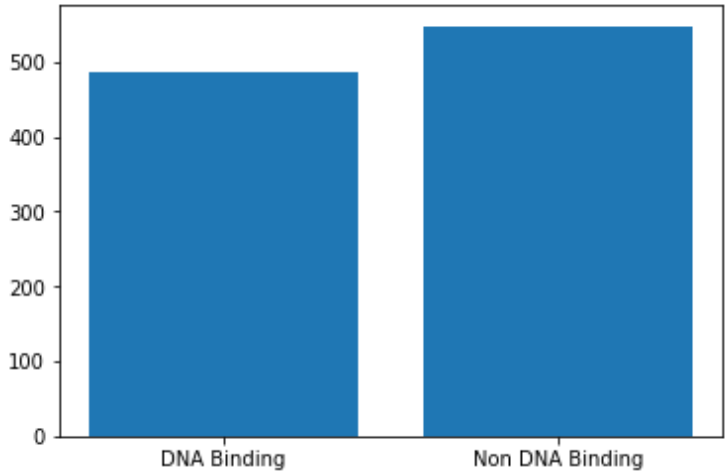


Figure 3.1: Class Label Count of PDB 1075 Dataset

For evaluating the predictor, credibility and accuracy of the dataset is crucial. The datasets we used to test our predictor were extracted from Protein Data Bank (PDB: `http://www.rcsb.org/pdb/home/dome.do`) using certain keywords such as "DNA-binding protein","DNA-binding" and so on. Our training dataset PDB1075 was originally extracted by Liu et al.[8].The PDB1075 dataset contains 525 positive DNA-binding protein sequences and 550 negative sequences. The validation set was compiled by Lou [9],was also extracted from Protein Data Bank contains 93 positive DNA-binding protein sequences and 93 negative ones. Both the datasets have been around for a couple of years,and contains desirable number of protein sequences.

3.2 Feature Selection Techniques

3.2.1 Feature representation for Methodology 1

Here for Methodology 1 we used 7 set of features used by Adilina et al. [10] These 7 set of features in total generates 32620 features.

1. **Monograms** :In order to find monograms recurrence of each individual amino acids was determined and then was normalized by the length of the sequence.

$$F_A = \frac{1}{L_M} \sum_{i=1}^{L_M} match(R_A, a_j) \tag{3.1}$$

where:
L_M = linear measure of the sequence
a_j = an amino acid from alphabet Σ

R_i= an amino acid at specific position i

The function works as following,

$$match(S1, S2) = \begin{cases} 1 & \text{if } S_1 == S_2 \\ 0 & \text{else} \end{cases} \tag{3.2}$$

So,There are 20 monograms.

2. **Bigram**: To find the bigrams, recurrence of two successive amino acids was taken into account.Just as the monograms,the frequencies of those bigrams were normalized as well. From the 20 amino acids 400 bigrams are generated.

$$F_A = \frac{1}{L_M} \sum_{i=1}^{L_M-1} match(R_iR_i + 1, S_j) \tag{3.3}$$

where:
S_j = a di-amino acid string taken from Σ^2

3. **Trigram**:Trigrams are determined same as the bigrams. Except for two successive amino acids, three are taken into account.

$$F_C = \frac{1}{L_M} \sum_{i=1}^{L_M-2} match(R_iR_i + 1R_i + 2, S_j) \tag{3.4}$$

where:
S_j = a tri-amino acid string taken from Σ^3

4. **Gapped bigram**:Gapped bigrams are formed by calculating frequency of each possible amino acid pairs of precise distance ignoring all others in between [11, 12].

$$F_D = \frac{1}{L_M} \sum_{i=1}^{L_M-g} match(R_iR_i + g + 1, S_j)(1 \leq g \leq 20) \tag{3.5}$$

where:
S_j = a di-amino acid string taken from Σ^2
g = distance between two amino acids in the sequence
In this paper, we have used gaps, g = 1 , 2 , . . . , 20 . Thus the total number of features generated was 80000.

5. **Monogram Percentile Separation**: Here monograms are determined only for partial sequences. On first iteration only 10% of the sequence is taken into account and monograms are determined only for the partial sequence. Then on each iteration we gradually increase the length of the partial sequence by 10% and repeat the whole step until we take 100% of the sequence. We can generate 200 features for each of the protein sequences.

$$F_E = \frac{1}{P_m} \sum_{i=1}^{P_m} match(a_i, a_k) \tag{3.6}$$

where:
P_m = partial linear measure of sequence
a_j = an amino acid from alphabet Σ

6. **Bigram Percentile Separation**:This process is fundamentally same as monogram percentile separation. Only instead of a individual amino acid, a pair is taken into account. This generates 4000 features in total.

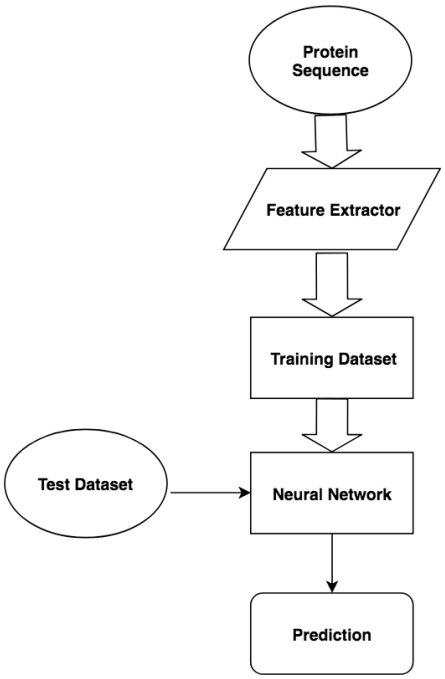
$$F_F = \frac{1}{P_m} \sum_{i=1}^{P_m - 1} match(R_i R_i + 1, S_j) \tag{3.7}$$

where:
P_m = partial linear measure of sequence
S_j = a di-amino acid string taken from the alphabet Σ^2

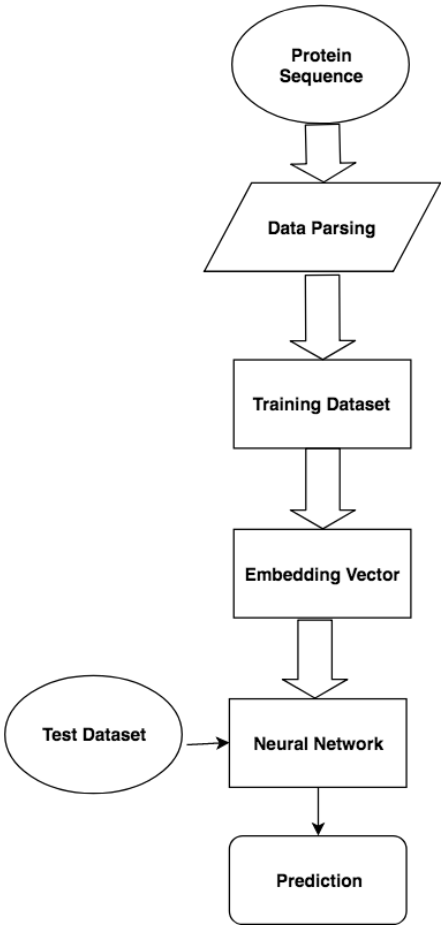
7. **Nearest Neighbor Bigram** :a_i and a_j are considered nearest neighbor bigram[8] if a_j is closest to a_i. Based on this concept, first 30 NNBs are considered to create 12000 features. Just as the previous ones, these values are normalized as well.

$$F_G = \frac{1}{L} distance(a_i, a_k)(1 \leq i \leq 20, k = 1, 2.....30) \tag{3.8}$$

where:
P_m = partial linear measure of sequence
a_j = an amino acid from alphabet Σ



(a) Methodology 1



(b) Methodology 2

Figure 3.2: Simplified methodology

3.2.2 Feature representation for Methodology 2

We extract the features with the help of a Convolutional Neural Network and an Embedding vector. To deal with sequences of varying length, we pad each sequence to a fixed length by appending a padding token to the end of each sequence. This greatly simplifies our model architecture, as our model can now expect an input of uniform length. We also discuss measures taken such that this does not affect the output of our model. Following is the architecture of model of feature extraction mode.

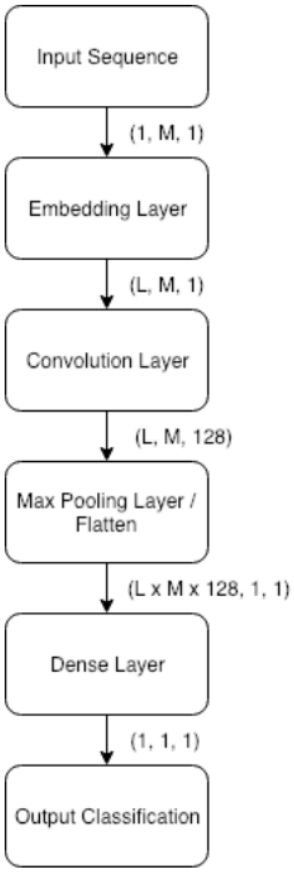


Figure 3.3: Architecture of Feature Extraction Model 2

3.2.2.1 Embedding Layer

The first layer in our model is a trainable embedding layer. Embedding layers are used to transform discrete inputs to points in vector space, called embedding vectors. Embedding vectors are a staple of natural language processing where they are used to represent words in an L-dimensional space, where L is the length of the vector. The distance relationships among the vectors, are a representation of their relation to one another. In natural language processing they represent the relation among input tokens, which are in general words, in a fixed set of possible words, i.e. a dictionary. For our model we consider each protein to be a discrete input token, and the set of 20 proteins to be our dictionary. We chose to ignore non-protein tokens, such as tokens used to pad our sequences, by representing them as zero vectors in our embedding space. This zero vectors do not have any affect on the outputs of the subsequent layers. The final output of the embedding layers is a uniform matrix of size LxM where M is the length of our input sequence.

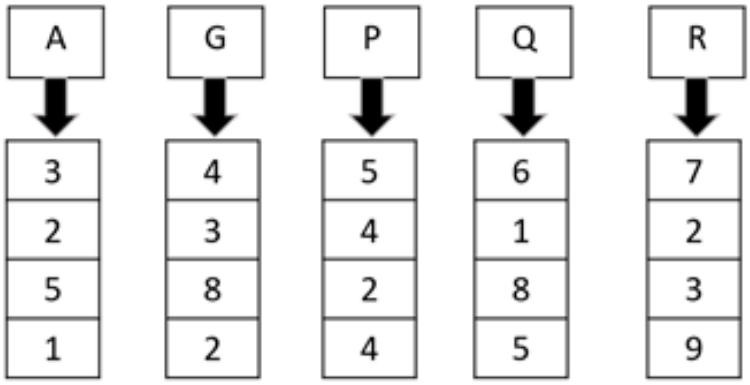


Figure 3.4: Simplified Model Architecture of Methodology 2

3.3 Classification Algorithms

For the purpose of classification we used deep artificial neural network. The deep learning model was composed of 12 layers excluding input and the output layer. There are four types of layers in our model; dense,activation,batch normalization and dropout. There's a brief description on each types of the layers below.

3.3.0.1 Dense Layer

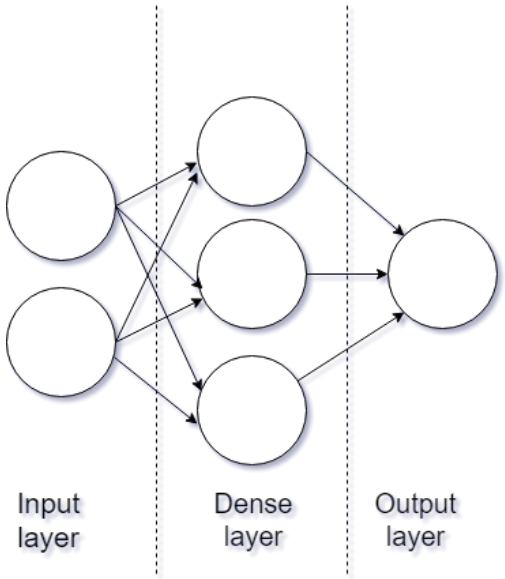


Figure 3.5: A simple Neural Network with one dense layer.

In a dense layer (also called a fully connected layer) each of the input nodes are connected to each of the output nodes. A simple figure of dense layer is shown on 3.5

3.3.0.2 Activation Layer

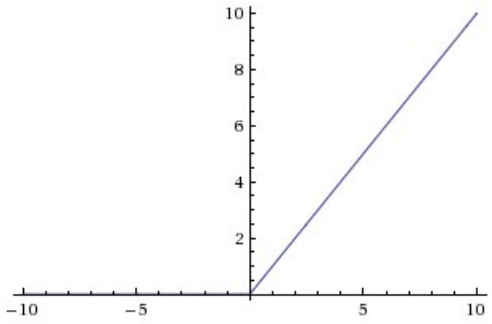


Figure 3.6: Rectified Linear Unit

An activation node determines output of one or more nodes through a function. We used the Relu as our activation function. [13] Relu functions,shown on equation 3.9 and figure 3.6, scales the output in a range of zero to one.

$$R(x) = max(0, x) \tag{3.9}$$

3.3.0.3 Dropout Layer

Next is the dropout layer. It is an elegant and simple way of dealing with overfitting which has been a challenge for deep neural net for a prolonged period of time. As mentioned by Srivastava et al.[14] dropout means randomly dropping a node along with all its connections from the network. In the process of randomly dropping a node it makes the network less dependent on a single node and thus reducing overfitting.

3.3.0.4 Batch Normalization Layer

Finally, the batch normalization layer. During the time of writing the document batch normalization was perhaps one of the single most important discoveries that happened in the field of deep learning.Change in distribution of each layer's input makes training Deep Neural Network difficult.Training Deep Neural Networks is complicated because as the parameters of the previous layers changes, distribution of each layer's inputs changes as well. This decelerates the training by requiring lesser learning rates and cautious parameter initialization, and makes it notoriously tough to train models with saturating nonlinearities. Sergey et al.[15] referred to this phenomenon as internal covariate shift, and address the problem by normalizing layer inputs.
To summarize, what we normally do is normalize the inputs of a network. So the general assumption was if inputs layers can be benefited by normalization, so should be the hidden layers as well. Batch normalization minimizes the amount by which the hidden unit values deviates. On top of that batch normalization qualifies each of the layers of the network learn by itself independently

of other layers. As mentioned by Sergy I. and Christian S. batch normalization has the following qualities among many others.

1. Batch normalization qualifies a network to have higher learning rates. Having the learning rate set to too-high may make the gradient get stuck at local minima or as well as explode or vanish. By normalizing the activations all through the network, batch normalization prevents insignificant change into parameters to being amplified into a large and sub-optimal changes.

2. While using Batch Normalization, a training example is used together with other examples in mini-batches, it no longer produces deterministic values for a given training example. In short it generalized the network and in the process diminish the need of Dropout[14].

3.3.0.5 Model Architecture

We started our experiment with a simple linear classifier and gradually added more layers as well as nodes on the layers. As seen on our experiment , we are benefited from layers to the architecture, but that is upto a certain point, adding more layers after that point only added overhead to the training process with little or no benefit. It was determined from experiment that 3 layers are the threshold value for number of layers. Adding more layers doesn't further help our cause. And for the number of nodes on each layer the idea was simple. The more nodes on the layer, the better it performed. So we kept adding more nodes to the layers until we ran out of VRAM on our GPU.

3.4 Performance Evaluation

In order to measure the validation of our classification algorithm we used a couple of widely used performance matrices. They are defined down below:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.10}$$

$$Sensitivity = \frac{tn}{n} \tag{3.11}$$

$$Specificity = \frac{tn}{tp + fp} \tag{3.12}$$

$$Specificity = \frac{tn}{tp + fp} \tag{3.13}$$

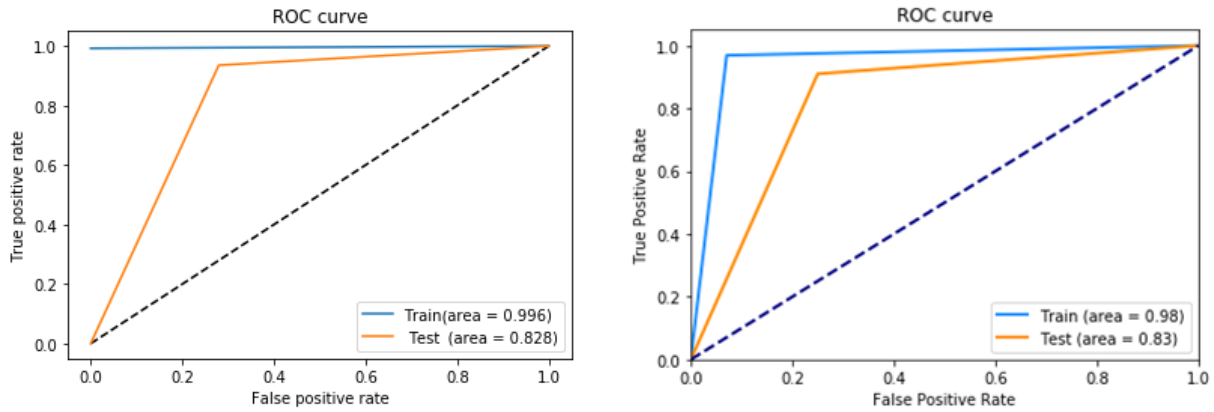
$$MCC = \frac{(tp * tn) - (fp * fn)}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{3.14}$$

Where:

n = number of instances of actual negative samples p = number of instances of actual positive samples tp = number of instances where positive samples are predicted correctly. tn = number of instances where negative samples are predicted correctly. fp = number of instances where negative samples are predicted incorrectly. fn = number of instances where positive samples are predicted incorrectly.

The very first three metrics, Accuracy,Sensitivity and Specificity has a value in range from 0 to 1. The higher the value is the better the classifier. 1 means a perfect classifier and 0 means the worst possible classifier.

The next one, MCC is ranged between +1 to -1. +1 means the perfect classifier, -1 means the worst and 0 means a random classifier



(a) Methodology 1 **(b)** Methodology 2

Figure 3.7: Area under ROC

Chapter 4

Experimental Analysis

In this section we discuss about the experiment in depth.

4.1 Experimental Setup

The experiments were conducted using two machines, first one was equipped with Intel Core i3-8100 Processor along with one nVidia Geforce GTX 1070ti graphics card and the second machine was equipped with Intel Core i3-3100 processor and had a nVidia Geforce GTX 1050ti graphics card. The whole application was written in Python 3.6 language with using a couple of libraries including but not limited to Keras, Scikit-learn and Matplotlib.

4.2 Comparison of Classification Algorithms

We compared our result with 12 other methods on the benchmark train and test dataset gradually in Table 4.1 and Table 4.2.Till our proposed method DPP-PseAAC was beyond doubt the best performing classifier for training dataset with accuracy of 95.91%. Our tool, **DeepDBP(ANN)** was a near perfect classifier for training dataset with an accuracy of **99.02%**. However in order to truly determine the degree of effectiveness of a model an evaluation through a independent dataset is necessary.

Table 4.1: Comparison of DeepDBP with previous methods on PDB 1075 dataset.

Method	Accuracy	Sensitivity	Specificity	MCC	auROC	auPR
DNAbinder	79.09	0.48	0.814	0.48	0.8140	-
DNA-Prot	72.55	0.8267	0.5976	0.44	0.789	-
iDNA-Prot	75.4	0.8381	0.6473	0.50	0.761	-
iDNA-Prot—dis	77.3	0.794	0.7527	0.54	0.831	-
PseDNA-Pro	76.55	0.79611	0.7363	0.53	-	-
iDNAPro-PseAAC	76.76	0.7562	0.7745	0.53	0.8392	-
HMMBinder	86.33	0.87	0.855	0.72	0.902	-
Kmer1 + ACC	75.23	0.7676	0.7376	0.50	0.828	-
Local-DPP	79.20	0.84	0.7450	0.59	-	-
iDNAProt-ES	90.18	0.9038	0.90	0.80	0.9412	-
DPP-PseAAC	95.91	0.941	0.9764	0.92	0.9884	-
Grouped Feature Selection	70.82	0.61	0.797	0.41	0.751	0.721
Recursive Feature Selection	71.04	0.62	0.799	0.43	0.751	0.724
DeepDBP(ANN)	**99.02**	**.98**	**0.97**	**0.992**	**0.996**	0.996
DeepDBP(CNN)	94.32	0.83	0.75	0.981	0.986	0.982

To our knowledge, till DeepDBP(CNN), Grouped Feature Selection had been the forefront method with accuracy of 82.26%. However our second methodology, DeepDBP(CNN)was able to gain even better accuracy off **84.31%**.Even though Grouped Feature Selection had a validation accuracy of 82.26%, the train accuracy wasn't as good as the validation accuracy. Which clearly indicates the model was underfitting; which led us to believe that there were improvements to be made. So we tried DeepDBP(ANN)and as predicted was able gain state-of-the-art result. But as described earlier DeepDBP(ANN)had very high training accuracy compared to validation accuracy. In order to solve the overfitting problem, we tried a different methodology, a novel approach, which not only solves the overfitting problem, but also get automates the feature extraction procedure. Which can be generalized to build classifier for different datasets and even for different problems as well.

Table 4.2: Comparison of DeepDBP with previous methods on PDB 186 validation dataset.

Method	Accuracy	Sensitivity	Specificity	MCC	auROC	auPR
DNAbinder	60.80	0.57	0.645	**0.22**	0.216	0.607
DNA-Prot	61.80	0.68	0.538	0.24	0.240	-
iDNA-Prot	67.20	0.667	0.667	0.34	0.344	-
iDNA-Prot—dis	80.64	0.800	0.800	0.54	0.831	-
PseDNA-Pro	76.55	0.7961	0.7961	0.53	-	-
iDNAPro-PseAAC	69.89	0.77	0.624	0.40	0.8392	0.775
HMMBinder	69.02	0.61	0.763	0.39	0.632	-
Kmer1 + ACC	70.96	0.83	0.591	0.43	0.431	0.752
Local-DPP	79.00	0.92	0.656	0.63	-	-
iDNAProt-ES	80.64	0.81	0.800	0.61	0.843	-
DPP-PseAAC	77.42	0.83	0.709	0.55	0.798	-
Grouped Feature Selection	82.26	0.95	0.699	0.67	0.823	0.745
Recursive Feature Selection	76.88	0.77	0.769	0.55	0.769	0.696
DeepDBP(ANN)	82.80	**0.98**	**0.97**	0.992	**0.99**6	**0.996**
DeepDBP(CNN)	84.31	0.83	0.75	0.981	0.986	0.982

Chapter 5

Conclusions, and Future Work

5.1 Conclusions

To conclude, at the time of writing the paper, to our knowledge this was the first approach toward deep learning in order to find DBP. As described previously it provides state of the art result with very short computation time. We tried deep learning with the conventional approach of extracting features with specified algorithm as well as feature extraction using deep learning. While our first approach was producing state of the art result, the second approach even exceeds the result of first one. And also while the first approach is dataset specific, the second approach is more generalized, can be applied to other datasets as well and doesn't require in-depth knowledge about the dataset.

5.2 Future Work

Furthermore we believe a LSTM [16] network with powerful enough machine can even improve the result we produced.

Bibliography

[1] M. S. Rahman, S. Shatabda, S. Saha, M. Kaykobad, and M. S. Rahman, "Dpp-pseaac: a dna-binding protein prediction model using chou's general pseaac," *Journal of theoretical biology*, vol. 452, pp. 22–34, 2018. 3

[2] M. Gao and J. Skolnick, "Dbd-hunter: a knowledge-based method for the prediction of dna–protein interactions," *Nucleic acids research*, vol. 36, no. 12, pp. 3978–3992, 2008. 3

[3] H. P. Shanahan, M. A. Garcia, S. Jones, and J. M. Thornton, "Identifying dna-binding proteins using structural motifs and the electrostatic potential," *Nucleic Acids Research*, vol. 32, no. 16, pp. 4732–4741, 2004. 3

[4] G. Nimrod, M. Schushan, A. Szilágyi, C. Leslie, and N. Ben-Tal, "idbps: a web server for the identification of dna binding proteins," *Bioinformatics*, vol. 26, no. 5, pp. 692–693, 2010. 3

[5] Y. Fang, Y. Guo, Y. Feng, and M. Li, "Predicting dna-binding proteins: approached from chou's pseudo amino acid composition and other specific sequence features," *Amino acids*, vol. 34, no. 1, pp. 103–109, 2008. 4

[6] K.-C. Chou, "Some remarks on protein attribute prediction and pseudo amino acid composition," *Journal of Theoretical Biology*, vol. 273, no. 1, pp. 236 – 247, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002251931000679X 5

[7] M. S. Rahman, S. Shatabda, S. Saha, M. Kaykobad, and M. S. Rahman, "Dpp-pseaac: A dna-binding protein prediction model using chouâ€™s general pseaac," *Journal of Theoretical Biology*, vol. 452, pp. 22 – 34, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022519318302327 5

[8] B. Liu, J. Xu, X. Lan, R. Xu, J. Zhou, X. Wang, and K.-C. Chou, "idna-prot— dis: identifying dna-binding proteins by incorporating amino acid distance-pairs and reduced alphabet profile into the general pseudo amino acid composition," *PloS one*, vol. 9, no. 9, p. e106691, 2014. 6, 8

[9] W. Lou, X. Wang, F. Chen, Y. Chen, B. Jiang, and H. Zhang, "Sequence based prediction of dna-binding proteins based on hybrid feature selection using random forest and gaussian naive bayes," *PLoS One*, vol. 9, no. 1, p. e86703, 2014. 6

[10] S. Adilina, D. M. Farid, and S. Shatabda, "Effective dna binding protein prediction by using key features via chou's general pseaac," *Journal of theoretical biology*, vol. 460, pp. 64–78, 2019. 6

[11] J.-M. Chang, E. C.-Y. Su, A. Lo, H.-S. Chiu, T.-Y. Sung, and W.-L. Hsu, "Psldoc: Protein subcellular localization prediction based on gapped-dipeptides and probabilistic latent semantic analysis," *Proteins: Structure, Function, and Bioinformatics*, vol. 72, no. 2, pp. 693–710, 2008. 7

[12] M. Ghandi, M. Mohammad-Noori, and M. A. Beer, "Robust $$k$$k-mer frequency estimation using gapped $$k$$k-mers," *Journal of Mathematical Biology*, vol. 69, no. 2, pp. 469–500, Aug 2014. [Online]. Available: https://doi.org/10.1007/s00285-013-0705-3 7

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994. 12

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. 12, 13

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. 12

[16] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014. 18