Classification of DNA-binding and non-DNA-binding proteins

Jain Paridhi, Shah Keval Rahul, Vedam Akshata Jaishankar, Erica Yap Ping Xuan
Team 16

Abbreviations

AI - Artificial Intelligence, CNN - Convolutional Neural Network, DBP - DNA-Binding Proteins, DL - Deep Learning, KNN - K-Nearest Neighbours, LogReg - Logistic Regression, LSTM - Long-Short Term Memory, MCC - Matthews Correlation Coefficient, ML - Machine Learning, NB - Naive Bayes, ReLu - Rectified Linear unit, RF - Random Forest, RNN - Recurrent Neural Network, SVM - Support Vector Machines

Introduction

Classification between DNA-binding and non-DNA-binding proteins has been a longstanding challenge in molecular biology. DNA-Binding Proteins (DBP) are characterised by DNA-binding domains and have an essential role in genetic activity. For instance, DBPs control gene expression (that is, which genes are expressed) by interacting with specific DNA sequences. This, in turn, affects processes like cell differentiation and responses to environmental stimuli (Hudson and Ortlund, 2014). Another key function of DBPs is linked to diseases; Mutations and malfunctions in DBPs are linked to various diseases including genetic disorders (Zhang et al., 2018). Other functions include DNA replication, packaging, repair and rearrangement (Gao and Skolnick, 2009). Since DBPs play such a key role in genetic activity, identifying them is a crucial step in understanding cellular processes and furthering medical research.

Protein sequences, composed of 20 possible amino acids, can vary widely in length, resulting in an enormous combination of potential patterns. This complexity and scale can be handled using ML and DL, replacing a complicated and time-consuming human process.

In this project, we aim to classify whether proteins are DNA-binding or not using ML and DL techniques we have learned in this module and other techniques that have previously been applied to this problem.

Related Works

Over the years, various ML algorithms have been applied to the DBP problem. Commonly used classifiers include SVM, RF, naïve Bayes classifier and kNN classifier, all of which have been used by multiple researchers (Mishra et al., 2019). Of these classifiers, SVM and RF seem to be the most commonly used. Xu et al. in 2015 applied an SVM to classify DBPs and achieved an MCC of 0.622. Nimrod et al. in 2010 applied an RF classifier with a sensitivity and precision of 0.90 and 0.35 respectively. In this project, we will use these classifiers among others as baselines for our dataset.

Additionally, ensemble classifiers combining several models have also been used. Such classifiers combine the output of the individual models in such a way that they can yield the best possible final predictions. The performance of these classifiers depends on several factors, especially hyper-parameter tuning (Sarmah et al., 2024). One application of an ensemble classifier in the DNA-Binding Protein problem is the StackDDPred model developed by Mishra et al. in 2019.

This classifier stacks SVM, LogReg, kNN and RF and achieves an MCC of 0.7990 (Mishra et al., 2019). The improved results of the StackDDPred model show the benefits of stacking several models to improve performance. Therefore, we will be stacking the best baseline models to create an ensemble classifier. We will also perform hyper-parameter tuning to further enhance the performance of our model.

Recently, DL approaches have also shown promise in DBP classification. For example, Shadab et al. (2020) developed a DL model, DeepDBP-ANN (artificial neural network) which achieved an impressive test accuracy of 82.80%. Another promising approach is the combination of CNN (Convolutional Neural Network) and bidirectional LSTM (BiLSTM), where the CNN captures local sequence features, and the BiLSTM captures long-term dependencies (Hu et al., 2019; Zhang et al. 2019). This architecture has shown improved performance, with Hu et al. reporting accuracies above 90%.

A simpler DL model that has been used is RNN in its variant forms of LSTM and GRU (Gated Recurrent Unit) (Shen et al., 2018). This RNN model divides the protein se-

quences into k-mers and then uses those k-mers as inputs for Word2Vec to generate embeddings for the RNN model. Our implementation of the RNN model is inspired by Shen et al.'s approach. However, we replace the GRU with a Bidirectional LSTM (BiLSTM) due to its enhanced capacity for capturing long-term dependencies and processing sequence data bidirectionally, potentially leading to improved performance.

Dataset

Separate training and testing data sets have been provided. The training dataset has 53,285 protein sequences and labels indicating whether or not they are DBPs, and the test dataset has 13,452 protein sequences with their corresponding labels. Based on the instructions given, the lengths of the sequences should be in the range 50-3,000 and there are 20 unique amino acids. Therefore, we filtered out any sequences that did not meet these criteria. We also filtered out sequences with missing values and sequences that were duplicates. Once this data cleaning was complete, the training data had 53,238 sequences and the testing data had 13,443 sequences.

We also filtered out sequences with low complexity regions (LCR). LCR are segments with a biased amino acid composition, often containing repetitive sequences (Mier et al., 2020). If used while training, the model might overfit to the repetitive pattern and generalize poorly for unseen data. The SEG algorithm is normally used to detect LCR in proteins, but it was computationally expensive for our dataset due to its size. Therefore, we used the Shannon entropy instead which detected no LCR in the training or testing data.

Balancing the Data

The training data set showed severe imbalance in positive (17,829 samples) vs. negative classes (35,409 samples). This causes models to be biased towards the majority classes and reduces its effectiveness. In order to balance the underrepresented positive class, we experimented with Synthetic Minority Over-sampling Technique (SMOTE) and assigning class weights. The latter was later rejected as it caused the baseline models to score poorly in observed performance metrics. Furthermore, SMOTE allows for diversity in feature space due to interpolation, as opposed to random oversampling, which could lead to data duplication.

Essentially, Synthetic Minority Over-sampling Technique (SMOTE) helps balance classes by generating synthetic data by interpolating between a minority class and its nearest neighbours. Through this, SMOTE ensures a balanced training dataset, which is crucial for improved model performance. Our original training dataset contained 53,238 samples which becomes 70,818 samples once SMOTE has been applied to the dataset.

Sr. No.	Method of Feature Extraction	Reason of Rejection
1	ESM	ESM allows a max length of 1024 while some of our sequences are of length 3000. We tried truncating sequences and running ESM, but it crashed due to lack of RAM.
2	ProtBERT	Runtime of 6 days for 53000 sequences with our configuration.
3	Unirep	Crashed due to a lack of RAM.
4	BioVec	Crashed due to a lack of RAM.
5	CNN+ BiLSTM	This is very popularly used currently but it would take 5 hours to run for our configuration.

Figure 1: Evaluation of Feature Extraction methods

Feature Extraction

FastText is a word embeddings and text classification library that generates embeddings by breaking down strings into smaller components, or n-grams. This approach enables FastText to capture sub word (or subsequence) information, making it well-suited for large-scale datasets with complex patterns. In the context of protein sequences, which are chains of amino acids, short motifs or subsequences often determine function and binding properties. FastText's n-gram embeddings capture these local patterns within sequences, enabling the model to identify binding tendencies based on small but influential motifs, essential for analysing protein binding behaviour.

While FastText provides meaningful feature embeddings, it is not designed specifically for proteins but more broadly for words. Therefore, we experimented with several other methods of feature extraction that are summarized in the Figure 1 above.

While these five methods failed, we were able to generate embeddings using Spectrum Kernel vectors. These are more commonly used with protein sequences and would answer our concern regarding FastText. To make a choice, we fitted our baseline models using features from both feature extraction methods and compared them. The models using Spectrum Kernel vectors had a mean MCC of 0.3016 and the models using FastText had a mean MCC of 0.3982. This difference in performance informed our choice to use FastText for feature extraction.

Methods

We aimed to utilise two models for predictions: a Recurrent Neural Network (RNN) with a bi-directional Long Short-Term Memory (LSTM) and a Stacking Ensemble, and thereby compare their performance on the test dataset given to us to choose our final model.

Our approach to the problem would include splitting the

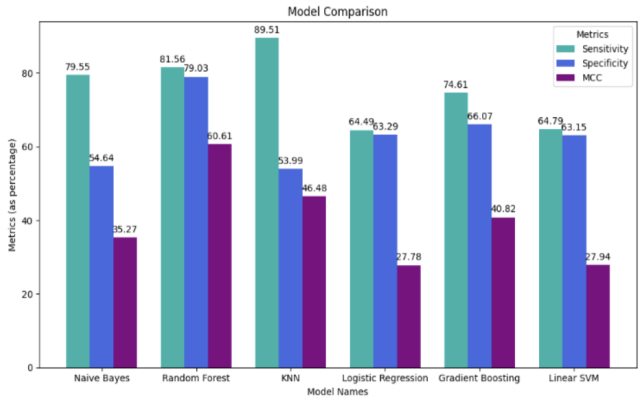

Figure 2: Performance Metrics

training dataset into train-test in order to cross-validate the hyperparameters and evaluate the performance metrics on the validation set before testing and predicting on the test set.

Model 1

The first step was to train all the baseline models to evaluate which ones to include in the ensemble classifier. We split the training data into train-test to enable us to evaluate performance metrics for each model. We used naïve Bayes, kNN, RF, SVM, GB, and LogReg. To evaluate the performance, we computed the Sensitivity, Specificity and MCC Score for each model. The performance of all the baseline model is visualized in Figure 2 above.

Based on these individual performances, we chose RF, kNN and GB as our base models and LogReg as our meta-model. Our reasoning for these decisions is given below:

1. RF: RF provides the most balanced performance and the highest MCC score. It is also suitable for high-dimensional datasets since it leverages multiple decision trees trained on different feature subsets and can handle noisy data effectively.
2. kNN: kNN was primarily chosen to improve the Sensitivity of the stacking model. It is also a simpler model which makes it a good choice to stack with models like RF and GB which are complex.
3. GB: GB was chosen to provide diversity to the stacked model. It is also effective because it builds an ensemble of weak learners, where each successive model corrects the errors of the previous one, which helps it capture subtler patterns in the data. Like RF, it is well suited for high-dimensional, noisy data.
4. LogReg: While LogReg does not give the best individual performance, it is a commonly used meta-model because of its simplicity and ability to combine the diverse model outputs.

The stacked model was fitted on the same data and evaluated using the same performance metrics. It showed an all-around improvement with a sensitivity of 0.8064, specificity of 0.8261, and an MCC score of 0.6326. To further enhance this performance, we experimented with hyperparameter tuning using the `RandomizedSearchCV` method to fine-tune key parameters:

- `rf__n_estimators` and `rf__max_depth` for RF
- `knn__n_neighbors` for KNN
- `gb__n_estimators` and `gb__learning_rate` for GB
- `final_estimator__C` for LogReg

These parameters control key aspects like model complexity and overfitting, helping to ensure balanced performance across metrics. We chose `RandomizedSearchCV` instead of `GridSearchCV` to reduce computational time and increase efficiency, as it samples a fixed number of parameter settings (in this case, 10) rather than trying every combination.

Once the optimal parameters were identified, the model was retrained on the entire training dataset (instead of the 80% used initially) and then evaluated on the test data.

Model 2

The second model is an RNN-based architecture with three bi-directional LSTM layers. RNN are well-suited for sequential data such as protein sequences, as they retain information from previous steps during data processing. It can also handle sequences of varying length (which are observed in our dataset). To solve the 'vanishing gradient' problem and capture long-term dependencies in the sequences, we transitioned from traditional RNN to LSTM layers. We further enhanced the model by making the LSTM layers bidirectional, enabling the model to process sequences from front-to-back and vice versa.

To prevent overfitting, we employed three regularization techniques. Firstly, a Dropout layer was added after each BiLSTM layer, which randomly removes a fraction of neurons during training to discourage over-reliance on specific neurons. Secondly, a kernel regularizer was applied to penalize large weights, encouraging simpler and more generalized models. Thirdly, early stopping was added to monitor validation loss and halt the training once the validation loss started to plateau, preventing overfitting due to extended training.

After experimenting with various combinations of hyperparameters, we identified three key parameters: the number of LSTM units, activation function, and dropout rate. We used nested for loops to find the best combination, iterating over LSTM units (50, 100, 150), activations ('tanh', 'relu'), and dropout rates (0.2, 0.25, 0.3). Figure 3 presents the top five configurations based on the MCC Score after testing on the 20% of the training data initially separated.

Sl. No.	LSTM units	Activation	Dropout	Sensitivity	Specificity	MCC Score
1	150	'relu'	0.25	0.7562	0.7235	0.4798
2	100	'relu'	0.2	0.7674	0.7106	0.4786
3	150	'tanh'	0.2	0.7424	0.7361	0.4784
4	50	'tanh'	0.2	0.7419	0.7278	0.4745
5	150	'relu'	0.2	0.7350	0.7363	0.4713

Figure 3: Top 5 Configurations on MCC Score

Figure 4: Visualization of training of the best RNN model

The best-performing model, with 150 LSTM units, 'relu' activation, and a dropout rate of 0.25, was saved and trained on the entire training dataset (Figure 4) before being applied to the test dataset.

Results and Discussion

Once the two models are trained on the entire training data, they are tested using the test data. The performance of both models is tabulated in Figure 5 below.

These results reveal both the strengths and limitations of the models built in this project. The stacked model and the RNN model each offer unique insights but demonstrate divergent performance across key metrics. The stacked model achieved a high specificity of 0.8472, indicating its reliability in correctly identifying non-DNA-binding proteins. However, its sensitivity of 0.4601 suggests that it struggled to accurately identify DBPs. Conversely, the RNN model had an increased sensitivity of 0.6802 – a significant improvement over the stacked model. However, this came at the expense of the specificity. Overall, the RNN seems to be a better model with higher MCC and F1 scores. This aligns with the recent rise of Deep Learning approaches to the DNA-binding protein problem.

Model	Sensitivity	Specificity	MCC Score	F1 Score	Accuracy
Stacked Model	0.4601	0.8472	0.3323	0.5208	0.7186
RNN Model	0.6802	0.7237	0.3869	0.6089	0.7093

Figure 5: Performance of models on test data

Overall, this project underscores the complexity of classifying DNA-binding proteins, as both models exhibited strengths in certain metrics while facing challenges in others. The high specificity of the stacked model and the improved sensitivity of the RNN model highlight the trade-offs inherent in this classification task. These findings emphasize the need for a model that excels in both sensitivity and specificity, capturing the intricate patterns necessary for DNA-binding protein identification.

References

W. H. Hudson and E. A. Ortlund
"The structure, function and evolution of proteins that bind DNA and RNA," Nature Reviews Molecular Cell Biology, vol. 15, no. 11, pp. 749–760, Nov. 2014, doi: https://doi.org/10.1038/nrm3884. *N. Zhang, Y. Chen, F.*

Zhao, Q. Yang, F. L. Simonetti, and M. Li
"PremPDI estimates and interprets the effects of missense mutations on protein-DNA interactions," PLOS Computational Biology, vol. 14, no. 12, p. e1006615, Dec. 2018, doi: https://doi.org/10.1371/journal.pcbi.1006615. *Nosiba Yousif*

Ahmed et al.
"An Efficient Deep Learning Approach for DNA-Binding Proteins Classification from Primary Sequences," The International journal of computational intelligence systems/International journal of computational intelligence systems, vol. 17, no. 1, Apr. 2024, doi: https://doi.org/10.1007/s44196-024-00462-3. *S. Shadab, M.*

T. Alam Khan, N. A. Neezi, S. Adilina, and S. Shatabda
"DeepDBP: Deep neural networks for identification of DNA-binding proteins," Informatics in Medicine Unlocked, vol. 19, p. 100318, Mar. 2020, doi: https://doi.org/10.1016/j.imu.2020.100318. *A. Mishra,*

P. Pokhrel, and M. T. Hoque
"StackDPPred: a stacking based prediction of DNA-binding protein from sequence," Bioinformatics, vol. 35, no. 3, pp. 433–441, Jul. 2018, doi: https://doi.org/10.1093/bioinformatics/bty653. *N. Wang, J.*

Zhang and B. Liu
"iDRBP-EL: Identifying DNA- and RNA- Binding Proteins Based on Hierarchical Ensemble Learning," IEEE/ACM

Transactions on Computational Biology and Bioinformatics, vol. 20, no. 1, pp. 432–441, Jan. 2023, doi: https://doi.org/10.1109/tcbb.2021.3136905.

Z. Shen, W. Bao, and D.-S. Huang
"Recurrent Neural Network for Predicting Transcription Factor Binding Sites," Scientific Reports, vol. 8, no. 1, Oct. 2018, doi: https://doi.org/10.1038/s41598-018-33321-1.

M. Gao and J. Skolnick
"From Nonspecific DNA–Protein Encounter Complexes to the Prediction of DNA–Protein Interactions," PLoS Computational Biology, vol. 5, no. 3, p. e1000341, Apr. 2009, doi: https://doi.org/10.1371/journal.pcbi.1000341.

R. Xu, J. Zhou, H. Wang, Y. He, X. Wang, and B. Liu
"Identifying DNA-binding proteins by combining support vector machine and PSSM distance transformation," BMC Systems Biology, vol. 9, no. S1, Feb. 2015, doi: https://doi.org/10.1186/1752-0509-9-s1-s10.

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, Jan. 2010, doi: https://doi.org/10.1093/bioinformatics/btq019.

Upasana Sarmah, P. Borah, and D. K. Bhattacharyya
"Ensemble Learning Methods: An Empirical Study," SN Computer Science, vol. 5, no. 7, Oct. 2024, doi: https://doi.org/10.1007/s42979-024-03252-y.

P. Mier et al.
"Disentangling the complexity of low complexity proteins," Briefings in Bioinformatics, vol. 21, no. 2, pp. 458–472, Jan. 2019, doi: https://doi.org/10.1093/bib/bbz007.

Hu, R. Ma, and H. Wang
"An improved deep learning method for predicting DNA-binding proteins based on contextual features in amino acid sequences," PLOS ONE, vol. 14, no. 11, p. e0225317, Nov. 2019, doi: https://doi.org/10.1371/journal.pone.0225317.