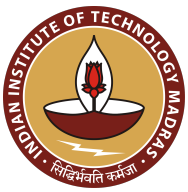

Indian Institute of Technology, Madras

Department of Computer Science & Engineering

Algorithmic Approaches in Computational Biology

Course Project Report

Resnet-Model predictor for classifying sigma promoters

Arnhav Datar -cs18b003@smail.iitm.ac.in ,
Anirudh Ajith -cs18b070@smail.iitm.ac.in

December 15, 2020

Abstract

The reliable detection and classification of promoters in DNA remains a challenging task even today. Recent advances in this field consist mainly, of the application of deep-learning techniques to solve this problem. We propose some modifications to a state-of-the-art deep-learning-based approach (by Amin et. al. [1]) used to identify and classify σ-promoters in the genome of *E. coli*. We use the large training set used by the original authors, and test our model on an independent test set. We obtain a test accuracy of **89.84%**. We furthermore do an analysis of our results using a confusion matrix to find for which classes the model is not performing up to the mark and how it can be improved.

Introduction

Promoters are short regions($\simeq$ 81 nucleotides in the *E. coli* Bacteria) in DNA. They are responsible for initiating transcription. Promoters are a vital component of expression vectors because they control the binding of RNA polymerase to DNA. Classifying these regions in DNA is of great importance to biologists because the initiation of the transcription is responsible for certain pathways and gene regulation. However, existing experimental methods that identify promoters are expensive and time-consuming. Also, formulating this as a computational motif finding problem does not yield good results since the instances of the promoters which occur, often differ significantly from the consensus strings.

This paper aims to improve upon the work by Amin et. al. [1] toward σ-promoter identification and classification (in *E. coli*) using convolutional neural network models and by applying deep-learning techniques. This is viable since large amounts of data about the *E. coli* genome and its σ-promoters are publicly available.

Deep-learning has shown a lot of promise in the field of bioinformatics [2, 3] in applications such as the prediction of protein targets and splicing sites in DNA, in analysis of clinical images, and recently, in the protein folding problem [4]. Skip connections and ResNet architectures[5] have been shown to improve performance in applications to bioinformatics. In this project, we extend work by Amin et. al. [1] by introducing skip-connections to their models.

Related Work

Given in Table 1 is a literature review for the subject with the corresponding accuracies. Below we have briefly explained the methods used in these papers. Amin et. al. [1] used a novel deep-learning-based approach for classification and achieved state of the art results from their model iPromoter-BnCNN. We seek to improve upon their work using additional deep-learning methods which have been shown to work well on other problems.

Bin Liu et. al.[6] used a 2-layer predictor for classification in their iPromoter-2L paper. They used a random forest for each layer. Meng Zhang et. al.[7] improved the accuracy using a multi-layer classifier called Multiply. They used a vast feature set that includes local and global

features. They followed up on this with a F-Score feature selection to select the best features from their feature set. However, being a feature-engineering approach there is a chance that their model has overfit.

The σ^{70} promoter is one of the most important and abundant promoters. It serves as a gate-keeping promoter. He et. al. used a feature engineering approach for classifying σ^{70} promoters. Qianzhong Li et. al.[8] used a novel position-correlation scoring matrix approach to classify σ^{70} promoters. Lin et. al.[9] in their iPro54 classifier used an SVM classifier on the pseudo-k-tuple nucleotide composition. Kai Song used a novel approach wherein a variable window Z-curve was used to extract the features to get good accuracies on *E. coli* and on *Bacillus subtilis*/ Silva et. al. [10] gave the DNA duplex stability to a neural network for the purpose of σ^{28} and σ^{54} classification. However, they get a lower score on our test-set.

Shah et. al. [11] introduced the BTSS finder tool for the same five classes we work on for *E. coli*, namely $\{\sigma^{24}, \sigma^{28}, \sigma^{32}, \sigma^{38}, \sigma^{70}\}$, they also worked on five other sigma factors in Cyanobacteria. They achieve a high accuracy although they operated on a different dataset. They used a feature engineering approach as well, with 30 features out of which the best 20 were extracted. However, feature engineering methods often lead to overfitting on the test set.

Method	Sn	Sp	Acc	MCC	Reference
iPromoter-BnCNN	**88.30%**	**88.00%**	**88.20%**	**0.763**	[1]
PCSF	78.90%	70.70%	74.80%	0.498	[8]
VW Z-curve	77.80%	82.80%	80.30%	0.61	[12]
Stability	76.60%	79.50%	78.00%	0.562	[10]
iPro54	77.80%	83.20%	80.5	0.61	[9]
iPromoter-2L	79.20%	84.20%	81.70%	0.634	[6]
MULTiPly	87.30%	86.60%	86.90%	0.739	[7]

Table 1: Result Review[1]

Results

Test Results

The test results are on the independent dataset described in Table 3. Using the above mentioned methods above we get an accuracy of 89.84% on the independent test-set. Refer Table 2 for detailed results.

metric	iPromoter-BnCNN (original)	our results
accuracy	88.2%	89.84%
specificity	88.0%	89.84%
sensitivity	88.3%	84.35%
MCC	0.763	0.735

Table 2: Results

We were able to surpass the iPromoter-BnCNN paper in accuracy and specificity as per Table 1, however, we are narrowly behind in the sensitivity and the MCC score. However the test dataset is rather small and hence a thorough comparison is not possible without access to a comparatively larger dataset is used.

Discussion

Confusion Matrix

Given below is the confusion matrix we obtained for the 5-class classification on the independant test set described in Table 3.

Predicted Class \ True Class	σ^{24}	σ^{28}	σ^{32}	σ^{38}	σ^{70}
σ^{24}	29 97%	2 50%	0 0%	0 0%	9 5%
σ^{28}	0 0%	1 25%	0 0%	0 0%	0 0%
σ^{32}	0 0%	0 0%	12 92%	0 0%	2 1%
σ^{38}	0 0%	0 0%	0 0%	2 20%	2 1%
σ^{70}	1 3%	1 25%	1 8%	8 80%	186 93%

As we can see above that the model performs very well on classifying σ^{24}, σ^{32} and σ^{70} classes, achieving an accuracy greater than 90%, for each class. It however doesn't perform so well for the σ^{28} and σ^{38} classes. We attribute this to the huge class imbalance for these promoters. This can be seen from Figures 1 and 2 and from Table 3.

Data

Description

The input data consists of a collection of sequential reads (each of length 81) obtained from the genome of a representative *E. coli* organism such that these reads have been partitioned into the following 7 classes:

$$\{\sigma^{24}, \sigma^{28}, \sigma^{32}, \sigma^{38}, \sigma^{54}, \sigma^{70}, \text{non-promoter}\}$$

Associated with these reads, the datasets also contain data regarding certain structural attributes of the dimers and trimers present in the reads. The reason for this, as stated by the original authors is that both the sequence of nucleotides and the structure of the resultant strand are important biological determiners for σ-promoters.

Therefore, the task at hand is to determine whether a given 81-mer is a promoter in the *E. coli* genome, and if it is, to further classify it into one of the 6 σ classes.

In this project, we will be using the same datasets as the ones used by Amin et. al. [1]). Given in Table 3 is a brief outline of the datasets we will be using. Since the independent benchmark dataset does not have any non-promoter or σ^{54} classes, we will not be able to obtain a test-accuracy for these classes. To ensure a fair comparison with other papers, we do not make classifiers for these classes.

Classes	Benchmark	Independent
Promoter	2860	256
Non Promoter	2860	0
σ^{24}	484	30
σ^{28}	134	4
σ^{32}	291	13
σ^{38}	163	10
σ^{54}	94	0
σ^{70}	1694	199

Table 3: DataSet

Imbalance in the dataset

As can be seen from Table 3, both the benchmark and independent datasets suffer from heavy class-imbalance. For example, the data contains 1694 instances of σ^{70} sequences but only 94 instances of σ^{54} ones. This poses a problem as standard data augmentation techniques cannot be used to produce realistic additional data. Various methods[13, 14] have been proposed in the past to handle imbalanced data. The authors of the original paper tackled this problem by creating a series of binary classifiers that would refine the set of strings they classify positively

into progressively smaller sets, when laid out in a stage-wise manner. We describe our similar approach in the following section.

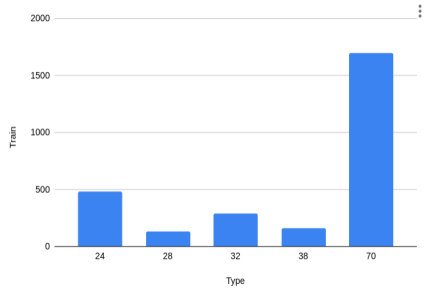

Figure 1: class-dist of training set

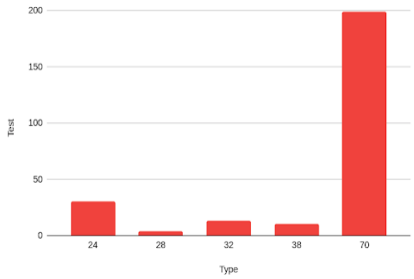

Figure 2: class-dist of independent test set

Input Generation

From each strand (of length 81), we generate 4 matrices that encode both the nucleotide sequence and the given structural properties of the strand:

1. A representation of the nucleotide sequence using a one-hot encoding for each nucleotide. The matrix formed will have a shape of $(81, 4)$.
2. A one-hot encoding representation for the sequence of trimers. Since there are $81 - 3 + 1$ trimers in a given strand, and there are 4^3 possible trimers, this matrix will have a shape of $(79, 64)$.
3. A matrix encoding the structural properties of the dimers in the strand. We are provided with a set of 90 structural properties for each of the dimers. Since there are $81 - 2 + 1$ dimers, we get a matrix of shape $(80, 90)$.
4. A matrix encoding the structural properties of the trimers in the strand. We are provided with a set of 12 structural properties for each of the trimers. Since there are $81 - 3 + 1$ trimers, we get a matrix of shape $(79, 12)$.

For a given strand, these 4 matrices are what are passed as input to our model.

Methods

Classification Approach

Both the independent dataset and the benchmark dataset was taken from the Supplementary Data section of [1]. Because of the heavy imbalance in the dataset (described above), we use a series of binary classifiers to discriminate between the σ-promoters. We create 4 binary classifiers which we train to partition the σ-promoter 81-mers into the sets as listed below. The reason for this precise choice of sets is that we are able to ensure that the sizes of the positive and negative sets for each of these classifiers are roughly equal, helping us combat problems caused by class-imbalance.

We then create a classification pipeline by laying these binary classifiers out in a stage-wise manner as depicted below.

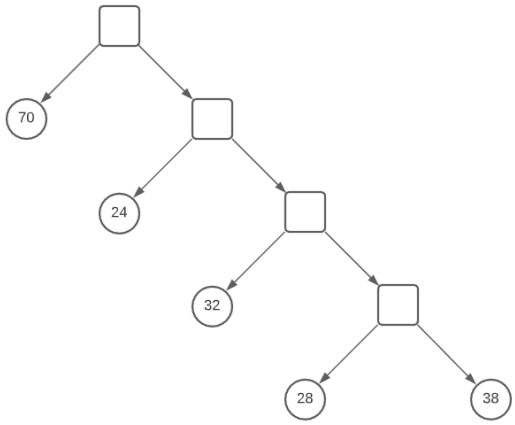

Figure 3: Classification Tree

The four corresponding binary classifiers are

1. σ^{70} vs $\{\sigma^{24}, \sigma^{28}, \sigma^{32}, \sigma^{38}\}$
2. σ^{24} vs $\{\sigma^{28}, \sigma^{32}, \sigma^{38}\}$
3. σ^{32} vs $\{\sigma^{28}, \sigma^{38}\}$
4. σ^{28} vs σ^{38}

The Neural Network

We seek to improve the model given by Amin et. al. [1]. Hence we propose the following modifications. For a complete view of the conv-net see Figure 5, 6.

Adding Skip connections

We experimented with the addition of skip-connections to the 4 branches of the conv-net to test the performance of a ResNet type approach. However, we found that the addition of skip-connections were beneficial to the performance of the model only when they were added to the 2 branches of the model which take in dimer and trimer structural properties as their input. We could not detect any improvement in our validation accuracy when we added skip connections to the branches which took in the one-hot encodings as input.

Adding MaxPool Layers

1. **At the start**

   We found that the addition of specific maxpool layers after the first convolution in each branch helped improve performance. We found that our model worked best when we

added maxpool layers of (size, stride) = (3,3) to the branches which took in trimer one-hot-encoding and trimer structural properties as input. We also added a maxpool layer of (size, stride) = (2,2) to the branch which takes in dimer structural properties as input.

2. **At the end**
   We also added maxpool layers of (size, stride) = (3,3) at the end of each branch (just before flattening and concatenation) to reduce the dimensionality of the data and the number of parameters in the model. We also found that this helped combat overfitting.

Reducing the layers

We also decreased the lengths of the branches that take in the one-hot encodings as input by dropping one convolutional layer from each of them. They contained 3 convolutional layers each in the model proposed by the original authors. We found that reducing the number of convolutions in each of these layers to just 2, helped prevent overfitting. We also found that reducing the number of fully-connected layers post-concatenation from 3 to 2 served the same purpose.

Acknowledgments

We would like to thank our course instructor Dr. Manikandan Narayanan for giving us the opportunity to conduct this project in a structured way while giving key insights and guidance. We would also like to thank the TAs of this course, particularly Aditya Jeevannavar, who also helped us and gave us valuable critique. We would furthermore like to thank our fellow classmates, for attending our presentation, and giving constructive criticism.

Author Contributions

We wrote the code for the report almost equally. We worked on a shared Google Colab notebook, so both of us kept contributing to the code. Meanwhile, we kept discussing ideas, so its very difficult to tell who had it. Hence we think that there is equal contribution in this regard.

In terms of writing the report similarly, we worked on a shared Overleaf document so both of us kept writing to the same document albeit to different sections. We again believe that both of us should get equal contribution in this regard. The slides were mostly prepared by Arnhav Datar, and were reviewed and improved by Anirudh Ajith.

Appendix

Changing the Classification Tree

The classification tree used by us in Figure 3, was based on providing balanced datasets to each classifier. However, this approach completely ignored any relevant biological information.

We tried to incorporate this in the model while keeping datasets as balanced as possible. We tried to use the tree described in Figure 4. The reasoning behind this approach was that the σ^{24} and σ^{32} are responsible for heat stress response and heat shock respectively. Hence we thought that they may also be structurally similar.

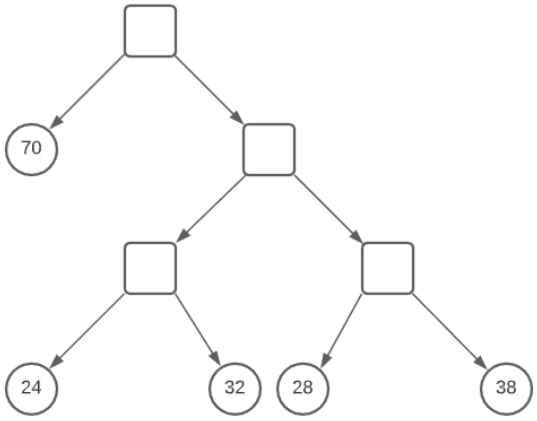

Figure 4: Classification Tree

However, this method reduced the accuracy. Some possible reasons:-

1. The σ^{24} and σ^{32} may not be structurally similar based on the dimer and trimer properties we are providing.
2. The test dataset after the σ^{70} promoters are classified involves only 57 samples. This number is too less to conclude anything.

Random Forest of One vs All classifiers

Motivated by the idea of allowing for a mechanism to learn the structure of the optimal classification tree by itself, rather than hardcode it like we had been doing, we experimented with the possibility of training one-vs-all binary classifiers for each of the five σ-factor classes.

Our plan was to pass a test strand (i.e. the 4 corresponding matrices that represent it) through each of the 5 one-vs-all binary classifiers to get their predictions (and the respective confidences) before feeding these 10 outputs to a random forest model which would act as the final stage in the pipeline and make the final prediction. However, class imbalance posed a major problem here as well. For the two smallest classes, namely σ^{28} and σ^{38}, we could not prevent the corresponding binary classifiers from either

1. always predicting false OR
2. overfitting the few training samples

Further Work

The results we have achieved were with a single validation set in the training data. To get a more statistically sound model, we need to do a k-fold cross-validation to get the best hyper-parameters. This may further improve the accuracy.

The test dataset was a small dataset with only 256 samples. To further solidify the results, a larger dataset is required. Furthermore, there is a significant imbalance in the dataset right now as can be seen in Figure 2 and 1. A more balanced dataset is required for better results.

In our initial proposal, we had mentioned that we may try attention models in case ResNet architectures do not give positive results. An approach in that direction can be attempted.

Code Availability

The code to run our models can be found on this link. The training code is available from the authors upon reasonable request.

Supplementary Data

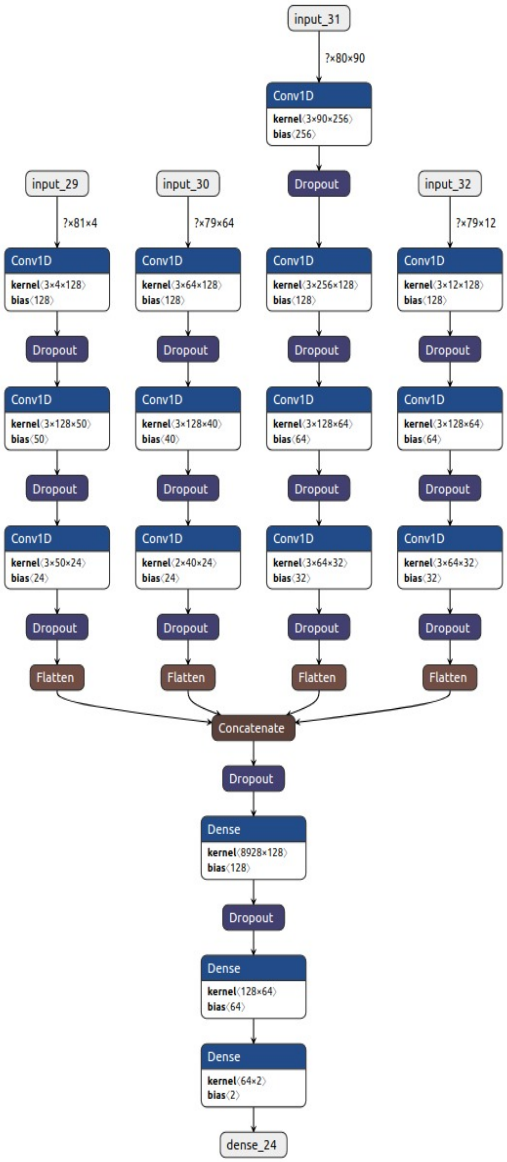

Figure 5: Original Model

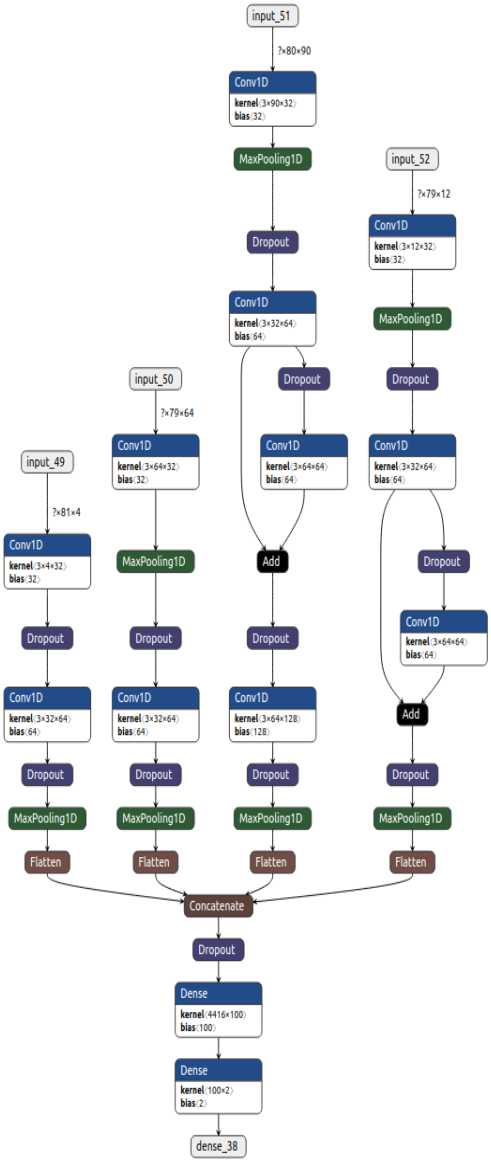

Figure 6: Our Model

References

[1] Ruhul Amin, Chowdhury Rafeed Rahman, Sajid Ahmed, Md Habibur Rahman Sifat, Md Nazmul Khan Liton, Md Moshiur Rahman, Md Zahid Hossain Khan, and Swakkhar Shatabda. iPromoter-BnCNN: a novel branched CNN-based predictor for identifying and classifying sigma promoters. *Bioinformatics*, 07 2020. btaa609.

[2] Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby. An introduction to deep learning on biological sequence data: examples and solutions. *Bioinformatics (Oxford, England)*, 33(22):3685–3690, Nov 2017. 28961695[pmid].

[3] Haoyang Li, Shuye Tian, Yu Li, Qiming Fang, Renbo Tan, Yijie Pan, Chao Huang, Ying Xu, and Xin Gao. Modern deep learning in bioinformatics. *Journal of Molecular Cell Biology*, 06 2020. mjaa030.

[4] Senior Andrew W. et. al. Improved protein structure prediction using potentials from deep learning. *Nature*, 2020.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Bin Liu, Fan Yang, De-Shuang Huang, and Kuo-Chen Chou. iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformatics*, 34(1):33–40, 09 2017.

[7] Meng Zhang, Fuyi Li, Tatiana T Marquez-Lago, André Leier, Cunshuo Fan, Chee Keong Kwoh, Kuo-Chen Chou, Jiangning Song, and Cangzhi Jia. MULTiPly: a novel multi-layer predictor for discovering general and specific types of promoters. *Bioinformatics*, 35(17):2957–2965, 01 2019.

[8] Qianzhong Li and Hao Lin. The recognition and prediction of σ70 promoters in escherichia coli k-12. *Journal of theoretical biology*, 242:135–41, 10 2006.

[9] Hao Lin, En-Ze Deng, Hui Ding, Wei Chen, and Kuo-Chen Chou. iPro54-PseKNC: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition. *Nucleic Acids Research*, 42(21):12961–12972, 10 2014.

[10] Scheila Silva, Franciele Forte, Ivaine Sartor, Tahila Andrighetti, Gunther Gerhardt, Ana Paula Delamare, and Sergio Echeverrigaray. Dna duplex stability as discriminative characteristic for escherichia coli σ^{54}- and σ^{28}- dependent promoter sequences. *Biologicals : journal of the International Association of Biological Standardization*, 42, 10 2013.

[11] Ilham Ayub Shahmuradov, Rozaimi Mohamad Razali, Salim Bougouffa, Aleksandar Radovanovic, and Vladimir B Bajic. bTSSfinder: a novel tool for the prediction of promoters in cyanobacteria and Escherichia coli. *Bioinformatics*, 33(3):334–340, 09 2016.

[12] Kai Song. Recognition of prokaryotic promoters based on a novel variable-window z-curve method. *Nucleic Acids Research*, 40:963–971, 11 2012.

[13] Sotiris Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36, 11 2005.

[14] Mahendra Sahare and Hitesh Gupta. A review of multi-class classification for imbalanced data. *International Journal of Advanced Computer Research*, 2:160–164, 09 2012.