DI 504 Course Project: Predicting the Sequence Specificity of DNA-binding Proteins by a CNN

Hüseyin Hilmi Kılınç
Department of Health Informatics
Graduate School Of Informatics
Middle East Technical University
Ankara, Türkiye
hilmi.kilinc@metu.edu.tr

I. Introduction

Transcription factors (TFs) are DNA-binding proteins that regulate gene expression by binding to specific DNA sequences (known as motifs or TF binding sites) [1]. Identifying these binding sites from DNA sequences is a key problem in regulatory genomics, important for understanding gene regulation and disease mechanisms. High-throughput experiments like ChIP-seq (Chromatin Immunoprecipitation followed by sequencing) allow genome-wide detection of TF binding in vivo by providing sequences (called "peaks") where a TF is bound in living cells. However, predicting TF binding directly from sequence is challenging because the sequence patterns are short and embedded in a noisy genomic context. Traditional motif-finding tools use position weight matrices (PWMs) or k-mer enrichment to represent and scan for motifs, but they have limited accuracy on complex genomic data.

In recent years, deep learning, particularly convolutional neural networks (CNNs), have dramatically improved TF binding site prediction by automatically learning motif features from raw sequences [2]. One early and influential model is DeepBind [3], which introduced a CNN-based approach to classify DNA/ RNA sequences by their binding affinity to a protein. DeepBind demonstrated that an end-to-end deep learning model can discover sequence motifs de novo and outperform previous motif-based methods. In Figure 1, we can see this adaptation that involves representing a genomic sequence window as a one-dimensional input with four channels corresponding to nucleotide bases (A, C, G, T), analogous to the three-channel (R, G, B) representation of 2D images. Consequently, modeling DNA sequences becomes conceptually similar to binary image classification. This formulation enables the detection of sequence motifs irrespective of their position within the window, aligning well with the objectives of motif discovery and binding site classification. In this project, I propose to implement a PyTorch-based reimplementation of the DeepBind model and apply it to TF ChIP-seq data. My goal is to replicate DeepBind's architecture and evaluate its performance on predicting TF binding sites, contributing variations such as reverse-complement data, augmentation, and comparing different motif (kernel) lengths. The following report details the problem background, related work, data preparation, model architecture, experimental results, and conclusions from the DeepBind replication study.

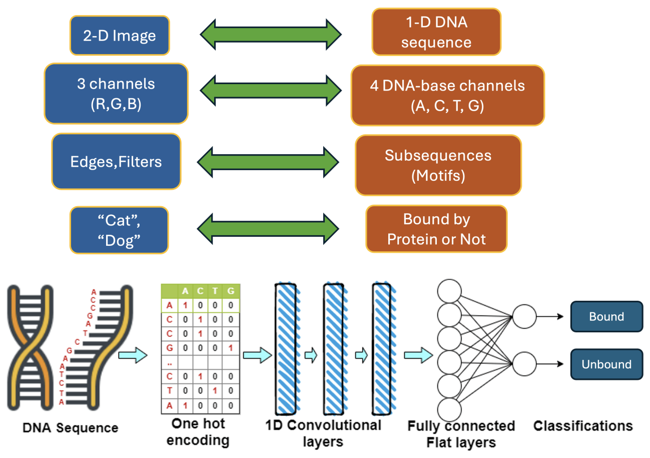

Fig. 1. The Adaptation of CNN from Computer Vision to Genomics

II. Literature Review

DeepBind presented by Alipanahi and his colleagues was one of the first deep learning models for sequence-based protein binding prediction [3]. It uses a single-layer CNN to scan input DNA/RNA sequences with motif-length convolutional filters, followed by a global max-pooling layer that captures the strongest motif match anywhere in the sequence. This pooled motif signal is fed into a fully connected network to output a binding score. By training on large experimental datasets, DeepBind learns PWMs-like (motif) filters automatically, detecting motifs that influence binding. Notably, DeepBind achieved higher accuracy (e.g. AUROC) than prevailing methods like PWMs or k-mer based classifiers on both in vitro protein-binding microarray (PBM) data and in vivo ChIP-seq data. DeepBind also introduced useful interpretation tools:

its learned convolution filters can be visualized as sequence logos (akin to motifs), and it can generate “mutation maps” to highlight how nucleotide changes affect binding affinity. This combination of improved accuracy and interpretability made DeepBind a seminal work in computational genomics.

Beyond DeepBind, several models have built upon its architecture to improve the accuracy and interpretability of transcription factor binding site prediction. One such model is FactorNet, which extended DeepBind by combining convolutional neural networks with long short-term memory (LSTM) networks [4]. This hybrid architecture enabled the model to capture both local motif structures and longer-range sequential dependencies. Additionally, FactorNet incorporated DNase-seq accessibility data as input, allowing it to contextualize binding predictions based on chromatin openness, thus improving performance in cell type–specific settings.

Similarly, DeepGRN by Chen and his colleagues introduced an attention-based framework for TF binding site prediction that integrated convolutional layers, bidirectional LSTM units, and dual attention modules [5]. This design allowed the model not only to learn motif-level features but also to dynamically focus on the most relevant regions of input sequences. DeepGRN demonstrated high predictive accuracy across multiple ChIP-seq and DNase-seq datasets and emphasized the interpretability of learned regulatory signals, which is crucial for understanding gene regulation in complex cellular contexts. In summary, DeepBind established the paradigm of CNNs for motif discovery and TF binding prediction, and subsequent research has generally confirmed and extended its approach. Key contributions from the literature include: (*i*) validating that global max pooling provides position invariance for motif detection ,(*ii*) demonstrating that the choice of filter length and number is crucial for performance, and (*iii*) incorporating recurrent or attention mechanisms to capture interactions between motif sites. In this project, I reimplementation focuses on the core DeepBind model, while acknowledging these advances. We also draw on insights like reverse-complement equivalence of DNA sequences and proper negative set construction to ensure a robust experimental setup.

III. Dataset

In this work, I evaluated the DeepBind model on a ChIP-seq dataset for a specific transcription factor (TF), derived from the ENCODE database. The positive examples consisted of genomic DNA sequences corresponding to high-confidence ChIP-seq peaks for the TF – these are regions experimentally found to be bound by the TF in vivo. Each positive sequence was taken around the summit of a ChIP-seq peak (the point of strongest TF-DNA crosslinking), and extended or trimmed all sequences to a fixed length window (in our case, a length such as 101 bp) centered on the summit. Using a fixed input length (e.g. 101 bp) simplifies model training, and sequences shorter than this were padded (with N’s, which are encoded as neutral zero vectors) while longer sequences were truncated to the desired length. This yields a consistent input matrix size (length $\times$ 4 channels) for the CNN. The label for each sequence is binary: 1 for an actual TF-bound sequence (experimental peak) and 0 for a negative sequence.

Constructing an appropriate set of negative examples is critical. Rather than using arbitrary genomic background sequences, I employed dinucleotide shuffling to generate negatives. For each positive sequence, I created a random sequence with the same length, overall nucleotide composition, and dinucleotide frequency, but scrambled order. This preserves low-level sequence characteristics (e.g. GC content and neighboring base frequencies) while disrupting any real motif instances. Such dinucleotide-preserved shuffles are a common strategy to make a fair negative class that is not trivially different from positives [6]. The model thus learns to detect specific motifs rather than simple compositional biases. I generated an equal number of shuffled negatives to match the number of positive peaks, resulting in a balanced dataset (50% bound, 50% unbound).

All DNA sequences (positive and negative) were then one-hot encoded into a binary matrix representation. In one-hot encoding, each nucleotide is represented by a 4-dimensional binary vector – e.g. A = (1,0,0,0), C = (0,1,0,0), G = (0,0,1,0), T = (0,0,0,1). An input sequence of length L becomes an L $\times$ 4 matrix, which can be viewed as 4 parallel “channels” (analogous to color channels in an image) indicating the presence of each base at each position. This encoding allows the convolutional network to scan for patterns across the sequence in a translationally invariant manner. I did not include any extra genomic features (such as chromatin accessibility or conservation) in this project. Thus, the model relies solely on the raw nucleotide sequence.

To augment the training data and account for DNA strand complementarity, we utilized reverse- complement augmentation. Each DNA sequence has a reverse-complement (RC) sequence representing the opposite strand (e.g. 5’-ACGT-3’ vs 5’-CGTA-3’). TF binding is typically independent of the strand (unless the TF has a directional preference), so an ideal model should score a sequence and its RC equally. I enforced this by including both versions of each sequence in training: for every sequence in the training fold, I added its reverse-complement as an additional sample with the same label. This effectively doubled the training set size and exposed the model to motifs in both orientations. As a result, the learned convolutional filters become capable of detecting motifs on either strand. (In my implementation, I did not tie the weights for RC pairs; I simply treated RCs as separate training examples. One could also enforce reverse-complement equivariance by constraining filters or averaging predictions, but explicit augmentation was sufficient for this purpose. We found that RC augmentation improved performance and helped the model generalize, as discussed later.

In summary, the data preprocessing pipeline was:

- **Peak sequence retrieval:** Obtain positive TF-binding sequences from ChIP-seq peak regions (fixed length around peak center).
- **Negative generation:** For each positive, create a dinucleotide-shuffled sequence as a negative example.

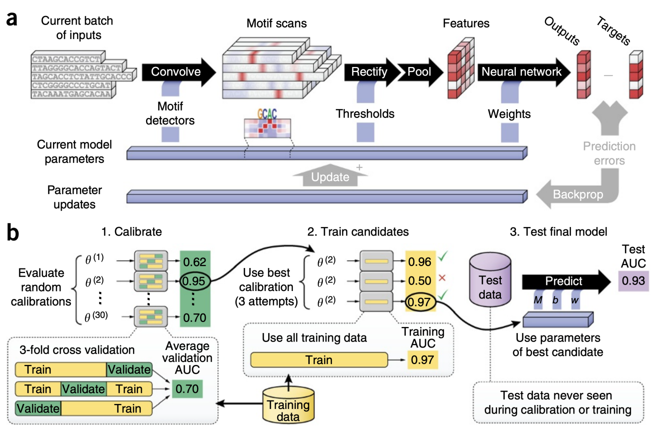

Fig. 2. DeepBind Model Architecture [3]. The overall workflow involves three main stages: calibration, where model hyperparameters are tuned; training, where the model learns from labeled data; and testing, where performance is evaluated on independent datasets to assess generalizability.

- **One-hot encoding:** Convert each DNA sequence (positive or negative) into a 4-channel binary matrix for CNN input.
- **Padding/trimming:** Pad sequences with N (zero vector) or trim as needed so that all inputs are of uniform length (e.g., 101+24+24 bp).
- **Augmentation:** If in reverse-complement mode, add the reverse-complement of each training sequence to the dataset to augment and enforce strand-invariant learning.

After preprocessing, we split the dataset for evaluation. We employed a stratified 3-fold cross- validation approach: the data was divided into three folds of roughly equal size, each containing a balanced mix of positives and negatives. I then performed three training runs, each time using two folds (67%) for training and the remaining fold (33%) for testing. This allows us to assess model performance on all data while guarding against overfitting to any particular subset. A portion of training in each fold as a validation set for tuning hyperparameters and early stopping were also reserved. Cross-validation provides a more robust estimate of performance given used dataset's limited size.

IV. Method

My model closely follows the original DeepBind architecture in Figure 2, but is implemented in PyTorch for flexibility. The network is a simple 1D convolutional neural network for binary sequence classification. Figure 1 outlines the architecture and data flow (from encoded sequence to binding prediction). The core components of the model are:

A. Convolutional Layer

The first layer performs a 1D convolution over the input sequence matrix to detect motif features. We use multiple convolutional filters (also called kernels), each of which is a small weight matrix that slides over the length of the sequence. Each filter has a width equal to a chosen motif length hyperparameter (e.g. 24 bp) and spans all 4 input channels. In our default configuration, I used 16 filters of length 24, meaning the model can learn up to 16 distinct motif patterns. The convolution is applied with stride 1 across the sequence, producing an output feature map for each filter. Conceptually, as the filter scans the sequence, it computes a score at each position indicating how well that subsequence matches the filter's motif pattern. I then applied zero-padding at the sequence ends so that the convolution covers motifs at the edges and the output length remains the same as input length. After convolution, a bias term is added to each filter's output.

B. Nonlinear activation

A ReLU (Rectified Linear Unit) activation to the convolution outputs were applied. ReLU sets negative values to 0 and keeps positive values, introducing nonlinearity and filtering out weak matches. This yields rectified feature maps where each position has a value indicating the presence strength of a motif instance (if the value is ¿0) or no significant match (value = 0). The use of ReLU is consistent with DeepBind and helps the model focus on strong motif hits.

C. Pooling Layer

A global max-pooling operation is applied to each feature map. This reduces each filter's sequence of activations to a single maximum value—the highest activation across all positions in the sequence for that filter. Max-pooling captures the most prominent motif occurrence for each filter, regardless of its position. This is an important design choice because it ensures that the model's prediction depends only on the strongest binding site in the sequence, rather than the number or exact position of sites. If a filter detects a transcription factor's motif, the pooling layer retains the highest score for the best match, converting the variable-length activation map into a fixed-length vector. While other pooling strategies (like averaging or local pooling) were tested, global max-pooling performed best. It also provides translational invariance, meaning a motif can appear anywhere in the sequence and still be effectively recognized.

D. Fully connected layer

The pooled outputs from all filters (forming a vector with length equal to the number of filters) are passed into a fully connected dense layer. In the simplest version of the model, a single output neuron with a sigmoid activation function predicts the probability of transcription factor (TF) binding. This produces a score between 0 and 1 for each input sequence. The weights connecting each filter's pooled activation to the output can be interpreted as the learned importance of each motif feature. In one variation, I experimented with adding a hidden layer—an additional dense layer with ReLU activation—between the pooling and output layers. This was intended to help the model learn combinations of motifs (e.g., cases where two motifs together increase binding confidence). However, this hidden layer did not noticeably improve performance, likely because the task could be sufficiently handled by a linear combination of motif signals and the dataset was not large enough to benefit from extra parameters. Therefore, the final model was kept as a one-layer CNN feeding directly into the output neuron, similar to the original DeepBind structure.

E. Regularization

To prevent overfitting, we applied several regularization techniques. Dropout was used on the fully connected layer (with a dropout rate of 0.5), randomly deactivating half of the units during training. This helps the model generalize better by reducing reliance on any single filter's signal. Additionally, we applied L2 weight decay to the convolutional weights and biases as a form of weight penalty. Early stopping based on validation loss was also used to prevent overtraining. These strategies were especially important given the relatively small size of our dataset.

The model was trained using binary cross-entropy loss and the SGD optimizer with a learning rate of 0.001. Each model was trained on mini-batches of 64 sequences. I monitored AUROC and AUPRC on the validation set, and the best-performing model per fold was saved for evaluation. Some hyperparameters were tuned empirically. I focused particularly on the filter length (motif length) as a key parameter. Alongside the default length of 24 base pairs, I tested shorter filters (e.g., 12 bp) and longer filters (e.g., 36 bp) to observe their effect on model performance. Due to limited time and computational resources, I performed a manual random grid search. To improve generalization, we augmented training data with reverse-complement sequences. The model architecture remained unchanged; it simply learned from both strands. For the reimplementation, DeepBind was originally developed using an older version of Theano, a now-deprecated deep learning framework. Due to the lack of official support and outdated dependencies, I relied on two unofficial GitHub implementations (provided in the codebase) as references. I carefully examined their structure and adapted the code to replicate the core architecture and training procedures of DeepBind, making necessary modifications to ensure compatibility with modern frameworks and reproducibility of results.

V. Results

To evaluate how different model configurations affect transcription factor binding site prediction, I experimented with varying the number of motifs (filters) and the motif lengths (kernel sizes) used in the convolutional layer. Specifically, I tested combinations where the motif length was set to 12, 24, or 36 base pairs, and the number of motifs was either 8 or 16. The baseline configuration, inspired by the original DeepBind architecture, used 16 filters of length 24.

The ROC and PRC curves in Figures 3 and 4, respectively summarize model performance across different settings. The results showed that the configuration using 8 filters with 24 bp length failed to learn meaningful patterns, as indicated by its flat ROC curve with an AUROC of 0.500 and a relatively lower AUPRC of 0.750. This result may suggest issues such as insufficient representational capacity or unstable convergence with this particular setting.

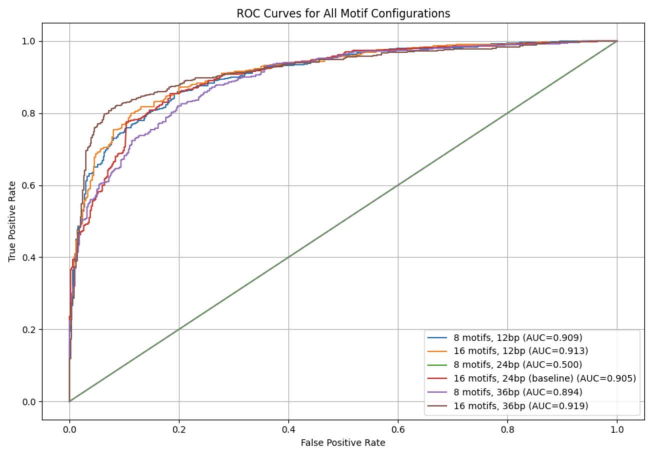

Fig. 3. Each curve represents the ROC performance of a DeepBind model trained with a different combination of number of motif filters (8, 16) and motif lengths (12, 24, or 36 bp). The baseline model (16 motifs, 24 bp) is highlighted for comparison. AUC values are provided in parentheses for each configuration.

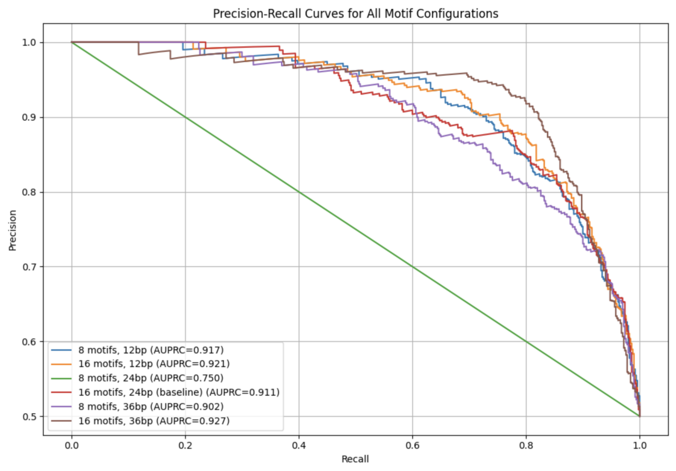

Fig. 4. Each curve shows the Precision-Recall performance of models with varying motif filter counts and motif lengths. The baseline configuration (16 filters, 24 bp motif) is annotated for reference. AUPRC values in parentheses indicate precision-recall tradeoff quality for each setting.

In contrast, other configurations performed quite well. Models using 12 bp and 36 bp filters, especially with 16 motifs, achieved higher predictive performance. For instance, the model with 16 motifs and 36 bp filters reached an AUROC of 0.919 and AUPRC of 0.927—the highest among all settings tested. Increasing the number of filters from 8 to 16 consistently improved both ROC and PR performance across all motif lengths. This supports the idea that having more convolutional filters enables the model to capture a richer variety of motif patterns, improving generalization.

The baseline configuration (16 motifs, 24 bp) achieved strong overall performance with AUROC and AUPRC values of 0.905 and 0.911, respectively. However, the configuration with longer filters (36 bp) slightly outperformed it, suggesting that capturing longer or composite motifs may be beneficial for certain TFs, especially in ChIP-seq datasets where binding contexts span broader regions.

In summary, the results demonstrate that both the length and number of motifs are crucial hyperparameters in CNN-based TF binding models. While the baseline setup is solid, increasing filter size and number—particularly to 36 bp and 16 motifs—can provide performance gains without compromising model stability.

VI. Conclusion and Discussion

In this project, I tried to reimplement the DeepBind deep learning model and applied it to single transcription factor ChIP-seq data, effectively predicting TF binding sites from DNA sequence. The PyTorch-based model—comprising a single convolutional layer with motif-length filters, global max pooling, and a sigmoid output—was able to learn the relevant DNA motifs and achieved high accuracy (AUROC 0.9) in distinguishing bound from unbound sequences. Through both literature review and experimental work, I identified several key factors that contributed to the model's performance; Filter (motif) length experiments showed that using a filter length around the known TF motif size yields the best results, whereas significantly shorter or longer filters can diminish performance. This emphasizes the need to choose model parameters mindful of the underlying biology. I also want to note that handling real genomic background (instead of shuffled sequences) and addressing class imbalance in genome-wide prediction would be necessary for deploying such a model in a genome scanning scenario. Nonetheless, within the scope of this project, I tried to demonstrate that a PyTorch-based DeepBind model can be trained on ChIP-seq data to accurately predict TF binding sites and recover known motif patterns. This exercise provided hands-on experience with deep learning in a bioinformatics context and validated the continued relevance of the DeepBind approach in the era of modern genomics.

Overall, the DeepBind framework remains a valuable and foundational method for motif-based sequence prediction problems. Its reimplementation not only confirms past results but also sets the stage for incremental improvements. By building on a solid baseline and incorporating new ideas (like different motif parameters and appropriate architecture tweaks), one can develop even more powerful models for deciphering the regulatory code of DNA. The knowledge gained here will be useful for future projects aiming to interpret genome sequences with deep learning, as we have seen how crucial design decisions.

References

[1] Z. Dai, D. Guo, X. Dai et al., ”Genome-wide analysis of transcription factor binding sites and their characteristic DNA structures,” BMC Genomics, vol. 16, Suppl. 3, p. S8, 2015.

[2] S. Park, Y. Koh, H. Jeon et al., ”Enhancing the interpretability of transcription factor binding site prediction using attention mechanism,” Scientific Reports, vol. 10, no. 13413, 2020.

[3] B. Alipanahi, A. Delong, M. Weirauch et al., ”Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning,” Nature Biotechnology, vol. 33, pp. 831–838, 2015.

[4] D. Quang and X. Xie, ”FactorNet: A deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data,” Methods, vol. 166, pp. 40–47, Aug. 2019.

[5] C. Chen, J. Hou, X. Shi et al., ”DeepGRN: prediction of transcription factor binding site across cell-types using attention-based deep neural networks,” BMC Bioinformatics, vol. 22, p. 38, 2021.

[6] J. Lanchantin, R. Singh, B. Wang, and Y. Qi, ”DEEP MOTIF DASHBOARD: Visualizing and understanding genomic sequences using deep neural networks,” Pacific Symposium on Biocomputing, vol. 22, pp. 254–265, 2017.