

Promoter Identification from Genomic Sequence through Deep Neural Networks

Xuan Zhang
A53106624
xuz043@ucsd.edu

Qi Ma
A53263366
qima@eng.ucsd.edu

Zhuo Yue
A53275690
zyue@eng.ucsd.edu

Wenmiao Yu
A53285474
wey003@eng.ucsd.edu

Abstract—Promoter sequences are the main regulatory elements of gene expression. Their recognition is fundamental for understanding gene expression patterns. To better understand the sequence-level information encoded, we developed deep learning models to predict the promoter region based on sequence-based features only. We used the experiment data of *homo sapiens* to train and test our model. The best model we have built has accuracy 0.9197, recall 0.92, precision 0.92, and F1-score 0.91.

Index Terms—promoter, gene, one-hot-encoding, CNN, prediction

I. INTRODUCTION

Cells are the basic building blocks of living things. The human body is composed of trillions of cells, all with their own specialized function. Each of our cells contains 46 strands of DNA. A single strand of DNA is made of millions of particles called nucleotides. These nucleotides come in 4 different types which labeled as "A, T, C, G" (Fig. 1). A gene is a special stretch of DNA sequence of ATCG that codes for something. Genes contains information for a cell to read and use.

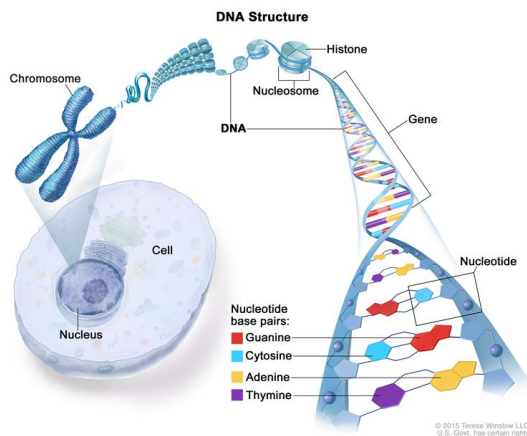


Fig. 1. DNA Structure

Genes make proteins and those proteins then interact with each other and all sorts of chemicals within the body to build things like freckles, eye pigments and some hormones (Fig. 2). A DNA molecule contains thousands of genes or unique protein recipes.

To achieve specific function, our cells need to synthesis many kinds of proteins according to the recipes provided by the gene. Central dogma (Fig. 3) describes the two steps from

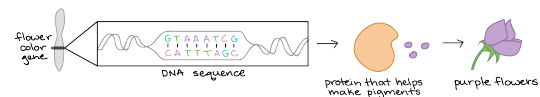


Fig. 2. Genes are the recipe of the functional protein

gene recipes to the functional product. First DNA will be transcribed into mRNA which is a copy of a segment of DNA. Then the mRNA will be translated into functional proteins.

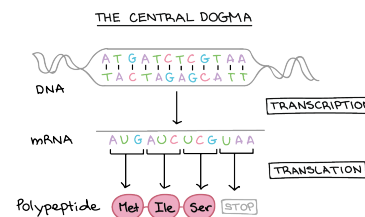


Fig. 3. Central Dogma

However, each DNA molecule contains thousands of genes or unique protein recipes. How does the cell know where is the start site of the gene?

It turns out, ahead of each gene there is a region called promoter (Fig. 4) that can recruit some transcription factors to initiates the transcription of a particular gene.

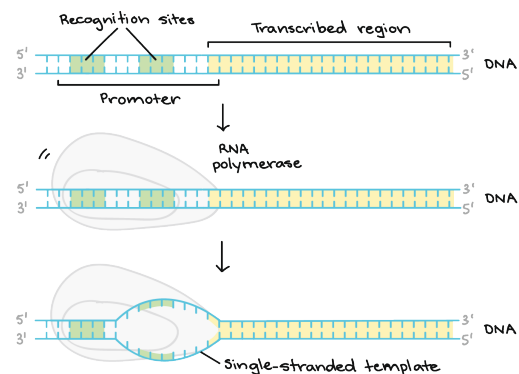


Fig. 4. Promoter of a gene

As promoters are typically immediately adjacent to the gene in question, positions in the promoter are designated

relative to the transcriptional start site, where transcription of DNA begins for a particular gene (i.e., positions upstream are negative numbers counting back from -1, for example -100 is a position 100 base pairs upstream). Eukaryotic promoter regulatory sequences typically bind proteins called transcription factors that are involved in the formation of the transcriptional complex.

In eukaryotes, at least seven different factors are necessary for the binding of an RNA polymerase II to the promoter. Promoters represent critical elements that can work in concert with other regulatory regions (enhancers, silencers, boundary elements/insulators) to direct the level of transcription of a given gene.

Promoters can be recognized from high-throughput experimental approaches genome-wide. Therefore, we use the database of promoter identified experimentally.

The **input** to our algorithm is the labeled sequence of promoter and non-promoter. We then use a neural network to **output** a predicted result that whether the provided sequence is a promoter or non-promoter.

II. RELATED WORK

Past studies have explored some promoter identification tools. TSSW [5] used an algorithm based on a linear discriminant function (LDF) to discriminate real poly-A signal regions. Solovyev and Shahmuradov then created PromH [6] which extended the TSSW program feature set and used linear discriminant functions that took into account conservation features and nucleotide sequences of promoter regions in pairs of orthologous genes. Promoter2.0 [4] used neural networks[1] and genetic algorithms [3] to recognize promoters from DNA sequence. PromCNN [7] utilized Convolutional Neural Networks (CNN) to grasp promoter sequence characteristics of five distant organisms and achieved significantly higher accuracy compared to previously developed promoter prediction programs.

III. DATASET ACQUISITION

A. EPD database

Biologist have identified the TSS region of genes. According to the TSS region, they could infer the promoter sequence, such promoter sequence can be obtained from EPD (Eukaryotic Promoter Database). From this database, we could download the fasta format which contains the promoter ID and promoter sequence information. In this work, we downloaded fasta file from EPD database, which is an annotated non-redundant collection of eukaryotic Pol II promoters, for which the TSS has been determined experimentally. And employ *SeqIO* to tackle the inputs and outputs.

B. Dataset Quality Control

Since we have focused our work on human gene promoter detection, we only need to query *Homo sapiens* (human) curated promoter database. However, there are still many other factors to determine in the data collection part. The first factor we need to take into consideration is the way to

select the positive and negative promoters. Since promoters tend to appear in the start sites in a gene sequence, it is highly possible that the negative/non-promoter is located far from the start sites. However, it does not mean that the further location we choose as non-promoter, the prediction accuracy is higher, since it might still overlaps with the next sequence, thus leading to high possibility that we choose the promoter of the next sequence as non-promoter. Therefore, based on the discussion above, we need to decide the start sites for both positive promoter, and negative promoter, the best length as inputs, then get the best combinations of positive and negative promoters. Besides that, we also find TATA-box will influence the final result. TATA-box is one of the core promoter element, which is normally found 28 bp upstream the transcription start site. The following plot shows that EPDnew promoter collection has a more focused TATA-box distribution compared to the Gencode annotation, suggesting a precise TSS mapping in EPDnew. In the actual experiment, we found the gene sequences with TATA-box performs worse in accuracy than non-TATA dataset. To achieve better accuracy, we are using the human, non-TATA dataset.

C. Sequences Trim

Sometimes the given data have different lengths for promoters and non-promoters, which caused problems if we want to feed the Numpy arrays as the input for the network. We'll start by trimming the sequence to the same length (200 nucleotides) using the *Biopython* package.

IV. METHODS

A. Input Data Manipulation

The data in EPD database is given in the fasta format, which cannot be identified in our networks. In our project, we used two different encoding approach. One is transforming the "A,G,C,T" sequence using integer-encoding. That is, we used [1, 2, 3, 4] to represent [A, C, G, T]. Another method is one-hot encoding, which is a representation of categorical variables as binary vectors. We used [1,0,0,0], [0,1,0,0], [0,0,1,0] and [0,0,0,1] to represent [A,C,G,T] respectively.

B. Convolutional Neural Networks

Convolution Neural Networks are similar to the Artificial Neural Networks but different with the essentially neural networks that use convolution in place of general matrix multiplications at least for the first layers. CNN is a sequence of layers and consists of convolutional layers, pooling layers and fully-connected Layer, which is designed to perform multidimensional data.[2] (eg. 2D image) Non-fully connected hidden layers make the forward function more efficient to build the model and reduce the number of learnable parameters in CNNs.

• Convolutional Layer

The convolution layer is the core for building a CNN model, which does most of the computational work. The CONV layers parameters consist of a set of learnable filters which are small matrices with size $W*L*D$, where

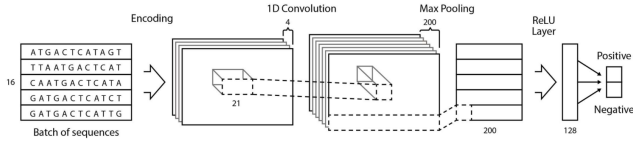


Fig. 5. CNN sequence architecture

w is the vertical width of the matrix, L is the horizontal width of the matrix, and D is the depth of input data, i.e. the number of channels. Each filter is spatially and extends through the full depth of the input data. During the forward pass, we slide each filter across the width and height of the input data and compute dot products between the entries of the filter and the input at any position.[2]. Convolution example for a $32 \times 32 \times 3$ image is shown as the Fig.6.

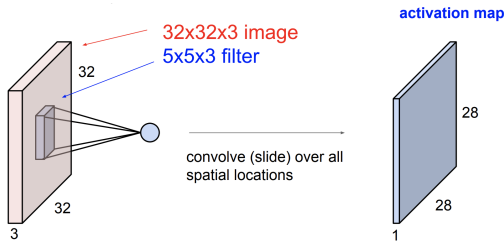


Fig. 6. Convolution filter for 2D image.[2]

The distance of each step of a filter sliding across the input data is called a stride. When a filters slides across a data, we can extract a feature map from the data. Feature map gives the responses of that filter at every spatial position. To extract multiple features, we use a set of filters and the number of filters is the number of feature maps and the depth of input data for next layer.

- **Max Pooling**

Feature maps obtained by the convolution layer will be fed into the max pooling layer to reduce the size of features and the number of parameters, and also help control over-fitting. The most common form is a max pooling layer with filters of size 2×2 and stride 2. For max pooling, we slide each filter across the width and height of the input data and keep the maximum value over 4 numbers (in the 2×2 region). The depth dimension will be preserved during max pooling.

- **Fully Connected Layer**

After the max pooling layer, CNN has a fully connected layer in which neurons have full connections to all activations in the previous layer. In this layer, we first stretch the input matrix into a vector, i.e. only one row matrix, and then compute the dot product between the vector and a row of weight matrix.

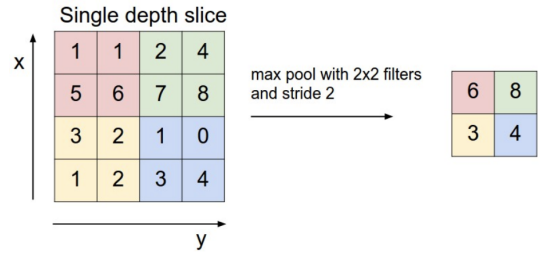


Fig. 7. Max pooling with filter 2×2 [2]

C. CNN Architecture Modeling

- **Framework**

In this work, we use Keras as the dominant framework. The models, layers, etc. methods help us to build neural networks efficiently.

- **Optimizer**

The output layer from the CNN are two neurons. For each neuron, it applies the **Softmax function** and assigns all the input into a probability between 0 and 1. For binary classification tasks like this one, the two probabilities will sum up to 1. When building CNN architecture, Keras provides us with a lot of optimizers for backpropagation to reduce total loss including SGD, RMSprop, Adam, L-BFGS, etc. Among those, we finally choose Adam due to its easy implementation and relatively smaller loss in the end. And also, Adam optimizer is suitable for this categorical problem, training with categorical cross-entropy as a loss function.

- **Early Stopping**

The goal for early stopping is reducing over-fitting and training process will stop the iteration of gradient descent before convergence. When we train our CNN model, we first split the training set into a training subset and a validation subset and train the CNN model only on the training subset. To implement early stopping, we will evaluate the loss on the validation subset every few epochs and stop training as soon as the loss on the validation starts increasing.

- **Dropout**

As what we have discussed in the former section, Convolutional Neural Networks (CNNs) can help analyze sequence characteristics of human promoters and build their predictive models. Actually, the shallow CNN model is able to classify the positive and negative gene sequences very well, but in the training process, we encountered overfitting, and to make the original model fancier, we can also add Dropout layers as a regularization technique to resolve overfitting by preventing complex co-adaptations on training data.

D. Initial model setup

We initialize our CNN model using 200 filters with size of $1 \times 21 \times 4$ and implement only one convolutional layer and then

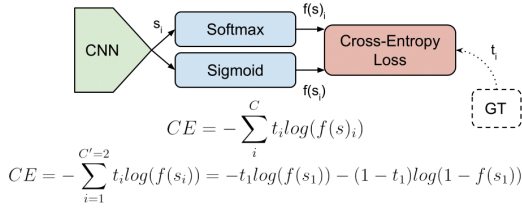


Fig. 8. The cross-entropy loss function, where t_i and s_i are the ground truth and the CNN score for each class i in C . As usually an activation function (Sigmoid/Softmax) is applied to the scores before the CE loss computation, we write $f(s_i)$ to refer to the activations.

max pooling layer followed by a fully connected layer. We also apply ReLu after the max pooling layer as rectifier to increase non-linearity and finally implement softmax to maintain the category prediction result.

The output in the last layer from the CNN architecture is a non-integer probability of how likely a sequence is to be a promoter. To make some other accuracy assessments later, we also make an array that rounds the predicted probabilities. This produces the binary labels we want: a label that will denote "yes" (1) or "no" (0).

Then we can create a confusion matrix and collect the True Positive, True Negative, False Positive, and False Negative.

V. EXPERIMENTS AND RESULTS

A. Selecting length and location of input

We need to choose what part of a promoter region to feed into our model for training. The database can tell us the exact location of the TSS (transcription start site) of each gene. Positions in the promoter are designated relative to the transcriptional start site. Biologists estimate the length of the promoter which is around 100 to 200 base pair.

Therefore, we decide to test different length and different location of our positive and negative data.

We first test the how different location influence the prediction performance. We fixed the input length to 200 base pair. For the positive data, we are sure it is upstream of the TSS region. As we mentioned before, positions upstream are negative numbers counting back from -1, for example -100 is a position 100 base pairs upstream. We treat the TSS as position 0. We tried 4 sets of positive regions which are (-400, -200), (-300, -100), (-200, 0) and (-100, 100). We picked 3 sets of negative regions which is the downstream region of TSS. They are (500, 700), (800, 1000), and (1000, 1200).

As shown in 10, after training on every combination of these positive and negative dataset. We found that the best result is obtained from the positive (-100, 100) and negative (800,1000). That is to say, the most concentrate promoter features are located around (-100, 100) relative to TSS region. We also realized that, for each negative dataset, when the promoter is further to the TSS region, the performance decreased. This finding makes perfect sense to the real case. Biologically, once the transcription factor binds to the promoter region, these factors can facilitate the recruitment of RNA pol II which is



Fig. 9. Model accuracy with different input Position

the enzyme to initiate transcription from TSS. Therefore, the promoter should locate very close to TSS region spatially.

To get the best training result, negative dataset is equally important to the positive dataset to separate the two population better. From our experiment, the best negative region is (800,1000) to the TSS. Apparently, 500 to 700 might be too close to the TSS which might also contain some other features to confuse the identification of promoter. Compare to (1000, 1200), (800, 1000) is a little bit better. Both of them are far from the promoter region to lead good prediction. The slightly decrease of performance of (1000, 1200) might be caused by the average gene length of human. Genes are distributed along the chromosomes and different gene has different length. The downstream of (1000, 1200) might be the beginning of the next gene. For this reason, our negative dataset might be contaminated by the promoter of other genes. This could be the bottleneck to improve the performance of our training.

Finally, we decide to use (-100, 100) as positive input and (800, 1000) as negative input. With such fixed region center, we try to adjust different input length to test the model accuracy. The result is shown in 10, the best prediction can be achieved by the input around 200 base pair.

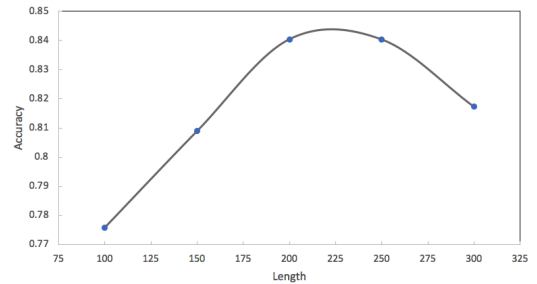


Fig. 10. Model accuracy with different input length

B. Performance optimization

With the model set up, we can start training the model using positive and negative sets which consist of relatively short sequences with a fixed length. We split the dataset into

three parts, 60% for training, 20% for validation to tune the hyperparameters, the rest 20% as testing.

We take the training and validation for experiment, and plot the accuracy and loss respectively. The plot below is the accuracy and loss trend for training and validation with the increment of iterations.

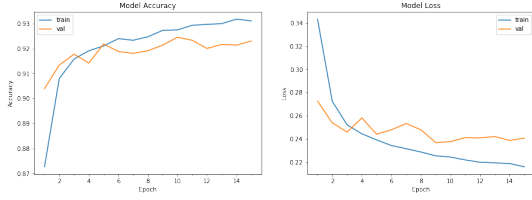


Fig. 11. Plots of <https://www.overleaf.com/9216818841wjfrntfnpkn> accuracy and loss.

We first set the 200 filters for CNN, however, we encounter overfitting problem in the training dataset. After reducing the number of filters to 8 and implementing early stopping, we finally got the evolution of training and validation accuracy at around 92%. The architecture and parameters of our final CNN model is shown as Fig.12. We can figure out that the trends of the accuracy and loss of validation set are similar to those of training set.

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 181, 8)	680
max_pooling1d_1 (MaxPooling1D)	(None, 1, 8)	0
flatten_1 (Flatten)	(None, 8)	0
dense_1 (Dense)	(None, 128)	1152
dense_2 (Dense)	(None, 2)	258
Total params: 2,090		
Trainable params: 2,090		
Non-trainable params: 0		

Fig. 12. Architecture and parameters of our CNN model.

C. Evaluation

Once we built up the model, the most important task in is evaluating the model. There are several metrics that we need to pay attention to.

Accuracy

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. The formula below is the way to calculate accuracy once we know the distribution of FP, FN, TP, TN.

$$Accuracy = (TP + TN)/(TP + FP + FN + TN)$$

Precision

Accuracy is a great measure but only when you have symmetric datasets where values of FP and FN are almost same. Therefore, we have to look at other parameters to evaluate the performance of the model. In binary classification, precision is the fraction of relevant instances among the retrieved instances.

$$Precision = TP/(TP + FP)$$

Recall/Sensitivity Recall is the ratio of correctly predicted positive observations to the all observations in actual class. We have got recall of 0.92.

$$Recall = TP/(TP + FN)$$

F1 Score

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not easy to understand as accuracy, but F1 Score is usually useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if FP and FN have similar cost. If the cost of FP and FN are very different, it's better to look at both Precision and Recall. In our case, F1 Score is 0.91.

$$F1_Score = 2 * (Recall * Precision)/(Recall + Precision)$$

In the setting of 8 filters, which is the best case we have achieved, we can get the prediction metrics for testing set as follows.

TABLE I
TESTING EVALUATION

Accuracy	Precision	Recall	F1-score
0.9197	0.92	0.92	0.91

VI. DISCUSSION

As I mentioned before, our first model has very severe overfitting problem even though with just one conv layer and implementing the early stopping.

To deal with it, we first reduce the number of filters of the conv layer. Each filter of the convolution layer will try to extract one feature map. If the intrinsic binding features within the promoter are not as many as 200, using too many filters will cause overfitting for sure. Fig. 14 shows the result after I decrease the filter number from 200 to 20. The result is still not good enough but apparently much better than the original one with 200 filters(Fig. 13).

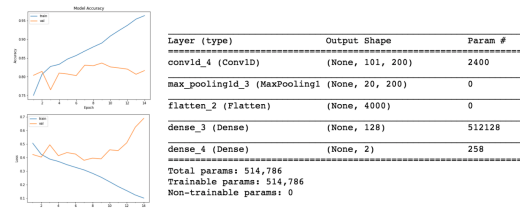


Fig. 13. Original: 1 conv layer with 200 filters

To further resolving the overfitting issue, I add one more conv layer and max pooling with just 5 filters. As showed in Fig. 15, the overfitting has been eliminated a lot after such modification. The better our model recognize the feature of the promoter region, the better it can predict. The best model can always reflect the real biology process. It turns out, in our cell, the transcription factor can recognize the promoter by some specific binding motif, the first layer of the feature

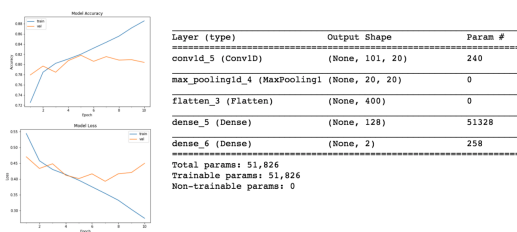


Fig. 14. Modification: 1 conv layer with 20 filters

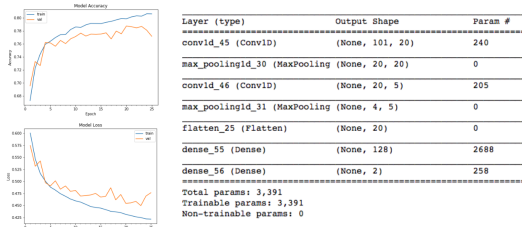


Fig. 15. Modification: Multiple conv layer with less filters

of these bind motifs is encoded in the sequence level which should be filtered out by the first conv layer. After the max pooling layer, our model will highlight and mix the first-layer feature. However, these transcription factors will not only bind the promoter according to the 1D sequence feature, to reflect the high-level interaction between these motifs, we need to add more conv layers to extract the features of higher level which is about the 2D or 3D conformation of the promoter and protein complex during promoter recognition in our cell. Thus, adding one more conv layer with just 5 filters to extract higher level feature can mimic the real binding process in cell. This could provide us more accuracy model to prevent overfitting a lot.

Besides, the kernel size has also been tuned from 21 to 11. Originally, we decide to use 21 because lots of similar works use this magic number. The best result of our model can be reach with the kernel size of 11 which might reflect the length of promoter motif interacted with the transcription factors.

VII. CONCLUSION AND FUTURE WORK

We developed a deep neural network to identify human promoter just based on the sequence feature. Our model not only focus on the 1D sequence feature extracted by the 1st conv layer, but also pay attention to the high level 2D and 3D structure to mimic the real binding process by implementing more convolutional layers with fewer number of filters. Through our experiment, we realized not just the model architecture can influence the training process a lot. Good input data is even more important.

For the future work, to achieve better prediction accuracy, we would like to develop better algorithm to select cleaner promoter and non-promoter region. We could identify the promoter region first based on the biology experimental data. For the negative set, we could filter out the region which

overlap with the positive site we have chosen to prevent any potential contamination.

Until now, we just tune the hyperparameter of the model structure itself. We did not play too much of the batch size, learning rate etc. Tuning these parameters will definitely lead to better training rate and accuracy.

Finally, once we improved our prediction rate high enough, we would like to test the performance of our model on other eukaryotes like mouse, to check the conservation of the promoter region between species.

VIII. CONTRIBUTIONS

Xuan Zhang conceived the original idea. We four designed the experiments. Xuan Zhang and Wenmiao Yu implement the input preprocess. Qi Ma and Zhuo Yue implement the network model. We four conducted input length and location experiments and performed data analysis. Zhuo Yue and Qi Ma performed performance optimization. Xuan Zhang performed the overfitting analysis. We four wrote the manuscript.

IX. GITHUB LINK (PRIVATE REPOSITORY)

<https://github.com/wmiaoX/ece228>

REFERENCES

- [1] P. Baldi and S. Brunak. “Bioinformatics: The Machine Learning Approach”. In: *Bioinformatics* (1998).
- [2] Justin Johnson Fei-Fei Li and Serena Yeung. *Course note of Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition*. Lecture note. 2017.
- [3] J.H. Holland. “Adaption in Natural and Artificial Systems.” In: (1975).
- [4] S. Knudsen. “Promoter2. 0: for the recognition of polII promoter sequences.” In: *Bioinformatics* 15 (1999), pp. 356–361.
- [5] A. Salamov and V. Solovyev. “The gene-finder computer tools for ana- lysis of human and model organisms genome sequences”. In: *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology* (1997), 294fffdfffdfffd302.
- [6] V.V. Solovyev and I.A. Shakhmuradov. “PromH: promoters identification using orthologous genomic sequences”. In: *Nucleic Acids Res.* 31 (2003), 3540fffdfffdfffd3545.
- [7] R.K. Umarov and V.V. Solovyev. “Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks.” In: *PLoS One*, 12 (2017).