Hpromoter: A Deep Learning-Based Human Promoter Classifier

Amir Souri

asou@itu.dk

June 1, 2021

Contents

1. *Introduction*

This chapter provides an overview of the necessary subjects in genomics related to this work. It presents the statement of the problem, objectives of the work, the significance of the work, the scope and limitation of the work, and the approach to overcome the problem.

1.1 Context

As the core technology in artificial intelligence, machine learning studies the algorithms that computer systems utilize to perform tasks by learning from data instead of following explicit instructions. Machine learning has been used in multiple fields, e.g. medical diagnosis [1], image processing [2], learning association [3], stock market index prediction [4], natural language processing [5], signal processing [6], etc. Since the last decade, researchers start focusing on the application of machine learning in genomics[1] [7, 8, 9, 10] and recently the application of deep learning[2] in genomics. The reason is, parallel sequencing technology known as next-generation sequencing (NGS) [11] has revolutionized the biological sciences. The NGS uses special machines, known as sequencing machines, to determine the sequence of nucleotides (As, Ts, Cs, and Gs)[3] in a piece of Deoxyribonucleic Acid (DNA). It reduces the cost for sequencing genome[4] and leads to sequencing the whole genome of different organisms. Hence it leaves us enormous unannotated data frequently. Alone the human genome comprises approximately 3.2×10^9 nucleotides. Although researchers have been applied deep learning in genomics, e.g., DeepSEA [12], DeepEnhancer [13], Basset [14], mRNN [15], ExPecto [16], DeeperBind [17], the amount of genomic data is soaring, and the challenges are growing.

The DNA is the blueprint of life [18]. It carries the information and instructions necessary to develop, survive and reproduce an organism. All living things have unique DNA within their cells. Although only 0.1 percent of each human genome is different [19], it contributes to differences in the appearance and health of humans.

In genetics[5], promoters are the essential DNA sequences that are present at non-coding DNA regions[6] in the genome. A promoter is a region on DNA, typically $\approx$ 60 base

[1] Genomics is the branch of molecular biology concerned with the structure, function, evolution, and mapping of genomes.

[2] Deep learning is a type of machine learning, inspired by the structure of a human brain, i.e., an artificial neural network with many hidden layers.

[3] Adenosine, Thymine, Cytosine, and Guanine.

[4] A genome is an organism's complete set of genetic instructions (genes).

[5] Genetics is the study of heredity and the variation of inherited characteristics.

[6] Non-coding DNA sequences are components of an organism's DNA that do not encode protein sequences.

pairs (bp) in prokaryotes[7] near the beginning of a gene [20]. It binds ribonucleic acid (RNA) polymerase, required for initiation of transcription [20]. However, the promoters do not have a fixed length, and their exact lengths are often defined only experimentally. Promoter length can vary from 100 to 1000 bp [21]. The eukaryotic[8] promoter region may have one or a combination of the promoter elements and motifs. Some of the discovered promoter motifs and elements are shown in Figure 1.1. It illustrates several promoter features in a fragment of the gene.

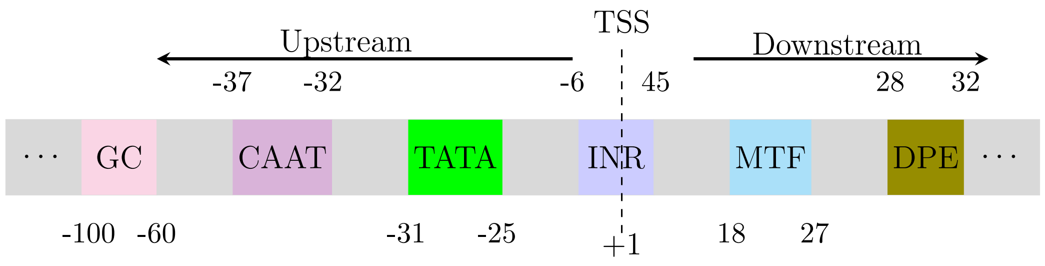

Figure 1.1: Promoter elements (boxes) and motifs.

As shown in Figure 1.1, the TATA-box is located between -31 to -25 bases to the Transcription Start Site[9] (TSS). The CAAT-box and GC-box can be found between -200 to-50 (evaluated using FindM[10]), but they usually find between -37 to -32 and -100 to -60, respectively. These features are strictly upstream of TSS. The Initiator motif (INR) is placed between -6 to 45 bases that overlap TSS. The Motif Ten Element (MTE) is located between 18 to 27 bases to the TSS. The Downstream Promoter Element (DPE) is placed between 28 to 32 bases to the TSS. Note that a promoter sequence is not necessarily contains the aforementioned promoter features, i.e., it might contain other features that have not been discovered yet or are latent[11]. Moreover, the features mentioned have a consensus sequence[12] (or canonical sequence). It means, e.g. the TATA-box may not be found in a promoter region exactly as *TATA* sequence, instead it may appear as *TATAAA* or *TATAA* or two other consensus TATA-boxes (termed TATA1 and TATA2) [22].

1.2 Related Work, Problem and Approach

First attempts for identification of promoters utilized different biological experiments such as mutational analysis [23] and immunoprecipitation assays [24] that were both expensive and time-consuming. Afterwards, various signal-based [25] and content-based [26] approaches are reported for the prediction of promoters. The signal-based approach relies on promoter features and ignores the non-features part of the sequence. However, since all promoter sequences do not show explicit features, the prediction performance of this approach is poor. The content-based approach relied on counting the frequency of k-mer by applying a k-length window across the sequence. This approach ignores the spatial information of the DNA nucleotides in the sequences. This work proposes a new deep learning-based approach to human promoter classification.

[7]Prokaryotes are unicellular organisms that lack nucleus, e.g bacteria.
[8]Eukaryotes are organisms whose cells have a nucleus. Humans are fairly typical eukaryotes.
[9]The transcription start site is the location where transcription starts. Transcription is the process of copying a segment of DNA into RNA to encode proteins.
[10]`https://ccg.epfl.ch/ssa/findm.php`
[11]The convolutional neural network tries to find certain number of such a features with certain length.
[12]The most commonly encountered nucleotides found at a specific location in DNA or RNA.

Accurate prediction of promoters is fundamental for interpreting gene expression [27] patterns, constructing and understanding genetic regulatory networks. For the last decade, researchers have sequenced the genomes of many organisms and use computational approaches to identify the organisms' genes. However, the promoters and TSS are still undetermined in most cases. The efficient software that accurately predicts promoters in newly sequenced genomes is not yet available in the public domain [28].

Besides, previous approaches require domain knowledge to hand-craft the features. Deep learning-based methods contrarily allow deploying more efficient models fed by raw data (DNA/RNA sequences) directly. Thus, the convolutional neural network (CNN) has gained attention in the promoter classification. For instance, CNNprom [28] uses CNN for short promoter sequence discrimination.

An important observation is, all the previously mentioned works have extracted the negative dataset from the non-promoter regions of the genome. Considering promoter sequences, they consist of rich features such as TATA-box, GC-Box, CAAT-Box, among others. Therefore, these features induce high accuracy in promoter classification due to inconsistency between the promoter (positive) and non-promoter (negative) samples.

Furthermore, the classification becomes unchallenged. For example, CNN models will count on some features presented at their specific locations to predict the promoter class. However, testing these models on the non-promoter sequences that comprise promoter features results in a high false-positive rate and low true-positive rate (recall). To stress this issue, consider that each gene has only one promoter, and alone chromosome[13] 2 contains 219879 CAAT-boxes [29] which is a lot more than the approximately 1400 genes in that chromosome [30]. Previous works have discussed the impact of this issue, e.g. [31].

The goal of this work is to overcome this issue using a method that merges some of the features of the positive to negative sequences. This method is experimental and follows the data analytics life cycle. This work conducts data collection, data exploration, data preparation, model building, model tuning and model evaluation.

1.3 Contributions

This work aims to prototype a deep learning model to perform binary classification of the human DNA sequences. The proposed model consists of parallel CNN layers followed by a gated recurrent units (GRU) layer that can predict whether a human DNA sequence is a promoter sequence. The implementation of this work requires the following steps:

I. Collect dataset:

Extract human promoter sequences from a promoter database that provides high-quality promoter sequences. That is the positive dataset.

II. Construct a non-promoter dataset (negative dataset):

Construct a negative dataset based on the positive dataset to prevent disparity between the two datasets concerning sequence structure.

[13]The DNA is compressed inside a chromosome because it is very long. Humans have 23 pairs of chromosomes that keep the DNA.

III. Data preparation (or preprocessing):

Transform raw data so that it can feed into a deep learning model. Clean the data to increase overall productivity and remove major errors and inconsistencies.

IV. Model building:

Employ an appropriate deep learning algorithm and leverage a deep learning library to prototype the binary classification task.

V. Model tuning:

Maximize the model's performance without overfitting by selecting appropriate hyperparameters and methods.

VI. Model evaluation:

Use k-fold cross-validation to estimate the skill of the model on new data. Select fair metrics that depend on the given classification task as well as the data.

2. *Data preprocessing*

This chapter describes how and where to collect the dataset, the structure of the dataset, properties of the dataset, the method to construct the negative dataset (non-promoter), the procedure to evaluate the proposed model, handling missing data, dealing with duplicates, and the technique to represent the data. These steps are required to understand the dataset, train and evaluate the proposed model, and avoid possible errors.

2.1 Dataset

The dataset is in fast-all (FASTA) format[1] [32], comprise human promoter sequences. It is built and downloaded from Eukaryotic Promoter Database (EPDnew)[2] [33]. The EPDnew is an annotated and non-redundant collection of eukaryotic RNA polymerase II (POL II) promoters where TSS had determined experimentally. Other databases provide promoter sequences collection, e.g. ENSEMBL[3] and NCBI[4] but EPDnew uses a new validation technique that leads to higher quality promoter sequences. This technique uses next generation sequencing data from high-hroughput promoter mapping experiments to derive sequences in the EPDnew. Besides, this technique collects new promoter from scratch by an automatic procedure taking primary experimental data and ENSEMBL gene annotation as input. Practically this means that individual entries are no longer propagated from one release to the next. This issue may require some explanations [33].

[1]It is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids have represented using single-letter codes.

[2]`https://epd.epfl.ch/EPDnew_select.php`

[3]`https://www.ensembl.org/index.html`

[4]`https://www.ncbi.nlm.nih.gov/`

The dataset contains 29598 sequences, all of the lengths 300 bp. All the sequences are collected from -249 to +50[5] bp relative to the TSS. Figure 2.1 shows the relative frequency distribution for features (motifs, boxes, elements) of the sequences. Its y-axis represents the features that are presented in the promoter sequences, e.g. *TATA & INR & CAAT & GC* means the promoter sequences contain four features namely, TATA-box, Initiator motifs, CAAT-box, and GC-box.

This work uses 5-fold cross-validation [34] to evaluate the proposed model. It uses 4-folds for training and 1-fold for validation/test. Thus, the model will trains five times, and the overall performance of the 5-folds will be calculated by averaging. The complete dataset (positive dataset combined with negative dataset) will be shuffled before feeding to the proposed model to make sure that the training/validation sets are representative of the overall distribution of the data. It is essential since the dataset is sorted by the promoter sequences' label.

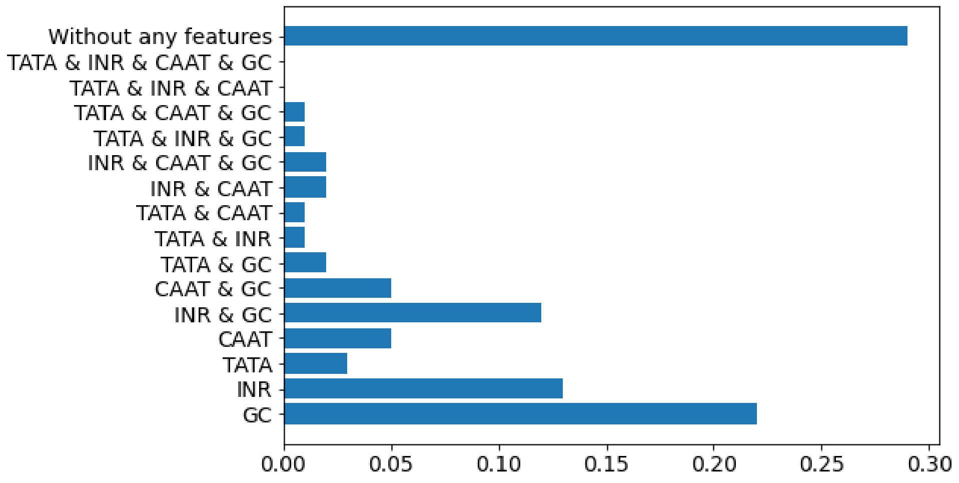

Figure 2.1: The relative frequency distribution of the features.

2.2 Negative dataset construction

To train an artificial neural network (ANN) whose ultimate task is binary classification, it needs to take both positive and negative samples (inputs). However, the collected dataset only contains the positive samples, while the model needs the negative samples additionally. There are two methods to achieve it. One is to randomly extract DNA fragments from non-promoter regions of the genome and use it as the negative dataset like many works that opted for this method [35, 36]. This method is not reliable since there is no similarity between positive and negative sequences. As a consequence, the model will discover primary features effortlessly to classify the two classes. For instance, the CAAT-box is present in many positive sequences somewhere between -200 to -50 bp. Thus, lacking this feature in the negative dataset results in high performance. Nevertheless, it fails dramatically when testing on a negative dataset contains the CAAT-box. In a nutshell, the bug of this method is the model learns to separate classes based on the existence or absence of a few uncomplicated features that give rise to a naive model.

Oubounyt et al. [37] introduced the other method to construct the negative dataset used in this work. In this method, the model cares less about (do not rely on) the features

[5]+1 refers to TSS location.

present in both positive and negative samples, i.e., it sets low weights to these features. On the other hand, it learns more complicated (latent) features. Deep learning models convergence[6] later, if feeding this type of constructed dataset. In return, it produces a more generalized model[7]. This method constructs a balanced dataset because, for each positive sequence, it generates a negative one. First, the method divides each positive sequence into 20 subsequences. Next, it chooses 12 of the subsequences arbitrarily, and substitutes their bases from the nucleotides (A, C, T, G) arbitrarily. The rest subsequences remain intact. This process is shown in Figure 2.2. The authors of DeePromoter [37] experimented with both methods and showed the goodness of the used method.

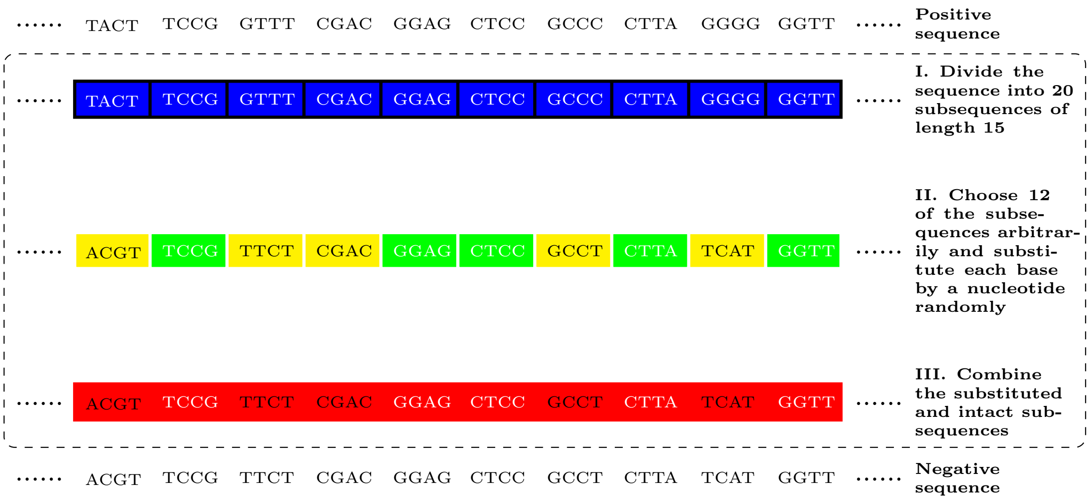
...... TACT TCCG GTTT CGAC GGAG CTCC GCCC CTTA GGGG GGTT ......
Positive sequence
I. Divide the sequence into 20 subsequences of length 15
II. Choose 12 of the subsequences arbitrarily and substitute each base by a nucleotide randomly
ACGT TCCG TTCT CGAC GGAG CTCC GCCT CTTA TCAT GGTT
III. Combine the substituted and intact subsequences
Negative sequence

Figure 2.2: The negative dataset construction method. Blue and red represent positive and negative dataset respectively. Green represents the intact subsequences while yellow represents the randomly chosen and substituted ones.

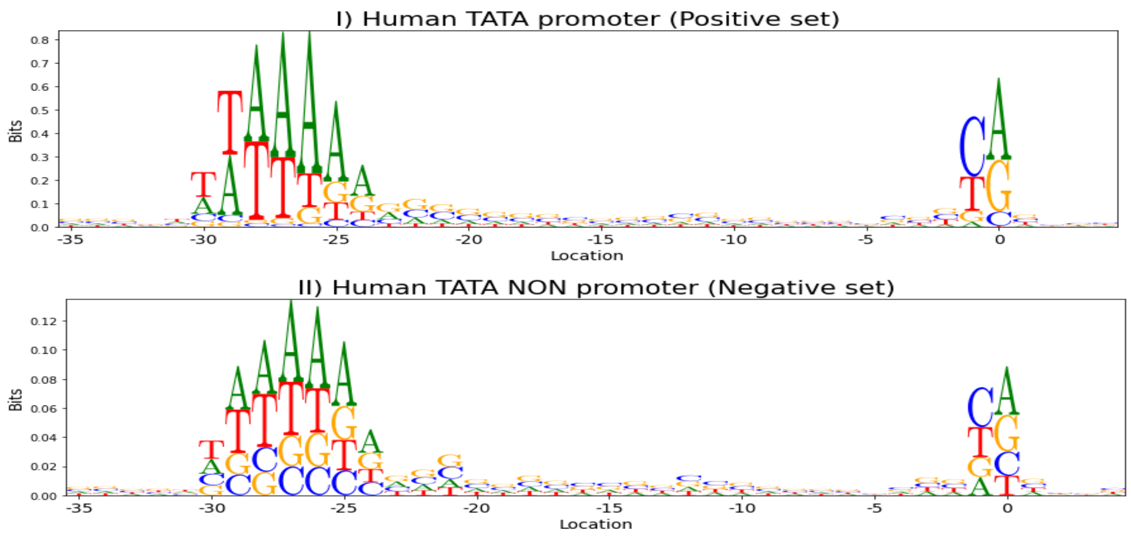

Figure 2.3: The sequence logo in human TATA promoters for both positive dataset (I) and constructed negative dataset (II). The plots show the conservation of the TATA and TSS motifs between the two datasets.

[6]A model converges when additional training will not improve the model.

[7]Generalization refers to the model's ability to adapt to new, previously unseen data, drawn from the same distribution as the one used to create the model.

The sequence logos[8] [38] for human TATA promoter data of both positive and negative datasets constructed based on the method are shown in Figure 2.3. It illustrates that both datasets have similar basic features, i.e., the TATA-box at -30 and –25 bp and the TSS at +1 bp.

2.3 Data cleaning

The real-world data often contain missing values. The reason can be data corruption or failure to record data. It is critical to handle missing data as many deep learning algorithms do not support missing values. Some techniques can solve the missing data issue. For example, deleting rows with missing values, imputing missing values for continuous variables, and predicting missing values. There was only one sequence that contained a missing value[9]. It was dropped as one sample does not impact the result. This sequence contained N[10]. For more information about different letter meanings in genomics, refer to IUPAC notation.

Duplicate data should probably remove from the dataset before modeling. If the dataset has duplicate rows, there is no need to worry about preserving the data. It is already part of the finished dataset, and one can merely remove or drop these rows from the cleaned data [39]. Duplicates in the training set cause the model to learn biases towards the duplicated inputs that lead to a model that is not robust, i.e., it cannot generalize to new data. Duplicates in the test set will result in unreliable and inaccurate performance evaluation of the model, causing the model to perform worse on unseen data. Although, the EPD is an annotated and non-redundant collection of eukaryotic POL II promoters. It is possible to have duplicated sequences while extracting them from a region. After inspecting the dataset, it turned out that there are 120 equivalent sequences which were removed. It is nuanced, and genomic data scientists should pay more attention before training a model on the genomics data extracted from a region.

2.4 Data representation

The data is sequences of string or text-based (nucleotides) that called categorical variables. To process the data further, i.e., feed data into the model, it needs to be numerical variables. There are several techniques to transform data into numerical variables. This work uses one-hot encoding [40]. It is nothing but mapping characters $A, C, G,$ and T to (1 0 0 0), (0 1 0 0), (0 0 1 0), (0 0 0 1), respectively. This process is shown in Figure 2.4. It applied to all sequences in the same way, i.e., A must be mapped to (1 0 0 0) in all the sequences. It applies as well to the other nucleotides.

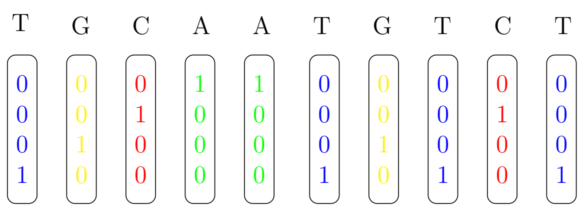

Figure 2.4: one-hot encoding of the DNA sequences.

[8]A way to display consensus sequences and relative frequency of bases and the information content.
[9]Everything other than A, a, T, t, C, c, G, g is considered as a missing value in this work.
[10]N symbol represents any one base.

3. *Model*

This chapter focuses on the proposed model, its architecture, hyperparameters, loss function, optimizer, the training process, techniques to avoid overfitting, how to save the best model, which metrics to use, and evaluation results.

3.1 The proposed model

Hpromoter is the proposed deep learning model designed to both accurately and efficiently classify a human promoter sequence. The model combines three parallel convolutional layers [41] with a gated recurrent units (GRU) layer followed by two fully connected layers, as shown in Figure 3.1. The main idea of training the model is to find several features with different lengths in the convolution layers. Then pass them to a bidirectional GRU[1] layer [42] to capture the dependencies between the learned features. Finally, the classification occurs via two fully connected layers. The proposed model is an variation of the deep learning model presented in [37]. The proposed model employed an GRU instead of LSTM. It also trained on all of the collected human promoter sequences instead of training two separated model trained on TATA-less promoter sequences and the rest sequences.

All convolutional layers first apply zero-padding[2] to the sequence to keep the length. The convolutional layers have kernel sizes[3] of 27, 14, 7, respectively. The number of features to be learned by each convolutional layer are 8, 32, 64, respectively. Both stride and dilation for all the convolutional layers were set to 1. Each convolutional layer is followed by a rectified linear activation function (ReLU)[4] and max pooling operations[5] with kernel sizes of 6 and stride 1, without any padding[6]. The hidden size (nodes) of the bidirectional GRU layer is set to 32. The fully connected layer has 128 nodes and followed by a ReLU. The output layer is a fully connected layer with one node followed by a sigmoid activation function for prediction. Five dropout layers [43] are used to prevent the model from overfitting. All with the probability of 0.5 for retaining the output of each node. One dropout layer after each convolutional layer, one after bidirectional GRU layer. The last one is placed after the fully connected layer.

[1]Bidirectional GRU is a sequence processing that consists of two GRUs. One takes the input in a forward direction, and the other in a backward direction.

[2]Adding zeros to both sides.

[3]It is also called filter size or window size.

[4]It is a piecewise linear function that will output the input directly if it is positive otherwise, it will output zero.

[5]It is a pooling operation that calculates the maximum.

[6]Padding was only applied before convolution operation in such a way to keep the length even after max pooling.

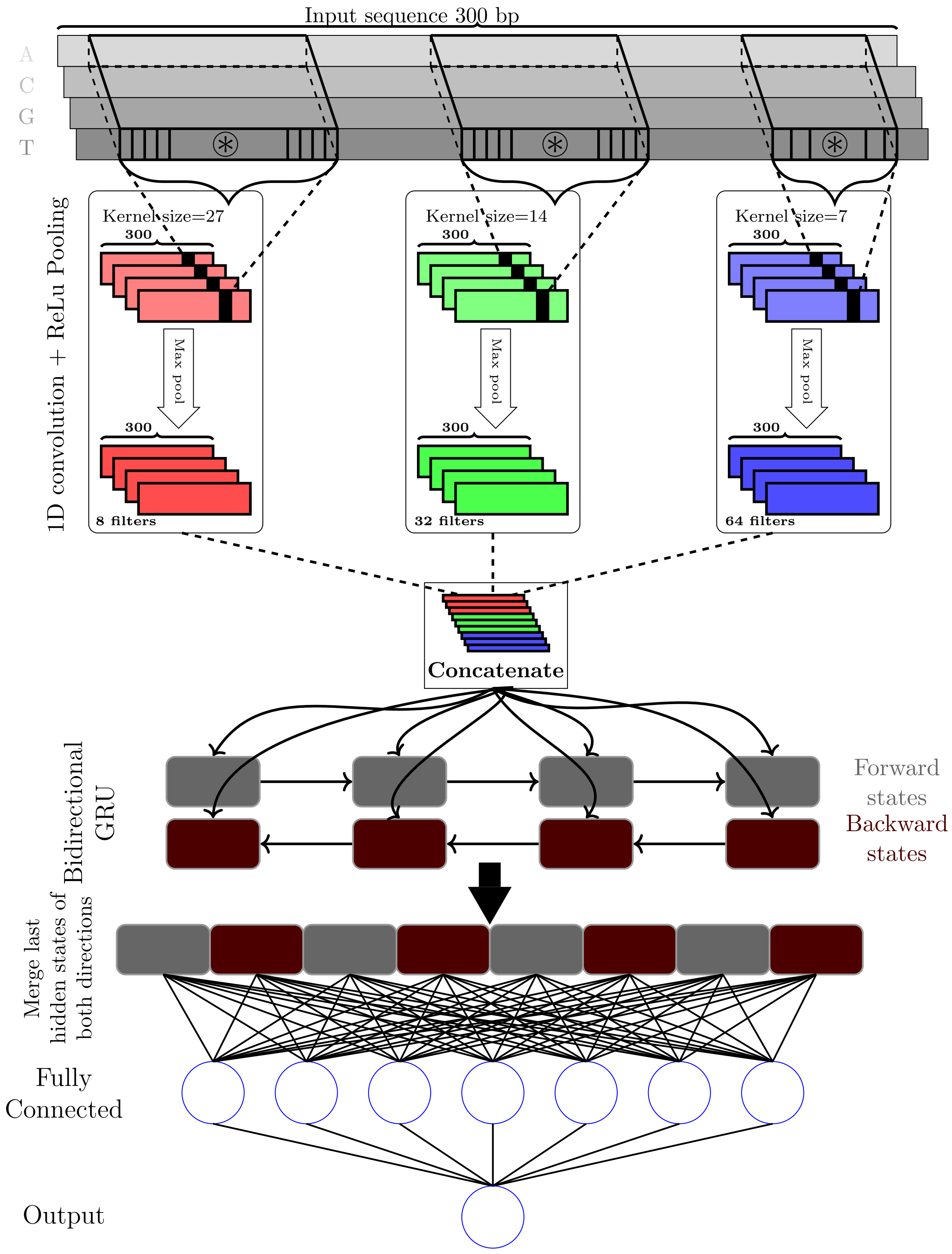

Figure 3.1: The architecture of the proposed model. The ⊛ represents convolution operation.

3.2 Training processing

PyTorch Lightning [44] framework is used for constructing and training the proposed model. The loss function is the binary cross-entropy [45] and Adam optimizer [46] is used for updating the parameters with a learning rate of 0.001. The batch size and the number of epochs used for training are 32, 100, respectively. Early stopping[7] [47] is applied based on validation loss with patience set to 3 that will avoid overtraining. ModelCheckpoint [48] was used to save the best version of the model during training based on the Matthew correlation coefficient (phi coefficient) value.

In what follows, I go through a high-level flow of a one forward pass of the network for one single sequence (input). It takes a DNA sequence (its shape is 300×4) of fixed length 300 with four features (channels), namely $[A,\ C,\ G,\ T]$[8], and then passes the same sequence to three distinct convolution layers. Each convolutional layer performs a convolution operation[9] and then an max pooling operation. The convolutional layers learn many abstract features with specific kernels. Hence the kernel size is an influential parameter, e.g., kernel size of 12 results in a promoter element of length 12. The max pooling operations summarize the abstract features present in a region of the feature map[10] generated by the convolution layers. Therefore, further computations will be performed on the summarised abstract features instead of precisely positioned abstract features generated by the convolution layers. It makes the model more robust to variations in the location of the abstract features in the input sequence.

After max pooling operations, all the learned abstract features are concatenated. The concatenated features pass to the bidirectional GRU layer. It will learn 64 (32 values from the forward states and 32 from backward states) various dependencies between the learned abstract features in both forward and backward directions. For example, it may find that if T is present at location 20, it leads to having G at location -10. Next, the last hidden states produced by the bidirectional GRU layer are merged and then pass to the fully connected layer. To classify the sequence, the outputs of the fully connected layer pass to the output layer that is again a fully connected layer with only one node followed by a sigmoid activation function.

3.3 Performance measurements

The evaluation of the proposed model performance measures four metrics. These metrics are precision, recall, F1-score, and Matthew correlation coefficient (MCC). Precision refers to the ratio between the correct positive outcomes (true positives) and all the positive outcomes [49]. For the problem statement, that would be the number of promoters that the model correctly classified out of all the promoters classified by the model. If precision is 1, all the promoter sequences (positive samples) are classified as promoter (positive) and none of the positive samples are classified incorrectly. It does not deal with non-promoter sequences (negative samples) at all hence precision 1 does not mean the model is good on classifying the non-promoter sequences. Mathematically,

[7]Early stopping is a form of regularization used to avoid overfitting.

[8]It was one-hot encoded during the data preparation phase.

[9]It is the simple application of a filter (kernel) to input, i.e., convolution is a linear operation that involves the multiplication of a set of weights with the input.

[10]The feature map is the output of one kernel applied to the previous layer.

$$Precision = \frac{TP}{TP + FP}$$

On the other hand, recall is a metric that quantifies the number of true positives out of all positive outcomes that could have been classified as positive (promoter sequence) [49]. Thus, for all the sequences that actually are promoter, recall tells how many promoter correctly classified. Even if recall is 1, all the promoter sequences may be classified correctly while there may also be non-promoter sequences that are classified as promoter. Mathematically,

$$Recall = \frac{TP}{TP + FN}$$

The MCC [50] has a range of -1 to 1 where -1 indicates a completely wrong binary classifier while 1 indicates a completely correct binary classifier. It will produce a high score only if the classification obtained decent results considering all confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally to the number of both positive elements and negative elements in the dataset. Mathematically,

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (FN + TN) * (FP + TN) * (TP + FN)}}$$

Where TP is true positive and represents correctly identified promoter sequences, TN is true negative and represents correctly rejected promoter sequences, FP is false positive and represents incorrectly identified promoter sequences, and FN is false negative and represents incorrectly rejected promoter sequences. The F1-score [51] conveys the balance between precision and recall. It can be used to select the best model based on both precision and recall. Mathematically,

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

3.4 Results and Discussion

This work used grid search method to find the optimal hyperparameters of the proposed model which result in the best performing classification based on the MCC. The hyperparameters are as follows. The number of convolution layers, kernel size, number of filters in each layer, the stencil size (aka kernel size) of max pooling, dropout probability, the number of bidirectional GRU layer, and hidden state (units) of bidirectional GRU layer.

The long short-term memory (LSTM) [52] and GRU are both a variation of recurrent neural network (RNN) [53]. The GRU is relatively new and has fewer gates. Each GRU layer has cells, as many as the number of time steps (nuclutides). Further, each cell is composed of multiple units and each unit itself includes an update gate and a reset gate. This idea applies to LSTM layer but its units include three gates namely, an input gate, an output gate and a forget gate. An GRU usually has similar or better performance than an LSTM, but it does so with fewer parameters and operations [42]. Thus, It takes less time to train compared with LSTM. On the other hand, LSTM should in theory remember longer sequences and outperform GRU in tasks requiring modeling long-distance dependencies. Chun et al. [54] have empirically evaluated both LSTM and GRU and mention that it cannot conclude which one is better. There is no simple way to decide which to use for the particular use case. Therefore, one should train both and then select the best one.

Therefore, this work prototyped another model the same as the Hpromoter but insted of the GRU layer utilized a LSTM layer to check if it improves the performance. The performances of models in terms of precision, recall, F1, MCC, and time for all 5-folds are shown in Table 3.1 and Table 3.2, respectively. Experiments were run on a machine with an AMD Ryzen 7 4800H and an NVIDIA GeForce RTX 2060.

Fold	Precision	Recall	F1	MCC	Time	Epochs
1	0.96	0.95	0.95	0.92	610	14
2	0.96	0.95	0.95	0.92	494	11
3	0.95	0.97	0.96	0.92	612	14
4	0.96	0.95	0.95	0.91	411	9
5	0.98	0.94	0.96	0.92	320	7
avg	0.96	0.95	0.95	0.92	489	11

Table 3.1: Performance of the Hpromoter.

Fold	Precision	Recall	F1	MCC	Time	Epochs
1	0.96	0.95	0.95	0.91	504	11
2	0.97	0.95	0.96	0.91	767	17
3	0.96	0.96	0.96	0.92	730	16
4	0.95	0.96	0.95	0.91	680	15
5	0.98	0.94	0.96	0.92	630	14
avg	0.96	0.95	0.96	0.91	662	15

Table 3.2: Performance of the model with LSTM.

In this work, GRU outperforms LSTM. The best version of the proposed model is highlighted in green in Table 3.1. The training and validation curves of the proposed model are shown in Figure 3.2 for each epoch (step). Figure 3.3 shows the confusion matrix of the proposed model. Cohen's Kappa [55] is a statistical measure of the agreement between two raters for categorical items. It measures the agreement between the promoter and non-promoter raters. It is calculated for the best model (fold 3), and it is shown in Table 3.3 along with precision, recall, and F1-score for each class.

Class	Precision	Recall	F1	Cohen's kappa
Promoter	0.95	0.97	0.96	0.92
Non-promoter	0.97	0.95	0.96	

Table 3.3: Performance of the Hpromoter fold 3 for each class and Cohen's kappa.

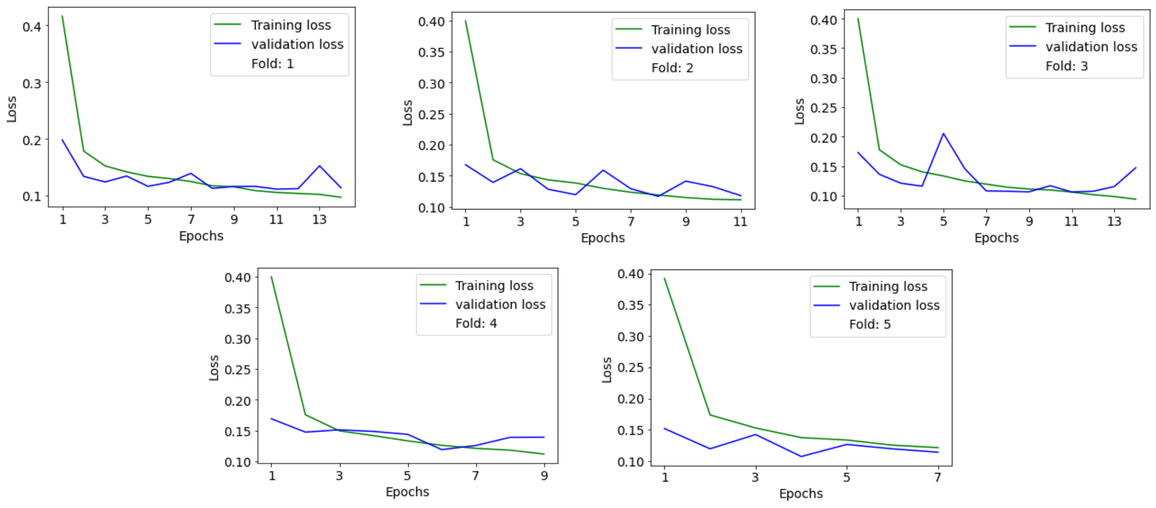

Figure 3.2: The training and validation curves of the proposed mode.

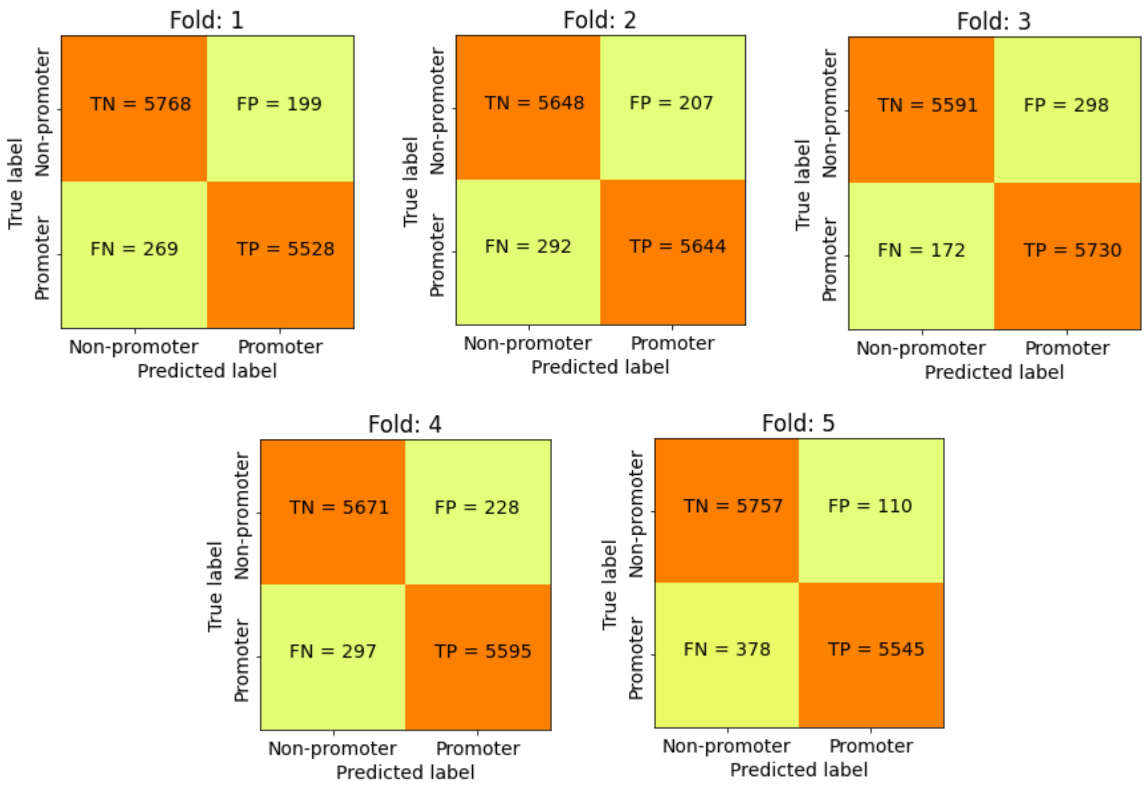

Figure 3.3: The confusion matrix of the proposed model.

4. *Conclusion*

This work prototyped the Hpromoter that is based on a combination of convolution neural network and bidirectional GRU to predict the short eukaryote promoter sequences in the human case. The essential part of this work was to classify human promoter sequences without relying on any rich promoter elements. It was achieved using a method to construct a negative dataset that forces the model to find some complicated features instead of only categorizing the promoter and non-promoter sequences based on the presence of some rich features. The proposed model classifies the human promoter sequences that contain one or more elements or no elements. Therefore, researchers can predict if a human DNA sequence is a promoter sequence using the proposed model without specifying the input sequence contains any elements, e.g., TATA-box. The proposed model gained decent performance and can be compared with the current state-of-the-art works.

The proposed model was trained with fixed-length sequences. However, many promoter elements are far away from TSS. In future work, different sequence lengths will be collected and used to train a CNN. As a result, researchers will be able to classify a human promoter sequence of various sequence lengths, i.e., researchers will not be restricted to provide fixed-length sequences for classification. Future research should also consider prototyping models to classify other organisms such as gallus, mulatta, and mice. Moreover, an autoencoder will be implemented to represent (encode) the promoter sequences instead of one-hot encoding. It will reduce the dimensionality and may improve the performance.

The source code is available on Github [56]. Note the results may slightly vary if you run the code, given the stochastic nature of the algorithm or evaluation procedure or differences in numerical precision. Consider running the code a few times.

References

[1] Patrick Walters ”Bharath Ramsundar Peter Eastman and Vijay Pande”. “Deep Learning for the Life Sciences.” In: First Edition. O’Reilly, 2019. Chap. CHAPTER 12 Prospects and Perspectives. ISBN: 978-1-492-03983-9. URL: `https://www.oreilly.com/library/view/deep-learning-for/9781492039822/`.

[2] Jan Erik Solem. *Programming Computer Vision with Python.* O’Reilly, 2012. ISBN: 9781449316549. URL: `https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/`.

[3] Ethem Alpaydın. “Introduction to Machine Learning”. In: Third Edition. The MIT Press Cambridge, Massachusetts London, England, 2014. Chap. Learning Associations. URL: `https://mitpress.mit.edu/books/introduction-machine-learning-third-edition`.

[4] Doina Bein Jithin Eapen Abhishek Verma. “Novel Deep Learning Model with CNN and Bi-Directional LSTM for Improved Stock Market Index Prediction”. In: (2019). DOI: `10.1109/CCWC.2019.8666592`.

[5] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. “Natural language processing: an introduction”. In: *Journal of the American Medical Informatics Association* 18.5 (Sept. 2011), pp. 544–551. ISSN: 1067-5027. DOI: `10.1136/amiajnl-2011-000464`. eprint: `https://academic.oup.com/jamia/article-pdf/18/5/544/5962687/18-5-544.pdf`. URL: `https://doi.org/10.1136/amiajnl-2011-000464`.

[6] Kathleen A. Kari Lila Randhawa Gurjit S. Hill. “ML-DSP: Machine Learning with Digital Signal Processing for ultrafast, accurate, and scalable genome classification at all taxonomic levels”. In: *BMC Genomics* 20 (2019). URL: `https://doi.org/10.1186/s12864-019-5571-y`.

[7] Satpathy R., Choudhury T., and Satpathy S. Mohanty S.N. Zhang X. “Data Analytics in Bioinformatics: A Machine Learning Perspective”. In: Wiley, Feb. 2021. Chap. Random Forest Algorithm in Imbalance Genomics Classification, pp. 173–190. ISBN: 978-1119785538. DOI: `10.1002/9781119785620.ch7`.

[8] Soh KP Szczurek E Sakoparnig T Beerenwinkel N. “Predicting cancer type from tumour DNA signatures”. In: *Genome Med* (2017). DOI: `10.1186/s13073-017-0493-2`. URL: `https://doi.org/10.1186/s13073-017-0493-2`.

[9] Leong HW Lu B. “GI-SVM: A sensitive method for predicting genomic islands based on unannotated sequence of a single genome.” In: *Bioinform Comput Biol* (2016). DOI: `10.1142/S0219720016400035`. URL: `https://doi.org/10.1142/S0219720016400035`.

[10] William Stafford Noble Maxwell W. Libbrecht. “Machine learning applications in genetics and genomics.” In: (2015). DOI: https://doi.org/10.1038/nrg3920.

[11] Sam Behjati and Patrick Tarpey. “What is next generation sequencing?” In: *Archives of disease in childhood. Education and practice edition* 98 (Aug. 2013). DOI: 10.1136/archdischild-2013-304340.

[12] Zhou J Theesfeld CL Yao K Chen KM Wong AK Troyanskaya OG. “Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk.” In: (2018). DOI: 10.1038/s41588-018-0160-6.

[13] Xu Min Wanwen Zeng Shengquan Chen Ning Chen Ting Chen Rui Jiang. “Predicting enhancers with deep convolutional neural networks”. In: (2017). DOI: 10.1186/s12859-017-1878-3.

[14] Kelley DR Snoek J Rinn JL. “Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks.” In: (2016). DOI: 10.1101/gr.200535.115.

[15] David A Hendrix Steven T Hill Rachael Kuintzle Amy Teegarden Erich Merrill III Padideh Danaee. “A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential”. In: (2018). DOI: 10.1093/nar/gky567.

[16] Zhou J Theesfeld CL Yao K Chen KM Wong AK Troyanskaya OG. “Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk.” In: (2018). DOI: 10.1038/s41588-018-0160-6.

[17] Alipanahi B. Delong A. Weirauch M. et al. “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning.” In: (2015). DOI: https://doi.org/10.1038/nbt.3300.

[18] Minnesota Academy of Science (1932-). *Journal of the Minnesota Academy of Science*. vb. 53–54. The Academy, 1987, p. 8. URL: https://books.google.dk/books?id=Gw9MAAAAYAAJ.

[19] J.V. Crisci et al. *Order & Diversity in the Living World: Teaching Taxonomy & Systematics in Schools*. National Association of Biology Teachers, 1993, p. 12. ISBN: 9780941212113. URL: https://books.google.dk/books?id=m8oPAQAAMAAJ.

[20] Arthur M. Lesk. “Introduction to Bioinformatics”. In: Fourth Edition. Oxford University Press, 2014. Chap. CHAPTER 12 Gene expression and regulation (Box 9.4). ISBN: 978–0–19–965156–6. URL: https://www.amazon.com/Introduction-Bioinformatics-Arthur-Lesk/dp/0199651566.

[21] Shaul Karni and Yifat Felder. “Analysis of Biological Networks:Transcriptional Networks - Promoter Sequence Analysis”. In: *Tel Aviv University* (2007). URL: http://www.cs.tau.ac.il/~roded/courses/bnet-a06/lec11.pdf.

[22] Margaret L. Grace Mahesh B. Chandrasekharan Timothy C. Hall and Alison J. Crowe. “Sequence and Spacing of TATA Box Elements Are Critical forAccurate Initiation from β-Phaseolin Promoter”. In: *THEJOURNAL OF BIOLOGICAL-CHEMISTRY* 279.9 (Feb. 2004), pp. 8102–8110. DOI: 10.1074/jbc.M309376200.

[23] Matsumine H Yamamura Y Hattori N Kobayashi T Kitada T Yoritaka A Mizuno Y. “A microdeletion of D6S305 in a family of autosomal recessive juvenile parkinsonism (PARK2).” In: (1998). DOI: 10.1006/geno.1997.5196.

[24] Jung-whan Kim et al. "Evaluation of Myc E-Box Phylogenetic Footprints in Glycolytic Genes by Chromatin Immunoprecipitation Assays". In: *Molecular and Cellular Biology* 24.13 (2004), pp. 5923–5936. ISSN: 0270-7306. DOI: 10.1128/MCB.24.13.5923-5936.2004. eprint: https://mcb.asm.org/content/24/13/5923.full.pdf. URL: https://mcb.asm.org/content/24/13/5923.

[25] S Knudsen. "Promoter2.0: for the recognition of PolII promoter sequences". In: *Bioinformatics (Oxford, England)* 15.5 (May 1999), pp. 356–361. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/15.5.356. URL: https://doi.org/10.1093/bioinformatics/15.5.356.

[26] U Ohler et al. "Interpolated markov chains for eukaryotic promoter recognition." In: *Bioinformatics* 15.5 (May 1999), pp. 362–369. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/15.5.362. eprint: https://academic.oup.com/bioinformatics/article-pdf/15/5/362/9732030/150362.pdf. URL: https://doi.org/10.1093/bioinformatics/15.5.362.

[27] J. Ma. *Gene Expression and Regulation.* Springer New York, 2010. ISBN: 9781441922038. URL: https://www.amazon.co.uk/s?k=9781441922038&i=stripbooks&linkCode=qs.

[28] Ramzan Kh. Umarov and Victor V. Solovyev. "Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks". In: *PLOS ONE* 12.2 (Feb. 2017), pp. 1–12. DOI: 10.1371/journal.pone.0171410. URL: https://doi.org/10.1371/journal.pone.0171410.

[29] Chowdhary R. Bajic V.B. Dong D. et al. "Genome-wide analysis of regions similar to promoters of histone genes . BMC Syst Biol 4, S4 (2010)". In: *BMC Systems Biology* (May 2010). DOI: 10.1186/1752-0509-4-S1-S4. URL: https://doi.org/10.1186/1752-0509-4-S1-S4.

[30] "Evelyn B. Kelly". "Encyclopedia of Human Genetics and Disease". In: First Edition. Vol. 1. GREENWOOD, 2013. Chap. Genetic Disorders 101. ISBN: 978-0313387135. URL: https://www.amazon.com/Encyclopedia-Human-Genetics-Disease-volumes/dp/0313387133.

[31] Leyi Wei et al. "Improved and Promising Identification of Human MicroRNAs by Incorporating a High-Quality Negative Set". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11.1 (2014), pp. 192–201. DOI: 10.1109/TCBB.2013.146.

[32] Geraldien A. Van der Auwera Brian D. O'Connor. "Genomics in the cloud". In: O'Reilly, Apr. 2020. Chap. The Reference Genome as Common Framework. ISBN: 9781491975190. URL: https://www.oreilly.com/library/view/genomics-in-the/9781491975183/.

[33] William Stafford Noble Maxwell W. Libbrecht. "EPD and EPDnew, high-quality promoter resources in the next-generation sequencing era". In: (2012). DOI: 10.1093/nar/gks1233.

[34] Ethem Alpaydın. "Introduction to Machine Learning". In: Third Edition. The MIT Press Cambridge, Massachusetts London, England, 2014. Chap. Cross-Validation and Resampling Methods. URL: https://mitpress.mit.edu/books/introduction-machine-learning-third-edition.

[35] Ying Qian et al. "An Improved Promoter Recognition Model Using Convolutional Neural Network". In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 01. 2018, pp. 471–476. DOI: 10.1109/COMPSAC.2018.00072.

[36] Hubbard TJ. Down TA. "Computational detection and location of transcription start sites in mammalian genomic DNA." In: *Genome Res* (Mar. 2002). DOI: 10.1101/gr.216102. URL: https://genome.cshlp.org/content/12/3/458.abstract#cited-by.

[37] Mhaned Oubounyt Zakaria Louadi Hilal Tayara K. Chong. "DeePromoter: Robust Promoter Predictor Using Deep Learning". In: *Frontiers in Genetics* 10 (2019). DOI: 10.3389/fgene.2019.00286.

[38] Thomas D. Schneider and R.Michael Stephens. "Sequence logos: a new way to display consensus sequences". In: *Nucleic Acids Research* 18.20 (Oct. 1990), pp. 6097–6100. ISSN: 0305-1048. DOI: 10.1093/nar/18.20.6097. eprint: https://academic.oup.com/nar/article-pdf/18/20/6097/3939026/18-20-6097.pdf. URL: https://doi.org/10.1093/nar/18.20.6097.

[39] Jason Brownlee. "Data Preparation for Machine Learning". In: v1.2. self-published, 2020. Chap. 5.8 Delete Rows That Contain Duplicate Data. URL: https://machinelearningmastery.com/data-preparation-for-machine-learning.

[40] Aurélien Géron. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow". In: 2nd ed. O'Reilly, 2019. Chap. Handling Text and Categorical Attributes. ISBN: 9781492032649. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[41] Frank Millstein. *Convolutional Neural Networks In Python: Beginner's Guide To Convolutional Neural Networks In Python.* North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2018. ISBN: 1986264025. URL: https://dl.acm.org/doi/book/10.5555/3235272.

[42] Ivan Vasilev. "Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch". In: Packt Publishing Ltd, 2019. Chap. Introducing gated recurrent unit. ISBN: 9781789956177. URL: https://www.packtpub.com/product/advanced-deep-learning-with-python/9781789956177.

[43] Geoffrey E. Hinton Nitish Srivastava Alex Krizhevsky Ilya Sutskever Ruslan R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* (2012). URL: https://arxiv.org/abs/1207.0580.

[44] Falcon et al. "PyTorch Lightning". In: *GitHub* 3 (2019). URL: https://github.com/PyTorchLightning/pytorch-lightning.

[45] I. Zafar et al. "Hands-On Convolutional Neural Networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python". In: Packt Publishing, 2018. Chap. cross entropy loss (log loss). ISBN: 9781789132823. URL: https://www.academicbooks.dk/da/content/hands-convolutional-neural-networks-tensorflow-0.

[46] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* (2014). URL: https://arxiv.org/abs/1412.6980.

[47] Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow”. In: 2nd ed. O’Reilly, 2019. Chap. Early stopping. ISBN: 9781492032649. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[48] Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow”. In: 2nd ed. O’Reilly, 2019. Chap. Using Callbacks. ISBN: 9781492032649. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[49] Ethem Alpaydın. “Introduction to Machine Learning”. In: Third Edition. The MIT Press Cambridge, Massachusetts London, England, 2014. Chap. Measuring Classifier Performance. URL: https://mitpress.mit.edu/books/introduction-machine-learning-third-edition.

[50] B.W. Matthews. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”. In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), pp. 442–451. ISSN: 0005-2795. DOI: https://doi.org/10.1016/0005-2795(75)90109-9. URL: https://www.sciencedirect.com/science/article/pii/0005279575901099.

[51] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. “Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation”. In: vol. Vol. 4304. Jan. 2006, pp. 1015–1021. ISBN: 978-3-540-49787-5. DOI: 10.1007/11941439_114.

[52] Ivan Vasilev. “Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch”. In: Packt Publishing Ltd, 2019. Chap. Introducing long short term memory. ISBN: 9781789956177. URL: https://www.packtpub.com/product/advanced-deep-learning-with-python/9781789956177.

[53] Ivan Vasilev. “Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch”. In: Packt Publishing Ltd, 2019. Chap. Introduction to RNNs. ISBN: 9781789956177. URL: https://www.packtpub.com/product/advanced-deep-learning-with-python/9781789956177.

[54] Junyoung Chung Caglar Gulcehre KyungHyun Cho Yoshua Bengio. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *CoRR* (2014). URL: https://arxiv.org/abs/1412.3555.

[55] Domenic Cicchetti and S.A. Sparrow. “Developing Criteria for Establishing Interrater Reliability of Specific Items: Applications to Assessment of Adaptive Behavior”. In: *American journal of mental deficiency* 86.2 (Oct. 1981), pp. 127–137. URL: https://pubmed.ncbi.nlm.nih.gov/7315877/.

[56] Amir Souri. *Hpromoter: A Deep Learning-Based Human Promoter Classifier.* https://github.com/amir-souri/Hpromoter. 2021.