Novel transformer networks for improved sequence labeling in genomics

Jim Clauwaert
Department of Data Analysis and Mathematical Modelling
Ghent University
jim.clauwaert@ugent.be

Willem Waegeman
Department of Data Analysis and Mathematical Modelling
Ghent University
willem.waegeman@ugent.be

Abstract

In genomics, a wide range of machine learning methods is used to annotate biological sequences w.r.t. interesting positions such as transcription start sites, translation initiation sites, methylation sites, splice sites, promotor start sites, etc. In recent years, this area has been dominated by convolutional neural networks, which typically outperform older methods as a result of automated scanning for influential sequence motifs. As an alternative, we introduce in this paper transformer architectures for whole-genome sequence labeling tasks. We show that those architectures, which have been recently introduced for natural language processing, allow for a fast processing of long DNA sequences. We optimize existing networks and define a new way to calculate attention, resulting in state-of-the-art performances. To demonstrate this, we evaluate our transformer model architecture on several sequence labeling tasks, and find it to outperform specialized models for the annotation of transcription start sites, translation initiation sites and 4mC methylation in *E. coli*. In addition, the use of the full genome for model training and evaluation results in unbiased performance metrics, facilitating future benchmarking.

1 Introduction

Machine learning methodologies play an increasingly important role in the annotation of DNA sequences. In essence, the annotation of DNA is a sequence labeling task that has correspondences with similar tasks in natural language processing. Representing a DNA sequence of length l as $(x_1, x_2, ..., x_l)$, where $x_i \in \{A, C, T, G\}$, the task consists of predicting a label $y_i \in \{0, 1\}$ for each position x_i, where a positive label denotes the occurrence of an event at that position, such as a transcription start site, a translation initiation site, a methylation site, a splice sites, a promotor start sites, etc. In most of these tasks, labelled data is often provided for one or several genomes, and the task consists of labelling genomes of related organisms. In other tasks, training data consists of partially-annotated genomes, i.e., some positives are known, while other positives need to be predicted by the sequence labeling model.

Although feature engineering from nucleotide sequences has deserved considerable attention in the last 30 years, the influence of the DNA sequence on biological processes is still largely unexplained. Early methods for labeling of DNA sequences typically focused on extracting features by moving windows of small subsequences, and using those features to train supervised learning models such

Preprint. Work in progress.

as tree-based methods or kernel methods. More recently, convolutional neural networks (CNNs) have been popular, starting from the pioneering work of Alipanahi et al. (1). The popularity of the CNN can be attributed to the automatic optimization of motifs or other features of interest during the training phase. Motif detection is typically done by applying a convolutional layer on a one-hot representation of the nucleotide sequence.

However, today, several obstacles remain when creating predictive models for DNA annotation. The prokaryotic and eukaryotic genome is built out of 10^7 and 10^{10} nucleotides, respectively. The detection of methylated nucleotides or start sites of the RNA transcription process, denoted as transcription start sites (TSSs), is viable on all nucleotides, and results in a large sample size that is highly imbalanced due to a low fraction of positive labels. Another problem arises when creating input samples to the model. Only a fraction of the DNA is bound to influence the existence of the annotated sites. In order to create a feasible sample input, the sequence is limited to a fixed window that is believed to be of importance. In general, the use of a fixed window surrounding the position of interest has become the standard approach for conventional machine learning and deep learning techniques. However, samples generated from neighboring positions process largely the same nucleotide sequence (or features extracted thereof), creating an additional processing cost that is correlated to the length of the window size chosen to train the model.

In practice, existing methods are not trained or evaluated using the full genome. The site of interest is in some cases already constrained to a subset of positions. This is exemplified by the site at which translation of the RNA is initiated, denoted as the Translation Initiation Site (TISs), where valid positions are delimited by the first three nucleotides being either ATG, TTG or GTG (2). This is often not applicable, and a subset of the full negative set is therefore sampled (e.g. prediction of TSS (3) (4) or methylation (5)). In general, the size of the sampled negative set is chosen to be of the same order of magnitude as the size of the positive set, constituting only a fraction of the original size (0.01% for TSS in *E. coli*). As the comparative sizes of the sampled and full negative set are extremely far apart, it is not possible to guarantee that the obtained performance metrics give a correct indication of the model's capability. Indeed, it is plausible that performances generalize poorly to the full genome.

In this study, we introduce a novel transformer-based model for DNA sequence labeling. Transformer networks have been recently introduced in natural language processing (6). These architectures are based on attention, and they outperform recurrent neural networks based on long short term memory and gated recurrent unit cells on several sequence-to-sequence labeling benchmarks. In 2019, Dai et al. (7) defined the transformer-XL, an improvement of the transformer unit for tasks constituting long sequences by introducing a recurrent mechanism that further extends the context of the predictive network. The transformer-XL performs parallelized processing over a set range of the inputs, allowing for fast training times. This architecture will be the starting point for the method that we introduce in this paper.

Our contribution is threefold. First, we define for the first time a transformer architecture for DNA sequence labeling, starting from a similar architecture that has been recently introduced for natural language processing. Second, we implement and discuss the use of a convolutional layer in the attention head of the model, an adaptation that shows a substantial increase in performances. Third, in contrast to recent pipelines for DNA sequence labeling, our model is evaluated using the full genome sequence for training and evaluation purposes, obtaining an unbiased performance that gives a trustworthy indication of the capability of the neural network. Thanks to the use of the attention mechanism, the model's architecture does not determine the relative positions of the input nucleotides w.r.t. the output label. Nucleotide sequences are processed only once, while still contributing to the prediction of multiple outputs, resulting in fast training times. In the experiments we benchmark a single transformer based model performing various annotation tasks and show it both surpasses previously published methods in performance while retaining fast training times.

2 Related work

In 1992, Horton et al. (8) published the use of the first perceptron neural network, applied for promoter site prediction in a sequence library originating from *E. coli*. Still, the development of algorithmic tools to annotate genomic features knows an earlier start. Studies exploring data methods for statistical inference on important sites based solely on their nucleotide sequence go back as

far as 1983, with Harr et al. (9) publishing mathematical formulas on the creation of a consensus sequence. Stormo (10) describes over fifteen optimization methods created between 1983 and 2000, ranging from: algorithms designed to identify consensus sequences (11) (12), tune weight matrices (13) and rank alignments (14; 15).

Increased knowledge in the field of molecular biology paved the way to feature engineering efforts. Several important descriptors include, but are not limited to, GC-content, bendability (16), flexibility (17) and free energy (18). Just this year, Nikam et al. published Seq2Feature, an online tool that can extract up to 252 protein and 41 DNA sequence-based descriptors (19).

The rise of novel machine learning methodologies, such as Random Forests and support-vector machines, have resulted in many applications for the creation of tools to annotate the genome . Liu et al. propose the use of stacked networks applying Random Forests (20) for two-step sigma factor prediction in *E. coli*. Support vector machines are applied by Manavalan et al. to predict phage virion proteins present in the bacterial genome (21). Further examples of the application of support vector machines include the work of: Goel et al. (22), who propose an improved method for splice site prediction in Eukaryotes; and, Wang et al. (23), who introduce the detection of σ^{70} promoters using evolutionary driven image creation.

Another successful branch emerging in the field of machine learning and genome annotation can be attributed to the use of deep learning methods. CNNs, initially designed for networks specializing in image recognition, incorporate the optimization (extraction) of relevant features from the nucleotide sequence during the training phase of the model. Automatic training of position weight matrices has achieved state-of-the-art results for the prediction of regions with high DNA-protein affinity (1). As of today, several studies have been published, applying convolutional networks for the annotation of methylation sites (24) (25), origin of replication sites (26) (27), recombination spots (28)(29), single nucleotide polymorphisms (30), TSSs (3) (4) and TISs (2). Recurrent neural network architectures, featuring recurrently updated memory cells, have been successfully applied alongside convolutional layers to improve the detection of methylation states (24) and TISs (2) using experimental data.

In contrast to the previously-discussed methods, where (features extracted from) short sequences are evaluated by the model, hidden Markov models evaluate the full genome sequence. However, due to the limited capacity of a hidden Markov model, this method is nowadays rarely used. Some applications include the detection of genes in *E. coli* (31) and the recognition of repetitive DNA sequences (32).

3 Transformer Network

Here we describe our transformer network for DNA sequence labeling. In Section 3.1, we adapt the auto-regressive transformer architecture of Dai et al. (7) to DNA sequences. Afterwards, an extension to the calculation of attention is described in Subsection 3.2.

3.1 Basic model

When evaluating the genome, only four input classes exist for each of the four nucleotides. During training, a non-linear transformation E is optimized that maps the input classes to hidden states $\mathbf{h}$.

$$\mathbf{h} = E(x), \qquad x \in \{A, T, C, G\}$$

where $\mathbf{h} \in \mathbb{R}^{d_{model}}$. The model is built out of several layers T and processes the genome in segments of length L. In each layer, the input hidden states are adjusted by summation with the output of the multi-head attention step, described in the following section. A final step in the layer performs layer normalization (33). Sequential steps through the layers t of the model are performed in parallel in segment s for all hidden states $\mathbf{h}^{s,t}$, stored as rows in the matrix $H^{s,t} \in \mathbb{R}^{L \times d_{model}}$

$$H^{s,t+1} = LayerNorm(H^{s,t} + MultiHead(H^{s,t}))$$

where $t \in [0, T[$. After a forward pass through T layers, a final linear combination of the outputs is performed to reduce the dimension of the output hidden states (d_{model}) to the amount of output classes. A softmax layer is applied to obtain the prediction for y.

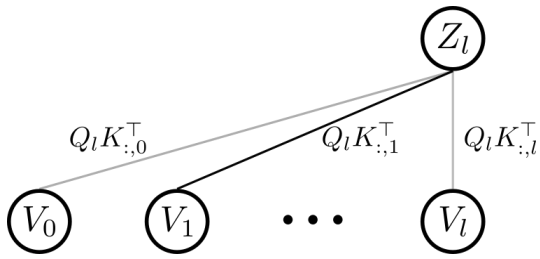

Figure 1: Attention is calculated by combination of the query Q, key K, and value V matrices, $Q, K, V \in \mathbb{R}^{L \times d_{head}}$. Intuitively, $QK^\top$ results in a matrix of weights that are used for linear combination with the values V derived from each node. Calculated attention is denoted as Z.

3.1.1 Multi-head attention

At the base of the transformer architecture lies the attention head. The attention head processes a set of inputs, stored in the matrix H, and evaluates these with one another to obtain an output matrix Z. A query (Q), key (K) and value (V) matrix hold the $\mathbf{q}$, $\mathbf{k}$, $\mathbf{v} \in \mathbb{R}^{d_{head}}$ vectors derived from the hidden states $\mathbf{h}$:

$$Q, K, V = HW^{q\top}, HW^{k\top}, HW^{v\top}$$

where $W^q, W^k, W^v \in \mathbb{R}^{d_{head} \times d_{model}}$ and $Q, K, V \in \mathbb{R}^{L \times d_{head}}$. Attention between different hidden states is thereafter calculated.

$$Z = Attention(H) = softmax(\frac{QK^\top}{\sqrt{d_{head}}})V$$

where $Z \in \mathbb{R}^{L \times d_{head}}$. Here the $softmax$ function is applied to every row of $QK^\top$. Intuitively, query and key embeddings from the input hidden states are created to evaluate relevance of hidden states with one another ($QK^\top \in \mathbb{R}^{L \times L}$), in line with the lock and key principle. In practice, this relevance is expressed by a set of weights applied to V, shown in Figure 1. The $softmax$ function rescales these weights to sum to 1, after division by the square root of d_{head}, applied to stabilize gradients (6). To increase the capacity of the model, an output is calculated using multiple attention heads (n_{head}), each featuring a unique set of W^q, W^k, W^v. This allows the model to define (through W^v) and combine (through W^q and W^k) multiple types of information from a single input H. As a final step, to create an output with dimensions equal to H, the columns of all Z matrices are concatenated and multiplied by W^o.

$$MultiHead(H) = ColConcat(Z^1(H), Z^2(H), ..., Z^{n_{head}}(H))W^o$$

where $W^o \in \mathbb{R}^{n_{head} d_{head} \times d_{model}}$.

3.1.2 Recurrence

A recurrence mechanism is implemented, described by Dai et al. (7). This allows for the processing of a single input (i.e. the genome) in sequential segments s of length L. In order to extend the context of information available past one segment, hidden states from segment $s-1$ of all layers but the last are accessible when processing s. The length of the span over which hidden states are retained is denoted by L_{mem}. In general, L_{mem} is equal to L. L is furthermore equal to the (maximum) span of hidden states used to calculate attention. Specifically, $H^{s,t,n}$, denoting the input used at position n of segment s for calculation of attention at layer $t+1$, is represented as follows:

$$H^{s,t,n} = [SG(\mathbf{h}^{s-1,t,n+1} \quad ... \quad \mathbf{h}^{s-1,t,L-1}) \quad \mathbf{h}^{s,t,0} \quad ... \quad \mathbf{h}^{s,t,n}]$$

where SG denotes the stop-gradient, signifying that no backpropagation is performed through these hidden states. This alleviates training times, as full backpropagation through intermediary values

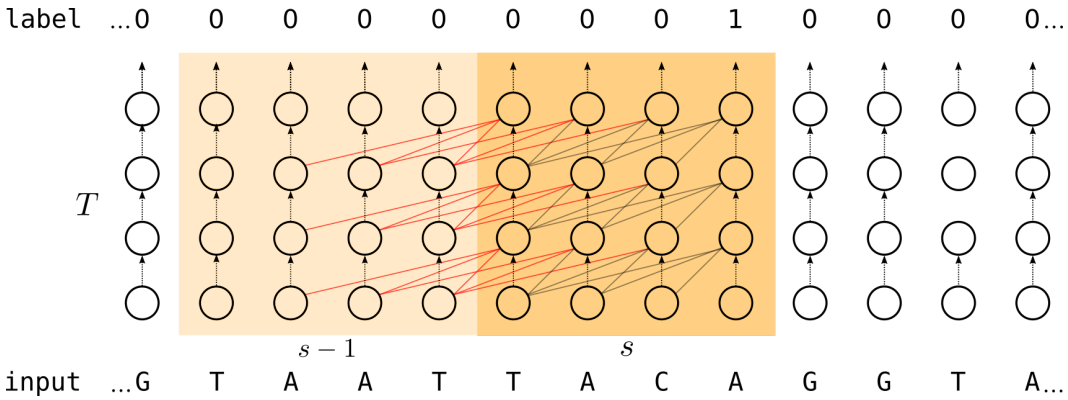

Figure 2: Simplified representation of the architecture of the implemented transformer network. The model processes the genomic strand through segments s of length L to predict the label. Data is sequentially processed in parallel through T layers. Within each segment, outputs are derived from the combination of data from the previous hidden states of the previous layer (grey connections). Cached data from the previous segment are also used, albeit no backpropagation is possible during training (red connections)

would require the model to retain the hidden states from as many segments as there are layers present in the model, a process that quickly becomes unfeasible for an increasing value of T. To inhibit the model from applying information downstream of the processed sequence, data is masked. Specifically, in each layer, hidden states $[\mathbf{h}^{n+1}, ..., \mathbf{h}^{L-1}]$ are masked when processing Z_n, $n \in [0, L[$. Figure 2 gives an illustration of the model architecture.

3.1.3 Relative Positional Encodings

Next to the information content of the input, positional information is important to calculate attention in each layer. Unlike previously discussed methods in the field, the architecture of the model does not inherently incorporate the relative positioning of the inputs with respect to the outputs. Positional information is added through the use of positional embeddings. These introduce a predefined bias to the input hidden states that is related to their relative distance with each other. During training, these biases are learned and can be used to incorporate positional information to the inputs. Attention between the hidden states i, j at position i is evaluated by expanding the algorithm (7).

$$A(H)_{i,j} = \underbrace{H_i W^{q\top} W^{k,H} H_j^\top}_{(a)} + \underbrace{H_i W^{q\top} W^{k,R} R_{i-j}^\top}_{(b)} + \underbrace{\mathbf{u}^\top W^{k,H} H_j^\top}_{(c)} + \underbrace{\mathbf{v}^\top W^{k,R} R_{i-j}^\top}_{(d)}$$

$$Z = softmax(\frac{A(H)}{\sqrt{d_{head}}})V$$

where $W^{k,H}$ (or W^k) is the weight matrix used to calculate the key (K) matrix for the input hidden states, $W^{k,R} \in \mathbb{R}^{d_{head} \times d_{model}}$ a new weight matrix used to obtain a unique key matrix related to positional information, $R \in \mathbb{R}^{L \times d_{model}}$ a matrix defining biases related to the distance between i and j. $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{d_{head}}$ are vectors that relate to content and positional information on a global level, optimized during training. Several elements make up the algorithm to obtain $A(H)$

(a): Attention based on query and key values of the input hidden states, described in the previous section.

(b): Bias based on content i ($H_i W^{q\top}$) and distance to j ($W^{k,R} R_{i-j}^\top$).

(c): Bias based on content at position j ($W^{k,H} H_j^\top$), unrelated to the relative position to i. $\mathbf{u}$ is optimized during training.

(d): Bias based on distance between i and j ($W^{k,R} R_{i-j}^\top$), unrelated to their content. $\mathbf{v}$ is optimized during training.

3.2 Extension: Convolution over Q, K and V

Important differences exist between the input sequence of the genome and typical natural language processing tasks. The genome constitutes a very long sentence, showing low contextual complexity at input level. Indeed, only four input classes exist. In contrast, meaningful sites and regions of interest are specified by motifs made out of multiple nucleotides. Through Q, K and V, attention is calculated based on the individual hidden states $\mathbf{h}$. In the first layer, hidden states of the segment solely contain information on the nucleotide classes. To expand the contextual information contained in $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$, an additional convolutional layer has been added to the attention head. Padding is applied to ensure the dimensions of the matrices to remain identical before and after the convolutional step. Applied on Q we get:

$$Q_{l,c}^{conv} = \sum_{i=0}^{d_{conv}} \sum_{j=0}^{d_{head}} Q_{f(l,i),j} W_{c,i,j}^{conv,Q}, \qquad f(l,i) = l - \lfloor \frac{d_{conv}}{2} \rfloor + i$$

where $W^{conv,Q} \in \mathbb{R}^{d_{head},d_{conv},d_{head}}$ is the tensor of weights used to convolve Q. A unique set of weights is optimized to calculate Q^{conv}, K^{conv} and V^{conv} for each layer. To reduce the total amount of parameter weights of the model, weights used to convolve Q, K and V are identical for all attention heads in the multi-head attention module. Figure 3 gives an overview of the mathematical steps performed within the attention head of the extended model.

4 Experiments and analysis

To highlight the applicability of the new model architecture for genome annotation tasks, it has been evaluated on multiple prediction problems in *E. coli*. All tasks have been previously studied both with and without deep learning techniques. These are the annotation of Translation Start Sites (TSSs), specifically linked to promoter sites for the transcription factor σ^{70}, the Translation Initiation Sites (TISs) and N4-methylcytosine sites. The genome was labeled using the RegulonDB (34) database for TSSs, Ensembl (35) for TISs and MethSMRT (36) for the 4mC-methylations. For every prediction task, the full genome is labeled at single nucleotide resolution, resulting in a total sample size of several millions. A high imbalance exists between the positive and negative set, the former generally being over four orders of magnitudes smaller than the latter. An overview of the datasets and their sample sizes are given in Table 1.

To include information located downstream of the position of interest, labels can been shifted accordingly. For all three prediction problems, a window up to 20 nucleotides downstream of the labeled nucleotide's position is commonly taken. In accordance, we have shifted the labels downstream with that amount, thereby including the information contained within this nucleotide sequence to the context of the model. The context of the model is defined as the nucleotide region that is linked indirectly, through calculation of attention in previous layers, to the output. As the span of hidden states used to calculate attention is equal to L, the context is equal to $T \times L$ nucleotides. After shifting the labels downstream, the range of the nucleotide sequence within the context of the model at position n is defined by $]n - T \times L + 20, n + 20]$.

The training, test and validation set are created by splitting the genome at three positions that constitute 70% (4,131,280-2,738,785), 20% (2,738,785-3,667,115) and 10% (3,667,115-4,131,280) of the genome, respectively. An identical split was performed for each of the prediction tasks. Split positions given are those from the *RefSeq* database and, therefore, include both the *sense* and *antisense* sequence within given ranges. For every listed performance, the model with a minimum loss on the validation set was selected and evaluated on the test set.

For this study, only a single model architecture was chosen to be used to evaluate all three annotation tasks. Hyperparameters were evaluated and selected on reduced datasets of all three problems. A final set of hyperparameters was selected to work all on all annotation tasks, listed in Table 2. Models were trained on a single GeForce GTX 1080 Ti and programmed using PyTorch (37).

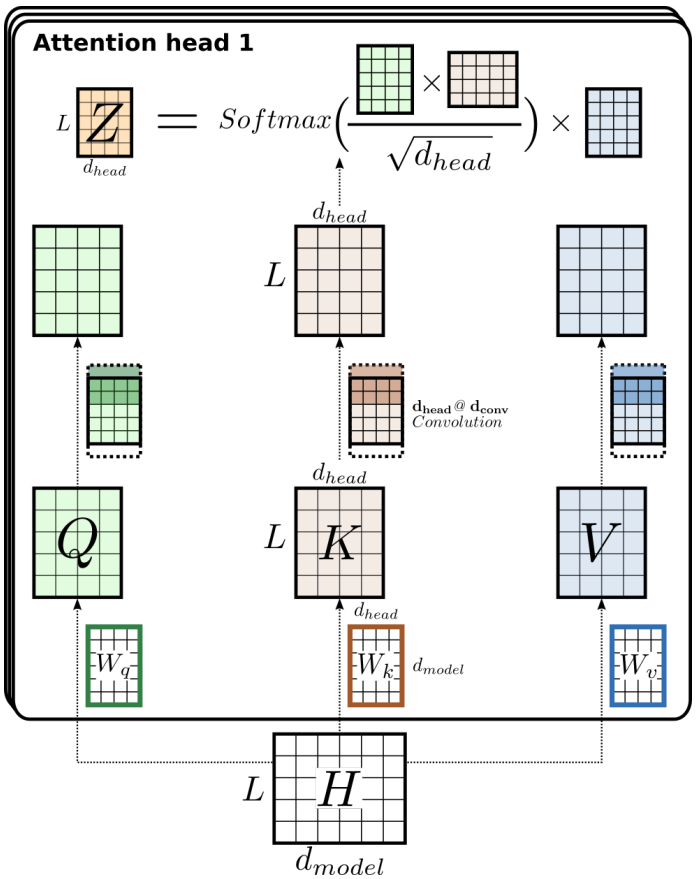

Figure 3: An overview of mathematical operations performed by the attention head to calculate attention Z. Multiple attention heads are present within each layer. Matrix dimensions are shown. An input H (holding L hidden states) is used to obtain the query (Q), key (K) and value (V) matrix through multiplication with W^q, W^k and W^v. A single convolutional layer, using as many kernels as d_{head}, enriches the rows of Q, K and V to contain information from multiple hidden states, defined by the kernel size d_{conv}. Padding is applied to uphold identical dimensions of Q, K and V before and after convolution.

Table 1: Overview of the dataset properties used in this study. From left to right: the name of the database, positive labels, negative labels and annotation task performed. All datasets are derived from *E. coli* MG1655 (accession: NC_000913.3, size: 9,283,304 nucleotides).

Dataset source	Positive labels	Negative labels	Annotation task
RegulonDB (34)	1,694 (0.02%)	9,281,610 (99.98%)	Transcription start sites
Ensembl (35)	4,376 (0.05%)	9,278,928 (99.95%)	Translation initiation sites
MethSMRT (36)	5,534 (0.06%)	9,277,770 (99.94%)	4mC methylation

4.1 Convolution over Q, K and V

Unlike application of the transformer architecture in natural language processing, the genome constitutes a very long sequence with a low contextual complexity at the input level. After processing of the input through T layers, transforming it as many times through summation (residual connection) in each layer, an output is obtained. By increasing T, a higher amount of attention heads are present in the network, effectively increasing the complexity and capacity of the model. However, as observed during hyperparameter tuning, increasing the amount of layers T did not result in any apparent improvement of the performance.

Table 2: Overview of the hyperparameters that define the model architecture. A single set of hyperparameters was selected to train a single model that showed to work well on all prediction tasks.

Hyperparameter	variable	value	Hyperparameter	variable	value
layers	T	6	segment length	L	512
dimension head	d_{head}	6	dimension model	d_{model}	32
# heads in layer	n_{head}	6	dimension input embedding	d_{embed}	32
learning rate	lr	0.0002	batch size	bs	10

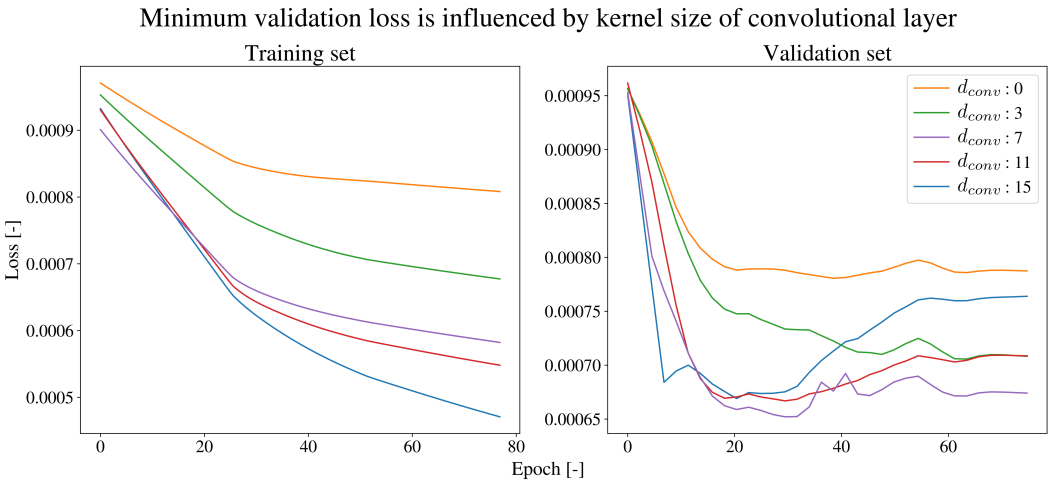

Figure 4: The smoothed loss on the training and validation set of the σ^{70} TSS dataset for different values of d_{conv}. In line with the loss curve of the validation set, the best performance on the test set was obtained for $d_{conv} = 7$. It can be observed that higher values of d_{conv} quickly results in overfitting of the model while lower values result in convergence of both the training and validation set at a higher loss.

Intuitively, the convolutional layer can be compared to its use in CNNs, where a single layer is commonly used to perform motif detection on nucleotide sequences. To illustrate, $QK^\top$ is the calculation of relevance between hidden states (see Figure 2). In the first layer, the $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ vectors are solely derived from the nucleotide input class. Convolution over the first dimension of Q,K and V incorporates information from d_{conv} neighboring nodes within $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$. Therefore, calculation of $QK^\top$ and sequential combination with V involves the combination of information from multiple hidden states. This extra step increases the contextual complexity within the attention heads without extending training times substantially, albeit at an increase of model parameters. An overview of the mathematical steps performed in the adjusted attention head is shown in Figure 3. To evaluate, performances were compared for different sizes of d_{conv} for the prediction of TSSs.

The results given in Table 3 represent the model performances for different sizes of d_{conv}. Performances are represented using the Area Under the Receiver Operating Curve (ROC AUC). Additionally, the total amount of model parameters and durations to iterate over one epoch are given. For all three annotation tasks, $d_{conv} = 7$ gives the best results. For the annotation of TISs and 4mC methylation, A significant increase of the ROC AUC score is observed, halving the difference between a perfect score of 1 and the performance of the model for $d_{conv} = 0$. For the annotation of TSSs, this difference is almost divided by four. The loss curves of the training and validation set given in Figures 4 show a stable convergence of the loss to a lower value on both the training and validation set for $d_{conv} < 7$. In contrast, $d_{conv} > 7$ increases the capacity of the model to a point where it clearly starts overfitting the training data without further decreasing the minimum loss on the validation set.

Alternatively, the reduction of the input/output resolution of the model was investigated. The use of k-mers results in a higher amount of input classes (i.e. 4^k), furthermore speeding up processing times by dividing the sequence length by k. Possibly, the increased information content in each of

Table 3: Performances of the transformer model on the test sets of the genome annotation tasks for different values of d_{conv}. All settings of d_{conv} are included for annotation of σ^{70} Transcription Start Sites (TSS). For the annotation of Translation initiation sites (TIS) and 4mC methylation, only the best setting for d_{conv} is given alongside the performance where no convolution is done. Performances are evaluated using the Area Under the Receiver Operating Curve (ROC AUC). For each setting, the total amount of model parameters and time (in seconds) to iterate one epoch during training is given.

Annotation	d_{conv}	ROC AUC	Model parameters	Epoch time (s)
σ^{70} TSS	0	0.919	185,346	337.5
σ^{70} TSS	3	0.966	241,218	364.8
σ^{70} TSS	7	0.977	314,946	371.0
σ^{70} TSS	11	0.970	388,674	379.2
σ^{70} TSS	15	0.973	462,402	383.8
TIS	0	0.996	185,346	337.5
TIS	7	0.998	314,946	371.0
4mC methylation	0	0.951	185,346	337.5
4mC methylation	7	0.985	314,946	371.0

the input classes results in better performances obtained by the model. Despite that, k-mers are represented by individual classes, and depending on where the DNA sequence is split, a sequence might be represented by multiple unique sets of input classes. As a result, motifs of importance to the prediction problem might be represented by multiple sets of input classes. The high similarity existent between the k-mer input classes with largely equal sequences (e.g. AAAAAA and AAAAAT) can be represented by the class embeddings, and are learned by the model during training. However, embeddings are optimized in function of the prediction problem (i.e. labeling). Because of the low amount of positive samples the creation of embeddings that correctly represent the similarity between classes is hindered. Moreover, a fraction of these input classes are bound to be only present in the vicinity of a positive label in either the training, validation or test set. Therefore, higher values of k quickly resulted in overfitting of the model on the training set. The use of pre-trained and fixed embeddings for each of the input classes can be used as an alternative to the adaptive embedding learned during training. A custom embedding for all classes present in a 3-mer or 6-mer setting was trained on a plethora of prokaryotic genomes using the word2vec methodology (38). As a result, model performances were improved, albeit still much worse than all models trained on single-nucleotide resolution data. The implementation of a convolutional layer in the attention head allows for the evaluation of multi-nucleotide motifs without changing the input/output resolution of the model, and seems to offer a working solution to processing and detecting motifs with varying lengths and nucleotide sequences in the DNA.

4.2 Selection of L_{mem}

The selection of L_{mem}, denoting the range of hidden states from the previous segment $(s-1)$ accessible for the calculation of attention in layer s, is an important factor effecting the training time of the model. Logically, L_{mem} is set to L during training time, allowing the use of L hidden states for every attention head in segment s. L_{mem} determines the shapes of H, Q, K and Z, influencing memory requirement and processing time. Unlike L, increasing L_{mem} does not reduce the amount of segments s in which the genome is partitioned. In other words, the value of L_{mem} has a bigger influence compared to L. In order to reduce training times, enabling applicability of the method for larger genomes, several models ($d_{conv} = 7$) were trained for different values of L_{mem} during the training phase (denoted by L_{mem}^{train}). Furthermore, performances on the test set were evaluated for different segment lengths L^{test}, where $L_{mem}^{test} = L^{test}$. ROC AUC performances and training times for the annotation of σ^{70} TSSs are shown in Table 4. Figure 5 shows the loss on the training and validation set in function of time.

From the data we state that lowering L_{mem}^{train} does not have any negative effect on the performance on the test set for $L^{train} = 512$. Moreover, decreasing L_{mem}^{train} improves convergence times on the validation set. The context of the model for $L_{mem}^{train} = 512$ spans 3,072 nucleotides ($L \times T$), a region multiple times larger than the circa 80 nucleotides window used in previous studies (4)(3)(39). For

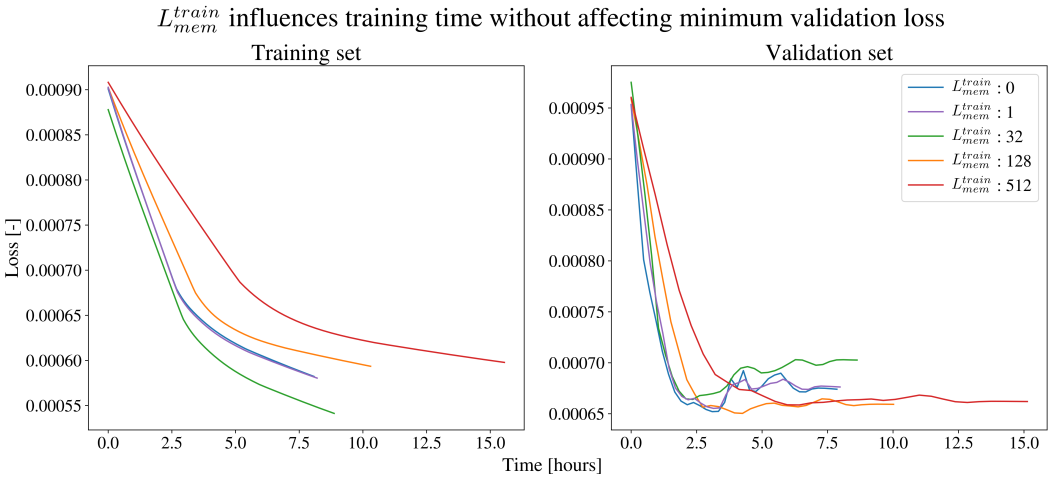

Figure 5: The smoothed loss on the training and validation set of the σ^{70} TSS dataset for different values of L_{mem}^{train}. The losses are given w.r.t. training times. All settings were trained for 75 epochs. Although increasing L_{mem} strongly influences convergence on the loss of both the training and test set, no improvements are seen on the minimum loss of the validation set.

Table 4: Performances of the transformer model on the prediction of σ^{70} Transcription Start Sites (TSS) for different values of L_{mem} during training (L_{mem}^{train}) and evaluation of the test set (L_{mem}^{test}). For each setting, $L_{train} = 512$ and $L^{test} = L_{mem}^{test}$. For each instance of L_{train}^{mem}, the time (in seconds) to iterate one epoch during training is given.

Annotation	L_{mem}^{train}	epoch time (s)	ROC AUC			
			L^{test} : 8	L^{test} : 32	L^{test} : 64	L^{test} : 512
σ^{70} TSS	0	371.2	0.664	0.957	0.977	0.977
σ^{70} TSS	1	384.2	0.638	0.945	0.968	0.969
σ^{70} TSS	32	415.5	0.611	0.940	0.969	0.971
σ^{70} TSS	128	482.6	0.593	0.907	0.963	0.973
σ^{70} TSS	512	728.7	0.570	0.883	0.954	0.972

$L_{mem}^{train} = 0$, the predictions are based on a sequence ranging from 1 to 512 nucleotides. Interestingly, the strong variation does not seem to negatively influence the model, and might even contribute to regularization, given the more stable results of the model on lower values of L^{test}. Further evaluation of the performances for different values of L^{test} support previous arguments, where values of L^{test} between 64 and 512 return relatively stable results for all models trained. Overall, given its influence on both training time and performance, a more in-depth study has to be made on the behavior of the model w.r.t. L and L_{mem} for specific genome annotation tasks.

4.3 Benchmarking

As a final step, our transformer model has been evaluated with the current state-of-the-art performances, listed in Table 5. Importantly, straightforward comparison of the transformer-based model with existing studies is not possible. Several factors have to be taken into account when evaluating given performance metrics. Most importantly, our model is trained and evaluated on the full genome. In contrast, recent studies feature a sampled negative set equal in size to the positive set. Yet, in the majority of studies cited (3)(39)(4)(5)(40), no data is made available featuring the positive and, more importantly, sampled negative set. To evaluate published CNNs, models were implemented as described and evaluated on the full genome. As a result, performances for CNNs trained on the full genome have been given. For models applying support vector machines, application on the full genome is not possible, and performances listed are those given by the study. Finally, previous reports using performances correlated to the relative sizes of the positive and negative set, such as accuracy, could not be used.

Table 5: Performances, given as Area Under the Receiver Operating Curve (ROC AUC), of recent studies and the transformer based model on the annotation of σ^{70} Transcription Start Sites (TSS), Translation Initiation Sites (TIS) and 4mC methylation. Performances listed are those reported by the paper, and generally constitute a smaller negative set. Additionally, performances with an asterisk (*) are obtained by implementation of the model architecture and training on the full genome. Applied machine learning approaches include Convolutional Neural Networks (CNN) and Support Vector Machines (SVM). When applicable, the amount of model parameters is given.

Annotation	Study	Approach	Model parameters	ROC AUC
σ^{70} TSS	Lin et al. (3)	SVM	-	0.909
σ^{70} TSS	Rahman et al. (39)	SVM	-	0.90
σ^{70} TSS	Umarov et al. (4)	CNN	395,236	0.949*
σ^{70} TSS	**This paper**	**transformer**	**314,946**	**0.977**
TIS	Clauwaert et al.(2)	CNN	445,238	0.995*
TIS	**This paper**	**transformer**	**314,946**	**0.998**
4mC meth.	Chen et al. (40)	SVM	-	0.886
4mC meth.	Khanal et al. (5)	CNN	**16,634**	0.652*/0.960
4mC meth.	**This paper**	**transformer**	314,946	**0.985**

The transformer based neural network outperforms other methods for the annotation of TSSs, TISs and 4mC methylation. New state-of-the-art performances achieved by the transformer model halve the difference between the ROC AUC of previous methods and a perfect score. With the exception of the CNN model for 4mC methylation, the model parameters are in line with previous neural networks developed. A single architecture for the transformer model ($d_{conv} = 7, L_{mem}^{train} = 0$) was trained to perform all three annotation tasks. Therefore, models were not optimized for each prediction task, and higher performances can be expected if hyperparameter tuning is performed for each task individually. Interestingly, the adjustment of the attention heads with the convolutional layer proved necessary to achieve state-of-the-art results using attention networks.

5 Conclusions and Future Work

In this paper we introduced a novel transformer network for DNA sequence labeling tasks. Unlike transformer networks in natural language processing, we added a convolutional layer over Q, K and V in the attention head of the model. As an effect, calculation of relevance ($QK^\top$) and linear combination with V extends the comparison of information to be derived from multiple (neighboring) hidden states. An improvement in predictive performance was yielded, indicating the technique to enhance the detection of influencing nucleotide motifs within the DNA sequence. Similar to CNNs, feature extraction and optimization from the nucleotide sequence is performed by the model during the training phase.

The efficacy of our transformer network was evaluated on three different tasks: annotation of transcription start sites, transcription initiation sites and 4mC methylation sites in *E. coli*. The training, test and validation set constitute full parts of the genome, easily created by slicing the genome at three points. No custom or unique datasets were created, aiding future benchmarking efforts. Moreover, the application of the full genome ensures generalization of the model's predictions and results in performances that correctly reflect the model's capability.

Models were trained within 2-3 hours. A single iteration over the prokaryotic genome on a single GeForce GTX 1080 Ti takes ca. six minutes. In general, convergence of the model on the validation/training set requires more iterations (epochs) in comparison to CNNs, an effect that is most likely correlated to L_{mem}^{train} and the inability to backpropagate through hidden states of previous segments. While still retaining positional information, the tranformer architecture does not assert the relative positions of the input nucleotides w.r.t. the output label, and allows for two important advantages that serve as an indication of the methodology to be better suited to the data type of the prediction problem. First, inputs are only processed once, and intermediary values are shared between multiple outputs. Second, increasing the context of the model, defined through L and L_{mem}, does not require larger neural networks, and is unrelated to the total amount of model parameters. These advantages improves the scalability of this technique. Specifically, a model with

a context spanning 3,072 nucleotides ($L = 512$, $L_{mem}^{train} = 512$) can process the full genome in ca. 12 minutes, as shown in Table 4.

Given the size of the eukaryotic genome, application of the technique on these genomes is not feasible at this point. Nevertheless, transformer-based models have not been studied before in this setting, and several areas show potential for further optimization of the training process time. These include the general architecture of the model, batch size, L^{train}, L_{mem}^{train}, learning rate schedules, etc.

References

[1] Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning.. *Nat Biotechnol,* **33**(8), 831–838 arXiv: cs/9605103 ISBN: 1087-0156 1546-1696.

[2] Clauwaert, J., Menschaert, G., and Waegeman, W. (April, 2019) DeepRibo: a neural network for precise gene annotation of prokaryotes by combining ribosome profiling signal and binding site patterns. *Nucleic Acids Research,* **47**(6), e36–e36.

[3] Lin, H., Liang, Z., Tang, H., and Chen, W. (2018) Identifying sigma70 promoters with novel pseudo nucleotide composition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* pp. 1–1.

[4] Umarov, R. K. and Solovyev, V. V. (February, 2017) Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PLOS ONE,* **12**(2), e0171410.

[5] Khanal, J., Nazari, I., Tayara, H., and Chong, K. T. (2019) 4mCCNN: Identification of N4-methylcytosine Sites in Prokaryotes Using Convolutional Neural Network. *IEEE Access,* pp. 1–1.

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (June, 2017) Attention Is All You Need. *arXiv:1706.03762 [cs],* arXiv: 1706.03762.

[7] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (January, 2019) Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv:1901.02860 [cs, stat],* arXiv: 1901.02860.

[8] Horton, P. B. and Kanehisa, M. (1992) An assessment of neural network and statistical approaches for prediction of E.coli Promoter sites. *Nucleic Acids Research,* **20**(16), 4331–4338.

[9] Harr, R., Häggström, M., and Gustafsson, P. (May, 1983) Search algorithm for pattern match analysis of nuleic add sequences. *Nucleic Acids Research,* **11**(9), 2943–2957.

[10] Stormo, G. D. (January, 2000) DNA binding sites: representation and discovery. *Bioinformatics,* **16**(1), 16–23.

[11] Roytberg, M. A. (1992) A search for common patterns in many sequences. *Bioinformatics,* **8**(1), 57–64.

[12] Lefèvre, C. and Ikeda, J.-E. (June, 1993) Pattern recognition in DNA sequences and its application to consensus foot-printing. *Bioinformatics,* **9**(3), 349–354.

[13] Stormo, G. D. and Hartzell, G. W. (February, 1989) Identifying protein-binding sites from unaligned DNA fragments.. *Proceedings of the National Academy of Sciences,* **86**(4), 1183–1187.

[14] Lawrence, C. E. and Reilly, A. A. (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function, and Genetics,* **7**(1), 41–51.

[15] Zhu, J. and Zhang, M. Q. (July, 1999) SCPD: a promoter database of the yeast Saccharomyces cerevisiae.. *Bioinformatics,* **15**(7), 607–611.

[16] Łozinski, T., Markiewicz, W. T., Wyrzykiewicz, T. K., and Wierzchowski, K. L. (1989) Effect of the sequence-dependent structure of the 17 bp AT spacer on the strength of consensus-like *E.coli* promoters *in vivo*. *Nucleic Acids Research,* **17**(10), 3855–3863.

[17] Ayers, D. G., Auble, D. T., and deHaseth, P. L. (June, 1989) Promoter recognition by Escherichia coli RNA polymerase: Role of the spacer DNA in functional complex formation. *Journal of Molecular Biology,* **207**(4), 749–756.

[18] Kanhere, A. and Bansal, M. (January, 2005) A novel method for prokaryotic promoter prediction based on DNA stability. *BMC Bioinformatics,* **6**(1), 1.

[19] Nikam, R. and Gromiha, M. M. (05, 2019) Seq2Feature: a comprehensive web-based feature extraction tool. *Bioinformatics,* btz432.

[20] Liu, B., Yang, F., Huang, D.-S., and Chou, K.-C. (January, 2018) iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformatics,* **34**(1), 33–40.

[21] Manavalan, B., Shin, T. H., and Lee, G. (2018) PVP-SVM: Sequence-Based Prediction of Phage Virion Proteins Using a Support Vector Machine. *Frontiers in Microbiology,* **9**.

[22] Goel, N., Singh, S., and Aseri, T. C. (January, 2015) An Improved Method for Splice Site Prediction in DNA Sequences Using Support Vector Machines. *Procedia Computer Science,* **57**, 358–367.

[23] Wang, S., Cheng, X., Li, Y., Wu, M., and Zhao, Y. (December, 2018) Image-based promoter prediction: a promoter prediction method based on evolutionarily generated patterns. *Scientific Reports,* **8**(1), 17695.

[24] Angermueller, C., Lee, H. J., Reik, W., and Stegle, O. (April, 2017) DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biology,* **18**(1), 67.

[25] Feng, P., Yang, H., Ding, H., Lin, H., Chen, W., and Chou, K.-C. (January, 2019) iDNA6mA-PseKNC: Identifying DNA N6-methyladenosine sites by incorporating nucleotide physicochemical properties into PseKNC. *Genomics,* **111**(1), 96–102.

[26] Dao, F.-Y., Lv, H., Wang, F., Feng, C.-Q., Ding, H., Chen, W., and Lin, H. (June, 2019) Identify origin of replication in Saccharomyces cerevisiae using two-step feature selection technique. *Bioinformatics,* **35**(12), 2075–2083.

[27] Li, W.-C., Deng, E.-Z., Ding, H., Chen, W., and Lin, H. (February, 2015) iORI-PseKNC: A predictor for identifying origin of replication with pseudo k-tuple nucleotide composition. *Chemometrics and Intelligent Laboratory Systems,* **141**, 100–106.

[28] Chen, W., Feng, P.-M., Lin, H., and Chou, K.-C. (April, 2013) iRSpot-PseDNC: identify recombination spots with pseudo dinucleotide composition. *Nucleic Acids Research,* **41**(6), e68.

[29] Yang, H., Qiu, W.-R., Liu, G., Guo, F.-B., Chen, W., Chou, K.-C., and Lin, H. (May, 2018) iRSpot-Pse6NC: Identifying recombination spots in Saccharomyces cerevisiae by incorporating hexamer composition into general PseKNC. *International Journal of Biological Sciences,* **14**(8), 883–891.

[30] Poplin, R., Chang, P.-C., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., Newburger, D., Dijamco, J., Nguyen, N., Afshar, P. T., Gross, S. S., Dorfman, L., McLean, C. Y., and DePristo, M. A. (October, 2018) A universal SNP and small-indel variant caller using deep neural networks. *Nature Biotechnology,* **36**(10), 983–987.

[31] Krogh, A., Mian, I. S., and Haussler, D. (November, 1994) A hidden Markov model that finds genes in E.coli DNA. *Nucleic Acids Research,* **22**(22), 4768–4778.

[32] Wheeler, T. J., Clements, J., Eddy, S. R., Hubley, R., Jones, T. A., Jurka, J., Smit, A. F. A., and Finn, R. D. (January, 2013) Dfam: a database of repetitive DNA based on profile hidden Markov models. *Nucleic Acids Research,* **41**(D1), D70–D82.

[33] Ba, J. L., Kiros, J. R., and Hinton, G. E. (July, 2016) Layer Normalization. *arXiv:1607.06450 [cs, stat],* arXiv: 1607.06450.

[34] Santos-Zavaleta, A., Salgado, H., Gama-Castro, S., Sánchez-Pérez, M., Gómez-Romero, L., Ledezma-Tejeida, D., García-Sotelo, J. S., Alquicira-Hernández, K., Muñiz-Rascado, L. J., Peña-Loredo, P., Ishida-Gutiérrez, C., Velázquez-Ramírez, D. A., Del Moral-Chávez, V., Bonavides-Martínez, C., Méndez-Cruz, C.-F., Galagan, J., and Collado-Vides, J. (January, 2019) RegulonDB v 10.5: tackling challenges to unify classic and high throughput knowledge of gene regulation in E. coli K-12. *Nucleic Acids Research,* **47**(D1), D212–D220.

[35] Cunningham, F., Achuthan, P., Akanni, W., Allen, J., Amode, M. R., Armean, I. M., Bennett, R., Bhai, J., Billis, K., Boddu, S., Cummins, C., Davidson, C., Dodiya, K. J., Gall, A., Girón, C. G., Gil, L., Grego, T., Haggerty, L., Haskell, E., Hourlier, T., Izuogu, O. G., Janacek, S. H., Juettemann, T., Kay, M., Laird, M. R., Lavidas, I., Liu, Z., Loveland, J. E., Marugán, J. C., Maurel, T., McMahon, A. C., Moore, B., Morales, J., Mudge, J. M., Nuhn, M., Ogeh, D., Parker, A., Parton, A., Patricio, M., Abdul Salam, A. I., Schmitt, B. M., Schuilenburg, H., Sheppard, D., Sparrow, H., Stapleton, E., Szuba, M., Taylor, K., Threadgold, G., Thormann, A., Vullo, A., Walts, B., Winterbottom, A., Zadissa, A., Chakiachvili, M., Frankish, A., Hunt, S. E., Kostadima, M., Langridge, N., Martin, F. J., Muffato, M., Perry, E., Ruffier, M., Staines, D. M., Trevanion, S. J., Aken, B. L., Yates, A. D., Zerbino, D. R., and Flicek, P. (January, 2019) Ensembl 2019. *Nucleic Acids Research,* **47**(D1), D745–D751.

[36] Ye, P., Luan, Y., Chen, K., Liu, Y., Xiao, C., and Xie, Z. (January, 2017) MethSMRT: an integrative database for DNA N6-methyladenine and N4-methylcytosine generated by single-molecular real-time sequencing. *Nucleic Acids Research,* **45**(D1), D85–D89.

[37] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017) Automatic differentiation in PyTorch.

[38] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (January, 2013) Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs],* arXiv: 1301.3781.

[39] Rahman, M. S., Aktar, U., Jani, M. R., and Shatabda, S. (February, 2019) iPro70-FMWin: identifying Sigma70 promoters using multiple windowing and minimal features. *Molecular Genetics and Genomics,* **294**(1), 69–84.

[40] Chen, W., Yang, H., Feng, P., Ding, H., and Lin, H. (November, 2017) iDNA4mC: identifying DNA N4-methylcytosine sites based on nucleotide chemical properties. *Bioinformatics,* **33**(22), 3518–3523.