An exploration of possible deep learning methods to model the relation between promoters and proteins

Robert van der Klis
1003241
robert.vanderklis@wur.nl

Supervisors:
Aalt-Jan van Dijk and Dick de Ridder

December 15, 2022

Abstract

Promoters and proteins share a complex biological relation through genes; from a gene's regulation one can infer certain properties about the corresponding protein, and vice versa. A better understanding of this relation could be applied to fields such as evolutionary research or drug design. We explored the viability of applying deep neural networks to model this relation by approaching the problem using a number of different neural network architectures. We discovered that transformers can classify a promoter sequence's corresponding protein domains with 2.78 times better than random performance, as defined by the positive likelihood ratio. Furthermore, we performed an initial analysis using generative adversarial networks and demonstrate these networks' suitability for modeling DNA sequences. Lastly, we provide ideas for how future researchers could use these models to generate fitting promoters for synthetic proteins.

1 Introduction

We do not yet fully understand all the intricacies of gene expression regulation. Understanding this relation is fundamental to many facets of biological research; more insight into gene regulation—important for all domains of life—could, for example, aid in evolutionary biology research by shedding more light on evolutionary processes. However, there are also ways this understanding may be directly applied. In synthetic biology we could use it to design regulatory regions for artificial genes. In medicine, we could use it to better predict effects of genetic variation in promoters, or we could design novel therapeutic drugs targeting regulatory regions if we can better predict what effect these drugs would have.

Gene expression is regulated through DNA regions called *promoters* and *enhancers*. In this paper, we will focus on promoters, as modeling the relation between gene regulatory regions and proteins is a novel area of research. Enhancers can be much farther away from genes than promoters (up to 1 Mb distance), and thus harder to analyse in relation to their genes [1]. Therefore, we aim to first research promoters, which might allow others to build on this work to include enhancers as well. Promoters are involved in DNA transcription by binding a variety of proteins that influence transcription. The protein that effectuates transcription is RNA polymerase, and by binding RNA polymerase more or less strongly, promoters can affect gene expression levels [2]. The other proteins which promoters may bind are called *transcription factors* (TFs) [3]. TFs may influence gene transcription in a variety of ways: by directly recruiting or blocking RNA polymerase; but also by recruiting *cofactors*, which are non-protein compounds that are required for some enzymes' activity (among which RNA polymerase); or affecting DNA accessibility by modifying nucleosomes and histones [4]. Which TFs bind to a promoter depends on multiple factors: the concentration of TFs, the promoter DNA sequence, but also the 3D structure of the promoter DNA [5]. In this paper, we focus on the effect of the promoter DNA sequence. TFs bind specifically to small patterns in DNA, called motifs [3], thus motifs confer specific properties to promoters by determining the conditions under which a gene will be expressed or silenced [6].

Similar expression patterns are thought to reflect similarity in function [7], and as expression patterns are influenced by promoter motifs [4], we can infer that genes with similar functions will share common promoter motifs. Some examples of this relationship exist: Allocco *et al.* [8] found that across the *Arabidopsis thaliana* genome, genes with highly correlated (84%) mRNA expression patterns had a 50% chance of sharing TF binding sites; common GC and GT motifs are found in several anaerobically induced genes in maize and *Arabidopsis thaliana* [9, 10]; and multiple adrenal steroidogenic enzymes share one promoter motif in mice [11]. An example of a more general relation between regulatory regions and genes was established by Zrimec *et al.* [12], who found that 58% of a gene's codon frequency variation could be determined using only its regulatory regions as input to a deep neural network (DNN) [13].

Given these findings, and the fact that coding regions and proteins are closely linked, it is likely that there is also a relationship between promoters and proteins. We will use neural networks to model this hypothesized relationship to determine whether it exists, and if so, its strength, so that we may eventually better understand gene regulation and potentially apply this relationship in synthetic biology. From the promoter sequence, we will try to predict protein characteristics ranging from general features such as domains, to very specific ones like the amino acid sequence. Conversely, we will try to use information contained in the protein to predict the promoter DNA sequence.

Earlier work has investigated related questions, such as classifying DNA into promoter and non-promoter regions [14, 15], classifying promoter sequences into subclasses (termed *sigma* classes) [16], or predicting TF interactions and binding sites [17]. Ji *et al.* [18] improved on previous models by adapting the Bidirectional Encoder Representations from Transformers (BERT) model for DNA [19, 20], allowing for many different applications. Tasks on which DNABERT achieved state-of-the-art performance include predicting promoter regions, identifying TF binding sites, recognizing splice sites, and identifying functional genetic variants. Alexander *et al.* [21] have modeled protein language to generate a representation of protein sequence data. However, there has been little investigation into the link between promoter and protein sequence using DNNs; as of yet, only one earlier study has explored this link [22]. Hoegen Dijkhof *et al.* encountered multiple difficulties attempting to teach an LSTM-LSTM architecture (consisting of two long short-term memory networks) the relation between promoter sequence and protein sequence—but changing the prediction target from protein sequences to protein domains using a CNN-LSTM network led to an improvement in performance. We will build on this work by optimizing the network architecture and exploring alternative ways to frame the problem in a deep learning setting. We train a variety of network architectures with different loss functions, and compare their ability to model the relation between promoters and proteins according to a performance measure such as their prediction accuracy. Next, we investigate which architectures were successful, and why. We conclude by determining the optimal network architecture for these purposes, and give recommendations for future research.

2 Material & Methods

We use three different approaches to study the relation between promoters and proteins (Figure 1). First, we *classify* protein domains based on promoter sequences using a CNN, DeePromoter, and DNABERT, to determine whether we can find a link between promoters and proteins using a very non-specific target (i.e. domains) (Figure 1a). Next, we switch to a harder, much more specific target: we attempt to *translate* promoter sequences to protein sequences and protein sequences to promoter sequences using LSTMs (Figure 1b). Lastly, we *generate* promoter sequences using a GAN—specifically, ExpressionGAN (Figure 1c) [23].

2.1 Datasets

We first developed our models using artificial data. The results of these tests were used to optimize hyperparameters and determine which properties of a dataset cause poor performance. These artificial data were created with two functions; one for creating pairs of promoter sequences and protein domains, and one for creating pairs of promoter sequences and protein sequences.

The function creating artificial pairs of promoter sequences and protein domains first makes n (a specified parameter) random promoter motifs with corresponding protein domains. Subsequently, random promoter sequences were created, and each promoter sequence was assigned a certain probability (speci-

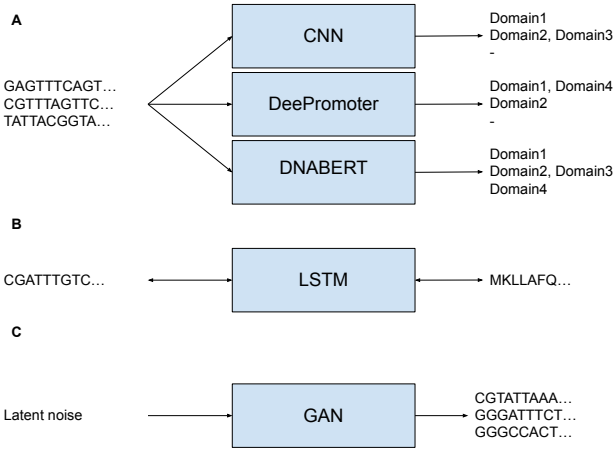


Figure 1: High-level overview of the workflow of this paper. A: Classification. We classified domains in a protein sequence from their corresponding promoter sequences using a CNN, DeePromoter, and DNABERT. B: Translation. We translated DNA sequences to protein sequences with an LSTM, and vice versa. C: Generation. We generated promoter sequences using a GAN, with random latent noise as input. All DNA sequences shown here are promoter sequences. Arrows point in the direction of data flow.

fied parameter) of containing a promoter motif. Motif presence in the promoter was 100% predictive of a domain label. Other parameters were the number of datapoints to create, the length of promoter sequences, and the length of the motifs. The scheme according to which parameters were varied and the default values for each parameter are shown in Table 1. Artificial datasets with these parameters were then used to train the DeePromoter network [15] and the CNN. The performance of the networks was evaluated and compared against performance on datasets with other parameters.

To create sequences for the translation (LSTM) and generation (ExpressionGAN) models, artificial promoter and protein sequences were created with various sets of parameters. DNA motifs of length 10 and corresponding amino acid motifs of length 5 were created. A length 10 DNA motif will occur once every $4^{10} \approx 10^6$ nucleotides by chance, whereas each dataset contained approximately 50,000 datapoints of 100 nucleotides each, working out to the DNA motif occurring on average only 5 times in the entire dataset by chance. A similar calculation can be made for amino acid motifs; amino acid motifs were even less likely to occur by chance.

Table 1: Scheme according to which the parameters in the artificial promoter sequence-protein domain dataset creation function were varied. Values in the second row were the default, and the values were adjusted one by one. That is, at all times, values in four out of five columns were at their default, and at most one parameter was at a non-default value.

Domain-promoter pairs	Motif probability	Number of sequences	Promoter length	DNA motif length
1	0.1	10,000	50	5
2	0.3	50,000	100	10
3	0.5	90,000	300	15
4	0.7		500	20
5	0.9		1,000	25
10				

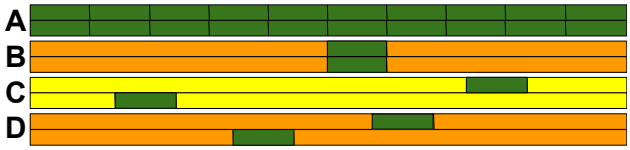


Figure 2: Schematic overview of artificially generated DNA sequences. Each of the eight horizontal bars indicates one DNA sequence. Green regions are motifs, orange regions are random nucleotides, yellow regions are As. A: DNA motif repeated ten times. B: one DNA motif at a fixed place. C: one DNA motif at a random place, where non-motif bases are As. D: one DNA motif at a random place. Protein sequences were created analogously.

Then, sequence pairs were generated in four different ways (see Figure 2 for a schematic overview):

- 1: DNA motif and AA motif repeated 10 times in their respective sequences
- 2: DNA motif and AA motif each occur once in each sequence, at a fixed position
- 3: DNA motif and AA motif each occur once in each sequence, at a random position. All non-motif DNA bases are As.
- 4: DNA motif and AA motif each occur once in each sequence, at a random position.

Characters outside of motifs were generated randomly, except in the third dataset. For example sequences, see Figure S1.

Biological data were obtained from EPDnew, UniProt, InterPro, and GenBank [24–27]. Promoter sequences were retrieved from every eukaryotic

species on EPDnew, by using the "Select / Download tool", and leaving all settings on default, except for "Select only the most representative promoter for a gene", which was turned on. We use promoters as either input or output depending on the training task, so we require a one-to-one relation where each promoter corresponds to one protein sequence, and each protein sequence corresponds to one promoter sequence [24]. For each found promoter the bases from 1000 bp before the transcription start site to 250 after (TSS-1000 to TSS+250) were selected. These cutoffs (TSS-1000 and TSS+250) were chosen as the core and proximal promoter generally extend a few hundred bases before the TSS, and a few dozen bases after the TSS [28]. The analyses were performed with various subranges of these bases: the DeePromoter model and our own CNN were trained on sets with ranges -1000 - +250 and -200 - 0, to compare whether the results were substantially different. DNABERT was trained on the range -500 - 0, as the input for DNABERT must be $\leq$ 512 bases in length, and we only trained it on one subset as DNABERT was the model that took by far the longest to train.

To be able to link these gene identifiers to protein sequences and domains, the UniProt ID mapping file was downloaded and used to uniquely link each gene ID to a protein ID [25]. These protein IDs were subsequently searched in the InterPro dataset [26]. However, the InterPro dataset integrates a diverse set of protein information: families, domains, superfamilies, conserved sites, and more. These regions occasionally overlap; for example, domains from SMART might overlap with domains from Pfam [29]. To obtain a standardised set that could be used to train a neural network, only the data from Pfam were selected [30]. The protein sequences were searched in the GenBank database using the UniProt ID mapping, since searching the protein sequences themselves in the UniProt database resulted in a lot of data being left out [27]. Sometimes, one UniProt protein ID in one species was coupled to multiple GenBank identifiers. In these cases, the first GenBank identifier found in the UniProt ID mapping file was kept to obtain a one-to-one combination of inputs and outputs.

The end result was promoter sequences paired with amino acid sequences and with Pfam domains of corresponding proteins. We subsequently created three subsets of the Pfam domains, by implementing a cutoff on the number of times a domain occurs in the dataset. Sequences that were left with no corresponding domains after this selection step were removed altogether. In the first subset the cutoff was 10, to remove exceedingly rare domains that are likely very hard to classify. After this selection step, 3,681 unique domains and 95,001 sequences were left. In the second subset the cutoff was 100, leaving 238 domains and 45,861 sequences, and in the third subset, the cutoff was 1,500, leaving 2 domains and 4,306 sequences. In future, we refer to these datasets as cutoff 10, cutoff 100, and cutoff 1500, respectively.

2.2 Networks

For classification of protein domains using the promoter sequences as input, three models were tried: DeePromoter [15], DNABERT [18], and a simple CNN of our own for comparison purposes (Figure 1a). DeePromoter was included as a baseline for comparison of the different models, as Hoegen Dijkhof *et al.* [22] reported their results using this model. DNABERT is the current state-of-the-art model for inference on DNA sequences [18].

For translation and generation, two different models were used: LSTMs (Figure 1b), for the same reason as DeePromoter above, and because several recommendations for future research were made by Hoegen Dijkhof *et al.* [22] which we could implement. Next, a generative adversarial network (GAN) created by Zrimec *et al.* [23] (ExpressionGAN) was used to generate promoter sequences [31] (Figure 1c). We generated promoter sequences and not protein sequences as we only had time to do one option, and as we had to choose one option of two, we elected to choose the option that was most applicable to research: researchers are generally interested in designing a protein and finding a fitting promoter, and not designing a promoter and then finding a fitting protein. Furthermore, the aim was to include information about the protein (i.e. domains) in the input to generate promoter sequences, and a conditional GAN (cGAN) generating promoters would be suited to this task [32]. In DNA, no such labels are present, so a cGAN generating protein sequences would be much harder to implement. Unfortunately, due to time constraints, we were not able to implement a cGAN, instead only performing initial analysis with a non-conditional standard GAN. Thus, we were not able to improve upon the model developed by Zrimec *et al.* [23], and we only managed to investigate this model's suitability for further exploration of the questions discussed in this paper.

A DeePromoter implementation in PyTorch was published by Bac [33], which we used. This implementation exactly follows the architecture as described in the paper by Oubounyt *et al.* [15]. The network was trained using a weighted binary cross entropy loss function, as without correcting for data imbalance, the network would simply predict 0 everywhere. The reason for this is that the vast majority of protein sequences have only a few domains. Using the full dataset as example, each sequence had on average 1.52 ± 0.91 domains, with each sequence having at least 1 domain, and at most 11 domains. On the other hand, there are a total of 3,681 unique domains in the full dataset. Therefore, with a one-hot vector with 3,681 total values of which on average 1.52 are 1 and the rest 0, predicting 0 everywhere would result in more than 99.9% accuracy. The weight used was $\frac{1}{\mu}$, where μ is equal to the fraction of 1s in the entire one-hot encoded dataset, and the network was optimized using stochastic gradient descent with a learning rate of 10^{-2}. DNABERT was implemented using Hugging Face Transformers, following the architecture as described by Ji *et al.* [18], only changing the number of output neurons depending on the number of domains in the dataset. Before training, sequences were tokenized using Hugging Face's AutoTokenizer. DNABERT was then trained with the default optimizer of Hugging Face, AdamW, using Hugging Face's built-in learning rate scheduler. We used the same loss function as for DeePromoter. The architecture of the CNN and LSTM is explained in Section S2. The authors of ExpressionGAN published their implementation [34], and we used this without any modifications; it was trained using the Adam optimizer and a Wasserstein loss function with learning rate 10^{-5}.

All scripts and models were run on a server provided by the Bioinformatics group of Wageningen University & Research. The server runs Ubuntu 18.04.6 LTS on two Intel Xeon Gold 6242 CPUs with 32 threads of 1200MHz each, two Tesla T4 GPUs, and 768GB of memory.

2.3 Network evaluation

For the classification models, a custom evaluation function was implemented to return the accuracy, precision, recall, and specificity. The same function was used for all three models to ensure that results were comparable. We define these performance measures as follows:

$$\begin{aligned}
\text{accuracy} &= \frac{TP + TN}{TP + FP + FN + TN} \\
\text{precision} &= \frac{TP}{TP + FP} \\
\text{recall} &= \frac{TP}{TP + FN} \\
\text{specificity} &= \frac{TN}{TN + FP},
\end{aligned}$$

where TP indicates the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives. Here, each positive is one specific promoter sequence being associated with one specific domain, and each negative is one specific promoter sequence not being associated with one specific domain.

The classification networks were additionally evaluated using the positive likelihood ratio, defined as

$$\begin{aligned}
\text{LR+} &= \frac{\text{recall}}{1 - \text{specificity}} && (1) \\
&= \frac{\Pr(P+ \mid D+)}{\Pr(P+ \mid D-)}, && (2)
\end{aligned}$$

where P+ indicates a positive prediction, and D+ and D- indicate a domain being either present or absent, respectively. The positive likelihood ratio is interpreted as the likelihood of a ground-truth positive being classified positive divided by the likelihood of a ground-truth negative being classified as positive.

As for the GAN, the way in which ExpressionGAN reports performance is defined as

$$\frac{1}{|G|} \sum_{i \in G} g_i - \frac{1}{|D|} \sum_{j \in R} r_j, \tag{3}$$

where g_i is the score the discriminator assigns the ith generated sequence, r_j is the score the discriminator assigns the jth real (ground truth) sequence, and G and R are the sets of all generated and real sequences, respectively. The discriminator attempts to assign generated sequences the lowest possible scores and ground truth sequences the highest possible scores, whereas the generator attempts to bring these scores closer together, in other words, to 'fool the discriminator'. Therefore, a score close to 0 means that the discriminator cannot separate generated and ground truth sequences well.

To compare promoter sequences generated using ExpressionGAN to biological promoter sequences, a set

of 640 generated sequences and 640 biological promoter sequences were searched for motifs using the FIMO tool in the MEME suite [35, 36]. As *Arabidopsis thaliana* is a widely studied plant, many of its motifs are known. Therefore, we generated sequences using ExpressionGAN, which we trained exclusively on *Arabidopsis* sequences. Likewise, the biological promoters were also a random subset of the EPDnew *Arabidopsis* dataset [24]. The motifs used in the search came from the 9th release of the JASPAR dataset of *Arabidopsis* motifs [37].

3 Results and discussion

We will first assess whether a relation can be found between promoters and their corresponding proteins, by attempting to classify protein domains based on promoter sequences using DNABERT, DeePromoter, and a CNN. Next, we will attempt to translate promoter sequences based on protein sequences and vice versa using LSTMs. Lastly, we will try to generate promoter sequences using ExpressionGAN.

3.1 Classification models suggest a relation between promoters and protein domains

We first wondered whether our CNN and DeePromoter would be suited to modeling the problem, so we verified this (and developed the CNN) by doing preliminary analysis on artificial data. When tested on artificial datasets with parameters as shown in the scheme in Table 1, the DeePromoter network performed well at predicting artificial protein domains from artificial promoter sequences, with test accuracy above 95% in most cases. However, several parameters appeared to have an particularly large influence on the network's performance (Table 2). Increasing the number of promoter-domain pairs to ten resulted in many more false negatives, and similar results were observed with longer promoters. On the other hand, decreasing the motif length resulted in more false positives.

The reason for the reduction in accuracy is clear: learning 10 patterns in one dataset is harder than learning 1 pattern in one dataset, and thus, accuracy decreases. As the default chance for any motif to be present was selected to be 30% (Table 1, predicting a domain to be absent as the default seems the optimal strategy as this would be correct in 70% of cases. Consequently, recall decreases.

Similarly, with a higher promoter length or shorter motifs, we expect accuracy to decrease: with a 10x higher promoter length, we expect each motif to be present approximately $5 \cdot 10 = 50$ times in the dataset by chance, and not to be labelled as such. Similarly, shorter motifs occur by chance once in every $4^5 = 1{,}024$ bases, meaning that each motif is present $\frac{100 \cdot 50000}{1024} \approx 5{,}000$ times on average in the dataset by chance. This conflicting information hampers training.

Even though decreasing motif length results in far more chance occurrences of motifs, increasing the promoter length by a factor of 10 decreases accuracy far more. It is likely that this is due to the increased number of parameters; longer sequences result in more input neurons for the dense layer, thus increasing the number of parameters in the network as a whole, making it harder to train. However, when interpreting these results to determine a strategy for biological data, we should note that longer artificial promoter sequences contain no additional information—shorter and longer artificial promoters both contain at most two motifs, in which rests all the information that may be used to predict the label. The information in biological promoters, on the other hand, is more spread out, instead of being concentrated in just two motifs: the regulatory elements of a gene are generally thought of as consisting of a core promoter, a proximal promoter, and distal regulatory elements [38]. Nevertheless, these results on artificial data provide a cautionary tale; incorporating more data without a commensurate increase in the number of labels or a substantial increase in the amount of information may do more harm than good. Hence, we conclude that incorporating much more data to include distal regulatory elements in the input is unlikely to improve results.

After this initial exploration on artificial data, three different networks were trained on biological data; a simple network, a DeePromoter-inspired network, and DNABERT. On the cutoff 1500 dataset—equivalent to binary classification as only two domains are left in this dataset—all three networks classify approximately 66% of datapoints correctly (except for the CNN, which performs worse with on the dataset consisting of a range of 200 bases) (Tables S1, 3 and 4). Moving to datasets with lower cutoffs, and thus more possible labels, the test accuracy increases.

Table 2: Performance statistics of DeePromoter-inspired network on artificial dataset with parameters as stated in Table 1.

	Default values	Domain-promoter pairs: 10	Promoter length: 1,000	Motif length: 5
Accuracy	0.98	0.91	0.8	0.94
Precision	0.96	0.94	1	0.78
Recall	0.93	0.54	0.35	1
Specificity	0.99	0.99	1	0.92

Table 3: Performance statistics of DeePromoter inspired network on biological dataset containing promoter sequences with domains of corresponding proteins as labels

Included nucleotides counted from TSS	# Datapoints	# Unique domains	Accuracy	Precision	Recall	Specificity
-1000 - +250	95,001	3,681	0.785	0.001	0.514	0.785
-1000 - +250	45,861	238	0.726	0.011	0.658	0.726
-1000 - +250	4,306	2	0.656	0.661	0.641	0.672
-200 - 0	95,001	3,681	0.785	0.001	0.511	0.785
-200 - 0	45,861	238	0.736	0.012	0.558	0.737
-200 - 0	4,306	2	0.695	0.667	0.781	0.609

However, the precision and recall together give more insight: using the largest dataset, the network predicts 1000x more positives than are actually present in the dataset (due to the weighted binary cross entropy we used to train the model), and even then, recall is only around 50%. Thus, it is clear that it is much harder for the network to correctly classify more domains, even though the test accuracy is higher.

However, even on these larger datasets, all three models performed better than random. With 3,681 unique domains, and using the longest input sequences, the LR+ (Equation (1)) is 2.28 for our own CNN, 2.39 for the DeePromoter-inspired network, and 2.78 for DNABERT. Thus, while performance (precision in particular) leaves much to be desired, it is clear there is a signal, and neural networks can discover it.

We initially expected the CNN to perform worst, DeePromoter to significantly improve upon it, and DNABERT to outperform both. This difference seems smaller than expected, which might be explained by the fact that both DeePromoter and DNABERT were not initially designed for this purpose. However, DNABERT *was* developed to identify motifs, and as we hypothesized that the relationship between promoters and proteins would manifest itself through motifs, we expected DNABERT to be able to adapt to this task during fine-tuning. DNABERT was, however, only pre-trained on human DNA, whereas we used data from all eukaryotes in the EPD database. Though motifs are generally highly conserved, promoters are not [39]. Furthermore, even though motifs are highly conserved between e.g. mammalian species, it is currently unknown whether this also holds for species as evolutionarily distant as *Homo sapiens* and *Saccharomyces pombe*, both of which are in the EPDnew database. A potential solution may be to give the model not just the promoter sequence as input, but also the species.

We would have liked to also use DNABERT to translate sequences from promoter to protein, as this would provide a natural way to incorporate information of the entire sequence into predicting the output, but we could not find a way to adapt DNABERT for this purpose. The reason for this is that we could only use DNABERT for multilabel classification, and could not determine how to adapt DNABERT to give sequences as output instead. An initial approach might be to give DNABERT a relatively large number of output neurons, which could then serve as an encoder hidden state of sorts, by giving these output neurons as input to an LSTM.

In summary, there is a relation between promoters and proteins, and this relation can be found by neural networks.

Table 4: Performance statistics of DNABERT on biological dataset containing promoter sequences with domains of corresponding proteins as labels

Included nucleotides counted from TSS	# Datapoints	# Unique domains	Accuracy	Precision	Recall	Specificity
-500 - 0	95,001	3,681	0.792	0.001	0.579	0.792
-500 - 0	45,861	238	0.891	0.018	0.372	0.894
-500 - 0	4,306	2	0.664	0.663	0.666	0.662

3.2 LSTMs cannot find a relation between promoters and protein sequences

Having concluded that there is a relation between promoters and proteins, we now proceed by investigating further how strong the link between promoters and proteins is. We do this by attempting to predict promoter sequences from protein sequences and protein sequences from promoter sequences. Performances of LSTMs trained on artificial datasets are shown in Table 5. The threshold indicates the proportion of bases that must match with the true motif to be counted as correct; a threshold of 0.8 means that out of the 10 bases in the motif, a given subsequence of length 10 in the output sequence must match at least 8 locations. The LSTM is able to learn to output sequences that repeat a motif 10 times, but cannot learn the pattern of one motif at a fixed place; even allowing for 2/10 mismatches, only 2% of the sequences produced contained the motif. Using frequency-aware cross entropy (FACE, see Section S2) as criterion did not improve performance, nor did adding attention (Section S2) [22, 40]. LSTMs were also not able to model the distribution of nucleotides in the artificial dataset well (Figure 3), and most sequences consisted of only a few nucleotides. The nucleotide distribution of the model with FACE but without attention might seem somewhat better in Figure 3, but the predicted sequences were all approx. 15 to 25 nucleotides long, whereas the ground truth sequences were 100 nucleotides long. Similar results were found when applying these models to biological data, both predicting protein sequences from promoters and vice versa. The use of FACE and attention did have some effect on which nucleotides or amino acids were predicted, but the output sequences were only slightly more diverse, and did not approach the ground-truth nucleotide distribution (Figure 3).

We conclude that LSTMs are not suitable to model the relation between promoters and proteins, but we cannot provide a clear explanation for this.

Table 5: Number of times the motif occurred in generated promoter sequences. For explanation of the datasets, see Section 2.1. Threshold means the proportion of bases that must match with the true motif. Only the GAN was tested on dataset 3 and 4, as the LSTM could already not find the pattern in dataset 2.

Threshold	Dataset	LSTM	ExpressionGAN
0.8	1	9.96	9.90
	2	0.02	1.06
	3	-	0.99
	4	-	0.24
0.9	1	9.96	9.90
	2	0.0	0.99
	3	-	0.88
	4	-	0.06
1.0	1	9.96	9.89
	2	0.0	0.99
	3	-	0.74
	4	-	0.003

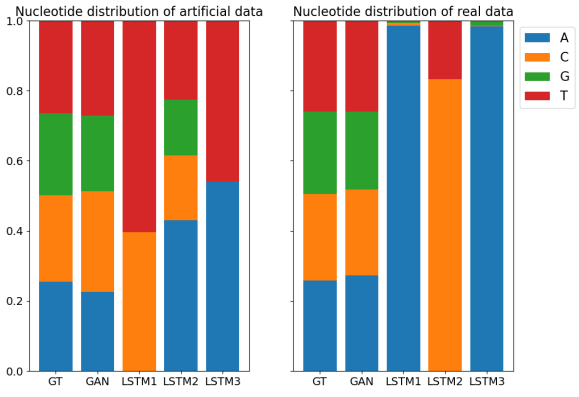


Figure 3: Nucleotide distribution of ground truth (GT) data vs. output of LSTM and ExpressionGAN. LSTM1 does not use frequency-aware cross entropy and attention, LSTM2 uses frequency-aware cross entropy but not attention, LSTM3 uses both frequency-aware cross entropy and attention.

3.3 GANs are a promising avenue for further research

Finally, we used ExpressionGAN to generate promoter sequences. In contrast to LSTMs, ExpressionGAN was able to model the artificial data well. The pattern with sequences repeated 10 times was easily found, as was the pattern with one motif per sequence at a fixed place (Table 5). Figure 4 displays the learning process over time (with scores as defined in Equation (3)). Performance on artificial datasets is visualised in Figure 4 (A, B). It appears that varying the location of a motif makes the pattern more difficult; dropping the number of times a motif is found on average from 100 times to 0.3 times per 100 sequences. In addition, the disciminator can separate generated from ground-truth sequences better in dataset 4 than in dataset 2 (Figure 4A, B). ExpressionGAN was also trained on dataset 3, which gave much better results.

A random background seems to have a large influence. To be able to quantify the performance of the generator, we also separately trained a discriminator to distinguish generated from ground-truth sequences. After training, the discriminator achieved a 100% test set accuracy on the dataset with motifs at variable locations. This discriminator also achieved a 70% test set accuracy on the dataset with motifs at fixed locations, while the generated sequences from this dataset contained 99 motifs for every 100 sequences. The latter result suggests that the discriminator is not deciding based on the motif; as the motif is generated at the correct location in almost every produced sequence. Rather, the discriminator is deciding based on the non-motif nucleotides. Thus, the non-motif nucleotides might not be random enough to fool the discriminator.

GANs may not be good at generating highly random outputs. In fields where GANs have been applied, structure is ubiquitous; images are highly structured, as are DNA sequences (even if that structure is highly complex). Predicting random sequences with one fixed pattern might be a conflicting goal for the network. The problem might simply be that GANs are not suited to this artificial task.

When trained on real data, we see that performance fluctuates more, but still gets quite close to 0 (Figure 4C, D). The fluctuations may be due to the heterogeneity of the dataset. A solution may be to condition the GAN on species; this might enable the GAN to learn which motifs are used in which species, thus

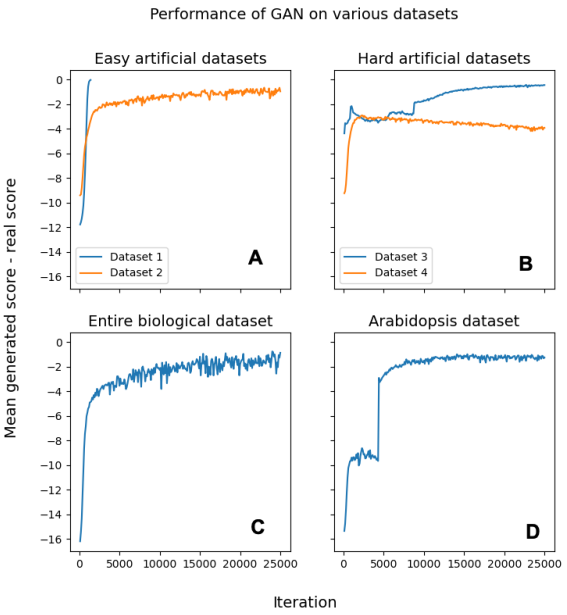


Figure 4: Performance of ExpressionGAN [23] on various datasets. A: performance on dataset 1 and 2; motif repeated 10x, and 1x motif at a fixed position. B: performance on dataset 3 and 4; 1x motif at a random position, non-motif bases are all A, and 1x motif at a random position, non-motif bases are random. C: performance on the full biological dataset. D: performance on a dataset with only *Arabidopsis thaliana* sequences. See Equation (3) for an explanation of how the scores work, and Section 2.1 for more in-depth explanation of the datasets.

improving the generated sequences. To investigate this, we trained a GAN on exclusively the *Arabidopsis* dataset. This improves convergence substantially (Figure 4D), and suggests cGANs may be worth exploring further.

To investigate biological relevance, we next analyse sequences for motifs; both biological *Arabidopsis* promoters and promoters generated by a model trained exclusively on *Arabidopsis*. The results are displayed in Figure 5. The number of significant matches in the generated dataset is approximately half that of in the biological promoters. Thus, the promoter sequences it generates truly have elements that correspond with biological promoter sequences. After 25,000 iterations, the motif distribution approaches that in the ground truth dataset quite well, although the counts are lower (approx. 9,000 motifs with $q < 0.05$ vs. approx. 5,500 in the generated dataset after 25,000 iterations). Interestingly, after 5,000 iterations we find over 80,000 motifs with $q < 0.05$. We analysed the generated datasets (Section S4), and found a certain group of motifs (DOF) that were significantly over-

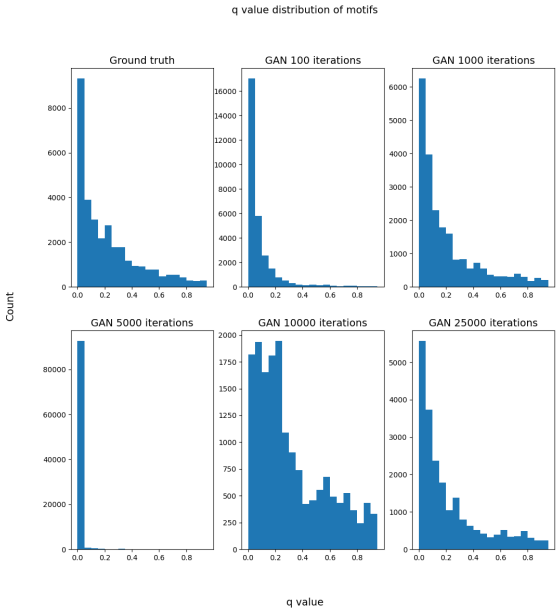


Figure 5: q value distribution of motifs found in biological *Arabidopsis* promoters vs. generated promoters at several time points.

represented in this generated set, at 50% vs. 15.7% in the ground truth data. Apparently, the model temporarily overfitted on this group of motifs. It seems prudent to include some additional measure of sequence quality to better assess model performance in the future. Options could be the entropy of generated motifs and the similarity of generated sequences. Summarizing, ExpressionGAN is a promising method to generate promoter sequences, and modifying this model to be conditional is likely to lead to even better results.

4 Conclusion and future outlook

In this paper, we aimed to investigate whether there is a relation between promoters and proteins, and the viability of applying deep neural networks to model this relation. We did this by first using DeePromoter, DNABERT and a CNN to classify protein domains based on promoter sequences. These results support the hypothesis that there is a relation between promoters and proteins, as they are able to do so with better than random performance. Next, we explored the possibilities further by applying an LSTM to translate promoter sequences to protein sequences and vice versa. However, even when combined with attention and frequency-aware cross entropy (FACE), they are not able to perform this task well. Finally, we used a GAN to generate promoter sequences with a nucleotide distribution that is highly similar to biological sequences, and contain many transcription factor binding site motifs. Thus, neural networks can model the relation between promoters and proteins, and there are many promising avenues for future study.

A number of interesting avenues for further research are available. DNABERT did not perform much better than other networks here, but we suggest several ways to further explore DNABERT's possibilities. DNABERT may be combined with other models to translate promoter sequences to protein sequences, for example by adding an output layer such as in the generator of the GAN used in this paper, or by using its output as input to an LSTM decoder. Another way to improve the performance of DNABERT could be to provide the species as input. It may also help to pretrain DNABERT on DNA of multiple different species, rather than just the human genome. This would almost certainly improve performance, but would also require a lot of compute. GANs were another method that seemed very promising; we consider it worthwhile to explore conditional GANs. This conditional GAN could be given information on the species and protein features such as domains or even the sequence itself. In this way, a model might be developed that could be used to generate fitting promoter sequences for synthetic proteins.

Data availability

The UniProt ID mapping file is available at their FTP server. The InterPro database of all UniProtKB proteins with corresponding InterPro entries is available here. EPD does not have an FTP server. Instead, the dataset was downloaded for each species individually from their website. The GenBank data for the corresponding species can be downloaded from their FTP server. DNABERT was downloaded and implemented through Hugging Face transformers, the Hugging Face page of DNABERT can be found here.

Code availability

All code used in this project is available at https://github.com/robertdvdk/Thesis.

Acknowledgements

I would like to thank Aalt-Jan van Dijk and Dick de Ridder for their guidance and frequent assistance, my buddy group Bogosort for the fruitful discussions, and the WUR Bioinformatics group as a whole for their hospitality. I also thank Ricky Siebeler, Freek Nijweide, and Jan Dirk van der Klis for their suggestions and support.

References

[1] R. Andersson and A. Sandelin, "Determinants of enhancer and promoter activities of regulatory elements," *Nature Reviews Genetics*, vol. 21, no. 2, pp. 71–87, Feb. 2020. DOI: 10.1038/s41576-019-0173-8. [Online]. Available: https://www.nature.com/articles/s41576-019-0173-8 (visited on 11/15/2022).

[2] R. C. Brewster, D. L. Jones, and R. Phillips, "Tuning promoter strength through RNA polymerase binding site design in *Escherichia coli*," *PLOS Computational Biology*, vol. 8, e1002811, 12 Dec. 2012. DOI: 10.1371/journal.pcbi.1002811. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1002811.

[3] P. D'haeseleer, "What are DNA sequence motifs?" *Nature Biotechnology*, vol. 24, pp. 423–425, 4 2006. DOI: 10.1038/nbt0406-423. [Online]. Available: https://doi.org/10.1038/nbt0406-423.

[4] S. A. Lambert, A. Jolma, L. F. Campitelli, P. K. Das, Y. Yin, M. Albu, X. Chen, J. Taipale, T. R. Hughes, and M. T. Weirauch, "The human transcription factors," *Cell*, vol. 172, pp. 650–665, 4 Feb. 2018. DOI: 10.1016/j.cell.2018.01.029. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0092867418301065.

[5] Z. Tianyin, S. Ning, Y. Lin, A. Namiko, H. John, M. R. S, B. H. J, G. Raluca, and R. Remo, "Quantitative modeling of transcription factor binding specificities using dna shape," *Proceedings of the National Academy of Sciences*, vol. 112, pp. 4654–4659, 15 Apr. 2015. DOI: 10.1073/pnas.1422023112. [Online]. Available: https://doi.org/10.1073/pnas.1422023112.

[6] T. Juven-Gershon, S. Cheng, and J. T. Kadonaga, "Rational design of a super core promoter that enhances gene expression," *Nature Methods*, vol. 3, pp. 917–922, 11 Nov. 2006. DOI: 10.1038/nmeth937. [Online]. Available: https://doi.org/10.1038/nmeth937.

[7] E. J. B. Williams and D. J. Bowles, "Coexpression of neighboring genes in the genome of arabidopsis thaliana," *Genome research*, vol. 14, pp. 1060–1067, 6 Jun. 2004. DOI: 10.1101/gr.2131104. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC419784/.

[8] D. J. Allocco, I. S. Kohane, and A. J. Butte, "Quantifying the relationship between co-expression, co-regulation and gene function," *BMC Bioinformatics*, vol. 5, p. 18, 1 2004. DOI: 10.1186/1471-2105-5-18. [Online]. Available: https://doi.org/10.1186/1471-2105-5-18.

[9] E. J. Klok, I. W. Wilson, D. Wilson, S. C. Chapman, R. M. Ewing, S. C. Somerville, W. J. Peacock, R. Dolferus, and E. S. Dennis, "Expression profile analysis of the low-oxygen response in Arabidopsis root cultures," *The Plant Cell*, vol. 14, pp. 2481–2494, 10 Oct. 2002. DOI: 10.1105/tpc.004747. [Online]. Available: https://doi.org/10.1105/tpc.004747.

[10] J. C. Walker, E. A. Howard, E. S. Dennis, and W. J. Peacock, "DNA sequences required for anaerobic expression of the maize alcohol dehydrogenase 1 gene," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 84, pp. 6624–6628, 19 Oct. 1987. DOI: 10.1073/pnas.84.19.6624. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/16578816.

[11] D. A. Rice, A. R. Mouw, A. M. Bogerd, and K. L. Parker, "A shared promoter element regulates the expression of three steridogenic enzymes," *Molecular Endocrinology*, vol. 5, pp. 1552–1561, 10 Oct. 1991. DOI: 10.

1210/mend-5-10-1552. [Online]. Available: https://doi.org/10.1210/mend-5-10-1552.

[12] J. Zrimec, C. S. Börlin, F. Buric, A. S. Muhammad, R. Chen, V. Siewers, V. Verendel, J. Nielsen, M. Töpel, and A. Zelezniak, "Deep learning suggests that gene expression is encoded in all parts of a co-evolving interacting gene regulatory structure," *Nature Communications*, vol. 11, p. 6141, 1 2020. DOI: 10.1038/s41467-020-19921-4. [Online]. Available: https://doi.org/10.1038/s41467-020-19921-4.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 7553 2015. DOI: 10.1038/nature14539. [Online]. Available: https://doi.org/10.1038/nature14539.

[14] Y. Qian, Y. Zhang, B. Guo, S. Ye, Y. Wu, and J. Zhang, "An improved promoter recognition model using convolutional neural network," *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, pp. 471–476, 2018. DOI: 10.1109/COMPSAC.2018.00072. [Online]. Available: doi:10.1109/COMPSAC.2018.00072.

[15] M. Oubounyt, Z. Louadi, H. Tayara, and K. T. Chong, "DeePromoter: Robust promoter predictor using deep learning," *Frontiers in Genetics*, vol. 10, 2019. DOI: 10.3389/fgene.2019.00286. [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2019.00286.

[16] M. Shujaat, A. Wahab, H. Tayara, and K. T. Chong, "pcPromoter-CNN: A CNN-based prediction and classification of promoters," *Genes*, vol. 11, p. 1529, 12 Dec. 2020. DOI: 10.3390/genes11121529. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/33371507.

[17] Ž. Avsec, M. Weilert, A. Shrikumar, S. Krueger, A. Alexandari, K. Dalal, R. Fropf, C. McAnany, J. Gagneur, A. Kundaje, and J. Zeitlinger, "Base-resolution models of transcription-factor binding reveal soft motif syntax," *Nature Genetics*, vol. 53, pp. 354–366, 3 Mar. 2021. DOI: 10.1038/s41588-021-00782-6. [Online]. Available: https://doi.org/10.1038/s41588-021-00782-6.

[18] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri, "DNABERT: Pre-trained bidirectional encoder representations from transformers model for DNA-language in genome," *Bioinformatics*, vol. 37, pp. 2112–2120, 15 Aug. 2021. DOI: 10.1093/bioinformatics/btab083. [Online]. Available: https://doi.org/10.1093/bioinformatics/btab083.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762. [Online]. Available: https://arxiv.org/abs/1706.03762.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805.

[21] R. Alexander, M. Joshua, S. Tom, G. Siddharth, L. Zeming, L. Jason, G. Demi, O. Myle, Z. C. Lawrence, M. Jerry, and F. Rob, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *Proceedings of the National Academy of Sciences*, vol. 118, e2016239118, 15 Apr. 2021. DOI: 10.1073/pnas.2016239118. [Online]. Available: https://doi.org/10.1073/pnas.2016239118.

[22] L. Hoegen Dijkhof, A.-J. van Dijk, and D. de Ridder, "Learning prokaryote promoter-protein grammar via deep learning," *MSc Thesis Wageningen University & Research*, 2022.

[23] J. Zrimec, X. Fu, A. S. Muhammad, C. Skrekas, V. Jauniskis, N. K. Speicher, C. S. Börlin, V. Verendel, M. H. Chehreghani, D. Dubhashi, V. Siewers, F. David, J. Nielsen, and A. Zelezniak, "Controlling gene expression with deep generative design of regulatory DNA," *Nature Communications*, vol. 13, no. 1, p. 5099, Aug. 2022. DOI: 10.1038/s41467-022-32818-8. [Online]. Available: https://www.nature.com/articles/s41467-022-32818-8 (visited on 10/05/2022).

[24] R. Dreos, G. Ambrosini, R. C. Périer, and P. Bucher, "EPD and EPDnew, high-quality promoter resources in the next-generation sequencing era," *Nucleic Acids Research*, vol. 41, pp. D157–D164, D1 Jan. 2013. DOI: 10.1093/nar/gks1233. [Online]. Available: https://doi.org/10.1093/nar/gks1233.

[25] UniProt Consortium, "UniProt: The universal protein knowledgebase in 2021," *Nucleic Acids Research*, vol. 49, no. D1, pp. D480–D489, Jan. 2021. DOI: 10.1093/nar/gkaa1100.

[26] M. Blum, H.-Y. Chang, S. Chuguransky, T. Grego, S. Kandasaamy, A. Mitchell, G. Nuka, T. Paysan-Lafosse, M. Qureshi, S. Raj, L. Richardson, G. A. Salazar, L. Williams, P. Bork, A. Bridge, J. Gough, D. H. Haft, I. Letunic, A. Marchler-Bauer, H. Mi, D. A. Natale, M. Necci, C. A. Orengo, A. P. Pandurangan, C. Rivoire, C. J. A. Sigrist, I. Sillitoe, N. Thanki, P. D. Thomas, S. C. E. Tosatto, C. H. Wu, A. Bateman, and R. D. Finn, "The InterPro protein families and domains database: 20 years on," *Nucleic Acids Research*, vol. 49, no. D1, pp. D344–D354, Jan. 2021. DOI: 10.1093/nar/gkaa977. [Online]. Available: https://doi.org/10.1093/nar/gkaa977 (visited on 10/05/2022).

[27] K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers, "GenBank," *Nucleic Acids Research*, vol. 44, no. Database issue, pp. D67–D72, Jan. 2016. DOI: 10.1093/nar/gkv1276. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4702903/ (visited on 10/05/2022).

[28] J. F. Kugel and J. A. Goodrich, "Finding the start site: Redefining the human initiator element," *Genes & Development*, vol. 31, no. 1, pp. 1–2, Jan. 2017. DOI: 10.1101/gad.295980.117. [Online]. Available: http://genesdev.cshlp.org/content/31/1/1 (visited on 10/31/2022).

[29] I. Letunic, R. R. Copley, S. Schmidt, F. D. Ciccarelli, T. Doerks, J. Schultz, C. P. Ponting, and P. Bork, "SMART 4.0: Towards genomic data integration," *Nucleic Acids Research*, vol. 32, no. suppl_1, pp. D142–D144, Jan. 2004. DOI: 10.1093/nar/gkh088. [Online]. Available: https://doi.org/10.1093/nar/gkh088 (visited on 10/05/2022).

[30] R. D. Finn, A. Bateman, J. Clements, P. Coggill, R. Y. Eberhardt, S. R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E. L. L. Sonnhammer, J. Tate, and M. Punta, "Pfam: The protein families database," *Nucleic Acids Research*, vol. 42, no. Database issue, pp. D222–D230, Jan. 2014. DOI: 10.1093/nar/gkt1223. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3965110/ (visited on 10/05/2022).

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Networks*, Jun. 2014. DOI: 10.48550/arXiv.1406.2661. [Online]. Available: http://arxiv.org/abs/1406.2661 (visited on 10/05/2022).

[32] M. Mirza and S. Osindero, *Conditional Generative Adversarial Nets*, Nov. 2014. DOI: 10.48550/arXiv.1411.1784. [Online]. Available: http://arxiv.org/abs/1411.1784 (visited on 10/05/2022).

[33] N. V. Bac, *Deepromoter implementation*. [Online]. Available: https://github.com/egochao/DeePromoter (visited on 11/06/2022).

[34] J. Zrimec, *Expressiongan*. [Online]. Available: https://github.com/JanZrimec/ExpressionGAN (visited on 11/06/2022).

[35] C. E. Grant, T. L. Bailey, and W. S. Noble, "FIMO: Scanning for occurrences of a given motif," *Bioinformatics*, vol. 27, no. 7, pp. 1017–1018, Apr. 2011. DOI: 10.1093/bioinformatics/btr064. [Online]. Available: https://doi.org/10.1093/bioinformatics/btr064 (visited on 10/20/2022).

[36] T. L. Bailey, J. Johnson, C. E. Grant, and W. S. Noble, "The MEME Suite," *Nucleic Acids Research*, vol. 43, no. W1, W39–W49, Jul. 2015. DOI: 10.1093/nar/gkv416. [Online]. Available: https://doi.org/10.1093/nar/gkv416 (visited on 10/20/2022).

[37] J. A. Castro-Mondragon, R. Riudavets-Puig, I. Rauluseviciute, R. Berhanu Lemma, L. Turchi, R. Blanc-Mathieu, J. Lucas, P. Boddie, A. Khan, N. Manosalva Pérez, O. Fornes, T. Y. Leung, A. Aguirre, F. Hammal, D. Schmelter, D. Baranasic, B. Ballester, A. Sandelin, B. Lenhard, K. Vandepoele, W. W. Wasserman, F. Parcy, and A. Mathelier, "JASPAR 2022: The 9th release of the open-access database of transcription factor binding profiles," *Nucleic Acids Research*, vol. 50, no. D1, pp. D165–D173, Jan. 2022. DOI: 10.1093/nar/gkab1113. [Online]. Available: https://doi.org/10.1093/nar/gkab1113 (visited on 10/20/2022).

[38] N. Q. K. Le, E. K. Y. Yapp, N. Nagasundaram, and H.-Y. Yeh, "Classifying Promoters by Interpreting the Hidden Information of DNA Sequences via Deep Learning and Combination of Continuous FastText N-Grams," *Frontiers in Bioengineering and Biotechnology*, vol. 7, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fbioe.2019.00305 (visited on 10/12/2022).

[39] K. R. Nitta, A. Jolma, Y. Yin, E. Morgunova, T. Kivioja, J. Akhtar, K. Hens, J. Toivonen, B. Deplancke, E. E. M. Furlong, and J. Taipale, "Conservation of transcription factor binding specificities across 600 million years of bilateria evolution," *eLife*, vol. 4, Mar. 2015. DOI: 10.7554/eLife.04837.

[40] S. Jiang, P. Ren, C. Monz, and M. de Rijke, "Improving neural response diversity with frequency-aware cross-entropy loss," in *The World Wide Web Conference on - WWW '19*, 2019, pp. 2879–2885. DOI: 10.1145/3308558.3313415. [Online]. Available: http://arxiv.org/abs/1902.09191 (visited on 10/05/2022).

[41] S. Robertson, *Nlp from scratch: Translation with a sequence to sequence network and attention*. [Online]. Available: https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html (visited on 11/06/2022).

[42] A. Galassi, M. Lippi, and P. Torroni, "Attention in Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4291–4308, Oct. 2021. DOI: 10.1109/TNNLS.2020.3019893. [Online]. Available: https://ieeexplore.ieee.org/document/9194070/ (visited on 11/19/2022).

Supplementary information

S1 Datasets

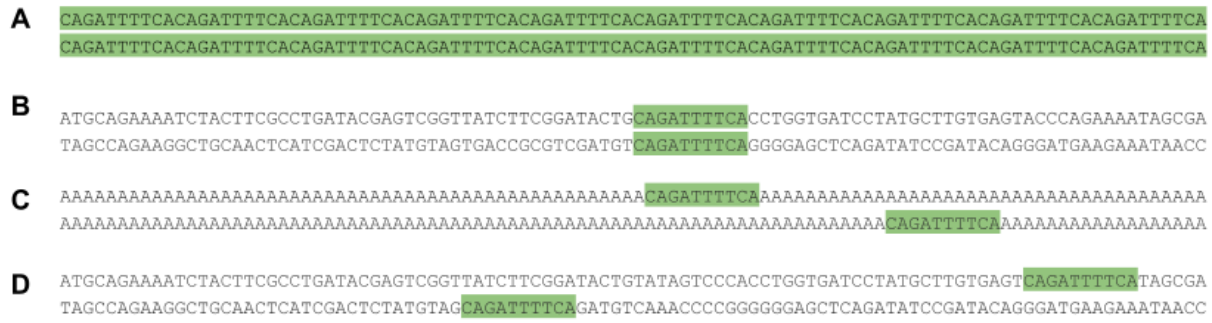


Figure S1: Example sequences from the artificial datasets. Motifs are highlighted in green. A: 10x repeated motif. B: 1 motif at a fixed place. C: 1 motif at a variable place, with the background consisting of all As. D: 1 motif at a variable place, with a random background.

S2 Network Architectures

The architecture of the CNN we built to compare to DeePromoter and DNABERT is shown in Figure S2. Let the length of one sequence be n, and the number of domains in the dataset be d. As we used one dataset from TSS-1000 to TSS+250, and one from TSS-200 to TSS, we have $n \in \{200, 1250\}$. DNA sequences of length n were converted to 5 one-hot vectors of length n, one each for A, C, G, T, and N. Then, a 1d convolutional layer with 5 input channels, 11 output channels, a kernel size of 5 and a stride of 1 transformed this input to 11 vectors of length n. This layer was then followed by batch normalization, a ReLU activation layer, and a 1d average pool with kernel size and stride 2. The vectors were subsequently flattened to form one long sequence of length $5.5n$. A linear layer used these as inputs to produce 50 outputs, after which the next linear layer used these to produce 20 outputs. The last layer used these 20 neurons as inputs to produce d outputs. All linear layers were followed by batch normalization, dropout, and a ReLU activation layer. Parameters were optimized using stochastic gradient descent with a learning rate of 10^{-2}. We used the loss function as defined in the explanation of the DeePromoter architecture.

The LSTM encoder-decoder we implemented was heavily inspired by Robertson [41]. The data were first tokenized (including start of sentence and end of sentence tokens), resulting in a mapping of characters to tokens (e.g. SOS = 0, EOS = 1, A = 2, C = 3, ...). In this fashion, sequences were processed to tensors of tokens. We then embed the input sequence to a length h sequence. We use an encoder and decoder that generate hidden states with h features. We tried several different values for h to determine the optimal network complexity; we used $h \in \{128, 512, 2048\}$. In the decoder, a ReLU activation layer was added between embedding and passing the sequences through the LSTM. We used a teacher forcing ratio of 0.5. As for the encoder and decoder with attention, the attention function and dropout ($p = 0.1$) were applied directly after the embedding. Attention is a concept that allows each step in the decoder to take a weighted combination of encoder hidden states as input. Intuitively, this allows a neural network to give more importance to certain parts of the input. We refer interested readers to [42]. We used two different loss functions for training the LSTMs: cross entropy loss, and frequency-aware cross entropy (FACE). FACE was introduced by Jiang *et al.* [40] to account for certain tokens being far less frequent than others when training Seq2Seq models (the *low-diversity problem*), and we implemented this by dividing each individual token's loss by its proportion of the total tokens in the dataset. This resulted in individual mistakes for low-frequency tokens contributing far more to the total loss than individual mistakes for high-frequency

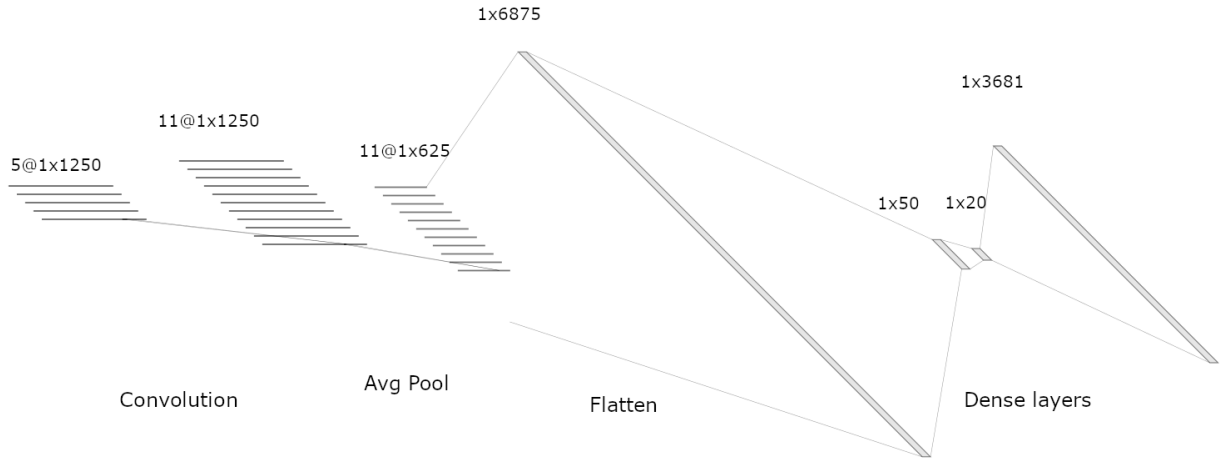


Figure S2: Schematic overview of the architecture of the CNN used to classify protein domains based on promoter sequences.

tokens. Stochastic gradient descent was used to optimize the parameters, with a learning rate of 10^{-3}.

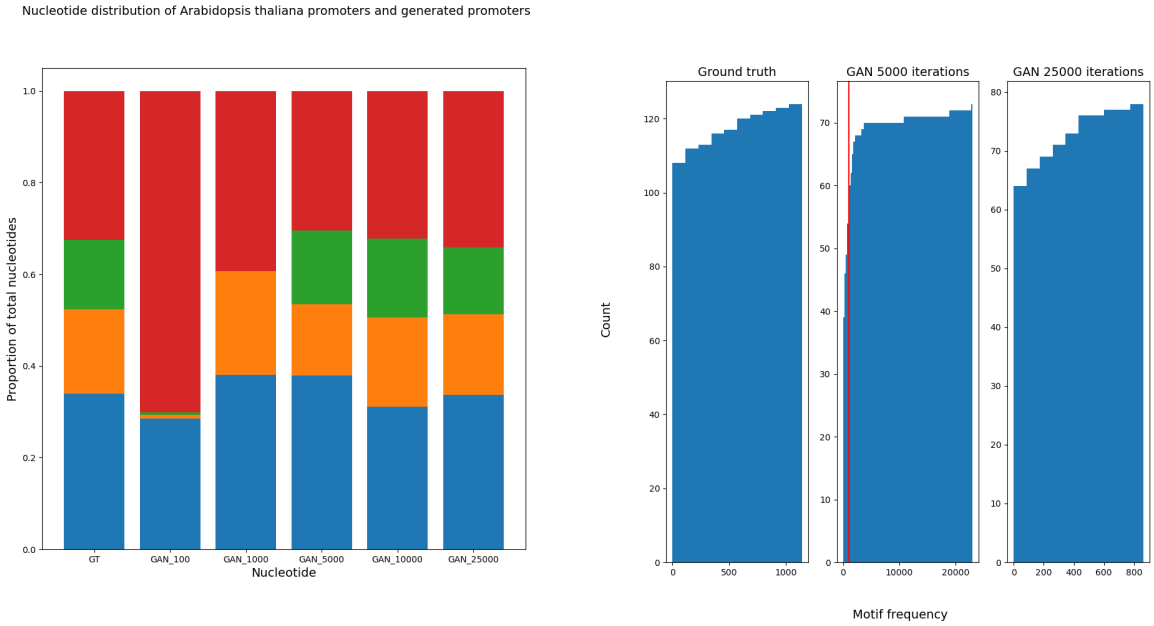
S3 CNN results

Table S1: Performance statistics of own network on biological dataset containing promoter sequences with domains of corresponding proteins as labels

Included nucleotides counted from TSS	# datapoints	# unique domains	Accuracy	Precision	Recall	Specificity
-1000 - +250	95,001	3,681	0.788	0.001	0.484	0.788
-1000 - +250	45,861	238	0.779	0.010	0.434	0.781
-1000 - +250	4,306	2	0.664	0.657	0.688	0.641
-200 - 0	95,001	3,681	0.786	0.001	0.646	0.786
-200 0	45,861	238	0.698	0.008	0.494	0.699
-200 - 0	4,306	2	0.617	0.612	0.641	0.594

S4 Outlier analysis

We analysed the sequences generated by the GAN after 5,000 iterations (hereafter referred to as GAN 5,000) in several different ways. We first looked at the nucleotide distribution of the ground truth and generated sequences, to determine whether the sequences generated by the GAN 5,000 had low diversity (Figure S3a). However, this appeared not to be the case; training the GAN longer appears to result in generated sequences that better approximate the ground truth nucleotide distribution, but the difference is only a few per cent. Next, we tried to determine whether the network generated one or a few motifs extremely frequently, thus overfitting on one or a few motifs. We created a cumulative histogram, where the x axis shows how frequently a motif occurred in the dataset, and the y axis shows how many motifs occurred x or fewer times in the dataset (Figure S3b). For example, the ground truth dataset had 105 motifs that occurred 100 times or fewer, 110 motifs that occurred 200 times or fewer, etc. In the plot for the GAN 5,000, we put a vertical line at x=1,000, as the maximum count for any one motif was approximately 1,000 for both the ground truth and



(a) Nucleotide distribution of ground truth vs. generated sequences

(b) Cumulative motif frequency histogram.

Figure S3: Comparison of sequences generated by ExpressionGAN at several different time points during training.

the GAN 25,000 datasets. Judging from the plot, however, it does not look like the GAN 5,000 overfitted on a small number of motifs; there are approximately 15 out of 73 motifs that occur 1,000 or more times in the dataset. In short, apart from the fact that DOF motifs are heavily overrepresented, we could not find an obvious reason for why the sequences generated by the GAN 5,000 had so many more motifs with $q < 0.05$.