Building the Gene Regulatory Network Using Deep Learning

Ashwini Suriyaprakash
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
ashwinis@mit.edu

Abstract

Understanding the interactions between regulatory elements and target genes within gene regulatory networks has wide applications in human health, especially in disease diagnosis and proactive clinical treatment. In this work, we propose TRNReg, a transformer-based deep learning approach that links regulatory elements with target genes. Through a self-attention mechanism, TRNReg identifies relationships between genomic regions, determines the relative impact of one region on another, and ultimately learns a robust, context-based representation of each region. TRNReg exceeded the baseline model in terms of accuracy, achieving 21% higher Pearson correlation coefficient during training and 1.5% higher correlation during testing. Furthermore, the biological interactions that can be discovered by TRNReg serve as a stepping stone for advancing the current knowledge of gene regulation.

1. Introduction

Gene regulatory networks (GRNs) are complex biological networks that govern the level to which a target gene is expressed as protein. Abstractly, a GRN can be visualized as a graph, in which nodes are either (1) target genes or (2) regulatory elements, DNA regions which upregulate or downregulate the expression of a target gene. In this graph, every edge between a regulatory element E and target gene G represents their interaction and can be characterized in one of two ways: either 'E upregulates G' or 'E downregulates G'. Comprehensively identifying the interactions within GRNs is critical to understanding disease since most disease-related mutations occur in regulatory elements and other non-coding DNA regions, and cause target genes to be expressed in an incorrect amount. In these situations, deeper insight into GRNs can allow clinicians to efficiently identify the mutated regulatory element and correct its behavior through treatments, such as drugs or gene-editing.

To accelerate the discovery of interactions within GRNs, recent machine-learning based approaches, including CellOracle, FigR, and GraphReg, have been developed and evaluated. These computational methods have been effective compared to traditional laboratory approaches since they explore the vast space of potential interactions more efficiently and narrow down this space into a small subset that can subsequently be tested in the laboratory. These methods leverage the availability of large public genomic datasets that characterize both DNA structure and gene expression of sequenced cells. However, these approaches have drawbacks. While CellOracle utilizes DNA structure and gene expression data independently instead of analyzing their correlation [4], FigR discovers new insights, only within human peripheral blood mononuclear cells, a specific type of immune cell [6]. GraphReg overcomes the limitations of CellOracle and FigR by not only considering DNA structure and gene expression in tandem but by also generalizing to different cell types. GraphReg links regulatory elements with target genes through a graph neural network-based method [1]. However, the quality of its predictions can be further improved by modifying this architecture.

In this work, we develop and present TRNReg, a transformer-based method that links regulatory elements with target genes. Improving upon GraphReg, TRNReg enhances prediction accuracy by replacing GraphReg's convolutional neural network (CNN) with a transformer, an increasingly popular deep learning architecture. The primary advantage of a transformer is that unlike CNNs which learn relationships between a genomic region and its direct neighbors, transformers learn relationships between a region and many other regions upstream and downstream. This advantage is important because the model learns a more robust representation of a genomic region since the biological behavior of a particular region can be influenced by other regions that are not proximal to it.

Similar to GraphReg, TRNReg processes epigenomic data, quantitatively describing DNA structure and conformation for a particular genomic region as input, in order

to predict that region's gene expression as output. The input data is forwarded to a transformer encoder, consisting of multiple Encoder layers, each with self-attention, layer normalization, and feed-forward network. The encoder learns context-dependent representations of genomic regions and passes these representations to subsequent steps in the pipeline, ultimately yielding an output describing the region's gene expression. This model was trained on genomic regions across eighteen chromosomes, validated on two chromosomes, and tested on two chromosomes in a human cell line. The performance of the model was evaluated by comparing Pearson correlation coefficients between the predicted output and ground-truth output. Because the original training data (approximately 1000 regions across 18 chromosomes) was not sufficient to improve training accuracy, several approaches were explored to artificially augment the size of the training set and are described in detail in the Methods section.

Compared to GraphReg, TRNReg achieved superior accuracy overall during both training and testing (20.98% higher Pearson correlation during training and 1.47% higher Pearson correlation during testing). Overall, TRNReg advances the current understanding of gene regulatory networks with the ultimate goal of improving disease diagnosis and treatment.

2. Methods

In the following sections, we delve into the TRNReg approach by providing an overview of the deep learning pipeline, describing the architecture details of the transformer encoder, and lastly covering model training and dataset augmentation.

2.1. Deep Learning Pipeline Overview

TRNReg improves upon an existing deep learning approach called GraphReg, which discovers interactions between genomic regulatory elements and target genes by leveraging graph neural networks. At a higher level, GraphReg's architecture (Fig.1) takes in three types of DNA epigenomic data as input (DNase-seq as a proxy for chromatin accessibility, H3K4me3 ChIP-seq for promoter activity, and H3K27ac ChIP-seq for enhancer activity). This data is first passed through a CNN, which learns a lower-dimensional representation of every input genomic region. Subsequently, these representations are then fed to a graph attention neural network, which leverages 3D interaction graphs to learn a context-based lower-dimensional representation of each genomic region. Finally, this representation is passed through a few convolutional layers to ultimately predict the gene expression (CAGE-seq) of that region.

Building off of GraphReg, TRNReg (Fig. 2) seeks to improve prediction accuracy by replacing the first CNN block

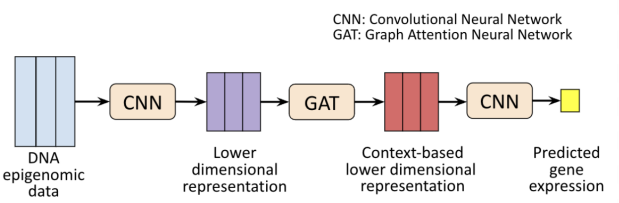

Figure 1. Overview of the Epi-GraphReg Model. Input DNA epigenomic data is processed through several neural network modules to predict target gene expression.

of GraphReg with a transformer encoder. Unlike CNNs which learn relationships between a genomic region and its direct neighbors, Transformers learn relationships between a region and many other regions upstream and downstream via self-attention, namely by computing attention scores between a region and every other region.

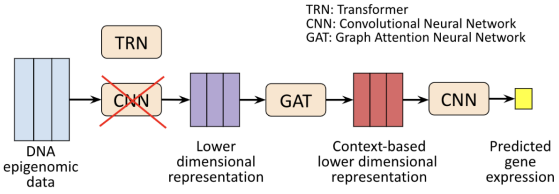

Figure 2. Overview of the TRNReg Model. In the TRNReg architecture, the first CNN module of GraphReg is replaced with a Transformer.

2.2. Transformer Architecture

The transformer in TRNReg mimics the behavior of the original transformer encoder described in Google's paper and was implemented using Python's Keras library [3]. The input to the transformer is a 6-megabase (Mb) genomic region split into smaller 5-kilobase (kb) regions, which are passed as input tokens to the encoder consisting of a stack of encoder layers. Each encoder layer consists of a multi-head attention layer, residual connection, layer normalization, fully-connected feed forward network, residual connection, and layer normalization as shown in Figure 3 below.

The number of encoder layers and attention heads per layer can be varied based on the user's preferences. The ultimate output of the encoder is a sequence of new tokens that are enhanced context-dependent representations of 5-kb regions. These representations are computed by analyzing relationships between a given token and other tokens in the sequence.

For better performance, a basic positional encoding with sines and cosines was applied to the input tokens prior to the encoder so that every token contains information about its position in the sequence.

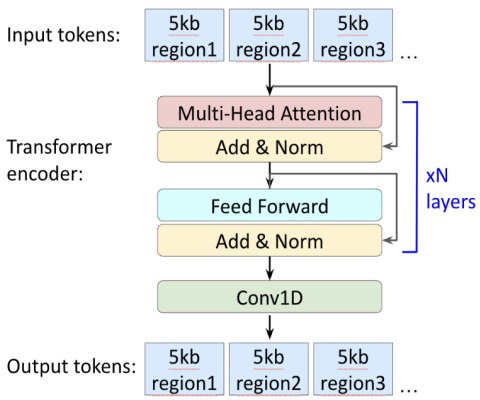

Figure 3. TRNReg Transformer Architecture. A sequence of input tokens is fed to the transformer encoder consisting of N encoder layers, ultimately yielding a sequence of tokens as output.

2.3. Model Training

TRNReg was trained, validated, and tested on epigenomic data belonging to a human cell line (K562) in the hg19 reference genome build. The model was validated on chromosomes 3 and 13, tested on chromosomes 4 and 14, and trained on all other chromosomes except the sex chromosomes X and Y. Every 6-Mb input to the transformer encoder was generated by starting at the beginning of the chromosome and shifting by 2 Mb to attain the next 6-Mb region. The total number of 6-Mb regions in the training set was approximately 1000, which is low especially for a transformer. This data deficiency will be addressed in the next section. Additionally, the model was trained for 30 epochs with the Adam optimizer, and the learning rate was varied with values 0.0002, 0.0001, 0.00001, and 0.000001.

2.4. Boosting Training Set Size

As mentioned in the previous section, the current training set size is inadequate for training a transformer, which requires large amounts of data (on the order of tens of thousands) to learn better local representations. Two approaches were explored to address this problem, namely increasing overlap between consecutive training data points and decreasing the size of each input region. Each of these approaches is described in more detail below.

1. Increasing overlap: To augment training size, we increased the overlap between consecutive training data points. This technique was drawn from a data augmentation approach used to boost the size of a dataset containing audio signals. In this work, the authors shifted every audio by a certain time interval to generate additional training samples [7]. Applying a similar approach to this problem, we attained the next training region input by shifting the current region by 200,000 bases instead of 2Mb (10x smaller number of bases), corresponding to a shift factor of 0.1. This resulted in a 10x larger training set. Multiple shift factors, including 0.1, 0.17, and 0.23, were tested for comparison, deliberately chosen to avoid redundancy in the training data across shift factors.

2. Decreasing input region size: As mentioned in the previous sections, every input fed to the transformer is a 6-Mb genomic region. With this size, the number of such regions within every chromosome ranged from a minimum of 18 regions (Chromosome 22) to a maximum of 121 regions (Chromosome 2). Since the training set comprised 18 chromosomes, the total number of training regions was approximately 1000. This was far too small for training. To address this issue, every chromosome was diced into regions that were 10x smaller (600000-base regions, rather than 6-Mb regions), resulting in a 10x larger training set.

3. Results

We demonstrate that TRNReg surpasses GraphReg in terms of prediction accuracy. As prior research noted, the transformer encoder was expected to perform better than a CNN because it learns relationships between a genomic region and many other genomic regions that are both proximal and distant [5]. This allows transformers to discover more robust representations of a given genomic region. In the following sections, we discuss the data involved in training TRNReg, describe the tested model hyperparameters, evaluate the performance of different data boosting approaches, and visualize the gene expression output.

3.1. Dataset for Training TRNReg

To compare the performance of TRNReg with that of GraphReg, we controlled for the input dataset used to train both models. Like GraphReg, TRNReg was trained on 1,090 6-Mb regions spanning 18 chromosomes, validated on 2 chromosomes, and tested on 2 chromosomes.

TRNReg processes three types of DNA epigenomic data as input (DNase-seq as a proxy for chromatin accessibility, H3K4me3 ChIP-seq for promoter activity, and H3K27ac ChIP-seq for enhancer activity). These publicly available datasets were downloaded from The Encyclopedia of DNA Elements (ENCODE) in bigwig format [1]. Every input to the transformer represented a 6-Mb region, split into a sequence of tokens, and the transformer learned a context-dependent representation of these tokens. This robust representation was then passed to subsequent steps in the pipeline, ultimately yielding a prediction of the CAGE gene expression output. To evaluate the model's performance, this prediction was quantitatively compared with the ground-truth gene expression using the Pearson correlation coefficient, which was averaged across all data inputs. This

average correlation was computed for both GraphReg (the baseline) and TRNReg to determine whether TRNReg's architecture yielded more accurate predictions.

3.2. TRNReg Model Hyperparameters

To achieve optimal performance, several hyperparameters were tested, including number of attention heads, number of attention layers, number of epochs, learning rate, and presence of an absolute positional encoding described in Table 1. The positional encoding marks every input token with its position in the sequence using sines and cosines, such that proximal tokens have similar positional encodings [2].

Hyperparameter	Tested Values
Number of attention heads	4, 8
Number of attention layers	1, 2, 4, 8, 16
Number of epochs	30
Learning rate	2×10^{-4}, 10^{-4}, 10^{-5}, 10^{-6}
Positional encoding	Present, Absent

Table 1. Tested Model Hyperparameters. The left column shows the hyperparameter, and the right column shows the various values of the hyperparameter tested.

Reducing the learning rate significantly to 10^{-6} yielded poor training accuracy at epoch 30 since convergence was slower. Furthermore, while TRNReg with positional encoding achieved similar training performance compared to TRNReg without positional encoding, it struggled to generalize to unseen data, performing worse during testing. Thus, for future sections of this paper, the optimal combination of hyperparameters used was number of attention heads = 4, number of attention layers = 1, number of epochs = 30, learning rate = 10^{-5}, and positional encoding = Absent.

3.3. Training Set Boosting Evaluation

As mentioned in the Methods section, transformers require on the order of tens of thousands of data points to be trained effectively. For this purpose, two approaches were used to augment the training set size:

The first approach was increasing overlap between successive training set inputs. Instead of shifting the current genomic region by 2Mb to get the next training datapoint as GraphReg does, we performed a 10x smaller shift of 200,000 bases and consequently augmented the training set size by 10x. This corresponded to a shift factor of 0.1. We evaluated and compared multiple shift factors, including 0.1, 0.17, and 0.23.

The second approach was decreasing the size of the input region itself. While every data input to GraphReg represented a 6-Mb region, we instead attempted to feed TRNReg regions that were 10x smaller (600000-base regions). This augmented the training set size by 10x.

Both of these approaches were evaluated. Most notably, increasing overlap between successive training inputs (particularly, with shift factor = 0.17) yielded significant accuracy gains over GraphReg during both training and testing. Accuracy was measured in terms of Pearson correlation where higher correlation means higher accuracy. Figure 4 below shows the training Pearson correlation over 30 epochs for GraphReg and different variations of TRNReg, while Figure 5 shows the testing correlation. Note that all TRNReg variations use the following combination of hyperparameters: number of attention heads = 4, number of attention layers = 1, number of epochs = 30, learning rate = 10^{-5}, and positional encoding = Absent.

As shown in Figure 4, reducing the shift factor (increasing the overlap between successive training inputs) resulted in higher training accuracy/correlation at epoch 30, compared to TRNReg trained on the original non-boosted dataset (blue curve) and baseline GraphReg (dotted black curve). At epoch 30, the variation of TRNReg with shift factor 0.17 (green curve) achieved a training correlation of 0.684, which was 20.98% higher than GraphReg's training correlation of 0.565. However, decreasing input region size by 10x (purple curve) performed significantly worse compared to GraphReg and other TRNReg variations as it achieved the lowest correlation.

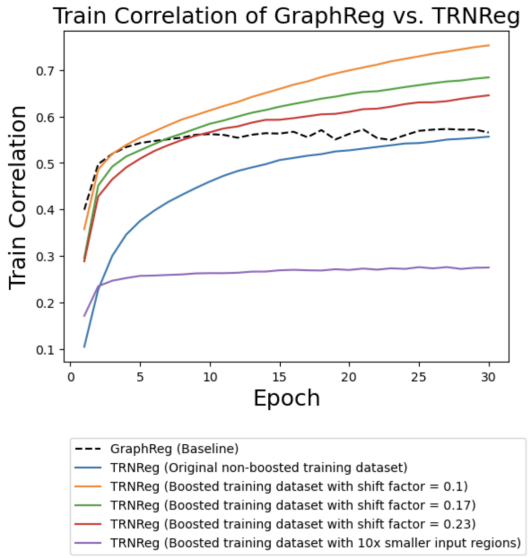

Figure 4. Correlation (Pearson correlation coefficient) over 30 epochs for GraphReg and different variations of TRNReg during training. The x-axis is the epoch number, and the y-axis is the correlation. The higher the correlation at epoch 30, the better. Reducing the shift factor resulted in higher correlation at epoch 30.

As shown in Figure 5, some TRNReg models achieved higher accuracy compared to GraphReg during testing. At epoch 30, the best-performing variation of TRNReg with

shift factor 0.17 (green curve) achieved a test correlation of 0.468, which was 1.47% higher than GraphReg's test correlation of 0.461.

For ease of interpretation, the final training and testing correlations for all models at epoch 30 are also presented non-graphically in Table 2.

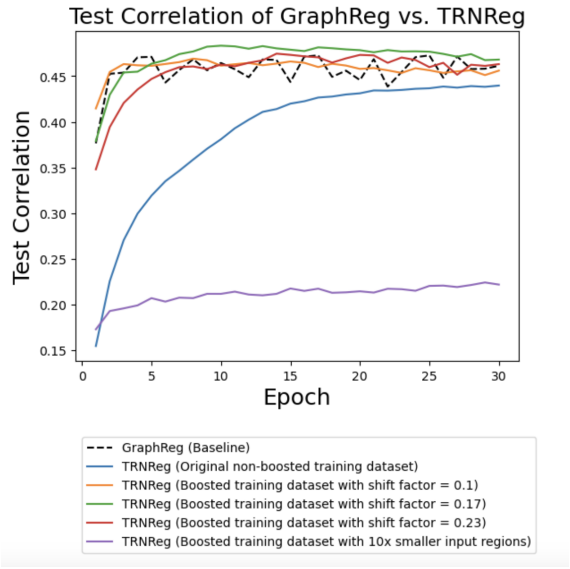

Figure 5. Correlation (Pearson correlation coefficient) over 30 epochs for GraphReg and different variations of TRNReg during testing. The x-axis is the epoch number, and the y-axis is the correlation. TRNReg with shift factor 0.17 (green curve) achieved the best correlation at epoch 30.

Model	Train Corr.	Test Corr.
GraphReg	**0.565**	**0.461**
TRNReg (original)	0.557	0.440
TRNReg (shift factor = 0.1)	0.753	0.456
TRNReg (shift factor = 0.17)	**0.684**	**0.468**
TRNReg (shift factor = 0.23)	**0.645**	**0.463**
TRNReg (10x small)	0.275	0.222

Table 2. Model Performance. For every model in the leftmost column, the corresponding final training and testing correlations at epoch 30 are displayed. TRNReg with shift factors of 0.17 (in green) and 0.23 (in red) surpassed GraphReg's correlation (bolded) during both training and testing, and 0.17 achieved higher correlation values of the two.

3.4. CAGE Expression Visualizations

For ease of comparison, the predicted CAGE gene expression output and ground-truth output were plotted together for all 5-kb regions in the training dataset in Figure 6 below. Every tick on the x-axis represents a single 5-kb region, and the corresponding y-axis value is the CAGE output for that 5-kb region. The solid red line denotes the CAGE value predicted by TRNReg, and the dotted black line denotes the ground-truth CAGE value. This visualization demonstrates that TRNReg's predictions reasonably match the ground-truth data.

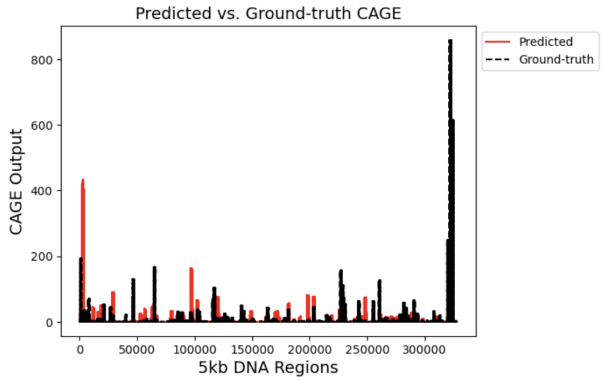

Figure 6. Predicted vs. Ground-truth CAGE output. The x-axis represents 5-kb regions, and the y-axis is the CAGE output for the corresponding region. Figure provides visual confirmation that CAGE output predicted by TRNReg reasonably matches the ground-truth value but with great differences at the edges.

4. Conclusion

Overall, TRNReg achieved superior training and testing accuracy compared to GraphReg. This result is promising because it strongly suggests that TRNReg can discover interactions between regulatory elements and target genes more accurately, compared to the state-of-the-art baseline GraphReg. The immediate next step is to use the pairwise attention weights learned by TRNReg to link regulatory elements with target genes and confirm the validity of these interactions with prior literature. In particular, we expect TRNReg to discover biologically-significant interactions, which GraphReg does not, and thus surpass GraphReg in advancing the current understanding of gene regulation.

In terms of other avenues for further exploration, it would be insightful to evaluate various publicly available transformer models beyond the vanilla transformer encoder. In addition, leveraging more datasets, such as those from different species, could help boost the training set size and may yield promising results, though doing this experiment relies on the assumption that GRNs are conserved across different species. Nevertheless, we demonstrate TRNReg's advantages over the state-of-the-art baseline and its implications in disease diagnostics and treatment.

5. Datasets

Epigenomic and gene expression data (CAGE-seq) for cell line K562 was downloaded from The Encyclopedia of DNA Elements (ENCODE). The 3D Hi-C chromatin interaction data for cell line K562 was downloaded from the NCBI Gene Expression Omnibus. Links to the data are listed below.

DNase-seq: encodeproject.org/files/ENCFF352SET
H3K4me3: encodeproject.org/files/ENCFF689TMV/
H3K27ac: encodeproject.org/files/ENCFF010PHG/
CAGE: encodeproject.org/files/ENCFF233CVF/
Hi-C: ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63525

References

[1] Alireza Karbalayghareh et al. Chromatin interaction aware gene regulatory modeling with graph attention networks. 2022. 1, 3

[2] Amirhossein Kazemnejad et al. The impact of positional encoding on length generalization in transformers. 2023. 4

[3] Ashish Vaswani et al. Attention is all you need. 2017. 2

[4] Kenji Kamimoto et al. Dissecting cell identity via network inference and in silico gene perturbation. 2023. 1

[5] Tianyang Lin et al. A survey of transformers. 2022. 3

[6] Vinay K. Kartha et al. Functional inference of gene regulation using single-cell multi-omics. 2022. 1

[7] Lucas Ferreira-Paiva. A survey of data augmentation for audio classification. 2022. 3