Learning Genome Architecture Using MSA Transformers

Emaad Khwaja [*][†]
UC Berkeley
Berkeley, CA 94702
emaad@berkeley.edu

Lucas Waldburger [*][‡]
UC Berkeley
Berkeley, CA 94702
lwaldburger@berkeley.edu

Abstract

Understanding genome architecture in organisms facilitates the study of gene functionality and mapping regulatory networks. Operons are a transcriptional unit that share a common regulatory mechanism and one of the most basic genome structures that are abundant in prokaryotic genomes. Traditionally, geneticists manually annotate operons by inspecting gene neighborhoods using gene-level and comparative genomic features. Existing computational methods that predict operons largely rely on gene-level features, but poorly integrate comparative genomic analysis. Recently, a deep learning model, OperonHunter, was developed that uses visual representation of genomic fragments attempts to further incorporate comparative genomic features. Since its development, transformers have gained attention for their superior performance in biological prediction tasks. Here we propose an operon MSA transformer to further improve the predictions on the operon detection task. Our hope is that a transformer can better capture genome architecture features than previous machine learning methods that detect operons.

1 Introduction

1.1 Operons are transcriptional units

Prokaryotic genomes contain gene clusters that share a common regulation mechanism in the form of an operon [1]. An operon is composed of a promoter, an operator, one or more contiguous genes in a common strand direction, and a terminator. These terms have been defined in Table 1. The genes in an operon often exhibit a similar metabolic or functional relationship and are co-transcribed on a polycistronic mRNA transcript.

One of the most widely studied operons is the *lac* operon in *Escherichia coli*, which controls the utilization of lactose (See Fig. 1). Lactose is a disaccharide composed of galactose and glucose subunits and its catabolism is disfavored in the presence of monosaccharides that are more easily

Table 1: Terminology used to describe common DNA sequences in an operon.

Term	Definition
Promoter	Site of transcription initiation
Operator	Site of transcription factor binding to exhibit control over gene expression
Gene	Sequence that encodes the synthesis of a mRNA transcript
Terminator	Site of transcription termination

[*]UC Berkeley - UCSF Joint Bioengineering Graduate Program
[†]Computer Science Division, UC Berkeley, CA 94720
[‡]Biological Systems & Engineering Division, Lawrence Berkeley National Laboratory, CA, 94720

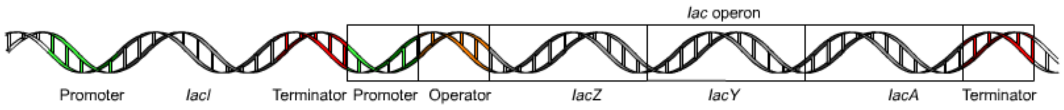

Figure 1: Genetic architecture of the *lac* operon required for the transport and metabolism of lactose in *E. coli*. Operons share transcriptional regulation and a common promoter and terminator.

converted into energy through central metabolism. For this reason, *E. coli* has evolved transcriptional control over sugar availability to maximize growth fitness in the presence of competitors. The *lac* operon contains three genes involved in breaking down lactose into its respective subunits and importing lactose from the extracellular environment. When lactose is present, the gene in the *lac* operon are constitutively expressed. In the presence of glucose, a transcription factor occludes the operator site thereby repressing genes in the *lac* operon. Since the 1960s when the lac operon was first identified, operons have been identified across the kingdoms of life. Although they are most present in bacterial and archaeal genomes, the operon represents one of the most ubiquitous genetic structures.

1.2 Motivation

Understanding the genome architecture in organisms facilitates the study of gene functionality and mapping regulatory networks [2]. Operon prediction represents a step toward understanding higher level gene organization and means to provide more complete gene annotations that can be applied towards identifying new antibiotics [3] and anti-phage defense systems such as CRISPR-Cas9 [4]. Geneticists annotate operons by using gene-level information such as intergenic distance, strand direction, gene size, and functional relatedness in the same genome. In addition, a comparative genomics approach can be used to consider gene neighborhood conservation in other related genomes. Operons can be experimentally validated to further understand regulation structure. However, manual operon validation is time-consuming since genomes are large and data analysis by humans is error-prone and often subjective. For example, the genome of *E. coli* consists of approximately 4,639,221 bases of DNA [5]. Scattered throughout this sequence are between 630 to 700 operons [6]. For this reason, there is a need for learning models that can accurately identify operons since the speed of new genomes being sequenced has rapidly outpaced the speed of functional characterization and experimental validation.

2 Related Work

2.1 Existing Methods

Existing computational methods used to predict operons largely focus on gene-level features. These methods include using Hidden Markov Models (HMMs) to identify shared promoter and terminator sequences flanking consecutive genes. Others use functional relatedness between genes in a genetic neighborhood. More advanced computational methods use machine learning to predict operons with architectures such as neural networks [7, 8]), support vector machines [9], decision-tree based classification [10], Bayesian probabilities [11, 12, 13], genetic algorithms [14], and graph-theoretical techniques [15, 12]. These tools have facilitated operon prediction by providing operon databases and web-based tools. However, many of tools that claim highest performance are no longer available online or have low-throughput web interfaces that limit their utility.

2.2 OperonHunter v1.1

A deep learning model, OperonHunter, was recently developed that uses visual representation of genomic fragments as an attempt to further incorporate comparative genomic features into operon prediction [2]. This method represents the state of the art in the operon prediction field. The authors train a visual representation learning model akin to how a geneticist would manually detect operon representations. They represented query genes as arrows with variable lengths corresponding to gene size, arrowhead directions corresponding to strand direction, arrow color corresponding to gene function class, and negative space between arrows corresponding to intergenic distance. Comparative genomic analysis is incorporated by aligning the query genes to close relatives of the query genome. The authors used a nuanced definition of an operon as containing two or more genes.

OperonHunter is trained on datasets from *E. coli* and *B. subtilis*, which are model prokaryotes with well-studied genomes and experimentally validated operons. The authors of OperonHunter compare their model to two existing methods claiming to have the highest accuracy. The first method, ProOPDB, uses an artificial neural network that predicts operons based on intergenic distance and gene functional relatedness [16]. Functional relatedness is calculated by scores in the STRING database Jensen et al. [17]. The STRING database captures functional relatedness through scores using information on gene neighborhoods, fusion, co-occurrence, co-expression, protein-protein interactions, and automatic literature mining. These data are compiled into the Prokaryotic Operon Database (ProOpDB) and as a web tool called Operon Mapper. Unfortunately, ProOpDB is no longer accessible online and using Operon Mapper is extremely low-throughput which is prohibitive to large phylogenetic analyses that rely on operon predictions. The second method, Door, uses a combination of linear- (decision tree) and nonlinear-based (logistic function) classifiers depending on the availability of experimentally validated operons for the query genome [18]. Door predicts operons based on intergenic distance, the presence of a DNA motif between consecutive genes, the ratio of the gene lengths and function similarity determined using Gene Ontology (GO), and the level of conservation of the genomic neighborhood.

Gene-pair prediction performance is determined by sensitivity (true positive rate), precision, and specificity (true negative rate). ProOpDB scored the highest sensitivity with the lowest precision while Door scored the lowest sensitivity and highest precision. OperonHunter performed more stably across these two metrics while scoring highest on accuracy, F1 score accuracy, and the Matthews Correlation Coefficient (MCC). On the full operon prediction task, all three tools performed substantially worse. OperonHunter had the highest performance of 85% exact operon matches to experimentally validated genetic boundaries. Therefore, when compared to other leading methods, OperonHunter was able to achieve top performance across established datasets (See Table 2), with markedly better sensitivity.

2.2.1 Dataset

Rather than using raw data as a source of inference, OperonHunter relies on data coming from PATRIC [19], a genetic visualization tool, to make predictions on whether grouped genes are in an operon. The authors of OperonHunter appear to use a completely novel procedure to produce images. In each image (see Fig. 2), the central queried genome is placed at the top row, and related queried genes appear in the rows below. For each queried gene, a total of 5000 base pairs were pulled in the corresponding genome neighborhood, with the queried gene in the center. Each arrow represents a single gene. The authors state that each arrow is scaled to reflect it size relative to the region; however, we found this statement to be misleading.

The authors state that the image is divided into three regions, however, they fail to mention that this division is done rather arbitrarily and gene size scaling happens after this division. To clarify, the central gene (red arrow) and the one immediately following are set to a total width 50% of the horizontal image width of 300 pixels. The regions before and after are each given 25% of the horizontal width, and within those regions, genes are scaled accordingly. Finally, a triangular arrowhead is placed either on the left or right side of the gene depending on the transcription direction. Furthermore, at any time only a maximum of 12 genes are given unique color corresponding to gene function similarly. Any gene with a number index of 12 or higher is simply assigned a color of black. Finally, the alpha value of the operon alignment image is scaled based on a calculated STRING score from the STRING database. The STRING score is a relative measure of distance within a pre-computed graph network, which clusters proteins by similar functionality and co-incidence in literature [20].

2.2.2 OperonHunter Architecture

OperonHunter relies on a ResNet-18 [21] architecture and was trained using FastAI. The authors performed data augmentation by introducing horizontal flips and zoom to images. The best performanced was trained on 50 epochs with a learning rate of 0.0001 and a batch size of 32.

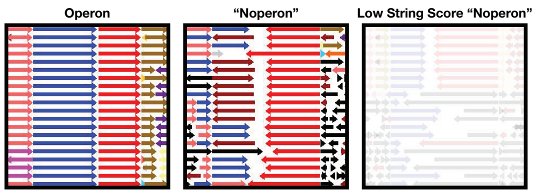

Figure 2: Operon and non-operon "noperon" images used for training. Note the incongruity in direction of arrows within the "Noperon." Low STRING scores are used to reduce the alpha channel. We noted significantly faded images such as the one displayed (right) in significantly higher frequency within the "noperon" images.

2.2.3 Drawbacks

Aside from the STRING score based alpha scaling, the scaling and coloring measures taken by OperonHunter merely serve to reproduce the view akin to PATRIC, however PATRIC scaling appears to be significantly different. The authors state that the generated images capture most of the prominent features mentioned earlier, such as gene conservation, functionality, strand direction, size, and intergenic distance. For the purposes of visualization, this type of data manipulation is acceptable for the sake of optimal human user experience. In the "Noperon" in Fig. 2, we can see the procedure can result in some bizarrely truncated arrow sizing when there are many genes in the image. In the context of scientific prediction, we find the measure to be immeasurably unjustified and subject to tremendous bias.

The authors make a point to mention the use of a pre-trained ResNet18 model, but we fail to see why this should offer any improvement. While the dataset may be small compared to ImageNet [22], which ResNet was trained on, the operon images do not have the same level of variability in terms of image content. Furthermore, the image recognition component of a pre-trained ResNet is completely blind to the validity of an operon. If this pre-training was truly important, it would have been fairly trivial to generate an ImageNet-scale dataset by assigning random sizing, direction, and alpha values to the operon images. We believe the largest drawback of using ResNet18, or any convolutional neural network (CNN) in general, comes from the small receptive field within a convolutional kernel. Genetic function often relies on very long context. Pleitropic effectors operate on the order of thousands of base pairs. Furthermore, transcription factors may physically contort DNA sequence in 3D to bring transcriptionally distant genes in close proximity of one another [23].

2.2.4 Ablation

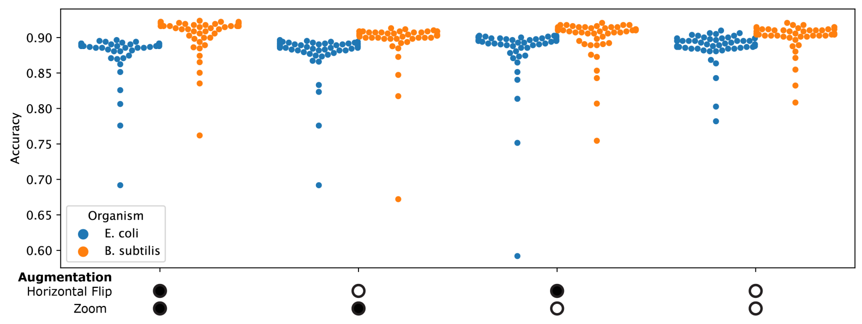

Figure 3: OperonHunter model ablations. We tested data augmentation methods combinatorially to determine their contributions to the OperonHunter performance on the operon prediction task. The presence (solid circle) or the absence (empty circle) of a data augmentation technique.

We performed ablation studies on OperonHunter to interrogate the contribution to the model's performance. OperonHunter extends their dataset by performing horizontal flips and zooms on their input images. We tested the contribution of these data augmentation techniques combinatorally as shown in Fig. 3. From these data, the data augmentation techniques do not significantly contribute to the predictive power of OperonHunter. A more complete ablation would include removing the alpha scaling based on STRING score in the images or removing the gene colors indicating function relatedness. However, we found the custom method of creating operon images difficult to use such that it was prohibitive to incorporate these further analyses.

3 Methods

We sought to demonstrate operon prediction using a machine learning model which addresses the sources of bias and architectural weaknesses of OperonHunter. We believe the introduction of sequence alignments was a truly useful source of information which could enhance Operon prediction. Specifically, the issues we sought to address were: bias and genetic under-representation or exaggeration from arbitrary scaling around the operon, extraneous data resulting from arbitrary arrow head placement and spacing between alignments, loss of long-context information from convolution, and false correlative potential coming from diagonal entries in the convolution kernel.

3.1 Operon MSA Transformer

We decided to implement a classifier which utilizes embeddings coming from a multiple sequence alignment (MSA) transformer [24]. While the original use of the MSA transformer was for protein sequence alignments, we felt the priorities of this implementation and nature of genetic transcript shared enough similarity with the intent of the MSA transformer for the architecture to be applicable.

3.2 Dataset

By creating arrow-head images, Assaf et al. [2] sought to represent both orientation and genetic locus information in an input. In order to avoid the biases created from the arbitrary image creation process, we represented the genetic alignments as 5000 $\times$ 20 arrays (5000 base pairs $\times$ 20 alignments per query gene). Separate channels were designated for orientation and gene index.

Orientation values were either set to 0 for intergenic (i.e. not defined by a genetic grouping), 1 for 3' $\rightarrow$ 5' transcription (negative direction), or 2 for 5' $\rightarrow$ 3' transcription (positive direction). Gene ID values were assigned in a similar fashion to those in Assaf et al. [2]. For the gene index values, we took a similar approach to the original dataset. For every alignment sample, we set the query gene index to 0. We considered a total maximum number of genes of 13, although there were alignments which contained as many as 83 unique genes within an alignment. Genes to the left of the query gene were assigned values -6 to -1, and 1 to 6 on the right side. Values were then globally incremented such that the minimum value of the array was 0. We initially tried to train directly from this input. However, we exceeded VRAM capacity on the GPU due to the fact that attention has $O(n^2d)$ data complexity [25, 26].

We reconsidered the scaling operation performed for the original dataset, and decided it was essentially a subsampling operation. We decided to do an even sampling to remove bias. We selected every 16th base until the array was 300 bases long. Such a procedure may create interruptions and/or loss of genes, which may have been the case had we randomly sampled positions, but genes can range from 400 to 1000 bases in length. Visual inspection (See Fig. 4) confirmed that subsampling was likely not resulting in loss genes. However, to make sure the model was still exposed to the full set of bases, we randomly shifted the start position. For example, we may select base pairs in position $[0, 15, 31, ..., 4991]$ or $[1, 16, 32, ..., 4992]$, etc. We also introduced random horizontal flips during training.

We implemented the transformer in Pytorch. Due to time constraints, we were only able to focus on the *B. subtilis* dataset.

3.2.1 Architecture

We present a visual diagram of our architecture in Fig. 5. For the base MSA transformer, we start with both MSA channels and apply independent random masking. We then add start and end tokens to the beginning and end of the alignments on the horizontal axis such that the tensors were of size 302 $\times$ 20. These MSAs were then passed through a 3D embedding space of dimension 768, bringing the embedded tensors to dimension 302 $\times$ 20 $\times$ 768. Just as in Rao et al. [24], we apply a learned positional embedding which is scaled based on the length and number of alignments.

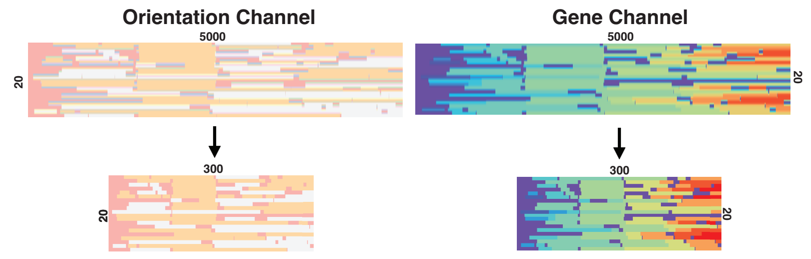

Figure 4: Sub-sampling Procedure Depicted. Images are compressed horizontally for the sake of visualization. In the orientation channel, intergenic regions (class 0) are labelled in pink, negative orientation (class 1) is shown in white, and positive orientation (class 2) is shown in yellow. For the gene channel, each gene is given a separate color, with the intergenic regions colored in dark purple. You can see the alignment of the query gene (light green) near the center of the image. The gene color in this case is only for illustrative purposes. Integer class labels were used for training and evaluation.

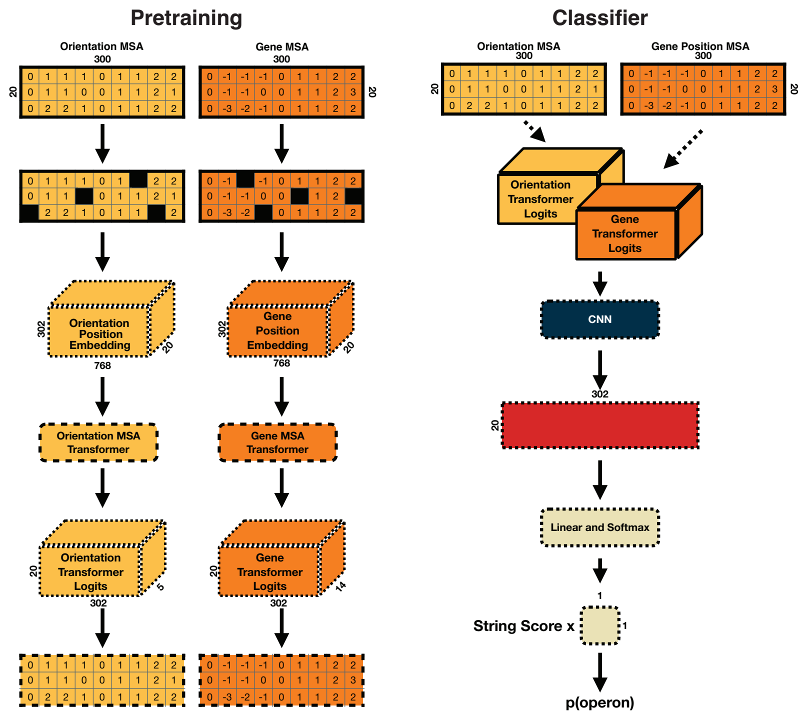

Figure 5: Architecture for pre-training transformer (left) which predicts masked entries, and the operon classifier (right), which outputs a probability value to predict if input is an operon. Randomly intialized values have dashed borders.

Embeddings are then passed to two separately constructed transformer modules. Each module had depth of 11, that is 11 repeating blocks of paired attention and feed forward blocks. This was the maximum number allowed by our GPU. Attention was similarly calculated using tied row and column attention.We masked the start and end token entries along the class index axis within the logits tensor by setting them to the minimum possible values, and then removed the start and end tokens along the position axis. The output logits were then converted into a predicted reconstruction of the input orientation and gene position MSAs.

The classifier utilized the output logits (with start and end tokens) from the transformer modules and stacked them along the class position axis. Zero padding was used to make the orientation logits the same size. We then used a convolutional neural network with a kernel size of 3 to collapse the logits dimension and return a 2D tensor of the same size as the original input MSA channels plus start and end tokens. This tensor was then unrolled into a 1D array and passed through a fully connected layer followed by a softmax activation function to produce a single value between 0 and 1. This value was multiplied by the STRING score to return a final value, which we designate as $p(operon)$. We utilize this value because high STRING scores have been used to classify operons with 80-90% accuracy on their own.

For our final evaluation, we calibrated a threshold using the validation set by which any value equal to or higher would be designated as an operon.

3.2.2 Pretraining

We used AdamW [27] as an optimizer, with learning rate set to .0003, $\beta_1 = .99$, $\beta_2 = .96$, weight decay $= .045$ with AMSGrad. Loss was evaluated via cross-entropy. We trained over 11 epochs with a batch size of 1, with each epoch taking approximately 1.5 hours to complete.

3.2.3 Classifier Training

We used identical hyperparameters to the pre-training stage. The labels corresponding to each input were either 1: Operon or 0: Not Operon. Since this was a single class prediction problem, we opted to use MSE Loss rather than cross-entropy to maintain higher resolution within the weights avoid vanishing gradients.

4 Results

Table 2: Model performance on the operon prediction task.

Model	Sensitivity (%)	Specificity (%)	Accuracy (%)	MCC	F1-Score
Operon MSA Transformer	89	87	86	.726	.879
OperonHunter	97	88	92	.84	.921
ProOpDB	93	88	91	.82	.908
Door	86	97	89	.908	.886

After training the classifier, we evaluated on the validation set. We plot the ROC curve in Fig. 6. From this curve, we determined the optimal cutoff threshold was .757. Output scores greater than this value were designated as operons.

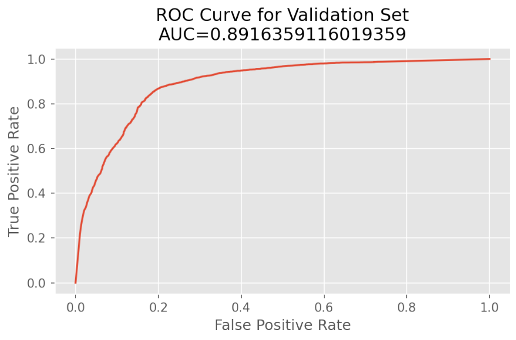

Figure 6: ROC Curve used to set cutoff threshold.

We then evaluated the performance on the test set, which we report in Table 2. We also report comparative values coming from Assaf et al. [2].

Ultimately, we were unable to outperform the reported metrics of OperonHunter using our transformer model. This may improve with extended training time or with architectural improvements, which are discussed in section 5.

5 Discussion and Future Directions

OperonHunter claims effective synthesis of gene- and phylogeny-level features to predict operon by leveraging a neural network trained with image recognition capabilities. Moreover, some of the operon prediction methods include testing dataset as part of the training dataset which results in a higher accuracy as opposed to testing on novel data. The authors also fail to mention how weighing operon prediction on comparative genomic analysis limits the generalization of predictions to less well-characterized genomes. We began this project with ambitious goals which would address every single weakness of OperonHunter. While we feel our implementation significantly reduced bias, there are some considerations we would hope to address in future attempts.

5.1 Additional Channels of Information

An even subsampling of the original vector will preserve relative sizing of genes and intergenic regions so long as they are not shorter than the cut-off. The model, however is blind to absolute positional information. This may be an important factor in operon prediction as single positional insertions or deletions can result in entire frame-shift mutations, dramatically changing the operation of a genomic transcript. We initially sought to include another channel which contained indices corresponding to each horizontal position such that the model would have some information of the subsampling procedure. This, however, required a numbered class for each position (5000), which alongside having another transformer would force us to sacrifice depth of the model, which is often the most important factor in transformer performance [28].

We also initially sought to specifically include base pairs as an additional channel. This is due to the fact that common motifs can be found in different regions, such as repeating G and C residues in non-coding portions or repeating A's in promoter sequence [29]. We were ultimately unable to do so due to time and memory constraints similar to those mentioned in the previous section.

5.2 Sampling Methods

We opted for a regular subsampling with a random starting position, but this may not be the most representative of the dataset. We would like to explore other subsampling methods, such as random sampling. It would also be interesting to collapse the gene size by assigning indices to grouped orderings of bases, although this would come at the cost of an expanded codebook size.

5.3 Cross-Attention

Our most ambitious vision involved the transformer calculating attention based on all channels simultaneously. This idea of cross attention seems to not be particularly mature, and has only been described in few recent publications. Cross-attention has so far been used as a proxy for skip exceptions in image processing models [30, 31], with the exception being from Chang et al. [32], who used recordings from three separate microphones in Amazon Echo to interpret commands. They did not release any code, and we did spend several days trying to implement, but were not able to debug in reasonable time.

5.4 Broader Impacts

Despite tremendous advances in targeted genome editing [33] and ambitious attempts to synthesize entire genomes [34], there are still limited capabilities to detect some of the most basic genetic structures in prokayotic genomes. While deep learning models have proved successful in protein-related tasks such as detecting functional annotations [35] and predicting fold [36], the ability for such models to capture genome architecture remains elusive.

Here we sought to extend the state of the art approach in operon prediction. We attempted to reproduce the results of OperonHunter as well as include ablation studies to interrogate the contributions of data augmentation techniques on the model's performance. We developed an operon MSA transformer to better synthesize the gene-level and comparative genomic features of operons. We view our work as a step towards understanding the design principles of genomic synteny and how this structure changes as a function of evolution.

References

[1] François Jacob and Jacques Monod. Genetic regulatory mechanisms in the synthesis of proteins. 3 (3):318–356. ISSN 0022-2836. doi: https://doi.org/10.1016/S0022-2836(61)80072-7. URL https://www.sciencedirect.com/science/article/pii/S0022283661800727.

[2] Rida Assaf, Fangfang Xia, and Rick Stevens. Detecting operons in bacterial genomes via visual representation learning. 11(1):2124. ISSN 2045-2322. doi: 10.1038/s41598-021-81169-9. URL https://www.nature.com/articles/s41598-021-81169-9. Number: 1 Publisher: Nature Publishing Group.

[3] Annalisa Pantosti, Andrea Sanchini, and Monica Monaco. Mechanisms of antibiotic resistance in staphylococcus aureus. 2(3):323–334. ISSN 1746-0921. doi: 10.2217/17460913.2.3.323.

[4] L. Medina-Aparicio, J. E. Rebollar-Flores, A. L. Gallego-Hernández, A. Vázquez, L. Olvera, R. M. Gutiérrez-Ríos, E. Calva, and I. Hernández-Lucas. The CRISPR/cas immune system is an operon regulated by LeuO, h-NS, and leucine-responsive regulatory protein in salmonella enterica serovar typhi . 193(10):2396–2407. ISSN 0021-9193. doi: 10.1128/JB.01480-10. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3133143/.

[5] Frederick R. Blattner, Guy Plunkett, Craig A. Bloch, Nicole T. Perna, Valerie Burland, Monica Riley, Julio Collado-Vides, Jeremy D. Glasner, Christopher K. Rode, George F. Mayhew, Jason Gregor, Nelson Wayne Davis, Heather A. Kirkpatrick, Michael A. Goeden, Debra J. Rose, Bob Mau, and Ying Shao. The complete genome sequence of escherichia coli k-12. 277(5331):1453–1462. doi: 10.1126/science.277.5331.1453. URL https://www.science.org/doi/10.1126/science.277.5331.1453. Publisher: American Association for the Advancement of Science.

[6] Heladia Salgado, Gabriel Moreno-Hagelsieb, Temple F. Smith, and Julio Collado-Vides. Operons in escherichia coli: Genomic analyses and predictions. 97(12):6652–6657. doi: 10.1073/pnas.110147297. URL https://www.pnas.org/doi/10.1073/pnas.110147297. Publisher: Proceedings of the National Academy of Sciences.

[7] Xin Chen, Zhengchang Su, Ying Xu, and Tao Jiang. Computational prediction of operons in synechococcus sp. WH8102. 15(2):211–222. ISSN 0919-9454.

[8] Thao T. Tran, Phuongan Dam, Zhengchang Su, Farris L. Poole, Michael W. W. Adams, G. Tong Zhou, and Ying Xu. Operon prediction in pyrococcus furiosus. 35(1):11–20. ISSN 1362-4962. doi: 10.1093/nar/gkl974.

[9] Guo-qing Zhang A, Zhi-wei Cao B, Qing-ming Luo A, Yu-dong Cai C, and Yi-xue Li B. Brief communication operon prediction based on SVM.

[10] Phuongan Dam, Victor Olman, Kyle Harris, Zhengchang Su, and Ying Xu. Operon prediction using both genome-specific and general genomic information. 35(1):288–298. ISSN 1362-4962. doi: 10.1093/nar/gkl1018.

[11] Joseph Bockhorst, Mark Craven, David Page, Jude Shavlik, and Jeremy Glasner. A bayesian network approach to operon prediction. 19(10):1227–1235. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg147.

[12] Martin T. Edwards, Stuart C. G. Rison, Neil G. Stoker, and Lorenz Wernisch. A universally applicable method of operon map prediction on minimally annotated genomes using conserved genomic context. 33 (10):3253–3262. ISSN 1362-4962. doi: 10.1093/nar/gki634.

[13] B. P. Westover, J. D. Buhler, J. L. Sonnenburg, and J. I. Gordon. Operon prediction without a training set. 21(7):880–888. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti123. URL https://doi.org/10.1093/bioinformatics/bti123.

[14] E. Jacob, R. Sasikumar, and K. N. R. Nair. A fuzzy guided genetic algorithm for operon prediction. 21 (8):1403–1407. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti156.

[15] Yu Zheng, Joseph D. Szustakowski, Lance Fortnow, Richard J. Roberts, and Simon Kasif. Computational identification of operons in microbial genomes. 12(8):1221–1230. ISSN 1088-9051. doi: 10.1101/gr.200602.

[16] Blanca Taboada, Cristina Verde, and Enrique Merino. High accuracy operon prediction method based on STRING database scores. 38(12):e130. ISSN 0305-1048. doi: 10.1093/nar/gkq254. URL https://doi.org/10.1093/nar/gkq254.

[17] Lars J. Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, Peer Bork, and Christian von Mering. STRING 8a global view on proteins and their functional interactions in 630 organisms. 37:D412–D416. ISSN 0305-1048. doi: 10.1093/nar/gkn760. URL `https://doi.org/10.1093/nar/gkn760`.

[18] Xizeng Mao, Qin Ma, Chuan Zhou, Xin Chen, Hanyuan Zhang, Jincai Yang, Fenglou Mao, Wei Lai, and Ying Xu. DOOR 2.0: presenting operons and their functions through dynamic and integrated views. 42: D654–659. ISSN 1362-4962. doi: 10.1093/nar/gkt1048.

[19] Thomas Brettin, James J. Davis, Terry Disz, Robert A. Edwards, Svetlana Gerdes, Gary J. Olsen, Robert Olson, Ross Overbeek, Bruce Parrello, Gordon D. Pusch, Maulik Shukla, James A. Thomason, Rick Stevens, Veronika Vonstein, Alice R. Wattam, and Fangfang Xia. RASTtk: A modular and extensible implementation of the RAST algorithm for building custom annotation pipelines and annotating batches of genomes. 5:8365. ISSN 2045-2322. doi: 10.1038/srep08365. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4322359/`.

[20] Christian von Mering, Lars J. Jensen, Berend Snel, Sean D. Hooper, Markus Krupp, Mathilde Foglierini, Nelly Jouffre, Martijn A. Huynen, and Peer Bork. STRING: known and predicted protein-protein associations, integrated and transferred across organisms. 33:D433–437. ISSN 1362-4962. doi: 10.1093/nar/gki005.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. URL `http://arxiv.org/abs/1512.03385`.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. URL `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

[23] György Vámosi and David Rueda. DNA bends the knee to transcription factors. 114(10):2253–2254. ISSN 0006-3495. doi: 10.1016/j.bpj.2017.10.047. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6129554/`.

[24] Roshan M. Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. MSA transformer. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8844–8856. PMLR. URL `https://proceedings.mlr.press/v139/rao21a.html`. ISSN: 2640-3498.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

[26] igrinis. Answer to "computational complexity of self-attention in the transformer model". URL `https://stackoverflow.com/a/65794564`.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. doi: 10.48550/arXiv.1711.05101. URL `https://arxiv.org/abs/1711.05101v3`.

[28] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. URL `https://openreview.net/forum?id=SylO2yStDr`.

[29] Frontiers | analysis of genomic sequence motifs for deciphering transcription factor binding and transcriptional regulation in eukaryotic cells | genetics. URL `https://www.frontiersin.org/articles/10.3389/fgene.2016.00024/full`.

[30] Manjin Sheng, Wenjie Xu, Jane Yang, and Zhongjie Chen. Cross-attention and deep supervision UNet for lesion segmentation of chronic stroke. 16. ISSN 1662-453X. URL `https://www.frontiersin.org/article/10.3389/fnins.2022.836412`.

[31] Olivier Petit, Nicolas Thome, Clément Rambour, and Luc Soler. U-net transformer: Self and cross attention for medical image segmentation. doi: 10.48550/arXiv.2103.06104. URL `https://arxiv.org/abs/2103.06104v2`.

[32] Feng-Ju Chang, Martin Radfar, Athanasios Mouchtaris, Brian King, and Siegfried Kunzmann. End-to-end multi-channel transformer for speech recognition. doi: 10.48550/arXiv.2102.03951. URL `https://arxiv.org/abs/2102.03951v1`.

[33] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A. Doudna, and Emmanuelle Charpentier. A programmable dual-RNAguided DNA endonuclease in adaptive bacterial immunity. 337 (6096):816–821. doi: 10.1126/science.1225829. URL `https://www.science.org/doi/10.1126/science.1225829`. Publisher: American Association for the Advancement of Science.

[34] Sarah M. Richardson, Leslie A. Mitchell, Giovanni Stracquadanio, Kun Yang, Jessica S. Dymond, James E. DiCarlo, Dongwon Lee, Cheng Lai Victor Huang, Srinivasan Chandrasegaran, Yizhi Cai, Jef D. Boeke, and Joel S. Bader. Design of a synthetic yeast genome. 355(6329):1040–1044. doi: 10.1126/science.aaf4557. URL `https://www.science.org/doi/full/10.1126/science.aaf4557`. Publisher: American Association for the Advancement of Science.

[35] Maxwell L. Bileschi, David Belanger, Drew H. Bryant, Theo Sanderson, Brandon Carter, D. Sculley, Alex Bateman, Mark A. DePristo, and Lucy J. Colwell. Using deep learning to annotate the protein universe. pages 1–6. ISSN 1546-1696. doi: 10.1038/s41587-021-01179-w. URL `https://www.nature.com/articles/s41587-021-01179-w`. Publisher: Nature Publishing Group.

[36] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin ídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. 596(7873):583–589. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL `https://www.nature.com/articles/s41586-021-03819-2`. Number: 7873 Publisher: Nature Publishing Group.