Identifying Operons with Deep Neural Networks

Rida Assaf[1,*], Fangfang Xia[3,4], and Rick Stevens[2,3]

[1]University of Chicago, Department of Computer Science, Chicago, 60637, USA
[2]University of Chicago, The University of Chicago Consortium for Advanced Science and Engineering, Chicago, 60637, USA
[3]Argonne National Laboratory, Computing Environment and Life Sciences Division, Lemont, 60439, USA
[4]Argonne National Laboratory, Data Science and Learning Division, Lemont, 60439, USA
[*]rida@uchicago.edu

ABSTRACT

Genes in prokaryotic genomes often assemble in transcription units called operons. Detecting operons are of significant importance to help infer functionality and detect regulatory networks. Several tools have been proposed to detect such operons computationally. We propose a new method, which we name Operon Hunter, that uses visual representations of genomic fragments and a residual neural network architecture to make operon predictions. Our method uses a pertained network via transfer learning to leverage big datasets.
We report the highest accuracy in the literature that we know of when tested on the standard datasets that are reported in various studies: E. coli and B. subtilis, with an F1 score of 0.83, outperforming the previously reported state of the art tools. Our method also demonstrates a clear advantage when it comes to detecting full operons rather than separate gene pairs.

Introduction

In prokaryotes, contiguous genes are often arranged into operons. These operons usually include metabolically or functionally related genes that are often grouped together on the same strand and transcribed in the same polycistronic messenger RNA[1–11]. Genes in the same operon share a common promoter and terminator[12]. We reserve the definition of Operons as those including at least two genes, distinguishing them from Transcription Units, which may include one or more genes[1]. It is estimated that more than 50% of bacterial genes form operons[13]. Since operons are often formed by genes that are related functionally or metabolically, predicting operons could help us predict higher levels of gene organization and regulatory networks[1,2,8–11,14]. Such predictions could help annotate gene functions[15], aid in drug development[16], and antibiotic resistance[17].

Different methods use different features of operons to make their predictions. Some methods focus on promoters and terminators and use Hidden Markov Models (HMMs) to make their predictions[18–20], others rely on gene conservation information[21], while others leverage functional relatedness between the genes instead[1,22]. Perhaps the most prominent features used in operon prediction are transcription direction and inter-genetic distances[1,9,13,24–32,34]. The idea that adjacent genes that are conserved across multiple genomes are likely to be co-transcribed makes gene conservation another important feature[10,35]. In addition to HMMs, different machine learning (ML) technologies are deployed for the predition of operons, such as neural networks[24,25], support vector machines[26], and decision tree-based classification[27]. Other tools utilize Bayesian probabilities[28–30], genetic algorithms[31], and graph-theoretic techniques[22,29].

Many tools perform their predictions on gene pairs, and then assemble their predictions that are made on an entire genome into operons by combining contiguous pairs that are labeled as operonic.[1,3]. These tools are often benchmarked using two genomes whose operons have been studied extensively, E. coli and B. subtilis[3].

Among the tools that predict operons, we focus on two methods. The first is reported to have the highest accuracy among operon prediction tools[7]. It is based on an artificial neural network that uses inter-genetic distance and protein functional relationships. The method uses gene neighborhood, fusion, co-occurrence and co-expression in addition to protein-protein interaction and literature mining to generate a score that can be used as input to their network[5,7]. The predictions made by this method were compiled into what is called the Prokaryotic Operon Database (ProOpDB)[8]. The second tool is called the Database of Prokaryotic Operons (DOOR) and was ranked as the best operon predictor among 14 other predictors[2], and was also reported by another study to have the second highest accuracy after ProOpDB[7]. Their algorithm uses a combination of a non-linear (decision-tree based) classifier and a linear (logistic function-based classifier) depending on the number of experimentally validated operons that could be used in the training[2].

Before delving into our method, it's worth pointing out some of the challenges that undermine operon prediction. Predictors that rely on features such as gene conservation or functional assignment are limited in the sense that they could not be applied to genomes or genomic fragments that lack these features. So while such predictors might perform well on gene pairs that include the necessary features, their performance might drop considerably when considering the whole genome[1]. Moreover, even though most methods validate their results by comparing their predictions to experimentally verified operons, the fact that the experimentally verified datasets are not available for all prokaryotes or that the datasets used vary between studies poses extra challenges when trying to compare prediction methods. Brouwer et al tried to compare several methods on a uniform dataset and noticed significant gap in the metrics reported. The drop was even higher when considering full operons rather than gene pairs[7,36]. Finally, Some methods include the testing dataset in the training dataset which leads to reported accuracies that are significantly higher, whereas the information flow would not be easily and readily transferable to other genomes[9]. With those challenges in mind, we describe out method in the next section and how we tried to alleviate some of the challenges, and present our results in section 3.

Methods

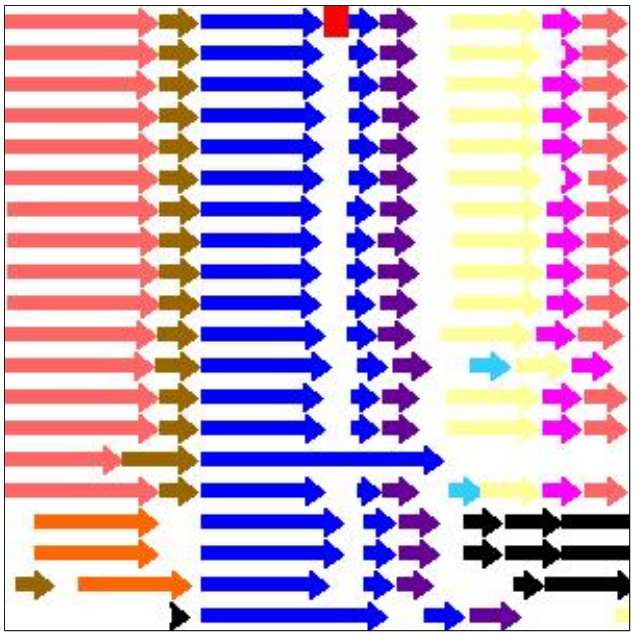
In our previous work[37], we demonstrated a successful method that relies on using images to represent genomic data to identify genomic islands. The idea is that using graphical representations makes it easier to leverage the powerful ML technologies that have become the state of art in solving computer vision problems. Algorithms based on Deep Neural Networks have proven to be superior in competitions such as ImageNet Large Scale Visual Recognition Challenge (ILSVRS)[38]. Deep learning is the process of training deep neural networks (i.e neural networks with many hidden layers). The depth of these networks allows them to learn more complex patterns and higher order relationships, at the cost of being more computationally expensive and requiring more data to work effectively. Improvements in such algorithms have been translated to improvements in a myriad of domains reliant on computer vision tasks[39]. In our approach, the images were generated by PATRIC (The Pathosystems Resource Integration Center), which is a bacterial Bioinformatics resource center that we are part of (https://www.patricbrc.org)[40]. More specifically, one of the services PATRIC provides is the compare region viewer service, where a query genome is aligned against a set of other related genomes anchored at a specific focus gene. The service starts with finding other genes that are of the same family as the query gene, and then aligning their flanking regions accordingly. Such graphical representations are appealing as they help users visualize the genomic areas of interest. We replicated the service by implementing an offline version forked from the production UI, which is necessary for computational efficiency and for consistency in the face of any future UI changes. Each row of arrows in the generated image represents a region in a genome, with the query genome being the top row. Each arrow represents a single gene, capturing its size and strand directionality. The arrows and the distance between them are scaled on each row to reflect the actual length and distance between the genes. Colors represent functionality. The blue arrows are reserved to represent the query gene pair. The rest of the genes share the same color if they belong to the same family, or are colored black if they are not found in the query genome's focus region. The families used in the coloring process are PATRIC's Global Pattyfams that are generated by mapping signature k-mers to protein functionality, using non-redundant protein databases built per genus before being transferred across genera[41].

We also introduce slight modifications to the produced images, to capture more meaningful operon-specific features. Namely, we highlight the inter-genetic distance between the query gene pair by filling the gap as a red rectangle. As a result, the generated images capture most of the prominent features mentioned earlier, such as gene conservation, functionality, strand direction, size, and inter-genetic distance. Figure 1 shows an example of how the generated images look like.

Images offer a natural way to compare genes (horizontally) and clusters across genomes (vertically) with 2D convolution. The fact that the compare region view sorts genomes by evolutionary distance allows the neural network to exploit locality and place more emphasis on close genomes via incremental pooling. An additional benefit of working with the image modality is to be able to leverage the state-of-the-art deep learning models, many of which were first developed in vision tasks and perfected over years of iterations. Other than our previously mentioned successful approach to detect genomic islands, Google researchers have used spectrogram (instead of text) in direct speech translation[42] and DNA sequence pile-up graphs (instead of alignment data) in genetic variant calling[43]. In both cases, the image-based models outperformed their respective previous state-of-the-art method based on traditional domain features. Further, the low-level visual feature patterns learned in pre-trained image models have been demonstrated to transfer to distant learning tasks on non-image data in several preliminary studies ranging from environmental sound classification to cancer gene expression typing[44]. Much like feature engineering methods, casting tabular data to images encodes information in a way more amenable to learning without explicitly adding information. It can also be easily integrated with other data modalities in the latent vector representation to prevent information loss. We hypothesize this emerging trend of representing data with images will continue until model tuning and large-scale pre-training in scientific domains start to catch up with those in computer vision.

When it comes to training the model, perhaps the most important step is building the right dataset. We wanted to stick to

Figure 1. Example of an image generated by our service to be fed as input to the neural network. Each arrow represents a single gene. Each row captures the area of interest in a genome. The query genome is the top row. The rest of the rows are genomes selected by evolutionary distance. The query gene pair are color-coded with blue, and the inter-genetic distance between them is colored red. The rest of the genes share the same color if they belong to the same family.



experimentally verified data instead of rely on other tools' predictions. But considering how the former is more limited, and how deep the models we were using are, it put the model at the risk of over-fitting. To get around that, we used a technique referred to as transfer learning[45]. In transfer learning, a model does not have to be trained from scratch. Instead, the idea is to retrain a model that has been previously trained on a related task. The newly retrained model should then be able to transfer its existing knowledge and apply it to the new task. This approach gives us the ability to reuse models that have been trained on huge amounts of data, while adding the necessary adjustments to make them available to work with more limited datasets, adding a further advantage to our approach of representing the data visually. We used the FastAI[46] platform to train and test our model. After testing the performance of the different available pre-trained models on our dataset, we observed the best performance when using the ResNet18 model. These models were previously trained on Imagenet, which is a database that contains more than a million images belonging to more than a thousands categories[38]. Thus, a model that was previously trained on ImageNet is already good at feature extraction and visual recognition. To make the model compatible with the new task, the top layer of the network is retrained on our operon dataset, while the rest of the network is left intact, which is more powerful than starting with a deep network with random weights. We also found better results when we changed the mode of transfer learning to also re-train the previous layers of the model (as opposed to just the last layer) by unfreezing their weights. For our training data, we aligned the query genome with the set of reference+representative genomes found on PATRIC. For each genome used, our program produces an image for every consecutive gene pair, capturing the surrounding 5 Kbp (kilo base pairs) flanking region. A balanced dataset was then curated from the total set of images created. In this supervised learning approach, we used the "known operons" section of the Operon DataBase (ODB)[47] to label our images. The Known Operons section of ODB contains experimentally verified operons. To construct the training dataset, we used 8 genomes (Table 1) with a significant number of labeled operons. The result was 4,861 images of Operonic gene pairs. We generated a close number of images representing non-Operonic gene pairs. To construct the negative dataset (the non-operons), we resorted to the same approach reported by ProOpDB[8] as the standard. The idea is to select the known Operon boundaries along with the respective upstream or downstream gene and label the resulting pair as non-operonic.

Results

To verify our results, we resort to two extensively studied genomes with experimentally verified operons: E. coli and B. subtilis, and report the achieved specificity, sensitivity, and F1 score. These genomes are the standard datasets for verifying operon prediction results in the literature, and serve as a good comparison ground. We compared our predictions to those made by ProOpDB and DOOR, the two tools with top accuracies as reported by independent studies[2,7,8]. To build the testing dataset,

Table 1. Breakdown of the training dataset: The genome names and the corresponding number of gene pairs used. Operon pairs were harvested from the Known Operons section of ODB.

Genome name	Operon pairs	non-Operon pairs
Escherichia coli	1393	1038
Listeria monocytogenes	806	537
Legionella pneumophila	611	246
Helicobacter pylori	563	196
Corynebacterium glutamicum	525	868
Photobacterium profundum	447	1354
Bacillus subtilis	329	328
Mycoplasma pneumoniae	187	349
Total	4861	4916

we used the results published in RegulonDB[48] and DBTBS[49], as is commonly done. RegulonDB is a database containing verified operons found in E. coli, and although it is regularly updated, we used the predictions made by the version that matches ProOpDB's release. DBTBS is a database that contains verified operon predictions for B. subtilis. The resulting dataset consists of 1081 operon gene pairs. We used the same approach mentioned earlier to construct a matching negative non-operon dataset. Namely, we used the boundaries of operons reported by ODB, RegulonDB and DBTBS, and reported each boundary with its neighboring gene as non-operonic pair. We then chose a subset of all the pairs to have a balanced dataset. We also filtered out the non-operonic pairs that were predicted by ProopDB or DOOR as operons, so the resulting negative dataset turned out to be slightly smaller than the positive operon dataset. Using predicted operons as well as verified ones diminishes the possibility of labeling an operonic gene pair with the wrong class. We skipped single-gene operons and rna genes as these are not handled by our tool. Table 2 shows a breakdown of the testing dataset. It is important to highlight that while other tools face the challenge of transferring their performance across lineages, our approach does not face that challenge, which is made clear by how different the genomes constituting our datasets are.

Table 2. Breakdown of the testing dataset: The genome names and the corresponding number of gene pairs used. Operon pairs were scoured from RegulonDB (for E. coli) and DBTBS (for B. subtilis).

Genome	Operon pairs	non-Operon pairs
E. coli	513	513
B. subtilis	568	328
Total	1081	841

It is worth noting at this point that ProOpDB included its testing data as part of its training data, and reported a decrease in accuracy when tested on separate dataset. In our experiments however, we exclude the testing dataset from the training dataset. So for example when we are evaluating the model's performance on E. coli, we train the model on all the images belonging to all the genomes except those that belong to E. coli, and then test the model's performance on the images generated from the E. coli genome. In addition, since we mentioned earlier that accuracies tend to change when trying to detect full operons rather than gene pairs[7], we present a more in-depth breakdown to show how much they change. To make full operon predictions, we first make predictions on every consecutive gene pair in a genome, and then merge the consecutively labeled operon pairs into full operons.
Tables 3 and 4 report the specificity and sensitivity achieved by the three mentioned tools over the constructed datasets, and Table 5 shows the resulting F1 score.

Table 3. Sensitivity (True positive rate): Percentage of the 1081 Operonic gene pairs that were detected by the different tools. With a breakdown of the E. coli gene pairs and the B. subtilis gene pairs.

Tool	All (1081 pairs)	E. coli (513 pairs)	B. subtilis (568 pairs)
ProOpDB	89%	90%	88%
DOOR	68%	81%	56%
Operon Hunter	86%	84%	88%

We can see that ProOpDB achieves the highest sensitivity, with Operon Hunter coming next and DOOR third. Note that the

results reported by ProOpDB vary from those reported in their paper, which could be attributed to the fact that we are reported results that cover the whole genome, and not just gene pairs with COG assignments as is required by their tool. Also note that DOOR's performance on E. coli seems to be significantly more superior than its performance on B. subtilis. To make sure that the algorithms are not simply over-predicting the positive class, we compare the predictions made on a negative dataset in Table 4.

Table 4. Specificity (True negative rate): Percentage of the 841 Non-operonic gene pairs that were detected by the different tools. With a breakdown of the E. coli gene pairs and the B. subtilis gene pairs.

Tool	All (841 pairs)	E. coli (513 pairs)	B. subtilis (328 pairs)
ProOpDB	35%	30%	34%
DOOR	93%	89%	100%
Operon Hunter	80%	83%	76%

To present a fair comparison in one table, we present the resulting F1 score for each tool in Table 5. It's worth noting that even though ProOpDB scored the highest sensitivity, it scored the lowest specificity. While the high sensitivity could be attributed to having the testing data as part of their training data, the low specificity could be due to a significant number of genes missing COG assignments, resulting in a lower overall F1 score.

Table 5. F1-score achieved by different tools.

	ProOpDB	DOOR	Operon Hunter
F1-score	0.5	0.79	0.83

Predicting full operons is a more challenging task that requires accurate boundary detection. We present how the 3 tools cope with full predictions in Table 6. We can see that Operon Hunter achieves the highest accuracy, with 62% of its predictions exactly matching full operons, and 24% being partial hits, while missing 14% of operons mentioned in the dataset, which is also the most across the three tools. Compare that to ProOpDB whose accuracy drops significantly to 46% when making exact hits on full operon predictions. DOOR comes in last getting only 21% of exact matches with full operons. Operon Hunter seems better adept to predict full operon boundaries accuractely, while ProOpDB and DOOR mostly get partial hits. Even though DOOR gets a small percentage of full hits, it does get all the other reported operons in the datasets partially, with the minimum number of absolute misses among the three tools (0%).

Table 6. Comparison of the results between OperonHunter, ProOpDB, and DOOR when considering full operon predictions. Full hits are the percentage of operons predicted that exactly match the validated results, partial hits are predictions that have different boundaries but intersect with the verified results, and misses are verified operons that are missed by the tool. The percentages are reported over 479 operons.

Tool	Full Hits	Partial Hits	Misses
ProOpDB	46%	49%	5%
DOOR	21%	79%	0%
Operon Hunter	69%	20%	11%

Author contributions statement

R.A. carried out the implementation and wrote the manuscript. R.S. and F.X. were involved in planning and supervised the work. All authors aided in interpreting the results. All authors provided critical feedback and commented on the manuscript.

Funding

PATRIC has been funded in whole or in part with Federal funds from the National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services [HHSN272201400027C]. Funding for open access charge: Federal funds from the National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services [HHSN272201400027C].

Availability of data and materials

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Additional information

The authors declare that they have no competing interests.

References

1. Romero,P.R. and Karp,P.D. (2004) Using functional and organizational information to improve genome-wide computational prediction of transcription units on pathway-genome databases. Bioinformatics, 20, 709–717.

2. Mao F, Dam P, Chou J, et al (2009) . DOOR: a database for prokaryotic operons. Nucleic Acids Res 2009;37:459–463.

3. Haller R, Kennedy M, Arnold N, Rutherford R (2010). The transcriptome of Mycobacterium tuberculosis. Appl Microbiol Biotechnol. 2010; 86:1–9.

4. Pelly, S., Winglee, K., Xia, F., Stevens, R. L., Bishai, W. R., & Lamichhane, G. (2016). REMap: operon map of M. tuberculosis based on RNA sequence data. Tuberculosis, 99, 70-80.

5. Taboada, B., Estrada, K., Ciria, R., & Merino, E. (2018). Operon-mapper: a web server for precise operon identification in bacterial and archaeal genomes. Bioinformatics, 34 (23), 4118-4120.

6. Price, M. N., Huang, K. H., Alm, E. J., & Arkin, A. P. (2005). A novel method for accurate operon predictions in all sequenced prokaryotes. Nucleic acids research, 33 (3), 880-892.

7. Zaidi, S. S. A., & Zhang, X. (2016). Computational operon prediction in whole-genomes and metagenomes. Briefings in functional genomics, 16 (4), 181-193.

8. Taboada, B., Ciria, R., Martinez-Guerrero, C. E., & Merino, E. (2011). ProOpDB: Prokaryotic Operon Data Base. Nucleic acids research, 40 (D1), D627-D631.

9. Taboada, B., Verde, C., & Merino, E. (2010). High accuracy operon prediction method based on STRING database scores. Nucleic acids research, 38 (12), e130-e130.

10. Bergman, N. H., Passalacqua, K. D., Hanna, P. C., & Qin, Z. S. (2007). Operon prediction for sequenced bacterial genomes without experimental information. Appl. Environ. Microbiol., 73 (3), 846-854.

11. Fortino, V., Smolander, O. P., Auvinen, P., Tagliaferri, R., & Greco, D. (2014). Transcriptome dynamics-based operon prediction in prokaryotes. BMC bioinformatics, 15 (1), 145.

12. Fran B, Perrin D, Monod J, et al (1960). The operon : a group of genes whose expression is coordinated by an operator. J Bacteriol;29:1727–9.

13. Okuda S, Kawashima S, Kobayashi K, et al (2007). Characterization of relationships between transcriptional units and operon structures in Bacillus subtilis and Escherichia coli. BMC Genomics;8:48.

14. Hodgman TC (2000). A historical perspective on gene/protein functional assignment. Bioinformatics;16:10–5.

15. Joon M, Bhatia S, Pasricha R, et al (2010). Functional analysis of an intergenic non-coding sequence within mce1 operon of M. tu- berculosis. BMC Microbiol;10:128.

16. Wang S, Wang Y, Liang Y, et al (2007). A multi-approaches-guided genetic algorithm with application to operon prediction. Artif Intell Med;41:151–9.

17. Pantosti A, Sanchini A, Monaco M (2007). Mechanisms of antibiotic resistance in Staphylococcus aureus. Future Microbiol;2: 323–34.

18. Yada,T., Nakao,M., Totoki,Y. and Nakai,K. (1999) Modeling and predicting transcriptional units of Escherichia coli genes using hidden Markov models. Bioinformatics, 15: 987–993.

19. Craven,M., Page,D., Shavlik,J., Bockhorst,J. and Glasner,J. (2000) A probabilistic learning approach to whole-genome operon pre- diction. Proc. Conf. Intell. Syst. Mol. Biol., 8: 116–127.

20. Tjaden,B., Haynor,D.R., Stolyar,S., Rosenow,C. and Kolker,E. (2002) Identifying operons and untranslated regions of transcripts using Escherichia coli RNA expression analysis. Bioinformatics, 18 (Suppl. 1): S337–S344.

21. Ermolaeva,M.D., White,O. and Salzberg,S.L. (2001) Predic- tion of operons in microbial genomes. Nucleic Acids Res., 29: 1216–1221.

22. Zheng,Y., Szustakowski,J.D., Fortnow,L., Roberts,R.J. and Kasif,S. (2002) Computational identification of operons in microbial genomes. Genome Res., 12: 1221–1230.

23. Ogata,H., Fujibuchi,W., Goto,S. and Kanehisa,M. (2000) A heur- istic graph comparison algorithm and its application to detect functionally related enzyme clusters. Nucleic Acids Res., 28: 4021–4028.

24. Chen,X., Su,Z., Xu,Y. and Jiang,T. (2004) Computational prediction of operons in Synechococcus sp. WH8102. Genome Inform., 15, 211–222.

25. Tran,T.T., Dam,P., Su,Z., Poole,F.L., Adams,M.W., Zhou,G.T. and Xu,Y. (2007) Operon prediction in Pyrococcus furiosus. Nucleic Acids Res., 35, 11–20.

26. Zhang,G.Q., Cao,Z.W., Luo,Q.M., Cai,Y.D. and Li,Y.X. (2006) Operon prediction based on SVM. Comput. Biol. Chem., 30, 233–240.

27. Dam,P., Olman,V., Harris,K., Su,Z. and Xu,Y. (2007) Operon prediction using both genome-specific and general genomic information. Nucleic Acids Res., 35, 288–298.

28. Bockhorst,J., Craven,M., Page,D., Shavlik,J. and Glasner,J. (2003) A Bayesian network approach to operon prediction. Bioinformatics, 19, 1227–1235.

29. Edwards,M.T., Rison,S.C., Stoker,N.G. and Wernisch,L. (2005) A universally applicable method of operon map prediction on minimally annotated genomes using conserved genomic context. Nucleic Acids Res., 33, 3253–3262.

30. Westover,B.P., Buhler,J.D., Sonnenburg,J.L. and Gordon,J.I. (2005) Operon prediction without a training set. Bioinformatics, 21, 880–888.

31. Jacob,E., Sasikumar,R. and Nair,K.N. (2005) A fuzzy guided genetic algorithm for operon prediction. Bioinformatics, 21, 1403–1407.

32. Salgado,H., Moreno-Hagelsieb,G., Smith,T.F. and Collado- Vides,J. (2000) Operons in Escherichia coli: genomic analyses and predictions. Proc. Natl Acad. Sci. USA, 97, 6652–6657.

33. Okuda,S., Kawashima,S., Kobayashi,K., Ogasawara,N., Kanehisa,M. and Goto,S. (2007) Characterization of relationships between transcriptional units and operon structures in Bacillus subtilis and Escherichia coli. BMC Genomics, 8, 48.

34. Yan,Y. and Moult,J. (2006) Detection of operons. Proteins, 64, 615–628.

35. Overbeek, R., M. Fonstein, M. D'Souza, G. D. Pusch, and N. Maltsev. (1999). The use of gene clusters to infer functional coupling. Proc. Natl. Acad. Sci. USA 96:2896–2901.

36. Brouwer RWW, Kuipers OP, Van Hijum SAFT (2008). The relative value of operon predictions. Brief Bioinform:367–75.

37. Rida Assaf, Fangfang Xia, and Rick Stevens (2019). Identifying Genomic Islands with Deep Neural Networks. Bioarxiv.

38. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (2015) ImageNet Large Scale Visual Recognition Challenge. *IJCV*.

39. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, Zbigniew Wojna (2016) Rethinking the Inception Architecture for Computer Vision. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818-2826.

40. Wattam AR, Davis JJ, Assaf R, Boisvert S, Brettin T, Bun C, Conrad N, Dietrich EM, Disz T, Gabbard JL, Gerdes S, Henry CS, Kenyon RW, Machi D, Mao C, Nordberg EK, Olsen GJ, Murphy-Olson DE, Olson R, Overbeek R, Parrello B, Pusch GD, Shukla M, Vonstein V, Warren A, Xia F, Yoo H, Stevens RL. (2017) Improvements to PATRIC, the all-bacterial Bioinformatics Database and Analysis Resource Center. *Nucleic Acids Res.* **45 (D1)**, D535-D542.

41. Davis JJ, Gerdes S, Olsen GJ, Olson R, Pusch GD, Shukla M, Vonstein V, Wattam AR and Yoo H (2016) PATtyFams: Protein Families for the Microbial Genomes in the PATRIC Database. Front. Microbiol. 7:118. doi: 10.3389/fmicb.2016.00118

42. Jia, Y., Weiss, R.J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z. and Wu, Y., 2019. Direct speech-to-speech translation with a sequence-to-sequence model. arXiv preprint arXiv:1904.06037.

43. Poplin, R., Chang, P., Alexander, D., Schwartz, S., Colthurst, T., & Ku, A. et al. (2018) A universal SNP and small-indel variant caller using deep neural networks. *Nature Biotechnology.*

44. Jeremy Howard, (2019) Lesson 2: Deep Learning 2019 - Data cleaning and production; SGD from scratch. Retrieved from https://www.youtube.com/watch?v=ccMHJeQU4Qw

45. How to Retrain an Image Classifier for New Categories | TensorFlow Hub | TensorFlow. (2018). Retrieved from https://www.tensorflow.org/hub/tutorials/image_retraining

46. FastAI | FastAI. (2018). Retrieved from https://docs.fast.ai/index.html

47. Okuda, S. and Yoshizawa, A.C. (2011). ODB: a database for operon organizations, 2011 update. *Nucleic Acids Res.39 (Database issue)*:D552-555.

48. Gama-Castro,S., Jimenez-Jacinto,V., Peralta-Gil,M., Santos- Zavaleta,A., Penaloza-Spinola,M.I., Contreras-Moreira,B., Segura- Salazar,J., Muniz-Rascado,L., Martinez-Flores,I., Salgado,H. et al. (2008) RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. Nucleic Acids Res., 36, D120–D124.

49. Sierro N., Makita Y., de Hoon M.J.L. and Nakai K. (2008). DBTBS: a database of transcriptional regulation in Bacillus subtilis containing upstream intergenic conservation information. *Nucleic Acids Res.*, 36 (Database issue):D93-D96; doi:10.1093/nar/gkm910.