EasyChair Preprint

№ 8146

Gene Prediction Using Deep Learning

Akhil Chaudhary, Divesh Kumar Singh, Tanmeya Kansal and Sachin Jain

May 31, 2022

Gene Prediction Using Deep Learning

Akhil Chaudhary[1,a], Divesh Kumar Singh[2,a], Tanmeya Kansal[3,a], Sachin Jain[4,a*]
[a]ABES Institute of Technology

Abstract: Over the last decade, different genomes have been sequenced in both plants and animals. The lower cost of genome sequencing shows that it has a significant impact on the research community in terms of genome annotation. Genome annotations help us understand the biological function of the sequences of these genomes. Gene prediction is one of the most important aspects of genomic annotation and is an open research issue. This article provides a theoretical overview of soft computing technologies for gene prediction. First, I will explain the questions related to the problem of gene prediction. Next, before discussing its application to gene prediction, let's take a brief look at soft computing technology. A list of various soft computing technologies for gene prediction is also available. Finally, some limitations of the direction of current and future research are presented.

Keywords: Gene, Gene prediction, Convolutional Neural Networks, ORF

1. INTRODUCTION

Deciphering the genetic regulatory code that determines gene expression is a key task in molecular biology. Deep learning algorithms have changed prediction in a number of fields, and they have the capability to do so again in genetics. CNNs and other deep learning and machine learning technologies are largely to blame for a revolution in natural language and image processing. These techniques are significantly used in modern world artificial intelligence technologies such as facial recognition, speech recognition, and self-driving vehicles. Deep learning methods are used in molecular biology, agriculture, genetics and medicine.

Essential protein identification is a hot topic and a difficult problem for scientists since essential proteins are necessary for cell survival. To overcome this challenge, a number of computational techniques have lately been presented. Traditional centralized approaches, on the other hand, do not accurately map the topological characteristics of biological networks. Identifying essential proteins is also an issue of asymmetry in learning. Few existing approaches based on shallow machine learning, on the other hand, are intended to deal with unbalanced attributes.

Essential genes are a group of genes that are required for an organism's survival or reproduction. Understanding the basic requirements of an organism, identifying illness genes, and discovering novel drug development targets all involve the ability to predict gene thinking. Using experimental wet lab approaches to find important genes is time-consuming, labor-intensive, and expensive. Many are anticipating essential genes by examining the association between gene essentiality and

all sorts of biological information, thanks to the accumulation of gene essential data in several model species and human cell lines. A technique of calculation has been presented.

1.1. Deep Learning

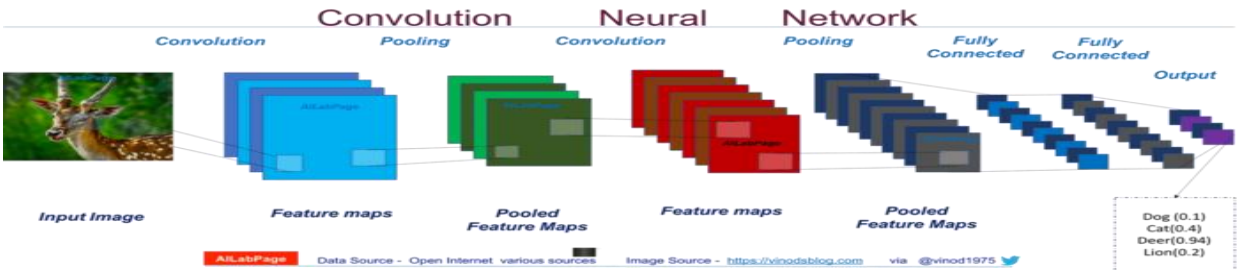
1.1.1. Architecture and Algorithm

Two or two artificial neural networks may readily be described as deep learning architectures (ANNs). More hidden layers are being added in order to improve prediction accuracy. DL employs more hidden layers than standard artificial neural networks. Weighting is used in classic deep neural networks (DNN). To obtain the output[1], A nonlinear activation function is applied to the bias-corrected input value. As a result, the goal of DNN training is to optimize the network weights so that the loss function is as little as possible. The higher-level characteristic is defined using each lower-level feature. In HCS applications, deep learning model technology comprises speech recognition [2], [3] , image analysis [4], text mining[5], health monitoring [6], [7], drug discovery [8], computer vision [9], object identification[10], and deep learning model technology [11]–[14]. The classification of typical deep learning architectures for HCS data processing and HCS-selected applications, particularly illness detection.

One of the applications that has been thoroughly examined utilizing the DL model is the one depicted in this diagram.

1.1.2. Convolutional Neural Network

CNN is a deep learning based architecture that is supervised. It is primarily employed in image analysis applications [15], [16]. A CNN is made up of three layers: a convolutional layer, a pooling layer, and a fully connected layer. Input picture is given to the kernel or filters in the convolution layer to create various feature maps. To keep the number of weights low, the pooling layer shrinks each feature map. Downsampling or subsampling is the term for this procedure. Global pooling, maximum pooling, and average pooling are examples of diverse pooling strategies. After these layers, the 2D feature map is transformed into a 1D vector and then classified using a fully connected layer. The Convolutional Neural Networks approach for image categorization is shown in Figure 1.



Adapted to a [17] Convolutional Neural Networks architecture with two folding layers (Figure 1). A pooling / subsampling layer follows each convolution layer. The result of the last pooling layer ID is sent to the fully connected layer and the output layer.

Figure 1 shows a Convolutional Neural Networks design with two folding layers. A pooling / subsampling layer follows each convolution layer. The result of the last pooling layer is fully connected. supply The last output layer is Layer 35. Deepr was proposed in [18] as an unique end-to-end DL for extracting key information from medical records and predicting anomalies. To forecast unexpected restart after a discharge, convolutional neural networks are used to a collection of discrete components. Then, using DL technology, Cheng [19] investigated the temporal aspects of the patient's EHR. In the second layer of the suggested DL, the convolution operator was applied to the temporal dimension of the patient's EHR matrix. To acquire advantage, the model leverages temporal fusion methods such as early fusion, late fusion, and delayed fusion. The EHR's temporal smoothness during the learning phase.

Dataset

For training and testing the model, two datasets: training and testing datasets were utilized. The datasets were utilized by Orphelia [20] and MGC [21]. The data for training yielded seven million ORFs from 700 bp segments. Gene annotations were obtained from GenBank [22], and fragments were taken from 131 fully sequenced prokaryotic genomes [20]. The data for training was parted into 10 mutually incompatible pieces on the basis of preset GC ranges. According to a recent research [21], developing many models on the basis of GC content is preferable than establishing a single model since fragments with comparable GC content have similar attributes such as codon usage. Segments of 700 bp in length were evaluated using three archaeal and eight bacterial genomes. The genomes utilized in the tests are listed in Table 1 along with their GenBank accession numbers and GC content. The 700 bp pieces were excised at random to provide each training genome a 1-fold genome coverage and each genome in the testing dataset a 5-fold genome coverage.

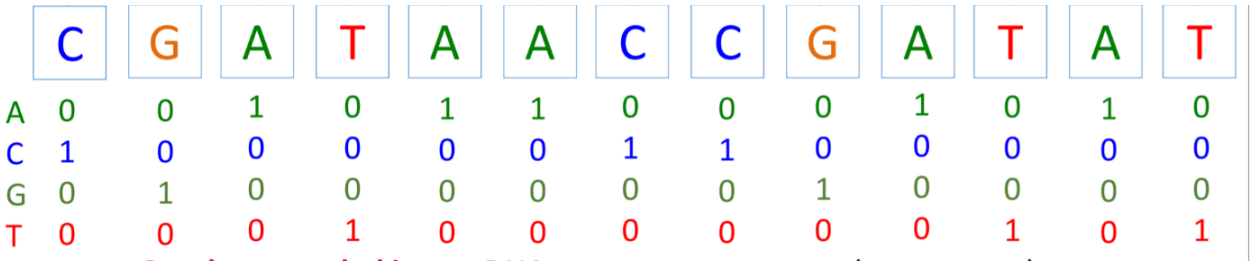
The Proposed Method

Data initialisation, training, Data classification, and Data post-processing are the three primary aspects of our suggested method. Before feeding the ORFs into the CNN models, we numerically encode them. For the classification step, ten CNN models are created. Finally, the candidate ORFs' gene probabilities are approximated using CNN classifiers, and the final gene set is chosen using a greedy approach.

Data Pre-processing

As in previous studies[23]–[25], character-level one-hot encoding to represent the ORFs were used. One-hot encoding is a method for numerically representing categorical data such as

nucleotides. Each nucleotide is shown as a single-hot vector with all zero entries except one at a given position. Each ORF is represented by a L4 matrix of length L. Figure 1 depicts the one-hot encoding of a DNA strand.



Training

In 1998, LeCun [27] created CNNs to recognise handwritten characters on bank checks.Natural language processing, Image and video recognition, and Computational biology are just a few of the domains where CNNs have lately been applied. CNNs are made up of layers of convolutional, nonlinear, pooling, and fully connected. The convolutional layer is the most important part of a CNN. It processes incoming data using a filter matrix, which is a matrix of parameters that is updated by a learning algorithm [26]. Filters with a window size of n glide across the input data to form a feature map, and a dot product is created between the input data and filter parameters. The first convolutional layer may catch sequence patterns, whereas succeeding convolutional layers can capture more intricate patterns [1], [28]. The output is subjected to a non-linear activation function known as the rectified linear unit (ReLU). The pooling layer is then utilized to lower the computational cost, use of memory and number of parameters by controlling over-fitting. It estimates the max result from a small window [15] before calculating the probability of prediction using a fully linked layer.

We utilize 1D CNNs because DNA strands are 1D arrays of nucleotides. Holdout validation is used to separate data into training and validation sets. The training dataset is utilized for 70% of the time, while the validation dataset is used for 30% of the time. The validation set is used to test models that have been trained with various hyperparameters. The findings of the GC range one validation dataset are used to choose hyper-parameters. The number of layers, filters, and filter window size are all depending on the data and application. [25], [29] are two examples. To determine the ideal settings for our problem, we use the Zeng et al testing-based method and train several models with different configurations. To begin, we'll use 16 filters and try out various filter window sizes: 5, 10, 21, 24, and 30. According to our research, a window size of 21 gives a maximum accuracy of 97.71 percent. After that, we put 16, 32, 64, 128, and 200 filters to the test. The 200 filters are the most precise, with an accuracy of 97.92 percent. Then we test two layers with 64 and 200 filters, which result in a 98 percent accuracy. Table 2 shows cross-validation of our model with various filter window widths and number of filters. Because it is adequate for most applications, we chose a batch size of 256. Finally, as shown in Fig. 2, the model with the best performance is picked to build the final CNN models from the whole training dataset.

We calculate the accuracy of CNN-MGP models for each GC range using cross-validation. We employ hold-out validation, which is a type of cross-validation method. The training dataset is divided into two sections: 70% is for training and 30% is for validation. Both the training and validation datasets have the same class percentage as the whole dataset. CNN-MGP is trained on a training dataset before being evaluated on a validation dataset. The accuracy of CNN-MGP models is shown in Table 3 and varies from 98 to 99.1 percent. CNN models with a larger GC range outperform ones with a smaller GC range.

Every model is divided into six tiers. In order to improve CNN performance while reducing overfitting, we add a dropout layer that removes parts of its output [30]. We chose a dropout rate of 50% as our target. After that, the result is converted to a 1D vector and put into a fully connected layer. A 128-neuron fully connected neural network is the fifth layer. Following that, we use a dropout layer. Atlast, a softmax output layer is utilized to determine the gene probability. [31], a lightweight Python deep learning programme, was used to create the CNN models. It operates on GPUs and is built on TensorFlow [32].

Data Classification and Post-Processing

For predicting genes for each metagenomics fragment, we get all full and partial ORFs from that fragment. Any whole ORF starts with an ATG, CTG, GTG, or TTG start codon, continues with a succession of codons, and ends with a stop codon. In an incomplete ORF, there are no start, stop, or both codons. After that, using a one-hot encoding approach, the ORFs are numerically encoded. We next choose an appropriate CNN model to score each ORF based on the GC content of the segment. CNN's output is the chance that an ORF encodes a gene. ORFs having a chance higher than 0.5 are considered candidate genes. There may be overlap between certain candidate genes, and only one candidate gene can be chosen. Genes in prokaryotes have a maximum overlap of 45 kb [33]. As a consequence, a greedy algorithm [20], [21] is used as a post-processing step to remove any overlapping genes and offer a final list of candidate genes. Because the candidate gene with the best likelihood is probably the required gene, we reject any putative ORFs that overlay it by more than 60 bp.

Performance Measures

Gene prediction achieved performance is evaluated by comparing the algorithm's predictions to the true gene annotation in fragments acquired from GenBank [22]. When the ORF aligns with at least 60 bp of a known gene in the same reading frame, it is considered a true positive (TP). A false positive, on the other side, arises when the predicted ORF is incorrectly recognised as a gene (FP). A false negative (FN) is also reported when a gene is incorrectly identified as a non-coding ORF. The prediction performance is assessed using the sensitivity, specificity, and harmonic mean. Sensitivity is a parameter that quantifies the percentage of genes correctly detected and is used to determine the likelihood of detection. Meanwhile, specificity is utilised to examine the

predictability of the findings, which measures the percentage of anticipated genes that are annotated. In contrast to the Orphelia and MGC gene prediction methods, The positive probability score as a scale of specificity was utilized. The means are calculated by the formulas below:

$$\text{Sensitivity}=TP_{gene}/(TP_{gene}+FN_{gene}) \quad (1)$$
$$\text{Specificity}=TP_{gene}/(TP_{gene}+FP_{gene}) \quad (2)$$
$$\text{HarmonicMean}=Spec\times Sens\times 2/(Spec+Sens) \quad (3)$$

Results

CNN-MGP models have been added to the test on a different dataset. Table 1 exhibits segments of 700 bp in length from the testing collection of eight bacterial genomes and three archaeal genomes. The CNN-MGP prediction is compared to the genuine gene annotation in GenBank [22]. The trials are also repeated ten times per genome. The mean and standard deviation for the sensitivity, specificity, and harmonic mean of 10 random replications per genome are computed as shown in Table 4. CNN-MGP has an average specificity of 94.87 percent, an average sensitivity of 88.27 percent, and an average harmonic mean of 91.36 percent. The average standard deviation of the harmonic mean is 0.14 percent.

We compare CNN-MGP against three state-of-the-art gene prediction programmes using the same test dataset: Orphelia [20], MGC [21], and Prodigal [34]. The contrast is shown in Table 4. In terms of sensitivity and harmonic mean, Prodigal surpasses CNN-MGP, whereas CNN-MGP excels Prodigal in terms of specificity. Prodigal, CNN-MGP, and MGC outperform Orphelia. On average harmonic mean, CNN-MGP outperforms Orphelia by 10%; nevertheless, its overall performance is equivalent to MGC's, with both techniques achieving an average harmonic mean of 91% for some genomes and MGC exceeding CNN-MGP for others.

Table 1:

Genomes	GC content (%)	GenBank accession no.
Archaeoglobus fulgidus	48.6	NC_000917
Methanocaldococcus jannaschii	31.4	NC_000909
Natronomonas pharaonis	63.4	NC_007426
Buchnera aphidicola	26.3	NC_002528
Corynebacterium jeikeium	61.4	NC_007164

Chlorobaculum tepidum	56.5	NC_002932
Helicobacter pylori	38.9	NC_000921
Prochlorococcus marinus	31.2	NC_007577
Wolbachia endosymbiont	34.2	NC_006833
Burkholderia pseudomallei	67.7	NC_006350
Pseudomonas aeruginosa	66.6	NC_002516

Table 2:

No. of filters	No. of convolutional layers	Filter window size	Accuracy
16	1	5	97.57
16	1	10	97.68
16	1	21	97.71
16	1	24	97.7
16	1	30	97.65
32	1	21	97.81
64	1	21	97.87
128	1	21	97.89
200	1	21	97.92
-64,200	2	21	98

Table 3:

GC range	Accuracy
0–36.57	98
36.57–41.57	98.4
41.57–46	98.5

46–50.14	98.3
50.14–54.28	98.3
54.28–58.14	98
58.14–61.85	98.3
61.85–65	98.8
65–68.28	99
68.28–100	99.1

Table 4:

Genomes	CNN-MGP			Orphelia			MGC			Prodigal		
	Sp	Sn	HM	Sp	Sn	HM	Sp	Sn	H.M	Sp	Sn	HM
A. fulgidus	94.95\(\pm{0.21}\)	86.15\(\pm{0.19}\)	90.33\(\pm{0.16}\)	88.57\(\pm{0.21}\)	80.58\(\pm{0.17}\)	84.38\(\pm{0.16}\)	95.04\(\pm{0.14}\)	84.13\(\pm{0.23}\)	89.31\(\pm{0.15}\)	95.79\(\pm{0.15}\)	96.13\(\pm{0.08}\)	95.96\(\pm{0.10}\)
M. jannaschii	96.13\(\pm{0.15}\)	93.60\(\pm{0.17}\)	94.85\(\pm{0.16}\)	95.20\(\pm{0.17}\)	90.46\(\pm{0.16}\)	92.77\(\pm{0.14}\)	97.19\(\pm{0.12}\)	92.63\(\pm{0.19}\)	94.85\(\pm{0.13}\)	95.14\(\pm{0.14}\)	95.15\(\pm{0.15}\)	95.15\(\pm{0.12}\)
N. pharaonis	96.17\(\pm{0.12}\)	82.99\(\pm{0.28}\)	89.09\(\pm{0.18}\)	75.99\(\pm{0.34}\)	68.74\(\pm{0.34}\)	72.17\(\pm{0.33}\)	95.28\(\pm{0.12}\)	85.79\(\pm{0.20}\)	90.29\(\pm{0.14}\)	97.48\(\pm{0.10}\)	95.77\(\pm{0.18}\)	96.62\(\pm{0.12}\)
B. aphidicola	97.03\(\pm{0.20}\)	92.67\(\pm{0.41}\)	94.80\(\pm{0.26}\)	95.54\(\pm{0.28}\)	89.40\(\pm{0.33}\)	92.37\(\pm{0.22}\)	98.01\(\pm{0.19}\)	91.11\(\pm{0.37}\)	94.43\(\pm{0.23}\)	96.65\(\pm{0.27}\)	96.97\(\pm{0.26}\)	96.81\(\pm{0.25}\)
C. jeikeium	95.72\(\pm{0.11}\)	87.37\(\pm{0.15}\)	91.35\(\pm{0.09}\)	79.52\(\pm{0.22}\)	74.23\(\pm{0.23}\)	76.79\(\pm{0.22}\)	96.13\(\pm{0.11}\)	87.70\(\pm{0.23}\)	91.72\(\pm{0.17}\)	95.31\(\pm{0.19}\)	94.99\(\pm{0.10}\)	95.15\(\pm{0.10}\)
C. tepidum	94.46\(\pm{0.14}\)	81.09\(\pm{0.28}\)	87.24\(\pm{0.10}\)	77.51\(\pm{0.22}\)	66.95\(\pm{0.23}\)	71.85\(\pm{0.21}\)	93.42\(\pm{0.14}\)	79.08\(\pm{0.24}\)	85.65\(\pm{0.18}\)	94.35\(\pm{0.14}\)	88.15\(\pm{0.19}\)	91.14\(\pm{0.11}\)
H. pylori	96.24\(\pm{0.15}\)	91.22\(\pm{0.13}\)	93.66\(\pm{0.11}\)	94.17\(\pm{0.20}\)	88.99\(\pm{0.22}\)	91.5\(\pm{0.20}\)	97.77\(\pm{0.14}\)	89.70\(\pm{0.22}\)	93.56\(\pm{0.17}\)	95.29\(\pm{0.14}\)	93.07\(\pm{0.14}\)	94.16\(\pm{0.12}\)

P. marinus	98.15\(\pm{0.07}\)	89.12\(\pm{0.13}\)	93.42 \(\pm{0.07}\)	94.41\(\pm{0.20}\)	84.98\(\pm{0.24}\)	89.45\(\pm{0.20}\)	97.71\(\pm{0.11}\)	87.92\(\pm{0.20}\)	92.55\(\pm{0.12}\)	97.52\(\pm{0.17}\)	91.96\(\pm{0.20}\)	94.66\(\pm{0.15}\)
W. endosymbiont	82.71\(\pm{0.38}\)	90.90\(\pm{0.27}\)	86.61 \(\pm{0.27}\)	86.24\(\pm{0.20}\)	83.79\(\pm{0.20}\)	84.99\(\pm{0.20}\)	88.25\(\pm{0.20}\)	87.85\(\pm{0.20}\)	88.05\(\pm{0.20}\)	81.52\(\pm{0.41}\)	92.27\(\pm{0.25}\)	86.56\(\pm{0.31}\)
B. pseudomallei	95.31 \(\pm{0.06}\)	86.99\(\pm{0.12}\)	90.96 \(\pm{0.08}\)	69.54\(\pm{0.31}\)	64.79\(\pm{0.22}\)	67.08\(\pm{0.26}\)	94.79\(\pm{0.13}\)	87.84\(\pm{0.25}\)	91.18\(\pm{0.18}\)	94.28\(\pm{0.09}\)	96.47\(\pm{0.09}\)	95.37\(\pm{0.08}\)
P. aeruginosa	96.73\(\pm{0.08}\)	88.86\(\pm{0.13}\)	92.63 \(\pm{0.09}\)	71.21\(\pm{0.20}\)	68.40\(\pm{0.18}\)	69.78\(\pm{0.19}\)	96.16\(\pm{0.09}\)	91.70\(\pm{0.11}\)	93.88\(\pm{0.08}\)	96.47\(\pm{0.05}\)	97.88\(\pm{0.06}\)	97.17\(\pm{0.05}\)
Average	94.87	88.27	91.36	84.35	78.3	81.19	95.43	87.76	91.4	94.53	94.44	94.43
Average SD	0.15	0.21	0.14	0.25	0.24	0.22	0.15	0.22	0.16	0.17	0.15	0.14

2.3.2. Conclusion

Deep learning has lately received a lot of attention as a potential solution to a variety of bioinformatics problems. The aim of this project is to examine how CNNs affect gene prediction and how they might be utilized to predict genes in metagenomics fragments. CNNs have been successful in predicting DNA binding sites and promoters, among other bioinformatics applications.

2. REFERENCES

[1] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[2] M. Ravanelli, *Signals and Communication Technology Automatic Speech Recognition A Deep Learning Approach hm med Related papers the essence of knowledge Deep Learning Methods and Applications Foundations and Trends … ritesh patra DEEP LEARNING: METHODS AND APPLIC*.

[3] A. Graves, A. R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, no. 3, pp. 6645–6649, 2013, doi: 10.1109/ICASSP.2013.6638947.

[4] G. Litjens *et al.*, “A survey on deep learning in medical image analysis,” *Med. Image Anal.*, vol. 42, no. 1995, pp. 60–88, 2017, doi: 10.1016/j.media.2017.07.005.

[5] J. Bian, B. Gao, and T. Y. Liu, “Knowledge-powered deep learning for word embedding,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8724 LNAI, no. PART 1, pp. 132–148, 2014, doi: 10.1007/978-3-662-44848-9_9.

[6] E. Gawehn, J. A. Hiss, and G. Schneider, “Deep Learning in Drug Discovery,” *Mol. Inform.*, vol. 35, no. 1, pp. 3–14, 2016, doi: 10.1002/minf.201501008.

[7] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016, doi: 10.1016/j.neucom.2015.09.116.

[8] D. Ravi *et al.*, “Deep Learning for Health Informatics,” *IEEE J. Biomed. Heal. Informatics*, vol. 21, no. 1, pp. 4–21, 2017, doi: 10.1109/JBHI.2016.2636665.

[9] X. Chen, S. Xiang, C. L. Liu, and C. H. Pan, “Vehicle detection in satellite images by hybrid deep convolutional neural networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, 2014, doi: 10.1109/LGRS.2014.2309695.

[10] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *Int. J. Rob. Res.*, vol. 34, no. 4–5, pp. 705–724, 2015, doi: 10.1177/0278364914549607.

[11] M. Längkvist, L. Karlsson, and A. Loutfi, “A review of unsupervised feature learning and deep learning for time-series modeling,” *Pattern Recognit. Lett.*, vol. 42, no. 1, pp. 11–24, 2014, doi: 10.1016/j.patrec.2014.01.008.

[12] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, “Single Sample Face Recognition via Learning Deep Supervised Autoencoders,” *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 10, pp. 2108–2118, 2015, doi: 10.1109/TIFS.2015.2446438.

[13] S. E. Kahou *et al.*, “EmoNets: Multimodal deep learning approaches for emotion recognition in video,” *J. Multimodal User Interfaces*, vol. 10, no. 2, pp. 99–111, 2016, doi: 10.1007/s12193-015-0195-2.

[14] G. Rostami, M. Hamid, and H. Jalaeikhoo, “Impact of the BCR-ABL1 fusion transcripts on different responses to Imatinib and disease recurrence in Iranian patients with Chronic Myeloid Leukemia,” *Gene*, vol. 627, no. 1, pp. 202–206, 2017, doi: 10.1016/j.gene.2017.06.018.

[15] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, “A vision of IoT: Applications, challenges, and opportunities with China Perspective,” *IEEE Internet Things J.*, vol. 1, no. 4, pp. 349–359, 2014, doi: 10.1109/JIOT.2014.2337336.

[16] J. Cong and B. Xiao, “Minimizing in Convolutional Neural Networks,” *Int. Conf. Artif. Neural Networks*, pp. 281–290, 2014.

[17] P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh, “Deepr: A Convolutional Net for Medical Records,” *IEEE J. Biomed. Heal. Informatics*, vol. 21, no. 1, pp. 22–30, 2017, doi: 10.1109/JBHI.2016.2633963.

[18] Y. Cheng, F. Wang, P. Zhang, and J. Hu, “Risk prediction with electronic health records: A deep learning approach,” *16th SIAM Int. Conf. Data Min. 2016, SDM 2016*, pp. 432–440, 2016, doi: 10.1137/1.9781611974348.49.

[19] T. K. Ho, “Random decision forests,” *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 1, pp. 278–282, 1995, doi: 10.1109/ICDAR.1995.598994.

[20] K. J. Hoff, M. Tech, T. Lingner, R. Daniel, B. Morgenstern, and P. Meinicke, “Gene prediction in metagenomic fragments: A large scale machine learning approach,” *BMC Bioinformatics*, vol. 9, pp. 1–14, 2008, doi: 10.1186/1471-2105-9-217.

[21] A. El Allali and J. R. Rose, “MGC: A metagenomic gene caller,” *BMC Bioinformatics*, vol. 14, no. SUPPL9, 2013, doi: 10.1186/1471-2105-14-S9-S6.

[22] D. A. Benson *et al.*, “GenBank,” *Nucleic Acids Res.*, vol. 41, no. D1, pp. 36–42, 2013, doi: 10.1093/nar/gks1195.

[23] M. Antonino and D. Gangi, “for DNA Sequence Classification,” vol. 2, pp. 162–171, doi: 10.1007/978-3-319-52962-2.

[24] R. K. Umarov and V. V. Solovyev, “Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks,” *PLoS One*, vol. 12, no. 2, pp. 1–12, 2017, doi: 10.1371/journal.pone.0171410.

[25] H. Zeng, M. D. Edwards, G. Liu, and D. K. Gifford, “Convolutional neural network architectures for predicting DNA-protein binding,” *Bioinformatics*, vol. 32, no. 12, pp. i121–i127, 2016, doi: 10.1093/bioinformatics/btw255.

[26] I. G. and Y. B. and A. Courville, “Deep learning Book pdf,” *Nature*, vol. 29, no. 7553, pp. 1–73, 2016.

[27] K. S. Choi, J. S. Shin, J. J. Lee, Y. S. Kim, S. B. Kim, and C. W. Kim, “In vitro trans-differentiation of rat mesenchymal cells into insulin-producing cells by rat pancreatic extract,” *Biochem. Biophys. Res. Commun.*, vol. 330, no. 4, pp. 1299–1305, 2005, doi:

10.1016/j.bbrc.2005.03.111.

[28] T. F. Gonzalez, “Handbook of approximation algorithms and metaheuristics,” *Handb. Approx. Algorithms Metaheuristics*, pp. 1–1432, 2007, doi: 10.1201/9781420010749.

[29] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, “Deep learning for computational biology,” *Mol. Syst. Biol.*, vol. 12, no. 7, p. 878, 2016, doi: 10.15252/msb.20156651.

[30] B. Mele and G. Altarelli, “Lepton spectra as a measure of b quark polarization at LEP,” *Phys. Lett. B*, vol. 299, no. 3–4, pp. 345–350, 1993, doi: 10.1016/0370-2693(93)90272-J.

[31] et al (2015) K. D. learning library for theano and tensorflow. Chollet F, “download.” https://keras.io/

[32] C. Gerard, “TensorFlow.js,” *Pract. Mach. Learn. JavaScript*, pp. 25–43, 2021, doi: 10.1007/978-1-4842-6418-8_2.

[33] A. S. Warren and J. C. Setubal, “The Genome Reverse Compiler: An explorative annotation tool,” *BMC Bioinformatics*, vol. 10, 2009, doi: 10.1186/1471-2105-10-35.

[34] L. J. H. Doug Hyatt, Gwo-Liang Chen, Philip F LoCascio, Miriam L Land, , Frank W Larimer, “Integrated nr Database in Protein Annotation System and Its Localization,” *Nat. Commun.*, vol. 6, no. 1, pp. 1–8, 2010, [Online]. Available: http://dx.doi.org/10.1016/B978-0-12-407863-5.00023-X%5Cnhttp://www.nature.com/doifinder/10.1038/ismej.2009.79%5Cnhttp://www.nature.com/doifinder/10.1038/nature09916%5Cnhttp://dx.doi.org/10.1038/srep25982%5Cnhttp://dx.doi.org/10.1038/ismej.2010.144%5Cnhttp