UNIVERSITY OF CALIFORNIA
RIVERSIDE

Accurate Prediction Methods on Biomolecular Data

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Md Abid Hasan

December 2019

Dissertation Committee:

Dr. Stefano Lonardi, Chairperson
Dr. Christian Shelton
Dr. Eamonn Keogh
Dr. Karine G Le Roch

The Dissertation of Md Abid Hasan is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

Firstly, I am grateful to Almighty Allah for giving me the strength to continue through this challenging path of achieving the highest degree and honor of my life. Many times during my Ph.D. life, I felt vulnerable, broken and lost. I sought refuge to HIM and HE always showered his blessings on me even though I do not deserve HIS blessings.

Secondly, I am grateful to my advisor, Prof. Stefano Lonardi – an excellent educator, a brilliant researcher, and a genuine, humble human being. He has been patient with me, supported me during my prolonged period of failure in obtaining any substantial outcome in research. He allowed me space to grow and encouraged me to learn and directed me to success in research. For someone like me who was alien to the environment after coming to a foreign land, Prof. Lonardi has been a figure of trust and reliability from day one till the very end. I will carry his teachings with me, not just as a researcher, as an educator but also as a human being, for the rest of my life and will try to improve myself from the time I had him as a guardian.

I'd also like to thank all my committee members – Prof. Christian Shelton, Prof. Eamonn Keogh, and Prof. Karine G Le Roch for their time and valuable suggestion in my research which helped me put forward better work and ultimately helped complete my Ph.D. works.

I am very grateful to my family. They always had my back. Whenever I felt weak and frustrated, they were there for me. Their endless support and belief always encouraged me and helped me go through this seemingly unending long journey. I will always be indebted to my mother Anjuman Ara Begum, my father Mosharraf Hossain, my brother

Jaowad Hasan and my sister Dr. Maherun Nesa for their unconditional, unlimited support. They have always been my anchor and without them I am nothing.

I can not forget my friends in Riverside who were there for me during some of the most difficult times I spent during my Ph.D. They have gone out of their way to drag me into their life, accepted me as their friend and for that, I will always be grateful to them. Especially, Rahat Mahmud, Ahmed Mahdi, and Gouri Chakraborty – you guys have accepted me when I was most vulnerable, saved me and loved me beyond my imagination. I will cherish your friendship for the rest of my life. Also all my friends from the Bangladesh community in Riverside, I am very much grateful for your support.

I thank all my badminton buddies with whom I become very good friends in such a short period of time. You guys are my only escape during this Ph.D. life. and I enjoyed every moment I spent with you. Ming, Hua, Charles, Ji, Max, Tushar, Yuki, Ken, Zi, and Yuxiang – I am forever grateful for your friendship and your company.

Finally, my labmates, Dr. Weihua, Dr. Abbas, Dr. Hamid, Dr. Rachid, Dr. Hind, Miguel, Huang, Parker, Iram, Dipankar, and Qihua, thank you for all your support and help. I could not have asked for any better lab-mates and friends than you. You have accepted me and valued me more than I deserved. Thank you very much and I hope we all stay connected just like we were in our lab.

For everyone else I haven't mention, and/or I can not mention, thank you, everyone. This journey was never easy and it is impossible to walk alone in this path. I stand where I am because of all of you. I am grateful and I thank you. May God bless all of you.

To my mother Anjuman Ara Begum for everything she sacrificed our family.

To all my friends and family.

ABSTRACT OF THE DISSERTATION

Accurate Prediction Methods on Biomolecular Data

by

Md Abid Hasan

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2019
Dr. Stefano Lonardi, Chairperson

With the recent advancements in sequencing technologies, molecular biologists are producing ever-increasing amounts of biomolecular data. Extracting useful information from these massive data sets requires efficient and effective data mining and machine learning methods. In this dissertation, we explore the use of supervised machine learning (ML) to solve some challenging classification problems in molecular biology.

First, we devise an ML model for classifying cancer types from very sparse somatic point mutation data. Accumulation of mutation and epigenetic modifications in somatic cells results in various cancer. For this purpose, we propose a method called mClass for efficient feature (gene) ranking that uses clustering, normalized mutual information and logistic regression. We show that somatic mutation data has sufficient discriminative power for cancer type classification.

Next, we address the problem of gene essentiality prediction in microbes. Essential genes are significant to identify since their function is vital for the survival of the organism. Our proposed deep learning architecture called DeeplyEssential exclusively uses features ex-

tracted from the primary sequence of genes and their corresponding proteins, to maximize the utility and practicality of the tool. DeeplyEssential achieved state-of-the-art performance over previously proposed methods as well as expose and study a hidden performance bias affected previous models.

Finally, we consider the problem of predicting the enhancer regions in the human genome from chromatin data. Enhancers contribute to the transcription of target genes. We propose a convolutional neural network framework named Epi2En that takes advantage of epigenetic ChIP-seq data. Epi2En's classification performance is not only very strong on cross-validation experiments, but also when testing across different cell-lines.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the recent advancement in sequencing technologies and other modern technologies in the Life Sciences, one would be inclined to believe that Bioinformatics is a relatively recent branch of research. However, the foundations of Bioinformatics were laid in the early 1960s when the desktop computer was still a concept and DNA was yet to be sequenced [50]. Since its beginning about 50 years ago, Bioinformatics has evolved through many phases. As more genomic data was being produced and more computational power was becoming available, scientists started to develop computational models for genomic data analysis *in silico*. The path towards modern Bioinformatics was paved through the advancement in molecular biology that allowed DNA manipulation and efficient software development that helps analyze such genomic data. In the 1990s to 2000s Bioinformatics research went through major improvements, especially due to advancement of sequencing technologies and the increasing processing power available, which together led to historical scientific achievements, like the completion of the Human Genome Project.

In recent years, these sequencing and processing technologies have continued to improve, which has led to generation of massive amounts of sequencing data. According to [30], the European Bioinformatics Institute (EMBL-EBI) that curates and analyses Life Science data, stored about 120 petabytes of data by the end of 2016, which grew to 160 petabytes at the end of 2018. Figure 1.1 shows the increase in the amount of storage at EMBL-EBI over the last few years. The EMBL-EBI databases includes resources like ArrayExpress [4], BioSamples [31], ChEMBL [98], Ensembl [34], GWAS catalog [14], InterPro [100], Pfam [39], PRIDE [110], RNACentral [133], SIFTS [35], UniProt [135] and many more. These data resources include sequencing data, gene expression data, mass spectrometry data, microarray data, protein data, among others. Figure 1.2 shows the exponential growth of these databases by platform and by data resources. A significant amount of research in Bioinformatics depends on the integration of multiple heterogeneous types of data. The integration of various resources available in EMBL-EBI is shown in Figure 1.3 [29]. Being able to process such large interconnected datasets requires efficient data mining techniques and vast amounts of computational power.

In the last few years, an increasing number of machine learning techniques have being used for knowledge extraction in various areas of Bioinformatics [78]. Determining the location and splicing structure of a gene, identifying the presence of regulatory elements, predicting the 3D structure of proteins, are examples of many problems in Genomics where machine learning has been successfully used. A variety of computational techniques have been applied in systems biology, evolutionary biology, and biological text mining. A list of

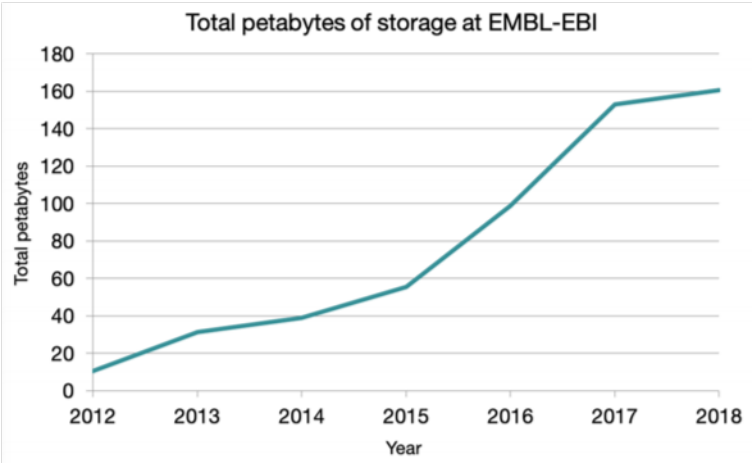

Figure 1.1: Disk capacity, storing raw data including all backups and space reserved for immediate future growth at EMBL-EBI (source [30])

machine learning methods in Bioinformatics includes Bayesian classifier, logistic regression, support vector machine, random forests, k-nearest neighbor, k-means clustering, hierarchical clustering, as well as probabilistic graphical models (e.g., hidden Markov models) for knowledge discovery and stochastic heuristics for optimization. Figure 1.4 (obtained from [78]) illustrates some of the bioinformatics problems solved using machine learning methods.

When deep learning techniques became very popular for image classification (circa 2012), researchers started to explore their application in bioinformatics problems [99]. Because of its ability to extract valuable knowledge from big data, deep learning has become a popular tool for research in bioinformatics. Conventional machine learning models heavily depend on proper feature representation which requires domain expertise. Deep learning models overcame this limitation due to their ability to identify complex interactions among features without laborious pre-processing of the input data. Figure 1.5 illustrates the increase in popularity of deep learning methods in the scientific literature.

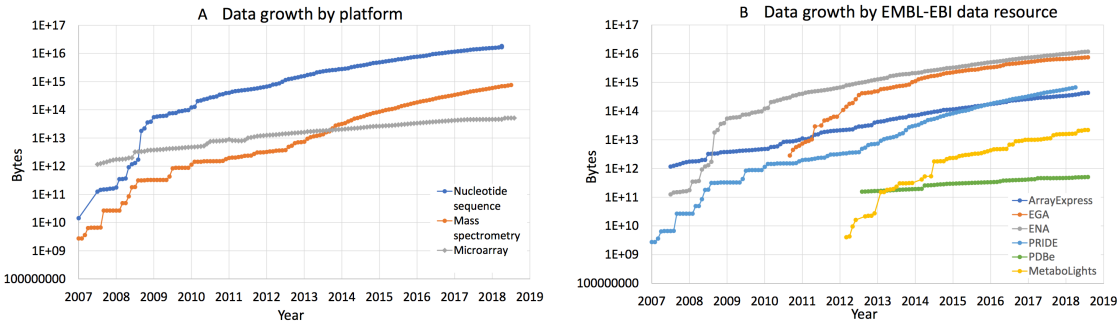

Figure 1.2: Accumulation of data by data type (left); accumulation of data by data resource (right) (source [30])

As we explore the possibility of using statistical and machine learning model efficiently in bioinformatics research, we found that there are areas where a more accurate and efficient application of such techniques are desirable. From the necessity of finding better application and learning models, we have applied various machine learning techniques that further improve the performance of the problems we tried to address in this thesis.

The overall theme of this dissertation is to build accurate prediction models for a variety of biomolecular data. In Chapter 2 we explore the problem of cancer type classification using somatic point mutation data. Our proposed method selects a significant subset of the genes by employing normalized mutual information and uses a logistic regression as the classifier.

In Chapter 3 we introduce a deep neural network (DNN) framework for the prediction of essential genes across thirty bacterial species using only sequence data. Our model makes minimal assumptions about the input data to carry out the prediction thus maximizing its practical application compared to the existing prediction model that requires

Figure 1.3: Dependencies between different databases and resources at EMBL-EBI, determined by the exchange of data (source [29])

more complex data such as structural and topological features. Special attention was given to the "data leak" issue, that is the presence of copies of orthologous genes in the dataset which causes hidden biases in the prediction model.

In Chapter 4 we explore the use of a convolutional neural network (CNN) for the prediction of genome-wide enhancers (distal *cis*-regulatory elements), across multiple human cell-lines, using twelve epigenetics markers. Due to the remote location relative to the target gene(s), their functional complexity, and their lack of discriminating motifs and evolutionary-conserved sequences, it is challenging to identify enhancer regions in the genome. Our CNN based prediction model identifies complex interactions among the fea-

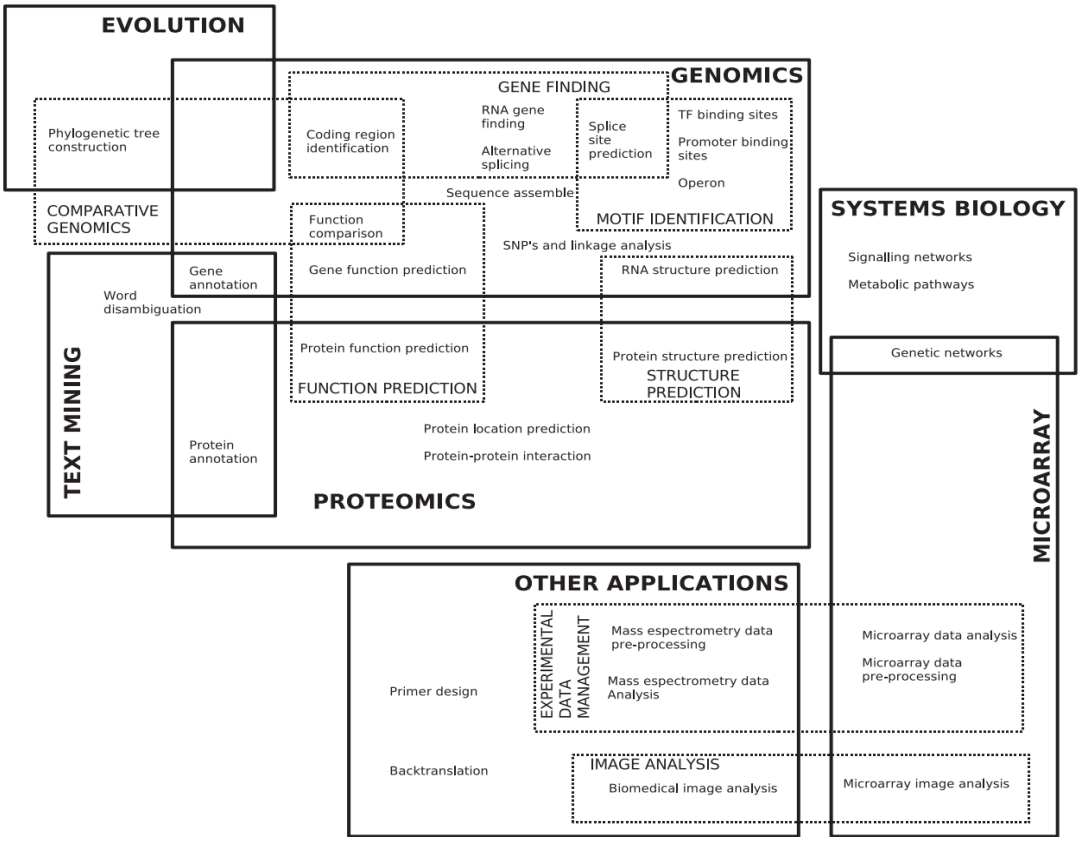

Figure 1.4: Application of machine learning techniques in Bioinformatics (source [78])

tures for an improved, more accurate enhancer region prediction than previously published methods.

Finally, in Chapter 5 we present a supervised classifier for predicting stable/unstable nucleosomes in *P. falciparum*. Due to the lack of ground truth, our model takes the prediction output from ThIEF – that identifies genome-wide trajectories of epigenetics marks [115]. For this prediction model, we have used features that are extracted from primary DNA sequences and binding preferences of DNA to identify the dynamics of nucleosomes.

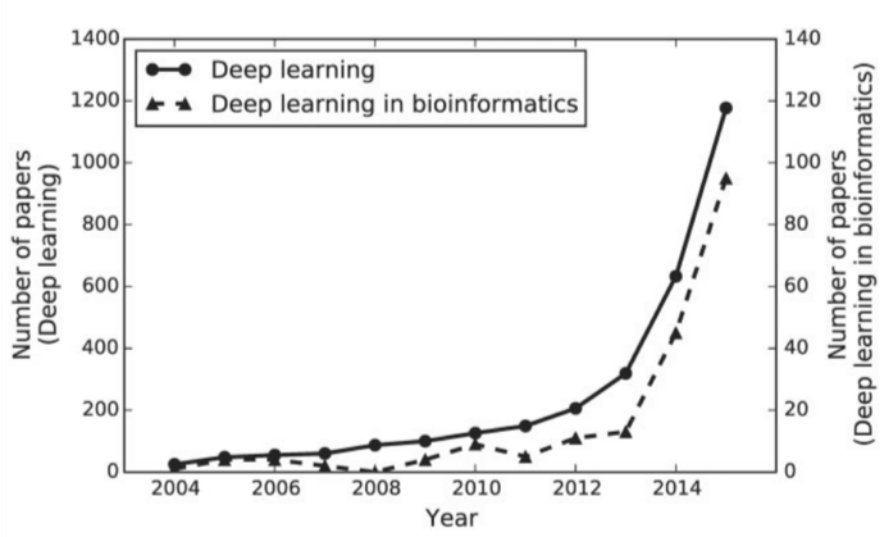

Figure 1.5: Deep learning article approximation based on the search results on http://www.scopus.com with the two queries: 'Deep learning', 'Deep learning' AND 'bio*'. [99]

Chapter 2

mClass: Cancer type classification with somatic point mutation data

2.1 Introduction

Cancer is a complex disease that results from an accumulation of DNA mutations and epigenetic modifications in somatic cells. Remarkable scientific progress has shed light on almost every biological aspect of this disease. Despite this progress, cancer is still one of the most challenging disease of our time with an increasing numbers of new cases and resulting in 14.6% of all human death each year [131]. Not all tumors are cancerous and not all cancers are the same. There is no single test that can diagnose cancer type with perfect accuracy. The diagnosis process requires careful examination and extensive testing to determine whether a person has cancer and which type. Traditional cancer diagnosis method involves lab tests, genetic tests, tumor biopsies, etc. The effective differentiation

of cancers with similar histopathological appearance can indicate the most effective drug treatment and increase survival rates (see, e.g., [53, 91, 55, 146]).

Technological advancements in sequencing technologies has resulted in a dramatic increase in the quantity and quality of sequencing data related to cancer, now available in databases such as The Cancer Genome Atlas [134] and the International Cancer Genome Consortium [147]. These vast repositories provide genomic data from thousands of patients across different cancer subtypes [3]. The abundance of this data has enabled researchers to devise new statistical approaches for the accurate identification of cancer types and subtypes. Cancer classification methods use gene expression data and/or somatic point mutation such as copy number variation, translocations and small insertions and deletions. Several methods have been proposed to accurately predict cancer types and subtypes (see, e.g., [11, 24, 27, 146]). The classification of cancer based on the somatic point mutation data can be challenging because of the high dimensionality and sparsity of the data. In cancer patients only a few genes are mutated with high frequency, while most of the genes have a low rate of mutation [79].

The literature on cancer classification methods is extensive. For instance, in [86] the authors proposed a pan-cancer classification method based on gene expression data. They used over nine thousand samples for 31 cancer types to train a method in which a genetic algorithm carries out the gene selection and a nearest neighbor method is used as a classifier.

The authors of [26] proposed to find discriminatory gene sets by measuring the relevance of individual genes using mean and standard deviation of each sample to the class

centroid. In [142] the authors introduced new scoring functions to design a stable gene selection method. Their method scores genes based on the assumption that discriminatory genes have different mean values across different classes, small intra-class variation and relatively large inter-class variation.

The authors of [15] combined the clustering gene selection with statistical tests such as T-test and F-test and the gene selection method proposed in [26] to deal the high dimensionality in gene expression data. Genes are assigned to clusters if they are close to the centroids after applying k-means clustering.

In [146], the authors proposed a deep neural network for the classification of multiple cancer types from somatic point mutation data, called DeepGene. To the best of our knowledge, DeepGene is the state-of-the-art for multiple cancer classifications using somatic point mutation data. DeepGene clusters genes based on mutation occurrence and uses a sparse representation to index non-zero elements. The data is then fed into a fully connected deep neural network that learns specific cancer types.

In this paper, we address the shortcomings of existing methods dealing with the sparsity and high-dimensionality of somatic point mutation data by proposing an efficient feature selection method based on information theory. A logistic regression model demonstrates the effectiveness of our approach for cancer type classification. Although in a medical setting the task of predicting cancer type from somatic point mutation data might not be practical, here we investigate the fundamental question on whether somatic point mutation data has sufficient discriminative power to allow for cancer type classification.

2.2 Methods

Given m individuals affected by cancer, the input to our feature selection method is composed of the class labels, i.e., the cancer type for the m individuals, and the mutation frequency of all genes for the m individuals. Selected features are then fed into a classifier as described below.

Let n be the number of human genes for which somatic point mutation data is available. Let $C \in \{1 \dots l\}^m$ be the vector containing the class labels where l is the number of cancer types, and let $G \in \{0 \dots k\}^{m \times n}, k \in \mathbb{N}$ be the matrix representing the number of mutations observed in each gene (i.e., $G(i, j) = k$ if gene i has k mutations in sample j).

The significance of a gene being involved in a particular type of cancer depend on its mutation frequency. Genes with higher mutations are expected to be more relevant for the causation of cancer [25]. In our method, we disregard genes that contain less than $t\%$ mutations across all samples. This filtering step removes non-significant genes from further consideration thus reduce the adverse impact of the data sparsity. Our feature selection model has two steps. First, we cluster genes based on their pairwise similarity. Then, we rank genes using a normalized mutual information criterion [43].

2.2.1 Gene clustering

Grouping similar genes into clusters allows our method to identify and eliminate redundant genes within a cluster without compromising the efficiency of the feature selection. The reduction of data also reduces the complexity of downstream steps. Since G is a

sparse matrix, we use the cosine similarity because of its good mathematical properties on sparse vectors. Given two n-dimensional vectors X and Y the cosine similarity is defined as

$$s(X,Y) = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}$$

where X_i and Y_i are the i-th components of vector X and Y. Gene p is assigned to the cluster of gene q if the cosine similarity between row vectors $G[:,p]$ and $G[:,q]$ is higher than a predefined threshold e. According to this procedure, it is possible that the same gene could end up in multiple clusters. To select unique genes out of these clusters, we rank the genes based on mutation count and mutual information with the class label within the cluster as described next.

2.2.2 Normalized mutual information

Our gene selection method relies on an information theoretic measure that evaluates the predictive ability of each gene. Let X be a discrete random variable where each event $x \in X$ occurs with probability $p(x)$. The *entropy* $H(X)$ of variable X is the sum of the information content of each discrete event weighted by the individual event probability, that is $H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$.

Given two discrete random variables X and Y with joint probability $p(x,y)$ and marginal probabilities $p(x)$ and $p(y)$, the *conditional entropy* of variable Y conditioned on variable X is defined as $H(Y|X) = \sum_{x \in X, y \in Y} p(x,y) \log_2(p(x)/p(x,y))$. Similarly, $H(X|Y) = \sum_{x \in X, y \in Y} p(x,y) \log_2(p(y)/p(x,y))$. We have that $H(Y|X) = H(Y)$ iff X and Y are independent random variables. The *mutual information* $I(X,Y)$ is the gain of infor-

mation about random variable X due to additional information from random variable Y, that is

$$I(X,Y) = H(X) - H(X|Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Given a set F of features (the set of genes in G in this case) and class variables C, the feature selection based on mutual information finds a subset $S \subset F$ such that the mutual information $I(C,S)$ is maximized. In order to achieve that goal we use the Normalized Mutual Information based Feature Selection (NMIFS) technique. NMIFS is a heuristic algorithm that selects one feature at a time. NMIFS differs from other mutual information based feature selection technique such as MIFS [7], MIFS-U [76] and mRMR [108] in that it does not depend on the parameter used to control the redundancy penalization. Also NMIFS does not assume that the random variables have uniform probability distribution.

Given features $f_i \in F - S$ and $f_s \in S$ we express the mutual information as

$$I(f_i, f_s) = H(f_i) - H(f_i|f_s) = H(f_s) - H(f_s|f_i) \tag{2.1}$$

where $H(f_i)$ and $H(f_s)$ are the entropies and $H(f_i|f_s)$ and $H(f_s|f_i)$ are conditional entropies.

The mutual information $I(f_i, f_s)$ is non-negative, and attains its maximum at $\min\{H(f_i), H(f_s)\}$. We can define the normalized mutual information between f_i and f_s as

$$\text{norm}I(f_i, f_s) = \frac{I(f_i, f_s)}{\min\{H(f_i), H(f_s)\}} \tag{2.2}$$

The *average normalized mutual information* is a measure of redundancy between f_i and $f_s \in S$ for $s = 1, \ldots, |S|$ and it defined as

$$\frac{1}{|S|} \sum_{f_s \in S} \text{norm}I(f_i, f_s)$$

where $|S|$ is the cardinality of subset S. Our gene selection criterion selects a gene $f_i \in F - S$ that maximizes

$$J(C, f_i) = I(C, f_i) - \frac{1}{|S|} \sum_{f_s \in S} \text{norm}I(f_i, f_s) \tag{2.3}$$

where $I(C, f_i)$ is the mutual information between feature f_i and class variable C.

2.2.3 Feature selection

A sketch of mClass' algorithm is shown as Algorithm 1. The algorithm first determines the number of mutations of each gene from the input matrix G. Then it computes the cosine similarity between all pairs of genes that have a mutation percentage across all sample of at least $t\%$. Genes are assigned to the same cluster when their similarity exceeds threshold e. The process assigns each gene to one or more clusters. The top v genes from each clusters are selected into a representative list R'.

Next, mClass collects the unique set of genes U from the representative set R'. It then calculates the mutual information between all features/genes $f_i \in U$ and the class variable C. To calculate Equation 2.2 and Equation 2.3 mClass discretizes the gene mutation values into d equal-width bins. The gene $\hat{f}_i$ which has the maximum mutual information with the class variable C is selected as the first feature in S (S is the final set of ranked genes). That gene is then removed from U. For all the other genes in U mClass first calculates the normalized mutual information between all pair of genes in U and S using Equation 2.2. A gene $f_i \in U$ is selected when it maximizes Equation 2.3. The gene is then added to S and removed from U. This process is repeated until all genes are given a rank in the ordered set S. Instead of deciding on a predefined number of features *a priori* to be

used in the classifier, we select a variable number of genes in S based on their ability to classify the data.

2.2.4 Cancer type classifier

As said, we employ a logistic regression (LG) multi-class classifier for a given number of genes in the ranked set S. The linear model describes the probabilities describing the possible outcome of a single trial using logistic function. Here we use a One-vs-Rest (OvR) for the multi-class classification implementation with L_2 regularization. For the binary case, the L_2-regularized logistic regression optimizes the following cost function

$$\text{minimize}_w \sum_{x,y} \log(1 + \exp(-w^T x.y)) + \lambda w^T w) \tag{2.4}$$

The objective is to find the feature weights (w) that minimizes the cost function in Equation (2.4). Here x is the feature vector (genes) and y is the class label. The hyper-parameter λ used to control the strength of regularization was left as the default value (as defined by `scikit-learn`). As said, the classifier is fed the genes in S incrementally. To determine the final set of features we select genes based on their ability to accurately classify the dataset. The model decomposes the optimization problem in Equation (2.4) in a OvR fashion so that the binary classifier can be trained on all classes.

2.3 Experimental Results

In this section, we describe the experimental setup, i.e., datasets and the parameters used in the feature selection and classification, as well as other implementation details.

Data: Gene mutation data $G \in \{0, k\}^{m \times n}$, similarity measure threshold e, mutation count threshold t, discretization value d, v, class variable C
Result: Ordered set of genes S
set $R \leftarrow \emptyset$;
for *each gene* $f_i \in G$ **do**
  if *number of mutation of* $f_i > t$ **then**
    $R \leftarrow R \cup \{f_i\}$;
  end
end
set $CL \leftarrow \emptyset$;
for *each gene* $f_i \in R$ **do**
  create a new cluster in CL for f_i;
  for *each gene* $f_j \in R$, $j \neq i$ **do**
    if *cosine similarity* $s(f_i, f_j) > e$ **then**
      assign f_i and f_j to same cluster in CL
    end
  end
end
set $R' \leftarrow \emptyset$;
for *each cluster* $cl \in CL$ **do**
  set $R' \leftarrow R' \cup \{\text{top } v \text{ genes in } cl\}$
end
collect unique genes $U \leftarrow set(R')$;
discretize gene mutation values in d equal-width bins;
select the first feature $\hat{f}_i = \text{argmax}_{f_i \in U}\{I(C; f_i)\}$;
set $U \leftarrow U - \{\hat{f}_i\}$;
set $S \leftarrow \{\hat{f}_i\}$;
for *each gene* f_i *in* U **do**
  calculate $I(f_i; f_s)$ for all pairs (f_i, f_s) with $f_i \in U$ and $f_s \in S$;
  select feature $f_i \in U$ that maximizes J in Equation (3);
  set $U \leftarrow U - \{f_i\}$;
  set $S \leftarrow S \cup \{f_i\}$;
end
return ordered set S;

Figure 2.1: mClass feature selection method

Data preprocessing, feature selection and classification evaluation steps were implemented in Python. All tested classifiers are available from the Python package `scikit-learn`.

2.3.1 Datasets

We used two cancer datasets to test mClass. The first dataset is a twelve-type cancer dataset from The Cancer Genome Atlas (TCGA) [134]. The dataset was assembled by selecting the genes across all samples for all cancer types that contain mutations. Table 2.1 shows the basic statistics of each cancer type. Observe that the number of samples and the number of mutations varies significantly across cancer types. After removing samples that have less than five mutations across all genes, the dataset contained 3,151 samples and 23,236 genes. The second dataset from TCGA contains four cancer types, namely COAD, SKCM, LAML and KIRC. It contains 1,043 samples with a total of 363,285 mutations across 25,286 genes. Details about this dataset and the corresponding experimental results are discussed in Section 2.3.5.

2.3.2 Parameters

mClass' feature selection uses four parameters: the similarity measure threshold e for the clustering step, the minimum mutation count threshold t to eliminate non-informative genes, the number v of top genes selected from each cluster and the number of bins d used for discretizing gene mutation values (see Algorithm 1).

In our experiments, parameter t was set to 1 which has the effect of disregarding genes with less that 1% mutation across the samples. As said, the pairwise gene similarity

Cancer type	*Number of samples*	*Number of mutations*
ACC	90	18,272
BLCA	130	37,948
BRCA	982	83,360
CESC	194	45,293
HNSC	279	49,264
KIRP	161	13,640
LGG	286	9,228
LUAD	230	68,270
PAAD	150	30,123
PRAD	332	11,802
STAD	289	130,050
UCS	57	10,129
Total	3,180	507,379

Table 2.1: Sample and mutation statistics for the twelve-type cancer dataset

is calculated using the cosine similarity measure and genes are assigned into same cluster if the similarity between them is greater than the similarity threshold e. The algorithm then selects the top $v\%$ genes from each cluster for gene ranking step. The values for e, t, v and d were selected experimentally based on ability of the method to accurately classify the datasets using the selected number of features. For instance, Table 2.2 shows the classification accuracy of mClass+LG (mClass's feature selection followed by logistic regression) on the twelve-type cancer dataset, for various choices of e. Based on this analysis, we selected $e = 0.55$. Similarly, we tested the values of v in the range 5%-25%, and we obtained the highest classification accuracy with $v = 10\%$. A similar experimental analysis (not shown) indicated that $d = 5$ was the optimal choice for these datasets. Incidentally, the same value of d was used in [17].

2.3.3 Evaluation metrics and comparison with DeepGene

We have used the evaluation metrics introduced in [146] to compare the results. All evaluation experiments were performed by randomly selecting 90% of the input data as training data and 10% of the input as testing data. We compared the ten-fold cross validation accuracy of mClass+LG (mClass's feature selection followed by logistic regression) and testing accuracy against state-of-the-art DeepGene [146].

Similarity threshold (e)	*Classification Accuracy*
0.50	0.708
0.55	**0.718**
0.60	0.715
0.65	0.715
0.70	0.715
0.75	0.715

Table 2.2: Classification accuracy of mClass+LG as a function of similarity threshold e on the twelve-type cancer dataset

As said, mClass selects the optimal number of features in a forward selection fashion. We compared mClass' cross-validation results with with DeepGene, which employs a convolutional neural network (CNN) as the classifier. The performance of DeepGene was calculated in three different configuration: clustered gene filter and indexed sparsity reduction, only cluster gene filter and only indexed sparsity reduction.

Method	*Cross-validation Accuracy*
DeepGene (CGF + ISR)	0.655
DeepGene (CGF)	0.638
DeepGene (ISR)	0.649
mClass+LG	**0.675**

Table 2.3: Ten-fold cross validation accuracy for mClass+LG and DeepGene (three configurations) on the twelve-type cancer dataset

The ten-fold cross-validation results between mClass and three configuration of DeepGene on the twelve-type cancer dataset is shown in Table 2.3. Observe that the classification accuracy of mClass outperformed all three configurations of DeepGene proposed in [146]. The classification accuracy of mClass is more than 3% higher than the best configuration of DeepGene.

We also compared the testing accuracy of mClass with (i) the best configuration of DeepGene and (2) LG on the full dataset (i.e., no feature selection). The logistic regression classifier in mClass uses balanced weights to counter the imbalance in the number of samples in the dataset. Using the forward feature selection technique described in Algorithm 1, the testing accuracy of the classifier was measured by adding ranked gene one at a time. Figure 2.2 shows the progression of forward feature selection. mClass obtains the best testing accuracy $(TP + TN)/(TP + TN + FP + FN)$ of 0.718 using a collection of top 3,676 genes which is 9.6% higher than the accuracy obtained by the best configuration of DeepGene with an average precision $TP/(TP+FP)$ of 0.74, recall $TP+(TP+FN)$ of 0.718 and F-Score $(2 \times precision \times recall)/(precision + recall)$ of 0.711 as shown in Table 2.5. Figure 2.3 illustrates the confusion matrix for the twelve-type cancer dataset. Observe that with mClass + LG, false positives rate is highest for BRCA while BLCA has the highest rate of false negatives. Table 2.4 summarizes the testing accuracy of these three methods.

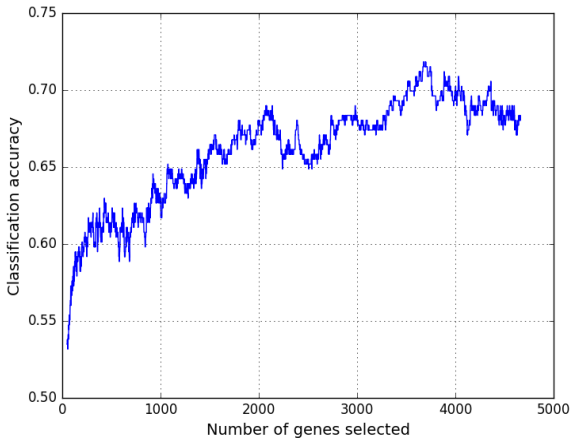

Figure 2.2: Classification accuracies as a function of the number of feature (genes) selected

Method	*Classification accuracy*
Full dataset (no feature selection)	0.677
DeepGene (CGF+ISR)	0.655
mClass+LG	**0.718**

Table 2.4: Testing accuracies of mClass+LG, DeepGene and LG on full dataset (twelve-type cancer dataset)

Cancer type	*Precision*	*Recall*	*F-Score*	*Support*
ACC	1.00	0.83	0.91	12
CESC	0.88	0.47	0.61	15
UCS	0.33	0.50	0.40	2
PAAD	0.80	0.92	0.86	13
KIRP	0.89	0.64	0.74	25
STAD	0.70	0.50	0.58	32
LGG	0.95	0.91	0.93	23
BLCA	0.67	0.38	0.48	16
HNSC	0.81	0.46	0.59	28
PRAD	0.68	0.78	0.73	50
LUAD	0.90	0.75	0.82	12
BRCA	0.62	0.88	0.71	88
Average/Total	0.74	0.72	0.71	316

Table 2.5: Classification results on twelve-type cancer dataset

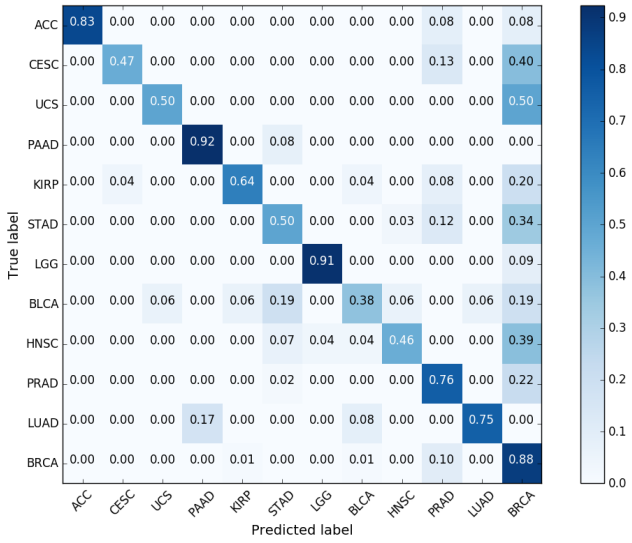

Figure 2.3: Normalized confusion matrix for the twelve-type cancer dataset

2.3.4 Testing other classifiers

As said, mClass+LG uses a logistic regression as the classifier for the cancer classification datasets. We have tested the classification accuracies of other classifiers following mClass' feature selection. We employed Support Vector Machine (SVM) both with the linear and RBF kernel, k-nearest neighbor (KNN), Naive Bayes and Random Forest. All the classifiers were available from the Python package `scikit-learn`.

To classify the data using SVM with the RBF kernel, we optimized the parameter C and γ using 10-fold cross validation (keeping other parameters to default). The highest accuracy was obtained with $C = 2e^2$ and $\gamma = 2e^{-5}$. We have used the same parameter C for the linear kernel version of the SVM. The classification with KNN employed Euclidean distance and Pearson correlation coefficient. The 10-fold cross validation showed an optimal

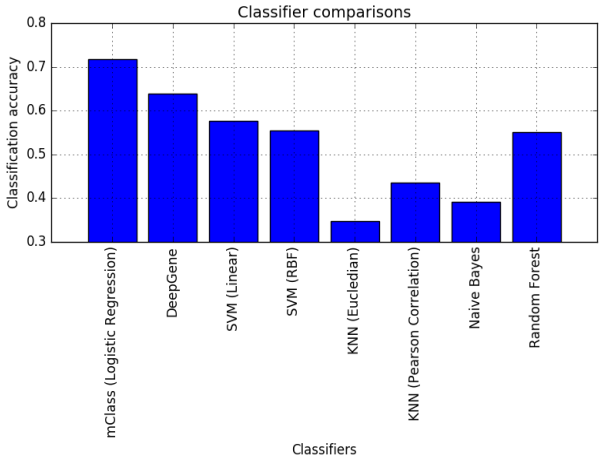

Figure 2.4: Classification accuracy of mClass+LG, DeepGene and other classifiers applied to the features selected by mClass

accuracy of 0.316 for Euclidean distance using a threshold of 3 and an accuracy of 0.436 with the Pearson correlation coefficient using a neighborhood size of 4. The ensemble Random Forest classifier's employed a maximum of 1,000 trees in the forest. We set the minimum number of samples required to split an internal node to 9. All other parameters were set to default.

The performance of the various classifier is shown in Figure 2.4. The experimental results show a significant advantage of LG over all other classifiers. mClass+LG achieves (i) a 9.6% testing classification improvement over the best configuration of DeepGene (ii) a 24.6% improvement over the linear kernel SVM, (iii) a 29.6% improvement over the RBF kernel SVM, (iv) a 106.9% improvement over KNN with Euclidean distance, (v) a 64.6% improvement over the KNN with Pearson correlation coefficient, (vi) a 83.6% improvement over Naive Bayes and (vii) a 30.3% improvement over Random Forest.

2.3.5 Experimental results on the four-type dataset

As mentioned above, we used a second dataset consisting four type of cancers, namely COAD, SKCM, LAML and KIRC. After removing genes with less than 1% mutations across all samples, the dimension of the dataset was reduced to 1043×25286. The dataset contains 154 samples for COAD, 345 samples for SKCM, 158 samples for LAML and 386 samples for KIRC. Total number of mutations in this dataset is 363,285. We used the same parameter values for e, t, v and d as in the previous experiment. The 10-fold cross-validation peaked with an accuracy score of 89.5% with 1,132 genes. For testing accuracy, the dataset was divided into training and testing dataset of size 698 (67%) and 345 (33%), respectively. Using 1,132 features, mClass+LG achieves an accuracy of 87.5% on this dataset. Table 2.6 shows the average precision and f1-score for each class in this dataset. Figure 2.5 shows the normalized confusion matrix for our classifier. We could not compare the performance of mClass with DeepGene on this second dataset because, according to the authors, the data pre-processing code necessary to feed the training model for DNN is not available anymore.

Cancer Type	*Precision*	*Recall*	*F-score*	*Support*
COAD	0.93	0.75	0.83	51
SKCM	0.97	0.88	0.92	101
LAML	0.65	0.98	0.79	56
KIRC	0.94	0.88	0.91	137
Avg/Total	0.90	0.87	0.88	345

Table 2.6: Testing accuracies on the four-type cancer dataset using mClass

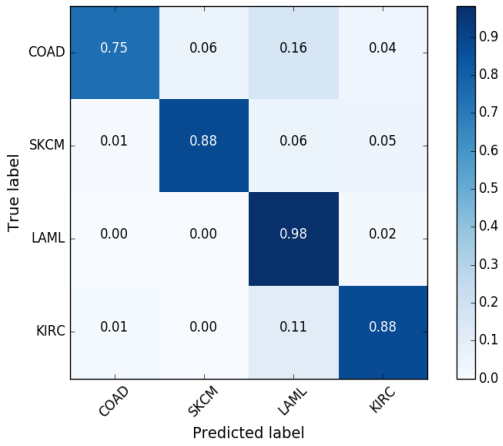

Figure 2.5: Normalized confusion matrix for the four-type cancer dataset

2.3.6 Comparisons of predicted genes

We compared the genes selected by mClass+LG using the 12-types dataset with genes from Cancer Gene Census (CGC). At the time of writing the CGC database contains 719 genes. About 90% of these genes contain somatic mutations, 20% contain germline mutation and 10% contain both types of mutations. We compared mClass' selected genes against the selection carried out by Mutsig 2.0, Mutsig CV [80], MutationAccessor [121] and Muffin [25]. These latter methods predicts cancer genes by analyzing cancer somatic mutation data from 18 types of cancer. We examined the top 100, 500 and 1000 genes produced by these methods, and counted how many of these genes were annotated in the CGC database.

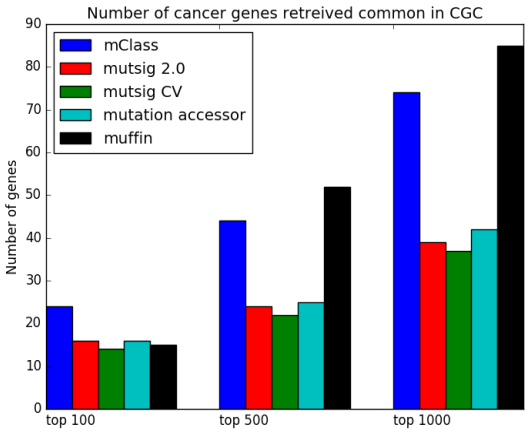

Figure 2.6: Number of CGC genes produced by mClass, Mutsig 2.0, Mutsig CV, MutationAccessor and Muffin in their top 100, 500 and 1000 selection

Figure 2.6 shows these counts for mClass, Mutsig 2.0, Mutsig CV, MutationAccessor and Muffin. Observe that for the the top 100 genes, mClass identifies about 50% more CGC genes than MutSig 2.0, MutSig CV and MutationAccessor. mClass identifies more CGC genes than Mutsig 2.0, Mutsig CV and MutationAccessor for the 500 and 1000 case. However, mClass falls short by 18% and 14% than Muffin in identifying CGC genes in top 500 and top 1000 genes. Although the purpose of mClass was not identifying driver genes, it is remarkable that the top ranked genes selected by mClass contains a large proportion of cancer driver genes.

Chapter 3

DeeplyEssential: A Deep Neural Network for Predicting Essential Genes in Microbes

3.1 Introduction

Essential genes are those genes that are critical for the survival and reproduction of an organism [69]. Since the disruption of essential genes induces the death of an organism, the identification of essential genes can provide targets for new antimicrobial/antibiotic drugs [61, 28]. The set of essential genes is also critical for the creation of artificial self-sustainable living cells with a minimal genome [68]. Essential genes have also been a cornerstone in understanding the origin and evolution of organisms [73].

The identification of essential genes via wet-lab experiments is labor intensive, expensive and time-consuming. Such experimental procedures include single gene knock-out [20, 52], RNA interference, and transposon mutagenesis [123, 33]. Moreover, these experimental approaches can produce contradicting results in [90]. With the recent advances in high-throughput sequencing technology, computational methods for predicting essential genes has become a reality. Some of the early prediction methods used comparative approaches by homology mapping, see, e.g., [102, 148]. With the introduction of large gene database such as DEG, CEG, and OGEE [95, 143, 21], researchers designed more complex prediction models using a wider set of features. These features can be broadly categorized into (i) sequence features, i.e., codon frequency, GC content, gene length [104, 128, 145], (ii) topological features, i.e., degree centrality, cluster coefficient [112, 1, 93, 23], and (iii) functional features, i.e., homology, gene expression cellular localization, functional domain and molecular properties [141, 22, 38, 107, 90]. More recent studies about 3D structure of protein modeled as network can also be incorporated in topological features set [97, 44].

Sequence-based features can be directly obtained from the primary DNA sequence of a gene and its corresponding protein sequence. Functional features such as network topology require knowledge of protein-protein interaction network, e.g., STRING and HumanNET [132, 63]. Gene expression and functional domain information can be obtained from databases like PROSITE and PFAM [62, 46]. Some of the less studied bacterial species, however, lack these functional and topological features, which prevents the use of classifiers that rely on them. Sequence-based classifiers are the most practical methods because they use a minimal amount of features.

Several studies have been published on the problem of predicting essential genes from their sequence. In [128], the authors developed a tool called ZUPLS that uses (i) a Z-curve derived from the sequence, (ii) homology mapping and (iii) domain enrichment score as features to predict essential genes in twelve prokaryotes after training the model on two bacteria. Although ZUPLS worked well on cross-organism prediction, the limited number of bacterial species used as training dataset cast doubts on the ability of ZUPLS to generalize to more diverse bacterial species. In [87], the authors proposed a computational method that employs PCA on features derived from the gene sequence, protein domains, homologous and topological information. Among the studies that predict essential genes across multiple bacterial species, [107] employed several genomic, physio-chemical and sub-cellular localization features to predict gene essentiality across fourteen bacterial species. In their work, the authors dealt with the redundancy in the dataset (i.e., homologous genes shared by multiple bacterial genomes) by clustering genes based on their sequence similarity. In [104], nucleotide, di-nucleotide, codon, and amino acid frequencies and codon usage analysis were used for predicting essentiality in sixteen bacterial species. The authors used CD-HIT [84] for homology detection in both essential and non-essential genes. In [103], the authors identified essential genes in fifteen bacterial species using information theoretical features, e.g., Kullback-Leibler divergence between the distribution of k-mers ($k = 1, 2, 3$), conditional mutual information and entropy features. Although their work showed promising results for intra-organism and cross-organism predictions, the model performed rather poorly when trained on the complete bacterial dataset. Recently, [90] showed the most extensive prediction analysis of thirty-one bacterial species. The authors employed the fea-

tures proposed in [107], with additional features such as transmembrane helices and Hurst exponent. Their algorithm used a regularized feature selection method called least absolute shrinkage and selection operator (Lasso) and used SVM as the classifier.

The latest work in gene essentiality prediction [5] uses network-based features and Lasso for feature selection with Random Forest as the classifier. The authors used a recursive feature extraction technique to compute 267 features in three different categories i.e. *local features* such as degree, *egonet features* which refers to the node and the induced subgraph formed by a node and all of its neighbors and *regional features* which is a combination of local and egonet features. They also used fourteen network centrality measures as a separate feature set for the essentiality prediction. Finally, they combined their network-based features with the sequence based features in [90] and [128] for their prediction model. For the models in [90], [5] and [128], the authors down-sampled non-essential genes to balance the training set but did not realize that their dataset contained multiple copies of homologous genes which created a "data leak" issue which biased their results (see below).

In this work we propose a feedforward deep neural network (DNN) called DeeplyEssential that uses features derived solely from the primary gene sequence to identify essential genes in bacterial species, thus maximizing its practical application compared to other predictors that require structural or topological features which might not be readily available. To the best of our knowledge, this is the first time a deep neural network has been used for gene essentiality prediction.

3.2 Materials and Methods

Genetic data for thirty bacterial species were obtained from the database DEG, which is a curated and comprehensive repository of experimentally-determined bacterial and archaeal essential genes. Among the thirty bacterial species, nine are *Gram-positive* (GP) and twenty-one are *Gram-negative* (GN). DEG provides the primary DNA sequence and corresponding protein sequence for both essential and non-essential genes, as well as gene functional annotations. We only considered protein-coding genes, i.e., we excluded RNA genes, pseudogenes, and other non-coding genes. At the time of writing, DEG contained 28,876 essential protein-coding genes (of which 8,746 belonged to a GP species and 20,130 belonged to a GN species) and 209,026 non-essential protein-coding genes (of which 45,002 were GP and 164,024 were GN). Table 3.1 shows the basic statistics of the dataset. Observe that the dataset is highly unbalanced: while species NC_000907 and NC_002771 have approximately the same number of essential and non-essential genes and bacteria NC_000908 has more essential genes than non-essential genes, for ten bacterial species less than 10% of their genes are essential. In order to improve the performance of our classifier, we balanced the dataset by downsampling non-essential genes.

3.2.1 Feature selection

As said, various intrinsic gene features, such as protein domains, protein-interaction network data, etc. have been used for predicting gene essentiality [103, 87]. DeeplyEssential utilizes codon frequency, maximum relative synonymous codon usage (RCSU), codon adaptation index (CAI), gene length and GC content. Along with these DNA-derived fea-

Table 3.1: The thirty bacterial species used for our experiments (GP is Gram-positive, GN is Gram-negative)

Accession	GP/GN	# Essential genes	# Non-essential genes
NC_000907	GN	1284	1024
NC_000908	GP	762	188
NC_000913	GN	1810	14000
NC_000915	GN	646	2270
NC_000962	GP	4144	17586
NC_000964	GP	542	7808
NC_002163	GN	788	5602
NC_002505/002506	GN	1558	5886
NC_002516	GN	906	21266
NC_002745	GP	604	4562
NC_002771	GP	620	644
NC_003197	GN	460	8456
NC_004347	GN	804	2206
NC_004631	GN	1422	15822
NC_004663	GN	650	8906
NC_005966	GN	998	5188
NC_006351/006350	GN	1010	10444
NC_007297	GP	454	2674
NC_007795	GP	702	5082
NC_008463	GN	670	1920
NC_008601	GN	784	2658
NC_009009	GP	436	4104
NC_009511	GN	1070	8630
NC_010729	GN	1488	6870
NC_011375	GP	482	2354
NC_011916	GN	960	6448
NC_016776	GN	1094	7486
NC_016810	GN	706	8070
NC_016856	GN	210	10420
NC_007650/007651	GN	812	10452

tures, DeeplyEssential also uses amino acid frequency and sequence length from the protein sequences.

Codon frequency

Codon frequency has been recognized as an important feature for gene essentiality prediction [90, 107]. Given the primary DNA sequence of a gene, its codon frequency is computed by sliding a window of three nucleotides along the gene. The raw count of $4^3 = 64$ codons is then normalized by the total number of genes. Observe in Figure 3.1 that the codon frequency can be quite different in the two classes. For instance, codon AAA, GAA, TGA, GAT, AAG, ATT and AGA had at least 30% difference in their normalized codon frequency between essential and non-essential genes.

Gene length and GC content

Other distinguishing features for gene essentiality are gene length and GC content. Figure 3.2 shows the distribution of gene length in GP, GN and complete dataset (GP+GN). Observe that in the complete dataset and the GN dataset, genes have a similar average length in the two classes, while in the GP dataset essential genes are on average longer than non-essential genes. As said, the GC content is another informative feature of essentiality prediction. Figure 3.3 shows the difference in distribution in GC content between two classes. Observe that non-essential genes have higher GC content than essential genes.

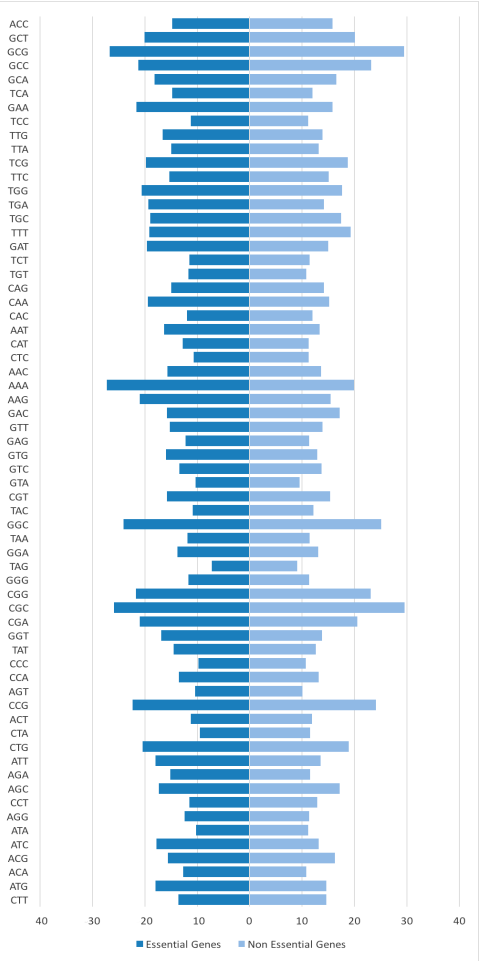

Figure 3.1: Normalized codon frequency of gene sequences in GP + GN dataset

Relative synonymous codon usage

Unbalanced synonymous codon usage is prevalent both in prokaryotes and eukaryotes [101]. The degree of bias varies among genes not only in different species but also among genes in the same species. Differences in codon usage in one gene compared to its surrounding genes may imply its foreign origin, different functional constraints or a different regional mutation. As a result, examining codon usage helps to detect changes

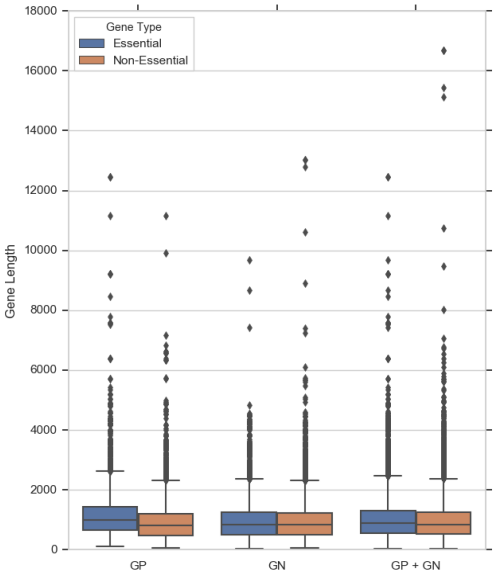

Figure 3.2: Distribution of gene lengths in datasets GP+GN, GN, GN

in evolutionary forces between genomes. Essential genes are critical for the survival of an organism thus codon usage acts as a strong distinguishing feature. To calculate the relative synonymous codon usage we compare the observed number of occurrence of each codon to the expected number of occurrences (assuming that all synonymous codons have equal probability). Given a synonymous codon i that has an n-fold degenerate amino acid, we compute the *relative synonymous codon usage* (RCSU) as follows

$$\text{RCSU}_i = \frac{X_i}{(1/n)\sum_{i=1}^{n} X_i}$$

where X_i is the number of occurrence of codon i, and n is 1, 2, 3, 4, or 6 (according to the genetic code).

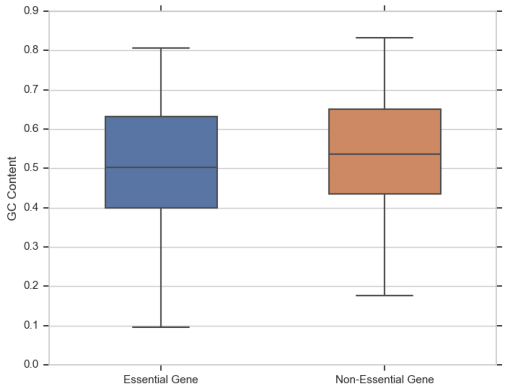

Figure 3.3: GC content distribution in essential and non-essential gene sets in GP + GN dataset

Codon adaptation index

The *codon adaptation index* (CAI) estimates the bias towards certain codons that are more common in highly expressed genes [101]. The CAI is defined by the geometric mean of the relative adaptedness statistics. The *relative adaptedness* for codon i is defined on the relative frequency of the codon in a species-specific reference set of highly expressed genes. Formally, the relative adaptedness is defined by

$$r_i = \frac{\text{RCSU}_i}{\text{RCSU}_{max}} = \frac{X_i}{X_{max}}$$

where RCSU_{max} and X_{max} are corresponding RCSU and X value of the most frequently used codon. The CAI for a gene is defined by

$$\text{CAI} = \left(\prod_{i=1}^{L} r_i \right)^{\frac{1}{L}}$$

where L is the number of codons in the gene excluding methionine, tryptophan, and stop codon. The value of CAI ranges from zero to one, where zero indicates no bias and higher values indicating a higher proportion of the most abundant codons.

Protein sequence features

Another informative set of features used for the prediction of gene essentiality are those derived from the corresponding protein sequences. Previous studies have used frequency of rare amino acids, and the number of codons that are one-third base mutations removed from the stop codons [90]. DeeplyEssential only uses amino acids frequencies and the lengths of the protein sequences.

Combining all the features

Given the primary DNA sequence of a gene, we generate $4^3 = 64$ values for the codon frequency, and one value for the GC content, gene length, CAI and RCSU_{max}. From the protein sequence, we compute the amino acid frequency vector (20 components), and one value for the protein length. The total number of features used by DeeplyEssential is 89.

3.2.2 Multi-layer perceptron

Multi-layer perceptron (MLP) consists of multiple layers of computational units where the information flows in the forward direction, from input nodes through hidden nodes to the output nodes without any cycles [124]. MLP networks have been used successfully for several molecular biology problems, see, e.g. [85, 126, 47]. The architecture of DeeplyEssential is composed of an input layer, multiple hidden layers, and an output layer. The output layer encodes the probability of a gene to be essential. The addition of dropout layer makes the network less sensitive to noise in the training and increase its ability to generalize by randomly assigns zero weights to a fraction of the neurons [129].

Let $\overrightarrow{x} = (x_1, \cdots, x_n)^T$ be the input to the MLP. Let vector y denotes the output of the i^{th} hidden layer. The output y^i depends on the input in the previous layer as follows

$$y^i = a(W^i x^{(i-1)} + b^{(i-1)})$$

where a is the activation function, b is the bias and W is the weight matrix for each edge in the network. During training, the network learns the weights W and the bias b. DeeplyEssential uses a rectified linear unit (ReLU) in each neuron in the hidden layers. ReLU is an element-wise operation that clamps all negative values to zero.

In the output layer DeeplyEssential uses a sigmoid as the activation function to perform discrete classification

$$y = \frac{1}{1 + e^{-x}}$$

The loss function is binary cross-entropy defined by

$$\sum_{c=1}^{M} \hat{y}_{o,c} \log(p_{o,c})$$

where M is the number of classes (two in our case), $\hat{y}$ is the binary indicator if class label c is the correct classification for observation o, and p is the predicted probability observation o is of class c. Figure 3.4 illustrates the architecture of the neural network used in DeeplyEssential.

3.3 Results and Discussion

3.3.1 Classifier design and evaluation

As mentioned in Section 2.1, the number of non-essential genes is significantly larger than the number of essential genes. To address this imbalance in the training set

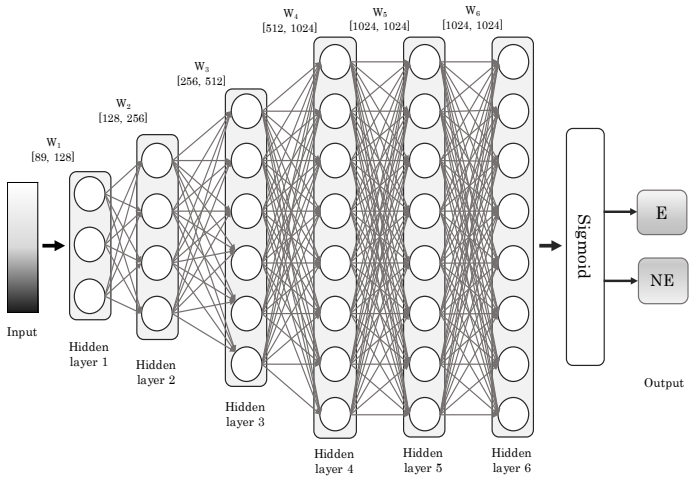

Figure 3.4: The architecture of the neural network used in DeeplyEssential

and allow for unbiased learning, we randomly down-sample non-essential genes. In [145], the authors showed that balancing the dataset did not negatively influence the prediction of gene essentiality.

Model hyper-parameters

Recall that each gene (and its corresponding protein) is represented by 89 features in the input layer. The deep learning architecture of DeeplyEssential was determined by running extensive experiments on the training data over a wide range of hyper-parameters. The number of hidden layers, the number of nodes in each of the hidden layers, the batch size, the dropout rate and the type of optimizer were selected by optimizing the performance of the classifier. Table 3.2 lists the range of hyper-parameter considered and the values of the hyper-parameter selected for the final architecture of DeeplyEssential.

Table 3.2: Hyperparameters for DeeplyEssential

Parameters	Range	Selected Parameter
# hidden layers	[2 - 8]	6
# nodes	[32, 64, 128, 512, 1024, 2048]	128, 256, 512, 1024, 1024, 1024
dropout rate	[0.1 - 0.5]	0.3
epochs	–	100 (early stopping)
optimizer	sgd, adam, adadelta, RMSProp	adadelta

Observe in Figure 3.4 that the final fully-connected layer reduces the 1024 dimensional vector to a two-dimensional vector corresponding to the two prediction classes (essential/non-essential). The sigmoid activation function forces the output of the two neurons in the output layer to sum to one. Thus their output value represents the probability of each class. Among the available optimizer in Table 3.2, we chose adadelta because of its superior performance. Adadelta is parameter-free, thus we do not need to define the learning rate. The training was run for 100 epochs with early stopping criteria.

We trained DeeplyEssential on three datasets, namely GP, GN, and GP+GN (see Section 2.1 and Section 3.2). For each dataset, 80% data is used for training, 10% data for validation and 10% data for testing. The random selection was repeated ten times, i.e., ten-fold cross-validation was performed to complete the inference.

Evaluation metrics

The tools described in [90], [107], [104] and [103] are currently unavailable. We ran DeeplyEssential on the datasets used in the corresponding papers, and compared DeeplyEssential's classification metrics to the published metrics.

We evaluated the performance of DeeplyEssential using the Area Under the Curve (AUC) of the Receiver Operating characteristic Curve (ROC). ROC plot represents

Table 3.3: Basic statistics for GP, GN, and GP+GN (balanced and unbalanced)

Dataset	# Training Samples	# Validation Samples	# Test Samples
GP	7,065	883	884
GN	14,364	1,795	1,797
GP+GN (bal)	21,432	2,678	2,680
GP+GN (unbal)	90,571	11,321	11,322

the trade-off between sensitivity and specificity for all possible thresholds. Although our primary evaluation measure is the AUC score, we report the following additional performance measures

$$\begin{aligned} \text{Sensitivity(Sn)} &= \frac{TP}{(TP+FN)} \\ \text{Specificity(Sp)} &= \frac{TN}{(FP+TN)} \\ \text{PPV} &= \frac{TP}{(TP+FP)} \\ \text{Accuracy} &= \frac{(TP+TN)}{(TP+FN+TN+FP)} \end{aligned}$$

where TP, TN, FP and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively.

All experiments were carried out a Titan GTX 1080 Ti GPU, running Keras 2.1.5.

3.3.2 Gene essentiality prediction

We collected essential and non-essential gene for thirty bacterial species as described in Section 2.1 into three datasets, namely GP, GN, and GP+GN. After re-balancing the dataset by down-sampling non-essential genes, we extracted the features for each gene as explained in Section 2.2. Table 3.3 shows the basic statistics for each dataset.

Table 3.4: Training classification performance of DeeplyEssential on GP, GN, GP+GN

Metric	GP	GN	GP+GN
AUC	0.838	0.823	0.842
Sensitivity	0.741	0.784	0.801
Specificity	0.758	0.708	0.721
PPV	0.774	0.722	0.749
Accuracy	0.749	0.745	0.762

Table 3.4 shows the training classification performance of DeeplyEssential, averaged over ten repetitions. The violin plot in Figure 3.5 shows the distribution of AUCs across the ten repetitions of the experiment, which appears very stable. The receiver operator curves (ROC) are shown in Figure 3.7. DeeplyEssential yielded an area under the curve of 0.838, 0.829 and 0.842 for GP, GN, and GP+GN on average, respectively. The ROC curve also indicates the relation between the number of training samples and stability in model performance. Observe that DeeplyEssential's performance was more stable on the GP+GN dataset than the GP dataset (which contains the smallest number of samples). Also the precision-recall curves shows the ability of our model to yield low false positive rate and low false negative rate across all dataset consistently.

3.3.3 Comparison with down-sampling methods

As said in Section 3.2, the gene essentiality dataset is highly unbalanced. It is well-known that class imbalance can negatively affect the performance of a classifier [144]. To quantify how class imbalance affects the performance of our classifier we trained DeeplyEssential on the full (unbalanced) dataset that has 322.6% more non-essential genes than essential genes. Figure 3.6 shows that the sensitivity and Positive Predictive

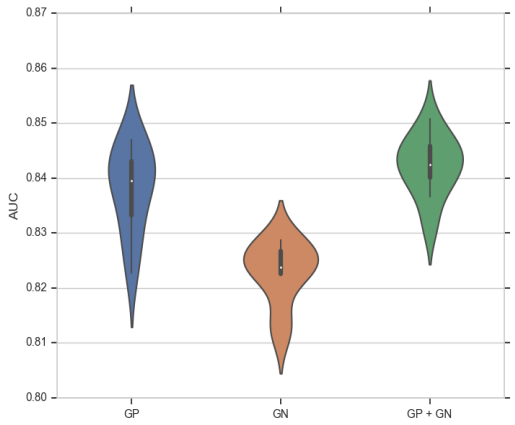

Figure 3.5: Violin plot of DeeplyEssential's AUC across ten experiments on the GP, GN, GP+GN datasets

Table 3.5: Comparing the performance of DeeplyEssential on down-sampled dataset against methods that solely use sequence features; numbers in boldface indicate the best performance

Method	# features	AUC	Sensitivity	PPV
Liu *et al.* 2017	40	0.794	0.715	0.243
Azhagesan *et al.* 2018	267	0.838	0.754	0.321
ZUPLS	274	0.705	0.663	0.255
DeeplyEssential	89	**0.842**	**0.801**	**0.749**

Value (PPV) of the classifier trained on unbalanced data are much worse than the balanced dataset. As said, some of the existing methods use down-sampling to address this problem. Both Liu *et al.* 2017 [90] and Azhagesan *et al.* 2018 [5] randomly down-sampled the majority class data to match the size of the minority class. DeeplyEssential also uses this approach. Table 3.5 shows the performance DeeplyEssential compared to the two published methods that use down-sampling. Observe that DeeplyEssential achieves the best AUC, sensitivity, and PPV.

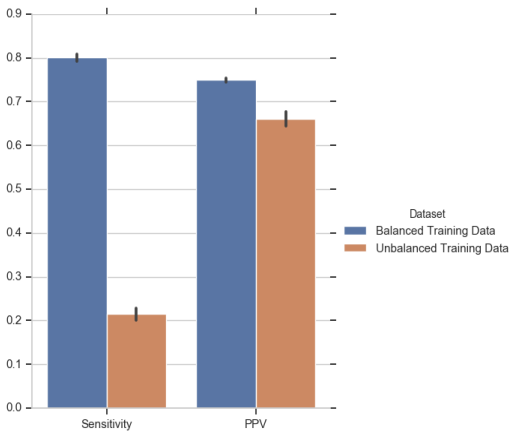

Figure 3.6: Comparing the prediction performance of DeeplyEssential when trained on balanced or unbalanced GP+GN dataset

3.3.4 Identification of "data leak" in the gene essentiality prediction

Bacteria are unicellular organisms with a relatively small set of genes. Across bacterial species, a significant fraction of genes is conserved because they perform similar fundamental biological functions. These conserved genes are quite similar at the sequence level. All published methods rely on a dataset containing multiple bacteria on which genes have been labeled essential or non-essential. Let x and y be two homologous genes, i.e., x and y have a very similar sequence. If x is used on the training and y if used for testing, this introduces a bias, or a "data leak". Training examples and testing examples are supposed to be distinct, and in this hypothetical scenario, they are not.

To quantify the effect of the data leak issue, we clustered the set of all genes across the thirty bacterial species using OrthoMCL [83]. OrthoMCL is a popular method for clustering orthologous, homologous and paralog proteins which use reciprocal best hit alignment to detect potential in-paralog/recent paralog pair, and reciprocal alignments best

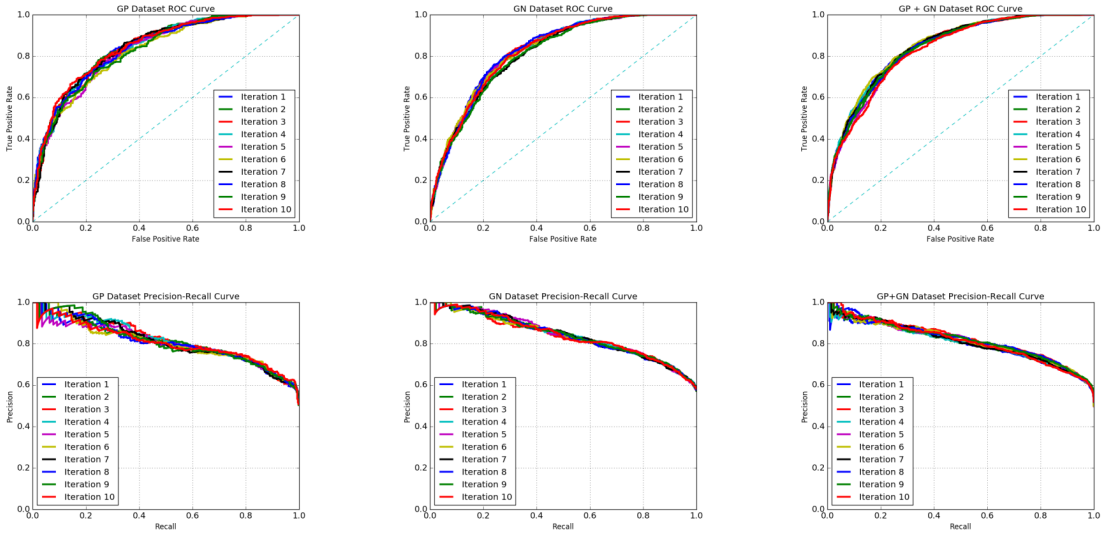

Figure 3.7: DeeplyEssential's ROC and AUPR curves on GP, GN, GP+GN

hits across any two genomes to identify potential ortholog pairs. A similarity graph is then generated based on the proteins that are interlinked. To split large clusters, a Markov Clustering algorithm (MCL) is then invoked [136]. Inside MCL clusters, weights between each pair of proteins are normalized to correct for evolutionary differences.

As said, OrthoMCL produces a list of clusters where each cluster consists of genes that have been determined to be orthologous. To quantify the effect of gene sequence similarity on the prediction performance, we created a dataset where no gene from a single cluster can end up in both the training set and the testing set. The modified dataset contains 11,168 training samples, 2,798 validation samples, and 4,270 testing samples. The prediction was repeated ten times. Table 3.6 shows the clustering step heavily influences DeeplyEssential's prediction performance: AUC decreased by more than 7% (on average), while the accuracy decreased by 6.9% (along with a significant decrease in all performance measures).

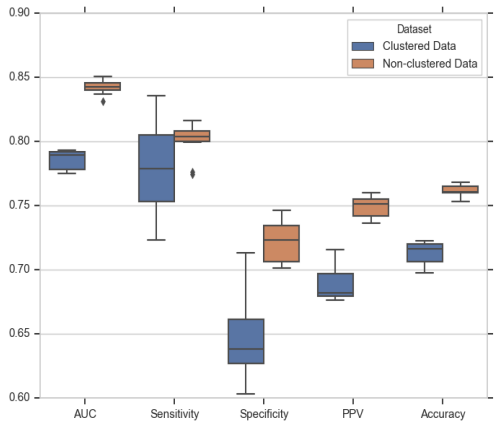

Figure 3.8: Effect of "data leak" on DeeplyEssential's prediction performance

Table 3.6: Comparing the effect of clustering on the prediction performance of DeeplyEssential on the GP+GN dataset

Metric	Non-clustered	Clustered	Difference (%)
AUC	0.842	0.786	7.12%
Sensitivity	0.801	0.780	2.69%
Specificity	0.721	0.646	11.60%
PPV	0.749	0.688	8.86%
Accuracy	0.762	0.713	6.87%

Figure 3.8 shows the difference in performance before and after clustering. While the AUCs were stable across experiments, sensitivity, specificity, and PPV varied largely across experiments for the clustered dataset.

3.3.5 Comparison with methods that cluster orthologous genes

Some published studies have addressed the data leak issue by identifying homologous genes using sequence similarity metrics. In [103], the authors used the Kullback-Leibler divergence (KLD) to measure the distance between k-mer distribution (for $k = 1, 2, 3$) ob-

Table 3.7: Comparing the performance of DeeplyEssential and Ning *et al* and Nigatu *et al* on their respective datasets [104], [103]; numbers in boldface indicate the best performance

Method	Clustering method	AUC
Ning *et al* 2014	CD-HIT	0.758
DeeplyEssential	OrthoMCL	**0.818**
Nigatu *et al* 2017	*Kullback-Leibler divergence*	0.650
DeeplyEssential	OrthoMCL	**0.840**

tained from sequences. In [104], the authors used CD-HIT to remove redundancy in the training data and improve the generalization ability of their model. As explained in the previous section, DeeplyEssential uses OrthoMCL to cluster homologous genes to prevent similar genes to appear in both training and testing dataset. Table 3.7 shows the performance comparison of DeeplyEssential with [104] and [103] on their respective datasets.

Observe that in both cases DeeplyEssential achieves the best predictive performance. As said, although each of these two approaches uses a distinct method to find orthologous genes, the use of the same dataset for the experiments ensures a fair comparison.

3.3.6 Feature importance

DeeplyEssential uses exclusively sequence-based features and yet produces higher prediction performance. Unlike other machine learning classifiers, the DNN architecture does not readily provide any insight about the feature set that contributed maximally towards the prediction performance. To understand the impact of a feature on the predictive performance, we carried out an ablation study which removes the feature(s) from the input and observes the performance difference to determine the importance of a feature. However, this type of study is not very informative in the presence of highly correlated features. In this case, the absence of a feature can be compensated by another feature which is highly

correlated with the former feature. To address this issue, we first computed the pairwise Pearson correlation among all input features. Figure 3.9 illustrate the heatmap of the pairwise correlation. Each axis shows the indices of the features: indexes 0–65 contains DNA specific feature, index 68–89 contains protein specific features. GC content, CAI and RSCU_{max} have a negative correlation with all other features. There were nineteen pair of features showing a correlation higher than 0.9 (in absolute value). For our ablation study, we either removed one feature at a time (if uncorrelated) or one of the 19 feature pairs to test the performance changes on the GP + GN dataset using 5-fold cross-validation. We measured the difference in AUC and ordered the features based on their impact in decreasing the predictive performance (Figure 4.7). Observe that codon TTT caused the highest AUC decrease (3.5%) while AGA, TTC, CGT, CGA, gene length, protein length, GC content, CAI, amino acids K, L, R, W, Y, C, G, E, F, and pairs of correlated features CCG+CGC, TAA+TTA, gene length+L, D, and protein length+T induced 2.5% –3% AUC decrease. Our finding that gene and protein length are highly informative features for essentiality prediction reconfirm their prominence, as illustrated in other sequence-based methods, e.g., [90] and [128]. Moreover, it is well-known that in essential genes within the functional category related to information storage and process, encoded amino acids K, L and subcategories of encoded amino acids C, G, E, F are preferentially suited at the leading strand where these are responsible for energy production and conversion, carbohydrate transport and other essential metabolic processes [88].

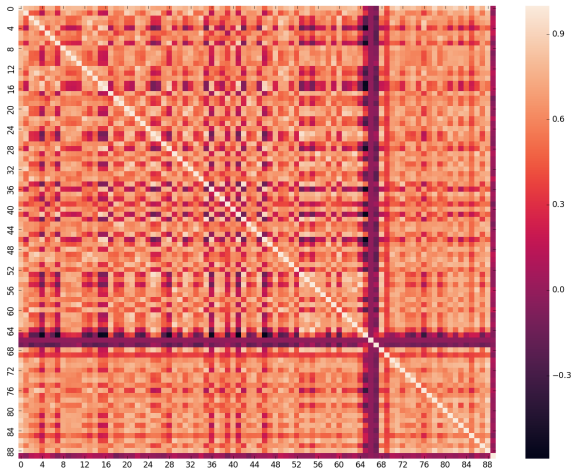

Figure 3.9: Pairwise correlation among all features; features 0–65 are DNA specific feature; features 68–89 are protein specific features

3.3.7 Discussion

A large number of structural and functional features have been used for gene essentiality prediction, i.e. producibility, choke points, load scores, damages, degree of centrality, clustering coefficient, closeness centrality, betweenness centrality, gene expression, phyletic retention, among others. These features cannot be obtained from the gene sequences and are often not available for many bacterial species. To maximize its practical utility, DeeplyEssential uses exclusively features derived directly from the sequence.

Previous works have addressed the high imbalance of the training dataset by either down-sampling non-essential genes or by clustering orthologous genes across species. In order to make a meaningful and fair comparison, we compared DeeplyEssential's performance to both approaches. In fact, our experiments showed that DeeplyEssential has better predictive performance both on down-sampled and clustered datasets. On the down-sampled dataset used in [90], DeeplyEssential demonstrated an improvement of

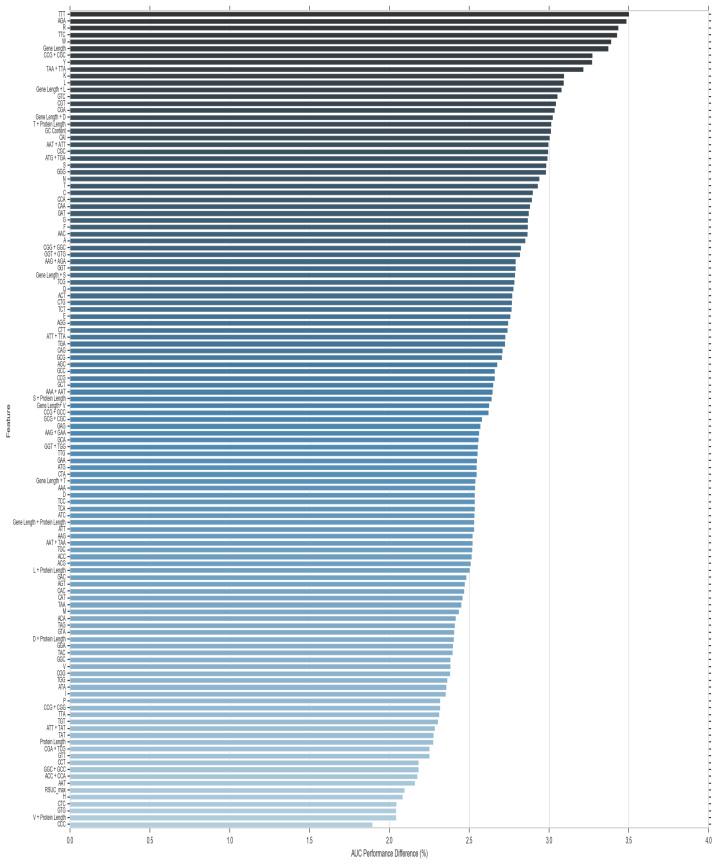

Figure 3.10: Changes in AUC predictive performance due to the removal of a feature or pairs of correlated features

12.8% in AUC compared to [90]. In addition, DeeplyEssential produced significantly better sensitivity and precision than the three methods in Table 3.5, achieving 6.2% improved sensitivity and 137.4% improved precision compare to [5]. If one uses all the 597 features in the prediction model in [5], then this latter method achieves 1.7% improved AUC compared to DeeplyEssential. We believe that collecting this very large amount of features from multiple databases does not warrant the additional (minor) benefit in predictive performance. DeeplyEssential also achieved better performance on clustered datasets. Table 3.7 shows 7.9% and 29.2% improved AUC compared to [104] and [103], respectively.

As an alternative to the proposed approach that uses a carefully selected set of features as input, one could consider training a CNN-based network that uses exclusively one hot encoding of the DNA and protein sequence as input. One hot encoding is a process that converts categorical variables into a form that is convenient for the prediction by machine learning models. We expect that the limited size of the available training data would be insufficient to allow for CNN to extract relevant features. As a consequence, the CNN-based classifier would not be as accurate compared to the architecture proposed here.

3.4 Conclusion

We proposed a deep neural network architecture called DeeplyEssential to predict gene essentiality in microbes. DeeplyEssential makes a minimal assumption about the input data (i.e, it only uses the gene sequence), thus maximizing its practical application compared to other predictors that require structural or topological features which might not be readily available. Extensive experiments show that DeeplyEssential has better predictive performance than existing prediction tools. We believe that DeeplyEssential could be further improved if more annotated bacterial data was available, making it an essential tool for drug discovery and synthetic biology experiments in microbes.

Chapter 4

Epi2En: A Convolutional Neural Network for Genome-wide Enhancer Prediction

4.1 Introduction

Enhancers are distal *cis*-regulatory elements that affect the cellular transcriptome by controlling gene expression [12, 106, 16, 13, 113]. Unlike promoters which serve as anchor points for recruiting protein complexes needed for transcription (e.g., transcription factors), enhancers are positioned 20kb or further away from their target gene(s); sometimes they are located on a different chromosome. Enhancers are known to interact with their target gene(s) through promoter-enhancer looping [125, 57, 66, 37]. In order to ensure developmental robustness, some genes involved in these processes contain multiple enhancers

with overlapping functions and their functional mechanisms appear to be independent of their target gene location [111]. In evolutionary biology, enhancers and other *cis*-regulatory elements that produce phenotypical changes have been investigated to understand developmental differences across species. Genetic variation in non-coding regions, especially enhancers, represents the primary basis in many human genetic diseases which are driven by dysregulation of gene expression [49]. Despite extensive studies on the identification of enhancers, their complex functional properties across cell-lines and tissue-types, their low evolutionary conservation, and their variable motif composition, makes it analytically challenging to systematically and precisely identify them.

Initial efforts in identifying enhancers were based on sequence conservation because of the negative evolutionary selection pressure on regulatory sequences [117, 137, 105, 139, 92]. This approach is however unreliable since studies showed that mammalian regulatory elements show low conservation among different species and consequently they cannot be effectively characterized based on orthologous regions [138]. Recent advancements in high-throughput sequencing technologies have allowed researchers to collect experimental data on proximal and distal *cis*-regulatory elements and their interactions. These identification methods either rely on detected binding sites for transcription factors and other DNA-binding proteins (ChIP-Seq) or the presence of open chromatin (DNase-Seq).

Several supervised and unsupervised machine learning methods had been developed for genome-wide enhancer predictions based on either ChIP-Seq or DNase-Seq data. For instance, ChromHMM utilizes a hidden Markov model and leverages a clustering of genome-wide profiles in multiple cell-lines to assign chromatin states [41]. Segway is an-

other unsupervised method that produces a segmentation of the genome based on dynamic Bayesian networks [59]. Several supervised methods for predicting enhancers have been proposed in the literature. CSI-ANN uses an artificial neural network [48], ChromaGenSVM combines genetic algorithms with support vector machines [45], REFCS uses random forests [119] and DELTA uses adaptive boosting [94]. More recently, the authors of [72] proposed a model with three-component (DEEP-ENCODE, DEEP-FANTOM5, and DEEP-VISTA) for predicting enhancers using (i) histone marks from the ENCODE cell-lines, (ii) tissue-specific sequence characteristics from FANTOM5 experiments and (iii) sequence features derived from developmental enhancers from the VISTA database. DEEP-ENCODE uses eleven histone tail modifications and 351 sequence characteristics as input features. It combines four ensemble models that are trained on four cell-line and contains a total of $4,000$ classifiers to predict enhancers. PEDLA is a deep learning-based algorithm for enhancer prediction that uses 1,114-dimensional features including histone modifications, transcription factors and co-factors, chromatin accessibility, CpG islands, evolutionary conservation, sequence signatures and occupancy of transcription factor binding sites. [89].

Although the use of machine learning has significantly improved the accuracy of enhancer prediction, published methods suffer from a few shortcomings. For instance, some of these methods are trained on a small number of samples which could lead to over-fitting and does not allow the model to generalize. In fact, the lack of generalization is still a significant issue for enhancer prediction tools. Most of these methods are highly dependent on the specific cell-line or tissue, while the number of tissues in mammals reaches in the hundreds. In addition to histone marks, some methods require the integration of additional

data from various other sources like the DNase I hypersensitivity site (DHS) extracted from DNase-Seq data [10] and/or DHS data combined with transcription factor binding sites motifs extracted from databases like HOCOMOCO [75]. The integration of heterogeneous data types is tedious and time-consuming and often these additional data are not available for all cell-lines of interest.

To address these issues we have developed a convolutional neural network called Epi2En that can predict enhancer regions across multiple cell lines exclusively based on chromatin data (histone marks and CTCF). To the best of our knowledge, Epi2En is the first use of a convolutional neural network for enhancer prediction.

4.2 Materials and Methods

4.2.1 Labeled data

Epi2En was trained on three human cell-lines, namely Gm12878, H1-hesc, and Hep-g2. Two additional cell-line data, namely K562 and Hela-S3, were used exclusively for testing. Enhancer regions were generated as it was done in DEEP-ENCODE and PEDLA. The absence of a genome-wide set of experimentally-verified enhancers across different cell-lines justified this choice.

Briefly, PEDLA considers a region to be an enhancer based on the presence of specific histone marks across cell-lines. PEDLA non-enhancers set (negative examples) contains 10% promoters and 90% random genomic regions that are not annotated as either enhancers or promoters.

DEEP-ENCODE instead considers a region to be an enhancer based on the binding of proteins like CTCF and Pol2, as well as DNase data and FAIR arrays. An unsupervised clustering algorithm is used to label enhancers according to their functional annotations described by 25 states. The final set of enhancers is then selected based on the regions with the highest confidence. Since DEEP-ENCODE does not provide non-enhancers regions, we used random genomic regions not labeled as enhancers. We generated these negative examples based on the distribution of the lengths of enhancer and non-enhancer regions across all cell-line data. (see, e.g., Figure 4.2 for Gm12878)

In the rest of the manuscript, we will refer to the DEEP-ENCODE enhancers as DE and will refer to the PEDLA enhancers as PE. Table 4.1 summarizes the basic statistics of the DE dataset and PE dataset. Observe that DE contains a much higher number of enhancers. Irrespective on the dataset we always used a 1:10 positive/negative ratio to train Epi2En, unless otherwise specified. Hela-S3 and K562 cell-line data were collected from PEDLA and used for independent testing of Epi2En. To understand the potential overlap between these two datasets, we declared an enhancer region in DE to match an enhancer region in PE if the corresponding intervals overlapped by at least 1bp. The Venn diagram in Figure 4.1 shows the number of common matching intervals.

4.2.2 Features

Our feature data includes (1) architectural protein CTCF, (2) enhancer-specific histone marks H3k4me1 and H3k27ac, (3) activating histone marks H3k9ac, H3k4me2 and

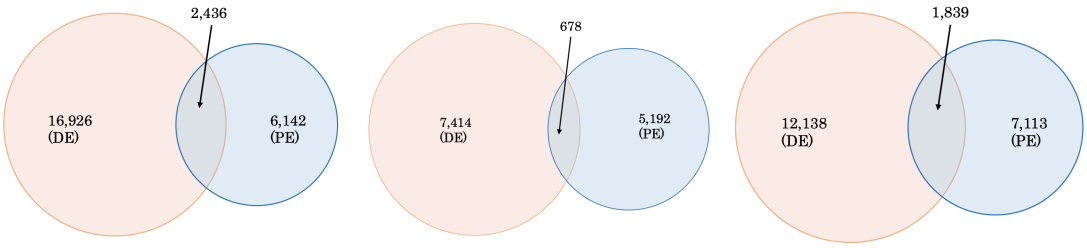

Figure 4.1: Venn diagram for the enhancer regions in the PE dataset (blue) and DE dataset (pink). Gm12878 cell-line (left), H1-hesc cell-line (middle) and Hep-g2 cell-line (right)

Table 4.1: Training/Testing data statistics

Dataset	Source	Cell-line	# enhancers	# non-enhancers
DE	DEEP-ENCODE	Gm12878	19,362	185,244
		H1-hesc	8,092	711,680
		Hep-g2	13,977	369,259
PE	PEDLA	Gm12878	8,578	85,789
		H1-hesc	5,870	320,131
		Hep-g2	8,952	89,529
		Hela-S3	6,692	66,929
		K562	13,423	134,239

H3k4me3, (4) repressive histone marks H3k27me3 and H3k9me3, (5) active gene body and elongation histone marks H3k36me3, H4k20me1, and H3k79me2 and (6) histone variant H2A.Z in nucleosomes. According to [16] and [109] histones at enhancers regions are decorated with specific marks such as H3k4me1 and H3k27ac but not limited to only these modifications. H3k4me3 has been shown as a prominent feature of active promoter but also detectable at active enhancers bound by RNA Pol II. H3k4me2 is present in both enhancers and promoters. H3k9ac is an acetylation mark that also has been detected at enhancers. Other histone modifications, i.e. H3k9me3 and H3k79me3, have been detected at putative enhancers and poised enhancers. H3K36me3, H4K20me1, and H3K79me2 are as-

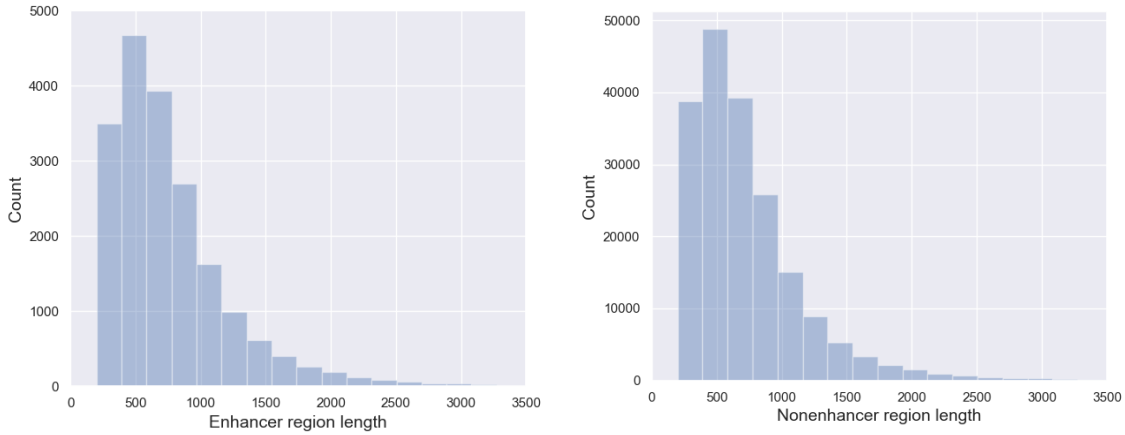

Figure 4.2: Distribution of enhancer/non-enhancer length in Gm12878 cell-line (DE dataset). The length of a few enhancer/non-enhancer reaches about 10kbp (not shown here)

sociated with actively transcribed regions, active and accessible regions and transcriptional transition regions, respectively. Finally, nucleosomes have varying degrees of stability (see, e.g., [115]). Variant H2A.Z increases the thermal mobility of nucleosomes on the DNA template and facilitates transcription factors binding events. Their presence of H2A.Z in active/poised enhancer and promoter regions of multiple cell-lines is supported by several studies [120, 6, 32, 60, 65, 67, 74]. Feature data were obtained from the ENCODE repository [36]. ChIP-Seq data for histone marks are represented as the average read count after mapping the reads to the reference genome.

As shown in Figure 4.2, the length of genomic regions associated with positive and negative examples varies substantially. To fit the features on these variable-length regions into a uniform-length vector to be fed into the network, we divided each region into a fixed number of bins b. Each positive and negative example is represented by a $12 \times b$ matrix

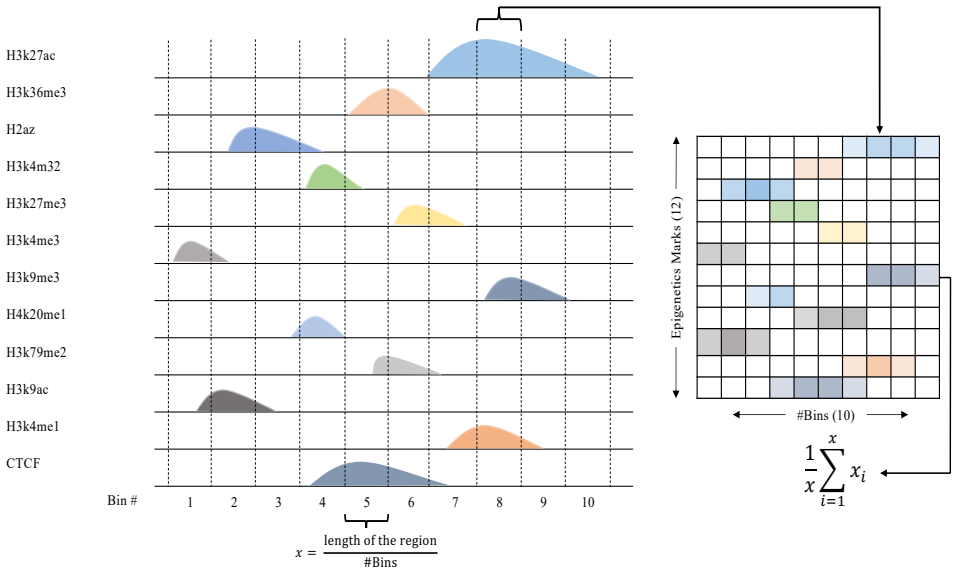

Figure 4.3: Converting epigenetic peaks into the input matrix for Epi2En.

where each row represents an epigenetic marker and each column represents the average signal intensity of each epigenetic marker in that bin. Figure 4.3 illustrates the construction of an input matrix for one particular sample. To choose the optimal value for b, we tested Epi2En with several choices of the number of bins, namely $b = \{5, 10, 15, 20\}$. For each choice of b we computed the difference in Epi2En's performance on the training and testing set. A small difference indicates that that model is stable. According to experiments on $b = [10, 15]$ seems to minimize the difference of both GM (geometric mean of sensitivity and specificity) and precision between the training set and testing set (see Figure 4.4). In all experiments below, we used $b = 10$.

Figure 4.4: The log difference in variance of GM and precision between training and testing for several choices of the number of bin b (x axis)

4.2.3 Convolutional neural network framework

Convolutional neural networks (CNN) were first popularized by [81] and have since extensively used in a wide variety of applications. In this work, we have used a 1-D CNN for the classification of enhancer regions using the Keras framework. Recall that our training set contains N_{samp} samples in paired form $(\mathbf{X}_i, y_i)$, where $\mathbf{X}_i$ are matrices of size $N_f \times b$ and $y_i \in \{0, 1\}$ for $i \in \{1, \cdots, N_{samp}\}$. The architecture of Epi2En is composed of several layers (Figure 4.5) as explained below.

Convolution

In the first layer, we used a 1-D CNN with N_{out} filters each of length k. This CNN performs a sliding window operation across all bin positions, which produces an output feature map of size $N_{out} \times (b - k + 1)$. Each sliding window operation applies N_{out} different

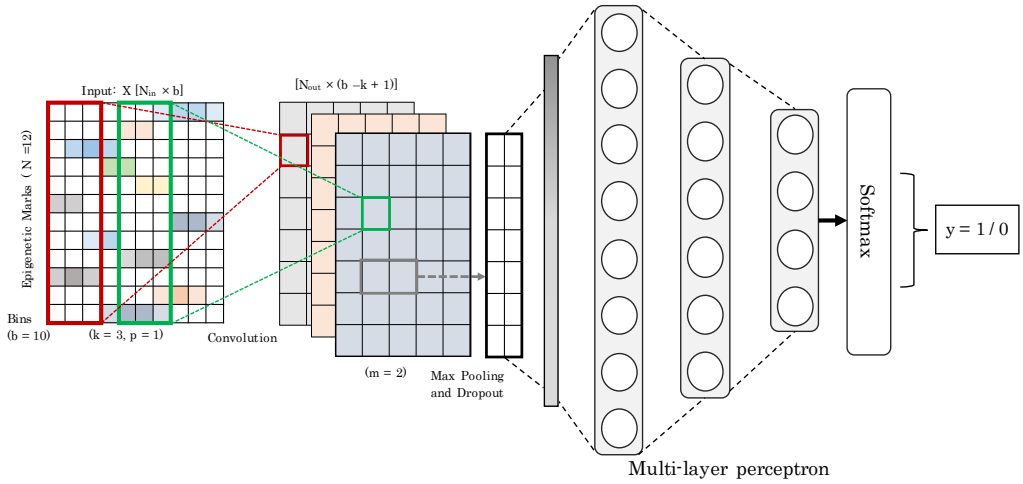

Figure 4.5: The architecture of the neural network in Epi2En.

filters on k consecutive input bins from start position $p_s = 1$ to $(b - k + 1)$. In Figure 4.5 the rectangles illustrates a sliding window with $k = 3$ ($p_s = 1$ for the red rectangle, $p_s = 6$ for the green rectangle). Given an input matrix X of size $N_f \times b$ and a feature map Z, $Z_{p_s,i}$ is generated from the p_s^{th} sliding neighborhood window and the i^{th} hidden filter, where $p_s \in \{1, \cdots, b - k + 1\}$ and $i \in \{1, \cdots, N_{out}\}$, as follows.

$$Z_{p_s,i} = B_i + \sum_{j=1}^{N_f} \sum_{r=1}^{k} W_{i,j,r} X_{j,p_s+r-1}$$

where matrix W (size $N_{out} \times N_f \times k$) and B (size $N_{out} \times 1$) are the trainable parameters of the convolution layer. We finally applied the rectified linear unit function (ReLU), which is an element-wise operation that clamps all negative values to zero.

Pooling and Dropout

We used max-pooling on the convolution layer to learn from the features. Max-pooling selects the maximum values in a certain range which forms a smaller representation

of a larger region for a given sample. Max-pooling was applied on an input Z of size $N_{out} \times a = b - k + 1$. With a pooling size of m, the output V of this layer is of size $N_{out} \times \lfloor \frac{a}{m} \rfloor$. Entry $V_{i,q}$ of this output layer was computed as follows.

$$V_{i,q} = \max_{j=1}^{m} Z_{i,m(q-1)+j}$$

where $q \in \{1, \cdots, \lfloor a/m \rfloor\}$ and $i \in \{1, \cdots, N_{out}\}$.

The grey rectangle in Figure 4.5 shows the results of a max-pooling operation on the feature map where $m = 2$. A dropout layer [130] randomly zeros the input to the next layer during training with a chosen probability of 0.25. The dropout layer regularizes the network and prevents over-fitting.

Feed-forward neural network

The learned regions from the max-pool layers were then fed into a multi-layered perceptron (MLP) to learn the mapping of input features to enhancer/non-enhancer labels. The fully connected layers have alternating linear and non-linear layers. Each layer learns to map its input to a hidden feature space. The output y^i depends on the input from the $(i-1)^{th}$ layer as follows

$$y^i = a(w^i x^{(i-1)} + B^{(i-1)})$$

where a is the activation function, B is the bias and w is the weight matrix for each edge in the network. Epi2En uses softmax as the activation function at the output layer to perform discrete classification, thus

$$y = \left[\frac{\exp(x_1)}{\sum_c \exp(x_c)} \cdots \frac{\exp(x_C)}{\sum_c \exp(x_c)} \right]^T$$

where C encodes the two class types. The output $f(x^i)_c$ of each node c in the output layer represent the probability $f(x^i)_c = p(c|x^i)$ that input sample x^i belongs to class type c. The loss function is binary cross-entropy, defined by

$$\sum_{c=1}^{M} \hat{y}_{o,c} \log(p_{o,c})$$

where $M = 2$ is the number of classes, $\hat{y}_{o,c}$ is the binary indicator (which takes value one if class label c is the correct classification for observation o otherwise), and p is the predicted probability observation that o belongs to class c.

We used stochastic gradient descent (SDG) to train our model via back-propagation [9]. Given a set of training samples, instead of calculating the true gradient of the objective using all training samples, SGD calculates the gradient per sample and updates accordingly on each training sample. To counter the class imbalance problem during training, we assigned empirically decided weights associated with each class (i.e. for the minority class the weight was assigned to 0.65).

4.3 Experimental results

4.3.1 Model design and parameter

As said, Epi2En utilizes a convolutional neural network to predicts enhancers regions using epigenetic data. For each cell-line, positive and negative examples were divided into three sets: 80% of the data is used for training, 10% is used for the validation and 10% is used for testing. Recall that each region is represented by a feature matrix of size 12×10. The architecture of Epi2En was determined by running extensive cross-validation

Table 4.2: Hyper-parameter values for Epi2En

Hyper-parameters	Selected values
#1D convolution layers	3
Convolution layer filter size	128, 64, 64
Convolution layer kernel size	3, 3, 3
Maxpool layer pool size	2
Dropout rate	0.25
# fully connected layers	3
Fully connected layer filter sizes	128, 64, 32
Activation functions	relu, softmax
epochs	100 (early stopping)
Loss function	binary_cross_entropy
Optimizer	SGD

experiments over a wide range of hyper-parameters. The number of convolution layers, the batch size, the dropout rate, the filter and kernel sizes, the number of fully connected layers and the number of nodes in each layer and the type of optimizer were selected by optimizing the performance of the classifier. Table 4.2 shows the parameters that were chosen for the Epi2En.

All experiments were carried out on a Titan GTX 1080 Ti GPU, running Keras 2.1.5. The training was run for 100 epochs with early stopping criteria.

The performance of the model was evaluated by computing (1) $accuracy = (TP + TN)/(TP+TN+FP+FN)$ where TP, TN, FP and FN indicates true positives, true negatives, false positives and false negatives, respectively, (2) $GM = \sqrt{sensitivity \times specificity}$ is the geometric mean of sensitivity and specificity, where $sensitivity = TP/(TP + FN)$ and $specificity = TN/(TN + FP)$ and (3) $F1 = 2/(1/recall + 1/precision)$ is the harmonic mean of recall (sensitivity) and $precision = TP/(TP + FP)$. We used 10-fold cross validation to obtain estimates of these performance indicators.

4.3.2 Performance evaluation

To evaluate the performance of our model, we performed a cross-validation analysis on the cell-line-specific dataset from the two sources described above (Table 4.1). For both DE and PE dataset, we sampled the non-enhancer set to match the 1 to 10 positive to negative ratio.

After the hyper-parameter optimization described above, we performed the ten-fold cross validation on each cell-line. On average Epi2En achieved 99.97% accuracy, 99.87% GM (99.99% sensitivity, 99.74% specificity), 99.86% F1-score and 99.97% precision on the Gm12878 cell-line (PE dataset). On the H1-hesc cell-line (PE data) Epi2En achieved 99.90% accuracy, 99.71% GM and 99.94% precision. On the Hep-g2 cell-line (PE data) Epi2En achieved 99.94% accuracy, 99.81% GM and 99.96% precision. The cross-validation performance of Epi2En was equally high on DE datasets. Epi2En produced >99% accuracy, >99% GM and >99% precision across all DE cell-lines.

Figure 4.6 shows the learning profile from one of the cell-lines for the gradual performance of the model across many epochs on training and validation dataset. The figure shows a similar learning profile between the training and validation set across many epochs which implies that the model is not over-fitting.

To gain insights about the importance of each of the twelve epigenetic features on the prediction capability of Epi2En, we performed an ablation test to evaluate how the absence of each feature would affect the model's performance. Figure 4.7 shows the change in the AUC score due to the absence of each feature. The highest drop in performance is due to the removal of H3k27ac, which is an active enhancer marker known to be associated

Table 4.3: Performance comparison of Epi2En against PEDLA and DEEP-ENCODE on PE and DE dataset (GM is the geometric mean of sensitivity and specificity)

Dataset	Cell-line	Method	Accuracy	GM	F1-score	Precision
PE	Gm12878	PEDLA	95.11%	97.30%	77.44%	63.18%
		Epi2En	99.97%	99.87%	99.86%	99.97%
	H1-hesc	PEDLA	97.65%	96.97%	88.31%	81.69%
		Epi2En	99.90%	99.71%	99.49%	99.94%
	Hep-g2	PEDLA	94.64%	97.00%	76.26%	61.64%
		Epi2En	99.94%	99.81%	99.71%	99.96%
DE	Gm12878	DEEP-ENCODE	94.36%	90.73%	73.35%	64.25%
		Epi2En	99.94%	99.85%	99.70%	99.97%
	H1-hesc	DEEP-ENCODE	92.46%	83.64%	64.17%	56.47%
		Epi2En	99.89%	99.63%	99.33%	99.93%
	Hep-g2	DEEP-ENCODE	94.31%	88.83%	71.98%	64.69%
		Epi2En	99.93%	99.76%	99.65%	99.95%

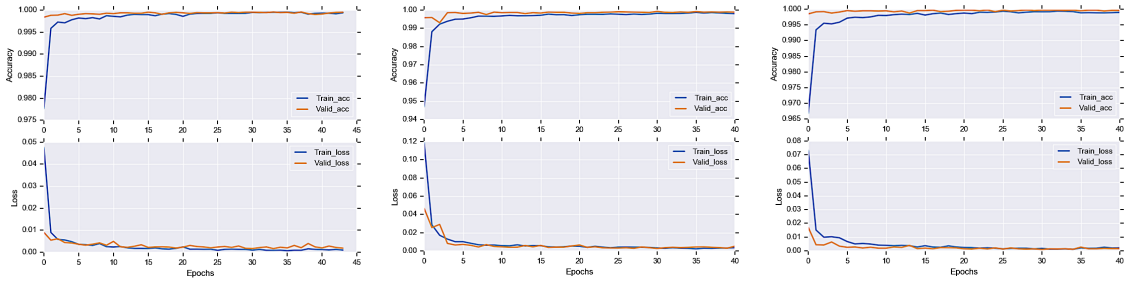

Figure 4.6: Learning profile (accuracy and loss function) of Epi2En during training on the Gm12878 (left), H1-hesc (middle), and Hep-g2 (right) cell-lines (DE dataset)

with active/poised enhancers. Previous study has demonstrated that histone modifications H3k27ac and H3k4me1 are deposited at nucleosomes flanking enhancer elements [16].

4.3.3 Comparison of Epi2En with existing methods

We primarily compared the performance of Epi2En with two state-of-the-art methods, namely PEDLA and DEEP-ENCODE. Five additional methods were tested below. Table 4.3 shows the performance comparison between Epi2En, PEDLA and DEEP-

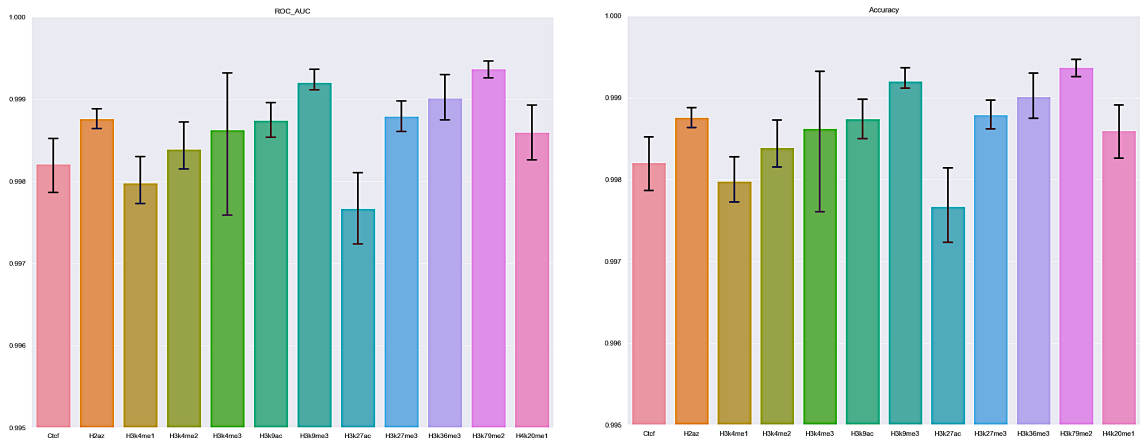

Figure 4.7: ROC and Accuracy ablation analysis on the twelve features used by Epi2En on Gm12878 cell-line data (DE dataset)

ENCODE. Across all measures Epi2En outperformed both PEDLA and DEEP-ENCODE on their respective datasets using optimal settings.

We also evaluated the performance of Epi2En against PEDLA and DEEP-ENCODE on the basis of class-imbalance. In these experiments, we produced a 1:p ratio between positive and negative examples for $p = 1, 2, \ldots, 9$. For each value of p we evaluated the performance of the three methods via cross-validation. These experiments were performed on H1-hesc cell-line. Figure 4.8 shows the performance comparison between Epi2En and PEDLA and DEEP-ENCODE for several choices of p. Observe that Epi2En consistently outperformed Epi2En and PEDLA across all values of p.

We finally compared the cross-validation performance of Epi2En with other supervised and unsupervised methods mentioned earlier. On the H1-hesc cell-line RFECS, CSI-ANN, DELTA, DEEP-ENCODE and PEDLA made 75,691, 30,173, 112,044, 17,960 and 20,686 enhancer predictions, respectively. ChromHMM and Segway made 26,869 and

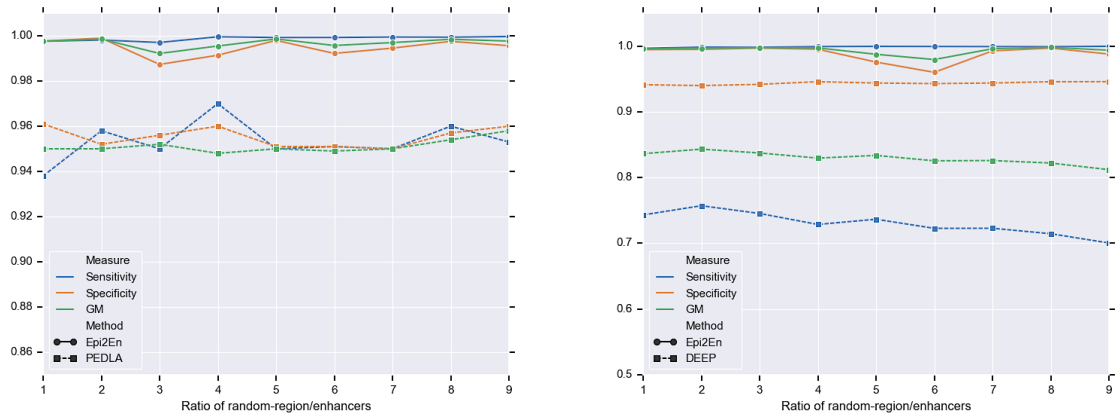

Figure 4.8: Comparing the performance of Epi2En, PEDLA and DEEP-ENCODE for several choices of the ratio of positive vs negative examples on H1-hesc cell-line data

Table 4.4: Performance comparison of Epi2En with other existing method on H1-hesc cell-line dataset

	REFCS	CSI-ANN	DELTA	ChromHMM	Segway	PEDLA	DEEP-ENCODE	Epi2En
Accuracy	93.67%	95.58%	87.78%	94.03%	91.01%	97.65%	92.46%	99.91%
Sensitivity	64.19%	65.50%	73.56%	37.67%	12.89%	96.16%	74.32%	99.95%
Specificity	97.89%	98.63%	89.84%	99.75%	98.94%	97.80%	94.13%	99.94%
GM	79.26%	80.34%	81.29%	61.30%	35.71%	96.97%	83.64%	99.71%
F1-score	71.71%	73.06%	60.40%	53.74%	20.90%	88.31%	64.52%	99.49%

131,689 predictions, respectively. Epi2En trained on ground-truth data from PEDLA made 26,348 enhancer predictions. All methods used the parameter, structure, and thresholds provided in their respective studies. The experimental results for RFECS, CSI-ANN, DELTA, DEEP-ENCODE, and PEDLA were collected from the respective literature. Table 4.4 summarizes the performance of all methods on H1-hesc enhancer predictions.

4.3.4 Experiments across cell-lines

To illustrate the generalization ability of Epi2En, we evaluated the predictive performance of our method using available cell-line data in various ways. First, we trained

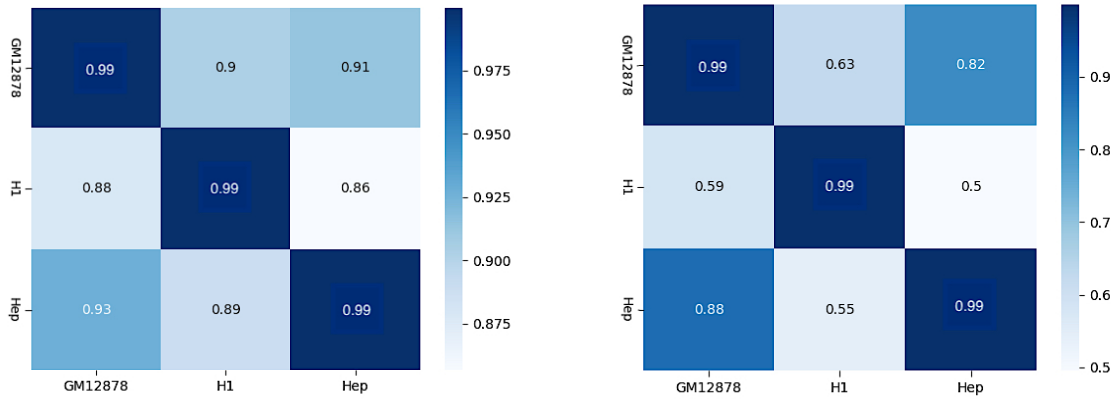

Figure 4.9: Training Epi2En on one cell-line (y-axis) and testing Epi2En on another cell-line (x-axis) on the DE dataset. Accuracy (left), and geometric mean of sensitivity and specificity (right)

Epi2En on the data for one cell-line and tested it on another cell-line. Figure 4.9 shows the accuracy and the geometric mean (GM) of sensitivity and specificity for all choices of train/test on three cell-lines (DE dataset). Observe that the performance of Epi2En varies greatly depending on the particular combination of testing/training, suggesting that some pair of cell-lines are more similar than others. For example, Epi2En trained on Gm12878 can predict enhancer regions in Hep-g2 cell line with high accuracy and vice versa. Instead, the performance of Epi2En trained on Hep-g2 is relatively poorly on H1-hesc and vice versa.

Next, we performed a series of experiments in which we predicted enhancer regions for a particular cell-line using a model trained on the other two cell-lines. Table 4.5 and Figure 4.10 shows the average testing performance for the three cell-lines (DE dataset). Epi2En produced an average AUC of 0.813, 0.61, 0.942 when testing on Gm12878, H1-hesc, and Hep-g2, respectively. Epi2En also produced a very high accuracy and precision

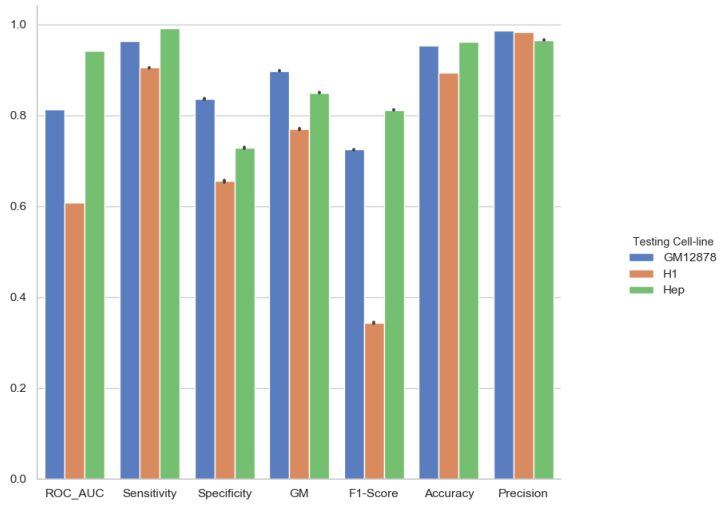

Figure 4.10: Epi2En performance on leave-one-out experiments (DE dataset).

Table 4.5: Leave-one-out analysis on both DE and PE datasets

Dataset	Train	Test	ROC	Sensitivity	Specificity	GM	F1-score	Accuracy	Precision
DE	Gm12878 + H1-hesc	Hep-g2	0.942	0.992	0.728	0.849	0.812	0.962	0.967
	Gm12878 + Hep-g2	H1-hesc	0.61	0.905	0.655	0.77	0.343	0.894	0.983
	Hep-g2 + H1-hesc	Gm12878	0.813	0.963	0.837	0.897	0.725	0.952	0.986
PE	Gm12878 + H1-hesc	Hep-g2	0.936	0.989	0.819	0.9	0.853	0.973	0.981
	Gm12878 + Hep-g2	H1-hesc	0.591	0.899	0.647	0.763	0.301	0..897	0.985
	Hep-g2 + H1-hesc	Gm12878	0.836	0.967	0.935	0.951	0.786	0.965	0.995

across all tested cell-lines. Figure 4.11 shows the AUC curve for the same leave-one-out experiments. Observe that the performance on H1-hesc is considerably worse than the two other cell-lines. Figure 4.12 shows the AUC curves for these experiments on the PE dataset. Both figures show similar predictive performances.

Finally, we have measured the predictive performance of Epi2En on two additional cell-line, namely Hela-S3 and K562. Genome-wide ChIP-Seq data for these cell-lines were

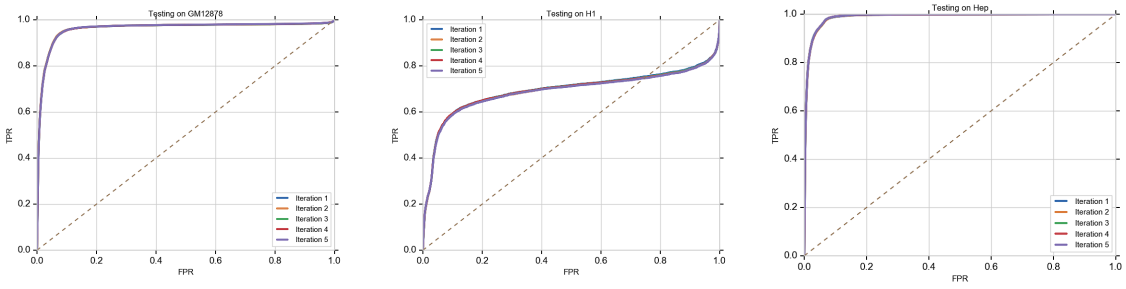

Figure 4.11: AUC curve for the leave-one-out experiments on the DE dataset. (left) train on GM12878 and H1-hesc, test on Hep-g2; (middle) train on GM12878 and Hep-g2, test on H1-hesc; (right) train on H1-hesc and Hep-g2, test on GM12878. The experiments were repeated five times.

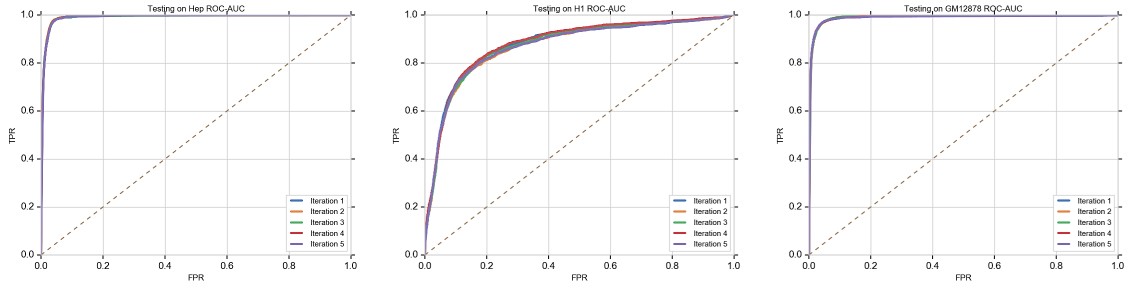

Figure 4.12: AUC curve for the leave-one-out experiments on the PE dataset. (left) train on GM12878 and H1-hesc, test on Hep-g2; (middle) train on GM12878 and Hep-g2, test on H1-hesc; (right) train on H1-hesc and Hep-g2, test on GM12878. The experiments were repeated five times.

downloaded from the ENCODE project. We trained our model on the three cell-lines (Gm12878, H1-hesc and Hep-g2) and tested on Hela-S3 and K562 and the enhancer list for both these cell-lines were obtained from PEDLA. Table 4.6 shows Epi2En's performance on Hela-S3 and K562. Observe that the combined model predicts the enhancers regions with more than 90% accuracy in Hela-S3 and more than 87% accuracy for K562, with more than 92% precision rate.

Table 4.6: Prediction performance of Epi2En on Hela-S3 and K562, based on training on Gm12878, H1-hesc and Hep-g2; GM is the geometric mean of sensitivity and specificity

Test	Accuracy	GM	F1-score	Precision
Hela-S3	90.06%	66.87%	53.18%	92.87%
K562	87.26%	53.44%	31.05%	92.86%

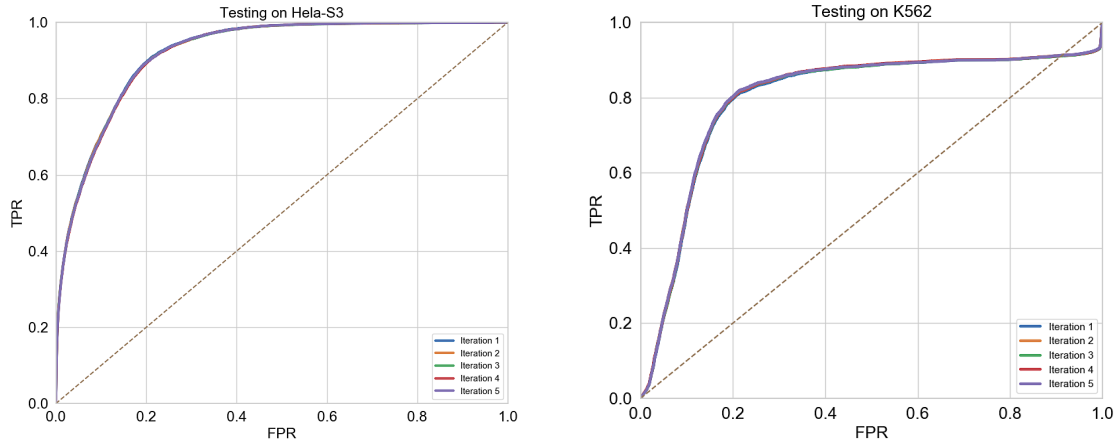

Figure 4.13: AUC curve of testing on Hela-S3 (left) and K562 (right) with the model trained on Gm12878, H1-hesc and Hep-g2 (DE datase). The experiments were repeated five times.

An additional validation marker was available for the K562 cell line in the DE dataset, namely genome-wide p300 binding sites. Enhancer regions are known to be enriched of p300 binding sites (see, e.g., [58]). In order to compute this enrichment, (i) we selected an arbitrary 2Mbp window on chromosome 1, (ii) we generated 10,000 random regions in this window according to the distribution of enhancer length as in Figure 4.2, (iii) we ran Epi2En on the twelve epigenetic markers for these 10,000 regions, (iv) we computed the fraction of predicted-positive regions that overlapped with p300 binding sites. Over five iteration of this experiment, we observed an average of 56.89% predicted-positive regions overlapping with a p300 binding site. In comparison, a classifier that predicts positives

with probability 1/10 generated predicted-positive regions that on average overlapped with 38.78% of the p300 binding sites.

4.4 Discussion and Conclusions

We introduced a convolutional neural network called Epi2En that can predict enhancer regions across multiple cell lines exclusively based on chromatin data, namely histone marks and CTCF, which are most common epigenetic features available across cell-lines. The dependency of Epi2En from only twelve epigenetic features enables our tool to be used in the widest set of cell-lines and tissue type.

Our extensive cross-validation experimental results show that Epi2En achieves excellent predictive accuracy, and it outperforms existing state-of-the-art methods. The performance of Epi2En across cell-lines is, however, not as impressive. Although Epi2En was superior to the other methods across cell-lines, there still is room for improvement towards building a truly general model for predicting enhancer regions across cell-lines and tissue-types. Even more ambitiously, based on the evidence of highly conserved general mechanisms of epigenetic regulation in mammals [42, 64], Epi2En could represent the first building block in predicting enhancers across species.

Epi2En can be readily extendable with the availability of more data from more cell-lines. Similar to any machine learning method, Epi2En's performance is highly dependent on the availability of training data. Epi2En would also be easy to extend to incorporate additional features resulting from new biological insights into their relevance in enhancer prediction. For instance, conformation capture data, such as Hi-C or ChIA-PET

are likely to improve the prediction of enhancer [127, 54]. All these additional data could enhance the generalization capability of Epi2En, moving towards the goal of achieving a truly general prediction tool for enhancers.

To the best of our knowledge, Epi2En is the first use of a convolutional neural network for enhancer prediction. The principle architecture of Epi2En is not only limited to predict enhancer regions. With properly prepared feature data, our model could be adopted for the prediction of other functional elements/domains such as promoter, insulator, and repressors.

Chapter 5

Classification of Stable/Unstable Nucleosome Sequences in *Plasmodium falciparum*

5.1 Introduction

Advancements in high-throughput DNA sequencing technology has enabled life scientists to carry out increasingly large-scale experiments. In genomics and epigenetics, it is relatively common to run multiple genome-wide experiments: for example, one can use ChIP-Seq/MNase-Seq/FAIRE-Seq/NOMe-Seq to obtain nucleosome levels or specific histone tail post-translational modifications (e.g., methylation, acetylation, phosphorylation, ubiquitination) at different cell types/tissues (which allows to understand cell type-specific marks), or at different time points for a particular cell process cycle (which allows to explore

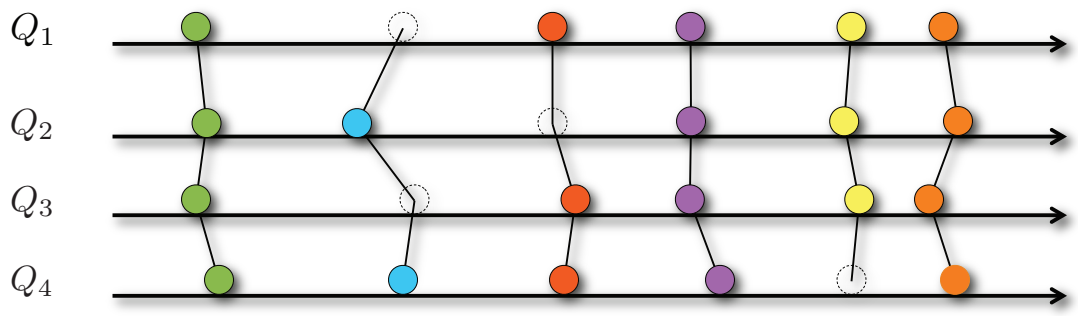

Figure 5.1: An illustration of the problem of aligning epigenetic features on four maps Q_1, Q_2, Q_3, Q_4; the input is a set of locations for the feature of interest (e.g., CHiP-seq peaks for nucleosome or specific histone marks) illustrated as circles here; the output is an assignment of features to trajectories, in a way that most parsimoniously explain the data; dotted circles indicate gaps or "missing features"

the dynamics of epigenetic marks). The ENCODE project [40], modENCODE [51, 122], phychENCODE [2], and the NIH Roadmap Epigenomics Project [8] are notable examples of these efforts. Similar comparative analysis of epigenetic marks could be carried out for a set of closely related organisms (e.g., human-chimp) if the correspondences between the genomes are known (e.g., using liftover by [71]).

Epigenetic maps are usually obtained by (i) sequencing a DNA sample enriched for a feature of interest, (ii) mapping short reads to the reference genome and (iii) running a peak-calling algorithm (e.g., MACS/MACS2 [149], NOrMAL [116], Puffin [114]). ThIEF:LP answers the fundamental question on how to compare multiple genome-wide epigenetic maps, when features are expected to shift or be missing. The model allows the possibility of the nucleosome of physically sliding along the genome or compensate for the noise in the peak detection process. In the example in Figure 5.1, the objective is to align four maps. Circles represent features to align; dashed circle mark the gap; trajectories are indicated with solid lines, which represent the most "likely" explanation of the data.

ThIEF:LP compares epigenetic maps by aligning them in a similar way the DNA sequences are aligned. It arranges multiple nucleosome/feature maps on top of each other, with the objective to build a *trajectory* for each individual nucleosomes/feature across time points or cell types, in a way that a total cost (i.e., total "traveled distance" in the case of nucleosomes) is minimized. Such a set of trajectories are called an *alignment*. Similar to multiple sequence alignment ThIEF:LP allow "insertion" or "deletions" of nucleosomes/features at specific time points or cell types. For instance, Figure 5.2 illustrates the output of ThIEF:LP in the IGV browser for a region of chromosome 10 of *P. falciparum*. ThIEF:LP aligned eight nucleosome maps over multiple time points, where each trajectory is assigned a unique ID, using alternating colors. Observe that in some trajectories (e.g., the one with ID 10_4546), nucleosomes are stable, while in others (e.g., the one with ID 10_4547) nucleosomes are evicted then bind later to the same location. We should note that it not easy to tag individual nucleosomes and image them under microscopes (see, e.g., [140]), so there is no way with current technology to obtain the "true" trajectories of nucleosomes genome-wide. Due to the lack of "ground-truth" about trajectories that would allow us to objectively evaluate the efficiency of ThIEF:LP, in this work, we decided to demonstrate the utility of ThIEF:LP by developing a supervised classifier that can accurately predict the position of stable and unstable nucleosomes from the primary DNA sequences. A nucleosome is defined as *stable* when it appears in approximately the same position in the genome at all time points of an experiment. A nucleosome is defined *unstable* when it appears approximately the same position in at most half of the time points of an experiment.

5.2 Finding trajectories of epigenetic marks with ThIEF:LP

An *epigenetic map* $Q = \{f_1, \ldots, f_n\}$ is defined as a set of n epigenetic features f_i, where a feature could be a nucleosome or a specific histone mark. Each feature $f \in Q$ is described by vector $f = (\mu, a_1, \ldots, a_l)$ where μ is the genomic coordinate of f in the genome (e.g., chromosome number and position in the chromosome) and each a_j, $j = 1, \ldots, l$ is an *attribute* of that feature (e.g., confidence score of a nucleosome, strength of a histone mark, p-value, peak width, "fuzziness" of a nucleosome, etc.). For convenience, it is assumed that features in Q are ordered by their position μ.

Given k epigenetic maps $Q_1, Q_2, \ldots, Q_k$, the goal is to align them so that the total alignment cost is minimized. For instance when $k = 2$, we either match feature $f \in Q_1$ to a feature $g \in Q_2$ so that we minimize a cost $\Delta(f, g)$ (e.g., some distance between the vectors $f = (\mu^f, a_1^f, \ldots, a_l^f)$ and $g = (\mu^g, a_1^g, \ldots, a_l^g)$) or report that feature $f \in Q_1$ has no match in Q_2 (insertion/deletion). In this latter case we will say that we have an alignment *gap*. For k maps, the cost function is $\Delta(q_1, q_2, \ldots, q_k)$ where q_i is a feature in the i-th map. Note that $\Delta(q_1, q_2, \ldots, q_k)$ is supposed to be defined for all combinations of $q_1, q_2, \ldots, q_k$.

While the framework allows features with $l \geq 0$ attributes, the paper only uses the genomic coordinate (i.e., $l = 0$). In that case $\Delta(q_1, q_2, \ldots, q_k)$ is simply the *absolute deviation*, i.e., the sum of absolute distance between the coordinate of each feature and the mean of those k coordinates.

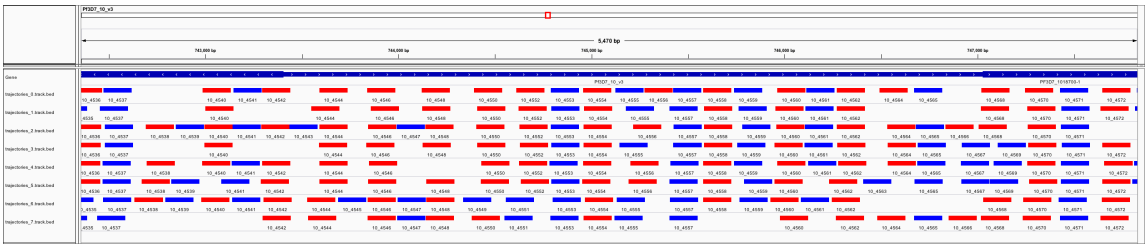

Figure 5.2: IGV snapshot of nucleosomes (red and blue rectangles - 147bps long) in region [742,356-747,829] of chromosome 10 of *P. falciparum* at eight time points; ThIEF assigns nucleosomes to trajectories which can be identified by an integer ID (zoom in to see them), and displayed in alternating colors

5.2.1 ThIEF:LP

ThIEF:LP casts the alignment problem as a *weighted k-partite matching* problem. Our problem is slightly more general than the k-partite matching problem because we need to deal with gaps. First ThIEF:LP builds a hyper-graph $H = (V, E)$, where each vertex $v \in V$ represent a genomic feature, and an hyper-edge $e \in E$ connects a subset of vertices (i.e., $e \subseteq V$) representing a possible alignment. By construction, the graph is k-partite: each hyper-edge contains at most one vertex from each partition (feature map). Hyper-edges can skip a partition to model gaps in the alignment.

We build the graph H iteratively. First, we create edges between the nodes in the first two maps (according to the criteria below), then extend them as hyper-edges to the other maps. Given a current (hyper)edge $e \in E$, in order to limit the size of H we extend it to a vertex v in the new partition only if (i) v has a position within $[-\delta, +\delta]$ of the position of the hyper-edge (computed as the average of the position of the nodes that belong to e) and (ii) it does not cross other hyper-edges.

Once H is built, we solve the weighted k-partite matching via an integer linear program (ILP) [18], as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{|E|} w_i x_i \\ \text{subject to} \quad & \sum_{i \in S_j} x_i = 1 \qquad j = 1, \ldots, |V| \\ & x_i \in \{0, 1\} \qquad i = 1, \ldots, |E| \end{aligned}$$

where binary variable x_i is associated to hyper-edge $e_i \in E$, S_j is the set of hyper-edges incident to node $v_j \in V$, and weight w_i is the absolute deviation of hyper-edge e_i, i.e., the sum of absolute distance between the coordinate of each feature in e_i and their mean.

The integer program is relaxed to a linear program by allowing each variable to take values in the interval $[0, 1]$. The linear program is solved using the off-the-shelf solver GLPK [96].

5.3 Detecting stable and unstable nucleosomes

As an example of application of ThIEF, we analyzed eight nucleosome maps from a study on the human malaria parasite [70]. In this study, synchronized *P. falciparum* parasites were collected at eight different stages of intra-erythrocytic development with five hour intervals (T5-T40). The eight samples were digested to enrich for nucleosome-bound DNA using the MNase protocol, then the digested samples were paired-end sequenced on an Illumina sequencing instrument. Raw reads provided by [70] were mapped to *P. falciparum* 3D7 genome v13.0 (available from `www.plasmoDB.org`) using Bowtie2 [77] allowing a maximum of one mismatch per read. Reads that mapped to multiple locations in the genome, reads that were PCR duplicates, and reads that mapped to ribosomal RNA or transfer RNA

were discarded. The final datasets contained about 409 M mapped paired-end reads, with an average read length of 100 bp.

We used PuFFIN [114] to generate nucleosomes positions (for all eight time points) from the aligned reads. Then, we used ThIEF:LP to produce nucleosome trajectories by solving 14 independent optimization problems (one for each chromosome of *P. falciparum*). PuFFIN detected a total of 770,238 nucleosomes, with an average of 96,280 nucleosomes per time point. Nucleosome positions were consistent with the one reported in [70].

ThIEF:LP reported a total of 141,363 trajectories (average of 10,097 trajectories for chromosome). A trajectory generated by ThIEF:LP was considered *stable* if it contained nucleosomes at each time point (i.e., no missing nucleosomes). Trajectories with no more than four nucleosomes (out of eight) were considered as *unstable* (i.e., at least three missing nucleosomes). According to this definition we detected 43,122 stable trajectories (about 31% of the total) and 50,783 unstable trajectories (about 36% of the total). Trajectories that were not stable or unstable were not used in the experiments (about 33% of the total). Chromosomes by chromosomes, the percentage of stable trajectories ranged from 33.23% to 35.39% of the total number of trajectories. The percentage of unstable trajectories for all chromosomes ranged from 33.5% to 40.4%. The distribution of stable and unstable trajectories along the *P. falciparum* chromosomes is shown in Figure 5.3. Observe that there are regions devoid of stable nucleosomes, and regions very enriched for unstable nucleosomes (e.g., the telomere of chromosome 13).

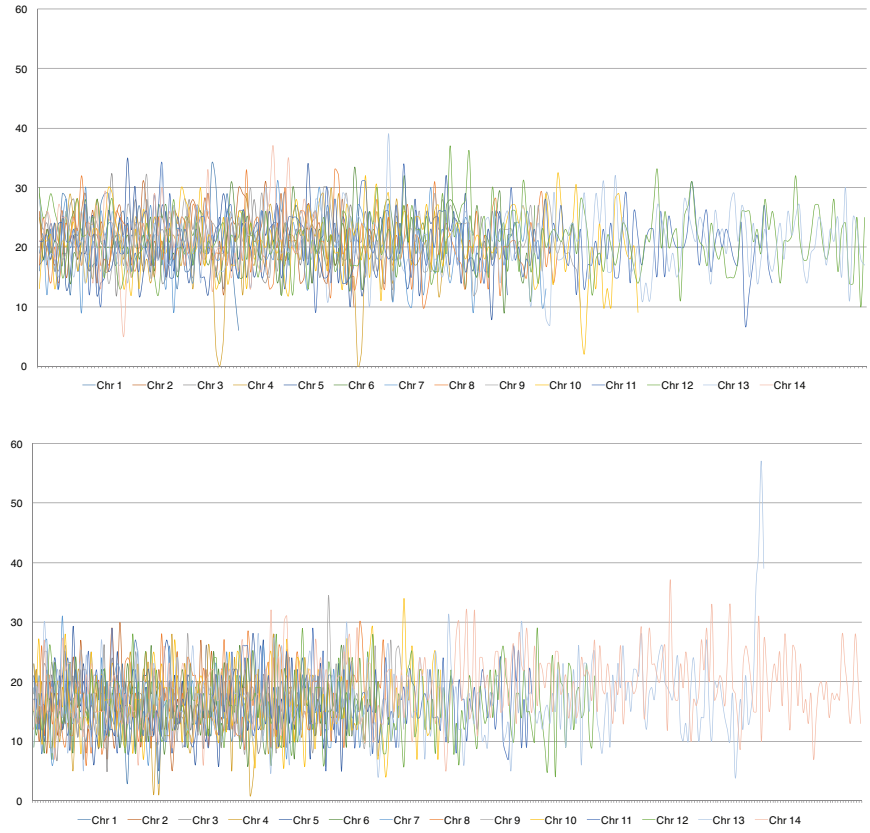

Figure 5.3: Distribution of stable (TOP) and unstable (BOTTOM) nucleosome trajectories along the 14 human malaria chromosomes (counts in a sliding window of 50Kbp)

5.4 A classifier for stable and unstable nucleosomes binding sites

First, we extracted the binding sites from the genome of *P. falciparum* for each trajectory. For each trajectory t labeled stable or unstable, we computed the average position of the nucleosomes in t, then selected 147 bp centered at that position from the malaria genome. Recall that a nucleosome consists of approximately 146-147 bp of DNA wrapped in superhelical turns around a histone octamer complex. Using this procedure we produced a training set composed of 43,122 147bp-long sequences representing stable nucleosomes,

and 50,783 147bp-long sequences representing unstable nucleosomes. We chose a Support Vector Machine (SVM) with radial basis function (RBF) kernel as our binary classifier. We explored many features to use in the classification, including k-mer distributions for $2 \leq k \leq 5$ and other physico-chemical properties of DNA (e.g., purine-pyrimidine, amino-keto, strong-weak H-bond). By running an extensive set of cross-validation experiments we determined that the five features described next were the most informative.

The first four features were obtained from the 3-mer distribution. For each sequence we collected its 3-mer frequencies, then computed the eigenvalues of four 4×4 matrices corresponding to 3-mers that have a middle nucleotide being `A`, `C`, `G`, and `T`, respectively. We represented each matrix with its leading eigenvalue, for a total of four features. The fifth feature was the DNA stability ΔG which is expressed by in terms of free energies of di-nucleotide stacks ΔG_{KL}, as described in [118]. The stability ΔG is measured by

$$\begin{aligned} \Delta G &= \frac{x_{GC}^2}{4}\left[\sum_{\mathcal{A}} \Delta G_{KL} + \sum_{\mathcal{B}} \Delta G_{KL} - \sum_{\mathcal{C}} \Delta G_{KL}\right] \\ &+ \frac{x_{GC}}{2}\left[\frac{1}{2}\sum_{\mathcal{C}} \Delta G_{KL} - \sum_{\mathcal{B}} \Delta G_{KL}\right] + \frac{1}{4}\sum_{\mathcal{B}} \Delta G_{KL} \end{aligned} \tag{5.1}$$

where $\mathcal{A} = \{GG, CC, GC, CG\}$, $\mathcal{B} = \{AA, TT, AT, TA\}$ and $\mathcal{C} = \{GA, AG, CT, TC, GT, TG, CA, AC\}$; ΔG_{KL} is the standard melting free energy parameter where di-nucleotide stacks are calculated from stacking free energy parameters ΔG_{KL}^{ST}. Table 5.1 lists the value of ΔG_{KL}^{ST} and ΔG_{KL}, obtained from [118]. We z-normalized the feature vectors before training the SVM.

KL	A	T	G	C
A	-1.11	-1.34	-1.06	-1.81
T	-0.19	-1.11	-0.55	-1.43
G	-1.43	-1.81	-1.44	-2.17
C	-0.55	-1.06	-0.91	-1.44

KL	A	T	G	C
A	-1.04	-1.27	-1.29	-2.04
T	-0.12	-1.04	-0.78	-1.66
G	-1.66	-2.04	-1.97	-2.70
C	-0.78	-1.29	-1.44	-1.97

Table 5.1: (LEFT) Stacking free energy parameters ΔG^{ST}_{KL}, (RIGHT) Standard melting free energy parameters ΔG_{KL}

Table 5.2 shows the classification results (precision, accuracy, recall, and area under the ROC curve) when training the SVM on the DNA binding sites of stable/unstable trajectories for even-numbered chromosomes of *P. falciparum*, and testing it on the odd-numbered chromosomes, for several choices of the penalty parameter C and kernel parameter γ in libSVM [19]. If TP, FP, TN and FN are true positive, false positive, true negative, and false negative, respectively, *precision* is defined as $TP/(TP + FP)$, *accuracy* is $(TP + TN)/(TP + TN + FP + FN)$, *recall* is $TP/(TP + FN)$, and the F-Score (also known as the F_1-score) is $(2\ precision\ \ recall)/(precision\ +\ recall)$. Observe that the SVM classifier makes a prediction for 66%-70% of tested nucleosome binding sites, and the precision of the prediction is very high. In 91%-95% of the predictions, the classifier can correctly determine solely from the DNA sequence whether the nucleosome will be stable or unstable. This is a surprisingly remarkable result.

γ	C	*precision*	*accuracy*	*recall*	*F-score*	AUC
0.5	2	91.32%	79.60%	70.50%	0.795687	0.905418
0.5	8	92.48%	79.18%	68.64%	0.787988	0.908831
2	0.125	92.40%	79.97%	70.23%	0.798028	0.91509
2	0.5	92.59%	79.37%	68.91%	0.79015	0.913866
2	2	93.23%	79.28%	68.19%	0.787704	0.91498
8	0.03125	95.17%	80.32%	68.55%	0.797005	0.924962
8	0.125	94.08%	79.33%	67.58%	0.786582	0.921428
8	0.5	93.95%	79.14%	67.32%	0.784348	0.919447
32	0.007812	96.31%	79.79%	66.70%	0.788176	0.926078
32	0.03125	96.10%	79.39%	66.11%	0.7833	0.929091
32	0.125	94.65%	79.27%	67.00%	0.784592	0.92311
128	0.125	94.72%	79.23%	66.88%	0.78399	0.923386
512	0.125	94.71%	79.03%	66.51%	0.781401	0.923546
2048	0.03125	95.26%	79.04%	66.10%	0.780456	0.926556
8192	0.03125	94.99%	79.07%	66.36%	0.781336	0.925219

Table 5.2: Classification results on the stable/unstable dataset for *P. falciparum*, by training the SVM on the odd-numbered chromosomes, and testing on the even-numbered chromosomes; AUC is the area under the ROC curve

Chapter 6

Conclusions

In this dissertation we showed how machine learning techniques can solve a few selected problems in the analysis of genomics data. We proposed a normalized mutual information-based gene ranking technique (mClass) that uses sparse somatic point mutation data for the classification of multiple cancer types. We have implemented a deep neural network architecture (DeeplyEssential) for the prediction of essential genes in microbes. We proposed a convolutional neural network framework (Epi2En) for genome-wide identification of enhancer regions across multiple human cell lines. Finally we built a prediction model to identify the dynamics of nucleosomes using DNA primary sequence and DNA binding preferences. Even though our proposed models produced state-of-the-art performances, we believe the further improvements are achievable by the availability of additional training data and new advances in machine learning.

6.1 Publications

This dissertation includes three peer-reviewed publications and one unpublished manuscript.

Full list of publication by Md Abid Hasan:

- Polishko A., **Hasan Md.**, Pan W., Bunnik EM., Roch KL., Lonardi S., *ThIEF: Finding Genome-wide Trajectories of Epigenetics Marks*, Proceedings of 17th International Workshop on Algorithms in Bioinformatics (WABI), Boston, USA, 2017

- **Hasan M.A.**, Lonardi S., *mClass: Cancer Type Classification with Somatic Point Mutation Data*, Proceedings of 16th International Conference RECOMB-CG, Qubec, Canada 2018

- **Hasan M.A.**, Lonardi S., *DeeplyEssential: A Deep Neural Network for Predicting Essential Genes in Microbes*, Proceedings of 6th International Workshop on CNB: Modelig, Analysis and Control, New York, 2019 (to appear in BMC Bioinformatics)

- **Hasan M.A.**, Lonardi S., *Epi2En: Convolutional Neural Network for Genome-wide Enhancer Prediction*, submitted to BMC Genomics, 2019

- Lonardi S., Muñoz-Amatriaín M., Liang Q., Shu S., Wanamaker S.I., Lo S., Tanskanen J., Zhu T., Luo M.C., Alhakami H., Ounit R., **Hasan M. A.**, Verdier J., Roberts P.A., Santos J. RP., Doležel J., Vrána J., Hokin S.A., Farmer A. D., Cannon S.B., Close T.J., *The Genome of Cowpea (Vigna unguiculate [L.] Walp.)*, The Plan Journal, v. 98 (5) pp. 767-782, 2019

Bibliography

[1] Marcio L Acencio and Ney Lemke. Towards the prediction of essential genes by integration of network topology, cellular localization and biological process information. *BMC Bioinformatics*, 10:290, September 2009.

[2] Schahram Akbarian, Chunyu Liu, et al. The PsychENCODE project. *Nature Publishing Group*, 2015.

[3] D Amar, S Izraeli, and R Shamir. Utilizing somatic mutation data from numerous studies for cancer research: proof of concept and applications. *Oncogene*, 36(24):3375–3383, June 2017.

[4] Awais Athar, Anja Füllgrabe, Nancy George, Haider Iqbal, Laura Huerta, Ahmed Ali, Catherine Snow, Nuno A Fonseca, Robert Petryszak, Irene Papatheodorou, Ugis Sarkans, and Alvis Brazma. ArrayExpress update - from bulk to single-cell expression data. *Nucleic Acids Res.*, 47(D1):D711–D715, January 2019.

[5] Karthik Azhagesan, Balaraman Ravindran, and Karthik Raman. Network-based features enable prediction of essential genes across diverse organisms. *PLoS One*, 13(12):e0208722, December 2018.

[6] Artem Barski, Suresh Cuddapah, Kairong Cui, Tae-Young Roh, Dustin E Schones, Zhibin Wang, Gang Wei, Iouri Chepelev, and Keji Zhao. High-resolution profiling of histone methylations in the human genome. *Cell*, 129(4):823–837, May 2007.

[7] R Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Netw.*, 5(4):537–550, 1994.

[8] Bradley E Bernstein, John A Stamatoyannopoulos, Joseph F Costello, Bing Ren, Aleksandar Milosavljevic, Alexander Meissner, Manolis Kellis, Marco A Marra, Arthur L Beaudet, Joseph R Ecker, Peggy J Farnham, Martin Hirst, Eric S Lander, Tarjei S Mikkelsen, and James A Thomson. The NIH roadmap epigenomics mapping consortium. *Nat Biotech*, 28(10):1045–1048, 10 2010.

[9] Léon Bottou. Stochastic learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning: ML Summer Schools*

2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures, pages 146–168. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[10] Alan P Boyle, Lingyun Song, Bum-Kyu Lee, Darin London, Damian Keefe, Ewan Birney, Vishwanath R Iyer, Gregory E Crawford, and Terrence S Furey. High-resolution genome-wide in vivo footprinting of diverse transcription factors in human cells. *Genome Res.*, 21(3):456–464, March 2011.

[11] Ryan P Browne, Paul D McNicholas, and Matthew D Sparling. Model-based learning using a mixture of mixtures of gaussian and uniform distributions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):814–817, April 2012.

[12] Michael Bulger and Mark Groudine. Enhancers: the abundance and function of regulatory sequences beyond promoters. *Dev. Biol.*, 339(2):250–257, March 2010.

[13] Michael Bulger and Mark Groudine. Functional and mechanistic diversity of distal transcription enhancers. *Cell*, 144(3):327–339, February 2011.

[14] Annalisa Buniello, Jacqueline A L MacArthur, Maria Cerezo, Laura W Harris, James Hayhurst, Cinzia Malangone, Aoife McMahon, Joannella Morales, Edward Mountjoy, Elliot Sollis, Daniel Suveges, Olga Vrousgou, Patricia L Whetzel, Ridwan Amode, Jose A Guillen, Harpreet S Riat, Stephen J Trevanion, Peggy Hall, Heather Junkins, Paul Flicek, Tony Burdett, Lucia A Hindorff, Fiona Cunningham, and Helen Parkinson. The NHGRI-EBI GWAS catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic Acids Res.*, 47(D1):D1005–D1012, January 2019.

[15] Z Cai, L Xu, Y Shi, M R Salavatipour, R Goebel, and G Lin. Using gene clustering to identify discriminatory genes with higher classification accuracy. In *Sixth IEEE Symposium on BioInformatics and BioEngineering (BIBE'06)*, pages 235–242. ieeexplore.ieee.org, October 2006.

[16] Eliezer Calo and Joanna Wysocka. Modification of enhancer chromatin: what, how, and why? *Mol. Cell*, 49(5):825–837, March 2013.

[17] Hannah Carter, Sining Chen, Leyla Isik, Svitlana Tyekucheva, Victor E Velculescu, Kenneth W Kinzler, Bert Vogelstein, and Rachel Karchin. Cancer-specific high-throughput annotation of somatic mutations: computational prediction of driver missense mutations. *Cancer Res.*, 69(16):6660–6667, August 2009.

[18] Yuk Hei Chan and Lap Chi Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical programming*, 135(1-2):123–148, 2012.

[19] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[20] Lei Chen, Xiuchun Ge, and Ping Xu. Identifying essential streptococcus sanguinis genes using genome-wide deletion mutation. *Methods Mol. Biol.*, 1279:15–23, 2015.

[21] Wei-Hua Chen, Pablo Minguez, Martin J Lercher, and Peer Bork. OGEE: an online gene essentiality database. *Nucleic Acids Res.*, 40(Database issue):D901–6, January 2012.

[22] Jian Cheng, Wenwu Wu, Yinwen Zhang, Xiangchen Li, Xiaoqian Jiang, Gehong Wei, and Shiheng Tao. A new computational strategy for predicting essential genes. *BMC Genomics*, 14:910, December 2013.

[23] Jian Cheng, Zhao Xu, Wenwu Wu, Li Zhao, Xiangchen Li, Yanlin Liu, and Shiheng Tao. Training set selection for the prediction of essential genes. *PLoS One*, 9(1):e86805, January 2014.

[24] Davide Chicco, Peter Sadowski, and Pierre Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '14, pages 533–540, New York, NY, USA, 2014. ACM.

[25] Ara Cho, Jung Eun Shim, Eiru Kim, Fran Supek, Ben Lehner, and Insuk Lee. MUFFINN: cancer gene discovery via network analysis of somatic mutation data. *Genome Biol.*, 17(1):129, June 2016.

[26] Ji-Hoon Cho, Dongkwon Lee, Jin Hyun Park, and In-Beum Lee. New gene selection method for classification of cancer subtypes considering within-class variation. *FEBS Lett.*, 551(1-3):3–7, 2003.

[27] C K Chow, H Zhu, J Lacy, M W Lingen, W P Kuo, and K Chan. A cooperative feature gene extraction algorithm that combines classification and clustering. In *2009 IEEE International Conference on Bioinformatics and Biomedicine Workshop*, pages 197–202. ieeexplore.ieee.org, November 2009.

[28] Anne E Clatworthy, Emily Pierson, and Deborah T Hung. Targeting virulence: a new paradigm for antimicrobial therapy. *Nat. Chem. Biol.*, 3(9):541–548, September 2007.

[29] Charles E Cook, Mary Todd Bergman, Robert D Finn, Guy Cochrane, Ewan Birney, and Rolf Apweiler. The european bioinformatics institute in 2016: Data growth and integration. *Nucleic Acids Res.*, 44(D1):D20–6, January 2016.

[30] Charles E Cook, Rodrigo Lopez, Oana Stroe, Guy Cochrane, Cath Brooksbank, Ewan Birney, and Rolf Apweiler. The european bioinformatics institute in 2018: tools, infrastructure and training. *Nucleic Acids Res.*, 47(D1):D15–D22, January 2019.

[31] Mélanie Courtot, Luca Cherubin, Adam Faulconbridge, Daniel Vaughan, Matthew Green, David Richardson, Peter Harrison, Patricia L Whetzel, Helen Parkinson, and Tony Burdett. BioSamples database: an updated sample metadata hub. *Nucleic Acids Res.*, 47(D1):D1172–D1178, January 2019.

[32] Menno P Creyghton, Albert W Cheng, G Grant Welstead, Tristan Kooistra, Bryce W Carey, Eveline J Steine, Jacob Hanna, Michael A Lodato, Garrett M Frampton, Phillip A Sharp, Laurie A Boyer, Richard A Young, and Rudolf Jaenisch. Histone H3K27ac separates active from poised enhancers and predicts developmental state. *Proc. Natl. Acad. Sci. U. S. A.*, 107(50):21931–21936, December 2010.

[33] Lara M Cullen and Greg M Arndt. Genome-wide screening for gene function using RNAi in mammalian cells. *Immunol. Cell Biol.*, 83(3):217–223, 2005.

[34] Fiona Cunningham, Premanand Achuthan, Wasiu Akanni, James Allen, M Ridwan Amode, Irina M Armean, Ruth Bennett, Jyothish Bhai, Konstantinos Billis, Sanjay Boddu, Carla Cummins, Claire Davidson, Kamalkumar Jayantilal Dodiya, Astrid Gall, Carlos García Girón, Laurent Gil, Tiago Grego, Leanne Haggerty, Erin Haskell, Thibaut Hourlier, Osagie G Izuogu, Sophie H Janacek, Thomas Juettemann, Mike Kay, Matthew R Laird, Ilias Lavidas, Zhicheng Liu, Jane E Loveland, José C Marugán, Thomas Maurel, Aoife C McMahon, Benjamin Moore, Joannella Morales, Jonathan M Mudge, Michael Nuhn, Denye Ogeh, Anne Parker, Andrew Parton, Mateus Patricio, Ahamed Imran Abdul Salam, Bianca M Schmitt, Helen Schuilenburg, Dan Sheppard, Helen Sparrow, Eloise Stapleton, Marek Szuba, Kieron Taylor, Glen Threadgold, Anja Thormann, Alessandro Vullo, Brandon Walts, Andrea Winterbottom, Amonida Zadissa, Marc Chakiachvili, Adam Frankish, Sarah E Hunt, Myrto Kostadima, Nick Langridge, Fergal J Martin, Matthieu Muffato, Emily Perry, Magali Ruffier, Daniel M Staines, Stephen J Trevanion, Bronwen L Aken, Andrew D Yates, Daniel R Zerbino, and Paul Flicek. Ensembl 2019. *Nucleic Acids Res.*, 47(D1):D745–D751, January 2019.

[35] Jose M Dana, Aleksandras Gutmanas, Nidhi Tyagi, Guoying Qi, Claire O'Donovan, Maria Martin, and Sameer Velankar. SIFTS: updated structure integration with function, taxonomy and sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. *Nucleic Acids Res.*, 47(D1):D482–D489, January 2019.

[36] Carrie A Davis, Benjamin C Hitz, Cricket A Sloan, Esther T Chan, Jean M Davidson, Idan Gabdank, Jason A Hilton, Kriti Jain, Ulugbek K Baymuradov, Aditi K Narayanan, Kathrina C Onate, Keenan Graham, Stuart R Miyasato, Timothy R Dreszer, J Seth Strattan, Otto Jolanki, Forrest Y Tanaka, and J Michael Cherry. The encyclopedia of DNA elements (ENCODE): data portal update. *Nucleic Acids Res.*, 46(D1):D794–D801, January 2018.

[37] Job Dekker, Marc A Marti-Renom, and Leonid A Mirny. Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data. *Nat. Rev. Genet.*, 14(6):390–403, June 2013.

[38] Jingyuan Deng, Lei Deng, Shengchang Su, Minlu Zhang, Xiaodong Lin, Lan Wei, Ali A Minai, Daniel J Hassett, and Long J Lu. Investigating the predictability of essential genes across distantly related organisms using an integrative approach. *Nucleic Acids Res.*, 39(3):795–807, February 2011.

[39] Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, Erik L L Sonnhammer, Layla Hirsh, Lisanna Paladin, Damiano Piovesan, Silvio C E Tosatto, and Robert D Finn. The pfam protein families database in 2019. *Nucleic Acids Res.*, 47(D1):D427–D432, January 2019.

[40] ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 2012.

[41] Jason Ernst and Manolis Kellis. ChromHMM: automating chromatin-state discovery and characterization. *Nat. Methods*, 9(3):215–216, February 2012.

[42] Genevieve D Erwin, Nir Oksenberg, Rebecca M Truty, Dennis Kostka, Karl K Murphy, Nadav Ahituv, Katherine S Pollard, and John A Capra. Integrating diverse datasets improves developmental enhancer prediction. *PLoS Comput. Biol.*, 10(6):e1003677, June 2014.

[43] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Trans. Neural Netw.*, 20(2):189–201, February 2009.

[44] Fazle E Faisal, Khalique Newaz, Julie L Chaney, Jun Li, Scott J Emrich, Patricia L Clark, and Tijana Milenković. GRAFENE: Graphlet-based alignment-free network approach integrates 3D structural and sequence (residue order) data to improve protein structural comparison. *Sci. Rep.*, 7(1):14890, November 2017.

[45] Michael Fernández and Diego Miranda-Saavedra. Genome-wide enhancer prediction from epigenetic signatures using genetic algorithm-optimized support vector machines. *Nucleic Acids Res.*, 40(10):e77, May 2012.

[46] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, Erik L L Sonnhammer, John Tate, and Marco Punta. Pfam: the protein families database. *Nucleic Acids Res.*, 42(Database issue):D222–30, January 2014.

[47] Alex Finnegan and Jun S Song. Maximum entropy methods for extracting the learned features of deep neural networks. *PLoS Comput. Biol.*, 13(10):e1005836, October 2017.

[48] Hiram A Firpi, Duygu Ucar, and Kai Tan. Discover regulatory DNA elements using chromatin signatures and artificial neural network. *Bioinformatics*, 26(13):1579–1586, July 2010.

[49] Long Gao, Yasin Uzun, Peng Gao, Bing He, Xiaoke Ma, Jiahui Wang, Shizhong Han, and Kai Tan. Identifying noncoding risk variants using disease-relevant gene regulatory networks. *Nat. Commun.*, 9(1):702, February 2018.

[50] Jeff Gauthier, Antony T Vincent, Steve J Charette, and Nicolas Derome. A brief history of bioinformatics. *Brief. Bioinform.*, August 2018.

[51] Mark B Gerstein, Zhi John Lu, et al. Integrative analysis of the *Caenorhabditis elegans* genome by the modENCODE project. *Science*, 2010.

[52] Guri Giaever, Angela M Chu, Li Ni, Carla Connelly, Linda Riles, Steeve Véronneau, Sally Dow, Ankuta Lucau-Danila, Keith Anderson, Bruno André, Adam P Arkin, Anna Astromoff, Mohamed El-Bakkoury, Rhonda Bangham, Rocio Benito, Sophie Brachat, Stefano Campanaro, Matt Curtiss, Karen Davis, Adam Deutschbauer, Karl-Dieter Entian, Patrick Flaherty, Francoise Foury, David J Garfinkel, Mark Gerstein, Deanna Gotte, Ulrich Güldener, Johannes H Hegemann, Svenja Hempel, Zelek Herman, Daniel F Jaramillo, Diane E Kelly, Steven L Kelly, Peter Kötter, Darlene LaBonte, David C Lamb, Ning Lan, Hong Liang, Hong Liao, Lucy Liu, Chuanyun Luo, Marc Lussier, Rong Mao, Patrice Menard, Siew Loon Ooi, Jose L Revuelta, Christopher J Roberts, Matthias Rose, Petra Ross-Macdonald, Bart Scherens, Greg Schimmack, Brenda Shafer, Daniel D Shoemaker, Sharon Sookhai-Mahadeo, Reginald K Storms, Jeffrey N Strathern, Giorgio Valle, Marleen Voet, Guido Volckaert, Ching-Yun Wang, Teresa R Ward, Julie Wilhelmy, Elizabeth A Winzeler, Yonghong Yang, Grace Yen, Elaine Youngman, Kexin Yu, Howard Bussey, Jef D Boeke, Michael Snyder, Peter Philippsen, Ronald W Davis, and Mark Johnston. Functional profiling of the saccharomyces cerevisiae genome. *Nature*, 418(6896):387–391, July 2002.

[53] T R Golub, D K Slonim, P Tamayo, C Huard, M Gaasenbeek, J P Mesirov, H Coller, M L Loh, J R Downing, M A Caligiuri, C D Bloomfield, and E S Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[54] David U Gorkin, Danny Leung, and Bing Ren. The 3D genome in transcriptional regulation and pluripotency. *Cell Stem Cell*, 14(6):762–775, June 2014.

[55] Jennifer Gudeman, Michael Jozwiakowski, John Chollet, and Michael Randell. Potential risks of pharmacy compounding. *Drugs R. D.*, 13(1):1–8, March 2013.

[56] H R Hassanzadeh and M D Wang. DeeperBind: Enhancing prediction of sequence specificities of DNA binding proteins. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 178–183. ieeexplore.ieee.org, December 2016.

[57] Bing He, Changya Chen, Li Teng, and Kai Tan. Global view of enhancer-promoter interactome in human cells. *Proc. Natl. Acad. Sci. U. S. A.*, 111(21):E2191–9, May 2014.

[58] Nathaniel D. Heintzman, Gary C. Hon, R. David Hawkins, Pouya Kheradpour, Alexander Stark, Lindsey F. Harp, Zhen Ye, Leonard K. Lee, Rhona K. Stuart, Christina W. Ching, Keith A. Ching, Jessica E. Antosiewicz-Bourget, Hui Liu, Xinmin Zhang, Roland D. Green, Victor V. Lobanenkov, Ron Stewart, James A. Thomson, Gregory E. Crawford, Manolis Kellis, and Bing Ren. Histone modifications at human enhancers reflect global cell-type-specific gene expression. *Nature*, 459(7243):108–112, 2009.

[59] Michael M Hoffman, Orion J Buske, Jie Wang, Zhiping Weng, Jeff A Bilmes, and William Stafford Noble. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nat. Methods*, 9(5):473–476, March 2012.

[60] Gangqing Hu, Kairong Cui, Daniel Northrup, Chengyu Liu, Chaochen Wang, Qingsong Tang, Kai Ge, David Levens, Colyn Crane-Robinson, and Keji Zhao. H2A.Z facilitates access of active and repressive complexes to chromatin in embryonic stem cell self-renewal and differentiation. *Cell Stem Cell*, 12(2):180–192, February 2013.

[61] Wenqi Hu, Susan Sillaots, Sebastien Lemieux, John Davison, Sarah Kauffman, Anouk Breton, Annie Linteau, Chunlin Xin, Joel Bowman, Jeff Becker, Bo Jiang, and Terry Roemer. Essential gene identification and drug target prioritization in aspergillus fumigatus. *PLoS Pathog.*, 3(3):e24, March 2007.

[62] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Marco Pagni, and Christian J A Sigrist. The PROSITE database. *Nucleic Acids Res.*, 34(Database issue):D227–30, January 2006.

[63] Sohyun Hwang, Chan Yeong Kim, Sunmo Yang, Eiru Kim, Traver Hart, Edward M Marcotte, and Insuk Lee. HumanNet v2: human gene networks for disease research. *Nucleic Acids Res.*, 47(D1):D573–D580, January 2019.

[64] Shalu Jhanwar, Stephan Ossowski, and Jose Davila-Velderrain. Genome-wide active enhancer identification using cell type-specific signatures of epigenomic activity. September 2018.

[65] Chunyuan Jin and Gary Felsenfeld. Nucleosome stability mediated by histone variants h3.3 and H2A.Z. *Genes Dev.*, 21(12):1519–1529, June 2007.

[66] Fulai Jin, Yan Li, Jesse R Dixon, Siddarth Selvaraj, Zhen Ye, Ah Young Lee, Chia-An Yen, Anthony D Schmitt, Celso A Espinoza, and Bing Ren. A high-resolution map of the three-dimensional chromatin interactome in human cells. *Nature*, 503(7475):290–294, November 2013.

[67] Sam John, Peter J Sabo, Thomas A Johnson, Myong-Hee Sung, Simon C Biddie, Stafford L Lightman, Ty C Voss, Sean R Davis, Paul S Meltzer, John A Stamatoyannopoulos, and Gordon L Hager. Interaction of the glucocorticoid receptor with the chromatin landscape. *Mol. Cell*, 29(5):611–624, March 2008.

[68] Mario Juhas, Leo Eberl, and George M Church. Essential genes as antimicrobial targets and cornerstones of synthetic biology. *Trends Biotechnol.*, 30(11):601–607, November 2012.

[69] Mario Juhas, Leo Eberl, and John I Glass. Essence of life: essential genes of minimal genomes. *Trends Cell Biol.*, 21(10):562–568, October 2011.

[70] Philip Reiner Kensche, Wieteke Anna Maria Hoeijmakers, Christa Geeke Toenhake, Maaike Bras, Lia Chappell, Matthew Berriman, and Richárd Bártfai. The nucleosome

landscape of plasmodium falciparum reveals chromatin architecture and dynamics of regulatory sequences. *Nucleic Acids Research*, 44(5):2110–2124, 2016.

[71] W James Kent, Robert Baertsch, Angie Hinrichs, Webb Miller, and David Haussler. Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc. Natl. Acad. Sci. U. S. A.*, 100(20):11484–11489, 30 September 2003.

[72] Dimitrios Kleftogiannis, Panos Kalnis, and Vladimir B Bajic. DEEP: a general computational framework for predicting enhancers. *Nucleic Acids Res.*, 43(1):e6, January 2015.

[73] Eugene V Koonin. Comparative genomics, minimal gene-sets and the last universal common ancestor. *Nat. Rev. Microbiol.*, 1(2):127–136, November 2003.

[74] Manching Ku, Jacob D Jaffe, Richard P Koche, Esther Rheinbay, Mitsuhiro Endoh, Haruhiko Koseki, Steven A Carr, and Bradley E Bernstein. H2A.Z landscapes and dual modifications in pluripotent and multipotent stem cells underlie complex genome regulatory functions. *Genome Biol.*, 13(10):R85, October 2012.

[75] Ivan V Kulakovskiy, Yulia A Medvedeva, Ulf Schaefer, Artem S Kasianov, Ilya E Vorontsov, Vladimir B Bajic, and Vsevolod J Makeev. HOCOMOCO: a comprehensive collection of human transcription factor binding sites models. *Nucleic Acids Res.*, 41(Database issue):D195–202, January 2013.

[76] N Kwak and Chong-Ho Choi. Input feature selection for classification problems. *IEEE Trans. Neural Netw.*, 13(1):143–159, 2002.

[77] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nat Meth*, 9(4):357–359, 04 2012.

[78] Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, and Victor Robles. Machine learning in bioinformatics. *Brief. Bioinform.*, 7(1):86–112, March 2006.

[79] Michael S Lawrence, Petar Stojanov, Craig H Mermel, James T Robinson, Levi A Garraway, Todd R Golub, Matthew Meyerson, Stacey B Gabriel, Eric S Lander, and Gad Getz. Discovery and saturation analysis of cancer genes across 21 tumour types. *Nature*, 505(7484):495–501, January 2014.

[80] Michael S Lawrence, Petar Stojanov, Paz Polak, Gregory V Kryukov, Kristian Cibulskis, Andrey Sivachenko, Scott L Carter, Chip Stewart, Craig H Mermel, Steven A Roberts, Adam Kiezun, Peter S Hammerman, Aaron McKenna, Yotam Drier, Lihua Zou, Alex H Ramos, Trevor J Pugh, Nicolas Stransky, Elena Helman, Jaegil Kim, Carrie Sougnez, Lauren Ambrogio, Elizabeth Nickerson, Erica Shefler, Maria L Cortés, Daniel Auclair, Gordon Saksena, Douglas Voet, Michael Noble, Daniel DiCara, Pei Lin, Lee Lichtenstein, David I Heiman, Timothy Fennell, Marcin Imielinski, Bryan

Hernandez, Eran Hodis, Sylvan Baca, Austin M Dulak, Jens Lohr, Dan-Avi Landau, Catherine J Wu, Jorge Melendez-Zajgla, Alfredo Hidalgo-Miranda, Amnon Koren, Steven A McCarroll, Jaume Mora, Brian Crompton, Robert Onofrio, Melissa Parkin, Wendy Winckler, Kristin Ardlie, Stacey B Gabriel, Charles W M Roberts, Jaclyn A Biegel, Kimberly Stegmaier, Adam J Bass, Levi A Garraway, Matthew Meyerson, Todd R Golub, Dmitry A Gordenin, Shamil Sunyaev, Eric S Lander, and Gad Getz. Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*, 499(7457):214–218, July 2013.

[81] Y Lecun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, November 1998.

[82] S M Lee, S M Yoon, and H Cho. Human activity recognition from accelerometer data using convolutional neural network. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 131–134. ieeexplore.ieee.org, February 2017.

[83] Li Li, Christian J Stoeckert, Jr, and David S Roos. OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Res.*, 13(9):2178–2189, September 2003.

[84] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, July 2006.

[85] Yifeng Li, Chih-Yu Chen, and Wyeth W Wasserman. Deep feature selection: Theory and application to identify enhancers and promoters. *J. Comput. Biol.*, 23(5):322–336, May 2016.

[86] Yuanyuan Li, Kai Kang, Juno M Krahn, Nicole Croutwater, Kevin Lee, David M Umbach, and Leping Li. A comprehensive genomic pan-cancer classification using the cancer genome atlas gene expression data. *BMC Genomics*, 18(1):508, July 2017.

[87] Yan Lin, Fa-Zhan Zhang, Kai Xue, Yi-Zhou Gao, and Feng-Biao Guo. Identifying bacterial essential genes based on a feature-integrated method. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, February 2017.

[88] Yan Lin and Randy Ren Zhang. Putative essential and core-essential genes in mycoplasma genomes. *Sci. Rep.*, 1:53, August 2011.

[89] Feng Liu, Hao Li, Chao Ren, Xiaochen Bo, and Wenjie Shu. PEDLA: predicting enhancers with a deep learning-based algorithmic framework. *Sci. Rep.*, 6:28517, June 2016.

[90] Xiao Liu, Bao-Jin Wang, Luo Xu, Hong-Ling Tang, and Guo-Qing Xu. Selection of key sequence-based features for prediction of essential genes in 31 diverse bacterial species. *PLoS One*, 12(3):e0174638, March 2017.

[91] Dan L Longo. Tumor heterogeneity and personalized medicine. *N. Engl. J. Med.*, 366(10):956–957, March 2012.

[92] G G Loots, R M Locksley, C M Blankespoor, Z E Wang, W Miller, E M Rubin, and K A Frazer. Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons. *Science*, 288(5463):136–140, April 2000.

[93] Yao Lu, Jingyuan Deng, Judith C Rhodes, Hui Lu, and Long Jason Lu. Predicting essential genes for identifying potential drug targets in aspergillus fumigatus. *Comput. Biol. Chem.*, 50:29–40, June 2014.

[94] Yiming Lu, Wubin Qu, Guangyu Shan, and Chenggang Zhang. DELTA: A distal enhancer locating tool based on AdaBoost algorithm and shape features of chromatin modifications. *PLoS One*, 10(6):e0130622, June 2015.

[95] Hao Luo, Yan Lin, Feng Gao, Chun-Ting Zhang, and Ren Zhang. DEG 10, an update of the database of essential genes that includes both protein-coding genes and noncoding genomic elements. *Nucleic Acids Res.*, 42(Database issue):D574–80, January 2014.

[96] Andrew Makhorin. Glpk (gnu linear programming kit), 2008.

[97] Noël Malod-Dognin and Nataša Pržulj. GR-Align: fast and flexible alignment of protein 3D structures using graphlet degree similarity. *Bioinformatics*, 30(9):1259–1265, May 2014.

[98] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michal Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.*, 47(D1):D930–D940, January 2019.

[99] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Brief. Bioinform.*, 18(5):851–869, September 2017.

[100] Alex L Mitchell, Teresa K Attwood, Patricia C Babbitt, Matthias Blum, Peer Bork, Alan Bridge, Shoshana D Brown, Hsin-Yu Chang, Sara El-Gebali, Matthew I Fraser, Julian Gough, David R Haft, Hongzhan Huang, Ivica Letunic, Rodrigo Lopez, Aurélien Luciani, Fabio Madeira, Aron Marchler-Bauer, Huaiyu Mi, Darren A Natale, Marco Necci, Gift Nuka, Christine Orengo, Arun P Pandurangan, Typhaine Paysan-Lafosse, Sebastien Pesseat, Simon C Potter, Matloob A Qureshi, Neil D Rawlings, Nicole Redaschi, Lorna J Richardson, Catherine Rivoire, Gustavo A Salazar, Amaia Sangrador-Vegas, Christian J A Sigrist, Ian Sillitoe, Granger G Sutton, Narmada Thanki, Paul D Thomas, Silvio C E Tosatto, Siew-Yit Yong, and Robert D Finn. InterPro in 2019: improving coverage, classification and access to protein sequence annotations. *Nucleic Acids Res.*, 47(D1):D351–D360, January 2019.

[101] Estuko N. Moriyama. Codon usage, 2003.

[102] A R Mushegian and E V Koonin. A minimal gene set for cellular life derived by comparison of complete bacterial genomes. *Proc. Natl. Acad. Sci. U. S. A.*, 93(19):10268–10273, September 1996.

[103] Dawit Nigatu, Patrick Sobetzko, Malik Yousef, and Werner Henkel. Sequence-based information-theoretic features for gene essentiality prediction. *BMC Bioinformatics*, 18(1):473, November 2017.

[104] L W Ning, H Lin, H Ding, J Huang, N Rao, and F B Guo. Predicting bacterial essential genes using only sequence composition information. *Genet. Mol. Res.*, 13(2):4564–4572, June 2014.

[105] Marcelo A Nobrega, Ivan Ovcharenko, Veena Afzal, and Edward M Rubin. Scanning human gene deserts for long-range enhancers. *Science*, 302(5644):413, October 2003.

[106] Chin-Tong Ong and Victor G Corces. Enhancer function: new insights into the regulation of tissue-specific gene expression. *Nat. Rev. Genet.*, 12(4):283–293, April 2011.

[107] Krishnaveni Palaniappan and Sumitra Mukherjee. Predicting "essential" genes across microbial genomes: A machine learning approach. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 2, pages 189–194. ieeexplore.ieee.org, 2011.

[108] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, August 2005.

[109] Len A Pennacchio, Wendy Bickmore, Ann Dean, Marcelo A Nobrega, and Gill Bejerano. Enhancers: five essential questions. *Nat. Rev. Genet.*, 14(4):288–295, April 2013.

[110] Yasset Perez-Riverol, Attila Csordas, Jingwen Bai, Manuel Bernal-Llinares, Suresh Hewapathirana, Deepti J Kundu, Avinash Inuganti, Johannes Griss, Gerhard Mayer, Martin Eisenacher, Enrique Pérez, Julian Uszkoreit, Julianus Pfeuffer, Timo Sachsenberg, Sule Yilmaz, Shivani Tiwary, Jürgen Cox, Enrique Audain, Mathias Walzer, Andrew F Jarnuczak, Tobias Ternent, Alvis Brazma, and Juan Antonio Vizcaíno. The PRIDE database and related tools and resources in 2019: improving support for quantification data. *Nucleic Acids Res.*, 47(D1):D442–D450, January 2019.

[111] Michael W Perry, Alistair N Boettiger, Jacques P Bothma, and Michael Levine. Shadow enhancers foster robustness of drosophila gastrulation. *Curr. Biol.*, 20(17):1562–1567, September 2010.

[112] Kitiporn Plaimas, Roland Eils, and Rainer König. Identifying essential genes in bacterial metabolic networks with machine learning methods. *BMC Syst. Biol.*, 4:56, May 2010.

[113] Jennifer L Plank and Ann Dean. Enhancer function: mechanistic and genome-wide insights come together. *Mol. Cell*, 55(1):5–14, July 2014.

[114] A. Polishko, E. M. Bunnik, K. G. Le Roch, and S. Lonardi. PuFFIN: A parameter-free method to build nucleosome maps from paired-end reads. *BMC Bioinformatics*, 15(Suppl 9):S11, 2014.

[115] Anton Polishko, Md Abid Hasan, Weihua Pan, Evelien M Bunnik, Karine Le Roch, and Stefano Lonardi. ThIEF: Finding genome-wide trajectories of epigenetics marks. In *17th International Workshop on Algorithms in Bioinformatics, WABI 2017*, page 19. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, August 2017.

[116] Anton Polishko, Nadia Ponts, Karine G Le Roch, and Stefano Lonardi. NORMAL: accurate nucleosome positioning using a modified gaussian mixture model. *Bioinformatics (Oxford, England)*, 28(12):i242–9, June 2012.

[117] Shyam Prabhakar, Francis Poulin, Malak Shoukry, Veena Afzal, Edward M Rubin, Olivier Couronne, and Len A Pennacchio. Close sequence comparisons are sufficient to identify human cis-regulatory elements. *Genome Res.*, 16(7):855–863, July 2006.

[118] Ekaterina Protozanova, Peter Yakovchuk, and Maxim D Frank-Kamenetskii. Stacked-unstacked equilibrium at the nick site of DNA. *J Mol Biol*, 342(3):775–785, Sep 2004.

[119] Nisha Rajagopal, Wei Xie, Yan Li, Uli Wagner, Wei Wang, John Stamatoyannopoulos, Jason Ernst, Manolis Kellis, and Bing Ren. RFECS: a random-forest based algorithm for enhancer identification from chromatin state. *PLoS Comput. Biol.*, 9(3):e1002968, March 2013.

[120] Suhas S P Rao, Miriam H Huntley, Neva C Durand, Elena K Stamenova, Ivan D Bochkov, James T Robinson, Adrian L Sanborn, Ido Machol, Arina D Omer, Eric S Lander, and Erez Lieberman Aiden. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7):1665–1680, December 2014.

[121] Boris Reva, Yevgeniy Antipin, and Chris Sander. Predicting the functional impact of protein mutations: application to cancer genomics. *Nucleic Acids Res.*, 39(17):e118, September 2011.

[122] S Roy, J Ernst, P V Kharchenko, and P Kheradpour. Identification of functional elements and regulatory circuits by Drosophila modENCODE. *Science*, 2010.

[123] Nina R Salama, Benjamin Shepherd, and Stanley Falkow. Global transposon mutagenesis and essential gene analysis of helicobacter pylori. *J. Bacteriol.*, 186(23):7926–7935, December 2004.

[124] Jürgen Schmidhuber. Deep learning in neural networks: an overview. *Neural Netw.*, 61:85–117, January 2015.

[125] Tom Sexton and Giacomo Cavalli. The role of chromosome domains in shaping the functional genome. *Cell*, 160(6):1049–1059, March 2015.

[126] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.*, 19:221–248, June 2017.

[127] Andrea Smallwood and Bing Ren. Genome organization and long-range regulation of gene expression by enhancers. *Curr. Opin. Cell Biol.*, 25(3):387–394, June 2013.

[128] Kai Song, Tuopong Tong, and Fang Wu. Predicting essential genes in prokaryotic genomes using a linear method: ZUPLS. *Integr. Biol.*, 6(4):460–469, April 2014.

[129] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[130] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[131] Bwkp Stewart, Christopher P Wild, and Others. World cancer report 2014. 2014.

[132] Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, Lars J Jensen, and Christian von Mering. The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic Acids Res.*, 45(D1):D362–D368, January 2017.

[133] The RNAcentral Consortium. RNAcentral: a hub of information for non-coding RNA sequences. *Nucleic Acids Res.*, 47(D1):D221–D229, January 2019.

[134] Katarzyna Tomczak, Patrycja Czerwińska, and Maciej Wiznerowicz. The cancer genome atlas (TCGA): an immeasurable source of knowledge. *Contemp. Oncol.*, 19(1A):A68–77, 2015.

[135] UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.*, 47(D1):D506–D515, January 2019.

[136] Stijn Marinus Van Dongen. *Graph clustering by flow simulation*. PhD thesis, 2000.

[137] Axel Visel, Matthew J Blow, Zirong Li, Tao Zhang, Jennifer A Akiyama, Amy Holt, Ingrid Plajzer-Frick, Malak Shoukry, Crystal Wright, Feng Chen, Veena Afzal, Bing Ren, Edward M Rubin, and Len A Pennacchio. ChIP-seq accurately predicts tissue-specific activity of enhancers. *Nature*, 457(7231):854–858, February 2009.

[138] Axel Visel, Shyam Prabhakar, Jennifer A Akiyama, Malak Shoukry, Keith D Lewis, Amy Holt, Ingrid Plajzer-Frick, Veena Afzal, Edward M Rubin, and Len A Pennacchio. Ultraconservation identifies a small subset of extremely constrained developmental enhancers. *Nat. Genet.*, 40(2):158–160, February 2008.

[139] Axel Visel, Edward M Rubin, and Len A Pennacchio. Genomic views of distant-acting enhancers. *Nature*, 461(7261):199–205, September 2009.

[140] Mari-Liis Visnapuu and Eric C Greene. Single-molecule imaging of DNA curtains reveals intrinsic energy landscapes for nucleosome deposition. *Nat Struct Mol Biol*, 16(10):1056–1062, 10 2009.

[141] Wen Wei, Lu-Wen Ning, Yuan-Nong Ye, and Feng-Biao Guo. Geptop: a gene essentiality prediction tool for sequenced bacterial genomes based on orthology and phylogeny. *PLoS One*, 8(8):e72343, August 2013.

[142] Kun Yang, Zhipeng Cai, Jianzhong Li, and Guohui Lin. A stable gene selection in microarray data analysis. *BMC Bioinformatics*, 7:228, April 2006.

[143] Yuan-Nong Ye, Zhi-Gang Hua, Jian Huang, Nini Rao, and Feng-Biao Guo. CEG: a database of essential gene clusters. *BMC Genomics*, 14:769, November 2013.

[144] H Yin and K Gai. An empirical study on preprocessing High-Dimensional Class-Imbalanced data for classification. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 1314–1319. ieeexplore.ieee.org, August 2015.

[145] Yongming Yu, Licai Yang, Zhiping Liu, and Chuansheng Zhu. Gene essentiality prediction based on fractal features and machine learning. *Mol. Biosyst.*, 13(3):577–584, February 2017.

[146] Yuchen Yuan, Yi Shi, Changyang Li, Jinman Kim, Weidong Cai, Zeguang Han, and David Dagan Feng. DeepGene: an advanced cancer type classifier based on deep learning and somatic point mutations. *BMC Bioinformatics*, 17(Suppl 17):476, December 2016.

[147] Junjun Zhang, Rosita Bajari, Dusan Andric, Francois Gerthoffert, Alexandru Lepsa, Hardeep Nahal-Bose, Lincoln D Stein, and Vincent Ferretti. The international cancer genome consortium data portal. *Nat. Biotechnol.*, 37(4):367–369, April 2019.

[148] Xue Zhang, Marcio L Acencio, and Ney Lemke. Corrigendum: Predicting essential genes and proteins based on machine learning and network topological features: A comprehensive review. *Front. Physiol.*, 7:617, December 2016.

[149] Yong Zhang, Tao Liu, Clifford A. Meyer, Jérôme Eeckhoute, David S. Johnson, Bradley E. Bernstein, Chad Nusbaum, Richard M. Myers, Myles Brown, Wei Li, and X. Shirley Liu. Model-based analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9):R137, 2008.