PROMOTER RECOGNITION IN HUMAN DNA SEQUENCE USING DEEP LEARNING

Project report submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

by

AyshikaKapoor(161080)

Uday Sharma(161683)

UNDER THE GUIDANCE OF

Dr. Sunil Datt Sharma

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May 2020

TABLE OF CONTENTS

DECLARATION BY THE SCHOLAR

We hereby declare that work reported in the B-Tech project report entitled "**PROMOTER RECOGNITION IN HUMAN DNA SEQUENCE USING DEEP LEARNING**" submitted at **Jaypee University of Information Technology, Waknaghat, India,** is an authentic record of our work carried out under the supervision of **Dr. Sunil DattSharma.** We have not submitted this work elsewhere for any other degree or diploma.

Ayshika.

Uday Sharma

Ayshika Kapoor : 161080

Uday Sharma : 161683

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

[illegible]

Dr. Sunil Datt Sharma

Date :

Head of the Department/Project Coordinator

ACKNOWLEDGMENT

We take this opportunity to express gratitude to our guide **Dr. Sunil Datt Sharma** for his insightful advice, motivating suggestions, valuable guidance, help and support in the successful completion of this project. We are also thankful for his constant encouragement and advice throughout our B.Tech. Program. The facilities provided by the department during the B.Tech project are also equally acknowledgeable. We would like to convey our thanks to **Prof. M.J Nigam** (HOD), teaching and non teaching staff of Electronics and Communication Engineering Department for their invaluable help and motivational support.

Guided by: -

Dr. Sunil Datt Sharma

Ayshika   Uday Sharma

Ayshika Kapoor : 161080   **Uday Sharma : 161683**

CERTIFICATE

This is to certify that the work which is being presented in this project report titled "**Promoter Recognition In Human DNA Sequence Using Deep Learning**" for partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Electronics and Communication Engineering** and submitted to the Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Ayshika Kapoor(161080) and Uday sharma (161683) during a period of July 2019 to May 2020 under the supervision of Dr. Sunil Datt Sharma (Assistant Professor (Senior Grade)), Department of Electronics and Communication Engineering), Jaypee University of Information Technology, Waknaghat.
The above statement is made correct to the best of our knowledge.

Date:-…………………………….

[illegible]

Dr. Sunil Datt Sharma

Assistant Professor (Senior Grade)

Department of Electronics and Communication Engineering

JUIT, Waknaghat.

LIST OF ACRONYMS AND ABBREVIATIONS

DCDE	Deep Convolution Divergence Encoding
DNADeoxyribonucleic Acid	
RNA	Ribonucleic Acid
mRNA	Messenger Ribonucleic Acid
UV	UltraViolet
TSS	Transcription Start Site
CpG	Cytosine Photodiester Guanine
CG	Cytosine Guanine
LS-GKM	Large Scale GKM
SD	Statistical Divergence
KL	Kullback Leibler
JD	Jensen Shannon Divergence
CNN	Convolution Neural Network
SVM	Support Vector Machine
MSVM	Multiple Support Vector Machine
NCBI	National Centre for Biotechnology Information
KLD	Kullback Leibler Divergence
ReLU	Rectified Linear Unit
TN	True Negative
TP	True Positive
FN	False Negative
FP	False Positive

S_n	Sensitivity
S_p	Specificity
ACP	Averaged Conditional Probability
STFT	Short Time Fourier Transform
DSP	Digital Signal Processing
DFT	Discrete Fourier Transform
A	Adenine
T	Thymine
G	Guanine
C	Cytosine

LIST OF SYMBOLS

P_p	Probability density for promoters
P_{np}	Probability density for non – promoters
Ik	Informative kmers
X	Input vector
Y	Output
λ	Threshold
X[k]	Input signal
G[k]	Window function for L – point

LIST OF FIGURES

List of Figures	**Page Number**
Fig.1.1. Protein Synthesis	1
Fig.1.1.1. Structure of RNA and DNA	2
Fig.1.1.2. Central Dogma	3
Fig.1.1.4. Transcription and Translation	4
Fig.1.2.1 Eukaryotic Cell	5
Fig.1.2.2 Prokaryotic bacteria	6
Fig.1.3 Promoters	6
Fig.1.3.1 Promoter Sequences	7
Fig.2. Framework of DCDE	13
Fig.3.1. Output of Informative kmers Settlement	17
Fig.4. Basic CNN structure used in secondary encoding	19
Fig.4.1. Multiplication with kernel	19
Fig.4.3. After Pooling layer, Flattened as Fully Connected Layer	21
Fig.5.1. Example 1.1	24
Fig.5.2. Example 1.2	24
Fig.5.3. Example 2.1	25
Fig.5.4. Example 2.2	25
Fig.5.5. Example 3.1	25
Fig.5.1.1. Example 4.1 27	
Fig.5.1.2. Example 4.2	27
Fig.5.2.1. High gamma	27
Fig.5.2.2. Low gamma	27
Fig.5.3.1. Good margin	28
Fig.5.3.2. Bad margin	28
Fig.7.1. Framework for classifying promoters and non - promoters33	
Fig.8.1. Binary sequence	35
Fig.8.2. Spectrograms of a DNA sequence	36
Fig.9.1. Working of CNN	38

LIST OF TABLES

ABSTRACT

Extraction of human promoter features is still a major obstacle in genome research but knowing gene expression control is an essential part of understanding gene activity.Consequently, reliable recognition and characterization of promoters is a high priority in the study of the human genome. Promoter information can be helpful in elucidating gene control and expression pathways, which may also shed light on the role of modern which uncharacterized genes.While several machine learning algorithms were developed for the recognition of promotersbut due to the complex existence of promoters none had produced satisfactory performance. For gene analysis, we had proposed a model by applying technique using **Spectrograms** to extract important features from promoters. The spectrograms enable continuous visualization of the local frequency in the nucleotide series, and indeed the individual nucleotide information shown by the spectrogram colors. Then, CNN is applied on Spectrograms to extract essential features and classify Promoters and Non – Promoters. With our results we can see that the efficiency of our model may be further improved by extracting further characteristics from spectrogram.Analyzing spectrograms is also a successful means of recognizing promoters.

CHAPTER 1

INTRODUCTION

Promoter assumption is an aspect popular in many gene prediction techniques. The diseases caused by promoters such as diabetes, asthama, beta thalassemia, etc. have been found in many recent research. Therefore the detection of promoters is involving the new methodology in which researchers are involved. We will address a DCDE algorithm in our paper, and later suggest our own algorithm using spectrograms for the identification of promoters.

1.1 Protein Synthesis

Protein production in living-things cells is called Protein Synthesis. Two processes involved are: **Transcription** and **Translation**.

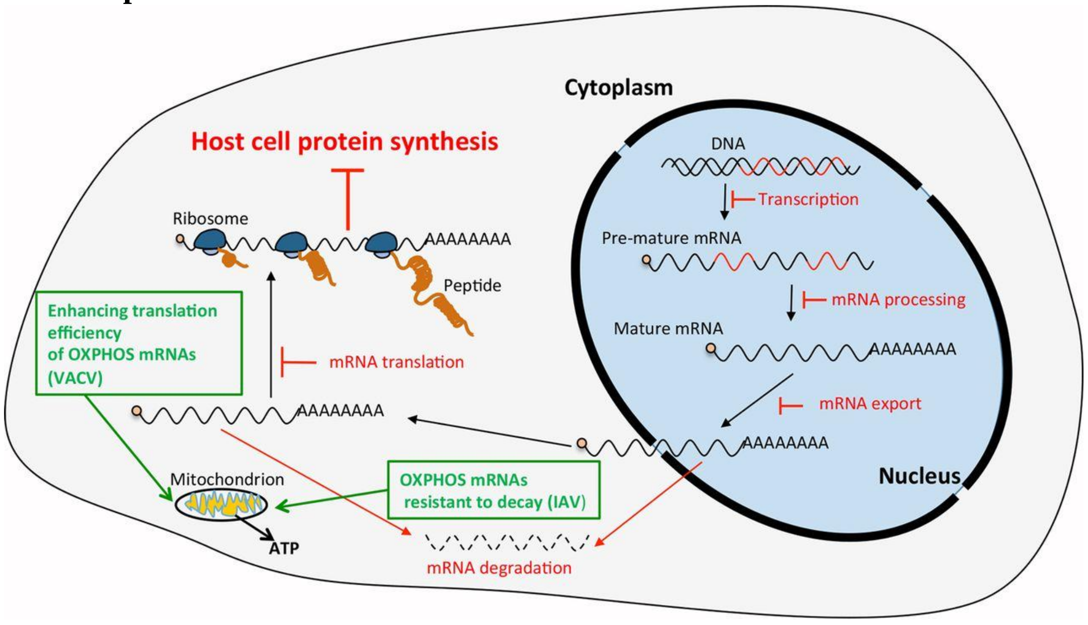


Fig.1.1. Protein Synthesis

Table 1.1 Important Biological Terms

Term	**Meaning**
Central dogma	This cycle in which material in genes flows into proteins as DNA is translated into RNA and then RNA into protein: DNA → RNA → protein
RNA	RNA indicates ribonucleic acid. It is single-stranded nucleic acid that holds the instructions written in Deoxyribonucleic Acid.
Polypeptide	They 're chainsmade up of amino acids.
Codon	Successive arrangement of 3 nucleotides during translation that corresponds to

	the maximum amino acid or the stop / start signal.
Transcription	Where a gene sequence of DNA is translated to form an RNA molecule
Translation	This method is called synthesis, during which a messenger rna is used to combine amino acids as chains of polypeptide.
Mutation	A alteration of a sequence of genes.

1.1.1 Structure Of RNA

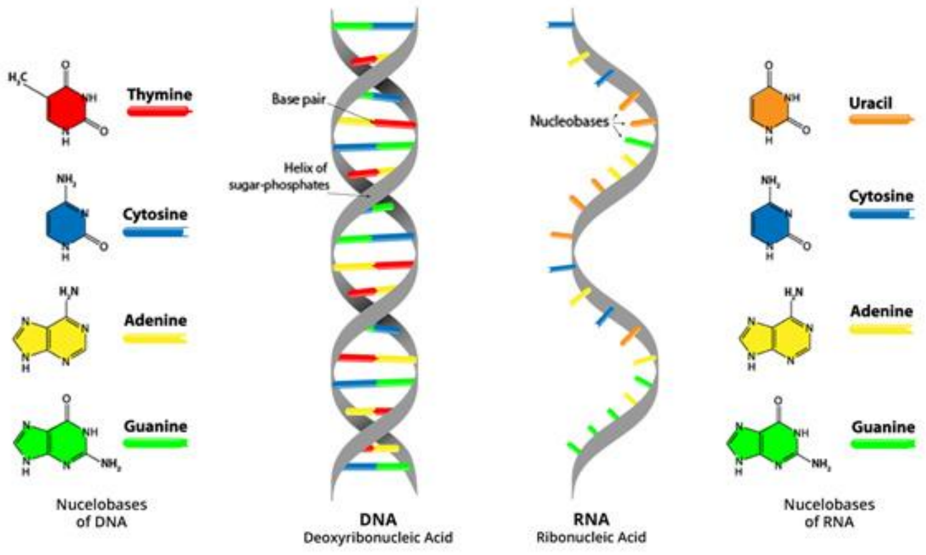


Fig.1.1.1 Structure of RNA and DNA

1. DNA alone can't compensate for gene expression. Helping perform instructions in DNA is required.
2. The nucleotide is constituted, like DNA and RNA, of something like a 5-carbon sugar ribose, a phosphate ring, and even a nitrogen base.

But there are three major distinctions between DNA and RNA:

Ribonucleic acid (RNA) uses ribose sugar, rather than deoxyribose.

3. The RNA is typically single-stranded, rather than double-stranded.
4. RNA includesuracil than just thymine.

The above points help to differentiate DNA from RNA through enzymes within the cell.

1.1.2 Central dogma

In Gene an encoding of polypeptides is expressed in two phases. Knowledge in this cycle flows from DNA to RNA into protein, a spatial relation known as the basic dogma of molecular biology.

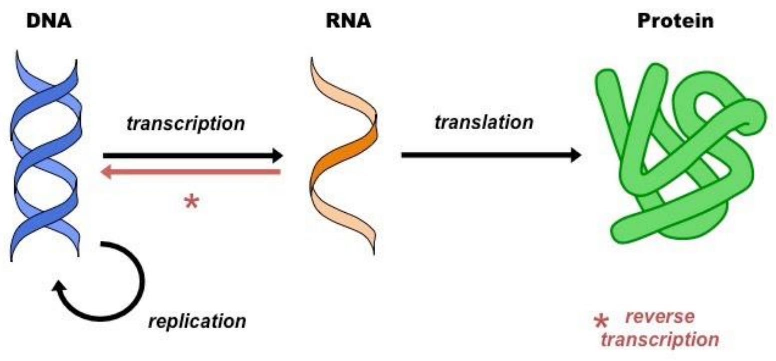


Fig. 1.1.2 Central Dogma

1.1.3 The genetic code

Transcription is the first step in decoding genetic messages, during which a nucleotide sequence is copied from DNA into RNA. The next step in genetic encoding is to join together amino acids to form a protein.

The commands in which amino acids are joined together determine a protein 's shape, properties, and function.

1.1.4 Transcription and Translation

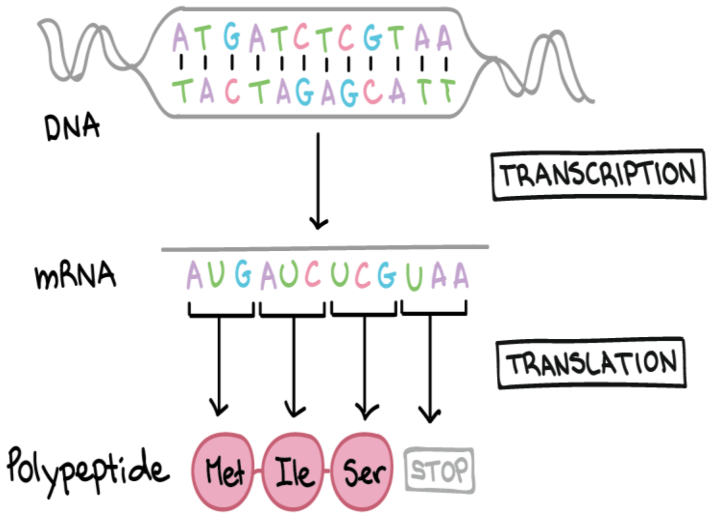


Fig.1.1.4. Transcription and Translation

Clear diagram of the core dogma, displaying the molecular sequences concerned.

2 DNA strands are sequenced with the following:

5'-ATGATCTCGTAA-3' 3'-TACTAGAGCATT-5'

Transcription over one strand of DNA produces an mRNA that coincides almost in sequence with the other strand of DNA. Whatever the biochemical difference amongst DNA and RNA, mRNA shares the DNA Ts with us. The mRNA order is to:

5'-AUGAUCUCGUAA-5'

In Translation studying mRNA nucleotides in groups of three, so every specifies that also amino acid (or provides completion of the translation with a stop signaling).Transcription is a sequence of DNA that has been translated or transcribed into a common "alphabet" of RNA. In eukaryotes, the RNA molecule needs to undergo processing to become a mature messenger RNA.

Translation is the mRNA sequence is decoded to determine a polypeptide amino acid sequence. The name translation reflects that the nucleotide sequence of the mRNA sequence has to be translated into an entirely different amino acid "language."

1.1.5 Mutations

Sometimes cells make mistakes in copying their genetic information which causes mutations. Mutations may be irrelevant, affecting the way proteins are formed and genes are expressed.

1.1.6 Common mistakes and misconceptions

- **During protein synthesisAmino acids are not made.** Most of us would suppose the production of proteins is designed to produce amino acids. Fortunately, they aren't formed in translation; they can be used as the essential elements for making proteins.
- **Mutations have neither severe nor detrimental consequences at all.** Sometimes people hear the word "mutation" in the media and analyse it as meaning that a person gets a sicknesses or deformities. Mutation is the form of genetic diversity, and even though it is dangerous to certain mutations, others are invisible to the naked eye and others are even healthy!
- **Variations of three nucleotides, rearrangements will not lead to mutations in the frameshift.** Instead, the protein quickly contributes to or extracts one or more amino acids. Insertions and deletions are not different versions of 3 nucleotides but can greatly change the amino acid sequence of the protein.

1.2 Prokaryotic and Eukaryotic

1.2.1 Eukaryote

Any single cell or organism has a well identified nucleus. This has a nuclear membrane surrounding the nucleus, in which there are well-defined chromosomes. The eukaryotic cells also contain organelles, including mitochondria (cellular energy exchangers), a Golgi device (secretory device), an endoplasmic reticulum (cellular membrane structure), and lysosomes (a digestive device in many cell types). Specific cases for this are the complete lack of mitochondria and a nucleus in red blood cells, and the unavailability of mitochondria in the genu Monocercomonoids.

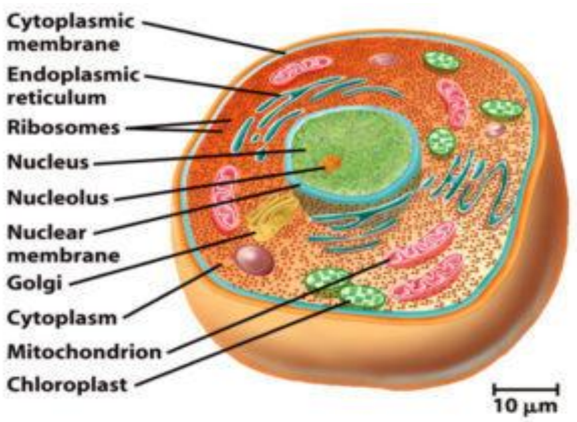


Fig.1.2.1. Eukaryotic Cell

1.2.2 Prokaryotes

This is a single-celled body, containing the oldest and most primitive type of life on earth. Prokaryotes constitute of bacteria and Achaeans, as set out in the Three Domain System. Many prokaryotes, for example cyanobacteria, are photosynthetic cells and are capable of photosynthesis. Most prokaryotes are highly versatile and are able to live and expand under many forms of intense environments, including hydrothermal vents, hot springs, swamps, wetlands, and human and animal guts (Helicobacter pylori). Prokaryotic or single-cell bacteria can be found nearly anywhere and form part of the microbiota of man. We live on your skin, your organs, your surroundings and physical things.

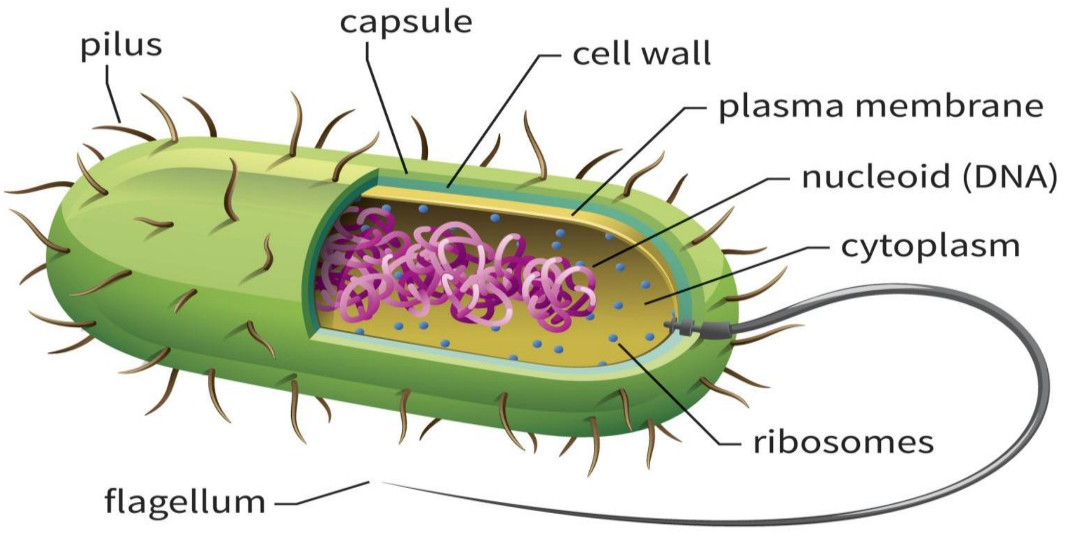


Fig. 1.2.2. Prokaryotic Bacteria

1.3 Promoter

A site at which RNA polymerase and transcription factors bind to initiate transcription of specific genes to mRNA is in a DNA molecule.

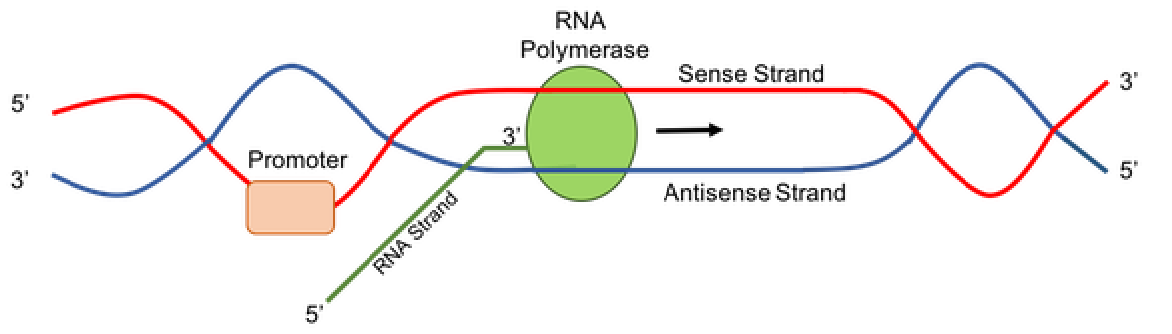


Fig. 1.3. Promoter

This same enzyme that catalyzes RNA, which is RNA polymerase, needs to be bound to the DNA just next to a transcription gene to occur. Promoters have special Dna fragments including such return elements that hand off RNA polymerase and proteins also including Rna transcription factors to a secure starting binding site. Such transcription factors bind similar sequences of reciprocal nucleotide activators or repressors bound to different promoters and control the expression of the genes.

1.3.1 Promoter sequences

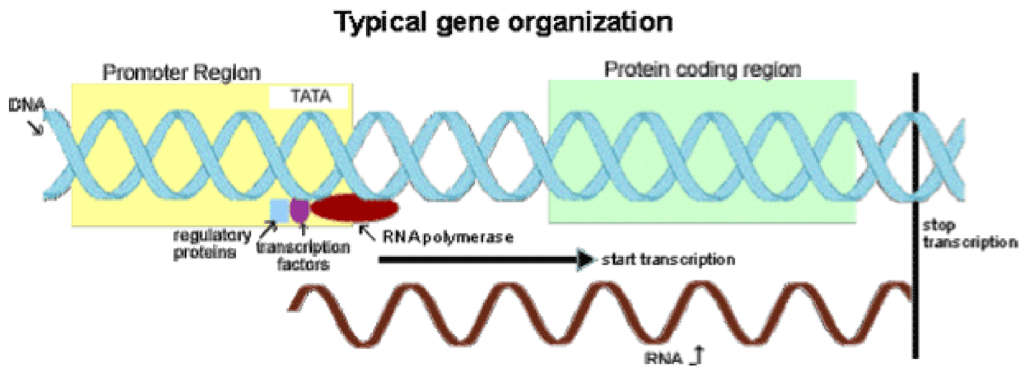


Fig. 1.3.1. Promoter Sequence

Those are DNA sequences, which determine where RNA polymerase transcription of a gene starts. Such sequences are usually located directly upward or at the conclusion of the introduction transcription site at 5. Ribonucleic acid (RNA) and the essential transcription factors attach to the promoter region and begin sequencing. This defines the path of transcription and decides which DNA sequence is to be transcribed; this strand is known as the sense strand.

1.3.2 Identification of promoter location

Also specifically assigned to the gene to classify promoters, positions in the promoter are assigned position in relation to the transcriptional starting site where DNA transcription usually starts for a particular gene (i.e. upstream viewpoints are negative numbers collecting back from -1, for example -100 is upstream location 100 base pairs).

1.3.3 Detection of promoter

A variety of different of algorithms were designed to facilitate the detection of promoters in the human genome, and promotor prediction is a common characteristic of many gene prediction methods. Before the coding sequence -35 and -10, there is a Promoter area. The nearer the promoter zone(region) is the more frequently the gene is transcribed to the coding sequence. There is no clear hierarchy for the promoter areas (regions), as there are for coding sequence.

1.4 CpG islands in promoters

In about 70 per cent of promoters near the transcription start site of a gene (proximal promoters) own an island CpG in humans. CpG islands are roughly 200 to 2000 base pairs long, have a C: G base pair density of > 50 percent, and have DNA areas where a cytosine nucleotide is preceded by a guanine nucleotide that is repeated in the sequential order of bases at its base. 5'→ 3 'way.

1.5 Diseases caused due to mutation

Few Events are associated with changes in the promoter or transcription factor of other genetic disorders.

· Examples include:

- Asthma
- Beta thalassemia
- Syndrome of Rubinstein-Taybi

1.6 Core promoter

1. This is portion of the proximal promoter that comprises the transcription starting sites.

2. It is the minimum DNA sequence stretch that suffices to guide correct transcription initiation. Usually 60 to 120 base pairs are an appropriate range of a core promoter long.

3. Short sequences involving beginning transcription sites (TSSs).

4. Comprises a binding domain to holoenzymes for RNA polymerase (RNA polymerase I, RNA polymerase II, or RNA polymerase III).

5. A broad network of regulatory aspects that influence to initiating RNA polymerase transcription consequently target any particular gene 's core promoter.

6. The core promoter includes the site(s) to start transcriptions (TSS).

7. The proximal promoter further forms part of the core promoter portion that has been upstream of something like the TSS.

8. The central promoter from the TSS is roughly -34 bp upstream.

1.7 Proximal promoters

A promotor DNA region that contains the transcription start sites that really are adjacent to the start sites is considered a **proximal promoter**.

1.8 Gene transcriptions

DNA is an epigenome double helix, accompanied by the related nucleotides. An enzyme, actually an RNA polymerase, is chemically associated by biochemical signals on account of either of the strands of this double helix. When phosphorylated, the polymerase begins using the template strand to catalyze the forming of ribonucleic acid. As that catalysis having over than one starting nucleotide and also more than one ending nucleotide (a stop site) alongside the DNA, almost every sequence of catalyzed nucleotides that ultimately create a certain RNA is part of a gene. The catalysis from each RNA shows a template DNA transcription, especially the gene transcription.

1.9 Gene Expression

Proteins dictate cell function and proteins are coded by Genes. Even in a single cell there are thousands of genes expressed that dictate what the cell will do. However, each step in the discharge of protein information from DNA to RNA gives the cell with a potential control point to self-adjust its functions by changing the quantity and type of protein it produces.

At a given time the sum of a particular protein in a cell represents the balance between that protein's synthetic and degrading biochemical pathways. Remember that protein production begins at transcription (DNA to RNA), and continues on the synthetic side of this balance with translation (RNA to protein). Consequently, regulation of these processes plays a crucial role in understanding which proteins are found in a cell and in what quantities. Plus the way a cell produces its RNA transcripts and newly formed proteins also impacts the protein levels significantly.

1.10 K-mers

This are found in a biological longitude sequence. In the sense of gene expression mainly used.

1.11 Successful Examples of genetic modification

1.11.1 Mouse-ear cress

We understand a number concerning the chromosomes. This delicate plant allows one to consider the genetic nature of other plant traits related to: heat, low nitrogen needs, low temperatures as well as freezing, high temperatures, light (e.g. shade tolerance), tolerance to UV radiation, photosynthetic behavior, low pH and aluminum in the soil, high pH, growth rate, flowering cycle, greening during maturation, plant architecture, fertility I After they have been found in the mouse-ear cress, the genes for these characteristics can be used to reconfigure cultivated species.

1.11.2 Western corn rootworm, European corn borer

GM species which occupy fields are routinely used to facilitate production. This tends to add harmful insect-resistant species like that of the European corn borer and the rootworm of western maize. The last one was exotic from America where two major rootworm species caused damage that would dig into stems and cause the crops to bend. Though no opposite scientists have discovered in Slovenia, crop rotation answers that question. The same applies to cotton, where different species spray less than 80 per cent.

1.11.3Bananas

In certain countries across the globe bananas are the primary source of nutrient. The rise of new diseases is compromising their development as reported in Uganda reports. Luckily, a genetic modification has been used by Ugandan scientists to inject a pepper gene into bananas which protects it.

1.11.4 Onions that do not make you cry

In 2008, a New Zealand development team headed by Colin Eady developed an onion that won't make you weep when it's sliced. In the early phases, adding a single gene that regulate the mechanism of the onion enzyme that tends to make your eyes water has achieved success in accomplishing two things: first, onions no longer flood your eyes, and second, they also have substances that are much more valuable to sulfur than regular onions.

CHAPTER 2

DEEP CONVOLUTIONAL DIVERGENCE ENCODING

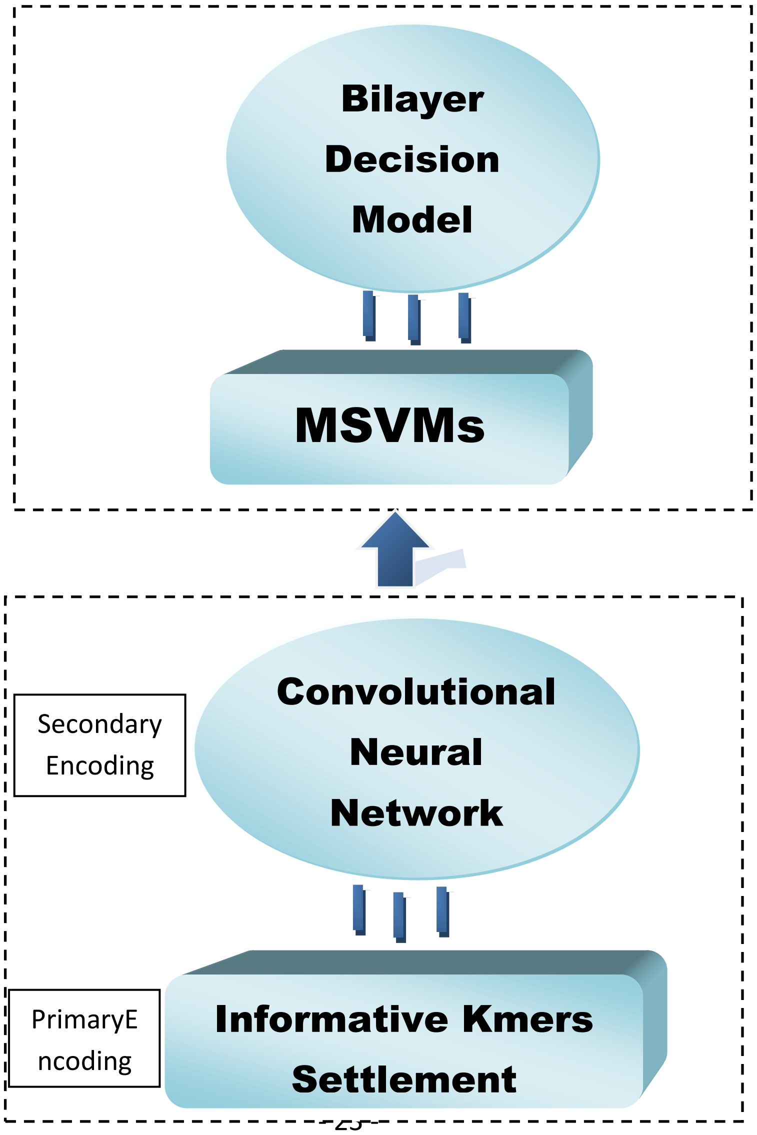
In genes, promoter is the region where transcription start sites (TSSs) are present. It helps in regulating gene biological activity and also in initiating gene transcription. To order to better understand the regulation of the human gene it is important to define the promoters. Several bioinformatics approaches were used to classify the promoter region in humans, and were predicted with less computational complexity[4][8].

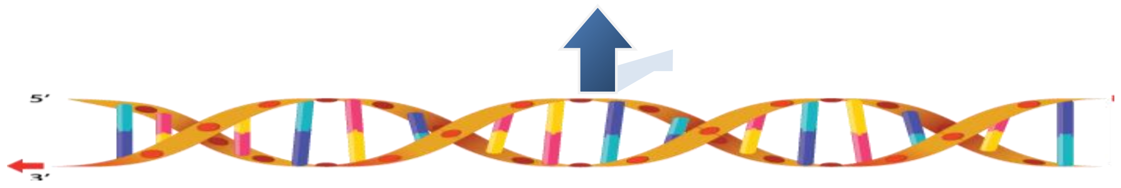
However it is hard to describe the recognition of human promoters[7]. Today, encoding the original DNA sequences and extracting the most useful features for defining promoters are a fascinating task and are being studied by all. Context, signal and structure are essentially the three main regions for defining the promoter regions. Nonetheless, the use of extraction of a single element can not yield satisfactory results. The structural function of human DNA can be used to classify both proximal and core promoter areas.However, it has both noisy promoter profile and its sequence-dependent limitations. The high content of CpG exists only in 70 per cent of human promoters as a signal function [1]. CG islands are high frequency locations of CpG areas. The background function is based on kmers and this approach is favored by all, because it is both biologically and statistically relevant. Compared to the signal and formation features; context function will give us more knowledge to implement machine learning methods and create an efficient model.Some of the frequency based searching techniques include :

- **SeqGL –** It is an algorithm used to identify sequence signals of the Transcription Factors. Using lasso regularization it is used for training on a discriminative pattern. It displays higher accuracy and sensitivity for signal detection in DNA sequence. Its limitation includes not identifying on the new or unseen data[1].
- **LS –GKM –** It is an algorithm that has been used for gapped kmers and can be implemented on large scale data. It's drawback involves lacking sequence location specificities[11].

It was concluded, therefore, that the DCDE (Deep Convolutional Divergence Encoding) approach is an efficient way to classify the region of human promoters while preserving the positional significance of genes[1]. DCDE includes both methods of statistical divergence and of deep learning.The uncertainty can be analyzed using statistical probabilities from information theory, and could be estimated by entropy of information. Entropy is the uncertainty measure within a set of examples. Therefore the stochastic features could be described in numerical form. Kullback-Leibler (KL) divergence can be applied for estimating conditional entropy and in extracting kmer features. The only downside of using this approach was its ignoring the specificity of the position[7].So we should use insightful kmers settlement as our first phase of DCDE to encrypt both promoters and non-promoters while preserving their spatial specificity.This method involves a series of SD methods as well as optimization of feature extraction. The SD methods includes J divergence, JensenShannon(JS) divergence and KullbackLeibler (KL) divergence.

Convolution Neural Network (CNN) is commonly used for pattern recognition, classification of images, analysis of medical images, etc. It has the ability to learn automatic functionality, i.e. learn by itself. CNN is also used in the gene research and biological issues to obtain successful outcomes[2]. In the second phase of DCDE we will combine multiple CNNs for strong encoding of insightful kmers of positional character recognition matrices in order to obtain more discriminative encoding functions [3]. After the complex deeply encoded features from CNN, we will be using multiple SVM (Support Vector Machines) for different non promoter datasets. SVM performs better as compared to other machine learning methods like Markov model, linear and quadratic discriminant analysis method, and relevance vector machine. As other methods could not handle large amount of high dimensional datasets therefore, MSVM is being used for promoter recognition. At last, to integrate multiple SVMs we will design a Bilayer Decision model for human promoter recognition.





Human Genomic Sequence

Fig. 2 – Framework of DCDE

There are two types of promoters present in the human gene namely proximal and core promoters. Nearly 250bps upper part of TSS is present at the proximal promoter. The central promoter has a area across the TSS of approximately [-50,+50] bps. In our research the genomic region is around [-200,+50] bps of TSS[7]. In human promoter region mostly we consider both the introns and exons in our samples.

For executing the DCDE; we will be taking 4 datasets which consists of two for training (promoters + non-promoters) and two for testing (promoters + non-promoters). The training dataset consists of 4000 samples and the testing dataset consists of 200 samples. The datasets consist of equal number of promoters and non-promoters. We had downloaded this dataset from NCBI (National Centre for Biotechnology Information).

CHAPTER 3
INFORMATIVE KMERS SETTLEMENT

In DCDE the distribution of likelihood of all potential kmers at each nucleotide location is counted and defined. The feature space size of the L-length gene sequence will be 4^k and hence the total function element comprising N-sequences will become $N*L*4^k$, resulting in very high memory consumption as L or N increases.Hence, the best way to pick the most precise and detailed kmers is to use series of SD methods which would include J divergence, JS divergence, and KL divergence. Then we can encode the original series using distribution of chance which will optimize the gap between promoters and non-promoters. As a result, interesting and insightful positional encoding matrices for kmers will be generated which would include the promoters positional specificity.

3.1Statistical Divergence :

This is used primarily to increase the difference between promoters and non-promoters. Optimize the chance densities of kmers used by J divergence, JS divergence, or KL divergence after measurement of the distance[5]. We can end up settling the most interesting and insightful kmers after optimization, and eliminate the obvious floating kmers. Let p_p € R^{4k} be the density of probability for promoters and p_{np} € R^{4k} be densities of probability on 3 types of non-promoters, such that for exons: r=1, introns: r=2 and 3'UTR: r=3; Consequently, the statistical divergence between p_{np} and p_p could be described as:

$$SD_r(p_p||p_{np}) = \sum_{i=1}^{4^k} d_i$$

3.1.1 KL Divergence :

This type of divergence is asymmetrical and is a measure of how one kind of probability distribution differs from the other. It does not satisfy the triangle inequality. It could also be defined as follows :

$$D_r(p_p||p_{np}) = \sum_{i=1}^{4^k} d\,(p_p(i)\,, p_{np}(i)) = -\sum_{i=1}^{4^k} p_p(i)\, Ln \frac{p_p(i)}{p_{np}(i)}\,;$$

3.1.2 J Divergence :

This type of divergence is symmetric and is therefore considered as symmetric estimate method[5]. It could also be defined as follows :

$$J\,D_r\,(p_p||p_{np}) = \frac{1}{2}D_r\,(p_p||p_{np}) + \frac{1}{2}D_r(p_{np}||p_p)$$
$$= \frac{1}{2}\sum_{i=1}^{4^k} d\left(p_p(i), p_{np}(i)\right) + d\,(\,p_{np}(i), p_p(i)\,)$$

3.1.3 JS Divergence :

This type of divergence is also called Informative Radius (IRad). It's finite and symmetric. It could also be defined as follows :

$$JS\,D_r\,(p_p||p_{np}) = \frac{1}{2}D_r(p_p||p^r) + \frac{1}{2}D_r(p_{np}||p^r)$$
$$= \frac{1}{2}\sum_{i=1}^{4^k}(d\left(p_p(i), p^r(i)\right) + d(p_{np}(i), p^r(i))$$

Where$p^r = \frac{1}{2}p_p + \frac{1}{2}p_{np}$.

So according to the above equations we can represent **$SD_r\,(p_p||p_{np})$** as follows **:**

KLD :$d_i = -p_p(i)\, ln \frac{p_{np}(i)}{p_p(i)}$

JD : $d_i = \frac{1}{2}(p_p(i) ln\, \frac{p_p(i)}{p_{np}(i)} + p_{np}(i) ln \frac{p_{np}(i)}{p_p(i)})$

JSD : $d_i = \frac{1}{2}(p_p(i) ln\, \frac{p_p(i)}{\frac{1}{2}\left(p_p(i)+p_{np}(i)\right)} + p_{np}(i) ln \frac{p_{np}(i)}{\frac{1}{2}\left(p_p(i)+p_{np}(i)\right)})$

3.2 Optimization :

Finally after evaluating d_i from different SD approaches we may settle the most insightful kmers and remove the raising floating k-mers. Firstly, we will sort the d_i ,I= 1,2…. 4^k in descending form so that a new vector $\mathbf{d^r} = [d_1, d_2….. d_{4^\wedge k}]^T$ is formed. For settling the most informative kmers we could solve using following optimization problem :

$$\boldsymbol{Min\ m^r \frac{\sum_{i=1}^{m^r} d_i}{\sum_{i=1}^{4^k} d_i} -} □$$

Where m^r indicates the amount of kmers settled and some reflects 0.98 threshold amount. Let G^r be the collection of insightful settled kms. We will obtain a positional encoding matrix for kmers informative i.e. $\mathbf{ik}^r = [ik_1, ik_2.... ik_{L-k+1}] € R^{mr*(L-k+1)}$which is given by following :

$$\boldsymbol{Ik_{j,t} = Fkc\ (tri_j = G_t)\ € \ R^{mr}}$$

*Where j = 1,2, ... the base index is L-k+1, and t = 1,2, ... ,M^r is G^r index, and trij at position j is each kmer.*We can describe

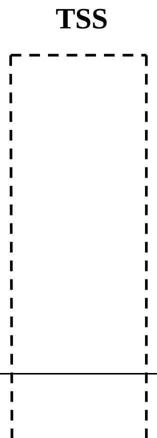
$$\boldsymbol{Fkc(tri_j = G^r) = \begin{cases} FC_t\ ,\ if tri_j = G_t \\ 0\ ,\ otherwise \end{cases}}$$

Thus, we can conclude after this step that after applying different SD methods we could encode the same sequences in different ways. The table below can show that JSD can settle the most informative kmers and the most common floating kmers can be eliminated.

TABLE 3.1. THE ELIMINATED COMMON KMERS AND SETTLED KMERS NUMBERS RESPECTIVELY

KLD	4/60	6/58	5/59
JD	5/59	5/59	4/60
JSD	10/54	14/50	9/55

After writing a python code for the first step of DCDE we could conclude upon our results obtained as :



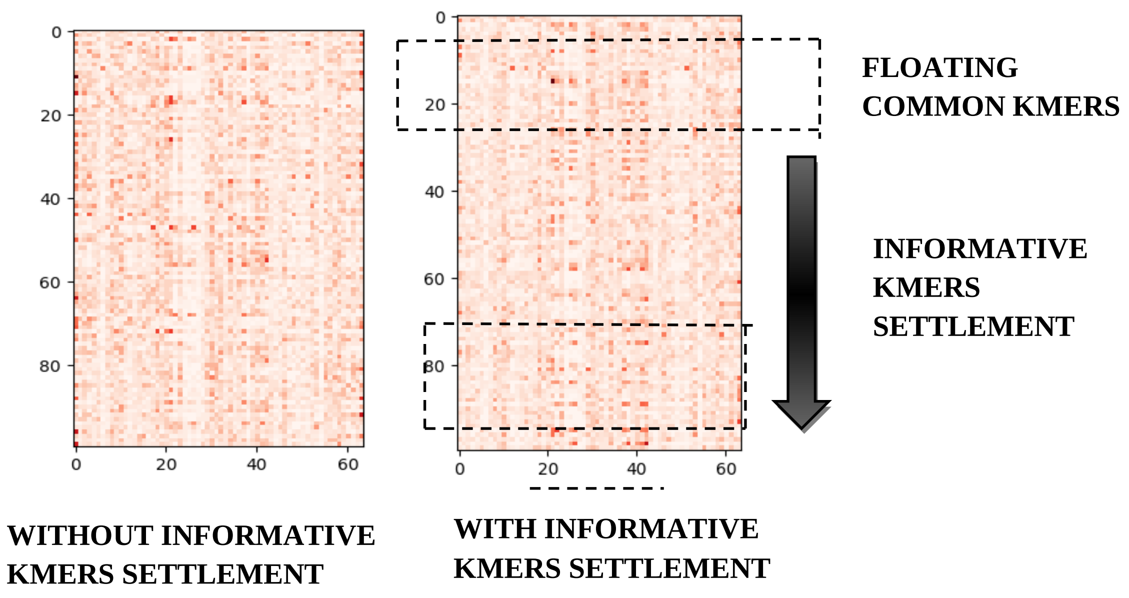


Fig.3.1. Output of Informative Kmers Settlement

The above Fig.3.1. are the heat maps showing the statistical difference between k-mer (k=3) distributions of promoters and non-promoters observed using KLD process. As we can see we can settle the most insightful k-mers and remove the typical floating k-mers while retaining the TSS which is the essential positioning function.

CHAPTER 4

CONVOLUTIONAL NEURAL NETWORK

CNN is a type of classifier used in deep learning. It's main applications include image classification, image recognition, face recognition, etc. This could also help boost estimation of the usable gene elements and examine biological problems[3]. Therefore, in the second phase of DCDE we will be applying CNN which will conduct the secondary encoding feature.

CNN's simple steps involve designing the batch of sequences to be checked and learned and passing through chain of convolutionary layers with filters typically known as kernels or feature map, then the pooling layer and the completely attached layers and then applying the Softmax function to identify an object with probabilistic values between 0 and 1. The figure below is a full flow of CNN used for secondary encoding in the genomic sequences.

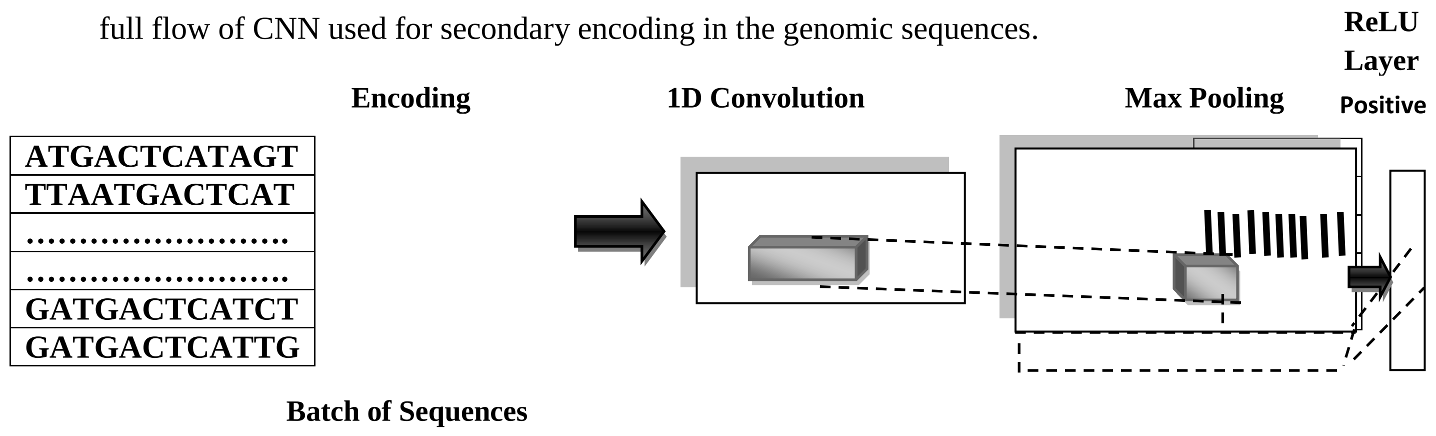


Fig. 4. Basic CNN Structure used in secondary Encoding

4.1 Convolution Layer

–That's the first layer required to fully separate sequence characteristics from the array. It uses two image matrix-like inputs, and a filter or kernel.

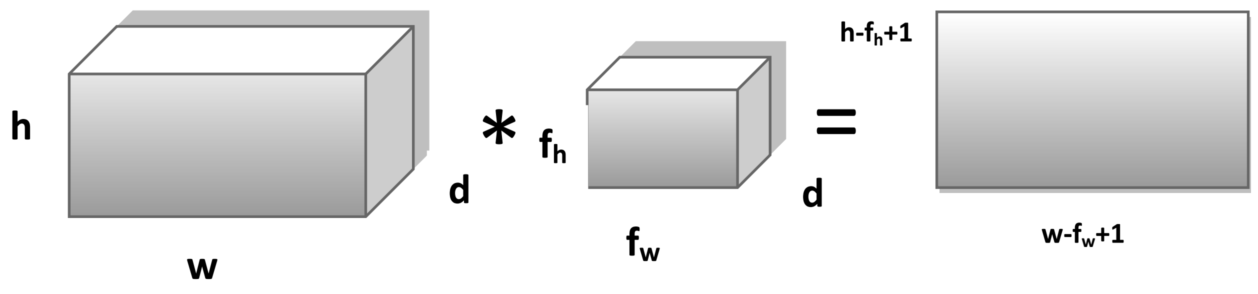


Fig. 4.1. Multiplication with kernel

The only problem with the output feature maps is that they are much sensitive to the location of the features in input. Here, ReLU(Rectified Linear Activation) feature will use a given number of kernels to learn deep features and functionality.ReLU is important if we want to put non-linearity in CNN. Since, we want our CNN to learn for non-negative linear values.Several other nonlinear functions, such as tanh or sigmoid, are available and may even be used. However most data scientists are using ReLU, since ReLU efficiency is higher.

4.2 **Max Pooling Layer –** This layer decreases the number of specifications if the data or visuals are too big. It provides a framework by outlining the existence of features in patches to downsample the feature maps. It works independently on each feature map, which therefore generates a new collection of the same number of shared feature maps. Therefore, pooling requires choosing the process of pooling and is almost like a drain. The scale of the operation / filter pooling is smaller than that of the function maps. "Spatial pooling," also known as downsampling, can minimize each map's dimensionality (input / output) but also helps to preserve valuable information.This is often used to preserve invariance of the function and to decrease next layer input capacity. Spatial Pooling is of 3 types :

1*Max Pooling*

2*Average Pooling*

3 *Sum Pooling*

Max Poolingis a form of pooling that takes the biggestfeature element from the map of the rectified function.

Average Poolingtakes the largest element from the feature map.

Sum Pooling takes the combination of all elements present in the feature map.

4.3 Fully Connected Layer -

We flatten our matrix into some kind of vector in this layer and send it into a totally linked layer, like a neural network.

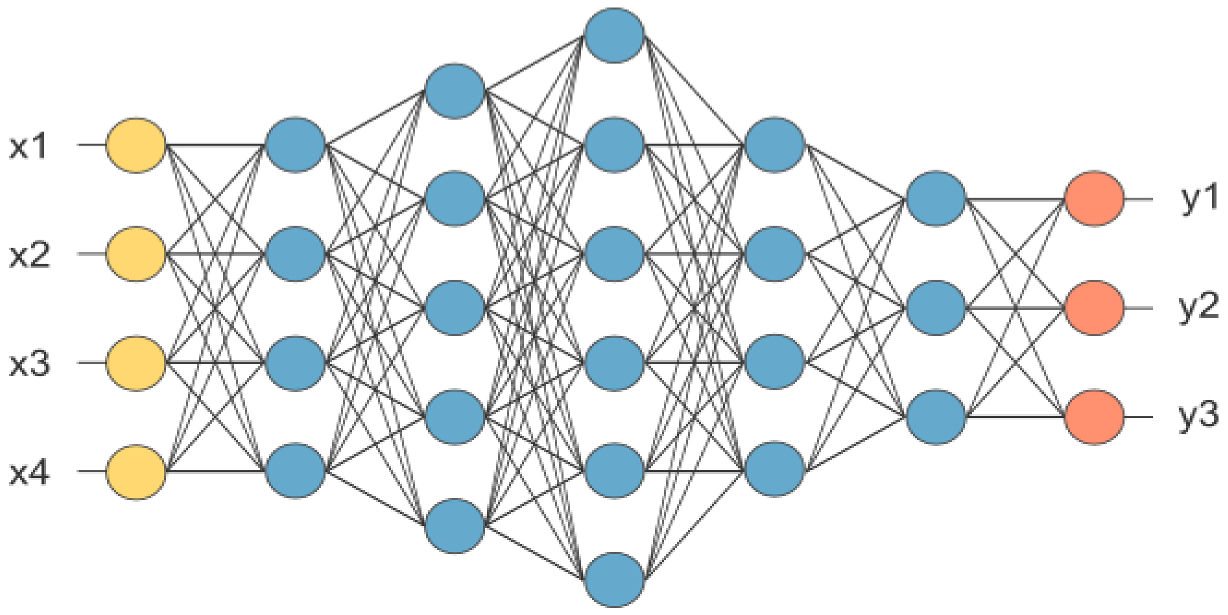


Fig. 4.3. After Pooling Layer, Flattened as Fully Connected Layer

The function map matrix is translated as a vector (x1 , x2, x3, ...) in the figure described. We've combined these features together to create a model using the fully connected layers. Ultimately, for classifying the outputs we have can use activation functions such as **softmax** or **sigmoid**.

Softmaxis a method of activation used to transform numbers into percentages that add it up to one. It outputs a matrix representing the distribution of probability of a number of possible outcomes.

We have implemented all the above steps in Python using **Keras** model. Keras is a deep learning model which uses Theano and Tensorflow libraries at it's backend for defining and training network models. The basic steps used in Keras model are :

1) Load data
2) Define the Keras model
3) Compile the Keras model
4) Fit the Keras model
5) Evaluate the Keras model
6) Tie everything together
7) Make the predictions

The first step is to define the libraries which we will be using in our code. Then load the data or image. We will then end up creating a Sequential Model, adding layers one by one.For different

SD methods we may use the defined filters, feature map size and output feature map size with the aid of the table below.

TABLE 4.1. The structure of CNNs in DCDE

CNN	Intron & Promoter	3'UTR & Promoter	Exon & Promoter
JD	128/12/64	128/8/64	128/8/64
JSD	133/5/68	128/8/64	133/5/68
KLD	128/15/64	128/15/64	128/3/50

The table above describes the three types of CNNs which correspond to the three SD methods used in DCDE. Can define the above details as;example, 64/15/128 could be read as convolution layer is using 64 filters, 15 feature map size and 128 as output feature size from fully connected layer.

The layers that are totally connected claim the **dense** level. We'll define the number of neurons in the layer as the first statement, and use the activation argument to determine the activation function. On the first two rows, we will be using the rectified linear unit activation function referred to as ReLU.

Dropout is a technique used where randomly selected neurons are ignored while training is occurring. It is used to reduce the overfitting of data.

It uses efficient libraries called as backend when compiling the Keras model, such as Theano and TensorFlow. The backend will dynamically pick the right way to practice and make decisions for running on the hardware, such as CPU or GPU.While compiling, we need to specify some additional properties, such as finding the best connection weights in our data - set to map inputs to outputs. We will need to define the failure function to be used to determine a range of weights. The optimizer was used during the training to search by different weights and any optional metrics we would most like to collect and analyze.

The **Loss function** tells "how good" the model is at making the predictions. It has its own curves and derivatives. The slope of the curve tells that how we should change the parameters to make our model more accurate. We use the cross entropy as the reason for loss as this loss is described in Keras for classification problems "**binary_crossentropy**". It's used where the target values are defined in set of {0,1}. Crossentropy evaluates a score which would elucidate the average difference between actual and predicted distributions of probabilities. So the ranking is reduced, and a complete crossentropy '0' rating is obtained.

The optimizer can be used as the accurate stochastic gradient descent algorithm known as, **"adam."** This is a common gradient descent variant, as it automatically tunes itself and yields good results.It is a method used for stochastic optimization. It is basically a collection of methods used for maximizing and minimizing an objective function whenever randomness is present.

- When the model is executed;we can train or match our model on our loaded data by calling the**fit()**. Training takes place over epochs and each epoch is divided into lots.
- **Epoch**: In the training dataset, each pass will start happening through every rows.

- **Batch**: The model must look for one or so more samples within the epoch but when weights are modified.

  In general, each epoch consists during one or much more batches depending on the defined batch size and the model is appropriate for several **epochs**. The training cycle runs through the dataset over a predetermined number of iterations and is stated using the argument epochs. We must also define the number of set of data rows inside each epoch before the model weights are modified and specify using the argument **batch size**.

  Now, using the evaluate( )function, we could also evaluate our model on our training dataset and transfer it on to the same input and output used to train the model. This would thus help create a forecast for each pair and accumulate scores and average losses.

  The **evaluate()**function displays the list containing two values. One being the model's loss upon this dataset and the second is model's accuracy. We are only interested in recording the accuracy so disregard the importance of the loss.

CHAPTER 5

SUPPORT VECTOR MACHINE

SVM is a classifier that collects information for problems with supervised and unsupervised learning and separates data by a hyperplane. It is a supervised model of learning. This hyperplane is a line in two dimensional space splitting into two sections where every class lies on either side. We have the following dataset, for example, on a plane;

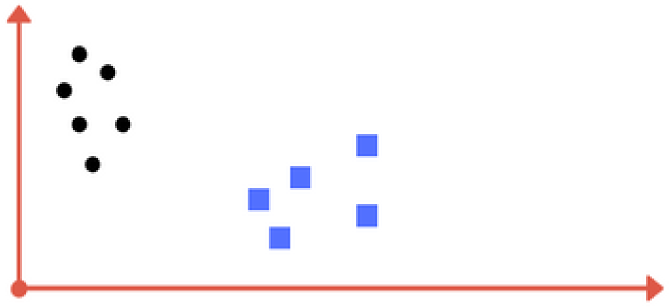


Fig. 5.1. Example 1.1

Thus, the above dataset could be separated by a hyperplane in the following manner :

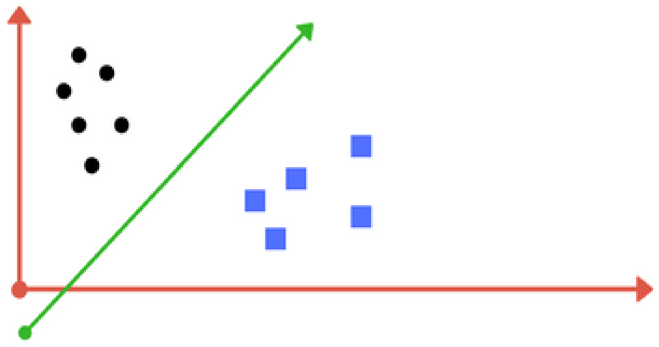


Fig.5.2. Example 1.2

Now, consider a dataset like following :

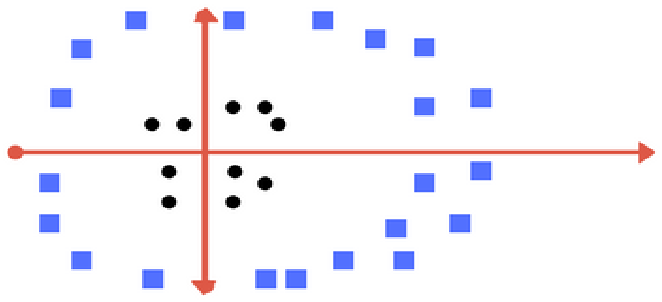


Fig. 5.3. Example 2.1

Because there's no line in x-y plane that could divide the two sections, we can implement transformation and add another one dimension called z-axis. We can presume point value in z plane, w = $x^2 + y^2$ so we can modify it as point distance from z-origin. Then a simple break is evident as:

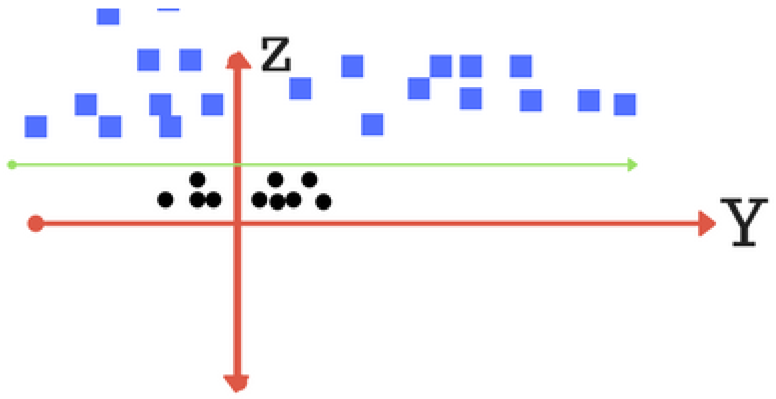


Fig.5.4. Example 2.2

When we convert this line back into the original plane, it will trace the circular boundary as seen in the illustration and such conversions are kernels.

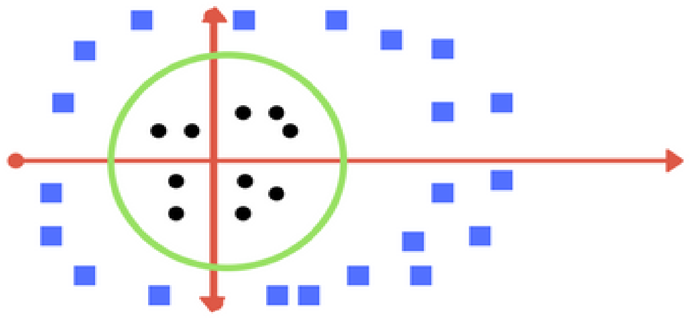


Fig.5.5. Example 3.1

In linear SVM the hyperplane learning is achieved by using linear algebra to transform the problem. The kernel plays an important part here.

In **linear kernel,** the equation for prediction of a new input using the dot product between the input (x) and each support vector (xi) can be calculated as follows:

$\mathbf{F(x) = B(0) + sum(a_i * (x, x_i))}$

This equation is used for calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B(0) and a_i , for each input must be estimated from the training dataset.

The **polynomial kernel** can be written as

$$Pr(x,x_i) = 1 + sum(x * x_i)^\wedge d$$

and **exponential can be written** as ;

$$Ka(x,x_i) = exp(-gamma * sum((x — x_i^2)).$$

5.1 Regularization

The Regularization Parameter; also attributed to as "C parameter" in the sklearn library of python, specifies the SVM optimization how often one needs to prevent misclassifying of sample of training.

The algorithm can only use a smaller-margin hyperplane for the large values of C if that hyperplane does a decent job of having all the training points properly classified[9]. Conversely, even though the hyperplane misclassifies further values, a very small value of C will lead the optimizer to search for a larger-margin separating hyperplane.

The following figures are examples of possible different parameters for the regularization. Despite the lower regularization value the left one has some misclassification and the higher value leads to result like the down one.

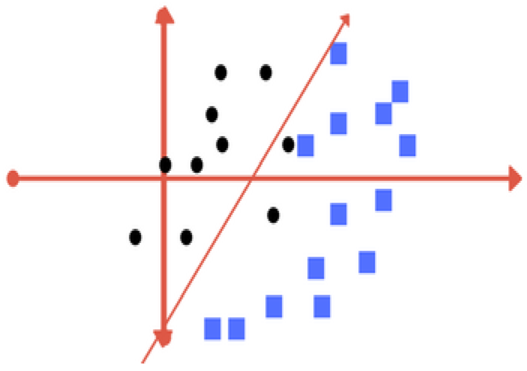


Fig.5.1.1. Example 4.1

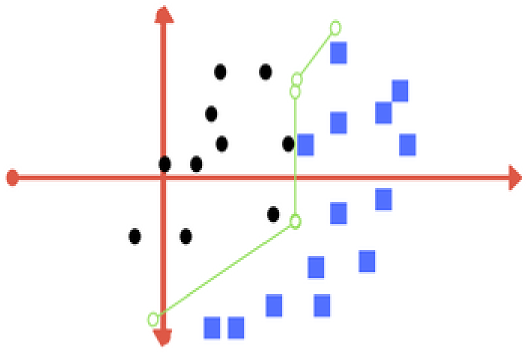


Fig.5.1.2. Example 4.2

5.2 Gamma

The gamma parameter is used to define how far a single training sample's influence could reach, with low values i.e. 'far' and high values means 'close.' In several other words, with low gamma, points for the separating line are considered far away from possible separation line in calculation.

When high gamma means in measurement the points near to potential line should beconsidered.

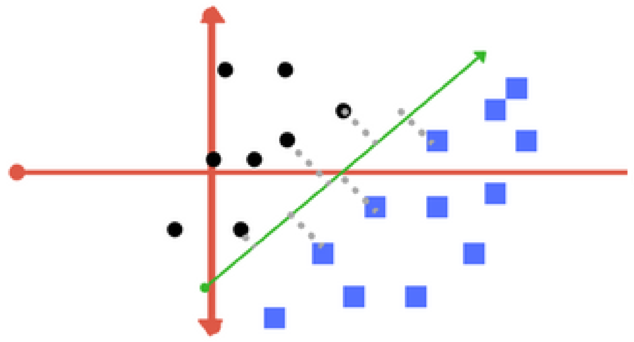


Fig.5.2.1. High Gamma

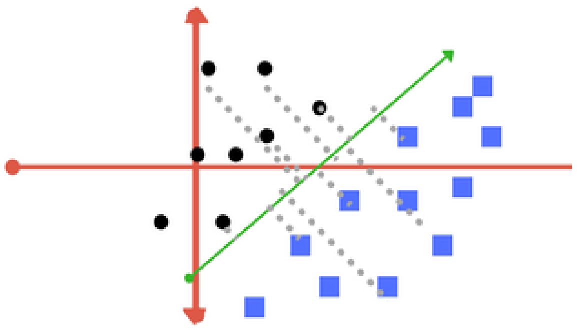


Fig.5.2.2. Low Gamma

5.3 Margin

A margin is a line dividing the nearest class points. A decent margin for all classes is one where the difference is wide. The charts below offer a positive and poor visual indication of margin. A good margin makes it possible for the points to be in their respective classes without crossing into another.

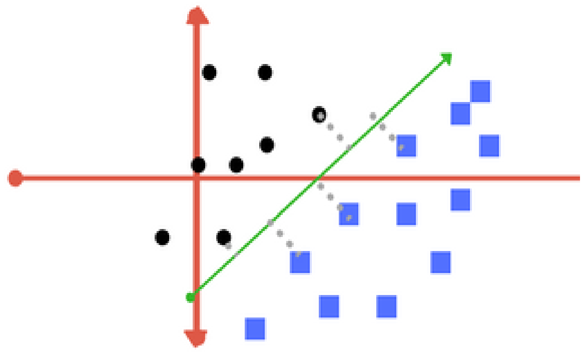


Fig .5.3.1. Good Margin

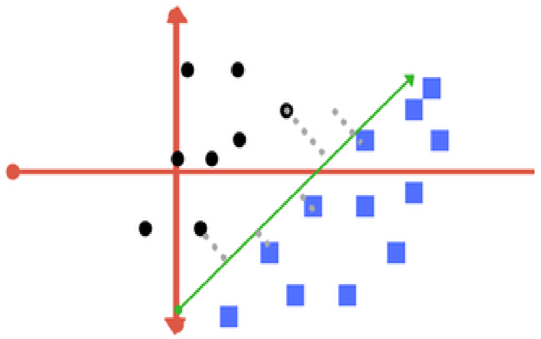


Fig.5.3.2. Bad Margin

In Bioinformatics; SVM offers excellent performance by storing vast volumes of complicated data. This has also been a very successful identification algorithm for promoters. To address the complex heavy depth encoding features resulting from CNN; three non-linear SVMs are applied to generate an effective deep learning framework for various non-promoter sets including exons, introns, and 3'UTR. Applied with the Gaussian Radial Basis Kernel Function (RBF), the highly nonlinear SVM is used to map heavy depth encoding features into high-dimensional feature space to achieve efficient classification of the promoter. The SVC is used in the code and is imported from library. It’s implementation is based on libsvm. It uses parameters **‘C’** for regularization; uses **kernel** with ‘RBF’ function; **gamma** is used with ‘auto’ or ‘par’(‘auto’ is used to indicatethat no explicit value of gamma was passed); **tol** is used as tolerance for stopping criterion and **probability** is used to make estimation of probability, this must be done before calling fit, and this approach can slow down.As a result, we will be getting Sensitivity, Specificity and Averaged Conditional Probability which could help in better evaluating the overfitting results.

Sensitivity, $S_n = \frac{TP}{TP+FN}$

Specificity , $S_p = \frac{TN}{TN+FP}$

Averaged Conditional Probability, $ACP = \frac{1}{4}(\frac{TP}{TP+FN} + \frac{TP}{TP+FP} + \frac{TN}{TN+FP} + \frac{TN}{TN+FN})$

Where, TP is the number of promoters which have been established correctly.

FN is the number of non-promoters which were not correctly identified.

TN is the number of non-promoters correctly identified.

FP is the number of promoter which was not identified correctly.

CHAPTER 6
BILAYER DECISION MODEL

To incorporate several SVMs that are implemented in DCDE's thirst phase, we must build a Bilayer Decision model that is an optimal way to find conclusions rather than majority voting algorithm. In DCDE we had first encoded the gene 'g' into positional encoding vectors of informative k-mers. In the second phase we were using Convolution Neural Network (CNN) to profoundly encode. Let $F^r(x^r)$ be the three mark outputs of three SVMs, and the three likelihood outputs created by $P^r(x^r)$.

And, by using bilayer decision model, we can aggregate the outputs from the three SVMs to decide if it is human promoter or not. And there are two phases to the bilayer judgment pattern. At first, we can forecast first, relying on the following law of decision:

$$\boldsymbol{Y_1} = \begin{cases} \boldsymbol{max\{p^r(x^r)\}}, & \boldsymbol{if\ \sum_{r=1}^{3} \frac{(f^r(x^r)+1)}{2} = 1} \\ \boldsymbol{p^r(x^r), if} & \boldsymbol{\sum_{r=1}^{3} \frac{(f^r(x^r)+1)}{2} = 1} \\ \boldsymbol{min\{p^r(x^r)\}}, & \boldsymbol{Otherwise} \end{cases}$$

And the final prediction can be obtained by using majority probability rule :

$$\boldsymbol{Y} = \begin{cases} \boldsymbol{+1, if\ y_1 \geq \lambda} \\ \boldsymbol{-1, Otherwise} \end{cases}$$

Where, Y is the final expected value for the 'g' gene, and the value for λfalls within the scale of $0<\lambda<1$. Assume the level is 0.5. So, we should achieve our end result that if Y = +1 therefore 'g' is a promoter otherwise it is a non-promoter.

CHAPTER 7
INTRODUCTION TO SPECTROGRAMS

Color spectrograms of biomolecular sequences is used to provide knowledge of the local nature of segments of DNA as analytical techniques. These spectrograms offer a synchronous outlook of the specific nucleotide sequence size, as well as the local nucleotide information obtained by spectrogram color[12]. They are useful not only for the identification of genes and other regions with confirmed biological importance, but also for the exploration with potentially important regions that are currently unknown, characterized by distinct visual differences in the spectrogram that are not readily observable by the study of character strings.
Therefore, after introducing DCDE we suggested our own theory of classifying promoters and non-promoters with aid of spectrograms. As we could see how useful spectrograms can be for visualizing DNA sequences and thus, they could also be useful in classifying promoters.

The key explanation why the field of digital signal processing did not have any effect on biomolecular sequence analysis was that the former refers to numerical sequences, while the latter refers to character strings. Therefore, we noted that if we allocate proper numerical values to each character, digital signal processing of biomolecular sequences offers a range of novel and useful tools[14]. After conversion of sequences to numerical values we will be able to plot the spectrograms and further classify the DNA sequences as promoters and non – promoters.We can achieve conversion of DNA sequences to spectrogram by using software like MATLAB.
Shown below is the black diagram of promoter identification with help ofspectrograms:

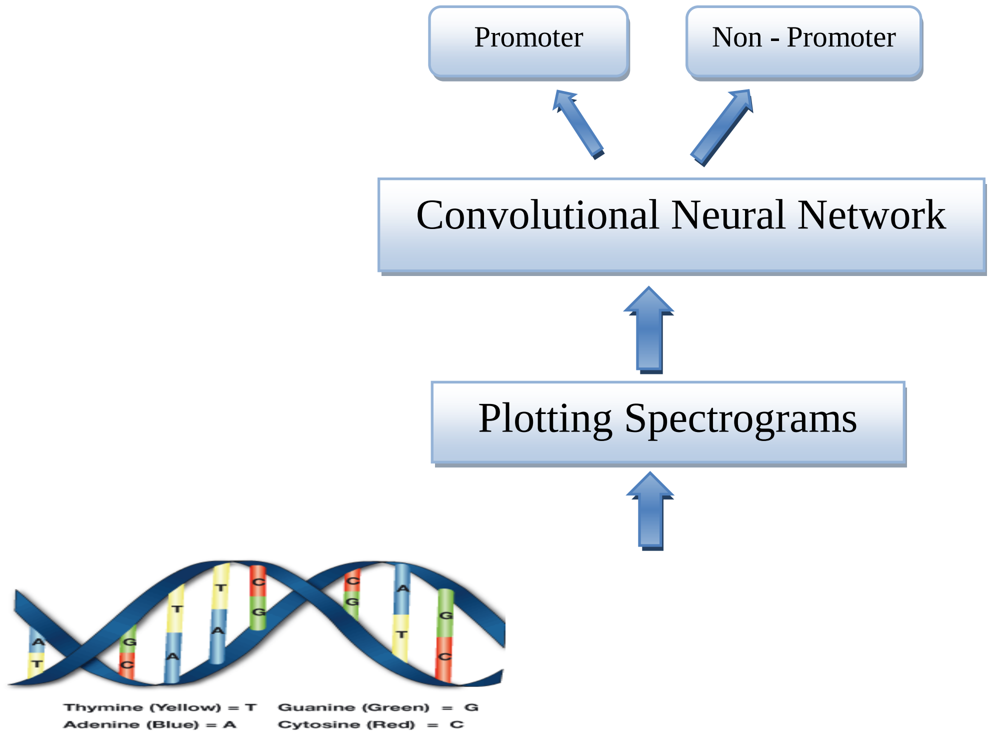


Fig 7.1 Framework for classifying promoters and non – promoters using Biospectrals

7.1 Spectrograms

A spectrogram may be produced by an optical spectrometer, a band-pass filter bank, Fourier transform, or a spectral transform.[12] Usually, a spectrogram is depicted as a heat map, i.e. a picture with the intensity displayed by varying color or light intensity. A standard format is a plot with two geometrical dimensions: one direction represents time, and the other source frequency; a third effect is defined by the strength or color of each position in the picture, demonstrating the amplification of a specific frequency at a point in time.

7.2 Short Time Fourier Transform

Short-time Fourier transform (STFT) is a series of Windowed Signal Fourier transforms[13]. STFT provides time-located frequency information for situations where frequency components of a signal differ over time, while standard Fourier transformation provides average frequency information over the entire time interval of the signal.

The STFT pair is given by ;

$$X_{STFT}\,[m,n] = \sum_{k=0}^{L-1} x[k] g[k-m]\, e^{-j2\pi nk/L}$$

$$X[k] = \sum_m \sum_n X_{STFT}[m,n] g[k\text{-}m] e^{-j2\pi nk/L}$$

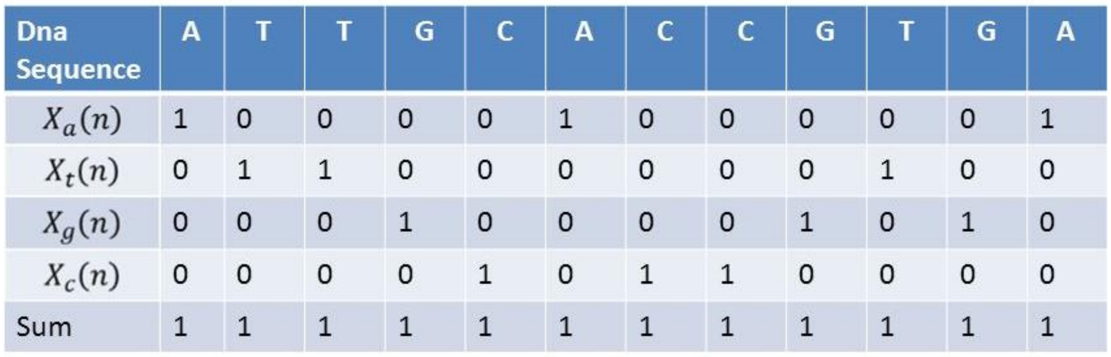
Where, x[k] denotes a signal and g[k] denotes a window function for L-point.

The x[k] STFT can be interpreted as the x[k] g[k – m] Fourier transform of the product.

CHAPTER 8
CONVERSION OF DNA SEQUENCES TO SPECTROGRAMS

Lots of signals and processes are continuous in nature. Genomic material, however, comes in the form of discrete sequences. Numerical sequences can represent DNA (deoxyribonucleic acid), and both molecules, and proteins. Digital signal processing (DSP) has evolved into numerical sequence processing[13]. Thus, it offers various effective and productive methods that can be used for genomic data analysis. To apply the DSP techniques, numerical sequences will represent the DNA. For that, there are two options:

• Establish four binary segments, one for each character (base), which determines that a character is occurring at a specific position (1) or not (0). Those are classified as Sequences of Indicators.



Dna Sequence	A	T	T	G	C	A	C	C	G	T	G	A
$X_a(n)$	1	0	0	0	0	1	0	0	0	0	0	1
$X_t(n)$	0	1	1	0	0	0	0	0	0	1	0	0
$X_g(n)$	0	0	0	1	0	0	0	0	1	0	1	0
$X_c(n)$	0	0	0	0	1	0	1	1	0	0	0	0
Sum	1	1	1	1	1	1	1	1	1	1	1	1

Fig 8.1 Binary sequence

• Assign the four characters A, T, G , and C to real - valued or complex – valued symbolic numbers; In this way a precise dimension structure is obtained which represents the rest of the series of characters[15].

STFT provides a localized measure of a long sequence 's frequency output. There are two steps to this:

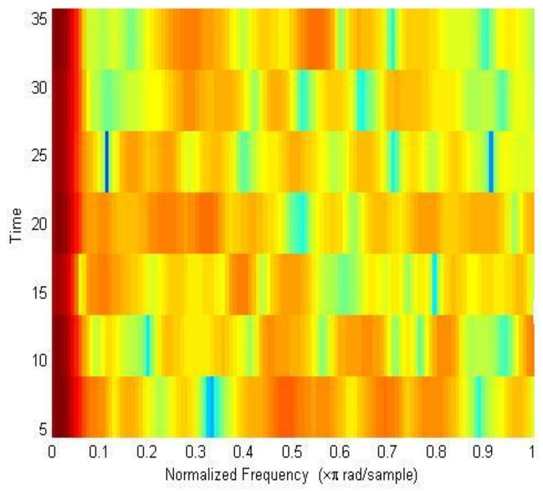
• To divide it into short sections, use a long sequence sliding window to add a sliding window to the long sequence.

• Take the DFT for each segment and take the DFT for each segment.The individual DFTs form the form segments of the STFT matrix columns. A map is considered a spectrogram of the magnitude of the STFT values[13].

For graphical rendering a color (T = red;G = yellow; A = blue; C = green) is assigned to each transformed sequence. The colors are superimposed upon the image for each base. The brightness corresponds to the magnitude (signal intensity) at a given spot on the spectrogram, and the color corresponds to the dominant base at that frequency / period. If no single base predominates, the relative magnitudes are used to measure an intermediate color. The greater the redness the higher the level of energy. We will perform this in MATLAB using following steps :

- Store each sequence is separate files in format of '.dat'.
- Read the '.dat' file and then generate numeric code for text data.
- Plot spectrogram of converted numeric sequence using **s = spectrogram(x)** syntax.
- Similarly, make the completetraining dataset consisting of 4000 sequences (2000 promoters + 2000 non – promoters) and testing dataset consisting of 200 sequences (100 promoters + 100 non – promoters).
- We can analyze that some spectrograms have very high level of energy whose sequences contain large number of 'T' characters. While some have very low energy level which contain large number of 'A' characters. Therefore, the colors assigned to each variable are true.

Examples of the spectrogram plotted using DNA sequences is shown below :



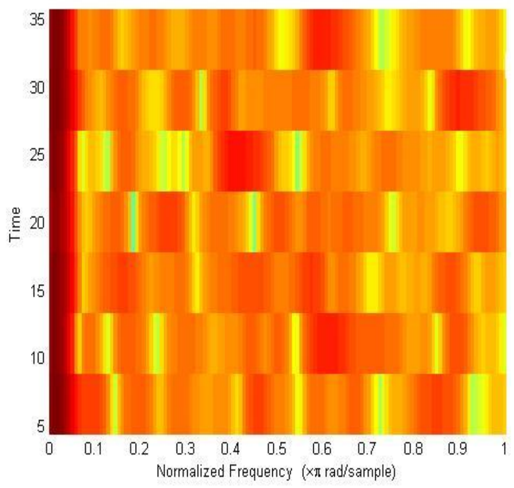


Fig 8.2 Spectrograms of a DNA sequence

CHAPTER 9
PROMOTER IDENTIFICATION USING CNN

The Convolution Neural Network (CNN) is a class of DeepLearning Networks. CNNs mark an important advance in identification of pictures. They are most widely used to examine visual imagery, and also function in image classification behind the scenes. Classification of images is the process of input and output of a class, or the probability that the input is a class. A CNN blends learned features with input data and makes use of convolution 2D layers. That means this kind of network is suitable for 2D image processing. CNNs currently use very little preprocessing as opposed to other image classification algorithms. A CNN operates by extracting images from features. That eliminates the need for extraction of the manual feature. No training on the features! They 're learned while a set of images trains on the network. This makes deep learning models extremely precise for tasks with computer vision. CNNs learn to detect features through tens or hundreds of secret layers. Each layer increases the complexity of the features learnt.
In CNN following are the basic steps needed:

1) **CONVOLUTION**

   A convolution is a combination of two functions, which proves how one function modifies another's form.The images are converted and seen in form of 1's and 0's. It consists of **feature detector** also called as ***kernel*** or filter. It convolutes the input image with the feature detector and gives **feature map** as output also called Convolved Feature. At this step we had reduced the size of image and found important features of our image.
   To increase the non – linearity of image we use **Rectifier** function namely ReLU.

2) **MAX POOLING**

   It is used to introduce **spatial invariance** in our image. From the feature map obtained in above step we apply Max Pooling to make our **Pooled Feature Map.** This step is necessary to get rid of unimportant features i.e. preserve the most important features. It also helps preventing **overfitting** of data.

3) **FLATTENING**

   In this step we convert the values taken in Pooled Feature Map and put them along one column to get huge vector of inputs.This flattened layer will be our input layer for Artificial Neural Network.

4) **FULL CONNECTION**

In this step we add our Convolutional Neural Network to our Artificial Neural Network. Fully connected layers are also called as **hidden layers**. The main purpose of ANN is to combine our features into more attributes that predict the classes even better. After passing through all these layers an output is predicted and a **loss** is calculated. Then it is back propagated so that the feature detectors and weights are adjusted. And this keeps on going.

We could summarize all our steps with help of following figure –

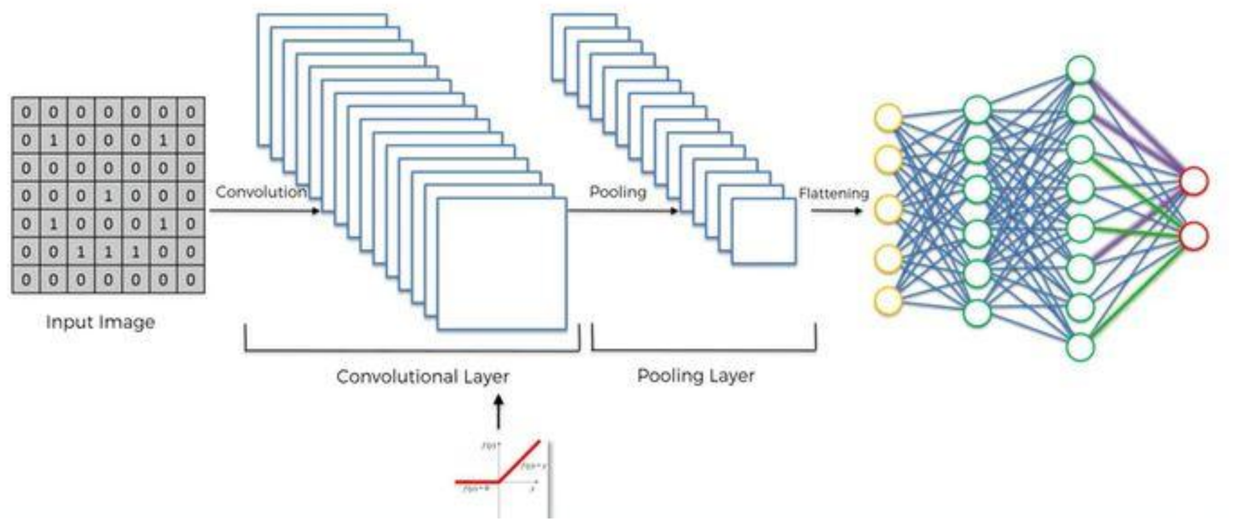


Fig 9.1 Working of CNN

Thus, to classify our promoters and non – promoters we will use CNN . We will be implementing CNN using Python. Given below is a step by step process to build CNN.

1. Install **Tensorflow** and **Keras**.

   Tensorflow is a symbolic math libraryused for machine learning applications such as neural networks.

   Keras is designed to enable fast experimentation with deep neural networks and it focuses on being user-friendly, modular, and extensible.
2. From keras import ‘Sequential’ model. Also import layers like ‘Convolution2D’, ‘MaxPooling2D’ , ‘Flatten’ and ‘Dense’.
3. Initialize the CNN by creating object of Sequential( ).
4. Add Convolution layer using Convolution2D( ) by choosing number of filters = 16, number of rows of each filter = 3 and number of columns of each feature detector = 3.

Choose activation function ‘relu’. In our program we had used 3 convolution and pooling layers to increase our accuracy.

5. Add Pooling layer using Pooling2D( ) by choosing pool size as 2*2.
6. Add Flattening layer using Flatten( ).
7. Add Fully Connected layer using Dense( ) by choosing number of hidden nodes = 128 and activate using ‘relu’ and then apply second fully connected layer by activating it using ‘sigmoid’ and output node = 1.
8. Compile the CNN model choosing optimizer as ‘adam’ and loss as ‘binary_crossentropy’ and metrics as ‘accuracy’.
9. Fit the CNN model built to our image dataset using **image augmentation** to prevent overfitting of dataset. Using ImageDataGenerator we can apply image augmentation.
10. Using train_datagen we will create this training set composed of all these augmented images extracted from our ImageDataGenerator, and also create our test set using test_datagen that will be used to evaluate the model performance.
11. Fit the model on training set using steps per epochs = 1, epochs = 50 and validation steps = 40.

After implementing our CNN we had obtained maximum **training accuracy of 71.88%** and **testing accuracy of 50.08% .**

CHAPTER 10
COMPARISON OF MODELS

In the first part of report we had implemented DCDE (Deep Convolution Divergence Encoding Method) which involves series of four steps. The results obtained from each step is as follows :

1. **Informative Kmers Settlement**–Extracted the most important features and saved in files 'x_train.npy', 'x_test.npy', 'y_train.npy' and 'y_test.npy'.
2. **CNN** – Classified Promoters and Non – Promoters with accuracy of 57.5% and further saved the important features in 'FC_train_feature.npy' and 'FC_test_feature.npy' .
3. **Multiple SVM** – Applied SVM on most important features and achieved accuracy of 66.50%, sensitivity of 67.74% , specificity of 65.42% and ACP of 66.54% and saved the predicted and probability in 'pred_testlabel.npy' and 'proba_testlabel.npy' respectively.
4. **Bilayer Decision Model**–Applying it we achieved **accuracy of 88.27%**, sensitivity of 80.02%, specificity of 77.51% and ACP of 78.45%.

Thus, we could see how after each step the model increased our accuracy of classifying the Promoters and Non – Promoters.

In second part of our report we had transformed the dataset into .dat file type and plotted **spectrogram** for each DNA sequence. Now our dataset consisted of 4000 training images (2000 Promoters + 2000 Non – Promoters) and 200 testing images (100 Promoters + 100 Non – Promoters). After applying CNN to our images we could achieve **training accuracy of 71.88%** and **testing accuracy of 50.08% .** Although, we could see that we haven’t achieved a higher value of accuracy but it is more than our previous model. Therefore, we got a new direction to classify Promoter and Non – Promoter.

Table 10.1 Comparison of models

MODEL	ACCURACY UPTO CNN
DCDE	57.5%
BIOSPECTRAL CLASSIFICATION	60.98%

CONCLUSION AND FUTURE WORK

In this project report we have specified the human promoters and non-promoters on the basis of dataset containing of promoters and non-promoters samples. At first, we had implemented a research paper in which explored Deep Convolutional Divergence Encoding method which also takes in account the positional specificity of the genes. DCDE's first phase was Informative K-mers Settlement in which we had done primary encoding and created a positional encoded matrices that may well encoded both promoters and non-promoters preserving the uniqueness of the position. We had one of the most insightful k-mers settled and the common floating k-mers removed. In phase two of DCDE we had used Convolutional Neural Network (CNN) which performs the secondary encoding of the matrix we had formed in step 1. Also we could find the discriminativekmer features with the help of CNN. Then, in the third step of DCDE we had applied multiple SVMs. We would get three label outputs by applying three SVM. In the last step of DCDE we had used Bilayer Decision Model which would integrate the outputs from three SVMs and would use decision and majority probability rule to finally predict whether the gene is a promoter or a non-promoter. After applying all steps we could achieve a good accuracy of 88.27% .

Although DCDE is a very efficient method for promoter recognition yet, it has some limitations like time complexity. Thus, after implementing and understanding the behavior of promoters we thought of performing the Spectral Analysis. So, we converted the DNA sequences into numeric form and plotted their spectrograms. After making a dataset of 4000spectrogram images we had trained our model using CNN and used it to classify the promoters and non-promoters. Till now, we had attained training accuracy of 71.88% and testing accuracy of 50.08%. Although, our results are not satisfactory but we got to work on a new direction. In future work, we could use more methods to increase our accuracy and to successfully classify promoters and non – promoters.

REFERENCES

1. Wenxuan Xu, Lin Zhu, and De – Shuang Huang, "*DCDE : An Efficient Deep Convolutional Divergence Encoding Method for Human Promoter Recognition*" ,IEEE Transaction on nanobioscience, Vol. 18, No. 2, April 2019.
2. W. Wang, J. Shen, "*Deep visual attention prediction*", IEEE Trans. Image Process., vol. 27, no.1, pp. 38-49, Jan. 2018.
3. Victor Solovyev , Ramzan umarov, "*Prediction of Prokaryotic and Eukaryotic Promoters using Convolutional Deep Learning Neural Networks*" , February 3,2017.
4. S. Mitra and L. Narlikar, "*No promoter left behind (NPLB): Learn de novo promoter architectures from genome – wide transcription start sites*", Bioinformatics, vol. 32, no. 5, pp. 779-781, Mar. 2016.
5. R. Singh, J. Lanchantin, G. Robins, and Y. Qi, "*DeepChrome: Deep Learning for predicting gene expression from histone modifications*", Bioinformatics, vol. 32, no. 17, 2016.
6. Tung Hoang, Changchuan Yin, Hui Zheng, Chenglong Yu, Rong Lucy He, Stephen S.T. Yau, *"A new method to cluster DNA sequences using Fourier power spectrum",* Elsevier, vol. 372, pp. 135-145, 7 May 2015.
7. J. Zeng, X.-Y. Zhao, X-Q. Cao, and H. Yan, "*SCS : Signal, context, and structure features for genome – wide human promoter recognition*", IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 7, no. 3, pp 550-562, Jul. 2010.
8. D. S. Huang and J. X. Du, "*A Constructive hybrid structure optimization methodology for radial basis probabilistic neural networks*", IEEE Trans. Neural Network, vol. 19, vol. 12, pp. 2099-2115, Dec. 2008.
9. V.B. Bajic, A.Chong, S.H. Seah, and V. Brusic, "*An intelligent system for vertebrate promoter regnition*", IEEE Intell. Syst., vol. 17, no. 4, pp. 64-70, Jul 2002.
10. David Sussillo, Anshul Kundaje, Dimitris Anastassiou, *"Spectrogram Analysis of Genomes"* , EURASIP Journal on Applied Signal Processing, pp. 29-42, July 2003.
11. D. Lee, "*LS-GKM : A new gkm-SVM for large scale datasets*", Bioinformatics, vol. 32, no. 14, pp. 2196-2198, 2016.
12. Nevenka Dimirova, Yee Him Cheung, "*Methods and systems for identification of DNA patterns through spectral analysis"*, May,2015.

13. Tung Hoang, Changchuan Yin, Hui Zheng, Chenglong Yu, Rong Lucy He, Stephen S.T. Yau, "*A new method to cluster DNA sequences using Fourier power spectrum*", 5 March, 2015.
14. Nevenka Dimitrova, Yee Him Cheung, Michael Zhang, "*Analysis and visualization of DNA Spectrograms : Open possibilities for genome research*", October, 2006.
15. Nilay Chheda, Naman Turakhia, Manish K. Gupta , Ruchin Shah and Jigar Raisinghani, "*Biospectrogram : a tool for spectral analysis of biological sequences*" , Vol. 00 no. 00 2012 Pages 1–3.

APPENDIX

Code for DCDE Model :

1. **Informative kmers Settlement –**

```
# informative kmers settlement
from __future__ import print_function
import numpy as np
from kpal.klib import Profile
from Bio import SeqIO
from sklearn.preprocessing import normalize

def Kmer_PWM(handle,kmer,length):
    PWM_k = np.zeros((length,251-kmer,4**kmer),dtype=np.float64)
    for i, seq_record in enumerate(SeqIO.parse(handle, "fasta")):
        print(i)
        seq_array = Profile.from_sequences([str(seq_record.seq)],kmer)
        for j in range(251-kmer):
            t = seq_array.dna_to_binary(str(seq_record.seq[j:j+kmer]))
            PWM_k[i,j,t] = seq_array.counts[t]
    return PWM_k

def Kmer_fren(handle,kmer):
    p = Profile.from_fasta_by_record(handle, kmer)
    t = next(p)
    tt = t.counts.astype(float)/t.total
    for i in p :
        t = i.counts.astype(float)/i.total
        tt = np.row_stack((tt,t))
    return tt

def IFkmer_SD(PP, Q, SD, T):
    # SD
    tdd = np.zeros((P.shape),dtype=np.float64)
    # KL -----------------------------------------
    if SD == 'KLD':
        for j in range(len(PP)):
            print(j)
            a = PP[j,:]
            div = a * np.log(a / Q)
            div[np.isnan(div)]=0
            div[np.isinf(div)]=0
            div_m = div.mean(axis=0)
tdd[j,:]=div_m
    # JD -----------------------------------------
    if SD == 'JD':
        for j in range(len(PP)):
            print(j)
            a = PP[j,:]
            div1 = a * np.log(a / Q)
            div1[np.isnan(div1)]=0
            div1[np.isinf(div1)]=0 #P->Q
            div2 = Q * np.log(Q / a)
            div2[np.isnan(div2)]=0
            div2[np.isinf(div2)]=0 #Q->P
            div = (div1+div2)/2
            div_m = div.mean(axis=0)
            tdd[i,:] = div_m
    #
    # JSD -----------------------------------------
    if SD == 'JSD':
        for j in range(len(PP)):
            print(j)
            a = PP[j,:]
            o = (a+Q)/2
            div1 = a * np.log(a / o)
            div1[np.isnan(div1)]=0
            div1[np.isinf(div1)]=0 #P->Q
            div2 = Q * np.log(Q / 0)
            div2[np.isnan(div2)]=0
            div2[np.isinf(div2)]=0 #Q->P
            div = (div1+div2)/2
            div_m = div.mean(axis=0)
            tdd[i,:] = div_m
    #-----------------------------------------------
    tdd_m = tdd.mean(axis=0)
    tdd_mm = -np.sort(-tdd_m)#value
    d=np.argsort(tdd_m)#index

    for i in range(len(tdd_mm)):
        R = np.sum(tdd_mm[0:i])/tdd_mm.sum()
        if  R > T:
            break
    index = d[0:i]
    return index

def
IFkmer_trn_tst(H_trn_p,H_trn_n,H_tst_p,H_tst_n,num_trn_p,num_trn_n,num_tst_p,num_tst_n,
kmer, SD, T):
    trn_p = Kmer_fren(open(H_trn_p,'r'), kmer)
    trn_n = Kmer_fren(open(H_trn_n,'r'), kmer)
    tst_p = Kmer_fren(open(H_tst_p,'r'), kmer)
    tst_n = Kmer_fren(open(H_tst_n,'r'), kmer)

    np.save('trn_p.npy',trn_p)
    np.save('trn_n.npy',trn_n)
    np.save('tst_p.npy',tst_p)
    np.save('tst_n.npy',tst_n)
#============================================================================
    PWM_trn_p = Kmer_PWM(open(H_trn_p,'r'),kmer,num_trn_p)
    PWM_trn_n = Kmer_PWM(open(H_trn_n,'r'),kmer,num_trn_n)
    PWM_tst_p = Kmer_PWM(open(H_tst_p,'r'),kmer,num_tst_p)
    PWM_tst_n = Kmer_PWM(open(H_tst_n,'r'),kmer,num_tst_n)

    np.save('PWM_trn_p.npy',PWM_trn_p)
    np.save('PWM_trn_n.npy',PWM_trn_n)
    np.save('PWM_tst_p.npy',PWM_tst_p)
    np.save('PWM_tst_n.npy',PWM_tst_n)
#============================================================================
    index = IFkmer_SD(trn_p, trn_n, SD, T)
    np.save('index.npy',index)
#============================================================================
XX_trn = np.row_stack((PWM_trn_p,PWM_trn_n))
XX_tst = np.row_stack((PWM_tst_p,PWM_tst_n))
YY_trn = np.zeros(len(X_trn),dtype=np.int64)
YY_trn[0:num_trn_p]=1
YY_tst = np.zeros(len(X_tst),dtype=np.int64)
YY_tst[0:num_tst_p]=1

XX_trn = XX_trn[:,:,index]
XX_tst = XX_tst[:,:,index]

#    for j in range(len(X_trn)):
#        XX_trn[j,:,:] = normalize(XX_trn[j,:,:], norm='l2')
#    for j in range(len(X_tst)):
#        XX_tst[j,:,:] = normalize(XX_tst[j,:,:], norm='l2')
```

```
    return (XX_trn, XX_tst, YY_trn, YY_tst)

#===========================================================================
# Main
kmer = 3 # (1,2,3,4,5,6) recommend
num_trn_p = 2000
num_trn_n = 2000
num_tst_p = 100
num_tst_n = 100
SD = 'KLD' # 'KLD' 'JD' 'JSD'
T = 0.98 # threshold in manuscript 2.1.B
H_trn_p = 'training_2_positive.fasta'
H_trn_n = 'training_2_negative.fasta'
H_tst_p = 'test_2_positive.fasta'
H_tst_n = 'test_2_negative.fasta'

(XX_trn,XX_tst, YY_trn, YY_tst) = IFkmer_trn_tst(H_trn_p, H_trn_n, H_tst_p, H_tst_n,num_trn_p, num_trn_n,num_tst_p,num_tst_n, kmer, SD, T)

np.save('xx_train.npy',XX_trn)
np.save('xx_test.npy',XX_tst)
np.save('yy_train.npy',YY_trn)
np.save('yy_test.npy',YY_tst)
```

2. **Convolutional Neural Network –**

```
from __future__ import print_function
import numpy as np
from sklearn.preprocessing import normalize
import keras
from keras import regularizers
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D
import theano
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

def  DCDE_CNN(img_rows,  img_cols,  filter_size,  kernel_size,  pool_size,  xx_train,  yy_train,
xx_test, yy_test):

    input_shape = (img_rows, img_cols)
    #Conv layer
    model = Sequential()
    model.add(Conv1D(filter_size, kernel_size=kernel_size,
                     activation='relu',
                     kernel_regularizer=regularizers.l2(0.01),
                     input_shape=input_shape))
    ##MaxPooling
    model.add(MaxPooling1D(pool_size=pool_size))
    #full connection
    model.add(Flatten())
    model.add(Dense(output_size,activation='relu'))
    model.add(Dropout(0.25))
    #softmax
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.binary_crossentropy,
                  optimizer=keras.optimizers.Adam(),
                  metrics=['accuracy'])

    model.fit(x_train, y_train,
              batch_size=batch_size,
            epochs=epochs,
            verbose=1,
validation_data=(xx_test,                                        yy_test))
#==================================================================
  get_feature = keras.backend.function([model.layers[0].input],model.layers[3].output)
  FC_train_feature = get_feature(x_train)
  FC_test_feature = get_feature(x_test)
  FC_train_feature = normalize(FC_train_feature, norm='l2')
  FC_test_feature = normalize(FC_test_feature, norm='l2')

  return (FC_train_feature, FC_test_feature)
#==================================================================
kmer = 3
batch_size = 40
num_classes = 2
epochs = 1
filter_size = 64
kernel_size = 5
pool_size = 2
output_size = 128
#==================================================================
xx_train = np.load('xx_train.npy')
xx_test = np.load('xx_test.npy')
yy_train = np.load(y'y_train.npy')
yy_test = np.load('yy_test.npy')
index = np.load('index.npy')
img_cols, img_rows  = len(index), 251-kmer
yy_train_new = yy_train
yy_test_new = yy_test
yy_train = keras.utils.to_categorical(yy_train, 2)
yy_test = keras.utils.to_categorical(yy_test, 2)
```

```
np.save('yy_train_new.npy',yy_train_new)
np.save('yy_test_new.npy',yy_test_new)
#==========================================================================
#CNN
(FC_train_feature, FC_test_feature) = DCDE_CNN(img_rows, img_cols, filter_size,
kernel_size, pool_size, xx_train, yy_train, xx_test, yy_test)

np.save('FCctrain_feature.npy',FCctrain_feature)
np.save('FCctest_feature.npy',FCctest_feature)
```

3. **Multiple SVM –**

```
from __future__ import print_function
import numpy as np
from sklearn.svm import SVC

def svc(traindata,trainlabel,testdata,testlabel):
    print("Start training SVM...")

    (L,D) = traindata.shape
    b = np.zeros(L,dtype=np.float64)
    x = traindata.mean(axis=0)
    for i in range(L):
        a = np.linalg.norm(traindata[i,:]-x)
        b[i] = a
    mx = np.median(b)
    par = 1/np.square(mx)

    svcClf = SVC(C=5.0,kernel="rbf",tol=1e-4,gamma=par,probability=True)
#    svcClf = SVC(C=5.0,kernel="rbf",tol=1e-4,gamma='auto',probability=True)
    svcClf.fit(traindata,trainlabel)

    pred_testlabel = svcClf.predict(testdata)
    proba_testlabel = svcClf.predict_proba(testdata)
    num = len(pred_testlabel)
    accuracy = len([1 for j in range(num) if testlabel[j]==pred_testlabel[j]])/float(num)
    print("cnn-svm Accuracy:",accuracy)
    return pred_testlabel, proba_testlabel
#==================================================================
FC_train_feature = np.load('FC_train_feature.npy')
FC_test_feature = np.load('FC_test_feature.npy')
y_train_new = np.load('y_train_new.npy')
y_test_new = np.load('y_test_new.npy')
#==================================================================
#SVM
pred_testlabel, proba_testlabel = svc(FC_train_feature,y_train_new,FC_test_feature,y_test_new)
np.save('pred_testlabel.npy',pred_testlabel)
np.save('proba_testlabel.npy',proba_testlabel)
#==================================================================
#evaluate
pre = pred_testlabel
ins1 = np.where(y_test_new==1)
ins2 = np.where(y_test_new==0)
Sn = float(np.sum(pre[ins1]==1)) / float((np.sum(pre[ins1]==1)+np.sum(pre[ins2]==1)))
Sp = float(np.sum(pre[ins2]==0)) / float((np.sum(pre[ins2]==0)+np.sum(pre[ins1]==0)))
t = float(np.sum(pre[ins2]==0)) / float((np.sum(pre[ins2]==0)+np.sum(pre[ins2]==1)))
SP = float(np.sum(pre[ins1]==1)) / float((np.sum(pre[ins1]==1)+np.sum(pre[ins1]==0)))
ACP = (Sn + Sp + SP + t)/4
print(Sn,Sp,ACP)
```

4. **Bilayer Decision Model**

```
from __future__ import print_function
import numpy as np
from sklearn.metrics import roc_auc_score
```

```
#==========================================================================
# BD model
PE_pred_testlabel = np.load('PE_pred_testlabel.npy')
PE_proba_testlabel = np.load('PE_proba_testlabel.npy')

PI_pred_testlabel = np.load('PI_pred_testlabel.npy')
PI_proba_testlabel = np.load('PI_proba_testlabel.npy')

PU_pred_testlabel = np.load('PU_pred_testlabel.npy')
PU_proba_testlabel = np.load('PU_proba_testlabel.npy')

yy_train_new = np.load('yytrain_new1.npy')
yy_test_new = np.load('yytest_new1.npy')
yy_train = np.load('yytrain1.npy')
yy_test = np.load('yytest1.npy')

TT = 0.5 # threshold in manuscript 3.2
#==========================================================================
sum = PE_pred_testlabel + PI_pred_testlabel + PU_pred_testlabel
vote = np.zeros((sum.shape),dtype=int)
vote[np.where(sum>1)] = 1
prob = np.zeros((PU_proba_testlabel.shape),dtype=float)
p = np.column_stack((PE_proba_testlabel[:,1],PI_proba_testlabel[:,1],PU_proba_testlabel[:,1]))
p_max = p.max(axis = 1)
p_mean = p.mean(axis = 1)
p_min = p.min(axis = 1)

prob[np.where(sum==3),1] = p_max[np.where(sum==3)]
prob[np.where(sum==2),1] = p_mean[np.where(sum==2)]
prob[np.where(sum==1),1] = p_min[np.where(sum==1)]
prob[np.where(sum==0),1] = p_min[np.where(sum==0)]
```

```
prob[:,0] = 1 - prob[:,1]

auc = roc_auc_score(y_test,prob)
print(auc)

pre = np.zeros((y_test_new.shape),dtype = int)
pre[np.where(prob[:,1]>TT)] = 1
#==================================================================
#show performance
ins1 = np.where(y_test_new==1)
ins2 = np.where(y_test_new==0)
Sn = float(np.sum(pre[ins1]==1)) / float((np.sum(pre[ins1]==1)+np.sum(pre[ins2]==1)))
Sp = float(np.sum(pre[ins2]==0)) / float((np.sum(pre[ins2]==0)+np.sum(pre[ins1]==0)))
t = float(np.sum(pre[ins2]==0)) / float((np.sum(pre[ins2]==0)+np.sum(pre[ins2]==1)))
SP = float(np.sum(pre[ins1]==1)) / float((np.sum(pre[ins1]==1)+np.sum(pre[ins1]==0)))
ACP = (Sn + Sp + SP + t)/4
print(Sn,Sp,ACP)
np.save('Pre.npy',pre)
np.save('Prob.npy',prob)
```

Code for Biospectral Analysis Method :

1. **Plotting Spectrograms –**

```
clc;
clear all;
clf;
x=textread(‘file_path.dat','%s');
aa=char(xx);
[r1 c1]=size(aa);
cc=0;
for ii1=1:r1
for jj1=1:c1
```

```
cc=[ccaa(ii1,jj1)];
end
end
 L1=sqrt(-1);
% % %2 prog. To convert text data to discrete form
Dlength=length(cc);
for k=1:Dlength-1
code=cc(k+1);
switch (code)
case {'A','a'}
I6(:,i)=1';
case {'G','g'}
I6(:,i)=2';
case {'C','c'}
I6(:,i)=3';
case {'T','t'}
I6(:,i)=4';
end
end
spectrogram(I6);% finding out the STFT_
```

2. **CNN –**

```
#importing keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

#Initializing the CNN
classifier = Sequential()

#Step 1 - Convolution
classifier.add(Convolution2D(16,( 3, 3), input_shape = (128, 128, 3), activation = 'relu'))
#Step 2 - Pooling
```

```
classifier.add(MaxPooling2D(pool_size=(2,2)))

classifier.add(Convolution2D(16,(3,3), activation= 'relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))

classifier.add(Convolution2D(16,(3,3), activation= 'relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))


#Step 3 - Flattening
classifier.add(Flatten())

#Step 4 - Full Connection
classifier.add(Dense( activation= "relu",units =64))
classifier.add(Dense( activation= "sigmoid", units = 1))

#Compiling the CNN
classifier.compile(optimizer= 'adam', loss = 'binary_crossentropy', metrics= ['accuracy'])

#Fitting CNN to images
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizont
al_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('/content/drive/My Drive/Colab Notebooks/cat
egories/Train',target_size=(128, 128),batch_size=32,class_mode='binary')

testing_set = test_datagen.flow_from_directory('/content/drive/My Drive/Colab Notebooks/categ
ories/Test',target_size=(128, 128),batch_size=32,class_mode='binary')

classifier.fit_generator(training_set,steps_per_epoch=1,epochs=50,validation_data=testing_set,v
alidation_steps=40)
```

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 20-07-20

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _AYSHIKA KAPOOR & Uday Sharma_ **Department:** _ECE_ **Enrolment No** _161080 & 161683

Contact No. _**8808064814 & 9459388007**- **E-mail.** ayshika.kapoor@gmail.com **& Uday007sharma@gmail.com**

Name of the Supervisor: _______DR. SUNIL DATT SHARMA________

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ______PROMOTER RECOGNITION IN HUMAN DNA HUMAN SEQUENCE USING DEEP LEARNING ________

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 69

Udaysharma

- Total No. of Preliminary pages = 12
- Total No. of pages accommodate bibliography/references = 3

Ayshika

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....16.......(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 20.07.20

(Signature of Guide/Supervisor)

[illegible] 21/07/2020

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Abstract & Chapters Details**	
			Word Counts	
			Character Counts	
Report Generated on		**Submission ID**	Page counts	
			File Size	

Checked by
Name & Signature

Librarian

....

Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at plagcheck.juit@gmail.com