Interpretable Machine Learning Towards Precision Medicine

Internship Report

of

Summer Research Internship

Submitted by

Barnak Ghosh

Under the guidance of

Dr. Aparajita Dutta

Department of Computer Science
And Engineering
Indian Institute of Information
Technology, Guwahati

June 2023 – July 2023

Contents

1. Introduction 5
1.1 Problem Definition..... 5

2. Introduction to Python and Deep Learning Models 6
2.1 Introduction to Python Programming..... 6
2.2 Introduction to TensorFlow and Keras..... 6
2.3 Multi-layer Perceptron (MLP) 6
2.4 Convolution Neural Network (CNN) 7
2.5 Recurrent Neural Network (RNN) 7
2.6 Long Short-Term Memory (LSTM)..... 7
2.7 Practical Examples..... 8
2.7.1 Multi-Layer Perceptron (MLP) Using Iris Dataset..... 8
2.7.2 Sentiment Analysis using Convolutional Neural Network (CNN) on IMDB Movie Reviews..... 9
2.7.3 Sentiment Analysis on IMDB Movie Reviews using SimpleRNN..... 10
2.7.4 Sentiment Analysis on IMDB Movie Reviews using LSTM..... 12
2.8 Gated Recurrent Unit (GRU).....13
2.9 The Transformer..... 14

3. Existing Methodologies 15
3.1 Splicing..... 16

4. SpliceVec: Distributed Feature Representations for Splice Junction Prediction 18
4.1 Introduction..... 18
4.2 Methodology..... 18
4.3 Implementation of SpliceVec..... 20
4.4 Result and Discussion..... 20
4.5 Conclusion..... 20

5. SpliceVisuL: Visualization of Bidirectional Long Short-term Memory Networks for Splice - Junction Prediction 20
5.1 Introduction..... 20
5.2 Methodology..... 20
5.3 Visualization Technique..... 21
5.3.1 Smooth Gradient with Noisy Nucleotide Embeddings..... 21
5.3.2 Integrated Gradient with Nucleotide Embeddings..... 21
5.3.3 Omission of a Single Nucleotide..... 22
5.3.4 Occlusion of k-mers..... 22
5.4 Experimental Setup..... 22
5.5 Results and Discussion..... 22
5.6 Conclusion..... 23

6. Bibliography 23

Abstract

Precision medicine has transformed healthcare through personalized therapies based on individual genetics and disease characteristics. The significance of splicing in cellular functions and multifactorial disorders like cancer is the main focus of this report's investigation into the use of deep learning models and visualization approaches in precision medicine. The study offers detailed instructions on how to create and evaluate deep learning models using Python, TensorFlow, and Keras. It emphasizes the use of pre-processing and biomarker extraction techniques using publicly accessible genetic datasets like GENCODE. SpliceVec, a computational framework for precise splicing event prediction, is introduced in the study as a solution to the problem of comprehending splicing events. We investigate visualization techniques for complicated splicing pattern interpretation. Through personalized therapies and a deeper comprehension of how genes are regulated in illnesses, this work highlights the potential of precision medicine and improves patient care. Precision medicine develops individualized therapeutics and promotes healthcare by fusing computational technologies with biological knowledge.

Acknowledgment

My sincere appreciation to Dr. Aparajita Dutta for his valued advice and support throughout my effort. I would like to thank Ms. Debleena Bhattacharjee (Ph.D., Dept. of CSE, Indian Institute of Information Technology Guwahati) for helping me to comprehend the intricate biology and deep learning processes. The internship program has benefited greatly from their assistance. A special thanks go out to Biswajit Senapati, Eshwar Gelam, Mrityunjoy Deka, and Partha Pratim Sarmah, my co-author internship colleagues, for their efforts in the internship.

1. Introduction

Precision medicine has fundamentally altered the healthcare sector by personalizing therapies for each patient based on their genetic makeup, environmental factors, and particular disease features. We are now much better equipped to decipher complex biological data and gather important information about multifactorial illnesses like cancer thanks to the development of deep learning algorithms and advanced visualization techniques. This study investigates how deep learning models and visualization methods are used in precision medicine, with a focus on how biological mechanisms, notably splicing, work to manage distinct disorders.

The report's first section provides step-by-step instructions for developing and analyzing deep learning models using well-known programming languages like Python and frameworks like TensorFlow and Keras. The report examines the application of machine learning in precision medicine to look at how deep learning models may assist with the detection and understanding of these complex illnesses. Highlighting cellular processes, particularly splicing, as possible regulators of such circumstances helps to explain the intricate mechanisms behind the beginning and development of multifactorial disorders.

This work demonstrates the value of openly available datasets like GENCODE for prepping genomic sequences and optimizing them for deep learning models through data cleaning, feature extraction, and normalization. It directs computational biologists and researchers in precision medicine to find insights and therapy targets using deep learning models and visualization tools by emphasizing biomarker extraction as essential to understanding biological processes. This thorough how-to handbook emphasizes the revolutionary potential of precision medicine, enhancing patient care and therapy results for complicated illnesses.

1.1 Problem Definition

Understanding splicing, a crucial biological process involved in gene expression, is essential in the disciplines of molecular biology and precision medicine. Splicing activities control the variety of proteins that genes encode, and they can significantly affect cancer and other diseases. The study and interpretation of splicing events from massive genomic data, however, face significant challenges due to their complexity and the need for powerful computational tools.

This study paper concurrently addresses two major challenges. SpliceVec, a cutting-edge computational framework designed to precisely anticipate and classify splicing events in genomic sequences, is the first new technology introduced in this section. Finding alternative splicing patterns and predicting their functional effects will help us understand how splicing control affects illness. The study also looks at the visualization of splicing events in order to draw pertinent conclusions and weigh potential applications. To help researchers and computational biologists intuitively explore and comprehend complicated splicing patterns, potent visualization techniques are being developed. This will help them find hidden patterns and learn important biological information. Overall, the research study addresses the requirement for accurate splicing event prediction and categorization by developing the SpliceVec computational framework. Additionally, it aims to advance our understanding of gene regulation and its effects on a variety of illnesses by providing effective visualization techniques that make it simpler to analyze splicing processes.

2. Introduction to Python and Deep Learning Models

Practical teaching on Python programming and well-known deep learning frameworks like TensorFlow and Keras are provided in this chapter. It aims to equip the reader with the skills required to effectively create and instruct learning models. One has to have a thorough grasp of the underlying concepts of these programming tools in order to effectively utilize the power of deep learning in a range of sectors, including precision medicine.

2.1 Introduction to Python Programming:

Python is a versatile and well-liked programming language praised for its clarity and ease of use. It offers a friendly environment for new programmers as well as a wide range of data processing and scientific computing tools and frameworks. Understanding the foundations of Python is crucial for developing learning models for precision medicine utilizing deep learning frameworks like TensorFlow and Keras.

2.2 Introduction to TensorFlow and Keras:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

2.3 Multi-layer Perceptron (MLP):

MLP is the abbreviation for multi-layer perception. It is made up of thick, completely linked layers that may change any input dimension into the required dimension. A neural network with numerous layers is referred to as a multi-layer perception. In order to build a neural network, we join neurons so that some of their outputs are also their inputs.

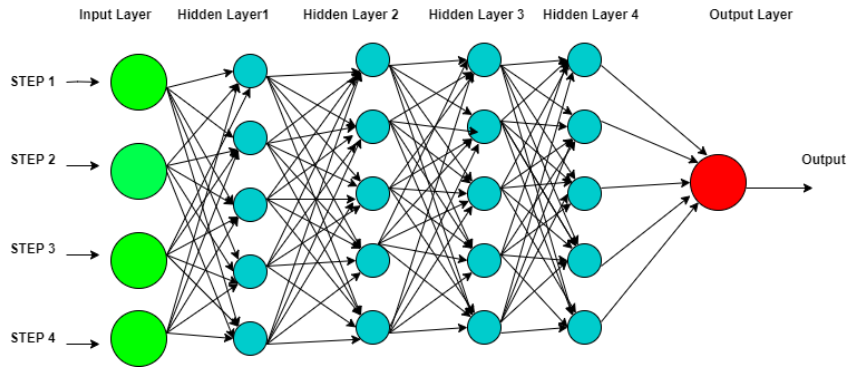


Fig: Multi-Layer Perceptron (MLP) diagram with four hidden layers and a collection of single nucleotide polymorphisms

2.4 Convolutional Neural Network (CNN):

Machine learning includes convolutional neural networks (CNNs, sometimes known as convents). It is a subset of the several artificial neural network models that are employed for diverse purposes and data sets. A CNN is a particular type of network design for deep learning algorithms that are utilized for tasks like image recognition and pixel data processing.

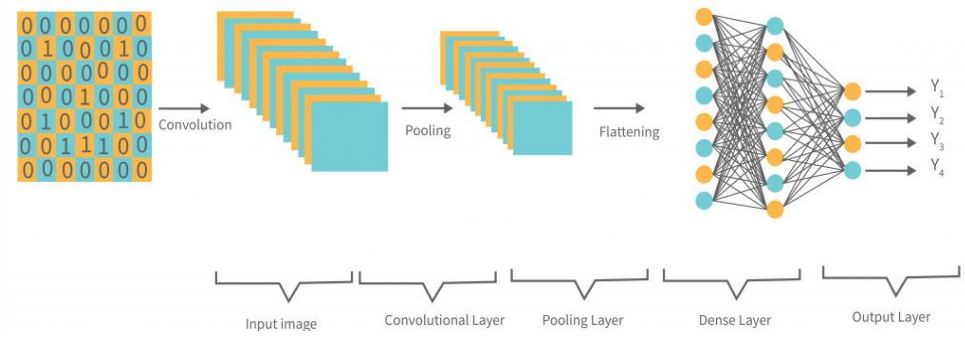


Fig: CNN architecture

2.5 Recurrent Neural Network (RNN):

An artificial neural network that employs sequential data or time series data is known as a recurrent neural network (RNN). These deep learning algorithms are included in well-known programs like Siri, voice search, and Google Translate. They are frequently employed for ordinal or temporal issues, such as language translation, natural language processing (NLP), speech recognition, and picture captioning. Recurrent neural networks (RNNs) use training data to learn, just like feedforward and convolutional neural networks (CNNs) do. Because they use data from earlier inputs to affect the present input and output, they are characterized by their "memory". Recurrent neural networks' outputs are reliant on the previous parts in the sequence, unlike typical deep neural networks, which presume that inputs and outputs are independent of one another.

2.6 Long Short-Term Memory (LSTM):

A key element of deep learning, particularly in the context of recurrent neural networks (RNNs), is the Long Short-Term Memory (LSTM) model. Traditional RNNs frequently experience vanishing and inflating gradient issues, which make it difficult for them to accurately detect long-range dependencies in sequential data. LSTM was created to overcome these issues. LSTMs are capable of selectively storing and retrieving information across long time intervals thanks to the inclusion of specialized memory cells and gating mechanisms. As a result, they are skilled at managing sequential data with considerable time delays. This special capacity has been widely used in tasks requiring sequential data such as time series forecasting, speech recognition, natural language processing, and other tasks involving sequential data, securing LSTM's place as a potent and crucial tool in deep learning practitioners' toolboxes.

2.7 Practical Examples:

2.7.1 Multi-Layer Perceptron (MLP) Using the Iris Dataset:

Introduction:

The MLP is a type of neural network with several layers of linked neurons that is frequently used for classification tasks. A well-known benchmark dataset in machine learning is the Iris dataset, which comprises measurements of floral characteristics for three different types of Iris flowers.

In order to categorize iris species using the Iris dataset, we want to create a scalable MLP model. A neural network called MLP is very good at classifying and recognizing patterns. Iris flower samples with four characteristics and three species make up the Iris dataset. By defining the number of hidden layers and neurons, users may alter the MLP design. On the training set, the model is developed, and the testing set is used for evaluation.

Dataset and Pre-processing:

The Iris dataset is obtained using `load_iris` from `sklearn.datasets`, divided into features (X) and labels (y). `train_test_split` from `sklearn.model_selection` splits the data into training and testing sets.

Customization of MLP Architecture:

The number of hidden layers and neurons in each layer are interactively set by users. This personalization enables testing with different model setups.

Model Architecture and Training:

The MLP model, built with TensorFlow's Keras API, has an input layer with four neurons and user-specified hidden layers with ReLU activation. The output layer contains three neurons for iris species classification. The model is compiled with loss function, optimizer, and accuracy metric, and trained using `model.fit`.

Evaluation and Performance Metrics:

The model's performance is evaluated on the testing set using `model.predict`. The `classification_report` provides detailed metrics, including precision, recall, F1-score, and support for each class.

Results and Discussion:

The performance of the model is examined in the classification report using a user-specified MLP architecture. On accuracy, precision, recall, and F1-score, the effects of various settings are explored.

Fig: Output

	Precision	Recall	F1-score	Support
Setosa	1.00	1.00	1.00	19
Versicolor	0.94	0.94	0.94	16
Virginica	0.94	0.94	0.94	10
Accuracy			0.96	45
Macro Avg.	0.96	0.96	0.96	45
Weighted Avg.	0.96	0.96	0.96	45

2.7.2 Sentiment Analysis using Convolutional Neural Network (CNN) on IMDB Movie Reviews:

Introduction:

On the IMDB movie review dataset, we develop a convolutional neural network (CNN) for sentiment analysis. Finding the sentiments represented in a text, such as the positive or negative sentiments in movie reviews, is the work of sentiment analysis. The CNN model seeks to categorize the reviews as favorable or negative based on their sentiment. The IMDB dataset comprises movie reviews expressed as word indices.

Dataset and Pre-processing:

The Keras API of TensorFlow is used to load the IMDB movie reviews dataset. It has a predetermined vocabulary that can contain up to 10,000 words and a review that can have no more than 500 words. 'sequence.pad_sequences' is used to pre-process the data by padding the sequences to have the same length.

CNN Model Architecture:

The Sequential API of TensorFlow was used to create the CNN model. It starts by creating dense vectors from word indices using an embedding layer. After that, features are extracted using a 1D convolutional layer with 64 filters and ReLU activation. The most significant characteristics are then extracted using a global max-pooling layer. For additional feature refining, two fully linked dense layers with ReLU activation are added. The output layer employs a sigmoid activation for binary sentiment categorization.

Model Compilation and Training:

Binary cross-entropy loss and the Adam optimizer are used in the model's construction. For assessment, the accuracy metric is employed. With a batch size of 32, the CNN model is trained on the training set for five epochs. During training, the model's performance is tracked using the validation data.

Evaluation:

The model's performance is assessed on the testing set after training. To evaluate the model's performance in sentiment classification, the test loss and accuracy are computed using 'model.evaluate' and printed.

Conclusion:

On the IMDB movie review dataset, the CNN model displays encouraging sentiment analysis outcomes. Based on their substance, it effectively categorizes movie reviews as favorable or negative. Deep learning models have the capacity to comprehend and analyze textual material, as shown by CNNs' application in natural language processing tasks like sentiment analysis.

This study successfully uses a CNN model to do sentiment analysis on movie reviews, demonstrating the value of deep learning in handling textual data for sentiment categorization. CNN is an effective tool for many applications of natural language processing because it can extract significant characteristics from sequences. The model may perform better and have more generalization potential with additional investigation and tweaking.

Fig: Output

Epoch 1/5
782/782 [==============================] - 20s 25ms/step - loss: 0.4176 - accuracy: 0.7998 - val_loss: 0.2882 - val_accuracy: 0.8785
Epoch 2/5
782/782 [==============================] - 19s 25ms/step - loss: 0.2008 - accuracy: 0.9247 - val_loss: 0.2562 - val_accuracy: 0.8982
Epoch 3/5
782/782 [==============================] - 19s 24ms/step - loss: 0.0873 - accuracy: 0.9713 - val_loss: 0.2903 - val_accuracy: 0.8948
Epoch 4/5
782/782 [==============================] - 19s 25ms/step - loss: 0.0239 - accuracy: 0.9955 - val_loss: 0.3572 - val_accuracy: 0.8906
Epoch 5/5
782/782 [==============================] - 19s 24ms/step - loss: 0.0063 - accuracy: 0.9994 - val_loss: 0.4069 - val_accuracy: 0.8910
782/782 [==============================] - 4s 5ms/step - loss: 0.4069 - accuracy: 0.8910
Test Loss: 0.4069
Test Accuracy: 0.8910

2.7.3 Sentiment Analysis on IMDB Movie Reviews using SimpleRNN

Introduction:

On the IMDB movie review dataset, we execute sentiment analysis using a Simple Recurrent Neural Network (SimpleRNN). Classifying movie reviews as good or negative is a component of sentiment analysis. The IMDB dataset comprises word indices of movie reviews, and the

objective is to create a SimpleRNN model that can comprehend the sequential structure of the data.

Dataset and Pre-processing:

The Keras API of TensorFlow is used to load the IMDB movie reviews dataset. To guarantee constant input dimensions for the model, the vocabulary is limited to a maximum of 10,000 words, and each review is padded or shortened to a length of no more than 500 words. For model evaluation during training, the training data is further divided into training and validation sets.

SimpleRNN Model Architecture:

The Sequential API of TensorFlow is used to build the SimpleRNN model. It begins with an embedding layer that turns word indices into dense vectors to make it easier to express words in meaningful ways. Then, to capture the sequential dependencies inside the reviews, the SimpleRNN layer is introduced with 64 units. For binary sentiment classification, a Dense layer with one neuron and a sigmoid activation function are utilized.

Model Compilation and Training:

Binary cross-entropy loss and the Adam optimizer are used in the compilation of the SimpleRNN model. The performance of the model is evaluated using the accuracy measure. A 32-person batch size is used during the model's five epochs of training on the training set. The model's development during training is tracked using the validation data.

Evaluation:

The performance of the SimpleRNN model is assessed on the test set after training. Insights about the model's capacity to generalize and categorize movie reviews based on sentiment are provided by the test loss and accuracy, which are computed and printed.

Conclusion:

On the IMDB movie review dataset, the SimpleRNN model exhibits success in sentiment analysis. It is appropriate for natural language processing jobs because of its recurring nature, which enables it to take into account the sequential information included in the reviews. The model's performance on the test set demonstrates how well it can categorize movie reviews as positive or negative with accuracy. The performance and generalization potential of the model may be improved by further investigation of other recurrent architectures and hyperparameter adjustment.

As a result, this code demonstrates how to use a SimpleRNN model for sentiment analysis, emphasizing the value of recurrent neural networks in handling sequential data for applications involving natural language processing.

Fig: Output

Epoch 1/5
500/500 [==============================] - 63s 123ms/step - loss: 0.7175 - accuracy: 0.5265 - val_loss: 0.6062 - val_accuracy: 0.6550
Epoch 2/5
500/500 [==============================] - 60s 119ms/step - loss: 0.4489 - accuracy: 0.7963 - val_loss: 0.5435 - val_accuracy: 0.7338
Epoch 3/5
500/500 [==============================] - 60s 119ms/step - loss: 0.3455 - accuracy: 0.8569 - val_loss: 0.4578 - val_accuracy: 0.7994
Epoch 4/5
500/500 [==============================] - 59s 118ms/step - loss: 0.2022 - accuracy: 0.9225 - val_loss: 0.5272 - val_accuracy: 0.7780
Epoch 5/5
500/500 [==============================] - 58s 117ms/step - loss: 0.1256 - accuracy: 0.9543 - val_loss: 0.5842 - val_accuracy: 0.7822
782/782 [==============================] - 23s 28ms/step - loss: 0.5976 - accuracy: 0.7750
SimpleRNN Test Loss: 0.5976
SimpleRNN Test Accuracy: 0.7750

2.7.4 Sentiment Analysis on IMDB Movie Reviews using LSTM

Introduction:

On the IMDB movie review dataset, we execute sentiment analysis using Long Short-Term Memory (LSTM), a sort of recurrent neural network (RNN). It is the goal to categorize movie reviews as having good or negative feelings. The code seeks to create an LSTM model that can comprehend the sequential nature of the data. The IMDB dataset comprises movie reviews expressed as word indices.

Dataset and Pre-processing:

The Keras API of TensorFlow is used to load the IMDB movie reviews dataset. To guarantee constant input dimensions for the LSTM model, the vocabulary is limited to a maximum of 10,000 words, and each review is padded or shortened to a length of no more than 500 words. To track the model's progress during training, the training data is further divided into training and validation sets.

LSTM Model Architecture:

The Sequential API of TensorFlow is used to construct the LSTM model. It starts with an embedding layer that turns word indices into dense vectors to make it easier to express words in meaningful ways. To identify long-term dependencies within the sequential data, a 64-unit LSTM layer is applied. For binary sentiment classification, a Dense layer with one neuron and a sigmoid activation function is utilized.

Model Compilation and Training:

Adam optimizer and binary cross-entropy loss are used to build the LSTM model. The model's performance is evaluated using the accuracy measure. With a batch size of 32, the model is

trained on the training set for five iterations. The validation data is used to track the model's development throughout training.

Evaluation:

The effectiveness of the LSTM model is assessed on the test set after training. Insights about the model's capacity to generalize and categorize movie reviews based on sentiment are provided by the test loss and accuracy, which are computed and printed.

Conclusion:

The IMDB movie review dataset serves as a testbed for the LSTM model's efficacy in sentiment analysis. It is ideally suited for comprehending sequential data like movie reviews due to its recurring nature, particularly the capacity to capture long-term dependencies. The model's performance on the test set demonstrates how well it can categorize movie reviews as positive or negative with accuracy. The performance and generalization abilities of the model may be improved with more testing and hyperparameter modification.

As a result, this code illustrates how an LSTM model may be applied to sentiment analysis, emphasizing the value of recurrent neural networks in processing sequential data for applications involving natural language processing.

Fig: Output:

Epoch 1/5
500/500 [==============================] - 58s 111ms/step - loss: 0.4187 - accuracy: 0.8025 - val_loss: 0.3118 - val_accuracy: 0.8728
Epoch 2/5
500/500 [==============================] - 56s 113ms/step - loss: 0.2409 - accuracy: 0.9081 - val_loss: 0.3311 - val_accuracy: 0.8714
Epoch 3/5
500/500 [==============================] - 56s 113ms/step - loss: 0.1590 - accuracy: 0.9420 - val_loss: 0.4083 - val_accuracy: 0.8612
Epoch 4/5
500/500 [==============================] - 56s 112ms/step - loss: 0.1089 - accuracy: 0.9626 - val_loss: 0.4692 - val_accuracy: 0.8650
Epoch 5/5
500/500 [==============================] - 56s 113ms/step - loss: 0.0797 - accuracy: 0.9741 - val_loss: 0.4795 - val_accuracy: 0.8526
782/782 [==============================] - 24s 30ms/step - loss: 0.5085 - accuracy: 0.8450
LSTM Test Loss: 0.5085
LSTM Test Accuracy: 0.8450

2.8 Gated Recurrent Unit (GRU):

Recurrent neural network (RNN) architecture with a gated recurrent unit (GRU) was developed by Cho et al. in 2014. By introducing gating mechanisms, it overcomes the drawbacks that typical RNNs suffer, such as vanishing gradients and long-term dependence problems. Reset and update gates in the GRU regulate the information flow through hidden states. The input and forget gates are combined into a

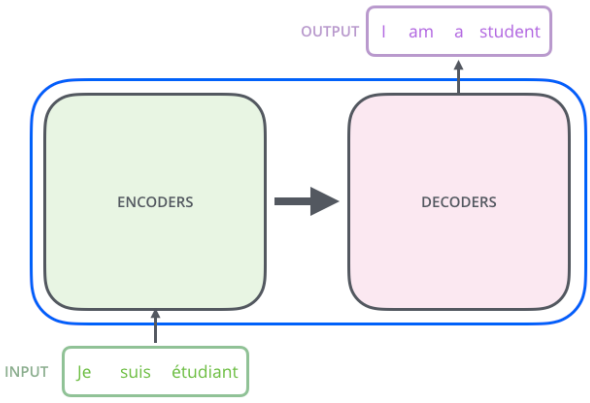
single update gate in GRU, not LSTM, which reduces computational complexity and makes implementation simpler. GRU's capacity to efficiently capture long-term dependencies while minimizing vanishing gradient issues has helped it perform well in a number of natural language processing applications. It is a well-liked option for activities requiring sequential data and other applications because of its effectiveness and simplicity.

2.9 The Transformer

The Transformer is an innovative deep learning model that makes use of attention processes to greatly improve training efficiency and performance across a range of applications, including neural machine translation. The Transformer outperforms the Google Neural Machine Translation model in a few particular tasks, but its greatest benefit comes from its effective parallelization abilities, making it the suggested reference model for Google Cloud's Cloud TPU product. The Transformer was first described in the paper "Attention is All You Need," and it has been implemented in TensorFlow as a component of the Tensor2Tensor package. Harvard's NLP lab has also annotated a PyTorch version of The Transformer. In order to make The Transformer accessible to readers without extensive topic knowledge, we have simplified and introduced the essential ideas in this essay piecemeal.

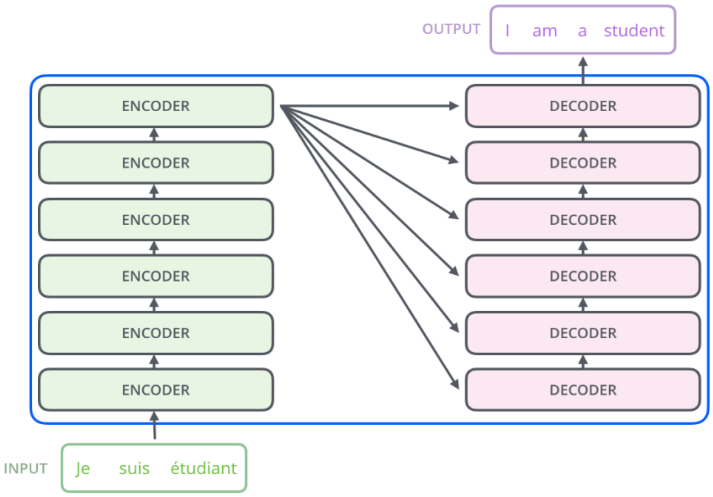
Copyright: jalammar.github.io/illustrated-transformer/

Fig: Taking a broad view, imagine the model as a single, empty box. A machine translation program would take a phrase written in one language and translate it into the other language.



Copyright: jalammar.github.io/illustrated-transformer/

Fig: A decoding component, an encoding component, and relationships between them may be seen as we crack into that Optimus Prime goodie.



Copyright: jalammar.github.io/illustrated-transformer/

Fig: The encoding component is a stack of encoders (the article arranges them in a stack of six, but one may certainly experiment with different configurations); there is nothing special about the number six). A stack of identical-number decoders makes up the decoding component.

Conclusion:

This chapter provided detailed guidance on how to create learning models using the Python programming language and well-known deep learning frameworks like TensorFlow and Keras. It discussed Python's basic concepts, introduced TensorFlow and Keras, and used the IMDB movie reviews dataset to illustrate sentiment analysis using a variety of neural network architectures. The examples demonstrated the importance of data pre-processing, model creation, training, and assessment using classification metrics and a visual depiction of accuracy and loss over epochs. In order to fully understand sentiment analysis in precision medicine and to use deep learning techniques in their own projects and research, readers need to be able to create and train learning models for analyzing sentiments in various domains efficiently.

3. Existing Methodologies

The translation of genetic data from the genome into functional proteins—which are essential for controlling a variety of biological processes—is a key process known as gene expression. A gene is first transcribed into messenger RNA (mRNA), which contains exons and introns. This is the first step in the process. The transformation of pre-mRNA into mature mRNA then occurs through RNA processing, which includes crucial processes like splicing and polyadenylation. In contrast to polyadenylation, which cleaves the mRNA and adds a string of adenine bases to its 3' end, splicing comprises the removal of introns and ligation of exons. With the use of statistical, computational, and machine learning methods, it is now possible to predict the actions of these cell factors from genetic sequences, opening the door to a better understanding of their functions in illness and the effects they have on phenotypes and genotypes.

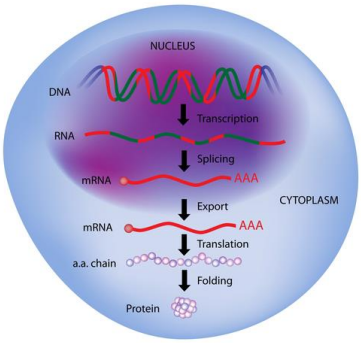


Fig: An overview of the important stages involved in the process of gene expression. Image Copyright: Alila Medical Media, Image ID: 106781666 via Shutterstock.com

3.1 Splicing:

A crucial step in eukaryotic gene expression known as splicing involves joining exons and deleting introns in order to create mature mRNA. The determination of gene structure and genome annotation depends on accurate splice site prediction. The places of splicing are designated by the consensus splice sites GT (donor site) and AG (acceptor site), and this process is aided by a sophisticated molecular apparatus termed the spliceosome during or shortly after transcription. Additionally, alternative splicing (AS), in which certain exons are either included or removed from the pre-mRNA transcript, is a frequent occurrence. A single gene's functional potential is boosted by AS's contribution to the rise in protein isoform variety.

Alternative splicing has wide-ranging effects because it is influenced by the cellular environment and genetic components close to the splice sites. Variations in gene expression levels result from it, and mutations at splice sites can interfere with healthy gene expression, causing a variety of illnesses and disorders. Cystic fibrosis, Parkinsonism, spinal muscular atrophy, myotonic dystrophy, amyotrophic lateral sclerosis, accelerated aging, and a number of cancers have all been linked to splicing anomalies. In order to accurately identify splice sites in genomes, predict protein isoforms resulting from gene sequences, extract significant splicing features and forecast splicing patterns in various cellular contexts, studies of splicing are conducted. By better understanding, the complex dynamics of alternative splicing and improving splice site prediction, computational and machine-learning advancements have advanced our understanding of gene regulation and shed light on the molecular mechanisms underlying splicing-related diseases.

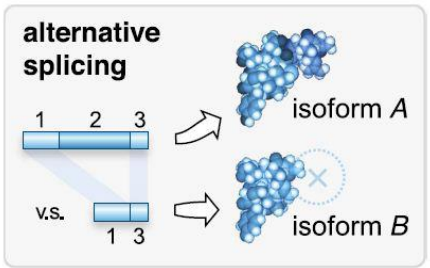


Fig: Gene can generate different proteins by the process of alternative splicing [4]

Splicing difficulties:

Accurately identifying actual splice sites and handling uneven positive and negative data in the genome provide obstacles for predicting splicing patterns. False positives and false negatives in the prediction of splice sites are caused by non-canonical splice sites and the predominance of GT and AG dinucleotides. Additionally, splicing in exonic and intronic cis-regulatory regions is influenced by regulatory mechanisms other than core splice site motifs. It might be challenging to comprehend the relevant splicing characteristics and choose the optimal sequence lengths for experiments. To solve these problems, computational and machine learning models must be created that can predict splice sites with high accuracy while taking regulatory considerations into account. Our knowledge of splicing processes and their significance in gene expression and illness is improved by better splice site prediction.

Models for Splicing problems:

Model splicing for problems to date, the objectives of splicing have been achieved through a number of experimental, statistical, and machine learning-based methodologies. It has been recognized that computational approaches that predict the impact of genetic alterations on splicing offer an exciting alternative to experimental methods since they are faster and have a smaller search space. Threshold scores are arbitrary, it's challenging to decide which tool to employ first, and there aren't any reliable standard interpretation criteria, therefore the computational tools have limitations. Computational and experimental models must work together for the objective to be accomplished.

Machine Learning Methods for Predicting Splicing Regulation:

This table lists the many machine learning methods for predicting splicing regulations, along with their particularities and the information they make use of.

Approach	Novelty	Features
Hidden Markov Model	Capturing higher-order dependencies in genome sequences	Consensus and degeneracy features
Neural Network	Preprocessing of input sequences to capture sequence features via a sliding window of nucleotides	Sequence features
Support Vector Machine	Appropriate and novel kernels for SVM	Positional (most common), compositional and dependency features
Combined models	Choice of better preprocessing Method combined with appropriate classification tool	Depends on the classifier being used
Deep Networks	Deeper networks where hidden variables represent sequence features	Sequence features

Table: Machine learning approaches for predicting the splicing regulations

4. SpliceVec: Distributed Feature Representations for Splice Junction Prediction

4.1 Introduction:

The subject of genomics has advanced significantly in recent years, and splice junction prediction is an important area of study. Splice junctions must be correctly identified and categorized in order to comprehend gene expression and the role of alternative splicing in various biological processes. We got the opportunity to work on SpliceVec during my internship, a cutting-edge technique for predicting splice junctions that use distributed feature representations.

4.2 Methodology:

The recommended method may be divided into two stages: the feature representation stage, which creates a distributed representation for each splice junction using either the word2vec or doc2vec framework, and the classification of splice junctions using MLP. We will discuss each of these in detail in the subsections that follow.

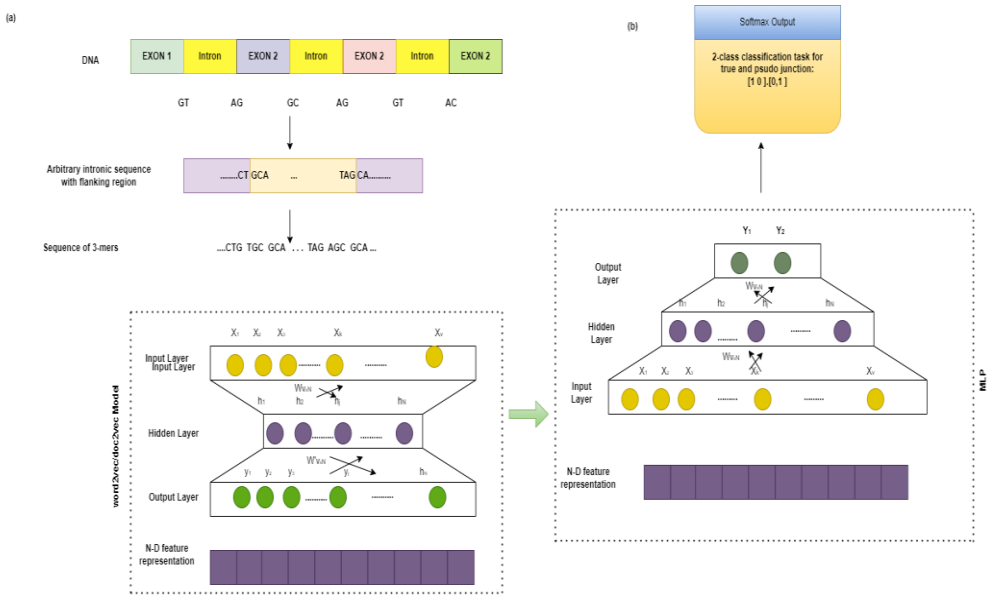


Fig: Proposed approach: (a) feature representation; (b) splice junction classification. [1]

4.2.1 Data:

The most recent version of GENCODE annotation (version 26), which is based on the GRCh38 human genome assembly, was used to extract true and false splice junctions. From protein-coding genes, a total of 294,576 splice junctions were recovered. Introns ranged in length from 1 to 1,240,200 nucleotides (nt). Notably, the human MST1L gene contains the shortest known eukaryotic intron, which is 30 base pairs (bp). Only introns longer than 30 bp were taken into consideration in order to rule out potential sequencing errors, leading to 293,889 splice junctions.

Consensus splice site sequences to make sure they weren't mistakenly categorized as splice junctions, GT, and AG were picked at random when looking for consensus splice site sequences. This selection criterion is required because the GT-AG consensus dinucleotide is present at the donor and acceptor locations of more than 98% of canonical splice junctions. Rather than the 10-300,000 (nt) range utilized in Deep Splice, spurious splice junctions on the same chromosome were examined in this investigation. This series has attempted to mimic the characteristics of actual introns.

4.2.2 Understanding Distributed Representation:

In the field of natural language processing (NLP), vector representations of words or phrases are crucial. Traditional local representations like N-grams and bag-of-words have limitations because of their sparsity and high dimensionality. Distributed representation, on the other hand, has excelled in difficult NLP problems. This type of representation is trained using the relationships and interactions between words that exist in a variety of contexts within a particular corpus.

The Word2vec models developed by Mikolov et al. use shallow neural networks to calculate continuous word vector representations. These models arrange each word in an n-dimensional space and group them together based on their similar syntactic and semantic characteristics. Word2vec provides a continuous bag of words (CBOW) and skip-gram designs. While skip-gram predicts the context words given the current word, CBOW predicts the current word given its surrounding context words.

To further expand this model to texts of various lengths, including phrases, paragraphs, and documents, the doc2vec model was developed. Furthermore, Doc2vec offers two architectures: a distributed bag of words (DBOW) and distributed memory (DM). DBOW uses a method comparable to the word2vec skip-gram architecture to anticipate context words from the document vector. On the other hand, DM predicts the current word based on both adjacent context words and the document vector, similar to the CBOW architecture of word2vec. According to the doc2vec paradigm, word vectors are shared across documents, but document vectors are exclusive to each document and can only be learned in the context of that particular document.

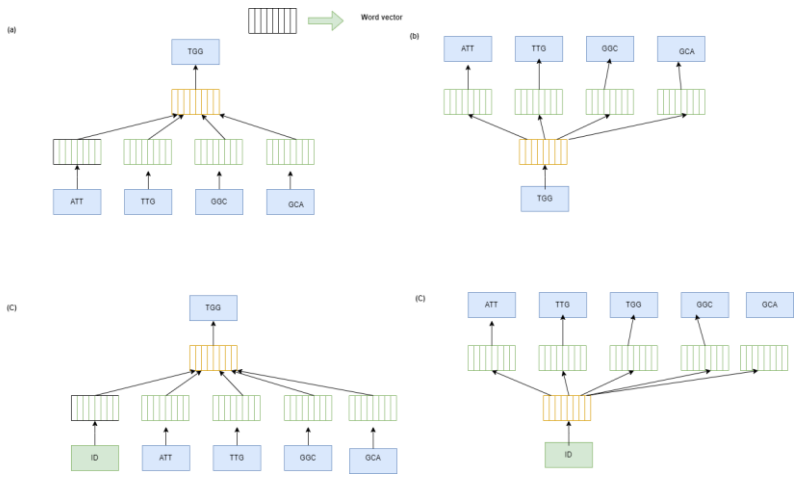


Fig: Architecture of word2vec and doc2vec models. (a) CBOW for word2vec (b) skip-gram for word2vec (c) DM for doc2vec and (d) DBOW for doc2vec [1]

4.3 Implementation of SpliceVec:

Using the knowledge we gained from the literature research, we looked at the current SpliceVec implementation. In order to comprehend how the word2vec and doc2vec models were trained on biological sequences, we looked at the code and its functionalities. Even though we didn't directly use SpliceVec, we nonetheless gained a thorough knowledge of its technological details.

4.4 Results and Discussion:

Using the knowledge we gained from the literature research, we looked at the current SpliceVec implementation. In order to comprehend how the word2vec and doc2vec models were trained on biological sequences, we looked at the code and its functionalities. Even though we didn't directly use SpliceVec, we nonetheless gained a thorough knowledge of its technological details.

4.5 Conclusion:

SpliceVec is a fresh method for splice junction feature representation. In order to record splicing signals close to junctions, it makes use of vector embeddings in an n-dimensional feature space. SpliceVec outperforms prior models in terms of accuracy by utilizing a Multilayer Perceptron (MLP), even with small datasets and unbalanced sample sizes. It has outstanding computational efficiency and is 12.94 times faster than leading models at predicting non-canonical splice junctions. SpliceVec is a useful tool in genomic research because of its adaptation to many species and versatility. It offers fresh perceptions of different splicing mechanisms and has promise for biomedical uses.

5. SpliceVisuL: Visualization of Bidirectional Long Short-term Memory Networks for Splice Junction Prediction

5.1 Introduction:

The knowledge of gene activity, the detection of genetic disorders, and the identification of prospective therapeutic targets all depend on the precise prediction of splice junctions in genomic sequences. Bidirectional long short-term memory networks (BLSTMs) have demonstrated promising results for detecting long-distance relationships and patterns in sequential data. In this brief study, we present "SpliceVisuL," a state-of-the-art method that concentrates on the visualization of bidirectional long short-term memory networks to enhance splice junction prediction.

5.2 Methodology:

Bidirectional Long Short-Term Memory Networks (BLSTMs), a kind of recurrent neural network (RNN), are used by SpliceVisuL to analyze splice junctions. The BLSTM architecture effectively gathers contextual information from the past and future of elements by processing

sequences in both forward and backward orientations. Its capacity to record intricate temporal correlations makes it useful for applications that need sequential data.

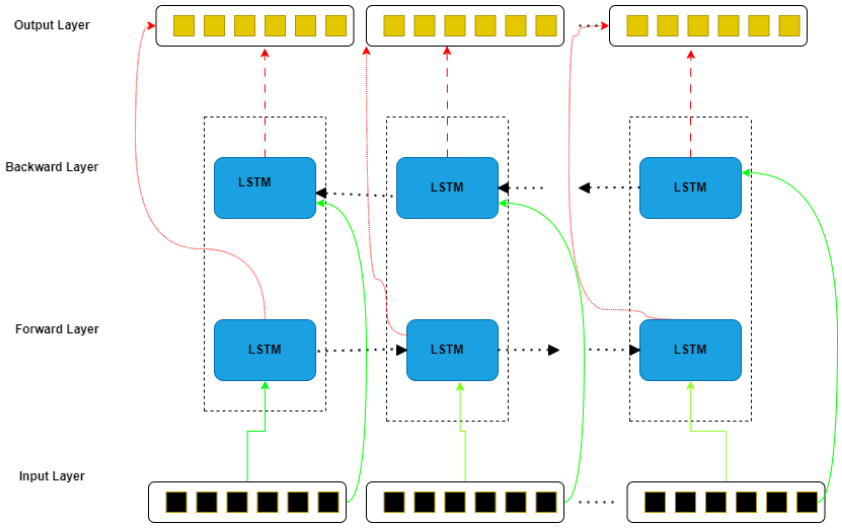


Fig: Bidirectional Long Short-Term Memory Networks (BLSTMs) [2]

5.3 Visualization Technique:

SpliceVisuL suggests a novel visualization method to enhance the interpretability of BLSTM models used for splice junction prediction. Researchers may learn about the inner workings of the model and how genetic sequences are assessed to create predictions by employing this method. Researchers can discover crucial sequence characteristics involved in splice site identification by visualizing the internal representations and hidden states of the BLSTM layers.

5.3.1 Smooth Gradient with Noisy Nucleotide Embeddings:

Sensitivity maps, which are created in image classification tasks using the gradient of unnormalized output probabilities with respect to the input picture, provide information on how even tiny changes in the pixels can have a significant impact on the result. But these sensitivity maps usually contain noise. We employ the notion of smooth gradients to circumvent this by averaging the gradients from several noisy embeddings for each position in the sequence. The inclusion of Gaussian noise with a standard deviation of 0.15 to the nucleotide embeddings produces averaged gradients, also known as smooth gradients, which enable more accurate and distinct identification of crucial spots.

5.3.2 Integrated Gradient with Nucleotide Embeddings:

The creation of sensitivity maps is the same objective of integrated gradients as it is of smooth gradients. Instead of computing a single gradient, integrated gradients calculate the average gradient as the input varies along a linear path from a baseline to the provided input. In our research, we consider 50 steps and compute the average gradient at each point along the series. Our technique is based on the zero-embedding vector, which is

suggested for text inputs. With the use of a visualization technique that meets key attributes of attribution techniques, such as sensitivity and implementation invariance, we can assess the contributions of each location in the sequence to the model's predictions.

5.3.3 Omission of a Single Nucleotide:

We calculate the omission score for each sequence position to determine its importance. This score is determined by contrasting the feature representation for the original sequence obtained from the fully connected layer with that produced when the nucleotide at the specified location is replaced with a neutral element (N). Higher values denote more significance of the relevant sequence location, while the omission score measures the difference between the two representations.

5.3.4 Occlusion of k-mers:

In order to look at variations in the expected output of the model, we omit portions of the sequence. This requires the employment of a sliding window of length l, cantered at position (l + 1)/2, and the conversion of the nucleotides inside the window to N. The resultant deviation value, which was determined by comparing the absolute differences between the model outputs with and without occlusion, is recorded in the window's middle. We propose two occlusion variants: fixed length occlusion, which occludes a specified number of nucleotides (k), and variable length occlusion, which takes into consideration occlusion windows of different lengths. The biggest divergence determines the significance of a genetic area in the variable length occlusion.

5.4 Experimental Setup:

Using benchmark splice junction datasets, studies were carried out to gauge SpliceVisuL's efficacy. On these datasets, the BLSTM models were trained, and the visualization method was used to comprehend the model's behavior when making predictions. To show how the BLSTM's decision-making can be interpreted, we must use carefully chosen sample instances.

5.5 Results and Discussion:

According to the studies, SpliceVisuL improves the BLSTM models' ability to anticipate splice junctions while also making them easier to understand. We may acquire an understanding of the model's decision-making process and identify sequence elements that greatly aid in splice site detection by visualizing the hidden states and internal representations. Researchers are now better able to validate and decipher model predictions because of this enhanced understanding.

Researchers in genetics and biology might benefit greatly from the visualization of BLSTM models with SpliceVisuL. By assisting in the discovery of significant genomic sequence patterns connected to splice junctions, the method improves understanding of gene function and hereditary disorders.

5.6 Conclusion:

An innovative method to improve the interpretability of Bidirectional Long Short-term Memory Networks for splice junction prediction is presented by SpliceVisuL. The approach finds crucial sequence elements linked to splice site identification and offers insightful information into the model's decision-making process by visualizing the internal representations and hidden states. This development advances knowledge of the control of gene expression and gene function.

6. Bibliography

[1] Aparajita Dutta, Tushar Dubey, Singh, Kusum Kumari Singh, Ashish Anand. "SpliceVec: Distributed Feature Representations for Splice Junction Prediction." Department of CSE, Indian Institute of Technology, Guwahati, India; Department of BSBE, Indian Institute of Technology, Guwahati, India.

[2] Aparajita Dutta, Aman Dalmia, Athul R, Kusum Kumari Singh, Ashish Anand, "SpliceVisuL: Visualization of Bidirectional Long Short-term Memory Networks for Splice Junction Prediction." Department of CSE, Indian Institute of Technology, Guwahati, India; Department of EEE, Indian Institute of Technology, Guwahati, India; Department of BSBE, Indian Institute of Technology, Guwahati, India.

[3] S M A Care Series. The Genetics of Spinal Muscular Atrophy SMA CARE SERIES a-The Genetics of Spinal Muscular Atrophy.

[4] Claudia Ben-Dov, Britta Hartmann, Josefin Lundgren, and Juan Valc´arcel. Genome-wide analysis of alternative pre-mrna splicing. Journal of Biological Chemistry, 283(3):1229–1233, 2008.

[5] Jindan An, Xiaodong Zhu, Hongwei Wang, and Xiudong Jin. A dynamic interplay between alternative polyadenylation and microRNA regulation: implications for cancer. International journal of Oncology, 43(4):995–1001, 2013.

[6] Michael KK Leung, Andrew Delong, Babak Alipanahi, and Brendan J Frey. Machine learning in genomic medicine: A review of computational problems and data sets. Proceedings of the IEEE, 104(1):176–197, 2016.

[7] Corinne Vigouroux and Gis`ele Bonne. Laminopathies: One gene, two proteins, five diseases. Nuclear Envelope Dynamics in Embryos and Somatic Cells, pages 153–172, 2003.

[8] Jon McClellan and Mary-Claire King. Genetic heterogeneity in human disease. Cell, 141(2):210–217, 2010.

[9] Neelam Goel, Shailendra Singh, and Trilok Chand Aseri. An improved method for splice site prediction in DNA sequences using support vector machines. Procedia Computer Science, 57:358–367, 2015.

[10] Neeraj Sharma, Patrick R Sosnay, Anabela S Ramalho, Christopher Douville, Arianna Franca, Laura B Gottschalk, Jeenah Park, Melissa Lee, Briana Vecchio-Pagan, Karen S Raraigh, et al. Experimental assessment of splicing variants using expression minigenes and comparison with in silico predictions. Human mutation, 35(10):1249–1259, 2014.