Novel Approaches for Effective Classification of DNA Sequences using Deep Learning Techniques

Submitted To

Dr. paresh saxena

In fulfillment of the course requirements in BITS F464 Machine Learning

Submitted By:

Name	**BITS ID**
Shireen Shaikh	2021B1A43007G
Mridul Mishra	2021B5A42367G

Birla Institute of Technology and Science, Hyderabad Campus, India

1. <u>Introduction:</u>

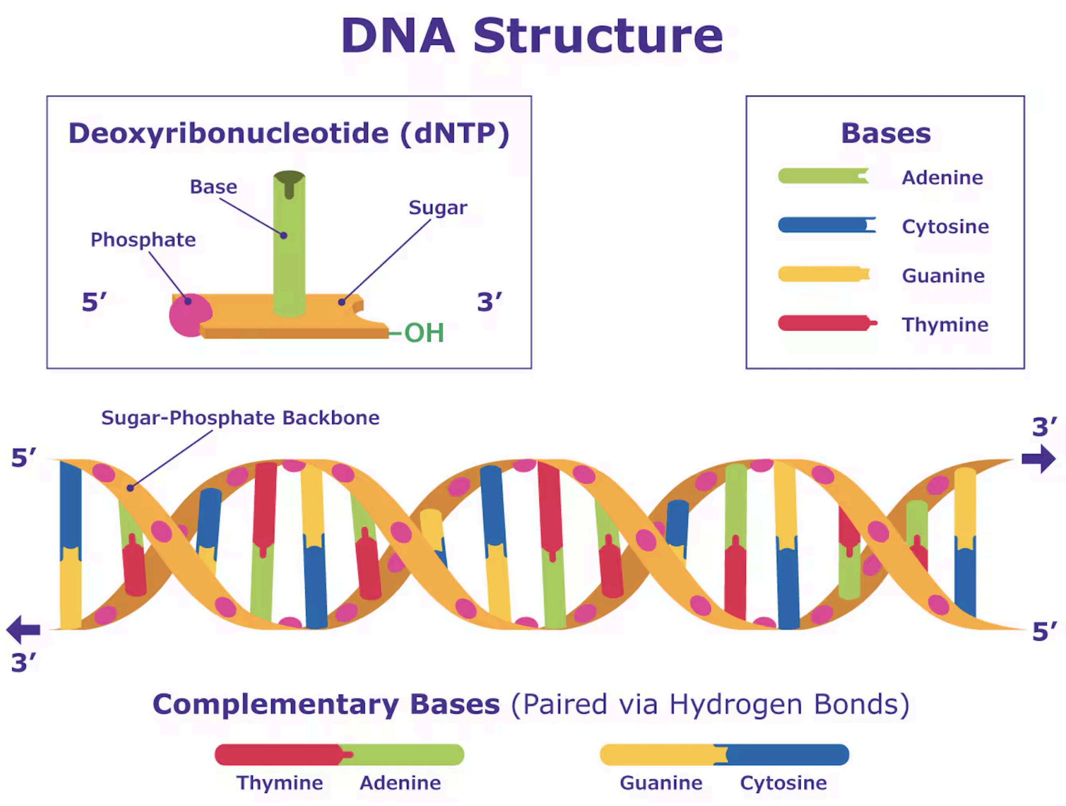


Fig.1 :- A Schematic of DNA Sequence

Despite advancements in genomics, accurate DNA sequence classification remains challenging due to the complex nature of genetic data. Traditional models often fail to encapsulate intricate patterns and nuances of gene sequence, leading to suboptimal performance. This project aims to enhance DNA classification accuracy by developing and comparing neural network architectures using deep learning techniques to explore ways of more optimal genetic analysis and understanding.

DNA sequences are essential for understanding organisms. DNA, or deoxyribonucleic acid, is the genetic blueprint for all living beings, guiding their development, functioning, growth, and reproduction. It contains hereditary information passed from parents to offspring, ensuring the continuity of genetic traits. DNA sequences are composed of four nucleotide bases: adenine (A), thymine (T), guanine (G), and cytosine (C). Advances in sequencing technologies, such as Next-Generation Sequencing (NGS), Third-Generation Sequencing, and CRISPR-based techniques, have vastly increased available DNA data. This abundance of data has sparked interest among researchers and scientists in classifying these sequences.

Our project report starts with a comprehensive view over 2 selected papers followed by an implementation of the methodology discussed in one of the papers. Further, we have researched upon a novel method and have showcased the implementation.

1.1 Literature review:-

Review of the Paper "DNA Sequence Classification by Convolutional Neural Network" by Nguyen Ngoc Giang et al.

Deep Learning is a branch of artificial intelligence that uses neural networks with multiple layers to analyze and interpret large datasets. By mimicking the human brain's information processing, deep learning models can identify complex patterns and relationships within the data. Previous studies have employed deep learning models, such as boosting and deep neural networks, to predict protein disorder and splicing patterns in tissues, achieving high accuracy and performance.

In this paper, it was proposed that DNA sequences themselves could be used as an input to neural networks by converting the textual information into numerical form. This process involved transforming the continuous sequence of DNA into words and then using one-hot vector representation, where a concatenation of binary word vectors represents the text. This input was then fed into a model utilizing CNN (Convolutional Neural Networks). CNNs use convolutional layers to analyze input data by applying filters or kernels to create feature maps, capturing spatial patterns and features like edges, textures, and shapes. Each filter slides over the input data, performing a convolution operation to form a hierarchical feature representation. Subsampling techniques reduce dimensionality while retaining information, and fully connected layers link every neuron in one layer to every neuron in the next, allowing for feature extraction. SoftMax converts values into a probability distribution, ensuring the output probabilities sum to 1, with each output neuron corresponding to a class.

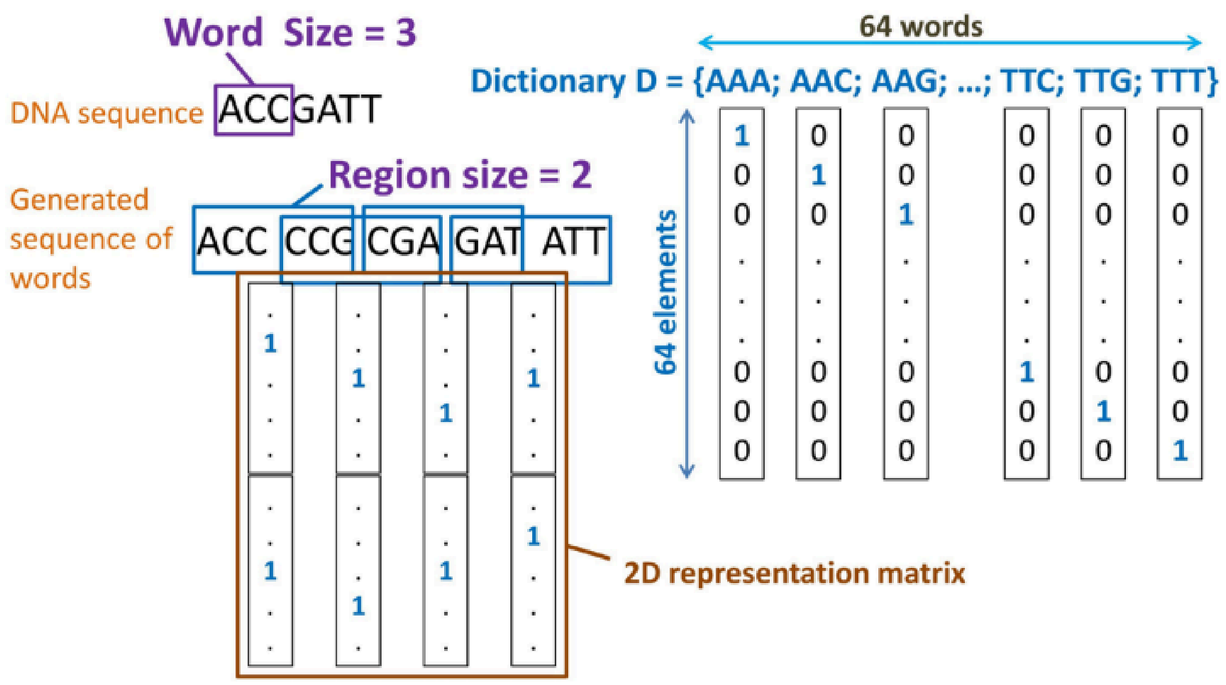


Fig.2 :- One Hot Vector Representation

The evaluation in the research paper utilized 12 datasets of DNA sequences, each containing sequences of 500 base pairs categorized into "Positive" (regions wrapping around histone proteins) and "Negative" classes. The Model architecture consisted of two convolutional layers, each followed by subsampling layers, then a fully connected layer with 100 neurons and a dropout value of 0.5. Finally, a SoftMax layer was used to predict the labels.

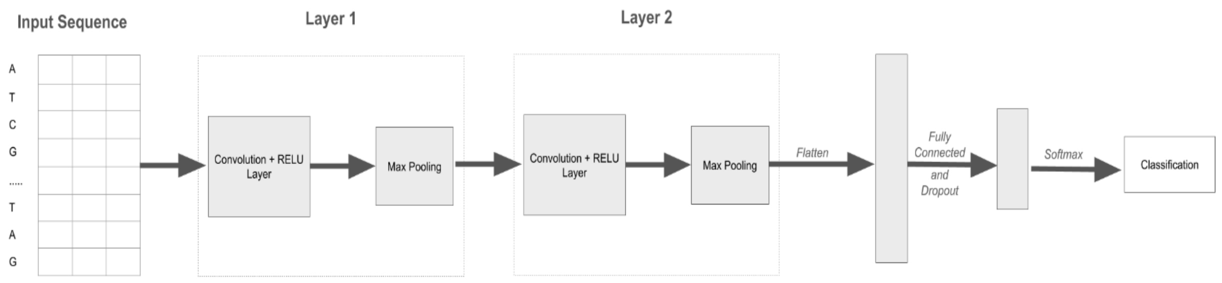


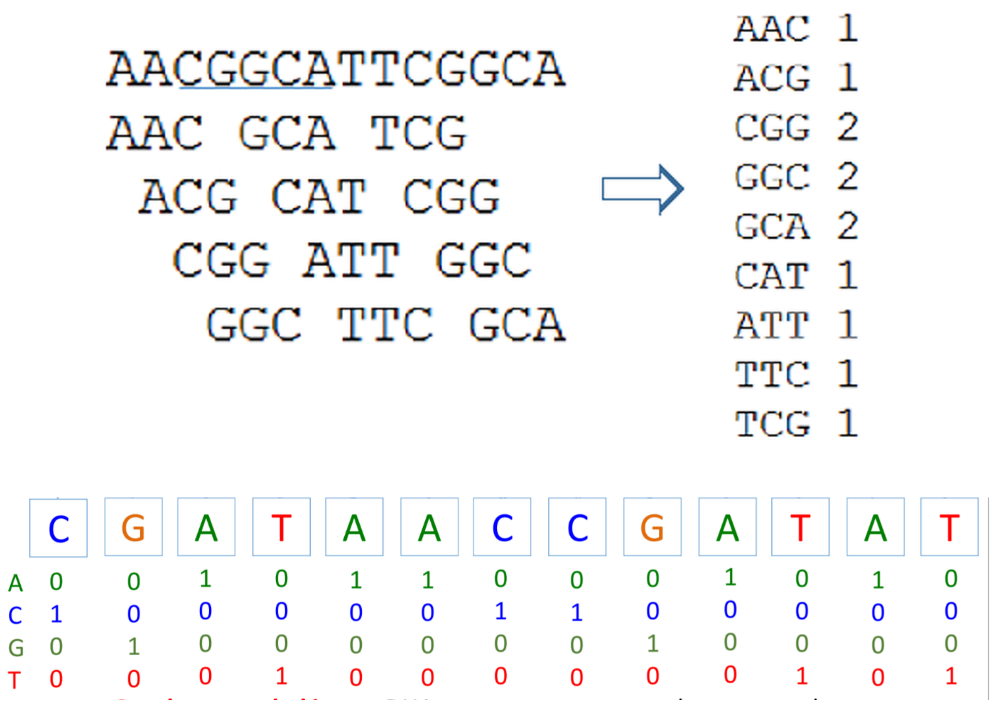
Fig.3 :- Visualization of the Model Architecture

The proposed model showed significant improvements across all validation datasets, with accuracy increases ranging from nearly 1% to over 6%, outperforming previous approaches. The study highlights the potential of CNNs in bioinformatics, demonstrating their ability to tackle sequence-related challenges and accurately classify DNA sequences for advancements in genetic research and understanding.

Review of the Paper "DNA Sequencing using Machine Learning and Deep Learning Algorithms" by Varada Venkata Sai Dileep et al.

DNA sequencing is causing a revolution in genomics giving deep looks into how genetic information is processed and made sense out of. Yet the intricacy and the sheer vast amount of nucleotide data pose big barriers to analyzing and understanding the data, one of the major issues is that the whole process is very time-consuming. For an estimate of the sheer enormity of the nucleotide data - there are nearly Three billion counts of nucleotides in the human genome alone!. Machine learning and deep learning models have become strong tools to predict and analyze genomic data, providing better correctness and simultaneously proving more effective as compared to the old ways. Major motivation of this research is to quantify and compare the accuracy scores of five different deep-learning models on a single dataset (to avoid diversification of the problem already at hand thus to keep things simple!).

When using CNN, it is not possible to simply input the raw text data of the DNA sequence nucleotides for feature extraction and classification. Therefore it first must be converted to a numerical form. A dictionary can be used to store the words as values and numbers as respective keys. However, one more problem arises! We cannot treat the DNA sequence like regular textual data, unlike regular texts it has no words, only a single 100 character entity with no spaces. K-mer counting comes to help to encode characters into words and then those words can be converted into numbers using one-hot encoding.



AACGGCATTCGGCA
AAC GCA TCG
ACG CAT CGG
CGG ATT GGC
GGC TTC GCA
AAC 1
ACG 1
CGG 2
GGC 2
GCA 2
CAT 1
ATT 1
TTC 1
TCG 1
C G A T A A C C G A T A T
A 0 0 1 0 1 1 0 0 0 1 0 1 0
C 1 0 0 0 0 0 1 1 0 0 0 0 0
G 0 1 0 0 0 0 0 0 1 0 0 0 0
T 0 0 0 1 0 0 0 0 0 0 1 0 1

Fig.4 :- 3-mer counting(top) and One-hot-encoding(bottom)

Machine Learning algorithms like decision trees, random forests, and Naïve Bayes algorithms have been used extensively to study genomics by analyzing the DNA sequencing data. Decision uses a top to bottom approach where we start from the root node and split data based on Gini impurity with the goal of maximizing the information gain. While this method is intuitive and based on logic it is prone to one of the several devils of machine learning - overfitting!. Random forest method handles this issue by creating an ensemble of decision trees, thereby enhancing robustness and reducing variance.

A different approach which groups data based on probability of occurrence is Naïve Bayes, based on Bayes' theorem. It offers simplicity and efficiency in classification tasks but only at the cost of heavy time-consumption. In this article despite its strong assumptions of feature independence, it has shown remarkable accuracy in DNA sequencing, achieving 98.00% accuracy with k-mer encoding. This highlights its potential in handling the probabilistic nature of genomic data.

Deep learning models, particularly Convolutional Neural Networks and Long Short-Term Memory, have come to the fore in genomic data analysis. The ability of CNN to capture spatial hierarchies using convolutional layers fits perfectly with the feature extraction tasks from DNA sequences. Various encoding techniques such as k-mer encoding converts sequence into words, enhances CNN performance by providing data in a structured manner. Oftenly two deep learning models can be clubbed together to provide better predictions such as CNN and Bidirectional LSTM, Which considers dependencies in both forward and backward direction. Transform Learning works in a similar way; it combines results from multiple neural networks, thus integrating the strengths of different models and creating a robust ensemble that duly outperformed the traditional CNN models.

The inherently large size of data used for DNA sequencing makes the whole training process and data exploration time consuming and very intensive computationally. Not to mention the highly complex DNA sequence which takes highly sophisticated encoding algorithms and techniques to parse the data so that the model can read it, train on it and generate accurate predictions. CNN's and bidirectional LSTMs have proven to be effective in DNA sequencing via Deep learning. As the field of DNA sequencing and bioinformatics in general evolve in the future, refining encoding techniques and enhancing model efficiency will be crucial in unlocking the full potential of genomic data analysis.

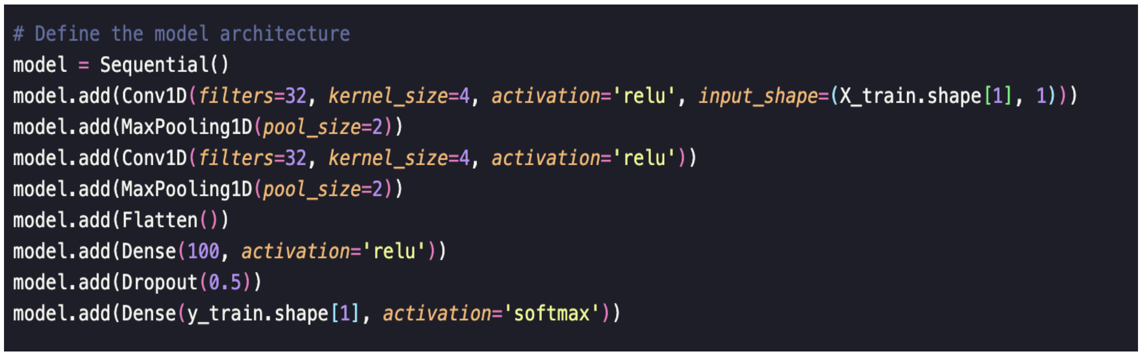
2. Implementation:-

The paper selected for the implementation is "DNA Sequence Classification by Convolutional Neural Network" by Nguyen Ngoc Giang et al. . Below are the steps described for the implementation:

1. **Exploratory Data Analysis:** Out of the 12 Datasets, we selected the **H3K4me1 dataset** for our work. The dataset was in a text file where every entry was made in 3 lines where - first line indicated the 'id' , second line indicated the 'sequence of dna', third line indicated the 'class'. The sequence is classified as '1' if the DNA wraps around histone and '0' if it doesn't. We found 17,266 sequences that were classified as '1' and 14,411 sequences classified as '0'. An important criteria to maintain uniformity in the dataset is the base length that indicates the no of nucleotides present in the sequence. Through statistical analysis, we found that the base length had a standard deviation of 1.591140, implying that the base lengths were not equal. We proceeded to find 2 outliers and eliminated them. Additionally, to aid the process of hyperparameter optimization, we have split the data set into Validation data set and Training data set.

2. **Sequence Representation and Embedding:** We followed the 3-mer representation where 3 letters of the sequence are considered at a given time. This gives 64 possible representations from 4 nucleotides. We then followed a one-hot encoding method which represents a single 3-mer using a vector of size 64 of values 0 and 1.

3. **Model architecture and the parameters:** We have built the model following the architecture given in the paper. A step by step procedure is explained below and also explained with a code snippet:-



```
# Define the model architecture
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=4, activation='relu', input_shape=(X_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=32, kernel_size=4, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(y_train.shape[1], activation='softmax'))
```

Fig.5 :- Base CNN model Architecture

a) Conv1D Layer:- For the base model, we have implemented a 2 layered CNN. For each layer we have used 32 filters, and a kernel of size 4 and in the end ReLU activation function. Filters represent the number of output channels after convolution has been performed, while Kernel represents the size of a convolution filter being used to perform convolution on the image.

b) 1D Max-pooling Layer:- After each layer we have used a 1D Max Pooling layer Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map. We have trained the model for a pool size of 2.

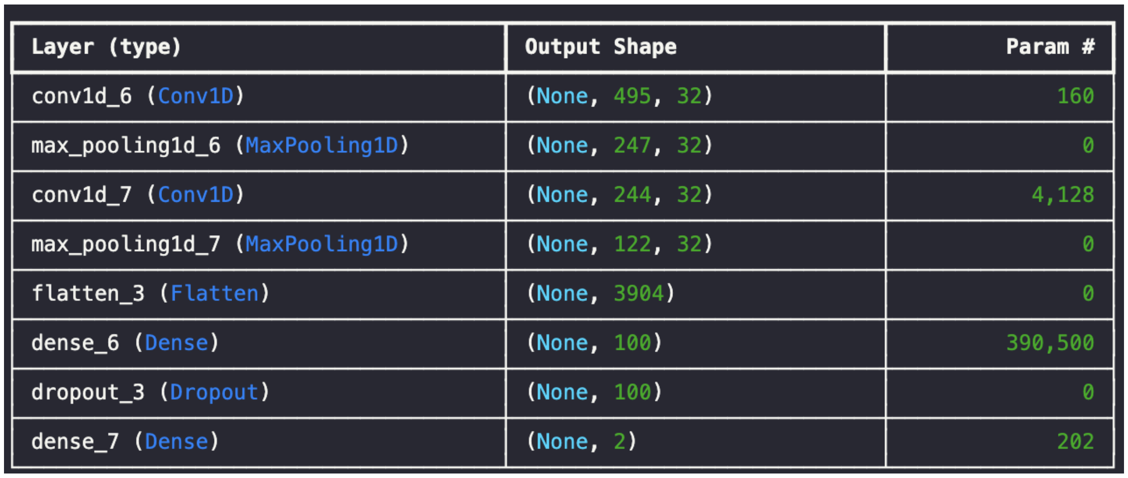
c) Dropout and Flatten Layer:- Flatten layers are used to transform a multidimensional output into linear to pass it onto a Dense layer.

d) Dense layer:- The dense layer is a simple Layer of neurons in which each neuron receives input from all the neurons of the previous layer, thus called as dense**.** This layer is activated using a ReLU activation function and the output is obtained.

e) Dropout layer:- Dropout is a way of cutting too much association among features by dropping the weights (edges) at a probability. This is followed by another Dense Layer.

f) Dense Layer number 2:- This is the last layer in the network and output is obtained after this. This is activated by a Softmax Activation function.

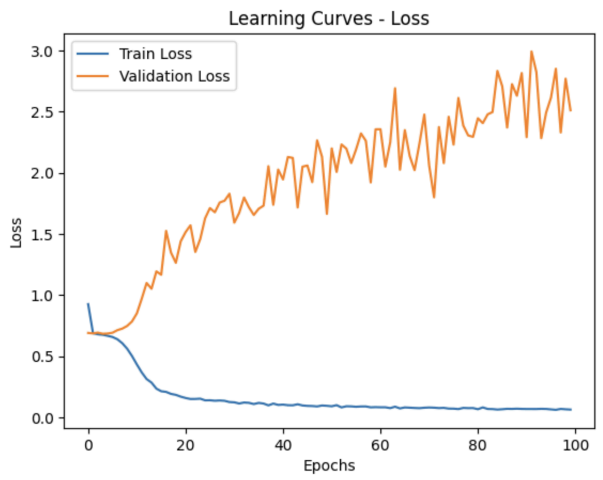
g) Optimization:- ADAM (Adaptive moment estimation) is an iterative optimiser which is used to minimize the loss function (here, at a learning rate of 0.0001). As we are essentially solving a classification problem, and using a softmax activation function for the outputs, using a categorical cross entropy loss is a good practice. This optimiser (ADAM) is an extension of the STOCHASTIC GRADIENT DESCENT algorithm which is used to update the weights during training.

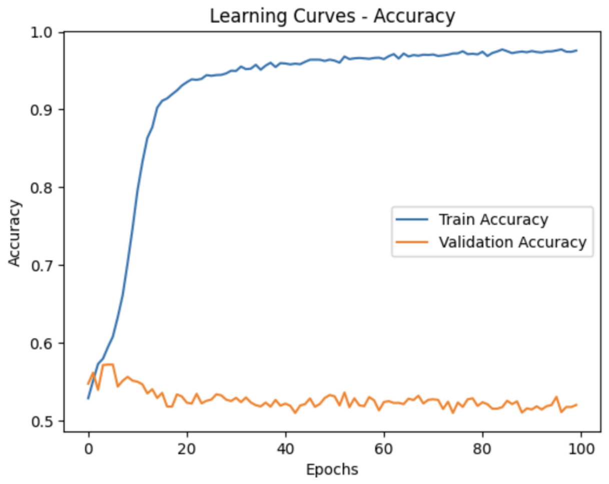


Layer (type)	Output Shape	Param #
conv1d_6 (Conv1D)	(None, 495, 32)	160
max_pooling1d_6 (MaxPooling1D)	(None, 247, 32)	0
conv1d_7 (Conv1D)	(None, 244, 32)	4,128
max_pooling1d_7 (MaxPooling1D)	(None, 122, 32)	0
flatten_3 (Flatten)	(None, 3904)	0
dense_6 (Dense)	(None, 100)	390,500
dropout_3 (Dropout)	(None, 100)	0
dense_7 (Dense)	(None, 2)	202

Fig.6 :- A description of the Architecture

This model is trained for 100 epochs and after training [Train, Test and validation] accuracy scores as well as loss is calculated. Finally Learning/Loss curves are plotted along with the ROC curve to show the comparison between True positive classification rate and False Positive Classification rate. ROC curve has been plotted only for this base model to show how good or bad the model performs in comparison to a random classifier (shown by dotted line in the ROC curve). The results are presented below.





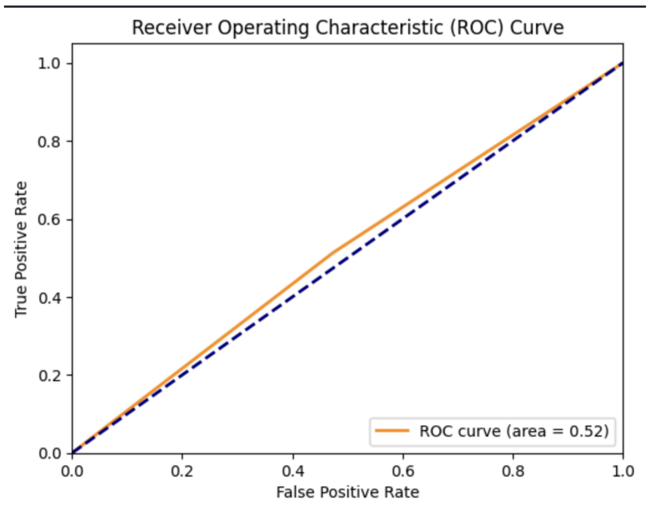


Fig.7 :- Accuracy and Loss Learning curves for training and Validation data and ROC (receiver operating characteristic) curve.

	MAX Accuracy	MAX Loss	AVG Accuracy	AVG Loss	MIN Accuracy	MIN Loss
Train	0.99	1.86	0.53	2.52	0.516	0.0023
Validation	0.57	2.99	0.52	2.51	0.51	0.68

Table showing the Max, Min and Avg Loss and Accuracy Scores

3. Inferences:-

A classical case of overfitting can be seen towards the end of the training process. This can be inferred from the curves of the Learning accuracy of Training and Validation set. Here the model has completely memorized the training data, giving accuracy scores as high as 99% on the training set whereas accuracy on the validation set is as low as 51%. Therefore to prevent overfitting from happening several methods and techniques have been explored in the NOVELTY section of this report with the model implementation using those techniques as well.

4. Novelty and the Improvements made:-

We identified three broad areas in the paper where a novel approach could be implemented. These are:

1. Sequence Representation
2. Sequence Embedding
3. Model Architecture

Additionally, our paper only mentioned a general architecture and thus we performed rigorous experimentations. Through an extensive literature search, we propose the following:

- (Sequence Representation) **Utilize a 4 mer representation of the DNA:-**

  While a 3-mer representation might be beneficial for reducing computational load, 4-mers capture more specific patterns and can represent more unique combinations compared to 3-mers. There are 44=2564^4 = 25644=256 possible 4-mers, while there are 43=644^3 = 6443=64 possible 3-mers. 4-mers can potentially improve model performance by providing richer feature representations, leading to better differentiation between classes. On the downside, 4-mers increase the complexity of the model as there are more unique k-mers to handle. This can lead to higher computational requirements for training and memory usage.

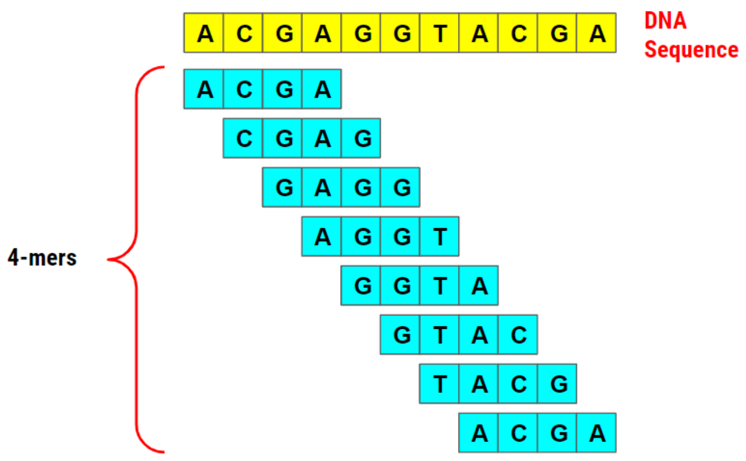


Fig.8 :- 4-mer counting scheme

Our sequences are relatively long (500 bases), allowing the capture of more specific patterns with 4-mers. Additionally, we have a substantial amount of data (17,000 sequences labeled as 1 and 12,000 as 0), so the model can handle the increased complexity.

- (Sequence Embedding) **Utilize a Word2Vec method for sequence embedding:-**

  Utilizing the Word2Vec method for sequence embedding involves treating k-mers (subsequences of length k) as "words" and DNA sequences as "sentences." The first step is to generate k-mers from the DNA sequences, creating overlapping subsequences that capture local sequence context. For example, a sequence "ATGCGT" with k=4 would yield k-mers "ATGC," "TGCG," and "GCGT."

  These k-mers are then used to train a Word2Vec model, which learns to represent each k-mer as a dense vector in a high-dimensional space. The resulting embeddings capture the semantic relationships between k-mers based on their co-occurrence patterns in the sequences, providing a meaningful representation of the DNA sequence that can be used as input for machine learning models. This method leverages the power of Word2Vec to capture biological sequence features and improve the performance of downstream tasks such as classification or clustering.

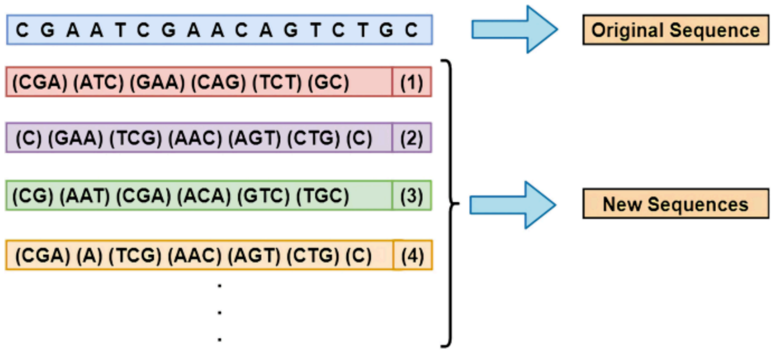

C G A A T C G A A C A G T C T G C
Original Sequence
(CGA) (ATC) (GAA) (CAG) (TCT) (GC) (1)
(C) (GAA) (TCG) (AAC) (AGT) (CTG) (C) (2)
(CG) (AAT) (CGA) (ACA) (GTC) (TGC) (3)
(CGA) (A) (TCG) (AAC) (AGT) (CTG) (C) (4)
New Sequences

Fig.9 :- Word2Vec encoding scheme

*(As a side note, we also experimented with BERT, fasttext and GNN but they did not improve our performance.)

- (Model Representation) **Utilize Hybrid Models instead of CNN:-**

  While CNN provides a good base for sequence classification, a combination of CNN with other models has been proven to perform better. The advantages could be as follows:

  Combining Strengths: Hybrid models can integrate the strengths of different types of neural networks. For instance, CNNs are excellent at extracting local features and patterns through convolutional layers, while Recurrent Neural Networks (RNNs) or Long Short-Term Memory networks (LSTMs) are proficient at capturing sequential dependencies and long-term relationships in the data. Combining these can lead to a more comprehensive understanding of the sequences.

Better Handling of Sequential Data: DNA sequences are inherently sequential, and models like LSTMs or Bidirectional LSTMs can effectively handle such data by learning dependencies between nucleotides that are far apart. This capability is often crucial for understanding biological functions that rely on distant sequence motifs.

Improved Feature Representation: Hybrid models can capture both local features (via CNN layers) and long-range dependencies (via RNN or LSTM layers). This dual capability can lead to richer and more informative feature representations, which can enhance model performance.

Enhanced Generalization: By leveraging multiple types of layers, hybrid models can potentially generalize better to new, unseen data. Each component of the hybrid model may mitigate the weaknesses of the others, leading to more robust predictions.

Adaptability to Complex Biological Data: Biological sequences often exhibit complex patterns that might not be fully captured by a single type of model. Hybrid models can adapt more effectively to these complexities, potentially improving accuracy and insight.

Mitigation of Overfitting: Hybrid models can incorporate various regularization techniques across different types of layers, helping to prevent overfitting, especially when working with large and complex datasets.

- **(Model Representation) Utilize LSTM:-**

Long Short-Term Memory networks (LSTMs) are a specialized type of recurrent neural network (RNN) designed to address the challenges of learning and remembering long-term dependencies within sequential data, which is particularly crucial in genomics and bioinformatics.

In genomics, LSTMs are preferred due to their ability to effectively capture the complex temporal relationships inherent in DNA and RNA sequences. Unlike traditional RNNs, LSTMs incorporate a memory cell that allows them to maintain information over long sequences, mitigating the vanishing gradient problem that often hinders the training of deeper networks on sequential data. This capability is essential when analyzing DNA sequences, which can span thousands to millions of base pairs, and where
The interactions between distant base pairs can influence gene regulation, protein folding, and disease mechanisms.

Long Short-Term Memory networks (LSTMs) offer several key advantages in genomics and bioinformatics. They excel at capturing long-term dependencies within sequential data, making them ideal for modeling complex relationships in DNA and RNA sequences. Unlike traditional recurrent neural networks (RNNs), LSTMs mitigate the vanishing gradient problem, allowing them to effectively learn from sequences of varying lengths without the need for fixed-size inputs. This capability is crucial in genomics, where sequences can span thousands to millions of base pairs. LSTMs automatically extract relevant features from genetic data, such as motifs and patterns, without requiring manual feature engineering. Moreover, LSTMs can be integrated into hybrid models with other architectures like CNNs, leveraging their complementary strengths in feature extraction and sequence modeling. These advantages make LSTMs indispensable in genomics, driving advancements in gene prediction, sequence classification, protein structure prediction, and personalized medicine.

- (Model Representation) **Utilize BiLSTM:-**

  Bidirectional Long Short-Term Memory networks (BiLSTMs) enhance the capabilities of standard LSTMs by processing data in both forward and backward directions, capturing context from past and future states for a more comprehensive understanding of sequential data. This dual processing approach is especially beneficial in genomics and bioinformatics, where the order of nucleotides can influence biological functions. BiLSTMs excel at identifying complex patterns and dependencies within DNA or RNA sequences, improving the modeling of biological phenomena such as gene regulation and alternative splicing. By integrating information from both directions, BiLSTMs enhance predictive accuracy in tasks like sequence classification, where understanding both upstream and downstream contexts is crucial. Additionally, BiLSTMs can complement other deep learning architectures, enabling the development of hybrid models that leverage their bidirectional context-awareness alongside the feature extraction capabilities of CNNs.

  The following section dives deeper into the implementation of our novelties.

NOVELTY - 1

We first fine tuned the base model (CNN+One hot encoding and 3-mer representation) and compared the results. Following describes our changes:

Adding CNN Layer:- With the increase in the number of layers, the features extracted will be more specific. Effect of this can be directly seen in the results discussion section. Along with adding one more CNN layer we have also increased the number of filters and kernels.

L2 regularizer:- To prevent overfitting as seen in the previous case, using a regularization to penalize the weights themselves will be beneficial. Therefore as a step towards improvement we have used L2 regularizer.

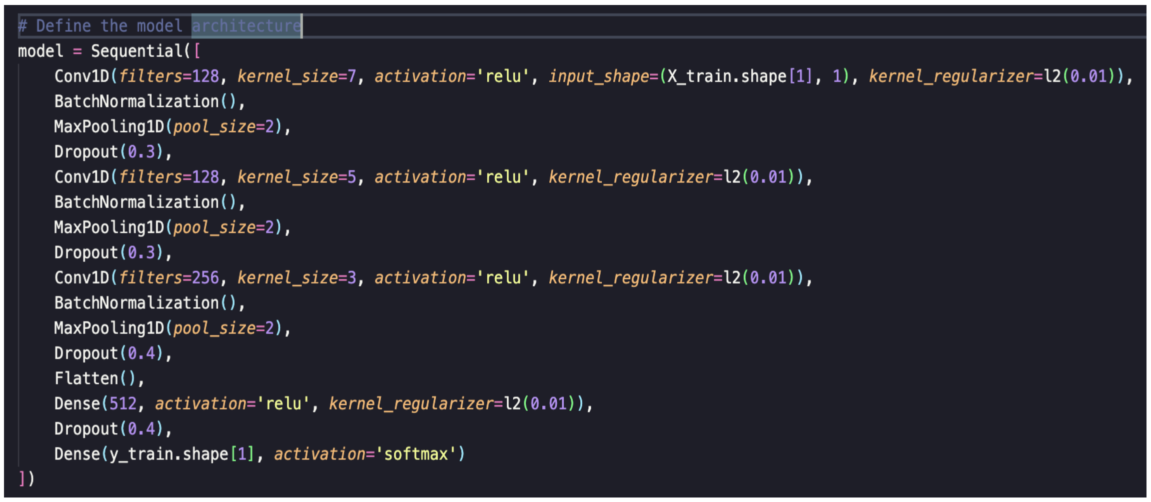
Another change we have made that seems to have an effect on the accuracy is adding a max pooling layer and dropout layer after every cnn layer.

Callbacks:- One of the main issues during training is overfitting, where the model starts to memorize the data set itself. For this different call back techniques which essentially monitor the validation loss can be used such as:-

Early-Stopping technique:- Paired with a model checkpoint is used which monitors the validation loss at each epoch and stops the training process when the loss stops improving(reducing).

ReduceLR on plateau:- Models often benefit from a reduction by a factor of 2-10 in learning rate once the learning stagnates. This Call back monitors the validation loss and if no improvement is seen for a certain 'patience' number of epochs, the learning rate is reduced.

Optimizing hyperparameters:- To optimize the model we did hyperparameter tuning of parameters such as the learning rate, number of epochs and number of filters in the convolutional layer. Model training was done for 100 epochs. The results obtained showed an improvement in the performance of the model but it is still far from being satisfactory.



```
# Define the model architecture
model = Sequential([
    Conv1D(filters=128, kernel_size=7, activation='relu', input_shape=(X_train.shape[1], 1), kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.3),
    Conv1D(filters=128, kernel_size=5, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.3),
    Conv1D(filters=256, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.4),
    Flatten(),
    Dense(512, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.4),
    Dense(y_train.shape[1], activation='softmax')
])
```

Fig.10 :- CNN+One hot encoding and 3-mer representation, hyperparameter tuned

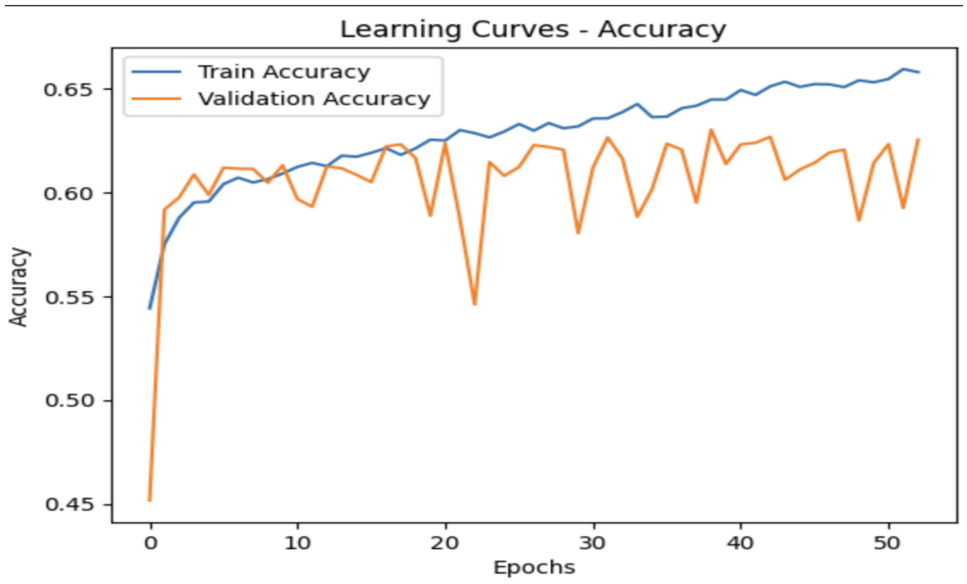
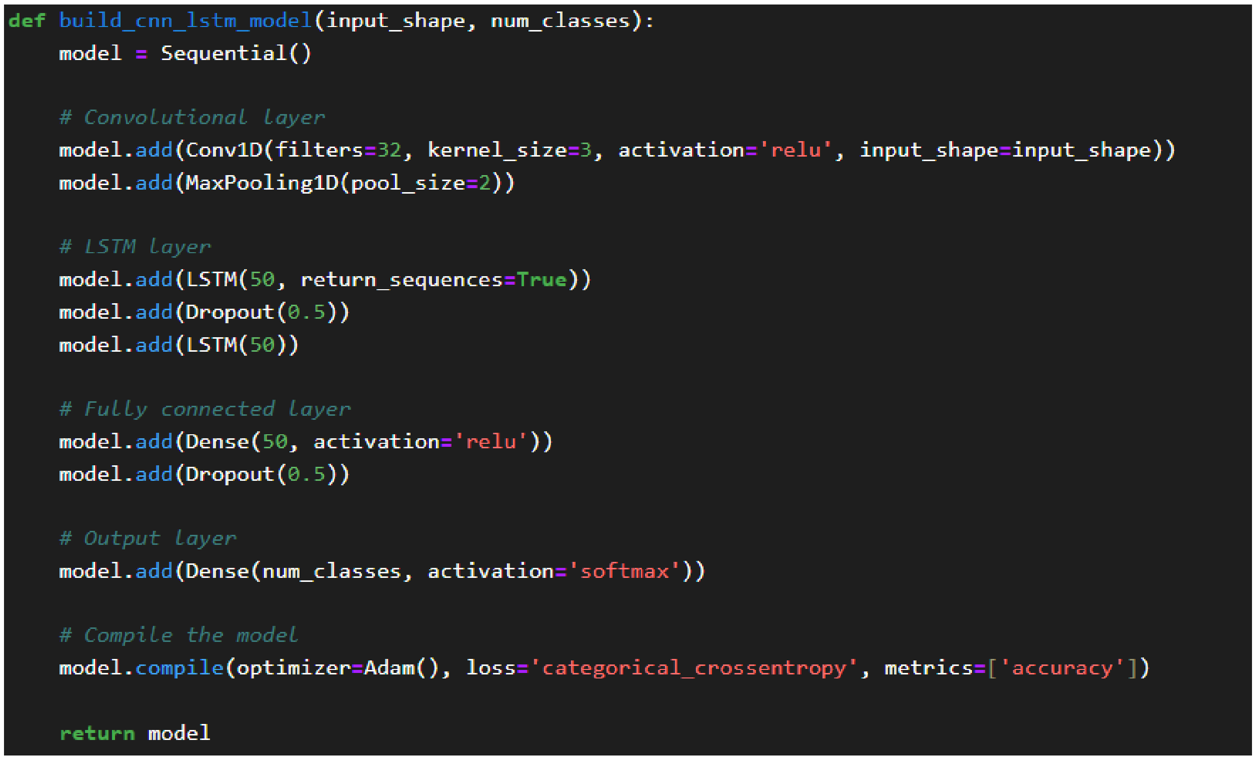


Fig.11 :- Modified CNN Architecture

Results: Best performance was at an accuracy of **64%**.

NOVELTY - 2

The second model utilizes a 4-mer representation along with Word2Vec for sequence embedding. These sequences were then fed into a hybrid model of CNN and LSTM.



```python
def build_cnn_lstm_model(input_shape, num_classes):
    model = Sequential()

    # Convolutional layer
    model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=input_shape))
    model.add(MaxPooling1D(pool_size=2))

    # LSTM layer
    model.add(LSTM(50, return_sequences=True))
    model.add(Dropout(0.5))
    model.add(LSTM(50))

    # Fully connected layer
    model.add(Dense(50, activation='relu'))
    model.add(Dropout(0.5))

    # Output layer
    model.add(Dense(num_classes, activation='softmax'))

    # Compile the model
    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

Fig.12 :- CNN + LSTM Architecture

Model Architecture: The model begins with a convolutional layer that applies 32 filters of size 3 to the input data, capturing local patterns such as DNA motifs by sliding these filters across the input. The ReLU activation function is used in this layer to introduce non-linearity. A max pooling layer with a pool size of 2 follows, reducing the dimensionality of the feature maps and retaining key features while lowering computational complexity.

Next, the model incorporates two LSTM layers. The first LSTM layer, containing 50 units, is set to return sequences, allowing its outputs to be passed as sequences to the next LSTM layer. This setup helps capture long-range dependencies and temporal patterns in the sequence data, which is crucial for understanding the sequential nature of DNA or RNA. A dropout layer with a rate of 0.5 is added after the first LSTM layer to prevent overfitting by randomly setting half of the input units to zero during training. The second LSTM layer, also with 50 units, further processes the sequential data and produces a fixed-length vector.

Following the LSTM layers, the model includes a fully connected dense layer with 50 units and ReLU activation, which integrates the features extracted by the previous layers into a higher-level representation. Another dropout layer with a rate of 0.5 is applied here to further mitigate overfitting.

Finally, the model concludes with an output dense layer that has units corresponding to the number of classes, using the softmax activation function to generate class probabilities. The model is compiled using the Adam optimizer, which adjusts the learning rate during training for quicker convergence, and the categorical cross entropy loss function.

Model ran for 100 epochs, with a batch size of 64.

Results: Our best performance was at an accuracy of **68.87%** with a loss of 0.63.

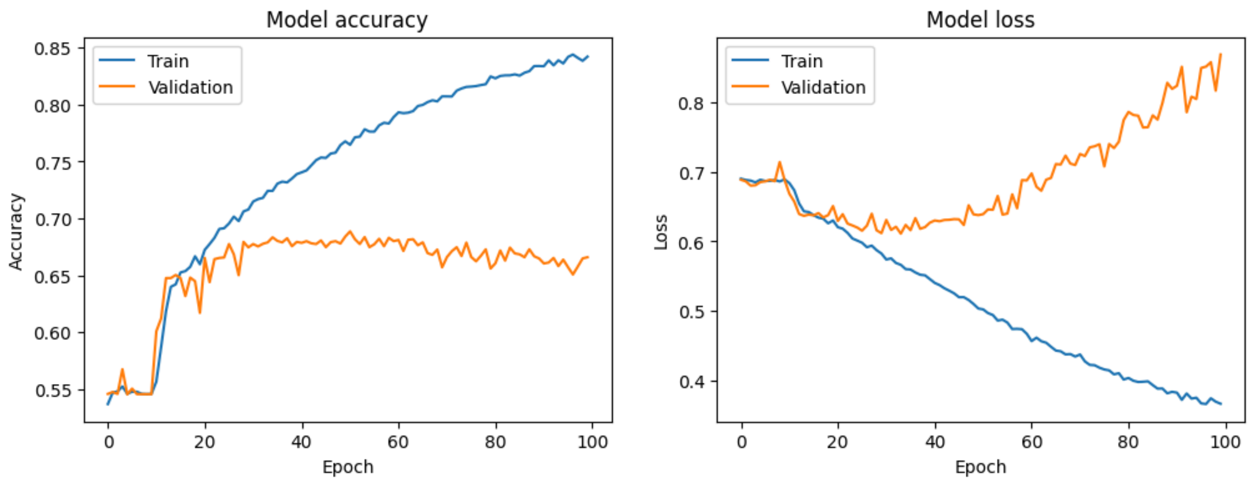


Fig.13 :- CNN + LSTM model Learning Curves (Accuracy and Loss)

The high jump of almost **5%** in the classification accuracy when a CNN + LSTM model was implemented, describes the ability of the LSTM to capture the dependencies across the DNA sequence which complements well with the local/spatial feature capturing ability of the CNN.

<u>NOVELTY - 3</u>

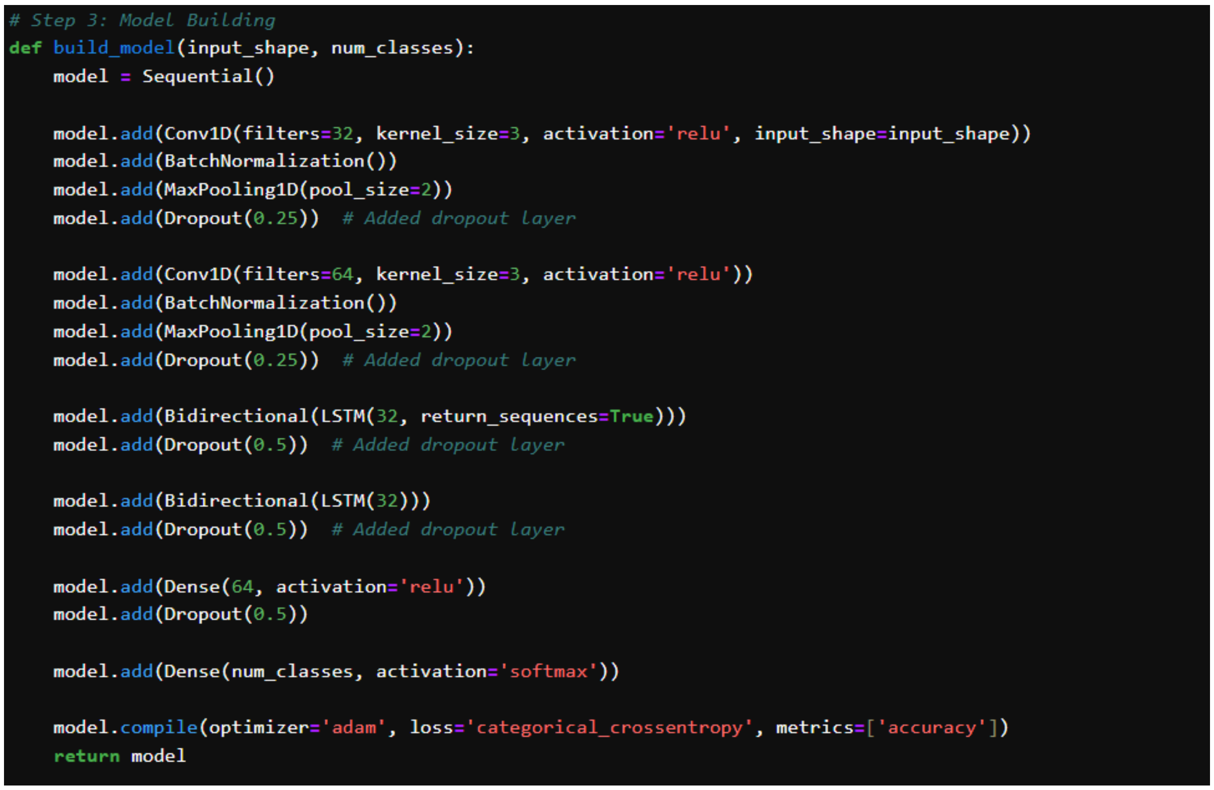
The third model utilizes a 4-mer representation along with one hot encoding method for sequence embedding. These sequences were then fed into a hybrid model of CNN and BiLSTM.

The model starts with a convolutional layer that uses 32 filters of size 3 to scan the input data. The ReLU was chosen as the activation function. Following this, a batch normalization layer is included. A max pooling layer with a pool size of 2 follows. To prevent overfitting, a dropout layer with a dropout rate of 0.25 is added, randomly setting a quarter of the input units to zero during each training update.

The model then introduces a second convolutional layer with 64 filters of size 3, further refining the feature extraction process. This is followed by another batch normalization layer and a max pooling layer with a pool size of 2. An additional dropout layer with a 0.25 rate is included to enhance regularization. Following the convolutional layers, the model incorporates two Bidirectional LSTM layers. The first BiLSTM layer, with 32 units and return_sequences set to True, processes the input sequences in both

forward and backward directions, capturing dependencies from both past and future contexts. A dropout layer with a rate of 0.5 is added after the first BiLSTM layer to mitigate overfitting. The second BiLSTM layer, also with 32 units, further processes the data. Another dropout layer with a 0.5 rate is applied to enhance regularization.

Subsequently, a fully connected dense layer with 64 units and ReLU activation is included, transforming the features extracted by the previous layers into a higher-level representation. A dropout layer with a rate of 0.5 follows to prevent overfitting. The model concludes with an output dense layer with units corresponding to the number of classes, using the softmax activation function to produce class probabilities. The model is compiled with the Adam optimizer, known for its efficiency and adaptive learning rate capabilities, and the categorical cross entropy loss function.



```
# Step 3: Model Building
def build_model(input_shape, num_classes):
    model = Sequential()

    model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.25))  # Added dropout layer

    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.25))  # Added dropout layer

    model.add(Bidirectional(LSTM(32, return_sequences=True)))
    model.add(Dropout(0.5))  # Added dropout layer

    model.add(Bidirectional(LSTM(32)))
    model.add(Dropout(0.5))  # Added dropout layer

    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))

    model.add(Dense(num_classes, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Fig.14 :- CNN +Bi LSTM Architecture

Results: Our best performing model had an accuracy of **70.51%** with a loss of 0.6018.

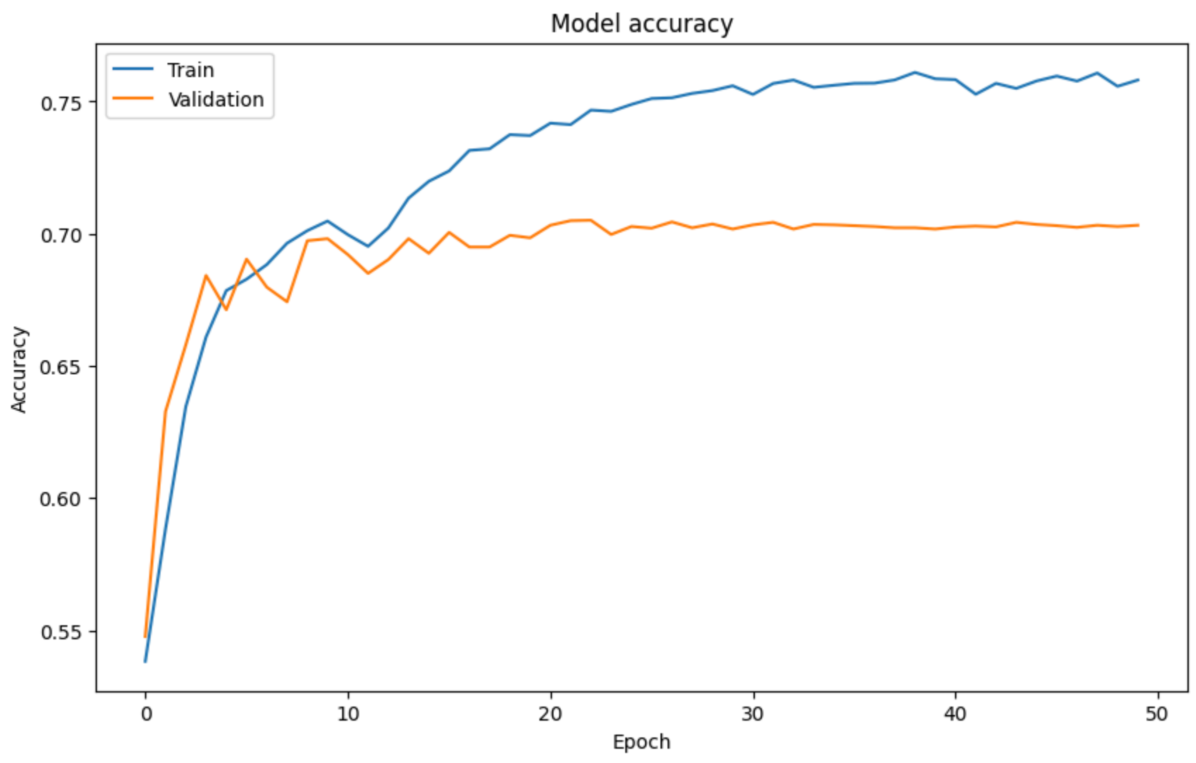


Fig.15 :- CNN + BiLSTM with One-hot encoding model learning curve (accuracy)

The classification accuracy increases even more (even though only by a little) when a CNN + Bi-LSTM model is implemented, describing the added benefit of the Bi-directional pattern capturing ability of a Bi-LSTM along with the ability to capture the dependencies across the DNA sequence. However, we don't stop here. Next up, using a different embedding might increase the accuracy even more.

NOVELTY - 4

We used the same architecture as the above but now used Word2Vec for sequence embedding.

Result: Our best performing model had an accuracy of maximum **71.87%** and a minimum of **58.27%** with a loss of 0.58. Thus an increase of **2%** over the one-hot encoding CNN+Bi-LSTM model and a whopping increase of **15%** over the base CNN model. Thus this is the best performance obtained across all the models obtained.

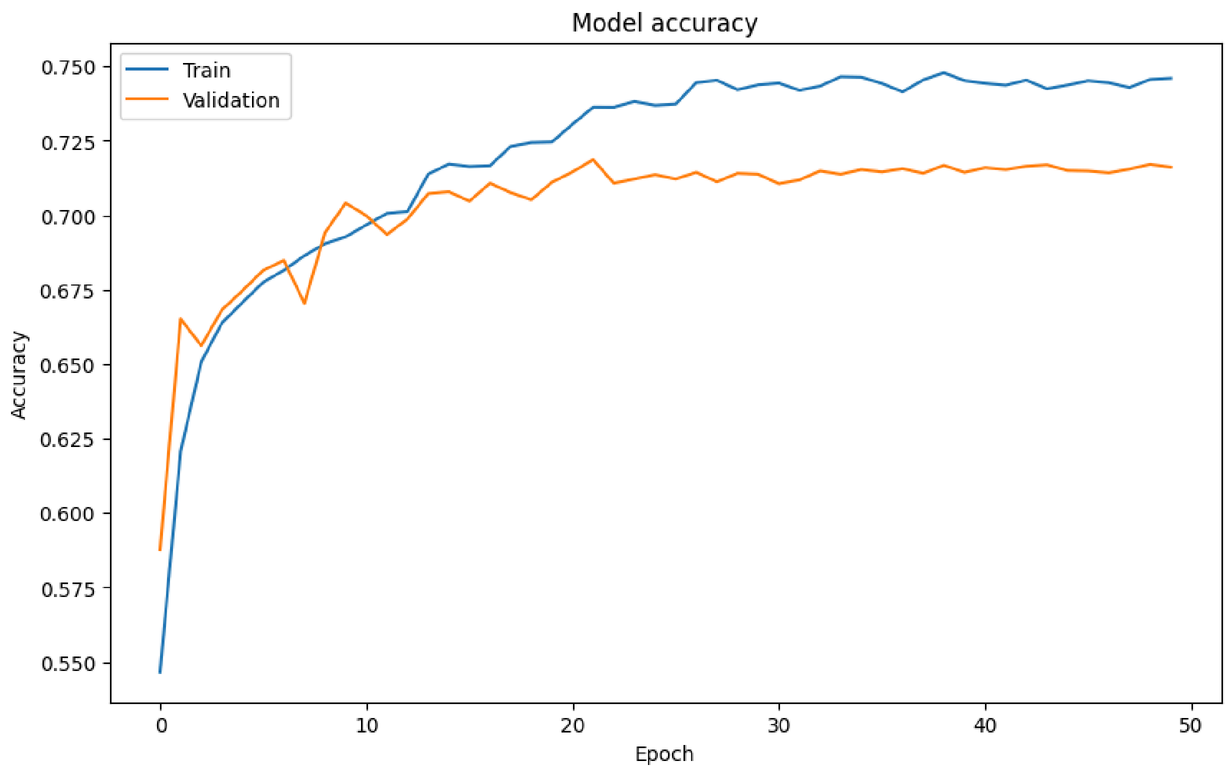


Fig.16 :- CNN + BiLSTM with Word2Vec embedding model learning curve (accuracy)

5. Tabulation of results:-

The below given table compares the minimum and the maximum accuracy scores obtained across all the models implemented. Accordingly Using a 4-representation along with the Word2Vec Embedding implemented with a CNN + Bi LSTM model gives the best performance.

Representation	Embedding	Model	Min Accuracy	Max Accuracy
3-mer	One-hot encoding	Base CNN	51%	57%
3-mer	One-hot encoding	Fine tuned CNN	53%	64%
4-mer	Word2Vec	CNN+LSTM	54.56%	68.87%
4-mer	One-hot encoding	CNN+BiLSTM	63.28%	70.51%
4-mer	Word2Vec	CNN+BiLSTM	58.7%	71.87%

Table.2 :- A comparison of minimum and maximum accuracy scores of our models

6. Conclusions:-

In our research, we aimed to enhance the performance of DNA sequence classification models by exploring different model architectures and embedding techniques. Initially, we fine-tuned a base model combining a Convolutional Neural Network (CNN) with one-hot encoding and a 3-mer representation. Our modifications included adding an additional CNN layer to extract more specific features, incorporating an L2 regularizer to prevent overfitting, and employing callback techniques like Early Stopping and ReduceLR on plateau to monitor and optimize the training process. Hyperparameter tuning was also performed to adjust learning rates, epochs, and filter numbers, which significantly improved the model's robustness.

In the second approach, we introduced a 4-mer representation along with Word2Vec for sequence embedding, which was then fed into a hybrid CNN and LSTM model. The convolutional layer captured local patterns in the DNA sequences, while the LSTM layers captured long-range dependencies, crucial for understanding the sequential nature of DNA. This model achieved an accuracy of 68.87%, demonstrating the effectiveness of combining CNNs with LSTMs for sequence classification.

Our third model utilized a 4-mer representation with one-hot encoding, followed by a hybrid CNN and Bidirectional LSTM (BiLSTM) architecture. The BiLSTM layers processed the sequences in both forward and backward directions, capturing dependencies from both past and future contexts. This model achieved an accuracy of 70.51%, highlighting the superior capability of BiLSTM in handling sequential data.

Finally, we further improved the architecture by using Word2Vec for sequence embedding in the hybrid CNN and BiLSTM model. This combination yielded the highest accuracy of 71.87%, demonstrating that the use of Word2Vec embeddings effectively enhanced the model's ability to understand and classify DNA sequences.

Overall, our exploration of different embedding techniques and model architectures, particularly the hybrid CNN and BiLSTM model with Word2Vec embedding, provided significant improvements in DNA sequence classification accuracy, showcasing the importance of advanced sequence embedding methods and hybrid architectures in bioinformatics.

Please find our code on this link:- https://github.com/toasted-breads-2/Machine_Learning_Project