2025 International Conference on Electrical, Computer and Communication Engineering (ECCE),
13-15 February 2025, CUET, Chattogram-4349, Bangladesh

2025 International Conference on Electrical, Computer and Communication Engineering (ECCE) | 979-8-3503-5750-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ECCE64574.2025.11013923

Enhanced DNA Sequence Classification Through Hyperparameter Optimized Convolutional Neural Networks

Md Mehedi Hasan
Department of Computer Science and Engineering
Port City International University
Chattogram, Bangladesh
mdmehedihasan1515@gmail.com

Hura Jannat Poly
Department of Computer Science and Engineering
Port City International University
Chattogram, Bangladesh
hurazannat876@gmail.com

Rayhanuzzaman
Department of Computer Science and Engineering
International Islamic University Chittagong
Chattogram, Bangladesh
rayhanuzzamanr@gmail.com

Azmain Yakin Srizon
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology
Rajshahi, Bangladesh
azmainsrizon@gmail.com

***Abstract*—DNA sequence classification is a fundamental task in bioinformatics, with applications ranging from gene prediction to disease diagnosis. Convolutional Neural Networks (CNNs) are a deep learning-based method that is suggested for DNA sequence classification. Two significant biological datasets, Promoter and Splice, are utilized for evaluation. The raw DNA sequences are transformed into numerical representations through k-mer feature extraction, serving as input for the CNN model. Multiple convolutional and pooling layers, followed by a fully linked layer for classification, make up the carefully planned CNN architecture. Extensive hyperparameter tuning is conducted to optimize the model, including adjustments to parameters such as learning rate, kernel size, and number of layers. Experimental results indicate that the tuned CNN model achieves a classification accuracy of 97% on both the Promoter and Splice datasets. The performance of the proposed model is compared with other baseline machine learning models, demonstrating its superior effectiveness. This study underscores the utility of CNNs in DNA sequence classification tasks and provides valuable insights for genomic research and applications in bioinformatics.**

***Index Terms*—DNA Sequence Classification, Convolutional Neural Networks, Hyperparameter Tuning, k-mer Feature Extraction, Promoter Dataset, Splice Dataset, Machine Learning, Bioinformatics**

I. Introduction

DNA sequence classification is a critical task in bioinformatics, with applications spanning gene expression prediction, genetic disease diagnosis, and biomarker discovery. The advent of next-generation sequencing technologies has exponentially increased the volume of genetic data, necessitating the development of efficient and accurate classification methods. By categorizing DNA sequences into functional groups, researchers gain insights into gene regulation, evolutionary biology, and disease mechanisms.

For the classification of DNA sequences, conventional machine learning techniques like Support Vector Machines (SVM) [7] and Random Forests have been used. However, these methods often require significant feature engineering and may struggle to capture the complex, hierarchical patterns characteristic of DNA sequences. Deep learning techniques have become strong substitutes, especially Convolutional Neural Networks (CNNs) [2], [11], [17]. CNNs inherently learn hierarchical features through convolutional layers, making them well-suited for sequence data where spatial locality and feature representation are crucial.

Recent advancements in CNN-based models have demonstrated their potential in identifying gene promoters, splice sites, and other genomic features. Despite their success, challenges remain in optimizing hyperparameters and ensuring generalizability across diverse datasets. Addressing these challenges is essential for maximizing the performance of deep learning models in DNA sequence classification.

The results demonstrate significant performance improvements, positioning the proposed model as a state-of-the-art approach for DNA sequence classification. The objectives of this study include:

- Developing and applying a parameter-tuned CNN model for DNA sequence classification.
- Evaluating the impact of hyperparameter optimization on enhancing model performance.
- Establishing a robust framework to support future research in genomic classification tasks.

These contributions highlight the potential of deep learning techniques in bioinformatics, paving the way for future advancements in the field.

II. Literature Review and Analysis

DNA sequence classification is crucial in bioinformatics, disease prediction, and genomic research. The ability to clas-

sify sequences into functional categories aids in understanding gene regulation and genetic variations. With the rise of high-throughput sequencing, accurate and scalable classification methods have become essential.

Traditional machine learning models like SVM, Random Forest, and k-NN rely on handcrafted features such as k-mer frequency representations [1], [10]. While effective, they struggle to capture the complex, hierarchical patterns in DNA sequences. Deep learning, particularly CNNs, has revolutionized this domain by learning spatial hierarchies directly from sequence data, eliminating the need for manual feature extraction.

Several studies have explored deep learning for DNA classification. Rizzo et al. [2] utilized CNNs but lacked architecture optimization. Akkaya and Kalkan [1] used k-mer-based representations but did not systematically tune hyperparameters. Juneja et al. [3] employed traditional machine learning with k-mer counting, while Morales et al. [4] applied CNNs to genomic signal classification without DNA-specific adaptations. Nair et al. [5] explored ANN with Chaos Game Representation but faced limitations in capturing spatial dependencies. Anveshrithaa et al. [6] focused on promoter prediction with machine learning but did not explore CNNs.

Despite progress, challenges remain in optimizing CNN architectures for DNA sequence classification. This study addresses these gaps by introducing a parameter-tuned CNN model that enhances classification performance, particularly on the Promoter and Splice datasets, demonstrating superior feature extraction and hierarchical learning.

III. Methodology

This section describes the methodology employed in this study for DNA sequence classification using Convolutional Neural Networks (CNNs). The approach involves five key steps: dataset description, data preprocessing, CNN model architecture design, hyperparameter tuning, and training strategies. An overview of the methodology is presented in Figure 1.

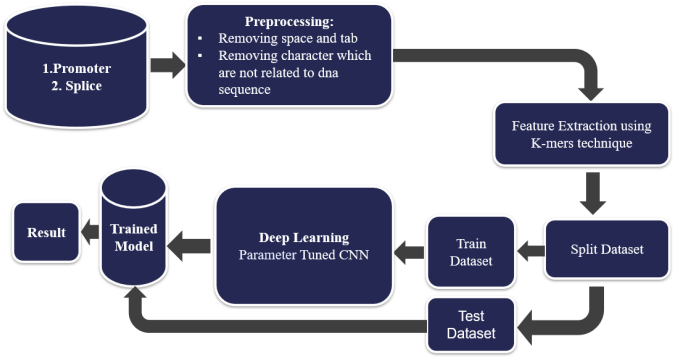

Fig. 1. Overview of the methodology for DNA sequence classification.

A. Dataset Description

For this study, we utilize two well-known biological datasets: the Promoter dataset and the Splice dataset. These datasets are widely used in genomics and bioinformatics for understanding gene regulation and splice site prediction, respectively [1], [2].

1) Promoter Dataset: The Promoter DNA dataset contains genetic sequences essential for studying gene expression, aiding bioinformatics research, and identifying regulatory elements [1]. It is also widely used for training machine learning models to investigate transcriptional mechanisms.

The dataset consists of 106 DNA sequences, each containing 57 base pairs. Of the 106 samples, 53 are positive samples representing sequences from promoter regions critical for gene activation, while the remaining 53 are negative samples without promoter sequences. These sequences are represented using the four nucleotide bases: A (adenine), C (cytosine), G (guanine), and T (thymine) [1].

2) Splice Dataset: The Splice DNA dataset contains genetic sequences used to study the splicing process in molecular biology. It consists of 3,190 samples, each with 60 base pairs. The dataset is divided into three classes: N (Neither EI nor IE), IE (Intron-Exon junction), and EI (Exon-Intron junction).

These classes indicate the presence or absence of a splice junction, and if present, the type of splice junction (EI or IE). Exons are regions utilized to form mRNA, while introns are sequences removed during RNA transcription. The points where these two regions converge are referred to as splice junctions, making this dataset essential for understanding RNA splicing and its implications [2].

TABLE I
Summary of the Promoter and Splice Datasets

Dataset	Number of Classes	Number of Samples	Sequence Length
Splice	3	3,190	60
Promoter	2	106	57

B. Data Preprocessing

The first step in the DNA sequence classification process is data preprocessing. Raw DNA sequences are typically composed of letters from the alphabet $\{A, C, G, T\}$, representing the four nucleotide bases (adenine, cytosine, guanine, thymine). To convert these sequences into a format suitable for input into machine learning models, we use *k-mer feature extraction* [3]. In this process, the DNA sequence is split into overlapping substrings (k-mers) of length k. For example, a sequence of length 10 can be divided into 10-kmers of length 3, 4, or 5, depending on the choice of k. Each k-mer is then encoded as a numerical vector, representing the occurrence of nucleotide bases at each position within the k-mer. The resulting feature vectors are used as input for the CNN model.

For this study, we used value of $k = 3$ and $k = 4$ for k-mer extraction. The extracted features are then normalized and divided into training and test datasets [4]. The test set is used to assess the model's ultimate performance, whereas the training set is used to train the model.

C. CNN Model Architecture

The CNN model for DNA sequence classification is structured to effectively learn hierarchical spatial patterns and dependencies from the input sequence data [5]. The key layers and their functionalities are described below.

1) Convolutional Layers: Convolutional layers are responsible for extracting local patterns in the DNA sequences. Each convolution operation is defined as:

$$y[i,j] = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} x[i+m, j+n] \cdot w[m,n] + b,$$

where $x[i,j]$ represents the input feature map, $w[m,n]$ is the convolution kernel, b is the bias term, and k_h, k_w are the kernel dimensions. The first convolutional layer utilizes a kernel size of 5 with 64 filters and applies the ReLU activation function:

$$f(x) = \max(0, x).$$

Subsequent convolutional layers have kernel sizes of 3 and 5, with progressively increasing filter numbers (128 and 256, respectively), enhancing the model's capacity to detect complex patterns.

2) MaxPooling Layers: MaxPooling layers are used for downsampling the feature maps, reducing their spatial dimensions while preserving the most important features. The pooling operation is defined as:

$$y[i,j] = \max_{m,n} \{x[i+m, j+n]\},$$

where m, n iterate over the pooling window. This operation reduces the dimensionality and computational complexity of the network, ensuring efficient learning.

3) Dropout Layers: By randomly changing a portion of the input units to zero during training, dropout layers reduce overfitting. The output y of a Dropout layer is:

$$y_i = \begin{cases} x_i & \text{with probability } (1-p), \\ 0 & \text{with probability } p, \end{cases}$$

where p is the dropout rate, set to 0.5 in this model. This ensures robust learning by preventing co-adaptation of neurons.

4) Fully Connected Layer: The feature maps are flattened and then run through a fully connected layer following the convolutional and pooling layers. This layer combines all extracted features to form high-level representations. The activation function used is ReLU, defined earlier, with 512 units to learn complex non-linear mappings.

5) Output Layer: The output layer utilizes the softmax activation function for classification, producing a probability distribution over the classes:

$$P(y = c|x) = \frac{\exp(z_c)}{\sum_{k=1}^{K} \exp(z_k)},$$

where z_c is the logit corresponding to class c, and K is the total number of classes. This enables multi-class classification with normalized probabilities.

The architecture is designed to effectively capture local dependencies in DNA sequences while ensuring generalization to unseen data.

D. Hyperparameter Tuning

Optimizing the performance of the CNN model requires fine-tuning hyperparameters. A grid search strategy [13] is employed to identify the optimal configuration. The key hyperparameters are described below.

1) Learning Rate: The learning rate governs the step size during gradient descent, defined as:

$$\theta = \theta - \eta \cdot \nabla J(\theta),$$

where η is the learning rate, θ represents the model parameters, and $J(\theta)$ is the loss function. Learning rates in the range of [0.0001, 0.001, 0.01] are tested to ensure stable convergence without overshooting.

2) Kernel Size: The kernel size, representing the dimensions of the convolutional filters, affects the receptive field of the model. Kernel sizes of 3, 5, and 7 are experimented with to evaluate their impact on feature extraction, balancing local and global patterns.

3) Number of Filters: The number of filters in each convolutional layer determines the diversity of patterns the model can learn. Configurations with 64, 128, and 256 filters are tested, increasing the model's representational capacity while controlling computational cost.

4) Number of Layers: The depth of the model is adjusted to explore its effect on classification performance. Architectures with 2, 3, and 4 convolutional layers are evaluated, allowing the model to capture both low-level and high-level features.

The best hyperparameter combination is selected based on the validation set's highest classification accuracy. This ensures optimal model performance while minimizing overfitting.

E. Training Strategies

To ensure robust training and minimize the risk of overfitting, several training strategies are employed. These strategies enhance the model's performance and reliability during the optimization process.

1) EarlyStopping: The validation loss is tracked during the training process by using EarlyStopping. Training is stopped if the validation loss does not get better after a certain number of epochs. The stopping condition can be mathematically represented as:

$$\text{Stop training if } L_{\text{val}}(t) \geq \min_{t-k \leq t' \leq t} L_{\text{val}}(t'),$$

where $L_{\text{val}}(t)$ is the validation loss at epoch t, and k is the patience parameter. This strategy helps prevent overfitting by avoiding redundant epochs of training.

2) ReduceLROnPlateau: Following a given number of epochs in which the validation loss plateaus, ReduceLROnPlateau dynamically modifies the learning rate. The learning rate update can be expressed as:

$$\eta_{t+1} = \eta_t \times \text{factor}, \quad \text{if } \Delta L_{\text{val}}(t) < \epsilon,$$

where η_t is the learning rate at epoch t, factor is a predefined reduction factor (e.g., 0.1), and ϵ is the tolerance for improvement. This adjustment enables the model to converge

more effectively by taking smaller optimization steps during gradient descent.

3) ModelCheckpoint: ModelCheckpoint saves the model weights at specified intervals or when a monitored metric, such as validation loss, reaches its optimal value. Mathematically, the model at epoch t is saved if:

$$L_{\text{val}}(t) = \min_{t' \leq t} L_{\text{val}}(t'),$$

where $L_{\text{val}}(t)$ is the validation loss at epoch t. This ensures that the best-performing model is preserved for evaluation and deployment, providing a reliable checkpoint for future use.

These training strategies collectively ensure effective optimization and generalization of the CNN model while preventing overfitting and underfitting.

F. Research Instruments

The following research instruments were used in this study to implement and evaluate the DNA sequence classification model:

1) Programming Language: Python 3.10 was used for developing the model due to its extensive libraries and tools supporting machine learning and deep learning applications.

2) Deep Learning Framework: TensorFlow 2.0 and Keras were employed to build, train, and evaluate the Convolutional Neural Network (CNN) model.

3) Data Preprocessing Tools: The scikit-learn library was utilized for data preprocessing tasks, including encoding DNA sequences into numerical vectors, splitting the dataset into training and test sets, and performing feature scaling.

4) Optimization Tool: GridSearchCV from scikit-learn was used for hyperparameter tuning. This tool optimized parameters such as learning rate, kernel size, and the number of layers to improve model performance.

5) Hardware: The model training process was conducted on an NVIDIA T4 GPU, which significantly accelerated computations, particularly for handling large datasets and complex neural network layers.

6) Visualization Tools: Matplotlib and Seaborn were used to generate various visualizations, including accuracy vs. epoch plots, loss curves, and confusion matrices, aiding in the analysis of model performance.

IV. Results

Through classification tasks, the current section demonstrates how a parameter-tuned Convolutional Neural Network (CNN) operates on Promoter and Splice datasets. Standard classification metrics, such as accuracy, precision, recall, and F1-score computations, were used to evaluate the model's performance metrics. The model's greater capabilities is demonstrated by comparing its performance to earlier state-of-the-art methods.

A. Performance Metrics

The detailed performance metrics for the Promoter and Splice datasets are presented in Tables II and III. The CNN model achieves an accuracy of 97% on both datasets, with high precision, recall, and F1-score for all classes.

TABLE II
Performance Metrics for the Promoter Dataset

Metric	Class 0	Class 1	Overall
Precision	1.00	0.94	0.97
Recall	0.94	1.00	0.97
F1-score	0.97	0.97	0.97
Accuracy	0.94	1.00	0.97
Support	16	16	32

TABLE III
Performance Metrics for the Splice Dataset

Metric	Class 0	Class 1	Class 2	Overall
Precision	0.97	0.94	0.98	0.97
Recall	0.95	0.95	0.98	0.96
F1-score	0.96	0.95	0.98	0.97
Accuracy	0.95	0.95	0.98	0.97
Support	230	230	497	957

B. Comparison with Previous Work

To evaluate the effectiveness of our model, we compare our results with those from previous studies. Table IV presents a comparison between our parameter-tuned CNN model and other state-of-the-art models on both the Promoter and Splice datasets.

TABLE IV
Comparison of Performance with Previous Work

Model	Promoter Accuracy	Splice Accuracy	Method
U. M. Akkaya et al. [1]	95.45%	96.85%	CNN (dna2vec))
Our Model	**97%**	**97%**	Parameter-tuned CNN

Our parameter-tuned CNN model outperforms the previous state-of-the-art methods, achieving an accuracy of 97% on both the Promoter and Splice datasets. The results demonstrate the effectiveness of our hyperparameter tuning process [13], which enables the model to generalize well to different genomic datasets.

C. Performance Comparison with Other Models

To further evaluate the effectiveness of our parameter-tuned CNN model, we compare its performance with various models on both the Promoter and Splice datasets. Tables V and VI present the comparison of model accuracy.

As shown in the tables, our parameter-tuned CNN model achieves 97% accuracy on both the Promoter and Splice datasets, outperforming most traditional machine learning models.

D. Visualization of Results

In this subsection, we present the accuracy curves, loss curves, and confusion matrices for the CNN model on both the Promoter and Splice datasets.

1) Promoter Dataset Results: Figures 2 and 3 show the accuracy and loss curves for the CNN model on the Promoter dataset. Effective learning and convergence are indicated by the loss curve's persistent decline in both training and

TABLE V
PERFORMANCE COMPARISON ON THE PROMOTER DATASET

Model	Accuracy
KNN	0.74
SVM Sigmoid	0.74
Decision Tree (DT)	0.74
Logistic Regression	0.78
Random Forest (RF)	0.81
Gradient Boosting	0.89
Neural Network	0.78
Extra Trees	0.78
AdaBoost	0.74
SVM Polynomial	0.85
Bagging Classifier	0.81
SVM Linear	0.78
XGBoost	0.85
SVM RBF	0.93
Gaussian Process	0.81
ANN	0.97
RNN	0.94
CNN	**0.97**
LSTM	0.97

TABLE VI
PERFORMANCE COMPARISON ON THE SPLICE DATASET

Model	Accuracy
KNN	0.58
Quadratic Discriminant Analysis	0.62
Decision Tree	0.55
Passive Aggressive Classifier	0.56
Random Forest	0.69
Ridge Classifier	0.61
Gradient Boosting	0.66
XGBoost	0.69
SVM Linear	0.62
SVM with RBF	0.67
Extra Tree	0.68
Multi-Layer Perceptron	0.65
Bagging	0.63
AdaBoost	0.59
ANN	0.96
RNN	0.89
CNN	**0.97**
LSTM	0.91

validation loss, while the accuracy curve shows a steady improvement in both training and validation accuracy.

Figure 4 shows the confusion matrix for the Promoter dataset, highlighting the classification performance in terms of true positives, true negatives, false positives, and false negatives.

2) Splice Dataset Results: Figures 5 and 6 show the accuracy and loss curves for the CNN model on the Splice dataset. Similar to the Promoter dataset, the accuracy curve shows steady improvement, while the loss curve illustrates smooth convergence during training.

Figure 7 shows the confusion matrix for the Splice dataset, demonstrating the model's performance on classifying splice sites.

V. CONCLUSION AND FUTURE WORK

In this study, a parameter-tuned Convolutional Neural Network (CNN) was proposed for DNA sequence classification,

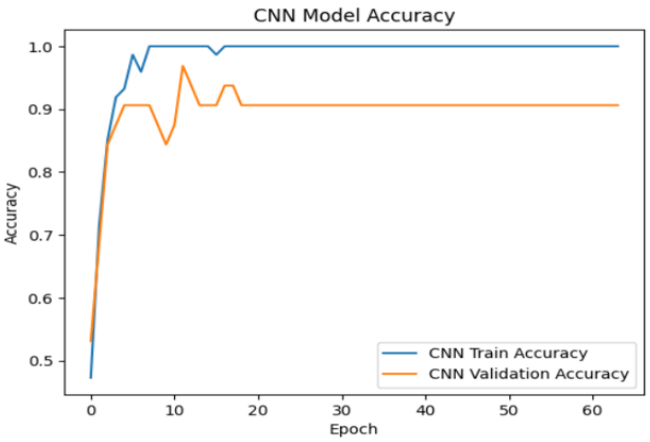

Fig. 2. Training and Validation Accuracy for the Promoter Dataset.

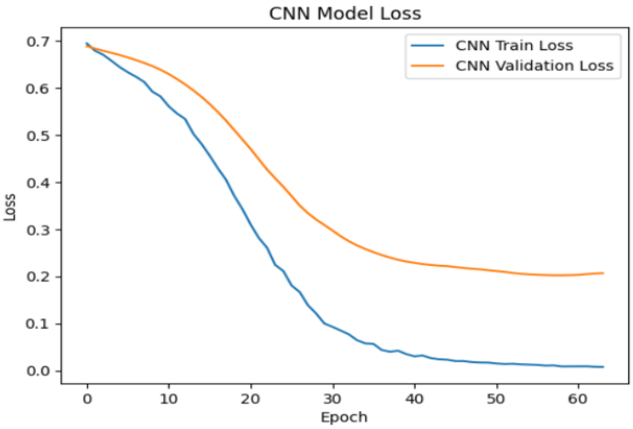

Fig. 3. Training and Validation Loss for the Promoter Dataset.

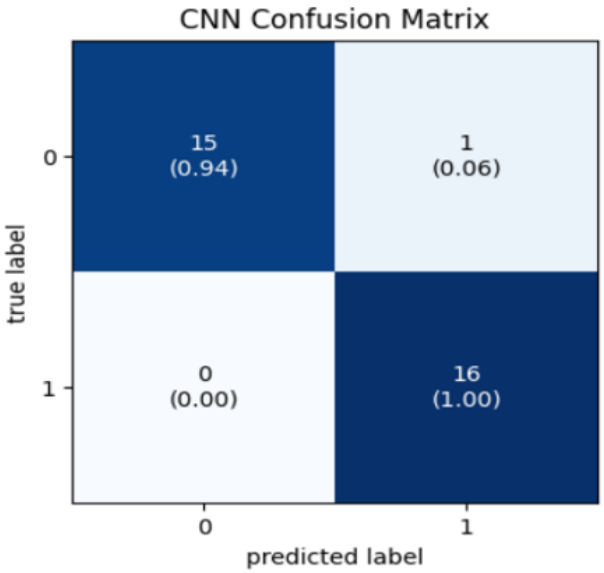

Fig. 4. Confusion Matrix for the Promoter Dataset.

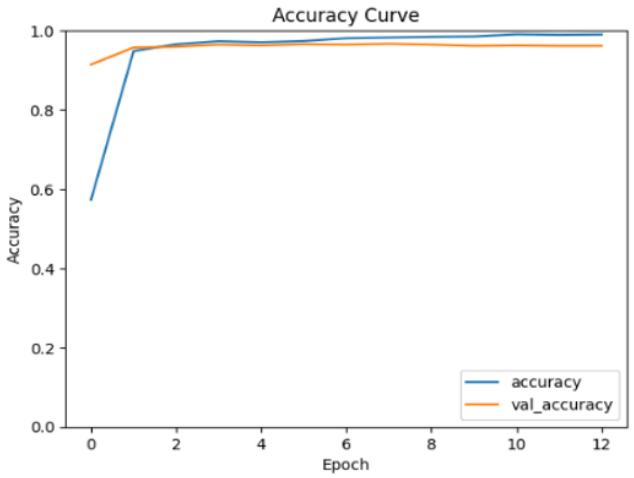

Fig. 5. Training and Validation Accuracy for the Splice Dataset.

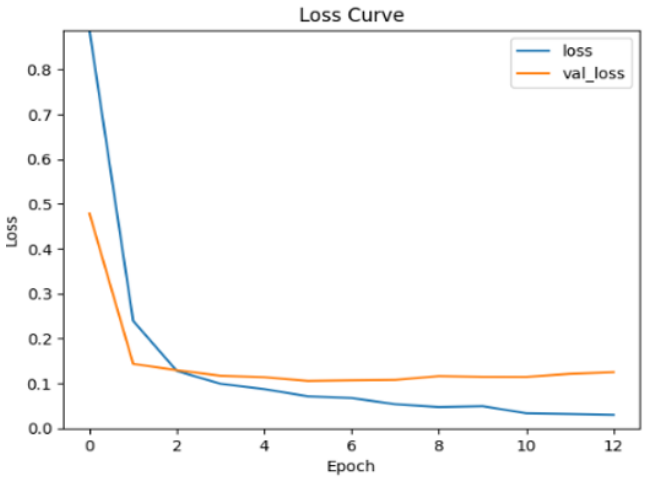

Fig. 6. Training and Validation Loss for the Splice Dataset.

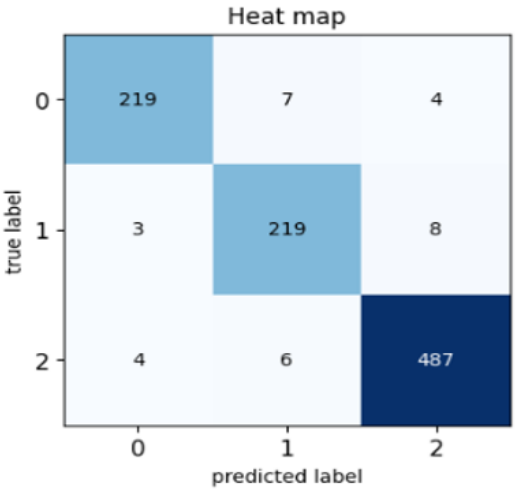

Fig. 7. Confusion Matrix for the Splice Dataset.

focusing on the Promoter and Splice datasets. The optimized model achieved a classification accuracy of 97%, outperforming previous methods. Key contributions include a novel CNN architecture, comprehensive hyperparameter tuning, and robust evaluation across biologically significant datasets. These results underscore the potential of CNNs in bioinformatics for solving complex problems, while emphasizing the importance of model optimization. Future research could explore transfer learning [17] and apply the model to additional genomic datasets, broadening its applicability in gene expression prediction, disease classification, and personalized medicine.

REFERENCES

[1] U. M. Akkaya and H. Kalkan, "Classification of DNA Sequences with k-mers Based Vector Representations," in *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, IEEE, 2021, pp. 1–5.

[2] R. Rizzo *et al.*, "A deep learning approach to DNA sequence classification," in *Computational Intelligence Methods for Bioinformatics and Biostatistics: 12th International Meeting, CIBB 2015, Naples, Italy, September 10-12, 2015, Revised Selected Papers 12*, Springer, 2016, pp. 129–140.

[3] S. Juneja *et al.*, "An approach to DNA sequence classification through machine learning: DNA sequencing, K Mer counting, thresholding, sequence analysis," *Int. J. Reliable and Quality E-Healthcare (IJRQEH)*, vol. 11, no. 2, pp. 1–15, 2022.

[4] J. A. Morales *et al.*, "Deep learning for the classification of genomic signals," *Mathematical Problems in Engineering*, 2020, pp. 1–9.

[5] V. V. Nair *et al.*, "ANN based classification of unknown genome fragments using Chaos Game Representation," in *2010 Second International Conference on Machine Learning and Computing*, IEEE, 2010, pp. 81–85.

[6] S. Anveshrithaa, B. Aathavan, and N. Jaisankar, "Promoter prediction in DNA sequences of Escherichia coli using machine learning algorithms," *Int. J. Scientific and Technology Research*, vol. 8, no. 11, pp. 3000–3004, 2019.

[7] M. Awad *et al.*, "Support vector machines for classification," in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, Springer, 2015, pp. 39–66.

[8] X. Chen and J. Chen, "Emerging Patterns and Classification Algorithms for DNA Sequence," *J. Softw.*, vol. 6, no. 6, pp. 985–992, 2011.

[9] P. Dixit and G. I. Prajapati, "Machine learning in bioinformatics: A novel approach for DNA sequencing," in *2015 Fifth International Conference on Advanced Computing Communication Technologies*, IEEE, 2015, pp. 41–47.

[10] A. Fiannaca *et al.*, "A k-mer-based barcode DNA classification methodology based on spectral representation and a neural gas network," *Artificial Intelligence in Medicine*, vol. 64, no. 3, pp. 173–184, 2015.

[11] H. Gunasekaran *et al.*, "Analysis of DNA sequence classification using CNN and hybrid models," *Computational and Mathematical Methods in Medicine*, 2021.

[12] M. A. Habib *et al.*, "Classification of DNA sequence using machine learning techniques," in *EasyChair*, Aug 4, 2022.

[13] B. A. Hamed *et al.*, "Optimizing classification efficiency with machine learning techniques for pattern matching," *Journal of Big Data*, vol. 10, no. 1, p. 124, 2023.

[14] P. S. Hossain *et al.*, "Enhancing Taxonomic Categorization of DNA Sequences with Deep Learning: A Multi-Label Approach," *Bioengineering*, vol. 10, no. 11, p. 1293, 2023.

[15] M. Khashei *et al.*, "A fuzzy intelligent approach to the classification problem in gene expression data analysis," *Knowledge-Based Systems*, vol. 27, pp. 465–474, 2012.

[16] M. La Rosa *et al.*, "Probabilistic topic modeling for the analysis and classification of genomic sequences," *BMC Bioinformatics*, vol. 16, pp. 1–9, 2015.

[17] F. Mock *et al.*, "BERTax: Taxonomic classification of DNA sequences with Deep Neural Networks," *BioRxiv*, 2021.

[18] A. El-Tohamy *et al.*, "A deep learning approach for viral DNA sequence classification using genetic algorithm," *Int. J. Advanced Computer Science and Applications*, vol. 13, no. 8, 2022.