Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.067390

ARTICLE

A Comparative Study of Data Representation Techniques for Deep Learning-Based Classification of Promoter and Histone-Associated DNA Regions

Sarab Almuhaideb[1,*], Najwa Altwaijry[1], Isra Al-Turaiki[1], Ahmad Raza Khan[2] and Hamza Ali Rizvi[3]

[1]Computer Science Department, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh, 11543, Saudi Arabia

[2]Chemical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, 721302, India

[3]Computer Science and Engineering Department, Punjab Engineering College, Sector 12, Chandigarh, 160012, India

*Corresponding Author: Sarab Almuhaideb. Email: salmuhaideb@ksu.edu.sa

Received: 01 May 2025; Accepted: 09 July 2025; Published: 23 September 2025

ABSTRACT: Many bioinformatics applications require determining the class of a newly sequenced Deoxyribonucleic acid (DNA) sequence, making DNA sequence classification an integral step in performing bioinformatics analysis, where large biomedical datasets are transformed into valuable knowledge. Existing methods rely on a feature extraction step and suffer from high computational time requirements. In contrast, newer approaches leveraging deep learning have shown significant promise in enhancing accuracy and efficiency. In this paper, we investigate the performance of various deep learning architectures: Convolutional Neural Network (CNN), CNN-Long Short-Term Memory (CNN-LSTM), CNN-Bidirectional Long Short-Term Memory (CNN-BiLSTM), Residual Network (ResNet), and InceptionV3 for DNA sequence classification. Various numerical and visual data representation techniques are utilized to represent the input datasets, including: label encoding, *k*-mer sentence encoding, *k*-mer one-hot vector, Frequency Chaos Game Representation (FCGR) and 5-Color Map (ColorSquare). Three datasets are used for the training of the models including H3, H4 and DNA Sequence Dataset (Yeast, Human, Arabidopsis Thaliana). Experiments are performed to determine which combination of DNA representation and deep learning architecture yields improved performance for the classification task. Our results indicate that using a hybrid CNN-LSTM neural network trained on DNA sequences represented as one-hot encoded *k*-mer sequences yields the best performance, achieving an accuracy of 92.1%.

KEYWORDS: DNA sequence classification; deep learning; data visualization

1 Introduction

The volume of biological data available to researchers is growing at an unprecedented pace, especially with recent developments in sequencing technologies. For example, GenBank contains 15.3 trillion base pairs from over 2.5 billion nucleotide sequences for 504,000 species [1]. Also, as of December 2024, the National Institutes of Health's (NIH) Sequence Read Archive (SRA) hosts a total of 34.5 million samples from all domains of life, comprising more than 107 petabases of sequencing data—equivalent to 38 petabytes of raw data [2]. Thus, the analysis of this huge data is an important step for many applications, such as finding changes in genes, associations with diseases and phenotypes, as well as the ability to identify potential drug targets. *Deoxyribonucleic acid (DNA)* is the main biomolecule that carries all genetic information and instructions. The task of *DNA sequence classification* involves determining whether an unlabelled

DNA sequence belongs to a certain class. The problem of sequence classification can be formulated as follows: Given a set of sequences $S = \{s_1, s_2, \ldots s_N\}$, defined over the alphabet $\sum$, and a set of class labels $L = \{l_1, l_2, \ldots l_Z\}$, it is required to learn a classifier C, which is a function mapping a sequence s to a class label $l \in L$. For biological sequences, such as DNA, the alphabet is $\sum_{DNA} = \{A, C, G, T\}$. A DNA sequence, such as $s_i = ATCCGCA$, may be mapped to a label from $L = \{coding, noncoding\}$ [3].

Given the vast volume and variety of biological data, relying on wet-lab experiments has become inefficient. Traditional manual laboratory experiments cannot scale to handle the millions of samples available in repositories such as GenBank and the SRA. There are several computational methods to classify DNA sequences such as sequence alignment and consensus search-based methods [4], however, recent studies show that machine learning techniques have been adopted to accomplish this task, with varying degrees of success [5]. The task of classifying DNA sequences remains challenging due to the nature of biological sequences. A DNA sequence is a non-numerical sequence, with unknown relationships between nucleotides that contribute to the structure and function of the produced proteins. Moreover, the variability in DNA sample lengths further complicates the classification task [5], highlighting the need for innovative, AI-driven solutions that mimic biological processes to improve accuracy and efficiency in this domain.

Many machine learning algorithms are designed to work on an explicit set of features. However, sequence data have no explicit features and thus need to be transformed first using a preprocessing step. The resulting feature space may be high dimensional, which results in high computation and memory requirements. There are many data representation techniques that are used to deal with DNA data, such as *label encoding* [6], *k-mer sentence encoding* [7] and *Frequency Chaos Game Representation (FCGR)* [8]. It has been observed that classification performance can vary depending on the data representation technique used [5,9]. The task of classifying DNA sequences is still an open area for research, especially with deep learning (DL) techniques. The most difficult aspect of the problem is the feature selection process, as DNA sequences lack explicit features, and traditional machine learning approaches generally require manual feature engineering, and do not perform as well as deep learning [10,11]. Traditional representations of DNA sequences suffer from high dimensionality [6], requiring appropriate DNA sequence representations and algorithms to meet the requirements of specific biological applications [5].

The aim of this research is to provide a deeper understanding of the impact of DNA sequence-representation techniques, i.e., the preprocessing step, on the performance of various deep learning models. In order to accomplish this task, we compare the two main types of representation: numerical and visual, using appropriate models. To compare the numerical representations, we use two custom-built convolutional neural networks (CNN), and two custom-built hybrid CNN-Long Short-Term Memory (CNN-LSTM) neural networks. To compare the visual representations, we use two publicly available successful models, namely: Residual Network (ResNet50) [12] and InceptionV3 [13]. We evaluate the performance of our proposed models using the three benchmark datasets, H3 occupancy, H4 occupancy [14], and DNA Sequence Dataset (Yeast, Human, Arabidopsis Thaliana). The benefits of this paper to future research are three-fold, firstly, the comparison of various data representation techniques allows researchers to select the most successful one. Secondly, for the numerical representation, we compare the convolutional and hybridized recurrent neural network architectures for the DNA classification task, to ascertain the more successful approach. Finally, we allow future researchers to determine whether their own custom-built network is more desirable, compared with using transfer learning on an existing and publicly available model. The comparative results demonstrate that one-hot vector representation of DNA sequence k-mers gave the best overall result with a hybrid CNN-LSTM model, achieving an accuracy of 92.1%. The rest of the paper is organized as follows: First, we cover the previous studies in the area of DNA sequence classification using deep learning and machine learning techniques. Then, an overview of our proposed deep learning

models for DNA sequence classification is presented. Next, we show our experimental settings, and then our performance evaluation. Finally, we discuss and present our conclusions.

2 Related Work

DNA sequence analysis is a key task in bioinformatics for understanding an organism's genes, functions, and evolutionary relationships. With the ever-increasing availability of DNA sequences, relying solely on wet-lab experiments is no longer efficient, as they require specialized equipment, time, and costly reagents [15]. DNA sequence analysis is an indispensable tool for understanding gene expression regulation, gene function, and genome structure, to name just a few. Many bioinformatics applications involve the classification of DNA sequences, such as identifying enhancers, promoters, splice sites, and transcription factor binding sites. Depending on the application, the problem can be formulated as a binary, multi-class, or multi-label classification task.

Machine learning algorithms have been widely used for biological sequence classification [16]. One of the earliest attempts to address biological sequence classification was conducted by [17], who evaluated the performance of k-nearest neighbor (KNN), Markov models, and support vector machines (SVM) in classifying various DNA and protein sequences. They found that SVM outperformed the other classifiers in most cases. In addition, the performance of SVM is influenced by the feature space used; hence, appropriate feature selection contributes to better classification accuracy. The superiority of SVM was also confirmed by She et al. [18], who compared SVM to the See5 algorithm [19], a more recent upgrade of the the C4.5 algorithm decision tree algorithm and rule-based classification for biological sequence classification. Singh et al. [20] investigated the problem of Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) sequence classification using complementary DNA. A dataset of 1582 sequences of SARS-CoV-2 and a non-SARS-CoV-2 classes was utilized. Biomarkers extracted from the dataset were ranked and fed into several ML algorithms, including: k-nearest neighbor, support vector machines, decision trees, and random forest. Results shows that random forest outperformed other algorithms in classifying SARS-CoV-2 sequences with an accuracy of 97.4%, sensitivity of 96.2%, and specificity of 98.2%. According to Ao et al. [21], the most widely used machine learning algorithms for biological sequence classification are: random forest (RF), support vector machine (SVM), naive Bayes (NB), logistic regression (LR), decision tree (DT), light gradient boosting machine (LGBM), and extreme gradient boosting (XGBoost). Although simple to use, these algorithms require manual feature crafting and selection—a process that demands experience and multiple iterations. In addition, manual feature extraction often fails to capture complex, hidden relationships between molecules.

The demonstrated success of deep learning in various fields has encouraged its adoption in numerous bioinformatics tasks [22–25]. This presents an opportunity to harness the power of DL methods as an alternative to more traditional alignment and consensus search-based techniques to classify DNA/Ribonucleic acid (RNA) sequences. In 2016, Rizzo et al. [8] proposed CNNs to classify the FCGR of DNA sequences. One thousand sequences were randomly selected from three categories of bacteria from the 16S bacteria dataset and used for training a CNN that outperformed an SVM classifier on the same task. Nguyen et al. [7] utilized a neural network architecture called seqCNN to classify DNA sequences from 10 different datasets. The authors used one-hot vector representations of DNA sequence k-mers from 12 nucleosome datasets and noticed a maximum accuracy gain of 6%. The proposed model consisted of 2 convolutional layers, each followed by subsampling layers, which are then followed by a fully connected layer with the dropout set to 0.5 to prevent overfitting. Trabelsi et al. [26] presented deepRAM, a DL tool that provides end-to-end model selection for predicting DNA and RNA binding specificities, by performing an unbiased comparison of previously proposed DL architectures. The architecture that used a k-mer embedding along with convolutional and recurrent layers was found to outperform all other models for comparison.

The classification of viral DNA sequences using DL models was undertaken by Gunasekaran et al. [6]. They compared the performance of standalone one-dimensional (1D) CNNs with those of other neural network architectures that combined CNNs with LSTM recurrent layers (CNN-LSTM and CNN-BILSTM). During data preprocessing, DNA sequences are converted to an array of integers where each nucleotide corresponds to a fixed number or the arrays were used to generate k-mers, which are concatenated to form a k-mer sentence and then fed into a word-embedding layer in the neural network for conversion into a dense feature-vector matrix. This enabled the impact of the type of sequence encoding on the performance of the model to be evaluated. After the accuracy, specificity, sensitivity, and precision of all the models were evaluated, it was concluded that k-mer encoding was associated with higher testing and validation accuracies, the CNN model with k-mer encoding provided a higher precision rate, and CNN-Bidirectional LSTM with label encoding achieved higher recall and sensitivity. Rehman et al. [27] proposed a CNN-based architecture for the identification of N4-methylcytosine sites. The proposed architecture, DCNN-4mC, incorporates a set of neural network layers with a skip connection making shallow features available to the dense layers. Experiments were conducted using a dataset of sequences from 12 different species. Results indicate that DCNN-4mC achieves 2% to 14% higher accuracy than state-of-the-art architectures.

Before feeding raw biological sequences into deep learning algorithms, it is essential to convert them into an appropriate numerical or structured representation. The performance of the final model is heavily influenced by the choice of representation techniques used to encode biological sequences. These techniques play a critical role in capturing the underlying biological patterns and ensuring the model can effectively learn meaningful features from the data [28]. Bhandari et al. [29] compared the effect of various DNA sequence-encoding techniques on various machine-learning and DL algorithms during the classification of *promoter sequences*. They found that using frequency-based tokenization reduced training time without compromising the sensitivity and specificity of classification compared to using one-hot vectors, and that CNNs performed the best out of all the aforementioned classifiers in classifying promoter vs. non-promoter sequences (binary classification) as well as in species-specific classification of promoter sequences (multiclass classification). Related to capturing long-term interactions, a sparse and wide neural network is utilized for learning on lengthy DNA sequences, referred to as SwanDNA [30]. The model features a circular and multi-scale dilated architecture intended to capture long-range interactions within DNA sequences. The authors employ self-supervised learning allowing SwanDNA to minimize its need for supervised labels.

The identification of *transcription factor binding sites* (TFBSs) has gained attention due to its importance in understanding gene expression regulation and disease mechanisms. DNA sequence and shape features were integrated using convolutional neural networks (CNNs) and attention mechanisms [31]. This approach addresses the challenge of combining local and global features, demonstrating improved predictive performance across numerous validated ChIP-seq datasets. To simultaneously incorporate both DNA sequence and shape features, the authors [32] developed a model for predicting TFBSs that combines attention mechanisms, convolutional networks, and Recurrent Neural Networks (RNNs). The findings indicate that the proposed model, DeepCTF, improves TFBS prediction accuracy using 12 $in{-}vitro$ datasets from Protein Binding Microarray (PBMs) when compared to existing models in the field. While attention mechanisms are useful for identifying long-range dependencies, they face challenges with local feature details. Conversely, convolutional operations excel at capturing local features but can sometimes miss the broader context. RNNs also face challenges in recognizing long-term dependencies in sequences. Amato et al. [33] used Graph Convolutional Neural Network (GCNN) for the classification of transcription factors in protein sequences. Protein sequences were represented as De Bruijn graphs. Experiments were conducted using four datasets generated from Swiss-Prot dataset. The performance of the proposed model was compared to an SVM model with the datset represented as one-hot encoding. The results show that the proposed model is effective and

demonstrate its ability to classify protein sequences, particularly by handling sequences of any length without requiring fixed-length representations or extensive preprocessing.

Transformers are advanced deep learning architectures originally introduced for natural language processing (NLP) tasks, where they demonstrated remarkable performance in modeling long-range dependencies and contextual relationships. Building on their superior performance in NLP, transformers have been applied to biological sequence classification tasks. In [28], a 1-Dimentional Deep CNN (1DCNN) and an LSTM were compared to a model consisting of dual transformer blocks used to extract deep features from DNA sequence data encoded using k-mers. The used DNA nucleotide sequence dataset was sourced from the NCBI database and includes sequences from Coronavirus disease (COVID), Middle East Respiratory Syndrome (MERS), dengue, hepatitis, and influenza. The deep features extracted were concatenated and then classified using four classical ML methods: random forests, bagging trees, gradient boost, and AdaBoost algorithms. The dual transformer along with random forests classifier achieved the best results among the compared models. Mock et al. [34] presented BERTax, which is based on BERT, a deep neural network that is designed for NLP problems to classify the superkingdom and phylum of DNA sequences taxonomically. In another work, Pipoli et al. [35] divide mRNA sequences into 3-mers to create an embedding matrix using word2vec [36], utilizing a domain-specific embedding called DeepLncLoc [37]. A transformer is then employed to predict gene expression levels. The study showed that using DeepLncLoc embeddings, a transformer outperforms an LSTM on gene expression levels prediction task. On the same task, Dong et al. [38] utilized the pretrained models DNA Bidirectional Encoder Representation (DNABERT) [39] and DNABERT-2 [40] to generate embeddings for the gene expression data, that were then classified using a CNN called TExCNN, and demonstrated a superior performance in comparison to DeepLncLoc [35].

The literature has evolved from traditional machine learning techniques such as SVMs and Decision Trees to deep learning architectures. Early works demonstrated the superiority of SVMs, especially with well-chosen features, while more recent studies have leveraged CNNs and RNNs for improved performance. CNNs have been particularly effective in capturing spatial patterns in encoded sequences, as shown by their success in tasks such as promoter prediction, TFBS identification, and bacterial DNA classification. However, CNNs may struggle to model long-range dependencies within sequences. To address this, hybrid models combining CNNs with LSTM or BiLSTM architectures have emerged, demonstrating improved classification accuracy, especially when using k-mer and label encoding techniques. Additionally, attention-based models and transformers have shown promise in enhancing the extraction of contextual information from DNA/RNA sequences. Given the need to model both local patterns and long-range dependencies in biological sequences, implementing CNN-LSTM models represents a promising approach for robust and accurate sequence classification.

3 Methodology

DL methods are more efficient and effective in many applications, including computer vision, speech recognition, and medical applications [41]. In this paper, we use DL architectures to build DNA classification models. Before raw DNA sequences can be fed to DL models, data must be transformed into a suitable representation to enable the DL model to extract features. The data-representation technique plays an important role in determining the classification accuracy [5,9]. Fig. 1 illustrates a schematic view of the experiments. In the following sections, we describe the two benchmark datasets employed, the five data-representation techniques utilized, and the architectures of the five DL models examined in this study.

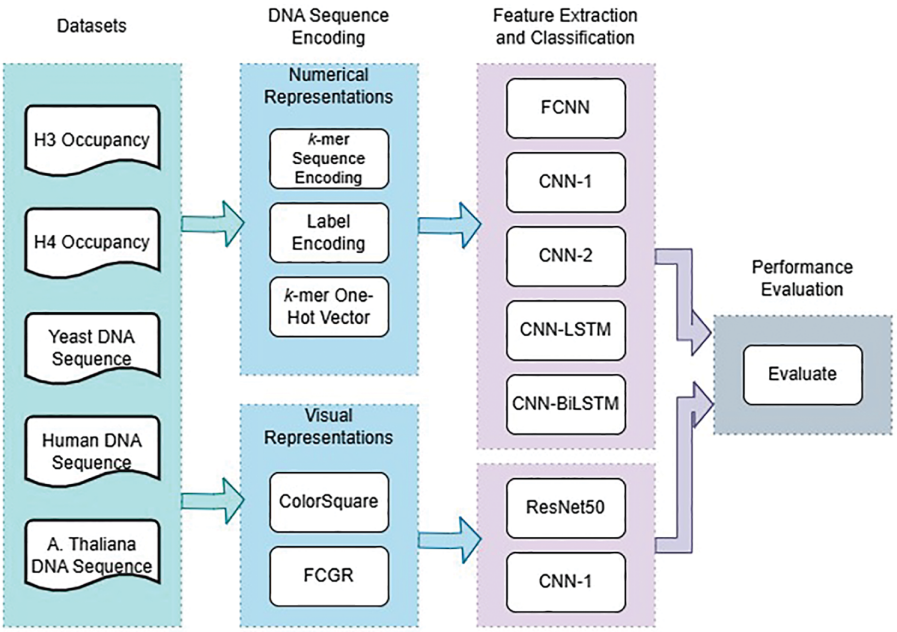


Figure 1: A schematic view of the DNA classification experiments

3.1 Dataset

We employed three datasets to evaluate the performance of various models and data-representation techniques for DNA classification: H3 and H4 occupancy [14], DNA Sequence Dataset (Yeast, Human, Arabidopsis Thaliana).

3.1.1 H3 and H4 Occupancy Dataset

H3 and H4 datasets include details on yeast DNA sequences wrapped around histone proteins, forming *chromatin* to be packed into a cell's nucleus. The packing process also prevents the DNA sequences from being tangled and becoming damaged. H3 and H4 refer to the type of *histone* around which the DNA sequence is wrapped. The *histone occupancy type* is one of the most important factors determining gene expression; hence, exploring computationally efficient processes to determine the type of histone occupancy in a DNA sequence is an active field of research because it enables a better understanding of expression patterns in genes. Both datasets include sequences with a length of 500 base pairs belonging to either a "Positive" or "Negative" class. H3 has 7667 positive and 7298 negative base pairs, and H4 has 6480 positive and 8121 negative base pairs. Sequences belonging to the "Positive" class contain regions that are wrapped around a histone protein of the type indicated by the name of the dataset ("H3" or "H4").

3.1.2 DNA Sequence Dataset

The DNA sequence dataset consisted of genomes of *Saccharomyces cerevisiae* (yeast), *Arabidopsis Thaliana* (A. Thaliana), and human samples reported by [29], and accessed through the University of California, Santa Cruz (UCSC) Genome Browser. The original dataset comprised approximately 6125 sequences from the yeast genome (UCSC version sacCer3), 41,671 sequences from the A. Thaliana genome (UCSC version araTha1), and 61,546 sequences from the human genome (UCSC version hg38). These sequences were curated to represent putative promoter regions, which are of particular interest in gene regulation studies. To ensure consistency and manageable dataset sizes for model development, a random selection of sequences

was performed following preprocessing and quality filtering. Specifically, 6000 sequences were retained for yeast, while 35,000 sequences each were selected for A. Thaliana and human genomes.

Each selected sequence represented a *putative promoter region* of 1000 base pairs in length, covering the region from −700 to +300 base pairs relative to the annotated *transcription start site* (TSS). The extraction of these sequences was conducted in two stages. In the first stage, 700 base pairs upstream of the TSS were retrieved using the UCSC Table Browser, with the Reference Sequence (RefSeq) Genes track serving as the input to create a custom annotation track. In the second stage, an additional 300 base pairs downstream of the TSS were extracted using the custom track, resulting in a complete 1000 bp promoter region for each gene, which was then saved in FASTA format [42].

To construct a balanced dataset for classification tasks, an equal number of background (negative) sequences were generated for each organism. These sequences were created through a synthetic shuffling process of the original promoter sequences. This ensured a 1:1 ratio of positive (true promoter) to negative (synthetic background) sequences, which is essential for reducing bias during model training. The final datasets were separately processed to develop and evaluate organism-specific machine learning models tailored to each genome.

3.2 DNA Sequence Representation

Preprocessing the raw data representing the DNA sequences into a suitable form for use as input to train the DL algorithms is an integral step towards building predictive systems for classification, as the method of encoding can have a significant impact on a model's performance. In this paper, we explore five methods to encode DNA sequences:

3.2.1 Numerical Representations

1. ***k*-mer Sentence Encoding:** A k-mer is a substring of size k contained in a DNA sequence [6]. For example, for $k = 3$, the sequence*AAGCT* can be represented as [AAG, AGC, GCT]. Each k-mer is replaced by a corresponding integer, for example (55, 23, 43). First, k-mers are extracted from each of the sequences, which are then assembled into a sentence-like structure to form a string. These encoded sequences are used with an embedding layer during training. This representation technique preserves positional information capturing local context and sequential dependencies, while explicitly exhibiting the multiplicity of amino acids (when $k = 3$) that are present in the DNA sequence. This aids in reflecting biologically meaningful patterns such as codon usage or regulatory element structures.
2. **Label Encoding:** In this encoding technique, each nucleotide is assigned a numerical value, for example, $A = 1$, $C = 2$, $G = 3$, and $T = 4$ [6]. Each nucleotide is then replaced with its corresponding numerical value. Thus, the sequence *AAGCT* would become 11324. The encoded sequences are then fed into an embedding layer for training. While this encoding preserves positional information about the nucleotides in a DNA sequence, the encoded data implies an ordinal relationship between the nucleotides that does not exist. Nucleotides are categorical entities without inherent ordering, and label encoding fails to preserve their identity or any biologically meaningful relationship. As such, it does not capture known structural or functional properties of DNA and may degrade model performance in tasks requiring the learning of sequence dependencies. For example, a model may mistakenly learn patterns based on numeric proximity, treating C and G as closer than A and T, or that the difference between A and G is 2, which does not correspond to any biological meaning.
3. **k-mer One-Hot Vector:** One-hot k-mer encoding preserves the exact identity of nucleotide motifs, which is biologically relevant since many functional elements (e.g., transcription factor binding sites, splice signals) depend on short, specific nucleotide patterns. Instead of creating a sentence-like string

from the k-mers extracted from the DNA sequence, this method constructs a one-hot vector matrix for each k-mer token [7]. For $k = 3$, a dictionary of 4^3 k-mers (one for each nucleotide combination of length 3) is created, and each k-mer is assigned a one-hot vector representation. For example, the sequence *AAGCT* becomes $[[0,0,1,\ldots,0],[0,1,0,\ldots,0],[1,0,0,\ldots,0]]$ of shape (3×64). This representation eliminates the implied ordinal relationship between the features in the input data while preserving the positional data and exhibiting the multiplicity of amino acids in the DNA sequence.

3.2.2 Visual Representations

1. **ColorSquare:** The ColorSquare encoding technique, also known as the *five-color map*, is derived from the representation proposed by [43]. A (20×25)-cell color grid represents the DNA sequence of length 500, where each nucleotide is represented by a square cell of the assigned color, namely, A = red, G = blue, T = green, and C = yellow, in a spiral pattern. Fig. 2a shows an example of a DNA sequence represented in ColorSquare from H3, while Fig. 2b shows one from H4. The square shape of this pictorial representation can be converted easily into a tensor, which is ideal for training neural networks. The compact nature of this representation technique allows us to represent long DNA sequences in a limited space without any loss of information. Furthermore, this representation does not imply an ordinal relationship between the features in the input data but does preserve the positional information.
2. **FCGR** (Frequency Chaos Game Representation) is a graphical representation of a sequence [44] that is used to convert a long 1D sequence into graphical form. The frequencies of the k-mers derived from DNA sequences are represented in the FCGR matrix, which can be utilized to create a grayscale image called the *sequence fingerprint*, as shown in Fig. 3a, which shows a sample from the H3 dataset, while Fig. 3b shows a sample from the H4 dataset. FCGR exhibits unique patterns to represent genomes that can be used to identify genome fragments. FCGR patterns can also be used to compare sequences. The algorithm to generate an FCGR of a DNA sequence is shown below.
   (a) Start from position (0, 0).
   (b) Four nucleotides are located at the four corners:
      - A: (−1, 1) upper left
      - T: (1, −1) lower right
      - G: (1, 1) upper right
      - C: (−1, −1) lower left
   (c) For each nucleotide in reverse order, move and mark the new location halfway between the current location and the nucleotide. For example, if the last letter is T, the position changes from (0, 0) to the midpoint between (1, −1) and (0, 0), which is (0.5, −0.5).
   (d) Repeat this procedure for all nucleotides in the k-mer.

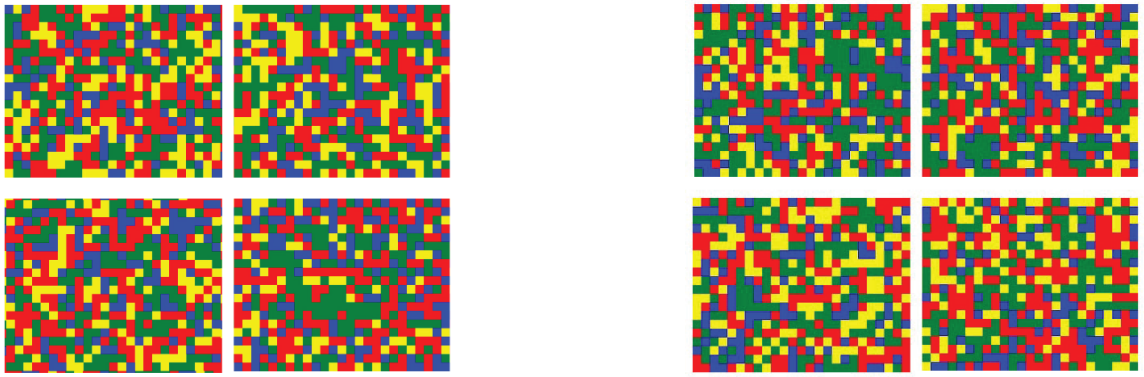


Figure 2: ColorSquare representation. (**a**) H3 ColorSquare example; (**b**) H4 ColorSquare example

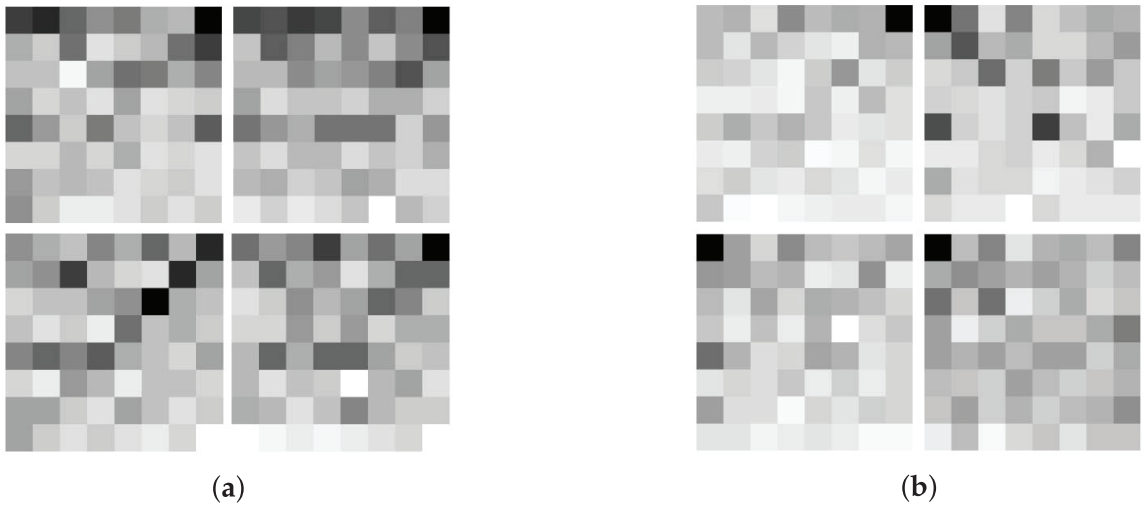


Figure 3: FCGR representation. (**a**) H3 FCGR example; (**b**) H4 FCGR example

3.3 Classification Models

In this paper, we use six different DL architectures (CNN (two), CNN-LSTM, CNN-BILSTM, ResNet50 and InceptionV3) to classify the various representations of DNA sequences based on the type of histone occupancy present. In addition, as a baseline, we include a simple fully connected neural network. Below, we present a brief description of each architecture utilized.

3.3.1 Numerical Representations

1. **CNN:** CNNs are among the best learning algorithms for understanding complicated structures. They have shown exceptional performance in tasks including picture segmentation, object detection, and computer vision. Compared to other DL models, CNNs require fewer parameters. As a result, the complexity of the model is reduced and the learning process is improved. We used two variants of a CNN with 1D convolutional layers.

Each model has an input size of (n, m), where n and m are the numbers of training instances and nucleotides in the DNA sequence, respectively. The last fully connected layer responsible for classifying instances uses the Softmax activation function, whereas the rest of the fully connected and convolutional layers use the Rectified Linear Unit (ReLU) function.

Various overfitting techniques were employed, such as early stopping and Dropout. The models were trained for an average of 25 epochs. The batch size used for the training process was 32 and the optimizer was Adam.

(a) CNN-1 processes input images of size $224 \times 224 \times 3$ through a series of convolutional layers. The model begins with two convolutional layers, Conv2D-1 and Conv2D-2, with 64 and 128 filters respectively, both using a 3×3 kernel size, LeakyReLU activation ($\alpha = 0.01$), and Batch Normalization (BatchNorm) to stabilize training. A dropout layer (0.3) follows to prevent overfitting. The next convolutional block consists of Conv2D-3 and Conv2D-4 with 256 and 512 filters, again using 3×3 kernels, LeakyReLU activation, and BatchNorm. Another dropout layer (0.3) is applied before the model transitions to a Global Average Pooling (GAP) layer, which reduces the spatial dimensions efficiently. The fully connected layers include Dense-1 with 512 units, LeakyReLU activation, and a dropout rate of 0.3 to further mitigate overfitting. Finally, the Dense-2 output layer applies a softmax activation for multi-class classification. The Adam optimizer is used to optimize the learning process, ensuring efficient weight updates during training. The architecture of CNN-1 is shown in Fig. 4.

(b) CNN-2 incorporates parallel convolutional layers and a self-attention mechanism to enhance feature extraction. The input layer processes images of size $224 \times 224 \times 3$. The initial feature extraction is performed by three parallel convolutional layers: Conv2D-1 (3×3 kernel), Conv2D-2 (5×5 kernel), and Conv2D-3 (7×7 kernel), each with 64 filters, LeakyReLU activation ($\alpha = 0.01$), and BatchNorm. The outputs of these parallel layers are concatenated, followed by a self-attention mechanism to capture long-range dependencies. Subsequently, Conv2D-4 applies 128 filters with a 3×3 kernel, LeakyReLU activation, and BatchNorm. A dropout layer (0.3) is added for regularization. Conv2D-5 further refines the extracted features with 256 filters and a 3×3 kernel, followed by another dropout layer (0.3). The model then employs a GAP layer to reduce spatial dimensions before transitioning to fully connected layers. Dense-1 consists of 512 units with LeakyReLU activation and a dropout rate of 0.3. Finally, the Dense-2 output layer uses a softmax activation for multi-class classification. The model is optimized using the Adam optimizer for efficient weight updates. Fig. 5 shows the architecture of CNN-2.

2. **CNN-LSTM:** The given neural network architecture combines Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM) and an attention mechanism to process image inputs of size $224 \times 224 \times 3$. It starts with four convolutional layers (Conv2D-1 to Conv2D-4) using 64, 128, 256, and 512 filters, respectively, with a 3×3 kernel size and LeakyReLU activation ($\alpha = 0.01$), along with BatchNorm after Conv2D-2 and Conv2D-4. Two max-pooling layers with a 2×2 filter size reduce spatial dimensions, while dropout layers (0.2) help prevent overfitting. The extracted features are flattened and passed to an LSTM-based sequential model consisting of two LSTM layers with 128 and 64 units using the hyperbolic tangent (*tanh*) activation. A Seq2Seq attention mechanism enhances feature importance before feeding the data into two fully connected layers: Dense-1 with 512 units (LeakyReLU, dropout 0.2) and Dense-2 as the output layer with a softmax activation for multi-class classification. The Adam optimizer is used for efficient training, making this architecture well-suited for tasks involving spatial and sequential dependencies in image-based data. Fig. 6 shows the architecture of CNN-LSTM.
3. **CNN-BiLSTM:** integrates CNN, BiLSTM, and self-attention to extract spatial and sequential features for classification. The input is an image of size $224 \times 224 \times 3$. The model begins with Conv2D-1 (64

filters, 3 × 3 kernel, LeakyReLU α = 0.01), followed by Conv2D-2 (128 filters, 3 × 3 kernel, BatchNorm, LeakyReLU). A BatchNorm layer is then applied, followed by MaxPooling (2 × 2) and Dropout (0.3). Further feature extraction is performed using Conv2D-3 (256 filters, 3 × 3 kernel, LeakyReLU) and Conv2D-4 (256 filters, 3 × 3 kernel, BatchNorm, LeakyReLU), followed by another BatchNorm layer, MaxPooling (2 × 2), and Dropout (0.3). The extracted features are flattened and passed to BiLSTM-1 (128 units, *tanh*) and BiLSTM-2 (64 units, *tanh*), with a self-attention mechanism capturing long-range dependencies. The final classification layers include Dense-1 (512 units, LeakyReLU α = 0.01, Dropout 0.3) and Dense-2 (softmax). The model is optimized using Adam with OneCycleLR for adaptive learning rate scheduling. Fig. 7 shows the architecture of CNN-BiLSTM.

4. **FCNN:** The model comprises an input layer followed by three hidden fully connected layers with 256, 128, and 64 neurons, respectively, each utilizing the ReLU activation function. It also incorporates two dropout layers with a rate of 0.3 after both the first and second hidden layers. Dropout was added to reduce overfitting and enhance generalization. The final output layer consists of two neurons with a Softmax activation function. The network was trained using the Adam optimizer and the categorical cross-entropy loss function.

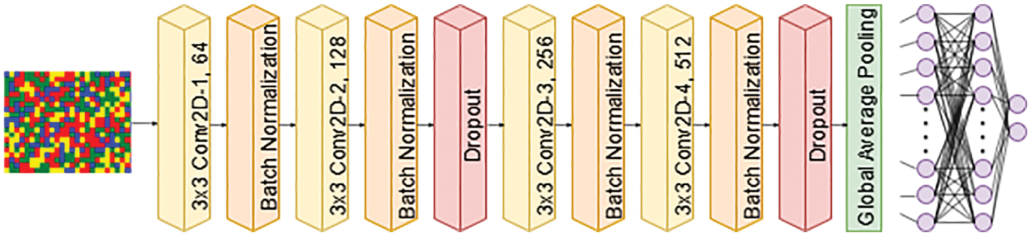


Figure 4: CNN-1 architecture

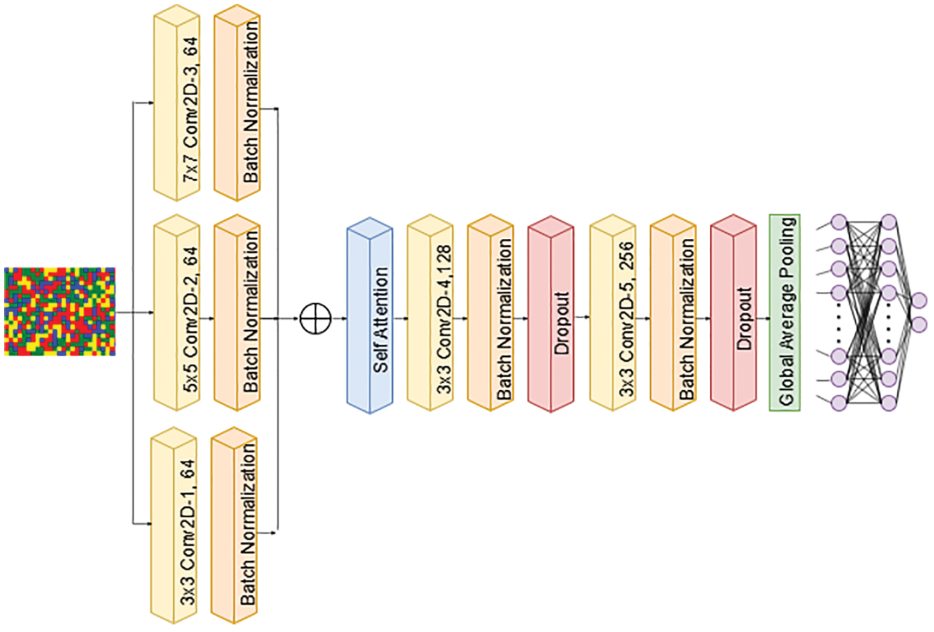


Figure 5: CNN-2 architecture

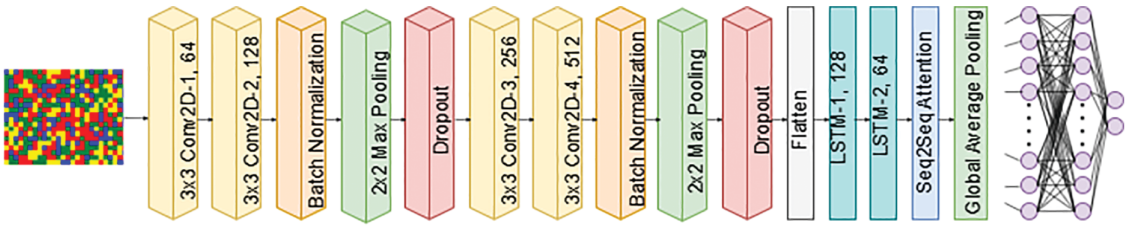


Figure 6: CNN-LSTM architecture

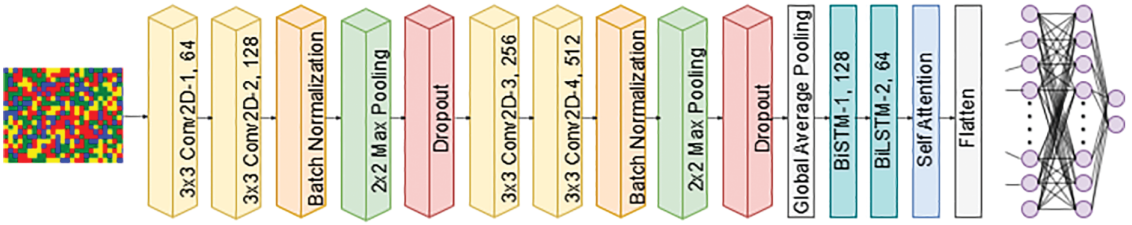


Figure 7: CNN-BiLSTM architecture

3.3.2 Visual Representations

1. **ResNet50:** ResNet-50 [12] is a variant of the ResNet architecture that has 48 convolutional layers, one maximum pooling layer, and one average pooling layer. It is commonly used for image classification owing to its performance on the ImageNet validation dataset (Fig. 8).
2. **InceptionV3:** InceptionV3 [13] is a superior and more computationally efficient version of the basic InceptionV1 model, introduced as GoogLeNet in 2014. Unlike previous versions, InceptionV3 features smaller asymmetric convolutions to reduce dimensionality and auxiliary classifiers to combat the issue of vanishing gradients (Fig. 9).

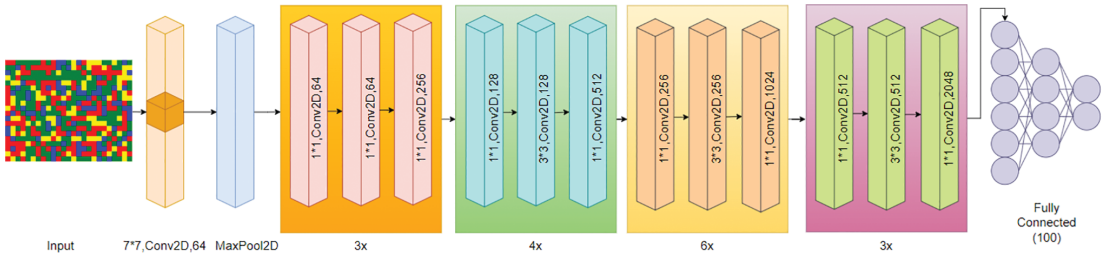


Figure 8: ResNet50 architecture

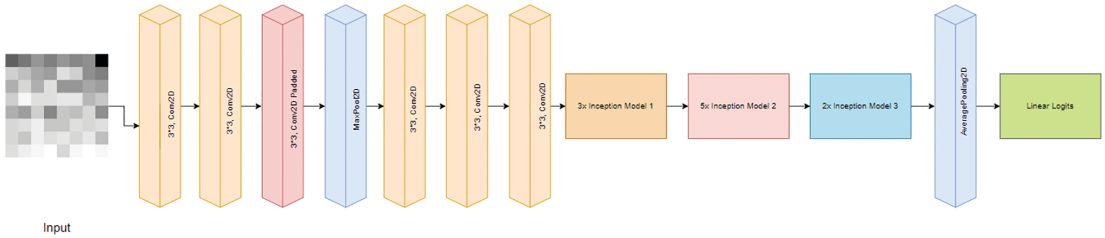


Figure 9: InceptionV3 architecture

4 Experimental Results

Training and evaluation of the deep learning models included in this paper were conducted on a Linux-based system on Google Cloud's Vertex AI platform. The machine had 4 CPUs, 15 GB RAM, and an NVIDIA Tesla T4 GPU. We used JupyterLab as the IPython notebook interface for pre-processing and modeling our data. The number of training epochs was set to 25 based on preliminary observations, where the training loss consistently followed a negative exponential trend and converged without notable fluctuations. No signs of overfitting were detected, and model performance remained stable across configurations. This choice also reflected a practical trade-off to optimize computational resources, given the large number of experimental scenarios evaluated in this study. The Adam optimizer was used in this study due to its efficient and stable convergence characteristics, particularly useful in handling the high-dimensional and sparse nature of biological sequence data. While alternative optimizers, such as stochastic gradient descent (SGD), are known for their generalization capabilities, Adam enables consistent training across a range of model and input configurations with minimal tuning. As the objective of this work was exploratory, focusing on comparative evaluation rather than fine-tuned optimization, Adam provided a practical balance between performance and training efficiency. The performance of the proposed models was evaluated using the following measures [45]: accuracy (Acc), which reflects the percentage of correctly predicted sequences; sensitivity (Sen), which indicates the fraction of actual positive sequences that were classified correctly; specificity (Spe), which measures the rate of correctly classified negative instances, and area under the curve (AUC), which visually represent the model's ability to distinguish between classes. We also report the $F1$-score, the harmonic mean of the precision and recall, as well as the training time of the models in seconds. We used non-parametric statistical test to analyze the $F1$ scores, namely the Friedman test for multiple related samples, and the Wilcoxon signed-rank test for two related samples, using SPSS Statistics Version 31.0.0.0. We also employed the Nemenyi test for post-hoc analysis [46].

4.1 Hyperparameter Optimization

We used GridSearch hyperparameter optimization approach. GridSearch ranges were adapted from prior studies in the literature addressing similar biological sequence classification tasks [27,47]. This approach ensured alignment with established practices while avoiding the computational overhead of extensive hyperparameter optimization, consistent with the exploratory scope of this study. Table 1 shows the details of the GridSearch inputs and best values for custom-designed models. The hyperparameter tuning of transfer learning architectures was performed across specified ranges: Learning Rate 0.001, 0.0005, 0.0001, Dropout Rate 0.2, 0.3, 0.4, and Batch Size 16, 32, 64. The remaining hyperparameters were derived from the original implementations by the respective authors. The optimal hyperparameters identified were as follows: for Inception V3, Learning Rate 0.001, Dropout Rate 0.2, and Batch Size 32; for ResNet50, Learning Rate 0.0001, Dropout Rate 0.4, and Batch Size 32.

Table 1: GridSearch hyperparameter optimization

Hyperparameter	Range	Best
	CNN-1	
Kernel Size	$\{3 \times 3, 5 \times 5, 7 \times 7\}$	3×3
Dropout Rate	$\{0.2, 0.3, 0.4, 0.5\}$	0.3
Learning Rate	$\{0.01, 0.001, 0.0001\}$	0.0001
ReduceLROnPlateau Factor	$\{0.1, 0.5\}$	0.1

(Continued)

Table 1 (continued)

Hyperparameter	Range	Best
Batch Size	{16, 32, 64}	32
	CNN-2	
Kernel Size	{3 × 3, 5 × 5, 7 × 7}	3 × 3
Dropout Rate	{0.2, 0.3, 0.4, 0.5}	0.3
Learning Rate	{0.01, 0.001, 0.0001}	0.001
Weight Penalty (L2)	{0.0001, 0.0005, 0.001}	0.0005
Batch Size	{16, 32, 64}	32
	CNN-LSTM	
Kernel Size	{3 × 3, 5 × 5}	3 × 3
Dropout Rate	{0.2, 0.3, 0.4, 0.5}	0.2
Learning Rate	{0.01, 0.001, 0.0001}	0.0001
LSTM Units	{64, 128, 256}	128
Batch Size	{16, 32, 64}	32
	CNN-BiLSTM	
Kernel Size	{3 × 3, 5 × 5}	3 × 3
Dropout Rate	{0.2, 0.3, 0.4, 0.5}	0.3
Learning Rate	{0.001, 0.0005, 0.0001}	0.0005
BiLSTM Units	{64, 128, 256}	128
Batch Size	{16, 32, 64}	32

4.2 Results

As mentioned in the introduction, the goal of our experiments is to demonstrate the impact of data representation on the performance of deep learning models for the classification of DNA sequences. The three dataset used in this study are H3 occupancy, H4 occupancy and DNA Sequence Dataset.

Seven deep learning architectures were developed: CNN-1, CNN-2, CNN-LSTM, CNN-BiLSTM, FCNN, ResNet50, and InceptionV3, in order to classify the DNA sequences into two classes (i.e., positive or negative) for H3 and H4, while into seven classes for DNA sequence dataset. Numerical and visual data representation techniques are utilized to represent the input datasets, including: k-mer sentence encoding, label encoding, k-mer one-hot vector, ColorSquare, and FCGR. All models were trained using 10-fold cross-validation. In the following subsections, we present the performance results of the developed models with respect to each type of data representation.

4.2.1 Numerical Representations

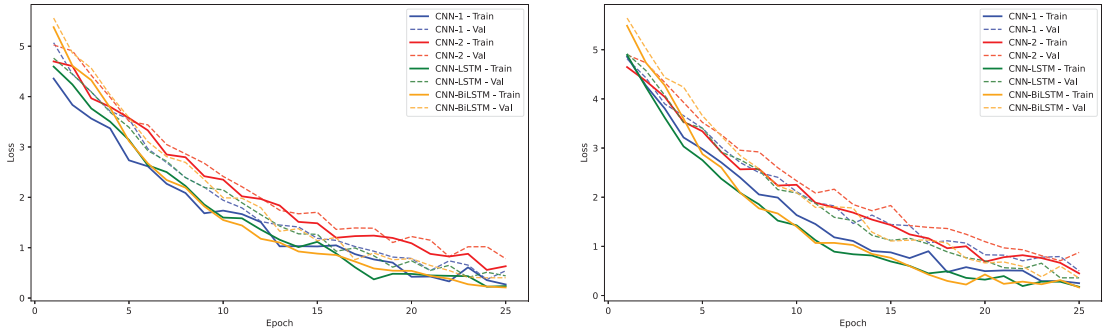
The performance comparison across different encoding schemes reveals notable trends in classification accuracy, sensitivity, specificity, and the $F1$-score for various models. Results are presented for each dataset quantitatively in tabular format and visually in the form of loss plots and AUC plots. For H3 and H4 dataset, when employing k-mer sentence encoding (Table 2, Figs. 10–13), the CNN-LSTM outperformed other models over the H3 dataset, achieving the highest accuracy of 0.8944, while CNN-BiLSTM excelled on the H4 dataset with an accuracy of 0.9000. Both models demonstrated superior sensitivity and specificity,

with CNN-LSTM showing a sensitivity of 0.8700 for H3 and specificity of 0.9201. In terms of F1 scores, CNN-LSTM maintained the highest values, reflecting a strong balance between precision and recall. Conversely, FCNN was the fastest model, completing tasks in 281.03 s for H3, whereas the more complex CNN-LSTM and CNN-BiLSTM required significantly longer training times. This suggests that the hybrid architecture effectively captures sequential dependencies in the encoded sequences, leading to improved classification performance. A similar trend is observed with label encoding (Table 3), where CNN-BiLSTM attained the highest accuracy of 84.90% for H3, while CNN-BiLSTM demonstrated the best performance for H4 with an accuracy of 84.45%. The F1 scores reflected a good balance between precision and recall for both models. However, specificity and *F*1 scores were generally lower in comparison to the *k*-mer sentence encoding method, suggesting that label encoding may not be as effective in distinguishing between classes, particularly for H4. The results from *k*-mer one-hot vector encoding (Table 4) indicate a substantial improvement in performance across all models. CNN-LSTM achieved the highest accuracy of 92.95% for H3 and 94.66% for H4, accompanied by the highest sensitivity and specificity values. This demonstrates the robustness of the one-hot encoding scheme, which likely preserves more discriminative information compared to the other encoding strategies. In contrast, FCNN was significantly faster across all models and presentations, at the expense of the effectiveness. The training loss curves (Figs. 10 and 12) followed the negative exponential behavior which is an indication of normal training process. Given that validation loss (dotted lines) followed the training loss (solid lines), there was no overfitting during the training process. From the training plots, there is a marginal improvement among the models, however, Table 2 provides a clear picture of which model performed best on validation dataset. The bold typeface in the table denotes the best score. Further to this, the AUC plots (Figs. 11 and 13) also demonstrateed that CNN-LSTM model performed best with AUC of 0.90 for *k*-mer sentence encoding, AUC of 0.85 for label encoding and AUC of 0.93 for one-hot vector encoding.

Table 2: Performance comparison using *k*-mer sentence encoding

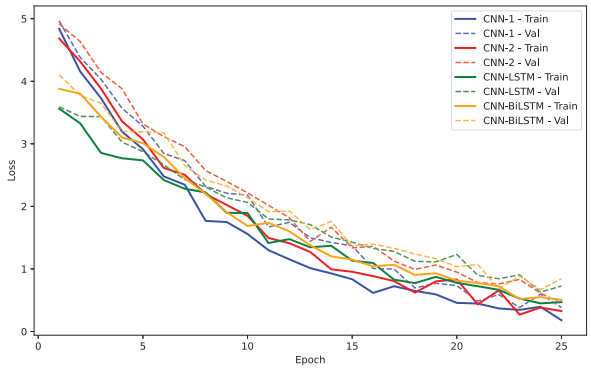
Model	H3					H4				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
CNN-1	0.8498	0.8400	0.8600	0.8514	304.62	0.8722	0.8500	0.8899	0.8551	**293.81**
CNN-2	0.8346	0.8199	0.8500	0.8355	337.85	0.8643	**0.8699**	0.8599	0.8506	332.41
CNN-LSTM	**0.8944**	**0.8700**	**0.9201**	**0.8941**	484.14	0.8911	0.8299	**0.9399**	0.8712	476.29
CNN-BiLSTM	0.8844	0.8600	0.9101	0.8840	509.49	**0.9000**	0.8500	**0.9399**	**0.8829**	505.68
FCNN	0.8005	0.7093	0.8922	0.7809	**281.03**	0.8275	0.8166	0.8401	0.8363	300.96

Note: The bold typeface in the table denotes the best score.



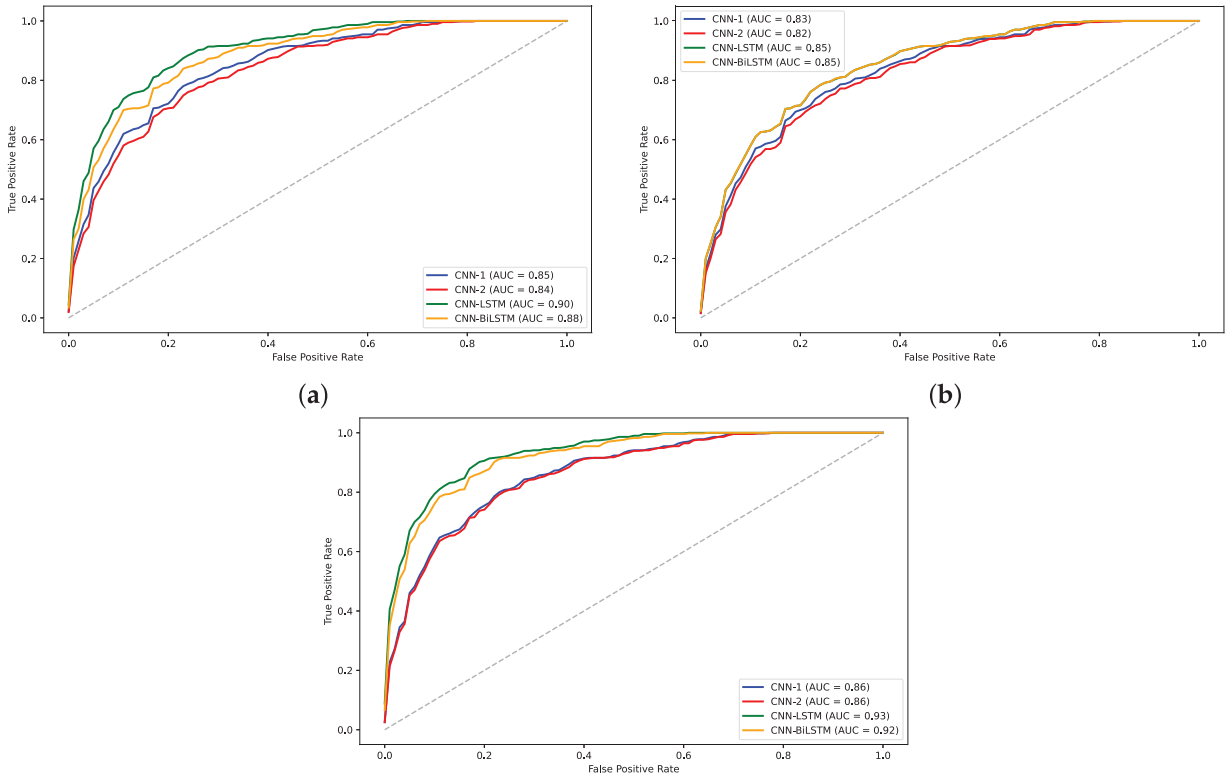
(a) (b)

Figure 10: (Continued)



(**c**)

Figure 10: Training. loss and validation loss plots for models on H3 dataset: (**a**) *k*-mer sentence encoding; (**b**) label encoding; (**c**) one-hot vector encoding



(**a**) (**b**)

(**c**)

Figure 11: AUC plots for models on H3 dataset: (**a**) *k*-mer sentence encoding; (**b**) label encoding; (**c**) one-hot vector encoding

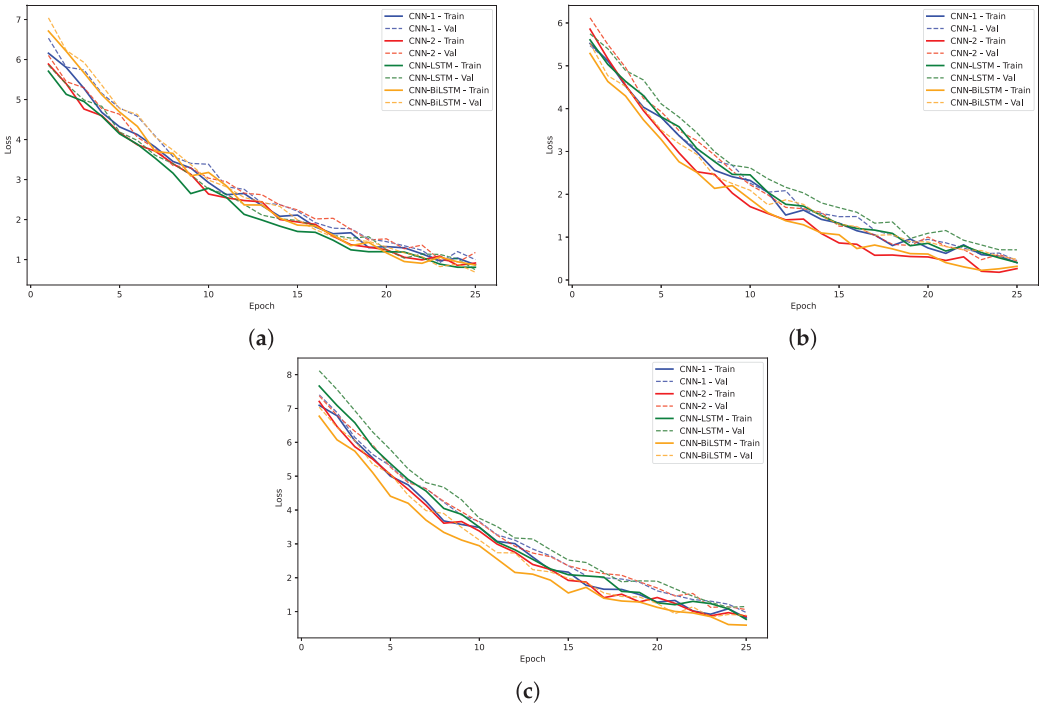


Figure 12: Training loss and validation loss plots for models on H4 dataset: (**a**) *k*-mer sentence encoding; (**b**) label encoding; (**c**) one-hot vector encoding

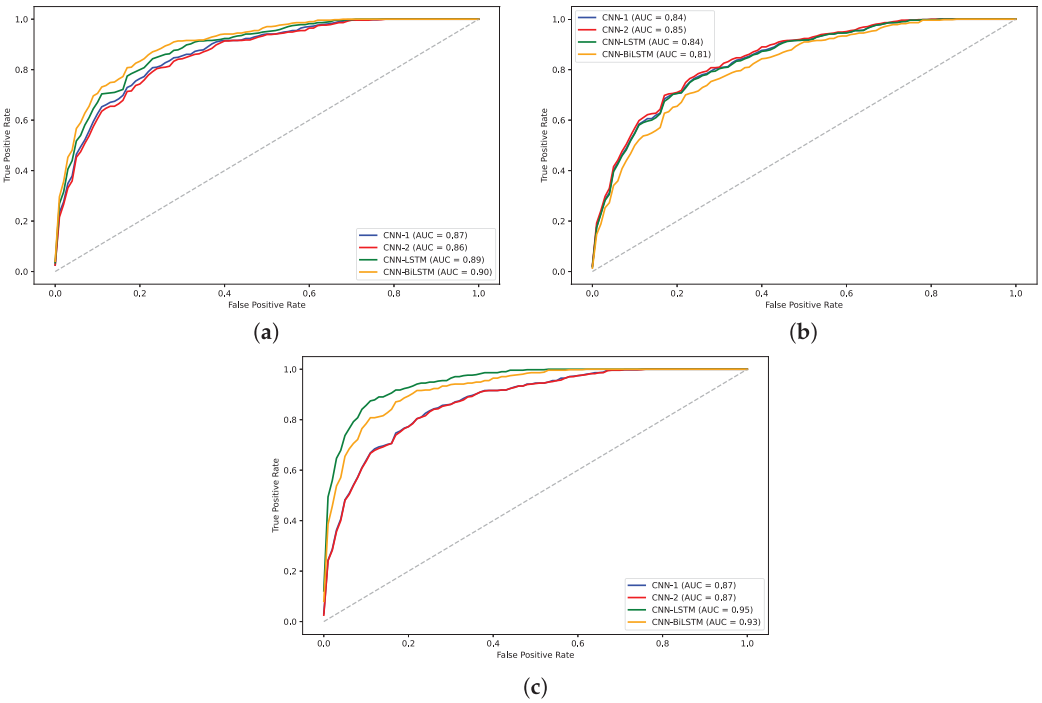


Figure 13: AUC plots for models on H4 dataset: (**a**) *k*-mer sentence encoding; (**b**) label encoding; (**c**) one-hot vector encoding

Table 3: Performance comparison using label encoding

Model	H3					H4				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
CNN-1	0.8244	0.8000	0.8500	0.8236	282.66	0.8355	0.7799	0.8799	0.8080	271.39
CNN-2	0.8146	0.8000	0.8300	0.8156	325.61	0.8333	**0.8000**	0.8599	**0.8099**	310.1
CNN-LSTM	0.8436	0.7899	**0.9000**	0.8381	460.48	**0.8445**	0.7000	**0.9599**	0.7999	445.29
CNN-BiLSTM	**0.8490**	**0.8100**	0.8900	**0.8461**	491.26	0.8300	0.6799	0.9498	0.7803	468.43
FCNN	0.5369	0.6400	0.4334	0.5808	**30.02**	0.5454	0.9473	0.0743	0.6922	**29.54**

Note: The bold typeface in the table denotes the best score.

Table 4: Performance comparison using k-mer one-hot vector encoding

Model	H3					H4				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
CNN-1	0.8541	0.8199	0.8900	0.8521	362.98	0.8777	0.8500	0.8998	0.8605	354.33
CNN-2	0.8495	0.8300	0.8700	0.8497	415.61	0.8765	0.8598	0.8899	0.8608	407.28
CNN-LSTM	**0.9295**	**0.9099**	**0.9501**	**0.9297**	542.87	**0.9466**	**0.9299**	**0.9599**	**0.9392**	534.37
CNN-BiLSTM	0.9196	0.9000	0.9401	0.9198	584.11	0.9365	0.9199	0.9498	0.9279	572.84
FCNN	0.7718	0.7700	0.7736	0.7718	**301.04**	0.7669	0.7881	0.7420	0.7848	**335.08**

Note: The bold typeface in the table denotes the best score.

The analysis of model performance using label encoding indicates that while models like CNN-BiLSTM and CNN-LSTM achieve relatively high accuracies on the H3 dataset, the performance does not significantly improve on the H4 dataset. This inconsistency may arise from the ordinal relationships inherent in label encoding, which LSTMs typically exploit; when such relationships are lacking, the LSTM may learn misleading patterns, leading to degraded performance. Statistical analysis of the $F1$ scores from models derived using custom and hybrid architectures with the three numerical representation methods on the H3 and H4 datasets, conducted via the Friedman test, reveals a Chi-Square value of 6.800 and a p-value of 0.079. This indicates no statistically significant differences among the models after excluding FCNN. This suggests that variations in performance are not meaningful, highlighting the need for caution when employing label encoding with models designed to learn dependencies in DNA classification tasks.

Further evaluation using DNA Sequence Dataset (Table 5, Figs. 14–19) across Yeast, A. Thaliana, and Human datasets shows that the CNN architectures generally outperform CNN-LSTM models. Notably, CNN-2 attained the highest accuracy across several configurations, particularly in 4-mer and 8-mer encoding, achieving up to 99% accuracy in certain cases. The CNN-LSTM models, while effective in some settings, exhibited lower accuracy and sensitivity, particularly in lower-order k-mer encodings, suggesting potential limitations in capturing long-range dependencies in DNA sequences. Most models demonstrated enhanced performance with 4-mer and 8-mer encoding. CNN-LSTM and CNN-BiLSTM lagged behind, especially on the *A. thaliana* dataset. Training times varied, with CNN-2 requiring more time, indicating a trade-off between accuracy and computational efficiency. Overall, larger k values appear to capture more informative features for DNA classification.

Table 5: Performance comparison using k-mer sentence encoding on DNA Sequence Dataset

Model	Yeast					A. Thaliana					Human				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
2-Mer															
CNN-1	0.9350	0.9600	0.9100	0.9365	**175.89**	**0.9900**	**0.9900**	**0.9900**	**0.9900**	**547.89**	0.9450	0.9200	0.9700	0.9435	**564.56**
CNN-2	**0.9550**	**0.9800**	**0.9300**	**0.9560**	206.12	0.9500	0.9200	0.9800	0.9484	586.52	**0.9900**	**1.0000**	**0.9800**	**0.9900**	608.72
CNN-LSTM	0.8300	0.7800	0.8800	0.8210	258.94	0.6550	0.5800	0.7300	0.6270	728.64	0.7400	0.7700	0.7100	0.7475	736.78
CNN-BiLSTM	0.7800	0.7100	0.8500	0.7634	278.33	0.8750	0.8400	0.9100	0.8704	774.31	0.7900	0.7300	0.8500	0.7765	786.57
4-Mer															
CNN-1	0.9550	0.9800	0.9300	0.9560	**203.67**	**0.9900**	**0.9900**	**0.9900**	**0.9900**	**586.91**	0.9500	0.9300	0.9700	0.9489	**583.67**
CNN-2	**0.9650**	**0.9900**	**0.9400**	**0.9658**	224.61	0.9700	0.9900	0.9500	0.9705	624.82	**0.9950**	**1.0000**	**0.9900**	**0.9950**	630.76
CNN-LSTM	0.8700	0.8300	0.9100	0.8645	271.45	0.8100	0.7700	0.8500	0.8020	761.21	0.8950	0.8300	0.9600	0.8877	764.85
CNN-BiLSTM	0.8800	0.8200	0.9400	0.8723	308.96	0.8800	0.9200	0.8400	0.8846	814.26	0.7850	0.6800	0.8900	0.7597	818.91
8-Mer															
CNN-1	0.9250	0.9500	0.9000	0.9268	**225.27**	**0.9950**	**1.0000**	**0.9900**	**0.9950**	**618.22**	0.9800	0.9900	0.9700	0.9801	**627.81**
CNN-2	**0.9550**	**0.9800**	**0.9300**	**0.9560**	237.21	0.9750	0.9900	0.9600	0.9753	662.47	**0.9900**	**1.0000**	0.9800	0.9900	666.37
CNN-LSTM	0.8750	0.8300	0.9200	0.8691	307.84	0.9550	0.9300	0.9800	0.9538	798.29	0.9700	0.9500	**0.9900**	0.9693	804.58
CNN-BiLSTM	0.8800	0.8600	0.9000	0.8775	339.72	0.9000	0.9200	0.8800	0.9019	841.68	0.9450	0.9200	0.9700	0.9435	876.12

Note: The bold typeface in the table denotes the best score.

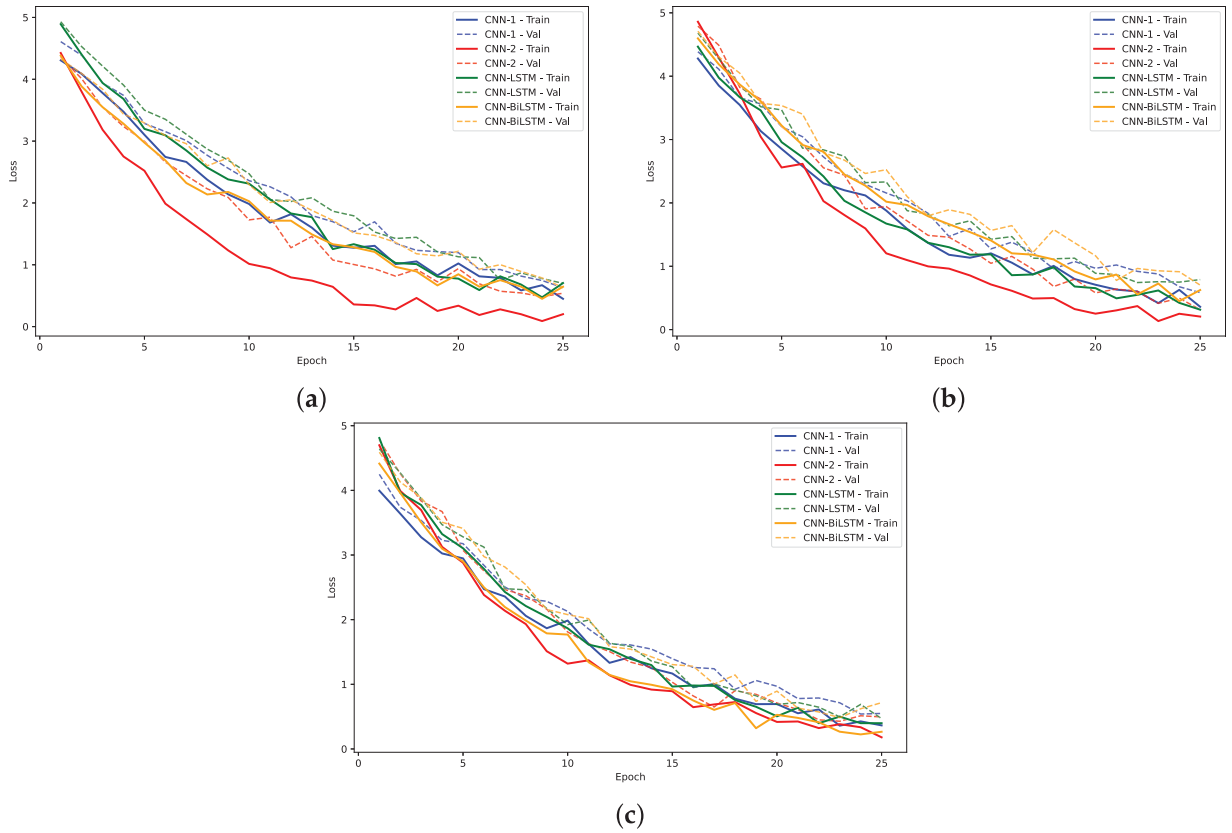


Figure 14: Training loss and validation loss plots for models on human dataset: (**a**) 2-mer sentence encoding; (**b**) 4-mer sentence encoding; (**c**) 8-mer sentence encoding

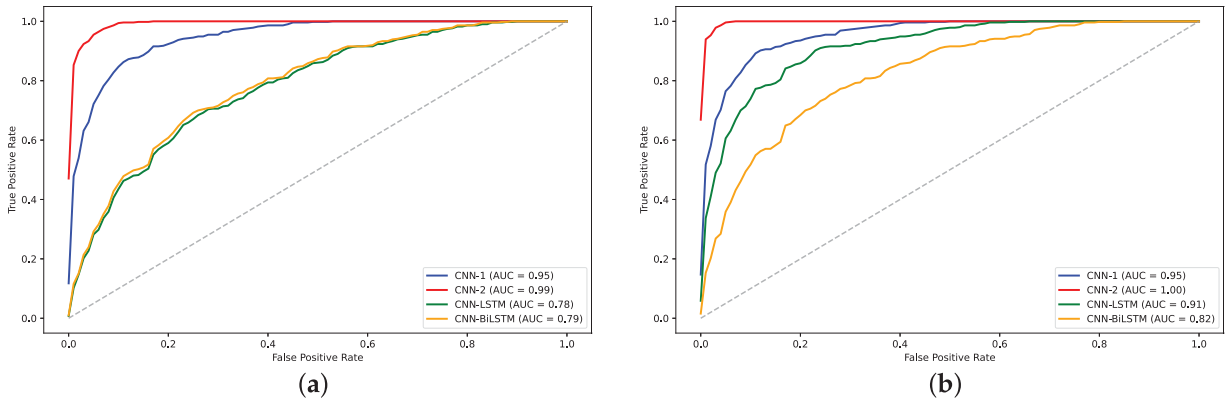
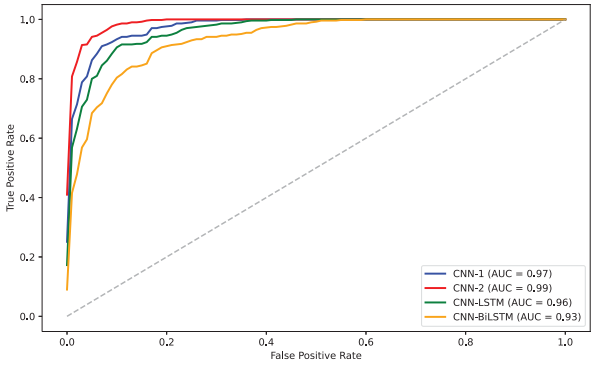
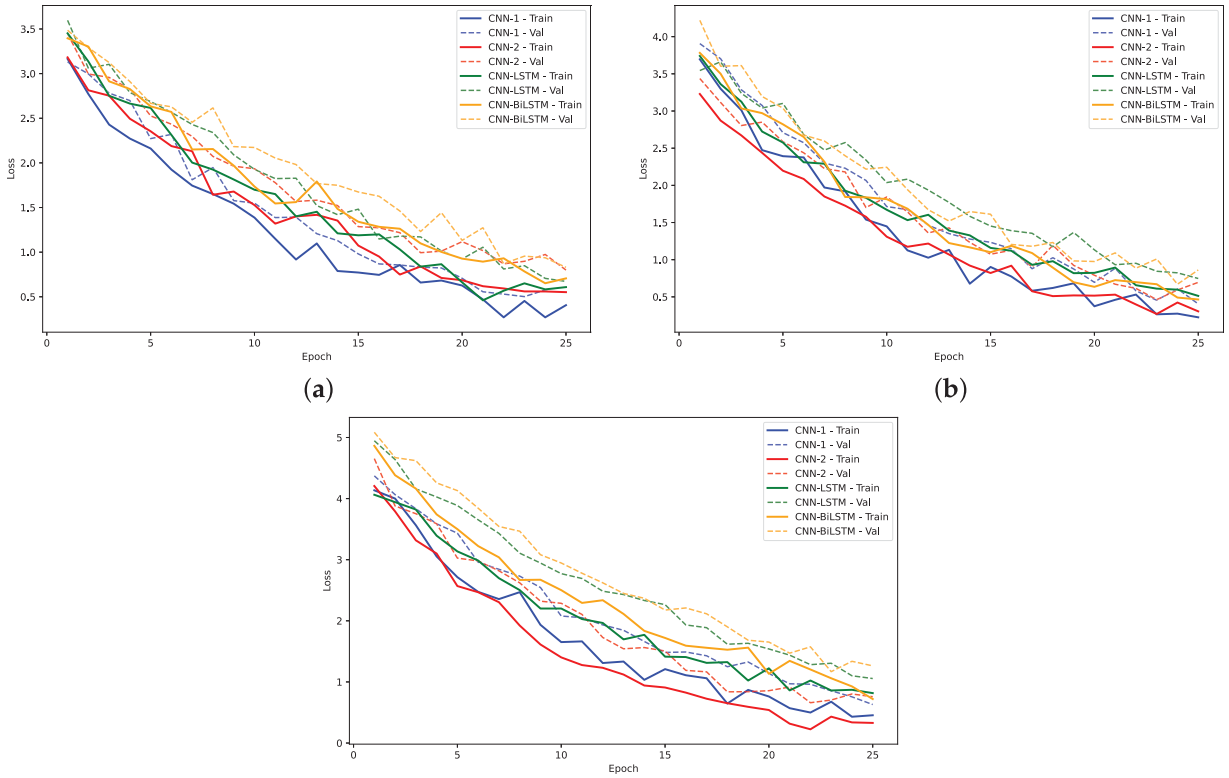


Figure 15: (Continued)



(**c**)

Figure 15: AUC plots for models on human dataset: (**a**) 2-mer sentence encoding; (**b**) 4-mer sentence encoding; (**c**) 8-mer sentence encoding



(**a**) (**b**)

(**c**)

Figure 16: Training loss and validation loss plots for models on yeast dataset: (**a**) 2-mer sentence encoding; (**b**) 4-mer sentence encoding; (**c**) 8-mer sentence encoding

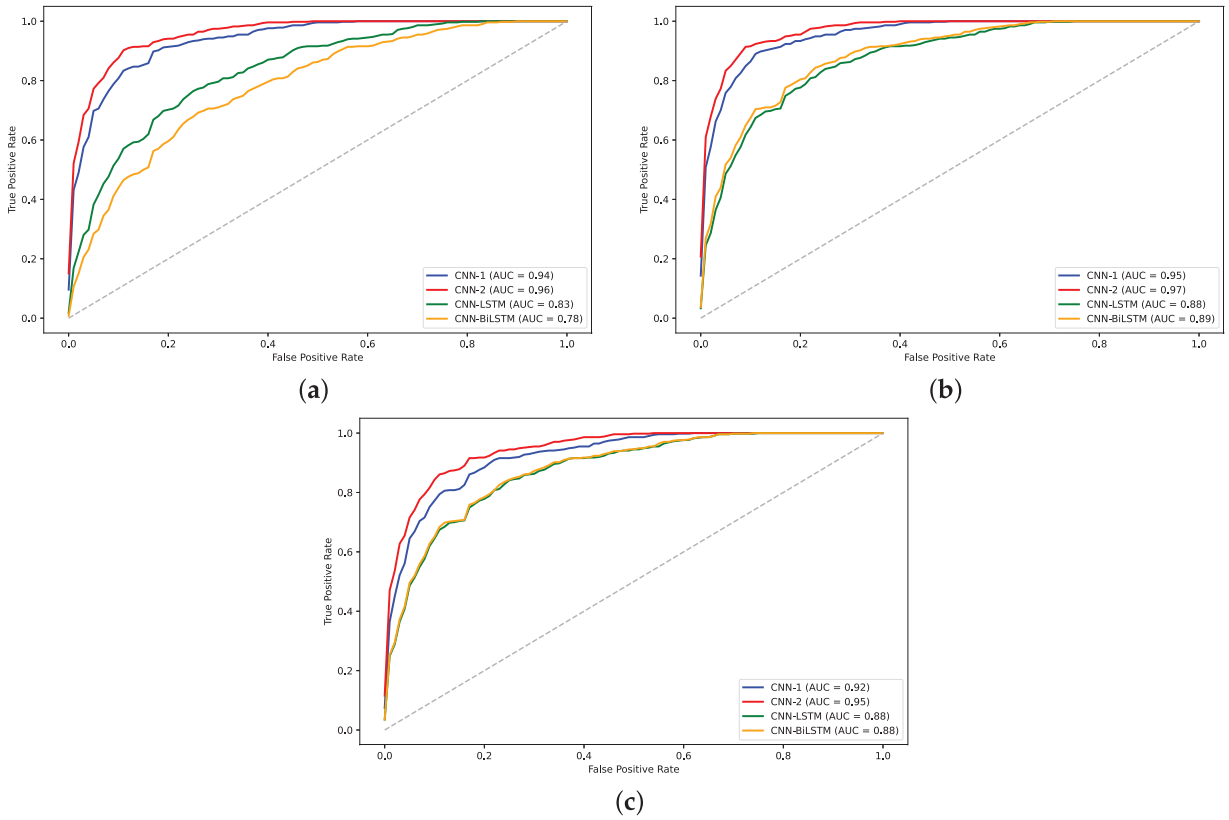


Figure 17: AUC plots for models on yeast dataset: (**a**) 2-mer sentence encoding; (**b**) 4-mer sentence encoding; (**c**) 8-mer sentence encoding

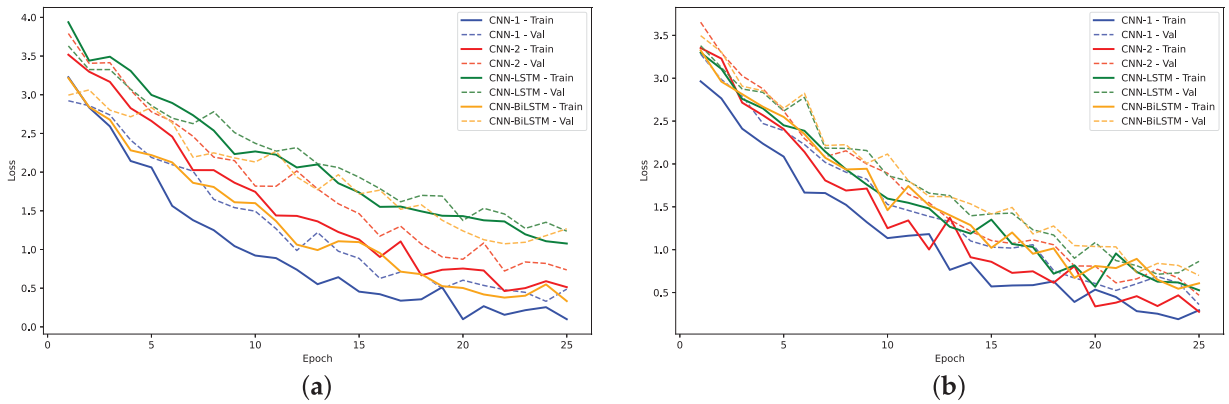
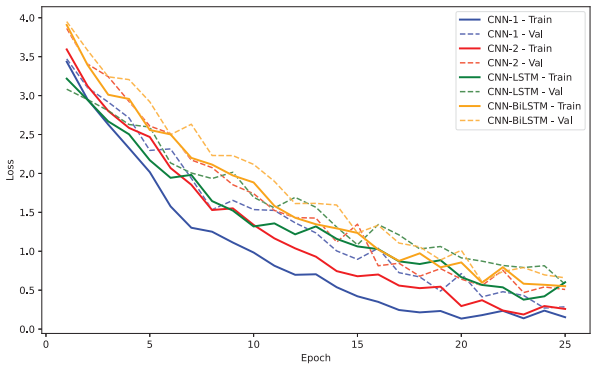
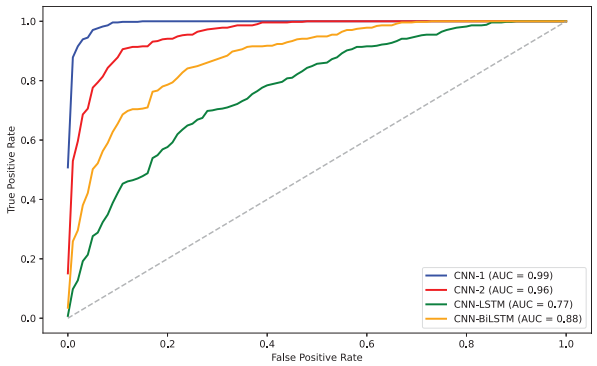


Figure 18: (Continued)

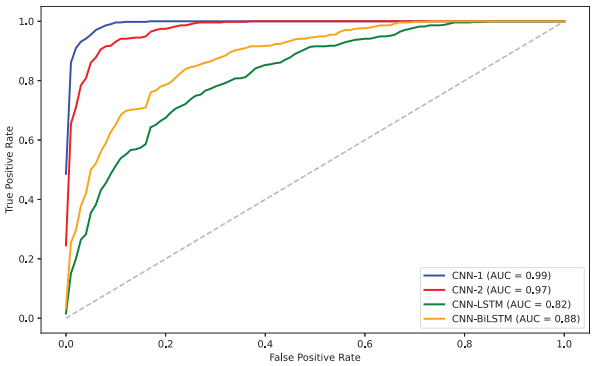


(**c**)

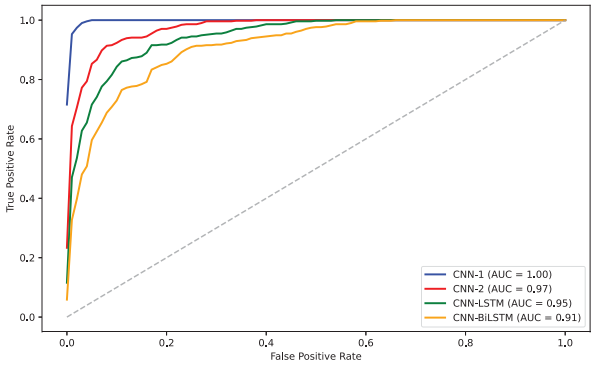
Figure 18: Training loss and validation loss plots for models on A. Thaliana Dataset: (**a**) 2-mer sentence encoding; (**b**) 4-mer sentence encoding; (**c**) 8-mer sentence encoding



(**a**)



(**b**)



(**c**)

Figure 19: AUC plots for models on A. Thaliana Dataset: (**a**) 2-mer sentence encoding; (**b**) 4-mer sentence encoding; (**c**) 8-mer sentence encoding

The training loss curves (Figs. 14, 16 and 18) verify the behavior given that the loss for the CNN models converged to lowest values. The negative exponential behavior is indicative of normal training behavior with validation loss following the training loss ensuring no overfitting. In addition, the AUC plots (Figs. 15, 17 and 19) verified the superior performance of CNN-2 model for human and yeast datasets while CNN-1

model for the A. Thaliana dataset with AUC value over 0.95. Table A1 presents a comparative analysis of performance using label encoding vs. one-hot vector encoding on the DNA Sequence Dataset.

The Friedman test results indicate a significant difference in the performance of the five models (CNN1, CNN2, FCNN, CNN-LSTM, and CNN-BiLSTM) across the various numerical data representations and datasets (H3 occupancy, H4 occupancy, Yeast DNA sequence, A. Thaliana, and Human DNA sequence). With a Chi-Square value of 28.610 and an asymptotic significance (p-value) of less than 0.001, we reject the null hypothesis, suggesting that at least one model's performance differs significantly from the others. The degrees of freedom (df = 4) confirm that this result is robust, indicating that the choice of model, data representation, or dataset has a notable impact on performance. The post-hoc analysis shows significant differences in performance between CNN1 and CNN2 compared to the other models (CNN-LSTM, CNN-BiLSTM, and FCNN). No significant difference was observed between CNN1 and CNN2, nor between CNN-LSTM and CNN-BiLSTM, or FCNN. Overall, the findings highlight the effectiveness of CNN-LSTM models in text-based k-mer encodings, whereas CNN architectures demonstrate superior performance in DNA Sequence Dataset classification. Hybrid models consistently require more training times in comparison to custum CNN models. Additionally, the choice of encoding significantly impacts model performance, with k-mer one-hot vector encoding emerging as the most robust approach across different datasets and tasks.

4.2.2 Visual Representations

In addition to numerical encoding methods, the evaluation of visual data representation techniques (Table 6, Figs. 20–23) highlights their impact on classification performance. Among the evaluated techniques, ColorSquare representation consistently resulted in poor performance across both datasets. Although FCGR provided some improvements, it still underperformed compared to CNN and CNN-LSTM models utilizing numerical representations such as label encoding, k-mer sentence encoding, and one-hot k-mer encoding. The highest performance was observed for ResNet50 model on H4 FCGR (i.e., 83.54%) lower than the 94.66% performance of CNN-LSTM for the H4 one-hot encoded vector. Conversely, the lowest performing model in terms of accuracy, sensitivity, and specificity was InceptionV3 with the ColorSquare representation technique applied to the H4 dataset. Additionally, ResNet50 showed more competitive training times, requiring 724.55 s on H3 and 702.69 s on H4, compared to InceptionV3's longer training times. These findings are in line with the AUC curves (Figs. 22 and 23) where the highest AUC of 0.84 was observed for the ResNet50 model on H4 FCGR. However, the training curves (Figs. 20 and 21) suggest that there was no overfitting (validation loss followed the training loss). These findings suggest that visual representation methods may not be as effective as numerical encoding strategies for this classification task. Table A2 provides a comparison of the performance of ResNet50 and Inception V3 based on visual data representations applied to the DNA Sequence Dataset. The Wilcoxon test results indicate a significant difference in the performance of ResNet50 and Inception V3 when evaluated using the two visual presentations across the same datasets. The Z-value of −2.803 suggests that ResNet50 performed significantly better than Inception V3. The asymptotic significance (p-value) of 0.005, which is below the conventional threshold of 0.05, confirms this finding. Therefore, we reject the null hypothesis and conclude that there is a statistically significant difference in the performance of the two models.

Table 6: Performance comparison using visual data representations

	COLORSQUARE									
	H3					H4				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
ResNet50	**0.8107**	**0.8400**	**0.7800**	**0.8198**	**782.96**	**0.7588**	**0.7199**	**0.7899**	**0.7260**	**763.14**
InceptionV3	0.7138	0.6699	0.7600	0.7058	886.58	0.6744	0.5799	0.7499	0.6126	858.29
	FCGR									
	H3					H4				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
ResNet50	**0.8269**	**0.9000**	0.7500	**0.8419**	**724.55**	**0.8354**	**0.8299**	**0.8399**	**0.8174**	**702.69**
InceptionV3	0.8059	0.8400	**0.7700**	0.8160	816.35	0.7966	0.7799	0.8099	0.7729	809.67

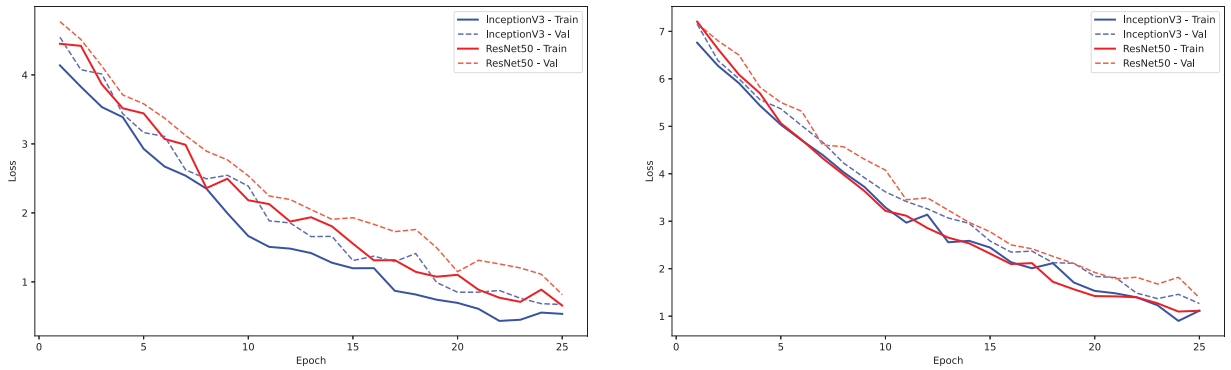


Figure 20: Training loss and validation loss plots for models on visual representations of H3 Dataset: (**a**) FCGR; (**b**) ColorSquare

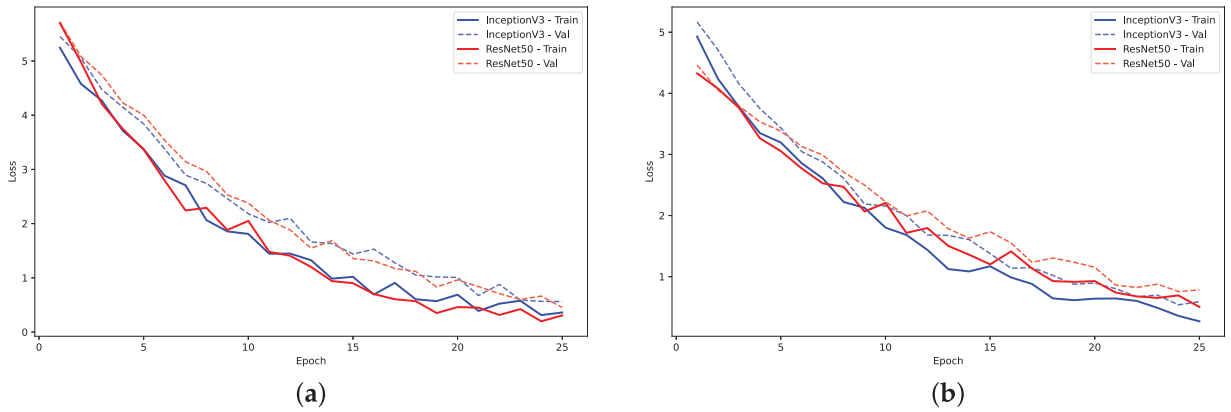


Figure 21: Training loss and validation loss plots for models on visual representations of H4 dataset: (**a**) FCGR; (**b**) ColorSquare

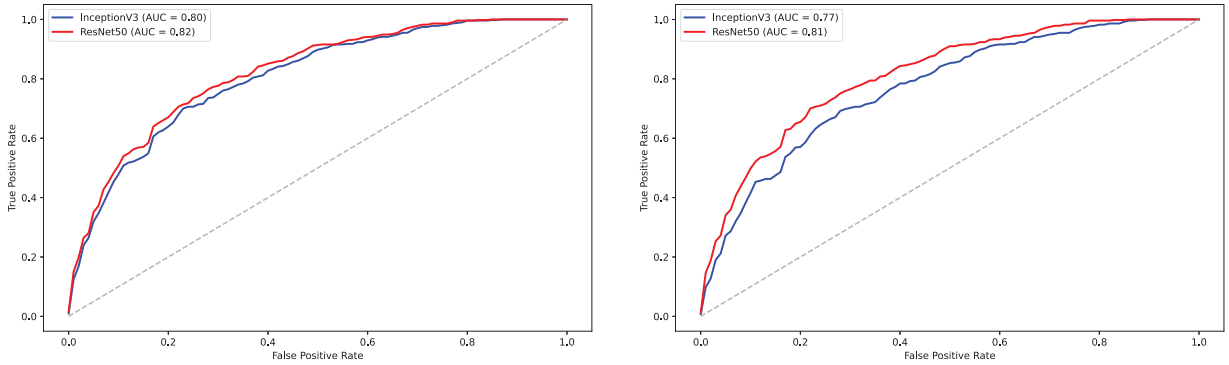


Figure 22: AUC plots for models on visual representations of H3 dataset: (**a**) FCGR; (**b**) ColorSquare

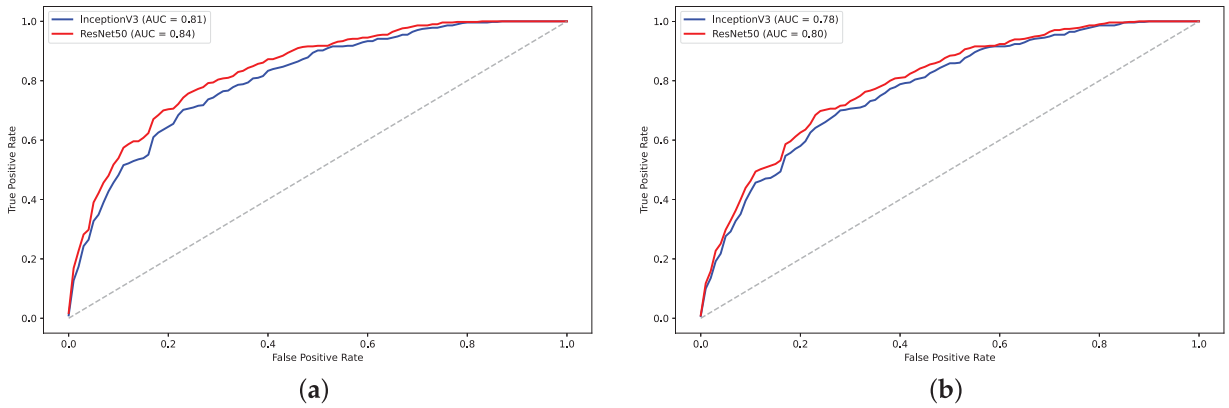


Figure 23: AUC plots for models on visual representations of H4 dataset: (**a**) FCGR; (**b**) ColorSquare

5 Discussion

This study empirically evaluates the impact of different data representation techniques on the performance of deep learning models in classifying DNA sequences. Specifically, our experiments focused on predicting key determinants of chromatin structure in DNA histone proteins. The results indicate that one-hot vector representation of DNA sequence k-mers yielded the highest overall classification accuracy, while the k-mer sequence encoding technique did not outperform the one-hot representation, it demonstrated comparable performance on certain models. Label-encoding yielded the worst results. In particular, the FCNN model showed that label-encoding was completely unreliable. These findings align with previous studies, such as [6], which also highlight the effectiveness of one-hot encoded k-mers in sequence-based classification tasks.

The comparison of training times from the tables indicates that the hybrid models, specifically CNN-LSTM and CNN-BiLSTM, consistently require more training time than their non-hybrid counterparts. This trend can be attributed to the increased complexity of these models, which combine convolutional layers with recurrent components to capture both spatial and temporal features in the data. While this complexity can enhance model performance in certain tasks, it also necessitates more computational resources and time for training. Considering the H3 and H4 datasets, the average training times for the models using different numeric representations indicate that label encoding is the most efficient, requiring only 318.01 s, followed by k-mer encoding at 382.63 s, and one-hot encoding at 441.05 s. The reduced complexity of label encoding

likely leads to faster computation. In contrast, one-hot encoding's longer training time can be attributed to the significant increase in dimensionality, which imposes greater computational demands. Further, The average training times for the models using the DNA sequence dataset indicate a clear trend: as the k value increases from 2-mer to 8-mer, the training time also increases, with averages of 521.11, 549.51, and 583.80 s, respectively. Larger k values increase complexity and training time, enhancing feature capture but posing efficiency challenges. As for visual representations, the average training times for ResNet50 and Inception V3 using the H3 and H4 datasets show that ColorSquare representation requires significantly more training time at 822.74 s compared to FCGR at 763.32 s. Notably, when considering results for the same architectures over the DNA sequence dataset, a similar observation can be seen.

Despite the theoretical appeal of visual representations, numerical encoding methods consistently delivered superior results across all experimental setups. The visual methods, although compact and maintaining positional information effectively, posed certain limitations when applied to standard, off-the-shelf deep learning models without domain-specific customization. The Frequency Chaos Game Representation (FCGR) outperformed the ColorSquare method, likely due to its ability to emphasize k-mer multiplicity within DNA sequences. Future research could investigate tailored deep learning architectures specifically designed to leverage these visual encoding advantages, potentially revealing conditions under which visual representations might outperform numerical methods.

The decision to employ transfer learning on image-based representations of DNA sequences was motivated by the capacity of pretrained CNNs to identify local and global dependencies efficiently, which is particularly valuable given the limited size and imbalance of biological datasets like H3, H4, and promoter regions. This approach bypasses the need for training complex custom models from scratch, which often require extensive hyperparameter tuning, large labeled datasets, and significant computational resources, making it a practical and scalable choice in bioinformatics applications where labeled data is scarce and sequence diversity is high. While custom models may offer more tailored solutions, they were beyond the scope of this study given the exploratory objectives and dataset size constraints.

Our results suggest that sequence-based representations emphasizing k-mer frequencies are particularly effective, as they capture the multiplicity of nucleotide patterns while preserving positional information. The superior performance of one-hot encoding may also stem from its ability to eliminate the artificial ordinal relationships implied by label-based and k-mer sentence encoding techniques. In particular, label encoding was observed to impair the performance of hybrid CNN-LSTM models, indicating that this method is suboptimal for tasks that rely on capturing sequential dependencies, as numeric labeling implies an ordinal relationship where none exists, degrading model performance. Embeddings such as Word2Vec or similar context-aware techniques (e.g., FastText, GloVe, or domain-specific embeddings) could potentially mitigate these limitations by effectively capturing semantic or contextual dependencies within DNA sequence data by mapping sequences into dense vector representations that preserve relational structure.

Our findings reinforce prior research [6] indicating that hybrid architectures incorporating both convolutional and recurrent layers tend to perform best for H3 and H4 datasets, where CNN models perform best for the DNA Sequence Dataset classification. The comparative analysis presented here provides a foundation for future studies to select the most effective combination of data representation techniques and neural network architectures for DNA classification tasks, allowing researchers to leverage the appropriate encoding strategies. In addition, by identifying which encoding methods yield higher classification accuracy with deep learning models, better classifiers for disease-associated DNA sequences can be built, optimizing bioinformatics tools used in clinical genomics, especially in identifying genetic mutations or regulatory regions linked to specific health outcomes. Furthermore, large-scale genomics efforts (which are becoming

standard) can use encoding-efficient models that reduce computational cost while maintaining high performance to interpret patient-specific genomic data, improving drug response predictions and disease risk assessments. Finally, our comparative framework and findings can be used as a benchmark in academic settings for evaluating new models, as well as a foundation for future studies in bioinformatics.

Many practical, real-world applications can benefit from the results presented in this research. Improvements in the classification of promoters and histone occupancy types contribute to a better understanding of how genes are regulated and expressed. This is particularly important for studying the activity of genes associated with diseases, whether in humans or plants. Such insights can help researchers more effectively manage and treat these diseases. Additionally, with the availability of complete genomes for many species, promoter classification models can accelerate the genome annotation process.

Despite the valuable insights provided by this study, there are several limitations. First, our study primarily focused on pre-existing deep learning models rather than designing novel architectures optimized for DNA classification, which could yield further performance improvements. Second, while we explored different data representations, hybrid approaches that integrate multiple encoding techniques were not extensively investigated and may offer further enhancements. Third, image-based representations were tested using standard deep learning architectures without specific optimizations for biological sequence data, which may have contributed to their lower performance. Finally, although the problem formulations for the classification of promoter and histone occupancy types are similar, the results from trained models are not necessarily generalizable to other types of biological sequences. This is due to various factors, including the unique characteristics of the biological phenomena associated with each sequence type—phenomena that remain poorly understood by researchers. The generalizability of bioinformatics models remains an open research challenge. To improve classification performance, accumulated biological knowledge over time must be more effectively incorporated, enabling models to leverage not only sequence data but also other data modalities, such as structural and image data.

6 Conclusions

In this study, we evaluated the impact of various numerical and image-based data representation techniques for encoding DNA sequences to understand their effect on the performance of deep learning models. Our results demonstrate that data representation significantly influences model performance, with sequence-based encoding techniques, particularly k-mer one-hot vector encoding, showing the highest overall classification accuracy. CNN-LSTM models performed exceptionally well with text-based k-mer encodings, whereas CNN-based architectures excelled in DNA Sequence Dataset classification, especially for the Yeast, Human and A. Thaliana datasets. Label encoding was found to degrade the performance of hybrid CNN-LSTM models, suggesting that this encoding is less suited for tasks requiring dependency learning, particularly when no ordinal relationship is present. Exploring alternative methods could enhance model performance and generalization, and thus represents an important avenue for future research. Conversely, k-mer sentence encoding and k-mer one-hot vector encoding demonstrated marginal performance variations across models, with the one-hot encoding emerging as the most robust representation.

On the other hand, visual data representations underperformed compared to numerical techniques in this study. This may be attributed to the use of off-the-shelf models with transfer learning, which may not be well-suited to the task of DNA sequence classification. Further research is needed to investigate custom-built neural networks tailored for this purpose. For future work, we aim to explore additional data representation techniques, such as Hilbert Curve representation, and investigate the use of similarity-based models, such as Siamese neural networks, to enhance classification accuracy. Additionally, we plan to focus

on developing custom models optimized for visual data representations to better capture the unique patterns within DNA sequences.

Acknowledgement: The authors would like to thank the reviewers for their valuable insights and constructive comments.

Funding Statement: This research is funded by the Researchers Supporting Project number (RSPD2025R857), King Saud University, Riyadh, Saudi Arabia.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Isra Al-Turaiki and Najwa Altwaijry; methodology, Najwa Altwaijry and Isra Al-Turaiki; implementation, Ahmad Raza Khan and Hamza Ali Rizvi; validation, Sarab AlMuhaideb and Ahmad Raza Khan; formal analysis, Isra Al-Turaiki, Najwa Altwaijry and Sarab AlMuhaideb; writing—original draft preparation, Isra Al-Turaiki, Najwa Altwaijry and Hamza Ali Rizvi; writing—review and editing, Sarab AlMuhaideb; visualization, Sarab AlMuhaideb, Hamza Ali Rizvi and Ahmad Raza Khan; funding, Najwa Altwaijry. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data openly available in a public repository: The yeast, *A. thaliana* and human genomes used in the study are openly available at https://drive.google.com/drive/folders/1VQ4r2SHGMmFMAq52oDhqYWz7cpTboeHE?usp= sharing (accessed on 8 July 2025). The microarray data used are available at ArrayExpress http://www.ebi.ac.uk/arrayexpress (accessed on 8 July 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Appendix A

Table A1 reports a performance comparison using label encoding and one-hot vector encoding on DNA Sequence Dataset. Table A2 compares the performance of ResNet50 and Inception V3 using Visual data representations on DNA Sequence Dataset.

Table A1: Performance comparison using label encoding and one-hot vector encoding on DNA Sequence Dataset

Model	Yeast					A. Thaliana					Human				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
Label Encoding															
CNN-1	0.8580	0.8320	0.8840	0.8542	182.34	0.8775	0.8450	0.9100	0.8734	656.97	0.8890	0.8600	0.9180	0.8856	**667.89**
CNN-2	**0.8645**	**0.839**	**0.89**	**0.8609**	194.66	**0.881**	**0.85**	**0.912**	**0.8771**	693.43	**0.8964**	**0.87**	**0.9229**	**0.9229**	711.24
CNN-LSTM	0.7940	0.7620	0.8260	0.7871	272.37	0.7584	0.7229	0.7940	0.7495	837.84	0.7699	0.7349	0.8049	0.7616	856.32
CNN-BiLSTM	0.8065	0.7810	0.8320	0.8014	291.59	0.7674	0.7449	0.7900	0.7621	881.26	0.7810	0.7520	0.8100	0.7744	928.47
FCNN	0.8161	0.7813	0.8510	0.8095	94.32	**0.7430**	0.7032	0.7829	0.7323	325.67	0.7624	0.7218	0.8030	0.7523	341.52
One-hot Vector Encoding															
CNN-1	0.9350	0.9600	0.9100	0.9365	243.63	**0.9898**	**0.9898**	**0.99**	**0.9898**	788.15	0.9465	0.9210	0.9720	0.9451	817.55
CNN-2	**0.955**	**0.98**	**0.93**	**0.956**	275.18	0.9505	0.9230	0.9780	0.9491	853.64	**0.9949**	**0.9949**	**0.9949**	**0.9949**	868.27
CNN-LSTM	0.8300	0.7800	0.8800	0.8210	330.46	0.6580	0.5840	0.7320	0.6306	1036.47	0.7435	0.7730	0.7140	0.7508	1088.44
CNN-BiLSTM	0.8236	0.7300	0.9173	0.8054	351.13	0.8785	0.8430	0.9140	0.8740	1112.61	0.7925	0.7320	0.8530	0.7791	1186.97
FCNN	0.9366	0.9213	0.9520	0.9356	**144.55**	0.9764	0.9716	0.9812	0.9762	**478.96**	0.9572	0.9328	0.9817	0.9562	**483.52**

Note: The bold typeface in the table denotes the best score.

Table A2: Performance comparison using Visual data representations on DNA Sequence Dataset

Model	Yeast					A. Thaliana					Human				
	Acc	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**	***Acc***	***Sen***	***Spe***	***F1***	**Time (s)**
ColorSquare															
ResNet50	**0.8390**	**0.804**	**0.8740**	**0.8331**	**447.29**	**0.8297**	**0.7753**	**0.8841**	**0.8199**	**1826.34**	**0.8180**	**0.7813**	**0.8547**	**0.8110**	**1867.12**
Inception V3	0.7890	0.7433	0.8346	0.7789	512.42	0.7783	0.7336	0.8230	0.7679	2085.47	0.766	0.7214	0.8106	0.7551	2018.09
FCGR															
ResNet50	**0.8770**	**0.8626**	**0.8913**	**0.8752**	**404.36**	**0.8792**	**0.8548**	**0.9036**	**0.8761**	**1686.17**	**0.8769**	**0.8425**	**0.9113**	**0.8725**	**1708.34**
Inception V3	0.8441	0.8133	0.8750	0.8392	457.86	0.8570	0.8216	0.8924	0.8517	1842.41	0.8425	0.8032	0.8818	0.8360	1884.68

Note: The bold typeface in the table denotes the best score.

References

1. Sayers EW, Cavanaugh M, Frisse L, Pruitt KD, Schneider VA, Underwood BA, et al. GenBank 2025 update. Nucleic Acids Res. 2025;53(D1):D56–61. doi:10.1093/nar/gkae1114.
2. Ghosh P, Fagnan K, Connor R, Pannu R, Wheeler TJ, Pop M, et al. Contributions of the Petabyte Scale Sequence Search Codeathon toward efforts to scale sequence-based searches on SRA. arXiv:2505.06395. 2025.
3. Xing Z, Pei J, Keogh E. A brief survey on sequence classification. ACM SIGKDD Explor Newsl. 2010;12(1):40–8. doi:10.1145/1882471.1882478.
4. Wang JT, Rozen S, Shapiro BA, Shasha D, Wang Z, Yin M. New techniques for DNA sequence classification. J Comput Biol J Comput Mol Cell Biol. 1999;6(2):209–18. doi:10.1089/cmb.1999.6.209.
5. Yang A, Zhang W, Wang J, Yang K, Han Y, Zhang L. Review on the application of machine learning algorithms in the sequence data mining of DNA. Front Bioeng Biotechnol. 2020;8:1032. doi:10.3389/fbioe.2020.01032.
6. Gunasekaran H, Ramalakshmi K, Rex Macedo Arokiaraj A, Deepa Kanmani S, Venkatesan C, Suresh Gnana Dhas C. Analysis of DNA sequence classification using CNN and hybrid models. Comput Math Methods Med. 2021;2021:e1835056. doi:10.1155/2021/1835056.
7. Nguyen NG, Tran VA, Ngo DL, Phan D, Lumbanraja FR, Faisal MR, et al. DNA sequence classification by convolutional neural network. J Biomed Sci Eng. 2016;9(5):280–6. doi:10.4236/jbise.2016.95021.
8. Rizzo R, Fiannaca A, La Rosa M, Urso A. Classification experiments of DNA sequences by using a deep neural network and chaos game representation. In: Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, CompSysTech '16; 2016 Jun 23–24; New York, NY, USA. p. 222–8. doi:10.1145/2983468.2983489.
9. Choong ACH, Lee NK. Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method. In: Proceedings of the 2017 International Conference on Computer and Drone Applications (IConDA); 2017 Nov 9–11; Kuching, Malaysia. p. 60–5.
10. Kautsar A. DNA sequence classification using machine learning models based on k-mer features. J Comput Digital Busin. 2025;4(2):100–5. doi:10.56427/jcbd.v4i2.762.
11. Vishal CH, Krishna S, Reddy JA, Likith N, Chiranjeevi N. Classification of DNA using machine learning. Int J Novel Res Develop (IJNRD). 2024 Apr;9(4):c66–70.
12. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 27–30; Las Vegas, NV, USA. p. 770–8.
13. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 27–30; Las Vegas, NV, USA. p. 2818–26.
14. Pokholok DK, Harbison CT, Levine S, Cole M, Hannett NM, Lee TI, et al. Genome-wide map of nucleosome acetylation and methylation in yeast. Cell. 2005;122(4):517–27. doi:10.1016/j.cell.2005.06.026.
15. Asim MN, Ibrahim MA, Zaib A, Dengel A. DNA sequence analysis landscape: a comprehensive review of DNA sequence analysis task types, databases, datasets, word embedding methods, and language models. Front Med. 2025;12:1503229. doi:10.3389/fmed.2025.1503229.
16. Sastri AP, Akbar S, Suresh K. Assessment of machine learning algorithms in DNA sequence data mining. In: Computational techniques for biological sequence analysis. Boca Raton, FL, USA: CRC Press; 2025. p. 61–77.
17. Deshpande M, Karypis G. Evaluation of techniques for classifying biological sequences. In: Proceedings of the Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013; 2013 Apr 14–17; Gold Coast, Australia.
18. She R, Chen F, Wang K, Ester M, Gardy JL, Brinkman FSL. Frequent-subsequence-based prediction of outer membrane proteins. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03; 2023 Aug 24–27; Washington, DC, USA. p. 436–45.
19. RuleQuest-Research. Data Mining Tools See5 and C5.0. [cited 2022 Dec 2]. Available from: https://www.rulequest.com/see5-info.html.

20. Singh OP, Vallejo M, El-Badawy IM, Aysha A, Madhanagopal J, Faudzi AAM. Classification of SARS-CoV-2 and non-SARS-CoV-2 using machine learning algorithms. Comput Biol Med. 2021;136:104650. doi:10.1016/j.compbiomed.2021.104650.
21. Ao C, Jiao S, Wang Y, Yu L, Zou Q. Biological sequence classification: a review on data and general methods. Research. 2022;2022(5):0011. doi:10.34133/research.0011.
22. Lazebnik T, Simon-Keren L. Cancer-inspired genomics mapper model for the generation of synthetic DNA sequences with desired genomics signatures. Comput Biol Med. 2023;164(6529):107221. doi:10.1016/j.compbiomed.2023.107221.
23. Amiri Z, Heidari A, Navimipour NJ, Esmaeilpour M, Yazdani Y. The deep learning applications in IoT-based bio- and medical informatics: a systematic literature review. Neural Comput Applicat. 2024;36(11):5757–97. doi:10.1007/s00521-023-09366-3.
24. Lefin N, Herrera-Belén L, Farias JG, Beltrán JF. Review and perspective on bioinformatics tools using machine learning and deep learning for predicting antiviral peptides. Molec Diver. 2024;28(4):2365–74. doi:10.1007/s11030-023-10718-3.
25. Kumar N, Srivastava R. Deep learning in structural bioinformatics: current applications and future perspectives. Brief Bioinform. 2024;25(3):bbae0424. doi:10.1093/bib/bbae042.
26. Trabelsi A, Chaabane M, Ben-Hur A. Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. Bioinformatics. 2019;35(14):i269–77. doi:10.1093/bioinformatics/btz339.
27. Rehman MU, Tayara H, Chong KT. DCNN-4mC: densely connected neural network based N4-methylcytosine site prediction in multiple species. Computat Struct Biotechnol J. 2021;19(1748):6009–19. doi:10.1016/j.csbj.2021.10.034.
28. Qayyum A, Benzinou A, Saidani O, Alhayan F, Khan MA, Masood A, et al. Assessment and classification of COVID-19 DNA sequence using pairwise features concatenation from multi-transformer and deep features with machine learning models. SLAS Technol. 2024;29(4):100147. doi:10.1016/j.slast.2024.100147.
29. Bhandari N, Khare S, Walambe R, Kotecha K. Comparison of machine learning and deep learning techniques in promoter prediction across diverse species. PeerJ Comput Sci. 2021;7:e365. doi:10.7717/peerj-cs.365.
30. Yu T, Cheng L, Khalitov R, Yang Z. A sparse and wide neural network model for DNA sequences. Neural Netw. 2025;184:107040. doi:10.1016/j.neunet.2024.107040.
31. Wang X, Lian Q, Qu P, Yang Q. TBCA: prediction of transcription factor binding sites using a deep neural network with lightweight attention mechanism. IEEE J Biomed Health Inform. 2024;28(4):2397–407. doi:10.1109/JBHI.2024.3355758.
32. Tariq S, Amin A. DeepCTF: transcription factor binding specificity prediction using DNA sequence plus shape in an attention-based deep learning model. Signal Image Video Process. 2024;16(6–7):5239–51. doi:10.1007/s11760-024-03229-7.
33. Amato D, Calderaro S, Bosco GL, Vella F, Rizzo R. Proteins transcription factor prediction using graph neural networks. In: Cerulo L, Napolitano F, Bardozzo F, Cheng L, Occhipinti A, Pagnotta SM, editors. Computational intelligence methods for bioinformatics and biostatistics. CIBB 2024. Lecture notes in computer science. Vol. 15276. Cham: Springer; 2024. doi:10.1007/978-3-031-89704-7_2.
34. Mock F, Kretschmer F, Kriese A, Böcker S, Marz M. Taxonomic classification of DNA sequences beyond sequence similarity using deep neural networks. Proc Natl Acad Sci U S A. 2022;119(1):e2122636119. doi:10.1101/2021.07.09.451778.
35. Pipoli V, Cappelli M, Palladini A, Peluso C, Lovino M, Ficarra E. Predicting gene expression levels from DNA sequences and post-transcriptional information with transformers. Comput Methods Programs Biomed. 2022;225(8):107035. doi:10.1016/j.cmpb.2022.107035.
36. Mikolov T. Efficient estimation of word representations in vector space. arXiv:1301.3781. 2023.
37. Zeng M, Wu Y, Lu C, Zhang F, Wu FX, Li M. DeepLncLoc: a deep learning framework for long non-coding RNA subcellular localization prediction based on subsequence embedding. Brief Bioinform. 2022;23:bbab360. doi:10.1101/2021.03.13.435245.

38. Dong G, Wu Y, Huang L, Li F, Zhou F. TExCNN: leveraging pre-trained models to predict gene expression from genomic sequences. Genes. 2024;15(12):1593. doi:10.3390/genes15121593.
39. Ji Y, Zhou Z, Liu H, Davuluri RV. DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. Bioinformatics. 2021;37:2112–20. doi:10.1101/2020.09.17.301879.
40. Zhou Z, Ji Y, Li W, Dutta P, Davuluri R, Liu H. Dnabert-2: efficient foundation model and benchmark for multi-species genome. arXiv:2306.15006. 2023.
41. Kim H. Deep learning. In: Kim H, editor. Artificial intelligence for 6G. Cham, Switzerland: Springer International Publishing; 2022. p. 247–303. doi:10.1007/978-3-030-95041-5_6.
42. Lipman DJ, Pearson WR. Rapid and sensitive protein similarity searches. Science. 1985;227(4693):1435–41. doi:10.1126/science.2983426.
43. Zhang Z, Song T, Zeng X, Niu Y, Jiang Y, Pan L, et al. Colorsquare: a colorful square visualization of DNA sequences. Match-Commun Math Comput Chem. 2012;68:621.
44. Almeida JS, Carriço JA, Maretzek A, Noble PA, Fletcher M. Analysis of genomic sequences by Chaos Game Representation. Bioinformatics. 2001;17(5):429–37. doi:10.1093/bioinformatics/17.5.429.
45. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Inform Process Manag. 2009;45(4):427–37. doi:10.1016/j.ipm.2009.03.002.
46. Nemenyi PB. Distribution-free multiple comparisons. Ann Arbor, MI, USA: Princeton University; 1963.
47. Soliman NF, Abd-Alhalem SM, El-Shafai W, Abdulrahman SESE, Ismaiel N, El-Rabaie ESM, et al. An improved convolutional neural network model for DNA classification. Comput Mater Contin. 2022;70(3):5907–27. doi:10.32604/cmc.2022.018860.