Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 5, No. 3, May 2024, Page 709-718
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v5i3.5170

Comparative Analysis of Deep Learning Architectures for DNA Sequence Classification: Performance Evaluation and Model Insights

Gregorius Airlangga

Engineering Faculty, Information System Study Program, Atma Jaya Catholic University of Indonesia, Jakarta, Indonesia
Correspondence Author Email: gregorius.airlangga@atmajaya.ac.id
Submitted: 11/05/2024; Accepted: 31/05/2024; Published: 31/05/2024

Abstract−The classification of DNA sequences using deep learning models offers promising avenues for advancements in genomics and personalized medicine. This study provides a comprehensive evaluation of several deep learning architectures, including Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), Gated Recurrent Units (GRUs), Bidirectional LSTMs (BiLSTMs), and hybrid models combining CNNs with various recurrent networks, to classify human DNA sequences into functional categories. We employed a dataset of approximately 100,000 labeled sequences, ensuring a balanced representation across seven distinct classes to facilitate a fair comparison of model performance. Each model was assessed based on accuracy, precision, recall, F1 score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). The CNN model demonstrated superior accuracy (74.86%) and the highest AUC (94.64%), indicating its effectiveness in capturing spatial patterns within sequences. LSTM and GRU models showed commendable performance, particularly in balancing precision and recall, suggesting their capability in managing sequential dependencies. However, hybrid models did not perform as expected, showing lower overall metrics, which highlighted challenges in model integration and complexity management. The findings suggest that while CNNs excel in feature extraction, sequence-based models like LSTMs and GRUs provide valuable capabilities in capturing long-range dependencies, essential for comprehensive genomic analysis. The study underscores the need for optimized hybrid models and further research into model robustness and generalizability.

Keywords: Deep Learning; DNA Classification; Convolutional Neural Networks; Recurrent Neural Networks; Genomic Data Analysis

1. INTRODUCTION

The exploration of the human genome has continually reshaped our understanding of biology and medicine, offering unparalleled insights into the genetic bases of diseases and individual traits [1]–[3]. As genomic technologies have advanced, notably through the introduction of next-generation sequencing, the scientific community now has access to data of unprecedented volume and complexity [4]. This revolution has not only opened new avenues of research but has also introduced significant computational challenges [5]. The effective interpretation and classification of vast DNA sequences, crucial for advancements in genetic research and clinical applications, have thus become paramount tasks within the field of bioinformatics [6]. Historically, the analysis of genomic data has been dominated by traditional bioinformatics techniques that include sequence alignment, gene prediction, and the identification of regulatory elements [7]. These methods have been instrumental in early genomic discoveries and have provided a basic framework for understanding genetic structures [8]. However, they often require extensive manual effort to extract meaningful information and are typically constrained by the computational resources needed to process large datasets [9]. As the scale of genomic data expands, these traditional approaches are increasingly inadequate, limited by their scalability and sensitivity in detecting complex genetic variations [10].

In response to these limitations, there has been a significant shift toward employing machine learning techniques, which offer more sophisticated methods for extracting patterns and predicting phenotypic outcomes from genomic data [11]. Among these, deep learning has emerged as particularly transformative, driven by its success in other data-intensive domains like image recognition and natural language processing [12]. Deep learning models, especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), including Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), have been adapted to capture the spatial and sequential patterns within DNA sequences, outperforming traditional models in accuracy and efficiency [13]–[15]. Despite these advancements, the integration of deep learning into genomic research faces significant challenges [16]. The primary issue is the interpretability of these models. The ability to understand and explain the decisions made by deep learning algorithms is crucial, particularly in genomics, where the biological relevance of predictions needs to be clear for them to be clinically and scientifically useful [17]. Additionally, there remains a substantial gap in the ability to integrate multimodal genomic data: combining genomic, transcriptomic, and proteomic data, in order to form comprehensive models that can navigate the complexities of human biology [18]–[20].

This research aims to tackle these pressing issues by conducting a thorough evaluation and comparison of several advanced deep learning architectures, specifically CNN, LSTM, GRU, and their hybrid forms. Our study focuses on their application in classifying human DNA sequences into predefined categories, assessing not only their performance but also their practical utility in real-world genomic tasks. The goal is to identify which

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 5, No. 3, May 2024, Page 709-718
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v5i3.5170

models most effectively balance accuracy with interpretability and can handle the diversity and complexity of genomic data. This study makes several pivotal contributions to the fields of computational biology and genomics. It benchmarks the effectiveness of various deep learning models against traditional genomic classification methods, providing a clear comparison in terms of performance metrics such as accuracy, precision, and recall. Importantly, it also advances efforts to enhance the transparency and interpretability of these models, proposing new approaches that could make deep learning more accessible and understandable to genomic researchers and practitioners. Following this introduction, the article meticulously discusses details the experimental design, data preprocessing strategies, and the specific architectures of the deep learning models employed. The subsequent sections present a comprehensive analysis of the results, evaluating the models' performance across various datasets and conditions. The discussion integrates these findings with broader genomic research themes, considering their implications for future studies and clinical applications. Finally, the conclusion summarizes the study's key contributions, acknowledges its limitations, and outlines prospective avenues for further research.

2. RESEARCH METHODOLOGY

This research methodology employs a systematic approach to explore and validate the capabilities of various deep learning architectures in the classification of complex DNA sequences. By combining meticulous data preparation, innovative model architecture exploration, and rigorous training and validation processes as presented in the figure 1, the study aims to advance genomic research and provide insights into the genetic sequences' underlying biological processes.

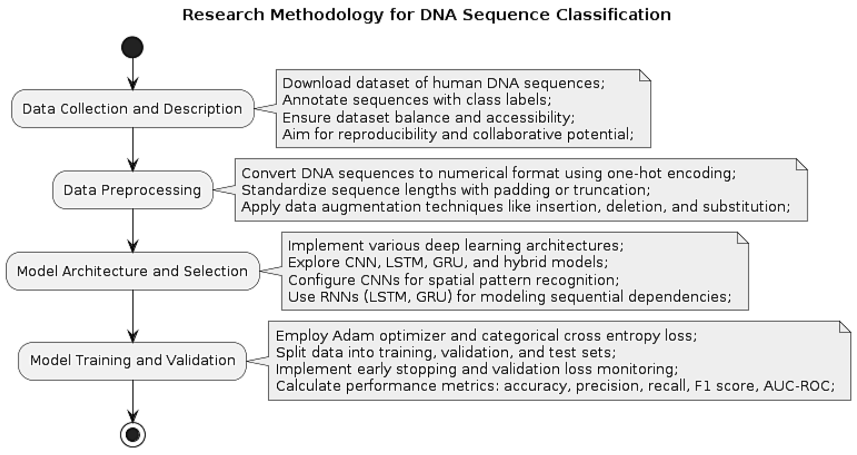
Research Methodology for DNA Sequence Classification
Data Collection and Description
Download dataset of human DNA sequences;
Annotate sequences with class labels;
Ensure dataset balance and accessibility;
Aim for reproducibility and collaborative potential;
Data Preprocessing
Convert DNA sequences to numerical format using one-hot encoding;
Standardize sequence lengths with padding or truncation;
Apply data augmentation techniques like insertion, deletion, and substitution;
Model Architecture and Selection
Implement various deep learning architectures;
Explore CNN, LSTM, GRU, and hybrid models;
Configure CNNs for spatial pattern recognition;
Use RNNs (LSTM, GRU) for modeling sequential dependencies;
Model Training and Validation
Employ Adam optimizer and categorical cross entropy loss;
Split data into training, validation, and test sets;
Implement early stopping and validation loss monitoring;
Calculate performance metrics: accuracy, precision, recall, F1 score, AUC-ROC;

Figure 1. Research Methodology Activity Diagram

2.1 Data Collection and Description

In this study, we utilized a curated dataset of human DNA sequences that play a critical role in advancing our understanding of genetic function and structure, and the dataset can be downloaded from [21]. This dataset is characterized by a rich variety of categories, each corresponding to different gene functions or structural characteristics, highlighting the complex nature of human genetic makeup. The selection of these sequences was guided by the objective to cover a broad spectrum of genetic functions, thereby providing a comprehensive base for the analysis. The data for this research was meticulously sourced from a publicly available genomic database. The choice of a public database was strategic, aimed at ensuring that our study's findings are reproducible, and that the dataset can be readily accessed by other researchers in the field. This openness is pivotal in fostering a collaborative environment where methodologies and discoveries can be easily validated and built upon by the scientific community.

Each sequence within our dataset comes annotated with a specific label, identifying it as belonging to one of seven distinct classes. These classes are designed to represent the diverse functions and characteristics of the sequences, thus enabling a structured approach to the classification task. Such labeling is crucial as it directly supports the supervised learning techniques employed in our analysis, providing a clear target for the predictive modeling. The dataset encompasses approximately 100,000 sequences, which have been carefully balanced across the different classes. This balance is crucial to ensure that the predictive models developed in the study are not biased toward the more frequently represented classes. Maintaining this equilibrium allows for a more accurate and generalizable understanding of how the models perform, making the outcomes of the research applicable to a wider range of genetic data. The structured and balanced nature of this dataset not only facilitates a more effective training process but also enhances the validity of the comparison across different machine learning models. This comprehensive approach to data collection and categorization establishes a robust foundation for exploring the capabilities of deep learning technologies in genetic sequence analysis, setting the stage for insightful discoveries and methodological advancements in the field of genomics.

2.2 Data Preprocessing

The data preprocessing stage is critical in preparing the raw DNA sequences for effective analysis using deep learning models. The DNA sequences, inherently composed of the nucleotides represented by the characters A, T, C, and G, require conversion into a numerical format that deep learning architectures can process. This transformation was achieved through the method of one-hot encoding. In this technique, each nucleotide is converted into a binary vector that uniquely represents it across four dimensions. For instance, adenine (A) is encoded as [1,0,0,0], thymine (T) as [0,1,0,0], cytosine (C) as [0,0,1,0], and guanine (G) as [0,0,0,1]. This encoding not only simplifies the representation of genetic information but also prevents any ordinal relationships between the bases, which could potentially bias the learning process. Further, given the intrinsic variability in sequence length within the dataset, it was necessary to standardize the lengths of the sequences to create uniformity in the input size for the models as presented in the equation (1). To achieve this, all sequences were adjusted to a fixed length of 1000 nucleotides. Sequences that were shorter than this threshold were padded with zeros at their ends, ensuring that they met the required length without altering their original genetic information. Conversely, sequences exceeding 1000 nucleotides were truncated, a compromise to maintain consistency across the dataset while still capturing the most significant portion of the genetic data.

$$g(s) = s \oplus [0,0,\ldots,0] \,\&\, "\{if\}\ |s| < L\ s[1{:}L] \,\&\, "\{if\}\ |s| > L \tag{1}$$

Where ($\oplus$) denotes concatenation and ($|s|$) represents the length of the sequence (s). Additionally, to bolster the robustness of the predictive models and to mitigate the risk of overfitting, we employed various data augmentation techniques. These techniques are particularly pivotal in genomic data analysis due to the frequent occurrence of small mutations in real-world scenarios. Our augmentation strategies included the random insertion, deletion, and substitution of nucleotides within the DNA sequences as presented in the equation (2) – (4) respectively. By introducing these modifications, the models were trained not only on the original data but also on potential variations of it, enhancing their ability to generalize from the training data to new, unseen genetic sequences. This approach is crucial for developing models that are resilient and can accurately interpret genetic data even when it deviates slightly from the examples seen during training. Together, these preprocessing steps: sequence encoding, padding, and augmentation, form a comprehensive framework that ensures the DNA sequences are optimally prepared for the subsequent stages of model training and validation. This preparation is essential for leveraging the full capabilities of deep learning in decoding the complex patterns embedded within genetic data, thereby paving the way for more accurate and reliable genetic sequence classification.

$$I(s,p,x) = s[1{:}p] \oplus [x] \oplus s[p+1{:}|s|] \tag{2}$$

$$D(s,p) = s[1{:}p-1] \oplus s[p+1{:}|s|] \tag{3}$$

$$S(s,p,x) = s[1{:}p-1] \oplus [x] \oplus s[p+1{:}|s|] \tag{4}$$

2.3 Model Architecture and Selection

In this study, we investigated the capabilities of various deep learning architectures to ascertain their suitability and efficiency in classifying complex DNA sequences. The architectures were carefully selected to explore different dimensions of sequence analysis, ranging from the identification of spatial patterns to the modeling of sequential dependencies. The first architecture we explored was the Convolutional Neural Network (CNN), renowned for its prowess in pattern recognition within two-dimensional data. We adapted the CNN for one-dimensional sequence data, configuring it to detect and interpret the spatial hierarchies intrinsic to DNA sequences. The architecture of the CNN included multiple convolutional layers that are designed to capture patterns and motifs indicative of the sequence's functional roles. These layers were interspersed with max pooling layers that help reduce the dimensionality of the data, thus simplifying the information without losing critical features. Following these convolutional and pooling layers, a flattening layer was introduced to transform the pooled features into a flat array for subsequent processing. The architecture culminated in several dense layers, which serve to interpret these features and execute the final classification based on the patterns recognized by the convolutional layers.

Next, we implemented the Long Short-Term Memory (LSTM) network, a type of recurrent neural network (RNN) specifically designed to address the challenge of learning long-range dependencies. Recognizing the sequential nature of DNA—where the significance of a nucleotide often depends on its context within the sequence—LSTM models are particularly suited for this analysis. The LSTM layers are adept at preserving information over long sequences, potentially capturing complex genetic regulations that are not immediately adjacent within the sequence. Dense layers followed the LSTM layers to classify the sequences into their respective categories based on the learned features. Additionally, we explored the Gated Recurrent Unit (GRU), another variant of RNN with a similar purpose but optimized to be more efficient than the LSTM. The GRU simplifies the LSTM architecture by using fewer parameters and modifying the gating mechanisms that regulate information flow within the unit. This efficiency does not significantly compromise the model's performance, making GRUs an attractive option for sequence analysis where computational resources or data availability might be limited.

To leverage the strengths of both CNNs and RNNs, we also experimented with hybrid models that combine these architectures, specifically, CNN-LSTM and CNN-GRU models. These hybrid models utilize convolutional layers at the initial stages to process the sequence and identify local patterns or motifs. The features extracted by the CNN layers are then fed into RNN layers, either LSTM or GRU, which integrate these features across the entire sequence. This combination allows the models to benefit from the local feature extraction capabilities of CNNs and the sequence modeling strengths of RNNs, providing a robust framework for understanding both the immediate and contextual implications of genetic sequences. Each of these models was chosen and designed with the aim of unraveling different aspects of the DNA sequences, from immediate spatial relationships to long-term sequential dependencies, offering a comprehensive approach to understanding the complexities of genetic data. Through this diverse selection of model architectures, the study aims to identify the most effective strategies for DNA sequence classification, contributing to advancements in genomic research and applications.

2.3.1 CNN Method

A CNN adapted for one-dimensional sequence data utilizes a convolution operation to extract spatial features from the sequence. This operation is defined mathematically in equation (5).

$$f(x) = \sigma\left(\sum_{i=1}^{k} w_i \cdot x_{n+i-1} + b\right) \quad (5)$$

Where (x) is the input sequence, (w) represents the weights of a convolutional filter of size (k), (b) is the bias term, and (σ) is a non-linear activation function such as the rectified linear unit (ReLU). The filter slides over the sequence, computing a dot product at each position, which captures local dependencies within the sequence. This convolutional process is repeated using multiple filters to capture various aspects of the sequence information, leading to a feature map for each filter. Besides on this equation, max pooling is a down-sampling operation used in CNNs to reduce the dimensionality of feature maps, thus decreasing computational complexity, and enhancing the detection of dominant features. The operation is defined by equation (6).

$$m(x) = \max\left(x_j, x_{j+1}, \dots, x_{j+p-1}\right) \quad (6)$$

Where (x) represents a segment of the output from the convolutional layer and (p) is the pooling size. The max pooling operation processes each segment of the feature map, taking the maximum value within a window of predefined size (p). This method helps in making the representation more abstract and invariant to small shifts and distortions in the sequence, which is beneficial for capturing the most significant features without retaining unnecessary detail.

2.3.2 LSTM Method

Long Short-Term Memory networks (LSTMs) are a specialized kind of Recurrent Neural Network (RNN) that can learn and remember over long sequences of inputs. LSTMs are particularly useful in applications where the learning of long-range temporal dependencies is critical, such as in language modeling, speech recognition, and biological sequence analysis. Unlike standard RNNs, LSTMs include mechanisms called gates that regulate the flow of information. These gates can add or remove information to the cell state, a memory-like feature of the network. This functionality helps the LSTM to avoid the long-term dependency problem typical of traditional RNNs. An LSTM unit is composed of the four components.

Firstly, $Cellstate((C_t))$, this acts as the 'memory' of the LSTM unit and carries relevant information throughout the processing of the sequence. Changes to the cell state are carefully regulated by structures called gates. Secondly, forget gate $((f_t))$, this gate decides what information is to be discarded from the cell state. It looks at the previous output (h_{t-1}) and the current input (x_t), and applies a sigmoid function to produce a range between 0 and 1 for each number in the cell state (C_{t-1}). A number of 1 represents “completely keep this” while a 0 represents “completely get rid of this”. Furthermore, Input gate $((i_t))$, the input gate decides what new information is added to the cell state. It operates in two parts, firstly, a sigmoid layer and a tanh layer. The sigmoid layer decides which values will be updated, and the tanh layer creates a vector of new candidate values, $(\widetilde{C_t})$, that could be added to the state. Lastly, output gate $((o_t))$, the output gate decides what the next hidden state (h_t) should be. The hidden state contains information about previous inputs. The information at the current cell state(C_t) influences the output, which is filtered by the output gate. The tanh of the cell state is multiplied by the output of the sigmoid gate, so that only the parts of the cell state that are decided to be output are actually produced. The LSTM transitions can be mathematically described by the equation (7) – (12).

$$\mathrm{f_t = \sigma(W_f \cdot [h_t - 1, x_t] + b_f)} \quad (7)$$

$$\mathrm{i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)} \quad (8)$$

$$\widetilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{9}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{c_t} \tag{10}$$

$$o_t = \sigma(W\_o \cdot [h\_\{t-1\}, x\_t] + b\_o) \tag{11}$$

$$h_t = o_t * \tanh(C_t) \tag{12}$$

In these equations, (σ) denotes the sigmoid function, which is used in the gate mechanisms to restrict values between 0 and 1, effectively acting as a filter. ($\tanh$) provides a way to regulate the information flow within the unit, normalizing the values between -1 and 1. The element-wise multiplication ($*$) indicates that the operations are conducted on each corresponding element of vectors independently. This intricate coordination of gates and states allows the LSTM to mitigate the gradient vanishing problem typical of traditional RNNs, enabling it to learn from data where inputs are separated by long time intervals.

2.3.3 GRU Method

The Gated Recurrent Unit (GRU) is an adaptation of the more complex LSTM network that simplifies and optimizes the recurrent formula while preserving the capability to learn long dependencies in sequence data. GRUs have been effectively used in various tasks like language modeling, machine translation, and sequence prediction due to their efficiency and simplicity. A GRU modulates the flow of information without separate memory cells and is generally less computationally intensive than LSTMs due to having fewer gates.

A GRU has two gates, firstly, update gate $((z_t))$, this gate decides how much of the past information (from previous time steps) needs to be passed along to the future. It effectively determines how much of the previous hidden state should contribute to the current hidden state, balancing between the old information and new candidates. Secondly, reset gate, $((r_t))$, this gate decides how much of the past information to forget. It can be seen as setting the state to ignore the previous hidden state and completely reset with the new input, allowing the model to drop any information that is no longer relevant. The operations within a GRU are described by the equation (13) – (16).

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{13}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{14}$$

$$\widetilde{h_t} = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b) \tag{15}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t} \tag{16}$$

Each component functions as follows, $((z_t))$, it controls the degree to which the GRU unit updates its activation, or hidden state. The gate is a vector that defines the proportion of the past hidden state that contributes to the candidate state. It applies a sigmoid function (σ) to constrain its output to the range [0,1], acting as a weight between retaining the old state and adopting the new state. Secondly, Reset gate $((r_t))$, it determines how much of the past information to discard, allowing the model to decide if and how to reset the hidden state given the new input. Like the update gate, it employs the sigmoid function to provide outputs between 0 and 1. Thirdly, candidate hidden state $\left((\widetilde{h_t})\right)$, it is formed by blending the previous hidden state and the current input while being modulated by the reset gate. This candidate state proposes how to combine new input information with past memory. Thirdly, final hidden state update $((h_t))$, it is a linear interpolation between the old state and the new candidate state, controlled by the update gate. This step decides the amount of past information to keep and how much of the new candidate state should be used to update the current state. The GRU's architecture allows it to perform similarly to the LSTM with fewer parameters, making it computationally more efficient and easier to modify and maintain. These dynamics make the GRU particularly useful in tasks where model complexity and training time are critical factors.

2.3.4 Hybrid Method

Hybrid models that combine Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) leverage the strengths of both architectures to process sequential data effectively. The CNN layers initially process the input sequence to extract local feature representations, which are then passed on to the RNN layers—either Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), to model temporal dependencies and sequence dynamics. The hybrid architecture typically operates as presented in the equation (17).

$$y_t = \mathrm{RNN}\big(\mathrm{CNN}(x_t)\big) \tag{17}$$

Where (x_t) represents the input sequence at time (t). The CNN component processes this input to capture spatial hierarchies and local contextual features within the data. This operation is crucial for understanding complex patterns in data, such as motifs in DNA sequences or features in audio signals. The CNN layers consist of multiple convolutional and pooling layers. Each convolutional layer applies several filters to the input, capturing various aspects of the input data through operations in the equation (18).

$$f(x) = \sigma\left(\sum_{i=1}^{k} w_i \cdot x_{n+i-1} + b\right) \tag{18}$$

This results in feature maps that highlight important features of the input data, reducing spatial dimensions but retaining critical information. These feature maps are then typically passed through pooling layers, which reduce the dimensionality and computational complexity while maintaining the most significant features. The output from the CNN, which consists of condensed feature representations, is then fed into an RNN. This part of the model captures temporal dependencies as presented in the equation (19).

$$h_t = \mathrm{RNN}(h_{t-1}, c_t) \tag{19}$$

Where (h_t) is the hidden state at time (t), (h_{t-1}) is the hidden state from the previous time step, and (c_t) is the current feature input from the CNN. Depending on the specific architecture, (RNN) can be either an LSTM or a GRU. The LSTM or GRU layers process these inputs to model long-range dependencies, making use of their gating mechanisms to maintain or forget information across longer sequences. The final output (y_t) of the hybrid model is typically obtained after the RNN layer has processed all the temporal information. This output can then be used for various tasks such as sequence classification, prediction, or feature generation. The integration of CNN and RNN layers allows these models to not only recognize complex patterns through convolutional processing but also to understand sequence dynamics, making them extremely powerful for tasks involving sequential data with important spatial and temporal components. This comprehensive approach harnesses the localized feature extraction capabilities of CNNs and the sequential data processing strengths of RNNs, providing a robust framework for understanding both immediate and contextual implications of sequences. The synergy between CNN and RNN components in hybrid models offers enhanced learning capabilities over using either architecture alone, especially in complex applications like genomic sequence analysis, speech recognition, and video processing.

2.4 Model Training and Validation

The process of training and validating the models implemented in this study was meticulously designed to optimize performance and ensure robustness. Each model utilized the Adam optimizer, a choice influenced by its adaptive learning rate capabilities that help in efficiently converging to the optimal set of weights. The optimizer adjusts the learning rate during training, which is particularly beneficial when dealing with complex datasets such as DNA sequences, where the gradient landscape can be highly variable. The loss function selected for this study was categorical cross entropy, which is commonly used in multi-class classification tasks. This choice is apt for our study since it measures the disparity between the predicted probabilities and the actual class output, thereby providing a robust mechanism for minimizing prediction errors across multiple classes. To rigorously validate the models and prevent overfitting: a common issue in machine learning tasks involving complex datasets: the dataset was carefully partitioned into a training set and a test set, with 80% of the data allocated for training and the remaining 20% held out for testing. Further refinement of the training process involved setting aside 10% of the training set to function as a validation set. This validation set played a crucial role during training, enabling continuous monitoring and tuning of the models' parameters. Early stopping was another critical component of our training methodology. This technique involves terminating the training process if the validation loss fails to improve after a certain number of epochs. By implementing early stopping, we mitigated the risk of overfitting, ensuring that the models maintained a generalizable performance rather than merely excelling on the training data.

The evaluation of each model's performance was conducted using several critical metrics, which together provide a rounded assessment of each model's capabilities. Accuracy was the primary metric, indicating the proportion of total predictions that were correct. This metric, while straightforward, offers a quick snapshot of model effectiveness but can sometimes be misleading, particularly in datasets where class distributions are imbalanced. To address this, precision and recall were also used as key metrics. Precision measures the accuracy of the positive predictions made by the model, while recall assesses the model's ability to identify all relevant instances per class. These metrics are particularly vital in scenarios where the cost of false negatives or false positives is high, such as in medical or genomic studies. The F1 score, which is the harmonic mean of precision and recall, was also calculated. This metric is crucial because it provides a balance between precision and recall, offering a single measure that evaluates the model's accuracy while considering both the false positives and false negatives. Additionally, the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) was employed. This metric is invaluable as it evaluates the model's performance across various classification thresholds, rather than at a fixed point, thereby providing insights into the effectiveness of the model under different conditions. Together, these metrics facilitated a comprehensive evaluation of the models, ensuring that the developed models are not only accurate in predicting DNA sequence classes but are also reliable and effective across various levels of decision thresholds. This thorough approach to model validation and performance evaluation underpins the credibility of our findings and supports the potential application of these models in real-world genomic sequence analysis.

The Adam optimizer is used for its adaptive learning rate capabilities, crucial for handling complex data landscapes as presented in equation 20.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}}+\epsilon}\widehat{m_t} \tag{20}$$

Where, θ represents the parameters to be optimized, η is the learning rate, $\widehat{m_t}$ and $\widehat{v_t}$ are bias-corrected first and second moment estimates, respectively. The symbol of ϵ is a small scalar added to the denominator to improve numerical stability. The loss function used is categorical cross entropy, which is defined in equation (21).

$$L = -\sum_{i=1}^{C} y_i \log(p_i) \tag{21}$$

Where, y_i is the actual label (0 or 1) indicating whether class i is the correct classification, p_i is the predicted probability of class i and C is the number of classes. Furthermore, the process of validation loss monitoring and early stopping are implemented as presented in the equation (22).

$$\text{Stop if } L_{val}(e) > L_{val}(e-n) \text{ for } n \text{ consecutive epochs} \tag{22}$$

Where $L_{val}(e)$ is the validation loss at epoch e. Performance is assessed using several metrics such as accuracy, precision, recall, F1 score and AU-ROC as presented from the equation (23) – (27).

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total predictions}} \tag{23}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{24}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{25}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \tag{26}$$

$$\text{AUC-ROC} = \int_0^1 TPR(t)\ dFPR(t) \tag{27}$$

Where, TP is true positives, FP is false positives, FN is false negatives. In addition, $TPR(t)$ is the true positive rate at threshold t, and $FPR(t)$ is the false positive rate.

3. RESULT AND DISCUSSION

3.1 Results

As presented in the table 1, the Convolutional Neural Network (CNN) model demonstrated the highest overall accuracy among the models tested, with a score of 74.86%. It also achieved a notably high Area Under the Curve (AUC) of 94.64%, indicating its strong capability in distinguishing between the different classes of DNA sequences. The precision was particularly high at 89.18%, suggesting that the predictions made by the CNN were highly reliable. However, its recall of 69.34% points to some challenges in identifying all relevant instances of each class, which slightly impacted its F1 score, settling at 78.00%. The Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models displayed similar patterns in performance, with accuracies of 73.95% and 72.63% respectively. The LSTM model had a slightly better balance between precision and recall, which is reflected in its F1 score of 76.46%, compared to the GRU's 73.42%. This suggests that while LSTM managed long-range dependencies in the sequences slightly more effectively than GRU, both struggled with comprehensively identifying all class-relevant sequences, as evidenced by their lower recall rates.

The Bidirectional LSTM (BiLSTM) model, designed to capture sequence information in both forward and reverse orientations, recorded an accuracy almost equivalent to the LSTM model but with a slightly higher AUC of 92.55%. However, its precision and recall rates led to a lower F1 score of 75.46%, indicating some limitations in balancing the detection of relevant instances and the precision of classifications. Hybrid models combining CNN with recurrent architectures—CNN_LSTM, CNN_GRU, and CNN_BiLSTM—showed a decrease in performance compared to their standalone counterparts. This was unexpected as hybrid models are typically anticipated to leverage the strengths of both CNNs (for spatial feature extraction) and RNNs (for sequence dependency capture). The CNN_LSTM model had an accuracy of 59.47%, the lowest among the hybrids, with a corresponding low recall of 46.60%, significantly affecting its F1 score (57.56%). Similarly, the CNN_GRU and CNN_BiLSTM models also underperformed, with accuracies of 55.71% and 61.10% respectively. These results suggest that the integration of these architectures did not synergize as anticipated for this dataset, possibly due to the added complexity overwhelming the models' ability to generalize from the training data.

The results highlight several key insights into the application of deep learning models for DNA sequence classification. The superior performance of the standalone CNN model underscores its effectiveness in capturing essential features from biological sequence data, likely due to its ability to detect complex patterns and motifs crucial for genomic classification. The slight underperformance of LSTM and GRU models relative to CNN might be attributed to the intrinsic challenges associated with processing very long sequences, where important information could be diluted or lost over long distances. Although designed to mitigate such issues, the practical limitations of these models in handling the extensive variability and complexity of genomic data could have restrained their effectiveness. The unexpected lower performance of hybrid models suggests that while theoretically advantageous, the practical implementation and tuning of these models are critical. Overfitting, difficulties in optimizing, or the inefficient merging of feature sets from CNN and RNN layers could contribute to their diminished effectiveness.

3.2 Trade-off Analysis

In this study, the performance trade-offs among various deep learning models used for DNA sequence classification present a complex landscape of strengths and weaknesses. Analyzing these trade-offs is crucial for understanding the suitability and applicability of each model type in real-world genomic research settings. Here, we explore the trade-offs related to accuracy, precision, recall, F1 score, and AUC, based on the results obtained.

The standalone CNN model emerged as the top performer in terms of accuracy and AUC, suggesting its superior ability to handle the spatial complexities inherent in DNA sequences. CNN's design allows it to capture essential motifs and patterns efficiently, a critical feature for genomic sequence analysis. However, the model's simplicity in not accounting for long-range dependencies, which are typical in sequential data, presents a trade-off. While CNNs are excellent at identifying local patterns within the data, they might miss broader contextual information that models like LSTM or GRU can capture. This limitation could be significant depending on the specific requirements of the genomic classification task, such as in scenarios where the functional implications of a sequence depend heavily on distant elements.

The divergence between precision and recall across the models is another key consideration. CNN showed high precision but lower recall, indicating a strong ability to correctly label true positive cases but at the expense of missing other relevant cases. This trade-off is particularly crucial in settings like medical diagnostics, where failing to identify a condition (low recall) can have more severe consequences than incorrectly identifying one (low precision). In such cases, a model that provides a more balanced precision and recall, as seen in the LSTM and GRU, might be preferable, even if it means sacrificing some accuracy. Regarding the robustness of the models, as indicated by the AUC metric, the CNN again showed superior performance, demonstrating its effectiveness in classifying sequences correctly across various decision thresholds. This robustness is beneficial, especially in clinical settings where threshold values can be adjusted to meet specific diagnostic criteria. However, the trade-off with CNN’s structure is its potential oversight of long-range dependencies which are better handled by sequence models like LSTM or GRU.

The performance of the hybrid models, which combine the strengths of CNNs and RNNs, was unexpectedly lower than the standalone models. This outcome suggests a significant trade-off related to the increased complexity and the dynamics of training such models. Hybrid models, while theoretically advantageous due to their integrated approach in capturing both local and long-range features, require careful tuning and potentially more extensive training data to perform optimally. Their complexity can be a major drawback, particularly where computational resources or labeled data are limited. Another important consideration is computational efficiency. For instance, the GRU model, which generally requires fewer parameters than LSTM, offers greater efficiency, which is crucial in resource-constrained environments. However, this efficiency might come at the cost of reduced performance capabilities, as reflected in the slightly lower metrics compared to LSTM. The decision to use a more efficient but potentially less capable model or a more demanding but robust model depends on the specific operational context, such as the availability of computational resources and the need for real-time analysis.

3.3 Constraints and Limitations

The performance outcomes of the various deep learning models applied in our study for DNA sequence classification reveal significant insights while also highlighting inherent constraints and limitations within the models and experimental setup. Understanding these limitations is crucial not only for interpreting the results accurately but also for guiding future research directions. The Convolutional Neural Network (CNN), which exhibited high precision, showed a comparatively lower recall. This indicates that while the model is reliable when it predicts a sequence class correctly, it fails to identify a significant number of relevant sequences. Such a limitation could stem from the CNN’s primary focus on local patterns, potentially overlooking the wider contextual dependencies crucial for identifying some classes. Additionally, despite the high AUC indicating good generalization across different thresholds, there might still be concerns regarding the model's potential to overfit to patterns that are not universally applicable across different genomic contexts.

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, designed to manage sequential data and long-range dependencies, also showed a more balanced approach between precision and recall. However, they too exhibited limitations in capturing all relevant instances, as reflected in their lower recall rates. This could be attributed to the inherent difficulties in managing long-range dependencies effectively, despite these models' design intentions. The complexity of these models and the associated training challenges, such as issues with vanishing or exploding gradients, might also contribute to their slightly lower performance metrics. The Bidirectional LSTM (BiLSTM) was expected to improve upon standard LSTM by capturing information from both forward and reverse directions. However, the improvements in recall and overall accuracy were not as significant as anticipated, which might indicate limitations in how effectively the model integrates bidirectional information. Alternatively, this may reflect an inherent limitation in the data where additional context from the reverse direction is less informative.

All hybrid models, including CNN_LSTM, CNN_GRU, and CNN_BiLSTM, demonstrated lower performance metrics compared to their standalone counterparts. This suggests potential issues with how features extracted by CNN layers are being integrated and utilized by the subsequent recurrent layers. It may also reflect an increase in model complexity that leads to difficulties in training effectively, potentially leading to overfitting where the model complexity with dual architectures does not generalize well on unseen data. The models were all trained and tested on a single dataset, which may limit the generalizability of the findings to other genomic datasets or real-world scenarios. Variations in sequence length, complexity, or class distribution in other datasets might yield different results. Additionally, the one-hot encoding used for DNA sequences provides a basic form of input that might not capture all nuances or biological relevancies of the sequences. Exploring more sophisticated encoding techniques could potentially enhance model performance. Moreover, while the study utilized a comprehensive set of metrics to evaluate model performance, the reliance on aggregate metrics like accuracy and F1 score might mask performance disparities across classes, especially if some classes are underrepresented or inherently more complex to classify.

Table 1. Deep Learning Results

Model	Accuracy	Precision	Recall	F1 Score	AUC
CNN	0.7486	0.8918	0.6934	0.78	0.9464
LSTM	0.7395	0.8667	0.6856	0.7646	0.9235
GRU	0.7263	0.8908	0.6258	0.7342	0.9181
BiLSTM	0.7397	0.8881	0.6575	0.7546	0.9255
CNN_LSTM	0.5947	0.7628	0.466	0.5756	0.8619
CNN_GRU	0.5571	0.7607	0.4025	0.5239	0.8422
CNN_BiLSTM	0.611	0.769	0.5039	0.6042	0.8696

4. CONCLUSION

This study assessed various deep learning models for classifying human DNA sequences, evaluating their performance across multiple metrics. The Convolutional Neural Network (CNN) achieved the highest accuracy and AUC, excelling in capturing spatial features. Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional LSTM (BiLSTM) models performed well, particularly in balancing precision and recall, suitable for sequential dependencies in genomic data. Hybrid models combining CNNs and RNNs underperformed, indicating the need for careful optimization to avoid overfitting and improve generalizability. The research emphasizes selecting the appropriate model based on task requirements: CNNs for high precision and RNNs for long-range dependencies. It also addresses challenges with complex models, such as data demands and tuning difficulties. Identified limitations, including overfitting and training challenges, point to future research areas like improving model robustness, exploring advanced data encoding methods, and increasing training data diversity to enhance deep learning models in genomic research.

REFERENCES

[1] S. M. Bakhtiar and E. Dilshad, *Omics technologies for clinical diagnosis and gene therapy: medical applications in human genetics*. Bentham Science Publishers, 2022.
[2] D. Juan, G. Santpere, J. L. Kelley, O. E. Cornejo, and T. Marques-Bonet, “Current advances in primate genomics: novel approaches for understanding evolution and disease,” *Nat. Rev. Genet.*, vol. 24, no. 5, pp. 314–331, 2023.
[3] S. Mukherjee, *The song of the cell: An exploration of medicine and the new human*. Simon and Schuster, 2022.
[4] M. Ben Khedher, K. Ghedira, J.-M. Rolain, R. Ruimy, and O. Croce, “Application and challenge of 3rd generation sequencing for clinical bacterial studies,” *Int. J. Mol. Sci.*, vol. 23, no. 3, p. 1395, 2022.
[5] R. Chataut, M. Nankya, and R. Akl, “6G Networks and the AI Revolution—Exploring Technologies, Applications, and Emerging Challenges,” *Sensors*, vol. 24, no. 6, p. 1888, 2024.
[6] P. Brlek *et al.*, “Implementing Whole Genome Sequencing (WGS) in Clinical Practice: Advantages, Challenges, and Future Perspectives,” *Cells*, vol. 13, no. 6, p. 504, 2024.
[7] M. A. Rather *et al.*, “Bioinformatics approaches and big data analytics opportunities in improving fisheries and aquaculture,” *Int. J. Biol. Macromol.*, vol. 233, p. 123549, 2023.
[8] W. S. Alharbi and M. Rashid, “A review of deep learning applications in human genomics using next-generation sequencing data,” *Hum. Genomics*, vol. 16, no. 1, p. 26, 2022.
[9] J. Yang, S. C. Han, and J. Poon, “A survey on extraction of causal relations from natural language text,” *Knowl. Inf. Syst.*, vol. 64, no. 5, pp. 1161–1186, 2022.
[10] E. Uffelmann *et al.*, “Genome-wide association studies,” *Nat. Rev. Methods Prim.*, vol. 1, no. 1, p. 59, 2021.
[11] R. Abdollahi-Arpanahi, D. Gianola, and F. Peñagaricano, “Deep learning versus parametric and ensemble methods for genomic prediction of complex phenotypes,” *Genet. Sel. Evol.*, vol. 52, pp. 1–15, 2020.

[12] S. Bianchini, M. Müller, and P. Pelletier, "Deep learning in science," *arXiv Prepr. arXiv2009.01575*, 2020.
[13] A. Kaur, A. P. S. Chauhan, and A. K. Aggarwal, "Prediction of enhancers in DNA sequence data using a hybrid CNN-DLSTM model," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 20, no. 2, pp. 1327–1336, 2022.
[14] A. A. Heydari and S. S. Sindi, "Deep learning in spatial transcriptomics: Learning from the next next-generation sequencing," *Biophys. Rev.*, vol. 4, no. 1, 2023.
[15] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mech. Syst. Signal Process.*, vol. 115, pp. 213–237, 2019.
[16] S. Rauschert, K. Raubenheimer, P. E. Melton, and R. C. Huang, "Machine learning and clinical epigenetics: a review of challenges for diagnosis and classification," *Clin. Epigenetics*, vol. 12, pp. 1–11, 2020.
[17] J. G. Greener, S. M. Kandathil, L. Moffat, and D. T. Jones, "A guide to machine learning for biologists," *Nat. Rev. Mol. cell Biol.*, vol. 23, no. 1, pp. 40–55, 2022.
[18] K. M. Boehm, P. Khosravi, R. Vanguri, J. Gao, and S. P. Shah, "Harnessing multimodal data integration to advance precision oncology," *Nat. Rev. Cancer*, vol. 22, no. 2, pp. 114–126, 2022.
[19] T. Das, G. Andrieux, M. Ahmed, and S. Chakraborty, "Integration of online omics-data resources for cancer research," *Front. Genet.*, vol. 11, p. 578345, 2020.
[20] Y. Wang, S. Tang, R. Ma, I. Zamit, Y. Wei, and Y. Pan, "Multi-modal intermediate integrative methods in neuropsychiatric disorders: A review," *Comput. Struct. Biotechnol. J.*, vol. 20, pp. 6149–6162, 2022.
[21] N. Vasani, "Human DNA Data." 2022.