*International Journal of Basic and Applied Sciences*, 14 (3) (2025) 298-310

**International Journal of Basic and Applied Sciences**

*Website: www.sciencepubco.com/index.php/IJBAS*
*https://doi.org/10.14419/cse17v46*
***Research paper***

# Taxonomic Classification of Bacteria Using Machine Learning Models on DNA Sequences

**Sudhir Anakal [1] *, Sarange Shreepad Marotrao [2], Vedavathi G R [3], Gargi Mishra [4], Anurag Vijay Agrawal [5], Archana Bhaskar [6], Surya D. [7], Anu Swedha Ananthan [8]**

*[1] Department of Master of Computer Applications, Sharnbasva University, Kalaburagi. Karnataka, India*
*[2] Department of Mechanical Engineering, Ajeenkya D Y Patil School of Engineering, Lohgaon, Pune, Maharashtra, India*
*[3] Department of Computer Science & Engineering (AI & ML)‹ East Point College of Engineering and Technology, Bidarahalli, Bengaluru, Karnataka, India*
*[4] Department of Computer Science and Engineering, Bharati Vidyapeeth's College of Engineering, Paschim Vihar, New Delhi, Delhi, India*
*[5] Department of Electronics and Communication Engineering, Bhagwant Institute of Technology, Uttar Pradesh, India*
*[6] School of Computer Science and Applications, Reva University, Bengaluru, Karnataka, India*
*[7] Department of Zoology, Madras Christian College, Tambaram, Chennai, Tamil Nadu, India*
*[8] Department of Microbiology, Justice Basheer Ahmed Sayeed College for Women, Chennai, Tamil Nadu, India*
**Corresponding author E-mail: sudhir.anakal@gmail.com*

*Received: June 10, 2025, Accepted: July 18, 2025, Published: July 25, 2025*

**Abstract**

This study investigates different deep learning architectures, especially 1D and 2D convolutional neural networks (CNNs), for DNA se-sequence classification using k-mer vector representations. The results show that k-mer vectors, especially those with k = 5, can effectively capture relevant features in DNA sequences and achieve high accuracy, precision, and recall across all taxonomic levels. Among the tested models, 1D CNN outperformed 2D CNN in terms of accuracy and training efficiency. However, the 2D CNN achieved a slightly higher accuracy without the nested layers, suggesting that critical information should be discarded due to the lack of input representation. As expected, the model performance declined at lower taxonomic levels due to sequence feature limitations and class imbalance, but still achieved 88% accuracy at the genus level. Notably, simple multi-layer neural networks outperformed CNN, indicating the potential of low-low-complexity models for genomic data classification. These findings suggest that while CNNs are efficient, simpler architectures can provide competitive performance in terms of information representation and task complexity.

***Keywords***: *Machine Learning; DNA Sequences; K-Mer Vector; Deep Learning.*

## 1. Introduction

Every living being on the planet contains genetic material that provides fundamental biological information for understanding and classifying it. This genetic material is made up of ribonucleic acid (DNA), a molecule constructed from chains of nucleotides [1]. Two chains of these nucleotides are called bases, and they form DNA in a helix structure. Thus, the nucleotide sequence is what characterizes the genome of every living being [2].

The bacterial genome is composed of a DNA molecule containing between 500,000 and 10 million base pairs [3]. The importance of classifying bacteria is to distinguish one organism from another and to group similar organisms according to criteria of interest to microbiologists or other scientists [4]. These criteria of interest can be in different areas such as biotechnology, the study of environmental samples, industry, or the diagnosis of diseases caused by bacteria. Another important aspect is the classification of bacteria by their DNA, as it facilitates the task of isolating and culturing them, considering that they are difficult to cultivate under laboratory conditions [5], and a more precise classification at the taxonomic level can even be made.

DNA sequence prediction is a difficult task due to the short read length, the incomplete and fragmented nature of the data [6], and the millions of bacteria that exist in the world. For this reason, machine learning models have been developed to extract features from this type of data [7]. This work seeks to study deep learning models to correctly classify bacteria using DNA sequences.

## 2. Literature review

Hossain et al. (2023) [8] developed a multi-label deep learning framework to enhance the classification capabilities of DNA sequences. Their approach was able to classify each DNA sequence into common taxonomic levels, improving resolution and classification accuracy

compared to traditional alignment-based models. The model exhibited excellent performance metrics, indicating its applicability to complex biological datasets and taxonomic hierarchies. Similarly, Soliman (2022) [9] proposed an improved convolutional neural network (CNN) DNA classification model. By improving the convolutional structure and optimizing parameters, the classification performance of the model was significantly improved, demonstrating the power of deep learning architectures in genomic sequence analysis.

Liang et al. (2020) [10] proposed DeepMicrobes, a deep learning-based metagenomic classification model that uses k-mer embedding and short-term memory (LSTM) networks for classification. This approach provides an alternative to computationally intensive alignment methods and is able to effectively extract taxonomic information from complex microbial communities.

Although Goneim et al. (2020) [11] focused on ovarian cancer rather than gene sequencing, they developed a model that combined CNNs with extreme learning machines (ELMs). The CNN layers acted as feature extractors, while the ELM performed the classification. The architecture showed high performance, indicating that such hybrid models are suitable for DNA classification tasks. However, Abbas et al. (2020) [12] also showed that such hybrid models are suitable for DNA classification tasks. The model successfully identified modification sites with consistent features, further demonstrating the importance of tree-based machine learning methods in genomic research, especially when interpretation and feature selection are critical. Abd-Alhalem (2020) [13] studied the use of CNNs combined with data reduction techniques for bacterial classification. The study showed that reducing the ensemble burden, such as aggregation and principal component analysis (PCA), can improve the computational efficiency of CNNs without affecting classification accuracy.

Finally, Malonzo and Lahdesmak (2023) [14] developed LuxHMM, a probabilistic model using hidden Markov models (HMMs) to analyze DNA methylation data through genome segmentation. This method can more accurately identify different methylated regions and determine the spatial and probabilistic characteristics of methylation patterns. Unlike deep learning methods that typically rely on large amounts of data, LuxHMM can efficiently model biological variation through a common analysis method based on fewer assumptions. Taken together, these studies demonstrate the versatility of machine learning techniques in DNA and RNA classification tasks, and the robustness of probabilistic detection methods (such as deep learning and hybrid models) in dealing with the complexity and massiveness of biological data.

Gunasekaran et al. (2021) [15] further explored the use of CNNs and hybrid models that combined CNNs with traditional machine learning algorithms such as support vector machines and decision trees. The hybrid models achieved higher generalization and accuracy, demonstrating the benefits of combining the feature extraction capabilities of CNNs with the decision-making capabilities of other classifiers.

## 3. Materials and methods

### 3.1. Software

Data processing, training, and evaluation of the deep learning models will be developed in a Python environment, using the TensorFlow and Keras libraries.

### 3.2. Database

The database used1 is part of a research study developed in the article: “Deep learning models for bacterial taxonomic classification of metagenomic data,” published in BMC Bioinformatics. It contains 28,000 short DNA sequences of the 16s gene from a group of bacteria. The sequences were generated using next-generation sequencing (NGS) technology called amplicon (AMP). Furthermore, the group of bacteria belongs to the phylum Proteobacterium and each sequence is labeled by taxa from class to genus. The data processing is available in [5]. Table 1 contains the information about the number of labels for each taxon.

**Table 1:** Information on the Number of Labels for the Different Taxa

| Taxon | Number of tags |
|---|---|
| Class | 3 |
| Order | 20 |
| Family | 37 |
| Genus | 96 |

Figures 1, 2, 3, and 4 show the number of data points for each label for the different taxa, as well as the names of the bacteria that comprise them.

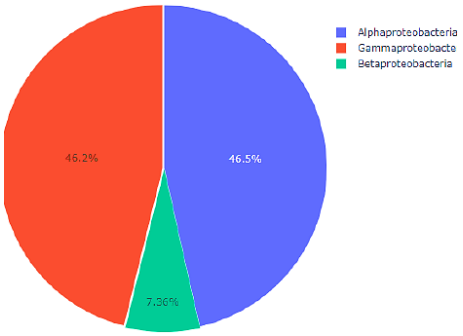


**Fig. 1:** Data Distribution in the Class Taxon.

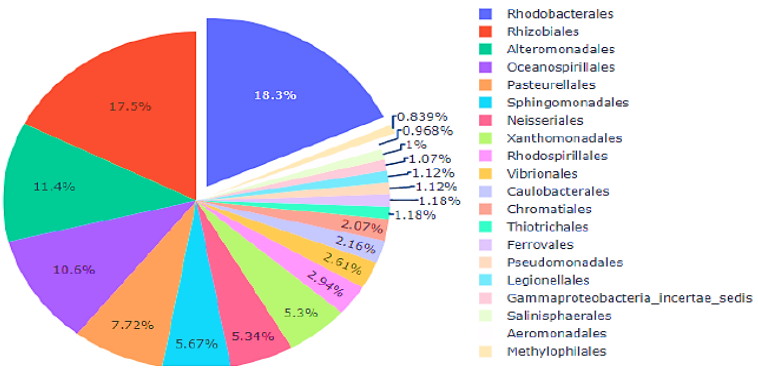


**Fig. 2:** Data Distribution in the Order Taxon.

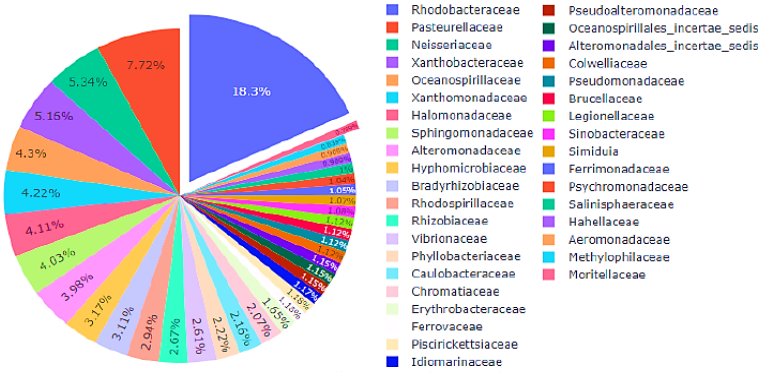


**Fig. 3:** Data Distribution in the Family Taxon.

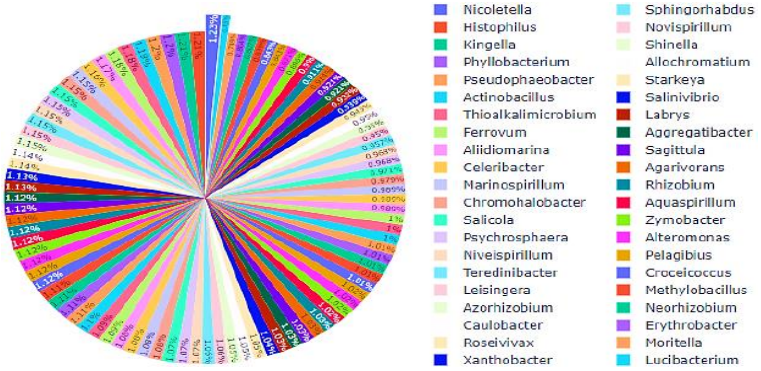


**Fig. 4:** Data Distribution in the Genus Taxon.

From the above, we have an unbalanced database. This can cause poor performance in minority classes, which is why the synthetic minority oversampling (SMOTE) technique was used. This technique selects an instance of the minority class at random and finds its k nearest neighbors; the synthetic instance is created at a randomly selected point between the two instances of the minority class [15].

## 3.3. Processing

### 3.3.1. K-mers

Authors [14-15] use k-mer vectors to obtain a vector representation of DNA sequences. This representation is based on obtaining substrings of a certain length k in the original sequences. This results in a sequence S of n characters determined by the alphabet: λ = A, G, C, T = 4, which represent the DNA nucleotides. The k-mers (K) will be all possible combinations of λ of size k that belong to S. The vector V contains all the substrings of K (Equation 1):

$$V = K_1, K_2, K_3 \ldots K_m \quad (1)$$

Where V will have a size of (Equation 2):

$$\lambda^k = m \quad (2)$$

K substrings are formed using a sliding window of the sequence S, starting at position 1 to position n-k+1. For example, Figure 5 shows the k-mers of size k=4 obtained from the sequence 'AAGTCAAGT':

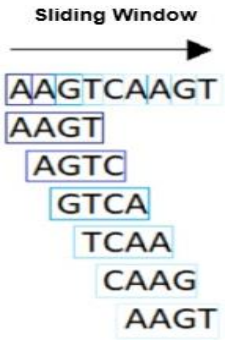


**Fig. 5:** Formation of K Substrings.

When using this type of representation, it is not important to maintain the order of the k-mers in the original sequence, as the goal is to find distinctive k-mers. Also, in the case of k-mers containing characters that do not represent a DNA nucleotide, for example, 'N', they could be eliminated [15].
According to [5], small values for the length of the k-mers can be sufficient to obtain sequence information, while avoiding defining a very large vector space. That is why, for this study, it was decided to use three different values of k: 3, 4 and 5. In this way, it will be possible to analyze which size of k offers the best information about the DNA sequence. The total number of k-mers for the different sizes of k is shown in Table 2.

**Table 2:** Length of the Vector V for the K Sizes Used

| Size of k | Total number of k-mers |
|---|---|
| 3 | 64 |
| 4 | 256 |
| 5 | 1024 |

### 3.3.2. K-mer frequency

Once all the k-mers in the sequence have been obtained, a vector f is created containing the total number of each k-mer in S (Equation 3):

$$\mathbf{f} = \mathbf{c_1, c_2, c_3 \ldots c_m} \tag{3}$$

Finally, the frequency of each k-mer is equal to the total number of k-mers in the sequence S over the total number of k-mers in the sequence [8] (Equation 4):

$$\mathbf{F_i} = \frac{\mathbf{c_i}}{\mathbf{n-k+1}} \tag{4}$$

Returning to the previous example with the sequence 'AAGTCAAGT' of size n=9 and k=4, the frequency of each k-mer is observed in table 3.

**Table 3:** Example of K-Mer Frequency

| K-mer | Total | Frequency |
|---|---|---|
| AAGT | 2 | 1/3 |
| AGTC | 1 | 1/6 |
| GTCA | 1 | 1/6 |
| TCAA | 1 | 1/6 |
| CAAG | 1 | 1/6 |

Furthermore, the length of the vector F is m = λ4 = 256.

### 3.3.3. Standardization

Standardizing a data set involves rescaling the distribution of values so that the mean is zero and the standard deviation is 1. Thus, the data set fits a Gaussian distribution with well-behaved mean and variance. In a deep neural network, rescaling the input data is important because it improves the stability and performance of the network [9].
Standardization requires the mean (μ) and standard deviation (θ) of the input data (Xin). The standardized data (Xout) are obtained from the following equation 5:

$$\mathbf{X_{out}} = \frac{\mathbf{X_{in}-\mu}}{\boldsymbol{\theta}} \tag{5}$$

For the classification of each taxon, there is a different number of labels, as noted above. Therefore, the output vector will depend on the taxon being classified. However, for all taxa, the one-hot vector representation will be used using the Keras function to categorical. For example, Table 4 shows the output vectors for the class taxon.

**Table 4:** Output Vector Representation for the Class Taxon

| Class | Vector one-hot |
|---|---|
| Alphaproteobacteria | 1 0 0 |
| Betaproteobacteria | 0 1 0 |
| Gammaproteobacteria | 0 0 1 |

### 3.4. Training, validation, and testing data

Before starting a deep learning model, it is necessary to divide the dataset into training, validation, and testing data. The training data represents 80% of the total data and, as its name suggests, is used to train all proposed models. The validation data represents 10% of the training data and is used to observe the performance of the models. Finally, the test data represents 20% of the total data and is only used with the chosen model for evaluation. The total data can be seen in Table 5.

**Table 5:** Number of Data Points

| K-mer | Total |
|---|---|
| Training | 20160 |
| Validation | 2240 |
| Test | 5600 |
| Total data | 28000 |

### 3.5. Multilayer neural network

In [5], [6], [7], and [12], deep neural networks are used for DNA sequence classification. Therefore, a multilayer neural network model will initially be tested to evaluate its performance with this type of data.
A two-hidden-layer sequential model was created with the following characteristics:

- Optimizer: Adam.
- Loss function: categorical crossentropy, as it is a multiclass neural network.
- Hidden layer activation: Relu.
- Output layer activation: Softmax. Learning rate: 0.01.
- Metric: Accuracy.

However, to obtain the appropriate number of neurons in each hidden layer, a different model was built for each k-mer length. In addition, different numbers of neurons, ranging from 10 to 150, were tested to obtain the most accurate model. Table 6 shows the number of neurons chosen for each layer.

**Table 6:** Number of Neurons for Each Layer and K-Mer

| Number of neurons in each layer<br>Name | K-mer 3 | K-mer 4 | K-mer 5 |
|---|---|---|---|
| Input layer | 64 | 256 | 1024 |
| Hidden layer 1 | 120 | 120 | 120 |
| Hidden layer 2 | 60 | 84 | 64 |
| Output layer | * | * | * |

(*) The output layer depends on the taxon to be classified. Table 7 shows the number of neurons in the output layer for each taxon.

**Table 7:** Number of Neurons for the Output Layer

| Output layer<br>Taxon | Number of neurons |
|---|---|
| Class | 3 |
| Order | 20 |
| Family | 37 |
| Genus | 96 |

This model will be used with the unbalanced and balanced databases using SMOTE. This will allow us to evaluate whether the synthetic classes created with SMOTE improve model performance.

### 3.6. Convolutional neural network (CNN)

Two types of CNN models are created: 1D and 2D. This will determine which type of network performs best.
The features used in the 1D CNN are:

- The kernel sizes in the convolutional layers are 3, 4, and 5 for k-mers 3, 4, and 5, respectively.
- A 1D MaxPooling layer of size 2 is used after each convolutional layer.
- The features used in the 2D CNN are:
- The kernel sizes in the convolutional layers depend on the k-mer: 3x3, 4x4, and 5x5 for k-mers 3, 4, and 5, respectively.
- 2D MaxPooling layer of size 2x2 after each convolutional layer.
- General features used in each CNN:
- Flattening layer followed by the convolutional layers.
- Optimizer: Adam.
- Loss function: categorical crossentropy, as it is a multiclass neural network.
- Layer activation: Relu.
- Output connected layer activation: Softmax. Learning rate: 0.01.
- Metric: Accuracy.

The structure of the input data changes depending on whether it is 1D or 2D. In a 1D CNN, the data sequence is one-dimensional, so an n × n matrix is created. On the other hand, the 2D CNN is two-dimensional, so a grayscale image with a depth of 1 (n × n × 1) is created. In both cases, n depends on the size of the k-mer:

- K-mer 3: n = 8
- K-mer 4: n = 16
- K-mer 5: n = 32

Figure 6 shows the representation of the input data as an image for each k-mer.

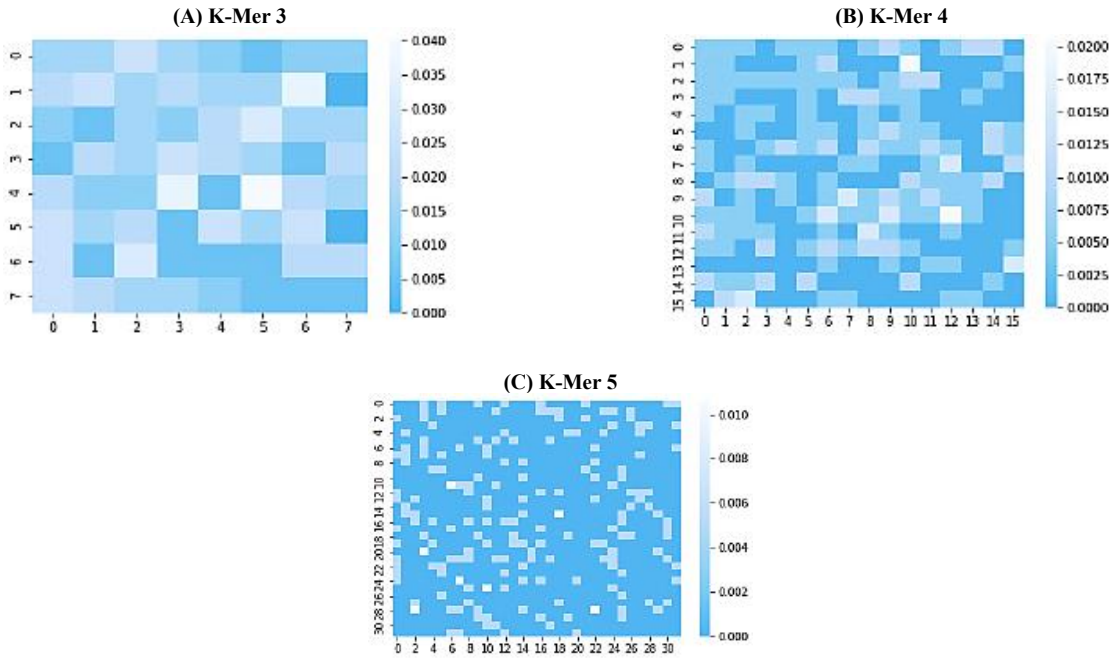


**Fig. 6:** Representation of the Input Data.

Once the structure of the input data was obtained, 7 models were built for each network type. The number of neurons in each layer was determined in the same way as for the multilayer neural network.

Model 1 was based on a LeNet-5 convolutional neural network, as this type of network was used in [12] to classify DNA sequences. Similarly, Model 3 was based on reference [7]. The following Tables (Table 8- 14) show the constructed models with the number of neurons in each layer.

**Table 8:** Model 1

| Layer | Number of Neurons |
|---|---|
| Convolutional 1 | 6 |
| Convolutional 2 | 16 |
| Fully Connected 1 | 120 |
| Fully Connected 2 | 84 |

**Table 9:** Model 2

| Layer | CNN 1D (k-mer 3) | CNN 1D (k-mer 4) | CNN 1D (k-mer 5) | CNN 2D (k-mer 3) | CNN 2D (k-mer 4) | CNN 2D (k-mer 5) |
|---|---|---|---|---|---|---|
| Convolutional 1 | 60 | 60 | 84 | 32 | 32 | 32 |
| Convolutional 2 | 100 | 100 | 100 | 68 | 100 | 36 |
| Fully Connected 1 | 120 | 120 | 90 | 60 | 120 | 120 |
| Fully Connected 2 | 84 | 84 | 112 | 64 | 84 | 64 |

**Table 10:** Model 3

| Layer | Number of Neurons |
|---|---|
| Convolutional 1 | 10 |
| Convolutional 2 | 20 |
| Fully Connected 1 | 500 |

**Table 11:** Model 4

| Layer | CNN 1D (k-mer 3) | CNN 1D (k-mer 4) | CNN 1D (k-mer 5) | CNN 2D (k-mer 3) | CNN 2D (k-mer 4) | CNN 2D (k-mer 5) |
|---|---|---|---|---|---|---|
| Convolutional 1 | 80 | 50 | 80 | 120 | 20 | 20 |
| Convolutional 2 | 120 | 120 | 90 | 60 | 120 | 120 |
| Fully Connected 1 | 112 | 84 | 112 | 64 | 84 | 64 |
| Fully Connected 2 | 80 | 50 | 80 | 120 | 20 | 20 |

**Table 12:** Model 5

| Layer | CNN 1D (k-mer 3) | CNN 1D (k-mer 4) | CNN 1D (k-mer 5) | CNN 2D (k-mer 3) | CNN 2D (k-mer 4) | CNN 2D (k-mer 5) |
|---|---|---|---|---|---|---|
| Convolutional 1 | 10 | 10 | 10 | 10 | 10 | 10 |
| Convolutional 2 | 20 | 20 | 20 | 20 | 20 | 20 |
| Fully Connected 1 | 30 | 30 | 30 | 30 | 30 | 30 |
| Fully Connected 2 | 60 | 120 | 120 | 60 | 120 | 120 |

**Table 13:** Model 6

| Layer | CNN 1D (k-mer 3) | CNN 1D (k-mer 4) | CNN 1D (k-mer 5) | CNN 2D (k-mer 3) | CNN 2D (k-mer 4) | CNN 2D (k-mer 5) |
|---|---|---|---|---|---|---|
| Convolutional 1 | 112 | 112 | 112 | 32 | 64 | 64 |
| Convolutional 2 | 90 | 90 | 60 | 60 | 90 | 60 |
| Fully Connected 1 | 120 | 100 | 120 | 120 | 100 | 100 |

**Table 14:** Model 7

| Layer | CNN 1D (k-mer 3) | CNN 1D (k-mer 4) | CNN 1D (k-mer 5) | CNN 2D (k-mer 3) | CNN 2D (k-mer 4) | CNN 2D (k-mer 5) |
|---|---|---|---|---|---|---|
| Convolutional 1 | 64 | 84 | 100 | 64 | 32 | 64 |
| Fully Connected 1 | 120 | 100 | 80 | 120 | 120 | 120 |

# 4. Results and analysis

## 4.1. Performance of the balanced and unbalanced datasets in the multilayer neural network

The multilayer neural network was trained for each k-mer and taxon, and the network was evaluated using the validation set. The metrics used to evaluate the model are:

- Accuracy: This is the percentage of correct answers for all classes.
- Precision: This defines how confident the model is in stating that a class truly belongs to that class.
- Recall: This is the model's response to how well it predicts a class.

Since precision and recall are obtained for each class of each taxon, the average of the metrics will be taken for general observation.

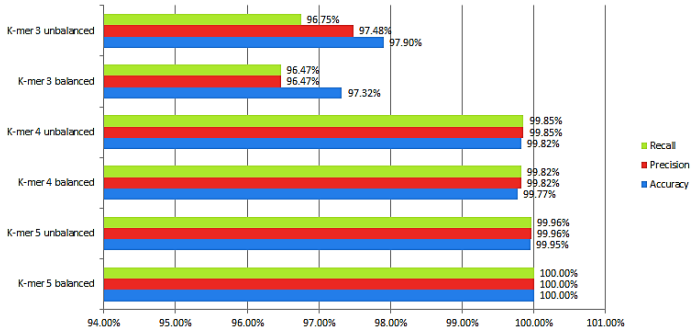


**Fig. 7:** Model Performance in the Taxon Class.

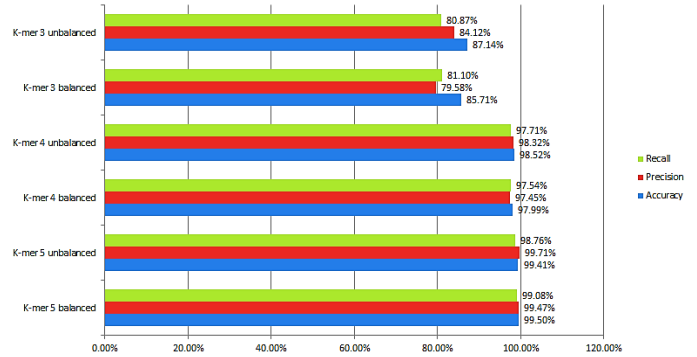


**Fig. 8:** Model Performance on the Order Taxon.

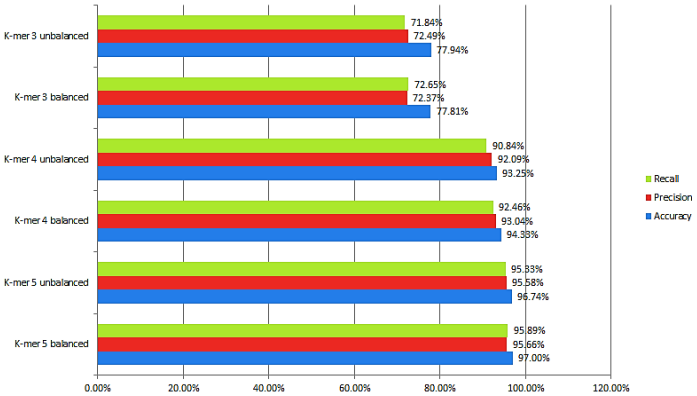


**Fig. 9:** Model Performance on the Family Taxon.

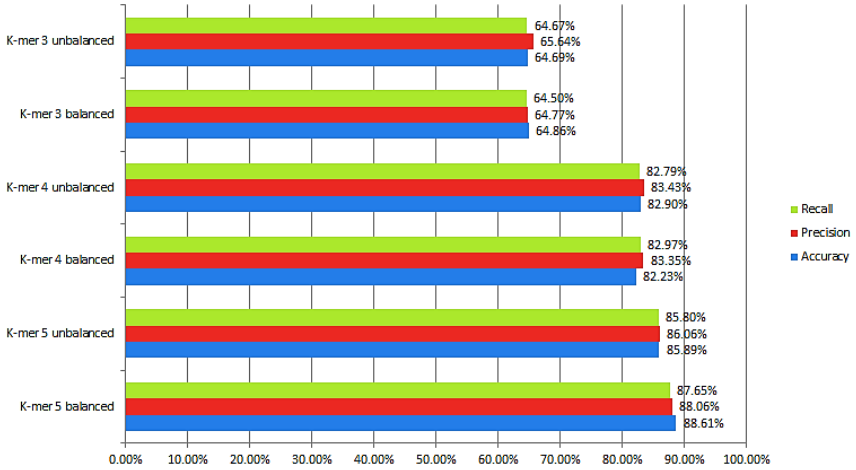


**Fig. 10:** Model Performance on the Genus Taxon.

From Figures 7, 8, 9, and 10, the performance for the balanced and unbalanced datasets is generally good and does not differ significantly, as the metrics have high percentages. In some cases, the unbalanced dataset performed better; however, only the balanced dataset achieved 100% accuracy in the taxon class with k-mer 5. Considering the results, it is best to use the balanced dataset to continue evaluating the following models.

Since high percentages were achieved in the precision and recall metrics, in the following models, only the accuracy metric will be observed for a more general assessment of their performance.

## 4.2. Multilayer neural network results

The multilayer neural network model was trained in 50 epochs to observe the performance in each taxon, in addition to observing the training time.

**Table 15:** Accuracy and Training Time for Each K-Mer and Taxon

| Taxon | K-mer 3 Accuracy (%) | Time (s) | K-mer 4 Accuracy (%) | Time (s) | K-mer 5 Accuracy (%) | Time (s) |
|---|---|---|---|---|---|---|
| Class | 97.32 | 64.1 | 99.77 | 69.42 | 100 | 101.29 |
| Order | 85.71 | 162.33 | 97.99 | 182.04 | 99.5 | 285.68 |
| Family | 77.81 | 303.87 | 94.32 | 349.01 | 97 | 528.81 |
| Genus | 64.86 | 56.58 | 82.23 | 63.07 | 88.61 | 97.89 |

From Table 15, it can be concluded that there is higher performance with a k-mer of 5 and lower performance with a k-mer of 3 for each taxa. Furthermore, accuracy also decreases as the taxa decrease in degree, but it remains at a high percentage. Furthermore, the training time is not very high.

It was previously mentioned that the highest possible accuracy is required for the taxonomic classification of bacteria, so we will seek to improve the taxon-genus classification with convolutional neural network models.

## 4.3. Development of different convolutional neural network models

The seven 1D and 2D CNN models were trained for each k-mer, but only the genus taxon was evaluated, as it has the lowest accuracy. The results obtained are shown in Figure 11.

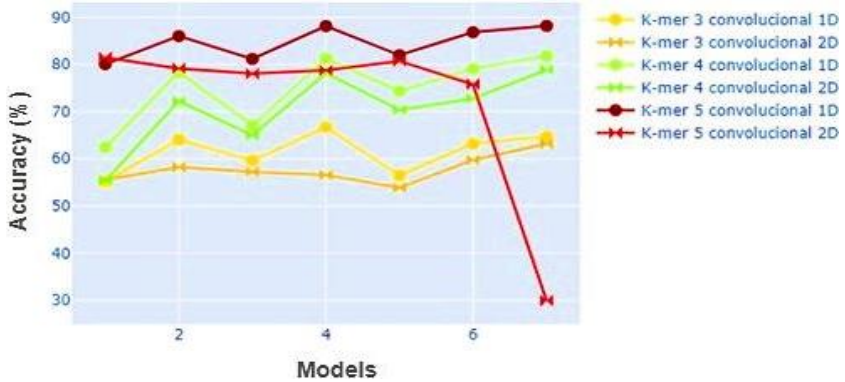


**Fig. 11:** Accuracy of the Models for Each K-Mer and CNN Type.

Considering the results of the multilayer neural network, it is also observed that k-mer 5 is more accurate than the other k-mers. Furthermore, models with 2D CNNs have lower accuracy compared to 1D CNNs. Therefore, for this type of data, convolutional models perform better when viewing the data sequences as one-dimensional. Considering that a 1D CNN is more effective at obtaining features in fixed-length segments and that it does not matter where that feature is in the segment, then the results are accurate, as it describes the sequence of k-mers used. In the methodology section, it was mentioned that there is a pooling layer after each convolutional layer. However, we observed that by removing the pooling layers, the accuracy percentage increased for both CNN types, as can be seen in Tables 16 and 17.

**Table 16:** Comparison of Accuracy and Training Time of the Models with and Without A Pooling Layer for K-Mer 4

| Models | 1D Convolutional Neural Network | | | | 2D Convolutional Neural Network | | | |
|---|---|---|---|---|---|---|---|---|
| | With pooling | | Without pooling | | With pooling | | Without pooling | |
| | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) |
| 1 | 62.41 | 87.55 | 77.85 | 86 | 55.36 | 202.69 | 82.32 | 283.94 |
| 2 | 78.39 | 136.19 | 81.11 | 187.88 | 72.23 | 334.7 | 981.78 | 1569.5 |
| 3 | 67.09 | 114.16 | 79.95 | 137.12 | 64.95 | 234.25 | 81.79 | 735.7 |
| 4 | 81.25 | 105.29 | 84.41 | 119.26 | 78.12 | 240.18 | 84.15 | 304.3 |
| 5 | 74.36 | 126.84 | 77.9 | 134.1 | 70.35 | 384.2 | 77.36 | 565.95 |
| 6 | 79.10 | 163.89 | 83.70 | 207.82 | 72.76 | 480.4 | 81.74 | 748.7 |
| 7 | 81.78 | 112.6 | 85.08 | 125.56 | 80.26 | 263.71 | 84.95 | 389.7 |

**Table 17:** Comparison of Accuracy and Training Time of the Models with and Without A Pooling Layer for K-Mer 5

| Models | 1D Convolutional Neural Network | | | | 2D Convolutional Neural Network | | | |
|---|---|---|---|---|---|---|---|---|
| | With pooling | | Without pooling | | With pooling | | Without pooling | |
| | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) |
| 1 | 80.04 | 155.5 | 85.4 | 161.5 | 81.47 | 202.69 | 87.18 | 1715 |
| 2 | 85.98 | 340.77 | 86.91 | 487.5 | 81.42 | 334.7 | 88.34 | 5936 |
| 3 | 81.16 | 198.63 | 84.06 | 315.9 | 78.03 | 234.25 | 87.58 | 5125 |
| 4 | 88.12 | 241.95 | 88.7 | 252.8 | 78.75 | 240.18 | 88.30 | 1145 |
| 5 | 81.56 | 210 | 84.5 | 257.8 | 80.7 | 384.2 | 85.6 | 2652 |
| 6 | 86.83 | 341.25 | 87.18 | 467 | 75.8 | 480.4 | 86.65 | 13575 |
| 7 | 88.16 | 238.9 | 88.25 | 248.42 | 29.95 | 263.71 | 88.43 | 2699 |

It can be observed that there is greater accuracy for both 1D and 2D CNNs of each k-mer when the pooling layers are removed. Training time also increases. The reason for adding a pooling layer is to reduce the size of the features received from the convolutional layer, thus reducing training time and making the features more robust [10]. Removing the pooling layer increases training time because the feature size is not reduced. It is possible that, in this type of data, the pooling layer does not consider features that contain important information, which is why removing the layer results in better performance.

The highlighted percentages are the models that achieved the highest accuracy. Similarly, it can be observed that k-mer 5 continues to perform better than k-mer 4. Therefore, we will work with k-mer 5.

In addition to removing the pooling layer, overfitting was also observed in each model. Model 4 of the 1D CNN obtained the highest accuracy percentage with 88.7% on the validation data, but on the training data, it obtained 97.03%. One solution to overfitting is to apply penalties to layer parameters during optimization. These penalties will encourage the network to maintain small weights during training (Abbas et al., 2023).

From the keras library, the kernel regularizer is used with l2, which calculates the sum of the squared values of the weights (Abbas et al., 2023).

**Table 18:** Accuracy of Each Model when Applying L2 Regularization

| Accuracy (%) Model | CNN 1D | CNN 2D |
|---|---|---|
| 1 | 86.78 | 89.10 |
| 2 | 88.08 | 88.16 |
| 3 | 88.7 | 88.74 |
| 4 | 88.61 | 89.33 |
| 5 | 86.021 | 87.72 |
| 6 | 89.46 | 88.03 |
| 7 | 88.39 | 88.43 |

Table 18 – 19 shows that both CNN types increased accuracy by up to 2%, with model 6 of the 1D CNN and model 4 of the 2D CNN achieving the highest accuracy compared to the other models. Therefore, both models were trained and evaluated for each taxon. When comparing the CNN models, it was observed that removing the connecting layers consistently improved the accuracy for 1D and 2D architectures—typically between 2% and 5%. This suggests that merging can remove important sequence features in k-mer-based representations. Among the models tested, 1D CNNs outperformed their 2D counterparts, confirming that one-dimensional architectures are more suitable for handling linear DNA sequence data. Further improvement was observed using L2 regularization, which reduced overlap and improved generalization. The best result was obtained by Model 6 (without 1D CNN pooling and with L2 regularization), reaching an accuracy of 89.46% in the classification of genera. Despite these advantages, multi-layer neural networks (MLNs) are competitive, especially for higher taxonomic levels, achieving comparable accuracies with short training times.

**Table 19:** Comparison of Accuracy and Training Time in Each Taxon with the Chosen Models

| Taxon | CNN 1D | | CNN 2D | |
|---|---|---|---|---|
| | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) |
| Class | 99.77 | 480.96 | 99.86 | 1284.26 |
| Order | 97.63 | 1289.85 | 99.19 | 3484.53 |
| Family | 93.7 | 2399.9 | 95.66 | 5904.8 |
| Genus | 89.46 | 434.63 | 89.33 | 1101.79 |

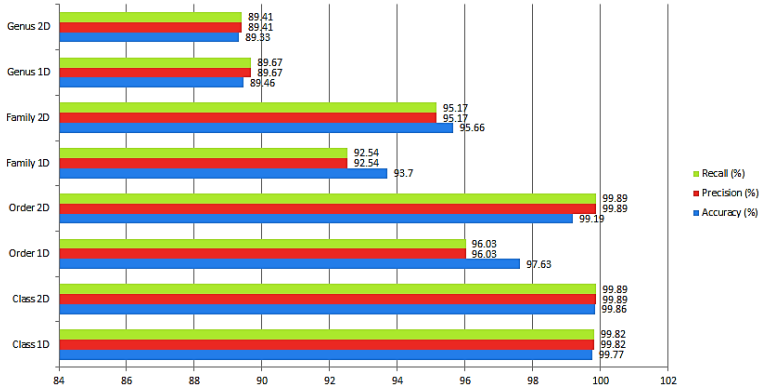


**Fig. 12:** Metric Results for the Selected Models.

Figure 12 shows the percentage of metrics for the two selected models for each taxon. Despite having a better result for the genus taxon, the accuracy for the other taxa did not surpass that of the multilayer neural network model. However, the convolutional neural networks also perform well in terms of precision, recall, and accuracy.

### 4.4. Model selected for the taxonomic classification of bacteria

Finally, the best model was selected considering accuracy, precision, recall, and training time. First, the training time of all models was considered. The multilayer neural network takes the shortest time, as it is the simplest network, followed by the 1D and 2D CNNs. Furthermore, although better results were obtained by removing the pooling layers, the time increased significantly, in some cases, double the time.

Now, when analyzing the metrics, the multilayer neural network achieves greater accuracy for the first three taxa, while the taxon genus achieves greater accuracy for the 1D and 2D CNN models. However, the difference is 1%. This is why the multilayer neural network was chosen. Multi-layer neural networks (MLNs) perform well in DNA taxonomic classification because k-mer frequency vectors can represent ordered and aggregated sequences, making the patterns mostly linear and discrete. Unlike CNNs, which are much better at drawing local features, MLNs efficiently learn from these global k-mer features without the need for complex architectures. Their simplicity allows for faster learning, making k-mer frequencies a powerful benchmark during DNA classification. To ensure the reproducibility of the results, 42 random seeds were used in all Python randomization modules, NumPy, TensorFlow, and the SMOTE algorithm. The SMOTE technique, implemented using the imblearn library, was fitted with random_state=42 and k_neighbors=5 to generate synthetic examples for minority classes. SMOTE was applied to the training dataset only, leaving the validation and test sets unchanged. This setup allowed for obtaining stable training data while maintaining consistent experimental conditions.

Figure 13 shows the accuracy and loss of the chosen model when trained for 100 epochs.

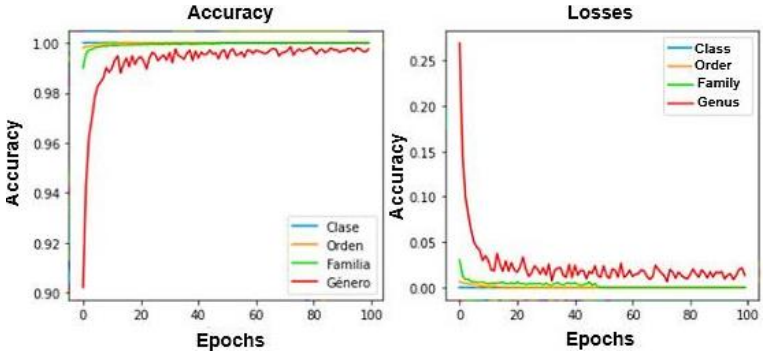


**Fig. 13:** Accuracy and Loss During Training for Each Taxon.

After training, the model was evaluated on the test data. The accuracy and training time for each taxon are shown in Table 20.

**Table 20:** Accuracy when Evaluating Test Data

| Taxon | Accuracy (%) | Time (s) |
|---|---|---|
| Class | 100 | 163.64 |
| Order | 99.21 | 450.68 |
| Family | 96.73 | 821.3 |
| Genus | 88.21 | 161.2 |

The other metrics were also observed, but this time for each of the classes.

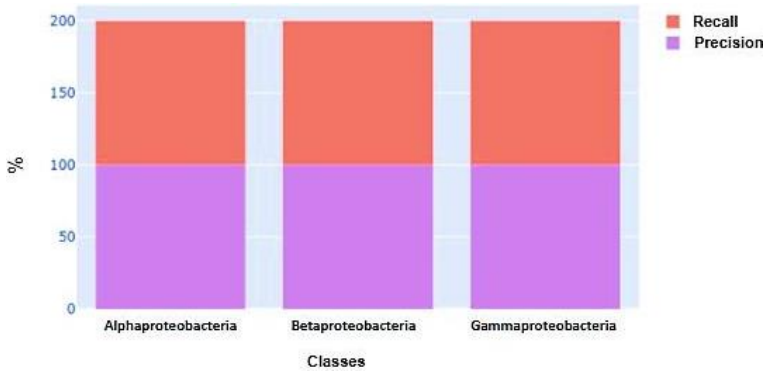


**Fig. 14:** Precision and Recall for the Class Taxon.

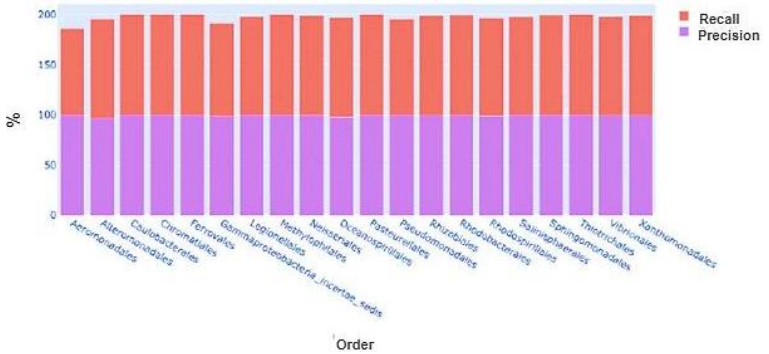


**Fig. 15:** Precision and Recall for the Order Taxon.

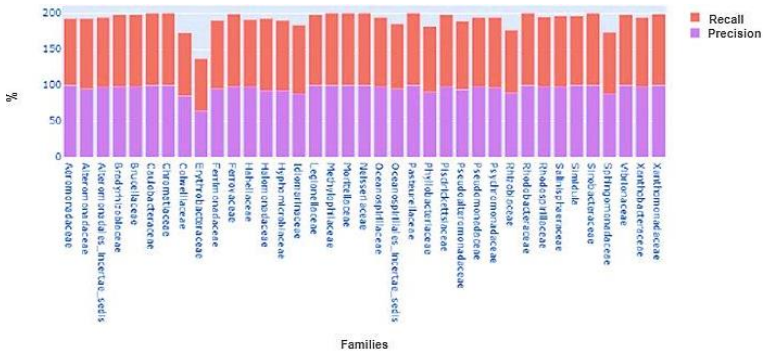


**Fig. 16:** Precision and Recall for the Family Taxon.

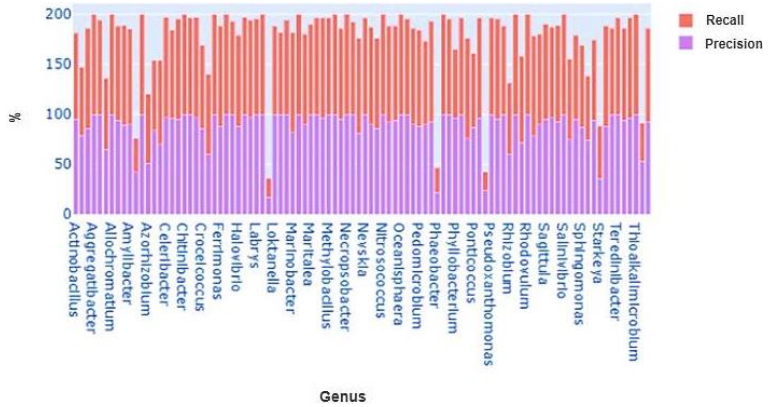


**Fig. 17:** Precision and Recall for the Genus Taxon.

Figures 14-17 show the precision and recall for each of the bacteria by taxon.
In the class and order taxons, it is observed that both metrics are high, meaning the model correctly classifies the classes.
In the family taxon, there are two classes with lower precision and recall than the others: Erythrobacteraceae and Colwelliaceae. Looking at the data distribution, these two bacteria are not the ones with the fewest data points. The Moritellaceae bacterium has the fewest data points and obtained the highest precision and recall. Therefore, there are two possible explanations. The first is that the synthetic data for the two bacteria did not have adequate information. The second is that the DNA sequence did not have this information.
Within the genus taxon, several bacteria obtained low percentages of precision and recall, especially Leisingera and Pseudophaeobacter. These are not part of the families with low precision and recall. Similarly, the two possible explanations given above could be the cause of the low results. Table 21 shows that simple multilayer neural network (MLN) architectures provide strong baseline accuracy, while 1D

CNNs—especially L2-based Model 6—achieve the highest accuracy (89.46%) in gender classification. improved over 2D CNNs. Overall, MLNs had a good balance between accuracy and efficiency, making them suitable for practical DNA classification tasks.

**Table 21:** Summary of Model Architectures

| Model Type | Model No. | Layers | Key Parameters | Pooling | Regularization | Best Accuracy (%) |
|---|---|---|---|---|---|---|
| MLN | — | 2 Hidden Dense Layers | Relu Activation, Adam Optimizer | — | None | 88.21 (Genus) |
| 1D CNN | 1 | 2 Conv + 2 Dense | Conv filters: 6, 16 | With/Without | L2 (λ=0.001) in selected runs | 89.46 |
| 1D CNN | 2 | 2 Conv + 2 Dense | Conv filters: 60, 100 | With/Without | L2 in selected runs | 86.91 |
| 1D CNN | 3 | 2 Conv + 1 Dense | Conv filters: 10, 20 | With/Without | L2 in selected runs | 88.70 |
| 1D CNN | 4 | 1 Conv + 2 Dense | Conv filters: 80 | With/Without | L2 in selected runs | 88.70 |
| 1D CNN | 5 | 3 Conv + 2 Dense | Conv filters: 10, 20, 30 | With/Without | L2 in selected runs | 84.50 |
| 1D CNN | 6 | 2 Conv + 1 Dense | Conv filters: 112, 90 | With/Without | L2 Applied | 89.46 |
| 1D CNN | 7 | 1 Conv + 1 Dense | Conv filters: 64 | With/Without | L2 in selected runs | 88.25 |
| 2D CNN | 1–7 | Same as 1D, adapted for 2D | Same filters as 1D models | With/Without | L2 in selected runs | 89.33 |

Compared with traditional machine learning approaches, our Multilayer Neural Network (MLN) showed the best performance. Specifically, MLN achieved 88.2% accuracy for genus-level classification, outperforming baseline models such as Support Vector Machines (SVMs) in [5], which achieved 82% accuracy under similar conditions. This highlights the advantage of deep learning models, even architectures such as MLNs, in capturing complex patterns within k-mer frequency datasets for DNA taxonomic classification.
The use of SMOTE helps to balance between classes, but may introduce bias, as synthetic samples do not represent true biological diversity. This creates risk and reduces the generalizability of real-world data. Transparent reporting and external validation are essential to address these issues.

### 4.5. Limitations and future extensions

This study has several limitations. First, the models were trained and validated on a specific DNA dataset, which may limit their generalizability to other organisms or larger genomic data sets. Sequence context by relying on K-mer frequency vectors is abstract and higher-level alignments may be missing. In addition, removing the mixing layers and adding regularization improved the accuracy, but these choices increased the computational cost and training time, creating a trade-off between performance and efficiency.
For future work, extending the models to use larger and more diverse datasets and evaluating performance on external validation sets will strengthen their robustness. Another direction is the development of real-time classification tools for practical applications in genomics using improved lightweight models or hardware accelerators. The introduction of explainable AI techniques can help interpret model predictions and improve confidence in biological research contexts.

## 5. Conclusions

This study evaluated various deep learning models for DNA sequence classification using k-mer vector representations, focusing on their performance across different taxonomic levels. The k-mer representation effectively captured key sequence features, with optimal results achieved at a k-mer size of 5. Performance declined with smaller k values, suggesting that larger k-mers retain more discriminative features. However, sizes beyond 5 were not explored, so conclusions about higher values remain speculative. Among the models tested, one-dimensional convolutional neural networks (1D CNNs) outperformed two-dimensional (2D CNNs) counterparts in both accuracy and training efficiency. The 1D format appeared more suited to extracting features from linear DNA sequences. Interestingly, removing the pooling layers in CNNs increased classification accuracy by up to 5% in 2D CNNs and slightly less in 1D CNNs—indicating that pooling may lead to the loss of crucial information, especially with smaller input sizes. Nevertheless, eliminating pooling also significantly increased training times, suggesting a trade-off between accuracy and computational efficiency. In some cases, 2D CNNs without pooling layers slightly outperformed 1D CNNs, showing that both data representations can be equally effective under the right conditions. Model performance decreased with lower taxonomic levels, likely due to the difficulty in distinguishing closely related sequences and class imbalance. Still, the models achieved 88% accuracy at the genus level, with high precision and recall across most classes. Notably, the simplest model—a multilayer neural network—achieved the best overall performance, despite being computationally expensive. This finding is significant given that most reviewed studies favored CNNs and other complex architectures. It suggests that simple architectures can still be powerful when appropriately tuned, though more comparisons with additional machine learning models are necessary for a conclusive assessment. Overall, the study highlights the effectiveness of k-mer encoding and deep learning in taxonomic classification of DNA sequences.

## References

[1] LaPierre, N., Ju, C. J. T., Zhou, G., & Wang, W. (2019). MetaPheno: A critical evaluation of deep learning and machine learning in metagenome-based disease prediction. *Methods*, 166, 74–82. https://doi.org/10.1016/j.ymeth.2019.01.002.

[2] Phan, D., Ngoc, G. N., Lumbanraja, F. R., Faisal, M. R., Abipihi, B., Purnama, B., Delimiyanti, M. K., Kubo, M., & Satou, K. (2017). Combined use of k-mer numerical features and position-specific categorical features in fixed-length DNA sequence classification. *Journal of Biomedical Science and Engineering*, 10(7), 390–401. https://doi.org/10.4236/jbise.2017.108030.

[3] Huang, Y., Yang, L., & Wang, T. (2011). Phylogenetic analysis of DNA sequences based on the generalized pseudo-amino acid composition. *Journal of Theoretical Biology*, 269(1), 217–223. https://doi.org/10.1016/j.jtbi.2010.11.007.

[4] Remita, M. A., Halioui, A., Diouara, A. A. M., Daigle, B., Kiani, G., & Diallo, A. B. (2017). A machine learning approach for viral genome classification. *BMC Bioinformatics*, 18, 208. https://doi.org/10.1186/s12859-017-1602-3.

[5] Tonkovic, P., Kalajdziski, S., Zdravevski, E., Lameski, P., Corizzo, R., Pires, I. M., Garcia, N. M., Loncar-Turukalo, T., & Trajkovik, V. (2020). Literature on applied machine learning in metagenomic classification: A scoping review. *Biology*, 9(11), 453. https://doi.org/10.3390/biology9120453.

[6] Rehman, M. U., Tayara, H., & Chong, K. T. (2022). DL-m6A: Identification of N6-methyladenosine sites in mammals using deep learning based on different encoding schemes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20, 904–911. https://doi.org/10.1109/TCBB.2022.3192572.

[7] Pham, T. D. (2007). Spectral distortion measures for biological sequence comparisons and database searching. *Pattern Recognition*, 40(2), 516–529. https://doi.org/10.1016/j.patcog.2006.04.021.

[8] Hossain, P. S., Kim, K., Uddin, J., Samad, M. A., & Choi, K. (2023). Enhancing taxonomic categorization of DNA sequences with deep learning: A multi-label approach. *Bioengineering*, 10, 1293. https://doi.org/10.3390/bioengineering10111293.

[9] Soliman, N. (2022). An improved convolutional neural network model for DNA classification. *Computers, Materials & Continua*, 70(3). https://doi.org/10.32604/cmc.2022.018860.

[10] Liang, Q., Bible, P. W., Liu, Y., Zou, B., & Wei, L. (2020). DeepMicrobes: Taxonomic classification for metagenomics with deep learning. *NAR Genomics and Bioinformatics*, 2(1), Article lqaa009. https://doi.org/10.1093/nargab/lqaa009.

[11] Ghoneim, A., Muhammad, G., & Hossain, M. S. (2020). Cervical cancer classification using convolutional neural networks and extreme learning machines. *Future Generation Computer Systems*, 102, 643–649. https://doi.org/10.1016/j.future.2019.09.005.

[12] Abbas, Z., Ur Rehman, M., Tayara, H., Zou, Q., & Chong, K. T. (2023). XGBoost framework with feature selection for the prediction of RNA N5-methylcytosine sites. *Molecular Therapy*, 31(6), 2543–2551. https://doi.org/10.1016/j.ymthe.2023.02.020.

[13] Abd-Alhalem, S. M. (2020). Bacterial classification with convolutional neural networks based on different data reduction layers. *Nucleosides, Nucleotides & Nucleic Acids*, 39(4), 493–503. https://doi.org/10.1080/15257770.2019.1645851.

[14] Malonzo, M. H., & Lähdesmäki, H. (2023). LuxHMM: DNA methylation analysis with genome segmentation via hidden Markov model. *BMC Bioinformatics*, 24, Article 58. https://doi.org/10.1186/s12859-023-05213-3.

[15] Gunasekaran, H., Ramalakshmi, K., Arokiaraj, A. R. M., Kanmani, S. D., Venkatesan, C., & Dhas, C. S. G. (2021). Analysis of DNA sequence classification using CNN and hybrid models. *Computational and Mathematical Methods in Medicine*, 2021, Article 1835056. https://doi.org/10.1155/2021/1835056.