Complex-Valued Deep Learning for Genomic Sequencing

Suttawee Lukuan[1], Prompong Sugunnasil[1], Sumalee Sangamuang[1], and Waranya Mahanan[1]

[1]Department of Data Science, Faculty of Engineering, Chiang Mai University, Chiang Mai 50200, Thailand
suttawee_lukuan@cmu.ac.th

Abstract. The work presents the application of complex-valued deep learning for classifying microbial organisms, highlighting its significance for rapid pathogen identification crucial in healthcare. It explores the efficiency of complex-valued neural networks over traditional real-valued networks, focusing on efficiency, computational resource usage, and accuracy in genome sequencing classification. The research employs theoretical analysis and empirical testing, comparing the performance of complex-valued and real-valued models. Findings indicate that complex-valued CNNs offer advantages in encoding genomic sequences and processing efficiency. The study's significance lies in its potential to advance pathogen classification methods, offering insights into the practical trade-offs between model complexity and computational efficiency, and contributing to the development of more effective tools for epidemic prevention and control.

Keywords: Organism classification · Bacteria · Complex-Valued neural network

1 Introduction

Identifying and classifying microbial organisms plays an important role in healthcare, especially in the prevention and management of epidemics. The rapid and precise identification of pathogens enables healthcare systems to implement timely and effective measures to prevent the spread of infectious diseases. The process of identifying and classifying microbes, including viruses, involves genomic sequencing

and bioinformatics analysis. This approach allows for accurately detecting pathogens, understanding their genetic makeup, and tracking their mutations and evolution. Accurate identification is essential for understanding the epidemiology of infectious diseases, tracking the spread of pathogens, and identifying new or emerging infectious agents [1].

Genome sequencing has become a pivotal tool in detecting and managing pathogens, revolutionizing infectious disease surveillance and response approaches. The advent of genome sequencing technologies has enabled rapid, accurate identification of viruses, even in the early stages of an outbreak. For instance, during the SARS-CoV-2 pandemic, the lack of routine viral genomic surveillance initially allowed the virus to spread unchecked in the U.S., highlighting the critical role of genomic data in tracking virus evolution and spread [2]. Genome sequencing provides detailed insights into the genetic makeup of viruses, facilitating the tracking of mutations and the emergence of new variants. This capability is crucial for understanding the dynamics of virus transmission and identifying areas lacking in surveillance, as demonstrated by the study on Aedes-borne viruses like Zika and Chikungunya in the Americas [3]. There are several genome sequencing techniques, such as Next-Generation Sequencing (NGS) [4] and Whole-Genome Sequencing (WGS)[5], and a variety of applications including diagnostic microbiology[6], crop improvement[7], and pathogen monitoring[8].

This research proposes a resource-efficient method for classifying the organism based on genome sequencing. Machine learning is widely used to assist the sequencing process and is an important tool in genome sequencing. A variety of machine learning is proposed to assist the task, ranging from supervised learning and unsupervised learning to deep learning. Despite advancements, genome sequencing can be expensive and time-consuming [9], which can be a huge challenge in this field of study. The strategy of this work is to use complex-value based machine learning. Complex numbers are a type of number that extends the traditional concept of

numbers to a new dimension of information. They consist of two parts: a real part and an imaginary part. This structure allows operations that are impossible with real numbers, enabling them to represent data in another form and solve a broader range of mathematical problems. Consequently, the complex number can enhance the machine learning model in terms of efficiency and resource consumption.

2 Literature Review

2.1 Microbial Organism Classification

Microbial Organism Classification is a critical field in microbiology that involves categorizing microorganisms based on various characteristics, including genetic, functional, and phenotypic traits. This classification is essential for understanding microbial diversity, ecology, and the specific roles microbes play in different environments, including their impact on human health. Various methods have been developed, each with its unique approach and application. The classification can be based on metabolite contents. This approach involves classifying microorganisms based on the volatile organic compounds (VOCs) they emit. Abdullah *et al.* (2019) utilized this method to classify microorganisms into their pathogenicity based on VOCs, using data from the KNApSAcK and mVOC databases. On the other hand, function-based classification is based on the functional repertoire of microorganisms rather than just phylogenetic markers. Zhu *et al.* (2015) proposed a classification scheme, functional-repertoire similarity-based organism network (FuSiON). This technique constructs a systemic approach to the organism's function and measures the function. More recent research on microbial organism classification is shown in Table 1.

Table 1. Summary of recent articles on microbial organism classification

Year	Authors	Target Organism	Classification Techniques
2023	Qu *et al.*[10]	River microbial communities	Ensemble machine learning models
2023	Choudhury *et al* [11].	Microbial communities in North Indian rivers	Iterative K-Means clustering, MLAs, Random Forest
2023	Alharbi and Vakanski [12]	Cancer (gene expression analysis)	Deep learning-based approaches
2023	Ahmad *et al.* [13]	Coxiella burnetii in soil	Two-phase featureranking, SVM, LDA,LR, MLP

2.2 Complex-Valued Machine Learning

Complex numbers are a type of number that extend the real numbers by adding an imaginary unit, denoted as i, where $i^2 = -1$. A complex number is generally expressed in the form $a + bi$, where a and b are real numbers, and i is the imaginary unit. The real part of the complex number is a, and the imaginary part is b. The complex number can be represented in various forms for different purposes. The exponential form of a complex number is an elegant and powerful way of representing complex numbers, utilizing Euler's formula. The formula connects complex exponentials with trigonometric functions. Given a complex number $z = a + bi$, where a and b are real number, we can rewrite the z into its exponential form as $z = re^{i\theta}$ where r is the magnitude of the complex number $r = |z| = \sqrt{a^2 + b^2}$ and θ is the phase angle $\theta = arg(z) =$ atan2(b,a). Despite the fact that the complex number can be converted into a 2D vector, this process does not incorporate the mathematical correlation between the real and imaginary part[14]. Complex numbers are fundamental in various fields, including engineering, physics, and applied mathematics, particularly in solving polynomial equations that do not have real solutions.

Complex-Value Machine Learning is an area in computer science that focuses on the application of complex numbers in machine learning algorithms. This approach involves using complex numbers and their properties to enhance the learning capabilities of algorithms. Unlike real-valued data, complex numbers have both a real and an imaginary component, which can be particularly useful in representing multidimensional data in a more compact form. The use of complex numbers allows for a more detailed representation of data, capturing additional information compared to real-valued counterparts. Moreover, machine learning can integrate complex values into the computational model to increase the capabilities. Table 2 displays the summary of the complex-number related techniques. There are two forms of implementation. One of them is to represent the data in the complex number format. For example, electroencephalography (EEG) signals can be transformed into the complex-value format to use as input for the model[15]. The other aspect of the implementation is the to use as part of the machine learning model, such as artificial neural network or deep learning. Dramsch *et al.* studied non-stationary physical data, seismic data, using complex-valued deep convolutional networks. The complex-valued deep convolutional networks is a form of convolutional neural networks that use the complex value as an additional component. However, there are several limitation to the complex-valued machine learning. One of the most important issues it that the implementation of complex-valued algorithms can be more challenging than traditional real-valued algorithms.

Table 2. Summary of Articles Utilizing Complex-Value Machine Learning

Year	Authors	Title	Domain	Complex-Value Usage
2021	Dramsch *et al.* [16]	Complex-valued neural networks for machine learning on non-stationary physical data	Seismic data	Complex-valued machine learning
2021	Wang *et al.* [17]	An efficient specific emitter identification method based on complex-valued neural networks and network com-pression	Specific emitter identification data	Complex-valued machine learning
2021	Zhang *et al.*[18]	An optical neural chip for implementing complexvalued neural network	Benchmark data (XOR logic gate, iris[19], Circle and Spiral, and MNIST[20])	Complex-valued machine learning
2016	Peker [21]	An efficient sleep scoring system based on EEG signal using complex-valued machine learning algorithms	Medical	Data representation and complex-valued machine learning
2012	Savitha *et al.* [22]	Fast learning circular complex-valued extreme learning machine (CCELM) for real-valued classification problems	Benchmark data[23]	Complex-valued machine learning

3 Complex-Valued Deep Learning for Microbial Organism Classification

3.1 Complex-Valued Artificial Neural Network

Artificial Neural Networks (ANNs) are a cornerstone of modern machine learning, drawing inspiration from the biological neural networks that constitute the mammal nervous system [24]. An ANN is a computational model mimicking how neurons interact in the human brain, enabling the machine to learn from observational data. It has been used to recognize patterns and solve complex problems in data analysis, such as image recognition and natural language processing [25].

One of the successful tasks is a comparative study for classifying quantum states that have demonstrated that complex-valued neural networks perform significantly better than traditional, witness-based methods[26]. And another study about PolSAR image classification[27] models mentioned that they are typically based on real-valued CNNs (RV-CNN). This means the models' parameters, inputs, and outputs are all real-valued. However, since the raw data of PolSAR images are generally complex-valued, it is impossible to input the complex-valued raw data into the real-valued model directly, but complex-valued neural network could use it directly. Its experimental results reveal that the method proposed in its study is much better than the real-valued model despite having comparable parameters.

There are several key components of the ANNs:

– **Neurons**. These are the basic units of computation in an ANN. Each neuron receives input, processes it, and passes on its output to the next layer of neurons.

– **Layers**. These are a series of interconnected neurons. They function as the primary structural and computational elements within an ANN [28].

– **Weights**.Weights represent the strength of the connection between neurons. There is a special type of weight called bias. The bias is added to fine-tune the output.

– **Activation Functions**. The activation function is a decision function to determine whether it should be activated. Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and Softmax. A different activation function is used for different tasks [29].

– **Loss Function**. This function measures the difference between the network's prediction and target values. It guides the training process by indicating how far off the predictions are. Common loss functions include Mean Squared Error (MSE), Mean Absolute Error (MAE), and Cross-Entropy Loss [30].

The operation of an ANN can be broadly divided into two main phases: training and recalling (also known as inference or prediction). The training phase is a process to determine the weights of the model. Normally, it involves determining the differences between the target and actual values and adjusting (updating) the weight. The training phase is computationally intensive and can take considerable time, especially for large networks and datasets. The famous training algorithm is the back-propagation algorithm [31]. The other approach to determine the weight is to use optimization like the Adam optimization algorithm [32]. In contrast, the recalling phase is a straightforward process to determine the result from the input and the weight without the weight update. The recalling phase is generally faster than the training phase, which involves straightforward computations through the already-trained network.

The following two formulas represent forward propagation of complex-valued neural networks, which is essentially identical to real-valued neural networks.

$$z_{[l,j]} = \sum_{k} \left[a_{[l-1,k]} w_{[l,j,k]} \right] + b_{[l,j]} \quad (1)$$

$$a_{[l,j]} = \sigma\left(z_{[l,j]}\right) \quad (2)$$

When z denotes the output of an inactivated layer, it represents the state before the activation function is applied but after forward propagation from the previous layer.

a denotes the activated layer, and w denotes the weights associated with the neural network connections. The symbol σ is introduced to represent the activation function, details of which will be mentioned in later sections.

The index notation on the lower right of each variable is to provide a clearer representation of variables. $z[l,k]$ denotes the inactivated state of node k in layer l, $a[l,k]$ denotes the activated value of node k in layer l, and $w[l,j,k]$ represents the weights associated with the connection from node k in the activated layer of the previous $l-1$ layer to branch j in layer l.

Complex-Valued Split-SoftMax Layer. In machine learning, particularly in neural networks, the SoftMax layer plays an important role in classification tasks. The SoftMax function is an activation function in a neural network model, especially in the output layer for multiclass classification problems. The general idea of the function is to read a vector of k and generate a vector of K real values that sum to 1. The result of the SoftMax function can be interpreted as probability. Given a vector $z \in R^k$, the SoftMax function $\sigma(z_k)$ for the k-th element is defined as:

$$\sigma\left(\mathbf{z}_k\right) = \frac{e^{z_k}}{\sum_{k=0}^{K-1} e^{z_k}} \qquad \text{for } k = 0,\ldots,K-1 \tag{3}$$

Where K is the number of classes, and the output of the SoftMax function is a probability distribution over K classes. On the other hand, Split-SoftMax function [33] is an extension of the SoftMax function to support complex values.The input of the layer is in a complex-valued format, and the final output is still a vector of probability. The complex-valued output of the last layer is converted into two real-valued outputs, one for the real parts and one for the imaginary parts. Then, SoftMax is applied to both outputs separately to get the probabilities of each class, and then the separate pairs are recombined to form a complex number again. Given a vector $z \in C^{1\times K}$, the Split-

SoftMax function $\sigma(z_k)$ for the k-th element is defined as:

$$\sigma_{\text{split}}\ (\mathbf{z}_k) = \frac{e^{\Re\{z_k\}} + ie^{\Im\{z_k\}}}{\sum_{k=0}^{K-1}\left(e^{\Re\{z_j\}} + e^{\Im\{z_j\}}\right)} \quad \text{for } k = 0, \ldots, K-1 \tag{4}$$

The Split-SoftMax [34] function considers real and imaginary parts. The advantage is concatenating $2n$ real-valued output nodes to n complex-valued output nodes, which may provide a more compact representation of complex relationships, potentially reducing the dimensionality of the output space, a complexvalued output vector here of $C^{1\times K}$ is equivalent to a real-valued output vector of $R^{1\times 2K}$ to a real-valued machine learning models' point of view.

$$\begin{pmatrix} 0.02 + 0.10i \\ 0.80 + 0.05i \\ 0.01 + 0.02i \end{pmatrix}^T \rightarrow \begin{pmatrix} 0.02 \\ 0.10 \\ 0.80 \\ 0.05 \\ 0.01 \\ 0.02 \end{pmatrix}^T$$

Fig.1. Complex Split-Softmax Layer's output (left) to a real-valued machine learning models' point of view to interpret is probabilities (right).

However, its disadvantage is the number of target classes must be even. If the number of target classes is odd, you could introduce a new dummy target class, which would not have any input vectors associated with it. This would ensure that the number of target classes is even. The traditional cross-entropy loss function for real-valued neural networks' classification task is.

$$C = -\sum_{k} \left[y_k \ln \left(a_{[L,k]} \right) \right] \qquad \text{for } k = 0,\ldots,K-1 \qquad (5)$$

When C represents the total loss, which is calculated by summing over all the classes. y_u is the true label for class u (either 0 or 1), and a_L is the predicted probability of that class (a value between 0 and 1) at the final layer of the neural network. *ln*() represents the natural logarithm function. The cross-entropy loss function aims to penalize the model for predicting a low probability for the true class and a high probability for other classes and to encourage the model to assign high probabilities to the true class. The goal of training the neural network is to minimize the value of C by adjusting the weights and biases of the model.

$$C = -2\sum_{k} \left[R\{y_k\} \ln \left(R\left\{ a_{[L,k]} \right\} \right) + I\{y_k\} \ln \left(I\left\{ a_{[L,k]} \right\} \right) \right] \qquad \text{for } k = 0,\ldots,K-1 \qquad (6)$$

$$\frac{\partial C}{\partial z_{[l,j]}} = \overline{a_{[L,J]} - y_{[J]}} \qquad (7)$$

The two formulas above are a modified version of the cross-entropy loss function called “split-cross-entropy loss function” for complex-valued neural networks and is R-derivative respectively. In this modified version, the loss function considers both the real and imaginary parts of the output and treats them as a separate class. The natural logarithm function is applied to both the real and imaginary parts of the predicted output, and the loss for each class is the sum of the products of the real and imaginary parts of the true label and the logarithms of the corresponding real and imaginary parts of the predicted output. Finally, the overall loss is the negative sum of the losses for all the classes. This modified loss function is used to train the complex-valued neural network, to minimize the difference between the predicted output and the true label. The reason why split-cross-entropy loss function is multiplied by -2 instead of -1 is to not having any constant multiplication at equation 7, otherwise it have to be multiplied by 0.5 or divided by 2.

Complex-valued back-propagation One of the widely known activation functions in classical real-valued neural networks is the hyperbolic tangent (tanh) function. While its output is bounded for all real numbers, it loses this property in the complex domain and includes singularities (e.g., at inputs like $-0.5\pi i, 0.5\pi i, 1.5\pi i, 2.5\pi i, \ldots$). Consequently, it is not advisable to employ tanh as an activation function in complex-valued networks. An analytic (holomorphic) function unavoidably has unbounded points, as per Liouville's theorem[35], unless it is a constant function, rendering it impractical as it yields the same output irrespective of the input. Therefore, non-analytic (non-holomorphic) activation functions are preferred. However, traditional back-propagation, designed for real-valued networks, is effective only for holomorphic or analytic functions. The ensuing eight equations correspond to the real-valued Adaptive Moment Estimation back-propagation algorithm (ADAM) [36], which does not perform optimally in complex-valued neural networks.

$$\frac{\partial C}{\partial b_{[l,j]}} = \frac{\partial C}{\partial z_{[l,j]}} \qquad (8)$$

$$v_{b\,[l,j]_{new}} = \beta_1 v_{b\,[l,j]_{old}} + (1-\beta_1)\frac{\partial C}{\partial b_{[l,j]}} \qquad (9)$$

$$s_{b\,[l,j]_{new}} = \beta_2 s_{b\,[l,j]_{old}} + (1-\beta_2)(\frac{\partial C}{\partial b_{[l,j]}})^2) \qquad (10)$$

$$b_{[l,j]_{new}} = b_{[l,j]_{old}} - \eta\left(\frac{\frac{v_{b[l,j]_{new}}}{1-{\beta_1}^t}}{\sqrt{\frac{s_{b[l,j]_{new}}}{1-{\beta_2}^t}+\varepsilon}}\right) \qquad (11)$$

$$\frac{\partial C}{\partial W_{[l,j,k]}} = \frac{\partial C}{\partial b_{[l,j]}} a_{[l-1,k]} \qquad (12)$$

$$v_{W\,[l,j,k]_{new}} = \beta_1 v_{W\,[l,j,k]_{old}} + (1-\beta_1)\frac{\partial C}{\partial W_{[l,j,k]}} \qquad (13)$$

$$s_{W_{[l,j,k]_{new}}} = \beta_2 s_{W_{[l,j,k]_{old}}} + (1-\beta_2)\left(\frac{\partial C}{\partial W_{[l,j,k]}}\right)^2 \quad (14)$$

$$W_{[l,j,k]_{new}} = W_{[l,j,k]_{old}} - \eta \left(\frac{\frac{v_{W_{[l,j,k]_{new}}}}{1-\beta_1{}^t}}{\sqrt{\frac{s_{W_{[l,j,k]_{new}}}}{1-\beta_2{}^t}+\varepsilon}} \right) \quad (15)$$

For complex-valued backpropagation, the loss function remains real-valued, serving to minimize empirical risk throughout the learning process. Although there's a minor adjustment to accommodate complex-valued inputs, the primary challenge in Complex-Valued Neural Networks (CVNN) lies in utilizing effective training methods. A problem emerges during the implementation of the learning algorithm, commonly referred to as backpropagation. Optimizing the network's parameters demands the use of gradients or any partial-derivative-based algorithm. However, standard complex derivatives are defined only for holomorphic or analytic functions. Fortunately, Wirtinger calculus [37] extends the concept of complex derivatives, encompassing holomorphic functions as a specific case. This extension enables the realization of "complex-valued backpropagation" for both holomorphic and non-holomorphic functions under Wirtinger Calculus. Presented here are eight equations corresponding to the Complex-Valued Adaptive Moment Estimation, or "CVADAM" [38] backpropagation algorithm, which operates effectively on complex-valued neural networks. Remind that the cost function in the complex-valued neural network, denoted as C, remains real-valued.

$$\frac{\partial C}{\partial b_{[l,j]}} = \frac{\partial C}{\partial z_{[l,j]}} \quad (16)$$

$$v_{b_{[l,j]_{new}}} = \beta_1 v_{b_{[l,j]_{old}}} + (1-\beta_1)\frac{\partial C}{\partial b_{[l,j]}} \quad (17)$$

$$s_{b_{[l,j]_{new}}} = \beta_2 s_{b_{[l,j]_{old}}} + (1-\beta_2)\left|\frac{\partial C}{\partial b_{[l,j]}}\right|^2 \quad (18)$$

$$b_{[l,j]_{new}} = b_{[l,j]_{old}} - \eta \left(\frac{\frac{\overline{v_{b[l,j]_{new}}}}{1-\beta_1{}^t}}{\sqrt{\frac{s_{b[l,j]_{new}}}{1-\beta_2{}^t}+\varepsilon}} \right) \quad (19)$$

$$\frac{\partial C}{\partial W_{[l,j,k]}} = \frac{\partial C}{\partial b_{[l,j]}} a_{[l-1,k]} \quad (20)$$

$$v_{W_{[l,j,k]_{new}}} = \beta_1 v_{W_{[l,j,k]_{old}}} + (1-\beta_1) \frac{\partial C}{\partial W_{[l,j,k]}} \quad (21)$$

$$s_{W_{[l,j,k]_{new}}} = \beta_2 s_{W_{[l,j,k]_{old}}} + (1-\beta_2) \left| \frac{\partial C}{\partial W_{[l,j,k]}} \right|^2 \quad (22)$$

$$W_{[l,j,k]_{new}} = W_{[l,j,k]_{old}} - \eta \left(\frac{\frac{\overline{v_{W[l,j,k]_{new}}}}{1-\beta_1{}^t}}{\sqrt{\frac{s_{W[l,j,k]_{new}}}{1-\beta_2{}^t}+\varepsilon}} \right) \quad (23)$$

Alternatively, you have the option to separately apply the ADAM algorithm to the real and imaginary components. Presented below are eight equations associated with the Complex (Split) Separable Adaptive Moment Estimation, often referred to as ”CSADAM” or ”splitADAM.” While the comparison of performances between ”CVADAM” and ”CSADAM” is yet to be tested, you can choose for either of these methods to train Complex-Valued Multilayer Perceptrons (CVMLPs).

$$\frac{\partial C}{\partial b_{[l,j]}} = \frac{\partial C}{\partial z_{[l,j]}} \quad (24)$$

$$v_{b_{[l,j]_{new}}} = \beta_1 v_{b_{[l,j]_{old}}} + (1-\beta_1) \frac{\partial C}{\partial b_{[l,j]}} \quad (25)$$

$$s_{b_{[l,j]_{new}}} = \beta_2 s_{b_{[l,j]_{old}}} + (1-\beta_2)(\Re\left\{\frac{\partial C}{\partial b_{[l,j]}}\right\}^2 + \Im\left\{\frac{\partial C}{\partial b_{[l,j]}}\right\}^2 i) \quad (26)$$

$$b_{[l,j]_{new}} = b_{[l,j]_{old}} - \eta \left(\frac{\frac{\Re\{v_{b[l,j]_{new}}\}}{1-\beta_1{}^t}}{\sqrt{\frac{\Re\{s_{b[l,j]_{new}}\}}{1-\beta_2{}^t}+\varepsilon}} - \frac{\frac{\Im\{v_{b[l,j]_{new}}\}}{1-\beta_1{}^t}}{\sqrt{\frac{\Im\{s_{b[l,j]_{new}}\}}{1-\beta_2{}^t}+\varepsilon}} i \right) \quad (27)$$

$$\frac{\partial C}{\partial W_{[l,j,k]}} = \frac{\partial C}{\partial b_{[l,j]}} a_{[l-1,k]} \quad (28)$$

$$v_{W_{[l,j,k]_{new}}} = \beta_1 v_{W_{[l,j,k]_{old}}} + (1-\beta_1) \frac{\partial C}{\partial W_{[l,j,k]}} \quad (29)$$

$$s_{W_{[l,j,k]_{new}}} = \beta_2 s_{W_{[l,j,k]_{old}}} + (1-\beta_2)(\Re\left\{\frac{\partial C}{\partial W_{[l,j,k]}}\right\}^2 + \Im\left\{\frac{\partial C}{\partial W_{[l,j,k]}}\right\}^2 i) \quad (30)$$

$$W_{[l,j,k]_{new}} = W_{[l,j,k]_{old}} - \eta \left(\frac{\frac{\Re\{v_{W[l,j,k]_{new}}\}}{1-\beta_1{}^t}}{\sqrt{\frac{\Re\{s_{W[l,j,k]_{new}}\}}{1-\beta_2{}^t}+\varepsilon}} - \frac{\frac{\Im\{v_{W[l,j,k]_{new}}\}}{1-\beta_1{}^t}}{\sqrt{\frac{\Im\{s_{W[l,j,k]_{new}}\}}{1-\beta_2{}^t}+\varepsilon}} i \right) \quad (31)$$

Convolutions Neural Network on Genomics

Although convolutions neural networks (CNNs) [39] have been applied to a variety of computational genomics problems, there remains a large gap in our understanding of how they build representations of regulatory genomic sequences. Here, we perform systematic experiments on synthetic sequences to reveal how CNN architecture, specifically convolutional filter size and max pooling, influences the extent to which sequence motif representations are learned by first layer filters. CNNs are designed to extract features from nucleotide and amino acids sequences. There is a research mentioned about a deep learning genome classification strategy targeting SARS-CoV-2 which is capable of working with 31,029 bp (base pairs), and such convolutions neural network is proven to be an effective way to correctly classify viruses, especially SARS-CoV-2, into their realms, families, genera, and subgenera.

Pooling is an important, if not necessary operation in Convolutional Neural Networks (CNNs) for several reasons, it reduces the spatial dimensions of the input

volume. This is crucial for reducing the computational complexity of the network and controlling overfitting. Max pooling and average pooling are frequent pooling methods in Real-valued Convolutional Neural Networks (RVCNNs). Average pooling is straightforward for both real-valued and complex-valued CNNs. But for real-valued CNNs' max pooling, it is not straightforward for complex-valued CNNs'. There are at least three ways to extend it here. First, split max pooling for complex-valued convolutional neural networks involves applying max pooling separately to the real and imaginary parts of the complex-valued activations. Second, magnitude max pooling is a pooling operation applied to complex-valued matrix where for each submatrix, the complex number with the highest magnitude (absolute value) is chosen. Third, real-part max pooling involves applying max pooling only to the real part of the complex-valued activations, then the imaginary part of such value is transferred. The corresponding imaginary part is also considered, serving as a tiebreaker in case of multiple maximum values in the real part.

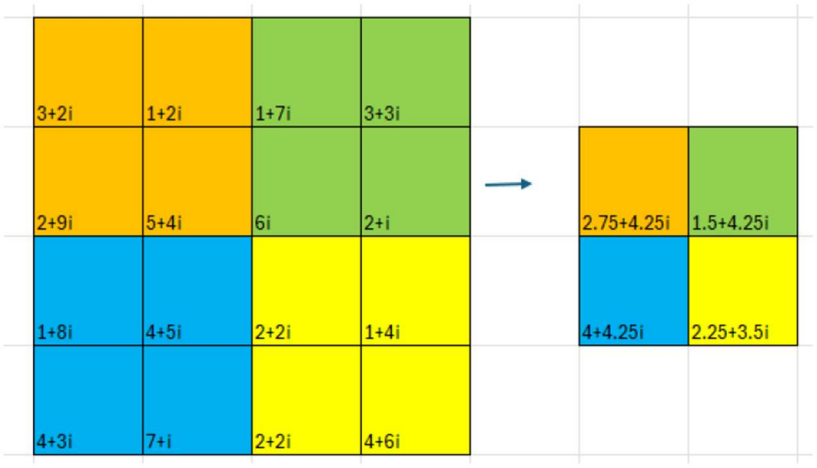


Fig. 2. An example of 2 × 2 Average Pooling

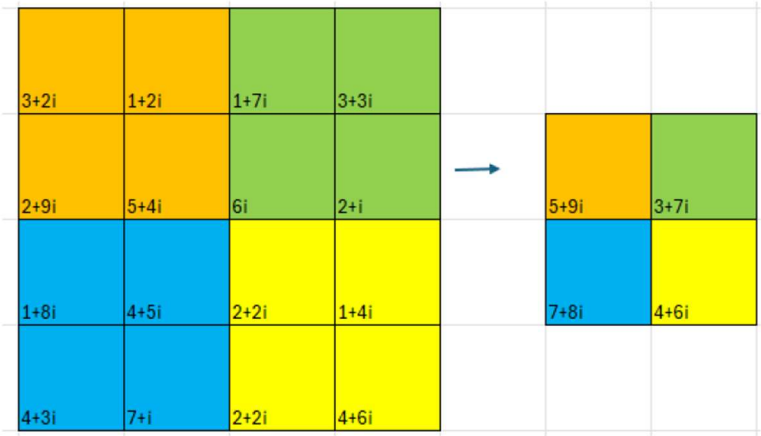


Fig. 3. An example of 2 × 2 Split Max Pooling

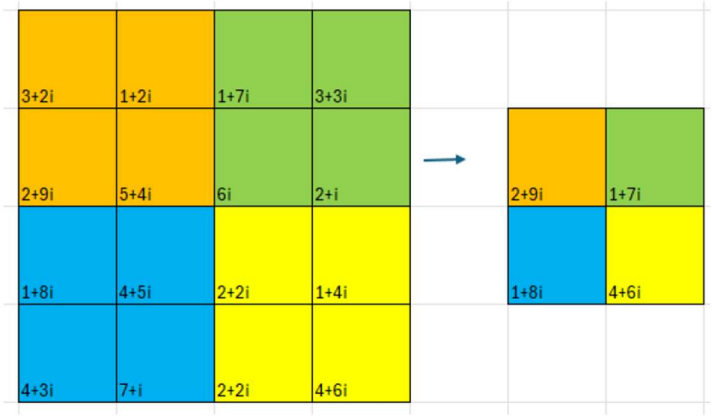


Fig. 4. An example of 2 × 2 Magnitude Max Pooling

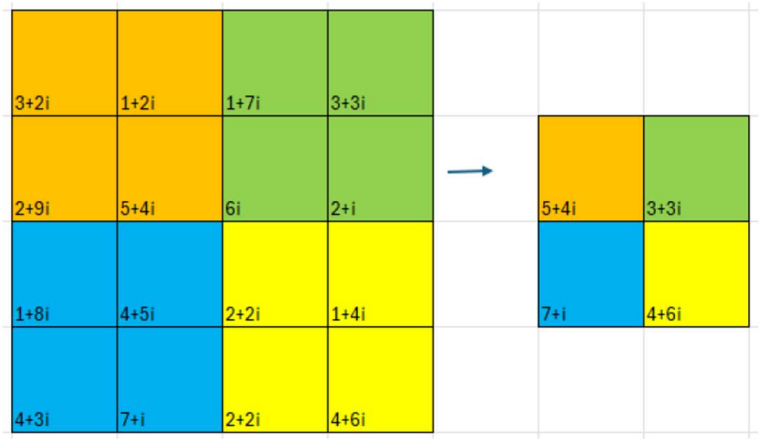


Fig. 5. An example of 2 × 2 Real-Part Max Pooling

3.2 Workflow

This study presents a pipeline for microorganism classification from nucleotides and amino acids. The working strategy of this research can be depicted in three stages. The first stage is to prepare the data for the analysis. Secondly, a 2D CNN is either trained or uses predefined-weight, or randomly generated to capture the structure of the data. Finally, the knowledge extracted by the CNN is used as the initial weight and fine-tuning for a better result firm. Figure 6 provides a visual representation of a three-stage process of the proposed method.

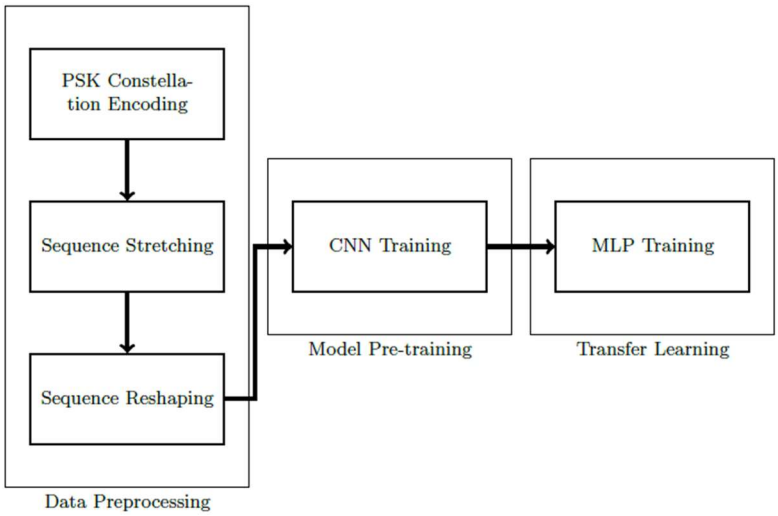


Fig. 6. Overall concept of the proposed method

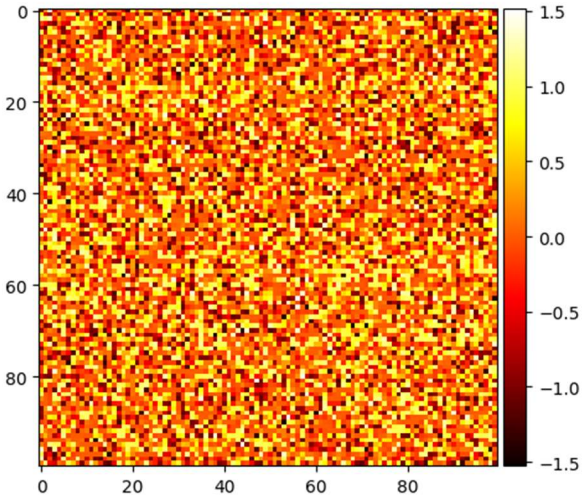
3.3 Data Preprocessing

The process of preparing data for complex-valued deep learning models involves several distinct yet interconnected steps, each of which is crucial for ensuring that the data is appropriately formatted and encoded for the learning task at hand. Figure 6 shows the visualization of the process. Initially, the PSK Constellation Encoding step is applied to the dataset. This involves encoding each nucleotide sequence which is composed by nucleotide bases such as A(adenine), C(cytosine), G(guanine), or T(thymine), into a complex number. The inspiration for this encoding comes from modulation techniques commonly used in digital communication systems, such as Phase-Shift Keying (PSK) constellation encoding, Amplitude-Phase-Shift Keying (APSK) constellation encoding, or Quadrature Amplitude Modulation (QAM) constellation encoding. These methods represent information as different phases of a carrier signal, thereby transforming the data into a format suitable for subsequent processing in a deep learning context. Subsequently, the data, which may initially be represented as a 1D sequence, undergoes a transformation in the Sequence Reshaping

step. This is where the 1D sequences are reshaped into a 2D square format, making them compatible for application of convolutional neural networks (CNNs).

An optional step involves applying the Discrete Fourier Transform (DFT). While not crucial, the DFT can expose frequency domain information within the signal, which can be instrumental in detecting certain periodic patterns. This step underscores the versatility of the preprocessing methods in adapting the data for various analytical purposes. Each of these steps plays a vital role in the preprocessing pipeline, ensuring

that the data is optimally prepared for the complex demands of deep learning models in bioinformatics, enhancing the accuracy and efficiency of the models used for analysis.



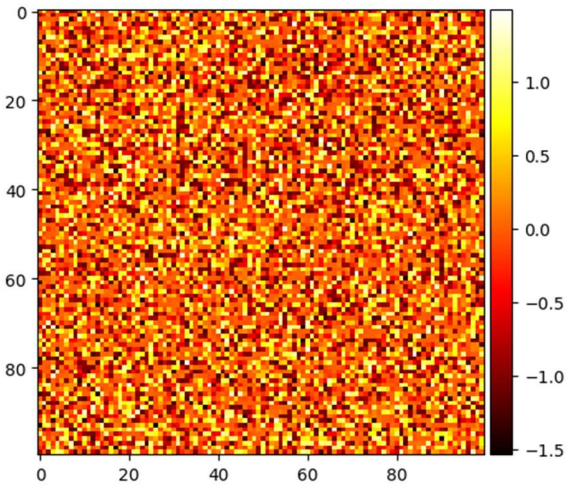


Fig. 7. Real Part (Up) and Imaginary Part (Down) of a Processed Nucleotide Sequence from a bacterium after encoding, resizing, and reshaping from 1D to 2D

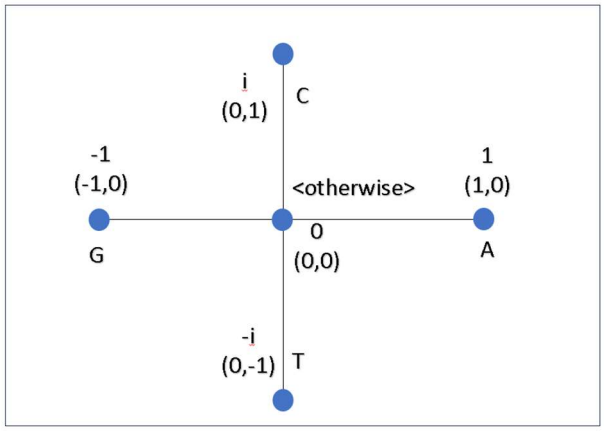


Fig. 8. PSK-based constellation encoding for nucleotides

3.4 Model Pre-Training

2D CNNs are used to find the pattern of the reshaped sequence. The 2D CNN is uniquely adept at handling two-dimensional data. The strength of CNNs lies in their ability to learn hierarchical features from the data. This capability is particularly valuable in signal classification tasks, as it enables the extraction of complex patterns and features that may not be immediately apparent.

3.5 Transfer Learning

Finally, transfer learning is employed. This technique involves using pre-trained models to enhance the performance of the current task. In this specific context, the features learned by the previously applied 2D CCNN are transferred to a MLP. This transfer is not merely a transfer of data but of learned insights and discovered patterns that the CNN has extracted from the nucleotide sequences. Once these features are integrated into the MLP, it can further refine them and perform more focused organism classification tasks. This approach takes advantage of the hierarchical feature extraction capabilities of CNNs and the pattern recognition and classification strengths of MLPs. MLP undergoes a specialized training process. This training aims to finetune the model specifically for organism classification tasks based on nucleotide and amino acid sequences. This step involves a detailed optimization of the model's weights and biases. By combining the strengths of CNNs in feature extraction and MLPs in pattern recognition and classification, these steps collectively enhance the accuracy and efficiency of organism classification based on DNA sequences. The model specifically for organism classification tasks based on nucleotide and amino acid sequences. This step involves a detailed optimization of the model's weights and biases. By combining the strengths of CNNs in feature extraction and MLPs in pattern recognition and classification, these steps collectively enhance the accuracy and efficiency of organism classification based on DNA sequences

4 Experiments and Result

4.1 Experimental Setup

The nucleotides and amino acids sequences to form the dataset is from 8 types of bacteria. They are Chlamydia trachomatis[40], Clostridium tetani[41], Escherichia coli[42], Lactobacillus bulgaricus[43], Mycobacterium tuberculosis[44], Salmonella enterica[45], Streptococcus suis[46], and Vibrio cholerae [47]. All of the dataset is retrieved from the National Center for Biotechnology Information (NCBI) (https://www.ncbi.nlm.nih.gov/). The number of each sequence is shown in Table 3 with a total number of 14009 and an average length of 39383.

Table 3. Number of Nucleotide Sequences

Organism	**Alias**	**Number of Nucleotide Sequences**	**Minimum Length**	**Maximum Length**	**Average Length**
Chlamydia trachomatis	B0	1648	1000	93379	9435.09
Clostridium tetani	B1	2895	1000	99926	23605.11
Escherichia coli	B2	1300	50737	51000	50866.97
Lactobacillus bulgaricus	B3	1772	35001	61987	46908.80
Mycobacterium Tuberculosis	B4	1201	43501	55988	48976.26
Salmonella enterica	B5	1770	50000	50999	50471.01
Streptococcus suis	B6	1955	45008	54948	50145.04
Vibrio cholerae	B7	1538	45006	54995	49364.54

This study uses two classes of artificial neural networks to demonstrate the proposed method's performance: multilayer perceptron (MLP) and convolution neural network (CNN). Both of the classes are implemented in both real number system and the complex value system. Figure 7 displays the overall architecture of CNNs for this article. Figure 8a uses the datatype of complex64, which is implemented by Python's numpy. Both the real and imaginary parts of the complex64 are float32. The aliases of the real number convolution neural network and complex-value convolution neural network are RVCNN and CVCNN, respectively. On the other hand, float32 data type in Figure 8c is 32-bit single precision floating point real number of Python's numpy. In Table 4, three classes of MLP are displayed. The core differences between each class include the activation function and the back-propagation algorithm. RVCNN_0 is a real-valued CNN which uses two real-valued channels after PSK encoding, real and imaginary parts, then feeds its flatten layer into RVMLP_0. RVCNN_1 is also a real-valued CNN. The difference is that it uses two real-valued channel after PSK encoding, magnitude, and phase parts, then feed its flatten layer into RVMLP_1.

Table 4. Architectures of Multilayer Perceptrons

Parameter	**RVMLP_0, RVMLP_1**	**CVMLP_0**	**CVMLP_1**
Input length	450	225	225
Hidden layer count	10	10	10
Hidden layer length (each)	8	6	6
Hidden layer Activation Function (each)	ReLU	cardioid	cardioid
Output length	10	5	5
Output layer Activation Function	Softmax	Split softmax	Split softmax
Back-propagation Algorithm	ADAM	CVADAM	CSADAM

The performance measurement of this work focuses on resource consumption in terms of encoding time and training time. Then, the performance during the training process is studied via the loss. Finally, the accuracy of the models is studied. This article shows the confusion of the matrix, accuracy, F1, and precision of all classes and all models. The formula of the complex cardioid activation function is stated below.

$$cardioid(z) = \frac{1+cos(arg(z))}{2} z \quad (32)$$

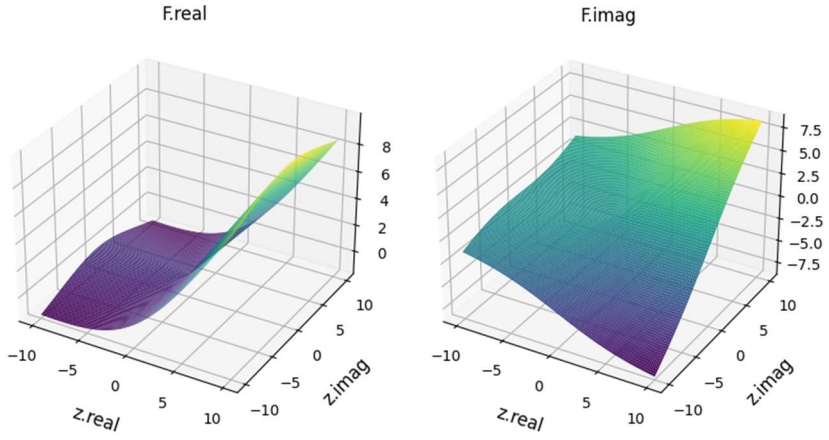
And its CR-derivatives are, which are used in complex-valued backpropagation…

$$\frac{\partial}{\partial z} cardioid(z) = \frac{1+cos(arg(z))}{2} + \frac{i}{4} sin(arg(z)) \quad (33)$$

$$\frac{\partial}{\partial \bar{z}} cardioid(z) = \frac{-i}{4} sin(arg(z))(\hat{n}(z))^2 \quad (34)$$

The following term is a normalization of a complex number z.

$$\hat{n}(z) = \frac{z}{|z|} = \frac{z^{\frac{1}{2}}}{\bar{z}^{\frac{1}{2}}} = e^{i\, arg(z)} \quad (35)$$



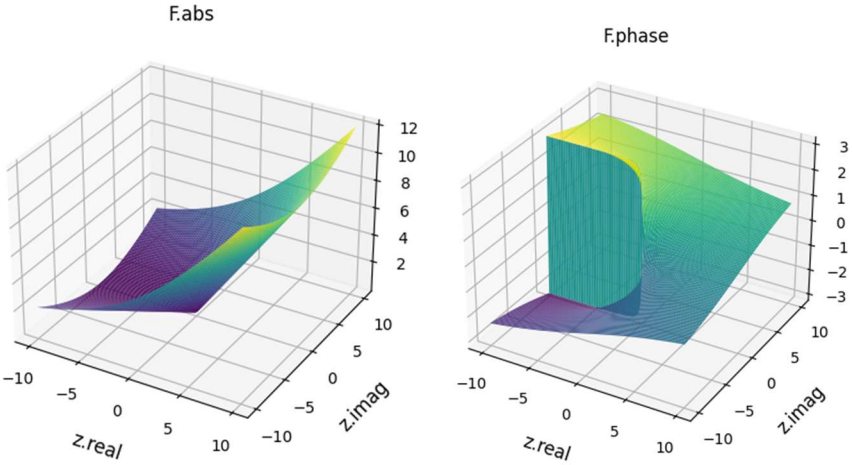
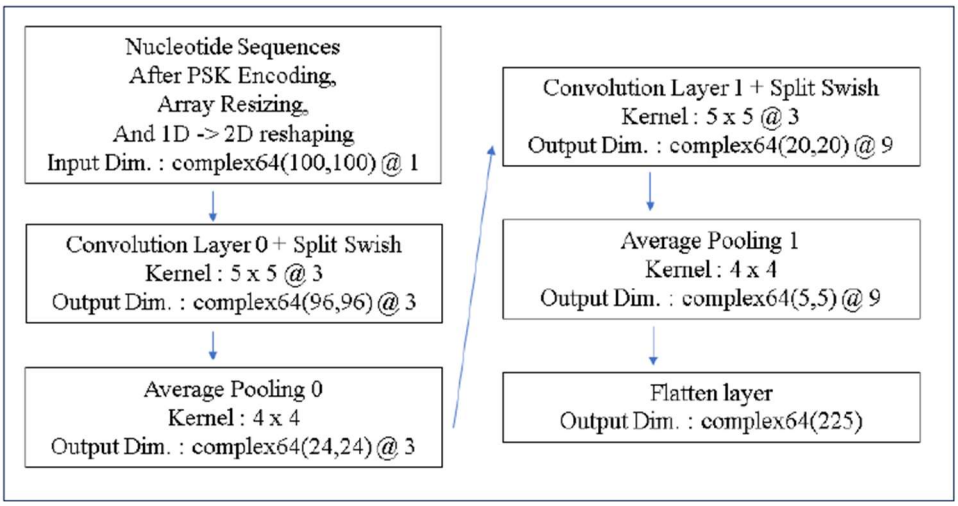


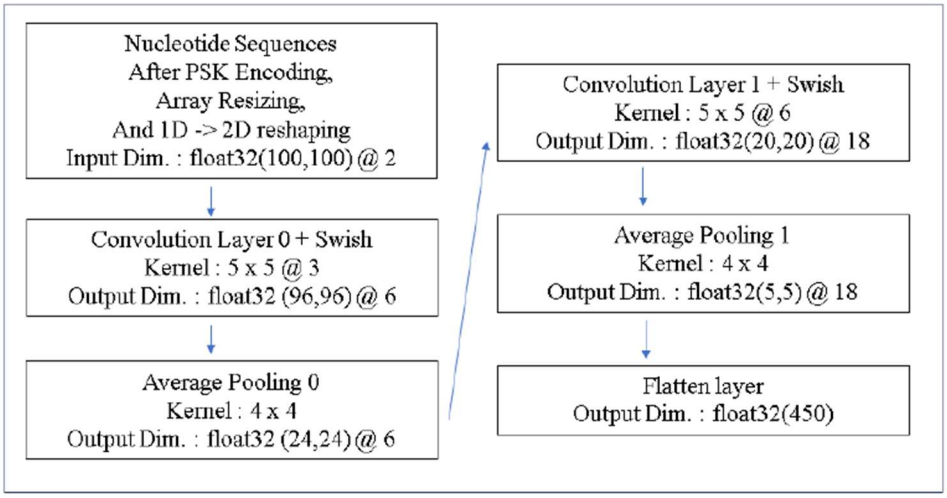
Fig. 9. Real part(Upper-left), Imaginary Part(Upper-right), Magnitude(Lower-left), and Phase(Lower-right) of the complex cardioid activation function

4.2 Experiment Results

In Table 5 and Table 6, the machine learning CVCNN 0 demonstrates superior efficiency in nucleotide encoding and overall processing time, outperforming its real-valued counterparts RVCNN 0 and RVCNN 1. However, when considering MLP models, the model RVMLP 1 exhibits the quickest training time, suggesting greater efficiency in certain training contexts. Despite a slightly higher parameter count in Complex-Valued MLPs CVMLP 0 and CVMLP 1, these models showcase commendable performance, particularly in validation accuracy, indicating better generalization to new data. This analysis reveals a nuanced trade-off between speed, efficiency, and accuracy, suggesting that the choice between Real-Valued and Complex-Valued models should be tailored to specific project requirements in genome sequencing classification.



(a) Complex-Valued Convolution Neural Network



(b) Real-Valued Convolution Neural Network

Fig. 10. Architectures of Convolution Neural Networks

Table 5. Comparison of Resource Consumption Between Real-Valued and Complex-Valued Convolutional Neural Networks

Parameter	**RVCNN_0**	**RVCNN_1**	**CVCNN_0**
Nucleotide Encoding Time (seconds)	718.96	695.33	**514.15**
Convolutional Neural Network Processing Time (seconds)	974.19	877.92	**787.80**
Total Time Before Sending to Multilayer Perceptron	1693.15	1573.25	**1301.95**
Total Real-valued Parameter Count	**23981**	23981	24056

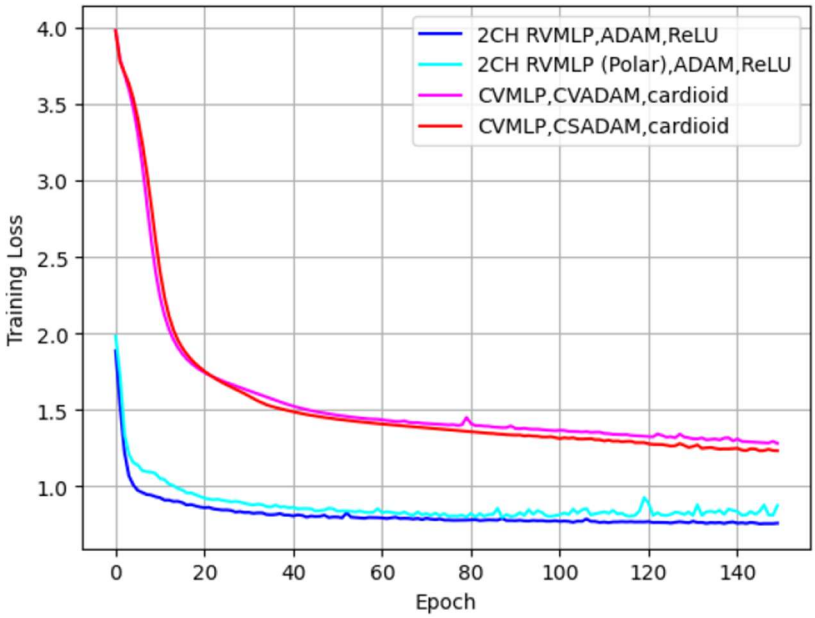
Table 6. Comparison of Resource Consumption Between Multilayer Perceptron Models

Multilayer Perceptron	**RVMLP_0**	**RVMLP_1**	**CVMLP_0**	**CVMLP_1**
Convolutional Neural Network	RVCNN_0	RVCNN_1	CVCNN_0	CVCNN_0
Convolutional Neural Network Processing Time (seconds)	1693.15	1573.25	**1301.95**	**1301.95**
Multilayer Perceptron Training Time (seconds)	5956.47	**5385.86**	6784.63	8659.23
Total Training Time (seconds)	7649.62	**6959.11**	8086.58	9961.18
Real-valued Trainable Parameter Count	4328	4328	**3524**	**3524**
Final Train Loss	**0.7599518**	0.8753283	1.2815065	1.2335334
Final Train Accuracy	0.7074240	0.6665306	0.7466857	**0.7568835**
Final Validation Accuracy	0.6823697	0.6509636	**0.7118725**	0.7030692

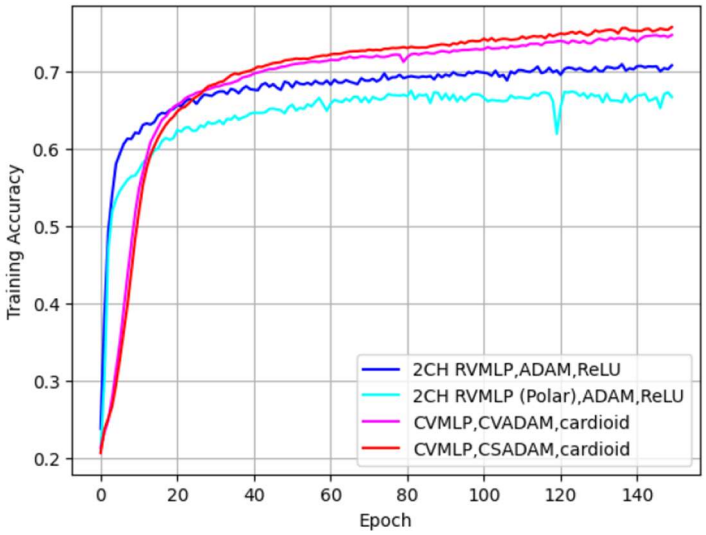
Figure 8 shows experiments assessing the evolution of the three indicators, training loss, training accuracy, and evaluation accuracy, and the number of epochs. In Figure 8a, all models show a rapid decrease in training loss as the number of epochs increases, which is typical as the model learns from the training data. The loss curves begin to plateau as the models approach an optimal state. The relation between training accuracy and the number of training epochs is studied in Figure 8c. The models appear to converge to a similar accuracy level by the end of the training, with minor fluctuations. The complex-valued, cardioid models seem to have a more consistent accuracy, whereas the real-valued, ReLU ones fluctuate more. Figure 8c displays studies on the relation between evaluation accuracy and a number of training epochs. The evaluation accuracy for all models shows less stability than the training accuracy, which is common due to the generalization of unseen data. The model labeled CVMLP, CVADAM, cardioid seems to have the highest evaluation accuracy, suggesting that with the CVADAM optimizer and cardioid activation function is particularly effective for this dataset.

According to Table 7, RVMLP_0 performs well in classifying B0 with high true positives but has a significant number of false negatives, mistaking B0 for B6 frequently. However, the model struggles with B2, B3, B5, B7, particularly B2 and B3, where it often confuses them with other classes, suggesting similarities in features, a lack of distinctiveness in the learning representation for these species, or RVMLP_0 lacks the capacities to distinguish these classes. B1, B4, and B6 are classified with high accuracy, indicating distinctive features that the model can learn effectively. Table 8 shows the confusion matrix of the model RVMLP_1. B1, B4, and B5 have a very high accuracy, similar to RVMLP_0, while B6 has less accuracy than RVMLP_0. B3 has a better performance compared to RVMLP_0, with some confusion with other classes. B2 and B7 have poor performance with a lot of confusion with other classes. B2 has no true positives, indicating a possible need for reevaluation of the feature representation for B2.

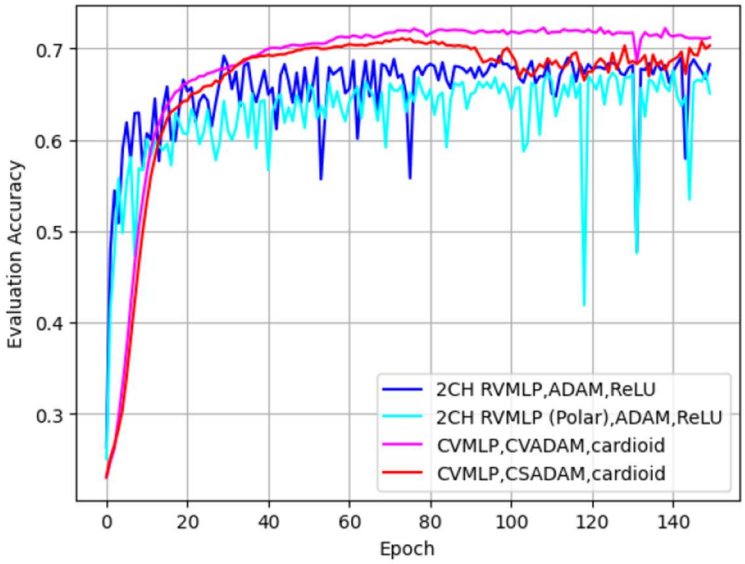
CVMLP_0 shows strong classification performance for B0 with fewer false negatives compared to RVMLP_0 and RVMLP_1 as shown in Table 9 and show the highest accuracy among the model for B1. For B2 to B7, the performance varies, with some confusion. As shown in Table 10, B0 and B1 perform excellently, especially B0, which has the highest accuracy. B3 to B7 have reasonable true positive rates but show some confusion with other classes. However, CVMLP_1 has a poor performance for B2.



(a) Relation between training loss and number of training epoch



(b) Relation between training accuracy and number of training epoch



(c) Relation between evaluation accuracy and number of training epoch

Fig. 11. Relation between three performance indicators and number of training epochs.

Table 7. Confusion Matrix for RVMLP_0

Actual/Predicted	B0	B1	B2	B3	B4	B5	B6	B7
B0	**306**	23	0	0	0	0	163	2
B1	15	**820**	0	2	0	1	14	17
B2	1	0	**79**	105	6	109	10	80
B3	0	0	72	**215**	0	91	10	144
B4	1	0	1	0	**352**	3	1	2
B5	1	0	59	89	6	**307**	4	44
B6	54	0	0	0	0	0	**511**	22
B7	6	0	14	100	1	6	56	**278**

Table 8. Confusion Matrix for RVMLP_1

Actual/Predicted	B0	B1	B2	B3	B4	B5	B6	B7
B0	**298**	36	0	13	0	1	135	11
B1	44	**813**	0	1	0	3	8	0
B2	13	1	**0**	218	0	131	5	22
B3	30	0	0	**376**	0	41	24	61
B4	2	0	0	3	**347**	5	3	0
B5	7	2	0	101	0	**387**	5	8
B6	121	3	0	2	0	0	**449**	12
B7	53	3	0	272	0	28	39	**66**

Table 9. Confusion Matrix for CVMLP_0

Actual/Predicted	B0	B1	B2	B3	B4	B5	B6	B7
B0	**363**	22	0	0	0	0	103	6
B1	26	**842**	0	0	0	0	1	0
B2	0	0	**28**	230	0	63	5	64
B3	0	5	10	**356**	0	28	4	129
B4	0	0	6	3	**347**	1	1	2
B5	0	3	14	222	0	**232**	4	35
B6	24	2	0	1	0	0	**523**	37
B7	4	2	0	107	0	1	46	**301**

Table 10. Confusion Matrix for CVMLP_1

Actual/Predicted	B0	B1	B2	B3	B4	B5	B6	B7
B0	**393**	19	0	2	0	0	70	10
B1	37	**830**	0	1	0	0	1	0
B2	1	0	**37**	200	5	105	3	39
B3	0	2	28	**334**	0	69	3	96
B4	1	0	1	2	**352**	314	3	20
B5	0	1	40	132	0	**314**	3	20
B6	82	1	0	2	0	0	**435**	67
B7	13	1	4	149	0	4	30	**260**

5 Conclusion and Future Works

The conclusion section of the article on complex-valued deep learning for microbial organism classification highlights the nuanced trade-offs encountered in the study between efficiency, accuracy, and speed. The experiments demonstrated that complex-valued convolutional neural networks (CVCNN) offer superior efficiency in nucleotide encoding and overall processing time compared to their real-valued counterparts. However, multilayer perceptron (MLP) models in certain contexts showed quicker training times, suggesting that for some applications, real-valued models might still offer advantages. Despite a slight increase in parameter count, complex-valued MLPs (CVMLP) showed commendable performance, especially in terms of validation accuracy, indicating better generalization to new data. The study suggests that the choice between real-valued and complex-valued models should be tailored to the specific requirements of genome sequencing classification projects, considering the trade-offs between model complexity, computational efficiency, and accuracy. The article underscores the potential of complex-valued deep learning in enhancing the accuracy and efficiency of microbial classification, contributing significantly to the field of bioinformatics and computational genomics.

References

[1] Colston, S.M., Fullmer, M.S., Beka, L., Lamy, B., Gogarten, J.P., Graf, J.: Bioinformatic genome comparisons for taxonomic and phylogenetic assignments using aeromonas as a test case. MBio 5(6), 10–1128 (2014)

[2] Kim, P.Y., Kim, A.Y., Newman, J.J., Cella, E., Bishop, T.C., Huwe, P.J., Uchakina, O.N., McKallip, R.J., Mack, V.L., Hill, M.P., et al.: A collaborative approach to improving representation in viral genomic surveillance. PLOS Global Public Health 3(7), e0001935 (2023)

[3] Da Silva Cândido, D., Pybus, O.G., Faria, N.R.: A38 genomic epidemiology quantifies gaps in aedes-borne virus transmission in the americas. Virus Evolution 5(Supplement 1), vez002–037 (2019)

[4] Metzker, M.L.: Sequencing technologies—the next generation. Nature reviews genetics 11(1), 31–46 (2010)

[5] Van El, C.G., Cornel, M.C., Borry, P., Hastings, R.J., Fellmann, F., Hodgson, S.V., Howard, H.C., Cambon-Thomsen, A., Knoppers, B.M., Meijers-Heijboer, H., et al.: Whole-genome sequencing in health care. European Journal of Human Genetics 21(6), 580–584 (2013)

[6] Török, M., Peacock, S.: Microbial whole-genome sequencing: Applications in clinical microbiology and public health. Molecular microbiology: diagnostic principles and practice pp. 32–48 (2016)

[7] Edwards, D., Batley, J.: Plant genome sequencing: applications for crop improvement. Plant biotechnology journal 8(1), 2–9 (2010)

[8] Bertelli, C., Greub, G.: Rapid bacterial genome sequencing: methods and applications in clinical microbiology. Clinical Microbiology and Infection 19(9), 803–813 (2013)

[9] Basho, R.K., Eterovic, A.K., et al.: Clinical applications and limitations of next-generation sequencing. American Journal of Hematology/Oncology® 11(3) (2015)

[10] Qu, Q., Xu, J., Kang, W., Feng, R., Hu, X.: Ensemble learning model identifies adaptation classification and turning points of river microbial communities in response to heatwaves. Global Change Biology 29(11), e5cafbfe385b3907c1af50158a84768cd46db9bf (2023). https://doi.org/10.1111/gcb.16985

[11] Choudhury, N., Sahu, T., Rao, A., Rout, A.K., Behera, B.: An improved machine learning-based approach to assess the microbial diversity in major north indian river ecosystems. Genes 14(5), 1082 (2023). https://doi.org/10.3390/genes14051082

[12] Alharbi, F., Vakanski, A.: Machine learning methods for cancer classification using gene expression data: A review. Bioengineering 10(2), 173 (2023). https://doi.org/10.3390/bioengineering10020173

[13] Ahmad, F., Khan, M.U.G., Tahir, A., Tipu, M.Y., Rabbani, M., Shabbir, M.: Two phase feature-ranking for new soil dataset for coxiella burnetii persistence and classification using machine learning models. Scientific Reports 12(1), 1–12 (2023). https://doi.org/10.1038/s41598-022-26956-8

[14] Virtue, P.M.: Complex-valued deep learning with applications to magnetic resonance image synthesis. University of California, Berkeley (2019)

[15] Peker, M.: An efficient sleep scoring system based on eeg signal using complex-valued machine learning algorithms. Neurocomputing 207, 165–177 (2016)

[16] Dramsch, J.S., Lüthje, M., Christensen, A.N.: Complex-valued neural networks for machine learning on non-stationary physical data. Computers & Geosciences 146, 104643 (2021)

[17] Wang, Y., Gui, G., Gacanin, H., Ohtsuki, T., Dobre, O.A., Poor, H.V.: An efficient specific emitter identification method based on complex-valued neural networks and network compression. IEEE Journal on Selected Areas in Communications 39(8), 2305–2317 (2021)

[18] Zhang, H., Gu, M., Jiang, X., Thompson, J., Cai, H., Paesani, S., Santagati, R., Laing, A., Zhang, Y., Yung, M., et al.: An optical neural chip for implementing complex-valued neural network. Nature communications 12(1), 457 (2021)

[19] Anderson, E.: The species problem in iris. Annals of the Missouri Botanical Garden 23(3), 457–509 (1936)

[20] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)

[21] Peker, M.: An efficient sleep scoring system based on eeg signal using complex-valued machine learning algorithms. Neurocomputing 207, 165–177 (2016)

[22] Savitha, R., Suresh, S., Sundararajan, N.: Fast learning circular complex-valued extreme learning machine (cc-elm) for real-valued classification problems. Information sciences 187, 277–290 (2012)

[23] Blake, C.L.: Uci repository of machine learning databases. http://www. ics. uci. edu/˜ mlearn/MLRepository. html (1998)

[24] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)

[25] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)

[26] Qu, Y. D., Zhang, R. Q., Shen, S. Q., Yu, J., & Li, M. (2023). Entanglement Detection with Complex-Valued Neural Networks. International Journal of Theoretical Physics, 62(9), 206.

[27] Ren, Y., Jiang, W., & Liu, Y. (2023). A New Architecture of a Complex-Valued Convolutional Neural Network for PolSAR Image Classification. Remote Sensing, 15(19), 4801.

[28] Nielsen, M.A.: Neural networks and deep learning, vol. 25. Determination press San Francisco, CA, USA (2015)

[29] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)

[30] Bishop, C.M., Nasrabadi, N.M.: Pattern recognition and machine learning, vol. 4. Springer (2006)

[31] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by backpropagating errors. nature 323(6088), 533–536 (1986)

[32] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[33] Smith, J. W. (2023). Complex-valued neural networks for data-driven signal processing and signal understanding. arXiv preprint arXiv:2309.07948.

[34] Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., & Ovarlez, J. P. (2023). Theory and Implementation of Complex-Valued Neural Networks. arXiv preprint arXiv:2302.08286.

[35] Anderson, E.: The species problem in iris. Annals of the Missouri Botanical Garden 23(3), 457–509 (1936)

[36] Basho, R.K., Eterovic, A.K., et al.: Clinical applications and limitations of nextgeneration sequencing. American Journal of Hematology/Oncology® 11(3) (2015)

[37] Bertelli, C., Greub, G.: Rapid bacterial genome sequencing: methods and applications in clinical microbiology. Clinical Microbiology and Infection 19(9), 803–813 (2013)

[38] Bishop, C.M., Nasrabadi, N.M.: Pattern recognition and machine learning, vol. 4. Springer (2006)

[39] Câmara, G. B., Coutinho, M. G., Silva, L. M. D., Gadelha, W. V. D. N., Torquato, M. F., Barbosa, R. D. M., & Fernandes, M. A. (2022). Convolutional Neural Network Applied to SARS-CoV-2 Sequence Classification. Sensors, 22(15), 5730.

[40] He, W., Jin, Y., Zhu, H., Zheng, Y., Qian, J.: Effect of chlamydia trachomatis on adverse pregnancy outcomes: a meta-analysis. Archives of Gynecology and Obstetrics 302, 553–567 (2020)

[41] Möller, J., Kraner, M.E., Burkovski, A.: More than a toxin: Protein inventory of clostridium tetani toxoid vaccines. Proteomes 7(2), 15 (2019)

[42] Bonten, M., Johnson, J.R., van den Biggelaar, A.H., Georgalis, L., Geurtsen, J., de Palacios, P.I., Gravenstein, S., Verstraeten, T., Hermans, P., Poolman, J.T.: Epidemiology of escherichia coli bacteremia: a systematic literature review. Clinical Infectious Diseases 72(7), 1211–1219 (2021)

[43] Mohtashami, M., Mohamadi, M., Azimi-Nezhad, M., Saeidi, J., Nia, F.F., Ghasemi, A.: Lactobacillus bulgaricus and lactobacillus plantarum improve diabetic wound healing through modulating inflammatory factors. Biotechnology and applied biochemistry 68(6), 1421–1431 (2021)

[44] Moule, M.G., Cirillo, J.D.: Mycobacterium tuberculosis dissemination plays a critical role in pathogenesis. Frontiers in cellular and infection microbiology 10, 65 (2020)

[45] Jajere, S.M.: A review of salmonella enterica with particular focus on the pathogenicity and virulence factors, host specificity and antimicrobial resistance including multidrug resistance. Veterinary world 12(4), 504 (2019)

[46] Uruén, C., García, C., Fraile, L., Tommassen, J., Arenas, J.: How streptococcus suis escapes antibiotic treatments. Veterinary Research 53(1), 1–33 (2022)

[47] Ramamurthy, T., Mutreja, A., Weill, F.X., Das, B., Ghosh, A., Nair, G.B.: Revisiting the global epidemiology of cholera in conjunction with the genomics of vibrio cholerae. Frontiers in public health 7, 203 (2019)