COMPUTATIONAL ANDSTRUCTURAL BIOTECHNOLOGY JOURNAL

journal homepage: www.elsevier.com/locate/csbj

SARS-CoV-2 virus classification based on stacked sparse autoencoder

Maria G.F. Coutinho [a], Gabriel B.M. Câmara [a], Raquel de M. Barbosa [b], Marcelo A.C. Fernandes [a,c,*]

[a] Laboratory of Machine Learning and Intelligent Instrumentation, IMD/nPITI, Federal University of Rio Grande do Norte, Natal, Brazil
[b] Department of Pharmacy and Pharmaceutical Technology, University of Granada, 18071 Granada, Spain
[c] Department of Computer and Automation Engineering, Federal University of Rio Grande do Norte, Natal, Brazil

ARTICLE INFO

Article history:
Received 21 June 2022
Received in revised form 4 December 2022
Accepted 5 December 2022
Available online 9 December 2022

Keywords:
COVID-19
Deep learning
SARS-CoV-2
Sparse autoencoder
Viral classification

ABSTRACT

Since December 2019, the world has been intensely affected by the COVID-19 pandemic, caused by the SARS-CoV-2. In the case of a novel virus identification, the early elucidation of taxonomic classification and origin of the virus genomic sequence is essential for strategic planning, containment, and treatments. Deep learning techniques have been successfully used in many viral classification problems associated with viral infection diagnosis, metagenomics, phylogenetics, and analysis. Considering that motivation, the authors proposed an efficient viral genome classifier for the SARS-CoV-2 using the deep neural network based on the stacked sparse autoencoder (SSAE). For the best performance of the model, we explored the utilization of image representations of the complete genome sequences as the SSAE input to provide a classification of the SARS-CoV-2. For that, a dataset based on k-mers image representation was applied. We performed four experiments to provide different levels of taxonomic classification of the SARS-CoV-2. The SSAE technique provided great performance results in all experiments, achieving classification accuracy between 92% and 100% for the validation set and between 98.9% and 100% when the SARS-CoV-2 samples were applied for the test set. In this work, samples of the SARS-CoV-2 were not used during the training process, only during subsequent tests, in which the model was able to infer the correct classification of the samples in the vast majority of cases. This indicates that our model can be adapted to classify other emerging viruses. Finally, the results indicated the applicability of this deep learning technique in genome classification problems.

1. Introduction

Since the emergence of the SARS-CoV-2 virus at the end of 2019, many works are been developed aiming to provide more comprehension about this novel virus. In March 2020, the World Health Organization (WHO) raised the level of contamination to the COVID-19 pandemic, due to its geographical spread across several countries. On July 9, 2021, the disease had registered more than 185 million confirmed cases, and more than 4 million confirmed deaths. In the case of a novel virus identification, the early elucidation of taxonomic classification and origin of the virus genomic sequence is essential for strategic planning, containment, and treatments of the disease [1–3].

One of the research field in the bioinformatics area is the analysis of genomic sequences. In the last years, many strategies based on alignment-free methods have been explored as an alternative for the alignment-based methods, considering the limitations of the second approach. Alignment-based programs assume that homologous sequences comprise a series of linearly arranged and more or less conserved sequence stretches, which is not always the case in the real world [4].

Among the alignment-free methodologies, there are some models based on deep learning (DL) techniques, that can provide significant performance in applications of genome analysis [5–7]. Deep neural networks (DNN) can improve prediction accuracy by discovering relevant features of high complexity [7].

Fig. 1 presents the genome analysis stages and how deep learning integrates this process. The genome analysis stages include the primary analysis, the secondary analysis, and the tertiary analysis. The primary and secondary analysis compose the genome sequencing. The primary analysis receives the biological sample and generates genomic data information, called “reads”, after the

* Corresponding author at: Department of Computer and Automation Engineering, Federal University of Rio Grande do Norte, Natal, Brazil.

E-mail addresses: rbarbosa@ugr.es (Raquel de M. Barbosa), mfernandes@dca.ufrn.br (

https://doi.org/10.1016/j.csbj.2022.12.007

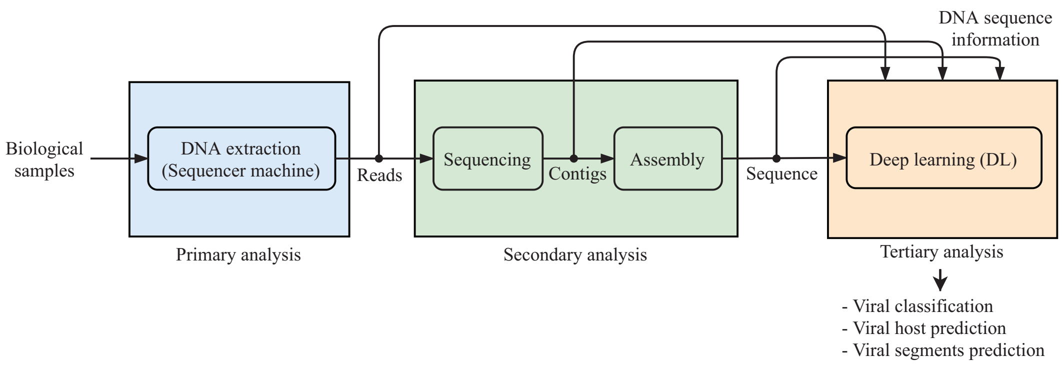


Fig. 1. Genome analysis stages with deep learning.

processing by the sequencer machine. Then, the secondary analysis processes the reads and produces the complete genome sequence. Lastly, the tertiary analysis provides the genome interpretation, which can be performed for many algorithms and techniques [8–10], as machine learning algorithms [11] and deep learning techniques [7]. The deep learning techniques have been successful used for the tertiary analysis in many viral classification problems associated with the diagnosis of viral infections, metagenomics, pharmacogenomics, and others [12–16].

Fig. 2 shows the steps of the tertiary analysis using DL, that are the mapping and processing stages. The mapping stage receives the DNA sequence information, that can be the reads, contigs, or

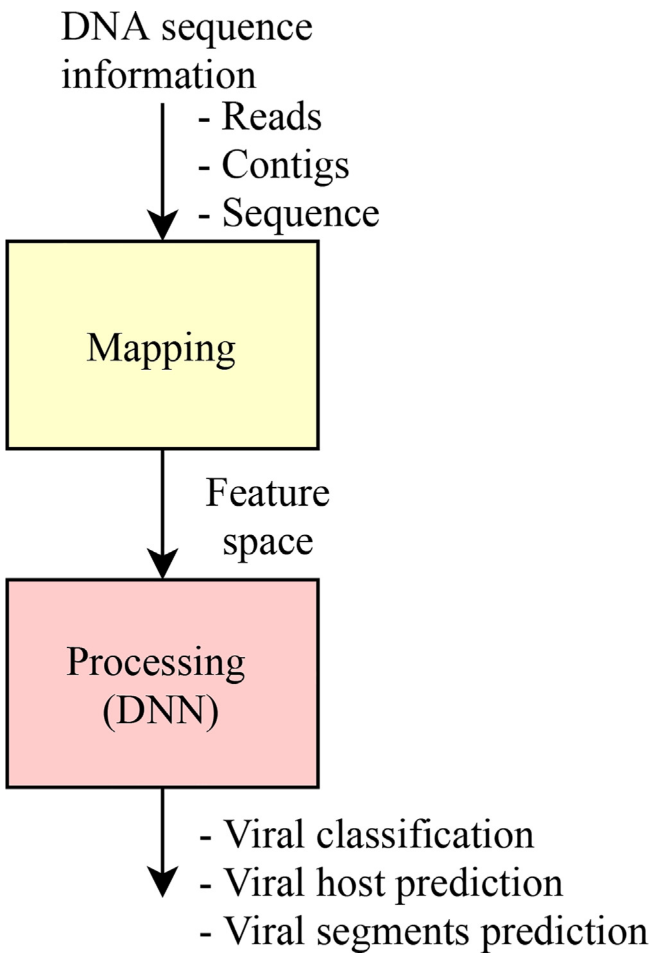


Fig. 2. Stages of viral genome analysis using deep learning.

the whole genome sequence, and maps this data into a feature space. Various mapping strategies have been present in the works from state of the art, such as one-hot encoding [17–19,14], number representation [12,13], digital signal processing [20], and other strategies, including multiple mapping strategies applied sequentially [21,22]. The processing stage consists of the utilization of a DNN to perform classification, prediction, and other assumptions about the genome information.

The mapping stage is crucial for the performance of the processing stage. The genome sequence length varies by the type of virus. Since the DNN only receive a fixed-size input, some researchers have not been using the whole or long sequence length. Nevertheless, longer sequences contain more information and thus are more convenient to make predictions [18].

The main contributions of this work are:

- To provide an efficient viral genome classifier for the SARS-CoV-2 virus, based on the stacked sparse autoencoder (SSAE) technique.
- To explore the utilization of a dataset based on *k*-mers image representation of the complete genome sequences as the SSAE input.
- To provide different levels of taxonomic classification.
- To deliver an approach that can be adapted to classify other emerging viruses.

The present paper is organized as follows: This first section presents a general introduction, exposing the motivations and contributions of the work. Section 2 discusses some related works from state of the art. Section 3 presents the materials and methods used to perform the experiments. Section 4 will present the results of each experiment, a discussion of the results, and a comparison with a work from the state of the art. Finally, Section 5 will present the final considerations regarding the obtained results, the implications of the work and our plans for the future.

2. Related works

Many works from the state of the art are using deep learning to solve biomedical problems [23–27]. Recently works in literature have been applying deep learning as tertiary analysis such as viral prediction, viral host prediction, and viral segments prediction [17,12,18,20,13,28,19,29,14,30–32,15,16,33–36].

The work from [37] uses a deep learning approach combining a CNN with a Bi-directional LSTM (BLSTM) to classify the SARS-CoV-2 among Coronavirus and detect sequences with regulatory or transcription motifs. For the DNN input, they used the one-hot vectors to represent DNA sequences as 2D matrices.

Table 1
State of the art references – Part 1.

Biology name	Group		Aim	Ref.	COVID-19	DNN
Genome prediction or sequence classification	Genome classification (taxonomic classification)	Viral classification	Viral Subtyping	[12]	–	CNN
			Primer design	[13]	Yes	CNN
			Identified virus sequence [14]	Yes	LSTM	
						CNN + FC
			Taxonomic classification	[15]	–	BLSTM
				[16]	–	CNN
	Genome prediction	Viral prediction	Identified virus sequence	[18]	–	CNN
				[17]	–	CNN
			Identified phage, chromossomes, plasmid	[29]	–	CNN
Host prediction	Host classification	Viral host classification	Predicting viruses among several hosts	[19]	–	BLSTM + CNN CNN
	Host prediction	Viral host prediction		[28]	Yes	CNN
Genome segments prediction	Genome segments classification	Viral segments classification	Prediction specific regions in the genome	[20]	–	CNN + FC
				[30]	–	CNN + BLSTM
				[31]	–	CNN + BLSTM
				[32]	–	CNN + BLSTM

Table 2
State of the art references – Part 2.

Input	**Output**	**Ref.**	**Biology fields**	**Bioinformatics**
The DNA or cDNA (RNA virus) of the virus. The whole or part of the genome is used.	Number of the classes	[12]	Metagenomics Diagnosis of viral infections Pharmacogenomics	Free alignments techniques
		[13]		
		[14]		
		[15]		
		[16]		
	Score	[18]	Metagenomics Phylogenetic analysis	
	Binary output	[17]		Score [29]
	Number of the classes	[19]	Metagenomics Phylogenetic analysis	
	Score	[28]	Metagenomics	
	Number of the classes	[20]	Transcriptome Analysis	
		[30]		
		[31]	Gene expression analysis	
		[32]		

Tables 1 and 2 present some works from the state of the art that applied DNNs in order to analyse viral genome sequences. Table 1 details the focus of each work as the biology name, the group, the aim, indicates if the proposal was or was not applied for the COVID-19 and present the DNN used. The DNNs applied in those references are divide into 5 groups (CNN + FC, LSTM + FC, BLSTM + FC, BLSTM + CNN + FC, CNN + BLSTM + FC), as we show in the last column of Table 1. Table 2 shows the details about the input and the output of the DNN, besides the biology fields and the bioinformatics area.

In the work presented in [12] was proposed a viral genome deep classifier (VGDC), the first viral genome subtyping based on deep learning techniques found in the literature. Their approach uses a Convolutional Neural Network (CNN) with 25 layers to classify several groups of viruses in subtypes. For the tests, were used five different datasets, each one containing genomes sequences of a specific type of virus. The whole virus genome sequence was used as the input to the network, where the corresponding ASCII code represented each nucleotide. The results indicated that the VGDC was able to achieve better results in comparison with previous works from the state of the art.

In [13] was proposed an approach to assist the tests in the detection of SARS-CoV-2, based on the use of DL techniques. For this, a CNN architecture with 4 layers was used to extract characteristics of the virus genomes, as well as to classify SARS-CoV-2 among Coronavirus type viruses. As presented in [12], the CNN received as input the whole virus genome sequences. The nucleotides were mapped in numerical values (C = 0.25, T = 0.50, G = 0.75, A = 1.0). Missing entries received a value of 0.0. The experiments showed that the CNN was able to correctly identify the sequences even in cases where the noise was added to the genome, reaching accuracies between 0.9674 (with noise) and 0.9875 (without noise). Through the results, the authors also identified a sequence as exclusive for the SARS-CoV-2 virus. They proposed the use of this sequence as a primer for PCR tests.

In [14], was proposed an approach to provide viral classification using the contigs (fragments of the genome sequence) and two different reverse-complement (RC) neural networks architectures: a RC-CNN and a RC-LSTM. These models were also applied to the SARS-CoV-2 virus.

In works presented in [15,16], a taxonomic classification for metagenomics applications is proposed. Both works used segments of genome (reads) with DL input (see Fig. 1), and the output is the number of the classes. In [15], it was proposed two DL models, one to classify species, and another to classify genus. In [16], a hierarchical taxonomic classification for viral metagenomic data via DL, called CHEER, was proposed. Similar to the work proposed in [15], the CHEER framework classifies the order, family, and genus.

Proposals presented in [17,18,29] used the contigs with DL input for viral prediction, and classification. In [17,18] a DL virus identification framework was proposed and both cases try to recognize if the input is a virus or not.

In work from [17], called ViraMiner, was proposed and approach to detect the presence of viruses on raw metagenomic contigs from different human samples. They used a CNN architecture with two different convolutional branches (pattern and frequency branch) in order to extract relevant features. The outputs of these branches are concatenated and inserted into the fully connected (FC) layer. The ViraMiner output produces a single value that indicates the likelihood of the sequence belonging to the virus class.

In the proposal presented in [18], called DeepVirFinder, the output is a score between 0 and 1 for a binary classification between virus and prokaryote. They fragmented the genomes into non-overlapping sequences of different sizes ($150, 300, 500, 1000$, and 3000 bp). The sequences were mapped for the network input using the one-hot encoding method. Since they increase the length of the input, i.e. the sequence fragment, they achieve better performance results, which was measured by the area under the receiver operating characteristic curve (AUROC). The maximum AUROC achieved was 0.98 for the 3000 bp fragment.

The work presented in [29] identifies metagenomic fragments as phages, chromosomes or plasmids using the CNN technique. The experiments were performed using artificial contigs and real metagenomic data. The network output, provided by a softmax layer, consists of 3 scores that indicate the probability that each fragment belongs to a specific class.

In the works from [28,19] are present DL architectures for host prediction and classification. [28] used a CNN to provide host and infectivity prediction of SARS-CoV-2 virus. In [19] was proposed an approach to predict viral host from three different virus species (influenza A virus, rabies lyssavirus and rotavirus A) from the whole or only fractions of a given viral genome.

In the works from [20,30–32] were proposed methodologies to predict or classify specific regions in the genome sequence. [20] presented a methodology for the classification of three different functional genome types: coding regions, long noncoding regions, and pseudogenes in genomic data. They used a digital signal processing (DSP) methods, called Genomic signal processing (GSP), that converts the nucleotide sequence into a graphical representation of the information contained in the sequence. A CNN with 19 layers was used to perform the classification results.

The authors in [30] proposed a DL framework to identify similar patterns in DNA N6-methyladenine (6 mA) sites prediction. This framework, called Deep6mA, is composed of a CNN to extract high-level features in the sequence and a Bi-directional LSTM (BLSTM) to learn dependence structure along the sequence, besides a fully connected layer that determines whether the site is a 6 mA site.

In [31] was provided a method based on CNN and BLSTM for exploring the RNA recognition patterns of the CCCTC-binding factor (CTCF) and identify candidate IncRNAs binding. The experiments conducted with two different datasets (human U2OS and mouse ESC) were able to predict CTCF-binding RNA sites from nucleotide sequences. Moreover, [32] propose a computational prediction approach for DNA–protein binding based on CNN and BLSTM.

Considering the importance of providing viral classification and the advantages of the use of DL techniques in several applications, especially for many viral classification problems, as presented previously, the main objective of this work is to generate an efficient viral genome classifier for the SARS-CoV-2 virus using the DNN based on the stacked sparse autoencoder (SSAE) technique. The SSAE has been successfully applied in many biomedical works from the state of the art [38–40,6].

Unlike most of the related works presented previously, this work intends to provide viral classification using the whole genome sequences, as presented in [12,13]. However, in [12,13] were used the length of the longest genome sequence of the dataset as the input of the DNN. So, it was necessary to add some padding for the missing entries. In this work, we will explore the utilization of k-mers image representation of the complete genome sequences as the DNN input, which will feasibly the use of genome sequences of any length and enable the use of smaller network inputs. The k-mers representation was used in many works that provide genome sequence classification, as presented in [41], which explores the spectral sequence representation based on k-mers occurrences. However, that work doesn't explore the k-mers image representation. So, our work's novelty consists in exploring the utilization of image representations of the complete genome sequences for the processing in the SSAE to provide an accurate viral classification of the SARS-CoV-2.

We performed some experiments to provide various levels of taxonomic classification of the SARS-CoV-2 virus, similar to the proposed experiments in [11], using the SSAE technique with a dataset of k-mers images representations, available on [42].

3. Materials and methods

This section will explain the dataset used in this work, describe the equations and other details about the k-mers image representation applied, and explain how the data were partitioned for the experiments. Besides, the DNN Architecture will be presented, detailing the number of layers and neurons applied and the platform used to implement the technique.

3.1. Dataset

For the experiments, we used a k-mers representation dataset of SARS-CoV-2 genome, available on [42]. This dataset is composed of $1,557$ virus instances of SARS-CoV-2, as also, a data stream of $11,540$ viruses from the Virus-Host DB dataset and the other three Riboviria viruses from NCBI (Betacoronavirus RaTG13, bat-SL-CoVZC45, and bat-SL-CoVZXC21). It also provides k-mers image representation of all data. The k-mers images were used to perform the experiments for this work. Assuming the dataset with D sequences (in this work $D = 1,557 + 11,540 + 3 = 13,100$ sequemces), each d-th sequence, stored in dataset, is expressed by

$$\mathbf{s}_d = \left[s_{d,1}, \ldots, s_{d,n}, \ldots, s_{d,N_d}\right] \tag{1}$$

where N_d is the length of d-th sequence and $s_{d,n}$ is the n-th nucleotide of the d-th sequence. Each n-th $s_{d,n}$ can be characterized as a symbol belonging to an alphabet of 4 possible symbols expressed by set $\{\mathrm{A,T,C,G}\}$ for DNA or by set $\{\mathrm{A,U,C,G}\}$ for RNA, that is,

$$s_{d,n} \in (\{\mathrm{A,T,C,G}\} \vee \{\mathrm{A,U,C,G}\}). \tag{2}$$

In k-mers representation, each d-th nucleotide sequence, $\mathbf{s}_d$, is grouped in k-mers sub-sequences [43,44] that can be expressed as

$$\mathbf{H}_d = \begin{bmatrix} \mathbf{h}_{d,1} \\ \mathbf{h}_{d,2} \\ \vdots \\ \mathbf{h}_{d,i} \\ \vdots \\ \mathbf{h}_{d,N_d-k} \\ \mathbf{h}_{d,N_d-k+1} \end{bmatrix} = \begin{bmatrix} s_{d,1} & \cdots & s_{d,k} \\ s_{d,2} & \cdots & s_{d,k+1} \\ \vdots & \ddots & \vdots \\ s_{d,i} & \cdots & s_{d,i+k} \\ \vdots & \ddots & \vdots \\ s_{d,N_d-k} & \cdots & s_{d,N_d-1} \\ s_{d,N_d-k+1} & \cdots & s_{d,N_d} \end{bmatrix} \tag{3}$$

where the matrix $\mathbf{H}_d$ stores the k-mers associated with each d-th sequence $\mathbf{s}_d$. The k-mers representations are based in each d-th matrix $\mathbf{H}_d$ and the matrix $\mathbf{\Gamma}$, call here as symbol matrix. The symbol matrix is expressed as

$$\mathbf{\Gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1 \\ \vdots \\ \boldsymbol{\gamma}_i \\ \vdots \\ \boldsymbol{\gamma}_M \end{bmatrix} = \begin{bmatrix} \gamma_{1,1} & \cdots & \gamma_{1,k} \\ \vdots & \ddots & \vdots \\ \gamma_{i,1} & \cdots & \gamma_{i,k} \\ \vdots & \ddots & \vdots \\ \gamma_{M,1} & \cdots & \gamma_{M,k} \end{bmatrix} \tag{4}$$

where each element $\gamma_{i,j} \in (\{A, T, C, G\} \vee \{A, U, C, G\})$. The symbol matrix, $\mathbf{\Gamma}$, stores all M possibilities of the k-mers, where

$$M = 4^k. \tag{5}$$

The k-mers count 1D representation can be expressed as

$$\mathbf{c}_d = [c_{d,1}, \ldots, c_{d,i}, \ldots, c_{d,M}] \tag{6}$$

where

$$c_{d,i} = \sum_{v=1}^{N_d-k+1} B_{d,i,v} \tag{7}$$

and

$$B_{d,i,v} = \begin{cases} 0 & \text{for} \boldsymbol{\gamma}_i \neq \mathbf{h}_{d,v} (\exists u = 1, \ldots, k : \gamma_{i,u} \neq s_{d,v+u-1}) \\ 1 & \text{for} \boldsymbol{\gamma}_i = \mathbf{h}_{d,v} (\forall u = 1, \ldots, k : \gamma_{i,u} = s_{d,v+u-1}) \end{cases} \tag{8}$$

So, the i-th $c_{d,i}$ indicates the number of occurrences of each d-th sub-sequence stored on the $\mathbf{\Gamma}$ matrix.

Table 3 shows a example of the k-mers count 1D representation values (with $k = 2$) for SARS-CoV-2 from China-Wuhan (ID: LR757995), USA-MA (ID: MT039888), Brazil (ID: MT126808), and Italy (ID: MT066156). The dataset provide in [42] has k-mers count 1D representation for $k = 2, \ldots, 6$.4.

The k-mers count 2D representation for each d-th sequence, $\mathbf{s}_d$, is described by

$$\mathbf{\Lambda}_d = \begin{bmatrix} \lambda_{d,1,1} & \cdots & \lambda_{d,1,L} \\ \vdots & \ddots & \vdots \\ \lambda_{d,i,1} & \cdots & \lambda_{d,i,L} \\ \vdots & \ddots & \vdots \\ \lambda_{d,L,1} & \cdots & \lambda_{d,L,L} \end{bmatrix} = \begin{bmatrix} c_{d,1} & \cdots & c_{d,L} \\ \vdots & \ddots & \vdots \\ c_{d,(i-1)\times L+1} & \cdots & c_{d,i\times L} \\ \vdots & \ddots & \vdots \\ c_{d,M-L+1} & \cdots & c_{d,M} \end{bmatrix} \tag{9}$$

where

$$L = \sqrt{M} = 2^k. \tag{10}$$

Finally, the k-mers image representation, for each d-th sequence, can be represented as

$$\mathbf{\Phi}_d = \begin{bmatrix} \phi_{d,1,1} & \cdots & \phi_{d,1,L} \\ \vdots & \ddots & \vdots \\ \phi_{d,i,1} & \cdots & \phi_{d,i,L} \\ \vdots & \ddots & \vdots \\ \phi_{d,L,1} & \cdots & \phi_{d,L,L} \end{bmatrix} \tag{11}$$

where $\phi_{d,i,j}$ represents each pixel associated with d-th image $\mathbf{\Phi}_d$. Each pixel, $\phi_{d,i,j}$, is be expressed as

$$\phi_{d,i,j} = \left\lfloor \frac{2^b - 1}{\max\{\mathbf{\Lambda}_d\}} \times \lambda_{d,i,j} \right\rfloor \tag{12}$$

where $\max\{\cdot\}$ is the maximum value in d-th matrix $\mathbf{\Lambda}_d$, $\lfloor \cdot \rfloor$ is the greatest integer less than or equal, and b is number of bits associated with the image pixels. Fig. 3 show the k-mers image representation, matrix $\mathbf{\Phi}$, (with $k = 6$ and $b = 8$) for Geminiviridae (ID: HE616777), Alphacoronavirus (ID: JQ410000), and SARS-CoV-2 (Betacoronavirus) from China-Wuhan (ID: LR757995) and Brazil (ID: MT126808).

In this work, we used k-mers image representation with $k = 6$. That choice of k value was based on the fact that when k is a small value, the existing property vectors for the k-mer may not contain enough genome information [45], however, when large k values are used, many k-mers do not appear in the sequence, which generates sparse feature vectors and causes the overfitting problem [46]. Besides, in the work presented in [17], the 6-mers reached the best performance in comparisons with other values of k $(3, 4, 5$ and $7)$.

The data of each experiment was partitioned using the holdout method, which splits the data into a training set and a validation set at random. We used the proportion of 80% for the training set and 20% for the validation set. Each class data was split respecting these percentages. The SARS-CoV-2 k-mers images were used only for the test set.

3.2. DNN architecture

All experiments were performed using the SSAE technique. In these models each hidden layer is composed of an individually

Table 3
Examples of k-mers count 1D representation values (with $k = 2$) for SARS-CoV-2.

k-mers ($k = 2$)	China-Wuhan (ID: LR757995)	USA-MA (ID: MT039888)	Brazil (ID: MT126808)	Italy (ID: MT066156)
AA	2862	2859	2853	2847
AC	2022	2022	2022	2022
AG	1741	1741	1742	1742
AT	2306	2309	2309	2308
CA	2085	2082	2084	2082
CC	886	888	888	888
CG	439	439	440	439
CT	2080	2081	2080	2082
GA	1612	1612	1612	1611
GC	1167	1167	1169	1168
GG	1092	1093	1092	1092
GT	1990	1990	1988	1989
TA	2373	2378	2377	2378
TC	1415	1412	1413	1413
TG	2589	2589	2587	2587
TT	3212	3217	3219	3216

Table 4
Examples of k-mers count 2D representation values (with $k = 2$) for SARS-CoV-2.

	China-Wuhan (ID: LR757995)		USA-MA (ID: MT039888)
$\mathbf{\Lambda}_{17}=$	$\begin{bmatrix} 2862 & 2022 & 1741 & 2306 \\ 2085 & 886 & 439 & 2080 \\ 1612 & 1167 & 1092 & 1990 \\ 2373 & 1415 & 2589 & 3212 \end{bmatrix}$	$\mathbf{\Lambda}_{32}=$	$\begin{bmatrix} 2859 & 2022 & 1741 & 2309 \\ 2082 & 888 & 439 & 2081 \\ 1612 & 1167 & 1093 & 1990 \\ 2378 & 1412 & 2589 & 3217 \end{bmatrix}$
	Brazil (ID: MT126808)		Italy (ID: MT066156)
$\mathbf{\Lambda}_{52}=$	$\begin{bmatrix} 2853 & 2022 & 1742 & 2309 \\ 2084 & 888 & 440 & 2080 \\ 1612 & 1169 & 1092 & 1988 \\ 2377 & 1413 & 2587 & 3219 \end{bmatrix}$	$\mathbf{\Lambda}_{79}=$	$\begin{bmatrix} 2853 & 2022 & 1742 & 2308 \\ 2084 & 888 & 439 & 2082 \\ 1612 & 1169 & 1092 & 1989 \\ 2377 & 1413 & 2587 & 3216 \end{bmatrix}$

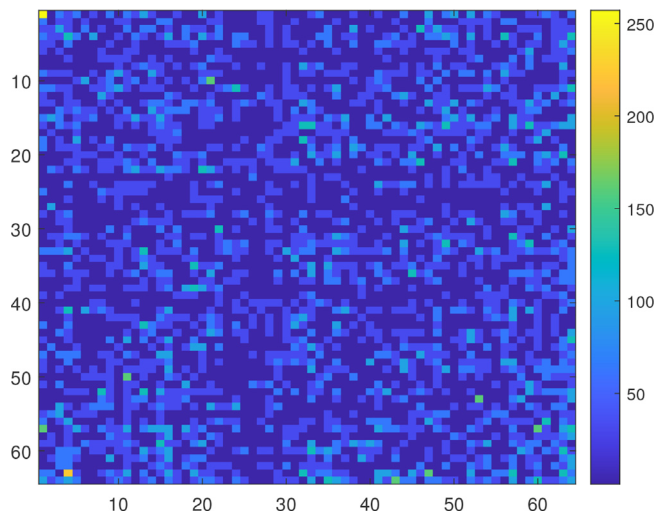


Fig. 3. Examples of k-mers images representation with $k = 6$. Based on Eq. 10, $L = 64$ and each image, matrix $\mathbf{\Phi}$ (see Eq. 11), is composed by 64×64 pixels with $b = 8$ (see Eq. 12).

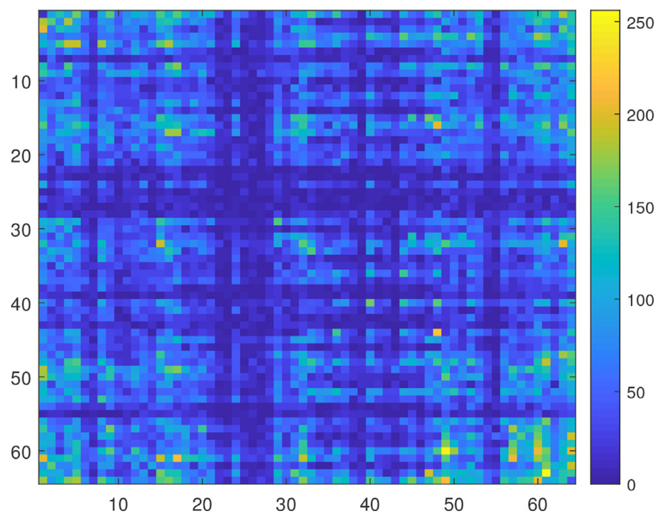


Fig. 3 (*continued*)

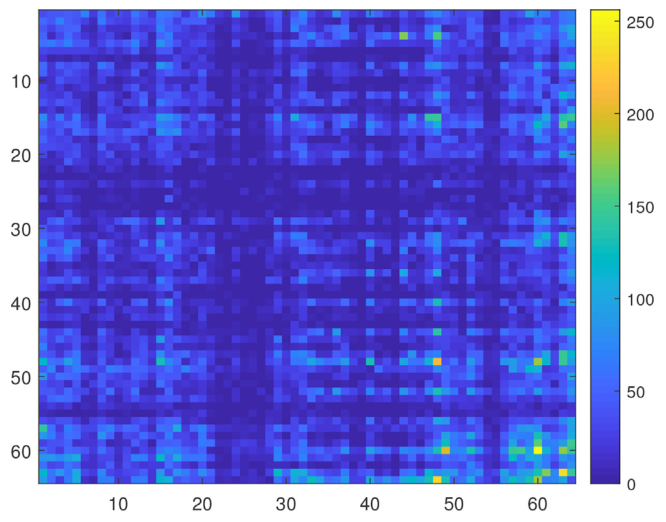


Fig. 3 (*continued*)

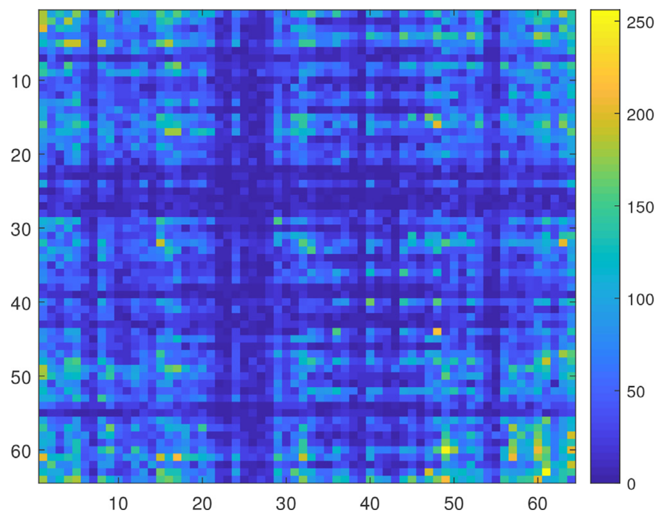


Fig. 3 (*continued*)

trained sparse autoencoder in an unsupervised way. A sparse autoencoder is an autoencoder whose training involves a sparse penalty, which functions as a regularizing term added to the loss function [47]. The autoencoder (AE) is a DL technique specialized in dimensionality reduction and feature extraction. The AE output can provide the reconstruction of the input information. These networks are composed of three layers: an input, a hidden and an output. The encoder is formed by the input and hidden layers, and the decoder is formed by the hidden and output layers [47]. For the output layer, we used a softmax layer, where the number of neurons consists of the number of classes of the experiment. Fig. 4 illustrates the DL SSAE with P inputs, K hidden layers, and a output layer. Each i-th hidden layer has Q_i neurons and the output layer has U neurons. Functions $\varphi(\cdot)$ and $f(\cdot)$ are the action functions in each p-th neuron (in each i-th hidden layer) and each u-th neuron in output layer, respectively.

For all experiments, the network architecture used three hidden layers ($K = 3$), containing 3000 neurons in the first hidden layer, Q_1, 1000 in the second hidden layer, Q_2, and 500 in the third hidden layer Q_3. For the softmax layer, the number of neurons corresponds to the number of classes of each experiment. So, the same model was used for all experiments, varying only the number of neurons of the output layer. For input of the SSAE, it was used k-mers images, with $k = 6$, generating images, matrix $\mathbf{\Phi}$, with 64×64 pixels (based on Eq. 10, $L = \sqrt{4^6} = 64$). Each d-th image, $\mathbf{\Phi}_d$, associated with a d-th viral genome sequence is reshaped into a vector expressed by

$$\mathbf{y}_d = \begin{bmatrix} y^0_{d,1} \\ y^0_{d,2} \\ \vdots \\ y^0_{d,i-1} \\ y^0_{d,i} \\ y^0_{d,i+1} \\ \vdots \\ y^0_{d,P-1} \\ y^0_{d,P} \end{bmatrix} = \begin{bmatrix} \phi_{d,1,1} \\ \vdots \\ \phi_{d,L,1} \\ \phi_{d,1,2} \\ \vdots \\ \phi_{d,L,2} \\ \vdots \\ \phi_{d,1,L} \\ \vdots \\ \phi_{d,L,L} \end{bmatrix} \tag{13}$$

with $P = 64 \times 64 = 4096$ values and applied to the SSAE. The number of neurons in output layer, U, is defined by the number of different viruses in a specific taxonomic level such as family, genus, realm and other. The output can be expressed by

$$\mathbf{o} = \begin{bmatrix} o_1 \\ \vdots \\ o_u \\ \vdots \\ o_U \end{bmatrix} \tag{14}$$

where each u-th output, o_u, represents a specific virus in a taxonomic level classification and is defined by

$$o_u = \begin{cases} 1 & \text{if} \mathbf{y}_d \text{istheu} - \text{thvirus} \\ 0 & \text{otherwise} \end{cases}. \tag{15}$$

Fig. 5 illustrates how the sequence information is passed through the DL-SSAE to perform the viral classification. The DL-SSAE input was normalized in the range of 0 to 1. First, the SSAE receives the training set as input to perform the training phase. Then, the validation set, which only contains samples that were not applied in the training phase, is used to identify the capacity of generalization of the DNN. After the network validation, the SSAE was applied for the test set, which only contains SARS-CoV-2 sequences. The SARS-CoV-2 k-mers images were not used for the training phase of the SSAE.

The SSAE was implemented in the Matlab platform (License 596681), adopting the deep learning toolbox. All network was trained with the Scaled Conjugate Gradient (SCG) algorithm. The loss function used for the training in each AE was the Mean Squared Error with L2 and Sparsity Regularizers, that can be expressed as

$$E = \frac{1}{I}\sum_{i=1}^{I}\sum_{u=1}^{U}\left(o^{ref}_{ui} - o_{ui}\right)^2 + \lambda \times \Omega_{weights} + \beta \times \Omega_{sparsity}, \tag{16}$$

where I is the number of training examples, U is the number of classes, $\Omega_{weights}$ is the L2 regularization term, λ is the coefficient for the L2 regularization term, $\Omega_{sparsity}$ is the sparsity regularization term, and β is the coefficient for the sparsity regularization term.

The loss function applied for the softmax layer was the Cross-Entropy. After the training in each layer, the results for the SSAE can be improved with the fine-tuning process, which perform the backpropagation on the whole network, as a multilayer network. In that process, we fine tune the network, which adjust the weights, by retraining the network with the training data in a supervised way [48]. In this work, we applied that retrained process to improve the classification results. The fine-tuning process also used the Cross-Entropy as the loss function, as in the softmax layer.

4. Results and discussion

We performed four different experiments to provide different levels of taxonomic classification of the SARS-CoV-2 virus, similar to the experimental methodology present in [11]. The details about the data and the network architecture used in each experiment are shown in Table 5. The data of each experiment was split into 80% for the training set and 20% for the validation set. The SSAE architecture was chosen by the observation of the MSE obtained with the reconstruction of the validation set in each AE. In order to validate the proposed idea of this work, the results are present by the confusion matrix for the validation and test sets. We also measured the performance of the viral classifier proposed with some popular classification metrics, as precision, recall, F1-score, and specificity. The precision value measure the percentages of all the examples predicted to belong to each class that are correctly classified, which corresponds to the positive predictive value. The recall, also called sensibility, corresponds to the percentages of all the examples belonging to each class that are correctly classified, which is the true positive rate. The F1-score can be interpreted as a weighted average of the precision and recall, and the specificity indicates the true negative rate. The column on the far right of each confusion matrix shows the percentages of precision per class, and the row at the bottom of each confusion matrix shows the percentages of recall per class. The cell in the bottom right of the plot of each confusion matrix shows the overall accuracy. Besides, for the validation set we also present the receiver operating characteristic (ROC) curve. The ROC curve measures the classification performance, that is the true positive rate and the false positive rate of each class, at various thresholds settings.

In Experiment 1, we intended to classify the viruses in 14 different classes, as presented in Table 5, which consists of 10 families (Adenoviridae, Anelloviridae, Circoviridae, Geminiviridae, Genomoviridae, Microviridae, Papillomaviridae, Parvoviridae, Polyomaviridae and Tolecusatellitidae), three orders (Caudovirales, Herpesvirales and Ortervirales) and Riboviria realm. The Riboviria class contains various families that belong to the realm Riboviria, including the Coronaviridae family. To ensure data balance, only the classes with at least 100 sequences from the original dataset were considered. For the classes with more than 500 sequences, only 500 sequences were selected at random, except for the Riboviria class, which was prioritized the Coronaviridade family sequences, to guarantee the correct classification of the test data (SARS-CoV-2 sequences), which is the focus of this work. In this particular case, were selected all Coronaviridade family sequences available in the dataset (206 samples), and the other 294 sequences were select from the rest of the Riboviria data at random. After this balancing, Experiment 1 comprised 3,433 samples of virus sequences.

The SSAE architecture used in Experiment 1 was the $4096 - 3000 - 1000 - 500 - 14$ architecture. The three AEs were trained for 400 epochs. The softmax layer was trained for 3000

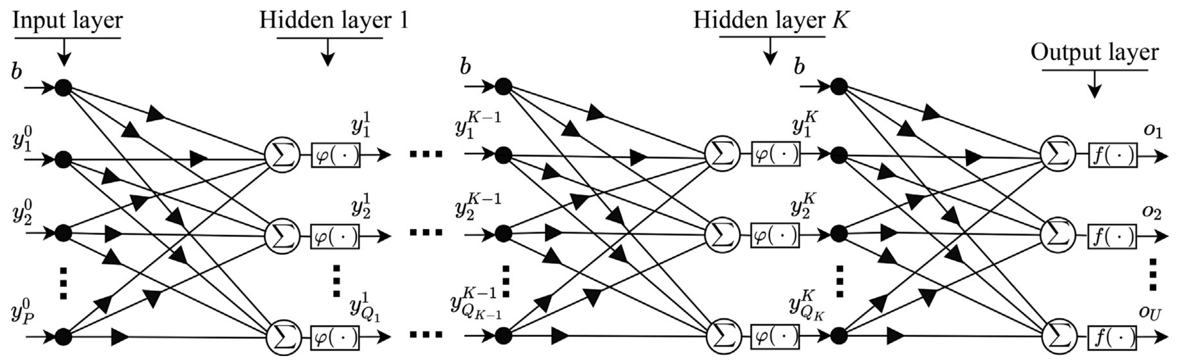


Fig. 4. Deep learning stacked sparse autoencoder architecture (DL-SSAE).

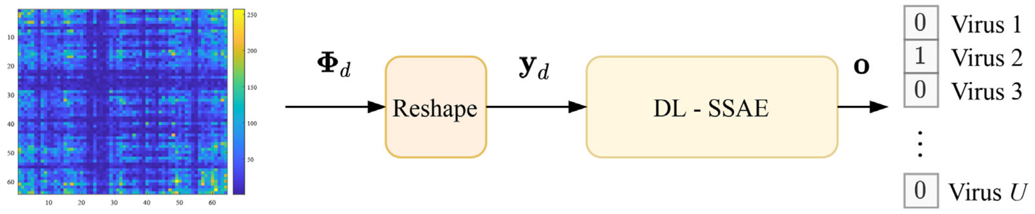


Fig. 5. Viral classification process using *k*-mers images representation with the DL-SSAE.

Table 5
Experiments data.

Experiments	Classes	Number of sequences	SSAE architecture $P - Q_1 - Q_2 - Q_3 - U$
Experiment 1	Adenoviridae	195	4096 – 3000 – 1000 – 500 – 14
	Anelloviridae	114	
	Caudovirales	500	
	Circoviridae	243	
	Geminiviridae	500	
	Genomoviridae	115	
	Herpesvirales	136	
	Microviridae	102	
	Ortervirales	214	
	Papillomaviridae	354	
	Parvoviridae	168	
	Polyomaviridae	142	
	Riboviria	500	
	Tolecusatellitidae	150	
Experiment 2	Picornaviridae	423	4096 – 3000 – 1000 – 500 – 8
	Caliciviridae	392	
	Coronaviridae	206	
	Potyviridae	232	
	Flaviviridae	217	
	Rhabdoviridae	186	
	Betaflexiviridae	129	
	Reoviridae	111	
Experiment 3	Alphacoronavirus	52	4096 – 3000 – 1000 – 500 – 4
	Betacoronavirus	123	
	Deltacoronavirus	20	
	Gammacoronavirus	9	
Experiment 4	Embecovirus	47	4096 – 3000 – 1000 – 500 – 4
	Merbecovirus	17	
	Nobecovirus	9	
	Sarbecovirus	46	

epochs or until reach the minimum gradient ($< 1 \times 10^{-6}$). Lastly, the fine-tuning was performed. For each experiment, the fine-tuning phase uses the same stopping condition as the softmax layer.

The confusion matrix and the ROC curve from the validation set of Experiment 1 are present in Figs. 6 and 7, respectively. In Experiment 1, the classification accuracy from the validation set reached 92%. This result is promising, especially considering the challenges of the classification in high-level taxonomies because of the high diversity of the viruses sequences. It is essential to mention that the balancing process may have caused the classification more complicated because some crucial sequences may have been excluded from the dataset. However, this result can be improved in many ways that will be discussed following.

Regarded to the classification performance per class, the precision value presented in the last column shows that the worse result was obtained from an order class (71.4% from the Herpesvirales). Among the five worst classification results, two are from order classes (71.4% and 83.3% from Herpesvirales and Ortervirales, respectively). Since these classes can contain viruses from many different realms and families, they can difficult the training process. The Riboviria realm, which is the focus of this work, reached a classification accuracy of 93%. Analyse the results per classes can give more understanding about the dataset used and the implications of this dataset for the results, which is important to make decisions for the next experiments.

The confusion matrix from the test set of Experiment 1 is present in Fig. 8. In the test phase of this experiment, all the 1557 sequences of SARS-CoV-2 was correctly classified as belonging to the Riboviria realm, so the classification accuracy reached 100%. For the test set, we only used samples of SARS-CoV-2. For that reason, the columns and rows corresponding to the classes that were not inferred in the test phase received the terminology NaN (Not a Number) in the confusion matrix plot.

Experiment 2 performs the classification of Riboviria families. As in Experiment 1, only classes with at least 100 sequences were considered. This experiment includes 1896 sequences separated into eight families (Picornaviridae, Caliciviridae, Coronaviridae, Potyviridae, Flaviviridae, Rhabdoviridae, Betaflexiviridae and Reoviridae). We used the $4096 - 3000 - 1000 - 500 - 8$ SSAE architecture. The three AEs were trained for 400 epochs each and the softmax layer was trained for 1000 epochs or until reaching the minimum gradient, as well as the fine-tuning phase.

The confusion matrix and the ROC curve from the validation set of Experiment 2 are present in Figs. 9 and 10, respectively. The

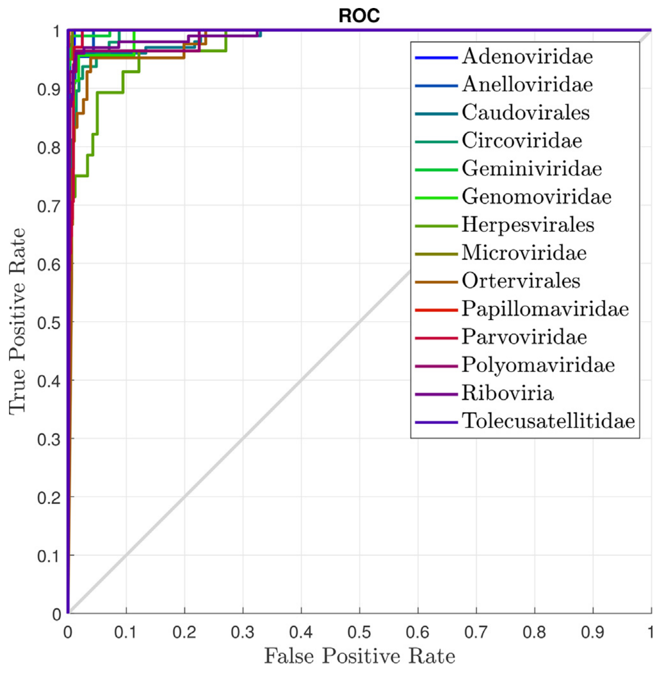
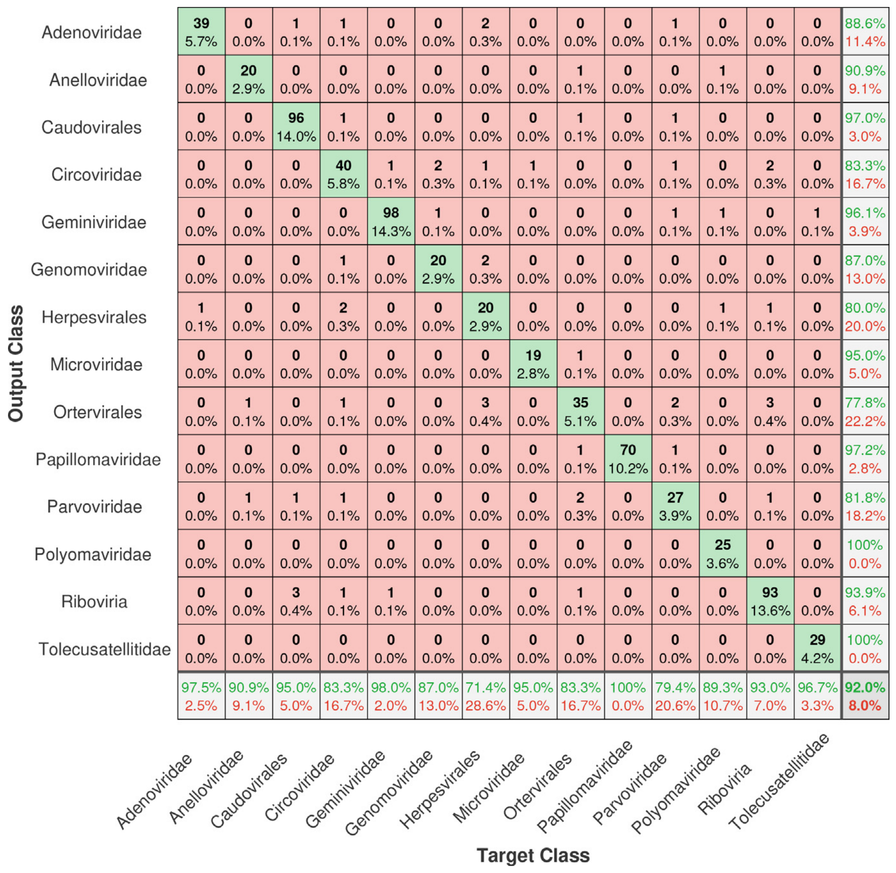


Fig. 7. ROC curve of the validation set from the Experiment 1.

classification accuracy from Experiment 2 reached 96.3%. From the 379 sequences applied in this validation, only 11 were not correctly classified. Besides, the SSAE classified all sequences that belong to the Coronaviridade family correctly. The ROC curve from Experiment 2 also provides excellent results.


Output Class
Target Class

	Adenoviridae	Anelloviridae	Caudovirales	Circoviridae	Geminiviridae	Genomoviridae	Herpesvirales	Microviridae	Ortervirales	Papillomaviridae	Parvoviridae	Polyomaviridae	Riboviria	Tolecusatellitidae	
Adenoviridae	39 5.7%	0 0.0%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	2 0.3%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	88.6% 11.4%
Anelloviridae	0 0.0%	20 2.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	90.9% 9.1%
Caudovirales	0 0.0%	0 0.0%	96 14.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	97.0% 3.0%
Circoviridae	0 0.0%	0 0.0%	0 0.0%	40 5.8%	1 0.1%	2 0.3%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	2 0.3%	0 0.0%	83.3% 16.7%
Geminiviridae	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98 14.3%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	1 0.1%	0 0.0%	1 0.1%	96.1% 3.9%
Genomoviridae	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	20 2.9%	2 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	87.0% 13.0%
Herpesvirales	1 0.1%	0 0.0%	0 0.0%	2 0.3%	0 0.0%	0 0.0%	20 2.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	1 0.1%	0 0.0%	80.0% 20.0%
Microviridae	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	19 2.8%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.0% 5.0%
Ortervirales	0 0.0%	1 0.1%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	3 0.4%	0 0.0%	35 5.1%	0 0.0%	2 0.3%	0 0.0%	3 0.4%	0 0.0%	77.8% 22.2%
Papillomaviridae	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	70 10.2%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	97.2% 2.8%
Parvoviridae	0 0.0%	1 0.1%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.3%	0 0.0%	27 3.9%	0 0.0%	1 0.1%	0 0.0%	81.8% 18.2%
Polyomaviridae	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	25 3.6%	0 0.0%	0 0.0%	100% 0.0%
Riboviria	0 0.0%	0 0.0%	3 0.4%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	93 13.6%	0 0.0%	93.9% 6.1%
Tolecusatellitidae	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	29 4.2%	100% 0.0%
	97.5% 2.5%	90.9% 9.1%	95.0% 5.0%	83.3% 16.7%	98.0% 2.0%	87.0% 13.0%	71.4% 28.6%	95.0% 5.0%	83.3% 16.7%	100% 0.0%	79.4% 20.6%	89.3% 10.7%	93.0% 7.0%	96.7% 3.3%	92.0% 8.0%

Fig. 6. Confusion matrix of the validation set from the Experiment 1.

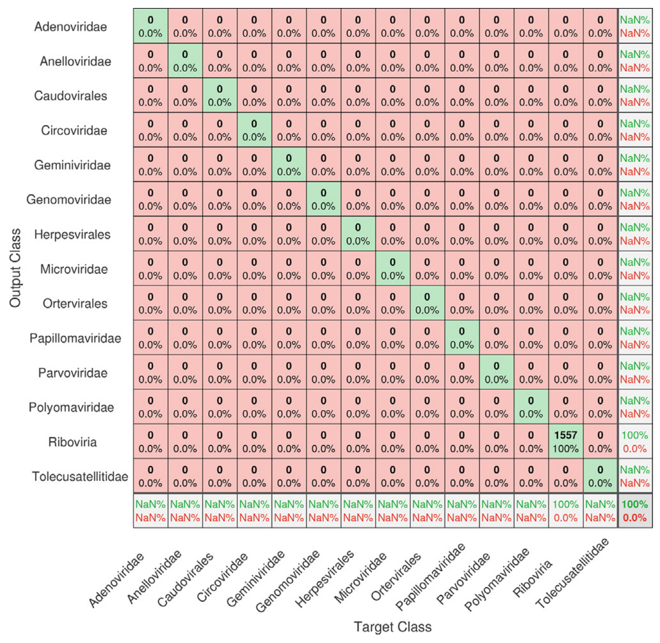


Fig. 8. Confusion matrix of the test set from the Experiment 1. NaN, which means Not a Number, appears in the columns and rows corresponding to the classes that were not inferred in the test phase.

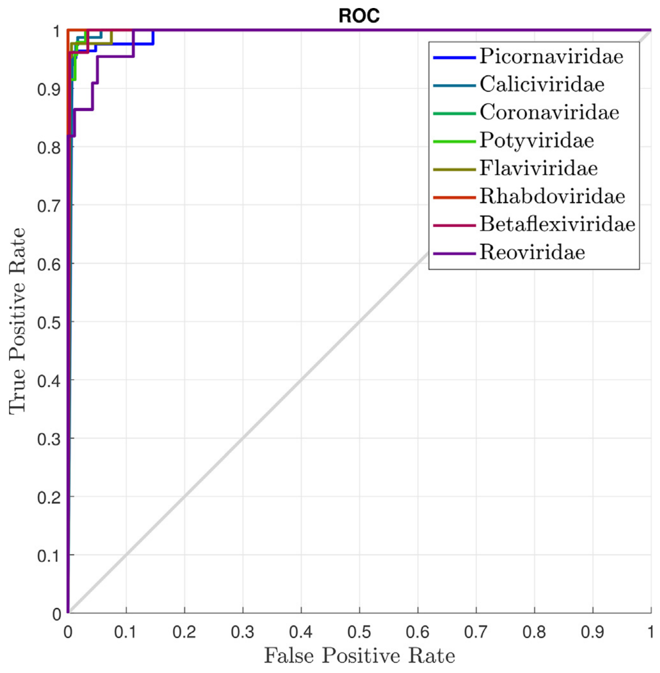


Fig. 10. ROC curve of the validation set from the Experiment 2.

The confusion matrix from the test set of Experiment 2 is present in Fig. 11. The SSAE achieve 100% of classification accuracy, i.e., all SARS-CoV-2 sequences applied in this experiment were perfectly classified as Coronaviridae family sequences.

In Experiment 3 we aim to provide the classification among the Coronaviridae genera. For this experiment, 204 sequences divided into four genera (Alphacoronavirus, Betacoronavirus, Deltacoronavirus and Gammacoronavirus) were used. The SSAE architecture used in this experiment was the $4096 - 3000 - 1000 - 500 - 4$ architecture. The three AEs were trained for 400 epochs each, and the softmax layer was trained for 2000 epochs or until reaching the minimum gradient.

Figs. 12 and 13 show the resulting confusion matrix and ROC curve from the Experiment 3, respectively. This experiment achieved 95% of classification accuracy of the validation set. The classification performance of the model obtained for the Betacoronavirus genus was 95.8%. Also, the ROC curve plotted for all classes of Experiment 3 provides satisfactory results.

Regarding the test set of Experiment 3, the confusion matrix is present in Fig. 14. The test phase of Experiment 3 achieved 98.9%

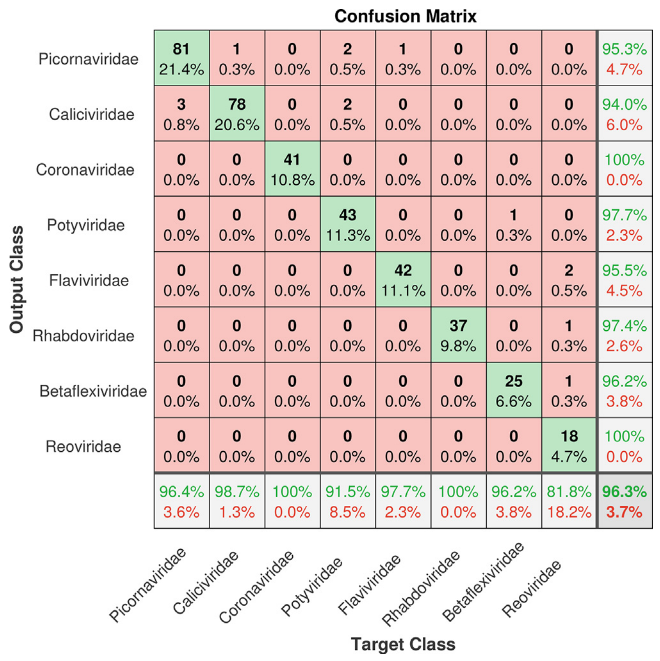


Fig. 9. Confusion matrix of the validation set from the Experiment 2.

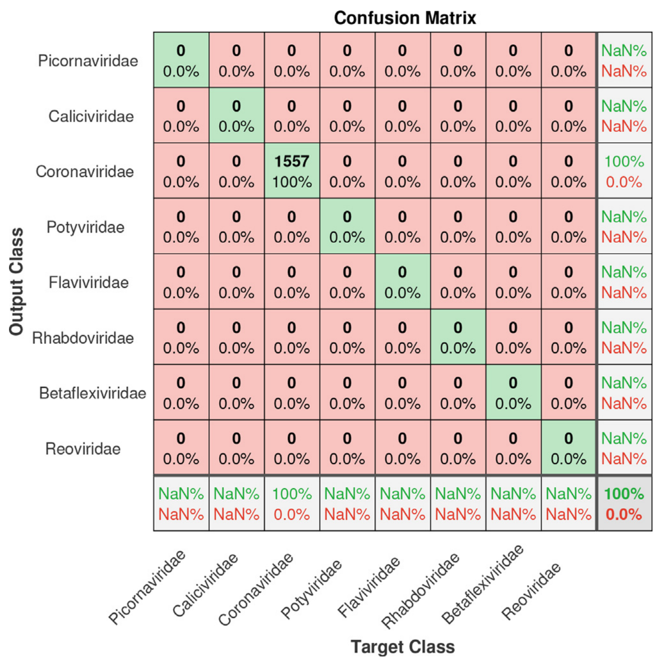


Fig. 11. Confusion matrix of the test set from the Experiment 2. NaN, which means Not a Number, appears in the columns and rows corresponding to the classes that were not inferred in the test phase.

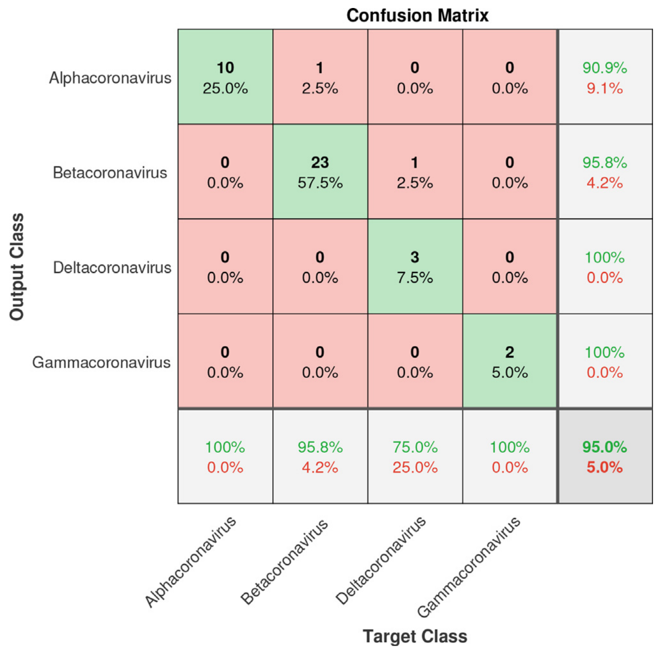


Fig. 12. Confusion matrix of the validation set from the Experiment 3.

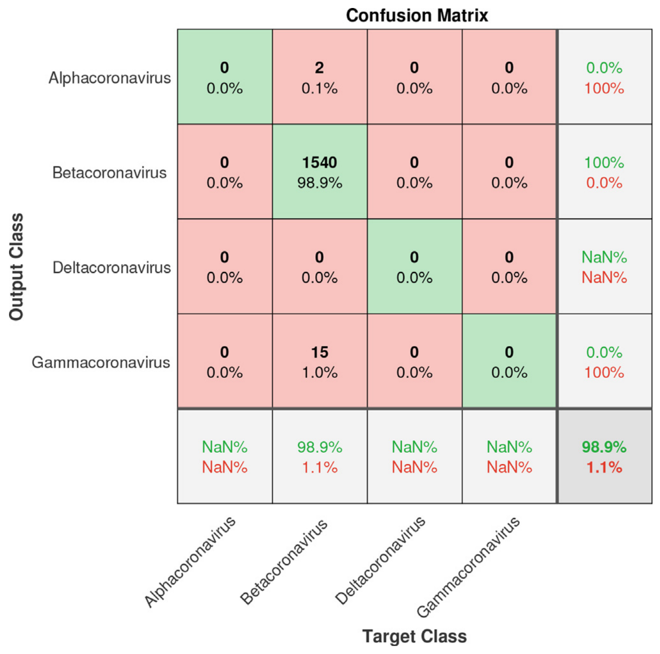


Fig. 14. Confusion matrix of the test set from the Experiment 3. NaN, which means Not a Number, appears in the columns and rows corresponding to the classes that were not inferred in the test phase.

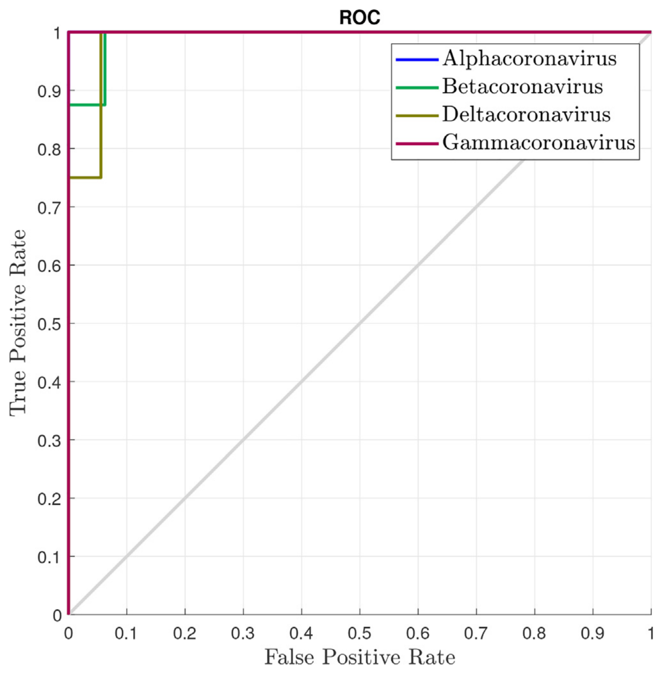


Fig. 13. ROC curve of the validation set from the Experiment 3.

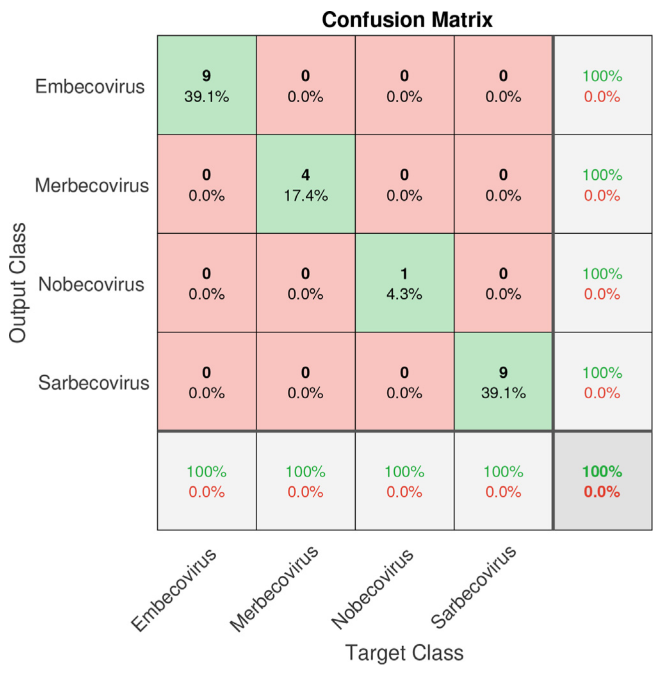


Fig. 15. Confusion matrix of the validation set from the Experiment 4.

of classification accuracy. In the validation phase of Experiment 3, the Betacoronavirus genus did not reach the highest performance, which probably explains these result in the test phase.

In Experiment 4, we provide the Betacoronaviridae subgenera classification. This test includes 119 genome sequences divided into four classes (Embecovirus, Marbecovirus, Nobecovirus and Sarbecovirus). The SSAE architecture was the same as the architecture used in Experiment 3 ($4096-3000-1000-500-4$), as well as the training parameters.

The confusion matrix and the ROC curve from the validation set of Experiment 4 are present in Figs. 15 and 16, respectively. In this experiment, the SSAE achieved the highest classification accuracy (100%), which is reaffirmed for the ROC curve plot. 17.

Fig. 15 exposes the confusion matrix from the test set of Experiment 4. In this case, the SSAE achieved 99.9% of classification accuracy, that is equivalent to only one sequence wrong classified.

Table 6 presents the results regarding some popular classification performance metrics obtained from the validation set. The first column of the table indicates the experiment proposed. The second column shows the overall accuracy for each experiment. The precision, recall, F1-score, and specificity are present in the others columns, which were obtained by the average of the values obtained for each class.

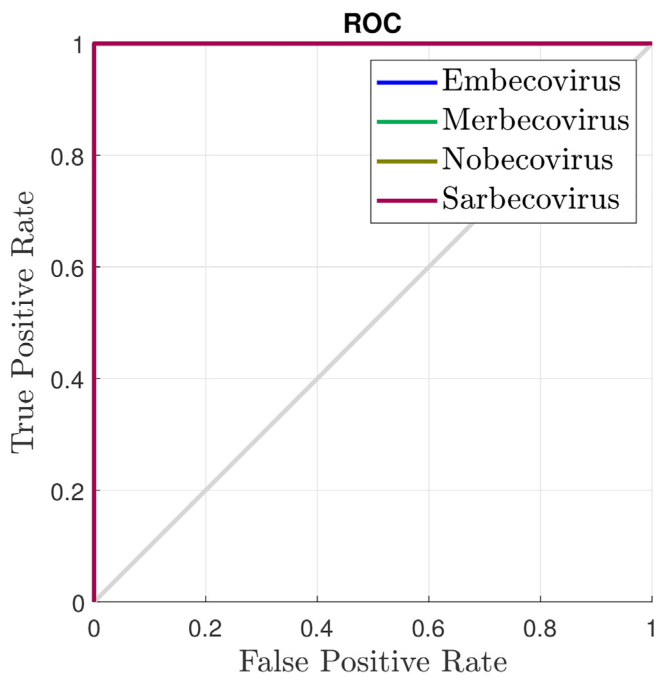


Fig. 16. ROC curve of the validation set from the Experiment 4.

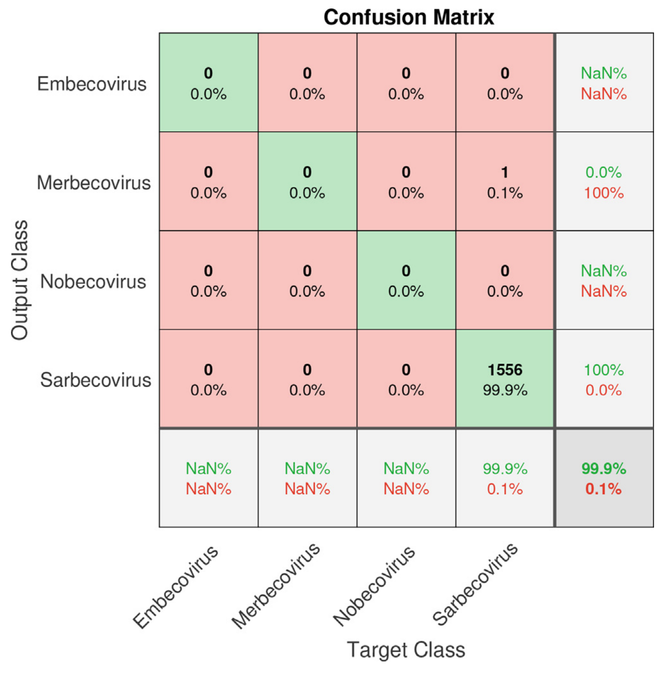


Fig. 17. Confusion matrix of the test set from the Experiment 4. NaN, which means Not a Number, appears in the columns and rows corresponding to the classes that were not inferred in the test phase.

All the metrics presented in Table 6 indicate that the viral classifier proposed performs great for all experiments. The highest performance was obtained for the Experiment 4. Besides, Experiments 2 and 3, reached values more than 0.95 for all the metrics evaluated. The classification performance slightly decreased in the Experiment 1, which is acceptable because of the high diversity of the viruses sequences applied. However, considering all the experiments, the specificity (true negative rate) reached values between 0.983 and 1.

Table 7
Classification performance metrics results obtained from the test set.

Experiment	Accuracy	Recall
1	1 (100%)	1 (100%)
2	1 (100%)	1 (100%)
3	0.989 (98.9%)	0.989 (98.9%)
4	0.999 (99.9%)	0.999 (99.9%)

Table 8
Mean Square Error obtained for the training, validation and test sets of each experiment.

Experiment	Training set	Validation set	Test set
1	5.8×10^{-13}	1.1×10^{-2}	3.4×10^{-103}
2	4.1×10^{-13}	9.4×10^{-3}	2.9×10^{-24}
3	5.8×10^{-13}	2.2×10^{-2}	5.3×10^{-2}
4	1.6×10^{-12}	1.1×10^{-12}	3.8×10^{-4}

Table 7 presents the results regarding some popular classification performance metrics obtained from the test set. The first column of the table indicates the experiment proposed. The second column shows the overall accuracy for each experiment. And the last column shows the recall, or true positive rate, which were obtained only for the class that corresponds to the SARS-CoV-2 samples. The other metrics (precision, F1-score, and specificity) are not presented because in the tests we do not have false positives samples.

When the SARS-CoV-2 samples were applied, all the experiments perform excellently. The accuracy reached values between 98.9% and 100%, as well as the recall (true positive rate). The results presented in Table 7 are very significant since the classification of the SARS-CoV-2 virus was the main objective of this study.

We perform the error calculation of the classification analysis by the mean square error (MSE) between the target output and the SSAE output of each experiment. Table 8 shows the MSE obtained for the training, validation and test sets.

Table 8 indicates that for all experiments, the MSE of the training set was very acceptable, as well as the MSE of the validation and test sets. As expected, for most experiments, the MSE of the training was lower than the MSE of the validation.

In order to provide results about the temporal complexity of our experiments, Table 9 shows the training time of each experiment, describing the training time of each layer (AEs and softmax), the fine-tuning phase, and the total training time. The training in each autoencoder was performed with an NVIDIA GeForce GTX GPU (Intel Core i5-9300H host CPU), and for the softmax layer and the fine-tuning phase, a CPU (Intel Core i5-9300H 2.4 GHz) was used.

Table 6
Classification performance metrics results obtained from the validation set.

Experiment	Accuracy	Precision	Recall	F1 score	Specificity
1	0.920 (92.0%)	0.924 (92.4%)	0.920 (92.0%)	0.931 (93.1%)	0.993 (99.3%)
2	0.963 (96.3%)	0.968 (96.8%)	0.971 (97.1%)	0.962 (96.2%)	0.997 (99.7%)
3	0.950 (95.0%)	0.979 (97.9%)	0.979 (97.9%)	0.955 (95.5%)	0.983 (98.3%)
4	1 (100%)	1 (100%)	1 (100%)	1 (100%)	1 (100%)

Table 9
Final SSAE training time of each experiment.

Experiment	First AE	Second AE	Third AE	Softmax Layer	Fine-tuning	Total Training Time
1	7h29m00s	19m12s	4m49s	44s	36m50s	≈ 8h30m
2	1h39m11*s*	11*m*56*s*	3m16s	10 s	5m24s	≈ 2h
3	32m24s	5m50s	2m22s	0s	27s	≈ 41m
4	19m15s	4m54s	1m16s	0s	11s	≈ 25m

Table 10
State of the art comparison associated with the classification accuracy of the validation set.

Reference	Algorithm	Exp. 1	Exp. 2	Exp. 3	Exp. 4
This work	SSAE	92%	96.3%	95%	100%
[11]	Linear Discriminant	91.7%	91.2%	98.1%	97.6%
[11]	Linear SVM	90.8%	89.2%	94.2%	98.4%
[11]	Quadratic SVM	95%	93.1%	95.2%	98.4%
[11]	Fine KNN	93.4%	90.3%	95.7%	97.6%
[11]	Subspace Discriminat	87.6%	89%	97.6%	98.4%
[11]	Subspace KNN	93.2%	90.4%	96.2%	97.2%
[11]	Average Accuracy	92%	90.5%	96.2%	97.6%

Table 11
State of the art comparison associated with the mapping pipeline.

Reference	Mapping pipeline	Number of inputs
This work	*k*-mers	4^k
[11]	*k*-mers + CGR + FFT + PCC	*D*

As shown in Table 9, most of the experiments finished the training in the softmax layer and the fine-tuning in seconds or a few minutes, That occurred because they reached the minimum gradient after some training epochs, indicating that the three AEs trained before extracted relevant information about the dataset used.

In Table 10, we compare the results obtained from each experiment and the results of another work from the state of the art, which performed taxonomy classification of the SARS-CoV-2 virus using different machine learning techniques.11.

Table 10 shows that for experiments 1 and 3, the techniques used in [11] achieve lower, equivalent or superior accuracy than the SSAE technique applied in our work. However, considering experiments 2 and 4, the SSAE technique provides superior classification accuracy results than all the techniques applied in [11]. For experiment 1, the DL-SSAE had a slightly lower accuracy compared to Quadratic SVM (difference was 3%), Fine KNN (difference was 1.4%), and Subspace KNN (difference was 1.2%). For experiment 3, the DL-SSAE had a slightly lower accuracy compared to Linear Discriminant (difference was 3.1%), Quadratic SVM (difference was 0.2%), Fine KNN (difference was 0.7%), Subspace Discriminat (difference was 2.6%), Subspace KNN (difference was 1.2%), and Average Accuracy (difference was 1.2%). However, it is essential to understand that the proposal presented in [11] uses a high-complexity mapping pipeline. This pipeline is composed of the *k*-mers followed by chaos game representation (CGR), Fast Fourier transform (FFT) and Pearson correlation coefficient calculation (PPC) for each *d*-th sequence. In the final, each *d*-th sequence is converted into a *D* dimension vector, where *D* is the number of sequences in the dataset (see SubSection 3.1). In other words, the ML input size used in the proposal presented in [11] is a function of the number of sequences of the dataset. This characteristic can be prohibitive for several viral classification applications with a large dataset. In the work way, the work proposal in this manuscript, each *d*-th sequence is converted into a 4096 elements vector (or bi-dimensional matrix of 64×64) for $k = 6$. In other words, the ML input is not dependent on the number of sequences in the dataset.

In all experiments of this work, the SSAE technique provided great performance results, especially for the test set. However, some strategies can be applied in future experiments to improve classification accuracy results. One of them consists in the use of the *k*-fold cross-validation scheme. We also intend to study data balancing alternatives based on the analysis of the results presented here. Besides, we plan to extend this work by applying the SSAE technique to classify the SARS-CoV-2 variants.

5. Conclusions

This work presented an efficient viral genome classifier for the SARS-CoV-2 virus using the DNN based on the stacked sparse autoencoder technique. Our model is able to classify genome sequences of the SARS-CoV-2 virus in various levels of taxonomy. We perform four experiments in order to classify realm, family, genus and subgenus. We explored the utilization of *k*-mers image representation of the whole genome sequence as the DNN input, which feasibility the use of genome sequences of any length and enable the use of smaller network inputs. We measured the effectiveness of the model by some popular classification performance metrics (accuracy, precision, recall, F1 score, and specificity), which are metrics used in many works in the literature, as presented in [12,14,15]. Besides, for each experiment, we plot the ROC curve for the validation set and the confusion matrix for the validation and test sets. All experiments provided great performance results, reaching accuracies between 92% and 100% for the validation set and between 98.9% and 100% for the test set, which contains only SARS-CoV-2 samples. These results indicated the applicability of using our model, based on the stacked sparse autoencoder technique, in genome classification problems. Our approach can be adapted to classify other emerging viruses. However, the model may require to be retrained to include new data and satisfy some conditions. It is essential to consider some implications of that training process since it is necessary to previously define the classes that will be used for training the SSAE. One of the requirements for the correct classification of a new virus by our model is that the training process includes samples that belong to the same class, which could be the family, genus or another taxonomic level of the new virus being classified. In the future, we plan to extend this work by performing experiments with another image representation of the genome sequences and applying the SSAE technique to classify the SARS-CoV-2 variants.

Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Finance Code 001.

CRediT authorship contribution statement

Maria G.F. Coutinho: Investigation, Methodology, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Gabriel B.M. Câmara:** Investigation, Methodology, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Raquel M. de Barbosa:** Data curation, Formal analysis, Investigation, Methodology, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing. **Marcelo A.C. Fernandes:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors wish to acknowledge the financial support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

References

[1] Lam TT-Y, Shum MH-H, Zhu H-C, Tong Y-G, Ni X-B, Liao Y-S, Wei W, Cheung WY-M, Li W-J, Li L-F, et al. Identifying sars-cov-2 related coronaviruses in malayan pangolins. Nature 2020:1–6.
[2] Andersen KG, Rambaut A, Lipkin WI, Holmes EC, Garry RF. The proximal origin of sars-cov-2. Nature Med 2020;26(4):450–2.
[3] R.L. Graham, R.S. Baric, Sars-cov-2: Combating coronavirus emergence, Immunity.
[4] Zielezinski A, Vinga S, Almeida J, Karlowski WM. Alignment-free sequence comparison: benefits, applications, and tools. Genome Biol 2017;18(1):186.
[5] Zou J, Huss M, Abid A, Mohammadi P, Torkamani A, Telenti A. A primer on deep learning in genomics. Nature Genet 2019;51(1):12–8.
[6] Tang B, Pan Z, Yin K, Khateeb A. Recent advances of deep learning in bioinformatics and computational biology. Front Genet 2019;10:214.
[7] Eraslan G, Avsec Ž, Gagneur J, Theis FJ. Deep learning: new computational modelling techniques for genomics. Nat Rev Genet 2019;20(7):389–403.
[8] Pareek CS, Smoczynski R, Tretyn A. Sequencing technologies and genome sequencing. J Appl Genet 2011;52(4):413–35.
[9] Pabinger S, Dander A, Fischer M, Snajder R, Sperk M, Efremova M, Krabichler B, Speicher MR, Zschocke J, Trajanoski Z. A survey of tools for variant analysis of next-generation genome sequencing data. Briefings Bioinform 2014;15 (2):256–78.
[10] Posada-Cespedes S, Seifert D, Beerenwinkel N. Recent advances in inferring viral diversity from high-throughput sequencing data. Virus Res 2017;239:17–32.
[11] Randhawa GS, Soltysiak MP, El Roz H, de Souza CP, Hill KA, Kari L. Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: Covid-19 case study. Plos One 2020;15(4):e0232391.
[12] Fabijańska A, Grabowski S. Viral genome deep classifier. IEEE Access 2019;7:81297–307.
[13] A. Lopez-Rincon, A. Tonda, L. Mendoza-Maldonado, E. Claassen, J. Garssen, A.D. Kraneveld, Accurate identification of sars-cov-2 from viral genome sequences using deep learning, bioRxiv.
[14] J.M. Bartoszewicz, A. Seidel, B.Y. Renard, Interpretable detection of novel human viruses from genome sequencing data, bioRxiv doi:10.1101/2020.01.29.925354.
[15] Liang Q, Bible PW, Liu Y, Zou B, Wei L. Deepmicrobes: taxonomic classification for metagenomics with deep learning. NAR Genom Bioinform 2020;2(1): lqaa009.
[16] J. Shang, Y. Sun, Cheer: hierarchical taxonomic classification for viral metagenomic data via deep learning, Methods.
[17] Tampuu A, Bzhalava Z, Dillner J, Vicente R. Viraminer: Deep learning on raw dna sequences for identifying viral genomes in human samples. PloS One 2019;14(9):e0222271.
[18] Ren J, Song K, Deng C, Ahlgren NA, Fuhrman JA, Li Y, Xie X, Poplin R, Sun F. Identifying viruses from metagenomic data using deep learning. Quant Biol 2020:1–14.
[19] F. Mock, A. Viehweger, E. Barth, M. Marz, Viral host prediction with deep learning, bioRxiv (2019) 575571.
[20] Morales JA, Saldaña R, Santana-Castolo MH, Torres-Cerna CE, Borrayo E, Mendizabal-Ruiz AP, Vélez-Pérez HA, Mendizabal-Ruiz G. Deep learning for the classification of genomic signals. Math Probl Eng 2020.
[21] Nguyen NG, Tran VA, Ngo DL, Phan D, Lumbanraja FR, Faisal MR, Abapihi B, Kubo M, Satou K, et al. Dna sequence classification by convolutional neural network. J Biomed Sci Eng 2016;9(05):280.
[22] Guo Y, Li W, Wang B, Liu H, Zhou D. Deepaclstm: deep asymmetric convolutional long short-term memory neural models for protein secondary structure prediction. BMC Bioinform 2019;20(1):1–12.
[23] Heidari A, Jafari Navimipour N, Unal M, Toumaj S. Machine learning applications for covid-19 outbreak management. Neural Comput Appl 2022:1–36.
[24] Heidari A, Toumaj S, Navimipour NJ, Unal M. A privacy-aware method for covid-19 detection in chest ct images using lightweight deep conventional neural network and blockchain. Comput Biol Med 2022;145:. https://doi.org/10.1016/j.compbiomed.2022.105461. https://www.sciencedirect.com/science/article/pii/S0010482522002530105461.
[25] Heidari A, Jafari Navimipour N, Unal M, Toumaj S. The covid-19 epidemic analysis and diagnosis using deep learning: A systematic literature review and future directions. Comput Biol Med 2022;141:. https://doi.org/10.1016/j.compbiomed.2021.105141. https://www.sciencedirect.com/science/article/pii/S0010482521009355105141.
[26] G.J.L., B. Abraham, S.M.S., M.S. Nair, A computer-aided diagnosis system for the classification of covid-19 and non-covid-19 pneumonia on chest x-ray images by integrating cnn with sparse autoencoder and feed forward neural network, Comput Biol Med 141 (2022) 105134. doi:https://doi.org/10.1016/j.compbiomed.2021.105134. https://www.sciencedirect.com/science/article/pii/S0010482521009288.
[27] Carracedo-Reboredo P, Liñares-Blanco J, Rodríguez-Fernández N, Cedrón F, Novoa FJ, Carballal A, Maojo V, Pazos A, Fernandez-Lozano C. A review on machine learning approaches and trends in drug discovery, Computational and Structural. Biotechnol J 2021;19:4538–58. https://doi.org/10.1016/j.csbj.2021.08.011. https://www.sciencedirect.com/science/article/pii/S2001037021003421.
[28] H. Zhu, Q. Guo, M. Li, C. Wang, Z. Fang, P. Wang, J. Tan, S. Wu, Y. Xiao, Host and infectivity prediction of wuhan 2019 novel coronavirus using deep learning algorithm, BioRxiv.
[29] Fang Z, Tan J, Wu S, Li M, Xu C, Xie Z, Zhu H. Ppr-meta: a tool for identifying phages and plasmids from metagenomic fragments using deep learning. GigaScience 2019;8(6):giz066.
[30] C. Pian, Z. Li, H. Jiang, L. Kong, Y. Chen, L. Zhang, Deep6ma: a deep learning framework for exploring similar patterns in dna n6-methyladenine sites across different species, bioRxiv.
[31] Kuang S, Wang L. Identification and analysis of consensus rna motifs binding to the genome regulator ctcf. NAR Genom Bioinform 2020;2(2):lqaa031.
[32] Zhang Y, Qiao S, Ji S, Li Y. Deepsite: bidirectional lstm and cnn models for predicting dna–protein binding. Int J Mach Learn Cybern 2019:1–11.
[33] Remita MA, Halioui A, Daigle B, Kiani G, Diallo AB, et al. A machine learning approach for viral genome classification. BMC Bioinform 2017;18(1):208.
[34] Ren J, Song K, Deng C, Ahlgren NA, Fuhrman JA, Li Y, Xie X, Poplin R, Sun F. Identifying viruses from metagenomic data using deep learning. Quant Biol 2020:1–14.
[35] L. Dey, S. Chakraborty, A. Mukhopadhyay, Machine learning techniques for sequence-based prediction of viral–host interactions between sars-cov-2 and human proteins, Biomed J.
[36] Bzhalava Z, Tampuu A, Bała P, Vicente R, Dillner J. Machine learning for detection of viral sequences in human metagenomic datasets. BMC Bioinform 2018;19(1):336.
[37] Whata A, Chimedza C. Deep learning for sars cov-2 genome sequences. IEEE Access 2021;9:59597–611. https://doi.org/10.1109/ACCESS.2021.3073728.
[38] Xu J, Xiang L, Liu Q, Gilmore H, Wu J, Tang J, Madabhushi A. Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images. IEEE Trans Med Imaging 2016;35(1):119–30.
[39] Pratiher S, Chattoraj S, Vishwakarma K. Application of stacked sparse autoencoder in automated detection of glaucoma in fundus images. Unconventional Optical Imaging, vol. 10677. International Society for Optics and Photonics; 2018. p. 106772X.
[40] Xiao Y, Wu J, Lin Z, Zhao X. A semi-supervised deep learning method based on stacked sparse auto-encoder for cancer prediction using rna-seq data. Comput Methods Programs Biomed 2018;166:99–105.
[41] Rizzo R, Fiannaca A, La Rosa M, Urso A. A deep learning approach to dna sequence classification. In: International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics. Springer; 2015. p. 129–40.

[42] R. de M. Barbosa, M.A. Fernandes, k-mers 1d and 2d representation dataset of sars-cov-2 nucleotide sequences, Mendeley Data v2. doi:https://doi.org/10.17632/f5y9cggnxy.2.https://data.mendeley.com/datasets/f5y9cggnxy/2.
[43] Mapleson D, Garcia Accinelli G, Kettleborough G, Wright J, Clavijo BJ. KAT: a K-mer analysis toolkit to quality control NGS datasets and genome assemblies. Bioinformatics 2016;33(4):574–6. https://doi.org/10.1093/bioinformatics/btw663.
[44] Chor B, Horn D, Goldman N, Levy Y, Massingham T. Genomic dna k-mer spectra: models and modalities. Genome Biology 2009;10(10):R108.
[45] Han G-B, Cho D-H. Genome classification improvements based on k-mer intervals in sequences. Genomics 2019;111(6):1574–82. https://doi.org/10.1016/j.ygeno.2018.11.001. https://www.sciencedirect.com/science/article/pii/S0888754318304476.
[46] Ghandi M, Lee D, Mohammad-Noori M, Beer MA. Enhanced regulatory sequence prediction using gapped k-mer features. PLOS Comput Biol 2014;10(7):1–15. https://doi.org/10.1371/journal.pcbi.1003711.
[47] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT press; 2016.
[48] The MathWorks, Train Stacked Autoencoders for Image Classification,https://www.mathworks.com/help/deeplearning/ug/train-stacked-autoencoders-for-image-classification.html (Sep 2020).