



# A Recurrent Neural Network approach for whole genome bacteria identification

Luis Lugo & Emiliano Barreto- Hernández

**To cite this article:** Luis Lugo & Emiliano Barreto- Hernández (2021) A Recurrent Neural Network approach for whole genome bacteria identification, Applied Artificial Intelligence, 35:9, 642-656, DOI: [10.1080/08839514.2021.1922842](https://doi.org/10.1080/08839514.2021.1922842)

**To link to this article:** <https://doi.org/10.1080/08839514.2021.1922842>



Published online: 20 May 2021.



Submit your article to this journal [↗](#)



Article views: 1074



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)



# A Recurrent Neural Network approach for whole genome bacteria identification

Luis Lugo and Emiliano Barreto- Hernández

Instituto de Biotecnología, Universidad Nacional de Colombia, Bogotá, Colombia

## ABSTRACT

The identification of bacteria plays an essential role in multiple areas of research. Those areas include experimental biology, food and water industries, pathology, microbiology, and evolutionary studies. Although there exist methodologies for identification, a transition to a whole-genome sequence-based taxonomy is already undergoing. Next-Generation Sequencing helps the transition by producing DNA sequence data efficiently. However, the rate of DNA sequence data generation and the high dimensionality of such data need faster computer methodologies. Machine learning, an area of artificial intelligence, has the ability to analyze high dimensional data in a systematic, fast, and efficient way. Therefore, we propose a sequential deep learning model for bacteria identification. The proposed neural network exploits the vast amounts of information generated by Next-Generation Sequencing, in order to extract an identification model for whole-genome bacteria sequences. After validating the identification model, the bidirectional recurrent neural network outperformed other classification approaches.



## KEYWORDS

Bacteria identification; whole genome sequence; recurrent neural networks

## Introduction

The identification of bacteria plays an important role in multiple areas of research. Experimental research in biology benefits from the elegant mechanisms of gene activity control in bacteria, the rapid rate at which they grow, and the powerful genetics they have (Lodish et al. 2004). Pathology and microbiology also benefit from bacteria taxonomy as it serves as the basis for understanding certain diseases (Mohamad et al. 2014). Moreover, fast identification of bacteria is critical to ensure the quality of water and food products (Singhal et al. 2015). It is also important from the evolutionary standpoint because it allows the documentation of changes in genes and proteins (Lodish et al. 2004).

There exist a number of criteria for the classification of bacteria and archaea, including morphology, genome size, lifestyle, relevance to human

**CONTACT** Luis Lugo  [lelugom@unal.edu.co](mailto:lelugom@unal.edu.co)  Instituto de Biotecnología, Universidad Nacional de Colombia, Bogotá, Colombia

This article has been republished with minor change. This change do not impact on the academic content of the article.

© 2021 Taylor & Francis

disease, molecular phylogeny using rRNA, and genomic sequence analysis (Mohamad et al. 2014; Pevsner 2015). Methods for bacteria identification include mass spectrometry (Singhal et al. 2015), whole-genome sequencing for clinical samples (Hasman et al. 2013), genome-wide Average Nucleotide Identity (Garrity and Kraft 2016; Varghese et al. 2015), pattern recognition in image processing (Mohamad et al. 2014), and deep neural network architectures (Bosco and Di Gangi 2016; Rizzo et al. 2015).

Moreover, as advances in Next-Generation Sequencing (NGS) generate high dimensional biological data at a faster rate (Schmidt and Hildebrandt 2017), and more curated databases of biological sequences are publicly available, there exists the need to process considerable amounts of high dimensional data in a systematic and efficient way. The unprecedented amount of genomic data generated by NGS platforms increases the demand for large-scale data analysis (Pais et al. 2014). Extracting useful information from those large datasets is challenging. Manual processing of such volumes of data is slow, expensive, and impractical. It is also error prone and highly subjective (Fayyad, Piatetsky-Shapiro, and Smyth 1996). Often, traditional software tools cannot process large datasets or efficient processing algorithms cannot be developed beforehand (Dahl 2015; Tan et al. 2018). In fact, available software systems and algorithms for biological sequence processing should be improved. Computational tasks pose some of the most significant challenges in processing efficiency regarding Whole-Genome Sequencing (WGS) projects (Scholz, Lo, and Chain 2012).

Deep neural networks provide a set of methods to analyze such high dimensional data efficiently (Murphy 2012), generating an explosion of deep learning applications for research in bioinformatics (Min, Lee, and Yoon 2016). Computational biology is another area that is taking the benefits of deep neural network approaches to leverage very large datasets of high dimensional data. Biological datasets are explored to extract hidden structures within them and to make accurate predictions. Applications of deep neural networks are found in regulatory genomics, drug discovery, cellular imaging, medical imaging, genomic data mining, biomedical signal processing, and sequence classification (Angermueller et al. 2016; Bosco and Di Gangi 2016; Min, Lee, and Yoon 2016; Nguyen et al. 2016; Rizzo et al. 2015; Webb 2018).

(Rizzo et al. 2015) proposed a machine learning architecture for DNA sequence classification. Its implementation uses Theano, a python library for artificial intelligence models. The system proposed is a convolutional neural network for processing RNA sequences. The representation of RNA sequences is performed using k-mers occurrences. The goal is to take advantage of convolutional neural networks ability to extract hidden features. Thus, the model can extract features represented by k-mers from input sequences. As far as testing is concerned (Rizzo et al. 2015) chose 16S rRNA sequences. They compare the proposed architecture with existing machine learning models,

and a previous model developed by the team. The previous model by the same team is a classifier based on a General Regression Neural Network (GRNN) model. The research concludes that CNN models get very good results at the different taxonomic levels. CNNs outperformed existing machine learning classifiers when processing full-length 16S rRNA sequences or 500bp fragments. Nonetheless, results obtained by the CNN are very similar to those obtained by the GRNN model. This happens when the models are processing full-length 16S rRNA sequences. When the models process 500bp 16S rRNA sequence fragments, the CNN architecture manages to outperform all the other models. That is quite important because in metagenomics, researchers usually have access only to fragments of DNA sequences.

The work in Bosco and Di Gangi (2016) compared two different types of deep learning models for automatic classification of bacteria species. Classification is performed without a previous feature extraction process. They propose convolutional neural networks and recurrent neural networks. For recurrent neural networks, authors use the long-short-term memory (LSTM) model. For testing purposes, the deep learning architecture was fed with a dataset encoded according to the IUPAC nucleotide representation. The dataset is downloaded from the Ribosomal Database Project (RDP) (Cole et al. 2013). The dataset uses the 16S ribosomal RNA to identify five ordered taxonomic ranks (Phylum, Class, Order, Family, and Genus). One architecture implements a separate neural network for each taxonomic rank, while the other architecture uses multitask learning by adding separate layers for each taxonomic rank on the top of the processing layers. Their research allowed to conclude that, overall, LSTMs have a better classification output than CNNs. Another conclusion states that multitask learning improves the performance of LSTM architectures, but it harms CNN models (Bosco and Di Gangi 2016).

Our main contribution is the application of deep neural networks to improve existing DNA sequence analysis tools for WWhole-GenomeSequence classification of bacteria. An improved identification system has the potential to help the ongoing transition from 16s ribosomal RNA taxonomy to a whole genome-based bacteria taxonomy (Garrity and Kraft 2016). Also, it takes advantage of the increasing amount of data generated from NGS technologies. This is particularly important because other methods become computationally more expensive as more data is available. But deep learning methods are data-intensive applications that benefit from an increasing amount of data available. Therefore, we propose a sequential deep learning architecture – a recurrent neural network – for bacteria identification (Figure 1), using NGS data.

## Methods

The identification of bacteria using their whole-genome sequences is a part of a broader set of tasks in machine learning: **sequence labeling** (Graves 2012). In sequence labeling, an input sequence of data points is transcribed with a discrete set of output labels. It is common to have sequence labeling tasks in time series data, where each data point in the input sequence is a time step. Nonetheless, the same approach works for non-temporal tasks such as protein secondary structure prediction and DNA sequence classification (Graves 2012).

Sequence classification is the most restrictive case of sequence labeling applications. The input sequence should be matched to a label sequence of unitary length. But this case has an advantage over the other cases because the algorithm can process the whole sequence before generating the label. This kind of classification is what we perform with the bacterial whole-genome sequences. Sequential models not only have the advantage of robustness against distortions and shifting in the input sequence, but also can discover what context information is essential for a correct output, as they can acquire a better knowledge of the overall sequence structure (Graves 2012).

The classical error rate in pattern classification – the classification error rate – enables the computation of a loss function for the models. The error rate is the relationship between correct input-target outputs and the testing set size (Graves 2012). For neural network outputs in classification systems with more than two classes, a softmax function provides normalized output activations that represent the class probabilities (Graves 2012). True class probabilities are obtained by representing the true labels with a 1-of-K coding scheme (Cho et al. 2014), a binary vector with one-hot encoding (Section 2.2). The cross entropy loss function provides the target function that we minimize to train the network. By doing so, we minimize the classification error rate.

Considering metrics for model evaluation, accuracy alone could be a poor estimator of neural network performance in some applications. Accuracy defines the relationship between the correct outputs and the total number of outputs. A more robust approach include the calculation of precision and recall. Precision defines the fraction of detections reported by the network that are correct. On the other hand, recall defines the fraction of true events in the testing set that were successfully detected by the network. A common summarization of neural network performance combines precision  $p$  and recall  $r$  into the balanced  $F$ -score (Goodfellow, Bengio, and Courville 2016; Rizzo et al. 2015). We use the four metrics to assess our model in section 3.

## **Recurrent Neural Networks**

Sequences found in biology naturally fit the processing power of recurrent neural networks (RNNs). That happens because of the temporal modeling capabilities of RNNs. RNNs are inspired by the cyclical connection of the neurons inside our brain. They store information from input sequences by using iterative function loops. RNNs are an ideal architecture for sequence labeling tasks because they store context information in a flexible manner. By learning what to store and what to ignore, they accept input data in different types and representations. Also, They can understand sequential patterns in the presence of sequential noise (Graves 2012).

In some applications regarding sequence analysis, the current output depends on both past inputs and future inputs. RNNs process input data in one direction, thus, context information is available from one side of the sequence. For instance, in time series data, context is available from the past but there is no way data after the current point in time can alter the context. Therefore, bidirectional RNNs were created to cope with those sequence analysis requirements. In this kind of networks, we assemble two hidden layers. One processes the sequence from the first input while the other processes the sequence from the last input. The output layer is connected to both backward and forward hidden layers and it only generates the output after processing the whole sequence (Graves 2012; Lipton, Berkowitz, and Elkan 2015).

Cells for the bidirectional layers include Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs). LSTMs are a redesign of RNNs around a memory cell. The redesign improves their ability to store context information on very long sequences. In fact, they solve complex long-time-lag tasks that have never been solved by existing models (Graves 2012). GRUs are motivated by LSTMs but are simpler to compute and implement. Also, they can dynamically control the time scale and forgetting behavior of the units in the network. Units comprise update and reset gates. Update gates select if the hidden state will be updated with a new hidden state. On the other hand, reset gates determine if the previous hidden state will be ignored (Martens and Sutskever 2011; Cho et al. 2014; Goodfellow, Bengio, and Courville 2016).

## **Sequence Representation**

Regarding sequence representations, a one-hot vector (Giang Nguyen et al. 2016) encoding converts genomic or protein sequences into a two-dimensional numerical matrix. Another digital encoding, the k-mers representation (Rizzo et al. 2015), generates a fixed-length representation using occurrences of overlapping subsequences with a length of k. K-mers are small DNA or protein sequences of k length. Based on the occurrence of

those small sequences, the system computes and spectral representation of input data. However, an important aspect of sequence labeling problems is the use of context in the input sequence. Based on results from Natural Language Processing (NLP), the seq2vec (Kimothi et al. 2016) extends the use of a single  $k$  length in a  $k$ -mers representation, to create a distributed representation of the sequences in a Euclidean space. The distributed representation, which considers different  $k$ 's, has the potential to capture contextual information in the original sequence. The dna2vec system (Ng 2017) uses a distributed representation as well, considering  $k = \{3, 4, 5, 6, 7, 8\}$  to generate vectors in a 100-dimensional space. The cosine similarity of those vectors is correlated to the Needleman–Wunsch similarity score.

The distributed representation provides contextual information and helps to deal with high dimensionality in the sequences – an important aspect considering that neural networks benefit from reasonably compact continuous vector inputs (Graves 2012; Ng 2017). An additional benefit is the invariance of the distributed representation to the order of the nodes in the FASTA file. Although scaffolds are ordered in a FASTA file, contigs are not necessarily ordered. Thus, we use a distributed representation for encoding the bacterial whole-genome sequences. Our representation takes inspiration from the multiple  $k$ -mers in dna2seq, but we consider a group of three  $k$ 's, namely  $k = \{3, 4, 5\}$ . When we added 6-mers to the sequence representation, we found no improvement on the model performance, thus, we consider only  $k = \{3, 4, 5\}$ . Our set of  $k$ 's generates a concatenated vector of only 1344 positions. Once the concatenated vector of histograms is calculated, we proceed to the standardization of the data, which is also known as scaling (Angermueller et al. 2016; Graves 2012). After standardizing input vectors, histograms for each  $k$ -mer are padded with zero to the length of the largest histogram and stacked into a matrix of  $3 \times 1024$ .

## Implementation

The whole system is implemented in Python 3.6. Three main modules are considered for the Python code: download module, dataset module, and identification system module.

The download module takes advantage of the *urllib.request* library to download compressed FASTA files into the local disk, using the NCBI FTP endpoint. On the other hand, the dataset module loads FASTA files from local disk and computes their distributed  $k$ -mers based representation.

The identification system module is implemented using the Tensorflow deep learning framework – an advanced dataflow system that provides one of the most efficient implementations for RNNs (Angermueller et al. 2016). Based on the recurrent neural network architecture in (Liu et al. 2016) – that



processes protein sequences of up to 800 residues – our base classification system has a bidirectional GRU hidden layer with 128 units. The forward and backward final states are concatenated before applying a dropout of 0.5. The model uses only a fully connected layer that takes the dropout result as input. Softmax generates the predicted labels in the fully connected layer. To initialize the weights, we use a normal distribution with zero mean:  $w_{i_0} = \mathcal{N}(0, 0.1)$ .

Results from Neural Machine Translation (NMT) prove that intermediate hidden states from RNN units can significantly improve the performance of the models (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015; Rocktäschel et al. 2015). Such approach has been applied in protein family classification (Lee and Nguyen 2016) and precursor miRNA identification (Park et al. 2017). The attention mechanism (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015) creates a context vector  $\mathbf{c}_t$  from a weighted combination of intermediate hidden states  $h_t$ . We use the global attention with general content-based score in (Luong, Pham, and Manning 2015) to improve the base model.

Along with the three main modules described beforehand, two additional modules provide a user interface to the identification model: a command line interface (CLI) module and a web application implementation. The web user interface provides easy access and installation. It was built using Flask and React JS. A docker container packs all libraries and frameworks, along with the code, the user interfaces, and the trained neural network model. The container is publicly available at Docker Hub ([https://hub.docker.com/r/lelugom/wgs\\_classifier](https://hub.docker.com/r/lelugom/wgs_classifier)). In the repository, you will find instructions to start the container and easily access the model through the web interface. The code, the model, and a small test dataset are available in a GitHub repository ([https://github.com/lelugom/wgs\\_classifier](https://github.com/lelugom/wgs_classifier)), along with instructions for installation and replication of results.

## Results and Discussion

All the training and evaluation tests were performed using a computer with 8GB of RAM, an Intel Core i7-7700HQ CPU, and a GPU NVidia GeForce GTX 1050 with 2GB of dedicated RAM. The CPU has 4 physical cores and 8 threads through hyperthreading. Its base frequency is 2.8 GHz, while the Max Turbo frequency is 3.8 GHz. It also has a 6 MB cache with 64-byte cache line length. On the other hand, the GPU is part of the Pascal architecture. It has 640 CUDA cores arranged in 5 streaming multiprocessors. Its clock runs at 1354 MHz, with a boost of 1.3x. Its memory interface width is 128-bits.

Comparisons of tests in computational intelligence need statistical methods to ensure differences are statistically significant. We use the Wilcoxon signed rank test, which is a well-known nonparametric method. It is less affected by

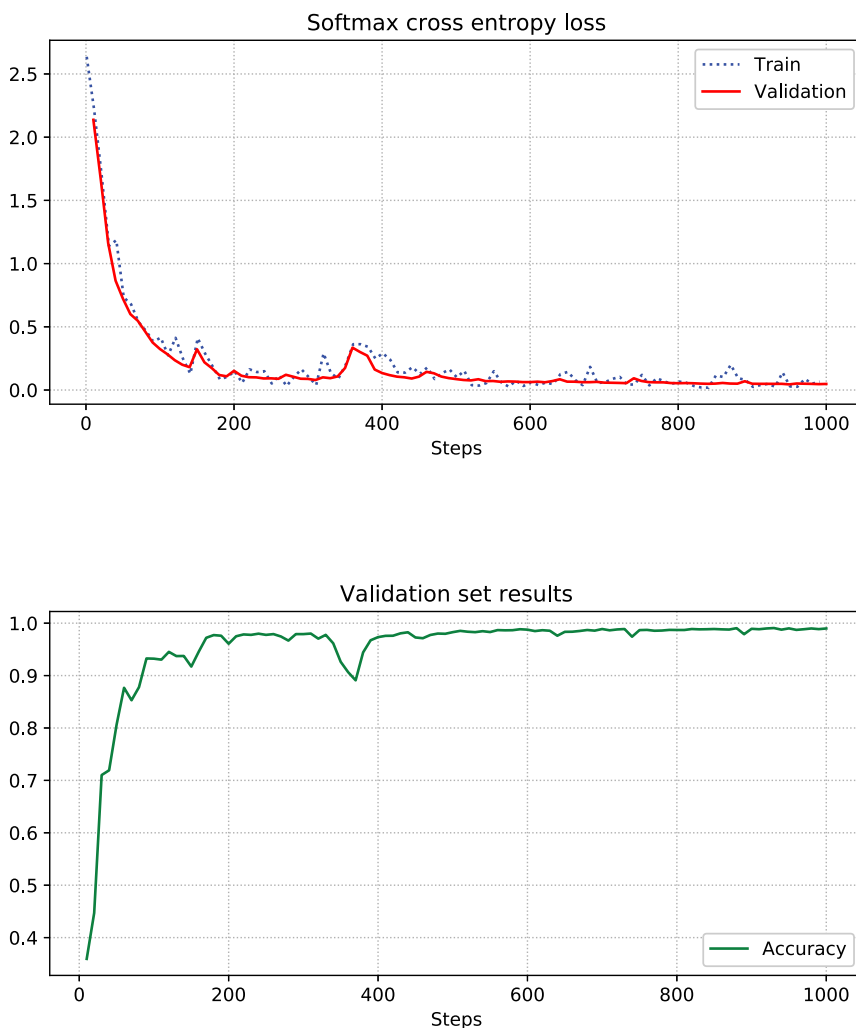


outliers and can be adapted to multiple data structures. This method tests if two samples represent different populations. Thus, it let us know if means from two samples have a significant difference (Derrac et al. 2011; Larsen and Marx et al. 2012). We use 5% as the minimum level of significance ( $p$ -value) for the tests throughout this section.

WWhole-genomesequences of bacteria are currently accessible through publicly available databases of biological sequences. GenBank was the selected database to gather an annotated set of bacterial DNA sequences, as it is part of the International Nucleotide Sequence Database Collaboration and contains WWhole-GenomeSequence entries (Benson et al. 2012). First, we obtained a recordset for WGS projects – in CSV format – from the NCBI website (<https://www.ncbi.nlm.nih.gov/Traces/wgs/?page=2&view=wgs&search=BCT>). Then, a python module performs text processing in the CSV file to extract the WGS project code, which is used to get the project URL. Project codes starting with NZ\_ do not have a valid HTML page, thus, they are ignored. The python module downloads the project HTML page from the URL and extracts the FTP address and the filename for the compressed FASTA file. Once the FTP address is extracted, the module proceeds with the download of the compressed FASTA file to the local disk, under a directory with the Taxonomic ID of the species.

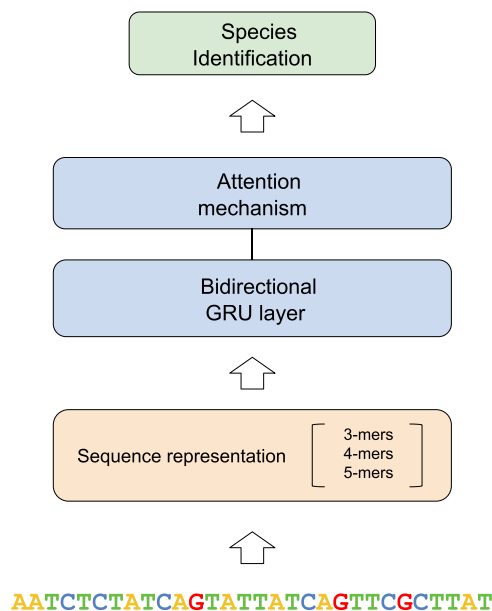
For the base recurrent neural network model, we use a threshold of one thousand sequences per species (14 species, see [https://hub.docker.com/r/lelugom/wgs\\_classifier](https://hub.docker.com/r/lelugom/wgs_classifier)). After randomly dividing the sequences into training (60%), testing (20%), and validation (20%) sets, the softmax ccross-entropyloss is minimized during the training process (Section 2). Learning rate was set to  $1e-3$  and batch size to 128. The Adam optimizer (Kingma and Ba 2014) minimized the loss function, with default parameters  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 1e^{-08}$ . Validation set results (Figure 2) include an accuracy of 99.0% at 950 steps. Precision was 99.8%, recall 99.93%, and F-score 99.86% at the same step. After finishing training at 3600 steps, results over the test set yielded an accuracy of 99.13%, precision 99.97%, recall 99.91%, and F-score 99.93%. No over-fitting was observed as can be seen when comparing training and validation loss curves. After analyzing the results of the training process with the validation set, we can observe that losses decrease the first 2800 steps and no further improvement is observed after 3600 steps. Thus, early stopping (Graves 2012) gives us a range between 2800 and 3600 steps for training the neural network. Our subsequent experiments used a step count in that range.

After performing 10-fold cross-validation, the base model generated an accuracy of  $0.98512 \pm 0.00798$  and f-score of  $0.98513 \pm 0.00799$  (14 species). Increasing the species coverage made the base model accuracy drop to  $0.76981 \pm 0.08550$  – with a minimum of one hundred sequences per species (111 species, see [https://hub.docker.com/r/lelugom/wgs\\_classifier](https://hub.docker.com/r/lelugom/wgs_classifier)).



**Figure 1.** Training curves for the bidirectional recurrent neural network, considering the first one thousand iterations.

The threshold of one hundred sequences per species (111 species) covers the critical, high, and medium priority groups of the WHO priority pathogens list (Tacconelli and Magrini 2017), although they cover only 4 out of 7 species in the Enterobacteriaceae family. Hence, we iterated over the base model, testing architecture and hyperparameter modifications to improve the recurrent neural network. We tested the addition of a dense module (Park et al. 2017), the addition of the attention mechanism (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015), a combination of the attention mechanism and a dense module, weighted loss for the imbalance in the dataset (Katevas et al. 2017), Glorot uniform initialization (Glorot and Bengio 2010), dropout wrappers (Gal and Ghahramani 2016), a Proximal Adagrad optimizer



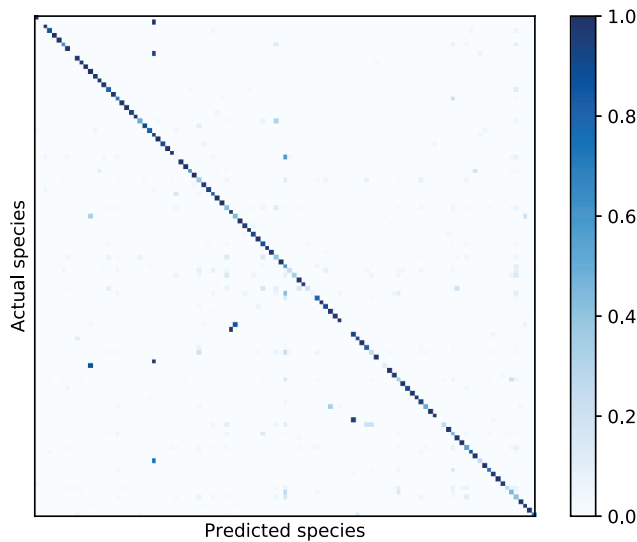
**Figure 2.** Architecture of the bacterial identification system. Starting from the encoding of the sequence using a k-mers based representation, the model then pass the sequence through a BRNN with an attention mechanism. A final softmax layer finds the label to identify the bacterial species.

**Table 1.** Variation of accuracy against minimum sequences per species. Comparison between the base model and the final model.

Seqs.	Base model	Final model	<i>p</i> -value
1000	0.9775 ± 0.0159	0.9945 ± 0.0028	0.005
500	0.9425 ± 0.0449	0.9871 ± 0.0126	0.027
250	0.8660 ± 0.0314	0.9503 ± 0.0046	0.027
100	0.7698 ± 0.0855	0.8903 ± 0.0041	0.006

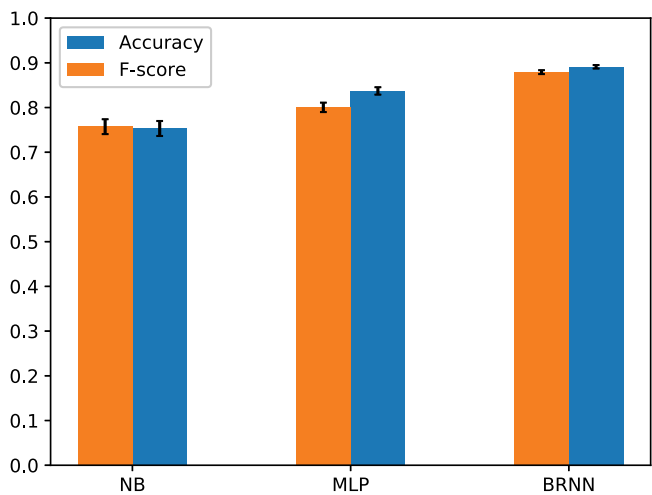
(Angermueller et al. 2016), layer size changes, and various learning rates. Overall, the attention mechanism generated the major improvement.

Our final model has a bidirectional GRU layer with 128 units, a global attention mechanism (Bahdanau, Cho, and Bengio 2015) – with a dense layer to concatenate the context vector to the final output state – and a dense layer for classification. Dropout of 0.5 provides regularization for the two dense layers. Also, the Adam optimizer with a learning rate of 1e-3 minimizes the softmax ccross-entropyloss function. A comparison between the base model and the final model for different thresholds of minimum sequences appear in Table 1. All results are statistically significant. We observed a substantial improvement against the base model in all the scenarios, although the lower the threshold, the higher the value of the final model improvement. With a threshold of one hundred sequences per species, the model achieves the broadest coverage. A confusion matrix with this threshold (Figure 3) let us know the main limitation of the model: It is clear from the matrix diagonal how the model struggles to classify certain species, impacting the overall accuracy of the model.



**Figure 3.** Confusion matrix for the final classification model covering 111 species. The accuracy of the model is 89.632%.

Finally, we select alternative methods for results comparison. Naive Bayes (NB) and Multilayer Perceptron (MLP) (Alpaydin 2014) are two classification models which provide a reference to compare our identification system results. It is important to note that we cannot use a matrix representation ( $3 \times 1024$ ) for Naive Bayes and MLP models. Instead, we encode data with the concatenated representation ( $1 \times 1344$ ) that holds  $k$ -mers of  $k = \{3, 4, 5\}$ . We performed 10-fold cross-validation. The dataset has a threshold of at least 100 sequences per species (111 species, see [https://hub.docker.com/r/lelugom/wgs\\_classifier](https://hub.docker.com/r/lelugom/wgs_classifier)). Results show that our classification model outperforms other machine learning methods (Figure 4, Table 2). There is a considerable increase in



**Figure 4.** Validation of results comparing Naive Bayes (NB), Multilayer Perceptron (MLP), and our identification model.

**Table 2.** Comparison between Naive Bayes (NB), Multilayer Perceptron (MLP), alternative RNN configurations, and our identification model (BRNN GRU).

Model	Metric	Value
NB	Accuracy	0.75313 $\pm$ 0.01675
	Precision	0.83775 $\pm$ 0.00461
	Recall	0.69128 $\pm$ 0.02774
	F-score	0.75719 $\pm$ 0.01662
MLP	Accuracy	0.83713 $\pm$ 0.00829
	Precision	0.79688 $\pm$ 0.01054
	Recall	0.80402 $\pm$ 0.01308
	F-score	0.80040 $\pm$ 0.01055
RNN GRU	Accuracy	0.88896 $\pm$ 0.00367
	Precision	0.87889 $\pm$ 0.00293
	Recall	0.87354 $\pm$ 0.00491
	F-score	0.87620 $\pm$ 0.00354
BRNN LSTM	Accuracy	0.89690 $\pm$ 0.00153
	Precision	0.89123 $\pm$ 0.00594
	Recall	0.88329 $\pm$ 0.00192
	F-score	0.88724 $\pm$ 0.00334
BRNN GRU	Accuracy	0.89107 $\pm$ 0.00392
	Precision	0.88068 $\pm$ 0.00515
	Recall	0.87767 $\pm$ 0.00486
	F-score	0.87917 $\pm$ 0.00436

performance when taking into account all the metrics used for model characterization. All differences were statistically significant ( $p \leq 0.5$ ) against validation methods. Other recurrent neural models provided additional means for validating results. Cross-validation tests with a model based on LSTM units and a unidirectional architecture helped us compare the model performance against common recurrent architectures. For the unidirectional architectures, we doubled the number of units to perform a fair comparison to our bidirectional identification model. Results in Table 2 show equivalent outputs for the alternative recurrent neural network configurations.

## Conclusions

Although a number of methods are available for the essential task of bacteria identification – including mass spectrometry, pairwise sequence comparison, and microscopic morphology – recurrent neural networks represent an automatic classification method which does not require any manual feature extraction. They are easily updated through the retraining of the model. Our identification system exploits the vast amounts of genomic information available in GenBank to infer the species of a given bacterial whole-genome sequence. GenBank provides the samples to train and test the prediction capabilities of our neural network architecture. The model has the potential to benefit diverse areas, such as pathology, microbiology, experimental biology, food and water industries, and evolutionary studies.

A distributed representation provides an excellent encoding for the bacterial genomic information in a low dimensional space. This is an important

aspect considering the high dimensionality and sparsity of one-hot encoding sequence representations. The combination of two or more k-mer lengths gives context to the distributed representation. Context takes advantage of positional information, a crucial aspect in biological sequences. From applications in Natural Language Processing, the additional context proved to be useful for our identification model efficiency. On top of that, the distributed representation does not require the assumption that nodes in the FASTA files are ordered.

Our bidirectional GRU model has a bidirectional GRU layer with 128 units, a global attention mechanism (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015) – with a dense layer to concatenate the context vector to the final output state – and a dense layer for classification. The model generates an accuracy of  $0.89107 \pm 0.00392$  for 111 spices. Also, it outperformed other machine learning methods in the classification of bacterial wwhole-genomesequences. Results show better numbers in all the metrics used to characterize the model (accuracy, precision, recall, and balanced F-score). Alternative recurrent neural network architectures – standard GRUs, bidirectional LSTMs – provided equivalent results, with no statistically significant differences.

*Future work* As more curated samples of wwhole-genomesequences are available at GenBank, the classification system can improve in two aspects. First, the number of species with at least o100 sequences will increase, growing the coverage of the recurrent neural network. Secondly, the accuracy of the system has the potential to improve because most of the species that the model struggles to identify have low sequence counts. Also, proper data augmentation techniques could increase the number of sequences for training. Those augmentation techniques should avoid direct modifications of the nucleotides in the wwhole-genomesequences. Nucleotide changes insert mutations in the sequences. Mutations in core parts of the sequence express alterations in vital functions, which generate a sequence that cannot represent a living organism.

## References

- Alpaydin, E. 2014. *Introduction to machine learning*. Cambridge, MA, US: MIT press.
- Angermueller, C., T. Pärnamaa, L. Parts, and O. Stegle. 2016. Deep learning for computational biology. *Molecular Systems Biology* 12 (7):878. doi:10.15252/msb.20156651.
- Bahdanau, D., K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. *arXiv Preprint arXiv:1409.0473*.
- Benson, D. A., M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. 2012. Genbank. *Nucleic Acids Research* 41 (D1):D36–D42. doi:10.1093/nar/gks1195.
- Bosco, G. L., and M. A. Di Gangi (2016). Deep learning architectures for DNA sequence classification. *International Workshop on Fuzzy Logic and Applications*, pp. 162–71. Springer.

- Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv Preprint arXiv:1406.1078*.
- Cole, J. R., Q. Wang, J. A. Fish, B. Chai, D. M. McGarrell, Y. Sun, C. T. Brown, A. Porras-Alfaro, C. R. Kuske, and J. M. Tiedje. 2013. Ribosomal database project: Data and tools for high throughput rRNA analysis. *Nucleic Acids Research* 42 (D1):D633–D642. doi:[10.1093/nar/gkt1244](https://doi.org/10.1093/nar/gkt1244).
- Dahl, G. E. (2015). *Deep learning approaches to problems in speech recognition, computational chemistry, and natural language text processing*. Ph. D. thesis, University of Toronto.
- Derrac, J., S. Garca, D. Molina, and F. Herrera. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1 (1):3–18. doi:[10.1016/j.swevo.2011.02.002](https://doi.org/10.1016/j.swevo.2011.02.002).
- Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth. 1996. From data mining to knowledge discovery in databases. *AI Magazine* 17 (3):37.
- Gal, Y., and Z. Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, 1019–27.
- Garrity, G. M., and C. S. Kraft. 2016. A new genomics-driven taxonomy of bacteria and archaea: Are we there yet? *Journal of Clinical Microbiology* 54 (8):1956–63. doi:[10.1128/JCM.00200-16](https://doi.org/10.1128/JCM.00200-16).
- Giang Nguyen, N., V. A. Tran, D. L. Ngo, D. Phan, F. R. Lumbanraja, M. R. Faisal, B. Abapihi, M. Kubo, and K. Satou. 2016. DNA sequence classification by convolutional neural network. *Journal of Biomedical Science and Engineering* 9 (9):280–86.
- Glorot, X., and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics, Sardinia, Italy, pp. 249–56.
- Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. Cambridge, MA, US: MIT Press.
- Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks*. Ph. D. thesis.
- Hasman, H., D. Saputra, T. Sicheritz-Ponten, O. Lund, C. A. Svendsen, N. Frimodt-Møller, and F. M. Aarestrup. 2013. Rapid whole genome sequencing for the detection and characterization of microorganisms directly from clinical samples. *Journal of Clinical Microbiology* 52 (1):139–46. doi:[10.1128/JCM.02452-13](https://doi.org/10.1128/JCM.02452-13).
- Katevas, K., I. Leontiadis, M. Pielot, and J. Serrà. 2017. Practical processing of mobile sensor data for continual deep learning predictions. *arXiv Preprint arXiv:1705.06224*.
- Kimothi, D., A. Soni, P. Biyani, and J. M. Hogan. 2016. Distributed representations for biological sequence analysis. *arXiv Preprint arXiv:1608.05949*.
- Kingma, D. P., and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv Preprint arXiv:1412.6980*.
- Larsen, R. J., and M. L. Marx. 2012. *An introduction to mathematical statistics and its applications*. Fifth ed. Boston, MA, US: Pearson Education.
- Lee, T. K., and T. Nguyen (2016). Protein family classification with neural networks. <https://cs224d.stanford.edu/reports/LeeNguyen.pdf>.
- Lipton, Z. C., J. Berkowitz, and C. Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv Preprint arXiv:1506.00019*.
- Liu, H., Z. Wang, B. Shen, and F. E. Alsaadi. 2016. State estimation for discrete-time memristive recurrent neural networks with stochastic time-delays. *International Journal of General Systems* 45 (5):633–47. doi:[10.1080/03081079.2015.1106731](https://doi.org/10.1080/03081079.2015.1106731).
- Lodish, H., A. Berk, S. L. Zipursky, P. Matsudaira, M. Krieger, and J. Darnell. 2004. *Molecular cell biology*. Fifth ed. New York, US: W.H. Freeman and CO.



- Luong, M.-T., H. Pham, and C. D. Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv Preprint arXiv:1508.04025*.
- Martens, J., and I. Sutskever (2011). Learning recurrent neural networks with hessian-free optimization. Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, pp. 1033–40. Citeseer.
- Min, S., B. Lee, and S. Yoon. 2016. Deep learning in bioinformatics. *arXiv Preprint arXiv:1603.06430 abs/1603.06430*.
- Mohamad, N. A., N. A. Jusoh, Z. Z. Htike, and S. L. Win. 2014. Bacteria identification from microscopic morphology: A survey. *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)* 3 (1):2319–1015.
- Murphy, K. P. 2012. *Machine learning: A probabilistic perspective*. Cambridge, MA, US: The MIT Press.
- Ng, P. 2017. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv Preprint arXiv:1701.06279*.
- Nguyen, N. G., V. A. Tran, D. L. Ngo, D. Phan, F. R. Lumbanraja, M. R. Faisal, B. Abapihi, M. Kubo, and K. Satou. 2016. DNA sequence classification by convolutional neural network. *Journal of Biomedical Science and Engineering* 9 (5):280. doi:10.4236/jbise.2016.95021.
- Pais, F. S.-M., P. De Cássia Ruy, G. Oliveira, and R. S. Coimbra. 2014. Assessing the efficiency of multiple sequence alignment programs. *Algorithms for Molecular Biology* 9 (1):4. doi:10.1186/1748-7188-9-4.
- Park, S., S. Min, H.-S. Choi, and S. Yoon. 2017. Deep recurrent neural network-based identification of precursor microRNAs. In *Advances in neural information processing systems*, Long Beach, CA, USA, pp. 2895–904.
- Pevsner, J. 2015. *Bioinformatics and functional genomics*. Hoboken, NJ, USA: John Wiley & Sons.
- Rizzo, R., A. Fiannaca, M. La Rosa, and A. Urso (2015). A deep learning approach to DNA sequence classification. International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics, pp. 129–40. Springer.
- Rocktäschel, T., E. Grefenstette, K. M. Hermann, T. Kočiský, and P. Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv Preprint arXiv:1509.06664*.
- Schmidt, B., and A. Hildebrandt (2017). Next-generation sequencing: Big data meets high performance computing. *Drug Discovery Today*.
- Scholz, M. B., -C.-C. Lo, and P. S. Chain. 2012. Next generation sequencing and bioinformatic bottlenecks: The current state of metagenomic data analysis. *Current Opinion in Biotechnology* 23 (1):9–15. doi:10.1016/j.copbio.2011.11.013.
- Singhal, N., M. Kumar, P. K. Kanaujia, and J. S. Viridi. 2015. [maldi-tof mass spectrometry: An emerging technology for microbial identification and diagnosis]. *Frontiers in Microbiology* 6. doi: 10.3389/fmicb.2015.00791.
- Tacconelli, E., and N. Magrini (2017). Global priority list of antibiotic-resistant bacteria to guide research, discovery, and development of new antibiotics. [http://www.who.int/mediacines/publications/WHO-PPL-Short\\_Summary\\_25Feb-ET\\_NM\\_WHO.pdf](http://www.who.int/mediacines/publications/WHO-PPL-Short_Summary_25Feb-ET_NM_WHO.pdf).
- Tan, P.-N., M. Steinbach, A. Karpatne, and V. Kumar. 2018. *Introduction to Data Mining*. Second ed. London, UK: Pearson Education.
- Varghese, N. J., S. Mukherjee, N. Ivanova, K. T. Konstantinidis, K. Mavrommatis, N. C. Kyrpides, and A. Pati. 2015. Microbial species delineation using whole genome sequences. *Nucleic Acids Research* 43 (14):6761–71. doi:10.1093/nar/gkv657.
- Webb, S. 2018. Deep learning for biology. *Nature Technology Features* 554:555–57.