UNIVERSITY OF PRINCE EDWARD ISLAND

DeLUCS-F: Deep Learning of Unsupervised Clusters of DNA Sequences using Sequence Fragments

by

Dylan Lewis

A thesis submitted in partial fulfillment for the degree of Honours in Computer Science

in the
Faculty of Science
School of Mathematical and Computational Sciences

April 2024

PERMISSION TO USE HONOURS PAPER

Title of paper: DeLUCS-F: Deep Learning of Unsupervised Clusters of DNA Sequences using Sequence Fragments

Name of Author: Dylan Colby Pratap Lewis

Department: School of Mathematical and Computer Sciences

Degree: Mathematics

Year: 2024

Name of Supervisor(s): Dr. Gurjit S. Randhawa, Dr. Paul Sheridan

Signature [of author]:

Address [Department]: 550 University Avenue, Charlottetown, PE C1A 4P3

Date: May 31, 2024

Declaration of Authorship

I, Dylan Lewis, declare that this thesis titled, 'DeLUCS-F: Deep Learning of Unsupervised Clusters of DNA Sequences using Sequence Fragments' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: April 25, 2024

UNIVERSITY OF PRINCE EDWARD ISLAND

Abstract

Faculty of Science
School of Mathematical and Computational Sciences

Honours in Computer Science

by Dylan Lewis

This study addresses the issue of the use of synthetic data by the DeLUCS pipeline, and investigates the replacement of artificially generated mimic sequences with biologically relevant data. This study introduces a novel pipeline, DeLUCS-F (**De**ep **L**earning of **U**nsupervised **C**lusters of DNA **S**equences using Sequence **F**ragments), which builds upon the DeLUCS pipeline by increasing the trustworthiness of its predictions, and increasing its performance. This is accomplished through the use of sequence fragments, used in place of artificial data for the sake of data augmentation. DeLUCS is considered the state-of-the-art in the application of unsupervised deep learning to the problem of taxonomic assignment. The main advantages of such a model are that 1) it can be applied to a variety of genomic data, including DNA, mitochondrial DNA, DNA segments, and individual genes, and was tested on vertebrate, bacterial, and viral data; 2) it is alignment-free, and thus can overcome various obstacles faced by alignment-based solutions (namely, that alignment-based solutions are prohibitively slow); 3) it involves deep learning, which has the ability to learn more complex patterns than classical machine learning algorithms can learn from big datasets; and 4) that it is unsupervised, meaning that it can be used when labels are unavailable, or can not be trusted, due to either the potential presence of errors, or the ever-changing nature of taxonomy. However, it relies on data augmentation to achieve its high performance (relative to other unsupervised methods), which is problematic as it is a "black box" model due to its use of both deep neural networks and a majority voting scheme, thus raising questions regarding whether the patterns it learns are from the true data, or the artificial data. This is especially concerning in the field of genomics, where it is still largely unknown how mutations can change the function of entire sequences, and thus artificial data that is biologically viable cannot be reliably generated. Thus, while DeLUCS achieves good results, its predictions are called into question as a result of the fact that it could be learning from biologically irrelevant data. This study describes relevant information in the field of taxonomic assignment and the use of synthetic data, the contributions made by this novel pipeline, the comparison of the results of classical unsupervised methods, DeLUCS, and DeLUCS-F, and presents potential future research.

Acknowledgements

This research was made possible by the assistance and through the support of my supervisors, Dr. Gurjit S. Randhawa and Dr. Paul Sheridan.

This research was enabled in part by the computational resources provided by ACENET (acenet.ca) and the Digital Research Alliance of Canada (alliancecan.ca).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, taxonomic assignment has been essential for the sake of allowing us to better react to emerging diseases [1–3]. Taxonomic assignment refers to assigning organisms based on their taxonomic ranking (e.g., Domain, Kingdom, Phylum) and to the organization and structuring of these taxa, and gives a basis for the comparison between organisms. For example, given a group containing a lion (*Panthera leo*), a red fox (*Vulpes vulpes*), a grey wolf (*Canis lupus*), and a coyote (*Canis latrans*), the latter three are closer to each other than to a lion, as they are of the same family (Canidae); similarly, the latter two are closer to each other than to a red fox, as they are of the same genus (*Canis*). The field of taxonomic assignment has been evolving at an increasing pace [4], in no small part due to the rapid explosion of available data, such as in the case of SARS-CoV-2, where it quickly became the most sequenced virus to date [5]. In the cases of severe disease outbreaks, being able to rapidly assign a given virus to a taxa, given its genetic information, can allow us to monitor and respond to the evolution of strains [2, 3, 5], and is crucial for allowing us to quickly both strategize appropriate public health measures, and determine the best treatment options [1, 2]. However, many of the existing solutions for taxonomic ranking suffer from poor performance when faced with large amounts of data [6].

1.1 The DeLUCS pipeline

In the 2022 study by Millán Arias et al. [7], the DeLUCS pipeline (**De**ep **L**earning for the **U**nsupervised **C**lustering of DNA **S**equences) was introduced, which can leverage this ever increasing amount of data through the use of deep learning, and which performs better (in terms of both the accuracy achieved on a given amount of data and how it handles data with complex relationships [6] [8, p. 34, 175, 319] [9]) than classical methods (e.g., K-Means++, Gaussian Mixture Models) as a result of being exposed to more data [6]. It is an alignment-free approach, meaning that it does not require sequences to share common ancestry nor be of similar lengths [7], nor does it require the time-consuming (and computationally expensive) process of attempting to find conserved contiguous strings of letters between sequences (accounting also for insertions and deletions) [10]. It deals with the problem of taxonomic assignment from a clustering (rather than classification) approach, meaning that it does not require the data to be labelled (thus it is not thrown off by incorrect labels) [7]. It outperforms classic clustering methods (K-Means++ and Gaussian Mixture Models) in this task of taxonomic assignment [7]. It can handle a variety of genetic data (including mtDNA, DNA segments, individual genes, and whole genomes), and as of 2022, it was used to cluster the largest dataset to date (consisting of over one billion total base pairs) [7]. The DeLUCS pipeline is as follows:

1. Cleaning the data:
   (a) Each sequence is cleaned (i.e., all non-A,C,T,G letters are removed), and in the case of evaluating the pipeline (i.e., labels are available), is associated with a label.
2. Preprocessing the data, and generating the mimics:
   (a) For each sequence, mimic sequences are generated by performing transitions and transversions (i.e., substituting a given base pair with a different base pair) on the original sequences. Then, for both the original sequence and the mimic sequence, the normalized *k*-mer count is calculated—where a *k*-mer refers to a string of *k* letters from the alphabet C,T,G,A (e.g., All the 2-mers of CTGCT would be CT, TG, GC, and CT. The 2-mer counts would be [2,1,1,1], and the normalized 2-mer counts would thus be [0.5, 0.25, 0.25]).
3. Training (and optionally, evaluating) the neural networks:
   (a) For each sequence, the training data will be filled with pairs of the original sequence's normalized *k*-mer count, and the mimic sequences' normalized *k*-mer count (thus, if there are to be three mimics, the training data will have three pairs of original and mimic sequences for each sequence). Ten neural networks are independently trained using this data, and the cluster assignments of these networks are associated with a ground truth using the Hungarian Algorithm; that is, it is made so that as much as possible, the clusters are in agreement between networks, either using labels (in the case of evaluation), or using only the cluster assignments predicted by the other networks to generate a consensus (in the case of deployment). Then, for each sequence, the majority vote of the ten networks is taken as the final prediction.
   (b) In the case of evaluating the network (i.e., when labels are used), the optimal mapping between the clusters and the labels (found by the Hungarian Algorithm) is used to calculate the accuracy (i.e., the sum of all correct predictions, divided by the total number of sequences).

The second step described above, where the mimic sequences are generated, is the main focus of this study, due to the issues of transparency and trustworthiness. With respect to transparency, DeLUCS takes the majority vote of ten neural networks; neural networks are considered a “black box”, in that it is very difficult to understand the reasoning behind their predictions [9]; additionally, the use of a majority voting scheme further decreases the interpretability of the model. This then leads to the issue of trustworthiness, where it cannot be trusted whether the model is making predictions based on biologically relevant (real) data or the potentially biologically irrelevant (synthetic) data that was used to augment the original dataset, and that is fed to the neural networks.

The issue of synthetic data ties into the fact that one of the main advantages of DeLUCS over other alignment-free methods is that it is an unsupervised method, and thus can be used when labels are unavailable. Equally, its predictions are unaffected by the ever-changing and potentially erroneous taxonomic labels; that is, instead of learning to pair a datapoint with a given label based on its attributes, it instead clusters together data based on hidden patterns within the data. However, due to the lack of explainability for the predictions of the model, it is difficult to discern whether the model is learning patterns from the original data, or from the artificial data. In the work of Solis-Reyes et al. [11], where a machine learning-based model (which was also alignment-free, used *k*-mers, and had the end result of predicting taxonomic rank) was trained on an augmented dataset to improve its performance, had also succeeded in training a different machine-learning-based model which was able to perfectly distinguish between the genuine and synthetic data, which raised into question the use of artificial data (and specifically, in the field of genomics, where our ability to generate meaningful synthetic DNA sequences is limited by our understanding of the language of DNA). Thus, while DeLUCS outperforms other unsupervised clustering methods, the fact that it relies on synthetic data to do so is problematic, as it jeopardizes one of the main advantages of unsupervised methods: assigning data into meaningful clusters based on patterns intrinsic to the data.

As an example of how the use of synthetic data is an issue here, consider the fact that it is generated in the DeLUCS pipeline by performing transitions and transversions (i.e., changing base pairs) on the original sequences. However, these transitions/transversions could cause changes in a sequence which, in nature, would not be observed (such as the insertion or deletion of a start codon, creating a sequence which is no longer biologically viable). This means that the use of synthetic data could cause the model to learn patterns in data which have no biological relevance. Thus, there is a strong need for the replacement of synthetic data in the DeLUCS pipeline with a biologically relevant alternative.

1.2 Purpose and research goals

This study presents DeLUCS-F (**De**ep **L**earning for the **U**nsupervised **C**lustering of DNA **S**equences using Sequence **F**ragments), a modified version of the DeLUCS pipeline, which removes the original pipeline's dependency on artificial data, by using sequence fragments instead of mimic sequences; thus, while the model remains a black box due to the use of neural networks and a majority voting scheme, it can be certain that it is learning patterns intrinsic to the data itself, rather than potentially learning patterns from the synthetic data.

The goals of this study are to investigate:

1. Whether the mimic sequences of DeLUCS can be replaced with a biological relevant alternative; in this case, sequence fragments.
2. The changes in performance between DeLUCS and DeLUCS-F as a result of changing from mimic sequences to sequence fragments, across the datasets used in the DeLUCS study.
3. The effect of fragment length on the performance of DeLUCS-F on each dataset.
4. Whether the change from mimic sequences to sequence fragment caused changes in how the pipeline responds to other hyperparameter values (namely, the *k*-mer length and the number of fragments).

This study employs the same datasets as used in the DeLUCS study, as this allowed for not only a direct comparison of DeLUCS-F with DeLUCS (using the hyperparameters and selection of data that were found most appropriate in the DeLUCS study), but also with the other algorithms against which DeLUCS had been compared in the DeLUCS study [7]. The datasets consist of: (I) complete mitochondrial genomes of vertebrates; (II) segments of bacteria genomes; (III) viral genes and complete viral genomes. In (I), a decision-tree approach had been taken, whereby the cluster with the most available sequences was chosen for further clustering, allowing for the assessment of performance at each taxonomic level, ranging from clustering amongst classes to clustering amongst families [7]. In (II), 3200 sequences averaging 433613 base pairs were included, analyzing performance when dealing with massive amounts of data. In (III), which consist of short (ranging from 1345 to 10991 base pairs) sequences, the performance can be compared against alignment-based solutions, as they are applicable to this dataset. The modification to the DeLUCS pipeline introduced in this study increased the trustworthiness of the model, as, despite the fact that DeLUCS-F is equally a black box, it relies only on real data to generate predictions regarding the data. It was found that this modification achieved the same or better performances for all but one of the 11 original datasets. A hyperparameter search was then performed to visualize how the change in the pipeline may have affected the relationship between the hyperparameters and performance (consisting of 270 tests on the worst scoring dataset) and found only a slight increase in performance with slightly different settings. Moreover, the relationship between the hyperparameters that was found was expected (as in, an performance was positively correlated with the length of fragments used, the resolution of the *k*-mers, and the number of fragments used; this is analogous to the findings in [7], that performance was positively correlated with the resolution of the *k*-mers, and the number of mimics used).

Future works should consider hyperparameter tuning, changing the nature of the neural networks (such as replacing the networks, which are sequential, with functional networks), and using DeLUCS-F to assess the accuracy of taxonomic labels using a decision-tree approach to clustering (note that this is in a different sense than what had been described in [7] by a decision-tree approach). Furthermore, more meaningful hyperparameter searches could be performed by performing multiple random searches rather than grid searches, iteratively narrowing the search upon regions that show promise. A relationship was found between the minimum fragment length needed to obtain the optimal accuracy for a given dataset, and the number of clusters used; this could be investigated to find whether this pattern holds in other datasets, and if so, the reason for such a relationship. Finally, all tests were conducted with all random seeds fixed to 0 to ensure perfect reproducibility; further studies should investigate the results of DeLUCS-F when the seeds are random, by averaging the results across ten trials for each set of hyperparameters.

The rest of this study is organized as follows. Chapter 2 lays out the ways in which alignment-free approaches outperform alignment-based approaches, the advantages of using deep neural networks in fields with large amounts of data, the downsides of supervised learning, and finally, how DeLUCS contributes to the field of taxonomic assignment, along with the downsides of the use of artificial data, especially in the context of the DeLUCS pipeline. Chapter 3 discusses this study's contributions to the DeLUCS pipeline, the methodology of the original pipeline and how DeLUCS-F differs in that respect, and the datasets used. Chapter 4 covers the various tests that were run in this study: how DeLUCS-F compares to the original DeLUCS pipeline along with other relevant methods, the analysis of the performance of DeLUCS-F on each dataset in relation to the relative fragment length used, and the results of a hyperparameter grid search on one of the datasets, which was conducted in order to investigate the relationship between the hyperparameters and the performance of DeLUCS-F. Chapter 5 considers potential reasons for the differences between the results of this study and the results of the original DeLUCS pipeline, and potential future works. Chapter 6 summarizes the results of this study, and the importance of this study's modifications to this model for the field of taxonomic assignment.

Chapter 2

Background Information

2.1 Methods for taxonomic assignment and their issues

The traditional approach to taxonomic assignment has been based on sequence alignment: that is, aligning an unknown sequence with many different known sequences, one-by-one, in the hopes of finding a match [10]. As of 2018, this was still the most widely used approach for sequence comparison [12]. However, there are numerous issues with this approach, most notably:

- Sequences of the same taxonomic rank may vary at single base pairs (via an insertion or deletion), and at chunks (via shuffling and recombination); thus, an alignment-based approach must be coupled with a scoring scheme in order to determine which sequences should be considered a match. However, the weighting for this scheme is not based on ground truths, nor is necessarily the same between studies, and slight differences in this scheme can lead to large differences in results [10].
- The optimal implementation of alignment-based approaches is based on dynamic programming, and thus the computational time complexity scales exponentially with the length of the sequence [13].
- An alignment-based approach is indifferent to the reasons for differences between sequences, and thus does not account for long-range interactions and shuffling events that do not result in change of function [13].
- Noncoding regions often vary between similar organisms (as differences here do not result in differing functions) and are thus interpreted as dissimilarity by alignment-based approaches [12].

In contrast, alignment-free approaches do not suffer from these issues. However, alignment-free methods, due to the fact that they are indifferent to the original order of the sequence [10], lose potentially useful information, and thus risk being at a disadvantage, in terms of discriminatory power. In terms of alignment-free methods, deep learning-based approaches stand to overcome this disadvantage, in comparison to other machine learning based approaches, due to their ability to learn complex patterns within even high-dimensional datasets [14], and thus have the potential to compensate for this loss of information.

Neural network-based approaches, and thus deep learning (where deep learning refers to neural networks with multiple hidden layers), stand to benefit from large amounts of data [6], giving them an advantage in the field of taxonomic assignment, where new sequences are being processed at an increasing rate [4]. Neural networks (and machine learners in general) can be divided into two groups: supervised, and unsupervised, where for the former, the network receives labelled training data, allowing it to learn the connections between data and the associated label. So far, deep learning in genomics has been most successful with respect to supervised approaches [15], which is reasonable, as supervised approaches are typically more accurate in general, as accuracy is defined in terms of the ability to correctly match observations to the desired label, and the fact that they are trained on labelled data allow them to search for patterns that directly map the features to the desired labels [16, 17]. However, labels are not always available (of the estimated 8.7 million (± 1.3 million) Eukaryotes on Earth, only 1.5 million eukaryotes have been classified [18]), and even when they are, taxonomy is constantly undergoing revisions [2, 3, 7, 19, 20], as there is no ground truth for taxonomic labels [7]. Additionally, labelling is a time-consuming process, is subject to human error (e.g., mistakes in data entry, or resulting from limited available information) [7], as well as errors resulting from shifts in the ways in which genetic material is sequenced [4]. Given that the success of supervised learners hinges on the correctness of the data labels, these errors and lack of ground truth for taxonomic labels are particularly problematic; thus, unsupervised approaches offer promise in the fact that they do not require labelled data.

As covered in [7], unsupervised pipelines in the field of genomics have typically relied on classical machine learning methods; for example, [21–25] use K-Means, while [26] uses Gaussian Mixture Models. However, classical clustering methods, while appropriate for low-dimensional data, perform poorly (in terms of both accuracy and efficiency) when clustering high-dimensional datasets, due to the curse of dimensionality [9]. Thus, there is a necessary tradeoff, when using classical clustering methods, between using enough features to represent the data adequately, and keeping the dimensionality low enough to ensure high performance. However, neural networks, which remap the features into different spaces (where at each layer, the features are mapped into a new space), do not suffer from these issues, and can thus learn more complex, higher-level patterns from the input data [9].

2.2 The DeLUCS pipeline and its issues

The DeLUCS pipeline, presented by Millán Arias et al., in 2022 [7], combines unsupervised clustering with deep learning, and can thus learn more complex patterns in the data than classic clustering methods. Additionally, as an unsupervised model, rather than predicting which datapoints correspond to which label based on various features, are responsible for using these features to find essential patterns within the data [15]. However, learning such patterns is most useful when it is clear how these patterns are found, and how they contribute to the decision-making process. This is an inherent issue for deep learning, and neural networks in general, as their processes and decisions are mostly hidden within a "black box" [9]. However, this issue is further compounded when they are trained using artificially generated data, as it is no longer clear whether the model learned patterns within the true data, or the artificial data. While this would not be problematic if artificial data could perfectly model true data, this is not the case, at least for genomics. In the study by Solis-Reyes et al. [11], the pipeline was run on a synthetic dataset (where artificial DNA sequences were created through the use of INDELible [27], which would create a synthetic sequence by performing insertions, deletions, and substitutions on an inputted sequence), with the advantage being that the ground truth for the labels existed, as the data was created to model known taxonomy. However, that study also found that a machine learner was able to perfectly distinguish between synthetic and natural data, which raised concerns about the use of synthetic data [11].

In the case of DeLUCS, it generates artificial data (in the form of "mimic" sequences) in order to assist the learning process. Each mimic sequence is generated by performing some combination of transitions and transversions (i.e., substituting base pairs) throughout a real sequence (i.e., the artificial data here is created by performing modifications on the real data). However, a given transition/transversion could cause a codon to become a start or stop codon, or vice-versa. In short, the start and stop codons are what separate the coding parts of a genome (what get transcribed to RNA, which then may be translated to proteins) from the noncoding parts (similar to how certain symbols denote the beginning and ending of comment blocks in a script). In the worst case, the addition/removal of start/stop codons could cause an entire cell to no longer be able to survive/replicate. Thus, while the generation of mimic sequences in the DeLUCS pipeline is biologically inspired [7], further rules need to be considered to prevent the simulation of sequences that would never be found in nature, in order to ensure that the artificial sequences are reliable data from which the neural networks can learn. This complication, and the potential increase in computation time required to generate sequences that would still be biologically viable using transitions/transversions, necessitates the need for a biologically valid and quick-to-compute alternative to these mimic sequences.

The contribution of this study to the DeLUCS pipeline was the replacement of the artificially generated mimic sequences with sequence fragments. This was with the intention of making the results of the pipeline more trustworthy, given the lack of transparency of such complex models. While the main goal of this study was to increase the extent to which the clusters discovered by DeLUCS modelled true patterns within the data, the replacement of mimic sequences with sequence fragments modification provided a less computationally expensive alternative, that, as shown in the results of this study, yields a higher performance.

Chapter 3

Materials and methods

To put into context the DeLUCS-F pipeline, this chapter first describes the methodology of the original DeLUCS pipeline, and then describes the differences with the DeLUCS-F pipeline. The various datasets are then described; these are the datasets used in the DeLUCS study [7], which were used to compare the performances of the two pipelines.

All tests were run on the Narval cluster of Compute Canada, using Python 3.10.2. Further details regarding the exact setup, and example commands necessary to replicate results, are described in Appendix B.

3.1 Methods

3.1.1 Original pipeline

The steps of DeLUCS, as described in [7] and as can be understood from reading the relevant code from the official DeLUCS GitHub repository, are as follows:

1. Cleaning the data (build_dp.py):
    - For a given directory, all subdirectories are searched, and contained FASTA files are read, the sequences are cleaned (i.e., the letters are capitalized, and only A, C, T, and Gs are kept), and the sequences are stored in a pickle file, along with their accession number, and their label (where the label is used only in the case of evaluating the DeLUCS pipeline, and is the name of the subdirectory in which that FASTA file was found).
2. Preprocessing the data, and generating the mimics (get_pairs.py):

(a) For each sequence:

   i. Its k-mer count is calculated, and is normalized (by dividing the k-mer count array by the sum of the counts). The label attached to that sequence (if available) is converted into a unique numeric ID (thus, if two sequences share a label, their numeric ID will be equivalent). This normalized k-mer count will be used in the testing set, and the numeric ID will be used as that testing data point's label.

   ii. One mimic is generated by randomly performing transitions on the original sequence, and then taking the normalized k-mer count of this newly created sequence. A second mimic is generated by a similar process, instead using transversions instead of transitions. At least one more mimic is generated by performing both transitions and transversions (where, given the hyperparameter n to the DeLUCS pipeline, n mimics are generated, $n-2$ mimics will be generated by both transitions and transversions).

   iii. The training features are extended with the following pairs of arrays of normalized k-mer counts:

   ```
   [(normal, transition), (normal, transversion),
   (normal, transition+transversion_1), ... ,
   (normal, transition+transversion_n)]
   ```

   where n is the number of mimics. Thus, at least three mimics must be used (in order to ensure one mimic being generated using only transitions, another being generated by only transversions, and $n-2$ being generated by both transitions and transversions.

   The purpose of having these true sequence/mimic pairs is because the neural networks will be using invariant information clustering as the loss function, and thus will be concerned with the mutual information between the two normalized k-mer counts.

(b) The training and testing features, and the testing labels are all saved to a pickle file.

3. Training (and optionally, evaluating) the neural networks:

   - Ten sequential neural networks are independently trained (batch size 512, 150 epochs, no early stopping, Adam optimizer, $\lambda = 2.8$, learning rate $= 8 \times 10^{-5}$) on the training data (the sequence/mimic pairs), and then predict the labels of the testing data (the sequences). They consist of linear, ReLU, and dropout layers, with the output layer being a softmax. They aim to minimize the negative weighted mutual information between the network's prediction for a sequence, and the network's prediction for that sequence's mimic pair [7].

     The loss function, given by Eq. (3.1), leverages the idea of Invariant Information Loss, as described in [28]. Invariant Information Loss allows for the combination of unsupervised clustering with deep learning in a way that avoids degenerate solutions (i.e., assigning to a single cluster all datapoints; as would be the result of combining classical unsupervised learning methods objective (such as K-Means) with deep learning) or having to add additional steps to the pipeline (such as pre-training and post-processing) [28].

$$\mathcal{L}(x,\ \tilde{x}) \ = \ -\lambda \ \cdot \ H(\Phi(x)) \ + \ H(\Phi(x) \mid \Phi(\tilde{x})) \tag{3.1}$$

The loss function used by the neural networks in DeLUCS is given by Eq. (3.1). This represents the weighted negative mutual information. The underlying concept is that using x, one can predict information about $\tilde{x}$ (where this relationship corresponds to the mutual information between these two variables. $\Phi(x)$ represents the compressed representation of x (wherein, only what is needed in order to predict $\Phi(\tilde{x})$ the compressed representation of $\tilde{x}$, is kept), and can be calculated by using deep neural networks, with a softmax as the output layer [7].

By minimizing the value $\mathcal{L}(x,\ \tilde{x})$, each neural network: 1) maximizes the amount of randomness of its output (i.e., the entropy of the neural network's prediction for x, denoted by $H(\Phi(x))$) preventing all sequences from being assigned to the same cluster; 2) while simultaneously minimizing the amount of randomness of the input, as the assumption is that each sequence contains information that can be perfectly predicted from its mimic, allowing a network to only need to know that distinguishing information (known as mutual information) and thus compressing their representation (this term represents the entropy of the network's prediction for x (the sequence), given the prediction for $\tilde{x}$ (the mimic), denoted by $H(\Phi(x) \mid \Phi(\tilde{x}))$) [7].

Note that the initial weights of the networks are initialized using the Kaiming method [29] (which assigns the initial weights using a probability function, and was specifically designed for deep neural networks which use ReLU layers, just as DeLUCS does) to avoid either term of the loss function from becoming dominant (i.e., an exploding gradient) [7].

- In the case of evaluating the model's performance (EvaluateDeLUCS.py):
  (a) The neural networks cluster the data into n clusters, where there are n unique labels (i.e., the number of distinct taxa into which the sequences of the dataset were divided).
  (b) For each network, the Hungarian Algorithm [30] is used to determine the optimal assignments for each cluster to the true taxonomic labels (by using the confusion matrix for each network as the cost matrix for the Hungarian Algorithm). This mapping between the clusters and true labels is used to both ensure consistency of clusters between neural networks before taking their majority vote, and to allow for the accuracy to be derived.

(c) For each testing data point (equivalently, for each sequence), the mode of the predictions of the neural networks (i.e., the majority vote) is then used as the overall cluster assignment for that point, and the accuracy is derived (being the sum of all correct predictions, divided by the total number of sequences). The output consists of the overall accuracy of the pipeline (i.e., the accuracy of the majority vote), and the confusion matrix of the majority vote.

- In the case of deploying the model (TrainDeLUCS.py):
  (a) The neural networks cluster the data into n clusters, where n is given as a hyperparameter.
  (b) The Hungarian Algorithm is used to determine the optimal assignments for each cluster between networks, such that the results between the networks agree with each other as much as possible in terms of the relative compositions of their clusters.
  (c) For each sequence, the majority vote of the neural networks is taken, and is used as the overall prediction. The output consists of the prediction for each sequence, along with the associated accession number for that sequence.

The modification to the pipeline that differentiates DeLUCS-F from DeLUCS is in step two (i.e., how the sequence/mimic pairs were generated), though quality-of-life improvements were added to all three steps, as described in Appendix C. However, various inconsistencies were found between the original pipeline as described in the DeLUCS study [7] and the pipeline as available on the linked GitHub page, and had to be dealt with before comparing the results of both pipelines. A detailed description of these inconsistencies is given in Appendix A, along with how addressing each inconsistency in the original pipeline affected its performance across all datasets, as measured by its accuracy.

The readme file from the DeLUCS GitHub page recommends using the updated version of the code at `github.com/Kari-Genomics-Lab/DeLUCS`, however, that version also contained the same inconsistencies. The code used and modified throughout this study is from the original GitHub page located at `github.com/millanp95/DeLUCS`. The updated version of the code, where these inconsistencies have been addressed, is available at `github.com/DylanCPL/DeLUCS-F/tree/Original-DeLUCS-with-fixes`.

3.1.2 Differences between the DeLUCS and DeLUCS-F pipelines

Comparing DeLUCS-F to DeLUCS, the first step of cleaning the data is the same. What differs is step two (i.e., get_pairs.py). Instead of generating mimics, n fragments of length L are copied from the original sequence, and their normalized k-mer counts are calculated. Then, the training features are extended with the following pairs of arrays of normalized k-mer counts:

```
[(normal, fragment 1), (normal, fragment 2), ... , (normal, fragment n)]
```

As a result of the fact that unlike the mimic sequences (where one was generated through transitions, another through transversions, and the rest through both), the fragments are generated through the same process, the use of at least three pairs per sequence is no longer required.

While the major change between DeLUCS and DeLUCS-F was the change from mimic sequences to sequence fragments, at all three steps, quality-of-life improvements were implemented to the associated scripts. These include small changes to prevent crashes, more flexibility in the arguments used to run DeLUCS, and more useful output files; these improvements are described further in Appendix C.

It is worth noting that the step that generates fragments can be run with either relative or fixed fragment length (where relative fragment length is defined by entering a proportion, and for each sequence, random fragments of that proportion of the sequence length is used). For all tests in this study, given that the datasets had varying sequence lengths, DeLUCS-F was run using relative sequence length, in order to standardize the hyperparameter usage between datasets. This allowed for the measurement of the effects of fragment length on the model's performance between datasets whose sequences had average lengths which differed by as much as a factor of seven. Additionally, when using relative fragment lengths, the fragment lengths were calculated on a sequence-by-sequence basis, as opposed to taking the average sequence length for a dataset. This is because some datasets could vary widely in their minimum and average sequence length; e.g., for Proteobacteria, the average sequence length was 434,150bp, but the minimum sequence length was 150,499, representing only 34.7% of the average sequence length.

The version of the code for DeLUCS-F that was used to generate the results in this study can be accessed at `github.com/DylanCPL/DeLUCS-F`.

3.2 Datasets

The performance of DeLUCS-F was evaluated using the same datasets as used in the original DeLUCS study [7], as this allowed for a clear comparison between the two pipelines. Additionally, this allowed for the comparison of DeLUCS-F with other algorithms (K-Means++, Gaussian Mixture Models, and a supervised analogue of the DeLUCS neural networks), using the results presented in the DeLUCS study [7]. For each dataset, the sequence statistics were verified (i.e., the total number of sequences, the minimum and maximum lengths, and the average length), and are presented in Table 3.1.

The datasets contain (I) mitochondrial genomes of vertebrates, (II) segments of bacterial genomes, (III) viral genes and genomes. All datasets were obtained from the DeLUCS GitHub repository. The information given in the DeLUCS study [7] regarding these datasets is summarized as follows:

I. The first dataset consists of mitochondrial genomes obtained from NCBI on November 16, 2020 [31].

   These are complete mitochondrial genomes for vertebrates (selecting sequences with lengths between 14,000 and 24,500 base pairs, and discarding both clusters that had less than 20 sequences, or that lacked a label), with the goal of use being to assess the performance of the pipeline at descending taxonomic ranks. To do this, a decision-tree approach had been taken, where at each level, the cluster with the largest number of available sequences was chosen for further investigation in the subsequent test. The number of sequences used between tests varied in order to achieve similar cluster sizes at each level (e.g., in the first test of clustering amongst Vertebrata, 500 sequences for Actinopterygii were used, while only 113 Actinopterygii sequences were used in the test which clustered amongst Actinopterygii, in order to prevent under/overrepresentation of some clusters) [7].

II. The second dataset consists of bacterial DNA obtained using GTDB, release 95, on January 18, 2021 [32]. These are randomly selected segments from full bacterial genomes (selecting sequences with minimum lengths of 150,000 base pairs) belonging to bacteria from eight different families, belonging to three different phyla. Half of the families correspond to Proteobacteria (while three correspond to Firmicutes, and the remaining one to Spirochaetes), and while one test attempts to cluster amongst all families, another test clusters only amongst Proteobacteria. Compared to (I) and (III), these sequences are much larger (433 kilobase pairs (kbp) on average, compared to 16kbp, and 5kbp, respectively), deal with many more sequences in a single test (there are 3,200 bacterial sequences (with 1,299 being Proteobacteria) compared to the largest test in (I) being 2,500 sequences, and the largest test in (III) being 1,633 sequences), and simultaneously clustering amongst clusters that present much more variation in inter-cluster distances. This poses the following challenges:

   (a) The scale of the data, and the amount of datapoints, mean that with an inefficient approach to either sequence classification or clustering, even an alignment-free solution could struggle to deal with the amount of data (whereas, since alignment-based methods require sequences to share a common ancestor (i.e., sequence homology) using such methods would be impossible).

   (b) The relatively larger variation in inter-cluster distances mean that the pipeline will need to simultaneously cluster amongst both closely-related and distantly-related families; thus, the algorithm may struggle in relying on a single range of relative distances to separate between datapoints, as some distant datapoints will be of the same family, while other nearby datapoints will be of different familes [7].

III. The third dataset consists of viral DNA obtained from NCBI [31] and the Hepatitis Virus Database [33] on October 14, 2020. These contain both deterministically obtained genome segments (specifically, segment 6 of the *Influenza A virus* genome, responsible for encoding the neuraminidase protein), and full virus genomes (in the case of *Dengue virus* and *Hepatitis B virus*). Unlike (I) and (II), the *Influenza A virus* dataset is small enough to be handled by alignment-based methods, and had been selected in order to demonstrate the fact that DeLUCS could handle a variety of genomic data. Additionally, the high accuracy obtained by DeLUCS for this particular dataset (99% for *Influenza A virus*) observed within that study demonstrated that DeLUCS could even be considered in applications where alignment-based solutions were already feasible [7].

Table 3.1 summarizes the sequence statistics.

Table 3.1: Details of the datasets on which all tests in this study were run.

Dataset	**Number of sequences**	**Number of clusters**	**Minimum sequence length (bp)**	**Average sequence length (bp)**	**Maximum sequence length (bp)**
Vertebrata	2,500	5	14,127	16,929	24,317
Actinopterygii	113	3	15,531	16,619	18,062
Neopterygii	1,475	6	15,468	16,687	19,763
Ostariophysi	363	3	15,664	16,636	17,998
Cypriniformes	545	7	16,061	16,608	17,280
Cyprinidae	447	12	16,070	16,632	17,426
Bacteria	3,200	8	150,499	433,614	500,000
Proteobacteria	1,600	4	150,499	434,150	500,000
Influenza A virus	949	5	1,345	1,409	1,468
Dengue virus	1,633	4	10,010	10,557	10,991
Hepatitis B virus	1,562	6	3,061	3,209	3,227

Test I is composed of datasets 1 through 6, which are the complete mitochondrial genomes of various vertebrates. Datasets 1 through 6 follow a decision-tree style approach, in that the cluster with the most available sequences is selected as the subsequent dataset (e.g., Actinopterygii is a class belonging to the subphylum Vertebrata). Test II is composed of datasets 7 and 8, which are segments of the complete genomes of various Bacteria. Note that dataset 8 (Proteobacteria) is a subset of dataset 7 (Bacteria). Test III is composed of datasets 9 through 11, which are viral genetic material: dataset 9 (*Influenza A virus*) is a single gene (responsible for the encoding of neuraminidase in *Influenza A virus*), while datasets 10 and 11 are the complete genomes.

Chapter 4

Results

The performance of DeLUCS-F is measured by the use of accuracy, as had been done in the DeLUCS study [7]. This is accomplished in the same manner in both pipelines:

> For each neural network, the taxonomic labels are mapped to the predicted clusters in an optimal manner (through the use of the Hungarian algorithm), the majority vote for each sequence is taken as the final prediction, and then using the labels, the accuracy is calculated as the sum of correct assignments divided by the total number of sequences [7].

For both DeLUCS and DeLUCS-F, all results obtained throughout this chapter were obtained by fixing all random seeds at 0. This was chosen instead of averaging the results of ten trials (for each test) performed with random seeds, as this ensured perfect reproducibility of the results of this study.

Table 4.1 combines the results of this study for the majority vote accuracies of DeLUCS and DeLUCS-F, and compares them with the results of other methods, as per the values reported in the DeLUCS study [7]. In that study, the performance of DeLUCS had been compared to that of K-Means++ and Gaussian Mixture Models, two classic unsupervised learning techniques (where their accuracies were also obtained by mapping their cluster assignments to the taxonomic labels in an optimal manner through the use of the Hungarian algorithm). Additionally, the accuracy of DeLUCS had been compared to the accuracy of a direct supervised analogue of DeLUCS; that is, a neural network with the same architecture, fed with the same data (except, having access to the labels during training). That supervised analogue used the cross-entropy loss function (in place of invariant information loss), and had been assessed using a 70/30 training/test split, and accuracy was used as the measure of performance.

Table 4.1: Accuracies for the various methods.

Dataset	Supervised	Unsupervised			
		GMM	K-Means++	DeLUCS	DeLUCS-F
Vertebrata	99%	72%	81%	93.40%	96.32%
Actinopterygii	98%	92%	96%	100.00%	100.00%
Neopterygii	99%	70%	82%	84.20%	84.88%
Ostariophysi	100%	68%	73%	88.15%	94.21%
Cypriniformes	87%	66%	77%	78.72%	79.08%
Cyprinidae	80%	84%	87%	89.49%	89.48%
Bacteria	98%	58%	60%	68.44%	79.44%
Proteobacteria	99%	50%	43%	90.06%	90.56%
Influenza A virus	100%	96%	99%	99.26%	99.47%
Dengue virus	100%	100%	100%	100.00%	100.00%
Hepatitis B virus	100%	100%	100%	100.00%	100.00%

For GMM (Gaussian Mixture Models) and K-Means++, the reported accuracies are the result of averaging 10 runs of the algorithms [7]. For DeLUCS and DeLUCS-F, their accuracies are that of clustering the test set (i.e., the original sequences); while for the supervised learner, its accuracy is that of classifying the test set (i.e., the 30% split of the sequences which the learner was not trained on). The values for GMM, K-Means++, and the supervised analogue of a neural network from DeLUCS were obtained directly from Tables 4-6 of [7].
For DeLUCS, and DeLUCS-F, their results were obtained by fixing all random seeds to 0, to ensure perfect reproducibility; additionally, the accuracy reported is that obtained by the majority vote of the ten independently trained neural networks.

DeLUCS and DeLUCS-F largely outperform the other unsupervised methods in all tests, and DeLUCS-F outperforms DeLUCS in all but four tests (Actinopterygii, Cyprinidae, *Dengue virus*, and *Hepatitis B virus*; for three of which, both models tied with a perfect accuracy). However, DeLUCS-F is still largely outperformed by the supervised learning method. This is reasonable, as supervised learning methods are trained on labelled data, and given that the performance measure used (accuracy) is defined as the ability to correctly match labels to observations, supervised models have the advantage that they can directly learn associations between features and the desired labels, and would thus typically be expected to outperform unsupervised models [16, 17].

In each of the three tests that for the datasets with less than five clusters, DeLUCS-F achieved its highest accuracy with smaller fragment lengths (where "smaller" is relative to the other datasets in that test), while for datasets with five or more clusters, relatively higher fragment lengths were required. This is similar to the findings in the DeLUCS study [7], where it was found that better performance was achieved on datasets with less than 150 sequences per cluster when using an increased number of mimics.

Throughout the following tests, the same hyperparameters were used as in the DeLUCS study. That is, the value for k is fixed at 6, and the number of fragments matches the number of mimics used per dataset, as described in the DeLUCS study [7]. The various tables for all three tests describe the accuracies (majority vote amongst ten independently trained neural networks, and average of the accuracies of those neural networks) of the original fixed pipeline, compared to the best accuracies obtained by the modified pipeline, along with the fragment length associated with this accuracy. When multiple fragment lengths led to the same highest accuracy for a given dataset, the shorter fragment length is given, as that led to the faster computation time for get_pairs.py, and this was typically the case for EvaluateDeLUCS.py.

4.1 Test I: Clustering using mitochondrial genomes of vertebrates

Figure 4.1 displays the relationship between the relative length of fragments used and the majority vote accuracy for DeLUCS-F, compared to the majority vote accuracy obtained by the DeLUCS pipeline for each dataset.

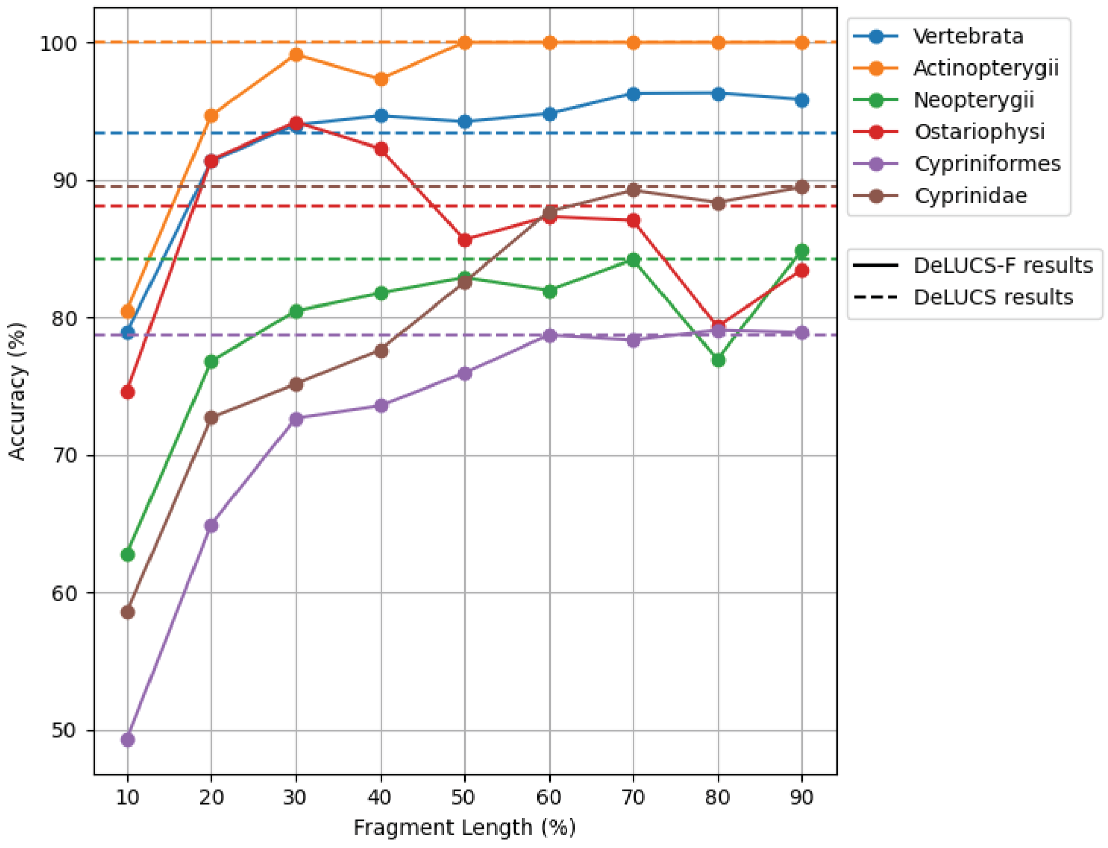
Note that there was not a clear association between fragment length and the overall accuracy of DeLUCS-F (i.e., the majority vote accuracy). For example, as shown in Figure 4.1, while there was a clear trend in Cyprinidae that increasing fragment length led to an increase in accuracy (except at 80%, which shown an accuracy lower than that obtained with a fragment length of 70%), in Ostariophysi, once the relative fragment length reached 30%, the optimal accuracy had been reached, and increases in fragment length beyond that point only led to relatively lower accuracies. However, this lack of a clear positive correlation between accuracy and fragment length may have had to do with the nondeterministic nature of the pipeline. While all random seeds had been fixed to 0 to ensure reproducibility, it may be worth investigating if a clearer relationship between accuracy and fragment length emerges as a result of averaging ten trials (for each set of hyperparameters) with random seeds.

Table 4.2 summarizes the important findings of the above table; namely, the fragment lengths which led to the highest majority vote accuracy for each dataset. DeLUCS-F outperforms DeLUCS in all datasets, except for Actinopterygii, where they both scored perfectly, and Cyprinidae, where DeLUCS-F was outperformed by 0.01%.

The datasets for which DeLUCS-F performed best with higher relative fragment length (i.e., larger than 50%) correspond to the datasets where there were five or more clusters (with such datasets ranging between having five and twelve clusters). It would appear that a shorter relative fragment length was sufficient for clustering in the cases where there were less than five clusters. While there was no clear pattern observed which described the ideal relative fragment length to use, however, one might be observed if more trials were performed for each dataset for each set of hyperparameters (with random seeds unfixed).

Given that a majority vote is typically rather uninterpretable (i.e., it is difficult to discern the reason for which each prediction was made), further results were obtained to investigate the

Figure 4.1: The effect of fragment length on the majority vote accuracy of DeLUCS-F for the datasets consisting of the complete mitochondrial genomes of vertebrates.



The effect of relative fragment length (where, for each sequence of a given dataset, n (potentially overlapping) fragments are randomly selected from the sequence, each of a length of the specified percentage of the total sequence's length) on the majority vote accuracy of DeLUCS-F, compared to the majority vote accuracy of DeLUCS, for the mitochondrial genome datasets. Fragment lengths below 30% led to the lowest accuracies for each dataset. DeLUCS-F outperforms DeLUCS on all but the Cyprinidae test, although, the points at which DeLUCS-F outperformed DeLUCS varied by fragment length.

While the labels of each sequential dataset refer to a subset of the previous label, each dataset is not a subset of the previous. That is, while Neopterygii is a subclass of the class Actinopterygii, the Neopterygii dataset contains 1475 sequences, while the Actinopterygii dataset contains only 40 Neopterygii sequences (and 113 sequences in total). This is a result of data availability, as explained in [7], which is the source of these datasets, and is intended to avoid over/underrepresentation of some of the clusters; e.g., while 1475 Neopterygii sequences (a subclass of Actinopterygii) were available, only 33 sequences for Polypteriformes (another subclass of Actinopterygii) were available.)

individual accuracies of the neural networks. This average of the individual accuracies of the ten independently trained neural networks, in relation to the fragment length, was plotted, as shown in Figure 4.2.

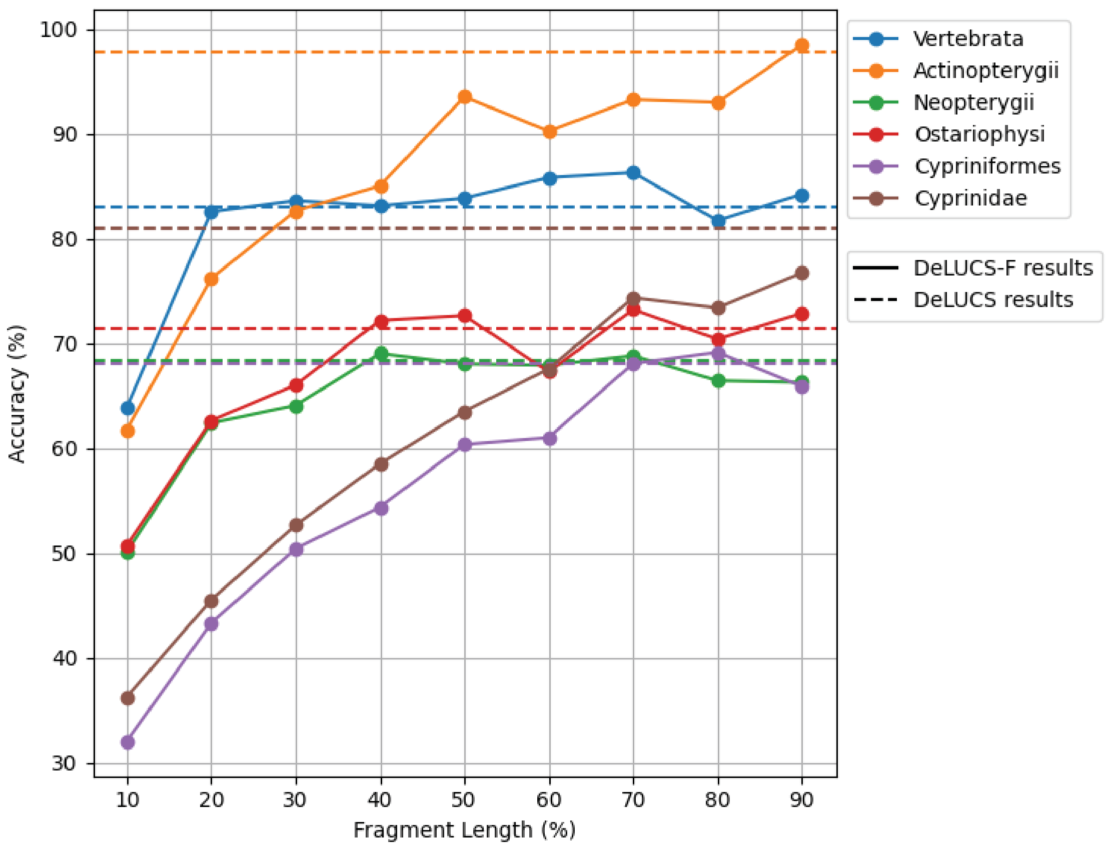
Table 4.3 summarizes the important findings of Figure 4.2; namely, the fragment lengths which led to the highest average of individual accuracies for each dataset. Interestingly, the fragment lengths which led to the highest average of individual accuracies do not correspond to those which led to the highest majority vote accuracy, except in the cases of Cypriniformes and Cyprinidae.

Table 4.2: Comparison between the highest majority vote accuracies obtained by DeLUCS-F, and the majority vote accuracies of DeLUCS for the datasets consisting of the complete mitochondrial genomes of vertebrates.

Dataset	DeLUCS MVA	DeLUCS-F highest MVA	Fragment length (relative)	Number of mimics/ fragments
Vertebrata	93.40%	96.32%	80%	3
Actinopterygii	100.00%	100.00%	50%	8
Neopterygii	84.20%	84.88%	90%	3
Ostariophysi	88.15%	94.21%	30%	8
Cypriniformes	78.72%	79.08%	80%	8
Cyprinidae	89.49%	89.48%	90%	8

Each of the scores for DeLUCS-F are the highest majority vote accuracies (MVA) obtained from the nine trials for each dataset (i.e., for each relative fragment lengths, from 10% to 90%). The number of fragments used for each dataset by DeLUCS-F match the number of mimics used by DeLUCS for these tests; the ideal number of mimics to be used for all datasets were outlined in the DeLUCS study [7]. DeLUCS only outperforms DeLUCS-F on Cyprinidae, which it does by only 0.01%.

Figure 4.2: The effect of fragment length on the average of the individual accuracies of the ten neural networks of DeLUCS-F for the datasets consisting of the complete mitochondrial genomes of vertebrates.



The effect of relative fragment length on the average of the individual accuracies of the ten independently trained neural networks of DeLUCS-F, compared to the same of DeLUCS, for the mitochondrial genome datasets. The average accuracies of the neural networks of DeLUCS and DeLUCS-F are lower than those obtained by the majority vote for each pipeline. Unlike in Figure 4.1, which had depicted the accuracies obtained by the majority vote for one trial for each set of hyperparameters for each dataset, these results show the average of ten trials; note that there is a stronger positive relationship between the relative fragment length and the average of individual accuracies, than between the relative fragment length and the majority vote accuracy.

This indicates that ten networks having a higher overall average accuracy does not necessarily mean that their majority vote will have a higher accuracy.

Table 4.3: Comparison between the highest average of individual accuracies obtained by DeLUCS-F, and the average of individual accuracies obtained by DeLUCS for the datasets consisting of the complete mitochondrial genomes of vertebrates.

Dataset	DeLUCS AIA	DeLUCS-F highest AIA	Fragment length (relative)	Number of mimics/ fragments
Vertebrata	93.40%	86.32%	70%	3
Actinopterygii	100.00%	98.50%	90%	8
Neopterygii	84.20%	69.06%	40%	3
Ostariophysi	88.15%	73.25%	70%	8
Cypriniformes	78.72%	69.17%	80%	8
Cyprinidae	80.94%	76.73%	90%	8

Each of the scores for DeLUCS-F are the highest of the average of individual accuracies (AIA) (i.e., the average of the accuracies of the ten independently trained neural networks), obtained from the nine trials for each dataset (i.e., for each relative fragment lengths, from 10% to 90%).

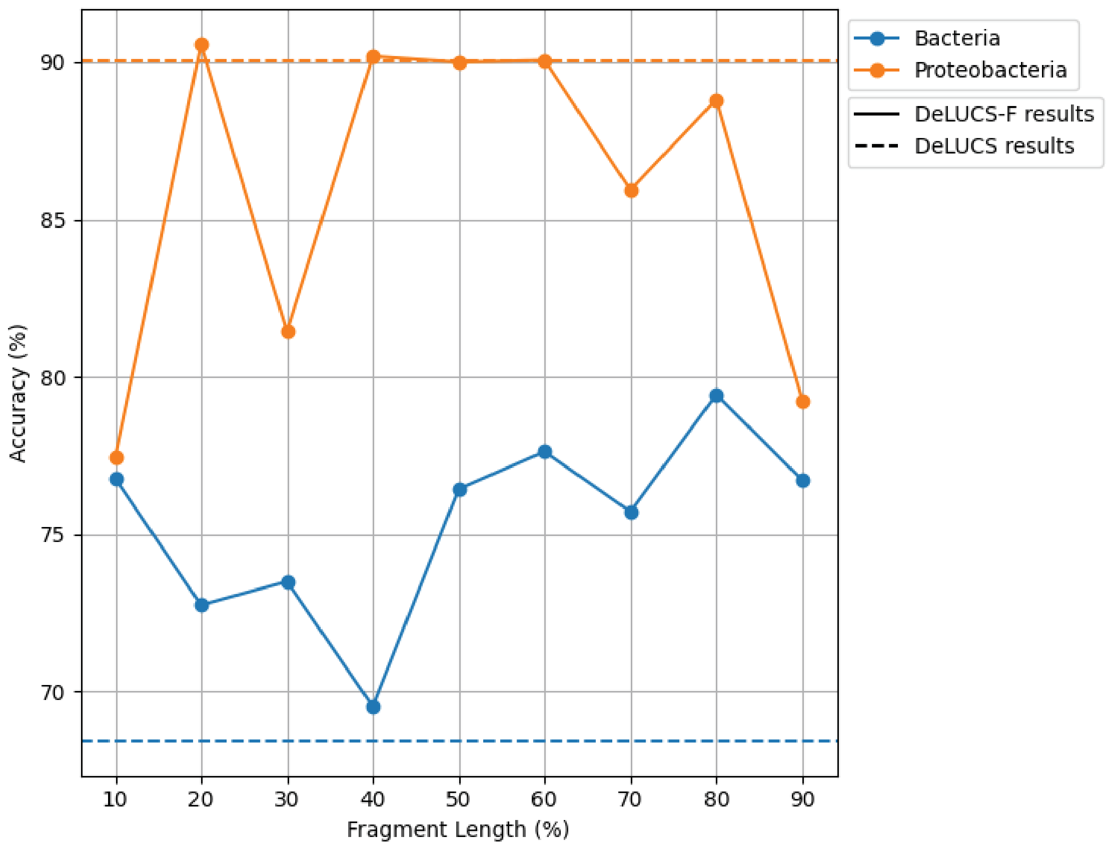
As per Table 4.3, while DeLUCS-F had outperformed DeLUCS on all but the Cyprinidae dataset in terms of majority vote accuracy (as shown in Table 4.2), DeLUCS-F is outperformed by DeLUCS in terms of the average accuracy of the individual neural networks, indicating that even though DeLUCS-F had lower average accuracies of its neural networks than DeLUCS, this did not detract from its majority vote accuracy, which often surpassed that of DeLUCS.

4.2 Test II: Clustering using bacterial genome segments

Figure 4.3 displays the relationship between the majority vote accuracy and the fragment length used to cluster amongst families of both Bacteria and Proteobacteria (where the Proteobacteria dataset is a subset of the Bacteria dataset, as explained in Materials and Methods) using segments of their genome. Both DeLUCS-F and DeLUCS scored much higher accuracies for the Proteobacteria dataset than for the Bacteria dataset. As discussed in the DeLUCS study, the lower accuracy obtained for the Bacteria dataset may be due to the heterogeneity of the dataset, in that the pipelines are simultaneously responsible for clustering both very similar and very distant families [7].

The most notable results of the test involving majority vote accuracy are captured in Table reftab: MVA bacterial. Note that DeLUCS-F outperforms DeLUCS in both datasets. Note that in order to cluster amongst Proteobacteria (a subset of the Bacteria dataset), DeLUCS-F required fragment lengths a quarter of the length required for the highest results obtained in the Bacteria dataset, and scored a much higher accuracy using only that. As noted in [7], the Bacteria dataset is much more challenging, as the inter-cluster distance varies widely, with some families being

Figure 4.3: The effect of fragment length on the majority vote accuracy of DeLUCS-F for the datasets consisting of segments of bacterial genomes.



The effect of relative fragment length on the majority vote accuracy of DeLUCS-F, compared to the majority vote accuracy of DeLUCS, for the bacterial genome segments datasets. Unlike Test I (where each subsequent datasets chosen represented subsets of the preceding datasets; however, where the datasets themselves were not true subsets, in terms of the specific sequences included), the Proteobacteria dataset is a perfect subset of the sequences used in the Bacteria dataset. For DeLUCS-F, results were found that are similar to those found in [7], where DeLUCS had performed better on the Proteobacteria dataset than the Bacteria dataset, which was thought to be a result of the fact that with Proteobacteria, the model was only dealing with approximately equally distant clusters, whereas the clusters in the Bacteria dataset were of largely varying distances.

distantly related, whereas others are very closely related. Additionally, this aligns with what was observed in the first collection of datasets: a small relative fragment length was sufficient to achieve the highest obtained accuracy in clustering Proteobacteria, where there were only four clusters; while in Bacteria, where there were eight clusters, a larger relative fragment length was required to achieve the highest accuracy.

Figure 4.4 displays the relationship between the average of the individual accuracies and the fragment length.

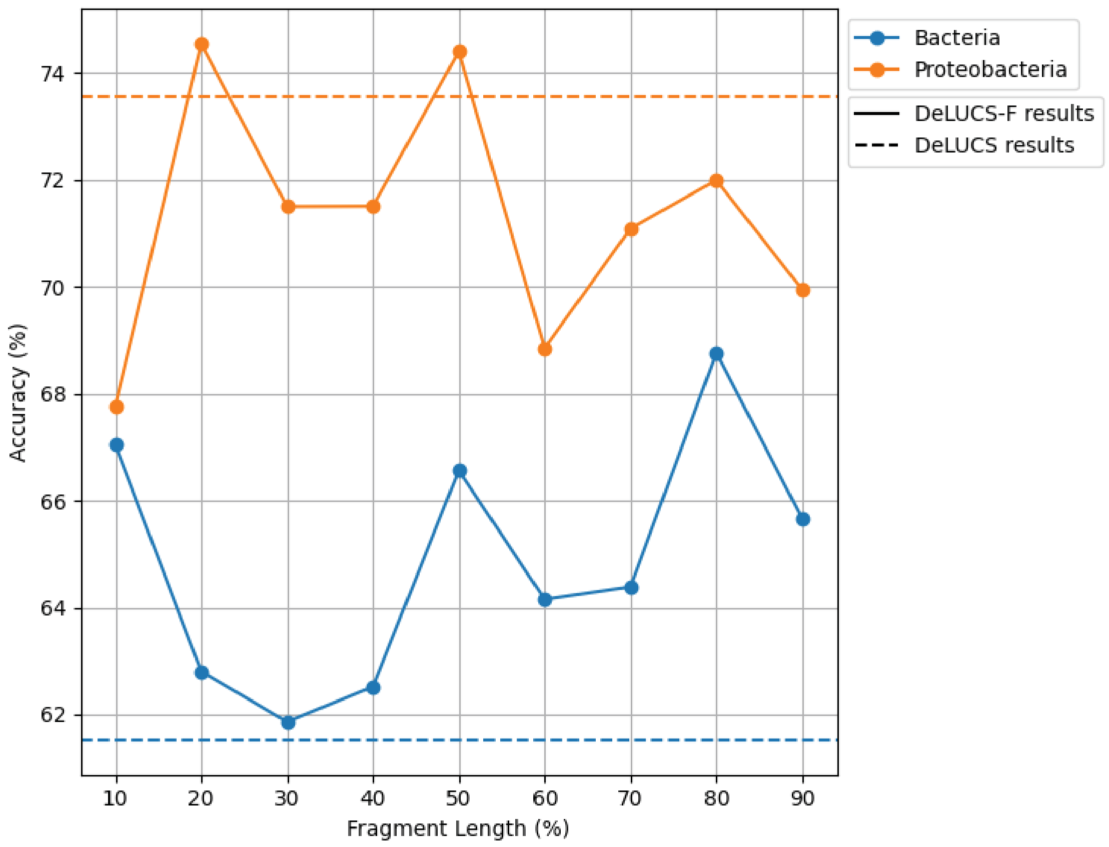
Table 4.5 summarizes the most important results from above. Notice that in contrast to Test I, DeLUCS-F outperforms DeLUCS in terms of the average accuracy amongst its neural networks.

Table 4.4: Comparison between the highest majority vote accuracies obtained by DeLUCS-F, and the majority vote accuracies of DeLUCS for the datasets consisting of segments of bacterial genomes.

Dataset	DeLUCS MVA	DeLUCS-F highest MVA	Fragment length (relative)	Number of mimics/ fragments
Bacteria	68.44%	79.44%	80%	3
Proteobacteria	90.06%	90.56%	20%	3

Each of the scores for DeLUCS-F are the highest majority vote accuracies (MVA) obtained from the nine trials for each dataset (i.e., for each relative fragment lengths, from 10% to 90%).

Figure 4.4: The effect of fragment length on the average of the individual accuracies of the ten neural networks of DeLUCS-F for the datasets consisting of segments of bacterial genomes.



The effect of relative fragment length on the average of the individual accuracies of the ten independently trained neural networks of DeLUCS-F, compared to the same of DeLUCS, for the mitochondrial genome datasets. As with the datasets from Test I, the average accuracies of the neural networks of DeLUCS and DeLUCS-F are lower than those obtained by the majority vote for each pipeline.

Table 4.5: Comparison between the highest average of individual accuracies obtained by DeLUCS-F, and the average of individual accuracies obtained by DeLUCS for the datasets consisting of segments of bacterial genomes.

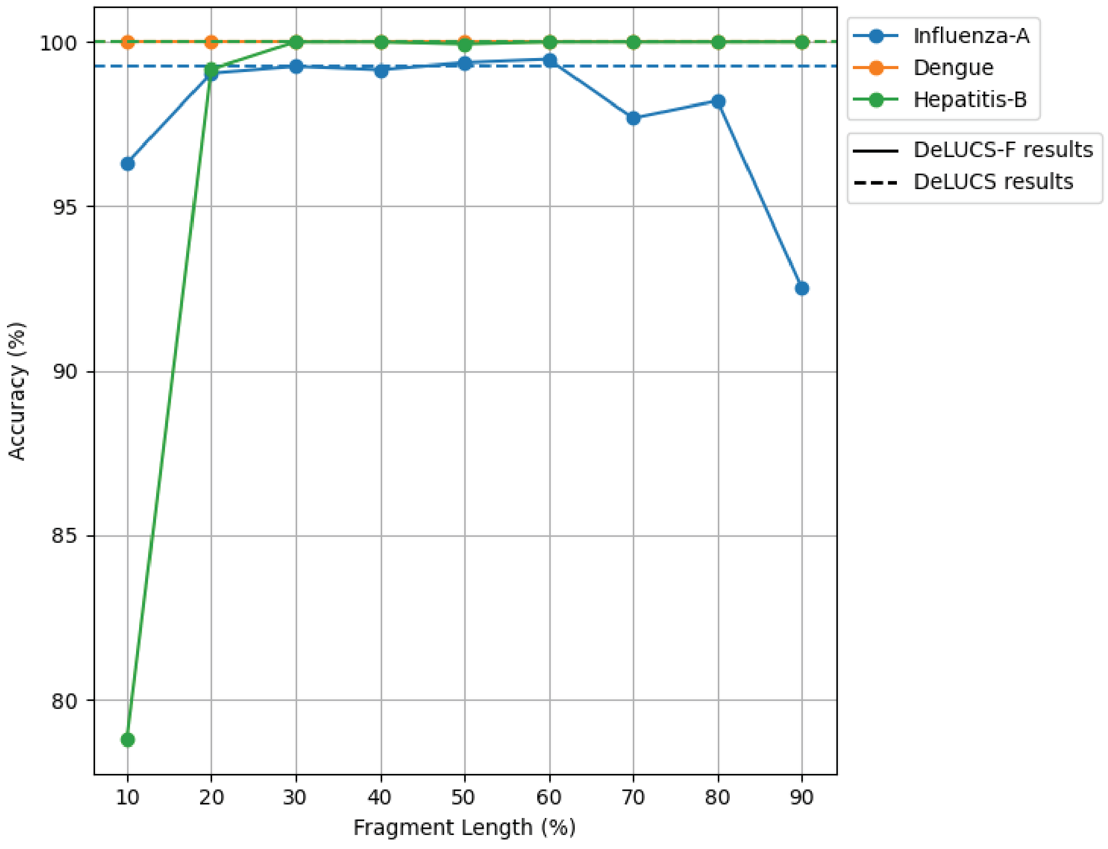
Dataset	DeLUCS AIA	DeLUCS-F highest AIA	Fragment length (relative)	Number of mimics/ fragments
Bacteria	61.53%	68.77%	80%	3
Proteobacteria	73.54%	74.54%	20%	3

Each of the scores for DeLUCS-F are the highest of the average of individual accuracies (AIA) (i.e., the average of the accuracies of the ten independently trained neural networks), obtained from the nine trials for each dataset (i.e., for each relative fragment lengths, from 10% to 90%).

4.3 Test III: Clustering using viral genes, and complete viral genomes

Figure 4.5 displays the relationship between the majority vote accuracy and the fragment length used to cluster various viral datasets. For *Influenza A virus*, only the gene that encodes the neuraminidase protein is used (segment 6 of the genome); for both *Dengue virus* and *Hepatitis B virus*, their full genomes are used.

Figure 4.5: The effect of fragment length on the majority vote accuracy of DeLUCS-F for the datasets consisting of viral genomes and genes.



The effect of relative fragment length on the majority vote accuracy of DeLUCS-F, compared to the majority vote accuracy of DeLUCS, for the viral genomes and genes datasets.

The highest majority vote accuracies obtained by DeLUCS-F, in comparison to the majority vote accuracies obtained by DeLUCS, are compared in Table 4.6. Note that the majority vote accuracies of DeLUCS-F are nearly identical to those of DeLUCS, with DeLUCS-F only marginally outperforming DeLUCS on the *Influenza A virus* dataset, and tying at perfect scores for the other two datasets. It is worth noting that three fragments consisting of only 10% of the sequence length was sufficient to obtain a perfect score in the *Dengue virus* dataset, and only 30% of the sequence length was required for a perfect score for *Hepatitis B virus*; 60%, however, was required to achieve the highest obtained accuracy for *Influenza A virus*. The same pattern was found here as was found with the previous datasets, where for the dataset with only four clusters (*Dengue virus*), a shorter relative fragment length was sufficient to achieve maximal accuracy.

Table 4.6: Comparison between the highest majority vote accuracies obtained by DeLUCS-F, and the majority vote accuracies of DeLUCS for the datasets consisting of viral genomes and genes.

Dataset	DeLUCS MVA	DeLUCS-F highest MVA	Fragment length (relative)	Number of mimics/ fragments
Influenza A virus	99.26%	99.47%	60%	3
Dengue virus	100.00%	100.00%	10%	3
Hepatitis B virus	100.00%	100.00%	30%	3

Each of the scores for DeLUCS-F are the highest majority vote accuracies (MVA) obtained from the nine trials for each dataset (i.e., for each relative fragment lengths, from 10% to 90%).

As shown in Table 3.1 from Chapter 3, the sequences from the *Influenza A virus* dataset had an average length of 1,409 base pairs; *Dengue virus* had 10,559, and *Hepatitis B virus* had 3,210.

It was found that when multiplying these average lengths by the relative fragment lengths associated with the highest majority vote accuracy of DeLUCS-F (from Table 4.6), similar average fragment lengths were obtained. This information is summarized in Table 4.7. In summary, it appears that three fragments of 1,000 base pairs long (with a k-mer resolution of 6) per sequence are sufficient to differentiate the various viral sequences with near perfect accuracy for DeLUCS-F, even when the sequence used consists of only a single gene. Future works may wish to investigate whether this pattern holds for other viral datasets.

Figure 4.6 displays the relationship between the average of the individual accuracies, and the fragment length used.

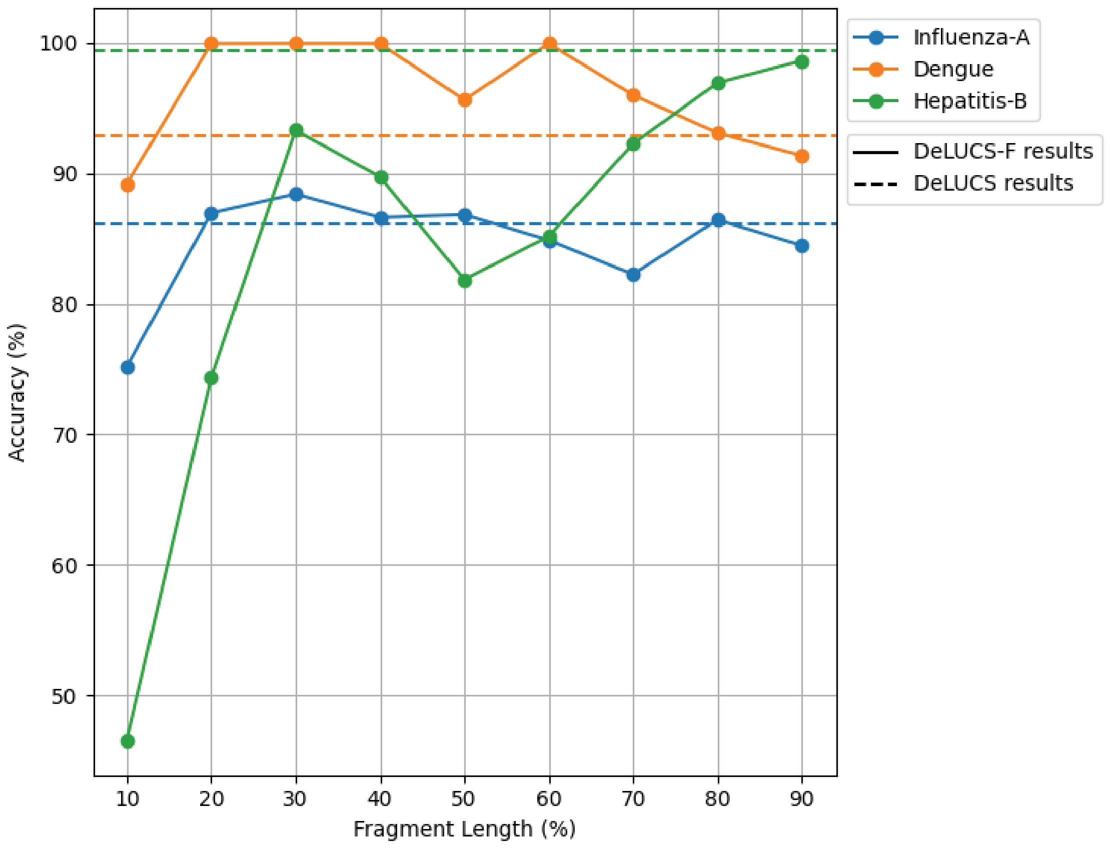
Table 4.8 captures the highest average of individual accuracies obtained by DeLUCS-F, and compares them to those scores obtained by DeLUCS. DeLUCS-F outperforms DeLUCS on both *Influenza A virus* and *Dengue virus*, but is outperformed by DeLUCS on *Hepatitis B virus*.

Table 4.7: Comparison between the highest majority vote accuracies obtained by DeLUCS-F and the associated relative fragment lengths, and the average length of those fragments, given the average sequence length for each dataset of viral genomes and genes.

Dataset	DeLUCS-F highest MVA	Average sequence length	Fragment length (relative)	Average fragment length
Influenza A virus	99.47%	1,468	60%	845
Dengue virus	100.00%	10,991	10%	1,055
Hepatitis B virus	100.00%	3,227	30%	963

Following the calculation used when generating sequence fragments, 60% of 1,409 is 845, 30% of 3,210 is 963, and 10% of 10,559 is 1,055 (note that for each calculation, the floor of the end result was taken, as this is how get_pairs.py of DeLUCS-F determines relative fragment length: by multiplying each sequence's length by the fragment length, and then taking the floor value).

Figure 4.6: The effect of fragment length on the average of the individual accuracies of the ten neural networks of DeLUCS-F for the datasets consisting of viral genomes and genes.



The effect of relative fragment length on the average of the individual accuracies of the ten independently trained neural networks of DeLUCS-F, compared to the same of DeLUCS, for the viral genomes and genes datasets. As with the datasets from Test I and from Test II, the average accuracies of the neural networks of DeLUCS and DeLUCS-F are lower those obtained by the majority vote for each pipeline.

Table 4.8: Comparison between the highest average of individual accuracies obtained by DeLUCS-F, and the average of individual accuracies obtained by DeLUCS for the datasets consisting of viral genomes and genes.

Dataset	DeLUCS AIA	DeLUCS-F highest AIA	Fragment length (relative)	Number of mimics/ fragments
Influenza A virus	86.13%	88.42%	30%	3
Dengue virus	92.89%	99.96%	30%	3
Hepatitis B virus	99.44%	98.64%	90%	3

Each of the scores for DeLUCS-F are the highest of the average of individual accuracies (AIA) (i.e., the average of the accuracies of the ten independently trained neural networks), obtained from the nine trials for each dataset (i.e., for each relative fragment lengths, from 10% to 90%).

4.4 Hyperparameter search on Cypriniformes

Given that the modifications to the DeLUCS pipeline affected the performance of the model for the datasets, it seemed reasonable that it may have also affected the relationship between the hyperparameters and the performance of the model. Thus, given that DeLUCS-F had the lowest majority vote accuracy on the Cypriniformes dataset (and thus DeLUCS-F had the most room for improvement on this dataset), a grid search was performed on the hyperparameters. A total of 270 tests were run, with the following hyperparameter values:

$k = \{3, 4, 5, 6, 7\}$
$n\_\text{fragments} = \{3, 4, 5, 6, 7, 8\}$
$\text{length} = \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%\}$

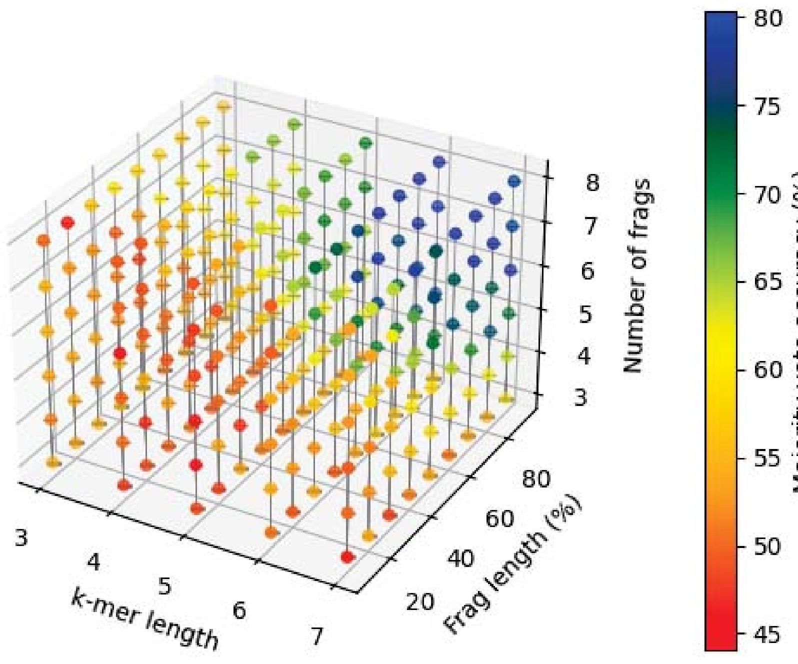
Note that a grid search was employed to obtain a clear visualization of the effects of varying these hyperparameters. However, should one be interested in finding the optimal hyperparameters, iterating through increasingly fine random searches is recommended instead; beginning with a wide and sparse search, followed by increasingly narrow and dense searches on the regions that showed the best accuracies.

Figure 4.7 demonstrates the effects of the variations in the three hyperparameters (shown as the *x*-, *y*-, and *z*-axes) on the majority vote accuracy (shown as the colour mapping).

As shown in Figure 4.7, there seems to be an overall trend where an increase in all of *k*, length, and the number of fragments lead to an increase in performance. In the DeLUCS study, it was found that setting $k = 6$ led to the best balance between high performance and low time complexity [7]. Here, the same relationship is observed, where it is only when $k \geq 6$ that the accuracies begin to exceed 75%.

In the DeLUCS study, it had also been found that setting the number of mimics to 8 led to the best performance in clustering among the sequences of the order Cypriniformes [7]; here, a similar observation is noted, in that accuracies surpassing 75% were obtained by models with a number

Figure 4.7: The effects of the value of *k*, the fragment length, and the number of fragments used on the majority vote accuracy of DeLUCS-F for the dataset consisting of the complete mitochondrial genomes of Cypriniformes.



Along the *x*-, *y*, and *z*-axes are the '*k*-mer length, the 'Frag length', and the 'Number of frags.' The '*k*-mer length' (3, 4, 5, 6, 7) denotes the value of *k*, that is, the length of the *k*-mers used. The 'Number of frags' (3, 4, 5, 6, 7, 8) denote how many fragments of each sequence were calculated, and thus how many sequence/fragment pairs were used to train each neural networks. The values of 'Frag length' (10, 20, 30, 40, 50, 60, 70, 80, 90) denotes the fragments' relative lengths, as a percentage of those fragments' corresponding sequences. Colour is used in this plot to denote the accuracy of the majority vote, ranging from red (45%) to blue (80%). Grey vertical lines appearing on the major horizontal grid marks, which 'skewered' all points, were added for better interpretability.

of fragments as low as 5, although the top five models (all with accuracies above 79%) had eight fragments, and $k = 7$. The highest accuracy, 80%, was obtained with $k = 7$, eight fragments, and a relative fragment length of 80%.

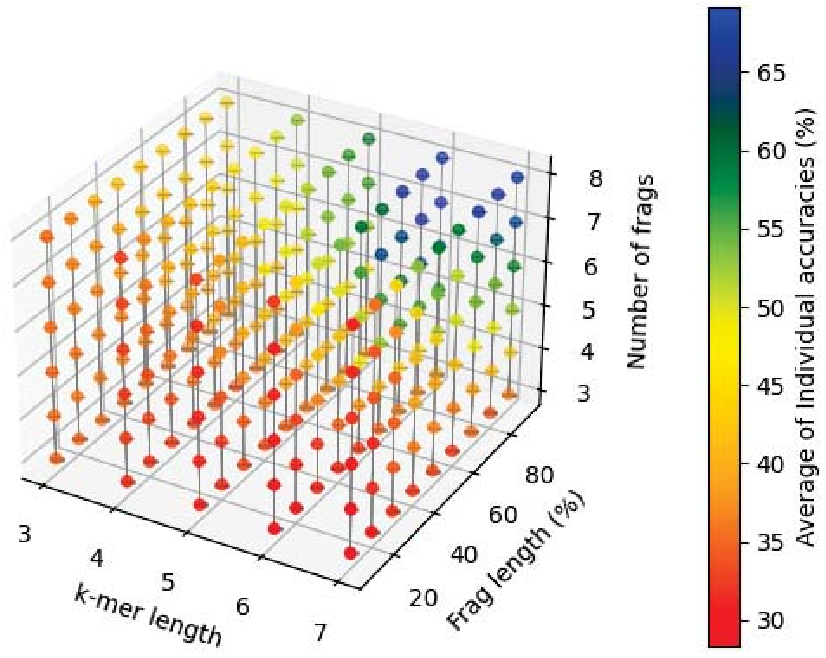
Notably, there is reasonable variation in how lengths affected the majority vote accuracy, however, the overall trend was that longer fragment length tended to result in a higher majority vote accuracy. Accuracies surpassing 75% were obtained by models using fragments of at least 50% of the length of the entire sequence; while relative fragment lengths below 50% seemed to be too little information for the model to obtain reasonable accuracy, regardless of the number of fragments, or the *k* values.

Some of the worst accuracies (around 45%) were obtained when the length was low, and either the *k* value or the number of fragments were high while the other was low. Similarly, when both the *k* value and the number of fragments were low, increasing the length of the fragments did not seem to result in an increase in accuracy. This suggests that for the Cypriniformes dataset, the overall success of the pipeline's clustering (measured by the accuracy of its majority vote) is contingent on having at least four or more fragments, a fragment length of at least 30% of each sequence, and having a *k* value of at least four.

It is worth noting that:

- All trials were run with all random seeds fixed at 0. It would be worth investigating, perhaps, the average majority vote accuracy of multiple models with the same hyperparameter setting, for each hyperparameter variation, using non-fixed random seeds.
- This type of 3D plot that allows one to discern information regarding each datapoint (rather than simply noting information regarding clusters of points), with three independent variables and one dependent variable, becomes quite difficult to interpret with any more than 200 datapoints, at its current resolution. With the 270 datapoints of this test, it is noticeably difficult to distinguish which datapoint belongs to which set of hyperparameters, though it is still possible. In the future, it would be worth performing a wider but sparser grid search in order to visualize the effects of the varying hyperparameters on the accuracy, as was done here, followed by a narrower grid search in the region with the highest scores. This could allow for better interpretability, as well as to cover more meaningful variation in the hyperparameters.
- Similar to the other tests, Figure 4.8 displays the effect of variation in the hyperparameters on the average of the individual accuracies. While the highest average of individual accuracies is much lower than the highest majority vote accuracy, a similar pattern is noticed as in the plot displaying the effect of these hyperparameters on the majority vote accuracy. That is, the highest accuracies of individual neural networks were obtained from a combination of high *k* values, larger number of fragments, and longer fragment lengths.

Figure 4.8: The effects of the value of *k*, the fragment length, and the number of fragments used on the average of the individual accuracies of the ten neural networks of DeLUCS-F for the dataset consisting of the complete mitochondrial genomes of Cypriniformes.



Along the *x*-, *y*, and *z*-axes are the '*k*-mer length, the 'Frag length', and the 'Number of frags.' The '*k*-mer length' (3, 4, 5, 6, 7) denotes the value of *k*, i.e., the length of the *k*-mers used. The 'Number of frags' (3, 4, 5, 6, 7, 8) denote how many fragments of each sequence were calculated, and thus how many sequence/fragment pairs were used to train each neural networks. The values of 'Frag length' (10, 20, 30, 40, 50, 60, 70, 80, 90) denotes the fragments' relative lengths, as a percentage of those fragments' corresponding sequences. Colour is used in this plot to denote the average of the individual accuracies of the ten independently trained neural networks, ranging from red (30%) to blue (70%). Grey vertical lines appearing on the major horizontal grid marks, which 'skewered' all points, were added for better interpretability.

Chapter 5

Discussion

DeLUCS and DeLUCS-F outperformed the classic clustering approaches, K-Means++ and Gaussian Mixture Models, with respect to accuracy across all datasets. DeLUCS-F outperformed DeLUCS in seven datasets, tied in three with perfect accuracies (Actinopterygii, *Dengue virus*, and *Hepatitis B virus*), and was outperformed in one dataset (Cyprinidae), where it was outscored by only 0.01%. Both DeLUCS and DeLUCS-F were outperformed by the supervised deep neural network (which was the supervised analogue of one of the ten neural networks used by DeLUCS) on all but two tests (Actinopterygii, and Cyprinidae), with the supervised learner often outperforming the unsupervised approaches by a large margin. However, the supervised learning method is relying on labels which are subject to dispute and human error, frequently change, may not be connected to a ground truth [7]; all these issues may be propagated by such a model. For example, such a model may be considered highly accurate with the current taxonomy, but should there be a shift, it may be negatively affected; that is, a supervised model is only as good as our current understanding of taxonomy, and thus does not yield as many interesting insights as unsupervised techniques. By a similar argument, where accuracy cannot necessarily be trusted above all else due to such issues with respect to the labels in this field, it would be a mistake to conclude that DeLUCS-F is necessarily better than DeLUCS simply because it scores higher accuracies on most tests. It would be worth investigating the differences in performances between DeLUCS and DeLUCS-F using other measures of clustering performances, as well as visualizing and comparing the clusters created by both pipelines.

As shown in the Bacteria dataset and the Cypriniformes dataset, it is clear that DeLUCS-F still struggles with getting stuck in local optima, and thus further modifications should be made. Some potential modifications that could potentially raise its performance, and which should be considered in future works include:

- Hyperparameter tuning of the neural networks (loss function, learning rate, number of layers, neurons per layer, activation functions, batch size, number of iterations, etc.).

- Instead of sequential models, employ functional models (i.e., instead of layers being linear, using a directed acyclic graph of layers).
- In place of the current neural networks, using convolutional networks, and instead of using pairs of normalized *k*-mer counts as input, using FCGRs (Frequency Chaos Game Representations, which uniquely map sequences to a 2D space) as input (note that these can be generated from normalized *k*-mer counts, and both the scripts of both DeLUCS and DeLUCS-F contain code that accomplishes this).
- Replacement of mimic sequences with sequence fragments in the iDeLUCS pipeline, which is an improved version of the DeLUCS pipeline [6].
- Implementation of soft voting scheme (i.e., the vote for neural networks with higher confidence for a given data point have more weight) rather than the current hard voting scheme (where currently, all neural networks have equally weighted votes).

Additionally, computation speed could be improved by either:

- Parallelizing the training of the neural networks, potentially increasing the speed of step three (EvaluateDeLUCS.py) ten-fold.
- Increasing both the accuracy and stability of the individual neural networks, such that majority voting is no longer necessary, and training only one neural network is sufficient.

Another point of interest is regarding how Test I was performed. In the DeLUCS study, it was described as following a decision-tree approach, whereby for each taxonomic level, the subject of the following test would be the cluster with the most available sequences [7]. However, the decision aspect is based on data availability. It would be interesting to investigate, in a future study, should sufficient data be available, the following version of a decision-tree approach:

> At the end of each clustering, the pipeline is then run on each cluster, until a given endpoint is reached (such as surpassing a depth of five levels, or reaching a cluster size of less than 150). At the end of the various tests, the results could then be visualized, and compared to the existing taxonomic labels, by colouring each point with its true taxonomic label, and assigning shapes to each point using the cluster labels.

The above would yield insights into both the types of incorrect predictions made by DeLUCS-F, as well as investigating the possibility of alternative methods of clustering the sequences, other than using the standard taxonomy. As noted by Kraichak et al. [34], the existing taxonomic labels, while intending to group together organisms of similar features with the assumption that they will also be genetically similar, are arbitrary; thus, newer methods, such as using the time of divergence between related organisms as a measure, are worth consideration . Thus, the use of DeLUCS-F along with the above-described test could yield interesting insights into a novel type of structure that could be used to rank the relative similarities of organisms. However, before such a test can be run, given that different hyperparameter values (i.e., the k-mer length, number of fragments, and the fragment length) would lead to different clusters, further work needs to be done to investigate the relationship between the hyperparameters and the shapes and sizes of the discovered clusters.

In the DeLUCS study [7], a k value of 6 was found to be optimal (in terms of the tradeoff between accuracy and computational cost), while the number of mimics to be used depended on the minimum number of sequences per cluster. In this study, through the hyperparameter search on the dataset consisting of Cypriniformes, it was found that a k value of 6 was sufficient for a reasonable accuracy (at $k = 6$, accuracies began to exceed 75%, with the highest accuracy for that hyperparameter search having been 80%), with the highest accuracy being obtained at $k = 7$. Similarly, while having five fragments was sufficient for reasonable accuracies (above 75%), the best accuracy was found with eight fragments. Thus, the results of this hyperparameter search show that both DeLUCS and DeLUCS-F seem to share the same relationship between their hyperparameter settings and their performances. However, DeLUCS-F adds a new hyperparameter that DeLUCS did not use; that is, the length of the sequence fragments.

In that hyperparameter search, it was found that a relative fragment length of 50% of each sequence's length was sufficient for a reasonable accuracy (above 75%), while a relative fragment length of 80% was required to achieve the highest accuracy (80%). However, the hyperparameter search was performed on the dataset consisting of Cypriniformes, which had seven clusters; it was found, in Tests I, II, and III, that the fragment length required to achieve optimal accuracy depended on the number of clusters within the dataset. Throughout the three tests, compared to other datasets within the same test, optimal accuracy had been achieved by DeLUCS-F using shorter relative fragment lengths when the dataset consisted of four or less distinct clusters. This may have to do with the fact that larger amounts of discriminatory information are required when there are more distinct clusters into which the sequences should be clustered (although, the exact relative fragment length depended on the type of data, and differed between the three tests, involving mitochondrial DNA, bacterial genome segments, and viral genomes and genes). Whether this pattern holds for other datasets would be interesting to explore, as this could yield further insights into the exact nature of this pattern, and could potentially reveal the true reason for this relation.

In terms of why DeLUCS-F outperforms the DeLUCS pipeline (with both fixes) on most datasets in terms of majority vote accuracy of the ten neural networks, it is worth first noting that it underperformed the DeLUCS pipeline in terms of the average of the individual accuracies obtained by the ten neural networks. Thus, each individual neural network had more incorrect predictions in the DeLUCS-F pipeline than the DeLUCS pipeline. However, as stated in [8, p. 193], the fact of a majority voter scoring higher than the individual voters is dependent on the voters being independent, and making uncorrelated errors. Thus, while the individual neural networks of DeLUCS-F were more likely to make errors than the neural networks of DeLUCS, it is likely that they are less prone to making the same type of errors as each other, and thus, more of their incorrect predictions were likely cancelled out by the majority vote, than for the neural networks of DeLUCS. However, further studies should investigate the predictions of each neural network for both DeLUCS and DeLUCS-F across the datasets to verify this.

Understanding the exact reasons for the overall predictions made by the pipeline, is made challenging as a result of both the majority vote, and the use of neural networks, both of which cause explainability to be lost. Though, unlike the DeLUCS pipeline, the DeLUCS-F pipeline does not rely on synthetic data, and thus while the decision-making process of the model remains a black box, it is basing these decisions on real data (namely, the normalized *k*-mer counts of the sequences and sequence fragments). In [11], the study found that while synthetic data can be useful for the sake of augmenting a dataset to increase a model's performance, a model was able to be trained to perfectly distinguish between genuine and artificial DNA sequences, which were generated using a similar algorithm involving transitions and transversions on the sequences used in that study; such a finding is a prime example of why one must be wary when using synthetic data to classify or cluster DNA sequences, especially when the model that uses such artificial data is largely uninterpretable. That being said, the fact that the artificial data was perfectly distinguishable from the original data in [11] may have been a unique finding, and future studies should consider training a model to determine whether the mimic sequences generated in the DeLUCS pipeline can also be distinguished from the original sequences.

Chapter 6

Conclusion

In the field of taxonomic assignment, alignment-free solutions are preferred to alignment-based for numerous reasons, most notably the fact that the time complexity of alignment-based solutions make them unfeasible for many applications. DeLUCS had outperformed the standard models for unsupervised learning in taxonomic assignment [7]. By using unsupervised learning in place of supervised learning, it is expected to learn hidden patterns within the data, and its predictions are unaffected by labelling, meaning that it can be used when labels are unavailable. Equally, it avoids the time-consuming process required to create those labels, and it is free from the effect of any errors introduced in the labelling process. Moreover, by using deep learning, it leverages the substantial amount of data available in this field, allowing for complex patterns to be learned, and for large amounts of data to be handled (as of 2022, DeLUCS was considered to have clustered the largest real datasets to date [7]). However, it employs neural networks and majority voting, both known to reduce the interpretability of the decisions made by a model; this issue is further compounded by the use of artificial mimic sequences, making it unclear whether the model is learning patterns based on the real data, or based on the generated data.

In DeLUCS-F, the DeLUCS pipeline was modified to use sequence fragments in place of the mimic sequences of the DeLUCS pipeline, resolving the issue of its reliance on artificial data. This change, in addition to increasing its trustworthiness by removing its reliance on artificially generated data (and thus ensuring that all patterns learned were patterns that existed in real data, rather than potentially in synthetic data), has led to the overall increase in performance and computation speed of the entire pipeline.

References

[1] G. S. Randhawa, M. P. Soltysiak, H. E. Roz, C. P. de Souza, K. A. Hill, and L. Kari. Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: COVID-19 case study. *PLoS ONE*, 15(4):e0232391, 2020.

[2] D. Emerson, L. Agulto, H. Liu, and L. Liu. Identifying characterizing bacteria in an era of genomics proteomics. *BioScience*, 58(10):925–936, 2008.

[3] C. C. Thompson, L. Chimetto, R. A. Edwards, J. Swings, E. Stackebrandt, and F. L. Thompson. Microbial genomic taxonomy. *BMC Genomics*, 14:913, 2013.

[4] R. De-Kayne, D. Frei, R. Greenway, S. L. Mendes, C. Retel, and P. G. Feulner. Sequencing platform shifts provide opportunities but pose challenges for combining genomic data sets. *Molecular Ecology Resources*, 21(3):653–660, 2021.

[5] J. Avila Cartes, S. Anand, S. Ciccolella, P. Bonizzoni, and G. D. Vedova. Accurate fast clade assignment via deep learning frequency chaos game representation. *Oxford University Press GigaScience*, 12:giac119, 2022.

[6] P. Millan Ariás, K. A. Hill, and L. Kari. iDeLUCS: a deep learning interactive tool for alignment-free clustering of DNA sequences. *Bioinformatics*, 39(9):btad508, 2023.

[7] P. Millan Ariás, F. Alipour, K. A. Hill, and L. Kari. Delucs: Deep learning for unsupervised clustering of DNA sequences. *PLoS ONE*, 17(1):e0261531, 2022.

[8] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, & Tensorflow*. O'Reilly Media Incorperated, Sebastopol, 2nd edition, 2019.

[9] R. Karim et al. Deep learning-based clustering approaches for bioinformatics. *Briefings in Bioinformatics*, 22(1):393–415, 2021.

[10] A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowsky. Alignment-free sequence comparison: benefits and applications and tools. *Genome Biology*, 18:1–17, 2017.

[11] S. Solis-Reyes, M. Avino, A. Poon, and L. Kari. An open-source k-mer based machine learning tool for fast accurate subtyping of HIV-1 genomes. *PLoS ONE*, 13(11):e0206409, 2018.

[12] J. Ren et al. Alignment-free sequence analysis applications. *Annual Review of Biomedical Data Science*, 1:93–114, 2018.

[13] S. Vinga and J. Almeida. Alignment-free sequence comparison—a review. *Bioinformatics*, 19(4):513–523, 2002.

[14] J. Liu, J. Li, H. Wang, and J. Yan. Application of deep learning in genomics. *Science China Life Sciences*, 63:1860–1878, 2020.

[15] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti. A primer on deep learning in genomics. *Nature genetics*, 51(1):12–18, 2019.

[16] V. Nasteski. An overview of the supervised machine learning methods. *Horizons Series B*, 4:51–62, 2017.

[17] C. Zhang, R. Dong, X. Zhang, and W. Li. A comparative analysis of supervised/unsupervised algorithms in PHM application. In *Global Reliability Prognostics Health Management (PHM-Yantai)*, pages 1–5, 2022.

[18] G. S. Randhawa, K. A. Hill, and L. Kari. Ml-DSP: Machine Learning with Digital Signal Processing for ultrafast and accurate and scalable genome classification at all taxonomic levels. *BMC Genomics*, 20:1–21, 2019.

[19] J. Rajkumari, P. Katiyar, S. Dheeman, P. Pandey, and D. K. Maheshwari. The changing paradigm of rhizobial taxonomy its systematic growth upto postgenomic technologies. *World Journal of Microbiology Biotechnology*, 38(11):206, 2022.

[20] W. L. Applequist. A brief review of recent controversies in the taxonomy nomenclature of Sambucus nigra sensu lato. In *International Symposium on Elderberry*, volume 1061, pages 25–33, 2013.

[21] J. Bao, R. Yuan, and Z. Bao. An improved alignment-free model for dna sequence similarity metric. *BMC Bioinformatics*, 15(1):105, 2014.

[22] A. Bustamam, H. Tasman, N. Yuniarti, F. Frisca, and I. Mursidah. Application of k-means clustering algorithm in grouping the DNA sequences of Hepatitis B virus (HBV). *AIP Conference Proceedings*, 1862(1), 2017.

[23] N. Aleb and N. Labidi. An improved K-means algorithm for DNA sequence clustering. In *26th International Workshop on Database Expert Systems Applications (DEXA)*, pages 39–42, 2015.

[24] Z. Huo and G. Tseng. Integrative sparse K-means with overlapping group lasso in genomic applications for disease subtype discovery. *The Annals of Applied Statistics*, 11(2):1011, 2017.

[25] G. Mendizabal-Ruiz, I. Román-Godínez, S. Torres-Ramos, R. A. Salido-Ruiz, H. Vélez-Pérez, and J. Alejandro Morales. Genomic signal processing for DNA sequence clustering. *PeerJ*, 6:e4264, 2018.

[26] M. Akhtar, E. Ambikairajah, and J. Epps. GMM-based classification of genomic sequences. In *15th International Conference on Digital Signal Processing*, pages 103–106, 2007.

[27] W. Fletcher and Z. Yang. INDELible: A flexible simulator of biological sequence evolution. *Molecular Biology Evolution*, 26(8):1879–1888, 2009.

[28] X. Ji J. F. Henriques and A. Vedaldi. Invariant information clustering for unsupervised image classification segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[30] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[31] E. W. Sayers et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acid Research*, 50:D20–D26, 2022.

[32] P. A. Chaumeil, A. J. Mussig, P. Hugenholtz, and D. H. Parks. GTDB-Tk: a toolkit to classify genomes with the Genome Taxonomy Database. *Bioinformatics*, 36(6):1925–1927, 2020.

[33] J. Hayer, F. Jadeau, G. Deleage, A. Kay, F. Zoulim, and C. Comet. HBVdb: a knowledge database for Hepatitis B virus. *Nucleic Acids Research*, 41:D566–D570, 2013.

[34] E. Kraichak, A. Crespo, P. K. Divakar, S. D. Leavitt, and H. T. Lumbsch. A temporal banding approach for consistent taxonomic ranking above the species level. *Scientific Reports*, 23(1):2297, 2017.

Appendix A

Inconsistencies in the original pipeline

This appendix describes the inconsistencies between the DeLUCS pipeline as described in the DeLUCS study [7], and the version of the pipeline as downloaded from the official DeLUCS GitHub repository (the most recent commit being 70e5ce7, as of April 2024). These two inconsistencies involve the number of pairs that were fed to the neural networks (supposed to be n but was instead $3(n - 2)$), and the fact that the pairs consisted of two copies of the same sequence, rather than sequences and mimics (i.e., the mimics had been discarded entirely).

No trends were observed across datasets as a result of implementing the various combinations of the described changes. Throughout this study, the version of DeLUCS to which DeLUCS-F is compared is the version with both changes made, described in this appendix, which includes changes to get_pairs.py (to ensure only n pairs are created), to PytorchUtils.py (to ensure that the mimic sequences are not overwritten by the original sequences), and also a minor change to mimics.py (to ensure the use of fixed random seeds, so that the results of all experiments are perfectly reproducible). This is because it was felt most appropriate to compare DeLUCS-F to the version of DeLUCS as described in the DeLUCS study [7], and because both these changes were implemented in the DeLUCS-F pipeline (i.e., DeLUCS-F uses pairs of sequences and fragments, and uses n such pairs).

A.1 Description of inconsistencies within the DeLUCS pipeline

Two inconsistencies were noticed between the original DeLUCS pipeline as described in the DeLUCS study [7] and as downloaded from the official DeLUCS GitHub repository. These are:

- The mimic sequences were being overwritten (i.e., discarded) before the neural networks had received them.

- This issue occurs within step three (the training and evaluation of the neural networks), as described in Chapter 3 of this study. After the training set (consisting of the pairs of sequence/mimic normalized k-mer counts) is loaded in, it is first scaled, and then fed to the constructor for a Seq_data object (line 152 of the original EvaluateDeLUCS.py script); Seq_data is a class within the helper script, PytorchUtils.py. This Seq_data object is then fed to the neural networks as their training set. However, line 29 of PytorchUtils.py (from within the Seq_data class) is the following:

  ```
  sample = {'true': self.data[idx, 0, :], 'modified':
  self.data[idx, 0, :]} #<--- We can enforce the prediction of
  #same vector
  ```

  This Seq_data object contains two columns: the 'true' column, and the 'modified' column. Note that self.data[idx,0,:] refers to the portion of the training set that contains the normalized k-mer counts of the original sequences, whereas self.data[idx,1,:] would refer to the portion that contains the normalized k-mer counts of the mimic sequences. Since both the 'true' and 'modified' columns are set to the same set of values (i.e., the normalized k-mer counts of only the original sequences), and since the neural networks use the 'modified' columns where the code refers to the use of mimic sequences, it appears that the mimic sequences had been unintentionally overwritten. It is worth noting that the correction to this line,

  ```
  sample = {'true': self.data[idx, 0, :], 'modified':
  self.data[idx, 1, :]}
  ```

  appears above the line that is used, however, it is commented out. The first commit in which the correct (but commented out) line appears is 8edea3d (titled "Enforcing True Predictions"), from May 12, 2022, where the above commented-out line was added, and the comment "<--- We can enforce the prediction of same vector" was added to the line that is used. Using the history of commits on the official DeLUCS GitHub repository, no version of DeLUCS which used the correct line was found.

- However, in the iDeLUCS pipeline, which is a modification of the original pipeline by some of the same authors, the 26th commit (285bd01, "Updated model", committed on July 14, 2022) contains the correct line at line 302 of utils.py, however, with the same comment that had appeared on the problematic line from the original DeLUCS pipeline, "<--- We can enforce the prediction of same vector". All subsequent commits also contained this correction. It is worth noting that prior to commit 26, which included the corrected line, this line (neither the incorrect nor the correct version) did not occur at all throughout the entire GitHub repository; that is, in the first instance in which this line was introduced to the iDeLUCS GitHub repository, it had already been corrected.

- Instead of creating n sequence/mimic pairs for each sequence, there exist $(n-2)$ duplicates of the pairs containing the transition mimics and the pairs containing the transversion mimics. Thus, there are instead $3(n-2)$ sequence/mimic pairs in total, where only n of those pairs are unique.
  - This issue occurs within step two (the creation of the mimic sequences), as described in Chapter 3 of this study. When get_pairs.py is run with the argument "modify" set to "mutation" (which is the default setting, and is the setting used in the pipeline as described in the DeLUCS study [7], as well as in this study), an inner loop causes line 56 to run $(n-2)$ times (where there are n mimics, and at least 3 mimics must be used). This line is responsible for adding to the training data the sequence/mimic pairs, including the pair containing the transition mimic, the pair containing the transversion mimic, and the pair containing a mimic created through both transitions and transversions. Thus, due to the fact that these pairs are added to the training data within an inner loop that runs $(n-2)$ times, the training data will contain $3(n-2)$ sequence/mimic pairs, rather than the intended n pairs. However, only the mimic created through both transitions and transversions is generated within the inner loop, whereas the transversion mimic and the transition mimic were generated beforehand; meaning that these are each added to the training data $(n-2)$ times, meaning that the training data contains $(2n-6)$ redundant copies. Thus, if only 3 mimics are used (with 3 being the minimum allowed when using the "mutation" setting), there will be no duplicate data; however, once the number of mimics exceeds 3, duplicate data will be added to the training data, for each sequence.
  - Once again, it was found that in the iDeLUCS pipeline, the 26th commit (285bd01, "Updated model", committed on July 14, 2022) contains corrected versions of the original line, at lines 252, 258, and 265 of utils.py; these correct lines are present in all subsequent commits. Similarly, it was found that this commit was the first instance in which this segment of code was introduced to the GitHub repository, and it had already been corrected.

It is worth noting that the original DeLUCS pipeline with only the get_pairs.py fix (i.e., producing only n pairs, but still overwriting mimic sequences with the original sequences) is equivalent to running the DeLUCS-F pipeline using the entire length of the sequence (i.e., setting relative length to 100%). Similarly, for three mimics, the original DeLUCS pipeline without any fixes is also equivalent to running the DeLUCS-F pipeline using the entire length of the sequence, using three fragments.

A.2 Comparing results of the original and changed versions of the pipeline

> Note: For these tests with the original pipeline, a line needed to be added to mimics.py to all versions (including the original version) to force NumPy to use a fixed seed when generating random numbers, thus ensuring the perfect reproducibility of all results. The random seeds had already been fixed to 0 in all other relevant scripts in the original version of DeLUCS, as downloaded from the official GitHub repository. Thus, this change merely ensured that mimics.py followed the same standard as the other scripts, in terms of reproducibility.

For all tests involving DeLUCS in this appendix, as well as throughout this entire study, this altered version of mimics.py had been used.

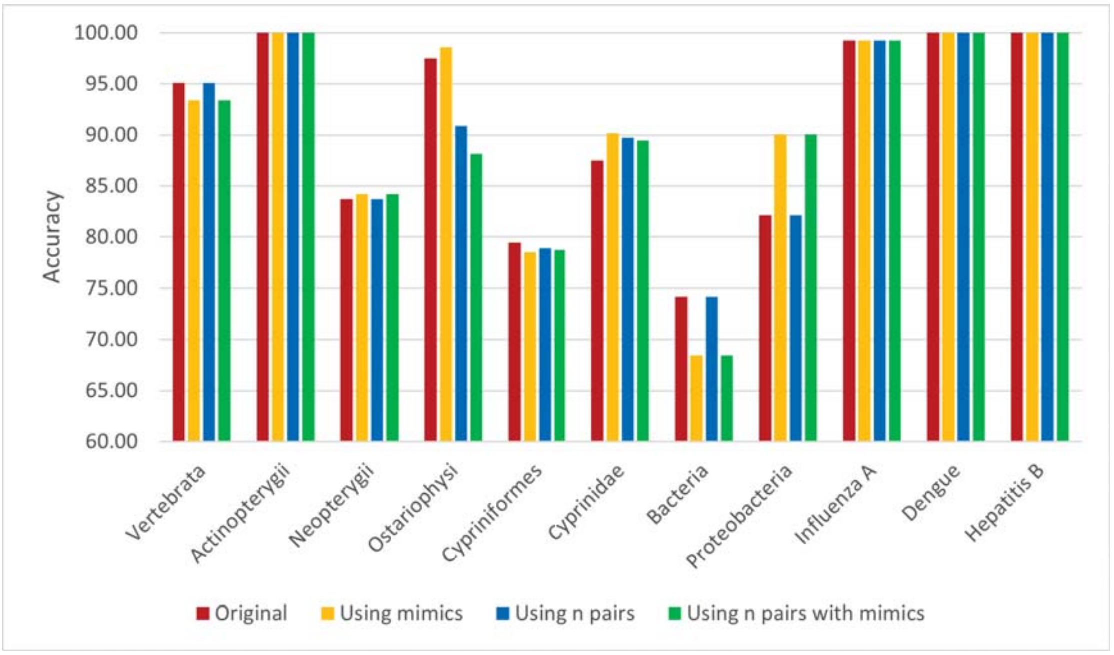
Table A.1 describes the accuracies of the majority vote (of the ten independently trained neural networks) of the DeLUCS pipeline with various changes implemented, as obtained by running the pipeline once on each dataset, with all random seeds fixed at 0. It is worth noting that slightly different results could be obtained by running the pipeline with different seeds, and future works investigating this difference between the original and changed versions of the pipeline are advised to unfix the random seeds, and average the results of ten trials.

Table A.1: Majority vote accuracies of the original DeLUCS pipeline, the version with mimics, the version with n pairs, and of the version with n pairs and mimics.

Dataset	Original	Using mimics	Using n pairs	Using n pairs with mimics	Number of pairs per sequence
Vertebrata	95.12%	93.40%	95.12%	93.40%	3
Actinopterygii	100.00%	100.00%	100.00%	100.00%	8
Neopterygii	83.73%	84.20%	83.73%	84.20%	3
Ostariophysi	97.52%	98.62%	90.91%	88.15%	8
Cypriniformes	79.45%	78.53%	78.90%	78.72%	8
Cyprinidae	87.47%	90.16%	89.71%	89.49%	8
Bacteria	74.19%	68.44%	74.19%	68.44%	3
Proteobacteria	82.13%	90.06%	82.13%	90.06%	3
Influenza A virus	99.26%	99.26%	99.26%	99.26%	3
Dengue virus	100.00%	100.00%	100.00%	100.00%	3
Hepatitis B virus	100.00%	100.00%	100.00%	100.00%	3

This table describes the majority vote accuracy of the DeLUCS pipeline with all variations of changes made. All tests were run for each dataset using the hyperparameters ($k = 6$, and using the same number of pairs), as described in the DeLUCS study [7]. "Original" uses $3(n-2)$ sequence/sequence pairs as input for the neural networks; this is how the pipeline is coded as per the official GitHub repository. "Using mimics" refers to the change made to PytorchUtils.py, and uses $3(n-2)$ sequence/mimic pairs. "Using n pairs" refers to the change made to get_pairs.py, and uses n sequence/sequence pairs. "Using n pairs with mimics" refers to both changes made, and uses n sequence/mimic pairs; this is how the pipeline is described in the DeLUCS study [7].

Figure A.1: Visualization of majority vote accuracies of the original DeLUCS pipeline, the version with mimics, the version with n pairs, and of the version with n pairs and mimics.



The effect of relative fragment length on the majority vote accuracy of DeLUCS-F, compared to the majority vote accuracy of DeLUCS, for the viral genomes and genes datasets.

As shown in Figure A.1, with respect to the majority vote accuracy, the original version of the pipeline as downloaded from the GitHub repository (which had been discarding the mimic sequences, and instead of feeding the neural networks with n sequence/mimic pairs, was feeding them with 3(n-2) sequence/sequence pairs), outperforms the various changed versions in clustering amongst organisms of the order Cypriniformes, and ties for highest performance with the version that fixes only the number of pairs for clustering of Vertebrata, and Bacteria. The version with both changes outperforms the original version for the Proteobacteria, Cyprinidae, and Neopterygii. The performances across the viral data and the class Actinopterygii were completely unaffected by these changes, in terms of majority vote accuracy. However, no clear and significant pattern was discovered in terms of how the various combinations of changes were affecting the performances of the pipeline.

The average of the individual accuracies obtained by the ten independently trained neural networks were then compared for each set of changes, for each dataset. The results of this test are displayed in Table A.2, and these results are visualized in Figure A.2.

As shown in Figure A.2, there is less variation in the average of individual accuracies in the same datasets which in Figure A.1 caused greater variation in the majority vote accuracies (Bacteria, Proteobacteria, and Ostariophysi), yet more variation in the average of individual accuracies in datasets that caused little variation in the majority vote accuracy (Actinopterygii, *Influenza A*

Table A.2: Average of individual accuracies of the neural networks of the original DeLUCS pipeline, the version with mimics, the version with n pairs, and of the version with n pairs and mimics.

Dataset	Original	Using mimics	Using n pairs	Using n pairs with mimics	Number of pairs per sequence
Vertebrata	84.93%	83.00%	84.93%	83.00%	3
Actinopterygii	94.07%	93.01%	97.35%	97.79%	8
Neopterygii	69.69%	68.39%	69.69%	68.39%	3
Ostariophysi	72.20%	73.44%	71.54%	71.38%	8
Cypriniformes	66.84%	65.60%	68.04%	68.07%	8
Cyprinidae	81.14%	80.65%	76.78%	80.94%	8
Bacteria	60.94%	61.52%	60.94%	61.52%	3
Proteobacteria	72.67%	73.54%	72.67%	73.54%	3
Influenza A virus	86.58%	86.13%	86.58%	86.13%	3
Dengue virus	94.29%	92.89%	94.29%	92.89%	3
Hepatitis B virus	99.79%	99.44%	99.79%	99.44%	3

This table describes the average of the individual accuracies of the ten independently trained neural networks of the DeLUCS pipeline with all variations of changes made. All tests were run for each dataset using the hyperparameters ($k = 6$, and using the same number of pairs), as described in the DeLUCS study [7]. "Original" uses $3(n-2)$ sequence/sequence pairs as input for the neural networks; this is how the pipeline is coded as per the official GitHub repository. "Using mimics" refers to the change made to PytorchUtils.py, and uses $3(n-2)$ sequence/mimic pairs. "Using n pairs" refers to the change made to get_pairs.py, and uses n sequence/sequence pairs. "Using n pairs with mimics" refers to both changes made, and uses n sequence/mimic pairs; this is how the pipeline is described in the DeLUCS study [7].

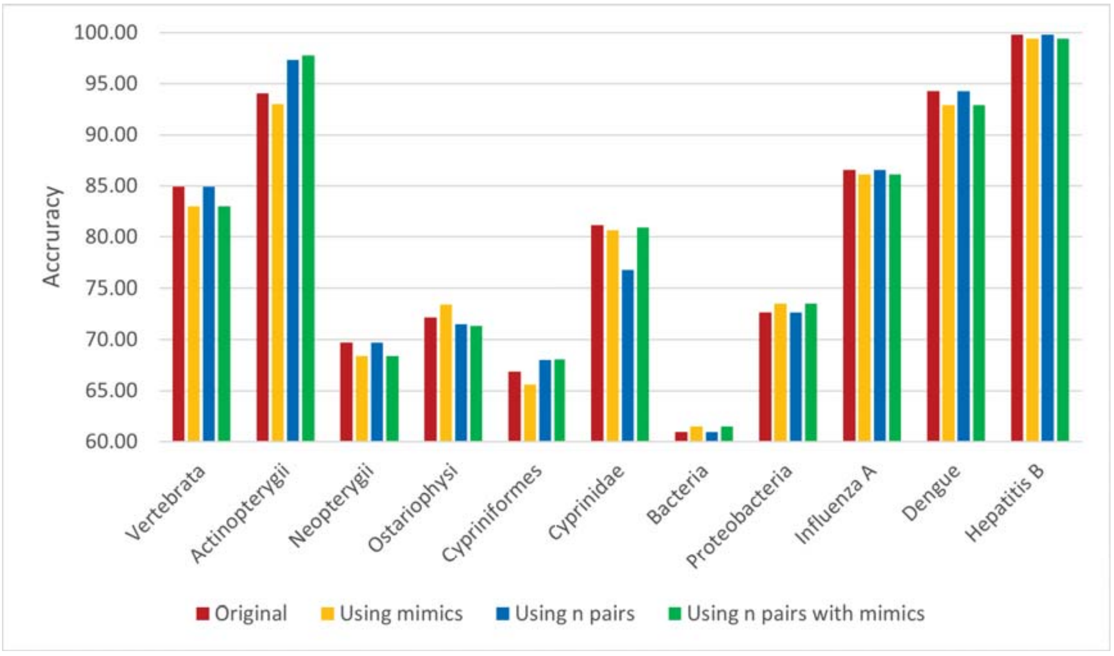
virus, *Dengue virus*, and *Hepatitis B virus*). Once again, however, no clear and significant pattern was discovered in terms of how the various combinations in fixes were affecting the average of individual accuracies.

A.3 Decision to use DeLUCS pipeline with both fixes

There are discrepancies between the performances for DeLUCS that were obtained in this appendix, and the values for DeLUCS reported in the DeLUCS study [7]. This may be due to the fact that while all random seeds were fixed here, according to the code for DeLUCS on the GitHub repository, the random seeds were not fixed for the generation of mimic sequences (in mimics.py, which is called by get_pairs.py). Discrepancies also exist between the values obtained in [7] for both the accuracy of the majority vote, and the average accuracy of ten individual neural networks (shown in Figure 8 of the DeLUCS study [7]), and the values obtained by us for any of the four versions of the DeLUCS pipeline (original, with the fix applied to get_pairs.py, with the fix applied to PytorchUtils.py, and with both fixes applied).

Thus, while it seems reasonable that the values reported in the DeLUCS study were produced by the DeLUCS pipeline as is available on the official GitHub repository, it is uncertain. For example, in the DeLUCS study, the fact is presented that the addition of Gaussian noise to

Figure A.2: Visualization of the average of individual accuracies of the original DeLUCS pipeline, the version with mimics, the version with n pairs, and of the version with n pairs and mimics.



This bar graph displays the average of the individual accuracies of the ten independently trained neural networks of the DeLUCS pipeline with all variations of changes made. All tests were run for each dataset using the hyperparameters ($k = 6$, and using the same number of pairs), as described in the DeLUCS study [7]. "Original" uses $3(n-2)$ sequence/sequence pairs as input for the neural networks; this is how the pipeline is coded as per the official GitHub repository. "Using mimics" refers to the change made to PytorchUtils.py, and uses $3(n-2)$ sequence/mimic pairs. "Using n pairs" refers to the change made to get_pairs.py, and uses n sequence/sequence pairs. "Using n pairs with mimics" refers to both changes made, and uses n sequence/mimic pairs; this is how the pipeline is described in the DeLUCS study [7].

the parameters of the network every 50 epochs increases the final accuracy of that network. However, it was not mentioned in the DeLUCS study that the DeLUCS pipeline adds Gaussian noise every 30 epochs, and had being doing so since commit 0f2fd2a (titled "Add files via upload"), committed to GitHub on February 1, 2021 (noting that the DeLUCS study was published on January 21, 2022). Note that commit 404e43d (titled "Merge remote-tracking branch 'origin/master'"), which was committed on May 3, 2021, the date that the DeLUCS study was received for publication, contains these inconsistencies, however, does not contain the issue regarding the number of mimics generated, as get_pairs.py was limited to generating a fixed number (3) of mimics at that time (i.e., that issue had been introduced after that date).

Since much of the DeLUCS study [7] describes the use of sequence and mimic pairs, and specifically, the use of n such pairs, the version of the DeLUCS pipeline with both fixes applied is used as the basis to which DeLUCS-F is compared.

Appendix B

Setup and replication of results

B.1 Implementation

Experiments were run on the Narval cluster of Compute Canada, with the step involving the training/evaluation of the neural networks being run using a job scheduler, using a single node, with 2500 to 3500MB of memory, 1 GPU (NVIDIA A100-SXM4-40GB), 4 CPU cores (2 x AMD Rome 7532 @ 2.40 GHz 256M cache L3), and between 15 to 25 minutes per test. Both the first and second steps (cleaning the data, and preprocessing the data) were able to be run directly from the command line.

The pipeline was run in Python 3.10.2. Necessary libraries were installed using Pip version 22.0.3. Additionally, the versions of the libraries used were:

- argparse: 1.1
- Biopython: 1.81
- CUDA: 11.7
- itertools: 8.12.0
- Matplotlib: 3.7.0
- NumPy: 1.24.2
- pickle: 4.0
- SciPy: 1.10.1
- Sklearn: 1.2.1
- Torch: 2.0.0

Random seeds (Torch, CUDA, NumPy, and random) for the evaluation step (i.e., the training and testing of the neural networks) throughout all experiments were fixed as 0. Seeds (for the random library) were also fixed at 0 for the data preprocessing step (where the beginning points of fragments of DNA were randomly selected from within each sequence).

The version of the code for the original DeLUCS pipeline, and that served as the base for the modifications described in this study, was the most recent version at the time (as of the time of writing, April 2024) and is commit 760dcfd on GitHub, found at
`github.com/millanp95/DeLUCS/tree/70e5ce7ef199a36a90dc961b959a23e180289d67`
(note that both modifications described in Appendix A were applied to this base). This version was committed on October 29, 2021.

B.2 Replication of results

Cluster computing was used as it was found that while the first two steps (build_dp.py and get_pairs.py) of the pipeline could be run in reasonable time on a personal computer, the third step (EvaluateDeLUCS.py) was prohibitively slow.

When beginning a new session in Narval, the following commands were run each time (after loading in a virtual environment that had been created at the beginning of the project, which can be created with the line "virtualenv /PATH/NAME", and then loaded with the line "source /PATH/NAME):

```
module load StdEnv/2020
module load python
```

As a result of some quality-of-life improvements added to the DeLUCS-F pipeline, the commands required to perform a test have changed. The following are the commands used to generate the results for *Influenza A virus*, with 3 mimics (with the modification set to "mutate"), and $k = 6$, using the original DeLUCS pipeline:

```
python build_dp.py --data_path=../data/Influenza-A/Test\ Files

python get_pairs.py --data_path=../data/Influenza-A/Test\ Files/train.p
--k=6 --modify='mutation' --output=../data/Influenza-A/testing_data.p
--n_mimics=3

python EvaluateDeLUCS.py --data_dir=../data/Influenza-A --
out_dir=../data/Influenza-A/results
```

In order to run the analogous test in DeLUCS-F (for this example, a relative fragment length of 60% is chosen, as that is the length at which optimal performance is reached for the *Influenza A virus* dataset, as defined by the highest accuracy), the commands required are:

```
python build_dp.py --data_path=../data/Influenza-A/Test\ Files

python get_pairs.py --data_path=../data/Influenza-A/Test\ Files/train.p
--k=6 --output=../data/Influenza-A/Test\
Files/testing_data_Influenza-A_k6_n3_l60percent.p --n_fragments=3
--length=0.6

python EvaluateDeLUCS.py --data_dir=../data/Influenza-A/Test\
Files/testing_data --input_filename=testing_data_Influenza-
A_k6_n3_l60percent.p --out_dir=../data/Influenza-A/results
--out_folder=results_k6_n3_l60percent_chunks
```

While the output of the original pipeline was text printed to the console, and a confusion matrix saved to the specified output folder, the modified pipeline does the same, in addition to saving the same outputted text to a .txt file in the specified output folder as well.

Appendix C

Quality-of-life improvements

The quality-of-life improvements (which, in addition to the use of sequence fragments in place of mimics, further differentiate the scripts used in DeLUCS from those used in DeLUCS-F) are as follows, categorized by the step to which they were added:

The quality-of-life improvements (which, in addition to the use of sequence fragments in place of mimics, further differentiate the scripts used in DeLUCS from those used in DeLUCS-F) are as follows, categorized by the step to which they were added:

1. Cleaning the data (build_dp.py):
   - Previously, if any file existed in the directory in which build_dp.py is looking for directories containing FASTA files, it would try to enter that file as if it were a directory, crashing the program. The script now first checks whether a given folder is in fact a directory before exploring it.
   - Additionally, build_dp.py would try to process any file as if it were a FASTA file, also crashing the program. It now checks whether a file is a FASTA before processing it.
2. Preprocessing the data, and generating the fragments (get_pairs.py):
   - To the header of the output file, get_pairs.py now saves information regarding the number of fragments, length of fragments, and *k* value, so that these do not have to be added as arguments when running EvaluateDeLUCS.py.
3. Training and evaluating the neural networks (EvaluateDeLUCS.py):
   - Previously, a user would be unable to change the name of the output of step two from "testing_data" (meaning that if one wanted to run multiple tests with varying hyperparameters on the same dataset, the original output file would be overwritten), as EvaluateDeLUCS.py would previously only accept a file titled "testing_data.p".

This was resolved by adding an argument to EvaluateDeLUCS.py that checks for the input file's name.

- In the DeLUCS pipeline, EvaluateDeLUCS.py was built to only run with $k = 6$, and giving it as input a file that had been created in step two with a k value of anything other than 6 would return an error. This has been fixed (requiring changes to both EvaluateDeLUCS.py and PytorchUtils.py).
- Previously, the confusion matrix would fail to save (and the program would crash) unless there was an output directory named "results" which was used as an argument for the output directory, and within that folder there existed a file named "Confusion Matrix.png". This meant that a user had to know in advance that such a crash would happen, and create the necessary directory and empty image file, before being able to successfully run EvaluateDeLUCS.py without crashing. Now, there are arguments for an output directory, as well as an output folder (useful for running multiple tests at once, and keeping the results separated using folder hierarchies). If an output folder is not specified, the output will be saved to the output directory directly. If a file called "Confusion Matrix.png" does not exist in the output folder, one will be created.
- Previously, the output of EvaluateDeLUCS.py was printed directly to the console; now, in addition to that, a text file is saved to the specified output folder (meaning that when hundreds of tests are run in parallel, it is no longer necessary to decode the (potentially overlapping) results from the console).
- At the end of EvaluateDeLUCS.py, the output now includes the time taken to train and evaluate the neural networks, the average of the individual accuracies of the neural networks, the number of fragments used, the length of the fragments, and the k-value (making it possible for another script to compile the various outputs into a single text file, which another script can then create plots out of, or run analyses on).