AI-Powered Classification and Early Detection of Dengue Virus Lineages for Timely Public Health Response

Emmanuel Nyandu Kagarabi (emmanuelnk@aims.ac.za)

African Institute for Mathematical Sciences (AIMS)

Supervised by:

Dr. Joicymara Xavier

Centre for Epidemic Response and Innovation(CERI), South Africa

Federal University of Vales do Jequitinhonha e Mucuri, Brazil

and

Dr. Houriiyah Tegally

Centre for Epidemic Response and Innovation(CERI), South Africa

Stellenbosch University, South Africa

24 October 2024

Submitted in partial fulfillment of a AI masters degree at AIMS South Africa

Abstract

The Dengue Virus (DENV), responsible for diseases such as dengue fever and severe dengue (or dengue hemorrhagic fever), currently causes over 100 million infections annually worldwide. It is considered one of the most widespread vector-borne viral infections, posing a significant global public health issue due to the high number of deaths it causes, as well as the substantial economic burden associated with its management, particularly in tropical and subtropical regions. In this study, we present an approach for classifying genomic sequences of DENV into lineages and facilitating the early detection of potential new variants. To achieve this, we designed a combination of feature extraction techniques and machine learning as well as deep learning algorithms. Using k-mer encoding, Frequency Chaos Game Representation (FCGR), and one-hot encoding, we transformed sequences into formats suitable for classification. To address class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was employed. Then, we implemented models such as Random Forest, XGBoost, LightGBM, Hierarchical Classifiers, and 1D and 2D Convolutional Neural Networks. Each of our models achieved over 97% accuracy and/or F1-Score. Additionally, we trained a Variational Autoencoder (VAE) to generate synthetic sequences with high similarity to known variants, which were further considered as potential new variants for classification. We then proposed and employed a voting-based detection algorithm to identify these potential new variants. Our results aligned with those of the Genome Detective Dengue Virus Typing Tool, affirming the model's rapid and accurate classification capabilities. Despite some limitations, such as only using 38 available lineages and the abstract nature of the algorithms, this work contributes to real-time genomic surveillance of dengue.

Keywords: Dengue virus, classification, machine learning, deep learning, genomic sequences, k-mer encoding, FCGR, one-hot encoding, hierarchical classification, Variational Autoencoder, variant detection.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Emmanuel Nyandu Kagarabi, 24 October 2024

Résumé

Le virus de la Dengue (DENV), responsable de maladies telles que la dengue et la dengue sévère (ou dengue hémorragique), provoque actuellement plus de 100 millions d'infections par an dans le monde. Il est considéré comme l'une des infections virales à transmission vectorielle les plus répandues, constituant un problème majeur de santé publique mondial en raison du nombre élevé de décès qu'il engendre, ainsi que du lourd fardeau économique lié à sa gestion, notamment dans les régions tropicales et subtropicales. Dans cette étude, nous présentons une approche pour classer les séquences génomiques du virus de la dengue en lignées et faciliter la détection précoce de nouveaux variants potentiels. Pour ce faire, nous utilisons une combinaison de techniques d'extraction de caractéristiques et d'algorithmes d'apprentissage automatique (Machine Learning) ainsi que d'apprentissage profond (Deep Learning). En utilisant l'encodage k-mer, la représentation du jeu du chaos en fréquence (FCGR) et l'encodage one-hot, nous avons transformé les séquences en formats adaptés à la classification. Pour palier au problème de déséquilibre des classes, nous avons utilisé la technique de suréchantillonnage synthétique des minorités (SMOTE). Ensuite, nous avons implementé des modèles tels que Random Forest, XGBoost, LightGBM, des classificateurs hiérarchiques et des réseaux de neurones convolutifs 1D et 2D. Chacun de nos modèles a atteint une précision supérieure à 97% dans la classification de sequences. De plus, nous avons entraîné un autoencodeur variationnel (VAE) pour générer des séquences synthétiques très similaires aux variants connus, qui ont ensuite été considérées comme des nouveaux variants potentiels à classifier. Nous avons ensuite proposé et utilisé un algorithme de détection basé sur un système de vote pour identifier ces nouveaux variants potentiels. Les résultats de notre algorithme ont été exactement les memes que ceux du logiciel Genome Detective Dengue Virus Typing Tool officiellemt utilisé pour tester les échantillons de la Dengue, confirmant les capacités rapides et précises de classification de notre modèle. Malgré certaines limitations, notamment le fait de n'avoir utilisé que 38 lignées disponibles et la nature abstraite des algorithmes utilisés, ce travail contribue à la surveillance genomique en temps reel de la dengue.

Mots-clés : Virus de la dengue, classification, apprentissage automatique, apprentissage profond, séquences génomiques, encodage k-mer, FCGR, encodage one-hot, classification hiérarchique, autoencodeur variationnel, détection de variants.

Déclaration

Je, soussigné, déclare par la présente que le travail contenu dans ce projet de recherche est mon travail original, et que tout travail réalisé par d'autres ou par moi-même antérieurement a été dûment reconnu et référencé en conséquence.

Emmanuel Nyandu Kagarabi, 24 octobre 2024

Contents

1. Introduction

1.1 Context

Approximately 1.7 million viruses exist on Earth, responsible for a variety of diseases such as COVID-19, HIV, Ebola, SARS, MERS, Dengue, Rabies, Measles, Chikungunya, MPOx, Oropouche, and Zika (Shiraj and Yousuf, 2024). Dengue is currently one of the most widespread vector-borne viral infections, particularly affecting tropical and subtropical regions (Hill et al., 2024). It is transmitted primarily by *Aedes* mosquitoes, mainly *Aedes aegypti* and *Aedes albopictus*, with environmental factors such as temperature and humidity influencing its spread (Roy et al., 2024). Dengue follows a human-to-mosquito-to-human transmission cycle and can present symptoms ranging from mild fever to life-threatening conditions such as dengue hemorrhagic fever (Guzman and Harris, 2015).

The World Health Organization (WHO) estimates there are between 100 to 400 million cases of Dengue annually, with an unprecedent rise in 2023 (World Health Organization, 2024a). Furthermore, around 3.8 billion people reside in regions where Dengue is endemic, particularly in Africa, the Americas, and Asia. The virus is present in over 100 countries worldwide and continues to spread to new areas (Chauhan et al., 2024). Dengue virus has four serotypes (DENV-1, DENV-2, DENV-3, and DENV-4), each with distinct genetic and antigenic properties. These serotypes also have multiple genotypes and several lineages (Hill et al., 2024). While infection with one serotype provides immunity to that specific serotype, secondary infections with different serotypes increase the risk of severe disease (World Health Organization, 2024b).

To control the spread of Dengue, various strategies have been investigated. Traditional methods such as vector control and insecticides are widely employed (Singh and Taylor-Robinson, 2017), and some vaccines have been developed (World Health Organization, 2023-11-24; Freedman, 2023), though none fully prevent infection across all four serotypes. Thus, there is an urgent need for more advanced approaches, particularly in genomic surveillance, to understand the virus's evolution and spread.

Indeed, genomic surveillance, critical for monitoring viral outbreaks, has been key in tracking and understanding the progression of the COVID-19 pandemic through tools like the PANGO system (Rambaut et al., 2020), a lineage nomenclature system designed to classify the diversity of the SARS-CoV-2 virus (which causes COVID-19) into lineages. Inspired by this, recent advances in Dengue genomic surveillance have introduced hierarchical lineage classifications, enhancing the ability to track the virus's evolution (Hill et al., 2024). Previously, the Dengue virus could only be classified into the main four serotypes and broad genotype sub-classifications, making it difficult to track the global spread of the virus.

As the volume of genomic data continues to grow exponentially (Benson et al., 2014), traditional methods of viral classification (such as alignment-based techniques) struggle to keep pace. Machine learning and deep learning techniques have emerged as powerful tools for processing large-scale genomic data and enabling real-time classification and surveillance of viral lineages (Qayyum et al., 2024). Integrating these methods in the genomic surveillance of Dengue could significantly enhance the ability to track and classify the virus's lineages.

1.2 Main Contribution

Current methods for classifying Dengue lineages rely on building phylogenetic trees by aligning genomic sequences with a reference genome, and defining lineages based on clades in the tree. While these

methods are robust, they are computationally expensive. Machine learning offers a promising alternative, particularly with alignment-free algorithms. To date, no published studies have applied machine learning or deep learning to classify Dengue sequences at the lineage level. This study leverages state-of-the-art genomic feature extraction techniques and machine learning algorithms to classify Dengue sequences into lineages. Additionally, we propose an early detection mechanism for potential new variants using pre-trained models and a voting system to enhance real-time surveillance.

1.3 Aim and Objectives

1.3.1 Objective

The main objective of this project is to develop machine learning and deep learning models for classifying Dengue genomic sequences into lineages, ultimately contributing to the advancement of Dengue genomic surveillance and variant detection.

1.3.2 Specific Aims

The specific aims of this study are to :

- Identify and apply some genomic feature extraction techniques to Dengue sequences.
- Design and implement machine learning and deep learning models to efficiently classify Dengue sequences into lineages.
- Generate synthetic Dengue virus sequences to mimic real-world data.
- Develop a framework for the early detection of new Dengue variants using pretrained models and a voting mechanism.

1.4 Thesis Outline

This thesis is organized into five chapters including the Introduction:

- Chapter 2 provides some related works on Dengue virus classification, and an overview of genomics, feature extraction techniques, and metrics used to evaluate machine/deep learning algorithms.
- Chapter 3 details the methodology, covering data collection, preprocessing, and the models employed in this study.
- Chapter 4 presents the results and analysis from the classification models, including the performance of the early detection mechanism.
- Chapter 5 summarizes the key findings, discusses the contributions of the study, addresses limitations, and proposes directions for future research in Dengue virus classification and surveillance.

2. Background

2.1 Literature Review

This section provides an overview of how machine learning (ML) and deep learning (DL) techniques have been applied to the classification of genomic sequences, with a particular focus on the Dengue virus. We also discuss early detection methods for emerging variants and how our work builds upon these advancements.

2.1.1 Machine Learning and Deep Learning for Viral Classification

Recent advances in ML and DL have led to innovative approaches for classifying viral sequences. One such example is COVID-DeepPredictor, introduced by Saha et al. (2021), a deep learning model designed to predict and classify six pathogenic viruses, including SARS-CoV-2, SARS-CoV-1, MERS-CoV, Ebola, Dengue, and Influenza, using genomic sequences. The features were extracted using the k-mer technique, with $k = 3$ being optimal for balancing runtime and accuracy. These features were then classified using a Long Short-Term Memory (LSTM) architecture, achieving exceptional accuracy between 99.51% and 99.94%, outperforming traditional classifiers such as Random Forest and Linear Discriminant Analysis.

Building on this, Gunasekaran et al. (2021) proposed a DL-based approach to classify DNA sequences of the same six viruses, using Convolutional Neural Networks (CNN) and hybrid models like CNN-LSTM and CNN-Bidirectional LSTM. By employing k-mer encoding (with $k = 6$ identified as optimal), their models achieved high accuracy, with CNN and CNN-Bidirectional LSTM attaining accuracies of 93.16% and 93.13%, respectively.

Similarly, Shiraj and Yousuf (2024) introduced virusBERT, a Transformer-based model for viral sequence classification. They used the same six pathogens previously mentioned and employed k-mer encoding (with $k = 3, 4, 5$) for feature extraction. The model achieved a strong performance in classifying viral sequences across all classes, with a training accuracy of 88.73% and validation accuracy of 82.53%. However, like the previously mentioned studies, they did not address lineage-specific distinctions, which are essential for monitoring viral evolution and detecting new variants.

In another study, Mumtaz et al. (2024) introduced DL-DVE, a model for classifying DENV serotypes using a relatively small dataset (2000 sequences, evenly distributed across the four serotypes). They applied tokenization and n-grams as feature extraction techniques and trained an LSTM-based generative model to generate new sequences. Their model, when combined with a FeedForward Neural Network (FNN) for classification, achieved an ROC-AUC score of 0.818, with accuracy, precision, recall, and F1-Scores all around 99%. However, the assumption of a balanced dataset across Dengue serotypes, and the lack of lineage-level classification limit its effectiveness in real-time genomic surveillance of Dengue.

Building on these earlier models, Qayyum et al. (2024) studied viral DNA classification across multiple pathogens, including COVID-19, MERS, Dengue, Hepatitis, and Influenza. They tested various models, including 1D CNNs, LSTMs, and a multi-transformer model, and used both k-mer and one-hot encoding for feature extraction. Among these methods, the multi-transformer model excelled, achieving over 99% accuracy and outperforming the other models by effectively capturing complex sequence patterns. This highlights the importance of transformer-based methods in virus classification, which could be helpful in our case of Dengue virus.

2.1.2 Lineage Classification and Early Variant Detection

For viral lineage classification and variant detection, several studies on SARS-CoV-2 offer insights into techniques that could be applied to Dengue. For instance, Ullah et al. (2022) developed a model that applies self-attention mechanisms to classify SARS-CoV-2 variants. They trained a 1D convolutional neural network (1D-CNN) with self-attention layers on one-hot-encoded genome sequence data. In addition to variant classification, the authors introduced methods for predicting and generating new variants. They used entropy as an uncertainty measure, with a predefined threshold, to identify novel variants; samples with entropy values exceeding this threshold were classified as new variants. For generating new variants, they employed a variational autoencoder-decoder (VAE) trained on known variants to create potential new sequences. These generated sequences were then classified as new variants using the same classifier. Similarly, Togrul and Arslan (2022) proposed a deep learning-based method for detecting SARS-CoV-2 variants from genomic sequences. Using convolutional neural networks (CNN) to extract discriminative features from SARS-CoV-2 genomes, they applied machine learning classifiers, including k-nearest neighbor (KNN), multilayer perceptron (MLP), support vector machine (SVM), and random forest (RF), to detect key variants such as Alpha, Gamma, Delta, Beta, and Omicron. The method achieved remarkable accuracy, with a reported 99.9% accuracy for detecting the main variants, highlighting the efficacy of deep learning in real-time viral surveillance.

These studies highlight the effectiveness of DL for variant detection, yet no parallel efforts have been made for Dengue lineage classification.

2.1.3 Limitations of Current Approaches

Current methods for classifying Dengue lineages rely on phylogenetic tree construction by aligning genomes to each other or to a reference genome using algorithms such as maximum likelihood, Bayesian inference, and neighbor-joining (Waman et al., 2016). These techniques, while robust, are computationally intensive. Machine learning, particularly alignment-free algorithms, offers an alternative. However, the body of research on classifying Dengue genomic sequences using ML/DL remains sparse. Only (Mumtaz et al., 2024) considered serotype-level classification, and no published studies have classified Dengue sequences at the lineage level using the newly introduced hierarchical nomenclature by Hill et al. (2024), designed to enhance the tracking of viral evolution.

2.1.4 Our Contribution: Classification and Early Detection of Dengue Lineages

Building on these methodologies, our study focuses on classifying Dengue genomic sequences at the lineage level using two broad categories of algorithms. On one hand, we use machine learning-based approaches, including flat classifiers (Random Forest, XGBoost, and LightGBM) as well as hierarchical classifiers, specifically Local Classifier Per (Parent Node, Level, and Node). On the other hand, we employ deep learning-based methods, including a 1D CNN with self-attention and 2D CNNs. To address the early detection of emerging variants, we propose an algorithm that integrates pre-trained models with a voting mechanism. This ensemble-based system evaluates outputs from multiple models to improve the robustness of variant detection, enabling timely public health responses.

2.2 Foundations of Genomic Science

Genomics is the study of an organism's complete set of genetic material, encompassing gene interactions and environmental influences (National Human Genome Research Institute, 2024). The fundamental unit of genomic information is DNA, a double-helix molecule made up of four nucleotide bases: Adenine

(**A**), Cytosine (**C**), Guanine (**G**), and Thymine (**T**), where A pairs with T and C pairs with G, as illustrated in the following Figure 2.1.

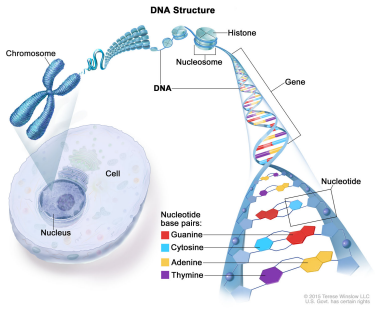


Figure 2.1: DNA Structure (LLC, 2018).

It is worth noting that if the sequence of one strand of a DNA molecule is known, the sequence of the other strand can be easily reconstructed (Isaev et al., 2004). Therefore, DNA sequences are always presented as sequences of single nucleotides, rather than paired ones.

Each organism's genome comprises its entire DNA sequence. For instance, the human genome consists of approximately 3 billion base pairs, with around $20,000 - 25,000$ genes that code for proteins (National Human Genome Research Institute, 2024). Alterations in DNA can lead to protein malfunctions, often causing diseases. Similarly, RNA viruses, such as the dengue virus, disrupt cellular functions by controlling the host's machinery to replicate their RNA and synthesize viral proteins (Natali et al., 2021).

2.3 Dengue Virus Overview

The dengue virus (DENV) is a small, spherical, single-stranded RNA virus comprising approximately 10,700 nucleotides (Kularatne and Dalugama, 2022). A member of the Flaviviridae family, which includes other notable viruses such as the Zika virus, West Nile virus, and tick-borne encephalitis virus (Martins et al., 2022), DENV is primarily transmitted to humans through the bite of infected Aedes mosquitoes, particularly Aedes aegypti and Aedes albopictus. Dengue follows a human-to-mosquito-to-human transmission cycle as illustrated in the following Figure 2.2:

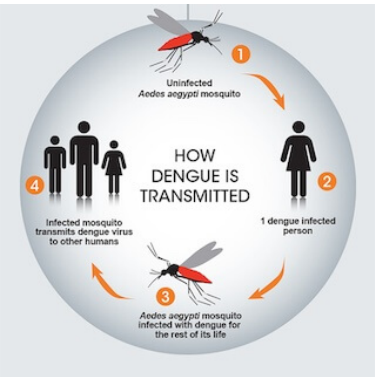


Figure 2.2: Dengue cycle (American Society for Microbiology, 2024): The dengue virus is transmitted from human to mosquitoes and vise-versa.

Dengue is one of the most rapidly spreading mosquito-borne viral diseases, with increasing incidence worldwide and significant health risks (Sabir et al., 2021).

The first recorded dengue outbreak occurred in the 18th century, but the virus likely existed long before. In recent decades, the frequency and severity of dengue outbreaks have escalated. The World Health Organization (WHO) estimates that dengue infects 100 to 400 million people annually, with approximately 500,000 severe cases requiring hospitalization (Gupta et al., 2023).

After an infected mosquito transmits the virus to a human host, symptoms typically appear within four to ten days. These symptoms range from high fever, headache, and muscle pain to more severe manifestations like bleeding and plasma leakage, particularly in severe dengue cases, which can be life-threatening (Mayo Clinic, 2024). While many people recover without complications, secondary infections with different dengue serotypes increase the risk of severe illness (Natali et al., 2021).

The dengue virus exists in four distinct serotypes (DENV-1, DENV-2, DENV-3, and DENV-4). Within each serotype, there are several genotypes that reflect the virus's genetic diversity. For instance, DENV-1 includes five genotypes, DENV-2 has six, DENV-3 has five, and DENV-4 has four (Hill et al., 2024). These genotypes help track the virus's evolution and geographical distribution, as certain genotypes are more prevalent in specific regions. This complexity poses challenges for both vaccine development and clinical management (World Health Organization, 2024b).

Currently, at a finer level, genotypes are subdivided into lineages. A lineage represents a group of viral strains with a common evolutionary origin, often identified through phylogenetic analysis of viral sequences (Centers for Disease Control and Prevention, 2023). The recently introduced dengue nomenclature system by Hill et al. (2024) adds two additional levels to the classification hierarchy: Major and Minor lineages, which are now cataloged on the Dengue Lineages website and integrated into the Genome Detective Dengue Virus Typing Tool. The introduction of these two additional sublevels aims to improve the genomic surveillance of Dengue. For example, under this newly introduced nomenclature system, the lineage "**3III_B.3.2**" can be broken down into four hierarchical levels: Serotype (**3**), Genotype (**III**), Major Lineage (**B**), and Minor Lineage (**B.3.2**). These lineage labels will be utilized in our classification algorithms in Chapter 3. Tracking viral lineages allows researchers to monitor the spread of specific variants and their potential impact on public health.

Mutations occurring during viral replication may have various effects on viral properties, such as increased transmissibility or immune evasion (Isaev et al., 2004). When a virus accumulates enough mutations to distinguish itself from previous strains, it may be classified as a new variant, which could pose additional challenges to public health efforts (Johns Hopkins Medicine, 2024).

2.4 Feature Extraction Techniques in Genomics

Just as the order of letters determines the meaning of a word in language, the information encoded in a DNA sequence is determined by the specific arrangement of letters from the DNA alphabet:

$$\Omega = \{\mathrm{A},\ \mathrm{C},\ \mathrm{G},\ \mathrm{T}\}. \tag{2.4.1}$$

In this context, a DNA sequence can be thought of as a sentence in an "alphabet" unique to genomics. However, to apply many machine learning and deep learning algorithms, DNA sequences must be transformed from raw text to numerical representations. Feature extraction, therefore, plays a crucial role in genomics by converting these raw genomic sequences into structured, quantitative formats that algorithms can process effectively. In this section, we discuss the efficiency of some widely used feature extraction methods.

2.4.1 Numerical Encoding

Genomic sequences can be numerically encoded in various ways, each mapping nucleotide bases to different numerical values. The following Table 2.1 provides a summary of the different encoding methods : ordinal, integer, real, and complex encoding.

Table 2.1: Representation of a genomic sequence of length n as a vector $S = (f(x_1), \ldots, f(x_n)) \in \mathbb{R}^n$.

Encoding Type	Ordinal	Integer	Real	Complex
Mapping : $f(x_i) =$	$\begin{cases} 0.25, x_i = \text{A} \\ 0.5, x_i = \text{T} \\ 0.75, x_i = \text{G} \\ 1.0, x_i = \text{C}. \end{cases}$	$\begin{cases} 0, x_i = \text{T} \\ 1, x_i = \text{C} \\ 2, x_i = \text{A} \\ 3, x_i = \text{G}. \end{cases}$	$\begin{cases} -0.5, x_i = \text{G} \\ -1.5, x_i = \text{A} \\ 0.5, x_i = \text{C} \\ 1.5, x_i = \text{T}. \end{cases}$	$\begin{cases} -1-j, x_i = \text{G} \\ 1+j, x_i = \text{A} \\ -1+j, x_i = \text{C} \\ 1-j, x_i = \text{T} \end{cases}$
References :	(Singh, 2022)	(Shaukat, 2023)	(Kwan, 2009)	(Bonidia et al., 2021)

2.4.2 One-Hot Encoding

In one-hot encoding, each nucleotide is represented by a vector of length 4. The four components of this vector correspond to the four nucleotide bases, with 1 indicating the presence of a particular base and 0 indicating its absence. A sequence S of n nucleotides is represented as a matrix of size $n \times 4$ using the following Equation 2.4.2:

$$S = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}, \text{ where } f(x_i) = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \text{ if } x_i = \text{A}, \\ \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \text{ if } x_i = \text{C}, \\ \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \text{ if } x_i = \text{G}, \\ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}, \text{ if } x_i = \text{T}. \end{cases} \tag{2.4.2}$$

This technique is widely used in deep learning models such as Convolutional Neural Networks (CNNs), where matrix inputs are required. The method has demonstrated effectiveness in various studies (Qayyum et al., 2024; Ullah et al., 2022).

2.4.3 k-mer Encoding

In k-mer encoding, a genomic sequence is broken into smaller, overlapping substrings (or k-mers) of length k. Each k-mer can be encoded using one of the previously presented techniques. The number of possible k-mers in a DNA sequence is 4^k, given the four possible nucleotide bases.

The total number of k-mers in a sequence of length n is given by:

$$\text{Total number of k-mers} = n - k + 1. \tag{2.4.3}$$

For example, the sequence 'ACGTT' can be split into 2-mers as (AC, CG, GT, TT), yielding $5-2+1 = 4$ k-mers. The value of k influences the trade-off between simplicity and information density. Smaller k

values yield simpler but less detailed features, while larger k values offer more complex features (Shaukat, 2023). Studies like Avila Santos et al. (2024) and Shiraj and Yousuf (2024) have explored the impact of varying k values on ML/DL model performance.

2.4.4 Frequency Chaos Game Representation (FCGR)

The Frequency Chaos Game Representation (FCGR) is a method of visualizing and analyzing DNA sequences by mapping k-mer frequencies into a two-dimensional fractal space (Paul et al., 2022). Each k-mer occupies a distinct position, and its frequency is represented by the intensity of the corresponding pixel. This method has been shown to perform well in sequence classification tasks (Thind and Sinha, 2023; Agbodoyetin, 2024).

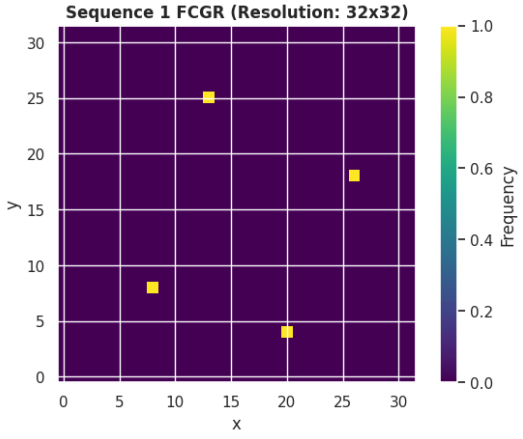
FCGR mapping is generated using the recursive positioning described in Equation (2.4.4), starting from the center of a unit square and moving toward the corner assigned to each nucleotide.

$$\begin{cases} X_0 = \left(\frac{1}{2}, \frac{1}{2}\right) \\ X_i = \frac{1}{2}\left[X_{i-1} + \mathcal{C}(S_i)\right], \forall i \in \{1, 2, \ldots, n\}. \end{cases} \tag{2.4.4}$$

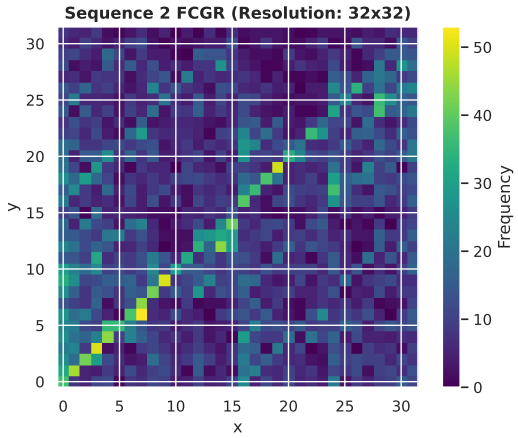
Here, $\mathcal{C}(S_i)$ represents the coordinates of the nucleotide $S_i \in \Omega$:

$$\mathcal{C}(S_i) = \begin{cases} (0,0), \text{ if } S_i = \text{ A}, \\ (0,1), \text{ if } S_i = \text{ C}, \\ (1,1), \text{ if } S_i = \text{ G}, \\ (1,0), \text{ if } S_i = \text{ T}. \end{cases} \tag{2.4.5}$$

In the following Figure 2.3, we illustrate the FCGR technique by picking one sequence of lineage **1I_K.2** in our dataset (see Chapter 3) and plotting it in Python:



(a) Sequence1 : "ATGC"



(b) Sequence 2: Dengue lineage sample

Figure 2.3: Illustration of FCGR using sequences from our Dengue dataset.

2.4.5 Digital Signal Processing (DSP)

In Digital Signal Processing (DSP), genomic sequences are converted into numerical signals, enabling the use of various signal processing methods (Randhawa et al., 2019). Specifically, each DNA sequence

$$S_i = (S_i(0), S_i(1), \ldots, S_i(n-1)), \text{ where } S_i(k) \in \Omega,$$

is first mapped to a numerical sequence

$$N_i = (f(S_i(0)), f(S_i(1)), \ldots, f(S_i(n-1)))$$

using a nucleotide-to-number encoding function f, as previously outlined in subsection 2.4.1. For instance, in integer encoding, $f(\mathsf{C}) = 1$, $f(\mathsf{G}) = 3$, $f(\mathsf{A}) = 2$, and $f(\mathsf{T}) = 0$. Therefore, the sequence $S = \mathsf{CGAT}$ is represented numerically as $N = (1, 3, 2, 0)$.

Next, the Discrete Fourier Transform (DFT) is applied to the numerical sequence, transforming it into its frequency-domain representation (Yin et al., 2020):

$$F_i(k) = \sum_{j=0}^{n-1} f(S_i(j)) e^{-\frac{2\pi i}{n} kj}, \quad \text{for } k = 0, 1, \ldots, n-1$$

The magnitude spectrum, $M_i(k) = |F_i(k)|$, is calculated by taking the absolute values of the elements in the DFT result. To quantify the similarity between two sequences, $X = M_i$ and $Y = M_j$, the Pearson Correlation Coefficient (PCC) is used, defined as (Randhawa, 2020):

$$r_{XY} = \frac{\sum_{k=0}^{n-1}(X_k - \overline{X})(Y_k - \overline{Y})}{\sqrt{\sum_{k=0}^{n-1}(X_k - \overline{X})^2} \times \sqrt{\sum_{k=0}^{n-1}(Y_k - \overline{Y})^2}}$$

where $\overline{X}$ and $\overline{Y}$ represent the mean values of X and Y, respectively. This approach generates a feature vector for each sequence based on its similarity to others, facilitating effective classification.

The list above is not exhaustive; additional formidable approaches, such as the *Z-curve* (Adetiba et al., 2022), have been employed, offering a distinctive 3D representation of any sequence as a 3D RGB image. Other methods include the Spaced-word Frequencies (SWF) (Leimeister et al., 2014), the Return Time Distribution (RTD) (Kolekar et al., 2012), and MASH (Ondov et al., 2016), among others.

2.5 Evaluation and Performance Metrics

A classifier is only as good as the metric used to evaluate it (Brownlee, 2020). In this section, we present an overview of the common metrics widely used to evaluate Machine Learning and Deep Learning algorithms. For simplicity, we consider the binary classification case, while the multi-classification case is a straightforward generalization using the one-versus-rest approach.

A flat classifier is a learning algorithm that, given an input instance, assigns a single label to it. In contrast, a hierarchical classifier associates a taxonomical list of labels with an input instance, assuming there is a hierarchy in the dataset outputs(labels). Below, we present some widely used metrics appropriate for each of these two types of classifiers:

2.5.1 Flat Classifier Evaluation Metrics

1. **Confusion Matrix**

   The confusion matrix provides insight not only into the performance of a predictive model but also into which classes are being predicted correctly, which incorrectly, and what types of errors are being made.

   Table 2.2: Binary Confusion Matrix

	Positive Prediction	**Negative Prediction**
Positive Class	True Positive (**TP**)	False Negative (**FN**)
Negative Class	False Positive (**FP**)	True Negative (**TN**)

   With the confusion matrix, we can easily compute the following metrics:

2. **Accuracy** quantifies the frequency with which the model accurately predicts the outcome. The calculation is performed using the following equation (2.5.1):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2.5.1}$$

3. **Recall** quantifies the number of correct positive predictions made out of all possible positive predictions. The calculation is performed using the following equation (2.5.2):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.5.2}$$

4. **Precision** quantifies the number of correct positive predictions made out of all positive predictions. The calculation is performed using the following equation (2.5.3):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.5.3}$$

5. **F-beta Measure or F-beta Score**

   The F-beta measure (Brownlee, 2020) combines both Precision and Recall in a single calculation, enabling their harmonic mean to be adjusted by a coefficient called beta (β), as given in Equation (2.5.4).

$$\text{F}\beta = \frac{(1 + \beta^2) \times (\text{Precision} \times \text{Recall})}{(\beta^2 \times \text{Precision}) + \text{Recall}}. \tag{2.5.4}$$

   The choice of the β parameter will determine the name of the F-beta measure. For example, a β value of 2 is referred to as the F2-measure or F2-score, while a β value of 1 is referred to as the F1-measure or F1-score. Three common values for the beta parameter are as follows:

   - $F0.5$-measure ($\beta = 0.5$): More weight on precision, less weight on recall.
   - F_1-measure ($\beta = 1$): Balances the weight on precision and recall.
   - F_2-measure ($\beta = 2$): Less weight on precision, more weight on recall.

6. **Matthews Correlation Coefficient**

   Considered as a performance indicator with great credibility, the Matthews correlation coefficient (MCC) provides a fair assessment even in the presence of class imbalances (Itaya et al., 2024). This metric is computed using the following Equation (2.5.5):

$$\text{MCC} = \frac{(\text{TP} \times \text{TN}) - (\text{FP} \times \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}. \tag{2.5.5}$$

   This coefficient ranges from -1 to 1, with a value of 1 indicating perfect classification, 0 indicating random categorization, and -1 indicating total discordance between expected and actual labels.

2.5.2 Hierarchical Classifier Evaluation Metrics

As noted by Silla and Freitas (2011), flat classification metrics may not adequately capture the performance of algorithms when dealing with hierarchical data. To address this, Kiritchenko et al. (2006) introduced Hierarchical Precision (hP), Hierarchical Recall (hR), and Hierarchical F-score (hF), which are analogous to their flat counterparts, Precision, Recall, and F1-score. These hierarchical metrics are defined as follows:

$$\text{hP} = \frac{\sum_i |\alpha_i \cap \beta_i|}{\sum_i |\alpha_i|}, \quad \text{hR} = \frac{\sum_i |\alpha_i \cap \beta_i|}{\sum_i |\beta_i|}, \quad \text{and} \quad \text{hF} = 2 \times \frac{\text{hP} \times \text{hR}}{\text{hP} + \text{hR}}, \tag{2.5.6}$$

where α_i represents the set of predicted classes for test sample i, including the most specific class and all its ancestor classes, while β_i represents the set of true classes and their ancestors for the same test sample. The summation is performed over all test instances.

In Chapter 3, the methodology for employing these metrics to evaluate classifiers will be discussed in detail.

3. Methodology

This chapter outlines the methodology used to classify Dengue genomic sequences into lineages and introduces a voting-based algorithm designed for the early detection of potential new variants.

3.1 Data Collection and Preprocessing

3.1.1 Data Collection

Our study began with the acquisition of Dengue genomic sequences from the publicly accessible Global Initiative on Sharing Avian Influenza Data (GISAID) database. We selected only complete genomic sequences with high coverage and accurate collection dates, resulting in a total of 7,689 samples as of 29/08/2024. Additionally, metadata such as the collection date, submission date, and geographical location were obtained for each sequence.

To determine the lineage for each sequence, we used the Genome Detective Dengue Virus Typing Tool. The sequences were categorized by serotype and cross-referenced with data from the Dengue Lineages Initiative, which includes 64 globally recognized minor lineages, as detailed by Hill et al. (2024). This data served as ground truth for our experiments.

Some sequences were labeled as "unassigned," "related," or "similar to..." by the Genome Detective Dengue Virus Typing Tool. To maintain consistency with officially recognized lineages, we excluded all such sequences. After this filtering, the final cleaned dataset comprised 3,527 sequences.

The final dataset includes features such as the genomic sequence, serotype, lineage (considering serotype, genotype, major lineage, and minor lineage), collection date, submission date, and geographical location. So, Figure 3.1 illustrates the distribution of sequences by serotype and lineage.

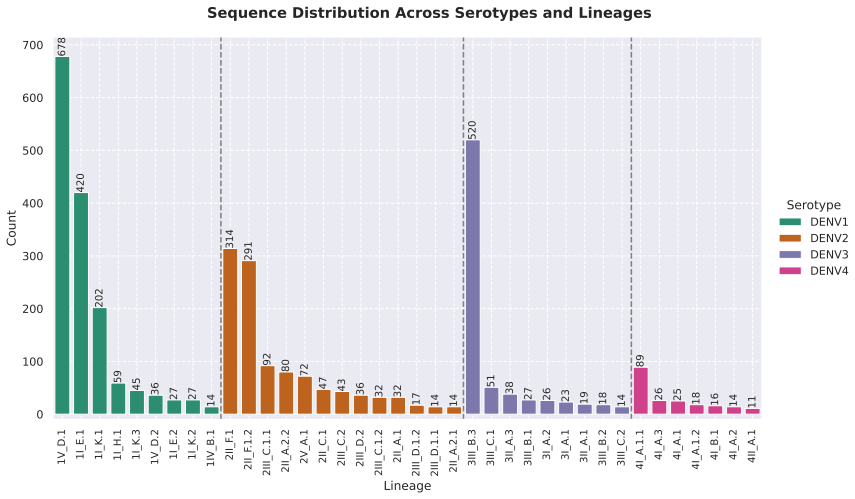


Figure 3.1: Sequence Distribution: The sequences are colored by serotype, with a total of $1,508$ samples for DENV1 (42.8% of the dataset), $1,084$ for DENV2 (30.7% of the dataset), 736 for DENV3 (20.9% of the dataset), and 199 for DENV4 (5.6% of the dataset). Additionally, we have counted the number of sequences per lineage within each serotype. The dataset is highly imbalanced in both cases.

3.1.2 Preprocessing

During the sequencing or genome assembly process, ambiguous bases are often introduced at positions where the correct nucleotide is uncertain or where multiple alternatives exist. The most common of these is "N," which indicates that any of the four nucleotides (A, C, G, T) could occupy that position, usually due to insufficient coverage during sequencing. We observed ambiguous characters such as B, R, M, D, V, Y, N, S, K, and W in the dataset and chose to remove them to ensure data consistency.

The next crucial step was feature extraction, where the categorical genomic sequences are transformed into numerical formats suitable for both Machine and Deep Learning algorithms. We employed three feature extraction techniques from those detailed in Section 2.4. First, methods like k-mer encoding and Frequency Chaos Game Representation (FCGR) were used, as they generate a fixed output shape regardless of sequence length. Second, one-hot encoding was applied, with post-padding to standardize all sequences to a uniform maximum length of 10,821 nucleotides (Figure A.1a). This uniform length is essential for all algorithms that require inputs of a fixed size, such as Convolutional Neural Networks (CNNs).

Before selecting any classification algorithm, we visualized the distribution of the sequences in a low-dimensional space using t-distributed Stochastic Neighbor Embedding (t-SNE). This technique allows for the representation of high-dimensional data in a reduced two-dimensional space ($\mathbb{R}^2$), making it easier to observe patterns. This is depicted in the following Figure 3.2.

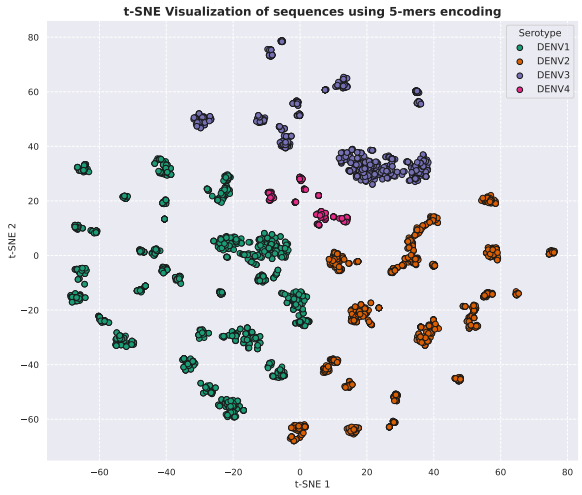


Figure 3.2: t-SNE scatter plot of k-mer-encoded sequences (k=5). Each point represents a genomic sequence, with colors distinguishing the four Dengue serotypes. The distinct clustering by serotype indicates that k-mer encoding effectively captures differences in genomic sequences. The dispersed clusters for DENV1 and DENV2 suggest greater genetic diversity within these serotypes, while the tighter clusters for DENV3 and DENV4 imply lower variability. The t-SNE visualization is sensitive to changes in the k-mer value. At this optimal k-mer value, the lineage clusters are distinct, but this is not the case for all k-mer values.

The insights gained from the t-SNE analysis in Figure 3.2 influenced the design and implementation of classification algorithms. In the next Section 3.2, we present two kinds of classifiers: Machine-Learning based(Flat, and Hierarchical), and Deep-Learning-based.

3.2 Classification Algorithms

As observed in Figure 3.1, the dataset presents a significant class imbalance problem: the majority class (lineage 1V_D.1) contains 678 sequences, while the minority class (lineage 4II_A.1) has only 11 sequences. This imbalance poses a challenge for most Machine Learning algorithms, which tend to favour the majority class at the expense of the minority classes (Brownlee, 2020), although all lineages are of equal importance in our study.

To address this issue for any learning algorithm, we first split the dataset into training and test sets, allocating 80% of the data for training and 20% for testing. We applied stratification to ensure that each lineage was represented proportionally in both sets.

Next, to counteract the imbalanced distribution, we employed the Synthetic Minority Oversampling Technique (SMOTE) (Brownlee, 2020; Gunasekaran et al., 2021). SMOTE works by generating synthetic samples for the minority class. Specifically, it selects a minority class instance and identifies its k nearest neighbours within the same class. A synthetic instance is then generated by interpolating between the selected instance and one of its neighbours. This process creates synthetic samples for the minority classes and can also apply undersampling to the majority class until a balanced distribution is achieved.

By applying SMOTE to our training dataset considering any feature extraction, we obtained a balanced distribution, where all classes contained 542 samples. This resulted in a training dataset with $20,596$ samples and a test dataset with 706 samples, for a total of $21,302$ samples across the whole dataset. For each ML classifier, we used the Optuna library (Akiba et al., 2019) to tune hyperparameters and optimise performance.

3.2.1 Machine Learning-Based Classifiers

We implemented six different classifiers, which included three flat classifiers and three hierarchical classifiers. The flat classifiers used were Random Forest, XGBoost, and LightGBM. These were selected for their proven efficiency in classification tasks, with LightGBM and XGBoost consistently performing well in a wide range of competitions and real-world applications (Ke et al., 2017).

The hierarchical classifiers included Local Classifier Per Parent Node (LCPN), Local Classifier Per Level (LCL), and Local Classifier Per Node (LCN), chosen to reflect the natural hierarchical structure inherent in the lineage definitions.

We employed 10-fold cross-validation, where the training data was randomly divided into 10 equal subsets. The classifier was trained on 9 subsets and validated on the remaining subset, and this process was repeated until all subsets had been used for validation.

The flat classifiers were evaluated using five performance metrics: Accuracy, Precision, Recall, F1-Score, and Matthew's Correlation Coefficient (MCC), as outlined in Section 2.5. The hierarchical classifiers were evaluated using specialized hierarchical measures, including Hierarchical Precision, Hierarchical Recall, and Hierarchical F1-Score, as defined in Equation (2.5.6). A brief summary of each classifier is provided in the following subsections.

(A) Flat Classifiers

(A.1) Random Forest

Random Forest (RF) is an ensemble learning algorithm widely used for classification and regression tasks (Rigatti, 2017). It operates by constructing a multitude of decision trees during training, and the final

prediction is determined by the mode (majority vote) of the individual tree outputs. Despite its ability to improve generalisation, Random Forest can still overfit, especially with improper hyperparameter tuning (Murphy, 2022). Therefore, careful selection of parameters is essential. Table 3.1 shows the hyperparameters and their search space used with `Optuna` to optimise the Random Forest model.

Table 3.1: Hyperparameters for the Random Forest.

Hyperparameter	**Description**	**Search Space**
`bootstrap`	Whether samples are drawn with replacement when building trees	{`True, False`}
`criterion`	Function used to evaluate the quality of a split	{`entropy, gini`}
`max_depth`	Maximum depth of each tree	1 - 20
`min_samples_leaf`	Minimum number of samples required in a leaf node	1 - 20
`min_samples_split`	Minimum number of samples required to split a node	2 - 20
`n_estimators`	Number of trees in the forest	50 - 1500

(A.2) XGBoost

XGBoost (Chen and Guestrin, 2016), short for eXtreme Gradient Boosting, is a highly popular algorithm for classification, regression, and ranking tasks. It uses boosting, a technique where trees are constructed sequentially, with each tree correcting the errors of the previous ones (Géron, 2019). In contrast to bagging, where multiple independent models are trained in parallel and their predictions averaged (as in Random Forests), boosting builds models iteratively to minimize errors. XGBoost's efficiency and effectiveness in handling large datasets have made it a top performer in machine learning competitions and applications (NVIDIA, 2024). Table 3.2 lists the hyperparameters and their search space used with `Optuna` for tuning XGBoost.

Table 3.2: XGBoost Hyperparameters.

Hyperparameter	**Description**	**Search Space**
`learning_rate`	Step size for weight updates	10^{-3} - 1 (log scale)
`max_depth`	Maximum depth of each tree	1 - 20
`colsample_bytree`	Fraction of features used per tree	0.5 - 1
`min_child_weight`	Minimum sum of instance weights in a child node	1 - 10
`n_estimators`	Number of trees in the model	50 - 1500
`subsample`	Fraction of samples used per tree	0.5 - 1.0

(A.3) LightGBM

LightGBM (Ke et al., 2017) (Light Gradient Boosting Machine) is a fast, scalable gradient boosting framework that excels in handling large-scale datasets. Like XGBoost, it is a frequent choice in compe-

titions due to its efficiency and high performance. LightGBM grows trees leaf-wise, choosing the leaf with the largest delta to split, making it more effective at minimizing loss than traditional level-wise growth. While this strategy offers speed advantages, it can lead to overfitting if not properly tuned. Table 3.3 presents the hyperparameters and search space with `Optuna`.

Table 3.3: LightGBM Hyperparameters.

Hyperparameter	**Description**	**Search Space**
`learning_rate`	Step size per iteration	10^{-3} - 1 (log scale)
`num_leaves`	Maximum number of leaves per tree	2 - 256
`feature_fraction`	Fraction of features used per tree	0.5 - 1.0
`bagging_fraction`	Fraction of data used for each iteration	0.5 - 1.0
`bagging_freq`	Frequency of subsampling to prevent overfitting	1 - 7
`max_depth`	Maximum depth of each tree	1 - 20
`min_child_samples`	Minimum samples in a leaf node	1 - 100
`lambda_l1`	L1 regularization term on weights (for sparsity control)	10^{-8}–10.0 (log scale)
`lambda_l2`	L2 regularization term on weights (to prevent overfitting)	10^{-8}–10.0 (log scale)
`n_estimators`	Number of boosting iterations (trees)	50 - 1500

(B) Hierarchical Classifiers

In our dataset, each label follows a hierarchical structure. For example, the lineage $\mathbf{1IV\_D.1}$ can be deconstructed into four levels: Serotype (**1**), Genotype (**IV**), Major Lineage (**D**), and Minor Lineage (**D.1**). When training a flat multi-class classifier, we assign a single label known as the leaf node (Minor lineage) to each sample (Figure 3.3(**a**)). However, this approach ignores valuable hierarchical information embedded in the data. Given the hierarchical nature of labels in our dataset, it is more appropriate to employ hierarchical classifiers. These models can predict hierarchical labels, enabling us to assess model performance across multiple levels of the hierarchy, providing deeper insights into its strengths and weaknesses.

The goal of our hierarchical classification is to predict four levels of the lineage: Serotype, Genotype, Major Lineage, and Minor Lineage, based on genomic sequence features.

We represent the dataset as $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ corresponds to the feature space and $\mathbf{y}$ represents the hierarchical labels (i.e., lineages). The hierarchical labels are structured as arrays, with the shape defined in Equation 3.2.1:

$$\text{Shape(y)} = N_{\text{samples}} \times M_{\text{levels}}, \tag{3.2.1}$$

where N_{samples} denotes the total number of samples, and M_{levels} refers to the number of hierarchical levels per lineage.

Hierarchical classification is increasingly relevant in machine learning, particularly for domains where classes are naturally organized in a hierarchy, as is the case with genomic sequences. This subsection outlines the core methods we explored, with further details provided in relevant literature (Kiritchenko et al., 2006; Miranda et al., 2023; Naik, 2017).

Indeed, there are three widely adopted local hierarchical classification algorithms: Local Classifier per Node (LCN), Local Classifier per Parent Node (LCPN), and Local Classifier per Level (LCL). These methods are visually represented in Figure 3.3, and their functionalities are then summarized.

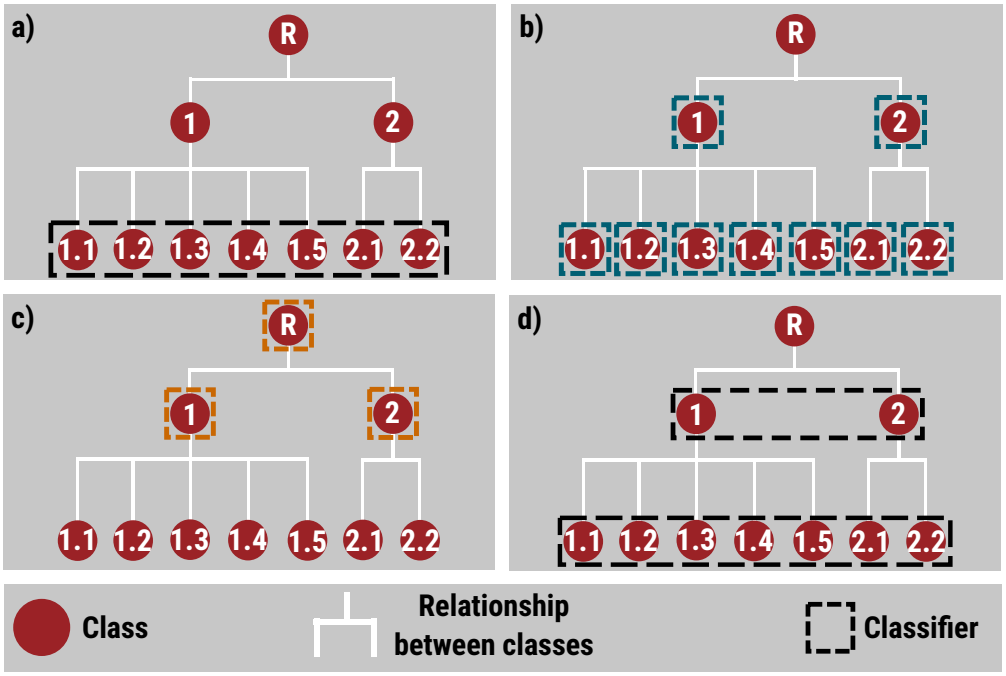


Figure 3.3: Common Local Hierarchical Classifiers (Miranda et al., 2023): (**a**) Flat multi-classifier – (**b**) Local Classifier per Node (LCN) – (**c**) Local Classifier per Parent Node (LCPN) – (**d**) Local Classifier per Level (LCL).

(B.1) Local Classifier per Node (LCN)

In the LCN approach, multiple binary classifiers are trained for each node in the hierarchy, except the root node. Each classifier focuses on distinguishing between sibling nodes, answering the question, "Is this instance a descendant of mine?" (Naik, 2017). This method, is illustrated in Figure 3.3(**b**), where nine classifiers should be used for the classification.

(B.2) Local Classifier per Parent Node (LCPN)

The LCPN approach involves training multi-class classifiers for each parent node. These classifiers determine which child node an instance belongs to by answering the question, "Which of my children does this instance belong to?" (Kiritchenko et al., 2006). So, Figure 3.3(**c**) illustrates the structure of this approach, where three binary classifiers are built for each child-parent relationship.

(B.3) Local Classifier per Level (LCL)

In the LCL approach, a multi-class classifier is trained at each level of the hierarchy to identify the parent node of an instance. This method answers the question, "Which node at this level is the parent of this instance?" (Naik, 2017). Figure 3.3(**d**) illustrates this method, where two classifiers are required to handle the multi-level predictions.

We implemented all three hierarchical classification algorithms using the Hiclass library (Miranda et al., 2023), which provides ready-to-use implementations. The performance of each algorithm was evaluated based on the hierarchical metrics detailed in Equation 2.5.6.

3.2.2 Deep Learning-Based classifiers

(A) 1D Convolutional Neural Network with Self-Attention

Convolutional Neural Networks (CNNs) are widely used in various classification tasks, particularly excelling in image classification. However, they have also demonstrated impressive accuracy with text data (Murphy, 2022). While 2D CNNs are commonly used for images and 3D CNNs for video data, 1D CNNs are effective for tasks like text and sequence classification (Bishop and Bishop, 2024).

Since CNNs require numerical input, we converted each genomic sequence into a numerical format using one-hot encoding. Additionally, because the sequence lengths varied across the dataset (see Figure A.1a), we applied a post-padding to standardize all sequences to a uniform length. Each sequence is a matrix with dimensions 10821×4, where 10821 represents the maximum sequence length and 4 corresponds to the four nucleotides in the DNA alphabet.

We designed an architecture capable of both extracting valuable features from the sequences and learning dependencies between them. Transformers, widely used in Natural Language Processing (NLP), excel at learning dependencies between features. After multiple trials, we adopted a hybrid architecture combining a 1D CNN with Self-Attention, taking inspiration from Ullah et al. (2022), and Qayyum et al. (2024). The architecture is illustrated in Figure 3.4 and described subsequently:

Figure 3.4: 1D CNN with Self Attention Architecture. Details are available here.

The architecture shown above, begins with a Conv1D layer that processes the one-hot encoded, padded genomic sequences. This layer applies 128 filters with a kernel size of 4 and a stride of 1. Each filter captures local patterns in the input sequences, such as the spatial relationships between nucleotides. The ReLU activation function

$$\text{ReLu}(x) = \max(0, x) \tag{3.2.2}$$

adds non-linearity to the model, enabling it to learn complex features.

After the initial convolution, we apply a second Conv1D layer with 64 filters, followed by MaxPooling1D with a pool size of 4. Max-pooling is essential for downsampling the input representation, reducing its dimensionality while preserving the most important features. This not only improves computational efficiency but also reduces the risk of overfitting.

Next, a third Conv1D layer with 32 filters captures finer details in the sequences, followed by another MaxPooling1D layer, and finally, a fourth Conv1D layer with 16 filters. Each of these layers maintains a consistent kernel size (4) and stride (1), with padding used to preserve input shapes across layers.

The core strength of this architecture lies in the Transformer-based Self-Attention mechanism, implemented using a SeqSelfAttention layer. This mechanism weighs different positions in the sequence based on their relevance to the output, allowing the model to capture long-range dependencies. The attention function is calculated using the following Equation 3.2.3:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left[\frac{\mathbf{Q}\mathbf{K}^{\intercal}}{\sqrt{d_k}}\right]\mathbf{V}, \tag{3.2.3}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ represent the query, key, and value matrices derived from the input, and d_k (set to 16 in this study) is the dimensionality of the key.

After applying the attention mechanism, the output is flattened and passed through a Dropout layer (with a dropout rate of 0.6) to prevent overfitting. The final Dense layer performs classification over 38 Dengue lineages using a softmax activation function:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{38} e^{z_j}}, \quad i = 1, \ldots, 38. \tag{3.2.4}$$

This ensures the output is a probability distribution across all 38 classes.

We compiled the model using categorical cross-entropy as the loss function:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{j=1}^{38} y_{ij}\log(p_{ij}), \tag{3.2.5}$$

where N is the total number of samples, y_{ij} represents the one-hot encoded true label for sample i and class j, and p_{ij} is the predicted probability for sample i in class j.

(B) 2D Convolutional Neural Network with Frequency Chaos Game Representation

We improved upon the 1D CNN with Self-Attention by implementing a 2D CNN using the Frequency Chaos Game Representation (FCGR) for each sequence. This converts the sequence into a grayscale image (matrix) of shape 32×32. The architecture is illustrated in Figure 3.5:

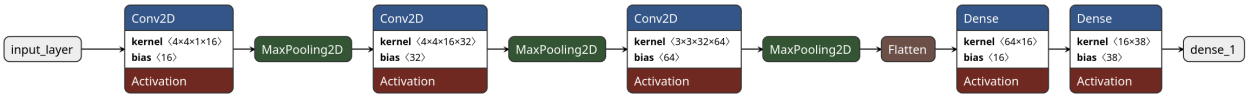


Figure 3.5: 2D CNN architecture: Each sequence is converted using FCGR into a grayscale image of size 32×32. The network consists of three convolutional layers with increasing filters (16, 32, 64) and kernel sizes of 4×4 and 3×3, each followed by ReLU activation. MaxPooling layers with a pool size of 2×2 and strides of 2×2 are applied after each convolution. The flattened output is passed through two dense layers, with the final softmax layer predicting the sequence's lineage across 38 classes. Details are available here.

3.3 New Variants Detection

Inspired by the approach outlined in Ullah et al. (2022), we implemented a Variational Autoencoder (VAE) to generate synthetic samples. Subsequently, a voting-based algorithm was designed to facilitate the early detection of potential new variants.

3.3.1 Synthetic Samples with Variational Autoencoder

Variational Autoencoders (VAEs), initially introduced by Kingma and Welling (2013), are part of the broader Generative Artificial Intelligence family, which includes models such as Generative Adversarial Networks (GANs) and Denoising Diffusion Probabilistic Models (DDPMs). These models have garnered significant attention for their ability to generate synthetic samples that closely resemble the original data, often indistinguishable from real data. The generated samples are applied across various fields, such as creating non-existent objects or augmenting data for training larger models. For instance, GAN-based websites like This Person Does Not Exist generate highly realistic human faces that cannot be easily distinguished from actual images.

Though diffusion models often excel in producing high-quality samples that can outperform both GANs and VAEs, they are considerably slower to train (Géron, 2019). In contrast, VAEs are simpler and faster to train, making them a practical choice for generating synthetic Dengue samples for our study.

VAEs work by learning a latent space distribution that captures the underlying structure of the input data. They consist of two core components: an **encoder** and a **decoder**. The encoder compresses the input data (in this case, one-hot encoded sequences) into a latent representation $\mathbf{z}$ that follows a Gaussian distribution $\mathcal{N}(\mathbf{z}|\mu, \sigma^2)$. The decoder then reconstructs the input from this latent space, generating new samples that resemble the original data. To ensure that the generated output closely matches the input, the total loss, which combines the reconstruction error and Kullback-Leibler (KL) divergence, is minimized (Kingma and Welling, 2013; Odaibo, 2019). The architecture we used for our VAE is presented in Figure 3.6, with further details provided subsequently:

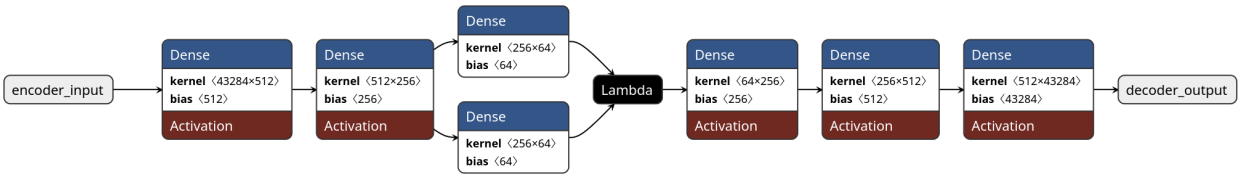


Figure 3.6: Variational Autoencoder Architecture used. Details are available here.

1. **Encoder : Compressing the Input to the Latent Space**

   The encoder compresses the input sequences into a smaller, lower-dimensional representation called the **latent space**. The input data(one-hot encoded sequences) first passes through two dense layers: the first with 512 neurons and the second with 256 neurons, both using the ReLU activation function to introduce non-linearity, allowing the network to capture complex relationships in the data. The encoder then outputs two vectors: z_μ, and $z_{\log(\sigma^2)}$, representing the mean and log-variance of the latent variable $\mathbf{z}$.

2. **Latent Space Sampling**

   The latent variable $\mathbf{z}$ is then sampled using the Reparametrization trick to ensure the model is differentiable during the training(Backpropagation):

$$\mathbf{z} = z_\mu + \epsilon \odot \exp\left(\frac{1}{2} z_{\log(\sigma^2)}\right), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3.3.1}$$

In our case, the latent dimension was set to 64, meaning each sequence is reduced to a 64-dimensional vector in the latent space.

3. **Decoder: Reconstructing the Sequence from the Latent Space**

   Once sampled, $\mathbf{z}$ is fed into the decoder to reconstruct the original genomic sequence. The decoder is designed symmetrically to the encoder, with two dense layers of 256 and 512 neurons (ReLU activation). The final layer of the decoder uses a sigmoid activation function to output a reconstructed vector, which has the same size as the original input sequence.

4. **Loss function: Balancing Reconstruction and Regularization**

   The VAE is trained by minimizing a combination of two loss functions:

   (a) **Reconstruction Loss**

   This loss measures how closely the reconstructed sequences match the original input sequences. We use binary cross-entropy to calculate the reconstruction loss across each nucleotide position, ensuring that the model learns to produce realistic genomic sequences.

$$\mathcal{L}_{recon} = -\sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right], \tag{3.3.2}$$

   where y_i and $\hat{y}_i$ are the original and reconstructed sequences, respectively.

   (b) **Kullback-Leibler Divergence Loss**

   This loss term regularizes the latent space by ensuring that the distribution of $\mathbf{z}$ is close to a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This makes the latent space smooth and continuous, which is essential for generating new and diverse sequences.

$$\mathcal{L}_{KL} = -\frac{1}{2} \sum_{i=1}^{d} \left[1 + z_{\log(\sigma^2)} - z_\mu^2 - \exp\left(z_{\log(\sigma^2)}\right) \right]. \tag{3.3.3}$$

   The total loss is then a combination of these two components (3.3.2) and (3.3.3):

$$\mathcal{L}_{VAE} = \mathcal{L}_{recon} + \beta \cdot \mathcal{L}_{KL}, \tag{3.3.4}$$

   where β (set to 1 in our case) is a hyperparameter that controls the trade-off between the two losses. A higher β value encourages better latent space regularization, while a lower β focuses more on accurate reconstruction.

Once the VAE generates synthetic sequences, they are converted back to their nucleotide form by reverse one-hot encoding, which maps the binary vectors back to the corresponding nucleotides (A, C, G, T). The generated sequences are then saved in `.fasta` format for further classification and analysis as described in Section 4.3.

3.3.2 Proposed Early Detection Algorithm

To facilitate the early detection of new Dengue variants, we propose a strategy that leverages at least three high-performing classifiers, each demonstrating near-perfect accuracy in classifying sequences into known lineages. Our approach employs a voting-based mechanism to determine whether a given sequence is novel or belongs to an established lineage.

In Machine Learning, two common voting methods are often utilized : Hard Voting and Soft Voting (Géron, 2019; Brownlee, 2020). Hard Voting selects the most frequently predicted class by taking the mode of all predictions, while Soft Voting considers the class with the highest average probability, aggregating the outputs from all classifiers. Both methods rely on the assumption that the classifiers are highly accurate($\geq 80\%$ accuracy score) and largely uncorrelated (Géron, 2019).

We chose to use at least three independent classifiers, each trained with different feature extraction techniques, and then apply a soft voting. The following Algorithm 1 details the voting process, where the predictions of each classifier contribute to the final decision on whether a sequence belongs to an existing lineage or not.

Algorithm 1 Voting-Based Detection Algorithm for New Dengue Variants

Input:
$M_1, M_2, ..., M_n$: Pretrained models
$L_1, L_2, ..., L_n$: True encoded lineage names for each model
Seq: New sequences
$Threshold$: Confidence threshold (default = 0.8)
Output: Predicted lineage or "Uncertain" for each sequence
Load and preprocess Seq into a tensor X
for each sequence x_j in X **do**
  Initialize $Proba\_Dict \leftarrow \{\}$ ▷ To store lineage probabilities
  for each model M_i **do**
    $P_i \leftarrow$ probabilities for x_j from model M_i
    for each lineage k in P_i **do**
      $lineage\_name \leftarrow L_i[k]$
      Append $P_i[k]$ to $Proba\_Dict[lineage\_name]$
    end for
  end for
  for each lineage in $Proba\_Dict$ **do**
    Compute average probability for each lineage
  end for
  $predicted\_lineage \leftarrow$ lineage with highest average probability
  $confidence \leftarrow$ highest average probability
  if $confidence < Threshold$ **then**
    Mark x_j as "Uncertain"
  else
    Assign x_j to $predicted\_lineage$
  end if
end for
Return the predicted lineages or "Uncertain" for each sequence
Flag sequences marked as "Uncertain" for further investigation.

4. Results and Discussion

4.1 Machine Learning Classifiers Results

This section focuses on comparing the outcomes of flat and hierarchical classifiers presented in Subsection 3.2.1. The comparison is conducted using two primary approaches: first, by considering the optimal k-mer (k=5) value, and second, by using the optimal resolution (=32) of the Frequency Chaos Game Representation (FCGR) feature extraction technique. As mentioned earlier in Chapter 3, all optimal hyperparameters were determined using the Optuna library and can be found in Section A.3.

4.1.1 Approach 1: 5-mer Encoding Feature Extraction Technique

Using the 5-mers encoding technique, each sequence is represented as a row vector of shape 1×4^5 (or 1×1024) in the vectorised matrix corresponding to the dataset. Each entry in this matrix is the frequency of a unique 5-mer in the sequence. Figure 4.1 shows the results:

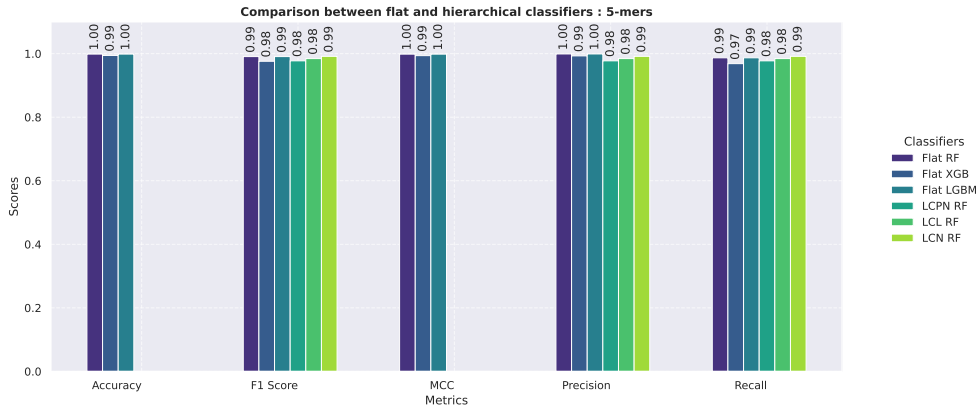


Figure 4.1: The x-axis represents the metrics: Accuracy, F1-Score, Matthews Correlation Coefficient (MCC), Precision, and Recall for flat classifiers (Random Forest, XGBoost, and LightGBM), and Hierarchical (F1-Score, Precision, and Recall) for hierarchical classifiers (LCPN, LCN, and LCL) with Random Forest as their base classifier. The y-axis shows the scores, ranging from 0 to 1, with higher scores indicating better performance.

From Figure 4.1, we observe the following: On one hand, the flat classifiers performed exceptionally well, with all metrics above 0.97. While XGBoost slightly lagged behind Random Forest and LightGBM, all three classifiers were able to accurately classify almost all sequences across the 38 Dengue lineages, as reflected by their high Precision and Recall.

On the other hand, the hierarchical classifiers, which predict across multiple levels (Serotype, Genotype, Major Lineage, Minor Lineage), also achieved high scores but were slightly outperformed by the flat classifiers. Despite this, they provided more detailed insights into where misclassifications occurred within the lineage hierarchy.

In summary, both flat and hierarchical classifiers performed with near-perfect F1-Scores, which combine both Precision and Recall, demonstrating their effectiveness in classifying Dengue sequences. Flat classifiers had a slight edge, but the hierarchical models offered valuable multi-level classification insights.

4.1.2 Approach 2: Frequency Chaos Game Representation Feature Extraction Technique with Resolution = 32

In this case, each sequence is represented as a 32×32 matrix, which is then flattened into a vector of shape 1×1024. The entries of this vector are derived using Equation 2.4.4. Figure 4.2 displays the results:

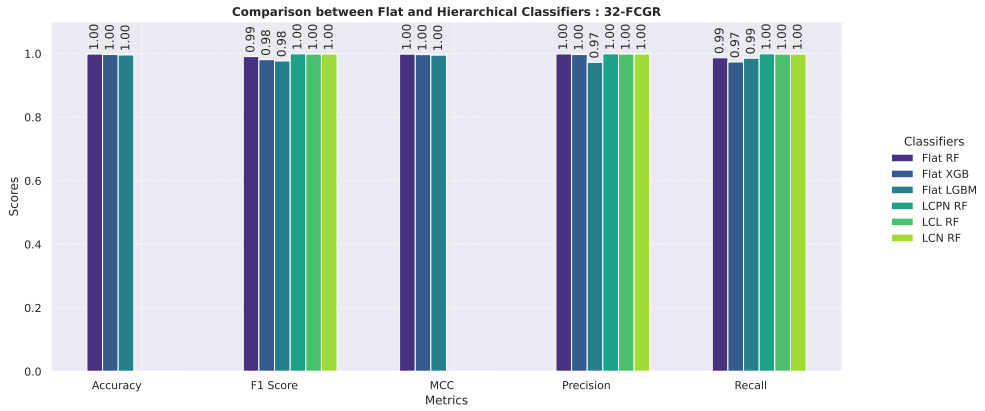


Figure 4.2: The x-axis represents the metrics: Accuracy, F1-Score, Matthews Correlation Coefficient (MCC), Precision, and Recall for flat classifiers (Random Forest, XGBoost, and LightGBM), and Hierarchical (F1-Score, Precision, and Recall) for hierarchical classifiers (LCPN, LCN, and LCL) with Random Forest as their base classifier. The y-axis shows the scores, ranging from 0 to 1, with higher scores indicating better performance.

As shown in Figure 4.2, all classifiers achieved near-perfect performance, slightly better than the results in Figure 4.1. The scores for Accuracy, F1-Score, MCC, Precision, and Recall remained consistently high. While XGBoost and LightGBM were slightly outperformed by other classifiers, the difference is not significant, and their performance was still strong.

As mentioned earlier, hierarchical classifiers provide more detailed insights by predicting labels across four levels (Serotype, Genotype, Major Lineage, and Minor Lineage). Achieving perfect (100%) hierarchical scores(F1-Score, Precision and Recall) demonstrates their ability to minimize misclassification across the 38 different lineages. This highlights their effectiveness in making accurate, multi-level predictions for each sequence.

Overall, the results from both feature extraction techniques, shown in Figures 4.1 and 4.2, confirm that the classifiers used in this study are highly effective at classifying Dengue genomic sequences into 38 lineages. We have also plotted the confusion matrices (Figures A.4 and A.7) for the flat classifiers, as well as the feature importance (Figure A.8) for Random Forest, to identify the most relevant features that enable the model to make accurate classifications.

Next, we will examine the performance of Deep Learning-based algorithms in Section 4.2.

4.2 Deep Learning Classifiers Results

In this section, we present the results of the Deep Learning-based models described in Subsection 3.2.2: the 1D CNN with Self-Attention and the 2D CNN using FCGR. Let's begin with the first model.

4.2.1 One-dimensional Convolutional Neural Network with Self-Attention Mechanism

For this model, each sequence is represented as a 10821×4 matrix using one-hot encoding. The model was initialized using the Xavier-Glorot kernel initializer. To prevent overfitting, we applied L_2 regularization with a coefficient of 0.001. The model was compiled using categorical cross-entropy loss, and the Adam optimizer with a learning rate of 0.0001. The batch size was set to 32, and the model was trained over 100 epochs. The training process is shown in Figure 4.3:

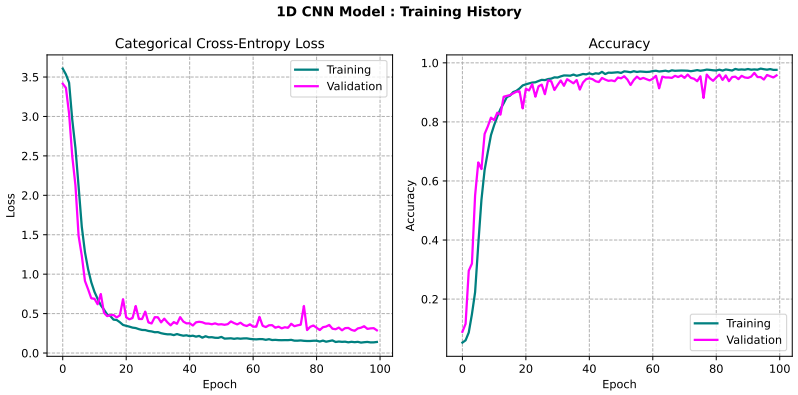


Figure 4.3: On the left, the categorical cross entropy loss curves decrease on both training and validation sets in the same range and converge towards values less than 0.5. On the Right, the accuracy curves increase in the same range and converge around 0.98. The visible patterns on all curves imply that our model is learning and generalize well to the unseen data. The slight fluctuations visible on the validation curves reveal a slight overfitting, subject to a further improvement.

The results shown in Figure 4.3, are further supported by the Receiver Operating Characteristic (ROC) and Precision-Recall curves in Figure 4.4:

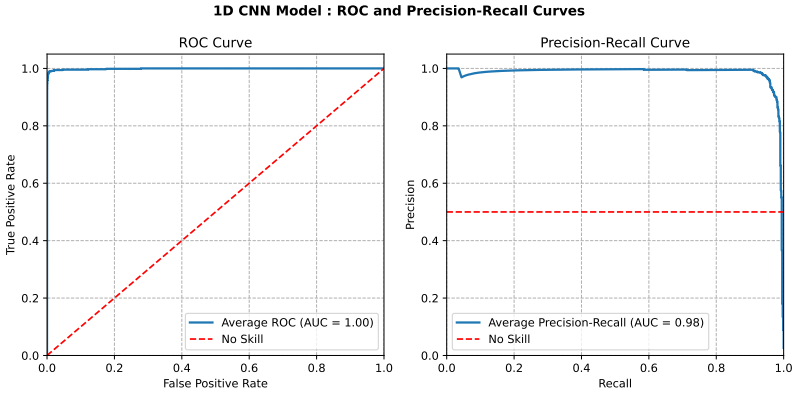


Figure 4.4: On the left, the ROC curve shows an Area Under the Curve (AUC) of 1.00, indicating excellent classification performance. On the right, the Precision-Recall curve reveals an AUC of 0.98, demonstrating the model's high Precision and Recall across all classes(lineages).

The results in Figures 4.3 and 4.4 show that the 1D CNN model with a Self-Attention mechanism is highly effective at classifying Dengue genomic sequences. However, the slight fluctuations in the loss and accuracy curves, along with a mean AUC of 0.98 on the Precision-Recall curve, might be optimized. Therefore, in the next subsection 4.2.2, we present an alternative approach to address this.

4.2.2 Two-dimensional Convolutional Neural Network

In this model, each sequence is represented as a 32×32 grayscale image (matrix) using the Frequency Chaos Game Representation (FCGR). This image is used as the input for the model shown in Figure 3.5. The model was initialized using the Xavier-Glorot kernel initializer. To avoid overfitting, L_2 regularization (0.001) was applied to each layer. The model was compiled with categorical cross-entropy loss and the Adam optimizer, with a learning rate of 0.0001. The batch size was set to 32, and the model was trained over 100 epochs. The training process is shown in Figure 4.5:

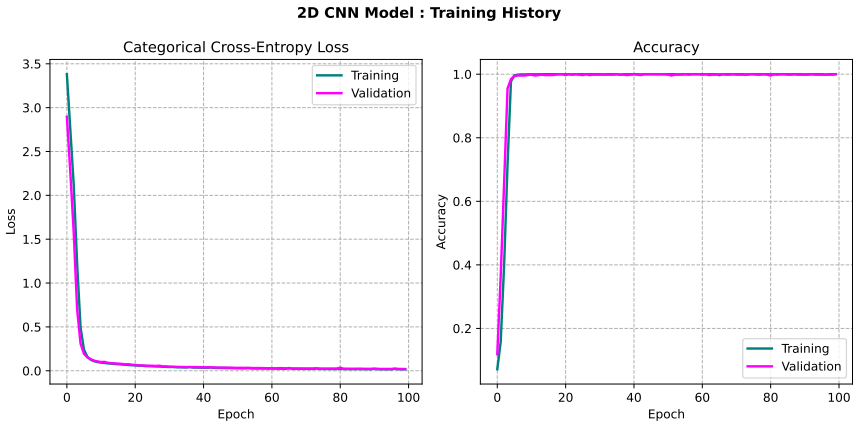


Figure 4.5: On the left, the categorical cross entropy loss curves decrease on both training and validation sets in perfectly the same range and converge rapidly towards 0.00. On the Right, the accuracy curves increase exaclty in the same range and converge rapidly to 1.00, the perfect score. The visible patterns on all curves imply that our model is learning perfectly and generalize perfectly to the unseen data without signs of overfitting.

The excellent performance shown in Figure 4.5 is further supported by the ROC and Precision-Recall curves in Figure 4.6:

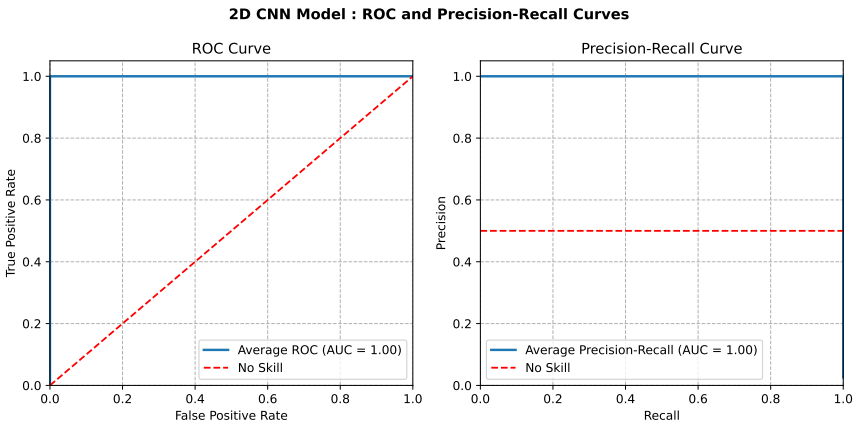


Figure 4.6: On the left, the ROC curve shows an Area Under the Curve (AUC) of 1.00, indicating excellent classification performance. Similarly, on the right, the Precision-Recall curve also shows an AUC of 1.00, demonstrating perfect Precision and Recall across all 38 classes.

Based on these results, it's clear that the suggested 2D CNN outperforms the baseline 1D CNN with

Self-Attention. While the 1D CNN achieved an average AUC of 0.98 in the Precision-Recall curve, the 2D CNN reached a perfect score of 1.00. This shows that the 2D CNN is more effective in classifying sequences across all 38 lineages. To further analyse how each model distinguishes between the lineages, we plotted the confusion matrices; one of them is given in the following Figure 4.7, and the other is placed as an appendix in Figure A.9 due to space constraints.

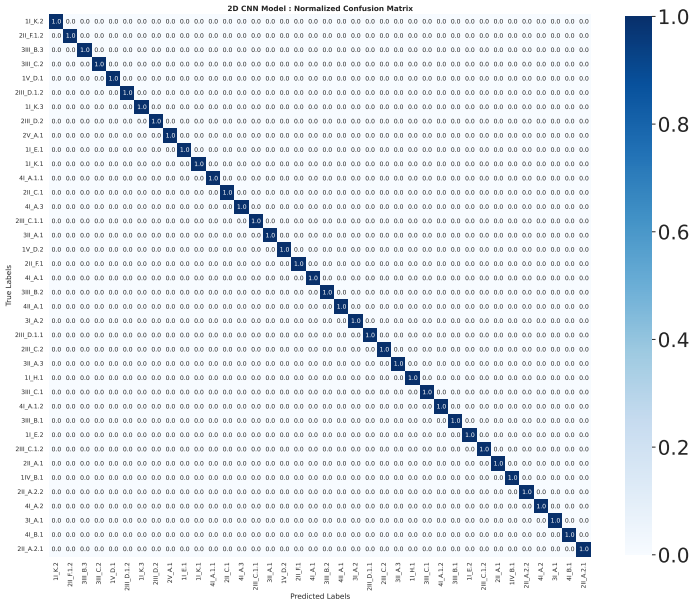


Figure 4.7: 2D CNN Normalized Confusion Matrix: Each row corresponds to the actual lineage, while each column represents the predicted lineage. The matrix is normalised, meaning the values range from 0 to 1. High values along the diagonal indicate correct classifications, while off-diagonal values suggest misclassifications. Therefore, we can see that the model is efficient, as it perfectly classified all samples.

In summary, all classifiers (flat, hierarchical, and deep learning-based) achieved near-perfect scores, with all metrics exceeding 97%. This provides strong evidence that the models successfully reached state-of-the-art performance in classifying Dengue sequences. As a result, they hold significant potential for enabling the early detection of new variants, as discussed in the next section 4.3.

4.3 Early Detection of Potential New Dengue Variants

Now that we have trained powerful classifiers, capable to classify Dengue genomic sequences into lineages, the next question is: *how and where to find sequences to be considered as new variants?*.

To address this, we trained a Variational Autoencoder(VAE) to generate synthetic samples much closer to the original Dengue sequences. The architecture of the VAE is shown in Figure 3.6, where it receives

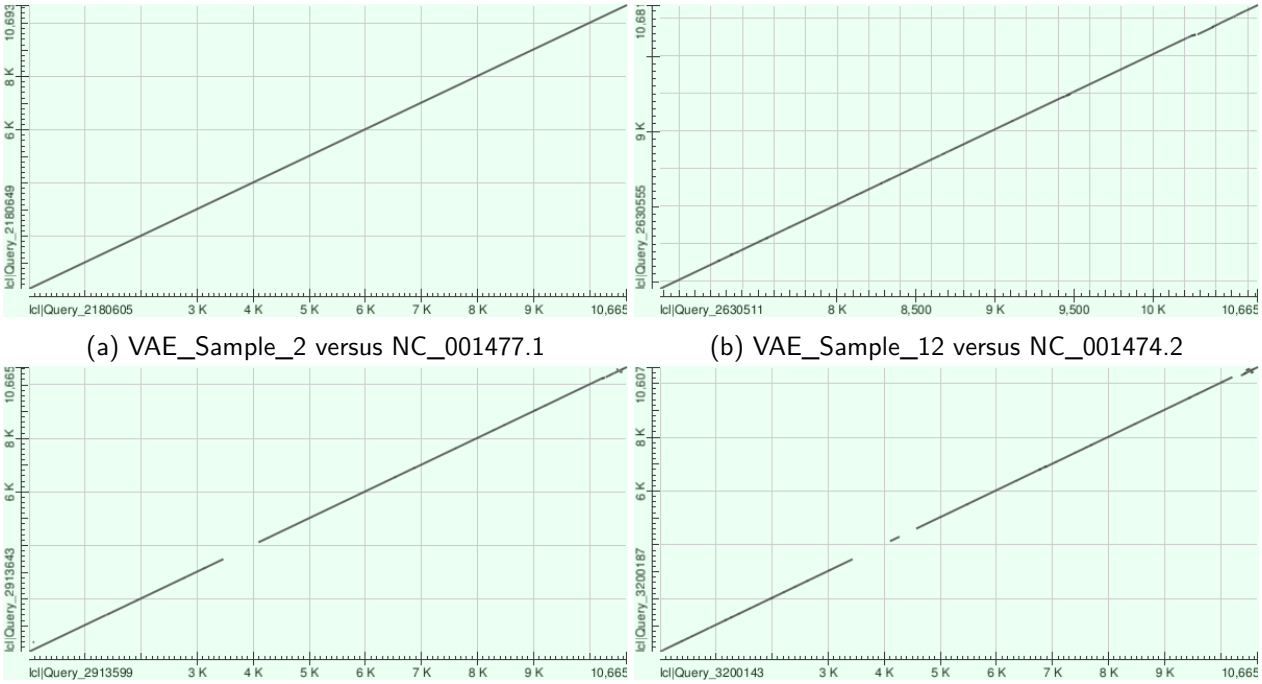
one-hot encoded sequences of shape 10821×4. Using a 64-dimensional latent space, the model was trained with a batch size of 32 over 250 epochs, as illustrated in Figure A.10. After training, we sampled 100 sequences from the decoder, and used the Nexclade tool to assess the quality of the generated sequences, discarding any invalid samples.

Out of the 100 samples, Nextclade validated 43 sequences with high scores (over 90%), showing similarity to known Dengue serotypes.

We then applied our soft voting-based proposed Algorithm 1 for early detection. In this step, we loaded three pretrained models: the LightGBM model (with 5-mers feature extraction), the 1D CNN with Self-Attention (using one-hot encoding), and the 2D CNN (with 32-resolution Frequency Chaos Game Representation). Each of them was loaded along with the saved lineages from the training.

A lineage was classified only if its mean probability exceeded 80%; otherwise, the sequence was flagged as 'Uncertain,' indicating a potential new variant. We then used the Genome Detective Dengue Virus Typing Tool to classify the same sequences and compared the results.

The Genome Detective Dengue Virus Typing Tool identified lineages for the samples, indicating also their associated known sequences of reference. Per serotype, we downloaded one of the indicated references from the NCBI database and aligned some samples using the NCBI-BLASTN tool. The resulting dot plots, shown in Figure 4.8, display the alignment results:



(a) VAE_Sample_2 versus NC_001477.1

(b) VAE_Sample_12 versus NC_001474.2

(c) VAE_Sample_35 versus NC_001475.2

(d) VAE_Sample_56 versus NC_002640.1

Figure 4.8: Dot-plots: The reference (Query) is aligned on the x-axis and the synthetic sample (Subject) is aligned on the y-axis; the numbers on the axes correspond to the positions of nucleotides within each sequence. The diagonal lines represent matched nucleotides at the same rank. Subfigure 4.8a shows a perfect alignment for DENV1: IV_B.1. Subfigure 4.8b shows a perfect match for DENV2: 2V_A.1. Subfigures 4.8c and 4.8d show minor gaps for DENV3: 3III_C.2 and DENV4: 4II_A.1, respectively. Full alignments, scores, and gaps for each case can be found here.

Since all samples showed high similarity to existing Dengue cases, we flagged any sequence that the Genome Detective Dengue Virus Typing Tool failed to fully classify as 'Uncertain'; potential new variant requiring deeper analysis by domain experts. The results from Genome Detective Dengue Virus Typing Tool and our voting algorithm are compared in Figure 4.9:

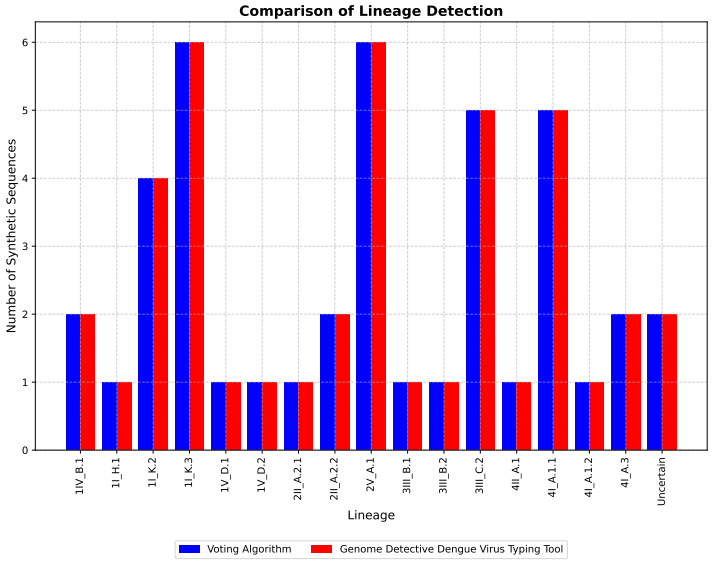


Figure 4.9: Lineage Detection: The classification report from our proposed variant detection algorithm matches perfectly with the results from the Genome Detective Dengue Virus Typing Tool. Both flagged two samples as 'Uncertain,' indicating potential new variants. In practice, these uncertain samples would require further analysis by some Oracle(domain experts). Full results are available here.

Over a span of ten days, we independently classified the valid sequences multiple times, each yielding consistent results. The classification times were also recorded, with results shown in Figure 4.10:

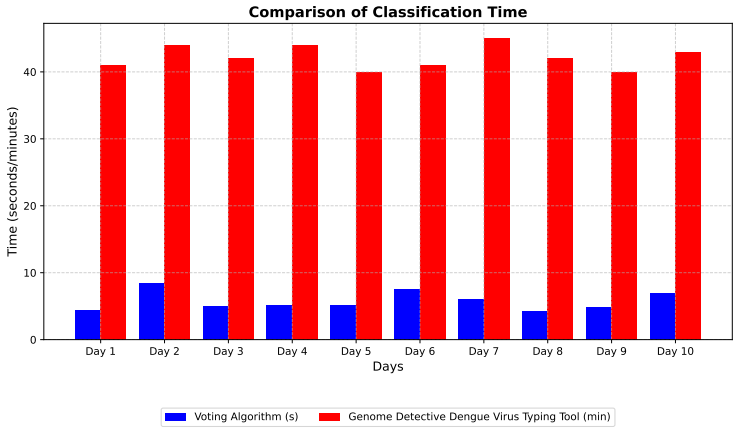


Figure 4.10: Timing Comparison: The voting ensemble classified the 43 sequences in under 10 seconds, whereas the Genome Detective Dengue Virus Typing Tool took at least 40 minutes.

In summary, our proposed algorithm is efficient and effective in flagging potential new variants quickly, offering valuable support for genomic surveillance and helping experts detect new Dengue variants in a timely manner.

5. Conclusion, Limitations and Perspectives

In this work, we explored the combination of some feature extraction techniques, widely used in Genomics, with Machine Learning (ML) and Deep Learning (DL) algorithms to effectively classify genomic sequences of the Dengue virus into lineages, and enable early detection of new variants. We specifically used k-mer encoding, Frequency Chaos Game Representation (FCGR), and one-hot encoding as feature extraction techniques to transform sequences into numerical formats suitable for ML and DL models. To address class imbalance, we applied the Synthetic Minority Oversampling Technique (SMOTE).

Following this, we considered two broad categories of algorithms to achieve classification from different perspectives. On the one hand, we employed ML algorithms, including flat classifiers (Random Forest, XGBoost, and LightGBM), as well as hierarchical classifiers namely Local Classifier Per (Parent Node, Level, and Node). On the other hand, we utilized DL algorithms, starting with a one-dimensional Convolutional Neural Network with an Attention mechanism. We later improved this by adopting a two-dimensional Neural Network, treating each sequence as a grayscale image using the FCGR.

The mentioned algorithms, coupled with the aforementioned feature extraction techniques, produced promising results, measured using metrics such as the F1-Score and/or Accuracy, all reaching a minimum of 97%, demonstrating their ability to efficiently classify Dengue sequences into lineages. Then, we generated synthetic sequences using Variational Autoencoder (VAE), where we considered only sequences with at least 90% similarity to existing Dengue variants confirmed by Nextclade. These sequences were subsequently considered as our potential new variants. We then proposed an algorithm for detecting them, based on the voting principle in Machine Learning. This algorithm uses at least three of the aforementioned pre-trained models to vote on the lineage of a new sequence.

By using both our proposed detection algorithm and the Genome Detective Dengue Virus Typing Tool server to classify these new sequences, we obtained consistent results. Some sequences were classified as "uncertain", requiring the intervention of an oracle(expert in the field) for further analysis and appropriate conclusions. These experiments, repeated over 10 days, prove that our algorithm allows the classification of new sequences in an extremely reduced time, while reaching the same conclusions as Genome Detective Dengue Virus Typing Tool, offering a rapid solution for Data Analysts.

The results obtained show that we have achieved our objectives and have made a contribution in this field. However, as with any human endeavor, our study has some imperfections.

The fact that our study was limited to 38 available lineages reduces the scope of some of our generalizations. Additionally, our algorithms process data abstractly, without considering underlying biological realities that could reveal crucial information.

This work represents a first initiative in this area and paves the way for future improvements. For instance, generalizing this study could involve accessing sequences from all current Dengue lineages. The incorporation of other Generative Artificial Intelligence algorithms, such as GANs or Diffusion Models, could produce high-quality sequences in sufficient quantity, enabling the use of sophisticated and officially recognized algorithms on large datasets. Furthermore, integrating biological information about the pathogen, coupled with interpretable and explainable AI techniques, could help overcome the "black-box" nature algorithms.

We hope that the completion of these tasks will contribute significantly to genomic surveillance and the elimination of Dengue, a current threat to public health.

Acknowledgements

First of all, I would like to thank God, Master of time and circumstances, for the health, safety, and wonders He continuously grants us, day after day, free of charge.

Next, I would like to express my gratitude to AIMS for selecting us to participate in this exceptional program across Africa. On this occasion, I would also like to extend my heartfelt thanks to Google DeepMind, whose generous support made our participation in this fully funded program possible. Without this, my studies at this level would not have been feasible.

In a special way, I deeply thank my supervisors, Dr. Joicymara Xavier and Dr. Houriiyah Tegally, for their guidance throughout this thesis and their unwavering support, despite their busy schedules. May you be richly blessed. I would also like to thank Sylvestre, my mentor at DeepMind, for his invaluable advice and numerous meetings that helped illuminate our path. Additionally, I would like to thank Mrs. Lynne Teixera for her wonderful and motivating English lessons at AIMS, which allowed us to go from zero to our current level of English. We are infinitely grateful to you.

To my family, friends, colleagues, and peers, I express my deepest appreciation for your support and encouragement during these 10 months, a period filled with challenges and sacrifices. Lastly, to my soulmate, Béatrice, who, despite the distance, called me every evening to offer her support and motivation, turning stress into joy, I thank you from the bottom of my heart.

References

Adetiba, E., Abolarinwa, J. A., Adegoke, A. A., Taiwo, T. B., Ajayi, O. T., Abayomi, A., Adetiba, J. N., and Badejo, J. A. Deepcovid-19: A model for identification of covid-19 virus sequences with genomic signal processing and deep learning. *Cogent Engineering*, 9(1):2017580, 2022.

Agbodoyetin, A. J. S. Ai-enhanced classification and early detection of sars-cov-2 variants for timely public health response. Master's thesis, African Institute for Mathematical Sciences (AIMS), Cape Town, South Africa, June 2024.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

American Society for Microbiology. Viruses, 2024. URL https://asm.org/browse-by-keyword/viruses. Accessed on September 20, 2024.

Avila Santos, A. P., de Almeida, B. L., Bonidia, R. P., Stadler, P. F., Stefanic, P., Mandic-Mulec, I., Rocha, U., Sanches, D. S., and de Carvalho, A. C. Biodeepfuse: a hybrid deep learning approach with integrated feature extraction techniques for enhanced non-coding rna classification. *RNA biology*, 21 (1):1–12, 2024.

Benson, D. A., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. Genbank. *Nucleic acids research*, 42(Database issue):D32, 2014.

Bishop, C. M. and Bishop, H. *Deep Learning: Foundations and Concepts*. Springer, 2024.

Bonidia, R. P., Sampaio, L. D., Domingues, D. S., Paschoal, A. R., Lopes, F. M., de Carvalho, A. C., and Sanches, D. S. Feature extraction approaches for biological sequences: a comparative study of mathematical features. *Briefings in Bioinformatics*, 22(5):bbab011, 2021.

Brownlee, J. *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning*. Machine Learning Mastery, 2020.

Centers for Disease Control and Prevention. Coronavirus disease 2019 (covid-19), Sept. 2023. URL https://www.cdc.gov/coronavirus/2019-ncov/variants/variant-classifications.html. Accessed: 2024-10-14.

Chauhan, N., Gaur, K. K., Asuru, T. R., and Guchhait, P. Dengue virus: pathogenesis and potential for small molecule inhibitors. *Bioscience Reports*, 44(8):BSR20240134, 2024.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Freedman, D. O. A new dengue vaccine (TAK-003) now WHO recommended in endemic areas; what about travellers? *Journal of Travel Medicine*, 30(7):taad132, 10 2023. ISSN 1708-8305. doi: 10.1093/jtm/taad132. URL https://doi.org/10.1093/jtm/taad132.

Géron, A. Hands-on machine learning with scikit-learn, keras & tensorflow farnham. *Canada: O'Reilly*, 2019.

Gunasekaran, H., Ramalakshmi, K., Rex Macedo Arokiaraj, A., Deepa Kanmani, S., Venkatesan, C., and Suresh Gnana Dhas, C. Analysis of dna sequence classification using cnn and hybrid models. *Computational and Mathematical Methods in Medicine*, 2021(1):1835056, 2021.

Gupta, G., Khan, S., Guleria, V., Almjally, A., Alabduallah, B. I., Siddiqui, T., Albahlal, B. M., Alajlan, S. A., and Al-Subaie, M. Ddpm: A dengue disease prediction and diagnosis model using sentiment analysis and machine learning algorithms. *Diagnostics*, 13(6):1093, 2023.

Guzman, M. G. and Harris, E. Dengue. *The Lancet*, 385(9966):453–465, 2015.

Hill, V., Cleemput, S., Pereira, J. S., Gifford, R. J., Fonseca, V., Tegally, H., Brito, A. F., Ribeiro, G., de Souza, V. C., Brcko, I. C., et al. A new lineage nomenclature to aid genomic surveillance of dengue virus. *PLoS biology*, 22(9):e3002834, 2024. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11118645/.

Isaev, A. et al. *Introduction to mathematical methods in bioinformatics*. Springer, 2004.

Itaya, Y., Tamura, J., Hayashi, K., and Yamamoto, K. Asymptotic properties of matthews correlation coefficient. *arXiv preprint arXiv:2405.12622*, 2024.

Johns Hopkins Medicine. A new strain of coronavirus: What you should know, 2024. URL https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/a-new-strain-of-coronavirus-what-you-should-know. Accessed: 2024-10-14.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL https://api.semanticscholar.org/CorpusID:216078090.

Kiritchenko, S., Matwin, S., Nock, R., and Famili, A. F. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Advances in Artificial Intelligence: 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Québec City, Québec, Canada, June 7-9, 2006. Proceedings 19*, pages 395–406. Springer, 2006.

Kolekar, P., Kale, M., and Kulkarni-Kale, U. Alignment-free distance measure based on return time distribution for sequence analysis: applications to clustering, molecular phylogeny and subtyping. *Molecular phylogenetics and evolution*, 65(2):510–522, 2012.

Kularatne, S. A. and Dalugama, C. Dengue infection: Global importance, immunopathology and management. *Clinical Medicine*, 22(1):9–13, 2022.

Kwan, H. K. Numerical representation of dna sequences. In *2009 IEEE International Conference on Electro/Information Technology*, pages 307–310, 2009. doi: 10.1109/EIT.2009.5189632.

Leimeister, C.-A., Boden, M., Horwege, S., Lindner, S., and Morgenstern, B. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, 30(14):1991–1999, 2014.

LLC, T. W. Medical and scientific illustration, 2018. URL https://www.teresewinslow.com. © 2018 Terese Winslow LLC, U.S. Govt. has certain rights. Staging images may be licensed separately. Created for the National Cancer Institute. Accessed: 2024-08-11.

Martins, I. C., Ricardo, R. C., and Santos, N. C. Dengue, west nile, and zika viruses: potential novel antiviral biologics drugs currently at discovery and preclinical development stages. *Pharmaceutics*, 14 (11):2535, 2022.

Mayo Clinic. Dengue fever: Symptoms and causes, 2024. URL https://www.mayoclinic.org/diseases-conditions/dengue-fever/symptoms-causes/syc-20353078. Accessed: 2024-09-21.

Miranda, F. M., Köhnecke, N., and Renard, B. Y. Hiclass: a python library for local hierarchical classification compatible with scikit-learn. *Journal of Machine Learning Research*, 24(29):1–17, 2023.

Mumtaz, Z., Rashid, Z., Saif, R., and Yousaf, M. Z. Deep learning guided prediction modeling of dengue virus evolving serotype. *Heliyon*, 2024.

Murphy, K. P. *Probabilistic machine learning: an introduction*. MIT press, 2022.

Naik, A. *Hierarchical classification with rare categories and inconsistencies*. PhD thesis, George Mason University, 2017.

Natali, E. N., Babrak, L. M., and Miho, E. Prospective artificial intelligence to dissect the dengue immune response and discover therapeutics. *Frontiers in Immunology*, 12:574411, 2021.

National Human Genome Research Institute. A brief guide to genomics, 2024. URL https://www.genome.gov/18016863/a-brief-guide-to-genomics/. Accessed: 2024-08-11.

NVIDIA. What is xgboost?, 2024. URL https://www.nvidia.com/en-us/glossary/xgboost/. Accessed: 2024-10-13.

Odaibo, S. Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv:1907.08956*, 2019.

Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., and Phillippy, A. M. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17: 1–14, 2016.

Paul, T., Vainio, S., and Roning, J. Detection of intra-family coronavirus genome sequences through graphical representation and artificial neural network. *Expert Systems With Applications*, 194:116559, 2022.

Qayyum, A., Benzinou, A., Saidani, O., Alhayan, F., Khan, M. A., Masood, A., and Mazher, M. Assessment and classification of covid-19 dna sequence using pairwise features concatenation from multi-transformer and deep features with machine learning models. *SLAS technology*, 29(4):100147, 2024.

Rambaut, A., Holmes, E. C., O'Toole, Á., Hill, V., McCrone, J. T., Ruis, C., Du Plessis, L., and Pybus, O. G. A dynamic nomenclature proposal for sars-cov-2 lineages to assist genomic epidemiology. *Nature microbiology*, 5(11):1403–1407, 2020.

Randhawa, G. S. *Machine Learning with Digital Signal Processing for Rapid and Accurate Alignment-Free Genome Analysis: From Methodological Design to a Covid-19 Case Study*. PhD thesis, The University of Western Ontario (Canada), 2020.

Randhawa, G. S., Hill, K. A., and Kari, L. Ml-dsp: Machine learning with digital signal processing for ultrafast, accurate, and scalable genome classification at all taxonomic levels. *BMC genomics*, 20: 1–21, 2019.

Rigatti, S. J. Random forest. *Journal of Insurance Medicine*, 47(1):31–39, 2017.

Roy, S., Biswas, A., Shawon, M. T. A., Akter, S., and Rahman, M. M. Land use and meteorological influences on dengue transmission dynamics in dhaka city, bangladesh. *Bulletin of the National Research Centre*, 48(1):32, 2024.

Sabir, M. J., Al-Saud, N. B. S., and Hassan, S. M. Dengue and human health: A global scenario of its occurrence, diagnosis and therapeutics. *Saudi Journal of Biological Sciences*, 28(9):5074–5080, 2021.

Saha, I., Ghosh, N., Maity, D., Seal, A., and Plewczynski, D. Covid-deeppredictor: recurrent neural network to predict sars-cov-2 and other pathogenic viruses. *Frontiers in genetics*, 12:569120, 2021.

Shaukat, M. A. Comparative study of encoded and alignment-based methods for virus taxonomy classification. *Scientific reports*, 13(1):18662, 2023.

Shiraj, T. B. and Yousuf, M. A. A study to classify virus genome through analyzing dna sequences using transformer model. In *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, pages 1275–1280. IEEE, 2024.

Silla, C. N. and Freitas, A. A. A survey of hierarchical classification across different application domains. *Data mining and knowledge discovery*, 22:31–72, 2011.

Singh, A. and Taylor-Robinson, A. W. Vector control interventions to prevent dengue: Current situation and strategies for future improvements to management of aedesin india. *Journal of Infectious Disease and Pathology*, 2(1):1–8, 2017.

Singh, S. *Machine learning and systems biology in genomics and health*. Springer, 2022.

Thind, A. S. and Sinha, S. Using chaos-game-representation for analysing the sars-cov-2 lineages, newly emerging strains and recombinants. *Current Genomics*, 24(3):187, 2023.

Togrul, M. and Arslan, H. Detection of sars-cov-2 main variants of concerns using deep learning. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5. IEEE, 2022.

Ullah, A., Malik, K. M., Saudagar, A. K. J., Khan, M. B., Hasanat, M. H. A., AlTameem, A., AlKhathami, M., and Sajjad, M. Covid-19 genome sequence analysis for new variant prediction and generation. *Mathematics*, 10(22):4267, 2022.

Waman, V. P., Kolekar, P., Ramtirthkar, M. R., Kale, M. M., and Kulkarni-Kale, U. Analysis of genotype diversity and evolution of dengue virus serotype 2 using complete genomes. *PeerJ*, 4:e2326, 2016.

World Health Organization. Weekly epidemiological record, 2023, vol. 98, 47 [full issue]. *Weekly Epidemiological Record = Relevé épidémiologique hebdomadaire*, 98(47):599 – 620, 2023-11-24.

World Health Organization. Disease outbreak news: Item 2024-don526, 2024a. URL https://www.who.int/emergencies/disease-outbreak-news/item/2024-DON526. Accessed: 2024-09-30.

World Health Organization. Global dengue burden, 2024b. URL https://worldhealthorg.shinyapps.io/dengue_global/. Accessed: 2024-09-30.

Yin, R., Luo, Z., and Kwoh, C. K. Alignment-free machine learning approaches for the lethality prediction of potential novel human-adapted coronavirus using genomic nucleotide. *Biorxiv*, pages 2020–07, 2020.

Appendix A. Appendices

A.1 Experimental Environment

In the following tables, we present the characteristics of Virtual Machines used with Colab. The Biopython library (version 1.84) was used to read and parse the sequences. Scikit-learn (version 1.4.2) assisted with preprocessing and machine learning models, while TensorFlow-Keras (version 3.4.1) was employed for all deep learning models. Imbalanced-learn (version 0.12.3) was utilized for SMOTE, and Optuna (version 4.0.0) was used for hyperparameter tuning, all in Python (version 3.10.12). Additionally, all deep learning model diagrams were plotted easily in Netron.

Table A.1: Characteristics of Virtual Machines used.

Preprocessing and Models Training

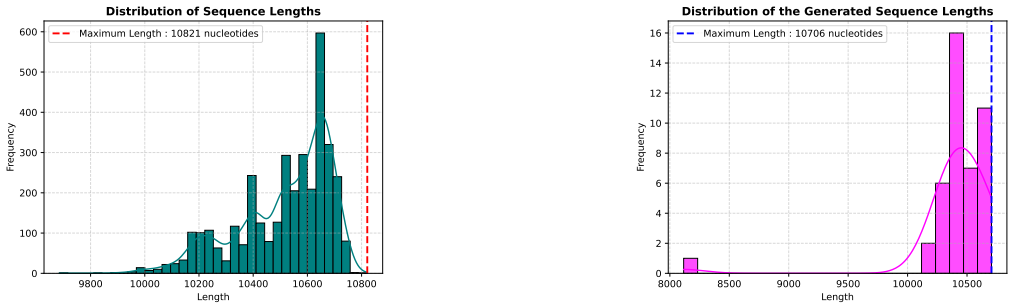
Characteristic	Details
Machine Type	n1-highmem-16
CPU Platform	Intel Haswell
vCPUs	16
RAM	102.2 GB
Disk Size (SSD)	200 GB
GPUs	2 x NVIDIA T4
GPU RAM	30 GB

Hyperparameters Tuning

Characteristic	Details
Machine Type	n1-highmem-96
CPU Platform	Intel Skylake
vCPUs	96
RAM	614.1 GB
Disk Size (SSD)	200 GB
GPUs	None

A.2 Distribution of Lengths Across Sequences

Figure A.1 illustrates the distributions of sequence lengths for both Original and Synthetic sequences sampled from the decoder of the Variational Autoencoder algorithm.



(a) Original sequence lengths

(b) Generated sequence lengths

Figure A.1: Distribution of sequence lengths: Subfigure A.1a shows the distribution of original sequence lengths, while subfigure A.8b displays the distribution of sequences generated by the VAE.

A.3 Machine Learning Classifiers Best Hyperparameters

In the following subsections, for each of the two feature extraction methods (K-mer encoding and FCGR), we present the best hyperparameters obtained for each model (Random Forest, XGBoost, and LightGBM). In addition, we identify optimal hyperparameters that achieve the highest performance within a reasonable time frame. It should be noted that each model was initially trained **10** times (`n_trials = 10`), and we report only the results from the run where the best parameters were achieved. Full details regarding these results can be accessed via this link.

A.3.1 k-mers encoding Feature extraction technique

1. **Random Forest**

Table A.2: Random Forest Hyperparameters for k-mers Technique.

Hyperparameter	**3-mers**	**4-mers**	**5-mers**	**6-mers**	**7-mers**	**8-mers**
`n_estimators`	256	1040	234	203	182	65
`max_depth`	12	4	19	11	18	13
`min_samples_split`	9	3	17	5	19	7
`min_samples_leaf`	2	20	16	6	18	20
`bootstrap`	`True`	`True`	`True`	`False`	`False`	`True`
`criterion`	`entropy`	`gini`	`gini`	`entropy`	`entropy`	`gini`
Score	**0.9943**	**0.9986**	**1.000**	**0.9986**	**0.9986**	**0.9986**

2. **XGBoost**

Table A.3: XGBoost Hyperparameters for k-mers Technique.

Hyperparameter	**3-mers**	**4-mers**	**5-mers**	**6-mers**	**7-mers**	**8-mers**
`max_depth`	5	6	14	16	15	13
`learning_rate`	0.0459	0.0961	0.0041	0.0348	0.0863	0.0043
`n_estimators`	398	462	345	541	314	331
`min_child_weight`	1	1	1	8	2	5
`subsample`	0.9523	0.7573	0.9751	0.5534	0.8616	0.8500
`colsample_bytree`	0.6328	0.8649	0.8915	0.7548	0.8952	0.9250
Score	**0.9915**	**0.9986**	**1.000**	**1.0000**	**0.9986**	**0.9986**

3. **LightGBM**

Table A.4: LightGBM Hyperparameters for k-mers Technique.

Hyperparameter	**3-mers**	**4-mers**	**5-mers**	**6-mers**	**7-mers**	**8-mers**
learning_rate	0.0539	0.0106	0.0340	0.0188	0.0014	0.0072
n_estimators	558	1083	458	193	843	323
max_depth	20	1	2	5	16	13
num_leaves	33	49	247	137	21	154
feature_fraction	0.5496	0.6903	0.6217	0.8329	0.6411	0.5597
bagging_fraction	0.9165	0.7017	0.8951	0.7415	0.5194	0.5814
bagging_freq	7	4	3	3	1	4
min_child_samples	57	89	8	76	98	77
lambda_l1	3.4677e-05	1.9044e-07	7.5168e-06	0.0002	5.1027e-06	0.0114
lambda_l2	0.0406	4.5062e-06	0.0039	4.1343e-07	0.0002	0.0046
Score	**0.9958**	**0.9986**	**1.000**	**1.0000**	**0.9986**	**0.9986**

Summary k-mer : Mean Score and Timing

Table A.5: Variation of Sequence shapes accross k-mers values.

k-mer value	**3**	**4**	**5**	**6**	**7**	**8**
Sequence shape	1×64	1×256	1×1024	1×4096	1×16384	1×65536

Table A.6: Mean Score and Time Spent for k-mers Technique: In the following table, we first compute the mean score across the three classifiers for each k-mer size and associate the recorded time taken to find the best parameters for all three classifiers. We observed that 5-mers is the optimal value, achieving a perfect mean score (1.00) in a reasonable amount of time.

Model	**3-mers**	**4-mers**	**5-mers**	**6-mers**	**7-mers**	**8-mers**
RF	**0.9943**	**0.9986**	**1.000**	**0.9986**	**0.9986**	**0.9986**
XGBoost	**0.9915**	**0.9986**	**1.000**	**1.0000**	**0.9986**	**0.9986**
LightGBM	**0.9958**	**0.9986**	**1.000**	**1.0000**	**0.9986**	**0.9986**
Mean Score	**0.9938**	**0.9986**	**1.000**	**0.9995**	**0.9986**	**0.9986**
Timing(s)	**1377.01**	**1486.16**	**2367.20**	**6117.57**	**25690.19**	**84460.54**

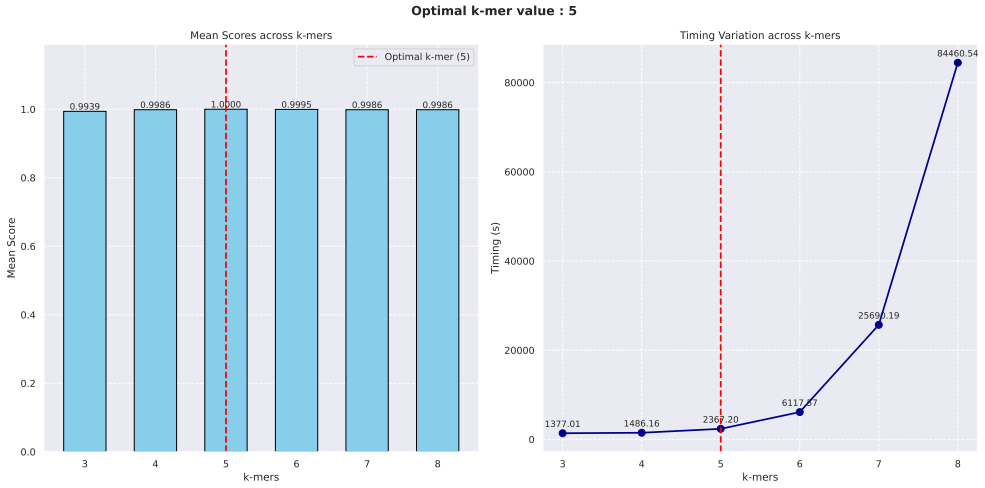


Figure A.2: The optimal k-mer value maximises the mean score in a reasonable timeframe.

While training the models with 10-Repeated Stratified Cross-Validation, we obtained the following results:

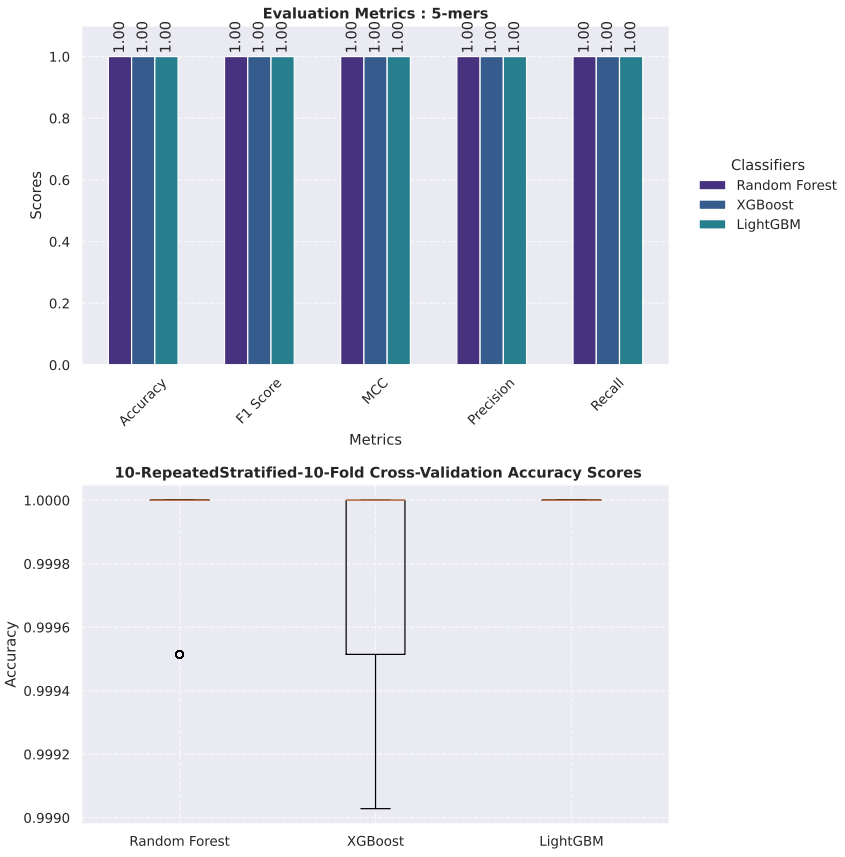


Figure A.3: 10-Repeated Stratified-10-Fold Cross-Validation: Perfect Scores.

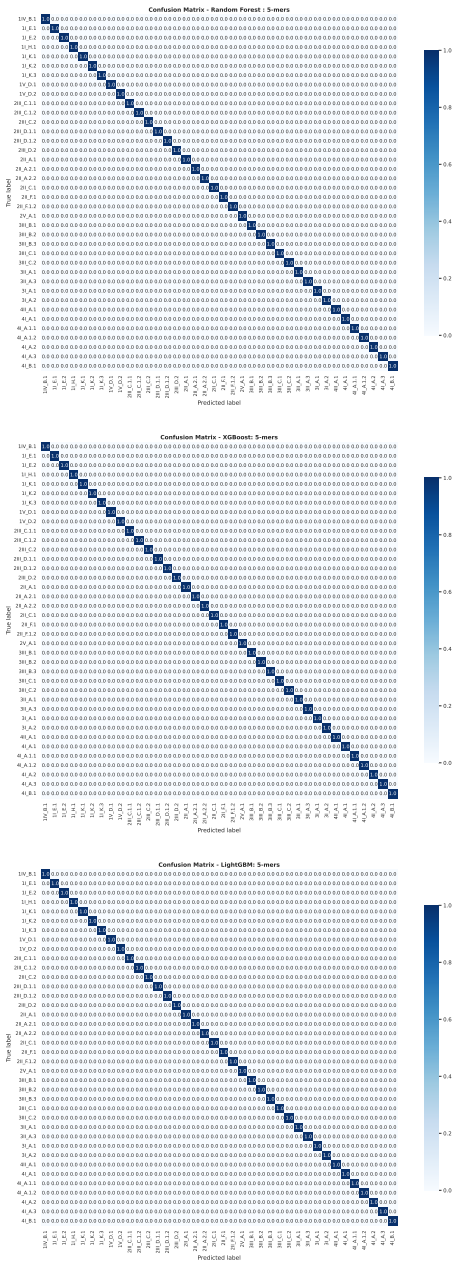


Figure A.4: Confusion Matrices using 5-mers Feature Extraction Technique: Perfect Classification.

A.3.2 FCGR Feature Extraction Technique

1. **Random Forest**

Table A.7: Random Forest Hyperparameters for FCGR Technique.

Hyperparameter	**16-FCGR**	**32-FCGR**	**64-FCGR**	**128-FCGR**
`n_estimators`	250	77	230	150
`max_depth`	9	12	12	19
`min_samples_split`	2	15	8	12
`min_samples_leaf`	4	5	6	3
`bootstrap`	`False`	`True`	`True`	`True`
`criterion`	`gini`	`entropy`	`entropy`	`entropy`
Score	**0.9972**	**1.0000**	**0.9986**	**0.9986**

2. **XGBoost**

Table A.8: Optimal XGBoost Hyperparameters for FCGR Technique.

Hyperparameter	**16-FCGR**	**32-FCGR**	**64-FCGR**	**128-FCGR**
`max_depth`	5	15	11	13
`learning_rate`	0.0575	0.0271	0.0019	0.0068
`n_estimators`	234	253	384	279
`min_child_weight`	9	9	8	7
`subsample`	0.7184	0.6411	0.9125	0.7302
`colsample_bytree`	0.6846	0.5212	0.5414	0.8981
Score	**0.9986**	**1.0000**	**1.000**	**0.9958**

3. **LightGBM**

Table A.9: LightGBM Hyperparameters for FCGR Technique.

Hyperparameter	**16-FCGR**	**32-FCGR**	**64-FCGR**	**128-FCGR**
learning_rate	0.0072	0.0870	0.0680	0.0797
n_estimators	1050	836	83	414
max_depth	3	1	3	10
num_leaves	250	26	3	3
feature_fraction	0.7222	0.8267	0.7840	0.5777
bagging_fraction	0.7114	0.5361	0.9634	0.6894
bagging_freq	4	1	2	1
min_child_samples	81	25	64	64
lambda_l1	3.4995e-06	0.0008	1.1450e-07	6.6207e-06
lambda_l2	1.3985	1.0356e-06	0.0004	9.7719e-05
Score	**0.9986**	**1.0000**	**1.0000**	**0.9972**

Summary FCGR : Mean Score and Timing

Table A.10: Variation of Sequence shapes accross FCGR resolutions.

Resolution-FCGR	**16**	**32**	**64**	**128**
Sequence shape	1×256	1×1024	1×4096	1×16384

Table A.11: Mean Score and Time Spent for FCGR Technique: In the following table, we first compute the mean score across the three classifiers for each Resolution-FCGR and associate the recorded time taken to find the best parameters for all three classifiers. We observed that 32-FCGR is the optimal value, achieving a perfect mean score (1.00) in a reasonable amount of time.

Model	**16-FCGR**	**32-FCGR**	**64-FCGR**	**128-FCGR**
RF	**0.9972**	**1.0000**	**0.9986**	**0.9986**
XGBoost	**0.9986**	**1.0000**	**1.000**	**0.9958**
LightGBM	**0.9986**	**1.0000**	**1.0000**	**0.9972**
Mean Score	**0.9981**	**1.000**	**0.9995**	**0.9972**
Timing(s)	**1280.82**	**2263.35**	**7391.19**	**32021.16**

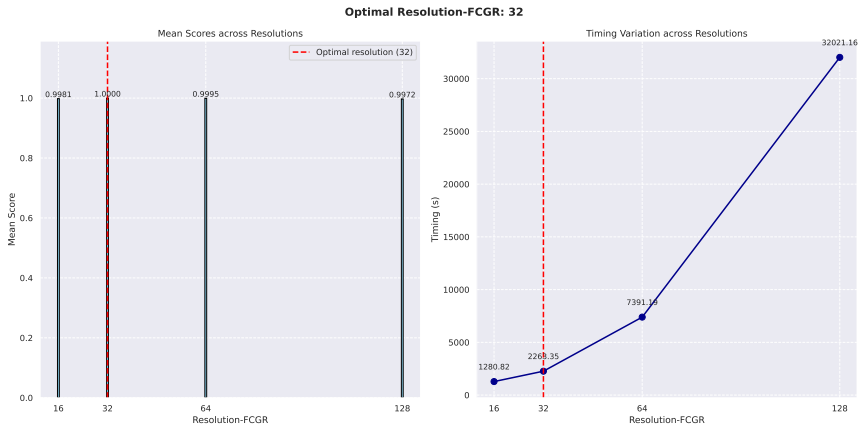


Figure A.5: The optimal FCGR value maximises the mean score in a reasonable timeframe.

While training the models with 10-Repeated Stratified Cross-Validation, we obtained the following results:

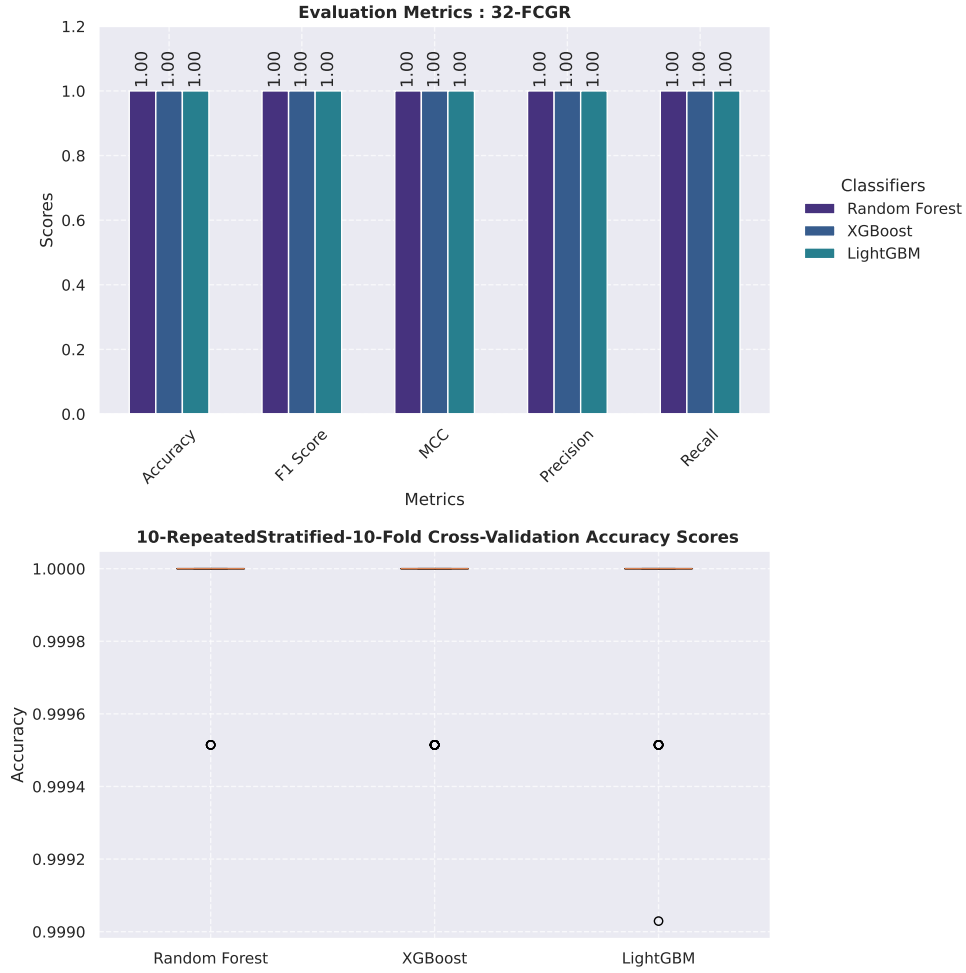


Figure A.6: 10-RepeatedStratified-10-Fold Cross-Validation Results: Perfect Scores.

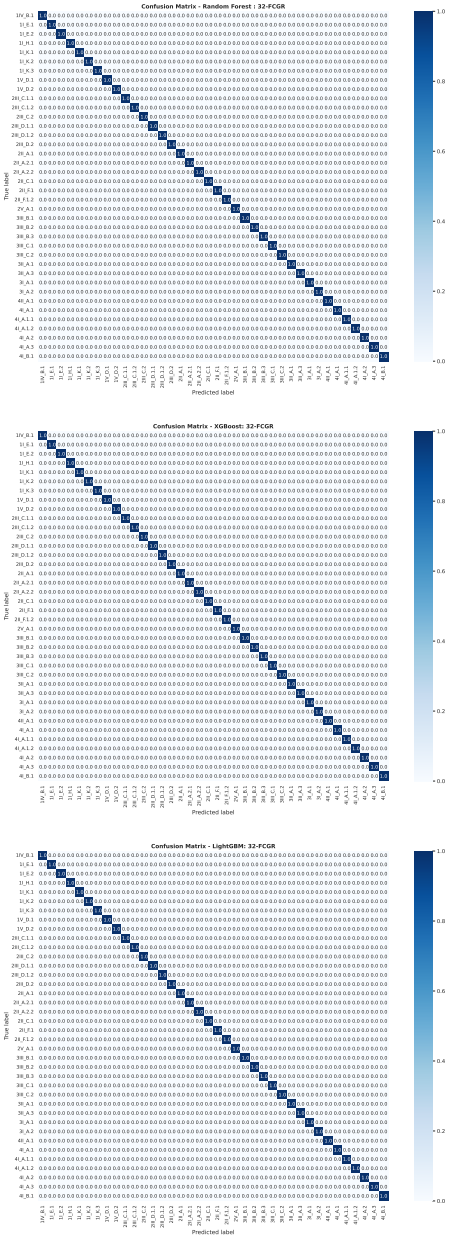
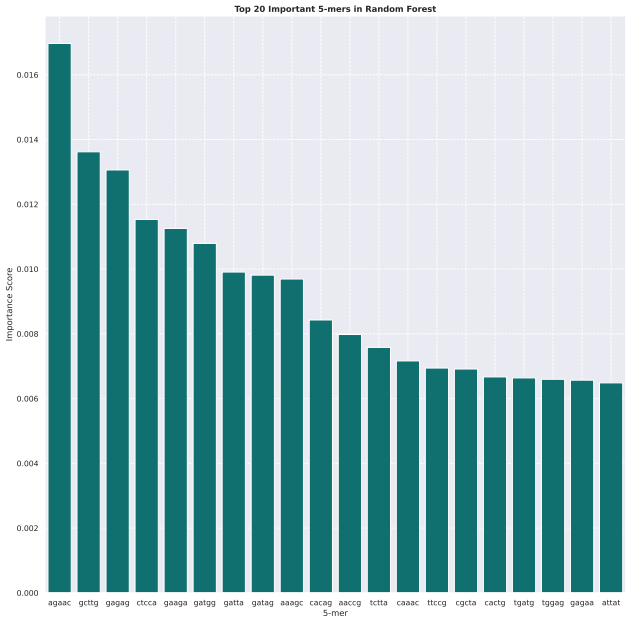
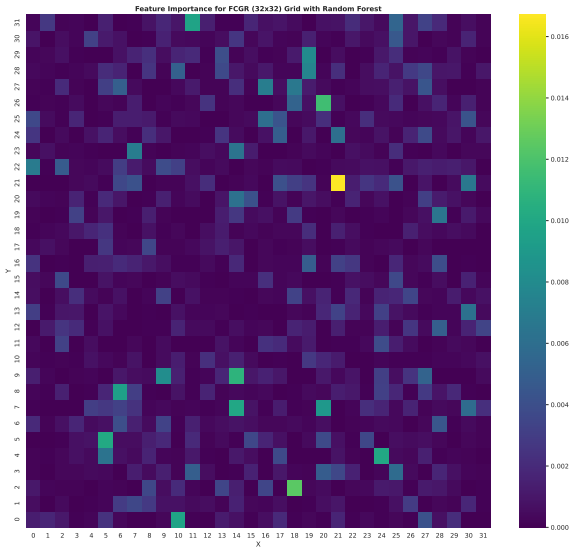


Figure A.7: Confusion Matrices using 32-FCGR Feature Extraction Technique: Perfect Classification.

A.4 Feature Importance in the Random Forest Model



(a) 5-mers Encoding Feature Extraction Technique: "AGAAC" is the most informative 5-mer.



(b) 32-FCGR Feature Extraction Technique: The yellow grid represents the most informative pixel.

Figure A.8: Feature Importance: These are the most relevant elements which help the Random Forest model to perfom the classification Task.

A.5 Additional Details on Some Deep Learning Models

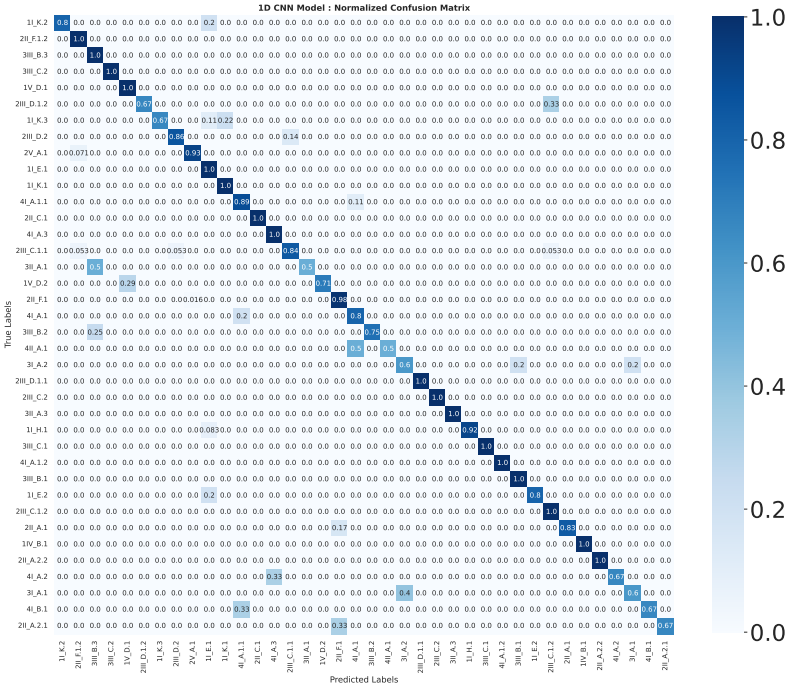


Figure A.9: Confusion Matrix for 1D CNN with Self-Attention: Some misclassifications are observed.

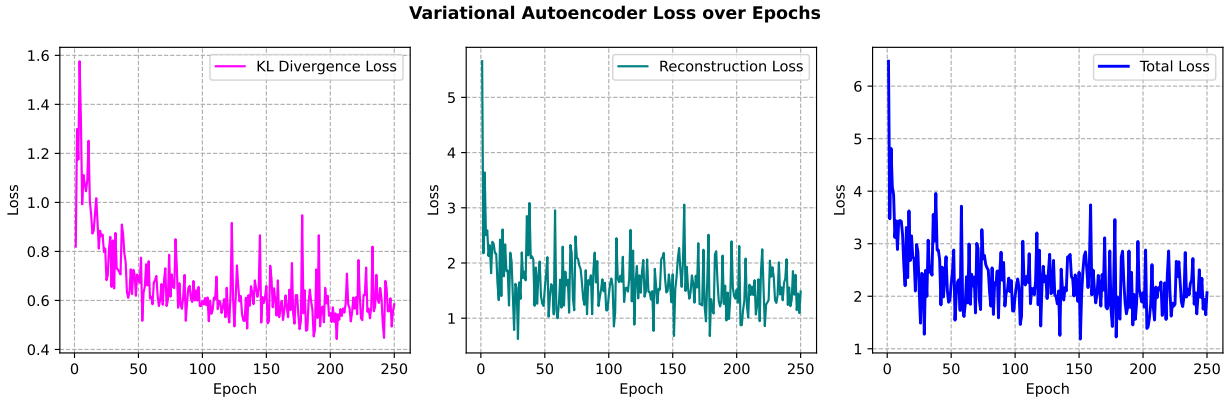


Figure A.10: VAE training history: KL Divergence, Reconstruction, and Total Loss decrease over epochs, indicating improved latent space learning and sequence reconstruction.