Pre-processing and analysis of genomic data using data mining tools

Théo FIGINI
M2 Computer Science
Academic year 2024-2025

Hosting organization : *Université des Antilles*

January, 13th, 2025–June, 16th, 2025

Referring teacher:
Emmanuel BIABIANY

Tutor :
Wilfried SEGRETIER

June, 16th 2025

Acknowledgements

I would like to express my heartfelt gratitude to all those who have supported me throughout this project. First and foremost, I would like to thank my tutor, Dr. **Wilfried SEGRETIER**, for his invaluable guidance and support during my internship.

I also extend my thanks to Dr. **Erick STATTNER**, Dr. **Stephane CHOLET**, and Dr. **David COUVIN** for their assistance and for providing me with the additional resources I needed to complete this project.

I would also like to thank my fellow students for their collaboration and support during this project.

Finally, I would like to express my gratitude to my family and friends for their unwavering support and encouragement throughout this journey.

Contents

Résumé

La tuberculose (TB), causée par Mycobacterium tuberculosis, reste l'une des maladies infectieuses les plus meurtrières dans le monde. Comprendre les différentes souches est essentiel pour améliorer les traitements et les stratégies de lutte. Ce projet vise à identifier les souches de TB à partir de données de séquençage du génome complet. Nous avons exploré des méthodes de regroupement non supervisées et développé des modèles de classification à l'aide de réseaux de neurones convolutifs (CNN). Des techniques comme les autoencodeurs et l'augmentation de données ont été utilisées pour améliorer les performances. Bien que les résultats soient encore préliminaires, ils montrent un potentiel pour différencier les souches de TB selon leurs profils génomiques, ouvrant la voie à des outils de diagnostic plus précis.

Abstract

Tuberculosis (TB), caused by Mycobacterium tuberculosis, remains one of the world's most deadly infectious diseases. Understanding its different strain types is vital for improving treatment and control efforts. This project focuses on identifying TB strains using whole genome sequencing data. We explored unsupervised clustering methods and developed classification models using convolutional neural networks (CNNs). Techniques such as autoencoders and data augmentation were also applied to enhance performance. While the results are preliminary, they demonstrate potential for differentiating TB strains based on genomic patterns, laying the groundwork for more refined diagnostic tools.

1 Introduction

Tuberculosis (TB) is a highly infectious disease caused by the bacterium *Mycobacterium Tuberculosis*. It primarily affects the lungs but can also impact other parts of the body. TB is a major global health concern, with millions of new cases and deaths reported annually. The disease is transmitted through airborne droplets when an infected person coughs or sneezes.

The emergence of multidrug-resistant (MDR) and extensively drug-resistant (XDR) strains of *M. tuberculosis* poses a significant challenge to TB control efforts. These strains are resistant to the most effective anti-TB drugs, making treatment more difficult and increasing the risk of transmission. The World Health Organization (WHO) has identified the need for new diagnostic tools, treatments, and vaccines to combat TB, particularly in the context of drug resistance.

This project aims to find a way to identify and analyze the strain of *M. tuberculosis* responsible for a TB infection using genomic data. By analyzing the genetic sequences of *M. tuberculosis* strains, we hope to develop a method that can accurately determine the strain type, which is crucial for effective treatment and control of the disease. We explored different approaches to achieve this goal.

We first use multiple clustering techniques to group similar strains based on the presence or absence of certain proteins. Then, we converted the genomic data into images of various resolutions and used a convolutional neural network (CNN) to classify the images. Our goal is to find new ways to identify and analyze *M. tuberculosis* strains using genomic data, which could lead to improved diagnostic tools and treatment strategies for TB.

2 Related Work

Genomics is a field that has been growing rapidly in the past few years. The advent of high-throughput sequencing technologies has made it possible to sequence the entire genome of an organism in a matter of days. This has led to an explosion of data, with the number of sequenced genomes increasing exponentially. This has created a need for new tools and algorithms to analyze this data. In this chapter, we review some of the existing tools and algorithms for analyzing genomic data. We also discuss some of the techniques we used in our project.

2.1 Genomic data analysis

Random forests

Another method that has been used for genomic data analysis is random forests (RF) [5]. This method is based on the idea of ensemble learning, where multiple decision trees are trained on different subsets of the data and then combined to make a final prediction.

AI applications in genomic analysis

Many researchers have used AI techniques to analyze genomic data. For example, [4] reviews different AI techniques that have been used for genomic data analysis, including CNNs, autoencoders, etc.

More recently, the authors of [18] designed a tool based on multiple LLM backends for multi-omics analysis with minimal human intervention.

2.2 Damerau-Levenshtein distance

The **Damerau-Levenshtein distance** is a string metric for measuring the edit distance between two sequences. This distance is named after Frederick J. Damerau and Vladimir I. Levenshtein, who independently introduced it in the 1960s [11]. It is a variation of the Levenshtein distance that allows for the transposition of two adjacent characters in addition to the standard operations of insertion, deletion, and substitution. The distance is defined as the minimum number of edit operations required to transform one sequence into another. The operations are:

- **Insertion**: Adding a character to the sequence.
- **Deletion**: Removing a character from the sequence.
- **Substitution**: Replacing one character with another.
- **Transposition**: Swapping two adjacent characters.

We tried to integrate the Damerau-Levenshtein distance in our clustering attempts, but did not manage to do so successfully. The main issues were time and resource constraints, as the algorithm is computationally expensive, especially for large datasets.

2.3 DBSCAN and HDBSCAN

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu in 1996 [7]. It works by defining a neighborhood around each point based on a distance metric (usually Euclidean distance) and a minimum number of points required to form a dense region. The algorithm proceeds as follows:

1. For each point in the dataset, find its neighbors within a specified radius ε.
2. If the number of neighbors is greater than or equal to the minimum number of points, a new cluster is formed.
3. The algorithm recursively expands the cluster by including all points that are reachable from the initial point.
4. Points that do not belong to any cluster are classified as noise.

HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is an extension of DBSCAN proposed by Ricardo J. G. Campello, David Moulavi, and Jörg Sander in 2013 [2], then further developed with Arthur Zimek in 2015 [3]. It builds on the principles of DBSCAN by introducing a hierarchical clustering approach. It is more robust to varying densities and can identify clusters of different shapes and sizes. The algorithm works as follows:

1. Compute the mutual reachability distance between points, which takes into account the distance to the nearest neighbor.
2. Build a minimum spanning tree (MST) based on the mutual reachability distances.
3. Extract clusters from the MST by varying the density threshold, resulting in a hierarchy of clusters.
4. Assign points to clusters based on their membership in the hierarchy, allowing for flexible clustering.

2.4 Introduction to CNNs

The fundamental idea behind **Convolutional Neural Networks (CNNs)** was introduced by Kunihiko Fukushima [9] in 1980 and later popularized by Yann LeCun in the 1990s with the LeNet architecture [10]. CNNs are a class of deep learning models specifically designed for processing structured grid data, such as images. They are particularly effective for tasks like image classification, object detection, and segmentation.

CNNs work by applying convolutional filters to the input data, which allows them to learn spatial hierarchies of features. It shares some similarities with the more conventional neural networks, but there are at least two types of layers that are specific to CNNs:

- **Convolutional layers**: These layers apply convolutional filters to the input data, extracting local features that are invariant to translation. The filters slide over the input data, computing dot products and producing feature maps.
- **Pooling layers**: These layers downsample the feature maps, reducing their spatial dimensions while retaining important information. Common pooling operations include max pooling and average pooling.

The architecture of a CNN typically consists of multiple convolutional and pooling layers followed by fully connected layers. The convolutional layers learn to extract features from the input data, while the fully connected layers perform the final classification or regression task. The training process involves optimizing the weights of the filters and fully connected layers using backpropagation and a loss function, such as cross-entropy for classification tasks.

2.5 K-fold cross validation

K-fold cross validation is a specific type of cross validation where the dataset is divided into k equally sized folds. The model is trained on $k-1$ folds and tested on the remaining fold, and this process is repeated for each fold. The final performance is averaged over all folds to obtain a more reliable estimate of the model's performance. The graphical representation of K-fold cross validation is shown in Figure 1.

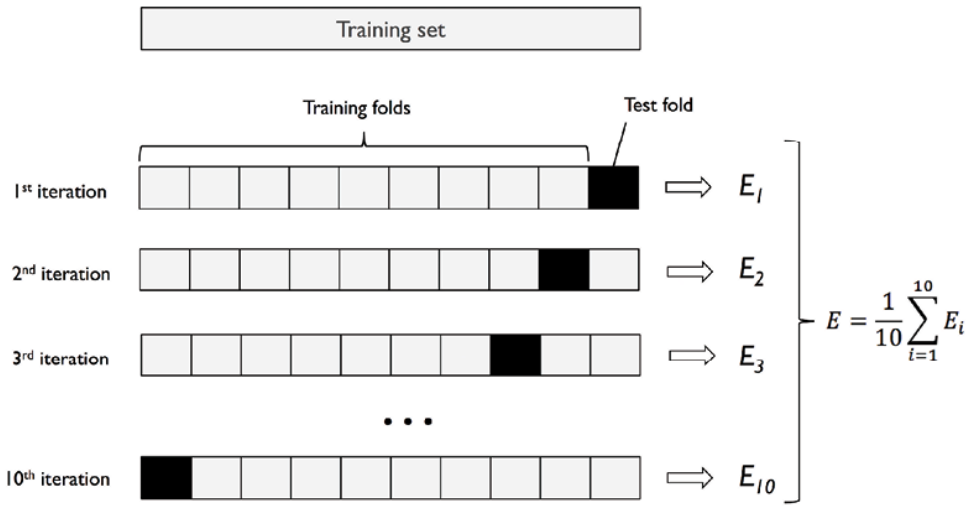

Figure 1: How K-fold cross validation works. [15]

Stratified K-fold cross validation

Stratified K-fold cross validation is a variation of K-fold cross validation that ensures that each fold has the same proportion of samples from each class. This is particularly useful for imbalanced datasets, as it ensures that each fold has a representative sample of each class. Although if this method is recommended for imbalanced datasets, we did not have enough time to implement it, so we used the standard K-fold cross validation.

2.6 SMOTE

Data augmentation is a technique used to artificially increase the size of a dataset by applying various transformations to the existing data. This can help improve the model's performance by providing more diverse training samples and reducing overfitting. In our project, we used SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples for the minority classes in our dataset.

SMOTE works by creating new samples by interpolating between existing samples in the feature space. It generates synthetic samples by selecting a minority class sample, finding its nearest neighbors, and creating new samples by interpolating between the selected sample and its neighbors. This helps to balance the dataset by increasing the number of samples in the minority classes, which can improve the model's performance on those classes.

New samples are generated by taking a random sample from the minority class and finding its nearest neighbors in the feature space. The new sample is then created by interpolating between the selected sample and its neighbors, as shown in Figure 2. It uses the following formula:

$$x_{new} = x_i + \lambda(x_j - x_i) \tag{1}$$

Where x_{new} is the new sample, x_i is the selected sample, x_j is one of its nearest neighbors, and λ is a random number between 0 and 1. This process is repeated until the desired number of samples is generated for the minority class.

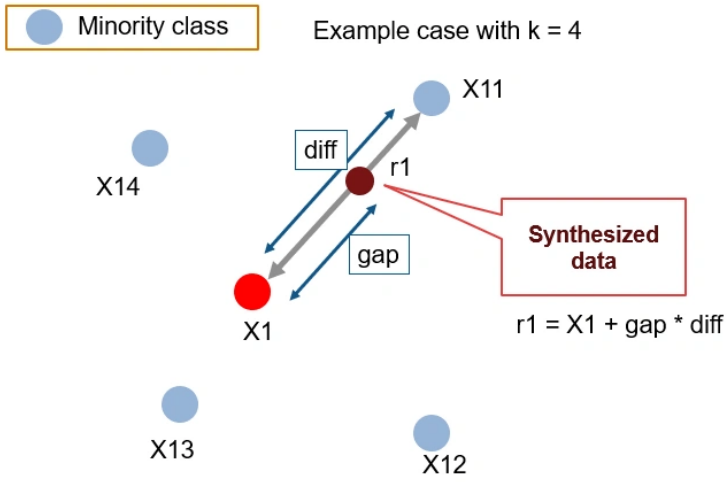

Figure 2: Illustration of the SMOTE algorithm. [16]

3 Context

3.1 Laboratoire de Mathématiques Informatique et Applications (LAMIA)

This project was done within the framework of the Master 2 Computer Science program at the **Laboratoire de Mathématiques Informatique et Applications (LAMIA)** of the **Université des Antilles**. The LAMIA is a research laboratory that is part of the Université des Antilles and is located in Guadeloupe. It focuses on the intersection of mathematics, computer science, and their applications in various fields, including data science, machine learning, and artificial intelligence. LAMIA conducts research that aims to advance theoretical understanding and practical applications of mathematical and computational methods.

3.2 Work Environment

The following tools and technologies were used to develop algorithms and document the project:

- **Python**: a programming language widely used in scientific computing, data analysis, and machine learning.
- **GitHub**: a version control platform for managing the source code.
- **GitHub Copilot**: for code completion.
- **Jupyter Notebook**: an interactive environment for writing and executing Python code, ideal for data analysis and visualization.
- **Sklearn**: a Python library for machine learning that provides various algorithms and tools for data preprocessing, model selection, and evaluation.
- **Pandas**: a Python library for data manipulation and analysis, providing data structures and functions for working with structured data.
- **Matplotlib**: a Python library for graphics and data visualization, including 3D plotting capabilities.
- LaTeX: a typesetting system used for writing technical and scientific documents.
- A personal computer running Windows 11 with a **Ryzen 7 6800H** processor and **32 GB** of RAM.

This internship was conducted within the office of the LAMIA with other students, which allowed for collaboration and sharing of ideas and resources.

3.3 Objectives

The main objective of this project is to explore and analyze genomic data, specifically the genomes of Tuberculosis (TB) strains, to identify patterns and relationships between different strains. As well as to apply various data mining and data analysis techniques, but also develop new methods and compare their performance to existing methods.

4 Genomic data

Genomic data refers to the information contained in the DNA of an organism, which includes the sequences of nucleotides (Adenine, Thymine, Cytosine, and Guanine). These are gathered through various methods, such as DNA sequencing, and can be used to study the genetic makeup of organisms, identify genetic variations, and understand the relationships between different species. The Institute Pasteur has a large collection of genomic data, including the genomes of various strains of Tuberculosis (TB), which is a major public health concern worldwide.

4.1 Institut Pasteur

The Institut Pasteur is a renowned research institute located in Paris, France, dedicated to the study of biology, microorganisms, diseases, and vaccines. Founded in 1887 by Louis Pasteur, the institute has played a pivotal role in the development of modern microbiology and immunology. It is known for its contributions to the understanding of infectious diseases and the development of vaccines, including those for rabies and tuberculosis. Founded by Louis Pasteur in 1887, the Institut Pasteur has been at the forefront of scientific research and innovation for over a century.

4.2 The FASTA format

The FASTA format is a text-based format for representing nucleotide or protein sequences [8, 17]. It is widely used in bioinformatics to store and exchange sequence data. A FASTA file consists of one or more sequences, each represented by a header line starting with a greater-than symbol (>), followed by the sequence itself on the next line. The sequence codes for a protein who then codes for a gene. The header line can contain additional information about the sequence, such as its identifier, description, or source. An example of a FASTA file is shown in Figure 3.

```
>sequence_id
ATCGATCGATCGATCGATCGATCG
>sequence_id_2
GCTAGCTAGCTAGCTAGCTAGCTAGCTA
>sequence_id_3
TAGCTAGCTAGCTAGCTAGCTAGCTAGC
>sequence_id_4
AGCTAGCTAGCTAGCTAGCTAGCTAGCT
>sequence_id_5
CATCGATCGATCGATCGATCGATCGATC
```

Figure 3: Example of a FASTA file containing two sequences.

The FASTA format is simple and easy to read, making it suitable for storing large amounts of sequence data. It is also compatible with many bioinformatics tools and software, allowing for easy analysis and manipulation of sequence data.

5 Clustering genomic data

5.1 Preprocessing Genomic Data

Our data is a collection of Tuberculosis (TB) genomes, where each file contains the proteins and their respective nucleotide sequences in FASTA format, for a total of 7057 files. These genomes were obtained from the **Institut Pasteur** and are part of the **Mycobacterium tuberculosis complex** (MTBC), which is a group of closely related bacteria that cause TB. The goal is to cluster these genomes based on their protein sequences and find ways to distinguish between different TB strains.

We started by creating a dataset containing the number of times each protein appears in each genome. This was done by parsing the FASTA files and counting the occurrences of each protein sequence, we then removed the outliers, namely "hypothetical proteins" and "putative proteins", which are not well characterized and do not provide useful information for clustering. The resulting dataset is a CSV file of 7057 columns plus 1 column for the protein names and 2732 rows plus 1 row for the header.

Considering the size of the dataset, we used Principal Component Analysis (PCA) set to 95% variance to reduce the dimensionality of the data while preserving as much variance as possible. PCA is a technique that transforms the data into a new coordinate system, where the first principal component captures the maximum variance, the second captures the second maximum variance, and so on.

We then calculated the silhouette score for different values of k, ranging from 2 to 10, to determine the optimal number of clusters. The silhouette score was calculated using the ‘silhouette_score‘ function from the ‘sklearn.metrics‘ module, which takes the data and the cluster labels as input. The optimal number of clusters is the one that maximizes the silhouette score.

5.1.1 First Results

The initial results showed that the silhouette score was highest for $k = 9$, indicating that this was the optimal number of clusters. However, the clusters were not well separated, or some points are isolated in their cluster, as seen in Figure 4, which suggests that the data may not be well suited for clustering with K-means, or that the features used for clustering are not informative enough. We also tried without applying PCA but achieved similar results. We also observed fewer clusters, as seen in Figure 6.

After reviewing the data, we found that there is significant overlap between the points, so we switched to 3D visualization to better understand the clusters. With Figure 5 and Figure 7, we can see that the clusters are still not well separated, but we can conclude that a 2D space was not appropriate for visualizing the clusters.

These results suggest that K-means clustering may not be the best approach for this dataset, or that the data may need further or a different preprocessing step to improve the clustering results. We will explore alternative clustering algorithms in the next section.

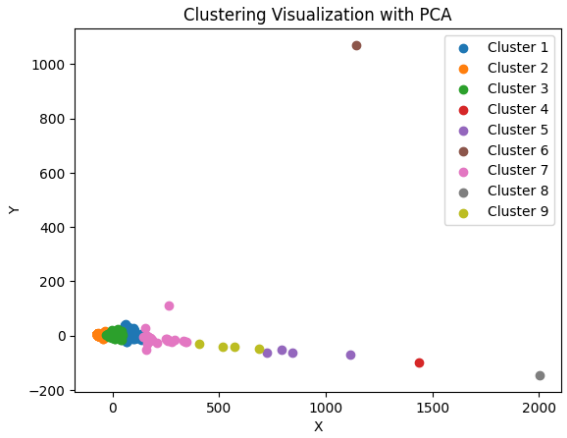

Figure 4: Clusters obtained with K-means clustering after applying PCA.

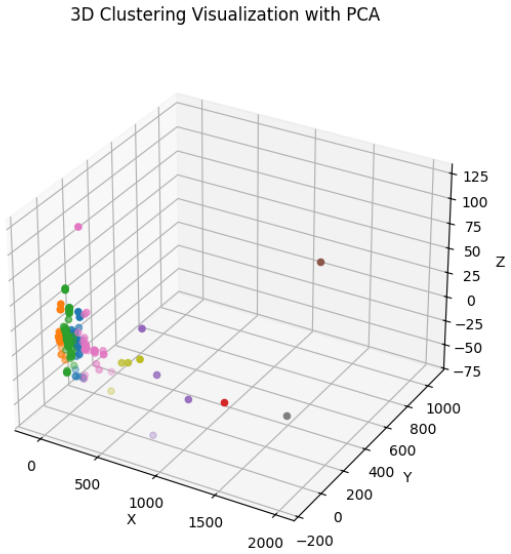

Figure 5: Clusters obtained with K-means clustering after applying PCA in 3D.

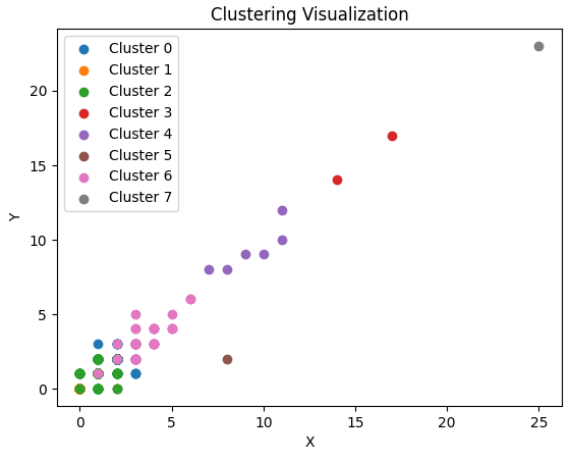

Figure 6: Clusters obtained with K-means clustering without applying PCA.

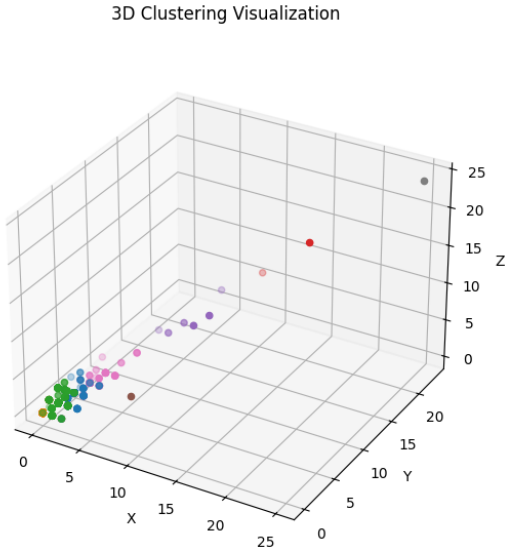

Figure 7: Clusters obtained with K-means clustering without applying PCA in 3D.

5.2 New Approach using DBSCAN and HDBSCAN

To address the poor clustering results obtained with K-means, we explored alternative clustering algorithms, specifically **DBSCAN** and **HDBSCAN**.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups points that are close together based on a distance measurement and a minimum number of points. It is particularly effective for datasets with varying densities and can identify noise points that do not belong to any cluster.

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is an extension of DBSCAN that builds a hierarchy of clusters and allows for more flexible clustering by varying the density threshold. It can also handle varying cluster shapes and sizes.

5.2.1 Results

During our testing, we applied both DBSCAN and HDBSCAN with an epsilon value of 1 and a minimum number of samples of 5. These techniques were both applied to the dataset with and without PCA. We can see from Figure 20 to Figure 11 that the clusters obtained with DBSCAN and HDBSCAN are more just a bit more discernible than those obtained with K-means clustering. But the clusters are still closely packed together, we can conclude that the problem lies with the way the dataset was created. A similar observation can be made when not applying PCA, as shown in Figure 20 to Figure 23 in the appendix.

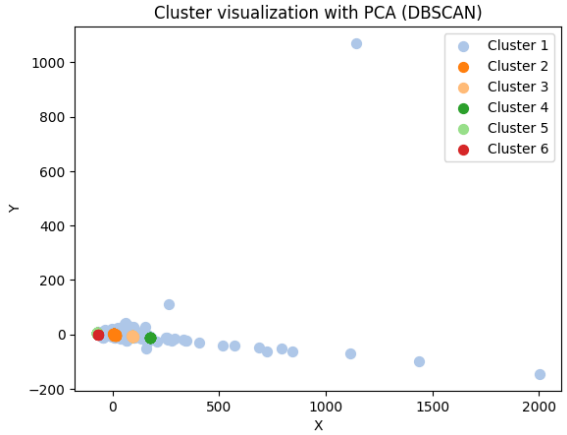

Figure 8: Clusters obtained with DBSCAN after applying PCA.

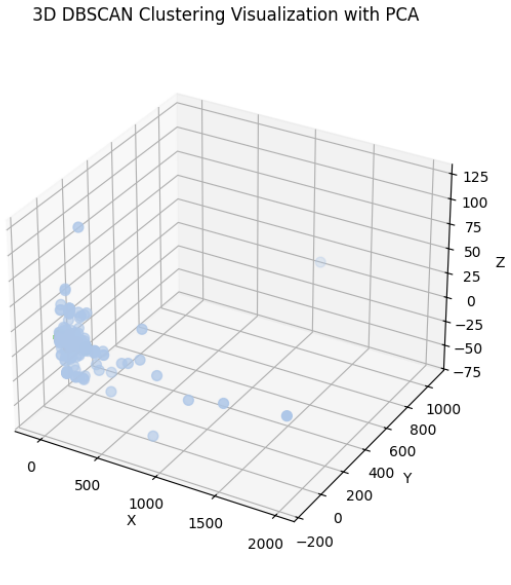

Figure 9: Clusters obtained with DBSCAN after applying PCA in 3D.

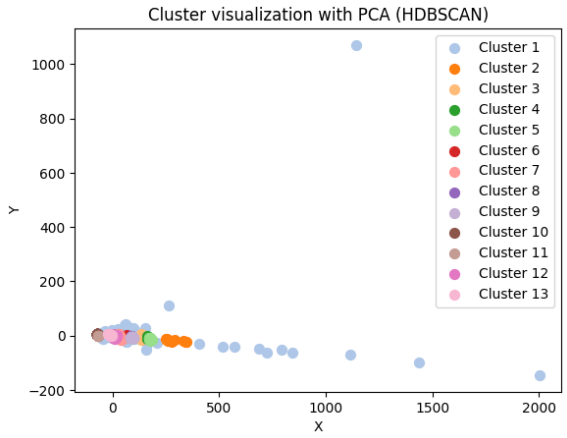

Figure 10: Clusters obtained with DBSCAN after applying PCA.

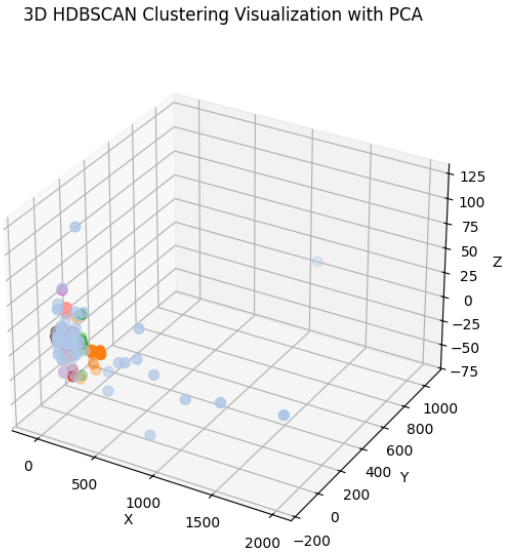

Figure 11: Clusters obtained with DBSCAN after applying PCA in 3D.

5.3 Trying an autoencoder

To further explore the clustering of the genomic data, we decided to try an autoencoder. After getting the embeddings from the autoencoder, we applied the K-means clustering algorithm to the encoded data. We can see in Figure 12 and Figure 13 that the clusters obtained with K-means clustering on the encoded data are more discernible than those obtained with K-means clustering on the original data. However, we observe a great intra-cluster distance, which suggests that the autoencoder may not have learned a good data representation, or that the dataset or features were not well suited for clustering.

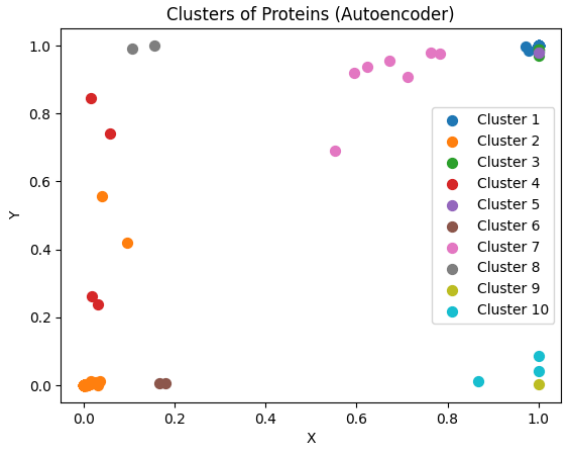

Figure 12: Clusters obtained with K-means clustering on the encoded data.

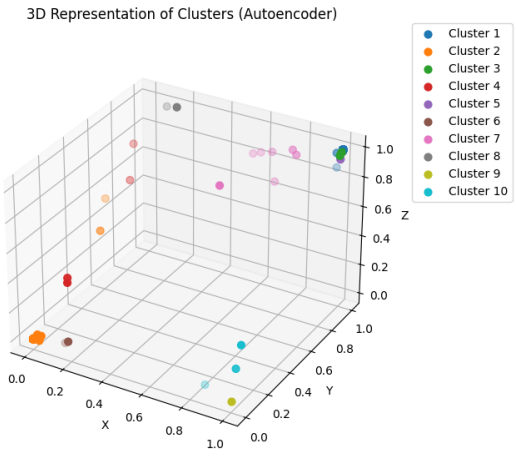

Figure 13: Clusters obtained with K-means clustering on the encoded data in 3D.

6 Using CNNs to identify TB Strains

The clustering approach did not yield satisfactory results, so we decided to explore a different approach by converting the genomic data into images. This approach allows us to leverage the power of Convolutional Neural Networks (CNNs) to classify the images and identify the TB strains. These images are generated from the genomic data by applying five metrics to the nucleotide sequences, which are then converted into pixel values. The pixel value is determined by a combination of three metrics, resulting in ten different combinations of metrics, each represented by a different color channel in the image (Red, Green and Blue, RGB). The resulting images can be used to train a CNN to classify the TB strains based on their genomic data. This approach has the potential to improve the classification accuracy and provide a more robust method for identifying TB strains.

6.1 Building the dataset

As mentioned earlier, each genome was converted into an image where each pixel is determined by a combination of three metrics of following metrics:

- **Chargaff** index [6]: provides the fraction of nucleic bases G and C of the sequence. The formula is:
$$100 \times \frac{Gcount + Ccount}{Sequence\_size} \tag{2}$$
- **Component** index [13]: evaluates the distribution of the bases in a sequence by comparing the number of each base A and T to the expected frequency of those bases in a balanced distribution. The formula is:
$$\frac{(Acount + Tcount) + \frac{Sequence\_size}{2}}{\frac{Sequence\_size}{2}} \tag{3}$$
- **Diversity** index [14]: measures the diversity of bases in a sequence by comparing the number of different subsequences of a given size to the expected frequency of those subsequences in a balanced distribution. The formula is:
$$\frac{Number_{\text{unique subsequences}}}{Expected_{\text{subsequences frequency}}} \tag{4}$$
- **Skew** index [1]: measures the asymmetry of the distribution of nucleobases in a sequence. The formula is:
$$\frac{(Gcount - Ccount)}{Sequence\_size} \tag{5}$$
- **Nuclescore** index [12]: empirical measure that combines multiple metrics from a genome sequence to synthesize the nucleotide information of a genome in order to differentiate species based only on their sequences. The formula is:
$$\log_2\left(\frac{Variance \times GC_{\text{ratio}} \times AT_{\text{count}}/GC_{\text{ratio}}}{\sqrt{Sequence\_size}}\right) \tag{6}$$

We get ten combinations of these metrics, each represented by a different color channel in the image after being normalized to the range [0, 255]. This conversion is applied at different resolutions, from 4px (2x2) to 100px (10x10), but for the training, all images were resized to a 32x32px resolution. The resulting images put into their corresponding class based on the strain they belong to, as shown in Figure 14. We have the following classes:

- **M** with 185 samples.
- **East Asian** with 1651 samples.
- **East-African Indian** with 267 samples.
- **Euro-American** with 4409 samples.
- **Indo-Oceanic** with 259 samples.

We also made mosaic images by combining the corresponding images from each combination of metrics resulting in a 5x2 grid rectangular image as shown in Figure 15. We chose a 5x2 grid because it creates complete images without any empty pixels, which is important for the training of the CNN. The images are also resized to a 20x50px resolution for training.

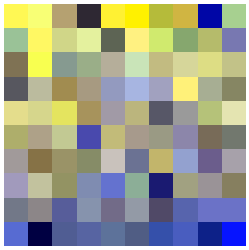

Figure 14: Example of an image generated from the genome of a TB strain using the Chargaff, Component and Diversity metrics at 100px resolution.

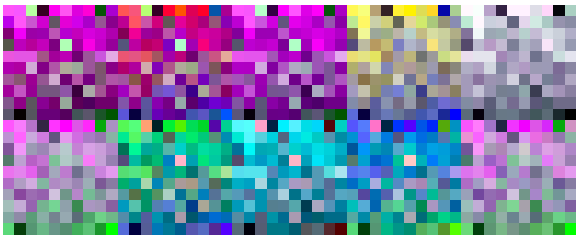

Figure 15: Example of a mosaic image obtained from the individual images at 100px resolution.

6.1.1 Initial testing

The resulting dataset is highly imbalanced, with some classes having significantly more samples than others. This can lead to biased models that perform poorly on minority classes. We decided to train our CNN on the dataset as-is, without any preprocessing or balancing techniques, to see how it performs on the imbalanced dataset. For the sake of brevity, we will only present the results for the combination of Chargaff, Composition and Diversity metrics at 100px resolution, for the mosaic images we will present the results for the 100px resolution as well.

6.1.1.1 Results

As shown in Figure 16, the 4px images yielded the worst performance, with a validation accuracy between 66% and 76%. We then observed a significant improvement in performance as the resolution increases with the accuracy starting to stabilize around 92% and 94% starting from the 36px resolution. Figure 17 and Table 1 show that the overall performances of the model are good, except for the East-African Indian and Indo-Oceanic classes with more moderate performances.

East-African Indian shows a moderate recall of 0.779 and a precision of 0.904, indicating that the model is able to correctly classify most of the samples in this class, but it also misclassifies some samples as other classes. The Indo-Oceanic class has a recall of 0.988, indicating that the model is able to correctly classify most of the samples in this class, but it has a lower precision of 0.718, indicating that the model misclassifies some samples as this class.

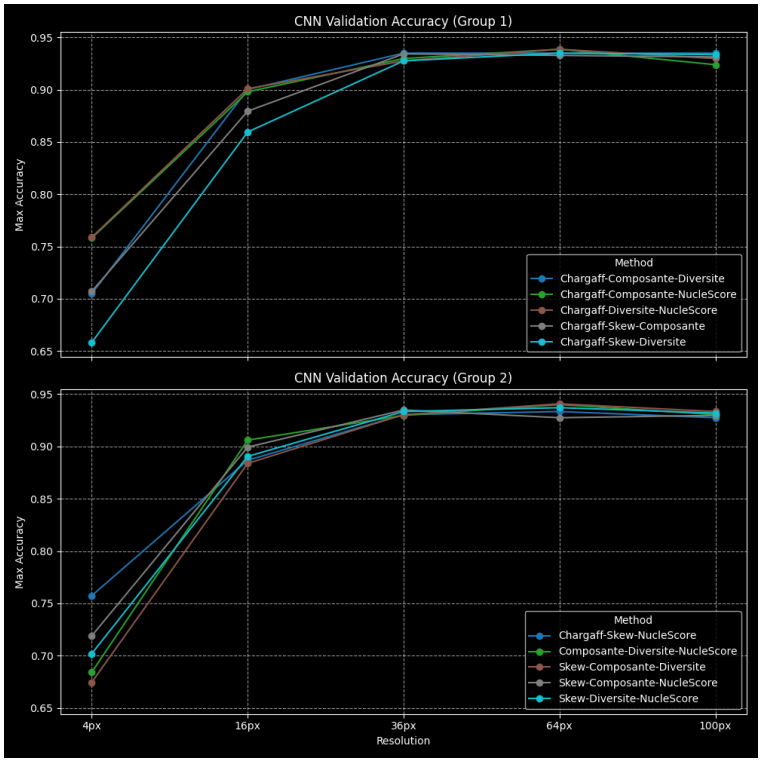

Figure 16: Graph showing the validation accuracy of the model on the different methods at 100px resolution without any preprocessing or balancing techniques applied.

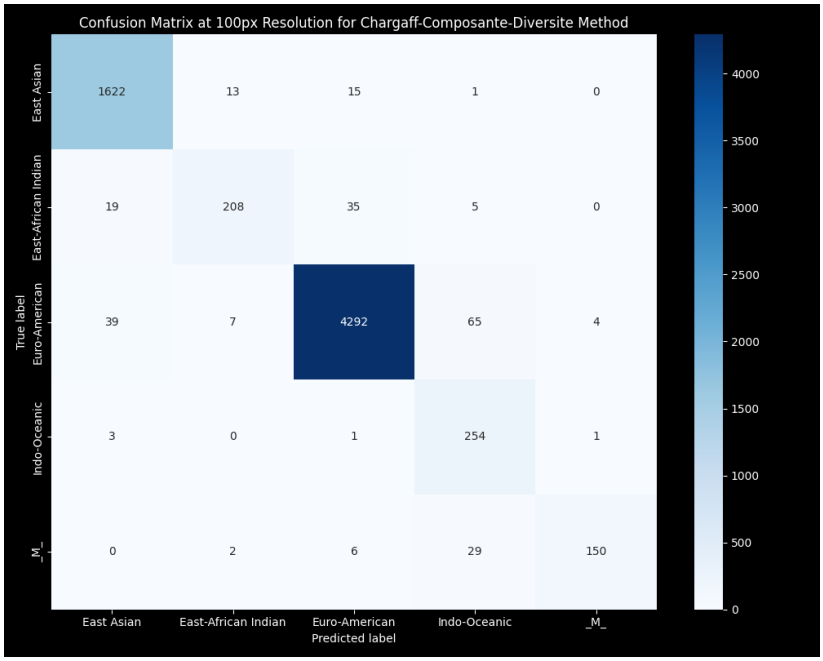

Figure 17: Confusion matrix of the model on the images obtained from the Chargaff-Component-Diversity combination at 100px resolution without any preprocessing or balancing techniques applied.

Class	**Precision**	**Recall**	**Specificity**	**Accuracy**
East Asian	0.965	0.982	0.988	0.987
East-African Indian	0.904	0.779	0.997	0.988
Euro-American	0.987	0.974	0.976	0.975
Indo-Oceanic	0.718	0.988	0.985	0.985
M	0.968	0.802	0.999	0.994

Table 1: Performance metrics of the model on the images obtained from the Chargaff-Component-Diversity combination at 100px resolution without any preprocessing or balancing techniques applied.

For the mosaic images, we observed similar results, with the model achieving a validation accuracy of around 95% at 100px resolution. While most of the metrics are similar to the individual images, the East-African Indian class improved its recall, the precision dropped to 0.662. This indicates that, while the model is able to correctly classify most of the samples in this class, other classes are confused with this class.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.976	0.970	0.992	0.987
East-African Indian	0.662	0.858	0.982	0.977
Euro-American	0.991	0.963	0.984	0.970
Indo-Oceanic	0.771	0.973	0.988	0.988
M	0.966	0.908	0.999	0.997

Table 2: Performance metrics of the model on the mosaic images at 100px resolution without any preprocessing or balancing techniques applied.

Overall, this model shows good performance on the imbalanced dataset, but there is still room for improvement. But we will explore different techniques to address the class imbalance issue in the following sections.

6.2 Mitigating class imbalance

We have multiple ways of addressing the class imbalance issue in our dataset, including:

- Cross validation
- Under-sampling
- Data augmentation
- Combining under-sampling and data augmentation

We will explore each of these techniques in the following sections and compare the results to see which one yields the best performance.

6.2.1 Cross validation

Cross validation is a technique used to evaluate the performance of a model by splitting the dataset into multiple subsets, or folds. The model is trained on a subset of the data and tested on the remaining data, and this process is repeated for each fold. This allows for a more robust evaluation of the model's performance, as it reduces the risk of overfitting to a specific subset of the data.

Remaking the dataset

During our testing, we realized that we were testing our model with images the model was trained on, which is not a good practice. We decided to split our dataset into a training set and a test set, with 90% of the samples in the training set and 10% in the test set. The files are separated into two folders to avoid any overlap between the training and test sets.

6.2.1.1 Results

During our testing, we used the K-fold cross validation technique with 5 folds with 15 epochs and a batch size of 32. We observed that the model was able to achieve a validation accuracy of around 96% during validation for all of 10 combinations of the metrics at 100px resolution, as shown in Figure 18. The confusion matrix in Figure 19 shows that the model was able to correctly classify most of the samples, with only a few misclassifications.

With Table 3 we can see that the model has good performance across most classes. Surprisingly enough, the M class, which has the least number of samples, has better performance than the East-African Indian class, which has more samples. This suggests that the model is able to generalize well to the minority classes, even with the class imbalance issue.

For the mosaic images, we applied the same k-fold cross validation technique with 5 folds and 15 epochs, and we observed similar results compared to the individual images. The East-African Indian class has a slightly lower precision but a slightly higher recall, indicating that the model is able to correctly classify most of the samples in this class, but it also misclassifies some samples as other classes. The Indo-Oceanic class has a recall of 1, indicating that the model was able to correctly classify all the samples in this class, but it has a lower precision of 0.684, indicating that the model misclassifies some samples as this class. Specificity and accuracy are nearly the same as the individual images, as shown in Table 4.

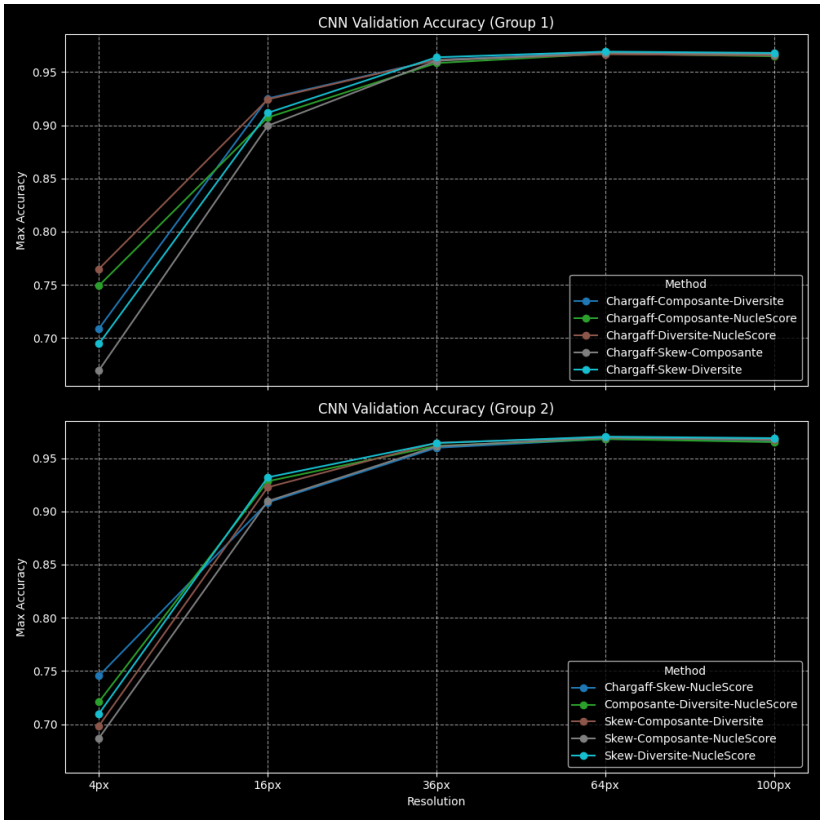

Figure 18: Graph showing the validation accuracy of the CNN model with k-fold cross validation applied on the different methods.

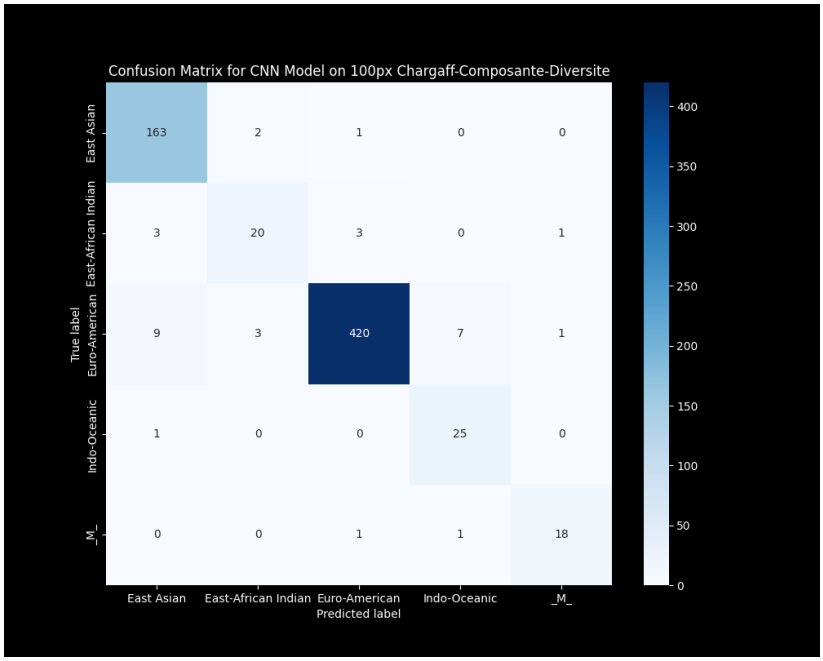

Figure 19: Confusion matrix of the CNN model with k-fold cross validation applied on the images obtained from the Chargaff-Component-Diversity combination at 100px resolution.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.926	0.982	0.975	0.976
East-African Indian	0.800	0.741	0.992	0.982
Euro-American	0.988	0.955	0.979	0.963
Indo-Oceanic	0.758	0.962	0.988	0.987
M	0.900	0.900	0.997	0.994

Table 3: Performance metrics with k-fold cross validation applied on the images obtained from the Chargaff-Component-Diversity combination at 100px resolution.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.932	0.988	0.977	0.979
East-African Indian	0.724	0.778	0.988	0.979
Euro-American	0.995	0.945	0.992	0.962
Indo-Oceanic	0.684	1.000	0.982	0.982
M	0.944	0.850	0.998	0.994

Table 4: Performance metrics with k-fold cross validation applied on the mosaic images at 100px resolution.

The performances of the model with cross validation are close to the ones obtained without cross validation. It seems to perform slightly better, but the difference is not significant enough to conclude that cross validation improves the performance of the model. However, it is important to note that cross validation is a good practice to evaluate the performance of a model, as it reduces the risk of overfitting to a specific subset of the data and provides a more robust evaluation of the model's performance.

6.2.2 Under-sampling

Under-sampling is a technique used to address the class imbalance issue by reducing the number of samples in the majority classes. In our application, we decided to reduce the number of samples in the Euro-American and East Asian classes to a total of 1000 samples each, while keeping the other classes unchanged. This was done to balance the dataset and ensure that the model is not biased towards the majority classes.

Results

This time we observe a drop in the validation accuracy of the model (Figure 28) compared to the previous results, with the accuracy being between 89% and 93% for the 100px resolution. There is also a significant drop in the performance of the model on the East-African Indian class, while the recall slightly improves, the precision drops significantly, indicating that the model identify other classes as East-African Indian. This is likely due to the reduced number of samples in the Euro-American and East Asian classes, which may have led to a less representative training set for these classes. It can be seen in Figure 29 and Table 5.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.964	0.982	0.988	0.987
East-African Indian	0.571	0.889	0.972	0.969
Euro-American	0.998	0.936	0.996	0.957
Indo-Oceanic	0.765	1.000	0.988	0.988
M	0.905	0.950	0.997	0.996

Table 5: Performance metrics with under-sampling applied on the images obtained from the Chargaff-Component-Diversity combinations at 100px resolution.

For the mosaic images, we observe an even more significant drop in the precision of the East-African Indian class, which drops to 0.361 from 0.724 with cross validation on these images (Figure 6). The model misclassifies a lot of Euro-American samples as East-African Indian, more than there are actual East-African Indian samples in the dataset (Figure 31).

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.952	0.952	0.984	0.976
East-African Indian	0.361	0.815	0.940	0.935
Euro-American	0.992	0.893	0.987	0.926
Indo-Oceanic	0.765	1.000	0.988	0.988
M	0.864	0.950	0.995	0.994

Table 6: Performance metrics with under-sampling applied on the mosaic images at 100px resolution.

Instead of improving the performance of the model like we hoped, under-sampling led to a drop in the performance of the model, especially on the East-African Indian class. Still, even with the lowest number of samples, the M class keeps stable metrics. We're still unsure why this is the case, but it could be due to the fact that the M class has a very distinct pattern that the model can pick up on, even with a reduced number of samples.

6.2.3 Data augmentation

For this test, we use the SMOTE-generated images to increase the number of samples in the minority classes, East-African Indian, Indo-Oceanic and M, the other classes are kept unchanged. The aforementioned classes are augmented so that the resulting number of samples contains 25% of generated images for each class.

Results

With data augmentation, we once again observe similar behavior for the East-African Indian and Indo-Oceanic classes between the individual and mosaic images. The individual images seem to yield a better precision metric for the Indo-Oceanic class (Table 7), while the mosaic images favors the East-African Indian class (Table 8) in that regard. But this time, those two classes are roughly on par with each other, with metrics that are close, if not slightly better than with cross validation. This technique seems to be promising, as it allows us to increase the number of samples in the minority classes while keeping the majority classes unchanged.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.976	0.976	0.992	0.988
East-African Indian	0.759	0.815	0.989	0.982
Euro-American	0.986	0.968	0.975	0.970
Indo-Oceanic	0.812	1.000	0.991	0.991
M	0.950	0.950	0.998	0.997

Table 7: Performance metrics the images obtained from the Chargaff-Component-Diversity combinations at 100px resolution with data augmentation applied on the training set.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.970	0.976	0.990	0.987
East-African Indian	0.815	0.815	0.992	0.985
Euro-American	0.993	0.970	0.987	0.976
Indo-Oceanic	0.722	1.000	0.985	0.985
M	0.947	0.900	0.998	0.996

Table 8: Performance metrics the mosaic images at 100px resolution with data augmentation applied on the training set.

6.2.4 Combining techniques

Results

This time, we combined the under-sampling and data augmentation techniques hoping to get the best results. But, while the results for the individual images are similar to the ones obtained from the previous techniques (Table 9), the mosaic images show a significant drop in the performance of the model (Table 10). The East-African Indian class has a precision of 0.307, which is the worst performance we have seen so far. Once again, the model misclassified a lot of Euro-American samples as East-African Indian, which may be a side effect of the combination.

Earlier, we stated that we would only present the results for images at 100px resolution for the Chargaff-Component-Diversity combination, but it's worth mentioning that, for the individual images, we observed a disparity in validation accuracy between the different combinations. The accuracy ranges from 75% to 82% at 100px resolution, they also represent the lowest accuracies observed for the individual images as shown in Figure 36.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.953	0.970	0.984	0.981
East-African Indian	0.706	0.889	0.985	0.981
Euro-American	0.995	0.952	0.992	0.966
Indo-Oceanic	0.743	1.000	0.986	0.987
M	0.950	0.950	0.998	0.997

Table 9: Performance metrics with combined under-sampling and data augmentation techniques applied on the images obtained from the Chargaff-Component-Diversity combinations at 100px resolution.

Class	Precision	Recall	Specificity	Accuracy
East Asian	0.968	0.922	0.990	0.973
East-African Indian	0.307	1.000	0.906	0.910
Euro-American	0.995	0.864	0.992	0.909
Indo-Oceanic	0.839	1.000	0.992	0.993
M	0.950	0.950	0.998	0.997

Table 10: Performance metrics with combined under-sampling and data augmentation techniques applied on the mosaic images at 100px resolution.

6.3 Comparing the results

We first ran the CNN model on the dataset without any preprocessing or balancing techniques applied, and we observed that the model was able to achieve a validation accuracy of between 92% and 94% at 100px resolution. However, this model was trained and tested on the same dataset, which means that the model may be biased. The dataset was also highly imbalanced, with some classes having significantly more samples than others.

After splitting the dataset into a training set and a test set, we applied k-fold cross validation which yielded decent results. But the East-African Indian and Indo-Oceanic classes had lower precision and recall metrics compared to the other classes, except for the Indo-Oceanic class which had a recall of 1 with the mosaic images. Then we tried under-sampling the dataset, which led to a drop in the performance for both individual and mosaic images. Data augmentation was then applied to the dataset, which allowed us to increase the number of samples in the minority classes. This led to better performance for the Indo-Oceanic class with the individual images, while the East-African Indian class had a better performance with the mosaic images. Finally, we combined under-sampling and data augmentation techniques, which led to a drop in the performance of the model for the mosaic images, while the individual images remained stable.

Overall, the best results were obtained with SMOTE data augmentation, which allowed us to increase the number of samples in the minority classes while keeping the majority classes unchanged.

7 Conclusion

During this project, we have explored various techniques for genomic data pre-processing and analysis, focusing on the Mycobacterium tuberculosis (MTB) genome. We have implemented several methods for data pre-processing and analysis, such as K-means clustering, DBSCAN, HDBSCAN and an autoencoder. However, we encountered several challenges, particularly with the clustering methods, which did not yield satisfactory results. We theorized that, either the data was not suitable for clustering due to its high dimensionality, or that they were not prepared correctly.

We also explored the use of Convolutional Neural Networks (CNNs) for image classification, which showed promising results. The images were generated from the genomic data by applying a combination of three of five metrics to the nucleotide sequences, resulting in ten different combinations of metrics, each represented by a different color channel in the image. We also created a dataset of mosaic images, combining the images from the ten different combinations of metrics.

The resulting dataset was highly imbalanced, with a significant number of samples belonging to two of the fives classes, Euro-American and East-Asian. To address this issue, we applied different techniques such as K-fold cross validation, under-sampling, data augmentation using SMOTE, and a combination of both under-sampling and data augmentation. The best results were achieved with the data augmentation technique, which improved the metrics for the Indo-Oceanic class on the individual images, and the East-African Indian class on the mosaic images.

Another behavior we observed was that, despite being the class with the lowest number of samples, the M class always had metrics on par or close to the majority classes. This suggests that the M class has a specific pattern that is distinct from the other classes, which could be further investigated in future work.

In conclusion, this project has provided valuable insights into the challenges and opportunities of genomic data pre-processing and analysis. We have demonstrated the potential of using CNNs for image classification of genomic data, and the importance of addressing class imbalance in the dataset. I personally learned a lot about the fields of genomics and bioinformatics. This project has been a great opportunity to apply my knowledge in machine learning and data analysis to a real-world problem.

A Additional Figures

A.1 DBSCAN and HDBSCAN Clustering

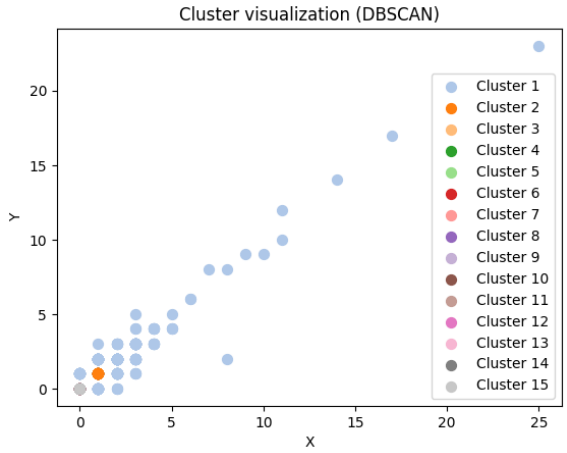

Figure 20: Clusters obtained with DBSCAN after applying PCA.

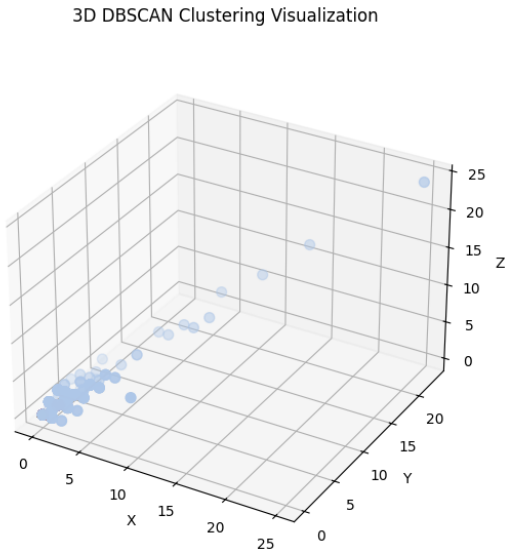

Figure 21: Clusters obtained with DBSCAN after applying PCA in 3D.

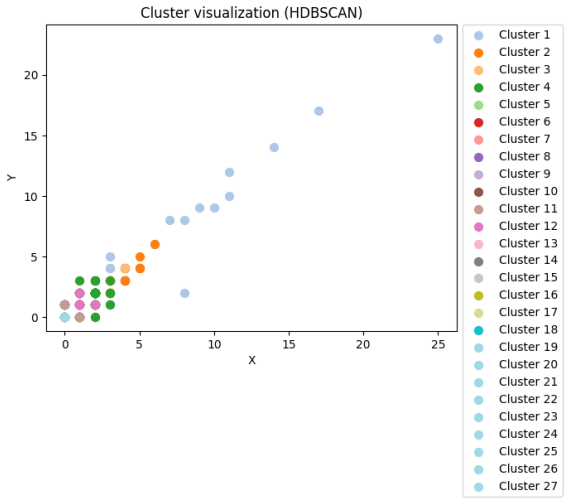

Figure 22: Clusters obtained with DBSCAN after applying PCA.

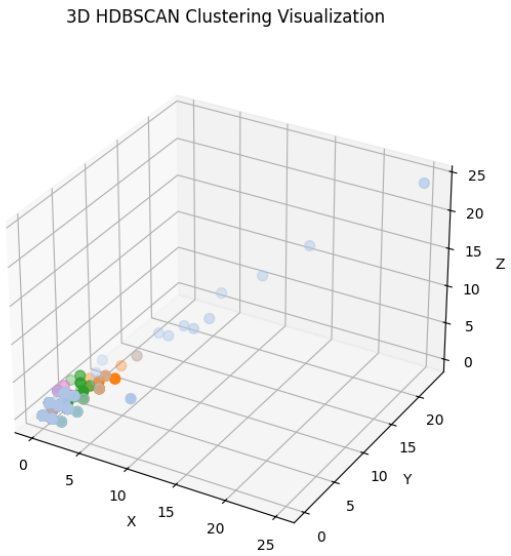

Figure 23: Clusters obtained with DBSCAN after applying PCA in 3D.

A.2 Initial CNN Model Performance

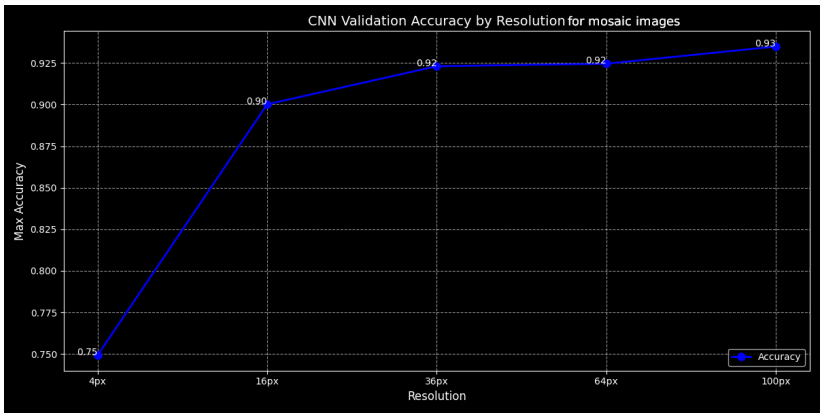

Figure 24: Graph showing the validation accuracy of the CNN model on the mosaic images at 100px resolution without any additional techniques applied.

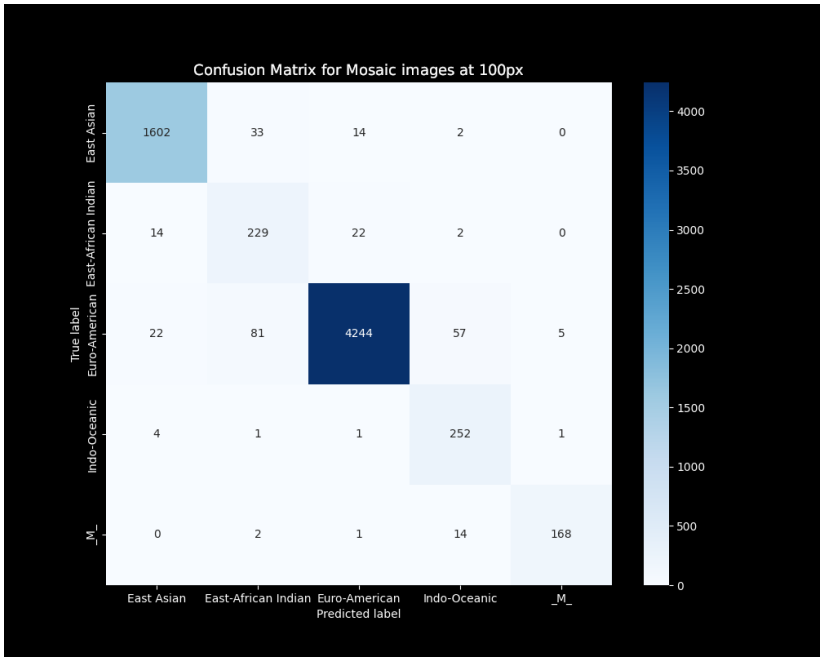

Figure 25: Confusion matrix on the mosaic images at 100px resolution without any additional techniques applied.

A.3 K-Fold Cross Validation

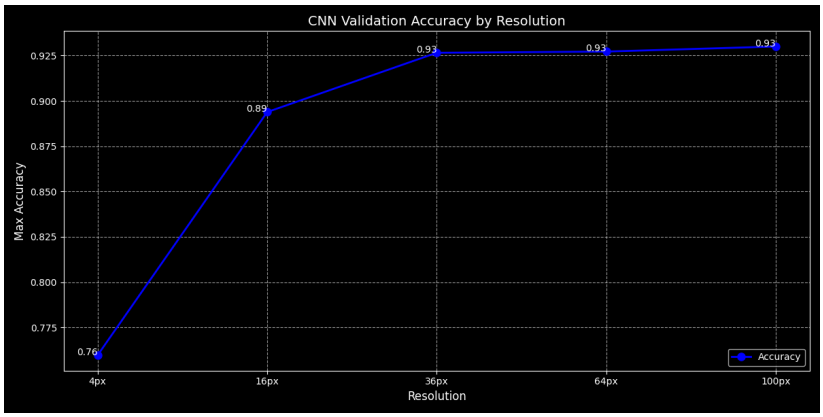

Figure 26: Graph showing the validation accuracy of the CNN model with k-fold cross validation applied on the mosaic images.

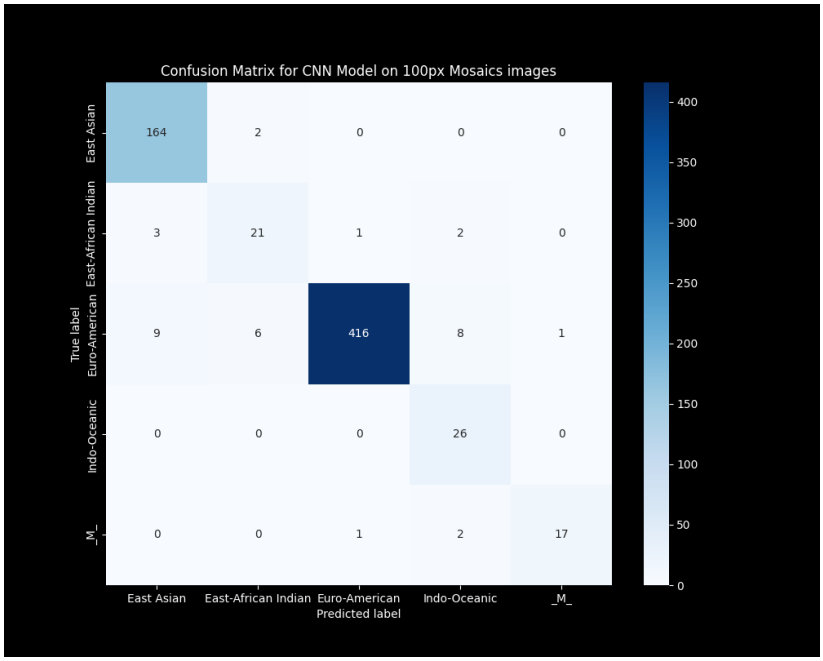

Figure 27: Confusion matrix with k-fold cross validation applied on the mosaic images at 100px resolution.

A.4 Under-Sampling techniques applied

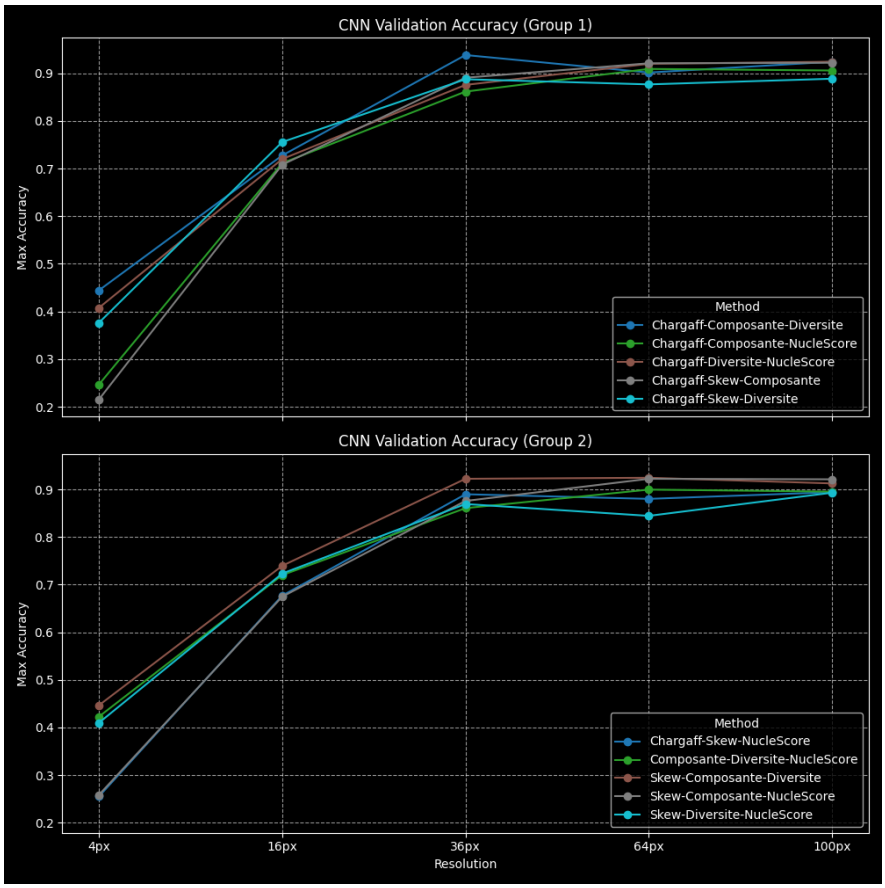

Figure 28: Graph showing the validation accuracy of the CNN model with under-sampling applied on the different methods.

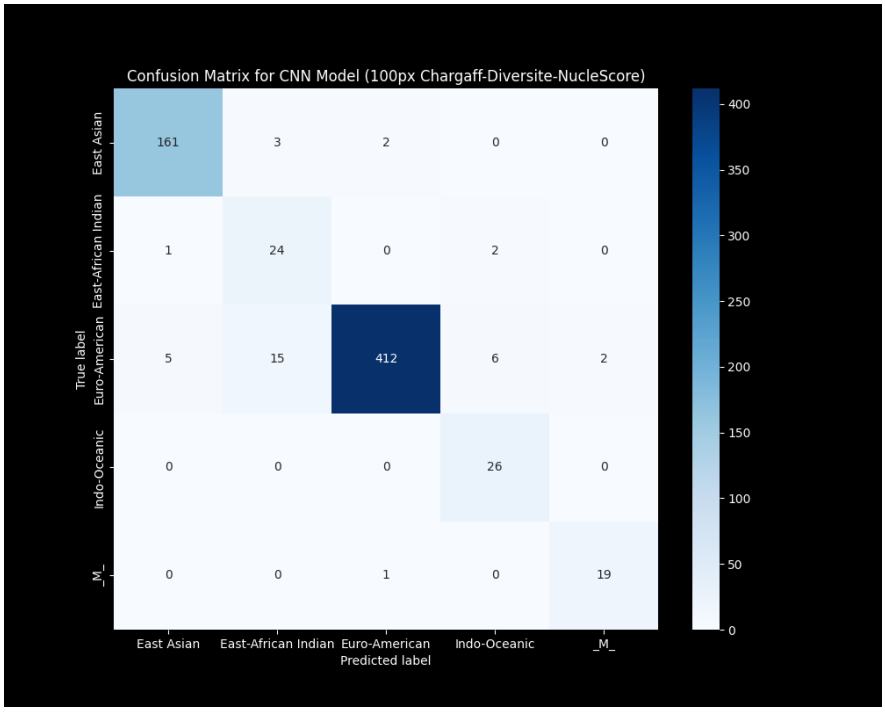

Figure 29: Confusion matrix with under-sampling applied on the images obtained from the Chargaff-Diversity-NucleScore combination at 100px resolution.

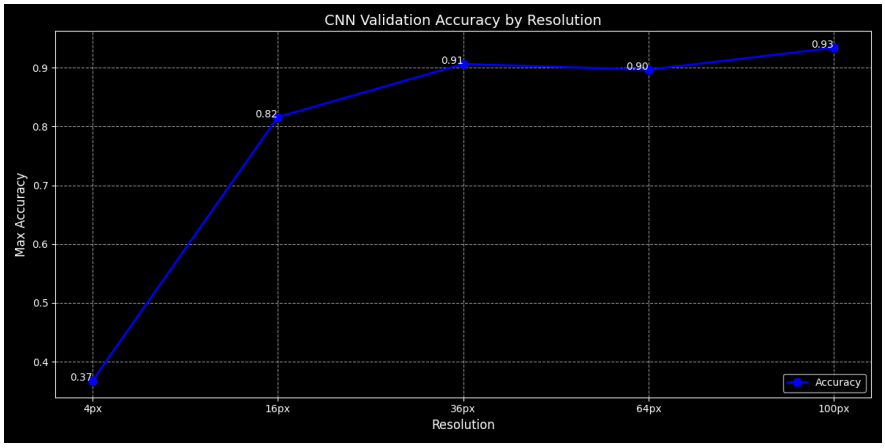

Figure 30: Graph showing the validation accuracy of the CNN model with under-sampling applied on the mosaic images.

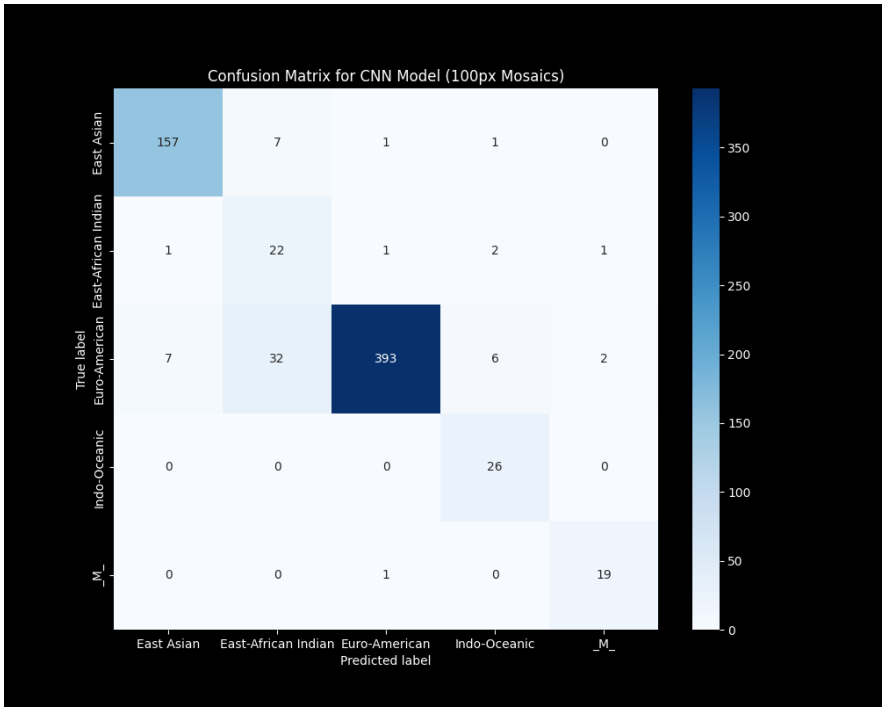

Figure 31: Confusion matrix with under-sampling applied on the mosaic images at 100px resolution.

A.5 Data Augmentation techniques applied

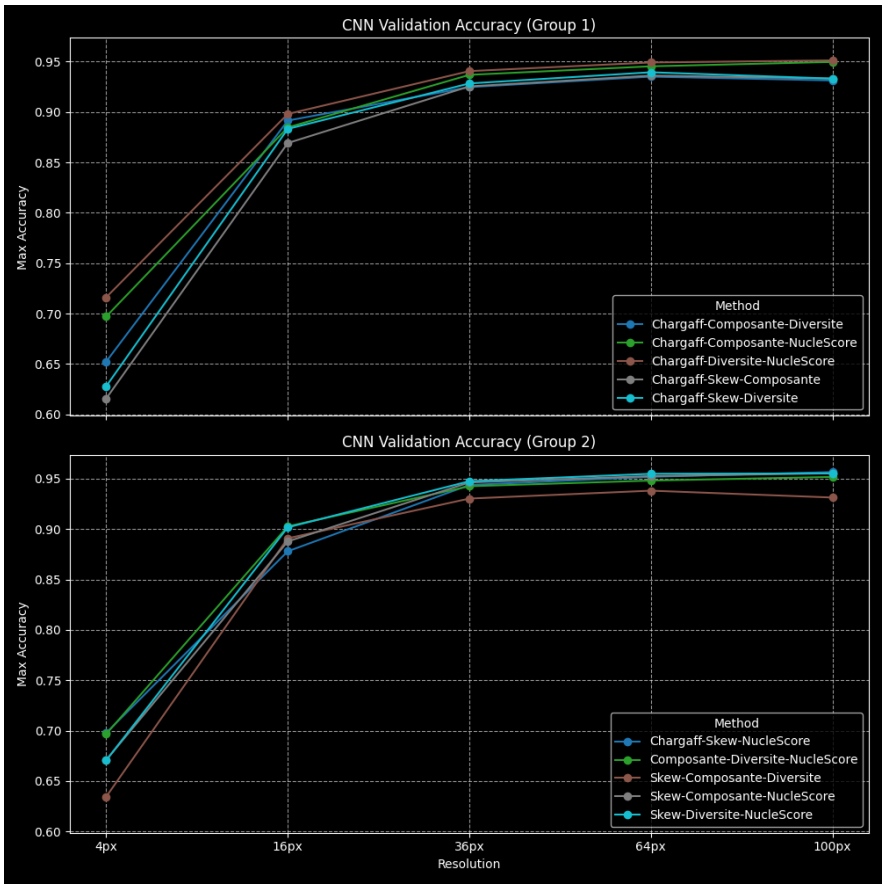

Figure 32: Graph showing the validation accuracy of the CNN model with data augmentation applied on the different methods.

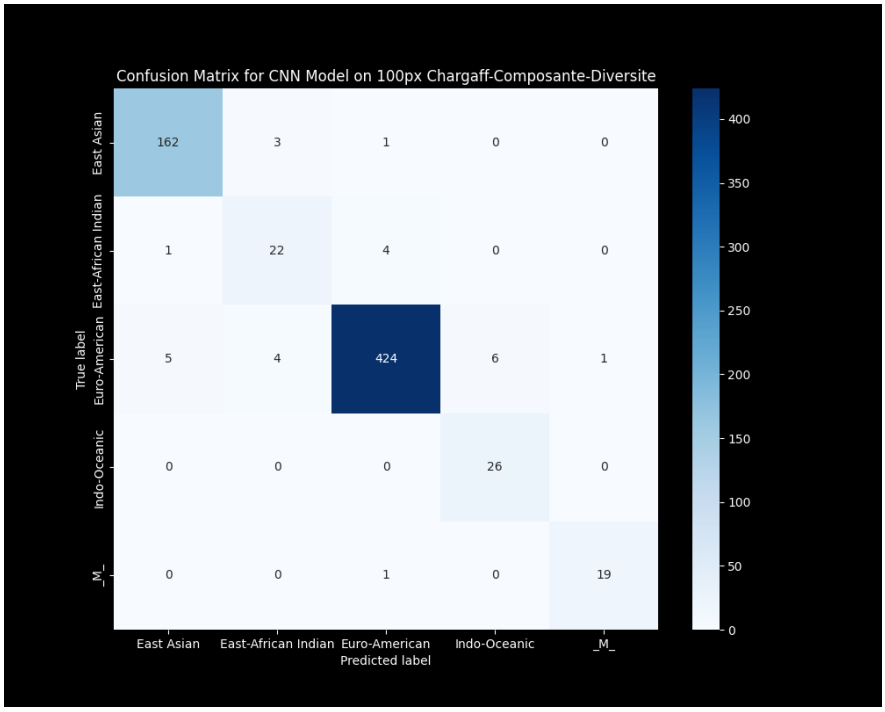

Figure 33: Confusion matrix with data augmentation applied on the images obtained from the Chargaff-Diversity-NucleScore combination at 100px resolution.

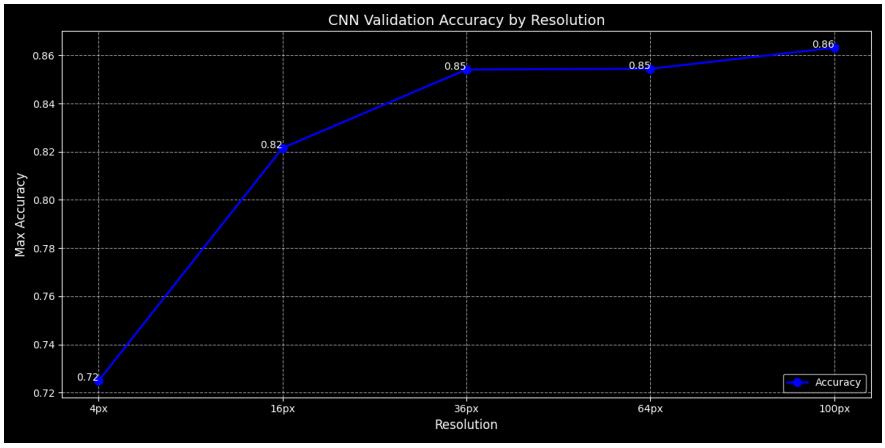

Figure 34: Graph showing the validation accuracy of the CNN model with data augmentation applied on the mosaic images.

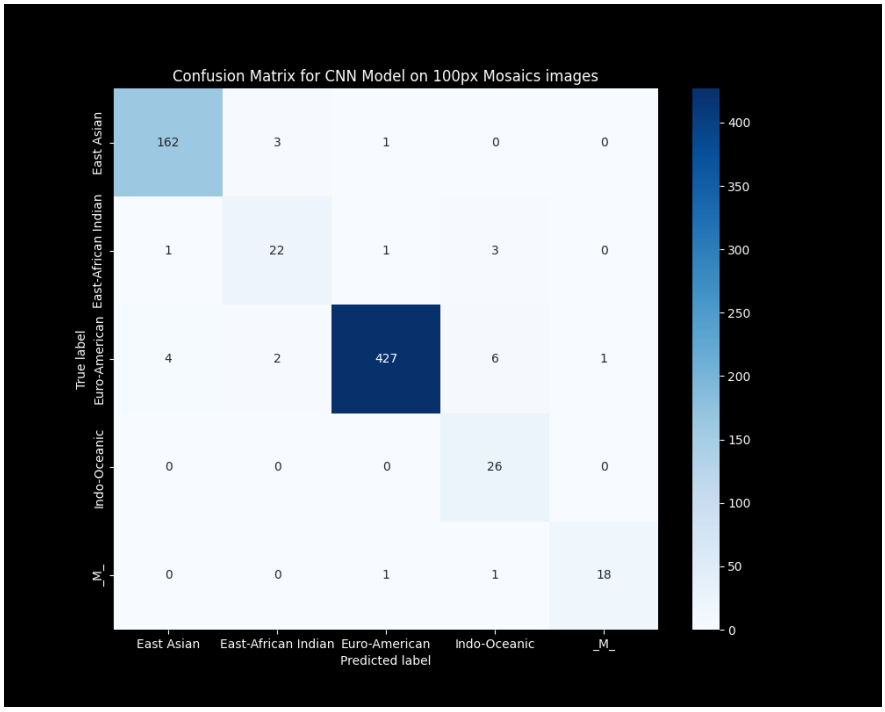

Figure 35: Confusion matrix with data augmentation applied on the mosaic images at 100px resolution.

A.6 Combined Techniques

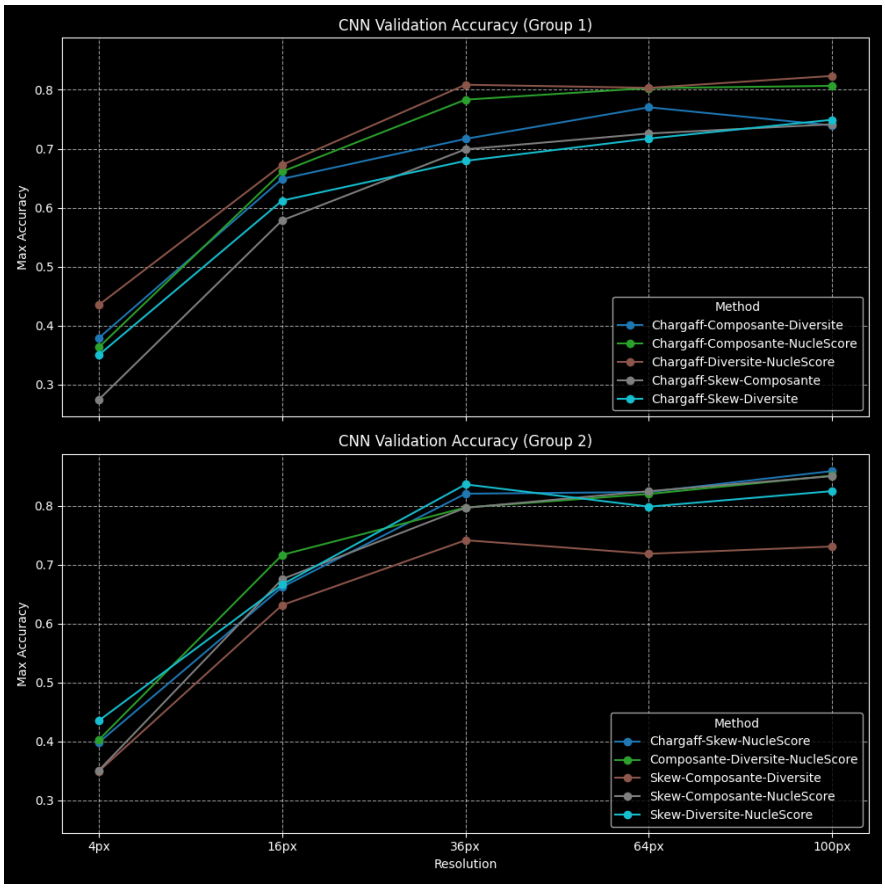

Figure 36: Graph showing the validation accuracy of the CNN model with combined undersampling and data augmentation techniques applied on the different methods.

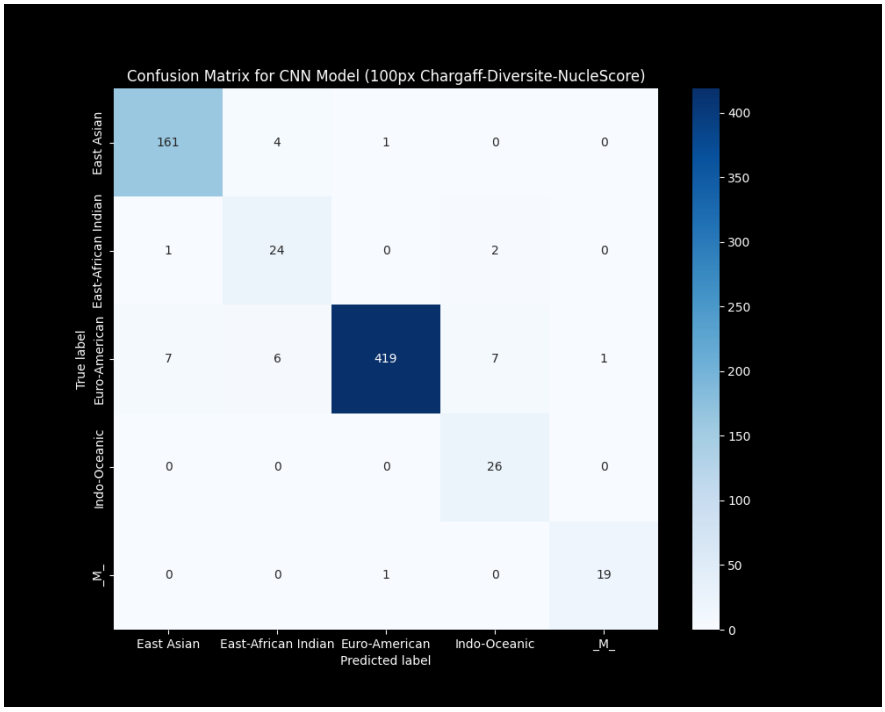

Figure 37: Confusion matrix with combined under-sampling and data augmentation techniques applied on the images obtained from the Chargaff-Diversity-NucleScore combination at 100px resolution.

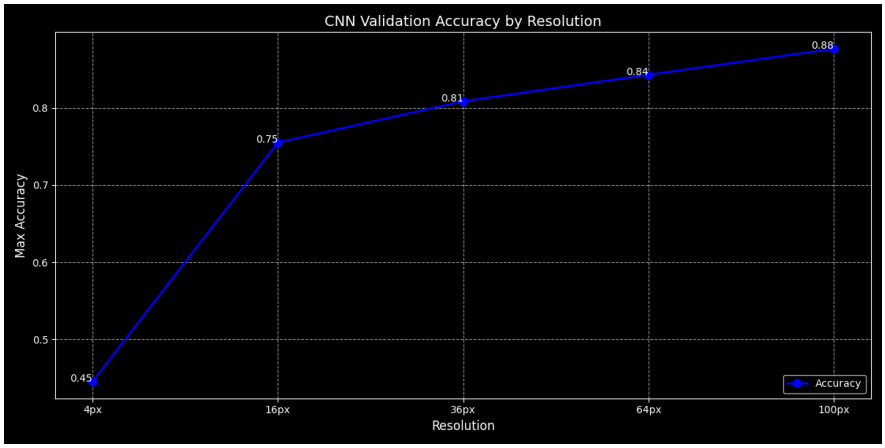

Figure 38: Graph showing the validation accuracy of the CNN model with combined under-sampling and data augmentation techniques applied on the mosaic images.

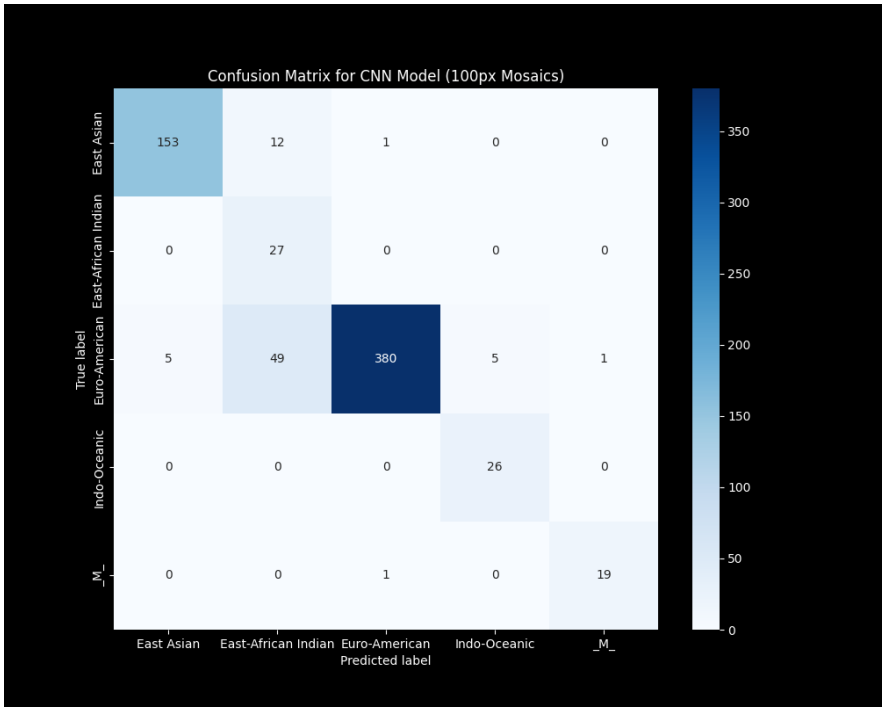

Figure 39: Confusion matrix with combined under-sampling and data augmentation techniques applied on the mosaic images at 100px resolution.

Bibliography

[1] Kazuharu Arakawa and Masaru Tomita. "The GC Skew Index: A Measure of Genomic Compositional Asymmetry and the Degree of Replicational Selection". In: *Evolutionary Bioinformatics* 3 (2007), p. 117693430700300006. DOI: 10.1177/117693430700300006. eprint: https://doi.org/10.1177/117693430700300006. URL: https://doi.org/10.1177/117693430700300006.

[2] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[3] Ricardo J. G. B. Campello et al. "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection". In: *ACM Trans. Knowl. Discov. Data* 10.1 (July 2015). ISSN: 1556-4681. DOI: 10.1145/2733381. URL: https://doi.org/10.1145/2733381.

[4] Claudia Caudai et al. "AI applications in functional genomics". In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 5762–5790. ISSN: 2001-0370. DOI: https://doi.org/10.1016/j.csbj.2021.10.009. URL: https://www.sciencedirect.com/science/article/pii/S2001037021004311.

[5] Xi Chen and Hemant Ishwaran. "Random forests for genomic data analysis". In: *Genomics* 99.6 (2012), pp. 323–329. ISSN: 0888-7543. DOI: https://doi.org/10.1016/j.ygeno.2012.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0888754312000626.

[6] T.A. Clarke and L.D. Hurst. "Gc-content evolution in prokaryotes and eukaryotes: a tale of two genomes". In: *Trends in Genetics* 22.11 (2006), pp. 632–640.

[7] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[8] *FASTA Format for Nucleotide Sequences*. URL: https://www.ncbi.nlm.nih.gov/genbank/fastaformat/.

[9] Kunihiko Fukushima. "Self-organizing Neural Network Models for Visual Pattern Recognition". In: *Plasticity of the Central Nervous System*. Ed. by Keiji Sano and Shozo Ishii. Vienna: Springer Vienna, 1987, pp. 51–67. ISBN: 978-3-7091-8945-0.

[10] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[11] V. I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals". In: *Soviet Physics Doklady* 10 (Feb. 1966), p. 707.

[12] Vincent Moco et al. "getSequenceInfo: a suite of tools allowing to get genome sequence information from public repositories". In: *BMC Bioinformatics* 23.1 (2022), p. 268. ISSN: 1471-2105. DOI: 10.1186/s12859-022-04809-5. URL: https://doi.org/10.1186/s12859-022-04809-5.

[13] Jan Mrázek and Samuel Karlin. "Strand compositional asymmetry in bacterial and large viral genomes". In: *Proceedings of the National Academy of Sciences* 95.7 (1998), pp. 3720–3725. DOI: 10.1073/pnas.95.7.3720. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.95.7.3720. URL: https://www.pnas.org/doi/abs/10.1073/pnas.95.7.3720.

[14] José Manuel Peregrín-Álvarez and John Parkinson. "The global landscape of sequence diversity". In: *Genome Biology* 8.11 (2007), R238. ISSN: 1474-760X. DOI: 10.1186/gb-2007-8-11-r238. URL: https://doi.org/10.1186/gb-2007-8-11-r238.

[15] Sebastian Raschka and Vahid Mirjalili. "Python machine learning second edition". In: *Birmingham, England: Packt Publishing* (2017), p. 192.

[16] SWASTIK. *Smote for Imbalanced Classification with Python, Technique*. Oct. 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

[17] Wikipedia contributors. *FASTA format — Wikipedia, The Free Encyclopedia*. 2025. URL: https://en.wikipedia.org/w/index.php?title=FASTA_format&oldid=1291980692.

[18] Juexiao Zhou et al. "An AI Agent for Fully Automated Multi-omic Analyses". In: *bioRxiv* (2024). DOI: 10.1101/2023.09.08.556814. eprint: https://www.biorxiv.org/content/early/2024/01/05/2023.09.08.556814.full.pdf. URL: https://www.biorxiv.org/content/early/2024/01/05/2023.09.08.556814.