

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Generative Model for Pseudomonad Genomes

Permalink

<https://escholarship.org/uc/item/3612z5zh>

Author

Kesapragada, Manasa

Publication Date

2023

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-ShareAlike License, available at <https://creativecommons.org/licenses/by-sa/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

GENERATIVE MODEL FOR PSEUDOMONAD GENOMES

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

APPLIED MATHEMATICS

by

Manasa Kesapragada

December 2023

The Master's thesis of
Manasa Kesapragada is approved:

Professor Hongyun Wang

Associate Professor Marcella M. Gomez

Peter Biehl
Vice Provost and Dean of Graduate Studies

Copyright © by
Manasa Kesapragada
2023

Contents

| | |
|--|-----------|
| Figures | iv |
| Abstract | v |
| Acknowledgments | vi |
| 1 Introduction | 1 |
| 2 Background | 3 |
| 2.1 Pseudomonas bacteria | 3 |
| 2.2 Generative modeling | 3 |
| 2.3 Generative Adversarial Networks (GANs) | 5 |
| 2.3.1 DC GANs | 8 |
| 2.3.2 WGAN | 9 |
| 3 Methods | 11 |
| 3.1 Pangenome Analysis | 11 |
| 3.2 Data Input | 14 |
| 3.3 Model | 16 |
| 3.3.1 Generative Model Architecture | 16 |
| 3.3.2 Model Training | 16 |
| 4 Results | 18 |
| 5 Future Works | 20 |
| 6 Conclusion | 23 |
| Bibliography | 27 |

Figures

List of Figures

| | | |
|-----|---|----|
| 2.1 | The functional and environmental range of <i>Pseudomonas</i> spp. Figure reference: [1] | 4 |
| 2.2 | Generative models model the distribution of individual classes: showing an example of the distribution of two classes | 5 |
| 2.3 | Generative Adversarial Networks (GANs): Figure illustrating the significance of each term in GANs. | 6 |
| 2.4 | Generative Adversarial Networks (GANs) - workflow. Fig reference: [2] | 7 |
| 3.1 | Parts of the pangenome. | 12 |
| 3.2 | Parts of the pangenome. | 13 |
| 3.3 | Gene Presence/Absence matrix | 15 |
| 3.4 | WGAN Model for Pseudomonad genomes | 17 |
| 4.1 | Generator output evaluation | 19 |
| 5.1 | Conditional GAN | 21 |

Abstract

Generative model for Pseudomonad genomes

by

Manasa Kesapragada

Recent advances in genomic sequencing have resulted in several thousands of full genomes of pseudomonads, a genera of bacteria important in many science areas ranging from biogeochemical cycling in the environment to bacterial pneumonia in humans. With these high-quality data sets, combined with tens of thousands of somewhat lower quality metagenomically assembled genomes, we create a generative model for pseudomonad genomes. We present a Generative Adversarial Network (GAN) model that generates gene family presence absence lists as a representation of a novel genome. We also demonstrate that the discriminator of this model can be used as a binary classifier to identify incorrect genomes with missing content. In the future, our desired model can be used to generate genomes within a given set of parameters such as, “Generate a genome that is root associated, drought resistant, salt tolerant that will produce this natural product”.

Acknowledgements

This project represents my summer internship experience at Lawrence Berkeley National Laboratory. This work was supported by the Office of Science, Office of Biological and Environmental Research, of the US Department of Energy under Award Numbers DE-AC02-05CH11231, DE-AC02-06CH11357, DE-AC05-00OR22725, and DE-AC02-98CH10886, as part of the DOE Systems Biology Knowledgebase. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.

I thank R Shane Canon and Paramvir S Dehal for giving me this opportunity to work on this great project. I thank Sean P Jungbluth and Marcin P Joachimiak for their incredible guidance. My advisor Marcella Gomez for giving me the guidance and support and believing in me. Ksenia Zlobina for being a great mentor and being beyond helpful.

Chapter 1

Introduction

Synthetic biology is leveraging the extensive DNA sequencing of microbial genomes obtained from natural environments, hosts, and industrial or laboratory settings. The vast collection of sequenced microbial genomes offers valuable insights into gene and trait characteristics, forming a foundation for designing and producing synthetic organisms with significant biotechnological and medical applications.

While pangenomics [3], a computational approach that analyzes the full genomic repertoire of a species, has become a prevalent tool for exploring lineage-specific gene family profiles, its applicability in synthetic organism design remains limited. The underlying rules governing the transformation of a pangenome into a functional genome remain elusive, hindering our ability to rationally design synthetic organisms with desired properties.

Despite the widespread interest in precisely designing and producing artificial microbial genomes, this goal poses a significant challenge for synthetic biology. This study specifically focuses on the gene content aspect of the problem.

To address this challenge, we propose using Generative Adversarial Networks (GANs)

Chapter 1 Introduction

as a potent artificial microbial genome design method. GANs, a class of machine learning algorithms, excel at generating realistic data by learning from real-world examples. Our approach involves training a generative model on gene family presence/absence profiles derived from the extensively studied *Pseudomonas* bacterial lineage, known for its extensive host range. The model is designed to create artificial gene presence/absence lists, forming the basis for constructing analog genome sequences. From this basic model, our goal is to advance toward generating complete genome sequences with additional specified properties. This novel approach holds promise for enhancing the precision and efficiency of artificial microbial genome design within synthetic biology.

Chapter 2

Background

§ 2.1 Pseudomonas bacteria

Organisms belonging to the Pseudomonas [4] genus exhibit exceptional metabolic and physiological adaptability, allowing them to thrive in various terrestrial and aquatic environments. Their significance stems from their role in both plant and human diseases, as well as their increasing potential in various biotechnological applications [1]. The remarkable diversity within the Pseudomonas genus stems from a rich evolutionary history. The common ancestor of Pseudomonas has navigated a vast array of abiotic and biotic environments, driving the evolution of numerous traits and lifestyles that exhibit considerable overlap across species as shown in Fig. 2.1.

§ 2.2 Generative modeling

Generative models learn to capture the statistical distribution of training data, allowing us to synthesize samples from the learned distribution. They model the distribution of

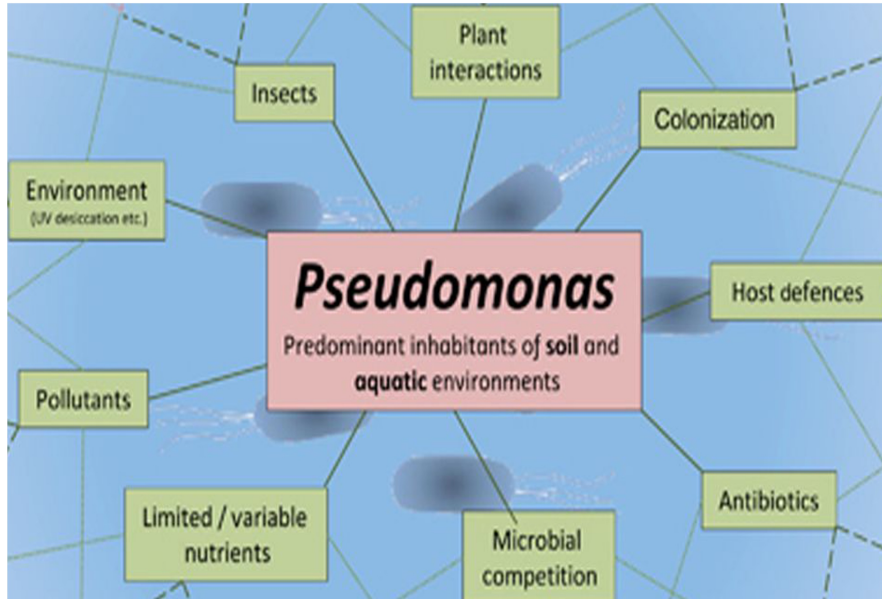


Figure 2.1: The functional and environmental range of *Pseudomonas* spp. Figure reference: [1]

individual classes, while discriminative models focus on learning the boundaries that distinguish between these classes. In Fig. 2.2 (a), the objective of a generative model in classifying blue and orange beads involves examining how the class is generated by considering both the features (x) and labels (y). The generative model utilizes the joint probability:

$$p(x, y) = p(y) * p(x|y)$$

where the model uses $p(x, y)$ as the score to determine whether x corresponds to the blue or orange class. The probability of y , denoted as $p(y)$, can be readily obtained from the available data. With observed data, the model can estimate the distribution of y , illustrated as a normal distribution in Fig. 2.2 (b). Thus, the model determines $p(x|y)$ for blue and orange in their respective distributions. When presented with new data and given y , the generative model classifies the new bead as blue, as depicted in

§2.3 Generative Adversarial Networks (GANs)

Fig. 2.2 (c), based on its clear alignment with the blue distribution.

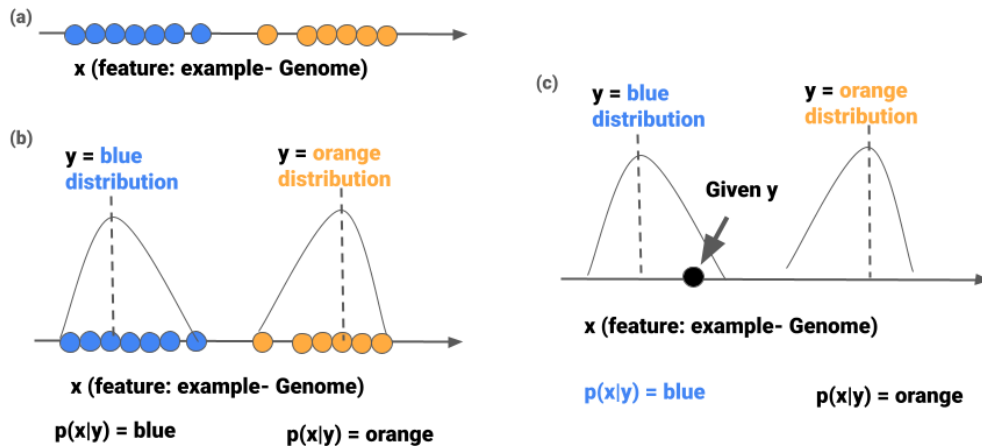


Figure 2.2: Generative models model the distribution of individual classes: showing an example of the distribution of two classes

§ 2.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [5] provide a way to learn deep representations without extensively annotated training data. GANs operate on a unique adversarial framework, where a generator and discriminator engage in a dynamic interplay, each striving to outperform the other. The generator generates data instances indistinguishable from real ones, while the discriminator learns to differentiate between genuine and synthetic samples. This adversarial training process drives GANs to create realistic outputs (Fig. 2.3).

The training process for a GAN model involves iteratively training the generator and discriminator. During each iteration, the generator produces a batch of fake data, and the discriminator is trained to classify the data as either real or fake. The generator is

Generative Adversarial Networks (GANs)

Generative

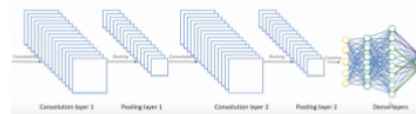
**Generates data
(Creates Fake Data)**

Adversarial

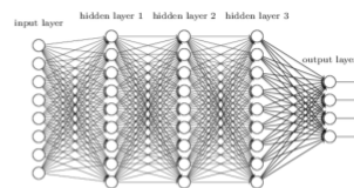
Generator and Discriminator each competing against each other

- Generator trying to fake the data
- Discriminator trying to validate that data
- When the probability of the output from discriminator is 0.5, then generator did a good job

Networks



Deep convolutional



Fully connected (Dense only)

Figure 2.3: Generative Adversarial Networks (GANs): Figure illustrating the significance of each term in GANs.

then updated based on the feedback from the discriminator. The main steps involved in training a GAN model are listed below as shown in Fig. 2.4:

- Define GAN Architecture: The first step in training a GAN model is to define the architecture of the model. This includes specifying the types of layers, the number of neurons in each layer, and the activation functions. The architecture of the GAN should be tailored to the specific application.
- Train discriminator: The discriminator is first trained on a set of real data. This training aims to teach the discriminator to distinguish between real and fake data. The discriminator is trained using a loss function.
- Train generator: The generator is then trained to produce fake data that can fool the discriminator. The generator is trained using a loss function that measures the

§2.3 Generative Adversarial Networks (GANs)

difference between the generated and real data.

- Alternate training: The training of the discriminator and generator is then alternated. This process continues for multiple epochs.
- Save generator model: Once the GAN model has been trained, the generator model can be saved. This generator model can then be used to generate new, realistic fake data.
- Save discriminator model: The discriminator model can also be saved and used as a binary classifier to distinguish between real and fake data.

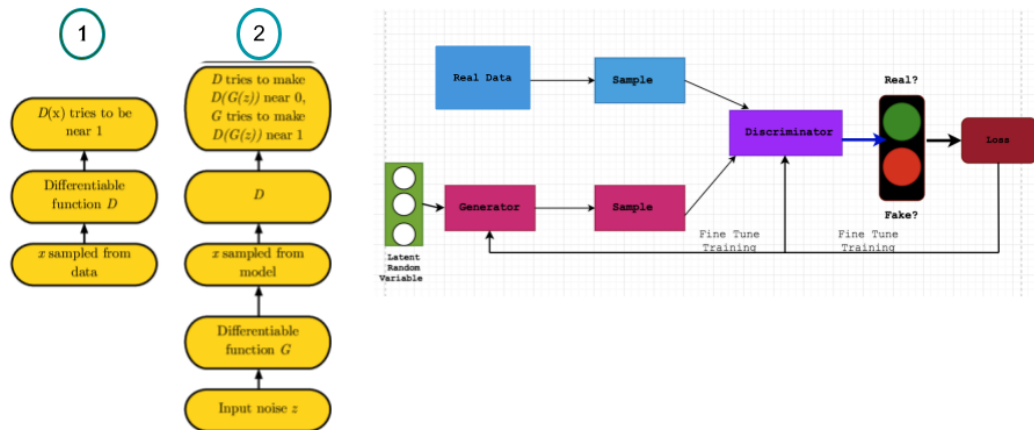


Figure 2.4: Generative Adversarial Networks (GANs) - workflow. Fig reference: [2]

During the training process, the discriminator and generator are trained against each other. This means that the parameters of the discriminator are held constant when the generator is being trained and vice versa. This helps to ensure that the discriminator and generator are constantly learning from each other.

2.3.1 DC GANs

Deep Convolutional Generative Adversarial Networks (DCGANs) [6] enhance the generation of intricate, high-dimensional data, by incorporating convolutional layers into the Generative Adversarial Network (GAN) architecture. Key attributes of DCGANs encompass the use of convolutional layers in both the generator and discriminator networks, efficient down-sampling and up-sampling through strided and fractional-strided convolutions, application of batch normalization for stable and accelerated training, utilization of ReLU activation functions to introduce non-linearity, and avoidance of fully connected layers in hidden layers to prevent overfitting and enhance flexibility in handling input sizes. DCGANs use Binary Cross-Entropy as their loss function. This function represents the average cost for the discriminator misclassifying real and fake observations. The higher the cost function, the worse the discriminator is performing. This indicates the generator wants to maximize this cost function, and the discriminator wants to minimize—which is often referred to as the "minimax game". Some of the issues which arise using this loss function are:

- **Mode Collapse:** Mode collapse occurs when a GAN fails to generate diverse class data. This can happen when the discriminator is trained using the binary cross-entropy loss function. The BCE loss forces the discriminator to output a value between 0 and 1, representing the probability that the input is real. As the discriminator improves, it approaches 0 or 1 for all inputs. This means that the discriminator will eventually classify all data as either extremely fake or extremely real. The generator, in turn, will try to produce data that the discriminator classifies as real. However, if the discriminator is always classifying data as either extremely

§2.3 *Generative Adversarial Networks (GANs)*

fake or extremely real, the generator will only need to produce a small number of data points to fool the discriminator. As a result, the generator will only produce a small number of different types of data, even though the real data is much more diverse. For instance, during training with the MNIST dataset [7], the GAN may only produce a single type of number instead of all 10, such as generating only the digit '2' and neglecting others.

- **Vanishing Gradient:** The vanishing gradient problem occurs because the binary cross-entropy loss restricts the discriminator's confidence values within the 0 to 1 range. As the objective is to approach a value close to 1, the computed gradients tend to approach zero. This situation deprives the generator of adequate information, hindering effective learning. Consequently, this imbalance leads to a robust discriminator but an ineffective generator.

One proposed solution to these issues is adopting Wasserstein loss [8], which approximates the Earth Mover's Distance (EMD) — the effort needed to transform one distribution into another. Wasserstein Generative Adversarial Networks (WGANs) [9] utilize this loss function to address the challenges associated with DCGANs.

2.3.2 WGAN

WGAN introduces the Wasserstein distance (also known as Earth Mover's distance) as the metric for measuring the difference between the generator and discriminator distributions. The key idea is to encourage the discriminator to output values that can be interpreted as a measure of distance between the generated and real data distributions. The Wasserstein distance requires the discriminator to be Lipschitz continuous. This

Chapter 2 Background

constraint helps achieve a more stable and meaningful measure of the difference between real and generated data distributions and helps mitigate issues such as mode collapse and vanishing gradient. In WGAN, Lipschitz continuity is typically enforced using two common techniques: Weight Clipping and Gradient Penalty

WGAN- Weight Clipping

In this approach, the discriminator's weights are clipped to a small constant value to enforce Lipschitz continuity. Weight clipping involves constraining the parameters of the discriminator within a specified range, preventing them from becoming too large. While this method helps stabilize training, it can still lead to training instability and other issues.

WGAN- Gradient Penalty

Instead of relying on weight clipping, WGAN with Gradient Penalty (WGAN-GP) penalizes the norm of the gradient of the discriminator with respect to its input. This penalty is added to the original Wasserstein loss to encourage the Lipschitz constraint without the need for aggressive weight clipping. WGAN-GP often leads to more stable training and avoids some of the problems associated with weight clipping.

In the current study, we use WGAN-GP in order to ensure model convergence and stability.

Chapter 3

Methods

Genus-level *Pseudomonas* genome data is sourced and downloaded from the National Center for Biotechnology Information (NCBI)[10], licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. The Genome Taxonomy Database (GTDB)[11] is used as a guide to identify *Pseudomonad* nucleotide sequence genomes of interest. In this study, the analysis is conducted on a randomly selected subset of 3,000 genomes from approximately 12,000 *Pseudomonad* genomes available on NCBI. The analysis utilized computational resources from the DOE Systems Biology Knowledge Base (KBase) [12] and the National Energy Research Scientific Computing Center (NERSC).

§ 3.1 Pangenome Analysis

The first step towards building the model is to perform a pangenome analysis. A pangenome (Fig. 3.1) is the entire set of genes from all strains within a group. It can be understood as the union of all the genomes of a group. The pangenome can be broken down into a "core pangenome" that contains genes present in all genomes,

a "shell pangenome" that contains genes present in two or more strains, and a "cloud pangenome" that contains genes only found in a single strain [13–15]. Pangenome

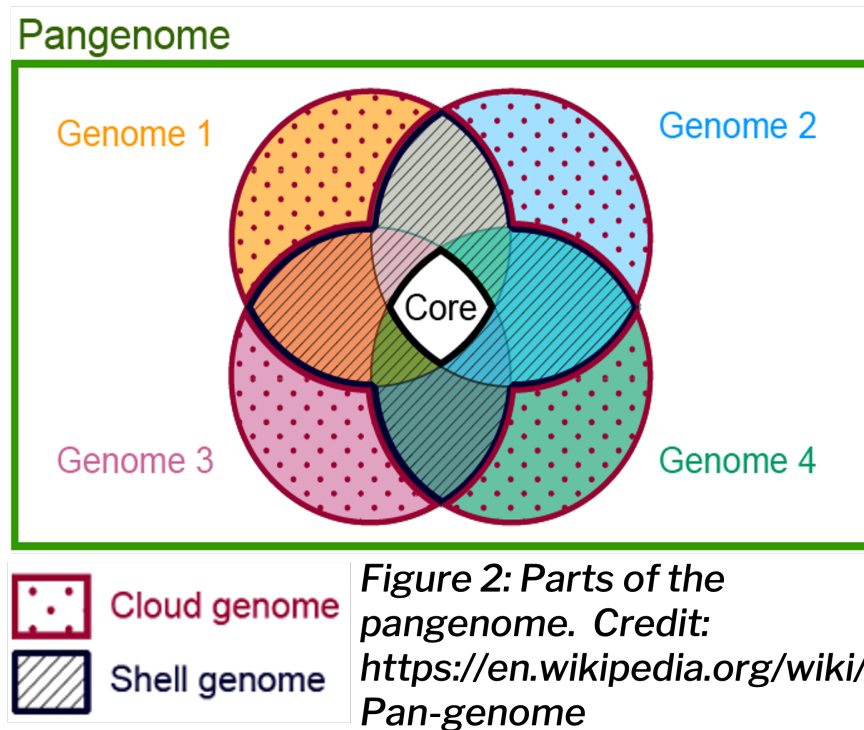


Figure 3.1: Parts of the pangenome.

analysis helps in understanding the genetic determinants of biological activity. We use PPanGGOLiN [16], a Free Software suite used to create and manipulate prokaryotic pangenomes from a set of genomic DNA sequences to perform pangenome analysis. This pipeline comprises four essential stages. The first stage involves annotation, where genes are labeled and categorized. The subsequent stage focuses on clustering, where genes with similar characteristics are grouped together to build gene families. Following clustering, the pipeline proceeds to the graph stage, where nodes represent these gene families, and edges depict their genomic neighborhoods. Finally, gene

§3.1 Pangenome Analysis

families are allocated to specific partitions in the partitioning stage, distinguishing them as 'persistent,' 'shell,' or 'cloud' based on their prevalence and occurrence patterns.

We run the ppangolin workflow by providing a tsv-separated file, a list with the first column being a unique genome name and the second column being its path to the associated FASTA file (compressed fna file). The PPanGGOLiN analysis on 3000 genomes was run using the NERSC Perlmutter HPC system. Around 450GB of memory was utilized, taking 13 hours to complete. PPanGGOLiN provides multiple outputs to describe a pangenome. The output yields four primary files/folders, each providing crucial information as shown in Fig. 3.2:

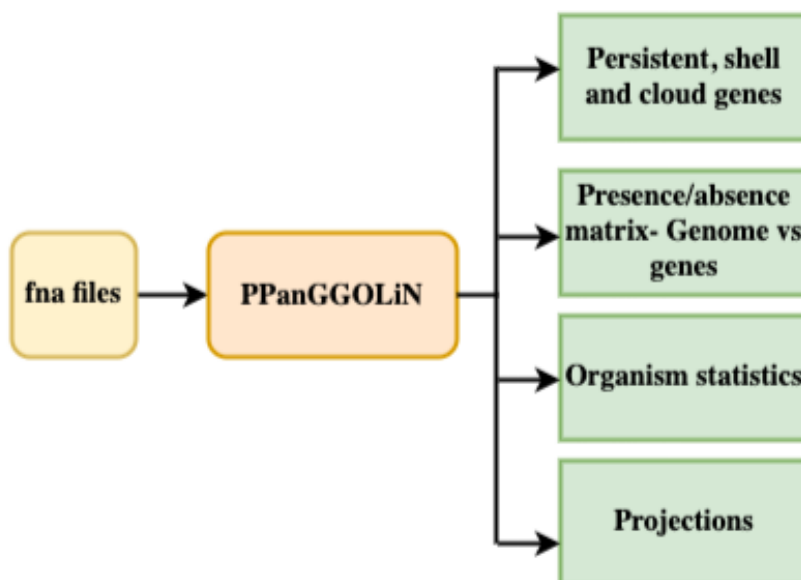


Figure 3.2: Parts of the pangenome.

- Lists categorizing genes as 'persistent,' 'shell,' or 'cloud' based on their prevalence and occurrence patterns.

Chapter 3 Methods

- A presence/absence matrix detailing the presence or absence of genes across different genomes.
- Organism statistics containing essential details such as the organism's name associated with the provided genome, the count of gene families present in that genome, etc.
- Projections displaying gene-related information, including the gene identifier, contig details, and the start and end positions of the genes.

We use the core, shell, and cloud genes lists from the output partition folder and `gene_presence_absence.Rtab` matrix file for the current study.

§ 3.2 Data Input

The input data for the GAN model consists of a presence-absence matrix extracted from the `gene_presence_absence.Rtab` file generated by PPanGGOLiN. This matrix is a two-dimensional array where the columns represent the genomes used in constructing the pangenome, and the rows correspond to gene families. A value of 1 indicates the presence of a gene family in a genome, while 0 signifies its absence.

Our focus in this analysis is on identifying genes that are present in two or more genomes. To achieve this, we filter the presence-absence matrix to include only:

- Core genes: Genes that are present in all genomes.
- Shell genes: Genes present in 10-95% of the genomes.

Genes with less than 10% occurrence, known as cloud genes, are excluded from the analysis. The data presented to the GAN model in this matrix can be described as a 2D tensor, where one dimension represents genomes, and the other represents the presence or absence of their corresponding genes. This 2D tensor serves as the input to the GAN model, which is responsible for generating new gene presence-absence matrices that share similar characteristics to the original data.


| | | org1 | org2 | org3 | org4 | |
|---|---|------|------|------|------|---|
| 1 |  | 1 | 1 | 1 | 1 | |
| . |  | 1 | 1 | 1 | 1 | |
| . |  | 1 | 1 | 1 | 1 | |
| . |  | 1 | 1 | 1 | 1 | |
| . |  | 1 | 1 | 1 | 0 | |
| i |  | 0 | 1 | 1 | 1 | |
| . |  | 0 | 0 | 1 | 1 | |
| . |  | 0 | 0 | 1 | 1 | |
| . |  | 0 | 1 | 0 | 0 | |
| F |  | 0 | 1 | 0 | 0 | |
| | | 1 | . | j | . | N |

Figure 3.3: Gene Presence/Absence matrix

§ 3.3 Model

To build the generative model, we use the Wasserstein GAN-Gradient Penalty method. Generative adversarial networks (GANs) [5] are a powerful class of generative modeling subjectively regarded as producing better samples than other methods [17]. GANs have two neural networks playing against each other where one, the generator, learns to generate reasonable data as training, and the other, the discriminator, learns to distinguish the generator's fake data from real data. The generated instances become negative training examples for the discriminator. Instead of using a discriminator to classify or predict the probability of generated data as being real or fake, we use WGAN, which changes the discriminator model with a critic that scores the realness or fakeness of a given data using Wasserstein loss [9]. The WGAN gradient penalty method [18] is used in order to ensure model convergence and stability.

3.3.1 Generative Model Architecture

The network architecture for our GAN model is designed to handle large matrix data, following recommendations from previous studies [19]. The model consists of a sequence of densely connected layers, each with 2^{14} neurons. This network size is approximately 70% of the input gene count (22843 core and shell genes). The generator input latent vector of size 2^{12} shown in Fig. 3.4.

3.3.2 Model Training

The input dataset is a gene presence-absence matrix comprising 3000 genomes and 22843 core and shell genes. We train the model on 80% of the dataset randomly

selected, using a batch size of 64. The Adam optimizer with a learning rate of 0.0002, beta_1 of 0.5, and beta_2 of 0.9 is utilized.

To ensure the stability of the training process, we update the critic model three times more frequently than the generator model. This approach is consistent with the recommendations for training WGAN models. The model is trained for 50 epochs on AMD EPYC 7763 CPUs.

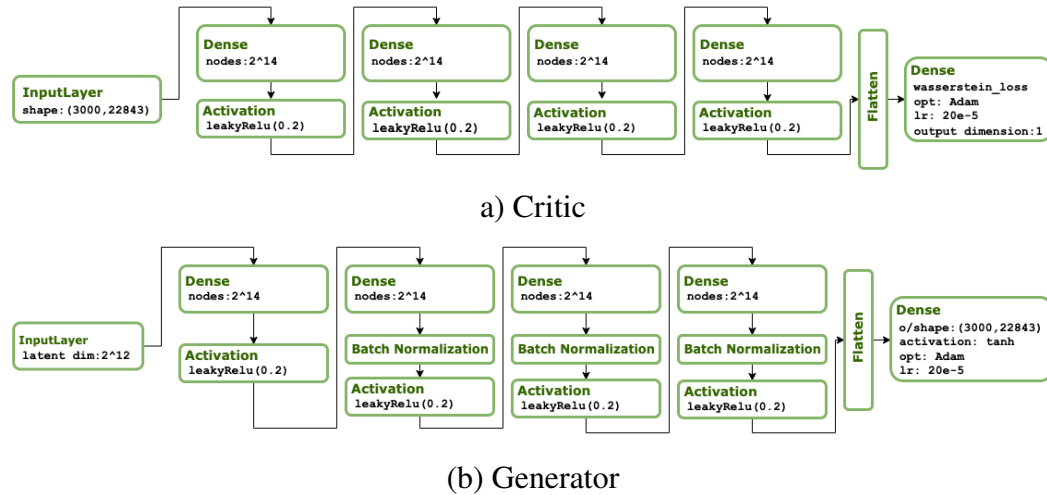


Figure 3.4: WGAN Model for Pseudomonad genomes

Chapter 4

Results

To evaluate the performance of the models, we first test the generator for the percentage of core genes present in their gene presence-absence lists. These gene presence-absence lists of the genomes have a median of 73% of the required core genes, which can be shown in a clustermap in Fig. 4.1, with the number of genomes on the y-axis and core genes on the x-axis.

Next, we test the critic model on previously unseen test genomes, constituting the 20% data split from before training. In this evaluation, the critic model predicted 598 genomes as real among 600 test genomes, demonstrating a 99% accuracy. However, when subjected to synthetic incorrect genomes created by omitting core genes, the critic correctly identified only one out of the four incorrect genomes. This result on the incorrect genomes is evident as the critic was trained only on the fake data generated by the generator but was not pre-trained with the fake data from the input source.

Hence, we plan to further train the model with a combination of real and incorrect genomes and test the critic on a larger number of incorrect genomes. These observations show that the model has performed fairly well on generating gene presence-absence

matrices with 73% accuracy, and the critic has identified 99% of the real genomes.

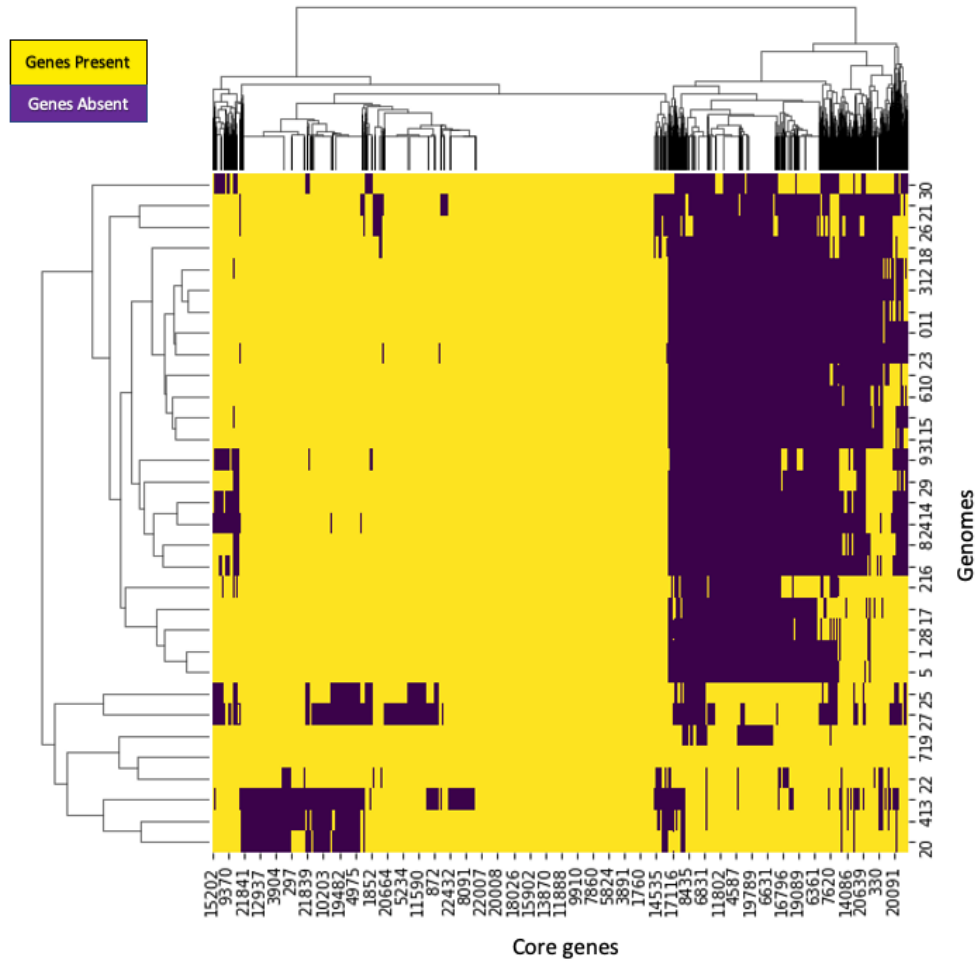


Figure 4.1: Generator output evaluation

Chapter 5

Future Works

The current work serves as an initial step in developing a model for generating pseudomonad genomes, with the potential for various applications of both the discriminator and generator models. Several planned directions for extending and utilizing the current model include:

- Extending the dataset to 9000 genomes: Initially, 12,000 Pseudomonad genomes were downloaded from NCBI, and around 9000 error-free genomes were identified for analysis. The plan is to extend the analysis to all 9000 genomes, providing a more comprehensive dataset for the model.
- Real/Fake genome identification: Utilizing the trained discriminator model as a binary classifier to distinguish between real and fake genomes. This process is crucial for ensuring if a genome has all the required genes.
- Pretraining the model with synthetic data: Currently trained with all real data, the model will be enhanced by incorporating fake data during pretraining. This involves generating synthetic genomes by removing the required core genes,

creating a more diverse training set.

- Finding missing genes and gap filling: Leveraging the generator model to identify incomplete sequences and fill gaps by generating the necessary core genes. This step contributes to enhancing the completeness and accuracy of the generated genomes.

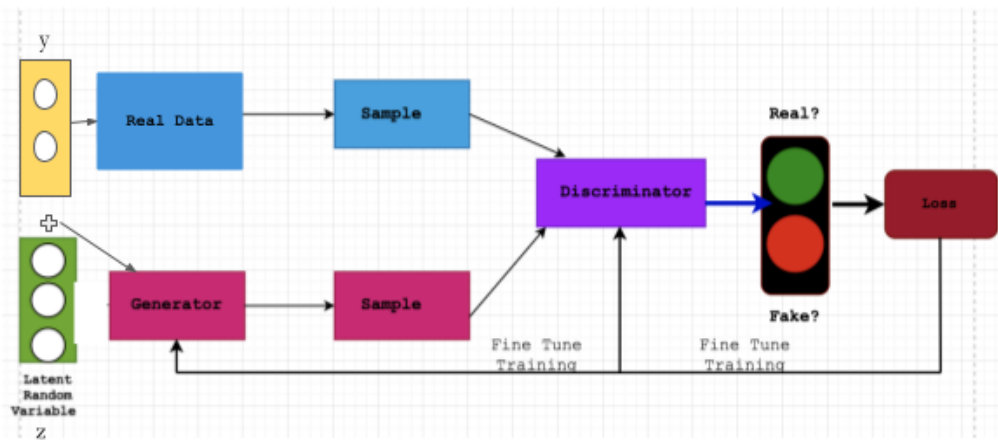


Figure 5.1: Conditional GAN

- CGAN: Gene position, function identification: Creating a Conditional Generative Adversarial Network (CGAN) utilizing both positional and functional characteristics: In this approach, the class is passed to the generator as a one-hot vector y combined with a noise vector Z as shown in Fig. 5.1. Simultaneously, the discriminator receives the class as a set of one-hot matrices, where the vector's size and the number of matrices reflect the quantity of classes. Ppangolin provides gene position information through a .tsv file in the Projections folder for each genome. In this file, genes are arranged so that the first line corresponds to

Chapter 5 Future Works

the gene at position 0 for a specific contig, the second line at position 1, and so on. This ordering aligns with the genes' start positions. Extracting gene position information from this file serves as a valuable input class for the CGAN.

- **Generate new genome with input parameters:** The ultimate goal is to create a comprehensive model capable of generating genomes with specified parameters. For example, generating a genome that is root-associated, drought-resistant, salt-tolerant, and capable of producing a specific natural product. This capability would provide a powerful tool for tailored genome synthesis.

In summary, the current work lays the groundwork for an evolving model with diverse applications, ranging from dataset expansion and authenticity verification to advanced capabilities such as conditional generation and parameter-based genome synthesis.

Chapter 6

Conclusion

In this work, we present three results: the calculated Pseudomonad bacteria pangenome, a generative model to identify incorrect genomes, and generated gene family presence/absence lists for artificial genomes. This research contributes to finding incomplete gene complements of low-quality genomes (e.g. metagenomically assembled genomes) as well as the study of gene functions and positional information by introducing a novel method for tackling this problem. Despite restricting the input features to only gene presence/absence, the model is apparently able to learn enough of the gene covariance structure to make artificial gene lists. The model can accurately predict real genomes, which has immediate applications in the assessment of genomes constructed from metagenomic sequencing (MAGs) which can often be incomplete or chimeric [20, 21]. Given the performance of our current model using only gene presence/absence, we believe that incorporation of additional biologically-significant gene features will significantly improve the model.

Ultimately, our goal is to create a model capable of generating genomes with desired characteristics, this will require augmenting our gene level features, such as functions

Chapter 6 Conclusion

and position; multi gene modules or operons and pathways; and organism level trait annotations, will make the generative model significantly more informative. However, even the most basic of these features, gene function and position, are not straightforward to assign, extract or encode into the feature tables. Gene functional annotation is imprecise and because bacterial genomes are often highly fragmented, even the ordering of genes on the chromosome is not known. Higher level features, such as organismal traits and traits that emerge under specific environmental interactions are very heterogeneous and incomplete from most genomes, which introduce additional challenges which may require development of new methods. Despite these challenges, we believe that this work represents a solid foundation to build towards designer genomes.

This research has been accepted at the Learning Meaningful Representations of Life Workshop during the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Bibliography

- [1] Mark W. Silby, Craig Winstanley, Scott A.C. Godfrey, Stuart B. Levy, and Robert W. Jackson, *Pseudomonas genomes: diverse and adaptable*, FEMS Microbiology Reviews **35** (201107), no. 4, 652–680.
- [2] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Bha, and Xiaolin Li, *Adversarial examples: Attacks and defenses for deep learning*, 2017.
- [3] Udway D.W. Otani H. and Mouncey N.J., *Comparative and pangenomic analysis of the genus streptomyces*, Scientific Reports **12** (2022).
- [4] N. J. Palleroni, *Pseudomonas*, Springer, New York, NY, 2010.
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 2014.
- [6] Alec Radford, Luke Metz, and Soumith Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, 2016.
- [7] Li Deng, *The mnist database of handwritten digit images for machine learning research*, IEEE Signal Processing Magazine **29** (2012), no. 6, 141–142.
- [8] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso A. Poggio, *Learning with a wasserstein loss*, CoRR **abs/1506.05439** (2015).
- [9] Martin Arjovsky, Soumith Chintala, and Léon Bottou, *Wasserstein generative adversarial networks*, Proceedings of the 34th international conference on machine learning, 2017.
- [10] National Center for Biotechnology Information Bethesda (MD): National Library of Medicine (US), *National center for biotechnology information (ncbi)* (1988).
- [11] Donovan Parks, Maria Chuvochina, Christian Rinke, Aaron Mussig, Pierre-Alain Chaumeil, and Hugenholtz Philip, *Gtdb: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy*, Nucleic Acids Research **50** (2021).
- [12] Arkin AP et al., *Kbase: The united states department of energy systems biology knowledgebase.*, Nature Biotechnology (2018).
- [13] Medini D et al., *The microbial pan-genome*, Current Opinion in Genetics and Development **15** (2005).
- [14] Vernikos G et al., *Ten years of pan-genome analyses*, Current Opinion in Microbiology **23** (2015), 148–154.
- [15] Wolf Y.I et al., *Updated clusters of orthologous genes for archaea: a complex ancestor of the archaea and the byways of horizontal gene transfer.*, Biology direct **7 46**. (2012).

BIBLIOGRAPHY

- [16] Gautreau G et al., *Ppangolin: Depicting microbial diversity via a partitioned pangenome graph.*, PLOS Computational Biology 16(3): e1007732. (2021).
- [17] Ian J. Goodfellow, *Nips 2016 tutorial: Generative adversarial networks*, ArXiv **abs/1701.00160** (2017).
- [18] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, *Improved training of wasserstein gans*, Advances in neural information processing systems, 2017.
- [19] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao, *Improving the speed of neural networks on cpus*, Deep learning and unsupervised feature learning workshop, nips 2011, 2011.
- [20] Mobberley JM. Nelson WC Tully BJ, *Biases in genome reconstruction from metagenomic data*, PeerJ (2020).
- [21] Chivian D. et al., *Metagenome-assembled genome extraction and analysis from microbiomes using kbase*, Nat Protoc (2022).

Bibliography