Decoding DNA Images for Identification of Infections

Shelly Meir, *Technion,* Roi Mahfud, *Technion,* Doron Lisiansky, *Technion,*

Abstract—Our team developed a bacterial classifier using Single-molecule DNA mapping. This method makes sequencing in metagenomic analysis more efficient and life saver. By offering longer read lengths and forgoing the need for complex library preparation and amplification. In order to achieve high accuracy by using our method, we need to improve sensitivity of DNA map analysis. We will present 2 main methods in addition to detailed simulations and experimental data, for the experiments we use images that are taken from the lab using fluorescence imaging of surface stretched, sequence specifically labeled DNA fragments. We first demonstrate how we use cross correlation algorithm to identify the bacteria. Second, a technique that based on images classifier using convolutional neural networks to categorize images into their corresponding genome. Finally, we demonstrate the capability and the limitations of each method to identifying species with long genomes such as bacteria with high sensitivity.

1 Introduction

Bacterial infections resistant to antibiotics are a global healthcare challenge. It takes a few days to diagnose the exact type of bacteria infecting the patient. Meanwhile, a random treatment is given, which can be ineffective in many cases. During those long days waiting for the lab result, many people die, especially in the case of the dangerous bloodstream infections.

Currently, The lab tests for the type of bacteria take a long time, they grow the bacteria to a large enough amount, so that the existing lab equipment can identify them.

In our research, we develop a new, more sensitive method than the existing ones, allowing the immediate identification of the bacteria without the need for growing them beforehand.

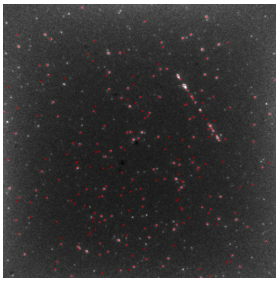


Fig. 1. DNA dyed fragments

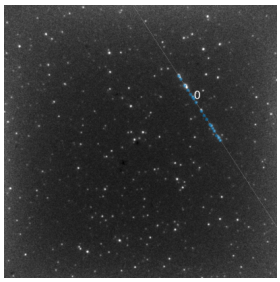
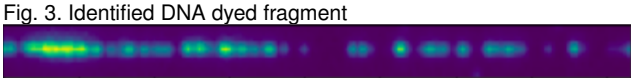


Fig. 2. DNA dyed marked fragment

Fig. 3. Identified DNA dyed fragment



2 Project dictionary

References

[1] Segment - A set of frames of a broken DNA strand of a bacteria.
[2] Genome - DNA sequence that describes a pattern of gens (we refer to genomes of the bacteria)
[3] Start offset - The genome offset where new segment begins
[4] Labeling efficiency - measurement of the enzyme ability to mask sequences (TCGA sequences in our case)
[5] Strech factor - Conversion rate from pixels to base pairs.
[6] Gain factor – The ratio between the digital level and between fluorophores/pixel.
[7] Dynamic range - ratio between the largest and smallest values that a certain quantity can assume.
[8] integration time - the time during which we hold the voltage signal as it maximizes and stabilizes so we can measure it

3 The Data

We used DNA molecule images with the ground truth of the corresponding genome sequence. The images are extracted from bacterial DNA fragments in the lab. We are using fluorescence imaging of surface stretched, sequence specifically labeled DNA fragments which basically staining DNA fragments with a special dye that marks specific short patterns like TCGA, stretching the DNA on a special surface, and finally the DNA fragments are imaged with a microscope.

In Fig. 1 we can see an example of DNA dyed stretched fragments. From this kind of images, each DNA fragment is identified as shown in Fig. 2 and 3. We will be working with 3d arrays of fragment images where the 3rd dimension is time.

4 Implementation

4.1 Classifier based on Cross-Correlation

Our first classifier uses Pearson cross-correlation to measure the distance between 2 sequences. For each segment Pearson cross-correlation looks for the best match in each bacterial genome. In order to simulate the real physical phenomenon and test the method with high significance to the real life, we have implemented functions to simulate the labeling efficiency of the enzyme, simulate the gain factor, integration time and the dynamic range of the camera, stretch factor is used to normalize the picture pixels to base pairs scale. Our solution is implemented in Python, and the main API will be presented .

4.1.1 Why Pearson?

In order to understand why Pearson's correlation was chosen we have to understand the Pearson's advantages and in which conditions it is the most accurate. Pearson's correlation coefficient is the test statistics that measures the statistical relationship, or association, between two continuous variables. It is known as the best method of measuring the association between variables of interest because it is based on the method of covariance. It gives information about the magnitude of the association, or correlation, as well as the direction of the relationship

The formula for Pearson correlation can be written as :

$$\rho = \frac{cov(X,Y)}{\sigma_x \sigma_y}$$

where:
cov is the covariance
σ_X is the standard deviation of X
σ_y is the standard deviation of Y

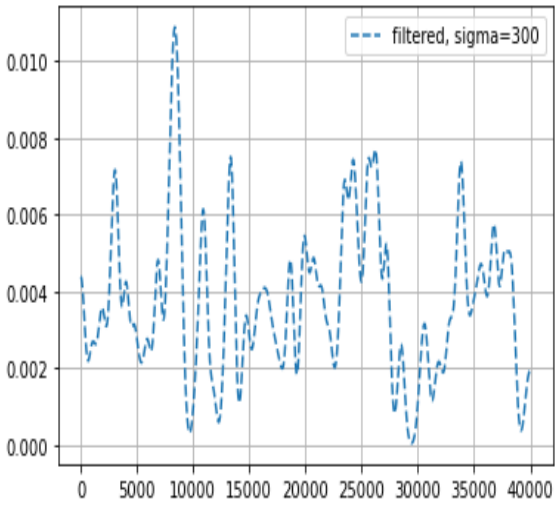
4.1.2 Main API

The main API we are using:

extract genomes(genomes directory):
Used to extract the bacterial images and corresponding genome .

calc gaussian1d without plot(masked genome):
applying gaussian filter on the masked genome .

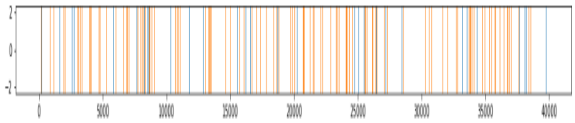
graph of the gaussian signal



segment creation(genome sequence) :
Used to create new random placed segment from genome.

remove labels(segment, remove labels rate) :
Used to remove marked patterns from the segment (trying to mimic the physical phenomenon of masking the genome by the enzyme - labeling efficiency of the enzyme).

blue labels are removed from original sequence



labeling efficiency example

scale expected(masked genome, convertion rate) :
Used to convert the genome sequence (bp) to same scale as the pictures (pixels) - Stretch factor .

max args with correlation Pearson(measured, expected):
applying the Pearson correlation between the measured picture (bacterial segment/simulated segment) and the expected genome.

4.2 Classifier based on Deep Learning

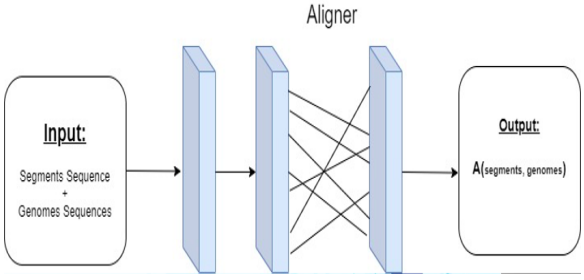
Our second attempt of implementation consists of 3 sub-models; Video Analyzer, Aligner and Matcher.

4.2.1 Video Analyzer

Video Analyzer is meant to receive raw data – an array of frames of genome fragments – which is a 3d array, and to output a 1d array which is the segment sequence. In order to do that we wanted to use a neural network which will be able to learn how to correctly reduce the dimensions of the given segment video. Due to lack of time we didn't manage to fully implement this model and so instead we used mean values of the 3d array in order to reduce its dimensions from 3 to 1.

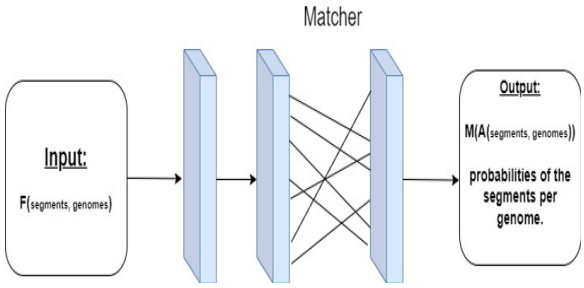
4.2.2 Aligner

The Aligner receives the Video Analyzer's output as an input, meaning a segment sequence, and a genome sequence. Its purpose is to output an alignment matrix of the given segment and genome, which then will be used by the Matcher in order to determine whether the given segment matches the given genome. This means that for each segment and for each genome, (segment, genome) pair will be given as an input to the Aligner. We implemented the Aligner using a U-Net structure with DiceLoss as a loss function and trained it on batches of 8 tuples of segment, genome and alignment matrix.



4.2.3 Matcher

As explained in the previous section, the Matcher receives the Aligner's output as an input, which is the alignment matrix of a given segment and genome. The Matcher then outputs the probability the given segment belongs to the given genome. To implement the Matcher we used a convolutional neural network which we trained on a dataset of pairs of alignment matrices and probability values, with SparseCategoricalCrossentropy loss function.



5 Experiments 1st method

5.1 Experiment Data

our data for this experiment consist of 3 main groups :
1) random genomes - created by python function
2) real genomes from ncbi - genomes taken from ncbi site
3) real genomes from our laboratory - lambda virus and ecoli genomes and their corresponding pictures that taken from in the laboratory

5.2 Experiment Work Flow

5.2.1 Extracting the data

first we are extracting the data and the pictures of the ecoli and lambda virus, we are creating n random genomes for testing and saving the genomes from ncbi.

5.2.2 creating simulated data

in order to simulate pictures for the ncbi genomes and random genomes we are creating new segments in 40kbp length from the corresponding genome, we are removing k percent of the labeled data(the 1's in the segment) to simulate the enzyme imperfection, next step is adding gain factor of 10k-30k and adding some gaussian noise to the data.

5.3 Testing function

compare labeling efficiency(random genome list, real genome list, real genomes data, real genomes segmented arr, labeling deletion percentage arr, metric function, noarmalization norm,add gain, convertion rate)

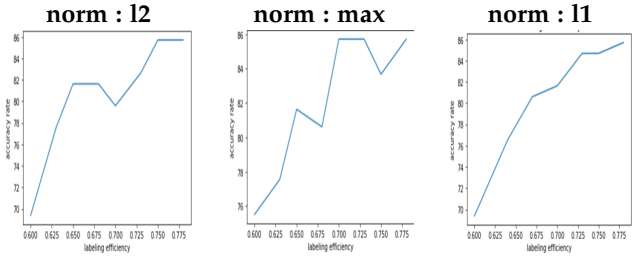
parameters :
-random genome list - list of created random genomes
-real genome list - list of genomes from ncbi
-real genomes data - ecoli and lambda genomes
-real genomes segmented arr - list of pictures list from each genome
-labeling deletion percentage arr- list of percentage values for labeling efficiency
-metric function - the metric function for comparing 2 density signals
-convertion rate - value of the strech factor

6 Results 1st method

in our experiment we are testing for each simulated/real segment/picture if the testing function is able to identify the correct genome which the corresponding segment came from, dataset contains off 10 real genomes, 2 are lambda and ecoli which we have extracted pictures of them from the lab.
in our results we will present different types of normalization to the data : l2,l1,max
different strech factors : 295 - 315
all of the graphs are the result of using pearson correlation and finding the genome that matching the most for specific segment for each of these parameters we are applying labeling efficiency with decreasing percentage rate and plotting the accuracy rate against the labeling efficiency percentage

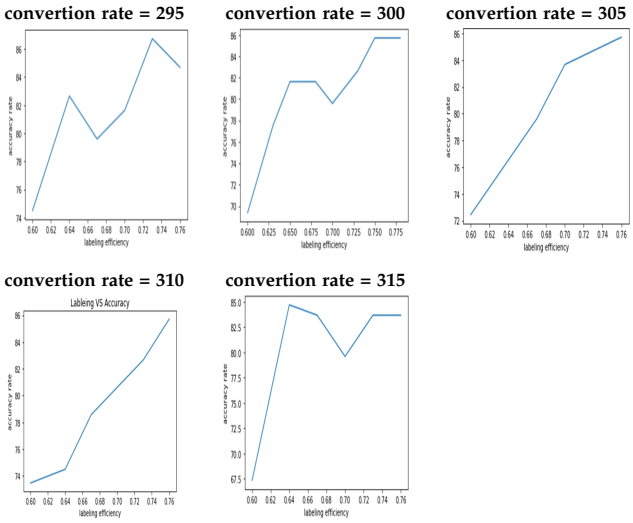
6.1 Testing Different labeling efficiency

metric function : max args with correlation pearson
convertion rate : 300
labeling efficiency values : 0.78, 0.75, 0.73, 0.7,0.68, 0.65,0.63, 0.6



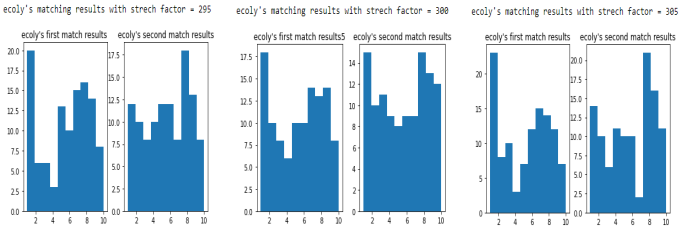
6.2 Testing Different convertion rate

metric function : max args with correlation pearson
noarmalization norm : l2
convertion rate : 295, 300 , 305 , 310 , 315
labeling efficiency values : 0.78, 0.75, 0.73, 0.7,0.68, 0.65,0.63, 0.6

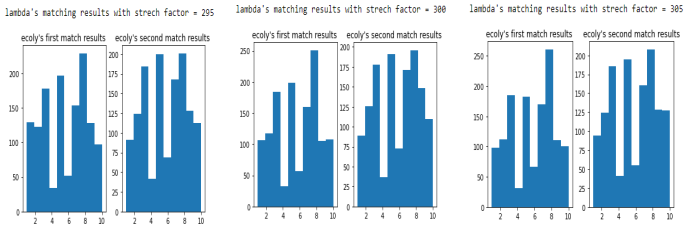


6.3 results on the real data

6.3.1 Ecoly main Results

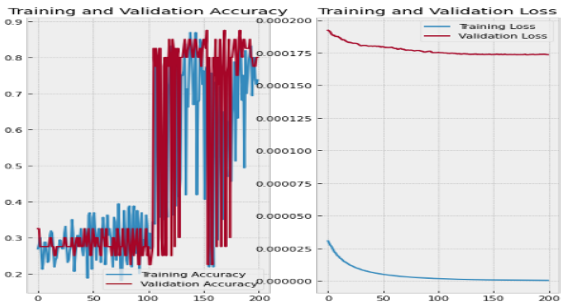


6.3.2 lambda main Results



7 Results 2nd method

7.1 Matcher Training Results



8 Discussion and Conclusion

In our project, we implemented an improved method for identifying bacterias based on single-molecule double-stranded DNA maps. We used intensity profiles extracted from microscopy images that we got from the lab. The profiles can be compared to a couple of referenced genome by generating an expected map and cross-correlating it with the measured map, then pick the referenced genome with the best correlation. To test our classifier's performance we ran several experiments that show the influence of different parameters in the model, such as labeling efficiency and convertion rate.

In section 6.1 we observe that the higher the labeling efficiency is, the better our model manages to classify the segments. This behaviour makes sense since higher labeling efficiency means the data we collected reflects the real data better. We notice that the graphs aren't monotone, which could be due to the fact that for each labeling efficiency value we created a new random genome.

On the real data we see that the results are not as good as they are on simulated data. Ecoli genome is almost 100 times longer than our simulated genomes with which we tested it in section 6.3.1. The first column in the graphs is the one referring to Ecoli genome, and we can see that it got the highest amount of matching segments, although each genome should have gotten the same amount. We suspect it happened due to Ecoli's long genome compared to the other genomes. Lambda genome on the other hand is roughly the same size as our simulated genomes, and in section 6.3.2 we see that the behaviour we experienced in the previous section doesn't occur here. Yet it is clear that the results on the real data are still not as satisfying as we would wish. This lead us to the second method.

In the second method we used Deep Learning in order to improve our classifier. We split our classification task into 3 minor tasks: data processing, genome-segment aligning, and scoring. First we want to process the raw data into a more simplified data with only the information we need, we do that with the Video Analyzer model. Then we compare each segment with each genome to get an alignment matrix, with Aligner model. Finally, with Matcher model we give score to each comparison of a given segment to all genomes. By using the 3 models, one by one, we get a classifier. Unfortunately we didn't manage to run tests with our improved model as the training process took too long.

9 Future Work

We suspect that our idea of an improved model using Deep Learning will perform better than our first method, and better than the methods hospitals and labs use nowadays. Thus, our suggestion for future work is to continue and delve in to this idea, perhaps change the 3 models to 2 models where one would manage the data processing and the other would do the classification task.

References

[1] Arno Bouwens, Jochem Deen, Raffaele Vitale, Identifying microbial species by single-molecule DNA optical mapping and resampling statistics NAR Genomics and Bioinformatics, Volume 2, Issue 1, March 2020, lqz007.