# Classification of bacterial motility using machine learning

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Yue Ma

August 2020

# Acknowledgements

I would first like to thank my mentor and adviser Dr. Hong Guo for allowing and supporting me to work on this interesting project. Secondly, I want to express my gratitude to my other committee members, Drs. Tongye Shen, Tian Hong, Michael A. Langston, and Haileab Tesfe Hilafu who have been extremely helpful with my research. I would also like to thank Drs. Gladys Alexandre, Ping Qian, and Barry Bruce for providing constructive advices on my research. I would also like to give special thanks to Drs Haobo Guo, Yufei Yue, and Mr. Hao Deng, and they have given me a lot of help on my research among many other things. I would like to acknowledge LaShel Brown for giving me encouragement throughout my stay at UTK.

# Abstract

Cells can display a diverse set of motility behaviors, and these behaviors may reflect a cell's functional state. Automated, and accurate cell motility analysis is essential to cell studies where the analysis of motility pattern is required. The results of such analysis can be used for diagnostic or curative decisions. Deep learning area has made astonishing progresses in the past several years. For computer vision tasks, different convolutional neural networks (CNN) and optimizers have been proposed to fix some problems. For time sequence data, recurrent neural networks (RNN) have been widely used.

This project leveraged on these recent advances to find the proper neural network for bacterial motility trajectory analysis for genotypic classification. This thesis was trying to answer two questions: (1) Which machine learning model can effectively classify the genotype of bacterial based on their motility patterns? And (2) Which motility parameters can best predict the bacterial genotype? The first question is addressed in the result 1 and 2 using different data formats and different machine learning models. The second question is addressed in result 3. Accordingly, this thesis is divided into three parts: (1) different traditional machine learning models are tested for predicting bacterial genotype using the coordinates' sequences extracted from microscopic videos. (2) bacterial genotype classification task is solved by using deep neural network and the raw videos. (3) different motility parameters are tested to find out the best for predicting bacterial genotypes.

It is found that neural network gives highest accuracy in classifying bacterial genotype using coordinates' sequences. Deep neural network with CNN-RNN can effectively classify the bacterial genotype using video data. Among popular motility parameters, some of them predict the bacterial genotype 20% better than others. The broader impact of this project is to automate trajectory analysis process and enable high-throughput trajectory analysis for research and clinical uses.

# Table of Contents


Background ..... 1
Technical Approach ..... 7
Data Preparation ..... 7
MPP classification ..... 8
Classifier Fusion ..... 9
k-NN classification ..... 10
k-Means Classification ..... 10
Decision Tree classification ..... 10
SVM classification ..... 11
Neural Network and Back Propagation classification ..... 11
Convolutional Neural Network ..... 14
CNN-RNN ..... 15
3D CNN ..... 16
Testing and validation accuracy ..... 16
Results ..... 17
Result 1: movement coordinates can be used to classify bacterial motility. ..... 17
Result 2: movement videos can be used to classify bacterial motility. ..... 31
Result 3: some motility parameters can better classify bacterial motility. ..... 35
Discussion and Conclusion ..... 41
References ..... 43
Appendix ..... 47
Vita ..... 51

# List of Figures

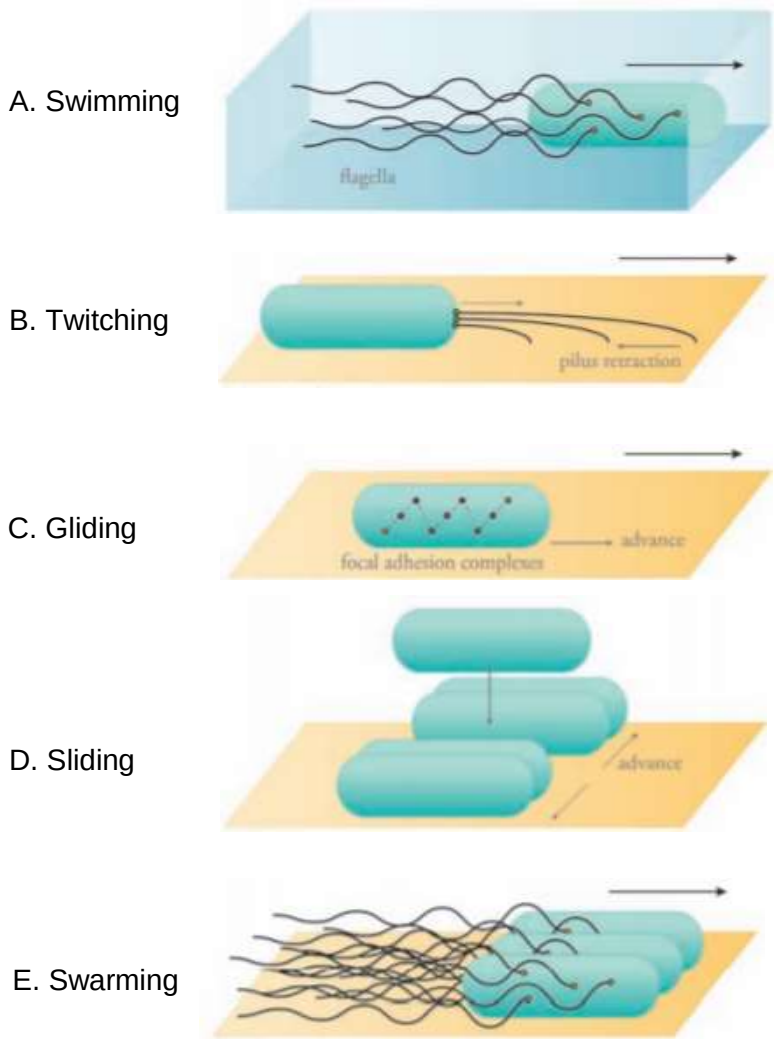
Figure 1 Mechanisms of bacteria motility ..... 2
Figure 2 Flagellum and Flagellation ..... 3
Figure 3 Swimming trajectories. ..... 4
Figure 4 Comparison of prokaryote and eukaryote ..... 6
Figure 5 An example of Classifier Fusion Table. ..... 9
Figure 6 Recurrent Neural Network. ..... 13
Figure 7 The repeating module in LSTM. ..... 13
Figure 8 Convolutional Neural Network. ..... 14
Figure 9 CNN-RNN mode for video classification. ..... 15
Figure 10 3D CNN for video classification. ..... 16
Figure 11 Feature distribution after ICA. ..... 17
Figure 12 Feature distribution before and after PCA. ..... 18
Figure 13 Prediction accuracy of each class using MPP. ..... 20
Figure 14 Prediction accuracy of each class with 10-fold cross validation. ..... 20
Figure 15 Prediction accuracy of fused classifiers. ..... 21
Figure 16 Prediction accuracy of k-NN. ..... 23
Figure 17 Prediction accuracy of k-Means. ..... 24
Figure 18 Prediction accuracy of decision trees. ..... 25
Figure 19 Prediction accuracy of single neural network. ..... 27
Figure 20 Prediction accuracy of support vector machine. ..... 28
Figure 21 Overall accuracy of different models using coordinate features. ..... 30
Figure 22 Prediction accuracies without and without transfer learning. ..... 32
Figure 23 Accuracy of CNN-RNN model with transfer learning. ..... 33
Figure 24 Class-specific accuracies of CNN-RNN model. ..... 34
Figure 25 Neural network architectures used for raw data. ..... 36
Figure 26 Prediction accuracies of 5-class classification task. ..... 37
Figure 27 Average correlation of each motility parameter. ..... 38
Figure 28 Best features for genotype prediction. ..... 39
Figure 29 Criteria of Events ..... 50

## Background

Motility is the ability of an organism to move independently, using metabolic energy, in contrast of mobility, the ability of an object to be moved. Bacterial motility is central to many biological functions, such as intestinal digestive processes[1], spreading of infections[2-3], photosynthesis activities[4], or biofilm formation[5]. For example, studies of bacterial motility in polymer solution is of medical interest in understanding the penetration process of mucus in our body by bacteria pathogens[6]. The motility behaviors of cancer cells in culture are used to evaluate the progression of tumor[7]. Therefore, automated, and accurate cell motility analysis is essential to cell studies where the analysis of motility patterns is necessary.

Cells can display various motility behaviors. Such behaviors include the moving from one location to another during embryonic development, migrating into a wound for healing, and moving to approach nutrients to avoid harmful substances. As is shown in figure 1, bacteria motility can be classified by the motility mechanisms such as swimming, swarming, gliding, twitching, sliding, and swarming[8]. Most bacteria move using flagella. Flagellation refers to the number and distribution of the flagella across the cell body, which decide the motility pattern of the bacteria. As is shown in figure 2, there are six categories of flagellation[9]: 1. Atrichous: no flagella. 2. Monotrichous, where a single flagellum is located at one end of the cell. 3. Amphitrichous, where a single flagellum at both ends. 4. Lophotrichous, where a cluster of flagella is located at one end of the cell. 5. Peritrichous, where there is a uniform distribution of flagella across the cell body. Bacterial flagella rotate up to 1700 Hz, 5 times faster than a formula-one racecar engine[10]. The structure and function of many, but not all, flagellar proteins are known. The overall structure can be divided into two substructures: a hook-basal-body and the external filament[11]. Besides the protein components, the mechanism of torque generation is also a question that has been asked by researchers in search for the answer.

The bacterial strain used in this project is Azospirillum brasilense. A. brasilense are soil bacteria that are widely used in agricultural practices to help colonize the roots of plants and to enhance plant growth. A. brasilense cells swim using a single polar flagellum for swimming[12]. They can change swimming direction by switching the direction of flagellar rotation. Bacterial mobility is decided by a single factor: the rotation of motor. Changes in speed and direction of the motor rotation can lead to different types of movement events such as reversal, pause, speed change [13]. As is shown in figure 3, these movement events can be characterized by the patterns in velocity and angular velocity.

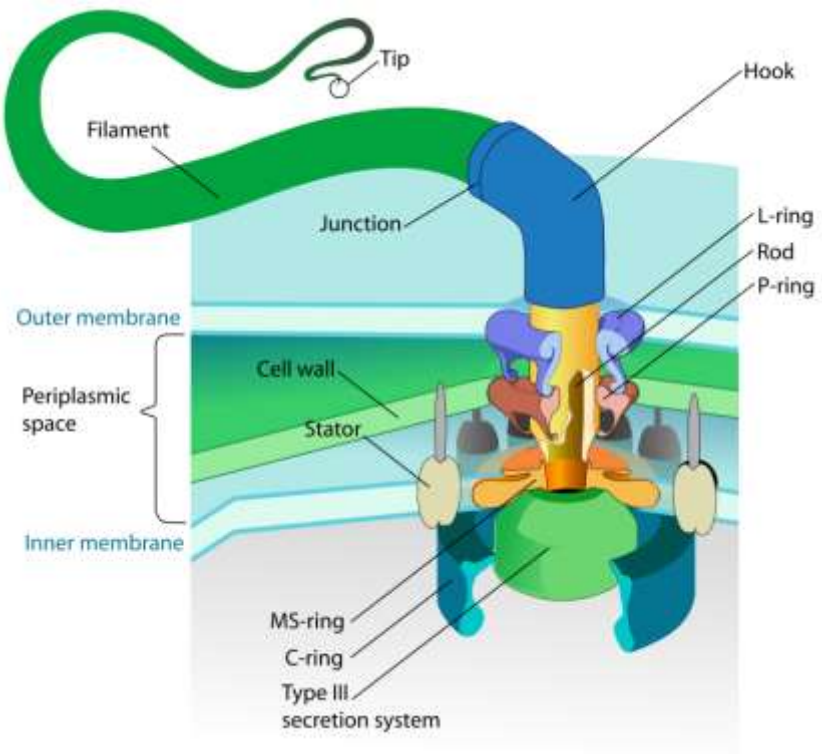



| Type | Motive Organelles | Cell diff. | $\mu$m/s | Environment | Movement |
|---|---|---|---|---|---|
| Swimming | Flagella | No | 25-160 | Liquid | Active |
| Twitching | Type IV pili | No | 0.06-0.3 | Surface | Active |
| Gliding | Unknown | No | 0.025-10 | Surface | Active |
| Sliding | None | No | 0.03-6 | Surface | Passive |
| Swarming | Flagella | Yes | 2-10 | Surface | Active |

Figure 1 Mechanisms of bacteria motility.

Tiny circles in (A-E) indicates the motors. Swimming and swarming are powered by flagella. Twitching is powered by pili. Gliding is active movement that requires focal-adhesion complexes. Sliding is passive translocation that is not powered by motor. They have different speed ranging from 0.025-160 um/s as shown in the table. This figure is adapted from [8]. The table is adapted from [40].

A




B

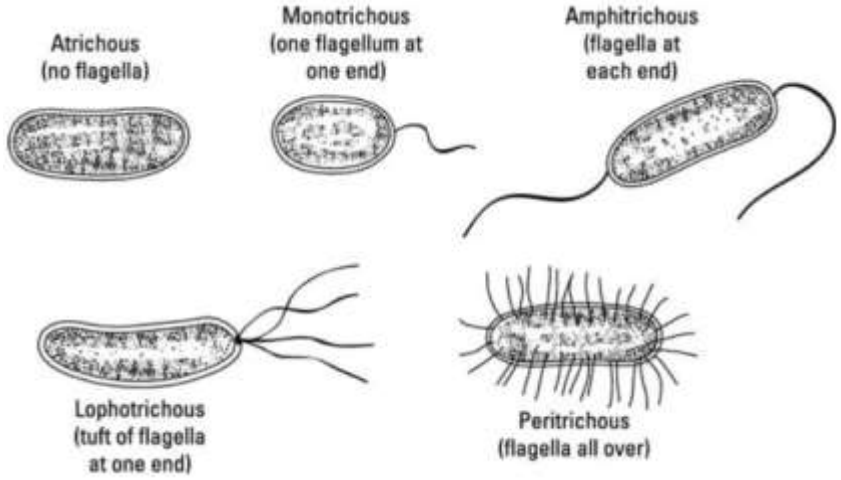



Figure 2 Flagellum and Flagellation

(A) Structure of bacterial flagellum. Flagellum is mainly composed of filament and HBB (Hook and basal body) and proteins joining them. This figure is adapted from [14][15] (B) Flagellation. Atrichous is when the cell has no flagellum. Monotrichous is when the cell has a single polar flagellum. Amphitrichous is when the cell has two flagella at each end. Lophotrichous is when the cell has a group of flagella at one end only. Peritrichous is when the cell has flagella all around. This figure is adapted from [16]

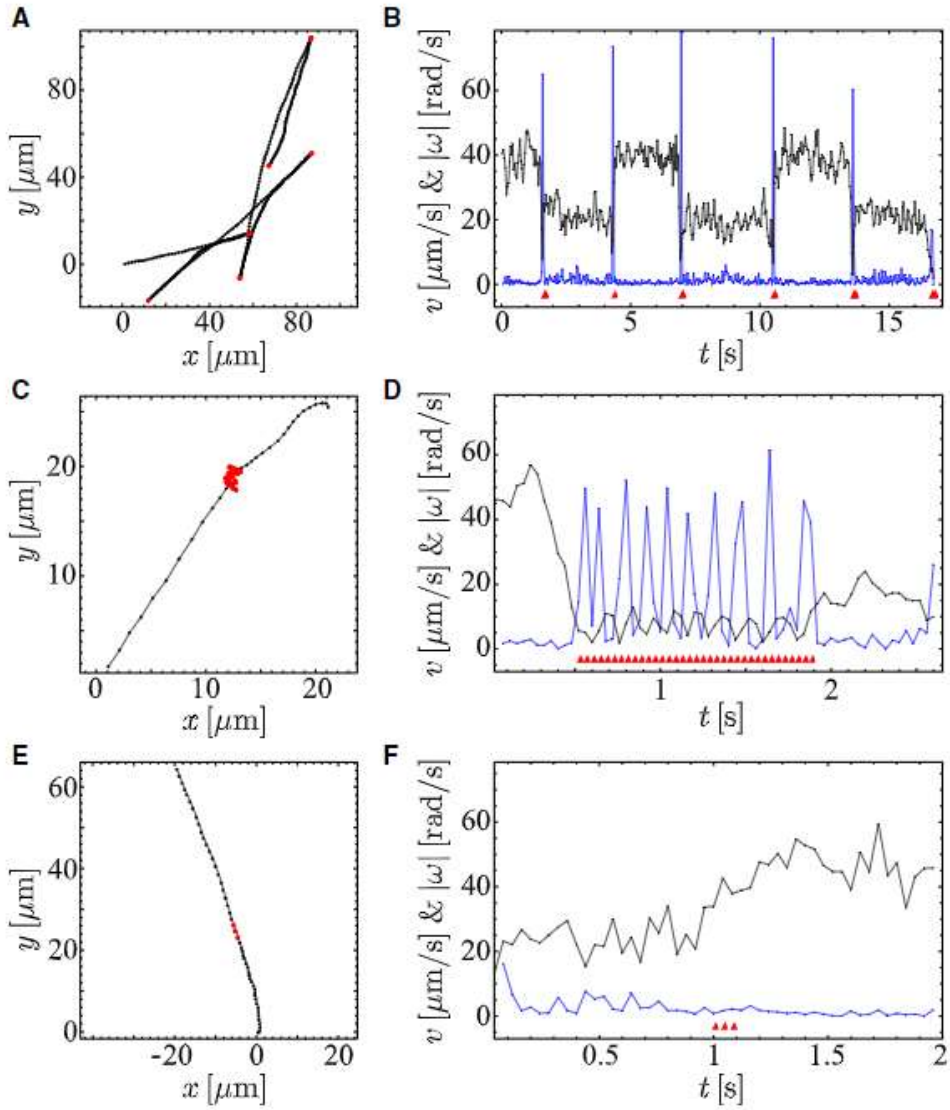



Figure 3 Swimming trajectories.

Swimming trajectories of P. putida with typical turning events (A) Trajectory with several successive reversals ($\phi_1 = 180^o$, positions of the reversals marked in red) (C) Trajectory with a typical pausing event ($\phi_2 = 0^o$, position marked in red) (E) Trajectory with a typical speed change event ($\phi_2 = 0^o$, position marked in red). (B,D,F) Speed (v, black) and absolute value of the angular velocity (|ω|, blue) over time for the trajectories (1, 3, 5), respectively. (Red triangles are time points of the corresponding events. This figure is adapted from [17]

The most computerized methods for motility behaviors analysis rely on manual feature engineering. Many tools designed by this principle give exciting performance. For example, neural progenitor cells were discriminated by morphology and motility behavior alone with 87% accuracy[18]. Tools such as CellTrack[19] and MotIW[20] are developed

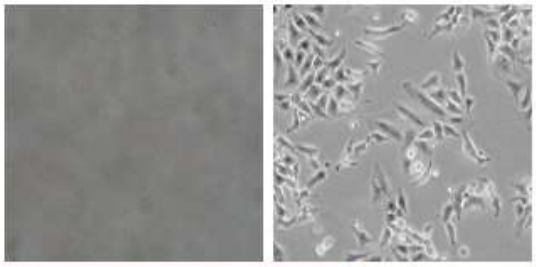
for tracking cells and analyzing the trajectories. These approaches follow the conventional paradigm of pattern recognition, which consists of two steps. The first step is feature extraction where parameters are selected based on domain knowledge. The second step is classifiers training where classifiers are trained based on the obtained features. However, in the real-world scenarios and most of the time, it is difficult to know what are the features that are most relevant for the task of interest or it is not clear how to select the right features that can be used to distinguish the cells.

Classification of bacterial motility behavior patterns without using any prior knowledge requires the training of low-level feature detectors. Deep learning machines are designed for these scenarios and have been shown to yield competitive performance. Deep learning models can learn hierarchies of features and build corresponding representations by themselves[21-23]. Despite their generic nature, the application of deep learning models was limited because of the existence of huge number of trainable parameters. This problem was mitigated by the invention of convolutional architecture from LeCun group in 1990s. The full connection between input and models are replaced by the use of filters whose values are forced to be shared across the entire input[24-25]. CNN has become the most popular model for computer vision tasks since then. Its accuracy surpassed human detection accuracy of 95% since 2015[26]. Videos are sequences of images. For sequential data analysis, recurrent neural networks (RNN) is the most well-known models due their ability to take advantage of the context information in temporal dimension. With loops in them, RNN allow information to persist. However, learning long-term dependencies is difficult because of the gradient vanishing problem in backpropagation. In order to solve this problem, Hochreiter and Schmidhuber proposed a structure called long short-term memory (LSTM). In standard RNNs, the repeating module is a single layer. LSTM replace this single layer by three parts: the input gate, the output gate and the forget gate. Each gate has their own weight and activation function to update cell and hidden state (that is, long-term memory and working memory) [27].

Many machine learning and deep learning methods have been used for cell mobility analysis such as cell tracking and event detection using support vector machine(SVM) [43], quantify cell trajectories in live cell imaging data [20], quantify state transitions to reveal progressive motility states using SVM [28]. While substantial success has been observed for the use with eukaryotes, it has been found to be challenging for bacterial imaging application due to several reasons as shown in figure 4. For example, the cell bodies of prokaryotes usually are much smaller than that of the eukaryotes but move much faster. Because of the fast movement, the field of vision needs to be much larger in order to have a good observation of prokaryotes movement. This leads to the use of relatively low-resolution microscope with lower contrast and more significant noises. Thus, bacterial movement is relatively harder to analyze. The velocity comparison between

eukaryotes and prokaryotes is listed in figure 4. Prokaryotic cells are significantly smaller than eukaryotic cells (0.1-0.5 um and 10-100 um for prokaryotic and eukaryotic cells, respectively). These natural features of prokaryotic and eukaryotic cells make motility analysis of bacterial cells harder than that of mammalian cells.

A



B

| organism | speed | speed in body lengths (bl) per sec | BNID and comments |
|---|---|---|---|
| | | bacteria and archaea | |
| *Ovobacter propellens* | 1000 μm/s | 200 bl/s | 111235 |
| *Thiovulum majus* | 600 μm/s | 90 bl/s | 107652, 111231, , cell length ≈7 μm |
| *Methanocaldococcus jannaschii* | 400 μm/s | 200 bl/s | 107649, measured at ≈80°C |
| *Bdellovibrio bacteriovorus* | 160 μm/s | 160 bl/s | 101969, has to catch other bacteria it preys on |
| *Vibrio cholerae* | 40-100 μm/s | 20-50 bl/s | 108083, sodium ion motor, one polar flagellum |
| *Caulobacter crescentus* | 40 μm/s | 20 bl/s | 108085, proton motor, one polar flagellum |
| *Spirochete Brachyspira hyodysenteriae* | 40 μm/s | 8 bl/s | 104904, assuming 5 μm cell length |
| *E. coli* | 16-30 μm/s | 8-15 bl/s | 101793, 106819, 108082, proton motor, 4-8 lateral flagella |
| *S. typhimurium* | 30 μm/s | 15 bl/s | 106818 |
| *Synechococcus* | 5-25 μm/s | 2-10 bl/s | 109314, mysterious propulsion by one third of wild isolates |
| *Myxococcus Xanthus motility system S* | >20 μm/min | >10 bl/min | 106811 |
| *Myxococcus Xanthus motility systemA* | 2-4 μm/min | 1-2 bl/min | 106811 |
| *Listeria monocytogenes* | 6 μm/min | 3 bl/min | 106823 *in vitro* motility assays |
| *Halobacterium halobium* | 2-3 μm/min | 1 bl/min | 111147 |
| | | eukaryotes | |
| *Ciliate Paramecium tetraurelia* | 100–1000 μm/s | 1-5 bl/sec | 108087, ciliated, assuming 200 μm cell length |
| *Tetrahymena thermophila* | 200–400 μm/s | 4–8 bl/sec | 111429, 111435, 111436, ciliated |
| *Gyrodinium dorsum* | 300 μm/s | 10 bl/sec | 111432, flagellated |
| green algae *Chlamydomonas Reinhardtii* | 50–150 μm/s | 5–15 bl/sec | 108086, 111430 |
| fish keratocytes - wound healing fibroblasts of the cornea | 10–50 μm/min | 0.7-3 bl/min | 106807, 106817 |
| *Amoeba Dictyostelium discoideum* | 10 μm/min | ≈1 bl/min | 106825 |
| human neutrophil | 9 μm/min | ≈1 bl/min | 106809 |
| glioma cells | 50 μm/hour | 4 bl/hour | 106810 |
| mouse fibroblastoid L929 cells | 30 μm/hour | 2 bl/hour | 106808 |
| human H69 small cell lung cancer cell | 16 μm/hour | 1 bl/hour | 106815 |

Figure 4 Comparison of prokaryote and eukaryote

(A) (left) A.brasilense (bacteria) from Alexandre's lab and (right) Hela cell (human cell) from ATCC under microscopes of 20x and 40x magnification power (B) Speed of prokaryotic and eukaryotic cell. (the first several entries of eukaryotes chart are unicellular organisms). This table is adapted from [39]

**Technical Approach**

## Data Preparation

Chapter 1

The data are the extracted coordinates of time series of different bacterial genotype. Sample size is 4029 with 5 classes(1 wildtype SP7 and 4 mutants ΔA1, ΔA4, ΔY1, ΔY4 corresponding to different deficits in chemotaxis system1 and 4). The sample sizes for each class are 26%(wildtype), 15%(ΔA1), 19%(ΔA4), 21%(ΔY1) and 19%(ΔY4). The sequential data for each sample is two dimensional: velocity and angular velocity. In chapter 1, only basic machine learning methods are tested. These basic methods are not applicable for time series data, so the first step is to convert the two time series into regular data format. Two approaches are used to convert sequential data into regular data: discrete Fourier transform(DFT) and discrete wavelet transform. By visualizing the distribution of the sequence lengths, it is clear most of them are longer than 60 steps(frames). Therefore, all series data are first truncated into 60 timesteps. For each dimension of the sequence, the DFT outputs 30 amplitudes of the cosine waves and 30 amplitudes of the sine waves. So, a 60-steps sequence of 2 dimensions is converted to 120 features by DFT. The same change is applied for wavelet transform. The Discrete Wavelet Transform uses a Haar filter. So, in all, 240 features are obtained by application of these two methods.

Chapter 2

This chapter has the same goal as the first chapter, but with different type of data. This part uses video data with 3 classes: SP7(wildtype), ΔA4(mutant) and ΔY1(mutant). The video is 30 frames per sec with 1080*1920 pixels each frame. The sum of video length of all samples is about one hour. The videos are split into about 1 second or 32 frames per clip. Each frame is downsized from 1080*1920 to 240*320 to reduce the computational burden. Given the small sample size, some YouTube videos are also used to pretrain the model before using the bacteria video data. These video data are processed and used in the same manner of bacteria video. This YouTube video data set is called UCF101 and organized by University of Central Florida[29].

Chapter 3

This chapter uses the same coordinates data from chapter 1, but the coordinates are not directly input into the models. Instead, the coordinates sequences are first calculated into 26 features that characterize movement patterns. These features are used as the direct input to train the models. A detailed list and calculation of these parameters can be found in the appendix.

## MPP classification

The data set used for MPP classification was reduced from 240 features to 60 features with the PCA method. There were 60 features and 4029 samples to classify. There were five classes used in the classification. Python code was used to create the graphs and calculate the data.

The approach used to calculate the maximum likelihood estimation was to approximate the Gaussian parameters using python. The parameters needed were the μ and covariance matrices from the training set. They were calculated in the program for both classes.

The following formulas were used calculate the predictions:
The posterior probability of a parameter given the data is estimated using Bayes rule (1):

$$P(\omega_j \mid x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \tag{1}$$

Given p(x) is same for all parameters as shown in equation (2)

$$p(x) = \sum_{j=1}^{c} p(x|\omega_j)(\omega_j) \tag{2}$$

The equation (1) can be reduced into (3)

$$g_i(x) = P(\omega_j \mid x) = p(x|\omega_j)P(\omega_j) \tag{3}$$

To simplify the calculation, it can be expressed as

$$g_i(x) = \ln p(x|\omega_j) + \ln P(\omega_j) \tag{4}$$

If we assume normal density for multi-dimensional x:

$$p(\vec{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})] \tag{5}$$

We get the discriminant function:

$$g_i(\vec{x}) = -\frac{1}{2}(\vec{x}-\vec{\mu_i})^T\Sigma^{-1}(\vec{x}-\vec{\mu_i}) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i) \tag{6}$$

Discriminant function 1: The assumption is that features are statistically independent. This may improve the performance of the model by forcing this generality and overcoming overfitting. Mathematically, this assumption is translated as $\Sigma_i = \sigma^2$. .:

$$g_i(\vec{x}) = \frac{\vec{\mu_i}^T}{\sigma^2}\vec{x} - \frac{\vec{\mu_i}^T\vec{\mu_i}}{2\sigma^2} + \ln P(\omega_i) \tag{7}$$

Discriminant function 2: The assumption is that the covariance matrix for all the classes are identical but not a scaler of identity matrix:

$$g_i(\vec{x}) = -\frac{1}{2}\left(\vec{x} - \vec{\mu_i}\right)^T \Sigma^{-1}\left(\vec{x} - \vec{\mu_i}\right) + \ln P(\omega_i) \tag{8}$$

Discriminant function 3: No assumption on the covariance matrices. The covariance matrices for each class is calculated.

$$g_i(\vec{x}) = -\frac{1}{2}\vec{x}^T\Sigma_i^{-1}\vec{x} + \vec{\mu_i}(\Sigma_i^{-1})^T\vec{x} - \frac{1}{2}\vec{\mu_i}^T\Sigma_i^{-1}\vec{\mu_i} - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i) \tag{9}$$

The cross validation was implemented with the MPP algorithm. The dataset was randomized and split into 10 groups. Each time, 4029*0.1=3626 samples are used for training and 4029*0.1=403 samples are used for testing. The average was calculated as the final accuracy.

## Classifier Fusion

| Predicted | | | | | |
|---|---|---|---|---|---|
| Truth | Classifier 1 | class 1 | class 1 | class 2 | class 2 |
| | Classifier 2 | class 1 | class 2 | class 1 | class 2 |
| | class 1 | .8 | .7 | .4 | .2 |
| | class 2 | .2 | .3 | .6 | .8 |

Figure 5 An example of Classifier Fusion Table.

Each decimal represents the accuracy of the fused model. For example, when classifier 1 predicts the new data to be class1 while the classifier 2 predict this new data to be class 2, there is 0.7 probability that this new data is actually class 1 and 0.3 probability that this new data is actually class 2. In this situation we should assign the new data to be class 1 since we know class 1 has higher probability.

Classifier confusion is the simplest level of model ensemble (which usually combines more than two models, but confusion table can only combine two classifiers.). It always uses the better one out of the two classifiers for each data point for the final predication. Take the example in figure 5. Two classifiers are trained separately. The prediction of each sample from each classifier is recorded in the confusion table. When a new data point comes in, if the classifier 1 predicts class1 while classifier 2 predicts class 2, the final predication will be 1, because 0.7 is larger than 0.3 which means that in this situation the true label of being class 1 has higher probability in the training set.

## k-NN classification

k-NN makes prediction more from memory than from learning, it does not have a feedback system, instead, it assigned the class label to a testing data based on its neighbors. If the majority of its k nearest neighbor is class 1, then the label is predicted to be class 1. The nearest neighbors are defined using Minkowski distance (10):

$$L_k(a, b) = \left( \sum_{i=1}^{k} |a_i - b_i|^p \right)^{\frac{1}{p}} \tag{10}$$

The p here represents the dimension of the Minkowski distance. For instance, a p of 2 is the same as the Euclidean distance formula.

## k-Means Classification

First, k cluster centers were arbitrarily assigned into the dataset. Then, the distances between each data point to all the centers are calculated and each data point is assigned to the closest center. After each data point is assigned to a center, the center is reset as the mean of all the points that are assigned to this center. In each round, each data point may be re-assigned to a different center or cluster. This process is repeated until no change of the assignment of each data point.

## Decision Tree classification

Decision tree is a graphical representation of decision rules that can most efficiently classify the given data point. Decision tree can easily handle nominal data. Gini's Diversity Index (11) is used as measure of impurity.

$$\text{Gini index} = 1 - \sum p^2(i) \tag{11}$$

## SVM classification

Support vector machine is a supervised method to construct a hyperplane in a high-dimensional space for classification or regression. A MATLAB function called fitcsvm for binary classification is used here. One linear algorithm and two non-linear algorithms (polynomial and Gaussian radial basis function) are tested. A binary classifier is trained for each class with one-again-the-rest method. Then all 5 binary classifiers are combined by comparing their scores.

## Neural Network and Back Propagation classification

Neural network is mostly inspired by the mechanism of our neurons. When a neurotransmitter binds to a receptor on the synapse, it changes the potential for part of the cell. When the potentials from several resources of stimulus converge and reach certain threshold, it causes an action potential and the entire cell is fired up. Following this idea, the earliest precursor of neural network was invented and named single-layer perceptron(SLP). A perceptron is a program that can learn from the examples and generate a response with “true” or “false”. Each dimension of the data serves as one stimulus to the neuron, after some the multiplication and summation operations, the result of the activation function may reach certain threshold and fire the neuron where the output is 1. Otherwise the neuron is not fired and the output is represented as 0. However, SLP has a glaring limitation that it can only classify samples that are linearly separable. Therefore, its uses are constrained to simple logic such as “NOT”, “AND” and “OR”. Later on, more layers and non-linear operations are added to better learn the internal presentation of data. With more and more techniques to tackle problems such as vanishing gradients, dead node, slow learning, more complicated architecture of multi-layer perceptron and neural networks are explored. One of the most important techniques in the learning process of a neural network is back propagation. Propagation refers to the calculation process of partial gradients along the computational graph of a neural network using chain rule. When the neural network is making a prediction, a forward propagation happens to pass the information from the input to the output. After the prediction is made, the learning happens by passing the information of the errors back to the network, this process is back propagation. Back propagation is used to update the weight and bias according to their influence on the output later. This influence is measured by their partial derivatives in terms of the cost function. Cross entropy is chosen as loss function (12):

$$\mathrm{H}(\mathrm{p},\mathrm{q}) = -\sum_{\mathrm{x}} \mathrm{p}(\mathrm{x})\log\mathrm{q}(\mathrm{x}) \tag{12}$$

Activation function:

$$\hat{\mathrm{Y}} = \frac{1}{1+\mathrm{e}^{-(\mathrm{w}^{\mathrm{T}}\mathrm{X}+\mathrm{b})}} \tag{13}$$

Partial derivatives for weights and bias:

$$\frac{\partial J}{\partial w} = \frac{1}{m} X\left(\widehat{Y} - Y\right)^{T} \tag{14}$$

$$\frac{\partial J}{\partial b} = \frac{1}{m} \left(\widehat{Y} - Y\right)^{T} \tag{15}$$

To update weights and bias:

$$w = w - \alpha * \frac{\partial J}{\partial w} \tag{16}$$

$$b = b - \alpha * \frac{\partial J}{\partial b} \tag{17}$$

In MATLAB, this can be implemented with a function called patternnet with 9 different variants of the above calculations listed below to update weights and bias:

**scg:** updates weight and bias values according to the scaled conjugate gradient method.
**gd:** updates weight and bias values according to gradient descent.
**gda:** updates weight and bias values according to gradient descent with adaptive learning rate.
**oss**: updates weight and bias values according to the one-step secant method.
**rp:** updates weight and bias values according to the resilient backpropagation algorithm.
**cgf:** updates weight and bias values according to conjugate gradient backpropagation with Fletcher-Reeves updates.
**cgp:** updates weight and bias values according to conjugate gradient backpropagation with Polak-Ribiére updates.
**gdm:** updates weight and bias values according to gradient descent with momentum.
**gdx:** updates weight and bias values according to gradient descent momentum and an adaptive learning rate.

In each step, the parameter is updated, the model of the last time step is the trained model. The concrete learning process is still same as single neural network where you use chain rule to calculate the gradients, save it and backpropagate these gradients to update the model. There are two very common differences between RNN and MLP. The first difference is the calculation of gradients. if you look at the computational graph, there are two input data flowing into the model, they have individual weights and gradients. During backpropagation, the individual gradients are summed together. Another difference is the concept of hidden states in RNN. They are not considered same as the

parameters of the model, instead they are placed in spots similar to the input data during model calculation.

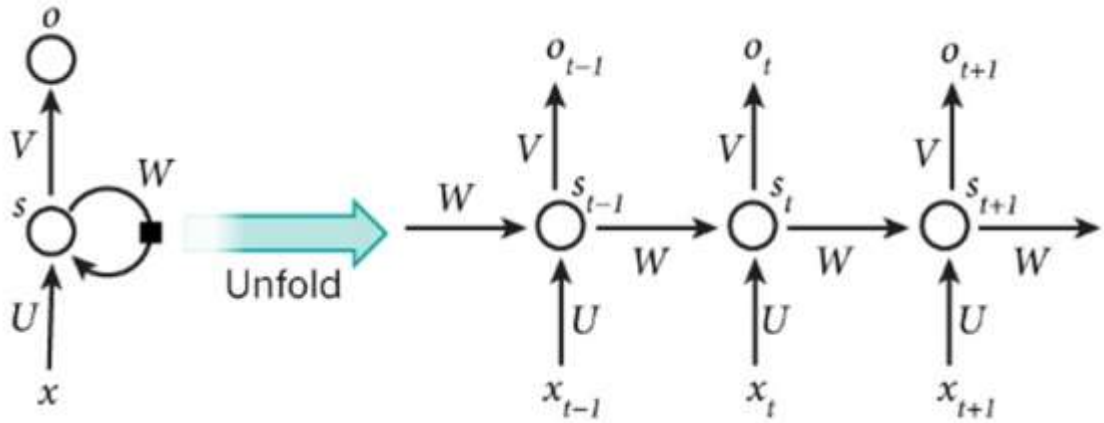



Figure 6 Recurrent Neural Network.

The representation in the left is RNN model with the output being feed back to the model again. It may be more intuitive to look at the representation in the right. It rolls out the model along the time steps, all the model parameters including W, U, V are same as those in the left. What changes in each step is the computation result including the stepwise output o and the hidden state s.

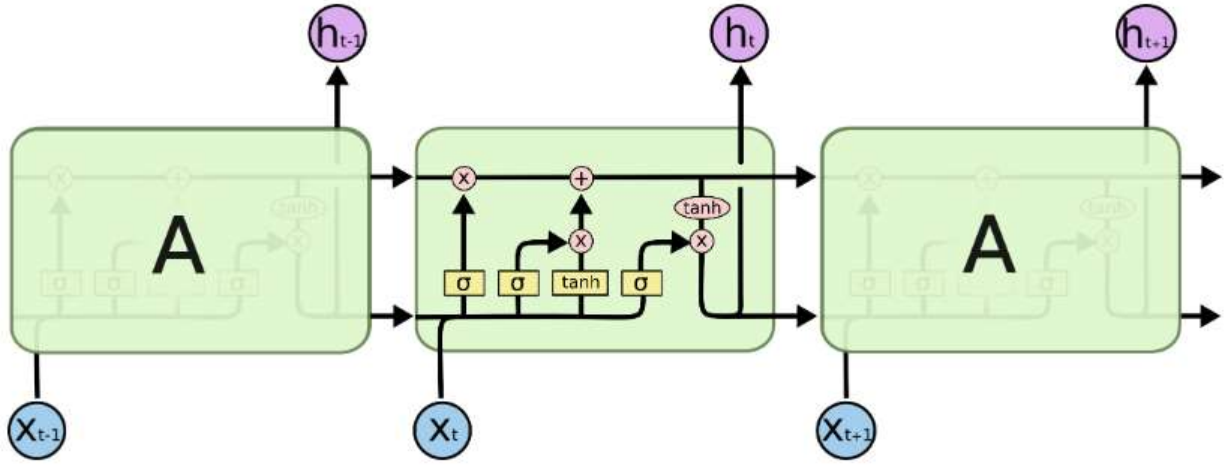



Figure 7 The repeating module in LSTM.

LSTM is one variant of RNN. It allows more control over the new information, memory, and output by introducing their weights to the model. These weights are the parameters that the model uses to make inference and try to improve.

long-short-term memory (LSTM) is a variant of RNN. It allows more control on the input, output and hidden state. The internal structure of each LSTM node is shown in figure 7. Mathematically, the model is learning to improve the parameters of:

| | | |
|---|---|---|
| Input gate | $i_t = \sigma(W^i x_t + U^i h_{t-1})$ | (17) |
| Forget gate | $f_t = \sigma(W^f x_t + U^f h_{t-1})$ | (18) |
| Output gate | $o_t = \sigma(W^o x_t + U^o h_{t-1})$ | (19) |
| New memory | $\tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1})$ | (20) |
| Final memory | $c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$ | (21) |
| Final hidden state | $h_t = o_t \cdot \tanh(c_t)$ | (22) |

## Convolutional Neural Network

CNN is widely used to process 2D image data. They are designed to extract features from image in a hierarchical manner from low level features such as straight line and shadow transition, to higher level features such as shapes and objects. A vanilla CNN can be

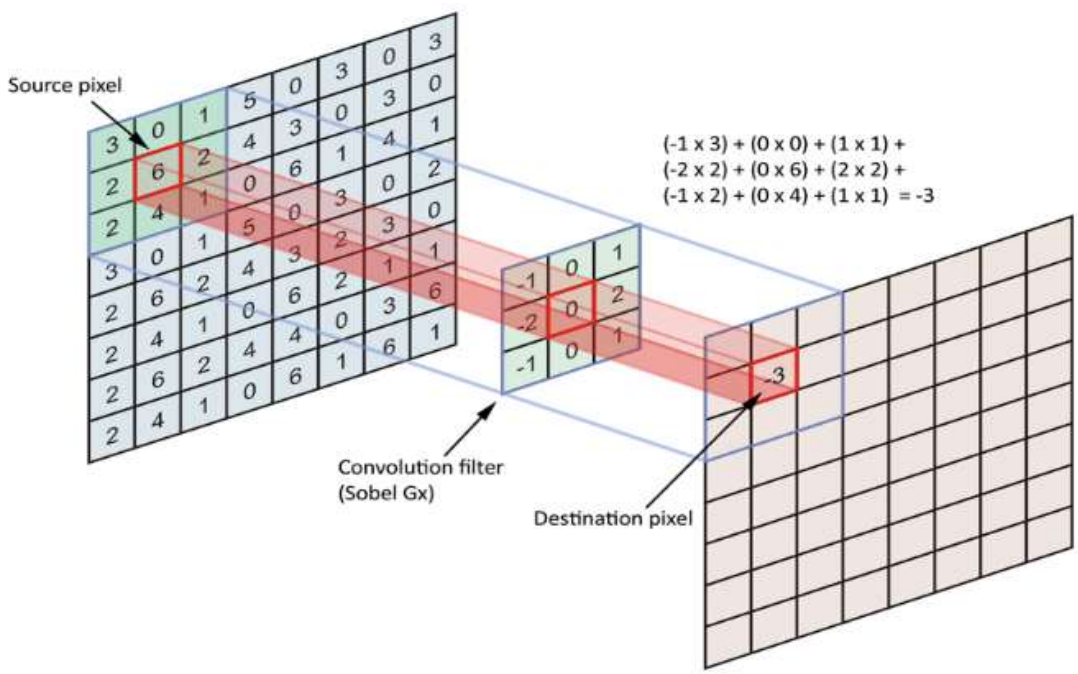



Figure 8 Convolutional Neural Network.

CNN is mostly used for image related tasks. Filters in CNN models serve as edge detector or feature detector. The result is computed by layers of convolution operation between the output from last layer and these filters. The size of the output of each layer is also decreasing because of this convolution operation. The model parameters are the values in the filter. By learning from the data, the model changes these filter value to make better inference.

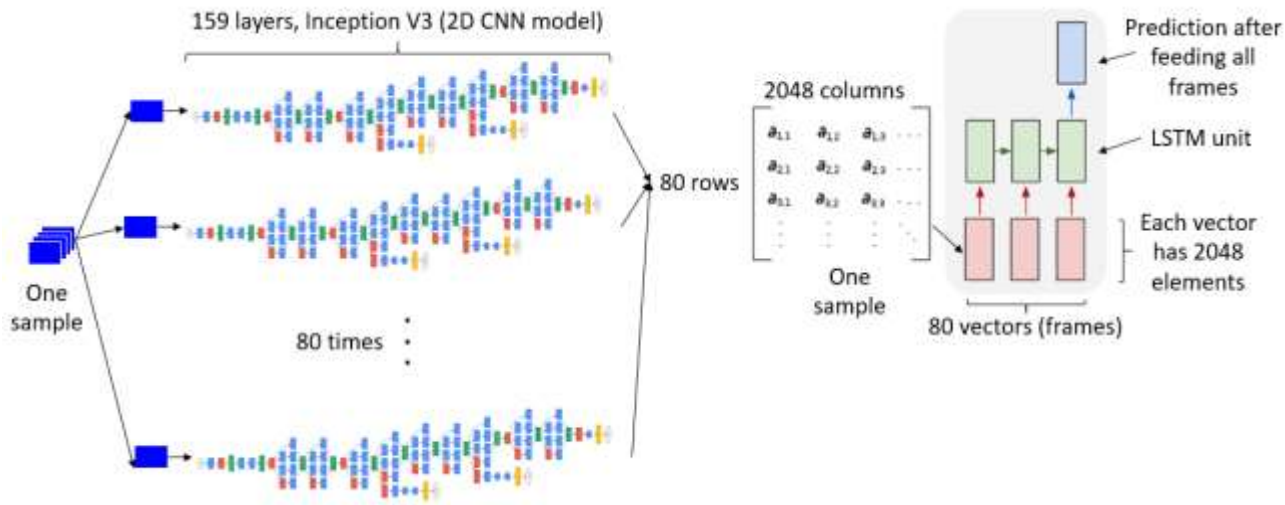



Figure 9 CNN-RNN mode for video classification.

The image data is first compressed by CNN model. Features of less dimension are extracted by this step and fed into the RNN model. RNN model again can have different number of nodes and different number of layers. In this study, the video is converted in to 2048 features with 80-timesteps.

thought as a 2D variant of the regular 1D MLP with a convolution operation on each layer. As illustrated in figure 8, the data is the source pixel and larger than the size of the filters. Filter is the matrix of trained parameters in each layer. When data flow through the model, the filter walks through the entire 2D data by convolutional operations (adding up the multiplication between each pair of elements in the data and the filter). This way the data is being compressed into features of smaller sizes. The intuition of this mechanics is that if the scanned area in the data is similar to the filter in terms of the element wise values, then the result of this convolutional operation should be larger, this way the feature is extracted and compressed into a single value. After layers of layers operation like this, the data is finally compressed to the level where the probability for each class can be derived.

## CNN-RNN

CNN-RNN is an architecture proposed in 2015 to process video data[30]. It combines the capability of processing image data from CNN and the capability of processing time series data from RNN. As shown in figure 9, each frame in the video is first converted from 2D features into to 1D features by CNN. The 1D features of each same are then fed into RNN to capture the structure between the frames. Since there are two stages in this model by design, the model can be trained either end-to-end or only at the RNN stage by using a pretrained CNN model.

## 3D CNN

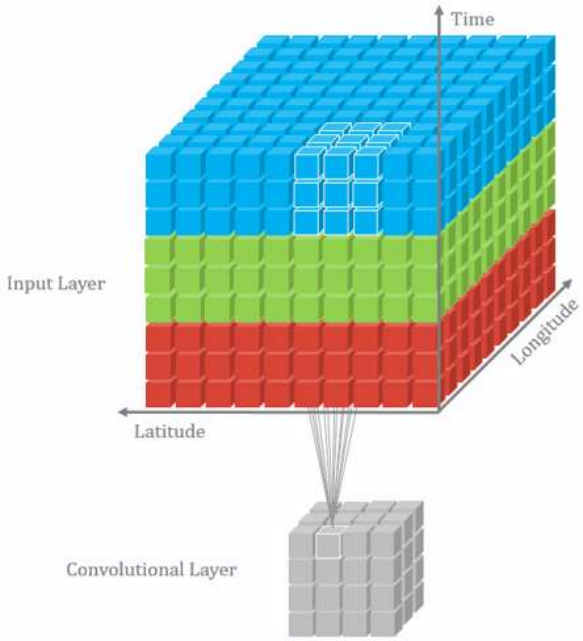



Figure 10 3D CNN for video classification.

The temporal dimension of the video is processed in the same way as image data. Instead of being directional, the temporal data is compressed together with the spatial data of the image.

3D CNN processes the temporal information in (sequential features) the same way it processes spatial information (2D features). 2D CNN compresses data by using a 2D filter. Similarly, a 3D filter is used in 3D CNN as shown in figure 10. A 3D convolution operation is used to extract the features from the data through the filter. The biggest difference between CNN-RNN model and 3D CNN model is that the former treats time as a directional dimension and process image and temporal dimension with different neural network structures, while the 3D CNN model treats time no differently than the way it treats image data, the size of the data collapse equally from all 3 dimensions and no direction in the temporal dimension distinguished.

## Testing and validation accuracy

The accuracy of a trained model usually refers to the validation accuracy instead of the testing one. The reason is that the validation data is closer to the real data than the training data in the sense that the model has never learned from them before. Because

of this reason, the validation accuracy is expected to be lower than the training accuracy. However, this is not the case in the project due to the use of dropout.

## Results

### Result 1: movement coordinates can be used to classify bacterial motility.

#### Result 1.1: dimension reduction reduces the skewness of feature distribution.

**ICA**

Independent component analysis is a method to separate multivariate signals to independent signals. The original 240 features are decomposed into 60 features by ICA. As shown in figure 11, these independent components tend to have different distribution than those from PCA.

**PCA**

As mentioned before, the velocity and angular velocity sequences of each sample are converted into frequency values from Fourier transformation and wavelet transformation. Each sequence data is converted into 60 frequency values by each method; therefore 240 features are generated. 60 most important features are selected from these 240 features by PCA. These 60 features are used for later classification. For comparison, top 5 features are also tested. As shown in the elbow graph in figure 12, the error rate of using the first 60 features is 1-96.02%=3.98%. The error rate of only using the first 5 features is 1-31.78% = 68.22%. The generated features are more normally distributed compared to the original 240 features. The principle components are more “normalized” compared to the input features as compared in the bottom graphs of figure 12.

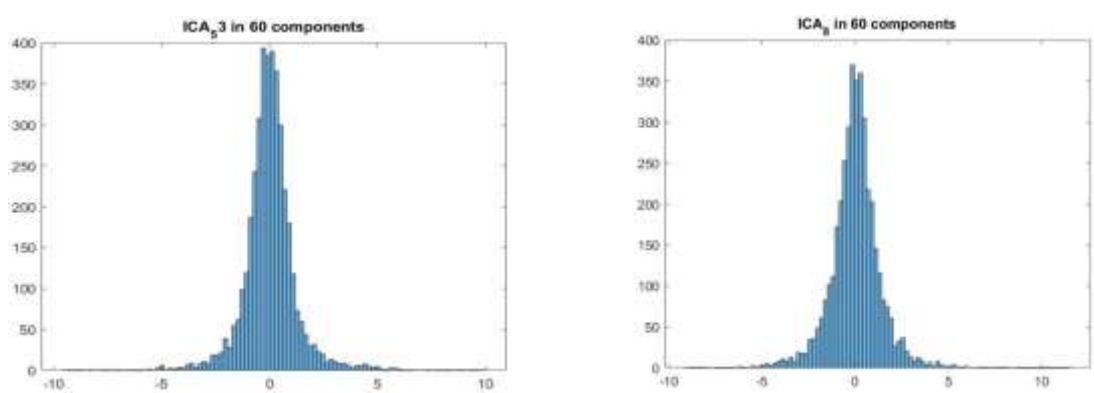
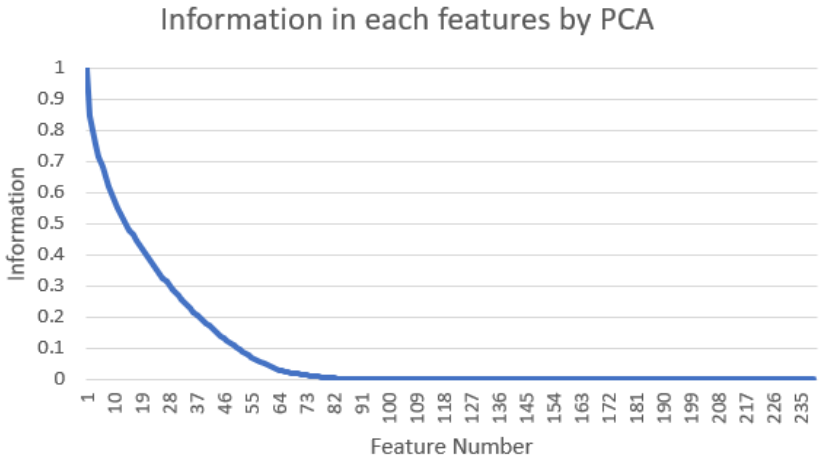



*Figure 11 Feature distribution after ICA*.

Features extracted by ICA is less skewed compared to the original features and less normally distributed compared to the features extracted by PCA.



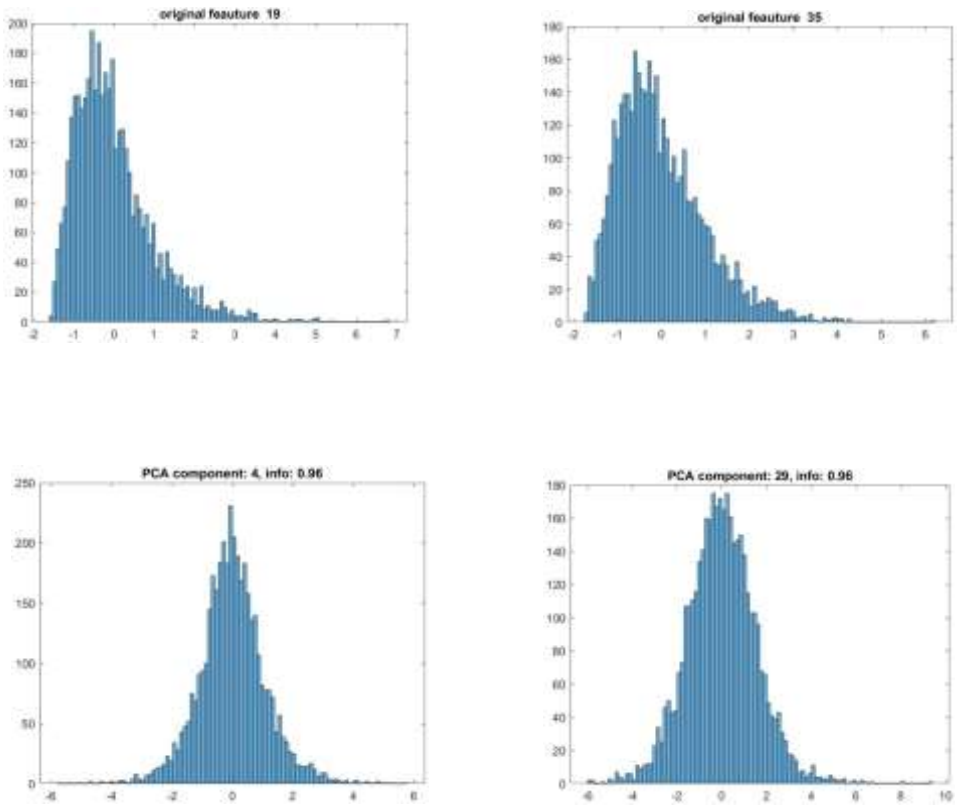




Figure 12 Feature distribution before and after PCA.

The elbow curve is used to decide the number of principle components. The best tradeoff is around 60-80 features. The original features tend to have different distributions than the principle component features.

Result 1.2: Single neural network gives best prediction.

**MPP classification**

As one of the most basic machine learning methods, MPP is first applied to predict the genotype. The overall accuracy is very low compared to other methods, but the results stay consistent with other methods. As introduced before, MPP has strong assumptions on the distribution of each feature and compute these distributions directly. Compared to the more complex models such as neural network, MPP does not have a feedback system to learn from the data gradually. This strong assumption of all features having normal distribution limits its learning ability. As shown by the bar chart in the left side of figure 13, predictions made by MPP are around 50% accuracy. Different assumption of covariance matrix of features does make a difference in terms of the accuracy. Case 3, which makes the predictions based on the actual covariance of each feature yields the highest accuracy of 51.3% which is 4% higher than that of the model case 1 which assumes the covariance matrix of features in each class to be an identity matrix. The fact that the computation with more accurate distribution parameters predicts better implies that the model is underfitting and there is more potential information can be extracted from the data.

The right side of figure 13 shows the prediction accuracy of each class. Similar to the results made by other machine learning models, the wildtype (class 1) has the highest accuracy around 75%. The high accuracy of wildtype can significantly improve the overall accuracy since it has the highest sample ratio (26%) in this data set. Class 2-5 corresponds to different mutant that are deficit in CheA of chemotaxis system 1, CheA of chemotaxis 4, CheY of chemotaxis system 1 and, CheY of chemotaxis system 4. These four mutant classes have relatively similar sample ratio from 15%-21%. The second highest class-specific prediction accuracy around 65% is from class 3 or mutnat ΔA4. In A. brasilense, Chemotaxis 4 has been discovered to play a role in controlling the probability of swimming reversals and is essential for all chemotaxis response or competitive wheat root surface colonization. CheA4 is essential for the cells to reverse swimming direction. In the study of [31], the lack of CheA4 almost eliminate the reversal event totally. The ΔA4 mutants decreased the swimming reversal frequency from ~50% of the wildtype to almost 0. Meanwhile, the lack of CheA4 also decreases the swimming speed compared to the wild type. These two quantified changes of ΔA4 makes it the most distinctive mutant from the wildtype among all the mutants in this dataset. Therefore, ΔA4 has the highest accuracy among all the mutant is not a surprising result.

The class of second highest accuracy is mutant ΔY1. As discovered by several studies[31-32], in A. brasilense, chemotaxis system 1 pathway contributes to the chemotaxis and aerotaxis. It has also been found to contribute to regulating changes in cell surface adhesive properties that impacts the cell-to-cell clumping and flocculating. In terms of the motility parameters, it regulates transient increases in swimming speed in

response to attractants. Comparing CheA and CheY, the former has been reported to play a more central role in regulating both the forward pathway that changes the motor rotation and also in activating a feedback loop through CheB that resets the sensitivity. Comparing Che1 and Che4, the functions of the former seems quite divergent in that it

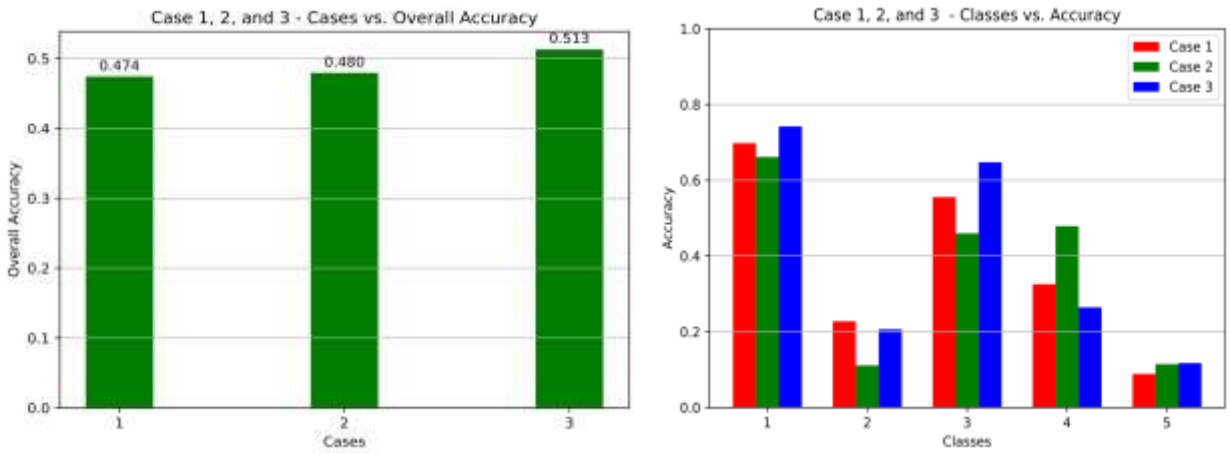



Figure 13 Prediction accuracy of each class using MPP.

The overall accuracy is around 51%. The accuracy for wildtype, ΔA1, ΔA4, ΔY1 and ΔY4 are around 75%, 22%, 65%, 50% and 10%. Case 1, 2 and 3 are different assumption on the covariance matrix for computing MPP. Case 3 has the weakest assumption and highest accuracy

.

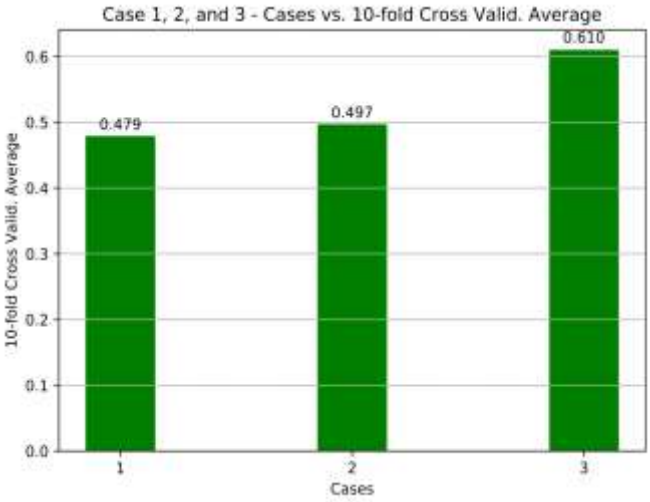
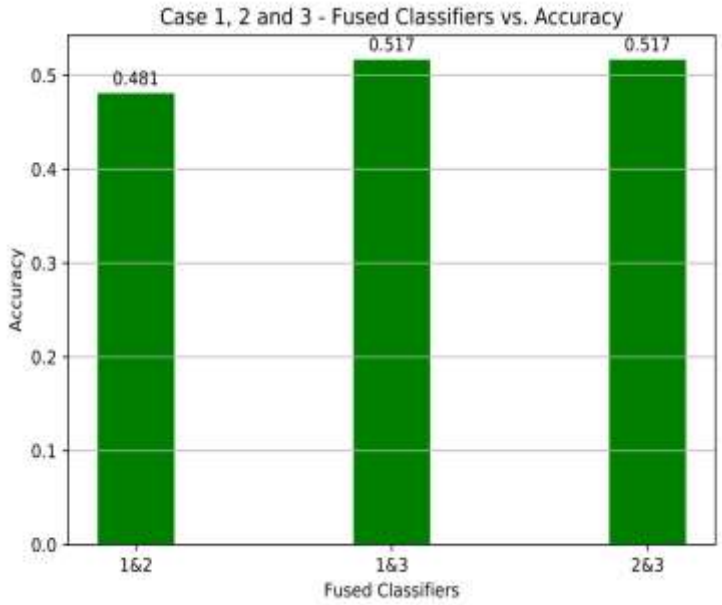



Figure 14 Prediction accuracy of each class with 10-fold cross validation.

The accuracies from cross-validation is a more stable estimation of the model's ability to predict new data




.

Figure 15 Prediction accuracy of fused classifiers.

Classifiers fusion is the simplest model ensemble and can improve the prediction sometime. In this study, the fused classifiers do have higher accuracy compared to the original single classifiers, but lower accuracies compared to that from the cross validation.

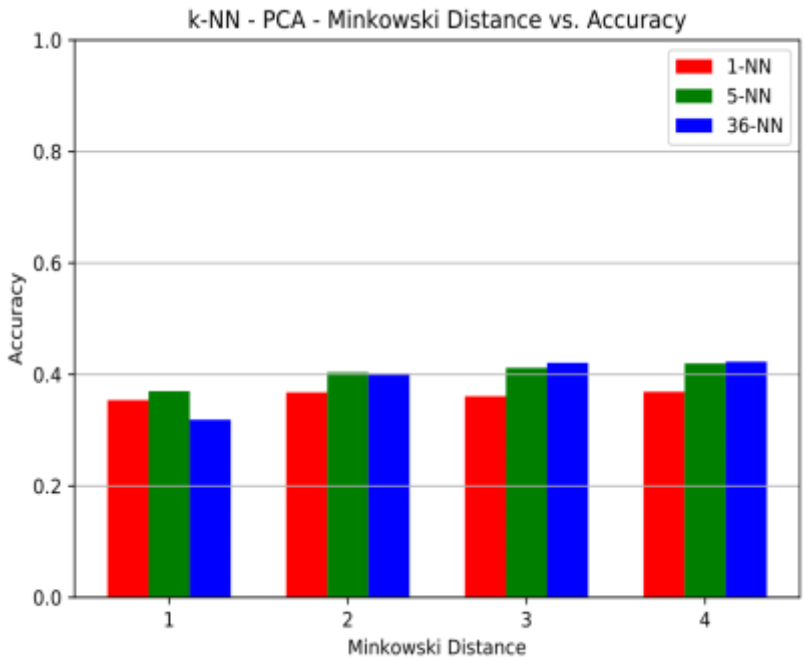
regulates both taxis behaviors as well as other functions. On one hand, Che1 affects the swimming speed and the swimming directions. On the other hand, it also affects clumping and flocculation. It mediates the transcription of extracellular polysaccharides and, therefore, changes the cell surface adhesive properties.

The result of k-fold cross validation is a more stable estimation of the model's ability to predict new data. The data set is randomly partitioned into 10 groups. As shown in figure 14, the average value of 10 times of estimation is slightly higher than the corresponding accuracy in figure 13. Case 3 was still the winning algorithm for the highest classification accuracies with 61% accuracy. Figure 15 shows the result of fused classifiers of the pretrained case1, case2 and case3. The best accuracy 0.517 comes from fusing classifier 2 and 3. Compared to the accuracy from classifier 2 (0.48) or 3 (0.513)only, there is a very slight improvement of 0.037 and .004. Although the accuracy of case3 with 10 fold cross validation is higher than the fused classifiers, it does not necessarily means the fused classifiers underperform the single classifiers, because the fused classifiers use the pretrained models from figure 13 which are known to give lower accuracy. Contrary to the motive of fusing classifiers, fused classifiers do not necessarily outperform the original

single classifiers. Different strategies such as voting mechanism needs to be tested to improve the fused classifiers.

**k-NN classification**

k-NN has 2 parameters: the number of neighbors k and the order in Minkowski distance calculation. To improve the prediction accuracy, different values for each parameter have been tested. As shown in figure 16, out of the values being tested, more neighbors tend to give better performance, but the trend is different for different Minkowski distance. When p is 1 (Manhattan distance), 5 neighbors gives better prediction than 36 neighbors. For the order of Minkowski distance , the model tends to perform better when the value is larger. Overall, this model does not predict well compared to other models tested in this study. The best accuracy is .423 when using 4 as the parameter in Minkowski Distance, and 36 as the k value. The reason of low accuracy may be from the limitation of this method itself. This method relies more on memory of the training sample rather than learning. The information extraction process of this model is predefined by the Minkowski distance; therefore, it does not have parameters to improve over the training process. It also does not utilize information such as the prior probability of each class or the distribution of features. When the new data does not fall into a space where the neighbors are universally from one class, the new data may be assigned to the wrong class. It has been studied and reported that the prediction of k-NN can be very sensitive to the measure of distance. The results from some studies show large gaps between the




| | Minkowski Distance: 1 | | | Minkowski Distance: 2 | | | Minkowski Distance: 3 | | | Minkowski Distance: 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | 1 | 5 | 36 | 1 | 5 | 36 | 1 | 5 | 36 | 1 | 5 | 36 |
| accuracy | 0.354 | 0.37 | 0.319 | 0.368 | 0.404 | 0.402 | 0.361 | 0.412 | 0.421 | 0.369 | 0.42 | 0.423 |

Figure 16 Prediction accuracy of k-NN.

K-NN has 2 parameters to fine tune. One is k, the number of neighbors, the another is the order for the distance measure. Different values for each parameter are tested, the highest prediction accuracy is made by 36-NN with order as 4 for Minkowski distance.

performance of different distances[33-34]. Unfortunately, only very limited number of distances are tested here. Given the known merits of k-NN such as its simplicity and its tolerance to noise, it may worth trying to improve the model by testing more distance measures. It would also be interesting to look at the class-specific accuracy instead of just the overall accuracy so that the model may be improved by balancing the data or retrain on specific class.

**k-Means clustering**

As an unsupervised model, k-means only clusters data, it does not predict the labels. It is used to explore the internal representation of the data. An objective way to decide the number of centers k is to use the elbow curve where the variation reduction is plotted against the number of k. However, in this study, the number of classes is known and used

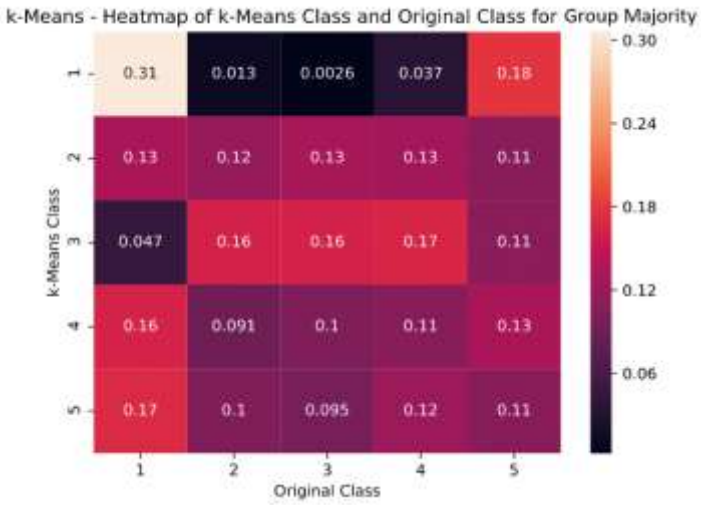



Figure 17 Prediction accuracy of k-Means.

The k-means classes are clusters assigned by the model without knowing or learning from the labels. These clusters are not necessarily one-to-one mapped to the 5 genotypes of the sample, but the correlation of the wildtype and ΔA4 are still highest which indicates these two classes have more distinctive swimming patterns than others.

as the number of k. As shown in the heatmap of figure 17, most of the k-Mean classes do not correlate well with the true labels, although class 1 still has the highest correlation.

The high correlation with class 1 indicates that the wildtype samples tend to cluster together better in the feature space. However, it does not explain which feature is most helpful in distinguishing the wildtype from the rest. Similar to the result of MPP, class 3 or mutant ΔA4 has second highest correlation which indicates mutant ΔA4 is more clustered and distinctive than the rest of the genotypes. The congruency of results shared by the supervised models and k-means also indicate the preprocessing of the features (Fourier transform, wavelet transform and PCA) maintain enough information to represent the swimming patterns.

Although the correlations of each genotypes are not very high, the fact that wildtype and ΔA4 have higher correlations agree well with the results from supervised learning. If the distribution of one class has more than one cluster center or the cluster simply spread out too much, it will be harder for them to be identified as one cluster or has a high correlation with the labels. The clusters formed by the k-means model do not necessarily correspond to one class from the sample labels. For example, there may be 2 phenotypes inside the wildtype that are identified and clustered separately by the model while the swimming patterns of ΔY1 and ΔY4 are very similar and clustered together by the model. This misclustering will lower the correlation for each of the classes. This may be the case for mutant ΔA1, mutant ΔY1 and mutant ΔY4.

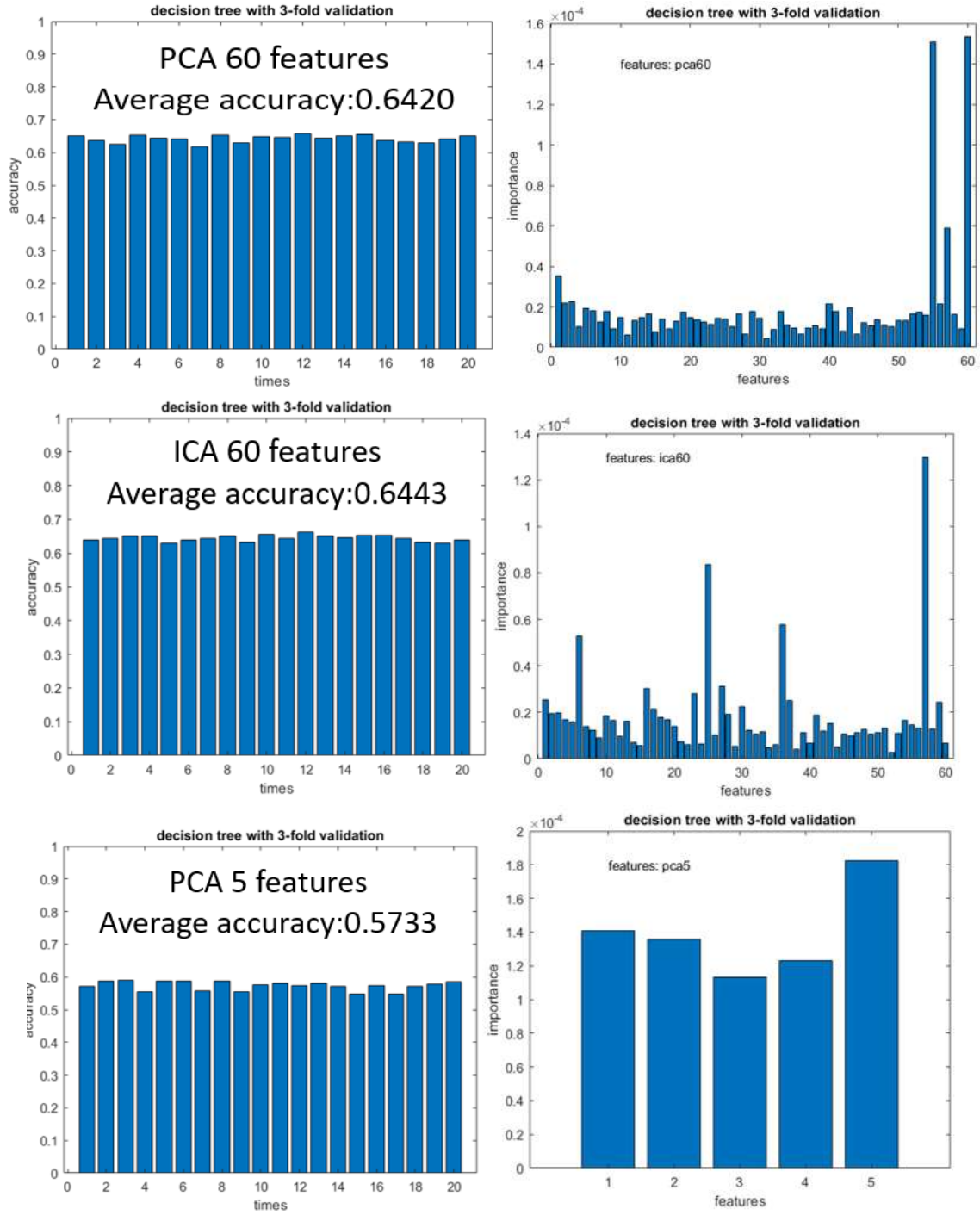



Figure 18 Prediction accuracy of decision trees.

In the left column, each bar represents the average accuracy from the 3-fold cross validation. Each average accuracy is further repeated 20 times.

**Decision Tree classification**

Figure 18 shows 64% overall accuracy from decision tree with 60 features. Each bar in the left is the average accuracy of a 3-fold cross validation which is further repeated 20 times to get better picture of the result. Even though PCA and ICA extract features differently, the accuracies from PCA and ICA extracted features are very close. The bar charts on the right show the importance of each feature, what is interesting is that the important features are more centralized in PCA extracted features compared to that of ICA extracted features. PCA tends to extract global features while ICA tends to extract local features. It implies there are more local features than global features that are important to the classification. All of these features either from PCA or ICA are not quite interpretable or visualizable due to the Fourier transform and feature extraction techniques, but the knowledge of feature importance can still be useful if the goal is to make inference with smaller model rather than explaining the model. One problem is unclear is that the indexes of features follow their importance from the elbow curve in figure 11, which means features with smaller index should be more important than features with larger indexes, but this is not the case for decision tree. Another interesting result is that, after taking away 55 features from the 60 PCA extracted features, the accuracy only decreased ~5% from 65% to 60%. The number of important features in these 60 PCA extracted features is smaller than 5.

**Single Neural Network classification**

Figure 19 shows the prediction accuracy by single neural network. 9 optimizers are tested for updating weights and bias, with 10 times of 4-fold cross validation. Although some optimizers make the model converge faster than the others, they reached very similar accuracies at 79%. This is dramatically higher compared to the results of other models and sets the benchmark for this dataset. However, some information is likely lost during the preprocessing of this dataset and there are several reasons that contributes to the due to low prediction accuracies: (1) the length of the sequence sets the lower bound of observable event frequency. For example, it is suggested by a study that adaptive cellular behaviors such as aerotaxis are tightly coupled with metabolism in A. brasilense. They form an aerotaxis band at a distance from the air-liquid interface within 2-3 min[35]. But you are only allowed to observe them for 2 seconds, it is less likely for you to notice this trend because the accumulative result is not significant to be observable. In the data preparation,

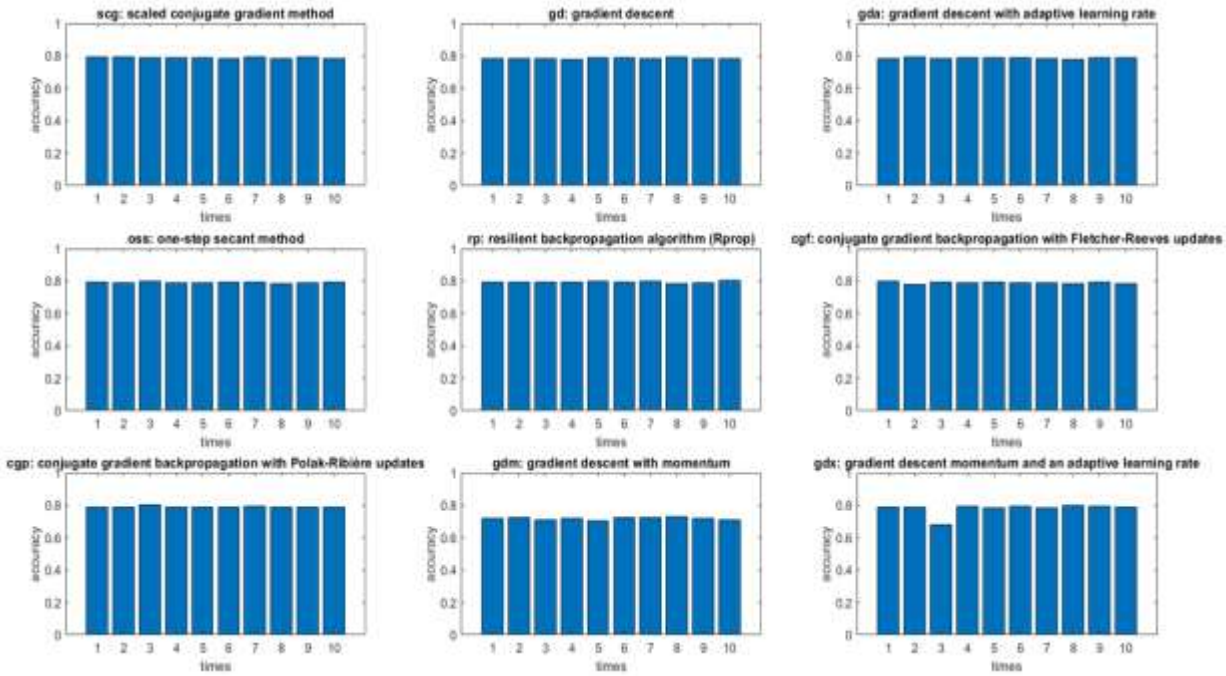



Figure 19 Prediction accuracy of single neural network.

Different optimizers are tested for updating weights and bias, with 10 times of 4-fold cross validation. The maximum times of epoch is 1000. For algorithms with fixed learning rate, it is set as 0.01. The minimum gradient to execute updating is 1e-05 and the maximum number for validation check is 6.

all the sequences are cut into 60 steps long which equals to 2 seconds in real time. This process put a lower bound on the event frequency that can be observed by the model. When an event happens only once every 4 seconds, it will show at most once or not show at all in the sequence of 2 seconds. First, this pattern cannot be recognized as a repeated event or the frequency of the event cannot be inferred by model. Second, the inconsistency between having or not having this even may confuse the model even though the event frequency underneath actually stays consistent. (2) The data is preprocessed by Fourier transform and wavelet transform. These methods have not been proved to be effective in extracting movement trajectory information. The information lost through these transform process sets another upper bound on the final prediction accuracy. (3) The setting of the data collection may not be optimal to distinguish the genotypes and there are many noises in the data. For example, it is known that the rings observed in the soft agar plates may be the result of pseudotaxis, which is a form of

translocation through the agar that does not result from chemotaxis signaling. Phenomenon like this serves as noise for the current classification task.

**SVM classification**

SVM is the second-best model among all modes tested. As shown in figure 20, SVM gives ~72% prediction accuracy with PCA 60 features and 71~% accuracy using ICA 60 features. Polynomial algorithm performs better when there are more features, while radial basis function performs better with less features. The bar chart on the right is relative improvement compared to their prior probability. The classes with highest accuracies are wildtype and mutant ΔA4. This result agrees with the results from MPP and k-means. The congruency proves that wildtype and mutant ΔA4 are most distinguishable classes in this data sets. These three methods provide different perspective for the same conclusion. In MPP, the model learns the parameters of the distribution of each class. This implies that wildtype and ΔA4 have most unique distribution compared to other genotypes. In k-means, the model learns the centers of each cluster. This means wildtype and mutant ΔA4 have

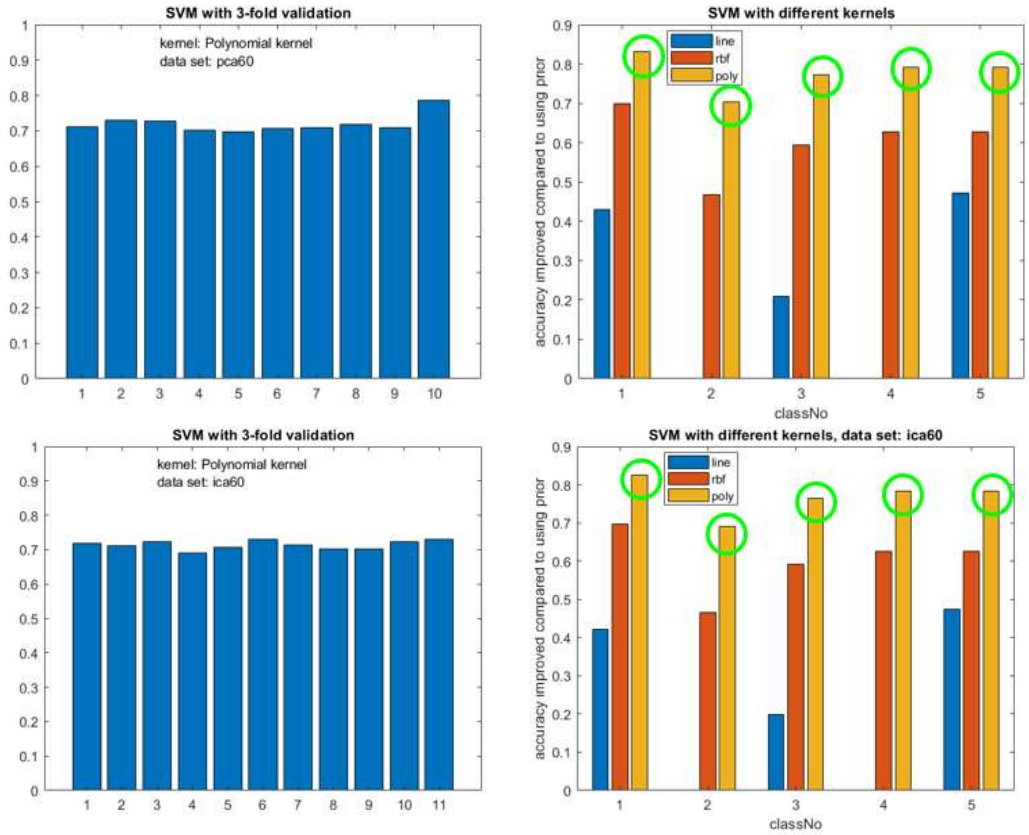



Figure 20 Prediction accuracy of support vector machine.

Accuracy is computed as the average accuracy of 10 times of 3-fold cross validation. Linear kernel, radial basis function kernel and polynomial kernel are used.

the most well-defined clusters compared to other genotypes. In SVM, the model learns the boundary of each class. This means wildtype and mutant ΔA4 have relatively clear boundary that can be separated from other genotypes.

Overall, traditional machine learning methods make decent prediction based on the coordinates data (figure 21). Single neural network or back propagation gives accuracy at ~79%. Support vector machine gives second highest accuracy at ~72%. The lowest accuracy, not surprisingly, comes from the unsupervised method k-means. Combining these results, we can have several conclusions: (1) using the coordinates data, traditional machine learning methods can predict the genotype correctly 80% of the time. (2) wildtype is most distinct genotype among all the classes (3) Δ4A is the most distinct genotype among all the mutant. (4) single neural network outperforms other models (5) the most distinct genotypes data form clusters that can be identified by unsupervised methods. (6) Extracted features from PCA and ICA contribute to similar prediction accuracy using decision tree and support vector machine.

The prediction accuracy is decided by both the quality of the data and the mechanisms of the model. The value of accuracy should always be considered in the context of these two factors. Knowledge of these two aspects sheds light on the possible accuracy range. For the lowest accuracy, the results should be compared with the benchmark of 26% accuracy (the highest prior probability of all classes). For the high benchmark, there has not been a very good standard other than the empirical results using the similar datasets. There are several approaches to be considered to further improve the performance of the models and the quality of the data: (1) use sequential data longer than 60 steps. (2) use models that can process sequential data directly such as RNN (3)optimize the data collection by designing the environment to differentiate the genotypes. For example, in A. brasilense, Che1 is known to play a role in aerotaxis and clumping through its regulation in swimming speed and other indirect mediations[32]. However, this dataset is not collected in the setting of air diffusion or gas perfusion. (4) remove features that have very low variance. If a feature has only 1 value or very few values compared to the number of class, it indicates this feature is useless and should not be input into the model. It would be helpful to get the statistics and visualize the distribution of the features we consider feeding into the model before training the model. There are several other factors that may contribute to the intrinsic noise in the dataset such as the crosstalk between signaling pathways, the resolution of the microscope, the accuracy of the tracking software.

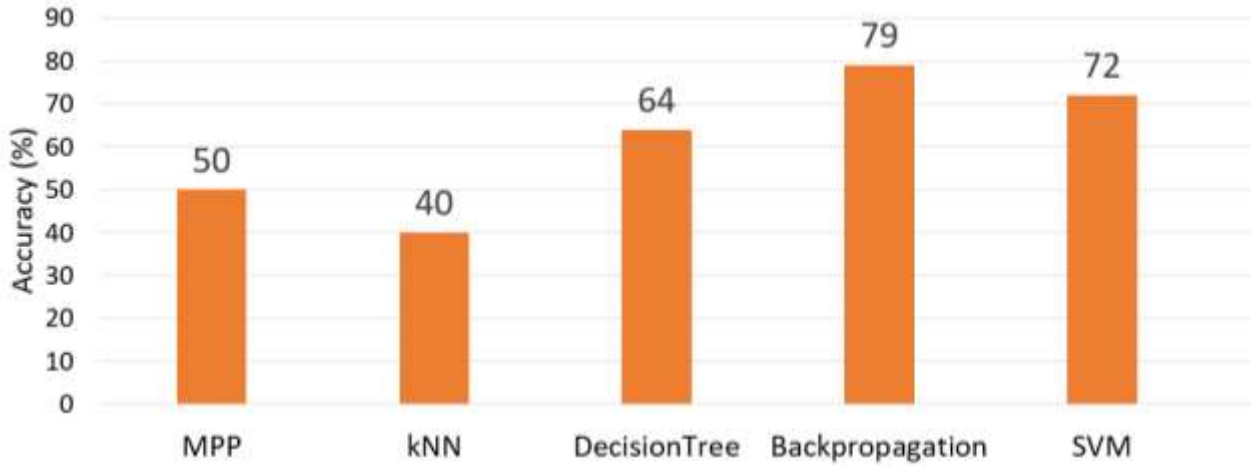



Figure 21 Overall accuracy of different models using coordinate features.

Out of all the algorithms tested, single neural network also labeled as backpropagation stands out with 79% prediction accuracy as the best model for this genotype classification task.

## Result 2: movement videos can be used to classify bacterial motility.

The results of the last study show several places where the prediction accuracy can be improved. In general, this include quality of the data and the choice of the models. In this study, rather than using the extracted coordinates, the video data is directly used in the hope to maintain more information by using the raw data. A glaring drawback of the coordinate's sequences data used in the last project is that the context of each bacterium and the information the collective behaviors of bacteria is lost. An individual bacterium can move differently based on its bacteria neighbors. Without this contextual information, the change of swimming patterns identified in the coordinate's sequence data may be more confusing than helpful. Video data keeps this information and hopefully will improve the model's ability to classify new data.

For the model, several deep learning neural networks are used in this project. Recurrent neural network (RNN) is invented to process sequential data. It has been widely used in tasks such as machine translation and stock price prediction. Therefore, it is chosen to process the sequential data rather than converting the sequence into a vector of features as in the last project. Since we are using video, each time point is no longer a vector of several features. Each time point in the video corresponds to a frame of image with RBG color channel. With a resolution of 1980*1920, this equals to 1980*1920*3 values for each time point. This is a very big load to the model. One way to process these image data is to rearrange these spatial values into a very long 1D vector by simply attaching each row of the image to the end of the previous row. Although the adjacent information above and below each pixel is lost during this arrangement, this approach achieved some success in areas like offline handwriting recognition[36]. However, a much better approach would be using another model before the RNN to compress the information and alleviate the burden on RNN. CNN is a neural network that is specialized in image related tasks. As mentioned before, CNN can recognize the object the picture by learning the hierarchical elements that constitute an object. More specifically, a structure called Inception V3 proposed in 2016 [37] is used in this project to process the image data. This preloaded Inception V3 is trained with YouTube videos. Although YouTube videos seem nothing like the bacterial videos to our eyes, they do share same low-level features that compose the higher-level features. This justifies the use of totally different video dataset to pretrain the model.

### Result 2.1: model with transfer learning is trained more efficiently

As mentioned before, two different architectures are used for this video classification task. One is CNN-RNN where the image information of each frame is processed by CNN and fed into the RNN model. Another is 3D CNN model where the temporal information is compressed simultaneously with the spatial information of the image. Training a large

model from scratch can take very long time and computational power. Transfer learning is an approach to accelerate the learning by using a pretrained model. Here, a pretrained Inception V3 is used to preload the weights of the model. Depending on whether using this pretrained model, there are two training approaches of the CNN-RNN model: (1) training with transfer learning which is labeled as two stage 2D CNN RNN in figure 22 and (2)training from scratch including both end-to-end CNN RNN and 3D CNN. While the end-to-end CNN-RNN gives more freedom to the model to learn from the data, it also takes much longer to train. Given the small sample size of the video data, end-to-end training may be more efficient.

As shown in figure 22, after several hours of training, the pretrained model did outperform other models or training approaches. None of the models trained from scratch was able to catch up the model preloaded with Inception V3 weights. The validation accuracy of the pretrained model (it is labeled as two stage 2D CNN RNN in the figure) is more than 30% higher than the validation accuracy from the models without using transfer learning. The higher efficiency of pretrained model result confirmed the idea that the bacterial videos share same low-level features with non-relevant videos from YouTube. Although the results presented here are from models with very limited training time. The pretrained model is expected to perform better in both short-term and long-term training. Apparently, when the training is relatively short, model with pretrained weights performs better because of the previous longer training time. What is less apparent is that, when the training is relatively long, the model with pretrained weights do not necessarily perform

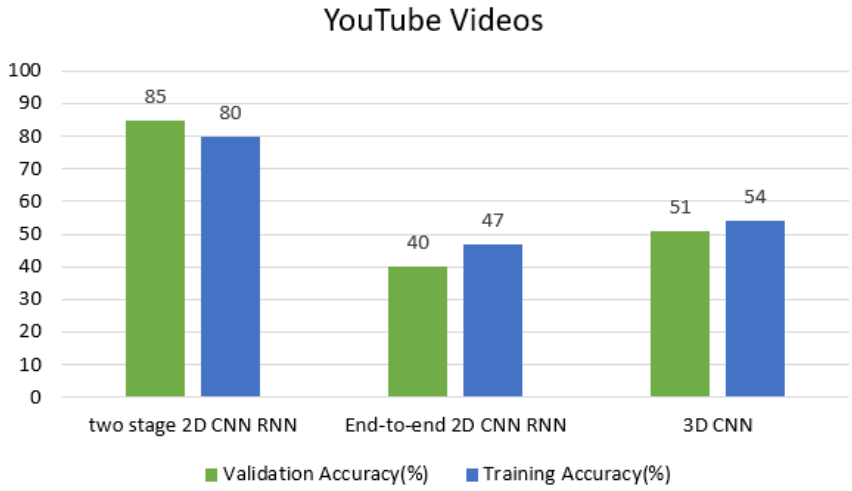



Figure 22 Prediction accuracies without and without transfer learning.

Two stage 2D CNN RNN and end-to-end 2D CNN RNN have same structure, but the former has pretrained weights and more efficient for the task. The accuracy of pretrained model is more than 30% higher than both models trained from scratch.

worse. With new task and data being presented to the model, the old weights will be washed off by the new information just as those models trained from scratch. If a pretrained model performs better in the beginning of the training, it is likely they maintain this efficiency throughout the training process.

Result 2.2: CNN-RNN model reaches 86% accuracy for bacteria classification.

After pretraining the models with YouTubes, the CNN-RNN model is further trained with the bacteria microscopic videos. This requires changing the top layers of the model since the number of class is changed from 101 classes of YouTube videos to only 3 classes of the bacteria videos. Although pretrained weights are loaded, they are not frozen and all layers are open to training. As shown in the left graph of figure 23, the model with pretrained weights reached 86% accuracy after 20 epochs.

This 86% accuracy is much higher than the prediction accuracy of 79% from the last study. Technically, the two results are not comparable because of the different dataset and different model structures. Each dataset has their advantage and challenge. For example, the setting of last project with coordinates data is more advantageous than this project with video data. The coordinates data was collected with chemical gradients that are believed to help demonstrate the behavioral difference of different genotypes, but these video data are collected without chemical gradients, therefore making it harder for the model to differentiate different genotypes. This disadvantage of the video data also explains the low accuracy of CheY1 in the next section.

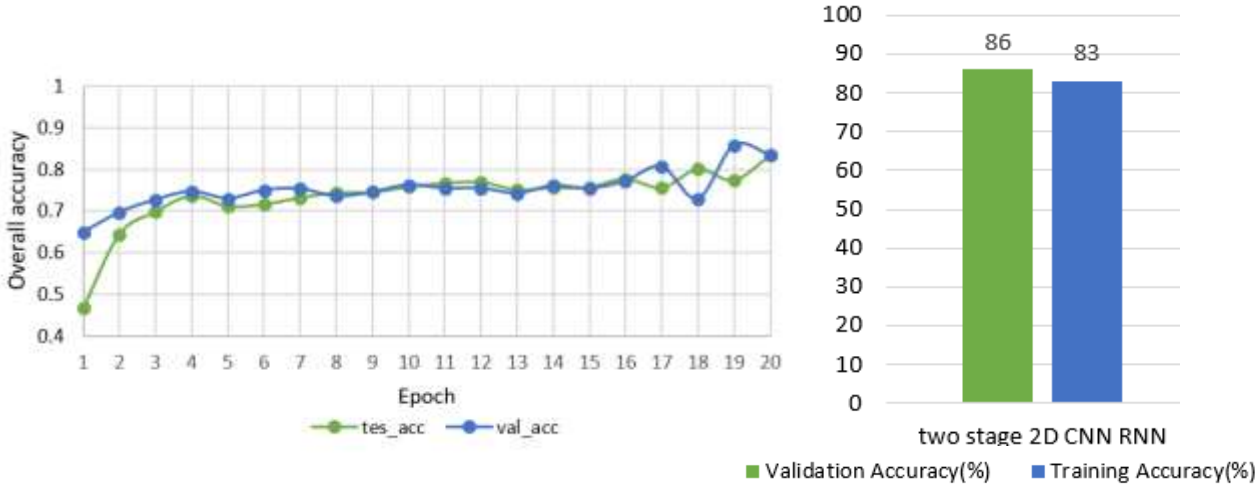



Figure 23 Accuracy of CNN-RNN model with transfer learning.

The validation accuracy is higher than the training accuracy due to the use of dropout. It is not uncommon to have higher validation accuracy than training accuracy when this technique is used.

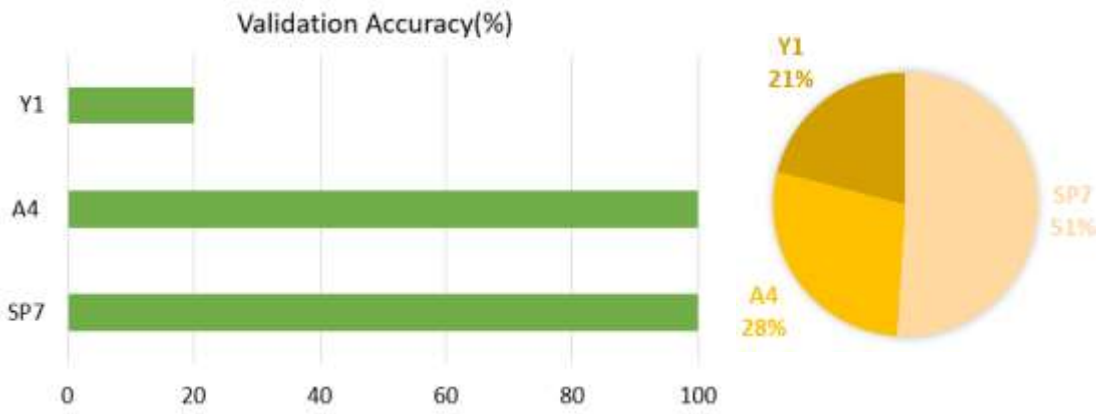



Figure 24 Class-specific accuracies of CNN-RNN model.

Wildtype(SP7) and ΔA4(A4) have 100% prediction accuracy while ΔY1 has extreme low accuracy. The prior probability of each genotype on the right explains the low accuracy of ΔY1.

Result 2.3: classes that share similar features are misclassified.

After looking at the overall accuracy, a check on the class-wise accuracy can help understand the model behavior. Figure 24 shows that the accuracies of wildtype(SP7) and mutant ΔA4(A4) are extremely high with both being 100%. As discussed before, mutantΔA4 has distinct swimming pattern compared to the wildtype. Under steady-state conditions, wildtype swim with long runs with an average probability of about 0.5 reversal/s of reversals. In contrast to the wildtype, mutants lacking CheA4 swim in straight runs and does not reverse. Mutant ΔA4 significantly decreased the frequency of reversal from 0.5 times/s to 0. Besides the frequency of reversal, mutants lacking CheA4 have a significant decrease in swimming speed. Under steady-state conditions, wildtype averagely swim around 35 um/s while mutant ΔA4 swims around 28 um/s which is one of the most dramatic decrease among all the mutant tested.

The low prediction accuracy for mutant ΔY1 and high accuracies of the other classes indicates that the potential improvement of this model can only come from improving the prediction for ΔY1. As mentioned before, this video data is not collected under the best settings to differentiate the behavioral difference of the genotypes. It is reported that CheY1 plays a role in aerotaxis and clumping by mediating the regulation of swimming speed. However, the environment of these bacteria does not have air diffusion, so mutant ΔY1 is believed to behave mostly same as the wildtype. In A. brasilense, it takes wildtype 2-3min to form an aerotactic band. Mutant ΔY1 takes twice the time of wildtype to form an aerotactic band. Since the video clips are 32 frames long which corresponds to roughly

1 second, it's hard for  models to capture such behavior. Most mutant ΔY1 sample are misclassified as wildtype (data not shown here). This is likely because of the higher prior probability of wildtype than mutant ΔY1 in the dataset. When the model cannot differentiate two classes, it is a smart strategy to always guess the class with higher probability. Making prediction simply using the prior probability without training any model is the most naïve way to make inference and is always used as the benchmark for training more complicated models.

## Result 3: some motility parameters can better classify bacterial motility.

The previous two projects discovered the model structure and data form that are most efficient in predicting bacteria genotypes. However, most researchers studying bacteria motility are not familiar with machine learning models or making inference. They are familiar with different motility parameters. They want to know which motility parameters can best characterize the swimming patterns and help to differentiate the genotypes. This task can be seen as a task to find the motility parameters that can best predict the genotypes. To find the motility parameters that can best predict the genotype, having a benchmark for the best accuracy can help understand how much information is not captured by these motility parameters. The gap between the benchmark and prediction accuracy made by each motility parameter provides some insights on how much "information" at most can be squeezed from the data for this task. Apart from this reason, it is also learned from the previous results that raw data that keep the sequential information tend to do a better job in training the model to classify the genotypes. Therefore, benchmark of accuracy for this task is using RNN model with the raw coordinates sequence data.

### Result 3.1: Raw sequences yields better accuracy than extracted features.

The presentation of the task decides the structure of the model; therefore, it also decides the prediction accuracy in the end. As a multi-class classification task, it can also be seen as a one-to-many binary classification problem, so both 5-classes and binary classification models are tested. As shown in figure 25, there are 2 architectures A and B. They share very similar model structure except the very top layer for computing probability for each class. In architecture A, there are 5 nodes for 5-class classification while in architecture B, there are only 2 nodes for binary classification. Both models take both velocity sequence and the angular velocity sequence as input. The data flows from the bottom to the top. The first part of this model is LSTM (Long Short-Term Memory) with 2 features (velocity and angular velocity) and 60 timesteps. These two features are processed separately until they are concatenated together and flow into the fully connected layers in the top of model. Dropout and the normalization layers are used. Note these two models are only used for the sequential data for benchmark accuracy. To make prediction using the motility parameters, a different and simpler model needs to be used.

Figure 26 shows that model trained with raw sequence yields ~91% prediction accuracy(~91%) which is much higher than that of any models that are trained by the manually selected motility parameters as models, even when all these motility parameters are used. On one hand, this superior result by using raw data is not surprising given how much the information is compressed from a sequence to a single value. On the other hand, the absolute value of this prediction accuracy using raw data is surprisingly high considering the best accuracy of using transformed sequential data is only 79% in the first project.

Note that the ground truth of the labels are genotypes rather than their behaviors. There is a gap between the label (genotype) and the prediction(phenotype). The mapping between the label and prediction is not guaranteed to be a one-to-one relationship. It is possible that two different genetic mutants behave same or there are different phenotypes inside the same genotype. Besides, the ground truth of behavior can never be fully known due to the digitalization of data or Nyquist theorem. The prediction using raw data

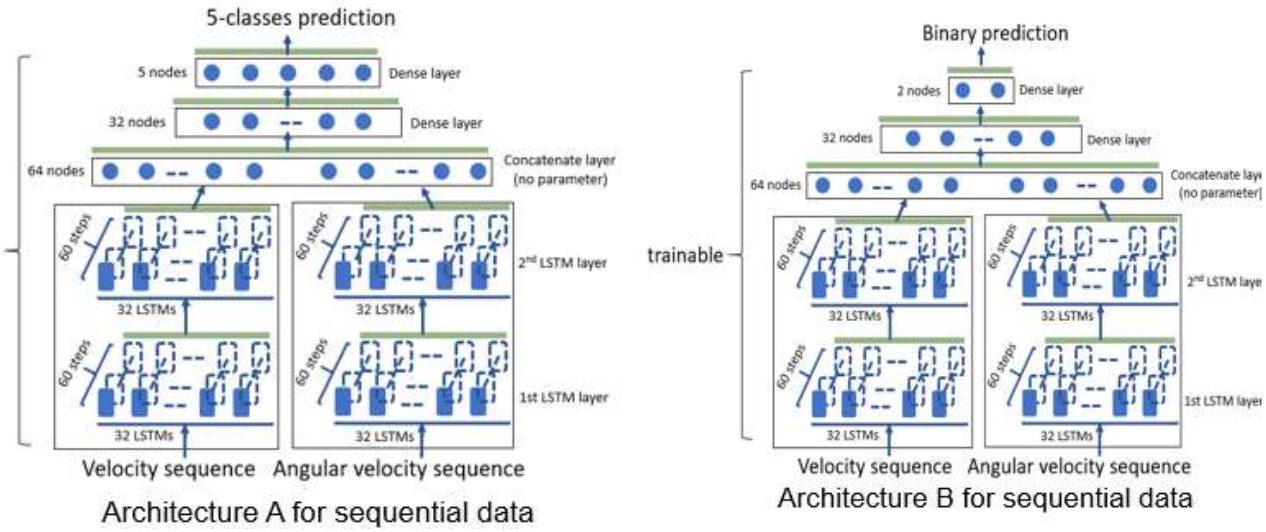



Figure 25 Neural network architectures used for raw data.

The data flow from the bottom to the top and go through LSTM and then fully connected layer. These two architectures share similar structure except the number of nodes in the top layer. The architecture on the left is for 5-class classification while the architecture on the right is for binary classification.

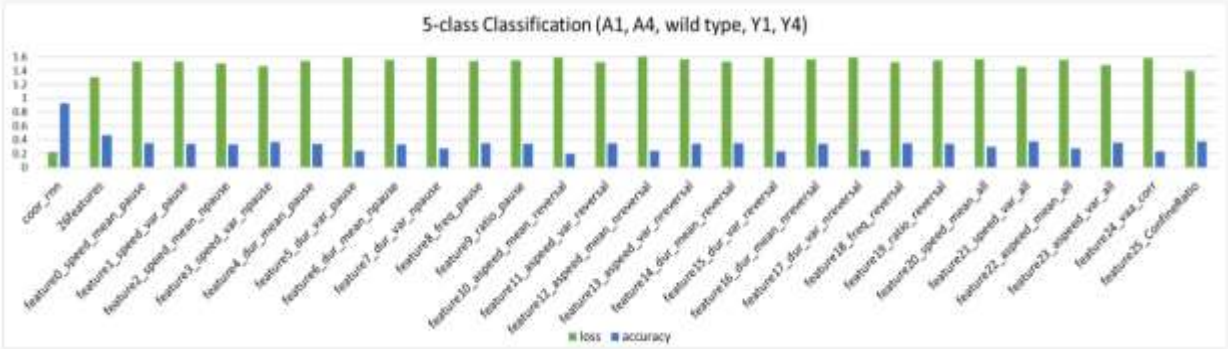



Figure 26 Prediction accuracies of 5-class classification task.

Raw sequences yield much higher accuracy than using any single or combined motility parameters.

provides a relative benchmark to evaluate how much relevant information is in the features selected by domain knowledge. The result also confirms a common case in deep learning area that raw data generally gives better results than manually selected features (although this probably comes with the drawback of longer training time). Deep learning models are well known for their ability in feature selection. At the stage of this study, it is not known what features are selected by this model, but it has been reported to visualize the weights of the nodes and try to infer the feature from it [38].

Result 3.2: Best features are consistent among different binary tasks.

As shown in figure 26, among all the models that only use motility parameters, the highest accuracy of ~47% is unsurprisingly made by the model using all features. This is 9% higher than the best accuracy of ~37% made by single motility parameter. With 25 times more features, the prediction accuracy when using all features only increased 9%. The increase of the accuracy is not proportional to the increase of number of features, this implies there are a lot of overlapped information between these features. To quantify this overlapping, the average correlation between each motility parameter with all the other motility parameters are calculated using formula 18. This overall correlation for motility parameter is calculated in two steps: First, the correlations between this motility parameter and all other motility parameters are calculated. Then the square root of the average of all the previous correlation is the overall correlation for this motility parameter. The overall correlations for each motility parameter are shown in figure 28. Note that the high overall correlation of a motility parameter does not indicate it to have good predicting power for the classification task. It only indicates this motility parameter is well correlated with other motility parameters.

$$overall\ correlation\ for\ feature\ n = \sqrt{\frac{\sum_{i=1,i!=n}^{26} PairCorrelation_i^2}{26}} \tag{18}$$

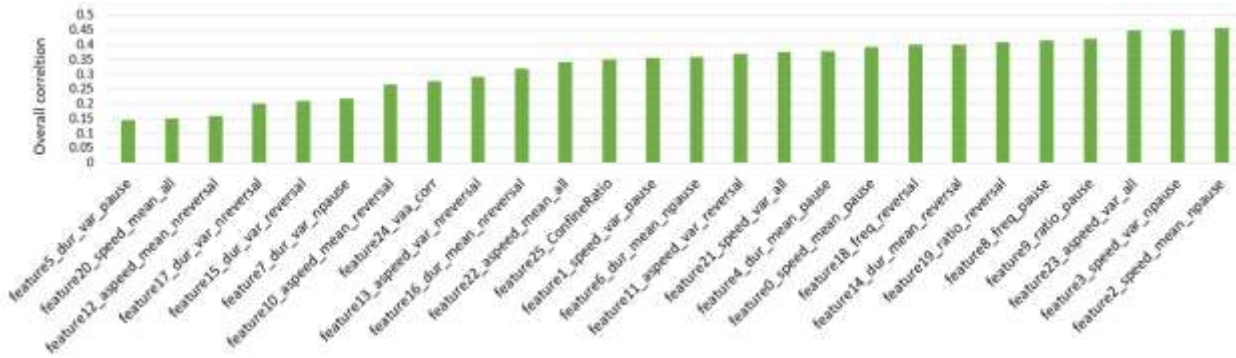



Figure 27 Average correlation of each motility parameter.

Higher value indicates the motility parameter contains more overlapping information with other motility parameters.

As is shown in figure 28, the most predicting single features are very consistent across the tasks of binary classification between each class and the wild type. The best features are feature13: aspeed_var_nreversal, feature2: speed_mean_npause, feature15: dur_var_reversal and feature17: dur_var_nreversal. Since all the binary classification models are classifying wildtype against one of the mutant genotypes, this consistency may indicate these features are the difference between the wildtype against all mutant genotypes.

Feature 13 is the variance of angular speed during non-reversal runs. Lower value of feature13 indicates the trajectory of the non-reversal runs have a more consistent curvature. The trajectory would look more homogenous in this way. In contrast, there is another complementary motility parameter: the variance of angular speed of reversals or feature11. The prediction made by using feature11 is almost 10% lower than those based on feature13. Given the fact that mutants are deficit in regulating their movement, it is reasonable to think that the high predicting power of feature13 may imply that mutants generally have lower variance of angular speed than the wildtype. The second motility parameter that gives best predictions is the average speed of non-pause runs or feature2. Given that feature2 is the motility parameter that has highest overall correlations with every other motility parameters, it is very interesting to see that feature3, which also has very high overall correlations with other motility parameters, did very bad in this binary classification task. Feature3 predicts around 20% worse compared to feature2. This may imply that feature2 and feature3 represent two groups of motility parameters that have

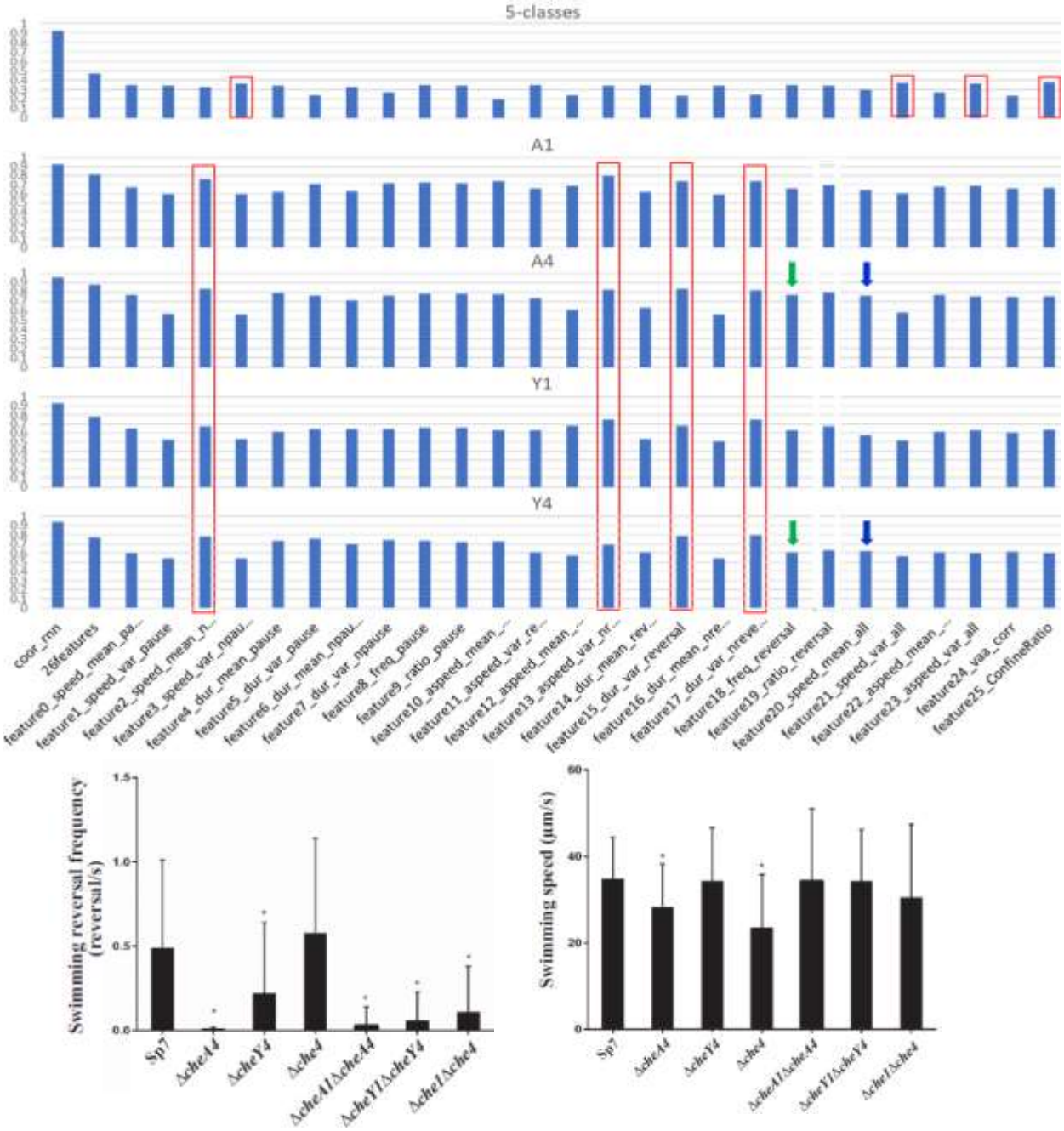



Figure 28 Best features for genotype prediction.

On the top, the bars circled by red boxes are most differentiating motility parameters shared by all binary classification tasks. The bars pointed by the green and blue arrows agree with the previous report that the average swimming speed and reversal frequency of mutants lacking cheA4 cheY4 are lower than the wildtype, but the difference between mutant ΔA4 and wildtype is larger than that between ΔY4 and the mutant. On the top are previous report on the motility parameters of the mutants.

very different trend in their distribution. It is expected that motility parameters that are correlated to feature2 tend to be a better predictor than those correlated more with feaature3. Another thing worth noting is, across the 4 binary classification tasks, feature2 is relatively effective in differentiating the wildtype and mutant ΔY1. This is a bit surprising. Che1 is known for controlling the swimming speed. While the wild type cells swim transiently faster about 5 μm/sec upon air addition, the ΔA1 and ΔcheY1 mutant cells swam with a reduced velocity under these conditions. However, no air was added for this dataset. Mutants lacking cheA1 and cheA4 have lower swimming sped. The last two features that give best predictions are variance of duration of reversal and non-reversal runs. It is known that mutants lacking cheA4 or cheY4 have lower frequency of reversals. This may translate to longer non-reversal durations.

Previous paper has reported that mutant lacking ΔA4 has most different reversal frequency and swimming speed than the wildtype compared to ΔY4. Similar result can be observed by the data pointed by the green arrows and blue arrows. The bars of ΔA4 are higher than the bar of ΔY4.

Result 3.3 : Best features for differentiating 5 genotypes are different than the best features for binary classification.

The results from binary classification models strike the contrast with the result of 5-classes model by having totally different group of features that are most useful for the prediction. The best features from 5-classes model are feature 3: speed_var_npause, feature 21: speed_var_all, feature 23: aspeed_var_all and feature 25: ConfineRatio. They are entirely different that those from the binary classification models. Since 5-class model needs to distinguish one mutant from another mutant while the binary model does not (it only needs to distinguish mutant from wildtype), the top features selected from 5-class model are likely the features that represents more the difference between the mutants than the features from binary classification model.

These top 4 motility parameters for 5-class classification tasks have very close prediction accuracy around 35%. Interestingly, feature3, which is compared to feature2 before, is not a very good feature for binary classification but is a good feature for differentiating all 5 classes together. This implies the variance of speed in non-pause runs varies wide among the mutants. Feature21 is the variance of speed. It is known that all mutants have a reduced average speed compared to the wildtype. As one of the best features to differentiate all the genotypes, this indicates the speed is reduced to different levels among the mutants. Feature23 is the variance of angular speed. On one hand, it seems reasonable thinking that mutants are deficit in regulating their movement, it is possible they tend to stay on a fixed angular speed without intervention and therefore have lower angular speed variance than the wildtype. On the other hand, the fluctuation of the liquid

in the environment may intervene their angular speed and the active regulation may server as a stabilizer. If this is true then the wildtype would have lower variance than the mutants. The last feature that is more useful in differentiating the five classes is confine ratio or feature25. Confinement ratio is calculated as the ratio of net distance and the absolute distance. High value of confinement ratio indicates the overall trajectory is lined up to one direction and the movement is very efficient in terms of moving towards a targeted direction. Since the mutants are deficit in regulating their movement. As shown in study [31], the trajectories of mutants lacking CheY4 or CheA4 seem more straight compared to the wildtype, but the motility parameter may be subject to the length of the sequential data. A movement with high confine ratio of 0.5 seconds duration may have low confine ratio of longer duration.

## Discussion and Conclusion

With the fast process made by machine learning researchers and computational power industry, the application of machine learning, especially deep learning is an unavoidable trend happening in every area of our lives. This project explored some of such scenarios in microbiology research and encountered some problems that may be addressed in the future. More and more models become open source. Even the pretrained models that took a lot of computation power and time become downloadable. The availability of these powerful models promoted transfer learning and fast implementation of the result from the newest machine learning research. This leaves the difficulties only in understanding the techniques and the quality of data.

In chapter one, the raw data is time series. Models such as LSTM can process these data properly. Due to the lack of knowing such models, the data were transformed into regular data first before input into any machine learning model. This unnecessary transformation causes an information loss in the data. Unsurprisingly the neural network in chapter 1 only reached 79% accuracy. Although it is the highest already compared to the rest of the machine learning methods here, it is 12% lower than the result from chapter 3. Although they were given exact same data, due to the improper use of models, the accuracy of prediction changed dramatically. This means that any claim on accuracy using a given dataset has to be examined with the model they used.

In chapter two, another very interesting direction to study would be reconstructing the trajectories from the videos rather than just classifying the genotypes. The problem for this is the lack of ground truth. This lack of ground truth or limitation in getting the labeled data is three-fold. First, video recording itself is a discretization of the natural process. The speed of bacteria movement is between 2-1000 um per second, this does not tell the frequency of any movement pattern. However, if it is compared to the common frequency of video frames, which is 25 or 30 frames per second, it is possible that the bacteria have

some movement patterns with frequency higher than this. If that is the case, then it will never be captured. Second, the bacteria are living in 3D world but the video camera can only capture the information as an image. When the bacteria swim out of the focus plane for the camera, all the information lost. Apparently, this is another big limitation for this project. There have been 3D videos of cells reported, but there are still many technical hinderance to overcome. If the final predication accuracy holds the highest concern, rather than the development of machine learning model itself, the effort or resources may be better rewarded when they are given to improve this hardware level issues. The last issue more attackable, yet not addressed in current project. They are better 2D video camera, better trajectory tracking automation software and more labor expense in labeling the trajectory data manually.

The last chapter reflected the gap between explanation and prediction utilities of machine learning models. Although the models can predict well for a complex task only after a very short time of training, it does not provide the explanation or meaning of the data. Neural network is most thought as a black box for its process of making inference. However, many researched have been done trying to tackle this problem. For example, feature visualization is a technique to see the middle layers of the model in the hope to understand how the raw data is processed into the probabilities for classification. This technique can give more intuition on what features are extracted at each layer and, hopefully, provides more explanation on how the classification is made. This may help more to pick out the motility parameter or even suggest a new one that can capture the traits of the bacteria motility and classify their moving patterns.

## References

1. Rawls, John F., Michael A. Mahowald, Andrew L. Goodman, Chad M. Trent, and Jeffrey I. Gordon. "In vivo imaging and genetic analysis link bacterial motility and symbiosis in the zebrafish gut." Proceedings of the National Academy of Sciences 104, no. 18 (2007): 7622-7627.
2. Chaban, Bonnie, H. Velocity Hughes, and Morgan Beeby. "The flagellum in bacterial pathogens: for motility and a whole lot more." In Seminars in cell & developmental biology, vol. 46, pp. 91-103. Academic Press, 2015.
3. Josenhans, Christine, and Sebastian Suerbaum. "The role of motility as a virulence factor in bacteria." International Journal of Medical Microbiology 291, no. 8 (2002): 605-614.
4. Glagolev, A. N., and V. P. Skulachev. "The proton pump is a molecular engine of motile bacteria." Nature 272, no. 5650 (1978): 280-282.
5. Lemon, Katherine P., Darren E. Higgins, and Roberto Kolter. "Flagellar motility is critical for Listeria monocytogenes biofilm formation." Journal of bacteriology 189, no. 12 (2007): 4418-4424.
6. Martinez, Vincent A., Jana Schwarz-Linek, Mathias Reufer, Laurence G. Wilson, Alexander N. Morozov, and Wilson CK Poon. "Flagellated bacterial motility in polymer solutions." Proceedings of the National Academy of Sciences 111, no. 50 (2014): 17771-17776.
7. Armstrong, Peter B. "The control of cell motility during embryogenesis." Cancer and Metastasis Reviews 4, no. 1 (1985): 59-79.
8. Kearns, Daniel B. "A field guide to bacterial swarming motility." Nature Reviews Microbiology 8, no. 9 (2010): 634-644.
9. Leifson, Einar. 1960. "Atlas of bacterial flagellation." Atlas of bacterial flagellation 171.
10. Magariyama, Y., S. Sugiyama, K. Muramoto, Y. Maekawa, I. Kawagishi, Y. Imae, and S. Kudo. "Very fast flagellar rotation." Nature 371, no. 6500 (1994): 752-752.
11. Aldridge, Phillip, and Kelly T. Hughes. "Regulation of flagellar assembly." Current opinion in microbiology 5, no. 2 (2002): 160-165.
12. Zhulin, I. B., and JUDITH P. Armitage. "Motility, chemokinesis, and methylation-independent chemotaxis in Azospirillum brasilense." Journal of bacteriology 175, no. 4 (1993): 952-958.
13. Theves, Matthias, Johannes Taktikos, Vasily Zaburdaev, Holger Stark, and Carsten Beta. "A bacterial swimmer with two alternating speeds of propagation." Biophysical journal 105, no. 8 (2013): 1915-1924.
14. Wikipedia contributors, "Flagellum," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Flagellum&oldid=964838618 (accessed July 7, 2020).
15. Jones, Christopher J., and Shin-Ichi Aizawa. "The bacterial flagellum and flagellar motor: structure, assembly and function." In Advances in microbial physiology, vol. 32, pp. 109-172. Academic Press, 1991.
16. Stearns, Jennifer, and Michael Surette. Microbiology for dummies. John Wiley & Sons, 2019.

17. Theves, Matthias, Johannes Taktikos, Vasily Zaburdaev, Holger Stark, and Carsten Beta. "A bacterial swimmer with two alternating speeds of propagation." Biophysical journal 105, no. 8 (2013): 1915-1924.
18. Cohen, Andrew R., Francisco LAF Gomes, Badrinath Roysam, and Michel Cayouette. "Computational prediction of neural progenitor cell fates." Nature methods 7, no. 3 (2010): 213-218.
19. Sacan, Ahmet, Hakan Ferhatosmanoglu, and Huseyin Coskun. "CellTrack: an open-source software for cell tracking and motility analysis." Bioinformatics 24, no. 14 (2008): 1647-1649.
20. Schoenauer Sebag, Alice, Sandra Plancade, Céline Raulet-Tomkiewicz, Robert Barouki, Jean-Philippe Vert, and Thomas Walter. "A generic methodological framework for studying single cell motility in high-throughput time-lapse data." Bioinformatics 31, no. 12 (2015): i320-i328.
21. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521, no. 7553 (2015): 436-444.
22. Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.
23. Deng, Li, and Dong Yu. "Deep learning: methods and applications." Foundations and trends in signal processing 7, no. 3–4 (2014): 197-387.
24. LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series." The handbook of brain theory and neural networks 3361, no. 10 (1995): 1995.
25. Bengio, Yoshua, Yann LeCun, and Donnie Henderson. "Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden Markov models." In Advances in neural information processing systems, pp. 937-944. 1994.
26. Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115, no. 3 (2015): 211-252.
27. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.
28. Kimmel, Jacob C., Amy Y. Chang, Andrew S. Brack, and Wallace F. Marshall. "Inferring cell state by quantitative motility analysis reveals a dynamic state system and broken detailed balance." PLoS computational biology 14, no. 1 (2018): e1005927.
29. Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." arXiv preprint arXiv:1212.0402 (2012).
30. Yao, Li, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. "Describing videos by exploiting temporal structure." In Proceedings of the IEEE international conference on computer vision, pp. 4507-4515. 2015.

31. Mukherjee, Tanmoy, Dhivya Kumar, Nathan Burriss, Zhihong Xie, and Gladys Alexandre. "Azospirillum brasilense chemotaxis depends on two signaling pathways regulating distinct motility parameters." Journal of bacteriology 198, no. 12 (2016): 1764-1772.
32. Bible, Amber, Matthew H. Russell, and Gladys Alexandre. "The Azospirillum brasilense Che1 chemotaxis pathway controls swimming velocity, which affects transient cell-to-cell clumping." Journal of bacteriology 194, no. 13 (2012): 3343-3355.
33. Prasatha, V. S., Haneen Arafat Abu Alfeilate, A. B. Hassanate, Omar Lasassmehe, Ahmad S. Tarawnehf, Mahmoud Bashir Alhasanatg, and Hamzeh S. Eyal Salmane. "Effects of distance measure choice on knn classifier performance-a review." arXiv preprint arXiv:1708.04321 (2017).
34. Prasath, V. B., Haneen Arafat Abu Alfeilat, Omar Lasassmeh, Ahmad Hassanat, and Ahmad S. Tarawneh. "Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier--A Review." arXiv preprint arXiv:1708.04321 (2017).
35. Alexandre, Gladys, Suzanne E. Greer, and Igor B. Zhulin. "Energy taxis is the dominant behavior in Azospirillum brasilense." Journal of Bacteriology 182, no. 21 (2000): 6042-6048.
36. Graves, Alex, and Jürgen Schmidhuber. "Offline handwriting recognition with multidimensional recurrent neural networks." In Advances in neural information processing systems, pp. 545-552. 2009.
37. Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.
38. Gilpin, Leilani H., David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. "Explaining explanations: An overview of interpretability of machine learning." In 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA), pp. 80-89. IEEE, 2018.
39. Milo, Ron, and Rob Phillips. Cell biology by the numbers. Garland Science, 2015.
40. Harshey, Rasika M. "Bacterial motility on a surface: many ways to a common goal." Annual Reviews in Microbiology 57, no. 1 (2003): 249-273.

## Appendix

The motility parameters are categorized into 4 groups: basic, regression, sequence, event. Since the raw data is a time sequence of coordinates, all parameters are encouraged to be expressed as $f(x, y, t, w, f)$ where $x, y$ are coordinates (e.g. 10 millimeters), $t$ is time (e.g. time step 3), $w$ is window size (e.g. 90 time steps), $f$ is frame rate (e.g. 30 frames/sec).

*is less preferred due to their high sensitivity to hyper-parameter w. **is subject to other hyper-parameters.

**basic:**

1. *Total distance: $d_{tot}$

$$\sum_{t=0}^{w} \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2}$$

2. *Net distance: $d_{net}$

$$\sqrt{(x_w - x_0)^2 + (y_w - y_0)^2}$$

3. *Maximum distance:

$$max(\sqrt{(x_{t+1} - x_0)^2 + (y_{t+1} - y_0)^2})$$

4. Confinement ratio:

$$\frac{d_{net}}{d_{tot}}$$

5. Mean speed: $v_{mean}$

$$\frac{d_{tot}}{w/f}$$

6. Mean straight-line speed: $v_{straightline_mean}$

$$\frac{d_{net}}{w/f}$$

7. Mean angular speed: $v_{angular_mean}$

$$\frac{f}{w} \sum_{t=0}^{w} arctan(\frac{y_{t+1} - y_t}{x_{t+1} - x_t})$$

8. Speed variance:

$$\frac{f^2}{w} \sum_{t=0}^{w} \left(\sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} - v_{mean}\right)^2$$

9. Angular speed variance:

$$\frac{f^2}{w} \sum_{t=0}^{w} \left(arctan(\frac{y_{t+1} - y_t}{x_{t+1} - x_t}) - v_{angular_mean}\right)^2$$

10. Forward progression linearity

$$\frac{v_{straightline_mean}}{v_{mean}}$$

11. **Mean absolute displacement with hyper-parameter $n$:

$$\frac{1}{w-n}\sum_{t=0}^{w-n}\sqrt{(x_{t+n}-x_t)^2+(y_{t+n}-y_t)^2}$$

12. **Mean squared displacement with hyper-parameter $n$:

$$\frac{1}{w-n}\sum_{t=0}^{w-n}((x_{t+n}-x_t)^2+(y_{t+n}-y_t)^2)$$

**regression:**

13. **Slope of linear least squares fit for $log(MSD)$ along hyper-parameter time shift: $slope_{MSD_timeshift}$

$$log(MSD) = slope_{MSD_timeshift} * log\left(\frac{n}{f}\right) + b$$

Note $\frac{n}{f}$ is time shift, it can also be expressed as $\Delta n \Delta t$ where $\Delta n$ is the number of time steps and $\Delta t$ is the time for each step. $b$ is trivial.

14. **Scaling coefficients with hyper-parameter $n$ and norm $v$: $\gamma_v$

$$\frac{log(\frac{\mathbf{1}}{w-n}\sum_{t=0}^{w-n}|(x,y)_{t+n}-(x,y)_t|^v)}{log(\frac{n}{f})}$$

15. **Slope of linear least squares fit for scaling coefficients $\gamma_v$ along hyper-parameter norm $v$: $slope_{scaling_norm}$

$$\gamma_v = slope_{scaling_norm} * v + b$$

$b$ is 0 usually. The plot of $\gamma_v$ against $v$ is called Moment Scaling Spectra (MSS).

**sequence:**

16. Correlation coefficient of speed $v_t$ and angular speed $u_t$

$$\frac{\sum_{t=0}^{w} v_t u_t}{\sqrt{\sum_{t=0}^{w} v_t^2 \sum_{t=0}^{w} u_t^2}}$$

17. Kullback–Leibler divergence ($v_t$ and $u_t$ should be normalized first)

$$\int_{v_min}^{v_max} p_{speed}(v) log(\frac{p_{speed}(v)}{p_{angularspeed}(v)})\, dv$$

18. Bhattacharyya distance

$$\int_{v_min}^{v_max} \sqrt{p_{speed}(v) p_{angularspeed}(v)} dv$$

**event:**

All parameters in basics and regression groups can be applied to the data filtered by the corresponding criteria for different events.

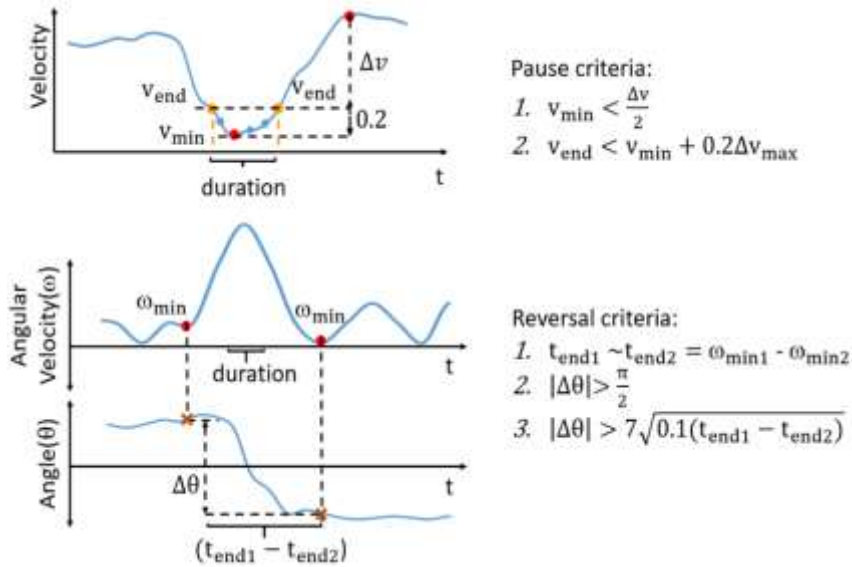


Figure 29 Criteria of Events

**example criteria on speed:

$$\frac{\Delta v}{v_{min}} \geq \alpha \tag{5}$$

$$\Delta v = \max(v_{pre} - v_{min}, v_{after} - v_{min})$$

where $v_{pre}$ and $v_{after}$ are local maxima before and after this local minima of the absolute velocity**.** $\alpha$ is the threshold (e.g. 0.7).

Similarly, **example criteria on direction:

$$\frac{|\Delta\theta_{max} - \Delta\theta|}{\Delta\theta_{max}} \leq \beta \tag{6}$$

$\beta$ is the threshold (e.g. 0.2).

Besides basic parameters, event-based parameters can also include:

1. Pause steps ratio

$$\frac{\sum_{i=1}^{n_total} n_steps_each_pause_i}{w}$$

2. Mean pause duration $n_steps_pause_{mean}$

$$\frac{1}{n_{total}} \sum_{i=1}^{n_total} n_steps_each_pause_i$$

3. Variance of pause duration

$$\frac{1}{n_{total}} \sum_{t=0}^{w} (n_steps_pause_i - n_steps_pause_{mean})^2$$

4. Similar parameters apply for reversal event

## Vita

Yue Ma studied pharmacology in her undergraduate. From 2012 to 2015, she researched nuclear receptors for drug development. In 2020, she finished her master's degrees in statistics and biology from The University of Tennessee.