Predicting the Presence of Bacterial Biofilm, Using Bacterial Drop Colony Assay Images

Paulina Chamely (pchamely), Rachel Adenekan (adenekan), Daiane Aizen (daizen)

1. Abstract:

Biofilm has been known to play a strong role in the pathogenicity of bacteria, enhancing the virulence of these pathogen [1]. In a clinical setting, the faster clinicians know if an infectious bacterial strain produces biofilm, the better they can tailor the treatment of their patients' infection to match its aggressiveness. We aim to build a classifier that can accurately identify the presence of biofilm from image data obtained from microscope images of bacterial colonies growing on an agar plates with the hope that it could be used as a tool to detect biofilming bacteria as quickly as possible. To do this, we trained a support vector machine (SVM) and two convolutional neural networks (CNNs) on a dataset consisting of pathogenic, biofilm forming bacterial strains of B.cereus as well as pathogenic non-biofilm forming bacterial strains. Our top model, the SVM, achieves an accuracy of 100% and an F1 score of 1 on the 2-category classification problem.

2. Introduction:

Paulina, one of our team members, has been conducting related research and cultured a large set of B. cereus bacterial strains that were isolated from patients that suffered from a range of systemic and local infections. On culturing this cohort of B.cereus strains, many were observed to develop biofilm. Bacterial biofilms are structured communities of densely packed bacterial cells that adhere to living/inert surfaces and are embedded in a self-produced matrix of extracellular polymeric substance [1]. Biofilms acts as a protective barrier allowing for the cells inside to become more resistant to the body's natural antimicrobials and other administered antibiotics. Resultantly, biofilm often indicates a very pathogenic bacterial strain.

Biofilm is often visible to the eye after culturing for a few days and current biofilm assays take advantage of this. Many involve culturing and incubating for 3-5 days to allow for a detectable amount of Biofilm to form, such that a more accurate assessment of biofilm presence can be made. In a clinical setting, the faster clinicians know if an infectious strain produces biofilm, the better they can tailor treatment of their patients infection. With this in mind, our aim is to utilize this dataset of pathogenic, biofilm and non-biofilm forming bacterial strains of B.cereus to train a classifier that can accurately identify the presence of biofilm. The inputs to our algorithm are microscope images of bacterial colonies. We then use SVM and convolutional neural networks to output a predicted bacteria type ("with biofilm" or "without biofilm").

3. Related Work:

One aspect that makes our project quite exciting is that groundbreaking advances have been made in the biofilming bacteria classification field in the past few months. In the most recent, and now state-of-the art, study, researchers used a TensorFlow CNN to outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition (~ 90% accuracy vs ~ 50% accuracy). A major strength of this approach is that a low number of microscopy images (< 600) per category was sufficient for high accuracy. A limitation of this study is that the technique was applied to bioleaching bacteria who form simpler biofilms, so the method might have limited performance when applied to other environmental samples with higher diversity. Despite this, the approach provides an alternative to standard time-consuming biochemical measurements.[2] This research was quite similar to our project and served as inspiration for our methods. Seeing how this network provided such positive results, we were eager to apply CNNs to our simpler problem of classifying bacteria as with biofilm or without biofilm.

Similarly, Zielinski et al. 2017 used deep CNNs to obtain bacterial image predictors which are then encoded and classified with either SVM or Random Forest. They used texture recognition so that they could classify using more features than just the shape of the bacteria, and the CNN enabled them to include features learned automatically, based on millions of images. This method was quite effective, as they were able to use a dataset of only 660 images to classify 33 different species of bacteria with an accuracy of 97.24%. This is greater than most previous work which was only able to classify a few species of bacteria. The authors do note that performance could have been improved if information on the color distribution of the images was extended.[3] We took inspiration from this approach by using both a CNN and an SVM, and by using one of the same CNNs they used (AlexNet).

Another related approach is that of Ferrari et al 2016 in which they use CNNs and SVM to count bacterial colonies from images. In addition to the algorithms, we took note of how they handled expanding their dataset, namely using image augmentation such as flips and other transformation. This is of particular interest to us since our original dataset was quite small. While their algorithm is able to

provide accurate counts with reliable outlier detection, we note than additional tools would be necessary to detect extended zones of confluent growth (where the colony forming unit is no longer visually discernible).[4]

Building upon the strengths of the various algorithms, we decided to use both CNNs and and SVM models with image augmentation for our prediction task, classifying bacteria as either "With Biofilm" or "Without Biofilm."

4. Dataset:

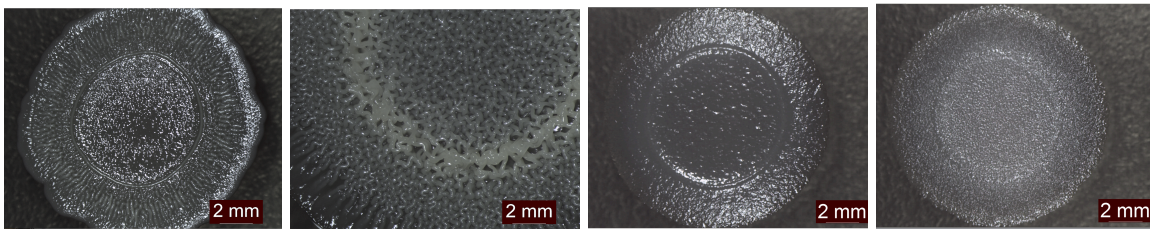

Figure 1: Example Images for the classification task: First two images with biofilm and the last two images without.

Data Collection: Our dataset consists of a collection of microscope bacterial drop colonies images from a set of 64 isolates of Bacillus cereus bacteria, a subset of which produce biofilm. For each of the 64 isolates, longitudinal images were taken to capture the gradual development of biofilm. A total of 213 images were obtained, with 27 of these classified as containing biofilm.

Data Augmentation: In order to obtain more data and balance the distribution between the positive and negative cases, artificial data synthesis was employed to "create" new images by transforming the ones already available. This was done using the *ImageDataGenerator* function of the Keras python deep learning library[5]. Image transformation involved image rotations, mirroring, resizing and re-scaling and random shifts in width and height. We created a final dataset of 400 biofilm positive examples and 405 biofilm negative examples. Prior to training the classifiers, all of the images were resized to the size required for the respective convolutional neural network packages used (227 X 227 pixels for AlexNet and 224 x 224 pixels for GoogleNet and the SVM) .

Training, Validation and Test sets: The images were randomly assigned to a training set, a validation set and a test set. For the both the SVM and the CNN's the data was split by a 60%-20%-20% ratio for the training, validation and test sets, respectively.

5. Methods:

To develop the ideal model for this project we built two different models and chose the one that performed the best. First we trained a simple Convolutional Neural Network (CNN) as a baseline model to identify the ideal parameters to be used to train another more complex and accurate CNN. Next, we built an SVM and compared its results with our CNN model to choose the optimal model in classifying our specific set of data.

5.1 Baseline (CNN Model - AlexNet)

We used a CNN as one of our main models as they are often the model of choice for imaging applications. Furthermore, the exact features of biofilm formation can be difficult to identify and so CNNs are ideal - they don't assume prior knowledge of features and instead can extract the training features from the images on its own.

To build our baseline model, we used the AlexNet CNN package as it is one of the fastest pre-trained CNN[6]. AlexNet is a pretrained CNN that is 8 layers deep. The first 5 layers are convolutional layers and the last 3 layers are fully connected layers. In between there are pooling and activation layers. The last 3 layers can be re-trained to classify new images with iterative transfer learning. Iterative Transfer Learning allows us to maintain all that was learned in the initial layers of a network and retrain just the last few layers of the network with our own training data such that the network adapts to our dataset and can classifying our images.[7] By using the fastest CNN we were able to quickly finetune the network with various data pre-processing steps and training options, to learn the optimal parameters to use when training the more complex and accurate pre-trained CNN, GoogLeNet.

5.2 CNN Model - GoogLeNet

Like AlexNet, GoogLeNet is a pre-trained CNN but its architecture is much more complex, as it is 22 layer deep and employs the unique "Inception module" level of organization. An inception module/cell represents a good local network topology (or sub-network) that is made up of convolutional and pooling layers, and these cells are then stacked on top of and connect to each other to create the more complex and deep overall CNN (Figure 3.). As seen in Figure 3 there are some intermediate softmax branches off of the middle cells that are used for training purposes and the final classification layer also employs the softmax function.[8]

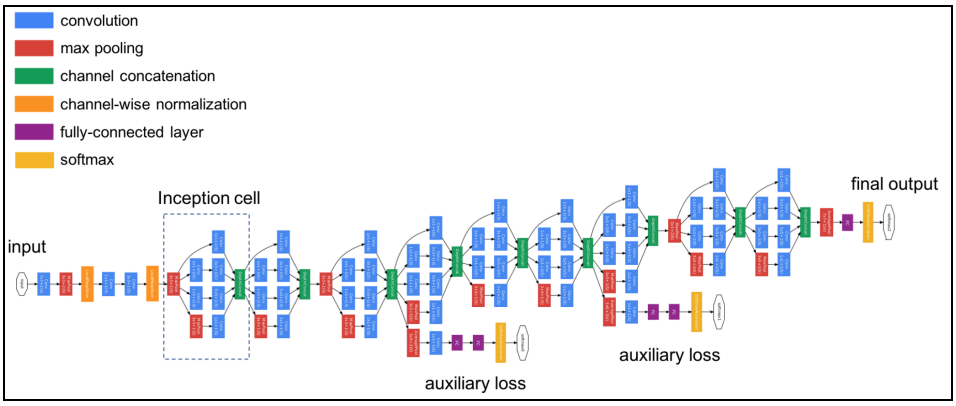

Figure 3. The architecture of GoogLeNet with multiple stacked Inception cells, making up a total of 22 layers[9]

The final 2 layers of this network consist of the last fully connected "learnable" layer and the final classification layer (softmax function) used to classify the input image. With iterative transfer learning, we re-trained the CNN by replacing the final two layers with new layers adapted to our data set, freezing the learned "weights" of the first 10 layers to retain that learned information and finally retraining the network (i.e. the non-frozen layers) on our own image data.

5.3 SVM Model

SVM

The SVM algorithm was performed with MATLAB's built in function fitcsvm.[10] The images were converted into greyscale images to limit the number of features in each training example. The SVM built in function that was used was:

SVM=fitcsvm(X, Y, 'KernelFunction', 'gaussian', 'OptimizeHyperparameters', 'auto', 'HyperparameterOptimizationOptions', struct('Holdout',0.25));

The X and Y input variables correspond the the matrix containing the training examples and their features and the vector containing the classes for each example. The {'KernelFunction', gaussian} "Name,Value" pair refers to the kernel function that is used to compute new features that will allow for a higher complexity algorithm to be trained. The kernel function chosen is the Gaussian kernel function, in which landmark points are artificially created and the new features are a measure of the similarity between the data points and the landmark points as shown in Equation 1.

Equation 1: $f_1 = similarity(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

The hypothesis of the SVM learning algorithm is now a function of the new features (f_i) instead of the original features, and it will predict a positive example when h(f) >= 0 as shown in Equation 2.

Equation 2: $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

With the 'OptimizeHyperparameters' option, the algorithm attempts to minimize cross-validation error by varying the specified parameters following the command. The 'auto' value utilized indicated that the parameters that are varied are the "BoxConstraint" and

KernelScale". The Box Constraint is a parameter that controls the maximum penalty imposed on margin-violating observations and therefore helps prevent overfitting (a surrogate the the C parameter from the SVM cost function learnt in class). It can take on values in the range [1e-3,1e3]. The Kernel Scale parameter is one with which all of the features in the X matrix are divided by before the kernel function is applied. This parameter is a surrogate to the sigma parameter from the Gaussian Kernel function (Equation 1) learnt in class.

Lastly, in the 'HyperparameterOptimizationOptions', the cross-validation method is specified. The ('Holdout', 0.25) structure indicates that 25% of the training data is left out to form the cross-validation set.[10] 25% of the training set (which represent 80% of the total data set) is actually 20% of the total data set. In this way, both the SVM classifier and the CNN classifier have the same amount of data in each the training, cross-validation and testing sets.

6. Experiments and Results:

6.1 Optimized Experimental Parameters

CNN Models:

Table 1: Parameters and iterations for the CNNs

CNN Model	miniBatch Size	Learning Rate	Epochs	Num Iterations
AlexNet	10	1e-1	4	192
GoogLeNet	10	3e-4	6	288

SNV Model:

Table 2: Optimized Parameters for the SVM classifier

Model	Box Constraint	Kernel Scale	Num Iterations
SVM	0.82114	999.72	30

6.2 Quantitative Evaluation (on test sets):

To evaluate our models, we calculated the accuracy, precision, recall, and F1-scores (when run on our test sets), the length of time taken to train the model as well as the training error. All were computed on the full test set. Since AlexNet was also trained on the initial dataset of smaller size, we also include results from that run.

Table 3: Quantitative comparison between classification methods.

Model	Accuracy (%)	Precision	Recall	F1 Score	Length of Run (mins)	Train error
AlexNet (w/initial dataset)	89.06	0.5455	0.75	0.63	16	0.1094
AlexNet (w/expanded datasets)	94.4	0.9494	0.9375	0.9434	128	0.031
GoogLeNet (w/expanded datasets)	96.25	0.9744	0.9500	0.9620	12	0.012
SVM (w/expanded datasets)	100	1	1	1	10.2	0.000

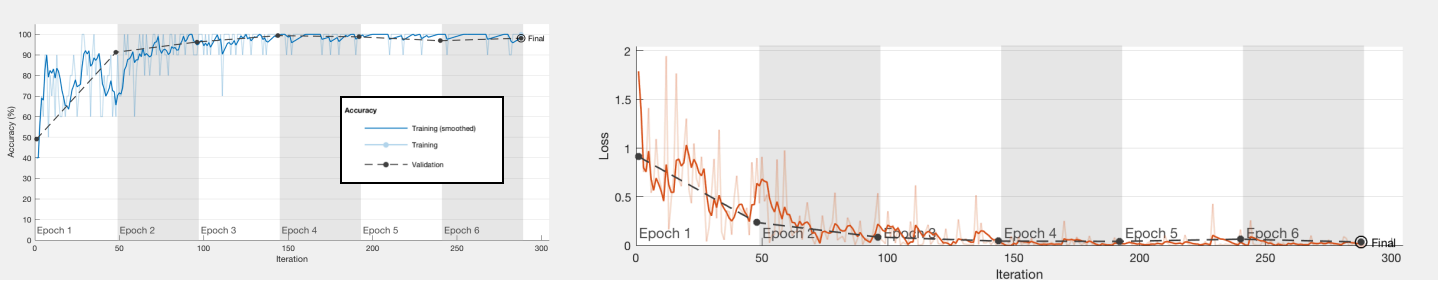

Figure 4: Plot of the training progress of GoogLeNet. This progress is characterized by accuracy (left) and loss (right).

6.2 SVM (top performing model):

Table 4: Confusion MAtrix for SVM model

	Predicted with Biofilm	Predicted without Biofilm
Actually with Biofilm	79 (49.1%)	0
Actually Without Biofilm	0	82 (50.9%)

The SVM model had an accuracy of 100%, a precision of 1, a recall of 1 and an F1 Score of 1, which indicated perfect performance. It is very unlikely that a classification model does a perfect job at predicting the class for new data points. This might have been the case because of the small amount of data available, specifically, due to the small number of examples in the test set. The smaller the test set, the smaller the chances of misclassifying an example. A learning algorithm that is optimized to fit a small data set very well is likely to exhibit very high performance metrics as it was seen here.

7. Discussion/Conclusions/Future Work:

The purpose of our project was to build a classifier that can accurately identify the presence of biofilm from image data obtained from microscope images of bacterial colonies growing on an agar plates with the hope that it could be used as a tool to detect biofilming bacteria. To do this, we trained a support vector machine (SVM) and two convolutional neural networks (CNNs) on a dataset consisting of pathogenic biofilm forming bacterial strains of B.cereus as well as pathogenic non-biofilm forming bacterial strains. Our top model, the SVM, achieves an accuracy of 100% and an F1 score of 100% on the 2-category classification problem. It is likely though that the SVM algorithm only performed that well due to the reduce number of training examples. The smaller the number of examples in the training set, then the less likely it is for the algorithm to misclassify one. The SVM likely performed better than the neural networks because it was the only algorithm that was validated over a wider range of parameters than the CNNs, that were validated with a single learning rate.

With more resources and time, we would look into look into training the network on longitudinal data so that we could create a model that not only detects Biofilm but detects it very early on (before it is very visible to the eye). We would also expand to more classification sets so that we could differentiate between various types of biofilm structures and determine which type of biofilm is present. Moreover, we would like to build learning curves in order to better asses how to improve the classifiers utilized.

8. Contributions:

Rachel Adenekan: Worked primarily on the AlexNet with Iterative Transfer Learning portion of the project.
Paulina Chamely: Worked on the GoogLeNet portion of the project as well as data collection and augmentation.
Daiane Aizen: Worked on the development and implementation of the SVM classifier.
All teammates provided code resources and proposed many strategies to improve the results, and contributed to all deliverables.

References:

1. Sawhney, Rajesh, and Vandana Berry. “Bacterial Biofilm Formation, Pathogenicity, Diagnostics and Control: An Overview.” *Indian Journal of Medical Sciences*, vol. 63, no. 7, 2009, p. 313., doi:10.4103/0019-5359.55113.
2. Buetti-Dinh, Antoine, et al. “Deep Neural Networks Outperform Human Expert's Capacity in Characterizing Bioleaching Bacterial Biofilm Composition.” *Biotechnology Reports*, vol. 22, 2019, doi:10.1016/j.btre.2019.e00321.
3. Zieliński, Bartosz, et al. “Deep Learning Approach to Bacterial Colony Classification.” *Plos One*, vol. 12, no. 9, 2017, doi:10.1371/journal.pone.0184554.
4. Ferrari, Alessandro, et al. “Bacterial Colony Counting with Convolutional Neural Networks in Digital Microbiology Imaging.” *Pattern Recognition*, vol. 61, 2017, pp. 629–640., doi:10.1016/j.patcog.2016.07.016.
5. “Keras: The Python Deep Learning Library.” *Keras Documentation*, keras.io/.
6. “Pretrained Deep Neural Networks. *MATLAB Documentation*, www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html.
7. “Alexnet.” *MATLAB Documentation*, www.mathworks.com/help/deeplearning/ref/alexnet.html.
8. “Classify Image Using GoogLeNet.” *MATLAB Documenation*, www.mathworks.com/help/deeplearning/examples/classify-image-using-googlenet.html#ClassifyImageUsingGoogLeNetExample-3.
9. Szegedy, Christian, et al. “Going Deeper with Convolutions.” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, doi:10.1109/cvpr.2015.7298594.
10. “Support Vector Machines for Binary Classification.” *MATLAB Documentation*, www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html.