**Image Classification of Bacteria Strains as Drug Resistant**

Shimona Manjunath

Table of Contents

[Abstract 3](#_Toc5918622)

[Introduction 4](#_Toc5918623)

[Data 5](#_Toc5918624)

[Preprocessing 6](#_Toc5918625)

[CNN Architecture 8](#_Toc5918626)

[Implementation 10](#_Toc5918627)

[Results 12](#_Toc5918628)

[Conclusion 13](#_Toc5918629)

[Reference 14](#_Toc5918630)

# 

# Abstract

The aim of the project is to perform classification of images of bacteria as drug resistant or non-resistant based on exposure to two drugs. The images are the growth of different strains of the bacteria pseudomonas aeruginosa after being exposed to two drugs tobramycin and carbenicillin. The lag in control growth and growth with exposure to bacteria is used to create the categorical feature. Since the current dataset of images is small (445), numerous data preprocessing techniques were applied to the dataset to create more augmented data points. Cropping, rotation, dividing the images along certain axis, were some of the techniques employed to generate more input from the small dataset. CNN algorithm was implemented to perform classification. The algorithm was modified and fine-tuned to achieve higher classification accuracy. The input images were of varied sizes ranging from 4000 x 4000 pixels to 10000 x 10000 pixels. The images were therefore cropped and resized before being fed into the model. This inherently introduced data loss and will impact the accuracy. The growth of the bacteria is concentric with irregular edges, and don’t provide clear features for the model to learn.

The bacterial strain Pseudomonas aeruginosa is a commonly occurring bacteria which is a multidrug resistant pathogen. This bacterium can lie dormant in 20 - 30% of the population, and can cause skin infections, respiratory infections, and food poisoning. There is an emergence of drug resistant strains in hospitals and clinics, and this is a cause for concern. Despite much research, there is no vaccine for Pseudomonas aeruginosa. Observing virulence factors and phenotypic growth of the bacteria is considered a faster approach versus genetic methods. A study conducted by PhD graduates of Georgia Institute of Technology involves observing the biofilm, the growth of numerous strains of Pseudomonas aeruginosa. They provide the image dataset for the classification process. The conversion of continuous data of growth lag to categorical feature was performed with guidelines from the dataset owners.

# 

# Introduction

The goal of this project is to perform binary classification of images. The images are the result of experiments performed by Sam Brown and Jennifer Rattray, from the microbiology department of Georgia Institute of Technology. The dataset is built and maintained by them. I perform the classification on their data as my project.

For an algorithm to learn and recognize images, a special type of artificial neural network is required, inspired by the brain. Convolutional Neural Network is based on the important operation convolution. Convolutional Neural Networks (ConvNet) are the most favored model for image classification. It is a Deep Learning algorithm, that accepts an input image, assigns significance (learnable weights and biases) to different aspects of the image. The larger underlying idea is that a local understanding of the image will suffice. The algorithm needs to be exposed to a lot of images to be trained well.

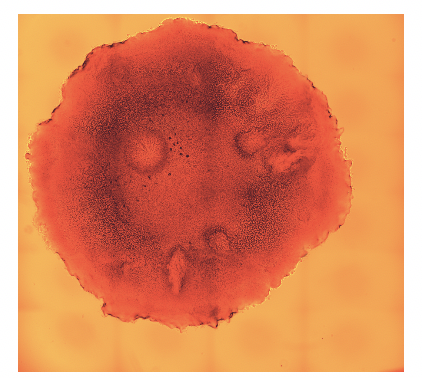
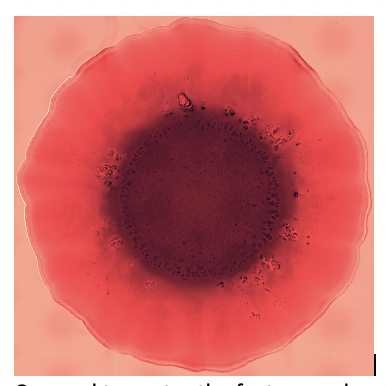
ConvNets have two parts, the hidden layers which perform feature extraction, and the other that performs the classification. There are several layers of convolutions and pooling, which detect and extract the features. The output layer will be reduced to a single vector of probability scores, based on which the classifier will predict the class.

Convolution is one of the main building blocks of a CNN. The term [convolution](http://timdettmers.com/2015/03/26/convolution-deep-learning/) refers to the mathematical combination of two functions to produce a third function. Here it merges two sets of information. The convolution is performed on the input data with the use of a filter or kernel (these terms are used interchangeably) to then produce a featuremap. Convolution is executed by sliding a filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map. Numerous convolutions are performed on the input, where each operation uses a different filter. This results in different feature maps. At the end, all the feature maps are put together as the final output of the convolution layer. An activationfunction is used to make the output of the convolution non-linear. Stride is the size of the step the convolution filter moves each time. Stride size is usually 1, meaning the filter slides pixel by pixel. By increasing the stride size, the filter is sliding over the input with a larger interval and thus has less overlap between the pixels. Because the size of the feature map is always smaller than the input, padding is done to prevent our feature map from shrinking. A layer of zero-value pixels is added to surround the input with zeros, so that the feature map will not shrink. In addition to keeping the spatial size constant after performing convolution, padding also improves performance and makes sure the kernel and stride size will fit in the input.

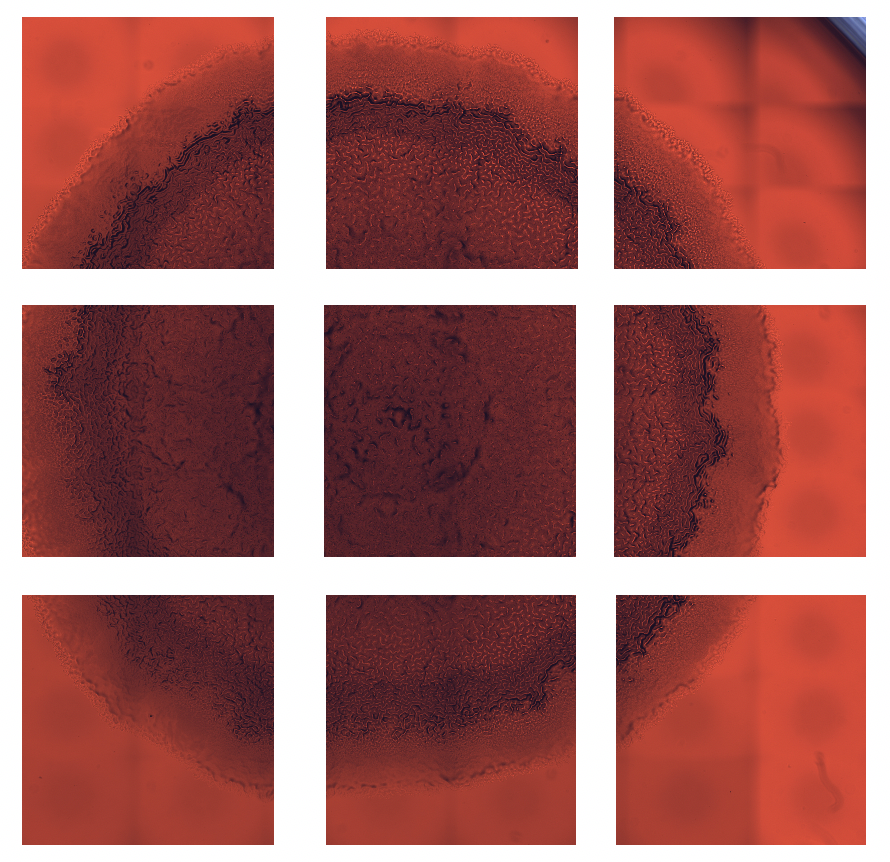
After a convolution layer, a poolinglayer is added in between CNN layers. The function of pooling is to continuously reduce the dimensionality to reduce the number of parameters and computation in the network. This shortens the training time and controls [overfitting](https://en.wikipedia.org/wiki/Overfitting). After the convolution and pooling layers, the classification part consists of a few fully connected layers. These fully connected layers can only accept 1 Dimensional data. To convert 3D data to 1D, the flatten function is used in Python. This essentially arranges our 3D volume into a 1D vector. The last layers of a Convolutional NN are fully connected layers. Neurons in a fully connected layer have full connections to all the activations in the previous layer. 

# Data

The data is the results from experiments conducted by microbiology students from Georgia Institute of Technology. The dataset contains 445 images and phenotypic information regarding the strain. The growth of the bacteria is captured on a bio film after 3 days of exposure to drugs, tobramycin and carbenicillin. In order to determine if a bacteria strain is drug resistant or not, the growth of the cluster exposed to drug is compared with a control growth of the same strain. The two cultures are allowed to grow for 3 days before they are compared. The observed phenotypic data is measured, and images captured. A bacterial strain is said to be resistant to drug if there is no difference (negative or zero lag) in growths observed, while a large difference implies a non-resistant strain. Continuous feature of lag in growth is converted to a categorical feature for binary classification. The bacteria grow in concentric circle patterns, this is a prominent feature of the images. The dataset consists of images of varying resolutions, from as large as 10,000 x 10,000 to 4,000 x 4,000 pixels. This produces an issue, as CNN models require input to be of fixed size. Manual cropping, resizing and other preprocessing strategies help overcome the varied sizes of the images. Due to use of dyes to color the growth cluster red, the images are of dark red hues, with many images that do not exhibit distinct features. The bacteria grow in concentric outward patterns. The lower number of images, the discoloration in images, and the creation of a continuous target feature, introduce factors that inherently introduce loss in training.

Sample input image Cropped to center the feature and resized



Sample input divided into tiles

# 

Sample input divided into tiles and rotated, to introduce feature along the same axis

# Preprocessing

The preprocessing in a ConvNet is less when compared to other classification algorithms. There are few preprocessing tasks performed before the data was fed to the model for classification.

Converting continuous to categorical feature: The continuous feature delta lag, which measures the difference in growth of a strain between control culture and the culture exposed to the drug, was selected as the target feature. The delta lag value implies that a strain is drug resistant if the value is high, and non-resistant if the value is negative or zero. The conversion strategy employed to convert was determined in consultation with Jennifer Rattray, one of the owners of the dataset. The values were mapped to categorial resistant and non-resistant bins. The degree to which, a lag in growth implies resistance of a strain to a drug is debatable. This feature is converted for the two drugs, tobramycin and carbenicillin.

Cropping: Cropping was performed to ensure the images are square shaped, or to center the feature. It is done manually for each image in the dataset so as to not lose information in the image. The images are resized manually to 4000 x 4000 pixels before being read into the convnet to a 500 x 500 input image size.

Data augmentation: Rotation of the image is a type of transformation. This is done to expose the convnet to a wide variety of variations. It makes it less likely that the neural network recognizes unwanted characteristics in the dataset. Breaking the image into grids or tiles is another transformation strategy, this increases the number of images, however does not maintain feature integrity as the bacterial growth is in concentric circles.

Data Normalization: The images once read into the model, they are normalized to between 0 and 1. Scaling input training vectors, ensures that the ranges of distributions of feature values will be different for each feature, and thus the learning rate would cause corrections in each dimension that would vary from one another.

One-hot Encoding: In one-hot encoding, the categorical data is converted into a vector of numbers. Machine learning algorithms cannot work with categorical data directly, hence it required to convert the categorical data in one hot encoding. In one hot encoding one Boolean column is generated for each category or class. One of the two columns take the value 1 for each data point.

Partitioning Training Dataset: For the model to generalize well, training data is split into two parts, one for training and another one for validation. This will help to reduce overfitting as the model would not have seen the validating data in the training phase, which will help boosting the test performance. The entire dataset is divided into 75% train and 25% test, and then the train dataset further into 75% train and 25% validate datasets.

# CNN Architecture

The cnn model should be large enough to capture relations in the data (textures and shapes in images) along with specifics like categories. Initial layers of the model capture high level relations between the different parts of the input like edges and patterns. Later layers capture information that aids in the final decision. If the complexity of the problem is high like Image Classification, the number of parameters and the amount of data required are also very large.

Image classification datasets should be typically very large. Data augmentation was used to generate more data points in order to improve generalization properties. Random cropping of rescaled images together with random horizontal ﬂipping produced augmented data. The goal of the random rescaling and cropping was to learn the important features of each object at different scales and positions.

Another method employed to perform preprocessing was ImageDataGenerator provided by Keras class, that defines the configuration for image data preparation and augmentation. This includes capabilities such as sample-wise standardization, feature-wise standardization. ZCA whitening, random rotation, shifts, shear and flips, dimension reordering and save augmented images to disk. Rather than performing the operations on the entire image dataset in memory, the API is designed to be iterated by the deep learning model fitting process, creating augmented image data just-in-time. This reduces memory overhead but adds some additional time cost during model training.

The Convolutional neural network consists of four main operations.

Convolution: It is the layer that extracts the features from an image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. Depending on the kernel or filter (feature detector) applied and its stride, the convnet reduces the image into a form that is easy to process further while saving the features that are critical for prediction. The convolved output is called a feature map or activation map. Five layers of convolution, with varied kernel sizes, and number of filters were added to the model. It was fine tuned to capture most of features before the fully connected layer.

Non-Linearity: This operation was performed after every convolutional operation. It introduced non-linearity in the convnet. Rectified Linear Unit, ReLU was used in this convnet. It is a max function of zero and input, an element wise operation (applied per pixel). It replaces all negative pixel values in the feature map by zero. Convolution is a linear operation, element wise matrix multiplication and addition, non-linearity is introduced by a non-linear function like ReLU (as real-world data is non-linear).

Output = max(0,Input)

ReLU

0

Pooling: Spatial Pooling (subsampling or down-sampling) reduces the dimensionality of each feature map while retaining the most important information. To describe an image, an aggregate statistic of the features at various locations can be used. Max pooling was used, where max value of a feature in a particular region of the image gets propagated.

Fully connected Layer: It is the classification layer, a traditional Multi-Layer Perceptron that uses softmax activation function in the output layer. Fully connected implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features to classify the input image into two classes.

There are numerous parameters that determine the architecture of the convnet.

Attributes / parameters that can be modified are input size, number of convolutional layers, filter size, stride, non-linearity function, pooling function. There are certain hyperparameters that can be configured to tune a convnet.

Learning rate: This controls the weight in the optimization algorithm. There is fixed learning rate, gradually decreasing learning rate, momentum-based methods or adaptive learning rates, depending on optimizer such as SGD or RMSProp produces the best results. SGD was employed with a learning rate of 1e-6.

Number of epochs: Number of epochs is the number of times the entire training set pass through the neural network. The number of epochs is increased until there is a small gap between the test error and the training error. It was observed that there is no significant improvement after 10 epochs.

Batch size: Mini-batch is usually preferred in the learning process of convnet. The batch size of 100 was implemented.

Activation function: Activation function introduces non-linearity to the final layer of the model. Softmax activation produced better results.

Number of hidden layers and units: It is good to add more layers until the test error no longer improves. It is however computationally expensive to train the network. Having a small number of layers may lead to underfitting. Five layers of Convolution followed by max pooling was used in the convnet.

Dropout - regularization: Dropout is a preferable regularization technique to avoid overfitting in deep neural networks. The method simply drops out units in neural network according to the desired probability. A low dropout of 0.2 was used so as to propagate more feature information.

Pre-trained model:

Transfer learning works by storing knowledge gained while solving one problem and applying it to a different but related problem. It is useful as training large deep networks is expensive and time consuming. VGG16 is a model with weights pre-trained on ImageNet. It is 16 layers deep and is trained on over 1000 categories.

# Implementation

The dataset was divided into training, validation and testing datasets. The initial input to convolutional layer was a 500 x 500 x 3 image where 500 is the height and width of the image and there are 3 channels. The target feature, the difference in control and drug growth, was converted to categorical for each dataset. The model was trained on the training dataset and validated with the validation data before testing the model. The data trained on two models, a simple ConvNet model and a pretrained model to classify. The simple cnn model couldn’t be improved beyond a certain test accuracy level. The Keras pretrained model did not perform well, yielding very low accuracy.

Simple CNN1:

The model had the below configuration. The images were converted from image matrix to an array, rescaled to between 0 and 1, and fed as input to the network. Results for resized, cropped and dataset divided into tiles showed that resized and cropped dataset produced the highest test accuracy. This dataset was used in further iterations. Test accuracy didn’t improve with modification/iteration to more than 87% and 93% for drugs tobramycin and carbenicillin respectively.

The first layer had 32-3 x 3 filters,

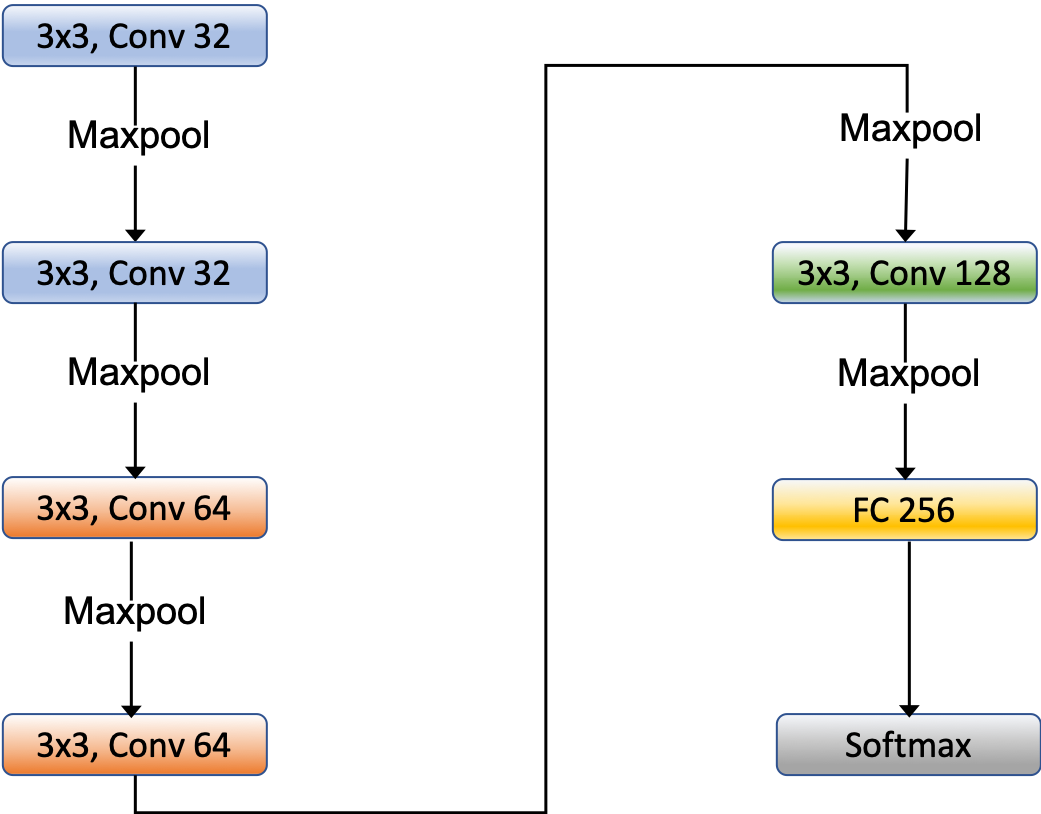
The second layer had 32-3 x 3 filters

The third layer had 64-3 x 3 filters

The fourth layer had 64-3 x 3 filters

The fifth layer had 128-3 x 3 filters.

There were three max-pooling layers each of size 2 x 2.

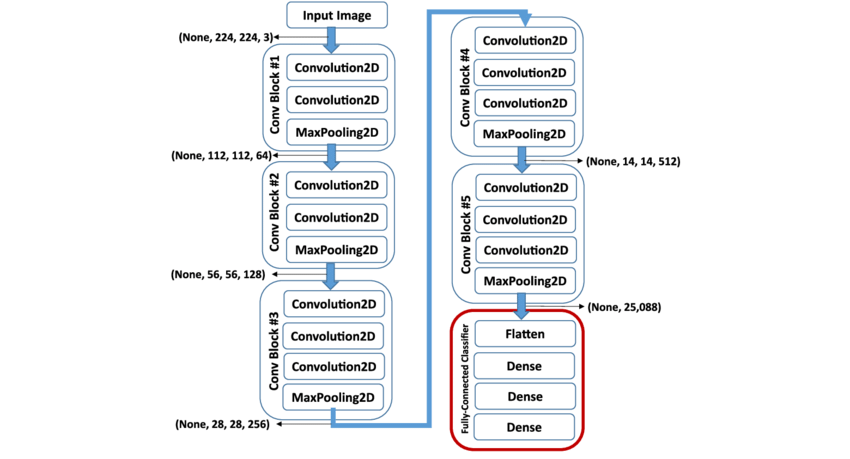


Simple CNN2:

This model had the same configuration as above. The images however were fed to the network through the Keras ImageDataGenerator function. Test accuracy didn’t improve with modification/iteration to more than 80% and 92% for drugs tobramycin and carbenicillin respectively.

Pre-trained Model:

The input was read as 224 x 224 x 3 and fed into the model. The model architecture is as below. The pretrained model VGG16 from Keras produced test accuracy of 23% and 40% for drugs tobramycin and carbenicillin respectively.



# 

# Results

The below table lists the test accuracies of the convnet constructed, with resized and cropped images used as input. It is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage.

Simple CNN1

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | |
| **Input Format** | Carbenicillin | Tobramycin |
| Resized | 87% | 76% |
| Resized and Cropped | 87% | 93% |
| Resized and Cropped Divided into tiles | 74% | 74% |

The cropped and resized images produced highest accuracy, the model was able to extract features and perform better on this input configuration.

Simple CNN2

|  |  |
| --- | --- |
| **Accuracy** | |
| Carbenicillin | Tobramycin |
| 80% | 93% |

The classifier benefitted from the use of ImageDataGenerator, which produced augmented images, exposing the model to more data points.

Pre-trained Model

|  |  |
| --- | --- |
| **Accuracy** | |
| Carbenicillin | Tobramycin |
| 23% | 40% |

This implies the classifier did not benefit much from transfer learning. The pretrained model has to be further fine-tuned to extract better results.

# 

# Conclusion

The low accuracies can be attributed to the small dataset and the imbalanced nature of the data. The target data is such that it creates 2 imbalanced datasets. The effect of class imbalance on classifier performance is undesirable. The impact of imbalance depends on the distribution of data points among the 2 classes. A larger input image dataset could provide a better perspective to the two categories. Adding more data points for the minor class ‘Resistant’ could help the model. Since the creation of new data is dependent on experimentation, data augmentation is a useful technique to explore as Deep learning models work best on large datasets. The irregular growth patterns do not have definitive features that can be learned easily, which in turn leads to low accuracy. Depending on the split of input images to training and test datasets, the imbalance in classes becomes more pronounced. This leads the model to memorize the data instead of truly learning. The model can be further improved by adding cost sensitivity. This will penalize the model as it makes mistakes during learning, reduce over fitting and improve the classifier. Using precision and recall as a measure of the classifier would perhaps be a more reliable measure instead of accuracy. Classification accuracy is typically not enough information to make classification decision. Precision is the number of True Positives divided by the sum of True Positives and False Positives. It is a measure of a classifier exactness. A low precision can indicate a large number of False Positives.

# Reference

Deep Learning

Ian Goodfellow and Yoshua Bengio and Aaron Courville

http://www.deeplearningbook.org; 2016

<https://keras.io/applications/#vgg16/>

Very Deep Convolutional Networks for large-scale image recognition

arXiv.1409.1556v6

https://keras.io/preprocessing/image/

A systematic study of the class imbalance problem in convolutional neural networks

Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski

arXiv:1710.05381

Learning from Imbalanced Data

 IEEE Transactions on Knowledge and Data Engineering ( Volume: 21 , [Issue: 9](https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=5173046) , Sept. 2009)

Haibo He; Edwardo A. Garcia