Introduction to machine learning

Final Project: Bacteria Segmentation

BUCAMP Maia
Kyung Hee University 2024
Student Number : 2024004016

June 18th 2024

Abstract

Bacteria are found everywhere, there are several fields where bacteria are studied, from healthcare to biology and agriculture labs. In this study, we present an approach to bacteria detection and segmentation. Our objective is to develop a robust system capable of segmenting bacteria within photomicrographs in different organisms (body, liquids, and plants) with limited computational resources.

We began by selecting a convolutional neural network (CNN) architecture from PyTorch, as we do not have a computer with a lot of power. We adapted this model for our task, utilizing a dataset of medical images containing bacteria photomicrographs. The fine-tuning process involves modifying the final classification layer for bacterial detection and integrating segmentation techniques to accurately delineate bacterial regions within images.

The performance of our approach is evaluated using standard metrics for segmentation quality. Experimental results demonstrate the effectiveness of the used methods in accurately detecting and segmenting bacteria within photomicrographs. Moreover, the transfer learning approach significantly reduces the computational resources required for model training, making it suitable for devices such as personal computers.

The implications of this work extend to various applications beyond healthcare, with a vast uses in different domains such as: environmental monitoring, food safety, and agricultural research. By accurately detecting and segmenting bacteria within photomicrographs, our developed system offers versatile utility across diverse domains. It can facilitate early detection of bacterial contamination in environmental samples, ensure food safety by identifying harmful bacteria in food products, and contribute to agricultural research by analyzing microbial populations in soil and plant samples.

In conclusion, our study highlights the versatility of transfer learning with pre-trained models for addressing complex challenges in bacterial detection across different domains. Future research directions may involve further optimization of the model architecture and exploration of additional datasets to enhance the system's performance and applicability in various contexts.

Contents

List of Figures

List of Tables

1 Introduction

Microorganisms are found in various environments and play essential roles in ecosystems and in the health care industry. The ability to accurately detect and segment bacteria in images is essential in several fields of study, research and work, including medical diagnosis, environmental monitoring, and food safety. In this study, we present an approach to bacterial detection and segmentation using the methods we have learnt in class.

The task of bacterial detection and segmentation involves identifying bacteria within digital images acquired through microscope or imaging devices. Traditional methods for bacterial detection and segmentation often rely on manual annotation or simplistic image processing techniques, which can be time-consuming and error-prone, especially for a large number of data.

To address these challenges, we focused on the U-Net architecture, a convolutional neural network (CNN), since it was developed for biomedical image segmentation tasks.

Our objective is to develop a robust system with limited computational resources. The model should be capable of detecting and segmenting bacteria within photomicrographs in a diversity of ecosystems. To achieve this we used a dataset of medical images containing bacteria under a microscope, as well 8 other types of bacteria under different conditions to train and evaluate our model.

Overall, our study demonstrates that by making some adjustments and using advanced techniques, we can enhance a computer's ability to detect bacteria in images. This improvement can be extremely beneficial, by accurately detecting and segmenting bacteria within images, our system can be useful across various fields, including healthcare, environmental monitoring, food safety, and agriculture. It can facilitate early detection of bacterial contamination, ensure food safety by identifying harmful bacteria, and could contribute to scientific research by analyzing microbial populations in diverse samples.

In this paper, we will first discuss the methods used for the Neural Network architecture and the concepts needed to fully understand a U-net. Next, we will cover the preprocessing of the datasets, which was an essential step to achieve optimal results. This process was the longest. We will then address the results of the training, followed by testing the trained model, including a confusion matrix. Finally, we conclude with remarks on future research directions.

2 Methods

There is a large consent that in order to have a successful Neural Network, we require thousands of annotated training samples. In this project we present a network and training strategy that relies in the strong use of data augmentation.

Due to the limitation in computational resources, we have used a pre-trained Neural Network from Pytorch. We have adapted this model for our task, utilizing a dataset of bacteria photomicrographs. The fine-tuning process involves modifying the final classification layer for bacterial detection and integrating segmentation techniques to accurately color bacterial pixels within the images.

The architecture consists of a contracting path to capture context and symmetric expanding path that enables precise localization. So that such a network can be trained end-to-end from very few images. Moreover, the network is fast, the segmentation of a 512x512 image takes less than a second on a recent GPU, according to the university of Freiburg.

2.1 U-Net

U-net was originally invented and first used for biomedical image segmentation. Its architecture can be thought of as an encoder network followed by a decoder network. Unlike classification where the end result of the the deep network is the only important thing, semantic segmentation not only requires discrimination at pixel level but also a mechanism to project the discriminative features learnt at different stages of the encoder onto the pixel space.

Therefore we have decided to use a U-Net not only because of it's perfect use for medical images and segmentation but also because of its widespread in its image segmentation architecture. The widespread is measured on its flexibility, optimization of modular design and success in all medical image modalities. Therefore being a perfect fit for photomicrographs, which are often found in the medical research area.

The architecture of the used U-net is the following:

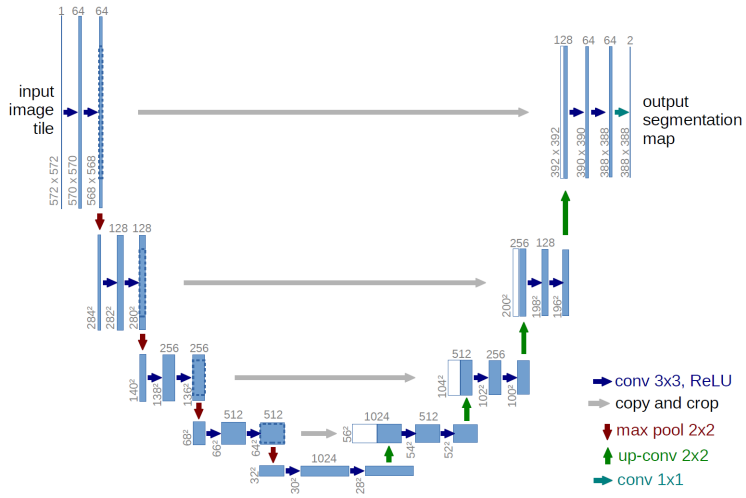


Figure 1: U-net architecture (example for 32x32 pixels in the lowest resolution)- Wikipedia

Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a **rectified linear unit** (ReLU) and a 2x2 **max pooling** operation with stride 2 for down sampling. At each down sampling step we double the number of feature channels. Every step in the expansive path consists of an up-sampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

2.1.1 ReLu

The rectified linear unit (ReLU) or rectifier activation function introduces the property of non-linearity to a machine learning model and solves the vanishing gradients issue.

The diagram on the next page has both visual representations of the Rectified Linear Unit (ReLU), in blue, whereas the green line is a variant of ReLU called Softplus. The other variants of ReLU include leaky ReLU, exponential linear unit (ELU) and Sigmoid linear unit (SiLU), etc., which are used to improve performances in some tasks.

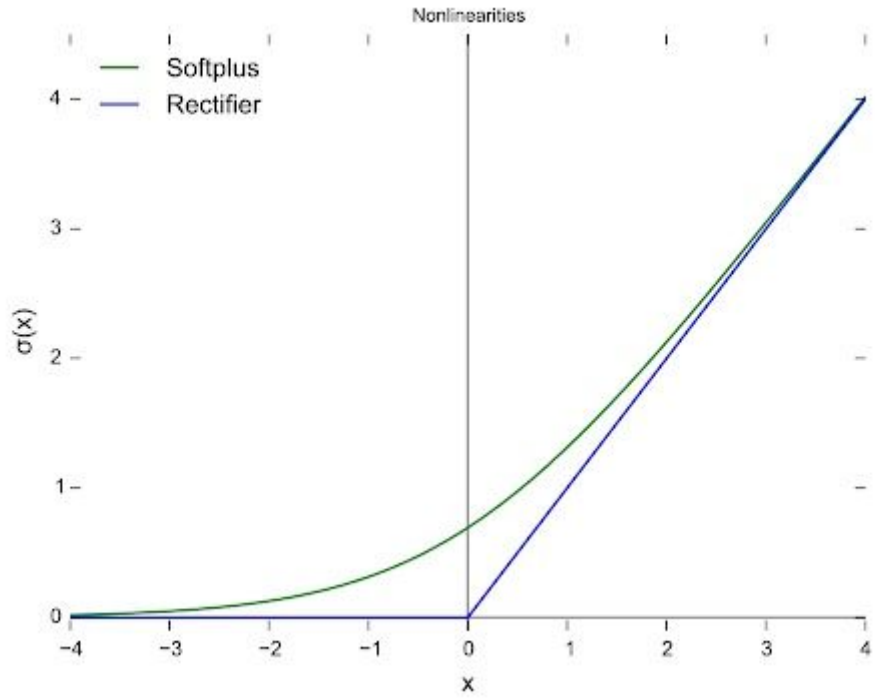


Figure 2: Example of a ReLU activation function (blue) and Softplus (green)- Wikipedia

As we have said previously, ReLU is used to solve the vanishing gradient issue. The derivative of the ReLU function helps the model avoid tending to zero, which is crucial for maintaining strong gradients during backpropagation. This ensures that the weights in the network are updated effectively, allowing the model to learn more efficiently. By preventing gradients from becoming too small, ReLU helps the neural network to converge faster and achieve better performance.

However there are some disadvantages, one of the main issues with ReLU is that all the negative values become zero immediately, which decreases the ability of the model to fit or train from the data properly.

This means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turn affects the resulting graph by not mapping the negative values appropriately. This can however be easily fixed by using the different variants of the ReLU activation function.

2.1.2 Max pooling

In the U-net that we decided to use, we use max pooling. Max pooling is a pooling operation commonly used in convolutional neural networks to reduce the spatial dimensions (width and height) of feature maps, thereby decreasing the computational load and the number of parameters in the network.

During max pooling, a window (typically 2x2 or 3x3) slides over the input feature map. At each position, the maximum value within the window is selected and retained, while the other values are discarded. This process effectively down-samples the feature map, preserving the most important features and reducing spatial resolution. We can observe the process on the image below:

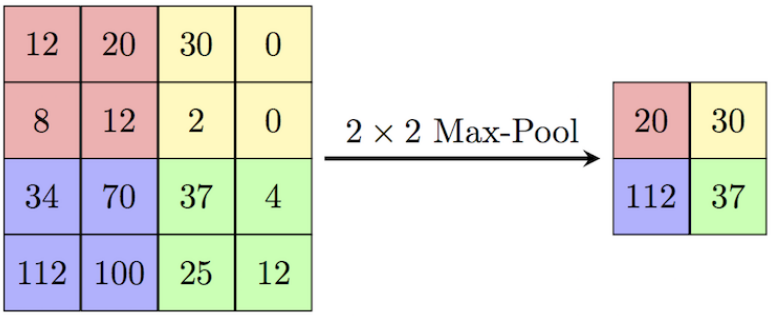


Figure 3: *Maxpooling 4x4 to 2x2 sample -Computersciencewiki.org*

There are four main reasons why we use max pooling. In the first place max pooling helps with **dimension reduction**. High-resolution images, such as digital-microscope images, are computationally expensive. Max pooling **reduces the image size**, making it easier for the neural network, which speeds up training and requires less memory. Secondly, it is **invariant to small changes** , it ensures that even if an object in an image is slightly shifted or rotated, the network can still recognize it. Which in this case is essential, as bacteria are not always in the same position and exactly the same shape, it is necessary for the network to be able to recognise bacteria in all the different ways possible. In a third place, by selecting the maximum values in each window, max pooling retains the most essential features while discarding less relevant information, this enhances **feature selection**. And lastly, max pooling helps the network see the bigger picture, summarizing information in larger regions enables the network to capture more significant spatial patterns, in other words it increases the receptive field.

Overall, max pooling is a critical component in our Neural Network, aiding in effective feature extraction and dimension reduction, which contribute to the model's ability to generalize well to unseen data.

2.2 Dice loss

We will use dice loss as the loss function as it is often utilized for image segmentation tasks, particularly in the domain of medical imaging. Unlike traditional cross-entropy, dice loss is better suited for datasets where the objects of interest (foreground) are much smaller than the background, which is the case of bacterial images. This loss function calculates the dice coefficient, which measures how well predicted and target masks overlap, ensuring accurate segmentation of fine details. By penalizing mistakes less harshly than cross-entropy, Dice loss improves boundary delineation and overall segmentation quality. Discussing the implementation and benefits of Dice loss in LaTeX subsections helps researchers and practitioners grasp its effectiveness in enhancing segmentation results for intricate medical images.

The dice loss mathematical expresion is the following :

$$\text{Dice Loss} = 1 - \frac{2 \times |X \cap Y|}{|X| + |Y|}$$

Where $|X \cap Y|$ denotes the number of overlapping pixels between X and Y .
$|X|$ and $|Y|$ represent the total number of pixels in X and Y, respectively.

3 Data preprocessing

To obtain the necessary data for our project, we utilized two main data sets. Due to the limited availability of training data for bacterial segmentation, we were unable to find other suitable datasets. However, these datasets meet the requirements for the detection and segmentation of various bacterial types, ensuring that our model can operate effectively and return the expected results.

The dataset, titled "Mixed segmentation dataset and StarDist model", comprises mixed training and test images of S. aureus, E. coli, and B. subtilis specifically for cell segmentation. As for the second data set used, there are annotated images of bdellovibrio bacteria under a phase contrast microscope. Annotations were made by hand by Pressé Lab at Arizona State University, the original paper, "Two-state swimming: Strategy and survival of a model bacterial predator in response to environmental cues" by Lance W.Q., J. Shepard Bryan IV, Zeliha Kilic, and Steve Pressé.

In a first place it is important to underline that the model has some specificities regarding the image format. Preprocessing was needed in order to have the images to have 3 channels and not 1 (which is common in bacterial images) and also to adjust the size. Therefore supplementary code was added. The author also changed the .tif images into .png as the model had difficulties with .tif images even though, they are said to be supported. As previously said, there was also a need to modify annotations in order for them to be in the correct format.

To create the training dataset for bacteria segmentation, the individual channels of the images were separated and processed using Fiji software. For the Nile Red fluorescence images, the shapes of S. aureus cells were manually outlined using ellipsoid selections to approximate their shapes. These outlines were then used to generate ROI (Region of Interest) maps, where each cell was marked with a unique number. The training images, originally 512 x 512 pixels, were divided into smaller images of 256 x 256 pixels.

For segmenting the S. aureus bright field images, the corresponding Nile Red fluorescence images were utilized along with the ROI masks created from the fluorescence images.

For processed images, pixel values are assigned as follows: 0 for the background, 1 for the cell body, and 2 for the cell boundary, as illustrated in Figure 4. For the first data set.

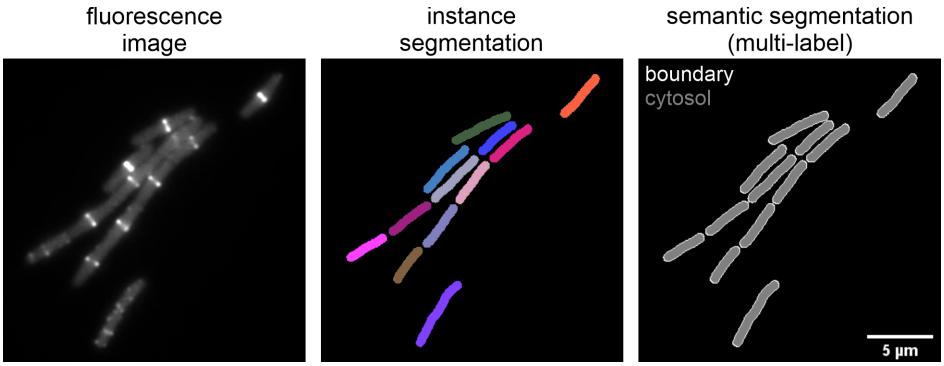


Figure 4: Segmentation B.subtilis -DeepBacs – Mixed segmentation dataset

We did not use the multi-label feature, as this data set was too small (11 images) and we wished to combine different data sets to have a variety of bacteria, this feature was not useful.

The second data set was annotated differently, therefore we prepossessed only 102 images due to computational limitations.

The primary goal was to convert the annotations of these RGB images into a format suitable for further analysis, particularly focusing on distinguishing bacteria and non-bacteria regions based on the provided annotations.

The RGB images in the selected subset were already annotated to highlight regions of interest, which were crucial for the analysis of an image. Each pixel in the RGB images was evaluated to determine its classification into two categories: bacteria and non-bacteria regions. The preprocessing task involved loading each RGB image and creating a corresponding annotation image of the same dimensions. Based on the specific

dataset's annotation conventions, a methodology was implemented to classify pixels. In this case, the red channel intensity was utilized as a criterion to distinguish bacteria from non-bacteria regions. We used python programming language, along with libraries such as OpenCV and NumPy, which facilitated efficient image processing operations, image processing was very efficient with only 1.68 seconds per file (102 images). After preprocessing, annotation images were saved with filenames corresponding to their respective RGB images. This facilitated easy linkage between original annotations and processed annotations.

```
import cv2
import numpy as np
ann_img = np.zeros((30,30,3)).astype('uint8')
ann_img[3, 4] = 1  # Set the label of pixel 3,4 as 1
cv2.imwrite("ann_1.png", ann_img)
```

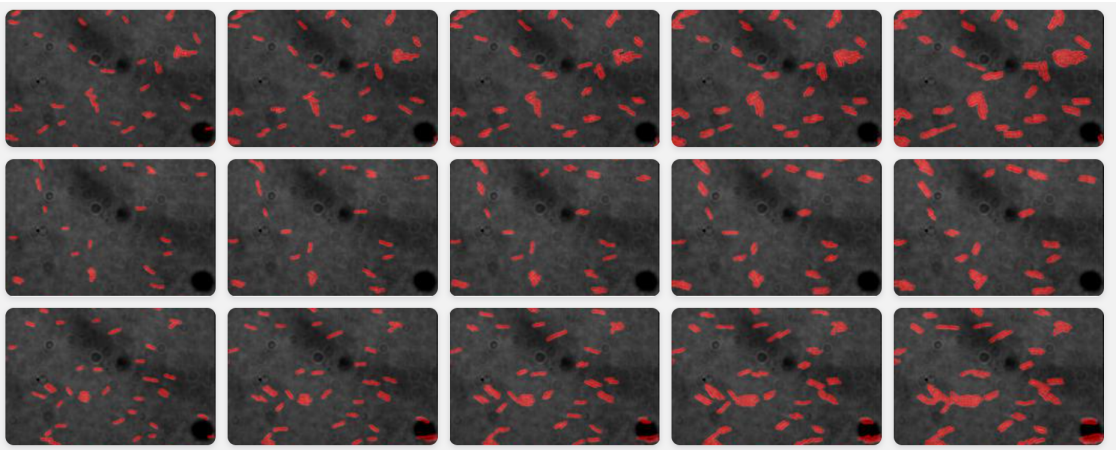


Figure 5: Image of annotation images - oval and round bacteria - pixel by pixel process

By focusing on a subset of 102 annotated images and applying systematic preprocessing steps, the dataset was prepared for further analysis. This approach not only optimized computational efficiency but also ensured that the annotations were correctly formatted for subsequent tasks, such as training machine learning models or conducting statistical analyses.

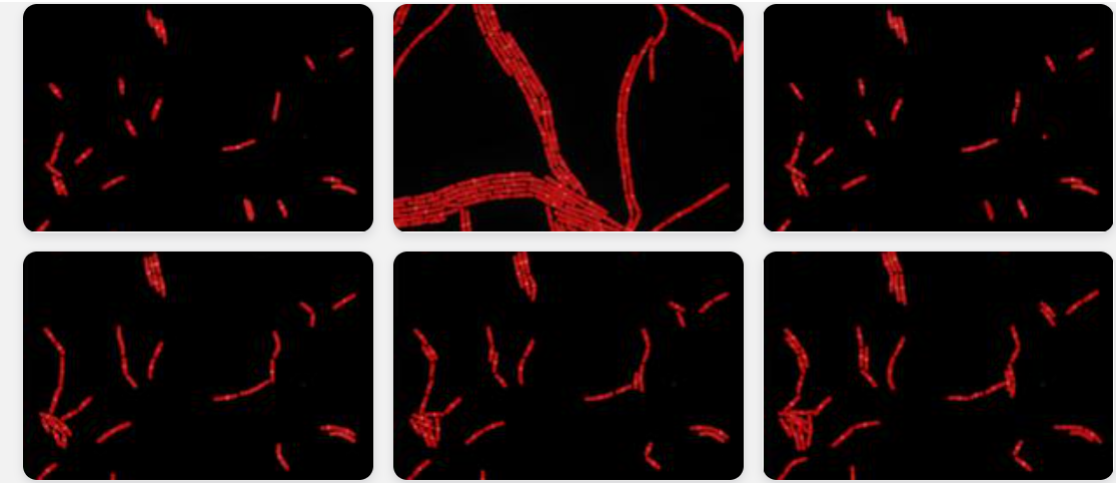


Figure 6: Image of annotation images - strand bacteria - pixel by pixel process

Both data sets included test data, inverse crime was not made.

4 Training

The original model was trained from scratch with 5k images and scored a Dice coefficient of 0.988423 on over 100k test images.

The model was said to be easy to use for multi-class segmentation, portrait segmentation, medical segmentation. However the author did encounter several issues. Such as image format, which we have previously discussed.

Training would have taken some time in the authors personal computer. Therefore the author decided on doing it at a PC-café in order to have enough power, training time on a personal computer is then biased. The training took a total of 10 minutes and 48 sec, this was efficient since we used a pretrained model.

As said before we have used dice loss to measure the loss results during training. The horizontal axis represents the training epochs, we did a total of 1000 epochs. An epoch refers to one complete pass through the entire training dataset. The vertical axis represents the dice loss value. Lower values of dice loss indicate better performance, meaning the predicted segmentation is closer to the ground truth segmentation.

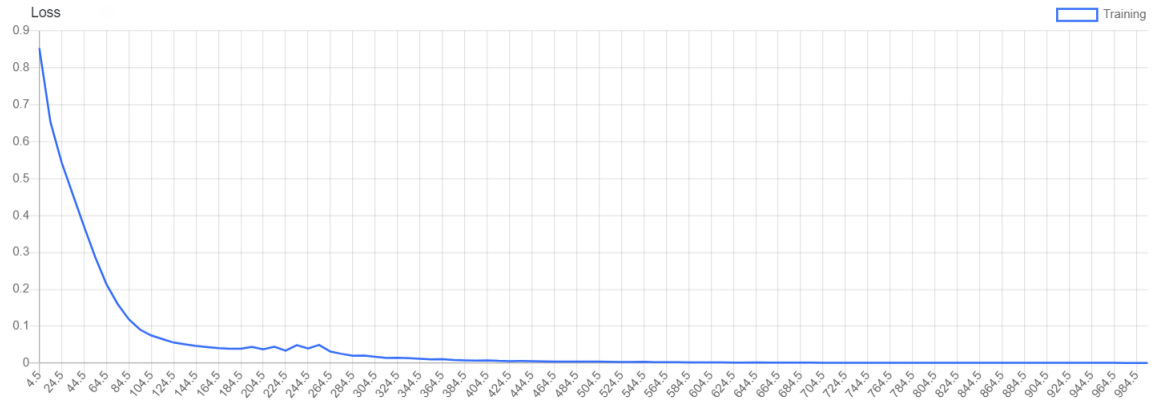


Figure 7: Loss curve - x-axis represents the number of epochs and y axis represents the dice loss values

As the curve indicates, the model reaches optimal performance after 400 epochs and maintains more or less that level thereafter. Consequently, we decided to proceed with manual testing, as additional epochs would not yield further improvements.

We were also able to represent the accuracy results in function of the number of epochs for training as well as for the validation. The results were the following:

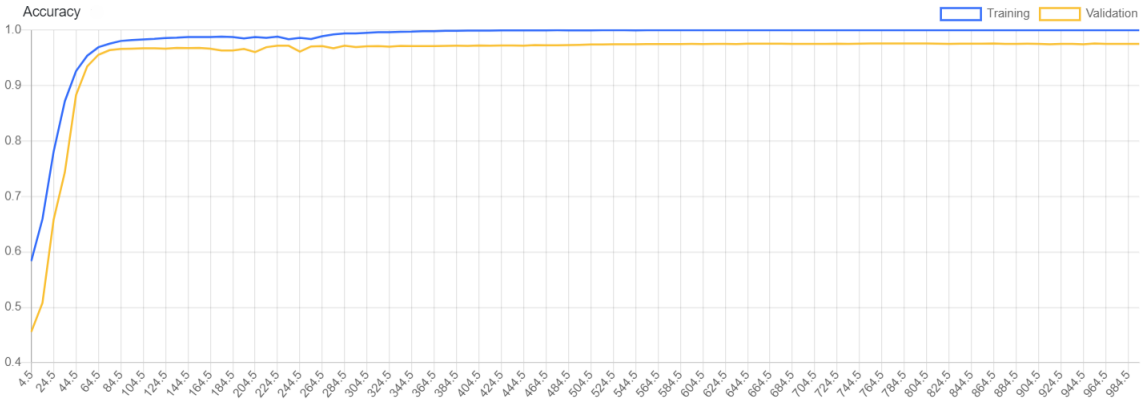


Figure 8: Accuracy results for training and validation

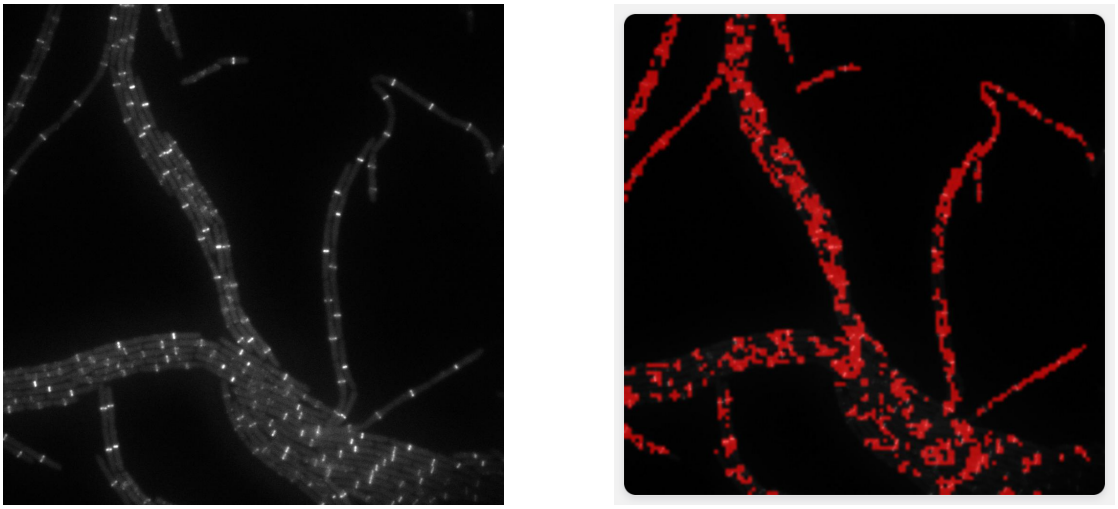
We observed that the validation data showed significantly lower accuracy compared to the training data. However, since further epochs did not improve results, we were satisfied with the current performance and proceeded to conduct manual testing with various methods.

5 Results

After training the accuracy given was 97.53% using validation data. Therefore after having a satisfying result we proceeded to inspect the results by testing it. In average we say a model is well trained above a 98% accuracy, however we could never obtain a better result, even using different methods and enhancing image contrast.

5.1 Testing

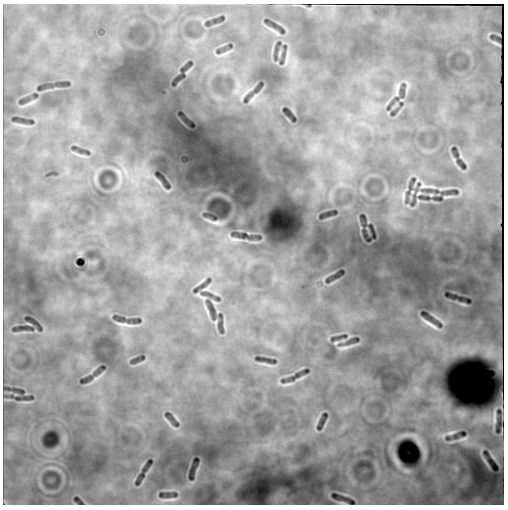
During the testing phase of the neural network, the training images demonstrated high accuracy for images of the same bacteria as the data set it was trained on. Although the segmentation was not perfect, the model successfully detected both spiral and oval bacteria, as the training data sets contained mainly these two types of bacteria.



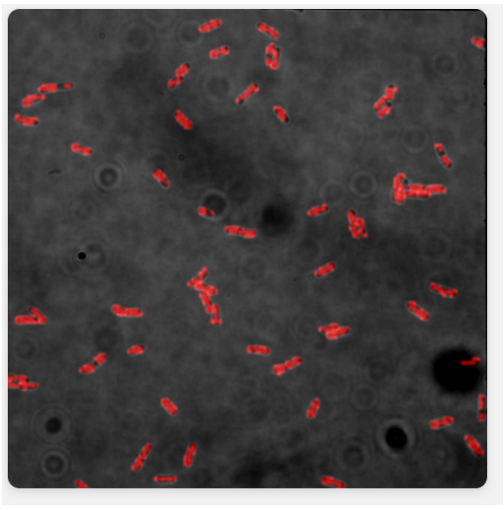
(a) *e.coli image - Pressé Lab*

(b) *e.coli image after segmentation*

Figure 9: Comparison of e.coli image and its segmented image



(a) *B.subtilis image - Pressé Lab*



(b) *B.subtilis image after segmentation*

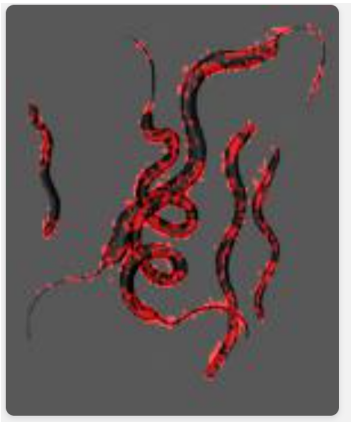
Figure 10: Comparison of B.subtilis image and its segmented image

5.1.1 Manual Test

The model was further evaluated using a Microorganism dataset from Rushi Lunagariya, which includes photos of Amoeba, Euglena, Hydra, Paramecium, Rod Bacteria, Spherical Bacteria, Spiral Bacteria, and Yeast.

During manual validation we observed two main outcomes. First, the color variations did not affect the differentiation of the bacteria due to the high contrast, as clearly shown by the spiral bacteria:

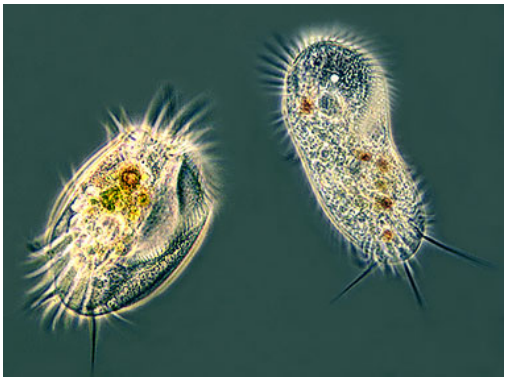
(a) *Original Image-104 - Rushi Lunagariya*



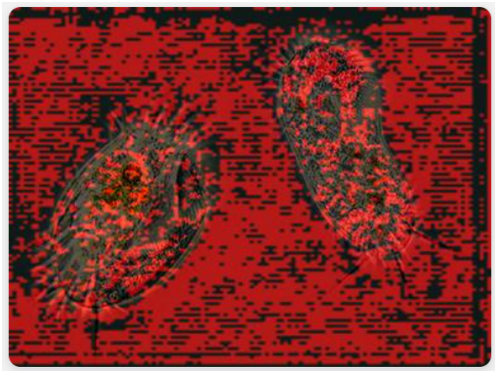
(b) *Image-104 Spiral after Segmentation*

Figure 11: Comparison of Image 104 Spiral Bacteria and Segmented Image

Second, some images subjected to supervised testing were not accurately segmented. For instance, Image-27-Amoeba demonstrates the lack of differentiation between bacteria and the background:



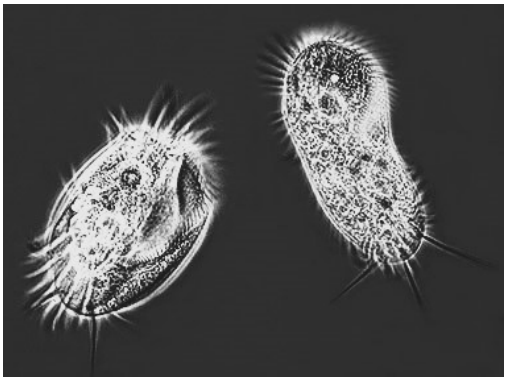
(a) *Original Image-27-Amoeba - Rushi Lunagariya*



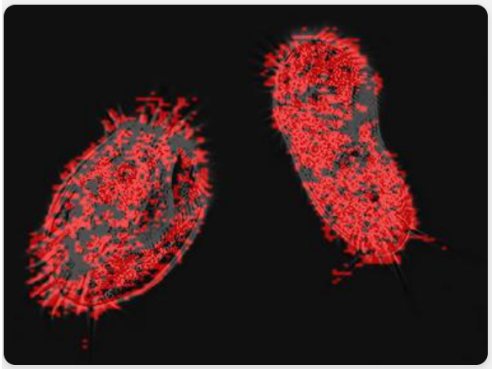
(b) *Image-27 after segmentation*

Figure 12: Comparison of Image 27 Amoeba and its segmented image

However, when applying a grayscale and contrast adjustment, the bacteria were satisfactorily segmented, showing a clear distinction from the background.



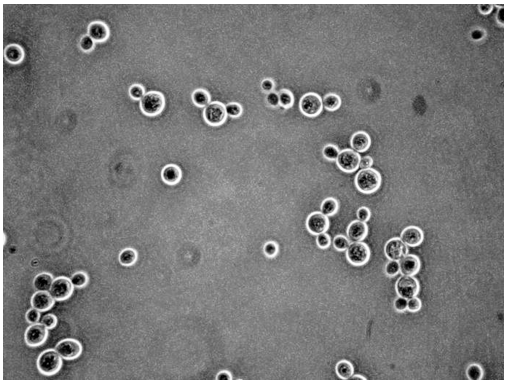
(a) *GrayScale and Contrast on Image-27-Amoeba - Rushi Lunagariya*



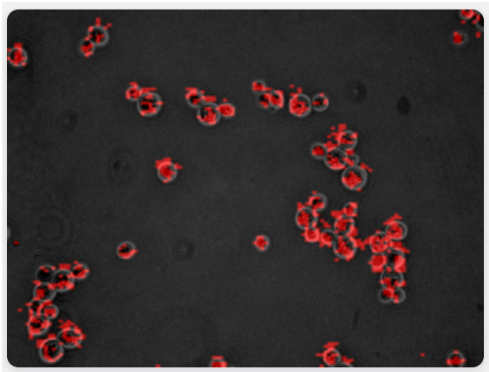
(b) *Image-27 with GrayScale after segmentation*

Figure 13: Comparison of Image 27- Amoeba GrayScale with contrast and its segmented image

Other images that yielded good results included yeast bacteria and hydra bacteria. The yeast bacteria segmentation was expect to be successful as it has a round shape, but it is important not to neglect the success of the model in differentiating bacteria and stains or dirt.



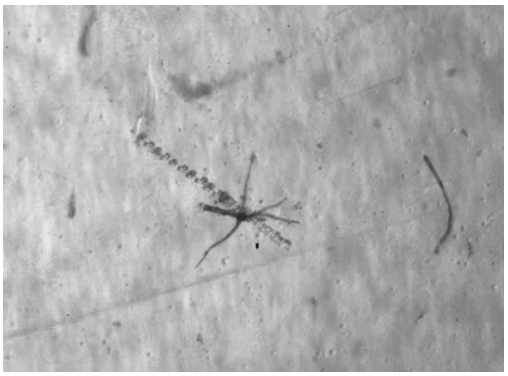
(a) *Image-41 Yeast Original Image - Rushi Lunagariya*



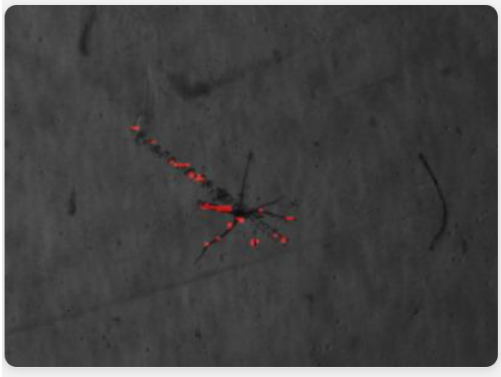
(b) *Image-41 Yeast after segmentation*

Figure 14: Comparison of Image-41 Yeasts with its segmented image

We are particularly pleased with the results of hydra image 35 segmentation, which exceeded our expectations. Although the original image appeared to be in black and white, it did not pass the grayscale filter, indicating that the image was not truly grayscale. The bacteria, with its unique shape, was not round, oval, or spiral, and the dirt was not segmented. Despite these challenges, the bacteria was detected with its distinctive shape, though the segmentation was only marginally accurate.



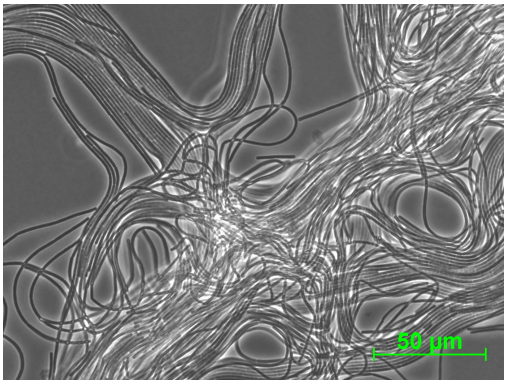
(a) *Image-35-Hydra Original - Rushi Lunagariya*



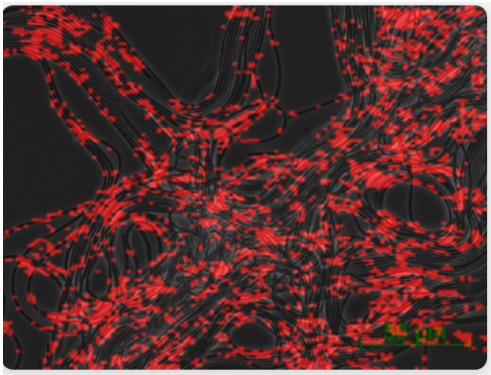
(b) *Image-35-Hydra after segmentation*

Figure 15: Comparison of Image-35 Hydra with its segmented image

Additionally, we tested the model on images sourced from the internet. One of these images, which shows string bacteria, produced a result we are very pleased with. Although the segmentation was not perfect, there was a discernible differentiation between the background and the bacteria which seemed like a hard task to do for our model.

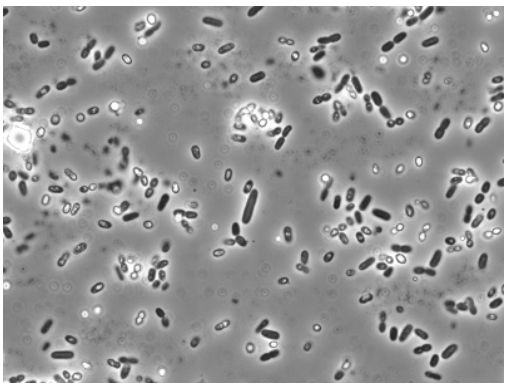


(a) *Internet image - filamentous bacteria*

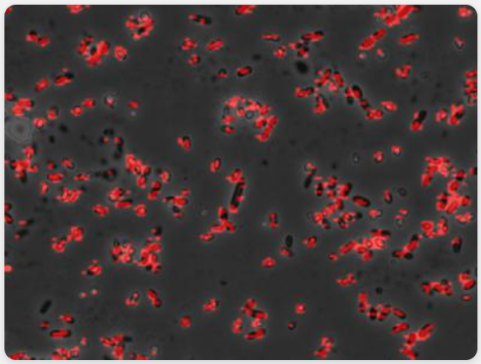


(b) *Internet image after segmentation- filamentous bacteria*

Figure 16: Comparison of filamentous from internet with its segmented image



(a) *Internet image - oval shape bacteria*



(b) *Internet image after segmentation- oval shape bacteria*

Figure 17: Comparison of oval bacteria from internet with its segmented image

5.2 Numerical Results

We were able to obtain the following data for each of the manually tested images, in which "Bacteria pixels" represent the percentage of pixels detected as bacteria and non-bacteria pixels the percentage of pixels that were detected as the background:

1. Image_test_data: Bacteria pixels: 3.23% - Non-bacteria pixels: 96.77%
2. Image_test_data_2: Bacteria pixels: 9.27% - Non-bacteria pixels: 90.73%
3. Yeast_test: Bacteria pixels: 4.98% - Non-bacteria pixels: 95.02%
4. Image_27_amoeba: Bacteria pixels: 56.81% - Non-bacteria pixels: 43.19%

5. Image_27_BW_contrast: Bacteria pixels: 17.68% - Non-bacteria pixels: 82.32%
6. Image-104: Bacteria pixels: 7.33% - Non-bacteria pixels: 92.67%
7. Hydra_35: Bacteria pixels: 0.26% - Non-bacteria pixels: 99.74%
8. test_internet_image_1: Bacteria pixels: 14.83% - Non-bacteria pixels: 85.17%
9. test_internet_image_3: Bacteria pixels: 23.09% - Non-bacteria pixels: 76.91%

These were the results for manual test data, as we do not have the truth values we can not go further with the study with this data. However this information has helped us see that in general,the number of pixels identified as bacterial regions is lower than the number of pixels that are classified as background. These results are coherent and help us evaluate the given values of the validation data set.

The overall accuracy is given as 97.53% by the model used, therefore there was 2.47% of the data that was not correctly classified. There is only two available data, bacteria % of pixels in validation data and % of pixels found as well as the corresponding data for the background. These values are average values of all the validation data, therefore values may not seam coherent.

	Truth values	**Found values**	**Misclassified percentage**
Bacteria pixel %	8.694%	15.28%	6.586%
Background pixel %	90.974%	84.72%	6.254%

Table 1: Confusion matrix - Comparison of Bacteria and Background Pixel Percentages

The percentage of bacterial pixels misclassified across the entire image is 6.586% higher than the actual values. As for non-bacterial pixels, the misclassification rate is 6.254% lower than the actual values. The model shows a different accuracy between training and manual test data, with a difference of 3.73%. This higher difference is primarily due to images like Image-27-Amoeba, which are nearly completely segmented. Excluding this image, the difference would be of 0.32%, indicating the model effectively segments images without strong contrasts. However it is also biased by images such as hydra-35 with a very low segmentation percentage of the bacteria. Overall, the model correctly identifies new images. However, these values seem inconsistent, as visually there are fewer regions classified as bacteria than expected.

6 Discussion

The performance of our model for bacterial detection and segmentation was satisfactory, but not perfect. While the results demonstrate the potential of the U-Net architecture and transfer learning for this task, several limitations must be acknowledged.

Firstly, the limited amount of training data posed a significant challenge. The original U-Net model was trained on a much larger dataset, consisting of 5,000 images, and achieved a dice coefficient of 0.988423 on over 100,000 test images. In contrast, our model was trained on only 113 primary images and tested on 113 additional images. This discrepancy in dataset size likely contributed to the performance gap between our model and the original.

The small dataset size not only limited the diversity of bacterial samples available for training but also constrained the model's ability to generalize to new, unseen data. In medical image segmentation, where variability in bacterial forms and imaging conditions is high, a larger and more varied dataset is essential for a good model performance. Our experimental results indicate that while the model could identify and segment bacteria, its accuracy and consistency were affected by the limited training data.

To improve the model's performance, increasing the number of training images is essential. A more extensive dataset would provide a richer representation of bacterial appearances and conditions, enabling the model to learn more generalized.

Another area for improvement is the preprocessing and image format handling. Our approach required converting TIFF images to JPEG format, which may have introduced some preprocessing artifacts. Ensuring consistent and high-quality input data is crucial for optimizing model performance. We discovered that the efficiency of the neural network in detecting bacteria improves under specific image conditions, such as applying grayscale and contrast adjustments. Therefore, it would be intriguing to apply these conditions to achieve optimal results.

Despite these limitations, our study successfully demonstrated it is possible to use U-Net for bacterial detection and segmentation with limited computational resources. The satisfactory results achieved with a small dataset highlight the potential for further improvements with additional data and optimization.It was surprising to see that Image-35 Hydra (Figure-15) was successfully detected despite variations in bacteria shape and image color. We are highly pleased with this outcome, demonstrating the U-net's capability to accurately segment bacteria across diverse forms.

Future work should focus on expanding the training dataset, exploring advanced data augmentation techniques, and refining preprocessing. Additionally, experimenting with different network architectures and hyper-parameters may yield further performance gains. The integration of these improvements could bring the model's performance closer to the high standards set by the original U-Net.

In summary, while our developed system has shown satisfactory results, there is a clear pathway for enhancements through increased training data and further optimization. This study underscores the importance of data availability and quality in developing robust machine learning models for complex tasks like bacterial detection and segmentation.

7 Conclusion

In this study, we explored the use of a convolutional neural network, specifically the U-Net architecture, for the task of bacterial detection and segmentation within photomicrographs. Our objective was to train an Neural Network capable of performing the wanted tasks with limited computational resources. Despite encountering challenges, particularly in terms of dataset size, our findings demonstrate the potential and versatility of the approach. Our approach not only showcased the effectiveness of machine learning techniques but also highlighted the challenges and opportunities in this domain.

Detecting and accurately segmenting bacteria in images is essential for applications, ranging from medical diagnosis to environmental monitoring and agricultural research. Traditional methods often rely on manual annotation or simplistic image processing, which are time-consuming and prone to errors, especially with large datasets. In contrast, our use of the U-Net architecture enabled us to achieve robust results with relatively limited computational resources.

Our methodology involved preprocessing datasets to ensure compatibility and optimal input format for the neural network. This step was crucial in preparing annotated images for training, where we focused on distinguishing bacterial regions from the background across different imaging modalities. The integration of dice loss as our primary loss function further refined our result analysis.

During training and validation, our model exhibited promising performance, including high accuracy rates in classifying bacterial pixels and distinguishing between different bacterial shapes such as round, spiral, and other variations. Despite encountering challenges such as variability in image quality and bacterial morphology, we remain pleased with the results. However, the performance gap between our model and the original U-Net trained on a much larger dataset underscores the importance of having a large and diverse training dataset in order to achieve optimal results.

Furthermore, manual testing against diverse datasets, including those sourced from medical and environmental studies, demonstrated the versatility and applicability of our approach. While our model achieved impressive segmentation accuracy in controlled environments, challenges remain in scaling up to larger datasets and real-world applications where environmental variability and imaging artifacts may pose additional hurdles. We have identified several improvements for future work that could enhance the model's

performance. Increasing the number of training images, as it will provide a more comprehensive representation of bacterial variations and improve the model's generalizability.

In conclusion, our study highlights the feasibility and potential of using U-Net for bacterial segmentation tasks, even with limited computational resources. While our current results are promising, they also point to the need for more extensive datasets and further optimization to achieve higher accuracy and consistency. This research lays the groundwork for future developments in bacterial detection and segmentation, emphasizing the importance of data quality and quantity in building robust machine learning models.

Through continued research and improvement, we could advance the capabilities of machine learning models in the domain of bacterial detection, ultimately contributing to advancements in various scientific and industrial fields.

8 References

- https://www.kaggle.com/code/anhvle/bacteria-segmentation - https://builtin.com/machine-learning/relu-activation-function: :text=The%20ReLU%20activation%20function%20is%20used%20to%20i
- https://computersciencewiki.org/index.php/File:MaxpoolSample2.png

GitHub :

- https://github.com/amine0110/Liver-Segmentation-Using-Monai-and-PyTorch- for image segmentation methods
- https://github.com/Project-MONAI
- https://github.com/HenriquesLab/DeepBacs/wiki/Segmentation

8.1 Data sets

- Google Data set search
- https://zenodo.org/records/5551009

  Description- Mixed training and test images of S. aureus, E. coli and B. subtilis for cell segmentation using StarDist, as well as the trained StarDist model.
- https://www.kaggle.com/datasets/rushilunagariya/micro-organisms-dataset Description -This is the Micro Organism data set. Which includes photos of: Amoeba, Euglena, Hydra, Paramecium, Rod Bacteria, Spherical Bacteria, Spiral Bacteria, Yeast. This is an Augmented Data set.

9 Appendices

9.1 Software & Packages

- Python
- PyTorch
- CUDA
- OpenCV
- NumPy
- 3D slicer
- ITK snap

10 Code

```
import cv2
    import numpy as np
    import os

    def label_image(input_path, output_path):
                # Read the input JPEG image
                img = cv2.imread(input_path)
                # Convert the image to grayscale for easier thresholding
                labeled_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                # Save the labeled image as a PNG file
                cv2.imwrite(output_path, labeled_img)

    def process_images(input_dir, output_dir):
                # Create the output directory if it doesn't exist
                if not os.path.exists(output_dir):
                        os.makedirs(output_dir)
                # Iterate through each file in the input directory
                for filename in os.listdir(input_dir):
                        if filename.lower().endswith('.jpeg'):
                            input_path = os.path.join(input_dir, filename)
                            output_path = os.path.join(output_dir, filename.replace('.jpeg',
'.png'))
                        # Process the image
                        label_image(input_path, output_path)
                print ('all done ')
                return
```