# PURE FALL 2022-2023 FINAL REPORT

U-NET

06 JANUARY 2023

OKAN ARIF GUVENKAYA

MAJOR: MECHATRONICS ENGİNEERİNG, SENIOR
DOUBLE MAJOR: COMPUTER SCIENCE AND ENGİNEERİNG, SENIOR

okanarif@sabanciuniv.edu
26780

# Table of Contents

# INTRODUCTION

This report includes details of the project called U-Net under the Program for Undergraduate Research (PURE) program given at Sabancı University in the Fall 2022-2023 education term. U-Net is a generic deep-learning solution for frequently occurring quantification tasks such as cell detection and shape measurements in biomedical image data (Falk et.al , 2019). This project aims to develop a generic deep-learning algorithm for the detection of different types of bacteria using the fluorescent and phase images of cells acquired by fluorescent microscopy. The report includes 5 main titles: Introduction, Procedure and Results, Conclusion, References, and Appendix.

# PROCEDURE AND RESULTS

## 1- Image Processing and Segmentation

In the beginning, the images series of the bacteria were taken via microscopy to observe their growth. The original image was too big. For computational efficiency, a specific part of the image was determined, and that area was taken as a reference. All other images were taken from the same cropped version of the same area from sequential big images. The shape of the images as follows:

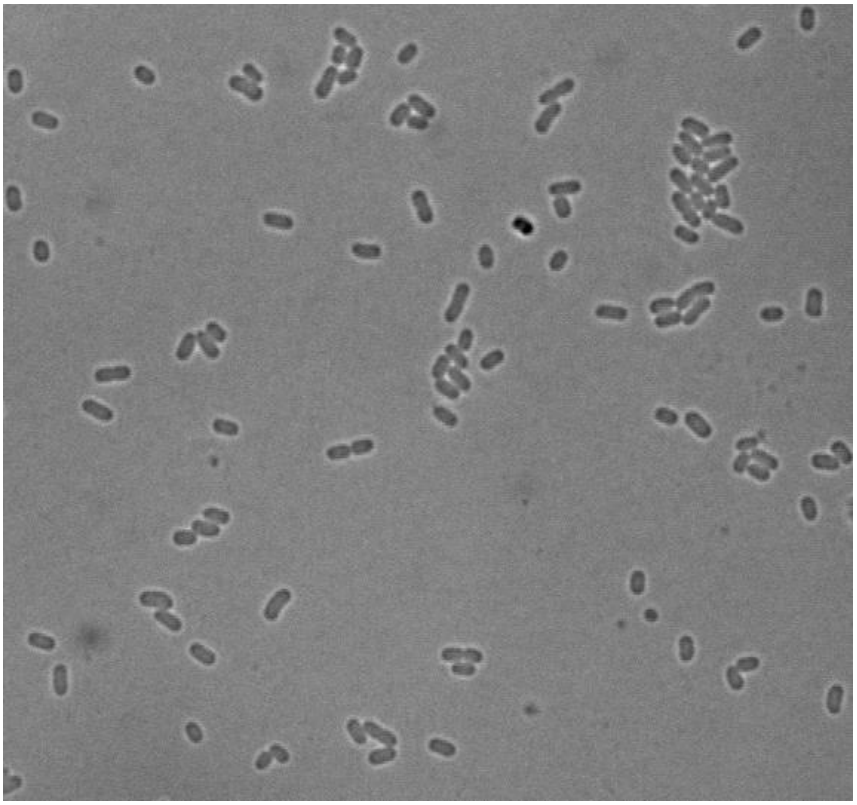

*Figure 1: Cropped Microscope Image*

For decreasing the error and provide better images to be ready for the deep learning part of the project, the images were converted into binary, in other words, black and white.

Because these masked images gave better accuracy results. Since there were lots of raw data images required for training, this process tried to fasten with code. A MATLAB code was written. The code itself is in the appendix part of the report.

An image consists of pixels. The image itself can be represented in RGB (Red Green Blue) or grayscale. The RGB images stored as matrix with 3 channels whereas grayscale images stored as one-dimensional matrix. In image processing and computer vision, the images are proceeded in grayscale. Because grayscale images take lower space. For one pixel of RGB images 8 bits for each colour at total 24 bits area required while in grayscale 8 bits is enough. So processing are easier and faster with grayscale. This why grayscale versions are used. And in the code, RGB image converted into grayscale by rgb2gray() built in function.

Since the image has too much noise, after converting grayscale, the image is convolved with Gaussian Filter via a built-in function named “imgaussfilt()”. The process of separating the bacteria from the background is called segmentation. This mask operation was done by painting pixels belong bacteria as white and black other ones. The result of the code on one of the tried images is as follows:

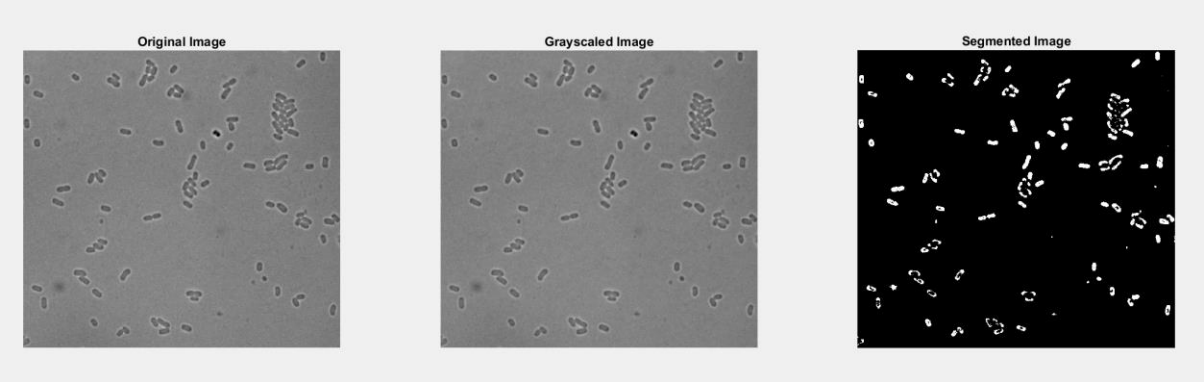


*Figure 2: Result of MATLAB Code*

The above result is one of the optimum result and optimized ones. However, when you look at the result, it is not perfect. Not all the pixels belong bacteria were painted white. Additionally, the original images have some external particles as powder and dust. They were also masked as bacteria in MATLAB code. Therefore, this method was not used.

There is a program called ImageJ for image processing. Since the code was not successful, this program was tried to be used. In each image, bacteria were segmented manually pixel by pixel. The external particles were painted as the background color. One of the results of segmentation process was as follows:

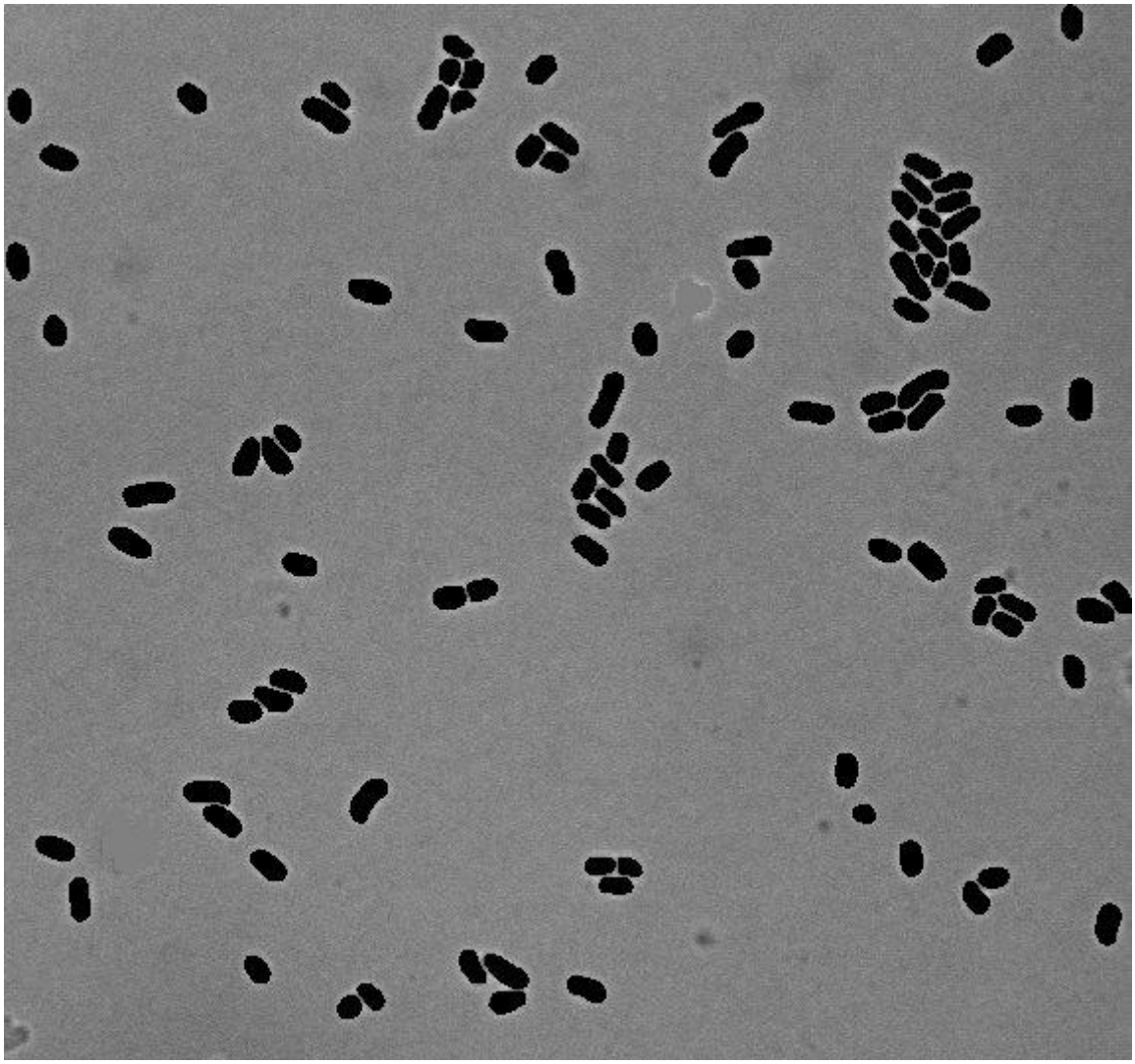

*Figure 3: Result of Manually Segmented Image via ImageJ*

After that, the images were converted into binary and the mask operation was completed in python code written via Google Colab. The code itself is in the appendix part of the report. One of the results of segmentation process was as follows:

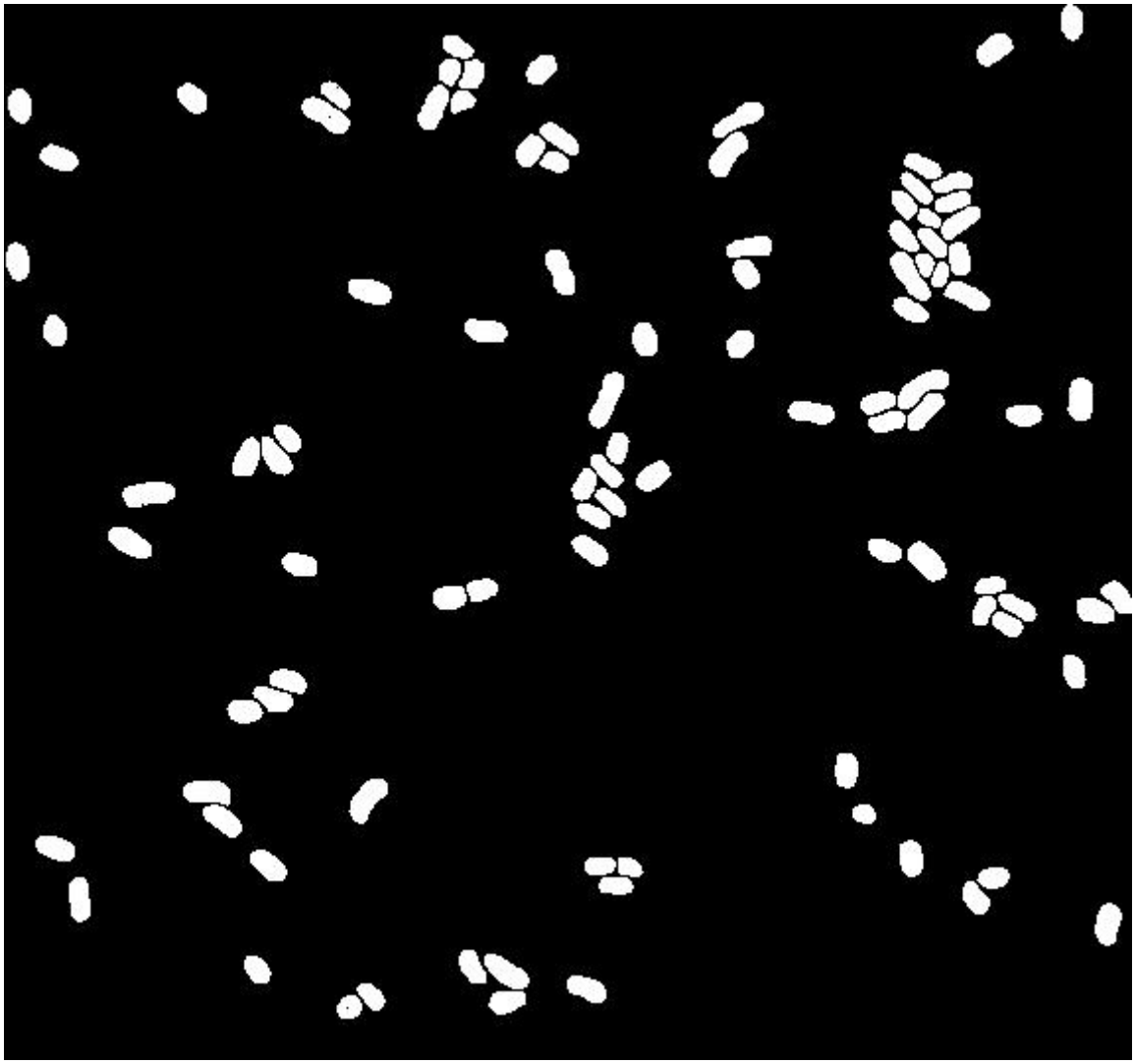

*Figure 4: Black and White Masked Image Result After Python Code*

As shown in the above figure segmentation process was completed successfully. All the bacteria pixels were painted and separated properly from other bacteria pixels. Also, there were no external particles in the masked image.

## 2- U-Net Deep Learning

There are several deep learning solutions for cell segmentation problem. One of them is U-Net which is used for this project.

The architecture of the U-Net is given in below figure:

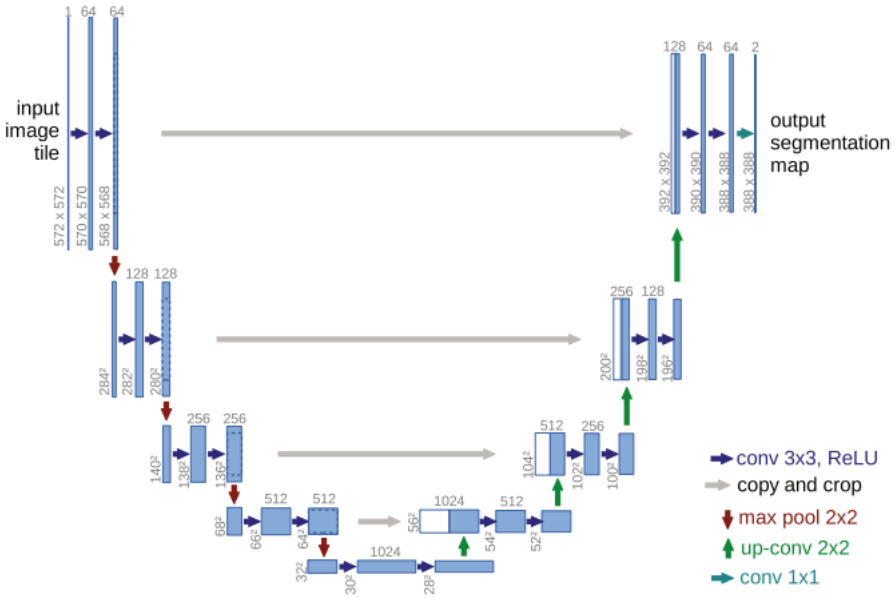


*Figure 5: U-Net Architecture*

This architecture can be implemented in Python by using TensorFlow deep learning framework.

The architecture:

1- Convolution: As mentioned images are kept as a matrix in computers. For convolution operation, you decide on a small window in other words filter. Then the filter moves through the image matrix and provides a dot product. There are several activation functions. In U-Net ReLu is used. ReLu (Rectified Linear Unit) function makes all negative numbers zero and keeps positive ones as it is. In the U-Net convolution architecture given above, the dot product result is applied with ReLu and a new matrix is created (Ronneberger et al., 2015).
2- Max Pool: It is a kind of convolution. Filter moves through the matrix and the max element is assigned to the new matrix. It reduces the size of the feature map and this causes a decrease in process time (Ronneberger et al., 2015).
3- Up Convolution: It is the opposite version of the convolution. To keep the image size as same as at the end, it is required to reunite it. In general convolutional Neural Network (CNN) architecture, this is not required, but it is necessary for segmentation problems (Ronneberger et al., 2015).
4- Copy and Crop: To obtain more features for a given input image in segmentation operation, copy and crop are done (Ronneberger et al., 2015).

By using this U-Net architecture, and implementing it into Python bacteria detection can be done.

# CONCLUSION

The working principle of the U-Net algorithm is understood in general. Some of the codes were written in python via Google Collab. The algorithm itself was improved and shaped to work for the detection of different types of bacteria. Image processing was learned and used. Both MATLAB and Python were used for image processing on images of cells acquired by fluorescent microscopy. Also, a program called ImageJ was learned for the manual editing of images.

# REFERENCES

Falk, T., Mai, D., Bensch, R., Çiçek, Ö., Abdulkadir, A., Marrakchi, Y., ... & Ronneberger, O. (2019). U-Net: deep learning for cell counting, detection, and morphometry. *Nature methods*, *16*(1), 67-70.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Lecture Notes in Computer Science, 234-241. doi:10.1007/978-3-319-24574-4_28

# APPENDIX

## MATLAB Code

```
clear all; close all; clc; % Clear all the previous variables
I_original = imread('Experiment-82_s1t002.jpg'); % Importing the original image

I = I_original;
% Converting GraySacel
[row, col, ch] = size(I); % Getting size of an image
if(ch == 3) % Checking image is rgb or grayscale
    I = rgb2gray(I); % Convert image into grayscale if it is not
end

I = double(I); % Required conversion for pixelwise processes
I_segmented = I;
x_p_1 = 176;
y_p_1 = 143;
intensity_value1 = I(y_p_1,x_p_1);

% Define Threshold
threshold = 105;
I = imgaussfilt(I); % Apply gaussian filtering

[row, col] = size(I);
for j=1:1:col
    for i=1:1:row
        if I(i,j) > threshold % threshold comparison
            I_segmented(i,j) = 0;
        else
            I_segmented(i,j) = 255;
        end
    end
end

%Plotting
I = uint8(I); % Converting the double array to image format
I_segmented = uint8(I_segmented); % Converting the double array to image format

figure(1);
subplot(1,3,1); imshow(I_original); title("Original Image");
subplot(1,3,2); imshow(I); title("Grayscaled Image");
subplot(1,3,3); imshow(I_segmented); title("Segmented Image");
```

## Python Code

```
from skimage.io import imsave,imread
from google.colab.patches import cv2_imshow
import cv2


max_image_num = 54


for x in range(max_image_num):
  fileName = "Experiment-82_s1t0"
  outputFileName = "Masked-Experiment-82_s1t0"


  if x <= 8:
    fileName = fileName + "0"
    outputFileName = outputFileName + "0"


  fileName = fileName + str(x+1)
```

```
outputFileName = outputFileName + str(x+1)
print(fileName)
print(outputFileName)
img = cv2.imread("/content/images_pre/" + fileName + ".jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh2 = cv2.threshold(gray,20,255,cv2.THRESH_BINARY_INV)
imsave("/content/images_post/"+ outputFileName + ".jpg", thresh2)
```