University of Bucharest

Faculty of Mathematics
and Computer Science

Major in Computer Science

Bachelor's Thesis

AUTOMATIC SEGMENTATION OF URINARY CELLS

Graduate
Matei-Constantin Goidan

Thesis Supervisor
Assoc. Prof. Sergiu Nisioi, PhD

Bucharest, June - July 2025

Abstract

This thesis aims to build upon the work presented in *A Clinical Microscopy Dataset to Develop a Deep Learning Diagnostic Test for Urinary Tract Infection* (2024)[8], by extending the segmentation task from binary to multiclass urinary cell segmentation. The study explores and compares convolutional neural network architectures with the goal of understanding their design principles, implementation differences and performances in a biomedical image context. Rather than focusing on the creation of a product, this thesis emphasizes the learning and thinking process involved in training, evaluating and analysing models for microscopy image segmentation. A lightweight web demonstrator is also developed as an addition component for visualization.

Rezumat

Această lucrare de licență își propune să continue cercetarea prezentată în *A Clinical Microscopy Dataset to Develop a Deep Learning Diagnostic Test for Urinary Tract Infection* (2024)[8], extinzând sarcina de segmentare de la o abordare binară la una multiclasă aplicată celulelor urinare. Studiul explorează și compară mai multe arhitecturi de rețele neuronale convoluționale, cu scopul de a înțelege principiile de proiectare, diferențele de implementare și performanțele acestora în contextul imaginilor biomedicale. Lucrarea nu se concentrează pe dezvoltarea unui produs finit, ci pune accent pe procesul de învățare și înțelegere implicat în antrenarea, evaluarea și analiza modelelor pentru segmentarea imaginilor microscopice. Un demonstrator web simplu a fost de asemenea realizat, având rolul de componentă suplimentară pentru vizualizarea rezultatelor.

Contents

Chapter 1

Introduction

1.1 Context and Problem Statement

Urinary tract infections (UTIs) are among the most common bacterial infections, especially affecting women [5]. They can cause major health problems if not detected and treated in time. Microscopic urine examination is among the most reliable methods of identifying signs of infection. However, this manual process is difficult, time-consuming, and subject to human error [8]. In clinical practice, faster diagnostic tools, such as urine dipsticks or cultures are widely used, but they often lack the necessary precision for accurate detection.

Recent advancements in deep learning have enabled significant progress in medical image segmentation [6], including the detection of cellular structures in microscopic images. Convolutional neural networks (CNNs) have proven to be particularly effective tools in biomedical computer vision, providing accurate feature classification and localization.

1.2 Existing Research

A recent paper, *A clinical microscopic dataset to develop a deep learning diagnostic test for urinary tract infection* (2024) [8], introduced a publicly available dataset of urinary microscopic images. Each image has associated a binary segmentation mask (foreground vs. background) and a multiclass mask labelling seven urinary cells types. These images where collected from symptomatic patients, allowing for direct and efficient diagnostics modelling.

The authors of that study proposed a custom U-Net architecture, as seen in Fig. 1.1, that uses patches extracted from the microscopic images, called Patch U-Net, to perform binary segmentation, identifying cells versus the background. Images were divided into patches of $256X256$ pixels, and augmented using technics such as rotations, zooming, flipping and translation, applied using Keras, that helped improve the performance of the model. Although this work demonstrated high performance in segmenting between foreground and background, their model does not perform multiclass segmentation.

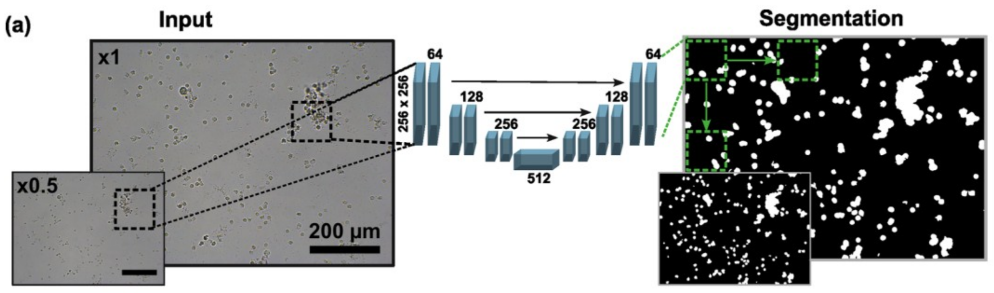


Figure 1.1: Illustration of the Patch U-Net segmentation workflow. High-resolution microscopic images are divided into smaller patches of size 256×256 for training and inference. These patches are passed through a U-Net architecture to produce binary segmentation masks, which are then reassembled to form the final segmented output.

1.3 Motivation

In this thesis, I want to take the next logical step and change the segmentation task from a binary classification problem into a multiclass classification one, where each pixel in the image must be assigned to a specific cell type or to the background. For this I have introduced additional complexity in the training and evaluation phase in order to achieve richer and more informative segmentation outputs.

The motivation behind it lies in understanding how convolutional neural network architectures are designed, trained, and evaluated when applied to image segmentation problems. While the problem addressed is related to urinary cell segmentation, the main focus is on how these models behave, how their complexity and performance differs and what trade-off exist when choosing one architecture over another. The UTI context serves as a practical application for exploring and testing the capabilities of deep learning models. As earlier said, the main focus of this thesis is to highlight the technical learning process through implementation, experiments and analysis.

1.4 Contributions

The main objective of this thesis is to explore and compare several convolutional neural network architectures applied to urinary cell segmentation in microscopic images. Specifically:

1. To extend the task from binary to multiclass segmentation using the annotated dataset introduced by *Liou et al* [8].

2. To implement three convolutional neural network architectures (a simple CNN, U-Net, and DeepLabV3+) and use them in the urinary cell segmentation.
3. To evaluate and compare the architectures in regards to performance and complexity.
4. Additionally, to support this process, create a lightweight web-based demonstrator that allows interactive visualization of segmentation outputs.

1.5 Structure of the Thesis

The remainder of the thesis is structured as follows:

- Introduces the dataset and the deep learning architectures used, along with training procedures and evaluation metrics.
- Presents the implementation details, including training results and the web-based demonstrator used to visualize model outputs.
- Conclude the thesis by summarizing the results and outlining possible directions for future development and research.

Chapter 2

Convolutional Neural Networks for Image Segmentation

2.1 Overview of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models that have proven exceptionally effective in patterns recognition, such as image classification and object detection [1]. Unlike typical fully connected networks, where each neuron in a layer is connected to all the neurons in the preceding layer, convolutional neural networks take advantage of the space arrangement of images through the use of convolutional layers that apply filters sliding across the image to extract the most relevant features. This architectural design allows CNNs neurons to be connected only to a smaller portion from the previous layer, helping the network to learn representations of the input data [7].

A typical CNN is composed of multiple building blocks [1, 7]:

- *Convolutional layers*: which apply learnable filters on the input image to extract important features in the form of feature maps. These layers are parametrized by filter size, stride, padding, number of filters, and activation functions. In this thesis, the convolutional layers are implemented using `Conv2D`.
- *Pooling layers*: such as `MaxPooling2D`, reduce the spatial dimensions of feature maps, making the representation more compact and resistant to minor shifts and noise.
- *Activation functions*: including `ReLU` and `Softmax`, introduce non-linearity and allow the network to model complex relationships in the data.
- *Flatten layers*: which convert multidimensional feature maps into one-dimensional vectors, usually for transition to dense layers.
- *Dense layers*: often used near the output, where each neuron is connected to all

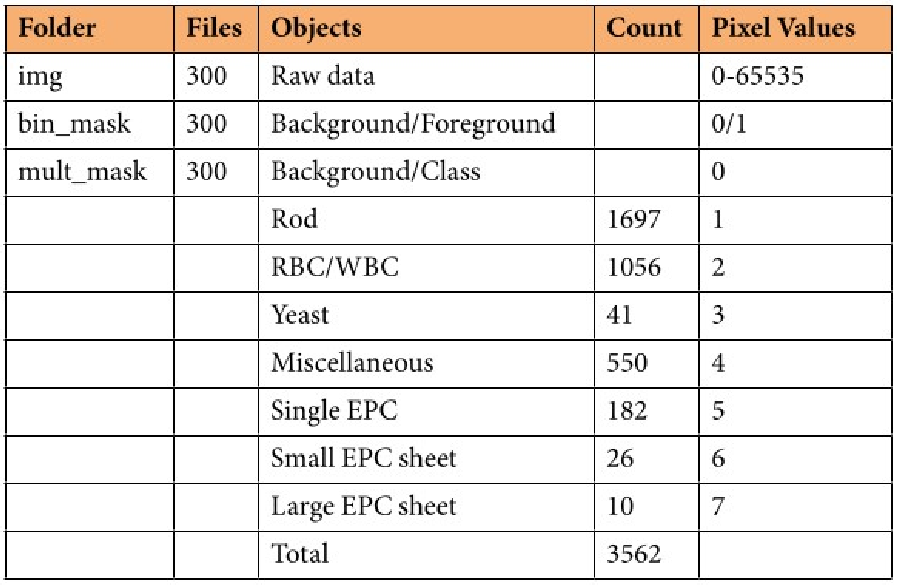
neurons in the previous layer and contributes to the final classification or regression decision. In semantic segmentation tasks, these are typically avoided or replaced by upsampling layers.

In semantic segmentation tasks, including biomedical segmentation, CNNs are trained to assign a class label to every pixel in an image. This requires not only understanding the local context but also maintaining global spatial information. Standard CNN architectures are typically adapted into encoder-decoder designs, where the encoder compresses the spatial information into feature representations, and the decoder reconstructs pixel-level class maps from those features.

CNNs have become the preferred solution for biomedical image analysis due to their ability to generalize well from annotated examples, handle noise, and extract meaningful structures from complex data. Their success is largely attributed to their capacity to learn end-to-end mappings directly from raw image inputs to pixel-level outputs, without the need for handcrafted features [6].

2.2 Dataset

The dataset contains 300 microscopic images, with $3,562$ manually annotated urinary cells. This images were acquired using standard laboratory equipment and have a size of $1392X1040$ pixels in `.tif` format. Therefore, the images are split equally between three directories: training, validation and testing. Each multiclass mask labels every pixel with an integer from 0 to 7. Fig. 2.1 provides an overview of the dataset.



Folder	Files	Objects	Count	Pixel Values
img	300	Raw data		0-65535
bin_mask	300	Background/Foreground		0/1
mult_mask	300	Background/Class		0
		Rod	1697	1
		RBC/WBC	1056	2
		Yeast	41	3
		Miscellaneous	550	4
		Single EPC	182	5
		Small EPC sheet	26	6
		Large EPC sheet	10	7
		Total	3562	

Figure 2.1: Dataset structure and class distribution. The dataset is organized into folders containing original microscopy images along with their corresponding binary and multi-class masks.

To better understand the variety of cell types and their mask, several random images, alongside their multiclass masks, were selected from the dataset and shown in Fig. 2.2.

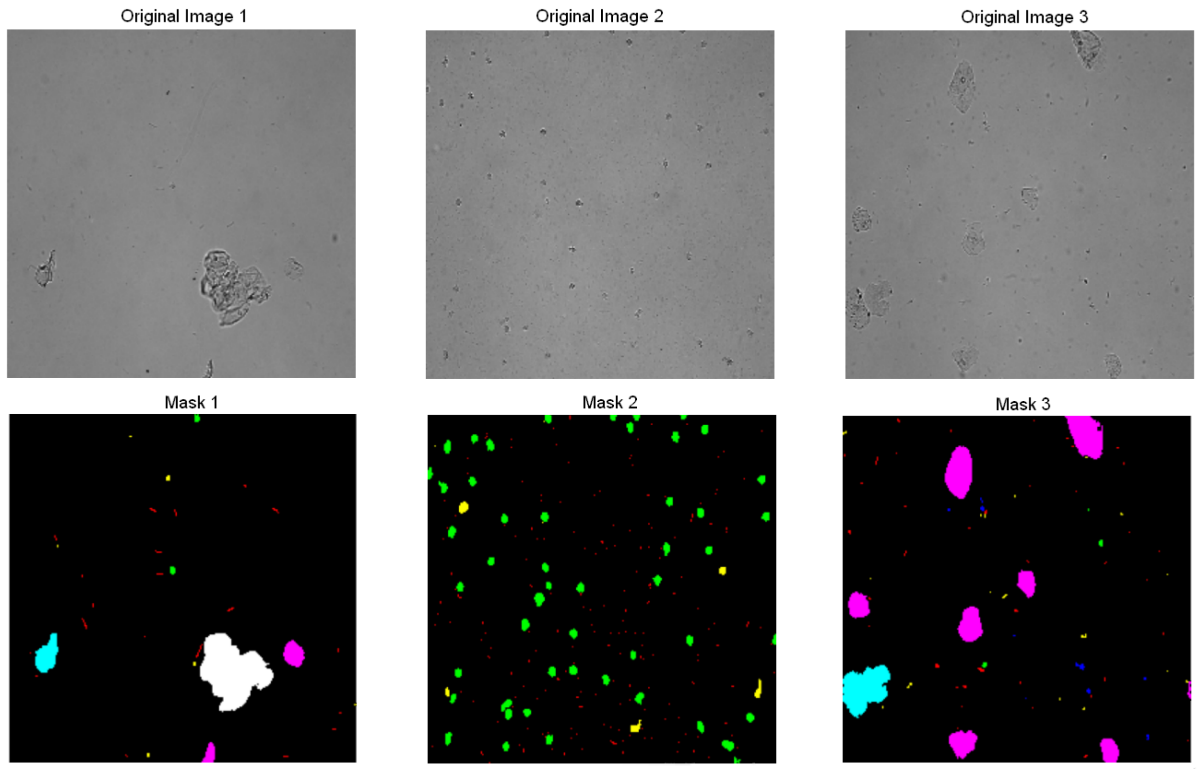


Figure 2.2: Random images with their mask selected from the dataset showcasing cells of different types and sizes.

2.3 Architectures

This section presents details of three convolutional neural network architectures used in image segmentation. My goal is not only to describe the technical components of each model, but also to highlight the reasoning behind their selection order and the process of moving from a simpler architecture to a more efficient one. The models included in this thesis are: a Simple CNN, U-Net and DeepLabV3+. Each of them illustrate a different level of complexity and segmentation capability, offering a comparative perspective on biomedical image analysis.

2.3.1 Simple CNN

As a starting point, I develop a simple Convolutional Neural Network to test how a minimal architecture performs on a multiclass segmentation task.

I used the original resolution of the images of 1040×1392 pixels to observe the performance under minimal interventions. I normalized pixel values to the $[0, 1]$ range and convert the segmentation masks to `uint8` format, with each integer value representing one of the eight defined classes (background and seven cell types). I apply basic data augmentation, including horizontal and vertical flips, to increase diversity in the training data.

The CNN model, as presented in Fig. 2.3 was implemented using the Keras Functional API [3]. Its structure includes:

- **Encoder:** two convolutional layers with `ReLU` activation and (3×3) convolutions (with 16 and 32 filters respectively), each followed by a (2×2) max pooling operation to reduce spatial dimensions.
- **Decoder:** two transposed convolutional layers, using (2×2) convolutions and stride 2, to progressively upsample the feature maps.
- **Output:** a final (1×1) convolutional layer with Softmax activation that outputs 8 channels, corresponding to the segmentation classes.

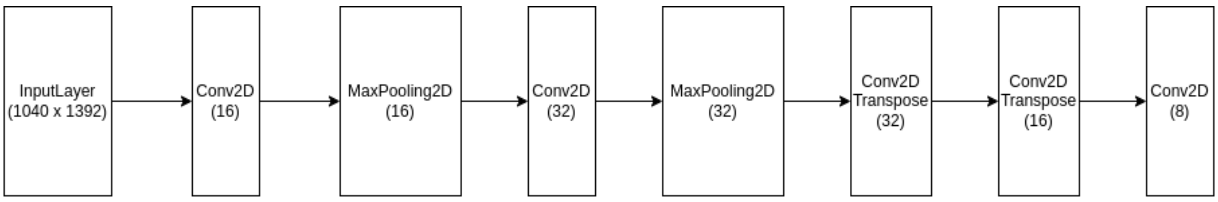


Figure 2.3: Architecture of the simple CNN used for multiclass segmentation, composed of convolutional layers for feature extraction and transposed convolutions for upsampling.

For training, I used `SparseCategoricalCrossentropy`, occasionally combined with auxiliary metrics such as Dice Score or Intersection-over-Union (IoU) during evaluation for loss function. As an optimizer I used Adam with a learning rate of 1×10^{-4}. The model compiled with 100 epochs in order to try to get better results.

The model performs well in detecting the background, having a Pixel Accuracy of 0.98, Mean Dice Score of 0.89 and Mean IoU of 0.80. However, it is unable to detect actual cells, never getting lose bellow 2.0, and the results are heavily biased by the dominance of the background class in most images.

Metric	Accuracy	Dice	IoU	Loss
Value	0.98	0.89	0.80	2.0

Table 2.1: Evaluation metrics of Accuracy, Dice, IoU and Loss for the Simple CNN model

2.3.2 U-Net

Given that the simple CNN does not perform adequately, I move on to another architecture, a U-Net implemented for multiclass segmentation. U-Net is a well-established convolutional architecture originally designed for biomedical image segmentation, and is known for its ability to accurately localize and distinguish small structures within images [11]. Its success lies in its encoder-decoder design with skip connections, which allow spatial information to flow directly from downsampling to upsampling layers.

Before training the model, I apply a series of preprocessing steps that follow the procedure from the original publication by *Liou et al* (2024) [8]. Each image is resized to 256×256 pixels and normalized to the range $[-1, 1]$. The masks are also resized to 256×256 using nearest-neighbour to preserve the class labels. These are then converted to one-hot encoded tensors with a depth of 8, representing the target classes. This allows the model to output per-pixel class probabilities using Softmax activation.

To improve generalization and reduce overfitting, I incorporate a variety of data augmentation strategies. These include random horizontal and vertical flips, random brightness adjustment with a maximum delta of 0.2, random contrast adjustment with a factor sampled between 0.8 and 1.2 and random rotations between 0, 90, 180 and 270 degrees.

I construct the U-Net architecture using a custom convolutional block, which I define and reuse throughout the model. Each block consists of two (3×3) `Conv2D` layers, each followed by `InstanceNormalization`, `BatchNormalization`, and ReLU activation. At the end of the block, I apply `SpatialDropout2D` with a rate of 0.2. All convolutional layers use `He` normal initialization and L2 regularization. These blocks are used in the encoder, bottleneck, and decoder parts of the network.

The full U-Net model, as seen in Fig. 2.4, is composed of the following components:

- **Encoder:** four `conv_block` layers with 32, 64, 128, and 256 filters, each followed by `MaxPooling2D`
- **Bottleneck:** a `conv_block` with 512 filters
- **Decoder:** four decoding stages, each consisting of:
  - a `Conv2DTranspose` layer for upsampling,
  - concatenation with the corresponding encoder output (Skip Connection),
  - a `conv_block` with decreasing number of filters: 256, 128, 64, and 32
- **Output:** a (1×1) `Conv2D` layer and Softmax activation, producing 8 output channels

For training, I use a loss function that combines:

- **Focal Loss**, to reduce the influence of well-classified classes and focus on harder examples.
- **Multiclass Dice Loss**, to directly optimize overlap between prediction and ground truth.

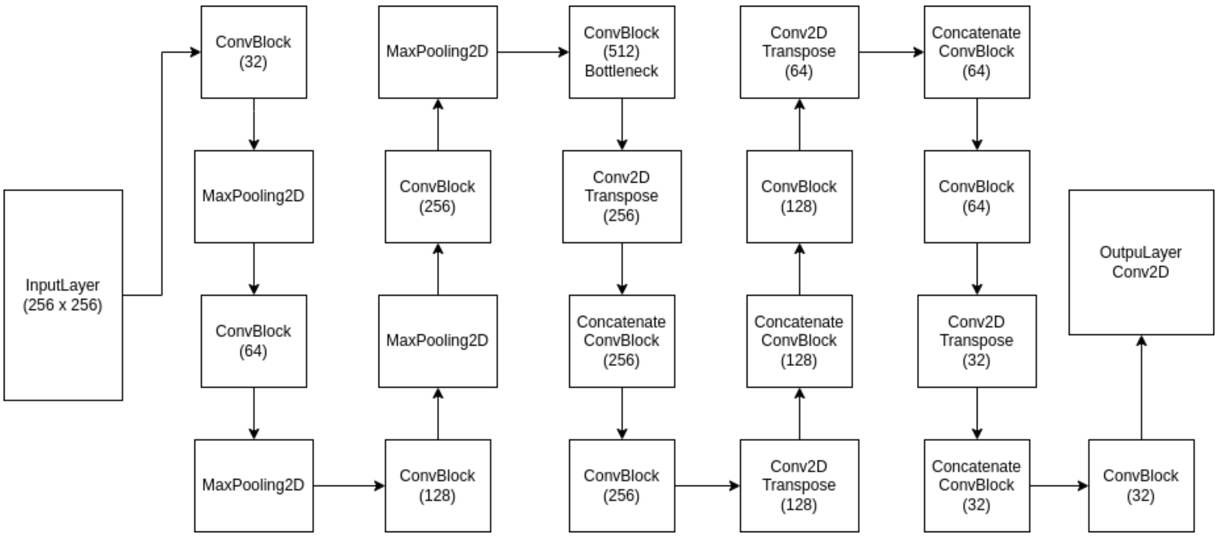


Figure 2.4: Architecture of the U-Net model used for multiclass urinary cell segmentation. The network consists of four encoding blocks, a bottleneck layer with 512 filters, and four decoding blocks. Each ConvBlock includes two convolutional layers followed by normalization and activation functions. Concatenate blocks are Skip connections that are used to concatenate encoder features with decoder layers.

Class-specific weights were introduced in the loss function to compensate for the severe imbalance in the dataset. For instance, rare classes like Rod or Yeast were assigned higher penalties. Small categories such as Rod and Yeast receive higher penalties to increase their influence during training. As seen in Tab. 2.2

Class	BKG	Rod	RBC/WBC	Yeast	Misc	single EPC	small EPC	large EPC
Weight	1	20	5	15	10	1	5	15

Table 2.2: Class weights used during training to address class imbalance in the multiclass segmentation task. Higher weights are assigned to underrepresented or clinically important categories such as Rods and Yeast, while dominant classes like background receive a lower weight.

Initially, I train the model for 50 epochs to observe its behaviour. The model quickly learns to segment the background and bigger cells but struggles with smaller ones. To address this, I extend training to 100 epochs, which helps the model learn those specific cells. Despite not reaching optimal performance on all classes, the U-Net model shows clear improvements over the simple CNN. It learned to segment dominant classes with high precision and began recognizing previously ignored rare classes after optimization.

Class	BKG	Rod	RBC/WBC	Yeast	Misc	single EPC	small EPC	large EPC
Dice Score	0.995	0.127	0.613	0.000	0.074	0.653	0.089	0.263

Table 2.3: Dice Score metrics for the U-Net model after 100 epochs. Showing the model still can't detect small sized cells.

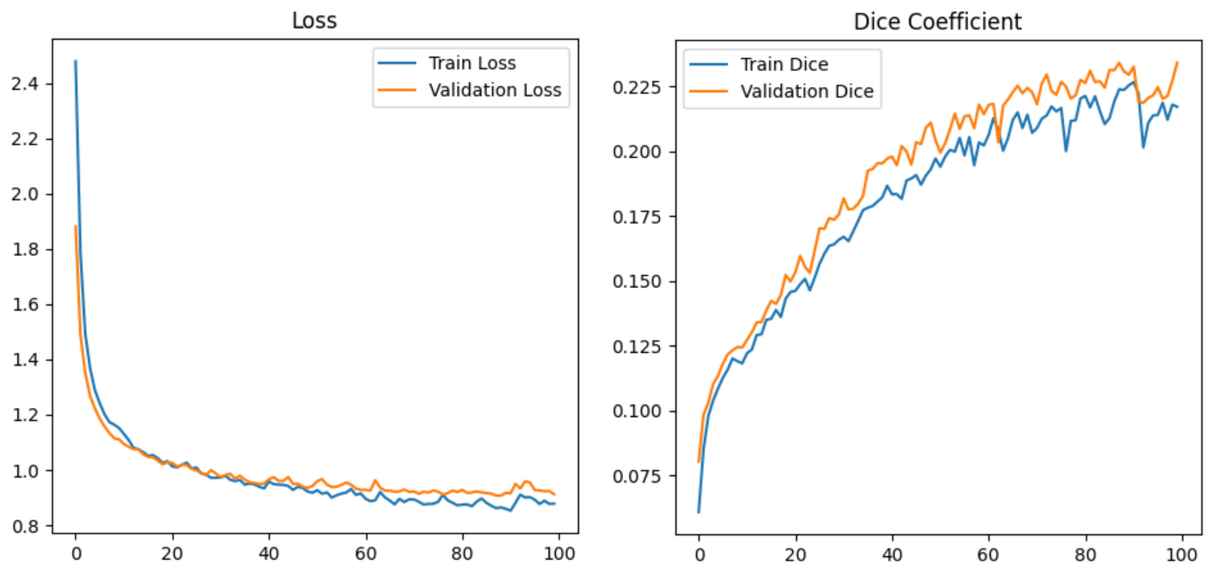


Figure 2.5: Loss and Dice Coefficient over 100 epochs for the U-Net model. The curves show stable convergence and indicate that the model generalizes well but doesn't optain a Loss smaller than 0.87 and a Dice Score greater than 0.23.

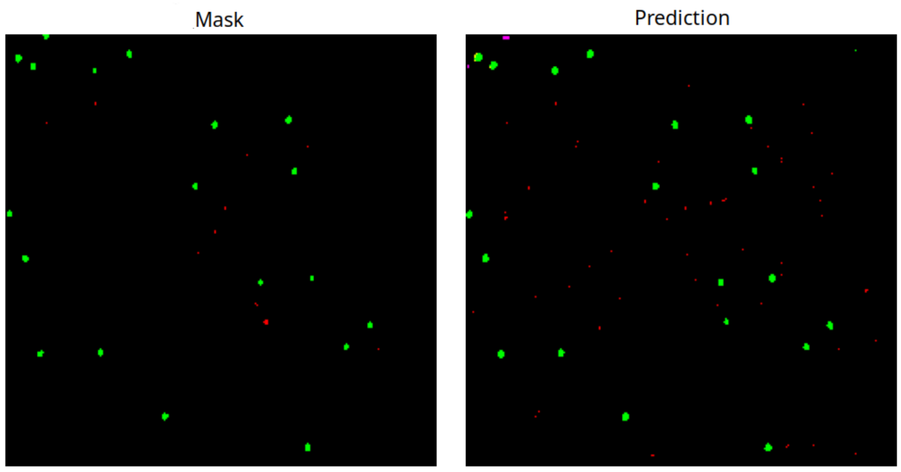


Figure 2.6: Visual comparison between the ground truth segmentation mask and the prediction generated by the U-Net model. RBC/WBC cells are shown in green, while Rod cells are highlighted in red. This example is selected to illustrate the model's performance on both a well-segmented class and a class for which precision is lower.

2.3.3 DeepLabV3+

DeepLabV3+, developed by Google, is a cutting-edge architecture that combines deep feature extraction and edge refinement. I selected this model for its superior performance on segmentation tasks and its capabilities of handling class imbalances and structural variability [2].

The backbone I use in the implementation is Xception (Extreme Inception), a CNN architecture, which replaces traditional convolutions with depthwise separable convolutions, which factorize a standard convolution into two steps: a depthwise convolution that applies one filter per input channel, and a pointwise convolution (1×1) that combines the outputs. This decomposition significantly reduces the number of parameters and computations, making the model both faster and more memory efficient, while preserving accuracy [4]. Xception was chosen based on the original DeepLabV3+ paper and is available as a pre-trained model in Keras [12], with intermediate layers well-suited for Skip connections.

The architecture consists of the following components, as depicted in Fig. 2.7:

- **Encoder:** a pretrained Xception model with depthwise separable convolutions and no top classification layers
- **ASPP module:** composed of five parallel branches:
  - global average pooling followed by 1×1 convolution and upsampling,
  - 1×1 convolution,
  - three 3×3 convolutions with dilation rates of 6, 12, and 18
- **Feature projection:** the concatenated ASPP output is processed by a 1×1 convolution to reduce dimensionality
- **Skip connection:** features from the `block3_sepconv2_bn` layer of the encoder are projected using a 1×1 convolution and concatenated with the upsampled ASPP output
- **Decoder:** two convolutional layers refine the concatenated features
- **Output:** a 1×1 convolution layer followed by Softmax activation produces 8-class per-pixel predictions, upsampled to the original input resolution

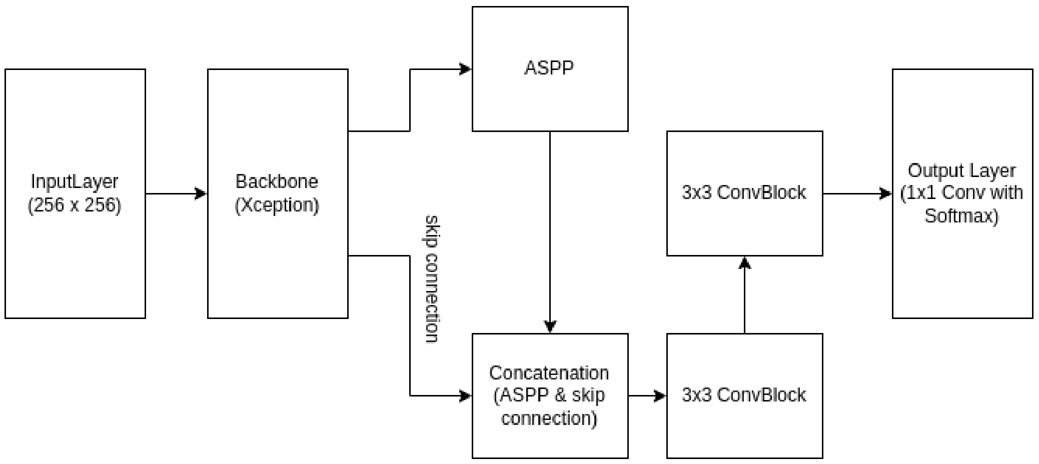


Figure 2.7: Architecture of the DeepLabV3+ model. The input passes through a pretrained Xception backbone, followed by ASPP concatenated with the encoder via Skip connections. The decoder is composed of two 3×3 convolution blocks and the final prediction is produced by a 1×1 convolution layer with Softmax activation.

The model is compiled using the Adam optimizer and a loss function similar to the one implemented for U-Net. The same class weights are applied in order to ensure a fair comparison between U-Net and DeepLabV3+. Starting from the first 100 epochs, I observe a clear improvement in the Dice Coefficient, with the model reaching a score of 0.6419. After training, DeepLabV3+ achieves even higher Dice Scores across most classes, with the exception of Rod and RBC/WBC, where performance remains limited due to class imbalance and visual similarity with other cell types, as shown by Tab. 2.4.

Class	BKG	Rod	RBC/WBC	Yeast	Misc	single EPC	small EPC	large EPC
Dice Score	0.995	0.041	0.497	0.000	0.083	0.797	0.398	0.664

Table 2.4: Dice Score metrics for the DeepLabV3+ model after 100 epochs. Even though the model has improved the scores for most classes we can see a decrease for Rod and RBC/WBC and no progress on Yeast.

Overall, DeepLabV3+ outperformed both the simple CNN and U-Net architectures in terms of class-wise consistency. Its modular design, multi-scale feature extraction, and refinement capabilities make it a strong candidate for multiclass biomedical image segmentation.

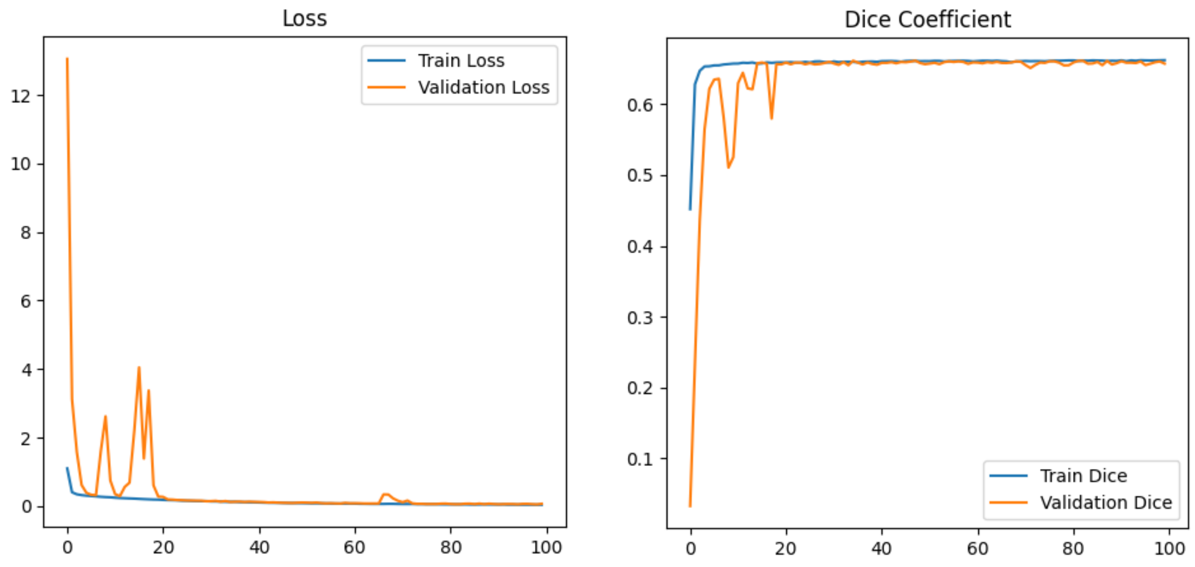


Figure 2.8: Loss and Dice Coefficient over 100 epochs for the DeepLabV3+ model. The model reaches a Loss score of 0.034 and a Dice score of 0.64.

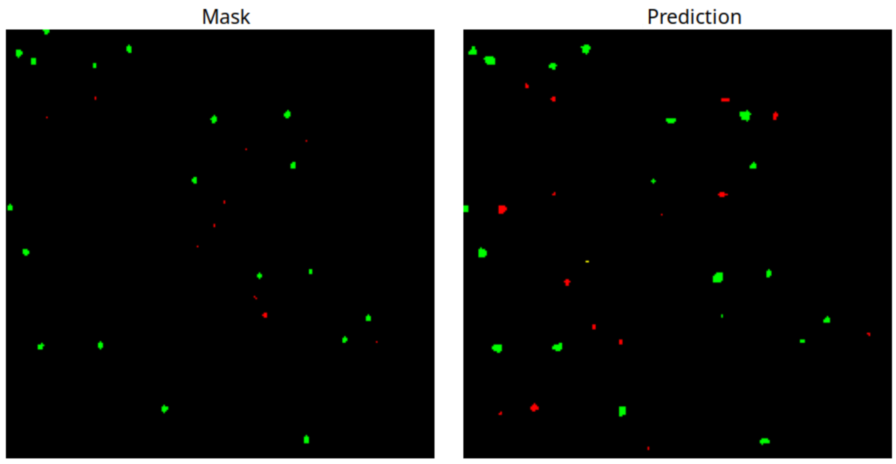


Figure 2.9: Visual comparison between the ground truth segmentation mask and the prediction generated by the DeepLabV3+ model. The same image is choose in order to better make the distinction between U-Net and DeepLabV3+. At the same time, showcasing RBC/WBC and Rod cells that got a smaller score with the model.

2.4 Results

The three architectures evaluated in this thesis: Simple CNN, U-Net and DeepLabV3+, demonstrate varying capabilities in their ability to handle cell types. The Simple CNN achieves high pixel accuracy, but fails to detect most cell structures. U-Net is significantly better at learning global and local features, showing better results across all classes. DeepLabV3+ further improves the segmentation accuracy through its encoder-decoder structure and ASPP module, achieving the highest Dice Score and best class detection.

These results confirm that segmentation performance improves consistently with model complexity. DeepLabV3+ proves to be the most effective for multiclass urinary cell segmentation, although it also requires the most computational resources.

Chapter 3

Implementation of the Demonstrator

3.1 Overview

As an aditional component to this thesis, I developed a lighweight demonstrator application named *CellSeg*, which offers a practical interface for testing the DeepLabV3+ segmentation model that I trained. The application allows users to upload microscopy images and receive real time segmentation results directly in the browser.

The system follows a client-server architecture and is composed of three main components:

- **React.js** frontend that handles uploaded images and display results.
- **FastAPI** backend (written in Python) that handles the processing of images and model inference.
- **DeepLabV3+** model implemented in Keras [3] and loaded on the server for prediction.

3.2 Backend and Model Integration

The backend of the application is utilizing FastAPI, a modern Python framework for building web APIs [10]. The backend is responsible for receiving image files from the frontend, processing them, running the pretrained DeepLabV3+ model, and return the segmentation mask to the user.

The DeepLabV3+ model is loaded into memory when the FastAPI server starts. Each request to the prediction endpoint includes an image file, which is converted from its encoded format into a NumPy. The image is then resized and normalized to satisfy the model's input requirements (size 256×256, RGB channels, pixel values in $[-1, 1]$).

The output tensor is transformed into a color coded segmentation mask using a predefined colormap, and the result is returned as a PNG image. This approach allows the integration of the model into a web interface, without exposing the user to its internal complexity.

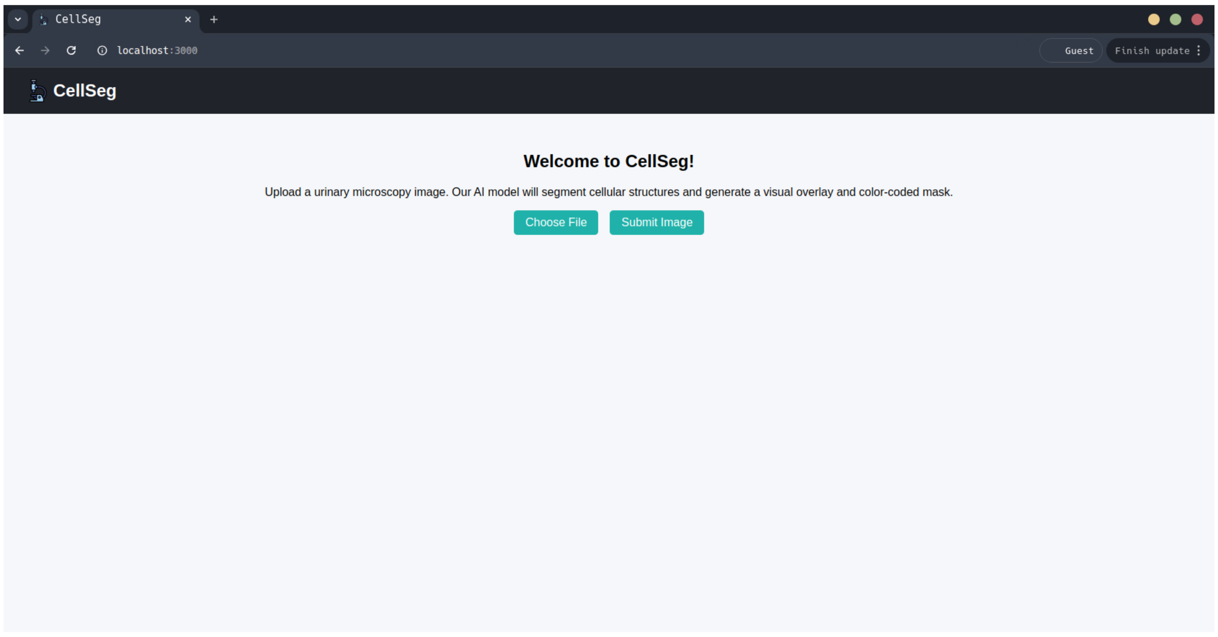
3.3 Web Interface

I used `React.js` to make the frontend of the *CellSeg* demonstrator, a popular JavaScript library for building interactive user interfaces [9]. The web interface allows users to upload microscopic images in standard formats (such as PNG or TIF) and displays the segmentation result returned by the backend.

Once the segmentation mask is rendered, the interface displays the original image on the left and an overlay of the original image with the predicted mask on the right. A slider beneath the overlay allows the user to adjust the opacity of the original image, making it easier to assess the accuracy of the segmentation. This functionality provides a quick visual inspection method for evaluating whether urinary cells are correctly segmented.

3.4 Example Usage

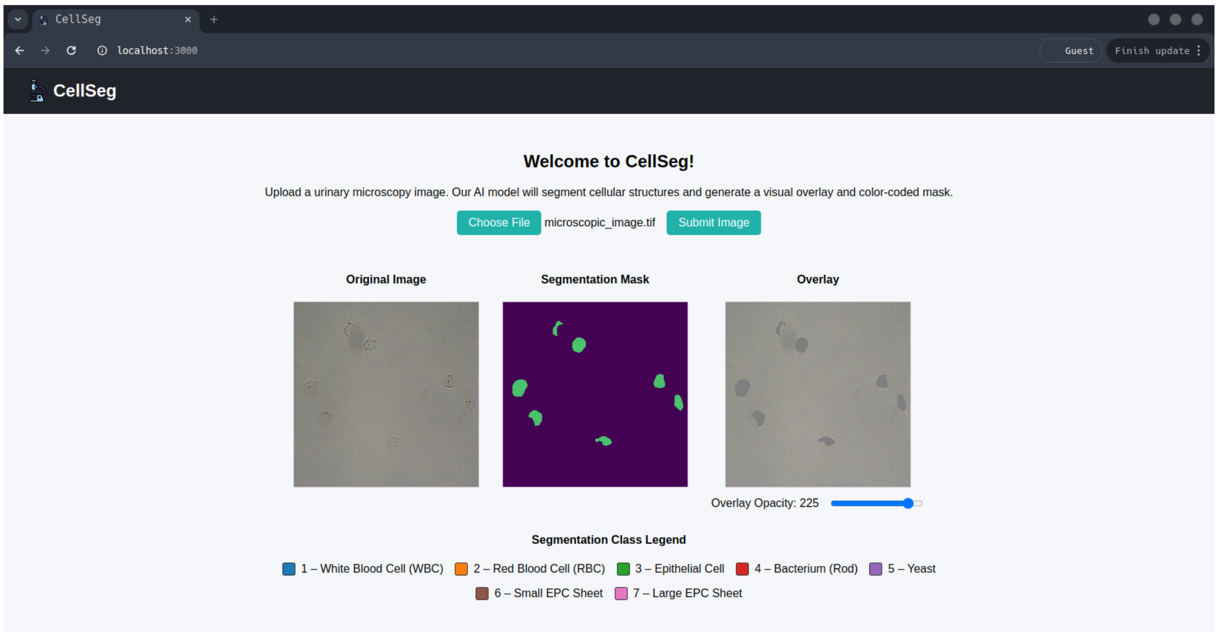
In the final section of the demonstrator, I will present how users interact with the *CellSeg* application. The initial stat of the interface contains two buttons, that allow the users to upload a microscopic image and submit it for segmentation, as presented in Fig. 3.1.


Welcome to CellSeg!

Upload a urinary microscopy image. Our AI model will segment cellular structures and generate a visual overlay and color-coded mask.

Figure 3.1: Start page of the CellSeg demonstrator, showing the upload controls before any image is submitted

After submitting an image, the frontend senad a `POST` request to the `/predict/` endpoint of the FastAPI backend. This returns a JSON response containing base64 encoded versions of the original image, the predicted segmentation mask and the overlay between the mask and the original. These base64 images are decoded and rendered side by side Additionally, a collor legend is displayed beneath the three output images. Each class is assigned a distinct RGB color, and the legend shows both the class name and its corresponding color. Fig. 3.2 illustrates that.



Welcome to CellSeg!

Upload a urinary microscopy image. Our AI model will segment cellular structures and generate a visual overlay and color-coded mask.

Overlay Opacity: 225

Segmentation Class Legend

🟦 1 – White Blood Cell (WBC) 🟧 2 – Red Blood Cell (RBC) 🟩 3 – Epithelial Cell 🟥 4 – Bacterium (Rod) 🟪 5 – Yeast
🟫 6 – Small EPC Sheet 🟪 7 – Large EPC Sheet

Figure 3.2: Segmented output displayed in the browser. From left to right: original image, predicted segmentation mask, and overlay with adjustable opacity. Below the images, a legend shows the color assigned to each segmentation class.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

The results demonstrate an improvement in segmentation performance as the architecture complexity increases. DeepLabV3+ outperforms both previous models in terms of finding urinary cells. To support this and facilitate experimentation, I also build a demonstrator named *CellSeg*, which I use extensively to test case to case specific prediction in order to improve the model.

Additionally, the main objective of familiarizing with convolutional neural networks and understanding the potential that artificial intelligence has when applied to solve medical problems was achieved. While I did not succeed in building a model capable of accurately segmenting all urinary cell types, the work carried out has enhanced the understanding of deep learning and image analysis.

4.2 Limitations

While the results obtained in this thesis are promising, several limitations should be acknowledged. First, the dataset used for training and evaluation is heavily imbalanced. Certain classes such as Yeast or Rod are significantly underrepresented. In addition, in some cases, these objects are so small that they occupy only a few pixels or the cell structures are so similar between them, making them easy to confuse. This combination makes it harder for the model to learn features of these classes. Second, the total number of annotated samples is relatively small for a deep learning task. Even with data augmentation, the risk of overfitting remains high. Fig. 4.1 illustrates that.

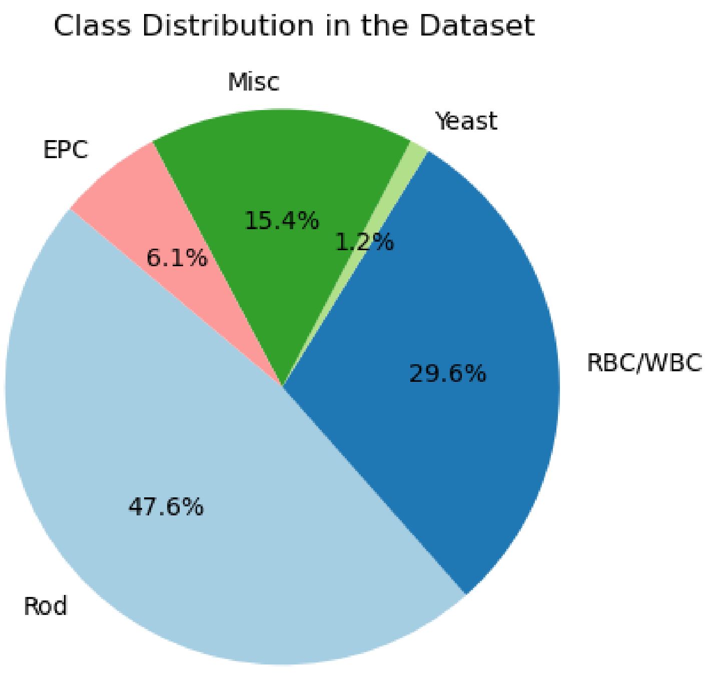


Figure 4.1: Visual representation of class imbalance in the dataset. The low frequency of certain cell types affect the model's ability to learn.

4.3 Future Work

After working with this dataset, a next step is to narrow the scope of segmentation to focus only on cell types that are well represented and are visually easier to distinguish. This would allow for a cleaner evaluation of model performance and its practical potential. This approach could be further complemented by incorporating external datasets to improve the overall robustness of the model.

The demonstrator, *CellSeg* could also be expanded by adding additional helpful tools. These could include the ability to download the generated segmentation mask or to display a count of each cell type detected by the model. Such features would improve the application and make testing and experimentation more effective.

Bibliography

[1] Shanmugamani Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a Convolutional Neural Network." In: *2017 International Conference on Engineering and Technology (ICET)*. IEEE. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2018, pp. 801–818.

[3] François Chollet et al. *Keras*. https://keras.io. Accessed: 09-05-2025. 2015.

[4] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 1251–1258. DOI: 10.1109/CVPR.2017.195.

[5] Family Medicine Austin. *Breaking the Cycle: Combating Antibiotic Resistance in UTI Treatments*. Accessed: 07-05-2025. 2024. URL: https://familymedicineaustin.com/breaking-the-cycle-combating-antibiotic-resistance-in-uti-treatments/.

[6] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, and Jun Chen. "Recent advances in convolutional neural networks." In: *Pattern Recognition* 77 (2018), pp. 354–377. DOI: 10.1016/j.patcog.2017.10.013.

[7] Andrej Karpathy and Fei-Fei Li. *CS231n Convolutional Neural Networks for Visual Recognition: Convolutional Networks*. https://cs231n.github.io/convolutional-networks/. Accessed: 25-03-2025. 2016.

[8] Kevin Liou, Aaron Rosenberg, Guneet Sandhu, Pooja Zaveri, Rachel Ostfeld, David Lansky, Manisha Narayan, Lauren G. McCoy, and Steven Horng. "A clinical microscopy dataset to develop a deep learning diagnostic test for urinary tract infection." In: *Scientific Data* 11.1 (2024). Accessed: 22-04-2025. DOI: 10.1038/s41597-024-02975-0. URL: https://www.nature.com/articles/s41597-024-02975-0.

[9] Meta Open Source. *React – A JavaScript library for building user interfaces.* https://react.dev/. Accessed: 18-05-2025. 2024.

[10] Sebastián Ramírez. *FastAPI.* https://fastapi.tiangolo.com/. Accessed: 04-06-2025. 2023.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI).* Vol. 9351. Lecture Notes in Computer Science. Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[12] Keras Team. *Keras Applications: Xception.* https://keras.io/api/applications/xception/. Accessed: 02-06-2025. 2024.