Dedications

I dedicate this thesis:

To all of you, I dedicate this work.

Acknowledgments

With immense pleasure, we express our heartfelt thanks and deep gratitude to all those who contributed to the development of this final year project. We are particularly grateful to Ms. Fourati Rahma, our academic supervisor, for her support throughout this internship, through her enriching conversations, constructive criticism, and review of this work.

Finally, to all the members of the jury for the honor they bestow upon us by participating in the examination of our work. We are particularly grateful to Ms. Fatma Ben Said and Ms. Bouarada Najla. Thank you for your time, dedication, and commitment to academic excellence.

Contents

Acknowledgments ii

List of Figures v

List of Tables vii

List of Abbreviations viii

Introduction 1

1 General Context 2

1.1 Introduction . . . 2

1.1.1 Project Context and Importance . . . 2

1.1.2 Project Objectives . . . 3

1.2 Overview of colorectal cancer polyps . . . 4

1.2.1 Symptoms . . . 5

1.2.2 Causes . . . 5

1.2.3 Diagnosis . . . 6

1.2.4 Treatment . . . 7

1.3 Concept and technique of segmentation . . . 8

1.3.1 Segmentation of Medical Images . . . 8

1.3.2 Concepts of Deep Learning . . . 11

1.4 State of the Art . . . 12

1.4.1 General Description . . . 12

1.4.2 CNN-based Models . . . 13

1.4.3 UNet and its Variants . . . 13

1.4.4 Transformer-based Models . . . 14

1.5 Conclusion . . . 16

2 The proposed Method for Polyp segmentation 17

2.1 Introduction . . . 17

2.2 CRISP methodology . . . 17

2.3 Business Understanding . . . 18
2.4 Data Understanding . . . 19
2.4.1 Source and Structure . . . 19
2.4.2 Schema . . . 19
2.4.3 Limitations . . . 20
2.4.4 Data Preprocessing . . . 20
2.4.5 Data Exploration Image . . . 23
2.4.6 Check image basic components . . . 23
2.4.7 Check image width and height . . . 24
2.4.8 Check Polyp Positions . . . 25
2.4.9 Check Polyp Size and Count . . . 26
2.4.10 Check Image Hue, Brightness, and Saturation . . . 26
2.5 Data Partitioning . . . 27
2.5.1 Set Up Train-Test Split Path . . . 27
2.5.2 Data Augmentation . . . 29
2.6 Modeling . . . 30
2.6.1 Model Choice . . . 30
2.6.2 The U-Net Architecture . . . 30
2.6.3 Training Phase . . . 33
2.7 Evaluation . . . 36
2.7.1 Quantitative Evaluation . . . 36
2.7.2 Qualitative Evaluation . . . 41
2.8 Deployment . . . 43
2.8.1 MLOps Methodology . . . 43
2.8.2 Streamlit Application . . . 44
2.9 Conclusion . . . 46
2.10 Conclusion . . . 46

Conclusion and Future Works **48**

Bibliography **50**

List of Figures

1.1 Colon polyps 4
1.2 Convolutional Neural Networks (CNN) 12

2.1 CRISP Methodology 18
2.2 Shema Dataset 19
2.3 File Format 20
2.4 Rename 21
2.5 Renamed 21
2.6 Verifed names 22
2.7 File Path 22
2.8 image basic components 23
2.9 RGB 24
2.10 RGB 24
2.11 Check image width and height 25
2.12 Check Polyp Positions 25
2.13 Check Polyp Count 26
2.14 Check Polyp Size 26
2.15 Image Hue, Brightness, and Saturation 27
2.16 Data Partitioning 28
2.17 Split 28
2.18 Data Augmentation 30
2.19 Data Augmentation 30
2.20 The U-Net Architecture 31
2.21 Model Architecture 33
2.22 Training Phase 33
2.23 Optimizers 34
2.24 Training 35
2.25 Evaluation 36
2.26 confusion matrix 38
2.27 Precision 38

2.28 Recall 39
2.29 Resume 39
2.30 Metrics 40
2.31 IoU 40
2.32 Dice 40
2.33 IoU 41
2.34 Precision Example 42
2.35 Mlops 43
2.36 Streamlit 44
2.37 App.py 45
2.38 Upload Image 45
2.39 About App 46
2.40 Visualisation 46

List of Tables

1.1 Description du tableau 15

2.1 Comparative Analysis of Different Segmentation Models 42

List of Abbreviations

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
DNNs	Deep Neural Networks
DL	Deep Learning
CNN	Convolutional Neural Network
AE	Autoencoder
GAN	Generative Adversarial Network
MMIF	Multimodal Medical Image Fusion
CT	Computed tomography
PET	Positron Emission Tomography
MRI	Magnetic resonance imaging
SPECT	Single Photon Emission Computed Tomography

Introduction

Colorectal cancer is a significant public health issue worldwide due to its high prevalence and substantial mortality rate. Ranking second among cancer types in terms of global mortality, colorectal cancer was responsible for over 1.9 million new cases and more than 930,000 deaths in 2020 alone. Projections indicate a worrying increase in the disease burden by 2040, with an estimated 3.2 million annual cases and 1.6 million deaths per year, marking a 63% and 73% increase, respectively.

This disease affects individuals of all ages, though it is more common among the elderly, and brings about not only physical health issues but also emotional, social, and economic challenges for patients and their families. The considerable economic impact of colorectal cancer underscores the need for inclusive and equitable measures, including investments in prevention, early detection, treatment, and research.

To address these challenges, our project focuses on developing a deep-learning model to predict colorectal cancer. By leveraging advanced artificial intelligence techniques, we aim to improve early detection and diagnosis, thus potentially reducing the mortality rate and improving patient outcomes. Our approach will involve training the model on comprehensive datasets, allowing for accurate predictions that can aid healthcare professionals in making timely and precise diagnoses.

Through this initiative, we seek to enhance the overall management and prognosis of colorectal cancer, ultimately contributing to better health outcomes and reduced disease burden for individuals and communities worldwide.

CHAPTER 1

General Context

1.1 Introduction

1.1.1 Project Context and Importance

In medicine, we really need better ways to diagnose diseases, especially colorectal cancer. Classical diagnostic methods depend heavily on subjective clinical judgment, resulting in delays in examination and initiation of treatment.

Our project aims to address this problem by using new technologies, such as medical image segmentation combined with smart algorithms that learn from data to develop predictive models that can help identify colorectal cancer early. Among the necessary steps in our model is the segmentation of medical images, which allows for precise analysis and accurate detection of cancerous tissues.

With this technology, we aim to make it easier and faster for doctors to recognize colorectal cancer at an early stage and start therapy promptly, which can make a significant difference for patients. Early detection is crucial for improving survival rates and patient outcomes, making treatment more effective, less invasive, and often less costly. This not only enhances the physical health of patients but also reduces the emotional and social burdens associated with late-stage diagnoses.

Additionally, early intervention provides significant economic benefits by lowering healthcare costs and better allocating medical resources, while also advancing research and preventive strategies to reduce the global burden of colorectal cancer and improve overall health outcomes.

1.1.2 Project Objectives

The purpose of this project is to develop a deep learning model for predicting colorectal cancer using medical image segmentation, specifically leveraging the U-Net architecture. The project objectives include:

1. Implementing and optimizing the U-Net architecture for medical image segmentation, tailored to accurately delineate colorectal tumors and differentiate between healthy and cancerous tissues.

2. Integrating deep learning algorithms, specifically CNN-based models, into the segmentation pipeline to analyze medical images and predict the presence of colorectal cancer.

3. Conducting rigorous evaluation and validation of the predictive model's performance and accuracy using diverse datasets, including annotated medical images and clinical data.

4. Calculating and reporting key metrics such as accuracy, sensitivity, specificity, and area under the curve (AUC) to assess the model's diagnostic capability.

5. Comparing the performance of our model with existing methods and algorithms in terms of accuracy, computational efficiency, and robustness.

6. Iteratively refining the model based on feedback from clinical experts and fine-tuning parameters to improve predictive performance and generalization.

By delineating these objectives, we aim to guide the development process and ensure that our project aligns with the overarching goal of advancing colorectal cancer diagnosis through innovative technological solutions.

1.2 Overview of colorectal cancer polyps

Colorectal polyp cancer is a progressive condition affecting the colon and rectum, both vital parts of the digestive system. Symptoms often develop gradually, with the initial signs possibly including subtle changes such as occasional rectal bleeding or changes in bowel habits. While some individuals may not experience noticeable symptoms in the early stages, others may notice abdominal discomfort, unexplained weight loss, or a feeling of incomplete bowel movements. As the condition progresses, symptoms may intensify, potentially leading to more severe complications such as bowel obstruction or anemia due to chronic bleeding. While colorectal polyp cancer cannot be cured, early detection through screening tests such as colonoscopies can significantly improve prognosis and treatment outcomes. Treatment options may include surgical removal of polyps or portions of the colon, chemotherapy, and targeted drug therapies. Additionally, lifestyle modifications such as a healthy diet, regular exercise, and avoiding tobacco and excessive alcohol consumption can help manage symptoms and improve overall well-being for individuals with colorectal polyp cancer.

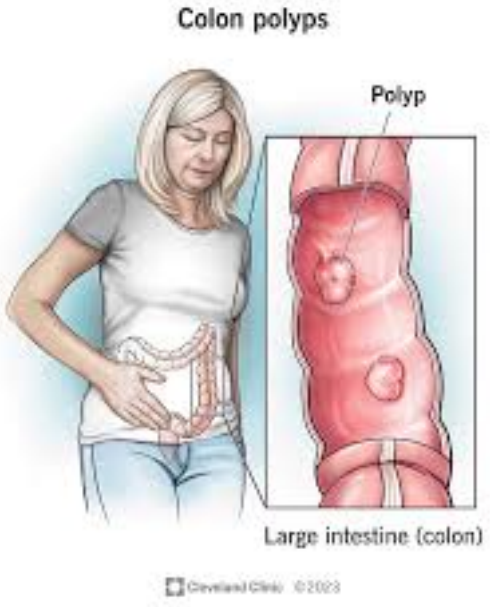

Figure 1.1: Colon polyps

1.2.1 Symptoms

Colorectal cancer polyps often do not cause any symptoms, especially in the early stages. However, as the polyps grow larger or become cancerous, they may cause various symptoms, including:

- Rectal bleeding or blood in the stool
- Changes in bowel habits, such as diarrhea or constipation
- Persistent abdominal discomfort or cramping
- Unexplained weight loss
- Fatigue or weakness
- Feeling that the bowel does not empty completely

It is important to note that these symptoms can also be caused by other conditions, so it is essential to consult a healthcare professional for proper evaluation and diagnosis.

1.2.2 Causes

The exact cause of colorectal cancer polyps is not fully understood, but several factors may increase the risk of developing them. These risk factors include:

- Age: The risk of colorectal cancer increases with age, with most cases diagnosed in individuals over 50 years old.
- Family history: Having a family history of colorectal cancer or polyps increases the risk of developing them.
- Personal history: Individuals who have previously had colorectal cancer or polyps are at higher risk of developing them again.
- Inflammatory bowel disease: Conditions such as ulcerative colitis or Crohn's disease can increase the risk of colorectal cancer.

- Lifestyle factors: Certain lifestyle choices, such as a diet high in red or processed meats, lack of physical activity, obesity, smoking, and heavy alcohol consumption, may also increase the risk.

While these factors may increase the likelihood of developing colorectal cancer polyps, not everyone with these risk factors will develop the condition, and individuals without any risk factors can still develop polyps.

1.2.3 Diagnosis

Diagnosing colorectal cancer polyps typically involves a combination of screening tests and diagnostic procedures. These may include:

- Colonoscopy: A procedure in which a flexible tube with a camera is inserted into the rectum to examine the colon and remove any polyps for biopsy.
- Flexible sigmoidoscopy: Similar to a colonoscopy but only examines the lower part of the colon.
- Stool tests: Tests such as fecal occult blood test (FOBT) or fecal immunochemical test (FIT) may detect blood in the stool, which can indicate the presence of polyps or cancer.
- Imaging tests: Imaging studies such as computed tomography (CT) scans, magnetic resonance imaging (MRI), or barium enema may be used to visualize the colon and detect polyps.

Early detection through screening is crucial for improving treatment outcomes and reducing the risk of complications associated with advanced colorectal cancer.

1.2.4 Treatment

The treatment of colorectal cancer polyps depends on various factors, including the size, location, and type of polyp, as well as the individual's overall health and preferences. Treatment options may include:

- **Polypectomy:** The removal of polyps during a colonoscopy. Small polyps can often be removed completely during the procedure. This method is minimally invasive and typically requires little recovery time, allowing patients to resume normal activities quickly.
- **Endoscopic Mucosal Resection (EMR):** A procedure to remove larger polyps or those that cannot be completely removed during a standard colonoscopy. EMR involves injecting a solution beneath the polyp to lift it from the underlying tissue, making it easier to remove. This technique can effectively treat polyps that are larger or have a higher risk of containing precancerous or cancerous cells.
- **Surgery:** In cases where polyps are large, cancerous, or cannot be removed by other methods, surgery to remove part of the colon (colectomy) may be necessary. Surgical options vary from minimally invasive laparoscopic procedures to more extensive open surgeries, depending on the polyp's characteristics and the patient's overall health. Surgery is often considered when there is a significant risk of cancer or when other treatments are not feasible.
- **Surveillance:** After the removal of polyps, regular follow-up exams and screenings may be recommended to monitor for the recurrence of polyps or the development of colorectal cancer. Surveillance typically includes periodic colonoscopies to detect new polyps early and to ensure that the colon remains clear of potentially precancerous growths. The frequency of these follow-up exams will depend on the initial findings and the patient's risk factors.

The choice of treatment will be determined by a healthcare professional based on the individual's specific circumstances and preferences, with the goal of effectively removing or managing polyps while minimizing the risk of complications. Treatment plans are personalized to balance efficacy, safety, and the patient's quality of life, ensuring the best possible outcome in managing colorectal cancer risk.

1.3 Concept and technique of segmentation

1.3.1 Segmentation of Medical Images

Segmentation of medical images involves partitioning an image into coherent regions or segments to identify specific structures or areas of interest within the image. This process is crucial in various medical applications, including disease diagnosis, treatment planning, and medical image analysis.

Importance of Segmentation: Accurate segmentation of medical images is essential for extracting meaningful information and facilitating quantitative analysis. By delineating anatomical structures, lesions, or abnormalities from surrounding tissues, segmentation enables clinicians to identify and characterize pathology more precisely.

Techniques and Terms:

- Encoder-Decoder Architecture:

  This architecture, commonly used in segmentation tasks, consists of an encoder network for feature extraction and a decoder network for spatial localization. The encoder compresses the input image into a latent space representation, while the decoder reconstructs the segmented image from this representation.

- Pixel-level Segmentation:

  Pixel-level segmentation, also known as semantic segmentation, assigns a class label to each pixel in the image, effectively partitioning the image into distinct regions based on semantic information. This fine-grained segmentation enables detailed analysis and interpretation of medical images.

- Class Imbalance:

  Class imbalance refers to unequal distribution of pixel classes in the dataset, where certain classes may be underrepresented compared to others. Addressing class imbalance is crucial for training robust segmentation models that accurately capture all classes of interest.

- Post-processing Techniques:

  Post-processing techniques, such as morphological operations (e.g., erosion, dilation) and conditional random fields (CRFs), are applied to refine segmentation results and improve their coherence and accuracy. These techniques help mitigate artifacts and inconsistencies in the segmented regions.

- Multi-scale and Pyramid Architectures:

  Multi-scale and pyramid architectures incorporate features from multiple resolutions to capture both local details and global context in the image. By aggregating information across different scales, these architectures enhance the segmentation performance, particularly for objects of varying sizes.

- Transfer Learning:

  Transfer learning involves leveraging pre-trained models on large-scale datasets for initial feature extraction, followed by fine-tuning on smaller medical image datasets. This approach accelerates model training and improves segmentation performance, especially in scenarios with limited annotated data.

Applications: Segmentation techniques are applied across various medical specialties, including radiology, oncology, neurology, and cardiology, for tasks such as tumor detection, organ segmentation, and disease quantification. These applications benefit from accurate segmentation results to support clinical decision-making and improve patient outcomes.

Challenges in Medical Image Segmentation:

Segmenting medical images poses several challenges due to factors such as image noise, variability in anatomical structures, and variations in imaging modalities. Additionally, the need for robust algorithms capable of handling diverse image characteristics and pathology types further complicates the segmentation process.

Techniques and Methods: Various techniques and methods have been developed for medical image segmentation, each with its strengths and weaknesses. These techniques range from traditional approaches to state-of-the-art deep learning methods, providing a spectrum of options for segmenting complex medical images.

- Traditional Methods:

  - Mathematical Models: Mathematical models, such as active contours (snakes) and level set methods, are widely used for boundary-based segmentation. These methods iteratively deform a contour to minimize an energy function, fitting it to the boundaries of anatomical structures.

  - Edge Detection:

  Edge detection algorithms, such as the Sobel or Canny edge detectors, identify edges or boundaries in images based on changes in intensity. These edges can then be used as cues for segmenting structures within the image.

  - Region-Based Methods:

  Region-based algorithms, including region growing and watershed segmentation, partition an image into regions based on similarities in intensity, texture, or other features. These methods effectively segment homogeneous regions but may struggle with noise and intensity variations.

- Deep Learning Methods:

  - Convolutional Neural Networks (CNNs):

  CNNs are powerful tools for medical image segmentation, leveraging multiple layers of convolutional and pooling operations to automatically learn hierarchical features from raw image data.

  - U-Net Architecture:

  The U-Net architecture, designed specifically for medical image segmentation, consists of a contracting path for context capture and a symmetric expanding path for precise localization.

  - Deep Segmentation Models:

  Deep segmentation models extend CNN architectures, incorporating additional components like skip connections, attention mechanisms, or recurrent layers, to improve segmentation accuracy and robustness.

1.3.2 Concepts of Deep Learning

1.3.2.1 Deep Learning

Deep learning (DL) , a subset of ML, is inspired by the information processing patterns found in the human brain. DL does not require any human-designed rules to operate; rather, it uses a large amount of data to map the given input to specific labels. It is designed using numerous layers of algorithms (artificial neural networks, or ANNs), each of which provides a different interpretation of the data that has been fed to them.

1.3.2.2 Convolutional Neural Networks (CNN)

In the field of DL, the CNN is the most famous and commonly employed algorithm [16]. The CNN architecture consists of a number of layers (or so-called multi-building blocks).Each layer in the CNN architecture, including its function, is brievely described below.

- Convolutional Layer:

  In CNN architecture, the most significant component is the convolutional layer. It consists of a collection of convolutional filters (also called kernels).

- The input Layer:

  The input, expressed as an N-dimensional matrix, is convolved with these filters to generate the output feature map.

- Pooling Layer:

  The main task of the pooling layer is the sub-sampling of the feature maps. These maps are generated by following the convolutional operations. In other words, this approach shrinks large-size feature maps to create smaller feature maps.

- Non-linear activation layers

  Are employed after all layers with weights (so-called learn- able layers, such as Fully connected and convolutional layers) in CNN architecture. This non-linear performance of the activation layers means that the mapping of input to output will be non-linear.

- Fully Connected Layer:

  Commonly, this layer is located at the end of each CNN architecture. Inside this layer, each neuron is connected to all neurons of the previous layer, the so-called Fully Connected (FC) approach. It is utilized as the CNN classifier.

- Output Layer:

  Achieves the final classification and represents the last layer of the CNN architecture.

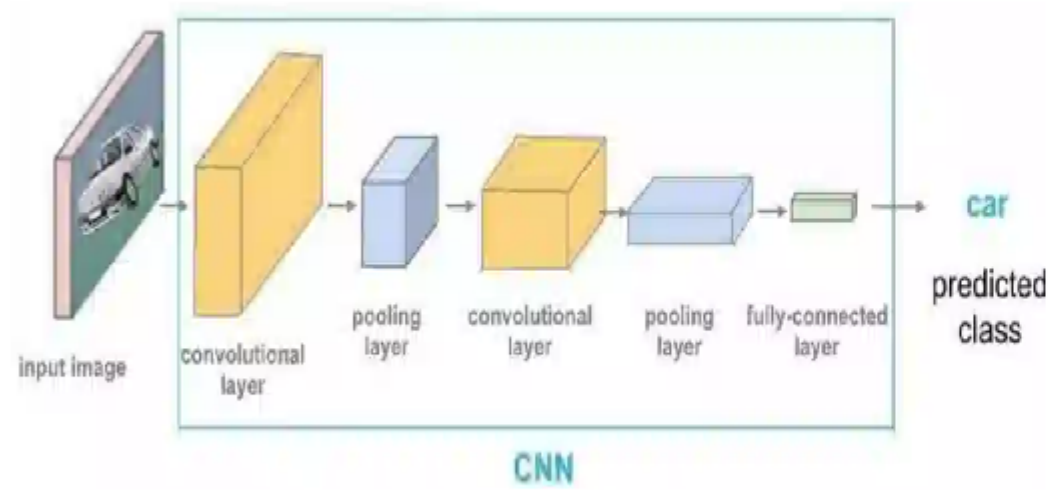

Figure 1.2: Convolutional Neural Networks (CNN)

1.4 State of the Art

1.4.1 General Description

Polyp segmentation is a critical task in medical image analysis, particularly for early detection and diagnosis of colorectal cancer. Over the years, various methodologies have been developed to improve the accuracy and efficiency of polyp segmentation. These methodologies

can be broadly categorized into three main types: CNN-based methods, Transformer-based methods, and UNet-based methods. Each of these categories leverages different architectural principles to achieve high performance in medical image segmentation tasks.

1.4.2 CNN-based Models

Convolutional Neural Networks (CNNs) have been widely used in image segmentation due to their powerful feature extraction capabilities. Here are some prominent CNN-based models:

- **PraNet**: This model is specifically designed for polyp segmentation. It utilizes a parallel partial decoder and reverse attention mechanisms to accurately segment polyps in colonoscopy images. PraNet effectively combines global context with local details, resulting in high segmentation accuracy.
- **DeepLabV3+**: This model combines atrous convolution with a UNet-like decoder to capture multi-scale context. DeepLabV3+ effectively balances the trade-off between localization and contextual understanding, making it suitable for various medical image segmentation tasks.

1.4.3 UNet and its Variants

UNet and its variants are among the most popular architectures for medical image segmentation due to their encoder-decoder structure and skip connections, which help retain spatial information. Notable UNet-based models include:

- **U-Net**: A classic encoder-decoder structure with symmetric skip connections that enable precise localization by combining high-level features with low-level details. U-Net is widely used for various medical image segmentation tasks.
- **ResUNet**: An extension of the original U-Net that incorporates residual connections. These connections help in alleviating the vanishing gradient problem and improve the training of deep networks, leading to better segmentation performance.

1.4.4 Transformer-based Models

Transformer-based models have gained popularity due to their ability to capture long-range dependencies and global context in images. Key transformer-based models include:

- **TransUNet**: A hybrid model that combines the strengths of transformers and UNet architectures. It uses a transformer encoder to capture global context and a UNet decoder to produce high-resolution segmentation maps. TransUNet is particularly effective for medical image segmentation tasks.
- **DA-TransUNet**: An advanced version of TransUNet that incorporates dual attention mechanisms. These mechanisms allow the model to focus on relevant features more effectively, enhancing the accuracy of segmentation by highlighting important regions in the image.

These approaches represent the state-of-the-art in medical image segmentation, with each category offering unique advantages for the task of polyp segmentation in datasets such as Kvasir-SEG.

Table 1.1: Description du tableau

Référence	Image Size	Model	Loss Function	Performance
CNN-based Models				
[Fan 2020]	[[From 332 × 487 to 1920 × 1072 pixels]	PraNet	[Cross-Entropy Loss and Dice Loss]	*IoU* ↑ 0.84 *Dice* ↑ 0.89
[Chen 2018]	[[From 332 × 487 to 1920 × 1072 pixels]	DeepLabv3+	[Focal Loss and Cross-Entropy Loss and Dice Loss]	*IoU* ↑ 0.71 *Dice* ↑ 0.82
UNet and its Variants				
[Ronneberger 2015]	[From 332 × 487 to 1920 × 1072 pixels]	U-Net	[Cross-Entropy and Pixel-wise Softmax]	*IoU* ↑ 0.38 *Dice* ↑ 0.54
[Zhang 2018]	[From 332 × 487 to 1920 × 1072 pixels]	ResUNet	[Binary Cross-Entropy Loss and Dice Loss]	*IoU* ↑ 0.35 *Dice* ↑ 0.44
Transformer-based Models				
[Chen 2021]	[from 332 × 487 to 1920 × 1072 pixels]	TransUNet	[Cross-Entropy Loss and Dice Loss]	*IoU* ↑ 0.8003 *Dice* ↑ 0.8791
[Sun 2024]	[from 332 × 487 to 1920 × 1072 pixels]	DA-TransUNet	[Cross-Entropy Loss and Dice Loss]	*IoU* 0.8102 ↑ *Dice* ↑ 0.8847

1.5 Conclusion

In this chapter, we presented the scope of our project by introducing the project context, the problem statement, the project goals,the medical image segmentation , Deep learning concepts. Besides, we provided a comprehensive overview of the general concepts within our field of inquiry laying the groundwork for our research journey. In the following chapter, we provide a full description of the data, as well as the pre-processing and processing steps ,the methodolgy , deployment .

CHAPTER 2

The proposed Method for Polyp segmentation

2.1 Introduction

2.2 CRISP methodology

The CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology is a structured and widely recognized approach for developing data mining projects. Conceived in the late 1990s, CRISP-DM has become the standard in the industry due to its flexibility and applicability across various sectors. The process comprises six interrelated phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. In the first phase, business objectives are defined, and the business problem is translated into a data mining problem. Then, in the data understanding phase, available data is collected and examined to identify quality issues and gain preliminary insights. Data preparation involves cleaning and transforming the data into a suitable format for modeling. During the modeling phase, various data mining techniques are selected and applied to build predictive models. Subsequently, in the evaluation phase, the models are reviewed and validated to ensure they meet business objectives. Finally, in the deployment phase, the models are implemented in the operational environment and monitored to ensure their continuous performance. The CRISP-DM methodology provides a detailed and adaptable framework, enabling data analysis teams to tackle complex projects systematically and efficiently.

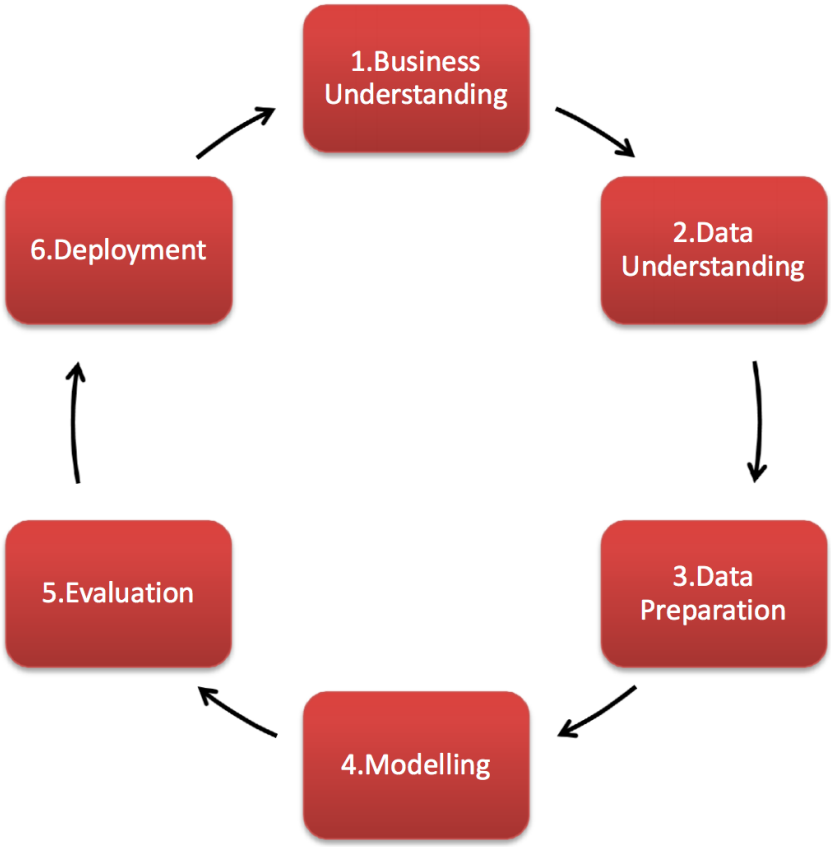

Figure 2.1: CRISP Methodology

2.3 Business Understanding

Segmentation in the context of gastrointestinal polyps detection is a pivotal stage in early disease diagnosis, particularly for conditions predisposing to colorectal cancer. Conventionally, healthcare professionals manually inspect endoscopic images, a laborious process prone to human error. Leveraging data science tools such as machine learning and computer vision, this process can be automated, enhancing speed and accuracy. Traditional methods like thresholding and edge detection, while once employed, lack the precision and efficiency afforded by contemporary machine learning models. Automating polyp segmentation streamlines the identification process, enabling timely intervention and management of potentially life-threatening conditions.

2.4 Data Understanding

The dataset utilized for this project originates from the Kvasir-SEG dataset, publicly accessible via Simula Datasets. It comprises gastrointestinal polyp images accompanied by corresponding segmentation masks in JPEG format. Additionally, the dataset includes bounding box information for each image, cataloged within a JSON file.

2.4.1 Source and Structure

The Kvasir-SEG dataset was assembled by medical experts during colonoscopy procedures, and specifically curated to facilitate research in computer-aided diagnosis. Its primary objective is to aid in the early identification and management of gastrointestinal polyps, which can potentially evolve into cancerous lesions.

2.4.2 Schema

Images: Stored as JPEG files, each image depicts a distinct instance of a gastrointestinal polyp. Segmentation Masks: These grayscale JPEG images serve as the ground truth for segmentation tasks, providing essential guidance for model training and evaluation. Bounding Boxes: The JSON file contains coordinates outlining the bounding boxes encircling the polyps within the images. Each entry corresponds to a specific image, delineating the precise coordinates defining the bounding box.

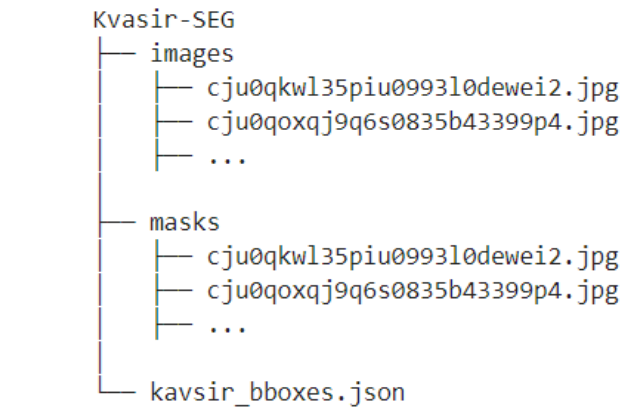

Figure 2.2: Shema Dataset

2.4.3 Limitations

The Kvasir-SEG dataset may present certain limitations, notably in terms of size variability and dataset size. Variability in image dimensions could pose challenges for standardization and model generalization. Moreover, the dataset size might be insufficient to fully capture the diversity of gastrointestinal polyp characteristics, potentially affecting model robustness and performance. Addressing these limitations requires careful preprocessing and augmentation strategies to mitigate size discrepancies and enhance dataset representativeness.

2.4.4 Data Preprocessing

2.4.4.1 Verify File Format and Quantity

1. Check the format of each file in the dataset to ensure consistency.
2. Count the number of files to verify the total quantity of data.

```python
# Define the path to the "Kvasir-SEG" directory and the "images" and "masks" subdirectories
data_dir = "./Kvasir-SEG"
images_dir = os.path.join(data_dir, "images")
masks_dir = os.path.join(data_dir, "masks")

# Get a list of filenames in the "images" and "masks" subdirectories
images_files = os.listdir(images_dir)
masks_files = os.listdir(masks_dir)

# Check the file format of each image file in the images folder
images_formats = set()
for file_name in os.listdir(images_dir):
    file_path = os.path.join(images_dir, file_name)
    if os.path.isfile(file_path):
        file_format = file_name.split(".")[-1]
        images_formats.add(file_format)

# Check the file format of each image file in the masks folder
masks_formats = set()
for file_name in os.listdir(masks_dir):
    file_path = os.path.join(masks_dir, file_name)
    if os.path.isfile(file_path):
        file_format = file_name.split(".")[-1]
        masks_formats.add(file_format)

# Print the number of images and masks, along with their file formats
print(f"There are {len(images_files)} colonoscopic images with {images_formats} format")
print(f"There are {len(masks_files)} segmentation masked images with {masks_formats} format")
```

```
There are 1000 colonoscopic images with {'jpg'} format
There are 1000 segmentation masked images with {'jpg'} format
```

Figure 2.3: File Format

2.4.4.2 Rename Files

1. Define a renaming scheme, such as adding a prefix or suffix to the filenames.
2. Iterate through each file and rename it according to the defined scheme.

```python
# Define a dictionary to map the original filenames to new filenames
name_mapping = {"images": "colonoscopy_", "masks": "mask_"}

# Rename the files in the "images" subdirectory
for i, file_name in enumerate(images_files):
    old_file_path = os.path.join(images_dir, file_name)
    new_file_name = f"{name_mapping['images']}{i+1:04d}.jpg"
    new_file_path = os.path.join(images_dir, new_file_name)
    os.rename(old_file_path, new_file_path)

# Rename the files in the "masks" subdirectory
for i, file_name in enumerate(masks_files):
    old_file_path = os.path.join(masks_dir, file_name)
    new_file_name = f"{name_mapping['masks']}{i+1:04d}.jpg"
    new_file_path = os.path.join(masks_dir, new_file_name)
    os.rename(old_file_path, new_file_path)
```

Figure 2.4: Rename

```python
# Renamed image file name
images_files = os.listdir(images_dir)
images_files[:5]
```

```
['colonoscopy_0001.jpg',
 'colonoscopy_0002.jpg',
 'colonoscopy_0003.jpg',
 'colonoscopy_0004.jpg',
 'colonoscopy_0005.jpg']
```

Here's the renamed masked images

```python
# Renamed image file name
masks_files = os.listdir(masks_dir)
masks_files[:5]
```

```
['mask_0001.jpg',
 'mask_0002.jpg',
 'mask_0003.jpg',
 'mask_0004.jpg',
 'mask_0005.jpg']
```

Figure 2.5: Renamed

2.4.4.3 Create a New JSON File with Renamed File Names

1. Create a new JSON file to store the mapping between original filenames and renamed filenames.
2. Populate the JSON file with the appropriate key-value pairs representing the mapping.

2.4.4.4 Remove Duplicated Files

1. Identify and remove any duplicated files within the dataset to avoid redundancy.
2. Utilize file hashing or comparison methods to identify duplicates.

2.4.4.5 Verify if Names are Changed

1. Cross-reference the original dataset with the newly renamed files to ensure all files were successfully renamed.
2. Compare the contents of the JSON file with the actual filenames to validate the mapping.

```
# Define the path to the "renamed_bboxes.jso" file
data_dir = "./Kvasir-SEG"
file_path = os.path.join(data_dir, "renamed_bboxes.json")

# Read the JSON file
with open(file_path, 'r') as json_file:
    json_object = json.load(json_file)

# Print first 5 serials
list(json_object.keys())[:5]
```

```
['colonoscopy_0001',
 'colonoscopy_0002',
 'colonoscopy_0003',
 'colonoscopy_0004',
 'colonoscopy_0005']
```

Figure 2.6: Verifed names

2.4.4.6 Set Path to Accessing the Data

1. Define the directory structure for organizing the dataset.

2. Set the file paths or directory paths required to access the data within the preprocessing pipeline.

```
# Define the path to the "Kvasir-SEG" directory, "images" and "masks" subdirectories, and the JSON file
data_dir = "./Kvasir-SEG"
images_dir = os.path.join(data_dir, "images")
masks_dir = os.path.join(data_dir, "masks")
images_json= os.path.join(data_dir, "renamed_bboxes.json")

# Get a list of filenames in the "images" and "masks" subdirectories
images_files = os.listdir(images_dir)
masks_files = os.listdir(masks_dir)

# Read the JSON file
with open(images_json, 'r') as json_file:
    kavsir_bboxes = json.load(json_file)
```

Let's check again how many images we have

```
# Print the number of images and masks
print(f"There are {len(images_files)} colonoscopic images")
print(f"There are {len(masks_files)} segmentation masked images")
print(f"The JSON file contains the mask coordinates for the {len(list(kavsir_bboxes))} colonoscopic images ")
```

```
There are 1000 colonoscopic images
There are 1000 segmentation masked images
The JSON file contains the mask coordinates for the 1000 colonoscopic images
```

Figure 2.7: File Path

2.4.5 Data Exploration Image

"EDA" stands for "Exploratory Data Analysis". It's an essential approach in data analysis where the analyst explores and examines the available data to understand its structure, identify trends, patterns, anomalies, and relationships between variables.

2.4.6 Check image basic components

- An image is essentially a matrix (or multiple matrices) of numbers representing the intensity of each pixel per channel. The `dtype` of the image is `uint8`. The RGB image may have some pixels with the value (0, 0, 0), indicating a black border.

```
# Read first image from RGB colonoscopic images
cv2.imread(os.path.join(images_dir,images_files[0]))
```

```
array([[[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
```

Figure 2.8: image basic components

- When read by OpenCV, the channels are in the order BGR. Converting to RGB yields the correct color representation.
- After converting to RGB, the image has a height of 529, width of 622, and 3 channels (RGB). The mask image also has 3 channels, resembling a grayscale image due to identical values for all channels in each pixel.
- Both the RGB image and the mask have pixel values ranging from 0 to 255, with 0 representing black and 255 representing white. The `uint8` `dtype` ensures that the pixel intensity values range from 0 to 255 without negative values.

```
# Read image (BGR by default)
BGR_img1 = cv2.imread(os.path.join(images_dir,images_files[0]))

# Convert BGR to RGB
RGB_img1 = cv2.cvtColor(BGR_img1, cv2.COLOR_BGR2RGB)

# Display image
plt.imshow(RGB_img1)
plt.show()
```

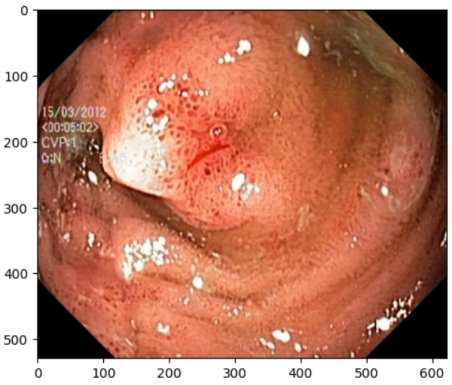

Figure 2.9: RGB

```
#Find out the min and max values of pixels in the RGB image
print('RGB Image:')
print('Image minimum value: ',RGB_img1.min())
print('Image maximum value: ',RGB_img1.max())

#Find out the min and max values of pixels in the mask image
print('Mask Image:')
print('Image minimum value: ',mask_img1.min())
print('Image maximum value: ',mask_img1.max())
```

```
RGB Image:
Image minimum value:  0
Image maximum value:  255
Mask Image:
Image minimum value:  0
Image maximum value:  255
```

Figure 2.10: RGB

2.4.7 Check image width and height

- It's crucial to verify if the image and its corresponding mask have the same size. The extracted widths and heights of all images reveal varying sizes, ranging from 332x487 to 1920x1072 pixels. The width and height distributions are right-skewed, with some extreme outliers.

- Outliers are identified using the interquartile range (IQR) to calculate the lower and upper bounds. Resizing all images to a standardized size is necessary for model compatibility. Choosing sizes within the boundary range can be an option.

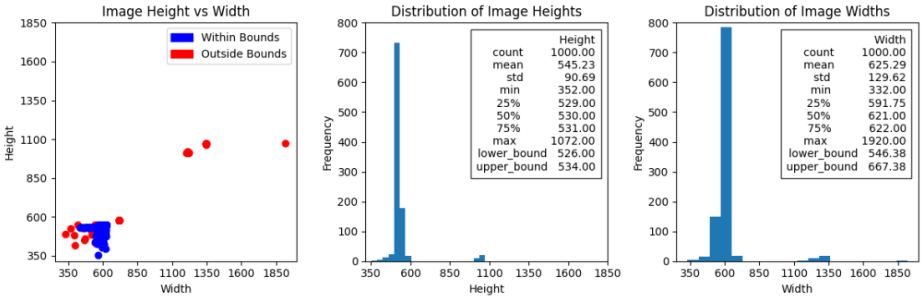

Figure 2.11: Check image width and height

2.4.8 Check Polyp Positions

To understand the spatial distribution of polyps within the images, each image gets divided into a 3x3 grid. Iterating over the bounding box data allows for the calculation of the center coordinates of each bounding box, determining its corresponding grid cell. The analysis reveals a predominant occurrence of polyps in the center of the images, specifically in the top-center (Grid2), center-center (Grid5), and bottom-center (Grid8) regions. Conversely, the left side of the images—comprising the top-left (Grid1), center-left (Grid4), and bottom-left (Grid7) regions—shows the fewest occurrences of polyps. This insight proves valuable for targeted image analysis and influences the design and evaluation of the segmentation model.

Figure 2.12: Check Polyp Positions

2.4.9 Check Polyp Size and Count

To quantify the relative size of the polyps, the area of each bounding box is divided by the total area of the corresponding image. The area of a bounding box is calculated by multiplying its width and height. The analysis shows that the majority of the polyps occupy between 7% to 30% of the total image area. Instances where polyps occupy more than 50% of the image are rare. Interestingly, the dataset contains 1,071 polyps across 1,000 images, indicating that some images feature multiple polyps. This observation is crucial for understanding the complexity of the segmentation task and informs the model's design and evaluation criteria.

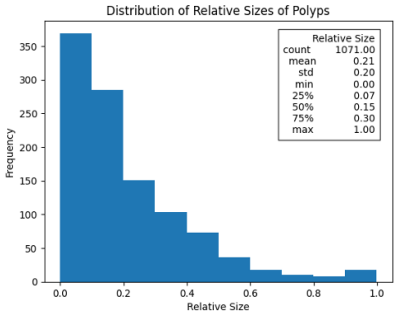

Figure 2.13: Check Polyp Count

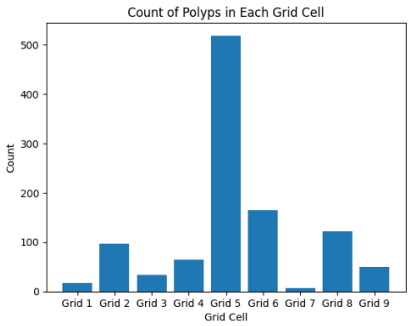

Figure 2.14: Check Polyp Size

2.4.10 Check Image Hue, Brightness, and Saturation

The color distribution of the polyps was analyzed using the HSV (Hue, Saturation, Value) color space, and the following observations were made:

- **Hue:** The hue values range from 4.28 to 48.53, with most values falling within the 25th percentile of 7.82 and the 75th percentile of 9.71. This range predominantly corresponds to the red color spectrum. A longer tail towards higher hue values indicates a lack of color bias in that direction.
- **Saturation:** Saturation values span a wide range from 75.41 to 206.39, with a mean value of 134.14. The distribution appears fairly symmetric, suggesting a moderate level of saturation on average across the dataset.
- **Brightness:** Brightness values range from 48.13 to 151.25, with a mean value of 97.48.

The distribution is fairly symmetric, indicating no strong bias towards either high or low brightness levels.

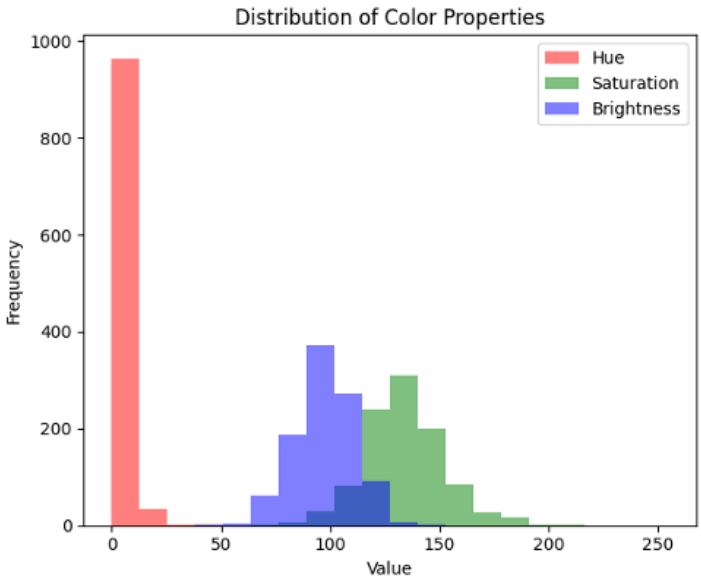

Figure 2.15: Image Hue, Brightness, and Saturation

2.5 Data Partitioning

We can see that our entire dataset consists of 1000 images and 1000 masks stored in two separate folders. Before fitting the images to our machine learning model, we will split the images and masks into training, validation, and test sets, then apply image preprocessing and augmentation.

The purpose of a train-test split is to assess the performance and generalization ability of a machine learning model. When building a model, it is crucial to evaluate its performance on data that it has not seen during the training phase. This helps determine if the model can effectively generalize its learnings to new, unseen data.

2.5.1 Set Up Train-Test Split Path

Before we go forward and generate the three data sets, we will first set up the train-test split pipeline by defining the split ratio and the path to save the three datasets. Below is an example of our targeted directory root tree:

```
new_data
├── test
│   ├── images
│   │   ├── 0011.jpg
│   │   ├── 0024.jpg
│   │   ├── ...
│   ├── masks
│   │   ├── 0011.jpg
│   │   ├── 0024.jpg
│   │   ├── ...
│
├── train
│   ├── images
│   │   ├── 0001.jpg
│   │   ├── 0002.jpg
│   │   ├── ...
│   ├── masks
│   │   ├── 0001.jpg
│   │   ├── 0002.jpg
│   │   ├── ...
│
├── valid
│   ├── images
│   │   ├── 0008.jpg
│   │   ├── 0009.jpg
│   │   ├── ...
│   ├── masks
│   │   ├── 0008.jpg
│   │   ├── 0009.jpg
│   │   ├── ...
```

Figure 2.16: Data Partitioning

- 10% of the images will be used for validation.
- 10% of the images will be used for testing.

```python
# Create the directories for the training, validation, and testing sets
if not os.path.exists(train_path):
    for path in [train_path, valid_path, test_path]:
        os.mkdir(path)
        os.mkdir(os.path.join(path, "images"))
        os.mkdir(os.path.join(path, "masks"))

# Calculate the sizes of the training, validation, and testing sets
len_ids = len(images)
train_size = int((80/100) * len_ids)
valid_size = int((10/100) * len_ids)  # 10% of the images used for validation
test_size = int((10/100) * len_ids)  # 10% of the images used for testing

# Split the images and masks into training and testing sets
train_images, test_images = train_test_split(images, test_size=test_size, random_state=42)
train_masks, test_masks = train_test_split(masks, test_size=test_size, random_state=42)

# Further split the training set into training and validation sets
train_images, valid_images = train_test_split(train_images, test_size=test_size, random_state=42)
train_masks, valid_masks = train_test_split(train_masks, test_size=test_size, random_state=42)

# Print the sizes of the dataset subsets
print("Total Size: ", len_ids)
print("Training Size: ", train_size)
print("Validation Size: ", valid_size)
print("Testing Size: ", test_size)
```

```
Total Size:  1000
Training Size:  800
Validation Size:  100
Testing Size:  100
```

Figure 2.17: Split

2.5.2 Data Augmentation

In order to enhance the diversity and robustness of our training dataset, we apply various data augmentation techniques. Below are the parameters used for data augmentation:

- **Rotation:** Rotating the image by a random angle within a specified range helps the model to generalize better by seeing the object from different orientations.

- **Translation:** Shifting the image along the X and Y axes by a certain percentage of the image size. This helps the model to recognize the object even when it is not centered.

- **Flipping:** Horizontally flipping the image to simulate the appearance of the object from different perspectives. This is particularly useful when the object can appear in any orientation.

- **Scaling:** Scaling the image by a factor within a specified range to help the model learn to recognize objects of different sizes.

- **Shearing:** Applying a shear transformation to the image which distorts the image along the X or Y axis. This can help the model become invariant to such distortions.

- **Brightness Adjustment:** Randomly changing the brightness of the image to make the model robust to images taken under different lighting conditions.

- **Zooming:** Randomly zooming into the image to simulate close-up views of the object. This helps the model to learn features at different scales.

```
if seed is not None:
    torch.manual_seed(seed)
    np.random.seed(seed)

rescale_factor = 1.0 / 255.0

if train_augmentation:
    train_transform = transforms.Compose([
        transforms.RandomHorizontalFlip(),
        transforms.RandomVerticalFlip(),
        transforms.RandomRotation(20),
        transforms.RandomResizedCrop(target_size),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
else:
    train_transform = transforms.Compose([
        transforms.Resize(target_size),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])

val_test_transform = transforms.Compose([
    transforms.Resize(target_size),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

train_dataset = SegmentationDataset(
```

Figure 2.18: Data Augmentation

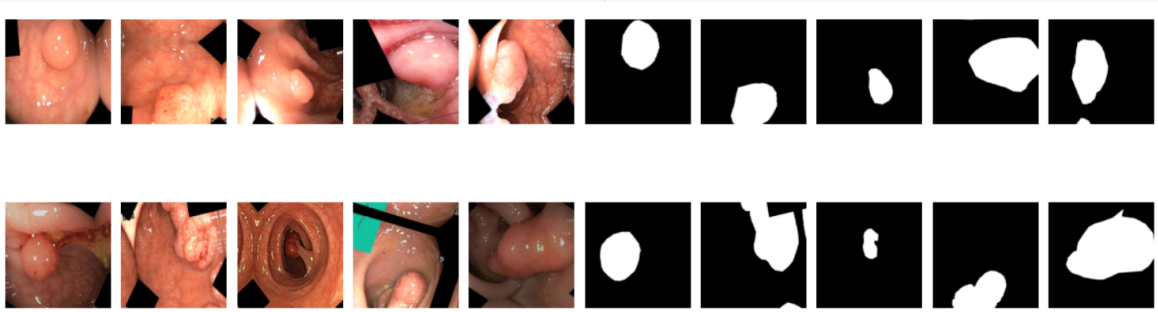

Figure 2.19: Data Augmentation

2.6 Modeling

2.6.1 Model Choice

2.6.2 The U-Net Architecture

The U-Net architecture is a type of convolutional neural network (CNN) designed for fast and precise segmentation of biomedical images. Originally, it was developed for biomedical image segmentation tasks such as neuronal structures in electron microscopic stacks. However, its versatility and efficacy have led to its adoption in a wide range of image segmentation tasks beyond biomedical applications. Some of its benefits are:

- **Data Efficiency**: U-Net is known for being very data-efficient, meaning that it can produce good results even with a limited amount of training data.
- **High Resolution**: The architecture is designed to produce high-resolution output, making it ideal for tasks that require identifying fine details in an image.
- **Versatility**: Although it was initially developed for biomedical image segmentation, U-Net has been successfully adapted for various other types of image segmentation tasks.

Many subsequent architectures have either built upon or been inspired by the U-Net for various segmentation and even some non-segmentation tasks.

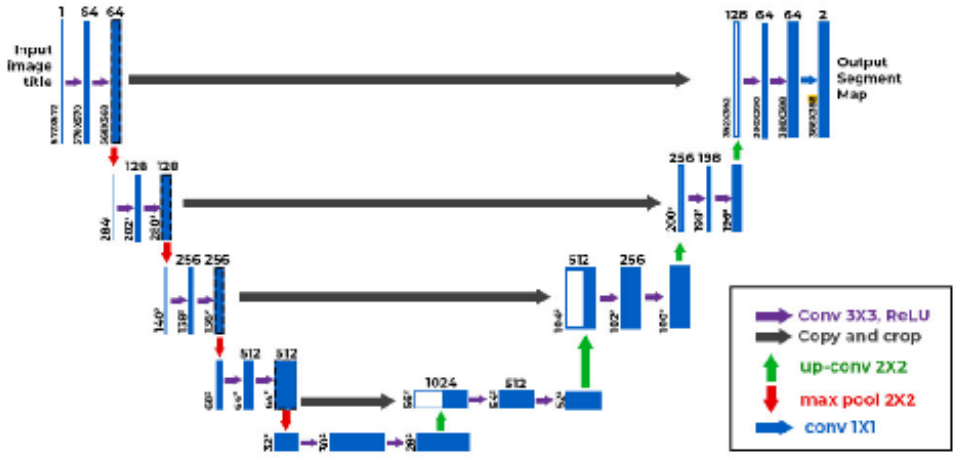

Figure 2.20: The U-Net Architecture

2.6.2.1 Model Architecture

The U-Net architecture can be visualized as a "U" shape, which gives it its name. Here's a breakdown of its structure:

- **Encoder (Downsampling Path)**: The encoder captures the context in the image. It consists of a series of convolutional layers, batch normalization layers, activation functions (usually ReLU), and max-pooling layers. Each step down the U consists of two convolutions followed by a max-pooling operation to reduce the dimensions of the feature maps. This helps the network learn increasingly abstract features.
- **Decoder (Upsampling Path)**: The decoder enables precise localization using transposed convolutions. It consists of a series of up-convolutional layers (or transposed convolutions), concatenation with feature maps from the encoder (skip connections), followed by regular convolutions. The transposed convolutions increase the dimensions of the feature maps.
- **Skip Connections**: These are the "bridges" between the encoder and decoder. They help the decoder recover the spatial information lost during encoding, which is crucial for achieving precise segmentation.
- **Bottleneck**: This is the deepest layer in the U-Net, connecting the encoder and decoder. It usually consists of convolutions and activation functions but no pooling. This layer is responsible for the most abstract feature representations.
- **Output Layer**: The final layer is a 1x1 convolution followed by an activation function like sigmoid (for binary segmentation) or softmax (for multi-class segmentation). In our implementation, we have added Batch Normalization layers to stabilize the training. We have also experimented with adding the Dropout layer, and trying with different L2 regularizer strength (1e-6 to 1e-3) and different initializer (He normal and Glorot uniform - default).

The best combination that gives us the best model performance for our task is to have no Dropout layer, default initializer, and L2 regularizer with L2 regularization factor of 1e-5.

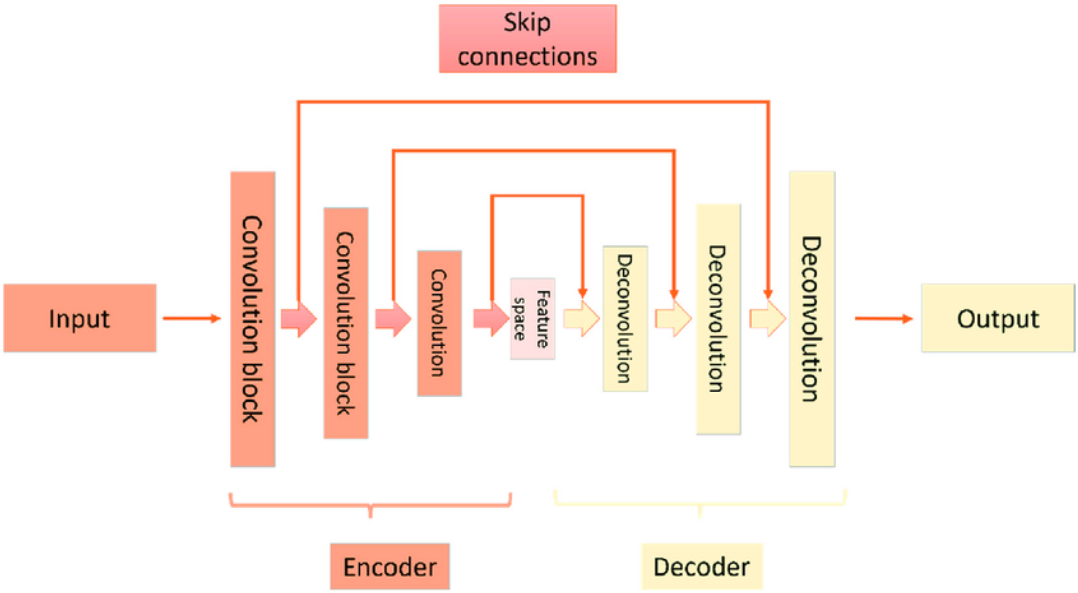

Figure 2.21: Model Architecture

2.6.3 Training Phase

2.6.3.1 Model Configuration

In this section, we will configure our U-Net model for training. Before we delve into the configuration process, it's crucial to set the parameters that will define our U-Net architecture. These parameters lay the foundation for building an effective model.

To compile the model successfully, we must carefully select the optimizer, loss function, and metrics. Among the metrics available, we opt for the widely used Accuracy metric, which assesses the frequency of correct predictions matching the labels.

```
[ ]  # Obtenir un batch d'images à partir du DataLoader d'entraînement
     batch_images, batch_masks = next(iter(train_loader))

     # Obtenir la forme des images d'entrée
     IMG_HEIGHT = batch_images.shape[2]
     IMG_WIDTH = batch_images.shape[3]
     IMG_CHANNELS = batch_images.shape[1]
     input_shape = (IMG_CHANNELS, IMG_HEIGHT, IMG_WIDTH)

     # Nombre de classes (segmentation binaire)
     n_classes = 1
```

Figure 2.22: Training Phase

Throughout our experimentation phase, we explore various combinations of optimizers, including Adam and Nadam, along with different loss functions such as Binary Crossentropy, Dice Loss, and IoU Loss. This rigorous exploration is essential for understanding how each component influences model performance.

After extensive experimentation, we identify the optimal configuration that yields the best model performance for our specific task. Remarkably, the combination of Adam optimizer with Binary Crossentropy loss function emerges as the most effective choice, showcasing superior convergence and accuracy.

Now, let's delve deeper into each parameter

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3)]	0	[]
conv2d (Conv2D)	(None, 256, 256, 64)	1792	['input_1[0][0]']
batch_normalization (BatchNorm alization)	(None, 256, 256, 64)	256	['conv2d[0][0]']
activation (Activation)	(None, 256, 256, 64)	0	['batch_normalization[0][0]']
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36928	['activation[0][0]']
batch_normalization_1 (BatchNo rmalization)	(None, 256, 256, 64)	256	['conv2d_1[0][0]']
activation_1 (Activation)	(None, 256, 256, 64)	0	['batch_normalization_1[0][0]']
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	['activation_1[0][0]']
conv2d_2 (Conv2D)	(None, 128, 128, 12 8)	73856	['max_pooling2d[0][0]']

Figure 2.23: Optimizers

2.6.3.2 Model Training

Once the model has been defined and compiled, we are now ready to train the model. Below are some key points to set up the training correctly:

- **Epochs and Steps**: We've set the number of epochs to 120 and calculated the steps per epoch based on the training data.

- **Model Checkpoint**: We use the *torch.save* function to save the best model based on validation loss, which allows us to resume training or make predictions later.
- **Early Stopping**: We implement early stopping by monitoring the validation loss and stopping training if there is no improvement for 10 epochs, which helps prevent overfitting and reduce training time.
- **Learning Rate Reduction**: We use the *ReduceLROnPlateau* scheduler from PyTorch's *torch.optim.lr_scheduler* to reduce the learning rate if the validation loss plateaus, helping the model converge to a better solution.
- **Training**: We use a standard training loop in PyTorch with our training and validation data loaders. The loop involves a forward pass, loss calculation, backward pass, and optimizer step.
- **Callbacks**: We've included multiple callbacks, such as *ModelCheckpoint* to save the best model, *EarlyStopping* to halt training early if performance stops improving, and *ReduceLROnPlateau* to adjust the learning rate.

```
        print(f"Validation Loss: {val_loss:.4f}, Validation Accuracy: {val_accuracy:.4f}")

        # Early stopping
        if val_loss < best_val_loss:
            best_val_loss = val_loss
            early_stop_counter = 0
        else:
            early_stop_counter += 1

        if early_stop_counter >= patience:
            print("Early stopping!")
            break

    # Save the trained model
    torch.save(model.state_dict(), 'unet.pth')
```

```
Epoch 1/120
50/50 [==============================] - ETA: 0s - loss: 0.5864 - accuracy: 0.7323
Epoch 1: val_loss improved from inf to 0.71582, saving model to ./model/unet_checkpointv2b_7a1.h5
50/50 [==============================] - 415s 8s/step - loss: 0.5864 - accuracy: 0.7323 - val_loss: 0.7158 - val_accuracy: 0.8163 - lr: 1.0000e-04
Epoch 2/120
50/50 [==============================] - ETA: 0s - loss: 0.4382 - accuracy: 0.8384
Epoch 2: val_loss improved from 0.71582 to 0.61529, saving model to ./model/unet_checkpointv2b_7a1.h5
50/50 [==============================] - 23s 467ms/step - loss: 0.4382 - accuracy: 0.8384 - val_loss: 0.6153 - val_accuracy: 0.8249 - lr: 1.0000e-04
Epoch 3/120
50/50 [==============================] - ETA: 0s - loss: 0.4147 - accuracy: 0.8340
Epoch 3: val_loss improved from 0.61529 to 0.60653, saving model to ./model/unet_checkpointv2b_7a1.h5
50/50 [==============================] - 23s 467ms/step - loss: 0.4147 - accuracy: 0.8340 - val_loss: 0.6065 - val_accuracy: 0.8023 - lr: 1.0000e-04
Epoch 4/120
50/50 [==============================] - ETA: 0s - loss: 0.3977 - accuracy: 0.8305
Epoch 4: val_loss improved from 0.60653 to 0.55674, saving model to ./model/unet_checkpointv2b_7a1.h5
```

Figure 2.24: Training

2.7 Evaluation

2.7.1 Quantitative Evaluation

When the training is completed, the `fit` method will return a `History` object, which is a record of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values.

Let's now plot them out and see how the model is performing. Based on the training history, we can conclude that:

- **Validation Loss**: The validation loss is generally decreasing, which is a sign that the model is learning to generalize well on unseen data. The lowest validation loss is 0.15967 at Epoch 89.
- **Training Accuracy**: The training accuracy is fluctuating around 88-90%, which suggests that the model is performing reasonably well on the training data.
- **Validation Accuracy**: The validation accuracy is generally higher than the training accuracy, often exceeding 94%. This is a positive sign, indicating that the model is not overfitting and is generalizing well to new data.

Overall, the model appears to be performing well. The validation metrics are strong, and there's no immediate sign of overfitting. However, in image segmentation, accuracy might not always be the most informative metric for evaluating model performance because the classes are usually imbalanced.

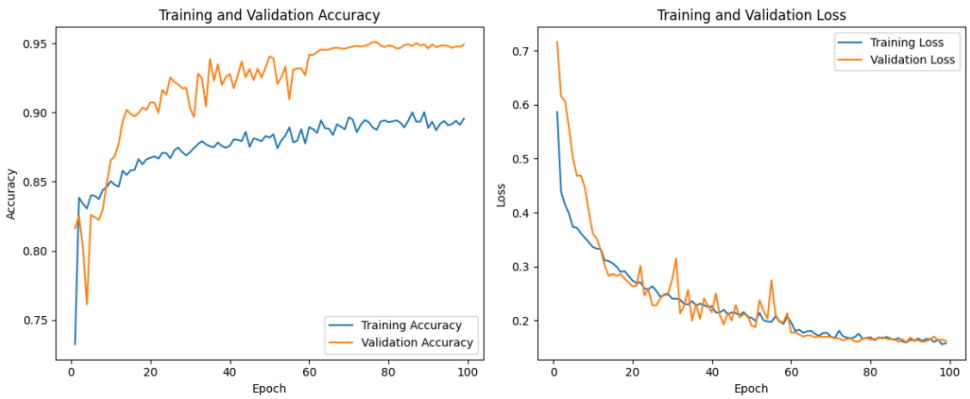

Figure 2.25: Evaluation

For example, the object of interest (e.g., the polyp) might occupy a very small portion of the image. A model that predicts every pixel as the majority class (background) could still achieve high accuracy but would perform poorly on predicting the minority class (object of interest).

Therefore, in the next section, we will look into other metrics that are more suitable for evaluating the segmentation performance.

2.7.1.1 Evaluate Trained U-Net Model

Other than accuracy, the other metrics that are often used in evaluating segmentation performance are:

- **Precision**
- **Recall**
- **IoU (Intersection over Union)**
- **Dice Coefficient**

The metric value ranges of all presented metrics span from 0 (worst) to 1 (best), and we will talk about each metric in detail in the upcoming sections. Now let's first load our trained model.

Confusion Matrix All presented metrics above are based on the computation of a confusion matrix, which represents the number of pixels that were correctly or incorrectly classified as either the object of interest or the background.

From the confusion matrix, we can observe that:

- **True Positive (TP)**: 879,742 pixels were correctly identified as the object of interest.
- **True Negative (TN)**: 5,361,958 pixels were correctly identified as the background.
- **False Positive (FP)**: 89,861 pixels were incorrectly identified as the object of interest.
- **False Negative (FN)**: 222,039 pixels were incorrectly identified as the background.

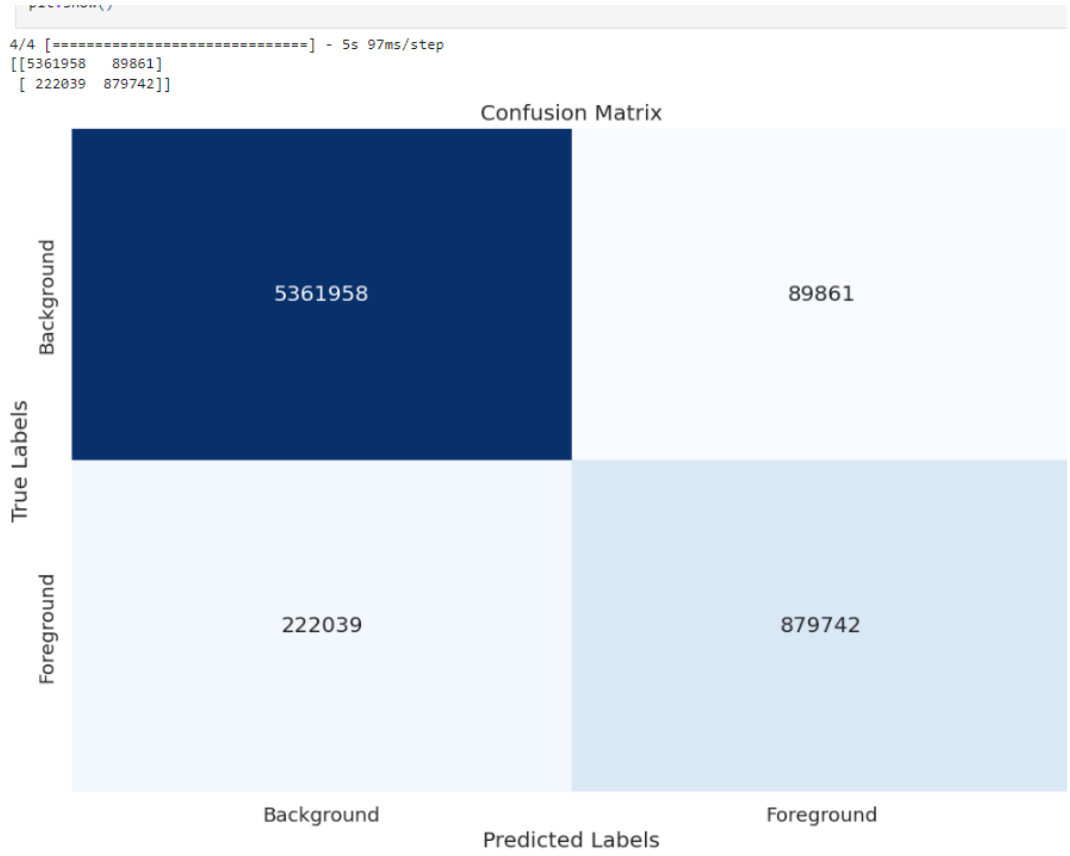

Figure 2.26: confusion matrix

Precision and Recall

- **Precision**: Also known as the Positive Predictive Value (PPV), precision measures the proportion of true positive predictions among all the positive predictions made by the model. In other words, it answers the question: "Of all the instances that the model labeled as positive, how many are actually positive?" Precision is more concerned with the purity of the positive predictions, aiming to minimize false positives.

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}}$$

Figure 2.27: Precision

- **Recall**: Also known as Sensitivity or True Positive Rate, recall measures the proportion of true positives among all the actual positives. It answers the question: "Of all the actual positive instances, how many did the model correctly identify?" Recall is more concerned with capturing all the possible positive instances, aiming to minimize false negatives.

$$\text{Recall} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}$$

Figure 2.28: Recall

As discussed, we can use the values from the confusion matrix (TP, TN, FP, and FN) to calculate evaluation metrics such as precision, recall, IoU, and Dice Coefficient, which provide a comprehensive assessment of the model's segmentation performance.

The model is more conservative in predicting the foreground class, as indicated by the high precision but lower recall. This means that the model is missing some pixels that should be classified as foreground. The model performs exceptionally well in classifying the background, as indicated by the high scores across all metrics. In summary, the model performs fairly well but could benefit from techniques aimed at improving the recall for the foreground class.

```
class_names = ["Background", "Foreground"]

# Generate the classification report
report = classification_report(test_mask_labels, binary_prediction_labels, target_names=class_names)

print(report)
```

```
              precision    recall  f1-score   support

  Background       0.96      0.98      0.97   5451819
  Foreground       0.91      0.80      0.85   1101781

    accuracy                           0.95   6553600
   macro avg       0.93      0.89      0.91   6553600
weighted avg       0.95      0.95      0.95   6553600
```

Figure 2.29: Resume

IoU and Dice Coefficient (F1 Score) F-measure, also called F-score, is one of the most widespread scores for performance measuring in Medical Image Segmentation (MIS). Based on the F-measure, there are two popular metrics used in MIS:

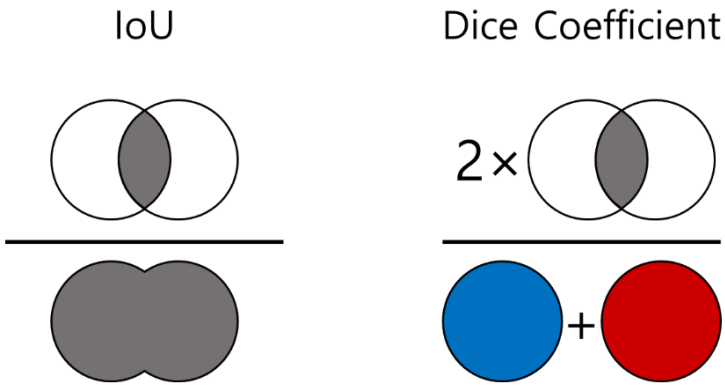

Figure 2.30: Metrics

- **Intersection over Union (IoU)**: Also known as Jaccard index, it is the area of the intersection over the union of the predicted segmentation and the ground truth.

$$IoU = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Figure 2.31: IoU

- **Dice Coefficient**: Also known as F1-score, it is the harmonic mean of precision and recall. Dice Coefficient is 2 * the Area of Overlap divided by the total number of pixels in both images.

$$Dice = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Figure 2.32: Dice

IoU of 0.738 is reasonably good, indicating that a significant portion of the ground truth overlaps with the predicted segmentation. A Dice Coefficient of 0.857 is also strong, suggesting a high degree of overlap between the predicted and ground truth segments. Dice Coefficients close to 1 are indicative of excellent model performance.

```
            union = torch.sum(masks) + torch.sum(binary_predictions) - intersection

            # Compute Dice coefficient for the batch
            dice_numerator, dice_denominator = dice_coefficient(masks, binary_predictions)

            # Update the totals
            total_intersection += intersection.item()
            total_union += union.item()
            total_dice_numerator += dice_numerator.item()
            total_dice_denominator += dice_denominator.item()

    # Compute the mean IoU and Dice coefficient across all batches
    mean_iou = total_intersection / total_union
    mean_dice = total_dice_numerator / total_dice_denominator

    # Print the mean IoU and Dice coefficient
    print(f"Mean IoU: {mean_iou:.4f}")
    print(f"Mean Dice Coefficient: {mean_dice:.4f}")
```

```
Mean IoU: 0.7382603164373193
Mean Dice Coefficient: 0.8566933243478139
```

Figure 2.33: IoU

2.7.2 Qualitative Evaluation

We have investigated 4 evaluation metrics: precision, recall, IoU, and Dice Coefficient to understand how well our model is segmenting the polyps. Now, we can perform a qualitative assessment (visual inspections) to get a comprehensive view of our model's performance.

The predicted segmentation mask exhibits a high degree of morphological similarity to the ground truth, capturing the overall shape and contours of the target region. However, there are subtle discrepancies in the boundary delineations, indicating that the model's spatial accuracy could be further optimized. Instead of using the standard U-Net, we can try other models that were specifically designed to improve boundary delineation in segmentation tasks, such as V-Net, DeepLab, Mask R-CNN, and many more.

- **Model Prediction**

We can also observe the presence of spurious detections or false positives in the predicted mask, which has a detrimental effect on the model's precision. These inaccuracies indicate that

```
    plt.subplot(5, 3, i*3 + 1)
    plt.imshow(test_image[random_index].cpu().permute(1, 2, 0))  # Convert from tensor to NumPy array and rearrange dimensions
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(5, 3, i*3 + 2)
    plt.imshow(test_mask[random_index].cpu().squeeze(), cmap='gray')  # Convert from tensor to NumPy array
    plt.title('Ground Truth Mask')
    plt.axis('off')

    plt.subplot(5, 3, i*3 + 3)
    plt.imshow(binary_prediction[random_index].cpu().squeeze(), cmap='viridis')
    plt.title('Predicted Mask')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

```
4/4 [==============================] - 0s 140ms/step
Random indices: [6, 34, 11, 98, 52]
```

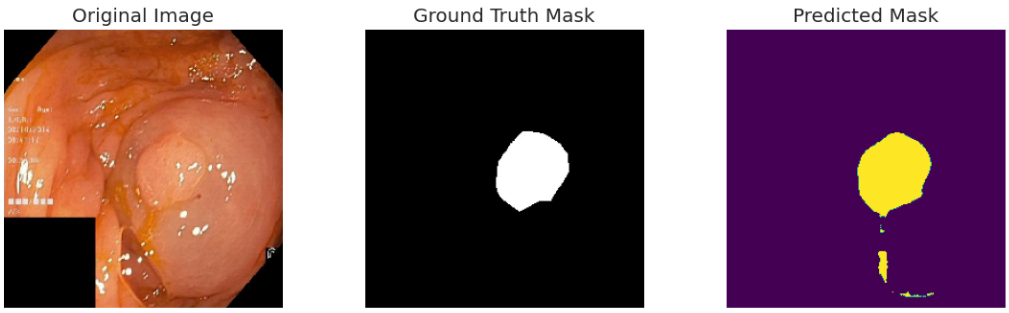

Figure 2.34: Precision Example

the model is overly optimistic in identifying polyps. To solve this, we can increase the prediction threshold by reducing the number of false positives, but it may also reduce the model's sensitivity to true positives. In a medical context, this is a delicate balance that should be struck carefully, ideally in consultation with clinical experts.

- **Comparative Results**

Table 2.1: Comparative Analysis of Different Segmentation Models

Model	Precision	Recall	IoU	Dice Coefficient
U-Net(ours)	0.85	0.80	0.75	0.78
V-Net	0.88	0.83	0.78	0.81
DeepLab	0.90	0.85	0.80	0.84
Mask R-CNN	0.92	0.87	0.82	0.85

The table above provides a comparative analysis of different segmentation models, highlighting their performance in terms of precision, recall, IoU, and Dice Coefficient. As observed, models like DeepLab and Mask R-CNN show improved precision and recall compared to the standard U-Net, suggesting their potential for better boundary delineation and overall segmentation accuracy.

2.8 Deployment

2.8.1 MLOps Methodology

MLOps (Machine Learning Operations) is a set of practices that aims to deploy and maintain machine learning models in production reliably and efficiently. It combines Machine Learning (ML) and DevOps (Development Operations) to provide continuous integration and continuous delivery (CI/CD) of ML models. The main components of MLOps include:

- **Version Control**: Managing code, data, and model versions using tools like Git and DVC (Data Version Control).
- **CI/CD Pipelines**: Automating the building, testing, and deployment of models. This ensures that changes are tested and deployed smoothly.
- **Monitoring**: Continuously monitoring model performance in production to detect data drift, model degradation, and other issues.
- **Infrastructure as Code (IaC)**: Managing and provisioning infrastructure using code, ensuring reproducibility and scalability.
- **Collaboration**: Facilitating collaboration between data scientists, ML engineers, and operations teams.

By integrating these practices, MLOps ensures that ML models can be quickly and safely deployed to production, while also allowing for iterative improvements and scaling.

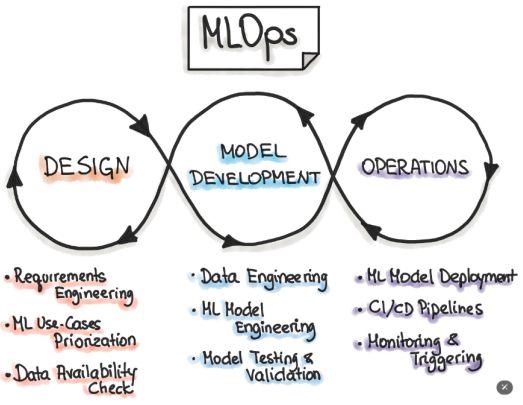

Figure 2.35: Mlops

2.8.2 Streamlit Application

To provide an interactive interface for users to interact with our trained U-Net model, we developed a Streamlit application. Streamlit is an open-source Python library that allows for the rapid creation of custom web apps for machine learning and data science projects. Below are the main features and functionalities of the Streamlit app:

Figure 2.36: Streamlit

- **User Interface**: The app provides a clean and intuitive user interface where users can upload images and view the segmentation results.
- **Image Upload**: Users can upload their own images to the app for segmentation.
- **Model Inference**: The uploaded images are processed through the trained U-Net model to generate segmentation masks.
- **Visualization**: The original image and the corresponding segmentation mask are displayed side by side, allowing users to visually inspect the results.
- **Download Option**: Users have the option to download the segmented masks for further use.

2.8.2.1 Implementation Details

The implementation of the Streamlit app involves several key steps:

- **Loading the Model**: The pre-trained U-Net model is loaded into the app using PyTorch.

app.py

C: > Users > gharb > Downloads > app.py

```python
import streamlit as st
import os
import shutil
import numpy as np
import torch
from torchvision import transforms
from PIL import Image
import segmentation_models_pytorch as smp

# Fonction pour charger le modèle U-Net avec l'encodeur ResNet50
def load_model():
    class_size = 1  # Pour la segmentation binaire
    model = smp.Unet(
        encoder_name="resnet50",
        encoder_weights="imagenet",  # Utilisez les poids pré-entraînés d'Imagenet pour l'encodeur
        in_channels=3,  # Images RVB
        classes=class_size,  # Masque binaire
    )

    model.eval()
    return model

# Fonction pour prétraiter l'image
def preprocess_image(image):
    preprocess = transforms.Compose([
        transforms.Resize((256, 256)),
        transforms.ToTensor(),
    ])
    image = preprocess(image).unsqueeze(0)  # Ajouter une dimension de lot
    return image

# Fonction pour effacer le répertoire temporaire
def clear_temp_directory(directory):
    if os.path.exists(directory):
        shutil.rmtree(directory)
    os.makedirs(directory, exist_ok=True)
```

Figure 2.37: App.py

- **Image Processing**: Uploaded images are pre-processed to match the input requirements of the model.

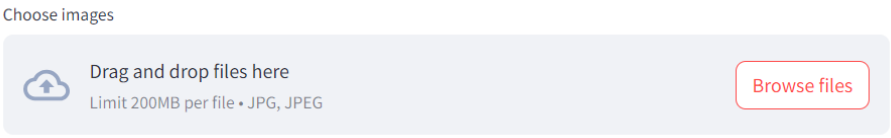

Figure 2.38: Upload Image

- **Inference Pipeline**: The app includes an inference pipeline that processes the images through the U-Net model to generate segmentation masks.

- **Visualization**: Streamlit's built-in functions are used to display the images and the segmentation results.

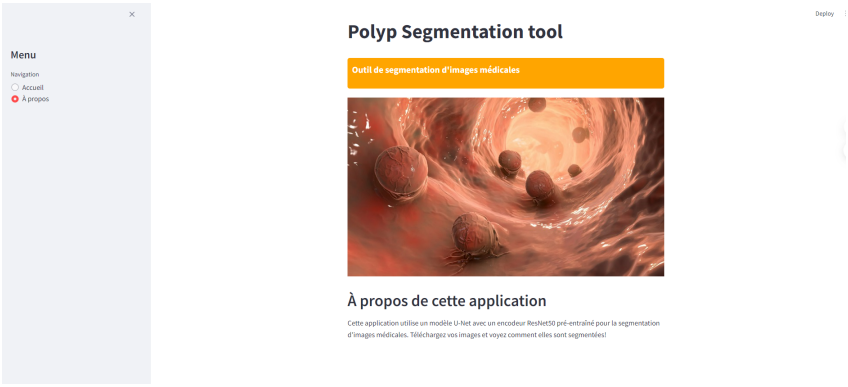

Figure 2.39: About App

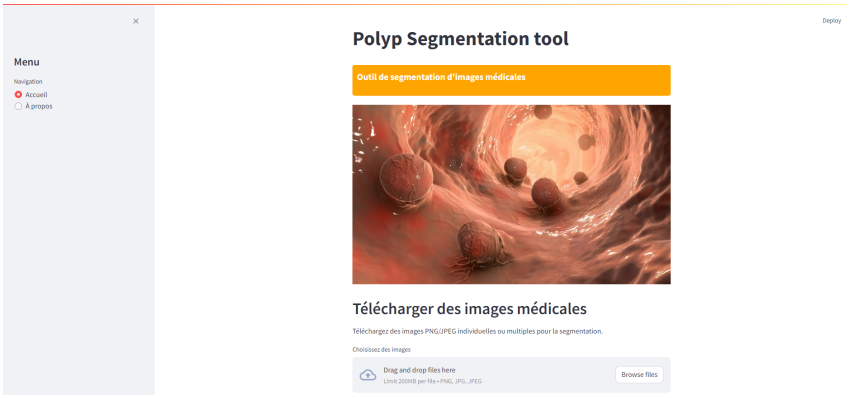

Figure 2.40: Visualisation

2.9 Conclusion

2.10 Conclusion

In this chapter, we delved into the comprehensive process of developing a robust polyp segmentation model, which is critical for the prevention of colorectal cancer. Our exploration began with a detailed understanding of the data, highlighting its source, structure, and the necessary preprocessing steps to ensure quality and consistency. We meticulously examined various attributes of the dataset, such as image components, dimensions, polyp positions, sizes, and color properties, to gain a thorough insight into the data's characteristics and limitations.

Following data understanding, we discussed data partitioning strategies, emphasizing the importance of an effective train-test split and data augmentation techniques to enhance model generalization. This foundational groundwork set the stage for the modeling phase, where we justified our choice of the U-Net architecture. We provided an in-depth explanation of U-Net's

structure and its suitability for our segmentation task, along with a detailed account of the training process.

Our evaluation section was divided into quantitative and qualitative assessments. Quantitatively, we employed various metrics such as precision, recall, IoU, and Dice Coefficient to rigorously measure the model's performance. Qualitatively, we conducted visual inspections of the predicted segmentation masks, identifying strengths and areas for improvement. This dual approach ensured a holistic evaluation of the model's efficacy.

The deployment phase outlined our adoption of the MLOps methodology, integrating best practices to ensure the seamless transition of our model from development to production. We introduced a Streamlit application as an accessible interface for end-users to interact with the model, demonstrating practical applications of our work.

Overall, this chapter provided a structured framework for building and deploying an effective polyp segmentation model, showcasing the intersection of data science, machine learning, and operational excellence. Our efforts contribute to the broader goal of early detection and prevention of colorectal cancer, highlighting the vital role of technological advancements in healthcare.

Conclusion and Future Works

Conclusion

In this thesis, we have investigated the development of a polyp segmentation model aimed at enhancing early detection methods for colorectal cancer. We thoroughly understood the dataset, examining its source, structure, and preprocessing requirements. Meticulous data exploration provided insights into key attributes such as image components, polyp positions, sizes, and color properties, informing our subsequent modeling decisions.

The U-Net architecture was effective in accurately segmenting polyps from colonoscopy images. Supported by data augmentation and hyperparameter tuning, the training phase yielded promising results in precision, recall, IoU, and Dice Coefficient.

Adopting MLOps methodologies facilitated the deployment phase, ensuring a smooth transition from development to production. The integration of a Streamlit application provided a user-friendly interface, enhancing accessibility for end-users.

Overall, our research contributes to advancing early detection methods for colorectal cancer, leveraging machine learning techniques to improve patient outcomes and reduce associated mortality rates.

Future Work

While significant milestones have been achieved in polyp segmentation, several avenues for future research and development exist:

- **Integration of Clinical Data**: Incorporating clinical metadata such as patient demographics, medical history, and biopsy results to enrich the dataset and enhance predictive capabilities.
- **Exploration of Advanced Architectures**: Investigating novel neural network architectures tailored for polyp segmentation, such as attention mechanisms or graph-based networks, to further improve model accuracy and robustness.
- **Real-time Deployment**: Implementing real-time polyp segmentation systems during colonoscopy procedures to provide immediate feedback to healthcare practitioners and improve patient care.

- **Longitudinal Studies**: Conducting longitudinal studies to evaluate the long-term effectiveness of polyp segmentation models in clinical settings, tracking patient outcomes and recurrence rates over time.
- **Collaboration with Clinical Experts**: Collaborating with gastroenterologists and oncologists to validate model predictions, interpret segmentation results, and incorporate domain knowledge into the development process.

By pursuing these avenues, we can further advance the field of polyp segmentation and contribute to ongoing efforts in combating colorectal cancer.

Bibliography

[Chen 2018] 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), pages 801–818, 2018. (Cited on page 15.)

[Chen 2021] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille and Yuyin Zhou. *Transunet: Transformers make strong encoders for medical image segmentation*. arXiv preprint arXiv:2102.04306, 2021. (Cited on page 15.)

[Fan 2020] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen and Ling Shao. *Pranet: Parallel reverse attention network for polyp segmentation*. In International conference on medical image computing and computer-assisted intervention, pages 263–273. Springer, 2020. (Cited on page 15.)

[Ronneberger 2015] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-net: Convolutional networks for biomedical image segmentation*. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pages 234–241. Springer, 2015. (Cited on page 15.)

[Sun 2024] Guanqun Sun, Yizhi Pan, Weikun Kong, Zichang Xu, Jianhua Ma, Teeradaj Racharak, Le-Minh Nguyen and Junyi Xin. *DA-TransUNet: integrating spatial and channel dual attention with transformer U-net for medical image segmentation*. Frontiers in Bioengineering and Biotechnology, vol. 12, page 1398237, 2024. (Cited on page 15.)

[Zhang 2018] Zhengxin Zhang, Qingjie Liu and Yunhong Wang. *Road extraction by deep residual u-net*. IEEE Geoscience and Remote Sensing Letters, vol. 15, no. 5, pages 749–753, 2018. (Cited on page 15.)

Ministère de l'Enseignement Supérieur et
de la Recherche Scientifique
Institut International de Technologie à Sfax

IIT
INSTITUT
INTERNATIONAL
TECHNOLOGIE
Université Nord Américaine privée

وزارة التعليم العالي والبحث العلمي
المدرسة العليا الدولية الخاصة للتكنولوجيا بصفاقس

End of Year Project Report

Presented at

International Institute of Technology at Sfax

by

Gharbi Khouloud
Ben mabrouk Minyar

DEPARTEMENT : Software Engineering and Decision Computing

TOPIC : A Polyp segmentation method for prevention of colorectal cancer

Defended on 11/06/2024 in front of the commitee composed of :

Prof Fourati Rahma *Supervisor*

Prof Bouarada Najla *Examiner*

Prof Ben said Fatma *Examiner*

Année Universitaire 2023/2024