ACKNOWLEDGEMENTS

Firstly, I would like to express my deep gratitude to Ph.D Vu Thi Ly, who directly guided, cared, enthusiastically instructed and created all favorable conditions during the time I was completing this project.

Secondly, I would like to thank the teachers in the Department of Information Technology - Military Technique Academy for teaching me knowledge about general subjects as well as specialized subjects, helped me have a solid theoretical foundation and facilitated me throughout my studies. In particular, I would like to extend my sincere appreciation to Assoc.Prof. Nguyen Quang Uy for his invaluable insights and unwavering motivation during my academic journey.

Finally, I would like to dedicate this research thesis to my mother, who has been a constant source of support and inspiration. Her unwavering encouragement, wisdom, and love have helped me overcome numerous challenges and achieve my goals. I am deeply grateful for her sacrifices and the countless ways she has enriched my life. Thank you, Mom, for everything.

Author

Trinh Thi Bao Anh

ABBREVIATIONS

No.	Abbreviation	Meaning
1	AI	Artificial Intelligence
2	ASPP	Atrous Spatial Pyramid Pooling
3	BSE	Binary Cross Entropy
4	CLR	LLeakyRelu
5	CNN	Convolutional Neural Network
6	DSC	Dice Similarity Coefficient
7	DSDF	Dual-Scale Dense Fusion
8	FCB	Fully Convolutional Branch
9	FCBFormer	Fully Convolutional Branch Trans-Former
10	FCN	Fully Convolutional Network
11	FFN	Feed-Forward Networks
12	FN	False Negative
13	FP	False Positive
14	FPR	False Positive Rate
15	IoU	Interest-Over-Union
16	LE	Local Emphasis
17	LN	Layer Norm
18	MiT	Mix Transformer
19	ML	Machine Learning
20	MLP	Multi Layer Perceptron

No.	Abbreviation	Meaning
21	MSA	Multiheaded Self-attention
22	MSE	Mean Squared Error
23	MSRFNet	ulti-Scale Residual Fusion Network
24	NLP	Natural Language Processing
25	PH	Prediction Head
26	ResNet	Residual Network
27	ReLU	Rectified Linear Unit
28	SFA	Stepwise Feature Aggregation
29	SOTA	State-of-the-art
30	TB	TransFormer Branch
31	TP	True Positive
32	TN	True Negative
33	TPR	True Positive Rate
34	ViT	Vision Transformer

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1. Research Motivation

Gastrointesinal (GI) lesions can occur in various parts of the GI tract. Some types of lesions can be precursors to dangerous cancers such as gastric cancer and colorectal cancer that result in over 640,000 deaths annually [27]. Lesion detection is often carried out through endoscopy, and diagnosis is carried out based on the examination of these images. However, endoscopic procedures may miss 20% to 47% of lesions [3] due to subjective and objective factors, such as equipment limitations, endoscopic techniques, and physician interpretation. This underscores the importance of developing computer-assisted lesion segmentation systems that can aid physicians to reduce the risk of missing lesions during procedures.

Using deep learning for lesion segmentation has achieved exceptional accuracy on benchmark datasets. Among various deep learning based methods, the encoder-decoder architecture of Unet [39], which combines low-level concrete features and high-level abstract features of an input image, has proven to be one of the most effective architectures for the segmentation task. Variants of Unet such as UNet++ [57], ResUNet++ [14] have improved upon the original Unet by adopting nested or stacking approaches. Recently, the ESFPNet architecture [42], which uses the Mix Transformer (MiT) encoder as the backbone and an efficient stage-wise feature pyramid (ESFP) as the decoder, is proved to be one of the most effective models for the segmentation task [17].

However, deep learning models in general and ESFPNet in particular for lesion segmentation still pose several limitations. First, the lesion areas usually have various textures, shapes, and sizes, which makes them difficult to recognize. Second,

the lesion segmentation is formulated as a classification problem at the pixel level where every image pixel needs to be classified into a normal class or abnormal class. In which, the imbalance between number of lesion pixels and normal pixels of small-sized lesion area reduces the effectiveness of deep learning models. A number of loss functions have been proposed to deal with these problems when training deep learning models. However, some limitations remain.

Therefore, to handle above challenging issues, this thesis proposes an approach for improving training an UNet-based model, specifically ESFPNet, for effective segmentation.

2. Research Purpose

The objective of this research is to enhance the effectiveness of deep learning techniques, specifically employing the ESFPNet architecture - a variant based on UNet, for the segmentation of biomedical images within the context of privately acquired datasets encompassing a variety of medical conditions.

3. Research Object The research subjects of this thesis are endoscopy datasets of different gastrointestinal diseases, including Stomach cancer, Esophageal cancer, Peptic ulcer, Positive H.pylori gastritis, and Negative H.pylori gastritis.

4. Scope Limitation

This thesis focuses on studying several models based on the UNet network in medical endoscopy image segmentation. Specifically, the thesis conducted experiments on five different mentioned datasets of gastrointestinal lesion images.

The thesis concentrates on UNet and Transformer-based models due to their effectiveness in medical image data. When conducting experiments with deep neural networks, several parameters need to be considered, such as initialization methods, the number of layers, the number of neurons, activation functions, optimization methods, and learning rates. However, this thesis cannot fine-tune all the different settings of these parameters.

5. Research Methodology

- In theory:

Conduct a thorough review of the algorithms and deep neural network models for biomedical image segmentation, including their strengths and limitations.

Study the theoretical concepts of deep learning and image segmentation that are necessary for understanding these algorithms and models.

Based on the literature review, proposed the most appropriate algorithms and models for the research.

- In practice:

Collect and prepare a dataset of biomedical images with annotated ground truth segmentations.

Implement the proposed algorithms and models in Python, using PyTorch and Tensorflow.

Train and fine-tune the models using the prepared dataset, optimizing hyperparameters and tuning the architecture where necessary.

Evaluate the performance of the models using appropriate quantitative and qualitative metrics, comparing the results with state-of-the-art methods.

Draw conclusions and provide recommendations for future research.

6. Scientific significance of the project

Research the algorithms based on the existing papers and methods, thereby continuing to research, develop this problem by proposing new approaches to improve generalization, better meet the accuracy, fast computation processing and real-time response.

7. Practical significance of the project

Providing a tool for endoscopic image segmentation and real-time endoscopic video segmentation.

8. Thesis Overview

The remainder of the thesis is organized as follows:

- Chapter 1: We present an overview of medical image segmentation and deep learning. Furthermore, this chapter provides fundamental insights into deep learning and introduces some basic architectural components that offer essential insights for subsequent chapters.

- Chapter 2: This chapter has examined various well-known works on medical image segmentation based on the U-Net architecture. We have clarified the improvements made in these architectures compared to the original UNet architecture.

- Chapter 3: In this section, we present the utilization of a deep encoder-decoder network, which we refer to as ESFPNet, for medical image segmentation. Additionally, we introduce our proposed loss function, the Focal-BCE-IoU loss. Furthermore, we incorporate Lasso regularization as a learning strategy..

- Chapter 4: Here, we provide a description of the datasets and experimental settings. This chapter also presents the segmentation results obtained to evaluate the effectiveness of our proposed approach. Furthermore, we showcase our demo application in this section.

Finally, we sum up the paper in Conclusion section.

Chapter 1
BACKGROUND

1.1. Medical Image Segmentation

Image segmentation is a method of dividing a digital image into subgroups called image segments, reducing the complexity of the image and enabling further processing or analysis of each image segment. Technically, segmentation is the assignment of labels to pixels to identify objects, people, or other important elements in the image. It is a key building block of computer vision technologies and algorithms. It is used for many practical applications including medical image analysis, computer vision for autonomous vehicles, face recognition and detection, video surveillance, and satellite image analysis.

Several relevant heuristics, or high-level image features, such as color or contrast, can be useful for image segmentation. These features are the basis for standard image segmentation algorithms that use clustering techniques like edges and histograms. These traditional image segmentation techniques based on such heuristics can be fast and simple, but they often require significant fine-tuning to support specific use cases with manually designed heuristics. They are not always sufficiently accurate to use for complex images. Newer segmentation techniques using machine learning and deep learning to increase accuracy and flexibility is considered especially effective for image segmentation tasks.

Medical image segmentation is a task of extracting regions of interest from medical images, such as computed tomography (CT), magnetic resonance imaging (MRI), and ultrasound images. Medical image segmentation is a crucial step in medical image analysis and is used for various applications, including diagnosis,

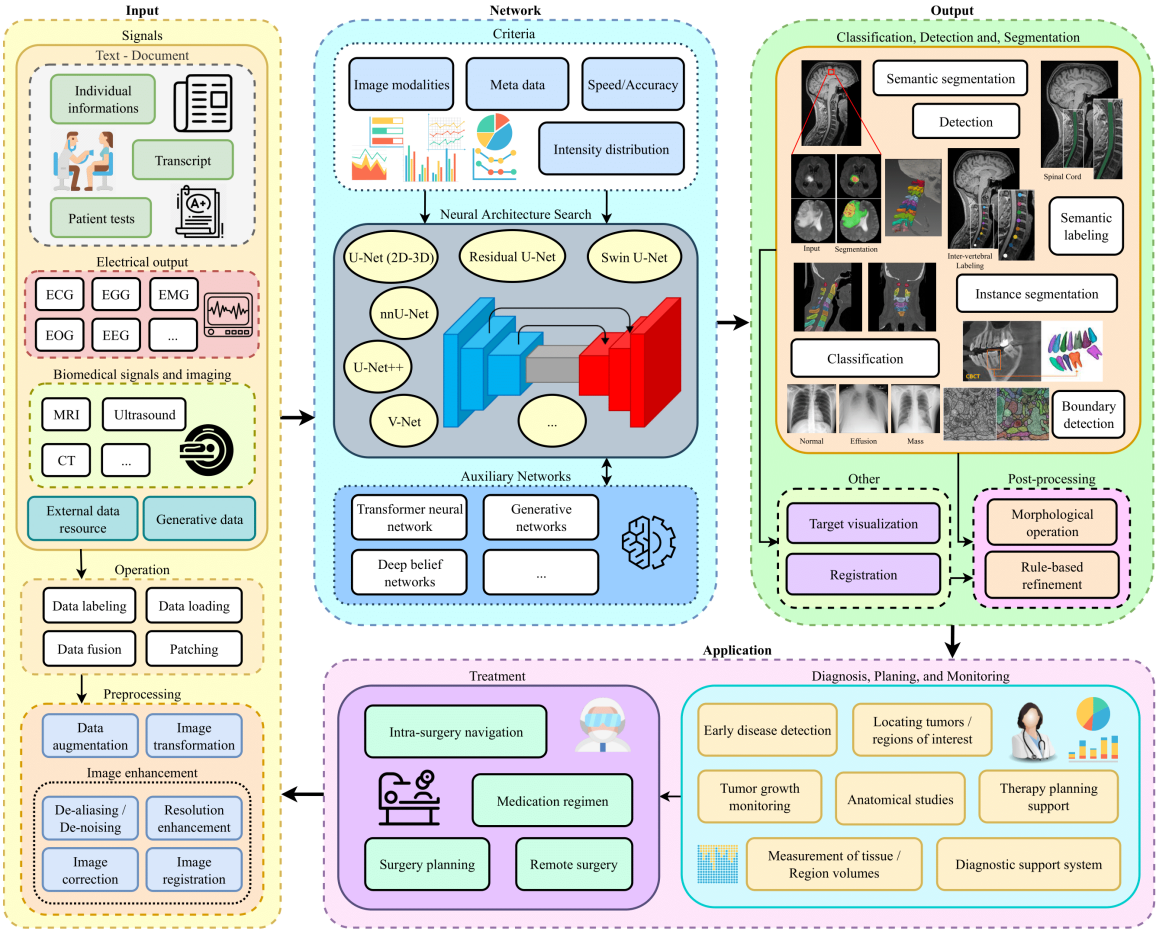

Figure 1.1: Applications of deep learning in medical image analysis and clinical use. Figure from [44].

treatment planning, and image-guided interventions (See Fig. 1.1).

In medical image segmentation, the goal is to partition the image into different regions, where each region represents a specific anatomical structure or pathological tissue. Segmentation methods in medical image analysis span traditional, machine learning, and deep learning techniques. Traditional methods like thresholding, region-growing, and watershed algorithms lay the groundwork. Machine learning employs k-means, fuzzy C-means, and SVM for data-driven segmentation. However, deep learning, specifically Convolutional Neural Networks (CNNs), has transformed the field. CNNs autonomously learn features, greatly enhancing accuracy and speed. These approaches collectively reshape medical image segmentation, enabling more precise clinical insights and applications. Several crucial architectures have emerged, shaping effective approaches in medical image segmentation, such as Fully Convolutional Network (FCN) [25], UNet [39],

SegNet [52], DeepLab [35], etc.

In recent times, there have been significant advancements in medical image segmentation, largely driven by deep learning techniques like convolutional neural networks (CNNs). These techniques can learn complex patterns from medical images, leading to more accurate segmentation results. Integration of attention mechanisms, transfer learning, and GANs has further improved segmentation quality, addressing challenges such as limited annotated data and complex anatomical structures.

Specific architectures like UNet and DeepLab variants, tailored for medical images, have also emerged, enhancing accuracy and applicability in clinical settings.

However, medical image segmentation faces significant challenges stemming from image variability, noise, complex anatomy, and limited annotations. Variability arises from diverse imaging sources, patient traits, and diseases. Noise, artifacts, and contrast variations complicate accurate segmentation. Complex anatomy, with intricate shapes and overlapping areas, requires precise delineation. Limited annotations, due to time-consuming expert labeling, hinder algorithm development. Overcoming these obstacles demands adaptable algorithms, noise-resilient techniques, advanced preprocessing, and innovative annotation strategies.

Gastrointestinal endoscopy images (Illustrated in Fig. 1.2) is type of medical images which play a vital role in evaluating and diagnosing issues related to the digestive system, including the esophagus, stomach, and other digestive regions. They provide a visual and detailed insight into the surface and structures of these organs through the use of a device known as an endoscope. Endoscopy images can offer information about health conditions, inflammation, injuries, and even early signs of cancer.

These images are typically captured by inserting a flexible endoscope into the internal organs through the digestive tract, and then transmitting the images to a

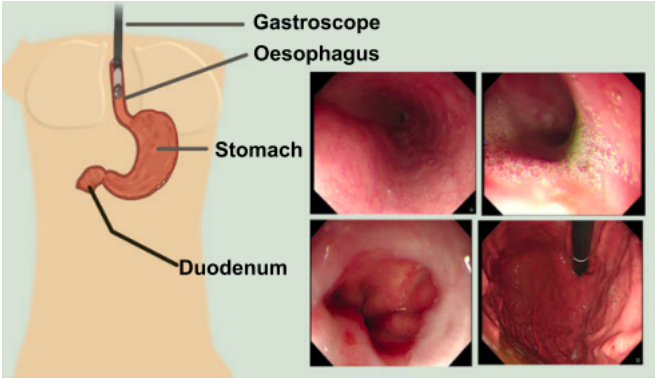

Figure 1.2: Gastrointestinal Endoscopy Images. Figure from [49].

screen for observation. These images have their unique characteristics, including color, surface structure, areas of injury, various types of lesions, and tumor-like growths.

However, the segmentation of endoscopy images faces several challenges. One of the main challenges is the diversity and complexity of structures and shapes of lesions and abnormalities within the images. For instance, a type of lesion like Seborrheic Keratosis can exhibit multiple variations in shape and color, making segmentation a complex task. Additionally, variations in lighting, angles, and image quality can further complicate the segmentation process. Furthermore, the dependence on the expertise of the endoscopy operator poses a challenge, as accurate segmentation requires a high level of knowledge and experience in both pathology and endoscopic techniques.

1.2. Deep Learning

Deep learning is a subfield of machine learning that uses artificial neural networks to model and solve complex problems. These networks are composed of layers of interconnected nodes or neurons that process input data and produce output predictions. Deep learning is characterized by its ability to automatically learn and improve through experience and by its ability to handle large and high-dimensional datasets.

Deep learning has been applied to a wide range of tasks such as image and

speech recognition, natural language processing, game playing, and autonomous vehicles. It has achieved remarkable success in these areas and has surpassed human-level performance in some cases.

One of the key advantages of deep learning is its ability to extract features and representations directly from raw data, eliminating the need for manual feature engineering. This allows deep learning models to learn complex patterns and relationships that would be difficult or impossible for humans to discern. However, deep learning also has some limitations, such as the need for large amounts of labeled data and computational resources, the difficulty in interpreting the reasoning behind the model's predictions, and the potential for overfitting to the training data.

Despite its limitations, deep learning has become a dominant approach in many fields, and its applications continue to grow as the technology advances.

1.2.1. Neural network

A neural network is a key component of deep learning, and is a mathematical model that is designed to mimic the way the human brain works. It is made up of interconnected layers of nodes, or neurons, which receive input data and perform mathematical operations to produce an output.

The input data is fed into the first layer of neurons, and each neuron calculates a weighted sum of the input data and applies an activation function to produce an output. The output of each neuron in the first layer is then passed as input to the next layer of neurons, and this process continues until the final output layer produces the prediction or classification result. Details of neural network are presented in Fig. 1.3

Deep neural networks can have many layers, hence the name ”deep learning”, and these layers are typically trained using large datasets through a process called backpropagation. During backpropagation, the error between the predicted output

and the actual output is calculated, and this error is used to adjust the weights of the neurons in the network so that the next prediction is more accurate.

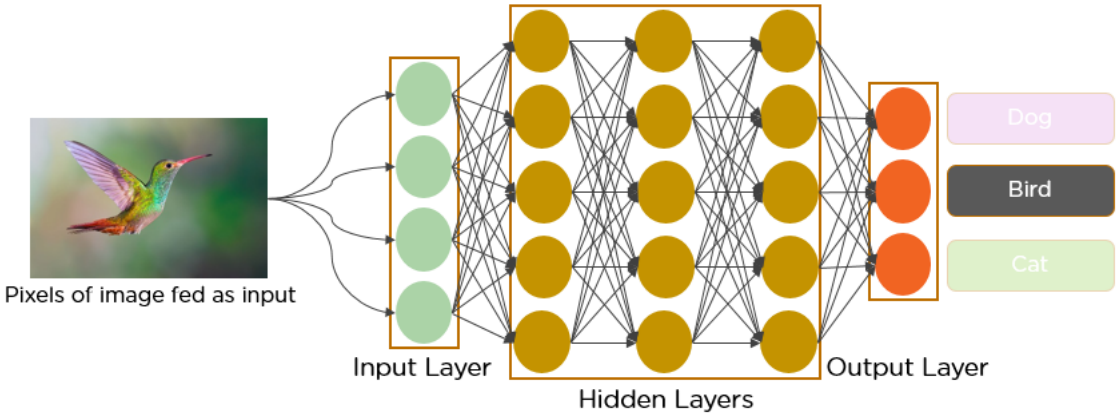

Figure 1.3: Deep Neural Network. Figure from [50].

Key factors influencing neutral network performance include:

1. *Architecture*: This pertains to how layers are organized and the neuron count in each layer. A well-designed architecture enhances efficiency and prediction accuracy. There are numerous common architectures, such as feedforward neural networks (FNN) [18] used for tasks like classification and regression, convolutional neural networks (CNN) [53] specialized in image and video processing, recurrent neural networks (RNN) [13], Long Short-Term Memory (LSTM) [47] and Gated Recurrent Unit (GRU) [31] for sequence data processing. There are also other specialized architectures like Generative Adversarial Networks (GANs) [20], Transformers [10], Residual Networks (ResNets) [28], and Capsule Networks (CapsNets) [46] for specific tasks. These architectures play a crucial role in addressing a wide range of challenges in the field of artificial intelligence and neural networks

2. *Optimization algorithms*: These are crucial for minimizing the cost function and finding optimal parameter values. Gradient Descent, SGD, Adam, etc., are popular choices.

3. *Learning rate*: The learning rate determines how quickly the weights of the neurons in the network are adjusted during training. A higher learning rate can lead

to faster convergence, but may also result in overshooting the optimal solution, while a lower learning rate can lead to slower convergence. Fig. 1.4 illustrates how learning rate affects on model's performance.

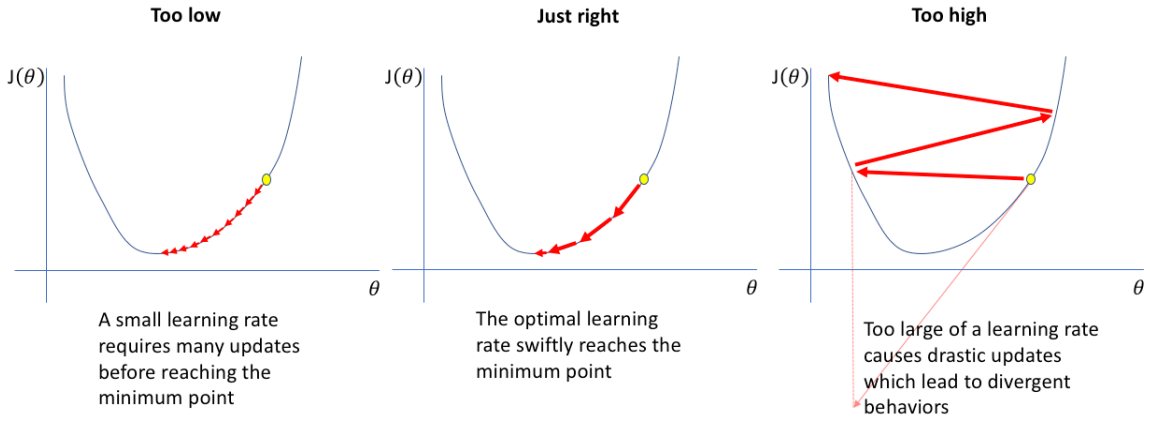

Figure 1.4: Effect of learning rate on performance of neural network. Figure from [37].

4. *Activation functions*: These functions introduce non-linearity, allowing the network to grasp complex input-output relationships. Fig. 1.5 illustrates some well-known activations.

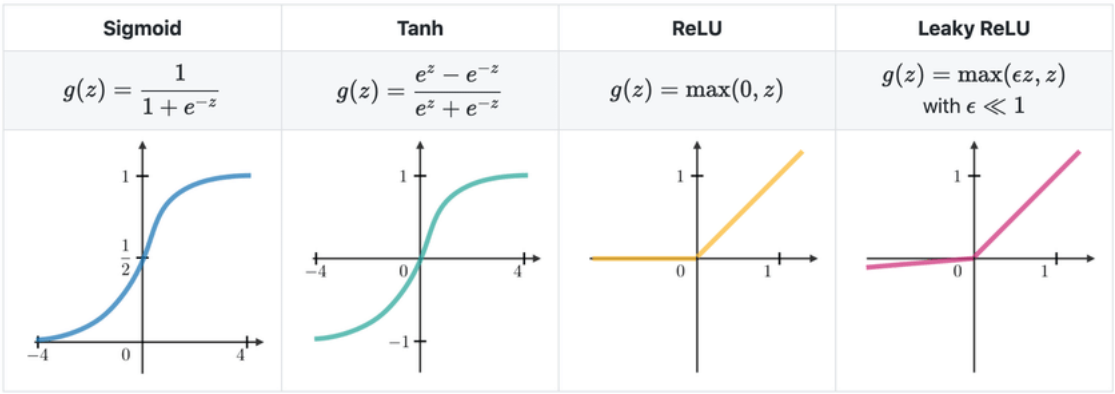

Figure 1.5: Most used activations in neural network. Figure from [30].

5. *Regularization*: Regularization techniques such as $L1$ or $L2$ regularization can help prevent overfitting by adding a penalty term to the loss function that encourages smaller weights and simpler models. Details in Eq. 1.1 and Eq. 1.2, respectively.

$$L_1 = Error(Y - \widehat{Y}) + \lambda \sum_{1}^{n} |w_i|, \tag{1.1}$$

$$L_2 = Error(Y - \widehat{Y}) + \lambda \sum_{1}^{n} w_i^2, \tag{1.2}$$

where Y is the groundtruth mask, $\hat{Y}$ is predicted mask, n is the number of model's weights, w_i is the value of weight i, λ is the regularization parameter.

6. *Hyperparameter tuning*: Parameters like layer count, neuron quantity, and learning rate need careful selection via hyperparameter tuning to optimize network performance.

1.2.2. Loss function

Loss functions in deep learning, especially in the context of image segmentation, play a crucial role in training neural networks. They quantify the dissimilarity between the predicted segmentation masks and the ground truth masks, helping the model learn to produce accurate segmentations. Here's an overview of some common loss functions used in image segmentation tasks:

1. *Pixel-wise Cross-Entropy Loss (Binary Cross-Entropy Loss):* It is suitable for binary image segmentation where each pixel is classified as foreground (object) or background (non-object). It also measures the dissimilarity between predicted pixel-wise probabilities and ground truth labels (0 or 1) and often used in binary semantic segmentation.

2. *Soft Dice Loss (F1-Score Loss):* This loss measures the overlap between predicted and ground truth masks by calculating the Dice coefficient. It is particularly useful for imbalanced datasets as it penalizes false positives and false negatives equally.

3. *Jaccard Index (Intersection over Union) Loss:* Similar to the Dice loss, it quantifies the overlap between predicted and ground truth masks. Concurrently, it also encourages better localization of object boundaries.

4. *Weighted Losses:* In cases where certain classes or regions are more impor-

tant than others, you can assign different weights to different classes or regions to balance the loss function.

5. *Focal Loss:* This addresses class imbalance by assigning higher weights to hard-to-classify examples and helps the model focus on correcting its mistakes, especially for rare classes.

6. *Boundary Loss:* The loss emphasizes accurate boundary prediction, often used in tasks where precise object boundaries are crucial, such as medical image segmentation.

7. *Adversarial Loss:* Combines segmentation loss with an adversarial loss to encourage realistic-looking segmentations in tasks like image-to-image translation.

8. *Regression Losses:* In some cases, regression losses like Mean Squared Error (MSE) or Huber loss are used when the output is not a segmentation mask but a continuous value (e.g., depth estimation).

It's important to choose an appropriate loss function that aligns with the specific goals and characteristics of the segmentation task. Experimentation with different loss functions and hyperparameters is often necessary to achieve the best results. Additionally, some segmentation models may use a combination of multiple loss functions to balance various aspects of the segmentation quality.

1.2.3. Performance Metrics

Performance metrics are used to evaluate the performance of a deep learning model. The choice of performance metrics depends on the type of task that the model is designed to perform. Firstly, we need to clarify the concepts within the confusion matrix.

A confusion matrix is a matrix representation of the prediction results of any binary testing that is often used to describe the performance of the classification model (or "classifier") on a set of test data for which the true values are known.

Each prediction can be one of the four outcomes, based on how it matches up to the actual value:

1. *True Positive (TP)*: Predicted True and True in reality.
2. *True Negative (TN)*: Predicted False and False in reality.
3. *False Positive (FP)*: Predicted True and False in reality.
4. *False Negative (FN)*: Predicted False and True in reality.

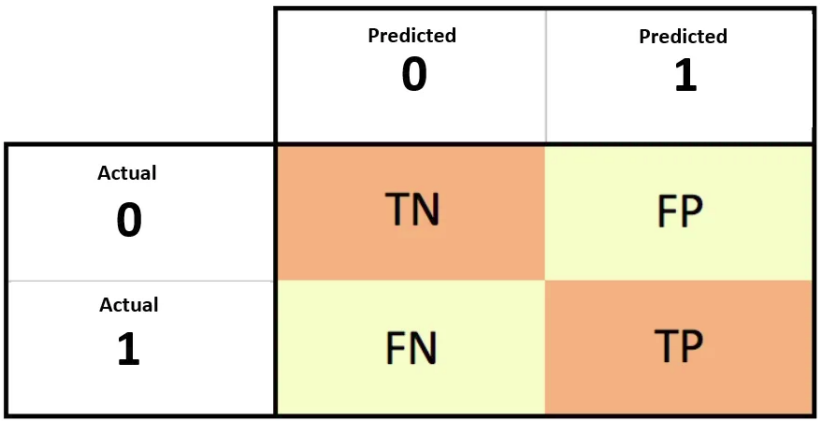

Figure 1.6: Confusion matrix with 2 class labels.

Here are some common performance metrics used in field of image segmentation: Please note that, in this section, N represents the total number of samples, TP_i denotes the true positive count for the i-th sample, TN_i is the true negative count for the i-th sample, FP_i is the false positive count, and FN_i is the false negative count for the same sample.

1. *mDice*: The mean Dice coefficient $(mDice)$ is a widely used metric in image segmentation tasks, including medical image analysis. It quantifies the similarity between the predicted segmentation mask and the corresponding ground truth mask. The Dice coefficient, also known as the F1-score, is calculated as the ratio of twice the intersection of the two masks to the sum of their sizes. Mathematically, it can be defined as:

$$mDice = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times TP_i}{2 \times TP_i + FP_i + FN_i} \tag{1.3}$$

The mDice score provides insight into the overall segmentation accuracy by considering both the precision and recall of the model's predictions.

2. *mIoU*: The mean of Intersection over Union $(mIoU)$ score (or Jaccard Index) measures the overlap ratio between the predicted mask and the ground truth mask. It is the area of overlap between these two masks divided by the area of union between them.

$$mIoU = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i + FN_i} \tag{1.4}$$

3. *mRecall*: The mean Recall $(mRecall)$ score assesses the ability of a segmentation model to correctly capture all relevant regions of interest in an image. It quantifies the ratio of true positive (TP) pixels to the total number of true positive (TP) and false negative (FN) pixels in all images. In other words, it measures the model's success in identifying all the actual positive areas within the ground truth masks.

The $mRecall$ score is defined as follows:

$$mRecall = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i} \tag{1.5}$$

4. *mPrecision*: Mean Precision $(mPrecision)$ is another essential evaluation metric in computer vision tasks, particularly in object detection and recognition. While $mIoU$ focuses on the overlap between predicted and ground truth masks, $mPrecision$ assesses the precision of the model's predictions, emphasizing the accuracy of positive detections.

It is calculated by measuring the ratio of true positive (TP) predictions to the sum of true positives (TP) and false positives (FP) for each instance:

$$mPrecision = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{1.6}$$

1.3. Encoder-Decoder Architechture

The Encoder-Decoder architecture first appeared in the paper [26] by Jonathan Long, Evan Shelhamer, and Trevor Darrell, published at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2015. It is subsequently widely used in numerous studies [19] [41] [5] [16]. The Encoder-Decoder architecture (Fig. 1.7) is a neural network architecture used in various tasks, including image segmentation. This architecture consists of two main parts: the Encoder and the Decoder.

1. *Encoder*: This part is responsible for transforming the input image into higher-level feature representations. Through a series of convolutional and pooling layers, the Encoder extracts important information from the image. These features are often high-depth feature maps that represent critical aspects of the image, such as edges and important regions.

2. *Decoder*: This part receives the features extracted from the Encoder and aims to reconstruct the original image or perform specific tasks like segmentation. The Decoder typically includes layers that transform features back into images using corresponding representations.

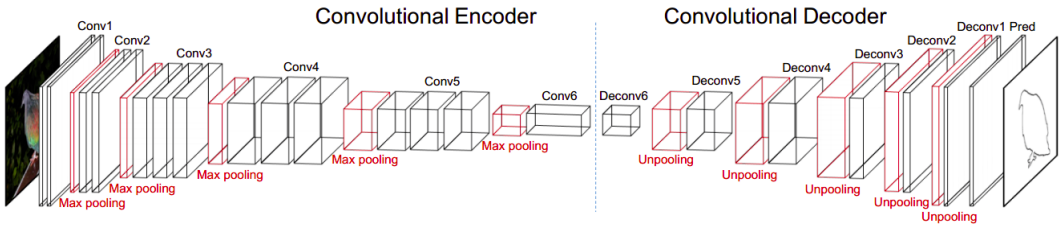

Figure 1.7: Encoder - Decoder Architechture. Figure from [51].

The Encoder-Decoder architecture is commonly used for tasks that require both global and local information from images, such as image segmentation. In image segmentation tasks, the Encoder can extract features related to edges, boundaries, and significant regions in the image. The Decoder can then utilize these features

to identify specific regions in the image and perform segmentation according to different classes. This architecture is normally enhanced with methods like skip connections to combine detailed information from different levels and improve the accuracy of segmentation.

1.4. UNet Architechtue

UNet [39] a convolutional neural network (CNN) architecture designed for image segmentation tasks. It was introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in 2015 and has since become one of the most popular and widely used networks for medical image segmentation.

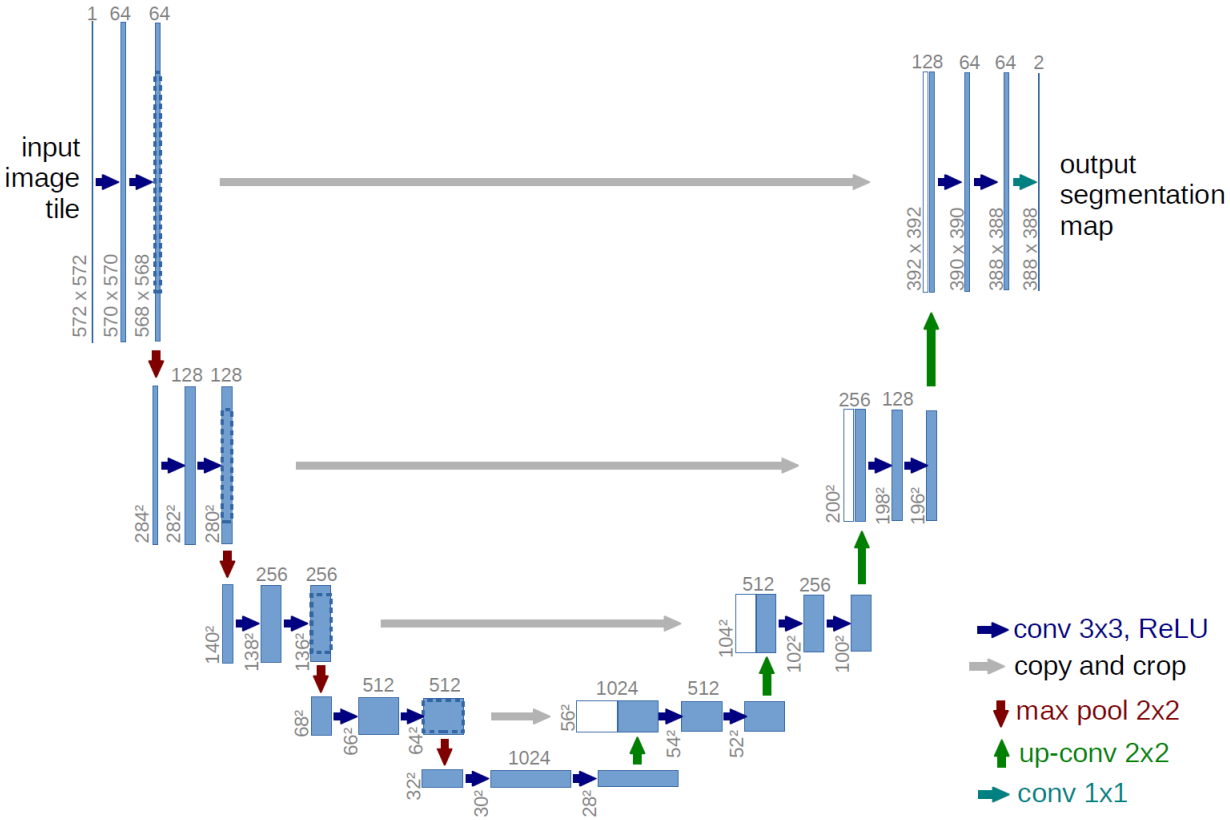

Figure 1.8: Unet Architechture. Figure from [39].

The network's architecture is depicted in Figure 1.8. It comprises two main pathways: a contracting path on the left and an expansive path on the right. The contracting path adheres to the conventional convolutional network structure, involving a series of repetitive operations. These operations include two 3x3 convo-

lutions (without padding), each succeeded by a rectified linear unit (ReLU), and a 2x2 max-pooling operation with a stride of 2 for downsampling. At each downsampling step, the number of feature channels is doubled. In contrast, each step in the expansive path entails an upsampling of the feature map followed by a 2x2 convolution (referred to as ”up-convolution”), which reduces the number of feature channels by half. Additionally, this step involves a concatenation with the corresponding cropped feature map from the contracting path and two 3x3 convolutions, each followed by a ReLU. The cropping step is essential due to the loss of border pixels during each convolution operation. Finally, at the last layer, a 1x1 convolution is employed to map each 64-component feature vector to the desired number of classes. In total, the network comprises 23 convolutional layers. To enable a seamless tiling of the output segmentation map, it is crucial to choose the input tile size so that all 2x2 max-pooling operations are applied to a layer with an even x- and y-size.

U-Net has been shown to be effective in a wide range of medical imaging applications, including brain tumor segmentation, liver segmentation, and lung segmentation, among others. Its success has led to the development of several variations and extensions, which will be introduced in the next chapter.

1.5. Residual Block

A residual block (Fig. 1.9), also known as a residual unit, is a fundamental building block in deep neural networks, particularly in convolutional neural networks (CNNs). It was introduced to address the vanishing gradient problem and enable the training of very deep networks effectively. Residual blocks play a crucial role in architectures like ResNet (Residual Network) [28].

The concept of a residual block involves the introduction of skip connections or shortcuts that allow the network to learn residual functions instead of attempting to learn the desired output directly. This is achieved by introducing identity

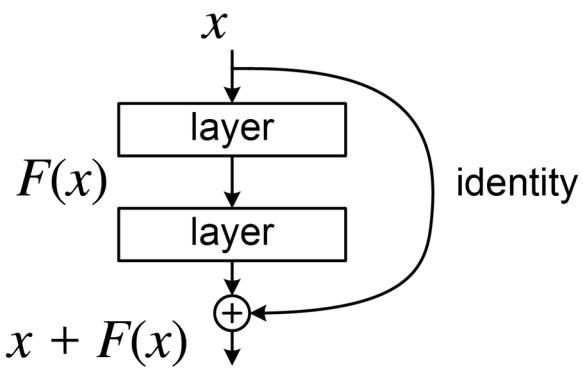

Figure 1.9: Residual Block. Figure from [28].

mappings that learn the difference between the input and the output of a layer. Mathematically, the residual function of a layer can be represented as:

$$F(x) = \mathcal{H}(x) - x \tag{1.7}$$

Here, $\mathcal{H}(x)$ represents the transformation learned by the layer.

The residual block typically consists of the following components:

1. *Input*: The input feature map x.

2. *Convolutional Layers*: One or more convolutional layers that perform feature extraction and transformation.

3. *Activation Functions*: Activation functions such as ReLU (Rectified Linear Unit) are often applied after the convolutional layers.

4. *Shortcut Connection*: The original input x is added element-wise to the transformed output $\mathcal{H}(x)$.

5. *Output*: The final output y of the residual block is calculated by:

$$y = F(x) + x \tag{1.8}$$

The addition of the input to the transformed output creates a bypass or shortcut connection that allows gradients to flow directly through the network without significant attenuation. This helps in training very deep networks more effectively and has led to significant improvements in performance.

Residual blocks have been widely adopted in various computer vision tasks, including image classification, object detection, image segmentation, and more. They have also inspired the design of subsequent architectures to improve training and convergence in deep networks [4] [45] [48].

1.6. Vision Transformer (ViT)

1.6.1. Transformer Encoder

The modern transformer was proposed in the 2017 paper titled 'Attention Is All You Need' [10] by Ashish Vaswani et al., Google Brain team. A Transformer block is a fundamental component within the Transformer architecture, used to construct both the Encoder and the Decoder. Each Transformer block consists of two main parts: the *Multi-Head Self-Attention Layer* and the *Position-wise Feed-Forward Neural Network*. Before delving into these two concepts, we need to have a clear understanding of *Scaled Dot-Product Attention*.

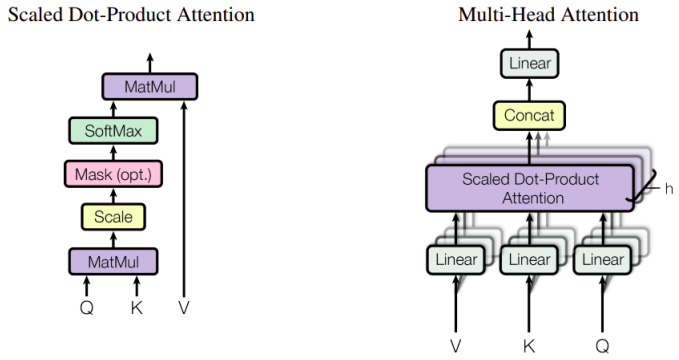

Figure 1.10: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Figure from [10].

Scaled Dot-Product Attention: The specific attention mechanism called "Scaled Dot-Product Attention" is shown in Figure 1.10. The input comprises queries and keys with a dimension of d_k and values with a dimension of d_v. To calculate this attention, we perform the following steps: first, we compute the dot products between the queries and all keys, then we divide each of these dot products by $\sqrt{d_k}$, and finally, we apply a softmax function to obtain the weights

corresponding to the values.

In practical applications, we perform the attention computation for a set of queries in parallel, which are organized into a matrix denoted as Q. Similarly, the keys and values are organized into matrices K and V, respectively. The resulting output matrix is computed as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1.9}$$

Please be aware that $\frac{1}{\sqrt{d_k}}$ is employed for the purpose of scaling the dot products. T stands for transpose matrix. Now, let's delve into our main concepts.

1. *Multi-Head Self-Attention Layer*: This is a crucial part that allows the model to focus on different parts of the input data. Self-attention enables the model to compute the importance of words/tokens in the sequence with respect to each other. Multi-head self-attention aggregates multiple instances of self-attention with different weights, allowing the model to learn various types of relationships within the data.

Rather than applying a single attention function with keys, values, and queries each having a dimension of d_{model}, it has been discovered that it is advantageous to perform linear projections of queries, keys, and values separately h times. These projections employ distinct learned linear transformations to reduce them to dimensions of d_k, d_k, and d_v, respectively. Subsequently, on these projected versions of queries, keys, and values, we execute the attention function in parallel, generating output values with a dimension of d_v. These outputs are then concatenated and subjected to an additional projection step, resulting in the final values, as depicted in Figure 1.10.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O, \tag{1.10}$$

where $head_i$ = $Attention(QW_i^Q, KW_i^K, VW_i^V)$, the projections are parameter matrices $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$ and $W^O \in$

$R^{d_{model} \times hd_v}$.

2. *Position-wise Feed-Forward Neural Network*: After the data has been rearranged through self-attention, it is passed through a separate feed-forward neural network for each position. This enables the model to learn complex representations from the data and prepares them for the next step. The Feed-Forward Neural Network consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{1.11}$$

Additionally, each Transformer block includes residual connections and layer normalization to stabilize the learning process and mitigate gradient vanishing issues. The combination of the Multi-Head Self-Attention Layer and Position-wise Feed-Forward Neural Network in each Transformer block enables the model to learn complex relationships in the data and effectively handle long-range interactions in lengthy data sequences. These Transformer blocks are stacked to form both the Encoder and the Decoder in the Transformer architecture, creating the capacity for self-learning and understanding intricate patterns within data sequences.

1.6.2. Vision Transformer

The Vision Transformer (ViT) model was introduced in a research paper published as a conference report at ICLR 2021, titled "An Image is Worth 16*16 Words: Transformers for Image Recognition in Scale." [2]. The Vision Transformer is an adaptation of the Transformer architecture for computer vision tasks, such as image classification, object detection, and segmentation.

The overall architecture of the ViT model is outlined as follows, following a step-by-step approach:

1. *Divide the image into fixed-size arrays called patches*: The standard Transformer model takes a 1D sequence of token embeddings as input. However, when dealing with 2D images, we transform the image $x \in R^{H \times W \times C}$ into a

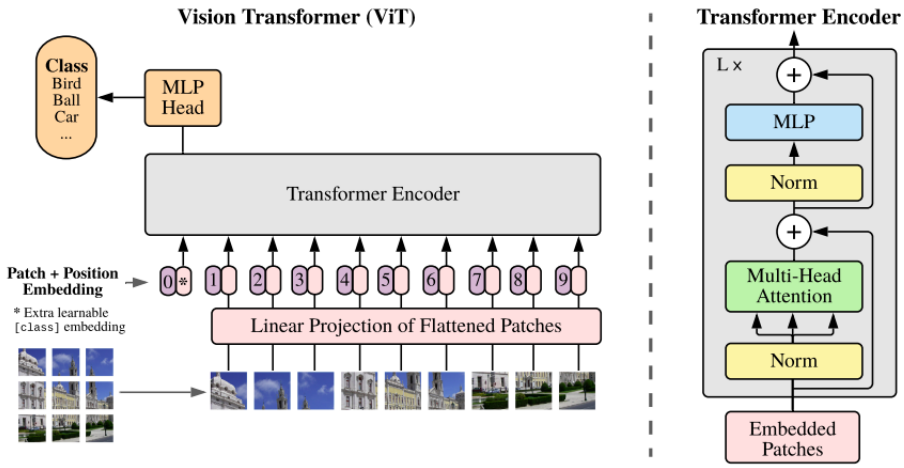

Figure 1.11: Vision Transformer. The input image is splited into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017) [10] .

sequence of 2D patches that have been flattened, denoted as $x_p \in R^{N\times(P^2\cdot C)}$. Here, (H, W) represents the original image's resolution, C signifies the number of channels, (P, P) denotes the resolution of each individual image patch, and $N = HW/P^2$ represents the resulting number of patches. Notably, this number of patches, N, also functions as the effective input sequence length for the Transformer.

2. *Flatten these image patches and generate lower-dimensional feature embeddings from these flattened image patches*: In the Transformer architecture, a consistent latent vector size D is maintained across all layers. To achieve this, we flatten the image patches and then employ a trainable linear projection (as defined in Eq. 1.12) to map them into D dimensions. The resulting vectors from this projection are what we call "patch embeddings."

3. *Include the ordering of the patches*: To preserve positional information, we incorporate position embeddings into the patch embeddings. In this regard, we employ conventional 1D position embeddings that can be learned. Interestingly, we have not observed substantial improvements in performance by

utilizing more advanced position embeddings that are sensitive to 2D structures.

4. *The sequence of feature embeddings becomes the input for the transformer encoder*: The Transformer encoder, as introduced [10], is composed of alternating layers of multiheaded self-attention (MSA) and MLP blocks (Eq. 1.13, 1.14). Layernorm (LN) is applied before every block, and residual connections after every block. Notably, similar to BERT's [class] token (read more at [21]), we prepend a learnable embedding to the sequence of embedded patches ($z_0^0 = x_{class}$), whose state at the output of the Transformer encoder z_L^0 serves as the image representation y (Eq. 1.15).

5. *Perform pre-training for the ViT model with image labels, then fine-tune it on a large dataset.*

6. *Fine-tune the model on task-specific datasets for individual problems.*

The MLP contains two layers with a GELU non-linearity.

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; ...; x_p^N E] + E_{pos} \qquad E \in R^{(P^2 \cdot C) \times D}, E_{pos} \in R^{(N+1) \times D} \tag{1.12}$$

$$z_l^{'} = MLP(LN(z_{l-1})) + z_{l-1} \quad l = 1...L \tag{1.13}$$

$$z_l = MLP(LN(z_l^{'})) + z_l^{'} \quad l = 1...L \tag{1.14}$$

$$y = LN(z_L^0) \tag{1.15}$$

1.7. Conclusion

This chapter provides fundamental backgrounds on deep learning, image segmentation, and analyzes some common components and techniques used in endoscopic image segmentation. In the next chapter, we will explore some related researches for lesion segmentation tasks.

Chapter 2
RELATED WORKS

The development of endoscopic image segmentation is still in its early stages, but it has the potential to revolutionize the way that endoscopic procedures are performed. The current development of endoscopic image segmentation is focused on the use of deep learning techniques. Deep learning models have been shown to be very effective at segmenting endoscopic images, especially for tasks such as polyp detection and classification.

An initial effort to tackle semantic segmentation using a deep neural network was proposed in [11]. This approach involves passing input images through a convolutional encoder to generate latent representations. Subsequently, fully connected layers are employed on the derived feature maps to generate pixel-level predictions. However, a primary drawback of this architecture stemmed from the utilization of fully connected layers, which led to the loss of spatial information and consequently led to a decline in overall performance.

To overcome this limitation, Long et al. introduced Fully Convolutional Networks (FCNs) [11]. The FCN architecture employs multiple convolutional blocks during the encoder path, comprising convolution, activation, and pooling layers. This configuration facilitates the capture of semantic representations. Similarly, in the decoding path, convolutional layers combined with up-sampling operations are used to achieve pixel-level predictions. The sequential up-sampling process in the decoding path aims to gradually enhance spatial dimensions for more precise segmentation outcomes.

Drawing inspiration from the FCN architecture and encoder-decoder models,

Ronneberger et al. devised the UNet model for biomedical image segmentation. Since its inception, UNet has provided a fundamental and effective architecture for image segmentation, particularly in the field of medicine. From this basic architecture, many Unet-based variants have emerged. Specific advancements in Unet include the enhancement of skip connections, replacing convolutional blocks with residual blocks, integrating attention mechanisms into the model, and most recently, the utilization of transformer blocks in the UNet backbone, which is a new trend in medical image segmentation. Below, we will present some popular image segmentation architectures based on UNet.

2.1. ResUNet++

2.1.1. Deep Residual U-Net architecture

Deep ResUnet [55] is a semantic segmentation neural network that harnesses the strengths of both UNet and residual neural networks. This fusion offers two significant advantages:

1. The inclusion of residual units facilitates the training process of the network, enhancing its overall performance.

2. The strategic use of skip connections within a residual unit and between lower-level and higher-level parts of the network enables seamless information propagation without degradation. This design allows for the creation of a neural network with significantly fewer parameters while achieving comparable, if not superior, performance in semantic segmentation.

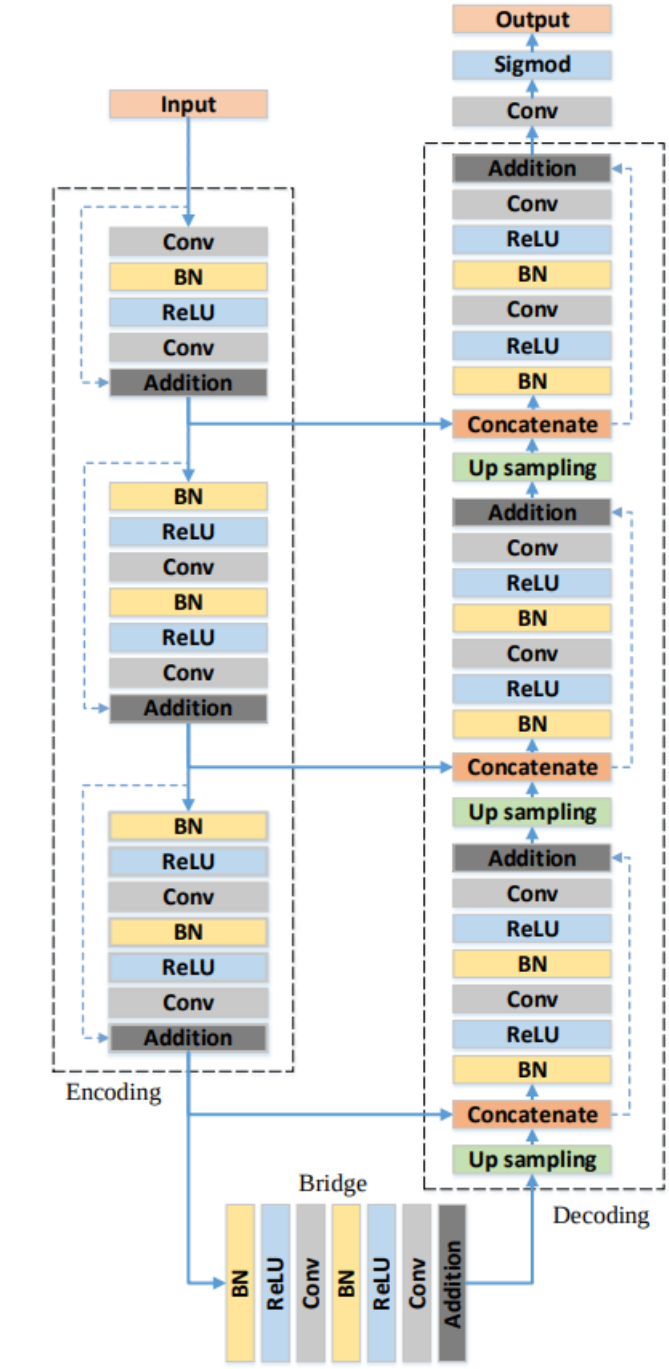

Figure 2.1: The architecture of the Deep Residual U-Net. Figure from [55].

Zhang's study employs a 7-level architecture for Deep ResUnet, consisting of three main components: encoding, bridge, and decoding. The encoding phase condenses the input image into compact representations. The decoding phase reconstructs these representations into pixel-wise categorization, i.e., semantic segmentation. The bridge component acts as an intermediary, connecting the encoding and decoding pathways.

All three components are constructed using residual units, each of which consists of two 3×3 convolution blocks and an identity mapping. Each convolution block comprises a Batch Normalization (BN) layer, a Rectified Linear Unit (ReLU) activation layer, and a convolutional layer. The identity mapping establishes a direct connection between the input and output of the unit.

The encoding path incorporates three residual units. Instead of using pooling operations to downsample the feature map size, a stride of 2 is applied to the first convolution block within each unit, effectively reducing the feature map size by half.

Similarly, the decoding path is composed of three residual units. Before each unit, there is an up-sampling of feature maps from lower levels, followed by concatenation with the feature maps from the corresponding encoding path. After the final level of the decoding path, a 1×1 convolution layer and a sigmoid activation layer are employed to project the multi-channel feature maps into the desired segmentation.

In total, the network consists of 15 convolutional layers, as compared to the 23 layers in UNet. Notably, the network eliminates the need for cropping, which is a necessity in UNet.

The ResUNet++ [14] architecture is based on the Deep Residual UNet [55]. The proposed ResUNet++ architecture takes advantage of the residual blocks (as mentioned above), the squeeze and excitation block, ASPP, and the attention block.

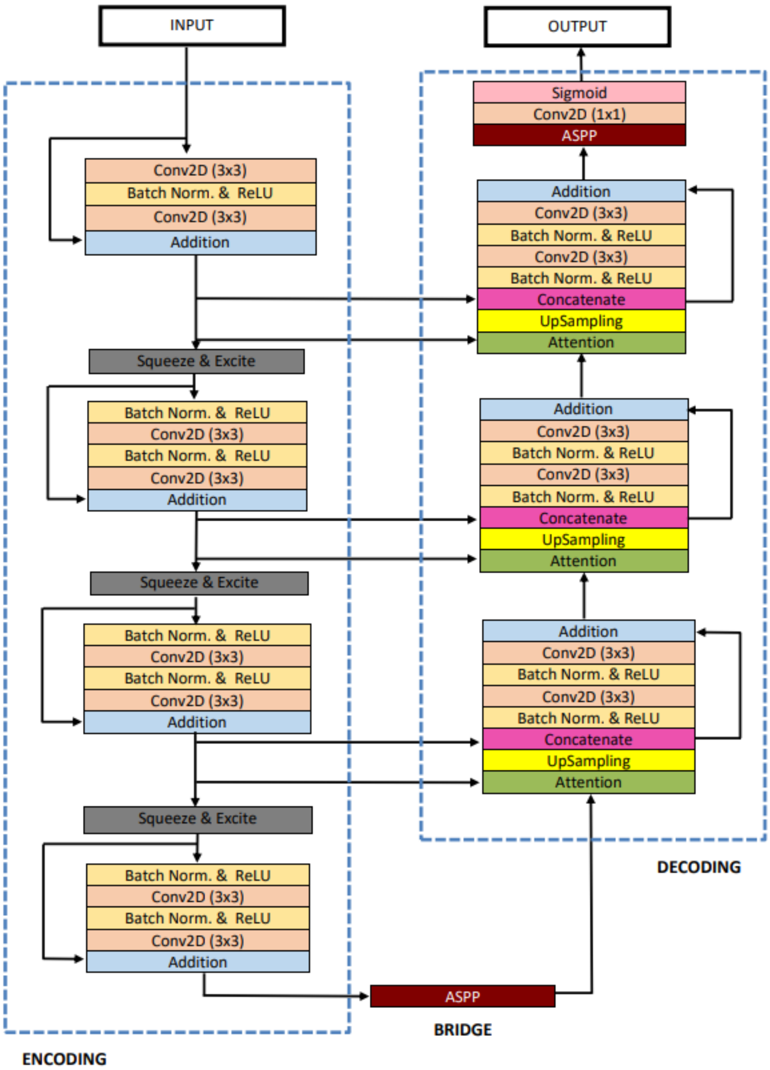

Figure 2.2: Block diagram of the ResUNet++ architecture. Figure from [14].

Next, we will systematically delve into these concepts one by one.

2.1.2. Squeeze and Excitation Units

The squeeze-and-excitation network, as introduced in [23], enhances the representational capacity of the network by recalibrating feature responses through precise modeling of inter-dependencies between channels. The primary objective of the squeeze-and-excite block is to enable the network to elevate its sensitivity to relevant features while suppressing unnecessary ones. This objective is accomplished through two distinct steps:

Firstly, in the ”squeeze” step (global information embedding), each channel

undergoes compression through global average pooling to generate channel-wise statistics.

Secondly, in the ”excitation” step (active calibration), the block aims to comprehensively capture channel-wise dependencies [23].

Within the ResUNet++ architecture, the squeeze-and-excitation block is integrated with the residual block. This integration serves to enhance effective generalization across diverse datasets and contributes to improved network performance.

2.1.3. Atrous Spatial Pyramidal Pooling

The concept of ASPP is inspired by spatial pyramidal pooling [29], which has proven effective in resampling features at multiple scales. ASPP, or Atrous Spatial Pyramid Pooling, aims to capture contextual information at various scales [34] [35]. This is+ achieved by employing multiple parallel atrous convolutions [33] with different rates on the input feature map. Atrous convolution allows for precise control over the field-of-view, enabling the capture of multi-scale information.

In this architecture, ASPP functions as a critical bridge between the encoder and decoder components, as illustrated in Figure 2.2. ASPP has demonstrated promising results in various segmentation tasks by providing valuable multi-scale information. Therefore, we leverage ASPP to capture essential multi-scale information for the semantic segmentation task.

2.1.4. Attention Units

The attention mechanism, primarily popularized in Natural Language Processing (NLP) [10], is designed to selectively attend to a specific subset of its input. This mechanism has also found applications in semantic segmentation tasks, such as pixel-wise prediction. Its role is to identify which regions of the neural network

warrant increased attention.

Furthermore, the attention mechanism serves the purpose of reducing the computational burden associated with encoding information from each polyp image into a fixed-dimensional vector. One of the primary advantages of attention mechanisms is their simplicity and adaptability to input of varying sizes. They enhance the quality of features, thereby improving overall results.

In the context of the previous approaches, UNet [39] and ResUNet [55], a direct concatenation of encoder feature maps with decoder feature maps is employed. Inspired by the success of the attention mechanism in both NLP and computer vision tasks, an attention block is incorporated into the decoder portion of the architecture. This inclusion allows the model to focus specifically on the critical regions within the feature maps.

2.2. ResNet34-UNet

The ResNet34-UNet [38] architecture combines two popular neural network architectures: ResNet [28] and UNet [39]. Details of UNet was presented in 1.4. Meanwhile, ResNet is a deep convolutional neural network architecture that introduced the concept of residual learning. It utilizes residual blocks (Details in 1.5.), which are composed of multiple convolutional layers. Residual learning involves adding shortcut connections (skip connections) that skip one or more layers, allowing gradients to flow more easily during training. This helps mitigate the vanishing gradient problem and enables the training of very deep networks.

The ResNet34-UNet architecture is not a standard or widely recognized neural network model. It is a combination of two popular architectures, ResNet and UNet, and might be used in various applications, but it doesn't have a specific standard implementation or a well-known pre-trained model like some other architectures.

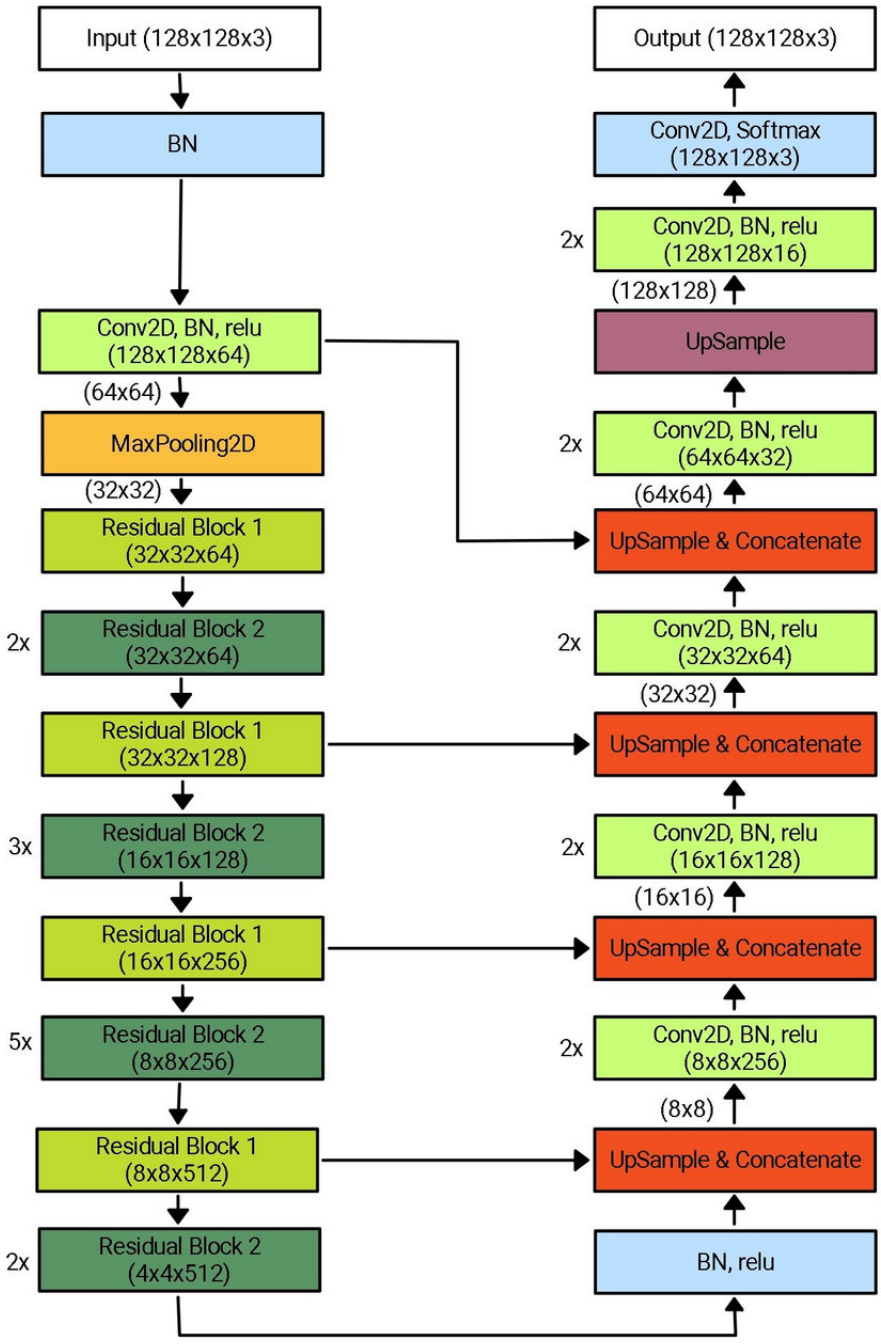

Figure 2.3: UNet architecture with a ResNet-34 encoder. Figure from [38].

Here's a general idea of how to construct a ResNet34-UNet architecture:

1. *Encoder (ResNet34)*: The encoder part is based on the ResNet-34 architecture, which is a specific variant of the ResNet architecture with 34 layers. It includes convolutional layers, batch normalization, and residual blocks. You can use pre-trained weights from a ResNet-34 model trained on a large dataset like ImageNet to initialize this part. The encoder's role is to extract features from the input image.

2. *Skip Connections*: Just like in a standard UNet architecture, you would add skip connections between corresponding layers in the encoder and decoder. These skip connections help preserve spatial information and assist in accurate segmentation. In ResNet-34, you can choose specific layers to establish these connections.

3. *Decoder (UNet)*: The decoder part follows the UNet architecture. It includes transposed convolutional layers (also known as "upconvolutions" or "deconvolutions") to progressively upsample the feature maps back to the original input image size. Skip connections are concatenated with the decoder's feature maps at each corresponding level to combine high-level and low-level information.

4. *Output Layer*: The final output layer of the network typically consists of one or more convolutional layers with softmax or sigmoid activation, depending on the specific task. For image segmentation, you might use a softmax activation with one channel per class to produce a probability map for each class.

In our experiment, we use dice coefficient loss to optimise the model.

2.3. MSRFNet

The Multi-Scale Residual Fusion Network (MSRFNet) [1] is capable of sharing multi-scale features with different receptive fields through the utilization of a Dual-Scale Dense Fusion (DSDF) block. The proposed DSDF block facilitates the rigorous exchange of information across two distinct resolution scales, and within our MSRF sub-network, we employ multiple DSDF blocks sequentially to

execute multi-scale fusion. This approach preserves resolution, enhances the flow of information, and propagates both high- and low-level features, resulting in the generation of precise segmentation maps. MSRFNet effectively captures object variations and consistently delivers improved results across various biomedical datasets.

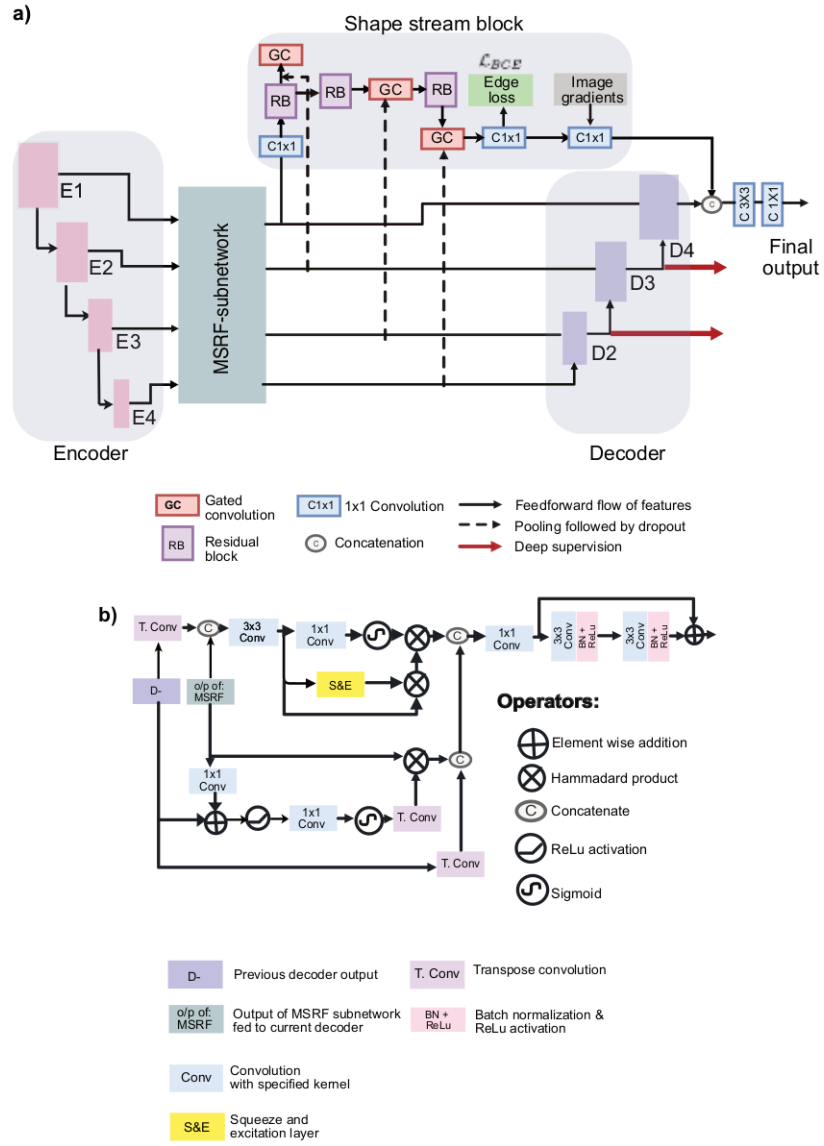

Figure 2.4: The proposed MSRFNet architectures. a) Overall block diagram of our network and b) overview of our decoder network. Figure from [1].

The MSRFNet architecture is comprised of several key components: an encoder block, the MSRF sub-network, a shape stream block, and a decoder block. Within the encoder block, you'll find squeeze and excitation modules. The MSRF

sub-network, on the other hand, is responsible for processing low-level feature maps obtained at each resolution scale of the encoder. Multiple DSDF blocks are integrated into the MSRF sub-network to facilitate this process. Following the MSRF sub-network, a gated shape stream is applied. In this architecture, the decoders are constructed using triple attention blocks. These triple attention blocks offer the advantage of employing spatial and channel-wise attention in conjunction with spatially gated attention, enabling the removal of irrelevant features from the MSRF sub-network. In the following sections, we will provide a brief overview of each component of our MSRF-Net.

2.3.1. Encoder

The encoder blocks (E1-E4) depicted in Figure 2.4(a) consist of a pair of consecutive convolution layers, followed by the incorporation of a squeeze and excitation SE module. The inclusion of the SE block in the network enhances its representational capacity by computing the interdependencies among the channels. During the ”squeezing” step, global average pooling is employed to consolidate feature maps across the spatial dimensions of the channels. In the ”excitation” step, a set of per-channel weights is generated to capture dependencies on a channel wise basis. In each stage of the encoder, downscaling of resolution is achieved using max pooling with a stride of two, and dropout is applied to facilitate model regularization.

2.3.2. The DSDF block and MSRF sub-network

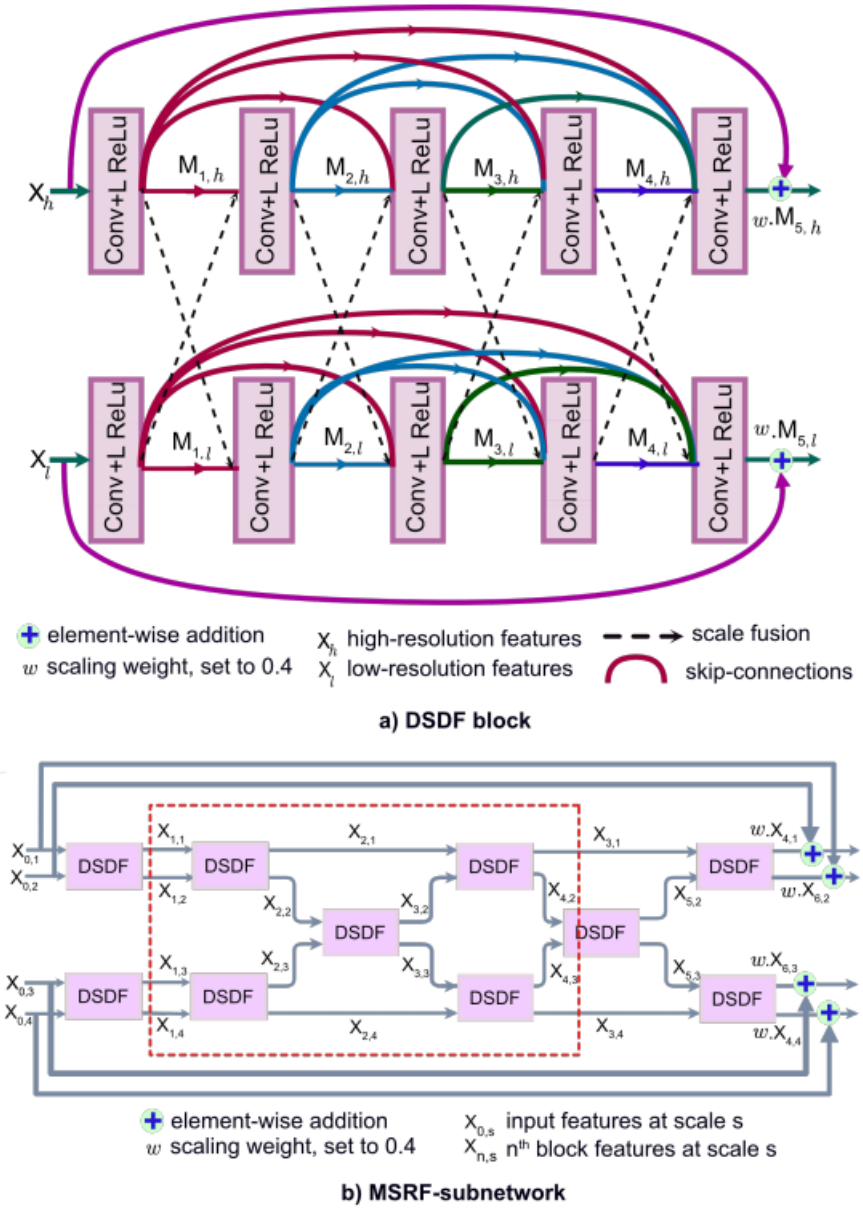

Figure 2.5: Components of our MSRF-Net, a) Proposed Dual-Scale Dense Fusion (DSDF) block and b) Multi-Scale Residual Fusion (MSRF). Dotted rectangle block in (b) represents multi-scale feature exchange in MSRF-Net. Figure from [1].

Ensuring consistent resolution throughout the feature encoding process plays a crucial role in enhancing the semantic richness and spatial accuracy of the target images. The DSDF block serves the purpose of facilitating information exchange between scales, preserving low-level features, and improving information flow while retaining the original resolution. This block consists of two parallel streams dedicated to different resolution scales, as depicted in Figure 2.5(a).

If we denote a 3×3 convolution operation followed by a LeakyReLU activation as $CLR(\cdot)$, each stream is equipped with a densely connected residual block comprising five consecutive CLR operations. The resulting feature map, denoted as $M_{d,h}$, after the d-th CLR operation, is computed from the high-resolution input X_h as follows:

$$M_{d,h} = CLR(M_{d-1,h} \oplus M_{d-1,l} \oplus M_{d-2,h} \oplus ... \oplus M_{0,h}), \tag{2.1}$$

where $\oplus$ is the concatenation operation, and h represents CLR operation is on the higher resolution stream of the DSDF block. $M_{d-1,l}$ is processed by a transposed convolutional layer with a 3×3 kernel size and stride of 2 before being concatenated. Similarly, for lower resolution stream the output of the $d - th\ CLR$ operation is denoted by $M_{d,l}$ and represented as:

$$M_{d,l} = CLR(M_{d-1,l} \oplus M_{d-1,h} \oplus M_{d-2,l} \oplus ... \oplus M_{0,l}) \tag{2.2}$$

In this context, $M_{d-1,h}$ undergoes processing via a convolutional layer with a 3×3 kernel size and a stride of 2 before undergoing concatenation. The variables d in Equation 2.1 and Equation 2.2 span the range from $1 \leq d \leq 5$.

At the outset, we designate X_h (or $M_{0,h}$) and X_l (or $M_{0,l}$) as the initial inputs for the high-resolution and low-resolution streams, respectively. Each CLR operation generates k output channels, which act as the growth factor regulating the quantity of new features that the layer can extract and propagate within the network. Given the variations in the growth factor across different scales, we opt to employ only

two scales simultaneously within the DSDF. This approach serves to mitigate the computational complexity of the model, making training more manageable.

Furthermore, we introduce local residual learning to bolster information flow while incorporating residual scaling as a preventive measure against instability. It's important to note that a scaling factor denoted by $0 \leq w \leq 1$ can be applied to control the extent of residual scaling. The final output of the DSDF block can be written as (see Figure 2.5(a)):

$$X_r = w \times M_{5,r} + X_r, \tag{2.3}$$

where $r \in [h, l]$ is the resolution with h indicating high-resolution representation and l for low resolution representation.

Next, we introduce an MSRF sub-network designed to leverage a dual-scale fusion mechanism in order to achieve a comprehensive global multi-scale context. This process involves handling all resolution scale pairs and feeding them into their corresponding DSDF blocks. Starting with the initial layer, each subsequent layer comprises four resolution scales, denoted as H and L to represent high-resolution and low-resolution feature sets, respectively. Each of these sets is further designated by the symbols $\hat{h}$ and $\hat{l}$.

The $DSDF(\cdot)$ function is responsible for merging features across scales within the DSDF block, where $X_{\hat{h},p}$ and $X_{\hat{l},q}$ are jointly computed from scale pairs p and q. Additionally, $\hat{X}$ signifies the feature exchange that occurs within the central DSDF. Remarkably, by the fourth layer of the MSRF sub-network, we have efficiently exchanged features across all scales, achieving global multi-scale fusion, as depicted in the red rectangular block in Figure 2.5(b).

It's worth noting that $X_{0,r}, \forall r \in \{1, 2, 3, 4\}$, is capable of transmitting its features to all parallel resolution representations through multiple DSDF blocks. This approach enables effective global feature exchange, even when the number of res-

olution scales exceeds 4. Similar to the DSDF block, the output of the last layer of the sub-network is scaled by the factor w and added back to the original input of the MSRF sub-network.

2.3.3. Shape stream

We have integrated the gated shape stream into the MSRFNet to facilitate shape prediction, as illustrated in the shape stream block in Figure 2.4(a). The DSDF blocks within the network play a pivotal role in extracting pertinent high-level feature representations that encompass vital information concerning shape and boundaries. These extracted features are subsequently employed within the shape stream.

We denote S_l as the feature maps of the shape stream, where l corresponds to the layer number, and X represents the output of the MSRF-sub-network. To ensure compatibility in spatial dimensions between X and S_l, we employ bilinear interpolation. Furthermore, we compute the attention map α_l at the gated convolution using the following expression:

$$\alpha_l = \sigma(Conv_{1\times1}(S_l \oplus X), \tag{2.4}$$

where $\sigma(\cdot)$ is the sigmoid activation function. Ultimately, we compute S_{l+1} as follows: $S_{l+1} = RB(S_l \times \alpha)$, where RB denotes a residual block consisting of two CLR operations, followed by the incorporation of a skip-connection.

The outcome of the shape stream is then combined with the image gradients derived from the input image. This amalgamation occurs just before the final CLR operation in the original segmentation stream. This step is undertaken to enhance the spatial precision of the resulting segmentation map.

2.3.4. Decoder

The decoder block $(D2 - D4)$ incorporates skip connections originating from both the MSRF sub-network and the previous decoder output (referred to as $D-$), with the exception of $D2$. In the case of $D2$, the connection from the previous layer is linked to the MSRF sub-network output of $E4$, as illustrated in Figure 2.4(a).

Within the decoder block (Figure 2.4(b)), we implement two distinct attention mechanisms. The first attention mechanism encompasses both channel and spatial attention, while the second attention mechanism utilizes a gating mechanism. To calculate channel-wise scale coefficients denoted as $X_{\alpha_{se}}$, we employ an SE block. Furthermore, spatial attention is computed in the same top stream, where the input channels (C) are condensed to a single channel using a 1×1 convolution operation. We apply the sigmoid activation function $\sigma(\cdot)$ to scale the values within the range of 0 to 1, generating an activation map. This map is then replicated C times, resulting in X_{α_s}. The output of the spatial and channel attention can be represented as:

$$D_{sc} = (X_{\alpha_s} + 1) \otimes X_{\alpha_{se}}, \tag{2.5}$$

where $\otimes$ denotes the Hadamard product and X_{α_s} is increased by a magnitude of 1 to amplify relevant features determined by the activation map. We also incorporate the attention-gated mechanism [40]. Suppose the features originating from MSRF-Net are denoted as X and the output from the preceding decoder block is represented as D^- In this context, the calculation of attention coefficients can be expressed as follows:

$$D_{AG} = \Omega(\sigma(\Psi(\theta(X) + \phi(D^-)))), \tag{2.6}$$

where $\theta(\cdot)$ is the convolution operation with stride 2, kernel size 1, and G channel outputs; $\phi(\cdot)$ is a convolution operation with stride 1 and kernel size 1×1 applied to D giving the same G channels; and $\Psi(\cdot)$ is convolution function with 1×1 kernel

size applied to a combined features from $\theta(\cdot)$ and $\phi(\cdot)$ making output channel equal to 1. Finally, $\sigma(\cdot)$ is applied to obtain the activation map on which transpose convolution operation $\Omega(\cdot)$ is applied. D_{AG} captures the contextual information and identifies the target regions and structures of the image. $\hat{D}_{AG} = D_{AG} \otimes X$ allows the irrelevant features to be pruned and relevant target structure and regions to be propagated further. $\hat{D}_{AG}$ is updated as:

$$\hat{D}_{AG} = \hat{D}_{AG} \oplus \Omega(D^{-}) \tag{2.7}$$

Now, the final output of the *triple attention decoder block* is $D_{\alpha} = D_{sc} \oplus \hat{D}_{AG}$, which is then followed by two CLR operations.

2.4. FCBFormer

FCBFormer [15] is a SOTA architecture for polyp segmentation in colonoscopy images which combines FCNs and transformers. The architecture uses two parallel branches which both start from a $h \times w$ input image: a fully convolutional branch (FCB) which returns full-size ($h \times w$) feature maps; and a transformer branch (TB) which returns reduced-size $\frac{h}{4} \times \frac{w}{4}$ feature maps. The output tensors of TB are then upsampled to full-size, concatenated with the output tensors of FCB along the channel dimension, before a prediction head (PH) processes the concatenated tensors into a full-size segmentation map for the input image. Next, we delve into the details of each part of the model.

2.4.1. Transformer branch (TB)

The transformer branch (TB) (Fig. 2.6b) is highly influenced by the state-of-the-art architecture for reduced-size polyp segmentation, the SSFormer [24]. Transformer encoder remains the original design of SSFormer, while the progressive locality decoder (PLD) introduced with SSFormer is improved by using the architecture shown in Fig. 2.6b (PLD+), where we use residual blocks (RBs) (Fig. 2.6f)

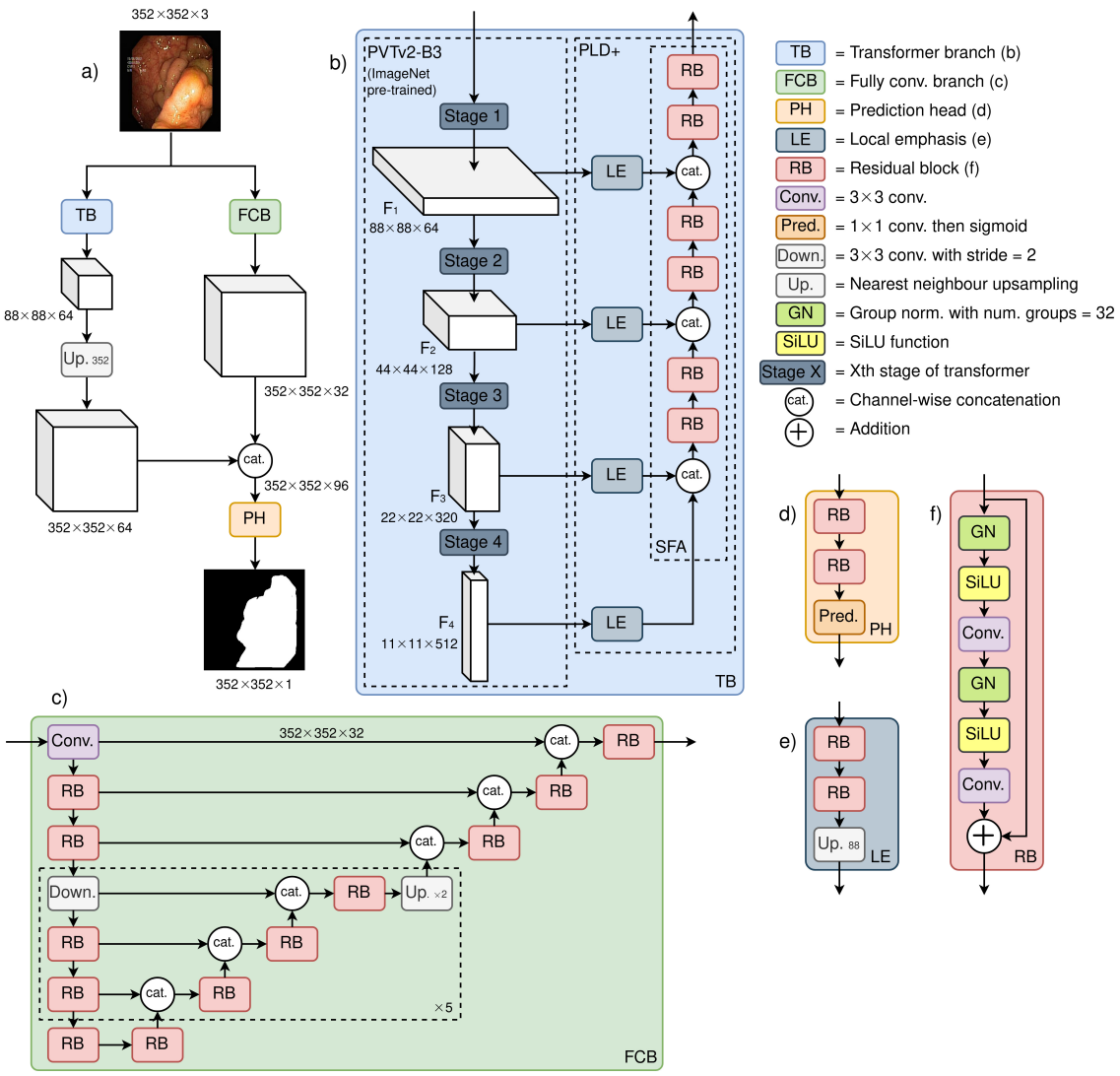

Figure 2.6: The architectures of a) FCBFormer, b) the transformer branch (TB), c) the fully convolutional branch (FCB), d) the prediction head (PH), e) the improved local emphasis (LE) module, f) the residual block (RB). Figure from [15].

to overcome identified limitations of the SSFormer's LE and SFA.

The design of these RBs draws inspiration from modern convolutional neural networks, which have shown significant performance improvements with the inclusion of elements like group normalization [54], SiLU activation functions [12], and residual connections [28]. In authors' analysis, they observed limitations in SSFormer's LE and SFA components, primarily due to their lack of these modern elements and a relatively small number of layers. Consequently, we made adjustments to these elements in FCBFormer to create the building blocks of PLD+. We have demonstrated the positive impact of these modifications through ablation

tests in the experimental section of this paper.

In a manner similar to SSFormer, the first convolutional layer in the LE blocks of FCBFormer returns 64 channels. Subsequent layers, with the exception of channel-wise concatenation, also return the same number of channels (64).

2.4.2. Fully convolutional branch (FCB)

In this architecture, FCB is composed of residual blocks (RBs), strided convolutional layers for downsampling, nearest neighbor interpolation for upsampling, and dense UNet-style skip connections (as shown in Fig. 2.6c). This design enables the extraction of highly fused multi-scale features at full size. When combined with the significant yet coarse features extracted by the TB, it allows us to predict full-size segmentation maps in the prediction head (PH).

To achieve this, within the FCB's encoder, we increase the number of channels returned by each layer by a factor of 2 in the first convolutional layer following the second and fourth downsampling layers. In contrast, within the FCB's decoder, we decrease the number of channels returned by each layer by a factor of 2 in the first convolutional layer following the second and fourth upsampling layers.

2.4.3. Prediction head (PH)

The up-sampled TB output and the output obtained from the FCB are combined prior to being processed in the PH, as illustrated in the Fig. 2.6d. The PH is responsible for generating the segmentation map by merging the essential yet coarse features extracted by TB with the finely detailed features extracted by FCB. Each layer of PH returns 64 channels, except the prediction layer which returns a single channel.

The loss function used was the sum of the BCE loss and the Dice loss.

2.5. Conclusion

In conclusion, this chapter has reviewed various works on medical image segmentation that are based on the U-Net architecture. The UNet has been widely used for medical image segmentation tasks due to its excellent performance and ability to handle different types of medical images. The reviewed works have shown that modifications to the original UNet architecture can lead to further improvements in segmentation accuracy and robustness. Some of the modifications include the addition of attention modules, residual connections, and recursive layers, as well as the use of various loss functions such as Dice loss and BCE loss. Additionally, cutting-edge techniques such as DSDF, shape stream or PLD+ have been used to improve the segmentation performance of the UNet-based models. Despite the success of UNet-based models in medical image segmentation, there is still room for improvement, particularly in addressing challenges such as handling imbalanced datasets and incorporating domain knowledge into the models.

Chapter 3
PROPOSED METHOD

3.1. ESFPNet Architechture

3.1.1. MixTransformer Encoder

CNN-based encoders, such as the Unet variants, have achieved significant success in image segmentation tasks. A CNN-based encoder, driven by the notion that each image pixel relies on its neighboring pixels, employs filters over an image patch to extract relevant local features. Yet, if a processing model utilized all image data instead of only the patches considered by the filters, then processing performance would be expected to improve. This concept is demonstrated in the case of Vision Transformers (ViT) [32]. The Mix Transformer (MiT) encoder [17] is a module built upon the ViT [32] network concept, utilizing four modules that combine diagonal pathways and self-attention across four stages. These stages not only offer high-resolution coarse features but also deliver fine-grained features with lower resolution. Furthermore, the high- and low-resolution features are commonly employed to enhance semantic segmentation performance.

The MiT encoder block contains these components:

Overlapped Patch Merging: In the Vision Transformer [32], the image is divided into non-overlapping patches (16x16 pixels). However, in the Mix Transformer encoder [17], the image is divided into overlapping patches. The author explains that this overlapping patch division is done to ”preserve local positional information between patches”. In other words, the Transformer network used in SegFormer [17] does not employ positional encoding like other typical Transformer networks. The patch division, essentially achieved through Convolution

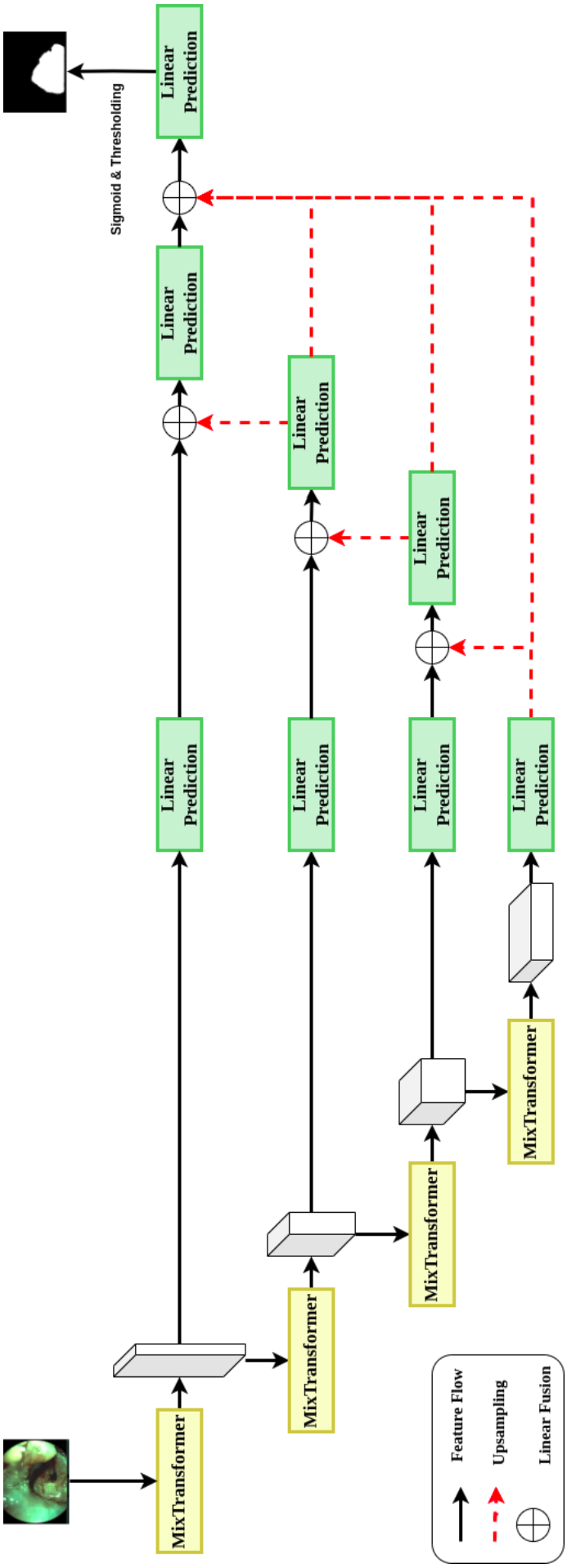

Figure 3.1: Block diagram of the ESPFNet architecture. ESFPNet utilizes the Mix Transformer (MiT) encoder as the backbone and uses an efficient stage-wise feature pyramid (ESFP) as the decoder to generate segmentation outputs.

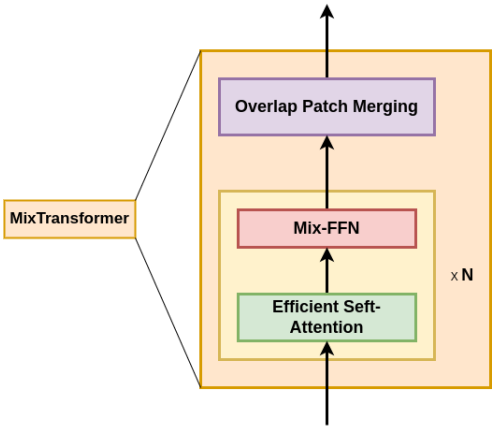

Figure 3.2: Components in Mix Transformer.

operations, aims to enhance the positional information between patches in the image. The process of dividing the image into overlapping patches is defined by three parameters: K (patch size), S (stride - the distance between adjacent patches), and P (padding). Generating these overlapping patches involves the use of Convolution operations (readers can refer to the author's implementation for details). In the experiments, the author sets $K = 7$, $S = 4$, $P = 3$ for Block 1 and $K = 3$, $S = 2$,$P = 1$ for Blocks 2, 3, and 4.

Efficient Self-Attention: The computation of self-attention, as known, will involve multiple heads. Each head has three matrices: Q, K, V, all of size (N, C), with $N = W \times H$ being the length of the sequence when inputted into the Transformer. The formula for calculating self-attention is as follows:

$$Attention(Q, K, V) = Softmax(\frac{QK}{\sqrt{d_{head}}})V \tag{3.1}$$

The computational cost will be $O(N^2)$, which becomes impractical for high-resolution images. The proposed solution to address this is to use a reduction factor denoted as R to reduce the length of the sequence:

$$\hat{K} = Reshape(\frac{N}{R}, C \cdot R)(K) \tag{3.2}$$

$$K = Reshape(C, C \cdot R)(\hat{K}) \tag{3.3}$$

In the experiments, R is set to $64, 16, 4, 1$ for the respective 4 blocks.

Mix-FFN: Vision Transformer uses positional encoding to retain information about the position of each pixel in the image. However, the size of positional encoding is usually fixed. When training and testing images have different sizes, an interpolation algorithm is required to adjust the positional encoding size to match the image size. Adding positional encoding in such a ”non-uniform” way unintentionally introduces noise to the input image and reduces the model's performance. SegFormer [17] employs Mix-FFN, combining Convolution with a Feed Forward Network, without the need for positional encoding. The formula is as follows:

$$x_{out} = MLP(GELU(Conv_{3x3}(MLP(x_{in})))) + x_{in} \tag{3.4}$$

where x_{in} is the feature from the self-attention module. Note that, depth-wise convolutions is used for reducing the number of parameters and improving efficiency

Although the modified block's architecture brings efficiency to the segmentation task,, the limitations of employing transformer-based encoders are evident. The self-attention layers used by transformers lack local inductive bias (the idea that image pixels are locally correlated, and their correlation maps are invariant to translation), leading to the problem of data hunger. To alleviate the data hunger challenge in applications constrained by small datasets, the widely adopted concept of transfer learning can be leveraged. MiT encoders [17], capitalizing on this idea, are pretrained on the extensive ImageNet database [22]. For our ESPFNet [43] architecture, we integrate these pretrained MiT encoders [17] as the backbone and retrain them with initialized decoders. This approach has proven to be a direct way to achieve strong performance on specific task-related datasets while still being capable of surpassing the performance of state-of-the-art CNN models.

3.1.2. Efficient stage-wise feature pyramid (ESFP) decoder

The prediction outcomes of the decoder are built upon multi-level features from the encoder, where local (low-level) features are extracted from the shallow parts of the encoder, while global (high-level) features are extracted from the deeper portions. Previous study [36] have demonstrated that the sufficiency of local features obtained in the shallow section of the transformer directly impacts the model's performance. However, the current Segformer [52] model aggregates these multi-level features equally to predict segmentation results, thus lacking the ability to adequately and selectively utilize the local features.

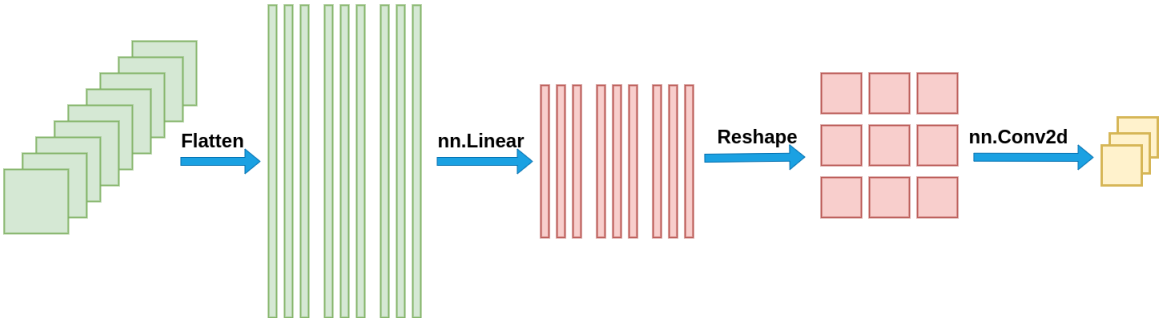

Figure 3.3: Linear Prediction is typically a part of a neural network and often consists of several linear layers.

Inspired by the structure of a lightweight channel-wise feature pyramid network (CfpNet) [9] that combines each feature pair and concatenates multi-level fused components for the final prediction, the architecture of the multi-stage feature pyramid (ESFP) [43] is introduced to efficiently exploit multi-stage features. ESFP [43] commences with linear predictions of each stage's output (efficient in terms of the number of connecting channels) and then linearly fuses these preprocessed features from global to local. These intermediate aggregated features are concatenated and collaborate to generate the final segmentations.

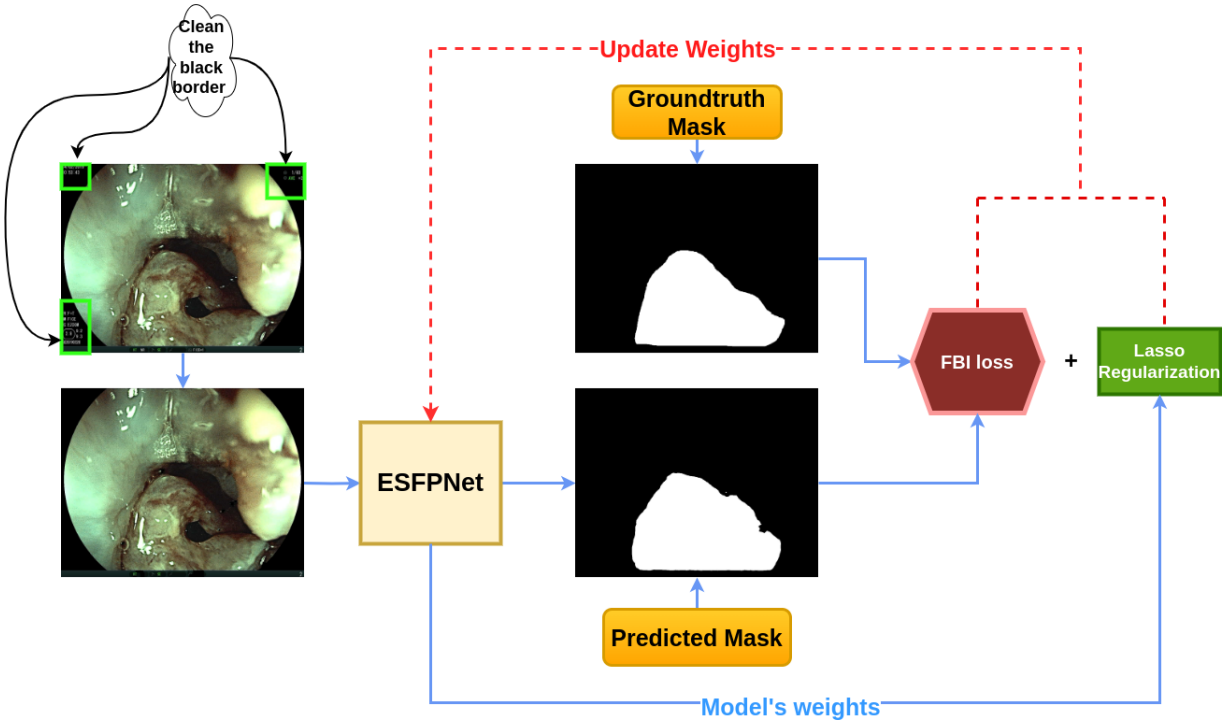

Figure 3.4: Proposed Strategy for Lesion Segmentation.

3.2. Training Strategy

3.2.1. Remove redundant endoscope device's information

The input images often contain redundant information (typically settings of the endoscope device) in the image corners. This somewhat affects the quality of the input images, so we propose removing this information using a threshold-based method. Specifically, the following is a description of the steps to remove the redundant endoscope device's information:

Conversion to Hue, Saturation, Value: Conversion to HSV automates black mask detection in colonoscopic images by converting RGB images to the HSV color model. This conversion streamlines color specification, vital for subsequent thresholding.

Channel V thresholding: Once the RGB to HSV conversion is complete, the image is prepared for thresholding. Thresholding allows for the selection of intensity values, making it possible to identify the objects to be automatically detected. In this context, channel V thresholding is employed, assigning values of 0.03 and

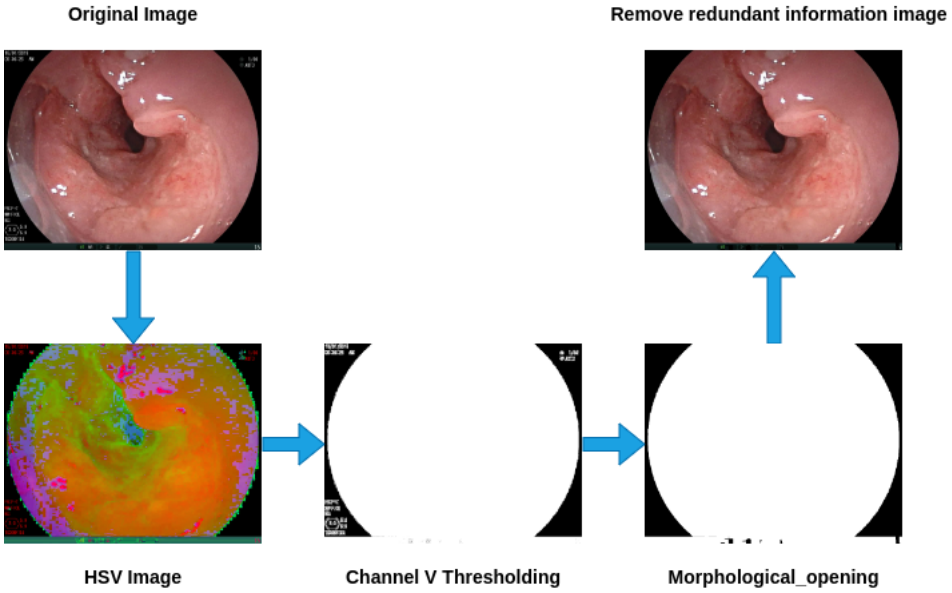

Figure 3.5: Removal of redundant information.

lower to the black mask. This approach effectively separates the relevant content within the colonoscopic image from the black borders.

Remove redundant information: Removing Redundant Information involves morphological operations on binary images using a kernel. ”Opening,” an erosion-dilation combo, reduces noise, corrects imperfections, and diminishes white object prominence. A final *bitwise-and* operation applies the cleaned mask to the original image for the desired result.

3.2.2. Focal-BCE-IoU Loss Function

The ESFPNet model [43] is trained by the combined loss function with two terms, i.e., BCE and IoU. However, the drawback of this loss function it is less effective with imbalanced datasets which include both large-sized and small-sized lesion areas. To handle the above issue of ESFPNet, we introduce the Focal-BCE-IoU (FBI) loss function which has three terms: BCE, IoU, and Focal loss term.

The IoU loss is a commonly used loss function in image segmentation tasks. It quantifies the dissimilarity between predicted and ground truth masks by measuring the spatial overlap between them and aims to maximize the intersection over union ratio, which is an important metric for evaluating segmentation mod-

els. Here, we apply the weighted IoU loss is defined as following equation.

$$\mathcal{L}_{IoU}^{w} = 1 - \frac{\sum_{i=1}^{N}(we_i \times y_i \times p_i)}{\sum_{i=1}^{N} we_i \times (y_i + p_i - y_i \times p_i)}, \tag{3.5}$$

where N is the number of pixels in the image, y_i is value of pixel i in the ground truth mask, p_i is the predicted value of pixel i belonging to the lesion area. The weight matrix we is calculated based on the ground truth mask. It is the difference between the mask before and after smoothed by applying average pooling operation [42]. This aims to guide the loss focusing on more important regions during the training process [8, 6, 7].

The BCE loss is a common loss function used in image segmentation tasks to segment two class. The computation of this loss is based on cross-entropy which is used to measure the difference between two probability distributions. Thus, the BCE loss measures the differences in information content between two classes, e.g., the ground truth masks and predicted masks. It is effective with equal data-distribution between classes. We also apply the weighted BCE loss that is calculated as follows:

$$\mathcal{L}_{BCE}^{w} = -\frac{1}{\sum_{i=1}^{N} we_i} \sum_{i=1}^{N} we_i \times [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)], \tag{3.6}$$

where the definitions of N, we, y_i, and p_i are similar to those in the IoU loss defined in Eq. 3.5.

The Focal loss is a specialized loss function that is introduced to address the issue of class imbalance due to the various sizes of lesion areas. This loss emphasizes the hard examples, e.g., the image with small sized lesion area, by assigning different weights to easy and hard examples. Specifically, in training a classifier, hard or misclassified examples are assigned higher weights while down-weighting easy examples. To focus on the important regions, we unitize the weighted Focal

loss is defined as follows:

$$\mathcal{L}^{w}_{\text{Focal}} = -\frac{1}{\sum_{i=1}^{N} we_i} \sum_{i=1}^{N} we_i \times \left[y_i (1-p_i)^{\gamma} \log(p_i) + (1-y_i) p_i^{\gamma} \log(1-p_i) \right], \tag{3.7}$$

where N, we, y_i, and p_i are defined as in Eq. 3.5 , γ is a hyperparameter which controls the amount of reducing the loss for well-classified examples. A higher γ values put more emphasis on hard examples.

Our proposed loss function is the combination of the BCE, IoU, and Focal losses to enhance the lesion segmentation problem with both large and small-sized lesion areas. This computation is presented in Eq. 3.8.

$$\mathcal{L}_{FBI} = \frac{1}{N} \times (\mathcal{L}^{w}_{Focal} + \mathcal{L}^{w}_{BCE} + \mathcal{L}^{w}_{IoU}), \tag{3.8}$$

where $\mathcal{L}^{w}_{Focal}$, $\mathcal{L}^{w}_{BCE}$, and $\mathcal{L}^{w}_{IoU}$ are defined in Eq. 3.7, Eq. 3.6, and Eq. 3.5, respectively. Using our proposed loss function helps ESFPNet reducing the importance of easy data samples and emphasizes learning from more challenging ones. This ensures the model focuses on learning diverse and complex regions. As a result, the accuracy of ESFPNet for lesion segmentation is increased.

3.2.3. Lasso Regularization

Lasso regularization is introduced as a regularization term in loss functions to reduce over-fitting [56]. It is calculated as Eq. 3.9

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{(\gamma \in \Gamma)} |\gamma|, \tag{3.9}$$

where x and y are the input and output of the training set, W denotes the trainable weights, λ controls the trade-off between the normal training loss l. The value of λ in our experiment is $1e-6$. Lasso regularization γ represents the regularization term applied to the model's parameters (in this case, the trainable weights W)

to prevent overfitting, hence, Γ represents the set of parameters or weights that are being regularized. Eq. 3.9 is known as "sparse training" because the value of γ tends to zero during the training process. After sparse training, channels with near-zero factors can be pruned by removing all their incoming and outgoing connections and corresponding weights, resulting in efficient network structures. Therefore, we leverage the Lasso regularization as to mitigate over-fitting during the model training process.

3.2.4. Lesion Segmentation Model

Our proposed model is built based on the ESFPNet model, which is introduced in Section 3.1. The input image is initially preprocessed to remove redundant information of the endoscopic equipment on the image by threshing the black corners. These are black borders shown in Fig. 3.1. Subsequently, it is resized to a fixed dimension, e.g., $352x352x3$, before being fitted into the ESFPNet backbone. The backbone network is responsible for extracting both shallow (local) and deep (global) features from the input image.

The training process of the ESFPNet is illustrated in Fig. 3.1. For each input image, the ESFPNet model generates a prediction mask. The FBI loss function is computed using this prediction mask and the ground truth mask. This computation is executed for all input samples within the training batch size of the dataset. Additionally, the Lasso regularization term is added to the loss function to encourage model sparsity by constraining the weights of uninformative features to zero. The input of Lasso is the model's weights. The resulting loss value and the Lasso value are used optimization the weights of the ESFPNet network by the gradient back-propagation algorithm.

In the testing and predicting processes, each input image is preprocessed as described above. Then, it inputs to the ESFPNet model to get the prediction mask. The prediction mask presents the lesion and normal (background) area by the white

pixels and the black pixels, respectively.

3.3. Conclusion

In this chapter, we introduce the proposed method. Specifically, we discuss preprocessing techniques aimed at removing redundant information from the endoscopy device positioned in various image angles. Subsequently, we present the proposed loss function, namely the FBI loss, and sparsity techniques, referred to as Lasso regularization, in our training strategy to enhance lesion segmentation performance. Following that, we describe the lesion segmentation model with the ESFPNet backbone trained using the proposed loss function.

Chapter 4
EXPERIMENTS AND SYSTEM SETUP

4.1. Datasets

4.1.1. Quantitative aspect

The dataset used in this study comprises five endoscopic image datasets, including Gastric cancer, Esophageal cancer, Peptic ulcer, Positive H. pylori gastritis, and Negative H. pylori gastritis. Figures 4.1, 4.2, 4.3, 4.4 and 4.5 illustrate these datasets, respectively, showing both the input images and their corresponding masks. General information about the dataset is presented in Table 4.1.

Dataset	**Total**	**Format**	**Image Size**
Gastric cancer	534	image: .JPEG/mask: .PNG	1280x995
Esophageal cancer	538	image: .JPEG/mask: .PNG	1280x995
Peptic ulcer	1159	image: .JPEG/mask: .PNG	1280x1024 1280x995 1280x959 1280x958
Positive H. pylori gastritis	2342	image: .JPEG/mask: .PNG	1280x994 1280x964 1280x960 1280x959 1280x958
Negative H. pylori gastritis	2038	image: .JPEG/mask: .PNG	1280x994 1280x964 1280x960 1280x959 1280x958

Table 4.1: Data description.

We divided the data into three sets: training, validation, and testing, with an 8:1:1 ratio, respectively. The specific number of samples in each set for each dataset is detailed in Table 4.2.

Dataset	**Train**	**Valid**	**Test**
Gastric cancer	427	53	54
Esophageal cancer	430	53	55
Peptic ulcer	927	115	117
Positive H. pylori gastritis	1873	234	235
Negative H. pylori gastritis	1630	203	205

Table 4.2: The specific number of samples.

4.1.2. Qualitative aspect

As can be observed, in the two datasets of endoscopic cancer images, the areas of lesions occupy a significant portion and are mostly confined to specific regions within the input images. On the other hand, for the datasets related to inflammation (colon ulcer and gastritis), the distribution of lesion regions is much more intricate. They can either cluster together in one location or scatter across various areas within the input images, exhibiting different sizes. The varying complexity of these lesion distributions in the images has a profound impact on the experimental outcomes.

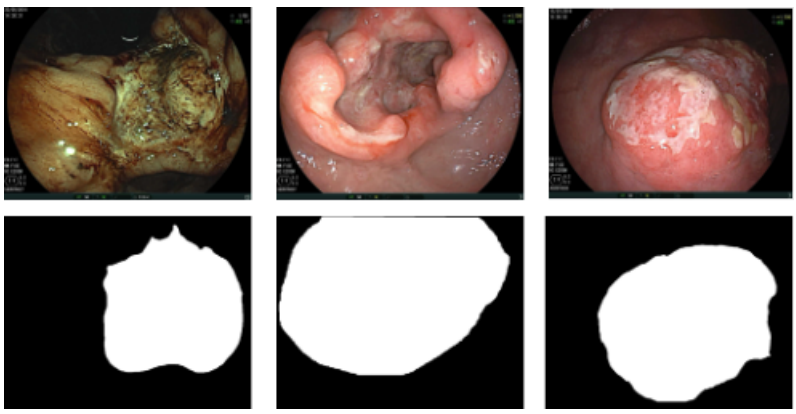

Figure 4.1: Gastric cancer samples.

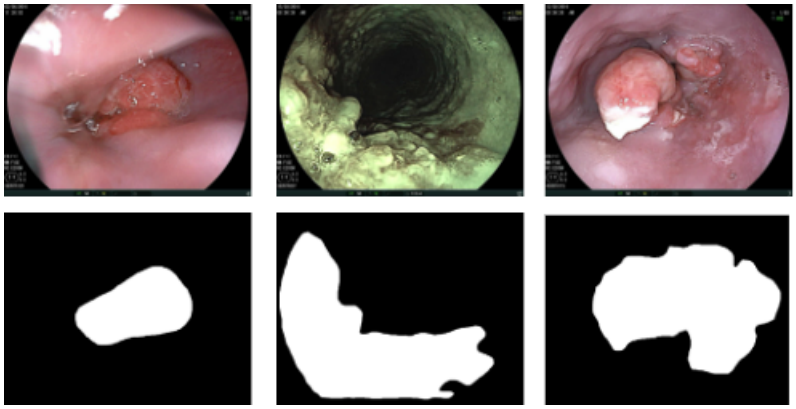

Figure 4.2: Esophageal cancer samples.

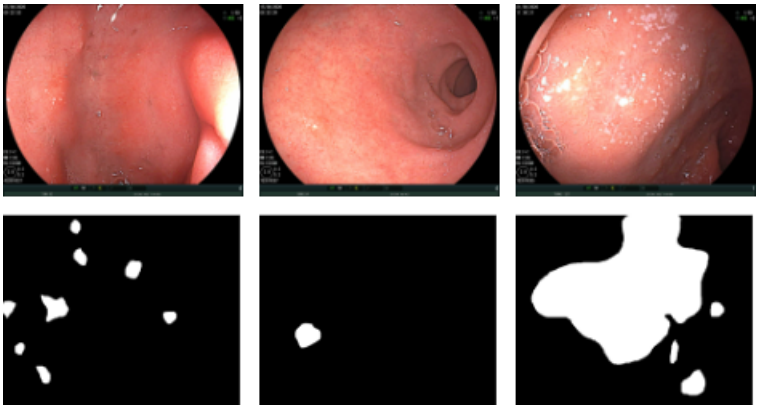

Figure 4.3: Peptic ulcer samples.

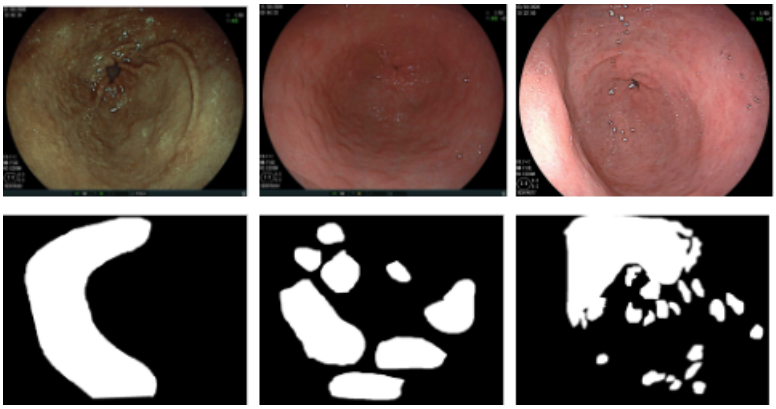

Figure 4.4: Positive H. pylori gastritis samples.

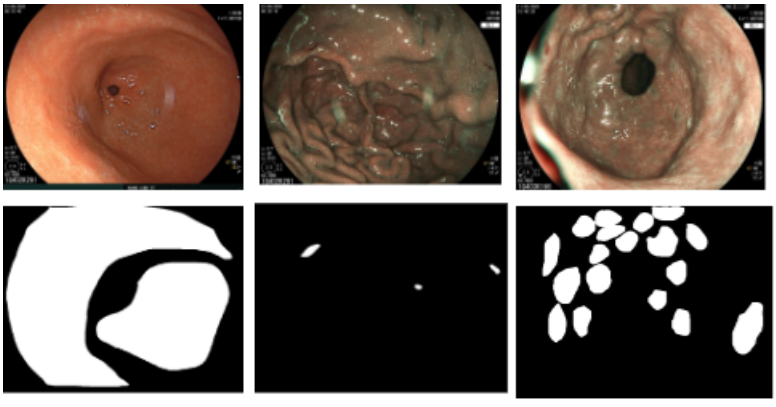

Figure 4.5: Negative H. pylori gastritis samples.

Note that every image capturing process is influenced in some way by factors that degrade the image quality to a certain extent. Colonoscopic imaging is no exception. It can be easily observed that our input images contain many noisy elements such as air bubbles, areas dazzled by light, or even regions obscured by endoscopic equipment. Here are some factors that influence the quality of endoscopic images:

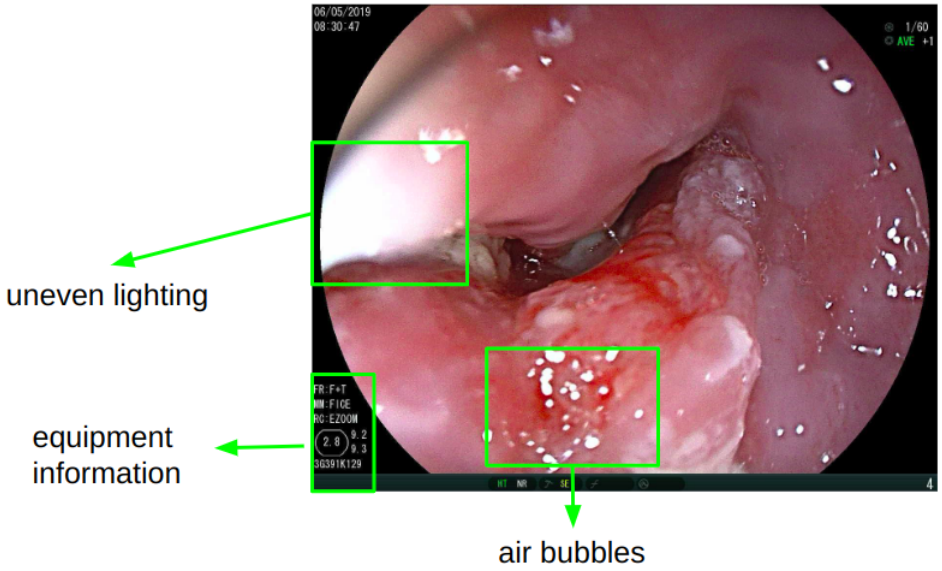

Figure 4.6: Device information in black corners, specular highlights and uneven lighting in endoscopic image.

Black mask: This stems from the fact that the lenses used in the colonoscopy image capture system have a black frame around the edge. In many cases, this mask is used to convey information, either related to the patient or the ongoing procedure. This black frame obstructs the development of digital image processing algorithms because it introduces false borders and covers a larger area for analysis that provides no useful information. For these reasons, applying various techniques to mitigate its effects becomes necessary. In Figure 4.6, the presence of a black mask in colonoscopy imaging can be observed. Please note that the black mask region contains meta-information about the examination time, equipment parameters, and so forth. These factors also somewhat influence the results of our image segmentation.

Specular highlights: Specular highlights (see Figure 4.6) are points of high intensity in the image due to the illumination of shiny objects. When a light source is directed onto an object, the light is reflected and captured by the camera. This process generates highly saturated areas in the image, which can result in unwanted outlines, subsequently complicating image processing. This effect is particularly significant in detecting polyps, which are generally round and resemble tumors. Due to their shape, they reflect light and generate specular highlights when illuminated, which can lead to algorithm malfunction.

Uneven lighting: Variations in the intensity and direction of lighting play a crucial role in the appearance of objects in digital images. The illumination of the colon during colonoscopy is variable, which, due to the colon's three-dimensional shape, creates shadows that accentuate or diminish certain aspects of the image. Varied illumination levels on the same object lead to different object representations, introducing undesirable lighting variability. The literature contains numerous publications addressing this issue. In Figure 6, an example of uneven lighting in colonoscopic imaging is provided to facilitate its detection.

4.2. Implementation Details

We implemented the model in Pytorch and accelerated training via NVIDIA GPUs. We trained these networks either on an NVIDIA RTX 3060. Before training, we removed the endoscope's parameters at the borders and corners of the input image, then resized the inputs to 352×352 pixels and normalized them for segmentation. We also employed random flipping, rotation, and brightness changing as data augmentation operations on the inputs. Our loss function is FBI loss. We used the default AdamW optimizer with the learning rate $1e-4$ and trained our models for 200 epochs.

4.3. Evaluation

4.3.1. Quantitative results

In case of Gastric cancer dataset, among the various models evaluated, our model stands out in terms of mDice, mIoU, and mRecall, achieving the highest values of 0.9107, 0.8625, and 0.9221, respectively. These metrics reflect our model's superior ability to accurately segment the target regions. However, it's worth noting that our model's precision is relatively lower compared to some other models, possibly indicating a trade-off between precision and recall. Nevertheless, the overall performance, especially in terms of mDice and mIoU, demonstrates the effectiveness of our approach in this segmentation task.

Metric	mDice	mIoU	mRecall	mPrecision
ResUNet++ [14]	0.7634	0.7233	0.7971	0.7908
ResNet34-UNet [38]	0.8073	0.6945	0.7960	0.8700
MSRFNet [1]	0.4194	0.3165	0.5282	0.4412
FCBFormer [15]	0.4644	0.3310	0.4216	0.6177
ESFPNet [42]	0.9103	0.8622	0.9177	**0.9029**
Ours	**0.9107**	**0.8625**	**0.9221**	0.8086

Table 4.3: Results of all models on the Gastric cancer dataset.

Next, in case of Esophageal cancer dataset, comparing the our proposed method to other architectures, it is evident that ours achieves superior results in terms of mDice and mIoU, with values of 0.8654 and 0.8332, respectively. These metrics indicate the effectiveness of our model in accurately delineating regions of interest in the images. Additionally, our method also excels in mPrecision, with values of 0.9084, respectively, showcasing its proficiency in correctly identifying and distinguishing between lesion regions and non-lesion regions. It's neccesary to note that the mRecall metric for our method is slightly lower than some other models, indicating that it may miss a small fraction of lesion regions. However, this trade-off between precision and recall can be adjusted depending on the specific

requirements of the segmentation task.

Metric	mDice	mIoU	mRecall	mPrecision
ResUNet++ [14]	0.7157	0.7018	0.7581	0.7495
ResNet34-UNet [38]	0.7204	0.5894	0.6667	0.8405
MSRFNet [1]	0.3216	0.2469	0.4248	0.3711
FCBFormer [15]	0.4285	0.3516	0.4592	0.4017
ESFPNet [42]	0.8633	0.8291	**0.8479**	0.8794
Ours	**0.8654**	**0.8332**	0.8262	**0.9084**

Table 4.4: Results of all models on the Esophageal cancer dataset.

The table 4.5 summarizes the performance of various models on the Peptic ulcer dataset. In terms of metrics, our proposed model outperforms the mDice, mRecall, mPrecision scores as 0.7005, 0.6973 and 0.7038, respectively, compared with the previous models. This signifies its outperformed segmentation capabilities in identifying the lesion regions. However, the mIoU value of our proposed model is slightly lower than that of the ESFPNet model as 0.003.

Metric	mDice	mIoU	mRecall	mPrecision
ResUNet++ [14]	0.3912	0.5870	0.3812	0.5641
ResNet34-UNet [38]	0.4266	0.2992	0.4715	0.5436
MSRFNet [1]	0.1089	0.0815	0.1631	0.1402
FCBFormer [15]	0.2615	0.3901	0.1722	0.5433
ESFPNet [42]	0.6892	**0.7239**	0.6966	0.6820
Ours	**0.7005**	0.7236	**0.6973**	**0.7038**

Table 4.5: Results of all models on the Peptic ulcer dataset.

In the table of Positive H. pylori gastritis dataset, our proposed model outperforms the other models in terms of mDice and mIoU scores, achieving values of 0.5227 and 0.6275, respectively. These metrics indicate the effectiveness of our model in accurately segmenting images. While our model shows competitive mRecall scores, it excels in mPrecision, with a score of 0.5783, showcasing its ability to correctly identify non-lesion regions. This demonstrates the superior performance of our model in comparison to the alternatives.

Metric	mDice	mIoU	mRecall	mPrecision
ResUNet++ [14]	0.0665	0.4097	0.0571	0.1865
ResNet34-UNet [38]	0.3723	0.2766	0.3500	0.4886
MSRFNet [1]	0.2023	0.1979	0.2782	0.4068
FCBFormer [15]	0.4051	0.3963	0.6189	0.2855
ESFPNet [42]	0.5141	0.6157	**0.6694**	0.5423
Ours	**0.5227**	**0.6275**	0.6178	**0.5783**

Table 4.6: Results of all models on the Positive H. pylori gastritis dataset.

Finnaly, with Negative H. pylori gastritis dataset, the proposed technique outperforms most others in mDice, mIoU, and mRecall, with scores of 0.4367, 0.5360, and 0.5194, respectively. This indicates our model's proficiency in accurately segmenting target regions. While the ESFPNet model excels in mPrecision, at 0.9977, the significant increase in mPrecision while mDice, mIoU, and mRecall are low in the task of injury segmentation often indicates that this model tends to generate many false positives. This means that the model is detecting many non-injured regions as injuries.

Metric	mDice	mIoU	mRecall	mPrecision
ResUNet++ [14]	0.0058	0.4153	0.0072	0.0997
ResNet34-UNet [38]	0.2909	0.2180	0.2390	0.8898
MSRFNet [1]	0.1232	0.1321	0.2267	0.2403
FCBFormer [15]	0.2294	0.1617	0.2243	0.5275
ESFPNet [42]	0.2541	0.1456	0.1456	**0.9977**
Ours	**0.4367**	**0.5360**	**0.5194**	0.5086

Table 4.7: Results of all models on the Negative H. pylori gastritis dataset.

In general, it can be observed that for both cancer datasets (comprising esophageal and gastric cancers), where the contrast between the affected and normal regions is not significantly pronounced, the proposed model demonstrates better learning capabilities from the input data. However, for the three datasets related to peptic ulcers and gastritis, due to the high complexity and imbalance in the input data, as described in Section 4.1, although our method yields better results compared to existing methods, the output outcomes are still not exceptionally

impressive, and there are several avenues for further development.

4.3.2. Qualitative results

From the visual results, it can be observed that our proposed model outperforms previous models in image segmentation. Specifically, for two datasets, esophageal cancer and gastric cancer, the two models using the ESFPNet backbone yield superior results compared to other models such as MSRFNet or FCBFormer and perform better than architectures based on ResNet. Our method demonstrates its effectiveness in effectively capturing regions at the periphery of the lesions.

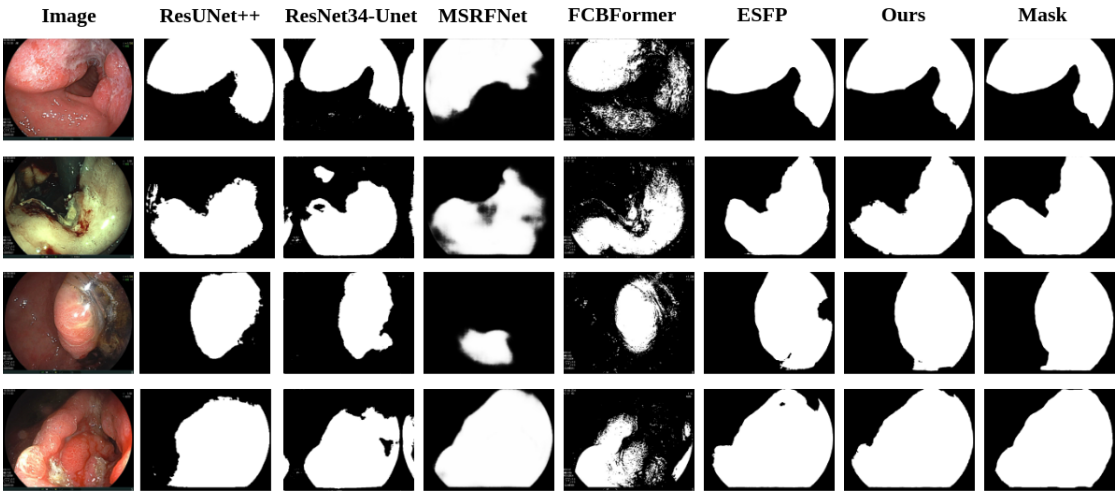

Figure 4.7: The Figure shows qualitative results of the proposed model in comparison with previous studies on Gastric cancer dataset.

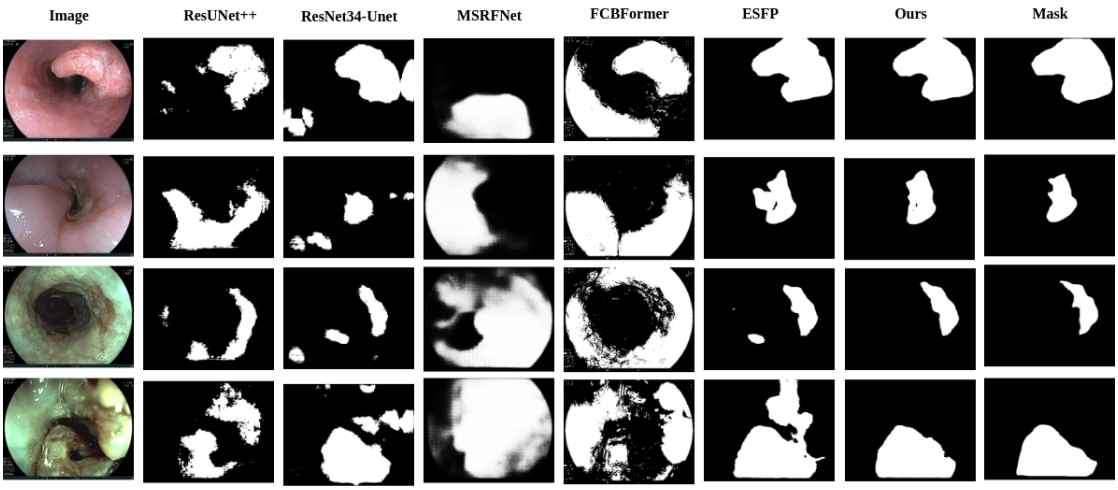

Figure 4.8: The Figure shows qualitative results of the proposed model in comparison with previous studies on Esophageal cancer dataset.

For the remaining three datasets, due to their complexity and lack of consistency, all models faced challenges when dealing with the problem of segmenting damaged regions in endoscopic images. MSRFNet and FCBFormer are two state-of-the-art architectures developed recently; however, through experiments, it becomes evident that they are not suitable for these datasets. Conversely, simpler architectures prove to be more effective. Among the surveyed models, our method yields the most promising results, though these results are not yet truly satisfactory, and there is still room for further improvement.

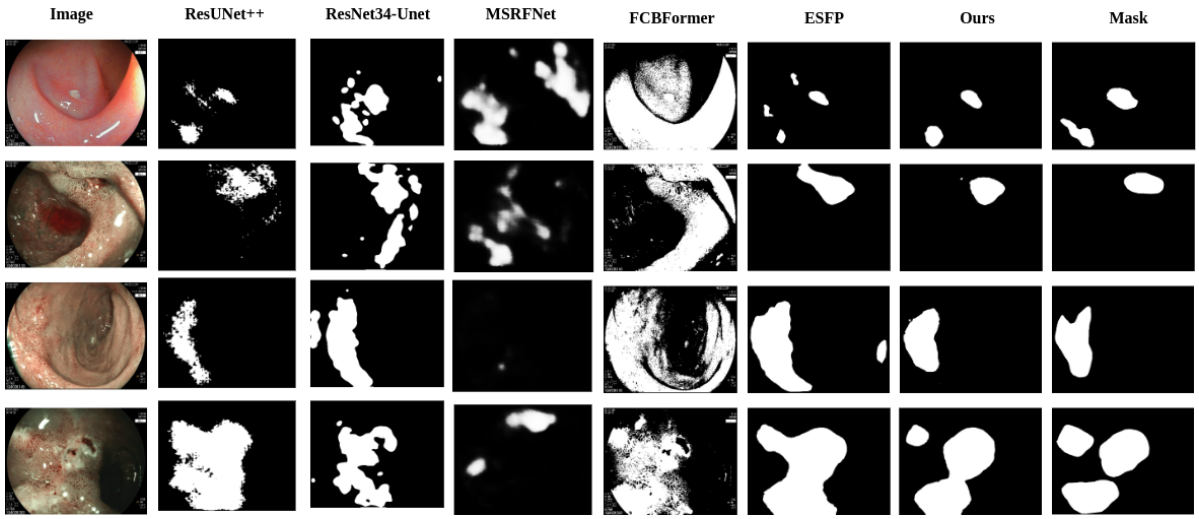

Figure 4.9: The Figure shows qualitative results of the proposed model in comparison with previous studies on Peptic ulcer dataset.

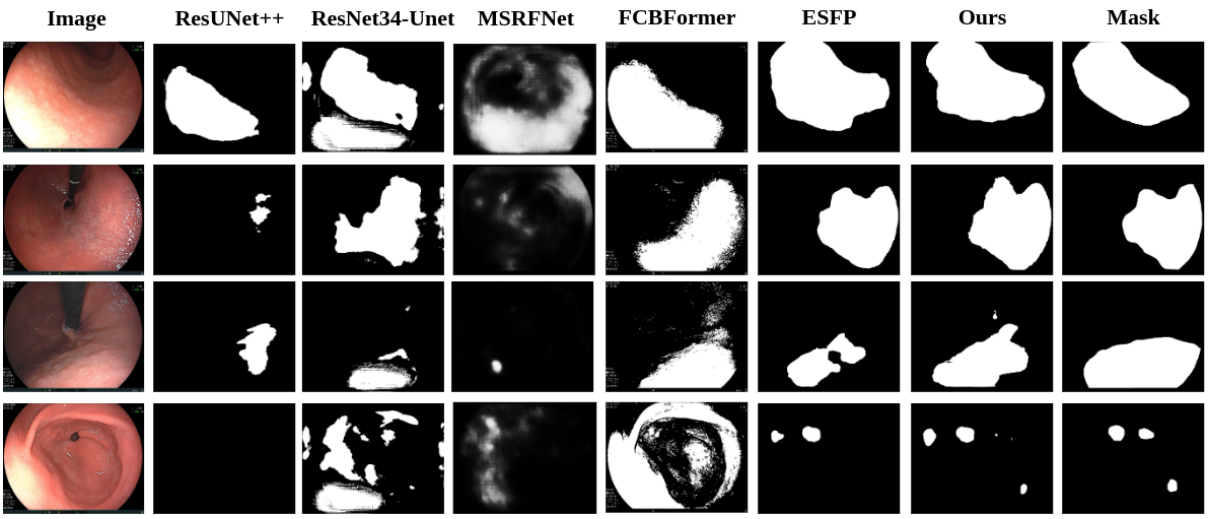

Figure 4.10: The Figure shows qualitative results of the proposed model in comparison with previous studies on Positive H. pylori gastritis dataset

.

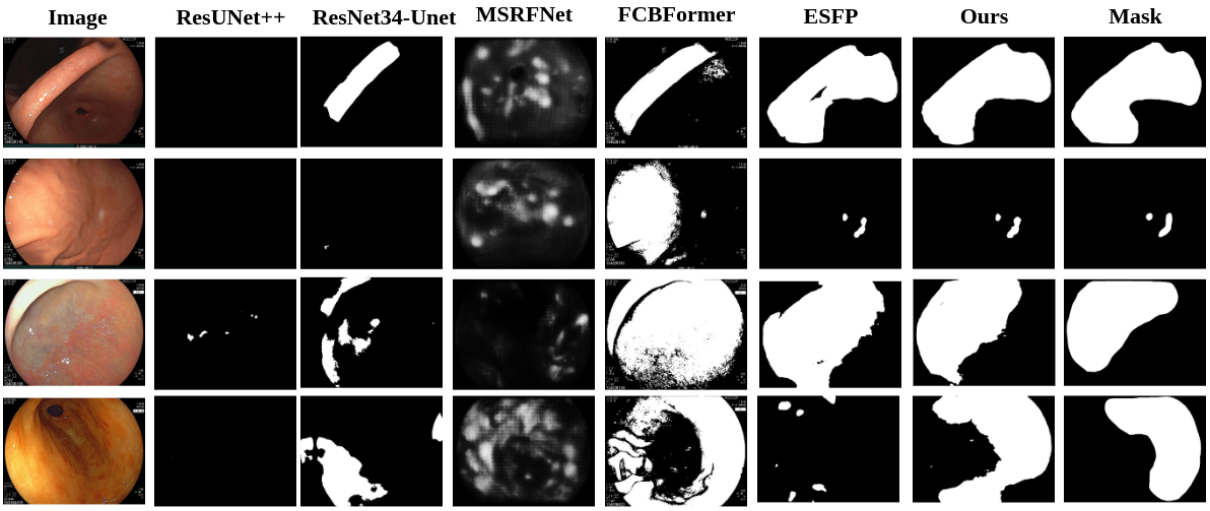

Figure 4.11: The Figure shows qualitative results of the proposed model in comparison with previous studies on Negative H. pylori gastritis dataset.

4.4. Demo Application

4.4.1. Streamlit-webrtc for real-time media streams

The interface is designed based on the Streamlit library. Streamlit is an open-source framework that empowers machine learning and data science professionals to swiftly develop and distribute attractive web applications. Tailored for Python enthusiasts in the machine learning field, it recognizes that data scientists and machine learning engineers often lack extensive web development experience and prefer a streamlined approach. Rather than investing weeks in mastering complex frameworks, they seek a tool that is both accessible and efficient. Streamlit fulfills this need by enabling the creation of visually appealing applications using just a handful of code lines, seamlessly displaying data and collecting essential modeling parameters.

Real-time video processing plays a crucial role in the development of various computer vision and machine learning models. This functionality is particularly valuable as it enables users to quickly assess the capabilities of their models by leveraging convenient video input from their devices. Specifically, in the context of endoscopy, real-time processing of endoscopic videos when deploying deep

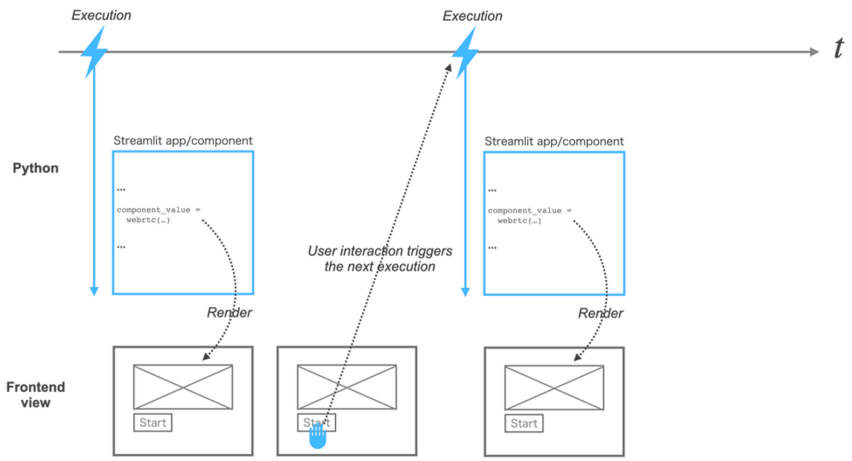

Figure 4.12: The basics of Streamlit's execution model: Upon each execution, the Python script is executed from top to bottom; Each execution of the Python script renders the frontend view, sending data from Python to JS as arguments to the component; The frontend triggers the next execution via *Streamlit.setComponentValue()*, sending data from JS to Python as a component value.

learning models on specialized equipment is essential. However, integrating real-time video processing into Streamlit poses a challenge, as Streamlit's native capabilities currently do not robustly support this feature.

Streamlit-webrtc is a component designed to empower Streamlit with the ability to seamlessly manage real-time media streams over a network. This component effectively addresses the challenge of incorporating real-time video processing into Streamlit applications.

4.4.2. Real-time Application Deployment

The interface has the following functionalities:

- Upload multiple images for processing.
- Upload and process real-time endoscopic video.
- Annotate regions of interest on various datasets.
- Showing the inference time for each image or each frame of video.

The following part visualizes the main interfaces of our demo application.

4.4.2.1. Image Inference

The image processing interface allows users to upload images from their local device, select an appropriate model for the chosen image data type. The segmentation result includes a binary predicted mask and the endoscopic image with the predicted lesion area annotated. The inference time for each image is also displayed.

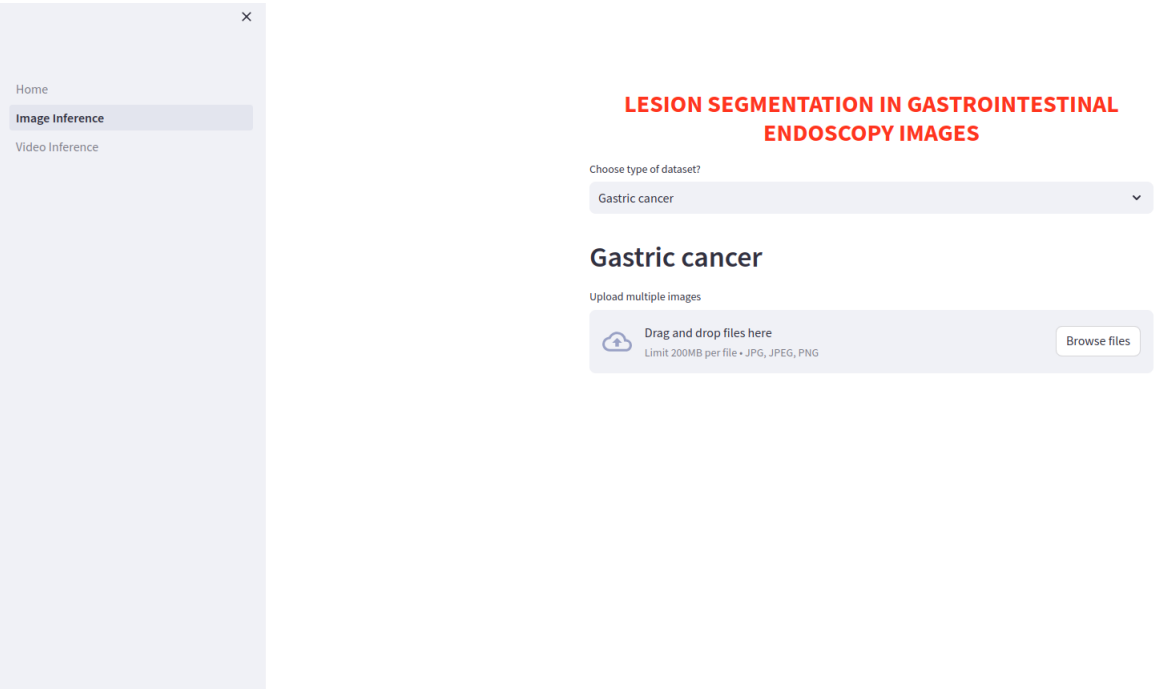

Figure 4.13: The interface allows uploading one or multiple images from the computer.

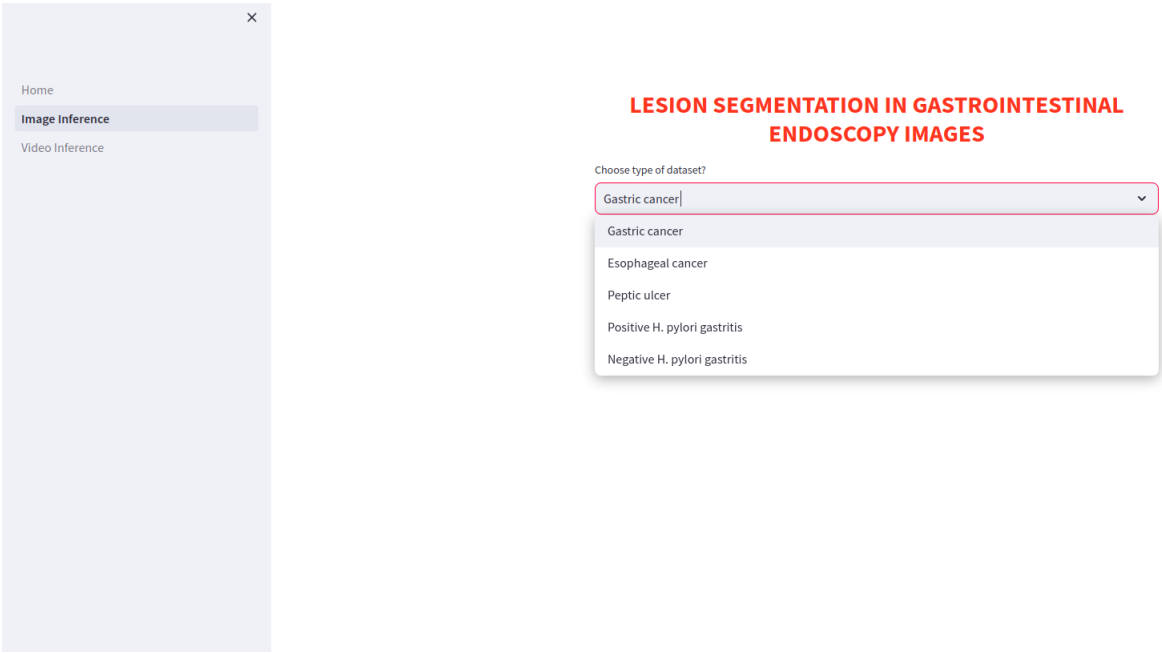

Figure 4.14: Component for selecting corresponding type of segmentation model for the uploaded images.

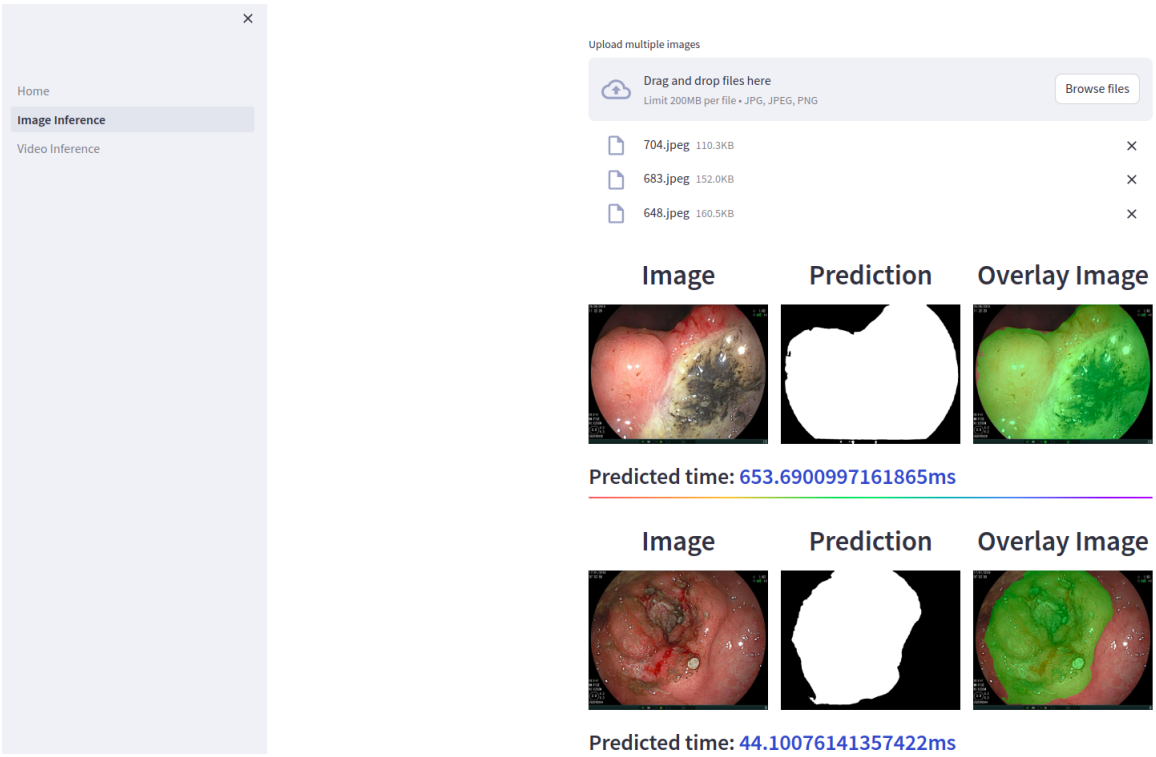

Figure 4.15: The output images will be the original images with the corresponding lesion area highlighted.

4.4.2.2. Video Inference

Similar to the image processing interface, the video processing interface allows users to upload endoscopy videos from their local devices and select an appropriate model for the chosen image data type. Here, due to various constraints, we only possess endoscopy videos of the stomach and esophagus. Real-time segmentation results are generated, which is of practical significance for our research. Inference time for each frame is also displayed.

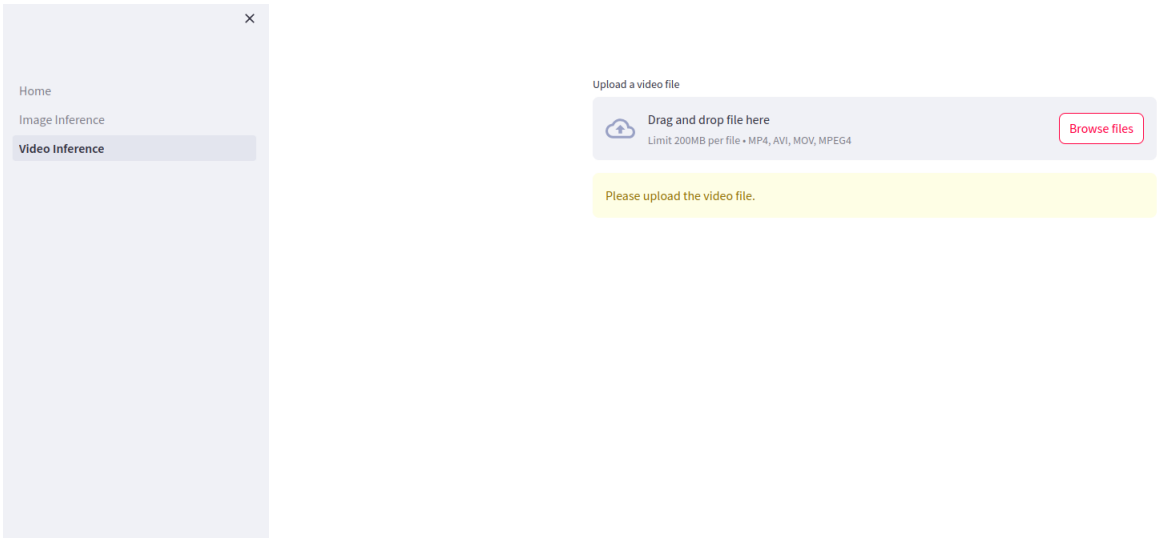

Figure 4.16: Upload endoscopy video.

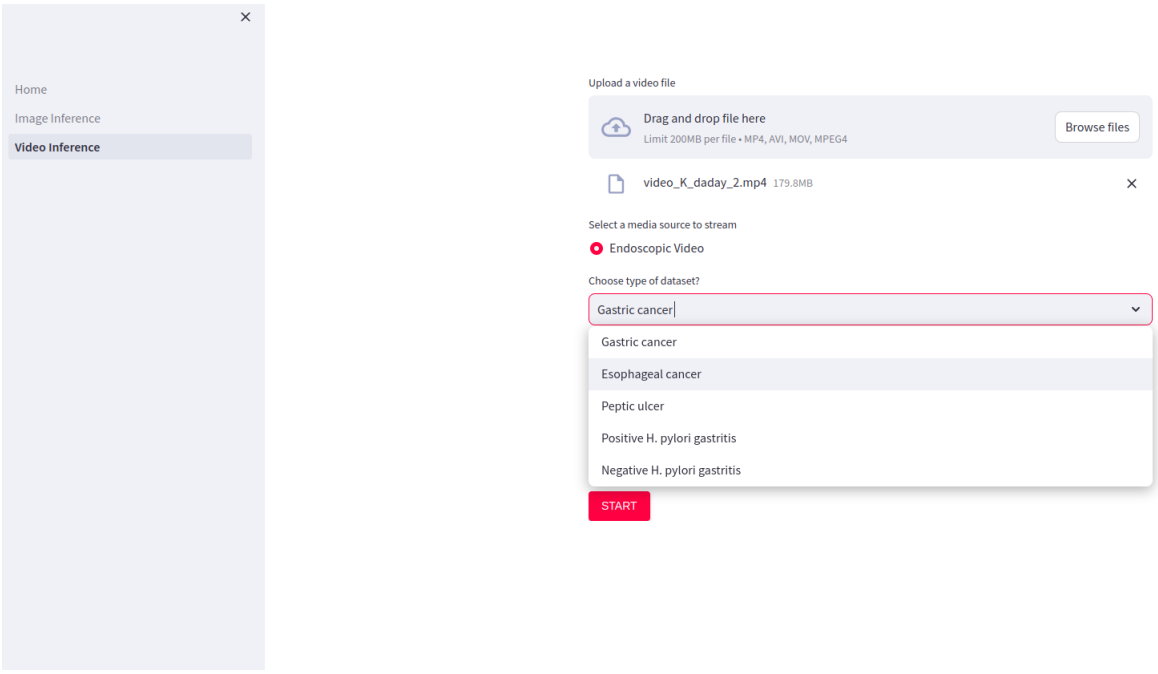

Figure 4.17: Select type of segmentation model.

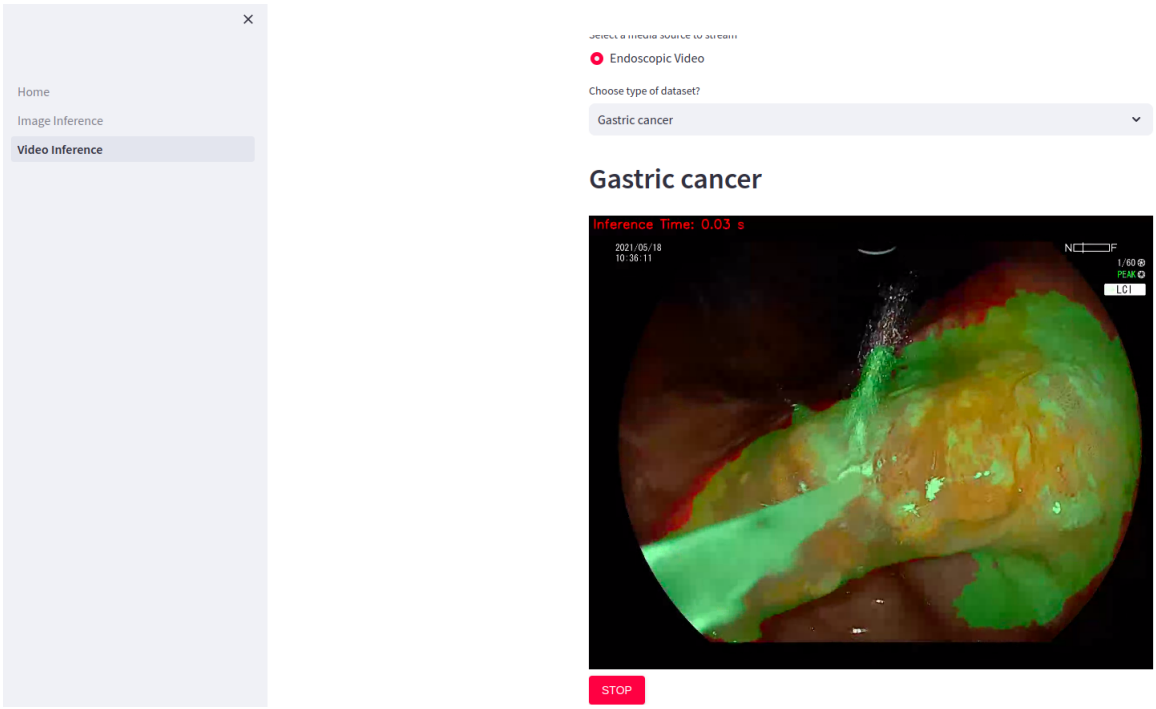

Figure 4.18: Showing the segmented video.

4.5. Conclusion

In this chapter, we discuss the datasets and experimental settings used for evaluating our proposed methodology. I have presented the results of our experiments, comparing the performance of our approach with state-of-the-art methods on our collected datasets. The outcomes of our study showcase the effectiveness of our approach and its potential to push the boundaries of lesion segmentation. Moreover, we also introduce a useful application for performing our work in a simple-to-deploy platform and easy-to-use python package.

CONCLUSION AND FUTURE WORK

1. Contribution

Firstly, we introduce using the deep network named ESFPNet for lesion segmentation.

Secondly, we propose a new loss function called FBI (Focal-BCE-IoU) and a learning strategy based on Lasso regularization for training deep learning models for medical image segmentation.

Thirdly, we provide a description of five new data sets for lesion segmentation and cancer detection.

Fourthly, we perform a comprehensive evaluation of the proposed approach on five different datasets. Both qualitative and quantitative evaluations are conducted using common performance metrics for segmentation task.

Finally, we compare our proposed approach with a recently state-of-the-art method, demonstrating superior performance in terms of segmentation accuracy. This comparison highlights the significance of our proposed method and its potential to advance the state-of-the-art in medical image segmentation.

2. Limitation

As with any research, there are several limitations to this thesis.

Firstly, our proposed methodology has only been evaluated on a limited number of datasets, and more extensive experiments on a larger variety of medical image datasets are required to fully assess its potential.

Secondly, while we have achieved better results in medical image segmentation, there is still room for improvement, and further research is needed to investigate other novel approaches and techniques to enhance the accuracy and efficiency of

medical image segmentation.

3. Future work

There are several directions for future work in this thesis.

Firstly, the proposed methodology can be further improved by exploring more cutting-edge deep learning techniques.

Secondly, the current experiments are conducted on a limited number of medical image datasets. In future work, more diverse datasets could be used to further validate the proposed methodology's generalizability and robustness.

Lastly, the demo application developed in this thesis could be further refined and developed into a fully functional tool for medical professionals to use in their clinical practice. This would require further collaboration with medical experts to ensure the application's usability and effectiveness.

BIBLIOGRAPHY

[1] Abhishek Srivastava et al. *MSRF-Net: A Multi-Scale Residual Fusion Network for Biomedical Image Segmentation*. 2022. arXiv: 2105.07451 [eess.IV].

[2] Alexey Dosovitskiy et al. "An Image is Worth 16*16 Words: Transformers for Image Recognition in Scale". In: *International Conference on Learning Representations*. 2021.

[3] AM Leufkens et al. "Factors influencing the miss rate of polyps in a back-to-back colonoscopy study". In: *Endoscopy* 44.05 (2012), pp. 470–475.

[4] Andreas Veit, Michael Wilber, and Serge Belongie. "Residual Networks Behave Like Ensembles of Relatively Shallow Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[5] Andriy Myronenko. "3D MRI brain tumor segmentation using autoencoder regularization". In: *International MICCAI Brainlesion Workshop*. Springer. 2018, pp. 311–320.

[6] Ange Lou and Murray Loew. "CFPNET: Channel-Wise Feature Pyramid For Real-Time Semantic Segmentation". In: *2021 IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 1894–1898. DOI: 10.1109/ICIP42928.2021.9506485.

[7] Ange Lou, Shuyue Guan, and Murray Loew. "CaraNet: context axial reverse attention network for segmentation of small medical objects". In: *Journal of Medical Imaging* 10.1 (2023), p. 014005.

[8] Ange Lou et al. “CaraNet: context axial reverse attention network for segmentation of small medical objects”. In: *Medical Imaging 2022: Image Processing*. Vol. 12032. International Society for Optics and Photonics. SPIE, 2022, pp. 81–92. DOI: 10.1117/12.2611802.

[9] Anran Lou and Murray Loew. “Cfpnet: Channel-wise feature pyramid for real-time semantic segmentation”. In: *arXiv preprint arXiv:2103.12212* (2021).

[10] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[11] Dan Ciresan et al. “Deep neural networks segment neuronal membranes in electron microscopy images”. In: *Advances in neural information processing systems* 25 (2012).

[12] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (GELUs)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[13] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[14] Dushyant Meher Jha, Hemanth Koppula, and Shreya Saxena. “ResUNet++: Hybrid deep learning networks for medical image segmentation”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 2791–2795.

[15] Edward Sanderson and Bogdan J. Matuszewski. “FCN-Transformer Feature Fusion for Polyp Segmentation”. In: *Medical Image Understanding and Analysis*. Springer International Publishing, 2022, pp. 892–907. DOI: 10.1007/978-3-031-12053-4_65. URL: https://doi.org/10.1007%2F978-3-031-12053-4_65.

[16] Eli Gibson et al. “Automatic multi-organ segmentation on abdominal CT with dense V-networks”. In: *IEEE Transactions on Medical Imaging* 37.8 (2018), pp. 1822–1834.

[17] Enze Xie et al. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*. 2021. arXiv: 2105.15203 [cs.CV].

[18] Frank Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain”. In: *Psychological review* 65.6 (1958), p. 386.

[19] Hao Chen et al. “DCAN: deep contour-aware networks for accurate gland segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2487–2496.

[20] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* (2014), pp. 2672–2680.

[21] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[22] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2009, pp. 248–255.

[23] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*. 2018, pp. 7132–7141.

[24] Jinfeng Wang et al. *Stepwise Feature Fusion: Local Guides Global*. 2022. arXiv: 2203.03635 [eess.IV].

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV].

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[27] Jorge Bernal et al. “Comparative Validation of Polyp Detection Methods in Video Colonoscopy: Results From the MICCAI 2015 Endoscopic Vision Challenge”. In: *IEEE Transactions on Medical Imaging* 36.6 (2017), pp. 1231–1249. DOI: 10.1109/TMI.2017.2664042.

[28] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2016, pp. 770–778.

[29] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[30] Karen Loaiza. “Deep learning for decision support in dermatology”. PhD thesis. Sept. 2020. DOI: 10.13140/RG.2.2.14692.60800.

[31] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder–decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[32] Li Yuan et al. “Tokens-to-token ViT: Training vision transformers from scratch on ImageNet”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 558–567.

[33] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[34] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[35] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[36] Maithra Raghu et al. “Do vision transformers see like convolutional neural networks?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by Marc’Aurelio Ranzato et al. Vol. 34. Virtual: Curran Associates, Inc., Oct. 2021, pp. 12116–12128.

[37] Minki. *Linear Regression with Tensorflow*. 2020. URL: `https://minkithub.github.io/2020/05/08/everydeeplearning1/`.

[38] Mustafa Alfarhan, Mohamed Deriche, and Ahmed Maalej. “Robust Concurrent Detection of Salt Domes and Faults in Seismic Surveys Using an Improved UNet Architecture”. In: *IEEE Access* PP (Dec. 2020), pp. 1–1. DOI: `10.1109/ACCESS.2020.3043973`.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: `1505.04597 [cs.CV]`.

[40] Ozan Oktay et al. *Attention U-Net: Learning Where to Look for the Pancreas*. 2018. arXiv: `1804.03999 [cs.CV]`.

[41] Ozgun Cicek et al. “3D U-Net: learning dense volumetric segmentation from sparse annotation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 424–432.

[42] Qi Chang et al. “ESFPNet: efficient deep learning architecture for real-time lesion segmentation in autofluorescence bronchoscopic video”. In: *arXiv preprint arXiv:2207.07759* (2022).

[43] Qi Chang et al. “ESFPNet: efficient deep learning architecture for real-time lesion segmentation in autofluorescence bronchoscopic video”. In: *Medical Imaging 2023: Biomedical Applications in Molecular, Structural, and Functional Imaging*. Vol. 12468. SPIE. 2023, p. 1246803.

[44] Reza Azad et al. *Medical Image Segmentation Review: The success of U-Net*. 2022. arXiv: 2211.14830 [eess.IV].

[45] Saining Xie et al. “Aggregated Residual Transformations for Deep Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 1492–1500.

[46] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic routing between capsules”. In: *Advances in neural information processing systems* (2017), pp. 3856–3866.

[47] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[48] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In: *British Machine Vision Conference (BMVC)*. 2016.

[49] Sharib Ali. “Where do we stand in AI for endoscopic image analysis? Deciphering gaps and future directions”. In: *npj Digital Medicine* 5 (Dec. 2022). DOI: 10.1038/s41746-022-00733-3.

[50] Tamanna. *Exploring Convolutional Neural Networks: Architecture, Steps, Use Cases, and Pros and Cons*. 2023. URL: https://medium.com/@tam.tamanna18/exploring-convolutional-neural-networks-architecture-steps-use-cases-and-pros-and-cons-b0d3b7d46c71.

[51] Vignesh Ungrapalli. *Universal Style Transfer*. 2017. URL: https://towardsdatascience.com/universal-style-transfer-b26ba6760040.

[52] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39 (Jan. 2017), pp. 2481–2495.

[53] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[54] Yuxin Wu and Kaiming He. “Group normalization”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[55] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. “Road Extraction by Deep Residual U-Net”. In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (May 2018), pp. 749–753. DOI: 10.1109/lgrs.2018.2802944. URL: https://doi.org/10.1109%2Flgrs.2018.2802944.

[56] Zhuang Liu et al. “Learning Efficient Convolutional Networks through Network Slimming”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

[57] Zongwei Zhou et al. “Unet++: A nested u-net architecture for medical image segmentation”. In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer. 2018, pp. 3–11.