UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

MASTER'S DEGREE IN COMPUTER ENGINEERING: ARTIFICIAL INTELLIGENCE AND ROBOTICS

Augmentation and Ensembles: Improving Medical Image Segmentation with SAM and Deep Networks

Supervisor
Prof. Nanni Loris

Co-supervisor
Prof. Fantozzi Carlo

Graduating
Carisi Lorenzo

ACADEMIC YEAR 2023-2024

Date of graduation 05/12/2024

This leads to a troubling question: how can these systems be so smart, yet also seem so limited?

— Yann LeCun

Abstract

This study explores novel ensemble strategies to improve image segmentation performance, particularly on medical image data. We investigate how the Segment Anything Model (SAM), despite not being explicitly trained for medical image segmentation, can still produce relevant information for model training. Building on these insights, we propose augmentation techniques that integrate SAM information directly into the images, enhancing the learning process of segmentation models. Each proposed augmentation method comes with its unique advantages, thereby to leverage the strengths of each approach, we introduce AuxMix, a model trained with three distinct SAM-based augmentation techniques. We conduct experiments on the state-of-the-art models, evaluating the effects of each technique independently and within an ensemble framework. The results show that our ensemble strategy, combining complementary information from each augmentation, leads to a robust and improved segmentation performance.

Contents

1 Introduction 1

2 Material and Methods 5

2.1 The Hybrid Semantic Network (HSNet) Model [26] 6

2.1.1 The Pyramid Vision Transformer (PVT) Encoder 7

2.1.2 Cross-Semantic Attention (CSA) 10

2.1.3 Hybrid Semantic Complementary (HSC) 11

2.1.4 Multi-Scale Prediction (MSP) 13

2.2 The Segment Anything Model (SAM) 14

2.2.1 Image Encoder 14

2.2.2 Prompt Encoder 15

2.2.3 Mask Decoder 15

2.2.4 Efficiency 15

2.2.5 Training and Losses 16

2.2.6 Segment Anything Model 2: Extension to Video 16

2.2.7 Fine-tuning of the SA Model 18

2.3 SAMAug: Augmenting Images with Foundation Model [13] 19

2.3.1 Masks Generation 20

2.3.2 Augmenting Input Images 21

2.3.3 Model Training with SAM-Augmented Images 22

2.3.4 Model Deployment with SAM-Augmented Images 24

2.4 Datasets and Methodology 25

2.4.1 The Kvasir-SEG Dataset [28] 25

2.4.2 The CVC-ColonDB Dataset [30] 27

2.4.3 The CVC-ClinicDB Dataset [31] 27

2.4.4 The ETIS-Larib Polyp Database [33] 29

2.4.5 The CVC-T Dataset [34] 29

2.5 Evaluation Metrics and Loss Functions 31

2.5.1 Structure Loss 31
2.5.2 Q-statistic 33
2.5.3 Dice Similarity Coefficient (DSC) 34

3 SegAug: Augmentations and Ensembles **37**
3.1 SegPrior RGB 37
3.1.1 Steps Breakdown 38
3.1.2 Usefulness of the Algorithm 38
3.2 SegPrior-Logits RGB 39
3.2.1 Steps Breakdown 39
3.2.2 Usefulness of the Algorithm 40
3.3 SegPrior HSV 40
3.3.1 Steps Breakdown 41
3.3.2 Usefulness of the Algorithm 41
3.4 SegPrior PCA 42
3.4.1 Steps Breakdown 42
3.4.2 Usefulness of the Algorithm 43
3.5 Ensemble Methods 44
3.5.1 Overview of Ensemble Approaches 44

4 Experiments and Results **47**
4.1 Model Training with SAM-Augmented Images 47
4.2 Segmentation Prior Augmentation Evaluation 48
4.3 Ensemble Strategy Evaluation 48

5 Conclusion **53**

Bibliography **63**

Chapter 1

Introduction

In April 2023, researchers at Meta, the multi-billion-dollar corporation, introduced a groundbreaking model named SAM, short for ”Segment Anything Model” [1]. This ambitious model aims to address the long-standing challenge of semantic image segmentation in Computer Vision (CV), a task that involves partitioning an image into segments, with each pixel corresponding to a distinct semantic class [2], an ability that has long been a secret of evolution.

To achieve state-of-the-art (SOTA) performance, the researchers took inspiration from advancements in Artificial Intelligence (AI), specifically in the field of Natural Language Processing (NLP). The introduction of transformers in mid-2017, through the ”Attention is All You Need” paper by Google researchers [3], enabled Deep Learning (DL) models to capture context over longer text spans. This advance led to improved generation, comprehension, and translation capabilities compared to previous methods like recurrent neural networks (RNNs) [4] and Long Short-Term Memory networks (LSTMs) [5], thus paving the way for what we now call foundational models, or Large Language Models (LLMs) [6].

A defining feature of these models is their vast number of parameters and their training on extensive text corpora from the web. This large-scale textual data enables Large Language Models (LLMs) to exhibit novel capabilities, collectively referred to as in-context learning. Among these are few-shot learning, where an LLM can solve a problem based on only a handful of examples, and zero-shot learning, where it can tackle new problems solely from natural language instructions, known as prompts.

The SAM project, therefore, emulates the structure of LLMs by implementing an architecture based on Visual Transformers (ViT) [7] for a total of over 600M parameters. This adaptation of the transformer model, introduced in mid-2021, successfully adapted the transformer concept to image segmentation, where Convolutional Neural Networks (CNNs) [8] had previously dominated as the leading architecture. The architecture was subsequently trained on an extensive dataset specifically curated by Meta, comprising 11 million images and over 1.1 billion masks. The project achieved remarkable results, with the model effectively segmenting a wide variety of objects and establishing a new state-of-the-art benchmark in the field.

However, despite SAM's foundational generalization capabilities, it still lacks domain-specific expert knowledge, leaving it less effective in certain domains [9], [10], such as medical image segmentation, where specialized state-of-the-art segmentation models outperform it. The aim of this work is, therefore, to enhance performance in these specialized domains.

To achieve this, we once again turn to the NLP field. While LLMs offer significant advantages, they are not without flaws, one of which is the phenomenon of hallucinations [11]. In this context, the model ”perceives patterns or objects that are nonexistent or imperceptible to human observers, creating outputs that are nonsensical or altogether inaccurate.” To mitigate this phenomenon, one promising approach is Retrieval-Augmented Generation (RAG) [12], where LLMs are not used as standalone models. Instead, they are integrated as a component in a broader pipeline, leveraging external knowledge sources to enhance the quality and accuracy of their outputs.

Specifically, we adopted the strategy outlined in Zhang et al. [13], where a SAM-based data augmentation technique is employed to produce new domain-specific images that are then feed to the domain-specific model in couple with the original ones during training. Data augmentation [14] is a Machine Learning (ML) approach that uses pre-existing data to create new data samples that can improve model optimization and generalizability. By introducing relevant patterns and information to the original images, SAM helps to downstream train a domain-specific segmentation model more effectively.

Building on the work of Zhang et al., we developed alternative methods to their SAMAug approach, aiming to incorporate additional outputs generated by SAM directly into the images, instead of relying solely on segmentation masks and stability scores. We experimented with various techniques to enrich the data. These included alternative combination methods, such as using Principal Component Analysis (PCA) and variance-based techniques, as well as exploring diverse channel representations beyond simply adding information to the RGB image channels.

An alternative fundamental deep learning technique in the training process of foundation models is fine-tuning. Fine-tuning is the process of adapting a pre-trained model for specific tasks or use cases. In this work, we also explored this technique, specifically to determine whether it could lead to a performance improvement in our pipeline.

Different SAM-augmentation techniques yield distinct training environments, even when applied to a shared baseline architecture, resulting in diverse performance outcomes during testing. Each trained model may better segment certain images and underperform on others. To leverage this, we employ ensemble learning, a meta-learning strategy in machine learning designed to enhance predictive accuracy by integrating the outputs of multiple models, known as base learners.

Ensemble learning combine these models to achieve performance levels that surpass any single model alone [15]. It is commonly categorized into three main methods: bagging, boosting, and stacking. Typically, ensemble methods generate multiple predictive models by varying aspects of the training process, such as model parameters, data subsets, or algorithms. The predictions of these models are then combined, often through averaging or voting, to form the final output. This strategy offers multiple advantages, including improved prediction accuracy, reduced overfitting, and greater robustness to noisy data. It is especially effective when base learners are diverse and exhibit uncorrelated errors. We therefore explore these ensemble approaches to gain a broader perspective on its potentiality.

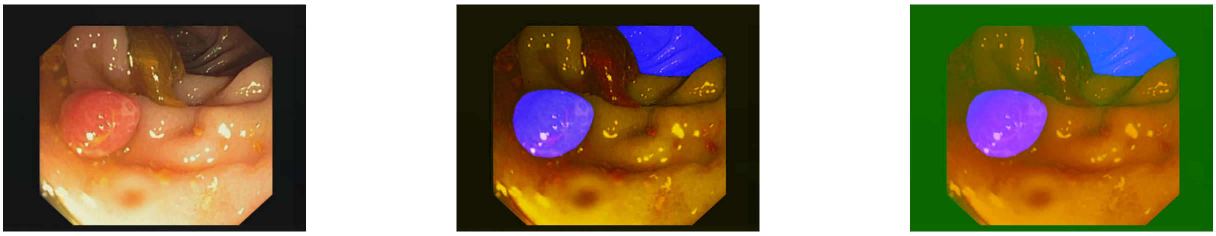

(a) original image (b) H-channel (c) PCA-based

Figure 1.1: Augmented versions of the 99th image from the CVC-300 dataset using some of the proposed methods: H-replacement and PCA-based analysis replacement.

To conduct experiments with the newly introduced SAM-based augmentations, we first focused on colorectal polyp (CRP) segmentation, a crucial area for early detection and prevention of colorectal cancer (CRC). CRPs are asymptomatic growths from the inner lining of the colon or rectum and while they are generally harmless, certain types may become dangerous if untreated, leading to cancer. According to Global Cancer Observatory (GCO) 2020 data, CRC is the second most common cancer and the third leading cause of cancer-related mortality globally, underscoring the importance of early detection and preventive strategies.

Regular screenings, such as colonoscopies, enable the identification and removal of potentially precancerous adenomatous polyps, lowering the risk of CRC progression. However, these protrusions vary considerably in size and can blend seamlessly with the surrounding mucosa, posing identification challenges even for skilled clinicians. With this challenge in mind, our initial experiments focused on CRP segmentation. We subsequently extended our research to additional datasets to assess the robustness of the segmentation methods across diverse applications, including camouflaged object segmentation (both naturally and artificially camouflaged objects) and rib segmentation.

The structure of this thesis is organized as follows.

Chapter 2 provides an overview of the theoretical foundations that underpin our work; it then introduces the baseline model, HSNet, which serves as a point of comparison for the novel augmentations and ensembles that we propose.

Chapter 3 details the newly developed models and ensemble techniques, describing the SAM-based augmentation strategies and the ensemble learning methods employed.

Chapter 4 is dedicated to presenting and discussing the experimental results, with particular emphasis on the impact of the proposed augmentations on segmentation accuracy across various domains.

Lastly, Chapter 5 concludes the thesis by summarizing key findings, discussing limitations, and suggesting potential directions for future work in this area.

Chapter 2

Material and Methods

Over the years, convolutional neural networks (CNNs) have become the cornerstone on polyp segmentation, demonstrating exceptional capabilities in capturing spatial hierarchies and local features with remarkable precision [16] [17] [18]. Among these, architectures like UNet [19] have gained prominence, thanks to their encoder-decoder structure, which effectively balances feature extraction and reconstruction. However, despite their widespread use, CNN-based models exhibit certain limitations, such as the loss of fine-grained details for smaller structures during down-sampling and the semantic inconsistency between encoder and decoder representations [20] [21]. Furthermore, while CNNs excel at modeling local patterns, they struggle to capture global context, a crucial element for comprehending the semantic relationships within an image [22] [23].

In recent years, the introduction of Transformers has brought about a paradigm shift in computer vision, enabling the modeling of long-range dependencies and global relationships through self-attention mechanisms [3] [7]. This development has led to significant advancements in various tasks, including medical image segmentation. Transformers offer an enhanced ability to perceive global context, which is particularly valuable in segmentation tasks. However, their performance often falls short when recovering fine details during the decoding phase, which remains essential for producing accurate and high-resolution segmentation masks.

Hybrid models that combine the strengths of CNNs and Transformers have emerged as a promising direction [24] [25]. These approaches aim to leverage the complementary strengths of both paradigms: the local precision of CNNs and the global contextual understanding of Transformers. Nevertheless, many of these solutions rely on straightforward architectural combinations, which fail to fully exploit the potential interactions between the two modalities, often resulting in suboptimal recovery of local details and semantic consistency.

To address these limitations, HSNet [26] was proposed in 2022 as a novel approach to polyp segmentation. HSNet redefines the integration of CNNs and Transformers by enabling a deeper and more synergistic interaction between the two. This innovative framework achieves a balance between global context modeling and local detail preservation, setting a new benchmark for segmentation tasks and addressing the challenges faced by conventional methods.

2.1 The Hybrid Semantic Network (HSNet) Model [26]

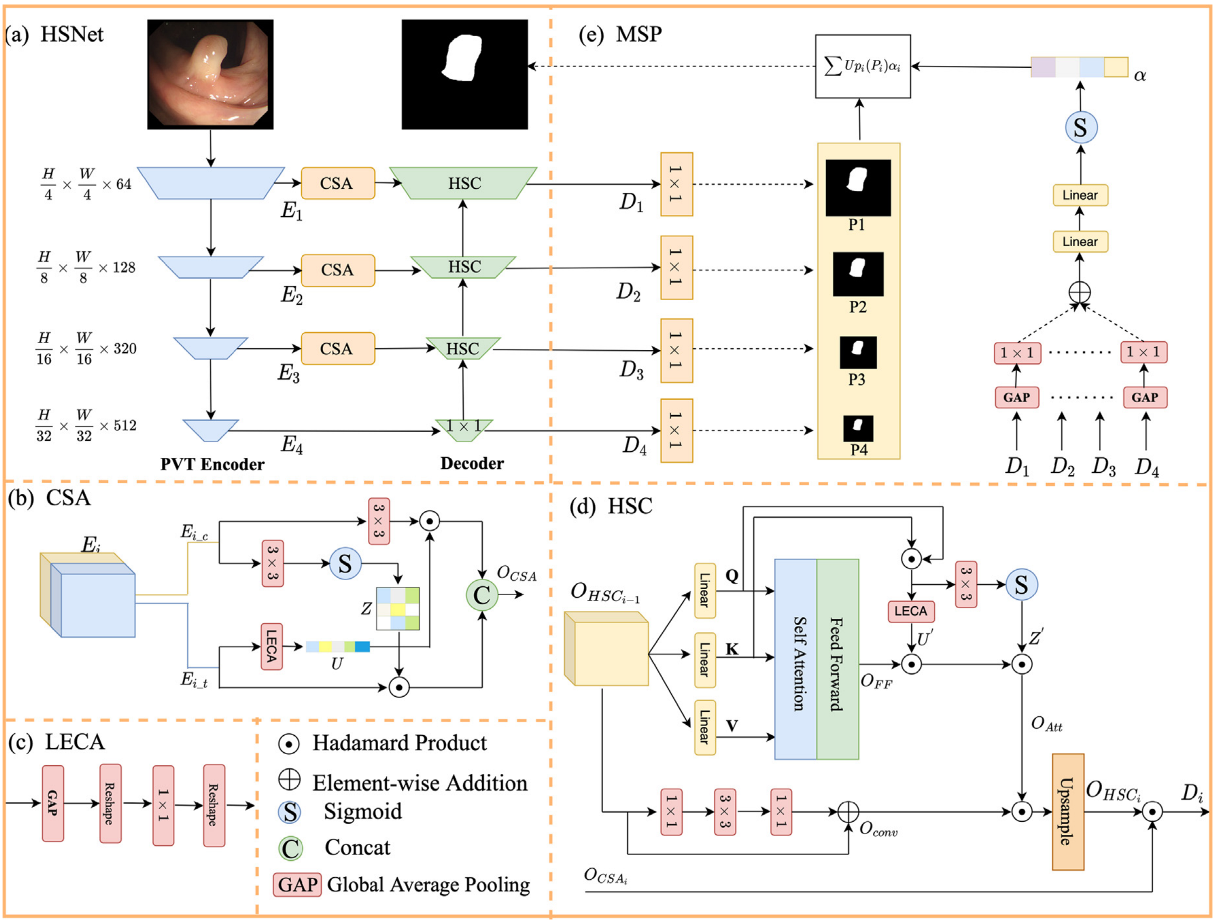


Figure 2.1: (a) Illustration of the HSNet framework, emphasizing its four core components: (b) the Cross-Semantic Attention (CSA) module, which mitigates semantic discrepancies between encoder and decoder features; (c) the Linear Efficient Channel Attention (LECA) module, designed to refine feature selection with minimal computational complexity; (d) the Hybrid Semantic Complementary (HSC) module, enabling synergistic integration of local and global semantics; and (e) the Multi-Scale Prediction (MSP) module, which ensures robust segmentation by balancing features across multiple scales. Adapted from [26].

The architecture of HSNet, depicted in Figure 2.1, is structured around four interconnected modules, each addressing specific challenges in the task of polyp segmentation:

- **PVT Encoder**: Responsible for extracting a hierarchical set of features, capturing low-level characteristics such as texture, color, and edges, which serve as the foundation for subsequent processing.

- **Cross-Semantic Attention (CSA)**: Aims to refine feature representations by suppressing irrelevant noise in low-level features and aligning them with high-level semantic information, thereby bridging the gap between the encoder and decoder stages.
- **Hybrid Semantic Complementary (HSC)**: Combines the strengths of Transformers and Convolutional Neural Networks (CNNs) to extract and complement diverse semantic cues, integrating local details with global contextual understanding.
- **Multi-Scale Prediction (MSP)**: Optimizes the segmentation process by leveraging multi-scale feature representations with adaptive weighting mechanisms, ensuring smooth gradient propagation and precise predictions at varying scales.

In the following sections, we will analyze the internal mechanics and design principles of these modules in greater detail, highlighting their contributions to the overall performance of HSNet.

2.1.1 The Pyramid Vision Transformer (PVT) Encoder

The PVT encoder processes the input in a hierarchical manner, generating multi-scale feature maps across four stages by progressively reducing spatial resolution and increasing the number of channels. For an input image of size $H \times W \times 3$, the feature map at the i-th stage is:

$$F_i \in \mathbb{R}^{\frac{H}{S_i} \times \frac{W}{S_i} \times C_i}, \tag{1}$$

where $i \in \{1, 2, 3, 4\}$, $S_i \in \{4, 8, 16, 32\}$ represents the spatial stride at the i-th stage, and $C_i \in \{64, 128, 320, 512\}$ denotes the number of channels at stage i. Each stage consists of the following components:

- **Patch Embedding**: The input image is divided into patches, and the spatial resolution is reduced. These patches are then projected into higher-dimensional embeddings.
- **Transformer Encoder Block**: The embedded patches, along with positional embeddings, are processed by the Transformer encoder, which aggregates features through the self-attention mechanism.

In this way, the encoder generates a pyramid of feature maps $\{F_1, F_2, F_3, F_4\}$, where each successive stage has a coarser resolution and larger channel dimensions, suitable for dense prediction tasks such as object detection and semantic segmentation.

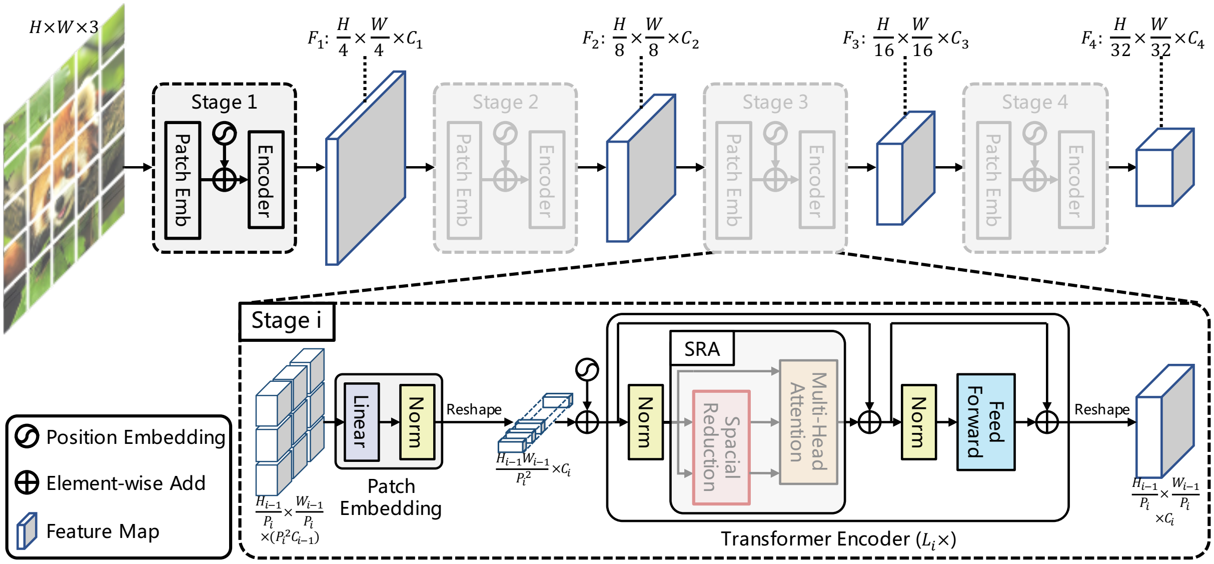


Figure 2.2: Overall architecture of Pyramid Vision Transformer (PVT). Courtesy of [27]

To efficiently process high-resolution feature maps (e.g., 4-stride), the Spatial Reduction Attention (SRA) layer is introduced to replace the traditional Multi-Head Attention (MHA) layer. Like MHA, the SRA receives a query Q, a key K, and a value V as inputs. However, the key difference is that the SRA mechanism optimizes the self-attention by introducing a spatial downsampling step. This process operates as follows:

1. **Spatial Reduction of Keys and Values**: The key matrix $K \in \mathbb{R}^{H\times W\times C}$ and value matrix $V \in \mathbb{R}^{H\times W\times C}$ are downsampled spatially by a factor r (e.g., $r = 2$), resulting in reduced versions $\hat{K} \in \mathbb{R}^{\frac{H}{r}\times\frac{W}{r}\times C}$ and $\hat{V} \in \mathbb{R}^{\frac{H}{r}\times\frac{W}{r}\times C}$. This reduces the number of spatial elements that the attention mechanism processes.

2. **Computation of Scaled Attention**: The scaled dot-product attention is calculated as:

$$A = \text{softmax}\left(\frac{Q\hat{K}^{\top}}{\sqrt{C}}\right), \tag{2}$$

   where $Q \in \mathbb{R}^{H\times W\times C}$ is the query matrix, $\hat{K}^{\top}$ is the transposed reduced key, and C represents the feature dimension. The softmax function ensures proper normalization of attention weights across spatial locations.

3. **Output Generation**: The attention weights A are then applied to the reduced value matrix $\hat{V}$, producing the refined feature representation:

$$O_{PVT} = A\hat{V}. \tag{3}$$

By reducing the spatial scale of K and V, SRA significantly decreases the memory and computational requirements of the self-attention operation. This makes it particularly effective for processing high-resolution feature maps, as required in the PVT encoder, without sacrificing performance.

PVTv2 improvements

PVTv2 introduces several key improvements over PVTv1, addressing limitations like computational complexity and lack of local continuity in image patches.

1. Linear Spatial Reduction Attention (SRA): Linear Spatial Reduction Attention (SRA) is an efficient mechanism introduced in PVT v2 to address the high computational cost of traditional self-attention operations in high-resolution inputs. Unlike the original SRA in PVT v1, which uses convolutions for spatial reduction, Linear SRA employs average pooling to downsample the spatial dimensions to a fixed size ($P \times P$) before applying the attention mechanism. This modification reduces computational complexity from quadratic to linear with respect to the input size, as the cost becomes independent of the spatial reduction ratio R. Specifically, for an input of size $h \times w \times c$, the complexity of Linear SRA is:

$$\Omega(\text{Linear SRA}) = 2hwP^2c, \tag{4}$$

compared to:

$$\Omega(\text{SRA}) = \frac{2h^2w^2c}{R^2} + hwc^2R^2, \tag{5}$$

in the original SRA. By adopting average pooling, Linear SRA retains performance while significantly improving efficiency, making it better suited for tasks requiring high-resolution inputs.

2. Overlapping Patch Embedding To capture local continuity information in images, PVT v2 employs overlapping patch embedding as a technique for tokenizing images. This method involves enlarging the patch window and allowing adjacent windows to overlap by half of their area, ensuring that the local features are better preserved. The feature map is padded with zeros to maintain the resolution throughout the process. Overlapping patch embedding is implemented using convolution with zero padding. Specifically, given an input of size $h \times w \times c$, it is passed through a convolution layer with a stride of S, a kernel size of $2S - 1$, padding of $S - 1$, and c' output channels. The result is an output of size $\frac{h}{S} \times \frac{w}{S} \times c'$, effectively capturing local continuity while reducing spatial dimensions.

3. Convolutional Feed-Forward Network PVTv2 incorporates convolutional layers into the feed-forward network, replacing traditional fully connected layers. This reduces parameters and enhances the model's ability to capture spatial features.

HSNet with PVTv2

By incorporating PVTv2 as its backbone, HSNet efficiently captures multi-scale features while maintaining computational efficiency. The hierarchical structure and SRA mechanism enable robust feature extraction across varying resolutions, making HSNet well-suited for dense prediction tasks such as segmentation.

2.1.2 Cross-Semantic Attention (CSA)

The architecture incorporates a method to bridge the semantic gap between low-level and high-level features. The low-level features, while rich in semantic details, often suffer from background noise, making it difficult for them to effectively integrate with high-level features. To address this, the architecture uses a lateral connection that combines these features, inspired by the strengths of both Transformers and Convolutional Neural Networks (CNNs).

The Transformer encoder extracts low-level features, which are then split along the channel dimension into two parts. Specifically,

$$E_{i_t} \in \mathbb{R}^{\frac{H}{S_i} \times \frac{W}{S_i} \times \frac{C_i}{2}} \quad \text{and} \quad E_{i_c} \in \mathbb{R}^{\frac{H}{S_i} \times \frac{W}{S_i} \times \frac{C_i}{2}},$$

where E_{i_t} retains global information and E_{i_c} contains local spatial details. Two parallel paths are employed to process these features: one to extract the 1D channel global descriptor U, and the other to capture the 2D spatial relationship descriptor Z.

Channel Global Statistical Descriptor from Transformer

For the channel global descriptor, the process begins by applying global average pooling to compute channel-wise statistics, followed by reshaping the tensor and applying a 1D convolutional layer. The transformation can be formulated as:

$$U = \sigma(\mathcal{R}^t(\mathcal{E}_1(\mathcal{R}(\text{AvgPool}(E_{i_t}))))), \tag{6}$$

Here, $\text{AvgPool}(\cdot)$ represents the global average pooling operation, which aggregates global statistics over channels. The reshaping operation $\mathcal{R}(\cdot)$ converts the pooled 2D tensor into a 1D vector, while $\mathcal{E}_1(\cdot)$ applies a 1D convolutional layer with kernel size $k = 5$ and stride $s = 1$. The operation $\mathcal{R}_t(\cdot)$ maps the 1D vector back to 2D space, and $\sigma(\cdot)$ represents the sigmoid activation function.

Spatial Global Descriptors from CNN

For the spatial global descriptor, a simple convolutional operation is applied to capture local spatial dependencies. This is formulated as:

$$Z = \sigma(\mathcal{F}_1(E_{i_c})), \tag{7}$$

where $\mathcal{F}_1(\cdot)$ is a 3x3 convolutional layer that reduces the output to a single channel, capturing the spatial relationships in the low-level features.

Interactive Attention Mechanism

The final output feature tensor is obtained by calibrating both the channel and spatial dimensions through an interactive attention mechanism. This process is formulated as:

$$O_{HSC} = \text{Concat}((\mathcal{F}_2(E_{i_c}) \odot U) \odot (E_{i_t} \odot Z)), \tag{8}$$

In this equation, $\odot$ represents the Hadamard product, which performs element-wise multiplication. The operation $\mathcal{F}_2(\cdot)$ applies another 3x3 convolutional layer, while Concat$(\cdot)$ concatenates the results of the channel and spatial information, integrating both global and local semantic features.

Through this mechanism, the channel global descriptor U, extracted from E_{i_t}, summarizes the global characteristics retained from the Transformer encoder. Meanwhile, the spatial descriptor Z, extracted from E_{i_c}, captures local relationships between adjacent features within the receptive field.

2.1.3 Hybrid Semantic Complementary (HSC)

In the proposed architecture, the decoder plays a critical role in transforming low-resolution feature maps into high-resolution segmentation masks. This transformation is achieved through a progressive up-sampling process that not only models global semantic cues but also recovers finer details of the features. To balance these two objectives, the authors introduce a Hybrid Semantic Complementary (HSC) module, which is composed of two branches designed to capture global and local features independently.

Improved Transformer Branch

In the first branch, the authors introduce an enhanced self-attention mechanism to capture long-range dependencies within the feature maps. Similar to the approach used in Pyramid Vision Transformer (PVT), the input features are first embedded into overlapping patches using convolutional layers. These patches are then processed by three fully connected layers to compute the query Q, key K, and value V. The affinity matrix is computed by taking the dot product between the query and key, which is then used to attend to the value tensor. This results in the self-attention output, denoted as O_{SA}. The formula for this process is:

$$O_{SA} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{9}$$

where d represents the dimensionality of the feature embedding. The resulting O_{SA} is passed through a Convolutional Feed-Forward module to refine the output, producing O_{FF}, as shown by:

$$O_{FF} = O_{SA} + \delta\left(\mathcal{F}_{C,2}\left(\mathcal{DW}\left(\delta\left(\mathcal{F}_{C,1}(O_{SA})\right)\right)\right)\right). \tag{10}$$

Here, $\mathcal{F}_{C,1}(\cdot)$ and $\mathcal{F}_{C,2}(\cdot)$ are linear layers with an expansion rate $r = 4$, $\delta(\cdot)$ denotes the ReLU activation function, and $\mathcal{DW}(\cdot)$ refers to a depth-wise convolution layer.

The query Q and key K are crucial as they carry important contextual information that the attention mechanism leverages. The authors further enhance the self-attention process by calculating two important descriptors—channel attention U' and spatial attention Z'—after fusing Q and K. These descriptors help to calibrate the final output O_{FF}. To align the dimensions for these calculations, the reshaped query and key tensors, Q and K, are expressed as:

$$Q' \in \mathbb{R}^{n\times d_i \times \frac{\times H\times W}{S_i}} \quad \text{and} \quad K' \in \mathbb{R}^{n\times d_i \times \frac{\times H\times W}{S_i}},$$

where n represents the number of heads in the multi-head self-attention (MHSA) layer, and d_i is the dimensionality of the feature embedding in the i-th stage. The channel attention U' is computed as:

$$U' = \sigma\left(\mathcal{R}_t\left(\mathcal{E}_2\left(\mathcal{R}(\text{Avg}(Q' \odot K'))\right)\right)\right), \tag{11}$$

and the spatial attention Z' is calculated as:

$$Z' = \sigma\left(\mathcal{F}_3(Q' \odot K')\right). \tag{12}$$

The final attention output O_{Att} is then obtained by applying both channel and spatial attention to the self-attention output O_{FF}:

$$O_{Att} = O_{FF} \odot U' \odot Z'. \tag{13}$$

CNN Branch with Bottleneck

In the second branch, a purely convolutional bottleneck architecture is used to model local feature details. This branch captures the fine-grained spatial relationships within the feature map. The output tensor from this branch is expressed as:

$$O_{Conv} = \delta\left(O_{HSC} + \mathcal{F}_6\left(\delta\left(\mathcal{F}_5\left(\delta\left(\mathcal{F}_4(O_{HSC})\right)\right)\right)\right)\right), \tag{14}$$

where $\mathcal{F}_4(\cdot)$ and $\mathcal{F}_6(\cdot)$ are 1x1 convolutional layers used to adjust the feature channel dimensions, and $\mathcal{F}_5(\cdot)$ is a 3x3 convolutional layer used for intermediate feature encoding.

Fusion Strategy

To obtain the final output of the HSC module, the two branches are fused by applying a 2x upsampling operation to the element-wise product of the attention-enhanced tensor O_{Att} and the local detail tensor O_{Conv}. The resulting output O_{HSC} is given by:

$$O_{HSC} = \text{Up}\left(O_{Att} \odot O_{Conv}\right), \tag{15}$$

where $\text{Up}(\cdot)$ denotes the up-sampling operation. This fusion of the global semantic features from the Transformer branch and the local details from the CNN branch enables the decoder to progressively refine the segmentation mask.

Finally, the output tensor O_{HSC} of the i-th stage is multiplied by O_{CSA_i} to obtain the decoder output tensor D_i. For the final stage, D_4 is computed using a 1x1 feature conversion layer.

2.1.4 Multi-Scale Prediction (MSP)

In the HSNet architecture a multi-scale prediction (MSP) module is also introduced. This module employs a learnable mechanism to compute weight coefficients for integrating predictions across different stages of the network. The design ensures retention of semantic information at multiple scales while facilitating gradient flow through the network via auxiliary loss functions.

The MSP module processes the decoder outputs D_i at various stages by applying four parallel 1×1 convolutional layers followed by upsampling layers, generating binary masks P_1, P_2, P_3, and P_4. To determine the weight coefficients α_i, a compact fully connected network is used. Initially, D_1, D_2, D_3, and D_4 are spatially compressed using global average pooling. These features are then aligned to the same dimension ($m = 64$ in this implementation) via 1×1 convolutional layers. The aligned features are combined through channel-wise addition and processed by two consecutive fully connected layers, culminating in the generation of α_i using the sigmoid activation function.

The final output mask O_{mask} is obtained through an adaptive weighted addition of the upsampled stage predictions. This process is expressed as:

$$O_{\text{mask}} = \sum_i \text{Up}_i(P_i)\alpha_i, \tag{16}$$

where $\text{Up}_i(\cdot)$ denotes the upsampling operation applied to P_i with magnifications of

$\{4, 8, 16, 32\}$ for $i \in \{1, 2, 3, 4\}$.

Implementation Framework and Architectural Modifications

To conclude, HSNet is a powerful segmentation model that was selected as the foundation for our work due to its strong performance in complex segmentation tasks. The experiments in this project were carried out using the `PyTorch` framework, with the optimization process guided by the AdamW algorithm. The model was trained over 100 epochs to strike a balance between computational efficiency and the risk of overfitting. The learning rate was initially set to 5×10^{-5}, with a reduction by a factor of 0.5 applied after the 15th and 30th epochs. Due to the small size of the training set, all available data was used for training, and the final epoch weights were adopted as the trained model.

A key aspect of the training process was the employment of the structure loss function detailed in 2.5.1.

In terms of model architecture, one significant adjustment was the removal of the normalization layer, typically employed in standard segmentation models to scale outputs between 0 and 1. While this is effective when foreground structures are consistently present, medical datasets often feature images with varying foreground content or even the absence of foreground structures. By removing the normalization layer, the model gained greater flexibility, allowing it to handle such variations without compromising its ability to generalize across different domains.

These architectural and methodological adaptations were essential in optimizing the model for cross-domain segmentation tasks, ensuring that it could effectively process diverse datasets, including those with camouflaged ribs and other domain-specific features.

2.2 The Segment Anything Model (SAM)

The **Segment Anything Model** (SAM) is an advanced model for flexible and promptable image segmentation. SAM enables real-time, interactive segmentation by utilizing pre-trained vision models and efficient processing techniques. The model is composed of three primary components: the image encoder, the flexible prompt encoder, and the fast mask decoder. These components work together to process images and generate segmentation masks based on various input prompts.

2.2.1 Image Encoder

SAM's image encoder is based on a *Vision Transformer* (ViT), specifically a pre-trained *Masked Autoencoder* (MAE), which has been adapted to handle high-resolution inputs. This encoder processes each image independently and generates an image embedding that is then passed on for further processing in the next stages. By leveraging the powerful pretraining methods of the ViT, SAM can scale efficiently and produce high-quality results. The image encoder plays a critical role in ensuring that SAM works effectively with large-scale, high-resolution images.

2.2.2 Prompt Encoder

The flexibility of SAM arises from its prompt encoder, which allows it to accept different types of inputs (referred to as ”prompts”) that guide the segmentation task. These prompts can be categorized as follows:

- **Sparse Prompts**: These include points, bounding boxes, and text. Points and boxes are represented through positional encodings combined with learned embeddings, while text is processed by a CLIP-based encoder that handles free-form textual descriptions.
- **Dense Prompts**: These involve segmentation masks, which are embedded using convolutional layers and then combined with the image embedding.

The prompt encoder is responsible for transforming these varied inputs into embeddings that are used by the mask decoder. This enables SAM to adapt to a wide range of segmentation tasks based on the type of prompt provided by the user.

2.2.3 Mask Decoder

The mask decoder is responsible for generating the final segmentation mask from the image and prompt embeddings. The decoder utilizes a modified *Transformer* architecture that incorporates both self-attention and cross-attention mechanisms. This allows the model to refine its segmentation results by considering both the image content and the provided prompts. After processing the embeddings through several layers, the decoder outputs a mask indicating the foreground probability for each pixel in the image.

To handle ambiguity in segmentation, SAM generates multiple masks (typically three) for each prompt. These masks represent different levels of granularity of the object: the whole object, parts of the object, and subparts. The model assigns a confidence score (estimated Intersection over Union, or IoU) to each mask, which helps rank the masks and select the most appropriate one.

2.2.4 Efficiency

SAM is designed with a strong emphasis on efficiency. The model is optimized to run in real-time, with the entire process—from prompt encoding to mask decoding—completing in approximately 50 milliseconds per image on a standard CPU. This fast runtime is essential for enabling interactive applications, where users provide prompts and receive segmentation results instantly.

2.2.5 Training and Losses

SAM is trained for promptable segmentation using a combination of *focal loss* and *Dice loss*. During training, the model learns to predict segmentation masks based on randomly sampled prompts, simulating an interactive environment. This training setup enables SAM to perform a wide variety of segmentation tasks, making it versatile for many real-world applications.

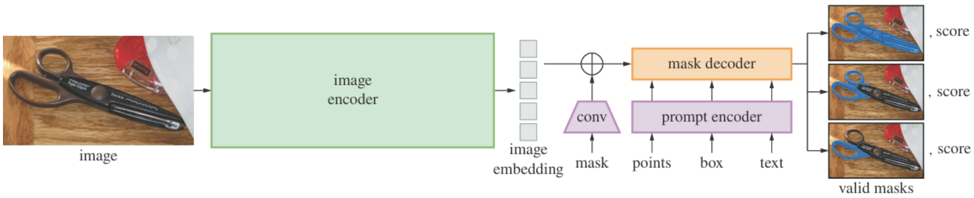


Figure 2.3: SAM pipeline scheme, courtesy of [1]

2.2.6 Segment Anything Model 2: Extension to Video

SAM2 extends the capabilities of the Segment Anything Model (SAM) to the video domain. It allows segmentation based on point, box, and mask prompts on individual video frames to define the spatial extent of objects to be segmented across time. While SAM operates in a similar manner spatially for each frame, SAM2 integrates temporal information by leveraging memories of past predictions and prompted frames.

Spatial and Temporal Integration

Spatially, SAM2 behaves similarly to SAM. The model uses a promptable, lightweight mask decoder to process image embeddings and prompts, outputting a segmentation mask for each frame. Prompts can be iteratively added to refine masks.

The major difference in SAM2 is its use of memory for temporal integration. The frame embedding used by SAM2's decoder is conditioned not only on the image encoder's output but also on the memory of previous frame predictions and prompts. This allows SAM2 to handle video frames that may include future prompts relative to the current frame. Memories are created by the memory encoder based on past predictions and stored in a memory bank for use in subsequent frames.

Memory Attention

Memory attention is a crucial feature in SAM2, allowing the model to condition the current frame's features on past frames' features and predictions, as well as any new prompts. The memory attention mechanism operates using multiple transformer blocks, with the first block taking the current frame's image encoding as input. These blocks perform self-attention, followed by cross-attention to memories of previous frames and object pointers, stored in the memory bank. This design enables SAM2 to effectively utilize both spatial and temporal features for segmentation.

Image Encoder and Mask Decoder

SAM2's image encoder follows a streaming approach for processing videos in real-time. It processes frames sequentially, consuming each frame as it becomes available. The image encoder is based on a hierarchical MAE (Masked Autoencoder) pre-trained model, which allows SAM2 to utilize multiscale features during mask decoding.

The prompt encoder and mask decoder are similar to those in SAM. The prompt encoder can handle sparse prompts (clicks, boxes, masks) and dense prompts (mask embedding). The decoder predicts multiple masks for ambiguous prompts, selecting the mask with the highest predicted Intersection over Union (IoU) when ambiguity remains.

Memory Encoder and Memory Bank

The memory encoder generates a memory by combining downsampled mask outputs with unconditioned frame embeddings. These memories are stored in a memory bank, which maintains information about past frames and prompts. The memory bank employs a FIFO queue, storing memories of up to N recent frames and M prompted frames. The memory attention cross-attends to both the spatial memory features and object pointers to ensure accurate segmentation.

Training Process

SAM2 is trained jointly on image and video data in an interactive manner. During training, sequences of frames are sampled, with up to 2 frames being prompted for correction. The model learns to sequentially predict ground-truth masklets, using initial prompts that are either the ground-truth mask, a positive click, or a bounding box. This training setup enables SAM2 to generalize across both spatial and temporal domains, making it suitable for video segmentation tasks.

2.2.7 Fine-tuning of the SA Model

In this work, we build upon the approach of previous researchers to fine-tune the Segment Anything Model (SAM) on a custom medical image dataset. The objective was to adapt SAM's pre-trained capabilities to improve its segmentation performance in the context of medical images, which present unique challenges such as variations in tissue types, lesion characteristics, and imaging modalities. By fine-tuning SAM, we leverage its ability to generalize from large-scale datasets while tailoring it to specific medical tasks.

The methodology we employed for fine-tuning SAM follows the steps outlined in prior studies that have successfully adapted SAM to medical image segmentation. Specifically, we utilized a medical dataset consisting of pairs of images and their corresponding ground truth masks, where the masks represent key anatomical structures, such as tumors or lesions.

We adapted the dataset for training by generating bounding box prompts from the ground truth masks. These bounding boxes were used as simplified representations of the regions of interest (ROI) to guide SAM during the segmentation process. SAM, which takes both image input and bounding box prompts, was fine-tuned to predict segmentation masks based on these inputs.

The training process was based on a custom loss function, derived from the MONAI framework, which combines the Dice similarity coefficient and cross-entropy loss. This combination is well-suited for segmentation tasks, as it balances the need for accurate pixel-wise classification with the ability to measure overlap between predicted and true segmentation masks. Specifically, the loss function is defined as follows:

$$\mathcal{L} = \lambda_{dice} \cdot \mathcal{L}_{dice} + \lambda_{ce} \cdot \mathcal{L}_{ce} \tag{17}$$

where $\mathcal{L}_{dice}$ is the Dice loss described in section 2.5, $\mathcal{L}_{ce}$ is the cross-entropy loss also described in section 2.5, and λ_{dice}, λ_{ce} are weight parameters that balance the contributions of each component.

The fine-tuning process involved initializing SAM with pre-trained weights and only updating the parameters related to the mask decoder. The rest of the network—comprising the vision encoder and prompt encoder—was kept frozen to prevent overfitting and preserve the generalization capabilities learned from large, diverse datasets. Training was performed over a few epochs, during which the model progressively refined its segmentation capabilities by minimizing the overall loss.

2.3 SAMAug: Augmenting Images with Foundation Model [13]

To address the limitations of SAM in domain-specific segmentation tasks, various approaches have been proposed to adapt its capabilities. Among these, one particularly promising is the SAMAug [13] method introduced in 2023, which serves as the foundation for this thesis. SAMAug improves medical image segmentation by utilizing the segmentation masks and stability scores generated by SAM. Instead of fine-tuning SAM, it employs a parameter-free fusion function to combine SAM's outputs with raw input images, creating augmented inputs for task-specific segmentation models. This method allows the segmentation model to benefit from SAM's foundational strengths while being trained on domain-specific datasets, proving effective across different medical segmentation tasks using both CNNs and Transformers.

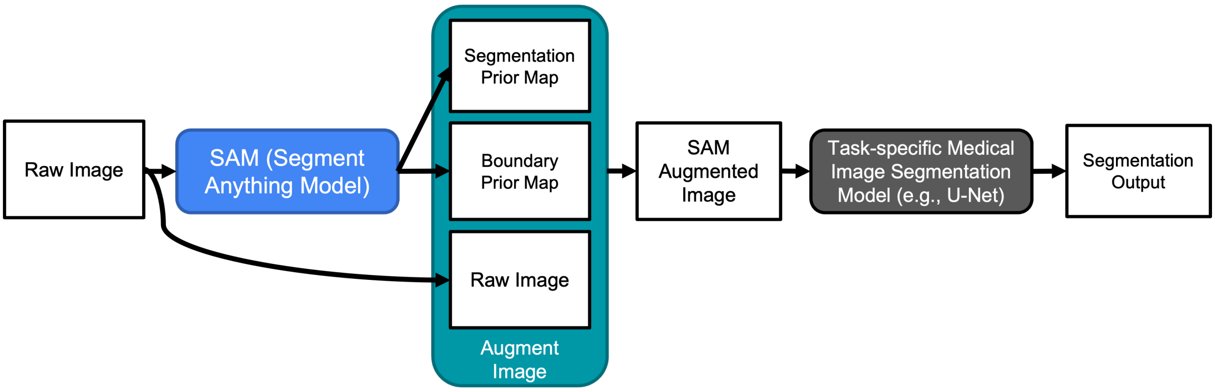


Figure 2.4: Overview of the SAMAug method: integration of SAM-generated segmentation masks with raw input images through a parameter-free fusion function, enhancing task-specific medical image segmentation models. Courtesy of [13]

This section therefore introduces the SAMAug method, detailing how it utilizes segmentation and boundary prior maps generated by SAM to augment input images for medical segmentation tasks. The process begins with the creation of segmentation and boundary prior maps, which are combined with the raw input image to produce augmented inputs. These augmented inputs are then employed to train task-specific segmentation models, incorporating both raw and augmented images into the learning objective. Finally, the section discusses how SAMAug enhances model deployment through ensemble predictions or confidence-based selection, improving segmentation accuracy for domain-specific applications.

2.3.1 Masks Generation

In the process of generating segmentation masks, SAM leverages a grid-based approach to handle a variety of plausible object locations in an image. When given a specific image, SAM produces multiple segmentation masks at different potential positions, each reflecting a possible region of interest. These masks are then stored in a list, each accompanied by a corresponding stability score, which quantifies the confidence of the mask's accuracy. The higher the score, the more reliable the mask is in representing an actual object in the image.

To construct a comprehensive view of the segmented regions, the first step is to generate two key outputs: the *segmentation prior map* and the *boundary prior map*. The segmentation prior map, denoted as $\text{prior}_{\text{seg}}$, combines all individual segmentation masks, weighted by their stability scores. This process aggregates the segmented regions into a unified representation of where objects are most likely to be located within the image.

The boundary prior map, $\text{prior}_{\text{boundary}}$, serves a slightly different purpose. Instead of focusing on the full segmentation, it only considers the outer boundaries of each detected region. These boundaries, when combined, form a map that highlights the edges of the segmented objects, providing additional spatial context to the overall scene.

Mathematically, we can express the *segmentation prior map* as follows:

$$\text{prior}_{\text{seg}} = \sum_{i=1}^{N} \text{stability}_i \cdot M_i \tag{18}$$

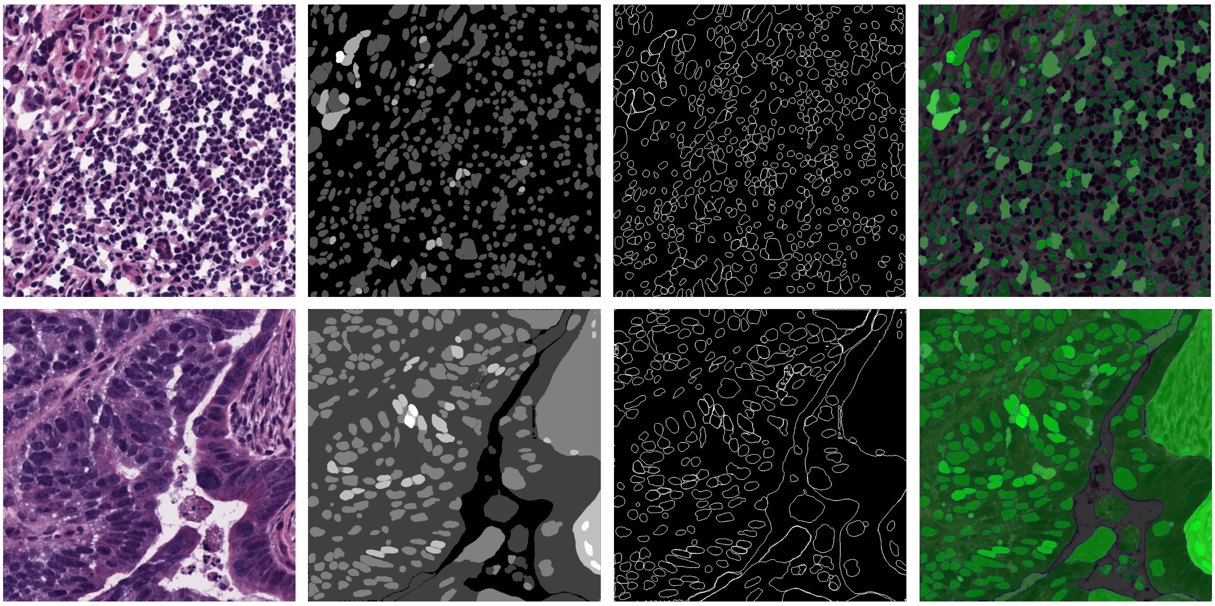
where M_i is the i-th segmentation mask, stability_i is the stability score associated with that mask, and N is the total number of masks. The idea here is that each mask contributes to the final prior map in proportion to its reliability.

For the *boundary prior map*, we look at the boundary of each mask, denoted by ∂M_i. These boundaries are combined to create the final map:

$$\text{prior}_{\text{boundary}} = \sum_{i=1}^{N} \partial M_i \tag{19}$$

This formula shows how the boundaries of all masks are accumulated to form a comprehensive view of the object boundaries within the image.

To help understand the process better, Figure 2.5 provides visual examples of both prior maps. The second column of the figure shows the segmentation prior map, which is a composite of all the masks that SAM generated, with each mask's contribution weighted by its stability score. The third column displays the boundary prior map, where we see only the boundaries of the segmented regions, highlighting the edges of the objects in the image.



Raw
Image Input

Segmentation Prior Map
by SAM

Boundary Prior Map
by SAM

SAM Augmented
Image Input

Figure 2.5: Examples of raw and augmented images used in the segmentation task, from the MoNuSeg dataset. Courtesy of [13]

2.3.2 Augmenting Input Images

Algorithm 1 SAMAug Function

```
**Input:** $tI$ (input image), $mask\_generator$ (SAM model mask generator)
**Output:** $tI$ (augmented image with segmentation and boundary priors)
$masks \leftarrow mask\_generator.generate(tI)$
$SegPrior \leftarrow \text{np.zeros}(tI.\text{shape}[0], tI.\text{shape}[1])$
$BoundaryPrior \leftarrow \text{np.zeros}(tI.\text{shape}[0], tI.\text{shape}[1])$
**for** $maskindex = 0$ **to** $len(masks) - 1$ **do**
    $thismask \leftarrow masks[maskindex].\text{'segmentation'}$
    $stability\_score \leftarrow masks[maskindex].\text{'stability\_score'}$
    $thismask\_binary \leftarrow \text{np.zeros}(thismask.\text{shape})$
    $thismask\_binary[\text{np.where}(thismask == \texttt{True})] \leftarrow 1$
    $indices \leftarrow \text{np.where}(thismask\_binary == 1)$
    $SegPrior[indices] \leftarrow SegPrior[indices] + stability\_score$
    $BoundaryPrior \leftarrow BoundaryPrior + \text{find\_boundaries}(thismask\_binary, \text{mode='thick'})$
    $BoundaryPrior[\text{np.where}(BoundaryPrior > 0)] \leftarrow 1$
**end for**
$tI[: , : ,1] \leftarrow tI[: , : ,1] + SegPrior$
$tI[: , : ,2] \leftarrow tI[: , : ,2] + BoundaryPrior$
**Return** $tI$
```

After generating the segmentation and boundary prior maps, the function proceeds to augment the input image x by incorporating these prior maps, adding information related to regions of interest (ROIs) and boundaries.

More formally, let $x \in \mathbb{R}^{H \times W \times C}$ represent the input image, where H is the image height, W is the width, and C is the number of channels.

Let also $\text{prior}_{\text{seg}} \in \mathbb{R}^{H \times W}$ be the segmentation prior map and $\text{prior}_{\text{boundary}} \in \mathbb{R}^{H \times W}$ be the boundary prior map.

The augmentation process consists of adding these prior maps to the input image's channels. We define the augmented image $x_{\text{aug}} \in \mathbb{R}^{H \times W \times 3}$ with three channels:

1. The first channel is filled with the raw grayscale image x, which is assumed to be in the first channel of the input image:

$$x_{\text{aug}}[:,:,0] = x[:,:,0]$$

2. The second channel is filled with the segmentation prior map $\text{prior}_{\text{seg}}$, enhancing the image with segmentation information:

$$x_{\text{aug}}[:,:,1] = x[:,:,1] + \text{prior}_{\text{seg}}$$

3. The third channel is filled with the boundary prior map $\text{prior}_{\text{boundary}}$, enriching the image with boundary information:

$$x_{\text{aug}}[:,:,2] = x[:,:,2] + \text{prior}_{\text{boundary}}$$

If the input image is in grayscale (i.e., it has only one channel), we expand the image to three channels by directly assigning the prior maps to the second and third channels:

$$x_{\text{aug}}[:,:,0] = x[:,:,0], \quad x_{\text{aug}}[:,:,1] = \text{prior}_{\text{seg}}, \quad x_{\text{aug}}[:,:,2] = \text{prior}_{\text{boundary}}$$

The final augmented image x_{aug} is produced by adding the segmentation and boundary prior maps into the image as described above, resulting in:

$$x_{\text{aug}} = \text{Aug}(\text{prior}_{\text{seg}}, \text{prior}_{\text{boundary}}, x)$$

the details of the implementation are described in pseudocode in Algortihm 1.

2.3.3 Model Training with SAM-Augmented Images

The input augmentation procedure, applied to each image sample in the training set, generates a new set of augmented images and their corresponding annotations. Specifically, for each image

x_i in the original training set, an augmented image $x_{\text{aug},i}$ is created, resulting in the augmented training set $\{(x_{\text{aug},1}, y_1), (x_{\text{aug},2}, y_2), \ldots, (x_{\text{aug},n}, y_n)\}$, where:

$$x_{\text{aug},i} \in \mathbb{R}^{w \times h \times 3}, \quad y_i \in \{0, 1\}^{w \times h \times C}.$$

Here, $x_{\text{aug},i}$ represents the augmented input image with dimensions $w \times h \times 3$ (height, width, and three color channels), and y_i is the corresponding annotation in the form of a binary mask of size $w \times h \times C$, where C denotes the number of classes for the segmentation task, $C = 1$ in case of binary segmentation background/foreground.

To train a segmentation model M, such as a U-Net, from the augmented training set, a typical objective function is employed. This function measures the discrepancy between the model's predictions and the true labels for each augmented sample. The objective function for training with SAM-augmented images is given by the sum of the losses across all n samples:

$$\mathcal{L} = \sum_{i=1}^{n} \text{loss}(M(x_{\text{aug},i}), y_i), \tag{20}$$

where loss represents a suitable loss function, such as cross-entropy or Dice loss, which quantifies the difference between the predicted segmentation map $M(x_{\text{aug},i})$ and the ground truth label y_i.

While the initial model training might solely rely on SAM-augmented images, a more flexible approach can be adopted in scenarios where SAM fails to generate plausible prior maps. In such cases, it becomes beneficial to incorporate both the raw images x_i and their corresponding SAM-augmented versions $x_{\text{aug},i}$ in the training process. This leads to the following modified learning objective, where both raw and augmented images contribute to the model's optimization:

$$\mathcal{L}_{\text{combined}} = \sum_{i=1}^{n} \beta \, \text{loss}(M(x_i), y_i) + \lambda \, \text{loss}(M(x_{\text{aug},i}), y_i), \tag{21}$$

Here, β and λ are hyperparameters that control the relative importance of the loss terms for the raw images and the augmented images, respectively. When $\beta = 0$ and $\lambda = 1$, the objective reduces to the original form given by Equation 20, which only considers the SAM-augmented images. On the other hand, setting both β and λ to 1 allows for a balanced contribution from both raw and augmented images in the training process.

In practice, the choice of β and λ is crucial for optimizing model performance. Typically, both parameters are set to 1 by default, though further experimentation may be required to adjust them depending on the specific dataset or task.

The loss functions used in both formulations, such as the spatial cross-entropy loss or Dice loss, are designed to capture the discrepancy between the predicted and true segmentation masks. These loss functions are minimized using optimization techniques like Stochastic Gradient Descent (SGD), with popular optimizers such as Adam employed to efficiently reduce the loss values during training. Through this optimization process, the model learns to generate accurate segmentation maps for both raw and SAM-augmented images, enhancing its robustness and generalizability.

2.3.4 Model Deployment with SAM-Augmented Images

In the context of segmentation models trained with both raw and SAM-augmented images, new opportunities for enhancing the inference phase arise. One potential approach for utilizing the trained model involves performing inference twice for each test image: once using the raw image and once using its corresponding SAM-augmented version. The final segmentation output can then be derived by combining the results of these two inferences through an averaging ensemble strategy. Mathematically, this process is expressed as:

$$\hat{y} = \tau\left(M(x) + M(x_{\text{aug}})\right), \tag{22}$$

where $M(x)$ represents the segmentation output of the raw image x, and $M(x_{\text{aug}})$ represents the output from the SAM-augmented image. The operator τ denotes a transformation function (e.g., a softmax) applied to the combined outputs, resulting in the final prediction $\hat{y}$.

Alternatively, another method of utilizing both outputs $M(x)$ and $M(x_{\text{aug}})$ involves selecting a single, most plausible segmentation output from the two candidates. This selection is made based on the entropy of the segmentation outputs, with the goal of choosing the prediction that is most certain. The process can be formally described as:

$$\hat{y} = \tau\left(M(x^*)\right), \tag{23}$$

where x^* is the input selected from either x or x_{aug}, depending on which one yields a segmentation output with the least uncertainty. The optimal choice of x^* is determined by minimizing the entropy of the segmentation prediction:

$$x^* = \text{arg} \min_{x' \in \{x, x_{\text{aug}}\}} \text{Entropy}\left(\tau\left(M(x')\right)\right). \tag{24}$$

Entropy, in the context of information theory, is a measure of uncertainty or unpredictability of a system. For a probabilistic classification task, entropy is defined as:

$$\text{Entropy}(p) = -\sum_{i=1}^{C} p_i \log p_i, \tag{25}$$

where p_i represents the probability of the i-th class, and C is the number of classes in the segmentation task. In the case of segmentation, the entropy measures the uncertainty of the predicted class distribution across the pixels in the output, with higher entropy indicating greater uncertainty and lower entropy indicating greater confidence in the prediction.

The entropy function quantifies the uncertainty of the model's predictions, with lower entropy indicating higher certainty in the segmentation output. In practice, a lower entropy score is often correlated with a higher segmentation accuracy. By selecting the image input x^* based on this criterion, the model can potentially improve its prediction by favoring the input version that the model is more confident about.

2.4 Datasets and Methodology

This section provides an overview of the datasets used in our study to evaluate the performance of the model in polyp detection and segmentation tasks. We focus on two primary datasets: Kvasir-SEG and ClinicDB, which were split into training and test sets. Additionally, we include generalization testing on three unseen datasets—ColonDB, ETIS, and Endoscene—to assess the model's adaptability to different data distributions.

After describing the datasets, we will explain the preparation process, including how the data was partitioned and the use of preprocessing techniques. Lastly, we will discuss the data augmentation strategies applied to improve model robustness and generalization, including resizing, multi-scale training, and random transformations, based on techniques introduced in [15].

2.4.1 The Kvasir-SEG Dataset [28]

The Kvasir-SEG dataset is an open-access resource specifically designed for research on polyp segmentation within gastrointestinal (GI) tract imagery, published in 2020. It is an extension of the original Kvasir dataset, with a dedicated focus on pixel-level segmentation of polyps. The dataset consists of meticulously labeled images, all of which have been validated by medical professionals, ensuring both accuracy and reliability. These images represent various segments of the digestive system, capturing both healthy and pathological tissues.

The original Kvasir dataset includes 8,000 images spread across 8 distinct classes, with 1,000 images per class. These images, collected from gastrointestinal endoscopies, feature a variety of conditions, including anatomical landmarks, pathological findings, and endoscopic procedures. It was designed to facilitate classification tasks for the detection of GI tract diseases. To enhance dataset quality, some images from the polyp class were replaced, and all were reviewed by gastroenterologists at Vestre Viken Health Trust in Norway. The Kvasir dataset was employed in the Multimedia for Medicine Challenge (Medico Task) during the MediaEval Benchmarking

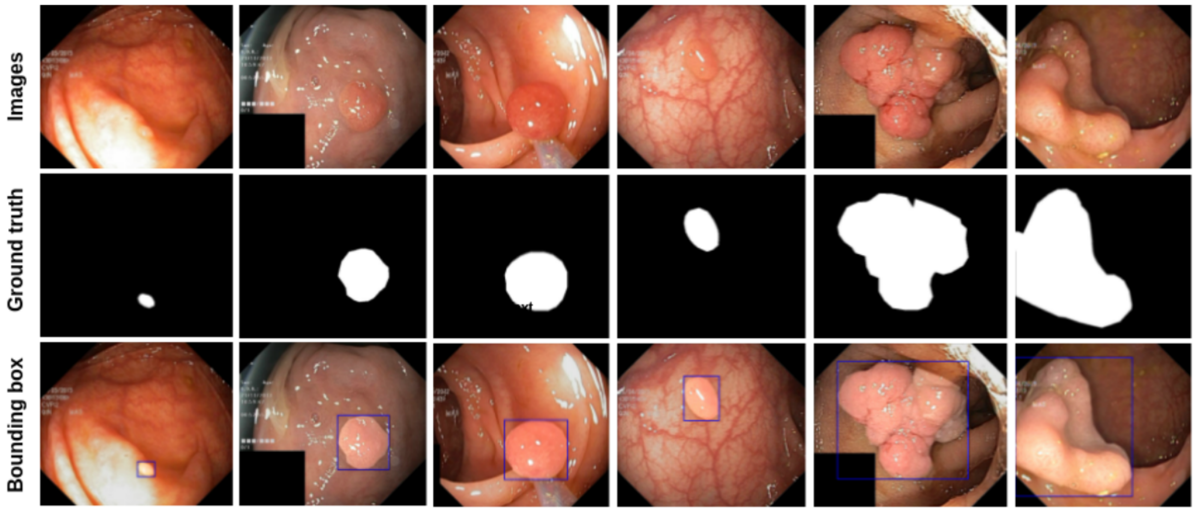


Figure 2.6: Samples from the Kvasir-SEG dataset. Courtesy of [29]

Initiative for Multimedia Evaluation in 2017 and 2018, focusing on multiclass classification of endoscopic findings. However, the dataset provided only frame-level annotations, lacking the pixel-level annotations necessary for segmentation tasks.

The Kvasir-SEG dataset, which is based on the original Kvasir dataset, extends it by providing pixel-level annotations for polyp segmentation. It includes 1,000 annotated images of polyps, captured at resolutions ranging from 720x576 pixels to 1920x1072 pixels. This dataset focuses primarily on the detection and segmentation of polyps, a crucial task for colorectal cancer screening, given that polyps are often precursors to cancer.

The dataset is organized into three main components:

- **Images Folder:** Contains 1,000 images of varying resolutions.
- **Masks Folder:** Includes corresponding binary masks that highlight the polyp regions.
- **JSON File:** Stores bounding box data for the polyps, facilitating the generation of segmentation masks.

Some images also feature an inset display that indicates the endoscope's position within the body during the procedure. The segmentation masks are created through manual tracing by a team of experts, with the pixels corresponding to the polyp region (Region of Interest, ROI) marked in white, while the rest of the image is filled with black. These binary masks distinctly represent the polyp region in white and the background in black.

The generation of the segmentation masks follows a rigorous process:

1. The polyp regions were manually outlined by a team consisting of engineers and medical professionals, using the Labelbox tool for image annotation. The annotations were

subsequently reviewed by an experienced gastroenterologist.

2. Once the annotations were complete, the coordinates of the outlined polyps were exported in a JSON file. These coordinates were then utilized to create the segmentation masks, with the outlines rendered on a black background and filled in with white.

2.4.2 The CVC-ColonDB Dataset [30]

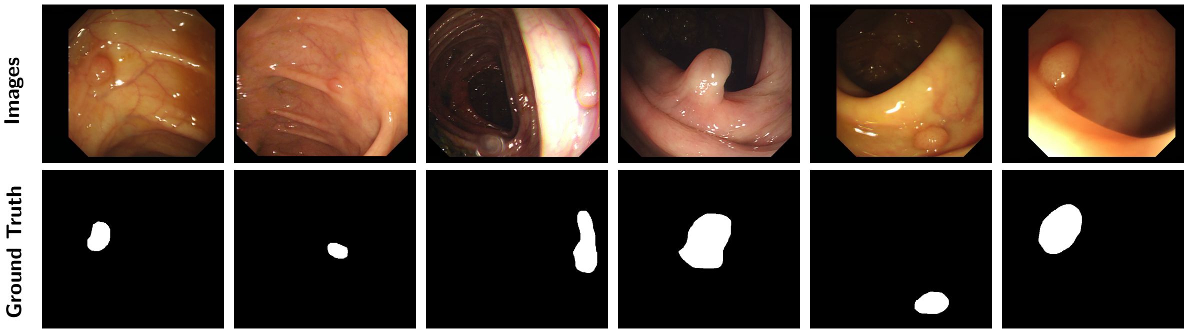


Figure 2.7: Samples from the CVC-ColonDB dataset.

The CVC-ColonDB dataset is a custom-built collection of colonoscopy images designed to evaluate segmentation and detection algorithms for polyps. Released in 2012, this dataset is widely used for benchmarking polyp detection and classification models, offering a diverse set of images with varying polyp appearances.

For this work, we utilized an updated version of the CVC-ColonDB dataset, which contains 380 high-quality images extracted from 15 randomly selected colonoscopy video sequences. Each case represents a different video, with 20 frames chosen per sequence to provide a broad array of viewpoints. The selected frames were picked to avoid redundancy, ensuring each frame presents a distinct perspective.

The images have a resolution of 500x574 pixels, with the central portion cropped to remove irrelevant black borders. The dataset includes only frames that contain polyps, ensuring that it covers a wide range of polyp appearances, including both flat and peduncular types. This careful selection process ensures the dataset represents various polyp forms without bias.

2.4.3 The CVC-ClinicDB Dataset [31]

The CVC-ClinicDB dataset is a publicly available resource published in 2015 designed to aid in the research and development of polyp detection and localization techniques in colonoscopy images. Created in collaboration with the Hospital Clinic of Barcelona, Spain, this dataset consists of 612 high-resolution images (576x768 pixels) sourced from 31 different video sequences captured using standard colonoscopy procedures under white light.

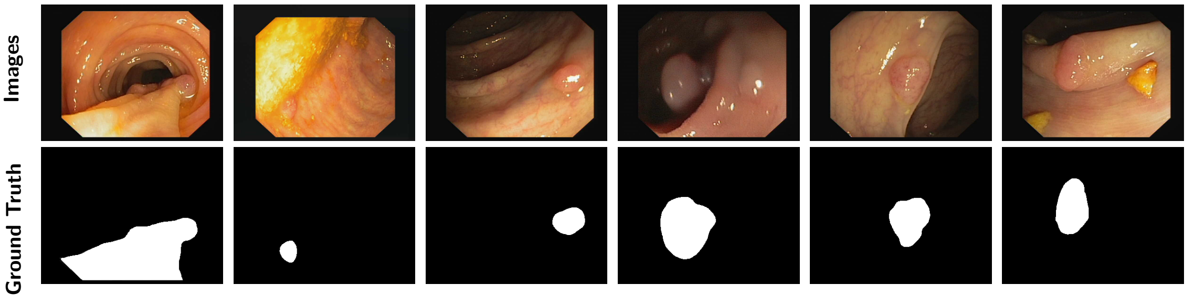


Figure 2.8: Samples from the CVC-ClinicDB dataset.

The dataset includes expert annotations for each polyp, where regions of interest are manually outlined to provide accurate segmentation masks. Additionally, the dataset includes annotations for specular highlights, a feature that complicates polyp detection, and detailed clinical metadata for each polyp. Polyps are categorized by size into three groups: diminutive (≤5 mm), small (6-9 mm), and large (>10 mm), and classified histologically as adenomatous (79.74%) or hyperplastic (20.26%). Furthermore, polyps are classified based on the Paris criteria, which categorizes polyps into six types based on their appearance.

The dataset is structured as follows:

- **Images Folder:** Contains 612 images, each with a resolution of 576x768 pixels.
- **Masks Folder:** Contains binary masks outlining the polyp regions in each image.
- **Specular Highlights Folder:** Includes annotations for light reflections.
- **Clinical Metadata File:** Contains additional clinical information, including polyp size, histological type, and classification according to the Paris criteria.

Additional Information [32]:

- **Purpose:** Training dataset for polyp detection.
- **Institution:** Hospital Clinic, Barcelona, Spain.
- **Content:** 612 SD frames (388×284) from Olympus Q160AL, 31 sequences.
- **Device:** Olympus Q160AL.

This dataset is a crucial resource for researchers focused on polyp detection, localization, and segmentation in colonoscopy images, providing both high-quality annotations and comprehensive clinical data.

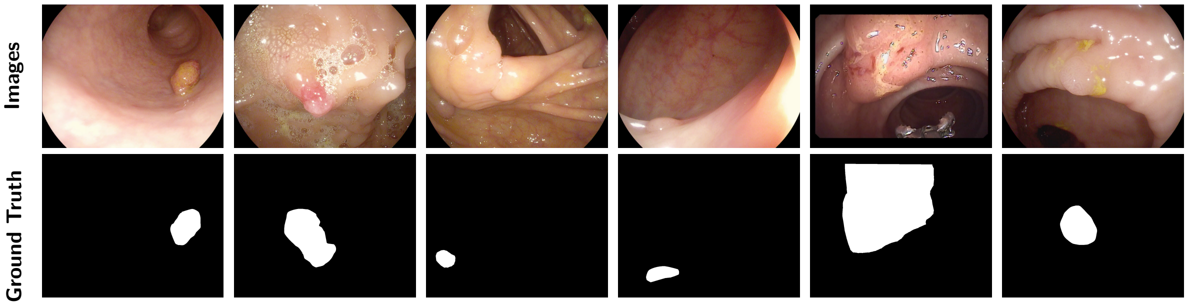


Figure 2.9: Samples from the ETIS-Larib PolypDB dataset.

2.4.4 The ETIS-Larib Polyp Database [33]

The ETIS-Larib Polyp Database (ETIS-Larib DB) is a significant resource for the detection and classification of colorectal polyps during colonoscopy procedures. Developed in 2014, this dataset provides high-resolution, real-world colonoscopy images and has become a benchmark for training and evaluating machine learning models in clinical settings.

The dataset contains 196 annotated images of polyps, captured under unfiltered, real-world colonoscopy conditions. These images, fixed at a resolution of 1225x996 pixels, represent a variety of polyp types and conditions, ensuring diverse and comprehensive data for model training. The ETIS-Larib DB has been widely used in automatic polyp detection research, including in the MICCAI 2015 Sub-Challenge on Automatic Polyp Detection in Colonoscopy Videos.

Though smaller in size compared to other major datasets like Kvasir-SEG and CVC-ClinicDB, the ETIS-Larib DB's focus on unfiltered, real-world imagery makes it an invaluable resource for developing robust polyp detection algorithms.

Additional Information [32]:

- **Purpose:** Testing dataset for polyp detection.
- **Institution:** Lariboisière Hospital, Paris, France.
- **Content:** 196 HD frames (1225×966) from Pentax 90i series, 34 sequences.
- **Device:** Pentax 90i series, Exera II videoprocessor.

2.4.5 The CVC-T Dataset [34]

The CVC-T dataset, also known as CVC-300, is a subset of the larger CVC-EndoSceneStill dataset introduced in 2016, which aims to advance endoluminal scene segmentation. The CVC-EndoSceneStill dataset combines data from the CVC-ColonDB and CVC-ClinicDB collections, resulting in a comprehensive set of 912 images from 44 video sequences across 36 patients.

Dataset	**Number of Samples**	**Resolution**	**Media Type**
ETIS-Larib	196	1225 × 996	Static images
Kvasir-SEG	1000	Variable	Static images
CVC-ClinicDB	612	384 × 288	Static images
CVC-ColonDB	380	574 × 500	Static images

Table 2.1: Comparison of the ETIS-Larib dataset with other key colorectal polyp detection datasets.

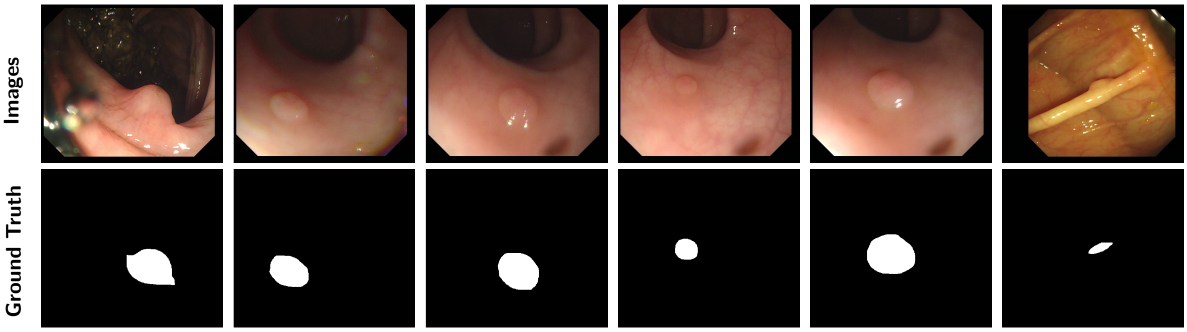


Figure 2.10: Samples from the CVC-T dataset.

The CVC-T dataset, a curated subset, contains 60 high-quality images selected from the CVC-EndoSceneStill collection. These images were specifically chosen to serve as a test set for evaluating polyp detection and segmentation algorithms, covering a range of polyp appearances and other GI conditions. The images have resolutions of 500x574 and 384x288 pixels and include various complexities, including specular highlights and black borders. Annotations include pixel-level masks for the polyp regions, lumen, background, and specular highlights.

The dataset is divided into training, validation, and test sets, with 60% of the images used for training, 20% for validation, and 20% for testing. Importantly, images from individual patients are not split across different sets, ensuring that the dataset can be used to evaluate models on unseen patient data.

The CVC-T dataset is publicly available and plays a vital role in benchmarking polyp detection and segmentation algorithms.

Dataset Preparation

To ensure reliable benchmarking in line with the original study, two datasets, Kvasir-SEG and ClinicDB, were selected for primary evaluation:

- **Kvasir-SEG**: Partitioned into 900 training images and 100 test images.
- **ClinicDB**: Divided into 548 training images and 64 test images.

Additionally, generalization testing was conducted on three unseen datasets—ColonDB, ETIS, and Endoscene—to evaluate the model's ability to adapt to varying data distributions.

To evaluate the performances of our techniques, we conducted experiments on five datasets commonly used in the field.

Preprocessing and Data Augmentation

To standardize inputs and mitigate variability, the following preprocessing and augmentation techniques were applied:

- **Image Resizing**: All input images were resized to 352×352 to match the model's input requirements.
- **Multi-Scale Strategy**: Training was performed at scales of 1.25, 1.0, and 0.75 to improve the model's robustness to scale variations.
- **Random Perspective Transformation**: Applied with a 50% probability to simulate real-world distortions.

We also experimented with a more dynamic data augmentation strategy that comprehends:

- **Multi-Scale Strategy**: Training was performed at scales of 1.25, 1.0, and 0.75 to improve the model's robustness to scale variations.
- **Random Perspective Transformation**: Applied with a 50% probability to simulate real-world distortions.
- **Random Color Adjustment**: Applied with a 20% probability to account for lighting variations across datasets.

These strategies were firstly introduced in [15] under the collective name of `DA3`.

2.5 Evaluation Metrics and Loss Functions

2.5.1 Structure Loss

To ensure high-quality predictions across all stages of the decoder, HSNet make us of a multi-stage joint loss function, which supervises each stage independently while maintaining an overall aggregated loss. The total loss function, defined as structure loss, is defined as:

$$L_{\text{total}} = \sum_{i=1}^{4} L_i, \tag{17}$$

where i indexes the decoder stages. Each stage's loss, L_i, combines two key components: the weighted Binary Cross-Entropy (BCE loss and the weighted Intersection over Union (IoU loss, mathematically formulated as:

$$L_i = L_{\text{IoU}}(P_i, G) + L_{\text{BCE}}(P_i, G), \tag{18}$$

where P_i represents the prediction mask at the i-th stage, and G denotes the ground truth mask.

Weighted Binary Cross-Entropy Loss

The weighted BCE loss aims to emphasize specific pixel regions, particularly boundaries, by assigning a weight w_j to each pixel based on its local context. This is mathematically expressed as:

$$L_{\text{BCE}}(P_i, G) = -\sum_j w_j \Big(G_j \log(P_{i,j}) + (1 - G_j) \log(1 - P_{i,j}) \Big), \tag{19}$$

where G_j is the ground truth label for pixel j, $P_{i,j}$ is the predicted probability at the i-th stage, and w_j adjusts the contribution of each pixel to the loss.

In the implementation, the weight w_j is computed using an average pooling operation over the ground truth mask G, enhancing boundary regions. The weight map is defined as:

$$w_j = 1 + 5 \cdot |\text{AvgPool}(G) - G|,$$

where the pooling operation helps emphasize boundary regions, thus making them more significant during training.

Weighted Intersection over Union Loss

The IoU loss is designed to maximize the overlap between the predicted mask P_i and the ground truth G. For the weighted IoU, the loss is defined as:

$$L_{\text{IoU}}(P_i, G) = 1 - \frac{\sum_j w_j P_{i,j} G_j}{\sum_j w_j \big(P_{i,j} + G_j - P_{i,j} G_j \big)}, \tag{20}$$

where the numerator computes the weighted intersection, and the denominator represents the weighted union.

The weights w_j are the same as those used in the BCE loss, ensuring consistency in the focus on critical regions such as object boundaries.

Practical Implementation

The structure loss combines the two components described above. First, P_i is converted to probabilities using a sigmoid activation, and the intersection and union are computed with weights applied to enhance boundary sensitivity. The final combined loss is averaged across all pixels:

$$L_i = \text{Mean}\Big(L_{\text{BCE}} + L_{\text{IoU}}\Big).$$

The implementation in PyTorch for the structure loss can be written as:

Motivation and Advantages

This loss function leverages both BCE and IoU metrics to optimize segmentation performance. The BCE component ensures pixel-wise classification accuracy, particularly for small or subtle features, meanwhile the IoU component improves structural alignment, encouraging the model to focus on overlapping regions between predictions and ground truth.

By dynamically adjusting the pixel-wise loss through context-dependent weights, this approach prioritizes boundary regions and adapts to various object shapes and sizes.

2.5.2 Q-statistic

In the context of ensemble methods, the Q-statistic serves as an important metric to assess the diversity between models within an ensemble. Diversity is crucial for ensuring that the ensemble benefits from combining models that complement each other, rather than producing redundant predictions. The Q-statistic quantifies the relationship between the predictions of two classifiers by evaluating their agreement and disagreement. It is defined as:

$$Q = \frac{N_{11}N_{00} - N_{10}N_{01}}{N_{11}N_{00} + N_{10}N_{01}}$$

where:

- N_{11}: Number of instances where both models correctly predict the positive class.
- N_{00}: Number of instances where both models correctly predict the negative class.
- N_{10}: Number of instances where the first model predicts the positive class while the second predicts the negative class.
- N_{01}: Number of instances where the first model predicts the negative class while the second predicts the positive class.

The Q-statistic ranges from -1 to 1:

- A value close to 1 indicates that the models are highly correlated and make similar predictions.
- A value near -1 signifies strong disagreement between the models.
- A value near 0 suggests independence between the models.

By leveraging the Q-statistic, researchers can evaluate and ensure sufficient diversity among the models in an ensemble, which is a key factor for boosting overall ensemble performance in tasks such as classification or semantic segmentation.

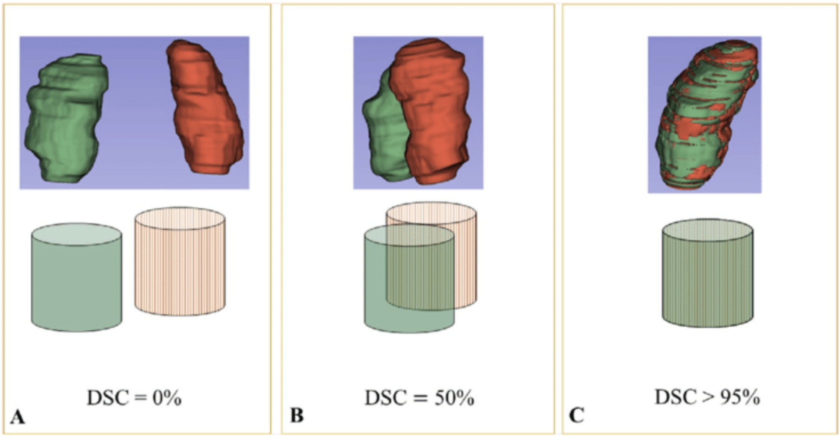


Figure 2.11: This figure visually demonstrates how the DSC (Cylinder Distance Calculation) metric works. It illustrates how the distance between cylinders varies across three different scenes. As the cylinders intersect, the DSC value increases, reflecting a more accurate alignment of the objects within the scenes. Courtesy of [35]

2.5.3 Dice Similarity Coefficient (DSC)

The Dice Similarity Coefficient (DSC) is a widely used metric for evaluating the performance of segmentation algorithms, particularly in biomedical image analysis. It measures the overlap between two sets, typically the predicted segmentation and the ground truth. The formula for the DSC is expressed as:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.1}$$

where $|X|$ and $|Y|$ denote the cardinalities of the two sets being compared. In the context of binary classification tasks, the DSC can be reformulated in terms of true positives (TP), false positives (FP), and false negatives (FN):

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{2.2}$$

This formulation highlights the metric's sensitivity to both precision and recall, making it particularly effective for addressing class imbalance—a common challenge in medical imaging.

In deep learning applications, particularly for segmentation tasks, the DSC serves two critical purposes: as a performance evaluation metric and as a differentiable loss function. Its differentiability allows for efficient optimization during training, providing an advantage over non-differentiable metrics such as the Intersection over Union (IoU). Using the DSC as a loss function enables models to directly optimize for segmentation accuracy.

The DSC is especially significant in biomedical imaging, where precise segmentation is essential. For instance, it is commonly applied to evaluate the quality of contour delineations in radiotherapy planning, ensuring accurate identification of organs-at-risk. Recent studies have demonstrated that deep learning models can predict DSC values with high reliability, thereby supporting improved clinical decision-making.

Chapter 3

SegAug: Augmentations and Ensembles

In this section, we present a series of methods aimed at enhancing image processing workflows by incorporating segmentation information directly into the image content. These methods are designed to improve both the analytical and visual properties of images, enabling more effective applications in areas such as object detection, image segmentation, and visual data analysis. The first method applies Principal Component Analysis (PCA) to reduce the dimensionality of images while simultaneously integrating segmentation priors to enhance semantic representation. The second and third methods focus on modifying the blue channel of an image by replacing it with segmentation logits or masks generated by the SAM model, embedding regions of interest identified by the segmentation process. Lastly, the fourth method alters the hue channel in the HSV color space by incorporating stability scores from segmentation masks, emphasizing semantic regions within the image's color structure.

All of these methods uses segmentation masks to modify image features, offering distinct yet complementary approaches to improving image analysis. By combining dimensionality reduction, logit-based channel embedding, and HSV-based segmentation prior integration, these methods form a comprehensive toolkit for advancing image processing workflows. These approaches are particularly relevant in fields such as medical imaging, computer vision, and autonomous systems, where precise feature extraction and semantic enrichment are critical. The following sections provide a detailed explanation of each method, highlighting their individual contributions to improving image segmentation and feature extraction.

3.1 SegPrior RGB

The Segmentation Prior Modification algorithm is a variant of the previous method designed to modify the blue channel of an image based on segmentation information generated by a mask generator. The algorithm computes a segmentation prior matrix, which aggregates the stability scores of different segmentation masks, and integrates this prior information into the blue channel of the image. This approach is particularly useful when enhancing images with spatial information from segmentation, while preserving the color structure of the original image. The method is applicable in areas such as image segmentation, visual enhancement, and object detection, where both the image content and segmentation results play a crucial role in analysis.

3.1.1 Steps Breakdown

1. **Segmentation Mask Generation**: The algorithm begins by generating segmentation masks for the input image using a mask generator. Each mask represents a region of interest within the image, and each is associated with a stability score that indicates the confidence of the segmentation.

2. **Segmentation Prior Calculation**: The $SegPrior$ matrix is initialized as an empty matrix. The algorithm iterates over each segmentation mask, modifying the $SegPrior$ by adding the stability score for regions where the mask indicates the presence of a feature or object. This results in a prior that reflects the confidence in various regions of the image.

3. **Channel Separation**: The image is separated into its red, green, and blue channels. The red and green channels are kept unchanged, while the blue channel will be modified to reflect the segmentation prior.

4. **Segmentation Prior Integration**: The modified blue channel is replaced with the $SegPrior$, which has been calculated from the segmentation masks. This modification injects the segmentation information into the image, highlighting areas of interest as determined by the segmentation model.

5. **Image Reconstruction**: The final step involves reconstructing the image by combining the unchanged red and green channels with the modified blue channel. The resulting image now reflects both the original image content and the additional segmentation information.

3.1.2 Usefulness of the Algorithm

This algorithm is particularly useful in scenarios where segmentation information is crucial for image analysis, but the original blue channel of the image does not provide significant information. By modifying the blue channel with the segmentation prior, the method enhances the visualization of relevant features and regions.

Algorithm 2 Segmentation Prior Modification on RGB Channels

```
procedure RG_segPrior($tI, mask\_generator$)
    $masks \leftarrow$ mask_generator.generate($tI$)
    $SegPrior \leftarrow$ np.zeros($tI.shape[0], tI.shape[1]$)
    for each mask in masks do
        $SegPrior \leftarrow SegPrior +$ mask.stability_score $\cdot$ mask.segmentation
    end for
    $r, g \leftarrow tI[:, :, 0], tI[:, :, 1]$
    $b \leftarrow SegPrior$
    $modded\_image \leftarrow$ np.dstack($r, g, b$)
    return $modded\_image$
end procedure
```

3.2 SegPrior-Logits RGB

The Logits-based Modification algorithm is a method used to modify the blue channel of an image based on segmentation logits produced by the SAM model. This technique is particularly useful when integrating segmentation results directly into the image, offering a visual representation of the segmentation model's predictions while preserving the original image content. Such an approach can be applied in fields like image segmentation, visual data analysis, and object detection, where combining image data with segmentation information provides enhanced insights.

3.2.1 Steps Breakdown

1. **Input Image Preparation**: The algorithm starts by setting the input image to the SAM model, preparing it for segmentation prediction. This allows the model to process the image and generate relevant outputs, such as the segmentation mask and its corresponding logits.

2. **Segmentation Logit Prediction**: The SAM model predicts the segmentation mask, returning the mask along with its associated logits. These logits represent the model's confidence in different regions of the image, highlighting areas where specific features or objects are present.

3. **Channel Separation**: The algorithm then separates the image into its individual color channels—red, green, and blue—where the red and green channels are retained as-is. The blue channel is set to be modified based on the segmentation logits.

4. **Logit Normalization**: The logits are normalized to fit within the standard image pixel range of [0, 255]. This ensures that the logits can be represented as valid pixel values and

applied to the blue channel of the image.

5. **Blue Channel Replacement**: The normalized logits replace the original blue channel of the image, resulting in a new image where the blue channel now reflects the segmentation information. This modification highlights the areas of interest as determined by the segmentation model.

6. **Image Reconstruction**: The final step involves reconstructing the image by combining the unchanged red and green channels with the modified blue channel. The resulting image now displays the segmentation information while maintaining the original color integrity.

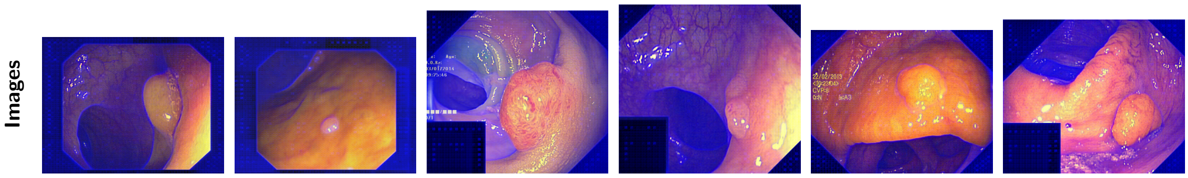


Figure 3.1: Examples of SegPrior-Logits RGB augmented images.

3.2.2 Usefulness of the Algorithm

This method is especially beneficial in scenarios where semantic features are important, meanwhile the blue channel of the image is not informative.

Algorithm 3 Replace Blue Channel with SAM Logits

```
procedure RG_logits($tI, mask\_generator$)
    $mask\_generator.predictor.set\_image(tI)$
    $mask, \_, logits \leftarrow$ mask_generator.predictor.predict$(return\_logits = True)$
    $r, g, b \leftarrow tI[:,:,0], tI[:,:,1], tI[:,:,2]$
    $b \leftarrow \frac{(mask[0]-\min(mask[0]))\times 255}{\max(mask[0])-\min(mask[0])}$
    $modded\_image \leftarrow$ np.dstack$(r, g, b)$
    return $modded\_image$
end procedure
```

3.3 SegPrior HSV

The HSV-based Segmentation Prior Modification algorithm is a method designed to enhance the semantic representation of an image by modifying its hue channel in the HSV color space using segmentation priors. This approach leverages stability scores from segmentation masks to create a semantic prior that highlights areas of interest while preserving the saturation and value channels. The method is particularly useful for tasks like object detection, visual data analysis, and feature extraction in computer vision.

3.3.1 Steps Breakdown

1. **Input Image Preparation**: The input image is converted to a floating-point format to prepare it for processing. This ensures the compatibility of pixel values with subsequent operations.

2. **Segmentation Prior Generation**: Using the segmentation masks generated by the SAM model, a semantic prior map is created. For each mask, the segmentation region is weighted by its stability score, and these weights are accumulated across all masks to produce the segmentation prior.

3. **Color Space Conversion**: The image is converted from RGB to HSV color space. This transformation separates the image into hue (H), saturation (S), and value (V) channels, allowing for isolated modifications to the hue channel.

4. **Channel Separation and Modification**: The segmentation prior is assigned to the hue channel (H), effectively encoding the semantic information in the hue component of the HSV image. The saturation (S) and value (V) channels remain unchanged.

5. **Image Reconstruction**: The modified hue channel is recombined with the original saturation and value channels to reconstruct the modified HSV image. The resulting image integrates semantic information directly into its color representation.

6. **Output Image Generation**: The processed image, with its hue channel reflecting the segmentation prior, is returned for further analysis or visualization.

3.3.2 Usefulness of the Algorithm

This algorithm is particularly effective in scenarios where the hue channel can serve as a visual representation of semantic information. By embedding segmentation priors into the hue channel, the algorithm provides an intuitive way to analyze and visualize areas of interest, making it applicable in fields like medical imaging, autonomous systems, and environmental monitoring.

Algorithm 4 Replace H Channel with Segmentation Prior

```
procedure SV_segPrior($tI, mask\_generator$)
    $masks \leftarrow$ mask_generator.generate($tI$)
    $SegPrior \leftarrow$ zeros($tI.shape[0], tI.shape[1]$)
    for all $mask \in masks$ do
        $SegPrior \leftarrow SegPrior +$ mask['segmentation'] $\times$ mask['stability_score']
    end for
    $hsv\_image \leftarrow$ color.rgb2hsv($tI$)
    $h, s, v \leftarrow SegPrior, hsv\_image[:, :, 1], hsv\_image[:, :, 2]$
    $modded\_image \leftarrow$ np.dstack($s, v, h$)
    return $modded\_image$
end procedure
```

3.4 SegPrior PCA

The PCA and Segmentation Prior Modification algorithm is a process used to enhance image features by reducing dimensionality through Principal Component Analysis (PCA) and incorporating segmentation information to modify the image channels. This algorithm can be particularly useful in applications such as image segmentation, object detection, and visual data analysis, where both spatial patterns (in the image) and semantic features (from segmentation) are essential.

3.4.1 Steps Breakdown

1. **Segmentation Mask Generation**: The algorithm begins by generating segmentation masks using a mask generator. These masks represent regions of interest within the image, and each mask is associated with a stability score, which indicates the confidence in the mask's correctness or reliability.

2. **Segmentation Prior Calculation**: The $SegPrior$ matrix is initialized as a blank matrix (zeros), and for each segmentation mask, the algorithm updates $SegPrior$. This update involves multiplying each mask by its corresponding stability score and accumulating the results. The segmentation prior thus reflects the confidence in various regions of the image based on the available segmentation masks.

3. **Dimensionality Reduction via PCA**: The image is then passed through a Principal Component Analysis (PCA) to reduce its dimensionality from 3 channels (RGB) to 2 channels. PCA is a statistical technique that finds the most significant directions of variance in the data and projects the image data onto those directions. The reduced representation captures the key features of the image in fewer dimensions, making it more efficient for

further processing.

4. **Scaling of PCA and Segmentation Data**: The PCA output and segmentation prior are scaled to the range [0, 255] to ensure that the data is properly adjusted for visualization and processing. The scaling step is essential because it normalizes the values, allowing for consistent interpretation and manipulation of the image channels.

5. **Image Reconstruction**: The final step reconstructs the image by combining the scaled PCA results and the segmentation prior into the red, green, and blue channels of the image. The red and green channels come from the PCA output, while the blue channel is influenced by the segmentation prior. This modified image now encodes both the reduced-dimensional representation of the image and the additional information from the segmentation.

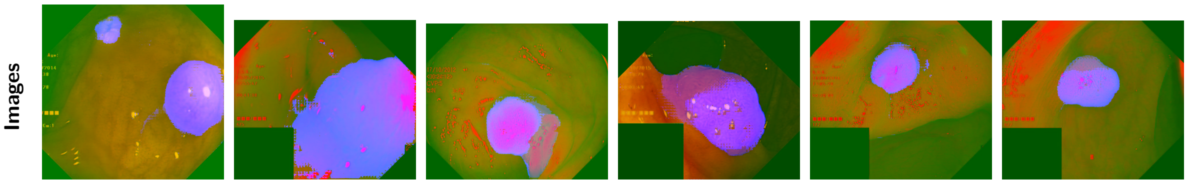


Figure 3.2: Examples of SegPrior PCA augmented images.

3.4.2 Usefulness of the Algorithm

This algorithm is beneficial in scenarios where image analysis needs to be enhanced by incorporating segmentation information. For example:

- **Image Segmentation**: The segmentation prior ($SegPrior$) improves the analysis by injecting spatially relevant data that reflects the structure and boundaries of objects in the image. By modifying the color channels based on segmentation, the image highlights areas of interest, making further analysis or feature extraction more accurate.

- **Dimensionality Reduction**: Reducing the dimensionality of the image using PCA can significantly simplify computational tasks, especially when processing large datasets or when the main features of the image can be captured in fewer dimensions. It also helps in reducing noise by focusing on the principal components of the image.

- **Visual Enhancement**: The combination of PCA and segmentation can also aid in visual enhancement, where segmentation-prioritized areas are highlighted in the image, making it easier to interpret or present visually.

This approach can be applied to various domains, such as medical imaging, computer vision, autonomous systems, and image-based machine learning tasks, where both feature extraction and segmentation are crucial for success.

Algorithm 5 PCA and Segmentation Prior Modification

```
procedure PCA_segPrior($tI, mask\_generator$)
    $masks \leftarrow$ mask_generator.generate($tI$)
    $SegPrior \leftarrow$ np.zeros($tI.shape[0], tI.shape[1]$)
    for each mask in masks do
        $thismask \leftarrow$ mask.segmentation
        $stability\_score \leftarrow$ mask.stability_score
        $SegPrior \leftarrow SegPrior +$ thismask $\times\ stability\_score$
    end for
    $pca \leftarrow$ PCA($n\_components = 2$)
    $image\_pca \leftarrow pca.fit\_transform(tI)$
    $image\_pca\_scaled \leftarrow$ scale($image\_pca$)
    $r, g \leftarrow image\_pca\_scaled[:, 0], image\_pca\_scaled[:, 1]$
    $b \leftarrow$ scale($SegPrior$)
    $modded\_image \leftarrow$ np.dstack($r, g, b$)
    return $modded\_image$
end procedure
```

3.5 Ensemble Methods

Ensemble methods are a powerful approach in machine learning, leveraging the predictions of multiple models to achieve better performance and robustness. By combining the strengths of different models or augmenting strategies, ensembles aim to mitigate the weaknesses of individual models and enhance overall accuracy. In this section, we discuss the various ensemble strategies implemented in this work, focusing on their design and how they integrate the novel methods described earlier.

3.5.1 Overview of Ensemble Approaches

To evaluate the impact of combining models trained with different strategies, we explored several ensemble configurations. Each configuration utilized models trained on specific data augmentation strategies and incorporated the novel algorithms presented in this thesis. The ensemble approaches tested include the following:

1. **DA Baseline Ensemble**: This ensemble serves as the control, comprising three HSNet models trained with data augmentation as the sole augmenting strategy. It represents the

performance achievable without integrating any of the new algorithms described in this work.

2. **SAMAug Ensemble**: This configuration builds on the DA3 Baseline Ensemble by training three HSNet models on SAMAug images generated using the SAM2 model. This ensemble explores the benefits of augmentations based on SAM2-produced segmentation masks.
3. **AuxMix Triple**: This ensemble combines one model trained with each of the following algorithms: DA3 Baseline, Logits-based Modification Algorithm, and SAMAug with SAM2. It aims to evaluate whether diversity in model training approaches can enhance ensemble performance.
4. **AuxMix Ninefold**: Extending AuxMix Triple, this ensemble includes three models for each strategy, resulting in a total of nine models (3x DA3 Baseline, 3x Logits-based Modification Algorithm, and 3x SAMAug with SAM2). This configuration evaluates the impact of increased model diversity within each algorithm.
5. **Segmentation-Prior Ensemble**: This ensemble includes one model from each of the following algorithms: SAMAug with SAM2, Segmentation Prior Modification Algorithm using SAM2, and DA3 Baseline. It tests the combination of models trained with segmentation logits and traditional augmentation strategies.
6. **Semantic Ensemble**: Similar to Segmentation-Prior Ensemble, this ensemble consists of one model for each algorithm, but replaces the DA3 Baseline with the Logits-based Modification Algorithm. This configuration assesses the performance of ensembles relying exclusively on segmentation-based augmentations.

Chapter 4

Experiments and Results

This section presents the outcomes of the conducted experiments and provides an analysis of the results.

4.1 Model Training with SAM-Augmented Images

We first experimented with the two different training approaches detailed in Section 2.3.3: one based solely on SAM-augmented images (Equation 20), and another combining both raw and SAM-augmented images (Equation 21). The second approach, which incorporates both raw and augmented images, yielded the best performance as shown in Figure 4.1.

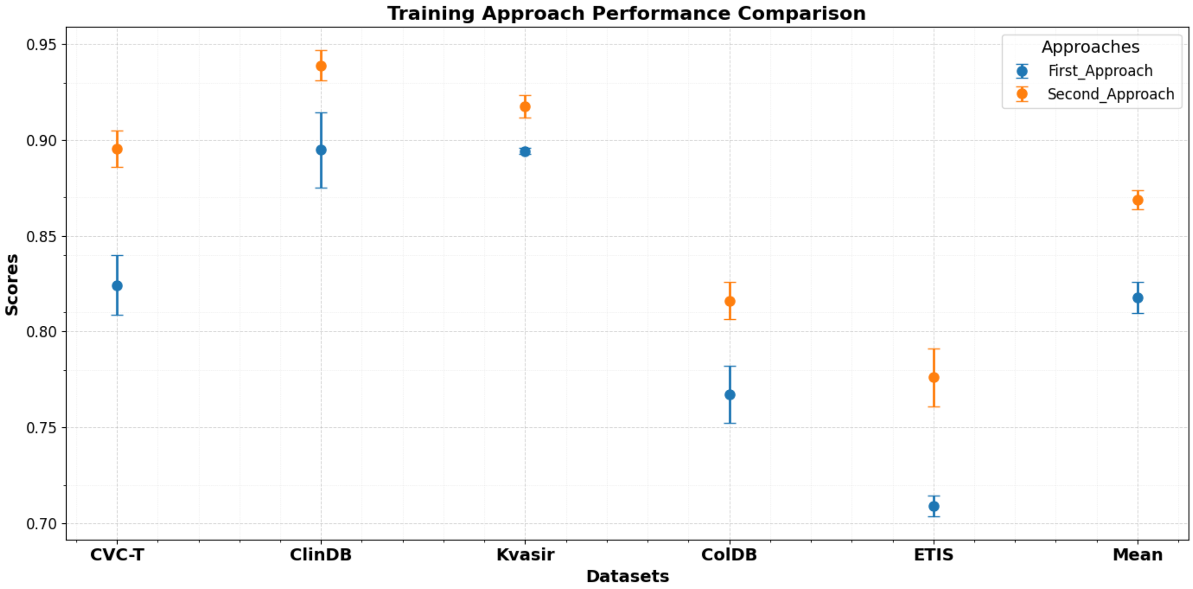


Figure 4.1: Comparison of model performance across multiple datasets for two training approaches. The plot shows the mean performance for each approach, with error bars representing the sample standard deviation over three test runs.

4.2 Segmentation Prior Augmentation Evaluation

Building on the findings from the SAMAug evaluation, we observed that the first approach led to a decline in performance, indicating that the initial enhancement did not have the desired impact. However, the second approach, which incorporated an entropy-based selection method, showed promising results. This approach suggested a more effective strategy for improving segmentation accuracy, as it appeared to address key limitations identified in the first approach.

Encouraged by the success of the entropy-based method, we decided to explore further improvements by testing the new set of enhancements, as outlined in 3. By maintaining consistency in our experimental setup, we were able to make a more direct comparison between the new and previous methods.

Method	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
No DA	0.886 ± 0.009	0.930 ± 0.005	0.925 ± 0.005	0.799 ± 0.009	0.806 ± 0.008	0.869 ± 0.002
DA	0.885 ± 0.007	0.933 ± 0.005	**0.927 ± 0.005**	0.793 ± 0.015	**0.811 ± 0.009**	0.870 ± 0.004
SAM2Aug	0.895 ± 0.005	0.939 ± 0.006	0.921 ± 0.006	0.817 ± 0.008	0.783 ± 0.006	0.871 ± 0.004
SAMAug	0.895 ± 0.010	0.939 ± 0.008	0.918 ± 0.006	0.816 ± 0.010	0.776 ± 0.015	0.869 ± 0.005
SegPriorHSVSAM2	**0.899 ± 0.005**	0.932 ± 0.009	0.912 ± 0.005	0.804 ± 0.007	0.763 ± 0.015	0.862 ± 0.005
SegPriorHSV	0.899 ± 0.006	0.936 ± 0.005	0.912 ± 0.004	0.796 ± 0.011	0.783 ± 0.008	0.865 ± 0.002
SegPriorLogitsSAM	0.890 ± 0.007	0.935 ± 0.008	0.920 ± 0.002	0.818 ± 0.007	0.794 ± 0.006	**0.871 ± 0.003**
SegPriorLogits	0.889 ± 0.010	0.932 ± 0.008	0.918 ± 0.005	**0.819 ± 0.005**	0.797 ± 0.006	0.871 ± 0.004
SegPriorPCASAM2	0.889 ± 0.006	0.940 ± 0.006	0.909 ± 0.005	0.810 ± 0.006	0.771 ± 0.012	0.864 ± 0.003
SegPriorPCA	0.895 ± 0.007	**0.943 ± 0.004**	0.913 ± 0.004	0.819 ± 0.010	0.760 ± 0.008	0.866 ± 0.003
SegPriorRGB	0.892 ± 0.004	0.934 ± 0.007	0.918 ± 0.005	0.807 ± 0.010	0.790 ± 0.005	0.868 ± 0.004

Table 4.1: Dice score coefficients (Mean ± Standard Deviation) for different methods on five datasets: CVC-T, ClinDB, Kvasir, ColDB, and ETIS.

Overall, both tables 4.1 4.2 provide valuable insights into how different segmentation prior augmentations impact performance, offering a clear view of the trade-offs involved in selecting the most effective method for specific datasets or tasks. The choice of presenting the results with mean scores and standard deviations, along with the detailed performance breakdown for each dataset, ensures that both the overall effectiveness and the consistency of the methods are thoroughly evaluated.

4.3 Ensemble Strategy Evaluation

In this section, we focus on the evaluation of the ensemble strategy, building on the insights gained from the Q-statistic comparison presented in Table 4.3. This table compares the performance of three different approaches: the baseline method (denoted as ”Baseline da”), the HSNet model enhanced with logits from SAM1 (”SAM1 logits”), and the SAMAug method using SAM2 (”SAM2 SAMaug”).

The Q-statistic, as shown in Table 4.3, reflects the relative performance of these approaches,

Method	**POLYP**
SegPrior-Logits RGB (wSAM2)	0.871
SAMAug (wSAM2)	0.871
SegPrior-Logits RGB (wftSAM)	0.871
Baseline (wDA)	0.870
Baseline	0.869
SAMAug (wftSAM)	0.869
SegPrior RGB (wftSAM)	0.868
SegPrior PCA (wftSAM)	0.866
SegPrior HSV (wftSAM)	0.865
SegPrior PCA (wSAM2)	0.864
SegPrior HSV (wSAM2)	0.862

Table 4.2: Methods ordered by performance on the Polyp dataset, ranked by average Dice score over 10 runs. Higher scores indicate better segmentation performance. In the table, `wftSAM` refers to ”with a fine-tuned version of SAM” and `wSAM2` refers to ”with SAM2”.

	Baseline da	**SAM1 logits**	**SAM2 SAMaug**
Baseline da	0.9945	–	–
SAM1 logits	0.9867	0.9918	–
SAM2 SAMaug	0.9885	0.9837	0.9910

Table 4.3: Q-statistic comparison across different approaches: Baseline da3, SAM1 logits, and SAM2 SAMaug.

and the results reveal distinct differences between them. Notably, the baseline approach performs well with a Q-statistic value of 0.9945, serving as a reference point for comparison. Meanwhile, the integration of SAM1 logits leads to a slight improvement, with a Q-statistic of 0.9918, demonstrating the effectiveness of adding logits from SAM1 to the baseline. The best performance, however, is achieved with SAM2 SAMaug, which reaches a Q-statistic of 0.9910, slightly outperforming SAM1 logits.

These results suggest that using a combination of methods, such as SAM1 and SAM2, could lead to further performance improvements. This opens the door to exploring ensemble methods, which combine multiple models to achieve superior results by leveraging their complementary strengths. The subsequent analysis, will explore how these ensemble strategies can enhance the overall segmentation performance.

Ensemble	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
AuxMix Ninefold	**0.904 ± 0.002**	0.938 ± 0.007	0.926 ± 0.002	**0.843 ± 0.002**	**0.829 ± 0.002**	**0.888 ± 0.002**
AuxMix Triple	0.900 ± 0.003	0.939 ± 0.008	0.924 ± 0.003	0.836 ± 0.005	0.818 ± 0.006	0.884 ± 0.003
DA Baseline	0.903 ± 0.002	0.921 ± 0.007	0.915 ± 0.002	0.818 ± 0.005	0.827 ± 0.004	0.877 ± 0.001
SAMAug	0.896 ± 0.003	0.941 ± 0.005	**0.927 ± 0.003**	0.825 ± 0.005	0.792 ± 0.006	0.876 ± 0.002
Segmentation-Prior	0.902 ± 0.003	**0.943 ± 0.006**	0.919 ± 0.004	0.835 ± 0.004	0.816 ± 0.007	0.883 ± 0.002
Semantic	0.895 ± 0.006	0.941 ± 0.008	0.924 ± 0.005	0.828 ± 0.005	0.796 ± 0.009	0.877 ± 0.003

Table 4.4: Dice score coefficients (Mean ± Standard Deviation) for different methods on five datasets: CVC-T, ClinDB, Kvasir, ColDB, and ETIS. The methods tested include: **AuxMix Ninefold**, which consists of an ensemble of 9 models; **AuxMix Triple**, which is based on an ensemble of 3 models; and **DA Baseline**, **SAMAug Ensemble**, **Segmentation-Prior**, and **Semantic**, each formed by an ensemble of 3 models. Each method's performance is averaged over 10 independent tests.

Method	POLYP
AuxMix Ninefold	0.888
AuxMix Triple	0.884
Segmentation-Prior Ensemble	0.883
Semantic Ensemble	0.877
Baseline Ensemble	0.877
SAMAug Ensemble	0.876

Table 4.5: Performance of different ensemble strategies, ordered by descending average Dice score. Higher scores indicate better segmentation results. All strategies are made with three models, except for the AuxMix Ninefold.

Method	**CVC-T**	**ClinDB**	**Kvasir**	**ColDB**	**ETIS**	**Average**
AuxMix	0.904	0.938	0.926	**0.843**	0.829	**0.888**
Ens2	0.899	0.935	0.927	0.840	**0.833**	0.887
HSNet	0.903	0.948	0.926	0.810	0.808	0.879
MIA-Net	0.900	0.942	0.926	0.816	0.800	0.877
P2T	0.879	0.923	0.905	0.761	0.700	0.834
DBMF	**0.919**	0.933	**0.932**	0.803	0.790	0.875
HarDNet	0.887	0.932	0.912	0.731	0.677	0.828
PraNet	0.871	0.899	0.898	0.709	0.628	0.801
SFA	0.467	0.700	0.723	0.469	0.297	0.531
U-Net++	0.707	0.794	0.821	0.483	0.401	0.641
U-Net	0.710	0.823	0.818	0.512	0.398	0.652
SETR	0.889	0.934	0.911	0.773	0.726	0.847
TransUnet	0.893	0.935	0.913	0.781	0.731	0.851
TransFuse	0.894	0.942	0.920	0.781	0.737	0.855
UACANet	0.910	0.926	0.912	0.751	0.751	0.850
SANet	0.888	0.916	0.904	0.753	0.750	0.842
MSNet	0.869	0.921	0.907	0.755	0.719	0.834
Polyp-PVT	0.900	0.937	0.917	0.808	0.787	0.869
SwinE-Net	0.906	0.938	0.920	0.804	0.758	0.865
AMNet	-	0.936	0.912	0.762	0.756	-
MGCBFormer	0.913	**0.955**	0.931	0.807	0.819	0.885

Table 4.6: Performance of our best ensemble strategy, compared with the proposed models in the literature.

Chapter 5

Conclusion

This work demonstrates the potential of ensemble methods to significantly improve segmentation performance, particularly when combining models trained with diverse augmentation strategies. Each augmentation provides unique training information, enabling the ensemble to better understand different sets of image features. By integrating complementary models, the ensemble approach enhances segmentation accuracy and robustness, as the models capture various aspects of the data.

While the current results show a slight improvement in performance with HSNet, there is considerable room for optimization. Future work should focus on tailoring augmentation strategies to target the specific characteristics of the dataset. This refinement could yield even better results, as augmentation directly influences how well the model adapts to the images at hand.

Additionally, this study opens the door for further exploration of different segmentation architectures. Expanding the ensemble beyond HSNet, by including other segmentators with varying architectures, could introduce more diversity in the learned features, leading to further improvements in segmentation accuracy. Fine-tuning SAM2 is another potential direction to enhance performance.

The success of ensemble methods in segmentation tasks also points to future opportunities for optimization. For instance, model selection could be improved by weighing the models based on their individual performance or dynamically choosing models based on the input characteristics. Such optimizations would enable the ensemble to adapt more effectively to varying input data, improving overall model performance in real-world scenarios.

The integration of ensemble methods into real-world applications, such as medical imaging and autonomous systems, could significantly enhance segmentation reliability and accuracy. In these critical fields, the ability to produce precise and reliable segmentation results is paramount. By refining ensemble strategies and exploring further innovations in segmentation architectures and augmentation techniques, we can make strides toward more effective and trustworthy segmentation systems.

In conclusion, this research underscores the power of ensemble methods in segmentation tasks. By combining models trained with different augmentations and embedding strategies, we have demonstrated a significant improvement in segmentation performance. With further fine-tuning and exploration of new models and strategies, these methods hold great promise for enhancing segmentation in a variety of domains.

Appendix A: Detailed Tables

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.9016	0.9262	0.9231	0.8035	0.7992	0.8707
2	0.8834	0.9354	0.9209	0.8029	0.8015	0.8688
3	0.8920	0.9289	0.9242	0.7877	0.8213	0.8708
4	0.8861	0.9257	0.9263	0.8049	0.8033	0.8693
5	0.8690	0.9356	0.9279	0.8166	0.7995	0.8697
6	0.8917	0.9384	0.9197	0.7903	0.8037	0.8687
7	0.8844	0.9243	0.9332	0.7947	0.7951	0.8663
8	0.8850	0.9348	0.9328	0.7971	0.8138	0.8727
9	0.8799	0.9275	0.9218	0.7871	0.8102	0.8653
10	0.8891	0.9240	0.9234	0.8044	0.8082	0.8698
Avg.	0.8862	0.9301	0.9253	0.7989	0.8056	0.8692

Table 1: Dice score coefficients for all ten different HSNet with the default data augmentation

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8721	0.9321	0.9238	0.8152	0.8098	0.8706
2	0.8783	0.9321	0.9302	0.7980	0.7981	0.8673
3	0.8948	0.9267	0.9268	0.7831	0.8049	0.8673
4	0.8868	0.9283	0.9309	0.7621	0.7993	0.8615
5	0.8899	0.9363	0.9301	0.7921	0.8235	0.8744
6	0.8835	0.9328	0.9348	0.8065	0.8042	0.8724
7	0.8829	0.9446	0.9266	0.7910	0.8219	0.8734
8	0.8810	0.9271	0.9181	0.8029	0.8149	0.8688
9	0.8843	0.9339	0.9284	0.7993	0.8194	0.8731
10	0.8934	0.9324	0.9239	0.7817	0.8120	0.8687
Avg.	0.8847	0.9326	0.9274	0.7932	0.8108	0.8697

Table 2: Dice score coefficients for all ten different HSNet with the new data augmentation.

Instance	**CVC-T**	**ClinDB**	**Kvasir**	**ColDB**	**ETIS**	**Mean**
1	0.9035	0.9313	0.9231	0.8268	0.7965	0.8762
2	0.9021	0.9463	0.9150	0.8334	0.7825	0.8759
3	0.8857	0.9347	0.9124	0.8130	0.7978	0.8687
4	0.8983	0.9437	0.9265	0.8115	0.7583	0.8677
5	0.8837	0.9460	0.9160	0.8189	0.7651	0.8659
6	0.9025	0.9431	0.9146	0.8100	0.7787	0.8698
7	0.9045	0.9249	0.9102	0.8105	0.7667	0.8633
8	0.8829	0.9421	0.9222	0.8044	0.7630	0.8629
Avg.	0.8954	0.9390	0.9175	0.8161	0.7761	0.8688

Table 3: Dice score coefficients for all eight different HSNet with SAMAug data augmentation, where SAM is fine-tuned

Instance	**CVC-T**	**ClinDB**	**Kvasir**	**ColDB**	**ETIS**	**Mean**
1	0.8939	0.9363	0.9198	0.8245	0.7790	0.8707
2	0.9052	0.9449	0.9190	0.8183	0.7831	0.8741
3	0.8927	0.9435	0.9313	0.8091	0.7896	0.8732
4	0.9007	0.9443	0.9225	0.8230	0.7886	0.8758
5	0.8901	0.9308	0.9168	0.8103	0.7791	0.8654
6	0.8923	0.9356	0.9284	0.8100	0.7779	0.8688
7	0.8911	0.9354	0.9146	0.8101	0.7774	0.8657
8	0.8993	0.9472	0.9131	0.8319	0.7776	0.8738
9	0.8878	0.9402	0.9217	0.8178	0.7789	0.8693
10	0.8949	0.9284	0.9188	0.8185	0.7955	0.8712
Avg.	0.8948	0.9387	0.9206	0.8173	0.7827	0.8708

Table 4: Dice score coefficients for all eight different HSNet with SAMAug data augmentation, where SAM is the SAM2 version

Instance	**CVC-T**	**ClinDB**	**Kvasir**	**ColDB**	**ETIS**	**Mean**
1	0.8915	0.9337	0.9116	0.8018	0.7829	0.8643
2	0.8955	0.9300	0.9258	0.7918	0.7980	0.8682
3	0.8917	0.9260	0.9143	0.7985	0.7834	0.8628
4	0.8966	0.9396	0.9221	0.8042	0.7970	0.8719
5	0.8886	0.9437	0.9190	0.8223	0.7883	0.8724
6	0.8939	0.9309	0.9137	0.8044	0.7889	0.8664
7	0.8844	0.9254	0.9165	0.8138	0.7872	0.8655
8	0.8915	0.9295	0.9145	0.8164	0.7918	0.8687
9	0.8972	0.9445	0.9254	0.8111	0.7886	0.8734
Avg.	0.8923	0.9337	0.9181	0.8071	0.7896	0.8682

Table 5: Dice score coefficients for all eight different HSNet with SegPriorRGB data augmentation, where SAM is fine-tuned

Instance	**CVC-T**	**ClinDB**	**Kvasir**	**ColDB**	**ETIS**	**Mean**
1	0.8936	0.9320	0.9147	0.8167	0.7895	0.8693
2	0.8973	0.9261	0.9156	0.8143	0.7920	0.8691
3	0.8647	0.9216	0.9201	0.8138	0.7976	0.8636
4	0.8884	0.9328	0.9163	0.8171	0.8020	0.8713
5	0.8918	0.9231	0.9237	0.8250	0.7981	0.8723
6	0.8854	0.9368	0.9109	0.8227	0.7977	0.8707
7	0.8972	0.9470	0.9171	0.8288	0.7999	0.8780
8	0.8975	0.9296	0.9237	0.8172	0.7861	0.8708
9	0.8840	0.9281	0.9170	0.8125	0.7966	0.8676
10	0.8855	0.9375	0.9247	0.8186	0.8053	0.8743
Avg.	0.8885	0.9315	0.9184	0.8187	0.7965	0.8707

Table 6: Dice score coefficients for all eight different HSNet with the logits of the SegPriorRGB data augmentation, where SAM is fine-tuned

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8961	0.9308	0.9193	0.8167	0.7876	0.8701
2	0.8888	0.9239	0.9221	0.8131	0.7885	0.8673
3	0.8801	0.9331	0.9208	0.8058	0.7969	0.8674
4	0.8823	0.9461	0.9224	0.8201	0.7906	0.8723
5	0.8930	0.9232	0.9179	0.8303	0.7905	0.8710
6	0.8846	0.9361	0.9188	0.8233	0.7908	0.8707
7	0.8873	0.9400	0.9203	0.8188	0.7944	0.8722
8	0.8974	0.9317	0.9216	0.8136	0.8007	0.8730
9	0.8966	0.9459	0.9203	0.8196	0.8049	0.8775
Avg.	0.8896	0.9346	0.9204	0.8179	0.7939	0.8713

Table 7: Dice score coefficients for all eight different HSNet with the logits of the SegPriorRGB data augmentation, where SAM is the SAM2 version

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.9075	0.9326	0.9103	0.7960	0.7884	0.8670
2	0.8956	0.9249	0.9079	0.8088	0.7797	0.8634
3	0.8939	0.9337	0.9145	0.7854	0.7913	0.8638
4	0.9030	0.9400	0.9133	0.7866	0.7894	0.8664
5	0.8914	0.9383	0.9159	0.8048	0.7758	0.8653
6	0.9024	0.9398	0.9114	0.7941	0.7759	0.8647
7	0.9020	0.9417	0.9190	0.7872	0.7799	0.8659
8	0.8928	0.9407	0.9096	0.8192	0.7756	0.8676
9	0.9049	0.9337	0.9065	0.7882	0.7976	0.8662
10	0.8969	0.9345	0.9071	0.7881	0.7750	0.8603
Avg.	0.8990	0.9360	0.9115	0.7958	0.7829	0.8651

Table 8: Dice score coefficients for all eight different HSNet with SegPriorHSV data augmentation, where SAM is fine-tuned

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8940	0.9297	0.9111	0.8072	0.7700	0.8624
2	0.8949	0.9286	0.9063	0.8014	0.7436	0.8550
3	0.9132	0.9146	0.9044	0.8030	0.7659	0.8602
4	0.8980	0.9398	0.9143	0.8052	0.7534	0.8621
5	0.9002	0.9469	0.9077	0.8184	0.7808	0.8708
6	0.8973	0.9307	0.9099	0.7953	0.7619	0.8590
7	0.8978	0.9410	0.9151	0.7981	0.7827	0.8669
8	0.8960	0.9269	0.9184	0.8114	0.7627	0.8631
9	0.8992	0.9378	0.9172	0.7977	0.7725	0.8649
10	0.8980	0.9261	0.9173	0.8014	0.7375	0.8561
Avg.	0.8989	0.9322	0.9122	0.8039	0.7631	0.8620

Table 9: Dice score coefficients for all eight different HSNet with SegPriorHSV data augmentation, where SAM is the SAM2 version

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8876	0.9433	0.9155	0.8257	0.7530	0.8650
2	0.8998	0.9451	0.9092	0.8191	0.7499	0.8646
3	0.8909	0.9455	0.9141	0.8019	0.7555	0.8616
4	0.8899	0.9463	0.9162	0.8116	0.7659	0.8660
5	0.9020	0.9407	0.9086	0.8205	0.7573	0.8658
6	0.8887	0.9463	0.9097	0.8313	0.7764	0.8705
7	0.8991	0.9361	0.9176	0.8260	0.7546	0.8667
8	0.9053	0.9345	0.9181	0.8058	0.7541	0.8635
9	0.8990	0.9473	0.9116	0.8301	0.7609	0.8698
10	0.8839	0.9442	0.9100	0.8189	0.7679	0.8650
Avg.	0.8946	0.9429	0.9131	0.8191	0.7595	0.8659

Table 10: Dice score coefficients for all eight different HSNet with SegPriorPCA data augmentation, where SAM is fine-tuned

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8912	0.9311	0.8990	0.8158	0.7843	0.8643
2	0.8853	0.9323	0.9123	0.8102	0.7750	0.8630
3	0.8856	0.9451	0.9039	0.8061	0.7711	0.8623
4	0.8811	0.9462	0.9099	0.8031	0.7722	0.8625
5	0.8967	0.9421	0.9143	0.8019	0.7808	0.8672
6	0.8965	0.9336	0.9139	0.8199	0.7692	0.8666
7	0.8830	0.9427	0.9141	0.8089	0.7528	0.8603
8	0.8851	0.9415	0.9114	0.8067	0.7461	0.8582
9	0.8901	0.9468	0.9041	0.8090	0.7753	0.8650
10	0.8987	0.9433	0.9074	0.8137	0.7822	0.8691
Avg.	0.8893	0.9405	0.9090	0.8095	0.7709	0.8639

Table 11: Dice score coefficients for all eight different HSNet with SegPriorPCA data augmentation, where SAM is the SAM2 version

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.9028	0.9227	0.9172	0.8219	0.8227	0.8775
2	0.8997	0.9201	0.9157	0.8124	0.8312	0.8758
3	0.9024	0.9226	0.9130	0.8235	0.8228	0.8769
4	0.9039	0.9144	0.9134	0.8214	0.8262	0.8759
5	0.9037	0.9279	0.9121	0.8186	0.8240	0.8773
6	0.9026	0.9347	0.9137	0.8239	0.8230	0.8796
7	0.9052	0.9089	0.9138	0.8136	0.8305	0.8744
8	0.9036	0.9213	0.9152	0.8139	0.8294	0.8767
9	0.8997	0.9201	0.9157	0.8124	0.8312	0.8758
10	0.9047	0.9159	0.9150	0.8210	0.8238	0.8761
Avg.	0.9028	0.9209	0.9145	0.8183	0.8265	0.8766

Table 12: Dice score coefficients for all ten different Ensembles created with the HSNet model trained with data augmentation

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.9007	0.9448	0.9264	0.8209	0.7917	0.8769
2	0.8971	0.9449	0.9225	0.8335	0.7981	0.8792
3	0.8902	0.9421	0.9258	0.8224	0.7900	0.8741
4	0.8964	0.9334	0.9287	0.8243	0.7967	0.8759
5	0.8956	0.9437	0.9279	0.8267	0.7943	0.8776
6	0.8934	0.9327	0.9304	0.8243	0.7909	0.8743
7	0.8950	0.9466	0.9277	0.8310	0.7825	0.8766
8	0.8993	0.9425	0.9213	0.8279	0.7986	0.8779
9	0.8924	0.9441	0.9291	0.8193	0.7913	0.8752
10	0.8945	0.9366	0.9271	0.8220	0.7824	0.8725
Avg.	0.8955	0.9411	0.9267	0.8252	0.7916	0.8760

Table 13: Dice score coefficients for all ten different Ensembles created with the HSNet model trained with SAMAug augmentation

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.9020	0.9500	0.9200	0.8372	0.8193	0.8857
2	0.9017	0.9376	0.9248	0.8273	0.8126	0.8808
3	0.9045	0.9323	0.9280	0.8302	0.8177	0.8825
4	0.8920	0.9412	0.9255	0.8319	0.8165	0.8814
5	0.9005	0.9509	0.9231	0.8424	0.8259	0.8886
6	0.8971	0.9265	0.9271	0.8376	0.8161	0.8809
7	0.9005	0.9334	0.9203	0.8350	0.8265	0.8831
8	0.9018	0.9422	0.9253	0.8439	0.8164	0.8859
9	0.9009	0.9467	0.9236	0.8399	0.8204	0.8863
10	0.9016	0.9315	0.9249	0.8360	0.8041	0.8796
Avg.	0.9003	0.9392	0.9243	0.8361	0.8175	0.8835

Table 14: Dice score coefficients for all ten different AuxMix Triple Ensembles

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8975	0.9488	0.9143	0.8399	0.8047	0.8810
2	0.9050	0.9397	0.9189	0.8437	0.8063	0.8827
3	0.9058	0.9464	0.9178	0.8341	0.8261	0.8860
4	0.9007	0.9470	0.9165	0.8358	0.8187	0.8837
5	0.9004	0.9456	0.9247	0.8299	0.8081	0.8817
6	0.9022	0.9471	0.9214	0.8379	0.8155	0.8848
7	0.8977	0.9325	0.9203	0.8329	0.8194	0.8806
8	0.9058	0.9366	0.9163	0.8330	0.8190	0.8821
9	0.9036	0.9487	0.9150	0.8301	0.8229	0.8841
10	0.9053	0.9411	0.9230	0.8319	0.8164	0.8835
Avg.	0.9024	0.9434	0.9188	0.8349	0.8157	0.8830

Table 15: Dice score coefficients for all ten different Segmentation-Prior Ensembles

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.8975	0.9435	0.9209	0.8370	0.7870	0.8772
2	0.9004	0.9487	0.9217	0.8255	0.8120	0.8817
3	0.8991	0.9480	0.9176	0.8329	0.7977	0.8791
4	0.8919	0.9327	0.9271	0.8304	0.8042	0.8773
5	0.8905	0.9476	0.9158	0.8283	0.7929	0.8750
6	0.8922	0.9415	0.9223	0.8276	0.7826	0.8732
7	0.8818	0.9411	0.9317	0.8231	0.7881	0.8732
8	0.8945	0.9288	0.9269	0.8196	0.7994	0.8738
9	0.8987	0.9491	0.9272	0.8263	0.7908	0.8784
10	0.9007	0.9329	0.9242	0.8317	0.8054	0.8790
Avg.	0.8947	0.9414	0.9235	0.8282	0.7960	0.8768

Table 16: Dice score coefficients for all ten different Semantic Ensembles

Instance	CVC-T	ClinDB	Kvasir	ColDB	ETIS	Mean
1	0.9066	0.9457	0.9211	0.8429	0.8292	0.8891
2	0.9042	0.9317	0.9282	0.8449	0.8278	0.8874
3	0.9042	0.9374	0.9275	0.8401	0.8286	0.8876
4	0.9052	0.9505	0.9249	0.8432	0.8288	0.8905
5	0.9034	0.9370	0.9269	0.8412	0.8335	0.8884
6	0.9035	0.9342	0.9252	0.8428	0.8305	0.8872
7	0.9030	0.9334	0.9276	0.8426	0.8276	0.8868
8	0.9053	0.9342	0.9242	0.8436	0.8290	0.8873
9	0.8981	0.9322	0.9259	0.8405	0.8287	0.8851
10	0.9012	0.9462	0.9267	0.8441	0.8259	0.8888
Avg.	0.9035	0.9382	0.9258	0.8426	0.8290	0.8878

Table 17: Dice score coefficients for all ten different AuxMix Ninefold Ensembles

Bibliography

[1] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: `2304.02643 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2304.02643`.

[2] G. Csurka, R. Volpi, and B. Chidlovskii, *Semantic image segmentation: Two decades of research*, 2023. arXiv: `2302.06378 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2302.06378`.

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: `1706.03762 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1706.03762`.

[4] R. M. Schmidt, *Recurrent neural networks (rnns): A gentle introduction and overview*, 2019. arXiv: `1912.05911 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1912.05911`.

[5] R. C. Staudemeyer and E. R. Morris, *Understanding lstm – a tutorial into long short-term memory recurrent neural networks*, 2019. arXiv: `1909.09586 [cs.NE]`. [Online]. Available: `https://arxiv.org/abs/1909.09586`.

[6] H. Naveed, A. U. Khan, S. Qiu, *et al.*, *A comprehensive overview of large language models*, 2024. arXiv: `2307.06435 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2307.06435`.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: `2010.11929 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2010.11929`.

[8] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: `1511.08458 [cs.NE]`. [Online]. Available: `https://arxiv.org/abs/1511.08458`.

[9] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, "Segment anything in medical images," *Nature Communications*, vol. 15, no. 1, Jan. 2024, issn: 2041-1723. doi: `10.1038/s41467-024-44824-z`. [Online]. Available: `http://dx.doi.org/10.1038/s41467-024-44824-z`.

[10] R. Deng, C. Cui, Q. Liu, *et al.*, *Segment anything model (sam) for digital pathology: Assess zero-shot segmentation on whole slide imaging*, 2023. arXiv: `2304.04155 [eess.IV]`. [Online]. Available: `https://arxiv.org/abs/2304.04155`.

[11] L. Huang, W. Yu, W. Ma, *et al.*, *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*, 2023. arXiv: `2311.05232 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2311.05232`.

[12] P. Zhao, H. Zhang, Q. Yu, *et al.*, *Retrieval-augmented generation for ai-generated content: A survey*, 2024. arXiv: `2402.19473 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2402.19473`.

[13] Y. Zhang, T. Zhou, S. Wang, P. Liang, and D. Z. Chen, *Input augmentation with sam: Boosting medical image segmentation with segmentation foundation model*, 2023. arXiv: `2304.11332 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2304.11332`.

[14] Z. Wang, P. Wang, K. Liu, *et al.*, *A comprehensive survey on data augmentation*, 2024. arXiv: `2405.09591 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2405.09591`.

[15] L. Nanni, A. Lumini, and C. Fantozzi, "Exploring the potential of ensembles of deep learning networks for image segmentation," *Information*, vol. 14, no. 12, 2023, issn: 2078-2489. doi: `10.3390/info14120657`. [Online]. Available: `https://www.mdpi.com/2078-2489/14/12/657`.

[16] R. Zhang, G. Li, Z. Li, S. Cui, D. Qian, and Y. Yu, "Adaptive context selection for polyp segmentation," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, A. L. Martel, P. Abolmaesumi, D. Stoyanov, *et al.*, Eds., Cham: Springer International Publishing, 2020, pp. 253–262, isbn: 978-3-030-59725-2.

[17] D.-P. Fan, G.-P. Ji, T. Zhou, *et al.*, "Pranet: Parallel reverse attention network for polyp segmentation," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, A. L. Martel, P. Abolmaesumi, D. Stoyanov, *et al.*, Eds., Cham: Springer International Publishing, 2020, pp. 263–273, isbn: 978-3-030-59725-2.

[18] D. Jha, P. H. Smedsrud, M. A. Riegler, *et al.*, "Resunet++: An advanced architecture for medical image segmentation," in *2019 IEEE International Symposium on Multimedia (ISM)*, 2019, pp. 225–2255. doi: `10.1109/ISM46123.2019.00049`.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, isbn: 978-3-319-24574-4.

[20] X. Xiao, S. Lian, Z. Luo, and S. Li, "Weighted res-unet for high-quality retina vessel segmentation," in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, 2018, pp. 327–331. doi: `10.1109/ITME.2018.00080`.

[21] V. Iglovikov and A. Shvets, *Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation*, 2018. arXiv: `1801.05746 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1801.05746`.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, 2016, pp. 770–778. doi: `10.1109/CVPR.2016.90`.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. doi: `10.1109/CVPR.2016.90`.

[24] Y. Gao, M. Zhou, and D. N. Metaxas, "Utnet: A hybrid transformer architecture for medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, M. de Bruijne, P. C. Cattin, S. Cotin, *et al.*, Eds., Cham: Springer International Publishing, 2021, pp. 61–71, isbn: 978-3-030-87199-4.

[25] J. Chen, Y. Lu, Q. Yu, *et al.*, *Transunet: Transformers make strong encoders for medical image segmentation*, 2021. arXiv: `2102.04306 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2102.04306`.

[26] W. Zhang, C. Fu, Y. Zheng, F. Zhang, Y. Zhao, and C.-W. Sham, "Hsnet: A hybrid semantic network for polyp segmentation," *Computers in Biology and Medicine*, vol. 150, p. 106 173, 2022, issn: 0010-4825. doi: `https://doi.org/10.1016/j.compbiomed.2022.106173`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0010482522008812`.

[27] W. Wang, E. Xie, X. Li, *et al.*, *Pyramid vision transformer: A versatile backbone for dense prediction without convolutions*, 2021. arXiv: `2102.12122 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2102.12122`.

[28] Y. M. Ro, W.-H. Cheng, J. Kim, *et al.*, Eds., *MultiMedia Modeling, 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part II* (Lecture Notes in Computer Science), 1st ed. Springer Cham, 2020, pp. XXX, 820, isbn: 978-3-030-37733-5. doi: `10.1007/978-3-030-37734-2`. [Online]. Available: `https://doi.org/10.1007/978-3-030-37734-2`.

[29] D. Jha, P. H. Smedsrud, M. A. Riegler, *et al.*, "Kvasir-seg: A segmented polyp dataset," in *MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part II 26*, 2020, pp. 451–462.

[30] J. Bernal, J. Sánchez, and F. Vilariño, "Towards automatic polyp detection with a polyp appearance model," *Pattern Recognition*, vol. 45, no. 9, pp. 3166–3182, 2012, Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011), issn: 0031-3203. doi: `https://doi.org/10.1016/j.patcog.2012.03.002`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0031320312001185`.

[31] J. Bernal, F. J. Sánchez, G. Fernández-Esparrach, D. Gil, C. Rodríguez, and F. Vilariño, "Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians," *Computerized Medical Imaging and Graphics*, vol. 43, pp. 99–111, 2015, issn: 0895-6111. doi: `https://doi.org/10.1016/j.compmedimag.2015.02.007`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0895611115000567`.

[32] J. Bernal, N. Tajbakhsh, F. Sanchez, *et al.*, "Comparative validation of polyp detection methods in video colonoscopy: Results from the miccai 2015 endoscopic vision challenge," *IEEE Transactions on Medical Imaging*, vol. PP, pp. 1–1, Feb. 2017. doi: `10.1109/TMI.2017.2664042`.

[33] J. Silva, A. Histace, O. Romain, X. Dray, and B. Granado, "Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer," *International Journal of Computer Assisted Radiology and Surgery*, vol. 9, no. 2, pp. 283–293, 2014, issn: 1861-6429. doi: `10.1007/s11548-013-0926-3`. [Online]. Available: `https://doi.org/10.1007/s11548-013-0926-3`.

[34] D. Vázquez, J. Bernal, F. J. Sánchez, *et al.*, *A benchmark for endoluminal scene segmentation of colonoscopy images*, 2016. arXiv: `1612.00799 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1612.00799`.

[35] M. Barat, G. Chassagnon, A. Dohan, *et al.*, "Artificial intelligence: A critical review of current applications in pancreatic imaging," *Japanese journal of radiology*, vol. 39, Feb. 2021. doi: `10.1007/s11604-021-01098-5`.