Alma Mater Studiorum · Università di Bologna

SCHOOL OF SCIENCE
Master's Degree in Mathematics
Curriculum Advanced Mathematics for Applications

Spatio-Temporal Ultrasound Segmentation: From Flow-Based Tracking to Promptable Transformers and State-Space Models

Master's Thesis in Machine Learning Applied to Healthcare

Supervisor:
Chiar.ma Prof.ssa
Federica Gerace

Presented by:
Nicolò Signani

Academic Year 2024-2025

Introduction

Ovarian cancer is one of the leading causes of cancer-related mortality worldwide. Early and accurate characterization of adnexal masses (benign or malignant) is critical and relies heavily on transvaginal ultrasound, a noninvasive, real-time imaging modality [1]. Manual detection is often subjective and dependent on the diagnosing physician's expertise; therefore, automated segmentation could be explored as a standardized approach to reduce diagnostic variability, minimize clinical workload, and ultimately improve patient outcomes. During examinations, ultrasound systems generate video clips that capture the dynamic nature of anatomical structures. SynDiag employs an image segmentation deep learning model to provide frame-by-frame accurate segmentation. However, this approach does not fully utilize the temporal dynamics inherent in video data, which could offer superior diagnostic contextual information compared to single-frame analysis. Furthermore, ultrasound imaging presents additional challenges for computer vision applications [41]:

- **Speckle Noise and Low Contrast**: Inherent ultrasound speckle (i.e. a common grainy, granular texture that obscures anatomical details) reduces boundary definition and complicate segmentation.

- **Probe Motion and Deformation**: Dynamic probe positioning and patient movement introduce substantial frame-to-frame variations, presenting challenges for segmentation models.

- **Lesion Characteristics**: The diverse properties of cystic, solid, and mixed lesions generate complex imaging patterns that challenge automated segmentation algorithms.

These factors limit the effectiveness of frame-only segmentation networks and motivate the incorporation of temporal context. In this thesis, we explore three strategies to introduce temporal coherence into ovarian ultrasound segmentation:

1. **Post-Processing Tracking (Chapter 3)**: Using SEA-RAFT optical flow estimation to warp and fuse U-Net masks across frames, reducing flicker.

2. **Promptable Transformer-based Model (Chapter 4)**: Extending the promptable SAM2 model with a self-sorting memory bank (Med-SAM2) to leverage the most informative past frames.

3. **State-Space-based Model (Chapter 4)**: Embedding state-space modules and boundary-aware losses into a Video Object Segmentation network (ViViM) for efficient, accurate long-range modeling.

We begin by giving some preliminary notions in Chapter 1. In Chapter 2, we characterize our privately collected dataset and preprocessing pipelines. In Chapters 3 and 4, as explained above, we will introduce the innovative architectures that will be used during the thesis. Chapter 5 delivers extensive evaluation on our ultrasound test set, quantifying segmentation metrics, boundary accuracy, and inference speed for each approach versus the U-Net baseline. Finally, we conclude with a discussion of clinical implications, limitations, and future research directions.

Through these contributions, we demonstrate that integrating temporal information, whether via optical-flow warping, self-sorting memory banks, or state-space modeling, substantially enhances the reliability and consistency of ultrasound segmentation for ovarian cancer diagnosis.

Contents

Chapter 1

Preliminary Notions

In this chapter some preliminary notions will be given to better understand this thesis contents. Particularly, we will explore the task of multiclass segmentation, along with the benchmark method for image segmentation, the U-Net, and its principal components. Lastly, we introduce Transformers, that are the building blocks of most of the proposed video object segmentation models.

1.1 Image Segmentation

Image segmentation is a fundamental and critical task in computer vision that involves partitioning images into multiple distinct regions based on shared characteristics such as color, intensity, texture or semantic meaning. This process serves as a foundational component for numerous applications including scene understanding, medical image analysis, robotic perception, video surveillance, augmented reality, and image compression [2].
Building upon the foundation of image segmentation, **Video Object Segmentation (VOS)** extends these principles to the temporal domain, representing a more challenging and complex task. VOS aims at segmenting objects of interest throughout the given video sequence, requiring models to maintain consistent object boundaries and identities across multiple frames while handling dynamic challenges such as object deformation, occlusions, illumination changes, and camera motion [3]. Generally, in VOS models, segmentation is still performed frame-by-frame, and temporal complexity is introduced by means of a memory bank that retains information about past frames.
In this section we will explore one of the main components of modern image segmentation models, CNNs, and we will present the baseline method used as benchmark during the thesis, highlighting the motivations that lead to the introduction of VOS models, instead of simply using image segmentation models.

1.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs)[4] are the foundation of computer vision related tasks (such as image classification and segmentation) and they are composed of three types of layers:

1. **Convolutional Layer**: whose primary function is to apply a set of filters (kernels) to the input image in order to extract spatial features such as edges, textures, and patterns.
2. **Pooling Layer**: responsible for performing downsampling along the spatial dimensions of the input, reducing the number of parameters.
3. **Fully-Connected Layer**: the final layer that provides probabilities and classification results.

The main innovation reside in the convolutional layers, which are responsible for extracting spatial features. In order to perform such extraction, each filter is slid across the input, performing element-wise multiplication between the kernel and a small portion of the input, and then summing the result to produce a single value. This process is repeated across the entire image to produce a new feature map, which highlights the presence of specific properties detected by the filters. Mathematically, given an input image I and a filter $K \in \mathbb{R}^{h_K \times w_K}$ the convolution operation at pixel (i, j) is defined as:

$$(I * K)(i, j) = \sum_{m=1}^{h_K} \sum_{l=1}^{w_K} K(m, l) I(i - m, j - l)$$

where h_K and w_K are the height and width of the kernel, respectively. An example of a 3×3 convolution is shown in Figure 1.1.

In CNNs, multiple convolutional layers are applied sequentially in order to hierarchically extract increasingly complex patterns. After each convolution, an activation function (usually ReLU) is applied to introduce non-linearity into the model, enabling it to learn more complex representations. Additionally, convolutional layers are alternated with pooling (and possibly normalization) layers in order to reduce dimensionality and stabilize training. Finally, the features extracted by the convolutional layers are flattened and passed through one or more fully connected layers to produce the final output. An example of a simple CNN is shown in Figure 1.2.

1.1.2 U-Net

Currently, a **U-Net** architecture is adopted as a baseline for multi-class ovary segmentation. In this section it will be presented the U-net structure, highlighting its strengths and weakness, and motivate the choice of introducing **Video Object Segmentation** end-to-end models.

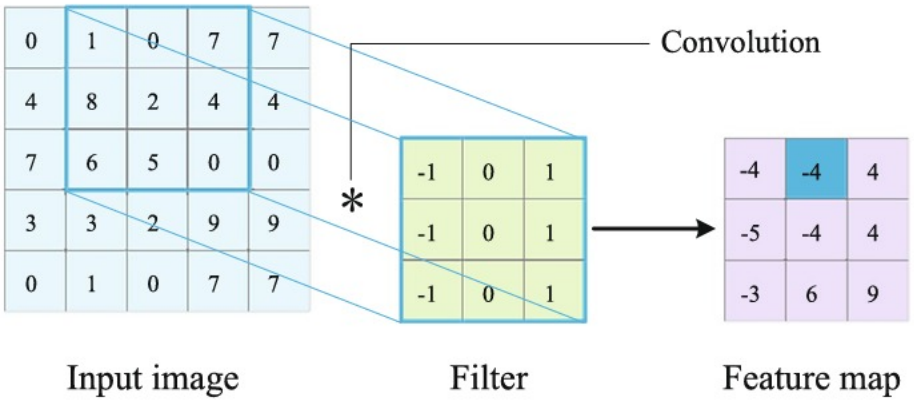


Figure 1.1: Example of a 3×3 convolution [4].

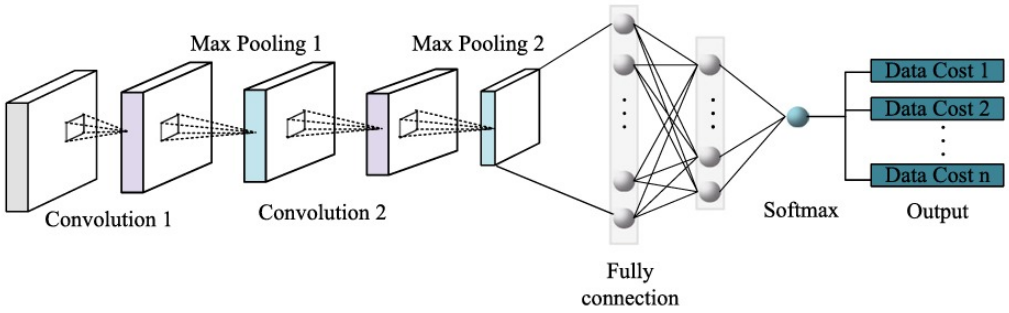


Figure 1.2: Structure of a CNN in an n-classification task [4]. The input data goes through two convolutional layers, two pooling layers and a fully connected layer. The Softmax activation function provides the probability of the data belonging to each category.

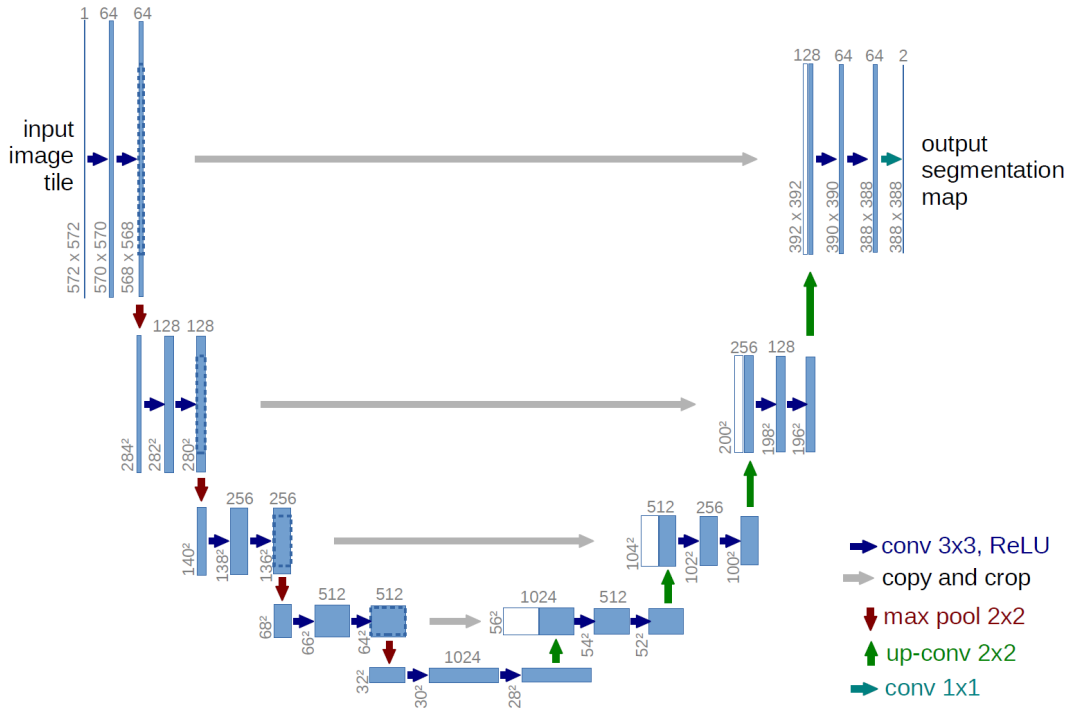


Figure 1.3: U-Net architecture, showing the symmetric encoder-decoder structure with skip connections.

(a) Input frame (b) Ground-truth mask (c) U-Net prediction

Figure 1.4: Example segmentation on a representative frame of our dataset. As we can see U-Net performs reasonably well on both the solid class (in red) and the non-solid class (in yellow).

U-Net Architecture

U-Net is a specialized CNN architecture that was originally developed for biomedical image segmentation tasks[16]. As shown in Figure 1.3, it consists of a contracting encoder path and an expanding decoder path, with skip connections that transfer high-resolution features from encoder to decoder.

- **Encoder:** Repeated application of two 3×3 convolutions (each followed by a ReLU activation function [5]), then a 2×2 max-pooling with stride 2 for downsampling. At each downsampling step the number of feature channels doubles.
- **Decoder:** Each step begins with a 2×2 up-sampling, reducing by half the number of feature channels, followed by concatenation with the corresponding feature map from the encoder via a skip connection. Two further 3×3 convolutions (with ReLU) refine the combined features.
- **Output layer:** A final 1×1 convolution maps to three channels (one per class), and a softmax activation produces per-pixel class probabilities.

Advantages and Limitations

Advantages

- **Localization with context:** Skip connections deliver high-resolution spatial information from the encoder directly to the decoder, giving precise boundary delineation, as we can see in Figure 1.4.
- **Data efficiency:** The fully convolutional design and extensive feature reutilization make the U-Net perform well on small medical datasets without the risk of overfitting.

Limitations

- **Frame independence:** U-Net processes each frame separately, without leveraging temporal continuity or dynamics.
- **Temporal inconsistency:** With no temporal context we could lose some useful information between adjacent frames, leading to a reduction of performance.
- **Computational cost:** Processing each frame separately can be slow for long video sequences.

To address these issues, we explore three approaches in the following chapters:

1. **U-Net + Tracking:** Tracking is explored as a post-processing step that enforces temporal consistency by linking segmented regions across frames.

2. **Vision-Transformer-Based Model:** End-to-end Promptable Video Object Segmentation model based on transformers that encodes temporal information through the utilization of a **Memory Bank** that stores past frames.

3. **State-Space-Based Model:** End-to-end Video Object Segmentation model that propagates spatial and temporal information through a State-Space model architecture (**Temporal Mamba Blocks**) to perform accurate, temporal aware segmentation.

1.2 Transformers

Transformers are a class of neural architecture based on a **self-attention** mechanism that have become the foundational blocks of modern deep learning models. Originally introduced for natural language processing [6], the transformer's key idea is to compute interactions between all positions of the input via learned attention weights, enabling efficient modeling of long-range dependencies.
To understand why transformers were revolutionary, consider how humans read and understand text. When you read a sentence like "The cat that was sitting on the couch stood up" your brain does not process each word independently. Instead, you automatically connect "cat" with "stood up" even though they're separated by several words. Traditional neural networks like RNNs [7] struggled with these long-range connections because they processed sequences step-by-step, making it difficult to maintain information across long distances. The transformer's key innovation is self-attention: a mechanism that allows every position in the input to directly interact with every other position, regardless of distance. The mathematical foundation of this idea is scaled dot-product attention:

Scaled dot-product attention Given queries $Q \in \mathbb{R}^{N \times d_k}$, keys $K \in \mathbb{R}^{N \times d_k}$ and values $V \in \mathbb{R}^{N \times d_v}$ (where N is the number of tokens), the basic attention operation, as shown in Figure 1.5, is

$$\text{Attention}(Q, K, V) \;=\; \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)\, V.$$

Intuitively, the attention mechanism works as follows:

1. **Query-key matching:** The model computes similarity scores (QK^T) between the current word being processed (query) and all words in the sequence (keys);
2. **Normalization:** The softmax function converts these raw scores into attention weights that sum to 1;
3. **Value aggregation:** These weights determine how much each word's representation (values) contributes to the final encoding.

The scaling factor $\sqrt{d_k}$ prevents the dot products from becoming too large.

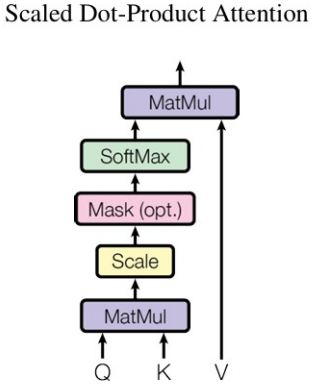


Figure 1.5: Scaled Dot-Product Attention.

Multi-head attention In order to allow the model to focus on different types of relationships simultaneously, the transformer uses H parallel attention heads, as shown in Figure 1.6:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O,$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. Each head has its own linear projections $W_i^{Q,K,V}$.

The entire model architecture is presented in Figure 1.7

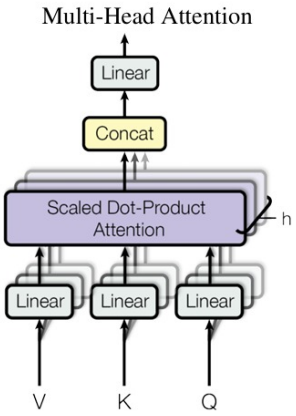


Figure 1.6: Multi-Head Attention consists of several attention layers running in parallel.

Key advantages The transformer architecture offers several advantages over previous sequence-to-sequence models:

- **Parallelization**: Unlike RNNs[7], all positions can be processed simultaneously during training, leading to significant computational speedups.
- **Long-range dependencies**: The self-attention mechanism allows direct connections between any two positions in the sequence, regardless of their distance.
- **Interpretability**: Attention weights provide insights into which parts of the input the model should focus on for each output.

These properties have made transformers the dominant architecture not only for natural language processing tasks but also for computer vision (Vision Transformer) [8]. Particularly, due to their ability to efficiently handle long-range dependencies, the vast majority of the models explored in this thesis will be Vision Transformer based models, which adapts the transformer architecture to process images by treating image patches as sequential tokens.

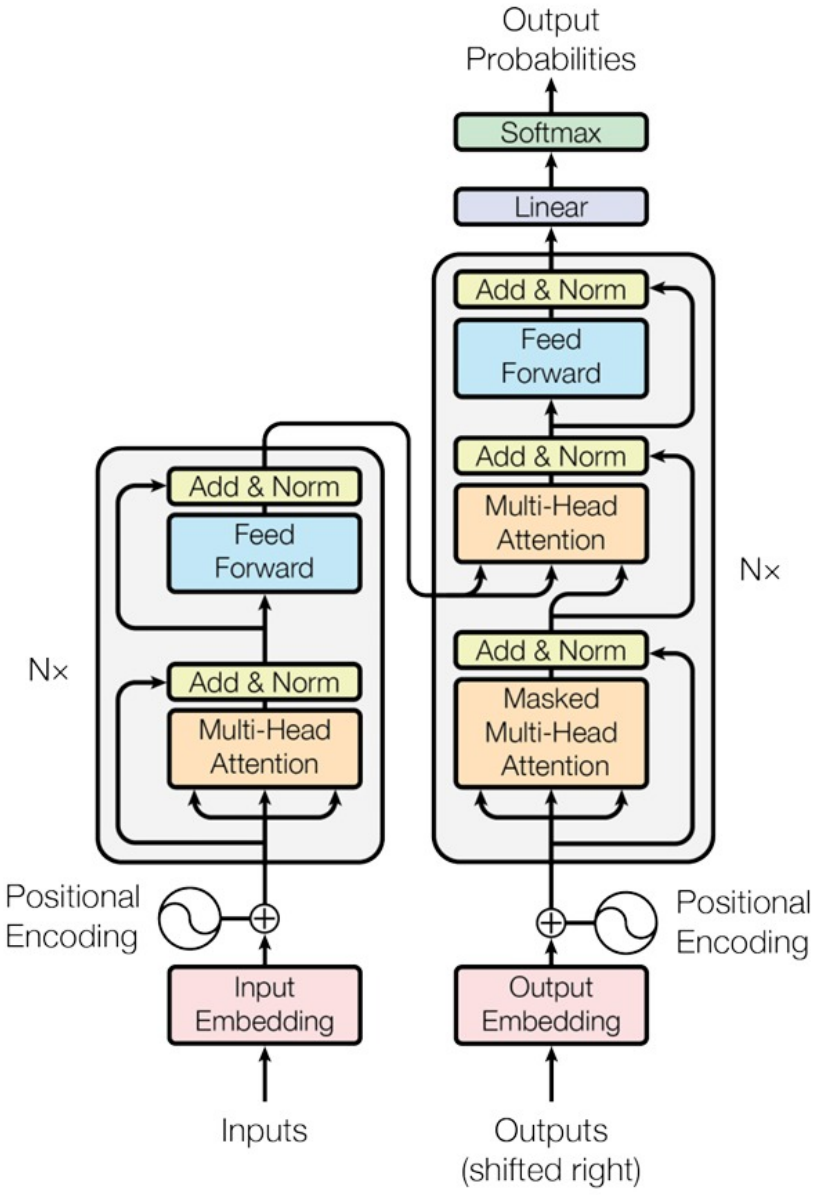


Figure 1.7: The entire transformer model architecture [6], comprising the innovative attention layer.

Chapter 2

Dataset Analysis

This chapter provides a detailed description of the **ovarian ultrasound dataset** used to train and evaluate the proposed methods. We begin by examining the most common histological subtypes represented in the data and by highlighting the main challenges in ovary segmentation. Next, we present a cross-validation strategy designed to assess model generalization and improve robustness. Finally, we describe the preprocessing steps applied to the data to mitigate overfitting.

2.1 Data Description

The dataset consists of transvaginal ultrasound (US) video sequences of the ovary, privately collected by gynecologists.

The US videos comprising the dataset are distributed between 33 different histological types. Figure 2.1 illustrates this distribution, highlighting the three most common categories:

1. **Serous Cystadenoma**: Benign, smooth-walled, purely cystic lesions lacking solid tissue.

2. **High-Grade Serous Adenocarcinoma**: Malignant epithelial tumors with mixed cystic-solid or predominantly solid components, often exhibiting necrosis and hemorrhage. Due to the high presence of solid component those are the most challenging of the three to correctly classify.

3. **Dermoid**: Benign tumors containing both fluid and solid elements (e.g., hair, keratin, calcifications).

The full dataset contains over 15,000 manually labeled frames, a representative frame for each of these subtypes and its ground truth mask are shown in Figure 2.3. Our deep learning models will be trained on the provided dataset to generate segmentation masks

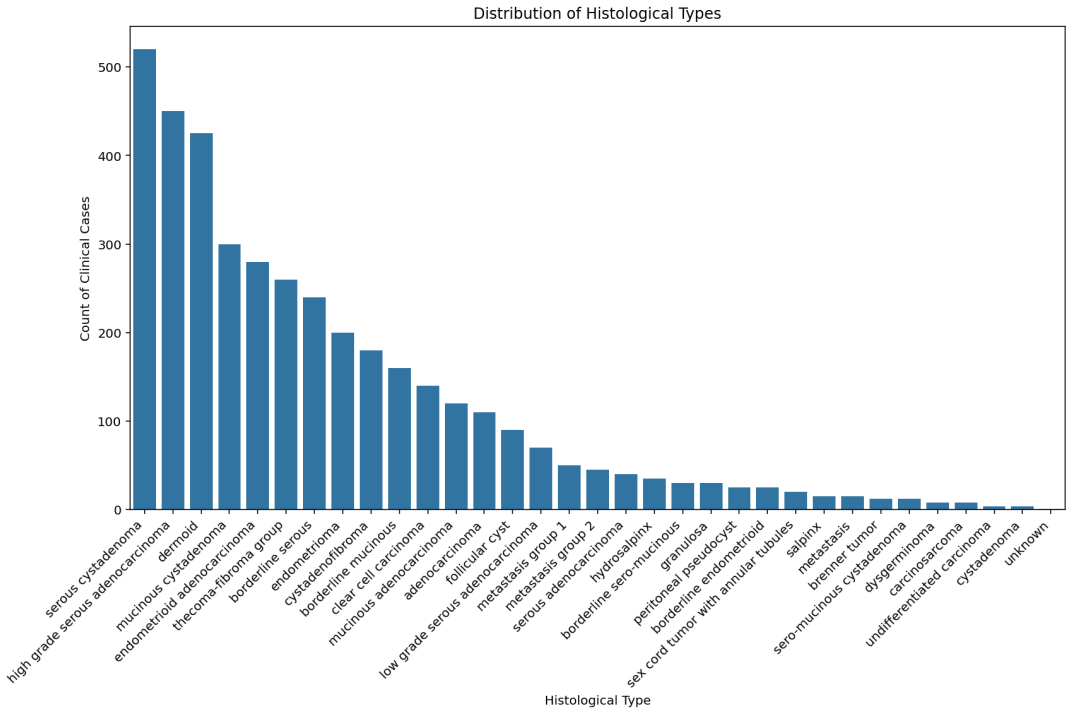


Figure 2.1: Counts of clinical cases divided by histological groups.

that closely match the ground-truth annotations, with the aim of developing an automated ultrasound segmentation tool. Particularly, each frame is segmented into three pixel-wise classes: **solid**, **non-solid** and **background**, for better understanding the classes see Figure 2.2. The solid class marks all echogenic mass tissue (i.e. the part of the lesion that reflects ultrasound waves, therefore looking bright), the non-solid class marks fluid or cystic regions (the ones appearing darker on ultrasound) and the background class includes all other structures outside the mass. This three-class definition is motivated by the fact that the morphological features of an adnexal mass are a good indicator of whether it can be considered as benign or malignant. In particular, the **IOTA Simple Rules** demonstrate that a large solid component increases the likelihood of malignancy, whereas unilocular cysts (single fluid chambers) or small solid components (<7 mm) are highly predictive of benignity, as described in [11]. Initially the full dataset is partitioned into training set (∼10,000 annotated frames) and and a test set (∼5,000 annotated frames). Analysis of the training masks distribution Figure 2.4 reveals:

- **Non-solid** presence in over 90% of frames.
- **Solid** presence in approximately 77% of frames.
- The most frequent mask combination is the one containing both solid and non-solid

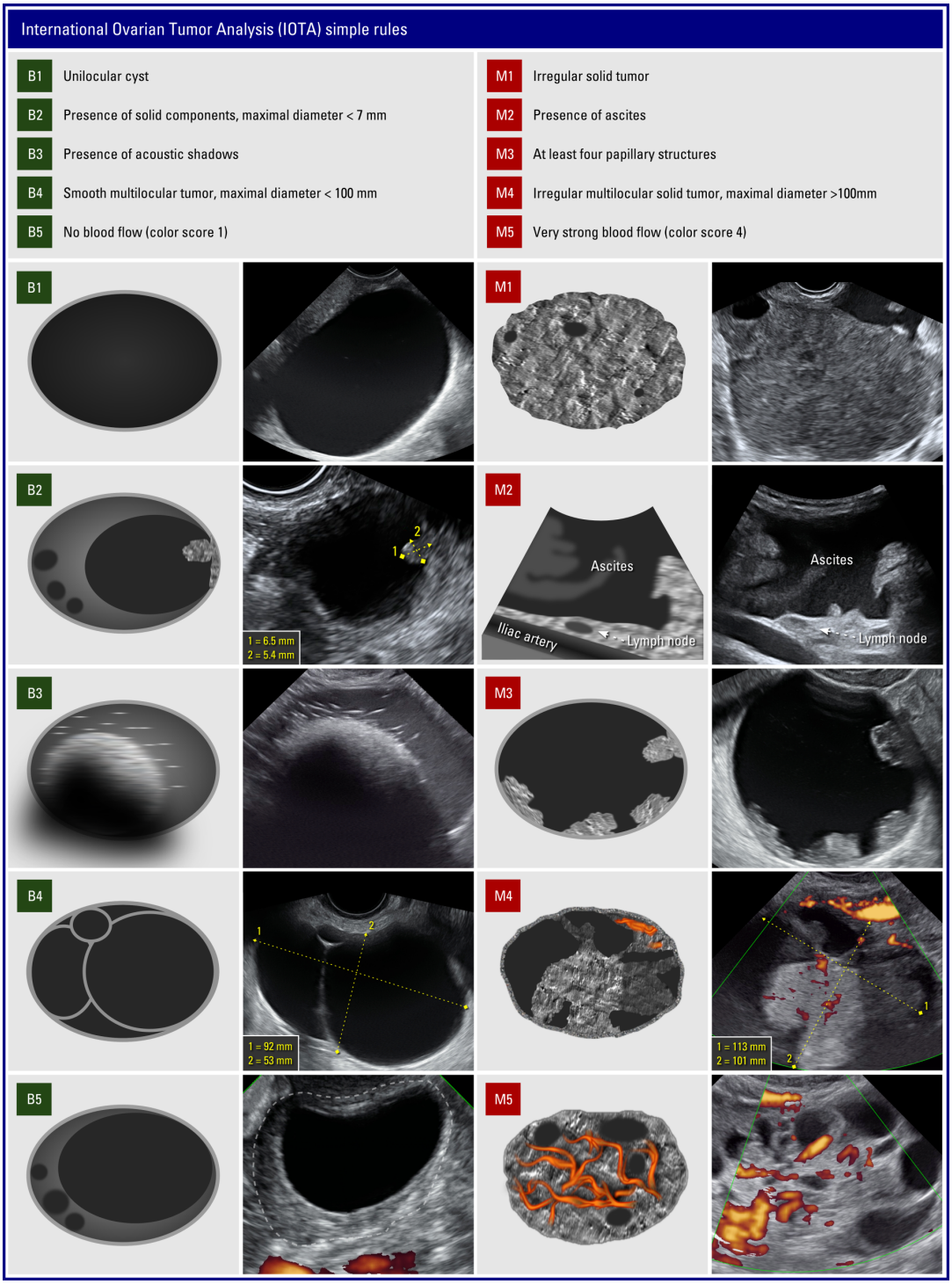


Figure 2.2: Illustration of the International Ovarian Tumor Analysis (IOTA) simple rules for adnexal mass characterization [12]. The left column (B1-B5) shows benign features, including unilocular cysts, small solid components (<7 mm), presence of acoustic shadows, smooth multilocular tumors (<100 mm), and absence of blood flow (color score 1). The right column (M1-M5) shows malignant features, including irregular solid tumors, presence of ascites, at least four papillary structures, irregular multilocular solid tumors (>100 mm), and very strong blood flow (color score 4).

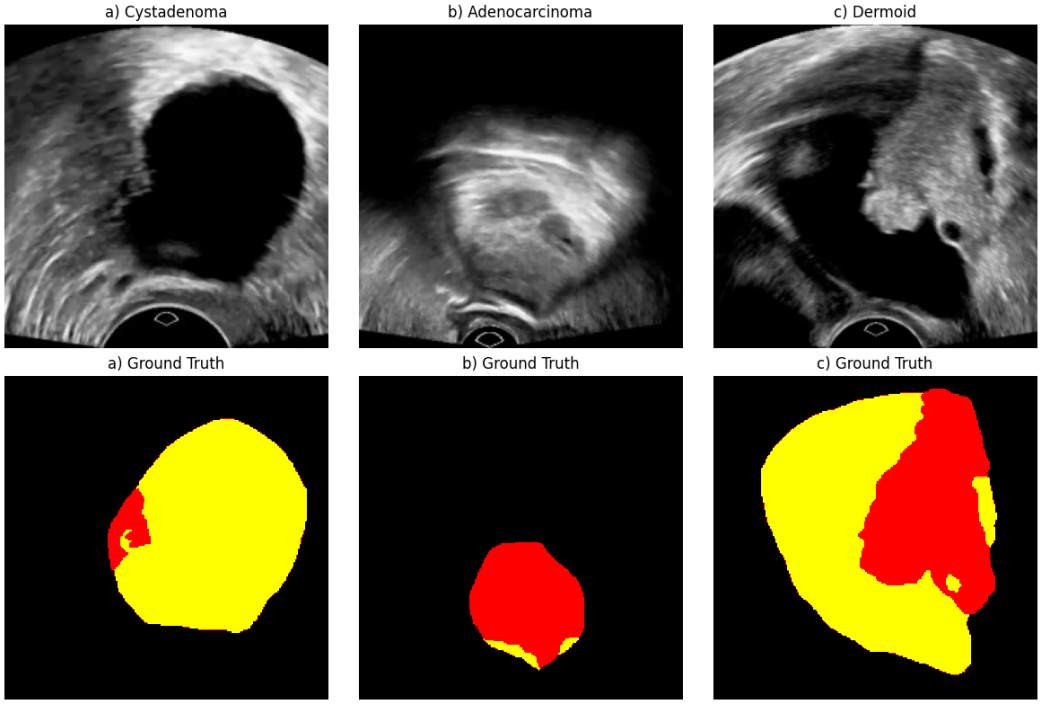


Figure 2.3: Representative ultrasound frames (top row) and corresponding ground-truth segmentation masks (bottom row) for the three most frequent histological subtypes, solid regions are coloured as red, non-solid regions as yellow. (a) **Serous cystadenoma**; (b) **High-grade serous adenocarcinoma**; (c) **Dermoid cyst**.

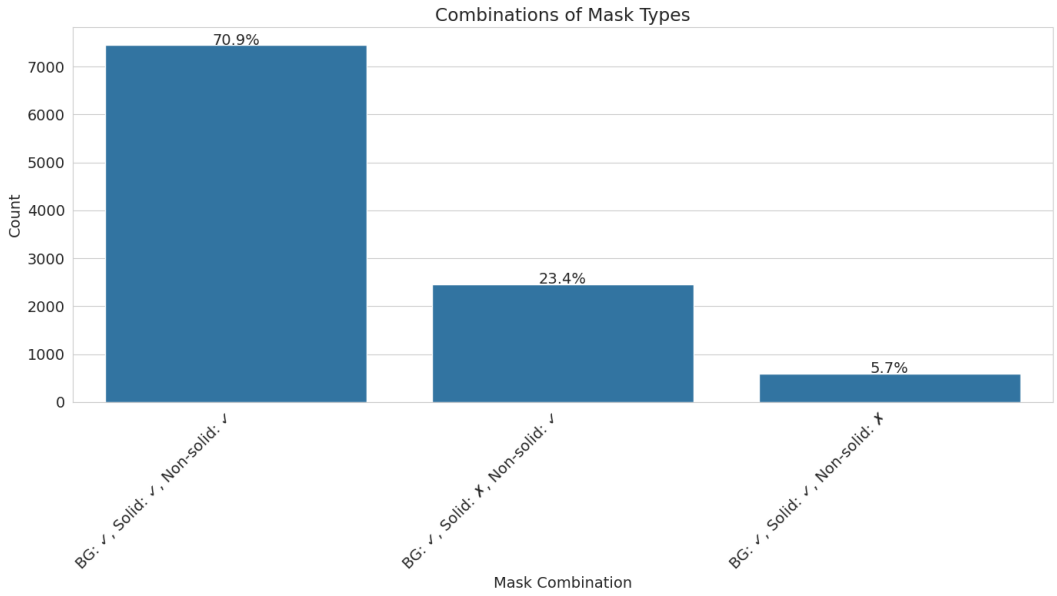


Figure 2.4: Distribution of mask-type combinations in the training set.

components, followed by only non-solid masks and, lastly frames with only solid masks.

These statistics reflect the predominance of benign cystic features (**serous cystadenomas** and **dermoids**) as explained before. Moreover, those results underline a moderate class imbalance between solid and non-solid categories, as well as an imbalance among histological cases. In the following sections, we will describe our dataset preparation pipeline, starting with the **preprocessing steps** we will adopt, followed by **cross-validation** during training, in order to address class imbalance, mitigate overfitting, and enhance model robustness.

2.2 Data Augmentation

Data augmentation is an essential technique in deep learning, especially in a medical imaging context where acquiring large and diverse datasets is challenging. By modifying the training dataset through various transformations, data augmentation enhances the model generalization, helps preventing overfitting and improves robustness of the model.

2.2.1 Classical Data Augmentation Techniques

Classical augmentations include geometric transformations (which move or crop the image) and photometric transformations (that change pixel intensities). These oper-

ations are a powerful tool for removing biases and simulating variability. Below are described the standard data augmentation techniques applied to our training dataset, following the official implementation of [13].

Random Cropping

Random cropping is a data augmentation technique that involves selecting a random portion (crop) of the original image to use as a training sample for the model. In practical terms, if the original image is $I(x, y)$ of width W and height H, we choose a random offset Δx, Δy, with $0 \leq \Delta x \leq W - w$, $0 \leq \Delta y \leq H - h$, and output the cropped image:

$$I'(u, v) = I(u + \Delta x, v + \Delta y), \quad u = 0, \ldots, w - 1, v = 0, \ldots, h - 1,$$

where (w, h) is the crop size. Afterward the cropped image is resized back to the original size. The idea is that by using different crops of the same image, the model is exposed to various contexts and orientations of the object, improving its ability to recognize the object regardless of its position or surrounding details.

Random Flipping

This technique mirrors the image along an axis. For example, a *horizontal flip* reflects left to right: for an image of width W, the pixel at (x, y) is mapped to $(W - 1 - x, y)$, so:

$$I'(x, y) = I(W - 1 - x, y).$$

A *vertical flip* inverts top to bottom:

$$I'(x, y) = I(x, H - 1 - y).$$

Random Rotation

This rotates the image by a random angle θ around its center (c_x, c_y). Each pixel (x, y) is mapped to

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x - c_x \\ y - c_y \end{pmatrix}.$$

In other words, the image is multiplied by the 2D rotation matrix $R(\theta)$. Rotation introduces orientation variance: the model sees objects at various angles, helping with its generalization.

Gamma Correction

This transformation is a nonlinear contrast adjustment defined as follows: given the pixel values $I(x, y)$ normalized to $[0, 1]$, we draw $\Gamma \sim \text{Uniform}(\gamma_{min}, \gamma_{max})$ (in our case $\gamma_{min} = 0.7, \gamma_{max} = 1.3$) and then apply the the following transformation:

$$I'(x, y) = 255 \times \left(\frac{I(x, y)}{255}\right)^{\Gamma}.$$

When $\Gamma < 1$, dark regions are brightened, when $\Gamma > 1$ dark regions are darkened.

Gaussian Blur

Gaussian blur involves smoothing the image by convolution with a Gaussian kernel. In particular a Gaussian blur with standard deviation σ computes

$$I'(x, y) = (I * G_\sigma)(x, y) = \sum_{u,v} I(u, v) \frac{1}{2\pi\sigma^2} \exp(-\frac{(x-u)^2 + (y-v)^2}{2\sigma^2}).$$

This replaces each pixel by a weighted average of its neighbors. Gaussian blur simulates low-resolution imaging. In data augmentation, blurring can make models robust to slight defocusing or noise variations.

Each of these classical augmentations preserves the class label while altering appearance. By applying them randomly during training, the model effectively sees many versions of each image. This diversity forces the network to learn invariant features (e.g. a lesion is recognized regardless of its exact position, orientation, or brightness) and thus improves generalization and robustness.

2.2.2 Fan Cropping

Clinical ultrasound frames are typically acquired in a fan-shaped (sector) field of view: pixels outside this sector carry no anatomical information and can introduce noise and waste computation. To focus the model on the relevant region, we apply a custom fan cropping step that uses the known sector (fan beam) binary mask to extract only the fan-shaped ROI from each frame.

Definition 2.2.1 (Fan Cropping)**.** *Let each original frame be $I(x, y)$ defined on a rectangular grid $\Omega = \{0, \dots, W-1\} \times \{0, \dots, H-1\}$. The binary fan mask is defined as follows:*

$$M(x, y) = \begin{cases} 1, & (x, y) \in \Omega_{fan}, \\ 0, & \textit{otherwise}, \end{cases}$$

where $\Omega_{fan} \subset \Omega$ is the set of pixels inside the fan beam. The fan-cropped image I_{fan} is then

$$I_{fan}(x, y) \;=\; I(x, y)\, M(x, y),$$

which zeroes out all pixels outside the fan. Next, we compute the minimal axis-aligned bounding box of Ω_{fan}:

$$x_{\min} = \min\{x \mid \exists\, y :\; M(x, y) = 1\}, \quad x_{\max} = \max\{x \mid \exists\, y :\; M(x, y) = 1\},$$

$$y_{\min} = \min\{y \mid \exists\, x :\; M(x, y) = 1\}, \quad y_{\max} = \max\{y \mid \exists\, x :\; M(x, y) = 1\}.$$

We then crop I_{fan} to this bounding box:

$$I_{crop}(u, v) = I_{fan}(x_{\min} + u,\; y_{\min} + v), \quad 0 \le u \le x_{\max} - x_{\min},\; 0 \le v \le y_{\max} - y_{\min}.$$

Finally, I_{crop} is resized to the original dimensions via bilinear interpolation.

The goals of this preprocessing step are:

- **Noise reduction:** Eliminates irrelevant background and machine annotations outside the fan, reducing noise.
- **Computational savings:** Cropping to the sector reduces the number of pixels processed by the network, speeding up both training and inference.
- **Focus on ROI:** Ensures the model only sees anatomically meaningful data, improving segmentation accuracy on the ovary.

In our pipeline, fan cropping is applied as a preprocessing step to both the frame and the relative ground truth mask. It has been crucial in reducing background noises and overall improving the model performance.

2.2.3 Max Numerosity

In our dataset, each clinical case corresponds to a video sequences of the same patient of varying length (number of frames). Without constraint, cases with very long videos would dominate the training set, potentially biasing the model and increasing computational cost. In Figure 2.5 is shown the distribution of frames per clinical case. To mitigate this effect, we introduce a preprocessing step called **max numerosity**, which limits the number of frames per case to a fixed upper bound $N_{\max}$.

Let a clinical case c consist of n_c frames:

$$\mathcal{F}_c = \{\, F_{c,1}, F_{c,2}, \ldots, F_{c,n_c} \}.$$

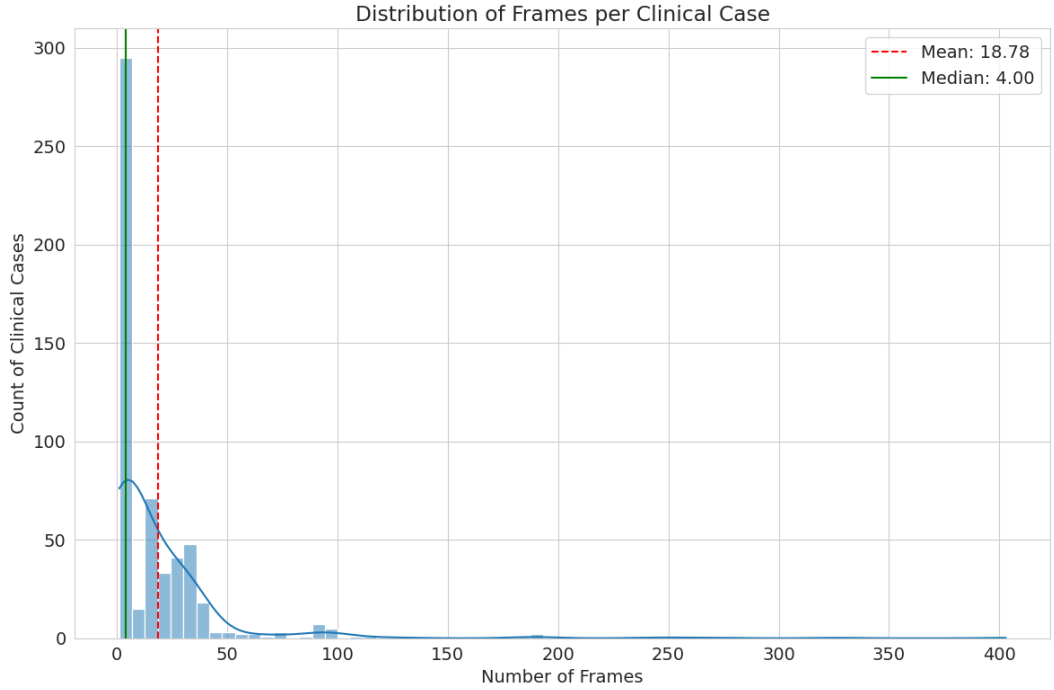


Figure 2.5: Number of frames per clinical case.

We choose a parameter $N_{\max} \in \mathbb{N}$. Then the selected subset of frames $\widetilde{\mathcal{F}}_c$ is defined as

$$\widetilde{\mathcal{F}}_c = \begin{cases} \mathcal{F}_c & \text{if } n_c \leq N_{\max}, \\ \{ F_{c,\, \lfloor \frac{k}{N_{\max}-1}(n_c-1) \rfloor + 1} \mid k = 0, 1, \ldots, N_{\max} - 1 \} & \text{if } n_c > N_{\max}. \end{cases}$$

In words:

- If the total frames n_c does not exceed $N_{\max}$, we keep all frames.
- Otherwise, we *equispatially* sample $N_{\max}$ frames by dividing the original sequence into $N_{\max}$ equal-length intervals and taking one representative frame from each.

The method is summarized in 1.
Thanks to the implementation of this preprocessing step we manage to obtain:

- **Balance across cases**: By capping at $N_{\max}$, no single patient contributes an outsized number of frames, reducing bias toward longer videos.
- **Computational efficiency**: Limiting frames per case bounds memory and compute requirements during training.

Algorithm 1: Equispatial sampling of frames

Input: Sequence of frames $\{F_{c,i}\}_{i=1}^{n_c}$ and parameter $N_{\max}$.
Output: Sampled set $\widetilde{\mathcal{F}}_c$ of size $\min(n_c, N_{\max})$.
if $n_c \leq N_{\max}$ **then**
  $\widetilde{\mathcal{F}}_c \leftarrow \mathcal{F}_c$
end
else
  initialize $\widetilde{\mathcal{F}}_c \leftarrow \varnothing$
  for $k \leftarrow 0$ **to** $N_{\max} - 1$ **do**
    $i_k \leftarrow \left\lfloor \frac{k}{N_{\max} - 1} (n_c - 1) \right\rfloor + 1$
    $\widetilde{\mathcal{F}}_c \leftarrow \widetilde{\mathcal{F}}_c \cup \{F_{c,i_k}\}$
  end
end
return $\widetilde{\mathcal{F}}_c$

- **Temporal coverage**: Equispaced sampling preserves coverage across the entire scan, ensuring that early, middle, and late frames are all represented.

By cross-validation, which will be explain in the next section, we determined the optimal value to be $N_{\max} = 3$, so each clinical case is represented by three frames. Introducing this parameter proved critical for enhancing model performance as well as mitigating bias and overfitting.
Another sampling possibility is **dynamical sampling** where, at every training epoch, instead of taking the frames equispatially distant, we randomly sample $N_{\max}$ frames from the full sequence.

Equispatial sampling guarantees uniform coverage of the entire sequence, while dynamic sampling increases the number of frames explored in training at the cost of occasionally selecting less informative frames. As we will show in Chapter 5, equispatial sampling yields more stable and higher segmentation accuracy in our ultrasound data.

2.3 Stratified Cross-Validation

In order to solve some of the problems stated above, a **Cross-Validation** technique is applied to the train dataset. Cross-validation is a resampling procedure used to evaluate a model's ability to generalize to an independent dataset. In classical settings we would use distinct datasets for training and validation. However, because annotated data are scarce, a single validation split may be unrepresentative: by chance the model can appear to perform very well or very poorly, and either outcome would not reliably reflect true

model performance. Instead of holding out a single validation set, the data are split into multiple subsets (folds), and the model is trained and validated repeatedly on different train/validation splits. This process allows us to get a more reliable estimate of validation performance, helps prevent overfitting, and leads to a better choices of the model's hyperparameters.

In particular:

- **Robust performance estimation**: By averaging results over multiple folds, we obtain metrics that are less sensitive to the particular choice of a single train/validation split.
- **Overfitting prevention**: Consistent performance across folds indicates that the model is not simply memorizing training samples.
- **Hyperparameter tuning**: Cross-validation provides feedback on how different model settings affect performance, leading to a more robust hyperparameters selection.
- **Helpful insights**: The variance of metrics across folds highlights where the model may struggle (for example, on rare lesion types).

To ensure reliable performance estimation and robust generalization across the diverse range of adnexal masses, a **Stratified 5-fold cross-validation** strategy has been adopted on the training set. This approach partitions the dataset into five equally sized subsets while preserving the distribution of the main attributes, such as histological type and the presence of solid and non-solid components.

Furthermore, given the moderate class imbalance observed in the segmentation masks (Figure 2.4), stratification is crucial, in order to avoid folds dominated by a particular mask-type combination. In Figure 2.6 is shown the consistent distribution of solid and non-solid masks per fold. To quantify this consistency, we compute an **imbalance score**, defined as the sum of the absolute deviations between the proportion of solid and non-solid masks in the training and validation sets of each fold. A lower score indicates better balance. Without this approach, the model could have very high performance on certain validation sets, while failing to generalize less represented subgroups, such as purely solid lesions. Moreover, since each video sequence contains **temporally correlated frames** we apply cross-validation at the video level, rather than at the frame level, to preserve temporal information and also to prevent frames from the same ultrasound sequence to appear simultaneously in training and validation splits.

2.3.1 Imbalance Score

The **imbalance score** is used for quantifying the consistency of the mask's type distribution, the lower the imbalance score is, the better the fold represents the original

Fold Statistics Summary

Fold	Train Frames	Val Frames	Train Solid %	Train Non-solid %	Val Solid %	Val Non-solid %	Imbalance Score
0	8058	2441	75.4%	96.1%	80.4%	88.3%	0.1278
1	8493	2006	75.9%	93.9%	79.7%	96.1%	0.0603
2	8384	2115	78.3%	93.8%	69.7%	96.4%	0.1115
3	8292	2207	76.7%	93.9%	76.2%	96.0%	0.0258
4	8769	1730	76.6%	93.9%	76.5%	96.1%	0.0232

Figure 2.6: Distribution of frames and masks in the different folds.

dataset. Mathematically, the imbalance score for a fold is the sum of absolute deviations of the train and validation mask ratios from the overall dataset ratios. Let:

- N = total number of frames
- S = total number of frames with solid components
- NS = total number of frames with non-solid components
- $N_{\text{train}}, N_{\text{val}}$ = number of frames in train / validation split
- $S_{\text{train}}, S_{\text{val}}$ = number of frames with solid components in train / validation split
- $NS_{\text{train}}, NS_{\text{val}}$ = number of frames with non-solid components in train / validation split

Then we define the ratios as:

$$r_{\text{solid}} = \frac{S}{N}, \quad r_{\text{nonsolid}} = \frac{NS}{N}$$

$$r_{\text{solid}}^{\text{train}} = \frac{S_{\text{train}}}{N_{\text{train}}}, \quad r_{\text{solid}}^{\text{val}} = \frac{S_{\text{val}}}{N_{\text{val}}}$$

$$r_{\text{nonsolid}}^{\text{train}} = \frac{NS_{\text{train}}}{N_{\text{train}}}, \quad r_{\text{nonsolid}}^{\text{val}} = \frac{NS_{\text{val}}}{N_{\text{val}}}$$

Then the imbalance score is computed as:

$$\text{ImbalanceScore} = \left|r_{\text{solid}}^{\text{train}} - r_{\text{solid}}\right| + \left|r_{\text{solid}}^{\text{val}} - r_{\text{solid}}\right| + \left|r_{\text{nonsolid}}^{\text{train}} - r_{\text{nonsolid}}\right| + \left|r_{\text{nonsolid}}^{\text{val}} - r_{\text{nonsolid}}\right|.$$

To summarize, the imbalance score quantifies how much the distribution of solid and non-solid frames in the training and validation sets deviates from the overall dataset distribution. A lower score indicates that the split preserves the global proportions more faithfully, while a higher score highlights stronger deviations and thus a more imbalanced partition. The introduction of the imbalance score is critical for constructing more balanced folds.

Chapter 3

Object Tracking

In this chapter, we formally define the **Object Tracking** problem and explore three representative families of trackers: detection-based methods, optical flow-based models, and spatio-temporal graphical models, and for each a state-of-the-art architecture will be presented. We then focus on an optical flow-based tracker, **RAFT**, and its enhanced variant **SEA-RAFT**, analyzing their design and innovations. Finally, we show how to integrate SEA-RAFT on top of U-Net segmentation masks to produce temporally coherent segmentation results.

3.1 Literature Review: Object Tracking

Object Tracking involves the estimation of object motion and state evolution across temporal sequences, ensuring consistent object localization over time. The problem can be formally define as the process of estimating the trajectory of one or more objects in a video sequence, given their initial locations or characteristics. Despite this straightforward definition, the tracking problem extends far beyond simple position estimation, in order to include a consistence object identity, the handling of appearance variations and the management of complex interactions between multiple objects and their environment. In order to solve the lack of temporal complexity in the U-Net model described in the previous chapter, we integrate a tracking stage **on top** of the frame-by-frame masks with the aim to:

- Enforce temporal consistency;
- Suppress spurious flicker (eliminating rapid changes in object segmentation masks between consecutive frames to preserve the original frames smoothness);
- Correctly propagate segmentation masks through occluded or low-contrast regions.

In this section we will review the three representative families of tracking methods that are usually applied in the context of **Object Tracking**.

3.1.1 Detection-based Tracking (SORT)

Simple Online and Real Time Tracking (SORT)[17] is one of the earliest approches for Object Tracking, it prioritizes computational efficiency and real-time performance while maintaining competitive tracking accuracy. SORT employs a **tracking-by-detection** structure, where object detection is performed independently on each frame (a common object detector is YOLO [19]), followed by a data association step that links detections across temporal sequences. SORT represents object states as

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T,$$

where (u, v) are the bounding box center coordinates, s is the *scale*, defined as the area of the bounding box and $r = \frac{w}{h}$ is the *aspect ratio*, where w and h are the width and height of the bounding box. By propagating (u, v, s) with a constant-velocity model (and keeping r fixed, unless the object's shape changes drastically), SORT's Kalman filter predicts where and how big each tracked object will be in the next frame, before new detections are associated. A **Kalman Filter**[18] is a mathematical algorithm that estimates the state of a dynamic system from a series of noisy measurements. This procedure does not limit to an estimate of the object's state (position, velocity) but also measure how confident the model is in that estimate. In SORT, the Kalman filter works as follows:

- **Initial Step**:The filter initializes a new track with the observed bounding box parameters $[u, v, s, r]$, and sets all velocity components to zero, creating the initial state vector $[u, v, s, r, 0, 0, 0, 0]$. Velocity estimation occurs through the iterative Kalman filter process as the object is tracked across subsequent frames.
- **Prediction Step**: For each frame predict where each tracked object should be based on its previous position and estimated velocity.
- **Update Step**: After an association phase (via the Hungarian Algorithm[20]), for each matched detection:
  - Compute the difference between the predicted $[u, v, s, r]$ and the detected $[u_{\text{det}}, v_{\text{det}}, s_{\text{det}}, r_{\text{det}}]$.
  - Calculate the **Kalman gain**, which indicates how much to trust the prediction versus the new measurement based on their uncertainties (for example, if the prediction is uncertain and the measurement is reliable the Kalman gain will be high and we will trust the measurement more).
  - Adjust the state by moving partially from the prediction toward the detection, weighted by the gain.
  - Reduce the state covariance (it measures the uncertainty of the prediction) because a real measurement has been preformed.

SORT is widely used because of its computational efficiency and suitability for real-time tracking applications. However, due to this simplicity, it has several limitations that could affect its performance in challenging tracking tasks. For example, the linear motion model, while computationally efficient, fails in representing complex motion patterns such as sudden direction changes or object occlusions. Moreover, it has been proved that SORT's performance are highly dependent on the quality of the object detector, which can undermine the robustness of the model.

3.1.2 Optical Flow based Model (FlowNet)

Optical Flow is a representation of the displacement field that maps pixels from one frame to the corresponding pixel in the next frame, indicating how pixels move from one frame to the next [21]. Optical flow estimation algorithms are responsible of computing such map. Moreover flow estimation is crucial for object tracking applications, as it provides motion information between consecutive frames, enabling the prediction of object locations and the understanding of motion patterns. **FlowNet** represents an important advancement in optical flow estimation, introducing the first end-to-end deep learning approach to this problem, substituting the traditional optimization-based methods. FlowNet introduced two CNN-based architectural variants: **FlowNetS**(Simple) and **FlowNetC**(Correlations), and they are both showned in Figure 3.1.

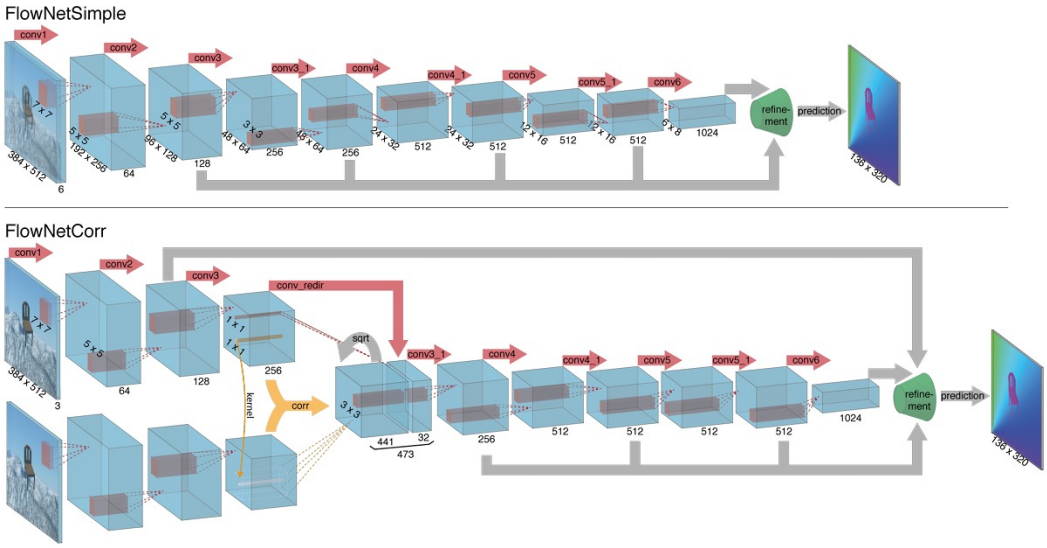


Figure 3.1: FlowNetS (top) and FlowNetC (bottom) architectures [21].

- **FlowNetS**: The FlowNetS architecture follows a straightforward encoder-decoder design, where two consecutive frames are stacked together and fed into a contracting

network (encoder) followed by an expanding network (decoder). Skip connections between encoder and decoder layers are inserted to help preserved fine-grained spatial information, essential for accurate flow estimation.

- **FlowNetC**: In FlowNetC a **correlation layer** has been added, in order to compute features correlation between the two input frames (that will be processed simultaneously and not stacked as before), performing patch comparison between the two feature maps. In details, given two multi-channel feature maps $\mathbf{f}_1, \mathbf{f}_2 : \mathbb{R}^2 \to \mathbb{R}^c$, with w, h and c being their width, height and number of channels, the correlation of two patches centered at $\mathbf{x}_1$ in the first map and $\mathbf{x}_2$ in the second map is defined as:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

for a square patch of size $K := 2k + 1$. This explicit correlation computation provides a more accurate flow estimation than the simple stacking approach used in FlowNetS.

In conclusion, FlowNet provides a roboust, efficient, end-to-end learning model for optical flow estimation, and modern tracking algorithm, like the one we will use for object tracking (**RAFT**) are build upon FlowNet's architecture.

3.1.3 Spatio-Temporal Graphical Model (CRFs)

Conditional Random Fields (CRFs) are a probabilistic framework for modeling structured prediction problems. In particular, CRFs are suited for object tracking due to their ability to model spatial and temporal consistency among subsequent frames [22][23].

Definition 3.1.1. *Let $G = (V, E)$ be a graph such that $Y = (Y_v)_{v \in V}$ so that Y is indexed by the vertex of G. Then (X, Y) is a **conditional random field** if, when conditioned on X, the random variables Y_v satisfy the Markov property with respect to the graph:*

$$p(Y_v | X, Y_w, w \neq v) = p(Y_v | X, Y_w, w \sim v),$$

where $w \sim v$ means that w and v are neighbors in G.

In particular the CRF models the conditional probability:

$$P(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_i \theta_i \phi_i(Y, X)\right)$$

where:

- $Z(X)$ is the partition function ensuring normalization;
- $\phi_i(Y, X)$ are feature functions that capture relationships between observations and labels;
- θ_i are learned parameters weighting the importance of each feature function.

Usually CRFs are represented as **factor graphs**, where nodes represent random variables and factors represent the relationships between them. In the context of object tracking we will have:

- **Unary Factors**: Model the relationship between individual observation and object states: $\psi_i(y_i, x_i) = \exp(\theta_i^T \phi(y_i, x_i))$. For example, in the context of object tracking applied to segmentation masks, Unary factors will indicate how likely is a pixel to belong to a particular class.
- **Pairwise Factors**: Model relationships between neighboring states $\psi_{ij}(y_i, y_j, x_i) = \exp(\theta_{ij}^T \phi(y_i, y_j, x_i))$. In our case, this terms will encourage neighboring pixels with similar features (like color or intensity) to have the same label, enforcing spatial smoothness.
- **Higher-order Factors**: Model complex interactions between multiple variables $\psi_C(y_C, x) = \exp(\theta_C^T \phi(y_C, x))$.

Modern tracking approaches use variants of CRFs, like **Linear-Chain CRFs** or **Fully-Connected CRFs**[24], combining them with deep learning networks in order to obtain robust, flexible and trainable models. However, CRFs-based architectures present some drawbacks such as a high computational complexity for complex graph structures and poor motion estimates (e.g., due to occlusion or low contrast) can propagate errors across frames, resulting in an inaccurate prediction. Despite these challenges, CRFs remain a powerful tool for Object Tracking tasks.

3.2 Recurrent All-Pairs Field Transforms

Recurrent All-Pairs Field Transforms (RAFT)[25] represents a significant advancement in optical flow estimation in order to address the fundamental limitations of previous methods highlighted in the previous sections, such as high computational complexity and challenges in correctly tracking occluded or fast-motion objects.

3.2.1 Model's Architecture

RAFT architecture is shown in Figure 3.2. It consists of three main components:

1. **Feature Encoder**: Extracts hierarchical feature representations from input frames.
2. **Correlation Volume**: Constructs and maintains a 4D correlation volume for correspondence matching.
3. **Update Operator**: Iteratively refines flow estimates through a recurrent neural network.

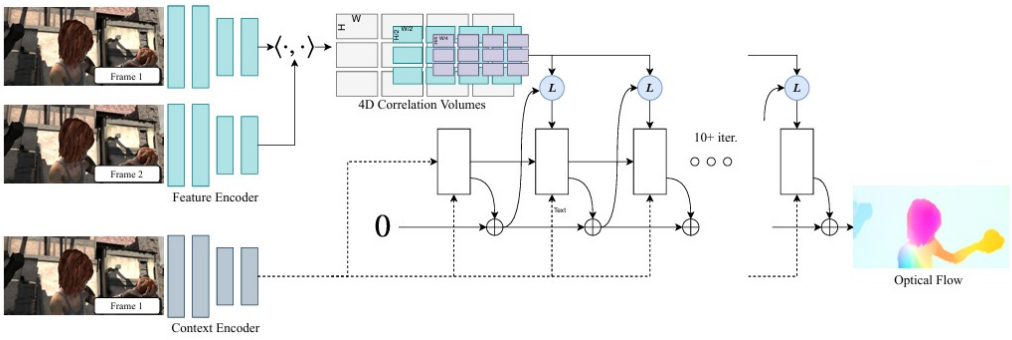


Figure 3.2: RAFT's architecture build by: (1) a feature encoder and a context encoder, (2) a correlation layer and (3) an update operator.

Feature Encoder

The feature encoder process both input frames I_1, I_2 through a shared convolutional neural network to extract multi-scale feature representations. In particular, RAFT's feature encoder consists of 6 residual blocks, 2 at 1/2 resolution, 2 at 1/4 resolution and 2 at 1/8 resolution, where each level captures different aspects of visual information. At last the encoder g_θ outputs features at 1/8 resolution $g_\theta : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{H/8\times W/8\times D}$, where D is set as 256. In addition to the feature network g_θ, a context network h_θ, that has the same architecture of the feature network but only extracts feature from the first frame, is used. This context network provides global information about the scene and help resolve ambiguities in motion estimation.

4D Correlation Volume

The 4D Correlation Volume is the key innovation in RAFT that ensures efficient all-pairs correlation computation. Given image features $g_\theta(I_1), g_\theta(I_2) \in \mathbb{R}^{H\times W\times D}$, the correlation volume $\mathbf{C}$ is computed as the dot product between all pairs of feature vectors:

$$\mathbf{C}(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H\times W\times H\times W}, \qquad C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh}.$$

Additionally, in order to efficiently handle both large and small displacements, RAFT constructs a **4-layer correlation pyramid** $\{\mathbf{C}^1, \mathbf{C}^2, \mathbf{C}^3, \mathbf{C}^4, \}$ by pooling the last two dimensions of the correlation volume with kernel sizes 1, 2, 4, and 8 and equivalent stride, so that $\mathbf{C}^k \in \mathbb{R}^{H \times W \times H/2^k \times W/2^k}$. Lastly, RAFT introduces efficient **lookup operations** to extract relevant information from the correlation volume:

$$L(\mathbf{C}, \mathbf{f}) = \text{Bilinear}(\mathbf{C}, \text{grid}(\mathbf{f})),$$

where $\mathbf{f}$ represents the flow estimate (initially set at zero, later updated iteratively as described in the next paragraph) and the lookup operation extracts correlation values in a local neighborhood around the predicted correspondences. In particular, given a current estimate of optical flow $(\mathbf{f}^1, \mathbf{f}^2)$, each pixel $\mathbf{x} = (u, v)$ in I_1 is mapped to its estimated correspondence in I_2: $\mathbf{x}' = (u + f^1(u), v + f^2(v))$, and the local grid around $\mathbf{x}'$ that we use to index from the correlation volume is defined as:

$$N(\mathbf{x}')_r = \{\mathbf{x}' + \mathbf{dx} \mid \mathbf{dx} \in \mathbb{Z}^2, \|\mathbf{dx}\|_1 \leq r\}$$

and since it is a grid of real number, bilinear sampling is used.

Iterative Updates

The update operator is implemented as a **Gated Recurrent Unit (GRU)** that iteratively refines flow estimates:

$$\textbf{Update Equation: } \Delta\mathbf{f}, h_{t+1} = \text{GRU}(h_t, x_t),$$

where h_t is the hidden state at iteration t, x_t is the input consisting of correlation features, flow features, and context features and $\Delta\mathbf{f}$ is the flow update that is added to the current estimate. Moreover, the update operator receives multiple type of information encoded in x_t:

- **Correlation Features**: Extracted from the correlation volume using lookup operations.
- **Flow Features**: Encoded representation of the current flow estimate.
- **Context Features**: Global scene information from the context network.

During this step a process of **Iterative Refinement** is performed, where we start with an initial flow estimate (usually zero) and then we apply the update operator multiple times, with the aim of producing a refined flow estimate. Lastly an upsampling procedure is performed in order to match the initial size.

Loss Function

RAFT is trained using a sequence loss that supervises intermediate flow estimates with exponentially increasing weights:

$$\mathcal{L} = \sum_{i=1}^{N} \gamma^{N-i} \left\| \mathbf{f}_{gt} - \mathbf{f}_i \right\|_1 ,$$

where:

- $\mathbf{f}_i$ is the flow estimate at iteration i;
- $\mathbf{f}_{gt}$ is the ground truth flow;
- γ is a discount factor (set at 0.8);
- N is the number of iterations.

Observation 3.2.1. *The loss function is used only during training to learn the GRU parameters. During both training and inference, flow estimates are generated through the iterative GRU update mechanism described above. The loss does not directly compute flow updates, as it trains the network to produce better flow updates through backpropagation.*

Advantages and Limitations of RAFT

RAFT brings several advantages over previous optical flow methods (such as **FlowNet**):

- **Dense All-Pairs Correlation**: By computing a full 4D correlation volume, RAFT captures fine-grained correspondences across the entire image, enabling accurate estimation even for small or fast-moving structures.
- **Iterative Refinement**: The recurrent update operator refines the flow estimate over multiple iterations, improving robustness to noise and occlusions and converging to a precise solution.
- **High Accuracy on Benchmarks**: RAFT has been shown to achieve state-of-the-art performance on popular optical flow datasets (e.g., Sintel, KITTI), demonstrating its reliability across diverse scenarios.

Limitations

Despite its strengths, RAFT has some drawbacks:

- **Computational and Memory Cost**: The construction and storage of the 4D correlation volume demand significant GPU memory, challenging for high-resolution inputs.

- **Low Inference Speed**: Iterative updates, while improving accuracy, increase inference time.

Overall, RAFT is an excellent choice for optical flow refinement and RAFT's architecture serves as a building block for more advanced flow methods designed to overcome its limitations, such as **SEA-RAFT**, which we introduce in the next section to address memory, speed, and robustness challenges in ultrasound imaging.

3.2.2 Simple Efficient Accurate RAFT

Simple Efficient Accurate RAFT (SEA-RAFT)[27] builds upon the original RAFT architecture introduced in the chapter before, improving the computational and accuracy limitations discussed above. While SEA-RAFT maintains the three-component architecture of RAFT (feature encoder, correlation volume, and update operator), it introduces several key modifications:

1. **Mixture of Laplace Loss Function**: Replaces the traditional RAFT L1 loss with a more robust loss function that better handles outliers and provides improved training stability.

2. **Direct Initial Flow Regression**: Instead of starting with zero initialization, SEA-RAFT directly regresses an initial flow estimate, significantly reducing the number of iterations required for convergence.

3. **Rigid-Flow Pre-training**: Incorporates a specialized pre-training strategy that improves generalization across different domains and datasets.

Mixture of Laplace Loss

SEA-RAFT replaces the standard L1 sequence loss with a **Mixture of Laplace (MoL)** loss, which models the prediction error at each iteration as a weighted sum of Laplace distributions. This gives the network the flexibility to handle both frequent small errors as well as occasional large errors without letting the outliers dominate the loss. For each pixel (u, v), the mixture is defined as:

$$\mathrm{MixLap}\big(x;\, \alpha, \beta_1, \beta_2, \mu\big) \;=\; \alpha\, \frac{\exp\big(-|x-\mu|/e^{\beta_1}\big)}{2\, e^{\beta_1}} \;+\; (1-\alpha)\, \frac{\exp\big(-|x-\mu|/e^{\beta_2}\big)}{2\, e^{\beta_2}}\,,$$

Particularly,

- The first component $\alpha\, \frac{\exp\left(-|x-\mu|/e^{\beta_1}\right)}{2\, e^{\beta_1}}$ is responsible for modeling frequent small errors;

- The second component $(1-\alpha)\,\frac{\exp\left(-|x-\mu|/e^{\beta_2}\right)}{2\,e^{\beta_2}}$ is responsible for handling occasional large outliers without overwhelming the loss.

We have:

- $\alpha(u,v) \in [0,1]$ is the mixing coefficient at each pixel that determinate the relative importance of each component;
- $\beta_1(u,v)$ and $\beta_2(u,v)$ are the log-scales of the two components;
- $\mu(u,v,d) = f_i(u,v,d)$ is the predicted flow at iteration i.

To explicitly encourage one component to behave like the standard L_1 loss, β_1 is fixed at zero (so that $e^{\beta_1} = 1$), making the first component equivalent to the standard L1 loss. The complete MoL loss is formulated as the discounted negative log-likelihood over all pixels, flow components, and refinement iterations:

$$\mathcal{L}_{\text{MoL}} \;=\; -\,\frac{1}{2\,H\,W}\sum_{i=1}^{N}\gamma^{\,N-i}\sum_{u,v}\sum_{d\in\{x,y\}}\log\Big[\text{MixLap}\big(\,f_{gt}(u,v,d);\;\alpha(u,v),\;0,\;\beta_2(u,v),\;f_i(u,v,d)\big)\Big],$$

where

- $\gamma \in (0,1)$ is the iteration discount factor;
- H, W are the frame height and width;
- N is the total number of refinement iterations:
- $\{\alpha,\, \beta_2\}$ are predicted by small auxiliary heads in the network.

In particular, a high α value indicates high confidence in the prediction, giving more weight to the L1 component. On the other hand, a lower α allows the model to predicts challenging regions (e.g. occlusions) by relying more on the second component, without overwhelming the loss.

Initial Flow Regression

Unlike RAFT, which initializes flow estimates to zero, SEA-RAFT includes a direct **flow regression network** (via multiple CNNs [26]) that predicts an initial flow estimate $\mathbf{f}_0$ from the context encoder, given both frames as input. This initial flow estimate leads to faster convergence and reduces computational costs, as fewer iterations are needed to reach convergence.

Rigid-Flow Pre-training Stratetgy

SEA-RAFT introduces a pre-training phase using **rigid-motion transformations** on the dataset TartanAir [28]. During this phase, the model learns to estimate optical flow for scenes undergoing rigid transformations (rotations, translations, and scaling). This pre-training strategy leads to a better model's generalization.

Overall, SEA-RAFT maintains the main RAFT's strengths, while addressing several limitations:

- **Improved Convergence Speed**: Direct initial flow regression reduces required iterations.
- **Enhanced Training Stability**: Mixture of Laplace loss provides more stable training.
- **Better Generalization**: Rigid-flow pre-training improves performance across different domains.

SEA-RAFT represents a significant step forward in optical flow estimation, combining the proven architecture of RAFT with innovative improvements that address its key limitations. Its enhanced accuracy, efficiency, and robustness makes it an excellent choice for our post-processing pipeline in medical image tracking. In the next section, we will explore how we leverage SEA-RAFT to warp and fuse U-Net segmentation masks, achieving temporally coherent ovary segmentation masks.

3.2.3 Mask Refinement with SEA-RAFT

To enforce temporal consistency in segmentation masks, we apply SEA-RAFT, to predict the optical flow between a pair of adjacent frames, performing a post-processing step on top of frame-by-frame U-Net predictions. The pipeline involves the following stages:

1. **Frame-by-frame Segmentation**: Each frame I_t is passed through the U-Net model described earlier to obtain an initial segmentation mask $M_t^{\text{U-Net}} \in [0,1]^{H\times W\times C}$, that does not take into account temporal consistency.
2. **Optical Flow Estimation**: For consecutive frames (I_{t-1}, I_t), temporal information is encoded in the optical flow predicted by SEA-RAFT: $F_{t-1\to t} \in \mathbb{R}^{H\times W\times 2}$, mapping each pixel in frame $t-1$ to its corresponding location in frame t.
3. **Mask Warping**: The previous mask $M_{t-1}^{\text{U-Net}} \in [0,1]^{H\times W\times C}$ is spatially transformed using $F_{t-1\to t}$ to obtain the next frame mask prediction:

$$M_t^{\text{warp}}(x,y) = M_{t-1}^{\text{U-Net}}(x-u, y-v), \quad (u,v) = F_{t-1\to t}(x,y)$$

with bilinear interpolation.

4. **Final segmentation mask**: The final segmentation mask is computed as a weighted average of the U-Net prediction and the warped segmentation mask from the previous step:
$$M_t^{\text{final}} = \alpha M_t^{\text{U-Net}} + (1 - \alpha) M_t^{\text{warp}},$$
where α is set as 0.5 in order to ensure equal contribution from both segmentation masks.

This tracking approach provides a solid method for encoding temporal information while maintaining the performance derived from U-Net segmentation. However, as we will show in Chapter 5, the results do not improve upon those obtained with U-Net alone. This limitation can be attributed to several factors. First, this method relies entirely on U-Net segmentation and cannot recover objects that are completely missed by the initial segmentation. Second, due to the lack of optical flow datasets for ovarian ultrasound images, SEA-RAFT has not been trained specifically for this domain, which presents particularly complex and challenging images.
Given these limitations, we introduce an alternative approach to address temporal consistency among frames in the next chapter: **end-to-end Video Object Segmentation models**, which aims to provide a more robust method at the cost of an increased model complexity.

Chapter 4

Video Object Segmentation

Video Object Segmentation (VOS) is the task of generating accurate, temporally coherent masks for one or more objects throughout an entire video sequence. Unlike frame-by-frame segmentation approaches, which ignore motion and can produce temporal inconsistent predictions, VOS methods integrate spatial appearance cues with temporal context, leveraging mechanisms such as memory banks, attention, and state-space models to propagate object identity over time.

In the medical domain, and particularly for ovarian ultrasound videos, VOS faces additional challenges: low contrast and noise, rapid probe motion causing large inter-frame displacements, and boundary details critical for diagnosis. General-purpose VOS architectures (e.g. STM, AOT, CUTIE) provide foundational techniques but often struggle to balance long-range modeling, boundary precision, and computational efficiency in this setting [41][35][39].

In this chapter, we first review three representative VOS architectures:

- **Space-Time Memory Networks (STM)**, which store past frames and masks in a differentiable memory bank.
- **Associating Objects with Transformers (AOT)**, which introduces an identity bank and hierarchical attention to handle multiple targets.
- **CUTIE**, which define an object transformer for better propagation over time.

We then motivate and introduce two methods suited for medical domain:

- **Med-SAM2**, which augments SAM2 with a self-sorting memory bank and prompting strategy to focus on the most informative frames.
- **ViViM**, which employs State-Space Models to capture long-range dependency and a boundary-aware affine constraint to sharpen edges.

Finally, we describe the training and inference pipelines for each model, including optimizers, loss functions and prompting strategy, and we describe the segmentation metrics that we will use to measure the performance of our models in Chapter 5.

4.1 Literature Review: Video Object Segmentation

Video Object Segmentation (VOS) aims to produce temporally coherent masks that delineate and track one or more target objects across an entire video sequence, preserving object identity and boundary precision over time. Unlike the two stages U-Net + tracker approach, where segmentation and temporal smoothing are separated, end-to-end VOS architectures embed temporal reasoning directly within the network, often through a learnable **memory bank** or attention mechanism that store relevant features from past frames.

Intuitively, this joint formulation is more robust: in a post-processing tracker, mask warping relies entirely on the quality of the U-Net predictions (e.g. if the U-Net fails to detect an object in one frame, the tracker can only propagate that error). By contrast, end-to-end VOS models learn to correct those errors leveraging temporal information. In this section, we review three state-of-the-art VOS methods by exploring their architectures, strengths, and limitations, before presenting the two approaches we use in our experiments: **MED-SAM2** and **ViViM**.

4.1.1 Space-Time Memory Networks

Space-Time Memory Networks (STM)[29] introduced the now popular **memory-based** architecture that maintains temporal consistency through the storage and retrieval of past frame information. STM architecture, as shown in Figure 4.1, consists of four main components:

1. **Memory Encoder**: Given T reference frames in the model's memory $\{I_t^M\}_{t=1}^T$ and their ground-truth masks $\{G_t^M\}_{t=1}^T$, the memory encoder E^M produces a pair of 3D key and value maps:

$$K^M \in \mathbb{R}^{T\times H'\times W'\times C_k}, \quad V^M \in \mathbb{R}^{T\times H'\times W'\times C_v},$$

where for each t, $(K_t^M, V_t^M) = E^M\big(I_t^M, G_t^M\big)$.

2. **Query Encoder**: For the current (query) frame I_Q, the query encoder E^Q produces a pair of 2D key and value maps:

$$K^Q \in \mathbb{R}^{H'\times W'\times C_k}, \quad V^Q \in \mathbb{R}^{H'\times W'\times C_v}, \quad (K^Q, V^Q) = E^Q(I_Q).$$

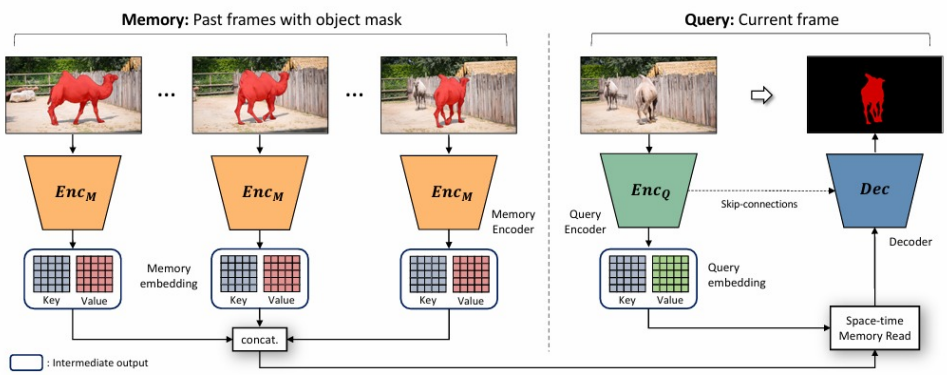


Figure 4.1: STM network consists of two encoders, one for the memory frames, the other for the query frame, along a space-time memory read block and a decoder.

3. **Space-time Memory Read**: At each query location i, we compute similarities $f(k_i^Q, k_j^M)$ of the query key against every key in the memory location j (which encodes both past frames and their segmentation masks), where k_i^Q represents the key vector at a single location i within the 2D key map K^Q and k_i^Q has dimensionality C_k (the key channel dimension). Those similarity scores become weights in a weighted sum of the corresponding memory values (effectively we will end up with a vector containing information about every pixel in all relevant past frames):

$$y_i = \left[v_i^Q, \frac{1}{Z_i}\sum_{\forall j} f(k_i^Q, k_j^M)\, v_j^M\right], \quad Z_i = \sum_{\forall j} f(k_i^Q, k_j^M),$$

where the similarity function is defined as:

$$f(k_i^Q, k_j^M) = \exp\left(k_i^Q \cdot k_j^M\right).$$

Here:

- k_i^Q is the key at query spatial location i.
- k_j^M and v_j^M are the key and value at memory location j.
- $[\cdot,\cdot]$ denotes concatenation of the retrieved memory embedding with the query value v_i^Q.

This step aggregates information from all past frames before passing the result into the decoder.

4. **Segmentation Decoder**: Finally, the decoder takes the output of the read operation and reconstructs the query frame's segmentation mask $\hat{M}_Q$ via convolutional layers.

Memory Update

After predicting the segmentation mask $\hat{M}_Q$ for the query frame, we can append $(I_Q, \hat{M}_Q)$ to the memory bank, potentially discarding the oldest entry to keep the memory bank size T fixed.

Strengths and Limitations

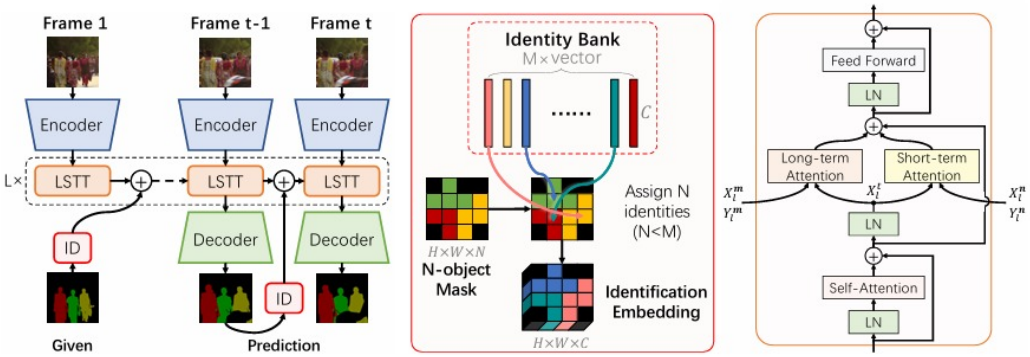
- **Strengths**: Long-term consistency due to the direct access to T past frames as well as robustness and capacity to handle occlusions.
- **Limitations**: The main drawback is that computational memory grows with the number T of stored past frames, in particular spatio-temporal attention over $T \times H' \times W'$ can be computationally expensive.

By explicitly learning to read from a space-time memory, STM successfully integrates spatial information with temporal context, in order to create a solid end-to-end VOS model. Over the years several extensions of the basic STM architecture have been proposed, one notable example is the **AOT** model, which we introduce in the next section.

4.1.2 Associating Objects with Transformers

Associating Objects with Transformers (AOT)[30] is the second video segmentation architecture we will explore, that tackles the problem by introducing an innovative identification mechanism that enables efficient tracking and segmentation of multiple objects simultaneously. Building upon STM, AOT addresses some of its limitations discussed above by creating a more efficient model focused on **multi-object scenarios**, by introducing two key innovations, as shown in Figure 4.2:

1. **Identification Mechanism for Multi-Object Association**: AOT proposes an identification system consisting of identification embedding and identification decoding components, in order to keep track of different objects along subsequent frames. The system maintains an identity bank $D \in \mathbb{R}^{M \times C}$ containing M identification vectors with C dimensions. For N target objects ($N < M$) in a video, each target is randomly assigned a different identification vector through a permutation matrix $P \in \{0,1\}^{N \times M}$:
$$E = \mathrm{ID}(Y, D) = YPD,$$
where $Y \in \{0,1\}^{T \times H \times W \times N}$ represents the one-hot masks of targets, and $E \in \mathbb{R}^{T \times H \times W \times C}$ is the resulting identification embedding. This step ensures that every object in the frame has its own unique ID, allowing multi-object tracking among frames.



(a) Overview (b) Identity assignment (c) l-th LSTT block

Figure 4.2: (a) Overview of AOT architecture. (b)Illustration of how the Identity Mechanism works. (c)The structure of the LSTT block (LN is a layer normalization module).

2. **Long Short-Term Transformer (LSTT)**: Unlike traditional single-layer attention mechanisms, AOT employs a hierarchical transformer architecture with three distinct attention modules:

   - **Self-Attention**: Learns associations among targets within the current frame
   - **Long-Term Attention**: Aggregates information from distant memory frames using non-local attention:

   $$\text{Att}^{LT}(X_l^t, X_l^m, Y^m) = \text{Att}^{ID}(X_l^t W_l^K, X_l^m W_l^K, X_l^m W_l^V, Y^m D)$$

   - **Short-Term Attention**: Handles nearby frames within spatial-temporal neighborhoods for temporal smoothness:

   $$\text{Att}^{ST}(X_{l,p}^t, X_{l,\mathcal{N}(p)}^n, Y_{l,\mathcal{N}(p)}^n) = \text{Att}^{LT}(X_{l,p}^t, X_{l,\mathcal{N}(p)}^n, Y_{l,\mathcal{N}(p)}^n)$$

   where X_l^t denotes input features at time t and block l, $\mathcal{N}(p)$ represents a spatial neighborhood centered at location p, $X_{l,p}^t$ is the feature of X_l^t at location p and W_l^K, W_l^V are projection parameters. Those attention layers ensure that we consider both short and long term information among frames.

3. **Multi-Object Decoding**: The identification decoding predicts:

   $$\hat{Y} = \text{softmax}(PL_D),$$

   where $L_D \in \mathbb{R}^{H\times W\times M}$ contains probability logits for all M identities, P is the same permutation matrix used in identification embedding and $\hat{Y}$ is the probability prediction of all the N targets. Finally, the predicted mask for each target is obtained by taking the argmax operation across the identity dimension: $\hat{M} = \text{argmax}(\hat{Y}, \text{dim} = M)$, which assigns each pixel to the most probable target identity, resulting in the final segmentation mask.

Strengths and Limitations

- **Strengths**: AOT provides an efficient handling of a variable numbers of target objects, performing robust multi-object association through identification mechanism. Moreover the hierarchical attention enables both long-term consistency and short-term smoothness.

- **Limitations**: Due to the multiple attention layers in LSTT blocks, AOT presents a computational overhead compared to the STM model introduced in the previous section. Additionally, AOT's identification mechanism is challenged by scenarios with very similar object appearances.

Although AOT was thought for multi-object scenarios, it is still a relevant state-of-the-art method and its architectural innovations (e.g. the hierarchical attention structure) could be a solid choice in single-object segmentation as well [30].

4.1.3 Putting the Object Back into Video Object Segmentation

Building on the pixel-level memory introduced in the two previous architectures, STM and AOT, **CUTIE**[31] introduces an object-level memory reading which aims at propagating the full object through time, instead of single pixels. CUTIE architecture is presented in Figure 4.3.

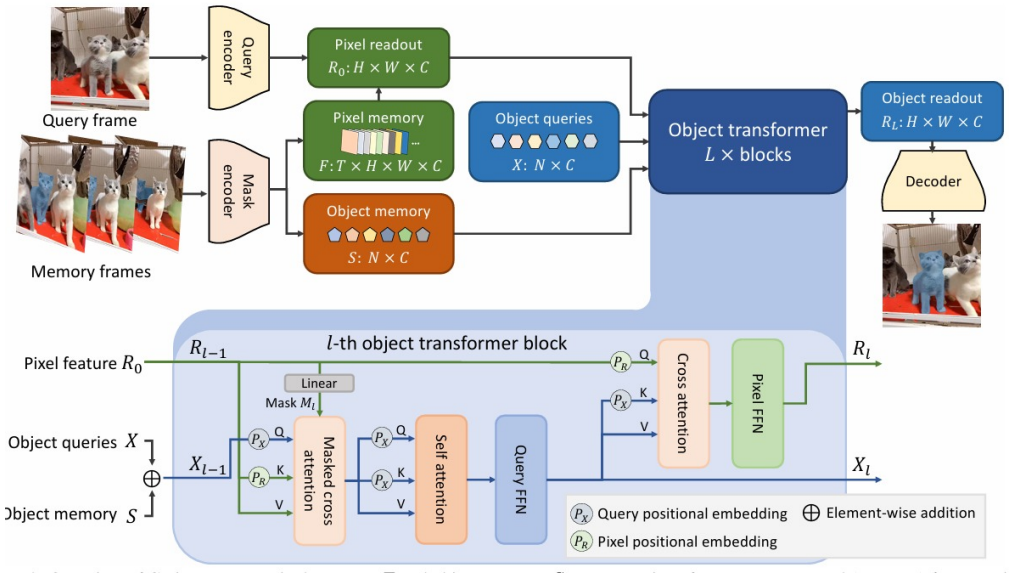


Figure 4.3: Overview of the CUTIE architecture, highlighting the novel object transformer with enrich the pixel feature with object-level semantics to produce a more accurate segmentation mask.

Given a query frame with encoded features $q \in \mathbb{R}^{HW \times C}$ and memory features $k \in \mathbb{R}^{THW \times C}$, traditional pixel-level methods compute attention weights:

$$A_{ij}^{\text{pix}} = \frac{\exp(d(q_i, k_j))}{\sum_m \exp(d(q_i, k_m))}$$

where $d(\cdot, \cdot)$ represents a distance function. Critically this approach alone lacks high-level consistency as each pixel q_i is matched independently, and can cause a drop in segmentation accuracy, especially in a scenario where the object is occluded for multiple frames and it reappears later on.

Object Transformer

CUTIE's architecture centers on an object transformer that employs a small set of learnable object queries $X \in \mathbb{R}^{N \times C}$ to facilitate top-down object-level reasoning. The system maintains two distinct memory components:

1. **Pixel Memory**: $F \in \mathbb{R}^{T \times H \times W \times C}$ storing high-resolution spatial features.
2. **Object Memory**: $S \in \mathbb{R}^{N \times C}$ storing compact object representations.

The object transformer operates through L transformer blocks, where each block l performs bidirectional updates:

$$X^l, R^l = \text{TransformerBlock}_l(X^{l-1}, R^{l-1}, S),$$

with initial conditions $X^0 = X + S$ and R^0 obtained from pixel memory readout.
The core innovation of CUTIE lies in the **masked cross-attention** mechanism. Standard cross-attention computes:

$$X_l' = \text{softmax}(Q_l K_l^T) V_l + X_l$$

where Q_l, K_l, V_l are linear transformations of queries and features. CUTIE introduces a masking matrix $M_l \in \{0, -\infty\}^{N \times HW}$ to enforce semantic separation:

$$X_l' = \text{softmax}(M_l + Q_l K_l^T) V_l + X_l.$$

The masking function is defined as:

$$M_l(q, i) = \begin{cases} 0 & \text{if} q \leq N/2 \text{ and } \hat{M}_l(i) \geq 0.5 \\ 0 & \text{if } q > N/2 \text{ and } \hat{M}_l(i) < 0.5 \\ -\infty & \text{otherwise} \end{cases}$$

where $\hat{M}_l(i) = \sigma(f_{\text{linear}}(R_{l-1}(i)))$ is a learned mask prediction. Specifically $M_l(q, i)$ determines whether the q-th query is allowed ($= 0$) or not($= -\infty$) to attend to the i-th pixel.

Object Memory Construction

The object memory $S \in \mathbb{R}^{N \times C}$ is constructed through weighted pooling operations. Given object features $U \in \mathbb{R}^{THW \times C}$ and N pooling masks $\{W_q \in [0,1]^{THW} : 0 < q \leq N\}$, the q-th object memory vector is:

$$S_q = \frac{\sum_{i=1}^{THW} U(i) W_q(i)}{\sum_{i=1}^{THW} W_q(i)}$$

The pooling weights incorporate foreground-background separation:

$$W_q(i) = \begin{cases} 0 & \text{if } q \leq N/2 \text{ and } M(i) < 0.5 \\ 0 & \text{if } q > N/2 \text{ and } M(i) \geq 0.5 \\ \sigma(f_{\text{PoolWeight}}(F(i) + R_{\sin}(i))) & \text{otherwise} \end{cases}$$

where σ is the sigmoid function, $f_{\text{PoolWeight}}$ is a 2-layer, N-dimensional MLP and $R_{\sin}$ represents 2D sinusoidal positional embeddings.

Positional Embeddings

The method employs positional embeddings to enable location-aware attention:

- **Object Query Embeddings**:

  $$P_X = E_X + f_{\text{ObjEmbed}}(S),$$

  where $E_X \in \mathbb{R}^{N \times C}$ is learnable and f_{ObjEmbed} is a linear projection.

- **Pixel Feature Embeddings**:

  $$P_R = R_{\sin} + f_{\text{PixEmbed}}(R_0),$$

  combining fixed sinusoidal embeddings with learned projections of initial readouts.

Strengths and Limitations

- **Strenghts**: The introduction of an Object Transformer and an Object Memory, along the classical Pixel Memory, allows CUTIE to produce more accurate segmentation masks and improve its robustness to the noise and occlusions [31].
- **Limitations**: CUTIE requires a segmentation mask for the first frame (One-shot VOS), which is problematic especially in inference settings. Additionally the method may struggle with sudden and significant appearance variation, that could be encountered in our dataset.

Overall, the three presented methods provide a solid choice for video object segmentation, however, considering the complexity of ultrasound videos, in order to perform accurate segmentation we will need specific architectures that focuses on medical data. For this reason we will introduce the two methods, Med-SAM2 and ViViM, that will be used for evaluation.

4.2 Med-SAM2: Medical Segment Anything Model 2

Ultrasound video analysis presents unique challenges that distinguish it from both natural video processing and static medical image segmentation. Unlike natural videos where objects maintain consistent appearance and motion patterns, ultrasound imaging involves probe movement, varying imaging angles, and anatomical structures that can appear and disappear as they move in and out of the frames [41]. Traditional video object segmentation methods like STM, AOT and CUTIE, while effective for natural scenes, struggle with the noise, low contrast, and temporal inconsistencies characteristic of ultrasound videos. In this section we will introduce **Med-SAM2** a transformer-based promptable end-to-end Video Object Segmentation model that addresses the challenges of medical VOS. Unlike the previously discussed methods, Med-SAM2 introduces a fundamentally different approach to temporal memory management, recognizing that in medical imaging, frame quality and relevance matter more than temporal proximity. In the following section we will start by describing SAM2, and then we will introduce Med-SAM2 along with training details.

4.2.1 Segment Anything Model 2

Segment Anything Model 2 (SAM2)[33] is an extension of the original Segment Anything Model [32], originally developed for promptable image segmentation, to promptable video (multi-)object segmentation. As we have seen in the three VOS methods presented before, the challenge in working with videos instead of images is overcome by the introduction of temporal awareness into the model, usually done by means of a **memory bank**, that stores past frames information, in order to obtain a spatio-temporal mask, called **masklet**. The main innovation with respect to the method above is the introduction of **prompts**, represented under the form of points, bounding boxes or segmentation masks of past frames, which are passed as input to the model in order to define a region of interest in which the model focuses its prediction. The notion of prompts is first introduced in SAM for single image segmentation, and it's naturally extended for the VOS task in the sense that the prompts are still passed on single frames, but those prompts are propagated during the segmentation to subsequent frames; prompts can also be passed during the segmentation to any frame.

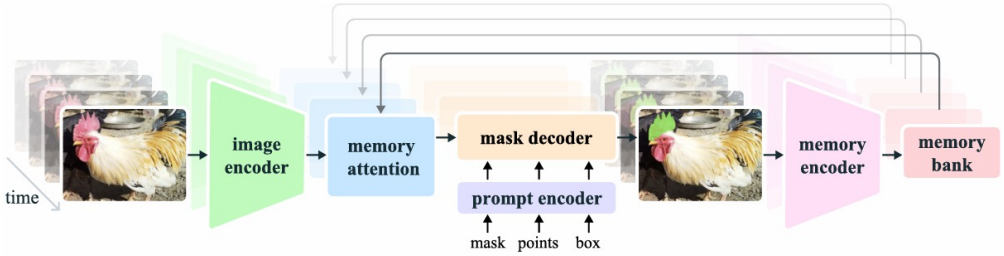


Figure 4.4: SAM2 architecture, depicting the new memory components to add temporal complexity to the model.

Architecturally SAM2 extend SAM to videos as shown in Figure 4.4, it is formed by 6 components:

- **Image Encoder**: Responsible for producing frame-by-frame feature embedding.
- **Memory Attention**: It conditions the current frame features on the past frames features, predictions and prompts.
- **Prompt Encoder**: Responsible for prompts embedding. Input prompts are divided in *sparse* prompt (points, bounding boxes) and *dense* prompts (masks).
- **Mask Decoder**: The mask decoder maps the image embedding, prompt embeddings, and an output token to a mask, producing the final output.
- **Memory Encoder**: It uses image embeddings from the image encoder with predicted mask information to produce memory features that are stored in the memory bank and propagated to subsequent frames for temporal-aware segmentation.
- **Memory Bank**: Responsible for maintaining information about past predictions, as well as past prompts.

We describe each component in detail below.

Image Encoder

The **Image Encoder** takes as input single frames (in a streaming approach) and its role is to provide feature embedding representing each frame. The encoder is built upon the Hiera architecture, a hierarchical Vision Transformer that has been pretrained using a Masked Autoencoder (MAE) [34]. This MAE pretraining strategy follows a self-supervised learning paradigm where random patches of input images are masked (typically up to 75% of all patches) and the model learns to reconstruct the missing pixels, this approach force the model to improve its generalization and robustness.

Memory Attention

This is the core component in order to define a temporal-aware model. The **Memory Attention** layer is responsible for conditioning the current frame's features on past frame's features (store in a memory bank) and new and past prompts. L transformer blocks are stacked, each one of them performs self-attention on the current frame (capturing long-range dependencies and contextual information), followed by cross-attention that considers past frames and prompts, followed by a Multi-Layer Perceptron (MLP).

Prompt Encoder

The **prompt encoder** in SAM2 is designed to handle different types of user input prompts (points, bounding boxes and masks) and convert them into a consistent 256-dimensional embedding format that the model can work with.

- **Point Prompts**: Each point gets converted into a $\mathbb{R}^{256}$ vector by combining two components:
  1. A positional encoding that captures where the point is located in the image.
  2. A learned embedding that indicates whether this point represents foreground (object we want to segment) or background (what we don't want).
- **Box Prompts**: A bounding box is represent as a pair of embeddings for its two corner points (top-left and botton-right), using the same logic described before.
- **Mask Prompts**: When a segmentation mask is provided as a prompt, we are interest in maintaining spatial correspondence with the image through convolutional processing. This process is done by progressive downsampling the input mask through various convolutional layers with different stride, in order to allow the model to learn hierarchical features at different scales, and GELU activations and layer normalization between each convolution.

Moreover, an occlusion head is also introduce in the model that predicts whether the object of interest is present in the current frame.

Mask Decoder

The **mask decoder** is a Transformer-based architecture that combines image and prompt information to generate a segmentation mask, the decoder structure is shown in Figure 4.5.

Due to the fact that a single input prompt may be ambiguous in the sense that it could lead to multiple valid masks, a modification to the model structure is necessary to resolve this ambiguity. The problem is eliminated by predicting multiple masks simultaneously

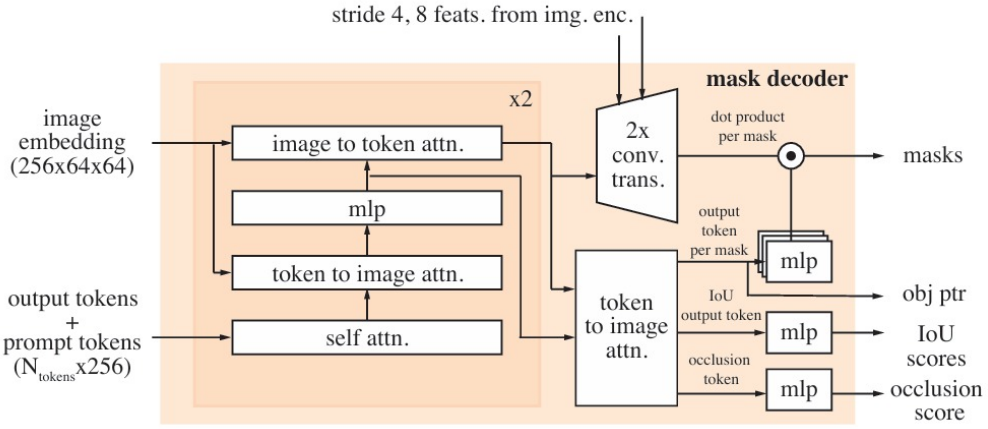


Figure 4.5: Mask decoder architecture. The design largely follows SAM, but the model also uses the mask token corresponding to the output mask as an object pointer and generate an occlusion score which indicates if the object of interest is visible in the current frame or not.

instead of predicting a single mask. If no subsequent prompts resolve this ambiguity, the model only propagates the mask with the highest predicted IoU for the current frame.

Memory Encoder

The **memory encoder** uses the image embeddings produced by the image encoder and it fuses them with the predicted mask information in order to produce memory features that will be stored in the memory bank and propagated to subsequent frames, to provide temporal-aware segmentation.

Memory Bank

SAM2's **memory bank** is similar to the ones of the models described before, it uses a First In First Out (FIFO) queue of memories of a fixed number of past frames and prompts, in order to retain information about past predictions and past prompts. As we will see, SAM2's memory bank is not optimal to work with US videos, and more in general with medical data. To solve those issues a new memory bank is constructed in **Med-SAM2**, which we will describe in the next section.

4.2.2 Med-SAM2 architecture

As seen above, in SAM2's original design, the system maintains a FIFO queue of memories to predict each frame [33]. This works well for natural videos where temporal order often correlates with relevance, due to the fact that recent frames are typically

most similar to the current frame. However, medical images present unique challenges that make temporal order less meaningful. In medical ultrasound, where anatomy may move in and out of view or change appearance, we do not always want to keep just the previous frames, but we may want to preserve those frames where the lesion or organ boundary was clearest, rather than simply the most recent frames. Due to those considerations, we introduce a variant of SAM2, **Medical SAM2 (Med-SAM2)**[35] that employs a **self-sorting memory bank** that dynamically selects informative frames embedding based on confidence and dissimilarity, regardless of temporal order, discarding low quality frames and keeping the frames that are most likely to help future predictions. Med-SAM2 architecture, including the new self sorting memory bank is represented in Figure 4.6, where all the other model's components are the same as in SAM2.

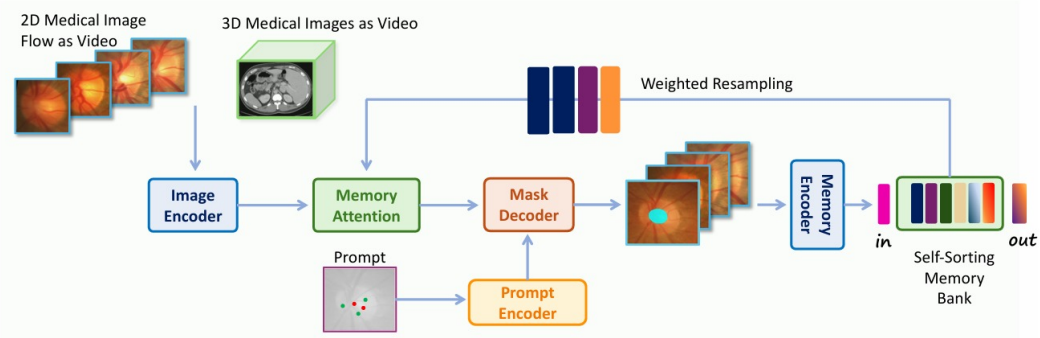


Figure 4.6: Med-SAM2 architecture largely follows SAM2, with the exception of the introduction of a self-sorting memory bank that maintains the most relevant past frames.

Self-Sorting Memory Bank

In this paragraph we describe more in depth Med-SAM2 self-sorting memory bank. Let $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T}$ be the input sequence of frames, and $\mathbf{P} = \{\mathbf{P}_t\}_{t=1}^{T}$ the optional prompts, then the model predicts $\mathbf{Y} = \{\mathbf{y}_t\}_{t=1}^{T}$ segmentation masks for each frame $\mathbf{x}_t$. After going through the image encoder ϵ_{img}, each frame outputs a feature embedding $\mathbf{E}_t = \epsilon_{\text{img}}(\mathbf{x}_t)$. The same process is extended to prompts, where we will get a prompt embedding $\mathbf{Q}_t = \epsilon_{\text{prompt}}(\mathbf{P}_t)$. We now introduce Med-SAM2 **self-sorting memory bank** M_t^{sort} that dynamically selects and retains the most informative embedding from past frames $\mathbf{E}_i$.

Updating the Memory Bank

At each time step t M_t^{sort} is updated based on M_{t-1}^{sort} and the embedding $\mathbf{E}_{t-1}$ of the previous frame. First the model predicts the segmentation mask $\mathbf{y}_{t-1}$ and computes the **IoU** confidence score c_{t-1} for frame $t-1$, such confidence score represents the model's

own estimate of how good its segmentation mask $\mathbf{y}_{t-1}$ for frame $t-1$ is. We then apply a quality filter $c_{t-1} \geq c_{\text{thresh}}$, so that only good enough predictions are stored in the memory bank. If the confidence score c_{t-1} satisfies the quality filter, then we form a candidate set $C = M_{t-1}^{\text{sort}} \cup \{\mathbf{E}_{t-1}\}$. As we want to maintain diversity among the memory bank's frames we compute the total dissimilarity for each embedding inside the candidate set:

$$D_i = \sum_{\substack{\mathbf{E}_j \in C \\ j \neq i}} 1 - \text{sim}(\mathbf{E}_i, \mathbf{E}_j), \quad \forall \mathbf{E}_i \in C$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function. The top K embeddings with the highest total dissimilarity are selected to form the updated memory bank:

$$M_t^{\text{sort}} = \text{TopK}_{\mathbf{E}_i \in C}(D_i),$$

where K is the size of the memory bank. This process guarantees that the memory bank contains the most relevant and diverse embedding. If the quality filter is not met, the memory bank is kept unchanged $M_t^{\text{sort}} = M_{t-1}^{\text{sort}}$.

Resampling the Memory Bank

Before computing the attention for frame t, higher selection probabilities are assigned to embeddings that are similar to the current one $\mathbf{E}_t$, in order to emphasize those during the attention operation. First we compute for the current frame $\mathbf{E}_t$ how similar it is to each stored embeddings $\mathbf{E}_i$ in the memory bank. The similarity scores are then normalized to create a probability distribution $p_{i,t}$:

$$p_{i,t} = \frac{\text{sim}(\mathbf{E}_t, \mathbf{E}_i)}{\sum_{\mathbf{E}_j \in M_t^{\text{sort}}} \text{sim}(\mathbf{E}_t, \mathbf{E}_j)}, \quad \forall \mathbf{E}_i \in M_t^{\text{sort}}.$$

Using this probability distribution, we perform resampling **with replacement** to create the importance weighted memory bank $\widetilde{M}_t^{\text{sort}}$:

$$\widetilde{M}_t^{\text{sort}} \;=\; \big\{\mathbf{E}_{i_k} \mid i_k \sim \text{Categorical}\big(\{p_{i,t}\}\big),\; k = 1, \ldots, K\big\}.$$

This step is performed to ensure that the memory bank dynamically adapts to emphasize the most relevant past embeddings for the current segmentation task. By performing sampling with replacement similar embeddings to the current one get amplified in the sense that they appear multiple times, and they will have more influence in the attention mechanism that is performed after, while very dissimilar frames will not be consider during segmentation.

4.2.3 Training And Inference

In this section we detail the training framework and inference pipeline for Med-SAM2 on our ovarian ultrasound video dataset, focusing in detail to the loss function used and the prompting strategy adopted.
Following the concepts introduced in Chapter 2, we train the model, starting from SAM2's pre-trained weights [33], with a 5-fold cross validation, where the training set is split into five stratified folds preserving the distribution of histological types and mask combinations. We tuned the model's hyperparameters by training on four folds and validating on the hold-out fold, rotating for three different hold-out folds, effectively performing three different trainings. Lastly, with optimal hyperparameters fixed, Med-SAM2 is retrained on the entire training set (all five folds combined) for the final model weights, maximizing data utilization. Beforehand, all the data is preprocessed with the same steps described in Chapter 2.

Optimizer and Scheduler

We trained our model with AdamW optimizer using weight decay, while the learning rate η follows a OneCycleLR scheduler over each epoch:

$$\eta(t) \text{ ramps from } \eta_{\min} \text{ to } \eta_{\max} \text{ and back.}$$

This cyclic schedule accelerates convergence (high learning rate helps escape local minima) and avoids overfitting by regularizing high-learning rate phases, the learning rate over each epoch is presented in Figure 4.7.

Loss Function

In order to address class imbalance and boundary precision, we optimize a composite loss, which is a weighted average between a **Focal loss** [37] and **Tversky loss** [38].

Definition 4.2.1 (Focal loss)**.** *: The focal loss is defined as:*

$$\mathcal{L}_{Focal}(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t),$$

where p_t is the predicted probability for the true class t, α_t is the class-specific weighting factor for class t and γ is the focusing parameter that controls how much to down-weight easy examples.

The focal loss addresses class imbalance through two complementary mechanisms. First, the focusing parameter γ automatically down-weights easy examples where the model exhibits high confidence in the correct prediction (p_t close to 1), while maintaining full loss contribution for hard examples with low confidence (p_t close to 0). When

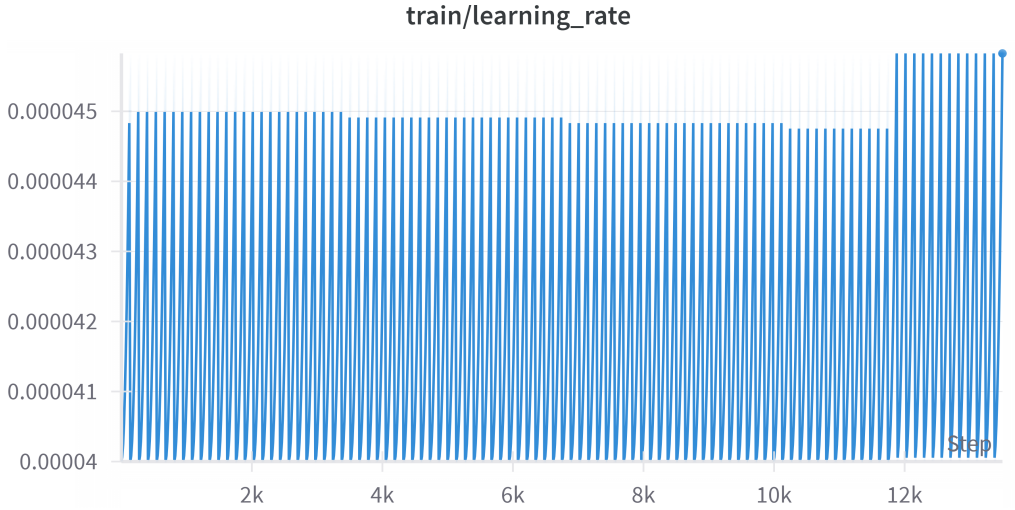


Figure 4.7: Learning rate during Med-SAM2 training with OneCycleLR, where η jumps from $\eta_{\min}$ to $\eta_{\max}$.

$\gamma = 0$, the focal loss reduces to standard cross-entropy; increasing γ progressively focuses training on challenging examples. Second, the class-specific weighting factor α_t provides explicit control over class importance, typically set inversely proportional to class frequency to help the model correctly classify underrepresented classes. For our segmentation task, we set $\alpha_t = [1/f_0, 1/f_1, 1/f_2]$ where f_i represents the frequency of class i (background, solid and non-solid) in the training dataset, and γ is set equals to 2.

Definition 4.2.2 (Tversky loss)**.** *: The Tversky loss is defined as:*

$$\mathcal{L}_{Tversky} = 1 - \frac{\sum_i p_i g_i}{\sum_i p_i g_i + \alpha \sum_i p_i(1 - g_i) + \beta \sum_i (1 - p_i) g_i},$$

where p_i is the predicted probability for pixel i, g_i is the ground truth label, and α and β are hyperparameters that control the penalty for false positives and false negatives, respectively.

The Tversky loss generalizes the Dice coefficient by introducing asymmetric penalties for false positives and false negatives through the parameters α and β. When $\alpha = \beta = 0.5$, the Tversky loss reduces to the standard Dice loss. By adjusting these parameters, we can explicitly control the trade-off between precision and recall: increasing α penalizes false positives more heavily (favoring higher precision), while increasing β penalizes false negatives more severely (favoring higher recall). This asymmetric penalization is particularly valuable for boundary-sensitive segmentation tasks where different

types of errors have varying clinical or practical significance. The Tversky loss directly optimizes for overlap-based metrics, making it especially effective at improving boundary precision and connectivity of segmented regions. Unlike pixel-wise losses that treat each pixel independently, the Tversky loss considers global shape properties and regional coherence, leading to more spatially consistent predictions. In our implementation, we set $\alpha = 0.3$ and $\beta = 0.7$ to prioritize recall over precision, minimizing false negatives in the model's predictions.
We thus define the final loss as:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{Focal}} + (1 - \lambda)\mathcal{L}_{\text{Tversky}},$$

where we set $\lambda = 0.4$, to give more relevance to Tversky loss.

Prompting Strategy

- **Prompting Strategy (Training)**: Following SAM2 original training, we also train our model passing as prompts points in order to simulate user interaction (clicks). For each training frame:

  1. We sample one *positive* click uniformly at random from the pixels of the target class (either solid or non-solid).
  2. We sample one *negative* click uniformly at random from the background (outside all object masks).

  These two-click prompts, one inside the region of interest and the other one outside, improve the model's boundary precision and produce a more accurate segmentation mask.

- **Prompting Strategy (Inference)**: In the absence of ground-truth masks during inference, we cannot provide positive or negative clicks. Instead, we create a fake prompt by placing four clicks at the corners of the frame, in order to pass as prompt the bounding box containing the full image.

Details on the training hyperparameters (e.g. learning rate, batch size, memory bank size) and the segmentation results will be presented in Chapter 5.

4.3 ViViM: Video Vision Mamba for Efficient Long-Range VOS

In the previous sections we have examined the challenges of medical VOS and demonstrated how general VOS models (AOT, CUTIE) struggle to perform accurate segmentation in medical contexts. To address these issues, we introduced Med-SAM2, which

implements a self-sorting memory bank that considers the most relevant past frames rather than simply using the last N frames. However, as we will explore in the next chapter, transformer-based models like Med-SAM2 have limitations in constructing long-term dependencies from a computational complexity perspective. This poses significant problems when processing longer sequences with limited memory resources. To overcome these computational challenges, we introduce State Space Models (SSMs), which enable efficient long sequence modeling. Specifically, we present **ViViM** (Video Vision Mamba)[39], an SSM-based model that incorporates temporal complexity while adding an improved boundary-aware constraint across frames. This approach addresses both the efficiency limitations of transformer architectures and the temporal modeling requirements of medical video segmentation.

4.3.1 State Space Models

In this section we will introduce **State Space Models (SSMs)**, which are the founding architecture of ViViM. SSMs are a class of machine learning algorithms used to make predictions about dynamic systems by modeling how their internal state evolves over time through differential equations. From Figure 4.8 it can be seen that an SSM is based on three time-dependent variables:

- $x(t) \in \mathbb{C}^n$ represents the n state variables.
- $u(t) \in \mathbb{C}^m$ represents the m state inputs.
- $y(t) \in \mathbb{C}^p$ represents the p outputs.

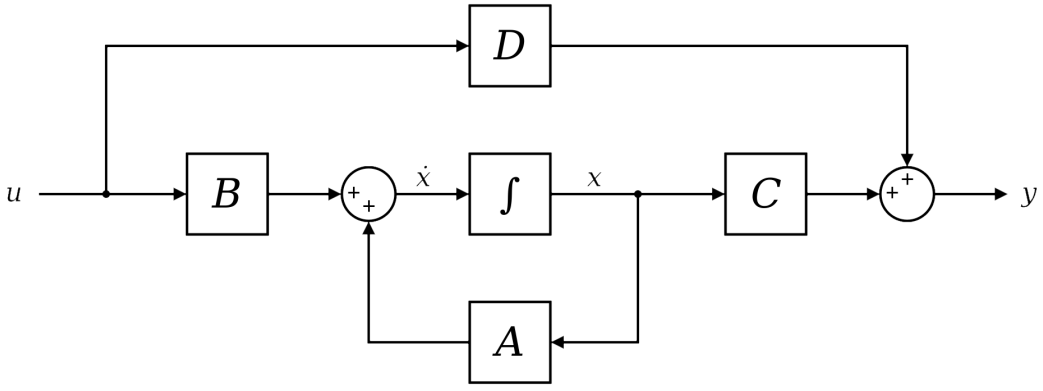


Figure 4.8: Block diagram representation of the continuous state-space equations.

As well as four learnable matrices:

- $A \in \mathbb{C}^{n \times n}$ is the state matrix, responsible for the control of how the current state evolves into the next state.
- $B \in \mathbb{C}^{n \times m}$ is the control matrix, that describes how external inputs influence the state evolution.
- $C \in \mathbb{C}^{p \times n}$ is the output matrix.
- $D \in \mathbb{C}^{p \times m}$ is the command matrix, that allows inputs to directly influence outputs.

SSMs can be reduce to the following system of equations:

$$x'(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

The first equation, the state evolution equation, tells us how the internal state evolves over time based on the current state and inputs, while the second one, the output equation, tells us how output are generated from the current state and inputs. Moreover, since the term $Du(t)$ can be seen as a skip connection and it is therefore easy to compute, it will be set at 0, simplifying the previous system to:

$$x'(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

Since the resulting system is continuous, it must be discretized before it can be actually used in our deep learning model.

Discrete View

The structured state space sequence models (also known as S4) and Mamba [40], which are at the basis of ViViM, are the discrete version of the continuous system define above, where the matrices A, B are discretized following the zero-order hold transformation (ZOH), which is defined as follows:

$$\overline{A} = \exp(\Delta A), \quad \overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I)\Delta B,$$

where Δ represents the time interval between consecutive discrete observation (i.e. $\Delta = (t_n - t_{n-1})$, and $\overline{A}, \overline{B}$ are the discretized matrices. Moreover since C and D represents instantaneous mappings from state and input to output respectively, they do not require discretization and remain unchanged in the discrete-time formulation. Hence the final discrete system will be:

$$x_t = \overline{A}x_{t-1} + \overline{B}u_t$$

$$y_t = Cx_t$$

Convolutive View

By iterating the equation of the previous system, we will get that:

$$x_0 = \overline{B}u_0$$

$$x_1 = \overline{A}x_0 + \overline{B}u_1 = \overline{AB}u_0 + \overline{B}u_1$$

$$x_2 = \overline{A}x_1 + \overline{B}u_2 = \overline{A}(\overline{AB}u_0 + \overline{B}u_1) + \overline{B}u_2 = \overline{A}^2\overline{B}u_0 + \overline{AB}u_1 + \overline{B}u_2$$

Then we can write the second equation as:

$$y_0 = Cx_0 = C\overline{B}u_0$$

$$y_1 = Cx_1 = C\overline{AB}u_0 + C\overline{B}u_1$$

$$y_2 = Cx_2 = C\overline{A}^2\overline{B}u_0 + C\overline{AB}u_1 + C\overline{B}u_2$$

Hence, by defining the convolutional kernel $\overline{K}_t = (C\overline{B}, C\overline{AB}, \ldots, C\overline{A}^t\overline{B})$, we will have $y = u * \overline{K}$.

This convolutive formulation reveals that the discrete SSM can be computed as a convolution between the input sequence u and the convolutional kernel $\overline{K}$. This dual representation (recurrent or discrete view and convolutional view) enables the model to alternate between the two equivalent formulation:

- **Training Phase**: The model utilizes the convolutive view to enable efficient parallel computation across the entire sequence using Fast Fourier Transform (FFT) algorithms. This allows the model to process all time steps simultaneously, making training more computationally efficient than Transformers.

- **Inference Phase**: The model switches to the recurrent view for autoregressive generation, processing tokens sequentially using the discrete formulation $x_t = \overline{A}x_{t-1} + \overline{B}u_t$. This provides constant memory usage regardless of sequence length and eliminates the need to recompute previous states.

The key idea is that both views are mathematically equivalent but computationally optimized for different scenarios. The convolutional view $y = u * \overline{K}$ is optimal when the entire input sequence is available (as it is during training), while the recurrent view excels for sequential generation (inference). This dual-mode operation allows S4 and Mamba architectures to combine the parallel training efficiency of CNNs with the memory-efficient sequential processing of RNNs, providing the best of both architectures.

4.3.2 Selective State Space Models (Mamba)

While the S4 architecture established the foundation for efficient state space models in deep learning, it still faced limitations in terms of selectivity and context-dependent processing. Mamba [40] addresses these challenges by introducing a selective mechanism that allows the model to dynamically focus on relevant information while filtering out irrelevant content, making it particularly effective for long sequence modeling.

Selection Mechanism

Traditional SSMs, like S4, use fixed parameters $\overline{A}$, $\overline{B}$, and C that remain constant across all inputs. This limitation means the model cannot adaptively focus on different parts of the input sequence based on content (similar to what Transformers do). Mamba introduces selectivity into the model by making these parameters input-dependent:

$$\overline{B}_t = s_B(x_t), \quad C_t = s_C(x_t), \quad \Delta_t = s_\Delta(x_t),$$

where s_B, s_C, and s_Δ are learned functions (typically linear projections) that compute parameter values based on the current input x_t:

- Δ_t **(timescale parameter)**: Controls how much the current input affects the hidden state update. A larger Δ_t makes the model focus more on the current input, while a smaller Δ_t preserves more of the previous state.
- $\overline{B}_t$ **(input matrix)**: Determines how much the current input influences the state evolution.
- C_t **(output matrix)**: Controls which aspects of the hidden state are relevant for the current output, enabling selective information retrieval.

As shown in Figure 4.9 a complete Mamba Block consists of:

1. **Input Projection**: Linear transformation to project the input to higher dimension.
2. **Selective SSM**: The selective state space layer.
3. **Activation Function**: Typically SiLU (Swish) activation function.
4. **Output Projection**: Linear transformation back to model dimension.
5. **Residual Connection**: Skip connection around the entire block.

Mamba's selective mechanism provides several key advantages:

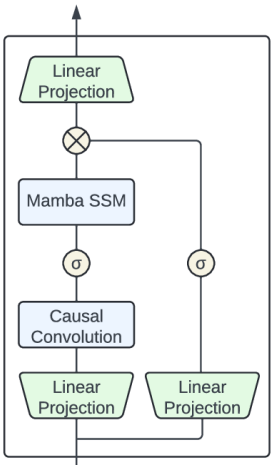


Figure 4.9: Mamba architecture, with at its core the Selective SSM mechanism described previously.

- **Content-based Filtering**: The model can ignore irrelevant information by learning appropriate Δ_t, $\overline{B}_t$, and C_t values.
- **Improved Long-range Dependencies**: Selective parameters allow the model to maintain important information over long sequences while forgetting irrelevant details.
- **Efficiency**: Maintains the linear complexity of traditional SSMs while improving the model performance.

While standard Mamba blocks excel at processing sequential data, their application to video understanding presents a fundamental limitation. Standard Mamba processes individual frames by treating spatial patches as sequential tokens, capturing intra-frame spatial dependencies, however, this approach fails to model the inter-frame temporal relationships that are at the basis of a VOS task. To address this limitation we introduce **Temporal Mamba Blocks**, that are the founding blocks of ViViM.

4.3.3 ViViM Architecture

In this section, we introduce **ViViM**[39] a Video Vision Mamba model for medical VOS, based on an extension of Mamba blocks in video settings, Temporal Mamba blocks, in order to add temporal complexity to the model. ViViM architecture is shown in Figure 4.10 and it is formed by the following components:

1. **Hierarchical Encoder**: The encoder processes video data through multiple stages with decreasing spatial resolution and increasing feature complexity, in order to

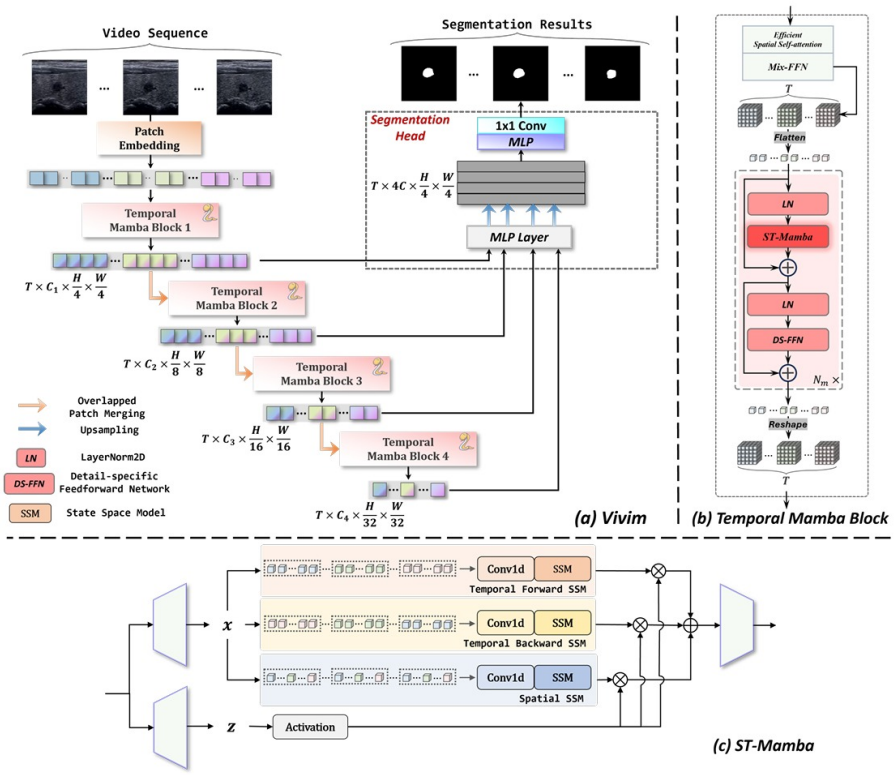


Figure 4.10: a) The overview of ViViM architecture, where the video is fed into a hierarchical encoder, followed by ST-Mamba blocks and a final CNN-based segmentation head. b) Temporal Mamba Block overview. c)ST-Mamba layer structure, including the three directional selective scan mechanism.

generate multi-scale feature representations that capture fine-grained local details at early layers and global context at deeper layers.

2. **Spatio-Temporal Mamba Block**: The key innovation of ViViM lies in the **ST-Mamba blocks**, which allows ViViM to process video sequences by introducing spatio-temporal selective scanning. Specifically, in order to explicit the relationship among frames, patches of each frame are unfolded along rows and column into sequences, and then those frame sequences are concatenated to constitute the temporal-first sequence, which, as we can see in Figure 4.10(c), is scanned along along the forward and backward directions to explore bidirectional temporal dependencies. Simultaneously, a spatial-first sequence is constructed by stacking patches along the temporal axis, and a scanning operation is performed in order to integrate information of each pixel from all frames (standard selective SSM model procedure). The spatio-temporal selective scan mechanism with three directions guarantees us that our model perform both spatial and temporal aware segmentation.

3. **Decoder**: CNN-based segmentation head that produces accurate, boundary-aware segmentation masks.

One of the most significant advantages of ViViM over Transformer-based video models (like Med-SAM2) lies in its computational efficiency. Both SSMs in ST-Mamba and self-attention in Transformers provide a crucial solution to model spatio-temporal context. Given a video sequence $K \in \mathbb{R}^{1\times T\times M\times D}$, where T represents the temporal sequence length (number of frames) M represents the spatial sequence length (number of patches per frame), D represents the feature dimension (number of channels) and N is the internal state size (hidden dimension) used inside the state-space model (in ST-Mamba is set to 16 by default), then the computational complexities of a global self-attention and SSM are:

$$\Omega(\text{self-attention}) = 4(TM)D^2 + 2(TM)^2D,$$
$$\Omega(\text{SSM}) = 4(TM)(2D)N + (TM)(2D)N^2,$$

so:

- The self-attention cost scales like $O((TM)^2D + (TM)D^2)$, which is quadratic in the total number of tokens $T \times M$.
- The SSM cost scales like $O((TM)DN + (TM)N^2)$, i.e. linear in TM (the sequence length), with additional cost proportional to the N^2.

Because N is much smaller than TM, the SSM in ST-Mamba achieves near-linear complexity, making it more efficient for long video sequences than transformer based methods, like Med-SAM2.

4.3.4 Training and Inference

The same approach described for Med-SAM2 is applied to ViViM training: we perform a 5-fold cross validation, then we proceeded to train ViViM on four folds and validating in the hold-out fold, rotating for three different hold-out folds, in order to fine tune the model hyperparameters. Lastly, we retrained ViViM on the full training set, comprising all five folds. Our data is preprocess with the same steps described in Chapter 1, like we did in Med-SAM2.

Optimizer and Scheduler

We trained our model with AdamW optimizer using weight decay, while the learning rate η follows a CosineAnnealingLR scheduler over each epoch. Cosine annealing smoothly decays the learning rate from its initial value $\eta_{\max}$ down to a minimum value $\eta_{\min}$ following a half-cosine curve, which helps avoiding sharp drops and encourages the optimizer to explore more early on and fine-tune later. Concretely, if you schedule over E epochs, the learning rate at epoch t is:

$$\eta(t) = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{t}{E}\pi)).$$

In particular at $t = 0$, $\eta(0) = \eta_{\max}$ and at $t = E$, $\eta(E) = \eta_{\min}$. The learning rate η during traninig can be seen in Figure 4.11

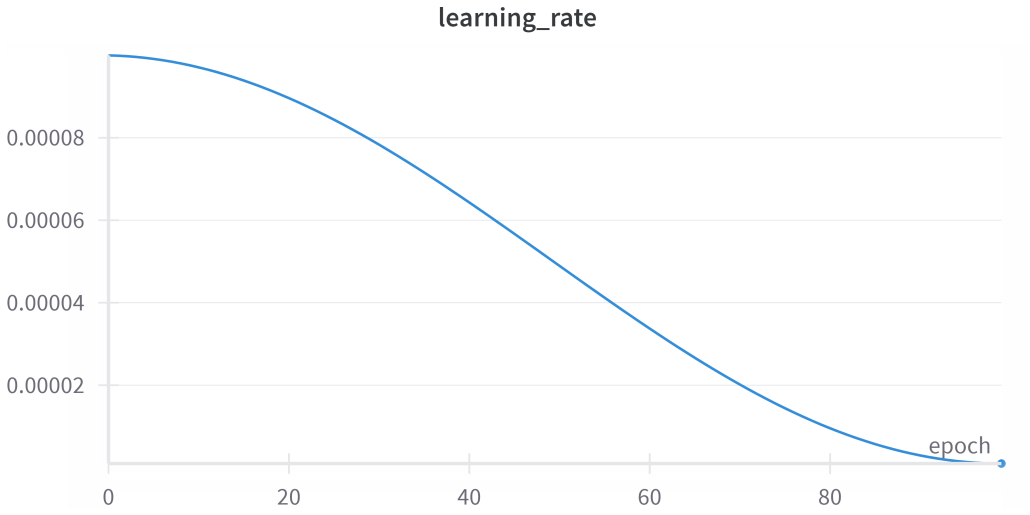


Figure 4.11: Learning rate during ViViM training with CosineAnnealingLR.

Loss Function

To encourage accurate, boundary-sharp, and temporally consistent segmentations, we train ViViM with a composite loss combining three terms:

$$\mathcal{L} = \mathcal{L}_{\text{seg}} + \lambda_1 \mathcal{L}_{\text{affine}} + \lambda_2 \mathcal{L}_{\text{ce}},$$

where λ_1, λ_2 are both set to 0.3.

1. Segmentation Loss $\mathcal{L}_{\text{seg}}$. As we did in Med-SAM2, we use a weighted average of class-balanced focal loss and Tversky loss to handle class imbalance and emphasize boundary accuracy:

$$\mathcal{L}_{\text{seg}} = \alpha \mathcal{L}_{\text{Focal}} + (1 - \alpha) \mathcal{L}_{\text{Tversky}}.$$

where, as we did before, α is set as 0.4, in order to emphasize more the Tversky loss term.

2. Boundary-aware Affine Loss $\mathcal{L}_{\text{affine}}$. To enforce structured, precised boundaries and discourage temporal copying, we:

1. Extract ground-truth edges B^t_{gt} and predicted edges B^t_{pred} on each small patch via Sobel filtering and a lightweight boundary head.
2. Use a pretrained MLP to compute two patch-wise affine matrices: $\hat{\theta}^t_i$ aligning $B^t_{pred} \to B^t_{gt}$, and $\hat{\theta}^1_i$ aligning $B^t_{pred} \to B^1_{gt}$.
3. Minimize

$$\mathcal{L}_{\text{affine}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \Big(\Delta_1 \cdot \|\hat{\theta}^t_i - I\|_F \; - \; \Delta_2 \cdot \|\hat{\theta}^1_i - I\|_F \Big),$$

where $\|\cdot\|_F$ is the Frobenius norm, I the identity, N_p denotes the number of patches and Δ_1, Δ_2 are two balancing hyperparameters to control the effects of $\hat{\theta}^t_i$ and $\hat{\theta}^1_i$, respectively set as $\Delta_1 = 1.00$ and $\Delta_2 = 0.01$. Minimizing this function we aim to push B^t_{pred} towards B^t_{gt} and away from B^1_{gt}, in order to improve the segmentation boundary precision, and avoid temporal copying among future frames masks.

3. Categorical Cross Entropy Loss $\mathcal{L}_{\text{ce}}$. To directly supervise the multiclass logits $z_{p,c}$ at each pixel p, we add

$$\mathcal{L}_{\text{ce}} = -\sum_{p} \sum_{c=1}^{C} \mathbf{1}_{\{y_p = c\}} \; \log\big(\hat{y}_{p,c}\big), \quad \hat{y}_{p,c} = \frac{\exp(z_{p,c})}{\sum_{k=1}^{C} \exp(z_{p,k})}.$$

This term encourages correct class predictions and complements the overlap-based and boundary-shaping losses.

Further details on the training hyperparameters (e.g. learning rate, batch size, video clip length) and the segmentation results will be presented in Chapter 5.

4.4 Evaluation Metrics

At inference time, for pixel-wise multiclass segmentation, with classes **background**, **solid**, and **non-solid**, the network outputs, for each pixel p, a probability vector $\hat{\mathbf{y}}_p = (\hat{y}_{p,1}, \hat{y}_{p,2}, \hat{y}_{p,3})$, which is converted to a hard prediction

$$\tilde{y}_p \;=\; \arg\max_{c\in\{1,2,3\}} \hat{y}_{p,c}\,.$$

Comparisons against the ground-truth mask $y_p \in \{1,2,3\}$ allow us to compute for each class c the entries of the **confusion matrix**:

$$\begin{aligned}
\mathrm{TP}_c &= \#\{\, p : (\tilde{y}_p = c) \wedge (y_p = c)\},\\
\mathrm{FP}_c &= \#\{\, p : (\tilde{y}_p = c) \wedge (y_p \neq c)\},\\
\mathrm{FN}_c &= \#\{\, p : (\tilde{y}_p \neq c) \wedge (y_p = c)\},\\
\mathrm{TN}_c &= \#\{\, p : (\tilde{y}_p \neq c) \wedge (y_p \neq c)\}.
\end{aligned}$$

Those variables hold the following meaning:

- **True Positives** TP_c: pixels predicted as class c and truly belonging to class c.
- **False Positives** FP_c: pixels predicted as class c but actually belonging to a different class.
- **False Negatives** FN_c: pixels of class c in the ground truth that were predicted as some other class.
- **True Negatives** TN_c: pixels neither predicted as class c nor actually class c.

These counts are essential for the computation of the evaluation metrics. By convention we compute each metric **per class** following the official implementation in [15].

Accuracy & Specificity

Accuracy measures the overall fraction of correctly classified pixels for class c (counting both correct positives and correct negatives). Formally:

$$\mathrm{Acc}_c = \frac{\mathrm{TP}_c + \mathrm{TN}_c}{\mathrm{TP}_c + \mathrm{TN}_c + \mathrm{FP}_c + \mathrm{FN}_c}\,.$$

Practically speaking, accuracy is the ratio of correct predictions over the total samples. While accuracy is an overall useful metric, it can be misleading when the Region Of Interest (ROI) is small compared to the background: for example, if 99% of pixels are **background**, a naive model that always predicts background would score 99% accuracy

despite missing all tumor regions, for this reason it's better to compute more comprehensive metrics (specificity, recall and precision) and segmentation specific metrics, like **IoU** and **Dice**.

Specificity measures the ability to correctly identify the negative class. For class c, it is:

$$\mathrm{Spec}_c = \frac{\mathrm{TN}_c}{\mathrm{TN}_c + \mathrm{FP}_c},$$

i.e. the fraction of non-c pixels correctly identified as not-c. In particular, high specificity means few false alarms (FP).

Precision & Recall

Precision for class c quantifies how many of the pixels predicted as c are actually c. It is defined as:

$$\mathrm{Prec}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c},$$

Thus, precision is the fraction of true positives among all predicted positives. A high precision means that when the model predicts class c, it is usually correct.

Recall for class c measures how well the model finds all the actual c pixels. It is defined as:

$$\mathrm{Rec}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c},$$

This is the fraction of true class-c pixels that were correctly predicted as c. High recall means few false negatives. Recall is critical in the task of tumor detection: missing a true tumor pixel (FN) can be more serious than a false alarm. So, in general, we focus on improving the **recall** of the model rather than its **precision**.

Jaccard Index

The **Jaccard index**, also known as **Intersection over Union (IoU)**, for class c measures the overlap between the predicted region and the ground-truth region of class c:

$$\mathrm{Jacc}_c = \frac{|\{\tilde{y} = c\} \cap \{y = c\}|}{|\{\tilde{y} = c\} \cup \{y = c\}|} = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c}.$$

Intuitively, **IoU** ranges from 0 (no overlap) to 1 (perfect overlap). It penalizes both false negatives and false positives, making it very suitable for segmentation tasks, and, in particular it's very popular in medical imaging. For example, if a tiny tumor is missed, IoU drops sharply.

Dice Coefficient

The Dice coefficient for class c is defined as:

$$\text{Dice}_c = \frac{2\,|\{\tilde{y} = c\} \cap \{y = c\}|}{|\{\tilde{y} = c\}| + |\{y = c\}|} = \frac{2\,\text{TP}_c}{2\,\text{TP}_c + \text{FP}_c + \text{FN}_c}\,.$$

Intuitively, **Dice** balances precision and recall: it is high only if both precision and recall are high. This makes it a good indicator of a model's positive-class performance.

Conclusive Remarks

In this chapter, we have explored the evolution of Video Object Segmentation, from early memory-bank methods like STM, through transformer-based architectures such as AOT and CUTIE. We then introduced two specialized medical VOS models, Med-SAM2 and ViViM, each designed to tackle the unique challenges of ovarian ultrasound segmentation, namely low contrast, rapid appearance changes, and the need for both long-range temporal coherence and precise boundaries. Med-SAM2 addresses these by incorporating a self-sorting memory bank containing the most informative past frames and predictions, while ViViM leverages state-space model structures and boundary-aware constraints to achieve efficient, accurate segmentation over extended sequences. Having established these foundations, the next chapter will focus on the results of those two methods, Med-SAM2 and ViViM, along with the results of the mask refinement approach using the SEA-RAFT tracking model, and the comparison with the benchmark method (U-Net).

Chapter 5

Experimental Results and Comparative Analysis

In this final chapter, we evaluate the segmentation performance of the three introduced methods: SEA-RAFT, Med-SAM2, and ViViM, compared to the baseline U-Net results, on the held-out test set introduced in Chapter 2. Our goals are to:

- Quantify pixel-level accuracy using standard segmentation metrics (IoU, Dice, precision, recall, specificity) for both the solid and non-solid classes;
- Visualize class-normalized confusion matrices to highlight per-class strengths and weaknesses;
- Compare inference runtimes (FPS, total and per-frame latency) to assess real-time deployability;
- Present qualitative segmentation examples to illustrate boundary accuracy and temporal consistency.

Particularly, we begin in the first section with the comparison between equispatial and dynamic sampling methods introduced with max numerosity as a preprocessing step to determine the effectiveness of each sampling method. Section 5.2 presents the evaluation results of the introduced VOS methods, SEA-RAFT as a post-processing mask refinement step and the benchmark U-Net method, providing empirical evidence that temporal-aware models provide better and more accurate segmentation masks. Lastly, in Section 5.3 we present our closing remarks and future direction ideas.

5.1 Training Configuration and Sampling Strategy Comparison

In this section we will present some further training details along with the choice of the models hyperparameters. In chapter 2 we have introduced various preprocessing

steps, ranging from the classical data augmentation techniques, to domain-specific ones (fan cropping and max numerosity). In particular, max numerosity is introduced with the scope of limiting each clinical case to $N_{\max}$ frames. In Chapter 2 we have compared two sampling schemes:

1. **Equispatially**: We divide the original video into $N_{\max}$ equal-length intervals and we select one representative frame for each intervals;

2. **Dynamically**: At every training epoch, randomly sample $N_{\max}$ frames from the full sequence.

As we show below, equispatial sampling yields more stable and higher segmentation accuracy in our ultrasound data.

Following the procedure of performing a five-folds cross-validation technique for optimal hyperparameter settings, in Figure 5.1 are shown the **Confusion Matrices** (row-normalized and column-normalized) of Med-SAM2 trained with equispatially sampled frames in comparison to the confusion matrix of Med-SAM2 trained with dynamically sampled frames. The results are presented for the first cross-validation training on the hold-out set used for validation. Med-SAM2 is trained for 100 epochs, with initial learning rate $\eta = 10^{-4}$, batch size of 16 frames, memory bank size of 16 frames and max numerosity $N_{\max} = 3$.

From Figure 5.1 we can observe that there is a performance drop in the segmentation metrics from equispatial to dynamic sampling, especially regarding solid class. This suggests that equispatial coverage of each lesion is better learned by the model rather than random frame selection. A similar trend holds for ViViM as well, as shown in Figure 5.2. Where ViViM was trained for 100 epochs, with initial learning rate $\eta = 10^{-4}$, batch size of 3 clips with each clip length of 5 (so effective batch size of 15 frames), and $N_{\max} = 3$.

From those results we adopt for both methods equispatial sampling as frame selection method for the final training.
Figure 5.3 plots the training loss over epochs:

- Med-SAM2 (left) converges smoothly to a stable minimum.
- ViViM (right) shows more oscillations, reflecting its complex composite loss.

Despite ViViM's oscillatory behavior, its segmentation performance (as we will show in the next section) exceeds Med-SAM2, demonstrating its robustness to complex temporal modeling. In the following section, we present our inference results of ViViM and Med-SAM2 against the U-Net and SEA-RAFT baselines.

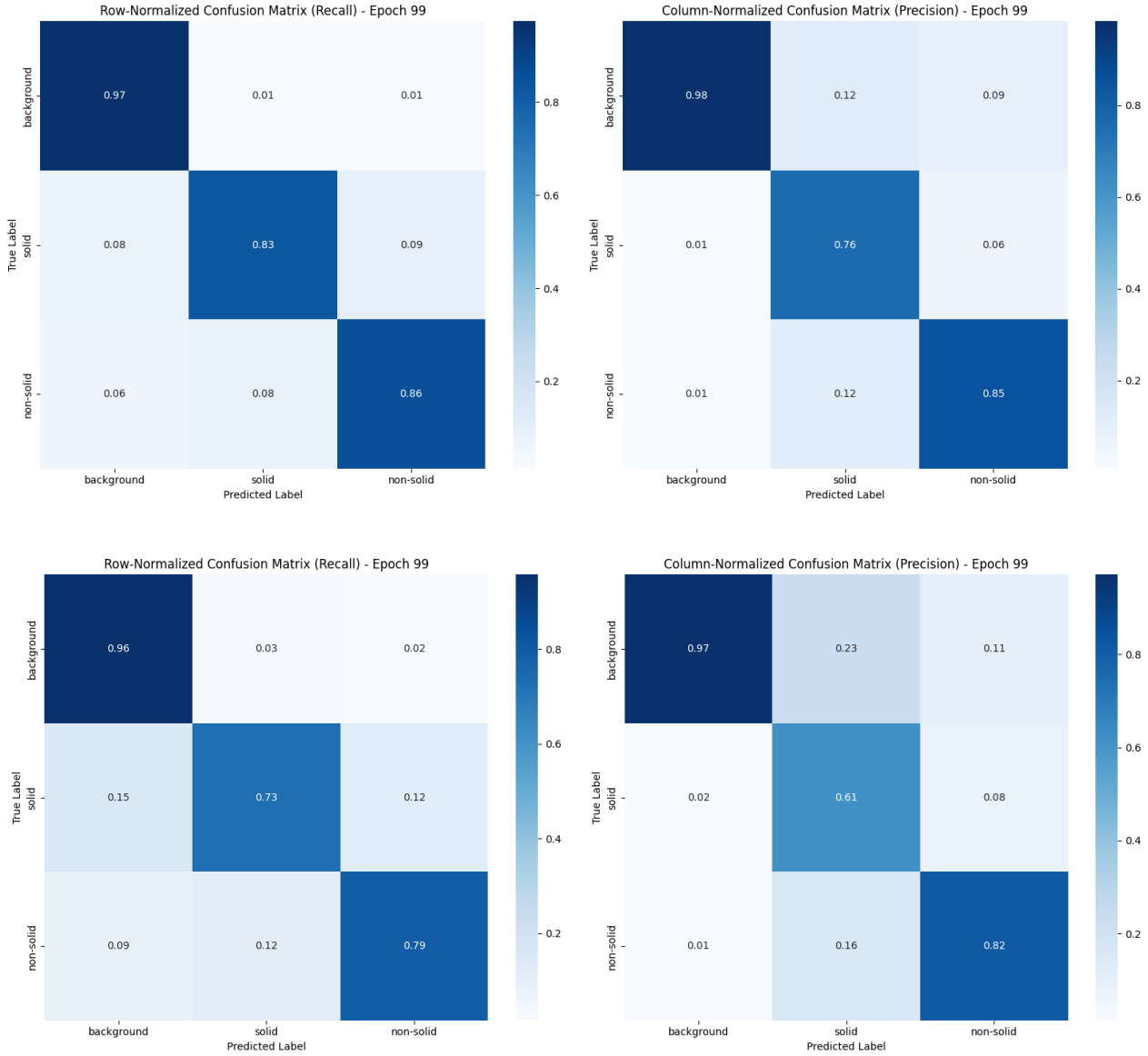


Figure 5.1: Comparison between the confusion matrices of Med-SAM2 with equispatial sampling (on top) and dynamic sampling (bottom). The confusion matrices represents respectively the recall per class and the precision per class.

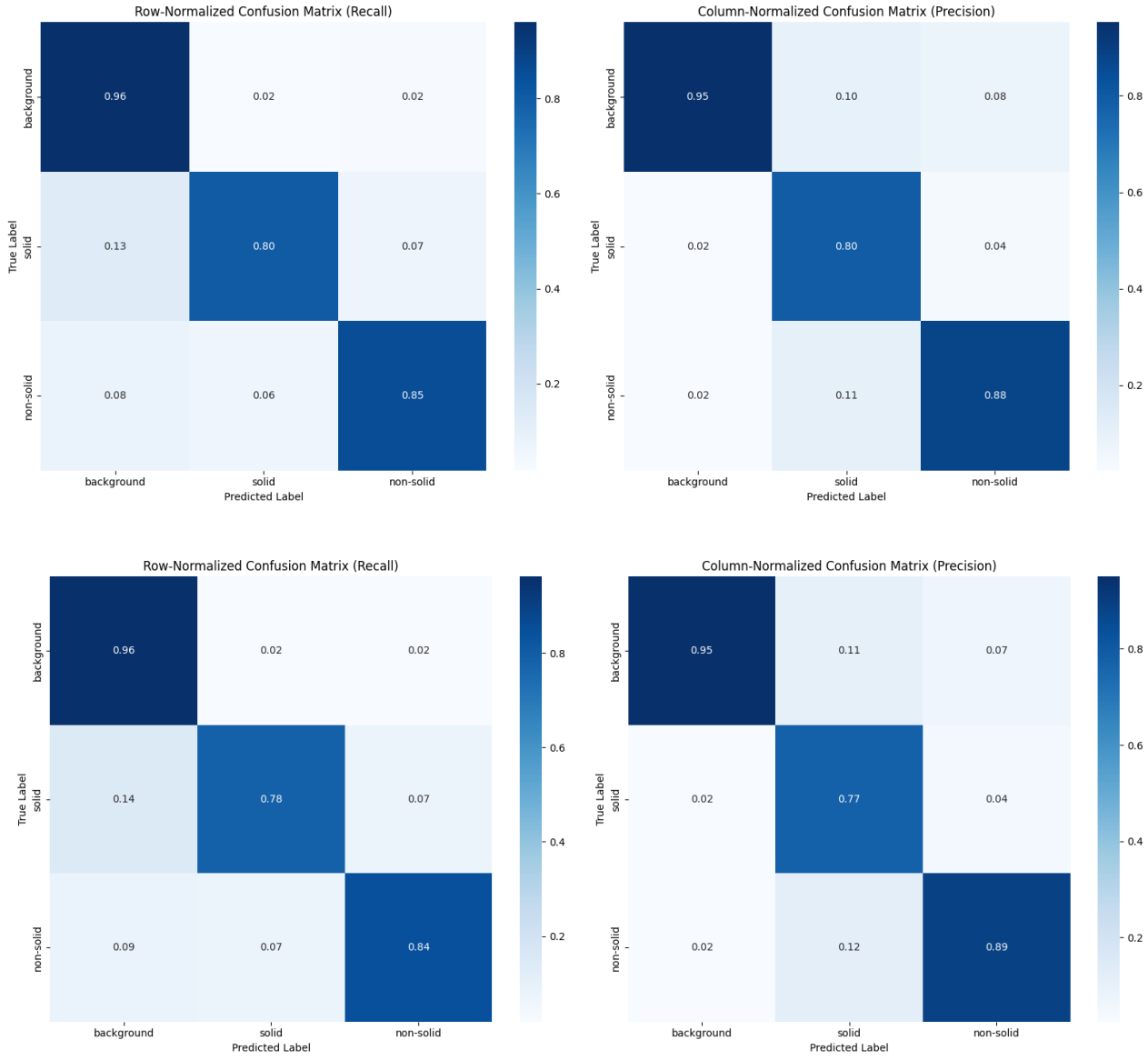


Figure 5.2: Comparison between the confusion matrices of ViViM with equispatial sampling (on top) and dynamic sampling (bottom).

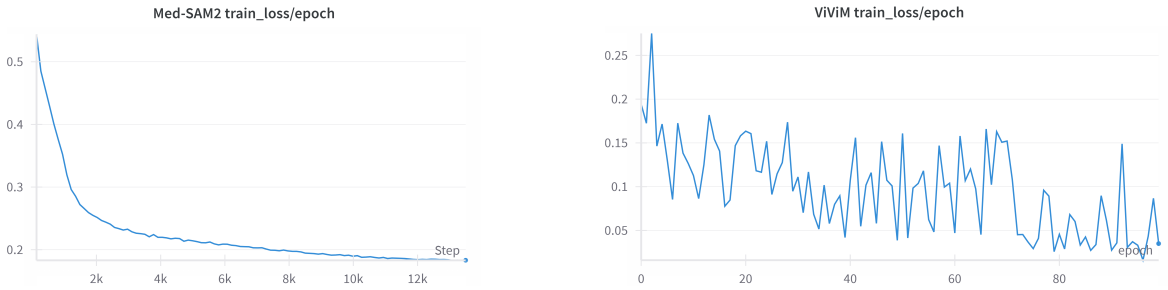


Figure 5.3: Comparison between Med-SAM2 and ViViM loss during the final training phase.

5.2 Comparative Performance Evaluation

In this section we will present the evaluation results on the test dataset of the three proposed methods, SEA-RAFT, Med-SAM2 and ViViM, for ultrasound video object segmentation in comparison with the baseline method (U-Net) to determine how temporal modeling impacts segmentation accuracy. We begin with the class-normalized confusion matrices for each method, then we will present a detailed comparison of the standard segmentation metrics.

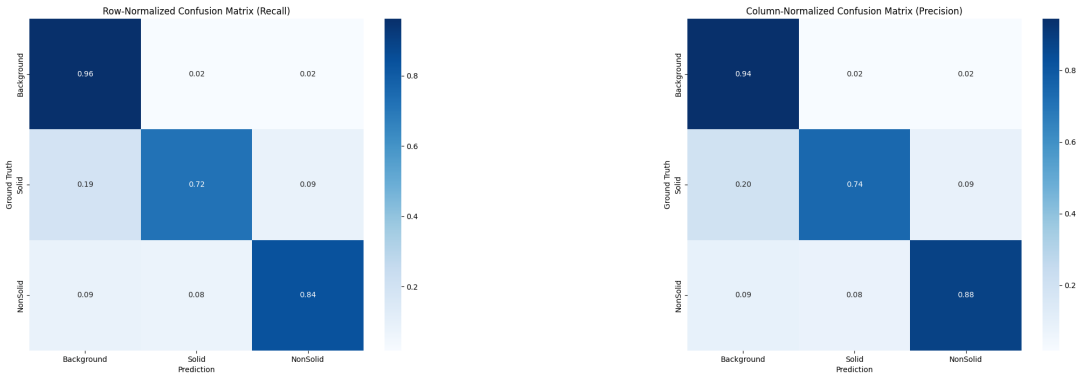


Figure 5.4: Confusion matrix normalized per row (left) and per column (right) of U-Net.

From the confusion matrices we can derived the following:

- **Prompt Mismatch in Med-SAM2**: Med-SAM2's performance drops from validation to inference (compare Figures 5.1 and 5.6) due to the change in prompting strategy: during training we used positive/negative prompts derived from ground-truth masks, whereas at inference (where no ground-truth is available) we supply

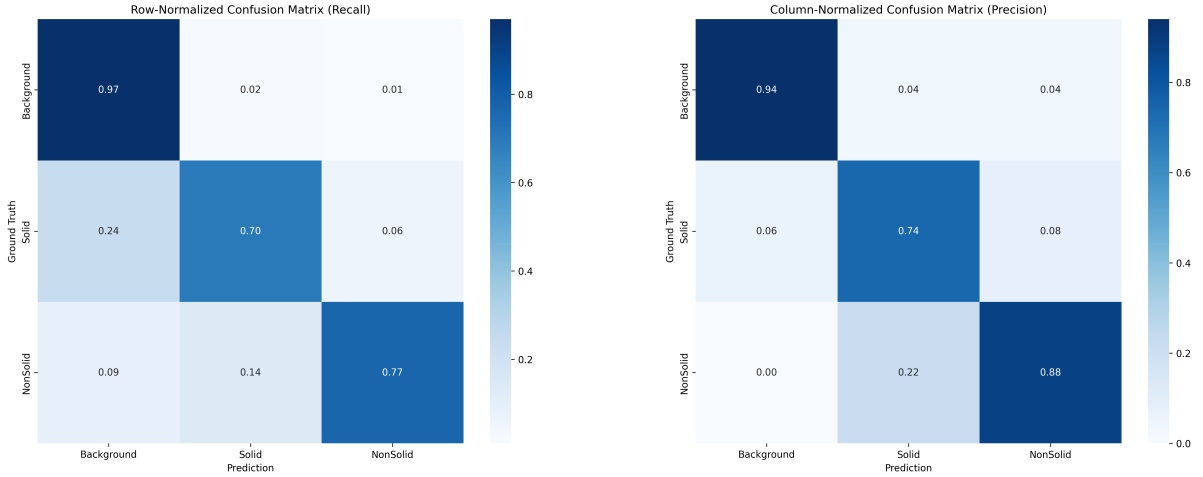


Figure 5.5: Confusion matrix normalized per row (left) and per column (right) of SEA-RAFT.

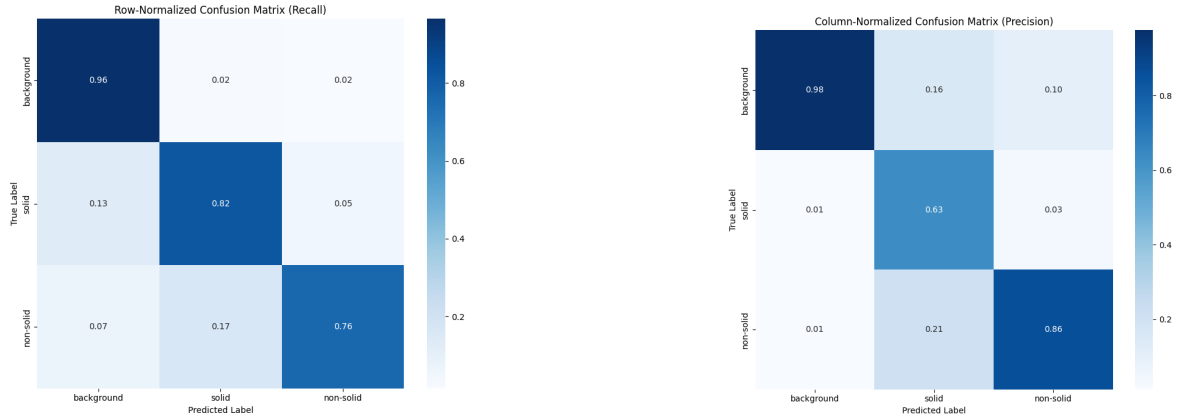


Figure 5.6: Confusion matrix normalized per row (left) and per column (right) of MedSAM2.

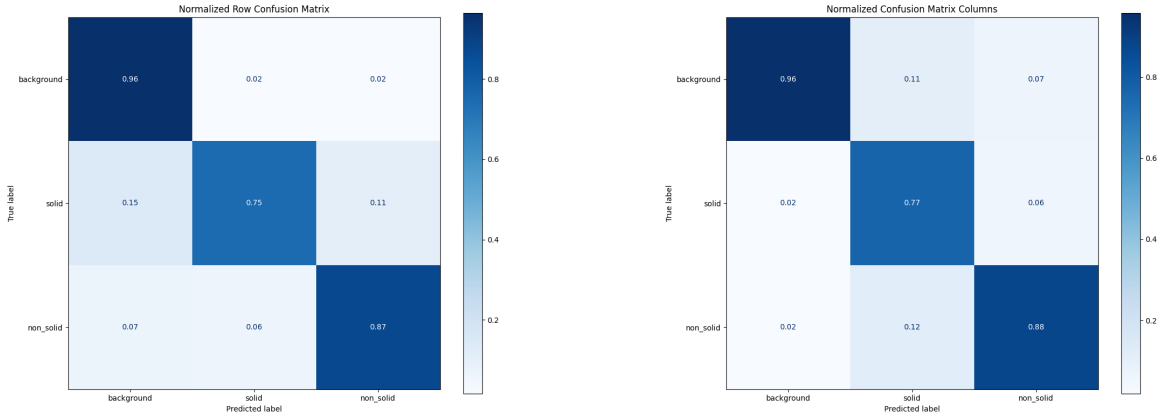


Figure 5.7: Confusion matrix normalized per row (left) and per column (right) of ViViM.

the model only a full-frame bounding box as prompt, in order to still have a fully automated model with no need of human interactions. This mismatch inevitably reduces lesion localization accuracy, yet Med-SAM2 still achieves higher recall for the solid class compared to U-Net, which is a good result given our goal of minimizing false negatives.

- **SEA-RAFT Performance**: SEA-RAFT does not outperform U-Net, having comparable to slightly worse performance, likely because it operates purely as a post-processing step on the U-Net outputs and has not been fine-tuned on ultrasound data. Any segmentation errors missed by U-Net cannot be corrected by flow-based warping alone.

- **ViViM Performance**: ViViM delivers the strongest balance of precision and recall across all classes, outperforming both the baseline and other temporal methods. Its state-space-based sequence modeling and composite loss function effectively capture long-range dependencies and produce accurate, boundary-aware segmentation masks.

Detailed Segmentation Metrics

Tables 5.1 and 5.2 summarize respectively for the class of solid and non-solid specificity, precision, recall, Jaccard (or IoU) and Dice metrics. Overall, temporal models, particularly ViViM, demonstrate some advantages over frame-only segmentation, validating the inclusion of temporal complexity in ultrasound video analysis. Moreover, we can observe that the U-Net still remains a solid choice for segmentation, as it is arguably the most consistent method across the two classes, while both ViViM and Med-SAM2 present a more evident gap between solid and non-solid class. Med-SAM2 specifically

outperforms all the other methods in the solid class, but has by far the worse performance in the non-solid class.

Model	Spec.	Prec.	Rec.	Jacc.	Dice
U-Net	0.968	0.741	0.723	0.493	0.578
SEA-RAFT	0.968	0.742	0.709	0.432	0.519
Med-SAM2	0.941	0.632	**0.823**	**0.512**	**0.636**
ViViM	**0.971**	**0.771**	0.748	0.453	0.532

Table 5.1: Segmentation metrics for the solid class. Bold indicates top performance.

Model	Spec.	Prec.	Rec.	Jacc.	Dice
U-Net	**0.971**	0.877	0.844	0.660	0.746
SEA-RAFT	0.968	0.880	0.772	0.639	0.730
Med-SAM2	0.976	0.855	0.762	0.602	0.710
ViViM	0.969	**0.882**	**0.874**	**0.688**	**0.770**

Table 5.2: Segmentation metrics for the non-solid class. Bold indicates top performance.

Table 5.3 confirms that ViViM's state-space architecture runs at over 140 FPS, significantly faster than Med-SAM2 (34 FPS) and even U-Net ($\approx$ 100 FPS), making it highly suitable for real-time clinical deployment, as we proved in the last chapter.

Model	FPS	Total Inference Time (s)	Avg Time per Frame (ms)
Med-SAM2	34.09	149.48	29.33
ViViM	**140.17**	**33.39**	**7.13**

Table 5.3: Inference speed comparison of Med-SAM2 and ViViM on test set.

Segmentation Visualization

Figure 5.8 shows example frames with ground-truth masks alongside predictions from U-Net, SEA-RAFT, Med-SAM2, and ViViM. Several observations stand out:

- **Med-SAM2 Boundaries**: The transformer-based Med-SAM2 often produces blurred and inaccurate edges, a known weakness of vision transformers on hard boundary tasks [36].

- **ViViM Precision**: Thanks to its state-space architecture and boundary-aware affine loss, ViViM delivers the sharpest, most accurate contours, closely matching the ground truth even in low-contrast regions.
- **U-Net vs. SEA-RAFT**: Both U-Net and SEA-RAFT generate generally clean masks but struggle with occlusions and texture variations. SEA-RAFT, as a post-processor, rarely improves upon U-Net and sometimes worsen the segmentation mask.

Overall, ViViM and U-Net produce the most visually coherent segmentations, with ViViM offering superior edge detail and temporal stability. SEA-RAFT's gains are marginal, while Med-SAM2 remains challenged by boundary accuracy.

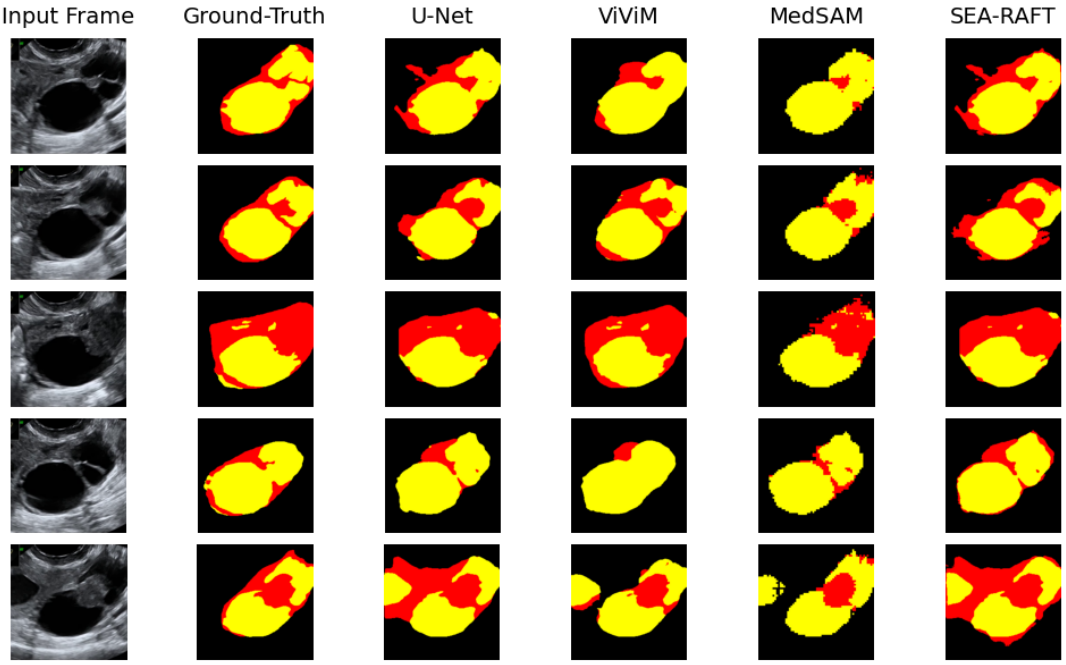


Figure 5.8: Qualitative comparison of segmentation results (red: solid class; yellow: non-solid class). From left to right: Input frame, Ground Truth, U-Net, ViViM, Med-SAM2, SEA-RAFT.

5.3 Closing Remarks and Future Directions

In this chapter, we have compared the four methods introduced throughout the thesis for the task of segmenting ovarian ultrasound videos. We close this chapter by providing some final thoughts on the models explored:

- **U-Net**: Our baseline was a U-Net, which, despite its architectural simplicity compared to the other introduced models, provides accurate and consistent segmentation masks. However, those masks can be prone to flickering (i.e. rapid changes of object's appearence) and temporal inconsistency, from which we derive the need for introducing temporal-aware models.
- **SEA-RAFT**: SEA-RAFT is explored as a post-processing technique to add temporal smoothing via optical flow estimation. However, due to the lack of pre-training on medical datasets and its reliance on U-Net's segmentation masks, it offers only marginal gains.
- **Med-SAM2**: Med-SAM2 scores well on solid lesions, which its self-sorting memory bank could be a factor in. However, the problematic boundary accuracy and the prompt mismatch from training to inference remain challenges.
- **ViViM**: ViViM delivers the best balance of precision, recall, and inference speed by means of a state-space model architecture (Mamba blocks) and composite loss that provides sharp boundary segmentation.

Overall, ViViM emerges as the most promising approach, validating that efficient state-space temporal modeling, combined with a boundary-aware loss, can outperform both the baseline method and transformer-based memory models in this domain.

Future Research Directions

Based on our findings, several promising ideas for future research emerge that could further advance video object segmentation in medical ultrasound applications:

- **Medical-Domain Optical Flow Datasets**: Current flow estimators (e.g. SEA-RAFT) are pretrained on natural-image benchmarks, which differ significantly from ultrasound videos. Constructing a dedicated ultrasound video flow dataset would enable pretraining or fine-tuning of flow networks on domain-relevant motion patterns, leading to more accurate mask warping and stronger post-processing results.
- **Adaptive Prompting for Med-SAM2**: Med-SAM2's inference prompts currently consist of a full-frame bounding box. Integrating an auxiliary detection model (e.g. YOLO) could generate more precise box or click prompts automatically. Such prompts would ensure consistency between training and inference prompting strategies, improving Med-SAM2's segmentation accuracy and consistency.
- **Composite Loss for Med-SAM2**: Exploiting a composite loss similar to ViViM could yield sharper boundaries and improved segmentation quality for the Med-SAM2 framework.

- **ViViM Hyperparameter Optimization**: Since ViViM is the best-performing model, careful hyperparameter tuning could further boost its performance and establish more robust optimization protocols for clinical deployment.

By pursuing these directions (domain-specific pretraining, adaptive prompting, structured losses, and hyperparameters tuning) we aim to develop VOS models that combine clinical reliability with real-time performance in ultrasound applications.

Conclusion

This thesis has explored the integration of temporal context into ovarian ultrasound segmentation through three different approaches: post-processing tracking, end-to-end transformer-based model and end-to-end state-space model. After giving some preliminary knowledge in Chapter 1, we began in Chapter 2 by analyzing our private ovarian ultrasound video collection of more than 10 000 annotated frames across 33 histological types. After quantifying class imbalances (solid, non-solid, background) and histological prevalence, we introduced stratified 5-fold cross-validation at the video level to ensure robust performance estimates during training. Multiple classical augmentation techniques (e.g. cropping, flippings, rotations) and domain-specific techniques (fan cropping and max numerosity frame sampling) are explored to mitigate overfitting and class imbalance. In Chapter 3 we explored Tracking as a post-processing mask refinement technique. Firstly, we surveyed three tracking paradigms: detection-based (with the SORT model), optical flow estimation (FlowNet) and spatio-temporal graphical models (CRFs), and identified RAFT as a state-of-the-art flow estimator framework. Its successor SEA-RAFT brings a Mixture-of-Laplace loss, direct initial regression, and rigid-flow pretraining for faster convergence and greater robustness. We then detailed our mask-refinement pipeline: warping U-Net outputs via SEA-RAFT flow estimate and fusing them with current predictions. While this reduced flicker, it could not recover missed regions and yielded only marginal gains as the produced segmentation masks are highly dependent on the U-Net ones. Chapter 4 was dedicated to the introduction of end-to-end VOS models. We reviewed three general-purpose state-of-the-art architectures: STM (space-time memory), AOT (identity-aware transformers), and CUTIE (object-level transformers), highlighting their memory mechanisms, attention layers, and computational trade-offs. Building on these, we introduced two medical-domain methods:

- **Med-SAM2** augments the promptable Segment Anything Model 2 with a self-sorting memory bank tailored to ultrasound's non-stationary anatomy and employs click-based prompts during training. At inference, full-frame box prompts yield strong solid-class results but expose boundary and prompt-mismatch weaknesses.

- **ViViM** embeds selective State-Space Mamba blocks for efficient long-range dependency, plus a boundary-aware affine loss, achieving the best balance of IoU, Dice,

precision, recall, and real-time speed (> 140 FPS).

Finally, in Chapter 5 we presented our experimental results, where we demonstrated that:

- SEA-RAFT produces smooths masks but fails to correct U-Net's misses.
- Med-SAM2 excels in solid-class recall yet suffers boundary blur under the current prompts and loss function.
- ViViM consistently outperforms all methods in both accuracy and runtime, validating state-space modeling and structured losses for clinical ultrasound VOS.

Lastly, to further improve our model's segmentation results, we proposed the following directions:

1. **Domain-Specific Flow Pretraining**: Build an ultrasound optical-flow dataset to fine-tune SEA-RAFT for more robust segmentation outputs.
2. **Adaptive Prompt Generation**: Integrate automatic lesion detectors to generate more precise Med-SAM2 prompts at inference.
3. **Structured Loss Extensions**: Apply boundary-aware loss to Med-SAM2 to sharpen the final masks.
4. **Hyperparameters Tuning**: Correctly tune ViViM's hyperparameters in order to further boost its performance.

This thesis contributes to the advancement of video object segmentation in medical ultrasound by demonstrating that state-space models, when combined with appropriate loss functions and extensive preprocessing, can achieve superior performance over traditional frame-by-frame approaches and transformer-based methods. The comprehensive evaluation of temporal modeling strategies, from post-processing tracking to end-to-end architectures, provides valuable insights for the development of clinically accurate ultrasound analysis systems. Our findings establish ViViM as a promising foundation for real-time, temporally coherent ovarian ultrasound segmentation, while identifying possibilities for further improvement across all evaluated approaches. The demonstrated ability to achieve both high accuracy and real-time performance (> 140 FPS) positions ViViM as a viable candidate for integration into clinical ultrasound workflows, potentially enhancing the consistency and efficiency of ovarian lesion segmentation and providing a reliable automated analysis tool that can support medical professionals in diagnosing tumors.

Bibliography

[1] Fishernova, D. (2011). Ultrasound scanning of the pelvis and abdomen for staging of gynecological tumors: a review. *Ultrasound in Obstetrics & Gynecology*, vol 38, no.3, pp. 246-266.

[2] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7), 3523-3542.

[3] Yao, R., Lin, G., Xia, S., Zhao, J., & Zhou, Y. (2020). Video Object Segmentation and Tracking: A Survey. *ACM Computing Surveys*, vol. 53, no. 4, pp. 1-36.

[4] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, vol. 57, article 99.

[5] Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). *arXiv preprint arXiv:1803.08375*.

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998-6008.

[7] Schmidt, R. M. (2019). Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. *arXiv preprint arXiv:1912.05911*.

[8] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*.

[9] Sayasneh, A., Ekechi, C., Ferrara, L., Kaijser, J., Stalder, C., Sur, S., Timmerman, D., & Bourne, T. (2015). The characteristic ultrasound features of specific types of ovarian pathology (review). *International Journal of Oncology*, *46*(2), 445-458.

[10] Timmerman, D., Schwarzler, P., Collins, W. P., Claerhout, F., Coenen, M., Amant, F., Vergote, I., Bourne, T. H., & Van Huffel, S. (2000). Simple ultrasound-based rules for the diagnosis of ovarian cancer. *Ultrasound in Obstetrics and Gynecology*, *16*(5), 500-507.

[11] Timmerman, D., Testa, A. C., Bourne, T., Ameye, L., Jurkovic, D., Van Holsbeke, C., Paladini, D., Van Calster, B., Vergote, I., Van Huffel, S., & Valentin, L. (2008). Simple ultrasound-based rules for the diagnosis of ovarian cancer. *Ultrasound in Obstetrics & Gynecology*, *31*(6), 681-690.

[12] IOTA Plus Foundation. Educational Materials. *IOTA Plus*, Available at: https://iotaplus.org/en/education/educational-materials .

[13] OpenCV. (2025). *The OpenCV Reference Manual* (4.x ed.). `https://docs.opencv.org/4.x/`

[14] Google Developers. *Classification: Accuracy, recall, precision, and related metrics*. Machine Learning Crash Course. `https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall`

[15] Lightning AI . *TorchMetrics*. `https://lightning.ai/docs/torchmetrics/stable/`

[16] O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[17] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking. *2016 IEEE International Conference on Image Processing (ICIP)* (pp. 3464-3468).

[18] Kalman R. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35-45.

[19] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.

[20] Kuhn H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, vol. 2, pp. 83-97.

[21] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., & Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. *Proceedings of the IEEE international conference on computer vision* pp. 2758-2766.

[22] Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the eighteenth international conference on machine learning*, pp. 282-289.

[23] Sutton, C., & McCallum, A. (2012). An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, vol. 4(4), pp. 267-373.

[24] Krähenbühl, P., & Koltun, V. (2011). Efficient inference in fully connected CRFs with Gaussian edge potentials. *Advances in Neural Information Processing Systems*, vol. 24, pp. 109-117.

[25] Teed, Z., & Deng, J. (2020). RAFT: Recurrent all-pairs field transforms for optical flow. *European Conference on Computer Vision*, pp. 402-419.

[26] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2462-2470.

[27] Wang, Y., Lipson, L., & Deng, J. (2024). SEA-RAFT: Simple, Efficient, Accurate RAFT for Optical Flow. *European Conference on Computer Vision*, pp. 39-56.

[28] Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., Scherer, S. (2020). Tartanair: A dataset to push the limits of visual slam. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4909-4916.

[29] S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim (2019) Video Object Segmentation using Space-Time Memory Networks. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9226-9235.

[30] Yang, Z., Wei, Y., & Yang, Y. (2021). Associating Objects with Transformers for Video Object Segmentation. *Advances in Neural Information Processing Systems*, vol. 34, pp. 2491-2502.

[31] Cheng, H. K., Oh, S. W., Price, B., Lee, J. Y., & Schwing, A. (2024). Putting the Object Back into Video Object Segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3151-3161.

[32] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W. Y., Dollár, P., & Girshick, R. (2023). Segment Anything. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015-4026.

[33] Ravi, N., Gabeur, V., Hu, Y., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C., Girshick, R., Dollár, P., & Kirillov, A. (2024). SAM 2: Segment Anything in Images and Videos. *Advances in Neural Information Processing Systems*, vol. 37.

[34] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[35] Zhu, J., Hamdi, A., Qi, Y., Jin, Y., & Wu, J. (2024). Medical SAM 2: Segment medical images as video via Segment Anything Model 2. *arXiv preprint arXiv:2408.00874.*

[36] Bai, X., Yu, Y., et al. (2024). FS-MedSAM2: Exploring the Potential of SAM2 for Few-Shot Medical Image Segmentation without Fine-tuning. *arXiv preprint arXiv:2409.04298.*

[37] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2999-3007.

[38] Salehi, S. S. M., Erdogmus, D., & Gholipour, A. (2017). Tversky loss function for image segmentation using 3D fully convolutional deep networks. *International Workshop on Machine Learning in Medical Imaging*, pp. 379-387.

[39] Yang, Y., Xing, Z., & Zhu, L. (2024). Vivim: a Video Vision Mamba for Medical Video Object Segmentation. *IEEE Transactions on Circuits and Systems for Video Technology.*

[40] Gu A. and Dao T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.

[41] Wang, R., Lei, T., Cui, R., Zhang, B., Meng, H., & Nandi, A. K. (2022). Medical image segmentation using deep learning: A survey. *IET Image Processing*, vol. 16, no. 5, pp. 1243-1267.