ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Weakly Supervised Video Object Segmentation

Master's Thesis

Filip Skogh
Department of Information Technology and Electrical Engineering

Advisors: Dr. Martin Danelljan
Dr. Lei Ke
Supervisor: Prof. Dr. Fisher Yu

October 9, 2024

Abstract

Video object segmentation is a fundamental problem in computer vision used in a variety of application across many fields. Over the past few years video object segmentation has witnessed rapid progress catalyzed by increasingly large datasets. These datasets consisting of pixel-accurate masks with object association between frames are especially labor-intensive and costly, prohibiting truly large-scale datasets. We propose a video object segmentation model capable of being trained exclusively with bounding boxes, a cheaper type of annotation. To achieve this, our method employs loss functions tailored for box-annotations that leverages self-supervision through color similarity and spatio-temporal coherence.

We validate our approach against traditional fully-supervised methods and various other settings on YouTube-VOS and DAVIS, achieving over 90% relative performance on $\mathcal{J}\&\mathcal{F}$ in comparison to fully-supervised models in the box-initialization setting, while scoring around 85% in the mask-initialization setting. We also investigate practical aspects of our model, achieving a relative performance of 87% on longer term videos with 1000s of frames. We also perform ablations both quantitatively and qualitatively and show visually how the loss function improves fine-detail along with failure cases. Moreover, our method is practical with over 22 frames per second on the YouTubeVOS validation set.

"Implementation errors are regularizers."

Acknowledgements

First and foremost, heartfelt gratitude goes to Dr. Martin Danelljan and Prof. Dr. Fisher Yu for giving me this opportunity and making this project possible. Beyond that, your expertise in computer vision has inspired both me, and my models, to learn a lot. I also want to thank Dr. Lei Ke for continuous support and the fruitful discussions during this period. Your presence have been crucial in every step of this project.

Additionally, I am grateful for the exceptional individuals who have led me to this point. These last five years have been wonderful, and as I look back on this chain of rather unlikely events filled with serendipity, I want to thank each one of you.

Contents

1 Introduction **1**
- 1.1 Focus of this Work . . . 1
- 1.2 Contributions . . . 2
- 1.3 Thesis Organization . . . 2

2 Background and Related Work **3**
- 2.1 Video object segmentation . . . 3
- 2.2 Weakly-Supervised Video Object Segmentation . . . 5
- 2.3 Unsupervised Video Object Segmentation . . . 6

3 Method **7**
- 3.1 XMem . . . 7
  - 3.1.1 Loss . . . 8
- 3.2 Spatial Consistency . . . 9
  - 3.2.1 Global consistency . . . 9
  - 3.2.2 Local consistency . . . 11
- 3.3 Temporal Consistency . . . 13
  - 3.3.1 Detach Regularization . . . 14

4 Experiments **17**
- 4.1 Datasets . . . 17
- 4.2 Metrics . . . 18
- 4.3 Implementation Details . . . 18
- 4.4 Results . . . 19
  - 4.4.1 Quantitative Results . . . 19
  - 4.4.2 Ablation study . . . 21
  - 4.4.3 Qualitative Results . . . 22

5 Discussion and Conclusion **25**
- 5.1 Performance . . . 25
- 5.2 Limitations . . . 26
- 5.3 Conclusion . . . 27

A The First Appendix **29**

List of Figures

2.1 Different video object segmentation settings showing varying levels of supervision given in the first frame during test time. Naturally each setting comes with models of varying accuracy, robustness and speed. 4

3.1 Figure sourced from XMem [7], illustrating the segmentation pipeline for a single $H_0 \times W_0$ RGB frame. Specifically, the query encoder extracts strided features $\boldsymbol{q}$ from the input frame. The query is then used to match previously extracted memory keys by constructing an affinity matrix $\boldsymbol{W}(\boldsymbol{k}, \boldsymbol{q})$. Using the affinities, the memory values are then read resulting in features $\boldsymbol{F}$. Along with the sensory memory $\boldsymbol{h}_{t-1}$ and frame features, the decoder then produces the predicted mask. Finally, the frame and mask are passed to the value encoder to extract new memory features, used in future timesteps. 8

3.2 Illustration of the projections used to compute the projection loss. Given an image with predicted mask $\hat{m} \in [0,1]^{H \times W}$ shown in blue, and the ground-truth bounding box $m \in \{0,1\}^{H \times W}$ shown in red, the masks are projected in to one-dimensional vectors in the vertical and horizontal dimension. The loss is then calculated using the projected vectors along with the dice loss. 10

3.3 Example of two potential ways to calculate the loss for $B = 2$, $N_o = 1$ and $T = 3$. On the left each $H \times W$ image is calculated independently and then aggregated over the $B \times N_o \times T$ dimensions. On the right the collection of images are treated as one single image of size $BH \times N_o TW$, with mask $\hat{m} \in [0,1]^{BH \times N_o TW}$ concatenated of $\hat{m}_{bnt}$ for $bnt \in [B] \times [N_o] \times [T]$ Note that both methods are permutation invariant with respect to dimensions $B \times N_o \times T$. 11

3.4 Example of an occlusion creating two disconnected masks which stays disconnected after projection onto the x-axis, since the split is aligned with the y-axis. This causes false positives in the x-axis projection. 11

3.5 Illustrating how the center pixel shown in red, can be used to create supervision from proximal pixels shown in green and blue. The green pixels satisfy the color similarity threshold with the center pixel, while the blue pixels do not satisfy the threshold. Shown here is a 3×3 kernel with dilation 2. In principle the pattern is arbitrary as long as the spatial locality assumption is valid. Importantly, the pairwise loss is completely agnostic to the red unmatched pixels. 12

3.6 Step i) match a patch in frame t with patches inside a region of interest in frame t'. Patches are colored green when the color similarity between patches are high enough, otherwise they are colored blue. Step ii) for each match we add the loss that the center pixel of patch should be consistent with all pixels in the other patch. In this case we use patch size $N = 3$ and search size $M = 5$ with disjoint search. If we did an exhaustive search we could have matched more, but it is a computational trade-off. 14

3.7 Examples of the degenerate minima where the predicted masks $\hat{m}$ have small area such that the satisfy the projection loss while having a low pairwise and temporal loss. We device a regularization technique to prevent this. 14

4.1 Histogram showing the number of frames per video in the YouTubeVOS training dataset. . . 18
4.2 Five randomly sampled frames from the DAVIS 2017 category `camel` showing single-object segmentation with different losses. The first row uses only projection loss, second row uses projection and pairwise loss, third row uses projection and temporal loss, and the last row uses all losses. 22
4.3 Five randomly sampled frames from the DAVIS 2017 category `horsejump-high` containing two objects. The first row uses only projection loss, second row uses projection and pairwise loss, third row uses projection and temporal loss, and the last row uses all losses. . . 23
4.4 **Failure cases:** illustrating instance mix-up and holes in segmentation. From left: similar instances close to each other. Middle: sudden light shift from leaving the shadow. Right: rapid motion of the flag along with motion blur. Examples are taken from YouTubeVOS and DAVIS. 23

A.1 Examples of erroneous labels in the YouTubeVOS dataset. The left column shows correctly filled masks while the middle and right columns show the examples of when the interior is missing. 29
A.2 Histogram of the fraction of bounding box covered by the precise mask. Evaluated on 1000 random videos from the YouTubeVOS training set. 29

List of Tables

3.1 We measure how accurate the temporal consistency assumption is on the training splits of YouTubeVOS-2018, DAVIS-2017 and MOSE. We sample images with pre-processing identically to as done during training. A true positive (TP) is defined as when the color threshold is satisfied and the labels of the two pixels match, while a false positive (FP) is when the color threshold is satisfied but the labels differ. Precision is calculated as $TP/(TP + FP)$. Supervision proportion is defined as the ratio of number of pixels that satisfies the color threshold, to the total number of pixels in the batch $B \times T \times N_o \times H \times W$. Both metrics are then averaged over all samples in the dataset. 15

4.1 State-of-the-art comparison on DAVIS and YouTubeVOS. M and B indicate whether the method is mask initialized or box initialized respectively during *test time*. Methods that use video masks during *training* is indicated in the "Uses video masks" column, note that for unsupervised methods, only raw videos were used, while our method trains on video bounding boxes. * indicates that image masks were used during training. 20

4.2 Comparison of our weakly-supervised method with fully-supervised state-of-the-art methods on long-term videos [20]. Values for the previous methods are sourced from [7]. * indicates that image masks were used during training. 20

4.3 Results on DAVIS when training using bounding boxes. All four cases with different combinations of projection, pairwise, and temporal loss use the same hyper-parameters. The maximum values are seen in bold, and the second largest are underlined. 21

4.4 Results on YouTubeVOS with mask-initialization when training using bounding boxes only of different combinations of projection, pairwise, and temporal loss. 21

4.5 Results when training using bounding boxes and first frame exact mask of different combinations of projection, pairwise, and temporal loss. Here we allow the first frame mask to be any frame in each video. 21

4.6 Results when training with first frame mask initialization and bounding boxes. All four cases with different combinations of projection, pairwise, and temporal loss use the same hyper-parameters. We allow the first frame mask to be any frame. 22

Chapter 1

Introduction

Video object segmentation is a fundamental task in computer vision, enabling machines to perceive and understand the dynamic world. As for humans, accurate and robust video object segmentation plays a pivotal role in a wide range of fields serving as a basis for applications such as dexterous grasping, autonomous agents, manufacturing, augmented reality and video editing.

Traditional video object segmentation methods have primarily relied on supervised learning approaches [51, 52, 7, 8, 21, 50, 49, 28], where large-scale annotated datasets are required for training. Although such methods have demonstrated impressive results on existing benchmarks [30, 31, 46, 48], there is still a lot of improvement that can be done in more complex scenes [10] and generalization to new environments and unseen objects. Furthermore, this insatiable hunger for annotated data poses a bottleneck as manual annotation efforts get harder and harder to keep up due to cost and time constraints [23, 46]. To alleviate this problem, methods using cheaper forms of annotations have great potential. This is especially relevant in the field of video object segmenting since the construction of pixel-accurate segmentation annotations for video sequences is especially expensive.

In this weakly supervised setting, the use of bounding boxes has gained prominence as a mean to provide partial supervision [34, 15]. Unlike pixel-level accurate masks, bounding boxes are more cost-effective to annotate but offer a weaker form of supervision, indicating the rough region of interest for the objects in each frame. The primary challenge in weakly supervised video object segmentation lies in devising an effective mechanisms to leverage the inherent temporal coherence and object motion cues present in videos along with spatial priors available in individual frames. Many such mechanisms have been explored [34, 15, 14, 19, 38, 18], employing different kinds of methods such as contrastive learning, temporal color coherency, cyclic-patch consistency, correspondence learning and combinations of them.

1.1 Focus of this Work

We explore the potential of video object segmentation with less of the supervision burden. Specifically, we focus on combining a state-of-the-art video segmentation model with a box-supervised loss aided by self-supervision. We believe this approach is beneficial for few reasons. Firstly, the box-annotation setting is a good trade-off between fully-supervised and unsupervised methods as previous unsupervised models are often too slow to be of practical use. They also use substantially different architectures making it hard to separate the impact of the supervision and the model itself, also causing difficulties when comparing them to other unsupervised models. In our case we have a powerful baseline with the same architecture that allows us to isolate the impact of the weakly and and self-supervised losses. Additionally, we can also be confident

in that the model is not causing bottlenecks. Moreover the methods is adaptable since the modularity allows any loss based model to implement it. Furthermore, it is flexible since it can also be combined with other losses. Its scalability is also important since the complexity is independent of model architecture, and the training times are just slightly impacted from this loss versus common losses such as dice and cross entropy.

The setting in which training involves only bounding boxes and test utilizes a complete first frame mask, is logical as it grants more control. In situations where a bounding box alone lacks specificity, and decisions regarding object selection within the box are uncertain, this approach becomes valuable. Moreover, instances might arise where the tracked object is a composition of multiple entities or just half of an object, further complicating the bounding box representation.

1.2 Contributions

This work introduces a video object segmentation model that can be trained in a mask-free setting, relying solely on bounding boxes. Using this model, we then explore the how methods with different supervision settings impact performance. More specifically, we assess the performance gap between our method and box-initialized, annotation-free, and fully-supervised methods. Our contributions are summarized as follows:

- integrate a mask-free spatio-temporal loss in to a video object segmentation setting,
- devise a regularization technique to prevent degenerate solutions, and
- investigate the performance gap between different unsupervised, weakly-supervised and fully-supervised video object segmentation methods.

1.3 Thesis Organization

The remainder of this paper is structured as follows: Section 2 provides a review of related works in the field of video object segmentation, detailing both fully-supervised methods as well as existing weakly and unsupervised approaches. Section 3 details the method, explaining the consistency based loss function, regularization, and how it is integrated in to the model. The experimental setup and the results are presented in Section 4, followed by a discussion in Section 5 where we also draw conclusions and outline future research directions.

Chapter 2

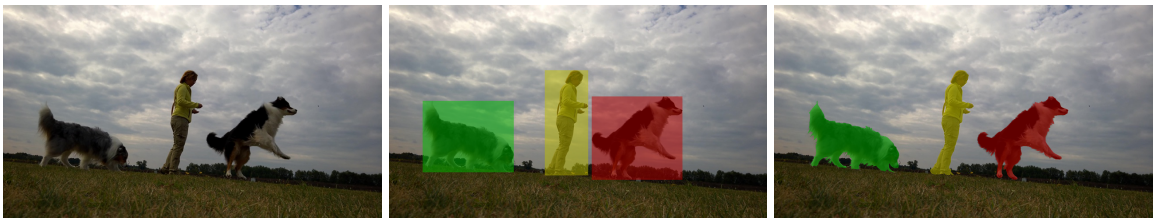
Background and Related Work

Semi-supervised video object segmentation, or video object segmentation as we call it in this thesis, involves the task of propagating a masked object highlighted in the first frame, throughout the entirety of the video. It is class-agnostic, meaning that any object given in the first frame should be propagatable. Despite being referred to as *semi-supervised*, it does not adhere to standard machine learning naming convention since it is typically trained in a fully supervised manner. Instead, the term pertains to what information is given at test time, where semi-supervised usually means that a segmentation mask is given in the first frame, weakly-supervised means that a bounding box is given, and unsupervised means that nothing is provided. To avoid confusion, this thesis uses these terms in the standard sense referring to how much supervision is given during *training*. Furthermore, because of this ambiguity, authors refer to video object segmentation using various terms such as dense tracking, one-shot or semi-automatic video object segmentation.

Theoretically, there is a performance hierarchy directly related to the level of supervision provided. For instance, a model trained on masks should perform at least as well as one trained on bounding boxes, since masks can be converted into bounding boxes. Additionally, bounding box supervision should be capable of producing a model of equal or greater quality than an unsupervised model. This is due to the stronger supervision signal and the greater amount of contained information. However, this concept does not consider the influence of available data quantity. For instance, if we access larger datasets with less supervision, there exists a crossing point where the less supervised model outperform others. In essence, it is the total amount of information that holds significance, not the marginal information per annotation. Comparing information across different settings is also challenging. For example, quantifying whether a scribble annotation contains more information than a human language expression is not straightforward.

2.1 Video object segmentation

Segmentation in images and videos. Segmentation, the process of partitioning scenes into meaningful segments, is one of the most general and fundamental tasks in computer vision. Given its broad definition it is often divided into sub-problems based on how *scene* and *meaningful* are defined. For example 2D image and videos are two of many scene configurations. Image segmentation can be seen as a special case of video segmentation, where the former have historically attracted more attention as since it was an important sub-goal and computationally tractable. Then there are the possible partitioning schemes, semantic segmentation [24] partitions semantically different classes and does not distinguish between multiple instances of the same class. Instance segmentation [12, 23] is similar to semantic segmentation but also discerns between different instances of the same class. It is often the case that only some objects are important to distinguish,



(a) Unsupervised (b) Bounding-box initialized (c) Mask initialized

(d) Language based (e) Scribble initialized

Figure 2.1: Different video object segmentation settings showing varying levels of supervision given in the first frame during test time. Naturally each setting comes with models of varying accuracy, robustness and speed.

while others are not, e.g. backgrounds. These two categories are referred to as *things* and *stuff* respectively. Panoptic segmentation [16] applies instance segmentation to the things category and semantic segmentation to the stuff category. These ideas are trivially extended to video setting by segmenting each frame in the video. Additionally, videos open up more settings such as propagating a highlighted object, often defined in the first frame of the video, see Figure 2.1 for common settings.

Early methods in video object segmentation. One of the initial approaches in video object segmentation involved background subtraction, already in the 1990s this approach was employed to segment and model humans using stationary cameras at 10 frames per second [44]. Although useful in some situations, this method comes with significant limitations since it operates under the assumption that the background is known and that the viewpoint remains either stationary or follows a predetermined motion.

The naïve way to extend image segmentation to video object segmentation by segmenting each frame independently is not trivial since the object to segment is not pre-defined. Instead the network finds out what to segment during test time. Many early methods solved this by relying on online learning [36, 4, 29, 13] where network is tuned during test time, this inherently makes it slow, processing frames at well below one frame per second. For example [4] trains a base network on ImageNet [9], then a parent network on a video object segmentation dataset, then finally during training the test network is fine-tuned on the first frame that is given as initialization. The rest of the frames are then segmented independently

Another problem with early methods is they usually use only one reference frame, either the first frame, making it hard for the network to handle large changes in object since only one example has been given. The other case where the previous frame is always looked at is also problematic due to occlusions and representation drift. For example [4] always relies on the first frame

Memory and matching based methods. As alluded to before, memory methods [28, 11, 7, 51, 52, 8, 21, 49, 50, 33] are important since all previous frames matter when predicting the next segmentation. Given that video object segmentation is object-agnostic it is especially important to use the information containing different viewpoints, perspectives and deformation in all previous frames. The attention mechanism [2, 35]

solves the problem of which previous information is important. Since videos are sequences in time and potentially in space as well, the attention mechanism naturally extends into video object segmentation. STM [28] formulates attention as for each pixel in a query frame which we want to segment, which previous pixels in both space and time should we attend to to predict the label. This formulation have been especially prolific whereas many state-of-the-art methods has since followed and built upon it [7, 51, 52, 8].

SSTVOS [11] takes further inspiration from the Transformer architecture [35] and uses multi-headed self-attention. To keep down the computational burden from attending over all spatio-temporal dimensions they use features and sparsity $\mathcal{O}(H^2W^2T^2)$. There is a trade-off when you search in the memory, for example global searching of STM [28] is more robust to occlusions and fast changes, while KMN [33] use a more local search based on that the queries should just search spatially close to where that are in previous frames.

XMem [7] focuses on handling longer videos, a property essential for many practical applications. This means that the memory size has to be virtually independent with respect to the number of frames for both matching and space. They solve this by using three separate memory stores: sensory, working, and long-term. Where the sensory memory has high spatial detail and is updated every new frame, the working memory is updated every few frames and stores longer temporal until it hits a pre-defined max size where memories are consolidated and put in to the long-term memory. The effectiveness of the memory potentiating algorithm uses features with high recall frequency, allowing memory to store relevant information over long time periods while still being compact, resulting in state-of-the-art performance with real-time speed.

Box-initialized methods. Some works [40, 42, 45, 3, 37, 22] consider the setting where a bounding box is given during test time rather than an exact mask. SiamMask [40] focuses on efficiency, achieving real-time segmentation with over 30 frames per second. It achieves this by using only the first frame during prediction where the object is cropped from the reference frame and a lightweight feature matching model can match the query frame. By exclusively relying on the first frame it sacrifices accuracy since objects may go under big appears shifts and deformations as time goes on. On the other spectrum is BoLTVOS [37] which achieves better performance but at the cost of being very slow, taking more than a second for a single frame. They explore the idea of first tracking at a box level and then converting the box to a mask. A conditional R-CNN [32] is first used to track the object conditioned on the first frame bounding box, then in another stage a Box2Seg network [26] is used to turn the box into a complete mask.

LWL [3] extends their standard video object segmentation model to the box-initialized setting by adding an initial box to mask step. The step encodes the bounding box and applies their already trained decoder to generate the pseudo-mask, which is then used as initialization. The bounding box encoder, being a single linear layer followed by two residual blocks, is then easily trained by freezing the rest of the network. BTRA [22] trains a student box initialized model along with a teacher mask initialized model to distill improved representations from the teacher. Furthermore, multiple intermediate memory frames are used along with the first and previous frame during segmentation. The memory frames are aggregated such that the method is still relatively efficient with around 8 frames per second. Still, our method is almost three times as fast.

2.2 Weakly-Supervised Video Object Segmentation

With the abundance of video data available online it is vital to enable models to learn without pixel level segmentation masks. Methods focusing on using cheaper forms of annotations such as bounding boxes [34, 15] are therefore an important study, but have yet to attract attention in the domain of video object segmentation.

In the image segmentation domain multiple weakly-supervised methods have been proposed [34, 1, 17].

Some of which show weak performance or suffer from being slow, preventing them from being extended in to the video domain. An efficient and performant image segmentation model is BoxInst [34], achieving great results on COCO instance segmentation [23] when trained in a weakly supervised setting with bounding boxes. The authors introduce a pair of loss functions: projection loss and pairwise loss. The projection loss ensures that the predicted mask is aligned with the ground-truth bounding box by comparing the projections of the predicted mask and ground-truth bounding box. In contrast, the pairwise loss promotes label consistency at a finer granularity. This is achieved by incentivizing neighboring pixels with similar color to be assigned the same label.

MaskFreeVIS [15] is a weakly-supervised color based method in the video instance segmentation setting, relying exclusively on bounding boxes for annotation. MaskFreeVIS extends the BoxInst approach in the temporal dimension by a K-nearest neighbor matching between patches in consecutive frames to derive a self-supervised loss. Notably, this matching scheme allows one patch to match zero to potentially multiple frames, giving rise to dense supervision. The matching is purely color-based and relies on a parameter free matching scheme. For efficiency, the matching process is confined to a radius surrounding the patch in the preceding frame, leveraging spatio-temporal locality.

2.3 Unsupervised Video Object Segmentation

Methods that use no annotation at all are also becoming increasingly performant with different forms of self-supervision showing promise [38, 18, 14, 19].

CRW [14] tackles unsupervised VOS by posing the problem as a contrastive graph walk problem where nodes are patches of an image and edges connect patches from neighboring frames. The problem is then to learn a path from an initial patch at timestep t, walk k frames, and then back to a patch in frame t which should be the same query patch, forming a cyclic consistency constraint. A problem with this approach is that disappearing objects disconnects the path breaking the cycles.

LIIR [19] further gains supervision by considering cross-video video affinities, contributing more negative examples, making instance discrimination easier. They combine this with intra-video reconstruction, creating an objective that promotes intra-video correspondences while penalizing unreliable associations both within and between different videos.

Colorization [38] is a method that learns to colorize a gray scale query image using temporal color coherency between the query frame and a single reference frame. They then re-use their color propagation model to propagate object labels, turning it in to a tracker and segmenter, outperforming alternative methods using optical flow.

MAST [18] performs dense-tracking by finding correspondences in pixel-space between frames, using the inter-frame correspondences, the mask can then be propagated from the given reference frame. To find correspondences they perform ablation studies with different color-spaces and find that the Lab color space is most effective. Additionally, they put effort in to maximizing the self-supervision achieved by matching with all previous frames at the cost of computational efficiency in relation to methods considering only a fixed number of frames. Still MAST runs at over 5 frames per second due to their two stage attention that first searches over coarse candidate regions and only applies the more costly fine-grain matching within the selected candidate regions.

Chapter 3

Method

To define our problem setting more formally we take inspiration of the notation in [41] and define $\mathcal{X}$ and $\mathcal{Y}$ where $\mathcal{X} = \mathcal{V} \times \mathcal{H}$ is the set of video inputs along with its annotation that informs the correct object. And $\mathcal{Y}$ is the set of video segmentations. The goal of video object segmentation is to find a function f^* that maps $x \in \mathcal{X}$ to the segmentation of the video $f^*(x) \in \mathcal{Y}$. Here $x = (v, h)$ where $v = \{I_t\}_t$ is the video and h the given aid.

Specifically, for deep learning based methods the goal is to learn a differentiable model f_θ parameterized by weights $\theta \in \Theta$ that approximates f^* given a dataset $\{(x_n, z(x_n))\}_{n=1}^N$. By empirical risk minimization the goal is to find

$$\theta^* \in \underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left(f_\theta(x_n), z(x_n)\right) \tag{3.1}$$

where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ is a loss function. In practise, due to the high dimensionality of the parameter space, $\dim \Theta \approx 6 \cdot 10^7$, we find an approximation to θ^* by using gradient based methods.

In the fully supervised case $z(x) = f^*(x)$ and $\mathcal{H}$ is the first frame mask, i.e every frame in the video x have a perfect mask label. In the box-supervised case we seek to approximate f^* while only using bounding boxes as supervision. Hence, $\mathcal{H}$ is the set of bounding boxes, and $z(x) = b^*(x)$ where b^* is a mapping from video to the to the smallest axis aligned bounding box covering the mask given by $f^*(x)$. Note that $\mathcal{H}$ can be different under test and training time, for example we can train with box-initialization but evaluate with mask-initialization. The rest of this thesis explores how to construct a weakly-supervised loss function $\mathcal{L}$ such that f_{θ^*} approximates f^* which is not obvious since the network is never shown outputs of $f^*(x)$ only $b^*(x)$.

3.1 XMem

We chose to adopt XMem [7] as the fully-supervised video object segmentation model, serving as a strong base with great model architecture. Furthermore, the efficient and high performing memory architecture of XMem complements our loss function that operates on individual frames and frame pairs, improving the occlusion handling. In this

To address the challenge of determining which previous frames and positions are relevant when producing a new mask prediction, XMem takes inspiration from the Atkinson-Shiffrin memory model. A model of human memory where multiple memory sources with different spatial and temporal fidelity work together.

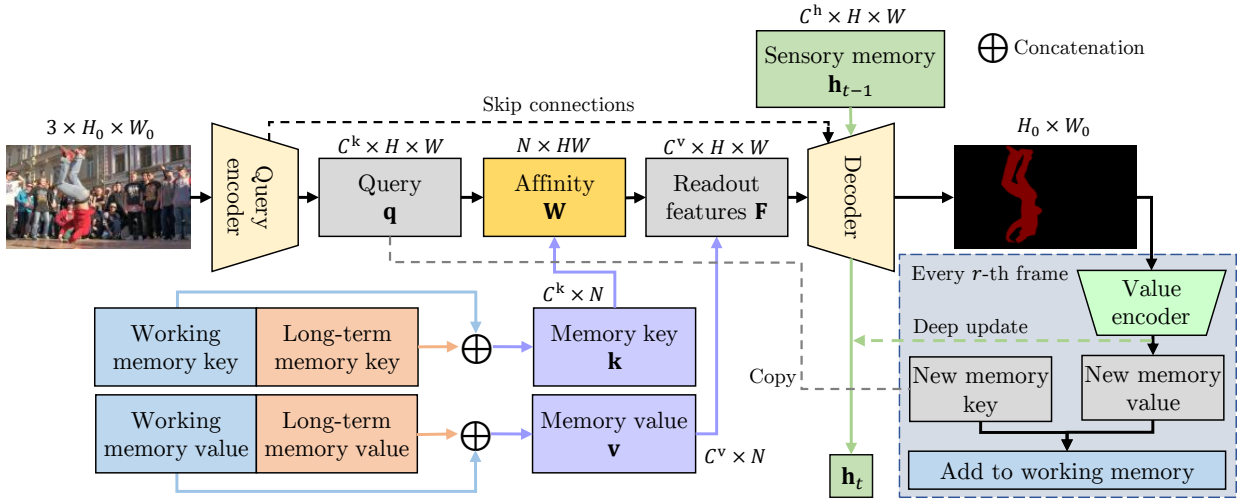


Figure 3.1: Figure sourced from XMem [7], illustrating the segmentation pipeline for a single $H_0 \times W_0$ RGB frame. Specifically, the query encoder extracts strided features $\boldsymbol{q}$ from the input frame. The query is then used to match previously extracted memory keys by constructing an affinity matrix $\boldsymbol{W}(\boldsymbol{k}, \boldsymbol{q})$. Using the affinities, the memory values are then read resulting in features $\boldsymbol{F}$. Along with the sensory memory $\boldsymbol{h}_{t-1}$ and frame features, the decoder then produces the predicted mask. Finally, the frame and mask are passed to the value encoder to extract new memory features, used in future timesteps.

This capability of searching both space and time allows the model to effectively handle prolonged occlusions and rapid spatial changes, while preventing representational drift. They implement this by a form of cross-attention based memory where a query frame is encoded to query $\boldsymbol{q} \in \mathbb{R}^{C^k \times HW}$ and performs matching over memory keys $\boldsymbol{k} \in \mathbb{R}^{C^k \times N}$ where $N = THW + L$ resulting in an affinity matrix $\boldsymbol{W} \in \mathbb{R}^{N \times HW}$. The N dimension contains memory information from the previous T frames along with L long-term memory prototypes. Using the memory values $\boldsymbol{v} \in \mathbb{R}^{C^v \times N}$ along with the affinity matrix The actual memory features are then read using $\boldsymbol{F} = \boldsymbol{v}\boldsymbol{W}$. Intuitively, we can interpret the memory key as weights learned to encode visual semantics allowing a query to match previous semantic information contained in the memory key with robustness to visual changes such as perspective changes, deformations, etc. Whereas the memory value is learned to provide cues about the tracked objects such that the decoder can produce masks. Unlike the approach of utilizing the scaled dot product attention commonly used in NLP, XMem introduces a scoring mechanism $\boldsymbol{S}(\boldsymbol{k}, \boldsymbol{q})$ based on negative squared anisotropic L2. Calculating $\boldsymbol{W}$ by taking the row-wise softmax of the similarity matrix $\boldsymbol{S}$.

Since the affinity matrix with dimensions $HWT + L \times HW$ grows linearly with respect to the video length T, its feasibility diminishes when applied to videos comprising thousands of frames. Furthermore, a linear decrease in efficiency is also not desired from a performance point of view. Consequently, to mitigate these issues, the capacity of the long-term and working memory is capped at a maximum size of $HWT_{max} + L_{max}$. Upon reaching this threshold, the memory is consolidated to compactly store only essential features.

3.1.1 Loss

XMem, along with other fully supervised state of the art models [52], utilize a combination of dice loss and cross entropy. More specifically, XMem uses the bootstrapped cross entropy where only the top p percentage of pixels with highest loss are back-propagated, making the network focus on pixels that are

hard to segment. In this section we ignore the batch dimension, essentially assuming a batch size of one. In practise the final loss is calculated by averaging over the batch dimension.

To formally describe the loss functions used by XMem, we assume for a timestep t that $\hat{m}^{(t)} \in [-\infty, \infty]^{N_c \times H \times W}$ is the unnormalized logits predicted by the network, where N_c is the number of classes, i.e the number of objects N_c and the background class. To simplify the notation we flatten the image dimensions such that $\hat{m}^{(i)} \in [-\infty, \infty]^{N_c \times HW}$. Now for a single timestep $t \in [T]$ we can assume that the likelihood of a pixel $i \in [HW]$ having the correct class m_i is given by

$$p_\theta(\hat{m}_i^{(t)}) = \frac{\exp \hat{m}_{m_i i}^{(t)}}{\sum_{c \in [N_c]} \exp \hat{m}_{ci}^{(t)}}. \tag{3.2}$$

To construct a loss from Equation 3.2 we apply the logarithm and negate, obtaining

$$l_i^{(t)} = -\log p_\theta \left(\hat{m}_i^{(t)} \right). \tag{3.3}$$

Now let $\mathcal{S}_t$ be a set containing the indices of the pixels with the top p largest losses from $m^{(t)}$ and to calculate the full loss we average over all pixels and time steps

$$\mathcal{L}_{BCE}(\hat{m}, m) = \frac{1}{T \lfloor pHW \rfloor} \sum_{t \in [T]} \sum_{i \in \mathcal{S}_t} l_i^{(t)}. \tag{3.4}$$

Note that if we set $p = 1$ we recover the standard negative log-likelihood, or equivalently, the cross entropy loss which got its name from the information theory perspective. As before, we assume we have a single batch, the dice loss is also calculated independently over frames and averaged. For dice loss we also calculate each object independently. So for object $c \in [N_o]$

$$d_c^{(t)} = 1 - \frac{\epsilon + 2 \sum_i \hat{m}_{ci}^{(t)} m_{ci}^{(t)}}{\epsilon + \sum_i \hat{m}_{ci}^{(t)} + \sum_i m_{ci}^{(t)}} \tag{3.5}$$

then we average over but we skip the

$$\mathcal{L}_{dice}(\hat{m}, m) = \frac{1}{T N_o} \sum_{t \in [T]} \sum_{c \in [N_o]} d_c^{(t)}. \tag{3.6}$$

Finally, the full loss is given by

$$\mathcal{L}_{oracle} = \mathcal{L}_{BCE} + \mathcal{L}_{dice}. \tag{3.7}$$

3.2 Spatial Consistency

In this section we describe the weakly and self-supervised losses used to replace the fully-supervised loss in Equation 3.7 that require exact masks.

3.2.1 Global consistency

The projection loss was introduced by Tian *et al.* [34] for image segmentation and later extended to videos by applying it independently to each frame [15]. It works on the premise that the predicted mask should be consistent with the bounding box ground-truth as illustrated in Figure 3.2. The projection loss provides a varied supervision signal depending on the shape of the objects to track. If the object is a rectangle it

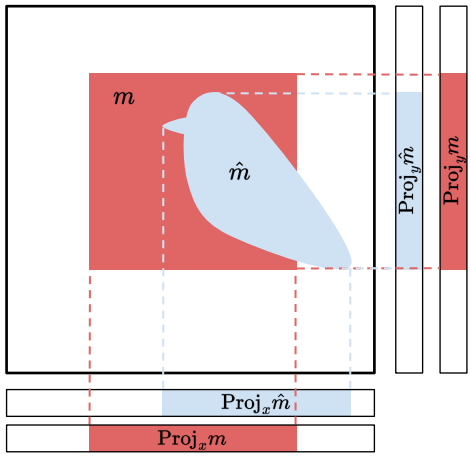


Figure 3.2: Illustration of the projections used to compute the projection loss. Given an image with predicted mask $\hat{m} \in [0,1]^{H\times W}$ shown in blue, and the ground-truth bounding box $m \in \{0,1\}^{H\times W}$ shown in red, the masks are projected in to one-dimensional vectors in the vertical and horizontal dimension. The loss is then calculated using the projected vectors along with the dice loss.

is lossless, while if the object is a diagonal line, a significant share of the information is lost. We can quantify this by calculating the relative area that the exact masks fills up in comparison to the bounding box. In Figure A.2 we do this for objects in a popular dataset and show its histogram. There are cases where this assumption is theoretically violated as shown in Figure 3.4 but this is not of concern since the case is exceedingly rare considering the rotation augmentation step during pre-processing. Furthermore, the projection loss is just one component of the loss.

To formally describe the projection loss we consider a binary problem for each object of the N_o objects. Let $\hat{m} \in [0,1]^{H\times W}$ be the mask prediction for an individual object in a single frame and $m \in \{0,1\}^{H\times W}$ its ground-truth bounding box. For alignment along the width dimension or the x-axis, we apply the projection, $\text{Proj}_x \colon [0,1]^{H\times W} \mapsto [0,1]^W$, obtaining

$$\begin{cases} \hat{p}_x = \text{Proj}_x\hat{m} = \max\limits_y(\hat{m}), \\ p_x = \text{Proj}_x m = \max\limits_y(m). \end{cases} \tag{3.8}$$

Similarly for the y-axis projection, $\text{Proj}_y \colon [0,1]^{H\times W} \mapsto [0,1]^H$ we have

$$\begin{cases} \hat{p}_y = \text{Proj}_y\hat{m} = \max\limits_x(\hat{m}), \\ p_y = \text{Proj}_y m = \max\limits_x(m). \end{cases} \tag{3.9}$$

Given the two projection, a loss l can be calculated by applying dice loss on the one dimensional projections in both dimensions,

$$l = \mathcal{L}_{dice}(\hat{p}_x, p_x) + \mathcal{L}_{dice}(\hat{p}_y, p_y) \tag{3.10}$$

where $\mathcal{L}_{dice}$ denotes the smoothed dice loss with $p = 2$,

$$\mathcal{L}_{dice}(\hat{m}, m) = 1 - \frac{\epsilon + 2\sum_i \hat{m}_i^p m_i^p}{\epsilon + \sum_i \hat{m}_i^p + \sum_i m_i^p}. \tag{3.11}$$

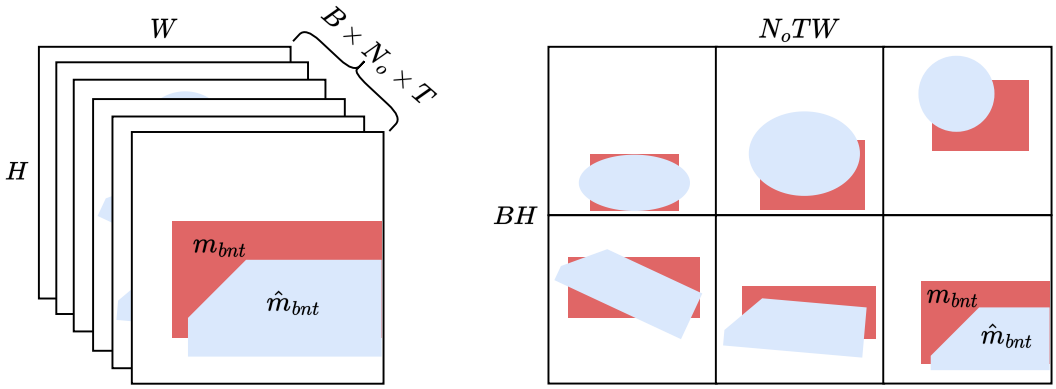


Figure 3.3: Example of two potential ways to calculate the loss for $B = 2$, $N_o = 1$ and $T = 3$. On the left each $H \times W$ image is calculated independently and then aggregated over the $B \times N_o \times T$ dimensions. On the right the collection of images are treated as one single image of size $BH \times N_oTW$, with mask $\hat{m} \in [0, 1]^{BH \times N_oTW}$ concatenated of $\hat{m}_{bnt}$ for $bnt \in [B] \times [N_o] \times [T]$ Note that both methods are permutation invariant with respect to dimensions $B \times N_o \times T$.

Equation 3.10 treats the case of projection loss for a single object c in a single frame t, which we denote by $l_c^{(t)}$, extending to the full video by taking averaging over both we have

$$\mathcal{L}_{proj} = \frac{1}{TN_o} \sum_{t \in [T]} \sum_{c \in [N_o]} l_c^{(t)}. \tag{3.12}$$

In practise, with batching, multiple objects and multiple frames there are multiple ways of calculating the projection loss. Since each video is described by a tensor with shape $B \times N_o \times T \times H \times W$, we could either see this as a large picture with dimensions $BN_oTH \times W$ and apply the loss. Or, the loss can be calculated independently for each $H \times W$ image and averaged over the BN_oT dimension. In this work we use the former approach, but note that for newer datasets with more occlusions and imbalance in the time dimensions that the former approach is interesting. Furthermore, an interesting note is that by merging objects in the N_o dimension to a single mask, no object association is needed between frames, reducing the annotation-burden.

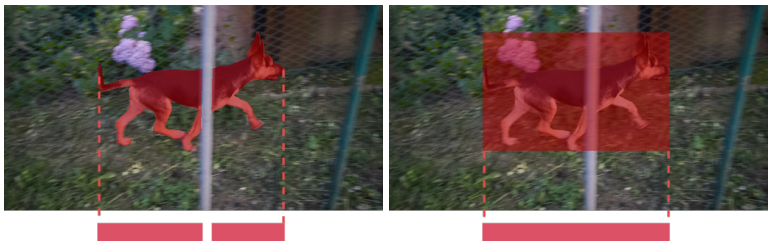


Figure 3.4: Example of an occlusion creating two disconnected masks which stays disconnected after projection onto the x-axis, since the split is aligned with the y-axis. This causes false positives in the x-axis projection.

3.2.2 Local consistency

To complement the coarse supervision provided by the projection loss and improve fine-detail Tian *et al.* [34] further introduces a loss based on a color-consistency heuristic. It states that *proximal pixels with the*

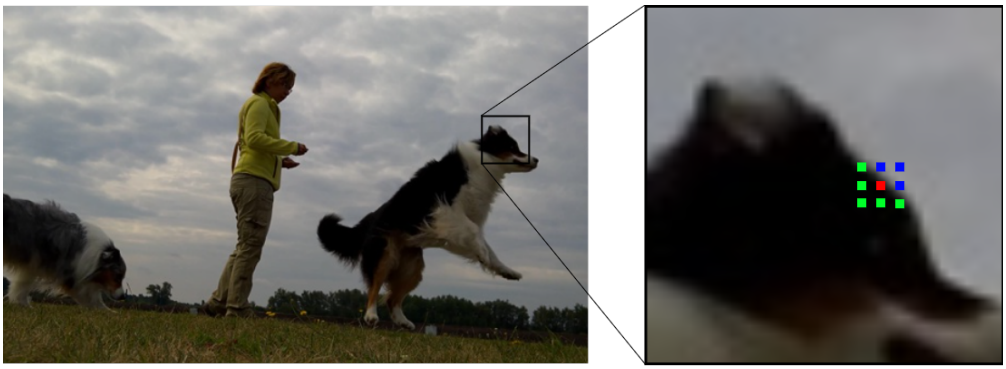


Figure 3.5: Illustrating how the center pixel shown in red, can be used to create supervision from proximal pixels shown in green and blue. The green pixels satisfy the color similarity threshold with the center pixel, while the blue pixels do not satisfy the threshold. Shown here is a 3×3 kernel with dilation 2. In principle the pattern is arbitrary as long as the spatial locality assumption is valid. Importantly, the pairwise loss is completely agnostic to the red unmatched pixels.

same color likely have the same label. Formally, let C be the color space and τ an appropriately chosen threshold, then for proximal pixels $p, p' \in C$ with labels y and y' respectively, the following statement holds with high probability

$$S(p, p') \geq \tau \implies y = y' \tag{3.13}$$

where $S : C \times C \mapsto \mathbb{R}$ is a color similarity function. From statement 3.13 we get an implicit dataset containing only positive labels, the case when two labels are equal. Unfortunately, it is hard to find a heuristic for when labels should be different. For example, the negation of the statement is not true, since an object often has multiple colors as evident from the black and white dog in Figure 3.5.

To simplify the color similarity function the lab color space is used as it is aligned with human vision [27]. Being designed in a way such that calculating the euclidean distance between two points in the color-space approximates is similar to the difference a human would perceive. If the RGB color-space were used, differences that are imperceptible to the human eye could have a large distance in color-space.

For each pixel p in a frame I we define a set $\mathcal{P}_p$ of proximal pixels to pixel p, in Figure 3.5 these are the green and blue pixels. In our case we let $\mathcal{P}_p$ be the pixels in a $K \times K$ kernel with center pixel p and arbitrary dilation. Resulting in up to $K^2 - 1$ supervision pairs as the kernel consist of K^2 pixels but we lose one supervision since the center pixel's class is always equal to itself. We also define $\mathcal{S}_p = \{\, p' \in \mathcal{P}_p \mid S(p, p') \geq \tau \,\}$ as the set of proximal pixels that satisfy the color threshold, corresponding to the green pixels in Figure 3.5. We follow [34] and use the exponential similarity function

$$S(p, p') = \exp\left(-\frac{1}{\gamma}||p - p'||_2\right) \tag{3.14}$$

where the temperature parameter γ can be adjusted to make the values more uniform in $[0, 1]$.

For two two pixels p and $p' \in \mathcal{S}_p$ we want to promote their respective labels y_p and $y_{p'}$ to be equal. Since the network outputs smooth labels m_p and $m_{p'}$, we instead optimize for them to be equal. Interpreting the output m_p of the network at pixel p as the probability of being foreground, the probability of pixel p and p' being equal is

$$p_\theta\left(y_p = y_{p'}\right) = m_p m_{p'} + (1 - m_p)(1 - m_{p'}). \tag{3.15}$$

The objective is then to train the network to assign high probability of having the same label, i.e. maximizing Equation 3.15. Since we want to jointly maximize this for all $p' \in S_p$ and for each pixel p in the image, we can instead minimize the negative log likelihood, and assuming independence we obtain

$$l = -\frac{1}{HW} \sum_{p} \sum_{p' \in \mathcal{S}_p} \log p_\theta \left(y_p = y_{p'} \right). \tag{3.16}$$

To obtain the complete loss for all the objects over the full video, we average over the individual losses $l_c^{(t)}$ given by Equation 3.16

$$\mathcal{L}_{pair} = \frac{1}{TN_o} \sum_{t \in [T]} \sum_{c \in [N_o]} l_c^{(t)}. \tag{3.17}$$

To stop the loss from being outweighed by background pixels, we only apply the pairwise loss pixels inside the bounding box.

3.3 Temporal Consistency

The temporal loss introduced by Ke *et al.* [15] extends the previous idea of intra-frame color-consistencies to the temporal domain by considering inter-frame color-consistencies. Leveraging the key premise that *proximal pixels in space, time and color-space, likely have the same label*. Since objects between two frames move and deform, the loss also has to be robust to such changes. Importantly, these changes can be assumed to be relatively small since for a sufficiently high frame rate, objects tend to move continuously and coherently. To accommodate this, a greater receptive field is used implemented by matching patches instead of pixels. We denote an $N \times N$ patch centered around pixel p in frame t by $P_p^{(t)}$.

Similarly to the matched pixel set in the projection loss, we define a set $\mathcal{C}_p^{tt'}$ which contains patches in frame t' that satisfies the color similarity threshold with patch P_p, containing the matched patches as seen in step *i* in Figure 3.6. Representing the matched patches $P^{(t')} \in \mathcal{C}_p^{tt'}$ as vectors we use the same similarity function defined in Equation 3.14. To derive the loss from the matched patches $P_p^{(t)}$ and $P^{(t')}$ we apply the consistency loss between the center pixel p and each pixel p' in the matched patch $P^{(t')}$.

$$l^{(t,t')} = - \sum_{P^{(t')} \in \mathcal{C}_p^{tt'}} \sum_{p' \in P^{(t')}} \log p_\theta \left(y_p = y_{p'} \right). \tag{3.18}$$

where $p_\theta \left(y_p = y_{p'} \right)$ is defined in Equation 3.15.

The cyclic-tube matching scheme was found to be best trade-off between performance and computational efficiency [15]. Which is where each matching is done in consecutive frames and the final frame is then connected back to the first frame, forming a cyclic consistency constraint. Using this we obtain

$$l = \frac{1}{T} \sum_{t=0}^{T-1} l^{(t,t+1 \bmod T)} \tag{3.19}$$

for a single object c, introducing the subscript c to Equation 3.19 and averaging over all the objects, we obtain

$$\mathcal{L}_{temp} = \frac{1}{N_o} \sum_{c \in [N_o]} l_c. \tag{3.20}$$

In similar fashion as was the case with the bootstrapped cross entropy and pairwise loss, we here use the bounding box as filters to make the network concentrate on the area around the object and not be overwhelmed or dominated by the background.

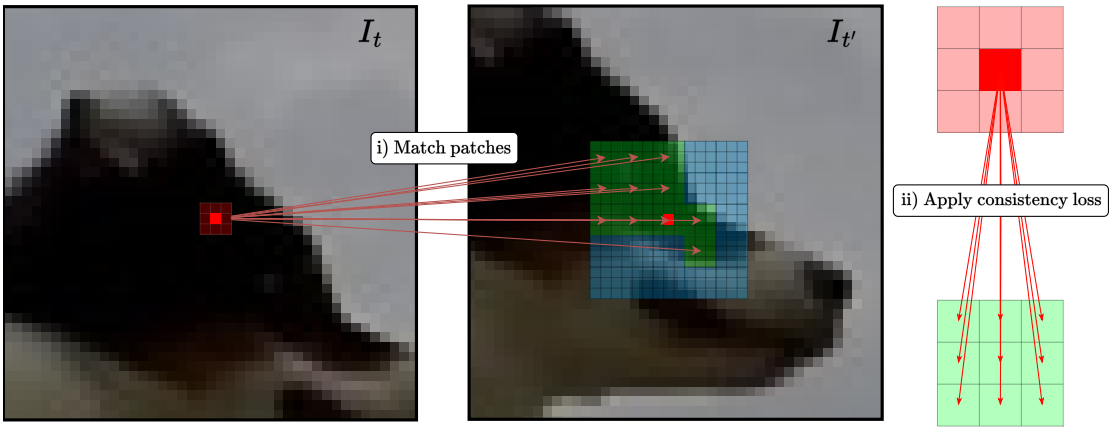


Figure 3.6: Step i) match a patch in frame t with patches inside a region of interest in frame t'. Patches are colored green when the color similarity between patches are high enough, otherwise they are colored blue. Step ii) for each match we add the loss that the center pixel of patch should be consistent with all pixels in the other patch. In this case we use patch size $N = 3$ and search size $M = 5$ with disjoint search. If we did an exhaustive search we could have matched more, but it is a computational trade-off.

3.3.1 Detach Regularization

If the network is trained with only the temporal loss, the loss can be circumvented by always outputting all zero, or all one masks, i.e $m = 0$ or $m = 1$. Since these are minimizers to the temporal loss as it maximizes Equation 3.15. When the projection loss is also active, the $m = 1$ case does not happen since it will be outside the bounding box and will be penalized by the projection loss, so instead the minima with all zeros $m = 0$ is preferred. But this minima also violates the projection loss, and instead another degenerate solution is reached characterized by having as few activate pixels as possible to satisfy the projection loss, while otherwise being zero, circumventing the temporal loss and pairwise loss.

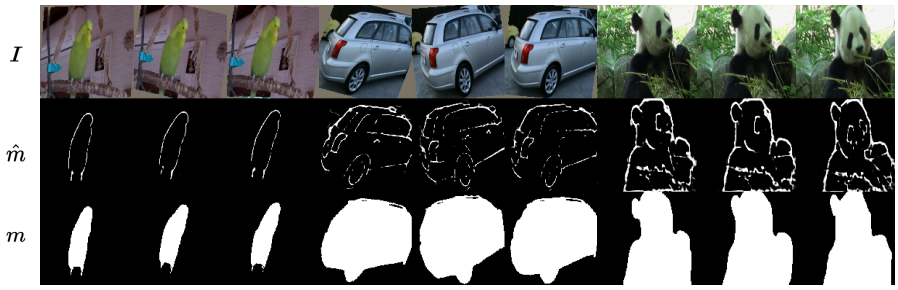


Figure 3.7: Examples of the degenerate minima where the predicted masks $\hat{m}$ have small area such that the satisfy the projection loss while having a low pairwise and temporal loss. We device a regularization technique to prevent this.

Similar to before, we let m_p^θ and $m_{p'}^\theta$ be the mask output between two pixels p and p' in frame t and t' from the network f_θ, where the superscript indicates that it depends on the network parameters θ. The temporal loss between the singular pixel-pair is then given by

$$L_{temp}(m_p^\theta, m_{p'}^\theta) = -\log\left(m_p^\theta m_{p'}^\theta + (1 - m_p^\theta)(1 - m_{p'}^\theta)\right). \tag{3.21}$$

and in the backward pass $\nabla_\theta L_{temp}(m_p^\theta, m_{p'}^\theta)$ is calculated and we update θ accordingly to make the two

Table 3.1: We measure how accurate the temporal consistency assumption is on the training splits of YouTubeVOS-2018, DAVIS-2017 and MOSE. We sample images with pre-processing identically to as done during training. A true positive (TP) is defined as when the color threshold is satisfied and the labels of the two pixels match, while a false positive (FP) is when the color threshold is satisfied but the labels differ. Precision is calculated as $TP/(TP + FP)$. Supervision proportion is defined as the ratio of number of pixels that satisfies the color threshold, to the total number of pixels in the batch $B \times T \times N_o \times H \times W$. Both metrics are then averaged over all samples in the dataset.

Threshold τ	**0**	**0.005**	**0.01**	**0.05**	**0.1**	**0.2**	**0.3**
YouTubeVOS-2018:							
Supervision proportion	100%	18.9%	11.2%	8.50%	5.75%	4.10%	1.95%
Precision	0.969	0.994	0.995	0.996	0.997	0.997	0.997
DAVIS-2017:							
Supervision proportion	100%	18.1%	16.2%	10.7%	8.00%	5.11%	3.41%
Precision	0.973	0.996	0.996	0.998	0.998	0.999	0.999
MOSE:							
Supervision proportion	100%	20.2%	18.0%	11.9%	8.91%	5.74%	3.82%
Precision	0.982	0.996	0.997	0.998	0.998	0.998	0.998

predictions m_p^θ and $m_{p'}^\theta$ consistent. A problem with this is that the network is encouraged to predict $m = 0$ or $m = 1$, which it does in practise. To solve this we instead update this loss to a two step procedure by first treating m_p^θ as a constant m_p with respect to θ and then $m_{p'}^\theta$ as $m_{p'}$, we then take the average of the two. This can be written as

$$L_{temp}^{reg}(m_p^\theta, m_{p'}^\theta) = \frac{1}{2}\left(L_{temp}(m_p^\theta, m_{p'}) + L_{temp}(m_p, m_{p'}^\theta)\right). \tag{3.22}$$

To get the full regularized temporal loss, $\mathcal{L}_{temp}^{reg}$ we average Equation 3.22 over all objects, all frames in the cyclic-connection and all pixel corresponding to the matched patches as before.

To see why Equation 3.22 provides regularization in comparison to $L_{temp}(m_p^\theta, m_{p'}^\theta)$, we consider the gradients $\nabla_\theta L_{temp}(m_p^\theta, m_{p'})$ and $\nabla_\theta L_{temp}(m_p, m_{p'}^\theta)$ individually. The former gradient will encourage m_p^θ to be closer to $m_{p'}$ while the latter encourage $m_{p'}^\theta$ to move towards m_p. In comparison $\nabla_\theta L_{temp}(m_p^\theta, m_{p'}^\theta)$ will point both masks toward zero or one at the same time. In practise we implement this in PyTorch by using `detach`, detaching the variable from the computational graph, such that it is not back-propagated anymore.

Chapter 4

Experiments

4.1 Datasets

Datasets have played a pivotal role in advancing the field of video object segmentation. Their function is to supervise models, and provide a standardized platform for evaluation such that methods can be compared. In this section, we introduce the datasets that was used to train our models.

DAVIS. DAVIS is a smaller scale VOS dataset with meticulously hand-labelled segmentation masks in high quality[30, 31]. Particularly, the 2016 version is single-object only, containing 30 videos for training and 20 videos for validation. The 2017 version is multi-object, i.e multiple objects can co-exist in a single frame and contains 60 videos for training and 30 videos for validation. The datasets contains 3 and 5 minutes of video respectively at 24 fps.

YouTubeVOS. The YouTubeVOS dataset series is based on Youtube videos with multiple objects in complex scences allowing for movement and occlusion, objects also disappear for at least one frame in a video 13% of times. The YouTubeVOS-2018 dataset [46] contains 3471 videos of varying lengths, see Figure 4.1, in 6 fps for training. The videos contain 65 categories and 5945 unique object instances. The validation set contains 474 videos with 65 seen categories and 26 unseen categories that is not included in the training set. In total there are 894 unique object instances. Both the training and validation set contains a combined 330 minutes of video with approximately 200k annotations. Since it such a large scale datasets, some annotations errors are prevalent, see Figure A.1.

The updated version, YouTubeVOS-2019 [48] augments the 2018 version with more annotations such that the training set contains 6459 unique object instances. Moreover, it adds more videos and annotations to the validation set resulting in 1063 unique object instances over 507 videos.

Static Images. For our static image dataset, we use a mix of five image segmentation datasets [47, 39, 53, 43, 6] totaling 37k images.

Long-Term Video Long-Term Video [20] is a small scale dataset containing three videos with lengths between 1k to 4k frames where each video has 20 uniformly sampled frames annotated. The dataset is further extended synthetically resulting in videos with between 4k to 11k frames by playing the videos back and forth [7].

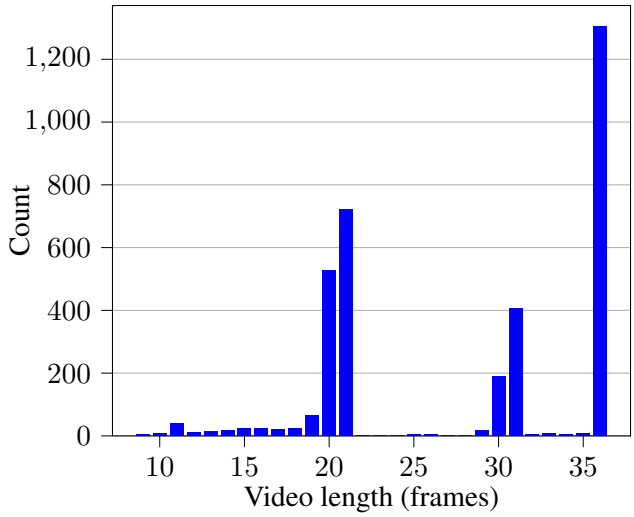


Figure 4.1: Histogram showing the number of frames per video in the YouTubeVOS training dataset.

4.2 Metrics

In video object segmentation there is no obvious scalar metric since the compared quantities range over both space and time. Hence, there has to be trade-offs and other design choices that may cause the metrics to not align with our intuition. Since DAVIS [30] introduced the $\mathcal{J}\&\mathcal{F}$ metric in 2016 newer video object segmentation datasets have followed them. The metric is defined as the mean of J, the Jaccard index, and $\mathcal{F}$, the contour accuracy and produce scores in the interval $[0, 1]$ where higher is better. The Jaccard index, also known as the intersection-over-union (IoU) is a classical set similarity metric and works well with imbalances which is important since the number of background pixels often dominate the number of foreground pixels in segmentation. For predicted mask M and ground-truth mask G It is calculated as follows:

$$\mathcal{J} = \frac{M \cap G}{M \cup G}. \tag{4.1}$$

Although positively correlated with $\mathcal{J}$, the contour accuracy $\mathcal{F}$ was deemed sufficiently uncorrelated and hence both metrics are used. It is based on the contours of the mask M and ground-truth G, and is calculated using morphological operators, calculating the precision and recall of the contours between them. The boundary accuracy is then the harmonic mean of the boundary precision and recall.

The final $\mathcal{J}\&\mathcal{F}$ metric is averaged over objects and over all validation videos. At first, a temporal consistency measure $\mathcal{T}$ was also considered, but was later discarded in [31] as it is largely affected by occlusions.

The inclusion of unseen categories in the YouTubeVOS validation sets allows it to report metrics on seen and unseen categories, giving insights on how the model generalizes to unfamiliar objects not included in the training portion. Following [7] $\mathcal{G}$ is defined as the average $\mathcal{J}\&\mathcal{F}$ over both the unseen and seen categories. For YouTubeVOS evaluation is done on a remote server through CodaLab since the validation labels are not publicly released.

4.3 Implementation Details

We base our implementation on XMem [7], a PyTorch based model, and replace its loss with the weakly-supervised losses described in Sections 3.2.2-3.3. As a backbone, XMem uses two ResNets from which the classification head and final convolutional stage have been discarded, producing frame features with stride

16. The query encoder uses the more powerful ResNet-50, while the value encoder leverages a ResNet-18 backbone. Importantly, these ResNet models were pre-trained for classification on ImageNet, and never saw any mask annotations.

Our loss function is implemented end-to-end on GPU, resulting in significant speed up during training in comparison to previous methods [34] relying on CPU libraries for color-space conversion.

Training. Training was conducted on a dataset composed of 8% DAVIS 2016 and 92% YouTube VOS 2018, we converted the masks in to bounding boxes such that the models relies exclusively on bounding boxes. All frames were also down-scaled to 480p via bi-cubic interpolation for improved loading speed. For the training hypter-parameters we mainly follow the training settings of XMem. The dataloader sample 8 frames and resizes them to 384×384 images with their corresponding bounding boxes for each batch. To maximize the information bounding boxes provide we do not apply the random cropping stage as XMem does. We use a batch size of 8 for main training and train for 110k iterations. In the setting with static image pre-training we follow [7] and deform an image into three different images, creating a pseudo-video. We then train using a batch size of 16 for 150k iterations.

We adopted a linear warm-up period for 10k iterations for the pairwise and temporal losses, ensuring the network initially emphasizes producing filled masks and then incrementally impose the losses that promotes fine details, improving the stability of training. The optimization phase employed the AdamW optimizer [25] with an initial learning rate of 1e-5, weight decay of 0.05, and after 80K iterations the learning rate is reduced by a factor of 10. During the training phase the backbone weights are not frozen.

Main training takes approximately 2 days on a single NVIDIA RTX A6000 GPU utilizing approximately 40 GB VRAM. But we noticed improved performance using 2 GPUs with data parallelism. To run multiple experiments concurrently we found it crucial to run on disjoint CPUs using `taskset` to not degrade training speed.

4.4 Results

In this section, we present results of our approach in comparison to traditional fully-supervised training methods and various other settings. We measure the supervision gap and and also do ablation study to understand the components of our loss function both quantitatively and qualitatively. Furthermore, we extend our method with additional data to assessing the potential of our approach when trained with other more cost-effective forms of annotation.

4.4.1 Quantitative Results

To the best of our knowledge we are not aware of other models with identical setting. Hence we compare to a verity of settings as seen in Table 4.1. In Table 4.1 we compare our method to state-of-the-art methods across different settings on the validation sets of DAVIS 2017 and YouTubeVOS 2018. These two was chosen since many other works did not report results on the 2016 and 2019 versions. Our method using only bounding boxes and additonally the image masks are trained in the same way with the only difference being that one is first pre-trained using masks. Furthermore, the results with bounding boxes initialization use the exact same model as the one initialized by the masks.

To gauge the model's practical performance we evaluate our model along with fully-supervised models on the Long-term Video dataset as shown in Table 4.2. The model we used is identical to our model used in Table 4.1.

Table 4.1: State-of-the-art comparison on DAVIS and YouTubeVOS. M and B indicate whether the method is mask initialized or box initialized respectively during *test time*. Methods that use video masks during *training* is indicated in the "Uses video masks" column, note that for unsupervised methods, only raw videos were used, while our method trains on video bounding boxes. * indicates that image masks were used during training.

Method	**Uses video masks**	**Init**	**DAVIS 2017 val**				**YoutubeVOS 2018 val**					
			$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	FPS	$\mathcal{G}$	$\mathcal{J}_s$	$\mathcal{F}_s$	$\mathcal{J}_u$	$\mathcal{F}_u$	FPS
Fully-supervised:												
STM [28]	✓	M	81.8	79.2	84.3	11.1	79.4	79.7	84.2	72.8	80.9	-
XMem [7]	✓	M	84.5	81.4	87.6	22.6	84.3	83.9	88.8	77.7	86.7	22.6
DeAOT [52]	✓	M	85.2	82.2	88.2	27.0	86.0	84.9	89.9	80.4	88.7	22.4
Unsupervised:												
Colorization [38]	✗	M	33.7	34.6	32.7	-	38.9	43.1	38.6	36.6	37.4	-
MAST [18]	✗	M	65.5	63.3	67.6	5.1	64.2	63.9	64.9	60.3	67.7	-
CRW [14]	✗	M	67.6	64.5	70.6	7.3	68.7	67.4	69.1	65.1	73.2	-
LIIR [19]	✗	M	72.1	69.7	74.5	-	69.3	67.9	69.7	65.7	73.8	-
Ours	✗	M	73.8	71.6	76.1	22.6	71.7	74.8	77.8	63.0	71.1	22.6
	✗*		79.0	76.5	81.6		77.7	77.5	81.6	71.5	80.2	
SiamMask [40]	✓	B	56.4	64.3	58.5	35	52.8	62.2	45.1	58.2	47.7	35
ROVOS [42]	✓	B	63.6	61.0	66.2	1.2	-	-	-	-	-	-
DMB [45]	✓	B	66.3	64.8	67.7	-	-	-	-	-	-	-
LWL [3]	✓	B	70.8	68.2	73.5	-	70.2	72.7	62.5	75.1	70.4	
BoLTVOS [37]	✓	B	71.9	68.4	75.4	0.7	65.7	67.3	-	58.9	-	0.74
BTRA [22]	✓	B	72.5	68.8	76.2	8.1	70.4	72.9	-	60.7	-	-
Ours	✗	B	67.4	65.1	69.7	22.6	65.1	69.2	70.1	57.1	64.1	22.6
	✗*		72.2	68.9	74.5		70.3	71.3	73.1	64.3	72.6	

Table 4.2: Comparison of our weakly-supervised method with fully-supervised state-of-the-art methods on long-term videos [20]. Values for the previous methods are sourced from [7]. * indicates that image masks were used during training.

Method	**Uses video masks**	Long Video (1×)			Long Video (3×)		
		$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
CFBI+ [50]	✓	50.9	47.9	53.8	55.3	54.0	56.5
CFBI [49]	✓	53.5	50.9	56.1	58.9	57.7	60.1
STM [28]	✓	80.6	79.9	81.3	75.3	74.3	76.3
AOT [51]	✓	84.3	83.2	85.4	81.2	79.6	82.8
AFB-URR [21]	✓	83.7	82.9	84.5	83.8	82.9	84.6
STCN [8]	✓	87.3	85.4	89.2	84.6	83.3	85.9
XMem [7]	✓	89.8	88.0	91.6	90.0	88.2	91.8
Ours	✗	78.2	77.3	79.2	77.7	76.2	79.3
	✗*	84.0	83.0	85.0	84.8	83.8	85.9

4.4.2 Ablation study

To understand the loss function we perform ablations to study how different compositions of the loss function affect the performance. In Table 4.3 the results of the ablation on the DAVIS datasets are reported while in Table 4.4 we report the results on the YouTubeVOS datasets. Note that for every loss composition the exact same model is used for evaluation across all the datasets. The hyper-parameters, except whether the loss is activate or not, are all the same between the ablations. To further see the how the model perform with more supervision, we train our model with access to full mask, given in the first frame, and where the rest of the frames are bounding boxes. Allowing the sampled sequence to use any mask from YouTubeVOS and DAVIS this simulates a larger training set. The results are shown in Tables 4.5 and 4.6.

Table 4.3: Results on DAVIS when training using bounding boxes. All four cases with different combinations of projection, pairwise, and temporal loss use the same hyper-parameters. The maximum values are seen in bold, and the second largest are underlined.

Loss function			**DAVIS 2016 val**			**DAVIS 2017 val**		
$\mathcal{L}_{proj}$	$\mathcal{L}_{pair}$	$\mathcal{L}_{temp}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
✓			76.4	78.2	74.5	69.9	68.0	71.7
✓	✓		77.1	77.8	76.4	69.8	68.0	71.6
✓		✓	<u>81.7</u>	<u>80.5</u>	<u>82.9</u>	**74.0**	<u>71.0</u>	**76.9**
✓	✓	✓	**81.9**	**82.0**	**81.8**	<u>73.8</u>	**71.6**	<u>76.1</u>

Table 4.4: Results on YouTubeVOS with mask-initialization when training using bounding boxes only of different combinations of projection, pairwise, and temporal loss.

Loss function			**YouTubeVOS 2018 val**					**YouTubeVOS 2019 val**				
$\mathcal{L}_{proj}$	$\mathcal{L}_{pair}$	$\mathcal{L}_{temp}$	$\mathcal{G}$	$\mathcal{J}_s$	$\mathcal{F}_s$	$\mathcal{J}_u$	$\mathcal{F}_u$	$\mathcal{G}$	$\mathcal{J}_s$	$\mathcal{F}_s$	$\mathcal{J}_u$	$\mathcal{F}_u$
✓			69.5	73.4	74.6	61.3	69.0	69.5	73.1	73.7	61.8	69.2
✓	✓		69.6	72.8	74.9	61.6	69.1	69.3	72.2	73.9	62.0	69.0
✓		✓	<u>71.1</u>	<u>74.4</u>	<u>77.1</u>	<u>62.6</u>	<u>70.4</u>	<u>70.9</u>	<u>74.0</u>	<u>76.3</u>	<u>62.9</u>	<u>70.2</u>
✓	✓	✓	**71.7**	**74.8**	**77.8**	**63.0**	**71.1**	**71.2**	**74.1**	**76.8**	**63.3**	**70.8**

Table 4.5: Results when training using bounding boxes and first frame exact mask of different combinations of projection, pairwise, and temporal loss. Here we allow the first frame mask to be any frame in each video.

Loss function			**YouTubeVOS 2018 val**					**YouTubeVOS 2019 val**				
$\mathcal{L}_{proj}$	$\mathcal{L}_{pair}$	$\mathcal{L}_{temp}$	$\mathcal{G}$	$\mathcal{J}_s$	$\mathcal{F}_s$	$\mathcal{J}_u$	$\mathcal{F}_u$	$\mathcal{G}$	$\mathcal{J}_s$	$\mathcal{F}_s$	$\mathcal{J}_u$	$\mathcal{F}_u$
✓			72.6	75.1	75.9	66.1	73.5	72.2	74.4	74.6	66.3	73.5
✓	✓		73.1	75.3	77.9	65.5	73.6	73.2	74.8	77.1	66.4	74.4
✓		✓	<u>74.7</u>	<u>76.6</u>	**80.1**	<u>66.7</u>	<u>75.5</u>	<u>74.5</u>	<u>75.9</u>	<u>79.1</u>	<u>67.3</u>	<u>75.8</u>
✓	✓	✓	**75.5**	**76.8**	**80.1**	**68.4**	**76.9**	**75.3**	**76.4**	**79.4**	**68.7**	**77.0**

Table 4.6: Results when training with first frame mask initialization and bounding boxes. All four cases with different combinations of projection, pairwise, and temporal loss use the same hyper-parameters. We allow the first frame mask to be any frame.

Loss function			**DAVIS 2016 val**			**DAVIS 2017 val**		
$\mathcal{L}_{proj}$	$\mathcal{L}_{pair}$	$\mathcal{L}_{temp}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
✓			74.1	73.9	74.2	69.3	67.5	71.2
✓	✓		77.2	76.4	78.0	71.6	69.5	73.7
✓		✓	83.3	81.1	85.5	73.1	70.2	75.9
✓	✓	✓	**85.1**	**83.6**	**86.6**	**76.3**	**73.4**	**79.3**

4.4.3 Qualitative Results

To illustrate the performance visually and give a deeper understanding of the loss functions and how they interact we evaluate our method qualitatively on DAVIS and YouTubeVOS. Figure 4.4 shows failure cases where our method struggles to produce coherent and correct segmentation masks. Figure 4.2 illustrates how different combinations of the losses work in a simpler scene with a relatively slow moving object captured by a almost still camera. The scene contains two instances of the same camel where only one is to be tracked. A more challenging scene is shown in Figure 4.3 where one object is imposed on the other, partially occluding the other. The objects are fast moving but still of high quality without blur or defects.

Figure 4.2: Five randomly sampled frames from the DAVIS 2017 category `camel` showing single-object segmentation with different losses. The first row uses only projection loss, second row uses projection and pairwise loss, third row uses projection and temporal loss, and the last row uses all losses.

Figure 4.3: Five randomly sampled frames from the DAVIS 2017 category `horsejump-high` containing two objects. The first row uses only projection loss, second row uses projection and pairwise loss, third row uses projection and temporal loss, and the last row uses all losses.

Figure 4.4: **Failure cases:** illustrating instance mix-up and holes in segmentation. From left: similar instances close to each other. Middle: sudden light shift from leaving the shadow. Right: rapid motion of the flag along with motion blur. Examples are taken from YouTubeVOS and DAVIS.

Chapter 5

Discussion and Conclusion

In this section, we delve into the key findings, limitations, and potential implications of our proposed box-supervised video object segmentation model. We highlight the model's strengths, weaknesses and shed light on areas that require further investigation. Our discussion encompasses both the practical implications of the model's performance and its broader contributions to the field of video object segmentation. We also propose future research directions and possible extensions to our model.

5.1 Performance

Comparison with fully-supervised methods. In Table 4.1 we see how our weakly-supervised model with mask-initialization scores 73.8% on $\mathcal{J}\&\mathcal{F}$ and 71.7% on $\mathcal{G}$ for the DAVIS 2017 and YouTubeVOS 2018 validation datasets respectively. While the fully-supervised methods score over 85% on these metrics across the datasets resulting in a large supervision gap. A potential explanation for this is that since our model is always trained in the box-initialization setting, the model may not learn to fully utilize the provided first frame mask as effectively as the fully-supervised models that have been trained using mask-initialization. Indicative of this theory, is the large gain in performance of our method when using the static-image pre-training with full masks, gaining almost 6 points across both datasets resulting in a relative performance of 90%, while the gain is a bit smaller in the box-initialization case. Although static-image pre-training increase the annotation costs of our model, it still uses less costly annotations since image masks do not require any object association between frames and is also of much smaller scale. Moreover, the lesser gap in the box-initialization setting also supports this theory since our model performs more closely to the fully-supervised model.

On the long-video dataset, see Figure 4.2, our model is more competitive outperforming the classical fully-supervised model STM [28] on the Long Video ($3\times$) dataset with our baseline method. Furthermore, using the static-image pre-training version of our model, we outperforms some of the more recent fully-supervised models such as AFB-URR [21] that is designed to handle long videos. Our performance here can largely be explained by the good long-term memory inherited from XMem.

Comparison with box-initialized methods. In Table 4.1 we see how our method weakly-supervised box-initialized model scores 67.4% on $\mathcal{J}\&\mathcal{F}$ and 65.1% on $\mathcal{G}$ on the DAVIS 2017 and YouTubeVOS 2018 validation datasets respectively. While the highest scoring fully-supervised method BTRA [22] scores 72.5% on $\mathcal{J}\&\mathcal{F}$ and 70.4% on $\mathcal{G}$ respectively. Given that our model only uses box-annotations this gap of around 5 points across both datasets is respectable. Having a relative performance of over 92% on both datasets. Taking inference speed in to account, our model runs faster than most of the box-initialized models, being

surpassed only by SiamMask [40] at 35 frames per second. However, our model significantly outperforms it by more than 11 points on both datasets. The performance is expected of SiamMask, since it only uses the first frame as reference, missing out on a lot of potential information our method has access to through using all previous frames as reference.

A notable characteristic of our model is that we use the same identical model for both the box and mask-initialized settings. Whereas other models that support multiple settings such as LWL [3], train an additional box-to-mask network to initialize their model with a mask instead of a bounding box.

Comparison with unsupervised methods. Compared to the unsupervised methods we achieve better results as expected, since we have access to bounding boxes during training. Although some of the unsupervised methods train on significantly larger datasets such as Kinetics [5] containing over 800 hours of video, while our dataset is less than 6 hours, the results are the same. Additionally, the unsupervised models suffer from low inference speed while some methods does not report any speed metrics.

Ablation results. In Table 4.3 and 4.4 we see the results of the ablations on DAVIS and YouTubeVOS respectively. On the DAVIS 2016 validation a large gap between using solely the projection loss, and using all three losses can be observed, demonstrating the additional supervision provided by the pairwise and temporal loss. Notably, adding the pairwise loss has a less significant inpact in both the projection only case and the temporal loss case. While the addition of the temporal loss have the greatest impact. It is somewhat surprising that the pairwise loss has less impact on the results considering the large effect it had in the image segmentation setting studied in BoxInst [34]. One reason for this may be that they considered that images have more supervision opportunities rather than videos where individual frames can be blurry and of lower quality. Figure 4.2 and 4.3 illustrates the ablation study visually, showing in the same order as the tables how the segmentations improve as losses are added. Especially noticeable are the extremities which are clearly improved when the pairwise and temporal loss is activate. For example in Figure 4.2 where the camel's non-convex legs are segmented almost perfectly, while blending together when using only the projection loss.

In general the gaps are bigger on the DAVIS dataset which is expected since DAVIS is of better quality resulting in a stronger supervision signal from the temporal and pairwise loss. At our threshold DAVIS also provides more supervision as seen in Table 3.1. Another interesting observation is that projection loss alone performs relatively well across all the datasets, improving as the dataset becomes more complex. We hypothesize that this is a result from the powerful memory of our underlying model XMem.

The ablations in Tables 4.6 and 4.5 are very similar to the previous two Tables 4.3 and 4.4 but shifted up by a constant. Although the setting is not as useful, given that video masks are used, it shows that the method has potential to scale and that the loss functions are effective at higher levels, being able to provide additional supervision to fill increasingly small details.

5.2 Limitations

A part of our method's efficacy relies on the underlying assumption of continuous and temporal coherence in colors as supervision is calculated pairwise between frame. It is important to note that these are not hard assumptions, but are instead violated on a continuous scale reducing the supervision density. Some scenarios that reduce supervision are abrupt object motions, occlusions and fluctuations in lighting conditions and can be seen in Figure 4.4. When this occurs the supervision relies more on intra-frame losses, such as the projection loss and pairwise loss.

Future work. There remain several promising directions for future research and enhancements to our

model. The most obvious being further training on tracking datasets since we only require bounding boxes during training. Furthermore, exploring how the temporal loss can be extended to use more frames, resulting in more supervision in a computationally efficient way. Exploring even weaker types of annotations is also interesting such as random points inside objects. Conducting a robustness analysis of the projection loss would also be interesting to see the model's sensitivity to annotation accuracy.

5.3 Conclusion

In conclusion, we have presented and validated our weakly-supervised video object segmentation model that supports both box and mask-initialization against traditional fully-supervised methods and various other configurations on datasets such as YouTubeVOS and DAVIS. The results we have obtained serve as a baseline for future research within the domain of weakly supervised video object segmentation. Furthermore we have both quantitatively and qualitatively showcased the affect that different combination of the loss functions imparts, particularly in capturing fine details. We also delved into the practical aspects of our model, demonstrating its efficacy across longer videos containing thousands of frames and also its segmentation speed. Finally, we have explored the flexibility of our approach with image masks and more annotations.

Appendix A

The First Appendix

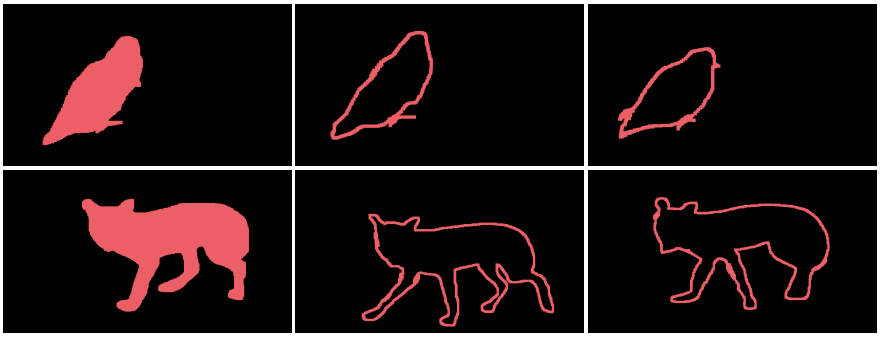


Figure A.1: Examples of erroneous labels in the YouTubeVOS dataset. The left column shows correctly filled masks while the middle and right columns show the examples of when the interior is missing.

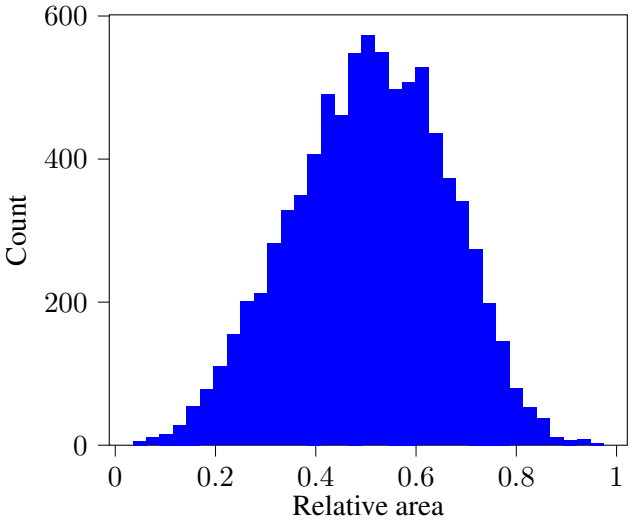


Figure A.2: Histogram of the fraction of bounding box covered by the precise mask. Evaluated on 1000 random videos from the YouTubeVOS training set.

Bibliography

[1] Aditya Arun, C. V. Jawahar, and M. Pawan Kumar. Weakly supervised instance segmentation by learning annotation consistent instances. In *European Conference on Computer Vision*, 2020.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[3] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. Learning what to learn for video object segmentation. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II*, page 777–794, Berlin, Heidelberg, 2020. Springer-Verlag.

[4] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, and Chi-Keung Tang. CascadePSP: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In *CVPR*, 2020.

[7] Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, page 640–658, Berlin, Heidelberg, 2022. Springer-Verlag.

[8] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11781–11794. Curran Associates, Inc., 2021.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[10] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip H. S. Torr, and Song Bai. Mose: A new dataset for video object segmentation in complex scenes, 2023.

[11] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W. Taylor. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *CVPR*, 2021.

[12] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 297–312, Cham, 2014. Springer International Publishing.

[13] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing. Maskrnn: Instance level video object segmentation. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19545–19560. Curran Associates, Inc., 2020.

[15] Lei Ke, Martin Danelljan, Henghui Ding, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Mask-free video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22857–22866, June 2023.

[16] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *CoRR*, abs/1801.00868, 2018.

[17] Viveka Kulharia, Siddhartha Chandra, Amit Agrawal, Philip H. S. Torr, and Ambrish Tyagi. Box2seg: Attention weighted loss and discriminative feature learning for weakly supervised segmentation. In *European Conference on Computer Vision*, 2020.

[18] Z. Lai, E. Lu, and W. Xie. Mast: A memory-augmented self-supervised tracker. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6478–6487, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[19] L. Li, T. Zhou, W. Wang, L. Yang, J. Li, and Y. Yang. Locality-aware inter-and intra-video reconstruction for self-supervised correspondence learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8709–8720, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[20] Yongqing Liang, Xin Li, Navid Jafari, and Jim Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3430–3441. Curran Associates, Inc., 2020.

[21] Yongqing Liang, Xin Li, Navid Jafari, and Jim Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3430–3441. Curran Associates, Inc., 2020.

[22] Fanchao Lin, Hongtao Xie, Chuanbin Liu, and Yongdong Zhang. Bilateral temporal re-aggregation for weakly-supervised video object segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(7):4498–4512, 2022.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.

[25] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[26] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In C.V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 565–580, Cham, 2019. Springer International Publishing.

[27] Wojciech Mokrzycki and Maciej Tatol. Color difference delta e - a survey. *Machine Graphics and Vision*, 20:383–411, 04 2011.

[28] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[29] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *Computer Vision and Pattern Recognition*, 2017.

[30] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[31] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[33] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 629–645, Cham, 2020. Springer International Publishing.

[34] Zhi Tian, Chunhua Shen, Xinlong Wang, and Hao Chen. BoxInst: High-performance instance segmentation with box annotations. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[36] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. *CoRR*, abs/1706.09364, 2017.

[37] Paul Voigtlaender, Jonathon Luiten, and Bastian Leibe. Boltvos: Box-level tracking for video object segmentation. *CoRR*, abs/1904.04552, 2019.

[38] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[39] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3796–3805, 2017.

[40] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. *CoRR*, abs/1812.05050, 2018.

[41] Wenguan Wang, Tianfei Zhou, Fatih Porikli, David Crandall, and Luc Van Gool. A survey on deep learning technique for video segmentation. *CoRR*, abs/2107.01153, 2021.

[42] Ye Wang, Jongmoo Choi, Kaitai Zhang, Qin Huang, Yueru Chen, Ming-Sui Lee, and C.-C. Jay Kuo. Video object tracking and segmentation with box annotation. *Signal Processing: Image Communication*, 85:115858, 2020.

[43] Tianhan Wei, Xiang Li, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. FSS-1000: A 1000-class dataset for few-shot segmentation. *CoRR*, abs/1907.12347, 2019.

[44] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[45] Fei Xie, Wankou Yang, Bo Liu, Kaihua Zhang, Wanli Xue, and Wangmeng Zuo. Discriminative segmentation tracking using dual memory banks. *CoRR*, abs/2009.09669, 2020.

[46] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas S. Huang. Youtube-vos: A large-scale video object segmentation benchmark. *CoRR*, abs/1809.03327, 2018.

[47] Qiong Yan, Jianping Shi, Li Xu, and Jiaya Jia. Hierarchical saliency detection on extended CSSD. *CoRR*, abs/1408.5418, 2014.

[48] Linjie Yang, Yuchen Fan, and Ning Xu. The 2nd large-scale video object segmentation challenge - video object segmentation track, October 2019.

[49] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. *CoRR*, abs/2003.08333, 2020.

[50] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by multi-scale foreground-background integration. *CoRR*, abs/2010.06349, 2020.

[51] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[52] Zongxin Yang and Yi Yang. Decoupling features in hierarchical propagation for video object segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[53] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe L. Lin, and Huchuan Lu. Towards high-resolution salient object detection. *CoRR*, abs/1908.07274, 2019.