Adaptive Memory Management in Video Object Segmentation

Ali Pourganjalikhan

A thesis
in
The Department
of
Computer Science and Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of computer science
Concordia University
Montréal, Québec, Canada

December 2021

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Ali Pourganjalikhan**

Entitled: **Adaptive Memory Management in Video Object Segmentation**

and submitted in partial fulfillment of the requirements for the degree of

Master of computer science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. René Witte	_____	Chair
Dr. René Witte	_____	Examiner
Dr. Tristan Glatard	_____	Examiner
Dr. Charalambos Poullis	_____	Supervisor

Approved _____
Dr. Leila Kosseim, Graduate Program Director

December 10, 2021 _____
Dr. Mourad Debbabi, Interim , Dean
Gina Cody School of Engineering and Computer Science

Abstract

Adaptive Memory Management in Video Object Segmentation

Ali Pourganjalikhan

Matching-based networks have achieved state-of-the-art performance for video object segmentation (VOS) tasks by storing every-k frames in an external memory bank for future inference. Storing the intermediate frames' predictions provides the network with richer cues for segmenting an object in the current frame. However, the size of the memory bank gradually increases with the length of the video, which slows down inference speed and makes it impractical to handle arbitrary-length videos.

This thesis proposes an adaptive memory bank strategy for matching-based networks for semi-supervised video object segmentation (VOS) that can handle videos of arbitrary length by discarding obsolete features. Features are indexed based on their importance in the segmentation of the objects in previous frames. Based on the index, we discard unimportant features to accommodate new features. We present our experiments on DAVIS 2016, DAVIS 2017, and Youtube-VOS that demonstrate that our method outperforms state-of-the-art that employ first-and-latest strategy with fixed-sized memory banks and achieves comparable performance to the every-k strategy with increasing-sized memory banks. Furthermore, experiments show that our method increases inference speed by up to **80%** over the every-k and **35%** over first-and-latest strategies.

We further investigate memory banks' attention during the training by proposing two regularization and studying their effects of performance.

Acknowledgments

First, I would like to thank my supervisor, Dr.Poullis for his guidance through my studies. He encouraged me to explore new areas and guided me with his broad knowledge in computer science, computer vision, and mathematics. My studies coincided with the global pandemic of Covid-19 which created plenty of unprecedented issues for everyone, including me. At each step, Dr.Poullis prioritized my physical and mental health which was heart-warming to come from a supervisor.

Secondly, I must thank my parents, Hamid and Azam, for believing in me and their unbounded support and love which made it possible for me to achieve my goals. Also, I must thank my sister, Helia, for her joyful presence and her love which sustains me through life.

I need to thank Soroush for his insights which helped me to broaden my perspective and make better decisions. Also, I am greatly thankful to all members of the Immersive and Creative Technologies (ICT) lab especially Amin for the amazing time we had together.

Lastly, a big thank you to Naghmeh for her love and support which helped me overcome challenges in my academic and personal life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in technology have made videos a crucial medium to transfer information and content. Abundant usage of videos has made video processing a critical step to understanding world, using computer. A fundamental task in video processing is video object segmentation (VOS). A dense segmentation mask can facilitate other tasks such as object tracking and video editing.

This work focuses on semi-supervised video object segmentation, and in this chapter, we discuss our motivation, contributions and provide the outline of this thesis.

1.1 Motivation

The outstanding advances of computers and the availability of annotated datasets have enabled us to design and train complex models with an enormous number of parameters. However, these approaches are far from real-world applications.For instance, current memory bank architectures and management methods are designed to perform on benchmark datasets and fail in real-world applications due to excessive use of memory and computation power. In this work, we focus on the limitations of memory banks to handle videos of arbitrary length and propose methods to alleviate this problem. An efficient and accurate VOS can be used as a foundation for more advanced tasks and a step toward better understanding of videos.

1.2 Contributions

Our contributions are two folded:

- We propose a novel memory management approach to maintain a fixed-size memory while handling videos of arbitrary size. Our method yields comparable results with state-of-the-art methods with faster inference speed and fixed-size memory.
- We investigate the limitations of current memory block and study two memory regularization methods and their effect on network performance.

1.3 Thesis Outline

The rest of this thesis is organized as follows:

In **Chapter 2**, we provide the background of object segmentation, attention methods, and metrics used to evaluate that are relevant to our work. This work assumes readers have a fundamental knowledge of neural networks.

In **Chapter 3**, we describe the current semi-supervised VOS approaches' inability to process videos of arbitrary size and propose an adaptive memory management method to alleviate this problem. This chapter was submitted as a regular paper to the IEEE/CVF Computer Vision and Pattern Recognition (CVPR) 2022 conference.

In **Chapter 4**, we investigate two memory regularization methods and study their effect on the training. Additionally, we provide supplementary materials for chapter 3, such as training details for completeness.

In **Chapter 5**, we give a conclusion to our work and discuss possible future avenues to explore with the help of our proposed methods.

Chapter 2

Background

This chapter provides a brief overview of the most relevant concepts and methods to understand the rest of this thesis.

2.1 Architectures

This section assumes a basic knowledge of neural networks and, specifically, convolution neural networks. Hence, we only cover a specific topics that are necessary to understand this paper.

2.1.1 ResNet

Resnet [14] is one of the most groundbreaking architectures, which led to advances in different fields of computer vision. The representation ability of Resnet makes it a reliable backbone to extract features in many computer vision applications.

Based on the universal approximation theorem, a single-layer feed-forward network with enough capacity can represent any function. However, in practice, a wide feed-forward network will lead to overfitting on training data. The most practiced approach to alleviate this issue is to increase the depth of the neural networks.

Deep neural networks produce better results with greater generalization ability. However, training deep neural networks has its limitations. The backpropagation process involves calculating the gradient at each layer of the neural network. As networks get deeper, gradients start to vanish.

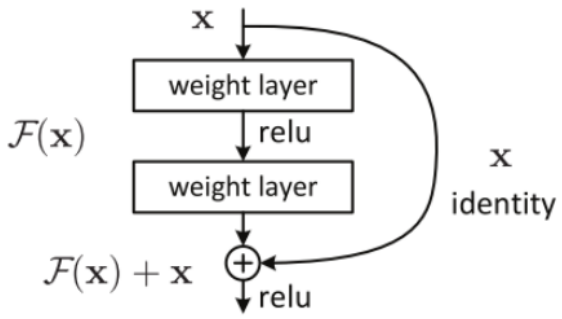

Figure 1: An overview of Residual Block. Figure courtesy of [14].

Residual blocks proposed in Resnet[14] try to mitigate this issue by introducing the identity shortcut connection. Figure 1 shows Residual block architecture. The basic rationale behind this architecture is that the added layer should not degrade the performance. As a result, added layers should be able to perform as an identical map between input and output. The residual block improves gradient flow in deep neural networks and facilitate the training process.

Resnet networks pretrained on Imagenet[9] can perform as a feature extractor with great generalization ability for other computer vision tasks.

2.1.2 U-Net

Encoder-decoder networks are common for image segmentation, and among them, U-Net [42] has gained popularity as it is simple and effective. We discuss U-Net as an exmaple of encoder-decoder network to help us better understand the architecture and concepts of this family of networks.

An overview of U-Net architecture is shown in figure 2. An input image is first processed through a multi-stage contraction path. At each stage, features are processed by a sequence of convolution layers with ReLU non-linearity, followed by a pooling layer to reduce spatial dimensions. Convolution layers at each stage are padded to have an input and output with the same spatial size. Furthermore, the number of channels is doubled at each stage. After the contraction path, input image information is embedded with low spatial resolutions at the bottleneck. The expansion path follows the contraction path. Similar to the contraction path, the expansion path is multi-stage and gradually increases image embedding resolution to reach the final

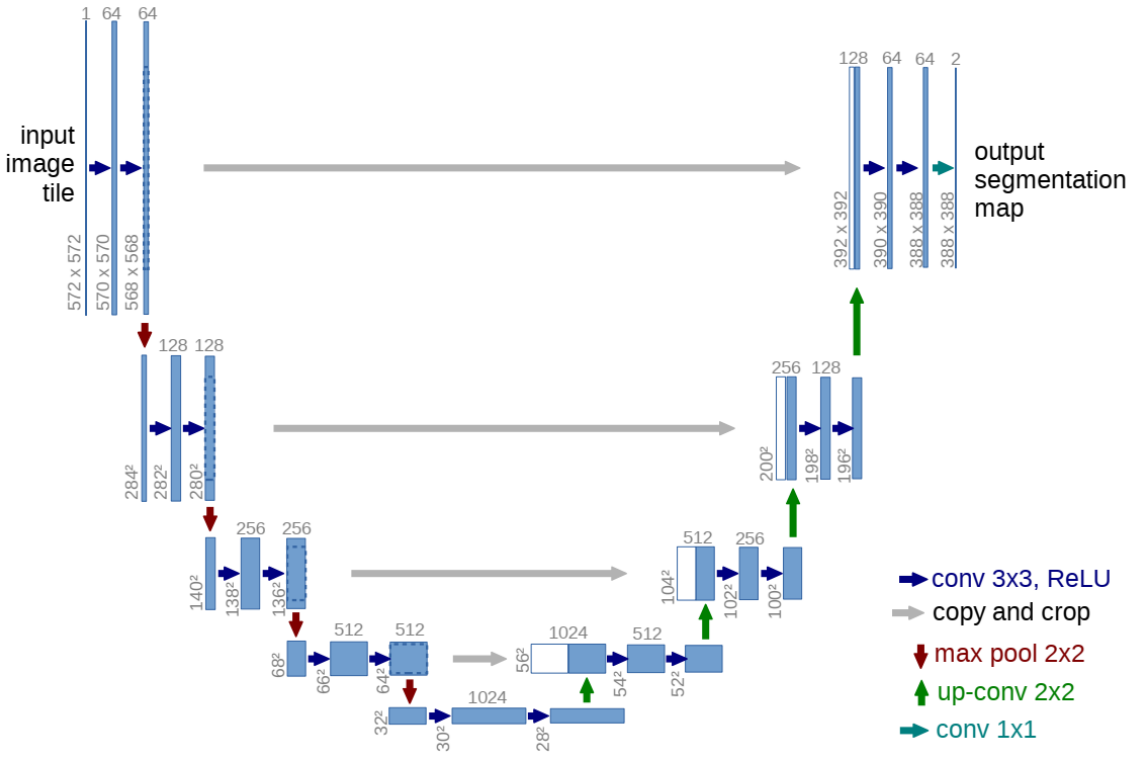

Figure 2: The architecture of U-Net with 572×572 input. Figure courtesy of [42].

segmentation mask. For each stage of the expansion path, the input is the output of the corresponding level in the contraction path before pooling, concatenated with the output of the previous level in the expansion path. At each level of the expansion path, the spatial dimension is doubled using upsampling, which can be inaccurate. Hence, combining the previous levels' output with the corresponding encoding levels helps the network achieve better results.

2.2 Evaluation metrics

To evaluate the quality of segmentation, the choice of metrics needs to be tailored to the end goal of the method [40]. In a semi-supervised video object segmentation setup, a per-frame segmentation mask is a binary classification problem between background and foreground. A global accuracy metric is unreliable when dealing with unbalanced data, which is the case in per-frame background-foreground segmentation.

In the following, we discuss two metrics to evaluate the quality of the segmentation. Region similarity evaluates the overall quality of the segmentation while Contour

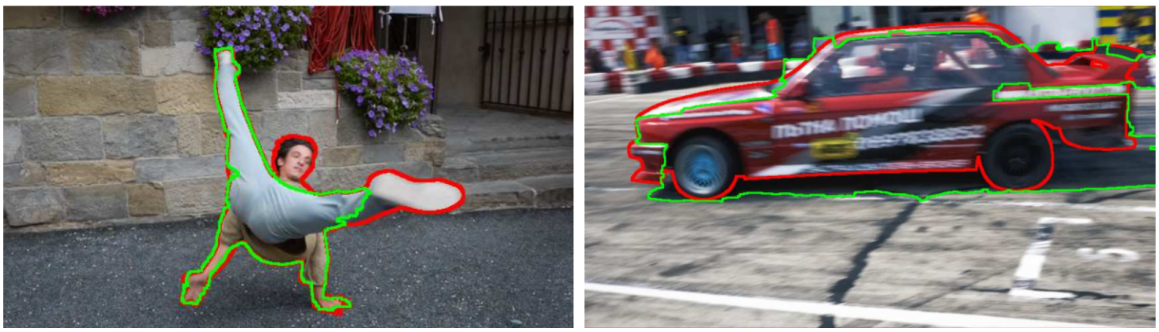

Figure 3: In these figures, the red contour shows the ground truth, and the green contour represents the predicted mask. **Left:** Predicted mask is missing major parts of the object. In this case, the mislabeled area is penalized by region similarity(J). **Right:** In addition to mislabeled regions, predicted boundaries are inaccurate. Hence, Contour accuracy (F) will penalize the predicted mask. Images are courtesy of [40].

accuracy focuses on boundary accuracy. Although these metrics are intuitively similar, [40] shows the degree of Independence is enough to justify using both of them. In figure 3, we show how these metrics have a different responses to segmentation mask.

2.2.1 Region similarity(J)

To measure the number of mislabeled pixels, [40] proposed Jaccard Index(JI):

$$JI = \frac{TP}{TP + FN + FP}$$

The Jaccard Index can also be defined as the intersection-over-union(IOU) between the ground truth mask and the predicted mask. IOU has been widely used as it provides scale-invariant information about the number of mislabeled pixels in the segmentation. IOU between predicted mask P and ground truth mask G can be formulated as:

$$J = \frac{|P \cap G|}{|P \cup G|}$$

2.2.2 Contour Accuracy (F)

In addition to Region similarity, contour accuracy is equally vital in applications such as video editing, where accurate boundaries are essential to the final output quality.

Although the Jaccard Index heavily penalizes mislabeled regions, it pays little attention to boundaries. To handle this shortcoming, [40] proposes contour accuracy

in addition to region accuracy with a focus on boundaries accuracy. Prediction mask P and ground truth mask G can be interpreted as a set of closed contours $C(P)$ and $C(G)$. Having the prediction and ground truth contours, one can measure the precision and recall. Precision is the fraction of predictions that are true-positive rather than false-positive, and recall is the fraction of true-positive rather than false-negative predictions.

Before measuring the precision and recall between $C(P)$ and $C(G)$, [40] applies dilation to expand the edges of both ground truth and prediction to compensate small inaccuracies. In an unbalanced setup such as this, precision and recall are better indicators of a classifier's quality than global accuracy. To demonstrate precision and recall in one metric, [40] uses *F-score* which is a harmonic mean between precision P and recall R.

$$F = \frac{2PR}{P + R}$$

2.3 Self-attention

Self-attention, first introduced in [47], is trying to mimic cognitive attention in humans, which is concentrating on relevant stimuli and disregarding irrelevant information. In other words, self-attention attempts to enhance an important part of the data while fading out irrelevant parts.

After introducing self-attention mechanism and transformers by [47], it became a popular approach in the machine translation, and Natural language Processing (NLP) community and researchers adapted it to handle different challenges.

After the great success of self-attention in the NLP domain, researchers started to expand the self-attention concept into the spatial domain. In this section, we discuss the self-attention module and various self-attention adaptations in the computer vision domain.

2.3.1 Multi-head self-attention

Multi-head attention was first proposed in [47] and widely adopted by other researchers in different areas. Multi-head attention is a combination of several attention modules that are executed in parallel. To better understand multi-head attention, we first look at the attention block.

For instance, between vectors $x1$ and $x2$, $x1$ is the attendee and generates a query, while $x2$ is the attendant and generates a key-value pair. The objective is to map the query and key-value pair to an output. In self-attention, the output is a weighted sum of values where weights are assigned a value based on the affinity of the query to the corresponding key. The most common affinity function is the dot-product affinity. A self-attention can be expressed as follows:

$$F(Q, K, V) = Softmax(QK^T)V$$

In transformers [47], multiple attention blocks are concatenated together and followed by layer normalization and fully connected layer to form a multi-head attention block. [37, 10] show that with proper modifications, multi-head attention can improve performance on computer vision applications.

2.3.2 Squeeze-and-Excitation(SE)

One of the main characteristics that make the convolution layer stand out in image processing is its attention to the locality of information. This assumption holds when extracting low-level features from an image. However, as the network gets deeper, extracted features become more high level, and attention to the global context of the image can improve the quality of extracted features using the convolution layer alone.

Squeeze-and-Excitation [15] is a simple yet effective attention mechanism that utilizes global information to model inter-dependency between the channels of a feature embedding. Involving global information in the process of the model decision has shown to be helpful in various applications [15, 5]. Squeeze-and-Excitation is done by sequentially applying the squeeze and excitation operation to a feature map. An overview of this mechanism is shown in figure 4.

The Squeeze-and-Excitation process starts by applying a transformation F_{tr} on X to reach U. The transformation is then followed by a squeeze operation F_{sq}. The squeeze operation is simply an average pooling or max-pooling along the spatial dimension. Squeeze F_{sq} output is the input to the excitation F_{ex} which is a two-layer fully connected network with a ReLU [1] activation function after the first layer and a Sigmoid activation after the second layer to ensure outputs are in the range $[0, 1]$. Finally, excitation output is the self-attention weights that emphasize the importance of specific channels and is used to reweight U channels. The mathematical expression

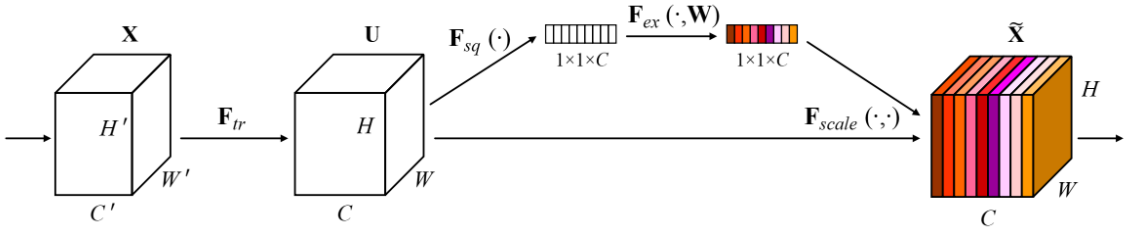

Figure 4: An overview of Squeeze-and-Excitation mechanism. Figure is courtesy of [15].

of Squeeze-and-Excitation is shown below:

$$
\begin{aligned}
U &= F_{tr}(X) \\
Z &= F_{sq}(U) = \frac{1}{HW} \sum_{H} \sum_{W} U_{:,ij} \\
\hat{Z} &= F_{ex}(Z, W) = \sigma(W_2 \delta(W_1, Z)) \\
\hat{X} &= F_{scale}(Z, U) = \sum_{C} Z_i U_{i,:,:}
\end{aligned}
$$

2.3.3 Non-Local

Squeeze-and-Excitation(SE) block can effectively involve global information in the network's decision. However, these methods fail to handle long-term dependencies in space-time dimension.

Recurrent operations [13, 53] are the most successful in capturing long-term dependency in sequential data. Recurrent operations are applied locally to the neighbourhood to capture long-term dependencies and need to be applied multiple times to handle long-term dependencies. This phenomenon makes recurrent approaches computationally inefficient and difficult to optimize [13, 53].

The equation for non-local operation is shown below.

$$
y_i = \sum_{j} f(x_i, x_j) g(x_j)
$$

f is the pairwise distance between corresponding input to the location of output x_i and every other input x_j, and $g(x_j)$ is the representation of the input at position j.

In video processing, a non-local operation disregards the temporal order of frames and treats all frames equally, and can benefit from past frames' information without forgetting through time. The downside to this approach is computational inefficiency

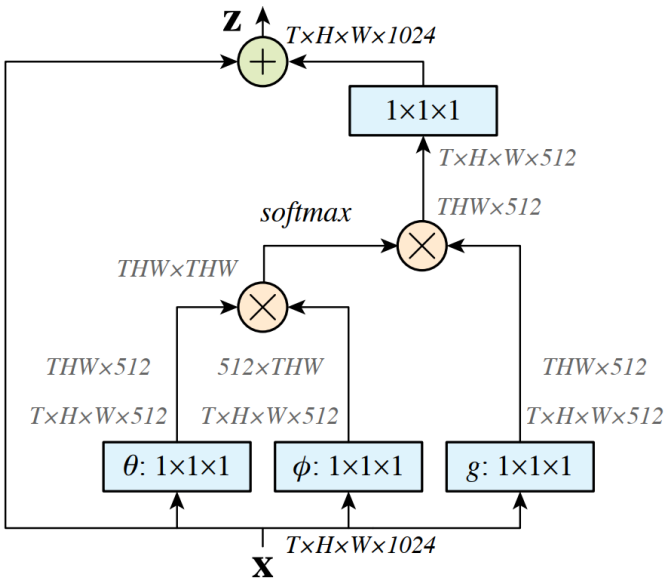

Figure 5: An Instantiation of Non-local mechanism with the input x of the size of $T \times H \times W \times C$ where T is number of frames processed at a time and the number of channels C is 512. Before computing pairwise affinity f, the number of channels is reduced via 1×1 convolution. Channel reduction is common to change the representation of input x and decrease the required computation to calculate pairwise affinity and softmax. Figure courtesy of [53].

to process long-term videos as the amount of computation necessary to process the input x grows by $O(n^2)$ to the size of input x. Figure 5 shows an instantiation of the Non-local operation.

Chapter 3

Adaptive memory managment in video object segmentation

Under review, "Adaptive Memory Management for Video Object Segmentation", A. Pourganjalikhan, C. Poullis

3.1 Introduction

Video object segmentation (VOS) is a fundamental computer vision task with many applications in self-driving cars [2], augmented reality [33], video editing [3], and many other video-related tasks. Additionally, video object segmentation serves as a building block in tasks such as interactive video object segmentation [7, 35, 30, 25] and video instance segmentation [11, 57].

In VOS, the target objects are annotated in their first appearance, and the objective is to segment them in subsequent frames. Early attempts on VOS [4, 49, 28], fine-tuned the network to learn the target objects' appearance. Fine-tuning a deep neural network with only one example (one-shot) at the target is challenging [12, 58]. Moreover, fine-tuning makes inference slow which makes it unsuitable for real-time applications.

Recent works [36, 48, 60], instead of learning object features implicitly, use a learned embedding space to embed and memorize object appearance and use that embedding to segment object in the subsequent frames by calculating affinity between current and past frames embeddings.

A key challenge in matching-based VOS is exploiting the previous frames' information. Using all previous frames' information is impracticable and redundant. Most recent works [34, 48, 60] rely only on the first-and-latest frame. The latest frame is visually close to the current frame, and the first frame provides reliable annotation for the object, avoiding drifting during the segmentation. This strategy disregards all intermediate frames' information.

To address this, Space-Time Memory (STM) network [36] employs a memory bank to store every-κ frame for subsequent inferences. While this method utilizes intermediate frames, information of the memory bank grows linearly as a function of the number of frames κ. Hence, it imposes significant memory requirements that prevent the processing of longer video sequences. The solution is to remove features from the memory bank when they become obsolete, ensuring a fixed-sized memory bank that allows the processing of videos of arbitrary length.

In this paper, we propose Least-Frequently-Used (LFU) feature removal based on the top-k index. Our method outperforms the first-and-latest strategy and achieves comparable results with the every-k sampling strategy while using a smaller, fixed-sized memory bank. In addition to handling arbitrary-length video sequences without imposing significant memory requirements, our method considerably improves the inference speed when compared to the top two sampling strategies, first-and-latest and every-k.

The main author, Ali Pourganjalikhan, formed the ideas, designed the algorithms and implemented the system, conducted the experiments, and led the writing of the scientific publication. The contributions of co-author, Charalambos Poullis, include manuscript review, supervision, and technical support.

3.2 Related Work

In this section, we review the most relevant works in video object segmentation.

3.2.1 Semi-supervised video object segmentation

Following the taxonomy proposed by [26], recent VOS methods can be categorized into implicit and explicit according to the approach followed to address the problem.

Implicit models aim at learning objects' representation by fine-tuning network

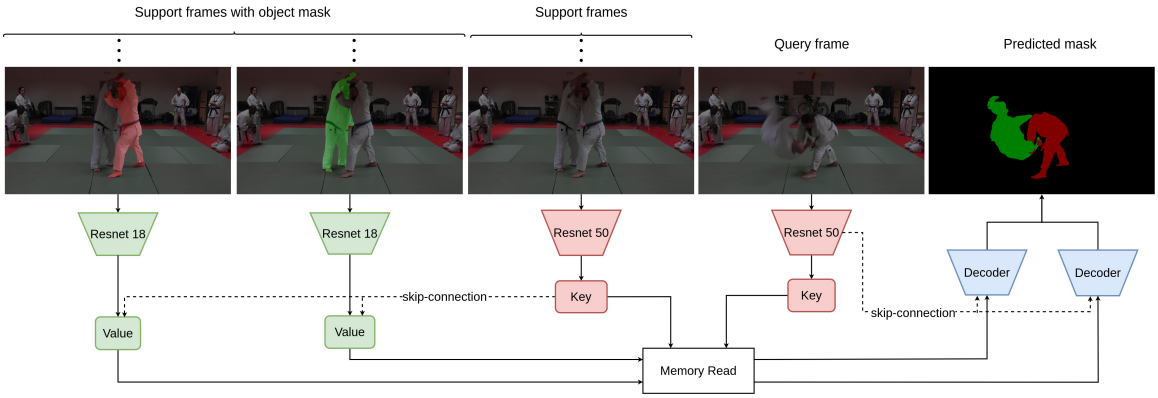

Figure 6: Overview of the network architecture. Parts with the same colour have shared weights. We illustrate the memory bank architecture in detail in Figure 7. Deep features learned by the key encoder are concatenated to the value encoder's input. The output at different levels of the key encoder is used to refine the mask at different scales of the decoder. We store the query key of specific frames for further inference. In the case of multiple objects, soft-aggregation is used to reach the final output.

weights for each object. Detection-based implicit methods such as [29, 4] segment objects in each frame independently without enforcing temporal consistency between consecutive frames. Propagation-based implicit methods such as [39, 17, 18, 23] propagate objects' masks between consecutive frames.

Implicit models mainly rely on online learning to adapt to different objects. The online learning process makes these methods very time-consuming during the inference and reduces the network's ability to handle deformation. On the other hand, explicit models use a fixed set of parameters for all sequences during the inference. The features of the object from the previous frames are used to classify the current frame pixels as foreground or background using similarity matching; hence often called matching-based methods as well. A matching-based network uses different sources of information. [34, 48, 54, 27] utilize information from first-and-latest frames to segment the object in the current image. [36] uses a memory bank to store the object representation in intermediate frames. Despite the difference in architecture and sources of information, when it comes to incorporating previous frames' information into the current frame segmentation process, most of the approaches use cosine or Euclidean distances to find the affinity between previous frames and current frames features.

3.2.2 Memory Banks

Networks can learn to read and write helpful information in external memory. For example, end-to-end memory networks have proven useful for document Q&A [20, 31, 46], visual tracking [59], video understanding [32], and summarization [21].

STM [36] encodes each image into the key-value pair and uses them as cues to segment the target object. In their framework, past frames are in memory, and the current frame forms the query. Using the keys, they estimate the affinity between memory and query frame to determine which memory values should be used during segmentation. Space-Time Correspondence Networks (STCN) [8] uses object-agnostic keys, which reduce the computation in multi-object scenarios.

STM [36] and its extensions and variants [43, 24, 16, 55, 50, 62, 63] store intermediate frames' key-value into a memory bank. The efficiency of the memory bank depends on the number of key-value pairs that can be stored. STM [36] adds features for every-k frame ($k = 5$ in their paper) in the memory bank.

Although increasing the number of frames in the memory bank improves performance, MiVOS [7] showed that only a few of the memory features meaningfully contribute to the segmentation process. Instead, MiVOS uses only the k-closest memory features for each query feature and discards the rest as they adversely affect the performance. Although MiVOS [7] ignores redundant features during inference, it still maintains these features in the memory bank for future inferences.

[26] propose an update and remove strategy to keep the memory bank size fixed. When adding new features to the memory bank, they perform an eligibility check for an update by ensuring that the feature's distance to its closest neighbour is lower than a threshold. The update is a running average between new features and their closest neighbour in the memory. To remove an obsolete feature that can not be merged, they keep track of its frequency being close to a query feature and then use a least-frequently-used strategy to remove it. Although [26] keeps the size of the memory bank fixed, both update and removal steps use a threshold that is dependent on an affinity metric. The affinity metric for this approach needs to be bounded since the threshold needs are decided beforehand.

3.3 Methodology

Given a sequence of frames and the masks of the target objects in their first appearance (objects typically appear in the first frame; however, new objects can also appear in the middle of the sequence), we segment the object in the rest of the frame sequence. This problem is a variant of one-shot image segmentation [44] in which the current frame is the **query** and the past frames with segmented objects mask are the **support** set. Figure 6 shows an overview of the proposed technique. We have chosen STCN [8] as the baseline since it is effective and minimal. Although we evaluate our method on STCN, the feature sampling strategy can be used for any matching-based network with a memory bank. For each query and support frame, we extract a key that is independent of the object. The affinity between the query and support keys is then used to select the corresponding support values for segmentation.

3.3.1 Encoding of key-value

Unlike STM [36], STCN[8], we use object agnostic key encoding with shared weights for support and query. We employ a Resnet50 [14] followed by a 3×3 convolution layer as a projection layer to encode the key. Spatial information is preserved by using the output of *res4* layer for the projection. The projection layer reduces the number of channels from 1024 to 64. The query key can be reused as a support key since the query and support key encoder have shared weights.

Value encoding is performed more frequently; therefore, we use a lighter backbone. We employ Resnet18 to encode the image and mask of an object. Unlike key encoding, value encoding is object-specific and only used for support frames. The output is then concatenated with the corresponding feature map from the key encoder and processed by two residual blocks. In this way, the network can use deeper backbone feature embeddings without any overhead.

We initialize both key and value encoders with pre-trained weights from Imagenet [9]. In the value encoder, the input consists of an image and a corresponding mask. Hence, we modify the first layer of the Resnet18 to have 4-channels and initialize the additional new weights to zero.

3.3.2 Memory read

A memory read is a visual attention operation to reconstruct the support value with respect to the affinity between the support key and the query key. Figure 7 shows an overview of our memory read module. After encoding, the key-value pairs (K^S, V^S) of the support frames are concatenated to form a space-time key-value. A memory read operation starts by calculating the affinity between the support key $K^S \in \mathbb{R}^{THW \times C^K}$ and the query key $K^Q \in \mathbb{R}^{HW \times C^K}$. T refers to number of frames in the support set. The affinity d_{ij} between a support feature K_i^S and a query feature K_i^Q is based on the negative squared distance and is given by

$$d_{ij} = \frac{dist(K_i^S, K_j^Q)}{\sqrt{C^k}} \tag{1}$$

where $dist(.,.) : \mathbb{R}^{C^k} \times \mathbb{R}^{C^k} \longrightarrow \mathbb{R}$ is the negative squared distance i.e. $-\|(K_i^S - K_j^Q)\|_2^2$. Similar to [36, 8], the affinity d_{ij} is divided by $\sqrt{C^k}$.

The affinity matrix $D \in \mathbb{R}^{THW \times HW}$ is then normalized along the dimension of the query features using Softmax and is used to calculate $D \in \mathbb{R}^{THW \times C^V}$ as the weighted summation of the support value.

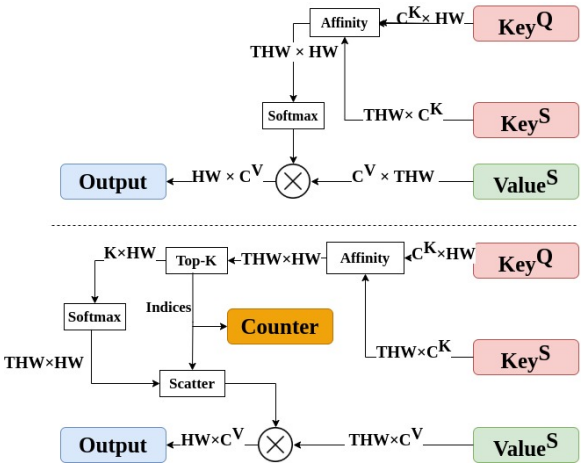

Figure 7: **Top:** Memory read block during training. H and W are the *res4* layer dimensions which have a stride of 16 on the input. **Bottom:** Memory read block during inference. After finding the affinity between query and support keys, we find the k-closest features for each query feature. The counter module keeps track of the support features' appearance in top-k.

3.3.3 Decoder

The decoder extends the work in STM [36]. It contains two consecutive refinement modules [34] which upsample the features' spatial size by four and reduce the number of channels from 1024 to 1. The output of the decoder is then upsampled by four to match the size of the input. At each step, we concatenate the features of the query key encoder with the input through skip-connections. In the first stage, we reduce the number of channels of skip-connections to match the number of channels of memory read output using a 3×3 convolution layer. In the case of multiple objects, soft aggregation [36] is used to reach the final mask.

3.3.4 Memory bank

During the inference, we use a memory bank to store the support frames' information. The size of the memory bank can grow with the length of the video. Storing all previous frames imposes significant memory requirements and slows down performance during inference.

To address this limitation, STM [36] suggested storing only the first-and-latest frames to maintain a fixed-size memory bank. However, experiments show that using intermediate frames improves the performance of the model. To benefit from intermediate frames, STM [36] stores every k frame ($k = 5$ in their experiments) into the memory bank. On an NVIDIA 1070 with 8G of memory, STCN [8] can handle only 100 frames in the memory bank, which is equivalent to a 15-second *story* on Instagram. In order to handle long videos, streams with arbitrary length, or to use embedded devices with limited resources, the first-and-latest strategy offers the best solution. Figure 8 shows different memory management strategies.

MiVOS [7] shows that after applying softmax to the affinity matrix between support and query key, the weights for most of the support features become small. Feature with small weights does not meaningfully contribute to the segmentation process. This phenomenon amplifies as the number of support features increases. MiVOS [7] shows that disregarding non-contributing features of the support set leads to more stable segmentation through time.

To the best of our knowledge, only the Adaptive Feature Bank (AFB) presented in [26] attempts to address this problem. However, this method uses a user-defined threshold which is data-dependent and requires adjustment for different videos. More

importantly, it supports only dot product or cosine similarity as a distance metric which, as shown in [8], are outperformed by the negative Euclidean distance.

We propose storing only the top-k features by updating the memory bank and removing obsolete features to overcome these limitations. The advantages are three-fold: it eliminates the threshold requirement, results in a constant fixed-sized memory bank, and is agnostic to the distance metric used.

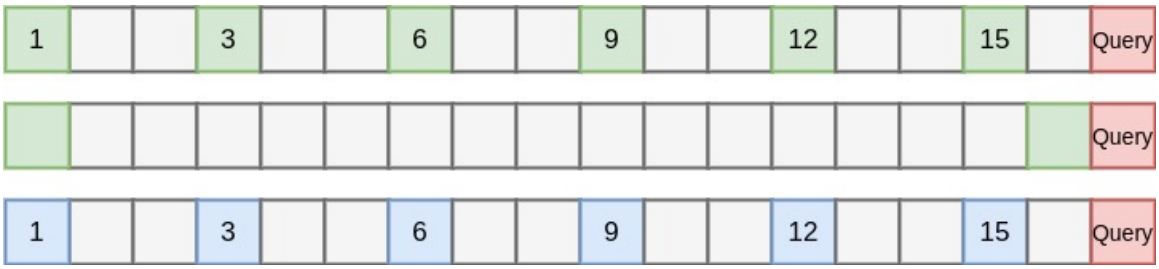

Figure 8: Various memory management strategies. **Top:** Storing every-k frames into the memory. The size of the memory bank grows linearly with the length of the video. **Middle:** Storing first-and-latest frames into the memory. Although this method uses a fixed-size memory bank, it disregards all intermediate frames. **Bottom:** We store every-k frames into the memory. However, we remove obsolete features from the memory to maintain a fixed-size memory bank.

Storing the top-k

As the number of features in the memory bank grows, only a fraction of those features will continue to be relevant and hence will have non-zero values after the softmax operation [7]. Additionally, frames that are temporally closer to the query frame are more likely to be visually similar. From the many cache management algorithms, the least-frequently-used (LFU) policy defined as $LFU = \frac{index}{age}$ is the most suitable for this task. Calculating the LFU score requires the index and the age. The index is calculated as the number of times that the feature has been referenced, i.e. the number of times it appeared in the top-k features matching a query feature. The age is the time and is calculated in terms of the number of frames that the feature has been in the memory.

Unlike [25], no updates are performed when adding a new frame to the memory. During removal, we remove enough features to make accommodate new features.

Methods	J-Mean	J-Recall	J-decay	F-Mean	F-Recall	F-decay	J&F-Mean
DAVIS 2016 validation set							
Every 5	90.4	98.1	4.1	93.0	97.1	4.3	91.7
first-and-latest	88.0	95.3	8.3	91.0	94.4	8.4	89.5
Ours	89.5	97.6	6.1	92.3	96.8	5.8	90.9
DAVIS 2017 validation set							
Every 5	82.0	91.3	6.2	88.6	94.6	8.6	85.3
first-and-latest	79.5	90.4	10.8	85.4	92.9	14.0	82.5
Ours	81.3	90.4	8.1	87.4	92.9	10.7	84.4

Table 1: Quantitative results on DAVIS 2016 and DAVIS 2017 validation sets. Evaluation is performed with J(Intersection-over-Union) and F(boundary accuracy) metrics. Mean is the average of individual objects J and F. Recall measures objects scoring higher than a threshold ($\tau = 0.5$ following the official benchmarks [40]) in a sequence. Decay indicates the performance drop between the first-and-latest quartile of a sequence.

Methods	J-seen	J-unseen	F-seen	F-unseen	J&F
every 5	82.6	79.3	86.9	87.6	84.1
First-and-latest	79.9	75.1	83.9	83.2	80.5
Ours	80.3	76.1	84.3	83.8	81.1

Table 2: Quantitative results on Youtube-VOS validation set. Final *J&F* score is unweighted average between seen and unseen metrics.

3.4 Implementation Details

We followed the training procedure in [8] as training is not the main focus of our work. As suggested in [8, 36, 7], we used two-stage training. First, we train the model on static images [61, 6, 22, 52, 45] with augmented deformations for 300,000 iterations with batch size of 16.

For the next stage, we use Youtube-VOS [56], and DAVIS [40, 41] to train the network for 150,000 iterations with a batch size of 8. In this stage, at each iteration, we pick three temporally ordered frames. Following [36, 60], we use the first frame to segment the second frame and use second frame predictions and the first frame to segment the third frame.

We used Adam [19] as optimizer, bootstrap crossentropy [7] as loss function and used 4 *P100* GPUs to train the model which took 5 days to complete. We used Pytorch [38] as a deep learning framework.

For inference, we used a *GTX1070* GPU and re-time STCN [8] to ensure fair

comparisons in our experiments. Given that support keys are object-agnostic, we initialize a single index and age counter at the beginning of the sequence, which is shared by all objects. In Youtube-VOS dataset [56], objects not always appear in the first frame. Using LFU, our method can successfully handle new objects appearing in the middle of the sequence.

3.5 Experiments

We evaluate our approach on DAVIS 2017 [41] validation set, and Youtube-VOS [56], two large-scale benchmarks with multiple objects in videos. In DAVIS 2017, all target objects are present in the first frame of the sequence. However, they can be occluded at the beginning or disappear and reappear in the middle of the sequence. In Youtube-VOS, some of the target objects first appear in the middle of the video, and the objective is to start tracking that object from that point onwards.

For evaluation, we use $J\&F$ from the DAVIS benchmark, which is an average between the region accuracy J and the boundary accuracy F. For each object, we calculate $J\&F$ score in each frame separately, and the object score is the mean of its score in different frames. The overall score is an average of each object score. This method prevents big objects or objects with more extended visibility from skewing the results.

We evaluate our approach with every-k and first-and-latest methods. In all experiments, we ensure a fair comparison with the first-and-latest approach by setting the size of the memory bank to two frames worth of features. We compare our method with first-and-latest memory utilization since it is the sole available method that can handle videos of arbitrary length. Additionally, we compare our approach with every-k since it is the best performing approach for memory write. Results show that our approach outperforms first-and-latest and has comparable results with the every-k method. We also investigate the memory utilization and inference speed of different approaches. Our method has better inference speed with minimal memory utilization.

3.5.1 Results on DAVIS 2016

DAVIS 2016 [40] validation set contains 20 videos with dense masks for single objects. A quantitative comparison of our approach with other methods is shown in Table 1. Our network outperforms the first-and-latest method by **1.4 %**. To evaluate the network's ability to handle multi-object scenarios closer to real-world applications, we next present the results on DAVIS 2017.

3.5.2 Results on DAVIS 2017

DAVIS 2017 is a multi-object extension of DAVIS 2016 dataset. It contains 120 videos that are 30 times smaller than the videos in the Youtube-VOS dataset. The validation set has 59 objects in 30 videos. The results on DAVIS 2017 are shown in Table 1. Our method outperforms the first-and-latest method by a significant margin and has comparable results with every-*5* method while using a smaller memory bank.

	Every 5	First-and-latest	Ours
Frame per Second	6.3	8.4	11.4
Memory utilization	7	2	2

Table 3: Comparison of Inference speed and memory utilization of different memory management strategies on DAVIS 2017 validation set. Inference speed shows the number of frames processed in a second in a multi-object video. Memory utilization shows the number of frames stored in the memory bank. For the every-5 method, we used the average number of frames in the memory. In our method, we set the size of the memory bank to be equal to 2 frames worth of features.

3.5.3 Results on Youtube-VOS

Youtube-VOS is the latest large-scale dataset for VOS. The training set has 3471 videos with 65 different object categories. The validation set has 507 videos with 26 unseen object categories and the object categories of the train set. The availability of unseen categories makes Youtube-VOS suitable to measure the generalization ability of the various methods in question.

Results for Youtube-VOS are shown in Table 2. Our method outperforms the first-and-latest method and underperforms when compared with every-k approach. Since the average length of videos for Youtube-VOS is longer than DAVIS2017, using

the same every-k setup, the Youtube-VOS causes increased memory use; it results in storing a more significant number of frames in the memory bank, which consequently makes it harder to compete with using only two frames worth of features.

3.5.4 Inference speed

The size of the memory bank directly affects the amount of computation in the memory read block. By limiting the size of the memory bank, the proposed method can increase the inference speed by 80 %. A comparison of inference speed and memory utilization is shown in Table 3.

The first-and-latest approach stores the latest frame into the memory at each step which slows down the speed. On the other hand, our method can be described as an extension to the every-k meaning that we only perform a memory write operation every few frames. Having fewer memory write operations leads to an increase in inference speed by 35%. To calculate the inference speed, we measure the total processing time on the whole DAVIS 2017 [41] validation set and divide it by the total number of frames.

Methods	DAVIS 2017			Youtube-VOS		
	J-Mean	F-Mean	J&F-Mean	J-Mean	F-Mean	J&F-Mean
Softmax storing	79.9	82.2	83.0	76.2	81.8	79.0
Top-k storing	81.3	87.4	84.4	78.2	84.0	81.1

Table 4: Ablation on top-k storing versus softmax weights storing

3.5.5 Qualitative results

Qualitative results are shown in Figure 9. The first six rows show how different approaches handle multiple objects exhibiting deformation and significant displacement due to motion. As we do not apply any spatial constraint to the segmentation, the network can handle significant displacements successfully. However, in the case of deformation, the object's appearance in the initial frames can quickly become distant -in metric space- from that in the current frame. In this case, the network relies heavily on the mask propagated from the previous frame. This makes the network prone to accumulating errors over time. On the other hand, our strategy can handle

deformation and recover from erroneous previous predictions by removing obsolete features from memory.

The bottom three rows in Figure 9 show the case of complete occlusion when tracking multiple objects. Given the object-agnostic nature of our baseline, we are using the same index counter for all of the objects. During an object's occlusion, features become inactive. As a result, they have a higher probability of being discarded from memory, and in the case they reappear later in the video sequence, the network may not track the object again.

3.5.6 Discussion

We analyzed the effectiveness of using top-k as an index. To do so, we remove the top-k storing from the memory bank and instead use softmax weights as the counter index. A comparison between the two approaches is shown in Table 4.

Even among the top-k features, softmax normalization is highly imbalanced toward the closest features, which score a high probability. This imbalance diminishes the network's ability to discard features that used to be deterministic but lost their effectiveness. Figure 10 shows a qualitative comparison between top-k and softmax weights used as the index. As seen, our method can handle the deformation and complex motions of objects.

Memory bank size. We further investigate the effect of memory bank capacity on model performance. For this purpose, we gradually increase the size of the memory bank and evaluate the performance of the network on Youtube-VOS [56] validation set. We use Youtube-VOS [56] since it has larger validation set with longer videos. Results are visualized in Figure 11.

Looking at the results, increasing the memory bank's capacity leads to a better $J\&F$ score. More importantly, only by doubling memory bank capacity from 2 to 4 frames worth of features, the network's $J\&F$ measure increases by 2%. The results show that using a memory bank with a capacity of four frames worth of features is the best trade-off between inference speed and accuracy for the Youtube-VOS dataset.

3.6 Conclusion

We presented a memory management strategy for semi-supervised video object segmentation. We employed a least-frequently-used(LFU) policy using the top-k index. Extensive experimentation on DAVIS 2016, DAVIS 2017, and Youtube-VOS demonstrates that our method outperforms fist-and-latest strategies with a fixed-sized memory bank and achieves comparable results with every-k strategies with an increasing-sized memory bank. Unlike state-of-the-art every-k methods, ours handles videos of arbitrary length with no additional overhead, which is crucial for real-world applications. Furthermore, our method facilitates video object segmentation of arbitrary-lengthed video streams under limited computational resources.

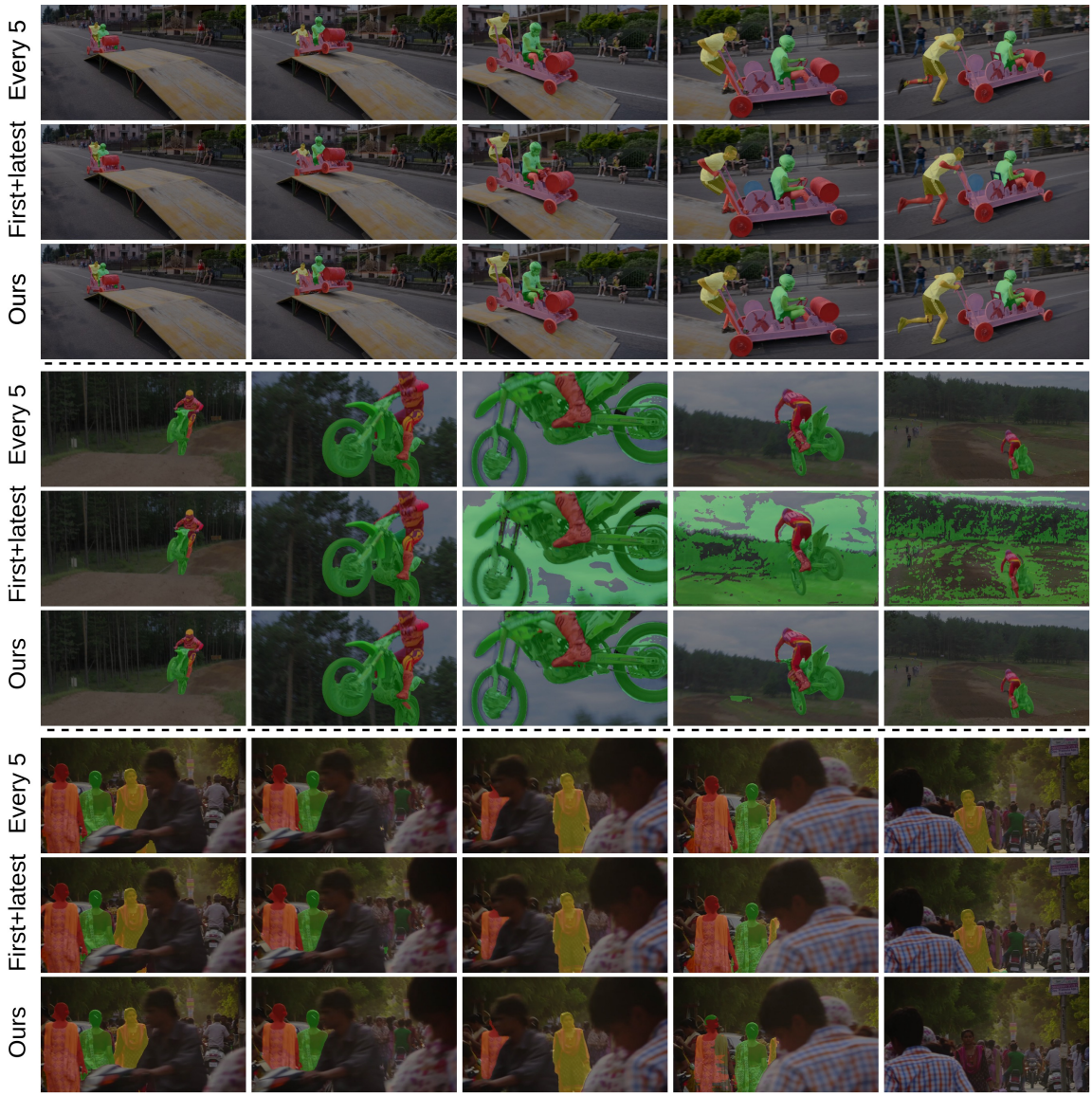

Figure 9: Qualitative results for DAVIS 2017. Frames are sampled from DAVIS 2017 [41] validation set. In each row, frames are temporally ordered from left to right. Frames are sampled from challenging situations and transitions. **Top:** Our method can successfully recover from the drifting. **Middle:** First-and-latest approach collapses as a result of fast object deformation. **Bottom:** Our method fails to re-identify object that has been completely occluded for a few frames. Features that belonged to this object got removed after being unused for a few frames. Since object form has not been changed through time, the first-and-latest method can successfully segment using first frame information.

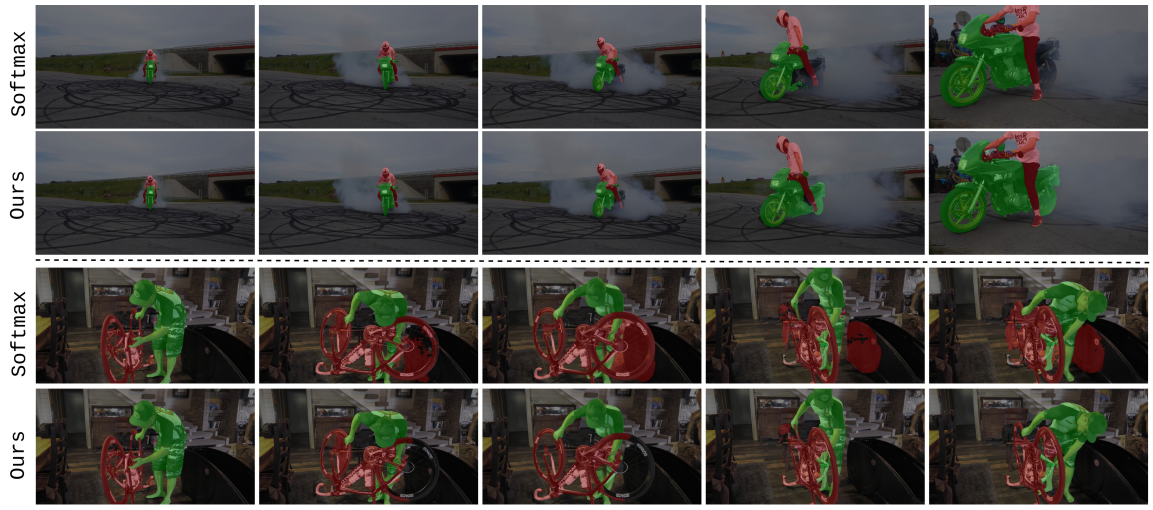

Figure 10: Qualitative results for top-k and softmax weights as an index. Frames are sampled from DAVIS 2017 [41] validation set. In each row, frames are temporally ordered from left to right. Frames are sampled from challenging situations and transitions. **Top:** top two rows shows top-k effectiveness to handle deformation of the object. **Bottom:** In the two bottom rows, we can see that using softmax weights, the network is unable to adapt to the object's new appearance as it changes toward the end of the sequence.

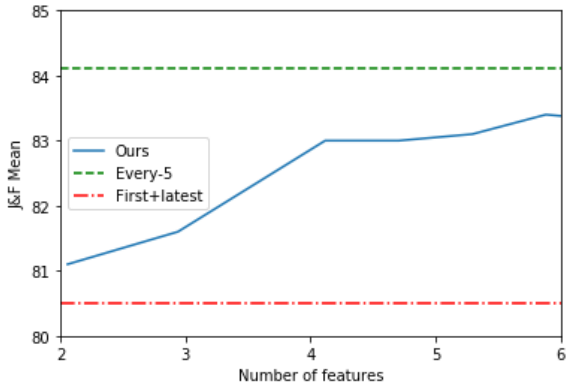

Figure 11: Effect of memory bank capacity on *J&F* metric. Every-5 and first-and-latest approach performance are shown for a clearer comparison. The size of the memory bank is specified in terms of the number of frames.

Chapter 4

Regularizers and Training

4.1 Memory bank regularization

During segmentation, we establish a pixel-wise connection between features of the support key and query key. As mentioned in [8, 7], the relation between support features and query features can be dominated by a small portion of discriminant features, which essentially limits other features information to be propagated. STCN [8] addresses this issue by replacing the dot product with negative Euclidean distance as the affinity metric since the dot product is biased toward activated features with a more significant norm. This section applies two regularization methods to the affinity matrix and studies their effect on the performance. The intuition is to reduce network reliance on discriminant features and improve network robustness to occlusions or deformations. To address this issue, we investigate dropout and support feature reallocation in the memory block.

Dropout was initially proposed to regularize the network during training and prevent it from over-fitting. Dropout randomly deactivates features during the training and makes the network perform the task using the remaining information. This operation increases the generalization ability and leads to more robust performance. After calculating the affinity matrix, we introduce a Dropout layer to randomly disable feature connection using binomial distribution $Pr(q)$ with $q = 0.5$. Dropout shifts attention to remaining features and forces the network to perform segmentation based on remaining information. This simulates occlusion or illumination change where query and support connection is affected by abrupt changes.

Methods	J-Mean	J-Recall	J-decay	F-Mean	F-Recall	F-decay	J&F-Mean
DAVIS 2016 validation set							
STCN + dropout	90.0	98.1	4.4	92.8	97.2	4.9	91.4
STCN + reallocation	89.7	97.3	4.4	91.2	96.4	5.5	90.5
STCN	90.4	98.1	4.1	93.0	97.1	4.3	91.7
DAVIS 2017 validation set							
STCN + dropout	79.1	88.1	9.4	85.5	92.3	10.5	81.6
STCN + reallocation	78.5	86.7	9.5	84.4	91.4	11.0	81.4
STCN	82.0	91.3	6.2	88.6	94.6	8.6	85.3

Table 5: Qualitative results on DAVIS 2016 and DAVIS 2017 validation sets. For evaluation, we followed the original STCN [8] strategy to ensure a fair comparison.

Methods	J-seen	J-unseen	F-seen	F-unseen	J&F
STCN + dropout	81.0,	76.9	84.9	84.0	81.6
STCN + reallocation	80.9	75.8	85.2	83.6	81.4
every 5	82.6	79.3	86.9	87.6	84.1

Table 6: Quantitative results on Youtube-VOS validation set.

In addition to dropout, inspired by [51], we also investigate the effect of support feature reallocation on network performance. The main goal is to reallocate the network's attention to a broader range of support features. The affinity matrix D shows pair-wise relation between query key K^q and support key K^s as is demonstrated in Eq. 1 in the manuscript. By taking the average along support key dimension, we create an attention map for support features.

$$d_{ij} = dist(k_i^s, k_j^q)$$
$$A_i = \sum_{j=1}^{HW} d_{ij}$$

The attention value of each support feature reflects its importance during the segmentation. We reallocate the attention between support features to facilitate information propagation and include more support features in the segmentation process. To do so, we sort support feature attention in descending order and get sorting index w for each feature i in support. Having sorted the index, we reweight the connection belonging to feature i in the affinity matrix using the following equation.

$$\hat{d_{ij}} = d_{ij} \times \frac{w_j}{\sum_{k=1}^{H} W w_k}$$

Multiplying normalized sorted index to corresponding pixels of affinity matrix increases the importance of the less discriminant features. Since most of the pixels

belong to the background, we only perform this operation on the pixels of each object separately. For each object in the support set, we store the corresponding object mask and multiply interpolated mask to support key to filter out background features.

We followed the training procedure suggested by STCN [8] to study the effect of these two regularizations on the network. We evaluated each method after complete training on Youtube-VOS [56], DAVIS 2017 [41] and DAVIS 2016 [40]. Quantitative Results for DAVIS 2017 and 2016 are shown in table 5 and Quantitative results for Youtube-VOS are shown in table 6.

On DAVIS 2016 [40] single object benchmark, both regularization methods achieve close results to the baseline. However, neither of them can improve the performance of the method. On DAVIS 2017[41] and Youtube-VOS [56] multi-object datasets, the network performance degrades using these regularization methods.

We randomly pick two objects from the sequence for multi-object training to save computation and reduce complexity in each iteration. However, both dropout and feature reallocation are harmful to multi-object training and add unproductive complexity to the network.

4.2 Training details

We follow [8, 7] strategies for data augmentation and training. They are mentioned here for completeness. However, we invite readers to look at the open-source code for more details.

4.2.1 Pre-training

At the pre-training stage, we used BIG[6], DUTS[52], HRSOD[61], FSS1000[22], and ECSSD[45]. BIG and HRSOD repeated five times in training data as they provide higher quality annotations. At each iteration, we randomly pick one image and generate three augmented samples from the base image. Before creating augmented samples, we apply random scaling, random horizontal flip, random color jitter, and random greyscale on the base image using Pytorch augmentation tools. To create augmented samples, we apply affine transform and another color jitter on the augmented base image. Each sample shorter side is then resized to 384, and a random crop applied after that resulting in 384×384 outputs. In this stage, we only train

the network with single object samples.

4.2.2 Main-training

At the main-training stage, we use Youtube-VOS [56], and DAVIS [41, 40]. DAVIS is repeated five times as it contains better annotations. We used the 480p version of DAVIS for training and resized Youtube-VOS such that the shorter edge is equal to 480. At each iteration, we sample three temporally ordered frames from a video to form the training sequence. The maximum temporal distance of sampled frames starts from 5 and increases by 5 at $[10\%, 20\%, 30\%, 40\%$. For the last 10% of iterations, the maximum temporal distance is changed back to 5 frames.

For augmentation, we apply random horizontal flip, random resized crop ($crop\_size = 384$), random color jitter and random grayscale on all images in the training sequence. The random seed is fixed for all images in the training sequence, and we apply the same transform on all images. Then, for each image, we perform color jitter and random affine transform. At each training sequence, we pick at most two objects to perform multi-object training.

4.2.3 Loss, Optimizer, and scheduling

Adam [19] optimizer is initialized with base learning rate of 10^-5, momentum of $\beta_1 = 0.9, \beta_2 = 0.999$, L2 weight decay of 10^-7, and step learning rate decay with decay ratio of 0.1. Decay is performed once in the middle of the pre-training stage and once in the middle of the main training. The learning rate schedule ends after each stage of training, and we initialize a new schedule for each stage. Batch normalization layers in the key and value encoder are frozen with the Imagenet pre-trained network parameters to speed up the training. For bootstrapped cross-entropy loss, we used 100% of pixels for the first 20k iterations. After that, we only use top-p% pixels with the highest error (hard pixel-mining) to compute gradients. p linearly decreases from 100 to 15 over 50k iteration and remains fixed after that.

Chapter 5

Conclusion and future work

Recent advances in deep learning made it possible to propose solutions for many complex problems in various areas. In the computer vision domain, deep learning methods can deliver a reliable results for semantic segmentation and object detection tasks.

Although deep learning approaches achieved considerable success in processing images in recent years, the amount of information in an image is limited compared to a video. The ability to process videos and extract meaningful information from videos is the next milestone to understanding the world through computer vision. To this end, video object segmentation is fundamental to many other tasks in video processing.

Currently, matching-based networks reached state-of-the-art results for semi-supervised video object segmentation. However, direct application of these methods to a real-world problem is not possible. This thesis addresses the shortcomings of current state-of-the-art methods and proposed adaptations that enable using these methods on arbitrary length videos. In particular, we proposed a novel memory management scheme that improves inference speed and enables the processing of long videos while respecting the computational limitation of the processing device. We showed that we could achieve comparable results with static methods while using a fraction of memory and computation power with adaptive memory management.

We further investigated possible regularization to memory bank to improve model robustness in edge cases such as abrupt illumination change, occlusion, and blur.

Calculating the similarity is a bottleneck in matching-based networks. With adaptive memory management, we use fixed-size memory, and as a result, the similarity calculation is no longer a limitation. As future work, similarity calculation can be done in a multi-scale feature embedding. This would improve model robustness for edge cases, especially when multiple objects with similar appearances are segmented.

Appendix

5.1 Detailed results

In this section, we provided detailed results of our experiments in this thesis. Results on Youtube-VOS are omitted.

Tables 8 and 7 show detailed results of our novel adaptive memory management on DAVIS 2016 and DAVIS 2017 validation sets.

Tables 9 and 10 shows per sequence results of applying memory bank regularization methods proposed in chapter 4 on DAVIS 2016 and DAVIS 2017 validation sets.

Sequence	J-Mean	F-Mean
bike-packing_1	0.783	0.863
bike-packing_2	0.894	0.918
blackswan_1	0.965	0.993
bmx-trees_1	0.534	0.941
bmx-trees_2	0.761	0.935
breakdance_1	0.926	0.945
camel_1	0.971	0.995
car-roundabout_1	0.987	0.988
car-shadow_1	0.974	0.997
cows_1	0.962	0.984
dance-twirl_1	0.916	0.939
dog_1	0.959	0.986
dogs-jump_1	0.887	0.959
dogs-jump_2	0.932	0.961
dogs-jump_3	0.945	0.977

drift-chicane_1	0.883	0.941
drift-straight_1	0.944	0.928
goat_1	0.918	0.957
gold-fish_1	0.854	0.878
gold-fish_2	0.783	0.870
gold-fish_3	0.911	0.948
gold-fish_4	0.923	0.963
gold-fish_5	0.925	0.938
horsejump-high_1	0.898	0.972
horsejump-high_2	0.845	0.988
india_1	0.925	0.912
india_2	0.745	0.750
india_3	0.660	0.689
judo_1	0.885	0.906
judo_2	0.846	0.867
kite-surf_1	0.532	0.919
kite-surf_2	0.396	0.490
kite-surf_3	0.788	0.946
lab-coat_1	0.090	0.272
lab-coat_2	0.000	0.000
lab-coat_3	0.969	0.962
lab-coat_4	0.944	0.933
lab-coat_5	0.926	0.908
libby_1	0.915	0.990
loading_1	0.975	0.987
loading_2	0.839	0.877
loading_3	0.913	0.967
mbike-trick_1	0.847	0.934
mbike-trick_2	0.828	0.849
motocross-jump_1	0.814	0.813
motocross-jump_2	0.858	0.814
paragliding-launch_1	0.851	0.920
paragliding-launch_2	0.759	0.929

paragliding-launch_3	0.133	0.419
parkour_1	0.956	0.984
pigs_1	0.927	0.961
pigs_2	0.805	0.895
pigs_3	0.925	0.910
scooter-black_1	0.786	0.925
scooter-black_2	0.898	0.895
shooting_1	0.315	0.411
shooting_2	0.908	0.884
shooting_3	0.830	0.947
soapbox_1	0.883	0.911
soapbox_2	0.819	0.880
soapbox_3	0.817	0.919

Table 7: Per sequence quantitative results of adaptive memory bank with memory bank size of 2 frames worth of features on DAVIS 2017 validation set.

Sequence	J-Mean	F-Mean
blackswan_1	0.962	0.990
bmx-trees_1	0.705	0.921
breakdance_1	0.907	0.925
camel_1	0.968	0.994
car-roundabout_1	0.987	0.987
car-shadow_1	0.973	0.997
cows_1	0.959	0.980
dance-twirl_1	0.912	0.931
dog_1	0.958	0.986
drift-chicane_1	0.876	0.941
drift-straight_1	0.945	0.934
goat_1	0.918	0.958
horsejump-high_1	0.917	0.976
kite-surf_1	0.753	0.926
libby_1	0.915	0.990

motocross-jump_1	0.861	0.810
paragliding-launch_1	0.657	0.466
parkour_1	0.956	0.985
scooter-black_1	0.946	0.910
soapbox_1	0.820	0.861

Table 8: Per sequence quantitative results of adaptive memory bank with memory bank size of 2 frames worth of features on DAVIS 2016 validation set.

	Dropout		Reallocation	
Sequence	J-Mean	F-Mean	J-Mean	F-Mean
blackswan_1	0.961	0.996	0.967	0.997
bmx-trees_1	0.729	0.941	0.68	0.907
breakdance_1	0.929	0.953	0.916	0.914
camel_1	0.966	0.995	0.971	0.988
car-roundabout_1	0.984	0.977	0.97	0.963
car-shadow_1	0.942	0.955	0.942	0.959
cows_1	0.961	0.989	0.964	0.986
dance-twirl_1	0.906	0.934	0.903	0.915
dog_1	0.942	0.985	0.955	0.981
drift-chicane_1	0.871	0.97	0.897	0.961
drift-straight_1	0.944	0.936	0.947	0.947
goat_1	0.904	0.954	0.919	0.958
horsejump-high_1	0.912	0.968	0.914	0.97
kite-surf_1	0.749	0.891	0.744	0.837
libby_1	0.896	0.98	0.908	0.979
motocross-jump_1	0.91	0.819	0.899	0.801
paragliding-launch_1	0.662	0.482	0.65	0.425
parkour_1	0.953	0.986	0.957	0.984
scooter-black_1	0.941	0.919	0.914	0.852
soapbox_1	0.951	0.945	0.936	0.928

Table 9: Per sequence quantitative results of different memory bank regularization on DAVIS 2016.

	Dropout		Reallocation	
Sequence	J-Mean	F-Mean	J-Mean	F-Mean
bike-packing_1	0.769	0.884	0.762	0.823
bike-packing_2	0.862	0.886	0.849	0.862
blackswan_1	0.961	0.996	0.967	0.997
bmx-trees_1	0.515	0.889	0.446	0.887
bmx-trees_2	0.753	0.924	0.739	0.92
breakdance_1	0.929	0.953	0.916	0.914
camel_1	0.966	0.995	0.971	0.988
car-roundabout_1	0.984	0.977	0.97	0.963
car-shadow_1	0.942	0.955	0.942	0.959
cows_1	0.961	0.989	0.964	0.986
dance-twirl_1	0.906	0.934	0.903	0.915
dog_1	0.942	0.985	0.955	0.981
dogs-jump_1	0.24	0.333	0.12	0.167
dogs-jump_2	0.604	0.742	0.582	0.737
dogs-jump_3	0.936	0.976	0.936	0.968
drift-chicane_1	0.871	0.97	0.897	0.961
drift-straight_1	0.944	0.936	0.947	0.947
goat_1	0.904	0.954	0.919	0.958
gold-fish_1	0.846	0.879	0.856	0.886
gold-fish_2	0.751	0.799	0.774	0.826
gold-fish_3	0.846	0.901	0.869	0.907
gold-fish_4	0.899	0.959	0.919	0.959
gold-fish_5	0.914	0.938	0.927	0.928
horsejump-high_1	0.893	0.968	0.895	0.968
horsejump-high_2	0.836	0.983	0.839	0.981
india_1	0.923	0.905	0.926	0.915
india_2	0.739	0.746	0.74	0.745

india_3	0.837	0.83	0.837	0.834
judo_1	0.884	0.914	0.861	0.886
judo_2	0.843	0.866	0.829	0.849
kite-surf_1	0.5	0.879	0.47	0.845
kite-surf_2	0.475	0.629	0.414	0.567
kite-surf_3	0.8	0.957	0.81	0.974
lab-coat_1	0	0	0	0
lab-coat_2	0	0	0	0.011
lab-coat_3	0.965	0.955	0.966	0.943
lab-coat_4	0.942	0.918	0.937	0.913
lab-coat_5	0.925	0.903	0.921	0.901
libby_1	0.896	0.98	0.908	0.979
loading_1	0.968	0.983	0.967	0.972
loading_2	0.638	0.765	0.578	0.695
loading_3	0.928	0.97	0.911	0.943
mbike-trick_1	0.809	0.871	0.826	0.902
mbike-trick_2	0.826	0.855	0.831	0.88
motocross-jump_1	0.815	0.838	0.831	0.82
motocross-jump_2	0.854	0.774	0.856	0.776
paragliding-launch_1	0.788	0.909	0.755	0.893
paragliding-launch_2	0.714	0.92	0.722	0.896
paragliding-launch_3	0.169	0.472	0.128	0.412
parkour_1	0.953	0.986	0.957	0.984
pigs_1	0.912	0.952	0.911	0.906
pigs_2	0.784	0.885	0.721	0.831
pigs_3	0.946	0.942	0.96	0.947
scooter-black_1	0.841	0.973	0.823	0.955
scooter-black_2	0.911	0.922	0.893	0.876
shooting_1	0.371	0.444	0.485	0.508
shooting_2	0.899	0.857	0.899	0.87
shooting_3	0.909	0.974	0.881	0.958
soapbox_1	0.886	0.913	0.876	0.902
soapbox_2	0.799	0.855	0.776	0.891

soapbox_3	0.84	0.913	0.838	0.924

Table 10: Per sequence quantitative results of different memory bank regularization on DAVIS 2017.

5.2 Reproduction

In order to reproduce results, the code and weights that are used in our experiments are made available online at `https://github.com/alipga/STCN`.

Bibliography

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

[3] Pascal Bertolino. Sensarea, a general public video editing application. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 3429–3431. IEEE, 2014.

[4] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 221–230, 2017.

[5] Bodhiswatta Chatterjee and Charalambos Poullis. On building classification from remote sensor imagery using deep neural networks and the relation between classification and reconstruction accuracy using border localization as proxy. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 41–48, 2019.

[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, Yu-Wing Tai, and Chi-Keung Tang. Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *CVPR*, 2021.

[8] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. *arXiv preprint arXiv:2106.05210*, 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. Ieee, 2009.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Qianyu Feng, Zongxin Yang, Peike Li, Yunchao Wei, and Yi Yang. Dual embedding learning for video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[13] Alex Graves. Long short-term memory. In *Supervised sequence labelling with recurrent neural networks*, pages 37–45. Springer, 2012.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[16] Li Hu, Peng Zhang, Bang Zhang, Pan Pan, Yinghui Xu, and Rong Jin. Learning position and target consistency for memory-based video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4144–4154, 2021.

[17] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G Schwing. Maskrnn: Instance level video object segmentation. *arXiv preprint arXiv:1803.11187*, 2018.

[18] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for multiple object tracking. *arXiv preprint arXiv:1703.09554*, 2017.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR, 2016.

[21] Sangho Lee, Jinyoung Sung, Youngjae Yu, and Gunhee Kim. A memory network approach for story-based temporal summarization of 360 videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1419, 2018.

[22] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. Fss-1000: A 1000-class dataset for few-shot segmentation. In *CVPR*, 2020.

[23] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 90–105, 2018.

[24] Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module. In *European Conference on Computer Vision*, pages 735–750. Springer, 2020.

[25] Chen Liang, Zongxin Yang, Jiaxu Miao, Yunchao Wei, and Yi Yang. Memory aggregated cfbi+ for interactive video object segmentation. In *CVPR Workshops*, volume 1, page 2, 2020.

[26] Yongqing Liang, Xin Li, Navid Jafari, and Qin Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. *arXiv preprint arXiv:2010.07958*, 2020.

[27] Huaijia Lin, Xiaojuan Qi, and Jiaya Jia. Agss-vos: Attention guided single-shot video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3949–3957, 2019.

[28] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*, pages 565–580. Springer, 2018.

[29] K-K Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. Video object segmentation without temporal information. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1515–1530, 2018.

[30] Jiaxu Miao, Yunchao Wei, and Yi Yang. Memory aggregation networks for efficient interactive video object segmentation. In *Proceedings of the IEEE/CVF CVPR*, pages 10366–10375, 2020.

[31] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

[32] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 677–685, 2017.

[33] King Ngi Ngan and Hongliang Li. *Video segmentation and its applications*. Springer Science & Business Media, 2011.

[34] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7376–7385, 2018.

[35] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Fast user-guided video object segmentation by interaction-and-propagation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5247–5256, 2019.

[36] 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*, pages 9226–9235, 2019.

[37] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[39] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2663–2672, 2017.

[40] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

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

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[43] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In *European Conference on Computer Vision*, pages 629–645. Springer, 2020.

[44] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017.

[45] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. In *TPAMI*, 2015.

[46] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[48] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9481–9490, 2019.

[49] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. *arXiv preprint arXiv:1706.09364*, 2017.

[50] Haochen Wang, Xiaolong Jiang, Haibing Ren, Yao Hu, and Song Bai. Swiftnet: Real-time video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1296–1305, 2021.

[51] Haochen Wang, Xudong Zhang, Yutao Hu, Yandan Yang, Xianbin Cao, and Xiantong Zhen. Few-shot semantic segmentation with democratic attention networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 730–746. Springer, 2020.

[52] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 136–145, 2017.

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[54] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. Ranet: Ranking attention network for fast video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3978–3987, 2019.

[55] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Shengping Zhang, and Wenxiu Sun. Efficient regional memory network for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1286–1295, 2021.

[56] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. In *ECCV*, 2018.

[57] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019.

[58] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6499–6507, 2018.

[59] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 152–167, 2018.

[60] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *European Conference on Computer Vision*, pages 332–348. Springer, 2020.

[61] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. Towards high-resolution salient object detection. In *ICCV*, 2019.

[62] Peng Zhang, Li Hu, Bang Zhang, Pan Pan, and DAMO Alibaba. Spatial consistent memory network for semi-supervised video object segmentation. In *CVPR Workshops*, volume 6, 2020.

[63] Zhishan Zhou, Lejian Ren, Pengfei Xiong, Yifei Ji, Peisen Wang, Haoqiang Fan, and Si Liu. Enhanced memory network for video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.