Hierarchical Memory Matching Network for Video Object Segmentation

Hongje Seong[1] Seoung Wug Oh[2] Joon-Young Lee[2]
Seongwon Lee[1] Suhyeon Lee[1] Euntai Kim[1,*]
[1]Yonsei University [2]Adobe Research

Abstract

We present Hierarchical Memory Matching Network (HMMN) for semi-supervised video object segmentation. Based on a recent memory-based method [33], we propose two advanced memory read modules that enable us to perform memory reading in multiple scales while exploiting temporal smoothness. We first propose a kernel guided memory matching module that replaces the non-local dense memory read, commonly adopted in previous memory-based methods. The module imposes the temporal smoothness constraint in the memory read, leading to accurate memory retrieval. More importantly, we introduce a hierarchical memory matching scheme and propose a top-k guided memory matching module in which memory read on a fine-scale is guided by that on a coarse-scale. With the module, we perform memory read in multiple scales efficiently and leverage both high-level semantic and low-level fine-grained memory features to predict detailed object masks. Our network achieves state-of-the-art performance on the validation sets of DAVIS 2016/2017 (90.8% and 84.7%) and YouTube-VOS 2018/2019 (82.6% and 82.5%), and test-dev set of DAVIS 2017 (78.6%). The source code and model are available online: https://github.com/Hongje/HMMN.

1. Introduction

Semi-supervised video object segmentation (VOS) aims to predict the foreground object mask in every frame of a video given an object mask at the first frame. Recently, memory-based VOS methods [33, 39, 27, 21, 22, 23] have achieved great success. A key idea of the memory-based methods is matching densely between *query* (*i.e.*, current frame) and *memory* (*i.e.*, past frames with given or predicted masks) to retrieve the memory at a pixel-level. Since the camera's field of view or objects in a video may move, spatio-temporal non-local and dense matching was performed to compute similarity for all matching possibilities.

There are two limitations of the existing memory-based methods: *temporal smoothness* and *fine-grained memory information*. Temporal smoothness is one of the strong constraints that we can assume for the VOS task. Previous VOS methods without memory often applied a local matching [43, 50] or local refinement [34, 16, 32, 49, 13, 53] between two adjacent frames for temporal smoothness. However, in the memory-based method [33], the non-local matching completely ignores the constraint and it raises the risk of false matches (*e.g.*, when multiple similar instances exist, see Fig. 3). Another weakness is the lack of fine-grained memory information. In the memory-based methods, a query encoder only takes the current frame without any target information. Thus the memory matching is the only source to get information of the target object mask. The previous memory-based methods conduct the memory matching only at the coarsest resolution, (*e.g.*, $1/16$ of the input resolution [33]), as shown in Fig. 1 (a). At the low resolution, while accurate matching is possible with high-level

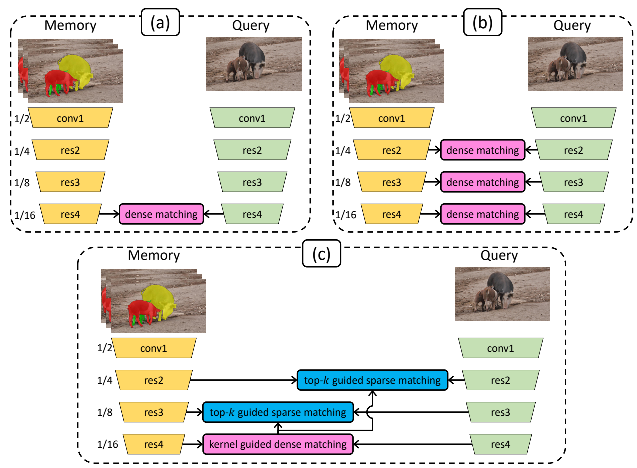

Figure 1. Previous memory-based methods densely match image features only at a coarse resolution, as shown in (a). To conduct the memory reading at multiple scales, one can naively apply dense matching at each scale (b), but it needs prohibitive computational cost and is not robust due to noisy low-level features. In our hierarchical memory matching architecture, shown in (c), fine-scale matching is guided by coarse-scale matching, resulting in efficient and robust memory matching in multiple scales.

*Corresponding author.

semantic features, we cannot expect fine-grained information that is also important to predict fine-detailed masks.

In this paper, we propose Hierarchical Memory Matching Network (HMMN) with two novel memory matching modules. To exploit the temporal smoothness, we propose *kernel guided memory matching* module. We restrict possible correspondences between two adjacent frames to a local window and apply kernel guidance to the non-local memory matching that imposes the temporal smoothness constraint. For long-range matching between distant frames, we track the most probable correspondence for each memory pixel to a query pixel and apply relaxed kernel guidance according to the temporal distance, resulting in a smooth transition from local to global memory matching. This module replaces the non-local memory reading in the previous memory-based networks.

To retrieve fine-grained memory information, we propose *top-k guided memory matching* module. The computational cost for the dense memory matching grows quadratically with increasing search space. Naively performing the memory reading at fine-scales [51] (Fig. 1 (b)) requires prohibitively heavy computation. Also, memory matching with the low-level features at a fine-scale is susceptible to noisy matches. Our top-k guided memory matching solves both the computational cost and the matching robustness issues. We first sample the top-k candidate memory locations for each query pixel using the matching similarity score at the coarse-scale. Then, we conduct fine-scale memory matching between each query pixel and the corresponding candidate memory locations, as shown in Fig. 1 (c). The top-k guided memory matching reduces the matching complexity at high-resolution significantly from $\mathcal{O}(TH^2W^2)$ to $\mathcal{O}(kHW)$, where T, H, and W are the time, height, and width of the feature map, and k is a constant. The coarse-to-fine hierarchical matching scheme makes our fine-scale memory matching robust even with low-level features. We note that some previous works [19, 55] also reduce memory matching complexity by extracting k matching candidates but they select candidates using features at the same scale. In contrast, we selected k matching candidates from high-level (*i.e.*, coarse-scale) semantic features, thus semantically more accurate matching candidates would be selected.

Our contributions are summarized as follows:

- We propose kernel guided memory matching module, imposing the temporal smoothness constraint to the non-local matching with all memory frames.
- We propose top-k guided memory matching module, resulting in efficient and robust fine-scale memory matching.
- With the two novel memory matching modules, we present Hierarchical Memory Matching Network (HMMN) that performs coarse-to-fine hierarchical memory matching effectively.
- Our network achieves state-of-the-art performance on both DAVIS and YouTube-VOS benchmarks.

2. Related Work

Semi-supervised Video Object Segmentation: Semi-supervised VOS [35, 36, 48] has been tackled in two ways: online-learning method and offline-learning method. The online-learning methods [5, 3, 44, 1, 28, 29, 47, 7, 30] fine-tune networks at test time using the given ground-truth mask at the first frame. The objective of fine-tuning is to let networks detect target objects for each video. Therefore, the online-learning method can expect accurate results by training a target-specific network, but they are subject to severe disadvantages at run-time because the network needs to be trained multiple times on the first frame during testing.

Offline-learning methods aim to train a network that works well for any input videos without test-time training. It has usually been solved by mask propagation or pixel-wise matching. The propagation-based methods [34, 16, 12, 20, 32, 15, 4, 54, 11] train a network to propagate the given mask sequentially from the first frame. Since the propagation is conducted in a short-time interval, the methods often exploit the temporal smoothness constraint but are not robust to occlusion. The matching-based methods [41, 13, 52, 43, 14, 50] predict a foreground mask in the current frame based on matching with previously predicted or given mask. Recently, STM [33] introduced a memory-based method for offline-learning VOS and demonstrated a significantly improved performance while achieving a fast run-time. Our approach follows the memory-based method, and we address the main limitations of existing methods.

Memory-based Video Object Segmentation: Memory networks [42, 31, 18] memorize external information as **key** and **value**, then the **value** is retrieved by query via non-local matching with the **key**. It was first proposed for natural language processing, and STM [33] repurposed the memory networks to memory-based VOS. STM retrieves memory using non-local and dense memory matching and finds the target object in the query using the retrieved memory. Extended from STM, EGMN [27] proposed the graph memory networks to update memory using query. GC [21] introduced a new global matching method for fast memory matching. Liang *et al*. [23] proposed an adaptive memory update scheme to reduce redundant computation at memory matching. Li *et al*. [22] explored a cyclic mechanism for both training and inference to boost performance. KMN [39] additionally conducted *memory-to-query* matching then applied 2D Gaussian kernels on the query for robust matching. The previous memory-based methods overlook the temporal smoothness, one of the most important cues for VOS, as they performed memory matching in a non-local manner. In addition, the previous works conduct memory matching only at the coarsest resolution, which

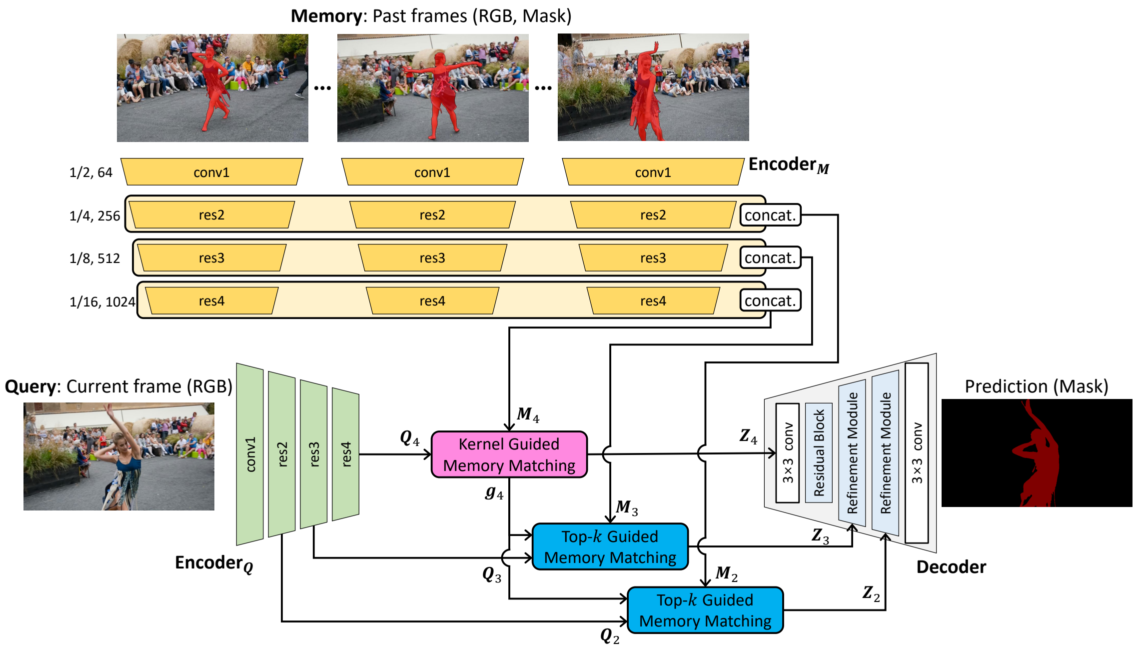

Figure 2. An overview of HMMN. Our network consists of two ResNet-based encoders for the query and the memory frames that extract multi-scale features, kernel guided memory matching block that operates on the coarsest scale, top-k guided memory matching blocks that operate on the finer scales, and a decoder that takes the memory reading results and produces the final mask prediction.

hard to expect to take fine mask information. We address the problems by introducing two matching modules, kernel guided memory matching and top-k memory matching. Note that our kernel guided memory matching is completely different from kernelization used in KMN [39], which generates kernel based on non-local matching thus does not exploit temporal smoothness.

3. Method

Our method, Hierarchical Memory Matching Network (HMMN), is based on STM [33]. Given a ground-truth object mask at the first frame, we sequentially predict the target object mask from the second frame to the last frame. The past frames concatenated with predicted or given masks are set to memory, and the current frame is used as a query.

The main distinction comes from the construction and the use of hierarchical memory. The objective of the hierarchical memory is to leverage memories in multiple scales, from low-resolution semantic features to high-resolution detailed features, on the memory-based VOS architectures. To efficiently read the information from the hierarchical memory, we design two types of memory matching modules based on the feature map's scale: kernel guided dense memory matching at the coarsest scale, and top-k guided sparse memory matching at fine scales. At the coarsest scale, we perform dense and non-local query-memory matching similar to STM [33] and other variants. But, we improve the robustness of the global matching through the kernel guidance that exploits temporal smoothness as an additional cue. At the finer scales followed by the coarsest level, we perform a sparse query-memory matching making use of the matching results from the coarsest level as guidance. Specifically, we take the top-k memory matching for each query point at the coarsest scale and use them to guide the sparse matching at the finer scales. In this way, we can retrieve fine-detailed memory information while taking a fractional computational cost compared to dense memory matching.

The overview of our network is shown in Fig. 2. In our network, memory and query frames are first fed into two independent ResNet50 [10]-based encoders. Both encoders extract multi-scale features – $\mathbf{Q}_S$ for the query frame and $\mathbf{M}_S$ for the memory frames – from ResNet50's S-th `res` block. We use three scales where $S \in \{2, 3, 4\}$ with the output scale of {1/4, 1/8, 1/16} with respect to the input image. At each scale, in the order of coarse-to-fine scales, we perform a memory read by matching the query and memory features, and then the outputs further go through the decoder to predict an object mask.

For the memory matching in the coarsest scale, the embedded query and memory $\{\mathbf{Q}_4, \mathbf{M}_4\}$ are fed into *kernel guided memory matching* module, and it outputs the updated feature ($\mathbf{Z}_4$) and a guidance ($\mathbf{g}_4$) which is the similarity matrix used for memory retrieval. For the finer scales (S is 2 or 3), *top-k guided memory matching* module is used instead. It takes a pair of embedded query and memory $\{\mathbf{Q}_S$, $\mathbf{M}_S\}$ along with the guidance ($\mathbf{g}_4$), and outputs the updated feature $\mathbf{Z}_S$. Finally, the decoder takes all the output features $\mathbf{Z}_S$ (either as the input or through a skip-connection), and makes a mask prediction. Note that, except for the new

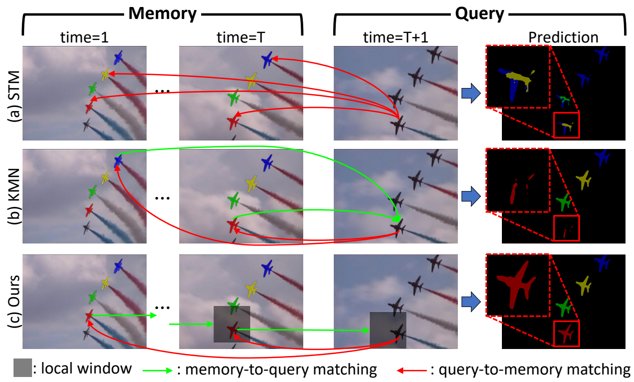

Figure 3. The effect of our kernel guided matching module. (a) STM does not make use of bi-directional memory-to-query matching. (b) KMN performs a memory-to-query matching in a non-local manner, thus it cannot make use of temporal smoothness prior. (c) Our memory-to-query matching is achieved by connecting local tracking, thus it can impose temporal smoothness on the query-to-memory matching results.

memory matching modules, we keep the rest of the network structure (*e.g.*, encoder and decoder design) as the same as STM [33].

3.1. Kernel Guided Memory Matching

With the embedded memory and query $(\mathbf{M}_4, \mathbf{Q}_4)$, extracted from each encoder at `res4` stage, we first encode **keys** $(\mathbf{k}_{M4}, \mathbf{k}_{Q4})$ and **values** $(\mathbf{v}_{M4}, \mathbf{v}_{Q4})$ via four independent 3×3 convolutional layers. Then, a non-local matching between memory and query is performed using **keys** as follows:

$$\mathcal{M}_4 = \mathbf{k}_{M4}\mathbf{k}_{Q4}^\top, \tag{1}$$

where $^\top$ indicates a matrix transpose. Based on the non-local matching $(\mathcal{M}_4)$, we compute the attention map $(\mathbf{g}_4)$ by

$$\mathbf{g}_4 = L_1\left(\mathcal{K}(\mathcal{M}_4) \odot \text{softmax}(\mathcal{M}_4)\right), \tag{2}$$

where $\odot$ indicates an element-wise multiplication, $L_1(\cdot)$ is L1 normalization which normalizes along the memory dimension, and $\mathcal{K}(\cdot)$ is 2D Gaussian kernel. Then, the memory **value** is retrieved using the attention map $(\mathbf{g}_4)$ as follows:

$$\mathbf{v}'_{M4} = \mathbf{v}_{M4}^\top \mathbf{g}_4. \tag{3}$$

Finally, the query **value** $(\mathbf{v}_{Q4})$ is concatenated with the retrieved **value** $(\mathbf{v}'_{M4})$ along the feature dimension to be the output.

Here, we impose the temporal smoothness, that is the common and strong constraint for videos, on the memory matching through the kernel prior $(\mathcal{K})$. If $\mathcal{K}(\cdot) = \mathbf{1}$, the output $(\mathbf{Z}_4)$ will be the same as the output from vanilla memory read block used in STM [33]. In other words, STM [33] retrieves memory solely based on non-local query-to-memory matching (*i.e.*, softmax$(\mathcal{M}_4)$), as illustrated in Fig. 3 (a). Thus, the fact that objects are likely to appear in

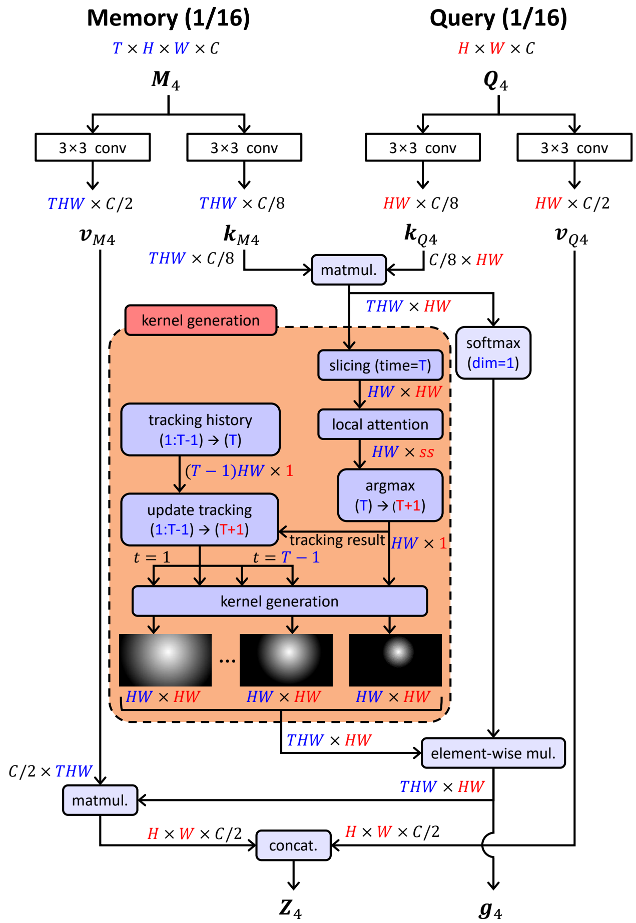

Figure 4. A detailed implementation of kernel guided memory matching module. We use blue and red to indicate memory and query dimensions, respectively. Note that we can access the tracking history (1:T-1)→(T) that is saved beforehand, thus only the tracking between the previous frame and the current frame (T)→(T+1) is needed to be newly computed.

similar local positions between adjacent frames (*i.e.*, temporal smoothness) is completely ignored. To harness this behavior, we additionally generate a kernel guidance $(\mathcal{K}(\cdot))$ based on spatio-temporal local matching. As illustrated in Fig. 3 (c), we conduct memory-to-query matching between two adjacent frames for every memory pixel. Here, we constrained the matching to perform only within a local region with a window size of s. Between every two adjacent frames, we track every pixel by selecting a single pixel within a local window that has the highest similarity score. This way, every memory pixel can reach to the best-matching query pixel by connecting local pixel-level tracking frame-by-frame. Based on the resulting memory-to-query matching, we generate 2D Gaussian kernels for every memory pixel with the standard deviation of σ^t. As the temporal distance of memory-to-query increases, the tracking error can be accumulated and the temporal smoothness constraint weakens. Thus, we relaxed the kernel guidance

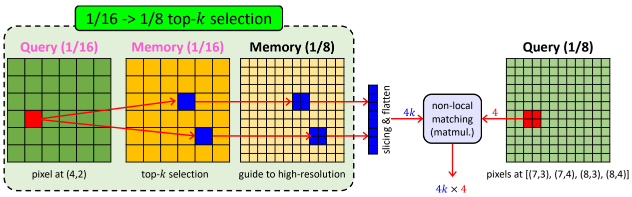

Figure 5. An example of top-k selection at `res3` stage. To simplify the illustration, k is set to 2 in this example.

by controlling the standard deviation according to the temporal distance by $\sigma^t = \sigma_{init} + (T - t)\sigma_{factor}$. This results in a smooth transition from local to global memory matching according to the temporal distance between query and memory features. A detailed implementation of kernel guided memory matching module is shown in Fig. 4.

Note that our kernel guidance is inspired by KMN [39], but the objective is completely different. KMN [39] used kernel only for robust matching from bi-directional attention, thus the kernels were generated based on non-local matching, as illustrated in Fig. 3 (b). Our kernel guidance, however, is based on fully local matching, and it effectively exploits the temporal smoothness as shown in Fig. 3 (c).

3.2. Top-k Guided Memory Matching

The main objective of computing a dense spatio-temporal attention map in memory matching module is to find when-and-where each query pixel attends to memory pixel. However, computing the dense attention map in high resolution requires prohibitively large computing resources as its computational complexity grows quadratically with regard to the feature map size. Thus, computing dense attention maps for finer levels of the feature hierarchy (`res3` and `res2`) is computationally too expensive. We address this issue by reducing the number of matching candidates in memory using top-k guidance.

Here, we assume that the matching result at high-resolution should be similar to that at low-resolution. By this assumption, we reuse the dense matching result at low-resolution as guidance for matching in higher resolution. An illustration of selecting k pixels and guiding to high-resolution for each query pixel is depicted in Fig. 5. Based on the low-resolution attention map ($\mathbf{g}_4$), which comes from `res4` stage, we select k best matching memory pixels for each query pixel via top-k operation. Then, only a sparse matching to selected pixels from the memory is performed.

Note that the selected k pixels in `res4` correspond to $4k$ and $16k$ pixels at `res3` and `res2` stages, respectively, thus we take k and $k/4$ for guiding each at `res3` and `res2` stage in order to have a similar computational overhead. This memory read module based on sparse matching can be efficiently implemented with a combination of common tensor operations. A detailed implementation of the top-k guided memory matching module is shown in Fig. 6. The outputs of top-k guided memory matching modules ($\mathbf{Z}_3$, $\mathbf{Z}_2$) are fed into the decoder through shortcut connections at the corresponding scale.

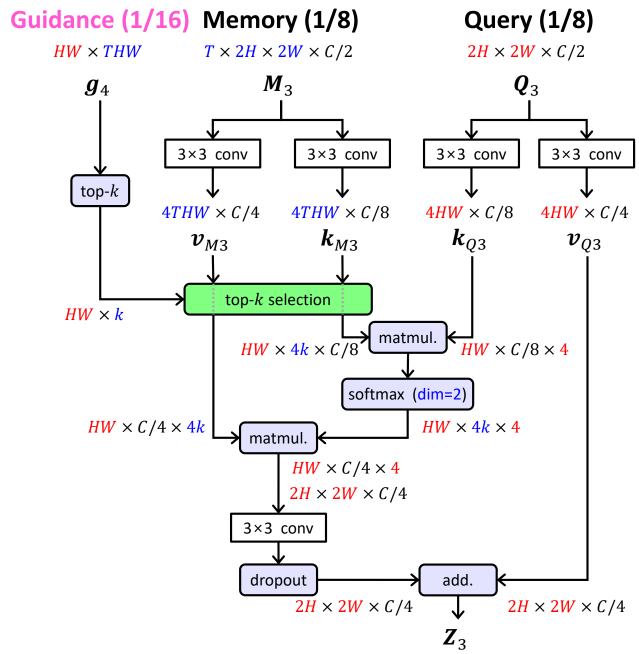

Figure 6. A detailed implementation of the top-k guided memory matching module at the `res3` stage. Memory and query dimensions are indicated using blue and red. The detailed implementation at the `res2` stage is provided in the supplementary material.

Note that, in the module, rather than directly using the retrieved **values** as the output, we place one convolutional layer followed by Dropout layer before added to the query **value** as residual. This design choice is due to the following observation. Without dropout, the model tends to converge to a sub-optimal state that does not make use of the matching results at the coarsest scale (*i.e.*, memory at `res4`). This sub-optimal model appears to take a shortcut for easier solutions, simply relying on low-level mask information (*i.e.*, memory at `res2` and `res3`) ignoring the high-level semantic matching. We were able to prevent this behavior by delivering the information in a restrictive way through a residual connection after a dropout layer that randomly drops the *whole* input feature during training. In this way, the network has to consider the output of top-k guided memory matching module as supplementary information to refine the memory matching at the coarsest resolution.

4. Experiments

4.1. Implementation Details

Training. For a fair comparison with STM [33], we follow the same training strategies. We initialize the encoders with ImageNet [38] pre-trained weights and randomly initialize the other layers. Then, we take the images with object masks in [8, 25, 9, 40, 6, 45] and pretrain HMMN on the

Method	OL	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	Time
e-OSVOS [30]	✓	86.8	86.6	87.0	3.4s
DyeNet [20]	✓	-	86.2	-	2.32s
RaNet [46]	✓	87.1	86.6	87.6	4s
STM (**+YV**) [33]		89.3	88.7	89.9	0.16s
CFBI (**+YV**) [50]		89.4	88.3	90.5	0.18s
KMN (**+YV**) [39]		90.5	89.5	91.5	0.12s
HMMN		89.4	88.2	90.6	0.10s
HMMN (**+YV**)		**90.8**	**89.6**	**92.0**	0.10s

Table 1. Comparison on DAVIS 2016 validation set. (**+YV**) indicates YouTube-VOS is additionally used for training, and OL denotes the use of online-learning strategies during test-time. Time measurements reported in this table are directly from the corresponding papers.

image datasets. Specifically, we generate three frames by augmenting each image via random affine transforms. The random affine transforms include rotation, shearing, zooming, translation, and cropping. During the pre-training, the dropout rate in top-k guided memory matching module (§3.2) is gradually decreased from 1 to 0.5.

After the pre-training on image datasets, the main training is done using either DAVIS 2017 [36] or YouTube-VOS 2019 [48] training set depending on the target benchmark. During main training, three frames are randomly sampled from a video with the gradually increasing maximum interval (from 0 to 25). The dropout rate in top-k guided memory matching module is gradually decreased from 0.5 to 0.

During both pre-training and main training, we minimize pixel-wise cross-entropy loss with Adam optimizer [17], and the learning rate is set to 1e-5. We use an input size of 384×384 and a mini-batch size of 4. According to [33], we employ the soft aggregation operation when multiple target objects exist in a video.

Inference. As in [33, 39], we take the first frame, the previous frame, and the intermediate frames sampled at every 5 frames for the memory in the coarsest scale ($\mathbf{M}_4$). For the fine-scale memories ($\mathbf{M}_3, \mathbf{M}_2$), we do not use the intermediate frames to avoid GPU memory overflow unless mentioned otherwise. We use the same number of k for top-k guided memory matching during training and inference, which is set to 32. The kernel guidance in §3.1 is used only during inference, as in KMN [39]. We have tried to use the kernel guidance during training, but there was no noticeable improvement. We set the standard deviation of σ_{init} and σ_{factor} into 3 and 0.5, respectively, and we used window size s of 7. We measure our run-time using a single NVIDIA GeForce 1080 Ti GPU.

4.2. Comparisons

We compare our HMMN against state-of-the-art methods on DAVIS [35, 36] and YouTube-VOS [48] benchmarks. For DAVIS benchmarks, 60 videos from DAVIS 2017 training set are used during main training following the common evaluation protocol [32, 49, 33, 39]. In addition, we report our results on DAVIS benchmarks using additional training videos from Youtube-VOS for a fair comparison with some recent methods [33, 39, 2, 50, 27, 37]. For Youtube-VOS benchmarks, the training set of 3471 videos are used. For all experiments, we either use the official evaluation code or upload our results to the evaluation server.

Method	OL	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
FRTM (**+YV**) [37]	✓	76.7	-	-
e-OSVOS [30]	✓	77.2	74.4	80.0
PReMVOS [28]	✓	77.8	73.9	81.7
LWL (**+YV**) [2]		81.6	79.1	84.1
STM (**+YV**) [33]		81.8	79.2	84.3
CFBI (**+YV**) [50]		81.9	79.1	84.6
EGMN (**+YV**) [27]		82.8	80.2	85.2
KMN (**+YV**) [39]		82.8	80.0	85.6
HMMN		80.4	77.7	83.1
HMMN (**+YV**)		**84.7**	**81.9**	**87.5**

Table 2. Comparison on DAVIS 2017 validation set.

Method	OL	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
CINN [1]	✓	67.5	64.5	70.5
DyeNet [20]	✓	68.2	65.8	70.5
PReMVOS [28]	✓	71.6	67.5	75.7
STM (**+YV**) [33]		72.2	69.3	75.2
CFBI (**+YV**) [50]		74.8	71.1	78.5
KMN (**+YV**) [39]		77.2	74.1	80.3
HMMN (**+YV**)		**78.6**	**74.7**	**82.5**

Table 3. Comparison on DAVIS 2017 test-dev set.

DAVIS [35, 36] is a densely annotated VOS dataset and mostly adopted benchmark to evaluate VOS models. To evaluate HMMN on DAVIS benchmarks, we use an input size of 480p resolution for all experiments. DAVIS dataset is divided into two sets: (1) DAVIS 2016, which is an object-level annotated dataset (single object); and (2) DAVIS 2017, which is an instance-level annotated dataset (multiple objects). The official metrics, region similarity $\mathcal{J}$ and contour accuracy $\mathcal{F}$, are measured for comparison. As shown in Table 1, our HMMN achieves state-of-the-art performance while taking a fast run-time on DAVIS 2016 validation set. Further, even without an additional YouTube-VOS dataset to train HMMN, we surpass most state-of-the-art methods.

We also conduct comparisons on DAVIS 2017 validation and test-dev sets, and the results are given in Table 2 and Table 3. As shown in the Tables, our HMMN significantly outperforms the current best results by **1.9**% and **1.4**% of $\mathcal{J}\&\mathcal{F}$ scores on DAVIS 2017 validation and test-dev sets, respectively. We omitted some comparable works in the tables. The full comparison tables are available in the supplementary material.

YouTube-VOS [48] is a large-scale benchmark for VOS. To evaluate our HMMN on YouTube-VOS benchmarks, we

Method	OL	$\mathcal{G}$	$\mathcal{J}_\mathcal{S}$	$\mathcal{J}_\mathcal{U}$	$\mathcal{F}_\mathcal{S}$	$\mathcal{F}_\mathcal{U}$
		YouTube-VOS 2018 validation set				
AGSS-VOS [24]		71.3	71.3	65.5	75.2	73.1
e-OSVOS [30]	✓	71.4	71.7	74.3	66.0	73.8
FRTM [37]	✓	72.1	72.3	65.9	76.2	74.1
STG-Net [26]		73.0	72.7	69.1	75.2	74.9
STM [33]		79.4	79.7	72.8	84.2	80.9
AFB+URR [23]		79.6	78.8	74.1	83.1	82.6
EGMN [27]		80.2	80.7	74.0	85.1	80.9
CFBI [50]		81.4	81.1	75.3	85.8	83.4
KMN [39]		81.4	81.4	75.3	85.6	83.3
LWL [2]		81.5	80.4	76.4	84.9	84.4
HMMN		**82.6**	**82.1**	**76.8**	**87.0**	**84.6**
		YouTube-VOS 2019 validation set				
STM* [33]		79.3	79.8	73.0	83.8	80.5
KMN* [39]		80.0	80.4	73.8	84.5	81.4
CFBI [50]		81.0	80.6	75.2	85.1	83.0
HMMN		**82.5**	**81.7**	**77.3**	**86.1**	**85.0**

Table 4. Comparison on YouTube-VOS validation sets. $\mathcal{G}$ is an average of $\mathcal{J}_\mathcal{S}$, $\mathcal{J}_\mathcal{U}$, $\mathcal{F}_\mathcal{S}$, and $\mathcal{F}_\mathcal{U}$. * denotes our reproduced result using our training setup.

reduce the input image to 480p resolution. We measured region similarity ($\mathcal{J}_\mathcal{S}, \mathcal{J}_\mathcal{U}$) and contour accuracy ($\mathcal{F}_\mathcal{U}, \mathcal{F}_\mathcal{U}$) for 65 of seen and 26 of unseen object categories separately. In Table 4, we compare HMMN with state-of-the-art methods on YouTube-VOS 2018 and 2019 validation sets. Note that only CFBI [50] officially reported for comparison on YouTube-VOS 2019 validation set, so we additionally report our reproduced results of STM [33] and KMN [39] using our training setup. As shown in Table 4, our HMMN surpasses the state-of-the-art methods in all official metrics on both YouTube-VOS 2018 and 2019.

Qualitative Comparison. Fig. 7 shows qualitative comparison with STM [33] and KMN [39]. In the figure, STM [33] almost failed to predict target objects when multiple similar objects have appeared or several occlusion occurred (DAVIS example). KMN [39] failed to predict a very small object (YouTube-VOS example). On the other hand, our HMMN predicted the target objects accurately in the challenging cases. More qualitative results are provided in the supplementary material.

4.3. Ablation Experiments

Module ablation. We conduct an ablation study on our two proposed memory matching modules to demonstrate the efficacy of those. We also compare our kernel guided memory matching with the kernelization method proposed in KMN [39]. As shown in Table 5, our kernel guidance is more effective than one from KMN, and the use of fine-scale memories through top-k guided memory module greatly boosts the performance to the state-of-the-art.

Temporal stability ($\mathcal{T}$). To validate the effectiveness of our HMMN on temporal smoothness quantitatively, we evaluate temporal stability ($\mathcal{T}$) [35] on DAVIS 2016 validation set. STM [33], KMN [39], and our HMMN achieved $\mathcal{T}$ scores (lower is better) of 17.2%, 15.2%, and 13.0%, respectively. This implies that our method significantly improves temporal stability over STM and KMN.

K* [39]	**K**	**T**	Time	DAVIS 2016	DAVIS 2017	YouTube-VOS 2018	YouTube-VOS 2019
			0.07s	89.2	82.2	79.2	79.3
✓			0.07s	89.5	83.3	79.8	80.0
	✓		0.07s	90.0	83.1	80.7	80.9
		✓	0.10s	**90.8**	83.6	81.1	81.2
✓		✓	0.10s	**90.8**	84.1	81.7	81.8
	✓	✓	0.10s	**90.8**	**84.7**	**82.6**	**82.5**

Table 5. **Module ablation study**. We report $\mathcal{J}\&\mathcal{F}$ and $\mathcal{G}$ scores for DAVIS and Youtube-VOS, respectively. The run-time is measured on DAVIS 2016 validation set. The baseline model is STM [33]. K* [39] denotes the kernelization proposed in [39], and **K** and **T** indicate our kernel guided memory matching and top-k guided memory matching modules, respectively.

k-pixel selection strategies. To validate the effectiveness of our top-k guidance (§3.2), we study various strategies to sample k memory pixels. As can be seen in Table 6 (a), fine-scale memory with simple sampling methods (random, stride) do not provide consistent improvement over the baseline (k=0). However, fine-scale memory with our top-k guidance yields significant performance improvement even with a small number of k.

The effect of k. We further study the effect of k during both training and inference by increasing the number of k from 32 to ∞. Here, k=∞ indicates using *dense* memory without sampling. As shown in Table 6 (b), using a dense fine-scale memory either in training and/or inference degrades the overall performance compared to using top-k sampled memory. We conjecture that, in fine-scale, the feature is not robust enough for the global and dense matching. In this case, top-k guidance could be beneficial to rejecting noises by restricting the search space into few reliable options. While our default setting is to set k=32 for both training and inference, we observed that the performance could be further improved by tuning k.

Dropout for high-resolution memory. Table 6 (c) shows the effect of dropout in top-k guided memory module. As we discussed in §3.2, our dropout strategy makes our network learn with hierarchical memories effectively.

Fine-scale memory management. Table 6 (d) shows that we can further boost our performance by exploiting fine-scale memories from the intermediate frames sampled from every 5 frames. However, this configuration requires too much GPU memory to store memory features, while performance improvement is marginal. We use the first and previous frames for fine-scale memory by default to run HMMN. Note that we use the intermediate frames for the coarse-scale memory.

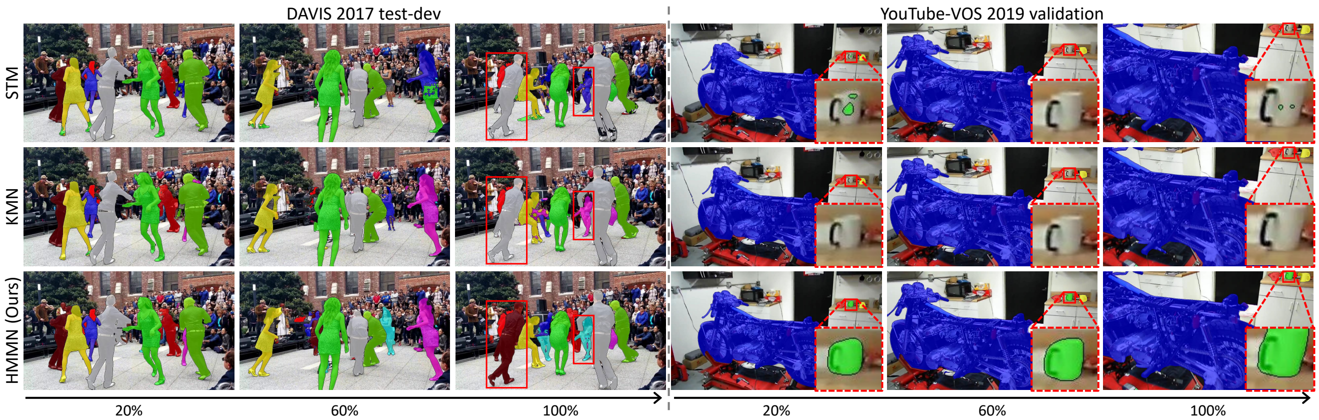

Figure 7. Qualitative comparison on DAVIS 2017 test-dev, and Youtube-VOS 2019 validation sets. We compare HMMN with STM [33] and KMN [39]. We marked significant improvements from STM and KMN using red boxes.

	k	
	0	83.1/80.9
random	64	82.9/80.5
	128	82.2/80.3
	256	81.9/81.1
stride	$2{\cdot}8^2$	82.4/80.3
	$2{\cdot}12^2$	83.3/80.5
top-k	8	82.2/81.4
	16	83.2/82.0
	32	**84.7/82.4**

(a) Comparison on various k-pixel selection strategies.

	k	Training 32	∞
Inference	32	84.7/**82.5**	82.3/81.5
	64	**84.9/82.5**	82.6/81.5
	128	84.6/82.1	82.8/81.4
	256	84.3/81.9	82.7/81.5
	∞	82.0/78.8	82.1/80.5

(b) Results with large number of k.

Training	
w/o Dropout	81.5/80.7
w/ Dropout	**84.7/82.5**

(c) Experimental results with and without dropout in top-k memory matching module.

Fine-scale memory frames	
First & Prev.	84.7/**82.5**
Every 5 frames	**84.9/82.5**

(d) Comparison of memory management strategies for top-k memory matching module.

Fine-scale memory stages	
None	83.1/80.9
`res2`	83.2/82.1
`res3`	83.2/81.6
`res2` & 3	**84.7/82.5**

(e) Memory stages for top-k memory matching module.

Window size (s)	
3×3	84.0/**82.5**
5×5	84.4/**82.5**
7×7	84.7/**82.5**
9×9	**84.8**/82.4
11×11	84.7/82.3
∞	84.2/81.0

(f) Window sizes in kernel guided memory matching module.

Standard deviation (σ_{init})	
1	83.6/82.0
3	**84.7/82.5**
5	84.0/82.4
7	83.8/82.4
9	83.8/82.2
11	83.7/81.9

(g) Standard deviations of Gaussian kernel in kernel guided memory matching module.

Table 6. **Ablation Study.** For each setting, we report results of $\mathcal{J}\&\mathcal{F}$ and $\mathcal{G}$ scores on DAVIS 2017 and YouTube-VOS 2019 validation sets, respectively.

Fine-scale memory stages. We ablate hierarchical memory stage-by-stage, and the results are given in Table 6 (e). As shown in the table, using memory hierarchies in both stages shows the best performance. If the hierarchical memory is used only in a single stage, interestingly, taking finer-scale memory (*i.e.*, `res2` stage) achieves better performance even we reduced the number of k to $k/4$ at `res2` stage. It is thought that the finer-scale memory can provide more complementary information to the memory from the coarsest scale.

Window size s & standard deviation σ_{init}. Tables 6 (f) and 6 (g) show the parameter search experiments for guidance kernel (§3.1). Choosing a too large and too small value for s and σ_{init} has degraded the performance. Therefore, we select proper window size s and standard deviation of Gaussian kernel σ_{init} as 7×7 and 3, respectively.

5. Conclusion

We presented two advanced memory matching modules that exploit temporal smoothness and hierarchical memory effectively. We demonstrated the efficacy of our HMMN through extensive experiments and achieved state-of-the-art performance on all evaluated benchmarks while keeping a fast run-time. We believe that our proposed two memory matching modules can be further extended to other matching-based vision applications such as video saliency detection, video instance segmentation, and semantic correspondence.

Acknowledgement. This work was supported by the Industry Core Technology Development Project, 20005062, Development of Artificial Intelligence Robot Autonomous Navigation Technology for Agile Movement in Crowded Space, funded by the Ministry of Trade, industry & Energy (MOTIE, Republic of Korea).

References

[1] Linchao Bao, Baoyuan Wu, and Wei Liu. Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. In *CVPR*, pages 5977–5986, 2018. 2, 6

[2] 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 *ECCV*, 2020. 6, 7

[3] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *CVPR*, pages 221–230, 2017. 2

[4] Xi Chen, Zuoxin Li, Ye Yuan, Gang Yu, Jianxin Shen, and Donglian Qi. State-aware tracker for real-time video object segmentation. In *CVPR*, pages 9384–9393, 2020. 2

[5] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, pages 686–695, 2017. 2

[6] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2014. 5

[7] Kevin Duarte, Yogesh S. Rawat, and Mubarak Shah. Capsulevos: Semi-supervised video object segmentation using capsule routing. In *ICCV*, October 2019. 2

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 5

[9] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998. IEEE, 2011. 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[11] Ping Hu, Jun Liu, Gang Wang, Vitaly Ablavsky, Kate Saenko, and Stan Sclaroff. Dipnet: Dynamic identity propagation network for video object segmentation. In *WACV*, pages 1904–1913, 2020. 2

[12] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing. Maskrnn: Instance level video object segmentation. In *NIPS*, pages 325–334, 2017. 2

[13] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G Schwing. Videomatch: Matching based video object segmentation. In *ECCV*, pages 54–70, 2018. 1, 2

[14] Xuhua Huang, Jiarui Xu, Yu-Wing Tai, and Chi-Keung Tang. Fast video object segmentation with temporal aggregation network and dynamic template matching. In *CVPR*, pages 8879–8889, 2020. 2

[15] Joakim Johnander, Martin Danelljan, Emil Brissman, Fahad Shahbaz Khan, and Michael Felsberg. A generative appearance model for end-to-end video object segmentation. In *CVPR*, pages 8953–8962, 2019. 2

[16] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for video object segmentation. *International Journal of Computer Vision*, 127(9):1175–1197, 2019. 1, 2

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[18] 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 *ICML*, pages 1378–1387, 2016. 2

[19] Zihang Lai, Erika Lu, and Weidi Xie. Mast: A memory-augmented self-supervised tracker. In *CVPR*, pages 6479–6488, 2020. 2

[20] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *ECCV*, pages 90–105, 2018. 2, 6

[21] Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module. In *ECCV*, 2020. 1, 2

[22] Yuxi Li, Ning Xu, Jinlong Peng, John See, and Weiyao Lin. Delving into the cyclic mechanism in semi-supervised video object segmentation. In *NIPS*, 2020. 1, 2

[23] Yongqing Liang, Xin Li, Navid Jafari, and Qin Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In *NIPS*, 2020. 1, 2, 7

[24] Huaijia Lin, Xiaojuan Qi, and Jiaya Jia. Agss-vos: Attention guided single-shot video object segmentation. In *ICCV*, October 2019. 7

[25] 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 *ECCV*, pages 740–755. Springer, 2014. 5

[26] Daizong Liu, Shuangjie Xu, Xiao-Yang Liu, Zichuan Xu, Wei Wei, and Pan Zhou. Spatiotemporal graph neural network based mask reconstruction for video object segmentation. In *AAAI*, 2021. 7

[27] Xinkai Lu, Wenguan Wang, Martin Danelljan, Tianfei Zhou, Jianbing Shen, and Luc Van Gool. Video object segmentation with episodic graph memory networks. In *ECCV*, 2020. 1, 2, 6, 7

[28] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, pages 565–580. Springer, 2018. 2, 6

[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, 2019. 2

[30] Tim Meinhardt and Laura Leal-Taixé. Make one-shot video object segmentation efficient again. In *NIPS*, 2020. 2, 6, 7

[31] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016. 2

[32] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *CVPR*, pages 7376–7385, 2018. 1, 2, 6

[33] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, October 2019. 1, 2, 3, 4, 5, 6, 7, 8

[34] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, pages 2663–2672, 2017. 1, 2

[35] 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*, pages 724–732, 2016. 2, 6, 7

[36] 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. 2, 6

[37] Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Learning fast and robust target models for video object segmentation. In *CVPR*, pages 7406–7415, 2020. 6, 7

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[39] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In *ECCV*, 2020. 1, 2, 3, 5, 6, 7, 8

[40] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):717–729, 2015. 5

[41] Jae Shin Yoon, Francois Rameau, Junsik Kim, Seokju Lee, Seunghak Shin, and In So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *CVPR*, pages 2167–2176, 2017. 2

[42] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015. 2

[43] 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 *CVPR*, pages 9481–9490, 2019. 1, 2

[44] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017. 2

[45] Jingdong Wang, Huaizu Jiang, Zejian Yuan, Ming-Ming Cheng, Xiaowei Hu, and Nanning Zheng. Salient object detection: A discriminative regional feature integration approach. *International Journal of Computer Vision*, 123(2):251–268, 2017. 5

[46] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. Ranet: Ranking attention network for fast video object segmentation. In *ICCV*, October 2019. 6

[47] Huaxin Xiao, Bingyi Kang, Yu Liu, Maojun Zhang, and Jiashi Feng. Online meta adaptation for fast video object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(5):1205–1217, 2020. 2

[48] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *ECCV*, pages 585–601, 2018. 2, 6

[49] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, pages 6499–6507, 2018. 1, 6

[50] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *ECCV*, 2020. 1, 2, 6, 7

[51] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by multi-scale foreground-background integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[52] Xiaohui Zeng, Renjie Liao, Li Gu, Yuwen Xiong, Sanja Fidler, and Raquel Urtasun. Dmm-net: Differentiable mask-matching network for video object segmentation. In *ICCV*, October 2019. 2

[53] Lu Zhang, Zhe Lin, Jianming Zhang, Huchuan Lu, and You He. Fast video object segmentation via dynamic targeting network. In *ICCV*, October 2019. 1

[54] Yizhuo Zhang, Zhirong Wu, Houwen Peng, and Stephen Lin. A transductive approach for video object segmentation. In *CVPR*, pages 6949–6958, 2020. 2

[55] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2