SPACE TIME RECURRENT MEMORY NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers have recently been popular for learning and inference in the spatial-temporal domain. However, their performance relies on storing and applying attention to the feature tensor of each frame. Hence, their space and time complexity increase linearly as the sequence length grows, which could be very costly for long videos. We propose a novel visual memory network architecture for the learning and inference problem in the spatial-temporal domain. We maintain a fixed set of memory slots in our memory network and explore different designs to input new information into the memory, combine the information in different memory slots and decide how to discard old information. Finally, this architecture is benchmarked on the video object segmentation and video prediction problems. Through the experiments, we show that our memory architecture can achieve competitive results compared to state-of-the-art transformer-based methods while maintaining constant memory capacity independent of sequence length.

1 INTRODUCTION

Network architectures for spatial-temporal reasoning are of fundamental importance to visual systems. Most real-life reasoning problems are presented in space-time, including but not limited to autonomous driving, video object segmentation, navigation, planning, etc. A great earlier example of a spatial-temporal reasoning model is the convolutional LSTM (Shi et al., 2015), where a convolutional network generates a spatially indexed memory tensor, which is updated as one iterates through frames, and appropriate output can be computed at each given time frame.

There are two main benefits to the recurrent memory architecture. First, it is **online**, meaning the system could generate the appropriate output at any given time in the sequence. The system would do so by taking into account the memory which has evolved through all the previous frames, hence potentially taking into account long-term information. Second, the memory size is **constant**, as the memory is updated at each time with the new information from each frame. Such properties make them ideal for embodied systems that require real-time processing.

However, one drawback of the convolutional LSTM or similar approaches is that the memory may confuse itself over *multi-modality*. Since there was only one feature tensor stored as memory, the system often has to face a dilemma when two equally valuable templates present themselves: they have to either only remember one of them, or somehow combine them in some manner. However, the combination may become more ambiguous when many diverse templates need to be remembered and lead to reduced performance. Another drawback is that they do not handle motion well. Although the memory could be indexed spatially, such indexing is rigid and once the object has moved in the space, the stored memory at the indexed location cannot move with the object motion.

More recently, spatio-temporal transformers (STM) (Oh et al., 2019) were proposed as a different approach for spatio-temporal reasoning. Motivated by transformers from natural language processing, in STM each frame forms two spatially-indexed feature tensors called the query and the key. Then, the convolutional features from each new frame is also converted to its query and key tensors, which are used to match with each previous frame to compute attention weights, used in a weighted sum of the spatially-indexed *value* tensor of each frame. STM has achieved state-of-the-art results in an important and difficult spatio-temporal reasoning problem: Video Object Segmentation (VOS).

However, STM needs to compute and store tensors for each frame, leading to significantly increased time and memory cost for long videos. The walkaround in (Oh et al., 2019) was to store one set of memory tensors for every 5th frame, but this still would not extend easily to e.g. videos with thou-

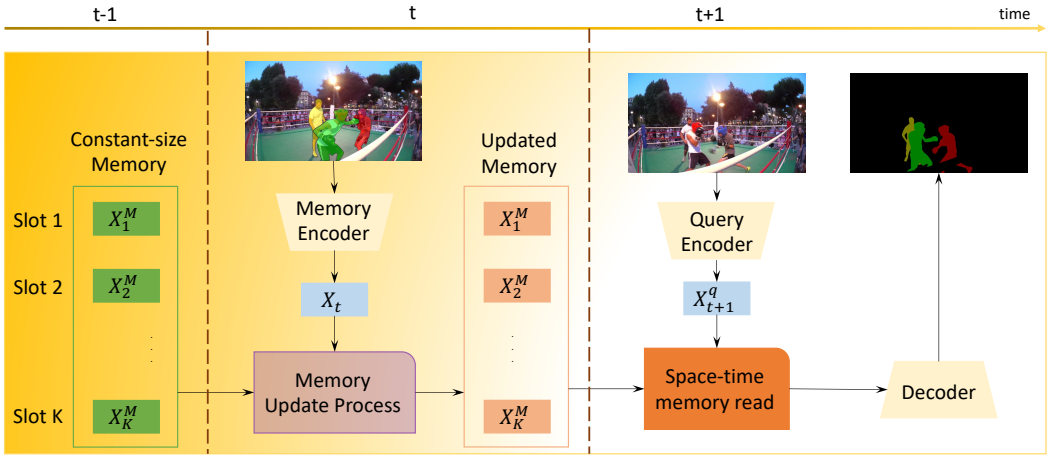

Figure 1: **The overview of our framework**. Assuming a constant-size memory with K slots, our framework consists 2 stages. 1) Memory Update: in this stage, the network generates a new feature map X_t based on the current frame with the corresponding masks using the memory encoder. This feature map is then used to update the memory (Sec. 3.2). 2) Inference stage: a feature map X^q_{t+1} is generated from the query image using the query encoder and goes through the space-time memory read module to access the memory and retrieve object-related information to create a new feature map. This feature map is finally decoded into our predicted mask (which will be then used to update the memory again). (Best viewed in color)

sands of frames. In this paper, we propose the Space-Time Recurrent Memory Network (STReMN, pronounced as "strem") which attempts to retain a *constant-sized* memory while enabling online processing for the spatio-temporal reasoning problem. Our network maintains a memory with a fixed amount of separate memory slots which are updated through time. When new information comes from a new frame, the system would decide whether to use it to update some current memory slots, and/or throw some previous slots out of the memory. Hence, compared to transformer-based networks which do not perform aggregation of memory from different frames, network design for our model is significantly more challenging. Yet, we believe that this kind of memory structure has more potential to scale up to long videos beyond a few seconds, and touches upon more fundamental challenges in memory organization that is valuable to many spatio-temporal reasoning problems. In the paper we explore multiple different design choices which shed light on the designs of such a memory network applied to practical tasks such as VOS and video prediction.

Our contributions can be summarized as follows:

- We introduce a novel space-time memory network which has a constant number of slots. The proposed memory network can handle a long sequence of images efficiently without increasing the size of the memory when the length of the image sequence grows.
- We explore several different memory update rules for the aggregation of information from multiple memory slots.
- Experiments show that our model is able to summarize and compress past information into a constant-size memory effectively. With our memory network that adopts constant-sized memory, we are able to match state-of-the-art performance of transformer-based models on the VOS task and outperform other recent methods on video prediction task.

2 Related Work

Implicit Memory. Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) improves over vanilla recurrent networks (RNN) by allowing the cell state (memory) to stay constant if there is no additional input. It also includes input and forget gates which allow information to be filtered before being used to update the memory. These gates are usually implemented with fully connected layers and hence cannot be applied to spatial-temporal sequences. Shi et al. (2015) re-

place the linear layers with convolutional layers instead hence generating spatially-indexed memory that can be used for spatio-temporal inference. PredRNN (Wang et al., 2017) modify RNN by allowing the memory state to flow vertically through stacked RNN layers and horizontally through time. Causal LSTM (Wang et al., 2018a) improves upon PredRNN by proposing the Highway Unit allowing the gradient to flow quickly to long-range input. E3D-LSTM Wang et al. (2018b) uses 3d conv and attention module to improve the RNN. MIM (Wang et al., 2019) models the stationary and non-stationary properties in the spatial-temporal dynamics by exploiting differential signal between adjacent recurrent states. However, most of these architectures only have a single memory vector/tensor which struggles to capture multiple appearances of objects in video reasoning problems (Kim et al., 2018).

Explicit Memory. Neural Turing Machine (NTM) (Graves et al., 2014) extended the traditional RNN with an external memory. NTM architecture includes a controller and a memory bank. These two components interact with each other through multiple read and write heads similar to the attention model in transformers. The controller receives information, interacts with the memory through read and write heads to retrieve related information, remove old information, and save new ones if useful. Based on the retrieved information, a corresponding response is returned. However, it is non-trivial to scale NTM's memory for visual tasks which require spatial information.

Space Time Memory (STM). STM (Oh et al., 2019) is a transformer-based network that utilizes an convolutional encoder to encode the features of each frame to 2 tensors, key and value. In the reading stage, STM matches the query frame and template frames at potentially different spatial locations. The computed attention score are then used for a weighted sum of the features from each frame. STM does not update existing memory slots and simply concatenates the feature tensors of each new frame. One advantage of this simple mechanism is that the information on each template is not contaminated by other ones. However, it is potentially time and memory consuming since old templates are never removed and the memory grows linearly w.r.t. the number of frames. To reduce the time/memory costs, STM uses a simple heuristic to add a new template every 5 frames, but its time/memory consumption can still be a significant issue in longer sequences.

Memory compression. Dai et al. (2019) improves upon the transformer network by storing previous hidden states in a fixed-size queue. The queue helps the model to trade-off the context length for the memory size. Rae et al. (2020) trade-offs the granularity of the memory for the context length by compressing multiple hidden states before adding them to the memory. Therefore, the queue can keep more information, given the same size queue as in Dai et al. (2019). In contrast to these approaches, our method additionally aggregates information across different memory slots, which allows the memory to keep any useful information in the old memory slots before they are removed. GC (Li et al., 2020) compresses the memory of STM by constructing a global memory by averaging the memory of past frames and rearranging the order of the attention operation. While GC maintains a constant-size memory like our method, its performance on VOS is not as strong as the original STM's performance possibly due to the use of simple average operation. PAM (Wang et al., 2021) also introduces a compact memory representation for the STM framework. Memory updates are performed only when significant changes between frames exist to avoid updating the memory with redundant information. Also, instead of performing memory updates with full images, pixel-wise memory updates are utilized in order to store only relevant information to the target object in the memory. Although PAM builds more efficient memory representations than STM does, its memory size still increases over time as there is no deletion operation. In contrast to PAM, our method maintains a fixed-size memory regardless of the video length, while still achieving competitive performance.

Episodic graph memory network (EGM). EGM (Lu et al., 2020) represents the memory as a fully connected graph with each node being a template. The model performs k steps reasoning using the memory with the query frame. This process allows the interaction among memory nodes with the query, and thus builds up a high order function of the nodes and the query. However, this model does not provide a way to update the memory. Given a new query frame, the model uses the first (i.e ground-truth) frame and previous segmented frame and samples $N-2$ segmented frames from the support set including past segmented frames to re-initialize the memory. In other words, the memory is reset each time making an inference. This makes EGM several times slower than our approach. Besides, resetting the memory too frequently will throw away valuable information built up from previous reasoning steps. Another disadvantage is that there are always exactly N different frames

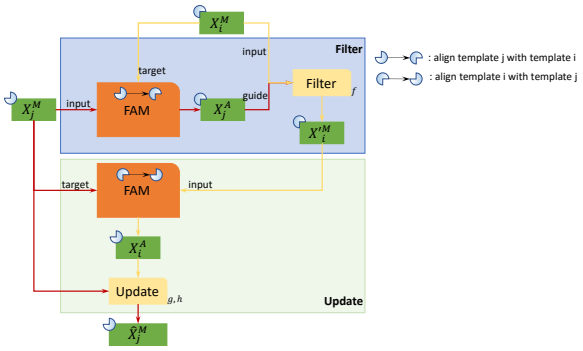

Figure 2: **Fusion Cell.** The Fusion Cell is used to distill information from X_i^M to X_j^M using the FAM and gating function, Filter and Update.

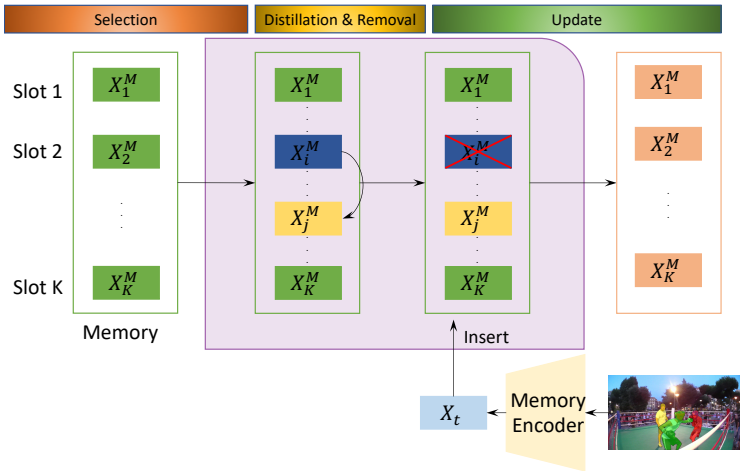

Figure 3: **Memory Update Process**. The update process includes 3 steps. **Selection step**: 2 templates, X_i^M and X_j^M where $i < j$, are chosen from the memory based on their similarity. **Distillation & Removal step**: the information from X_i^M is distilled into X_j^M using the Fusion Cell, and X_i^M is purged from the memory. **Update step**: a template X_t is generated using our Memory Encoder and the current frame together with its extra information such as objects' masks, optical flow, or previous frames if needed. This template is added to the end of the memory (i.e after X_K^M).

from the video in the memory at a time. Therefore, when the video becomes longer, the memory can miss lots of important information.

3 METHODS

3.1 OVERVIEW

Our framework includes a Query Encoder (QE), a Memory Encoder (ME), a Decoder (D), and a fixed-size memory $Mem = \{X_1^M, X_2^M, \ldots, X_K^M\}$. QE outputs a query feature map X_t^q with I_t as input. ME could be the same as QE if there is no additional information, but if there is additional information M_t (e.g. object mask) that should go into the memory along with the image, then ME takes (I_t, M_t) as input and produces a spatially-indexed feature tensor X_t^M. Both QE and ME are implemented using a CNN. The new memory feature X_t^M at each step is fed into the memory update process (Sec. 3.2). To generate the output at step $t+1$, we use the query feature X_{t+1}^q to read the memory using attention-based operation described in Sec. 3.3. The readout feature map is decoded to desired outputs using a Decoder implemented with a CNN upsampling layers in a fashion similar to U-Net. The following subsections describe our memory update and inference process in detail.

3.2 MEMORY UPDATE PROCESS

Our memory update process is depicted in Fig. 3. Let K and n be the maximum number and the current number of slots in the memory. If $n < K$, new templates are simply added to the memory. If $n = K$, then a template must be removed from the memory to create an empty slot for the new template. However, removing a template completely from the memory might be detrimental because the removed template might carry useful information that can only be observed in the corresponding frame. Hence, we want to distill the important information from the removed template to others. Besides, if some of the frames are ground truth and hence most reliable, it might be useful to always keep that frame in the memory.

- **A**: To remove the oldest template that's not ground truth and use it to update other templates.
- **B**: To choose the newest frame to remove and use it to update the remaining templates.
- **C**: To choose slots i and j such that and $(i, j) = \arg\min_{i,j} S(X_i^M, X_j^M)$ where $S(X_i, X_j)$ is a similarity function between 2 templates. In other words, X_i^M and X_j^M are the least similar pair of templates in the memory.
- **D**: To choose slots i and j such that $(i, j) = \arg\max_{i,j} S(X_i^M, X_j^M)$. In other words, X_i^M and X_j^M are the most similar pair of templates in the memory.

The similarity function between 2 templates $X_i^M, X_j^M \in R^{HW \times d}$ is defined based on the many to one matching between the keys of the 2 corresponding frames:

$$S(X_i^M, X_j^M) = \frac{1}{HW} \sum_p \max_q c(X_i^M(p), X_j^M(q)) \tag{1}$$

$$c(X_i^M(p), X_j^M(q)) = \frac{X_i^M(p) \cdot X_j^M(q)}{||X_i^M(p)||||X_j^M(q)||} \tag{2}$$

where $X_i^M(p)$ extracts the feature at spatial location p. S is the average similarity between the most similar location pairs in X_i^M and X_j^M. This approach allows the same object that are in different locations in the two memory slots to match with each other.

The Fusion cell (Fig. 2) is used to distill information from the removed template X_i^M to the target template X_j^M. Again we need to deal with the issue that the pixels in these 2 templates might not be aligned with each other. Besides, the removed template might contain information that is no longer useful, e.g. object parts that have moved out of the frame. Hence, we need to filter the information in X_i^M first, before updating X_j^M with it. In the filter stage, the noisy information is removed from X_i^M using X_j^M as our guide. To do that, X_j^M is aligned with X_i^M with an attention module named FAM, which aligns pixel information in X_i^M to different pixels in another slot:

$$FAM(X_i^M, X_j^M) = Softmax(X_i^M (X_j^M)^T) X_j^M \tag{3}$$

After that, a gating function is applied to filter out unnecessary features from X_i^M. We describe the entire fusion cell as follows:

$$X_j^A = FAM(X_i^M, X_j^M) \tag{4}$$

$$r = \sigma(f([X_i^M, X_j^A])) \tag{5}$$

$$X_i'^M = r \odot X_i^M \tag{6}$$

where f is implemented with a CNN, σ is the sigmoid function, and $\odot$ is the Hadamard product.

This fusion cell first aligns the pixels in X_j^M to the pixels in X_i^M, then combine the information into a gating function to filter out unnecessary features in X_i^M. The filter step helps to remove the noisy information from the template X_i^M before actually using it to update X_j^M.

In the subsequent update stage, $X_i'^M$ is first aligned to X_j^M to create an aligned template X_i^A. Then X_i^A is used to update template X_j^M as follows:

$$X_i^A = FAM(X_j^M, X_i'^M), z = \sigma(h([X_i^A, X_j^M])) \tag{7}$$

$$n = tanh(g([X_i^A, X_j^M])), \hat{X}_j^M = (1 - z) \odot X_j^M + z \odot n \tag{8}$$

where h, and g are implemented using CNN.

Note that in our framework, the update process happens among templates within the memory. The newest template is only copied to the memory after the update process takes place and does not directly interfere with that process. This property makes our model different from memory modules such as LSTM (Hochreiter & Schmidhuber, 1997) or NTM (Graves et al., 2014). Specifically, let $\{X_t^M\}$ be the memory at time t and X_t be the template of the newest frame. LSTM and NTM update formulae are functions between the new input and the current memory:

$$\{X_t^M\} = G(\{X_{t-1}^M\}, X_t) \tag{9}$$

Instead, we keep the new template X_t intact in the memory and choose to use one of the memory cells to update. An intuition for this choice is that the new information is fresh and might be the most relevant to instant predictions, hence it might be nice to keep it before blending it with the long term information to solidify our memory and build up the knowledge base about the world. This modification also allows the reasoning within the memory and create higher order interaction among memory cells. In section 4.1.3 we show ablation experiments comparing these two approaches.

3.3 Inference Stage

In the inference stage, the query frames goes through the Query Encoder to be encoded into a feature map and converted into a key-value pair (k^q, v^q).Similarly, each template in the memory is converted into a key-value pair and all templates are concatenated to form (k^M, q^M). Unlike k^q and v^q which only contain information from query frames, k^M and q^M carry information of every frame in the past that is condensed into a fixed number (i.e K) of templates in the memory thanks to our memory update module. Then each pixel i of the query template is densely matched with every pixel j of memory's templates through the attention mechanism similar to (Oh et al., 2019):

$$y_i = \left[\frac{1}{\Sigma_j \exp(k_i^{q\top} k_j^M)} \exp(k_i^{q\top} k_j^M) v_j^M, v_i^q \right] \tag{10}$$

where $1 \leq i \leq HW$, $1 \leq j \leq KHW$, and $[\cdot, \cdot]$ is the concatenation. $\mathbf{y}$ is then decoded into the output with a standard U-Net decoder.

4 Experiments

We conduct experiments on two difficult video tasks, video object segmentation and video prediction. They are described in the subsequent subsections.

4.1 Video object segmentation

The VOS problem is chosen as a benchmark because of following reasons. First, VOS requires both long and short term memory to segment the objects efficiently. Second, the variation of appearance of the object throughout the video is significant, which requires the network to keep a diverse memory. Finally, The segmentation task requires the network to preserve a huge number of spatial details, which is a challenging task for memory networks.

4.1.1 Implementation Details

Implementation. The architectures of the encoder and decoder are the same as those of Oh et al. (2019). However, we replaced batch normalization (Ioffe & Szegedy, 2015) layers with group normalization (Wu & He, 2018) in our backbone - Resnet 50 (He et al., 2016). This made the training more stable because of the small batch size due to memory constraints. The network uses 5 memory blocks including: 1 block for the ground truth (1st frame), 1 block for the latest frame, and 3

Table 1: Results on the DAVIS 2017 validation set compared with other methods that are only trained or fine-tuned on DAVIS 2017. **N/A**: not a memory-based method.

			Validation		
Method	**Backbone**	**Memory size**	**J**	**F**	**Avg**
FEELVOS (Voigtlaender et al., 2019)	DeepLab v3	N/A	65.9	72.3	69.1
STM (Oh et al., 2019)	Resnet 50	linear growth	69.2	74.0	71.6
KMN (Seong et al., 2020)	Resnet 50	linear growth	74.2	77.8	76.0
CFBI (Yang et al., 2020)	Resnet 101	N/A	72.1	77.7	74.9
GC (Li et al., 2020)	Resnet 50	constant	69.3	73.5	71.4
STReMN (Ours)	Resnet 50	constant	**76.1**	**81**	**78.5**

Table 2: Results on the DAVIS 2017 dataset while fine-tuning on both DAVIS and Youtube.

			Validation			Test-dev		
Method	**Backbone**	**Memory size**	**J**	**F**	**Avg**	**J**	**F**	**Avg**
STM (Oh et al., 2019)	Resnet 50	linear growth	79.2	84.3	81.8	69.3	75.2	72.2
STM (linear) (Oh et al., 2019)	Resnet 50	constant	77.7	-	-	-	-	-
KMN (Seong et al., 2020)	Resnet 50	linear growth	80.0	85.6	82.8	74.1	80.3	77.2
EGM (Lu et al., 2020)	Resnet 50	linear growth	80.2	85.2	82.8	-	-	-
CFBI (Yang et al., 2020)	Resnet 101	N/A	79.1	84.6	81.9	71.1	78.5	74.8
SwiftNet (Wang et al., 2021)	Resnet 50	linear	78.3	83.9	81.1	-	-	-
A-Game (Johnander et al., 2019)	Resnet 101	N/A	67.2	72.7	70	-	-	-
FEELVOS (Voigtlaender et al., 2019)	DeepLab v3	N/A	69.1	74.0	71.5	55.1	60.4	57.8
STReMN (Ours)	Resnet 50	constant	79.1	84.3	81.7	69.2	75.5	72.4

blocks for intermediate frames. Post-processing was also employed to remove some false positive predictions that are far away from the object in the previous frame.

Training. The model was trained in 2 stages: pre-training with images and fine-tuning with videos. In the **pre-training stage**, consistent with prior work, we created short clips consisting 5 frames by using images from COCO (Lin et al., 2014), ECSSD (Yan et al., 2013), SBD (Hariharan et al., 2011), MSRA10k (Cheng et al., 2015), and HKUIS (Li & Yu, 2015). We minimized cross entropy loss using Adam with learning rate 10^{-4}. In the **fine-tuning stage**, we randomly extracted video clips consisting of 50 frames from videos in the DAVIS training set. We fine-tuned the model with the Adam optimizer with a cosine learning rate scheme between $[10^{-5}, 10^{-7}]$. The image resolutions for training in both stages were 384×384.

4.1.2 Results on the DAVIS 2017 Dataset

The DAVIS dataset is one of the primary benchmarks for the VOS task and the most difficult one. There are 2 versions: single object - DAVIS 2016 and multiple objects - DAVIS 2017. Single-object segmentation is significantly easier hence we do not report on that task. DAVIS 2017 includes 60 videos, 30 videos, and 30 videos in the training, validation, and test-dev sets respectively. The objects range from common classes such as human or dog to rare classes such as string, car wheel, and drone. The number of frames in each video can be up to 130 frames.

Table 1 shows the quantitative results on the DAVIS 2017 validation set. Our result outperforms most other approaches when trained only on DAVIS, and on par with STM with more datasets. Note we only compare our methods with other methods that was trained on the training set and tested on the validation and testing sets. The online fine-tuning methods (Perazzi et al., 2017; Caelles et al., 2017; Voigtlaender & Leibe, 2017; Bhat et al., 2020; Bao et al., 2018; Li & Loy, 2018; Hu et al., 2018; Robinson et al., 2020), which fine-tune their models on each testing video separately, are not general memory models and hence orthogonal to our work.

Tab. 2 shows our results on the DAVIS 2017 validation and test-dev sets, when fine-tuned on both DAVIS and the YouTube datasets. Our model with a fixed size memory outperforms the constant-memory version of STM (Oh et al., 2019) that only uses the first and the previous frame, and is comparable with the linear-memory STM which adds a new template to the memory for every 5 frames. Among the recent methods better than us, KMN (Seong et al., 2020) is mainly about a data augmentation that we did not use, (Seong et al., 2020; Singh & Lee, 2017) which provided most of the improvement. CFBI Yang et al. (2020) used Resnet 101 to extract backbone feature. Besides, CFBI also utilizes an ASPP module Chen et al. (2018) to extract multi-scale features.

These augmentation and additional modules are orthogonal to our contribution and can be added to our architecture easily. EGM Lu et al. (2020), has to keep all previous frames' features to initialize the graph at each time step and thus increases the memory size linearly with the length of the video.

Table 3: The *I2M* model uses the new template to update current memory cells. In contrast, the *M2M* model uses another memory cell to update current cells. Both models are evaluated on Davis 2017 validation set

Methods	**J**	**F**	**Avg**
Input to memory (I2M)	69.2	74.7	72.0
Memory to memory (M2M)	76.1	81.0	78.5

Table 4: Results on DAVIS 2017 validation set for different memory update strategies.

Memory variants	**J**	**F**	**Mean**
Oldest vs remaining (**A**)	74.5	80.1	77.3
Latest vs remaining (**B**)	73.0	78.5	75.7
Least similarity pair (**C**)	76.1	81.0	78.5
Most similarity pair (**D**)	74.9	80.1	77.5

Figure 4: Qualitative results. (Best viewed in color)

4.1.3 Memory Analysis

We evaluated different memory update variants (see Sec. 3.2) on the DAVIS 2017 validation set. The results are showed in Table 4. We observe that the results of **C** and **D** are better than those of **A** and **B**. This can be explained that using one template to update/enhance all the others can create too much overlapped information. In the worst case, if the chosen template contains errors, these errors will be spread out over the whole memory.

Additionally, by choosing the least similar template pair to update, the model yields better performance than that of the model using the most similar template pair to update. This result goes against our intuition that the update process should combine similar templates and keep a diverse set of templates in the memory. One possible explanation is that by choosing the least similar pair of templates (i.e (i, j) such that $i < j$), we also add a prior which encourages the to regularly remove outdated information that are significantly different from other appearances from the memory.

We conduct an experiment on the variant in eq.(9) and show the results in Tab. 3. It shows that our update rule that always remembers the latest frame is significantly better than LSTM-style updates. More ablations, such as the number of memory slots, are shown in the supplementary material.

4.1.4 Qualitative Results

In Fig. 4 , we show some qualitative results on Davis 2017 dataset. The results show that our model is able to capture a variety of object poses, maintain a long-term memory after occlusion, and adapt well to most of the appearance change. The failure cases are discussed in the appendix.

Table 5: Video prediction results.

	Human 3.6			KTH		
Method	MSE/10	MAE/100	SSIM	PSNR	SSIM	LPIPS
ConvLSTM (Shi et al., 2015)	50.4	18.9	0.776	23.58	0.712	0.231
Causal LSTM (Wang et al., 2018a)	45.8	17.2	0.851	-	-	-
MIM (Wang et al., 2019)	42.9	17.8	0.79	-	-	-
E3D-LSTM (Wang et al., 2018b)	46.4	16.6	0.869	-	-	-
PredRNN (Wang et al., 2017)	48.4	18.9	0.781	27.55	**0.839**	0.204
MCnet + Residual (Villegas et al., 2017)	-	-	-	26.29	0.806	-
TrajGRU (Shi et al., 2017)	-	-	-	26.97	0.790	-
DFN (Jia et al., 2016)	-	-	-	27.26	0.794	-
Conv-TT-LSTM (Su et al., 2020)	-	-	-	**27.62**	0.815	0.204
STReMN (Ours)	**37.04**	**14.23**	**0.905**	27.56	0.829	**0.170**

4.2 VIDEO PREDICTION

Another task we chose to benchmark our approach is video prediction. In this task, the model needs to predict N frames in the future given the first M observed frames. Therefore, this task requires the memory to preserve both the spatial information (e.g. object appearance and background texture) and the temporal information (e.g. motion pattern).

Implementation details. As no segmentation mask is available, we use the same encoder for both the query and the memory. The input at step t is the stack of 3 consecutive video frames $(I_{t-3}, I_{t-2}, I_{t-1})$. The stacked video frames allows the memory to keep both appearance and motion patterns. The encoder is implemented with 6 CNN blocks. Each block consists 2 CNN layers followed by group normalization and Leaky RELU. The decoder includes 6 deconvolution blocks. Each block has 2 deconvolution layers followed by group normalization and leaky RELU. The memory module is the same as in the model for VOS and includes 5 slots.

Training. For a fair comparison, our model is not pretrained on other datasets. The model is trained using Adam optimizer with learning rate set to 10^{-3} to minimize the shrinkage loss (Lu et al., 2018) on L_1 and L_2 and the scale invariant gradient loss (Ummenhofer et al., 2017). The final model is obtained using Polyak-Ruppert Averaging.

Human3.6M: The Human3.6M dataset captures complex motion from general human actions. We follow (Guen & Thome, 2020) to use the walking videos with scene S1, S5, S6, S7, and S8 for training and S9 and S11 for testing. Each video is resized to 128×128. In both training and testing, the model needs to predict 4 frames in the future given the first 4 observed frames. Our approach outperforms other memory models by a significant margin on 3 different metrics mean squared error (MSE), mean average error (MAE), and SSIM as shown in Tab.5. Structural similarity index measure (SSIM) is a perceptual-based metric. Higher SSIM shows that our model can produce frames with better structural content.

KTH: This dataset has 25 people performing 6 types of action on 4 different scenes. First 16 people are for training and the rest for testing. The videos are resized to 128×128. In training, the model is provided with the first 10 frames of the video and needs to predict next 20 frames. We achieve comparable results with PredRNN and outperform other methods on PSNR, SSIM, and LPIPS.

5 CONCLUSION

In this paper, we proposed a novel memory network architecture with the potential for scaling up to long videos. We test multiple memory update strategies and choose the one that allows the interactions among memory cells. This allows our model to reason on high order information of past frames. Finally, we showed that our model worked well on two difficult tasks including video object segmentation and video prediction. We achieved the best result on the DAVIS 2017 validation set (when trained only on synthetic data and DAVIS) and competitive results when trained also with YouTube-VOS. Our results on the video prediction also show that our memory architecture outperforms other memory architectures. In the future, we hope that our model can be extended to other tasks such as video reasoning, reinforcement learning, and embodied AI.

REFERENCES

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5977–5986, 2018.

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 *European Conference on Computer Vision*, pp. 777–794. Springer, 2020.

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*, pp. 221–230, 2017.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S. Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015. doi: 10.1109/TPAMI.2014.2345401.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

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*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *arXiv preprint arXiv:1806.04166*, 2018.

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/8bf1211fd4b7b94528899de0a43b9fb3-Paper.pdf.

Joakim Johnander, Martin Danelljan, Emil Brissman, Fahad Shahbaz Khan, and Michael Felsberg. A generative appearance model for end-to-end video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 200–215, 2018.

G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5455–5463, June 2015.

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)*, pp. 90–105, 2018.

Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part X*, volume 12355 of *Lecture Notes in Computer Science*, pp. 735–750. Springer, 2020. doi: 10.1007/978-3-030-58607-2_43. URL https://doi.org/10.1007/978-3-030-58607-2_43.

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 *European conference on computer vision*, pp. 740–755. Springer, 2014.

Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep regression tracking with shrinkage loss. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 353–369, 2018.

Xiankai Lu, Wenguan Wang, Danelljan Martin, Tianfei Zhou, Jianbing Shen, and Van Gool Luc. Video object segmentation with episodic graph memory networks. In *ECCV*, 2020.

Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*, 2018.

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

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*, pp. 2663–2672, 2017.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Learning fast and robust target models for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7406–7415, 2020.

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

Xingjian Shi, Zhourong Chen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 2015:802–810, 2015.

Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Deep learning for precipitation nowcasting: A benchmark and a new model. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/a6db4ed04f1621a119799fd3d7545d3d-Paper.pdf.

Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 3544–3553. IEEE, 2017.

Jiahao Su, Wonmin Byeon, Jean Kossaifi, Furong Huang, Jan Kautz, and Anima Anandkumar. Convolutional tensor-train lstm for spatio-temporal learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13714–13726. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/9e1a36515d6704d7eb7a30d783400e5d-Paper.pdf.

Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5038–5047, 2017.

Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017.

P. Voigtlaender, Yuning Chai, Florian Schroff, H. Adam, B. Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9473–9482, 2019.

Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for the 2017 davis challenge on video object segmentation. In *The 2017 DAVIS Challenge on Video Object Segmentation-CVPR Workshops*, volume 5, 2017.

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 (CVPR)*, pp. 1296–1305, June 2021.

Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 879–888, 2017.

Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and S Yu Philip. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *International Conference on Machine Learning*, pp. 5123–5132. PMLR, 2018a.

Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *International conference on learning representations*, 2018b.

Yunbo Wang, Jianjin Zhang, Hongyu Zhu, Mingsheng Long, Jianmin Wang, and Philip S Yu. Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9154–9162, 2019.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1155–1162, 2013.

Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *Proceedings of the European Conference on Computer Vision*, 2020.

Table 6: Results on Moving Mnist dataset of video prediction problem

	Moving MNIST		
Method	**MSE**	**MAE**	**SSIM**
ConvLSTM Shi et al. (2015)	103.3	182.9	0.707
PredRNN Wang et al. (2017)	56.8	126.1	0.867
Causal LSTM Wang et al. (2018a)	46.5	106.8	0.898
MIM Wang et al. (2019)	44.2	101.1	0.91
E3D-LSTM Wang et al. (2018b)	41.3	86.4	0.92
DDPAE Hsieh et al. (2018)	38.9	90.7	0.922
Ours	33.1	83.3	0.93

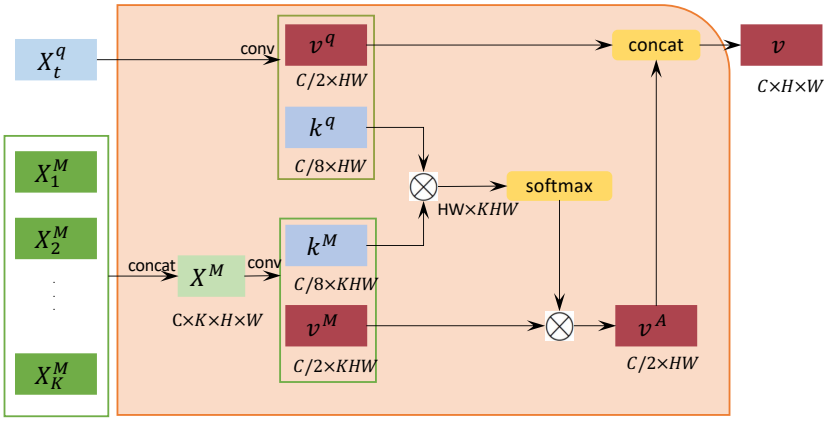

Figure 5: **Space-time memory module** - adapted from VOSSTM (Oh et al., 2019).

A Additional Experiment on Video Prediction

We also perform another experiment on the Moving Mnist dataset for the video prediction problem in Tab. 6. In this dataset, the model has to predict 10 frames into the future given 10 observations. In each video, there are 2 digits moving around the frame whose size is 32×32. We outperform other methods on all three MSE, MAE, and SSIM metrics.

B Details on Inference

Our inference module is adapted from STM to fit into our memory module. The details are described in Fig. 5.

Table 7: Results on Youtube VOS 2018 validation set. **R50/101**: use resnet 50/101 as backbone.

Methods	**Seen**		**Unseen**		
	J	**F**	**J**	**F**	**Avg**
GC(R50) (Li et al., 2020)	72.6	75.6	68.9	75.7	73.2
EGM(R50) (Lu et al., 2020)	80.7	85.1	74.0	80.9	80.2
CFBI(R101) (Yang et al., 2020)	81.1	**85.8**	**75.3**	83.4	**81.4**
PReMVOS(R101) (Luiten et al., 2018)	71.4	75.9	56.5	63.7	66.9
A-Game(R101) (Johnander et al., 2019)	67.8	-	60.8	-	66.1
KMN (R50) (Seong et al., 2020)	**81.4**	85.6	72.8	**84.2**	80.9
STM(R50) (Oh et al., 2019)	79.7	84.2	72.8	80.9	79.4
STReMN (Ours)	77.2	81.1	73.4	82.4	78.5

Table 8: The running time of each method

Methods	STM	EGM	Ours (4 slots)	Ours (5 slots)	Ours (6 slots)
FPS	2.3	0.93	3.3	3.0	2.6

C RESULTS ON YOUTUBE VOS

Youtube VOS is the largest dataset for video object segmentation task. However, the motions on this dataset are somewhat less complicated than DAVIS. We tested our model on the Youtube 2018 version. This dataset includes more than 4000 videos which includes 94 type of objects' categories such as human, animal, or vehicles. The validation set includes 474 videos including 65 seen categories and 26 unseen categories that cover 894 object instances. Unlike objects in DAVIS dataset, objects in Youtube can appear in the middle of the video. Because our model maintain separate memory for each object, this change does not impact significantly to the performance. Tab. 7 shows our result on Youtube 2018. Our model only fine-tuned on the DAVIS dataset significantly outperforms STM on YouTube. Our model fine-tuned on the YouTube dataset is competitive among memory-based methods. Especially, the model geneararlizes well to unseen categories better than EGM and STM. Among the models outperforming us, CFBI utilizes a stronger backbone, the recent KMN proposed a kernelized memory read and a pre-training trick which can be also added to our framework. KMN also has a linearly-growing memory, similar to STM.

D ABLATION ON TRAINING DATA - FINETUNING ON DAVIS 2017 AND YOUTUBE

Tab. 2 shows our results on the DAVIS 2017 validation and test-dev sets, when fine-tuned on both DAVIS and the YouTube datasets. Our model with a fixed size memory outperforms the constant-memory version of STM Oh et al. (2019) that only uses the first and the previous frame, and shows a comparable result with the linear-size-memory version of STM which adds new template to the memory for every 5 frames. KMN Seong et al. (2020) proposes to use a kernel approach, and utilizes Hide-and-Seek Seong et al. (2020); Singh & Lee (2017) augmentation which provides most of the improvement. CFBI Yang et al. (2020) uses Resnet 101 to extract backbone feature. Besides, CFBI also utilizes an ASPP module Chen et al. (2018) to extract multi-scale features. These augmentation and additional modules can be added to our architecture without changing our memory update module. EGM Lu et al. (2020), while maintaining a fixed-size graph, has to keep all previous frames' features to initialize the graph at each time step and thus increases the memory size linearly with the length of the video.

E RUNNING TIME

We measure the running time and the corresponding fps of each method on the video 7775043b5e on Youtube VOS which has 180 frames. All the methods are benchmarked using a single Quadro RTX8000 gpu with 16 cpu threads. Tab. 8 shows that our method beats other methods with linear growth memory when the input video is longer. Note 180 frames only span 7.5 seconds if we assume 24 frames a second. And our advantages will be much more significant with even longer videos that are more commonly seen in practice.

F ABLATION ON MEMORY SIZE

Tab. 9 and Fig. 6 shows the quantitative and qualitative results for different number of memory slots (MS). With $MS = 2$, our model is similar to STM with only the 1st and previous frames, and it also has the same performance. When $MS = 3$, our model has 1 slot to store the intermediate templates, and thus the overall IOU (J) and Boundary Overlapping (F) are improved. The improvement saturates after 6 slots.

The improvement looks small because most of the sequences were already segmented well and the performance are saturated. We observe significant improvements (IOU changes more than 10%) in

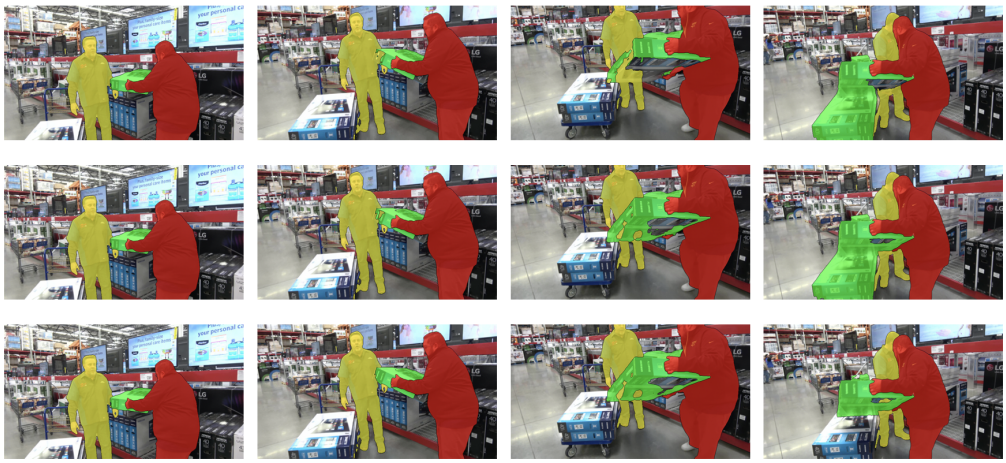

Figure 6: Comparison among models with different numbers of memory slots. First row: 2 memory slots. Second row: 3 slots; Third row: 6 slots (Best viewed in color)

some difficult sequences when increasing the total number of memory slots in Tab. 10. Notably, no sequence has decreased performance with more slots. Tab. 11 and 12 show the IOU for every sequence. The standard deviation column (σ) shows the change in IOU of each sequence when increasing the number of memory slots.

To inspect what really happens, we went through the *loading* video. In this video, the green box (sequence loading_2) rotates several times, displaying a multitude of appearances. By increase the number of memory slots, the model can remember more past appearances of the box and recover it better instead of drifting to the box on the background that has a similar appearance.

Table 9: Results on models with different memory size

	Validation		
	J	**F**	**Avg**
2 slots	77.7	83.2	80.5
3 slots	78.7	83.9	81.3
4 slots	79.0	84.1	81.5
5 slots	79.0	84.2	81.6
6 slots	**79.1**	**84.3**	**81.7**

G MORE QUALITATIVE RESULTS

We present other qualitative results in Fig. 7. Our model can track a wide range of object's categories. In the first video (first row), the model can maintain a wide range of appearances of the key chain. In the second video, the model shows that it can preserve a large amount of details of the

Table 10: Results on several sequences when using different numbers of memory slots. (We only choose sequences whose IOUs change more than 10%)

Video	**Sequence**	**2 slots**	**3 slots**	**4 slots**	**5 slots**	**6 slots**
pigs	pigs_1	0.705	0.933	0.936	0.936	**0.936**
	pigs_2	0.622	0.823	**0.832**	0.824	0.825
	pigs_3	0.963	0.963	0.963	0.963	**0.963**
loading	loading_1	0.965	0.965	0.965	0.965	**0.966**
	loading_2	0.483	0.6	0.739	0.749	**0.753**
	loading_3	0.818	0.845	0.848	0.853	**0.859**

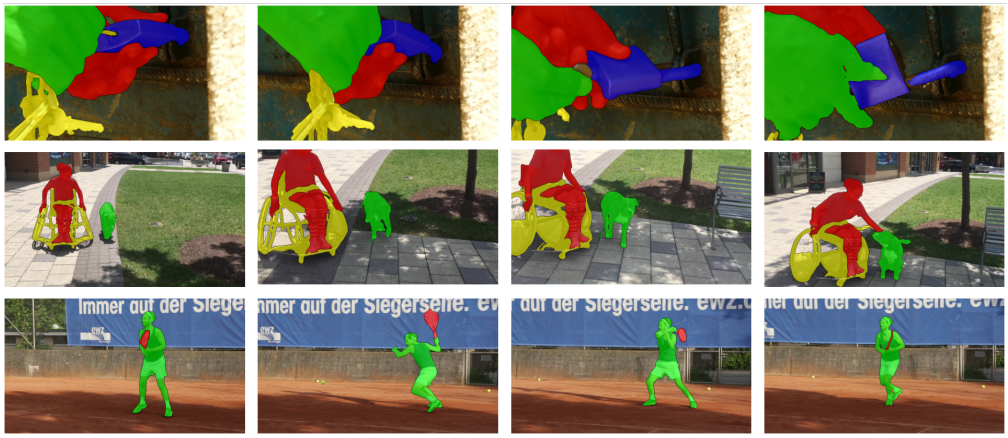

Figure 7: Qualitative results on DAVIS 2017 test-dev set. (Best viewed in color)

wheelchair despite the highly overlapped region. The third video shows that our model can quickly adapt the deformed appearance of the tennis racquet.

However, in some extreme cases as in Fig. 8 where the objects are too small or blurry, our model fails to capture the correct features for them. These cases suggest that we need to provide a dedicated module to handle multi-scale objects and a motion model to better segment the objects instead of only basing on the appearance model.

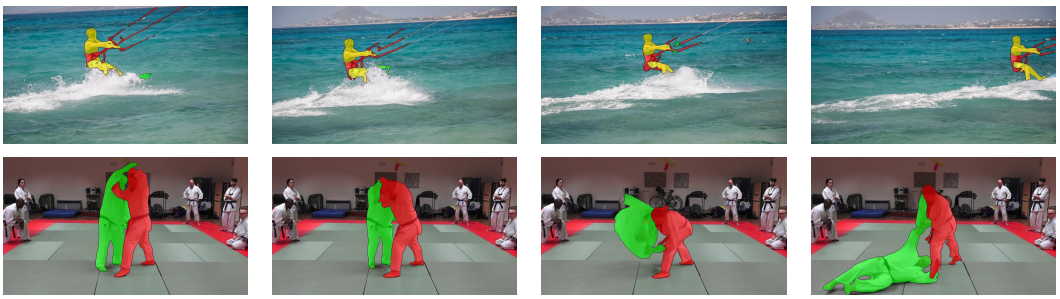

Figure 8: Failure cases. (Best viewed in color)

Table 11: The IOU of every sequence in DAVIS 2017 validation set. σ is the standard deviation of the IOU

Sequence	2 slots	3 slots	6 slots	σ
bike-packing_1	**0.727**	0.715	0.710	0.009
bike-packing_2	**0.847**	0.844	0.841	0.003
blackswan_1	0.962	**0.962**	0.961	0.001
bmx-trees_1	0.490	0.500	**0.510**	0.010
bmx-trees_2	0.742	0.743	**0.743**	0.001
breakdance_1	0.796	**0.802**	0.796	0.003
camel_1	0.850	**0.850**	0.848	0.001
car-roundabout_1	0.976	0.976	**0.982**	0.003
car-shadow_1	**0.968**	0.966	0.965	0.002
cows_1	0.961	0.961	**0.961**	0.000
dance-twirl_1	0.882	0.884	**0.887**	0.003
dog_1	0.961	0.961	**0.962**	0.001
dogs-jump_1	**0.859**	0.856	0.832	0.015
dogs-jump_2	0.934	0.934	**0.934**	0.000
dogs-jump_3	0.936	0.937	**0.938**	0.001
drift-chicane_1	0.722	0.766	**0.798**	0.038
drift-straight_1	0.936	0.939	**0.941**	0.003
goat_1	0.906	0.907	**0.908**	0.001
gold-fish_1	0.873	**0.875**	0.874	0.001
gold-fish_2	0.846	0.867	**0.877**	0.016
gold-fish_3	0.899	0.894	**0.894**	0.003
gold-fish_4	**0.918**	0.908	0.899	0.010
gold-fish_5	0.924	0.921	**0.930**	0.005
horsejump-high_1	**0.878**	0.877	0.876	0.001
horsejump-high_2	0.811	0.811	**0.815**	0.002
india_1	**0.915**	0.904	0.893	0.011
india_2	0.572	0.568	**0.607**	0.021
india_3	0.783	0.800	**0.819**	0.018
judo_1	0.850	0.851	**0.853**	0.002
judo_2	0.821	**0.823**	0.822	0.001

Table 12: The IOU of every sequence in DAVIS 2017 validation set. σ is the standard deviation of the IOU

Sequence	2 slots	3 slots	6 slots	σ
kite-surf_1	**0.431**	0.422	0.419	0.006
kite-surf_2	**0.468**	0.462	0.452	0.008
kite-surf_3	**0.76**7	0.765	0.751	0.009
lab-coat_1	0.000	0.000	0.000	0.000
lab-coat_2	0.000	0.000	0.000	0.000
lab-coat_3	0.913	0.923	**0.937**	0.012
lab-coat_4	0.924	0.921	**0.922**	0.002
lab-coat_5	0.865	0.879	**0.892**	0.014
libby_1	0.897	0.902	**0.905**	0.004
loading_1	0.965	0.965	**0.966**	0.001
loading_2	0.483	0.600	**0.753**	0.135
loading_3	0.818	0.845	**0.859**	0.021
mbike-trick_1	**0.839**	0.836	0.836	0.002
mbike-trick_2	**0.694**	0.684	0.683	0.006
motocross-jump_1	0.881	**0.882**	0.874	0.004
motocross-jump_2	0.880	**0.885**	0.884	0.003
paragliding-launch_1	0.796	0.817	**0.825**	0.015
paragliding-launch_2	0.678	0.712	**0.722**	0.023
paragliding-launch_3	**0.106**	0.094	0.094	0.007
parkour_1	0.949	0.950	**0.950**	0.001
pigs_1	0.705	0.933	**0.936**	0.133
pigs_2	0.622	0.823	**0.825**	0.117
pigs_3	0.963	0.963	**0.963**	0.000
scooter-black_1	**0.757**	0.739	0.740	0.010
scooter-black_2	**0.784**	0.777	0.730	0.029
shooting_1	0.638	0.641	**0.645**	0.004
shooting_2	0.869	0.862	**0.887**	0.013
shooting_3	0.892	**0.895**	0.893	0.002
soapbox_1	**0.824**	0.820	0.820	0.002
soapbox_2	**0.743**	0.727	0.728	0.009
soapbox_3	**0.704**	0.702	0.701	0.002