Video Object Detection with an Aligned Spatial-Temporal Memory

Fanyi Xiao(✉) and Yong Jae Lee(✉)

University of California, Davis, USA
{fyxiao,yongjaelee}@ucdavis.edu

Abstract. We introduce Spatial-Temporal Memory Networks for video object detection. At its core, a novel Spatial-Temporal Memory module (STMM) serves as the recurrent computation unit to model long-term temporal appearance and motion dynamics. The STMM's design enables full integration of pretrained backbone CNN weights, which we find to be critical for accurate detection. Furthermore, in order to tackle object motion in videos, we propose a novel MatchTrans module to align the spatial-temporal memory from frame to frame. Our method produces state-of-the-art results on the benchmark ImageNet VID dataset, and our ablative studies clearly demonstrate the contribution of our different design choices. We release our code and models at http://fanyix.cs.ucdavis.edu/project/stmn/project.html.

Keywords: Aligned spatial-temporal memory · Video object detection

1 Introduction

Object detection is a fundamental problem in computer vision. While there has been a long history of detecting objects in *static images*, there has been much less research in detecting objects in *videos*. However, cameras on robots, surveillance systems, vehicles, wearable devices, etc., receive videos instead of static images. Thus, for these systems to recognize the key objects and their interactions, it is critical that they be equipped with accurate *video* object detectors.

The simplest way to detect objects in video is to run a static image-based detector independently on each frame. However, due to the different biases and challenges of video (e.g., motion blur, low-resolution, compression artifacts), an image detector usually does not generalize well. More importantly, videos provide *rich temporal and motion information* that should be utilized by the detector during both training and testing. For example, in Fig. 1, since the hamster's profile view (frames 1–2) is much easier to detect than the challenging viewpoint/pose in later frames, the image detector only succeeds in detecting the leading frame of the sequence. On the other hand, by learning to aggregate useful information over time, a video object detector can robustly detect the object under extreme viewpoint/pose.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11212, pp. 494–510, 2018.
https://doi.org/10.1007/978-3-030-01237-3_30

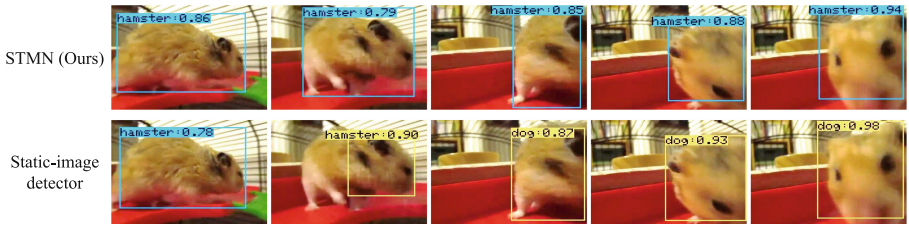

Fig. 1. Static image detectors (such as Fast-RCNN [17] or R-FCN [9]), tend to fail under occlusion or extreme pose (false detections shown in yellow). By learning to aggregate information across time, our STMN video object detector can produce correct detections in frames with challenging pose/viewpoints. In this example, it aggregates information from the easier profile views of the hamster (first two frames) to aid detection in occluded or extreme views of the hamster (third-fifth frames). (Color figure online)

Therefore, in recent years, there has been a growing interest in the community on designing video object detectors [14,18,19,25,29,50,51]. However, many existing methods exploit temporal information in an ad-hoc, post-processing manner – static object detections returned by an image detector like R-FCN [9] or Faster R-CNN [38] are linked across frames [19,25,29], or video segmentation is performed to refine the detection results [18]. Although these methods show improvement over a static image detector, exploiting temporal information as post-processing is sub-optimal since temporal and motion information are ignored during detector training. As such, they have difficulty overcoming consecutive failures of the static detector e.g., when the object-of-interest has large occlusion or unusual appearance for a long time.

More recent works [14,50,51] learn to exploit temporal information *during training* by either learning to combine features across neighboring frames or by predicting the displacement of detection boxes across frames. However, these methods operate on fixed-length temporal windows and thus have difficulty modeling *variable and long-term* temporal information. While the Tubelet Proposal Network [24] does model long-term dependencies, it uses *vectors* to represent the memory of the recurrent unit, and hence loses spatial information. To compensate, it computes the memory vectors at the region-level for each tube (sequence of proposals), but this can be very slow and depends strongly on having accurate initial tubes.

To address these limitations, we introduce the *Spatial-Temporal Memory Network* (STMN), which jointly learns to model and align an object's long-term appearance and motion dynamics in an end-to-end fashion for video object detection. At its core is the Spatial-Temporal Memory Module (STMM), which is a convolutional recurrent computation unit that fully integrates pre-trained weights learned from static images (e.g., ImageNet [11]). This design choice is critical in addressing the practical challenge of learning from contemporary video datasets, which largely lack intra-category object diversity; i.e., since video

frames are highly redundant, a video dataset of e.g., 1 million frames has much lower diversity than an image dataset with 1 million images. By designing our memory unit to be compatible with pre-trained weights from both its preceding and succeeding layers, we show that it outperforms the standard ConvGRU [4] recurrent module for video object detection.

Furthermore, in order to account for the 2D spatial nature of visual data, the STMM preserves the spatial information of each frame in its memory. In particular, to achieve accurate pixel-level spatial alignment over time, the STMM uses a novel MatchTrans module to explicitly model the displacement introduced by motion across frames. Since the convolutional features for each frame are aligned and aggregated in the spatial memory, the feature for any particular object region is well-localized and contains information across multiple frames. Furthermore, each region feature can be extracted trivially via ROI pooling from the memory.

In summary, our main contribution is a novel spatial-temporal memory network for video object detection. Our ablative studies show the benefits provided by the STMM and MatchTrans modules – integrating pre-trained static image weights and providing spatial alignment across time. These design choices lead to state-of-the-art results on the ImageNet video object detection dataset (VID) [1] across different base detectors and backbone networks.

2 Related Work

Static image object detection. Recent work that adopt deep neural networks have significantly advanced the state-of-the-art in static image object detection [7,9, 16,17,31,37–40]. Our work also builds on the success of deep networks to learn the features, classifier, and bounding box localizer in an end-to-end framework. However, in contrast to most existing work that focus on detecting objects in static images, this paper aims to detect objects in *videos*.

Video object detection. Compared to static image-based object detection, there has been less research in detecting objects in videos. Early work processed videos captured from a static camera or made strong assumptions about the type of scene (e.g., highway traffic camera for detecting cars or an indoor room for detecting persons) [8,46]. Later work used hand-designed features by aggregating simple motion cues (based on optical flow, temporal differences, or tracking), and focused mostly on pedestrian detection [10,23,35,45].

With the introduction of ImageNet VID [1] in 2015, researchers have focused on more generic categories and realistic videos. However, many existing approaches combine per-frame detections from a static image detector via tracking in a two-stage pipeline [19,25,43]. Since the motion and temporal cues are used as a post-processing step only during testing, many heuristic choices are required, which can lead to sub-optimal results. In contrast, our approach directly *learns* to integrate the motion and temporal dependencies during training. Our end-to-end architecture also leads to a clean and fast runtime.

Sharing our goal of leveraging temporal information *during training*, the recent works of Zhu et al. [50,51] learn to combine features of different frames with a feed-forward network for improved detection accuracy. Our method differs in that it produces a *spatial-temporal memory* that can carry on information across long and variable number of frames, whereas the methods in [50,51] can only aggregate information over a small and fixed number of frames. In Sect. 4.3, we demonstrate the benefits gained from this flexibility. Although the approach of Kang et al. [24] uses memory to aggregate temporal information, it uses a vector representation. Since spatial information is lost, it computes a separate memory vector for each region tube (sequence of proposals) which can make the approach very slow. In contrast, our approach only needs to compute a single *frame-level* spatial memory, whose computation is independent of the number of proposals.

Finally, Detect and Track [14] aims to unify detection and tracking, where the correlation between consecutive *two* frames are used to predict the movement of the detection boxes. Unlike [14], our spatial-temporal memory aggregates information across $t > 2$ frames. Furthermore, while our approach also computes the correlation between neighboring frames with the proposed MatchTrans module, we use it to warp the entire feature map for alignment (i.e., at the coarse pixel-level), rather than use it to predict the displacement of the boxes. Overall, these choices lead to state-of-the-art detection accuracy on ImageNet VID.

Learning with videos. Apart from video object detection, other recent work use convolutional and/or recurrent networks for video classification [4,26,42]. These methods tend to model entire video frames instead of pixels, which means the fine-grained details required for localizing objects are often lost. Object tracking (e.g., [30,33]), which requires accurate localization, is also closely-related. The key difference is that in tracking, the bounding box of the first frame is given and the tracker does not necessarily need to know the semantic category of the object being tracked.

Modeling sequence data with RNNs. In computer vision, RNNs have been used for image captioning [12,27,44], visual attention [2,32,47], action/object recognition [4,12], human pose estimation [6,15], and semantic segmentation [49]. Recently, Tripathi et al. [43] adopted RNNs for video object detection. However, in their pipeline, the CNN-based detector is first trained, then an RNN is trained to refine the detection outputs of the CNN.

Despite the wide adoption of RNNs in various vision tasks, most approaches work with vector-form memory units (as in standard LSTM/GRU). To take spatial locality into account, Ballas et al. [4] proposed convolutional gated recurrent units (ConvGRU) and applied it to the task of action recognition. Built upon [4], Tokmakov et al. [41] used ConvGRUs for the task of video object segmentation. Our work differs in three ways: (1) we classify bounding boxes rather than frames or pixels; (2) we propose a new recurrent computation unit called STMM that makes better use of static image detector weights pre-trained on a large-scale image dataset like ImageNet; and (3) our spatial-temporal memory is aligned

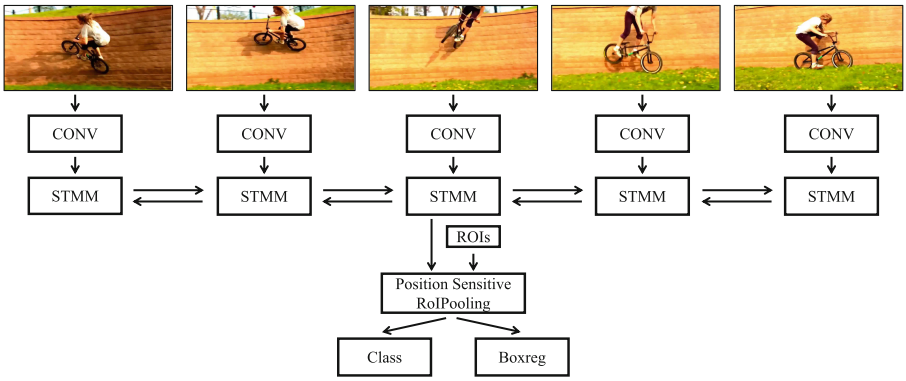

Fig. 2. Our STMN architecture. Consecutive frames are forwarded through the convolutional stacks to obtain spatial-preserving convolutional feature maps, which are then fed into the spatial-temporal memory module (STMM). In this example, in order to detect an object on the center frame, information flows into the center STMM from all five frames. The STMM output from the center frame is then fed into a classification and box regression sub-network.

frame-to-frame through our MatchTrans module. We show that these properties lead to better results than ConvGRU for video object detection.

3 Approach

We propose a novel RNN architecture called the Spatial-Temporal Memory Network (STMN) to model an object's changing appearance and motion over time for video object detection.

3.1 Overview

The overall architecture is shown in Fig. 2. Assuming a video sequence of length T, each frame is first forwarded through a convnet to obtain convolutional feature maps $F_1, F_2, ..., F_T$ as appearance features. To aggregate information along the temporal axis, the appearance feature of each frame is fed into the Spatial-Temporal Memory Module (STMM). The STMM at time step t receives the appearance feature for the current frame F_t, as well as a spatial-temporal memory $M^{\rightarrow}_{t-1}$, which carries the information of all previous frames up through timestep $t-1$. The STMM then updates the spatial-temporal memory for the current time step $M^{\rightarrow}_t$ conditioned on both F_t and $M^{\rightarrow}_{t-1}$. In order to capture information from both previous and later frames, we use two STMMs, one for each direction, to obtain both $M^{\rightarrow}$ and $M^{\leftarrow}$. These are then concatenated to produce the temporally modulated memory M for each frame.

The concatenated memory M, which also preserves spatial information, is then fed into subsequent convolution/fully-connected layers for both category

classification and bounding box regression. This way, our approach combines information from both the current frame as well as temporally-neighboring frames when making its detections. This helps, for instance, in the case of detecting a frontal-view bicycle in the center frame of Fig. 2 (which is hard), if we have seen its side-view (which is easier) from nearby frames. In contrast, a static image detector would only see the frontal-view bicycle when making its detection.

Finally, to train the detector, we use the same loss function used in R-FCN [9]. Specifically, for each frame in a training sequence, we enforce a cross-entropy loss between the predicted class label and the ground-truth label, and enforce a smooth $L1$ loss on the predicted bounding box regression coefficients. During testing, we slide the testing window and detect on all frames within each sliding window, to be consistent with our training procedure.

3.2 Spatial-Temporal Memory Module

We next explain how the STMM models the temporal correlation of an object across frames. At each time step, the STMM takes as input F_t and M_{t-1} and computes the following:

$$z_t = \texttt{BN}^*(\texttt{ReLU}(W_z * F_t + U_z * M_{t-1})), \tag{1}$$

$$r_t = \texttt{BN}^*(\texttt{ReLU}(W_r * F_t + U_r * M_{t-1})), \tag{2}$$

$$\tilde{M}_t = \texttt{ReLU}(W * F_t + U * (M_{t-1} \odot r_t)), \tag{3}$$

$$M_t = (1 - z_t) \odot M_{t-1} + z_t \odot \tilde{M}_t, \tag{4}$$

where $\odot$ is element-wise multiplication, $*$ is convolution, and U, W, U_r, W_r, U_z, W_z are the 2D convolutional kernels, whose parameters are optimized end-to-end. Gate r_t masks elements of M_{t-1} (i.e., it allows the previous state to be forgotten) to generate candidate memory $\tilde{M}_t$. And gate z_t determines how to weight and combine the memory from the previous step M_{t-1} with the candidate memory $\tilde{M}_t$, to generate the new memory M_t.

To generate r_t and z_t, the STMM first computes an affine transformation of M_{t-1} and F_t, and then ReLU [28] is applied to the outputs. Since r_t and z_t are gates, their values need to be in the range of $[0, 1]$. Therefore, we make two changes to the standard BatchNorm [22] (and denote it as $\texttt{BN}^*$) such that it normalizes its input to $[0, 1]$, instead of zero mean and unit standard deviation.

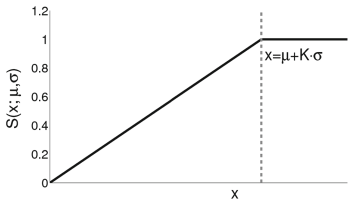

Fig. 3. $S(x; \mu, \sigma)$ squashes any value in $[0, +\text{inf})$ into range $[0, 1]$, with a linear scaling function thresholded at $\mu + K \cdot \sigma$. We set $K = 3$.

First, our variant of BatchNorm computes the mean $\mu(X)$ and standard deviation $\sigma(X)$ for an input batch X, and then normalizes values in X with the linear squashing function $S(X; \mu, \sigma)$ shown in Fig. 3. Second, we compute the mean and standard deviation for each batch independently instead of keeping running averages across training batches.

In this way, we do not need to store different statistics for different time-steps, which allows us to generate test results for sequence lengths not seen during training (e.g., we can compute detections on longer sequences than those seen during training as demonstrated in Sect. 4.3). Note that a key difference between BN* and instance/layer normalization [3,21] is that BN* guarantees that *each and every* value in its output is normalized within $[0, 1]$ (which is necessary for gating variables), whereas neither instance nor layer normalization ensures this property. Although simple, we find BN* works well for our purpose.

Differences with ConvGRU [4]**.** A key practical challenge of learning video object detectors is the lack of intra-category object diversity in contemporary video datasets; i.e., since video frames are highly redundant, a video dataset of e.g., 1 million frames has much lower diversity than an image dataset with 1 million images. The cost of annotation is much higher in video, which makes it difficult to have the same level of diversity as an image dataset. Therefore, transferring useful information from large-scale *image* datasets like ImageNet [11]—into the memory processing unit itself—would benefit our video object detector by providing additional diversity.

Specifically, we would like to initialize our STMN detector with the weights of the state-of-the-art static image-based RFCN detector [9] which has been pretrained on ImageNet DET images, and continue to fine-tune it on ImageNet VID videos. In practice, this would entail converting the last convolution layer before the Position-Sensitive ROI pooling in RFCN into our STMM memory unit (see Fig. 2). However, this conversion is non-trivial with standard recurrent units like LSTM/GRU that employ Sigmoid/Tanh nonlinearities, since they are different from the ReLU nonlinearity employed in the R-FCN convolutional layers.

Thus, to transfer the weights of the pre-trained RFCN static image detector into our STMN video object detector, we make two changes to the ConvGRU [4]. First, in order to make full use of the pre-trained weights, we need to make sure the output of the recurrent computation unit is compatible with the pre-trained weights before and after it. As an illustrative example, since the output of the standard ConvGRU is in $[-1, 1]$ (due to Tanh non-linearity), there would be a mismatch with the input range that is expected by the ensuing pre-trained convolutional layer (the expected values should all be positive due to ReLU). To solve this incompatibility, we change the non-linearities in standard ConvGRU from Sigmoid and Tanh to ReLU. Second, we initialize W_z, W_r and W in Eqs. 1–3 with the weights of the convolution layer that is swapped out, rather than initializing them with random weights. Conceptually, this can be thought of as a way to initialize the memory with the pre-trained static convolutional feature maps. In Sect. 4.3, we show that these modifications allow us to make better use of pre-trained weights and achieve better detection performance.

3.3 Spatial-Temporal Memory Alignment

Next, we explain how to align the memory across frames. Since objects *move* in videos, their spatial features can be mis-aligned across frames. For example,

the position of a bicycle in frame $t-1$ might not be aligned to the position of the bicycle in frame t (as in Fig. 2). In our case, this means that the spatial-temporal memory M_{t-1} may not be spatially aligned to the feature map for current frame F_t. This can be problematic, for example in the case of Fig. 4; without proper alignment, the spatial-temporal memory can have a hard time forgetting an object after it has moved to a different spatial position. This is manifested by a trail of saliency, in the fourth row of Fig. 4, due to the effect of overlaying multiple unaligned feature maps. Such hallucinated features can lead to false positive detections and inaccurate localizations, as shown in the third row of Fig. 4.

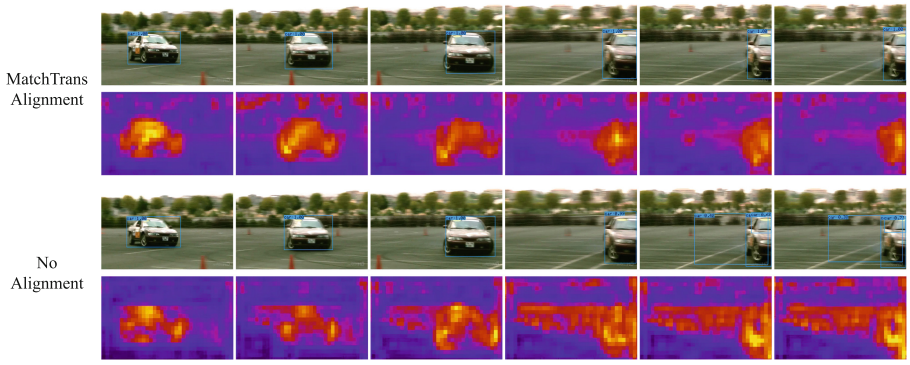

Fig. 4. Effect of alignment on spatial-temporal memory. In the first and second rows, we show the detection and the visualization of the spatial-temporal memory (by computing the $L2$ norm across feature channels at each spatial location to get a saliency map), respectively, with MatchTrans alignment. The detection and memory without alignment are shown in rows 3 and 4, respectively. Without proper alignment, the memory has a hard time forgetting an object after it has moved to a different spatial position (third row), which is manifested by a trail of saliency on the memory map due to overlaying multiple unaligned maps (fourth row). Alignment with MatchTrans helps generate a much cleaner memory (second row), which also results in better detections (first row). Best viewed in pdf. (Color figure online)

To alleviate this problem, we propose the MatchTrans module to align the spatial-temporal memory across frames. For a feature cell $F_t(x,y) \in 1 \times 1 \times D$ at location (x,y) in F_t, MatchTrans computes the affinity between $F_t(x,y)$ and feature cells in a small vicinity around location (x,y) in F_{t-1}, in order to transform the spatial-temporal memory M_{t-1} to align with frame t. More formally, the transformation coefficients Γ are computed as:

$$\Gamma_{x,y}(i,j) = \frac{F_t(x,y) \cdot F_{t-1}(x+i, y+j)}{\sum_{i,j \in \{-k,\dots,k\}} F_t(x,y) \cdot F_{t-1}(x+i, y+j)},$$

where both i and j are in the range of $[-k,k]$, which controls the size of the matching vicinity. With Γ, we transform the unaligned memory M_{t-1} to the aligned M'_{t-1} as follows:

$$M'_{t-1}(x,y) = \sum_{i,j \in \{-k,\ldots,k\}} \Gamma_{x,y}(i,j) \cdot M_{t-1}(x+i, y+j).$$

The intuition here is that given transformation Γ, we reconstruct the spatial memory $M'_{t-1}(x,y)$ as a weighted average of the spatial memory cells that are within the $(2k+1) \times (2k+1)$ vicinity around (x,y) on M_{t-1}; see Fig. 5. At this point, we can thus simply replace all occurrences of M_{t-1} with the spatially aligned memory M'_{t-1} in Eqs. 1–4. With proper alignment, our generated memory is much cleaner (second row of Fig. 4) and leads to more accurate detections (first row of Fig. 4). Since the computational cost is quadratic in k, we set $k = 2$ for all our experiments as this choice provides a good trade-off between performance and computation.

Our MatchTrans is related to the alignment module used in recent video object detection work by [50,51]. However, [50,51] use optical flow, which needs to be computed either externally e.g., using [5], or in-network through another large CNN e.g., FlowNet [13]. In contrast, our MatchTrans is much more efficient, saving computation time and/or space for storing optical flow. For example, it is nearly an order of magnitude faster to compute (on average, 2.9 ms vs. 24.3 ms for an 337×600 frame) than FlowNet [13], which is one of the fastest optical flow methods. Also, a similar procedure for computing transformation coefficients was used in [14]. However, in [14], the coefficients are fed as input to another network to regress the displacement of bounding boxes for tracking, whereas we use it to warp the entire feature map for aligning the memory. In other words, rather than use the transformation coefficients to track and connect detections, we instead use them to align the memory over time to produce better features for each candidate object region. We show in Sect. 4.1 that this leads to better performance on ImageNet VID.

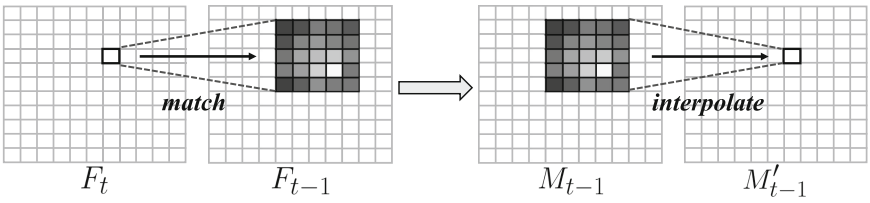

Fig. 5. The transformation coefficients Γ for position (x,y) are computed by matching $F_t(x,y)$ to $F_{t-1}(i,j)$, where i,j indexes a spatial neighborhood surrounding (x,y). The transformation coefficients are then used to synthesize $M'_{t-1}(x,y)$ by interpolating the corresponding $M_{t-1}(i,j)$ feature vectors.

3.4 Temporal Linkage During Testing

Finally, even though we enforce temporal smoothness in our spatial-temporal memory (i.e., at the feature level), we do not have an explicit smoothness constraint in the output space to ensure that detections in adjacent frames are spatially smooth. We therefore apply standard Seq-NMS [19] over our per-frame detections, following [14,51].

3.5 Approach Summary

Through the specially designed Spatial-Temporal Memory and MatchTrans modules, our STMN detector aggregates and aligns useful information from temporally nearby frames for video object detection.

4 Results

We show quantitative and qualitative results of our STMN video object detector, and compare to both state-of-the-art static image and video detectors. We also conduct ablation studies to analyze the different components of our model.

Dataset. We use ImageNet VID [1], which has 3862/555/937 videos for training/validation/testing for 30 categories. Bounding box annotation is provided for all frames. We choose ImageNet VID for its relatively large size as well as for ease of comparison to existing state-of-the-art methods [1,9,14,24,25,50,51].

Implementation details. For object proposals, we use DeepMask [36] for our method and our own baselines. We use the R-FCN detector [9] with ResNet-101 [20] as the backbone network. Following [14], we first train R-FCN on ImageNet DET, and then transfer its weights (using the method described in Sect. 3.2) to initialize our STMN detector and continue fine-tuning it on ImageNet VID. We set sequence length $T = 7$ during training. For testing, we observe better performance when using a longer sequence length; specifically, $T = 11$ frames provides a good balance between performance and GPU memory/computation (we later show the relationship between performance and test sequence length). We set the number of channels of the spatial memory to 512. To reduce redundancy within sequences, we form a sequence by sampling 1 in every 10 video frames with uniform stride. For training, we start with a learning rate of 1e-3 with SGD and lower it to 1e-4 when training loss plateaus. During testing we ensemble the detection results of the STMN detector with the initial R-FCN detector from which it started since it comes for free as a byproduct of the training procedure. We employ standard left-right flipping augmentation.

4.1 Comparison to State-of-the-Art

Table 1 shows the comparison to existing state-of-the-art image and video detectors. First, our STMN detector outperforms the static-image based R-FCN detector with a large margin (+7.1%). This demonstrates the effectiveness of our proposed spatial-temporal memory. Our STMN detector also achieves the best performance compared to all existing video object detection methods with ResNet-101 as the base network. Furthermore, in order to enable a fairer comparison to older methods that use Fast/Faster-RCNN + VGG-16 as the base detector and backbone network, we also train an STMN model with the Fast-RCNN as the base detector and VGG-16 as the backbone feature network. Specifically,

Table 1. mAP comparison to the state-of-the-art on ImageNet VID. For both the "R-FCN+ResNet-101" and the "Fast-RCNN+VGG-16" settings, our STMN detector outperforms all existing methods with the same base detector and backbone network. Furthermore, in both cases, our STMN outperforms the corresponding static-image detector by a large margin.

	Base network	Base detector	Test	Val
STMN (Ours)	ResNet-101	R-FCN	-	**80.5**
D&T [14]	ResNet-101	R-FCN	-	79.8
Zhu et al. [50]	ResNet-101+DCN	R-FCN	-	78.6
FGFA [51]	ResNet-101	R-FCN	-	78.4
T-CNN [25]	DeepID+Craft [34,48]	RCNN	67.8	73.8
R-FCN [9]	ResNet-101	R-FCN	-	73.4
TPN [24]	GoogLeNet	TPN	-	68.4
STMN (Ours)	VGG-16	Fast-RCNN	**56.5**	**61.7**
Faster-RCNN [1,19]	VGG-16	Faster-RCNN	48.2	52.2
ITLab VID - Inha [1]	VGG-16	Fast-RCNN	51.5	-

we first train a static-image Fast-RCNN detector and initialize the weights of STMN using a similar procedure as described in Sect. 3.2.[1] With this setting, our STMN achieves 61.7% val mAP, which is much higher than its static-image based counterpart (52.2%). This result shows that our method can be generalized across different base detectors and backbone networks.

When examining per-category results, our method shows the largest improvement on categories like "sheep","rabbit", and "domestic cat" compared to methods like [14]. In these cases, we see a clear advantage of aggregating information across multiple frames (vs. 2 frames as in [14]), as there can be consecutive"hard" frames spanning multiple (>2) frames (e.g., a cat turning away from the camera for several frames). On the other hand, we find that the three categories on which we perform the worst are "monkey","snake", and "squirrel". These are categories with large deformation and strong motion blur. When the per-frame appearance features fail to accurately model these objects due to such challenges, aggregating those features over time with our STMM does not help. Still, overall, we find that our model produces robust detection results across a wide range of challenges as demonstrated next in the qualitative results.

4.2 Qualitative Results

Figure 6 shows qualitative comparisons between our STMN detections and the static image R-FCN detections. Our STMN detections are more robust to motion blur; e.g., in the last frame of the "hamster" sequence, R-FCN gets confused

[1] Specifically, we convert the `conv5` layer in VGG-16 to an STMM module by initializing W_z, W_r and W in Eqs. 1−3 with the weights of `conv5`.

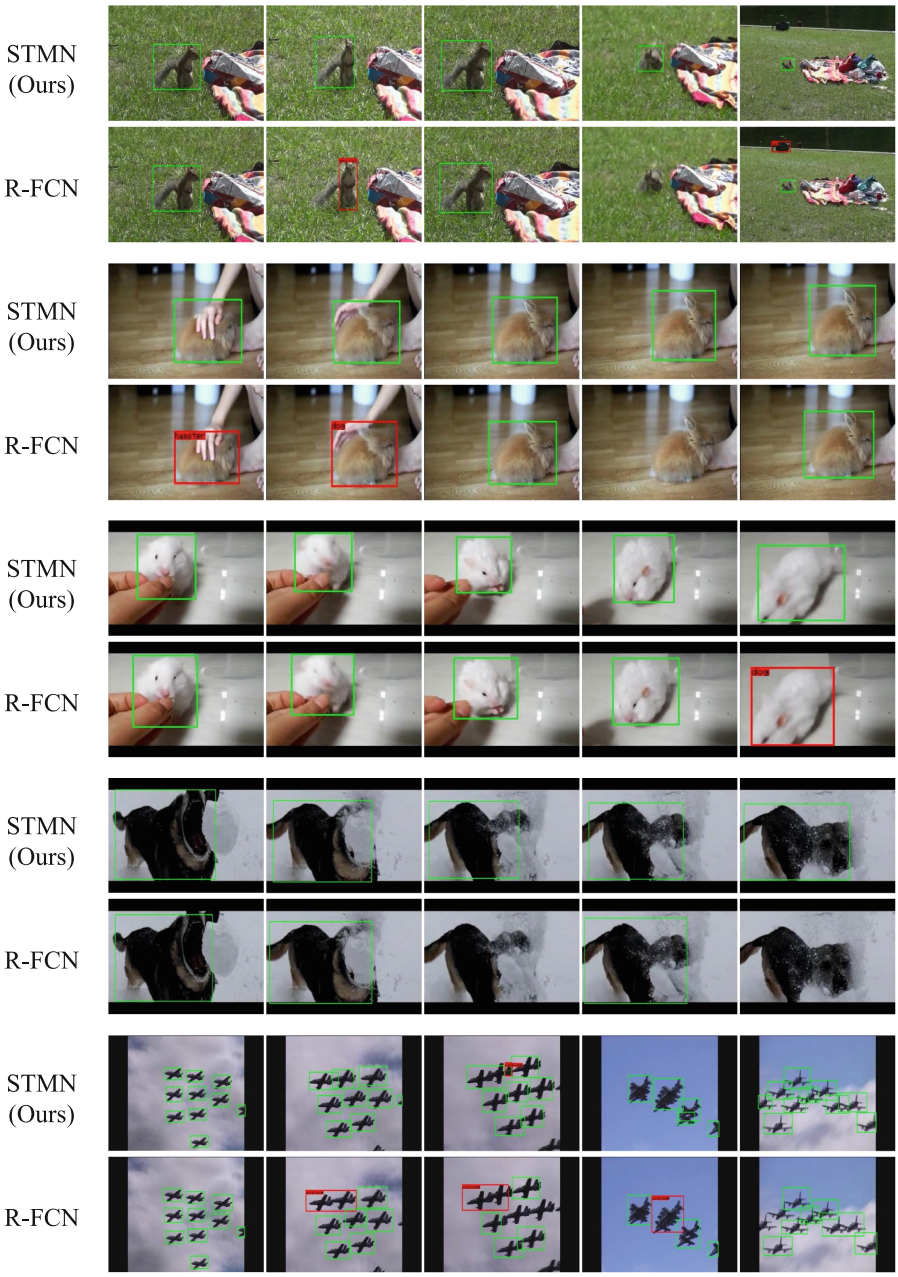

Fig. 6. Example detections produced by our STMN video object detector vs. R-FCN image detector. Green and red boxes indicate correct and incorrect detections, respectively. For any false positive detection due to misclassification or mislocalization, the predicted category label is shown at the top-left corner of the box. The ground-truth object in each sequence is: "squirrel","rabbit", "hamster","dog," and "airplane". Best viewed in pdf, zoomed-in. (Color figure online)

about the class label of the object due to large motion blur, whereas our STMN detector correctly detects the object. In the case of difficult viewpoint and occlusion ("dog" and "rabbit", respectively), our STMN produces robust detections by leveraging the information from neighboring easier frames (i.e., center frame in the"rabbit" sequence and the first frame in the "dog" sequence). Also, our model outputs detections that are more consistent across frames, compared with the static image detector, as can be seen in the case of "squirrel" and "rabbit". Finally, our STMN detector is also more robust in crowded scenes as shown in the "airplane" sequence.

4.3 Ablation Studies

We next conduct ablation studies to analyze the impact of each component in our model by comparing it to a number of baselines that lack one or more components. For this, we use Fast-RCNN as the base detector and VGG-16 as the backbone network since it is much faster to train compared to RFCN + ResNet-101. To ensure a clean analysis, we purposely do not employ any data augmentation during training for this ablative study.

Contribution of STMN Components. The first baseline, compared with our model, lacks the MatchTrans module and thus does not align the memory from frame to frame (STMN-No-MatchTrans). The second baseline computes the memory using ConvGRU [4], instead of our proposed STMM. Like ours, this baseline (ConvGRU-Pretrain) also uses pre-trained ImageNet weights for both the feature stack and prediction layers. Our final baseline is ConvGRU without pre-trained weights for the ensuing prediction FCs (ConvGRU-FreshFC).

Table 2. Ablation studies on ImageNet VID. Our improvements over the baselines show the importance of memory alignment across frames with MatchTrans (vs. STMN-No-MatchTrans), and the effectiveness of using pre-trained weights with STMM over standard ConvGRU (vs. ConvGRU-Pretrain and ConvGRU-FreshFC).

	STMN	STMN No-MatchTrans	ConvGRU Pretrain	ConvGRU FreshFC
Test mAP	50.7	49.0	48.0	44.8

Table 2 shows the results. First, comparing our STMN to the STMN-No-MatchTrans baseline, we observe a 1.7% test mAP improvement brought by the spatial alignment across frames. This result shows the value of our MatchTrans module. To compare our STMM with ConvGRU, we first replace STMM with ConvGRU and as with standard practice, randomly initialize the weights for the FC layers after the ConvGRU. With this setting (ConvGRU-FreshFC), we obtain a relatively low test mAP of 44.8%, due to the lack of data to train the large amount of weights in the FCs. This result shows that initializing the

memory by only partially transferring the pre-trained ImageNet weights is sub-optimal. If we instead initialize the weights of the FCs after the ConvGRU with pre-trained weights (ConvGRU-Pretrain), we improve the test mAP from 44.8% to 48.0%. Finally, by replacing `Sigmoid` and `Tanh` with `ReLU`, which is our full model (STMN), we boost the performance even further to 50.7%. This shows the importance of utilizing pre-trained weights in both the feature stacks and prediction head, and the necessity of an appropriate form of recurrent computation that best matches its output to the input expected by the pre-trained weights.

Length of Test Window Size. We next analyze the relationship between detection performance and length of test window size. Specifically, we test our model's performance with test window size 3, 7, 11, and 15, on ImageNet VID validation set (the training window size is always 7). The corresponding mAP differences, with respect to that of window size 7, are -1.9%, 0.0%, $+0.7\%$, $+1.0\%$, respectively; as we increase the window size, the performance tends to keep increasing. This suggests the effectiveness of our memory: the longer the sequence, the more longer-range useful information is stored in the memory, which leads to better detection performance. However, increasing the test window size also increases computation cost and GPU memory consumption. Therefore, we find that setting the test window size to 11 provides a good balance.

4.4 Computational Overhead of STMN

Finally, we sketch the computational overhead of our memory module. To forward a batch of 11 frames of size 337×600, it takes 0.52 and 0.83 s for R-FCN and STMN respectively, on a Titan X GPU. The added 0.028 (=0.31/11) secs/frame is spent on STMM computation including MatchTrans alignment.

5 Conclusion

We proposed a novel spatial-temporal memory network (STMN) for video object detection. Our main contributions are a carefully-designed recurrent computation unit that integrates pre-trained image classification weights into the memory and an in-network alignment module that spatially-aligns the memory across time. Together, these lead to state-of-the-art results on ImageNet VID. Finally, we believe that our STMN could also be useful for other video understanding tasks that require accurate spatial information like action detection and keypoint detection.

Acknowledgments. This work was supported in part by the ARO YIP W911NF17-1-0410, NSF CAREER IIS-1751206, AWS Cloud Credits for Research Program, and GPUs donated by NVIDIA. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARO or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

1. http://image-net.org/challenges/LSVRC/2015/results#vid
2. Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. arXiv preprint arXiv:1412.7755 (2014)
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
4. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. In: ICLR (2016)
5. Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. PAMI **33**(3), 500–513 (2011)
6. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: CVPR (2016)
7. Chen, X., Gupta, A.: Spatial memory for context reasoning in object detection. In: ICCV (2017)
8. Coifman, B., Beymer, D., Mclauchlan, P., Malik, J.: A realtime computer vision system for vehicle tracking and traffic surveillance. Transp. Res. C **6C**(4), 271–288 (1998)
9. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: NIPS (2016)
10. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 428–441. Springer, Heidelberg (2006). https://doi.org/10.1007/11744047_33
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
12. Donahue, J., et al.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR (2015)
13. Dosovitskiy, A., et al.: FlowNet: learning optical flow with convolutional networks. In: ICCV (2015)
14. Feichtenhofer, C., Pinz, A., Zisserman, A.: Detect to track and track to detect. In: ICCV (2017)
15. Fragkiadaki, K., Levine, S., Felsen, P., Malik, J.: Recurrent network models for human dynamics. In: ICCV (2015)
16. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
17. Girshick, R.: Fast R-CNN. In: ICCV (2015)
18. Han, P., Yuan, W., Lu, Z., Wen, J.R.: Video detection by learning with deep representation and spatio-temporal context (2015)
19. Han, W., et al.: Seq-NMS for video object detection. arXiv preprint arXiv:1602.08465 (2016)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
21. Huang, X., Belongie, S.J.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV (2017)
22. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
23. Jones, M., Snow, D.: Pedestrian detection using boosted features over many frames. In: ICPR (2008)

24. Kang, K., et al.: Object detection in videos with tubelet proposal networks. In: CVPR (2017)
25. Kang, K., et al.: T-CNN: tubelets with convolutional neural networks for object detection from videos. TCSVT (2017)
26. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
27. Kiros, R., Salakhutdinov, R., Zemel, R.S.: Unifying visual-semantic embeddings with multimodal neural language models. arXiv preprint arXiv:1411.2539 (2014)
28. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)
29. Lee, B., Erdenee, E., Jin, S., Nam, M.Y., Jung, Y.G., Rhee, P.K.: Multi-class multi-object tracking using changing point detection. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9914, pp. 68–83. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48881-3_6
30. Li, Y., Zhu, J., Hoi, S.C.: Reliable patch trackers: robust visual tracking by exploiting reliable patches. In: CVPR (2015)
31. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
32. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention. In: NIPS (2014)
33. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR (2016)
34. Ouyang, W., et al.: DeepID-Net: multi-stage and deformable deep convolutional neural networks for object detection. arXiv preprint arXiv:1409.3505 (2014)
35. Park, D., Zitnick, C.L., Ramanan, D., Dollar, P.: Exploring weak stabilization for motion feature extraction. In: CVPR (2013)
36. Pinheiro, P.O., Collobert, R., Dollar, P.: Learning to segment object candidates. In: NIPS (2015)
37. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
38. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
39. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: OverFeat: integrated recognition, localization and detection using convolutional networks. In: ICLR (2014)
40. Shrivastava, A., Gupta, A.: Contextual priming and feedback for faster R-CNN. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 330–348. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_20
41. Tokmakov, P., Alahari, K., Schmid, C.: Learning video object segmentation with visual memory. In: ICCV (2017)
42. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3D convolutional networks. In: ICCV (2015)
43. Tripathi, S., Lipton, Z.C., Belongie, S., Nguyen, T.: Context matters: Refining object detection in video with recurrent neural networks. arXiv preprint arXiv:1607.04648 (2016)
44. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: a neural image caption generator. In: CVPR (2015)
45. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. IJCV **63**(2), 153–161 (2005)

46. Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfinder: real-time tracking of the human body. PAMI **19**, 780–785 (1997)
47. Xu, K., et al.: Show, attend and tell: Neural image caption generation with visual attention. arXiv preprint arXiv:1502.03044 (2015)
48. Yang, B., Yan, J., Lei, Z., Li, S.Z.: Craft objects from images. In: CVPR (2016)
49. Zheng, S., et al.: Conditional random fields as recurrent neural networks. In: CVPR (2015)
50. Zhu, X., Dai, J., Yuan, L., Wei, Y.: Towards high performance video object detection. In: CVPR (2018)
51. Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: ICCV (2017)