An Experimental Study on NTMT: a Neural Turing Machine Framework for Tracking

Anonymous ICCV submission

Paper ID 2883

Abstract

Recent online learning methods advanced the state-of-the-art by utilizing deep learning to learn a domain specific model to adapt the local appearance changes. We introduce a new, one-shot learning approach, the Neural Turing Machine Tracking (NTMT), where the observed moving objects are learned and simply remembered in an external memory during the tracking process. With the NTM consisting of a deep neural network coupled with an addressable memory module, this is naturally the first choice to tackle the learning and memorization tasks pertinent to object tracking. The major challenge is to design a data preprocessing scheme that works naturally with existing NTM architecture. We utilize the ImageNet ILSVRC video detection dataset [27] to train and test our NTMT. The convincing results demonstrate the high potential for NTMT.

1. Introduction

Visual object tracking is one of the core problems of computer vision. Given a video with various moving objects, a bounding box specifying the object of interest in a single frame, our goal is to automatically determine the object's bounding box (or otherwise the object is invisible) in subsequent frames. Ideally, the tracking is performed in real time at the video frame rate.

State-of-the-art approach is to learn online a model of the object's appearance using examples directly derived from the input video. Recent methods such as TLD [17], Struck [14] and KCF [15] have demonstrated great successes in video object tracking. However, such learning-based trackers have several fundamental limitations. First, since the training data is the input video itself, the training data has limited variety in the appearance of the target object, and thus the learned model is relatively simple compared to the deep learning models in other areas in computer vision, where large datasets are available for training. Second, the weights of the network must be adapted to new appearance not seen before, where online stochastic gradient descent even on simple models may compromise the real-time performance. Third, online methods do not have long term memory, which means that they still suffer from accumulation errors. For example, when an object is moving slowly across the screen toward its occluder from behind, an online tracker may soon start to track the occluder instead of the object. Fourth, some methods use tracking-by-detection such as MDNet [23], while other methods in face-/hand-tracking (e.g., [34, 30]) are category specific and they cannot track general objects. Lastly, detection-based tracking approaches often do not model background, because their discriminative classifiers are trained to detect foreground objects.

To address the above issues, we propose one-shot learning and the Neural Turing Machine [13] for tracking (NTMT). In essence, NTM combines (deep) neural networks and an addressable memory module, and the combined system is analogous to a Turing Machine. Despite the scarcity of related research, [13, 29, 12] have revealed the high potential in coupling external memory with deep neural networks for understanding sequential data. NTM with its augmented memory can quickly encode and retrieve new information, and in the limit of "one-shot" learning, single observations are used in learning abrupt shifts in behavior, or the local object appearance in the case of tracking.

This paper proposes and implements the first NTM tracker: we train and test our NTMT using the ImageNet ILSVRC video detection dataset [27]. With a long-term memory, we formulate the tracking problem as a one-shot learning problem. Our NTMT has the following properties:

- one shot learning where the NTMT can be trained even with one object instance
- fast training as it does not need to back-propagate to retrain the network
- long-term memory for long-range tracking
- class-agnostic with pre-training using ILSVRC video detection dataset
- foreground and background are modeled (memorized) the same way

With the memory module, NTMT can forget the old and remember the new appearance of the target objects to track objects under deformation.

2. Related Work

Comprehensive review of related work in object tracking is out of the scope of this NTMT paper, and we refer readers to [33, 35] for a survey. Here we focus on reviewing the template tracking, generative models (NTMT learns, forgets the old and remembers the new appearance of the target object), tracking-by-detection (NTMT utilizes deep learning for object detection) and online learning-based tracking (c.f. NTMT uses one-shot learning). A brief review on Neural Turing Machine will also be given.

2.1. Templates and Generative Models

Simply put, our NTMT framework detects objects in each frame and tracks each object by learning and remembering its changing appearance over subsequent frames. This resembles the adaptive template tracking scheme [7, 21, 24] (vs static templates scheme [5]) where the target template is updated from previous frames. Previous templates are limited because they represent a single appearance and are vulnerable to drifting in long-range tracking.

To model more appearance variations, the generative models have been proposed. Early solutions date back to the Kalman filter [18], the particle filter [6] with their numerous extensions and variations (see [33]). Recent generative models can be built online during runtime [25, 19]. In practice, it is very difficult to precisely determine the distributions of the observations/measurements, where mismatch in the generative model and measurement distributions can lead to tracking failure manifested as drifting or losing tracks in the presence of severe occlusions.

2.2. Tracking-by-Detection and Online Learning

The tracking problem can be formulated as an online learning problem [33, 35]: given an initial bounding box containing the target object, learn a classifier online and evaluate it at multiple locations in subsequent frames. Each new detection can be used to update the model. Despite the success of online methods, since the training data is the video itself, it inherently limits the richness of the model they can learn.

Early tracking-by-detection includes support vector tracking [1], random forest classifiers [28], and boosting [11, 2]. These classical methods have been made online for object tracking. Using a large number of image features, in [14] an online learning approach was proposed using structured output SVM and Gaussian kernels to directly predict the target's location. In [17], the online tracking task was decomposed into tracking (following the object from frame to frame), learning (estimating the detector's errors and updating the detector), and detection (localizing all observed appearance and correcting the tracker). Structural constraints were used to guide the sampling process of a boosting classifier. In [15], high-speed tracking was proposed using kernelized correlation filters. Using the circulant property of the data matrix, it can be diagnoalized Discrete Fourier Transform, which can drastically reduce both storage and computation. In [4], rather than online learning, the authors propose a basic tracking algorithm to work with a fully-convolutional Siamese network trained end-to-end on the ILSVRC15 dataset for object detection and tracking in video, which achieves real-time performance and state-of-the-art accuracy.

2.3. Neural Turing Machine

NTMT tracks objects from a video sequence. Recurrent neural networks or RNNs [26] are a family of neural networks for processing sequential data which can remember and process past information. RNNs are known to be Turing-Complete [31] and therefore have the capacity to simulate arbitrary procedures. One of the most effective sequence models used in practical applications are gated RNNs, including long short-term memory (LSTM) [16].

The Neural Turing Machine (NTM) was first introduced in [13]. NTM is a RNN in a Turing Machine architecture. All of the components are differentiable so it can be efficiently trained with gradient descent. The first results show that the powerful NTM can infer useful algorithms, such as copy and paste, sorting or associative recall given input and output examples *only*. The NTM architecture consists of a controller unit and a memory unit. Similar to typical neural networks, the controller interacts with the external world via input and output vectors. In contrast to a standard neural network, the controller network interacts with a memory matrix using read and write operations. With the same processing ability on temporal sequences, NTM's access to the external memory modules make it more powerful than LSTM. With the external memory module NTM has also shown success in one-shot learning.

One-shot learning first appeared in [9] for learning object categories, where a generative model and variational Bayesian framework are used to learn categories from a handful of training examples, in the limiting case only one example. It emphasizes on knowledge transfer and shared features so that existing knowledge allows learning new knowledge with minimal training examples [22, 8, 10, 3]. Memory-augmented neural networks are particularly suitable for one-shot learning [29], where NTM can encode and retrieve new information into the working memory. In the following, we will describe our network design that is capable of memorizing and retrieving the correct features during the tracking process. We will also describe how to

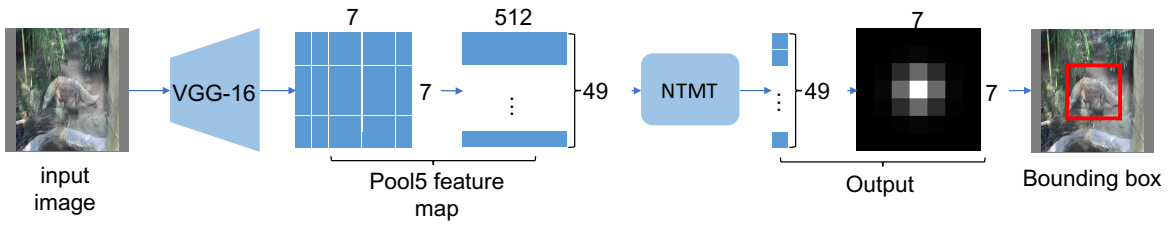

Figure 1. Overview of the NTMT.

tackle the technical challenges in making the original NTM amenable to high-dimensional sequential data, and in training our modified NTM for video object tracking.

3. Overview of NTMT

We give a technical overview of NTMT (Figure 1), which will be detailed in the following sections. Our approach assumes the bounding box of the tracking object to be given in the first frame of the input video. For each video frame, we extract local regions around the bounding box of the current and next frame. The extracted regions are normalized such that the bounding box is of a fixed size in a fixed aspect ratio. Using the VGG network [32], we extract high level *Pool5* features within the normalized extracted regions. The possible location of the tracking object in the next frame is quantized into a 7×7 grid.

Within each time step, we pass the VGG feature of each quantized location and the read/write vectors to the NTM. The NTM retrieves and compares the stored features in its memory module, and then return a confidence to determine whether the input VGG feature belongs to the tracking object or otherwise. This process is computed for each quantized location, resulting in a heat map to measure the possible location of the tracking object in the next frame. Lastly, we apply Marching Squares (2D version of [20]) to find sub-pixel contours, and non-maximum suppression to extract the bounding box location of the tracking object in the current frame.

Throughout the tracking process, the feature vectors stored in the memory module of NTM are updated automatically. Since feature vectors of both the tracking object and the background are passed into the NTM, the NTM naturally models the background together with the tracking object in a unified manner, thus achieving one-shot learning with single frame observations.

4. NTMT

Central to the novelty of NTM is its external memory module, which enables reliable long-term storage over its LSTM counterpart. In NTMT, when the first frame is presented, the features are hashed and stored in memory together with corresponding labels. On subsequent frames, each input feature is compared with memory through content focused addressing, and the corresponding label is retrieved, the feature stored being updated when necessary.

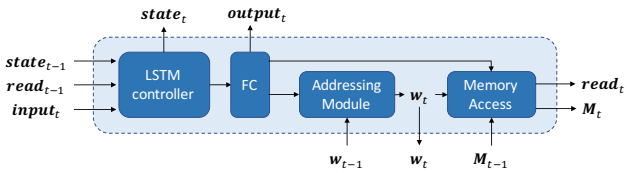

Figure 2. Neural Turing Machine architecture.

In this paper, we follow the architecture design of NTM proposed in [13] with modifications on how the data are presented to adopt the NTM to our tracking task. During each update cycle, the controller network (section 4.4) of NTMT receives the input features encoded from the current video frame and returns an output to indicate the location of the object being tracked.

4.1. Reading

Let $\mathbf{w}_t$ be a vector weightings over the n locations emitted by a read head at time t, which obeys the unity constraint in [13]. In NTMT, the read vector $\mathbf{read}_t$ of length m returned by the head is defined as matrix multiplication with $\mathbf{M}$ in memory:

$$\mathbf{read}_t = \mathbf{M}_{t-1}\mathbf{w}_t^T \tag{1}$$

4.2. Writing

Let $\mathbf{M}_t$ be the contents of the $m \times n$ memory matrix at time t, where n is the number of memory locations, and m is the depth of each memory location. NTMT memory write has two operations: an erase operation followed by an add operation, which are inspired from the input and forget gates in LSTM.

Given a weighting $\mathbf{w}_t$ emitted by a write head at time t alongside with an erase vector $\mathbf{e}_t$, where all of the m elements are within $[0, 1]$, we form the erase matrix $\mathbf{E}_t$ by computing the outer product between $\mathbf{e}_t$ and $\mathbf{w}_t$:

$$\mathbf{E}_t = \mathbf{1} - \mathbf{e}_t\mathbf{w}_t^T \tag{2}$$

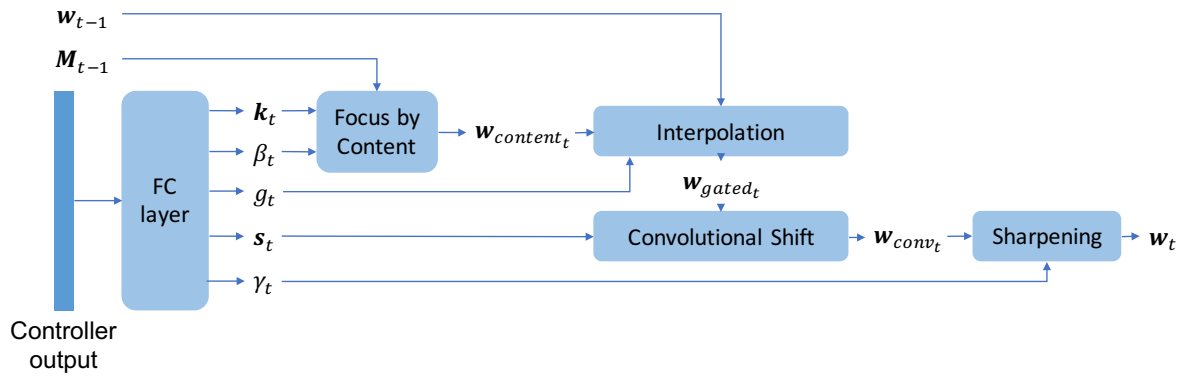

Figure 3. Memory addressing of the NTMT.

where $\mathbf{1}$ is a $m \times n$ matrix of 1's. Hence, the memory location will be reset to zero if the corresponding entry in $\mathbf{E}$ is 1. The write head also emits an add vector $\mathbf{a}_t$ of length m. The corresponding memory to write $\mathbf{A}_t$ is given by output product between $\mathbf{a}_t$ and $\mathbf{w}_t$:

$$\mathbf{A}_t = \mathbf{a}_t \mathbf{w}_t^T \tag{3}$$

Therefore, the updated memory is given by

$$\mathbf{M}_t = \mathbf{M}_{t-1} \cdot \mathbf{E}_t + \mathbf{A}_t \tag{4}$$

where $\cdot$ denotes the element-wise product between $\mathbf{M}_{t-1}$ and $\mathbf{E}_t$.

4.3. Memory Addressing

Memory addressing is another important component in NTMT which defines where in the memory should the network read from or write to. We produce the final address for each read or write head through two stages, focusing by content and sequential addressing (Figure 3).

Focusing by Content. Content Focusing is central to our task, as for each feature of subsequent frames presented, the tracker needs to find the most similar first-frame feature, and retrieve its confidence. First of all, a memory fetch key $\mathbf{k}_t$ of size m is produced by the controller. Then it is multiplied with the memory $\mathbf{M}_t$ after both are L_2-normalized to give a *cosine similarity*, which is used to produce the content focused address. Note that the $m \times n$ matrix $\mathbf{M}_t$ is normalized on a per-column basis.

$$\text{sim}_t = \frac{\mathbf{k}_t}{\|\mathbf{k}_t\|} \frac{\mathbf{M}_t}{\|\mathbf{M}_t\|_{col}} \tag{5}$$

A scalar key strength β_t is also drawn from the controller. It is multiplied with the similarity and produce the content-focused memory address $\mathbf{w}_{content_t}$ after applying a *softmax* operation.

$$\mathbf{w}_{content_t} = \text{softmax}(\beta_t \text{sim}_t) \tag{6}$$

Sequential Addressing. Although Content Focusing is principal way of memory access in NTMT, when the first frame features are presented, the memory is blank, thus the network should employ a sequential writing scheme. This is achieved by interpolation the address vector with address produced in previous time step and convolutional shifting. To achieve interpolation, g_t is a scalar value between 0 and 1 drawn from controller output.

$$\mathbf{w}_{gated_t} = g_t \mathbf{w}_{content_t} + (1 - g_t)\mathbf{w}_{t-1} \tag{7}$$

After interpolation, the address is allowed shifted by circular convolution. The kernel $\mathbf{s}_t$ is a 3-vector drawn from controller output. For example, a kernel of $[0\ 0\ 1]$ will result in a left shift of address and vice versa. A *softmax* is used to keep the 3-vector summing to 1.

$$\mathbf{w}_{conv_t} = \mathbf{s}_t * \mathbf{w}_{gated_t} \tag{8}$$

Since convolutional shifting tends to result in blurred weights, a final sharpening is applied. A sharpening scalar $\gamma_t \geq 1$ is extracted from controller output and applied as follows.

$$\mathbf{w}_t = \frac{\mathbf{w}_{conv_t}^{\gamma_t}}{\sum \mathbf{w}_{conv_t}^{\gamma_t}} \tag{9}$$

4.4. Controller Network

We use an LSTM controller with a single layer and a hidden size of 500. The input at time t is concatenated with the previous read value $\mathbf{read}_{t-1}$ before it is fed to the controller. The output of the controller is processed by two fully-connected layers to extract the parameters needed in memory addressing and the final output. The architecture is illustrated in Figure 2.

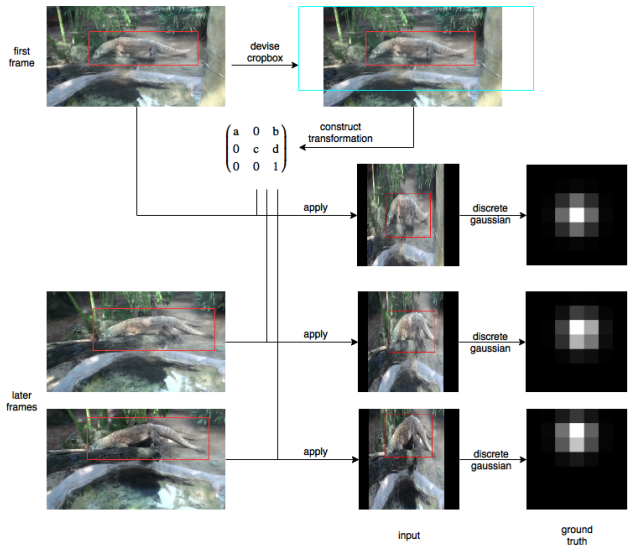

Figure 4. Ground truth preparation for NTMT.

5. Implementation

We use the ImageNet ILSVRC video detection dataset [27] to train and test our NTMT.

We observe that in NTM works [13, 29] NTM was applied on relatively low-dimensional inputs. In object tracking however, the VGG-16 *Pool5* feature of an input frame has 512 channels defined on a 7×7 feature map, totaling 25088 dimensions. To reduce the dimensionality, instead of feeding one frame at a step, we feed the network one feature at a step, such that a frame is read in at least 49 steps. The training and data pre-processing are detailed in the next section.

5.1. NTMT Data Preparation

Figure 4 shows the steps in preparing input and ground-truth for training NTMT. For the first frame, an ROI crop box is drawn such that the ground truth bounding box is at its center and has width and height $\frac{3}{7}$ that of ROI. Then the ROI is cropped and re-sized to 224×224 and the transformation matrix is stored. When training, all subsequent frames are normalized using the same transformation, and when testing, the transformation matrix is updated every step based on the output of previous frame. The sequence is terminated once the object goes out of scope.

The ground truth is a discrete 2D Gaussian with $\sigma = 0.75$ on the 7×7 grid, the center of which corresponds to the bounding box after the transformation. The ground truth is essentially a probability distribution of the target location.

5.2. NTMT Input Presentation

We use the ImageNet ILSVRC2015 VID dataset and extract each object into separate sequences if there are multiple objects in the video. We obey the training and validation split of the original dataset.

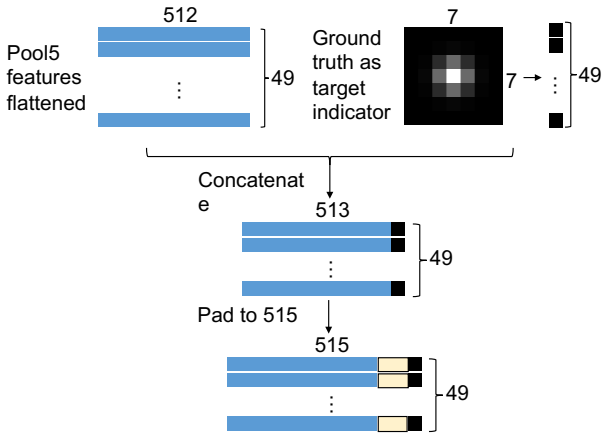

Figure 5. NTMT input presentation for first frame.

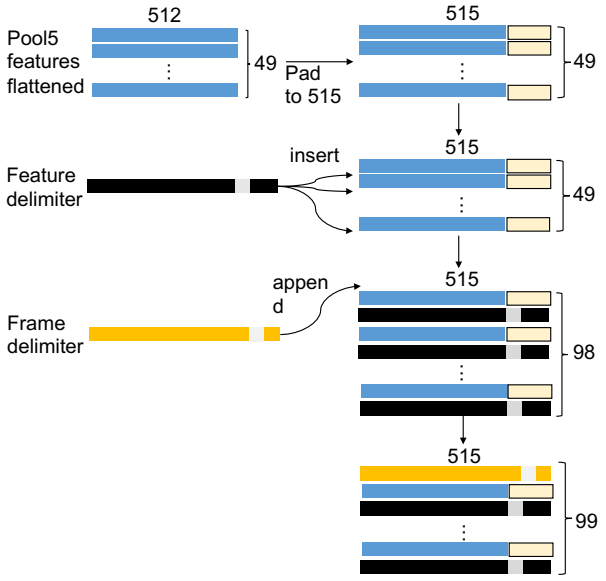

Figure 6. NTMT input presentation for subsequent frames.

Figure 5 and Figure 6 show the details of how the inputs are presented to the NTMT. First, using the VGG-16 network [32], we extract high level *Pool5* features within the normalized ROI. The flattened feature of the first frame will be combined with the flattened 7×7 ground-truth, the latter functions as target selector in the initialization step of tracker. This target selector is replaced with a dummy zero vector on all subsequent frames. Then every feature in subsequent frames is appended with a feature delimiter, upon seeing which the tracker should report the confidence of the feature it has just seen. Finally a frame delimiter is prepended to every frame to signal the tracker of the begin-

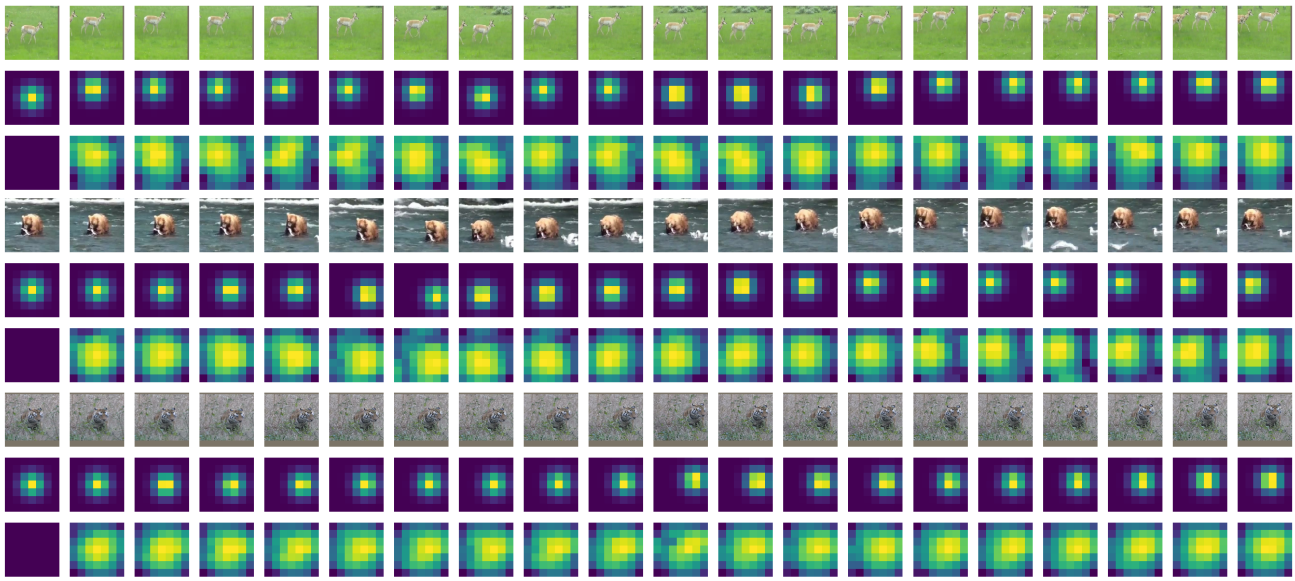

Figure 7. NTMT validation output.

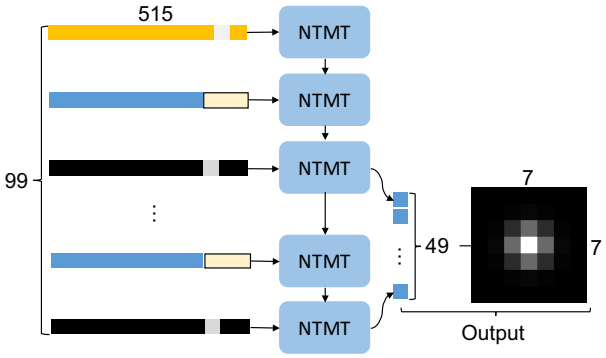

Figure 8. NTMT output collection.

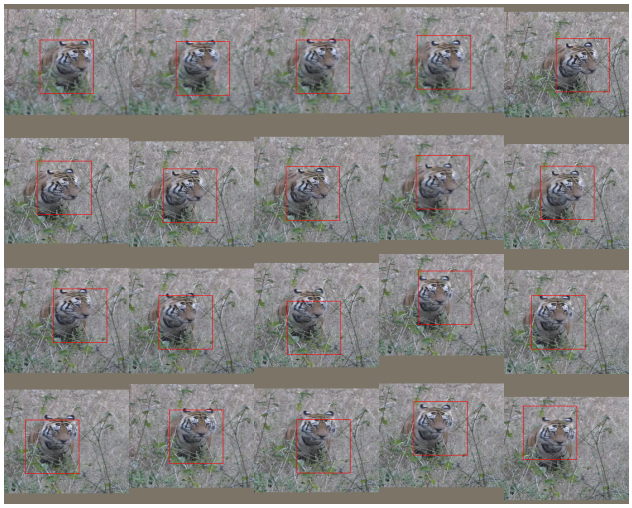

Figure 9. NTMT test output.

ning of a new frame. Each of the two types of delimiters as well as the target selector occupies an additional channel exclusively, causing the final input to have 515 channels. The final length of input is $49 + 99 \times (sequencelength - 1)$. The outputs of the time steps when feature delimiter is fed are collected as shown in Figure 8 and a *softmax* operation is applied to reconstruct the heat map. All other outputs are discarded and do not participate in the loss calculation or back propagation. shows the input and output for training. Finally, a *softmax cross entropy with logits* is used as loss function. The model is validated on the validation set of ILSVRC2015 VID dataset. i Some validation results are shown in Figure 7.

5.3. NTMT Testing

After the NTMT has been trained, we test the trained model using the validation set of ILSVRC2015 VID dataset. The first frame together with a ground truth bounding box are used to initialize the tracker. The tracker normalizes the frame the same way as described in Figure 4. The input bounding box will be stored as initial bounding box after transformation for later use. For each output heat map, we apply Marching Squares algorithm [20] with thresholds of $0.01\ 0.02\ 0.03\ \cdots\ 0.09\ 0.1$ to find contours at sub-pixel locations. Then we calculate the enclosing rectangle of each

Figure 10. NTMT additional test outputs.

contour, and rank the rectangles with the IoU metric against the normalized initial bounding box. The rectangle with the highest IoU is kept. The center of this region is calculated and the initial bounding box is translated to this center and used as as output. The inverse of the transformation matrix is applied to decode the normalized bounding box to input scale and reported as output. The output of time t is used to normalize the input at time $t + 1$, effectively shifting attention of the "camera" based on previous observation. A sample result is shown in Figure 9.

6. Conclusion

This paper presents a fundamentally novel approach to object tracking in videos. Our NTMT is motivated by the recent development of Neural Turing Machine, and how human eyes track objects when they are seen. NTMT can be regarded as one-shot learning where the changing appearance of the observed scene in the current frame, both the background and moving foreground, is learned by memorizing and recalling from the memory during the tracking process. We have presented our technical contributions that makes NTM practical in processing high-dimensional sequential data, and have demonstrated its efficacy by tracking non-rigid objects undergoing deformation and topological changes where online learning and other conventional tracking approaches may easily fail. We have only scratched the surface of the powerful NTM may offer in tracking, and we hope the idea and technical contributions presented in this experimental paper will spawn future research on NTMT and NTM in general.

References

[1] S. Avidan. Support vector tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(8):1064–1072, Aug. 2004.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, Aug. 2011.

[3] E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *IEEE CVPR*, 2005.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. *CoRR*, abs/1606.09549, 2016.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003.

[6] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[7] N. D. H. Dowson and R. Bowden. Simultaneous modeling and tracking (smat) of feature sets. In *IEEE CVPR*, volume 2, pages 99–105 vol. 2, June 2005.

[8] L. Fei-fei. Knowledge transfer in learning to recognize visual object classes. In *In: International Conference on Development and Learning (ICDL*, 2006.

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[10] M. Fink. Object classification from a single example utilizing class relevance metrics. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 449–456, Cambridge, MA, USA, 2004. MIT Press.

[11] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] A. Graves and *et al.* Hybrid computing using a neural network with dynamic external memory. *Nature*, 538, 2016.

[13] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[14] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. M. Cheng, S. L. Hicks, and P. H. S. Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, Oct 2016.

[15] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[17] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012.

[18] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

[19] J. Kwon and K. M. Lee. Visual tracking decomposition. In *IEEE CVPR*, pages 1269–1276, June 2010.

[20] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, 1987.

[21] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):810–815, June 2004.

[22] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 464–471 vol.1, 2000.

[23] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015.

[24] A. Rahimi, L.-P. Morency, and T. Darrell. Reducing drift in differential tracking. *Comput. Vis. Image Underst.*, 109(2):97–111, Feb. 2008.

[25] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *Int. J. Comput. Vision*, 77(1-3):125–141, May 2008.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[28] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1393–1400, Sept 2009.

[29] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap. One-shot learning with memory-augmented neural networks. *CoRR*, abs/1605.06065, 2016.

[30] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3633–3642, New York, NY, USA, 2015. ACM.

[31] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 440–449, New York, NY, USA, 1992. ACM.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[33] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1442–1468, July 2014.

[34] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004.

[35] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomput.*, 74(18):3823–3831, Nov. 2011.