# Learning Dynamic Siamese Network for Visual Object Tracking Supplementary Document

Anonymous ICCV submission

Paper ID 688

In this supplementary, we first add three important interpretations of our Dynamic Siamese network, which elaborates the details of network architecture and training implementation. Then, we validate the effective of proposed training strategy and online transformations.

## 1. Dynamic Siamese Network

We train our dynamic Siamese (DSiam) network on ILSVRC-2015 video dataset that has no overlap with popular benchmarks [1]. Thus, there is no overfitting risk to train DSiam network. Furthermore, We can construct DSiam network on multi layers of any feasible generally- or particularly-trained CNNs.

### 1.1. Multi-layer network architecture

We have shown and introduced DSiam network on single layer in our submitted manuscript. Here, we show the multi-layer DSiam in Fig. 1. To make it easy to understand, we denote four main parts of single-layer DSiam as 'var', 'Lvar', 'sup' and 'Lsup'. As shown in left subfigure of Fig. 1, 'var' means the target appearance variation transformation, i.e. $\tilde{\mathbf{T}} = \mathbf{V} * \mathbf{T}$; 'sup' denotes the background suppression transformation, i.e. $\tilde{\mathbf{F}}_z = \mathbf{W} * \mathbf{F}_z$; 'Lvar' and 'Lsup' represent processes of learning $\mathbf{V}$ and $\mathbf{W}$, respectively. For two-layer DSiam, each layer is equipped with the four parts, i.e. 'var', 'Lvar', 'sup' and 'Lsup', as shown in the right subfigure of Fig. 1 and produces a response map $\mathbf{S}^{l_1}$ or $\mathbf{S}^{l_2}$. Then, two response maps are fused via an elementwise fusion layer denoted as 'Elefusion' and defined as,

$$\mathbf{S} = \sum_{l \in \{l_1, l_2\}} \boldsymbol{\Upsilon}^l \odot \mathbf{S}^l, \tag{1}$$

where $\sum_{l \in \{l_1, l_2\}} \boldsymbol{\Upsilon}^l = \mathbf{1}$. $\Upsilon^l$ is learnt through joint training.

Dynamic Siamese network shown in Fig. 1 is actually a tracking process. Thus, to training such network is to find a good tracker. Specifically, at the beginning, the first frame $\mathbf{I}_1$ is cropped to get the target template $\mathbf{O}_1$ that is resized to $127 \times 127$ via 'Crop' layer. Then, we extract deep features of $\mathbf{O}_1$ and get $\mathrm{f}^{l_1}(\mathbf{O}_1)$ and $\mathrm{f}^{l_2}(\mathbf{O}_1)$. Here, we use the SiamFC as the neural network with $l_1$ and $l_2$ being conv5 and conv4, respectively. Note, our framework could also utilize other networks, as demonstrated in experiment part, DSiam could also get good performance with pre-trianed Vgg19 network. At frame $t$, an input frame $\mathbf{I}_t$ passes through 'Crop' layer and is cropped to get a search region, i.e. $\mathbf{Z}_t$, that is centered at the maximum value of $\mathbf{S}_{t-1}$ and is resized to $255 \times 255$. Then, we get $\mathrm{f}^{l_1}(\mathbf{Z}_t)$ and $\mathrm{f}^{l_2}(\mathbf{Z}_t)$ and transform these two deep features through background suppression transformations, i.e. 'sup' in Fig. 1. Meanwhile, $\mathrm{f}^{l_1}(\mathbf{O}_1)$ and $\mathrm{f}^{l_2}(\mathbf{O}_1)$ are transformed through target appearance variation transformation, i.e. 'var' in Fig. 1. The transformed deep features of $\mathbf{O}_1$ and $\mathbf{Z}_t$ are used to perform correlation and get the response maps $\mathbf{S}^{l_1}$ and $\mathbf{S}^{l_2}$, which are fused via Eq. (1) and get $\mathbf{S}_t$. After tracking, we can use $\mathbf{S}_t$ to get target template at frame $t$, i.e. $\mathbf{O}_t$, and region $\mathbf{G}_t$ centered at $\mathbf{O}_t$ but having the same size with $\mathbf{Z}_t$. Then, we get $\bar{\mathbf{G}}_t$ that mainly contain the target by multiplying $\mathbf{G}_t$ with a Gaussian weight map via 'Eltwise' layer. With deep features of $\mathbf{O}_t$ and $\mathbf{O}_1$, we learn the $\mathbf{V}$ via 'Lvar' in Fig. 1. With deep features of $\mathbf{G}$ and $\bar{\mathbf{G}}$, we learn the $\mathbf{W}$ via 'Lsup' in Fig. 1.

### 1.2. Back propagation of new layers

In following, we introduce how to propagate gradients w.r.t. Loss in new layers appeared in DSiam, including RLR layer, CirConv layer, Elefusion layer and Crop layer.

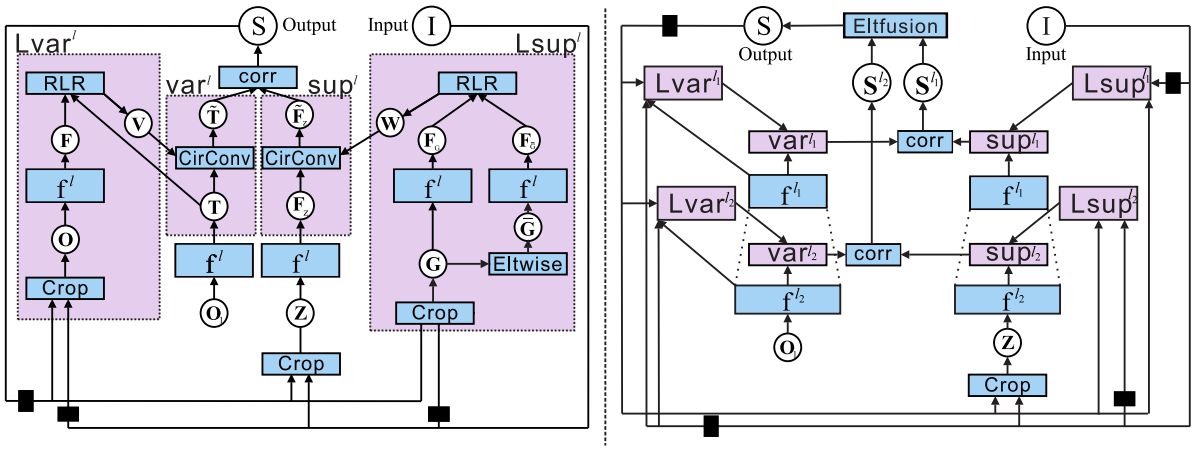


Figure 1. Dynamic Siamese networks. Left subfigure shows the DSiam on single layer, in which we use four abbreviations to denote four processes. 'var' represents the target appearance variation transformation, i.e. $\tilde{\mathbf{T}} = \mathbf{V} * \mathbf{T}$; 'Lvar' denotes process of learning variation transformation, i.e. $\mathbf{V}$; 'sup' is the background suppression transformation, i.e. $\tilde{\mathbf{F}}_z = \mathbf{W} * \mathbf{F}_z$; 'Lsup' denotes the process of learning background suppression transformation, i.e. $\mathbf{W}$. Right subfigure shows the DSiam on two layers. Each layer is equipped with four processes denoted above to produce a response map. Then, two response maps are fused through an elementwise fusion layer denoted by 'Eltfusion'.

For RLR and CirConv layers, we use vector as an example which can be extended to tensor case. Given two vectors $\mathbf{X} \in \Re^{n\times 1}$ and $\mathbf{Y} \in \Re^{n\times 1}$, we aim to calculate a transformation matrix $\mathbf{R} \in \Re^{n\times 1}$ to make $\mathbf{X}$ similar to $\mathbf{Y}$, i.e.

$$\mathbf{Y} = \mathbf{R} * \mathbf{X}, \tag{2}$$

where '$*$' is the circular convolution. Thus, Eq. (2) is denoted as the CirConv layer. $\mathbf{R}$ can be solved by

$$\mathbf{R} = \arg\min_{\mathbf{R}} \|\mathbf{R} * \mathbf{X} - \mathbf{Y}\|^2 + \lambda\|\mathbf{R}\|^2, \tag{3}$$

where $\lambda$ is to control the regularization degree and is regarded as a parameter during training. Thus, Eq. (3) represents the RLR layer.

**Gradients of CirConv layer**

Considering Eq. (2), during back propagation, given $\nabla_{\mathbf{Y}} L$, we aim to calculate $\nabla_{\mathbf{X}} L$ and $\nabla_{\mathbf{R}} L$, respectively.

For $\nabla_{\mathbf{X}} L$, we can write it as

$$\nabla_{\mathbf{X}} L = (\frac{\partial \mathbf{Y}}{\partial \mathbf{X}})^{\mathrm{T}} \nabla_{\mathbf{Y}} L, \tag{4}$$

where $\frac{\partial \mathbf{Y}}{\partial \mathbf{X}} \in \Re^{n\times n}$ is the Jacobian matrix. Furthermore, we can rewrite the circular convolution as matrix multiplication form and get

$$\mathbf{Y} = \mathbf{R} * \mathbf{X} = \mathrm{C}(\mathbf{X})^{\mathrm{T}}\mathbf{R} = \mathrm{C}(\mathbf{R})^{\mathrm{T}}\mathbf{X}, \tag{5}$$

where $\mathrm{C}(\mathbf{X})$ is to output a circulant matrix each row of which is the shift version of $\mathbf{X}$. Then, we can get

$$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}} = \mathrm{C}(\mathbf{R})^{\mathrm{T}}. \tag{6}$$

Thus, we have

$$\nabla_{\mathbf{X}} L = \mathrm{C}(\mathbf{R}) \nabla_{\mathbf{Y}} L. \tag{7}$$

Considering the fact that all circulant matrices can be diagonalized by the Discrete Fourier Transform (DFT), $\mathrm{C}(\mathbf{R})$ can be expressed as

$$\mathrm{C}(\mathbf{R}) = \mathbf{F}\mathrm{diag}(\hat{\mathbf{R}})\mathbf{F}^{\mathrm{H}}, \tag{8}$$

where $\mathbf{F}$ is a constant matrix which is irrelevant to $\mathbf{R}$, $\hat{\mathbf{R}}$ denotes the DFT of $\mathbf{R}$, and we have $\hat{\mathbf{R}} = \mathscr{F}(\mathbf{R})$ and $\mathscr{F}(\mathbf{R}) = \sqrt{n}\mathbf{F}\mathbf{R}$ Eq. 4 is equivalent to

$$\begin{aligned}\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{X}}L &= \mathbf{F}^{\mathrm{H}}(\mathbf{F}\mathrm{diag}(\hat{\mathbf{R}})\mathbf{F}^{\mathrm{H}})\nabla_{\mathbf{Y}}L \\ &= \mathbf{F}^{\mathrm{H}}\mathbf{F}\mathrm{diag}(\hat{\mathbf{R}})\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{Y}}L \\ &= \mathrm{diag}(\hat{\mathbf{R}})\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{Y}}L,\end{aligned} \tag{9}$$

afterwards, by simple algebra as follows

$$\begin{aligned}&(\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{X}}L)^{\mathrm{H}} = (\mathrm{diag}(\hat{\mathbf{R}})\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{Y}}L)^{\mathrm{H}}, \\ &(\nabla_{\mathbf{X}}L)^{\mathrm{H}}\mathbf{F} = (\nabla_{\mathbf{Y}}L)^{\mathrm{H}}\mathbf{F}\mathrm{diag}(\hat{\mathbf{R}}), \\ &(\nabla_{\mathbf{X}}L)^{\mathrm{T}}\mathbf{F} = (\nabla_{\mathbf{Y}}L)^{\mathrm{T}}\mathbf{F}\mathrm{diag}(\hat{\mathbf{R}}), \\ &((\nabla_{\mathbf{X}}L)^{\mathrm{T}}\mathbf{F})^{\mathrm{T}} = ((\nabla_{\mathbf{Y}}L)^{\mathrm{T}}\mathbf{F}\mathrm{diag}(\hat{\mathbf{R}}))^{\mathrm{T}}, \\ &\mathbf{F}\nabla_{\mathbf{X}}L = \mathrm{diag}(\hat{\mathbf{R}})\mathbf{F}\nabla_{\mathbf{Y}}L.\end{aligned} \tag{10}$$

Hence, Eq. 9 is equivalent to

$$\hat{\nabla}_{\mathbf{X}}L = \mathrm{diag}(\hat{\mathbf{R}})\hat{\nabla}_{\mathbf{Y}}L. \tag{11}$$

We may go one step further, since the product of a diagonal matrix and a tensor is just their element-wise product,

$$\hat{\nabla}_{\mathbf{X}}L = \hat{\mathbf{R}} \odot \hat{\nabla}_{\mathbf{Y}}L, \tag{12}$$

where $\hat{\nabla}_{\mathbf{X}}L$ denotes the fourier form of $\nabla_{\mathbf{X}}L$.
Finally, we get $\nabla_{\mathbf{X}}L$

$$\nabla_{\mathbf{X}}L = \mathscr{F}^{-1}(\hat{\nabla}_{\mathbf{X}}L). \tag{13}$$

Similarly, we can also get $\hat{\nabla}_{\mathbf{R}}L$ and $\nabla_{\mathbf{R}}L$,

$$\begin{aligned}\hat{\nabla}_{\mathbf{R}}L &= \hat{\mathbf{X}} \odot \hat{\nabla}_{\mathbf{Y}}L \\ \nabla_{\mathbf{R}}L &= \mathscr{F}^{-1}(\hat{\nabla}_{\mathbf{Y}}L).\end{aligned} \tag{14}$$

The gradient results of $\nabla_{\mathbf{R}}L$ and $\nabla_{\mathbf{X}}L$ in our submitted manuscript are represented in a wrong way. Eq. (14) and Eq. (13) should be the final results.

**Gradients of RLR layer**

When the above '*' circulant product, Eq. 3 has a close-form solution as follows,

$$\mathbf{R} = (\mathrm{C}(\mathbf{X})^{\mathrm{H}}\mathrm{C}(\mathbf{X}) + \lambda\mathbf{I})^{-1}\mathrm{C}(\mathbf{X})^{\mathrm{H}}\mathbf{Y}, \tag{15}$$

where $\mathbf{I}$ denotes the identity matrix.

Considering Eq. 15, during back propagation, given $\nabla_{\mathbf{R}}L$, we aim to calculate $\nabla_{\mathbf{X}}L$, $\nabla_{\mathbf{Y}}L$ and $\nabla_{\lambda}L$.

For $\nabla_{\mathbf{X}}L$, we can write it as

$$\begin{aligned}\nabla_{\mathbf{X}}L &= (\frac{\partial\hat{\mathbf{X}}}{\partial\mathbf{X}})^{\mathrm{T}}\nabla_{\hat{\mathbf{X}}}L \\ &= \sqrt{\mathrm{n}}\mathbf{F}\nabla_{\hat{\mathbf{X}}}L \\ &= \sqrt{\mathrm{n}}\mathbf{F}(\frac{\partial\hat{\mathbf{R}}}{\partial\hat{\mathbf{X}}})^{\mathrm{T}}\nabla_{\hat{\mathbf{R}}}L \\ &= \sqrt{\mathrm{n}}\mathbf{F}(\frac{\partial\hat{\mathbf{R}}}{\partial\hat{\mathbf{X}}})^{\mathrm{T}}\frac{1}{\sqrt{\mathrm{n}}}\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{R}}L \\ &= \mathbf{F}(\frac{\partial\hat{\mathbf{R}}}{\partial\hat{\mathbf{X}}})^{\mathrm{T}}\mathbf{F}^{\mathrm{H}}\nabla_{\mathbf{R}}L,\end{aligned} \tag{16}$$

where $\mathbf{F}$ is a constant matrix, $\hat{\mathbf{X}}$ denotes the DFT of $\mathbf{X}$, and we have $\hat{\mathbf{X}} = \mathscr{F}(\mathbf{X}) = \sqrt{n}\mathbf{F}\mathbf{X}$.

For $\frac{\partial \hat{\mathbf{R}}}{\partial \hat{\mathbf{X}}}$, similar to Eq. 9, we can get

$$\hat{\mathbf{R}} = \frac{\hat{\mathbf{X}}^{\star} \odot \hat{\mathbf{Y}}}{\hat{\mathbf{X}}^{\star} \odot \hat{\mathbf{X}} + \lambda}. \tag{17}$$

We then set $\hat{\mathbf{X}} = \mathbf{a} + \mathbf{b}\mathrm{i}$ where $\mathbf{a}, \mathbf{b} \in \Re^{\mathrm{n}\times 1}$, thus Eq. 17 can be written as

$$\hat{\mathbf{R}} = \frac{(\mathbf{a} - \mathbf{b}\mathrm{i}) \odot \hat{\mathbf{Y}}}{\mathbf{a}^2 + \mathbf{b}^2 + \lambda}. \tag{18}$$

Then we can get

$$\begin{aligned} \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{a}} &= \mathrm{diag}(-\frac{2\mathbf{a} \odot (\mathbf{a} - \mathbf{b}\mathrm{i}) \odot \hat{\mathbf{Y}}}{(\mathbf{a}^2 + \mathbf{b}^2 + \lambda)^2} + \frac{\hat{\mathbf{Y}}}{\mathbf{a}^2 + \mathbf{b}^2 + \lambda}), \\ \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{b}} &= \mathrm{diag}(-\frac{2\mathbf{b} \odot (\mathbf{a} - \mathbf{b}\mathrm{i}) \odot \hat{\mathbf{Y}}}{(\mathbf{a}^2 + \mathbf{b}^2 + \lambda)^2} - \frac{\hat{\mathbf{Y}}\mathrm{i}}{\mathbf{a}^2 + \mathbf{b}^2 + \lambda}), \end{aligned} \tag{19}$$

afterwards,

$$\begin{aligned} \frac{\partial \hat{\mathbf{R}}}{\partial \hat{\mathbf{X}}} &= (\frac{\partial \mathbf{a}}{\partial \hat{\mathbf{X}}})^{\mathrm{T}} \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{a}} + (\frac{\partial \mathbf{b}}{\partial \hat{\mathbf{X}}})^{\mathrm{T}} \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{b}} \\ &= \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{a}} - \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{b}}\mathrm{i} \\ &= -2\mathrm{diag}(\frac{(\mathbf{a} - \mathbf{b}\mathrm{i}) \odot (\mathbf{a} - \mathbf{b}\mathrm{i}) \odot \hat{\mathbf{Y}}}{(\mathbf{a}^2 + \mathbf{b}^2 + \lambda)^2}) \\ &= -2\mathrm{diag}(\frac{(\hat{\mathbf{X}}^{\star})^2 \odot \hat{\mathbf{Y}}}{(\hat{\mathbf{X}}^{\star} \odot \hat{\mathbf{X}} + \lambda)^2}). \end{aligned} \tag{20}$$

Thus, $\nabla_{\mathbf{X}} L$ can be achieved by Eq. 16 and Eq. 20. By setting $\mathbf{U} = (\hat{\mathbf{X}}^{\star} \odot \hat{\mathbf{X}} + \lambda)^{-1}$ which keeps the same algebra in the following text, Eq. 16 can be simplified as follows,

$$\nabla_{\mathbf{X}} L = \mathbf{F}(-2\mathbf{U}^2 \odot (\hat{\mathbf{X}}^{\star})^2 \odot \hat{\mathbf{Y}})^{\mathrm{T}} \mathbf{F}^{\mathrm{H}} \nabla_{\mathbf{R}} L. \tag{21}$$

For $\nabla_{\mathbf{Y}} L$, we can write it as

$$\begin{aligned} \nabla_{\mathbf{Y}} L &= (\frac{\partial \mathbf{R}}{\partial \mathbf{Y}})^{\mathrm{T}} \nabla_{\mathbf{R}} L \\ &= ((\mathrm{C}(\mathbf{X})^{\mathrm{H}} \mathrm{C}(\mathbf{X}) + \lambda \mathbf{I})^{-1} \mathrm{C}(\mathbf{X})^{\mathrm{H}})^{\mathrm{T}} \nabla_{\mathbf{R}} L. \end{aligned} \tag{22}$$

Similarly, we can get

$$\begin{aligned} \hat{\nabla}_{\mathbf{Y}} L &= \mathbf{U} \odot \hat{\mathbf{X}}^{\star} \odot \hat{\nabla}_{\mathbf{R}} L \\ \nabla_{\mathbf{Y}} L &= \mathscr{F}^{-1}(\hat{\nabla}_{\mathbf{Y}} L). \end{aligned} \tag{23}$$

For $\nabla_{\lambda} L$, we can write it as

$$\nabla_{\lambda} L = (\frac{\partial \mathbf{R}}{\partial \lambda})^{\mathrm{T}} \nabla_{\mathbf{R}} L. \tag{24}$$

After derivation of $\lambda$ of both side of Eq. 17 and simple algebra, we have

$$\begin{aligned} \mathbf{R} + (\mathrm{C}(\mathbf{X})^{\mathrm{H}} \mathrm{C}(\mathbf{X}) + \lambda \mathbf{I}) \frac{\partial \mathbf{R}}{\partial \lambda} &= \mathbf{0} \\ \frac{\partial \mathbf{R}}{\partial \lambda} &= -(\mathrm{C}(\mathbf{X})^{\mathrm{H}} \mathrm{C}(\mathbf{X}) + \lambda \mathbf{I})^{-1} \mathbf{R} \\ \frac{\partial \mathbf{R}}{\partial \lambda} &= -(\mathrm{C}(\mathbf{X})^{\mathrm{H}} \mathrm{C}(\mathbf{X}) + \lambda \mathbf{I})^{-2} \mathrm{C}(\mathbf{X})^{\mathrm{H}} \mathbf{Y}. \end{aligned} \tag{25}$$

Similarly, we can get

$$\begin{aligned}\hat{\nabla}_\lambda \mathbf{R} &= -\mathbf{U}^2 \odot \hat{\mathbf{X}}^\star \odot \hat{\mathbf{Y}} \\ \nabla_\lambda L &= \mathscr{F}^{-1}(\hat{\nabla}_\lambda \mathbf{R})^{\mathrm{T}} \nabla_{\mathbf{R}} L \\ \\ \frac{\partial \hat{\mathbf{R}}}{\partial \lambda} &= -\mathbf{U}^2 \odot \hat{\mathbf{X}}^\star \odot \hat{\mathbf{Y}} \\ \nabla_\lambda L &= \mathscr{F}^{-1}(\frac{\partial \hat{\mathbf{R}}}{\partial \lambda})^{\mathrm{T}} \nabla_{\mathbf{R}} L.\end{aligned} \tag{26}$$

**Gradients of Elefusion layer**
Considering Eq. (1), during back propagation, given $\nabla_{\mathbf{S}} L$, we aim to calculate $\nabla_{\mathbf{S}^l} L$ and $\nabla_{\mathbf{\Upsilon}^l} L$.
For $\nabla_{\mathbf{S}^l} L$, we have

$$\nabla_{\mathbf{S}^l} L = (\frac{\partial \mathbf{S}}{\partial \mathbf{S}^l})^{\mathrm{T}} \nabla_{\mathbf{S}} L = \mathbf{\Upsilon}^l \odot \nabla_{\mathbf{S}} L. \tag{27}$$

For $\nabla_{\mathbf{\Upsilon}^l} L$, we have

$$\nabla_{\mathbf{\Upsilon}^l} L = (\frac{\partial \mathbf{S}}{\partial \mathbf{\Upsilon}^l})^{\mathrm{T}} \nabla_{\mathbf{S}} L = \mathbf{S}^l \odot \nabla_{\mathbf{S}} L. \tag{28}$$

**Gradients of Crop layer** For Crop layer, during back propagation, we know $\nabla_{\mathbf{Z}_t} L$, $\nabla_{\mathbf{O}} L$, $\nabla_{\mathbf{G}} L$ and aims to calculate $\nabla_{\mathbf{S}} L$. Here, we use the similar solution of pooling operation that puts the value of $\nabla_{\mathbf{Z}_t} L$, $\nabla_{\mathbf{O}} L$, $\nabla_{\mathbf{G}} L$ directly to corresponding locations in $\mathbf{S}$.

### 1.3. Training details

Our dynamic Siamese network is actually a recurrent neural network (RNN) that produces an output at each time step and has recurrent connection from output to the input. It is difficult to train such network with a long video which needs huge storage space during training to store the gradients of each parameter at each time step. Thus, we use short videos that only have 10 frames as training data. Besides, we use the Matconvnet [4] as the training platform which is designed to train convolution neural network (CNN). To make Matconvnet suitable for training DSiam with the short videos, we unfold the RNN for 10 frames. Thus, each frame corresponds to a DSiam network; all DSiam networks share their parameters. In summary, we construct an unfold network that contains 940 layers with only 33 filters that need to be learnt.

We construct the short video dataset based on ILSVRC-2015 video dataset [3] which contains 4417 videos. We first select 1130 videos from ILSVRC-2015 by removing the video whose target has large size ratio and occupy the most of a frame. Then, we randomly generate 1000 short videos from the 1130 videos. For each short video, i.e. $\{\mathbf{I}_t | t = 1, ..., 10\}$, we know ground truth at each frame, i.e. $\{\mathbf{b}_t | t = 1, ..., 10\}$ with $\mathbf{b}_t$ being the bounding box indicating the location of target. During training, given $\mathbf{b}_1$, we first perform a forward process and get 10 response maps $\{\mathbf{S}_t | t = 1, ..., 10\}$. Then, we construct ground truth of these response maps according to $\{\mathbf{b}_t | t = 1, ..., 10\}$ and get $\{\mathbf{J}_t | t = 1, ..., 10\}$. $\mathbf{J}_t$ have the same size with $\mathbf{S}_t$ and indicates the true target location in $\mathbf{S}_t$ with 1 being target and -1 being background. Then, $\{\mathbf{S}_t | t = 1, ..., 10\}$ and $\{\mathbf{J}_t | t = 1, ..., 10\}$ are used to generate Loss at each frame and to activate the backward process. We set batch size as 1, i.e. a short video, and learning rate being $10^{-7}$ to $10^{-9}$, weight decay $0.0005$ and momentum $0.9$.

## 2. Experimental Results

### 2.1. Validation of joint training

We evaluate our DSiamM with and without joint training on OTB-2013 dataset. As shown in Fig. 2, with joint training, DSiamM get better tracking performance on both overlap and location error metrics. Specifically, As shown in Fig. 3, the first example shows that DSiamM with joint training is able to locate target with tighter bounding box; the second case shows that DSiamM with joint training captures the motorcycle while the DSiamM without jointing training fails, which shows that joint training helps to adapt target appearance change better. Since we make an initial exploration to train such network with video dataset, we believe that we could achieve better performance by carefully selecting train dataset and using more effective training strategy.

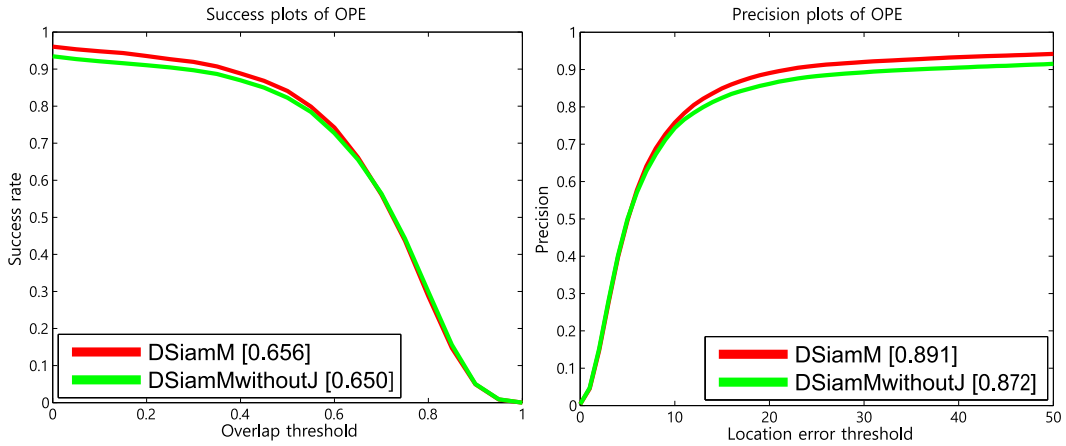


Figure 2. The success plot and precious plot of DSiamM with joint training (DSiamM) and DSiamM without joint training (DSiamMwithoutJ) on OTB-2013 dataset. The numbers in the legend indicate the representative precisions at 20 pixels for precision plots, and the area under curve scores for success plots.

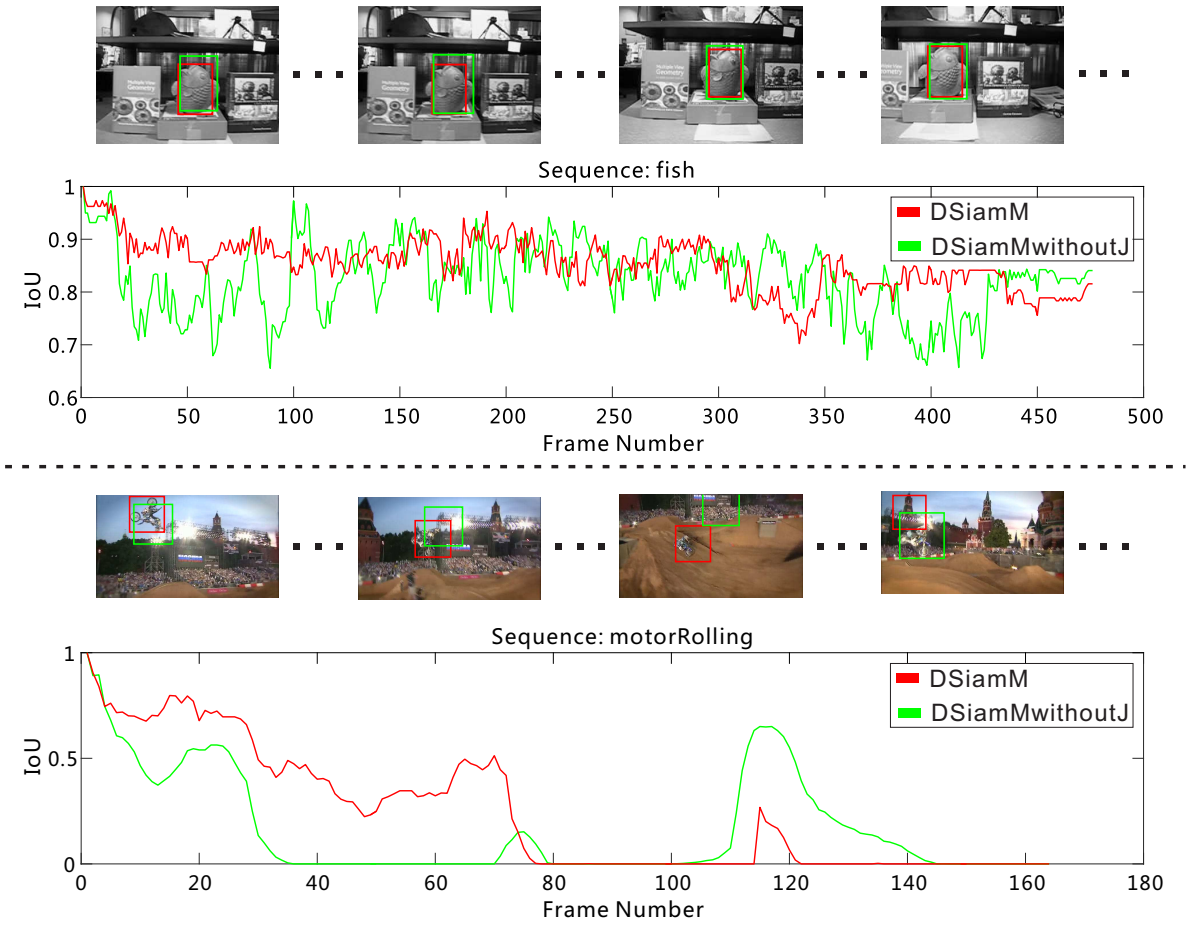


Figure 3. Two examples of DSiamM with joint training (DSiamM) and DSiamM without joint training (DSiamMwithourJ). DSiamM get more accurate target location than DSiamMwithoutJ in the 'fish' sequence; DSiamM captures the motorcycle while DSiamMwithoutJ misses the target at the beginning in the 'motorRolling' sequence.

## 2.2. Validation of V and W

In the submitted manuscript, we have shown that tracker based on DSiam network achieves better performance by using target appearance variation transformation, i.e. $\mathbf{V}$, and background suppression transformation, $\mathbf{W}$. Here, we further validate

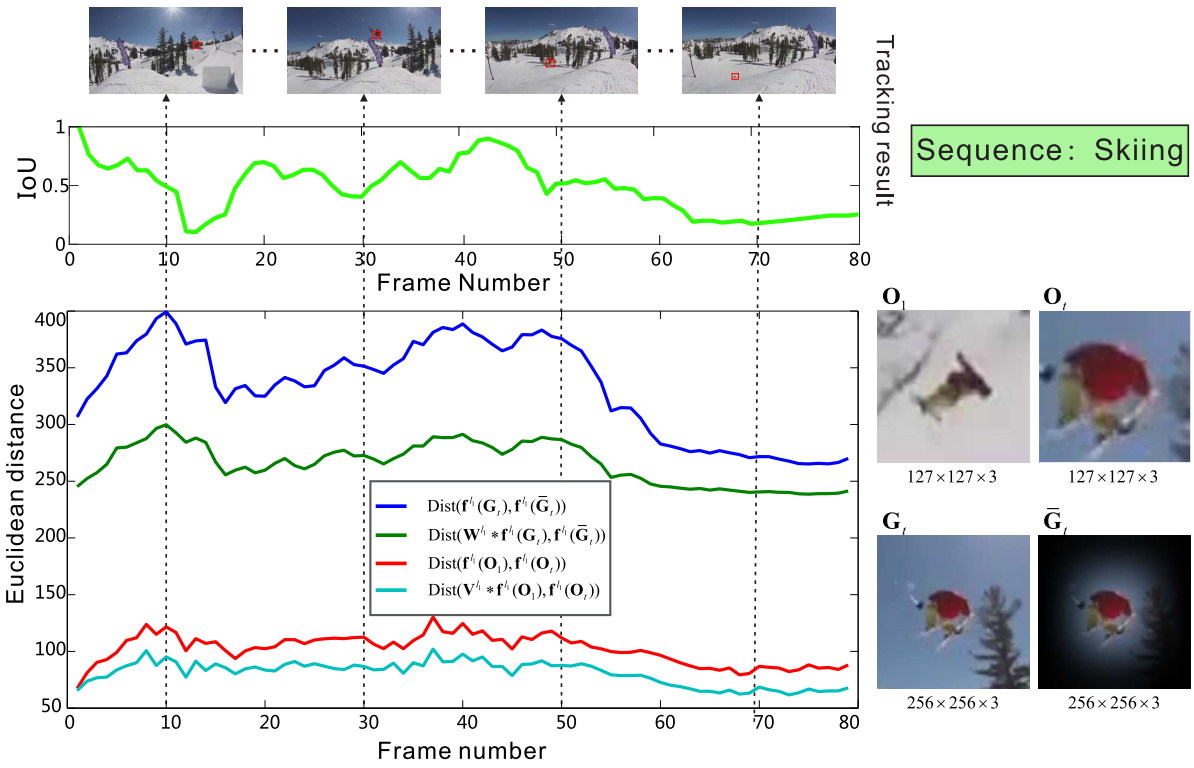


Figure 4. Validation of target appearance variation transformation, i.e. $\mathbf{V}$, and background suppression transformation, i.e. $\mathbf{W}$, according the Euclidean distance between deep features. We take the 'Skiing' sequence in OTB dataset as an example. Left subfigure shows the tracking results and Euclidean distance of four cases during tracking, which demonstrates that two transformations help to adapt to target appearance changes and suppress background interference. Right subfigure shows the $\mathbf{O}_1$, $\mathbf{O}_t$, $\mathbf{G}_t$ and $\bar{\mathbf{G}}_t$, when $t = 25$.

the function of $\mathbf{V}$ and $\mathbf{W}$ by comparing the similarity changes of deep features after transformation. Specifically, we take a challenge video as an example, i.e. 'Skiing' sequence in OTB dataset, in which the target changes significantly. During tracking, we calculate the similarity between the deep features of $\mathbf{O}_1$ and $\mathbf{O}_t$ via Euclidean distance, i.e. $\mathrm{Dist}(\mathrm{f}^{l_1}(\mathbf{O}_1), \mathrm{f}^{l_1}(\mathbf{O}_t))$, where $\mathrm{Dist}(\cdot)$ denotes the Euclidean distance. Meanwhile, we also calculate the similarity after performing $\mathbf{V}$ transformation, i.e. $\mathrm{Dist}(\mathbf{V}^{l_1} * \mathrm{f}^{l_1}(\mathbf{O}_1), \mathrm{f}^{l_1}(\mathbf{O}_t))$. As shown in Fig. 4, with target appearance variation transformation, the transformed deep feature $(\mathbf{V}^{l_1} * \mathrm{f}^{l_1}(\mathbf{O}_1))$ is more similar to the deep feature of target at $t$th frame $(\mathrm{f}^{l_1}(\mathbf{O}_t))$ than $\mathrm{f}^{l_1}(\mathbf{O}_1)$ during the whole tracking process, which makes our tracker be able to adapt the target changes and capture target even through it changes significantly. Similarity, for background suppression transformation, as shown in Fig. 4, the transformed deep feature $(\mathbf{W}^{l_1} * \mathrm{f}^{l_1}(\mathbf{G}_t))$ is more similar to the deep feature of $\bar{\mathbf{G}}_t$ than $\mathrm{f}^{l_1}(\mathbf{G}_t)$, which demonstrates that background suppression transformation does help to suppress the background interference and makes our tracker to get better tracking accuracy, as shown in the discussion part of submitted manuscript.

## 2.3. Challenge cases

We add 4 challenge videos collected from website to show that our tracker is able to track target even through the targets are surrounded by cluttered background or change significantly. We compare DSiamM with SiamFC [1] and HCF [2]. In the sequences of 'Dancer' and 'Player', SiamFC misses the target and track the object being similar with the target, since it does not consider the target appearance variation and background interference. HCF captures the target without adapting to the scale changes and runs much slower than SiamFC and DSiamM. DSiamM achieves better tracking accuracy than other two methods with target variation and background suppression transformation. In the sequences of 'Fighting' and 'Forest Gump', the target changes significantly, i.e. the target at following frame is entirely different from the template given at the first frame. SiamFC and HCF fail to capture targets. In contrast, DSiamM still tracks the target in these challenge situations. We have shown the whole tracking process of the four cases with '*.avi' files in the attached supplementary.


Figure 5. Two challenge videos in which target is surrounded by clustered background and similar objects.

## References

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *arXiv preprint arXiv:1606.09549*, 2016. 1, 7

[2] C. Ma, J. B. Huang, X. Yang, and M. H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 7

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and


Figure 6. Two challenge videos in which target changes significantly during tracking.

L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 5

[4] A. Vedaldi and K. Lenc. Matconvnet c convolutional neural networks for matlab. In *ACM MM*, 2015. 5