Real-Time Object Tracking via Meta-Learning: Efficient Model Adaptation and One-Shot Channel Pruning

Ilchae Jung,[1,2] Kihyun You,[1] Hyeonwoo Noh,[1,2] Minsu Cho,[1] Bohyung Han[2]
[1]Computer Vision Lab., POSTECH, Korea
[2]Computer Vision Lab. ECE, & ASRI, Seoul National University, Korea
{chey0313, kihyun13, shgusdngogo, mscho}@postech.ac.kr, bhhan@snu.ac.kr

Abstract

We propose a novel meta-learning framework for real-time object tracking with efficient model adaptation and channel pruning. Given an object tracker, our framework learns to fine-tune its model parameters in only a few gradient-descent iterations during tracking while pruning its network channels using the target ground-truth at the first frame. Such a learning problem is formulated as a meta-learning task, where a meta-tracker is trained by updating its meta-parameters for initial weights, learning rates, and pruning masks through carefully designed tracking simulations. The integrated meta-tracker greatly improves tracking performance by accelerating the convergence of online learning and reducing the cost of feature computation. Experimental evaluation on the standard datasets demonstrates its outstanding accuracy and speed compared to the state-of-the-art methods.

1 Introduction

Recent advances in deep neural networks have drastically improved visual object tracking technology. By learning strong representations of target and background using convolutional neural networks, many algorithms (Nam and Han 2016; Nam, Baek, and Han 2016; Han, Sim, and Adam 2017; Danelljan et al. 2016; 2017a) have achieved a significant performance gain. Based on the success, recent methods (Danelljan et al. 2019; Li et al. 2019; Huang, Lucey, and Ramanan 2017; Fan and Ling 2017; Zhu et al. 2018; Jung et al. 2018) have further advanced network architectures and training techniques for better accuracy and speed. Such manual design choices, however, often result a suboptimal solution in a limited exploration space eventually.

Meta-learning (Santoro et al. 2016; Andrychowicz et al. 2016; Finn, Abbeel, and Levine 2017) automates the optimization procedure of a learning problem through the evaluation over a large number of episodes. It facilitates exploring a large hypothesis space and fitting a learning algorithm for a particular set of tasks. Considering that an object tracking algorithm aims to learn a parameterized target appearance model specialized for individual video sequences, it is natural to adopt meta-learning for the effective optimization in tracking algorithms. In the context of object tracking based on deep neural networks, an episode corresponds to a realization of object tracking based on a parameterized model for a video sequence. The execution of an episode is computationally expensive and time-consuming, which makes it difficult to incorporate meta-learning into object tracking because it requires optimization over a large number of learning episodes. Meta-Tracker (Park and Berg 2018) circumvents this issue by simulating a tracking episode within a single frame. However, in this approach, meta-learning is limited to initializing the deep neural network for target appearance modeling at the first frame, where ground-truth target annotation is available. This is because Meta-Tracker relies only on the accurate ground-truth for meta-learning, although tracking after the first frame involves model adaptation based on estimated targets from the previous frames.

We introduce a novel meta-learning framework for object tracking, which focuses on fast model adaptation. In particular, our approach simulates the model adaptation by dividing it into the following two distinct cases: (1) initial adaptation, where the model is optimized for the one-shot target ground-truth at the first frame and (2) online adaptation, where the model is updated using the tracked targets in previous frames. Such a fine-grained simulation enables to capture distinct properties of two cases and allows the meta-learned model to generalize better. In our meta-learning for object tracking, we evaluate the learning algorithm in terms of its expected accuracy over various situations by simulating diverse challenging scenarios with hard examples. Moreover, we develop a one-shot channel pruning technique via meta-learning based on a single ground-truth target annotation at the first frame. Our main contributions are threefold:

- We propose a meta-learning framework for object tracking, which efficiently optimizes network parameters for target appearance through sophisticated simulation and test of tracking episodes in meta-training.
- We introduce a one-shot network pruning technique via meta-learning, which enables to learn target-specific model compression based on only a single ground-truth annotation available at the first frame.
- We demonstrate that our meta-tracker leads to significant

speed-up and competitive accuracy on the standard benchmark datasets.

2 Related Work

Most of the state-of-the-art visual tracking algorithms rely on the representations learned from deep neural networks, and often formulate object tracking as a classification (Nam and Han 2016; Jung et al. 2018) or a metric learning problem (Bertinetto et al. 2016; Zhu et al. 2018). While several recent algorithms show strong representation as well as computational efficiency, they still suffer from target appearance variations in challenging situations during tracking.

A pretraining stage is commonly adopted for a tracker to learn discriminative features for object tracking. MDNet and its variants (Nam and Han 2016; Jung et al. 2018) employ multi-domain learning to simulate various tracking scenarios and learn generic target representations. Meanwhile, (Bertinetto et al. 2016; Valmadre et al. 2017; Zhu et al. 2018) discuss representation learning based on correlation filters. However, these methods mainly focus on learning representations for target appearances, but the efficient optimization of model update procedure has been rarely explored.

Meta-learning is a framework to learn a learning algorithm under a certain distribution (Thrun and Pratt 1998; Hochreiter, Younger, and Conwell 2001). It has been explored to expedite learning procedures in few-shot classification (Santoro et al. 2016; Andrychowicz et al. 2016; Finn, Abbeel, and Levine 2017; Li et al. 2017), reinforcement learning (Finn, Abbeel, and Levine 2017; Al-Shedivat et al. 2018) and imitation learning (Duan et al. 2017). Meta-Tracker (Park and Berg 2018) adopts this idea for object tracking to demonstrate potential benefit for fast model updates. However, it applies meta-learning only to the adaptation at the first frame of an input video, and fails to show the practical efficiency of tracking algorithms through fast online model updates. Moreover, it relies on various heuristics to establish meta-learning, *e.g.,* learning rate adjustments, layer selections for parameter updates, and label shuffling.

Model compression is a useful technique to reduce the size of a deep neural network and accelerate its inference time (Han et al. 2015; Han, Mao, and Dally 2016). A popular approach to model compression is channel pruning (Wen et al. 2016; He, Zhang, and Sun 2017), which aims to remove a subset of channels in each layer based on their usefulness. Channel pruning often involves a complex optimization task or time-consuming validation and fine-tuning, which is not affordable in real-time object tracking. To tackle the limitation, (Choi et al. 2018) employ multiple expert autoencoders to compress deep features for object tracking, where the targets are divided into several coarse categories and then allocated to autoencoders for feature compression according to their class labels. However, the predefined categories limit the generality of the tracker and the multiple autoencoders are computationally expensive to learn.

Improving the efficiency of tracking algorithms by fast model adaptation and compression via meta-learning has not been explored. The proposed meta-learning approach resembles (Finn, Abbeel, and Levine 2017; Li et al. 2017) in the sense that we learn initial parameters and learning rates through meta-learning while we propose a novel method to estimate the distribution of the target task for stable meta-learning. Contrary to (Park and Berg 2018), our algorithm performs more sophisticated parameter estimation without using the complex heuristics for training. Also, our channel pruning is based on a one-shot ground-truth available at the first frame, and is realized within a meta-learning framework for efficient optimization.

3 Meta-learning for Fast Adaptation

This section presents an iterative framework for optimizing hyperparameters via meta-learning for fast model adaptation during object tracking. Building on MAML (Finn, Abbeel, and Levine 2017), our meta-learning framework is adapted for object tracking and thus consists of a series of tracking episode simulation, model test, and meta-parameter optimization.

Objective

The objective of model adaptation is to minimize the standard cross-entropy loss of an output $\mathbf{f}(\mathbf{x};\boldsymbol{\theta}) \in \mathbb{R}^2$ from a model parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$, given an input patch $\mathbf{x}$ with its binary label $\mathbf{y} \in \{[1,0]^\top, [0,1]^\top\}$, where $[1,0]^\top$ denotes the positive label while $[0,1]^\top$ is the negative one. Then, the loss function is formally given by

$$\mathcal{L}(\mathcal{D};\boldsymbol{\theta}) = -\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x},\mathbf{y})}\Big[\mathbf{y}^\top \log\big[\mathrm{Softmax}\big(\mathbf{f}(\mathbf{x};\boldsymbol{\theta})\big)\big]\Big], \quad (1)$$

where $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ is a dataset for learning target appearance, which is collected from tracking results in previously frames, and $p_{\mathcal{D}}(\mathbf{x},\mathbf{y})$ is the distribution of data in $\mathcal{D}$. The softmax activation function $\mathrm{Softmax}(\cdot)$ normalizes the output into a probability-like vector.

Tracking Simulation

Since the hyperparameters are meta-learned based on tracking simulation, the construction of realistic simulations at training time is crucial to the generalization of the meta-learned algorithms. The typical object tracking algorithms consider two types of model adaptation (Nam and Han 2016; Jung et al. 2018). One is performed at the 1^{st} frame using the ground-truth annotation while the other is at subsequent frames using estimated targets in previous frames. We call the former *initial adaptation* and the latter *online adaptation*. The major difference between the initial adaptation and the online adaptation stems from the availability of ground-truth labels in learning. We consider this aspect in the tracking simulation and use a different dataset for each adaptation during meta-learning.

Let Γ be a training video set and $\mathcal{V} \in \Gamma$ is an annotated video with tracking ground-truths for a single moving target. Specifically, an annotated video $\mathcal{V}$ is a sequence of tuples consisting of a frame and a ground-truth target annotation for the frame. We define the *tracking simulation* as a sequence of initial adaptation, online adaptation, and test dataset construction in meta-training for a target annotated in $\mathcal{V}$. Unlike the standard object tracking, which aims to estimate targets in every frame, tracking simulation performs

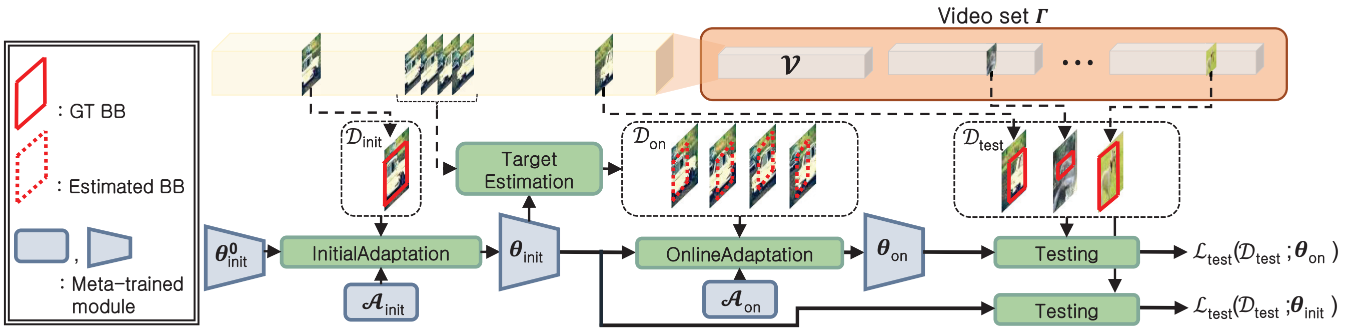
Video set Γ
$\mathcal{V}$
: GT BB
: Estimated BB
: Meta-trained module
$\mathcal{D}_{\text{init}}$
$\mathcal{D}_{\text{on}}$
$\mathcal{D}_{\text{test}}$
Target Estimation
θ^0_{init}
InitialAdaptation
θ_{init}
OnlineAdaptation
θ_{on}
Testing
Testing
$\mathcal{A}_{\text{init}}$
$\mathcal{A}_{\text{on}}$
$\mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}};\theta_{\text{on}})$
$\mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}};\theta_{\text{init}})$

Figure 1: A simulated tracking episode. We meta-learn a fast model adaptation algorithm based on simulated episodes. The model is adapted with ground-truth target $\mathcal{D}_{\text{init}}$ in the initial adaptation and with estimated target $\mathcal{D}_{\text{on}}$ in the online adaptation. The meta-parameters $(\boldsymbol{\theta}^0_{\text{init}}, \mathcal{A}_{\text{init}}, \mathcal{A}_{\text{on}})$ are learned to minimize test loss $\mathcal{L}_{\text{test}}$ in meta-training over multiple simulated episodes. $\mathcal{L}_{\text{test}}$ is defined using $\mathcal{D}_{\text{test}}$ containing both ground-truth from upcoming frames and hard examples from other annotated videos. The hard instances are used to simulate diverse clutter appearing in the standard object tracking.

only a single initial adaptation and a single online adaptation on the datasets that are artificially generated from $\mathcal{V}$. Following the convention in meta-learning literatures (Santoro et al. 2016; Finn, Abbeel, and Levine 2017), we call the tracking simulation as *an episode* or *a simulated tracking episode*.

The first step of a simulated tracking episode is the initial adaptation; the model parameter is learned for an initial dataset $\mathcal{D}_{\text{init}}$ by optimizing Eq. (1), where $\mathcal{D}_{\text{init}}$ is collected based on the target ground-truth annotation in a frame sampled from $\mathcal{V}$. The updated model parameter is then employed to estimate the target states in the unseen frames sampled from $\mathcal{V}$. The estimated targets are used to construct an online dataset $\mathcal{D}_{\text{on}}$ for online model adaptation. The online adaptation simulates the model adaptation using noisy labels, which commonly happens in real object tracking scenarios. Note that the optimization of Eq. (1) during the initial and online adaptations is guided by the hyperparameters that we learn in the meta-learning. The final step is to collect a test dataset $\mathcal{D}_{\text{test}}$ for meta-training to define a test loss $\mathcal{L}_{\text{test}}$ and a meta-optimization loss $\mathcal{L}_{\text{meta}}$, which are the objective functions of our meta-learning. Note that $\mathcal{D}_{\text{test}}$ is obtained from the ground-truth annotations in $\mathcal{V}$ and augmented with the negative samples collected from all other annotated videos in the training video set $\Gamma - \{\mathcal{V}\}$. Figure 1 illustrates the overall pipeline in a simulated tracking episode.

A simulated tracking episode results in a set of the collected datasets and a series of intermediate model parameters associated with the episode. Such information—datasets and parameters—is referred to as *an episode trajectory* and denoted by τ[1]. The objective function for meta-learning is defined over the trajectories from multiple episodes, where each episode is based on an annotated video $\mathcal{V} \in \Gamma$. We optimize the hyperparameters based on this objective function. Algorithm 1 describes the meta-learning procedure with the proposed simulated tracking episodes.

Algorithm 1 Meta-Learning for Fast Adaptation

```
1:  Input: A Training video set $\Gamma$,
    Meta-parameters $\mathcal{M} = \{\boldsymbol{\theta}^0_{\text{init}}, \mathcal{A}_{\text{init}}, \mathcal{A}_{\text{on}}\}$,
    Output: Learned meta-parameters $\mathcal{M}^*$

2:  while not converged do
3:      Sample a mini-batch of annotated videos from $\Gamma$
4:      for all video $\mathcal{V}$ in a mini-batch do
5:          Collect $\mathcal{D}_{\text{init}}$ based on $\mathcal{V}$
6:          for all $k$ in $1, ..., K_{\text{init}}$ do  // Eq. (2)
7:              $\boldsymbol{\theta}^k_{\text{init}} = \boldsymbol{\theta}^{k-1}_{\text{init}} - \boldsymbol{\alpha}^k_{\text{init}} \odot \nabla_{\boldsymbol{\theta}^{k-1}_{\text{init}}} \mathcal{L}(\mathcal{D}_{\text{init}}; \boldsymbol{\theta}^{k-1}_{\text{init}})$
            end for
8:          Collect $\mathcal{D}_{\text{on}}$ based on $\mathcal{V}$ and $\boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}$
9:          for all $k \in 1, ..., K_{\text{on}}$ do  // Eq. (3)
10:             $\boldsymbol{\theta}^k_{\text{on}} = \boldsymbol{\theta}^{k-1}_{\text{on}} - \boldsymbol{\alpha}^k_{\text{on}} \odot \nabla_{\boldsymbol{\theta}^{k-1}_{\text{on}}} \mathcal{L}(\mathcal{D}_{\text{on}}; \boldsymbol{\theta}^{k-1}_{\text{on}})$
            end for  // where $\boldsymbol{\theta}^0_{\text{on}} = \boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}$
11:         Collect $\mathcal{D}_{\text{test}}$ based on $\mathcal{V}$ and $\Gamma - \{\mathcal{V}\}$
12:         Set $\tau = (\mathcal{D}_{\text{init}}, \mathcal{D}_{\text{on}}, \mathcal{D}_{\text{test}}, \{\boldsymbol{\theta}^k_{\text{init}}\}^{K_{\text{init}}}_{k=1}, \{\boldsymbol{\theta}^k_{\text{on}}\}^{K_{\text{on}}}_{k=1})$
        end for
13:     Compute $\mathcal{L}_{\text{meta}}$ over a mini-batch of $\tau$ // Eq. (7)
14:     $\mathcal{M} \leftarrow \text{optimize}(\mathcal{M}, \nabla_{\mathcal{M}} \mathcal{L}_{\text{meta}})$  // Eq. (8)
    end while
```

Meta-parameters The meta-parameter $\mathcal{M}$ is a set of hyperparameters that are optimized by our meta-learning approach. The primary objective of meta-learning is to reduce computational cost and maintain accuracy, and $\mathcal{M}$ allows to find a better trade-off between computation and performance by making a sophisticated control of a gradient-based model adaptation procedure. We regard the initial model parameter $\boldsymbol{\theta}^0_{\text{init}} \in \mathbb{R}^d$ as a hyperparameter. In addition, we include learning rates as hyperparameters by extending a conventional scalar learning rate α to a per-parameter and per-iteration learning rate vector for the initial adaptation $\mathcal{A}_{\text{init}} = \{\boldsymbol{\alpha}^k_{\text{init}} \in \mathbb{R}^d\}^{K_{\text{init}}}_{k=1}$ and the online adaptation $\mathcal{A}_{\text{on}} = \{\boldsymbol{\alpha}^k_{\text{on}} \in \mathbb{R}^d\}^{K_{\text{on}}}_{k=1}$, where K_{init} and K_{on} are the number of gradient-descent iterations for the initial and online adaptations, respectively. The meta-parameter is a collection of all the hyperparameters, $\mathcal{M} = \{\boldsymbol{\theta}^0_{\text{init}}, \mathcal{A}_{\text{init}}, \mathcal{A}_{\text{on}}\}$.

[1]The formal definition of τ can be found in "Meta-parameter optimization" subsection in page 4.

Initial adaptation The initial adaptation uses the initial dataset $\mathcal{D}_{\text{init}}$, which is constructed based on a frame and its ground-truth annotation that are uniformly sampled from an annotated video $\mathcal{V}$; $\mathcal{D}_{\text{init}}$ contains positive samples that are tightly overlapped with the ground-truth annotation and negative samples that are loosely overlapped or separated from the annotation. The procedure for sampling the positive and negative examples is identical to the strategy of dataset collection for target appearance learning in the standard object tracking (Jung et al. 2018).

We adapt the initial model parameter $\boldsymbol{\theta}^0_{\text{init}}$ by the stochastic gradient-descent method for K_{init} iterations using Eq. (1), which is identical to real tracking scenarios. As a result of the model update, we obtain a target appearance model learned from a ground-truth annotation in a single frame. An iteration of the initial adaptation is given by

$$\boldsymbol{\theta}^k_{\text{init}} = \boldsymbol{\theta}^{k-1}_{\text{init}} - \boldsymbol{\alpha}^k_{\text{init}} \odot \nabla_{\boldsymbol{\theta}^{k-1}_{\text{init}}} \mathcal{L}(\mathcal{D}_{\text{init}}; \boldsymbol{\theta}^{k-1}_{\text{init}}), \qquad (2)$$

where $k = 1, \ldots, K_{\text{init}}$. The model parameter after the initial adaptation is given by $\boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}$.

Online Adaptation The online adaptation uses the online dataset $\mathcal{D}_{\text{on}}$, which includes the examples sampled from the estimated targets. The model parameterized by $\boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}$ is adopted to estimate the target states in the sampled frames by scoring candidate image patches. Similarly to the standard object tracking, the candidate image patches are sampled from a local area centered at the ground-truth target, and an image patch with a maximum score in each frame is selected as an estimated target. The dataset $\mathcal{D}_{\text{on}}$ contains positive and negative samples collected based on the estimated targets; the data collection strategy is identical to the one for $\mathcal{D}_{\text{init}}$, except that the estimated targets are used. Note that using the estimated targets for online adaptation turns out to be effective in handling the potential issues given by unreliable target estimation during tracking.

For online adaptation, we update $\boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}$ by stochastic gradient descent for K_{on} iterations based on Eq. (1). The formal description of the online adaptation is given by

$$\boldsymbol{\theta}^k_{\text{on}} = \boldsymbol{\theta}^{k-1}_{\text{on}} - \boldsymbol{\alpha}^k_{\text{on}} \odot \nabla_{\boldsymbol{\theta}^{k-1}_{\text{on}}} \mathcal{L}(\mathcal{D}_{\text{on}}; \boldsymbol{\theta}^{k-1}_{\text{on}}), \qquad (3)$$

where $\boldsymbol{\theta}^0_{\text{on}} = \boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}$ and $k = 1, \ldots, K_{\text{on}}$. The model parameter obtained from the online adaptation is denoted by $\boldsymbol{\theta}^{K_{\text{on}}}_{\text{on}}$.

Hard example mining for meta-training

Meta-learning aims to learn the meta-parameters maximizing the expected tracking accuracy of adapted models based on diverse tracking scenarios. In meta-training, we thus evaluate a trained model by testing its tracking performance. As explored in Meta-Tracker (Park and Berg 2018), the most straightforward choice for evaluating the model parameter is to measure its accuracy based on the ground-truth target annotation in the subsequent frames. However, this approach might not be sufficient for evaluating the robustness of the model because the dataset typically has limited diversity. Hence, we propose to use hard example mining to evaluate the model during testing in meta-training.

Hard example dataset The test dataset in meta-training is a union of two datasets as $\mathcal{D}_{\text{test}} = \mathcal{D}^{\text{std}}_{\text{test}} \cup \mathcal{D}^{\text{hard}}_{\text{test}}$, where $\mathcal{D}^{\text{std}}_{\text{test}}$ is obtained from $\mathcal{V}$ and $\mathcal{D}^{\text{hard}}_{\text{test}}$ is collected from $\Gamma - \{\mathcal{V}\}$. $\mathcal{D}^{\text{std}}_{\text{test}}$ consists of both positive and negative samples for a target in $\mathcal{V}$ and the strategy for collecting $\mathcal{D}^{\text{std}}_{\text{test}}$ is identical to the one for collecting $\mathcal{D}_{\text{init}}$, including the use of the ground-truth annotation. $\mathcal{D}^{\text{std}}_{\text{test}}$ is essential for evaluating learned models, but it may not be sufficient to handle diverse distracting objects. To consider such objects, we introduce $\mathcal{D}^{\text{hard}}_{\text{test}}$, which consists of the image patches obtained from the ground-truth annotations sampled from $\Gamma - \{\mathcal{V}\}$. We label all examples in $\mathcal{D}^{\text{hard}}_{\text{test}}$ as negative. Since these negative samples are actually target instances in the other videos, they are hard examples.

Test objective in meta-training To test the model with the adapted model parameters $\boldsymbol{\theta}$ in meta-training, we compute a test loss on the test dataset $\mathcal{D}_{\text{test}}$, which is given by a combination of the cross-entropy loss in Eq. (1) and the triplet loss defined below:

$$\mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}}; \boldsymbol{\theta}) = \mathcal{L}(\mathcal{D}^{\text{std}}_{\text{test}}; \boldsymbol{\theta}) + \gamma \mathcal{L}_{\text{tri}}(\mathcal{D}_{\text{test}}; \boldsymbol{\theta}) \qquad (4)$$

where γ is a hyperparameter controlling the relative importance of the two terms. The triplet loss is defined by

$$\mathcal{L}_{\text{tri}}(\mathcal{D}_{\text{test}}; \boldsymbol{\theta}) = \qquad (5)$$
$$\mathbb{E}_{p_{\mathcal{D}_{\text{test}}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)} \Big[\big(\xi + \Delta(\mathbf{x}, \mathbf{x}^+; \boldsymbol{\theta}) - \Delta(\mathbf{x}, \mathbf{x}^-; \boldsymbol{\theta}) \big)_+ \Big],$$

where $p_{\mathcal{D}_{\text{test}}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)$ denotes a distribution of triplets, $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)$, determined by the sampling strategy from $\mathcal{D}_{\text{test}}$, $(\cdot)_+ =: \max(0, \cdot)$, and ξ is a margin. Also, $\Delta(\mathbf{x}_1, \mathbf{x}_2; \boldsymbol{\theta})$ means the Euclidean distance between L_2 normalized outputs of the model as follows:

$$\Delta(\mathbf{x}_1, \mathbf{x}_2; \boldsymbol{\theta}) := \left\| \frac{\mathbf{f}(\mathbf{x}_1; \boldsymbol{\theta})}{\|\mathbf{f}(\mathbf{x}_1; \boldsymbol{\theta})\|^2} - \frac{\mathbf{f}(\mathbf{x}_2; \boldsymbol{\theta})}{\|\mathbf{f}(\mathbf{x}_2; \boldsymbol{\theta})\|^2} \right\|^2. \qquad (6)$$

Note that hard examples are involved for the computation of the triplet loss because we draw two positive samples $(\mathbf{x}, \mathbf{x}^+) \sim \mathcal{D}^{\text{std}}_{\text{test}}$ and a hard instance $\mathbf{x}^- \sim \mathcal{D}^{\text{hard}}_{\text{test}}$. Since the hard examples can be positive in some videos, we have to distinguish the hard examples from pure background instances. The cross-entropy loss cannot achieve this goal, and this is why we introduce the triplet loss and make the positive examples from a video clustered together while embedding them to be separated from the positives in the rest of videos and the backgrounds.

Meta-parameter optimization

We optimize the meta-parameters to encourage the adapted model parameters to minimize the expected meta-optimization loss, which is defined over multiple simulated tracking episodes. Given an episode trajectory of a tracking simulation, $\tau = (\mathcal{D}_{\text{init}}, \mathcal{D}_{\text{on}}, \mathcal{D}_{\text{test}}, \{\boldsymbol{\theta}^k_{\text{init}}\}^{K_{\text{init}}}_{k=1}, \{\boldsymbol{\theta}^k_{\text{on}}\}^{K_{\text{on}}}_{k=1})$, which is a collection of all the datasets and intermediate model parameters used in the episode, we optimize the meta-parameters using the following meta-optimization loss

$$\mathcal{L}_{\text{meta}}(\mathcal{M}) = \qquad (7)$$
$$\mathbb{E}_{p(\tau)} \big[\mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}}; \boldsymbol{\theta}^{K_{\text{init}}}_{\text{init}}) + \mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}}; \boldsymbol{\theta}^{K_{\text{on}}}_{\text{on}}) \big],$$

where $p(\tau)$ is a distribution of episode trajectories, defined by the strategy for sampling a video $\mathcal{V}$ from a video set Γ and for performing tracking simulation with meta-parameter $\mathcal{M}$. As shown in Figure 1 and Eq. (7), the meta-optimization loss adopts the test loss whose parameters $\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}$ and $\boldsymbol{\theta}_{\text{on}}^{K_{\text{on}}}$ are further represented by the meta-parameters $\mathcal{M}$. It implies that our meta-optimization loss $\mathcal{L}_{\text{meta}}$ is fully differentiable with respect to the meta-parameters $\mathcal{M}$ for optimization. By employing an optimizer such as ADAM (Kingma and Ba 2014), we update the meta-parameters using the following gradients:

$$\nabla_{\mathcal{M}}\mathcal{L}_{\text{meta}} = \mathbb{E}_{p(\tau)}\Bigg[\frac{\partial\mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}};\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}})}{\partial\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}}\frac{\partial\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}}{\partial\mathcal{M}} + \frac{\partial\mathcal{L}_{\text{test}}(\mathcal{D}_{\text{test}};\boldsymbol{\theta}_{\text{on}}^{K_{\text{on}}})}{\partial\boldsymbol{\theta}_{\text{on}}^{K_{\text{on}}}}\left\{\frac{\partial\boldsymbol{\theta}_{\text{on}}^{K_{\text{on}}}}{\partial\mathcal{M}} + \frac{\partial\boldsymbol{\theta}_{\text{on}}^{K_{\text{on}}}}{\partial\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}}\frac{\partial\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}}{\partial\mathcal{M}}\right\}\Bigg] \quad (8)$$

Eq. (8) illustrates that the meta-parameters are updated by optimizing tracking results, which implies that the meta-parameters $\mathcal{M}$ induce a strong object tracker based on the proposed fast adaptation algorithm.

4 Meta-learning One-Shot Channel Pruning

We present how to meta-learn to remove redundant channels given a ground-truth target bounding box at the first frame of a video. Our pruning network builds on the LASSO regression-based channel pruning (He, Zhang, and Sun 2017) and extends it by learning to predict a channel selection mask for an entire tracking episode from the first frame of a video. As in Section 3, our meta-learning for pruning is trained using simulated tracking episodes and thus enables the network to predict a mask appropriate for the future frames even in a previously unseen video. We first formulate the channel pruning task in a tracking episode and discuss how to leverage meta-learning for one-shot channel pruning. In this section, for notational simplicity, we regard a fully-connected layer as a 1×1 convolution layer.

Learning channel masks in a tracking simulation

We build on the LASSO pruning (He, Zhang, and Sun 2017) that learns channel selection masks $\mathcal{B} = \{\boldsymbol{\beta}^l\}_{l=1}^{L}$, where L is the number of layers in a CNN. The pruning network is trained to predict mask $\mathcal{B}$, which is regularized by sparsity constraints, so that the selected maps by mask $\mathcal{B}$ approximate original feature maps of the CNN. A channel selection mask $\boldsymbol{\beta}^l$ is a real-valued vector whose dimension is identical to the number of the channels in the l^{th} layer. We prune the channels corresponding to the zero-valued dimensions in $\boldsymbol{\beta}^l$.

Our LASSO pruning minimizes the following loss function:

$$\mathcal{L}_{\text{lasso}}(\mathcal{D},\mathcal{B};\boldsymbol{\theta}) = \lambda\sum_{l=1}^{L}\left\|\boldsymbol{\beta}^l\right\|_1 + \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}\Big[\sum_{l\in\mathcal{S}}\left\|F^l(\mathbf{x};\boldsymbol{\theta}) - F^l_{\mathcal{B}}(\mathbf{x};\boldsymbol{\theta})\right\|_2\Big], \quad (9)$$

where λ controls the degree of sparsity in channel selection and $\mathcal{S}$ is a set of layers to approximate the original feature map. $F^l(\mathbf{x};\boldsymbol{\theta})$ denotes a feature map in the l^{th} layer while $F^l_{\mathcal{B}}(\mathbf{x};\boldsymbol{\theta})$ represents a feature map pruned by mask $\mathcal{B}$.

Given an input image patch $F^0(\mathbf{x};\boldsymbol{\theta}) = F^0_{\mathcal{B}}(\mathbf{x};\boldsymbol{\theta}) = \mathbf{x}$, unmasked network sequentially computes features by $F^{l+1}(\mathbf{x};\boldsymbol{\theta}) = \sigma\big(\text{Conv}(F^l(\mathbf{x};\boldsymbol{\theta});\boldsymbol{\theta}^l)\big)$, where the parameters for l^{th} layer $\boldsymbol{\theta}^l$ and $\sigma(\cdot)$ is a non-linear activation function. Here we omit some components including pooling layers for notational simplicity. In contrast, the pruned feature maps are computed along layers as

$$F^{l+1}_{\mathcal{B}}(\mathbf{x};\boldsymbol{\theta}) = \boldsymbol{\beta}^{l+1}\odot\sigma\big(\text{Conv}(F^l_{\mathcal{B}}(\mathbf{x};\boldsymbol{\theta});\boldsymbol{\theta}^l)\big). \quad (10)$$

To consider pruning for a simulated tracking episode, we extend the objective of Eq. (9), defined on a dataset $\mathcal{D}$, into the following objective for an episode trajectory τ:

$$\mathcal{L}_{\text{ep}}(\tau,\mathcal{B}) = \mathcal{L}_{\text{lasso}}(\mathcal{D}_{\text{init}},\mathcal{B};\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}) + \sum_{k=1}^{K_{\text{on}}}\mathcal{L}_{\text{lasso}}(\mathcal{D}_{\text{on}},\mathcal{B};\boldsymbol{\theta}_{\text{on}}^{k}) + \mathcal{L}_{\text{lasso}}(\mathcal{D}_{\text{test}},\mathcal{B};\boldsymbol{\theta}_{\text{on}}^{K_{\text{on}}}). \quad (11)$$

This objective aims to find a single $\mathcal{B}$ that minimizes $\mathcal{L}_{\text{lasso}}$ over different pairs of parameters and datasets appearing in a single simulated tracking episode.

Meta-learning for channel mask prediction

The objective of Eq. (11) uses an episode trajectory from an entire simulated tracking episode to learn channel selection masks. However, speeding up the object tracking requires channels to be pruned even before the complete trajectory for a tracking episode is collected. Therefore, we further introduce one-shot channel pruning that amortizes optimization in Eq. (11) as a prediction of $\mathcal{B}$ based on a dataset coming from an initial frame of a video.

We build our pruning network on top of the CNN used for object tracking. It predicts $\mathcal{B}$ using $\mathcal{D}_{\text{init}}$ and $\boldsymbol{\theta}$. The prediction is denoted by a function outputting a set of vectors $\Psi(\mathcal{D}_{\text{init}};\boldsymbol{\theta},\boldsymbol{\phi}) = \{\psi^l(\mathcal{D}_{\text{init}};\boldsymbol{\theta},\boldsymbol{\phi})\}_{l=1}^{L}$, where $\psi^l(\mathcal{D}_{\text{init}};\boldsymbol{\theta},\boldsymbol{\phi})$ approximates $\boldsymbol{\beta}^l$, a channel mask for the l^{th} layer of the CNN. Note that $\boldsymbol{\phi} \in \mathbb{R}^c$ is a parameter of the pruning network that is not shared with the tracking CNN. Specifically, we construct the pruning network as $\psi^l(\mathcal{D}_{\text{init}};\boldsymbol{\theta},\boldsymbol{\phi}) = \frac{1}{N}\sum_{i=1}^{N}\text{MLP}\big(\text{AvgPool}\big(F^l(\mathbf{x}_i;\boldsymbol{\theta})\big);\boldsymbol{\phi}^l\big)$, where $\text{MLP}(\cdot;\boldsymbol{\phi}^l)$ is a multi-layer perceptron parameterized by $\boldsymbol{\phi}^l$ associated with the l^{th} layer, $\text{AvgPool}(\cdot)$ is a spatial global average pooling, and N is the number of samples in $\mathcal{D}_{\text{init}}$. Note that we pretrain $\boldsymbol{\theta}$ via meta-learning for fast adaptation and keep it fixed while optimizing $\boldsymbol{\phi}$ for training the pruning network.

The meta-learning objective of our one-shot pruning network $\Psi(\mathcal{D}_{\text{init}};\boldsymbol{\theta},\boldsymbol{\phi})$ is to minimize the loss

$$\mathcal{L}_{\text{prune}}(\boldsymbol{\phi}) = \mathbb{E}_{p(\tau)}\Big[\mathcal{L}_{\text{ep}}\big(\tau,\Psi(\mathcal{D}_{\text{init}};\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}},\boldsymbol{\phi})\big)\Big], \quad (12)$$

which substitute $\mathcal{B}$ in Eq. (11) with $\Psi(\mathcal{D}_{\text{init}};\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}},\boldsymbol{\phi})$ and train it with the samples from trajectory distribution $p(\tau)$, which is identical to the distribution in Eq. (7). In this work, we use $\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}$ in channel mask prediction $\Psi(\mathcal{D}_{\text{init}};\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}},\boldsymbol{\phi})$ so that the pruning network can exploit the parameters specialized for a video using the first frame with a target annotation. Since $\boldsymbol{\theta}_{\text{init}}^{K_{\text{init}}}$ is obtained from the initial frame of the video, our pruning network does not incur an additional

Table 1: Analysis of proposed components for meta-learning on OTB2015.

Method	$\mathcal{D}_{\text{on}}$	$\mathcal{D}_{\text{test}}^{\text{hard}}$	$\boldsymbol{\theta}_{\text{init}}^{0}$	$\mathcal{A}_{\text{init}}$	$\mathcal{A}_{\text{on}}$	Prunning	Succ(%)	Prec(%)	FPS
RT-MDNet (Jung et al. 2018)	-	-	-	-	-	-	65.0	88.5	42
MetaRTT	√	√	√	√	√		65.5	89.0	58
- Not using estimated target		√	√	√	√		58.6	81.2	54
- Not using hard instances	√		√	√	√		60.7	81.9	57
- Not using per-parameter / per-iteration lr (scalar lr)	√	√	√				60.9	82.8	57
MetaRTT+Prune	√	√	√	√	√	√	65.1	87.4	65
Baseline for Meta-Tracker (Park and Berg 2018)			√	√			56.8	81.2	40
- Using hard instances		√	√	√			62.7	86.2	41

computation cost in prediction at the future frames. Eq. (12) is a meta-learning objective as it defines an outer learning objective for an inner learning objective of Eq. (11).

5 Experiments

This section discusses the details of our approach in the application to RT-MDNet (Jung et al. 2018) and the performance of the proposed algorithm in comparisons to the state-of-the-art methods. We present the results from three versions of our trackers; MetaRTT is the model without one-shot network pruning, MetaRTT+Prune is the one with pruning, and MetaRTT+COCO is the model trained with additional data. The base model for all trackers is RT-MDNet.

Application to RT-MDNet

We apply our meta-learning algorithm to RT-MDNet, which is one of the state-of-the-art real-time object trackers. For efficient feature extraction from multiple image patches in a frame, RT-MDNet employs a two-stage feature extraction pipeline similar to (He et al. 2017). In the first stage, a convolutional neural network (CNN) computes a feature map from an input frame. To extract the feature descriptor from the shared feature map corresponding to an image patch, RT-MDNet employs RoIAlign layer (He et al. 2017), where a 3×3 dimensional feature map grid is interpolated and pooled. The feature descriptor for an image patch $\mathbf{x}$ is employed to compute the response for the patch using three fully connected layers, which is denoted by $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$.

Our definition of parameters $\boldsymbol{\theta}_{\text{init}}^{0}$ includes the parameters both before and after the RoIAlign layer. Therefore, the initial parameter $\boldsymbol{\theta}_{\text{init}}^{0}$ on both sides are meta-learned. However, similarly to the original RT-MDNet, the initial and online adaptations are not applied to the parameters for the shared CNN both during the tracking and the tracking simulation. Formally, there is no model adaptations in the shared CNN layers, *i.e.,* $\boldsymbol{\theta}_{\text{init}}^{k} = \boldsymbol{\theta}_{\text{init}}^{k-1}$ and $\boldsymbol{\theta}_{\text{on}}^{k} = \boldsymbol{\theta}_{\text{on}}^{k-1}$, while the model adaptions for the fully connected layers are given by Eq. (2) and (3). For meta-learning one-shot channel pruning, we define $\mathcal{S}$ in Eq. (9) as a set of the final shared convolutional layer and all the fully connected layers. We set K_{init} and K_{on} to 5 throughout our experiment.

Implementation details

We pretrain MetaRTT and MetaRTT+Prune on ImageNet-Vid (Russakovsky et al. 2015), which contains more than 3,000 videos with 30 object classes labeled for video object detection. We treat each object as a class-agnostic target to track but do not use its class label for pretraining. We randomly select 6 frames from a single video to construct an episode, and use the first frame for $\mathcal{D}_{\text{init}}$, the last frame for $\mathcal{D}_{\text{test}}^{\text{std}}$ and the remaining frames for $\mathcal{D}_{\text{on}}$. The meta-parameters are optimized over 40K simulated episodes using ADAM with fixed learning rate 10^{-4}. The network learning the pruning mask per layer is implemented as a 2-layer perceptron with a LeakyRelu activation and dropout regularization. We optimize the network by ADAM for 30K iterations with learning rate 5×10^{-5}. Other details follow the configuration of RT-MDNet. We pretrain MetaRTT+COCO on ImageNet-Vid and the augmented version of COCO (Lin et al. 2014). Since COCO is a dataset of images, we construct its augmented set by artificially generating pseudo-videos with 6 frames via random resizing and shift of the original images as in (Zhu et al. 2018). Our algorithm is implemented in PyTorch with 3.60 GHz Intel Core I7-6850k and NVIDIA Titan Xp Pascal GPU.

Ablation studies

Meta-learning for fast adaptation Table 1 illustrates the ablative results on OTB2015 (Wu, Lim, and Yang 2013), where we show the performance of several different model variants given by the adopted algorithm components. Specifically, $\mathcal{D}_{\text{on}}$ denotes whether each option employs the online adaptation based on the estimated targets; if the corresponding column is not checked, it means that meta-learning is performed using the ground-truth labels only. Also, $\mathcal{D}_{\text{test}}^{\text{hard}}$ indicates the use of hard examples for meta-training.

As expected, individual algorithm components make substantial contribution to improving accuracy and speed of our tracker. Our full algorithm maintains competitive accuracy and achieves significant speed-up compared to the baseline technique, RT-MDNet. For comparison with other meta-learning approaches for object tracking, we select Meta-Tracker (Park and Berg 2018). Unlike our meta-learning framework, Meta-Tracker employs meta-learning only for the initial adaptation, and it does not utilize the estimated targets for the online adaptation and the hard example mining for meta-training. According to our experiment, the meta-learning strategy proposed by Meta-Tracker shows relatively poor performance compared to MetaRTT while we also observe that the success rate and the precision of Meta-Tracker are improved by simply incorprating the component exploit-

Table 2: Accuracy comparison with respect to the number of iterations K_{init} and K_{on} for fast adaptation on OTB2015.

Method	K_{init}	K_{on}	Succ (%)	Prec (%)	FPS
RT-MDNet	50	15	65.0	88.5	42
	50	5	61.9	83.1	54
	5	15	60.6	80.8	43
	5	5	57.8	77.5	55
MetaRTT	5	5	65.5	89.0	58

Table 3: Analysis of one-shot channel pruning on OTB2015. PR denotes prune rate in this table.

Method	Succ (%)	Prec (%)	FPS	PR (%)
MetaRTT	65.5	89.0	58	0
MetaRTT+Prune	65.1	87.4	65	50 ± 5
- No meta-learning	61.7	83.1	64	49 ± 6
- No adaptive pruning	60.9	81.2	59	53

ing hard examples.

Table 2 presents how the number of iterations affects performance of our algorithm with meta-learning, MetaRTT, in comparison to RT-MDNet. MetaRTT achieves better success rate and precision while it has 38% of speed-up compared to RT-MDNet through a more efficient model adaptation. Note that a simple reduction of model update iterations in RT-MDNet degrades its accuracy while increasing speed.

Meta-learning for one-shot channel pruning Table 1 presents that MetaRTT+Prune is 12% faster than MetaRTT while achieving the almost identical success rate and precision. To analyze the proposed pruning method, we evaluate two variants of pruning strategies. One is the one-shot channel pruning without meta-learning, which optimizes the pruning network using a simulated tracking episode only at the first frame. The other option is the static pruning, which learns a universal channel pruning mask $\mathcal{B}$ offline for every video. Table 3 implies that both the meta-learning with complete simulated episodes and the input adaptive one-shot pruning are essential. It also presents that MetaRTT+Pruning improves speed substantially by removing approximately half of the network parameters with minimal accuracy loss.

Comparison with state-of-the-arts trackers

We compare the variations of our algorithm—MetaRTT, MetaRTT+Prune, and MetaRTT+COCO—to the state-of-the-art algorithms (Nam and Han 2016; Danelljan et al. 2017a; 2016; 2017b; Fan and Ling 2017; Jung et al. 2018; Zhu et al. 2018). We employ OTB2015 (Wu, Lim, and Yang 2015) and TempleColor (Liang, Blasch, and Ling 2015) datasets for evaluation. Because our goal is to accelerate tracking algorithms while maintaining their accuracy, the comparison with real-time trackers is more important. So, we highlight real-time trackers with solid lines in the graphs visualizing quantitative performance while the slow methods are represented with dashed lines.

Figure 2 shows that MetaRTT and MetaRTT+COCO outperform the compared state-of-the-art real-time trackers

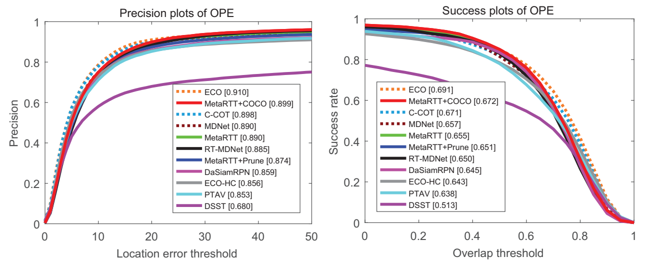

Figure 2: Quantitative results on OTB2015.

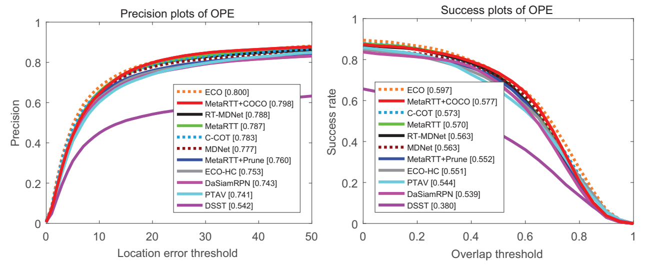

Figure 3: Quantitative results on TempleColor.

Table 4: Results of our algorithms on VOT2016, UAV123.

	UAV123		VOT2016
Method	Succ (%)	Prec (%)	EAO
RT-MDNet	52.8	77.2	0.322
MetaRTT	56.9	80.9	0.346
MetaRTT+COCO	56.6	80.9	0.350
MetaRTT+Prune	53.7	77.9	0.314

on OTB2015 in terms of accuracy while MetaRTT+Prune achieves competitive accuracy and improved speed compared to MetaRTT. Figure 3 illustrates that our trackers present outstanding performance in TempleColor as well. These results imply that the hyperparameters meta-learned on ImageNet-Vid work well on the standard benchmarks without manual tuning for the benchmarks; we fix all the hyperparameters except the meta-parameters in both datasets. The strong accuracy by MetaRTT+COCO also suggests that our tracker can be further improved by pretraining with better datasets.

Table 4 presents that the variations of our algorithm make significant improvements on other benchmark datasets, VOT2016 (Kristan et al. 2016) and UAV123 (Mueller, Smith, and Ghanem 2016).

6 Conclusion

We presented a novel meta-learning framework and its application to object tracking, which is more comprehensive and principled than the technique introduced in the prior study. The proposed meta-learning approach allows to learn fast adaptation and one-shot channel pruning algorithm, which leads to competitive accuracy and substantial speed-up. Our tracker achieves the state-of-the-art performance compared to the existing real-time tracking methods.

Acknowledgement This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT)[2014-0-00059, 2017-0-01780, 2016-0-00563], LG-CNS Co., Ltd., Seoul, and NRF Korea (NRF-2017M3C4A7069369).

References

Al-Shedivat, M.; Bansal, T.; Burda, Y.; Sutskever, I.; Mordatch, I.; and Abbeel, P. 2018. Continuous Adaptation via Meta-learning in Nonstationary and Competitive Environments. In *ICLR*.

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; and De Freitas, N. 2016. Learning to Learn by Gradient Descent by Gradient Descent. In *NeurIPS*.

Bertinetto, L.; Valmadre, J.; Henriques, J. F.; Vedaldi, A.; and Torr, P. 2016. Fully-Convolutional Siamese Networks for Object Tracking. In *ECCVW*.

Choi, J.; Chang, H. J.; Fischer, T.; Yun, S.; Lee, K.; Jeong, J.; Demiris, Y.; and Choi, J. Y. 2018. Context-Aware Deep Feature Compression for High-Speed Visual Tracking. In *CVPR*.

Danelljan, M.; Robinson, A.; Khan, F. S.; and Felsberg, M. 2016. Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In *ECCV*.

Danelljan, M.; Bhat, G.; Shahbaz Khan, F.; and Felsberg, M. 2017a. ECO: Efficient Convolution Operators for Tracking. In *CVPR*.

Danelljan, M.; Häger, G.; Khan, F. S.; and Felsberg, M. 2017b. Discriminative Scale Space Tracking. *TPAMI* 39(8):1561–1575.

Danelljan, M.; Bhat, G.; Khan, F. S.; and Felsberg, M. 2019. Atom: Accurate tracking by overlap maximization. In *CVPR*, 4660–4669.

Duan, Y.; Andrychowicz, M.; Stadie, B.; Ho, O. J.; Schneider, J.; Sutskever, I.; Abbeel, P.; and Zaremba, W. 2017. One-Shot Imitation Learning. In *NeurIPS*.

Fan, H., and Ling, H. 2017. Parallel Tracking and Verifying: A Framework for Real-Time and High Accuracy Visual Tracking. In *ICCV*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *ICML*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both Weights and Connections for Efficient Neural Network. In *NeurIPS*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressed Deep Neural Networks with Pruning, Trained Quantization, and Huffman Coding. In *ICLR*.

Han, B.; Sim, J.; and Adam, H. 2017. Branchout: Regularization for Online Ensemble Tracking with Convolutional Neural Networks. In *CVPR*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask R-CNN. In *ICCV*.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *ICCV*.

Hochreiter, S.; Younger, A. S.; and Conwell, P. R. 2001. Learning to Learn using Gradient Descent. In *ICANN*.

Huang, C.; Lucey, S.; and Ramanan, D. 2017. Learning Policies for Adaptive Tracking with Deep Feature Cascades. In *ICCV*.

Jung, I.; Son, J.; Baek, M.; and Han, B. 2018. Real-Time MDNet. In *ECCV*.

Kingma, D. P., and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kristan, M.; Leonardis, A.; Matas, J.; Felsberg, M.; Pflugfelder, R.; Čehovin Zajc, L.; Vojir, T.; Häger, G.; Lukežič, A.; and Fernandez, G. 2016. The visual object tracking VOT2016 challenge results. In *ECCVW*.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-SGD: Learning to Learn Quickly for Few Shot Learning. *arXiv preprint arXiv:1707.09835*.

Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; and Yan, J. 2019. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 4282–4291.

Liang, P.; Blasch, E.; and Ling, H. 2015. Encoding Color Information for Visual Tracking: Algorithms and Benchmark. *TIP* 24(12):5630–5644.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*, 740–755. Springer.

Mueller, M.; Smith, N.; and Ghanem, B. 2016. A Benchmark and Simulator for UAV Tracking. In *ECCV*.

Nam, H., and Han, B. 2016. Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. In *CVPR*.

Nam, H.; Baek, M.; and Han, B. 2016. Modeling and Propagating CNNs in a Tree Structure for Visual Tracking. *arXiv preprint arXiv:1608.07242*.

Park, E., and Berg, A. C. 2018. Meta-Tracker: Fast and Robust Online Adaptation for Visual Object Trackers. *ECCV*.

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

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-Learning with Memory-Augmented Neural Networks. In *ICML*.

Thrun, S., and Pratt, L. 1998. Learning to Learn: Introduction and Overview. In *Learning to Learn*. Springer. 3–17.

Valmadre, J.; Bertinetto, L.; Henriques, J. F.; Vedaldi, A.; and Torr, P. 2017. End-to-end Representation Learning for Correlation Filter based Tracking. In *CVPR*.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning Structured Sparsity in Deep Neural Networks. In *NeurIPS*.

Wu, Y.; Lim, J.; and Yang, M.-H. 2013. Online Object Tracking: A Benchmark. In *CVPR*.

Wu, Y.; Lim, J.; and Yang, M. 2015. Object Tracking Benchmark. *TPAMI* 37(9):1834–1848.

Zhu, Z.; Wang, Q.; Li, B.; Wu, W.; Yan, J.; and Hu, W. 2018. Distractor-Aware Siamese Networks for Visual Object Tracking. In *ECCV*.