Tracking by Instance Detection: A Meta-Learning Approach

G. Wang, C. Luo, X. Sun, Z. Xiong, W. Zeng

University of Science and Technology of China,
Microsoft Research Asia

Introduction:

- Automatic localization of a target, given a bounding box in the initial frame

- **Tracking problem:** closely related to the detection problem (instance detection)

- **Major difference:**
    - Object detection locates objects of some predefined classes and its output does not differentiate between intra-class instances
    - Object tracking only looks for a particular instance, which may belong to any known or unknown object class, that is specified in the initial frame

- Object detection techniques are used extensively in object tracking:
    - SiamRPN variants, ATOM, DiMP

Motivations & Contributions:

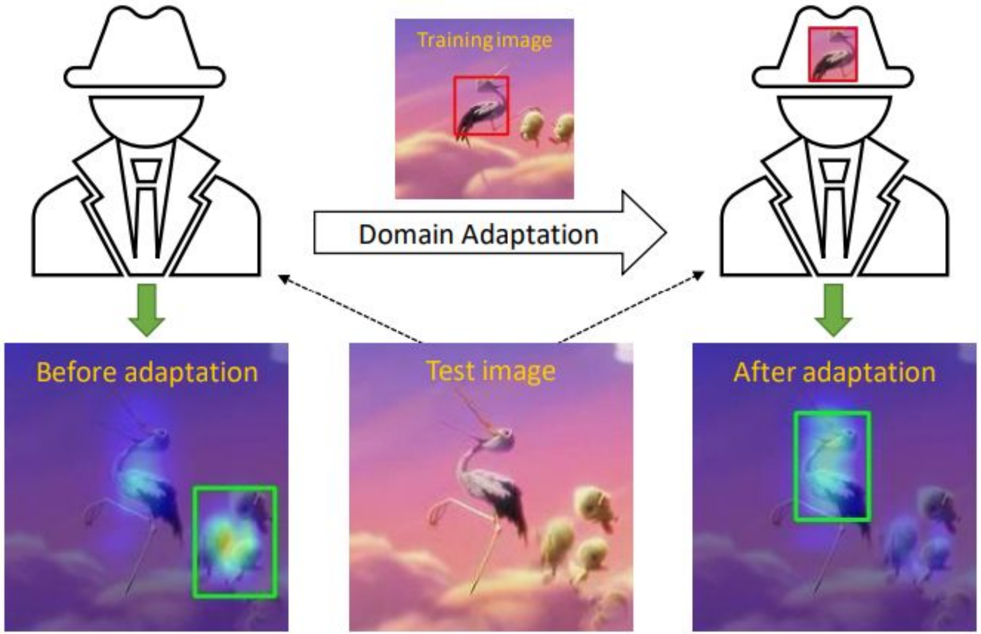
- **Aim:** Directly convert a modern object detector into a high-performance tracker
- **Main Challenge:** Obtain a good initialization of the detector without overfitting
- **Solution:** Meta-learning
- **Meta-learning Categorization:**
  - **Meta-Representation ("What?") :** Choice of representation of meta-knowledge (e.g., model parameters)
  - **Meta-Optimizer ("How?") :** Choice of optimizer to use for the outer level during meta-training (e.g., gradient descent forms)
  - **Meta-Objective ("Why?") :** Choice of meta-objective, task distribution, and data-flow
- **Model-Agnostic Meta-Learning (MAML):** A learning strategy to initialize SOTA detectors

Detector-MAML: Three-Step Approach

1- Pick any modern object detector trained with gradient descent

2- Conduct offline training (or initialization) with MAML on a large number of tracking sequences

3- Perform domain adaptation using the initial frame



- **During Tracking:** Training with more samples

Learning an Instance Detector with MAML

- ***Inner-Level Optimization*:** Given a video Vi, collect a set of training samples
- **Update detector on support set by a *k*-step GD**

k-step gradient descent (GD) algorithm

Parameters of a detector

Set of training samples: Support set

$$\boldsymbol{\theta}_k \equiv \mathrm{GD}_k(\boldsymbol{\theta}_0, \mathcal{D}_i^s)$$

Input image

Label in support set

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha \frac{1}{|\mathcal{D}_i^s|} \sum_{(x,y)\in\mathcal{D}_i^s} \nabla_{\boldsymbol{\theta}_{k-1}} \mathcal{L}(\underline{h(x; \boldsymbol{\theta}_{k-1})}, y),$$

Learning rate: a predefined constant

Data-label pair

Loss function

Detector model

Learning an Instance Detector with MAML (cont.)

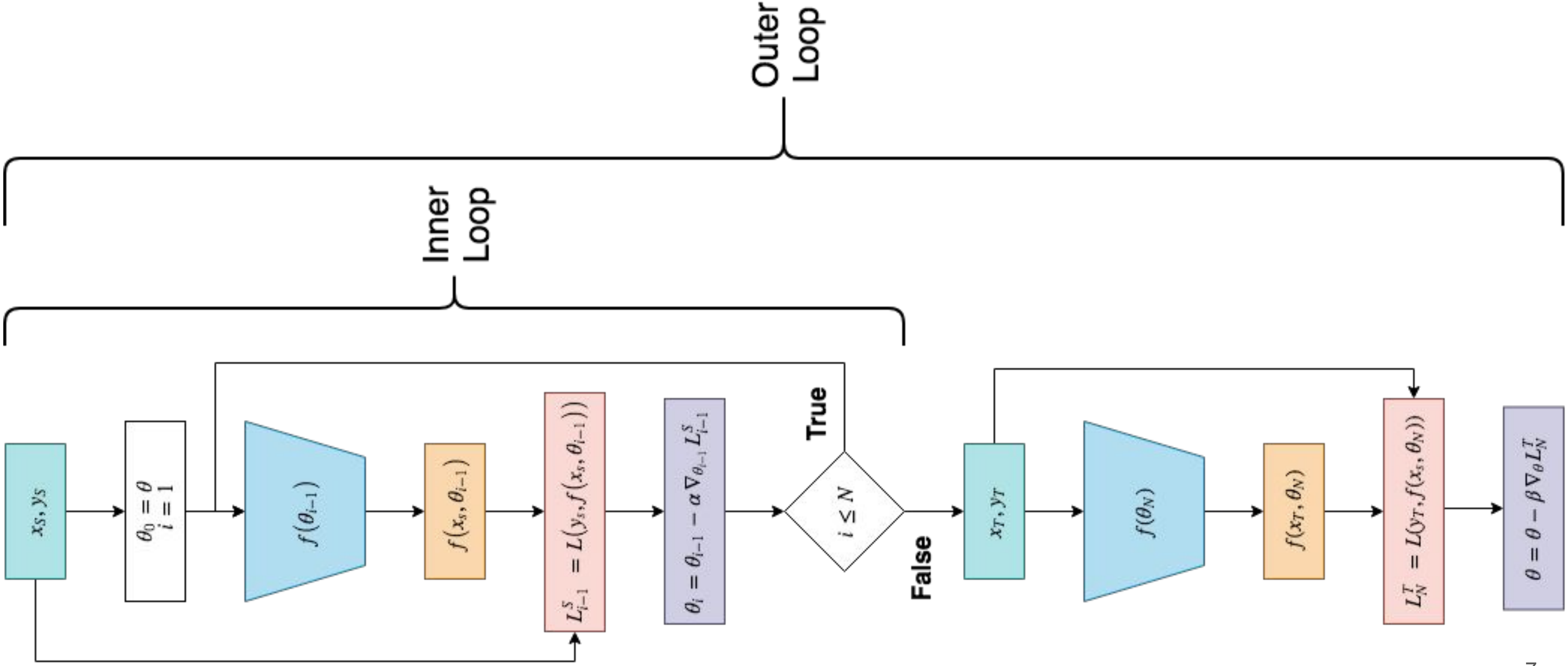
- ***Outer-Level Optimization***: Collect another set of samples from the same video V_i
- **Evaluate Generalization Ability of Trained Detector**
- **Goal:** Find a good initialization status for any tracking video
- Calculate the loss on the target set by applying the trained detector

$$F(\boldsymbol{\theta}_0, \mathcal{D}_i) = \frac{1}{|\mathcal{D}_i^t|} \sum_{(x,y)\in\mathcal{D}_i^t} \mathcal{L}(h(x; \boldsymbol{\theta}_k), y)$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}_0} \underbrace{\frac{1}{N}}_{\text{Total number of videos}} \sum_{i}^{N} F(\boldsymbol{\theta}_0, \mathcal{D}_i)$$

$$\mathcal{D}_i = \{\underbrace{\mathcal{D}_i^s}_{\text{Support Set: Three images}}, \underbrace{\mathcal{D}_i^t}_{\text{Target Set: One image}}\}$$

MAML Computational Graph



MAML Modifications

- **MAML Drawbacks :**
  - Backpropagation through GD steps is costly in terms of memory
  - Suffer from vanishing gradients
  - Catastrophic forgetting problem
  - Same weight on different pieces of knowledge within an episode
  - Hard to scale to tasks involving medium or large datasets

- **MAML++:** Introduces a set of tricks to stabilize the training of MAML
- **MetaSGD:** Train learnable learning rates for every parameter

MAML++: Multi-Step Loss Optimization

- ❑ Take the parameters after every step of inner-level GD to minimize the loss on target set, instead of only using the parameters after the final step
- ❑ **Trick:** initialization parameter θ_0 (before updating) also contributes to the outer-level loss

Crucial for stabilizing the gradients

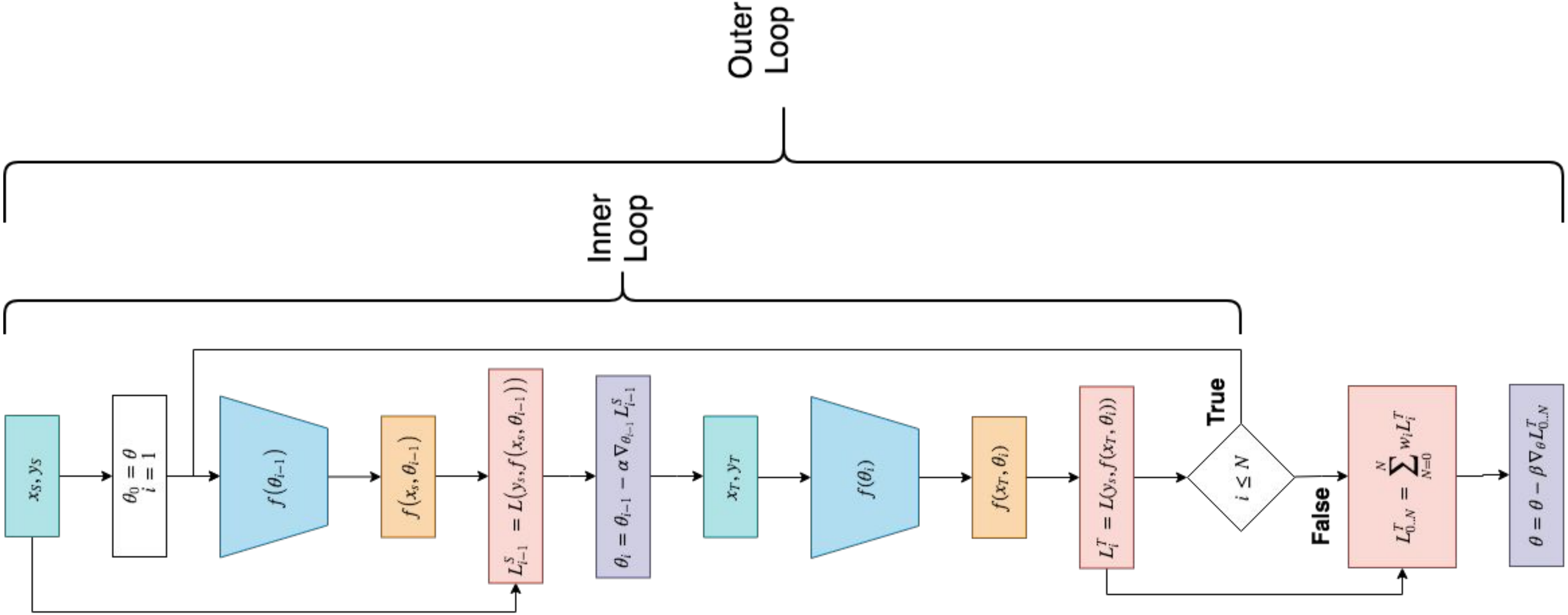
$$F(\boldsymbol{\theta}_0, \mathcal{D}_i) = \frac{1}{|\mathcal{D}_i^t|} \sum_{(x,y)\in\mathcal{D}_i^t} \mathcal{L}(h(x; \boldsymbol{\theta}_k), y)$$

Number of inner-level steps

$$F(\boldsymbol{\theta}_\mathbf{0}, \mathcal{D}_i) = \frac{1}{|\mathcal{D}_i^t|} \sum_{(x,y)\in\mathcal{D}_i^t} \sum_{k=0}^{K} \gamma_k \mathcal{L}(h(x; \boldsymbol{\theta}_k), y)$$

Loss weight for each step

MAML++ Computational Graph



Meta-SGD: Kernel-Wise Learnable Learning Rate (KLLR)

- A learnable learning rate for each parameter in the model
- α is a tensor with the same size as θ_k
- Setting up a learning rate for every parameter will double the model size
- Arrange the learnable learning rates in a kernel-wise manner

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha \frac{1}{|\mathcal{D}_i^s|} \sum_{(x,y)\in\mathcal{D}_i^s} \nabla_{\boldsymbol{\theta}_{k-1}} \mathcal{L}(h(x;\boldsymbol{\theta}_{k-1}), y)$$

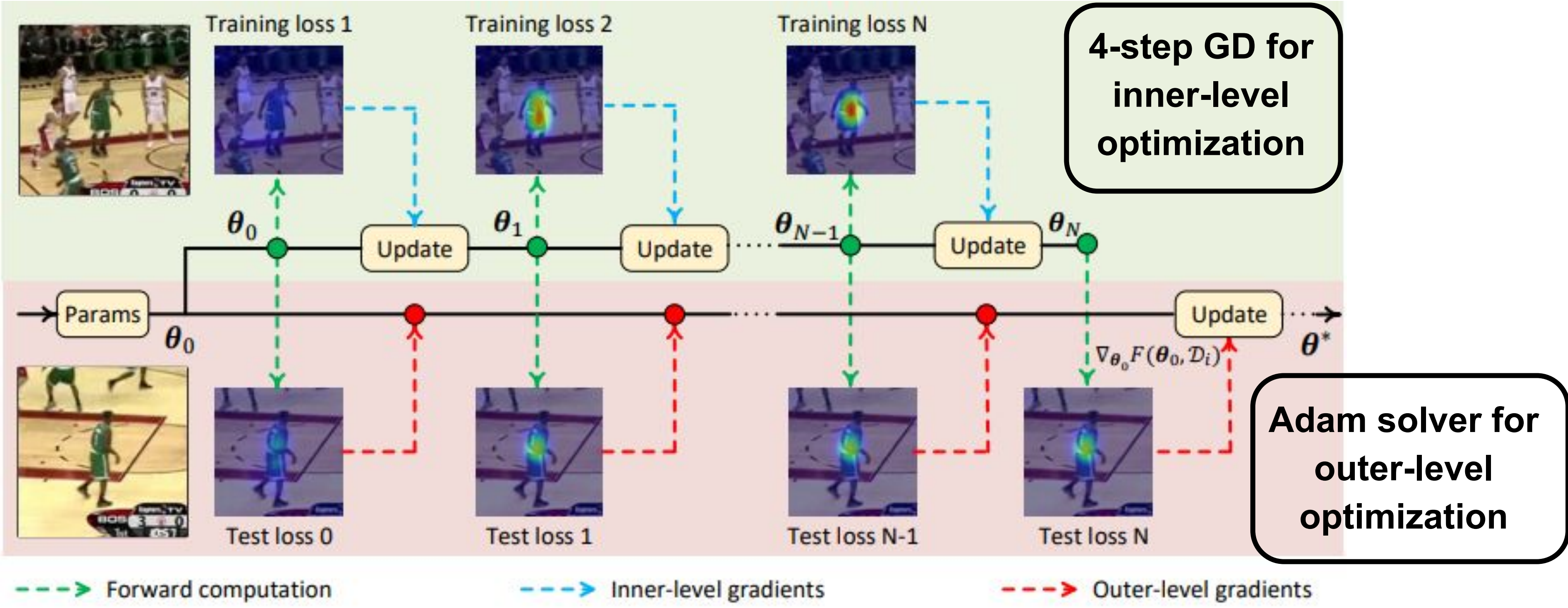
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\alpha} \odot \frac{1}{|\mathcal{D}_i^s|} \sum_{(x,y)\in\mathcal{D}_i^s} \nabla_{\boldsymbol{\theta}_k} \mathcal{L}(h(x;\boldsymbol{\theta}_k), y)$$

Element-wise product

The only difference compared to MAML is to parametrize task learning rate in vector form when meta-training.

Training Pipeline: Stabilized the Procedure by MAML++ & Meta-SGD

A few steps of SGD optimization is performed on the support images



Updated parameters after each step for calculating the meta-gradient based on testing images

1- Retina-MAML and FCOS-MAML

- **Single-stage detectors:** Backbone network, Classification head, Regression head

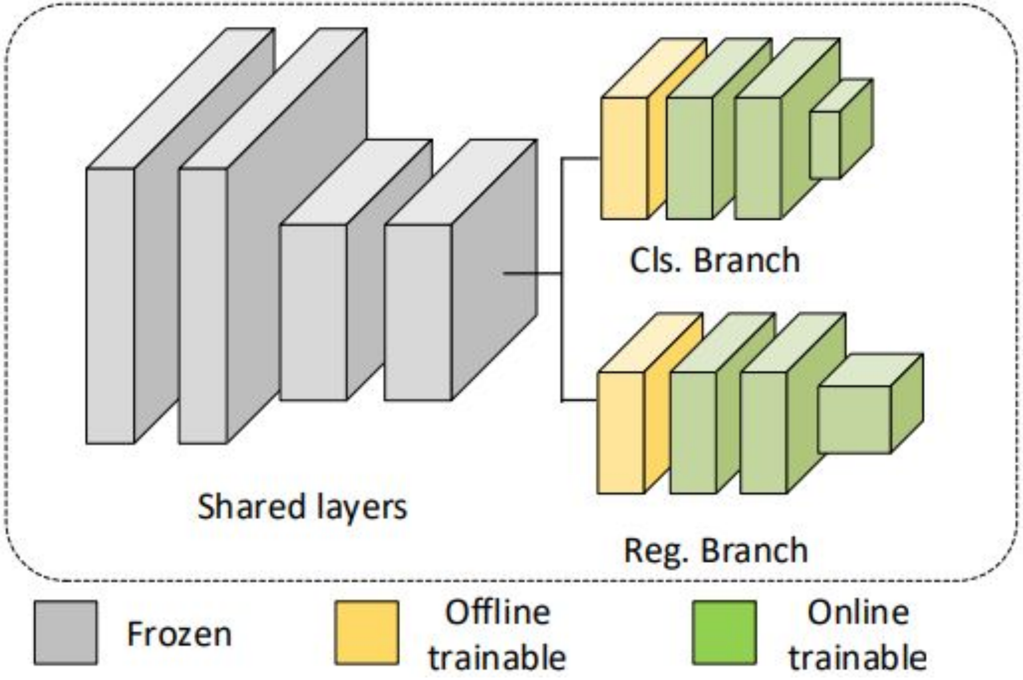
- **Anchor-based RetinaNet:**
  - Each pixel in the feature maps is associated with several anchors
  - **Classification head:** Classify whether each anchor has sufficient overlap with an object
  - **Regression head:** Predict the relative differences between each anchor and the corresponding ground-truth box

- **Anchor-free FCOS:**
  - **Classification head:** Classify whether each pixel in the feature maps is within the central area of an object
  - **Regression head:** Directly estimates the four offsets from the pixel to the object boundaries

Anchors: Predefined Prior Boxes

Network Architecture

❏ **Backbone:** ResNet-18



- First three blocks are frozen after ImageNet pre-training
- Block-4 is independently trained during offline training
- Block-5 is removed
- Online training only involves a subset of trainable layers

- **RetinaNet:** Pre-define a single anchor box with a size of 64 × 64 pixels

2- Offline MAML training: Loss

- **Retina-MAML:**

- Anchor box: Positive (or negative) label when its IoU with the GT box is greater than 0.5 (or less than 0.3)
- Classification branch: Focal loss
- Regression branch: Smooth L1 loss

- **FCOS-MAML:**

- Centerness scores: L2 loss
- Regression branch: L1 loss

2- Offline MAML training: Training Data

- ❑ **Datasets:** MS-COCO, GOT10k, TrackingNet, and LaSOT-train
- ❑ **LaSOT and TrackingNet:** Only sample one frame for every three or ten frames
- ❑ Training images are cropped and resized into a resolution of 263 × 263
- ❑ **Standard data augmentation:** Random scaling and shifting

2- Offline MAML training: Optimization

- ***Inner-level optimization***:
  - 4-step GD
  - Initialize KLLR α: 0.001
  - Initialize multi-step loss weights γk: Equal contribution and gradually anneal (parameters from later steps will attract more attention)

- ***Outer-level optimization***:
  - Adam optimizer with a starting learning rate 0.0001
  - Each iteration: Sample 32 pairs of images
  - Train for 20 epochs with 10,000 iterations per epoch

3- Domain Adaptation (Given an initial BB of target):

- ❏ Generate a patch with resolution 263 × 263
- ❏ Adopt zoom in/out data augmentation to construct the support set
- ❏ Update tracker by a 5-step GD

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\alpha} \odot \frac{1}{|\mathcal{D}_i^s|} \sum_{(x,y)\in\mathcal{D}_i^s} \nabla_{\boldsymbol{\theta}_k} \mathcal{L}(h(x;\boldsymbol{\theta}_k), y)$$

- ❏ **For each search region patch:**

- Detector locates hundreds of candidate bounding boxes
- Standard post-processing pipeline: Shape penalty and cosine window functions
- Tracking result: Candidate box with the highest score

3- Domain Adaptation (Given an initial BB of target):

❑ **During tracking (40 FPS on a single NVIDIA P100 GPU):**

- Gradually enlarge support set

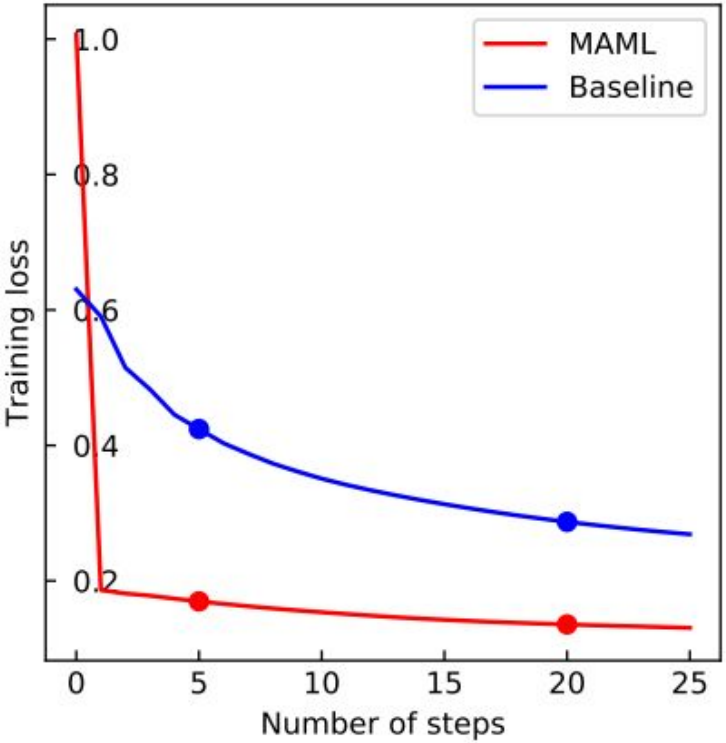
❑ **Online updating** (on updated support set & 1-step GD to maintain a high speed):

- After every n = 10 frames or when a distracting peak is detected (peak-to-sidelobe is greater than 0.7)
- Tracking result above a predefined threshold, it will be added into the support set
- Buffer at most 30 training images in the support set
- Earlier samples, except the initial one, will be discarded

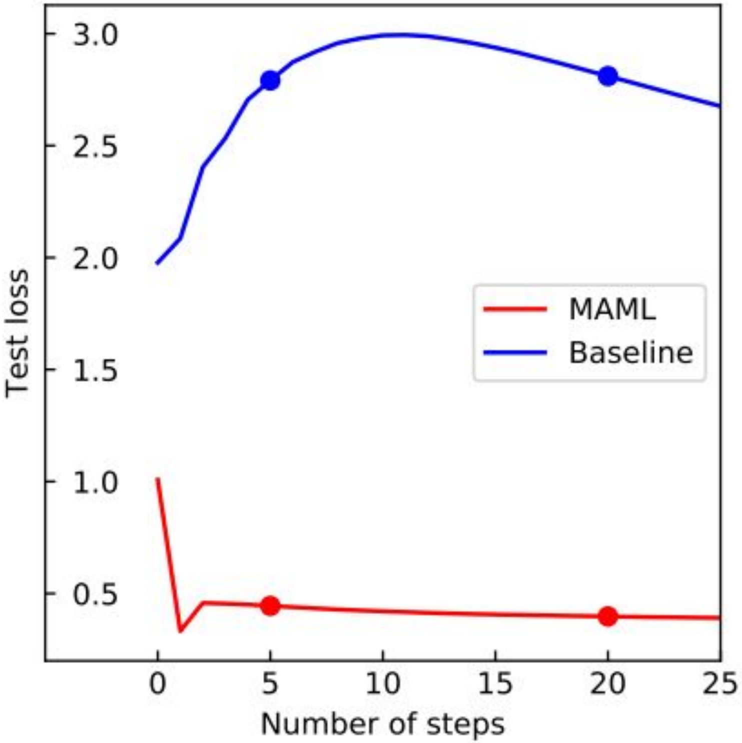
FCOS-MAML: Training Procedure

Baseline detector: Standard GD

Training image



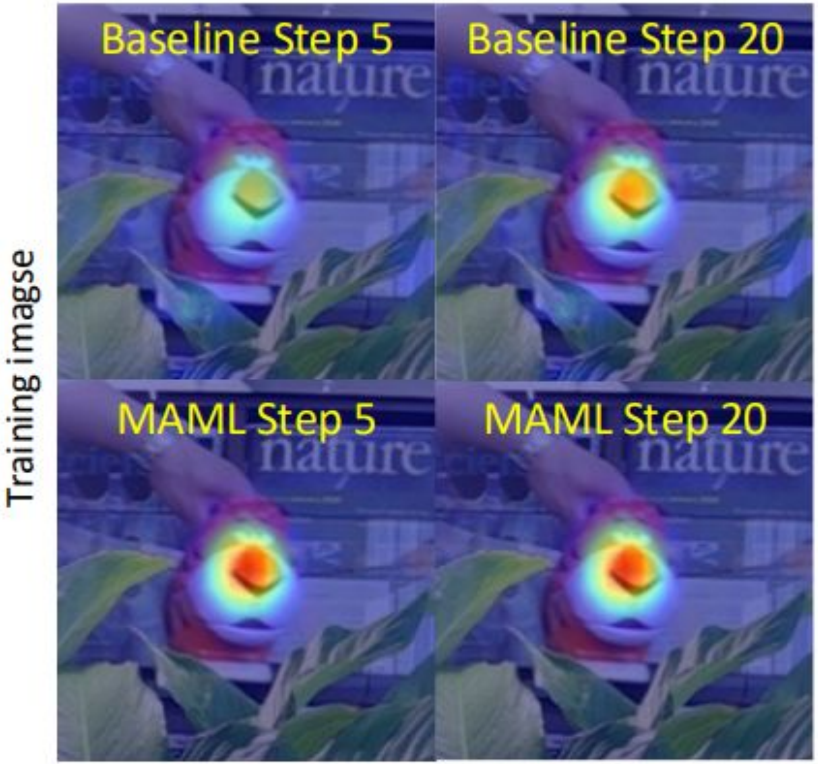
Testing image

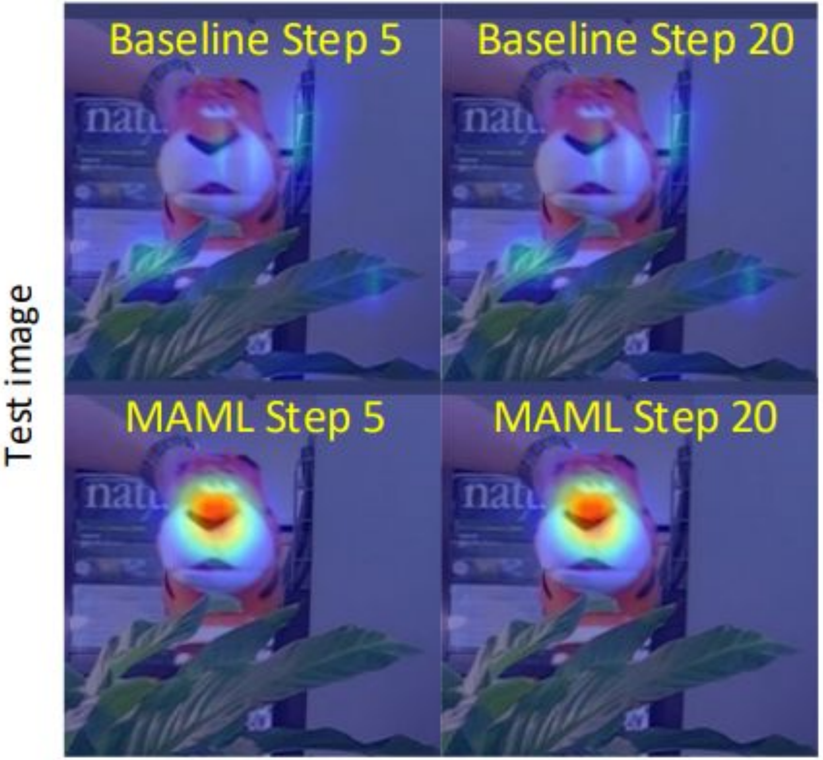


FCOS-MAML: Training Procedure

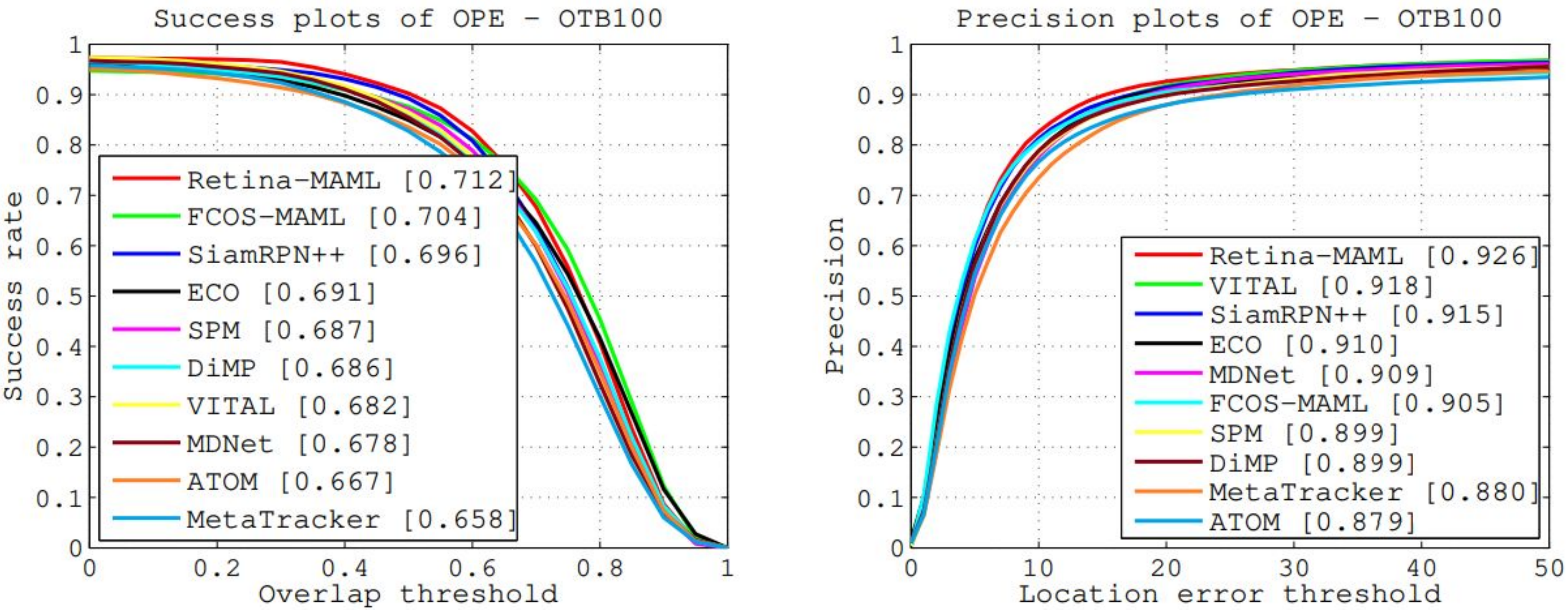
- MAML detector convergences quickly and has strong generalization ability

Baseline detector: Standard GD





Experiments on OTB-100 Dataset:



Experiments on VOT-2018 Dataset:

	↑EAO	↑Accuracy	↓Robustness
DRT [29]	0.356	0.519	0.201
SiamRPN++ [17]	0.414	0.600	0.234
UPDT [4]	0.378	0.536	0.184
LADCF [35]	0.389	0.503	0.159
ATOM [6]	0.401	0.590	0.204
DiMP-18 [3]	0.402	0.594	0.182
DiMP-50 [3]	0.440	0.597	**0.153**
FCOS-MAML	0.392	**0.635**	0.220
Retina-MAML	**0.452**	0.604	0.159

Experiments on TrackingNet & LaSOT Datasets:

AUC of Success Plot

Normalized Precision (N-Prec.)

	TrackingNet		LaSOT-test
	AUC	N-Prec.	AUC
C-RPN [9]	0.669	0.746	0.455
SiamRPN++ [17]	0.733	0.800	0.496
SPM [32]	0.712	0.779	0.471
ATOM [6]	0.703	0.771	0.515
DiMP-18 [3]	0.723	0.785	0.532
DiMP-50 [3]	0.740	0.801	**0.569**
FCOS-MAML	**0.757**	**0.822**	0.523
Retina-MAML	0.698	0.786	0.480

Thanks for your attention.

Questions / Answers