Object tracking and re-identification

Sigmund Rolfsjord

Overview

Curriculum:

Highly relevant video CVPR18
https://youtu.be/LBJ20kxr1a0?t=3038

Relevant til 1:08:00

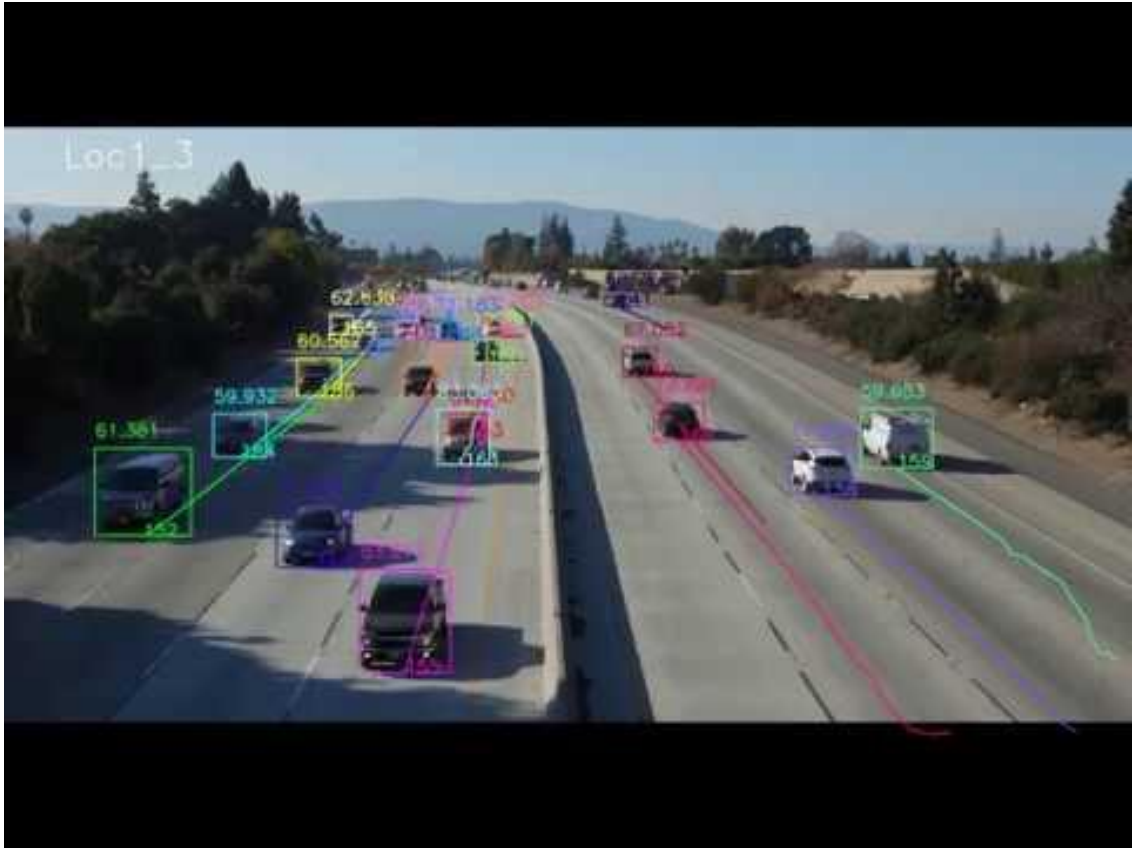
Overview of state-of-art: Slides,
http://prints.vicos.si/publications/files/365

Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning

Learning Multi-Domain Convolutional Neural Networks for Visual Tracking

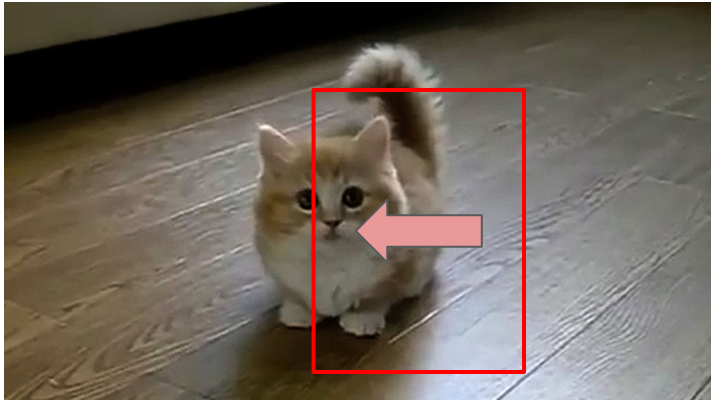
High Performance Visual Tracking with Siamese Region Proposal Network

Tracking



Learning movement

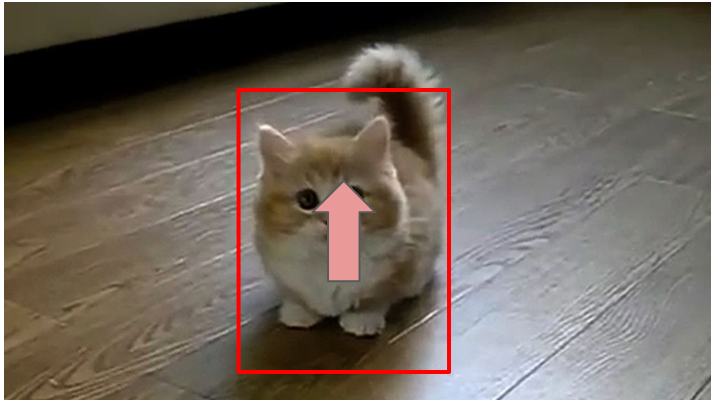
Left



Transition based tracking

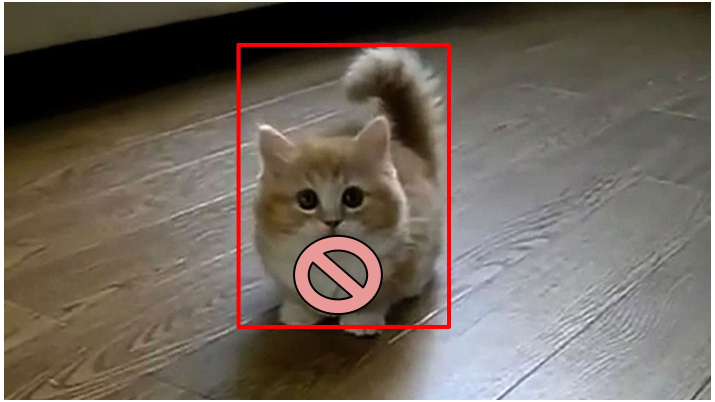
Learning movement

Right

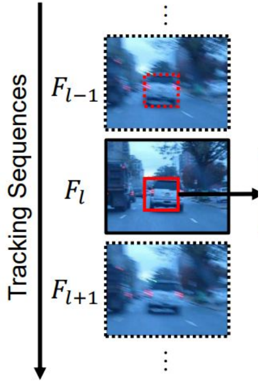


Learning movement

Stop

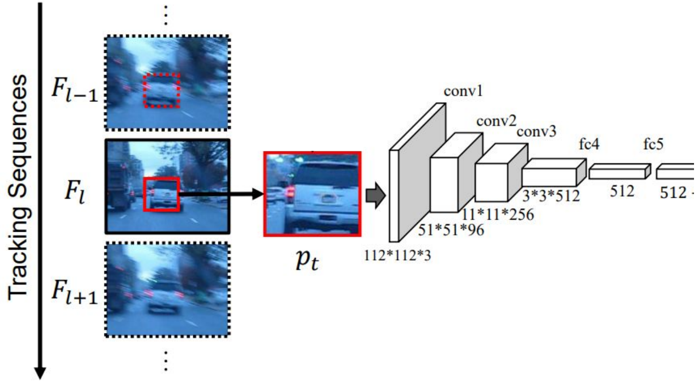


Tracking by learning transitions



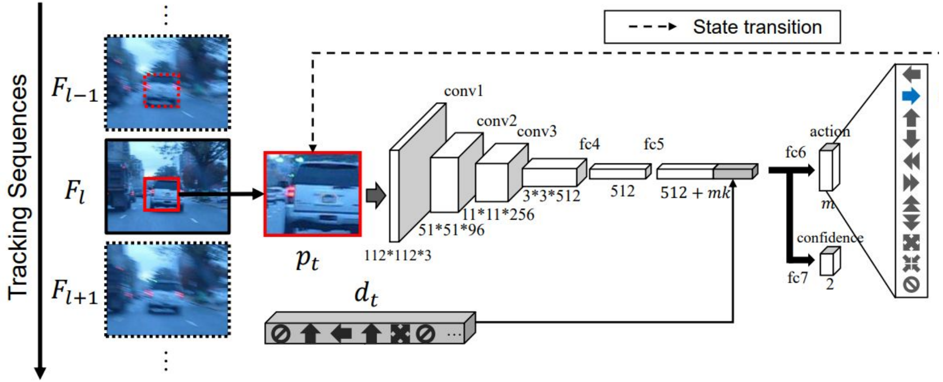
[Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning]

Tracking by learning transitions



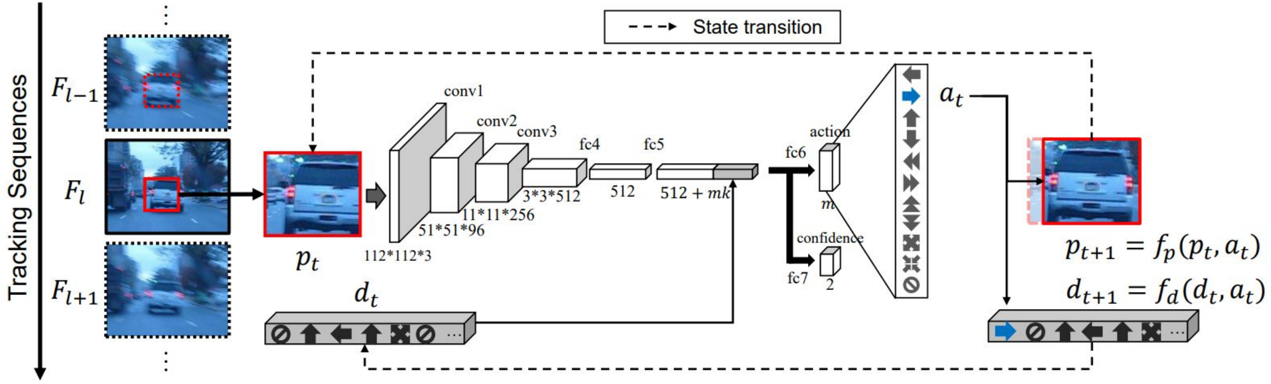
Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning

Tracking by learning transitions



[Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning]()

Tracking by learning transitions

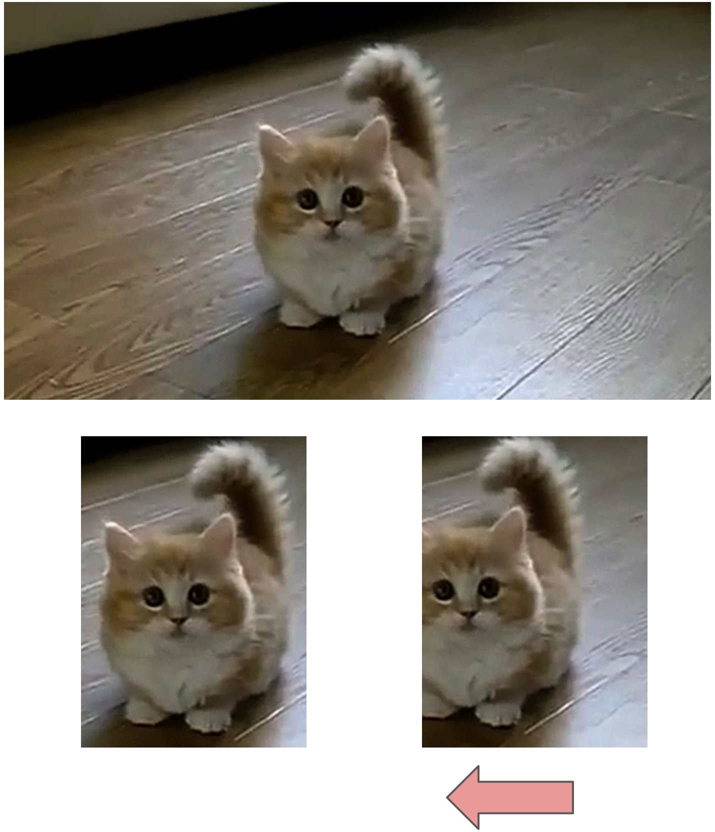


[Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning]()

Training the ADNetwork

Three step training process:

1. Supervised training with state-action pairs



Training the ADNetwork

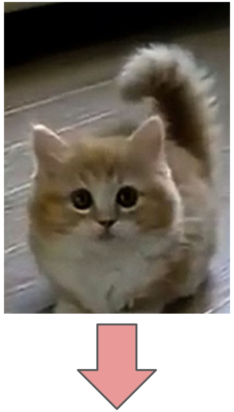
Three step training process:

1. Supervised training with state-action pairs
    a. Use tracking sequence or static data.
    b. Generate state-action pairs with backward action
    c. Train action and confidence score with softmax cross-entropy loss

$$o_j^{(act)} = \arg\max_a IoU(\bar{f}(p_j, a), G),$$

$$o_j^{(cls)} = \begin{cases} 1, & \text{if } IoU(p_j, G) > 0.7 \\ 0, & \text{otherwise.} \end{cases}$$

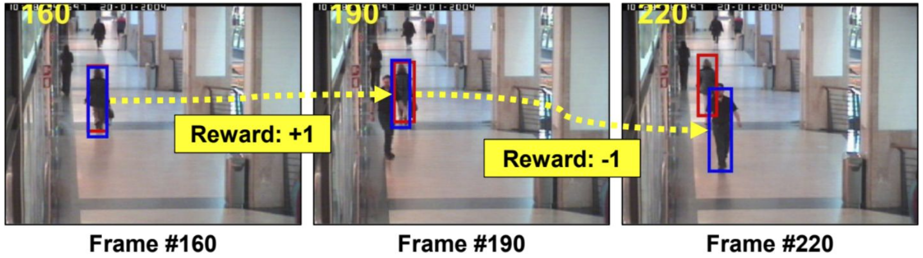
$$L_{SL} = \frac{1}{m}\sum_{j=1}^{m} L({o_j}^{(act)}, \hat{o_j}^{(act)}) + \frac{1}{m}\sum_{i=j}^{m} L({o_j}^{(cls)}, \hat{o_j}^{(cls)})$$



Training the ADNetwork

Three step training process:

1. Supervised training with state-action pairs
2. Train policy with reinforcement learning
    a. Input “real tracking dataset”, where multiple actions is required for each frame.
    b. Also work for unlabelled intermediate frames
    c. Iterate until stop-signal
    d. Give reward +1 if final result is success and -1 if it fails (<0.7 IOU)
    e. Set *z* (reward) for unlabelled steps as the same as the final reward.



[Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning]

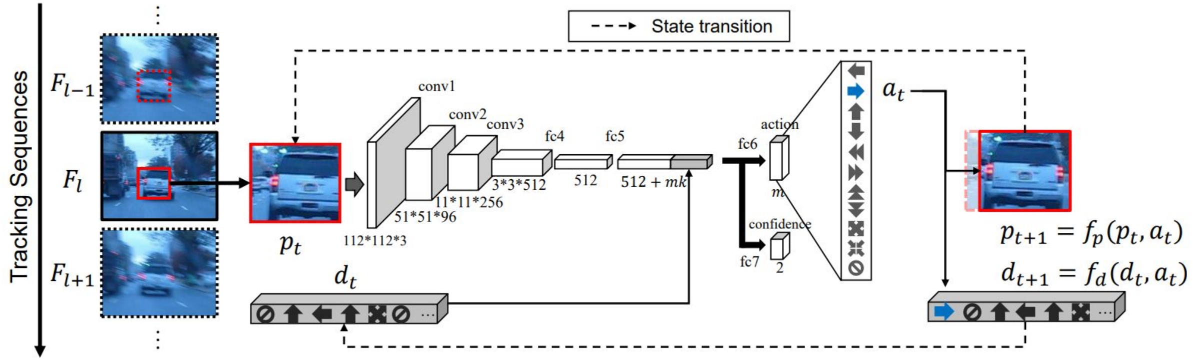
$$a_{t,l} = arg\max_{a} p(a|s_{t,l}; W_{RL}),$$

$$\Delta W_{RL} \propto \sum_{l}^{L} \sum_{t}^{T_l} \frac{\partial \log p(a_{t,l}|s_{t,l}; W_{RL})}{\partial W_{RL}} z_{t,l}.$$

Training the ADNetwork

Three step training process:

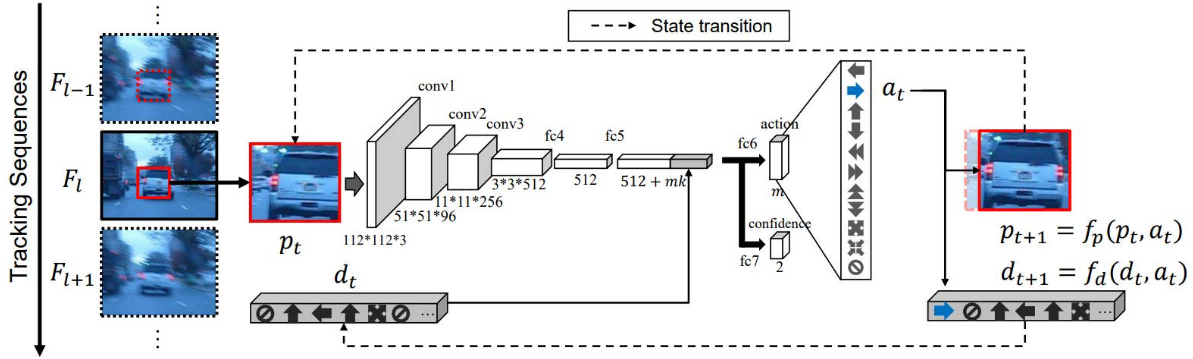
1. Supervised training with state-action pairs
2. Train policy with reinforcement learning



Training the ADNetwork

Three step training process:

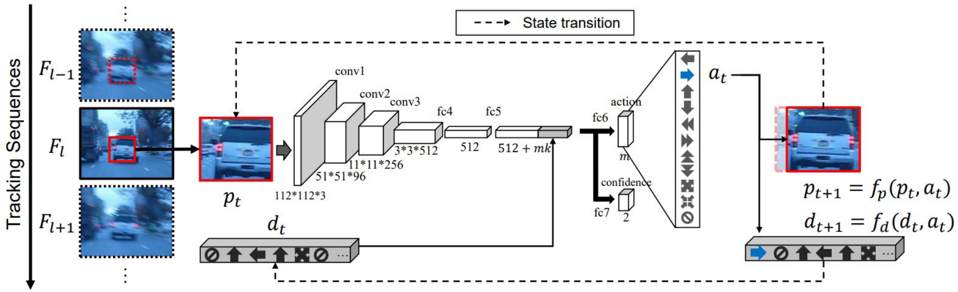
1. Supervised training with state-action pairs
2. Train policy with reinforcement learning
3. ???



Training the ADNetwork

Three step training process:

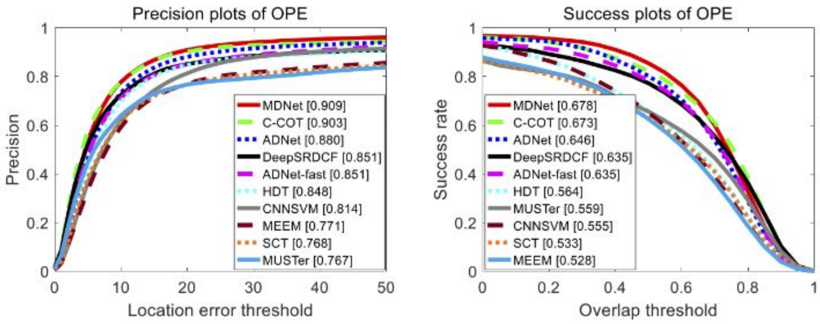
1. Supervised training with state-action pairs
2. Train policy with reinforcement learning
3. ~~Profit~~ Online-learning
    a. The network don’t know what it is tracking (basically object detection)
    b. Fine-tune fully connected layers (fc4-fc7)
    c. Train in the same way as in the supervised setting. Random sample boxes around the target region.
    d. Initial box trained with 300 surrounding boxes
    e. Boxes with confidence over 0.5 trained with 30 surrounding boxes.
    f. Relocating procedure with 250 random sampled boxes, if confidens is too low



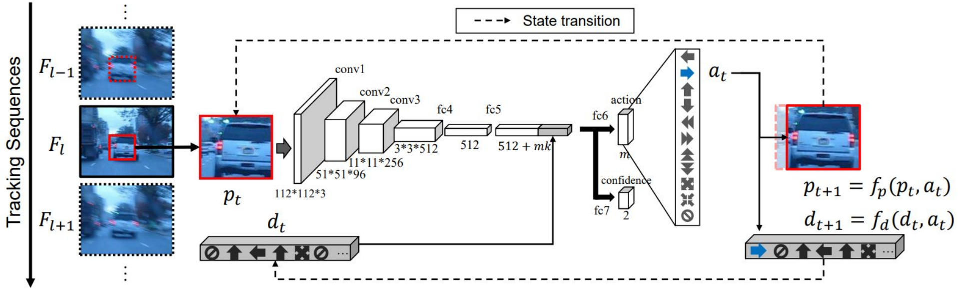
ADNetwork results

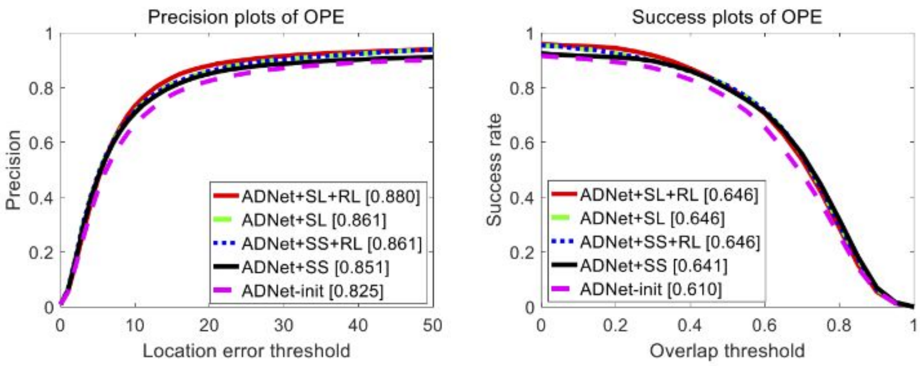
Table 1: Summary of experiments on OTB-100.

	Algorithm	Prec.(20px)	IOU(AUC)	FPS	GPU
	ADNet	88.0%	0.646	2.9	O
	ADNet-fast	85.1%	0.635	15.0	O
Non real-time	MDNet [24]	90.9%	0.678	< 1	O
	C-COT [9]	90.3%	0.673	< 1	O
	DeepSRDCF [8]	85.1%	0.635	< 1	O
	HDT [25]	84.8%	0.564	5.8	O
	MUSTer [15]	76.7%	0.528	3.9	X
Real-time	MEEM [42]	77.1%	0.528	19.5	X
	SCT [5]	76.8%	0.533	40.0	X
	KCF [13]	69.7%	0.479	223	X
	DSST [7]	69.3%	0.520	25.4	X
	GOTURN [12]	56.5%	0.425	125	O



(b) OTB-100





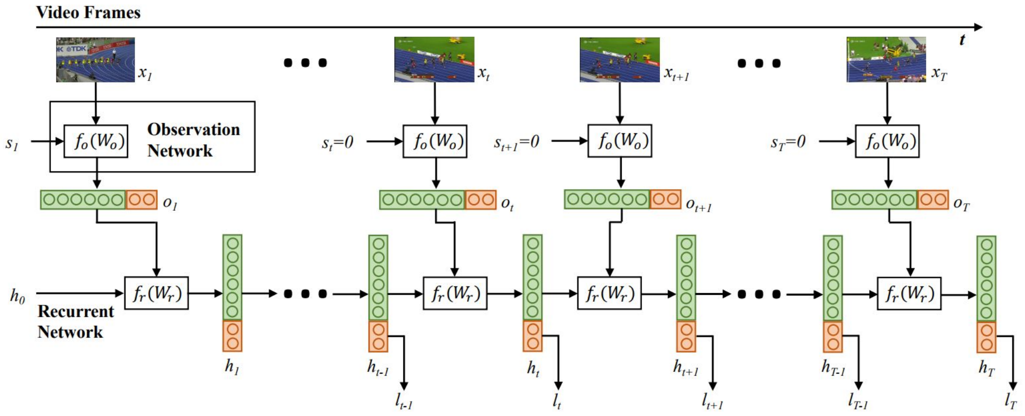
End-to-end tracking

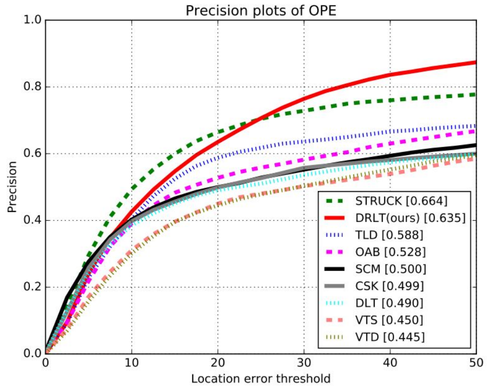
As an alternative to online-learning, you can use RNN.

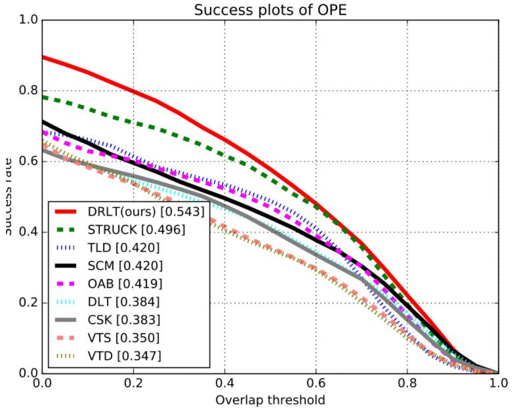
- Features trained on detection
- RNN on top

Very fast 270 fps on GTX 1080

Results far behind AD- and MDNet







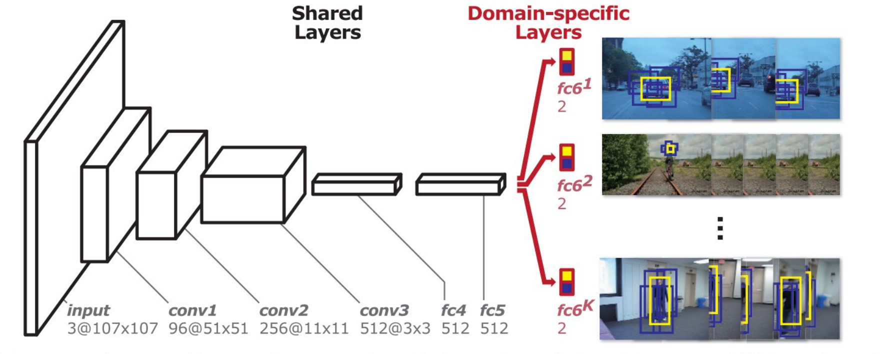
[Deep Reinforcement Learning for Visual Object Tracking in Videos]

Online-training based tracking

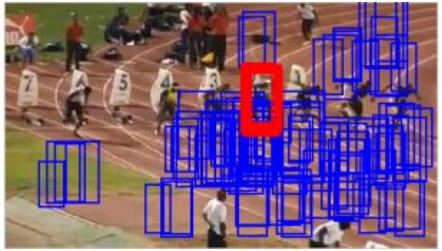
Online-training for detection - MDNet

Train domain specific detection:

- One final layer for each sequence
- Shared bottom network
- softmax cross-entropy loss, for negative/positive samples
- Random sample around

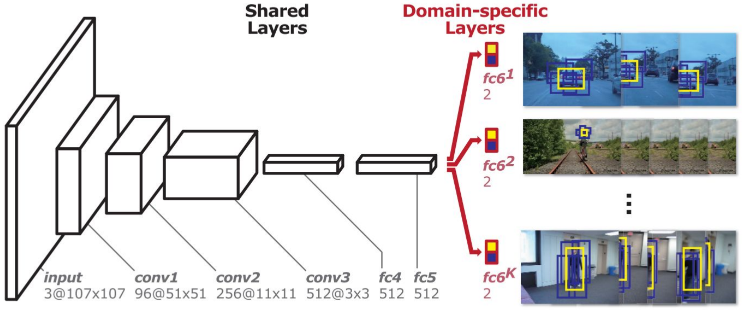


[Learning Multi-Domain Convolutional Neural Networks for Visual Tracking]



Training MDNet

- Generate surrounding boxes with centers from gaussian distribution
- Take 50 with IOU > 0.7 as positive and 200 with IOU < 0.5 as negative.
- Train bounding box regression on positive samples. (only first iteration)



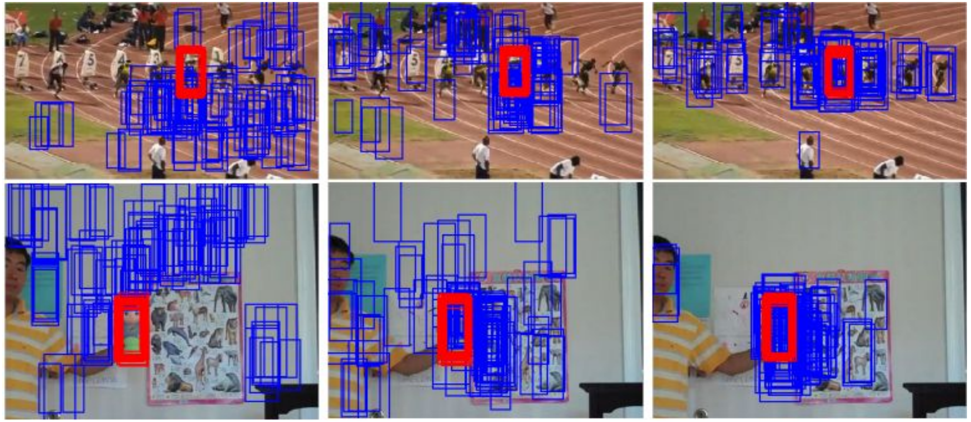
[Learning Multi-Domain Convolutional Neural Networks for Visual Tracking]

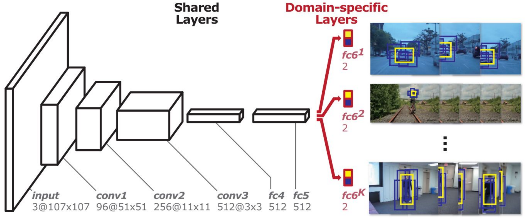
Training MDNet

Hard example mining:

- Remember scores for negative examples
- Sample negative examples with high positive score more frequently

Training data becomes more efficient for each batch.



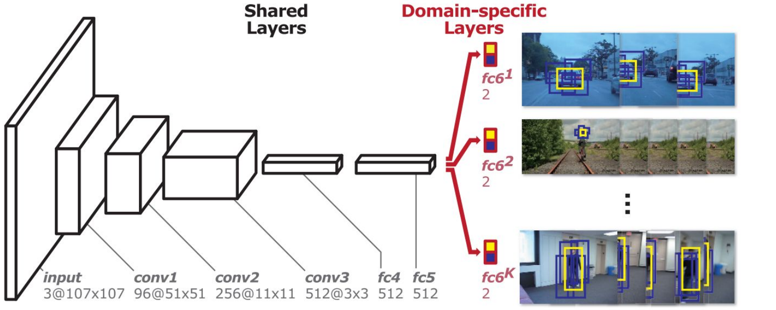


[Learning Multi-Domain Convolutional Neural Networks for Visual Tracking]

Tracking with MDNet

In addition to training procedure.

- If p(x | w) > 0.5 for most likely sample
  - Add sample boxes to online training set
  - Adjust x with bounding box regression
- Fine-tune network with online training set.

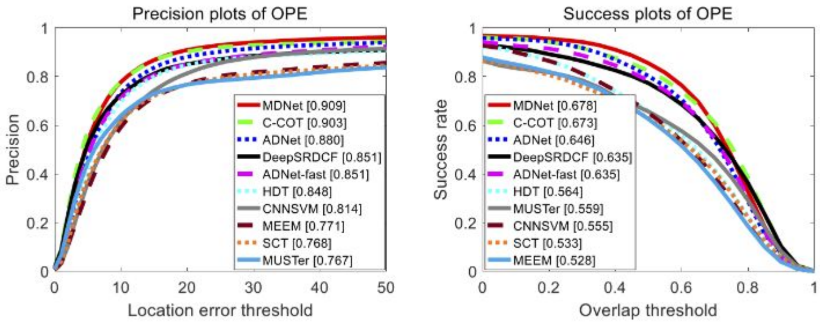


[Learning Multi-Domain Convolutional Neural Networks for Visual Tracking]

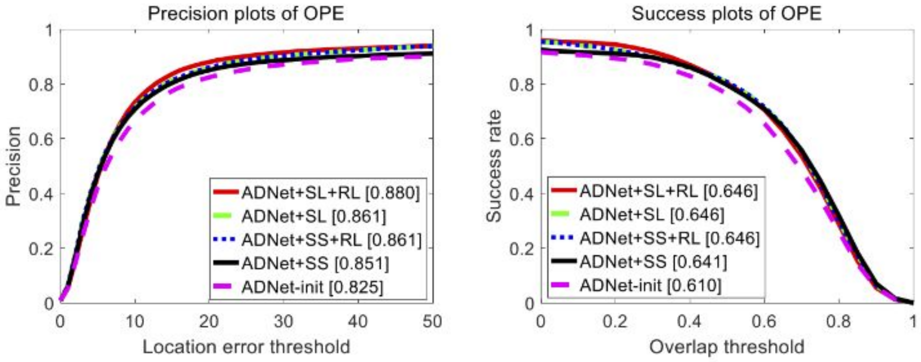
MDNet compared to ADNet

Table 1: Summary of experiments on OTB-100.

	Algorithm	Prec.(20px)	IOU(AUC)	FPS	GPU
	ADNet	88.0%	0.646	2.9	O
	ADNet-fast	85.1%	0.635	15.0	O
Non real-time	MDNet [24]	90.9%	0.678	< 1	O
	C-COT [9]	90.3%	0.673	< 1	O
	DeepSRDCF [8]	85.1%	0.635	< 1	O
	HDT [25]	84.8%	0.564	5.8	O
	MUSTer [15]	76.7%	0.528	3.9	X
Real-time	MEEM [42]	77.1%	0.528	19.5	X
	SCT [5]	76.8%	0.533	40.0	X
	KCF [13]	69.7%	0.479	223	X
	DSST [7]	69.3%	0.520	25.4	X
	GOTURN [12]	56.5%	0.425	125	O

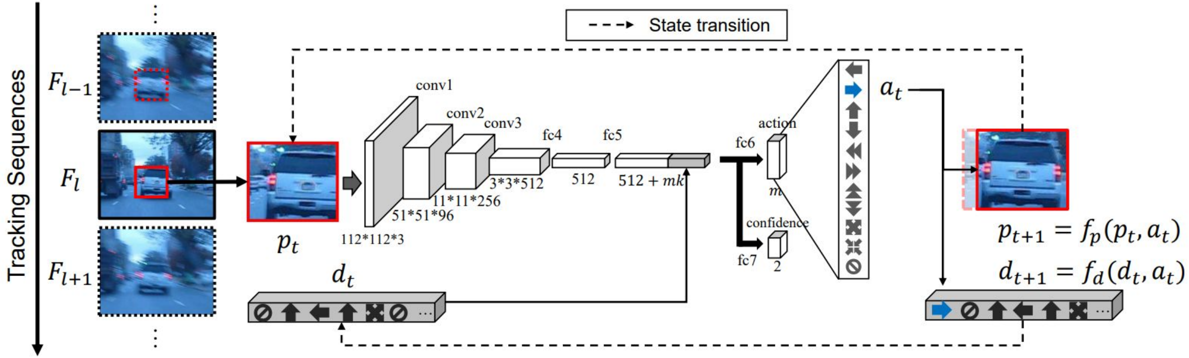


(b) OTB-100



ADNet is faster

ADNet is only using the “full MDNet” many samples, when it lose track.



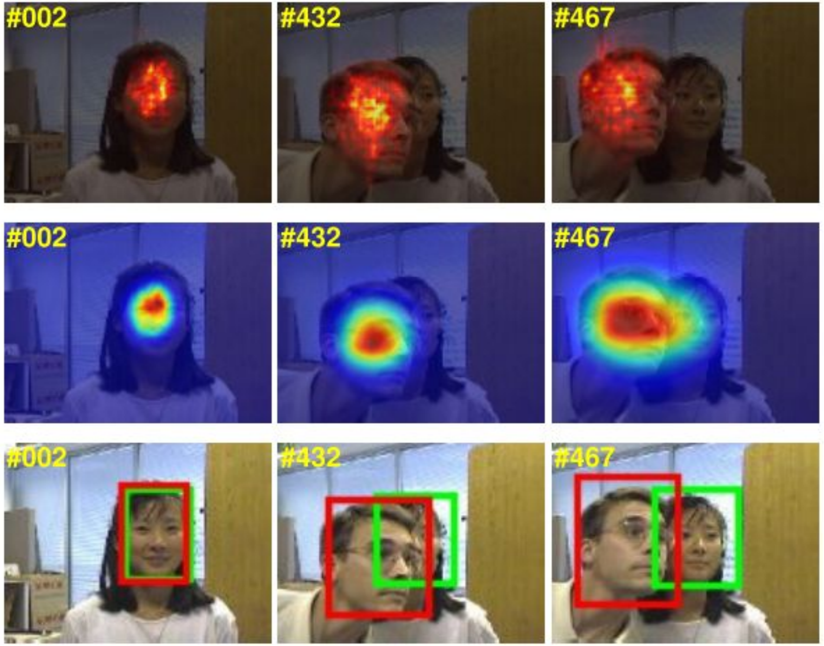
Other additions to MDNet

Problems with tracking networks:

Many videos only have one person, on cat etc. that your tracking. Mainly classifying person in the nearby region can give good results.

Effect is especially strong if the network is pretrained on detection or classification dataset.

Typically different way of forcing MDNet to focus on relevant features.



[Deep Attentive Tracking via Reciprocative Learning]()

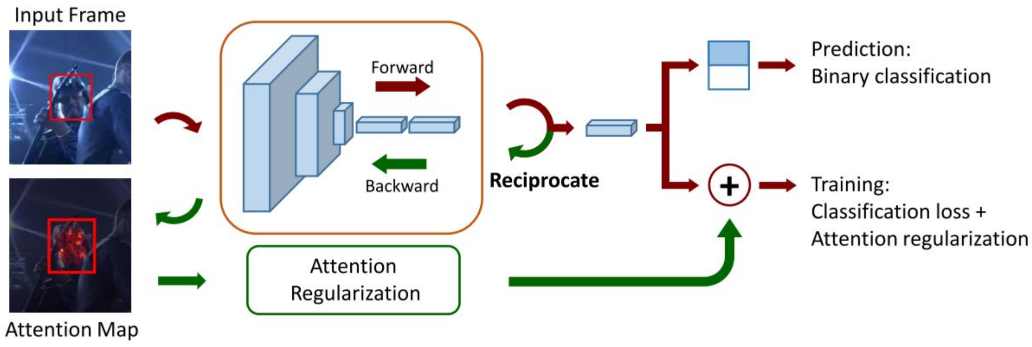
Deep Attentive Tracking via Reciprocative Learning

Finding attention-maps, by gradient.

A_c is the attention map for class c

I is an input feature map

$f_c(I)$ is the probability for class c



How can you change the features to influence the class.

$$A_c = \left.\frac{\partial f_c(I)}{\partial I}\right|_{I=I_0}$$

[Deep Attentive Tracking via Reciprocative Learning]()

Deep Attentive Tracking via Reciprocative Learning

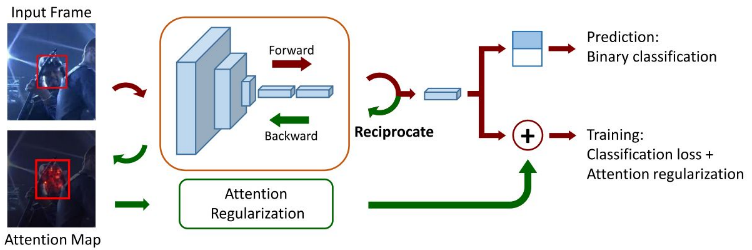
Finding attention-maps, by gradient.

Loss basically says:

Put high importance of features inside box (target)

Forcing the network to distribute attention to all regions of the object.

$$A_c = \frac{\partial f_c(I)}{\partial I}\bigg|_{I=I_0}$$



$$R_{(y=1)} = \frac{\sigma_{A_p}}{\mu_{A_p}} + \frac{\mu_{A_n}}{\sigma_{A_n}},$$

$$R_{(y=0)} = \frac{\mu_{A_p}}{\sigma_{A_p}} + \frac{\sigma_{A_n}}{\mu_{A_n}}.$$

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \cdot [y \cdot R_{(y=1)} + (1-y) \cdot R_{(y=0)}],$$

Deep Attentive Tracking via Reciprocative Learning

Deep Attentive Tracking via Reciprocative Learning

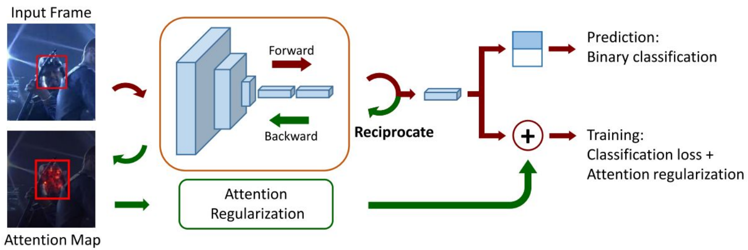
Finding attention-maps, by gradient.

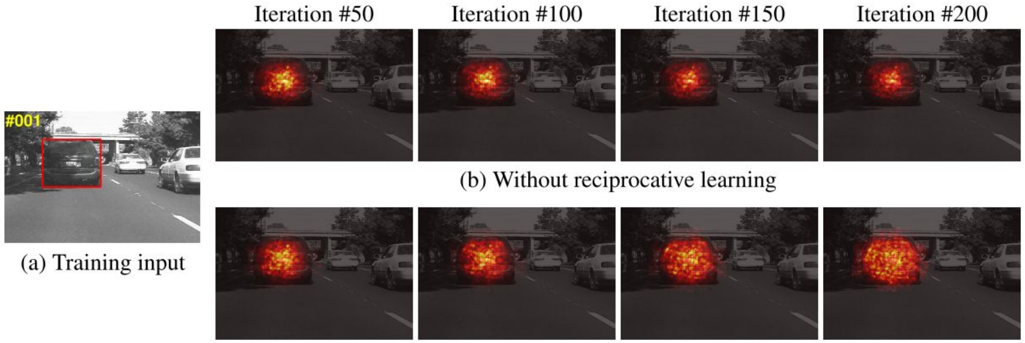
Loss basically says:

Put high importance of features inside box (target)

Forcing the network to distribute attention to all regions of the object.

Not only tracking object by some key feature.





(a) Training input

(b) Without reciprocative learning

(c) With reciprocative learning

[Deep Attentive Tracking via Reciprocative Learning]

Deep Attentive Tracking via Reciprocative Learning

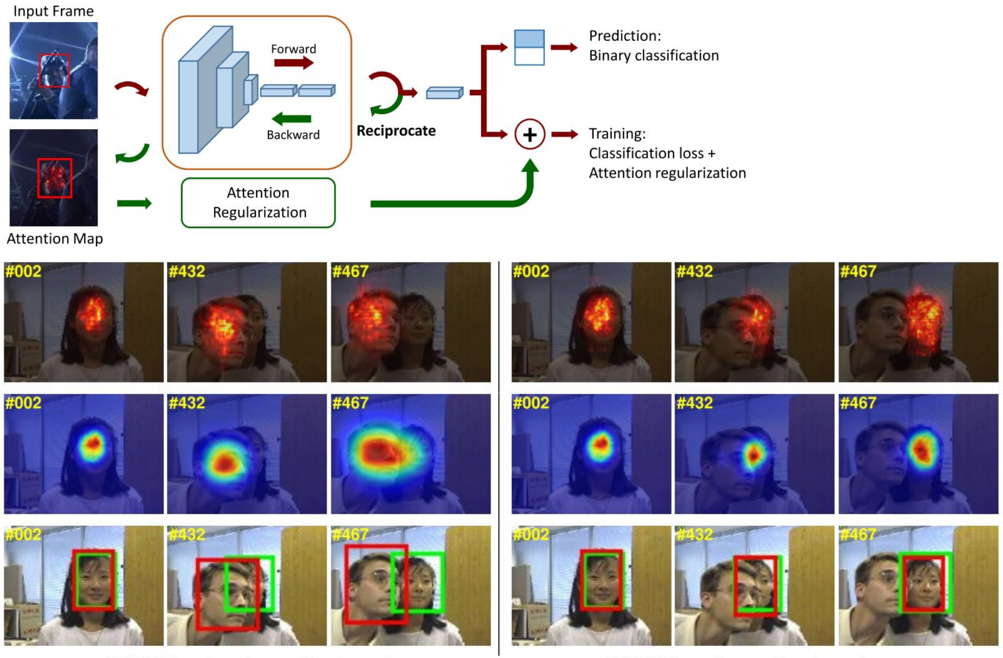
Finding attention-maps, by gradient.

Loss basically says:

Put high importance of features inside box (target)

Forcing the network to distribute attention to all regions of the object.

Not only tracking object by some key feature.

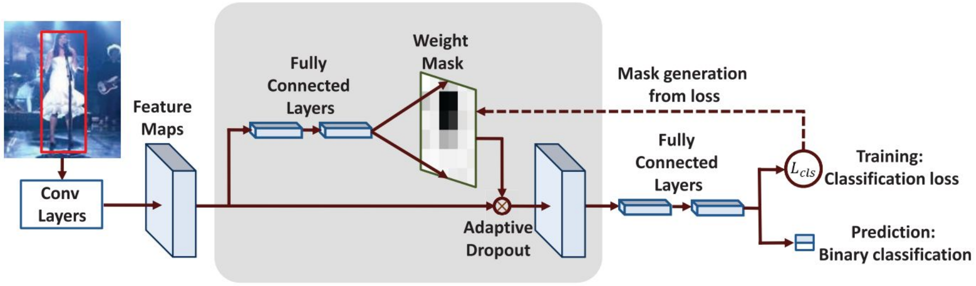


(a) Without reciprocative learning (b) With reciprocative learning

[Deep Attentive Tracking via Reciprocative Learning]

VITAL: VIsual Tracking via Adversarial Learning

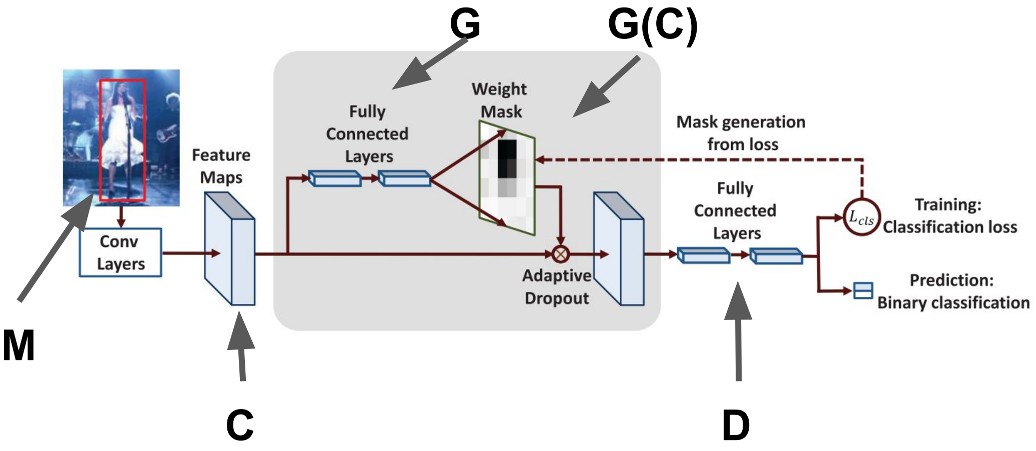
A different, but similar way to direct focus.



[VITAL: VIsual Tracking via Adversarial Learning]

VITAL: VIsual Tracking via Adversarial Learning

A different, but similar way to direct focus.



$$\mathcal{L}_{\text{VITAL}} = \min_G \max_D \; \mathbb{E}_{(C,M)\backsim P_{(C,M)}}[\log D(M \cdot C)] + \mathbb{E}_{C\backsim P_{(C)}}[\log(1 - D(G(C) \cdot C))] + \lambda \mathbb{E}_{(C,M)\backsim P_{(C,M)}} ||G(C) - M||^2,$$

[VITAL: VIsual Tracking via Adversarial Learning]

VITAL: VIsual Tracking via Adversarial Learning

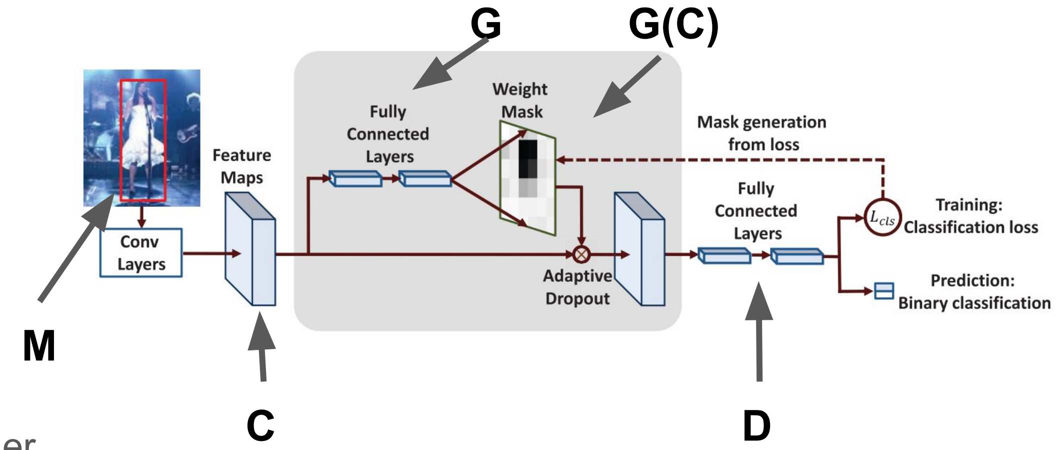
A different, but similar way to direct focus.

Loss is basically saying:

During training, remove features that are important for classification, but keep less relevant features, inside the mask.

Forcing network to learn tracking with harder features.

Masking is turned off during tracking.



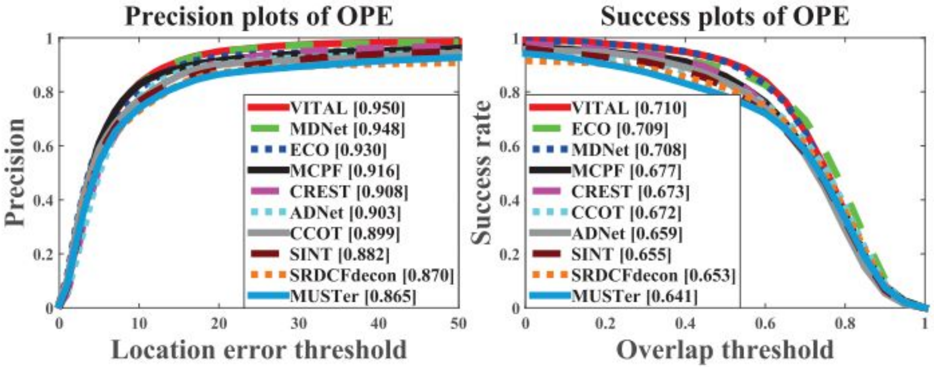
$$\mathcal{L}_{\mathrm{VITAL}} = \min_G \max_D \; \mathbb{E}_{(C,M)\backsim P_{(C,M)}}[\log D(M \cdot C)] + \mathbb{E}_{C\backsim P_{(C)}}[\log(1 - D(G(C)\cdot C))] + \lambda \mathbb{E}_{(C,M)\backsim P_{(C,M)}} ||G(C) - M||^2,$$

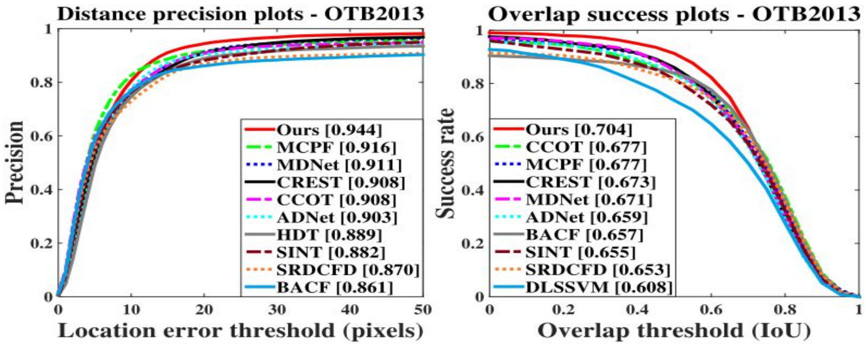
[VITAL: VIsual Tracking via Adversarial Learning]

Results - changing focus for MDNet

Results for VITAL and Reciprocal learning, on OTB-2013 (vital red on top)

Vital has best results, but reciprocal learning have an interesting point on mixing of similar objects.



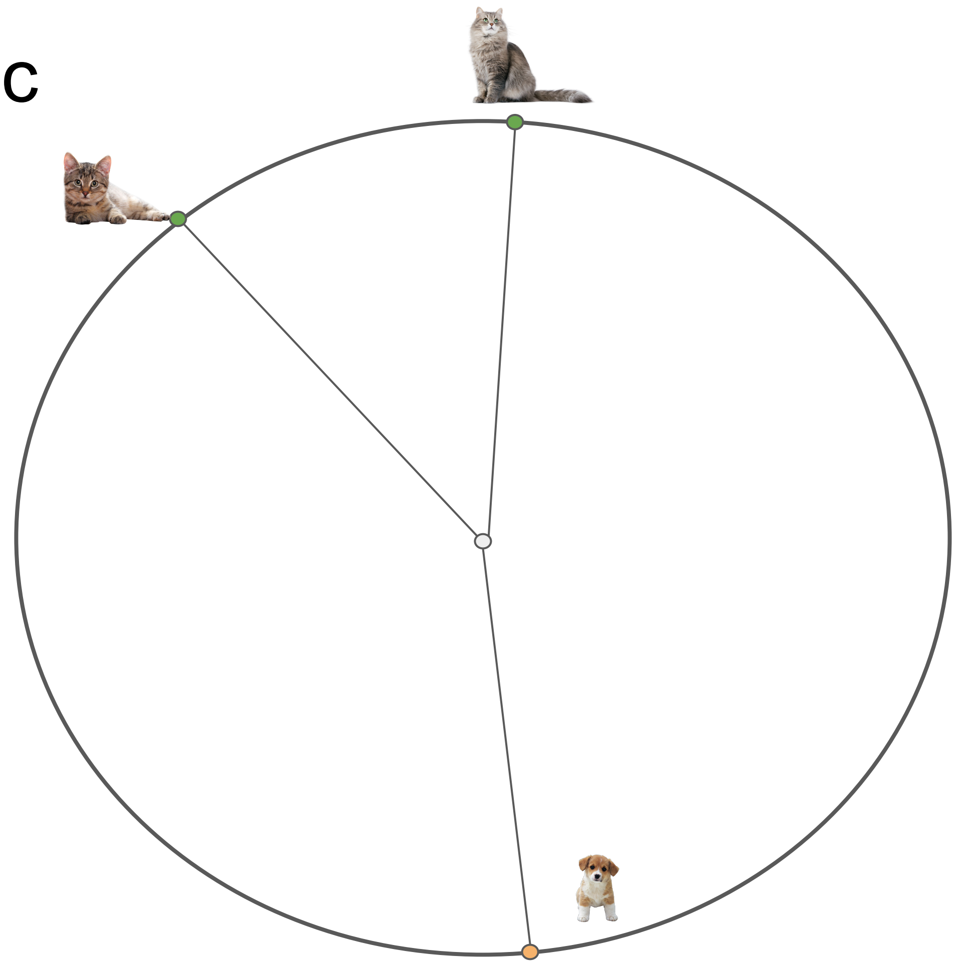


Matching based tracking

Learning distance metric

Learning to keep similar data close and different data far away.

You choose similarities...



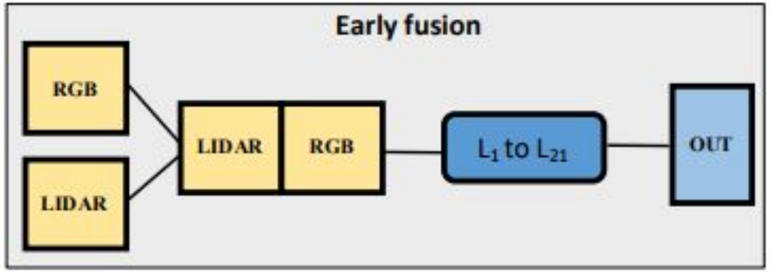
Learning distance metric

The easy solution?

Input channel wise.

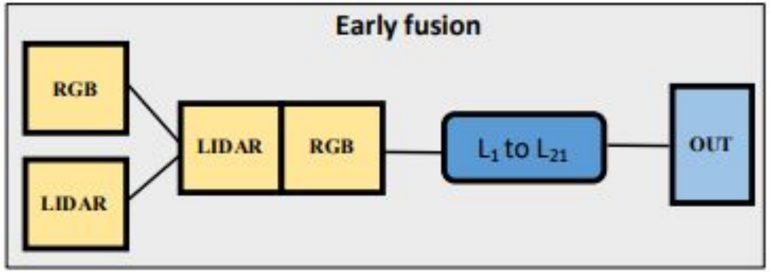
Give high value if different and low value if similar.

A viable solution.



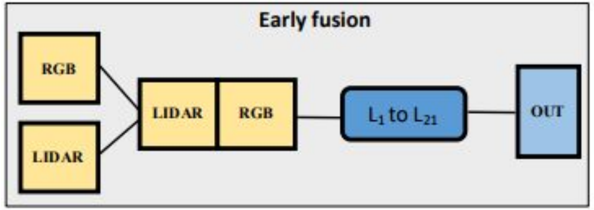
Learning distance metric

Remember concatenating channels from segmentation lecture...



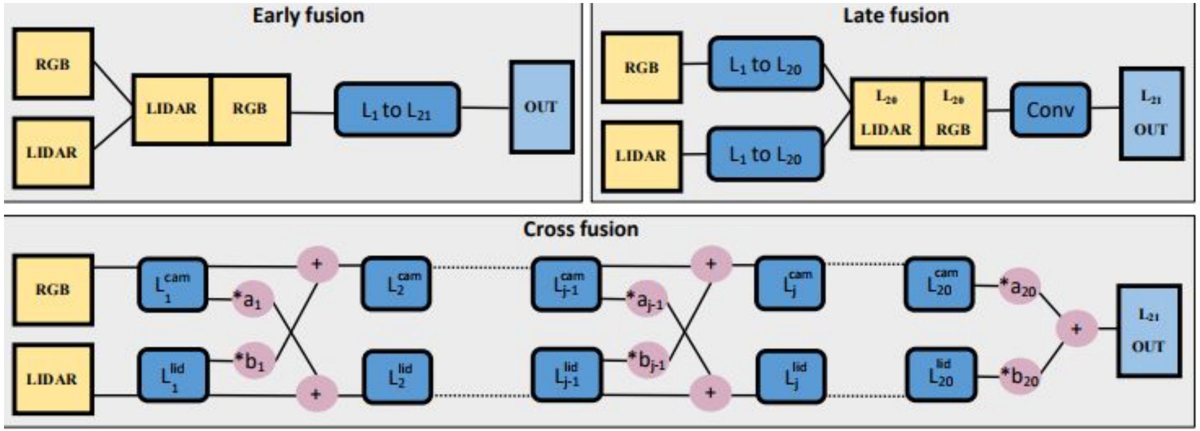
Learning distance metric

Mismatch in spatial domain can cause problems.



Learning distance metric

Mismatch in spatial domain can cause problems.



Learning distance metric - siamese networks

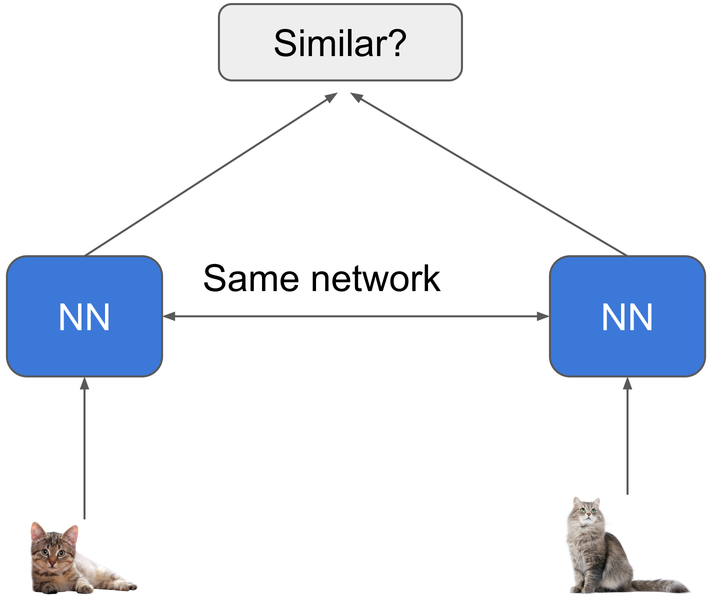
Loss eg.

- $y \, ||f(\mathbf{x}_1) - f(\mathbf{x}_2)||^2$
- $y \, f(\mathbf{x}_1)^T f(\mathbf{x}_2)$

Where y = 1 for similar samples and y = -1 for different samples

Fun fact: used for check signature verification in 1994

Signature verification using a" **siamese**" time delay neural **network**



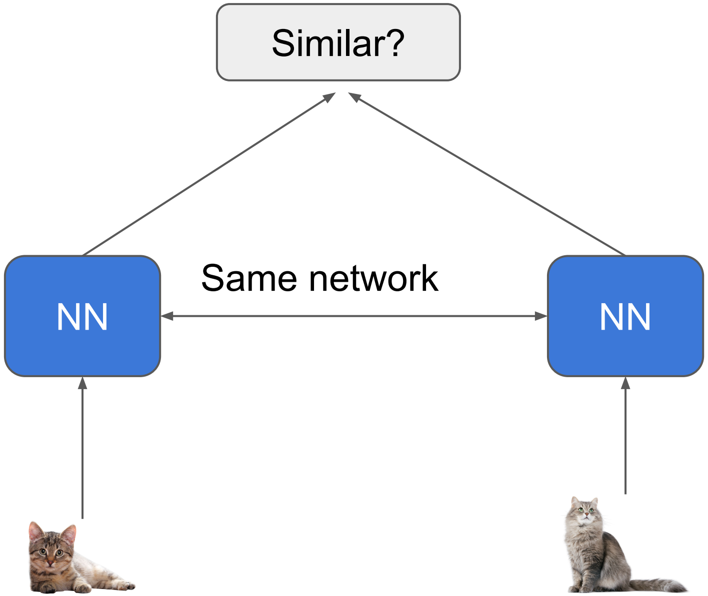
Learning distance metric - siamese networks

You don’t need to run the networks at the same time.

One representation can be stored as the output of a network. 80 bits in 1994

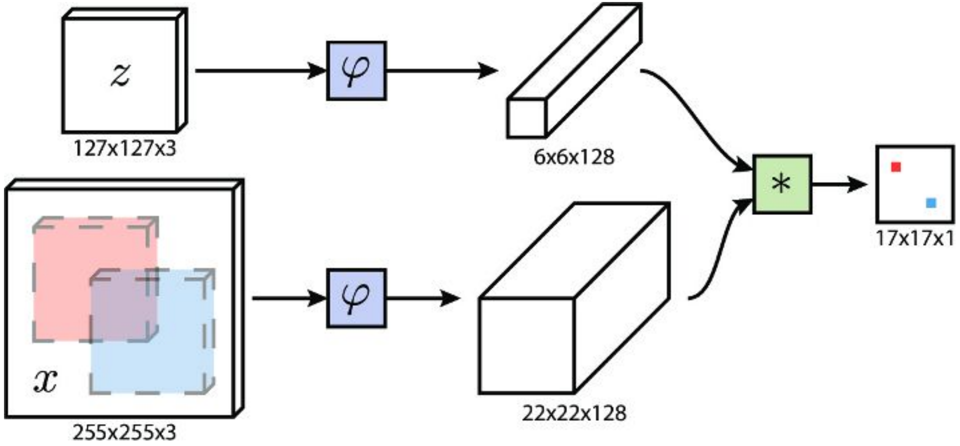
Checking can be done quickly

Signature verification using a" **siamese**" time delay neural **network**



Fully-Convolutional Siamese Networks for Object Tracking (SiamFC)

- Run a target image through your network
  - Crop and scale the bounding box
- Run a search image through your network
  - This output image should be larger
- Convolve/correlate the output patches
  - Is basically the same as taking the inner product for each position



Fully-Convolutional Siamese Networks for Object Tracking

$$\ell(y, v) = \log(1 + \exp(-yv))$$

SiamFC

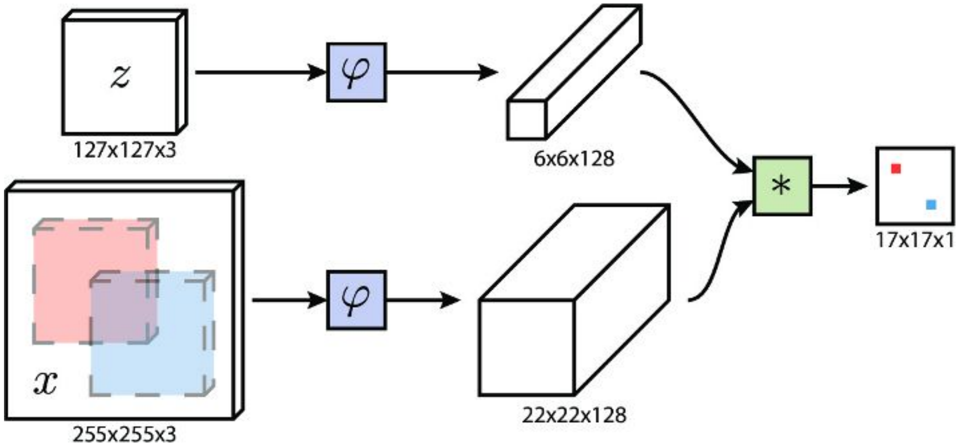
Optimizing:

$$\ell(y, v) = \log(1 + \exp(-yv))$$

Where v is the output response map (inner product). Not critical as other implementations use other loss, e.g. some weight regularization can be wise...

$$\arg\min_{w} \quad \frac{1}{2n}\|w \star x - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

End-to-end representation learning for Correlation Filter based tracking

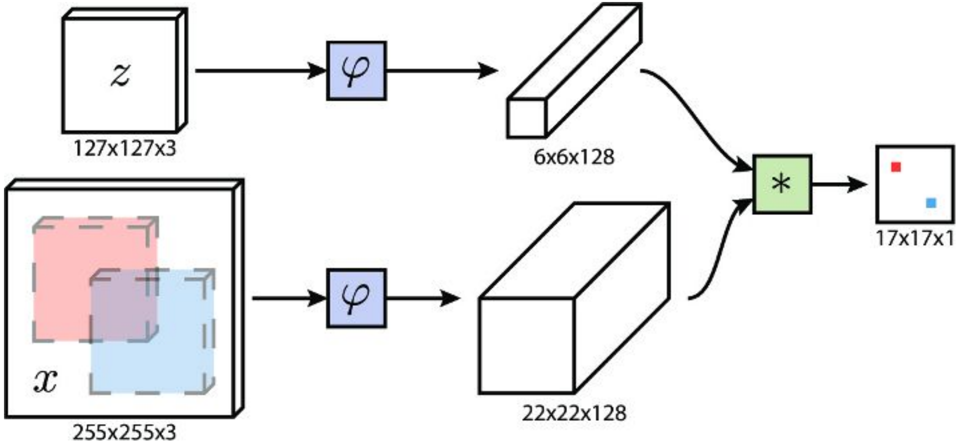


Fully-Convolutional Siamese Networks for Object Tracking

Training SiamFC

Pairs from one video sequence is sample randomly

An important aspect of training SiamFC is to utilize all the “negative regions”.



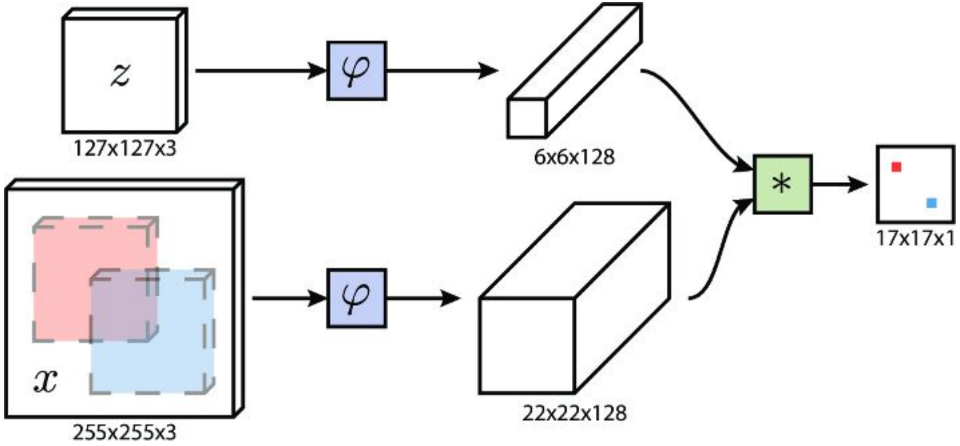
[Fully-Convolutional Siamese Networks for Object Tracking]

Training SiamFC

Pairs from one video sequence is sample randomly

An important aspect of training SiamFC is to utilize all the “negative regions”.

It may be unwise to just select the true position as positive, since the surrounding responses is heavily influenced by the tracked object.



Fully-Convolutional Siamese Networks for Object Tracking

Training SiamFC

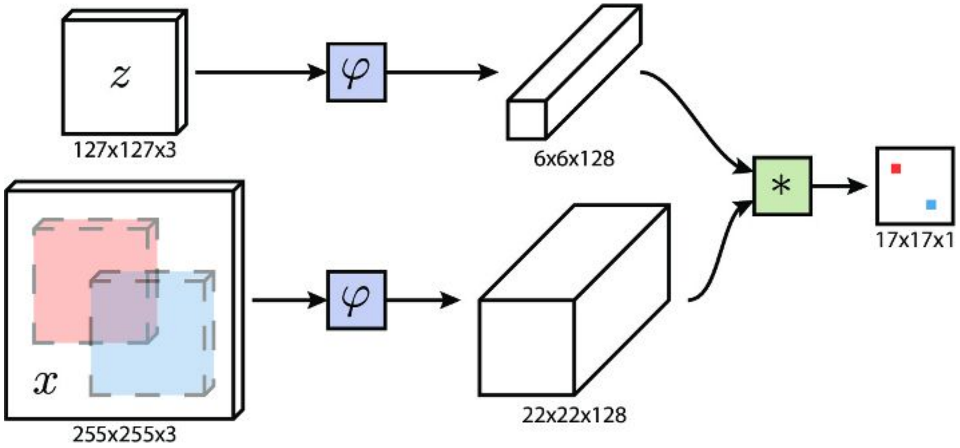
Pairs from one video sequence is sample randomly

An important aspect of training SiamFC is to utilize all the “negative regions”.

It may be unwise to just select the true position as positive, since the surrounding responses is heavily influenced by the tracked object.

A region corresponding to 16 pixels within the input image, is selected as positive and remaining pixels negative.

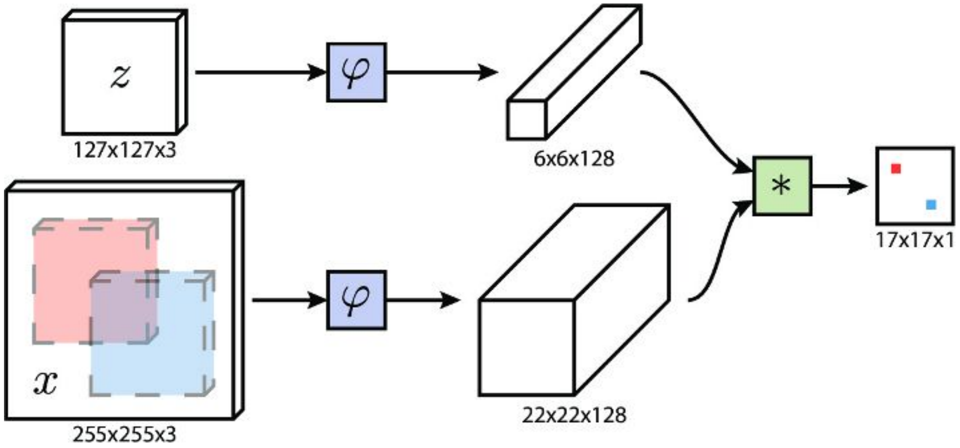
The loss is scaled to account for unbalanced classes.



Fully-Convolutional Siamese Networks for Object Tracking

Running SiamFC

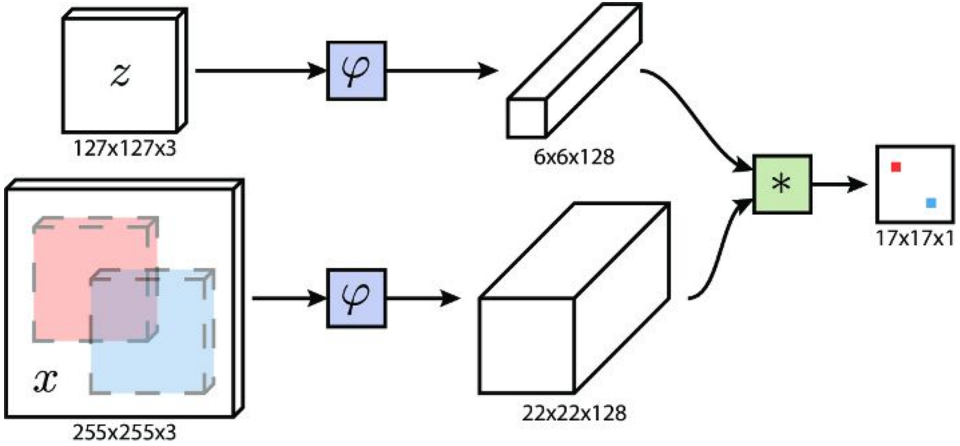
1. Find Z
   a. Run the target patch through the network and get a Z (6x6x128)



Fully-Convolutional Siamese Networks for Object Tracking

Running SiamFC

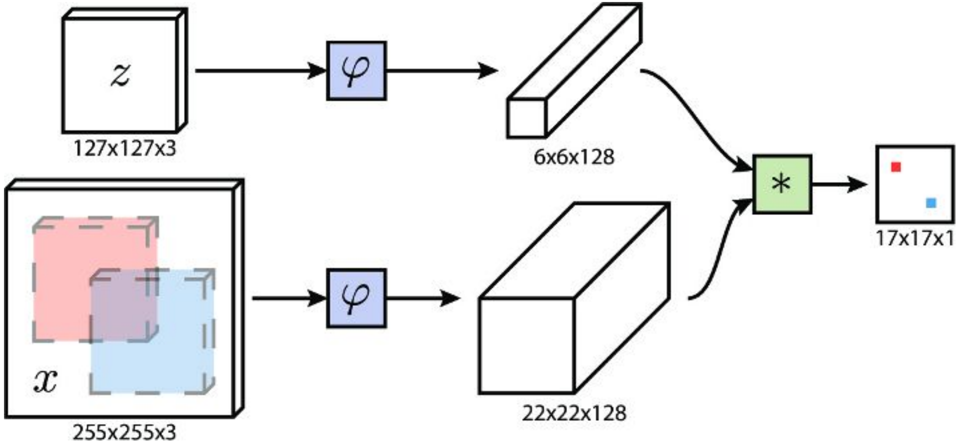
1. Find Z (6x6x128)
2. Find search region
   a. In the next image you extract a search patch around the expected center
   b. Padding is applied to ensure correct aspect ratio
   c. Add extra area around the expected center, proportional to the last bounding box
   d. Re-scale your image to 3 different sizes (1 original size)



[Fully-Convolutional Siamese Networks for Object Tracking]

Running SiamFC

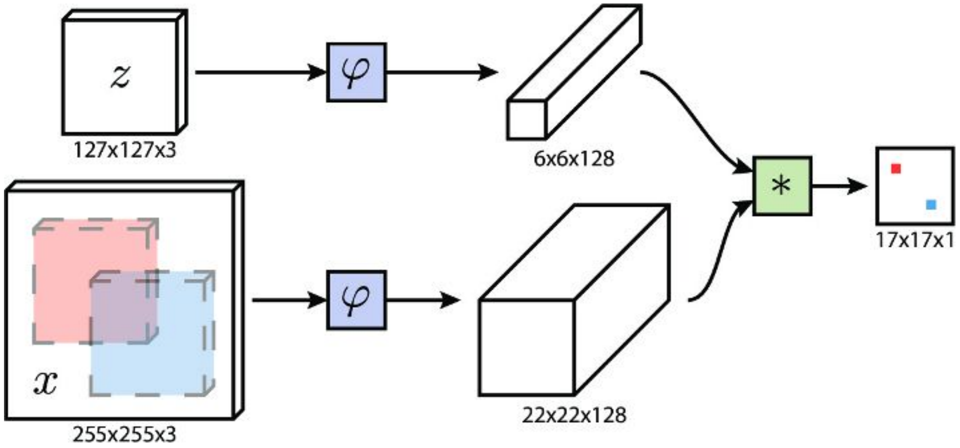
1. Z (6x6x128)
2. Find search region
3. Find max response location
    a. Run all 3 patches through the network and correlate with target Z
    b. Find the maximum response, both spatially and in scale.



[Fully-Convolutional Siamese Networks for Object Tracking](#)

Running SiamFC

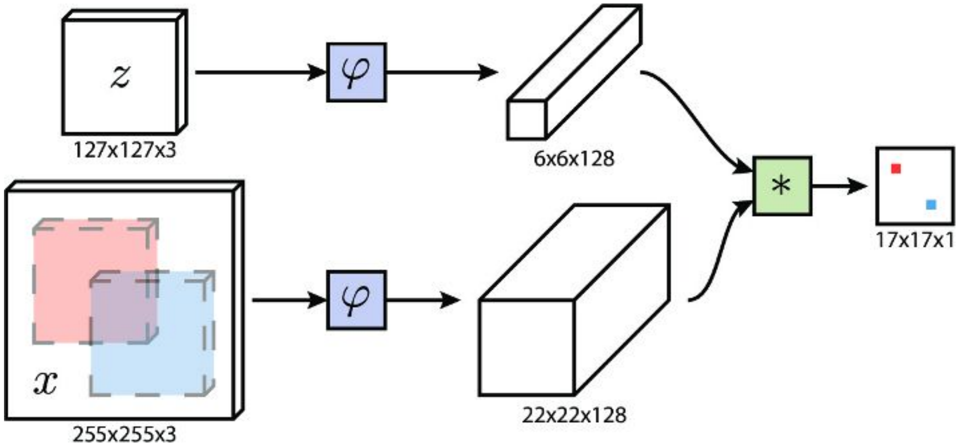
1. Z (6x6x128)
2. Find search region
3. Find max response location
4. Move track location
    a. Move the tracked location (next search region) to the area corresponding to maximum score.
    b. Scale corresponding to scale of maximum response patch
    c. You get a pixel delta, but need to rescale to input image
    d. Applying an additional cost to moving large distances can be beneficial



[Fully-Convolutional Siamese Networks for Object Tracking]

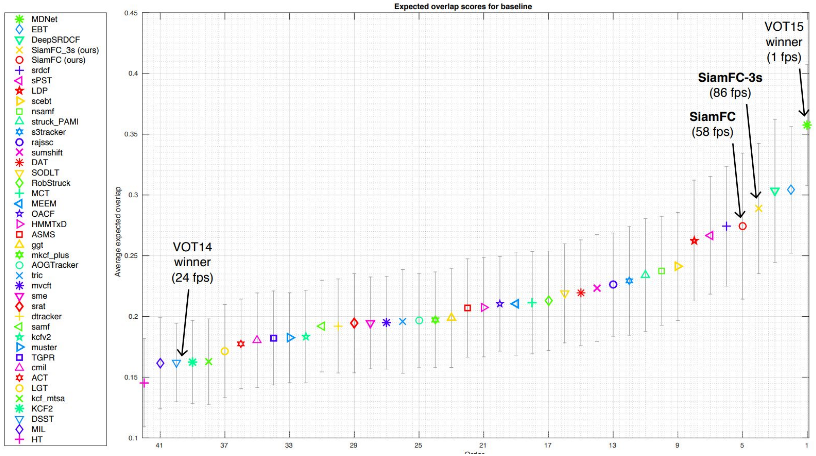
Running SiamFC

1. Z (6x6x128)
2. Find search region
3. Find max response location
4. Move track location
5. Update Z
    a. Update Z if confident
    b. Update with exponential average
    c. In long term tracking this may be less beneficial



[Fully-Convolutional Siamese Networks for Object Tracking]

SiamFC - Results



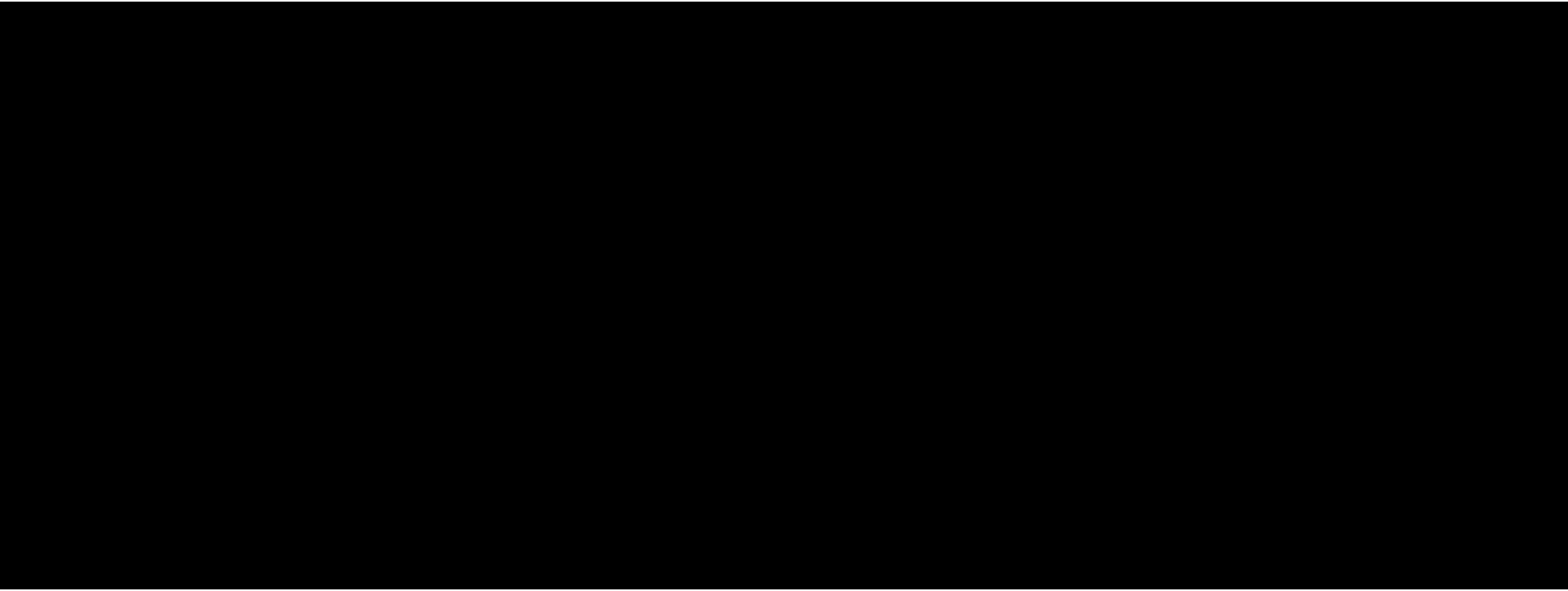
		OTB-2013				OTB-50				OTB-100			
		OPE		TRE		OPE		TRE		OPE		TRE	
Method	**speed** (fps.)	IoU	prec.	IoU	prec.	IoU	prec.	IoU	prec.	IoU	prec.	IoU	prec.
CFNet-conv1	83	57.8	71.4	58.6	71.7	48.8	61.3	51.0	63.6	53.6	65.8	55.9	67.6
CFNet-conv2	75	61.1	74.6	**64.0**	**77.9**	53.0	66.0	56.5	70.2	56.8	69.3	60.6	73.2
Baseline+CF-conv3	67	61.0	74.8	63.1	76.8	53.8	66.5	**57.4**	**70.8**	**58.9**	71.1	61.1	73.4
CFNet-conv5	43	61.1	73.6	62.6	75.7	**53.9**	**67.0**	56.6	70.1	58.6	71.1	60.8	72.7
Baseline-conv5	52	**61.8**	75.3	**64.0**	77.3	51.7	64.1	56.1	69.1	58.8	71.4	**61.6**	**73.7**
SiamFC-3s [3]		60.7	73.5	61.8	75.0	51.6	63.9	55.5	69.2	58.2	70.2	60.5	72.8
Staple [2]		60.0	72.5	61.7	74.2	50.9	63.4	54.1	67.5	58.1	**71.6**	60.4	72.8
LCT [22]		61.2	**78.0**	59.4	74.2	49.2	62.5	49.5	61.7	56.2	69.2	56.9	68.2
SAMF [18]		–	–	–	–	46.2	60.7	51.4	65.6	53.9	69.0	57.7	71.4
DSST [6]		55.4	67.5	56.6	68.4	45.2	56.6	48.4	60.1	51.3	63.1	–	–

Good framerate can in practise give much better results

[End-to-end representation learning for Correlation Filter based tracking]

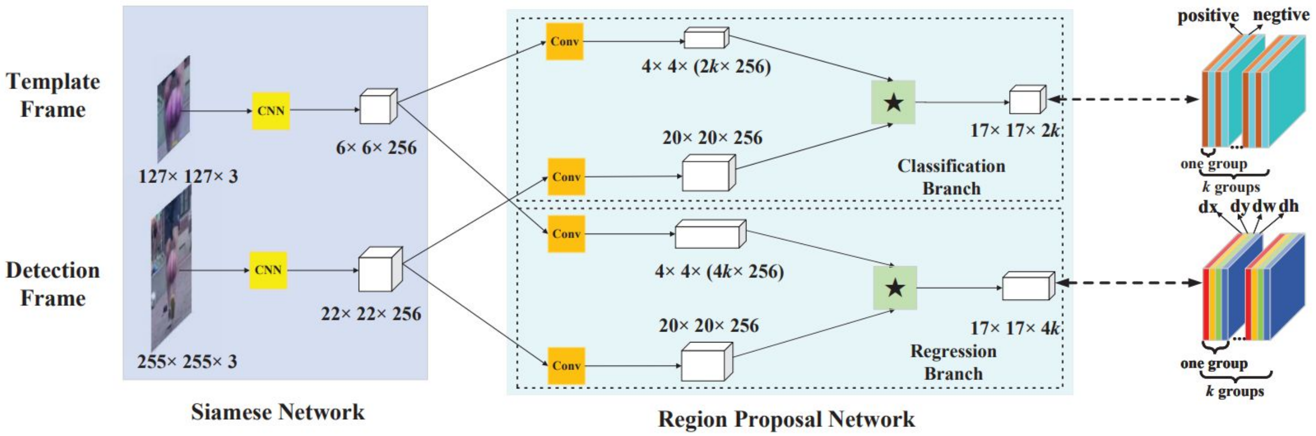
[Fully-Convolutional Siamese Networks for Object Tracking]

SiamFC response map



SiamFC additions - SiamRPN

Instead of running 3-5 different sized images, run a regression network



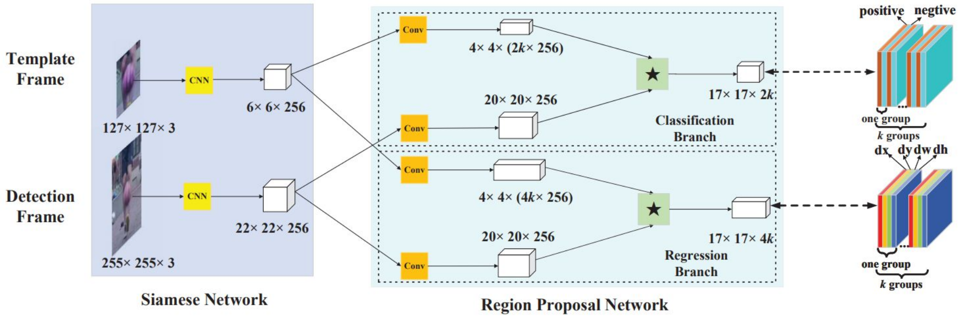
[High Performance Visual Tracking with Siamese Region Proposal Network]

SiamFC additions - SiamRPN

Instead of running 3-5 different sized images, run a regression network

Same loss as Faster RCNN. Softmax cross-entropy for classification.

Smooth L1 for box coordinates



$$\delta[0] = \frac{T_x - A_x}{A_w}, \quad \delta[1] = \frac{T_y - A_y}{A_h}$$
$$\delta[2] = ln\frac{T_w}{A_w}, \quad \delta[3] = ln\frac{T_h}{A_h}$$

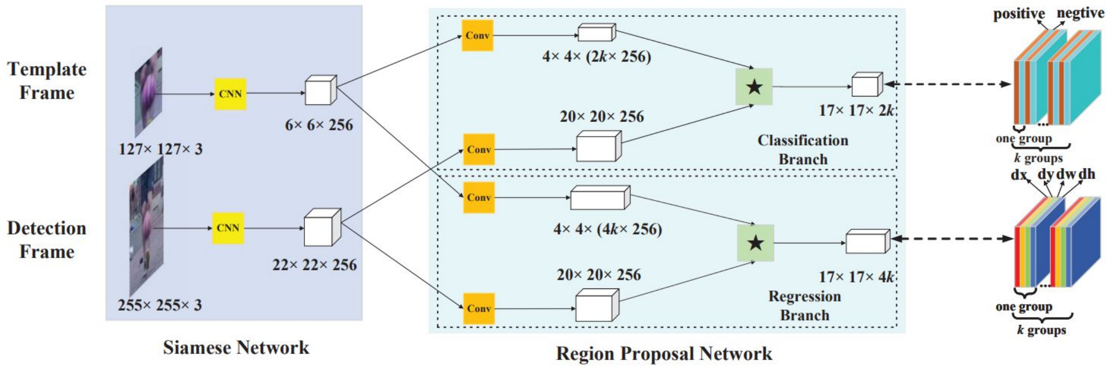
[High Performance Visual Tracking with Siamese Region Proposal Network]

$$L_{reg} = \sum_{i=0}^{3} smooth_{L1}(\delta[i], \sigma)$$

$$smooth_{L_1}(x, \sigma) = \begin{cases} 0.5\sigma^2 x^2, & |x| < \frac{1}{\sigma^2} \\ |x| - \frac{1}{2\sigma^2}, & |x| \geq \frac{1}{\sigma^2} \end{cases}$$

Training SiamRPN

- Use affine transformation on data to improve regression network
- More robust to rotation and scale changes

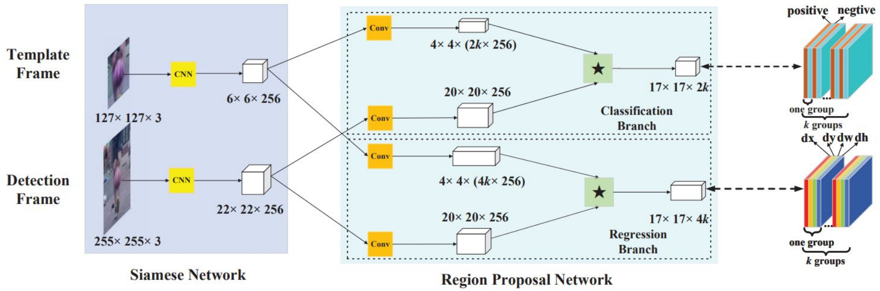


[High Performance Visual Tracking with Siamese Region Proposal Network]()

Running SiamRPN

Select K highest scores

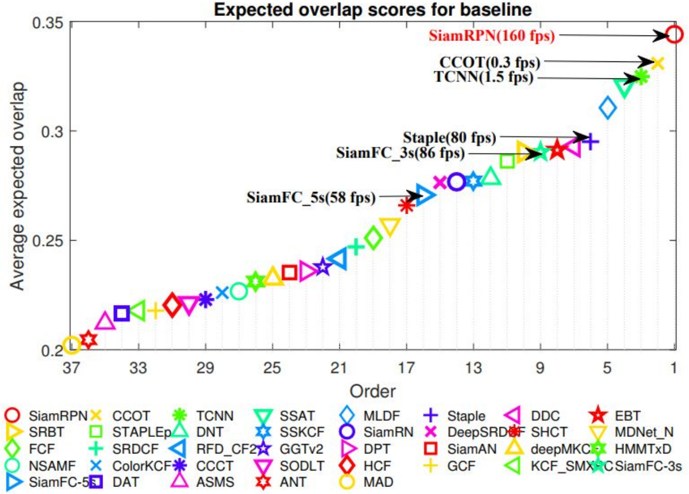
1. Use the confidence score from the classification network
2. Add a windowed penalty term (cosine window) to discurage large leapes in size, shape and posistion
3. Choose the regression box at the max-confidence posistion when accounting for penalty
4. No online adaption

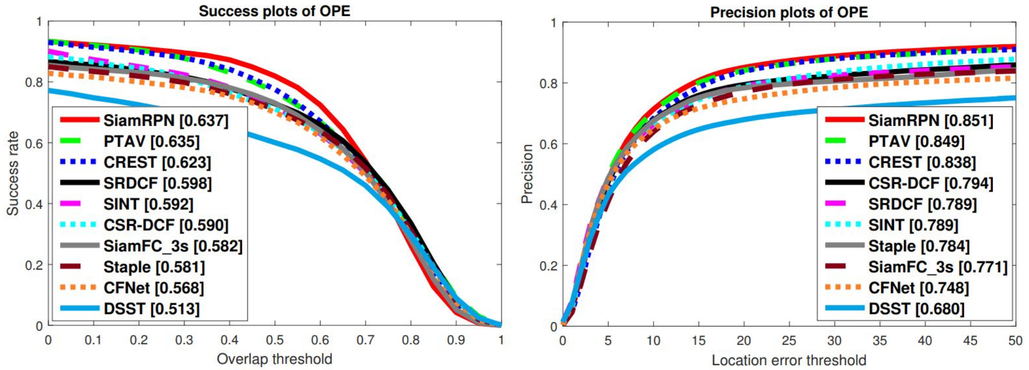


[High Performance Visual Tracking with Siamese Region Proposal Network]

SiamRPN - Results

160 fps on GTX 1060



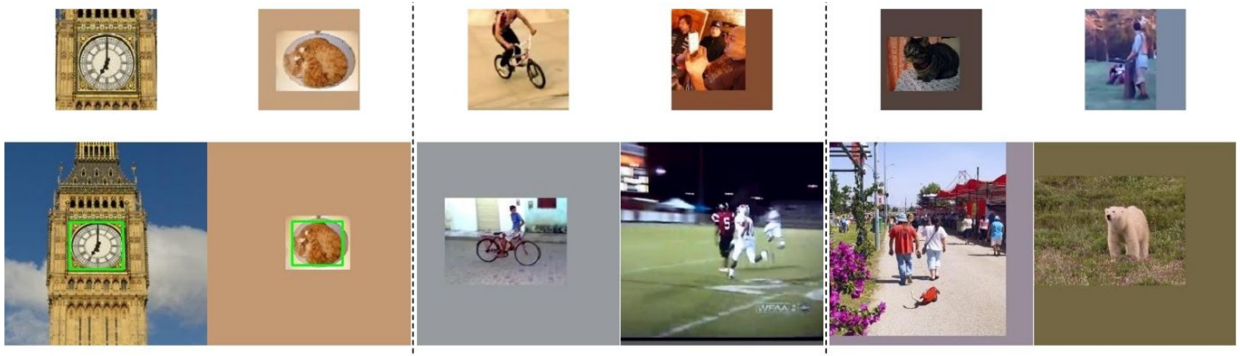


[High Performance Visual Tracking with Siamese Region Proposal Network]()

SiamFC additions - Distraction-training SiamRPN

Dataset contains few classes and background is often trivial.

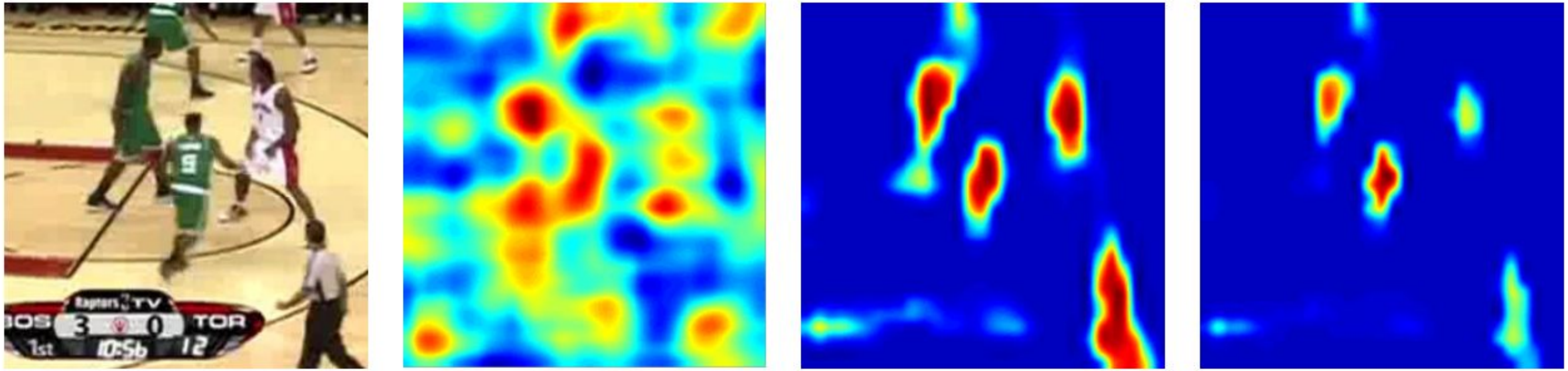
1. More categories
    a. Same-image augmentation
    b. Afiine transforms, motion blur, illumination
2. Semantic negative pairs
    a. Sampling objects from different sequences
    b. Sampling from same class



(a) detection pairs (b) negative pairs from the same categories (c) negative pairs from different categories

[Distractor-aware Siamese Networks for Visual Object Tracking]()

Response maps after distraction training



(a) ROI (b) SiamFC (c) SiamRPN (d) SiamRPN+

Distractor-aware Siamese Networks for Visual Object Tracking

Distraction-aware SiamRPN

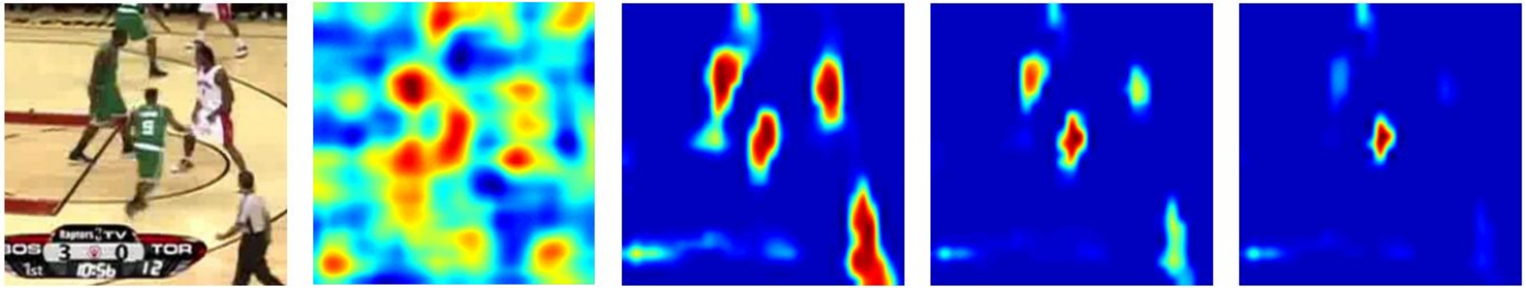
1. After each iteration, choose K other highes as distractors
2. Choose the response that match well with your target and less well with the distractors
   a. A person in a similar pose as Z, may give a higher score initially

$$q = \underset{p_k \in \mathcal{P}}{argmax} \;\; f(z, p_k) - \frac{\hat{\alpha} \sum_{i=1}^{n} \alpha_i f(d_i, p_k)}{\sum_{i=1}^{n} \alpha_i}$$

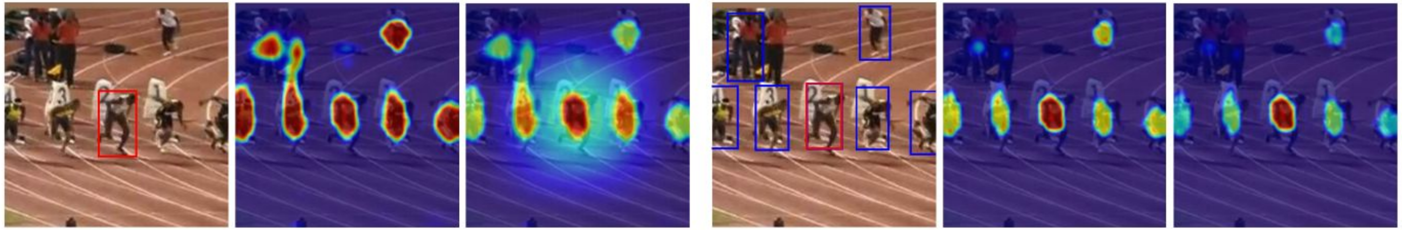
$$q = \underset{p_k \in \mathcal{P}}{argmax} \;\; (\varphi(z) - \frac{\hat{\alpha} \sum_{i=1}^{n} \alpha_i \varphi(d_i)}{\sum_{i=1}^{n} \alpha_i}) \star \varphi(p_k)$$

Distractor-aware Siamese Networks for Visual Object Tracking

Distraction-aware SiamRPN



(a) ROI (b) SiamFC (c) SiamRPN (d) SiamRPN+ (e) Ours



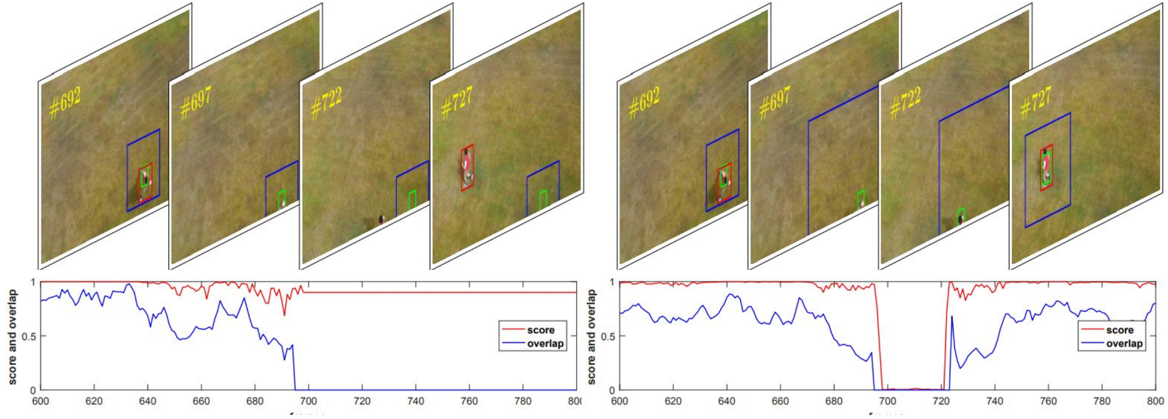
(a) General Siamese tracker (b) Distractor-aware Siamese tracker

[Distractor-aware Siamese Networks for Visual Object Tracking]()

Distraction-aware SiamRPN for long term tracking

Distraction aware training and inference give accurate score values.

When score is low, gradually increase the search region til it covers the whole image.



[Distractor-aware Siamese Networks for Visual Object Tracking]

Distraction-aware SiamRPN

Long term tracking give 110 FPS on TITAN X

Winner of ECCV 2018 Real-time Visual Object Tracking Challenge

Second place for ECCV 2018 Long-term Visual Object Tracking Challenge

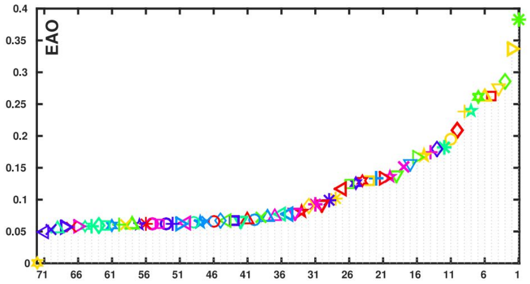


Fig. 5: The EAO plot (right) for the realtime experiment.

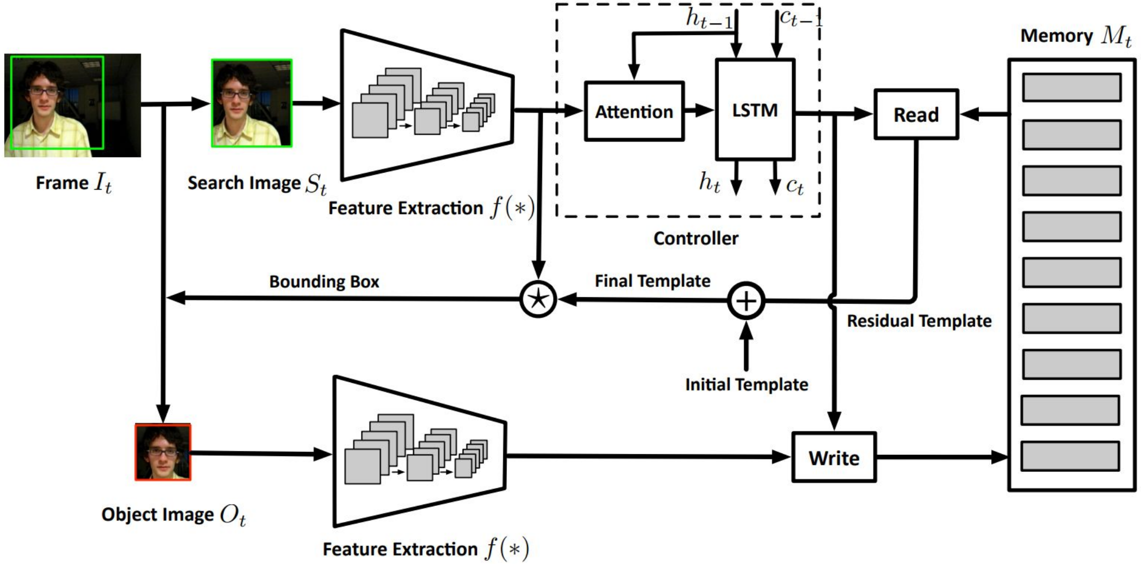
Trackers	OTB-2015		VOT2015			VOT2016			VOT2017			FPS
	OP	DP	A	R	EAO	A	R	EAO	A	R	EAO	
SiamFC	73.0	77.0	0.533	0.88	0.289	0.53	0.46	0.235	0.50	0.59	0.188	86
CFNet	69.9	74.7	-	-	-	-	-	-	-	-	-	75
Staple	70.9	78.4	0.57	1.39	0.300	0.54	0.38	0.295	*0.52*	0.69	0.169	80
CSRDCF	70.7	78.7	0.56	0.86	0.320	0.51	0.24	0.338	0.49	0.36	0.256	13
BACF	76.7	81.5	0.59	1.56	-	-	-	-	-	-	-	35
ECO-HC	78.4	85.6	-	-	-	0.54	0.30	0.322	0.49	0.44	0.238	60
CREST	77.5	83.7	-	-	-	0.51	0.25	0.283	-	-	-	1
MDNet	*85.4*	*90.9*	*0.60*	*0.69*	*0.378*	0.54	0.34	0.257	-	-	-	1
C-COT	82.0	89.8	0.54	0.82	0.303	0.54	0.24	0.331	0.49	*0.32*	0.267	0.3
ECO	84.9	**91.0**	-	-	-	0.55	**0.20**	*0.375*	0.48	**0.27**	*0.280*	8
SiamRPN	81.9	85.0	0.58	1.13	0.349	*0.56*	0.26	0.344	0.49	0.46	0.244	**200**
Ours	**86.5**	88.0	**0.63**	**0.66**	**0.446**	**0.61**	*0.22*	**0.411**	**0.56**	0.34	**0.326**	*160*

Distractor-aware Siamese Networks for Visual Object Tracking

Addition to SiamFC - Memory bank

Adding a memory network to SiamFC

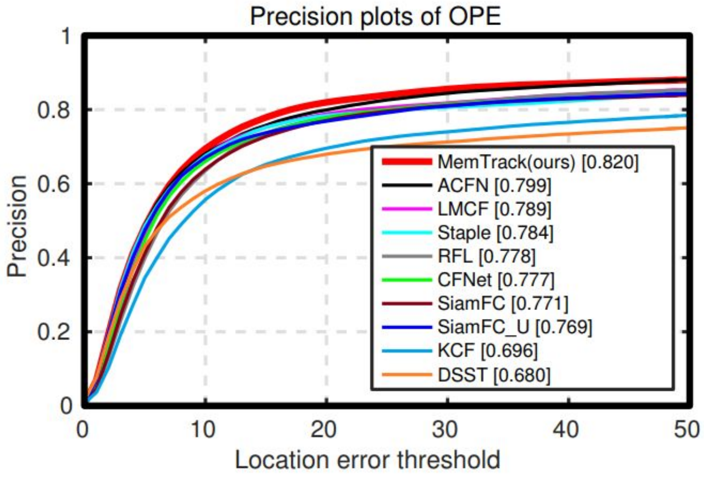
- Learns different representations of objects
- Exponential Average can mess templates up…
- Train with reinforcement learning
- 50 FPS

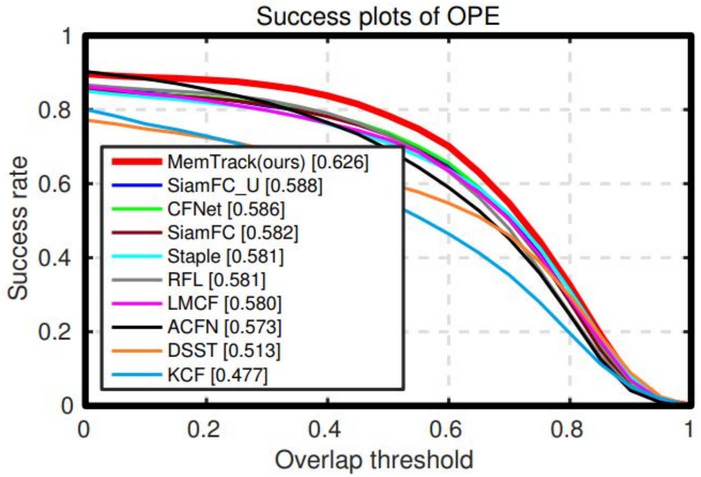


[Learning Dynamic Memory Networks for Object Tracking]

Learning Dynamic Memory Networks for Object Tracking

Third place for ECCV 2018 Long-term Visual
Object Tracking Challenge

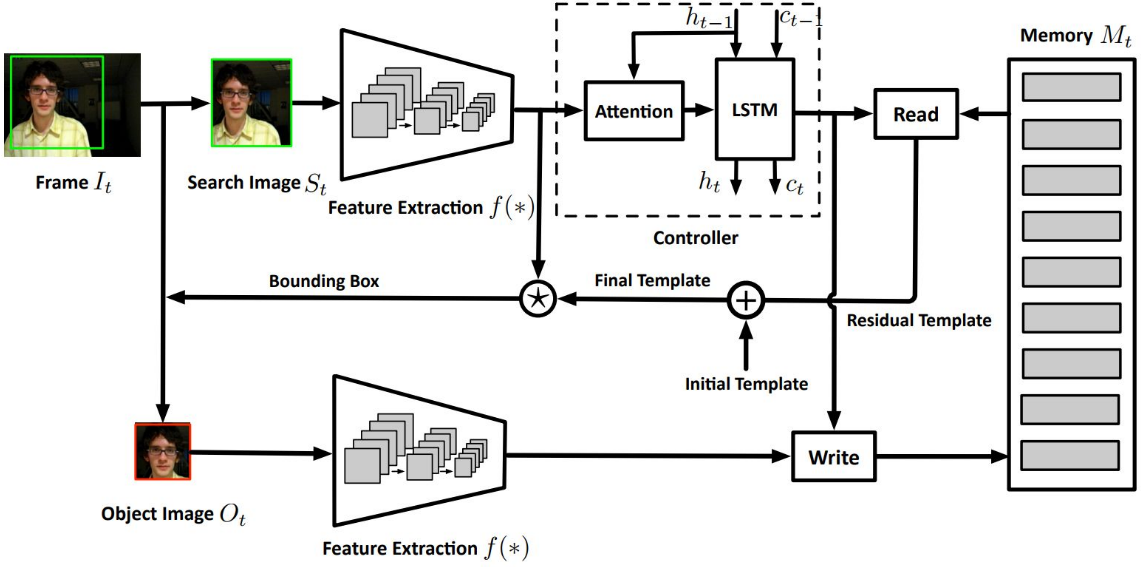




[Learning Dynamic Memory Networks for Object Tracking]

Learning Dynamic Memory Networks for Object Tracking

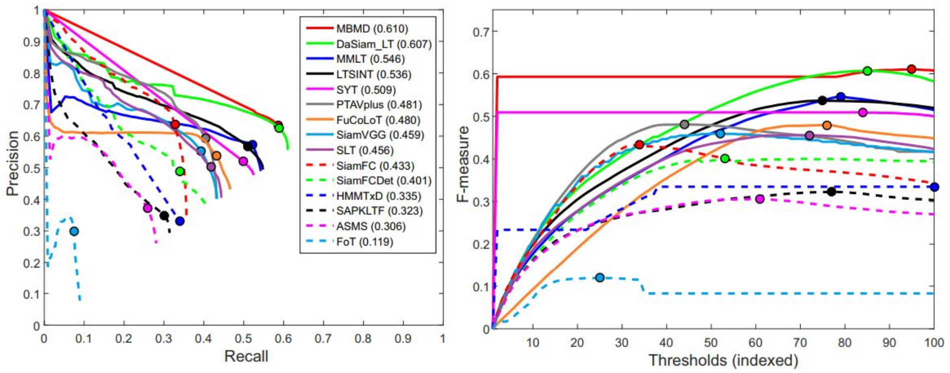
Can easily be combined with Distraction-Aware SiamRPN



[Learning Dynamic Memory Networks for Object Tracking]()

ECCV Visual Object Tracking Challenge 2018

- Winners of Long-term tracking and non-realtime tracking are similar/based to MDNet
- Winner of non-realtime tracking seems like a monster, running multiple deep nets etc.
- Slow but effective
- Matching based trackers are fast, and close in performance



Overview

- Transition based tracking
  - fast
  - easily utilise history
  - can be added to other methods
- Online-learning based methods
  - Often slow
  - Very accurate
  - State-of-art without realtime requiremets
- Matching based methods
  - Fast
  - Accurate
  - Are they as general?