FCNN:
Fusion of Convolutional Neural Networks for Object Tracking

VGIS 842
Cédric Magnan
Louise Abela
Péter Rohoska

VGIS Exam 2017

Agenda

- Introduction & Initial Problem
- Previous Work (MDNet &TCNN)
- Fusion Theory & Architecture
- FCNN (Our tracker) Implementation
- Final Results
- Discussion, Future Work & Conclusion

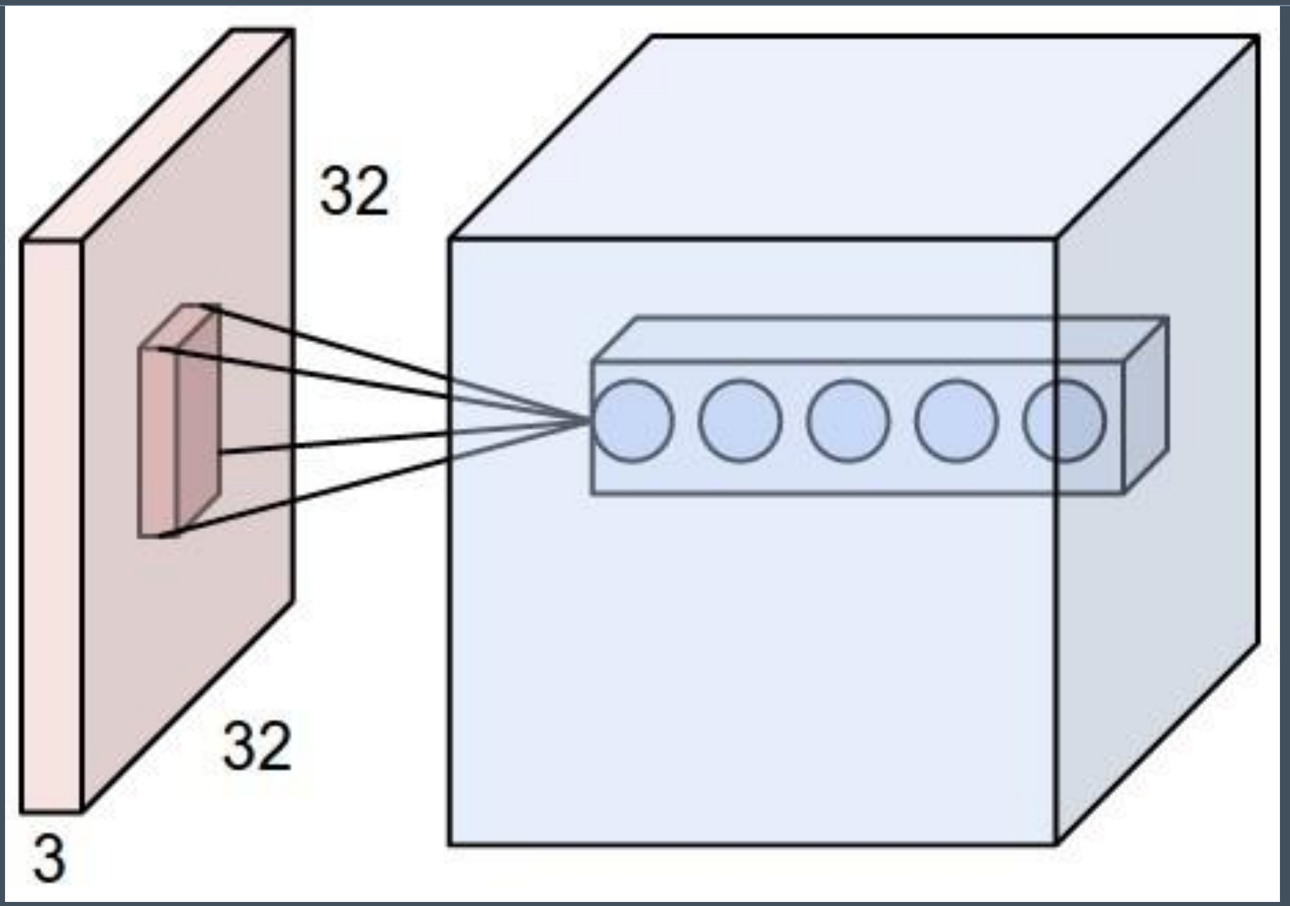
Object Tracking

Object tracking is a fundamental task in CV (Atonomous driving systems)

Trend shifted towards CNNs

SoTA algorithms can still be improved under different imaging conditions

Individual trackers handle the conditions differently



CNNs

Handle local structures (visual data)

Temporal

Can force the extraction of local features

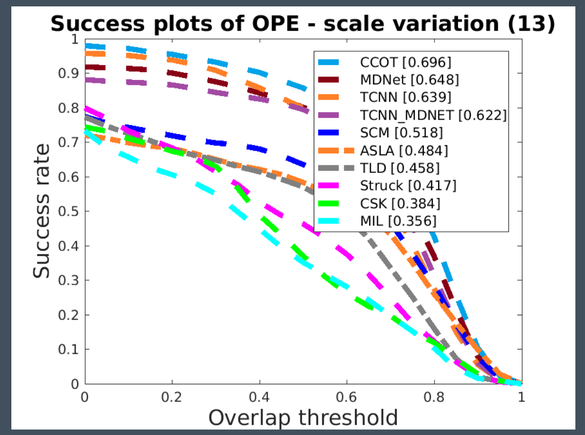
Best under imaging conditions

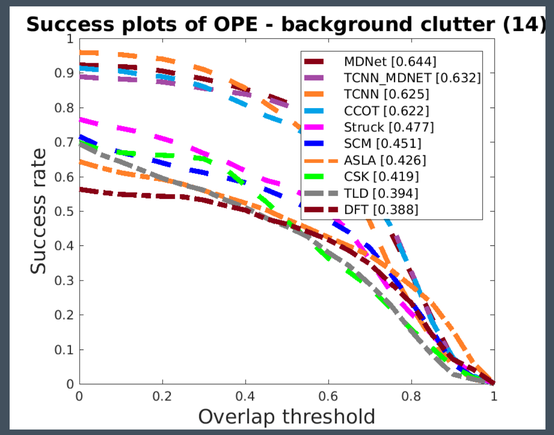
Problem Formulation

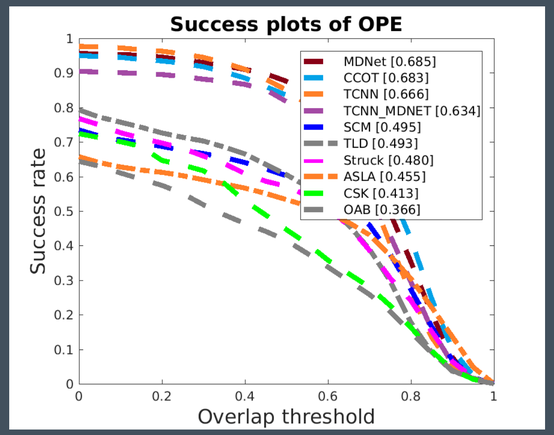
BC | SV | OCC | IV

LR | DEF | MB | FM

OV | IPR | OPR







OTB-50

Problem Formulation

Tracker	AUC	Precision	CNN
MDNet	0.708	0.948	Yes
TCNN	0.682	0.937	Yes
C-COT	0.673	0.899	Yes
DNT	0.664	0.907	Yes
SINT+	0.655	0.882	Yes
SRDCFdecon	0.653	0.87	No
DeepSRDCF	0.641	0.849	Yes
MUSTer	0.641	0.865	Yes
SINT	0.635	0.851	Yes

Name	First	Second	Third
Overall	MDNet (0.695)	TCNN (0.677)	CCOT (0.673)
IV	MDNet (0.658)	TCNN (0.647)	CCOT (0.635)
SV	MDNet (0.676)	CCOT (0.669)	TCNN (0.663)
OCC	CCOT (0.693)	MDNet (0.668)	TCNN (0.658)
DEF	MDNet (0.694)	TCNN (0.676)	CCOT (0.637)
MB	CCOT (0.649)	MDNet (0.646)	TCNN (0.642)
FM	CCOT (0.640)	MDNet (0.634)	TCNN (0.622)
IPR	MDNet (0.676)	TCNN (0.662)	CCOT (0.633)
OPR	MDNet (0.681)	TCNN (0.666)	CCOT (0.662)
OV	CCOT (0.733)	MDNet (0.689)	TCNN (0.637)
BC	MDNet (0.651)	TCNN (0.635)	CCOT (0.601)
LR	MDNet (0.627)	TCNN (0.620)	CCOT (0.569)

Success and precision results on OTB-50.

Trackers

Theory and Re-Implementation

MDNet

Multi-Domain training

Features from shared layers represent domain independent information

VGG-M on Imagenet

TCNN

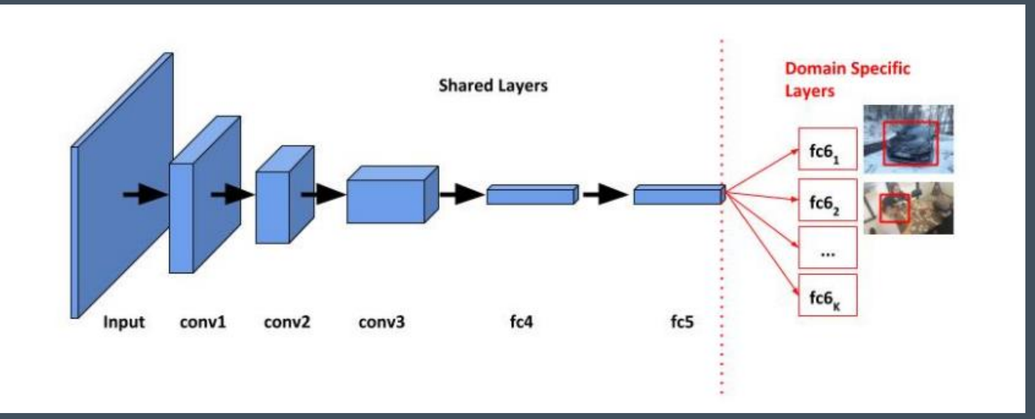
Maintain multiple CNNs

Stores appearence change in a tree structure

VGG-M on Imagenet

MDNet

Theory and Re-Implementation



Multi domain training method

Maintain multiple domain specific layers

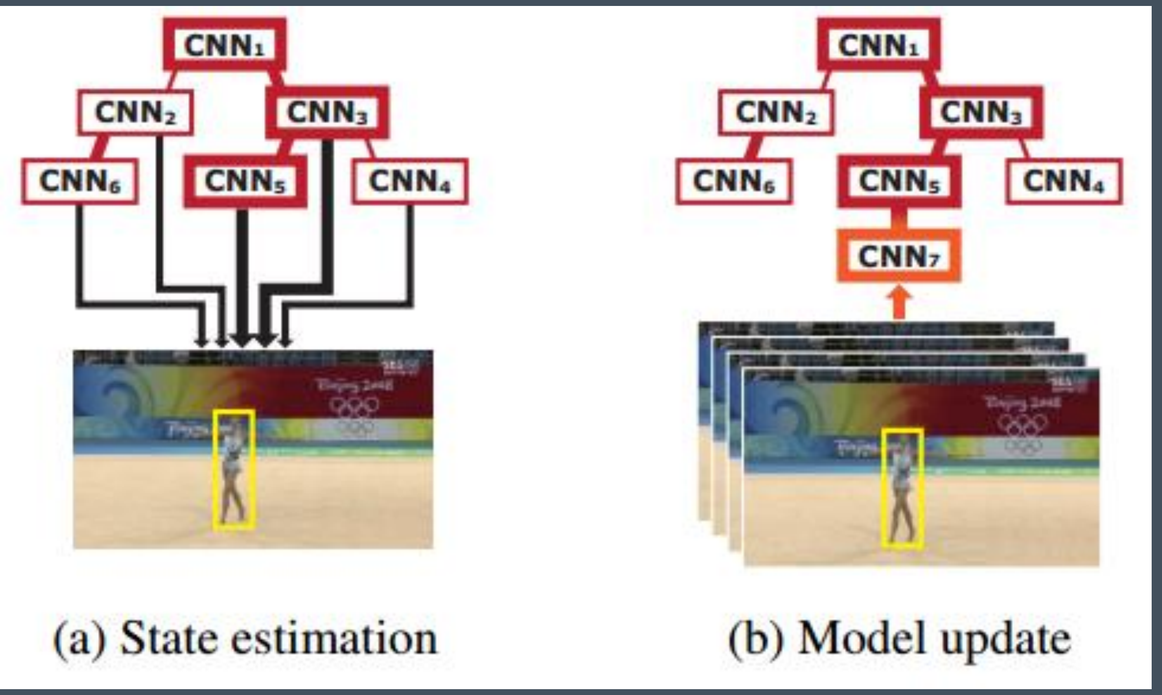
After training on K amount of sequences it has K domain specific layers

Train the network with respect to each domain iteratively to obtain generic target representations in the shared layers

At the end of testing all domain specific layers are discarded and replaced by a binary classification layer

TCNN

Theory and Re-Implementation



Maintains a tree-structure of CNNs in nodes during tracking based on target appearence change

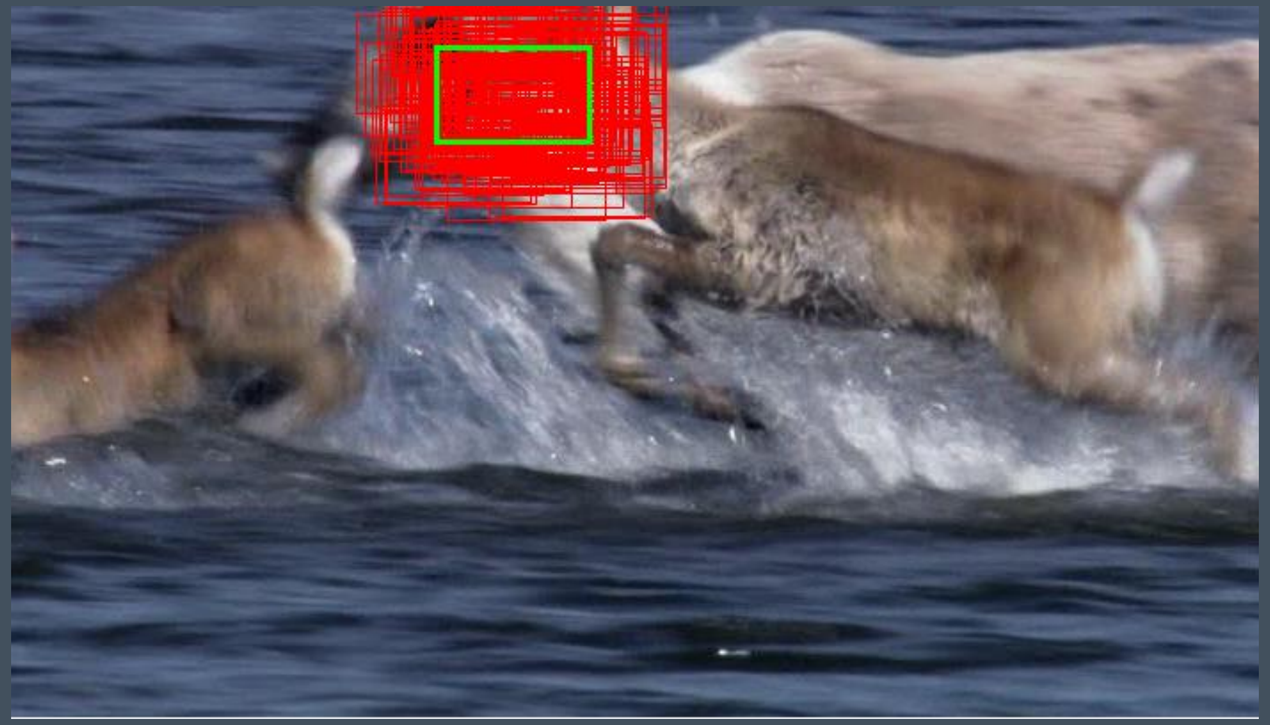
The proposed algorithm is effective to handle multi-modal target appearances, short-term occlusions and tracking failures.

The final target state is estimated by identifying the best sample in terms of a weighted average score from a set of active CNNs

During tracking online updates are performed along the active CNNs

Shared Implementation Details

Theory and Re-Implementation



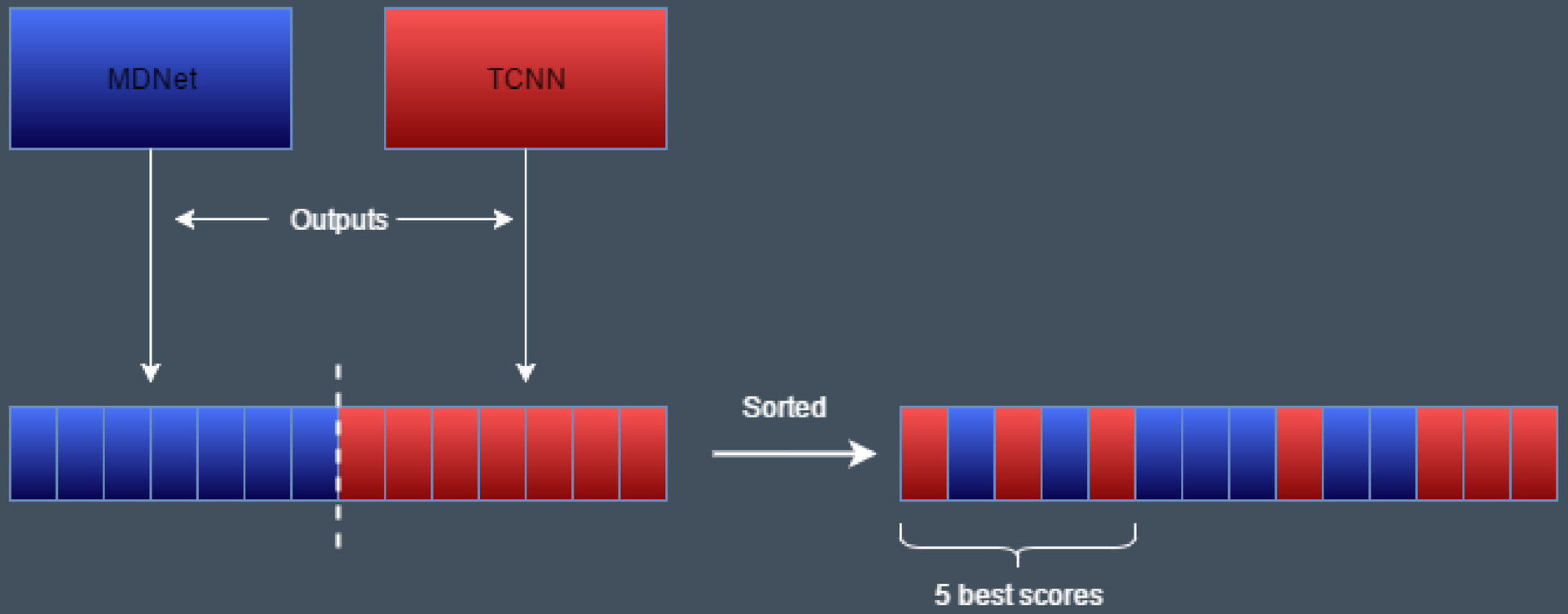
Initial finetuning based on samples generated around the first frame’s target position

Train a bbox regression model on the first frame and update it short term and long term

Perform online updates every 10th frame (different for T CNN)

Mean bounding box between the sampled bounding box es with the 5 best scores.

Combination of Scores



Ensemble

Create a stronger classifier with the use of multiple weak ones!

Weak classifiers are not able to perform robust tracking by themselves.

Has previously shown potential (AdaBoost)

Aggregate the prediction from classifiers.

For example tree structures

-> partitions the space of target object appearances (or equivalently the feature space) by using a binary tree, with a separate base tracker at each node (Gundogdu et al.).

Weaknesses:

- Increased storage
- Increased computation
- Decreased comprehensibility

Fusion of CNNs

Fusion of CNNs

Fusion Networks

Partition feature space (fx. RGB-D)

Fusion of features from the same feature space

Through fusion of CNNs can we learn a more robust classifier through training?

Late fusion (concatenating the last conv layers) of CNNs is a rare method for tracking

We don’t need increased storage, but computation time increases (more for training).

The decision method can be learned and needs fine tuning later (fx. Increasing the weights of better features -> scale layer) More on this later!

Fusion of CNNs

Fusion Architecture

- **Stream Layers**

  Two streams, take the same input with same measures (batch size (for training), image size, channels)

  Both streams end at the last conv layer

  Output has to be the same size (apply fx. zero padding to)

  Frozen weights in the streams, do not propagate back here

Fusion of CNNs

Fusion Architecture

- **Shared Layers**

Concatenate features from streams (2x the size of conv layer)

Fully connected layers (3) + ReLU(2) and Softmax

Fc layers progressively reduce the size

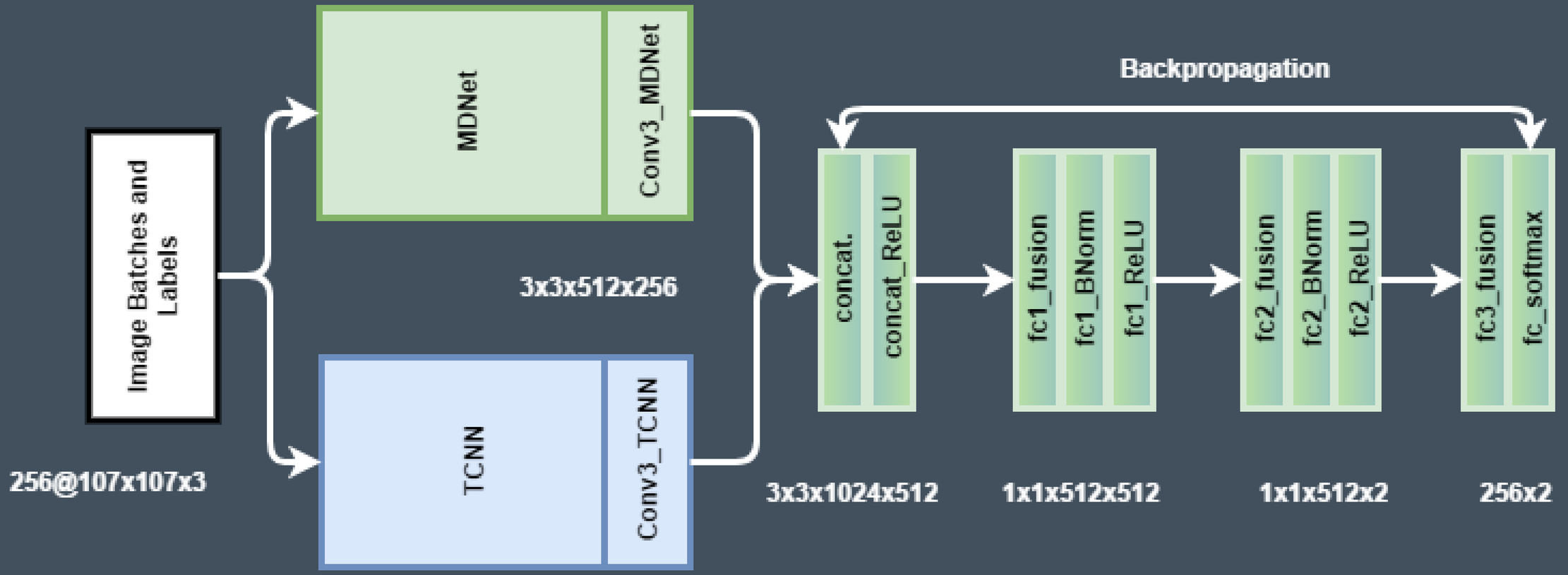
Batch normalization after the fully connected layers

More robust to bad initialization

Implementation of FCNN

Implementation

Fusion Architecture



Implementation

Directed Acyclic Graphs (DAG)

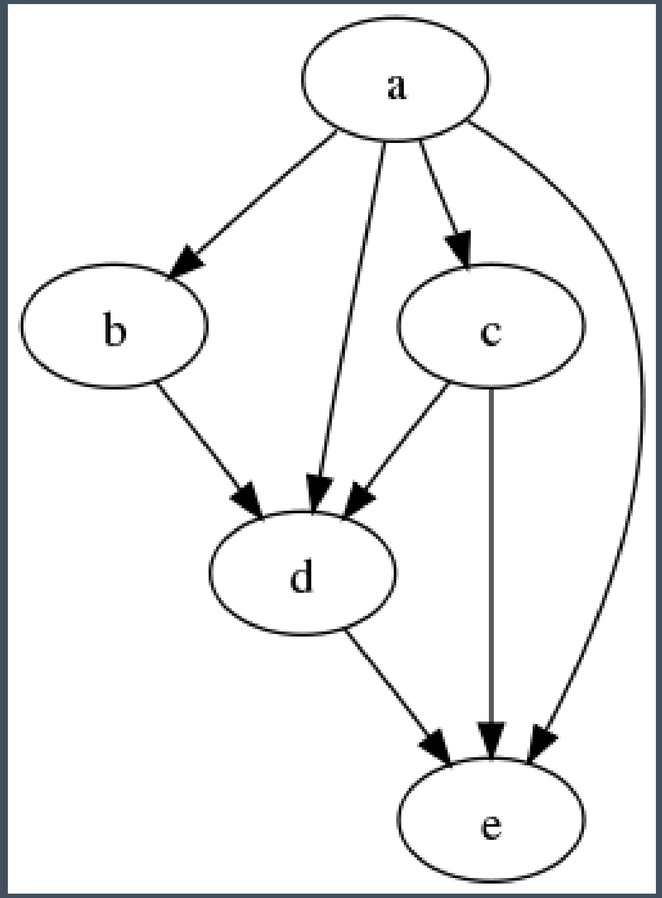
- Directed Acyclic Graphs (DAG)

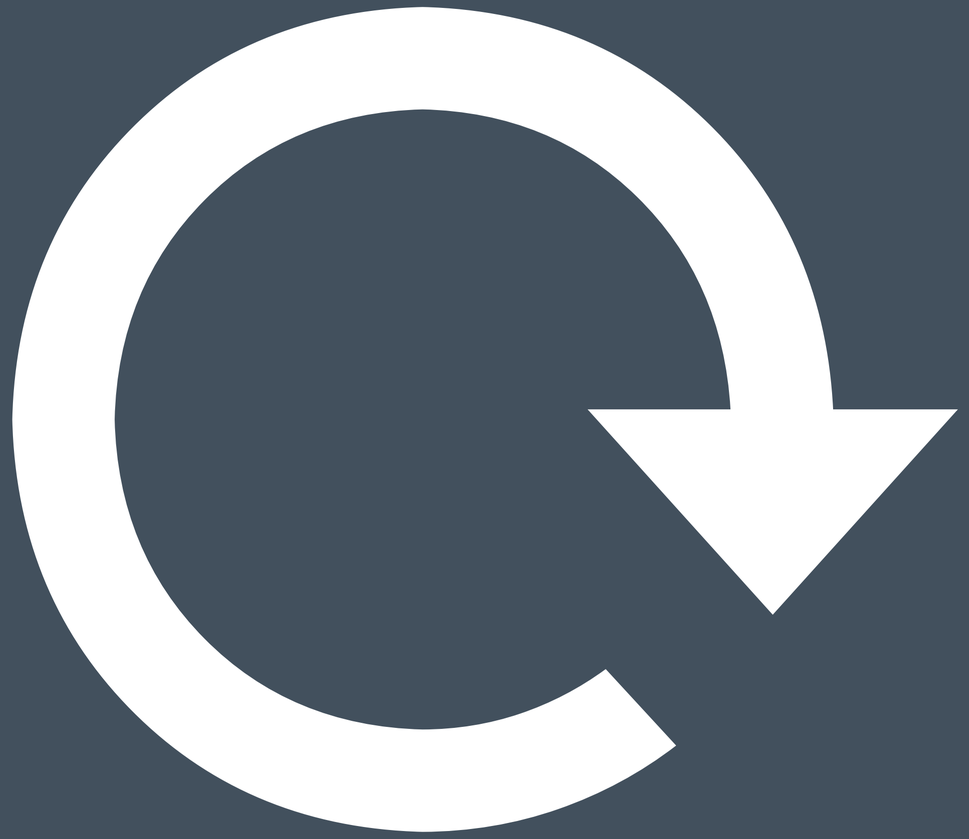
  DAG allows functions to have any number of inputs or outputs

  Variables and parameters can have an arbitrary number of outputs (a parameter with more of one output is shared between different layers), meaning it allows the concatenation of two network streams.

  Initiate one stream, stop at the last convolutional layer, execute the second stream, and then continue with the concatenation

  Implemented wrapper in MatConvNet!





Iterations

- 2(+1) for training
- 3 for tracking
- VOT2016 for training
- OTB50 test the network

Our first training algorithm

Create network

Creating a DagNN network and load the pre-trained models

Add new layers

Fully connected, batch normalization, activation and loss layers

Load Sequences

Information including image list, ground-truths. Shuffle frames.

Evaluate

For each epoch and sequence get batches run through the network, evaluate and update weights

Our first training algorithm

Create network

Creating a DagNN network and load the pre-trained models

Add new layers

Fully connected, batch normalization, activation and loss layers

Load Sequences

Information including image list, ground-truths. Shuffle frames.

Evaluate

For each epoch and sequence get batches run through the network, evaluate and update weights

Our first training algorithm

Create network

Creating a DagNN network and load the pre-trained models

Add new layers

Fully connected, batch normalization, activation and loss layers

Load Sequences

Information including image list, ground-truths. Shuffle frames.

Evaluate

For each epoch and sequence get batches run through the network, evaluate and update weights

Our first training algorithm

Create network

Creating a DagNN network and load the pre-trained models

Add new layers

Fully connected, batch normalization, activation and loss layers

Load Sequences

Information including image list, ground-truths. Shuffle frames.

Evaluate

For each epoch and sequence get batches run through the network, evaluate and update weights

Our first tracking algorithm

Load the trained model.

Input = 256 samples around target from the previous frame.

For each frame in the sequence get 256 scores from the end of the network, sort the scores, get the mean of the top 5 samples.

Our second tracking algorithm

Bounding Box Regression

Network model is the same!

Tracking Algorithm:

Train bounding box regression model on first frame (linear regression model)

Continue the same process as first tracker but in the end when a target is found adjust the bounding box using the regression model.

Our third tracking algorithm

Bounding Box Regression
+ Online Updates

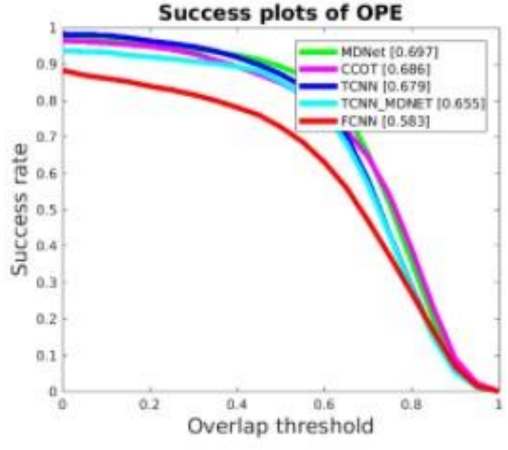
Network model is the same!

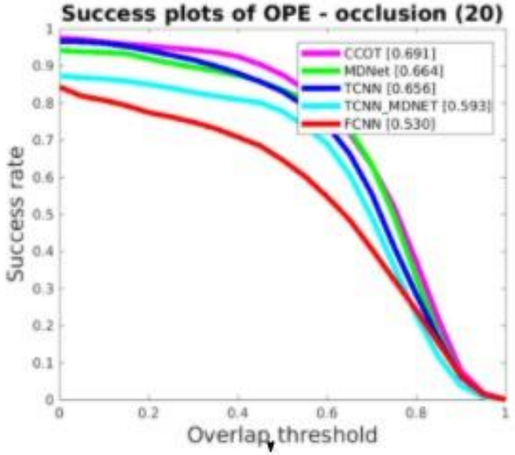
Tracking Algorithm:

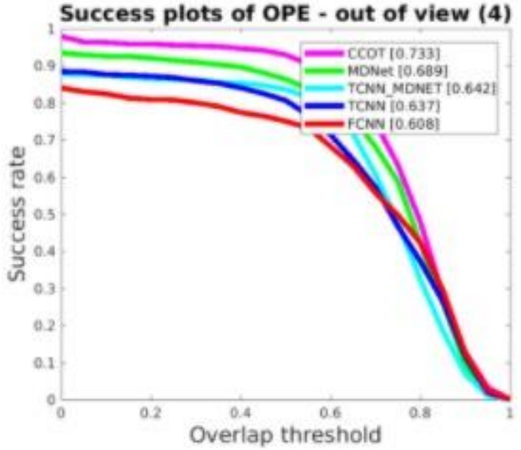
Generate short term (every 20) and long term (every 100) samples.

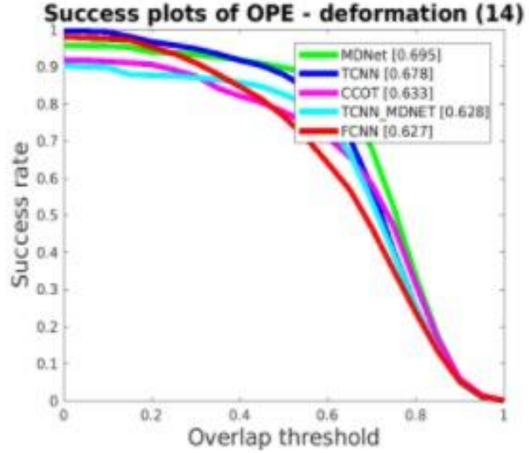
Finetune the network every 10 frames.

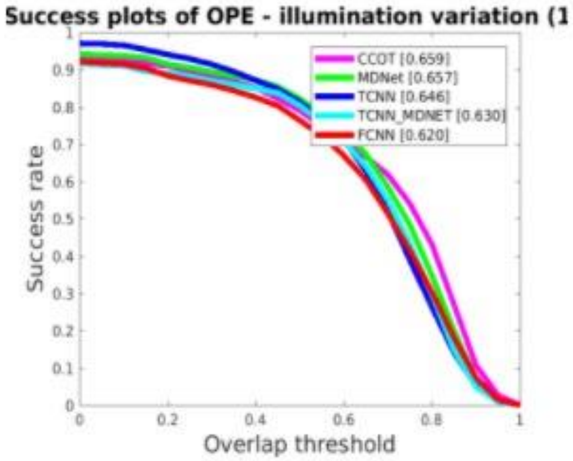
Use long term positive and short term negative samples.

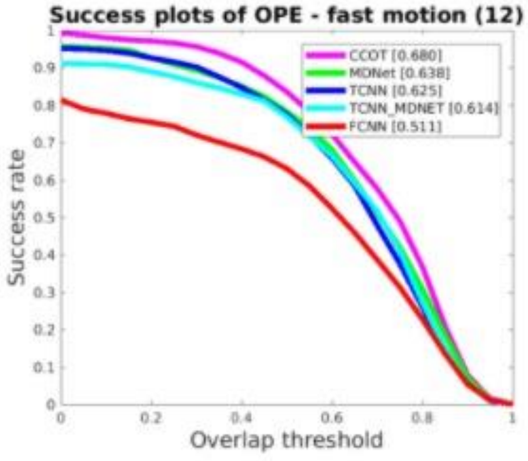

Success plots of OPE
MDNet [0.697]
CCOT [0.686]
TCNN [0.679]
TCNN_MDNET [0.655]
FCNN [0.583]
Success rate
Overlap threshold

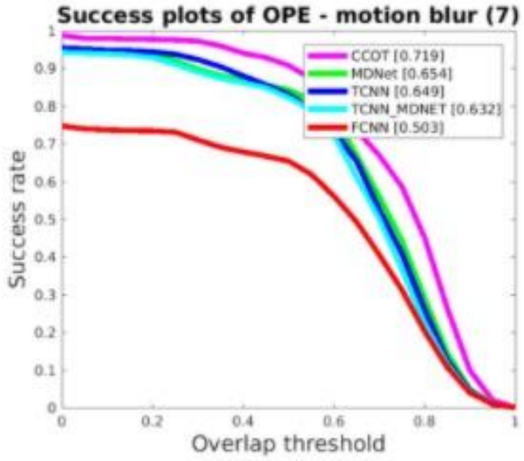

Success plots of OPE - occlusion (20)
CCOT [0.691]
MDNet [0.664]
TCNN [0.656]
TCNN_MDNET [0.593]
FCNN [0.530]
Success rate
Overlap threshold

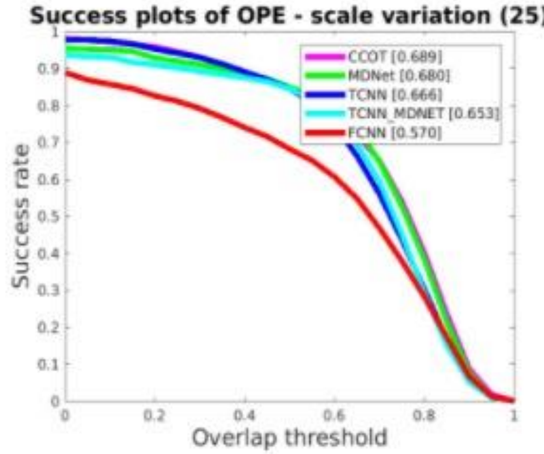

Success plots of OPE - out of view (4)
CCOT [0.733]
MDNet [0.689]
TCNN_MDNET [0.642]
TCNN [0.637]
FCNN [0.608]
Success rate
Overlap threshold

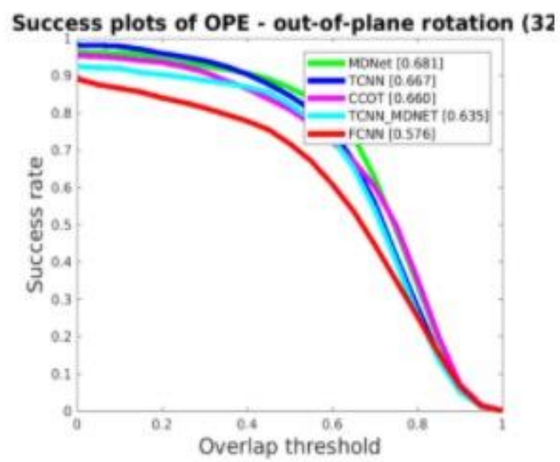

Success plots of OPE - deformation (14)
MDNet [0.695]
TCNN [0.678]
CCOT [0.633]
TCNN_MDNET [0.628]
FCNN [0.627]
Success rate
Overlap threshold

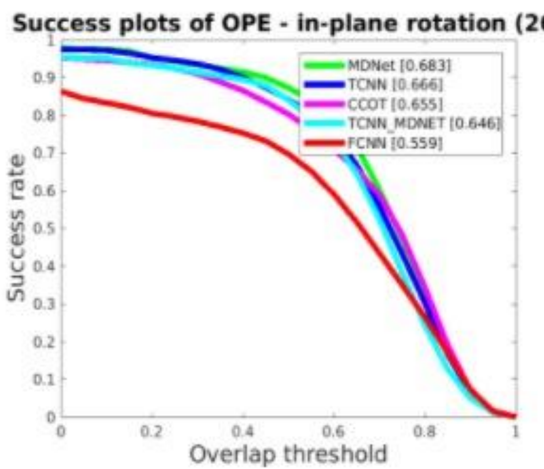

Success plots of OPE - illumination variation (1
CCOT [0.659]
MDNet [0.657]
TCNN [0.646]
TCNN_MDNET [0.630]
FCNN [0.620]
Success rate
Overlap threshold

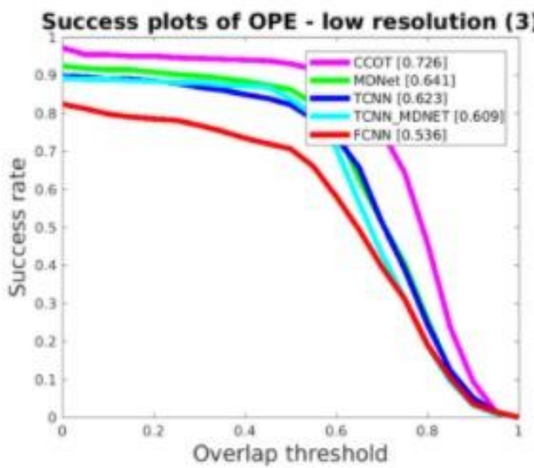

Success plots of OPE - fast motion (12)
CCOT [0.680]
MDNet [0.638]
TCNN [0.625]
TCNN_MDNET [0.614]
FCNN [0.511]
Success rate
Overlap threshold

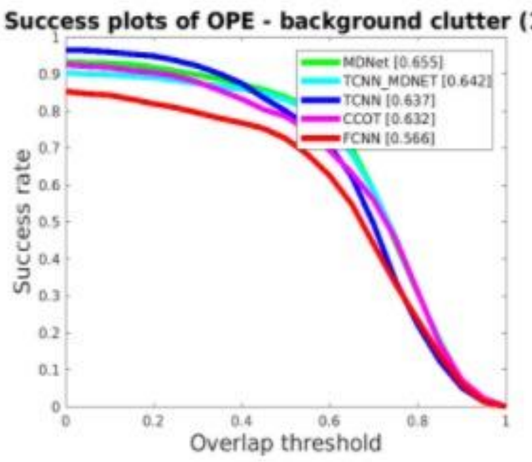

Success plots of OPE - motion blur (7)
CCOT [0.719]
MDNet [0.654]
TCNN [0.649]
TCNN_MDNET [0.632]
FCNN [0.503]
Success rate
Overlap threshold


Success plots of OPE - scale variation (25)
CCOT [0.689]
MDNet [0.680]
TCNN [0.666]
TCNN_MDNET [0.653]
FCNN [0.570]
Success rate
Overlap threshold


Success plots of OPE - out-of-plane rotation (32
MDNet [0.681]
TCNN [0.667]
CCOT [0.660]
TCNN_MDNET [0.635]
FCNN [0.576]
Success rate
Overlap threshold


Success plots of OPE - in-plane rotation (26
MDNet [0.683]
TCNN [0.666]
CCOT [0.655]
TCNN_MDNET [0.646]
FCNN [0.559]
Success rate
Overlap threshold


Success plots of OPE - low resolution (3)
CCOT [0.726]
MDNet [0.641]
TCNN [0.623]
TCNN_MDNET [0.609]
FCNN [0.536]
Success rate
Overlap threshold


Success plots of OPE - background clutter (16
MDNet [0.655]
TCNN_MDNET [0.642]
TCNN [0.637]
CCOT [0.632]
FCNN [0.566]
Success rate
Overlap threshold

Our second training algorithm

Procedure:

Add additional information from the label files considering the 5 labels.

Adding an additional layer.

Calculating weights by taking the current batch frames (in our case 8) label values, multiply them with the label weight add them up.

Label Weight:

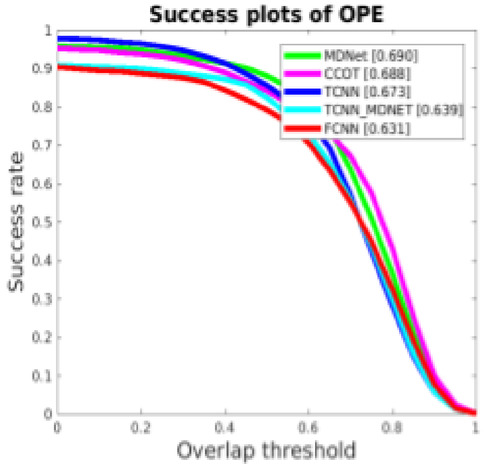
Calculated using the benchmark .

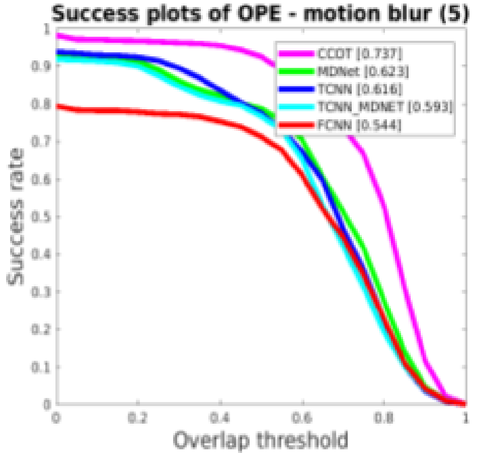
If the value is better in MDNet the difference between MDNet and TCNN is multiplied to 1.

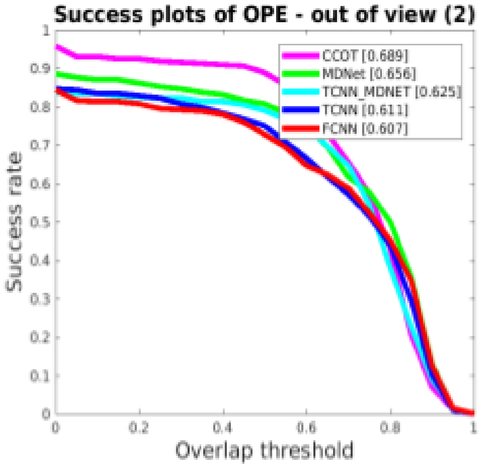
All values are added together to get the final weighs.

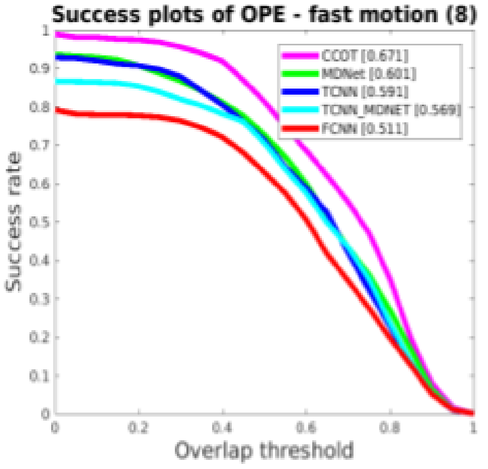
Experimenting using weights x1, x2 and x10 (discarded x10).

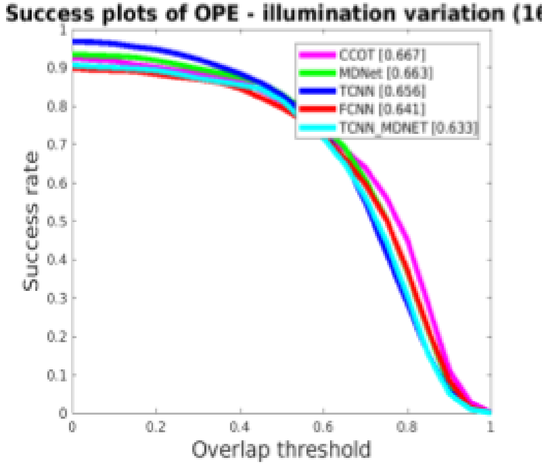
Qualitative Results

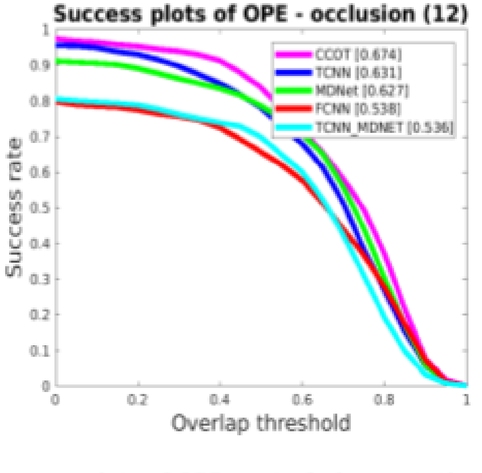

Success plots of OPE
MDNet [0.690]
CCOT [0.688]
TCNN [0.673]
TCNN_MDNET [0.639]
FCNN [0.631]
Success rate
Overlap threshold

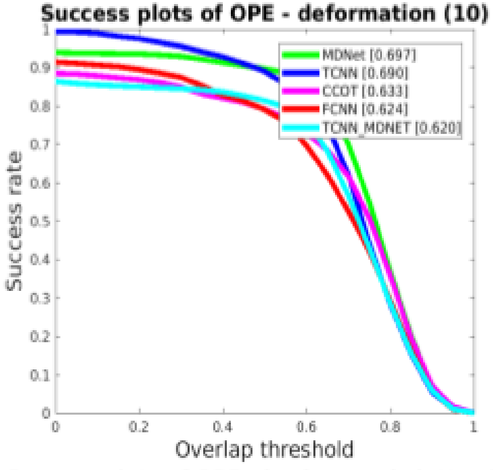

Success plots of OPE - motion blur (5)
CCOT [0.737]
MDNet [0.623]
TCNN [0.616]
TCNN_MDNET [0.593]
FCNN [0.544]
Success rate
Overlap threshold

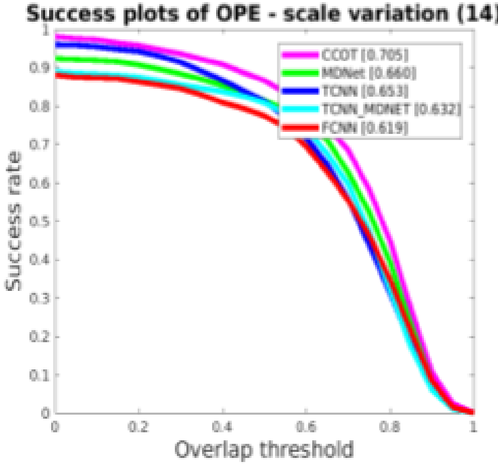

Success plots of OPE - out of view (2)
CCOT [0.689]
MDNet [0.656]
TCNN_MDNET [0.625]
TCNN [0.611]
FCNN [0.607]
Success rate
Overlap threshold

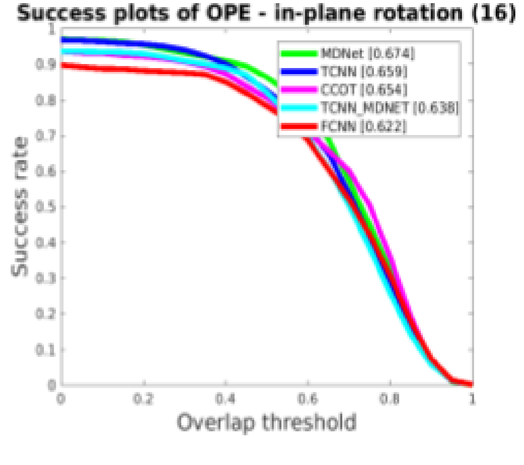

Success plots of OPE - fast motion (8)
CCOT [0.671]
MDNet [0.601]
TCNN [0.591]
TCNN_MDNET [0.569]
FCNN [0.511]
Success rate
Overlap threshold

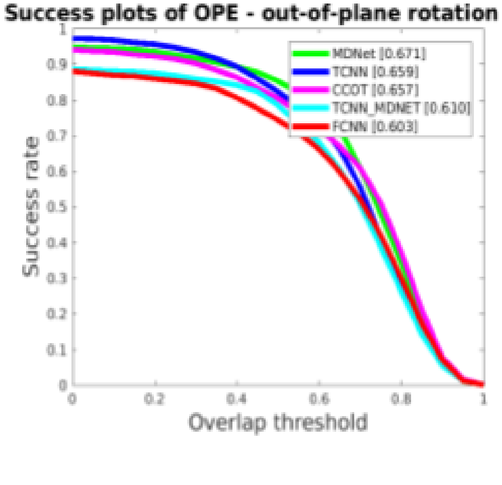

Success plots of OPE - illumination variation (1
CCOT [0.667]
MDNet [0.663]
TCNN [0.656]
FCNN [0.641]
TCNN_MDNET [0.633]
Success rate
Overlap threshold

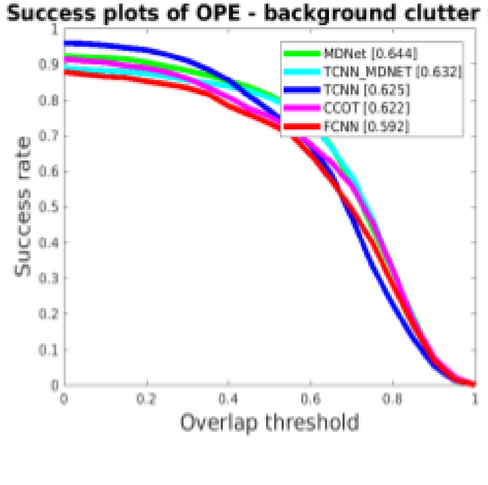

Success plots of OPE - occlusion (12)
CCOT [0.674]
TCNN [0.631]
MDNet [0.627]
FCNN [0.538]
TCNN_MDNET [0.536]
Success rate
Overlap threshold

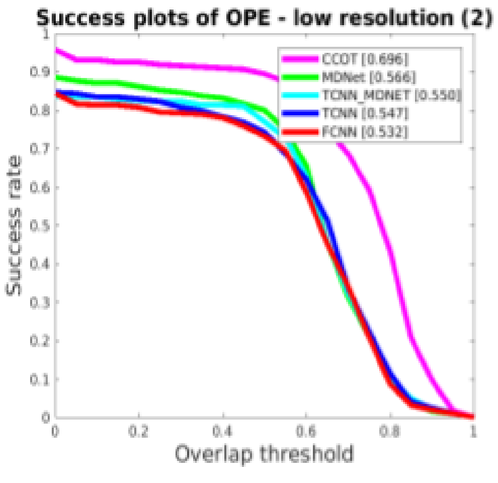

Success plots of OPE - deformation (10)
MDNet [0.697]
TCNN [0.690]
CCOT [0.633]
FCNN [0.624]
TCNN_MDNET [0.620]
Success rate
Overlap threshold


Success plots of OPE - scale variation (14)
CCOT [0.705]
MDNet [0.660]
TCNN [0.653]
TCNN_MDNET [0.632]
FCNN [0.619]
Success rate
Overlap threshold


Success plots of OPE - in-plane rotation (16)
MDNet [0.674]
TCNN [0.659]
CCOT [0.654]
TCNN_MDNET [0.638]
FCNN [0.622]
Success rate
Overlap threshold


Success plots of OPE - out-of-plane rotation (2
MDNet [0.671]
TCNN [0.659]
CCOT [0.657]
TCNN_MDNET [0.610]
FCNN [0.603]
Success rate
Overlap threshold


Success plots of OPE - background clutter (14
MDNet [0.644]
TCNN_MDNET [0.632]
TCNN [0.625]
CCOT [0.622]
FCNN [0.592]
Success rate
Overlap threshold


Success plots of OPE - low resolution (2)
CCOT [0.696]
MDNet [0.566]
TCNN_MDNET [0.550]
TCNN [0.547]
FCNN [0.532]
Success rate
Overlap threshold

Our third training algorithm

Weight by Average All Frames:

Calculating weights by taking all label values for each label (1s and 0s), multiply them with the label weight add them up.

Label Weight:

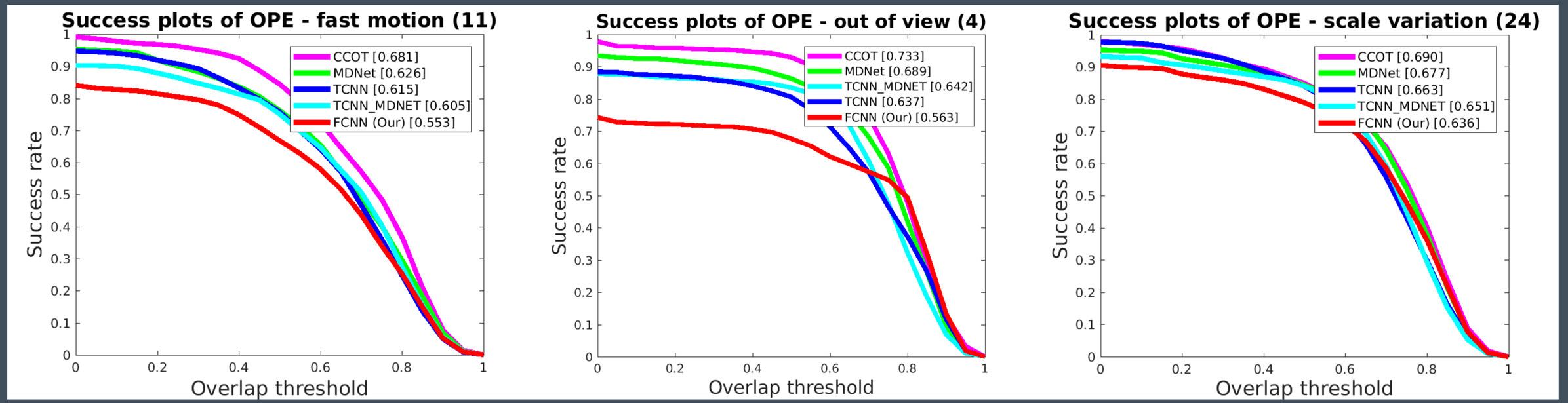
Calculated using the benchmark .

If the value is better in MDNet the difference between MDNet and TCNN is multiplied to 1.

All values are added together to get the final weights.

Final Results

Final Results

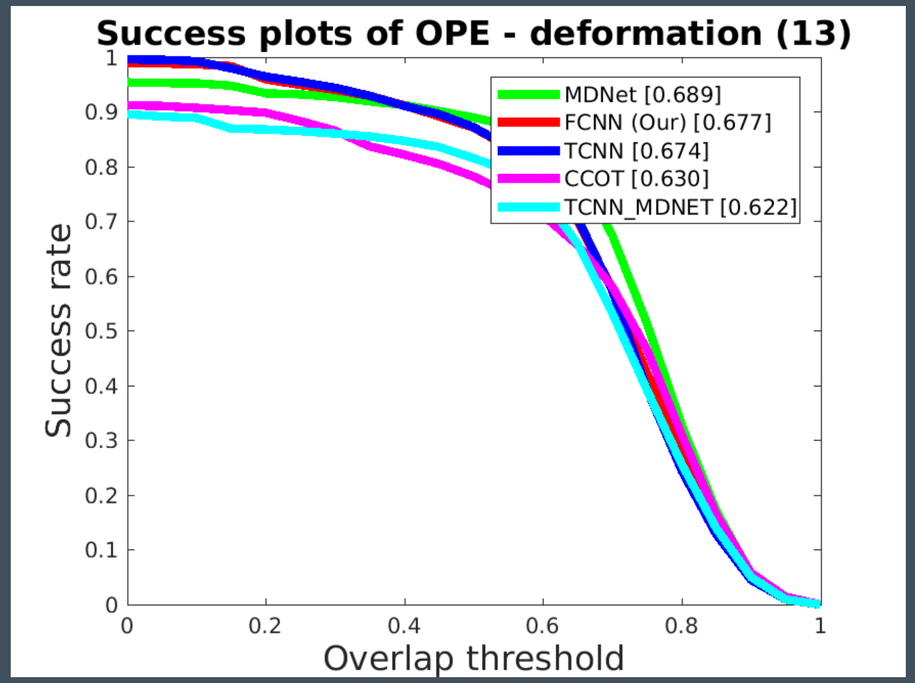


0.511 → 0.553

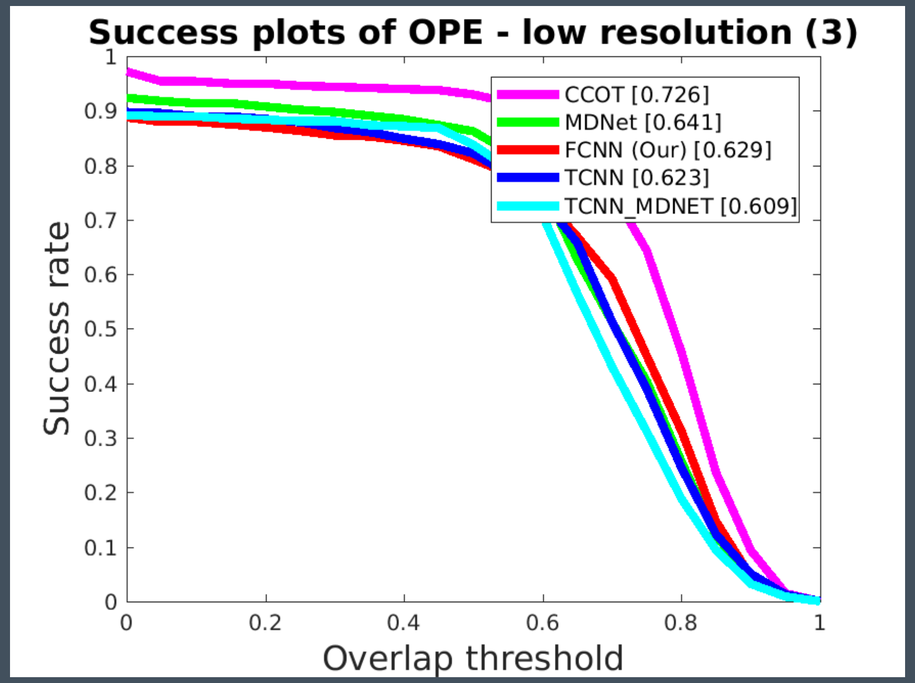
0.608 → 0.563

0.570 → 0.636

Final Results

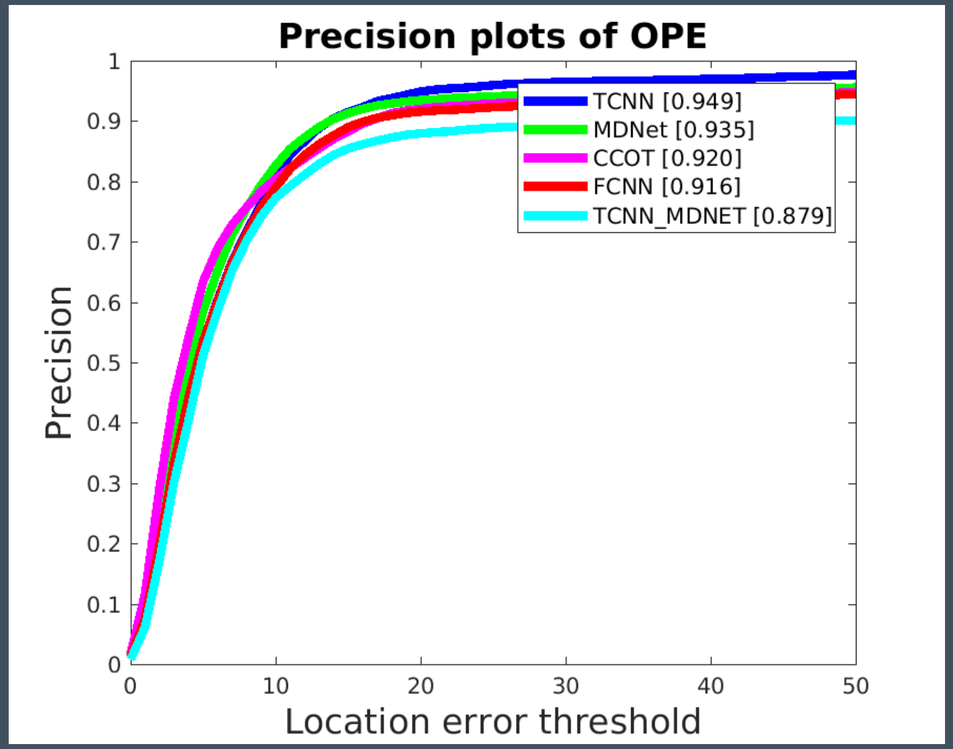


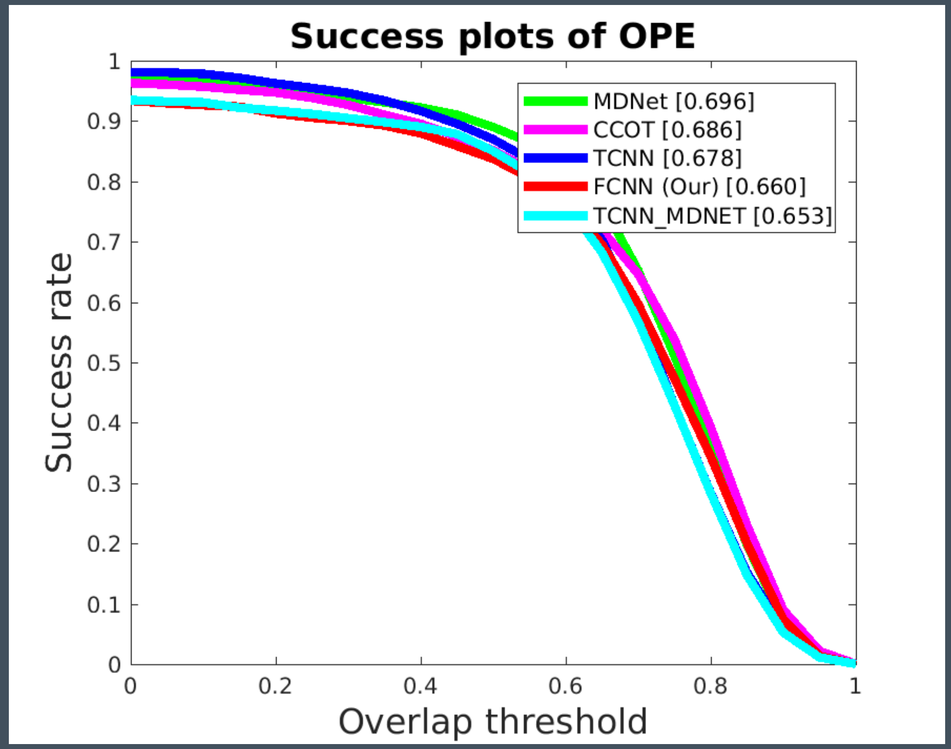
0.627 → 0.677



0.536 → 0.629

Final Results





0.818 → 0.916

0.631 → 0.660

Quantitative Results

Overall Scores

Tracker	Success (AUC)	Precision (Error)
MDNet	0.696	0.935
CCOT	0.686	0.920
TCNN	0.678	0.949
FCNN	0.660	0.916
TCNN_MDNet	0.653	0.879

FPS (on CPU)	Training
~9.5 sec/frame	15 hours

Qualitative Results

Qualitative Results

Qualitative Results

Discussion & Future Work

Discussion & Future Work

Training:

Train the last layer of the network multi-domain (sequence specific).

Increase the weights for low success imaging effects.

Intelligent weighting.

Use OTB for training (more labels).

Three stream fusion with CCOT or ECO.

Some parts in MDNet changed since the paper release.

Discussion & Future Work

Tracking:

Multiple FCNN in tree structure for target state prediction (increase tracking time).

Online updates based on appearence models saved in the tree structure.

Use GPU instead of CPU for tracking (never tried to run it… ☹).

Optimization.

Discussion & Future Work

MatConvNet:

Use SimpleNN for training and tracking (could increase performance).

Modifications in MatConvNet are SCAAAARY!

Some parts of the implementation of TCNN and MDNet remain to be a black box.

Conclusion

Conclusion

Presented an architecture for late fusion of CNN trackers performing on the same feature space.

Increased performance with manual weighting (still space for improvement).

The tracker reaches SoTA performance compared to other CNN trackers.

FCNN never reaches best results on OTB-50.

Thank you!

VGIS 842

Cédric Magnan
Louise Abela
Péter Rohoska

References

http://cs231n.github.io/convolutional-networks/

E. Gundogdu, H. Ozkan, and A. A. Alatan. “Ensemble Of adaptive correlation filters for robust visual tracking”. In: 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). Aug. 2016, pp. 15–22 . doi: 10.1109/AVSS.2016.7738031.

Andrea Vedaldi and Karel Lenc. “MatConvNet - Convolutional Neural Networks for MATLAB”. In: arXiv:1412.4564 [cs] (Dec. 15, 2014). arXiv: 1412 . 4564. url: http://arxiv.org/abs/1412.4564 (visited on 05/27/2017).