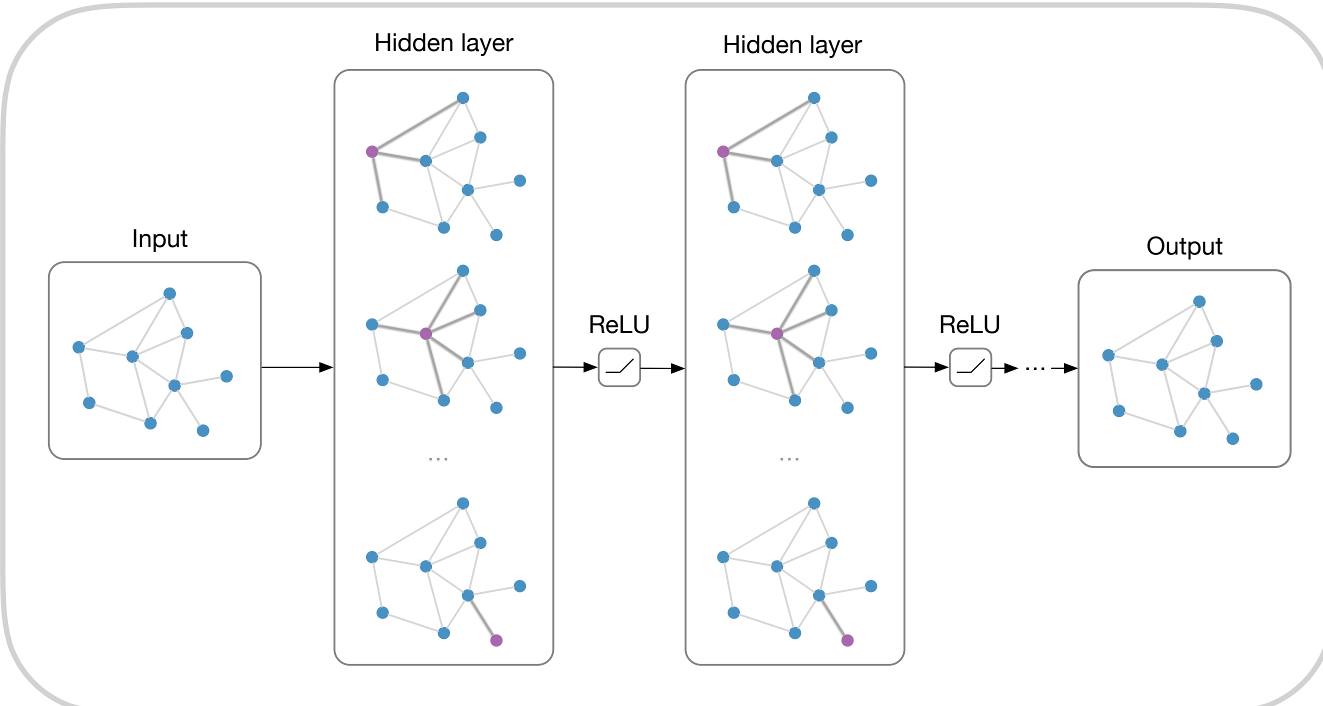
Deep Learning on Graph-Structured Data

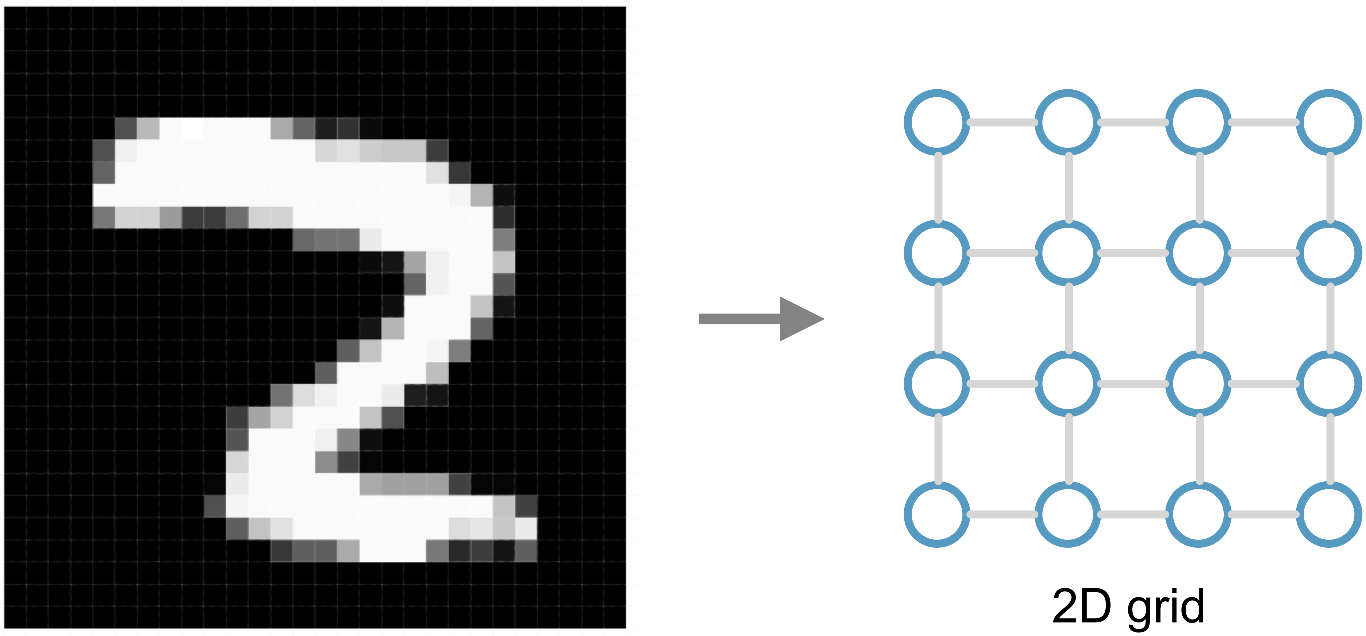


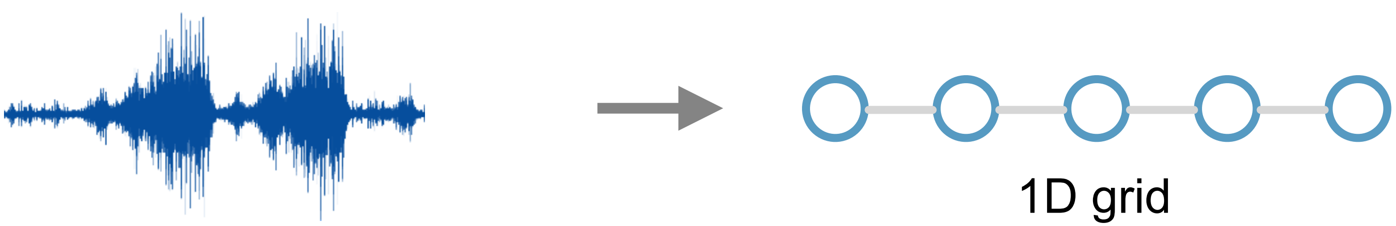
Thomas Kipf, 1 December 2016

Universiteit van Amsterdam

Recap: Deep learning on Euclidean data

Euclidean data: grids, sequences…

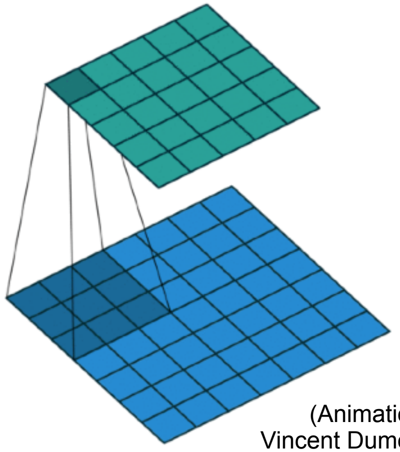




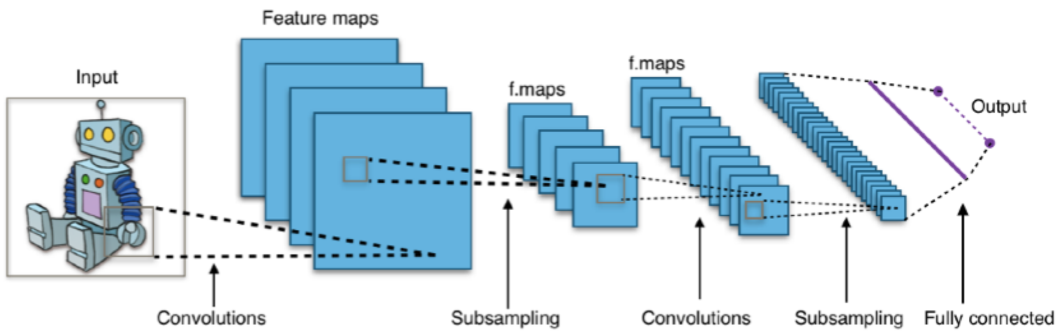
Recap: Deep learning on Euclidean data

We know how to deal with this:

Convolutional neural networks (CNNs)

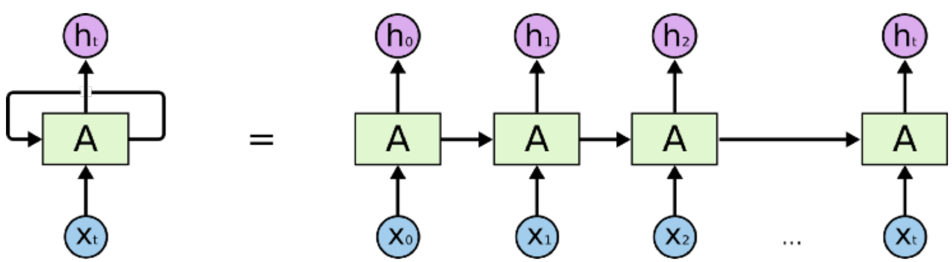


(Animation by Vincent Dumoulin)



(Source: Wikipedia)

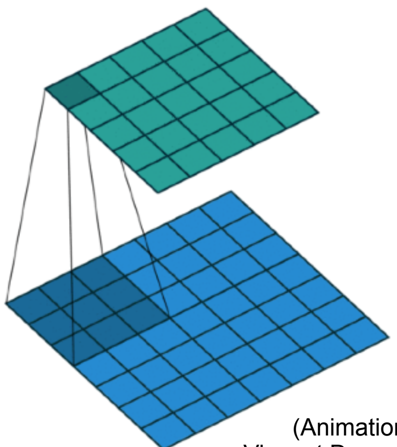
or recurrent neural networks (RNNs)



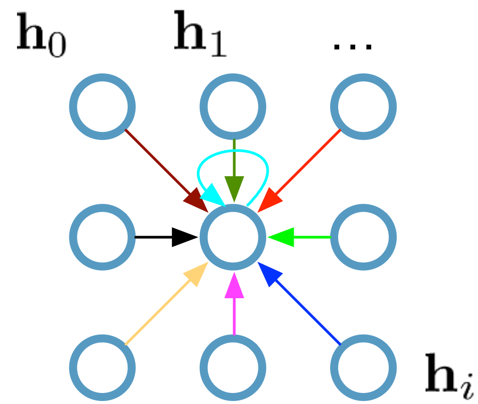
(Source: Christopher Olah's blog)

Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



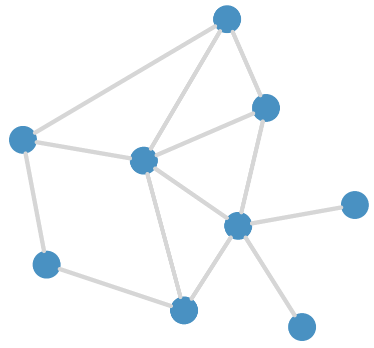
Update for a single pixel:

- Transform neighbors individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

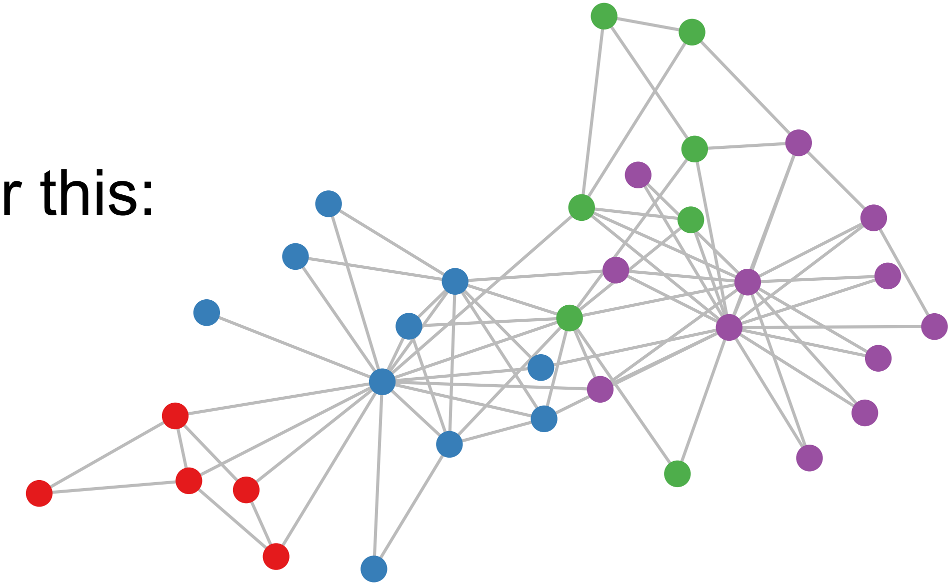
Full update: $\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \dots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$

Graph-structured data

What if our data looks like this?



or this:

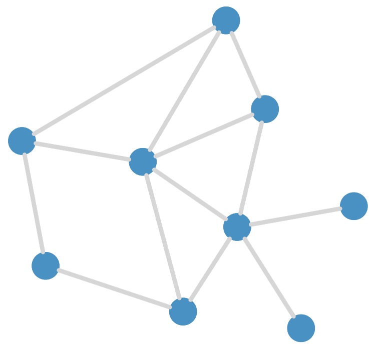


Real-world examples:

- Social networks
- World-wide-web
- Protein-interaction networks
- Telecommunication networks
- Knowledge graphs
- …

Graphs: Definitions

Graph: $G = (\mathcal{V}, \mathcal{E})$



$\mathcal{V}$: Set of nodes $\{v_i\}$, $|\mathcal{V}| = N$

$\mathcal{E}$: Set of edges $\{(v_i, v_j)\}$

We can define:

$\mathbf{A}$ (adjacency matrix): $A_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$

(can also be weighted)

Model wish list:

- Set of trainable parameters $\{\mathbf{W}^{(l)}\}$
- Trainable in $\mathcal{O}(|\mathcal{E}|)$ time
- Applicable even if the input graph changes

Spectral graph convolutions

Main idea:

Use **convolution theorem** to generalize convolution to graphs.

Loosely speaking:

A convolution corresponds to a multiplication in the Fourier domain.

Graph Fourier transform: [Hammond, Vandergheynst, Gribonval, 2009]

$\mathcal{F}_G[\mathbf{x}] = \mathbf{U}^T\mathbf{x}$ $\mathbf{U}$: eigenvectors of *graph Laplacian* $\mathbf{L}$

with $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ (normalized graph Laplacian)

and $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ (its eigen-decomposition)

D: degree matrix
$D_{ii} = \sum_j A_{ij}$

Spectral graph convolutional networks

Graph convolution: $\mathbf{g}, \mathbf{x} \in \mathbb{R}^N$

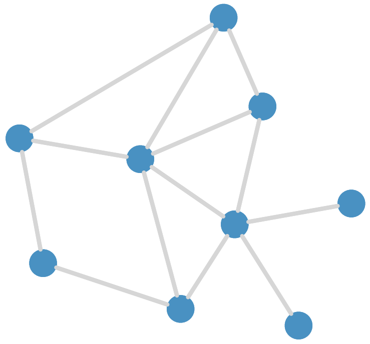
$$\mathbf{x} *_G \mathbf{g} = \mathcal{F}_G^{-1}\left[\mathcal{F}_G[\mathbf{g}] \odot \mathcal{F}_G[\mathbf{x}]\right] = \mathbf{U}\left(\mathbf{U}^T\mathbf{g} \odot \mathbf{U}^T\mathbf{x}\right)$$

or: $\mathbf{x} *_G \mathbf{g} = \mathbf{U}\,\mathrm{diag}(\hat{\mathbf{g}})\mathbf{U}^T\mathbf{x}$ with $\hat{\mathbf{g}} = \mathbf{U}^T\mathbf{g}$

Spectral CNN on graphs:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{U}\,\mathrm{diag}(\mathbf{w}^{(l)})\mathbf{U}^T\mathbf{h}_i^{(l)}\right)$$

[Bruna et al., ICLR 2014]

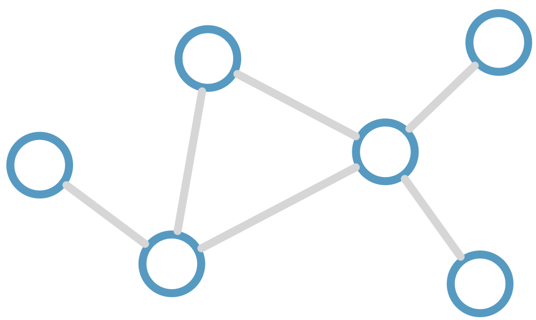


Limitations:

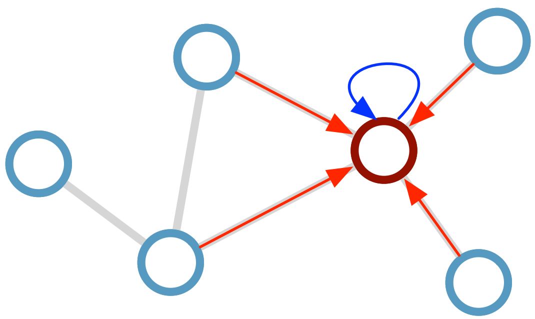
- Calculating $\mathbf{U}$ is expensive $\mathcal{O}(N^3)$
- Evaluating $\mathbf{U}^T\mathbf{x}$ is $\mathcal{O}(N^2)$
- Graph structure has to be fixed

Spatial graph convolutional networks (GCNs)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

$\mathcal{N}_i$: neighbor indices

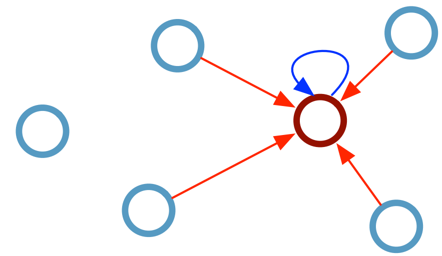
c_{ij} : norm. constant (per edge)

How is this related to spectral CNNs on graphs?

➡ Localized 1st-order approximation of spectral filters [Kipf & Welling, 2016]

Fully vectorized GCNs

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{H}^{(l)}\mathbf{W}_0^{(l)} + \tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}_1^{(l)}\right)$$

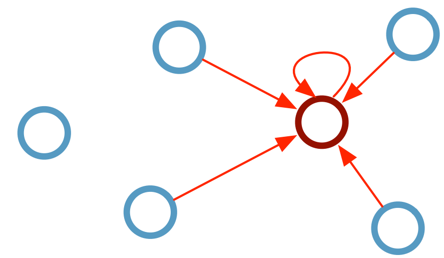


with $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ or $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$

$$\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)^T}, \ldots, \mathbf{h}_N^{(l)^T}]^T$$

Or treat self-connection in the same way:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}_1^{(l)}\right)$$



with $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_N)\tilde{\mathbf{D}}^{-\frac{1}{2}}$ or $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}(\mathbf{A} + \mathbf{I}_N)$

$$\tilde{D}_{ii} = \sum_j (A_{ij} + \delta_{ij})$$

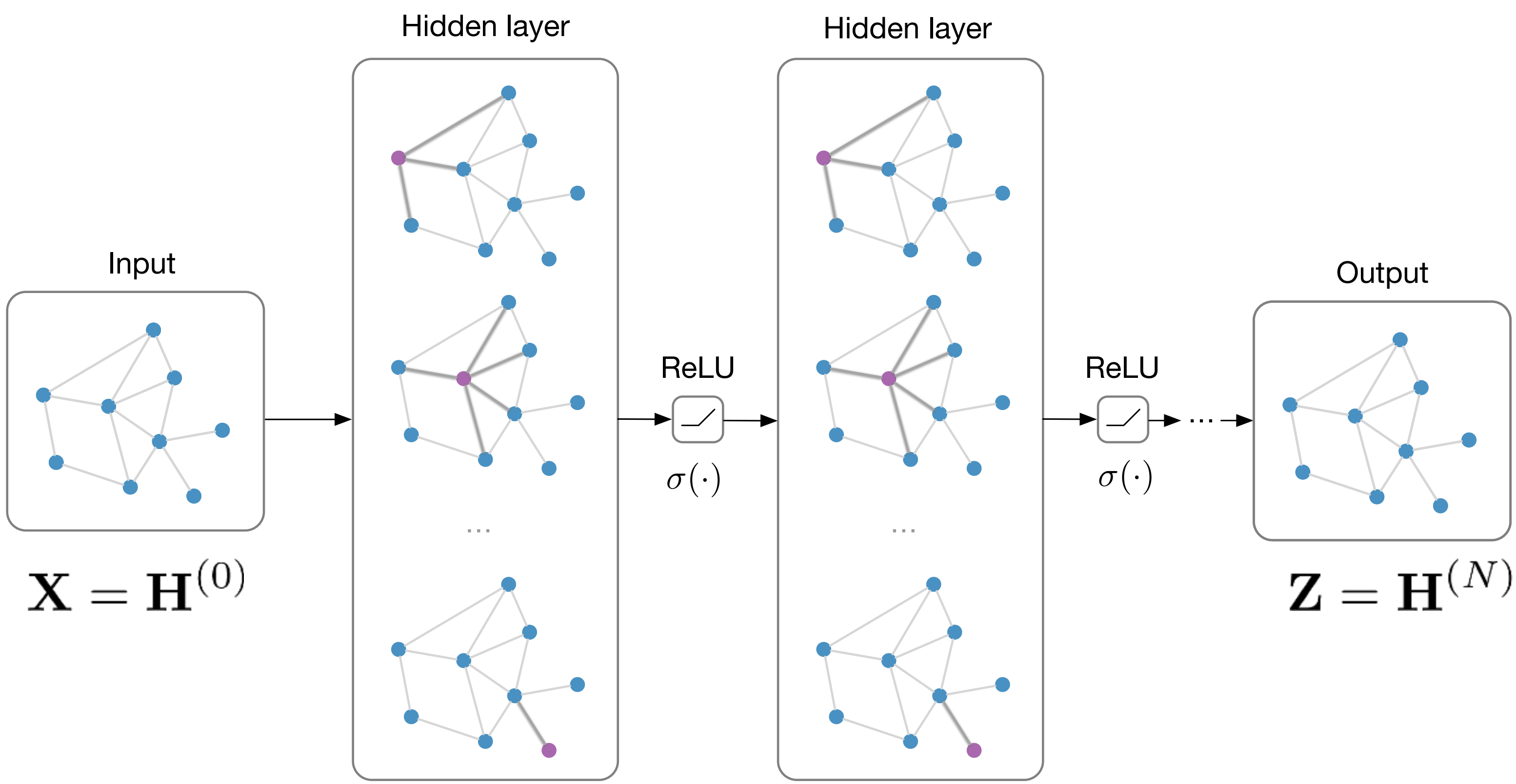
$\mathbf{A}$ is typically **sparse**

➡ We can use sparse matrix multiplications!

➡ Efficient $\mathcal{O}(|\mathcal{E}|)$ implementation in Theano or TensorFlow

GCN model architecture

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



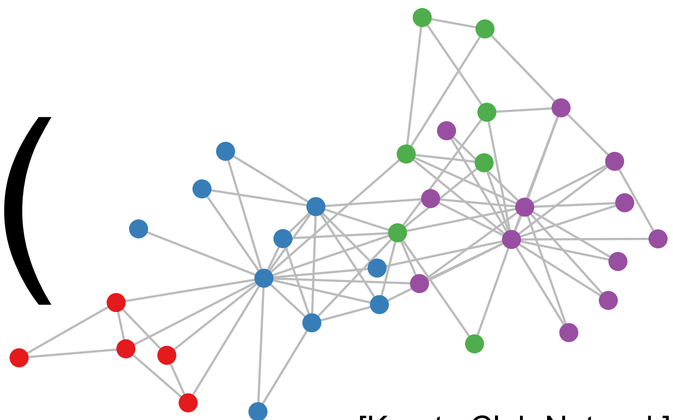
$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

What does it do? An example.

Forward pass through **untrained** 3-layer GCN model

Parameters initialized randomly

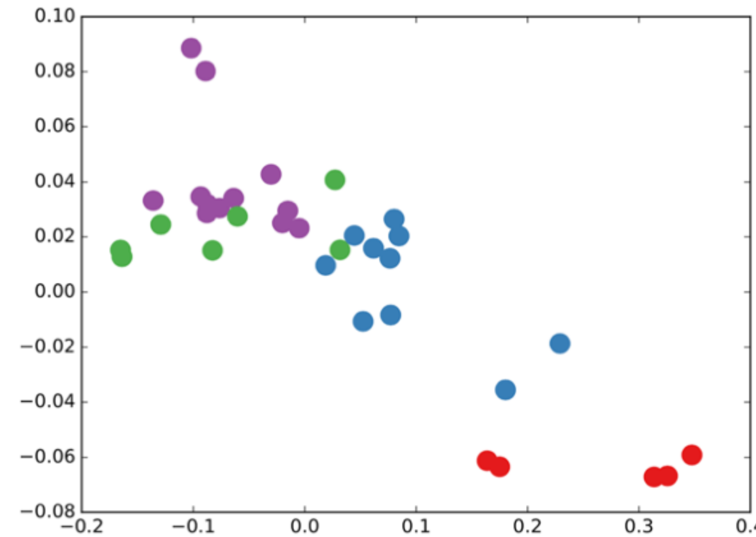
f(



[Karate Club Network]

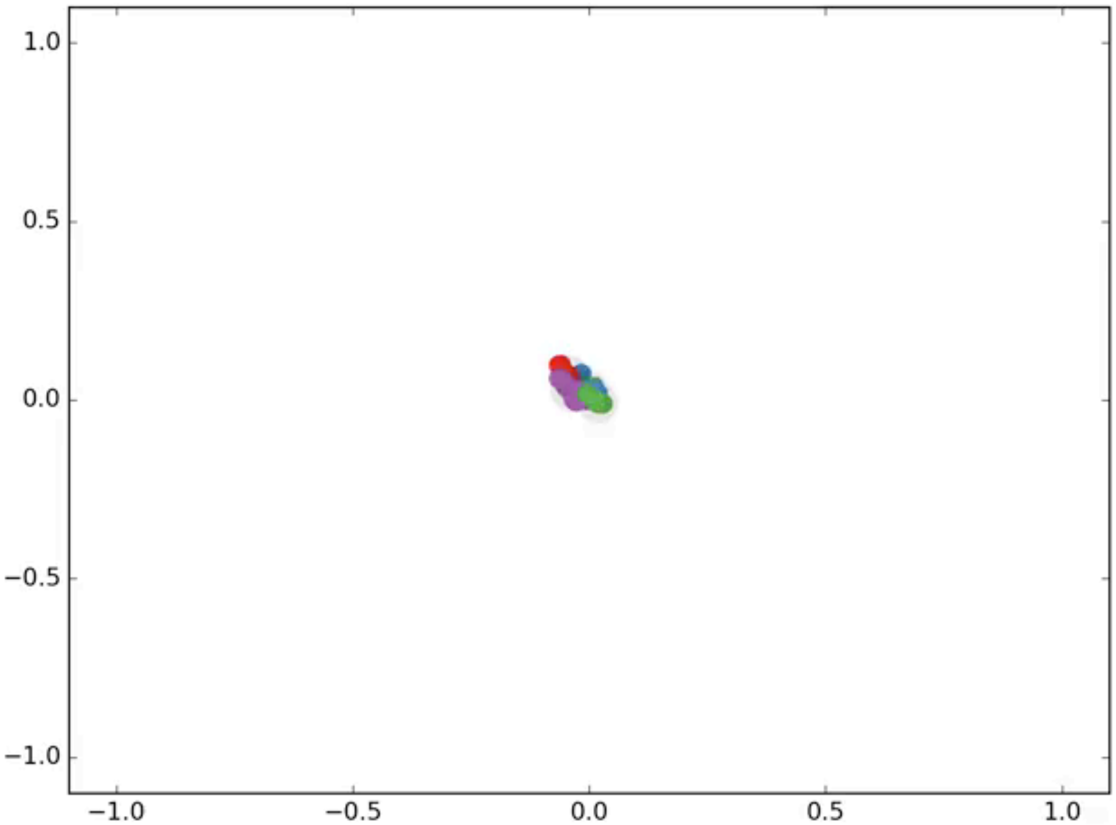
) =

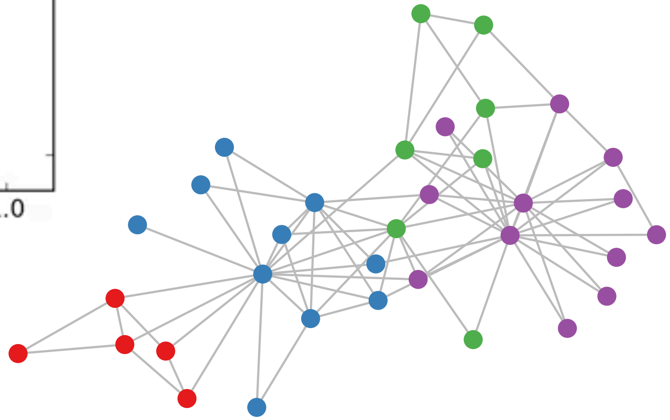
2-dim output per node



Produces (useful?) random embeddings!

Add labels and train (semi-supervised)





Video also available here:
http://tkipf.github.io/graph-convolutional-networks

Further reading

Blog post Graph Convolutional Networks:
http://tkipf.github.io/graph-convolutional-networks

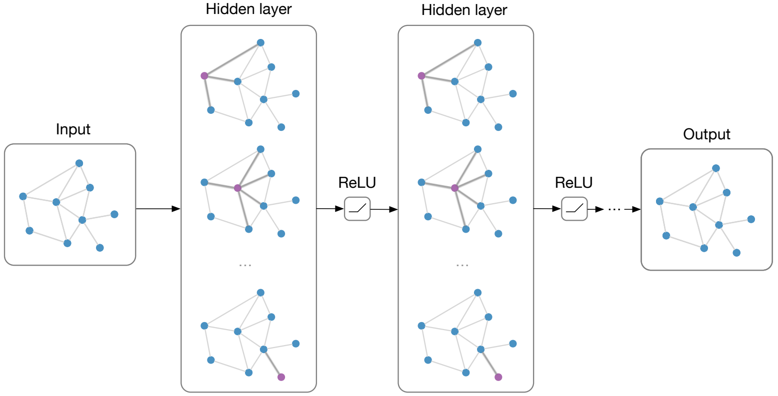
Code on Github:
http://github.com/tkipf/gcn

Paper (Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, 2016):
https://arxiv.org/abs/1609.02907

Questions? You can get in touch with me via:

- E-Mail: T.N.Kipf@uva.nl
- Twitter: @thomaskipf
- Web: http://tkipf.github.io

Interested in thesis projects? Get in touch!



VideoLSTM
Convolves, attends and flows for action recognition

Zhenyang Li

Kirill Gavrilyuk

Efstratios Gavves

Mihir Jain

Cees Snoek

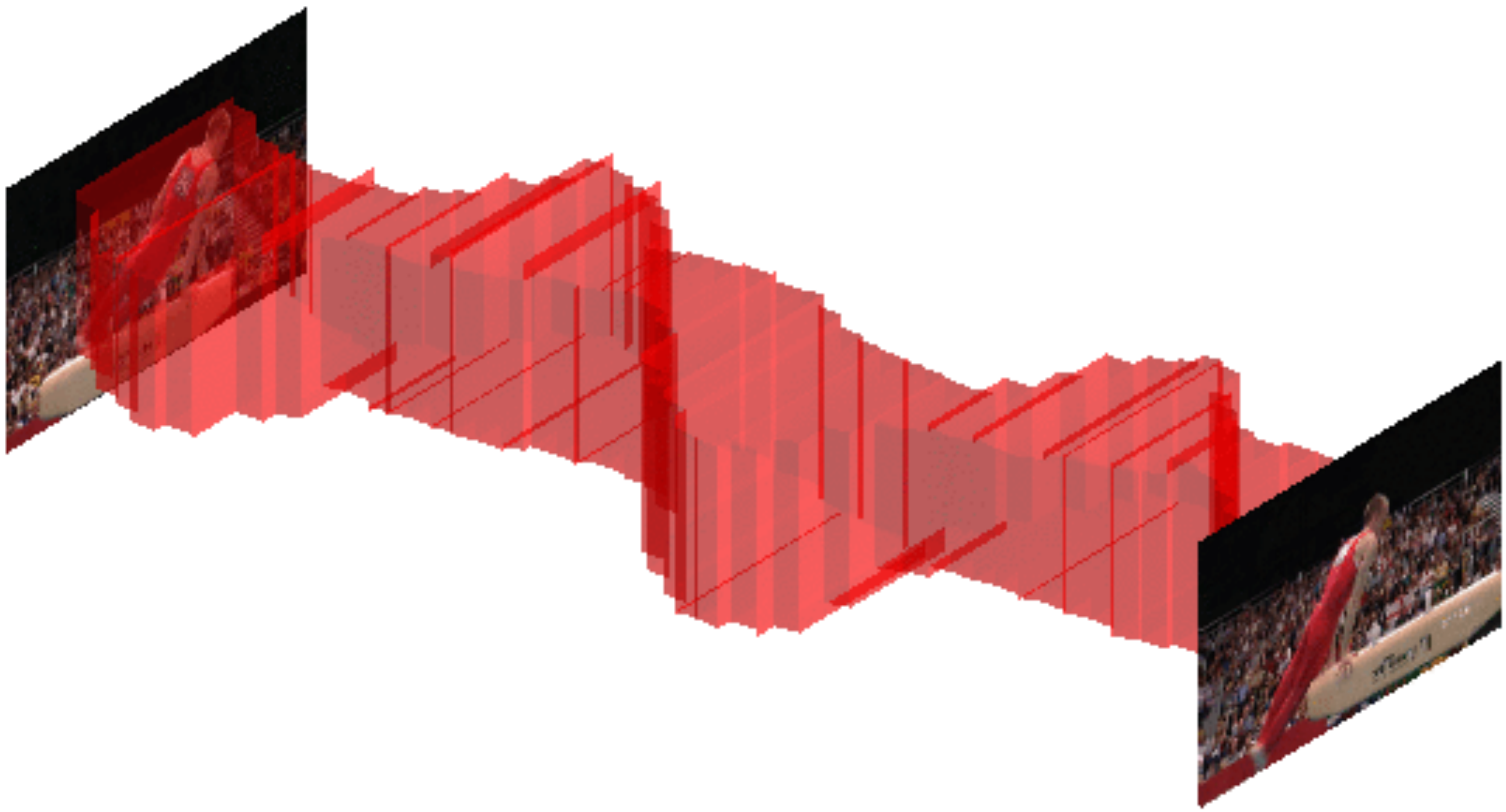
University of Amsterdam
The Netherlands

Deep Vision Lab

Motivation: Internet of things that video

Goal: Action Recognition

Understand **what** is happening **where** and **when**



Related work

DEEP LEARNING FOR ACTIONS

Ji et al. ICML10

Karpathy et al., CVPR14

Simonyan & Zisserman, NIPS14

Tran et al., ICCV15

ConvNet

3D convolutions

Need large amounts of data to learn filters

Two-stream

Learn spatial and temporal filters separately

We propose a more principled approach for learning frame-to-frame appearance and motion transitions.

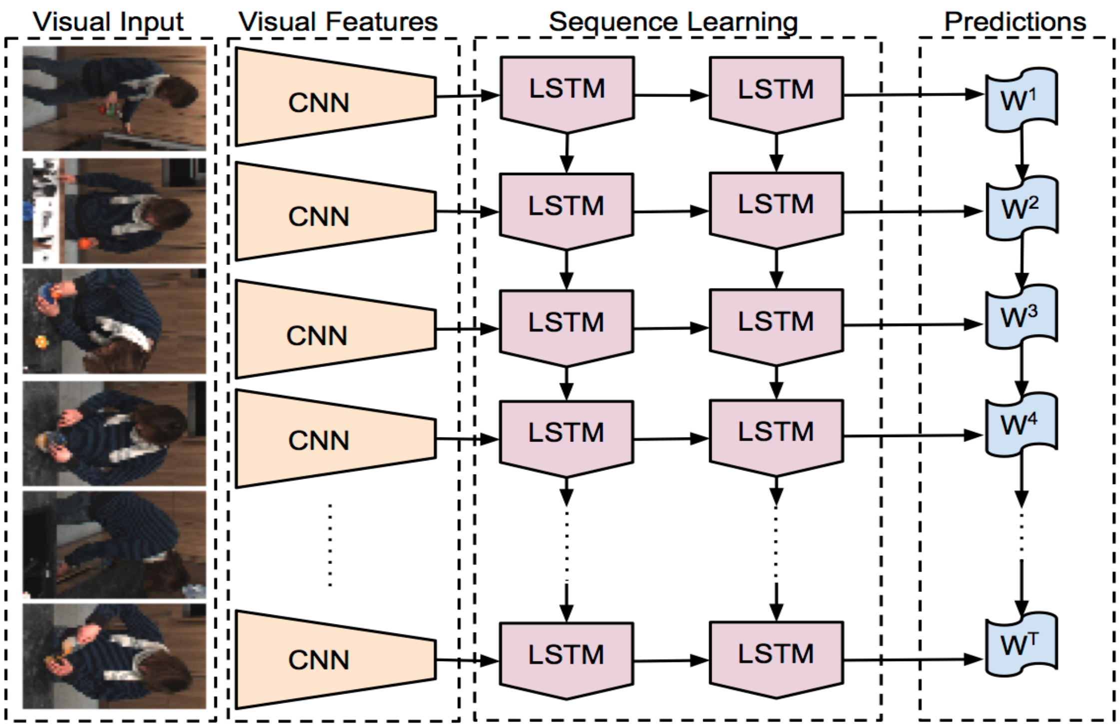
We localize the action as well.

Baccouche et al. ICANN10
Donahue et al. / Ng et al. CVPR15
Srivastava et al. ICML15

LSTM

LSTM models sequential memories in the long and short term

Use ConvNet fc **vectors** as input, no spatial information encoded



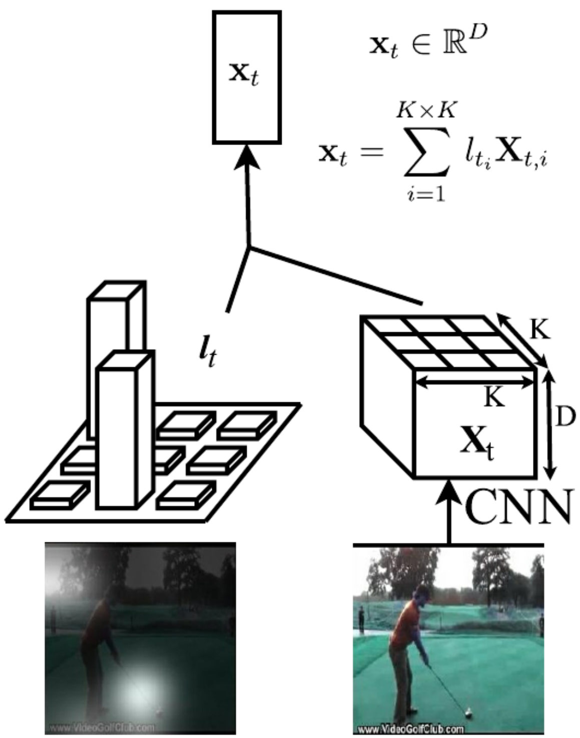
Rather than vectorizing a video frame, we rely on **convolutions**

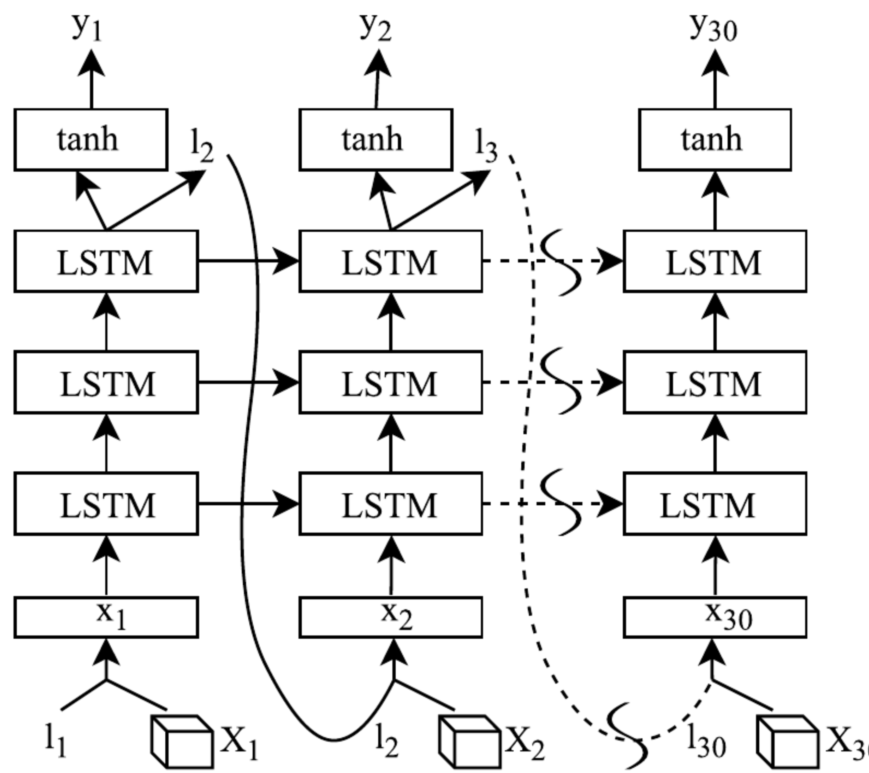
A(ttention)LSTM

Look for best locations leading to correct action classification

Stays close to soft-Attention architecture for image captioning [Xu et al. ICML15],

Vectorizes attention and appearance, and ignores the motion inside a video.



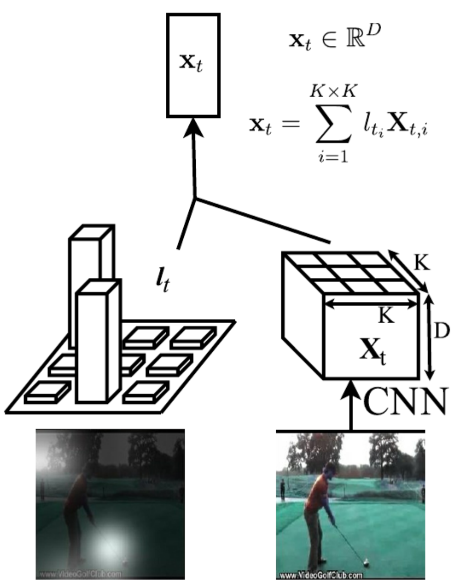


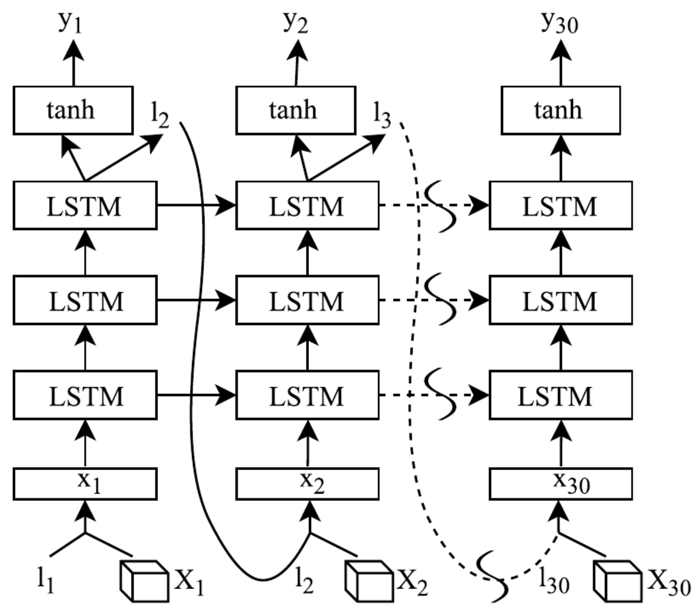
A(ttention)LSTM

Look for best locations leading to correct action classification

Stays close to soft-Attention architecture for image captioning [Xu et al. ICML15],

Vectorizes attention and appearance, and ignores the motion inside a video.





We add **convolutions** and **motion** for better action classification

We localize the action as well.

Our proposal: VideoLSTM

Model spatiotemporal dynamics of videos by

- preserving spatial structure of the frames over time
- adding motion-based attention
- enabling action localization from action class labels only

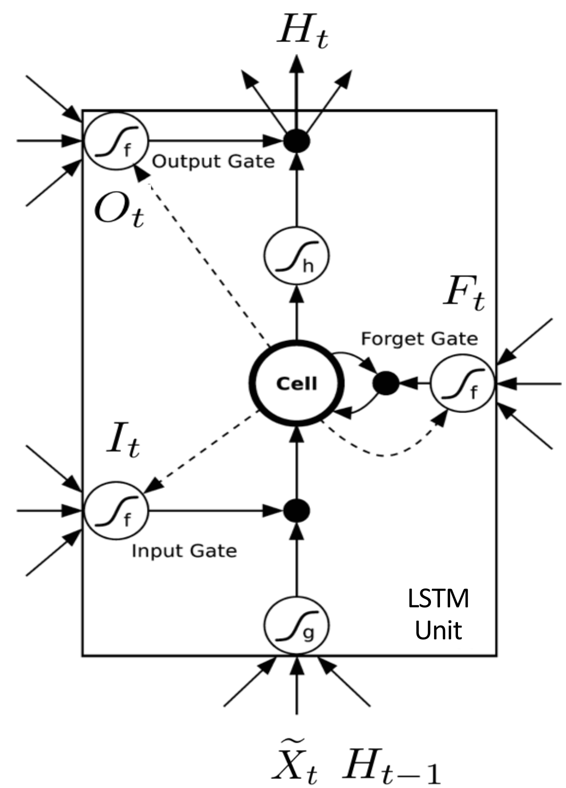
VIDEOLSTM

VideoLSTM convolves, attends and flows for action recognition.
Zhenyang Li, Efstratios Gavves, Mihir Jain, and Cees Snoek. Arxive16.
http://arxiv.org/abs/1607.01794

Convolutional (A)LSTM

Replace the fully connected multiplicative operations in an LSTM unit with convolutional operations

$$
\begin{aligned}
I_t &= \sigma(W_{xi} * \widetilde{X}_t + W_{hi} * H_{t-1} + b_i) \\
F_t &= \sigma(W_{xf} * \widetilde{X}_t + W_{hf} * H_{t-1} + b_f) \\
O_t &= \sigma(W_{xo} * \widetilde{X}_t + W_{ho} * H_{t-1} + b_o) \\
G_t &= \tanh(W_{xc} * \widetilde{X}_t + W_{hc} * H_{t-1} + b_c) \\
C_t &= F_t \odot C_{t-1} + I_t \odot G_t \\
H_t &= O_t \odot \tanh(C_t),
\end{aligned}
$$



Generate attention by shallow ConvNet instead of MLP

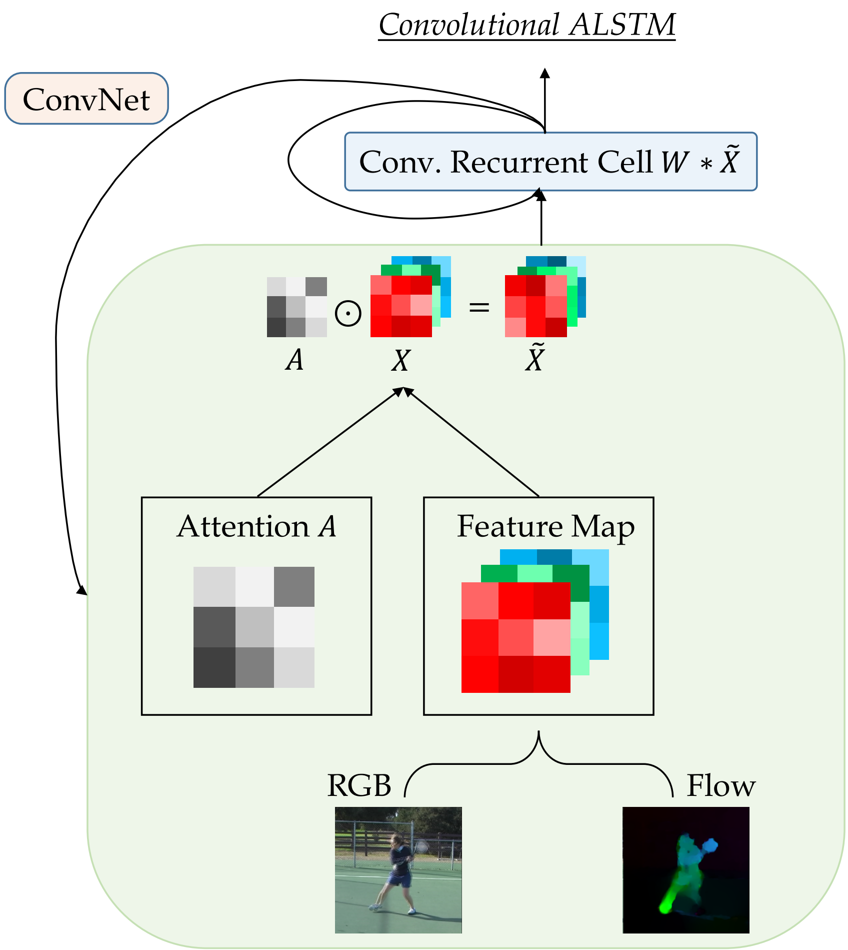
Convolutional ALSTM

Attention map is generated by a two-layer ConvNet

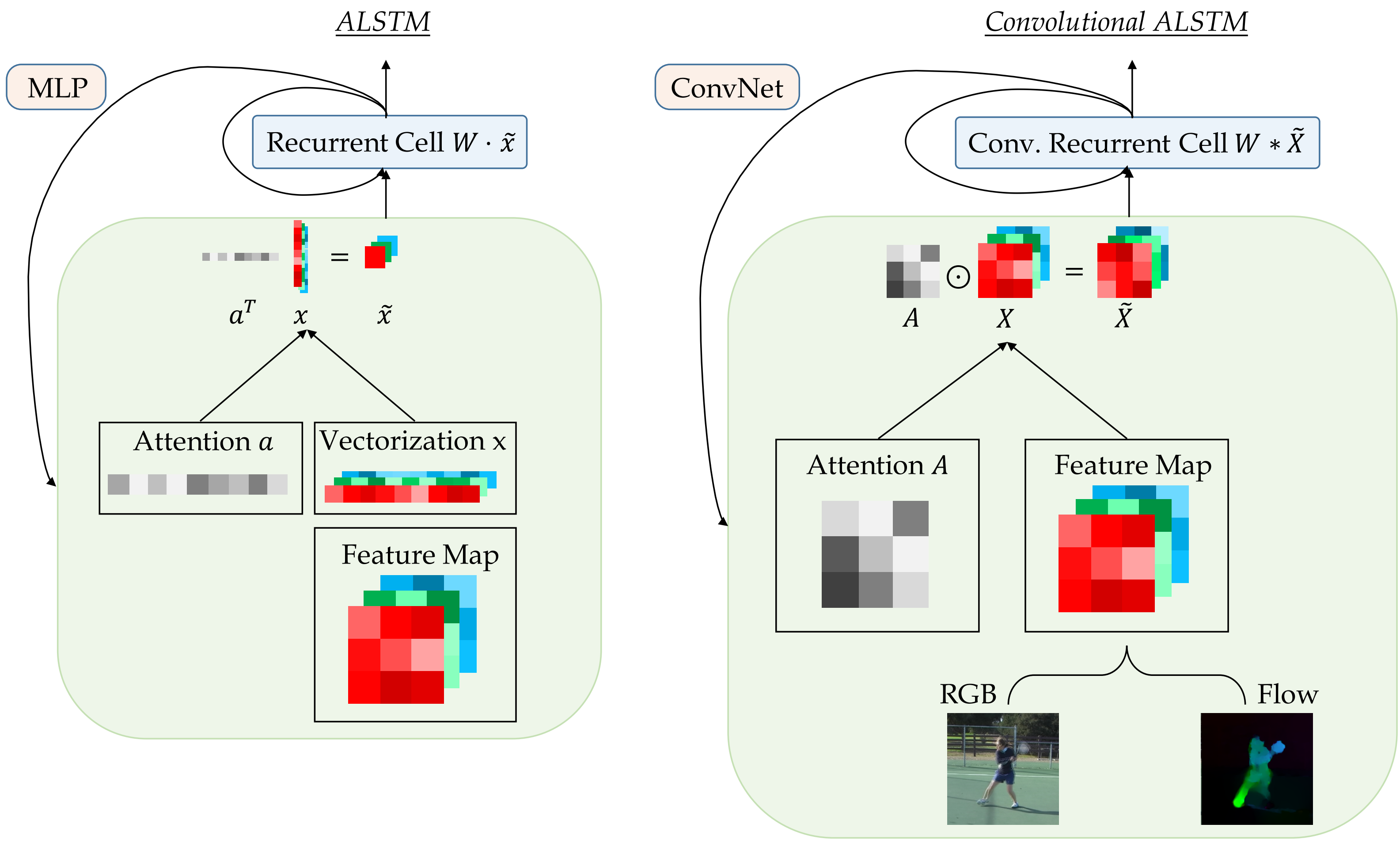
$$Z_t = W_z * \tanh(W_{xa} * X_t + W_{ha} * H_{t-1} + b_a)$$

$$A_t^{ij} = p(att_{ij} | X_t, H_{t-1}) = \frac{\exp(Z_t^{ij})}{\sum_i \sum_j \exp(Z_t^{ij})}$$

$$\widetilde{X}_t = A_t \odot X_t$$



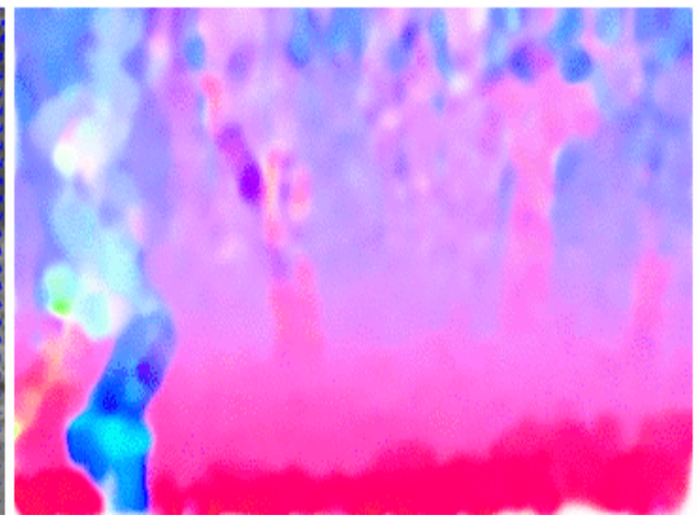
ALSTM vs Convolutional ALSTM



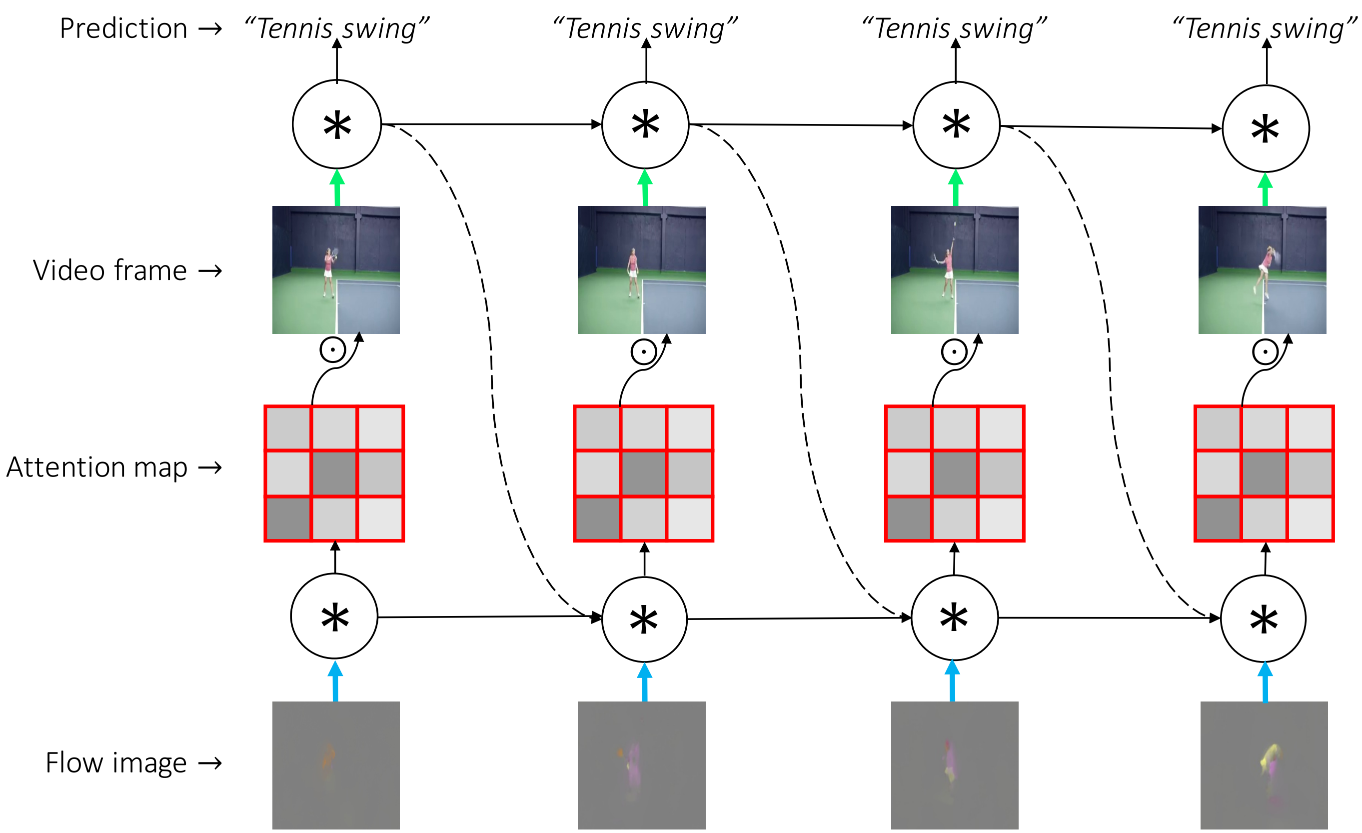
Convolutional ALSTM preserves spatial dimensions over time

Motion-based attention

Motion offers crucial clue where to attend in video



Motion-based attention



Motion information to infer the attention in each frame

Experimental setup

Datasets:

UCF101, **HMDB51** for action classification

Comparison using similar designs and training regime:

ConvNet: VGG-16 trained for both RGB frame and optical flow.

LSTM: Use subsequences of every 30 frames, extract fc7 or pool5 features at each frame as input.

Convolutions: 3x3 kernels for input-to-state and state-to-state transitions in LSTM, and 1x1 kernels to generate the attention map

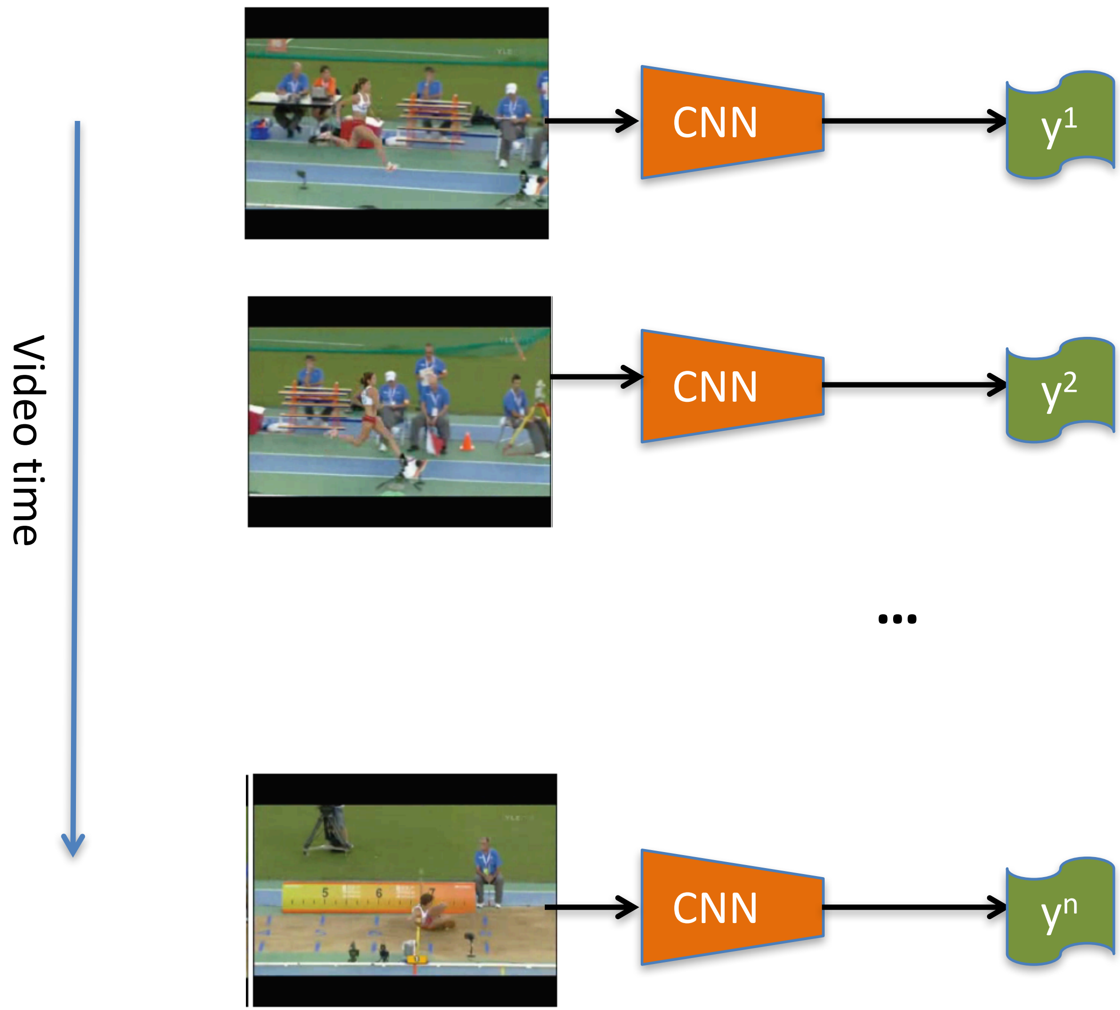
Experiments

1. What deep learning architecture?

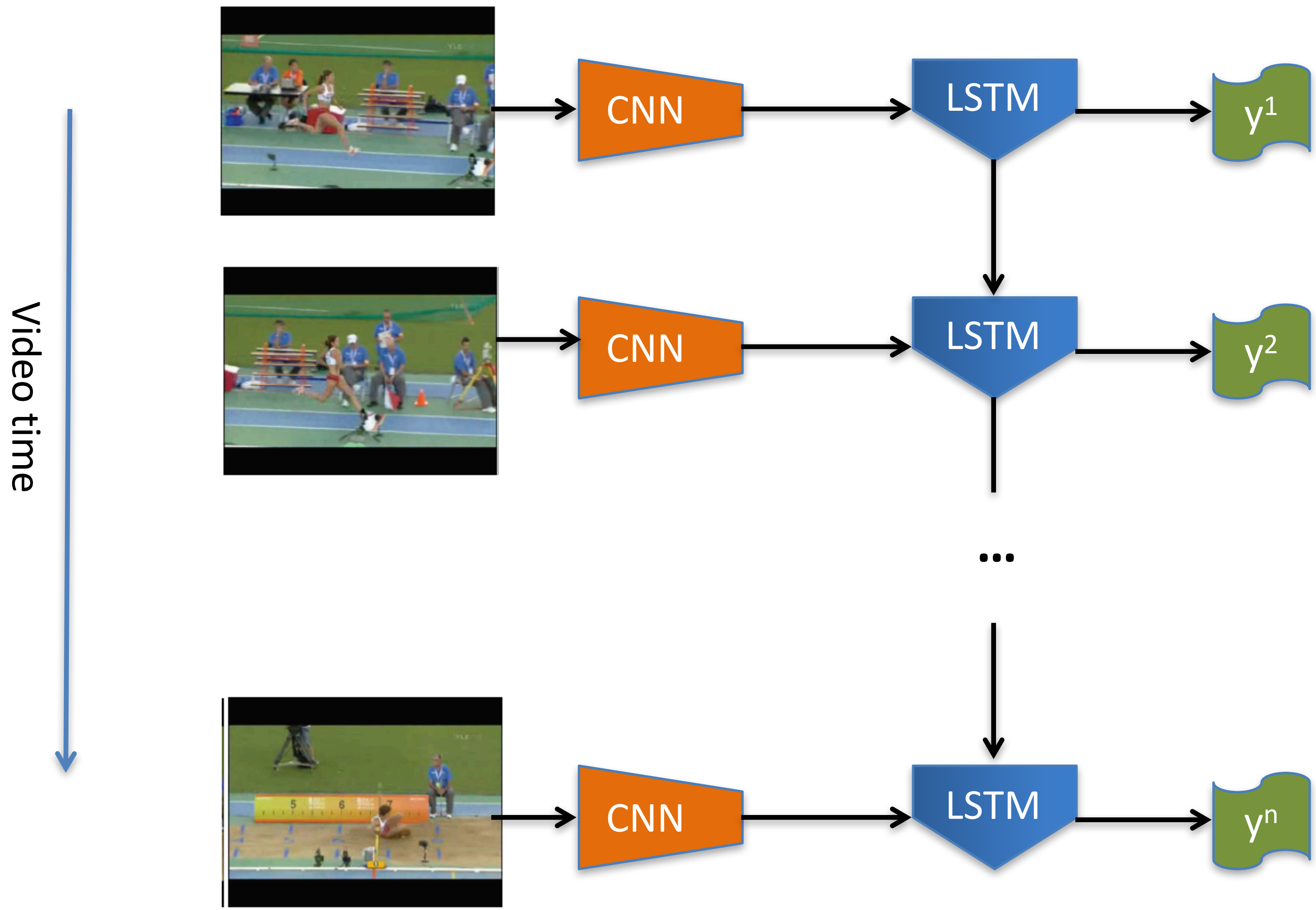
2. Influence of motion-based attention

3. Quality of action localization

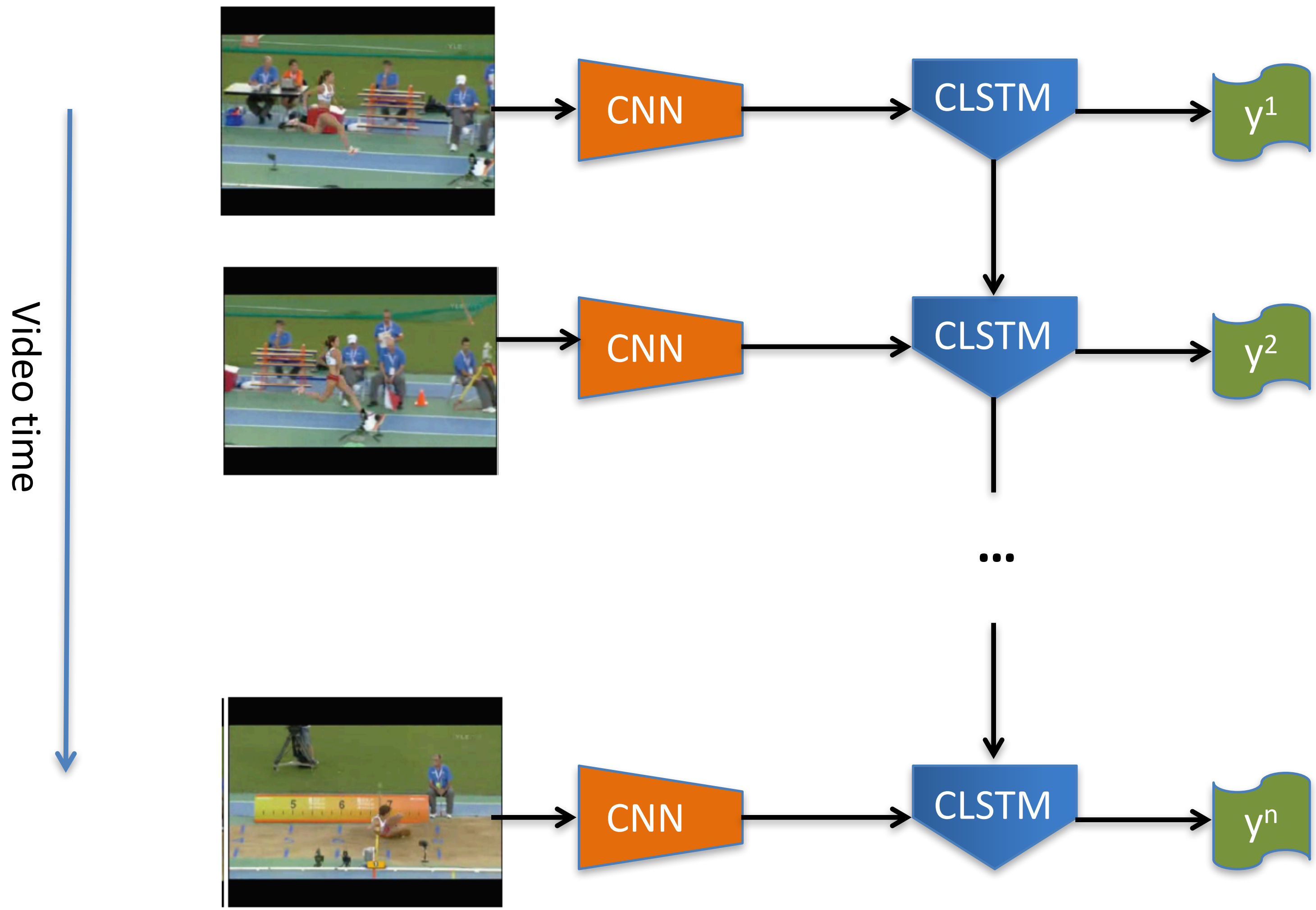
ConvNet



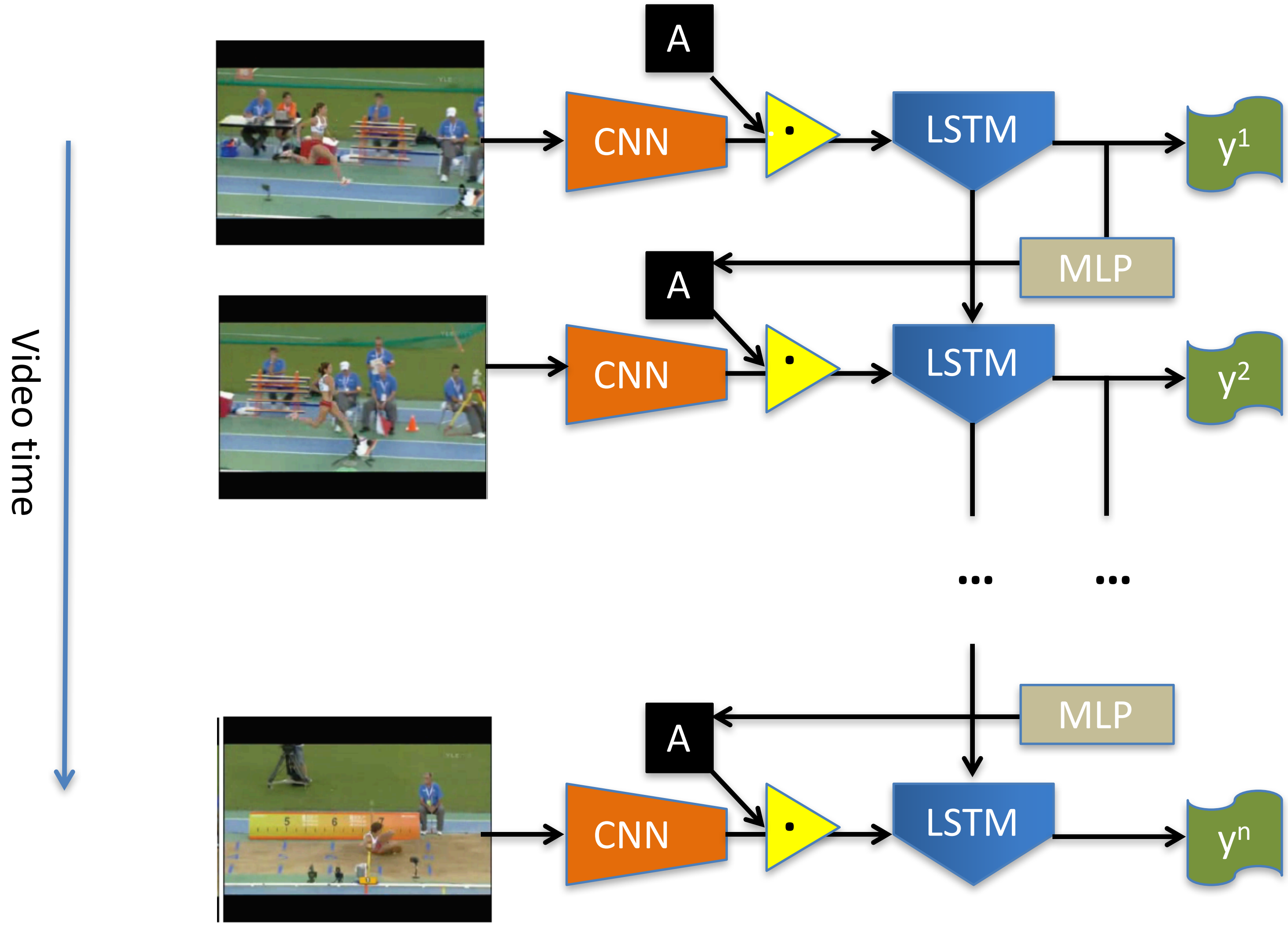
LSTM



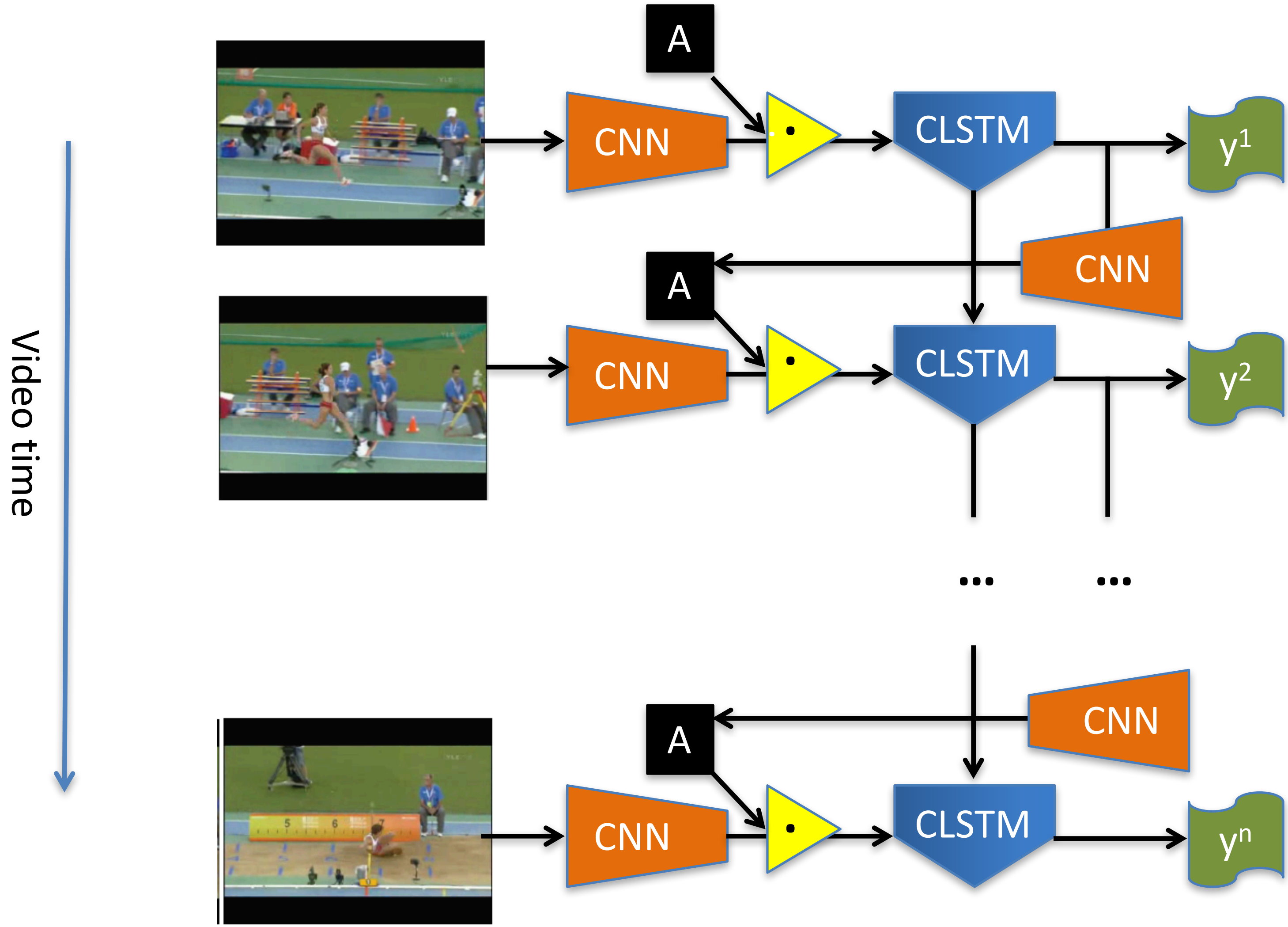
Convolutional LSTM



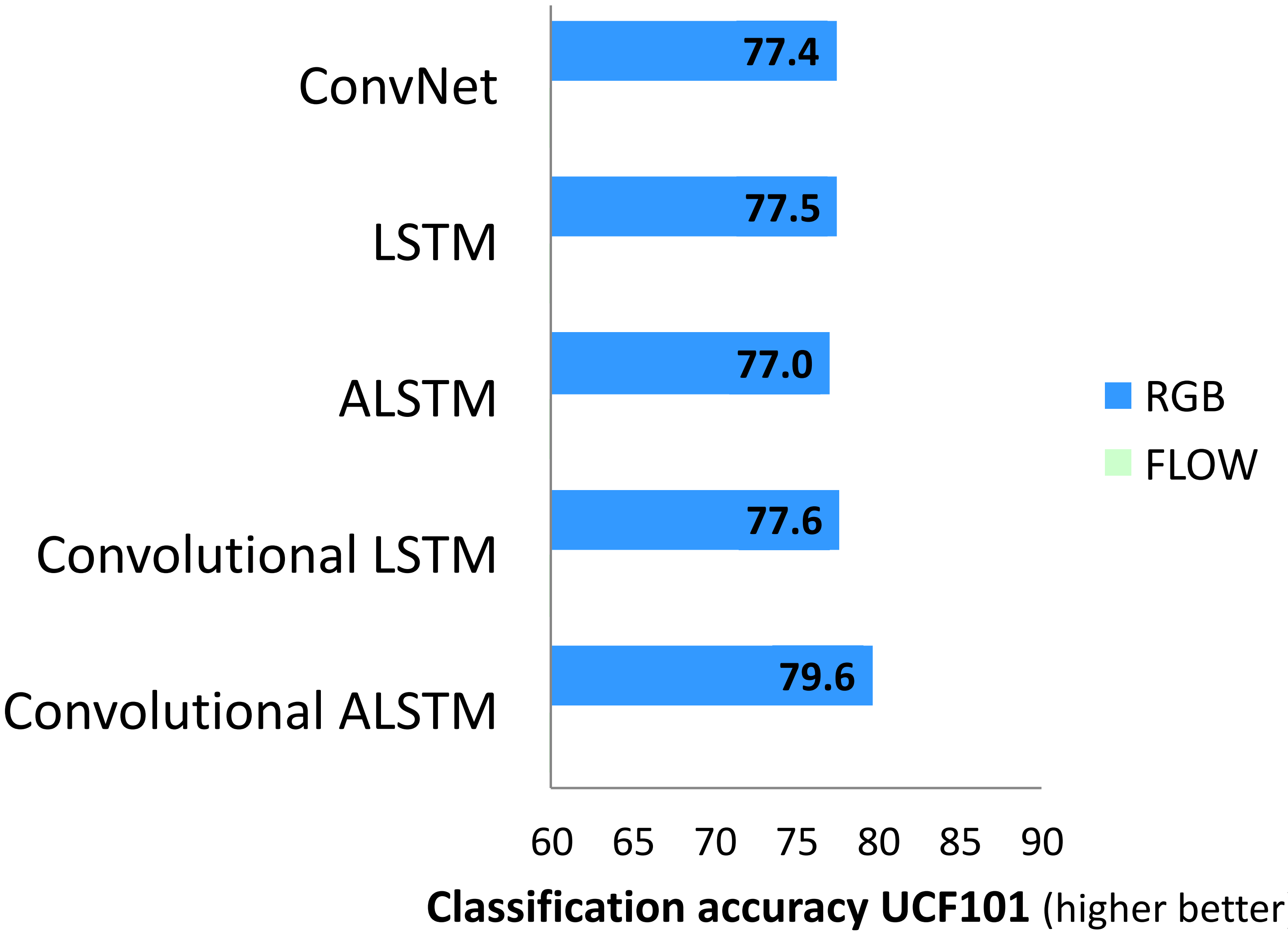
ALSTM



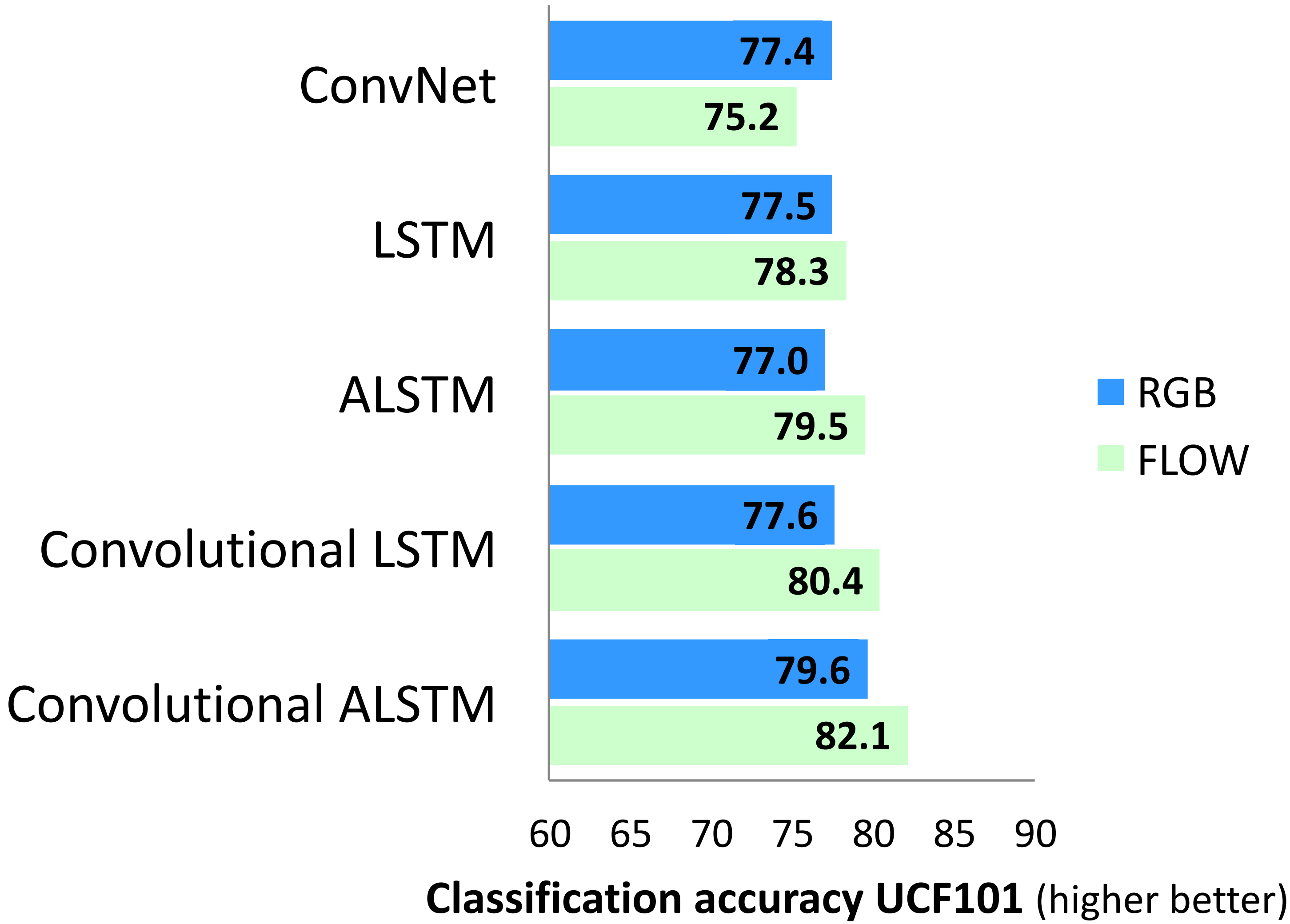
Convolutional ALSTM



Convolution, attention and flow



Convolution, attention and flow



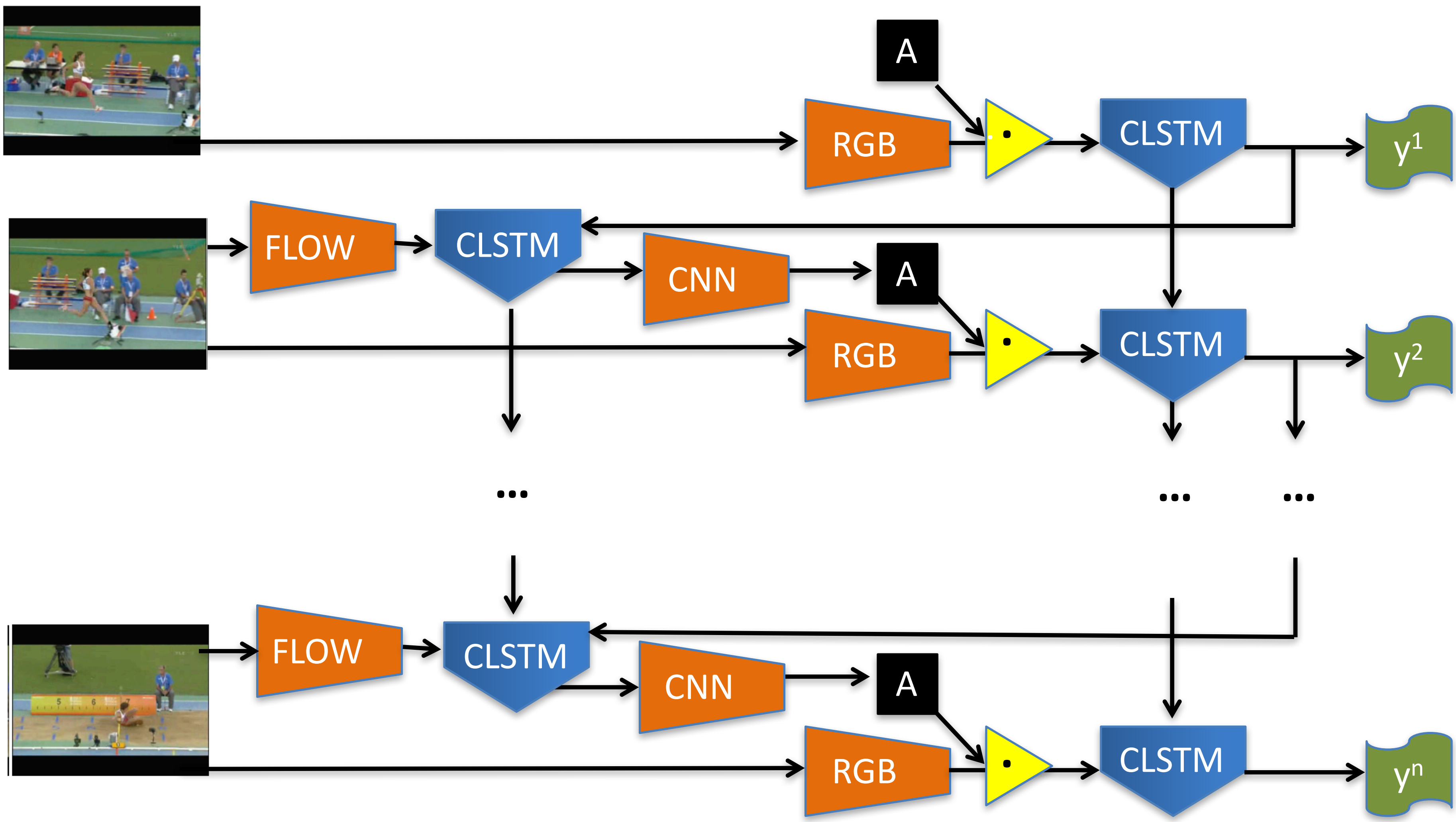
Experiments

1. What deep learning architecture?

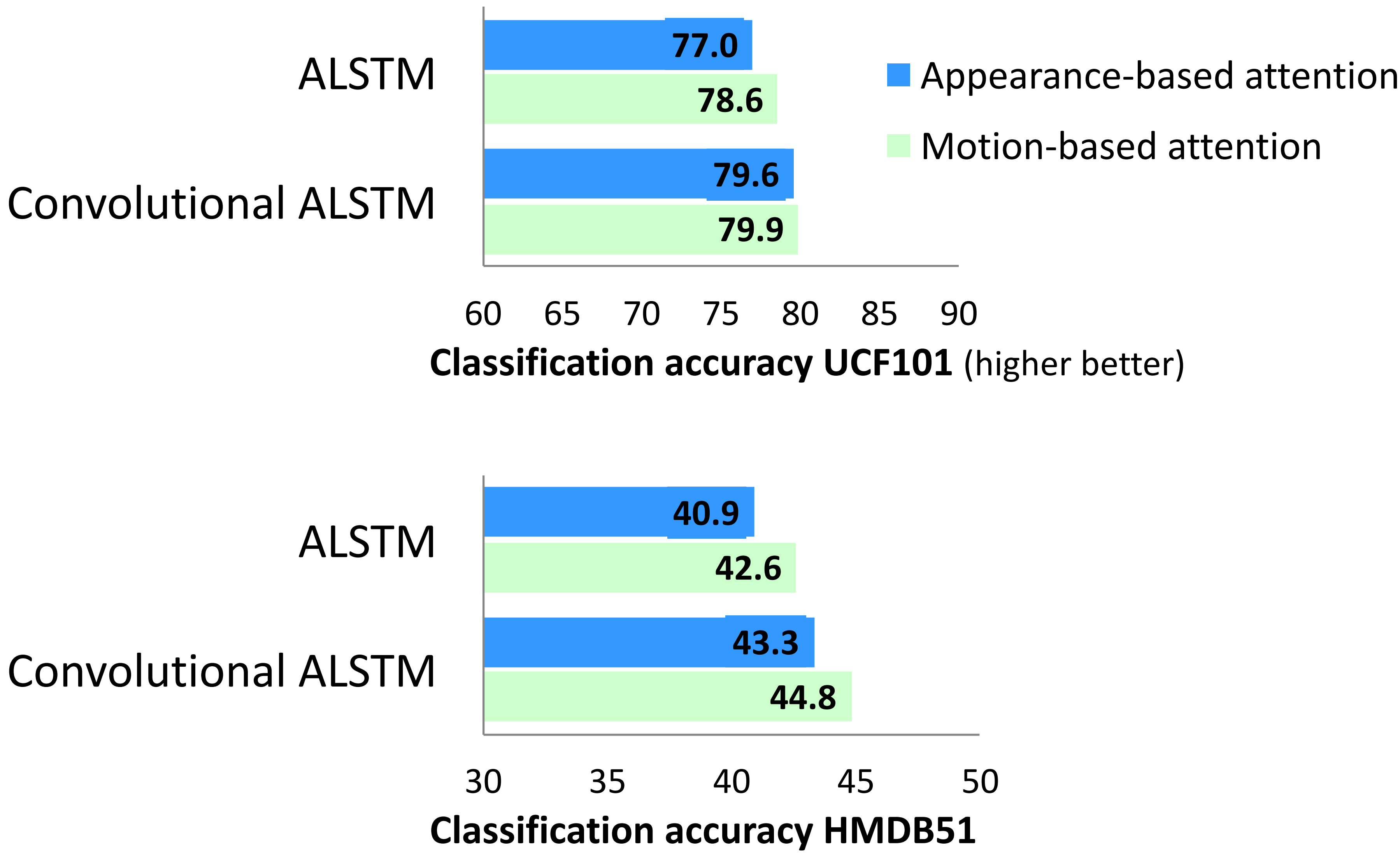
2. Influence of motion-based attention

3. Quality of action localization

Recap: Motion-based attention

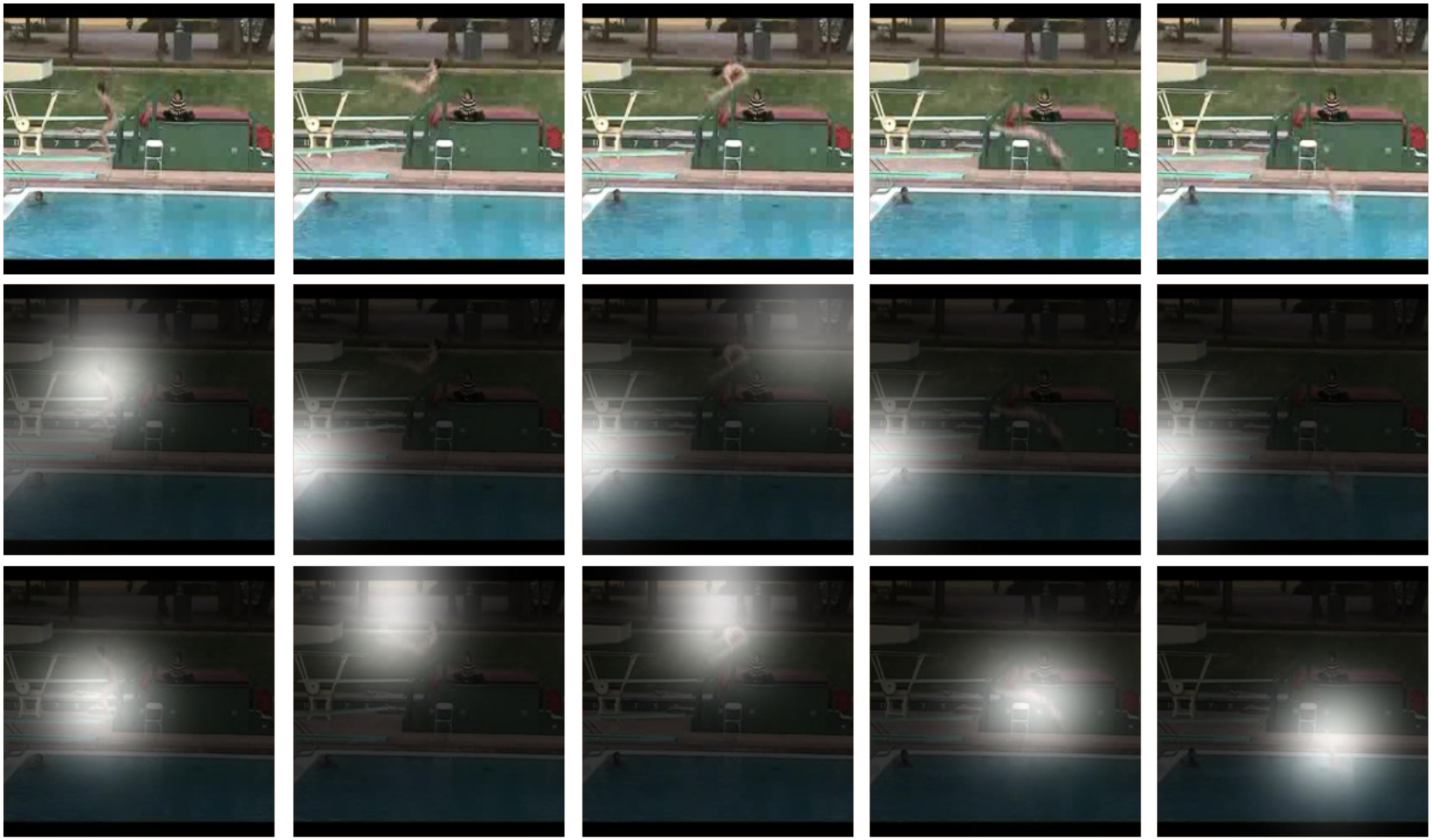


Motion attention makes more sense



Motion attention makes more sense

Motion attention makes more sense



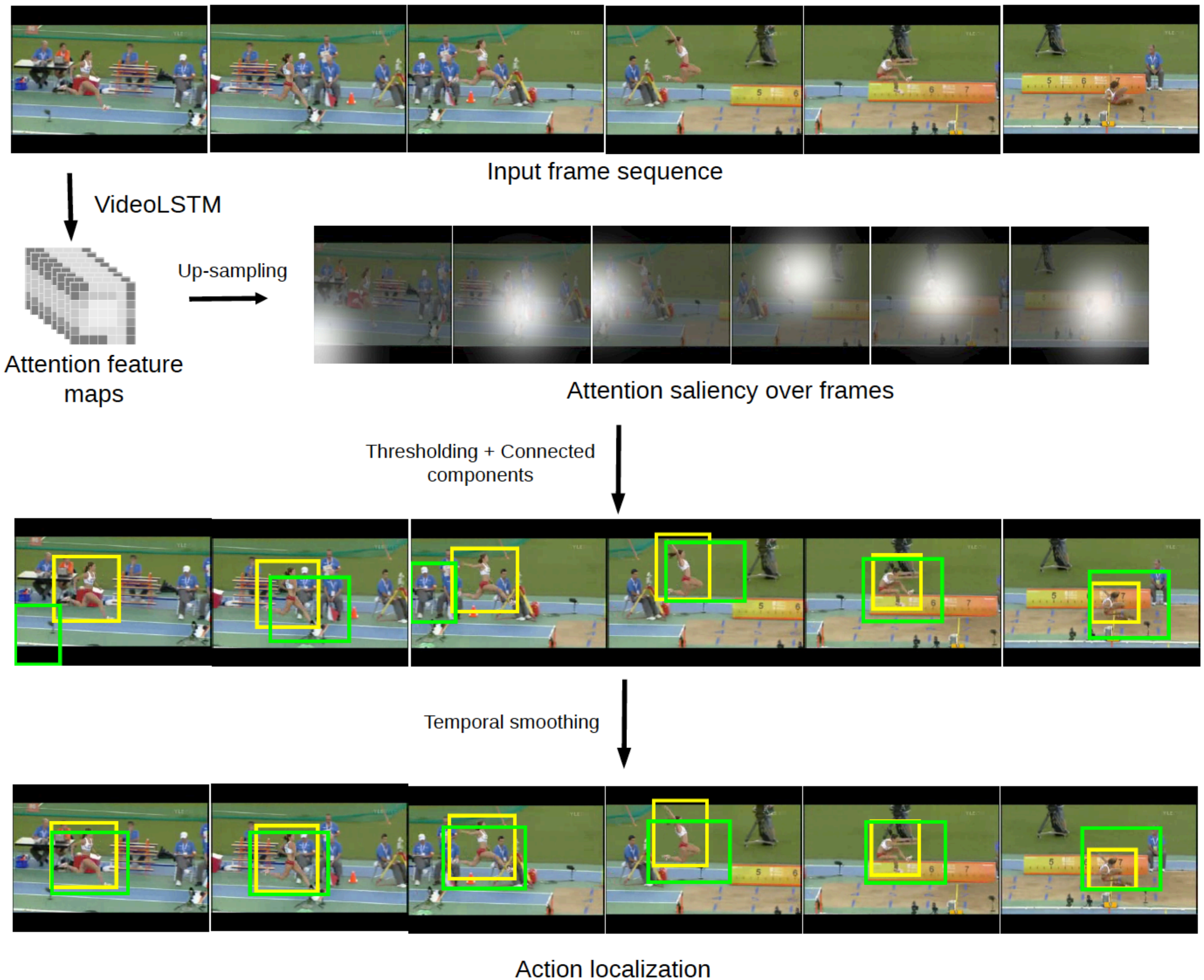
Experiments

1. What deep learning architecture?

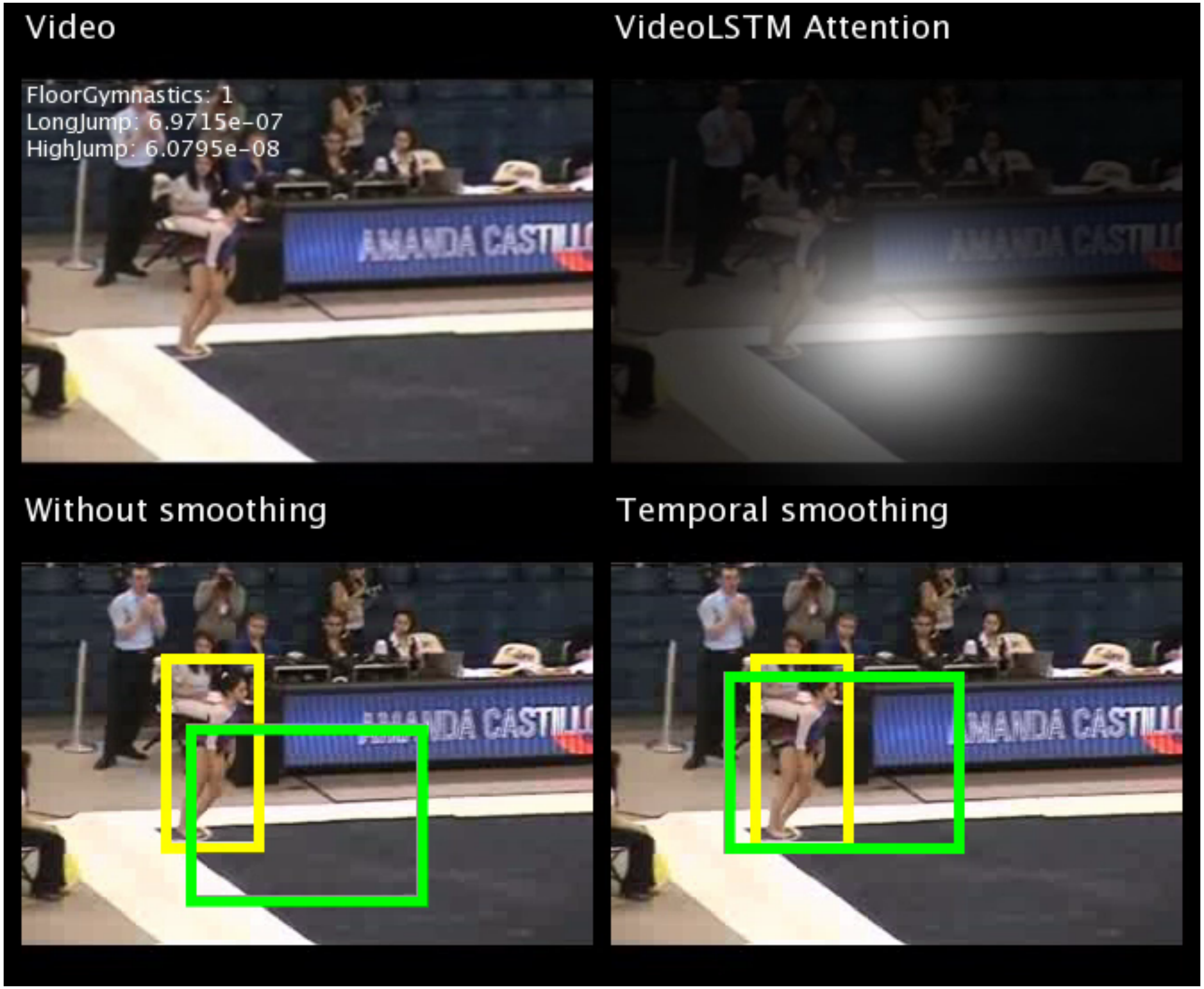
2. Influence of motion-based attention

3. Quality of action localization

Temporal smoothing

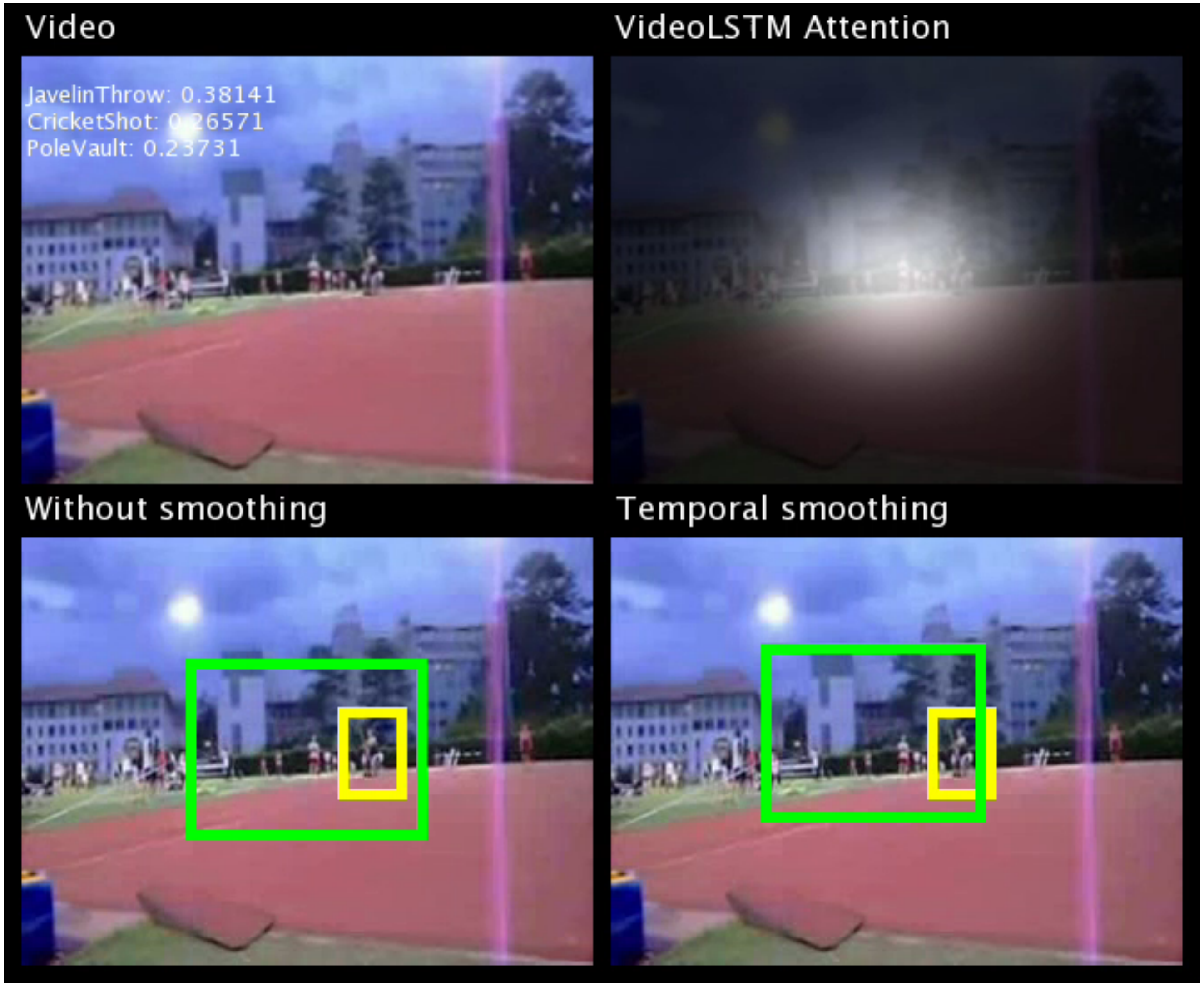


Qualitative results



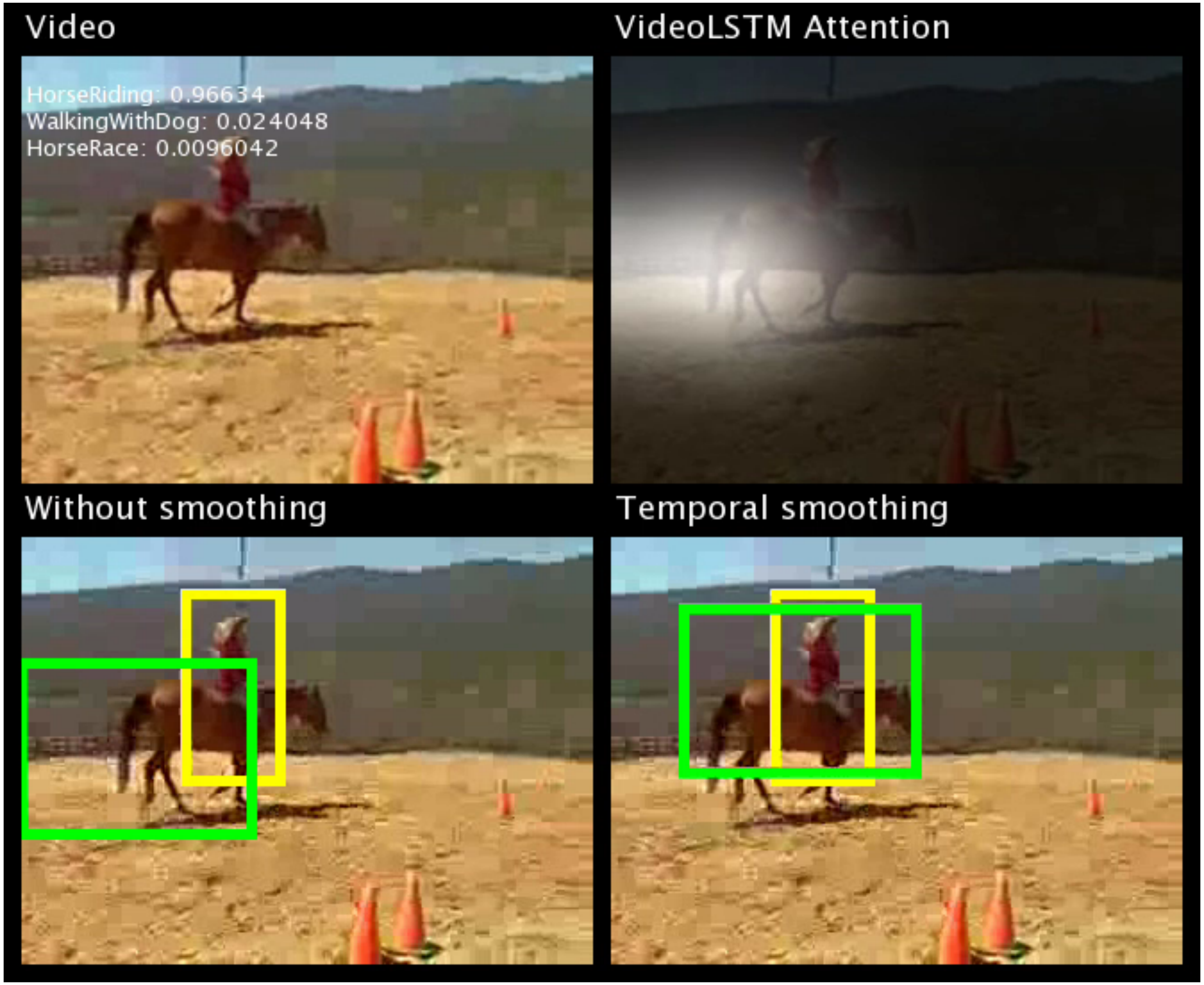
http://isis-data.science.uva.nl/zhenyang/videolstm/localization_examples/

Qualitative results



http://isis-data.science.uva.nl/zhenyang/videolstm/localization_examples/

Qualitative results



http://isis-data.science.uva.nl/zhenyang/videolstm/localization_examples/

Conclusions on VideoLSTM

Promising deep vision architecture for action localization

- Hardwires convolutions in attention LSTM
- Derives attention from what moves in video

Localization from a video-level action class label only

http://arxiv.org/abs/1607.01794

Siamese Instance Search for Tracking

Ran Tao, Efstratios Gavves, Arnold Smeulders

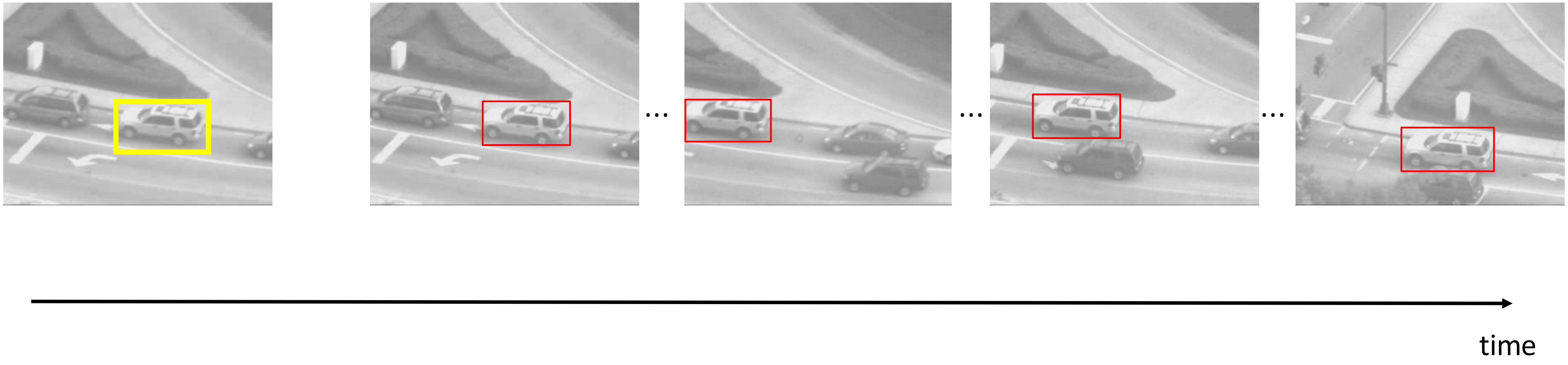
QUVA Deep Vision Lab

UNIVERSITEIT VAN AMSTERDAM

Ran Tao, Efstratios Gavves, Arnold Smeulders. *Siamese Instance Search for Tracking*. CVPR, 2016, Las Vegas.

(Single) Visual Object Tracking

Track the target's positions over time in a video, given a starting box in 1st frame



Applications

- Surveillance
- Robotics
- Human-computer Interaction
- Autonomous Driving
- Drones

Tracking is hard

- Start from 1 snapshot of the target
- But the target may change its appearance significantly due to illumination variation, scale change, rotation, etc. *[Smeulders et al, TPAMI, 2014: 13 hard aspects]*

- Track the ‘thing’ in the bounding box (i.e. unknown object)
- Unknown environment

How to handle the appearance variations of the target?

Prevalent paradigm in literature

Starting from the 1st frame, learn and update a target model on-the-fly

- **Target model**: target/non-target binary classifier, regressor
- **Update** the model using <u>the data inferred by the tracker itself</u>

Prevalent paradigm in literature

Starting from the 1st frame, learn and update a target model on-the-fly

- **Target model**: target/non-target binary classifier, regressor
- **Update** the model using the data inferred by the tracker itself

The data inferred by the tracker itself are not absolutely reliable → drifting

The proposed tracker: motivation

Since the only reliable data is the initial target region in the first frame, the proposed tracker only relies on the initial target. (no update)

The proposed tracker: motivation

Since the only reliable data is the initial target region in the first frame, the proposed tracker only relies on the initial target. (no update)

Then how to handle the appearance variations?

The proposed tracker: motivation

Since the only reliable data is the initial target region in the first frame, the proposed tracker only relies on the initial target. (no update)

Then how to handle the appearance variations?

Certain objects change appearance over time in a similar way. ➔
Can we learn a comparison mechanism (similarity metric) a priori, that is robust against typical appearance variations an object may have in videos?

Siamese INstance search Tracker (SINT)

candidate patches

Initial patch

$$\underset{j}{\operatorname{argmax}}\ m(initialpatch, patch_j)$$

predicted patch

Siamese INstance search Tracker (SINT)

Simply tracks the target by retrieving in every frame the candidate most similar to the initial target in the first frame

- No online updating
- No occlusion detection
- No geometric matching
- No combination of trackers

But still delivers state-of-the-art tracking performance (at the publication time).

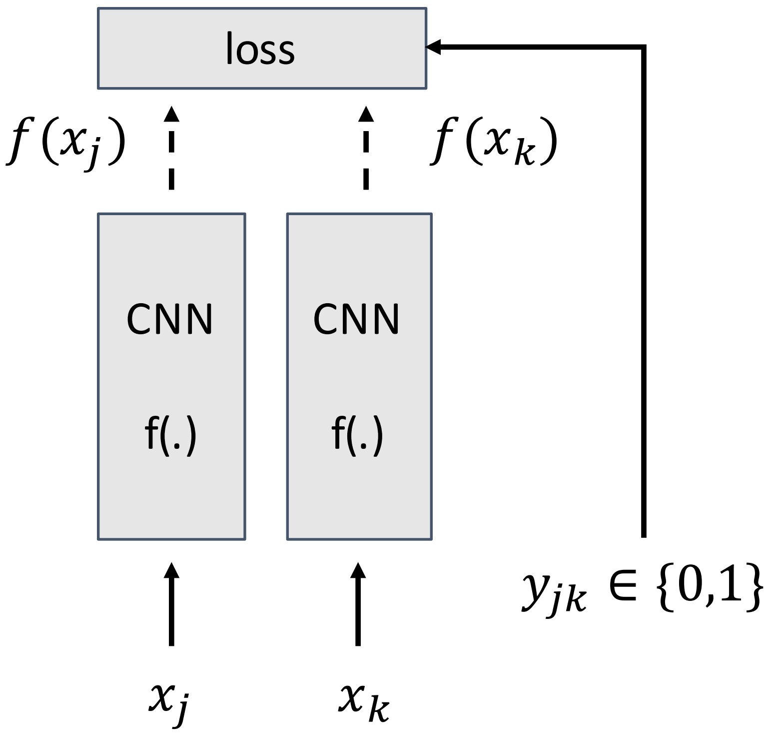
Strength is from the similarity function $m(\cdot,\cdot)$ learned offline using **Siamese network**.

Siamese INstance search Tracker (SINT)

Learn once on a rich video dataset with box annotations following an object.

Once learned, it is applied as is, without any further adapting, to track any previously unseen targets.

Similarity Function Learning



Marginal Contrastive Loss:

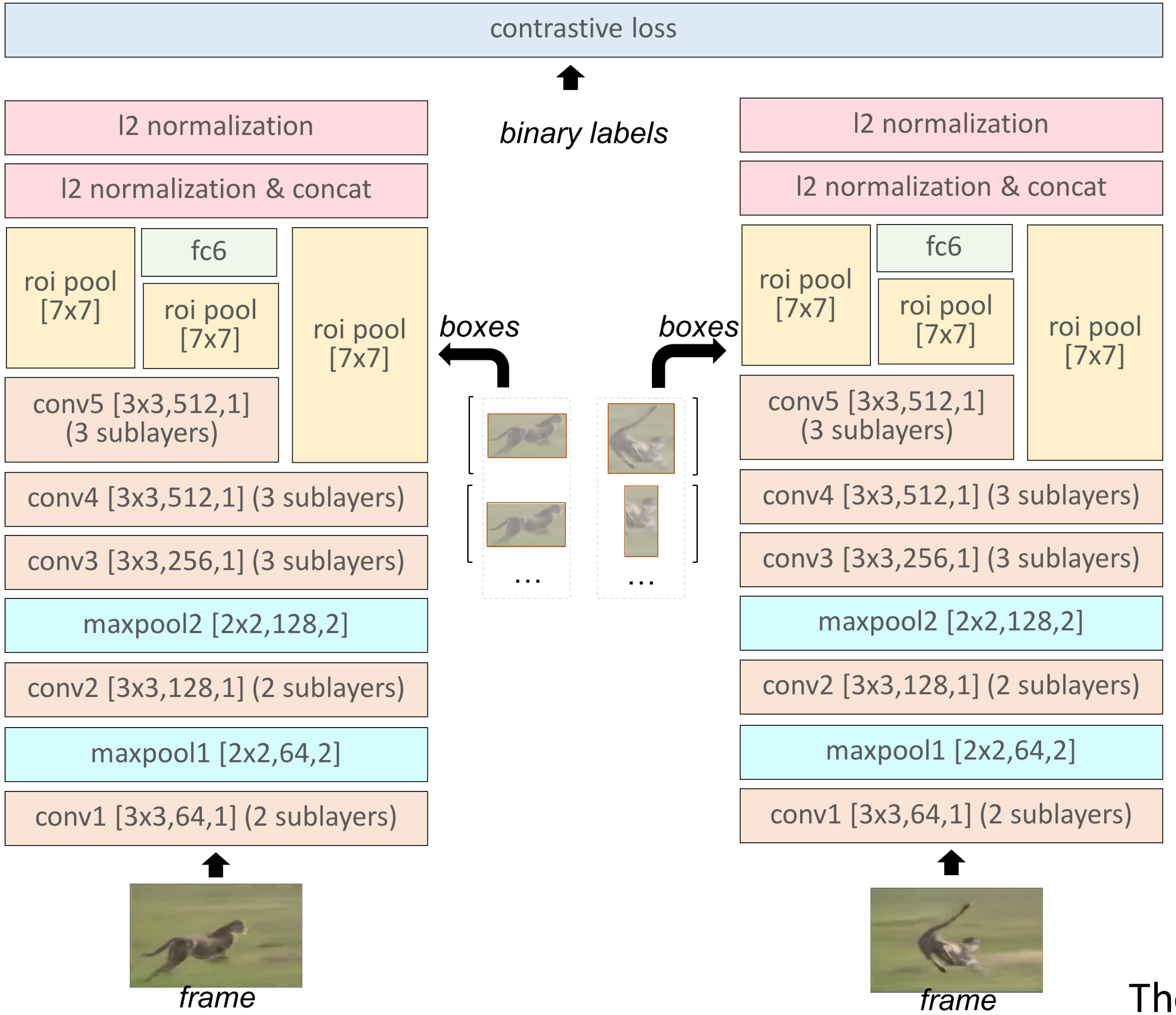
$$L(x_j, x_k, y_{jk}) = \frac{1}{2} y_{jk} D^2 + \frac{1}{2}(1 - y_{jk}) \max(0, \sigma - D^2)$$

$$D = \|f(x_j) - f(x_k)\|_2$$

Similarity function (after learning):

$$m(x_j, x_k) = f(x_j) \cdot f(x_k)$$

Network Architecture

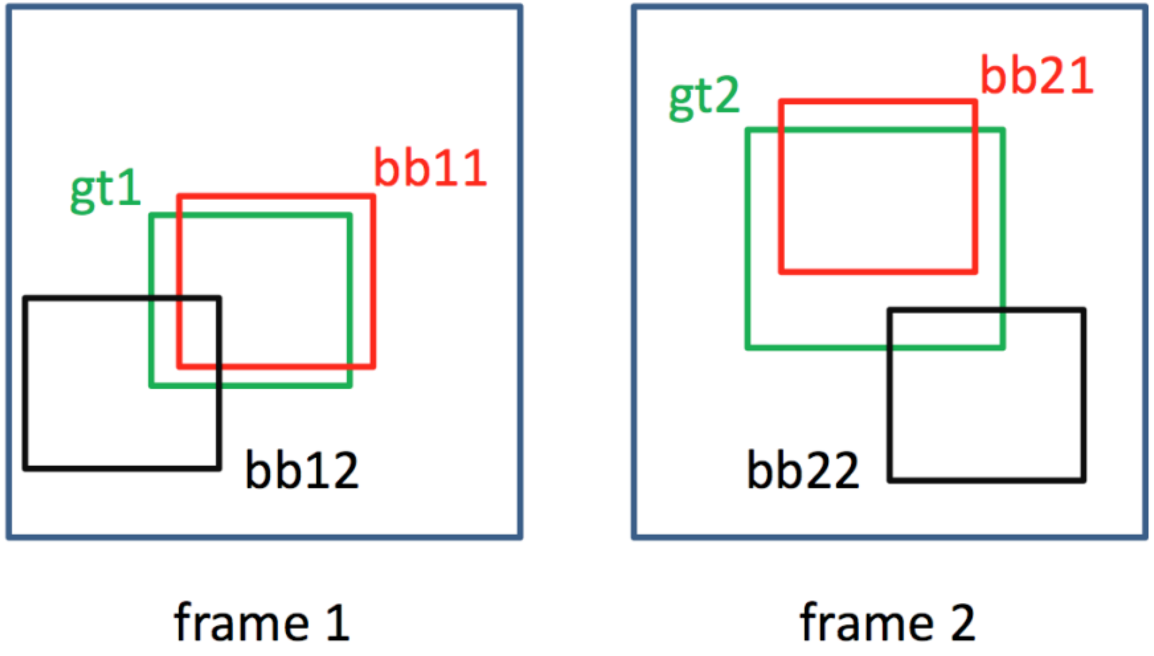


- Region-of-interest (ROI) pooling → process all boxes in a frame in one single pass through the network
- Very few max pooling → improve localization accuracy
- Use outputs of multiple layers (conv4_3, conv5_3, fc6) → to be robust in various situations (unknown environment)

The two branches share the parameters.

Training Pairs

Data: videos of objects with BBox annotation (ALOV)



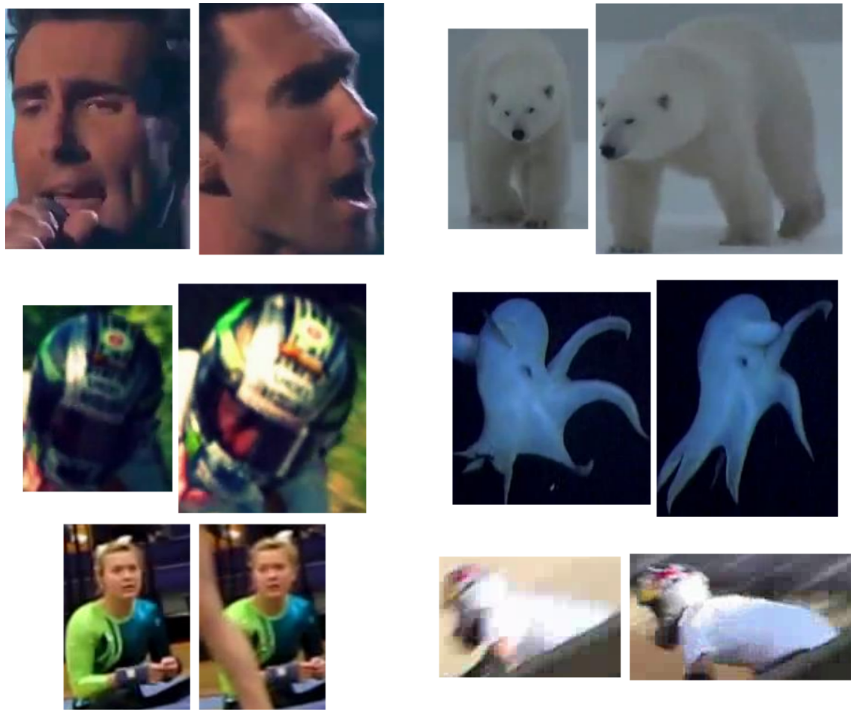
(gt1, gt2, 1)
(gt1, bb21, 1)
(gt1, bb22, 0)
(gt2, bb11, 1)
(gt2, bb12, 0)
...

>0.7, 1
<0.5, 0

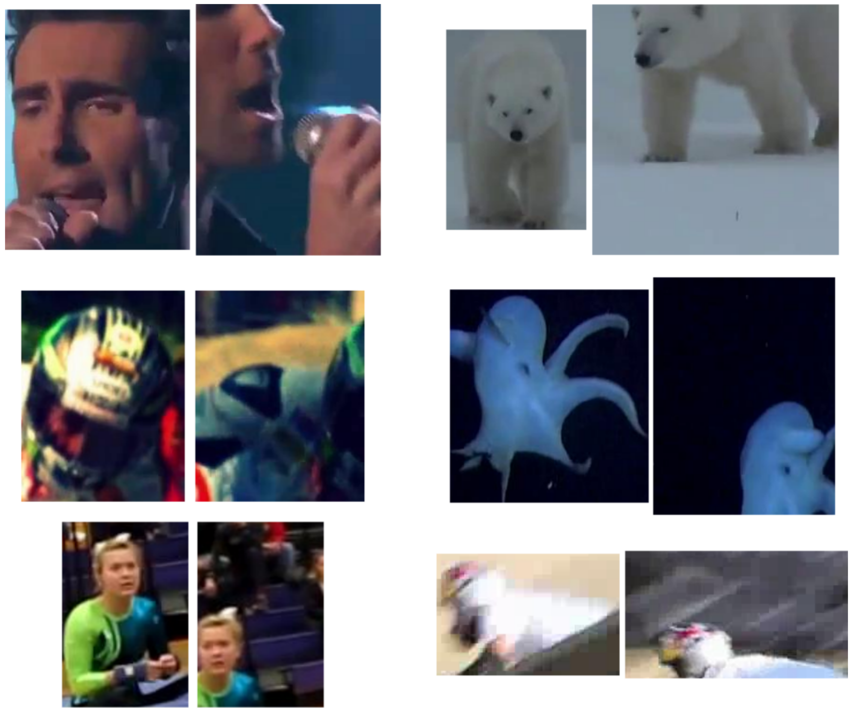
Smeulders *et al*, Visual tracking: An experimental survey, *TPAMI, 2014.*

Training Pairs

- 60,000 pairs of frames for training, 2,000 pairs for validation
- 128 pairs of boxes per pair of frames

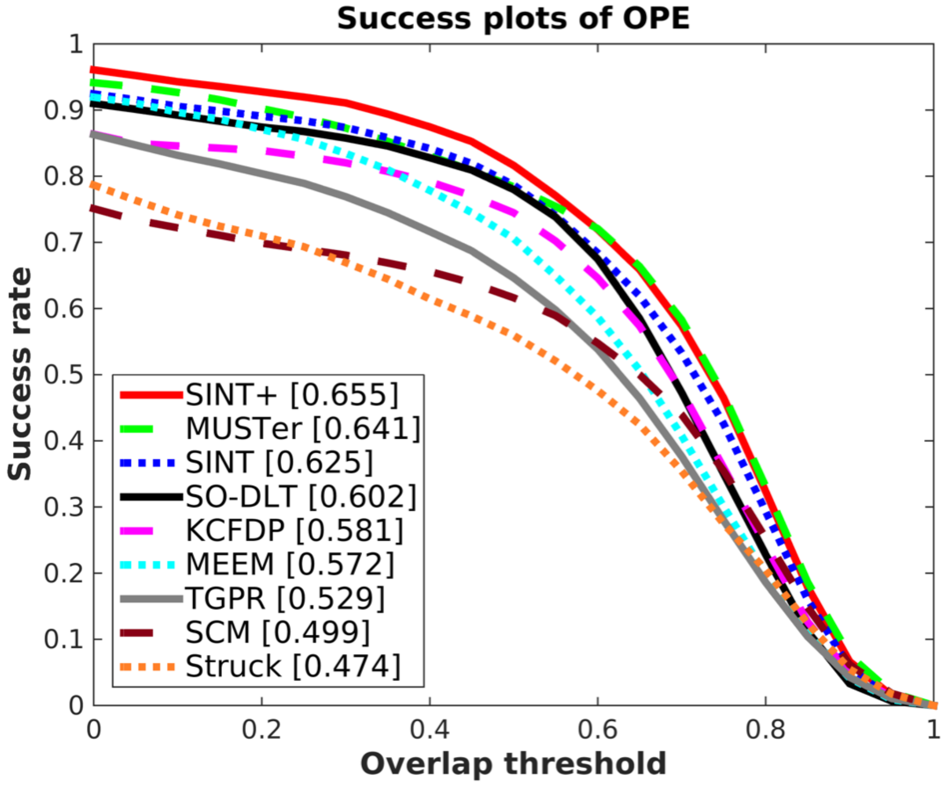


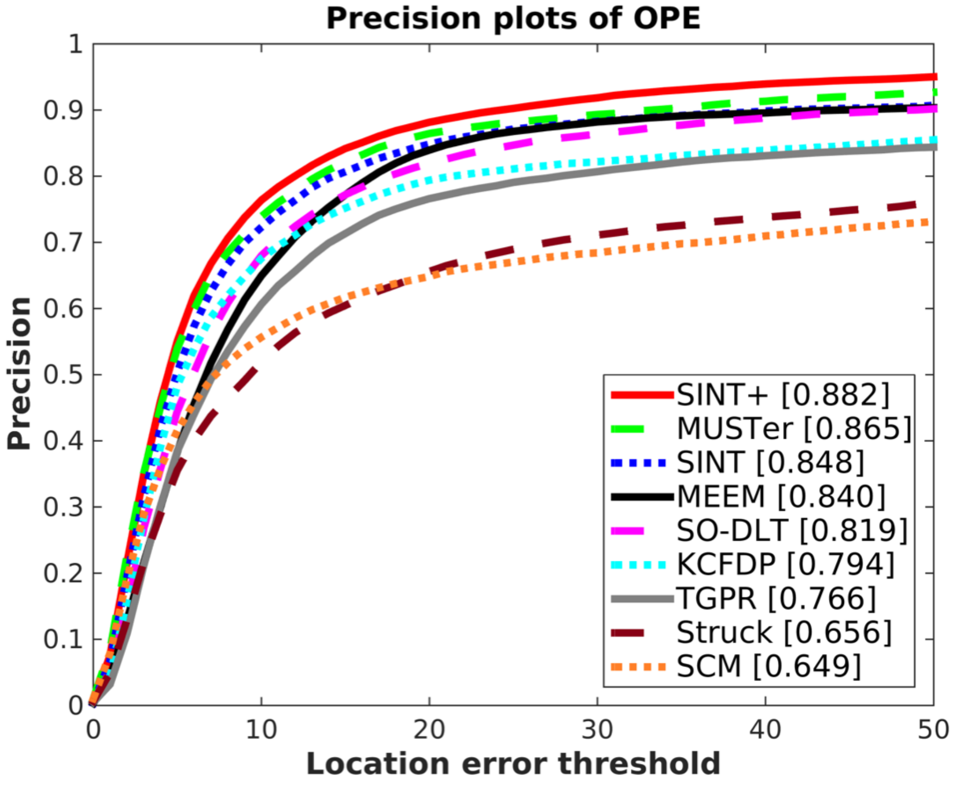
positive



negative

Results on OTB





SINT+: adaptive sampling range [Want et al, ICCV15] & optical flow to remove motion inconsistent samples

Large potential to improve SINT by integrating advanced online components

Qualitative Results

Can handle various types of appearance variations

The performance on subsequent frames will not be affected by the mistake made on the current frame.

Target Re-identification

- In the absent of any drifting, SINT allows for target re-identification after the target was absent for a long period of time, provided with a sampling over the whole image.

https://youtu.be/knaxUljyY_Q

Summary

- Siamese INstance search Tracker (SINT)
  - Retrieves in every frame the patch most similar to the 1 original patch of the target, nothing else
  - The strength is from the matching function, learned offline *generically*

- Allows target re-identification after the target was absent for a complete shot

- Establish **a new tracking framework**: it only requires one-time offline learning, and once learned, it is ready to track any new, previously unseen, targets, without any online learning.

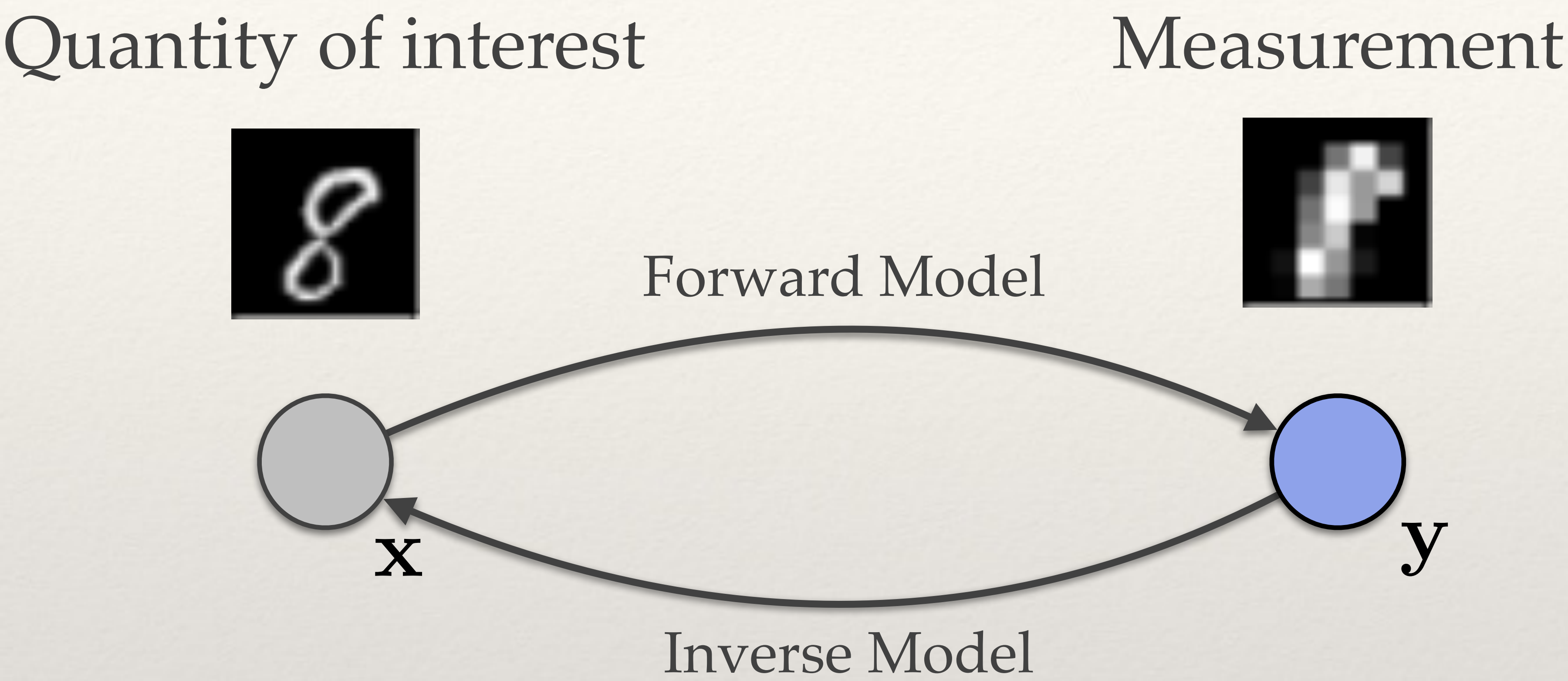
Patrick Putzky *& Max Welling*

Recurrent Inference Machines for Solving Inverse Problems

UNIVERSITY OF AMSTERDAM

Recurrent Inference Machines in Practice

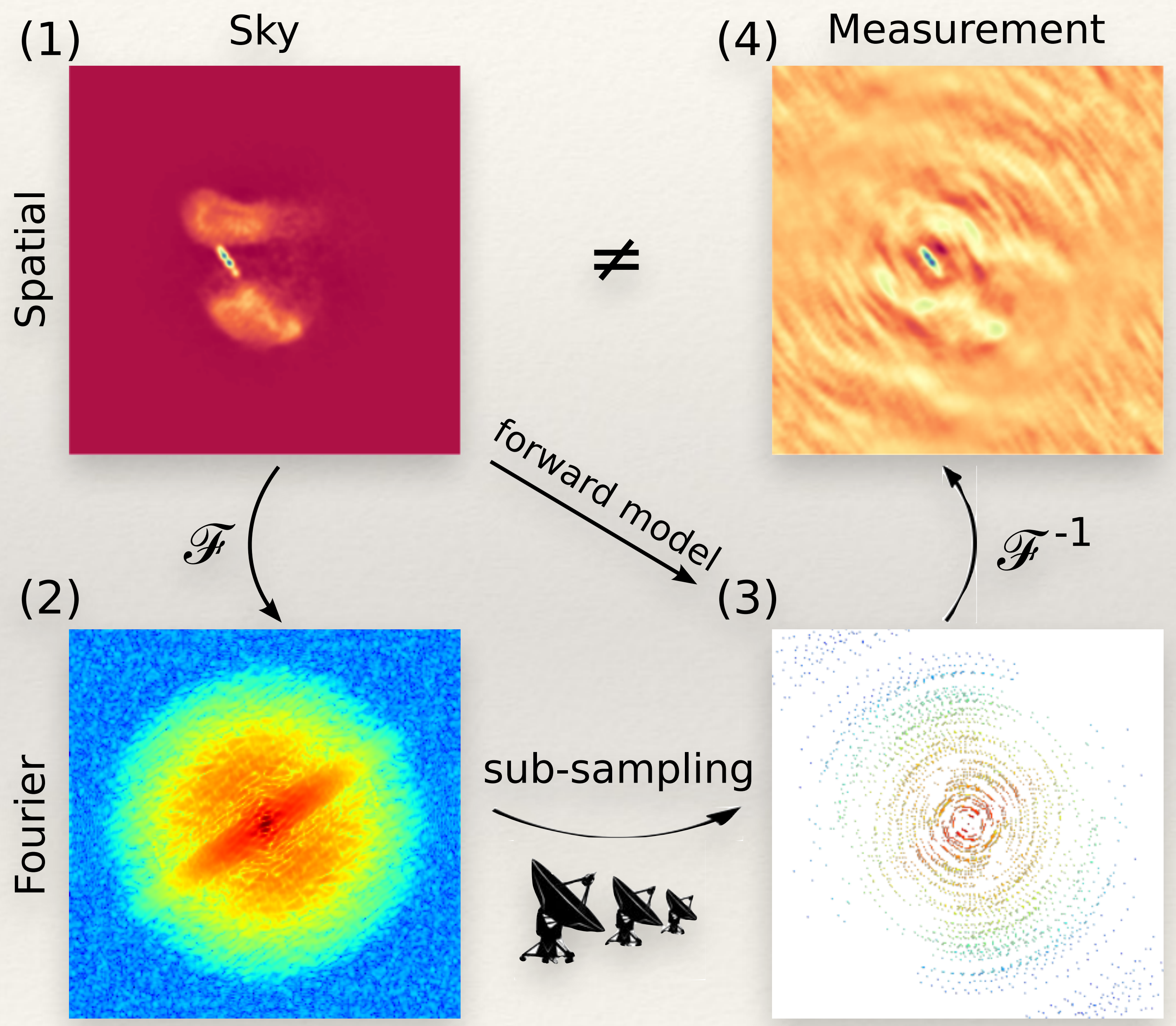
Inverse Problems



Forward Model $\mathbf{y} = g(\mathbf{x}) + n$

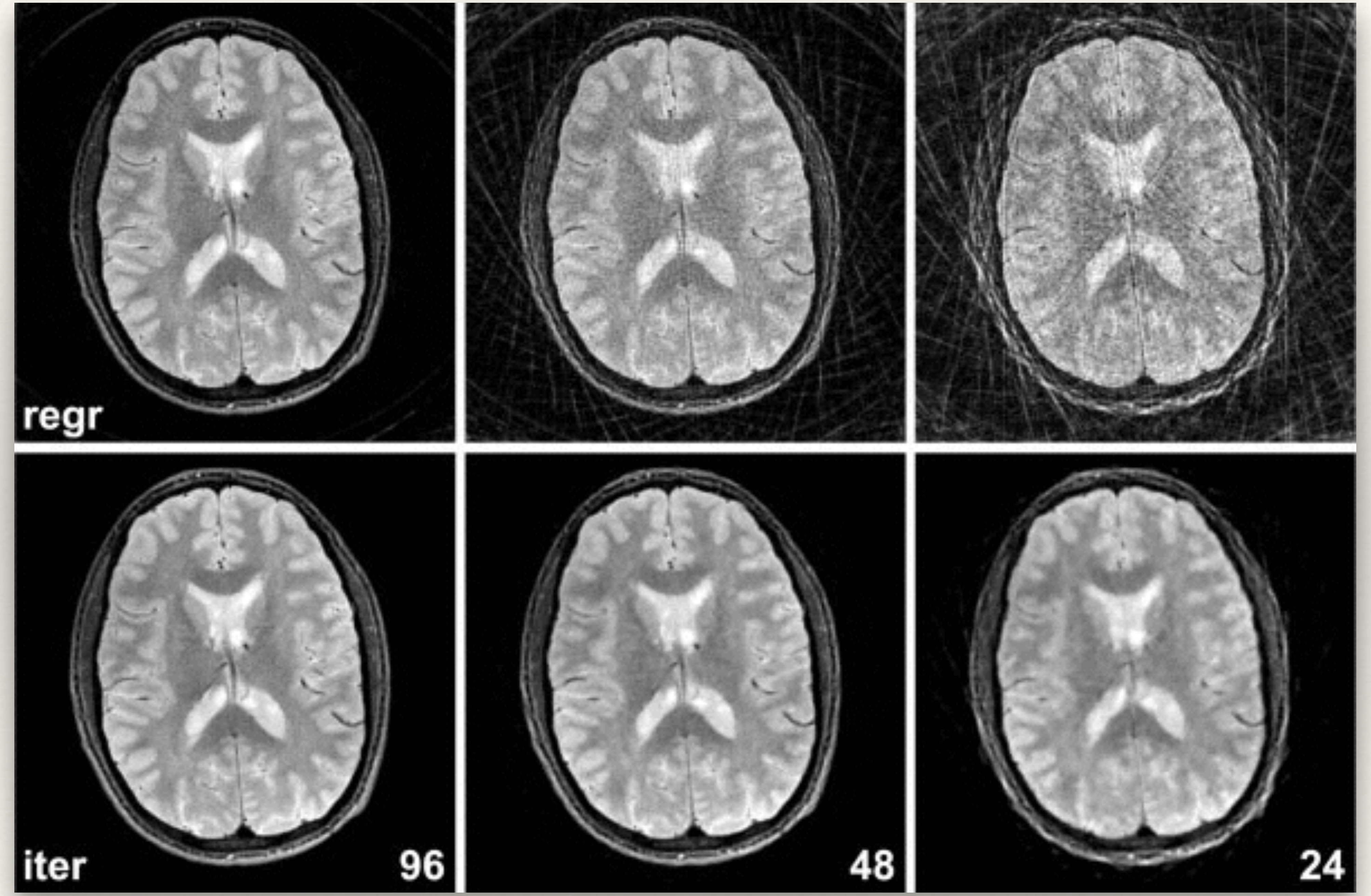
Inverse Model $\hat{\mathbf{x}} = h(\mathbf{y})$

Inverse Problems - Examples



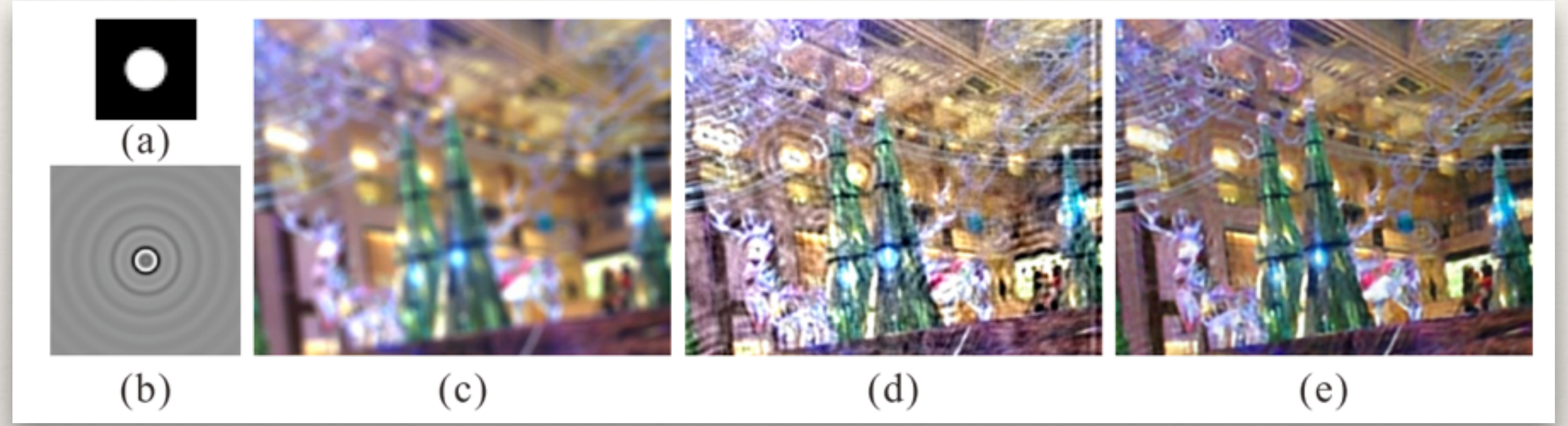
Up to 14.4 Gigapixels
With thousands of Channels

Inverse Problems - Examples



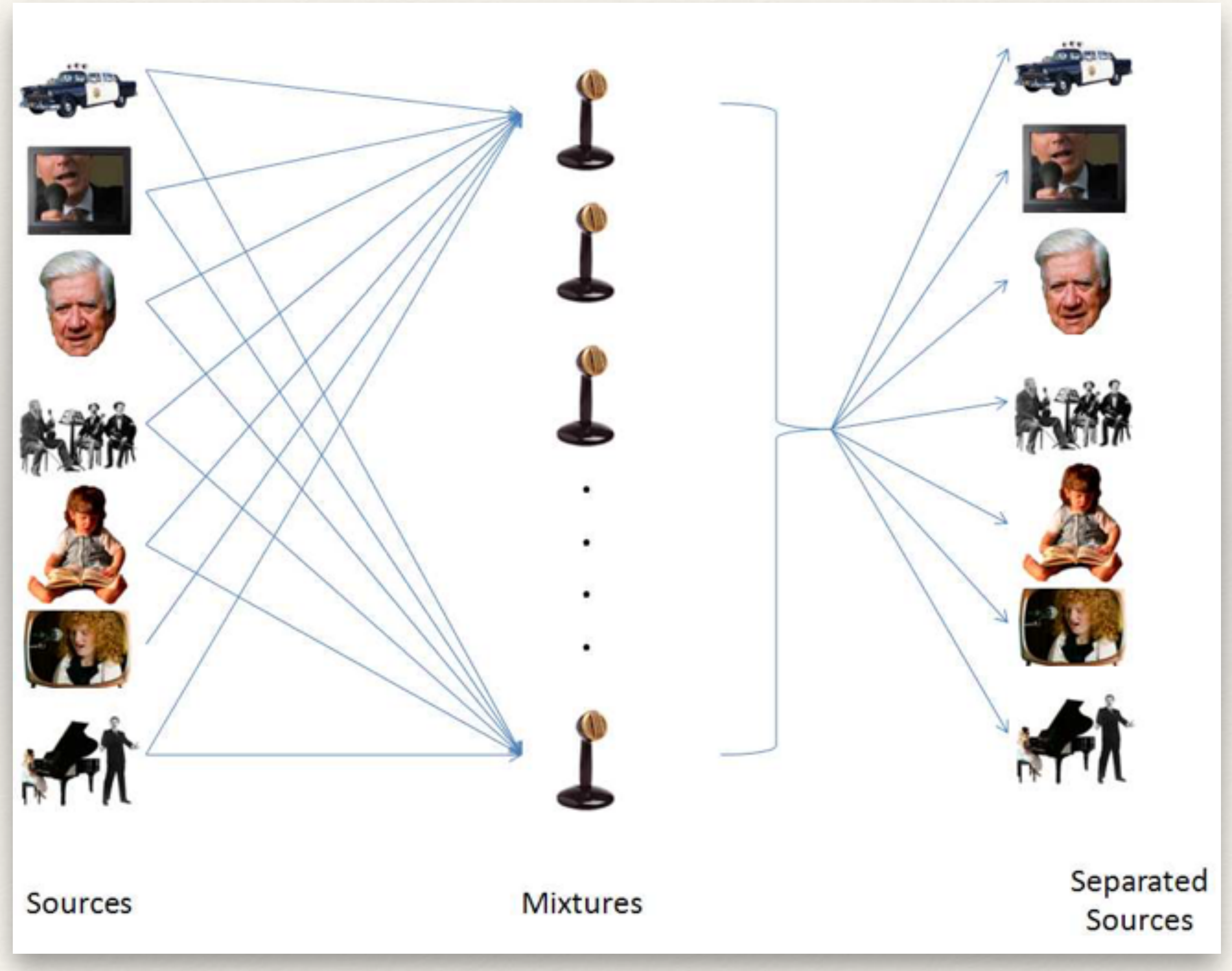
[Block et.al, 2007]

Inverse Problems - Examples

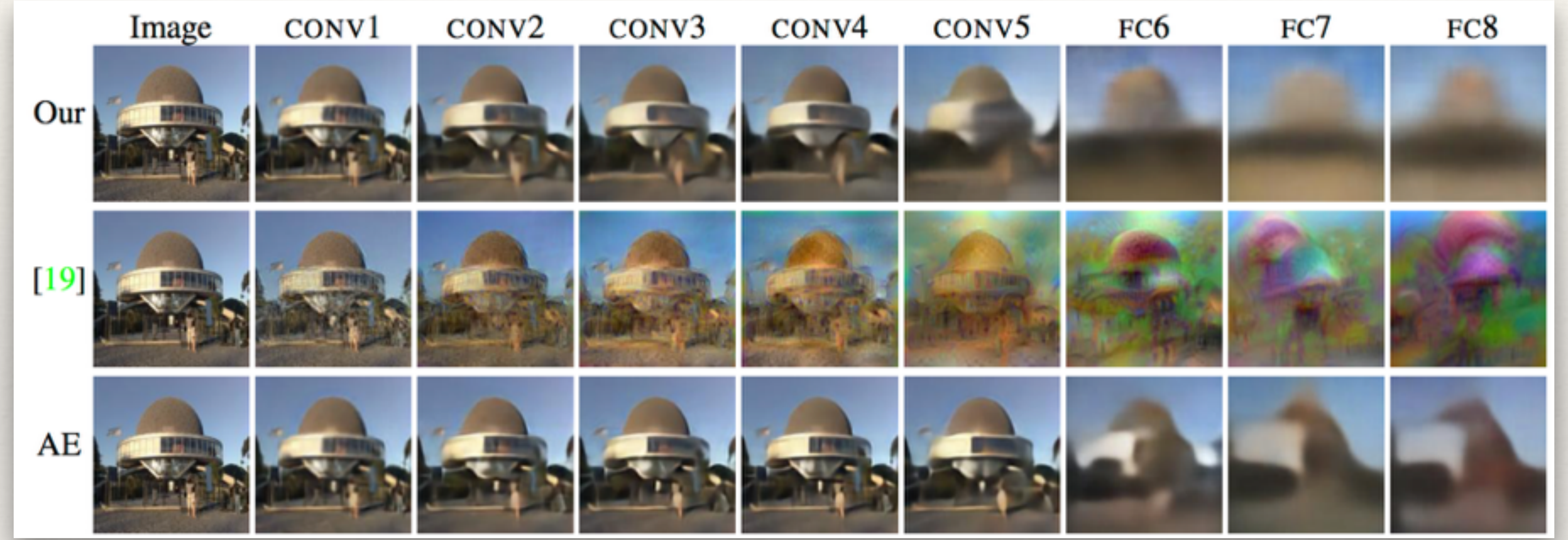


[Xu et al., 2014]

Inverse Problems - Examples



Inverse Problems - Examples

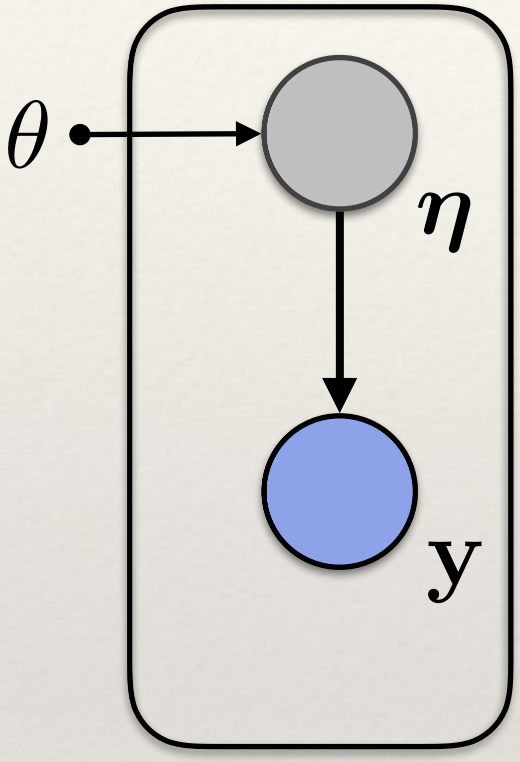


[Dosovitskiy & Brox, 2016]

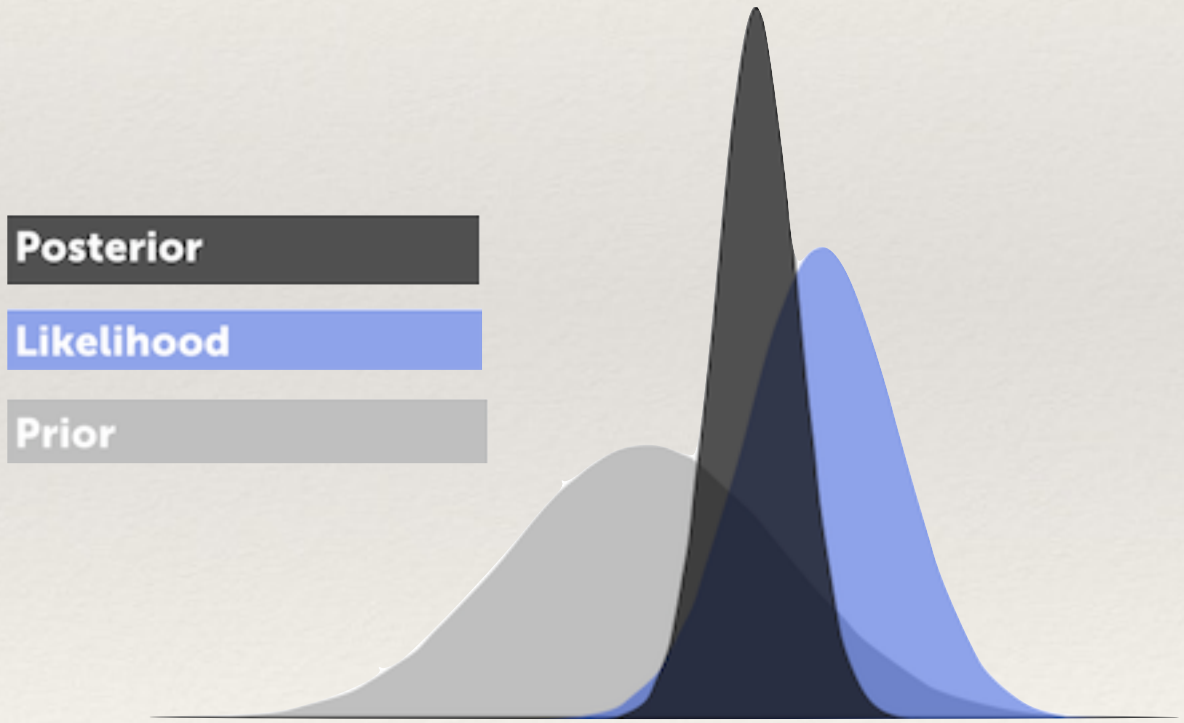
Inverse Problems - Examples

And many more…

Bayesian Inference



$$p_\theta(\boldsymbol{\eta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\eta})p_\theta(\boldsymbol{\eta})}{p(\mathbf{y})}$$



Iterative Bayesian Inference

$$p_\theta(\boldsymbol{\eta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\eta})p_\theta(\boldsymbol{\eta})}{p(\mathbf{y})}$$

Choose / learn a prior $p_\theta(\boldsymbol{\eta})$

For likelihood $p(\mathbf{y}|\boldsymbol{\eta})$

Choose inference method $\boldsymbol{\Gamma}$

Iterate

Iterative Bayesian Inference

$$p_\theta(\boldsymbol{\eta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\eta})p_\theta(\boldsymbol{\eta})}{p(\mathbf{y})}$$

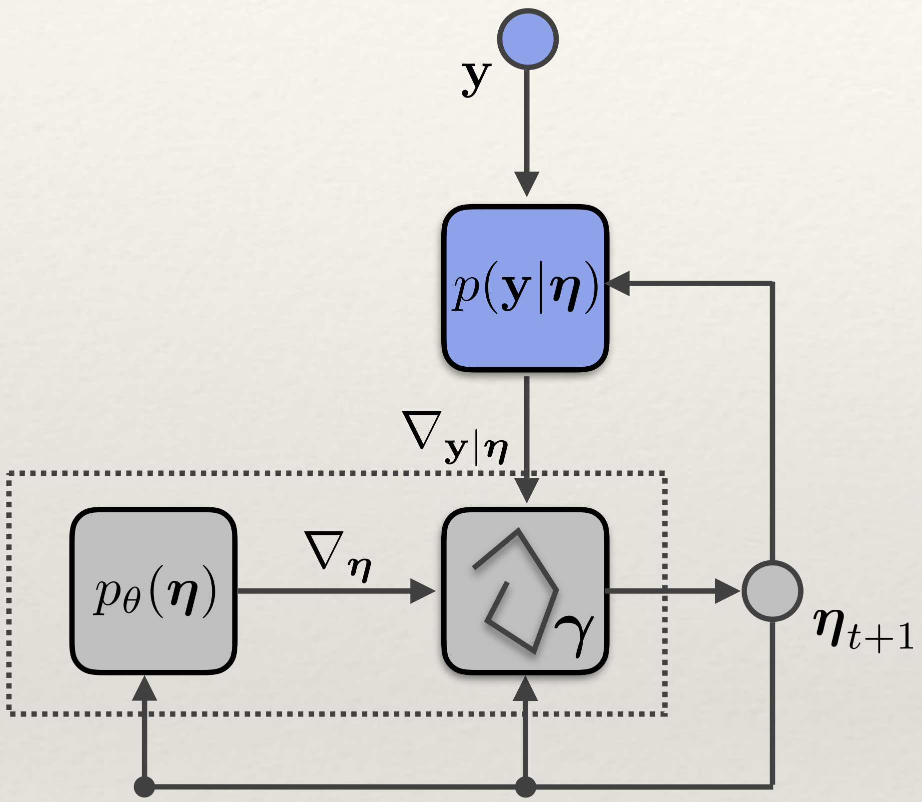
Choose/learn a prior $p_\theta(\boldsymbol{\eta})$

Choose inference method $\boldsymbol{\Gamma}$

For likelihood $p(\mathbf{y}|\boldsymbol{\eta})$

Iterate

Iterative Inference



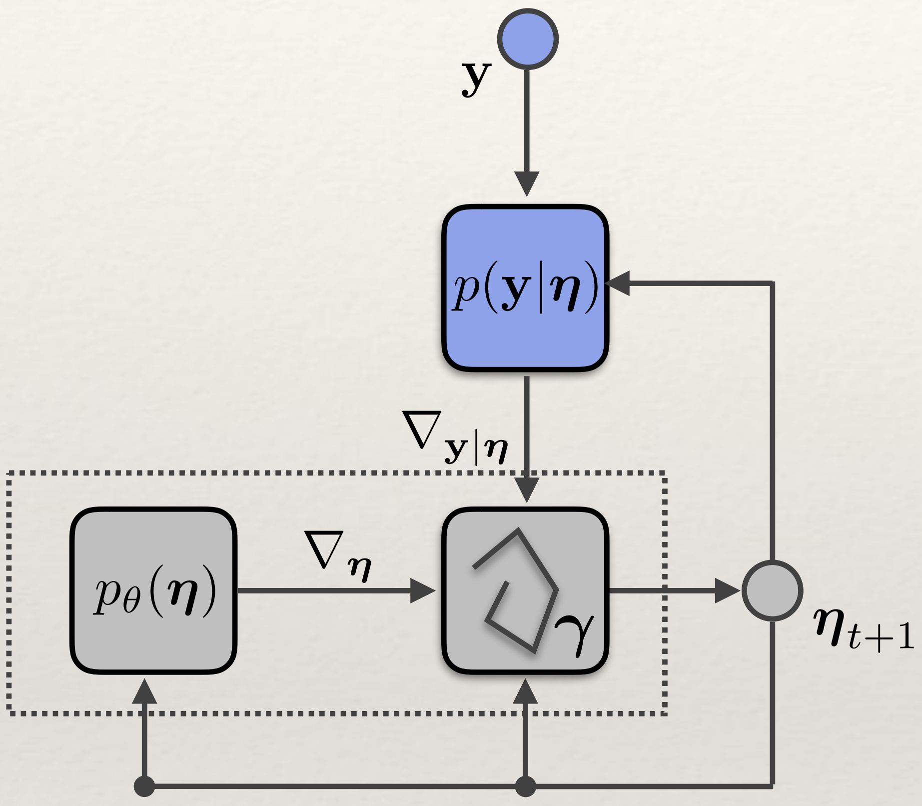
Maximum a posteriori (MAP) inference

$$\hat{\boldsymbol{\eta}} = \arg\max_{\boldsymbol{\eta}} p(\mathbf{y}|\boldsymbol{\eta}) p_\theta(\boldsymbol{\eta})$$

Gradient ascent

$$\begin{aligned}\boldsymbol{\eta}_{t+1} &= \boldsymbol{\eta}_t + \gamma_t \nabla \log p(\boldsymbol{\eta}|\mathbf{y}) \\ &= \boldsymbol{\eta}_t + \gamma_t (\nabla \log p(\mathbf{y}|\boldsymbol{\eta}) + \nabla \log p(\boldsymbol{\eta})) \\ &= \boldsymbol{\eta}_t + \gamma_t (\nabla_{\mathbf{y}|\boldsymbol{\eta}} + \nabla_{\boldsymbol{\eta}})\end{aligned}$$

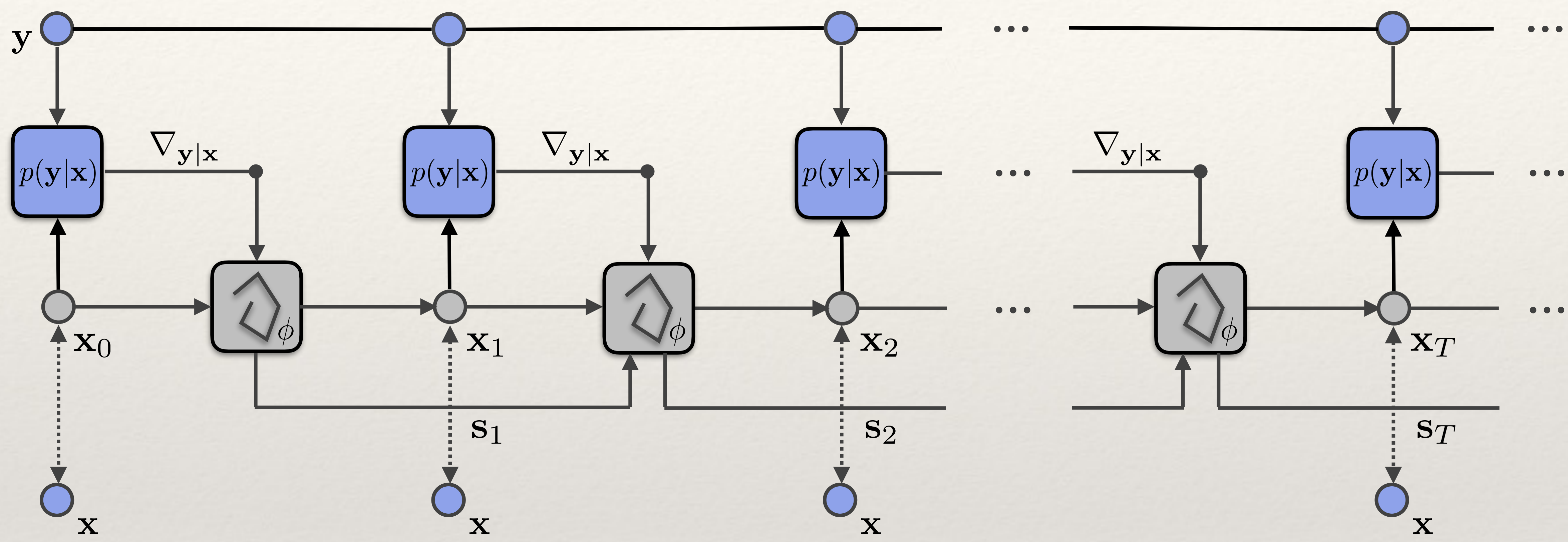
Recurrent Inference Machine



$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \gamma_t(\nabla_{\mathbf{y}|\boldsymbol{\eta}} + \nabla_{\boldsymbol{\eta}})$$

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + h_\phi(\nabla_{\mathbf{y}|\boldsymbol{\eta}}, \boldsymbol{\eta}_t, \mathbf{s}_t)$$

Recurrent Inference Machines in Time



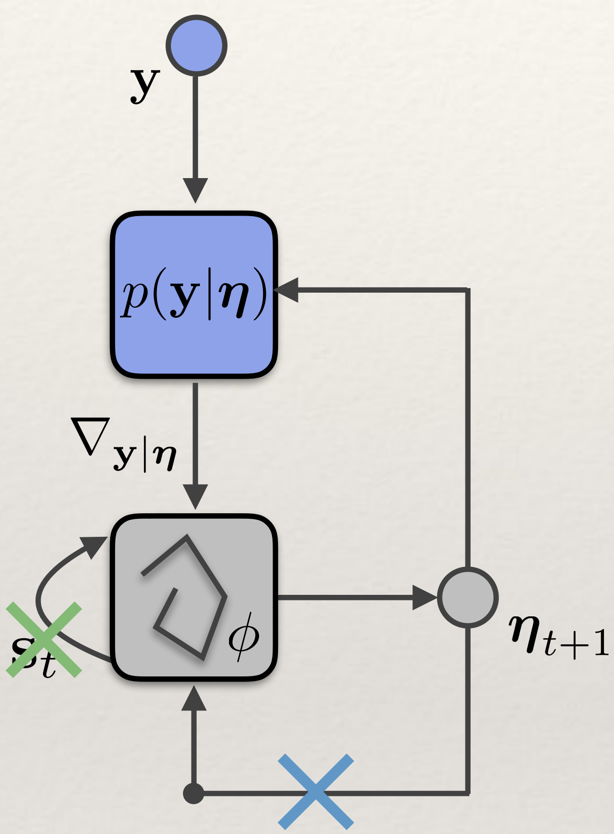
Objective $$g(\phi) = \frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}(\mathbf{x}^{(i)} - \hat{\mathbf{x}}_t^{(i)})$$

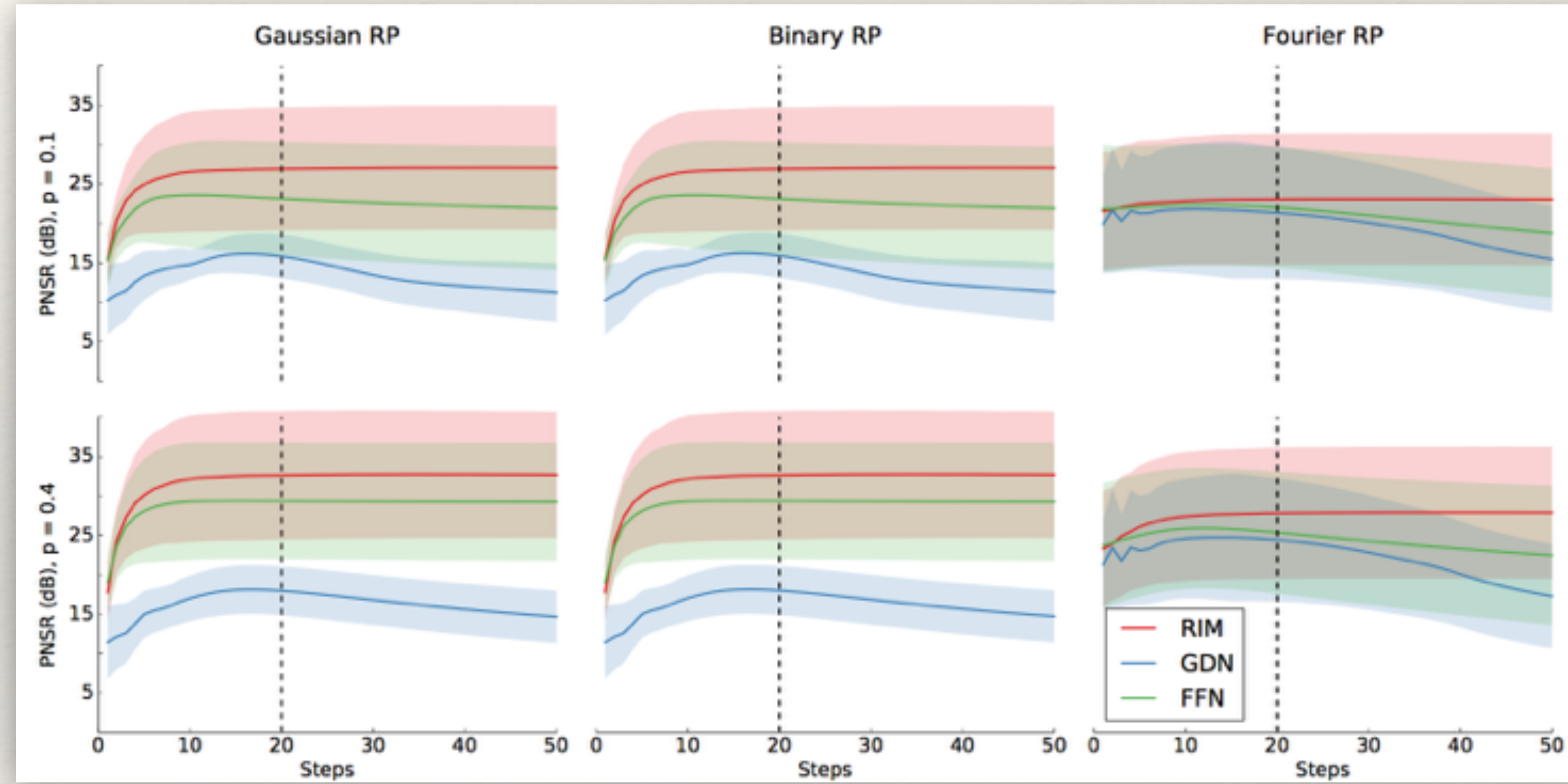
Simple Super-Resolution

Natural Images

200 training images, 481 x 321 pixel each, ~30 Megapixel

Reconstruction from Random Projections

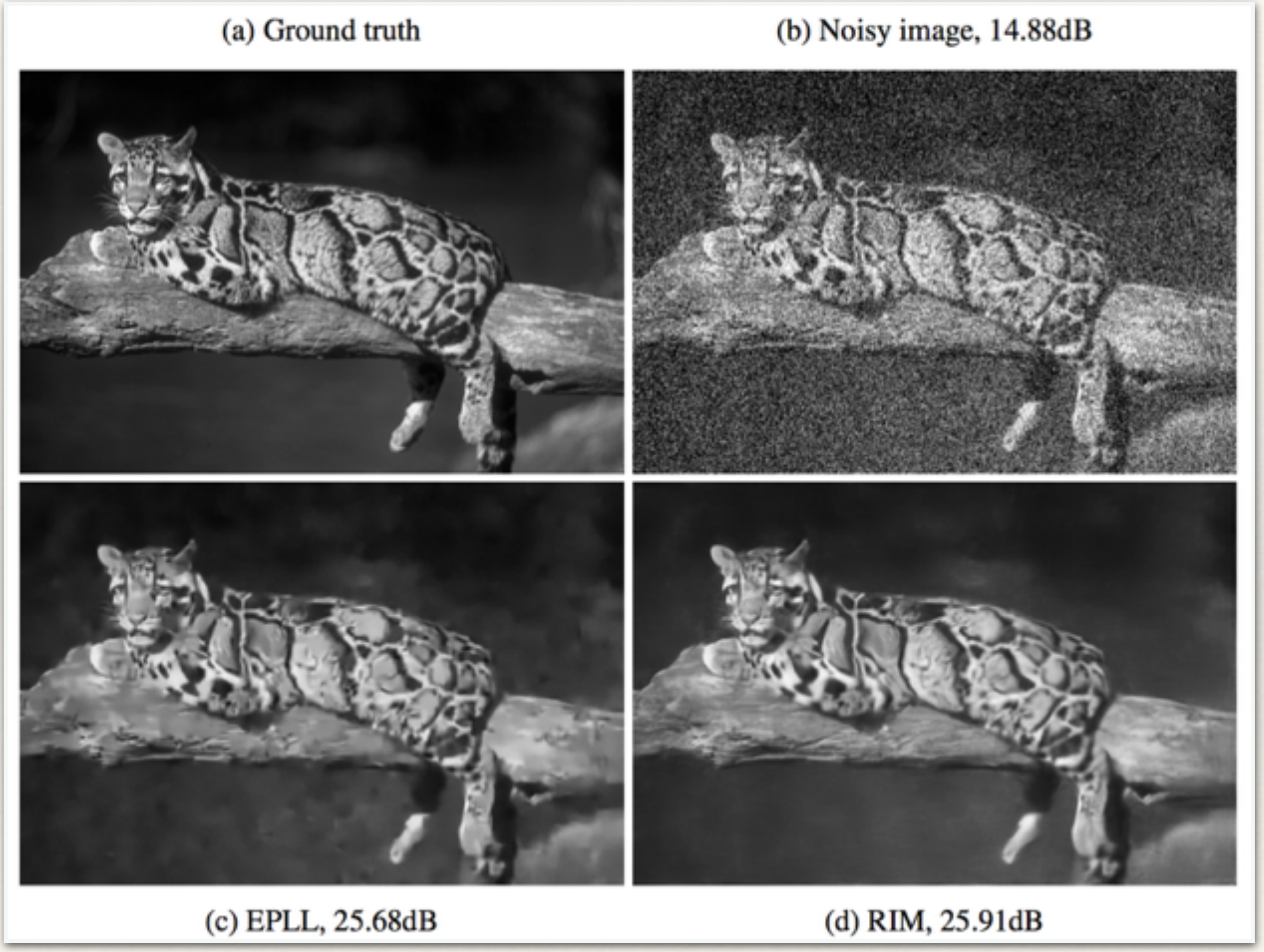




32 x 32 pixel image patches

Fast Convergence on all tasks

Image Denoising



Denoising trained on small image patches, generalises to full-sized images

Image Denoising

Grayscale

	Not Quantized		
σ	15	25	50
KSVD	30.87	28.28	25.17
5x5 FoE	30.99	28.40	25.35
BM3D	31.08	28.56(28.35)	25.62(25.45)
LSSC	31.27	28.70	25.72
EPLL	31.19	28.68(28.47)	25.67(25.50)
opt-MRF	31.18	28.66	25.70
MLP		28.85(28.75)	(25.83)
RTF-5		28.75	
RIM-3task	31.19(30.98)	28.67(28.45)	25.78(25.59)
RIM-denoise	**31.31**(31.10)	**28.91**(28.72)	**26.06**(25.88)

RGB

Method	PSNR
CBM3D	30.18
RTF-5	30.57
RIM (ours)	**30.84**(30.67)

Super-resolution

LR

HR

Bicubic Interpolation

RIM

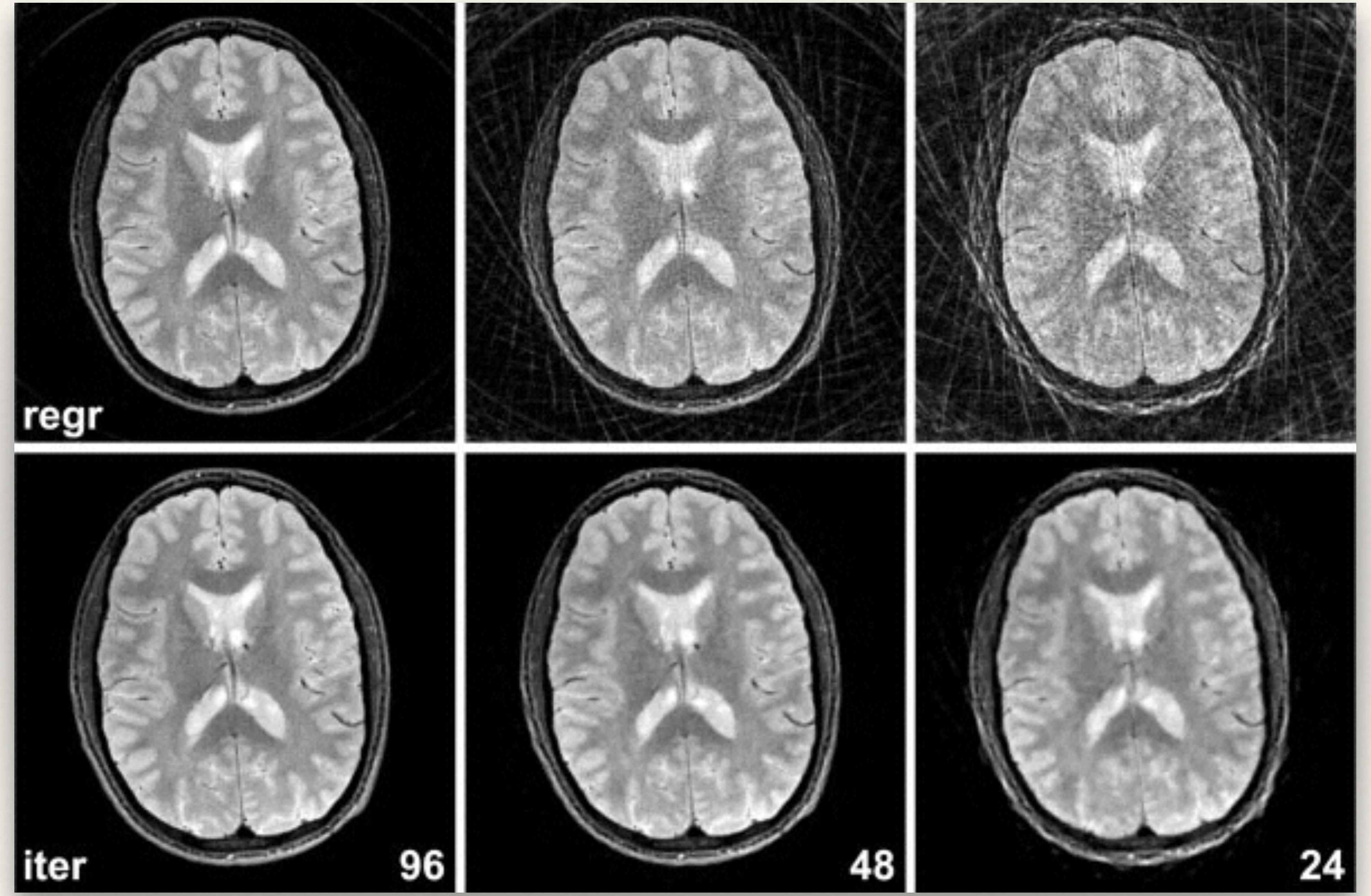
Super-resolution

(a) Original Image (b) Bicubic: 30.43/0.8326 (c) SRCNN: 31.34/0.8660

(d) A+: 31.43/0.8676 (e) SelfExSR: 31.18/0.8656 (f) RIM: **31.59/0.8712**

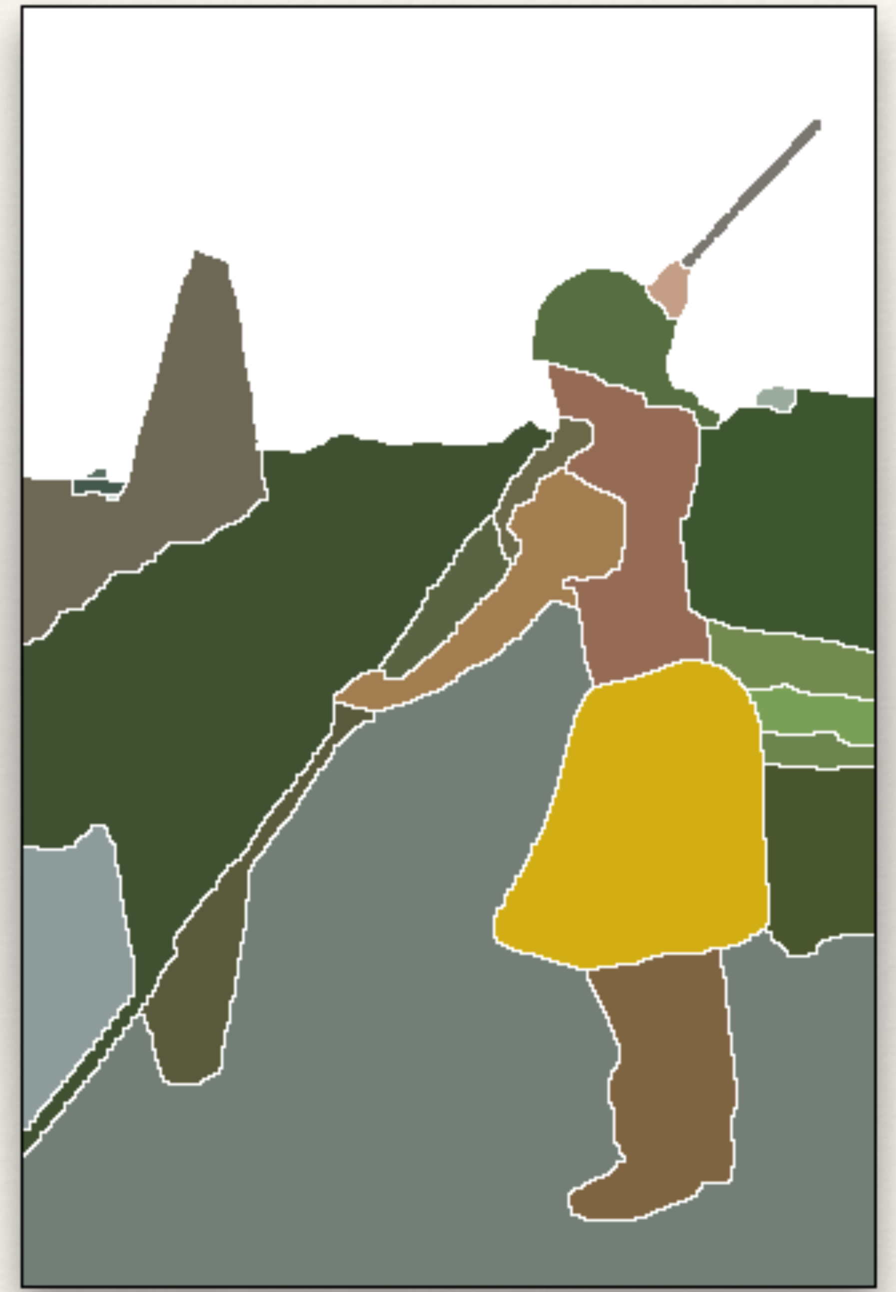
Metric	Scale	Bicubic	SRCNN	A+	SelfExSR	**RIM (Ours)**
PSNR	2x	29.55 ± 0.35	31.11 ± 0.39	31.22 ± 0.40	31.18 ± 0.39	$\mathbf{31.39} \pm 0.39$
	3x	27.20 ± 0.33	28.20 ± 0.36	28.30 ± 0.37	28.30 ± 0.37	$\mathbf{28.51} \pm 0.37$
	4x	25.96 ± 0.33	26.70 ± 0.34	26.82 ± 0.35	26.85 ± 0.36	$\mathbf{27.01} \pm 0.35$
SSIM	2x	0.8425 ± 0.0078	0.8835 ± 0.0062	0.8862 ± 0.0063	0.8855 ± 0.0064	$\mathbf{0.8885} \pm 0.0062$
	3x	0.7382 ± 0.0114	0.7794 ± 0.0102	0.7836 ± 0.0104	0.7843 ± 0.0104	$\mathbf{0.7888} \pm 0.0101$
	4x	0.6672 ± 0.0131	0.7018 ± 0.0125	0.7089 ± 0.0125	0.7108 ± 0.0124	$\mathbf{0.7156} \pm 0.0125$

Projects: MRI

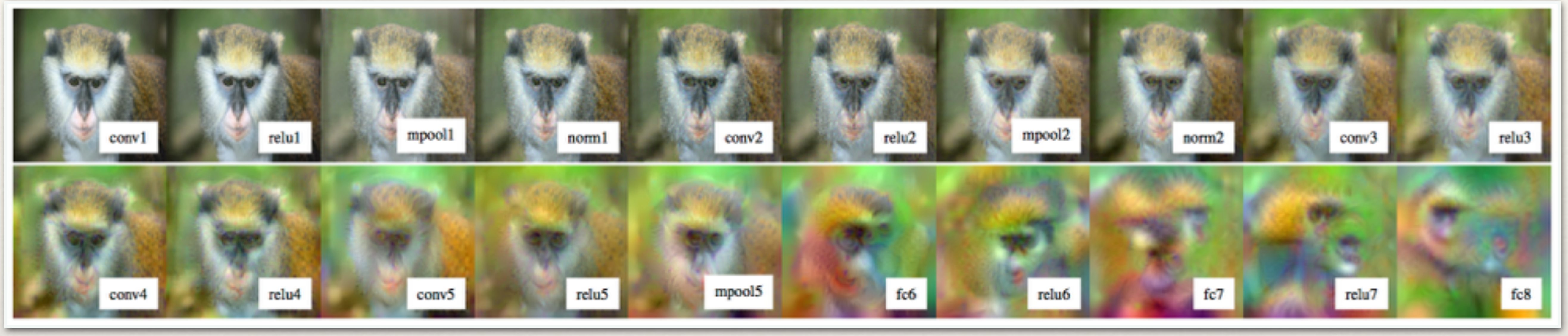


[Block et.al, 2007]

Projects: Content-Aware Image Restoration



Projects: Deep Visualisation



[Mahendran & Vedaldi, 2014]

[Yosinski et al., 2015]

Contact

Email: patrick.putzky@gmail.com
Room: C3.260
OpenReview: https://openreview.net/forum?id=HkSOlP9lg