Vision in the Deep Learning Era

Cees Snoek, UvA
Arnold W.M. Smeulders, UvA
Efstratios Gavves, UvA
Laurens van de Maaten, Facebook

Advanced School for Computing and Imaging

UNIVERSITY OF AMSTERDAM

Overview Day 2

Vision in the deep learning era

1. Convolutional Neural Networks
2. Case studies
3. Recurrent Networks

Action recognition by learning

4. Video representations
5. Spatiotemporal localization
6. VideoLSTM

Who am I?

Assistant Professor with the QUVA Lab

- ❑ Also, teaching Deep Learning
- ❑ Slides, code available at uvadlc.github.io

So far: Computer Vision & Deep Learning

Learn more on what is the objective* process that makes understanding and reasoning given input possible?

- ❑ * i.e. not necessarily in human terms

Recap from Day 1

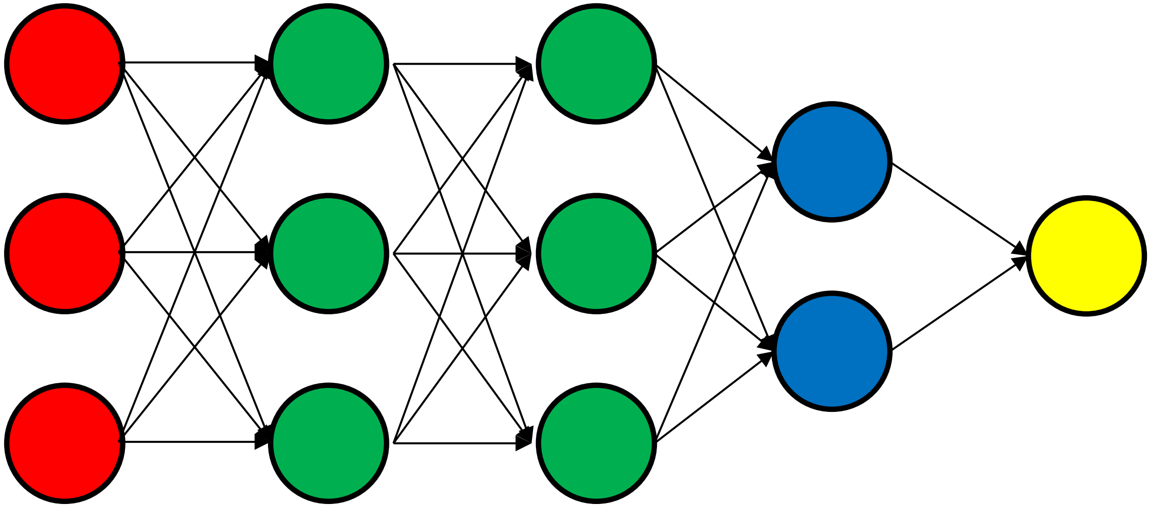
Visual Encoders ≡ Visual Structure

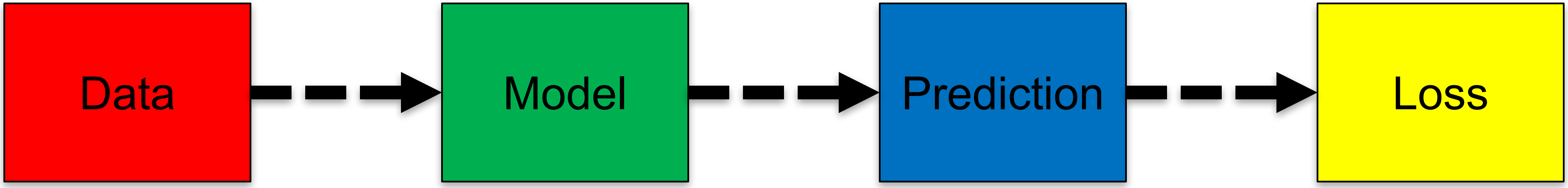
Computer Vision ≡ Learn Visual Structure

Learning Visual Structure:

- ❑ Feature learning
  - ❑ Clustering (e.g., k-means)
  - ❑ Sparse Coding
  - ❑ Autoencoders
- ❑ Feature pooling (Bag of Words, Fisher, VLAD)

A Neural Network perspective





Data

Model

Prediction

Loss

5 convolutional layers
2 fully connected layers
----
100 convolutional layers
50 batch normalization layers

Softmax

Sigmoid

Linear etc.

Cross entropy

Euclidean

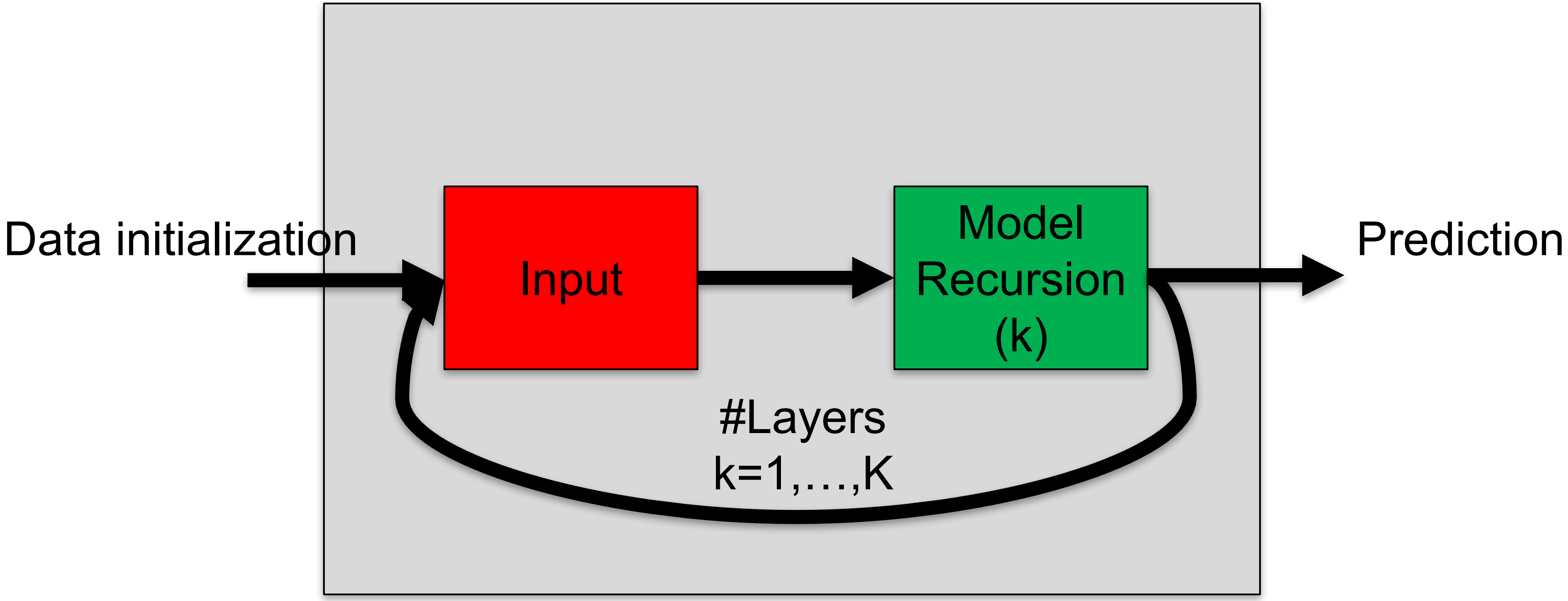
Contrastive

Or simpler …

In neural networks no real separation between layers

Output layer is just another layer

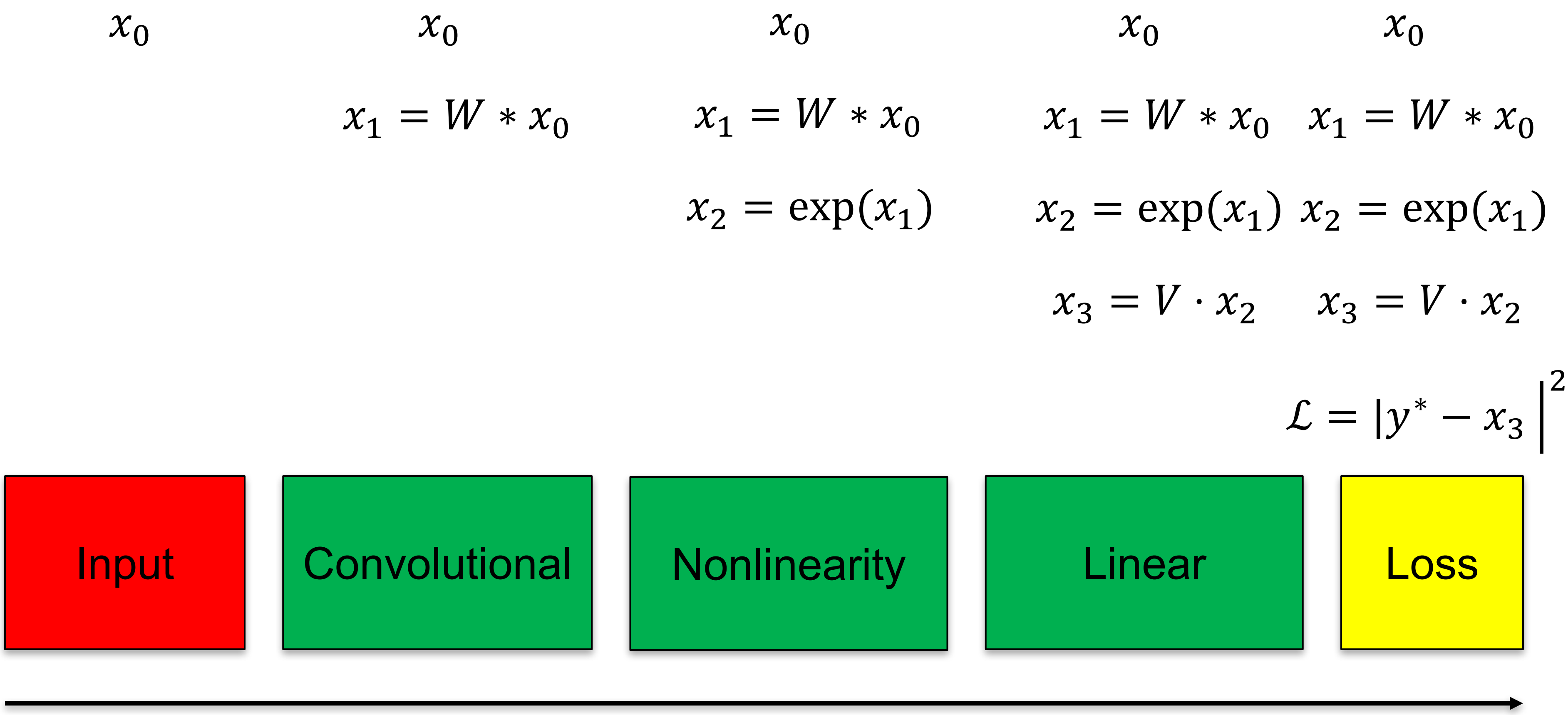
Loss layer is just another layer with one more input (targets)



$f(x) = V \cdot \exp(W * x)$ as a NN

A neural network is organized by layers

- ❑ 1 layer → 1 module → A single and simple operation



x_0

Input

x_0

$x_1 = W * x_0$

Convolutional

x_0

$x_1 = W * x_0$

$x_2 = \exp(x_1)$

Nonlinearity

x_0

$x_1 = W * x_0$

$x_2 = \exp(x_1)$

$x_3 = V \cdot x_2$

Linear

x_0

$x_1 = W * x_0$

$x_2 = \exp(x_1)$

$x_3 = V \cdot x_2$

$\mathcal{L} = |y^* - x_3|^2$

Loss

Module/Layer types?

Linear: $x_{i+1} = W \cdot x_i$ [parameteric → learnable]

Convolutional: $x_{i+1} = W * x_i$ [parameteric → learnable]

Nonlinearity: $x_{i+1} = h(x_0)$ [non-parameteric → defined]

Pooling: $x_1 = \text{downsample}\ (x_0)$ [non-parameteric → defined]

Normalization, *e.g.*: $x_1 = \ell_2(x_0)$ [non-parameteric → defined]

Regularization, *e.g.*: $x_1 = \text{dropout}(x_0)$ [non-parameteric → defined]

Practically, any 1st order [almost everywhere] differentiable function can be a module

Training Neural Networks

1. The Neural Network

$$\boxed{a_L(x;\theta_{1,\ldots,\mathrm{L}}) = h_L\,(h_{L-1}(\ldots h_1(x,\theta_1),\theta_{L-1}),\theta_L)}$$

2. Learning by minimizing empirical error

$$\boxed{\theta^* \leftarrow \arg\min_\theta \sum_{(x,y)\subseteq(X,Y)} \mathcal{L}(y, a_L(x;\theta_{1,\ldots,\mathrm{L}}))}$$

3. Optimizing with Gradient Descend based methods

$$\boxed{\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla_\theta \mathcal{L}}$$

Convolutional Neural Networks

Or just Convnets/CNNs

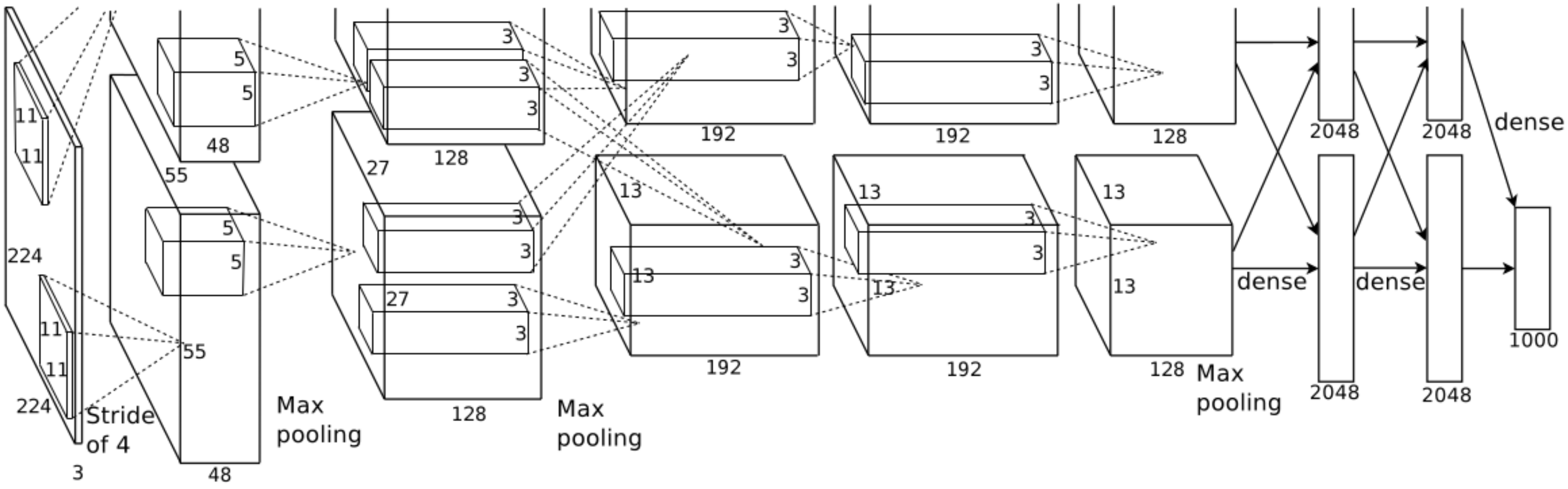


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Convnets vs NNs

Question: Spatial structure?

- ❑ NNs: don’t care
- ❑ Convnets: Convolutional filters

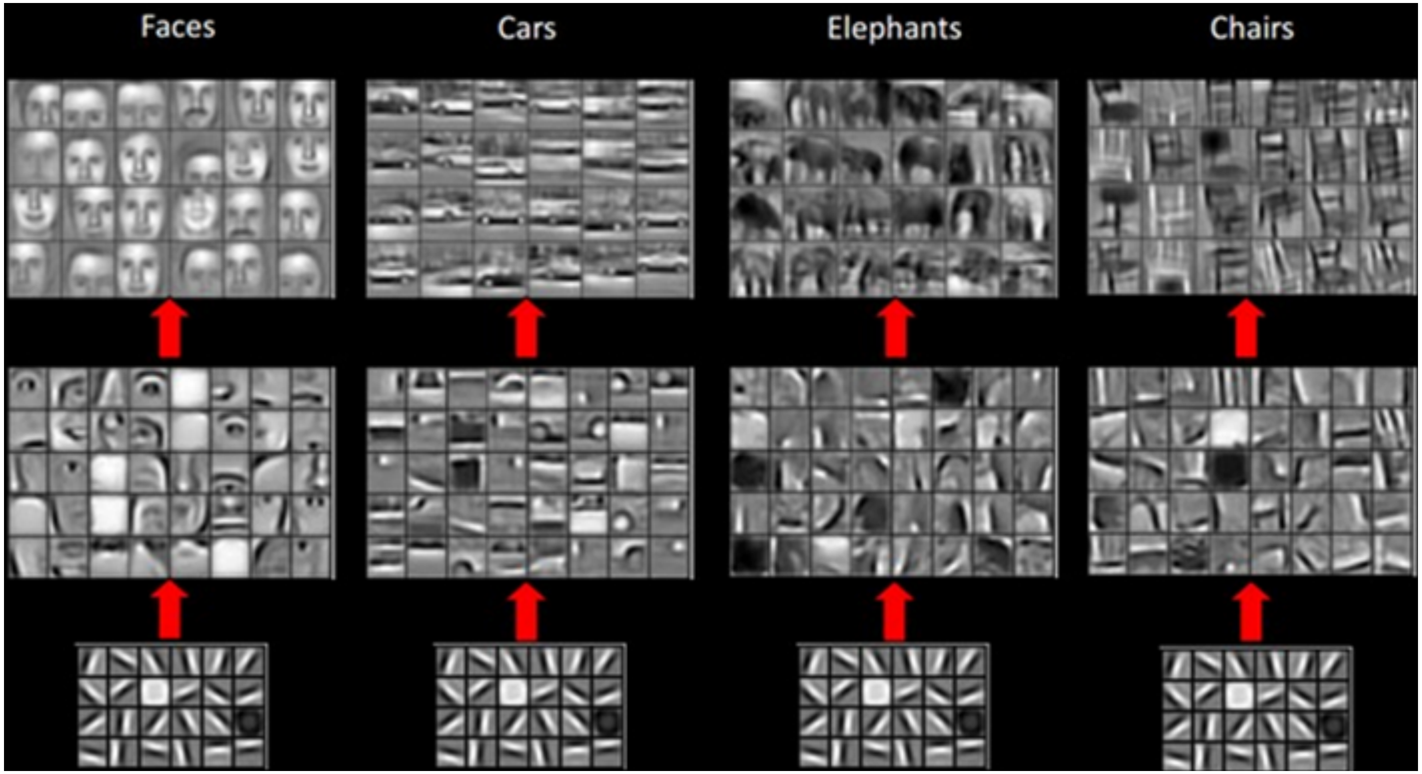
Question: Huge input dimensionalities?

- ❑ NNs: don’t care (in theory)
- ❑ Convnets: Parameter sharing

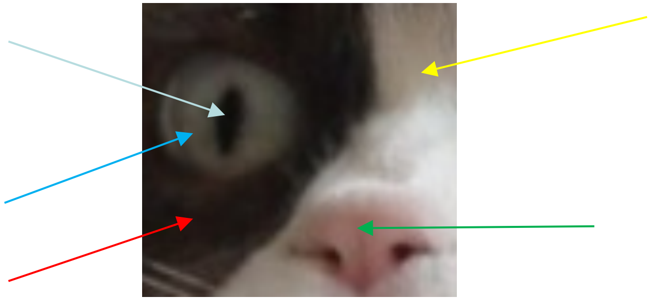
Question: Local variances?

- ❑ NNs: don’t care (much)
- ❑ Convnets: Pooling

Preserving spatial structure



Quiz: What does spatial mean?



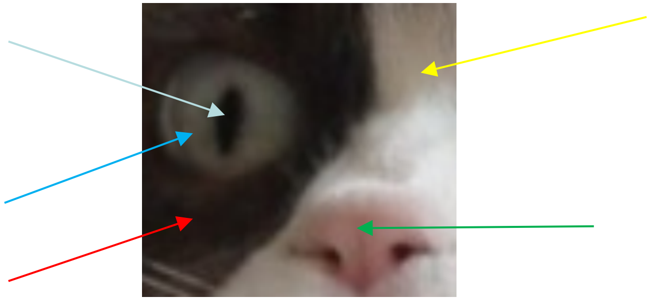
What does spatial mean?

One pixel alone does not carry much information

Many pixels in the right order though → tons of information

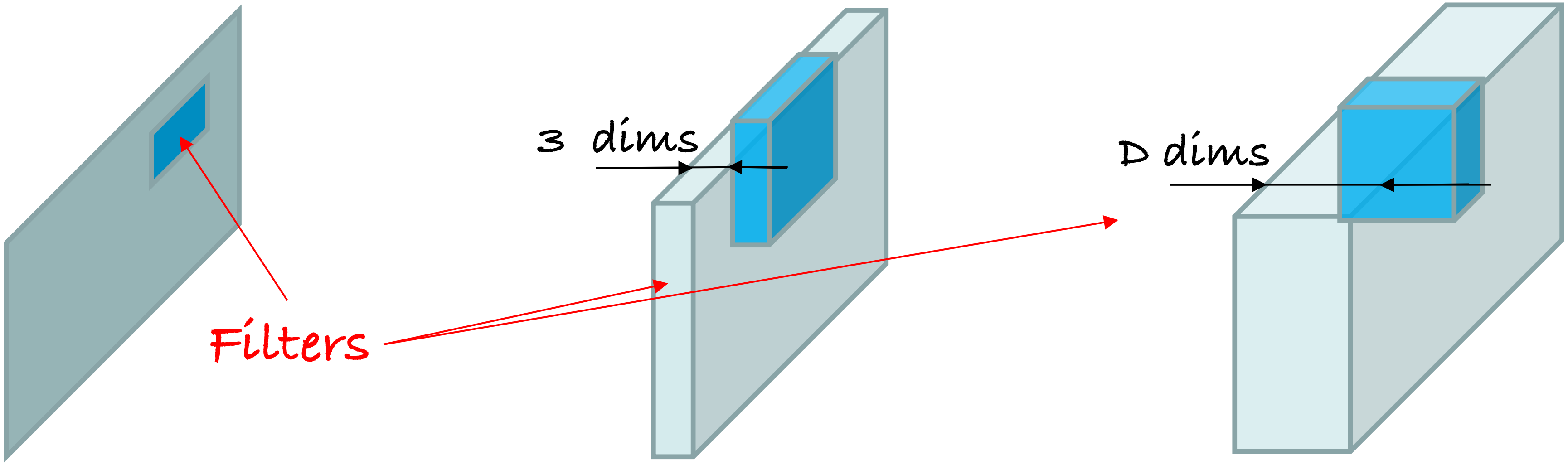
I.e., Neighboring variables are correlated

And the variable correlations is the visual structure we want to learn



Filters have width/height/depth

Grayscale RGB Multiple channels



How many parameters **per** filter?

$$\#params = H \times W \times D$$

Local connectivity

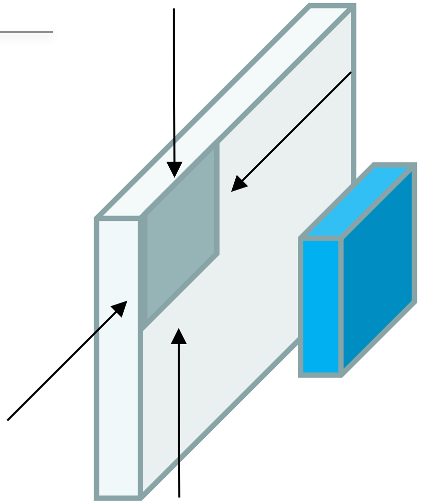
The weight connections are surface-wise local!

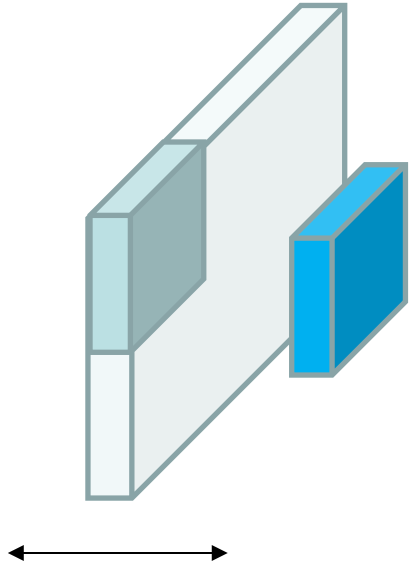
- ❑ Local connectivity

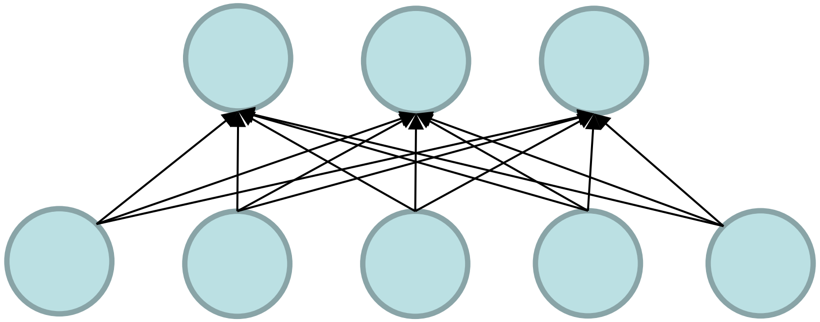
The weights connections are depth-wise global

For standard neurons no local connectivity

- ❑ Everything is connected to everything







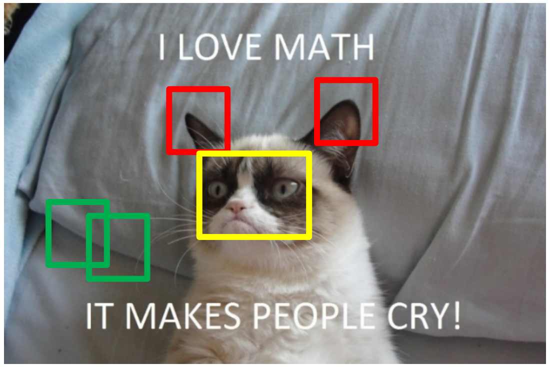
Parameter Sharing

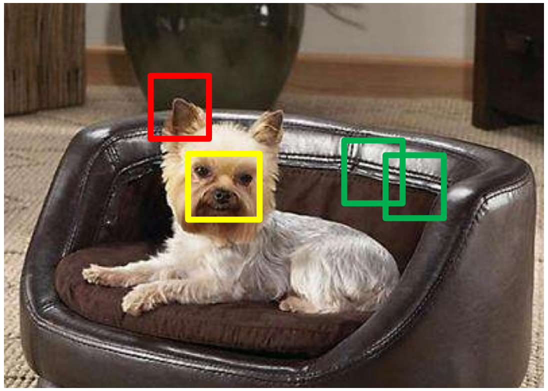
Natural images are stationary

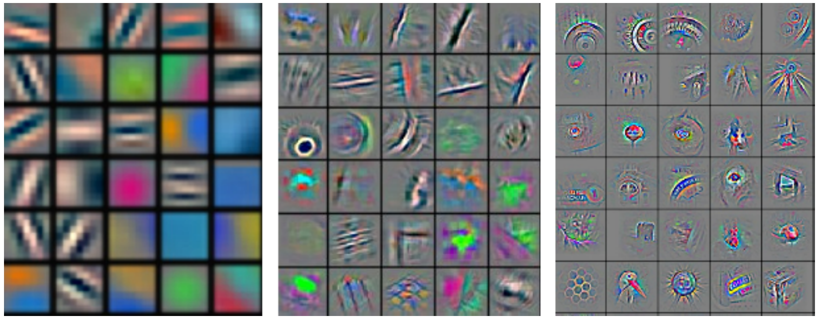
Visual features are common for different parts of one or multiple image

If features are **local** *and* **similar** across locations, why not **reuse** filters?

Localar pameter sharing → Convolutions







Convolutional filters

Original image

Convolutional filters

Original image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolutional filters

Original image

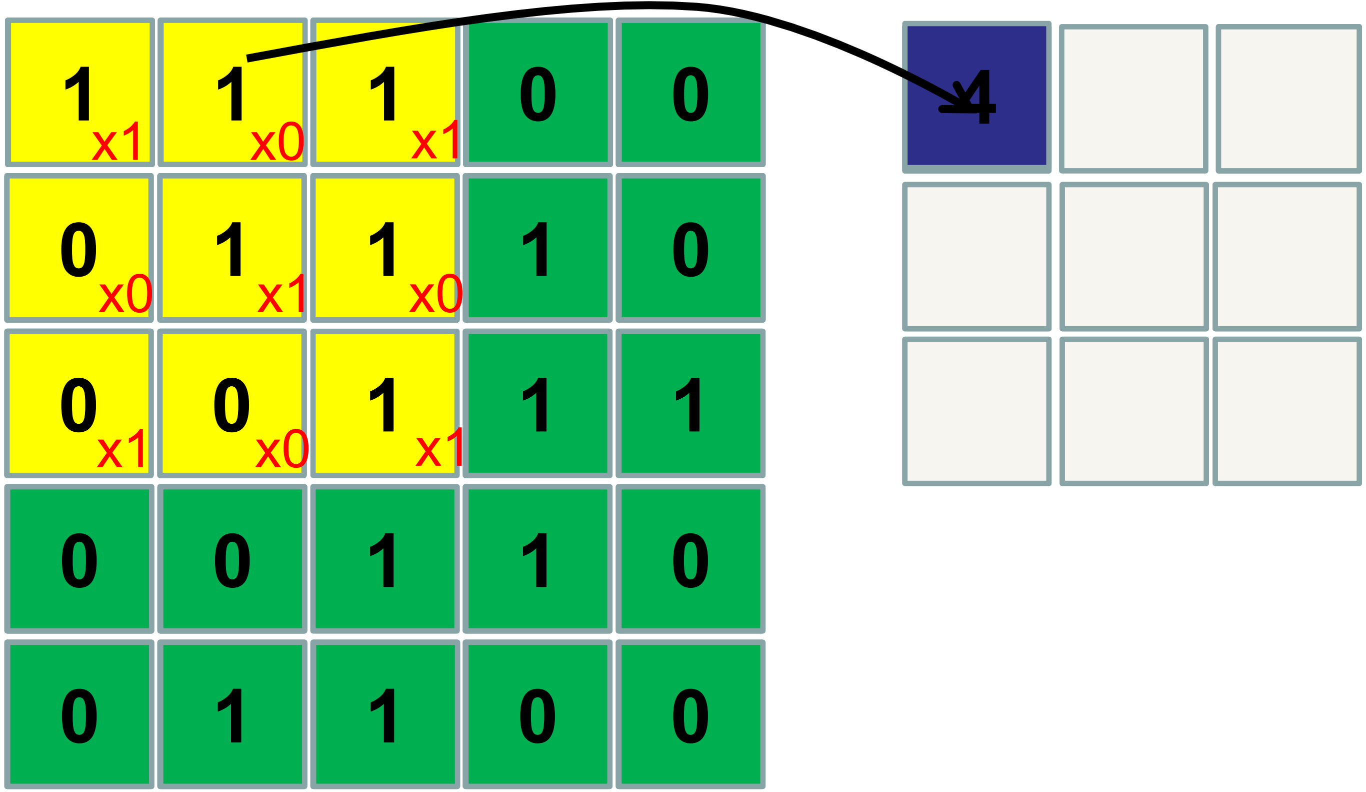
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

Result



1 x1	1 x0	1 x1	0	0
0 x0	1 x1	1 x0	1	0
0 x1	0 x0	1 x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \overbrace{I(x-i, y-j) \cdot h(i,j)}^{\text{Inner product}}$$

Convolutional filters

Original image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

1	1 x1	1 x0	0 x1	0
0	1 x0	1 x1	1 x0	0
0	0 x1	1 x0	1 x1	1
0	0	1	1	0
0	1	1	0	0

Result

4	3	

$$I(x,y)*h=\sum_{i=-a}^{a}\sum_{j=-b}^{b}\overbrace{I(x-i,y-j)\cdot h(i,j)}^{\text{Inner product}}$$

Convolutional filters

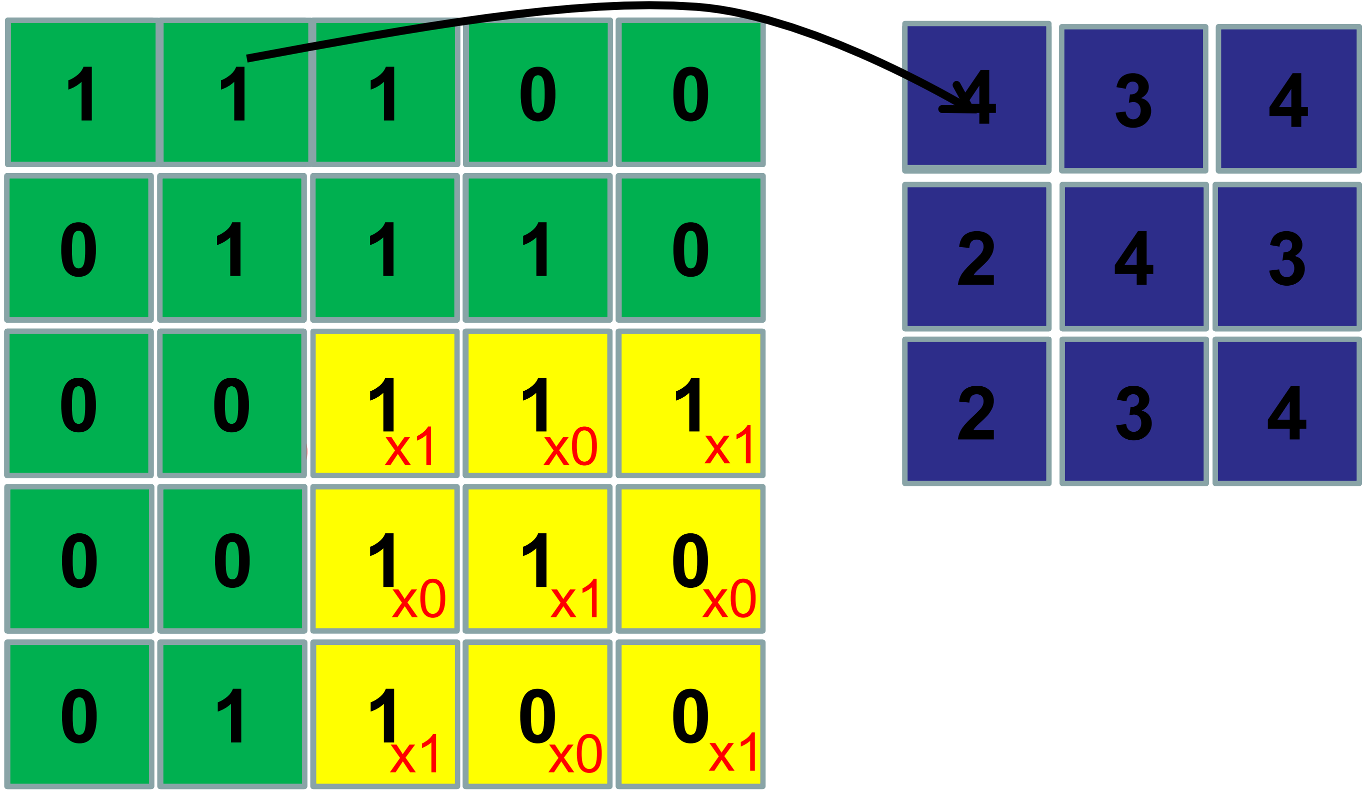
Original image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image



1	1	1	0	0
0	1	1	1	0
0	0	1_{x1}	1_{x0}	1_{x1}
0	0	1_{x0}	1_{x1}	0_{x0}
0	1	1_{x1}	0_{x0}	0_{x1}

Result

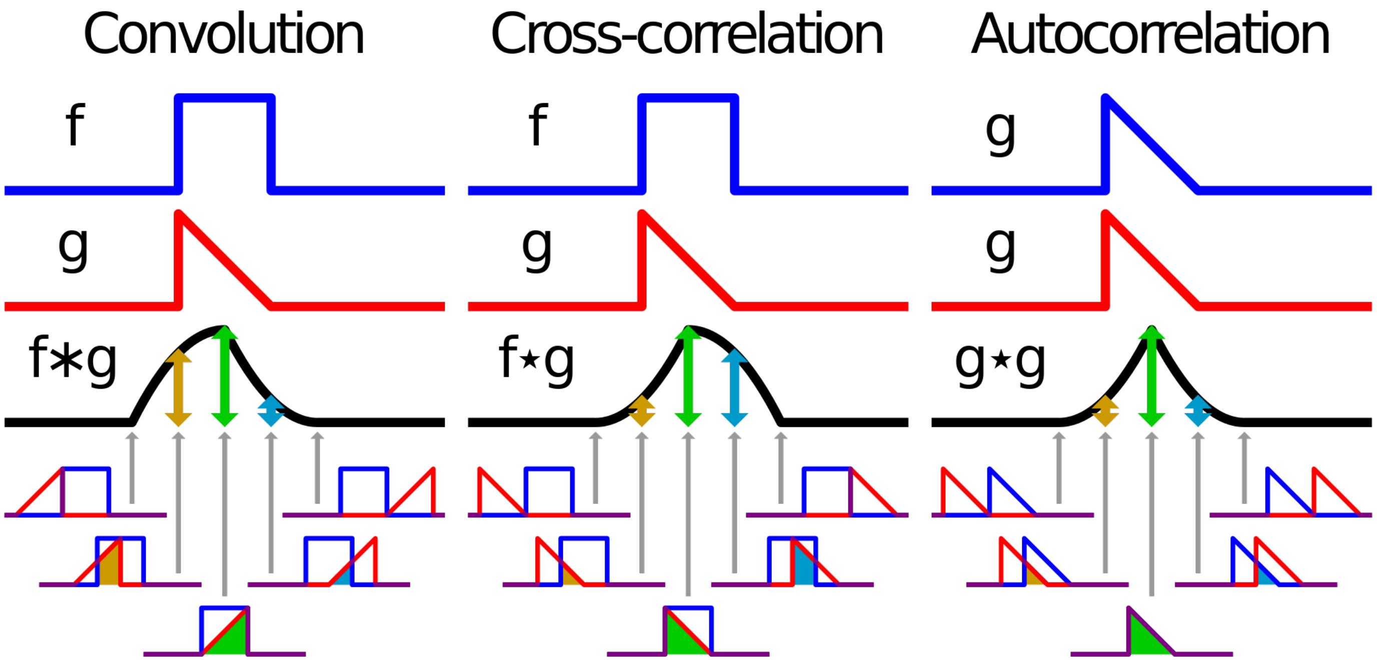
4	3	4
2	4	3
2	3	4

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \overbrace{I(x-i, y-j) \cdot h(i,j)}^{\text{Inner product}}$$

Why call them convolutions?

Definition *The convolution of two functions f and g is denoted by $*$ as the integral of the product of the two functions after one is reversed and shifted*

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau$$

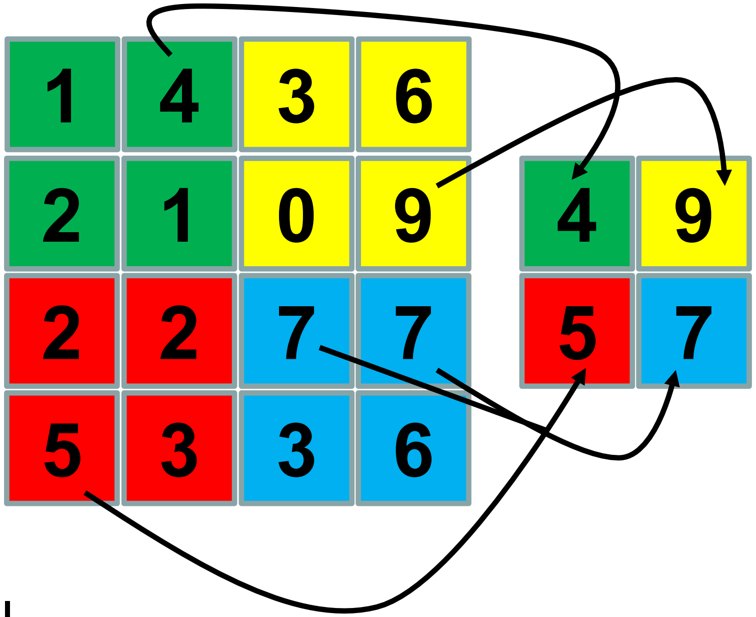


Pooling

Often we want to summarize the local information into a single code vector

Feature aggregation ≡ Pooling

- ❑ Pooled feature invariant to small local transformations. Only the strongest activation is retained
- ❑ Output dimensions → Faster computations
- ❑ Keeps most salient information
- ❑ Different dimensionality inputs can now be compared



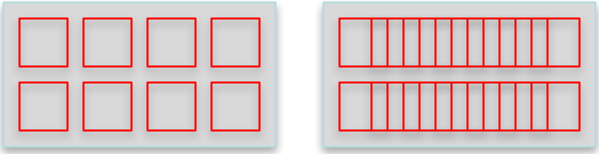
Popular pooling methods

- ❑ Average, Max, Bilinear, Convolutional

Implementation details

Stride

- □ every how many pixels do you compute a convolution
- □ equivalent to sampling coefficient, influences output size

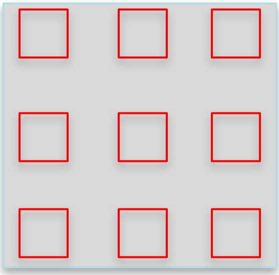


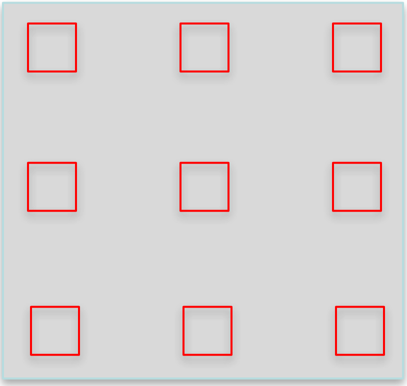
Padding

- □ Add 0s (or another value) around the layer input
- □ Prevent output from getting smaller and smaller

Dilation

- □ Atrous convolutions





Good practice

Resize the image to have a size in the power of 2

Use stride $s = 1$

A filter of $\left(h_f, w_f\right) = [3\times3]$ works quite alright with deep architectures

Add 1 layer of zero padding

Avoid combinations of hyper-parameters that do not click

- ❑ E.g. $s = 1$
- ❑ $[h_f \times w_f] = [3\times3]$ and
- ❑ image size $[h_{in} \times w_{in}] = [6\times6]$
- ❑ $[h_{out} \times w_{out}] = [2.5\times2.5]$
- ❑ Programmatically worse, and worse accuracy because borders are ignored

Nonlinearities

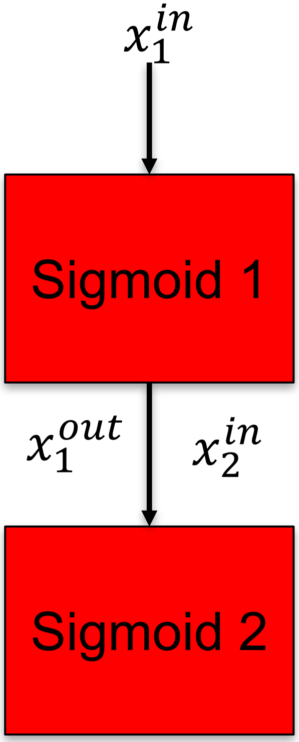
If we would only have N linear layers, we could replace them all with a single layer

$$W_1 \cdot W_2 \cdot \ldots \cdot W_N = W$$

Nonlinearities allow for deeper networks

Any nonlinear function can work although some are more preferable than others

Sigmoid



x_1^{in}

Sigmoid 1

x_1^{out} x_2^{in}

Sigmoid 2

Activation function $a = \sigma(x) = \frac{1}{1+e^{-x}}$

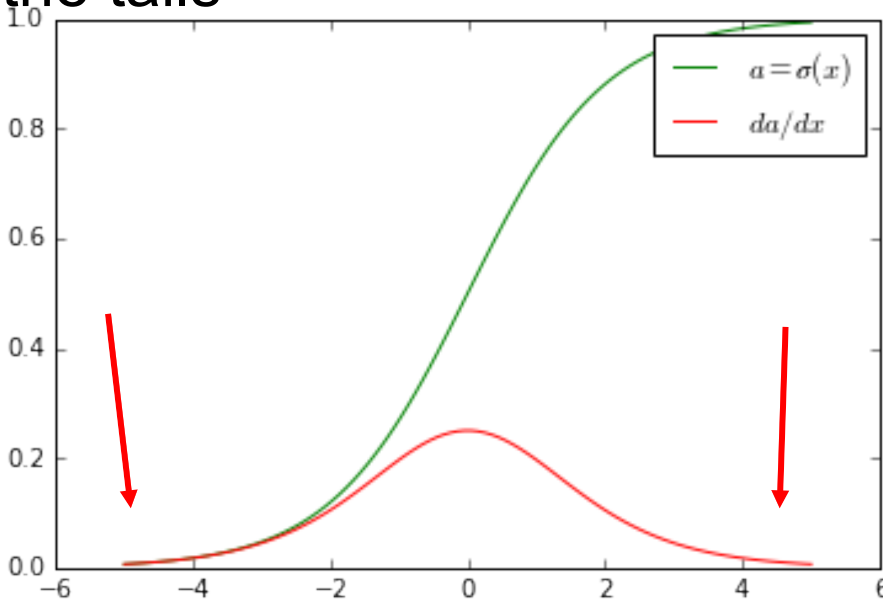
Gradient wrt the input $\frac{\partial a}{\partial x} = \sigma(x)(1 - \sigma(x))$

When $x_1^{in} \sim 0$ [normalized inputs] $x_2^{in} = x_1^{out} \sim 0.5$

- □ Introducing bias to hidden layer neurons
- □ Not recursive friendly

Gradients always < 1 and flat in the tails

- □ No serious upgrades
- □ Deep networks have problems



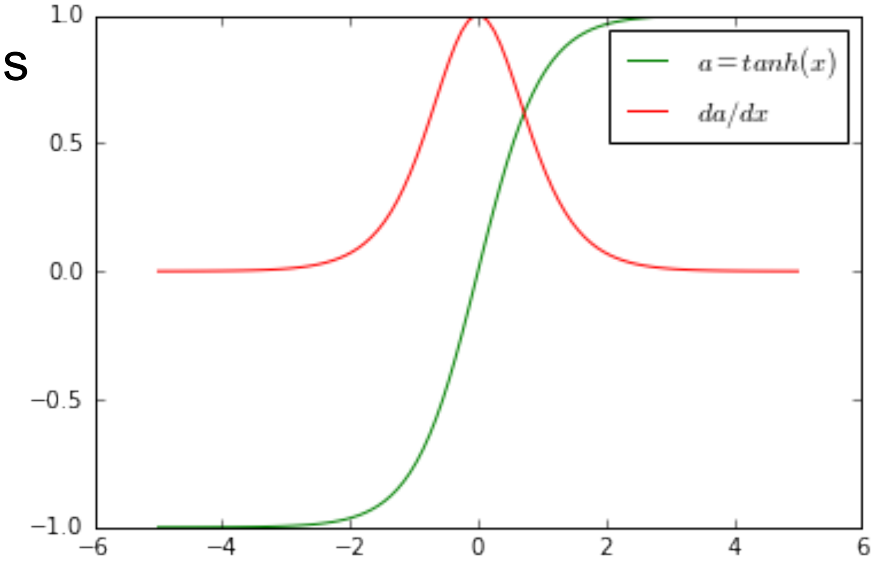
Tanh

Activation function $a = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Gradient with respect to the input $\frac{\partial a}{\partial x} = 1 - tanh^2(x)$

Similar to sigmoid, but with different output range

- ❑ $[-1, +1]$ instead of $[0, +1]$
- ❑ Stronger gradients, because data in subsequent module is centered around 0 (not 0.5)
- ❑ Less bias to hidden layer neurons
- ❑ Outputs positive or negative and likely to have zero mean

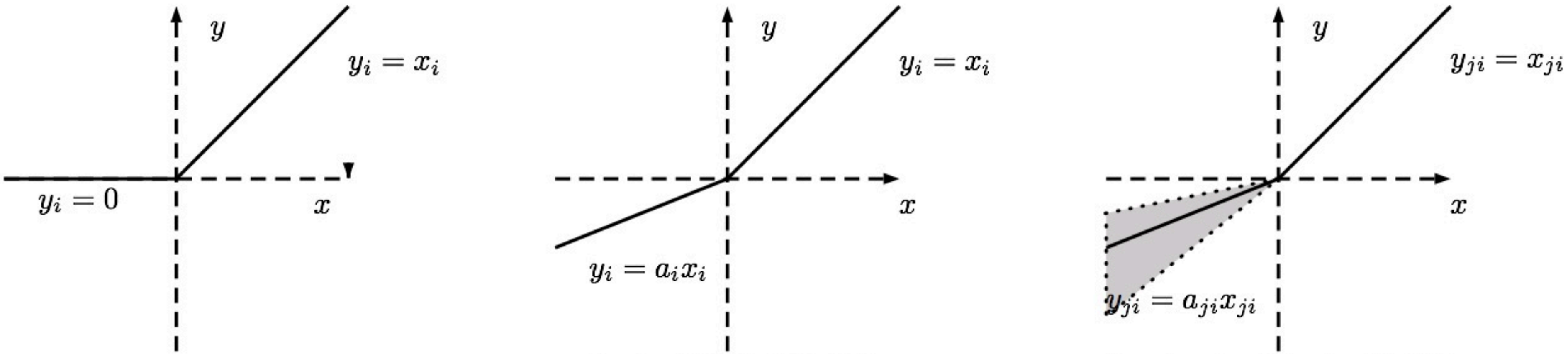


ReLU

Activation function $a = h(x) = \max(0, x)$

- ❑ Very popular in computer vision and speech recognition

Gradient wrt the input $\frac{\partial a}{\partial x} = \begin{cases} 0, if\ x \leq 0 \\ 1, ifx > 0 \end{cases}$



ReLU Leaky ReLU/PReLU Randomized Leaky ReLU

ReLU

Much faster computations, gradients

- No vanishing/exploding gradients
- People claim biological plausibility :/

Sparse activations

No saturation

Non-symmetric

Non-differentiable at 0

A large gradient during training can cause a neuron to "die". Higher learning rates mitigate the problem

Softmax

Activation function $a^{(k)} = softmax(x^{(k)}) = \frac{e^{x^{(k)}}}{\sum_j e^{x^{(j)}}}$

- ❑ Outputs probability distribution, $\sum_{k=1}^{K} a^{(k)} = 1$ for K classes
- ❑ Typically used as prediction layer

Because $e^{a+b} = e^a e^b$, we usually compute

$$a^{(k)} = \frac{e^{x^{(k)}-\mu}}{\sum_j e^{x^{(j)}-\mu}}, \mu = \max_k x^{(k)} \text{ because}$$

$$\frac{e^{x^{(k)}-\mu}}{\sum_j e^{x^{(j)}-\mu}} = \frac{e^{\mu} e^{x^{(k)}}}{e^{\mu} \sum_j e^{x^{(j)}}} = \frac{e^{x^{(k)}}}{\sum_j e^{x^{(j)}}}$$

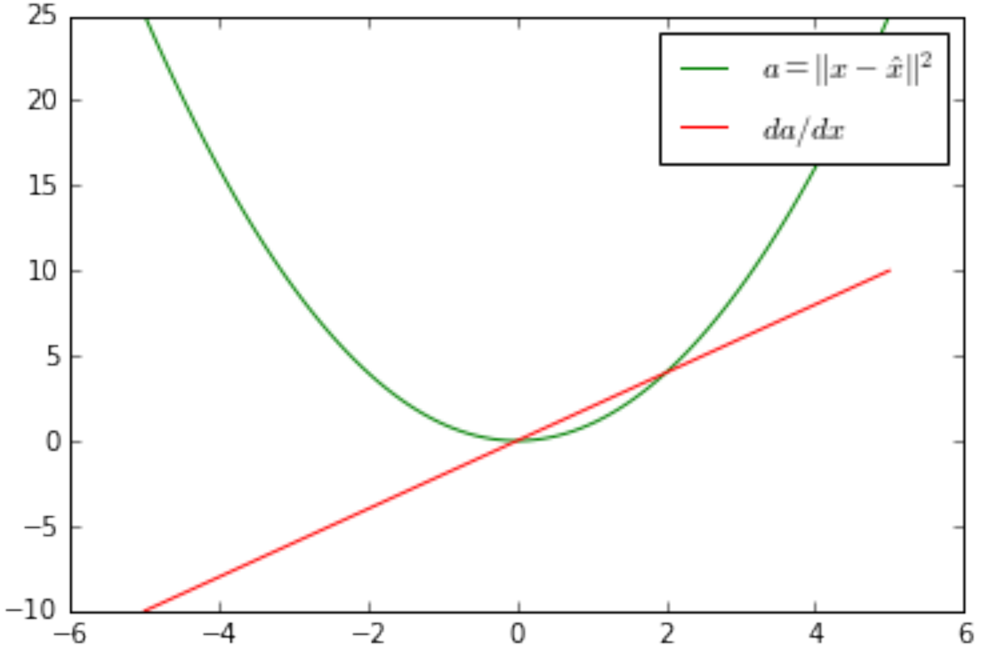
Avoid exponentianting large numbers → better stability

Euclidean Loss

Activation function $a(x) = 0.5\ \|y - x\|^2$

- ❑ Mostly used to measure the loss in **regression** tasks

Gradient with respect to the input $\frac{\partial a}{\partial x} = x - y$



Cross-entropy loss

Activation function $a(x) = -\sum_{k=1}^{K} y^{(k)} \log x^{(k)}, \quad y^{(k)} = \{0, 1\}$

Gradient with respect to the input $\frac{\partial a}{\partial x^{(k)}} = -\frac{1}{x^{(k)}}$

The cross-entropy is the most popular **classification loss** for classifiers that output probabilities (not SVM)

Cross-entropy loss couples well softmax/sigmoid module

- ❑ Often the modules are combined and joint gradients are computed

Generalization of logistic regression for more than 2 outputs

Case studies

Alexnet

- ❑ Or the modern version of it, VGGnet

ResNet

- ❑ From 14 to 1000 layers

Google Inception

- ❑ Networks as Direct Acyclic Graphs (DAG)

Alexnet

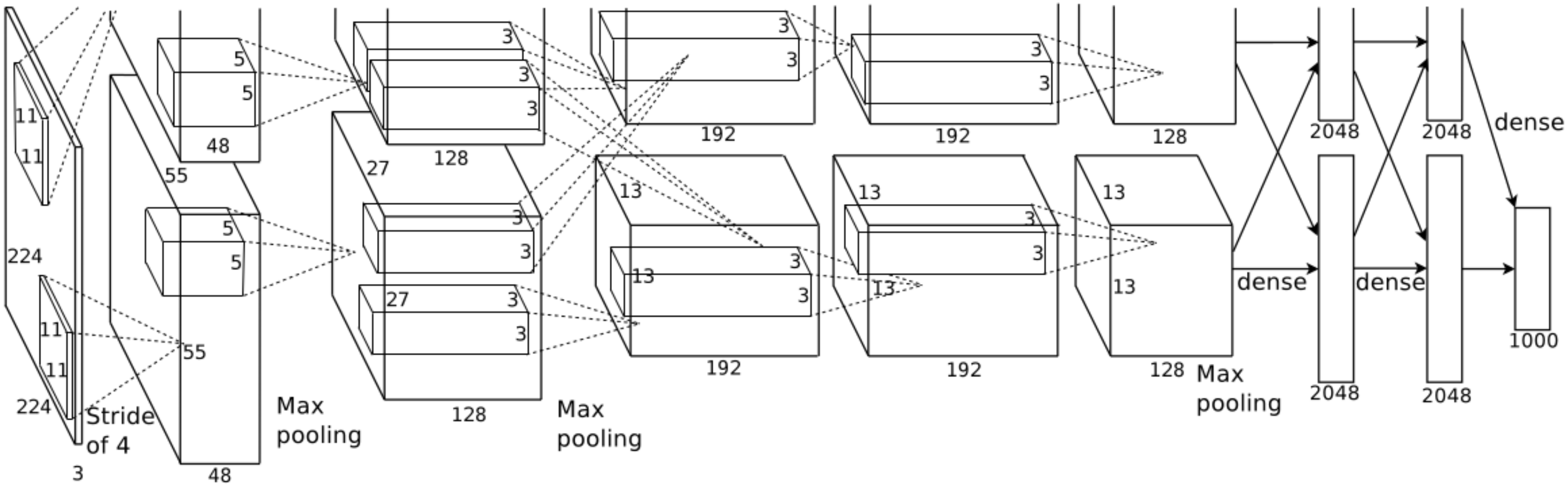
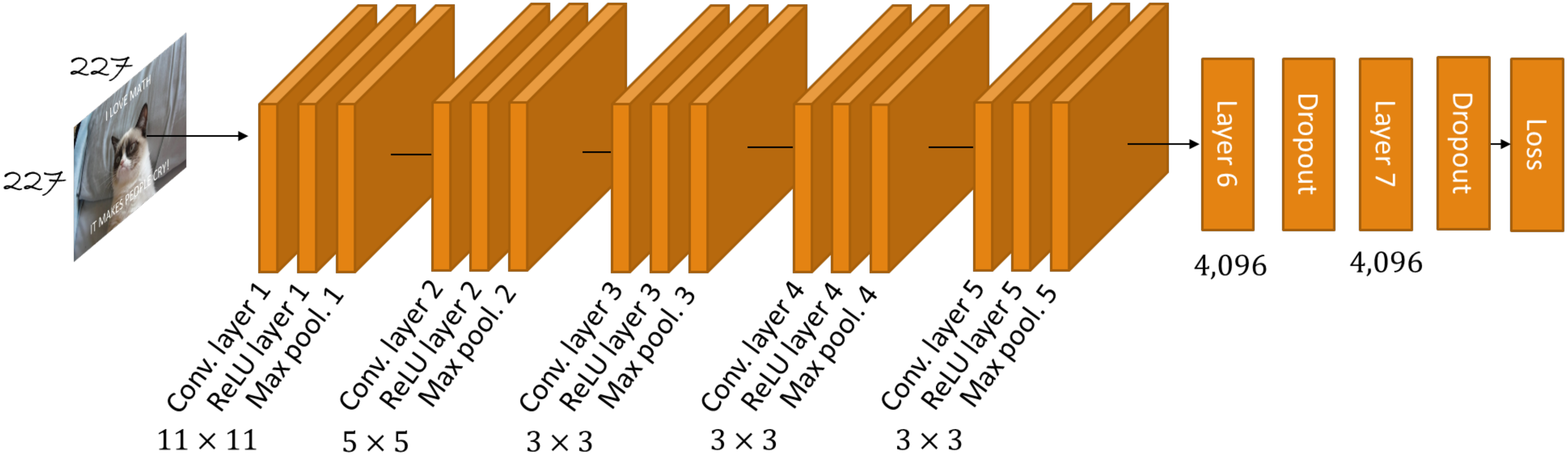


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

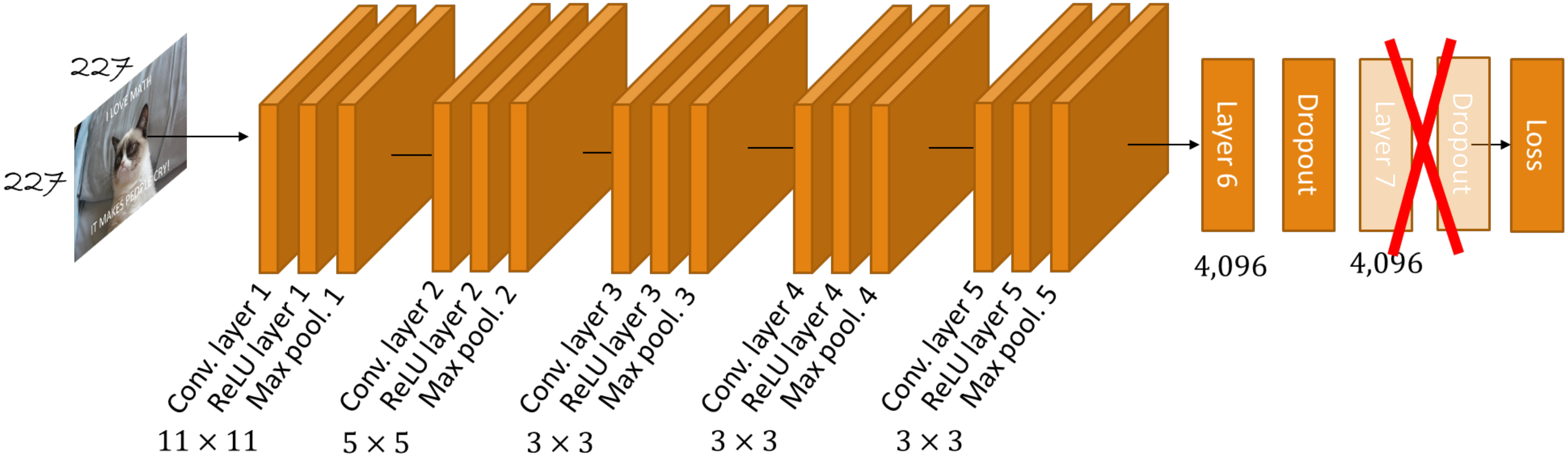
Architectural details

18.2% error in Imagenet



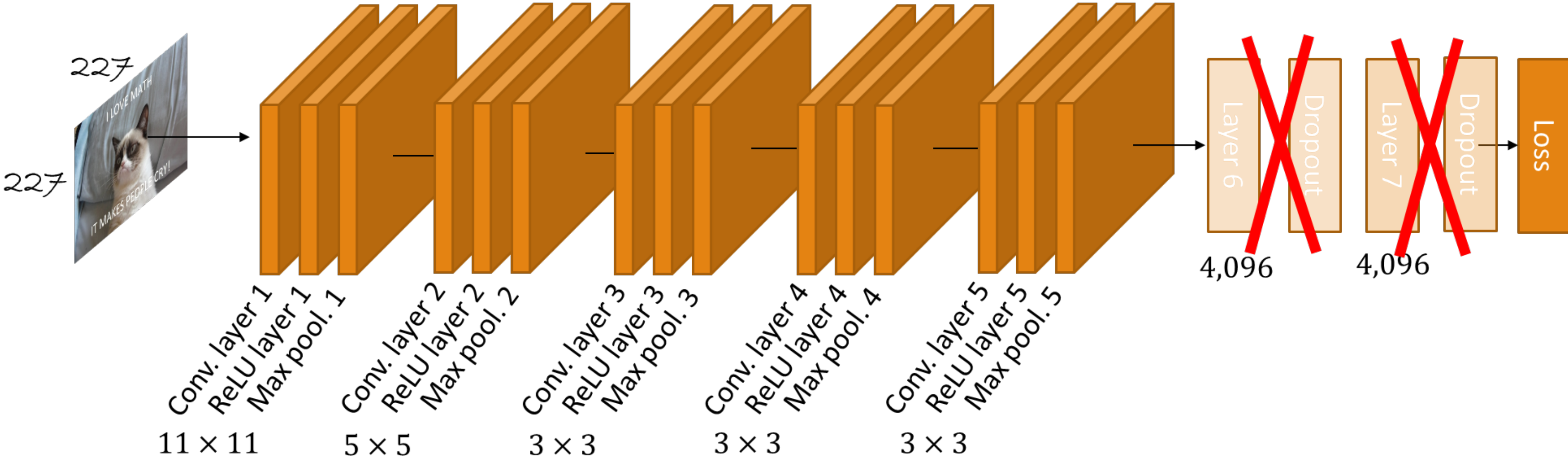
Removing layer 7

1.1% drop in performance, 16 million less parameters



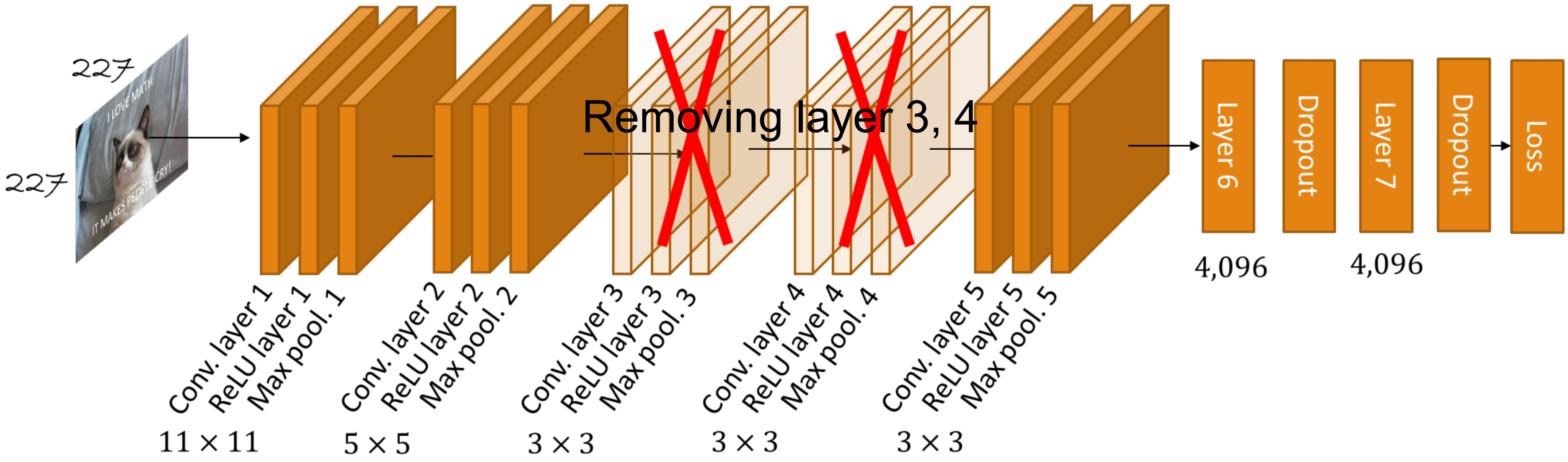
Removing layer 6, 7

5.7% drop in performance, 50 million less parameters



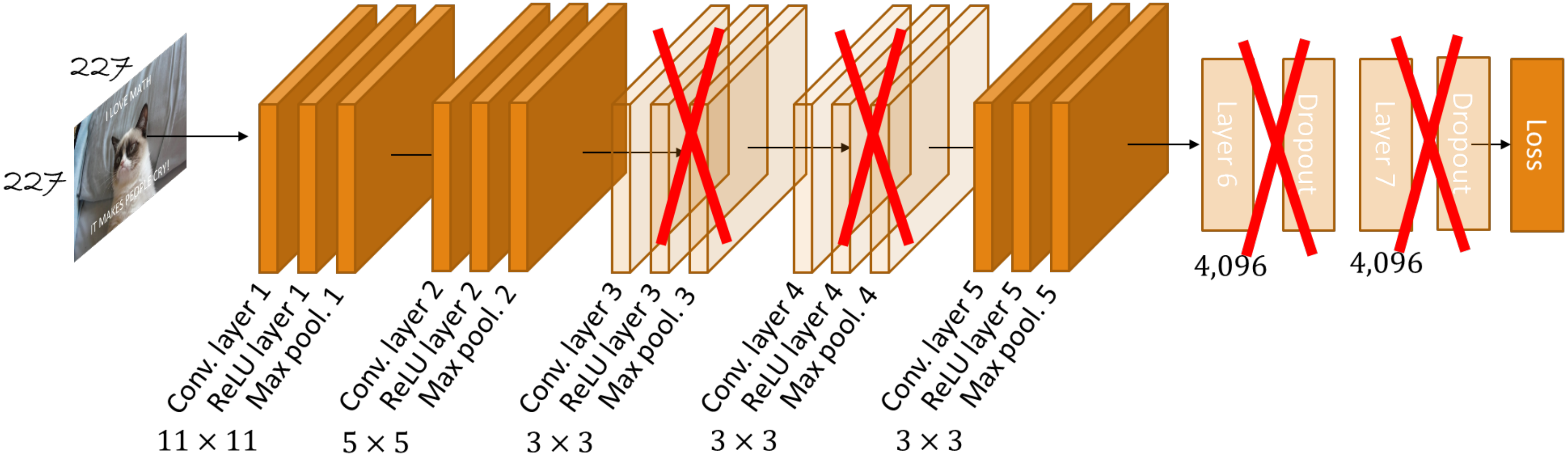
Removing layer 3, 4

3.0% drop in performance, 1 million less parameters. Why?

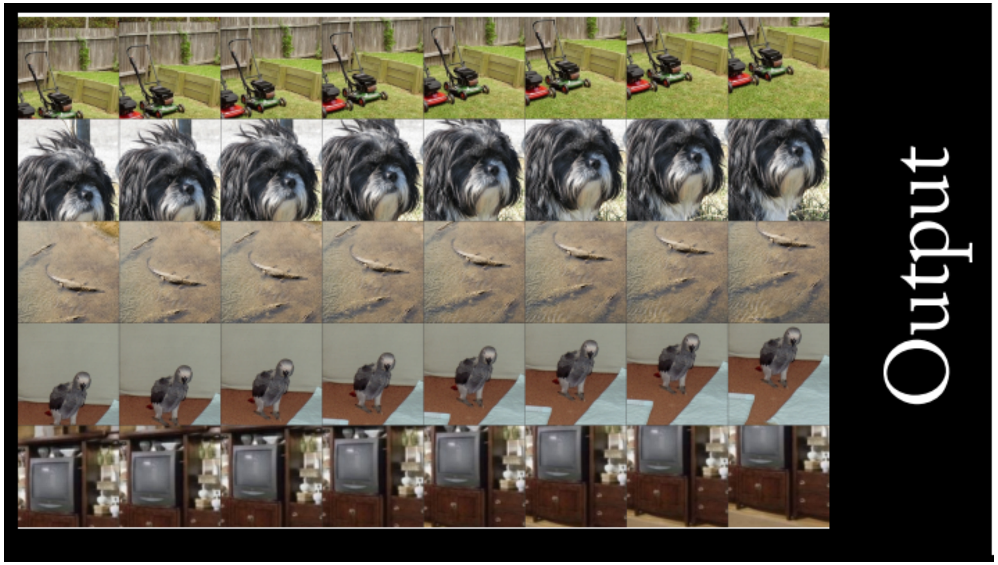


Removing layer 3, 4, 6, 7

33.5% drop in performance. Conclusion? Depth!

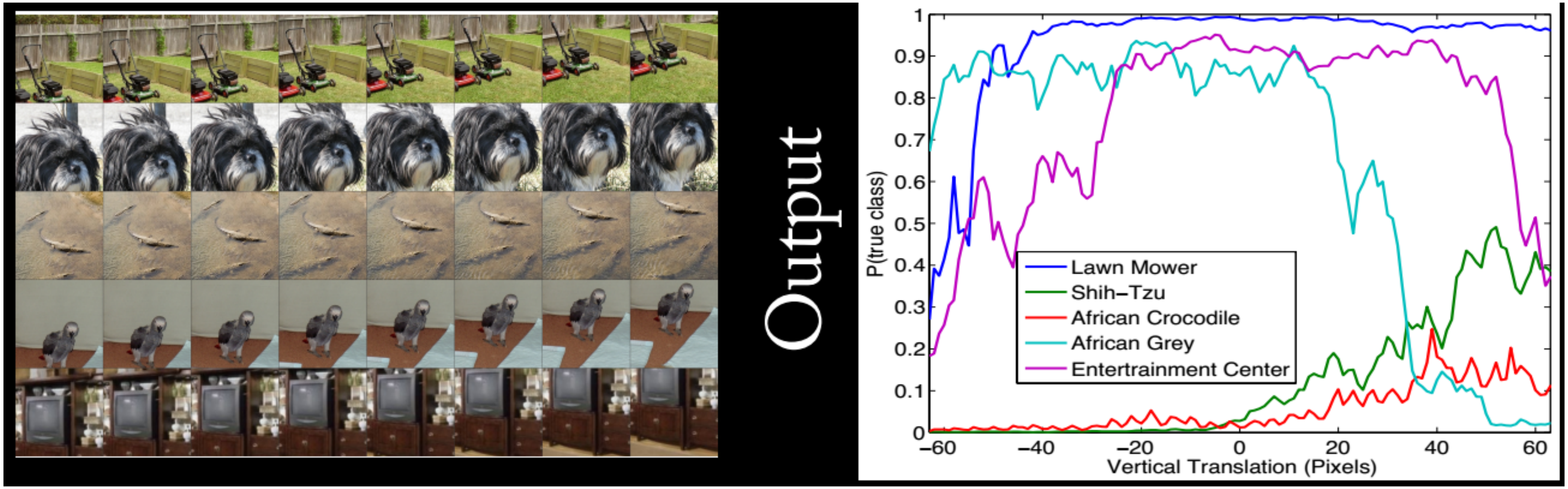


Quiz: Translation invariance?



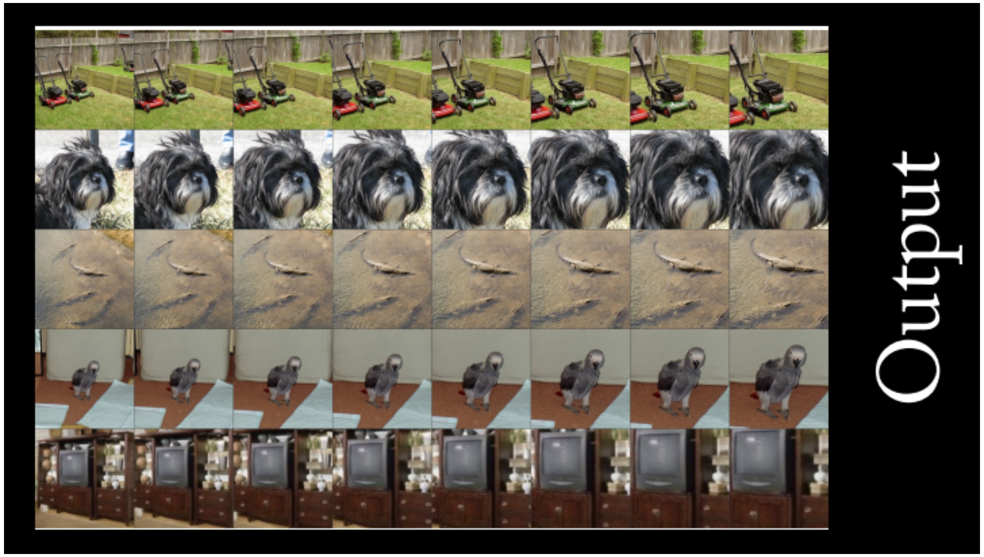
Credit: R. Fergus slides in Deep Learning Summer School 2016

Translation invariance



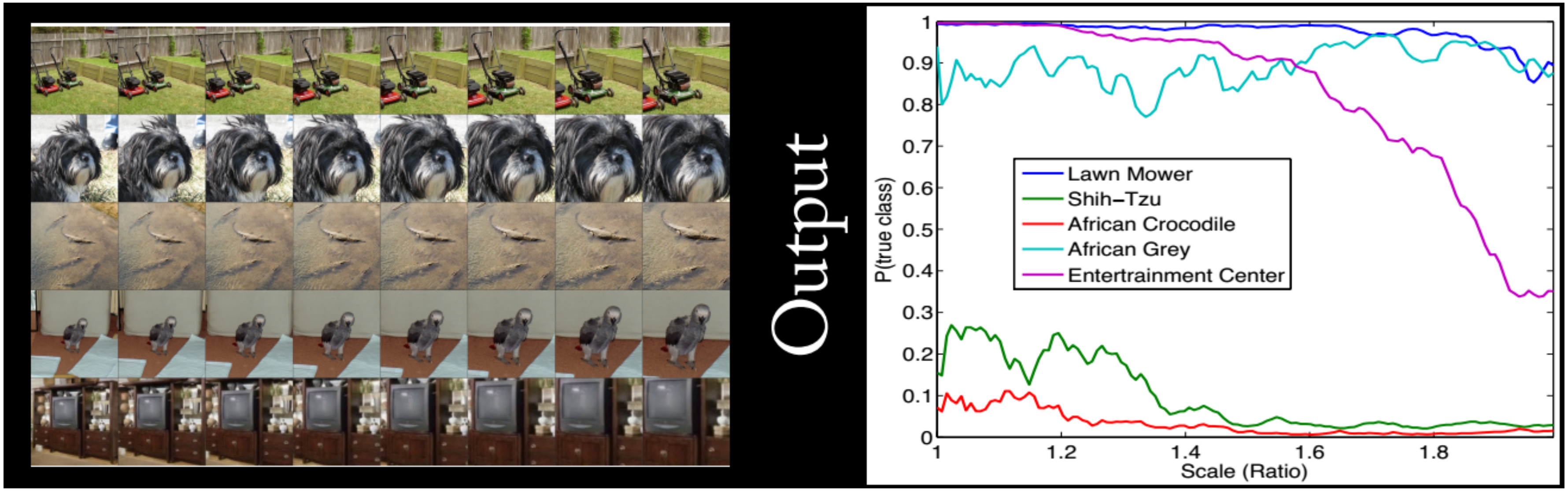
Credit: R. Fergus slides in Deep Learning Summer School 2016

Quiz: Scale invariance?



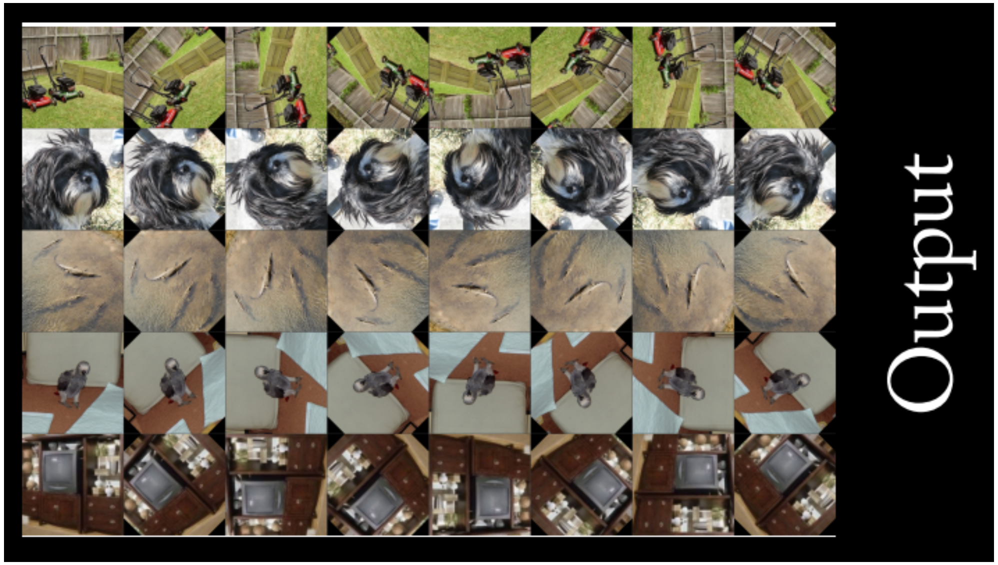
Credit: R. Fergus slides in Deep Learning Summer School 2016

Scale invariance



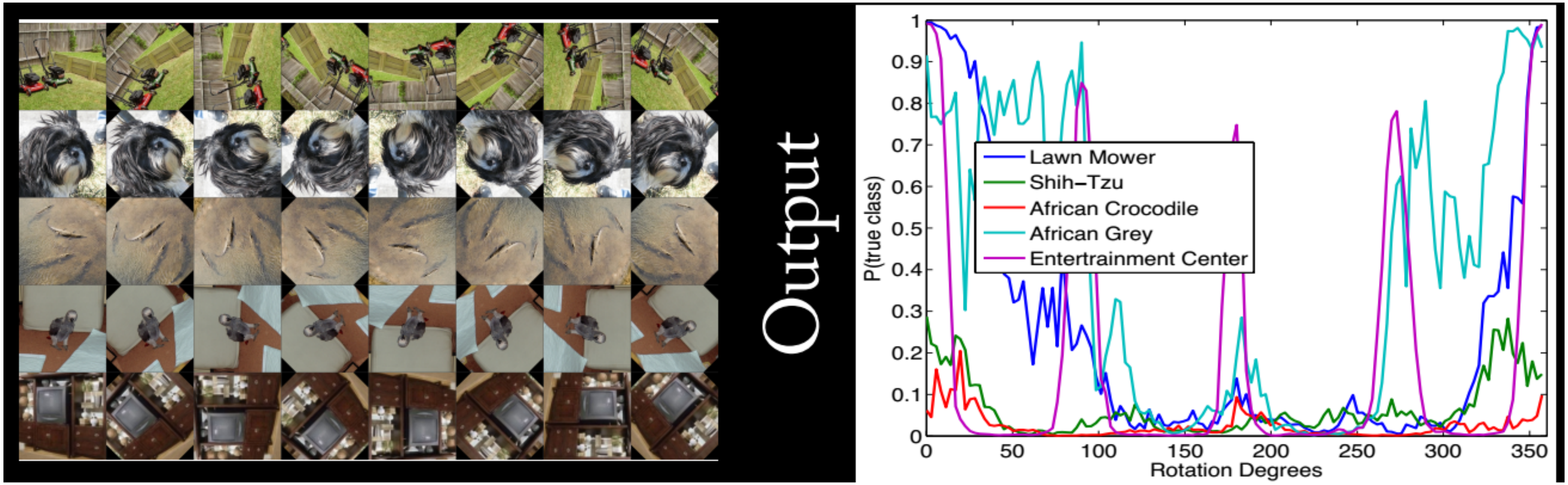
Credit: R. Fergus slides in Deep Learning Summer School 2016

Quiz: Rotation invariance?



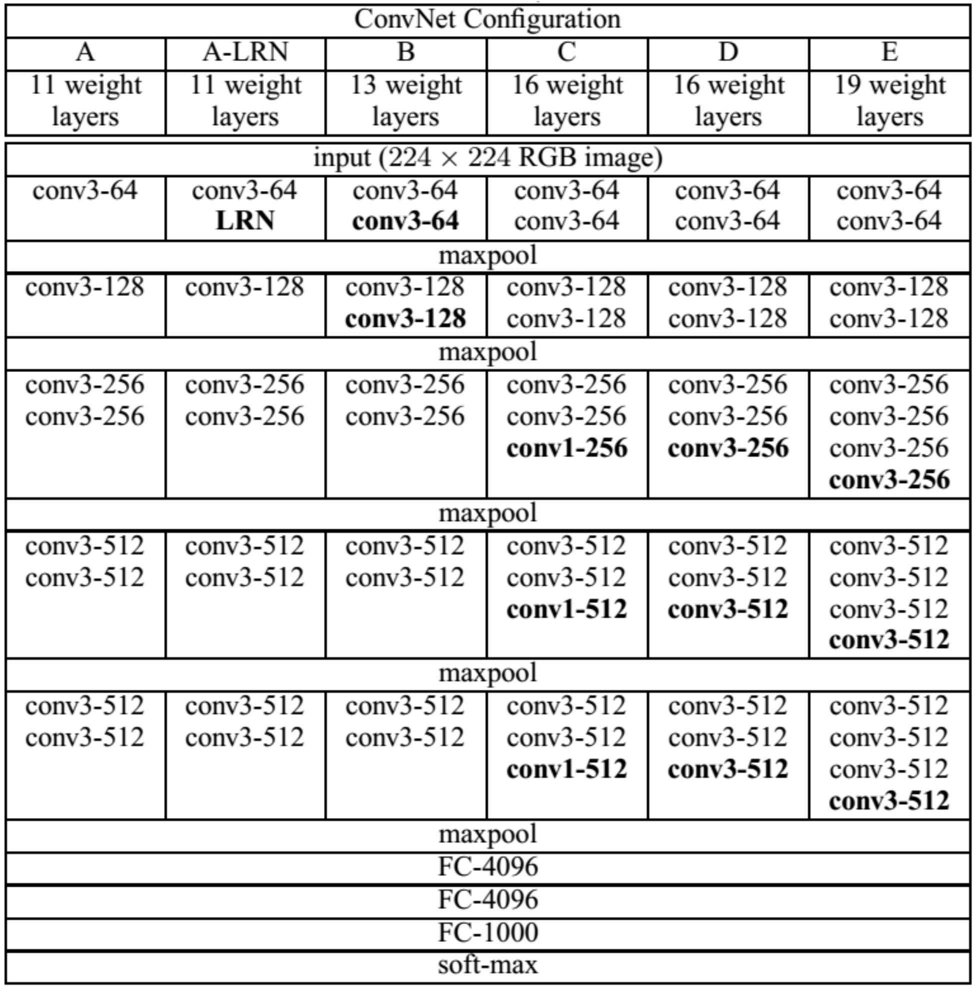
Credit: R. Fergus slides in Deep Learning Summer School 2016

Rotation invariance



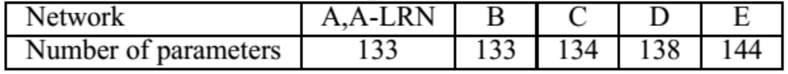
Credit: R. Fergus slides in Deep Learning Summer School 2016

More Depth? VGGnet



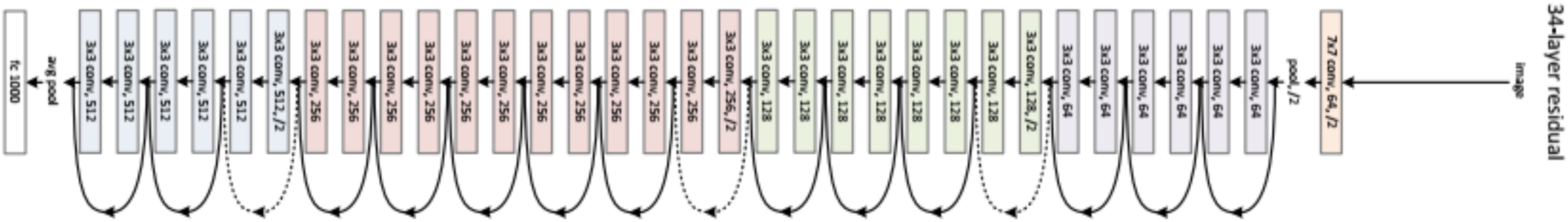
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).



Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

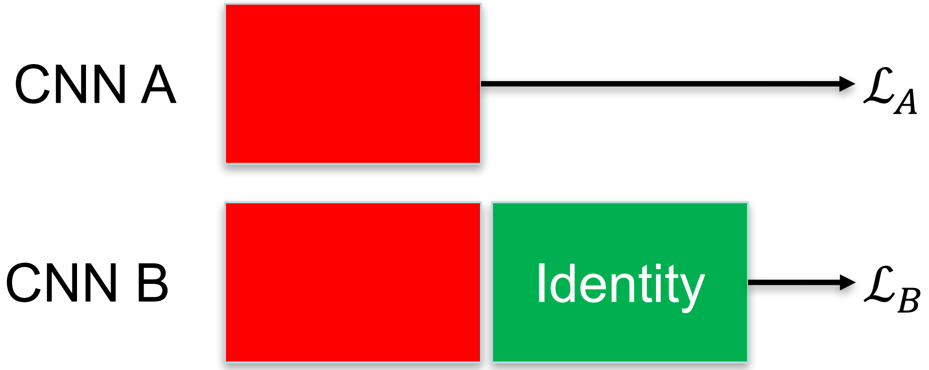
ResNet


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Hypothesis

Hypothesis: Is it possible to have a very deep network at least as accurate as averagely deep networks?

Thought experiment: Let’s assume two Convnets A, B. They are almost identical, in that B is the same as A, with extra “identity” layers. Since identity layers pass the information unchanged, the errors of the two networks should be similar. Thus, there is a Convnet B, which is at least as good as Convnet A w.r.t. training error



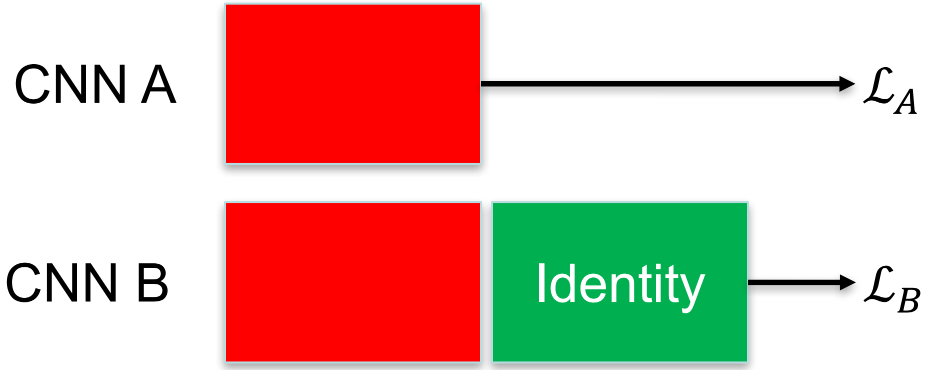
Testing hypothesis

Adding identity layers increases **training error**!!

- ❑ Training error, not testing error
- ❑ Not all networks are the same as easy to optimize

Performance degradation not caused by overfitting

- ❑ Just the task is harder



$\mathcal{L}_A^{train} < \mathcal{L}_B^{train}$!!!

Quiz: What looks weird?

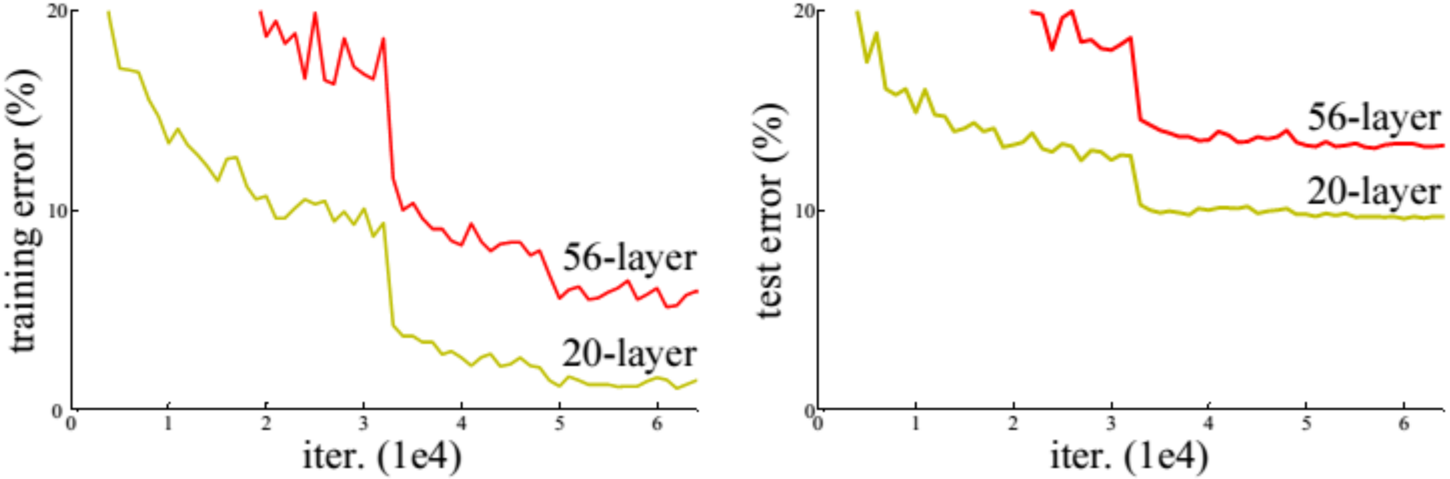


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Testing hypothesis

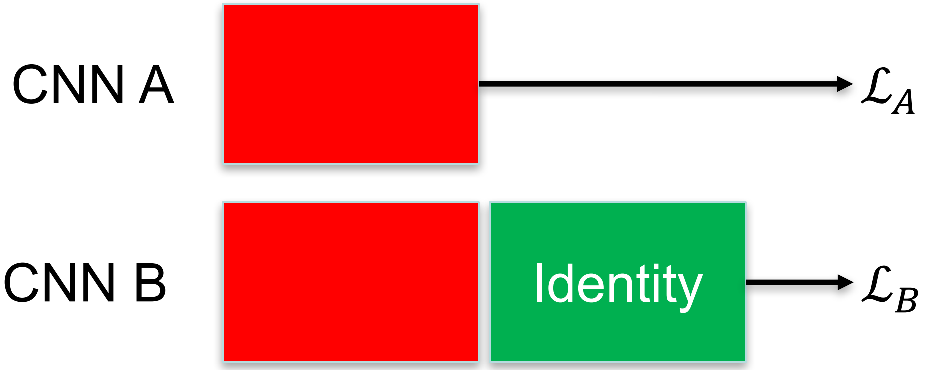
Adding identity layers increases **training error**!!

- ❑ Training error, not testing error

Performance degradation not caused by overfitting

- ❑ Just the optimization task is harder

Assuming optimizers are doing their job fine, it appears that not all networks are the same as easy to optimize



$\mathcal{L}_A^{train} < \mathcal{L}_B^{train}$!!!

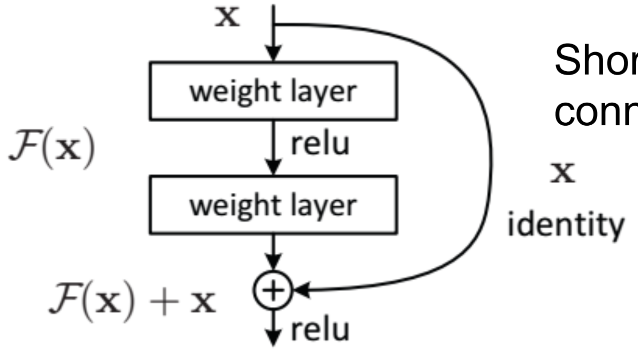
ResNet: Main idea

Layer models residual $\mathrm{F}(\mathrm{x}) = H(x) - x$ instead of $H(x)$

If anything, the optimizer can simply set the weights to 0

- ❑ This assumes that the identity mapping is indeed the optimal one

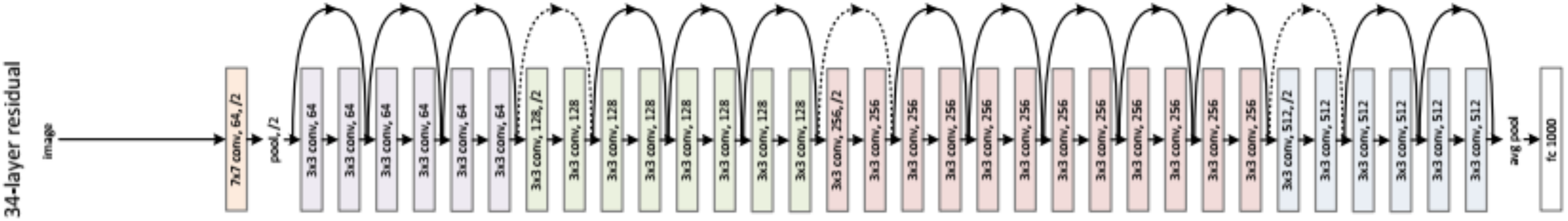
Adding identity layers should lead to larger networks that have <u>at least</u> lower training error



Shortcut/skip connections

Figure 2. Residual learning: a building block.

Smooth propagation



$$x_{l+1} = x_l + \mathrm{F}(\mathrm{x_l}) \quad x_{l+2} = x_{l+1} + \mathrm{F}(\mathrm{x_{l+1}}) \quad \ldots \quad x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

Additive relation between x_l, x_L

- ❑ Traditional NNs have multiplicative: $x_L = \prod_{i=l}^{L-1} W_i x_l$

Smooth backprop: $\frac{\partial \mathcal{L}}{\partial x_l} = \frac{\partial \mathcal{L}}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial \mathcal{L}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_L} \sum_{i=l}^{L-1} F(x_i)\right)$

- ❑ The loss closest to the output $\frac{\partial \mathcal{L}}{\partial x_L}$ is always there in the gradients

No degradation anymore

Without residual connections deeper networks are untrainable

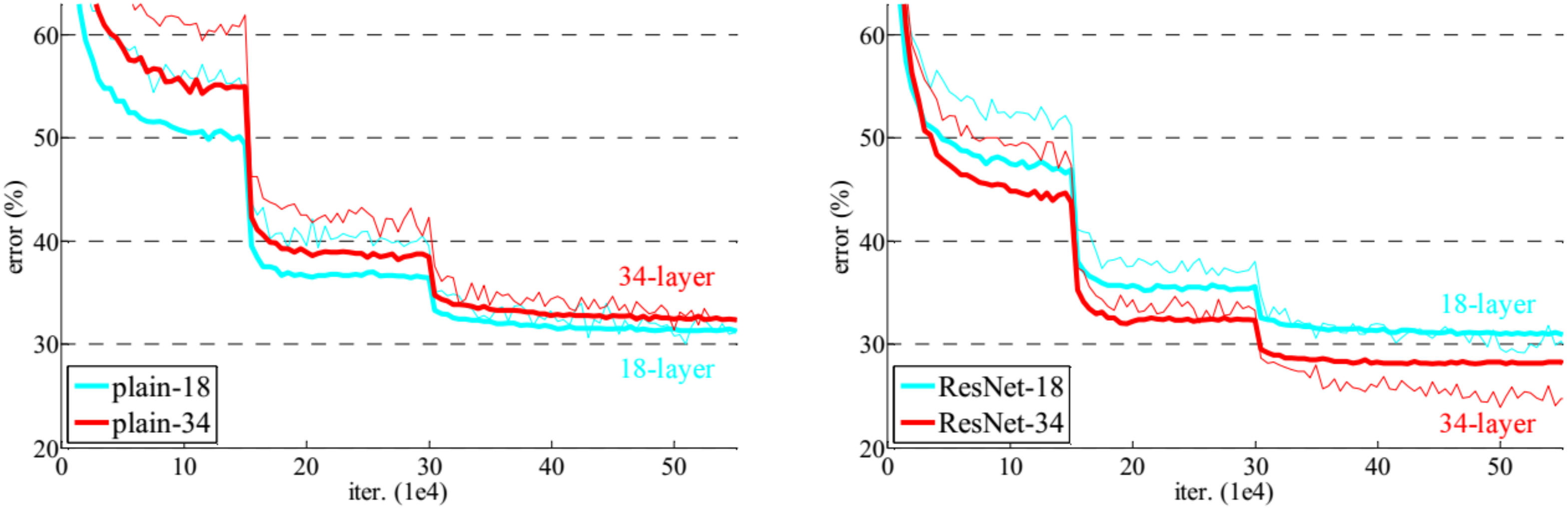


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

ResNet vs Highway Nets

ResNet: $y = H(x) - x$

Highway Nets: $y = H(x) \cdot T_x - x \cdot (1 - T_x)$

ResNet ⊆ Highway Nets

- ResNet ≡ Highway Nets: $T_x \sim Binomial$ with $E[T_x] = 0.5$

ResNet data independent

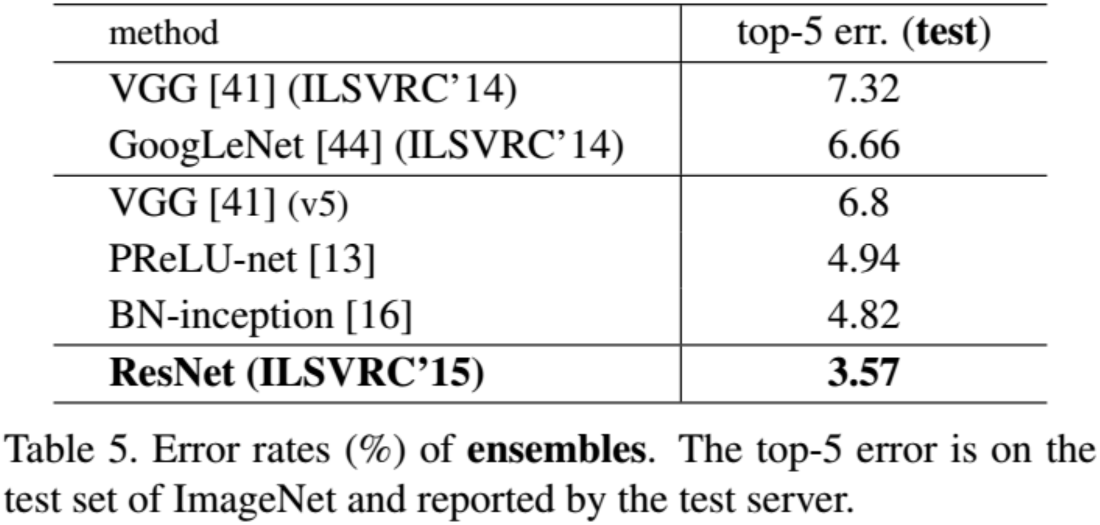
- Curse or blessing, depending on point of view
- Definitely simpler

ResNet breaks records

Ridiculously low error in ImageNet

Up to 1000 layers ResNets trained

- ❑ Previous deepest network ~30-40 layers on simple datasets



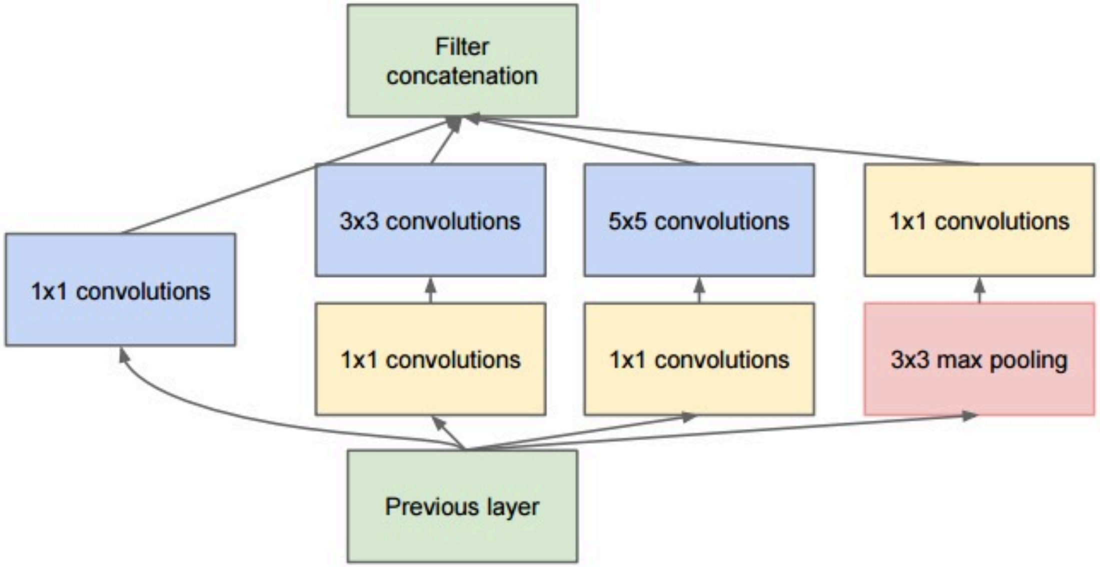
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

Google Inception V1

Instead of having convolutions (e.g. 3×3) directly, first reduce features by 1×1 convolutions

- ❑ E.g., assume we have 256 features in the previous layer
- ❑ Convolve with $256\times64\times1\times1$
- ❑ Then convolve with $64\times64\times3\times3$
- ❑ Then convolve with $64\times256\times1\times1$



Credit: https://culurciello.github.io/tech/2016/06/04/nets.html

Google Inception V2-V3

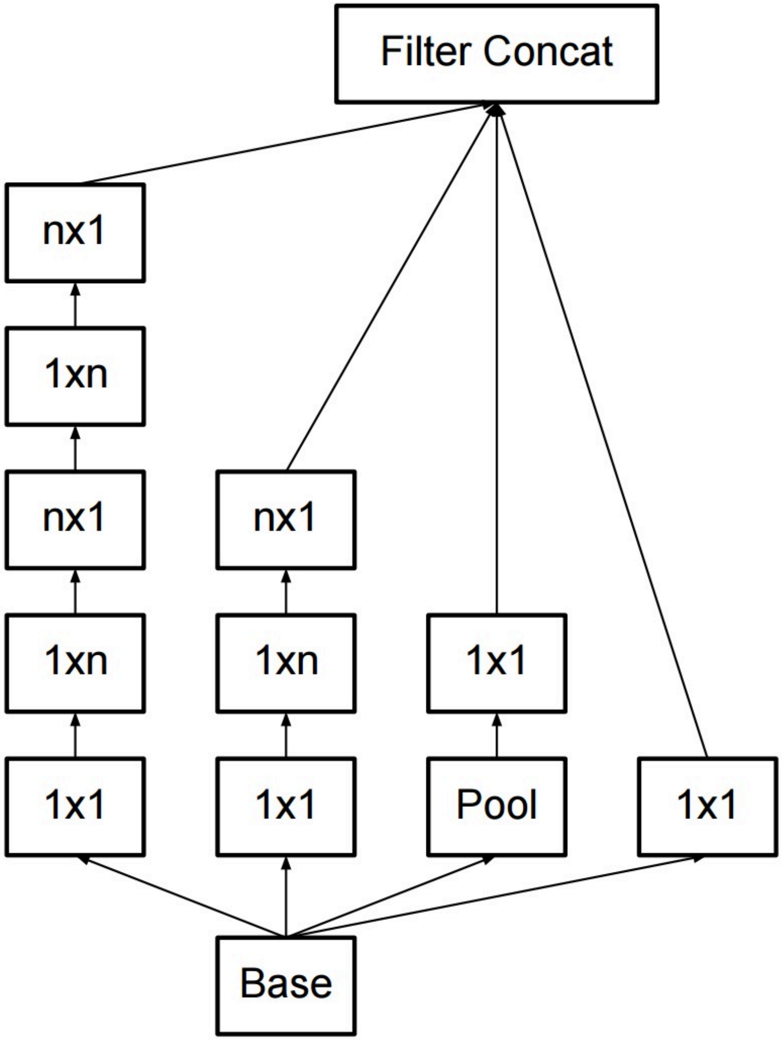
Decompose square filters to flattened convolutions

- ❑ Fewer operations $O(n)$ vs $O(n^2)$

Use 3×3 filters as 5×5 can be written as a function of 3×3

- ❑ Inspired by VGGNet

Balance depth and width

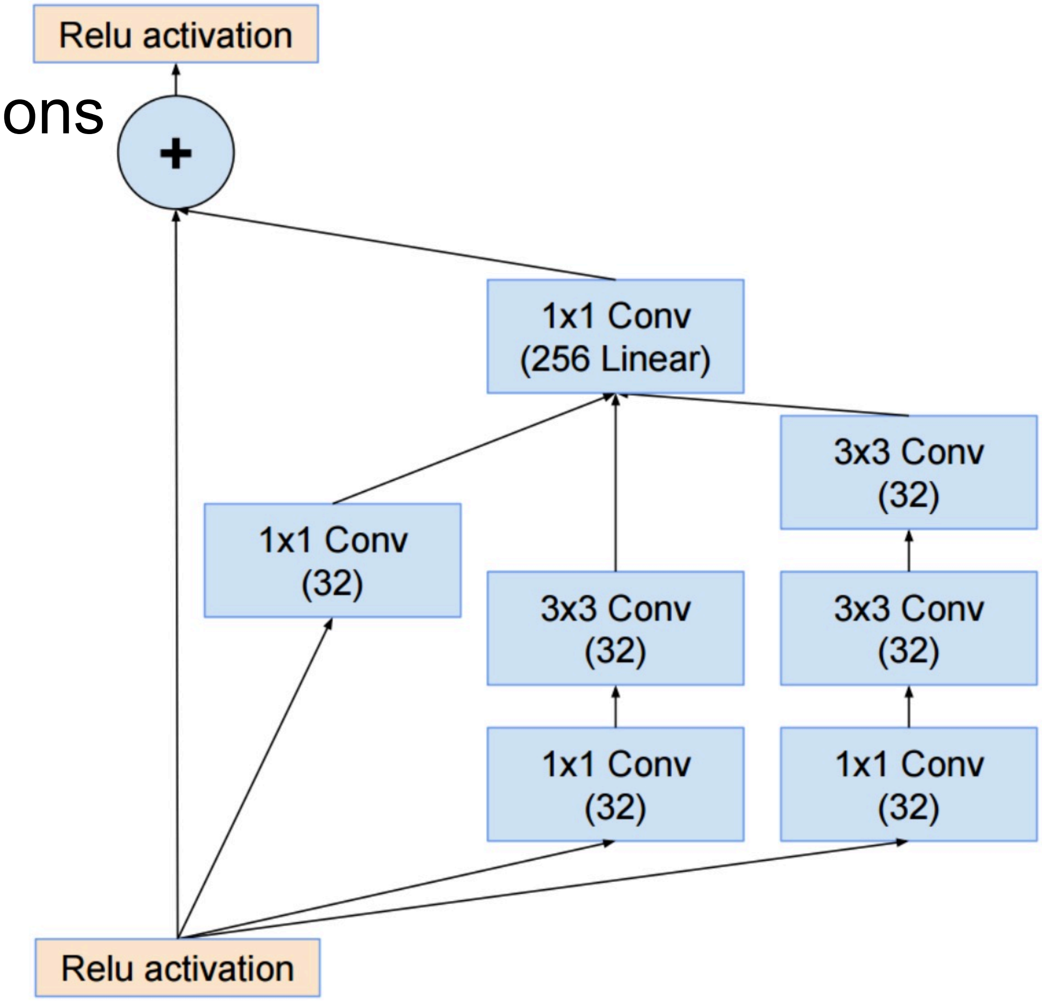


Credit: https://culurciello.github.io/tech/2016/06/04/nets.html

Google Inception V4

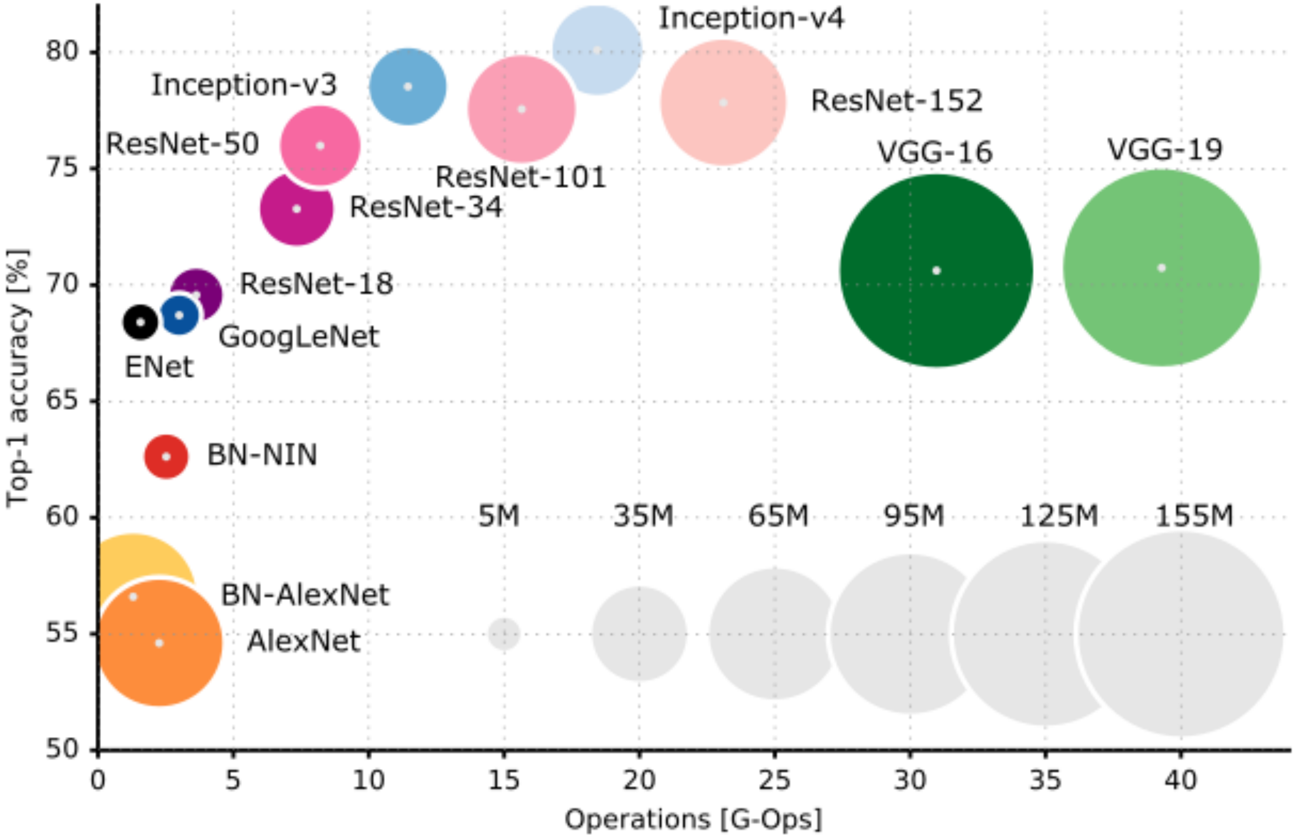
Similar to Inception V3
plus shortcut connections

- ❑ Inspired by ResNet



Credit: https://culurciello.github.io/tech/2016/06/04/nets.html

State-of-the-art



Credit: https://culurciello.github.io/tech/2016/06/04/nets.html

Recurrent Networks

So far, all tasks assumed ***stationary*** data

Neither all data, nor all tasks are stationary though

Sequential data

Or …

What

Or …

What about

Or …

What about inputs that appear in sequences, such as text? Could a neural network handle such modalities?

Memory needed

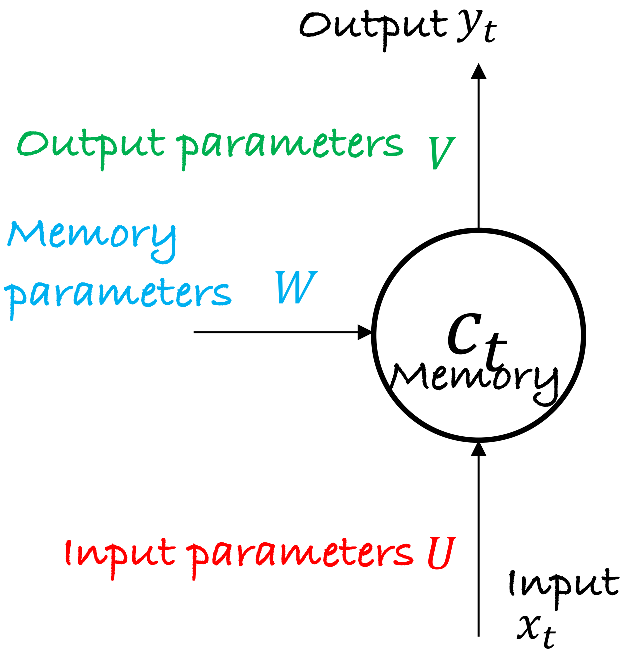
$$\boxed{\Pr(x) = \prod_i \Pr(x_i \,|\, x_1, \ldots, x_{i-1})}$$

What about inputs that appear in sequences, such as text? Could a neural network handle such modalities?

Recurrent Networks

Simplest model

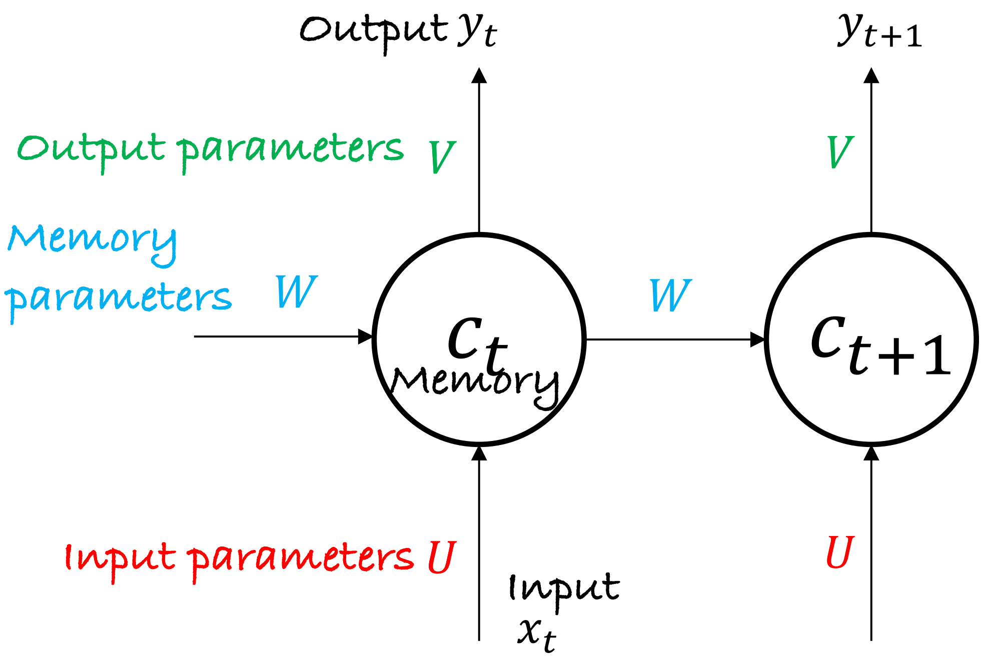
- ❑ Input with parameters U
- ❑ Memory embedding with parameters W
- ❑ Output with parameters V



Recurrent Networks

Simplest model

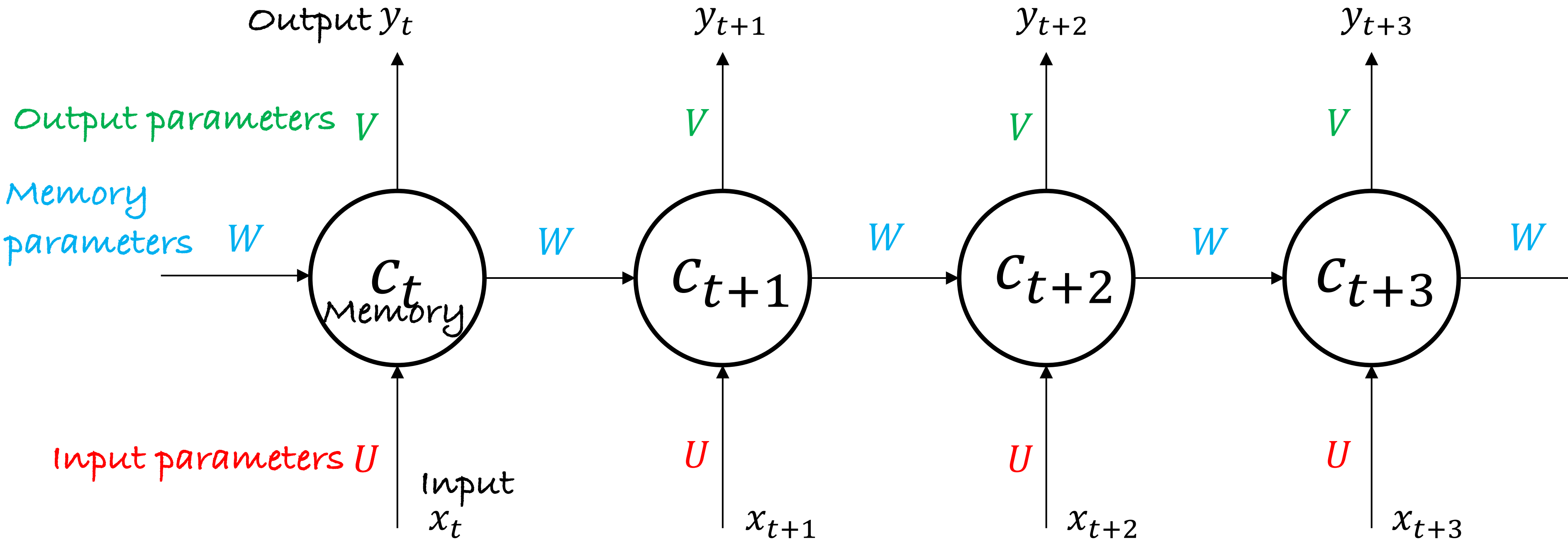
- ❑ Input with parameters U
- ❑ Memory embedding with parameters W
- ❑ Output with parameters V



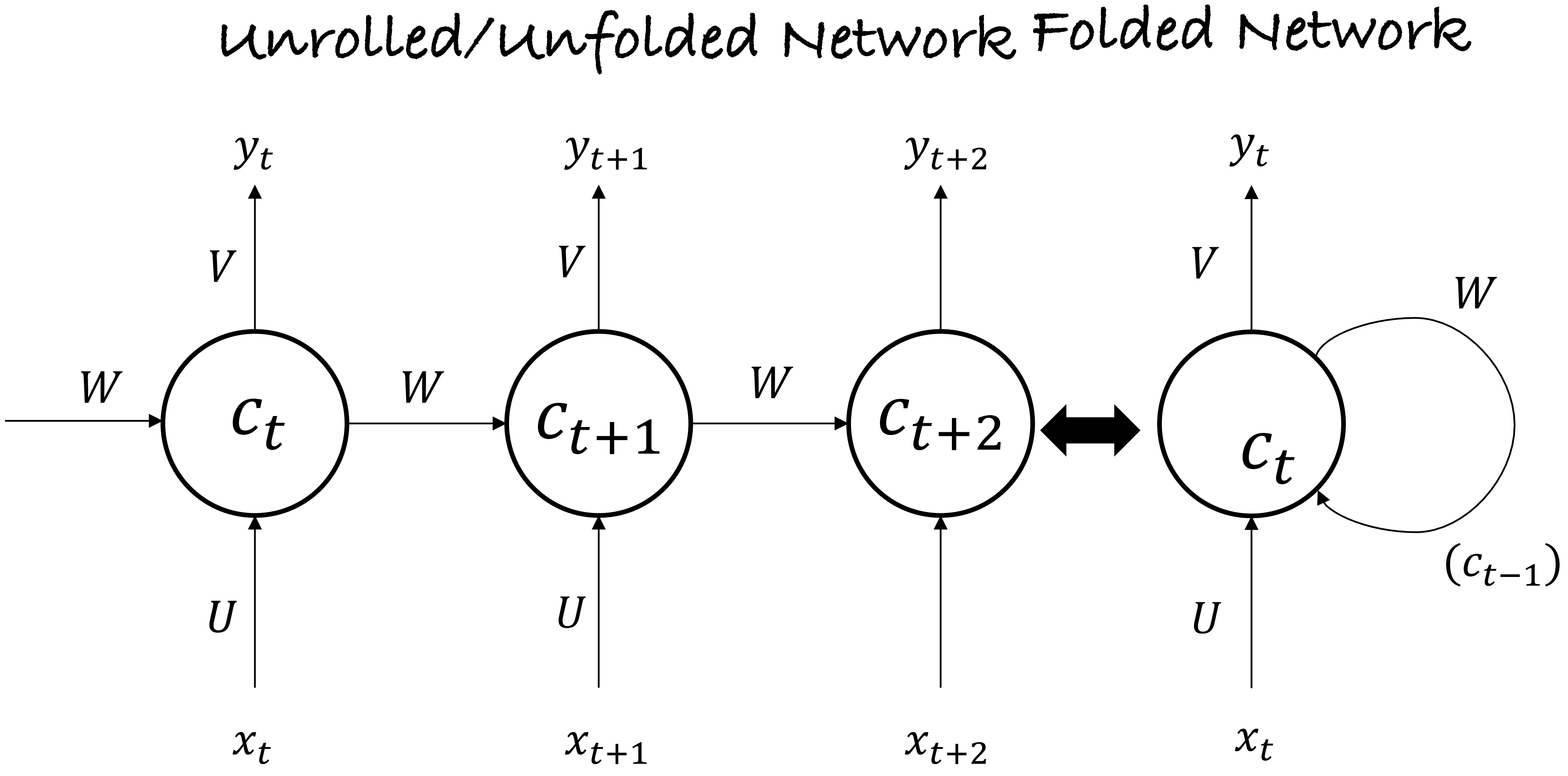
Recurrent Networks

Simplest RNN

- ❑ Input with parameters U
- ❑ Memory embedding with parameters W
- ❑ Output with parameters V



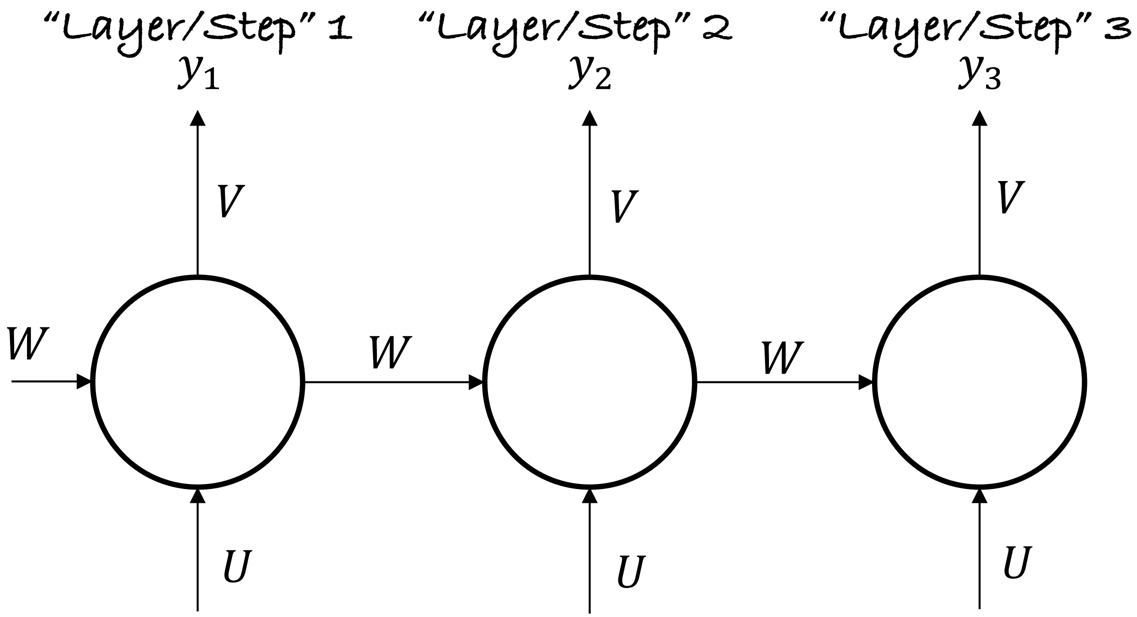
Folding the memory



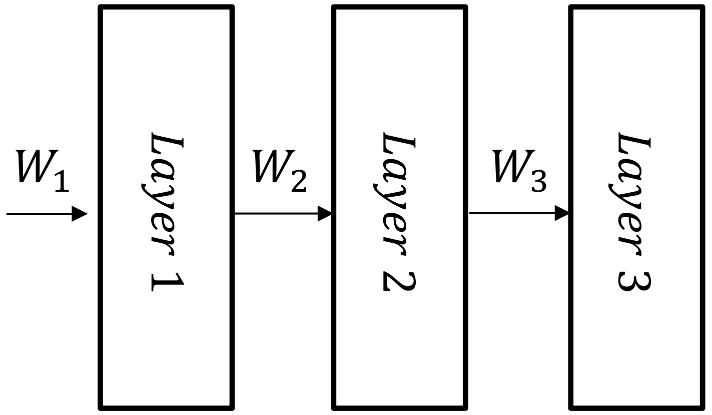
RNN vs NN

What is really different?

- ❑ Steps instead of layers
- ❑ Step parameters shared whereas in a Multi-Layer Network they are different



3-gram Unrolled Recurrent Network



3-layer Neural Network

Training an RNN

Cross-entropy loss

$$P = \prod_{t,k} y_{tk}^{l_{tk}} \quad \Rightarrow \quad \mathcal{L} = -\log P = \sum_t \mathcal{L}_t = -\frac{1}{T} \sum_t l_t \log y_t$$

Backpropagation Through Time (BPTT)

Be careful of the recursion. The non-linearity is influencing itself. The gradients at one time step depends on gradients on previous time steps

- ❑ Like in NN → Chain Rule
- ❑ Only difference: Gradients survive over time steps

RNN Gradients

$$\mathcal{L} = L(c_T(c_{T-1}(\ldots(c_1(x_1, c_0; W); W); W); W)$$

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{\mathrm{t}} \frac{\partial \mathcal{L}_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} = \frac{\partial \mathcal{L}}{\partial \mathrm{c}_{\mathrm{t}}} \cdot \frac{\partial c_t}{\partial \mathrm{c}_{\mathrm{t}-1}} \cdot \frac{\partial c_{t-1}}{\partial \mathrm{c}_{\mathrm{t}-2}} \cdot \ldots \cdot \frac{\partial c_{\tau+1}}{\partial \mathrm{c}_{\tau}} \leq \eta^{t-\tau} \frac{\partial \mathcal{L}_t}{\partial c_t}$$

The RNN gradient is a recursive product of $\frac{\partial c_t}{\partial \mathrm{c}_{\mathrm{t}-1}}$

Vanishing/Exploding gradients

$$\left.\frac{\partial \mathcal{L}}{\partial c_t} = \underset{<1}{\frac{\partial \mathcal{L}}{\partial \mathrm{c}_{\mathrm{T}}}} \cdot \underset{<1}{\frac{\partial c_T}{\partial \mathrm{c}_{\mathrm{T}-1}}} \cdot \frac{\partial c_{T-1}}{\partial \mathrm{c}_{\mathrm{T}-2}} \cdot \ldots \cdot \underset{<1}{\frac{\partial c_{t+1}}{\partial \mathrm{c}_{c_t}}}\quad\right\} \quad \frac{\partial \mathcal{L}}{\partial W} \ll 1 \Rightarrow \text{Vanishing gradient}$$

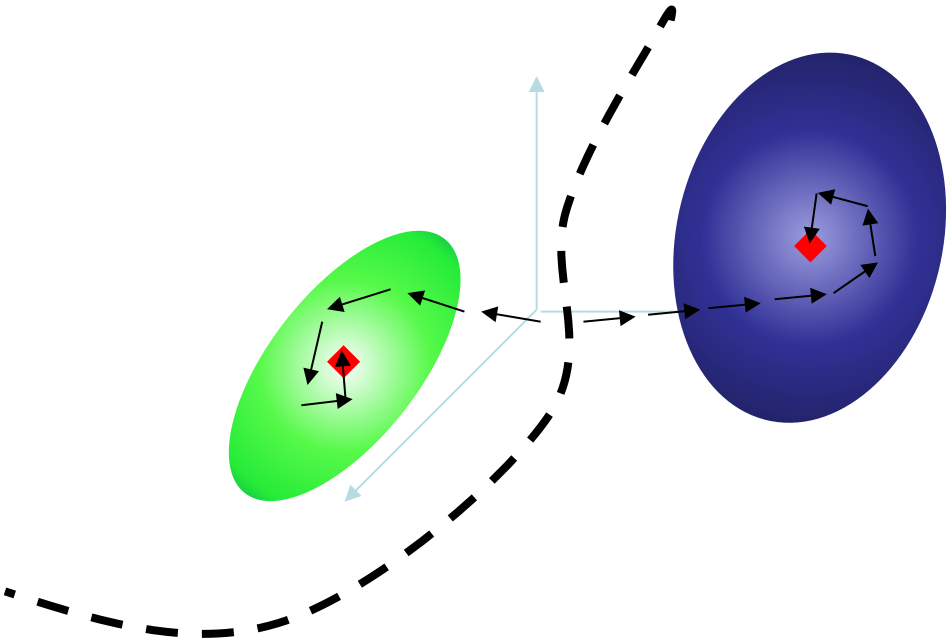
$$\left.\frac{\partial \mathcal{L}}{\partial c_t} = \underset{>1}{\frac{\partial \mathcal{L}}{\partial \mathrm{c}_{\mathrm{T}}}} \cdot \underset{>1}{\frac{\partial c_T}{\partial \mathrm{c}_{\mathrm{T}-1}}} \cdot \frac{\partial c_{T-1}}{\partial \mathrm{c}_{\mathrm{T}-2}} \cdot \ldots \cdot \underset{>1}{\frac{\partial c_1}{\partial \mathrm{c}_{c_t}}}\quad\right\} \quad \frac{\partial \mathcal{L}}{\partial W} \gg 1 \Rightarrow \text{Exploding gradient}$$

RNN & Chaotic Systems

The latent memory space is composed of multiple dimensions

A subspace of the memory state space can store information if multiple basins ◆ of attraction in some dimensions exist

Gradients must be strong near the basin boundaries



RNN & Chaotic Systems

In the figures $\mathrm{x_t} \propto c_t$ and $x_t \propto \mathrm{F}(Wx_{t-1} + Uu_t + b)$

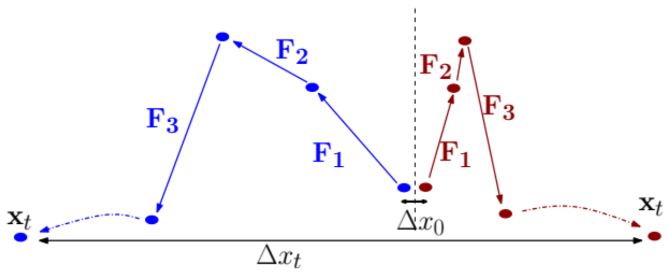


Figure 4. This diagram illustrates how the change in $\mathbf{x}_t$, $\Delta\mathbf{x}_t$, can be large for a small $\Delta\mathbf{x}_0$. The blue vs red (left vs right) trajectories are generated by the same maps $F_1, F_2, \ldots$ for two different initial states.

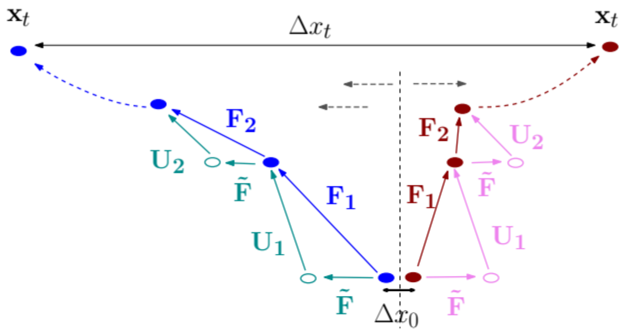


Figure 5. Illustrates how one can break apart the maps $F_1, ..F_t$ into a constant map $\tilde{F}$ and the maps $U_1, .., U_t$. The dotted vertical line represents the boundary between basins of attraction, and the straight dashed arrow the direction of the map $\tilde{F}$ on each side of the boundary. This diagram is an extension of Fig. 4.

Figures from:

Advanced RNN: LSTM

$\sigma \in (0,1)$: control gate – something like a switch

$\tanh \in (-1,1)$: recurrent nonlinearity

$$i = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

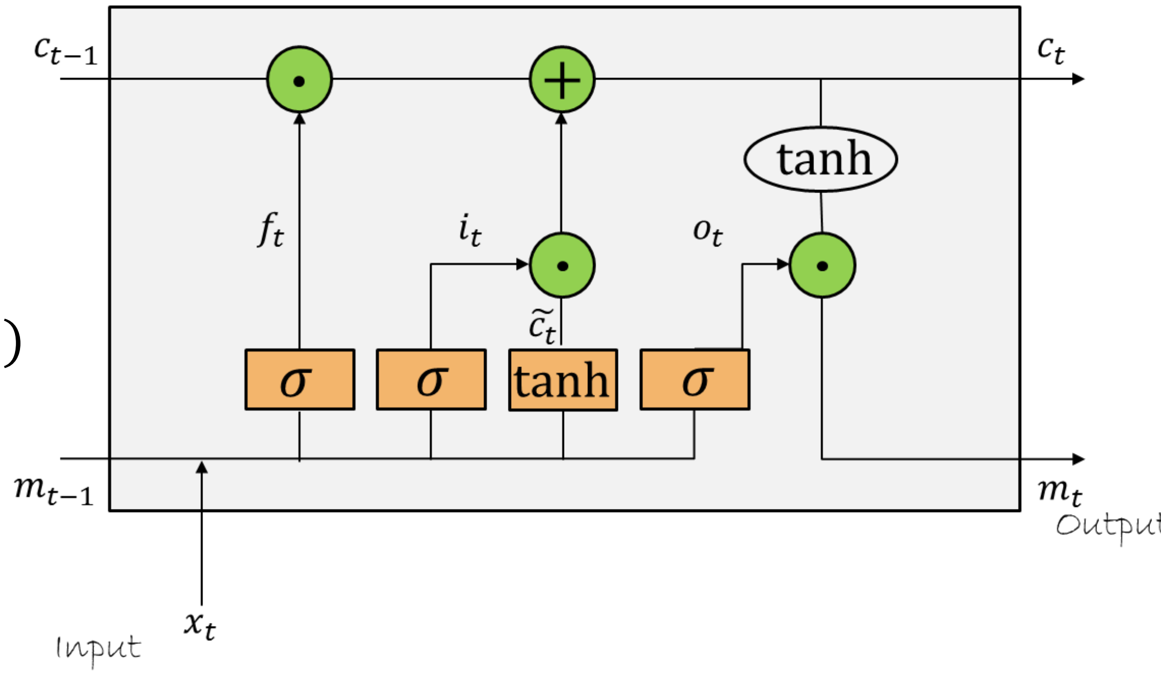
$$f = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\tilde{c}_t = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \tilde{c}_t \odot i$$

$$m_t = \tanh(c_t) \odot o$$



Computer Vision by Learning

Cees Snoek, UvA

Arnold W.M. Smeulders, UvA

Efstratios Gavves, UvA

Laurens van der Maaten Facebook

Advanced School for Computing and Imaging

University of Amsterdam

Overview Day 2

Vision in the deep learning era

1. Convolutional Neural Networks
2. Case studies
3. Recurrent Networks

Action recognition by learning

4. Video representations
5. Spatiotemporal localization
6. VideoLSTM

Goal of action recognition

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

Kissing

Shaking hands

Quiz: What matters for actions?

An action has . . .

Quiz: What matters for actions?

An action has . . .

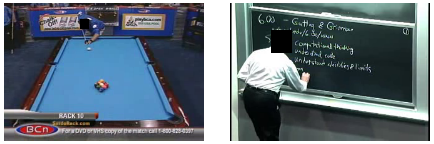
- actor
- appearance
- motion
- objects
- spatial extent
- temporal extent
- intention
- . . .

4. Video representations

In this chapter we consider video representation learning for action recognition. We analyze existing algorithms for their ability to capture intrinsic and extrinsic action properties, including actor, appearance, motion, objects, spatial and temporal extent.

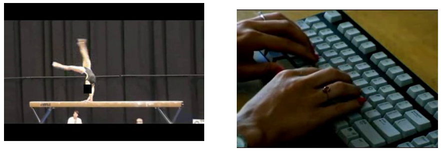
Action datasets are small scale

UCF101



101 classes / 13,320 clips / web video

THUMOS14



101 classes / 15,915 clips / web video

Hollywood2

12 classes / 1,707 clips / movies

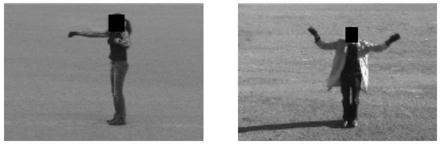
HMDB51

51 classes / 6,766 clips / diverse video

UCF Sports

10 classes / 150 clips / sports broadcasts

KTH



6 classes by 25 actors

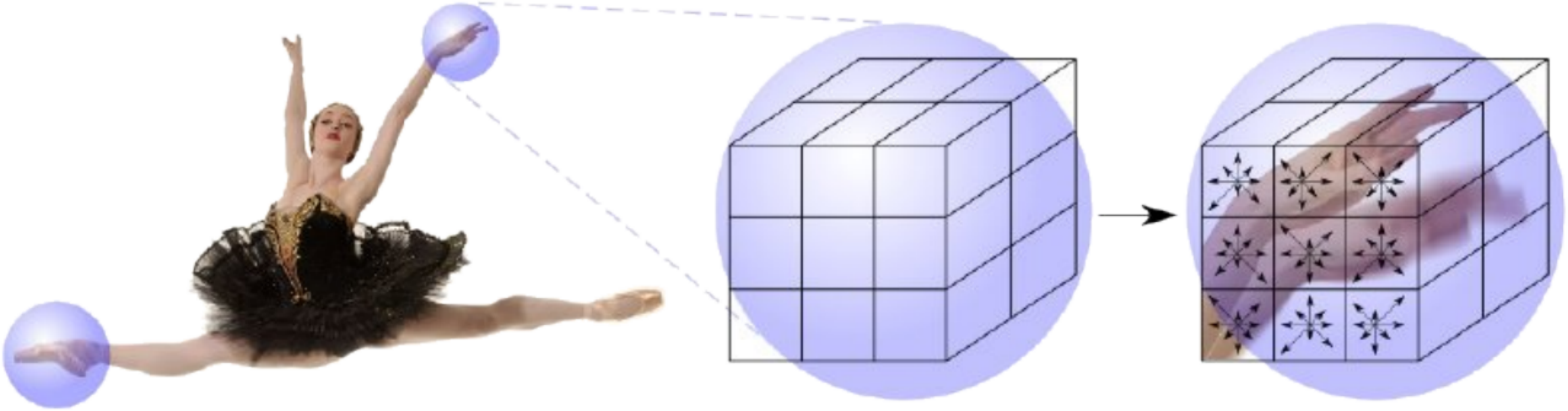
Recently, many new datasets proposed

Shallow action recognition

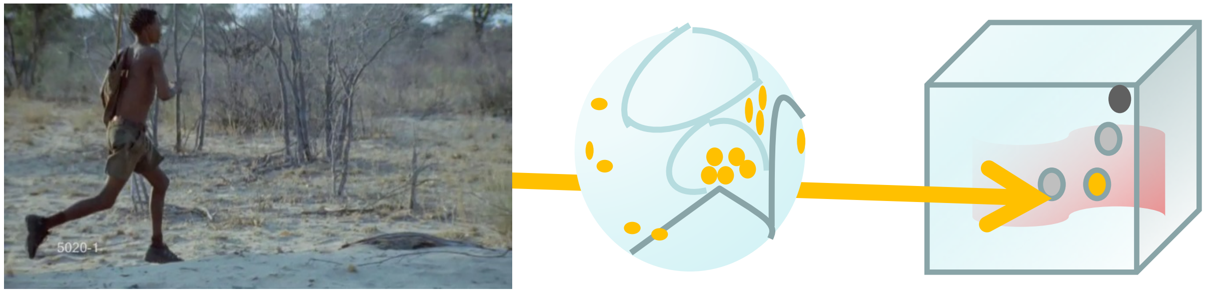
Spatio-temporal Interest point detection

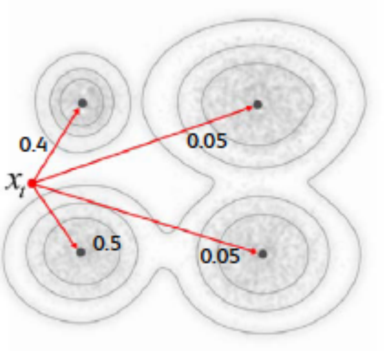
Space-time patch/trajectory

Space-time descriptor



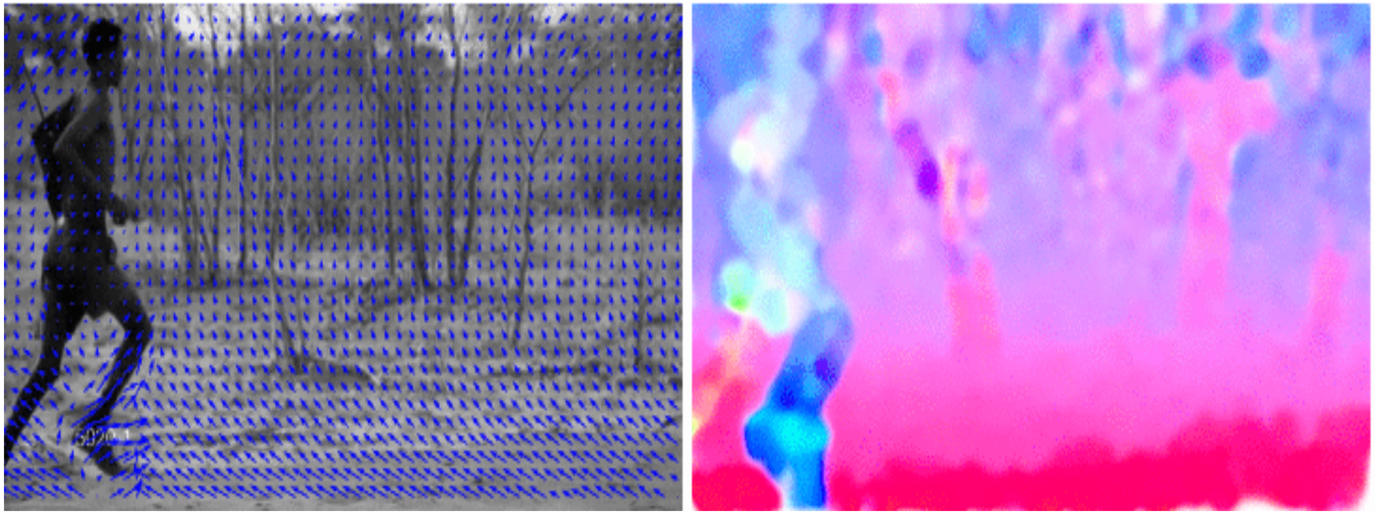
Followed by Bag-of-Words/Fisher vector and SVM



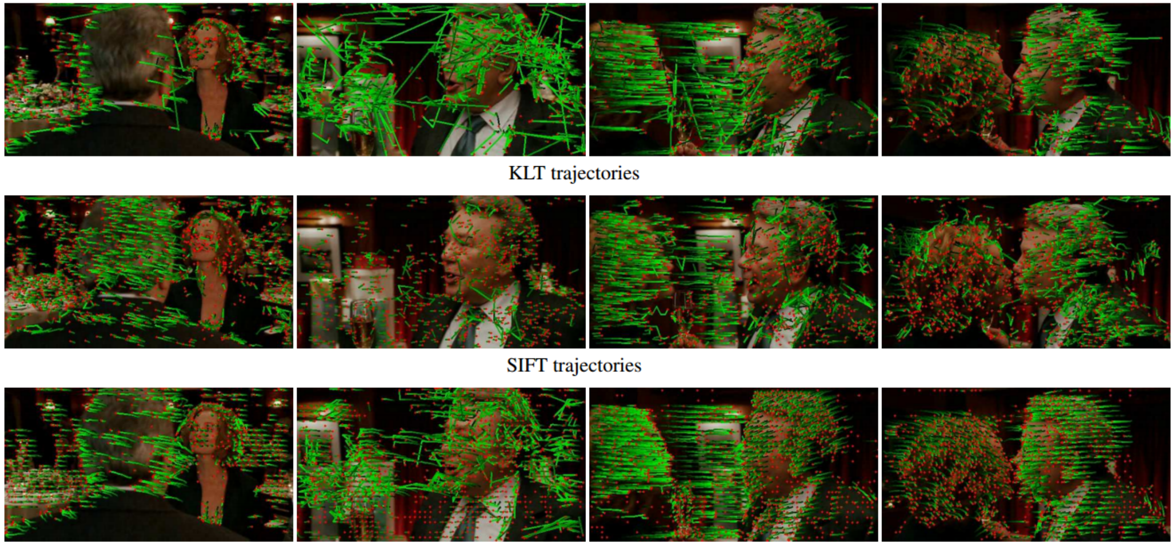


Motion is salient

Motion offers crucial clue where to attend in video



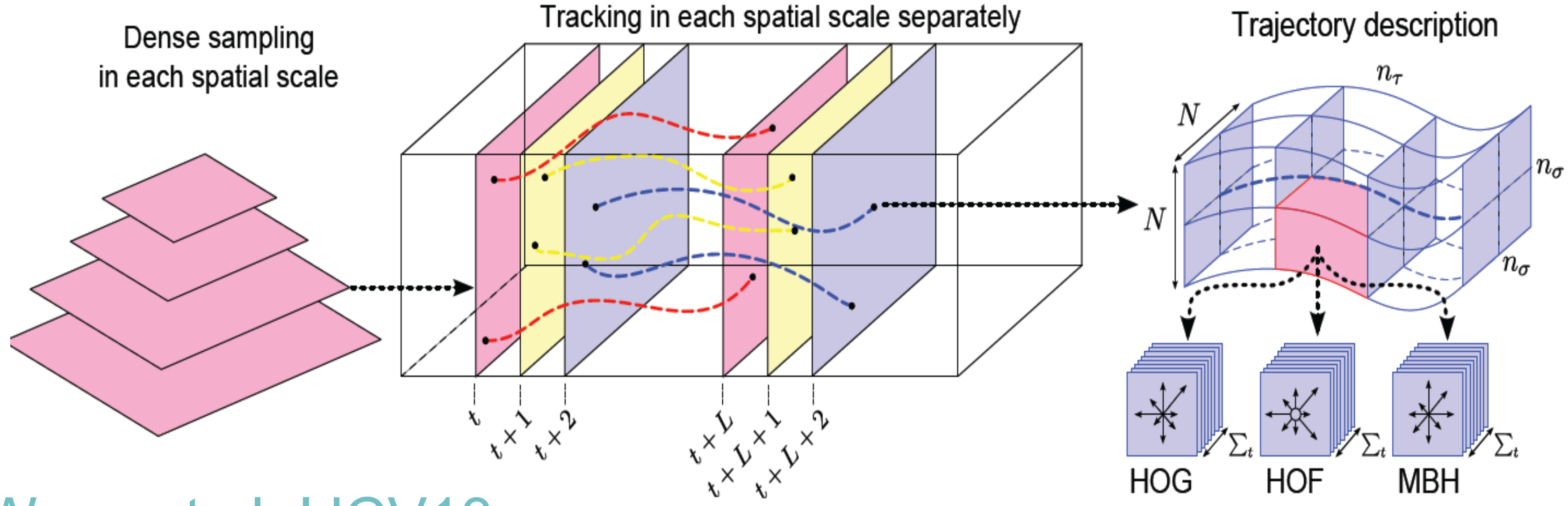
Flow trajectory descriptors



KLT trajectories

SIFT trajectories

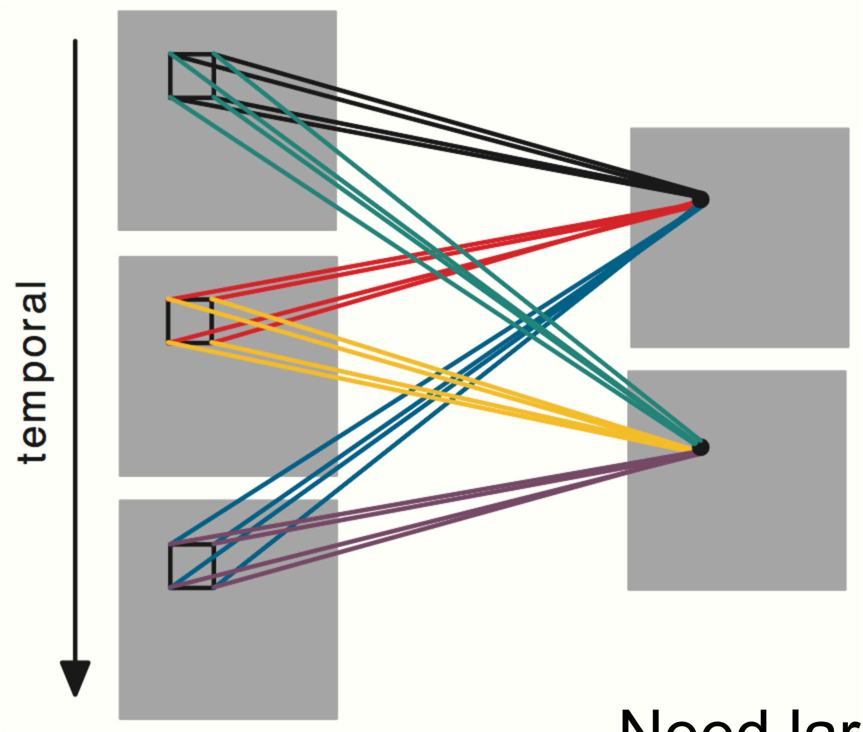
Dense trajectories



Wang et al. IJCV13

3D Convolutions

Extracts multiple features from both spatial and temporal dimensions by performing 3D convolutions



Need large amounts of data to learn filters

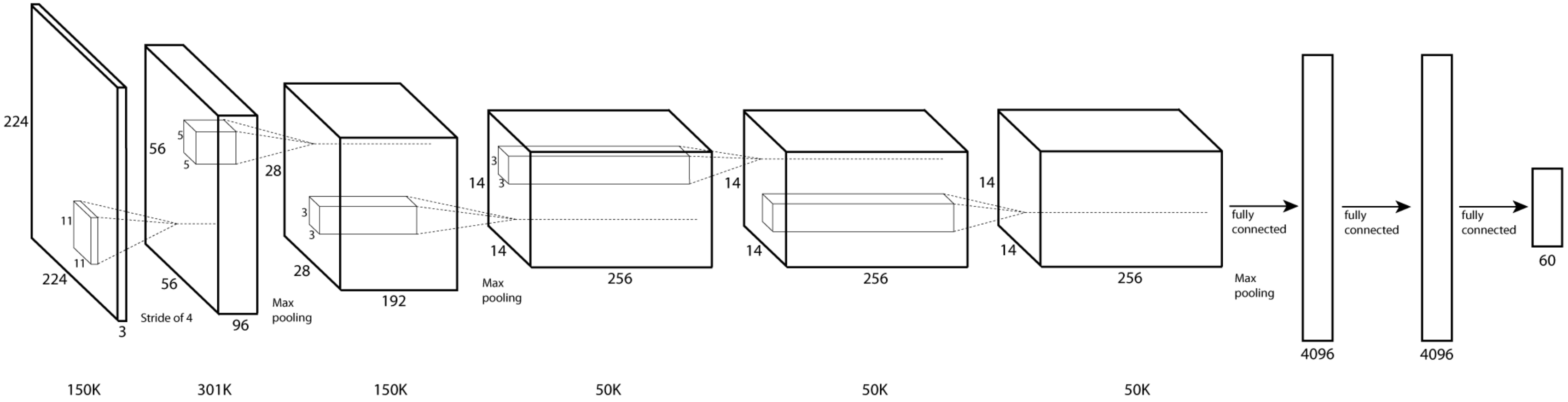
Ji et al. ICML10

Appearance only

Pre-train on ImageNet, fine-tune on video concepts

Average pooling over multiple frames per video shot

Good for objects and scenes, so and so for actions

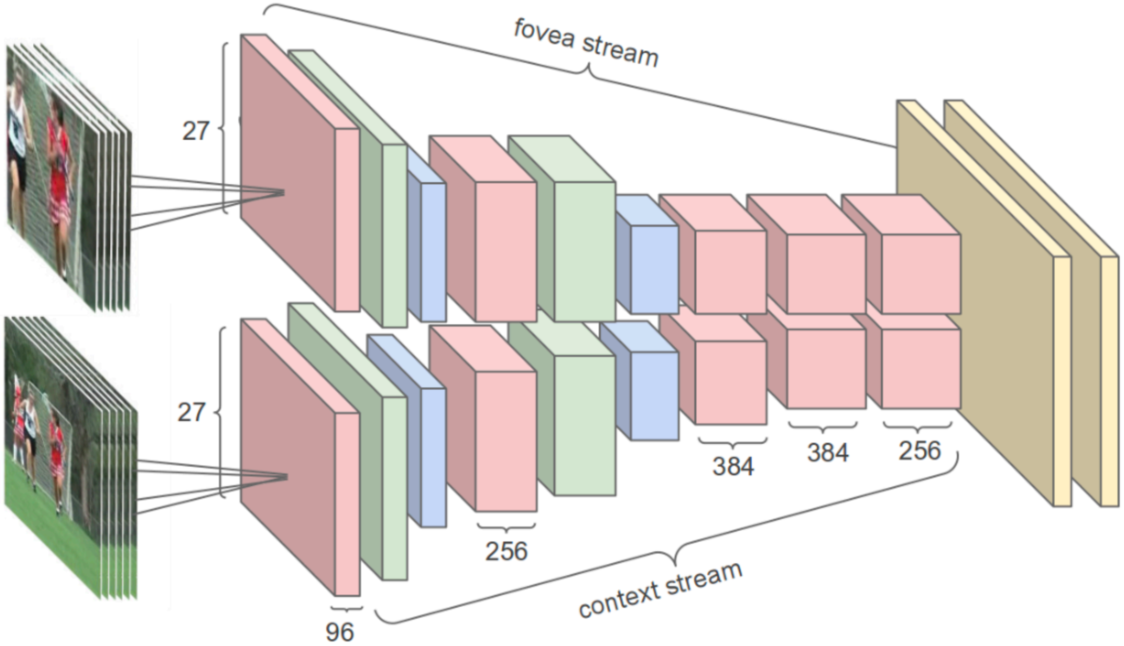


Snoek et al. TRECVID13

Appearance and spatial

Models spatial extent of action mildly by separating stream for center crop and entire frame

Considers various temporal pooling schemes



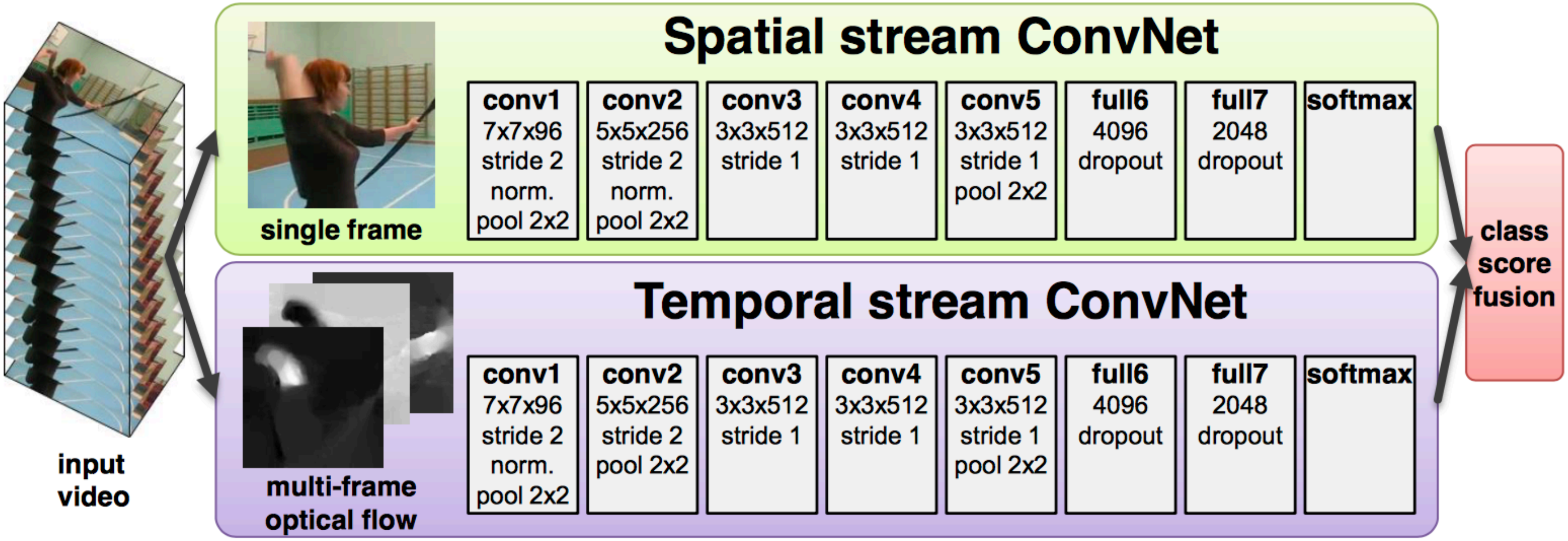
Introduces Sports1M dataset

Karpathy & Fei-Fei. CVPR14

Appearance and motion

Learn spatial and temporal filters separately

- Pre-train on ImageNet, fine-tune for actions
- Fusion by averaging or SVM



Spatial stream ConvNet

single frame

conv1	conv2	conv3	conv4	conv5	full6	full7	softmax
7x7x96 stride 2 norm. pool 2x2	5x5x256 stride 2 norm. pool 2x2	3x3x512 stride 1	3x3x512 stride 1	3x3x512 stride 1 pool 2x2	4096 dropout	2048 dropout	

Temporal stream ConvNet

multi-frame optical flow

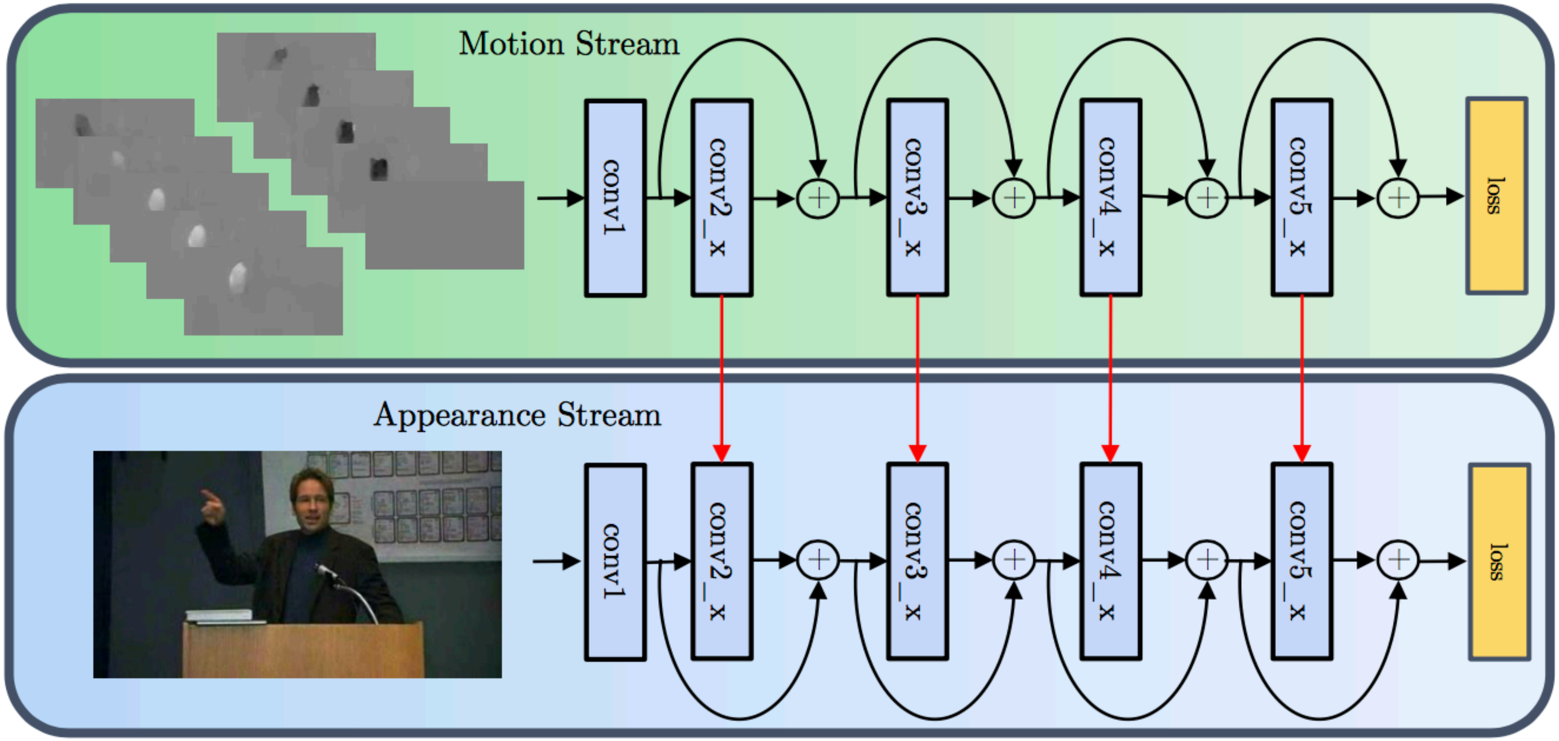
conv1	conv2	conv3	conv4	conv5	full6	full7	softmax
7x7x96 stride 2 norm. pool 2x2	5x5x256 stride 2 pool 2x2	3x3x512 stride 1	3x3x512 stride 1	3x3x512 stride 1 pool 2x2	4096 dropout	2048 dropout	

Simonyan & Zisserman, NIPS14

Appearance, motion and spatial

Generalize ResNet for video

Remove fully-connected layers

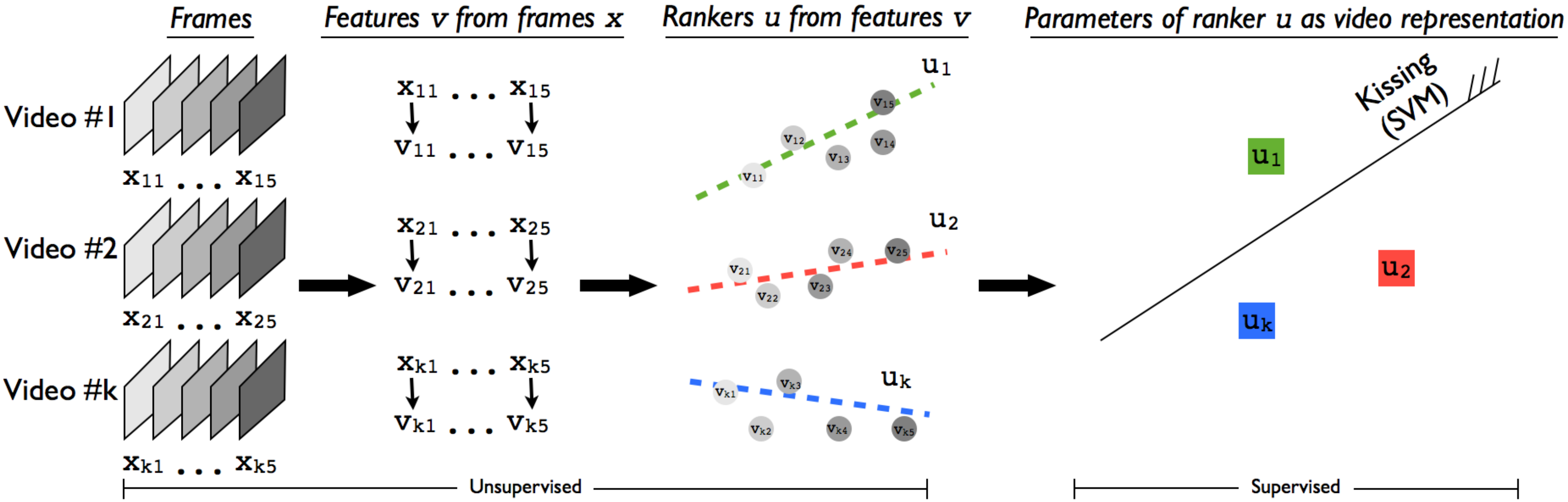


Feichtenhofer et al, NIPS16

Appearance, motion, and temporal

Key insight: exploit temporal order as soft label

Actions vary in appearance but order is preserved



Video-specific ranker parameters as representation

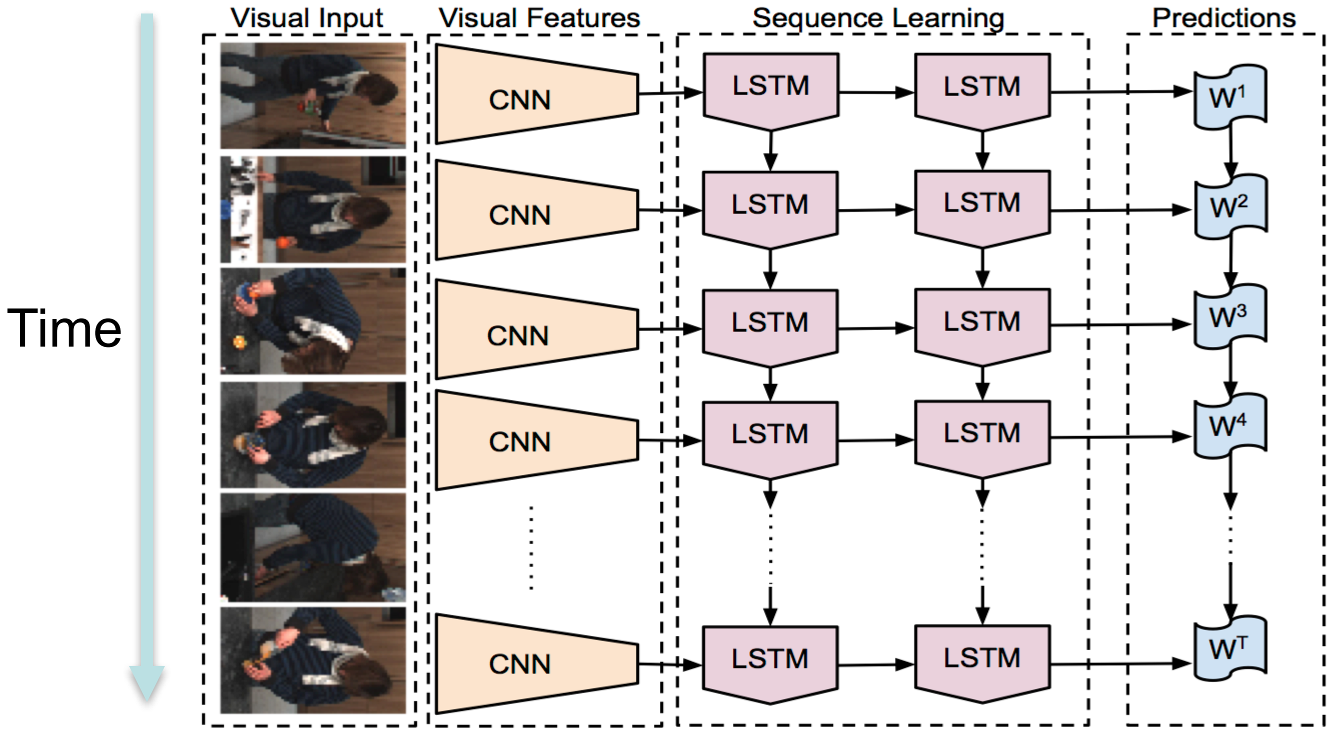
Similar actions → similar ranking parameters

Fernando et al. CVPR15

Appearance and temporal

LSTM models sequential memories in the long and short term

ConvNet-fc vectors as input, no spatial information encoded

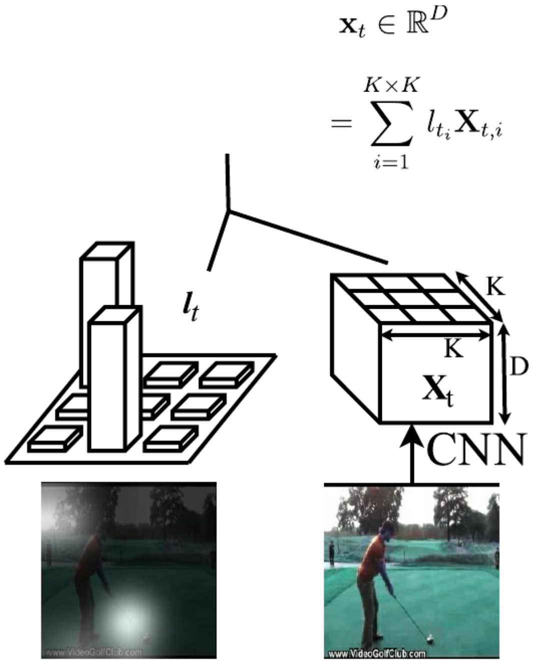


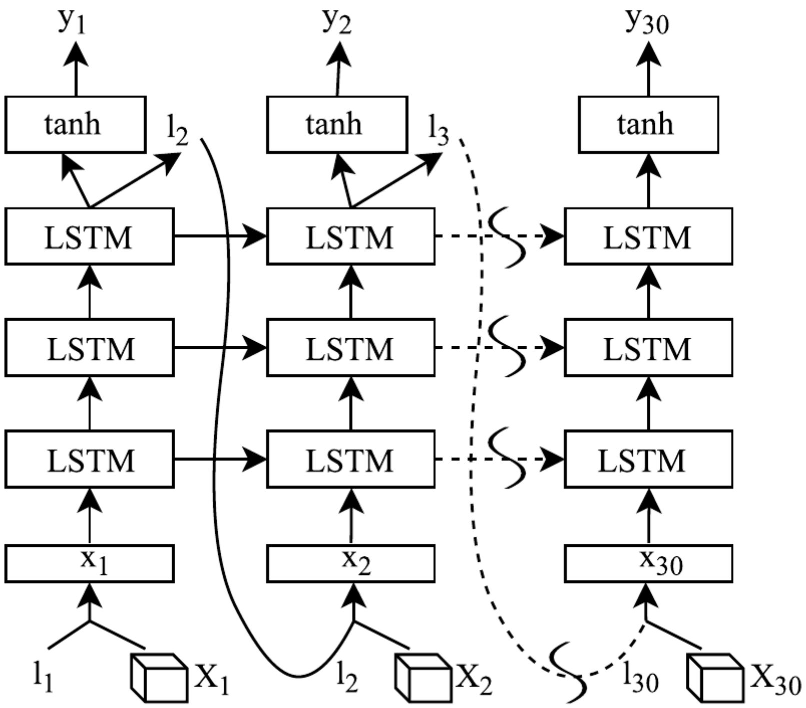
Baccouche et al. ICANN10 / Donahue et al. CVPR15 / Ng et al. CVPR15

Appearance, temporal, spatial

Look for best locations leading to correct action classification

- Stays close to soft-Attention for image captioning [Xu et al. ICML15],
- Vectorizes attention and appearance, ignores the motion inside a video.


$\mathbf{x}_t \in \mathbb{R}^D$
$= \sum_{i=1}^{K\times K} l_{t_i} \mathbf{X}_{t,i}$
l_t
K
K
D
$\mathbf{X}_t$
CNN
www.VideoGolfClub.com
www.VideoGolfClub.com

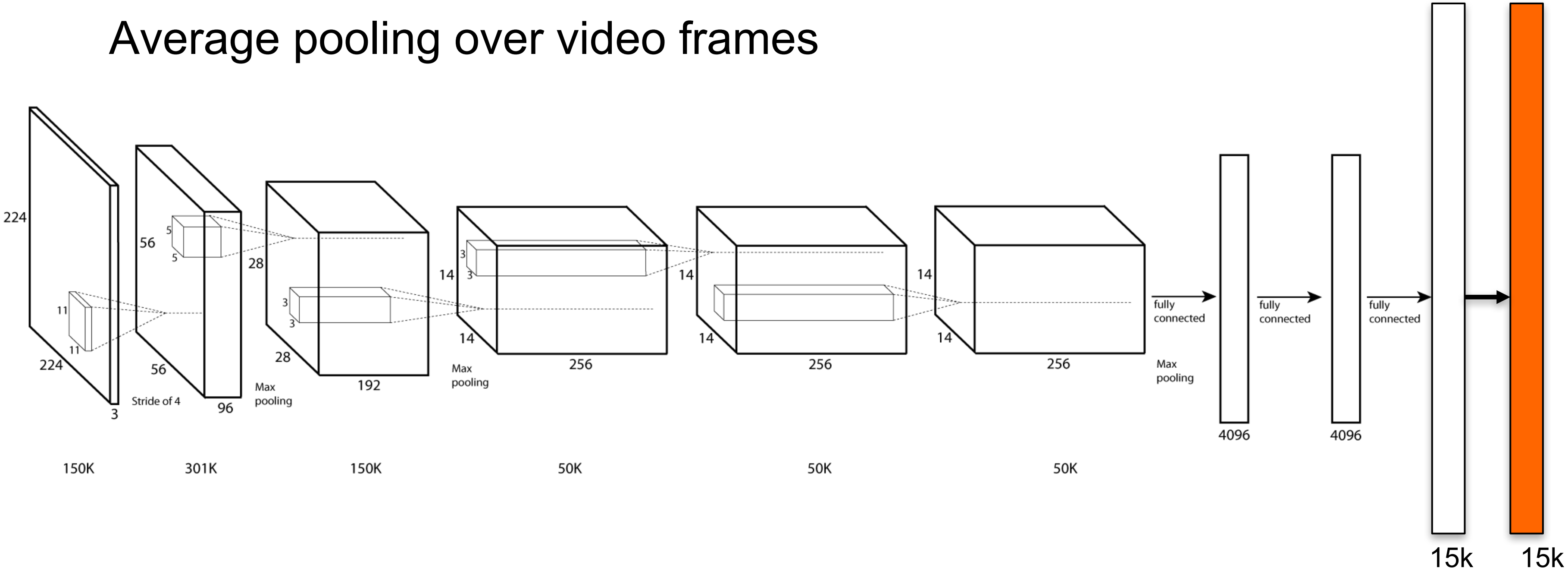


Sharma et al. NIPS15 / ICLR16

Encoding video by 15,000 objects

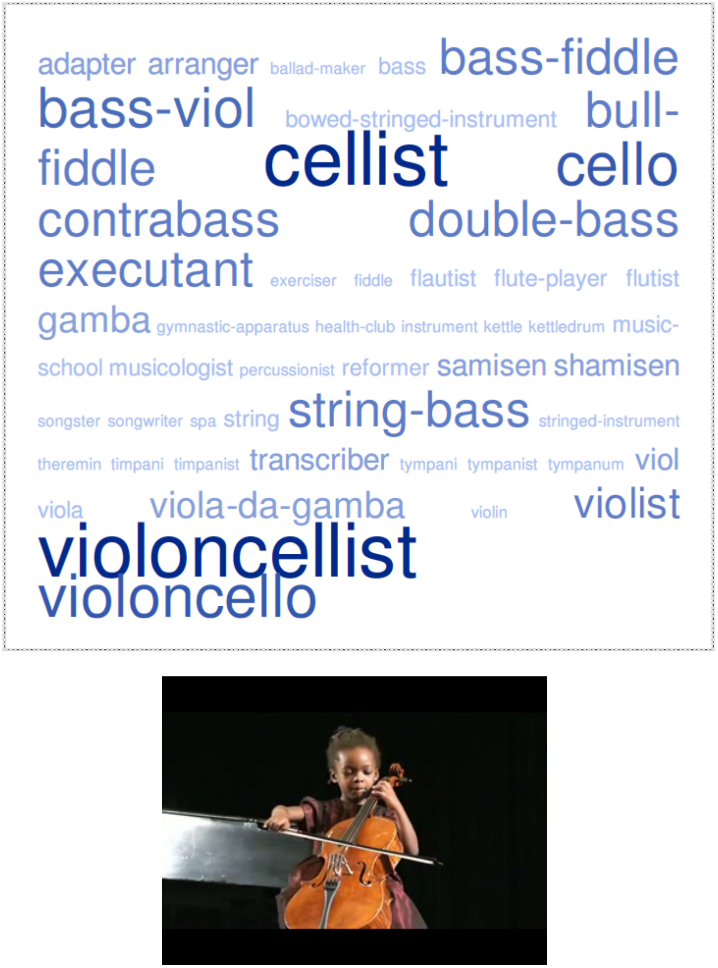
Krizhevsky-style cuda-convnet with dropout [NIPS12]

- Convolutional neural network with 8 layers with weights
- Trained using error back propagation
- Learns from annotations for 15,000 ImageNet object categories
- Average pooling over video frames

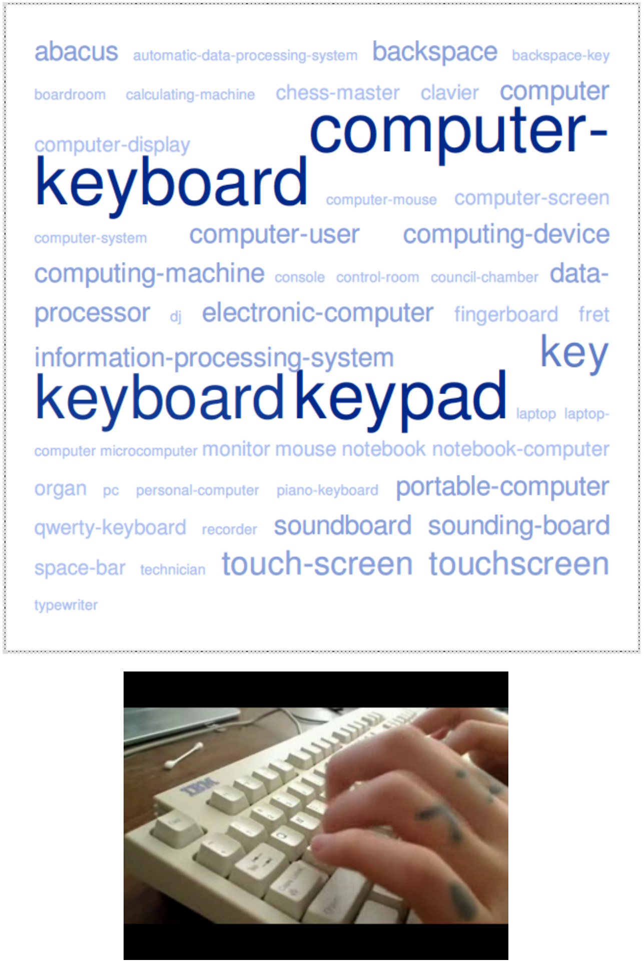


Jain et al. CVPR15

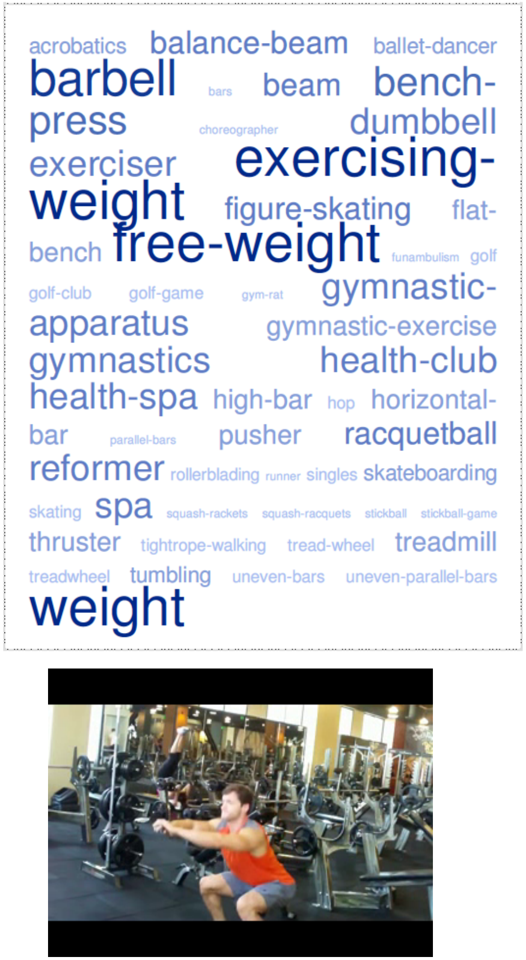
What objects emerge in actions?


adapter arranger ballad-maker bass bass-fiddle
bass-viol bowed-stringed-instrument bull-fiddle
cellist cello
contrabass double-bass
executant exerciser fiddle flautist flute-player flutist
gamba gymnastic-apparatus health-club instrument kettle kettledrum music-school musicologist percussionist reformer samisen shamisen
songster songwriter spa string string-bass stringed-instrument
theremin timpani timpanist transcriber tympani tympanist tympanum viol
viola viola-da-gamba violin violist
violoncellist
violoncello

Playing Cello

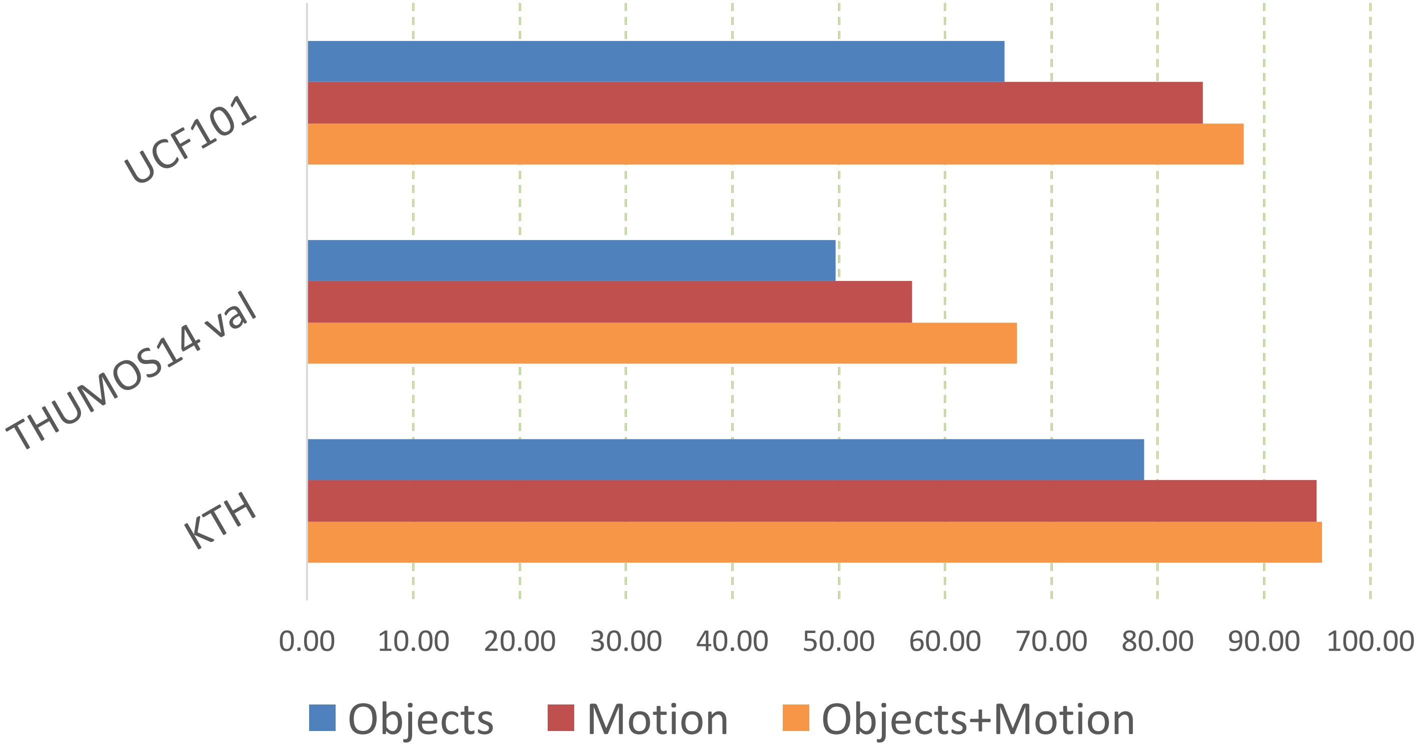

abacus automatic-data-processing-system backspace backspace-key
boardroom calculating-machine chess-master clavier computer
computer-display computer-keyboard
computer-mouse computer-screen
computer-system computer-user computing-device
computing-machine console control-room council-chamber data-processor dj electronic-computer fingerboard fret
information-processing-system key
keyboard keypad laptop laptop-computer microcomputer monitor mouse notebook notebook-computer
organ pc personal-computer piano-keyboard portable-computer
qwerty-keyboard recorder soundboard sounding-board
space-bar technician touch-screen touchscreen
typewriter

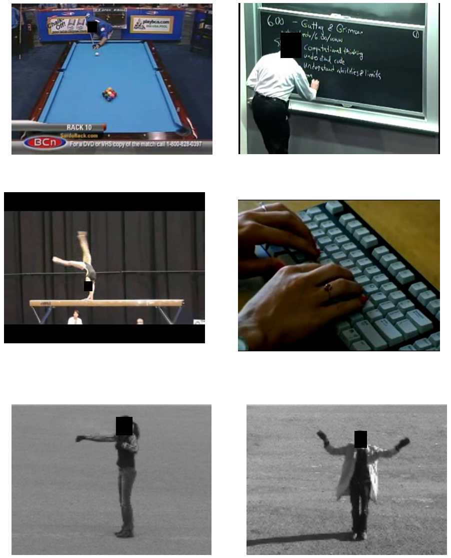
Typing


acrobatics balance-beam ballet-dancer
barbell bars beam bench-press choreographer dumbbell
exerciser exercising-weight figure-skating flat-bench free-weight funambulism golf
golf-club golf-game gym-rat gymnastic-apparatus gymnastic-exercise
gymnastics health-club
health-spa high-bar hop horizontal-bar parallel-bars pusher racquetball
reformer rollerblading runner singles skateboarding
skating spa squash-rackets squash-racquets stickball stickball-game
thruster tightrope-walking tread-wheel treadmill
treadwheel tumbling uneven-bars uneven-parallel-bars
weight

Bodyweight squats

Objects and motion




[redacted]
[redacted]
[redacted]
[redacted]
[redacted]

Objects combined with motion improve accuracy

Jain et al. CVPR15

Motion-reliant actions?

Wall Pushups

Tai Chi

Jumping Jack

Hula Hoop

Jump Rope

Trampoline Jumping

Lunges

Uneven Bars

Pull Ups

Military Parade

Bodyweight Squats

Boxing Speed Bag

Object-related actions

Playing Piano

Billiards

Baseball Pitch

Breast Stroke

Head Massage

Mixing

Soccer Penalty

Frisbee Catch

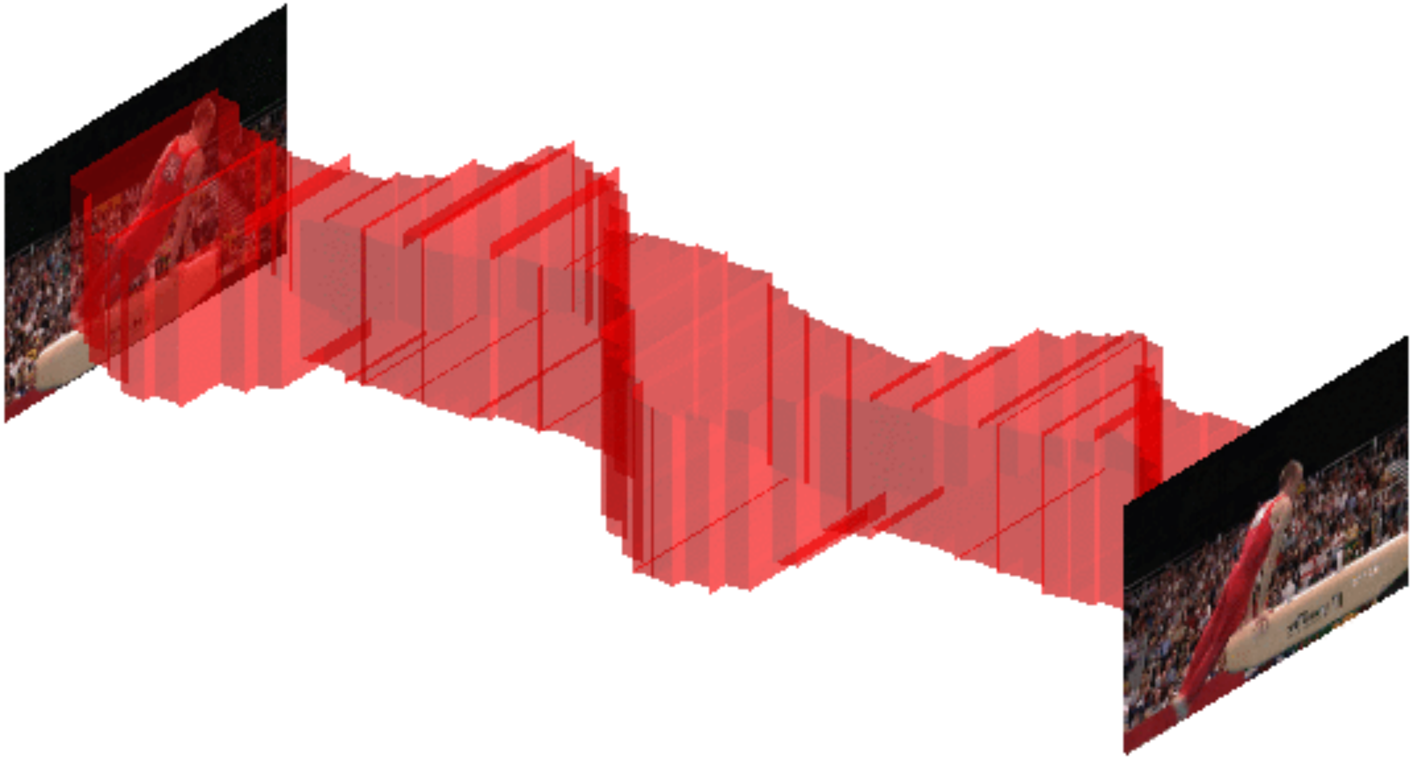
Rock Climbing Indoor

Archery

Cutting in Kitchen

Sumo Wrestling

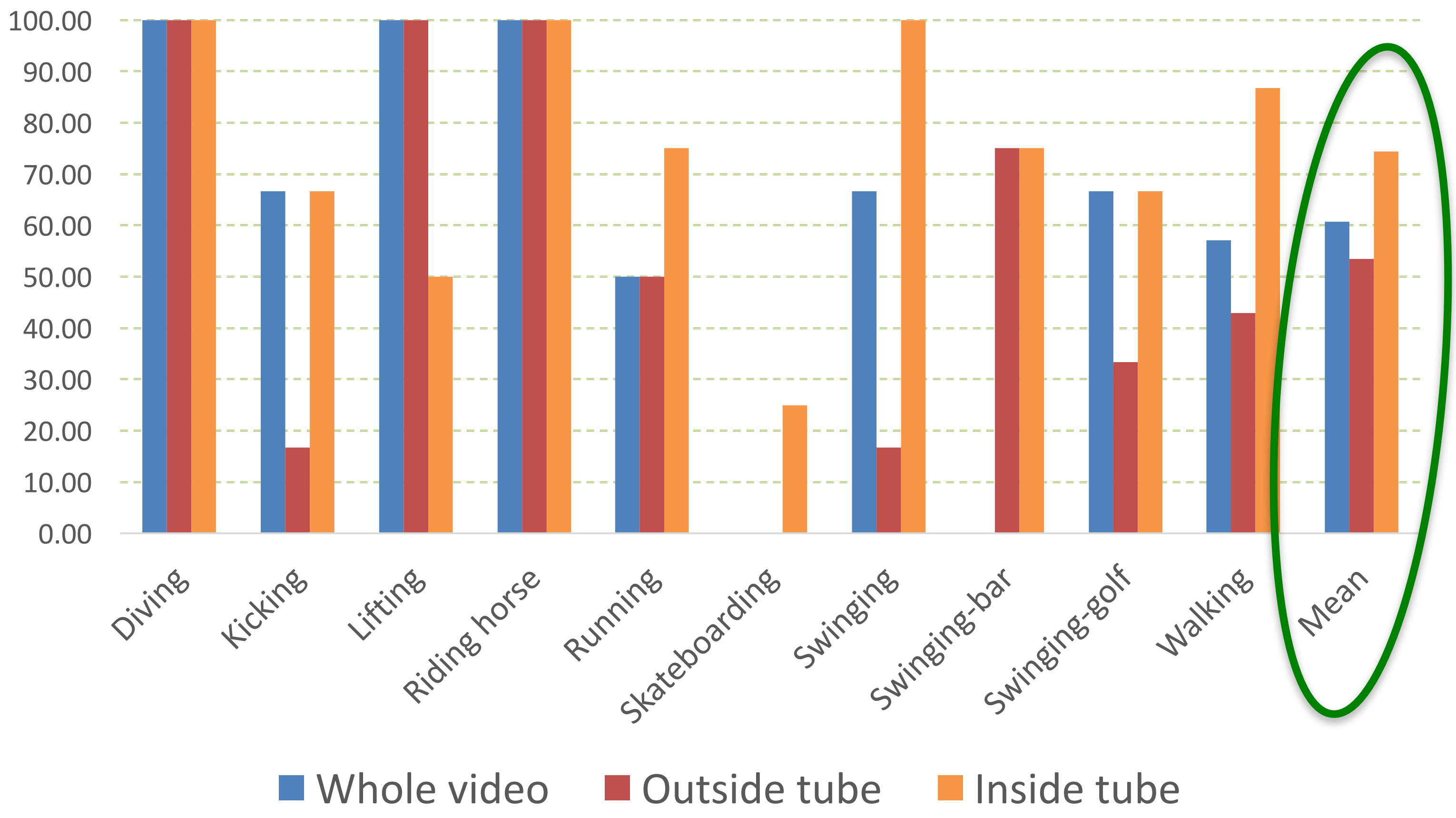
Where do objects aid most?



We consider three encodings

- Whole video
- Outside tube
- Inside tube

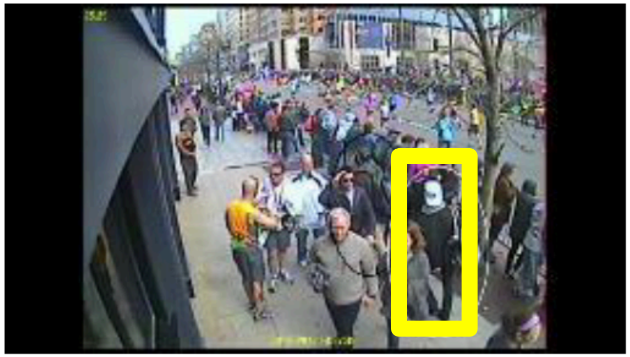
Objects, motion, spatial

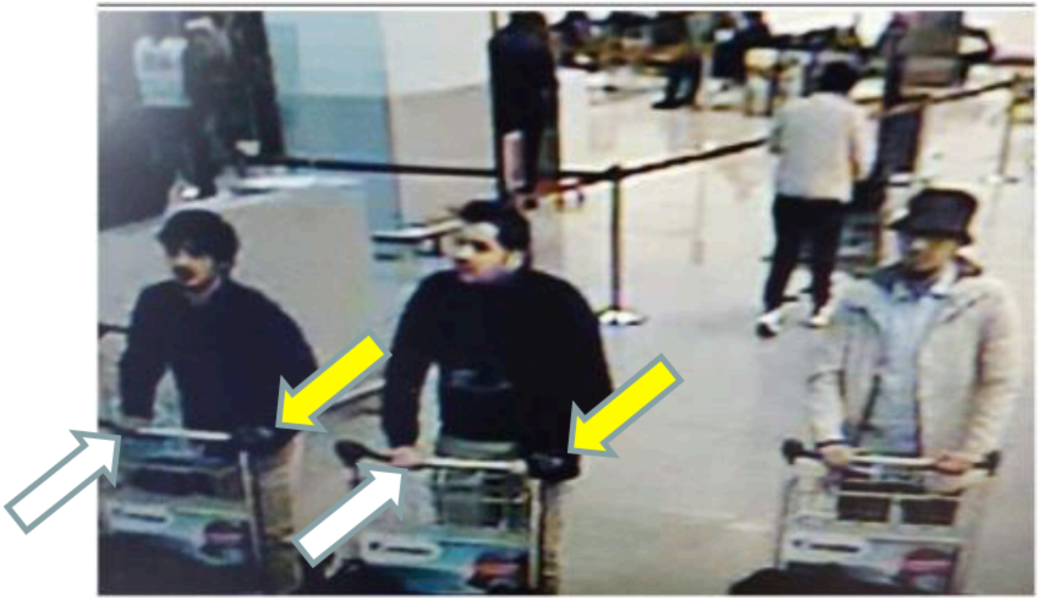


Objects aid most close to and involved in the action

Jain et al. CVPR15

Quiz: How to derive intention?





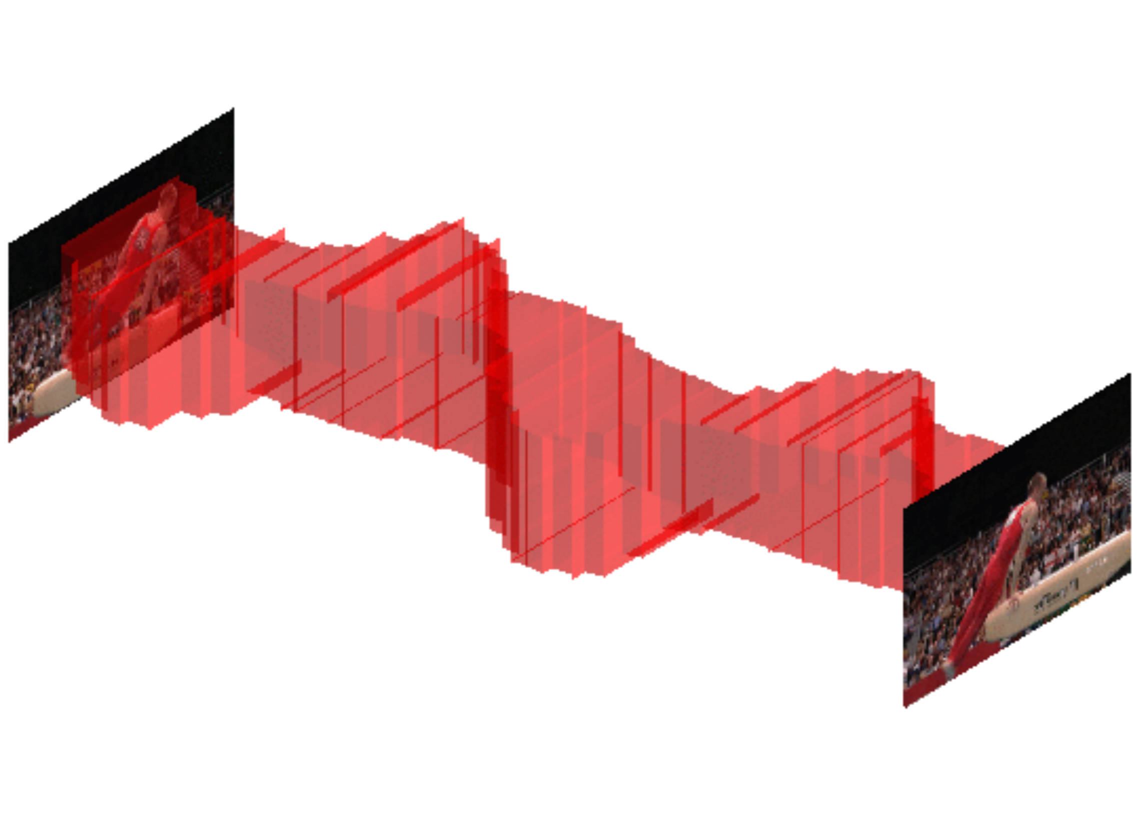
Summary

	Actor	Appearance	Motion	Objects	Spatial	Temporal
Ji et al.		+				
Snoek et al.		+				
Karpathy		+			+/-	
Simonyan		+	+			
Feichtenhofer		+	+		+	
Fernando		+	+			+
Donahue / Ng		+				+
Sharma		+			+	+
Jain			+	+	+	

Over-concentration on appearance, partly caused by dataset bias. Actor (and her pose) mostly ignored.

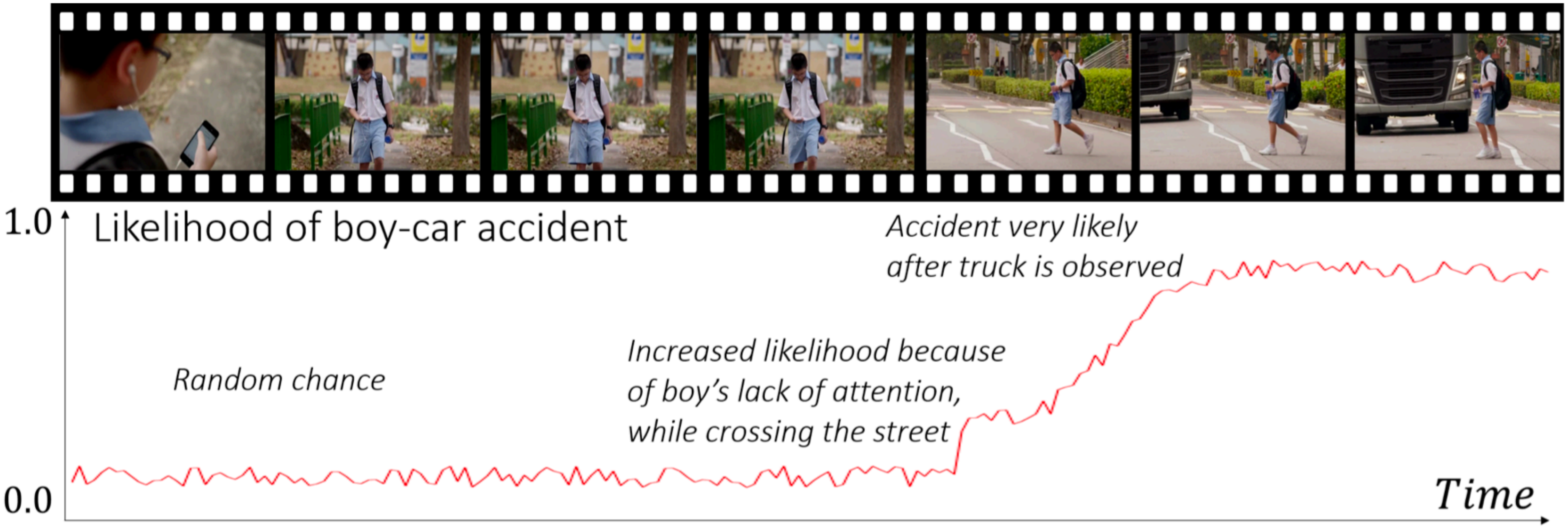
5. Spatiotemporal localization

In this chapter we consider how to determine the spatiotemporal extent of an action with the use of dedicated detectors and unsupervised proposals.



Temporal-only localization

Determine the start and end of an action interval

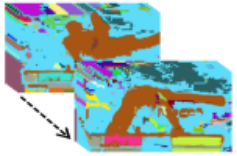


De Geest et al. ECCV16

Action proposals

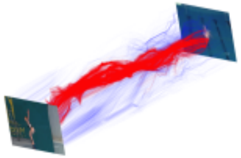
Supervoxels

Jain et al. *CVPR'14*
Oneata et al. *ECCV'14*



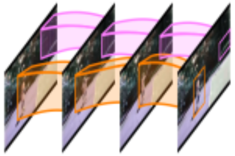
Trajectories

van Gemert et al. *BMVC'15*
Puskas et al. *ICCV'15*

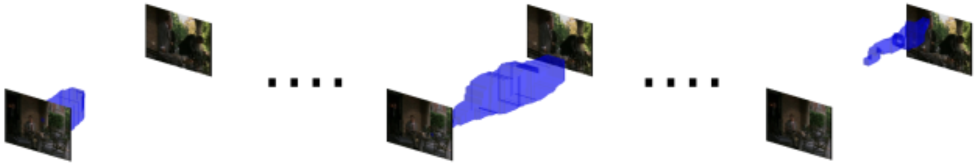


Detect & Track

Yu et al. *CVPR'15*
Weinzaepfel et al. *ICCV'15*



Action proposals



Action localization with proposals

At train time

Annotate spatiotemporal tubes with class labels
Extract video representation from tubes

Train favorite classifier

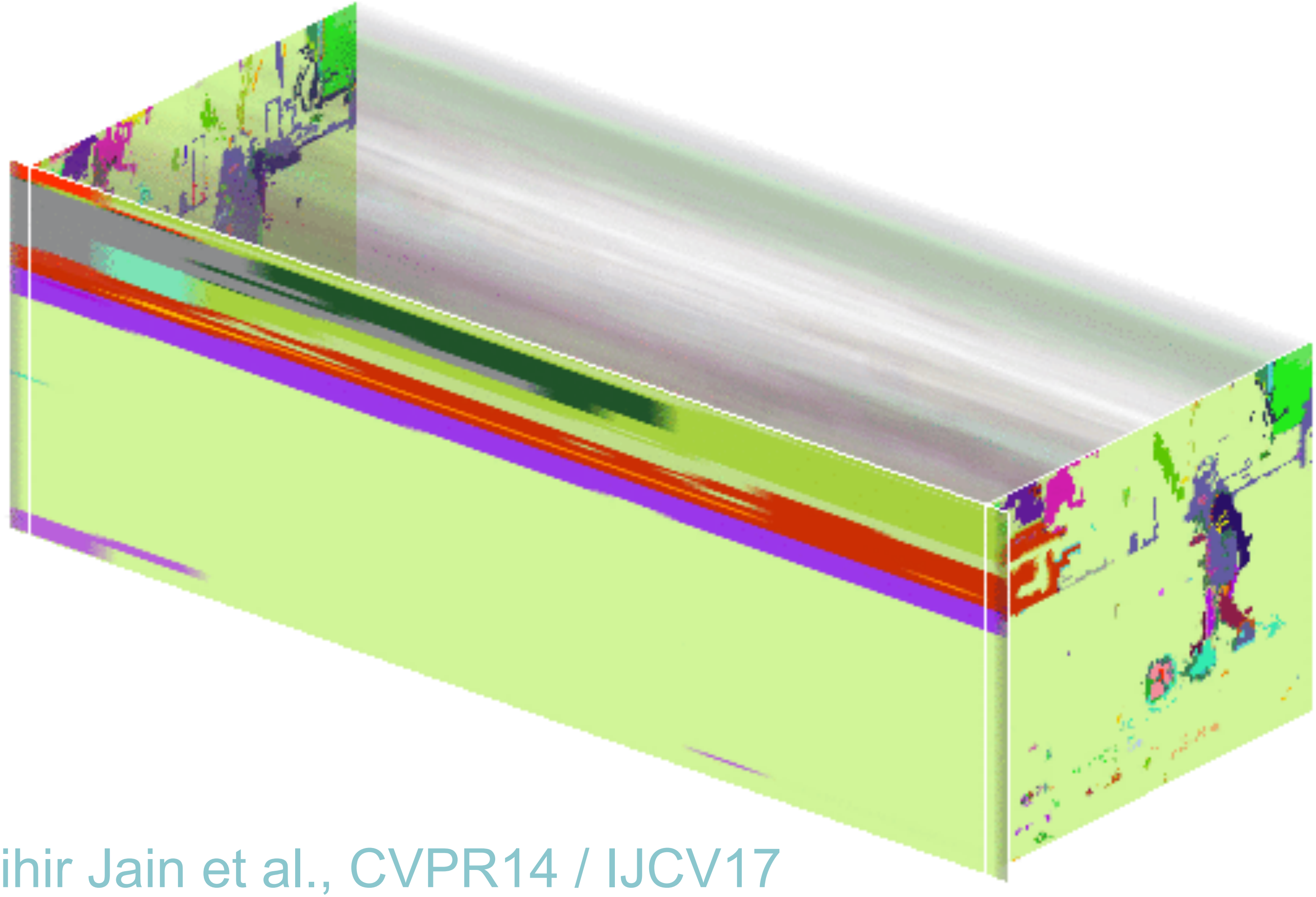
At test time

Extract action proposals

Extract video representation from each proposal

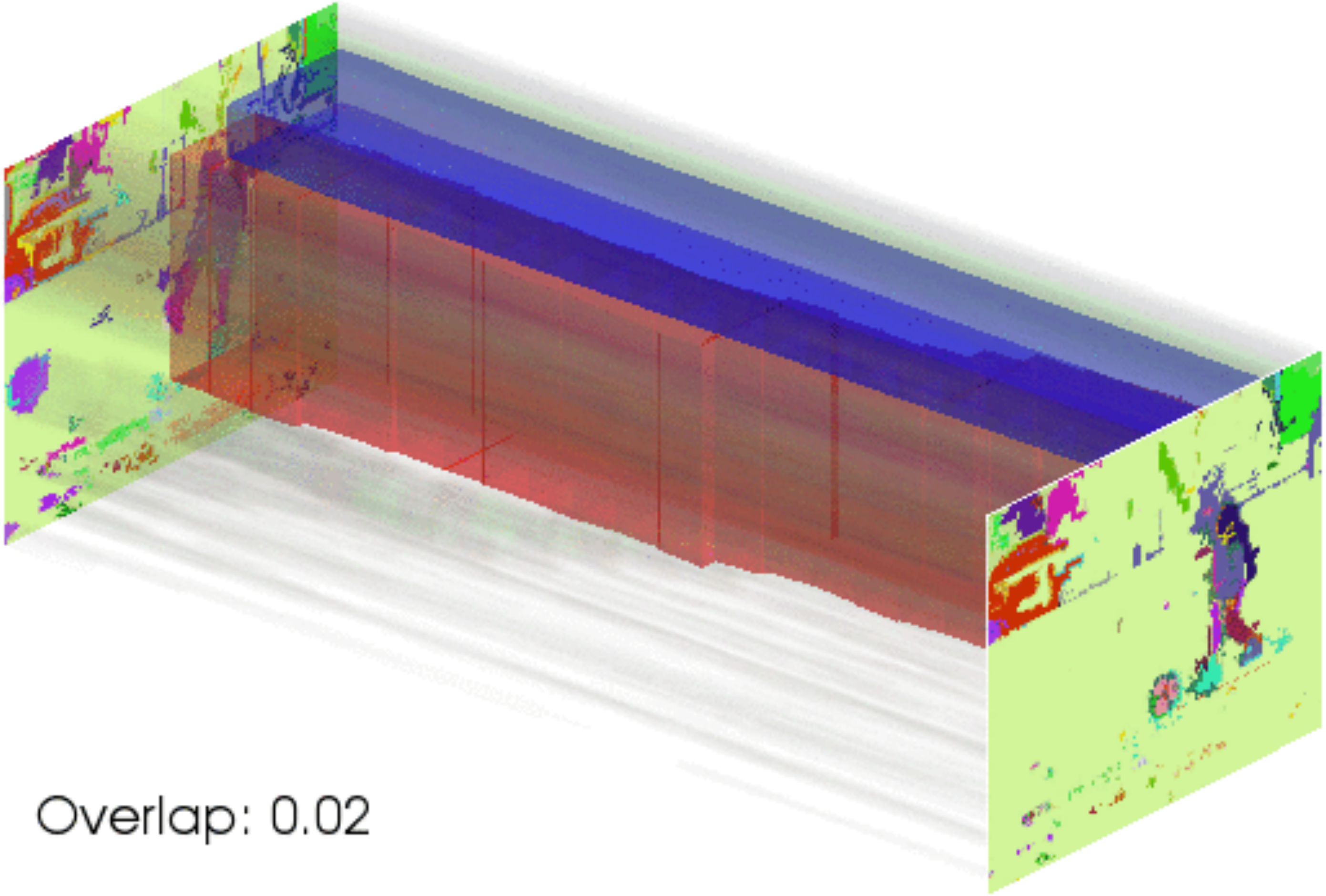
Classify all proposals, select proposal with maximum response

Proposals from supervoxels



Mihir Jain et al., CVPR14 / IJCV17

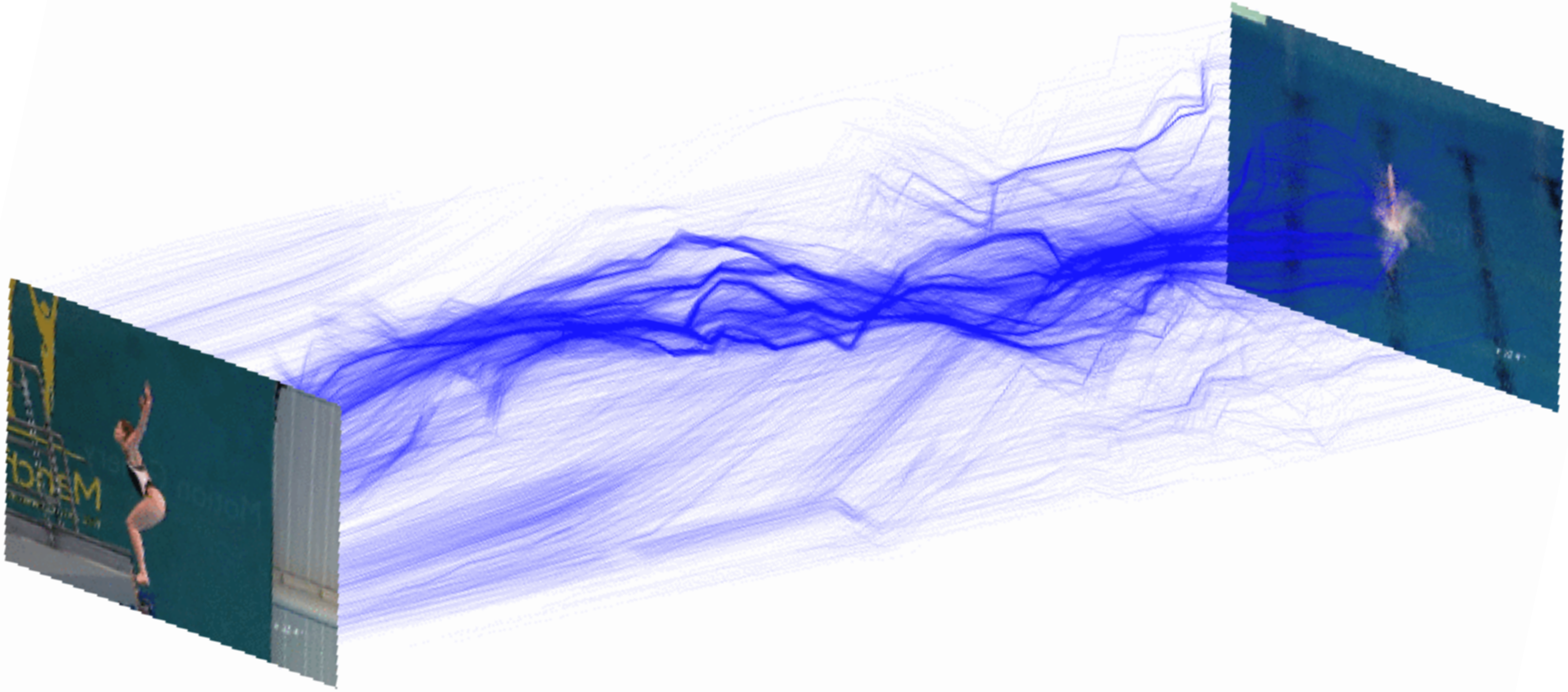
Proposals from supervoxels



Mihir Jain et al., CVPR14 / IJCV17

Proposals from trajectories

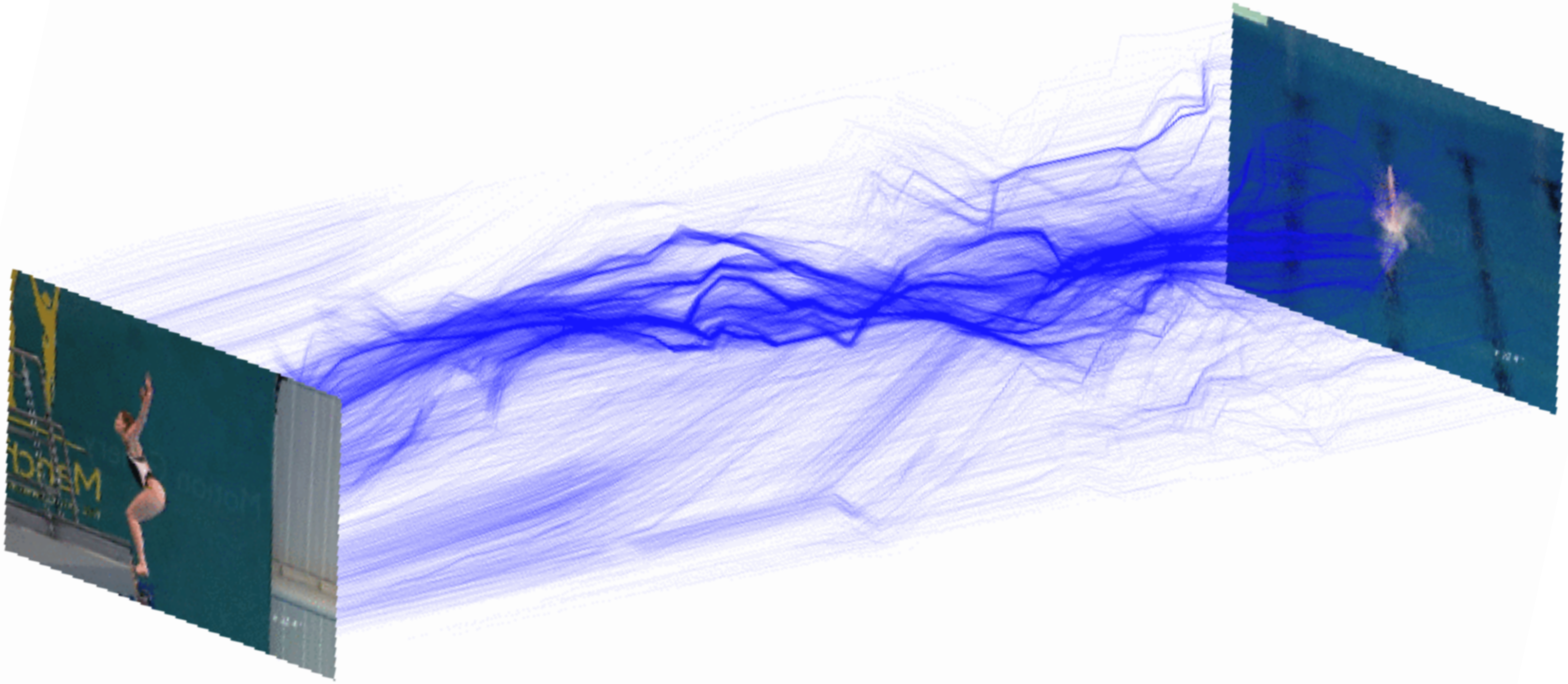
Proposals from video representation for action recognition



Van Gemert et al., BMVC 2015

Proposals from trajectories

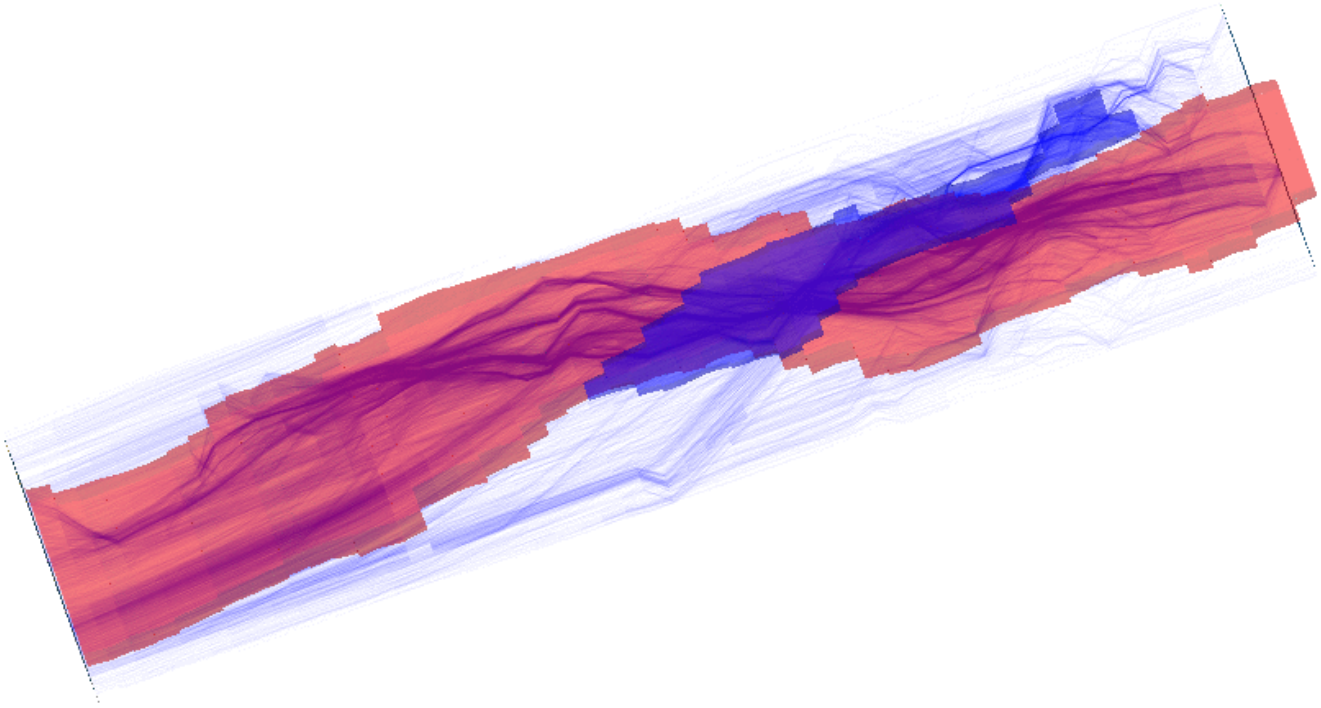
Basic idea: cluster improved dense trajectories



Van Gemert et al., BMVC 2015

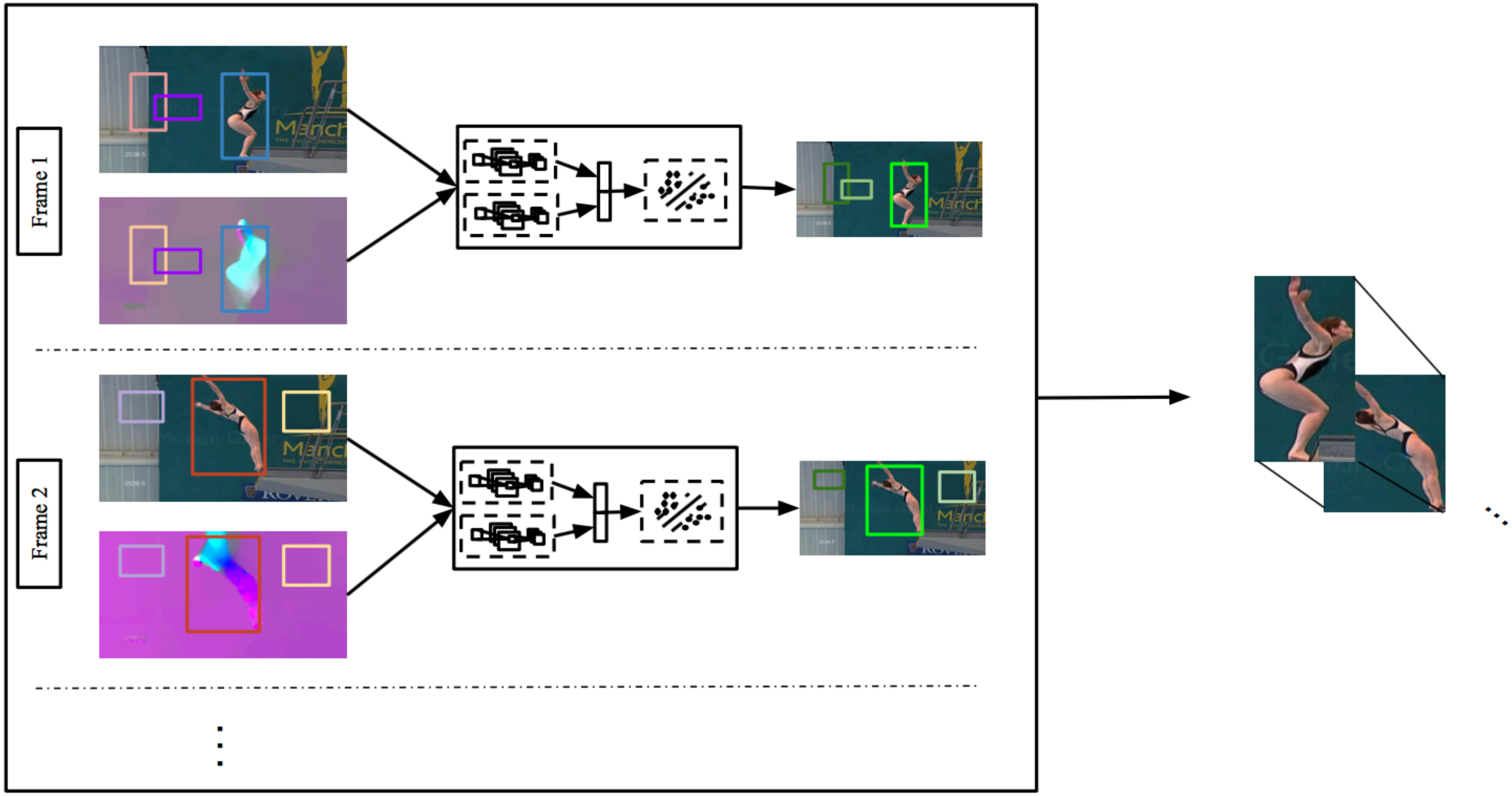
Proposals from trajectories

Group clusters into action proposals



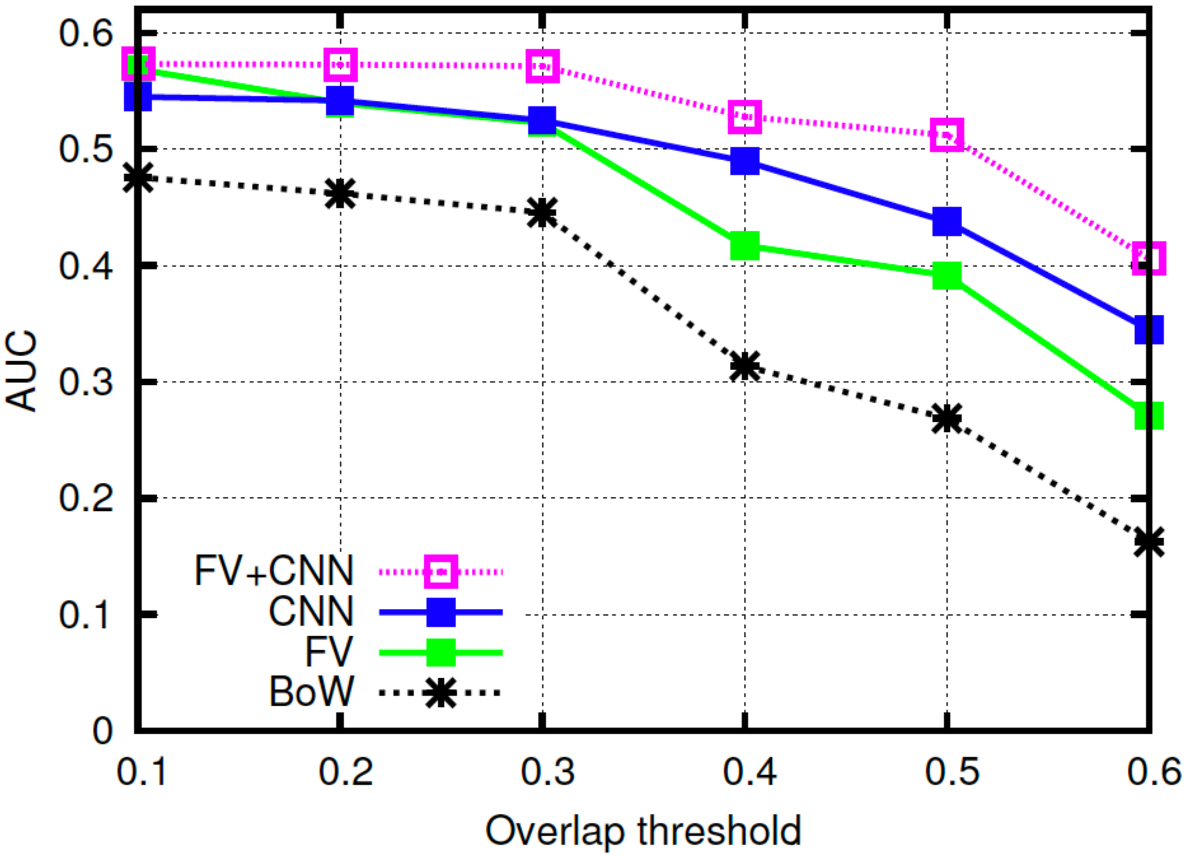
Van Gemert et al., BMVC 2015

Proposals by linking detections



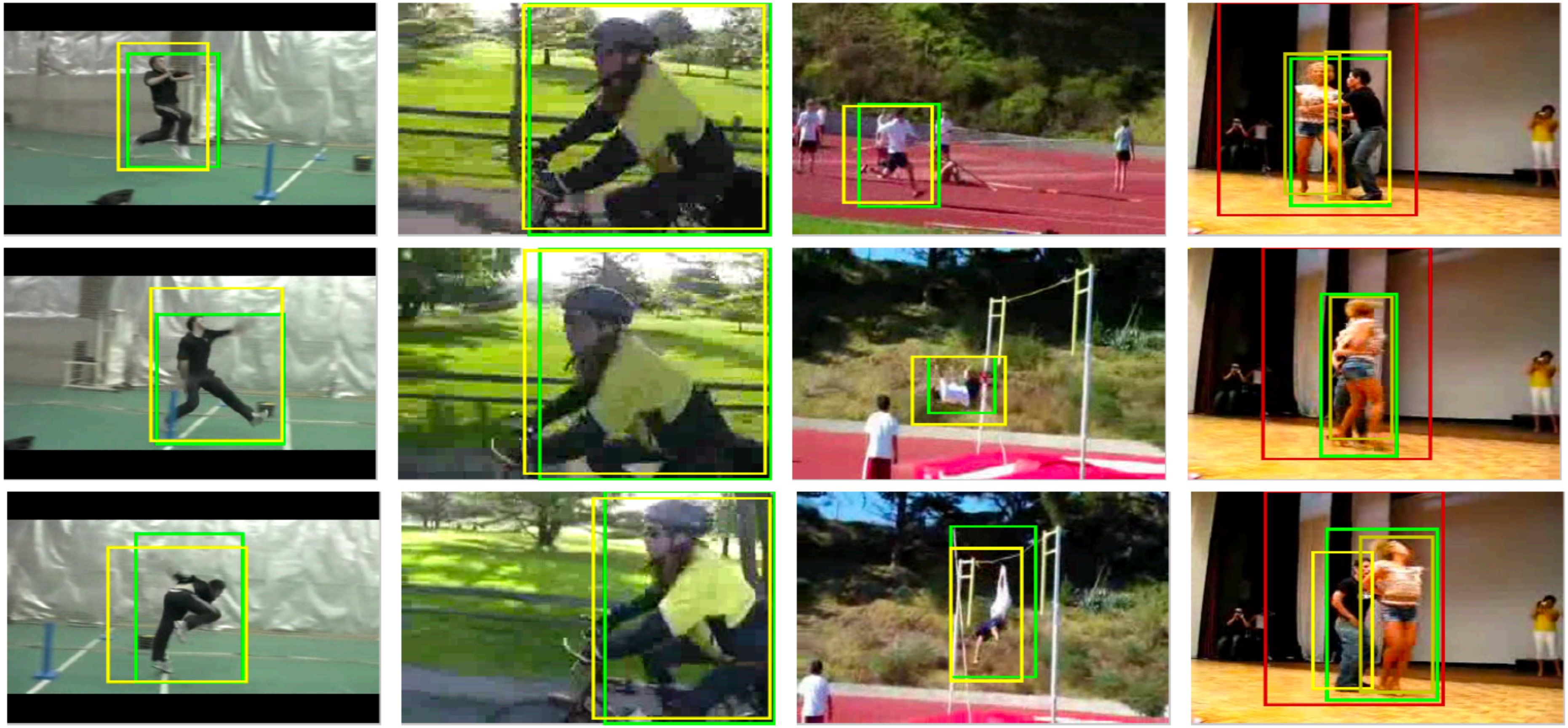
Gkioxari & Malik, CVPR 2015 / Weinzaepfel et al. ICCV 2015

Some results



Proposals profit from strong video representations

Qualitative results



(e) Cricket Bowling

(f) Biking

(g) Pole Vault

(h) Salsa Spin

Groundtruth

Good result

Bad result

Quiz: Limitations of proposals?

Quiz: Limitations of proposals?

Typically modeled as two-stage process

First extract proposals,
Then encode proposals with video representation

Not necessarily the same representation

Action proposals can be fast and effective but

Demand carefully labeled **box annotations** at train time.

6. 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

Li et al., arXive 1607.01794

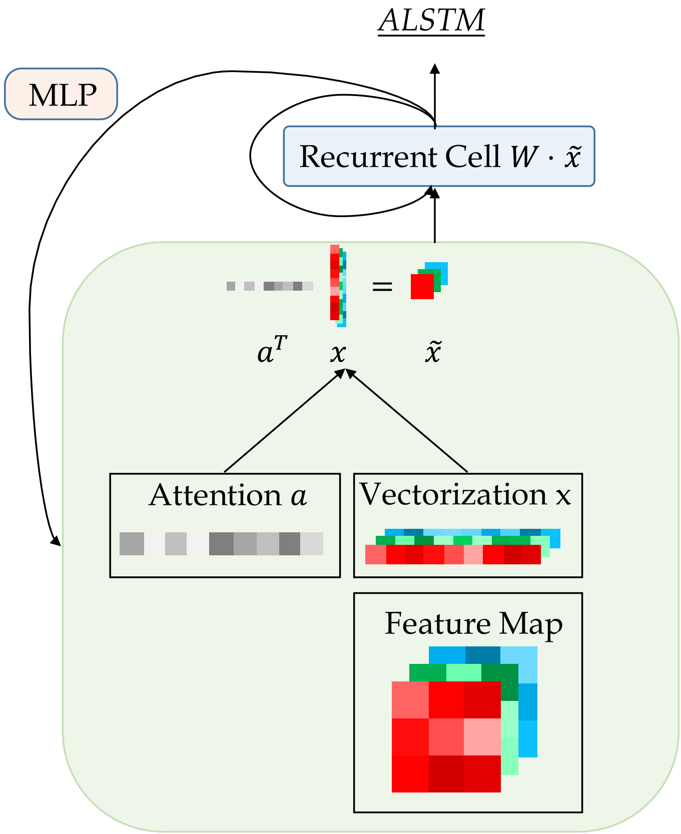
Convolutional (A)LSTM

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

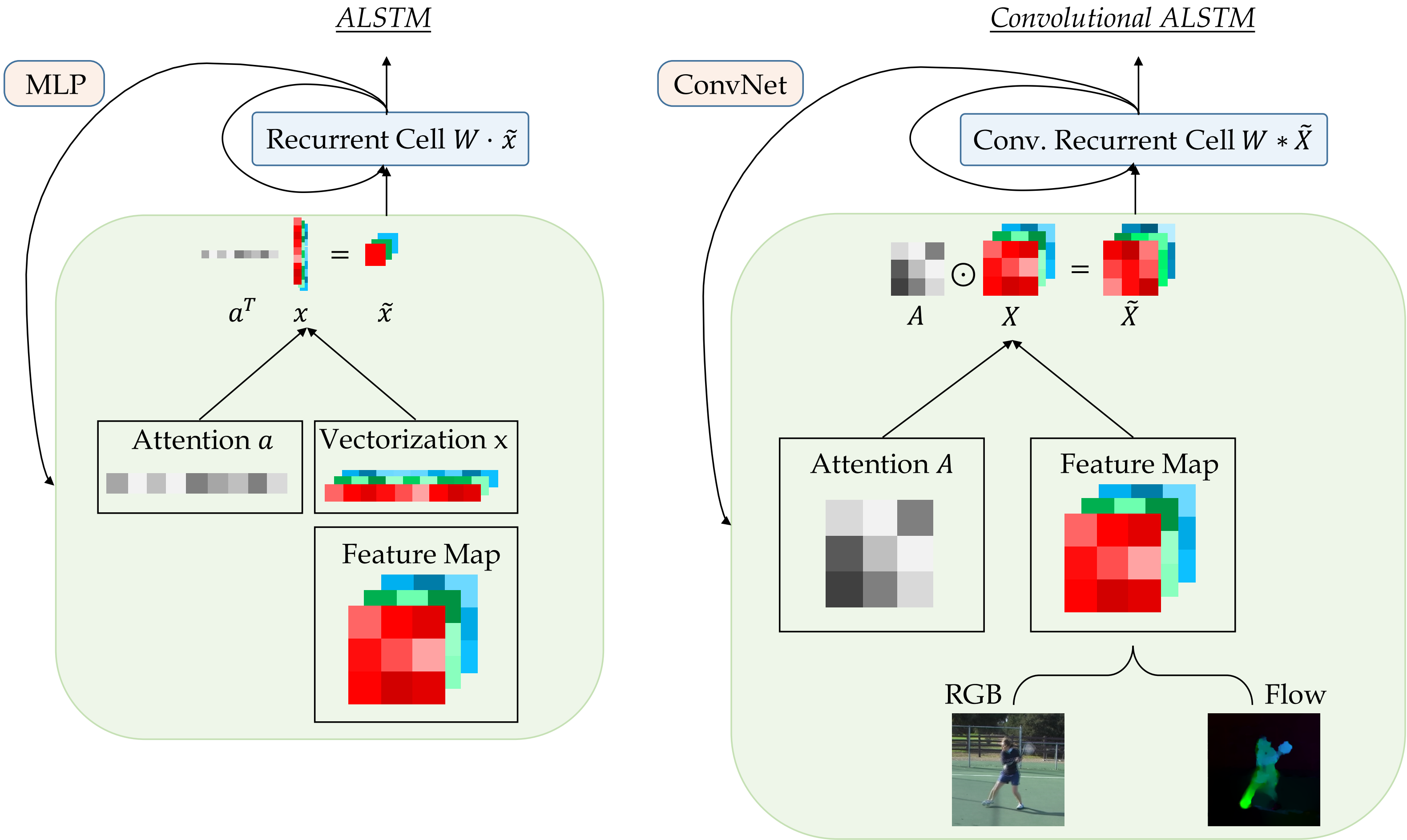
$$
\begin{aligned}
I_t &= \sigma(W_{xi} * \widetilde{X}_t + W_{hi} * H_{t-1} + b_i) \\
F_t &= \sigma(W_{xf} * \widetilde{X}_t + W_{hf} * H_{t-1} + b_f) \\
O_t &= \sigma(W_{xo} * \widetilde{X}_t + W_{ho} * H_{t-1} + b_o) \\
G_t &= \tanh(W_{xc} * \widetilde{X}_t + W_{hc} * H_{t-1} + b_c) \\
C_t &= F_t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

Generate attention by shallow ConvNet instead of MLP

A(ttention)LSTM

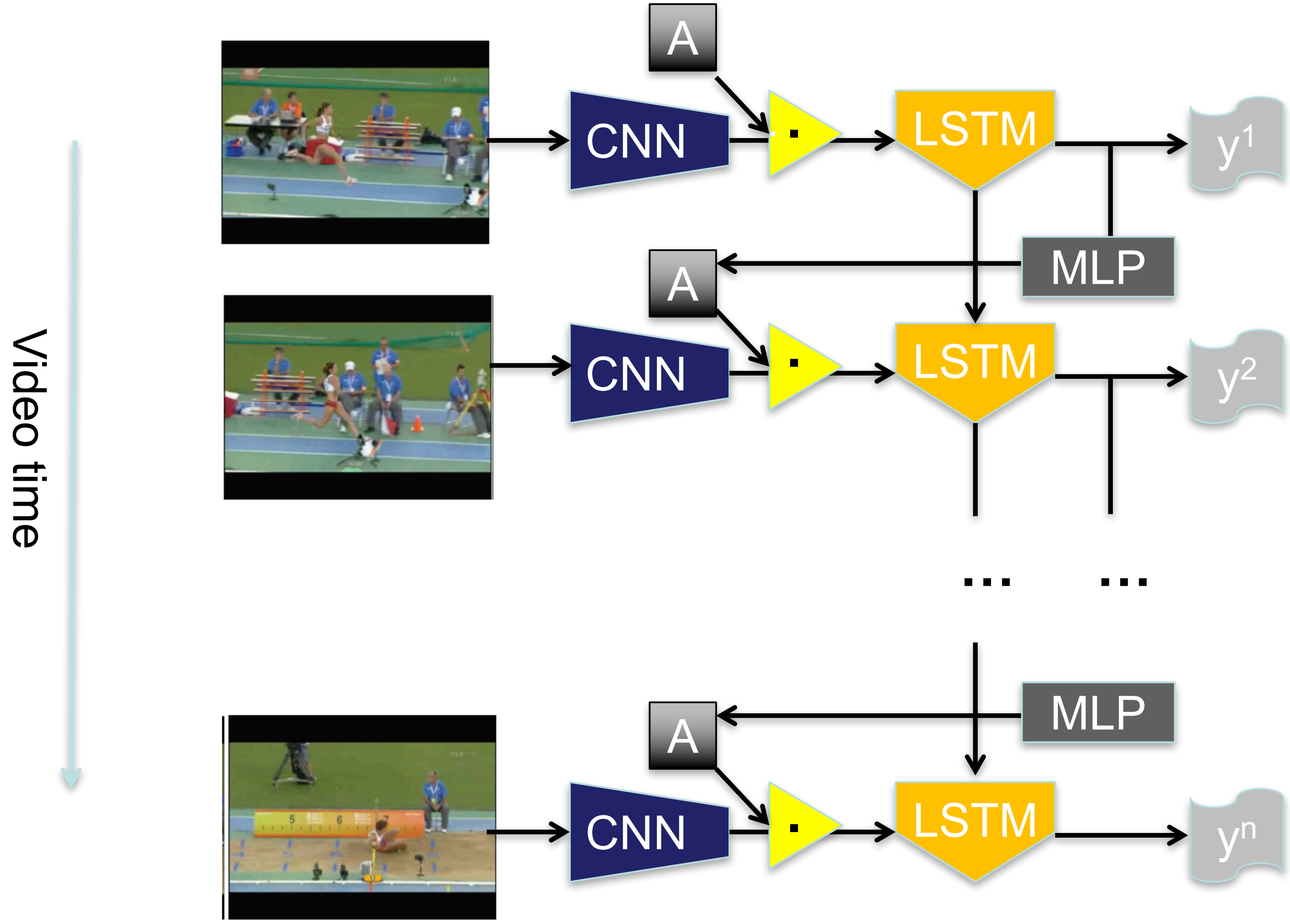


Convolutional ALSTM

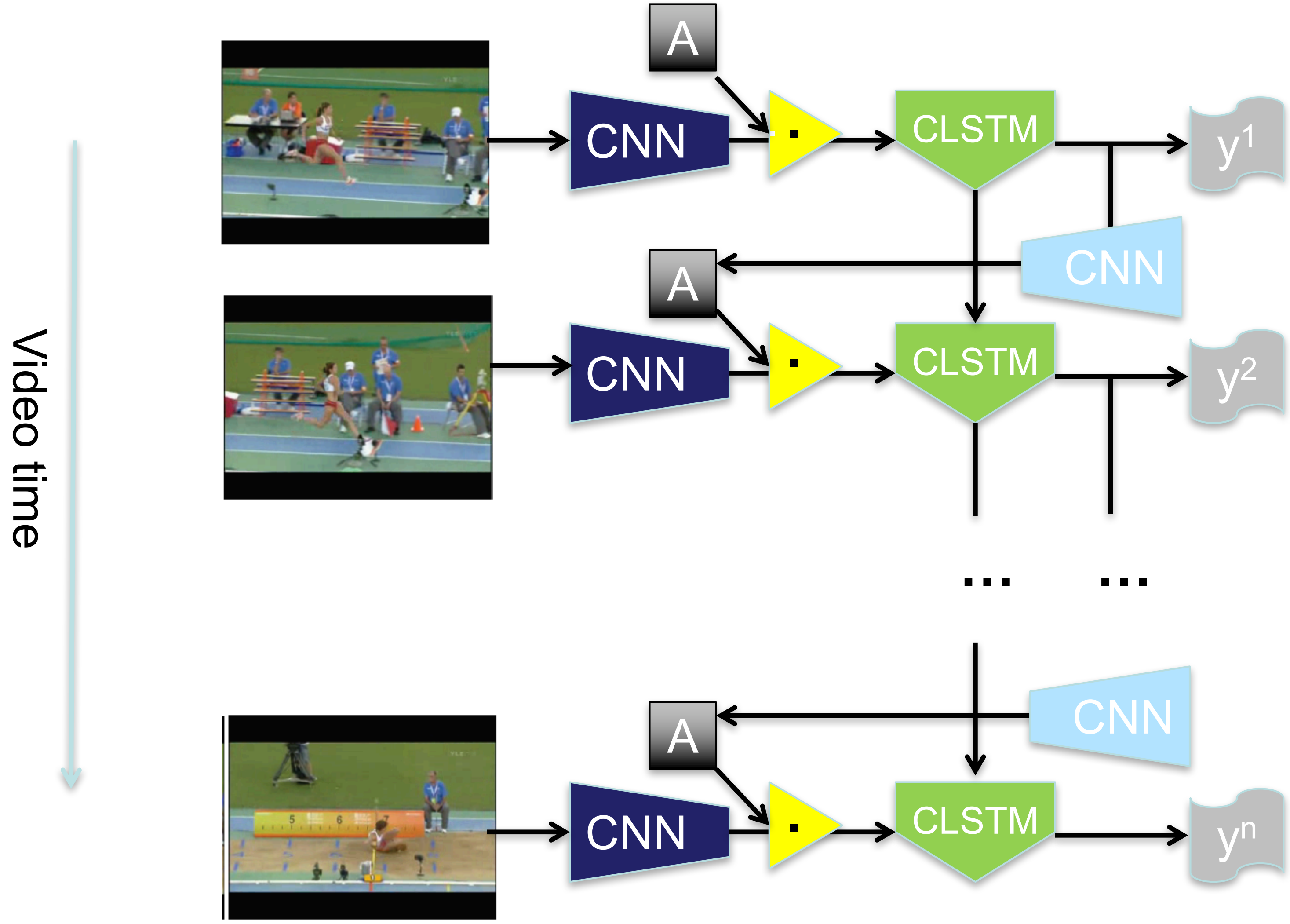


Convolutional ALSTM preserves spatial dimensions over time

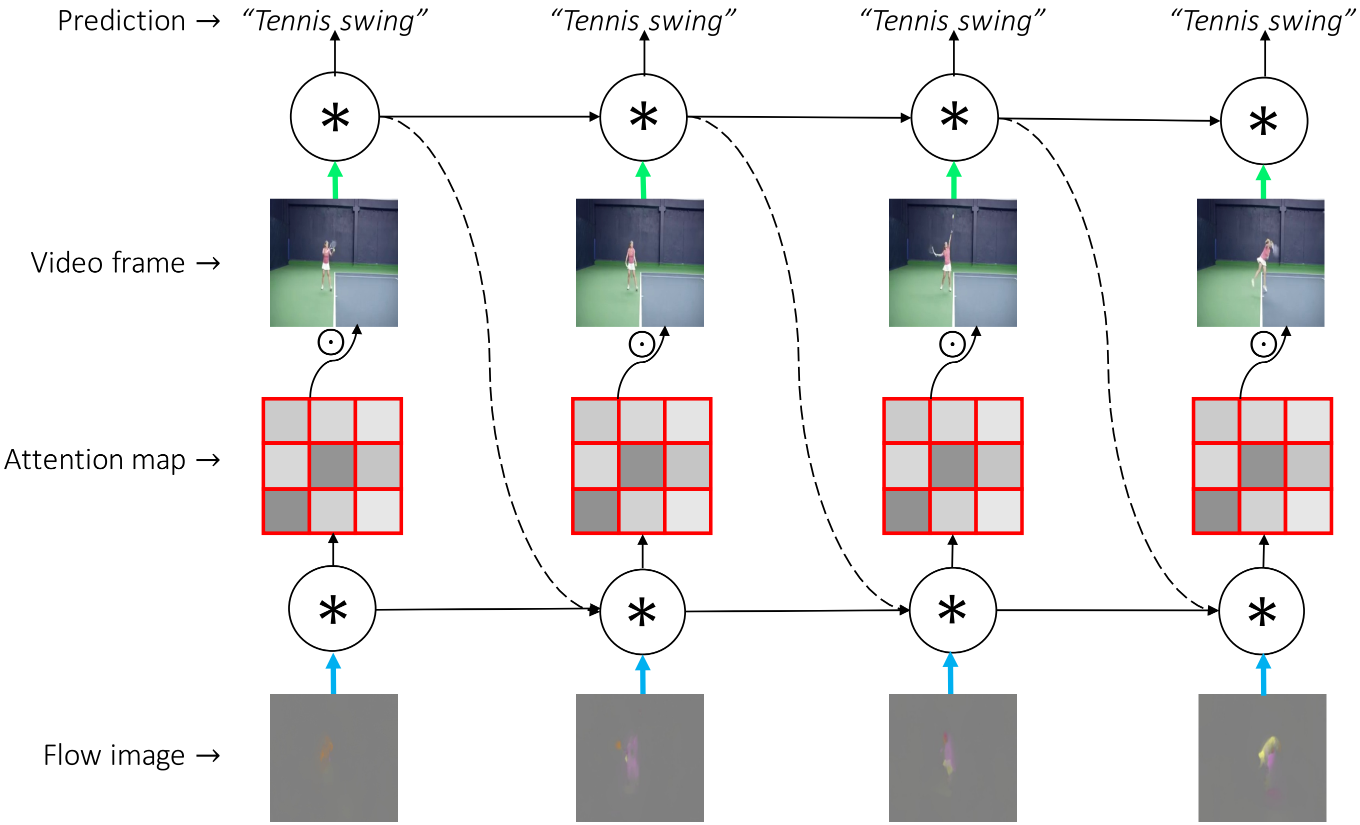
ALSTM



Convolutional ALSTM

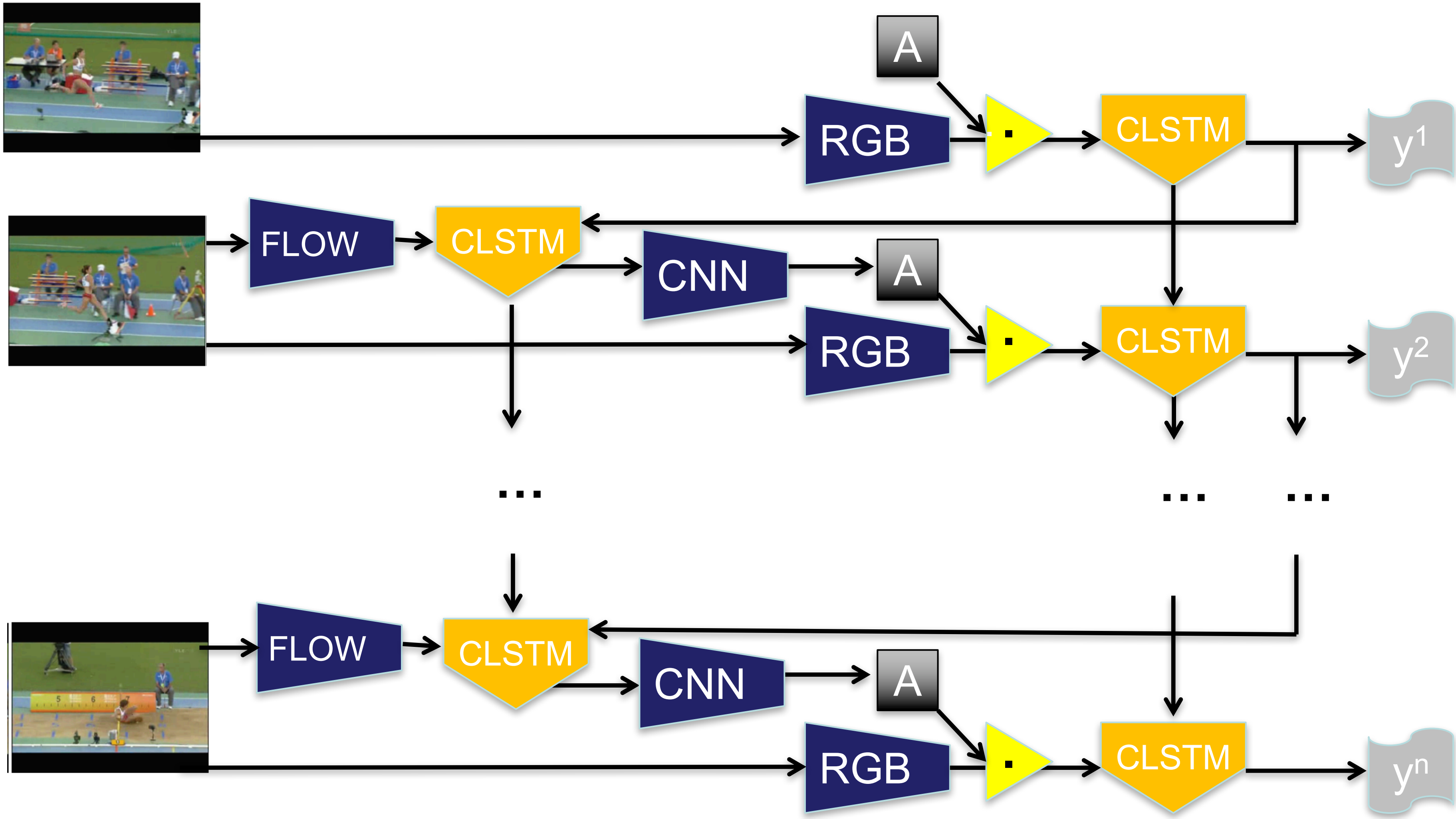


Motion-based attention

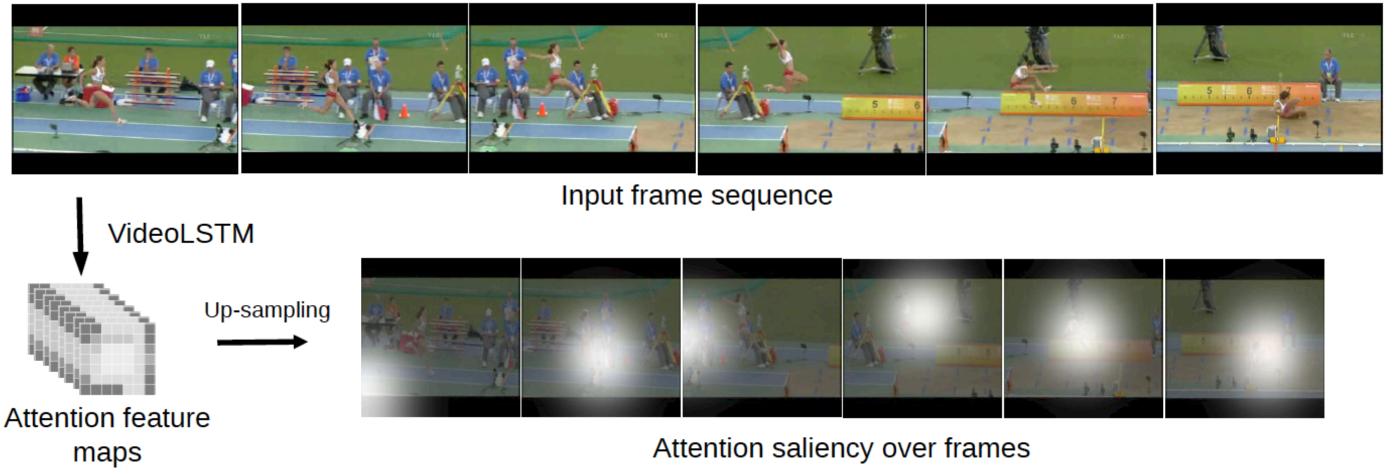


Motion information to infer the attention in each frame

VideoLSTM


A
RGB
·
CLSTM
y^1
FLOW
CLSTM
CNN
A
RGB
·
CLSTM
y^2
...
...
...
FLOW
CLSTM
CNN
A
RGB
·
CLSTM
y^n

Temporal smoothing



Input frame sequence

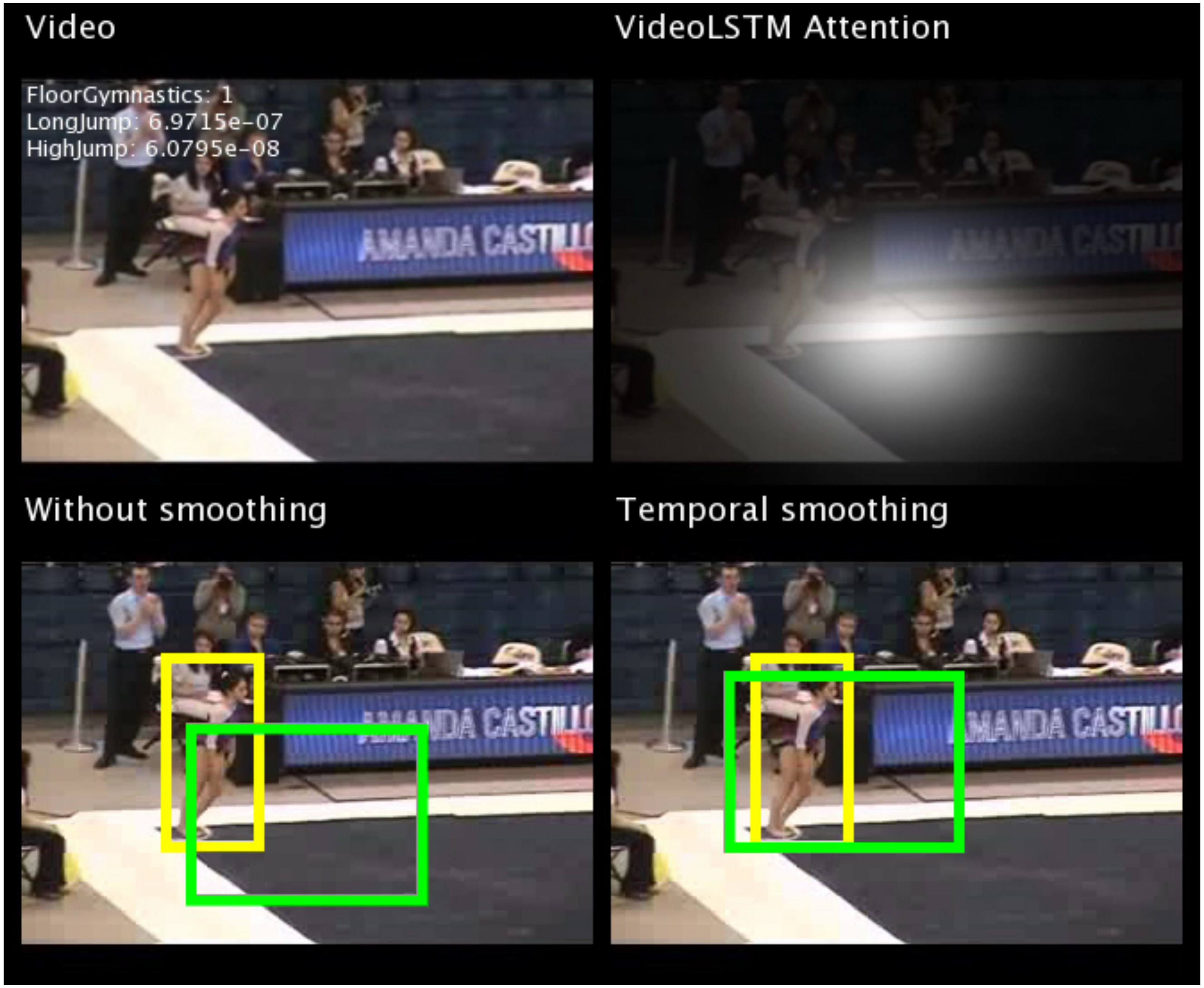
VideoLSTM

Attention feature maps

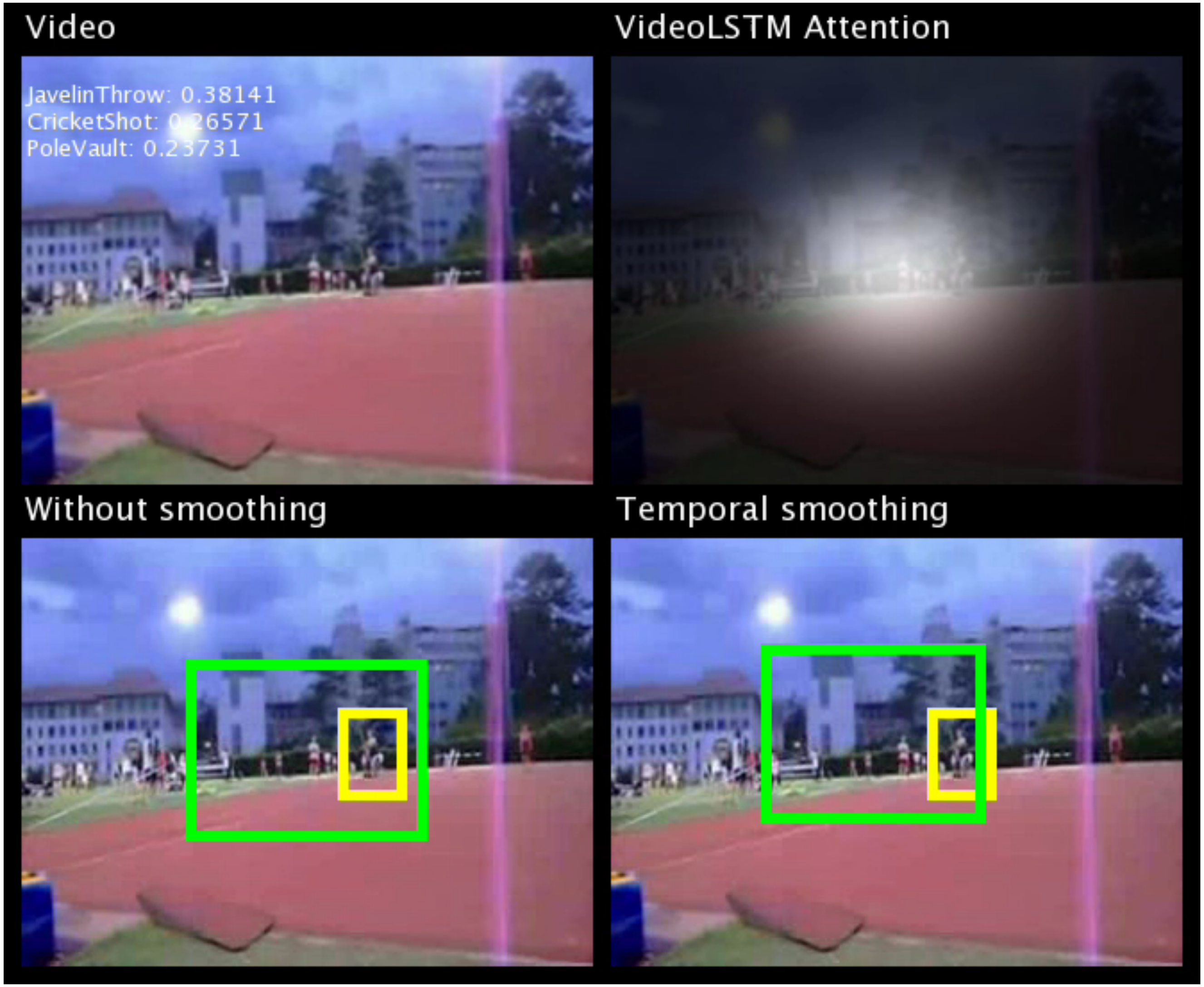
Up-sampling

Attention saliency over frames

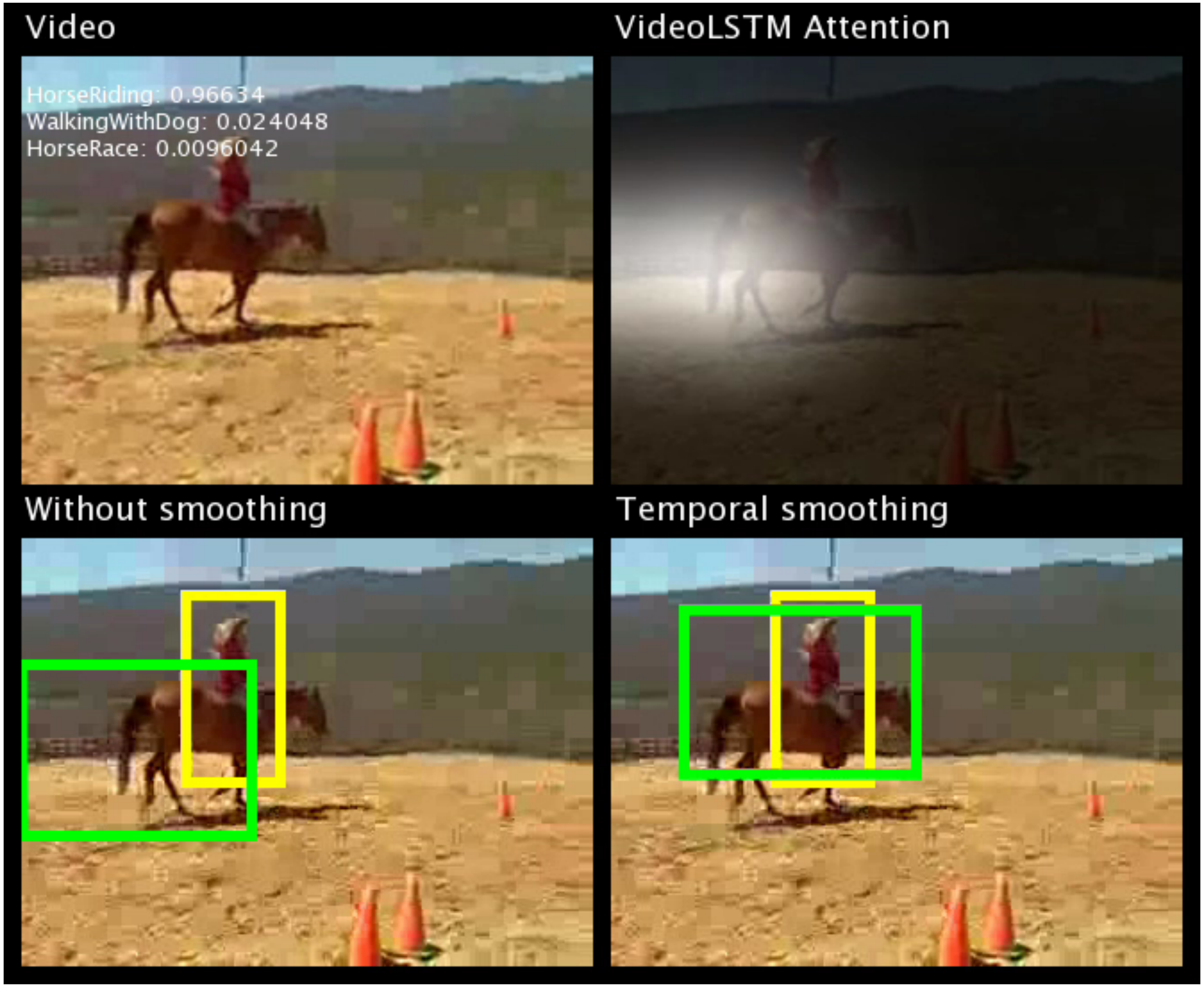
Qualitative results



Qualitative results



Qualitative results



Conclusions on VideoLSTM

Promising deep vision architecture for action localization

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

Localization from a video-level action class label only

Tomorrow

1. Bringing Structure to Deep Learning Vision
2. Understanding Deep Networks Visually
3. Object tracking by learning