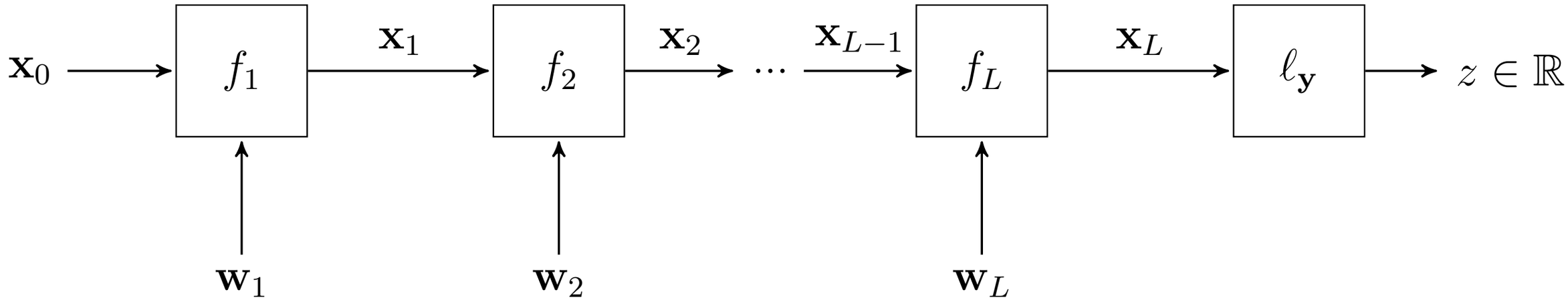
Deep learning

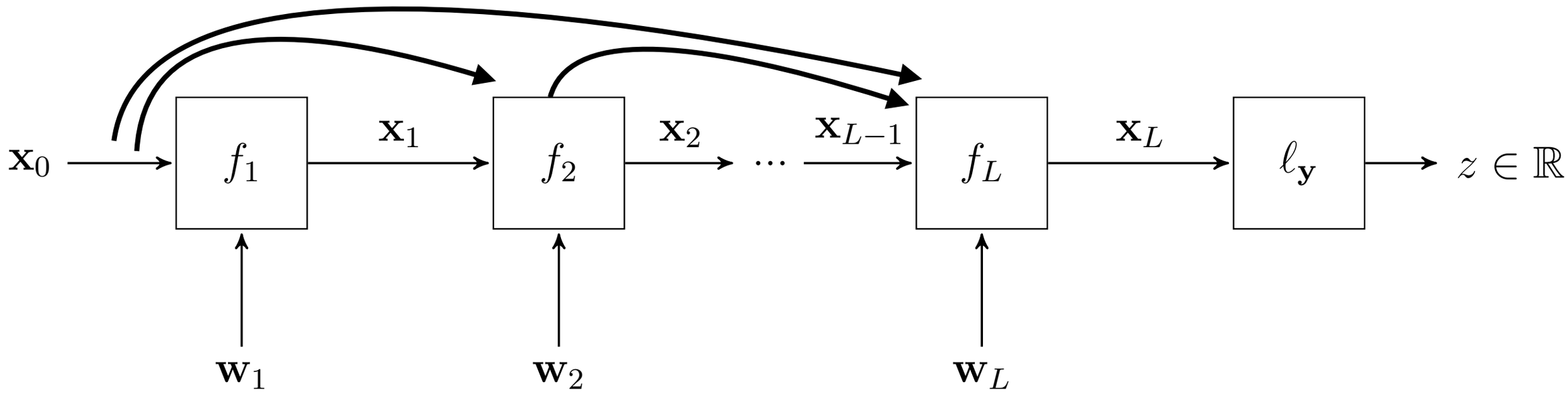
Outline

- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale, attention, recurrence, LSTMs
- Why does it work so well?

Recall: chaining modules

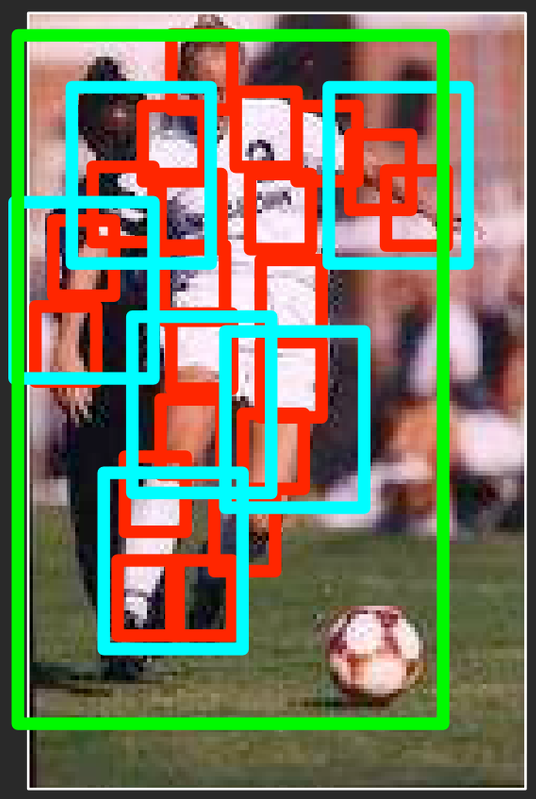


Extensions: DAG graphs



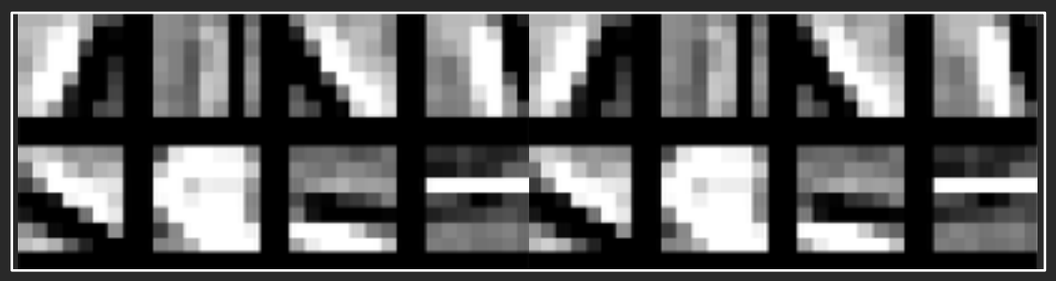
$$\mathbf{x}_1 = f_1(\mathbf{x}_0), \quad \mathbf{x}_2 = f_2(\mathbf{x}_1, \mathbf{x}_0), \quad ..., \quad \mathbf{x}_L = f_L(\mathbf{x}_1, \ldots, \mathbf{x}_{L-1}).$$

Why is this reasonable?



objects

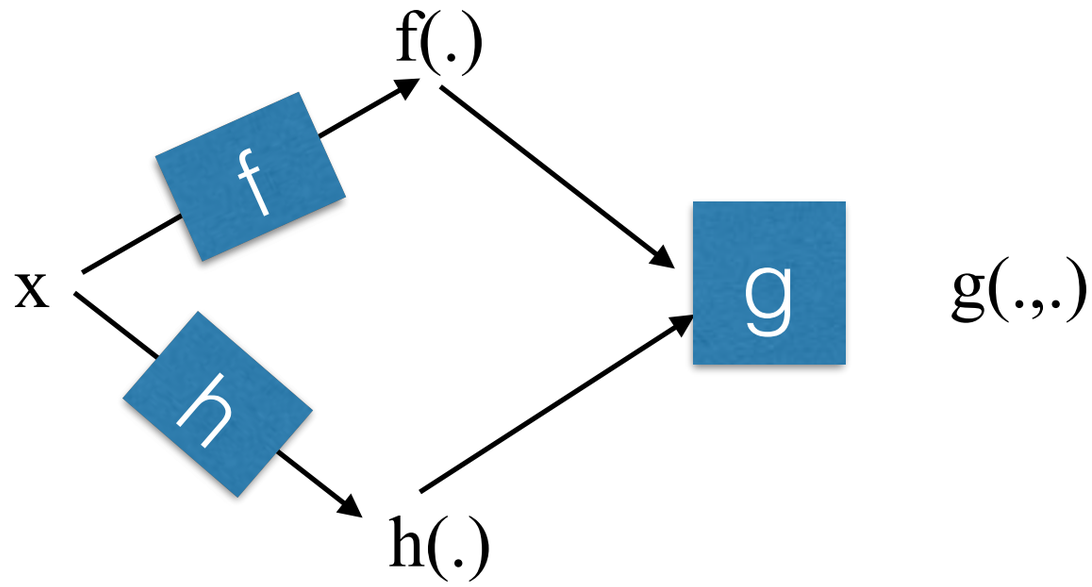
parts



subparts

Maybe objects depend on parts and subparts

Backprop on DAGs



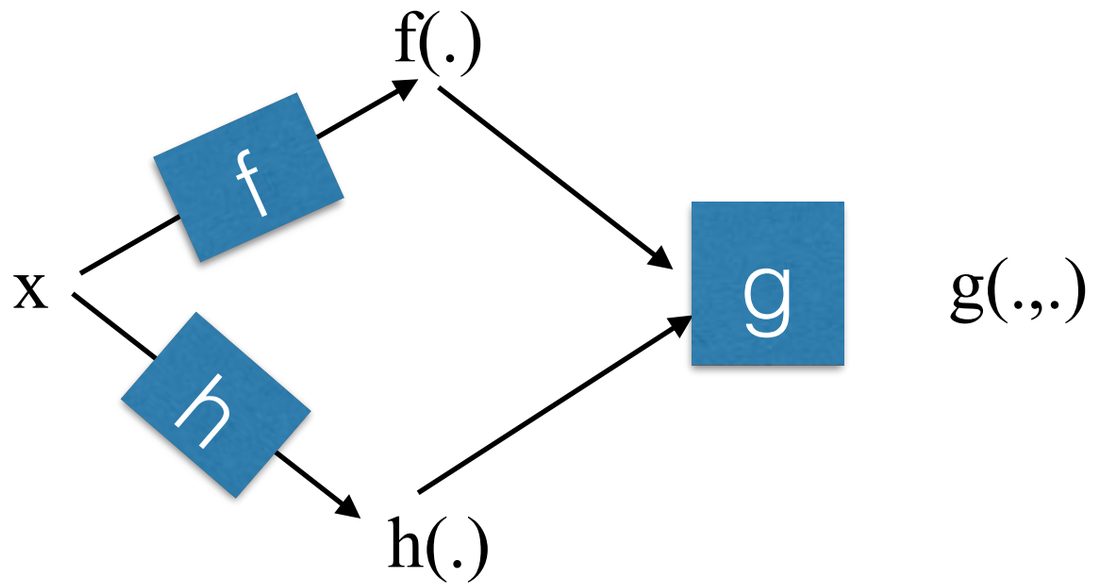
$$\frac{\partial g(f(x), h(x))}{\partial x} =$$

eg: $f(x) = ax + b$

$h(x) = x^2$

$g(y,z) = yz = x^2(ax+b)$

Backprop on DAGs

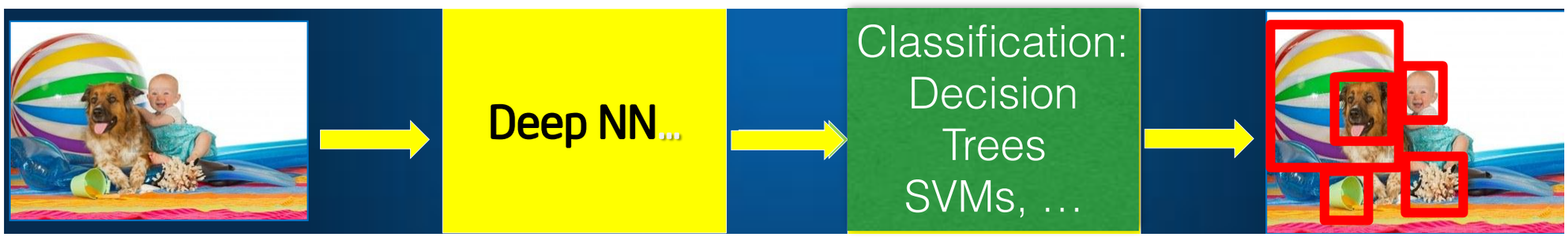


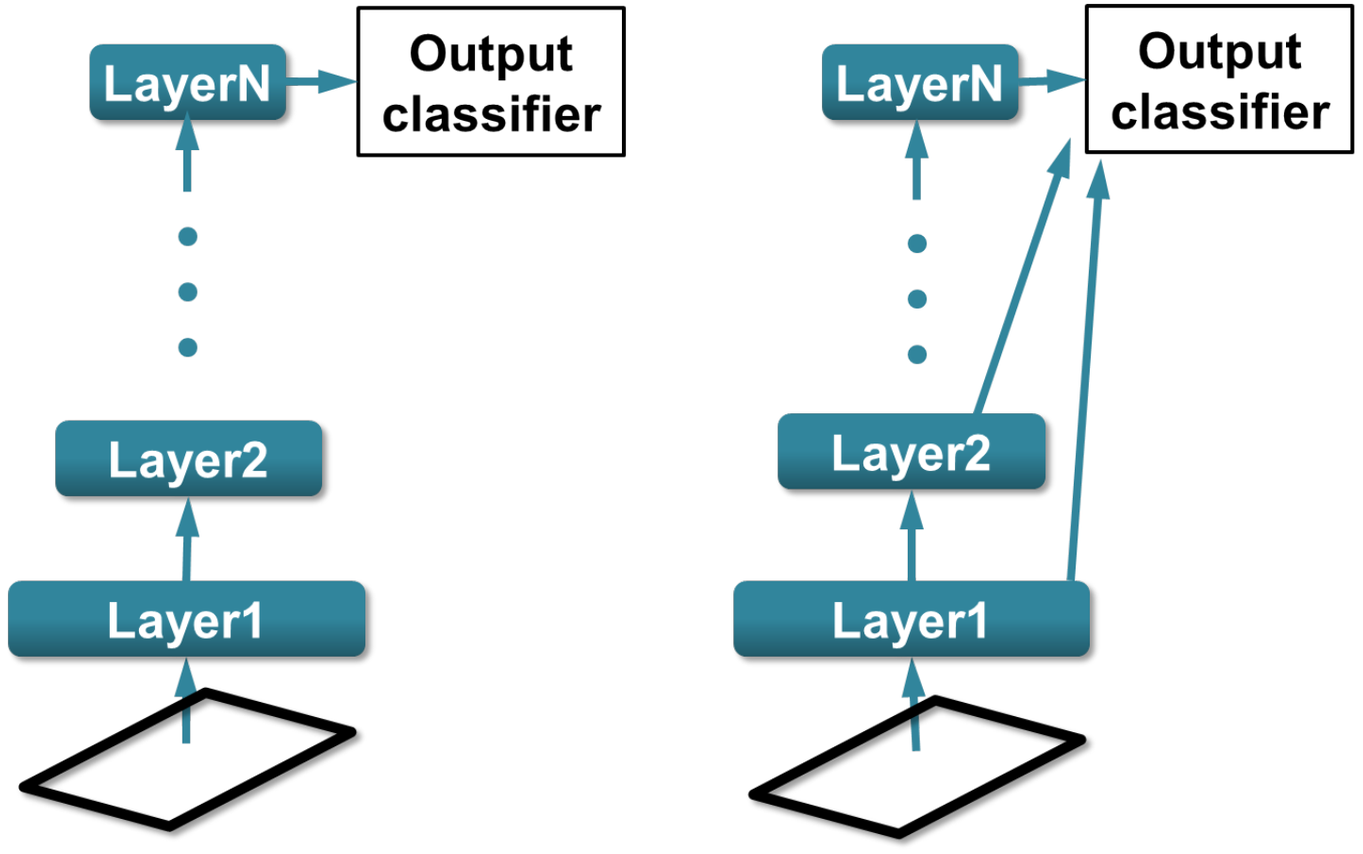
"Multivariable chain rule"

$$\frac{\partial g(f(x), h(x))}{\partial x} = \frac{\partial g}{\partial f}\frac{\partial f}{\partial x} + \frac{\partial g}{\partial h}\frac{\partial h}{\partial x}$$

Simply compute backprop signals along each path, adding up signals when paths merge

Multiscale classification

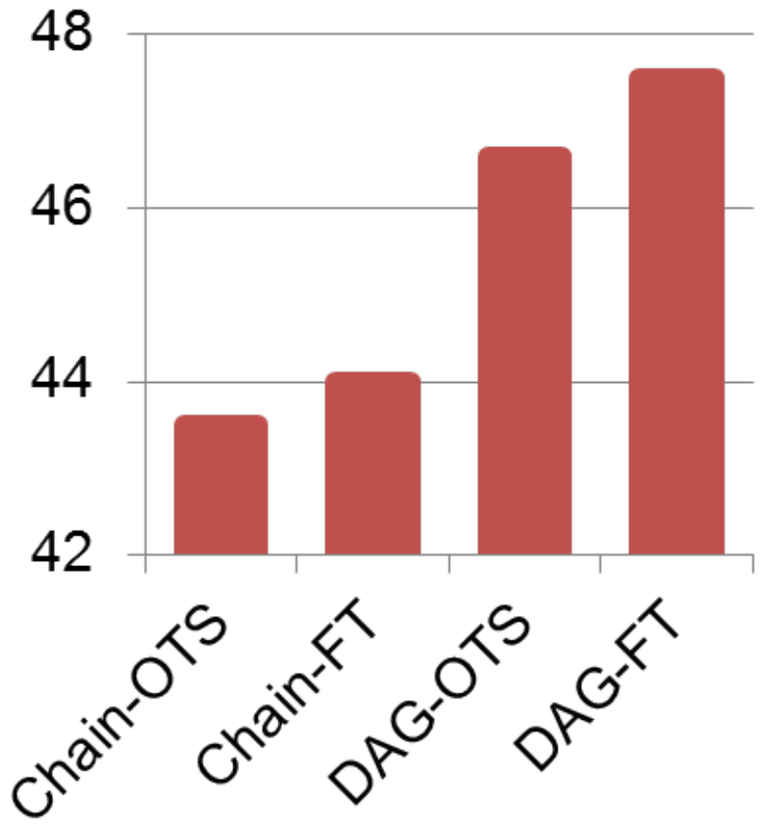




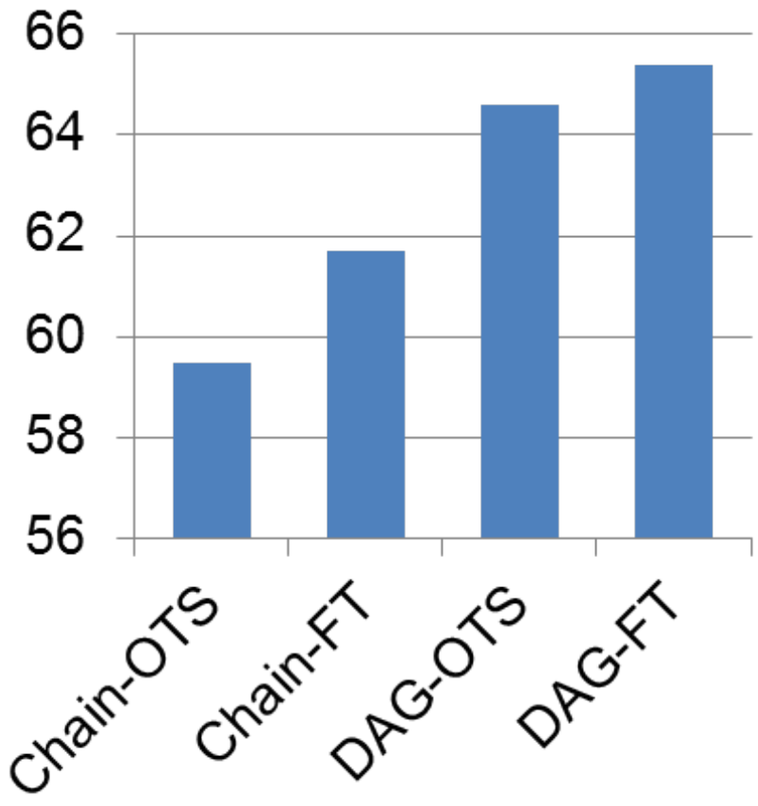
CNN

Multi-scale CNN

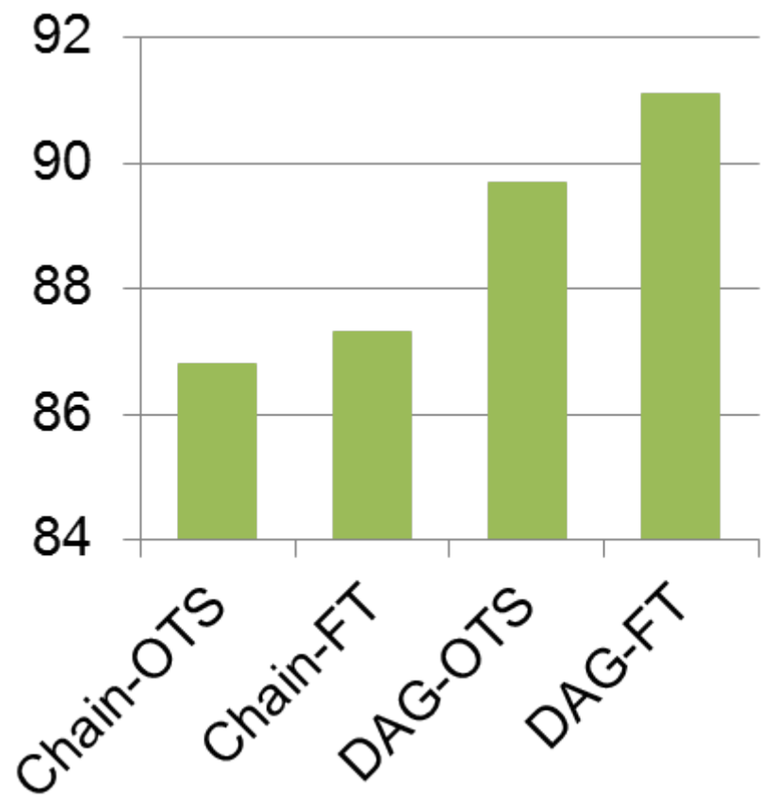
Multiscale classification



(a) SUN397



(b) MIT67

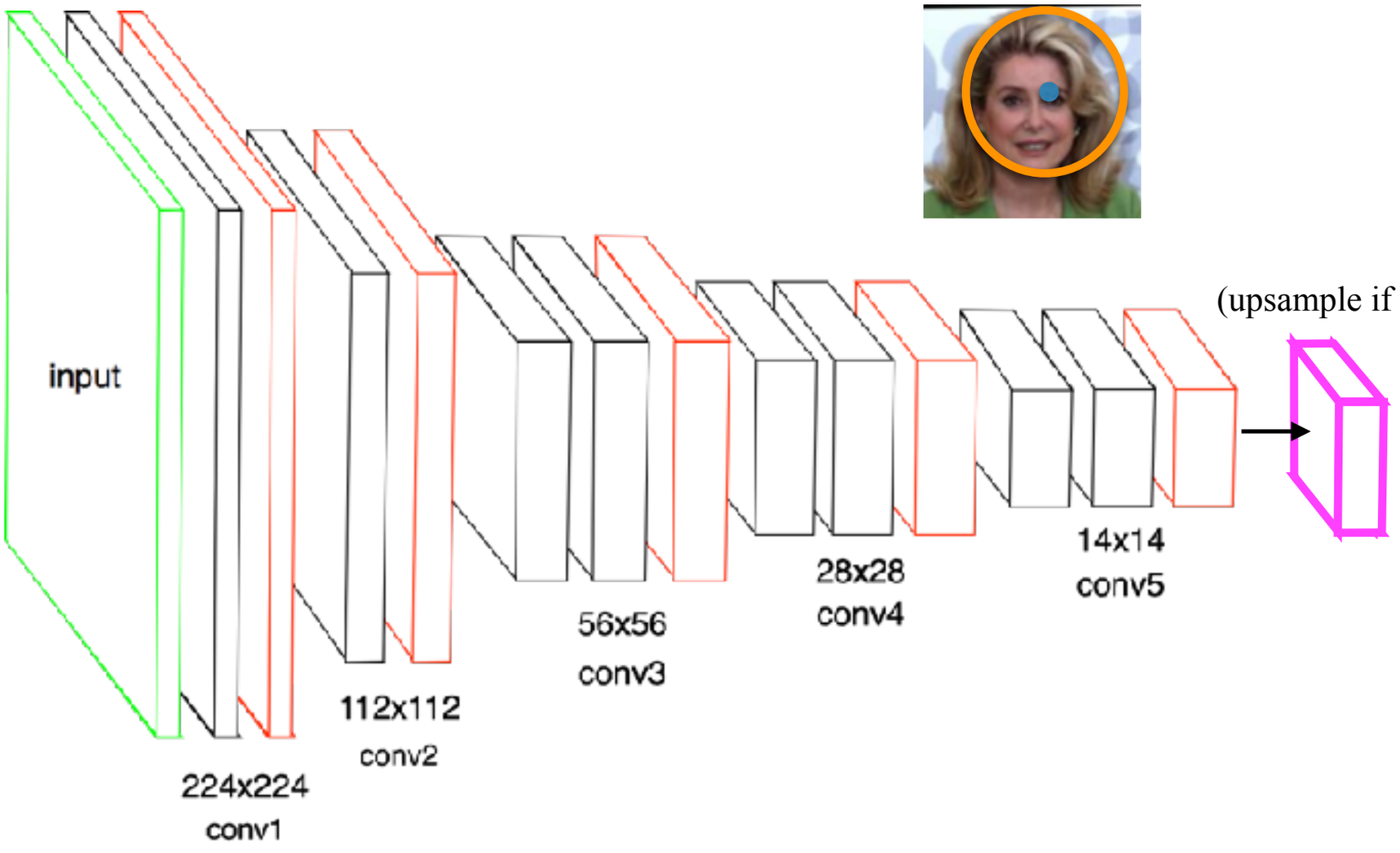


(c) Scene15

"DAG" = multiscale

FT = fine-tuning (use pre-trained model as starting point for gradient descent)

Pixel prediction

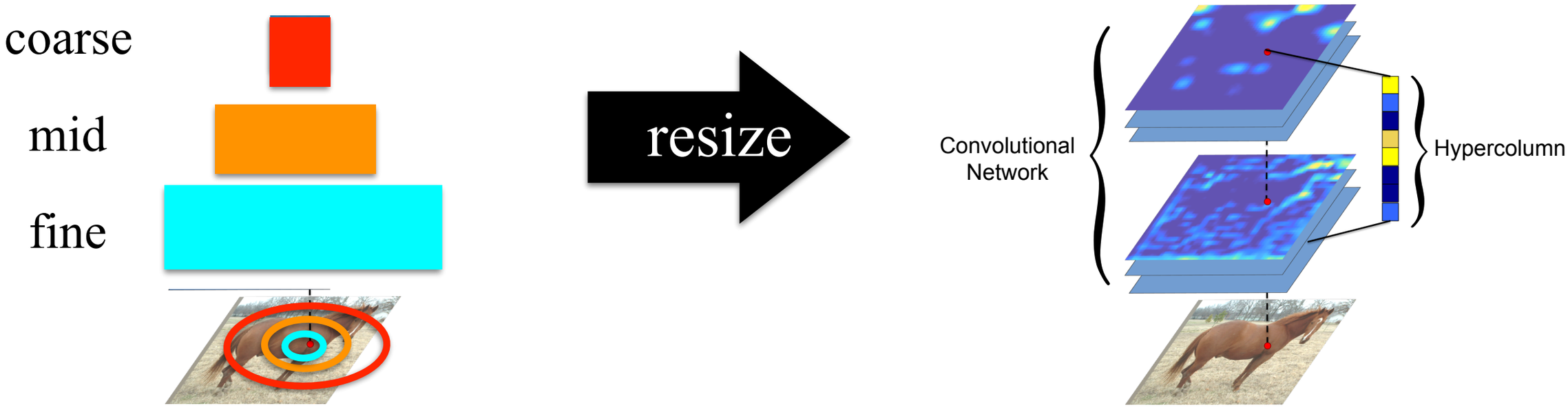


pixel class-labels
landmark heatmaps
normals/depth

Intuition: each pixel prediction makes use of a scanning-window (or receptive feild)

“Fully-convolutional networks (FCN)”
“Hypercolumns”
“Zoom-out models”

Multiscale pixel classification

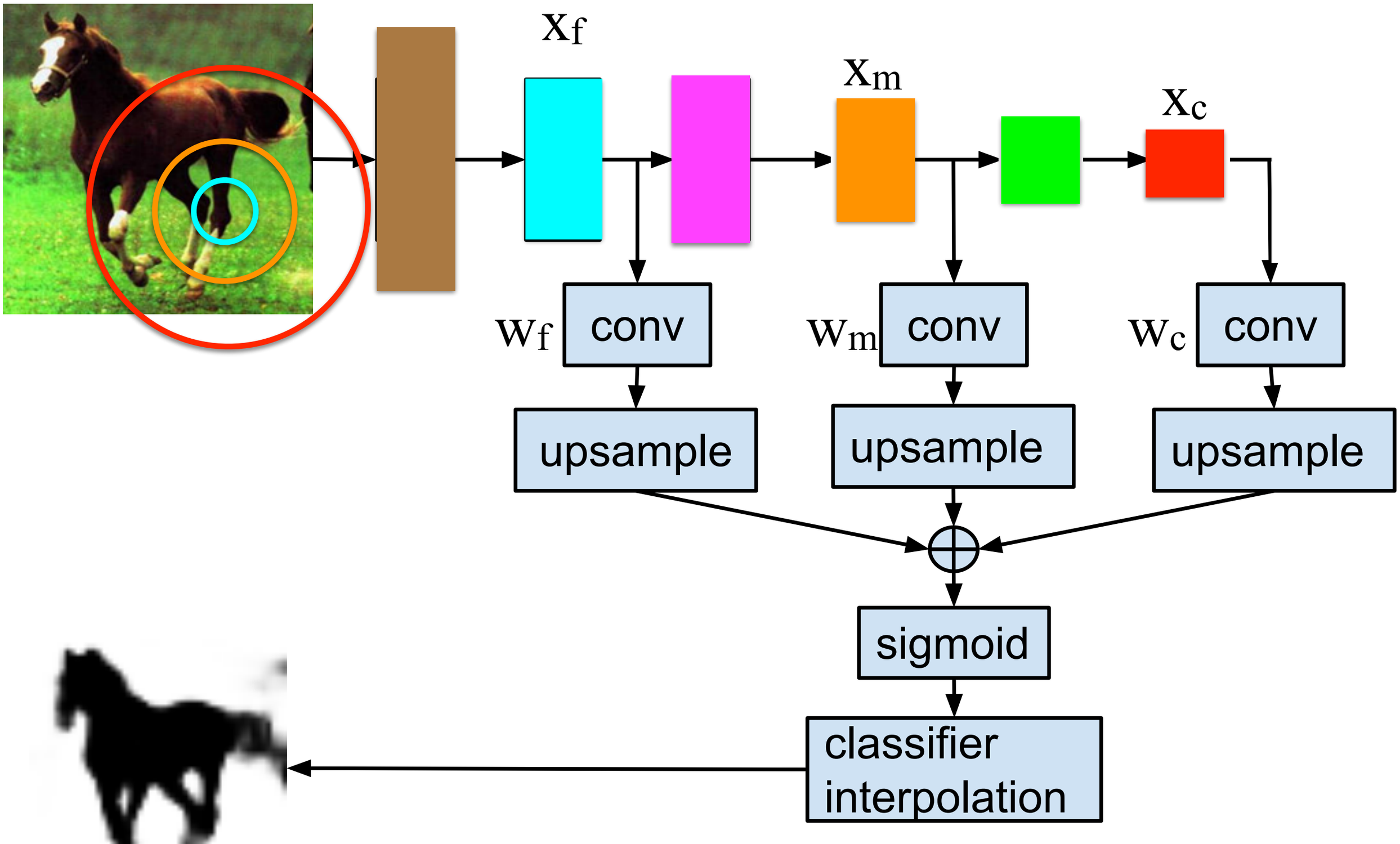


Hypercolumn: Multiscale “foveal” descriptor

Naive approach: resize feature maps and explicitly construct multiscale descriptor (hypercolumn) for each pixel

$$w \cdot x = \begin{bmatrix} w_c & w_m & w_f \end{bmatrix} \begin{bmatrix} x_c \\ x_m \\ x_f \end{bmatrix}$$

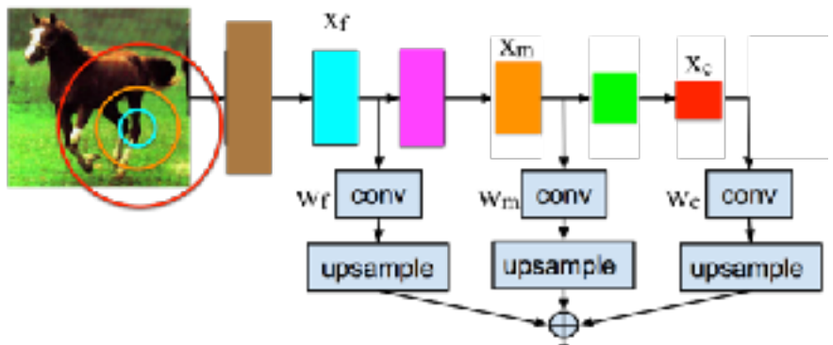
Alternate approach: resize *predictions*



Related topic: dilated convolutions

Don’t ever downsample, but make use of “zero-interlaced” filters

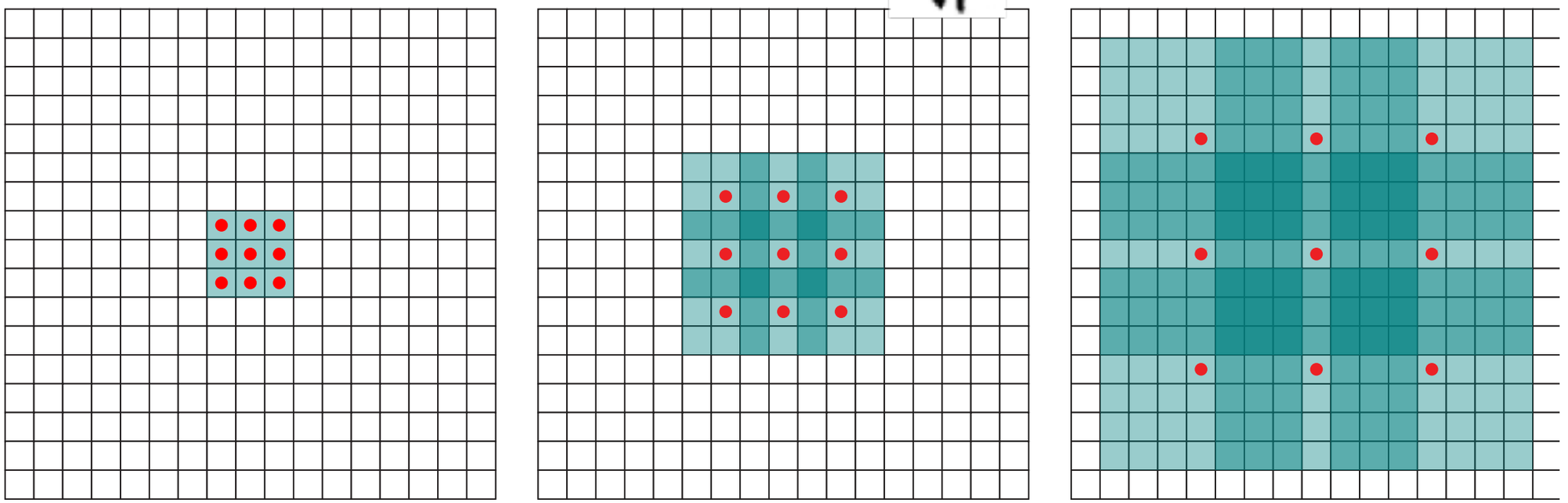
Published as a conference paper at ICLR 2016



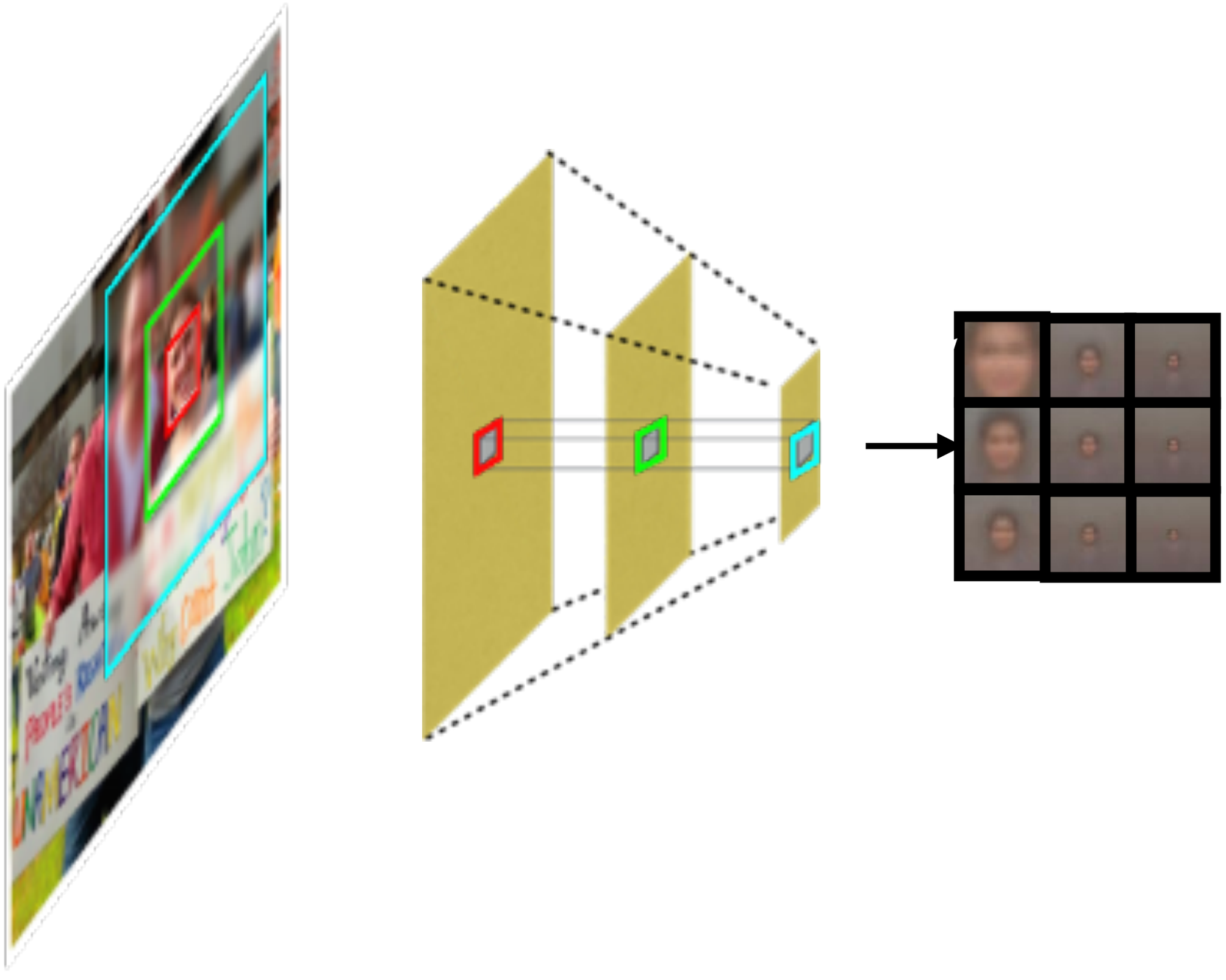
MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Fisher Yu
Princeton University

Vladlen Koltun
Intel Labs

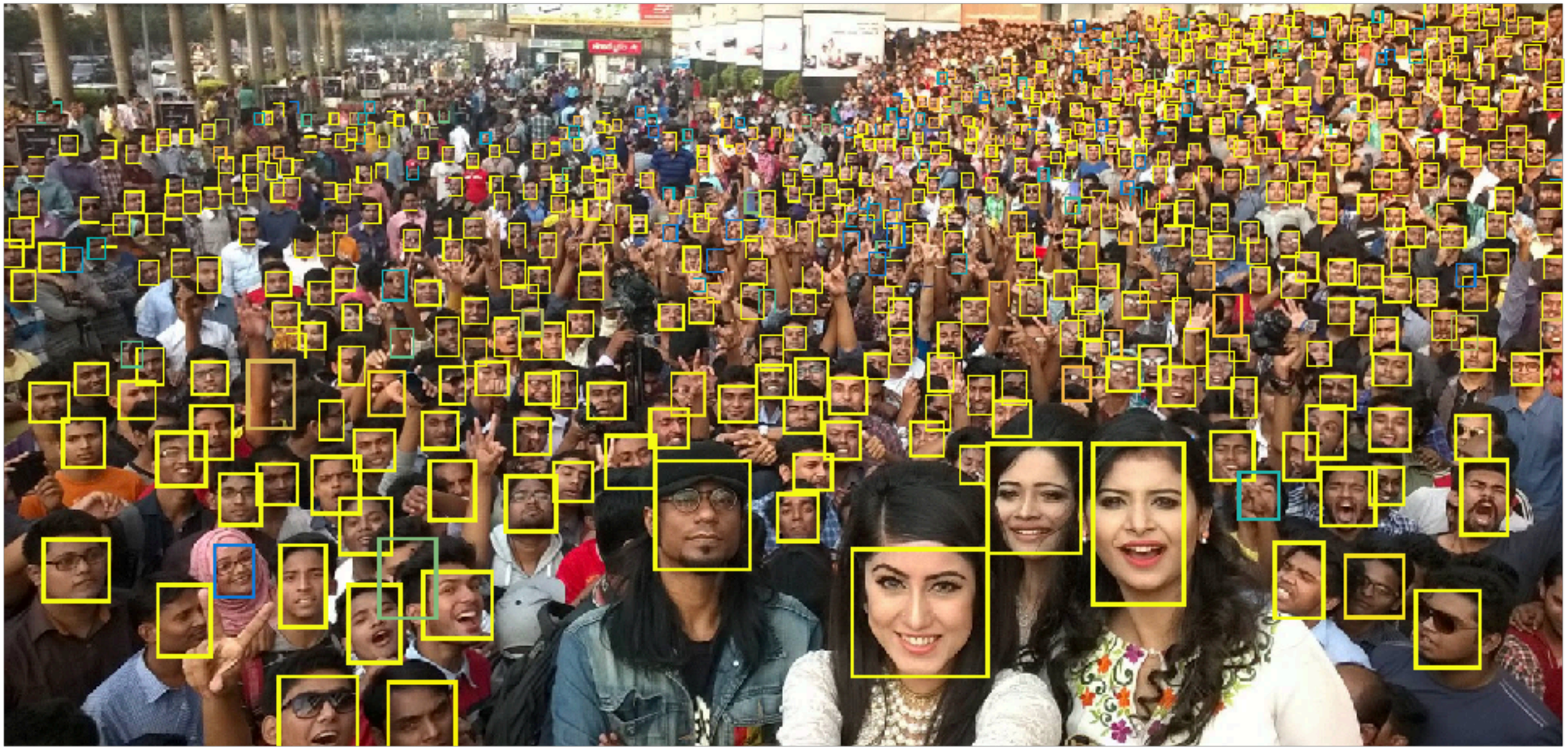


Object detection as heatmap prediction



Train sepearate heatmaps for large and small face detection

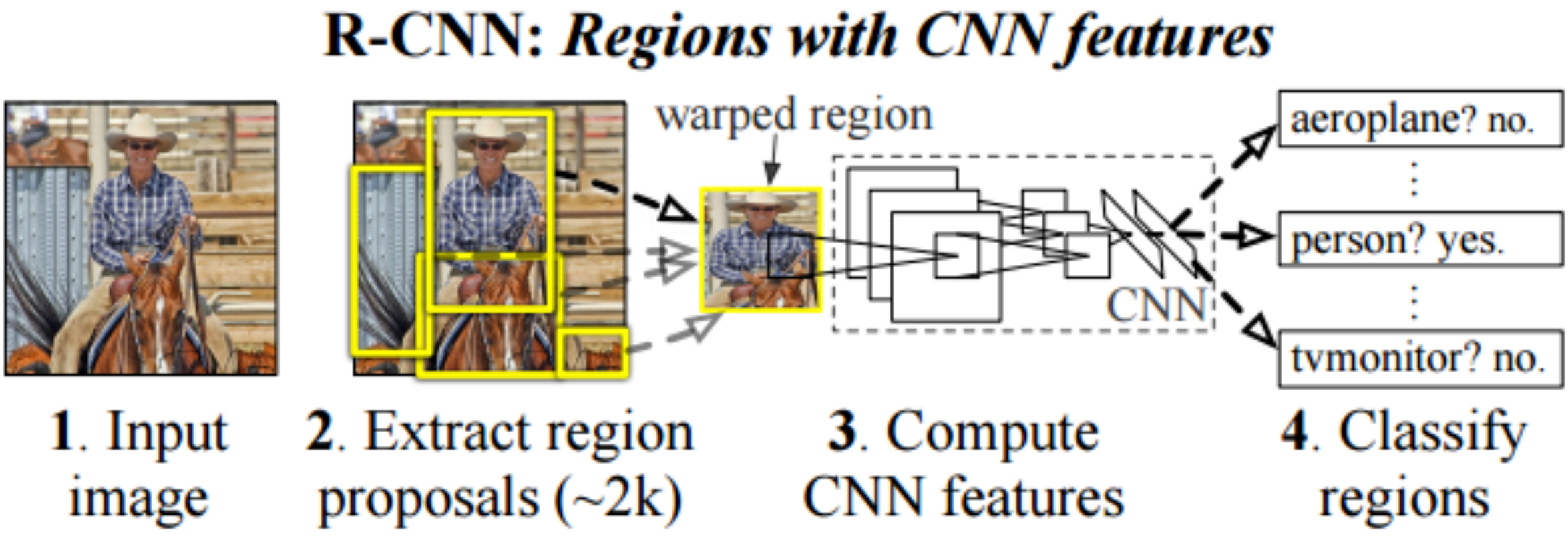
Can apply directly to object detection



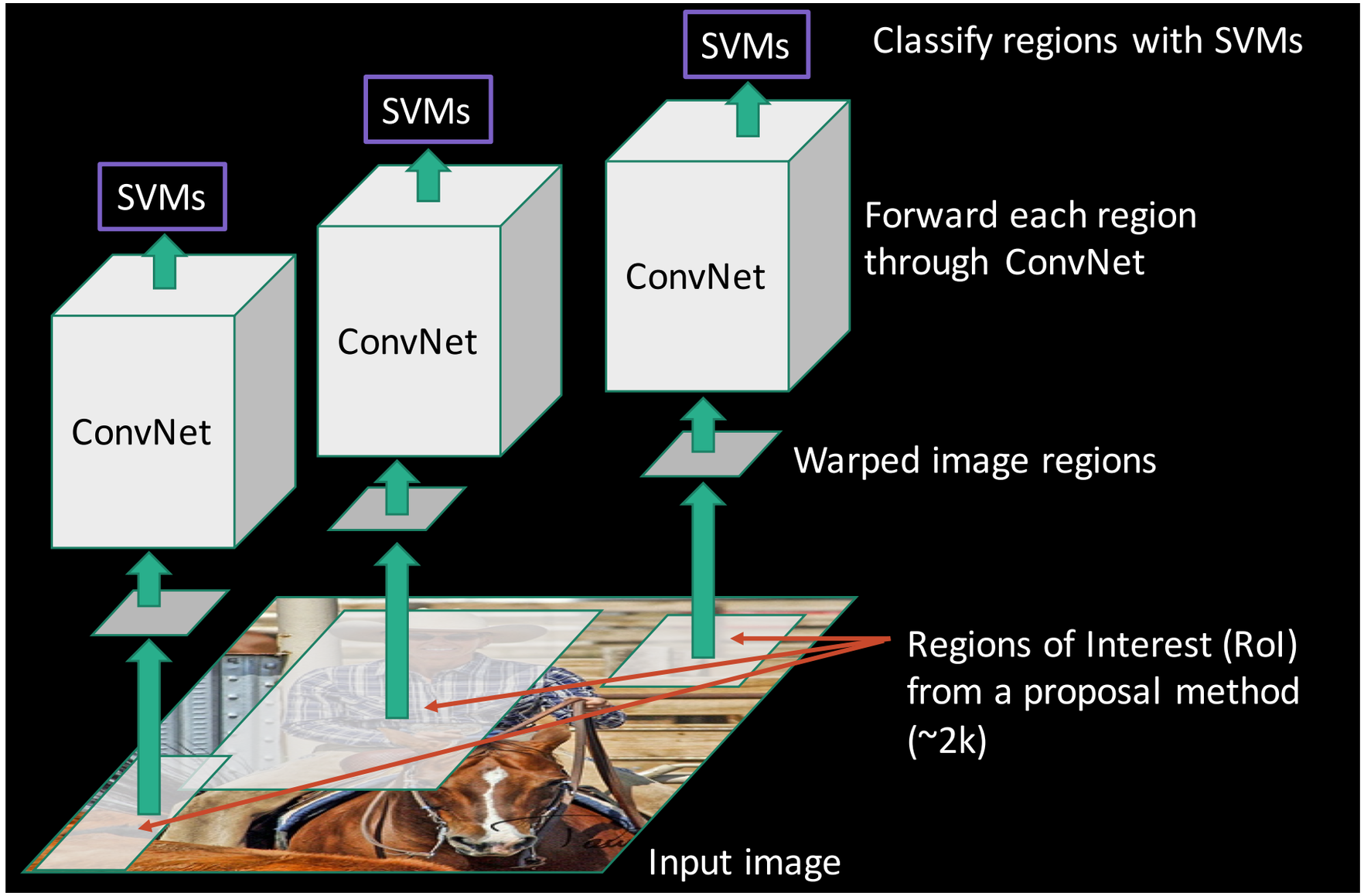
Hu et al, CVPR 17

10 ms on GPU

Dominant approach for object detection: region-based CNNs



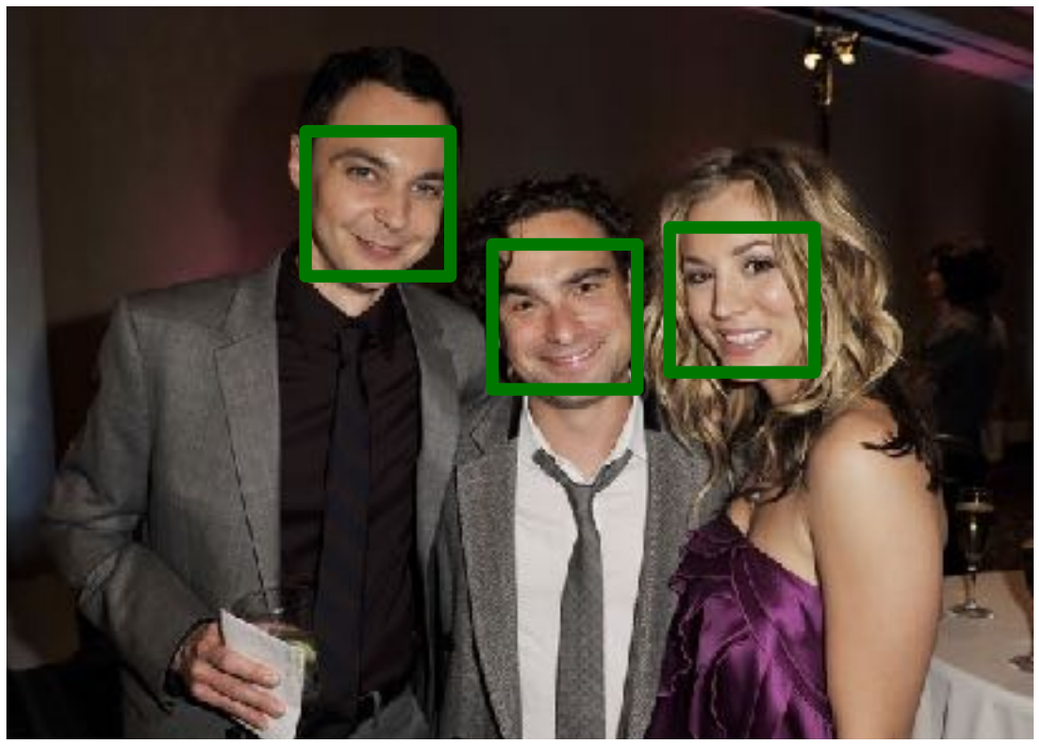
Region CNNs (RCNNS)

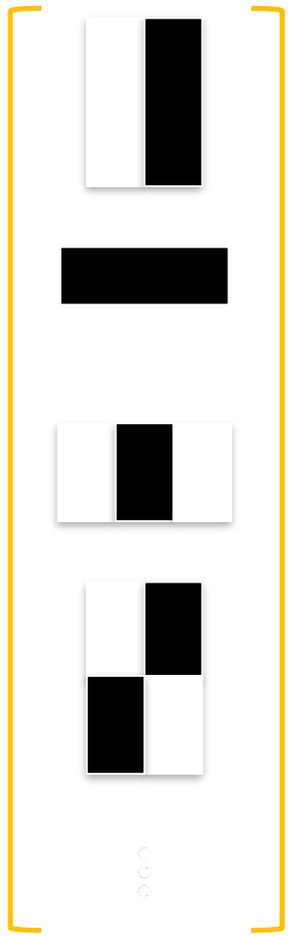


"Faster RCNN": train a CNN to produce object proposals (similar to face detector)

Aside:
attentional cascades for recognition

ViolaJones (OpenCV detection)





Conceptually, we can think of as a (sort-of) linear classifier defined on box-filter features (Haar wavelets)

https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

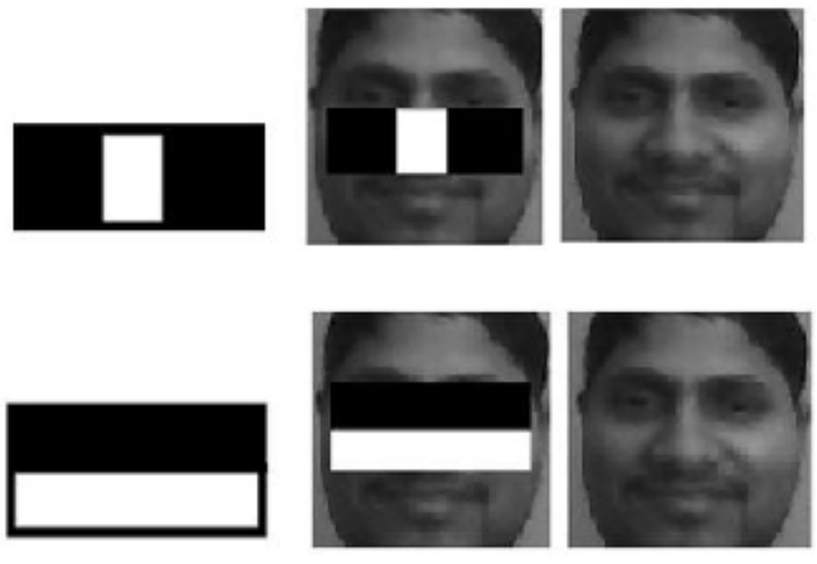
Attentional cascades

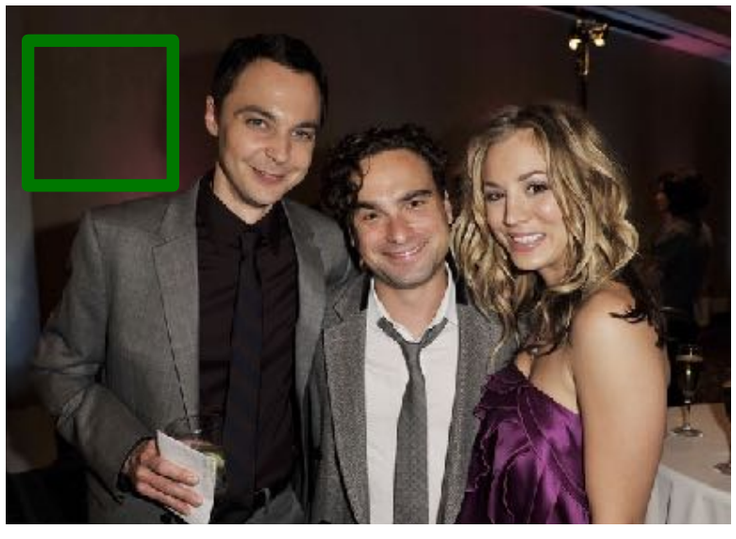
Final classifier is a linear combination of thresholded features

$$h(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

Learn a sequential cascade of classifiers

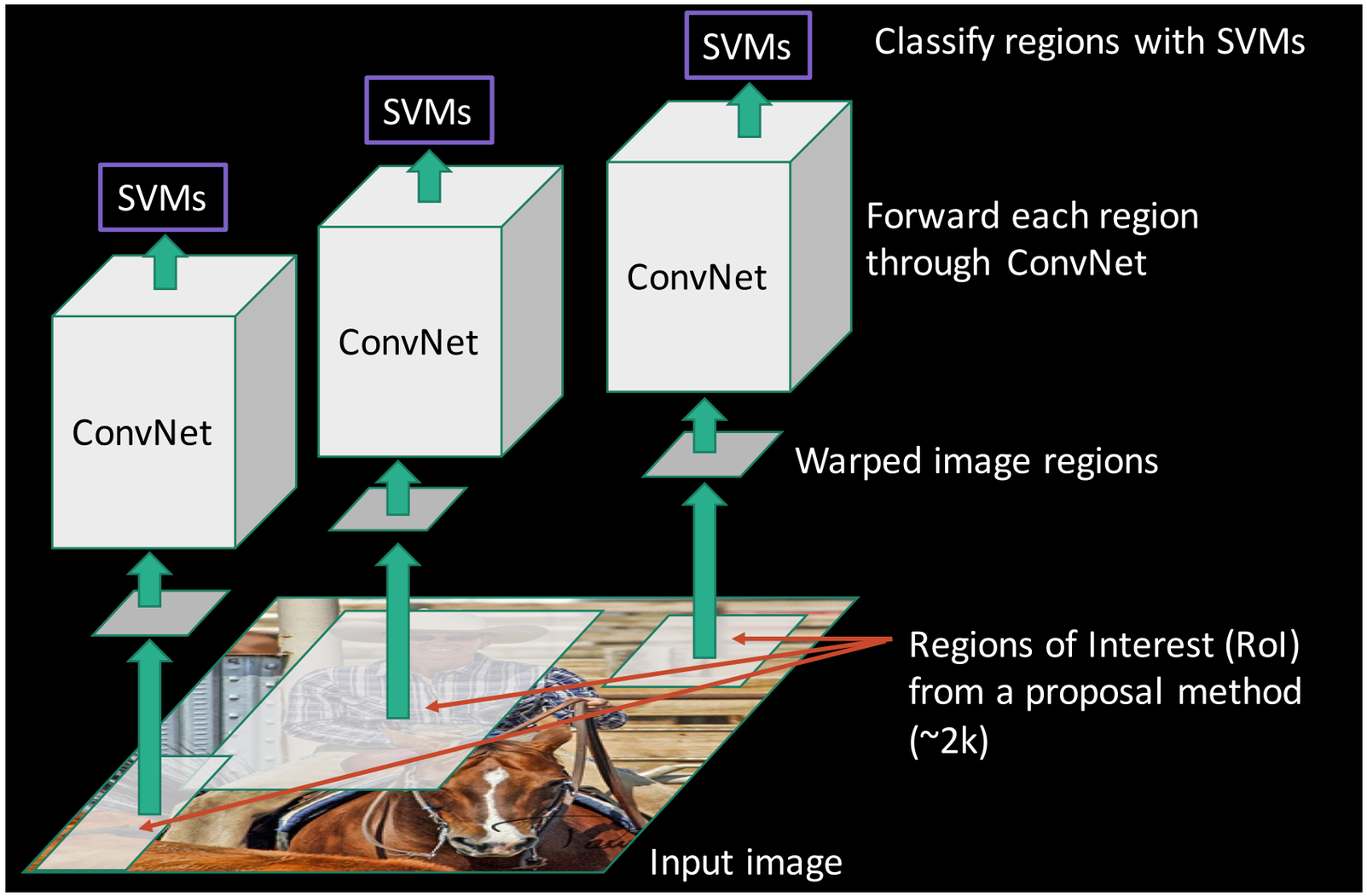




These 2 features can filter out 60% of BG windows (without missing essentially *any* faces)

This kind of *sparse* reasoning is currently missing in deep networks (but an active area of research known as *attentional* deep networks)

Proposal-based CNNs



Can we learn a proposal network that produces accurate object detectors?
Requires us to backprop through a warp
(e.g., compute gradient of loss wrt warp parameters)

Solution:
Spatial transformer networks

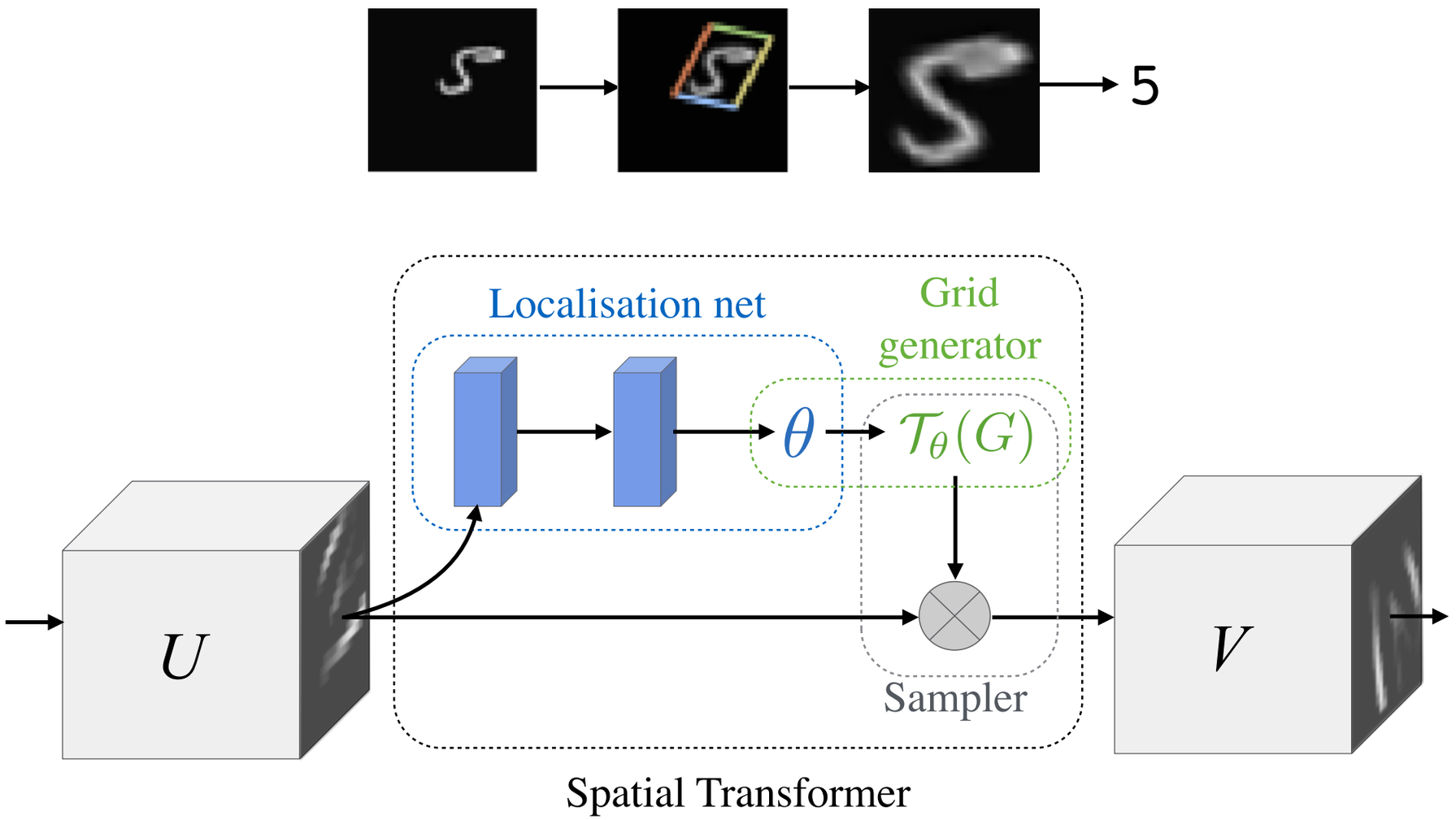


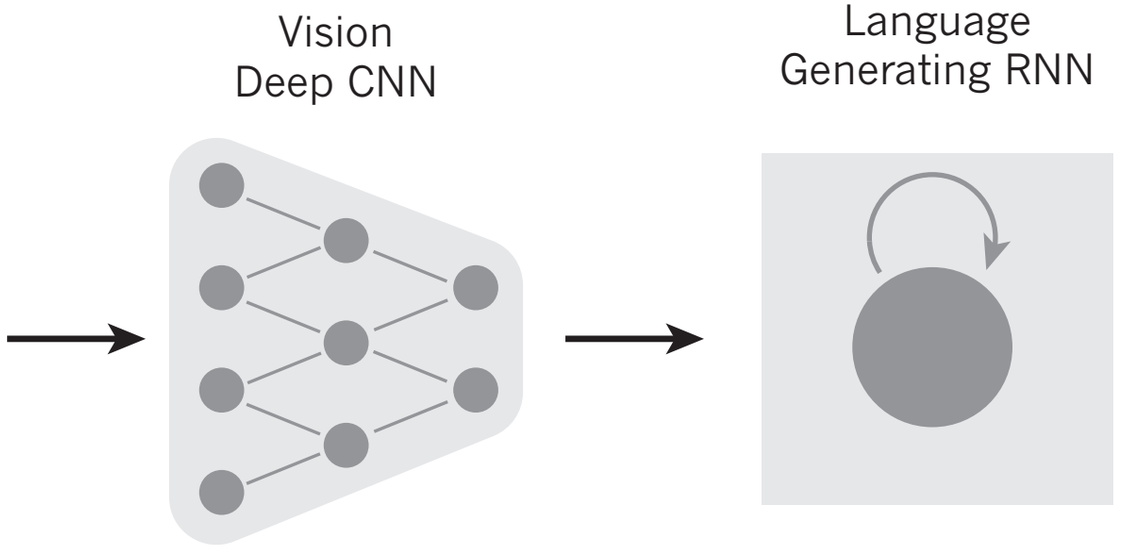
Figure 2: The architecture of a spatial transformer module. The input feature map U is passed to a localisation network which regresses the transformation parameters θ. The regular spatial grid G over V is transformed to the sampling grid $\mathcal{T}_\theta(G)$, which is applied to U as described in Sect. 3.3, producing the warped output feature map V. The combination of the localisation network and sampling mechanism defines a spatial transformer.

Outline

- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale, attention, recurrence, LSTMs
- Why does it work so well?

Extensions: recurrent CNNs

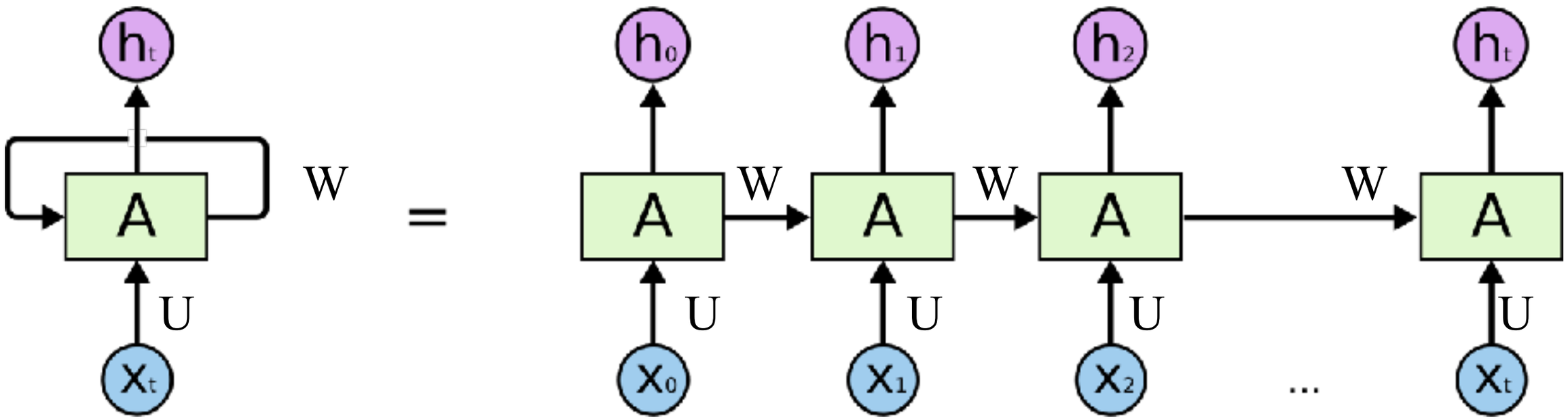
Activation depends on below layer and previous value of activation



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

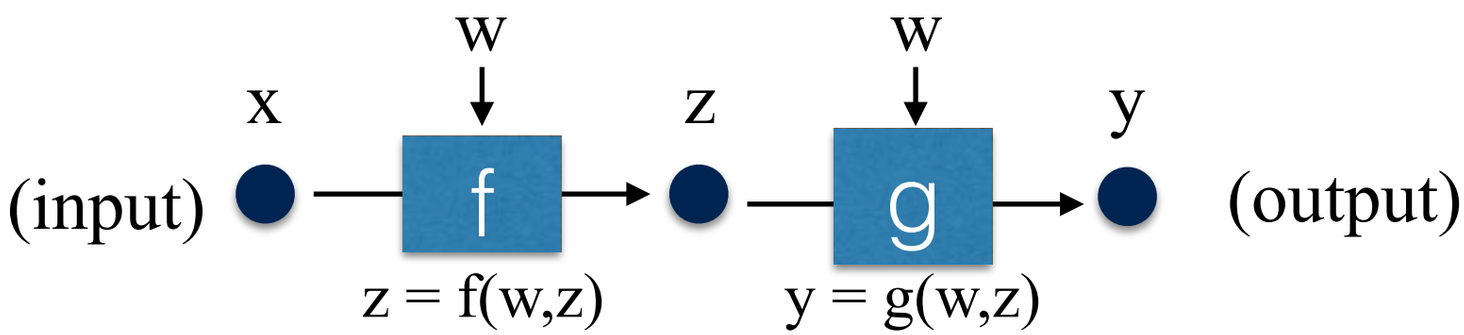
Extensions: recurrent CNNs



Can unroll into a “standard” CNN with tied weights

Does this complicate learning?

Recurrent nets



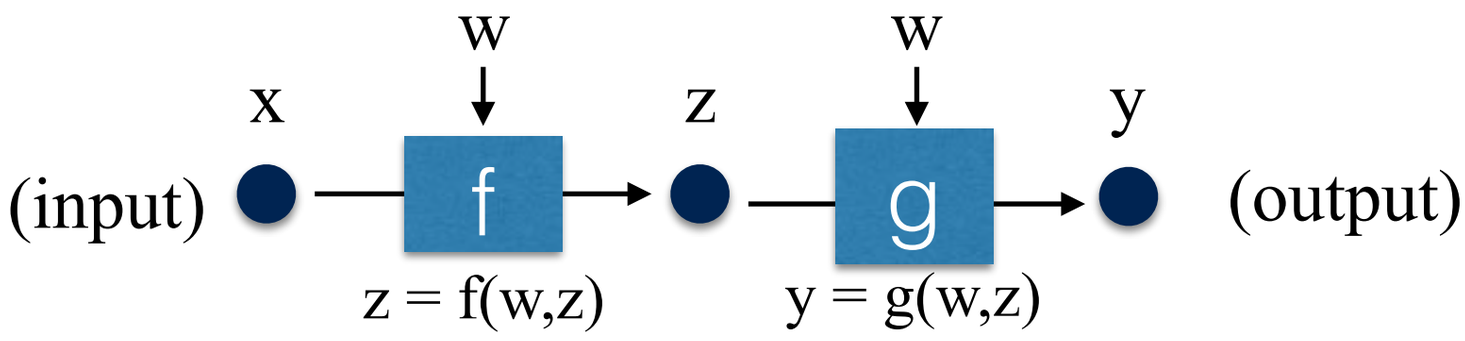
dy/dw = ?

eg: $z = wx$

$y = wz$

$y = w^2x$

Recurrent nets

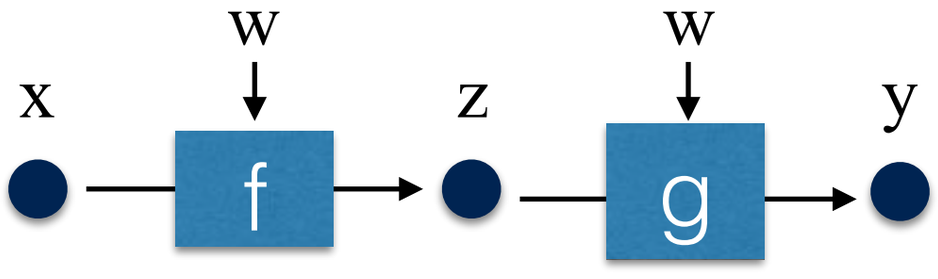


$$dy/dw = dg/dw + dg/dz\ dz/dw$$

$$dz/dw = df/dw$$

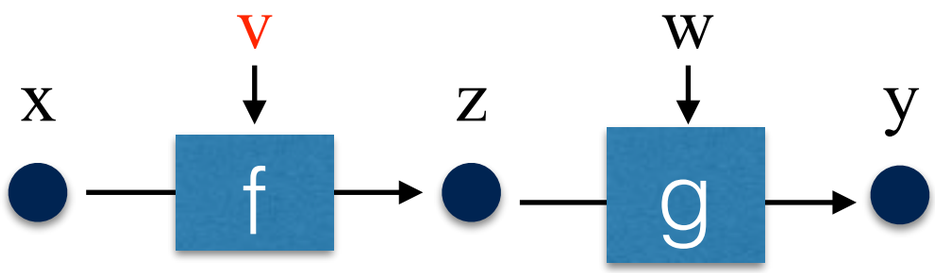
$$dy/dw = dg/dw + dg/dz\ df/dw$$

Recurrent nets



$$dy/dw = dg/dw + dg/dz\ df/dw$$

1. Naively apply backprop assuming weights aren't tied together

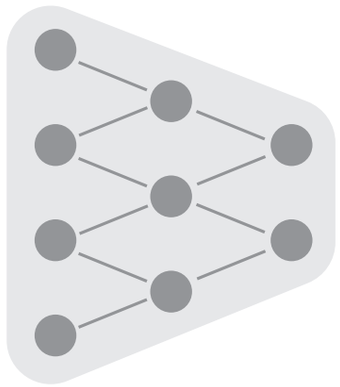


$$dy/dw = dg/dw$$
$$dy/dv = dg/dz\ df/dw$$

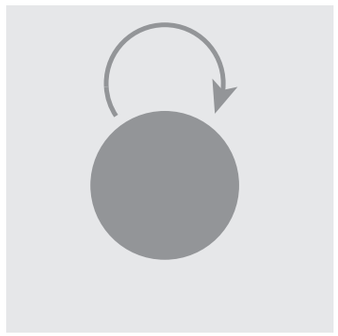
2. Post-hoc, simply add together gradients that are tied together!

Applications: image captioning

Vision
Deep CNN



Language
Generating RNN



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

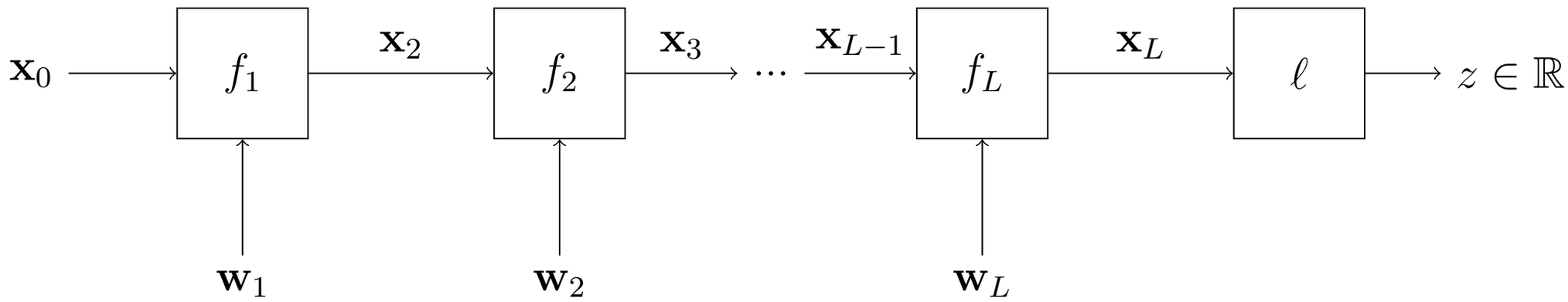
Applications: video analysis

Strangely, state-of-the-art deep features for video analysis not that much better than previous hand-designed counterparts

Still an open research question!

Final difficulties in training

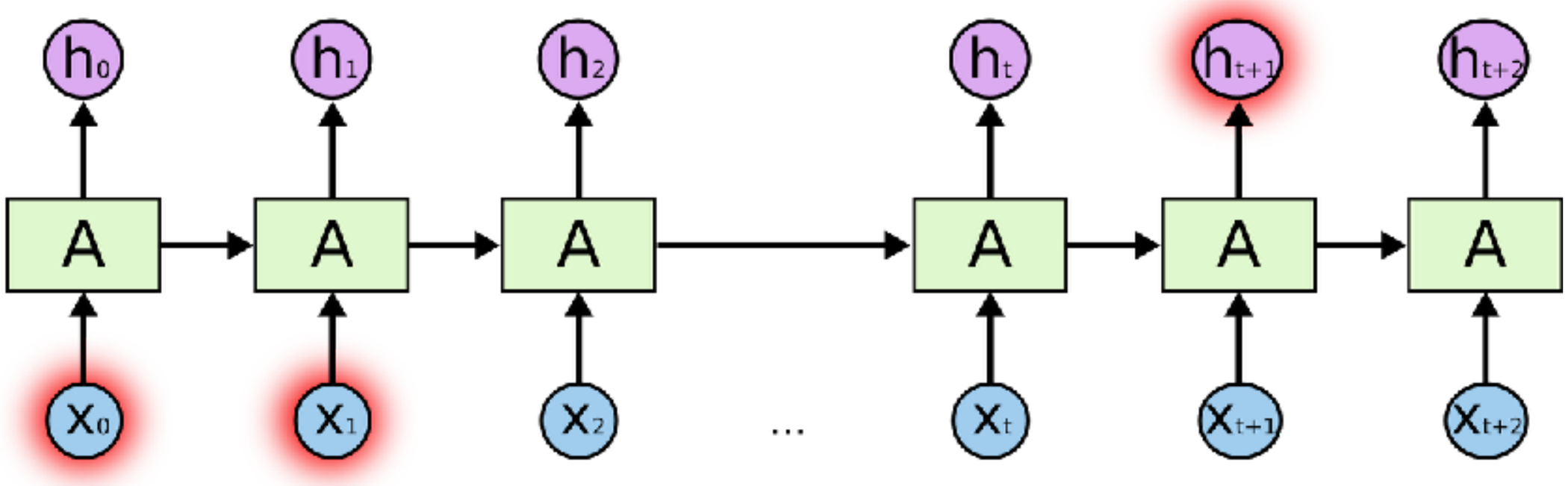
Vanishing or exploding gradients imply lower layers can be hard to learn
(would presumably be less of an issue with second-order optimization)



$$\frac{dz}{d\mathbf{w}_l} = \frac{dz}{d(\operatorname{vec}\mathbf{x}_L)^\top}\frac{d\operatorname{vec}\mathbf{x}_L}{d(\operatorname{vec}\mathbf{x}_{L-1})^\top}\cdots\frac{d\operatorname{vec}\mathbf{x}_{l+1}}{d(\operatorname{vec}\mathbf{x}_l)^\top}\frac{d\operatorname{vec}\mathbf{x}_l}{d\mathbf{w}_l^\top}$$

My intuition: multiscale skip connections and weight-tying (recurrence) alleviate this

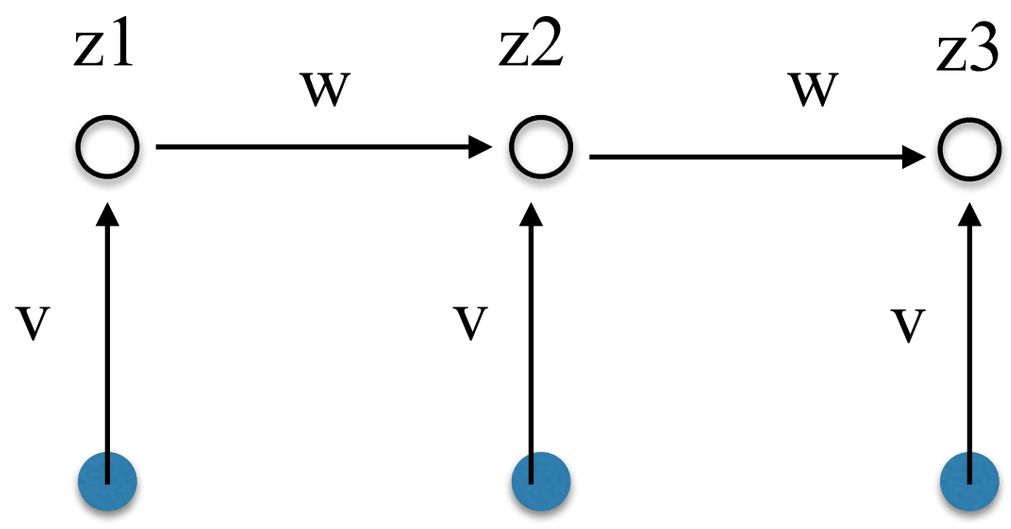
Long-range dependancies



Consider sequences where the next word (or video frame) is effected by an observation long ago

(e.g., a person walked into a building 10 minutes ago)

Approach



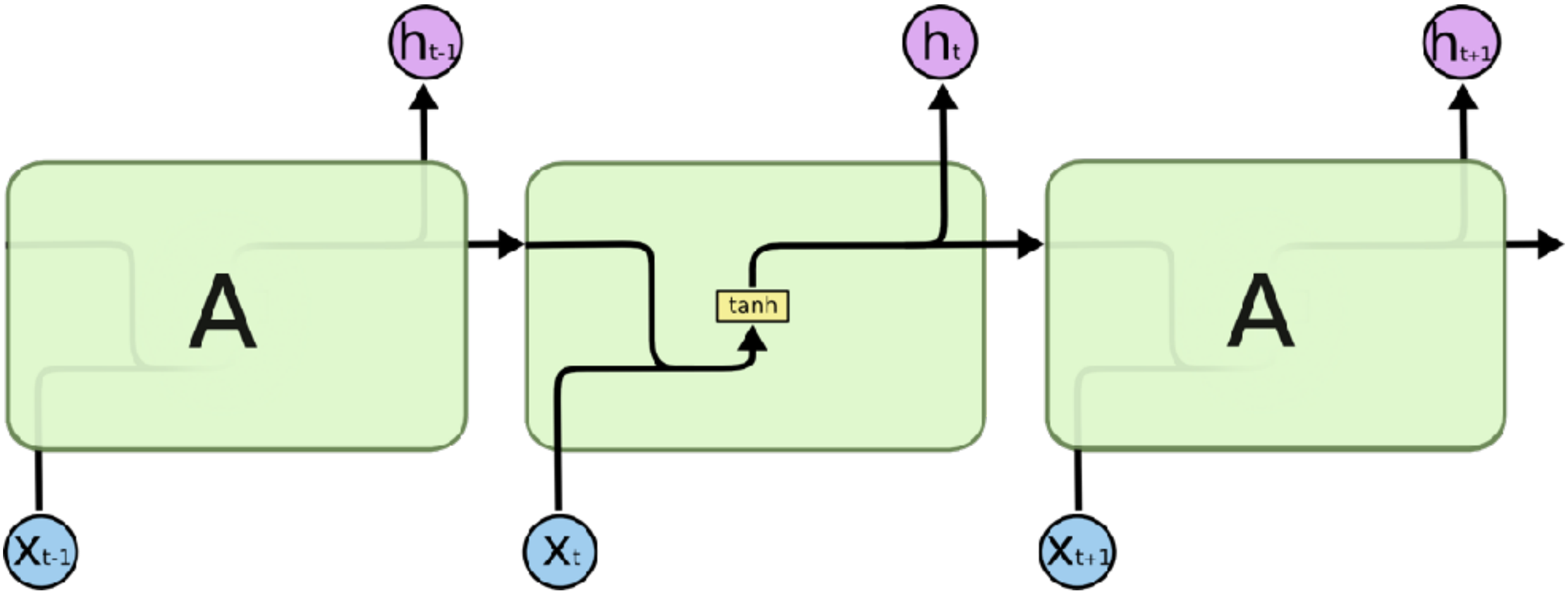
Add functionality to switch between $z_2 = wz_1 + vx_2$ (normal update)

$z_2 = vx_2$ (forget past)

$z_2 = z_1$ (remember)

To do so, *multiply* in sigmoidal activations that act as gates (returning 0 or 1)

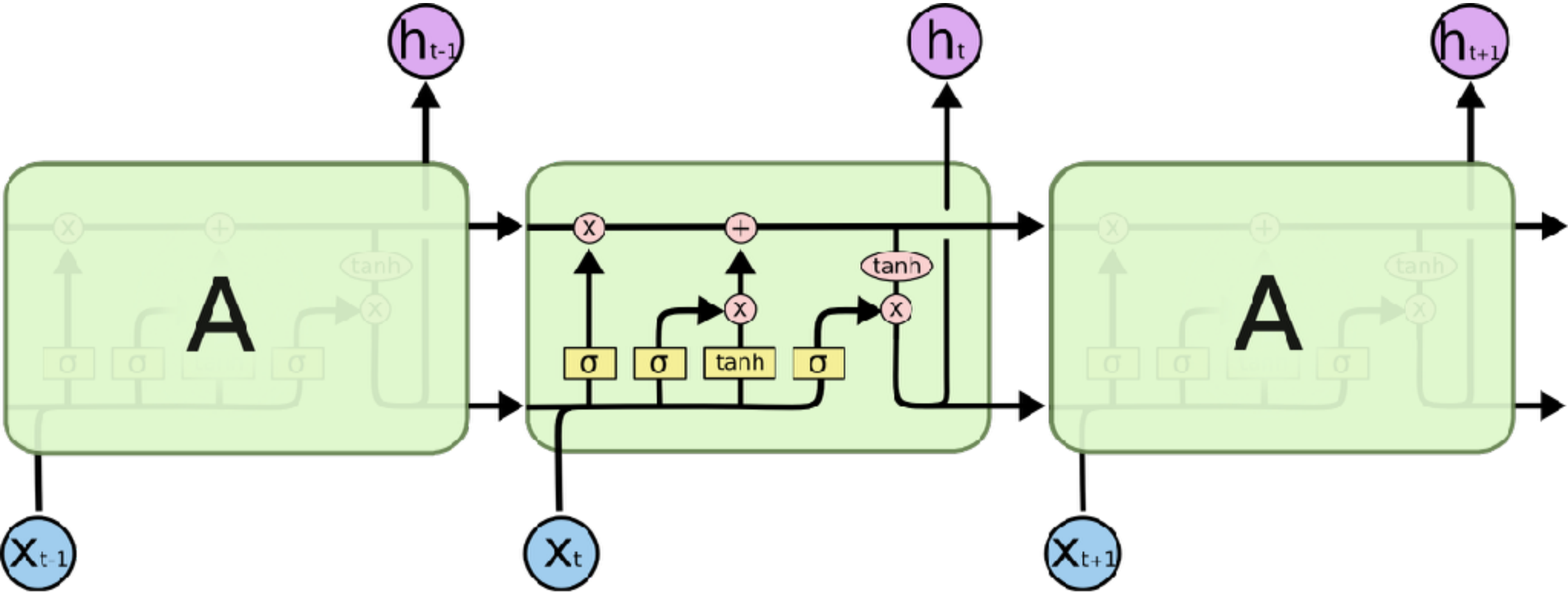
A closer look



Recurent models typically make use of tanh nonlinearities
(unbounded nonlinear functions tend to allow activations to grow over time)

LSTMs

Long short-term memory



Still kind of mysterious to me…

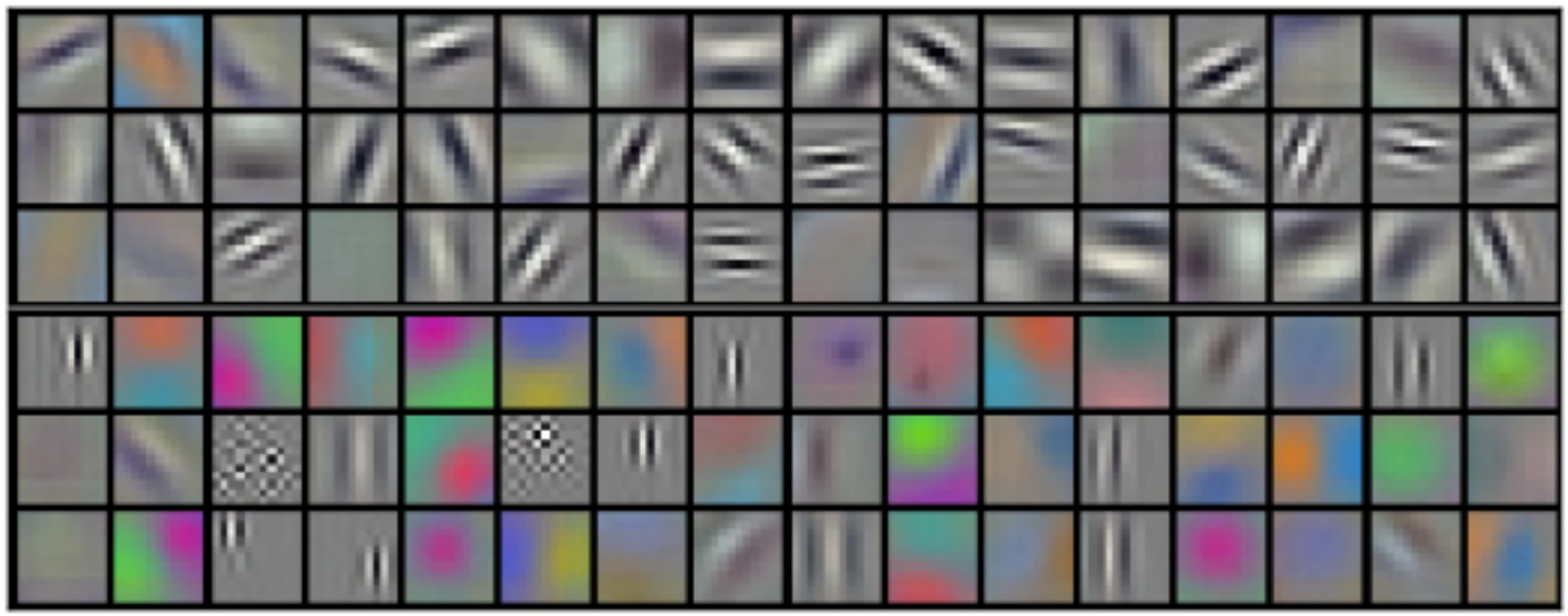
Outline

- Logistics
- Motivation
- Training
- Why does it work so well?
- Extensions

Why does it work so well?

Visualizations
Distributed representations
Compositional representations

Visualizations



First-layer 11x11 filters

Visualizaing other layers

Look for image patches that maximally excite particular activations

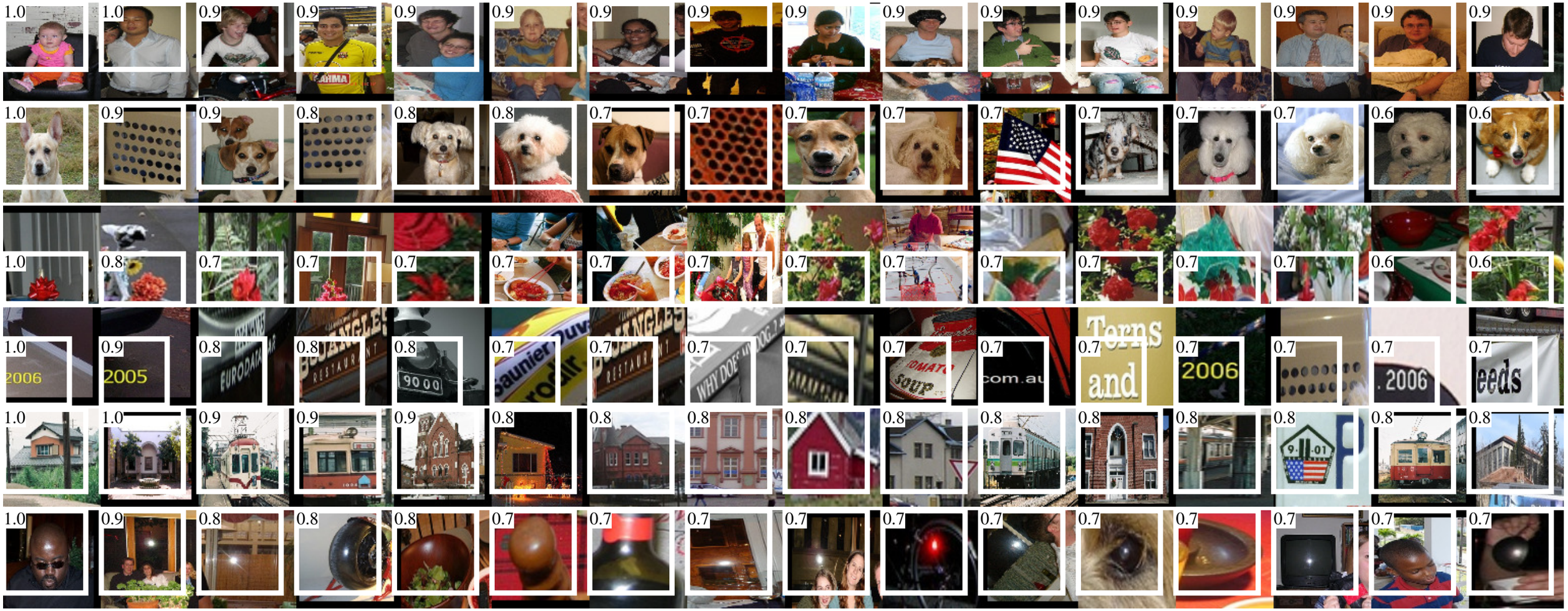
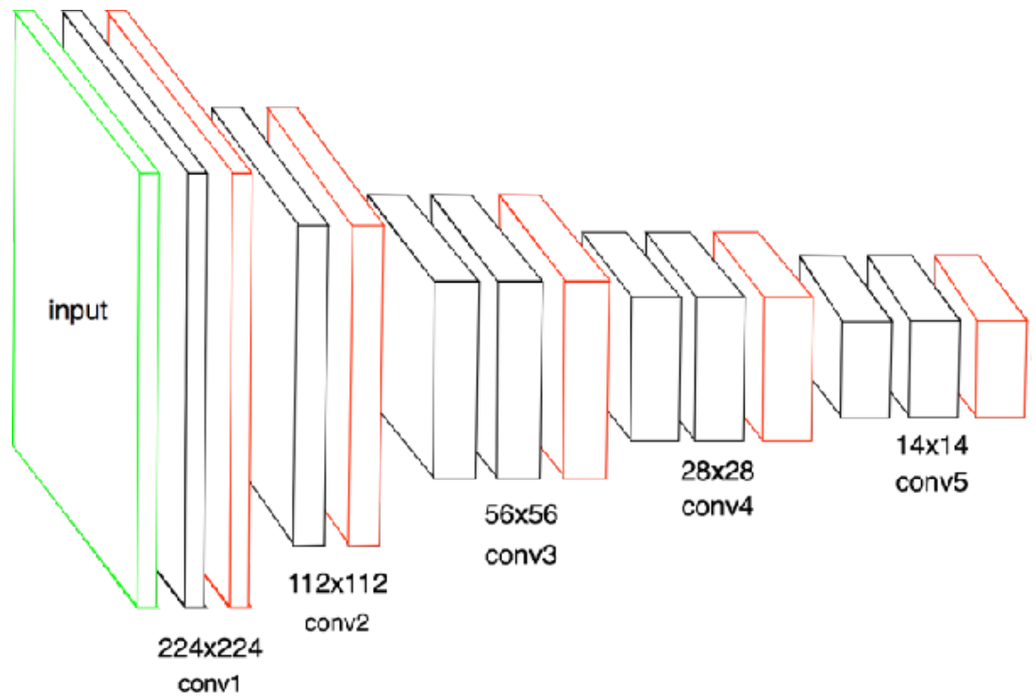


Figure 3: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

Fun with visualizations

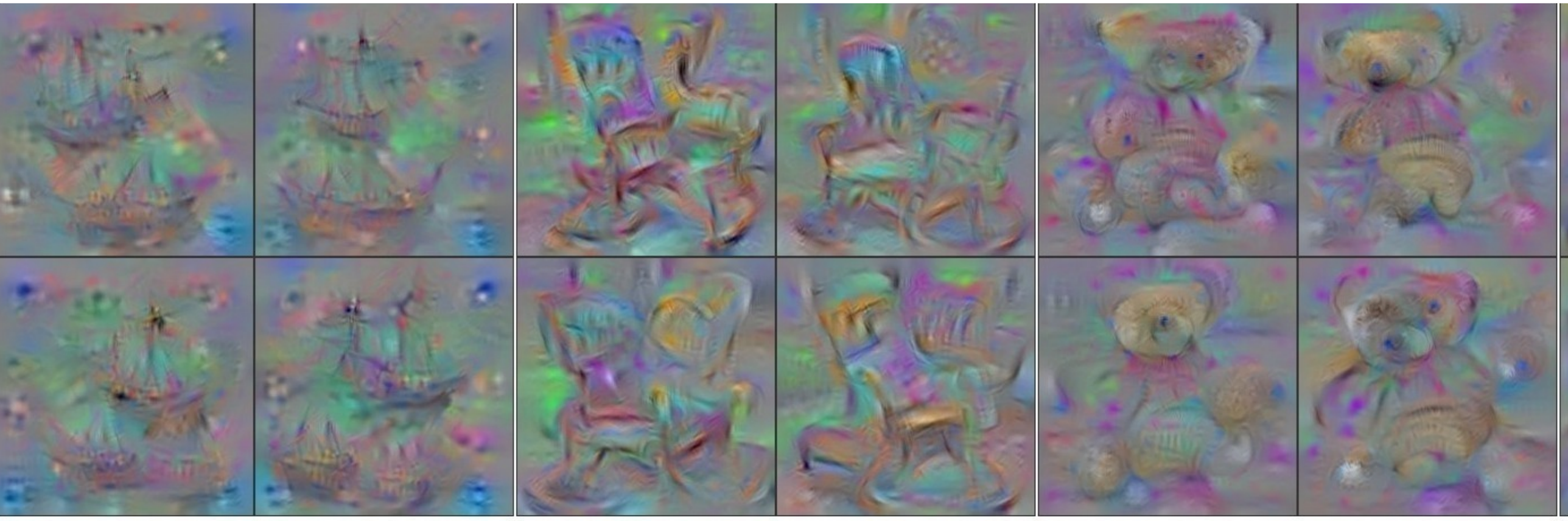
Perform backprop all the way to the pixel-level and update pixels so as to maximize activations



http://yosinski.com/deepvis#toolbox

Fun with visualizations

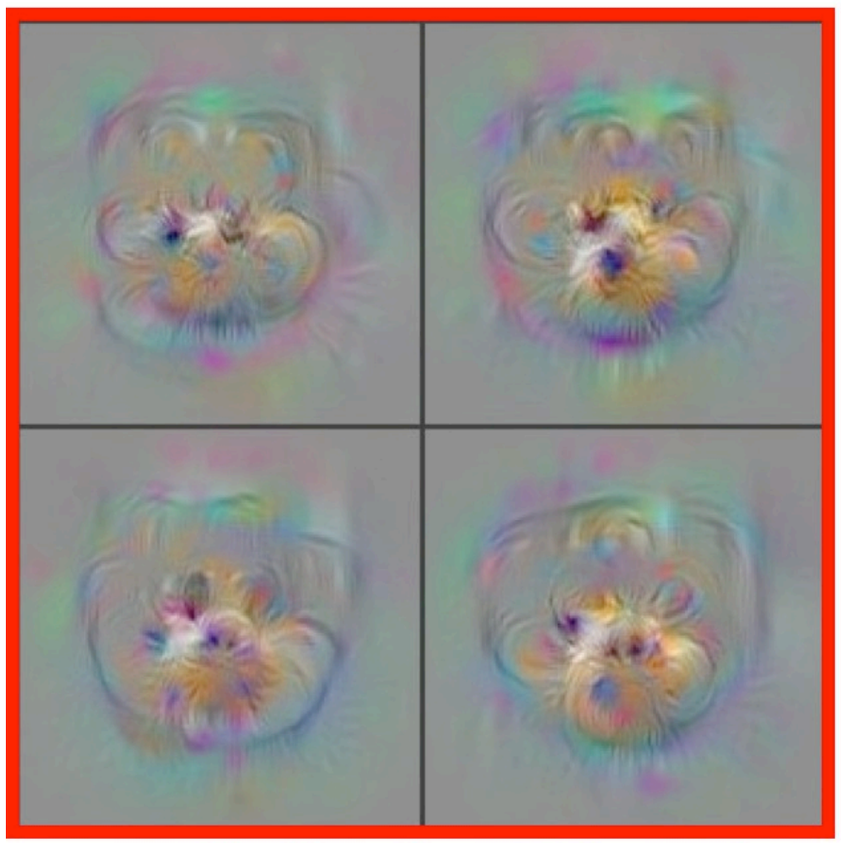
Optimize pixels for output activations



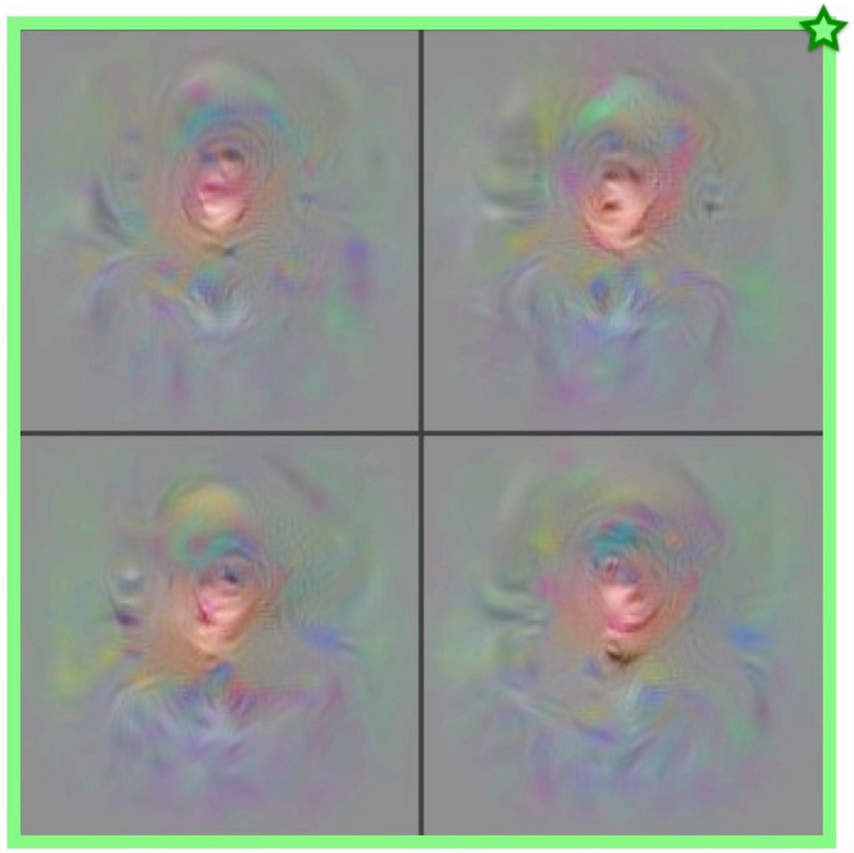
Pirate Ship | Rocking Chair | Teddy Bear

Fun with visualizations

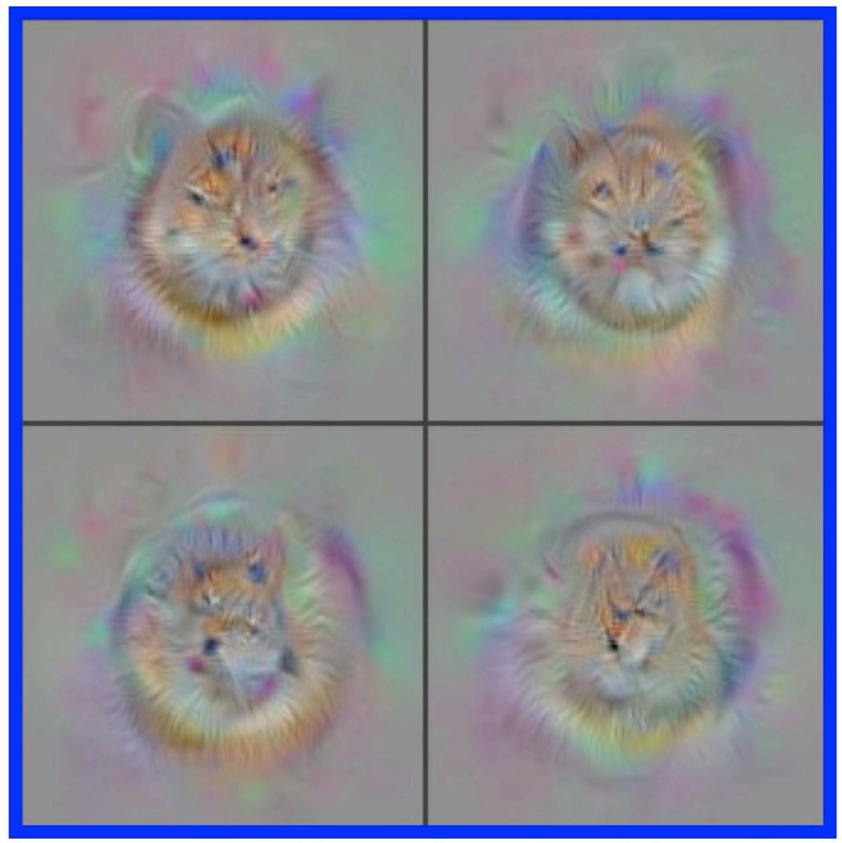
Optimize pixels for intermediate activations



$conv5_2$ (dog face + flower)



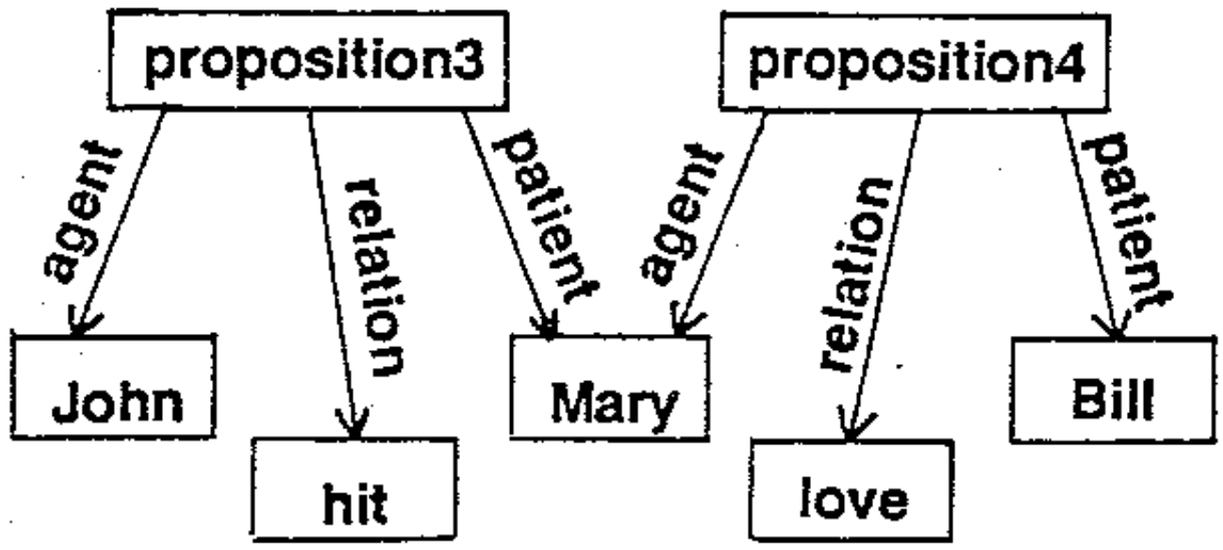
$conv5_{151}$ (human face + cat face)



$conv5_{111}$ (cat face)

GOFAI

"good-old fashioned AI"

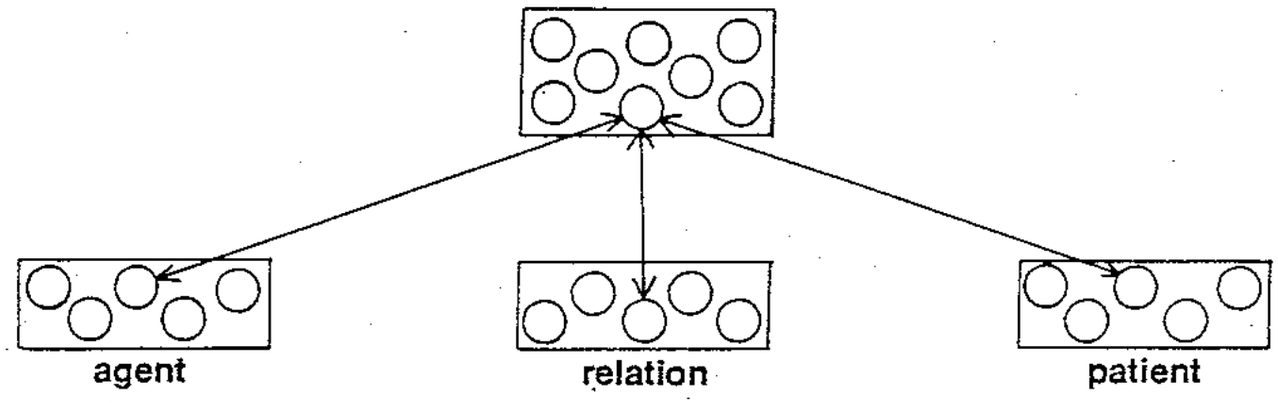


Typical knowledge-base of discrete concepts (eg, propositional logic)

LEARNING DISTRIBUTED REPRESENTATIONS OF CONCEPTS

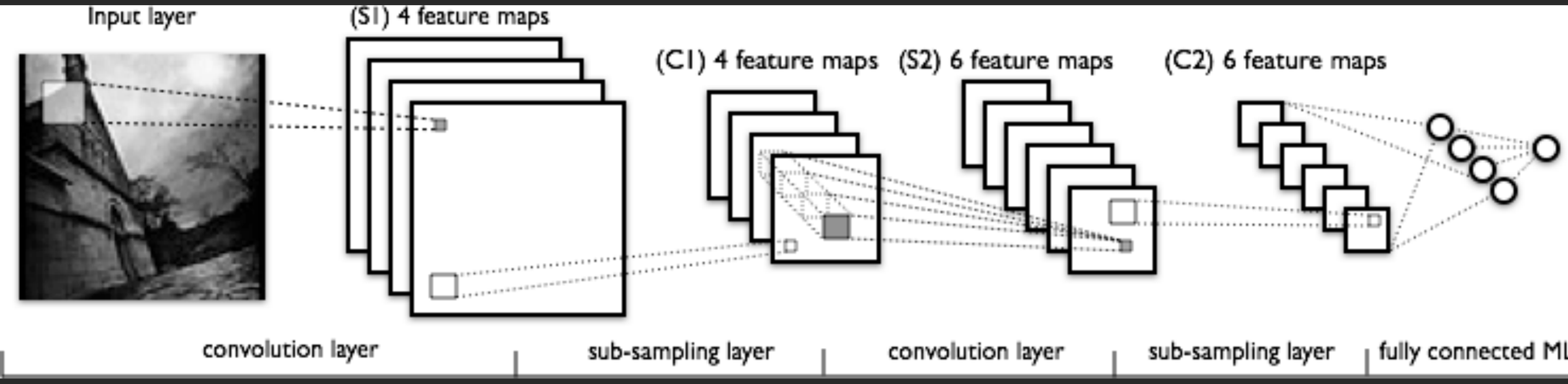
Geoffrey E. Hinton
Computer Science Department
Carnegie-Mellon University

"For example, if you learn that chimpanzees like onions, you will probably raise your estimate of the probability that gorillas like onions. If you subsequently learn that gibbons and orangutans do not like onions, your estimate of the probability that gorillas that like onions will fall, though it may be higher than it was originally"

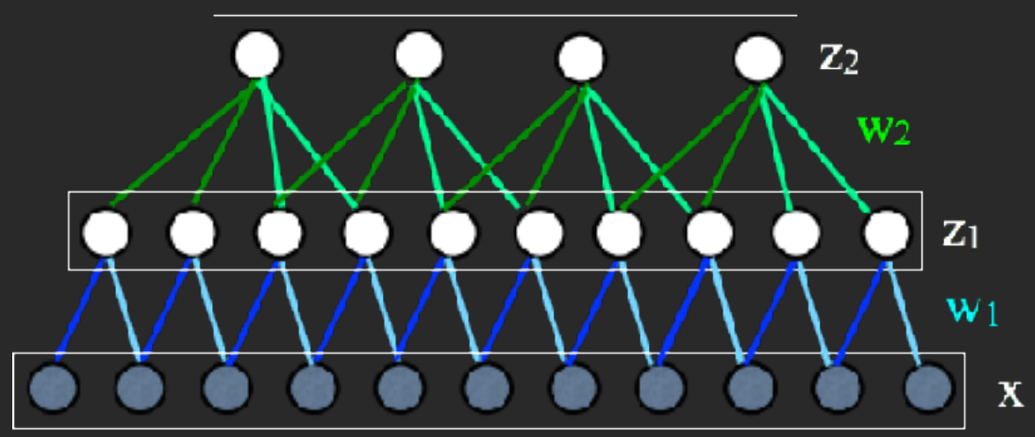


This chapter describes one type of representation that is less familiar and harder to think about than local representations. Each entity is represented by a pattern of activity distributed over many computing elements, and each computing element is involved in representing many different entities. The strength of this more complicated kind of representation does not lie in its notational convenience or its ease of implementation in a conventional computer, but rather in the efficiency with which it makes use of the processing abilities of networks of simple, neuron-like computing elements.

Compositional perspectives on deep learning

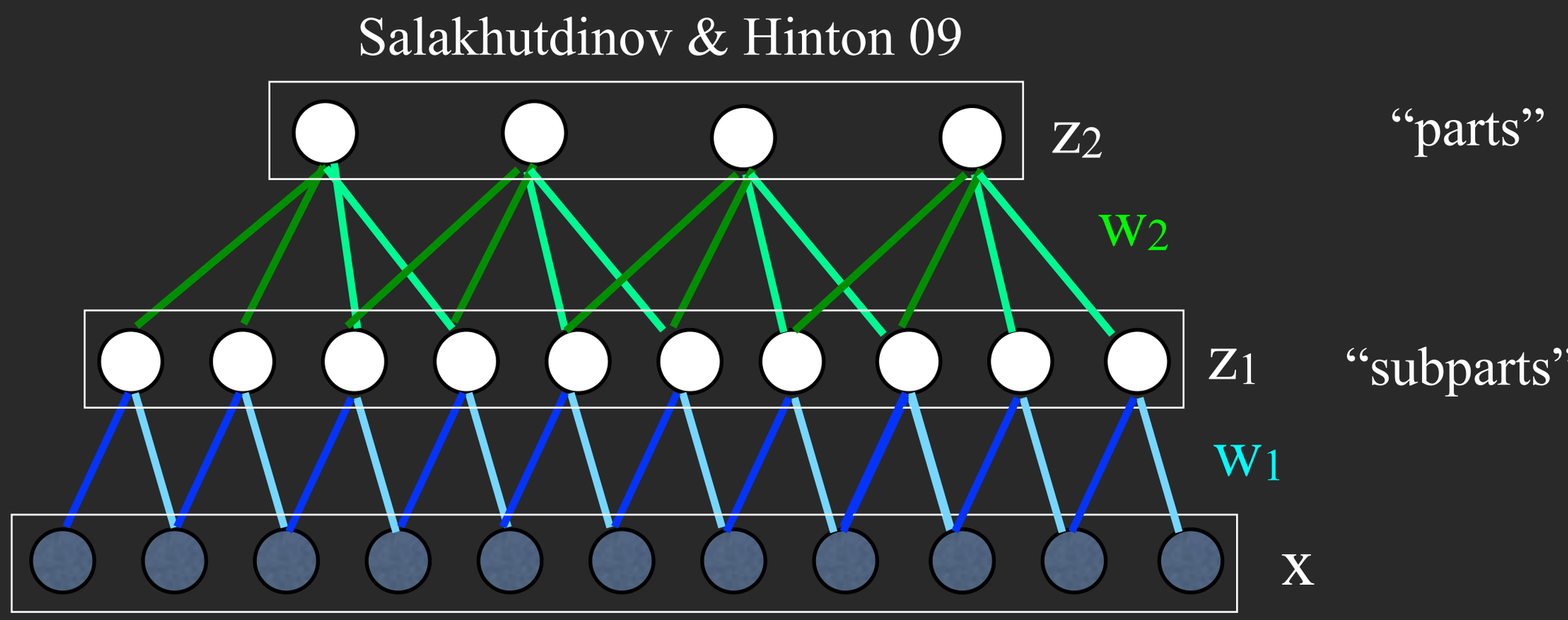


Convolutional Neural Nets (CNNs). Lecun et al



Boltmann machines. Hinton et al

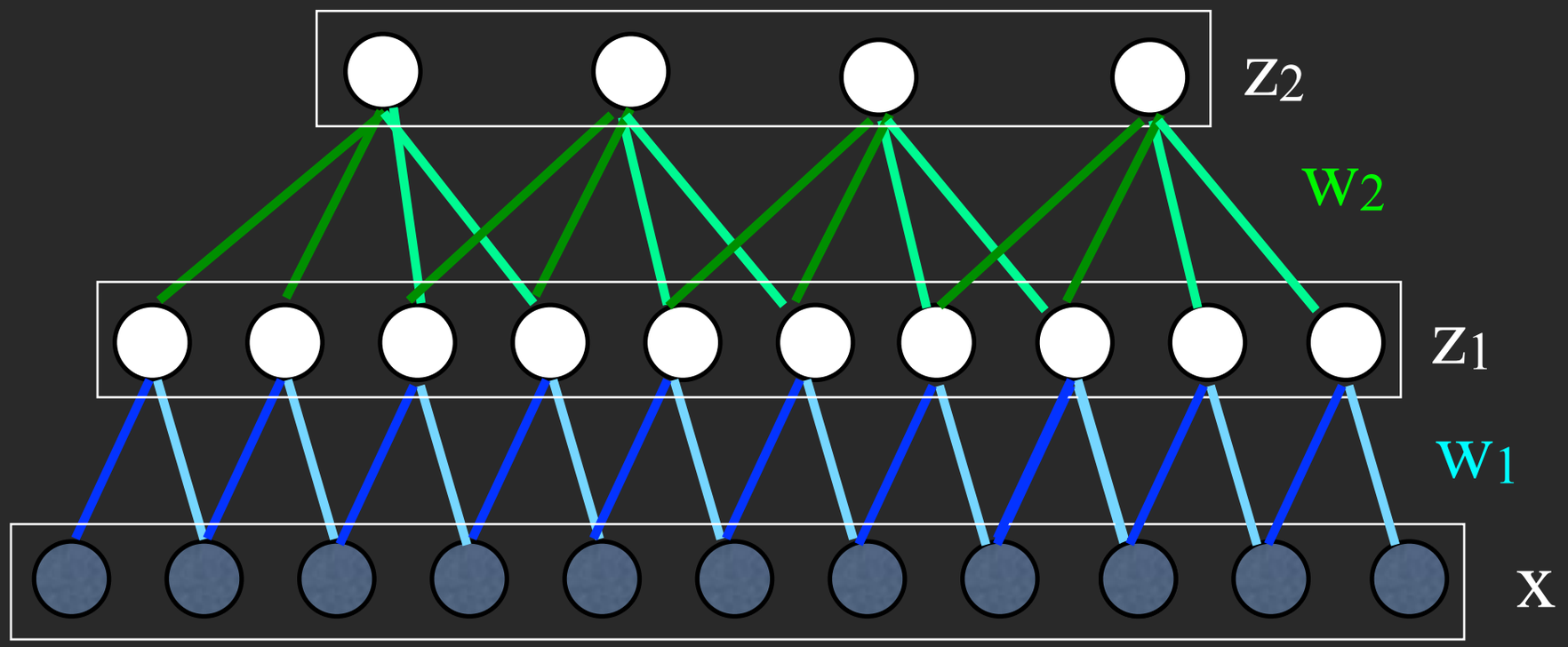
Deep Boltzmann Machines as deep latent-variable models



Binary latent variables: is there a (person, head, oriented edge) at a particular location(?)

1. Latent variables allow for sharing of information across a large collection of templates
2. Models can synthesize new combinations of latent variables

Deep Boltzmann Machines

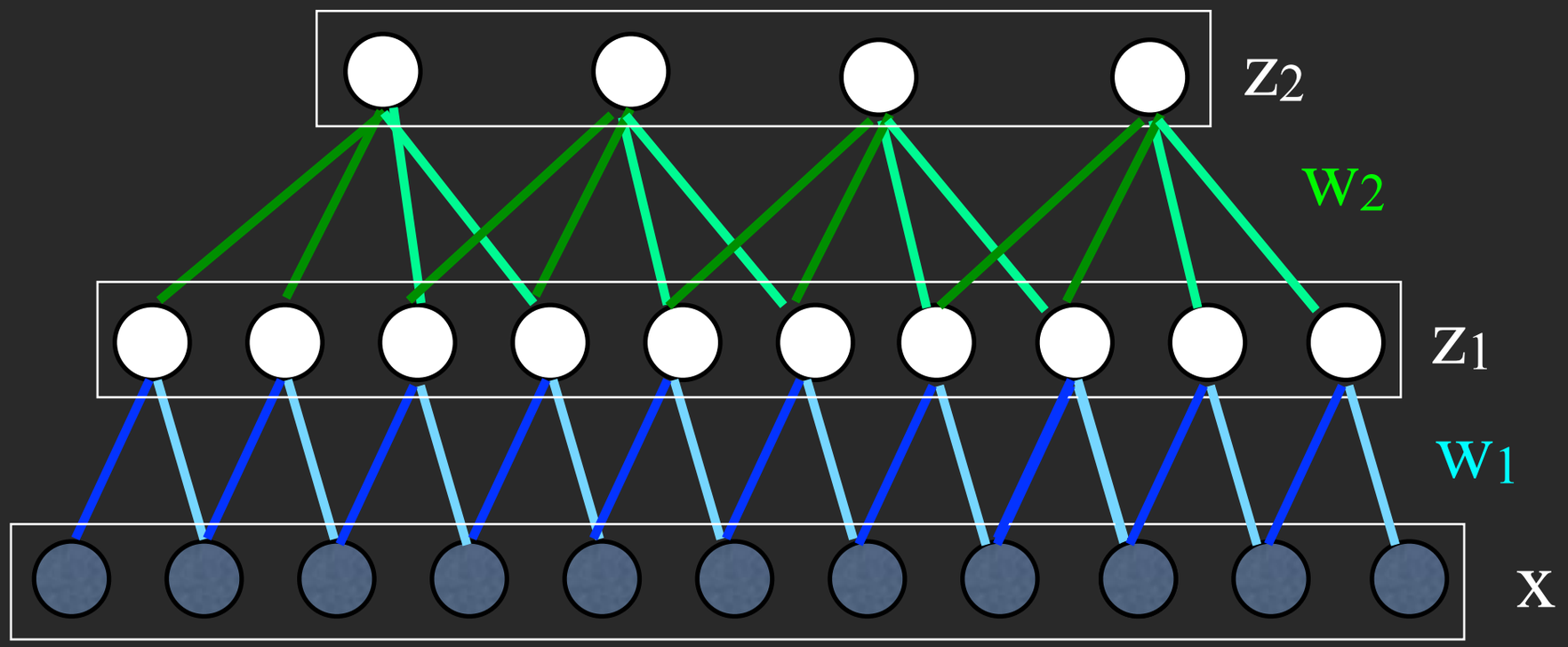


The messy details:

$$S_{Boltz}(x,z) = \frac{1}{2}x^T W_1 z_1 + \frac{1}{2}z_2^T W_2 z_2 + \ldots + b^T z, \quad z_i \in \{0,1\}$$

$$P(x,z) \propto e^{S_{Boltz}(x,z)}$$

Notation: hierarhical latent-variable models

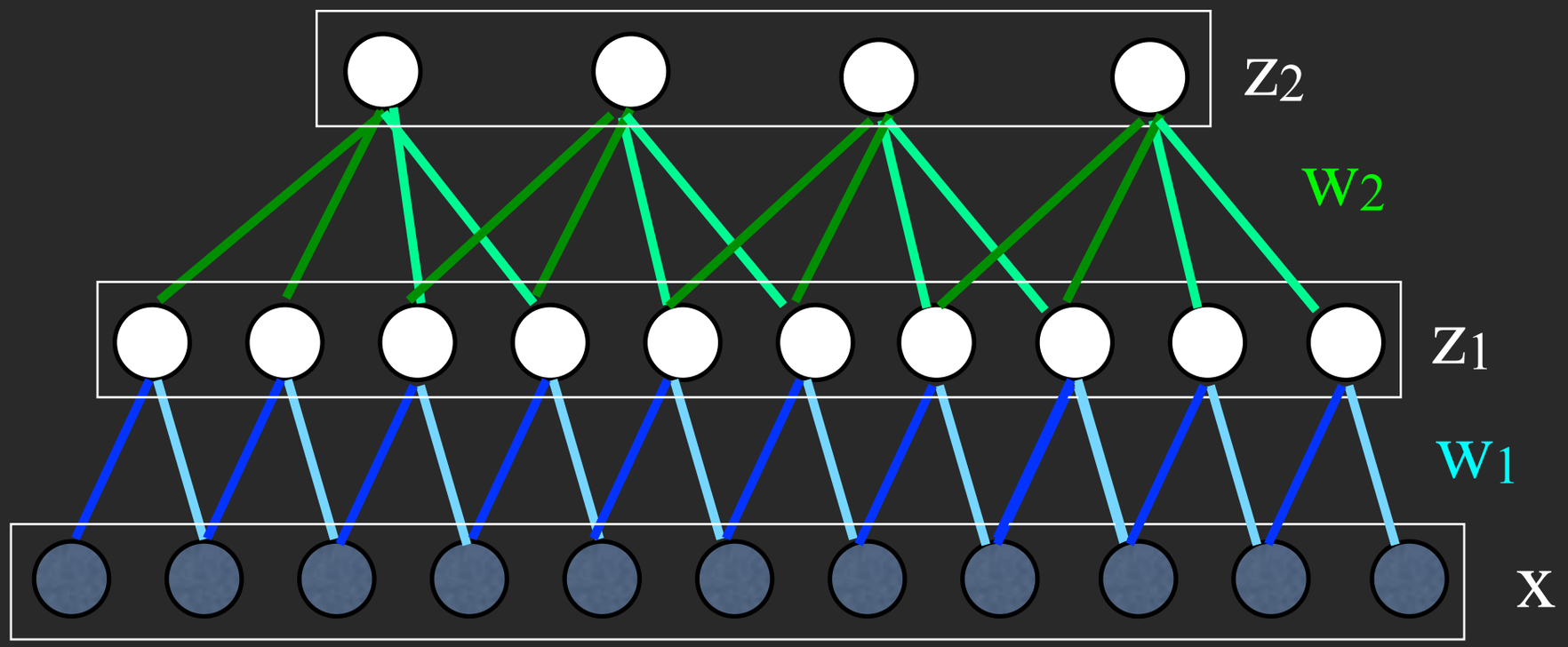


Let $z_0 = x$

$$P(z) \propto e^{S(z)} \quad \text{where} \quad S(z) = \frac{1}{2} z^T W z + b^T z$$

Boltzmann: $z_i \in \{0, 1\}$

Notation: hierarhical latent-variable models



Let $z_0 = x$

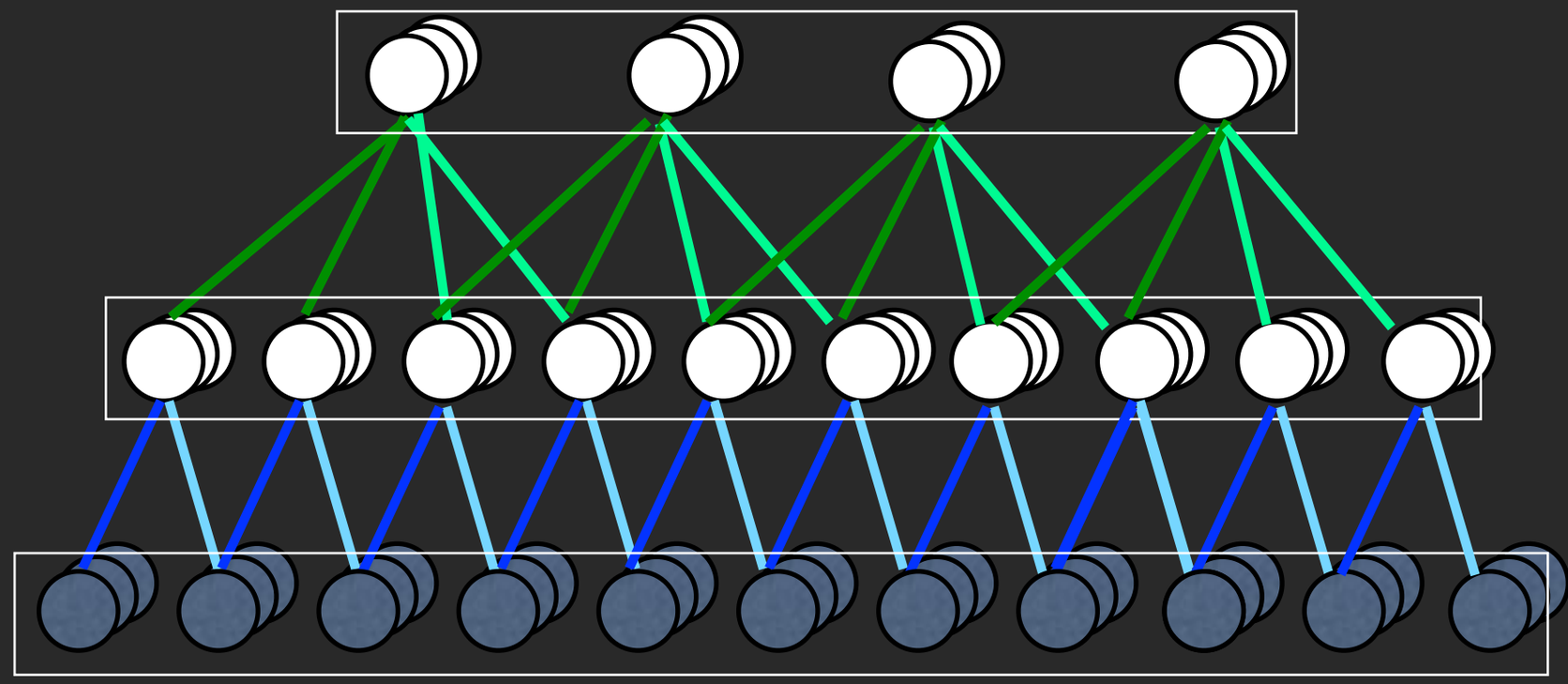
$$P(z) \propto e^{S(z)} \quad \text{where} \quad S(z) = \frac{1}{2} z^T W z + b^T z$$

Boltzmann: $z_i \in \{0, 1\}$

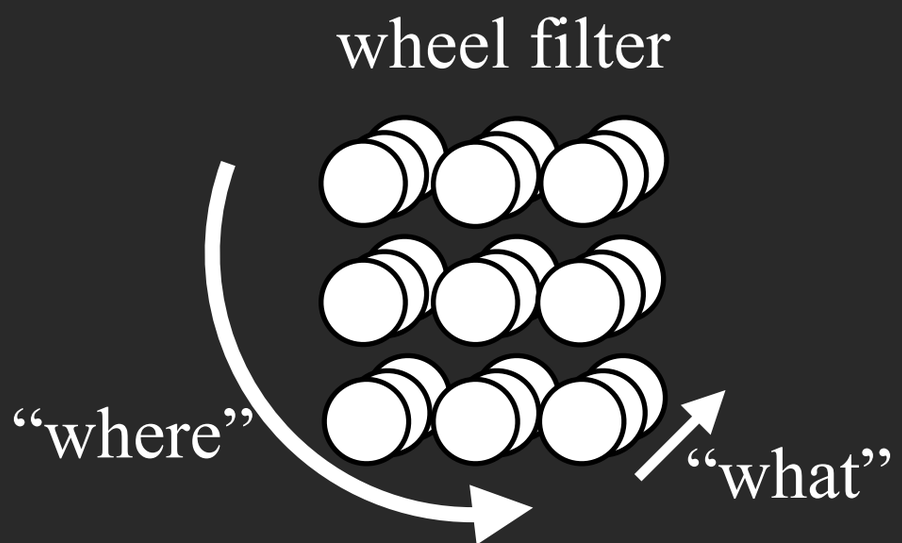
Gaussian: $z_i \in R$

Convolutional Boltzmann Machines

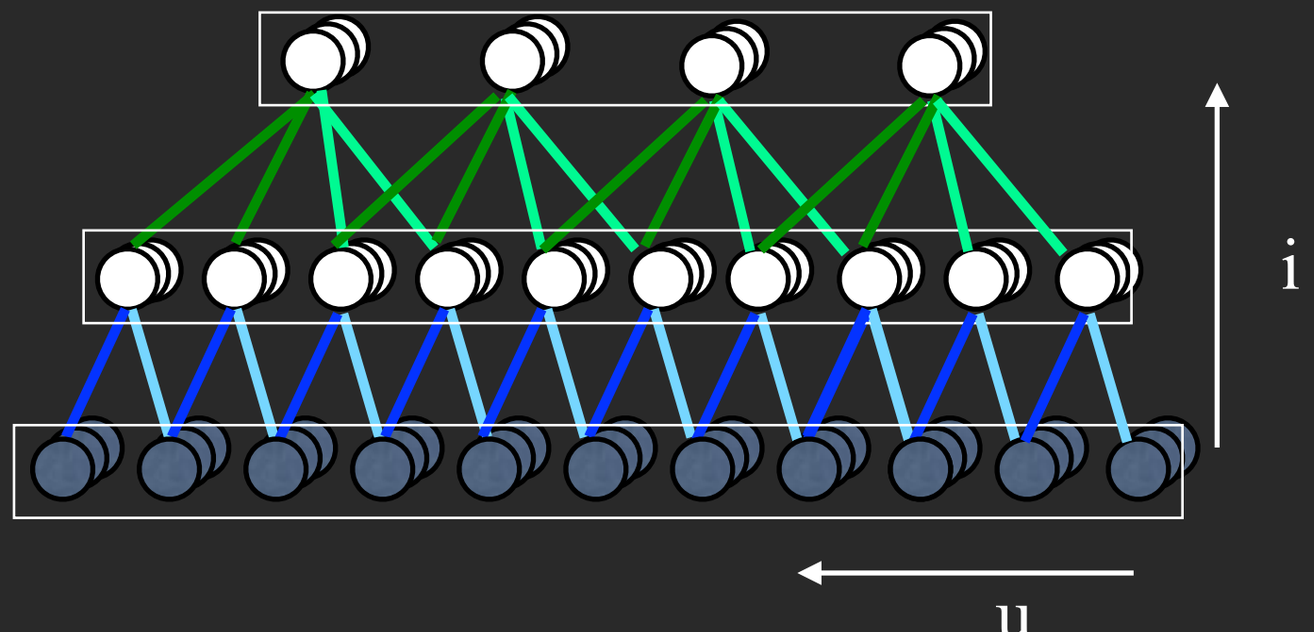
Lee et al, ICML 09



The activation of a binary variable given lower-level binary features are given by a *filter*



Convolutional Boltzmann Machines



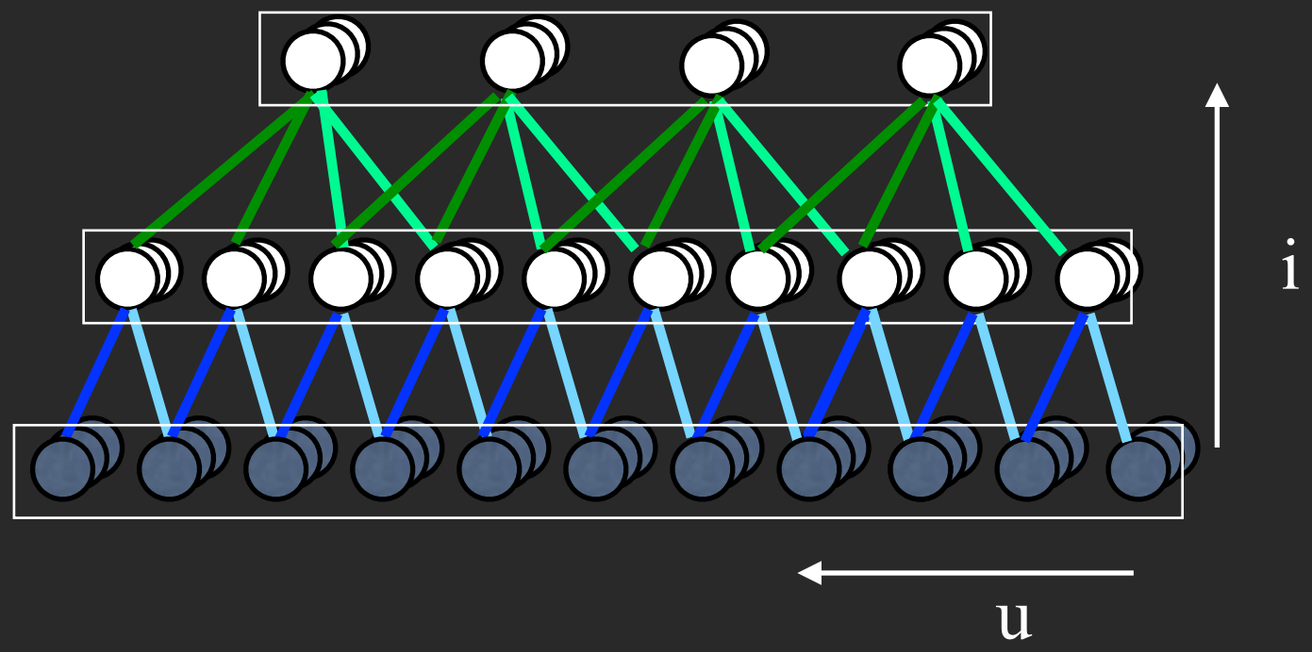
Inference: compute P(x|z)

Hard to do

Option 1 (Gibbs sampling): Iteratively update $z_i[u]$ by computing P($z_i[u]$| all else)

(Hinton)

Gibbs sampling



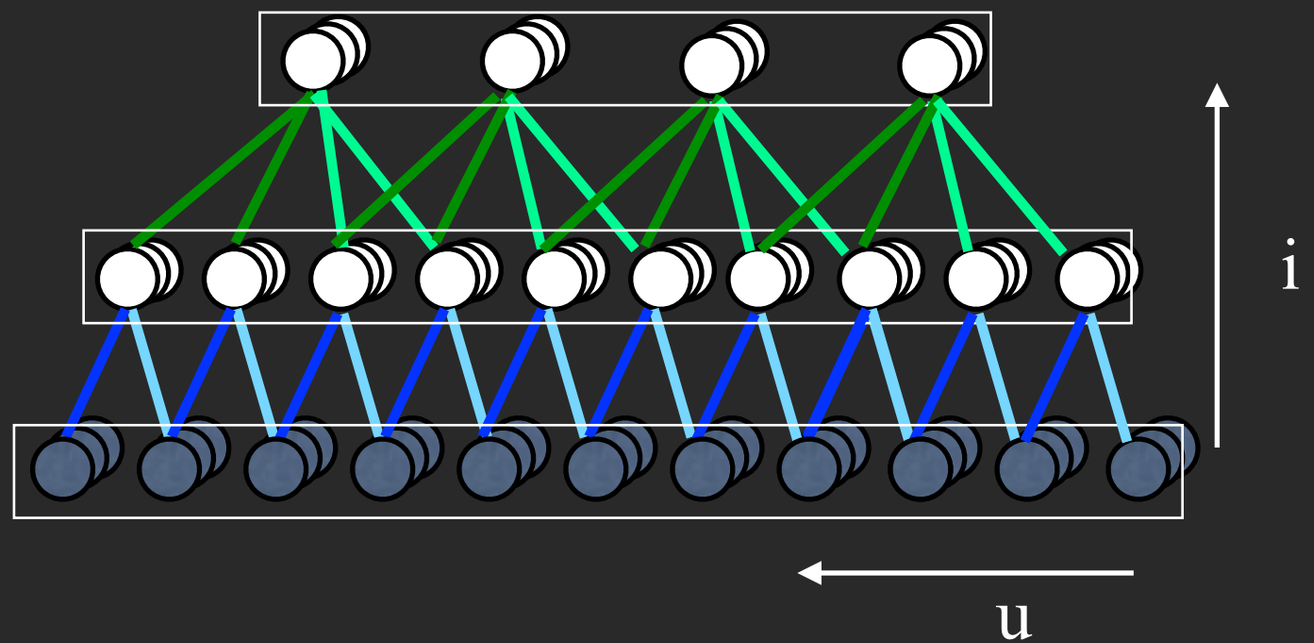
Iteratively update $z_i[u]$ by computing $P(z_i[u]|$ all else$)$

$$z_i[u] \sim \text{sigmoid}(b_i + top_i[u] + bot_i[u])$$

$$top_i[u] = \sum_v w_{i+1}[v] z_{i+1}[u - v] \qquad \text{“correlation”}$$

$$bot_i[u] = \sum_v w_i[v] z_{i-1}[u + v] \qquad \text{“convolution”}$$

Mean feild updates



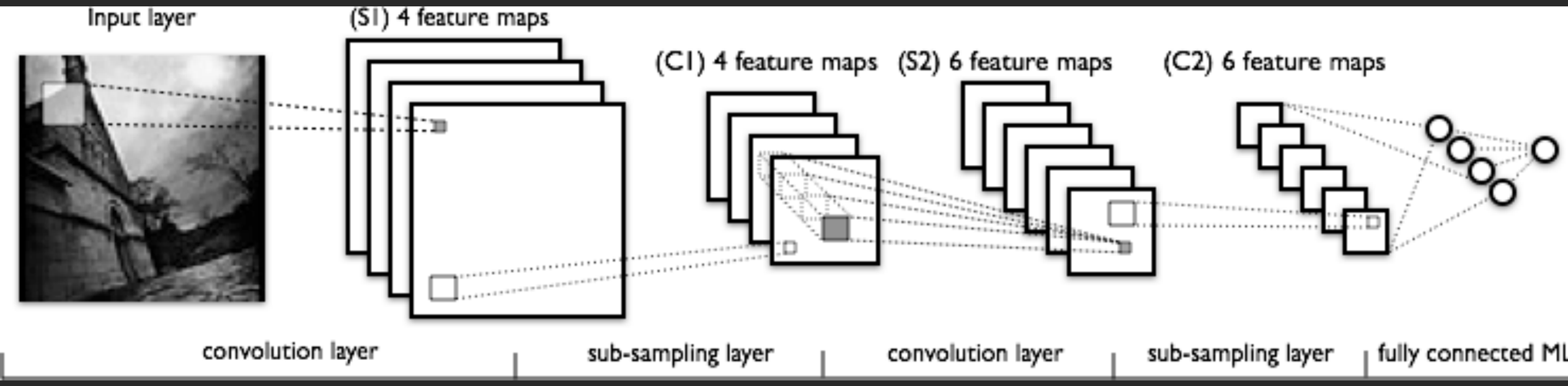
(Salakhutdinov & Hinton)

$$z_i[u] = \text{sigmoid}(b_i + top_i[u] + bot_i[u])$$

$$top_i[u] = \sum_v w_{i+1}[v] z_{i+1}[u - v] \qquad \text{“correlation”}$$

$$bot_i[u] = \sum_v w_i[v] z_{i-1}[u + v] \qquad \text{“convolution”}$$

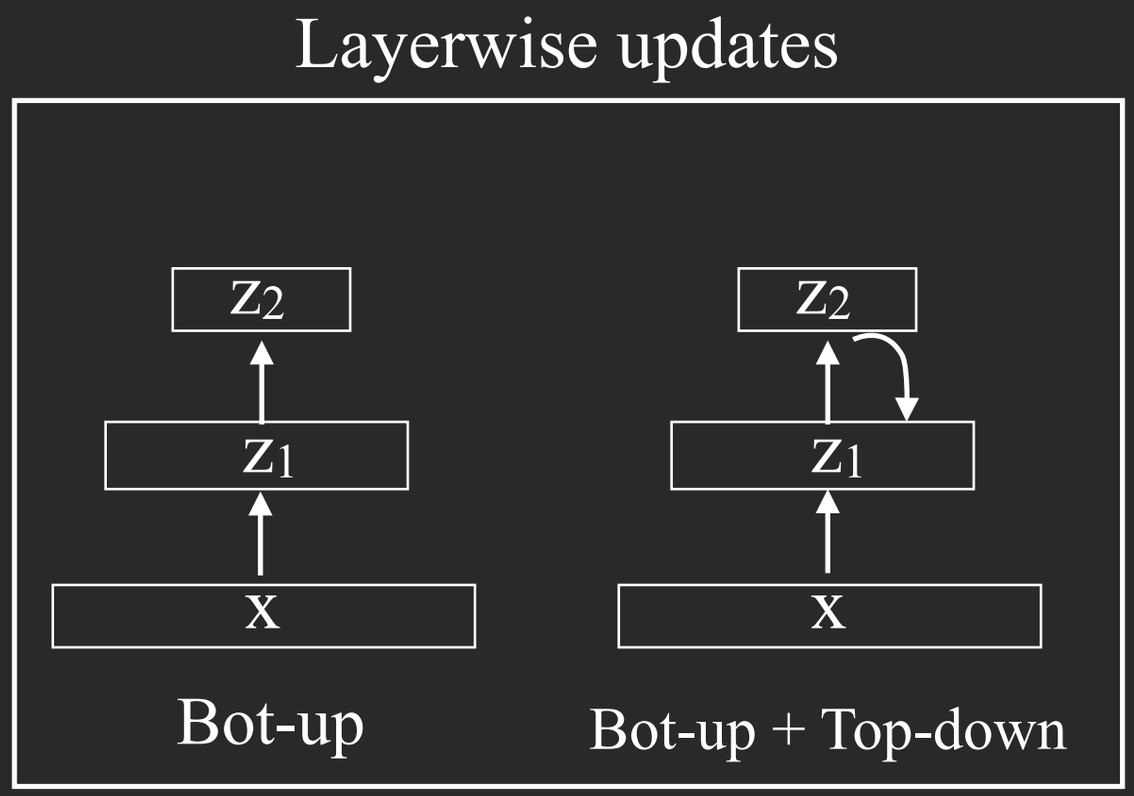
Claim: any sequence of updates can be written as a collection of filtering + nonlinear operations

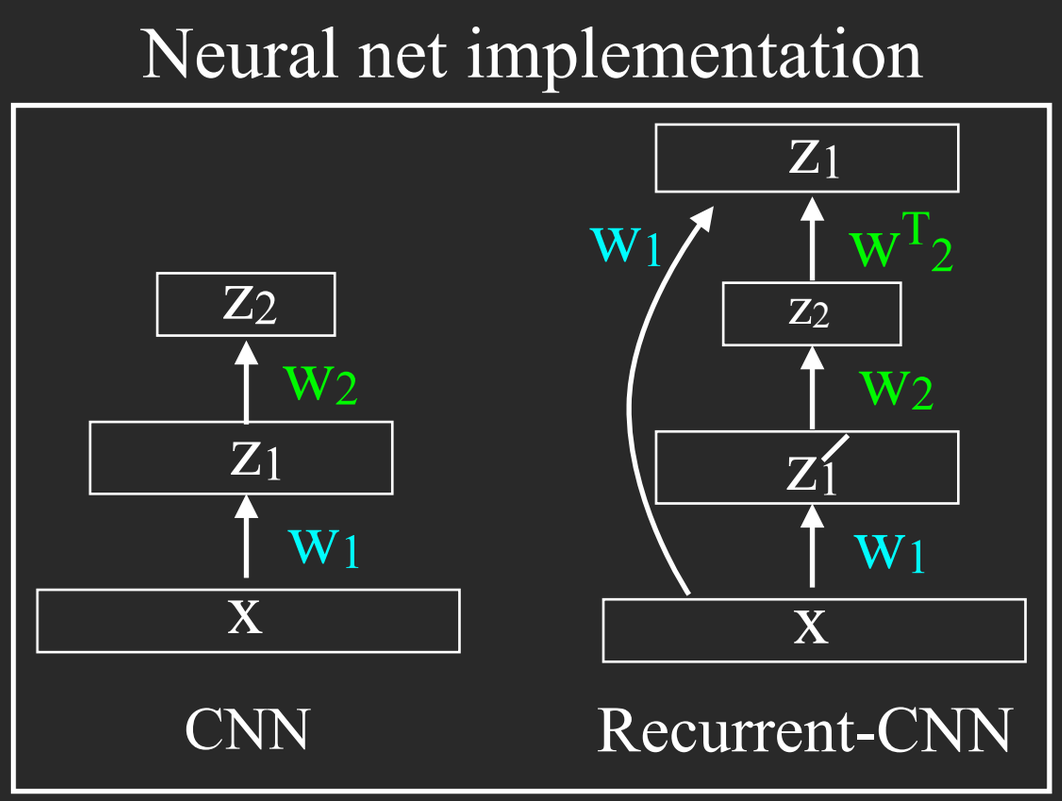


Convolution + sigmoidal activations mimic computations on a binary latent-variable model

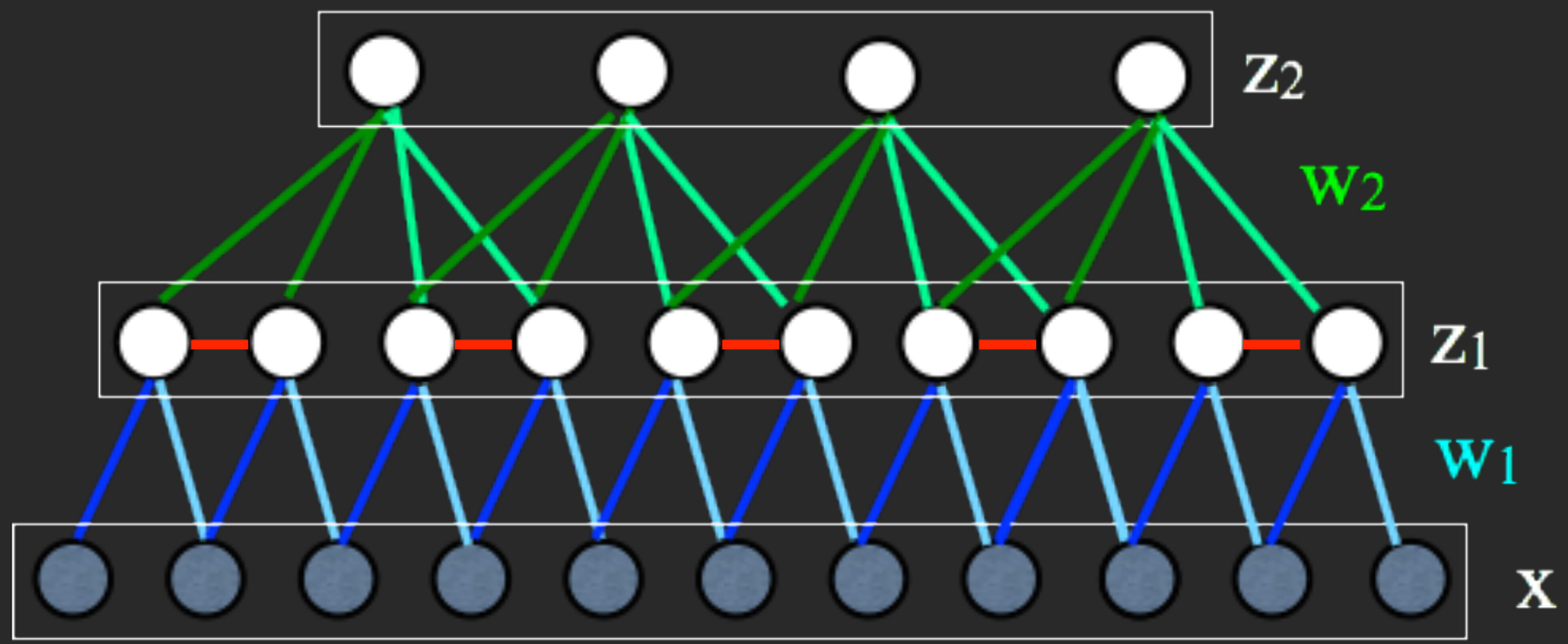
Use CNNS to learn to infer on Boltzmann machines

1. Use mean-field inference rather than Gibbs sampling (Salakhutdinov & Hinton)
2. Unroll sequence of mean-field updates into a recurrent neural net (Goodfellow et al)





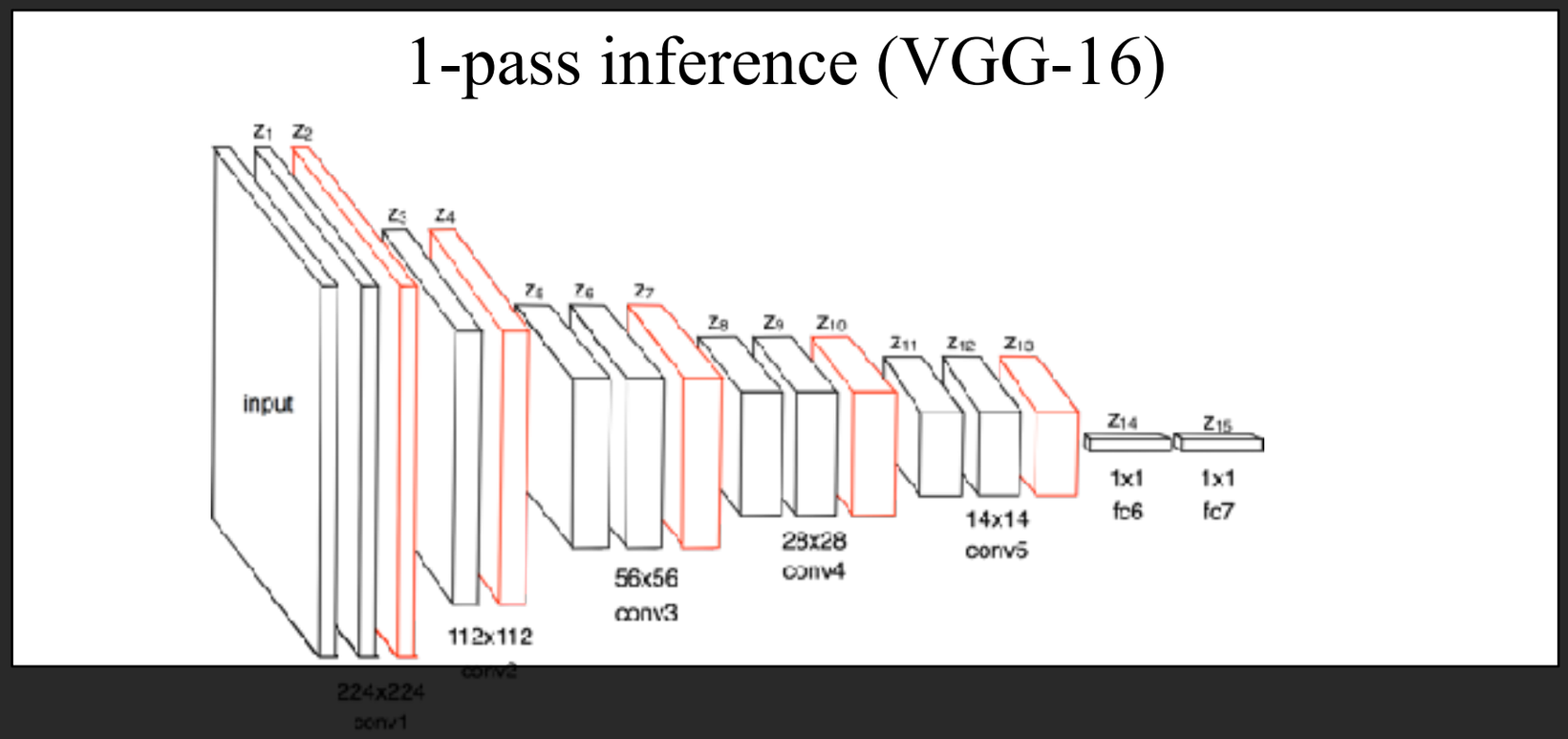
Top-down localization

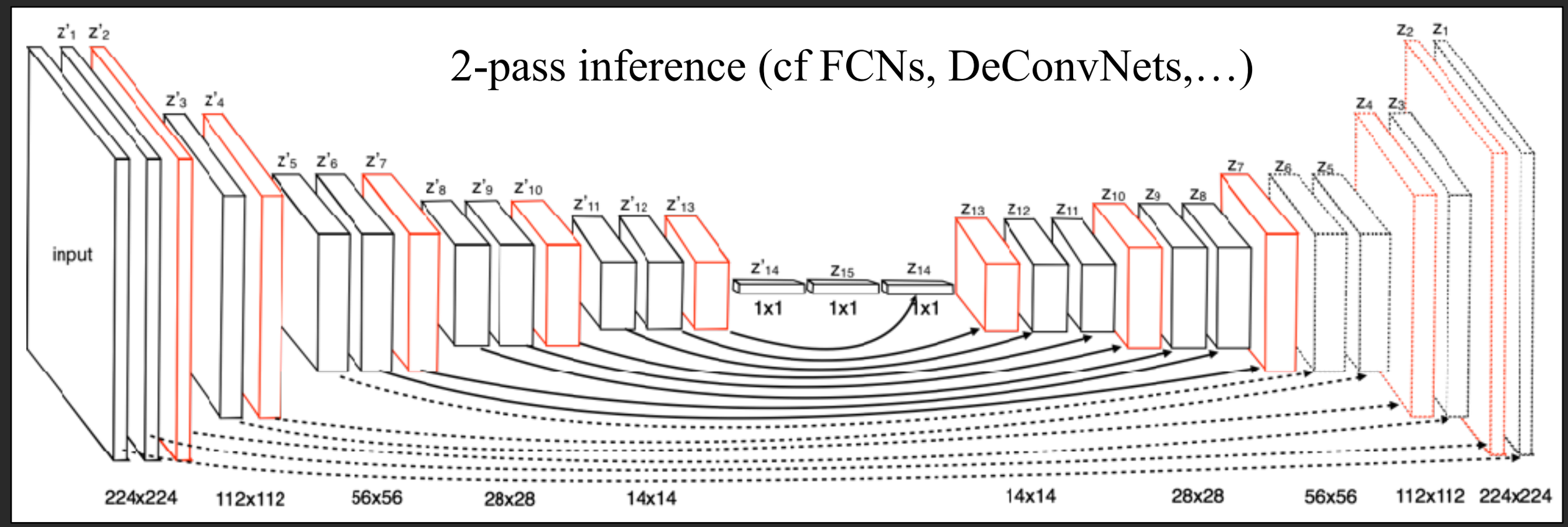


1. Model “max-pooling” using lateral inhibition connections (red edges)

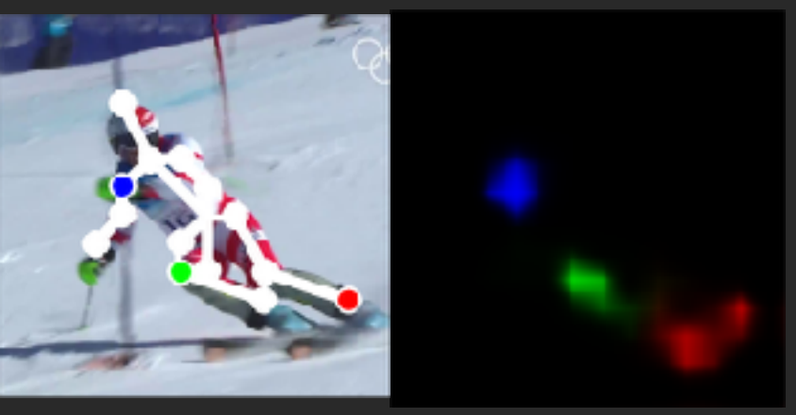
2. Above model allows for top-down localization
   e.g., a car “object” can influence the activation and location of a wheel “part”

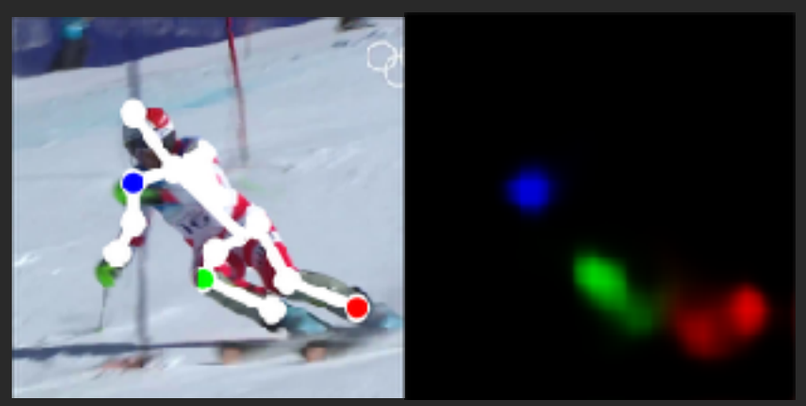
Implementation

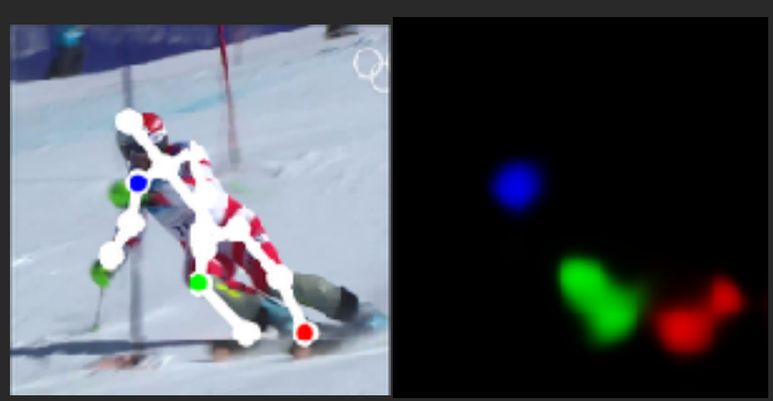




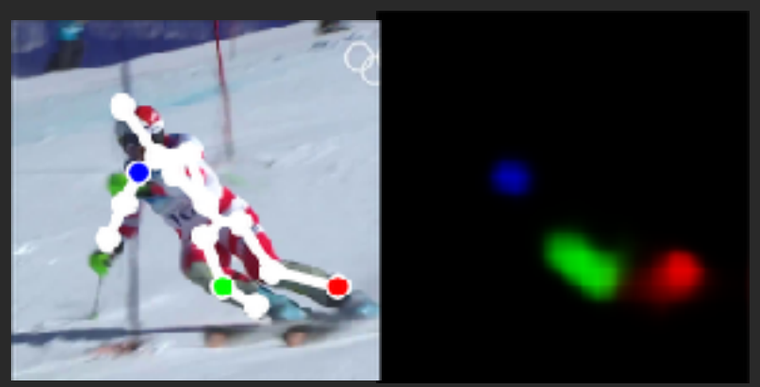
Coarse-to-fine

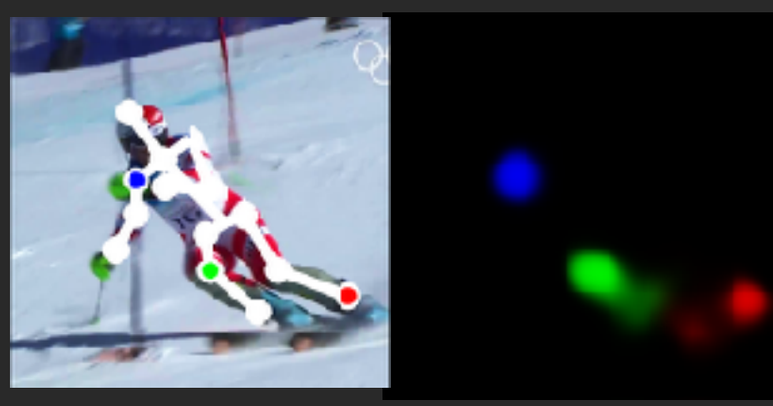






Bottom-up

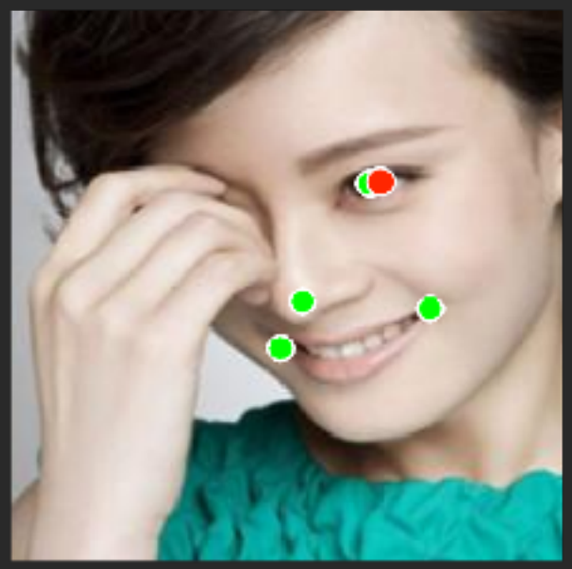




Top-down

Example results

Bottom-up

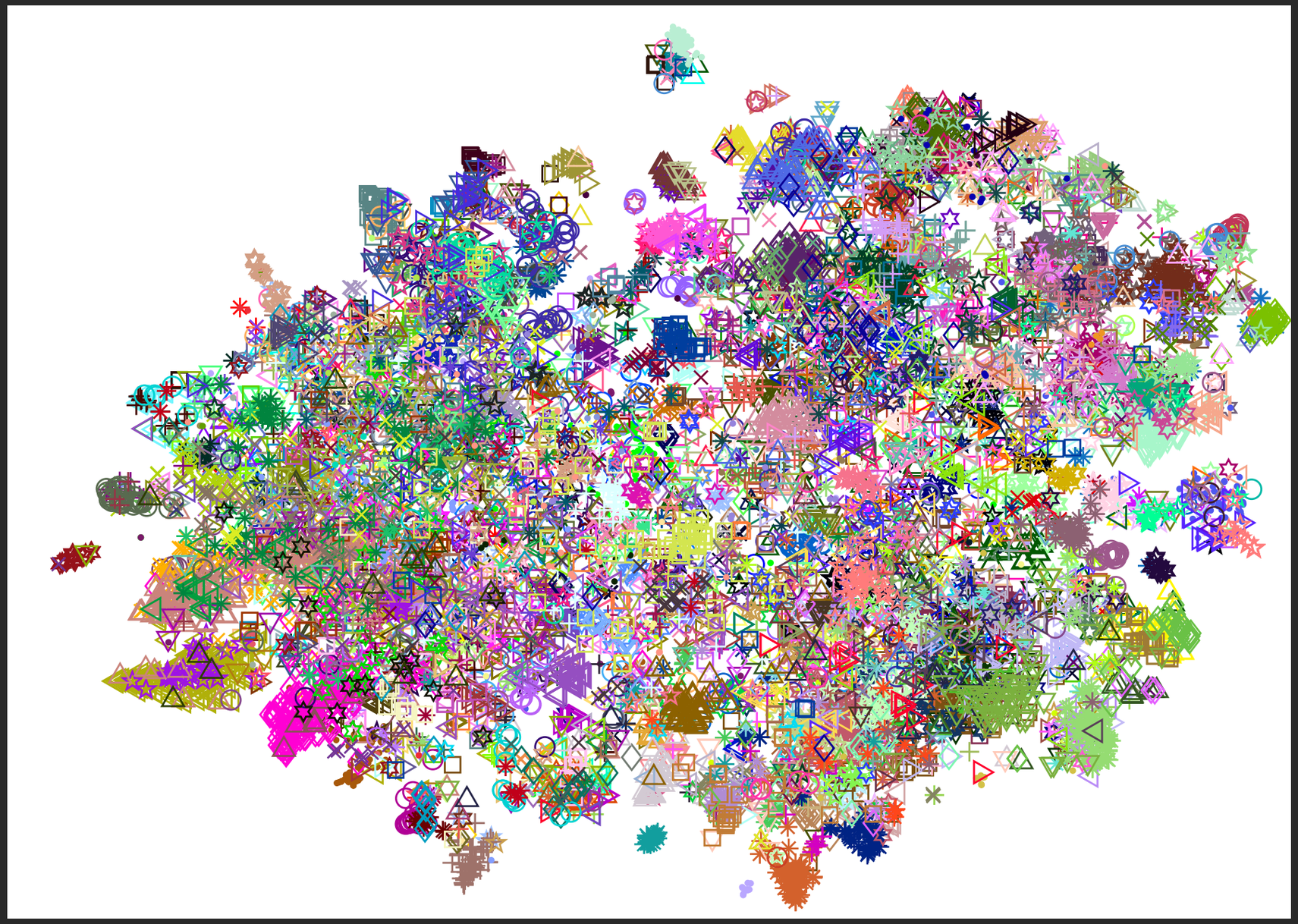



Alternate intepretation

$$= \begin{bmatrix} .1 \\ .4 \\ 0 \\ 1.2 \\ 0 \\ \vdots \end{bmatrix}$$

- Interpret the set of activations from fully-connected (FC7) layers as an embedding
- Apply standard embedding visualizations (MDS, TSNE)

Embedding visualizations



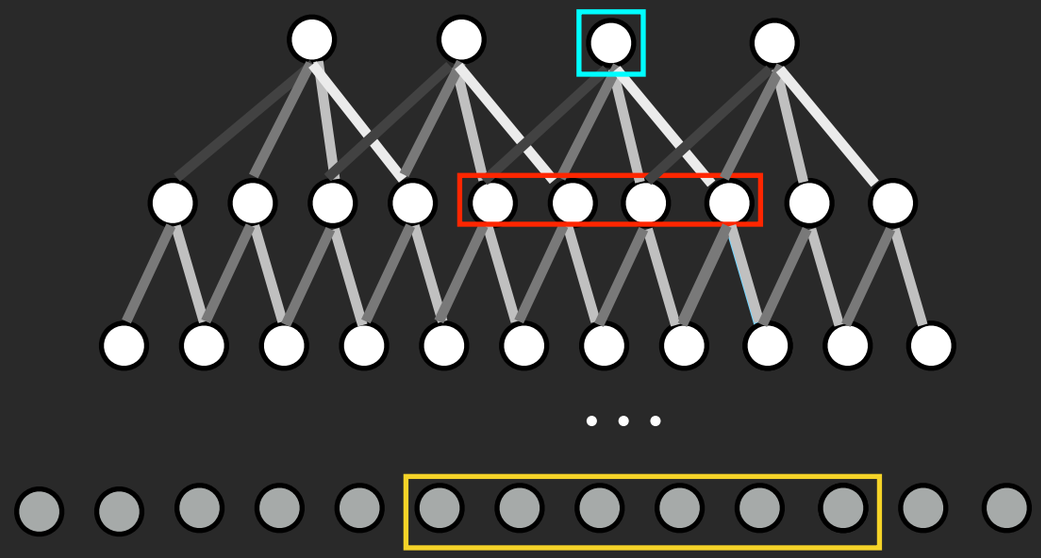
Colors correspond to classes

80k MS COCO images* in 5 minutes

Larry Zitnick

*All images are from the training set.

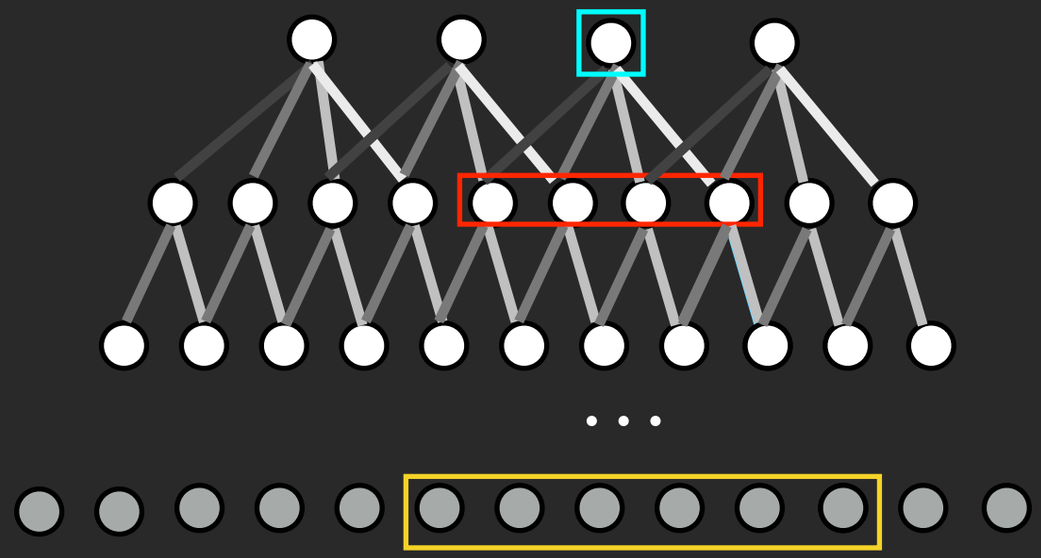
CNNs as hierarchical embeddings



Consider a particular neuron ‘*a*’ with a receptive feild of *NxN* pixels

$$a = \max(0, w \cdot x + b)$$

CNNs as hierarchical embeddings



Consider a particular neuron '*a*' with a receptive feild of *NxN* pixels

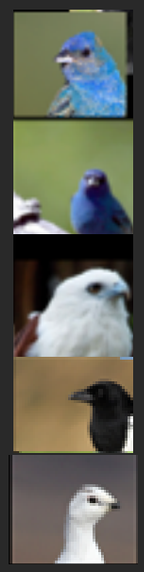
$$a = \max(0, w \cdot x + b)$$

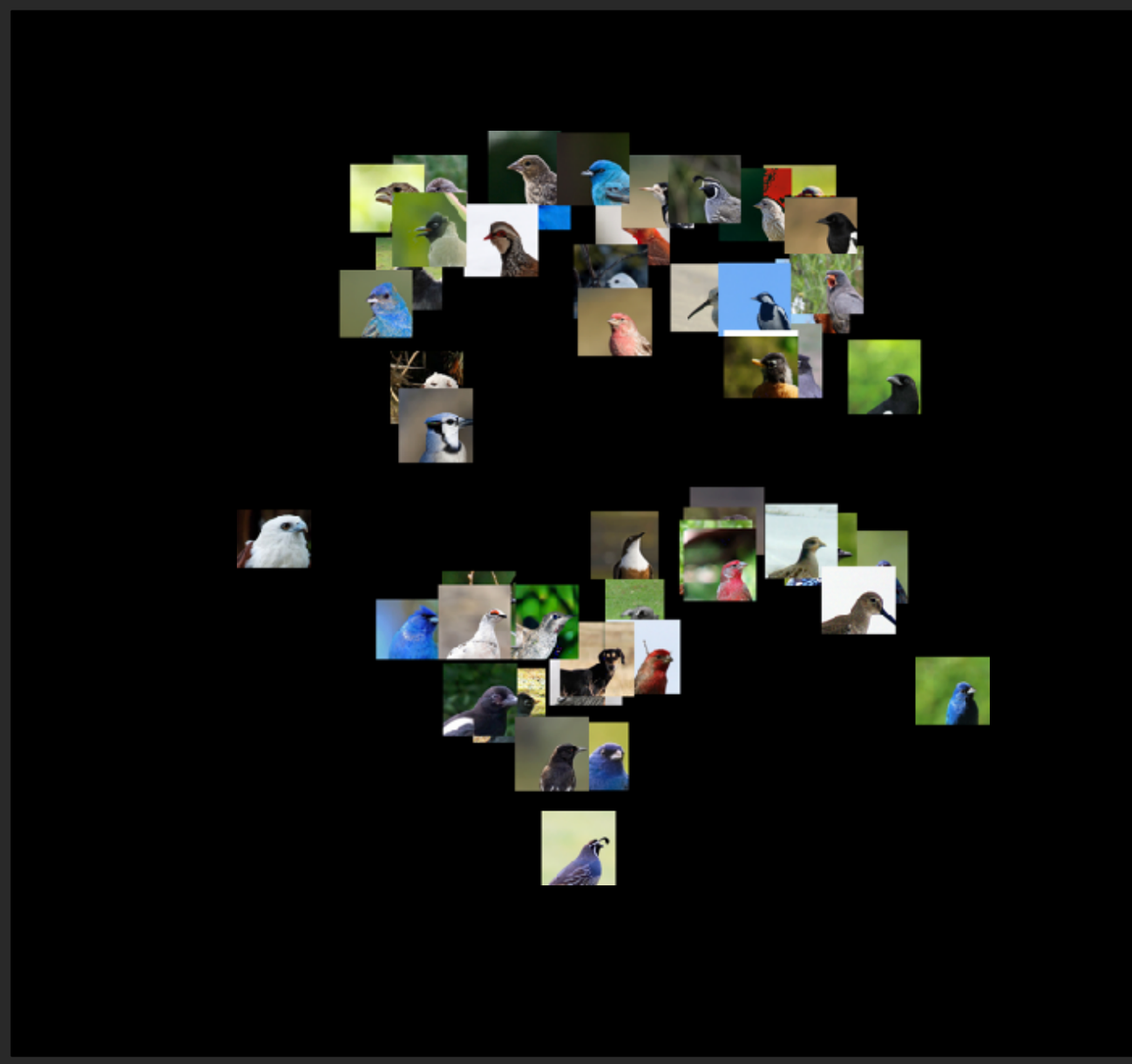
Interpret local neighbordhood of activations 'x' as an embedding for NxN patches

Claim:
Semantics are not manifested in individual neurons, but rather local retinotopic neighborhoods

Visualizing neurons

High-scoring patches vs (PCA) embeddings

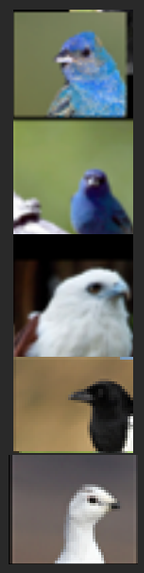


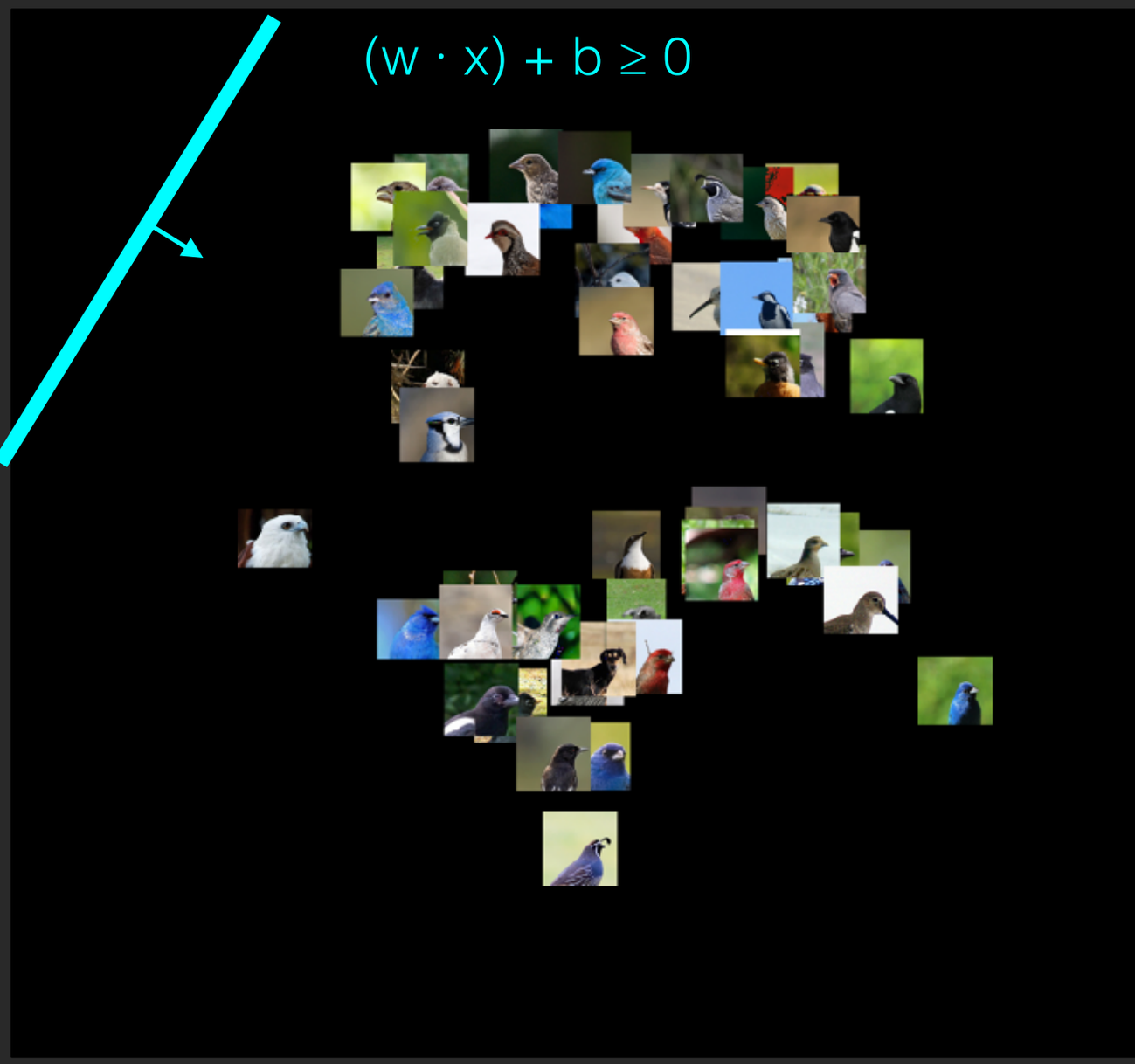


Embeddings often contain additional semantics (pose variation of beaks)

Visualizing neurons

High-scoring patches vs (PCA) embeddings

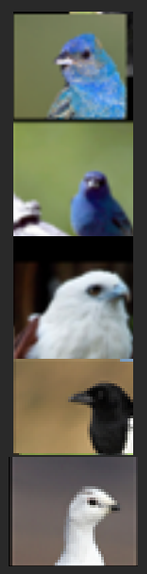


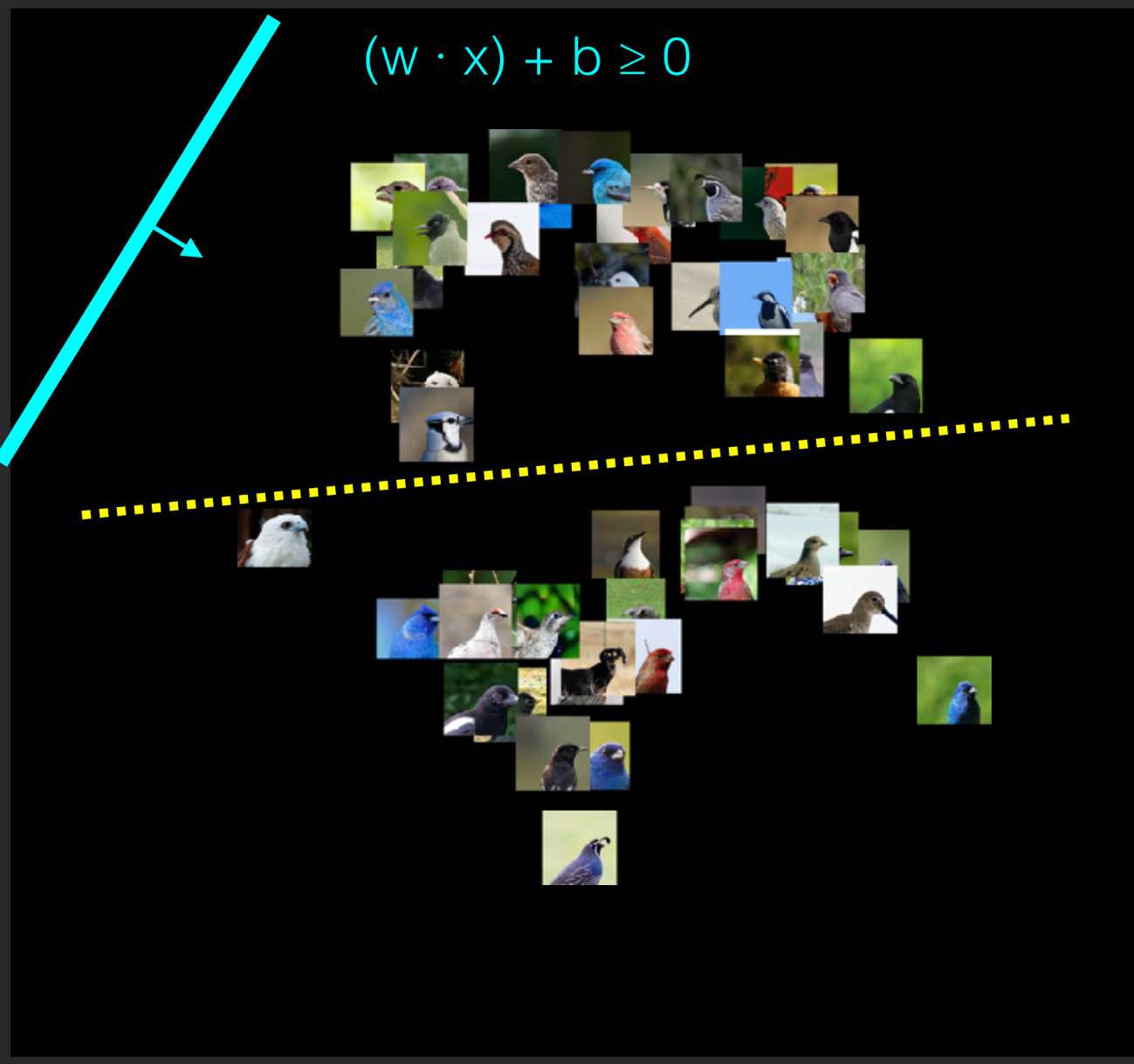


Convolutional filters are linear thresholds in embedded space

Visualizing neurons

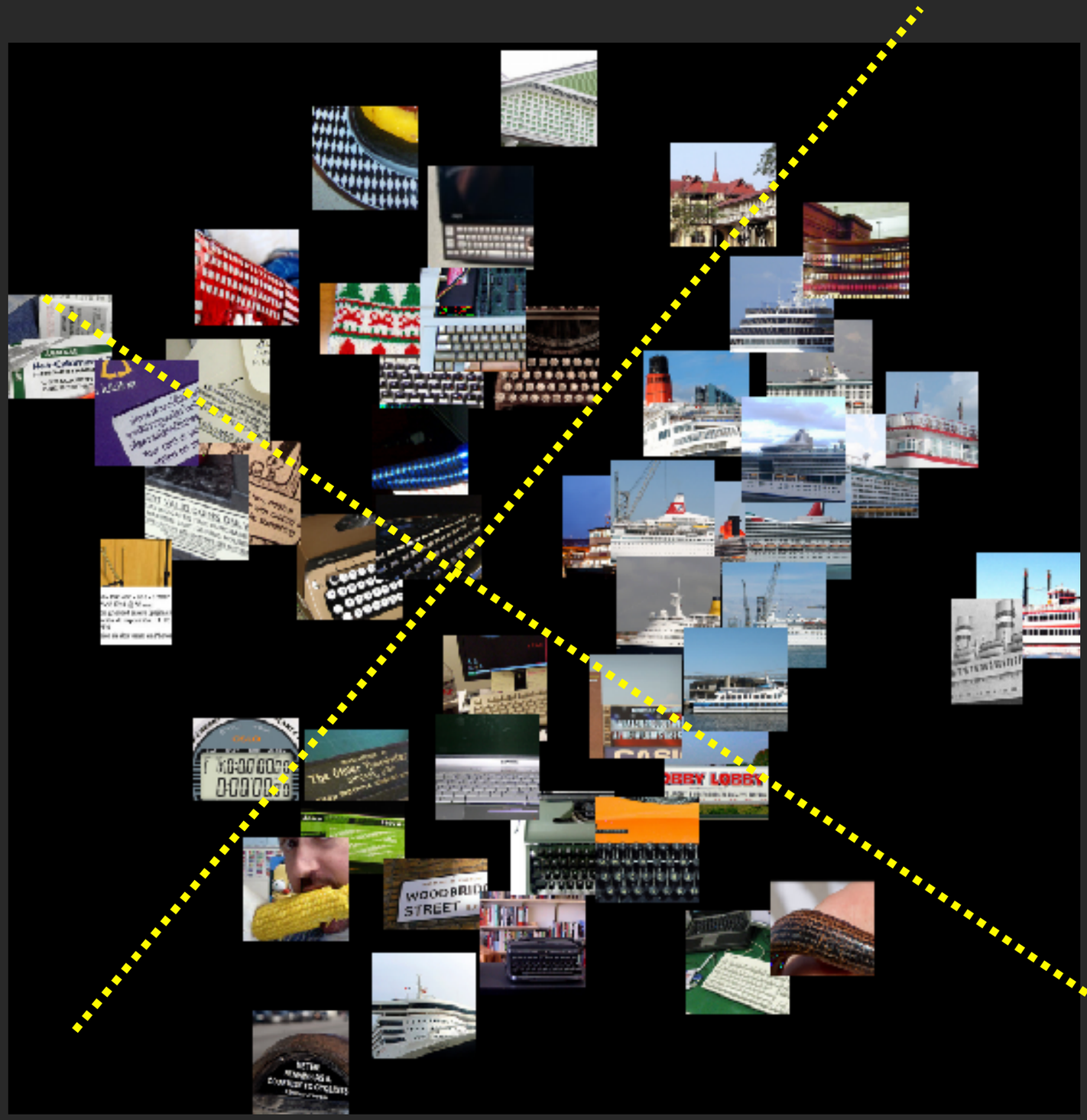
High-scoring patches vs (PCA) embeddings





Allow users to instantiate new filters by drawing new lines (left vs right-facing beaks)

Visualizing neurons (tSNE)



User-drawn filters that delineate ‘keyboards’ versus ‘cruise ships’

Outline

- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale, attention, recurrence, LSTMs
- Why does it work so well? distributed vs local codes