

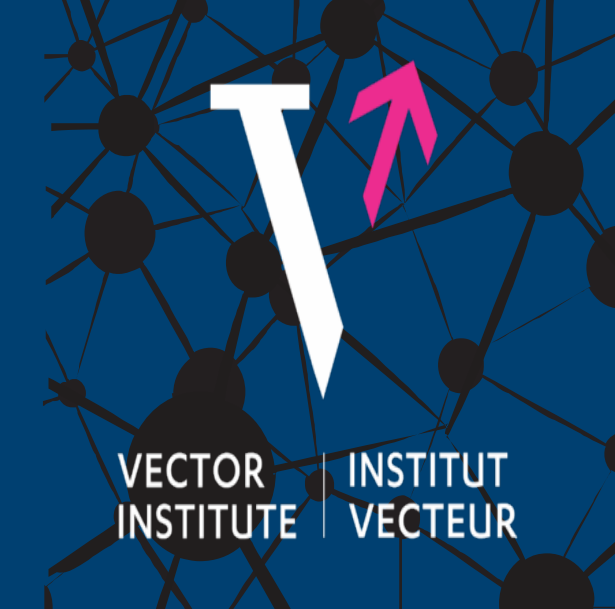
Introduction to Convolutional Networks

Richard Zemel

University of Toronto

June 19, 2017





Vector is an independent non-profit research institution located at the MaRS Discovery District in Toronto, Canada.

Idea developed in discussion with a local startup (CTO, CEO) in September. Funding arranged by January; launched March 30th. Focus on machine learning research and practice.

Vector Institute Faculty



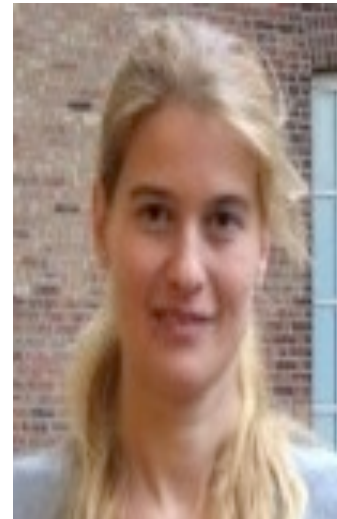
Geoff Hinton
Chief Scientific Advisor



Richard Zemel
Research Director



David Duvenaud



Sanja Fidler



Brendan Frey



Roger Grosse



Quaid Morris



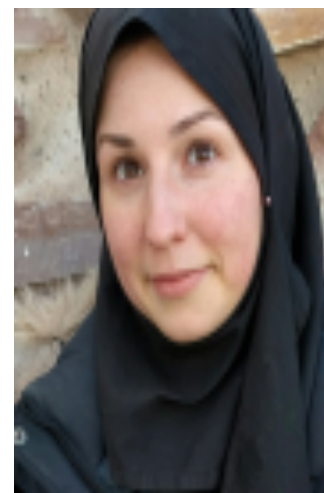
Daniel Roy



Graham
Taylor



Raquel
Urtason



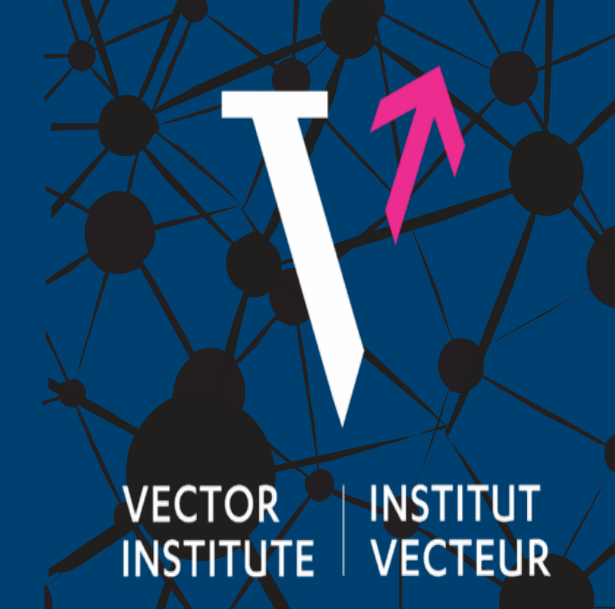
Marzyeh
Ghassemi



Murat Erdogdu



Jimmy Ba



We're hiring!

Looking for

- Research Scientists
- Post-doctoral fellows

For more information, visit us at
vectorinstitute.ai or email
careers@vectorinstitute.ai

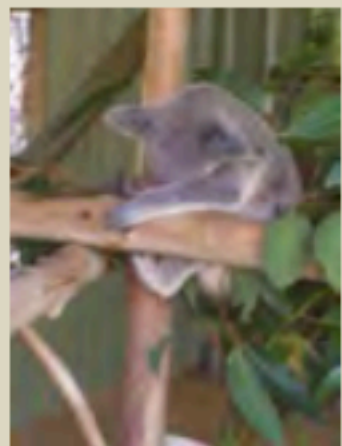


Outline

- Quick summary of convolutional networks
- Recent developments
- What is being learned?
- Applications:
 - Semantic segmentation
 - Image understanding
 - Few-shot learning

Neural Networks for Object Recognition

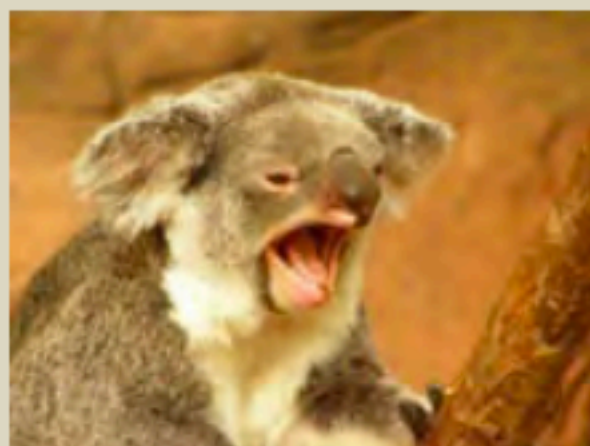
- People are very good at recognizing shapes
 - Intrinsically difficult, computers are bad at it (but getting better)
- Why is it hard?



occlusion



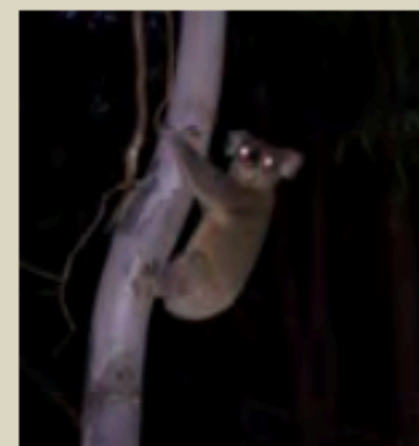
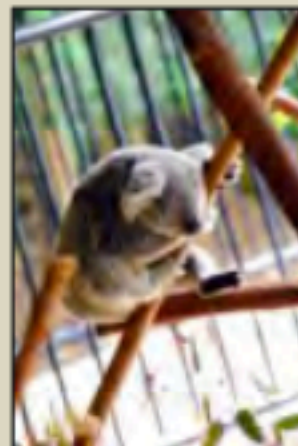
scale



deformation



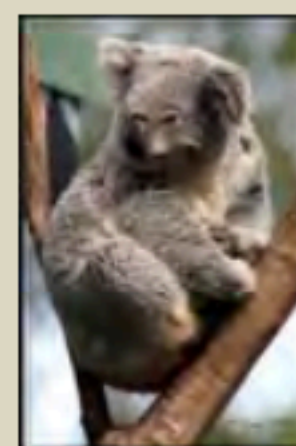
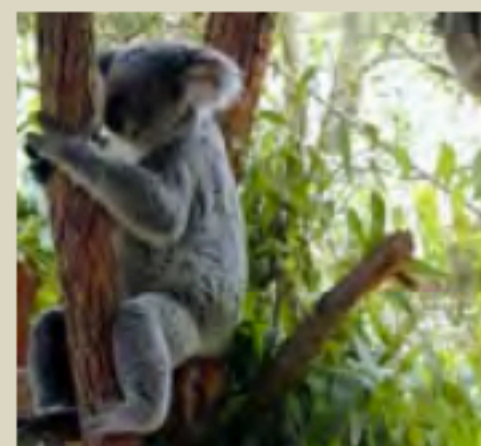
clutter



illumination



viewpoint



object pose

Huge within-class variation. Recognition is mainly about modeling variation.



[Lazebnik]

Tons of classes



~10,000 to 30,000

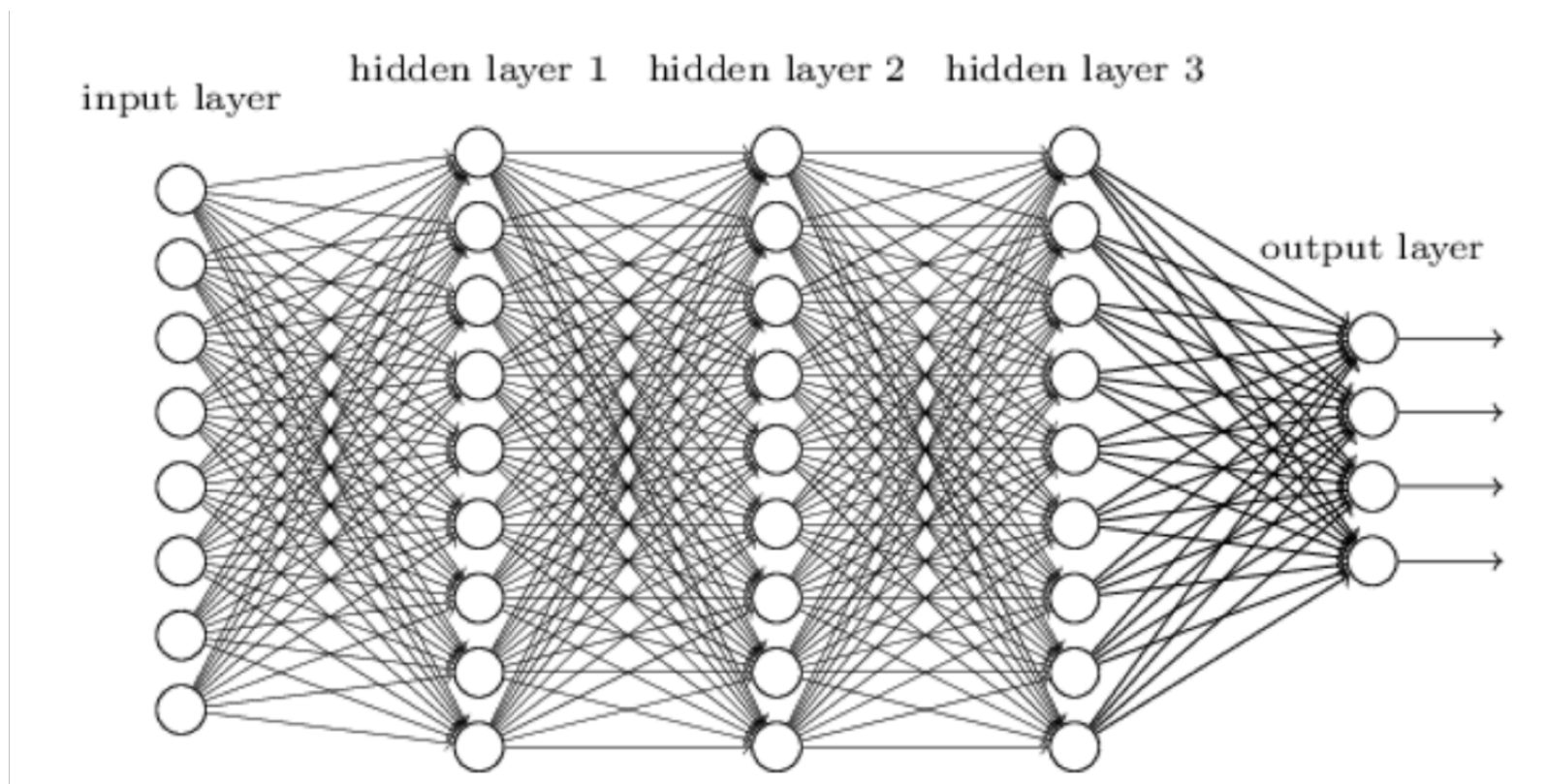
[Biederman]

The invariance problem

- Our perceptual systems are very good at dealing with invariances
 - translation, rotation, scaling
 - deformation, contrast, lighting, rate
- We are so good at this that its hard to appreciate how difficult it is
 - Its one of the main difficulties in making computers perceive
 - We still don't have generally accepted solutions

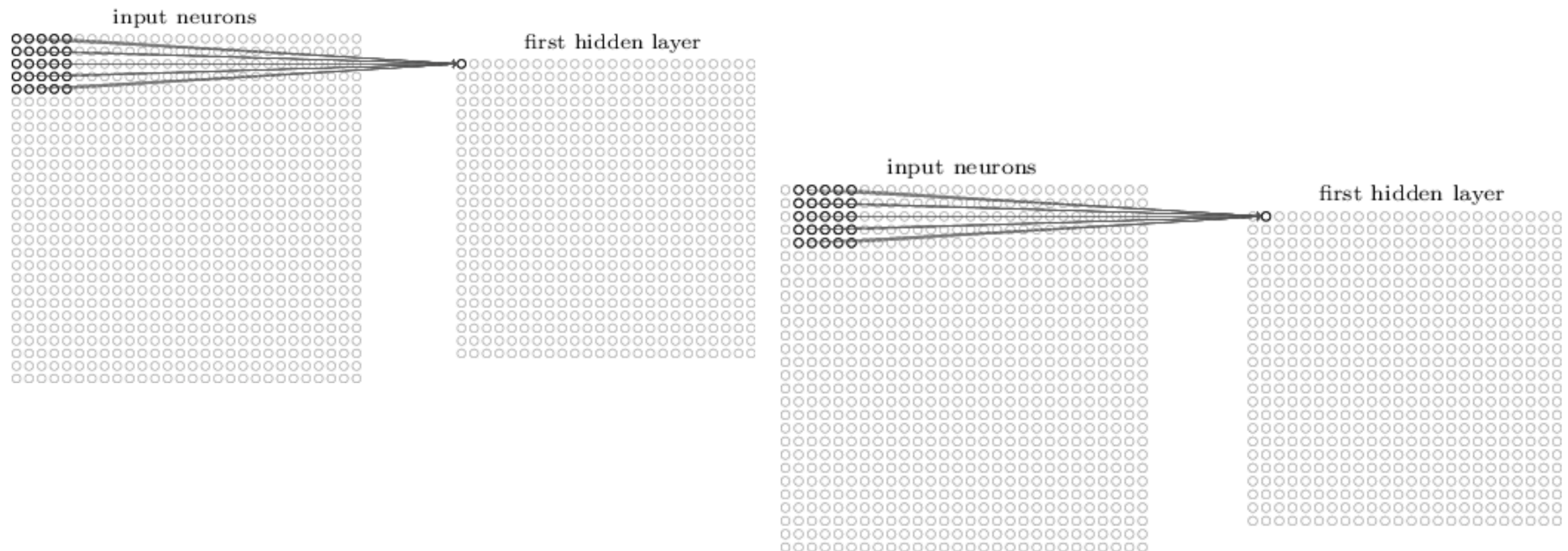
Applying neural nets to images

- Straightforward application of neural networks to images is one option
 - input images contain mega-pixels.
 - Fully-connected neural network prohibitive in terms of number of parameters to learn
- Build in some structure
 - Long-standing issue: tabula rasa vs. build in structure?

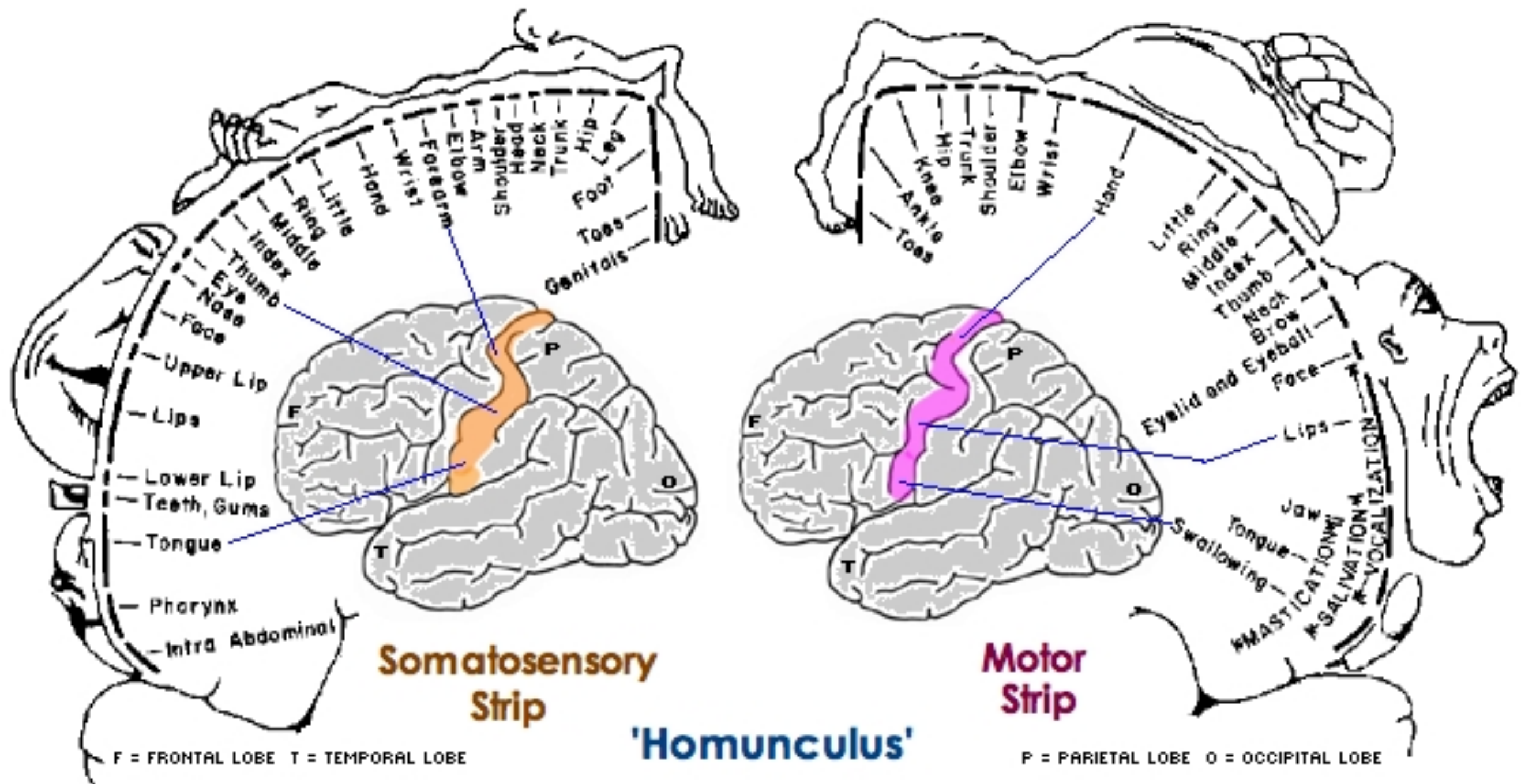


Local Receptive Fields

- Each neuron receives input from a subset of neurons in layer below: **receptive field**
 - Lowest level: Small window in image.
 - Arrange units **topographically**: neighboring units have similar properties due to overlapping receptive fields

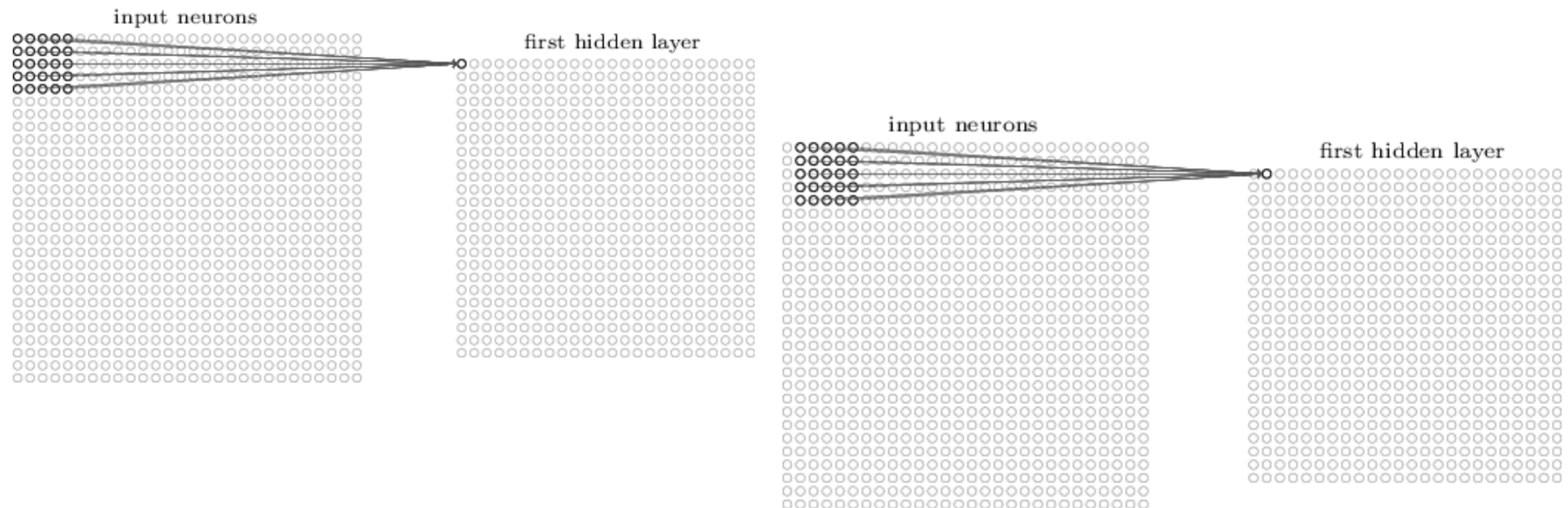


Topographic Maps



Local Receptive Fields

- Each neuron receives input from a subset of neurons in layer below: **receptive field**
 - Lowest level: Small window in image.
 - Arrange units **topographically**: neighboring units have similar properties due to overlapping receptive fields
 - Offset in RF between neighboring units: **stride**



Local Receptive Fields

What is a single unit/neuron responding to?

- Some local feature in the image.
- Weights known as **feature detector**, **kernel**, or **filter**

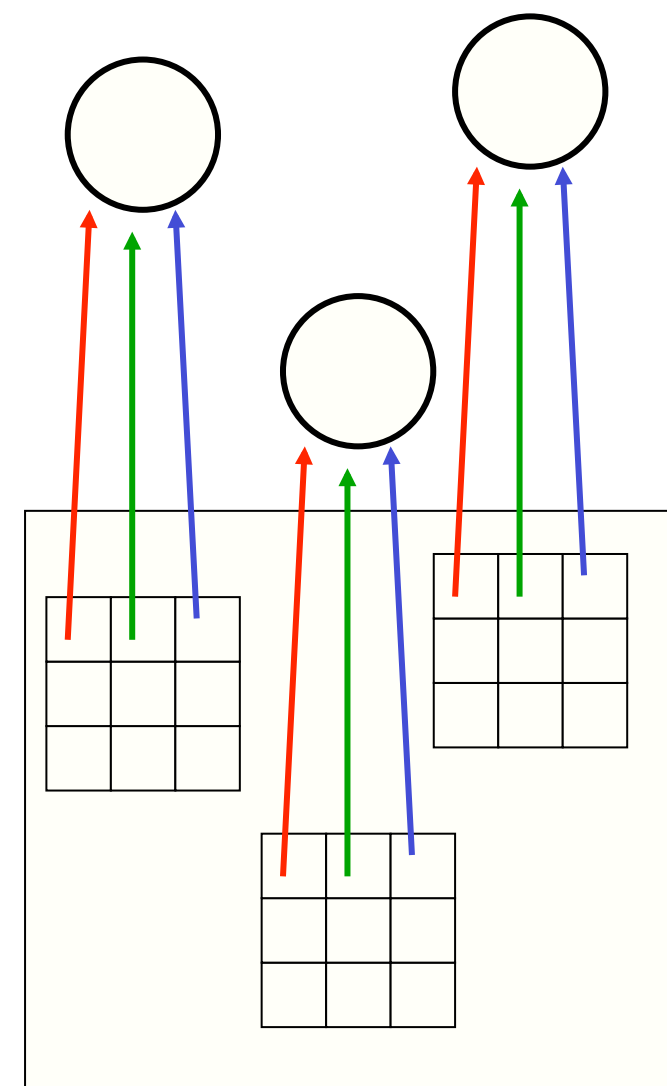
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

Shared Weights

- Use many different copies of the same feature detector: **replicated feature**.
 - Copies have slightly different positions.
 - Could replicate across scale and orientation.
 - Tricky and expensive
 - Replication reduces number of free parameters to be learned.
 - Stationarity: statistics are similar at different image positions (LeCun 1998)
- **Convolutional neural network: local RFs, shared weights**
- Use several different feature types, each with its own replicated pool of features.
 - Allows each patch of image to be represented in several ways.

The red connections all have the same weight.

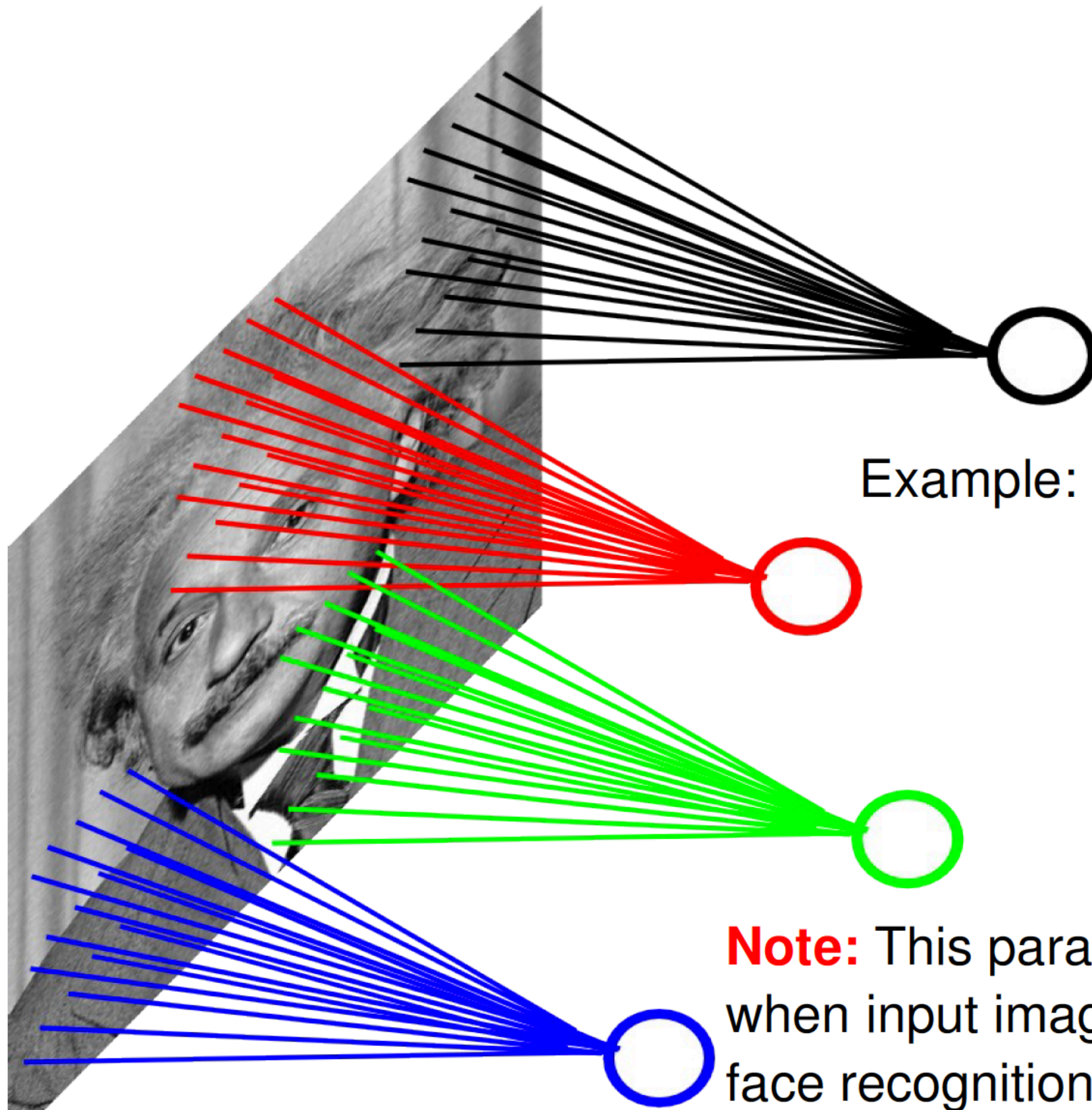


Examples of Feature Detectors



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

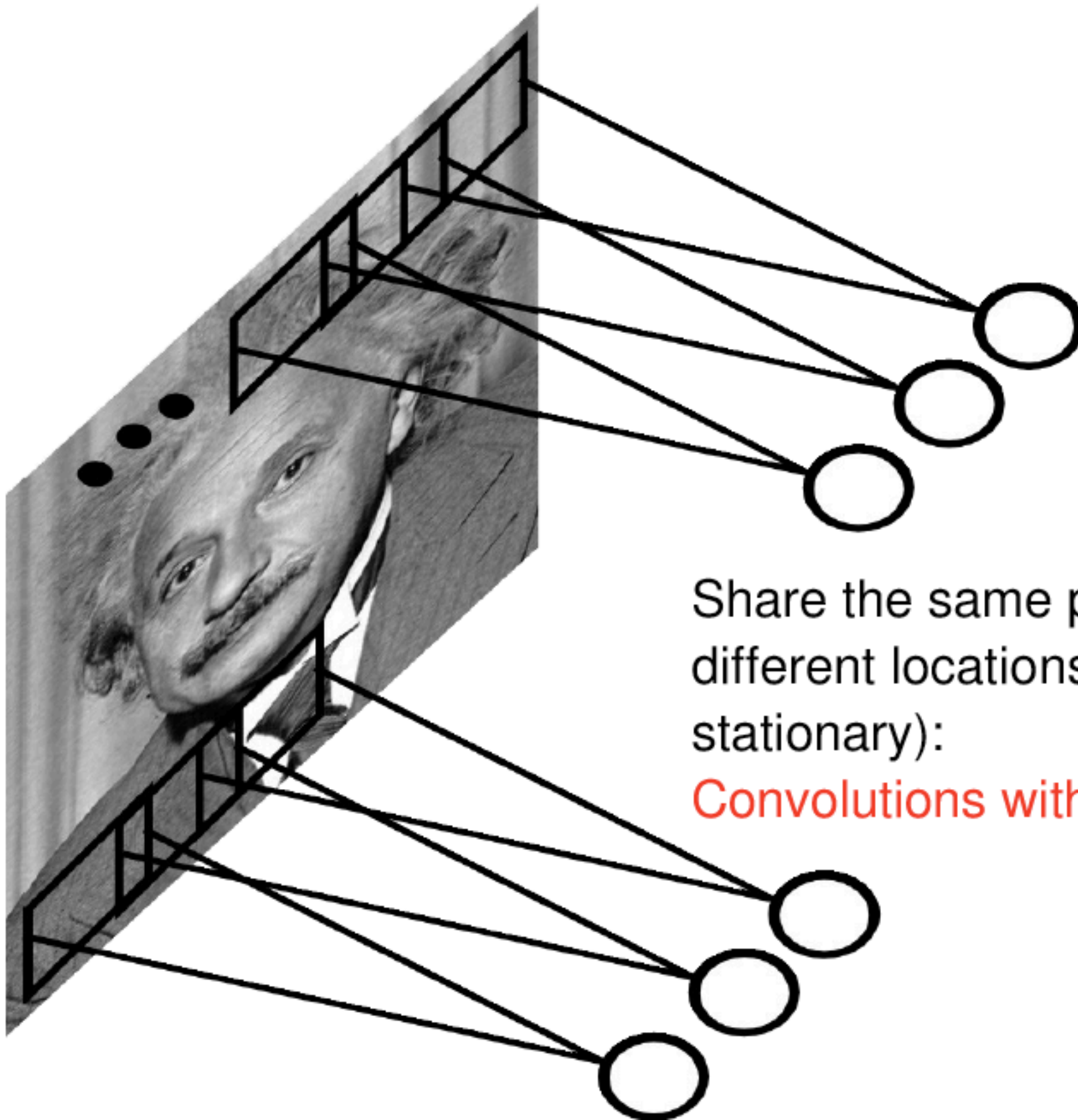
Local RFs



Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Weight sharing: Parameter saving



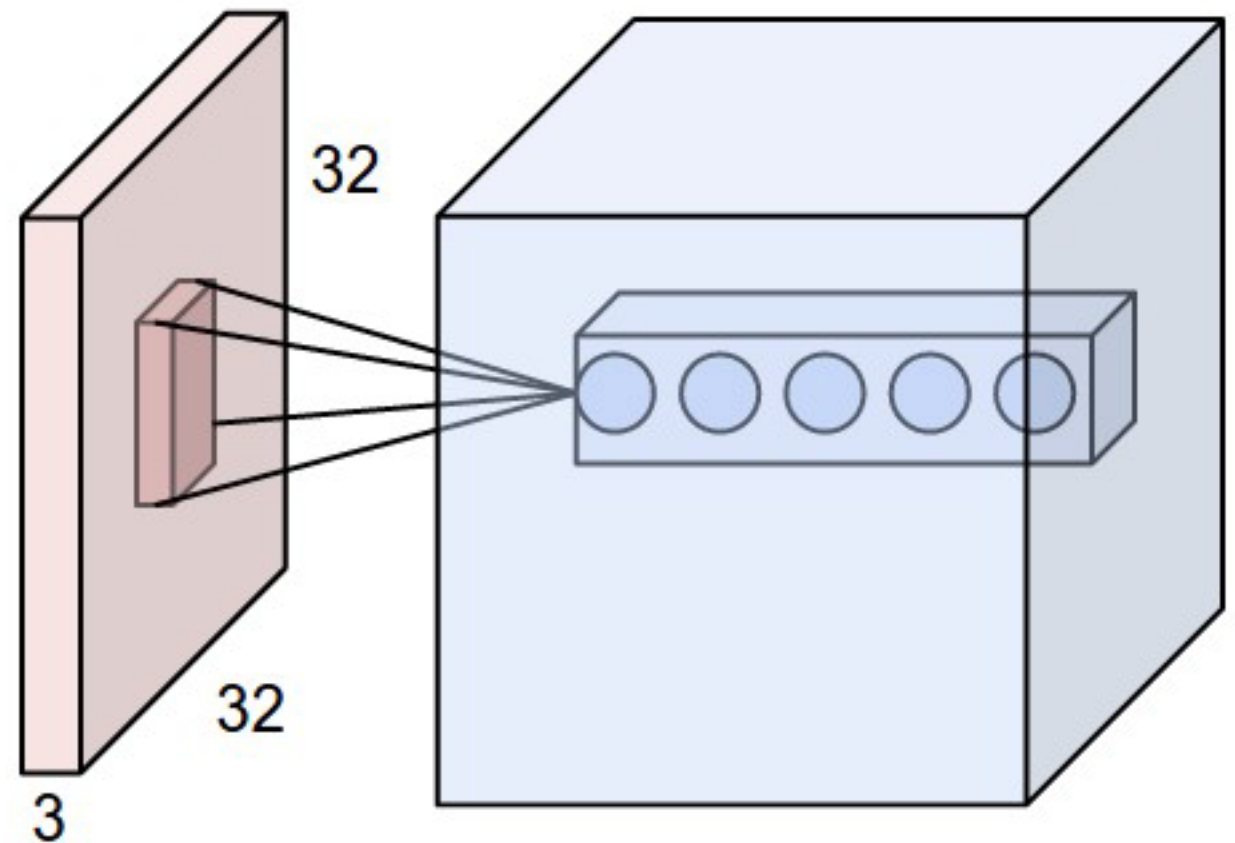
Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels

Convolutional Layer

Each input modality forms a different **channel**

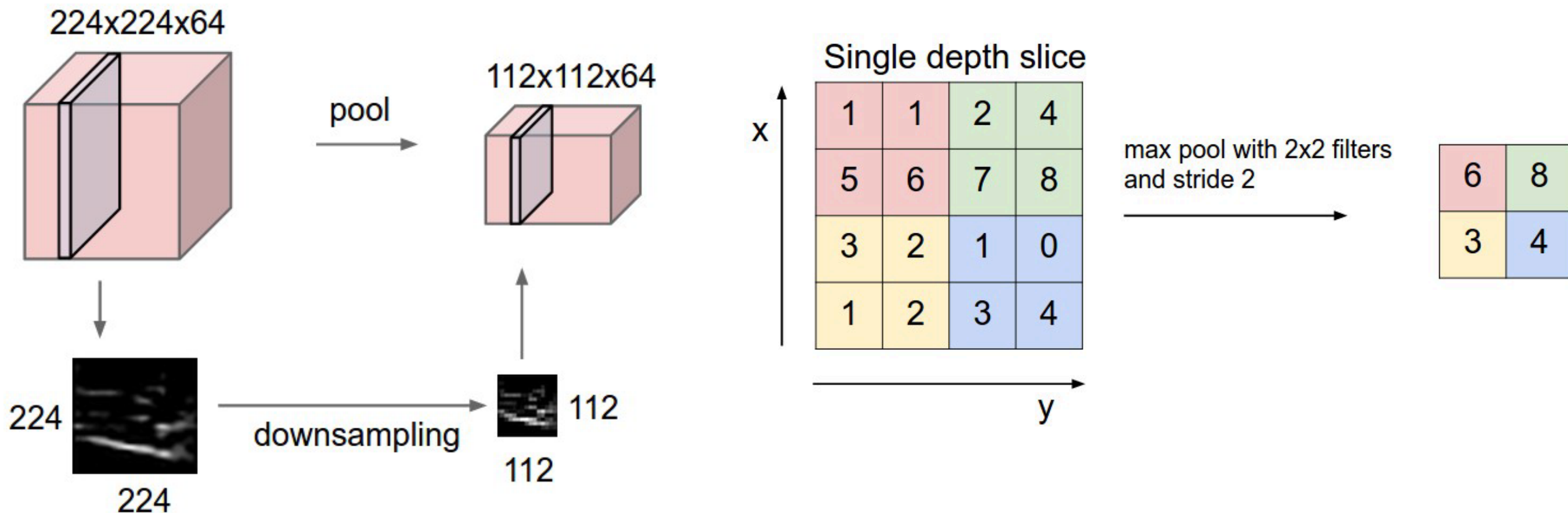
Activation of convolutional layer: **feature map**



Hyper-parameters of a convolutional layer:

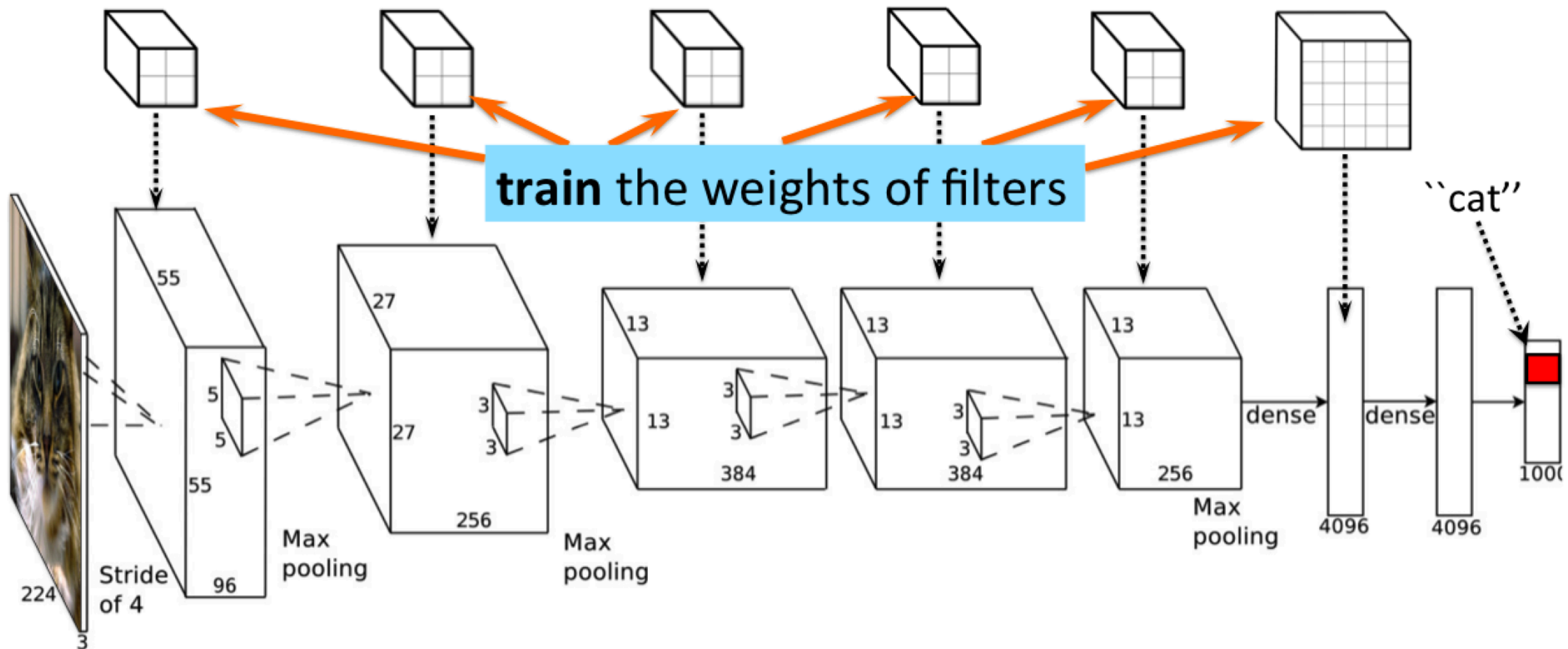
- The number of filters (controls the depth of the output volume)
- The stride : how many units apart do we apply a filter spatially (controls the spatial size of the output volume)
- The size $w \times h$ of the filters

Pooling



- Each convolutional layer followed by a pooling layer
- Pooling: combine filter responses within a window
 - Simplifies information in the output of the conv layer
 - Reduces dimensionality
 - Gain robustness wrt spatial location
- Standard forms: max, average

Finishing it off



- Add one or more fully connected layers at end, then produce output (classification prediction)
- Activation functions (sigmoid, tanh, RELU, ELU)
- Loss function (KL, hinge)
- Typically train by back-prop, SGD

Le Net

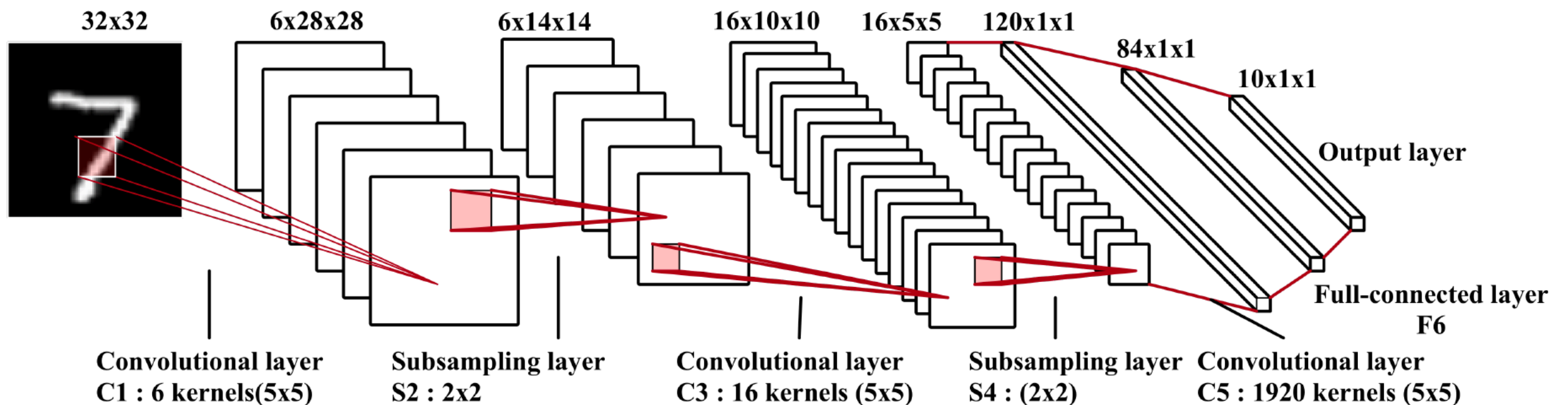
Main ideas:

- Local → global processing
- Retain coarse posn info

Main technique: weight sharing – units arranged in feature maps

Connections: 1256 units, 64,660 cxns, 9760 free parameters

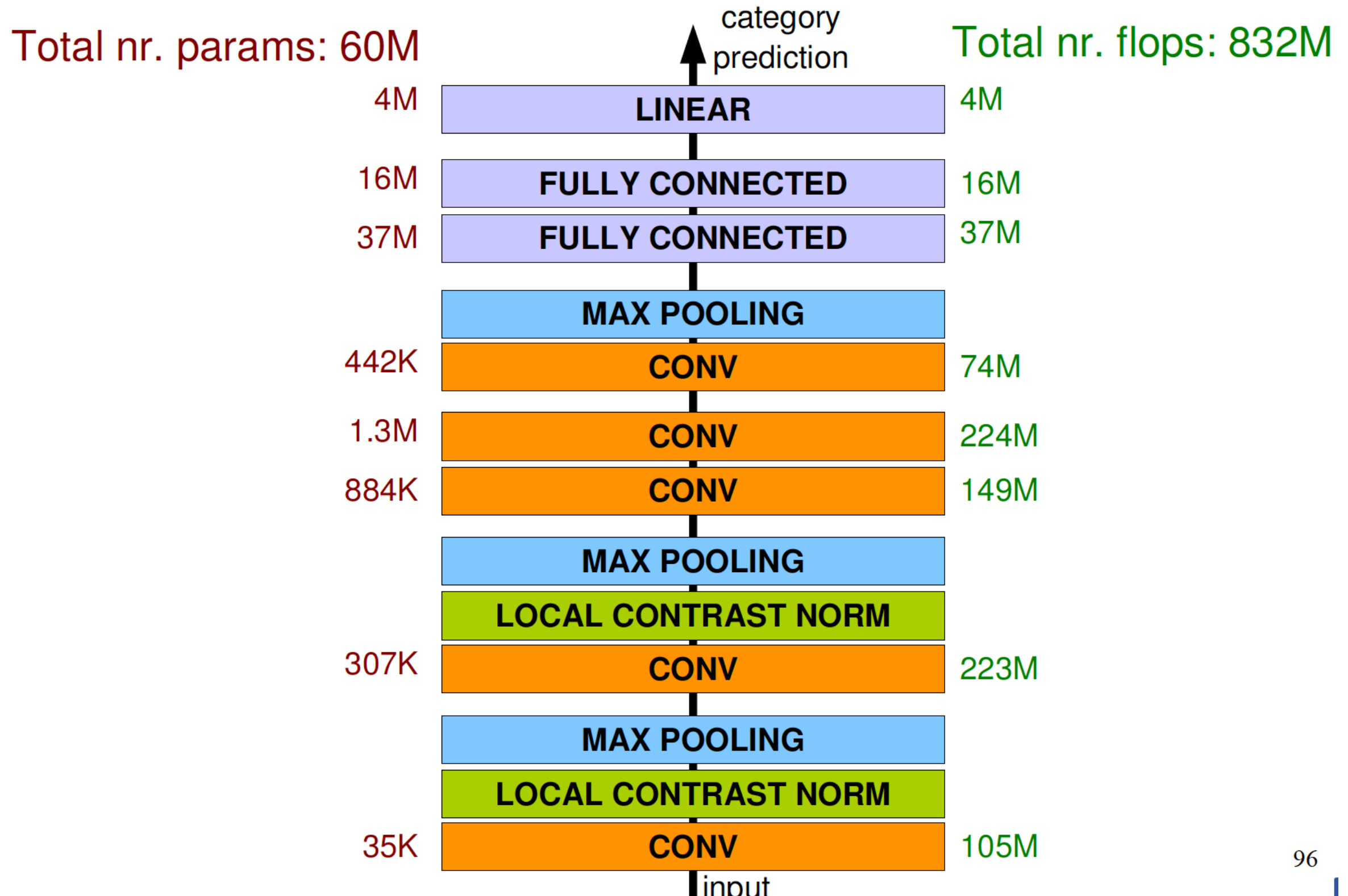
Results: 0.14% (train), 5.0% (test)



Outline

- Quick summary of convolutional networks
- **Recent developments**
- What is being learned?
- Applications:
 - Semantic segmentation
 - Image understanding
 - Few-shot learning

Modern CNNs: Alexnet (2012)



VGG (2014)

add more layers

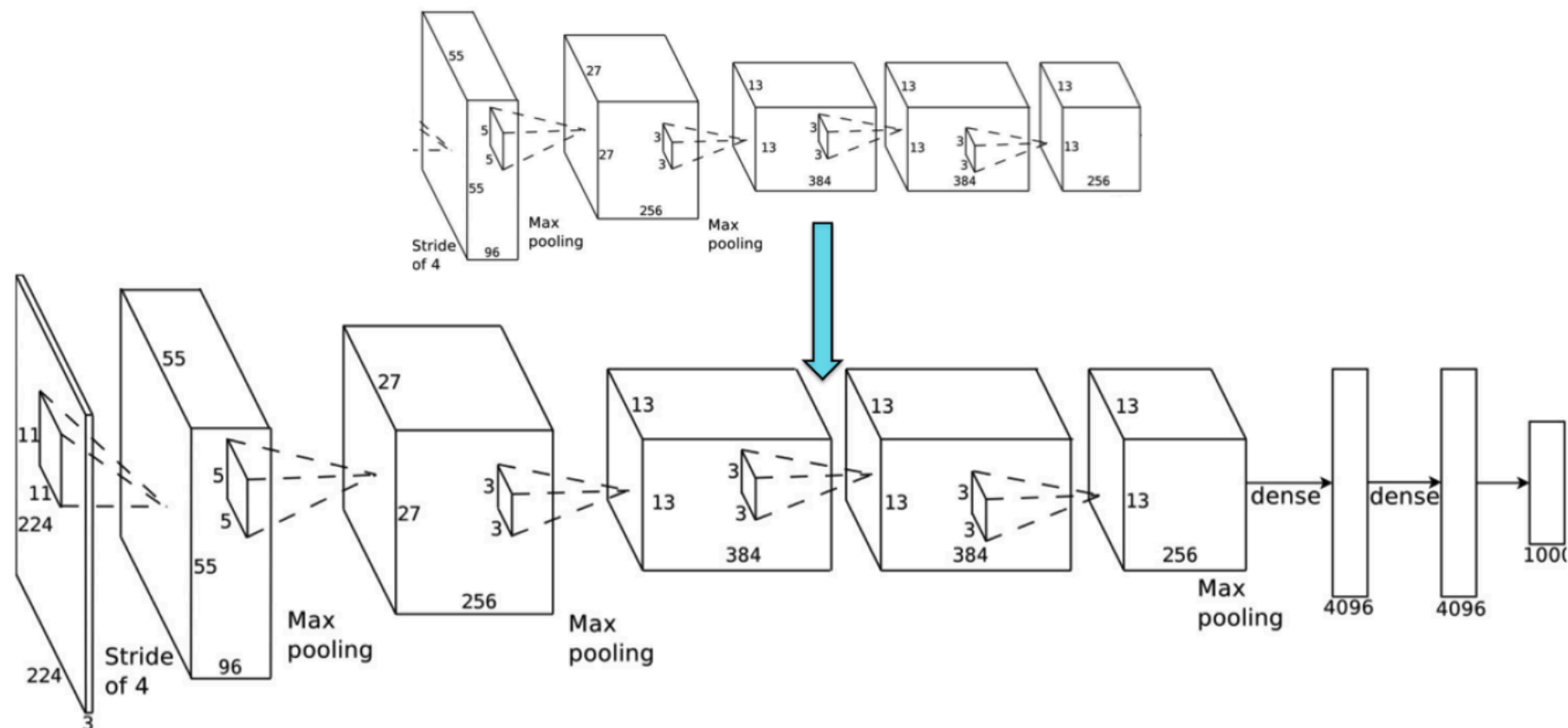
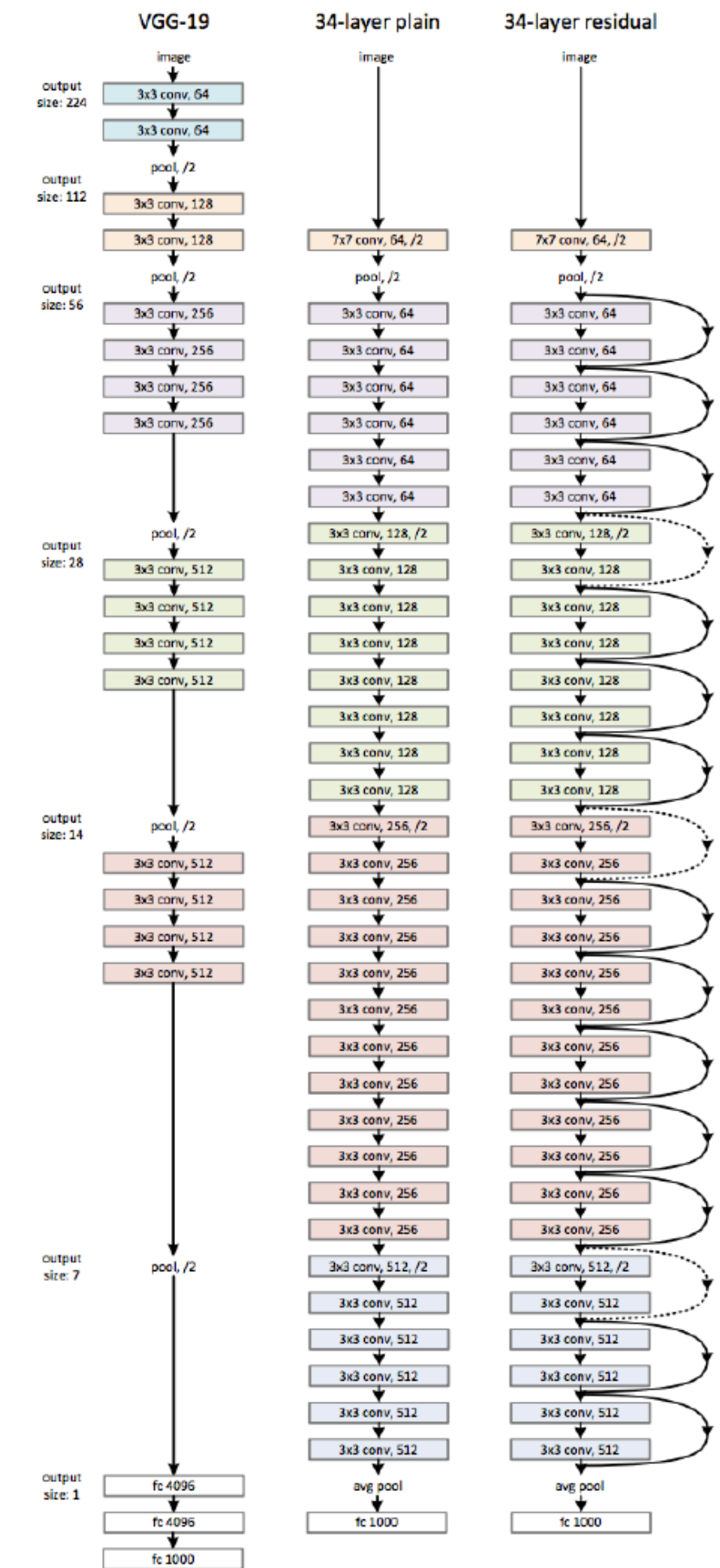
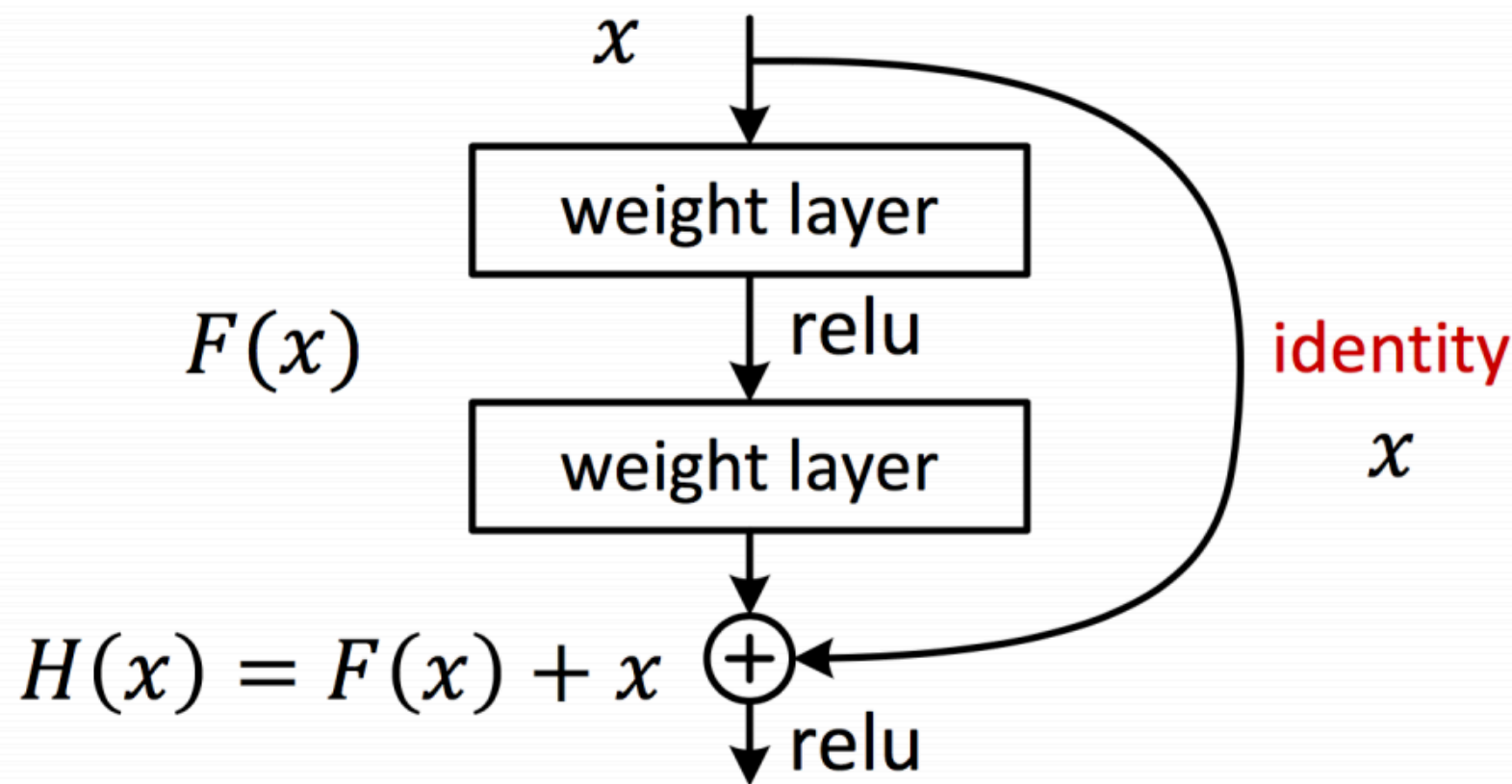


Figure : K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 2014

Residual Networks (2015)

Inspired by LSTM gating units

- Combats vanishing gradients
- Simple form of auto-regression



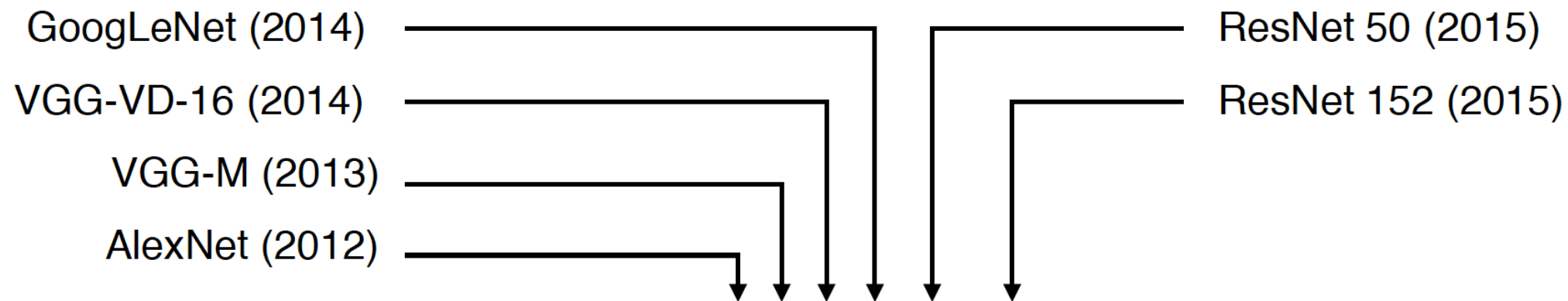
Highway Networks (2015)

$$\mathbf{x}_{l+1} = \phi_{l+1}(\mathbf{x}_l, \mathbf{W}_H) \cdot \tau_{l+1}(\mathbf{x}_l, \mathbf{W}_T) + \mathbf{x}_l \cdot (\mathbf{1} - \tau_{l+1}(\mathbf{x}_l, \mathbf{W}_T))$$

- Inspired by LSTM gating function
 - Combats vanishing gradients
 - Dynamically determines when data passed thru vs. transformed
- In general, **shortcut connections** allow training of very deep networks.

How Deep?

- 10 times deeper in a few years



16 convolutional layers



50 convolutional layers



152 convolutional layers



Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

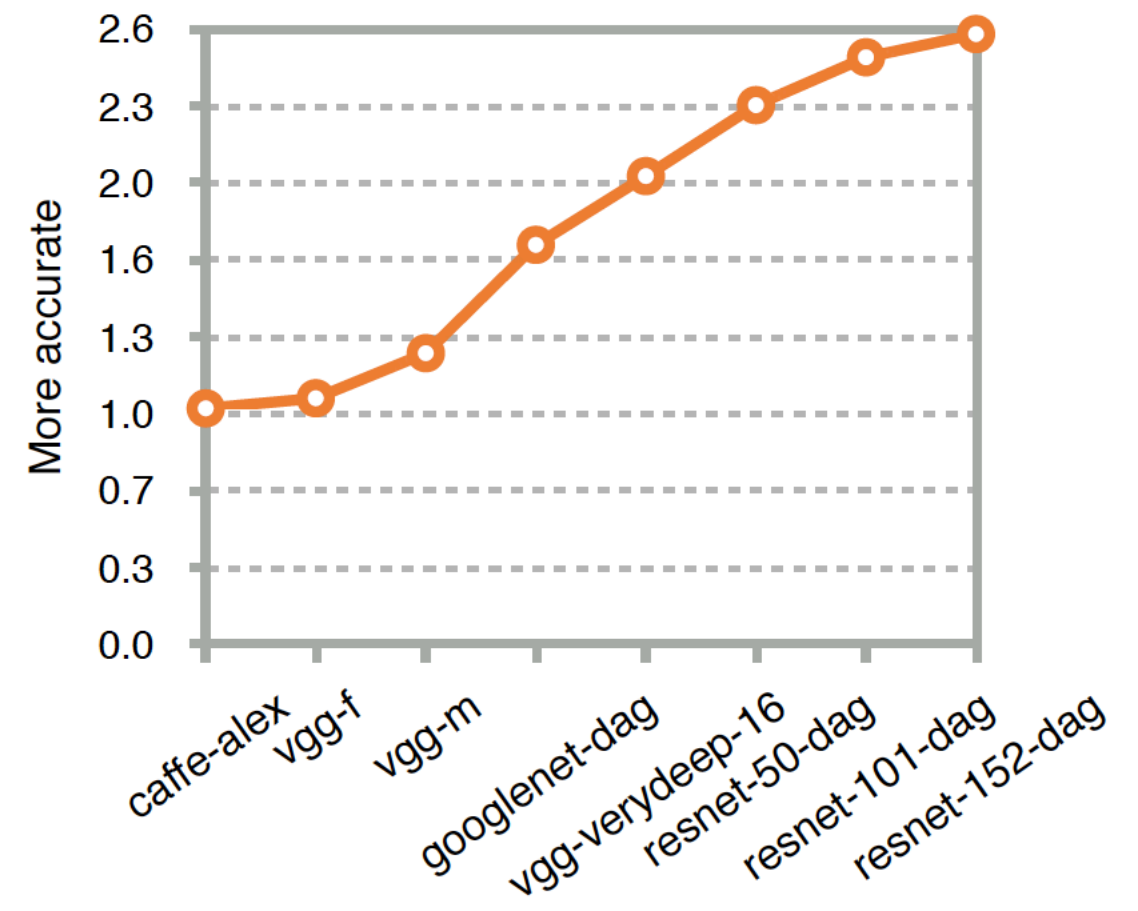
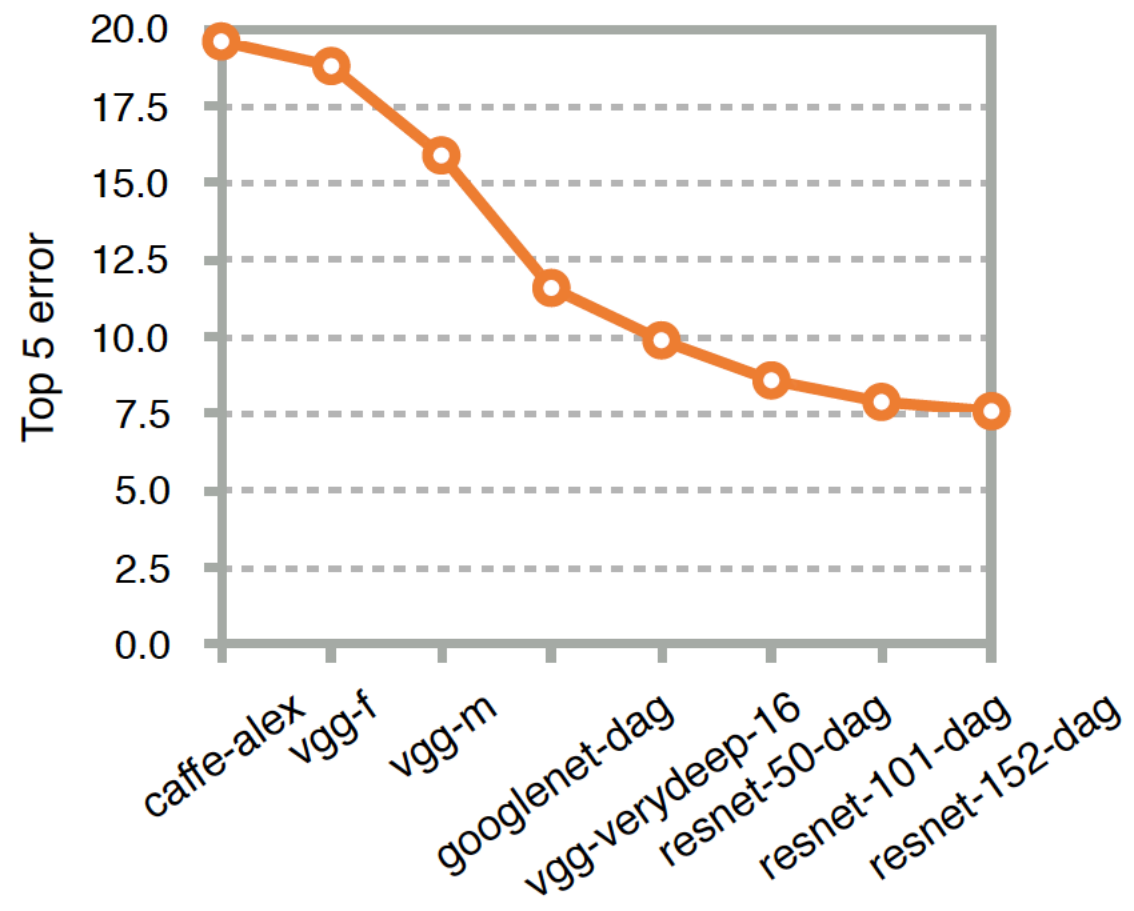
C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

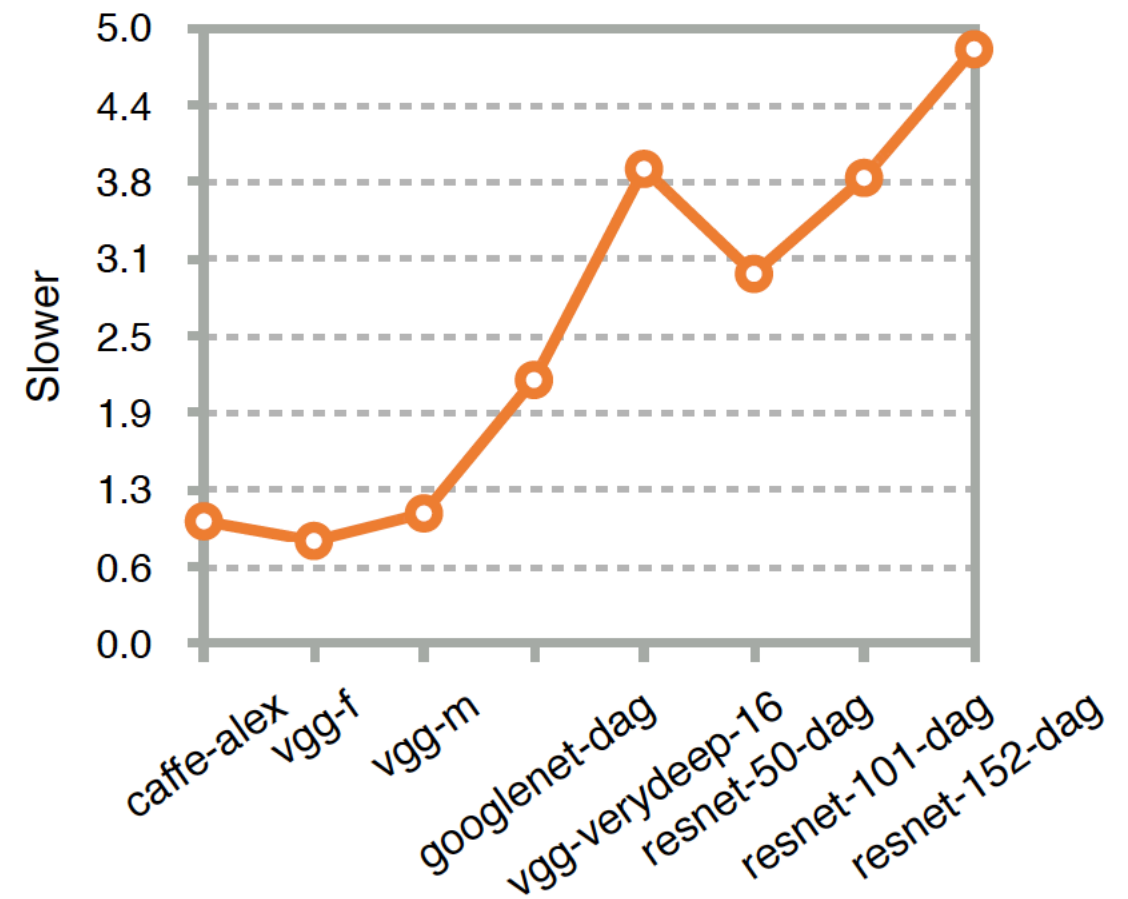
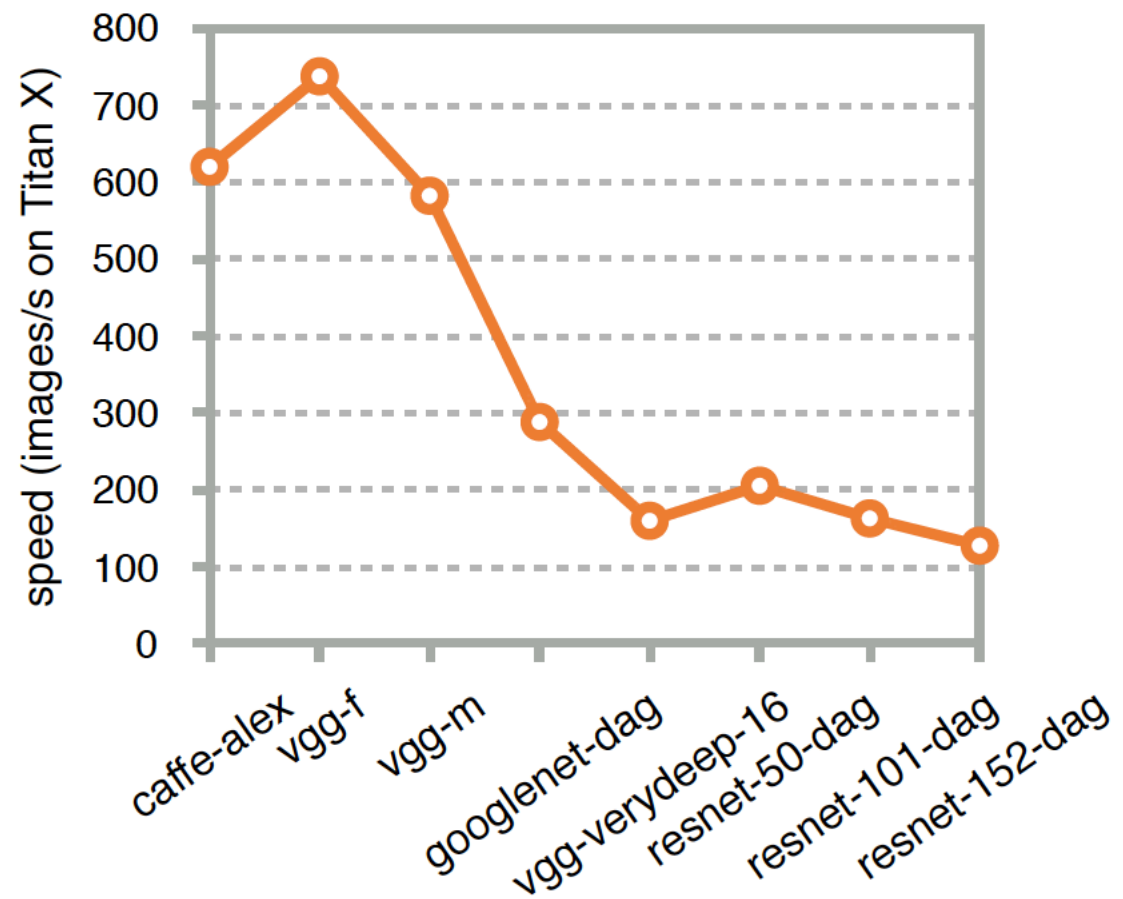
How Deep? Good?

- 3 times better



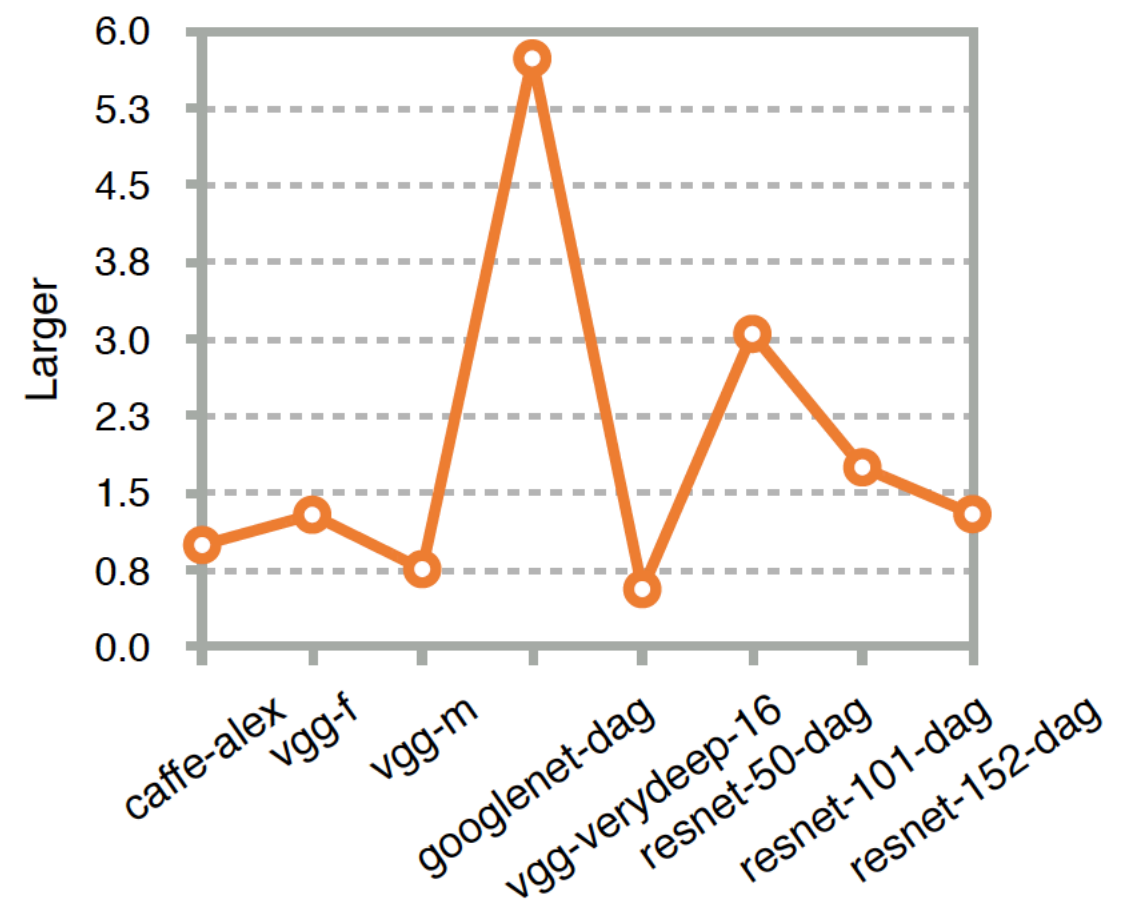
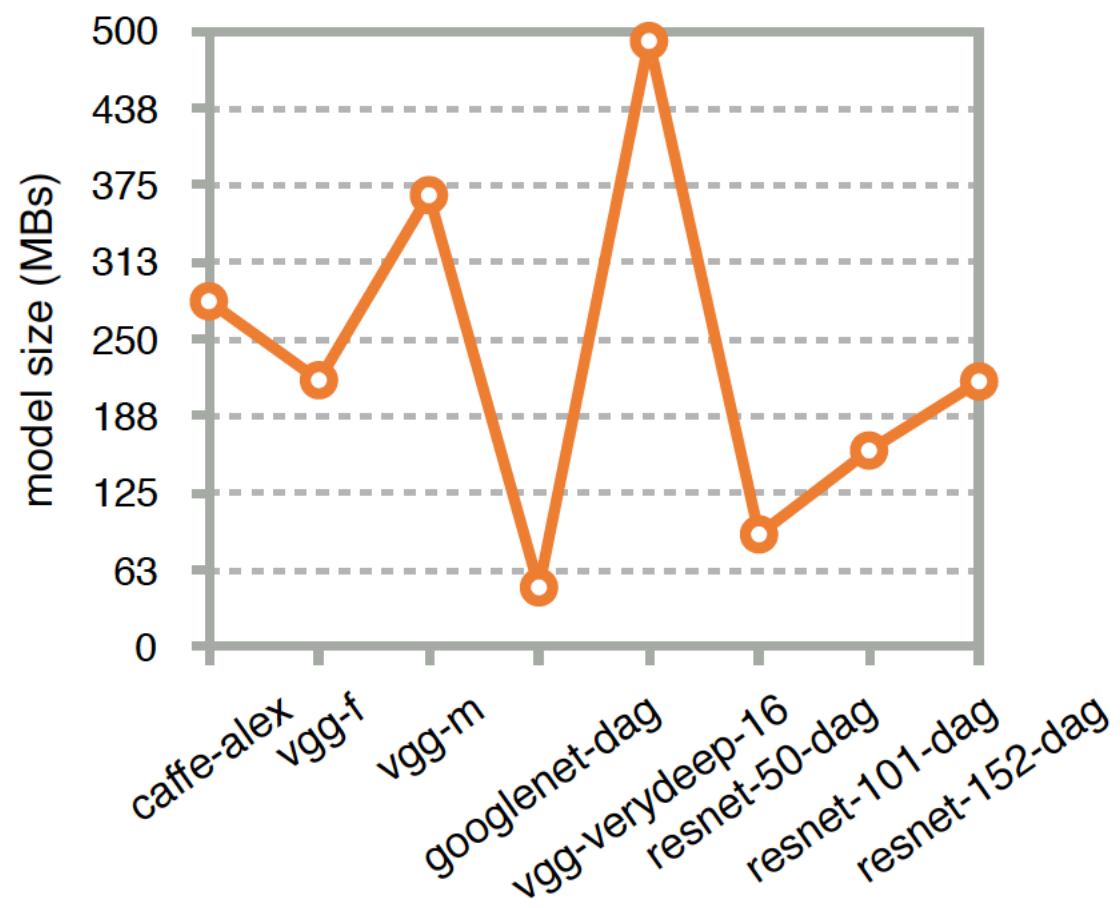
How Deep? Good? Slow?

- 5 times slower



How Deep? Good? Slow? Complex?

- Number of parameters about the same

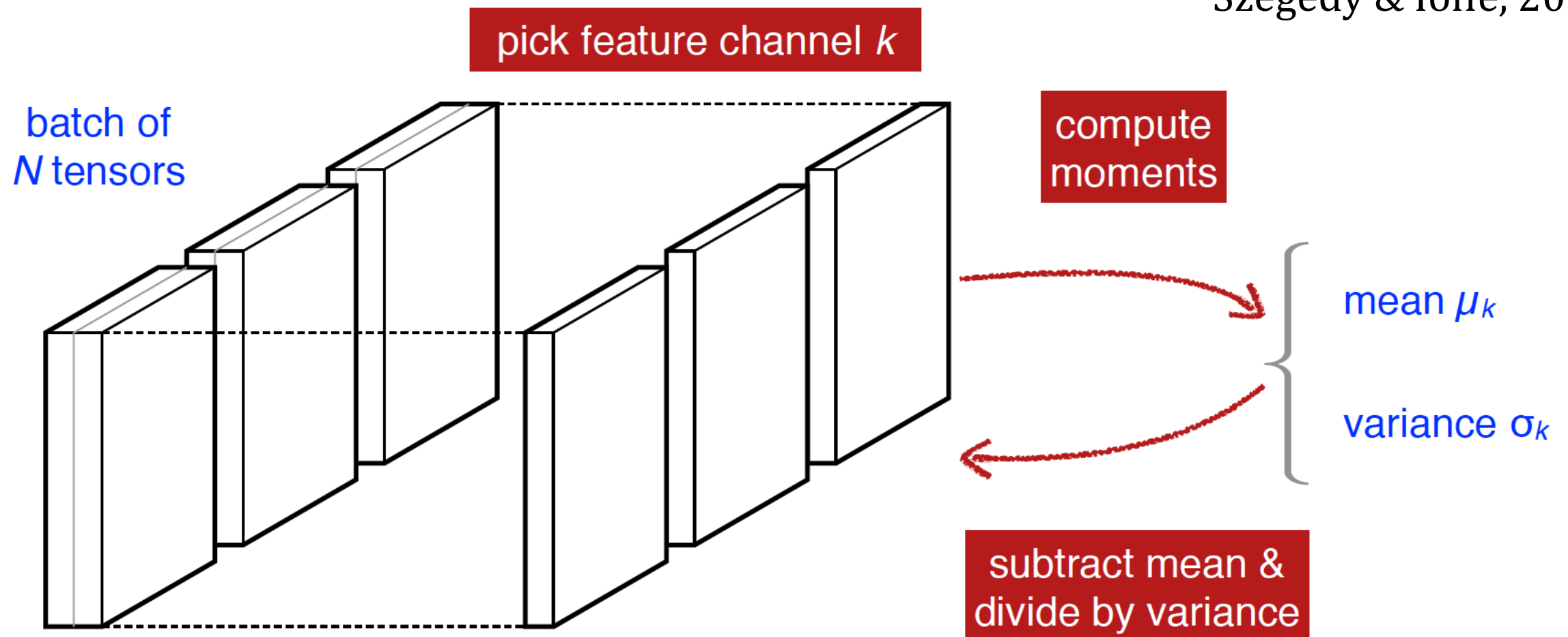


Recent Developments in CNNs

Normalization

1. **Batch normalization**: standardize response of each feature channel within a batch

Szegedy & Ioffe, 2015



$$\tilde{z}_{n,j} = \gamma \frac{z_{n,j} - \mathbb{E}[z_j]}{\sqrt{\frac{1}{|B(n)|} (z_{n,j} - \mathbb{E}[z_j])^2}} + \beta \quad \mathbb{E}[z_j] = \frac{1}{|B(n)|} \sum_{m \in B(n)} z_{m,j}$$

Normalization

1. Batch Normalization

2. **Layer normalization**: standardize response of each feature channel within a layer

[Ba, Kiros. Hinton, 2016]

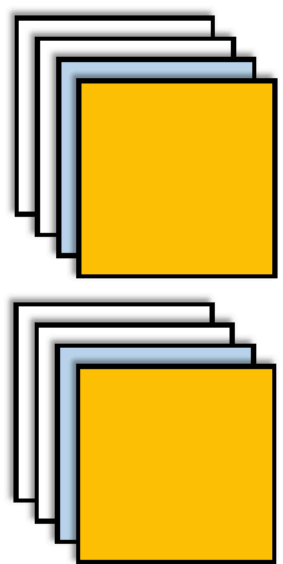
$$\tilde{z}_{n,j} = \gamma \frac{z_{n,j} - \mathbb{E}[z_n]}{\sqrt{\frac{1}{|L(j)|} (z_{n,j} - \mathbb{E}[z_n])^2}} + \beta \quad \mathbb{E}[z_n] = \frac{1}{|L(j)|} \sum_{k \in L(j)} z_{n,k}$$

– Good results in RNNs, not as good on CNNs

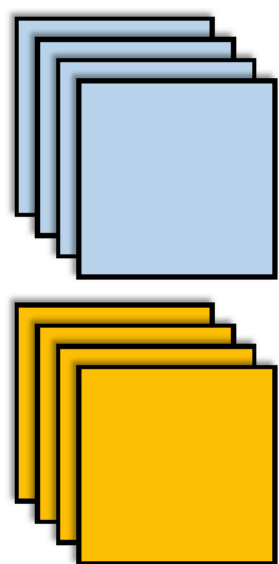
3. **Weights, spatial normalization** schemes

4. **Divisive normalization**: canonical computation in the brain

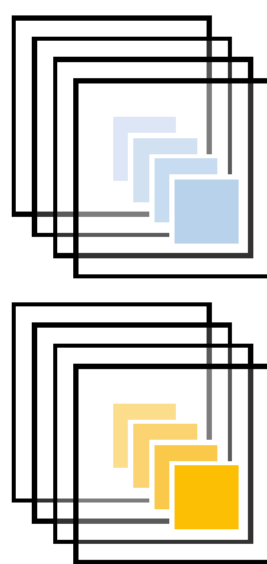
Divisive Normalization



(a) Batch-Norm



(b) Layer-Norm



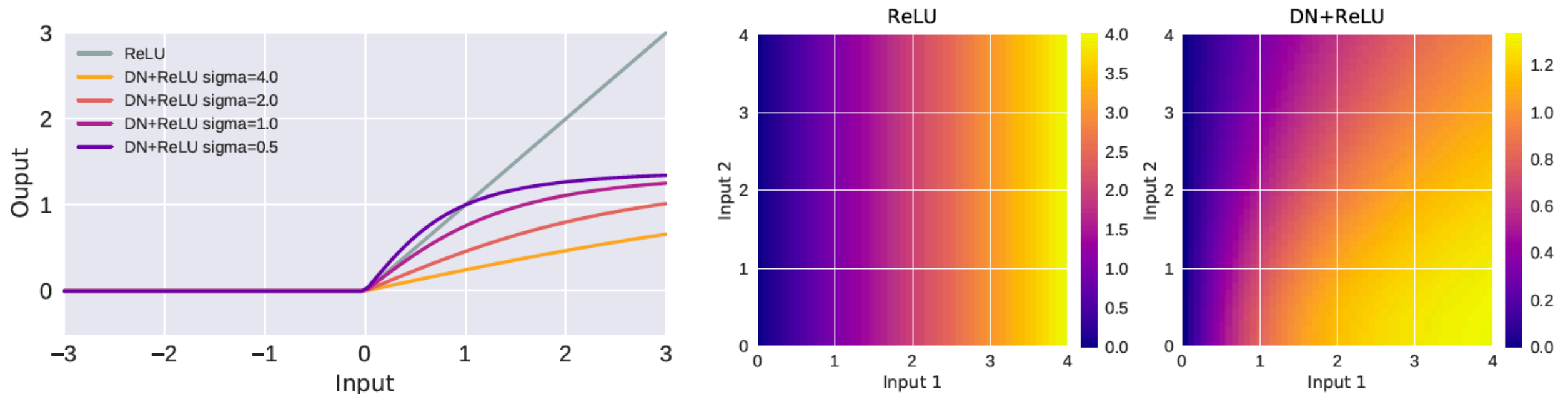
(c) Div-Norm

[Ren, Liao, Sinz, Urtasun, Zemel, 2017]

$$\tilde{z}_j = \gamma \frac{\sum_{z_i \in \mathcal{A}_j} u_i z_i}{\left(\sigma^2 + \sum_{z_k \in \mathcal{B}_j} w_k z_k^p \right)^{\frac{1}{p}}}$$

Model	Range	Normalizer Bias
BN	$\mathcal{A}_{n,j} = \{z_{m,j} : m \in [1, N], j \in [1, H] \times [1, W]\}$ $\mathcal{B}_{n,j} = \{v_{m,j} : m \in [1, N], j \in [1, H] \times [1, W]\}$	$\sigma = 0$
LN	$\mathcal{A}_{n,j} = \{z_{n,i} : i \in [1, L]\}$ $\mathcal{B}_{n,j} = \{v_{n,i} : i \in [1, L]\}$	$\sigma = 0$
DN	$\mathcal{A}_{n,j} = \{z_{n,i} : d(i, j) \leq R_{\mathcal{A}}\}$ $\mathcal{B}_{n,j} = \{v_{n,i} : d(i, j) \leq R_{\mathcal{B}}\}$	$\sigma \geq 0$

Divisive Normalization



Model	CIFAR-10 Acc.	CIFAR-100 Acc.
Baseline	0.7565	0.4409
Baseline +WD +Dropout	0.7795	0.4179
BN	0.7807	0.4814
LN	0.7211	0.4249
BN*	0.8179	0.5156
LN*	0.8091	0.4957
DN*	0.8122	0.5066

[Ren, Liao, Sinz, Urtasun, Zemel,
2017]

Network Design: Receptive Fields

Receptive field (RF)

- image region influencing neuron (unit) response
- anything outside the RF is invisible to neuron
- Important: large image structures cannot be analyzed by neurons with small RFs

Analysis: effective RFs smaller than we thought

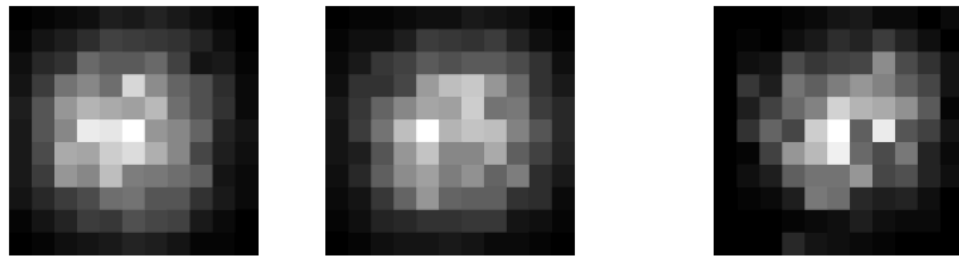
- **Effective Receptive Field (ERF)** of output unit: the region containing any input with non-negligible impact
- non-negligible: region of impact within 2-standard deviation of center pixel's impact
- For CNNs, we measure the impact as the scale of the partial derivatives, which can be computed

Effective Receptive Fields

ERF:

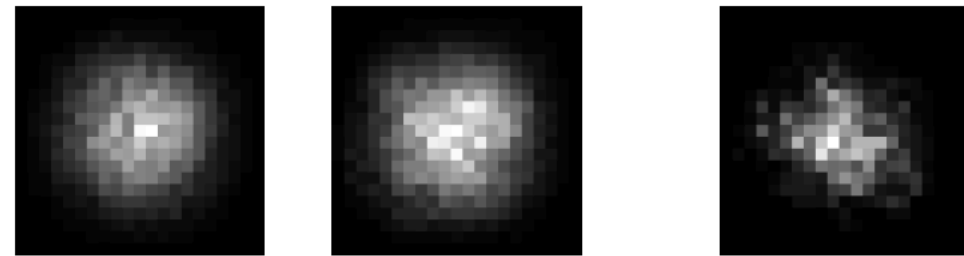
- Gaussian distribution: Fourier analysis, central limit theorem
- grows $O(\sqrt{n})$ over number of layers n in deep CNNs
- occupies $O(1/\sqrt{n})$ of full theoretical RF

5 layers, theoretical RF size=11



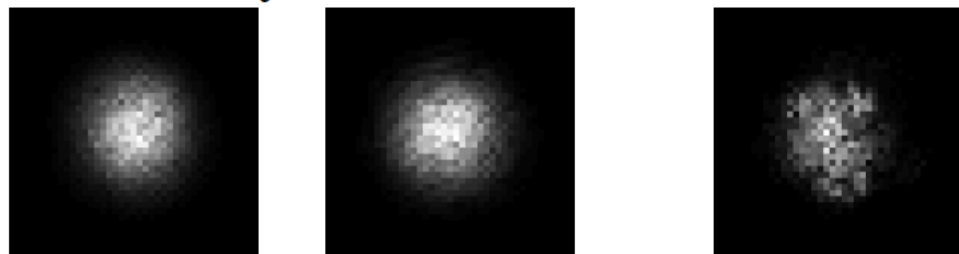
Uniform Random Random + ReLU

10 layers, theoretical RF size=21



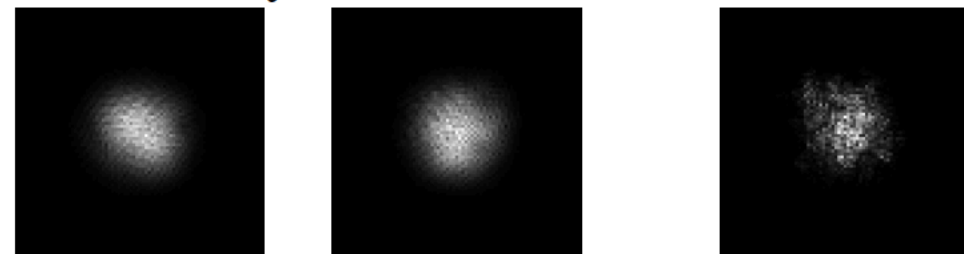
Uniform Random Random + ReLU

20 layers, theoretical RF size=41



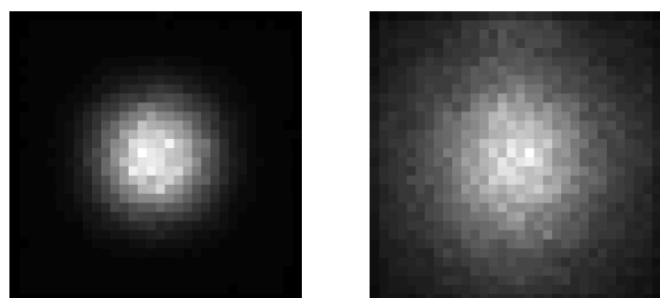
Uniform Random Random + ReLU

40 layers, theoretical RF size=81

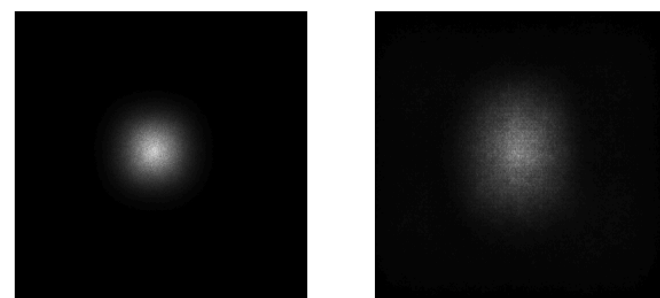


Uniform Random Random + ReLU

CIFAR 10



CamVid

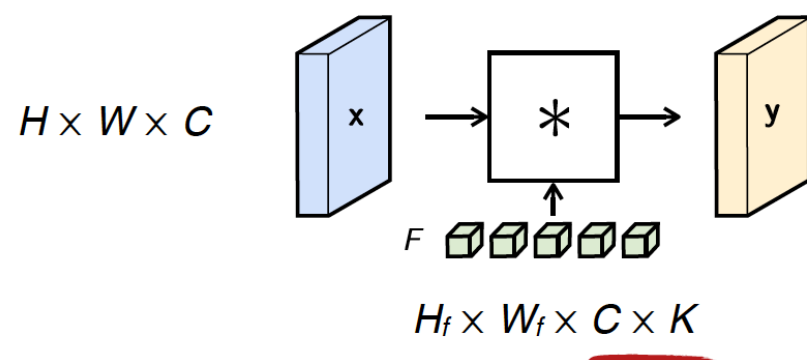


Before Training After Training Before Training After Training

[Luo, Li, Urtasun, Zemel 2016]

Enlarging Effective Receptive Fields

- Better initialization: diffuse power to periphery
- Architectural
 - Replace one large filter bank with a sequence of smaller ones (fewer parameters, gain extra nonlinearity, possibly faster)
 - Sparsify connections: randomly, learned, grouped

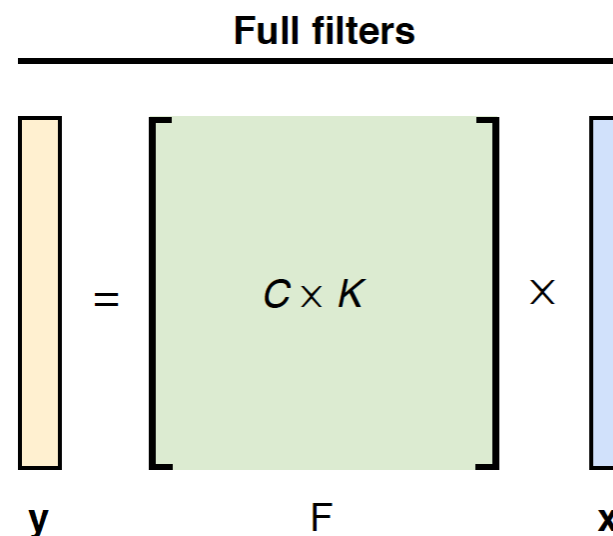


C = num. input channels

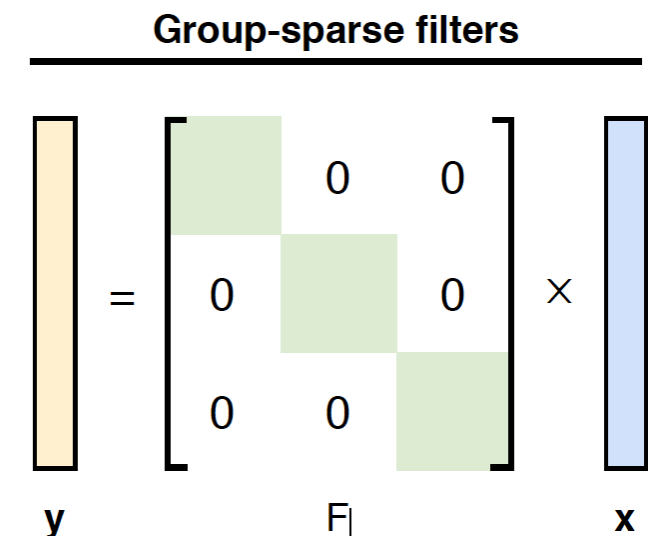
K = num. output channels

$$\text{Num. of operations} = \frac{H \times H_f}{\text{stride}} \times \frac{W \times W_f}{\text{stride}} \times \underline{C \times K}$$

$$\text{Num. of parameters} = H_f \times W_f \times \underline{C \times K}$$



complexity: $C \times K$



complexity: $C \times K / G$

[Vedaldi]

Outline

- Quick summary of convolutional networks
- Recent developments
- **What is being learned?**
- Applications:
 - Semantic segmentation
 - Image understanding
 - Few-shot learning

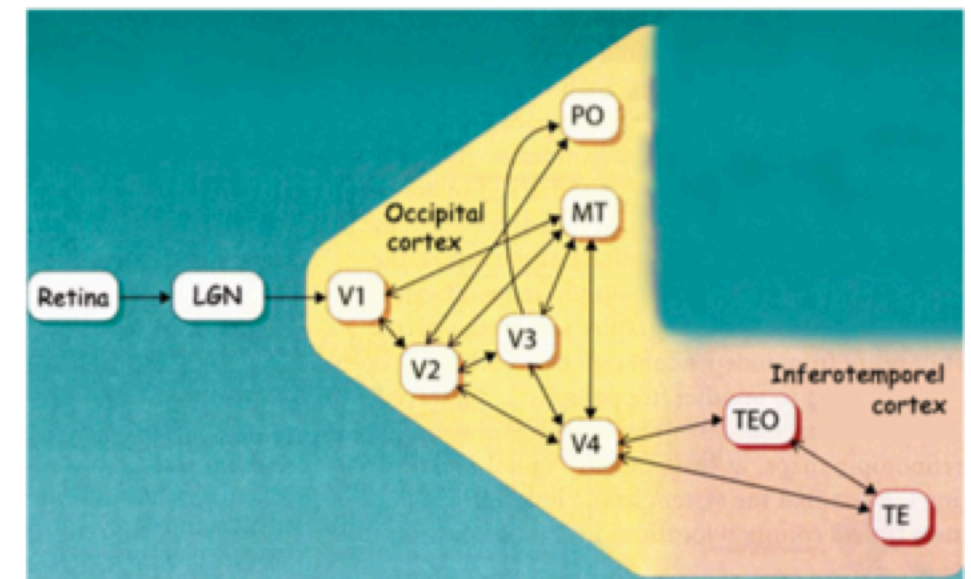
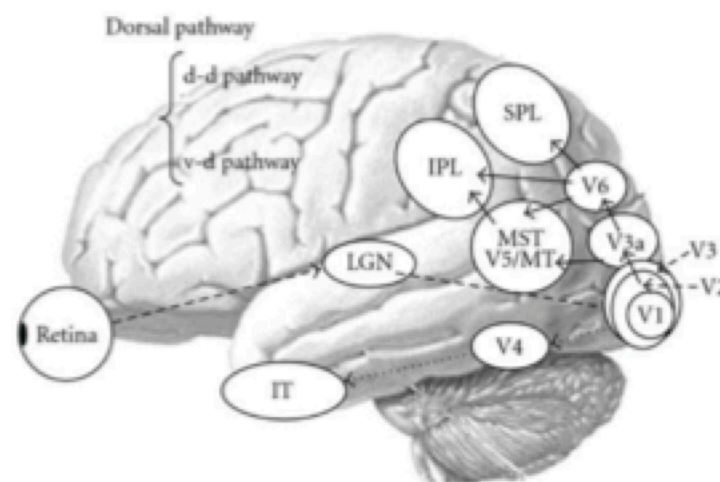
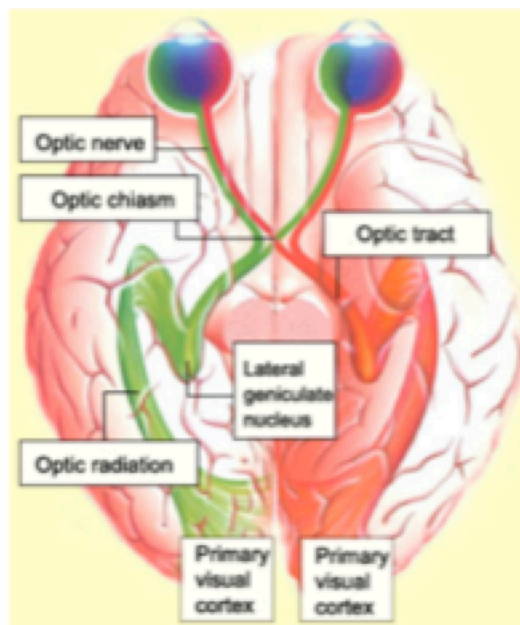
Representations in CNNs

What do we know about the these?

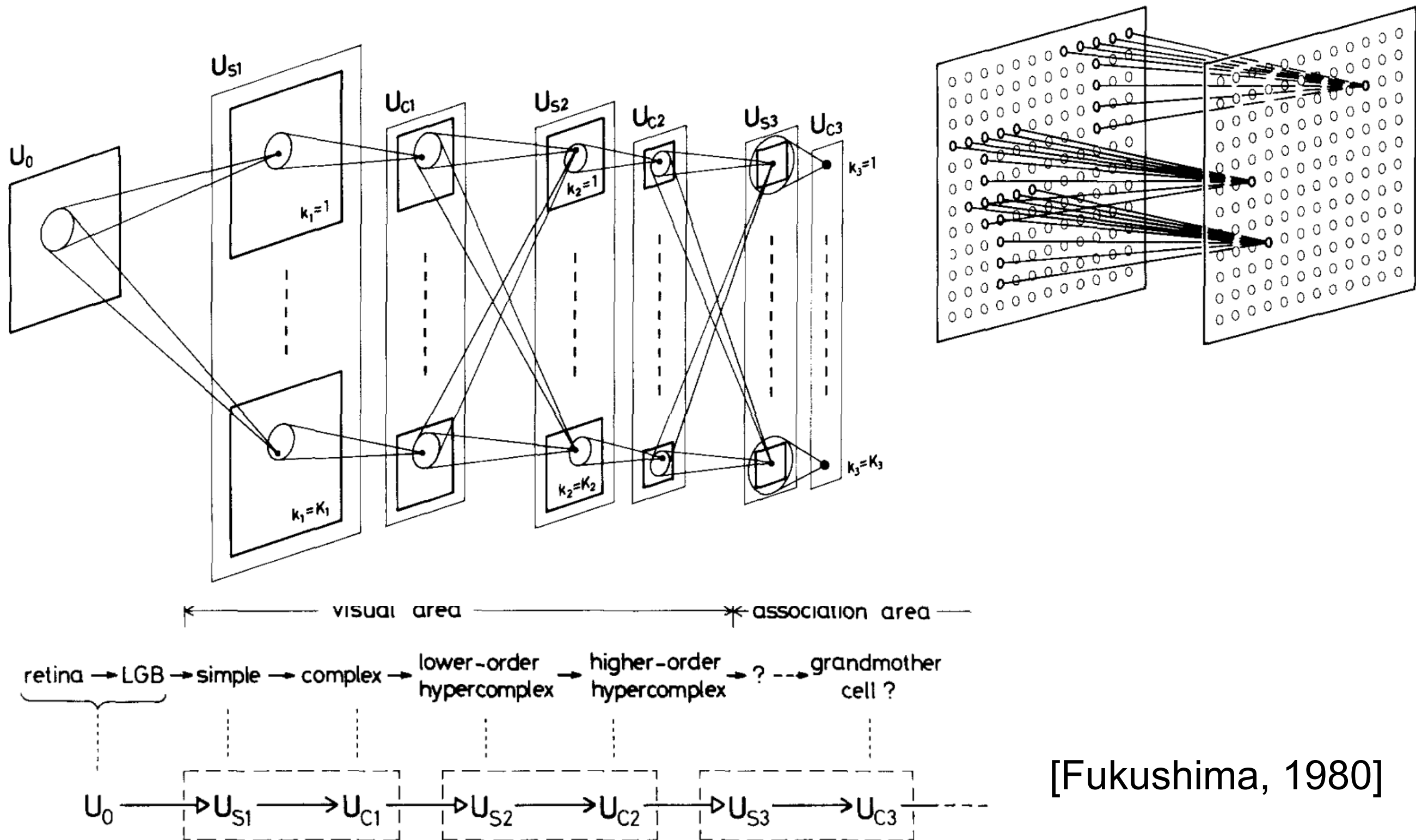
- Relevance to/from biology?
- Visualizations
- Theory & analysis

CNNs & Biology

- ConvNet inspired by visual neuroscience
 - Classical notions of simple and complex cells
 - Architecture similar to LGN-V1-V2-V4-IT hierarchy in visual cortex ventral pathway
 - LGN: lateral geniculate nucleus receives input from retina
 - 30 different areas of visual cortex, V1 & V2 principal
 - Infero-temporal cortex important in object recognition



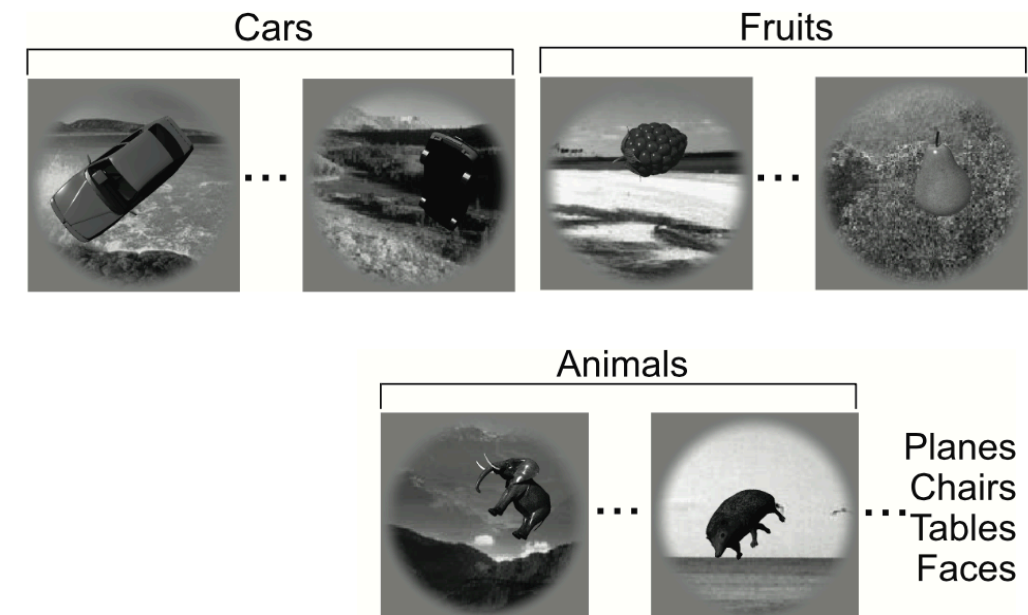
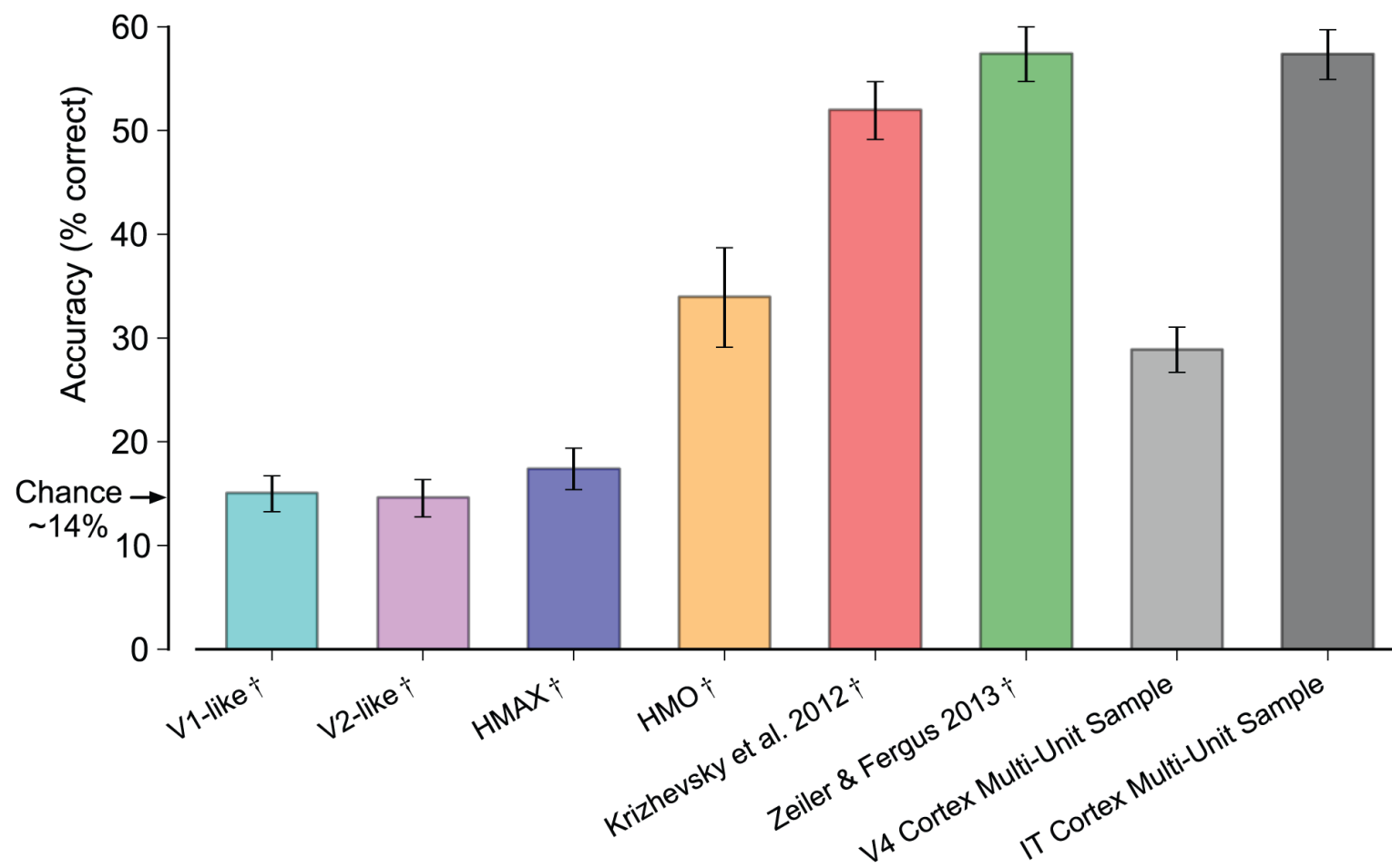
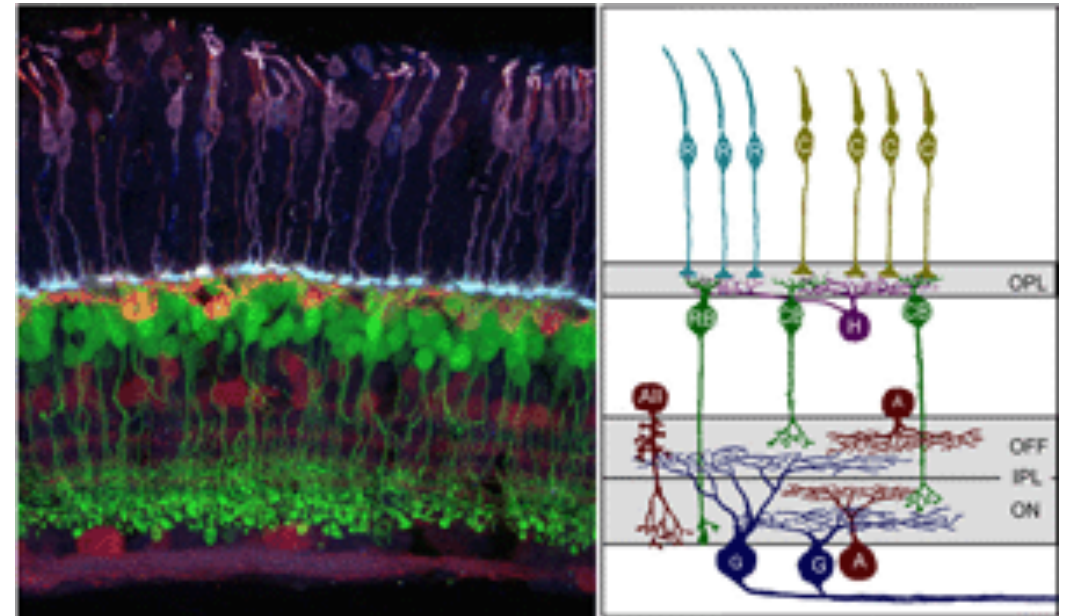
Original CNN: Bio-inspired



[Fukushima, 1980]

CNNs & Biology

- Some similarities with biology
 - Layered architecture
 - Local connectivity
- But myriad of differences
- Yet performance is somewhat analogous



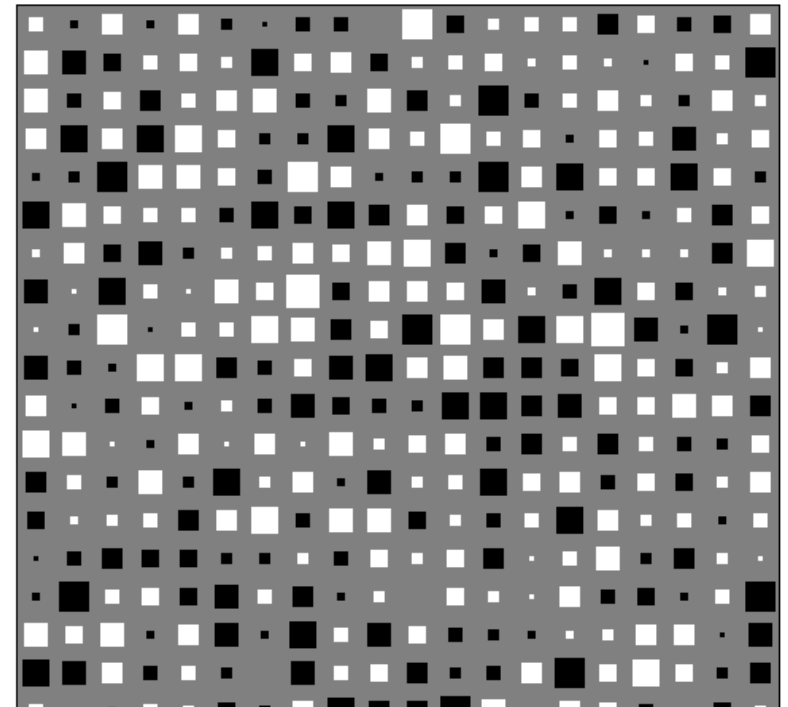
[Cadieu et al., 2014]

Visualizing the Representations

Can directly plot the filters:

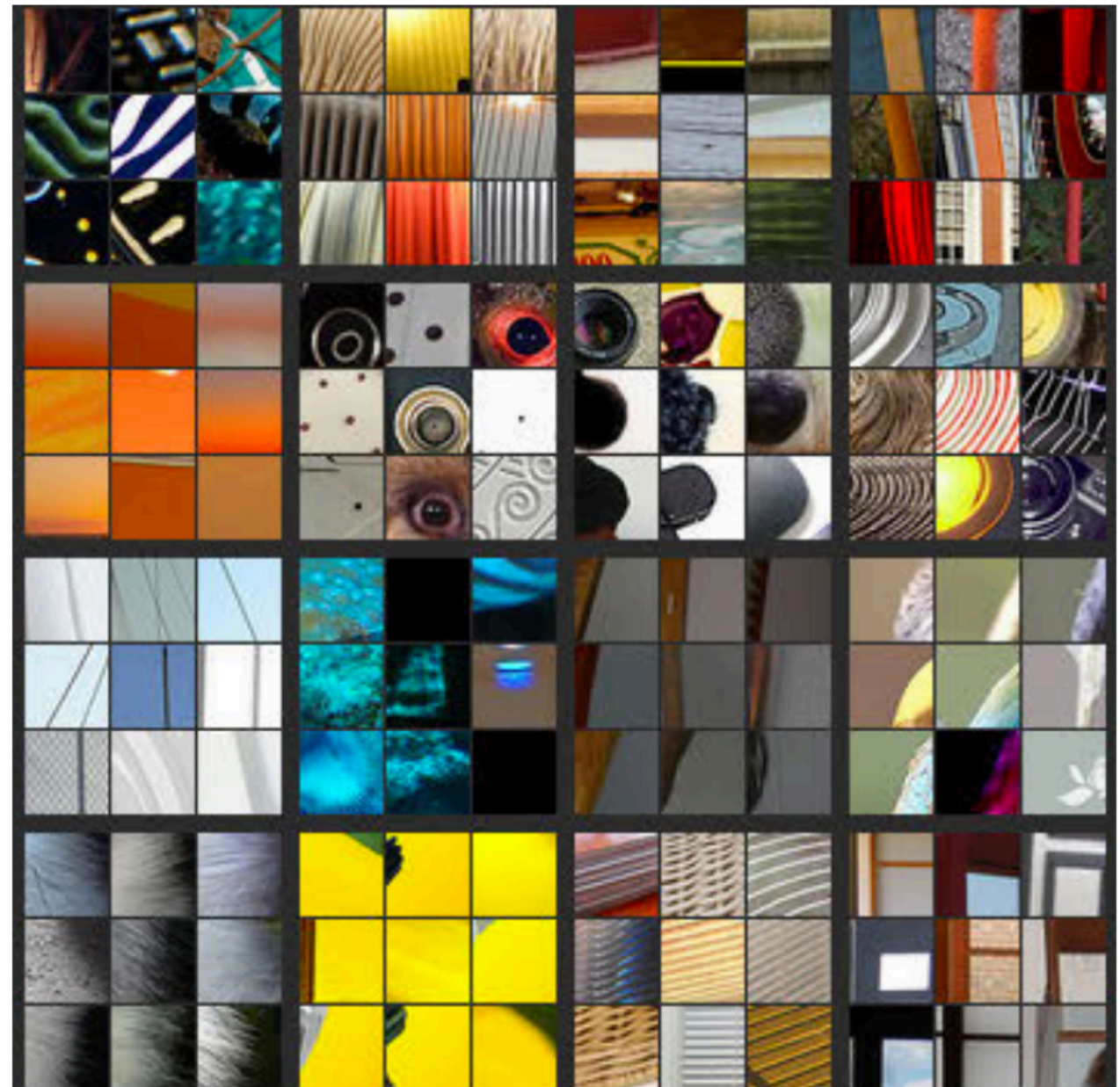
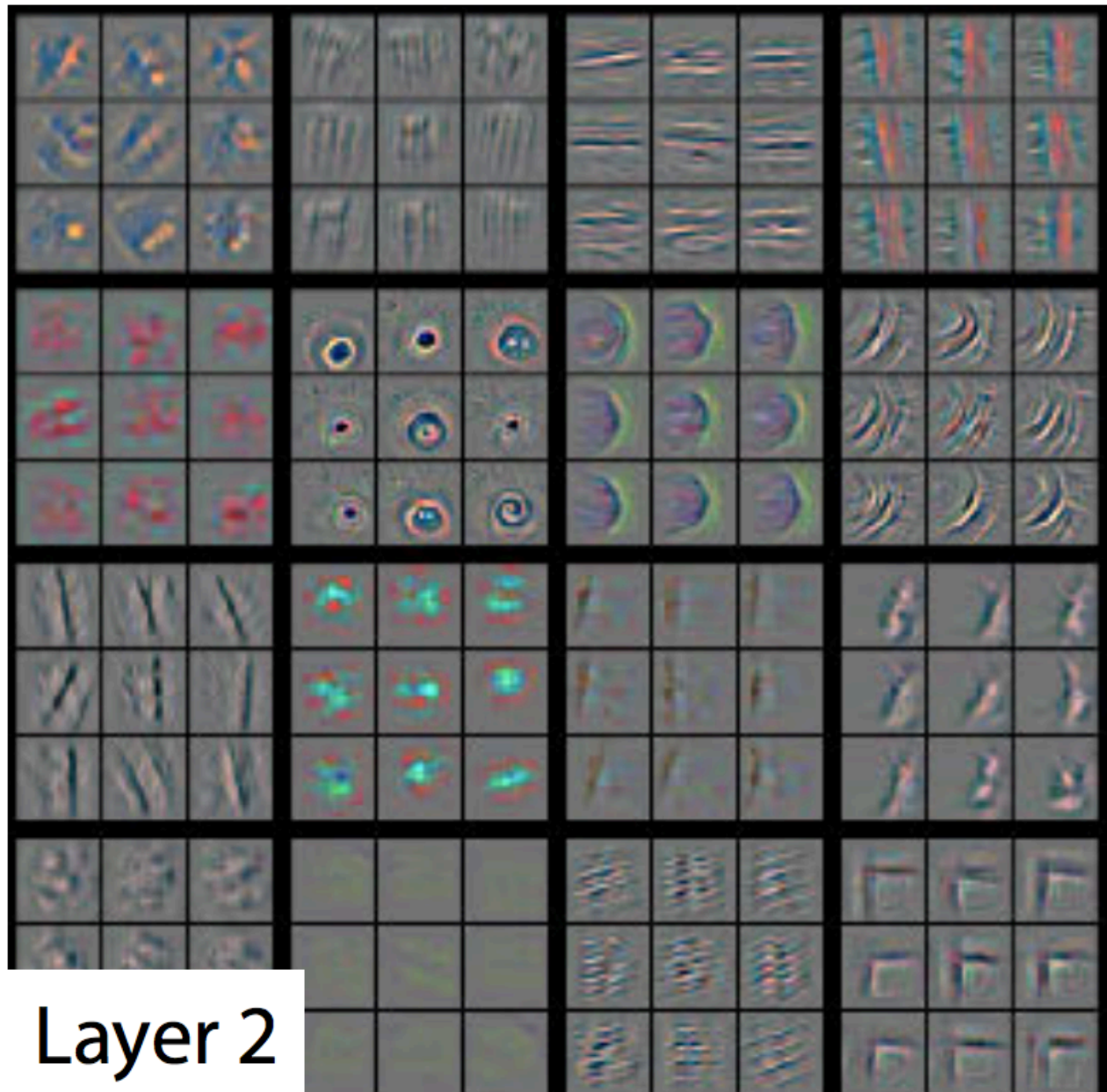
- Hinton diagram

- AlexNet, first layer



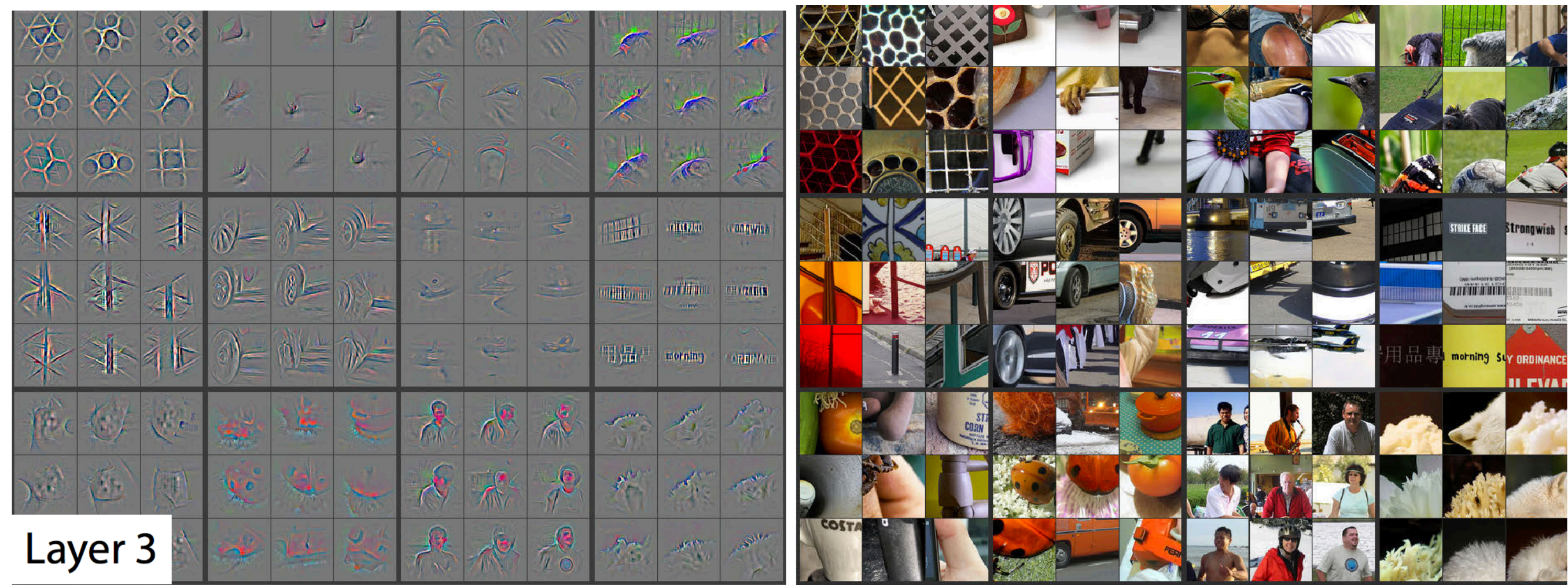
Visualizing the Representations

Can directly plot the filters (AlexNet, first layer)

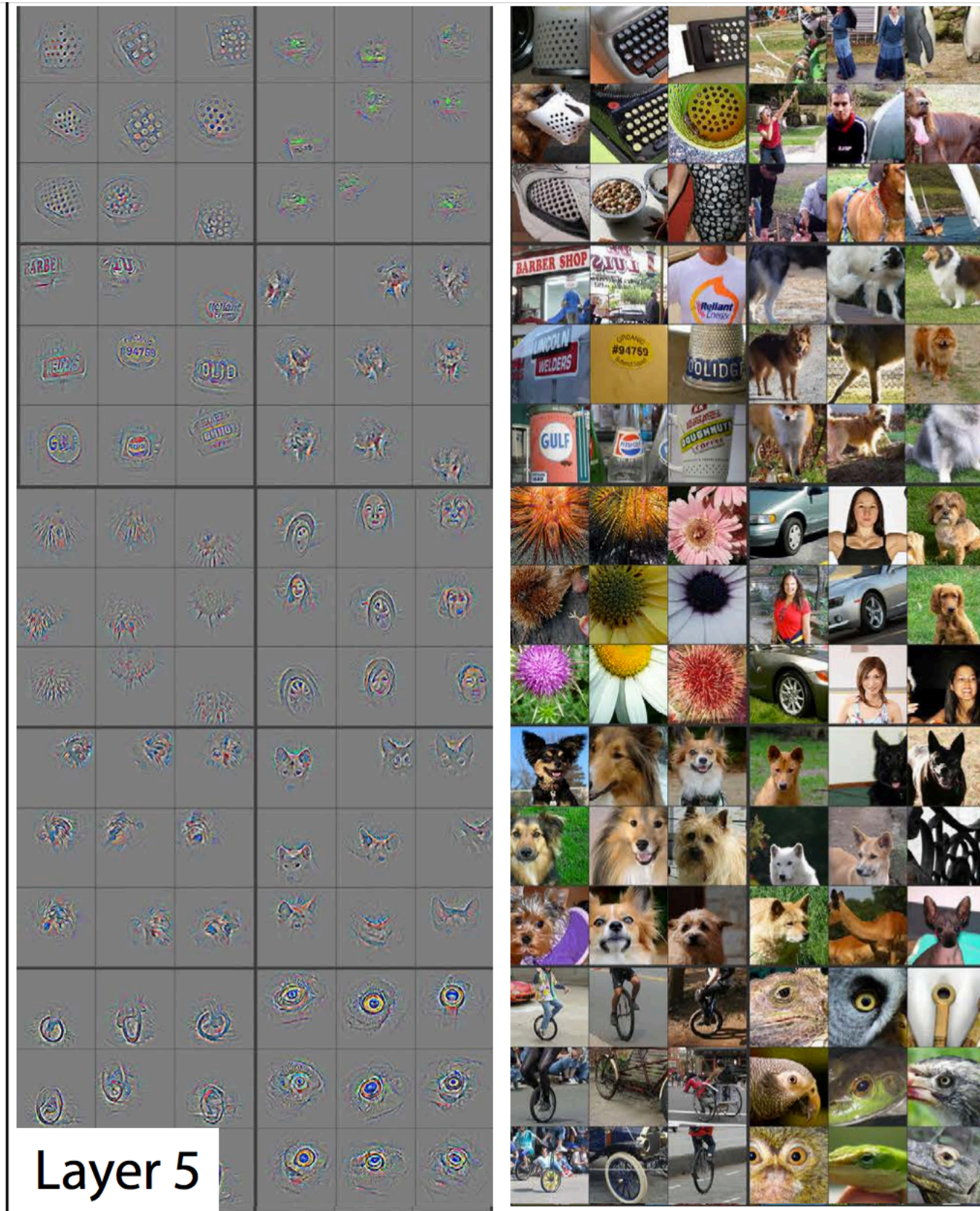
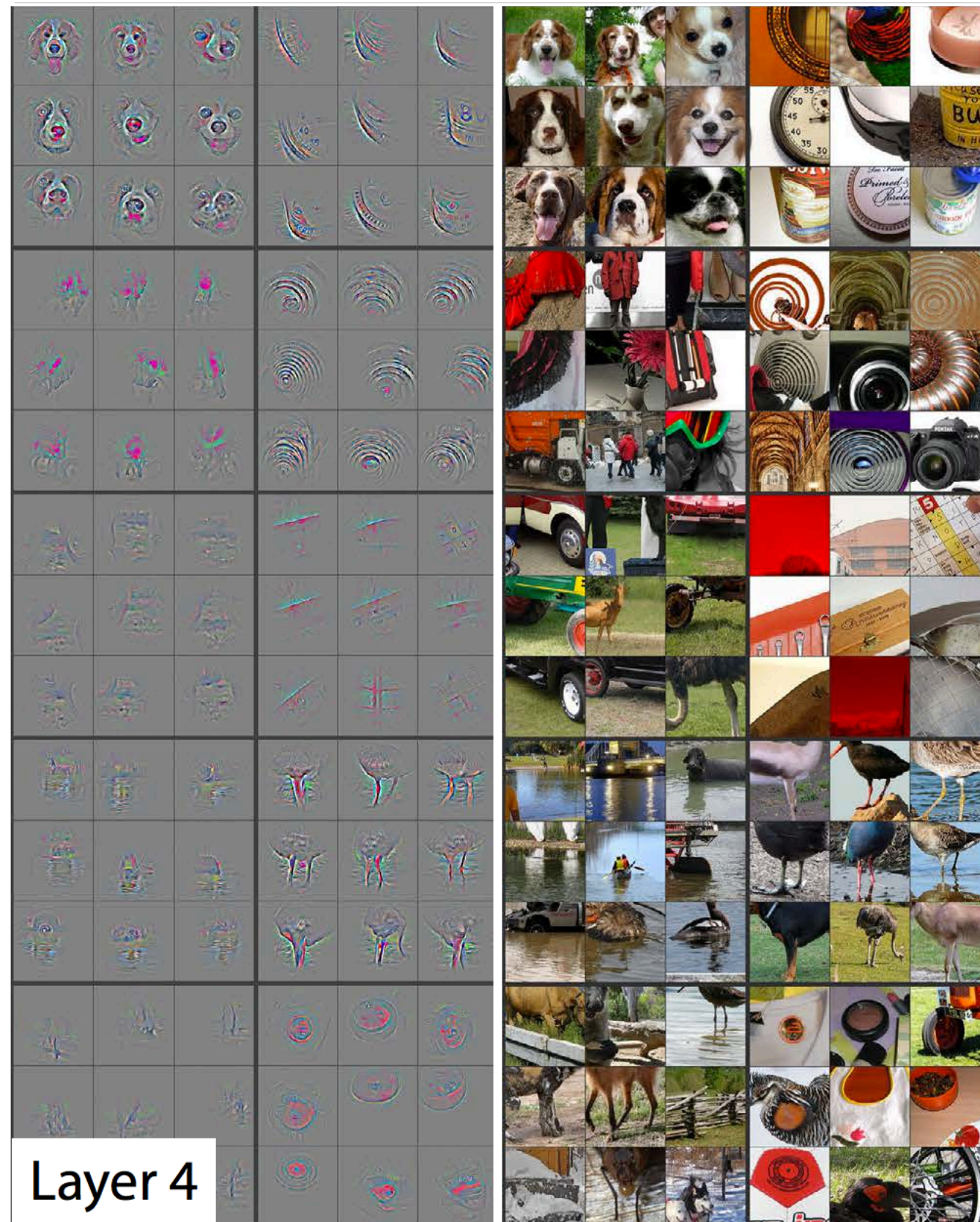


[Zeiler & Fergus, 2013]

Visualizing the Representations

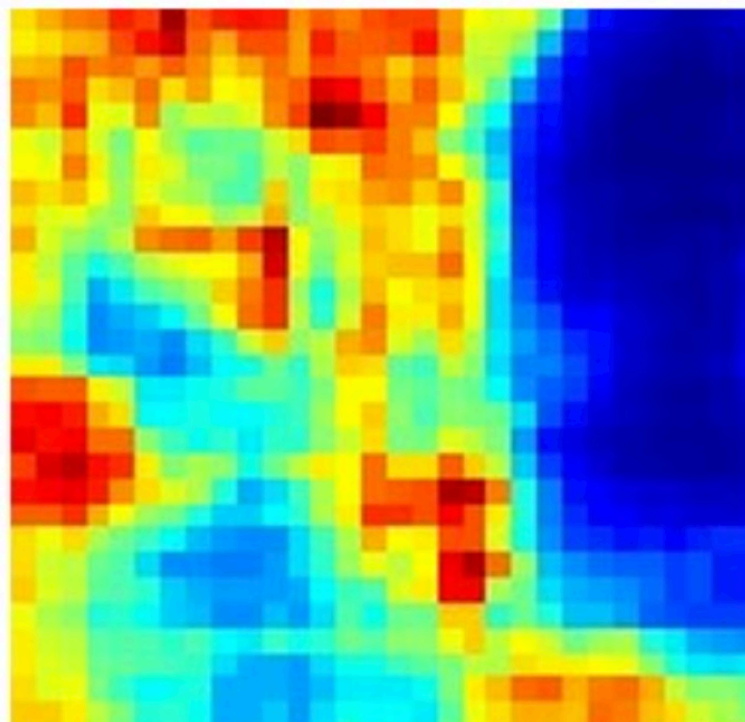
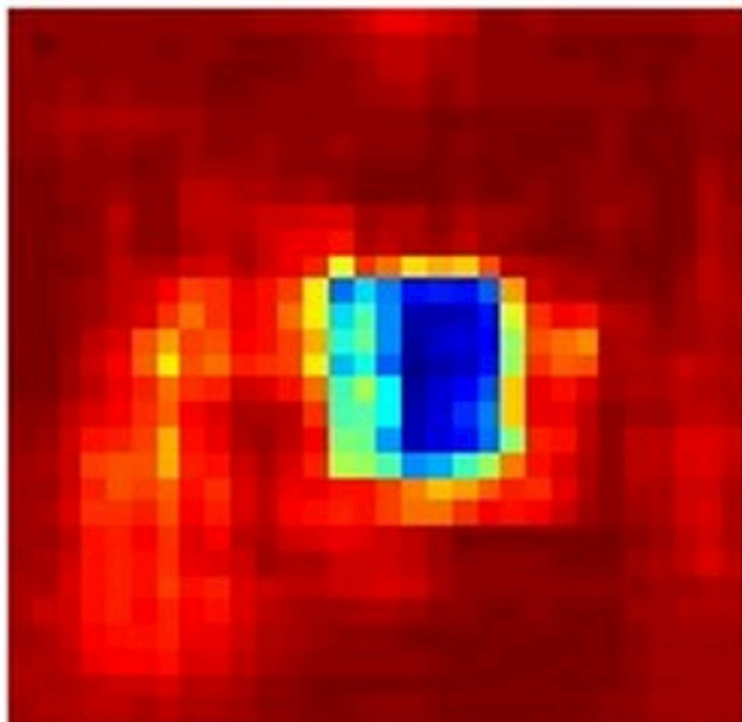


Visualizing the Representations



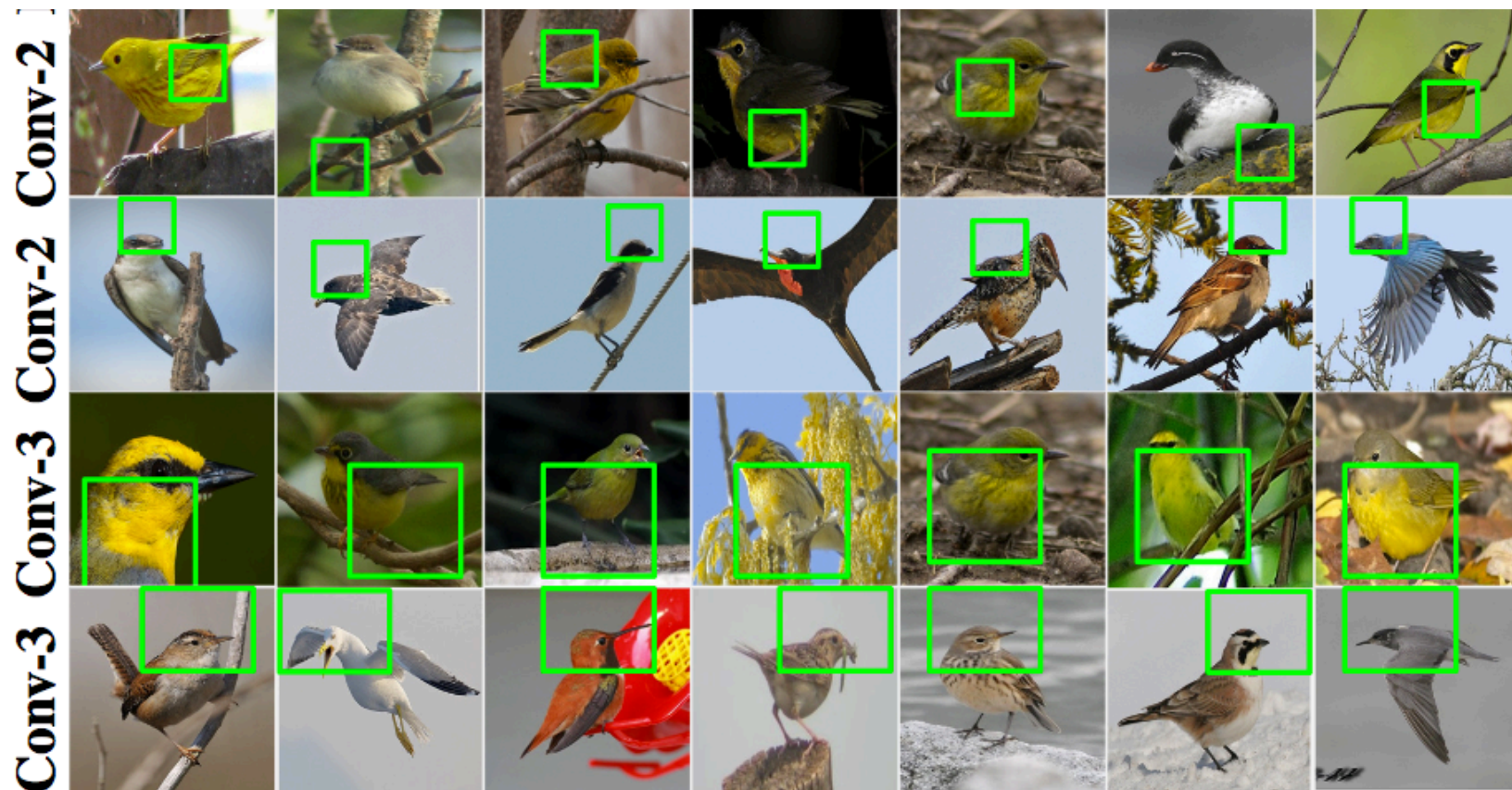
Analyzing the Representations

RF study: mask portion of image, examine effects on responses of unit, such as output



Parts-Based Representations

- One aim is to find parts in location-invariant way
- Can form clusters across space to encourage this

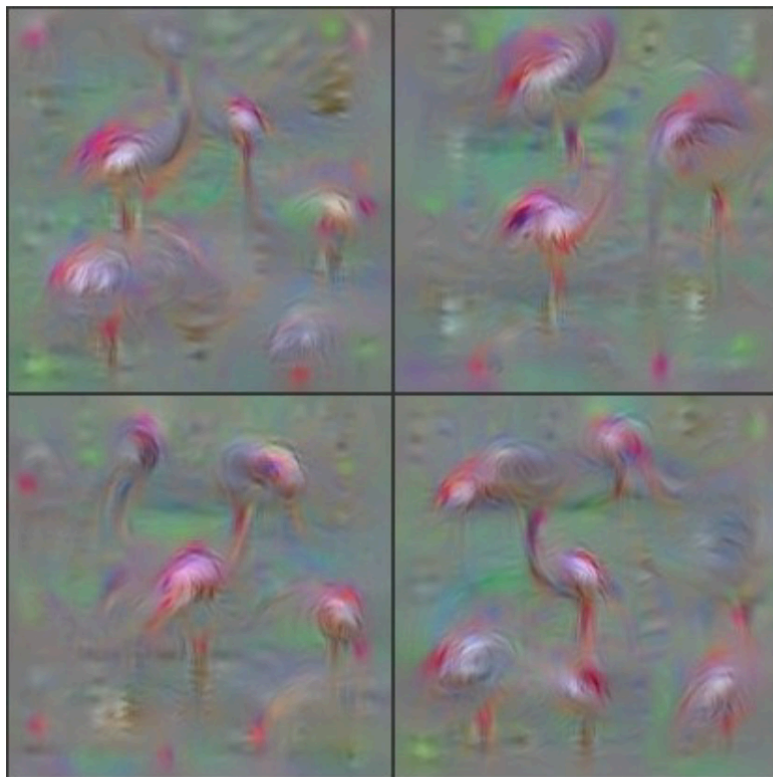
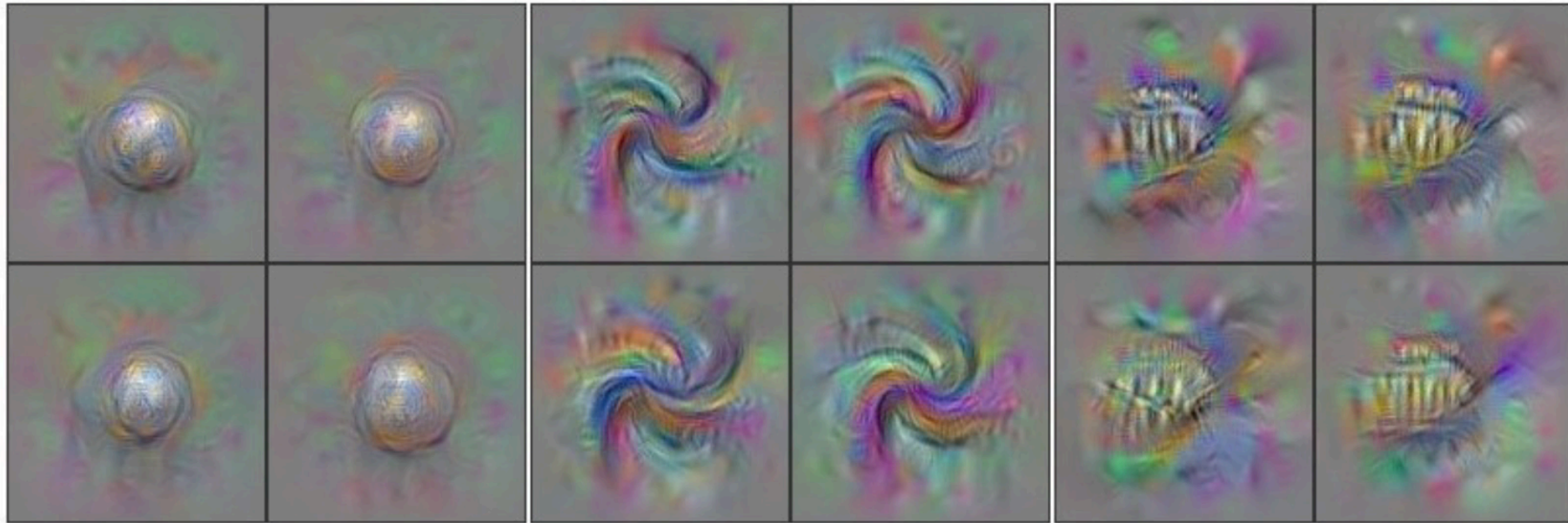


[Liao, Schwing, Zemel, Urtasun, 2016]

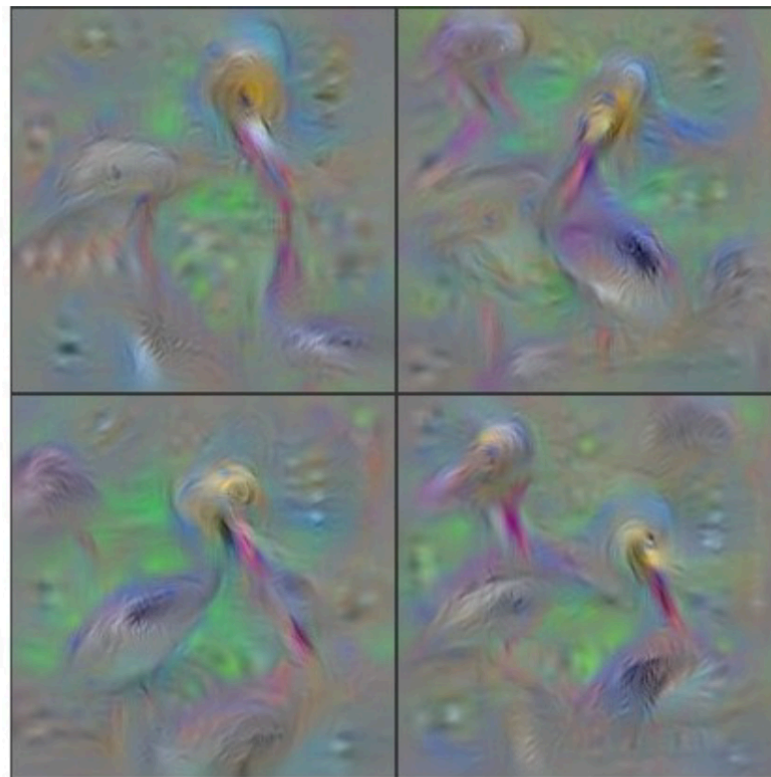
Analyzing the Representations

- Synthesize input maximally activates the unit, via gradient ascent
- Natural image priors play important role

Layer 4



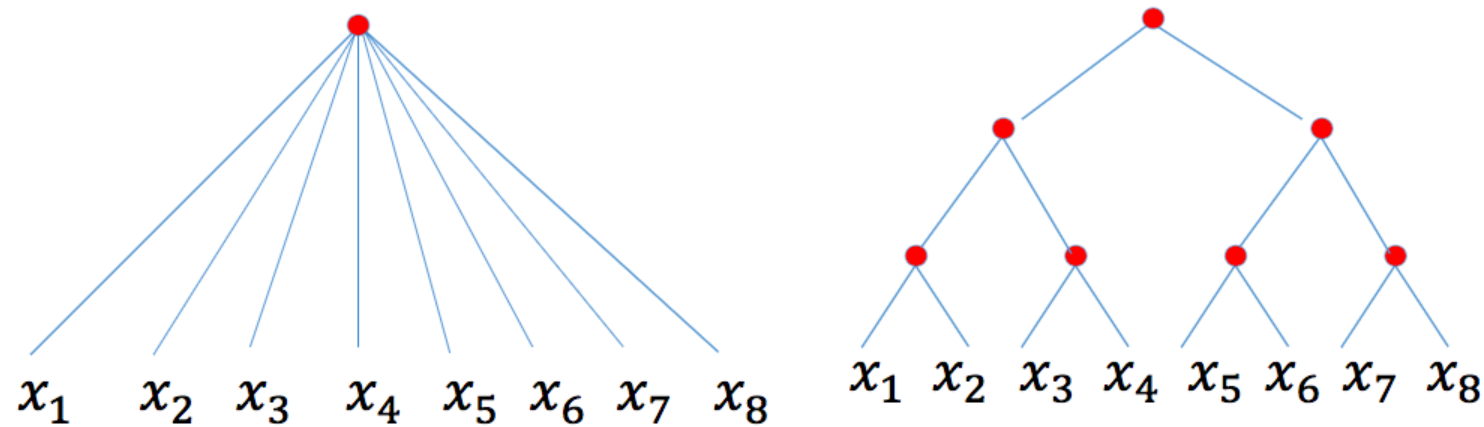
Flamingo



Pelican

[Yosinski et al., 2014]

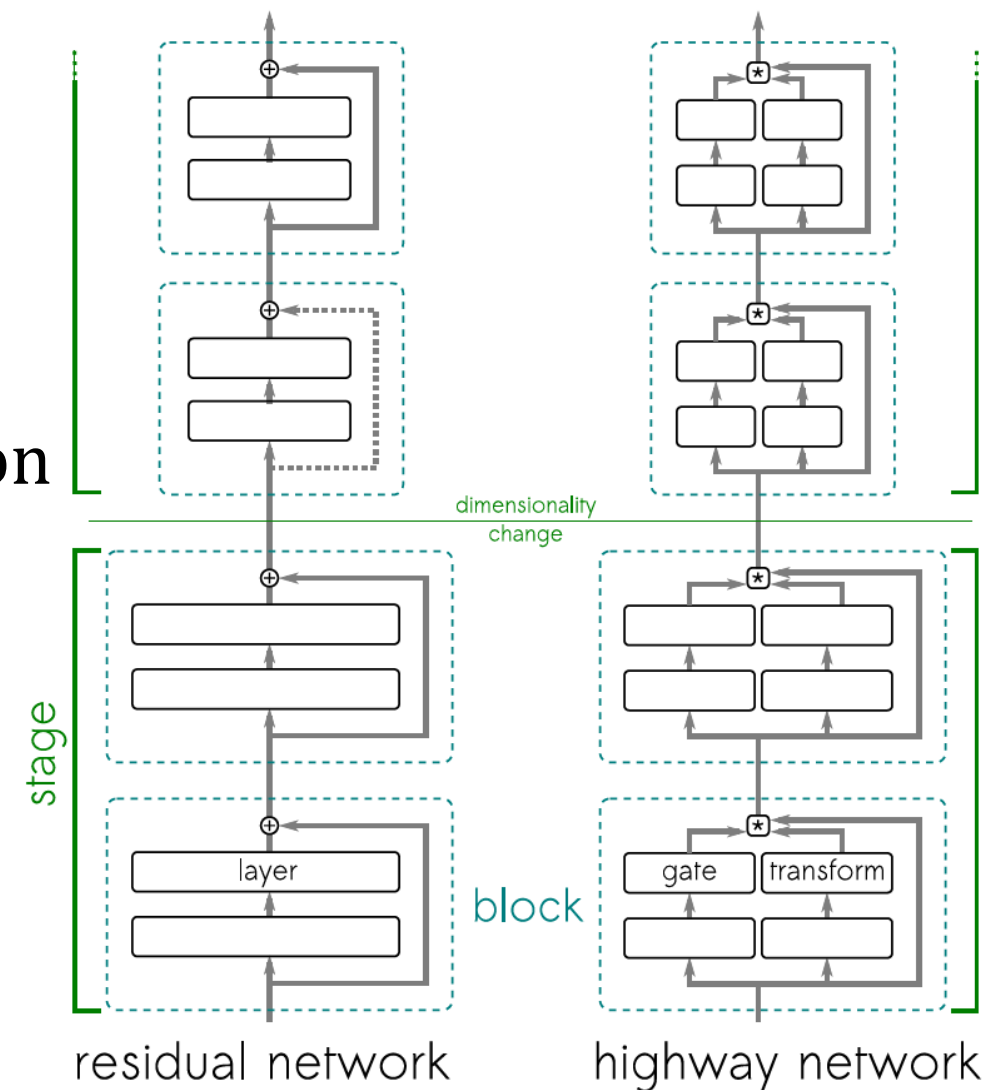
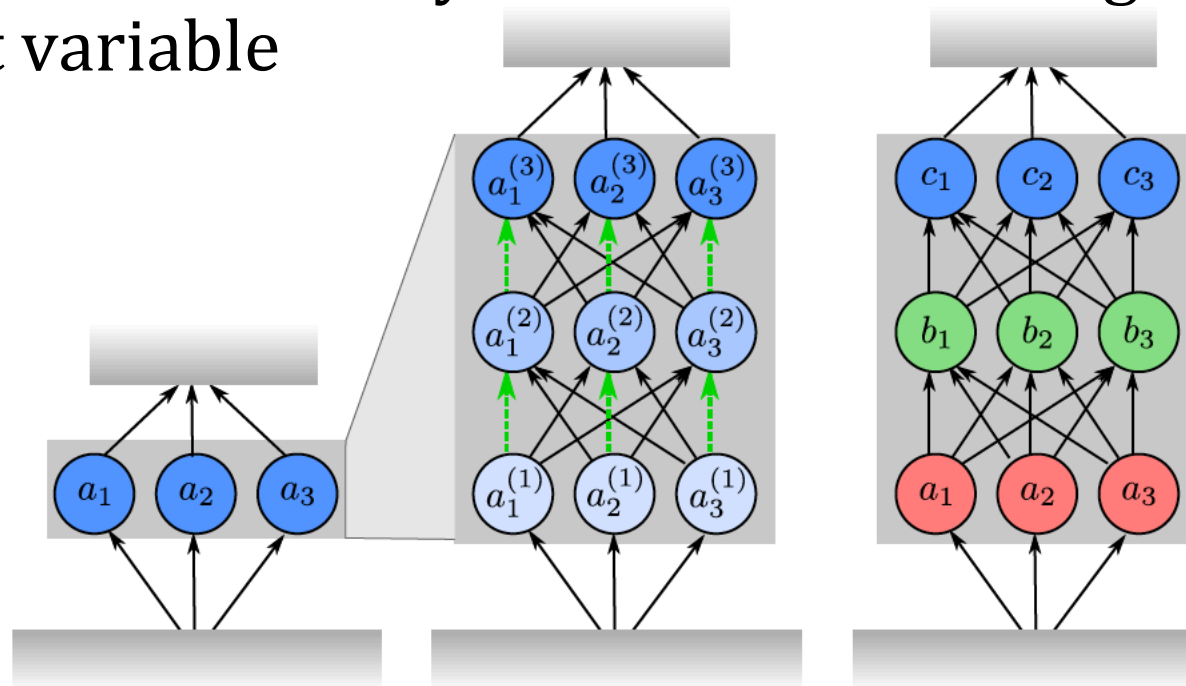
Theory of CNNs



- Both S_n and D_n are universal approximators of $f(x_1, \dots, x_8)$
 - Deep network provides better degree of approximation for **compositional** functions
$$f(x_1, \dots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$$
 - requires fewer trainable parameters to achieve same accuracy
- $\text{dist}(f, S_n) = \mathcal{O}(n^{-r/d}).$
- $\text{dist}(f, D_n) = \mathcal{O}(n^{-r/2}).$
- If the target function is scalable (same algorithm applies when input scale changes) and shift-invariant, then deep CNNs are natural approximators

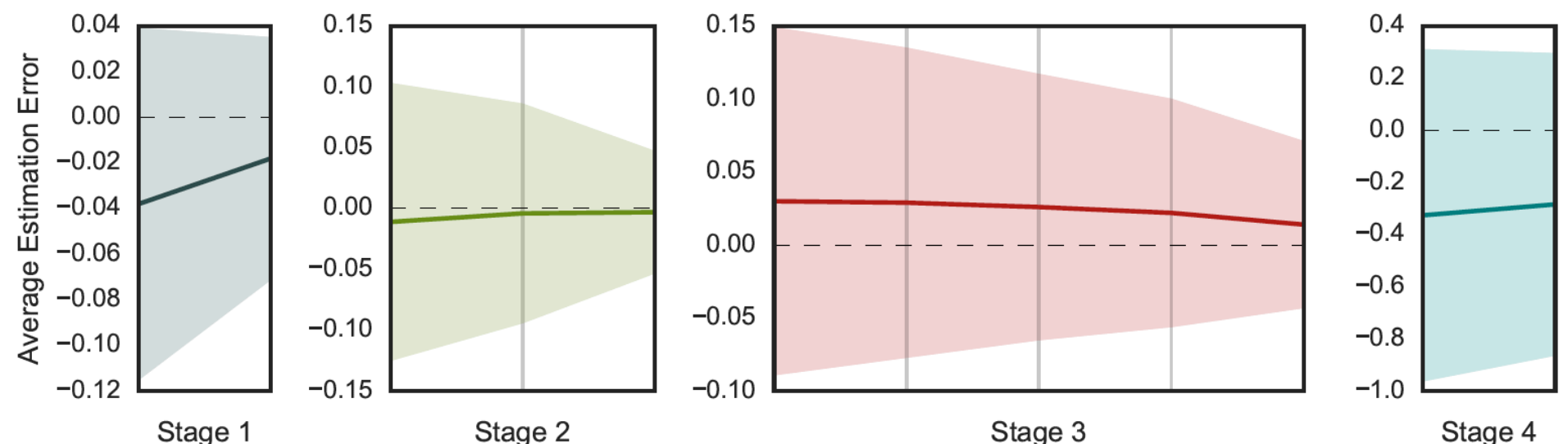
Representations in very deep nets

- Effects of lesioning, shuffling layers in highway, residual networks suggest alternative view of learned representation
- Unrolled iterative estimation of representations, formed in stages
- Units in different layers are all estimating common latent variable



$$\mathbb{E}_{\mathbf{x} \in \mathbf{X}} [a_i^k - A_i] = 0.$$

[Greff, Srivastava,
Schmidhuber, 2017]

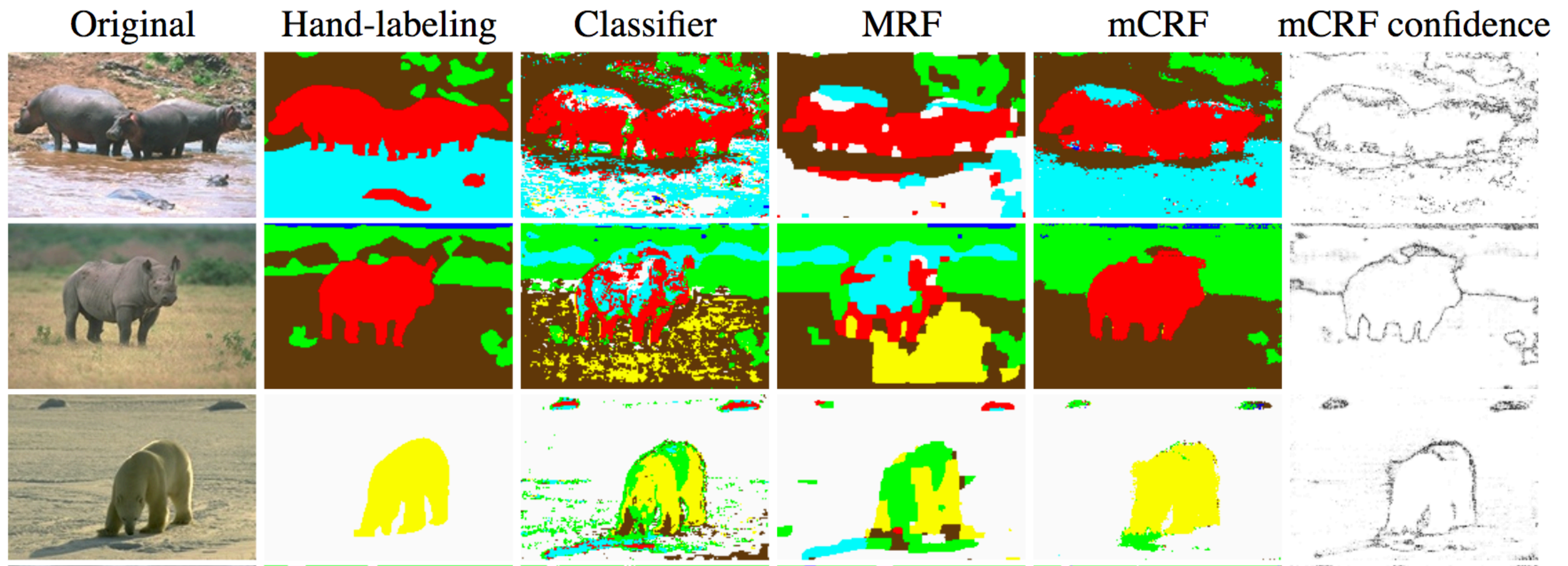


Outline

- Quick summary of convolutional networks
- Recent developments
- What is being learned?
- **Applications:**
 - **Semantic segmentation**
 - Image understanding
 - Few-shot learning

Applications: Semantic Segmentation

- CNN classifier predicts pixel labels:
 - 3-layers, sigmoid units, weight decay
- Utilize CRF to clean up
- Corel dataset: 60 train, 40 test; 80x120 pixels

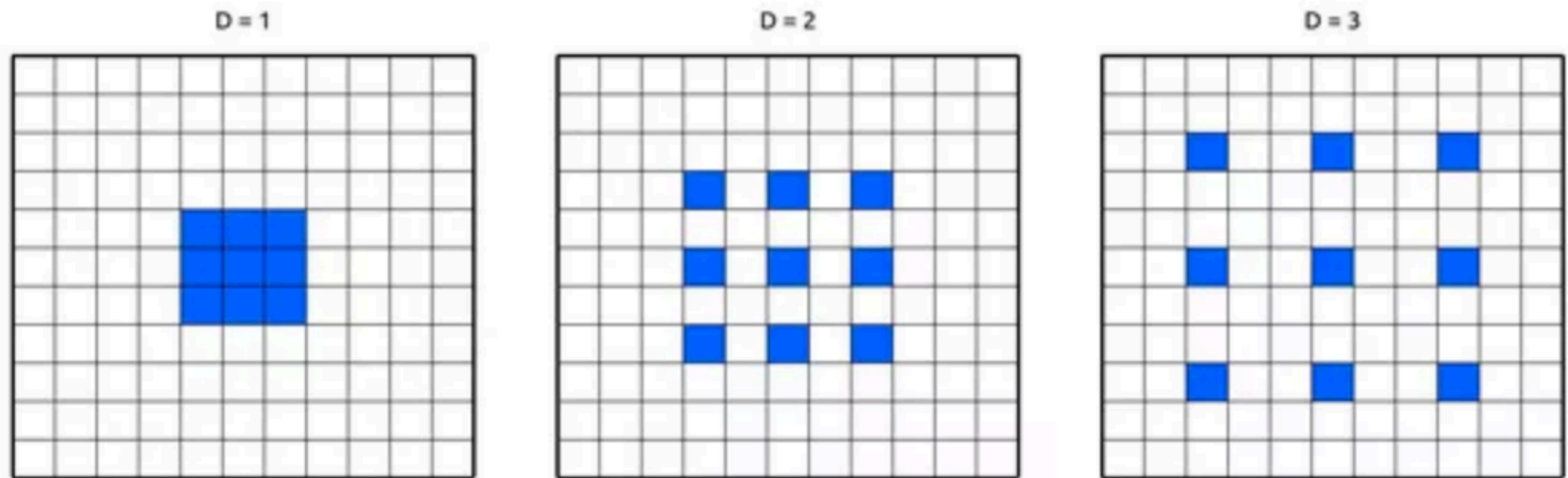


[He, Zemel, Carreira-Perpinan, 2004]

- MS-COCO: 80 object categories, 200k Images, 1.2M instances

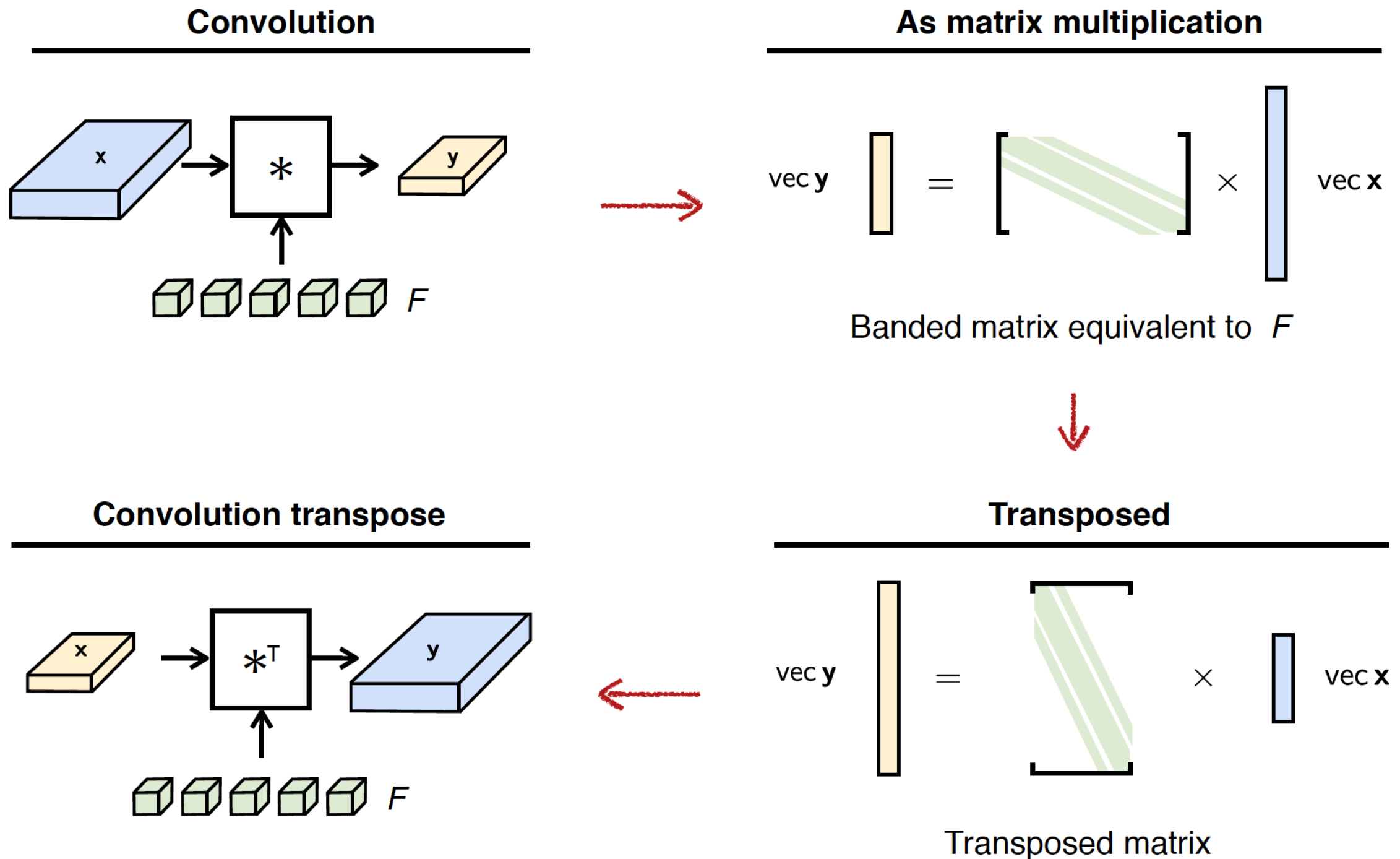
Up-sampling with convolutions

- Standard CNN: local connectivity, pooling \rightarrow down-sample input
- To predict dense pixel labels, utilize up-sampling
 1. Fractional stride
 2. Dilated convolutions

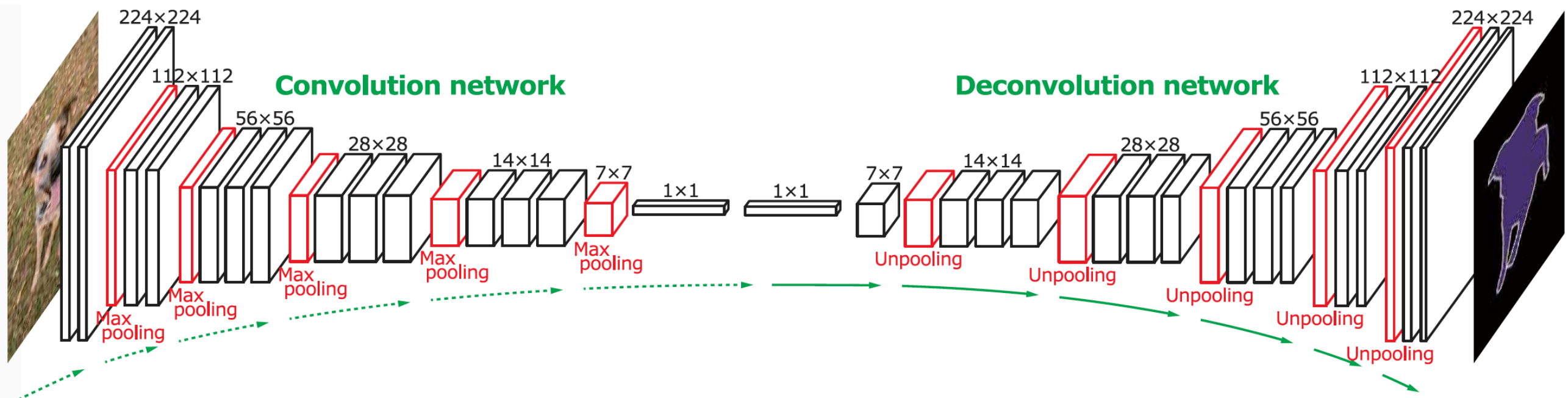
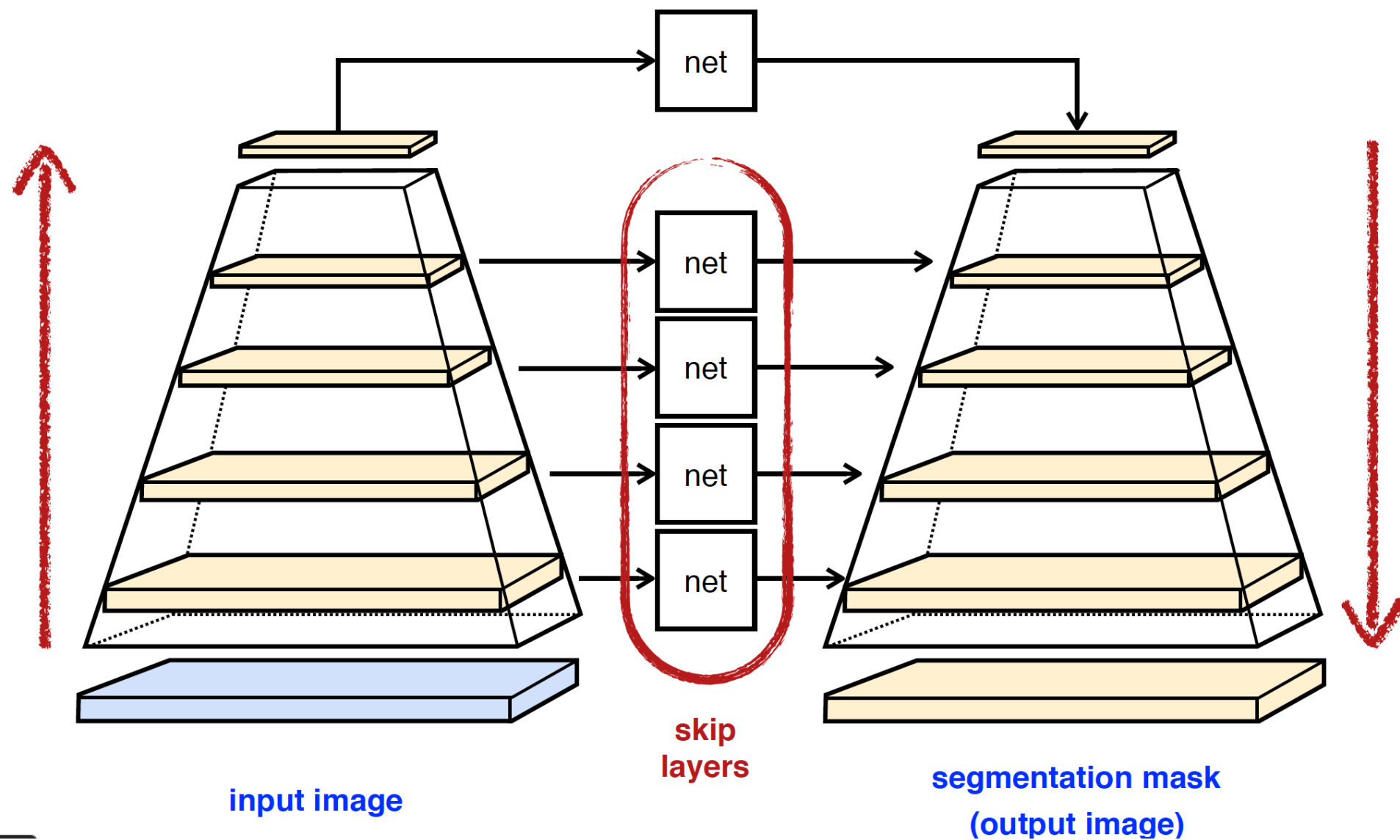


De-convolution

3. Deconvolution can be formulated as convolution transpose



Applying to semantic segmentation



Outline

- Quick summary of convolutional networks
- Recent developments
- What is being learned?
- **Applications:**
 - Semantic segmentation
 - **Image understanding**
 - Few-shot learning

Jamie Kiros



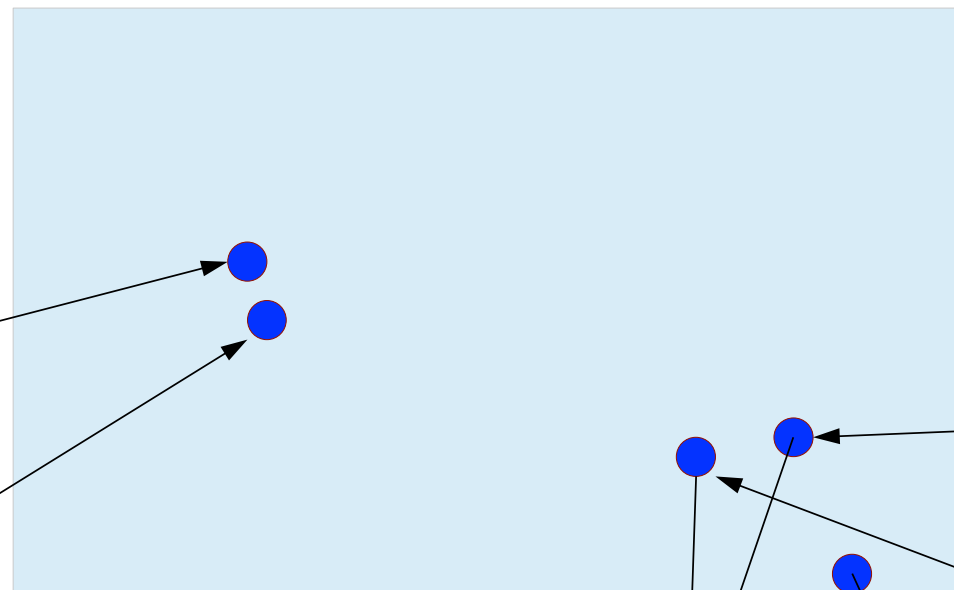
Captioning via Image/Text Embedding



A castle and
reflecting water



A ship sailing
in the ocean



Joint
space

Minimize:

$$\begin{aligned} \text{images} \longrightarrow & \sum_{\mathbf{x}} \sum_k \max\{0, \alpha - s(\mathbf{x}, \mathbf{v}) + s(\mathbf{x}, \mathbf{v}_k)\} + \\ \text{text} \longrightarrow & \sum_{\mathbf{v}} \sum_k \max\{0, \alpha - s(\mathbf{v}, \mathbf{x}) + s(\mathbf{v}, \mathbf{x}_k)\} \end{aligned}$$

[Kiros, Salak., Zemel, 2014]

Ranking experiments: Flickr8K and Flickr30K



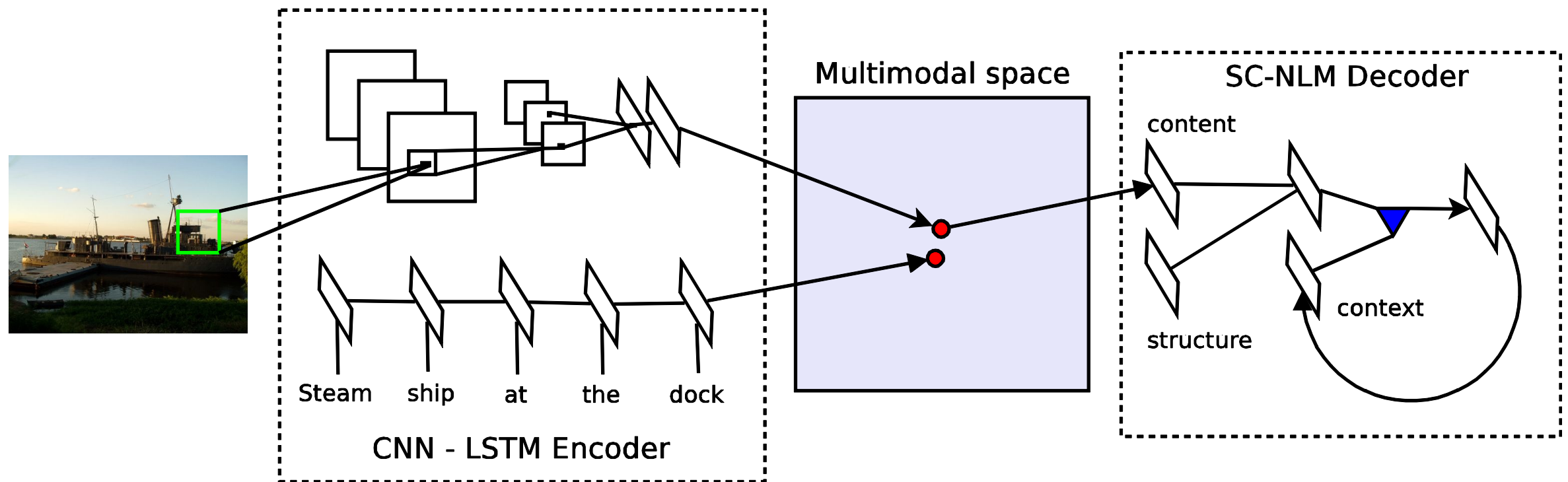
Flickr8K and Flickr30K: evaluation datasets

- Each image comes with 5 descriptions from independent annotators

Two tasks: image annotation and image search

- Hold out 1000 images and 5000 sentences
- For each image, rank all sentences
- For each sentence, rank all images
- Compute Recall @ K: fraction of times which correct truth is in top K
- Also compute median rank of the highest scoring ground truth

Generating via encoder-decoder model



Encoding → learn a joint embedding space of images and text:

- This allows us to condition on anything (images, words, phrases, etc)
- Natural definition of a scoring function (inner products in the joint space)

Decoding → use a language model that incorporates additional structure

Decoding: Adding structure



_____ (NN VBN IN DT NN)
DT

A _____ (VBN IN DT NN -)
NN

A bicycle _____ (IN DT NN - -)
VBN

A bicycle parked _____ (DT NN - - -)
IN

A bicycle parked on _____ (NN - - - -)
DT

A bicycle parked on the _____ (- - - - -)
NN

$$P(w_n | w_{1:n-1}, t_{n:n+k}, \mathbf{x})$$

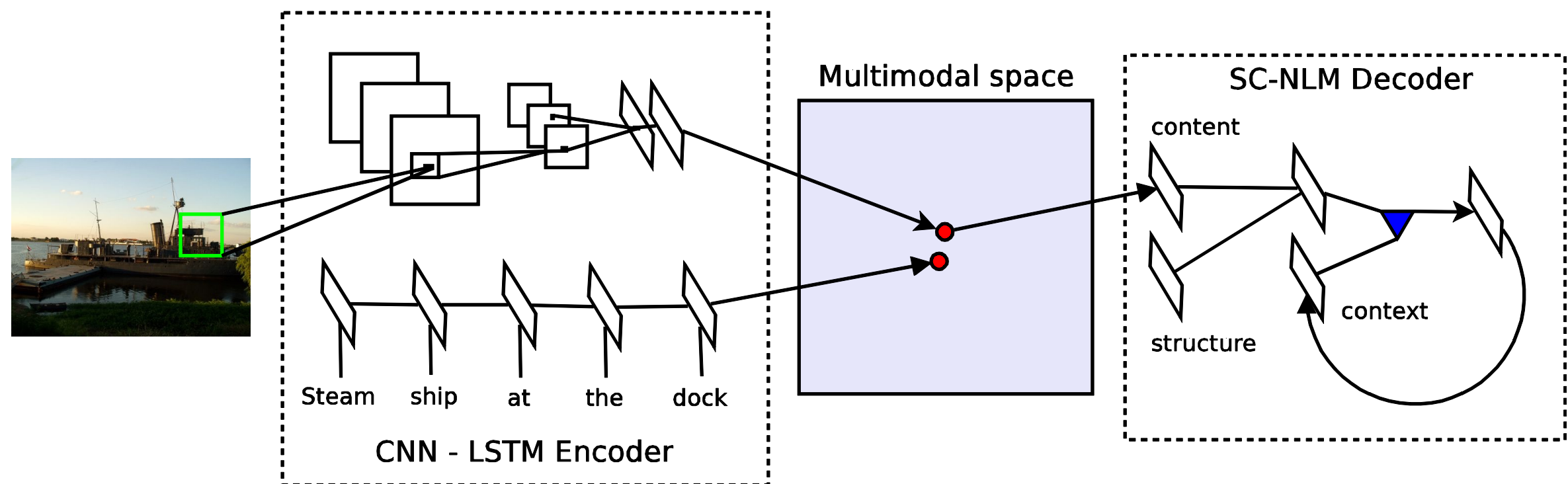
n-th
word

word
context

POS
context

[Kiros, Zemel, Salak., 2015]

Encoder-decoder model



$$P(w_n | w_{1:n-1}, t_{n:n+k}, \mathbf{x})$$

Diagram illustrating the components of the probability function $P(w_n | w_{1:n-1}, t_{n:n+k}, \mathbf{x})$:

- w_n : n-th word
- $w_{1:n-1}$: word context
- $t_{n:n+k}$: POS context

- Attribute combines image/sentence embedding with structure (forward POS soft template)

How to generate descriptions

Given an image:

- Embed the image using the conv-net
- Condition the language model on the embedding
- Sample a POS sequence from the training set
- Generate the MAP description
- Score how well it does
- Repeat many times and take the highest scoring description

Some good results - generation



L.Z

a car is parked in
the middle of nowhere .



a wooden table and chairs
arranged in a room .



there is a cat sitting on a
shelf .

a ferry boat on a marina
with a group of people .



a little boy with a bunch
of friends on the street .

Some failure types



the two birds are trying
to be seen in the water .
(can't count)



a giraffe is standing next
to a fence in a field .
(hallucination)



a parked car while
driving down the road .
(contradiction)



the handlebars are trying
to ride a bike rack .
(nonsensical)



a woman and a bottle of wine
in a garden . (gender)

Mad Libs



The (cat)
NN is in the box.

The cat is in the (box) .
NN

The cat is (sitting) in the box .
VBG

The (cute) cat is in the box .



This is a $\frac{(\text{bus})}{\text{NN}}$.

The bus is (parked) .
JJ

There is a (car) behind the bus .
NN

The tree is (on) the bus.
IN

Generate with style

(#3) Results



We were barely able to catch the breeze at the beach , and it felt as if someone stepped out of my mind . She was in love with him for the first time in months , so she had no intention of escaping . The sun had risen from the ocean , making her feel more alive than normal . She 's beautiful , but the truth is that I do n't know what to do . The sun was just starting to fade away , leaving people scattered around the Atlantic Ocean . I d seen the men in his life , who guided me at the beach once more .

Generate with style

(#3) Results



You re the only person on the beach right now
you know
I do n't think I will ever fall in love with you
and when the sea breeze hits me
I thought
Hey

Generate with style



Generated story about image
Model: Romantic Novels

“He was a shirtless man in the back of his mind, and I let out a curse as he leaned over to kiss me on the shoulder.”

He wanted to strangle me, considering the beautiful boy I’d become wearing his boxers.”

Visual Question-Answering



What color is the jacket?

- Red and blue.
- Yellow.
- Black.
- Orange.



How many cars are parked?

- Four.
- Three.
- Five.
- Six.



What event is this?

- A wedding.
- Graduation.
- A funeral.
- A picnic.



When is this scene taking place?

- Day time.
- Night time.
- Evening.
- Morning.

[Ren, Kiros, Zemel, 2016]

Outline

- Quick summary of convolutional networks
- Recent developments
- What is being learned?
- **Applications:**
 - Semantic segmentation
 - Image understanding
 - **Few-shot learning**

Modern Deep Learning:

More Data → Better Results

- AlexNet: “ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images”
- Deep Speech: “Large-scale deep learning systems require an abundance of labeled data ... we have thus collected an extensive dataset consisting of 5000 hours of read speech from 9600 speakers.”
- Neural Machine Translation: “On WMT En→Fr, the training set contains 36M sentence pairs.”
- AlphaGo: “The value network was trained for 50 million mini-batches of 32 positions, using 50 GPUs, for one week”

Few-Shot Learning

- Humans can easily recognize new categories from a single example
- How to adapt deep neural networks to new classes with few labeled examples?
- Could retrain on new classes, but significant risk of over-fitting

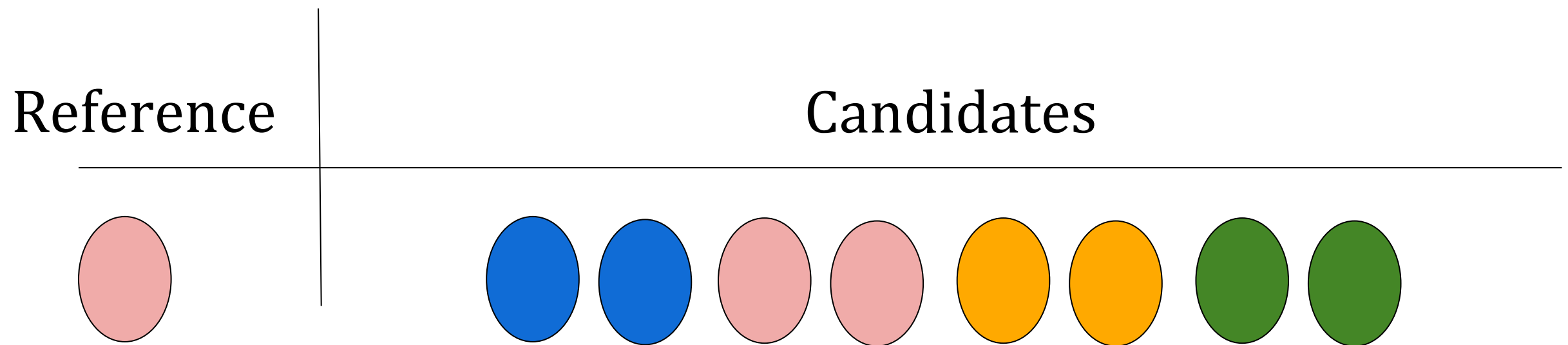


Image reproduced from: Lake, Brenden M., Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction." *Science* 350.6266 (2015): 1332-1338.

What is few-shot learning?


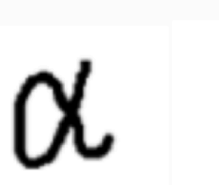
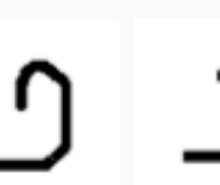
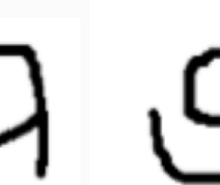

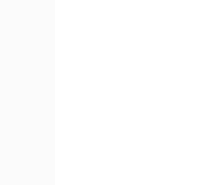



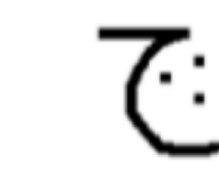
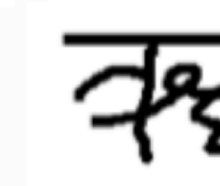
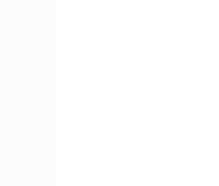
Learning how to represent objects (e.g. images) **given only a few instances of each object**

K-shot N-way Classification: Classify a reference image into one of N candidate classes, given K representatives of each class. For example, 2-shot 4-way classification:



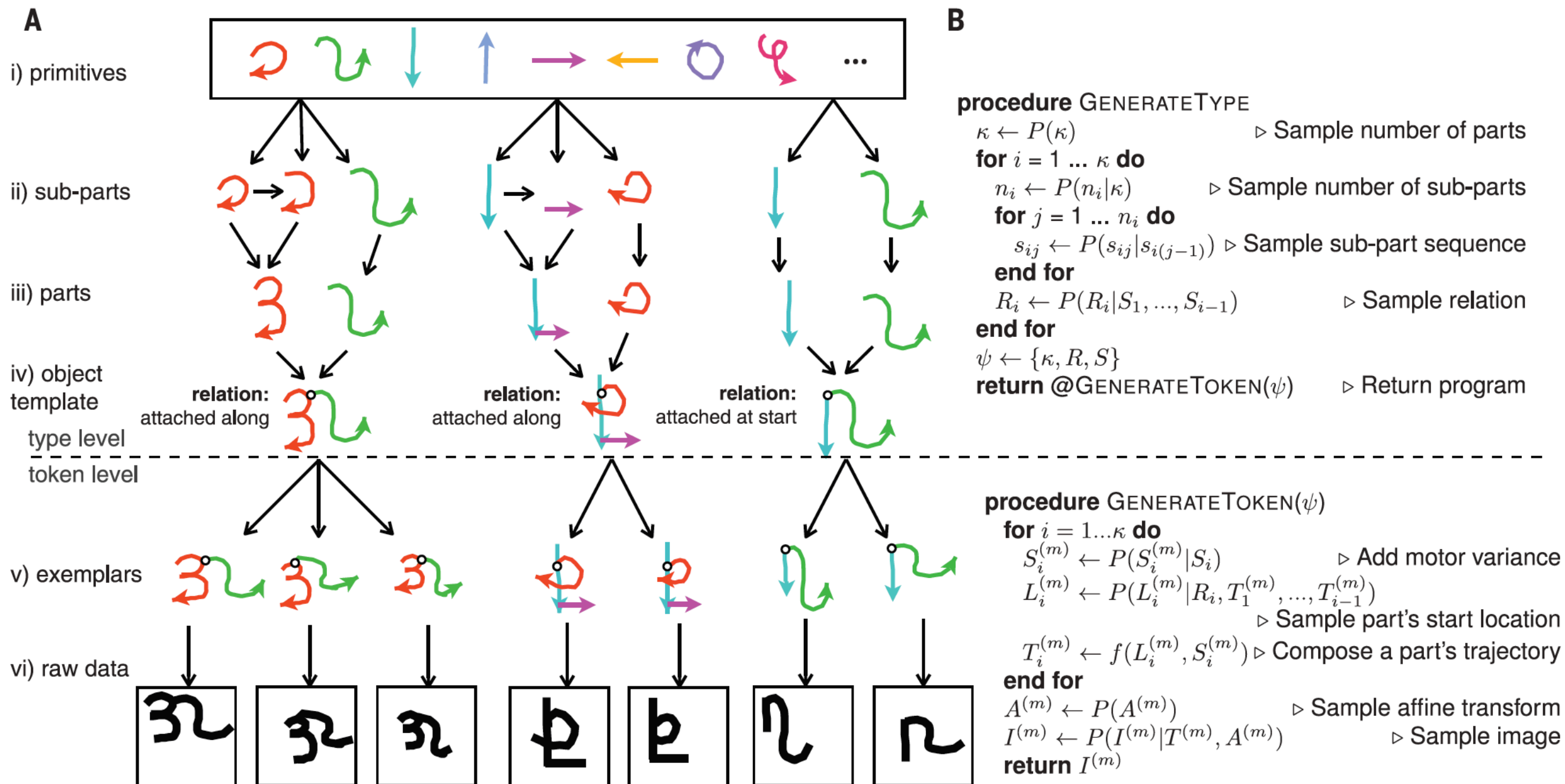
Fundamental question: **How to transfer knowledge from earlier experience to do few-shot learning?**

OmniGlott Dataset

							
							
Aurek-Besh	Futurama	Greek	Hebrew	Korean	Latin	Malay	Sanskrit

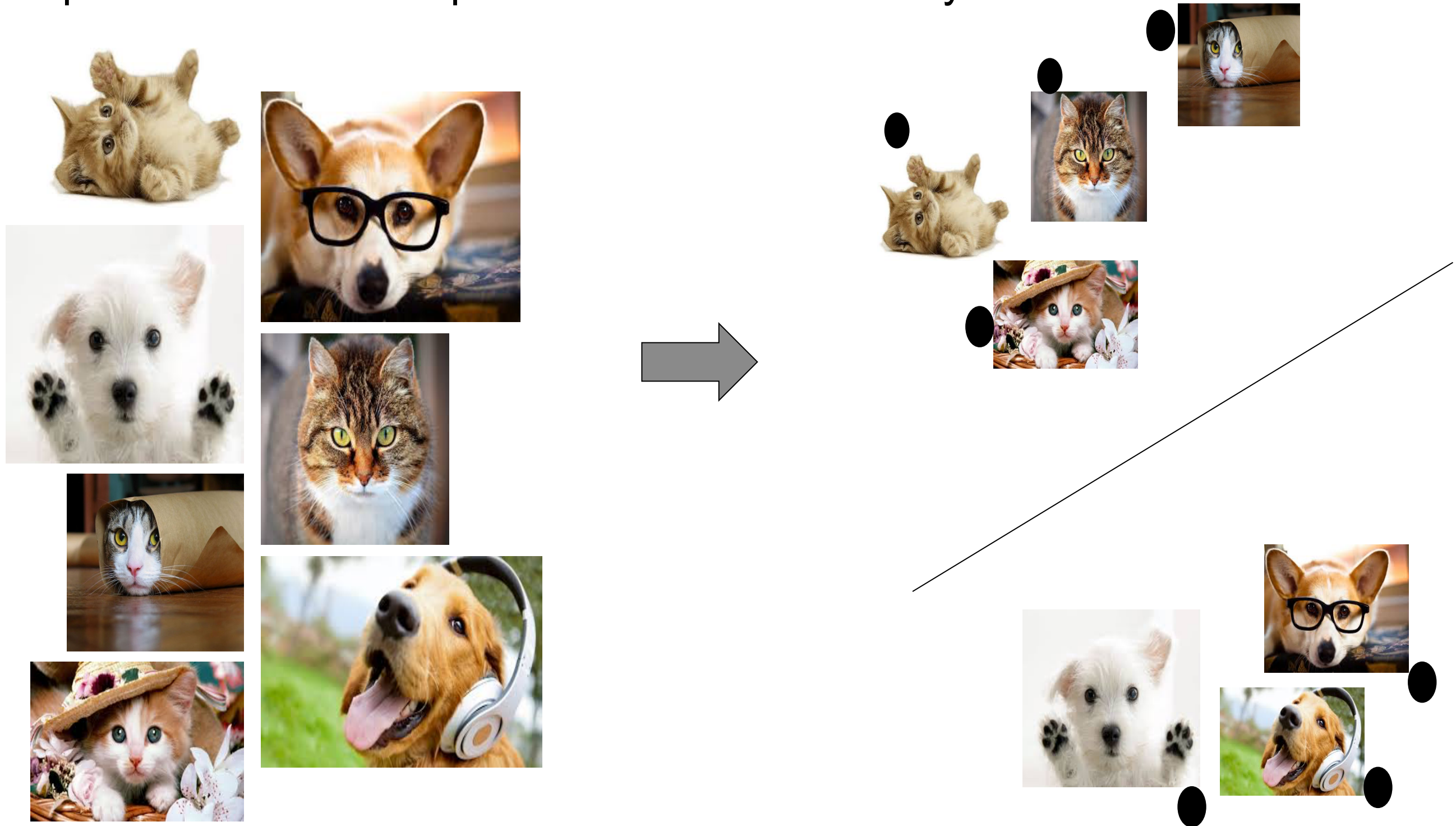


Probabilistic Programming with Parts

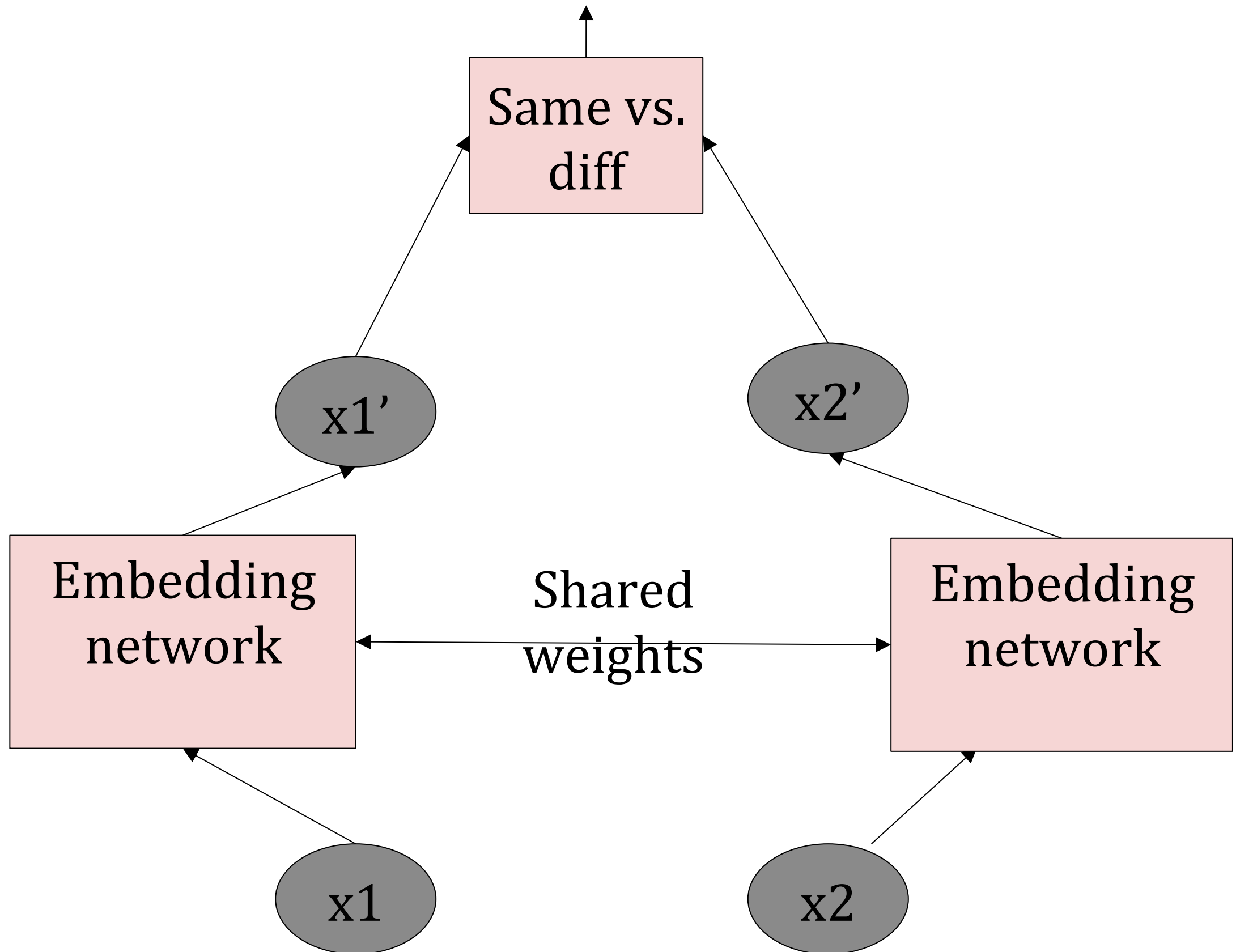


Metric learning









Learning how to represent objects (e.g. images) as points in some space where relative positions indicate similarity



Siamese Networks



Siamese Networks

		same	"cow" (speaker #1)	"cow" (speaker #2)	same
		different	"cow" (speaker #1)	"cat" (speaker #2)	different
		same	"can" (speaker #1)	"can" (speaker #2)	same
		different	"can" (speaker #1)	"cab" (speaker #2)	different

Verification tasks (training)



One-shot tasks (test)

[Koch, Zemel, Salak., 2015]

Siamese Networks

Method	Test
Humans	95.5
Hierarchical Bayesian Program Learning	95.2
Affine model	81.8
Hierarchical Deep	65.2
Deep Boltzmann Machine	62.0
Simple Stroke	35.2
1-Nearest Neighbor	21.7
Siamese Neural Net	58.3
Convolutional Siamese Net	92.0

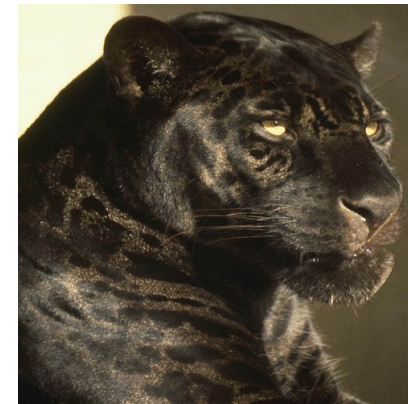
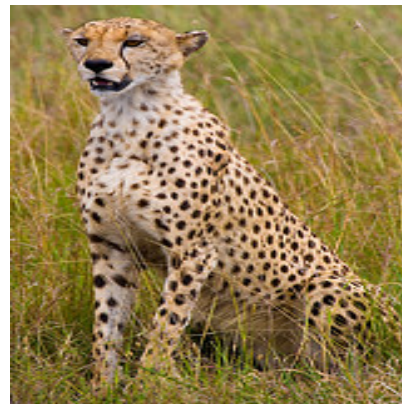
[Koch, Zemel, Salak., 2015]

Few shot learning, SOA

- Core question of one shot learning: can we **ask the model to predict classes** that are **shown rarely** in the training set?
- Current approaches:
 - Similarity: Koch et al. 2015 (Siamese net).
 - Embedding: Vinyals et al. 2016 (Train a classifier such that training resembles testing).
 - External memory: Santoro et al. 2016 (Use external memory to keep a mapping to new seen classes).
 - Meta-optimization: Ravi & Larochelle 2017 (Learn a parameterized optimizer through meta-optimization).

Small Data → Simple Model

- Retraining classifier on new classes would severely overfit
- In order to combat overfitting, choose **simple** model
- Nonparametric models scale with amount of data
 - **k-Nearest Neighbors on support set**
- Distances in raw input space may not help classification



- Learn transformation into a space where distances are informative for classification
- Neighborhood Components Analysis (Salakhutdinov & Hinton, 2007): classify via k-NN in latent space. Performs poorly in few-shot

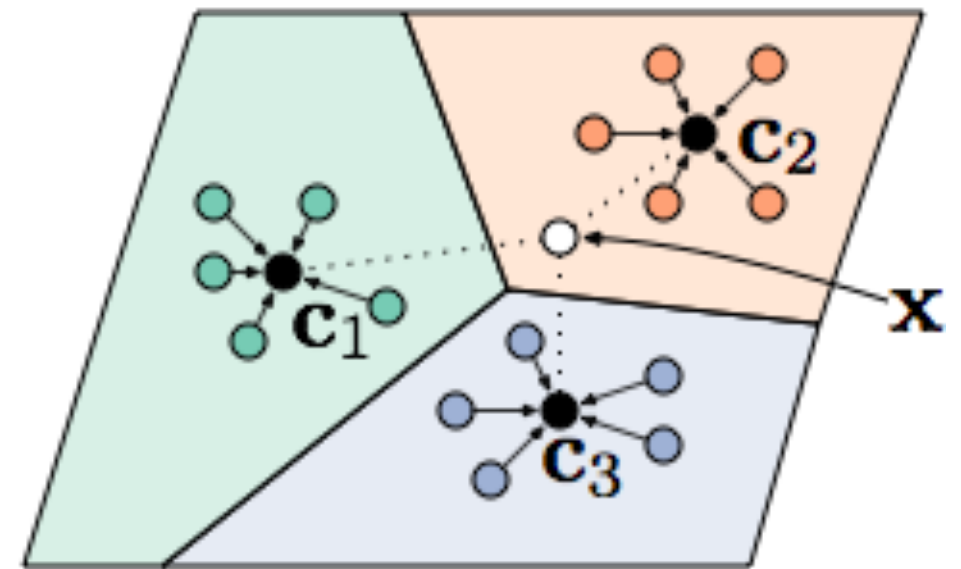
Prototypical Networks

- Assume points belong to a class cluster around a single prototype representation

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{\mathbf{x}_i \in S_k} f_\phi(\mathbf{x}_i)$$

- Classify unlabeled points by finding nearest class prototype

$$p_\phi(y = k | \mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))}$$



Jake Snell

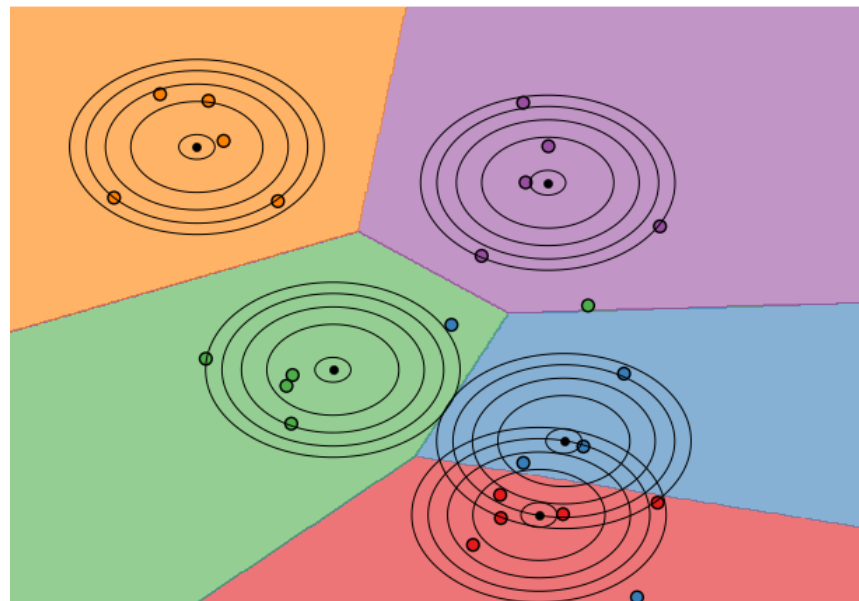


Kevin Swersky



Mixture Density Estimation View

- Fit mixture density model to support set and infer most likely cluster assignment for query points
- One cluster per class and take mean as cluster representative



- For Euclidean distance equivalent to isotropic Gaussian densities
- Other distances correspond to different class-conditional distributions

Euclidean Distance \Leftrightarrow Linear Model

$$p_{\phi}(y = k | \mathbf{x}) = \frac{\exp(-\|f_{\phi}(\mathbf{x}) - \mathbf{c}_k\|^2)}{\sum_{k'} \exp(-\|f_{\phi}(\mathbf{x}), \mathbf{c}_{k'}\|^2)}$$

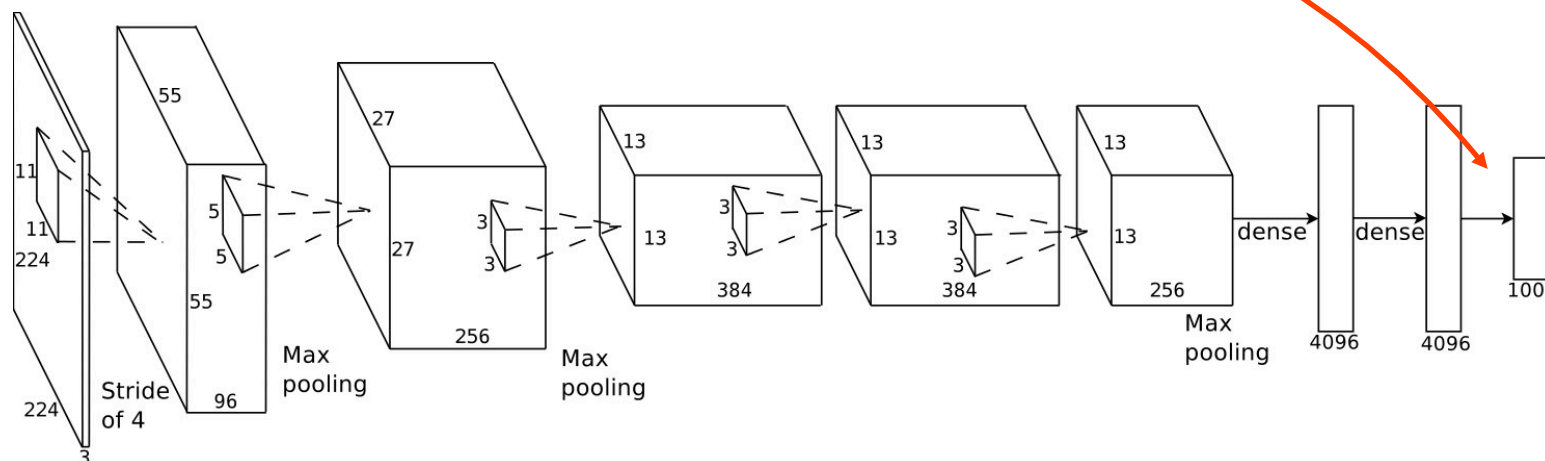
$$-\|f_{\phi}(\mathbf{x}) - \mathbf{c}_k\|^2 = -f_{\phi}(\mathbf{x})^{\top} f_{\phi}(\mathbf{x}) + 2\mathbf{c}_k^{\top} f_{\phi}(\mathbf{x}) - \mathbf{c}_k^{\top} \mathbf{c}_k$$

$$2\mathbf{c}_k^{\top} f_{\phi}(\mathbf{x}) - \mathbf{c}_k^{\top} \mathbf{c}_k = \mathbf{w}_k^{\top} f_{\phi}(\mathbf{x}) + b_k$$

$$p_{\phi}(y = k | \mathbf{x}) \propto \exp(\mathbf{w}_k^{\top} f_{\phi}(\mathbf{x}) + b_k)$$

$$\mathbf{w}_k = 2\mathbf{c}_k$$

$$b_k = -\mathbf{c}_k^{\top} \mathbf{c}_k$$



Softmax output layer
determined by class
prototypes!

Episodic Training

- Train using “episodes” to mimic n -shot, k -way test procedure
- For each training minibatch:
 - Subsample k training classes
 - Choose n examples as labeled support set
 - Take remaining examples as unlabeled query set
- Maximize probability of true class for query examples
- For better performance, make training batches more difficult by subsampling $> k$ classes

Experiments

- Datasets

- 1. Omniglot

- 28 x 28 grayscale images
 - 1623 characters from 50 alphabets, each with 20 examples
 - 1200 classes for training, 423 for testing (< Lake)

- 2. minilImagenet

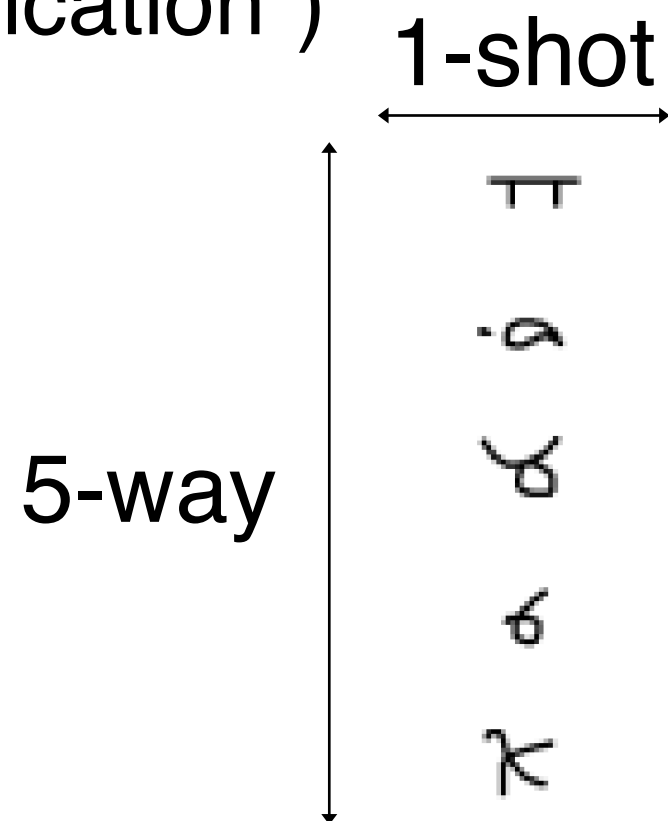
- 84 x 84 color images
 - 100 real-world classes, each with 600 examples
 - 64 used for training, 16 validation, 20 test

- Embedding architecture

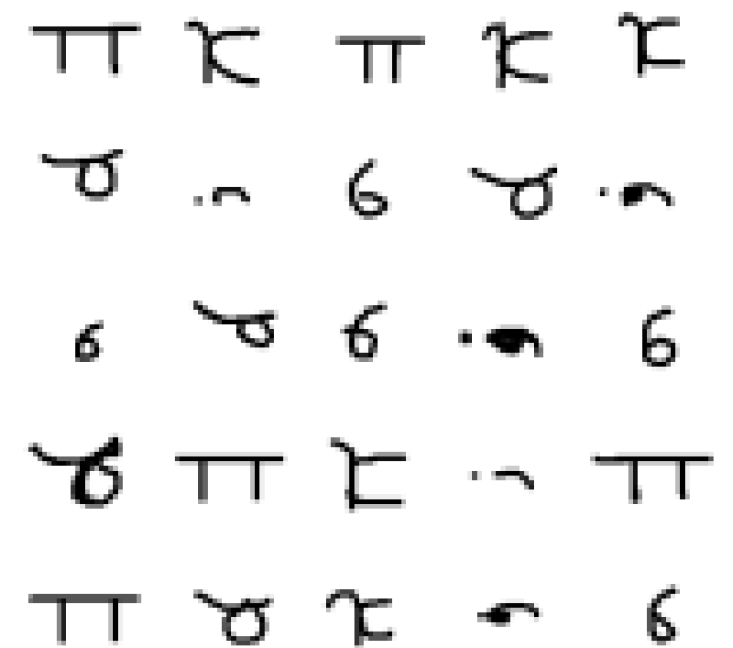
- Four convolutional layers, each with 64 filters
 - 64-dimensional embedding for Omniglot
 - 1600-dimensional embedding for minilImagenet

Few-Shot Setup: Omniglot

- Disjoint training classes and test classes
- At test time, shown “support set” with n examples each of k unseen classes (“ n -shot k -way classification”)



Support Set



Query Set

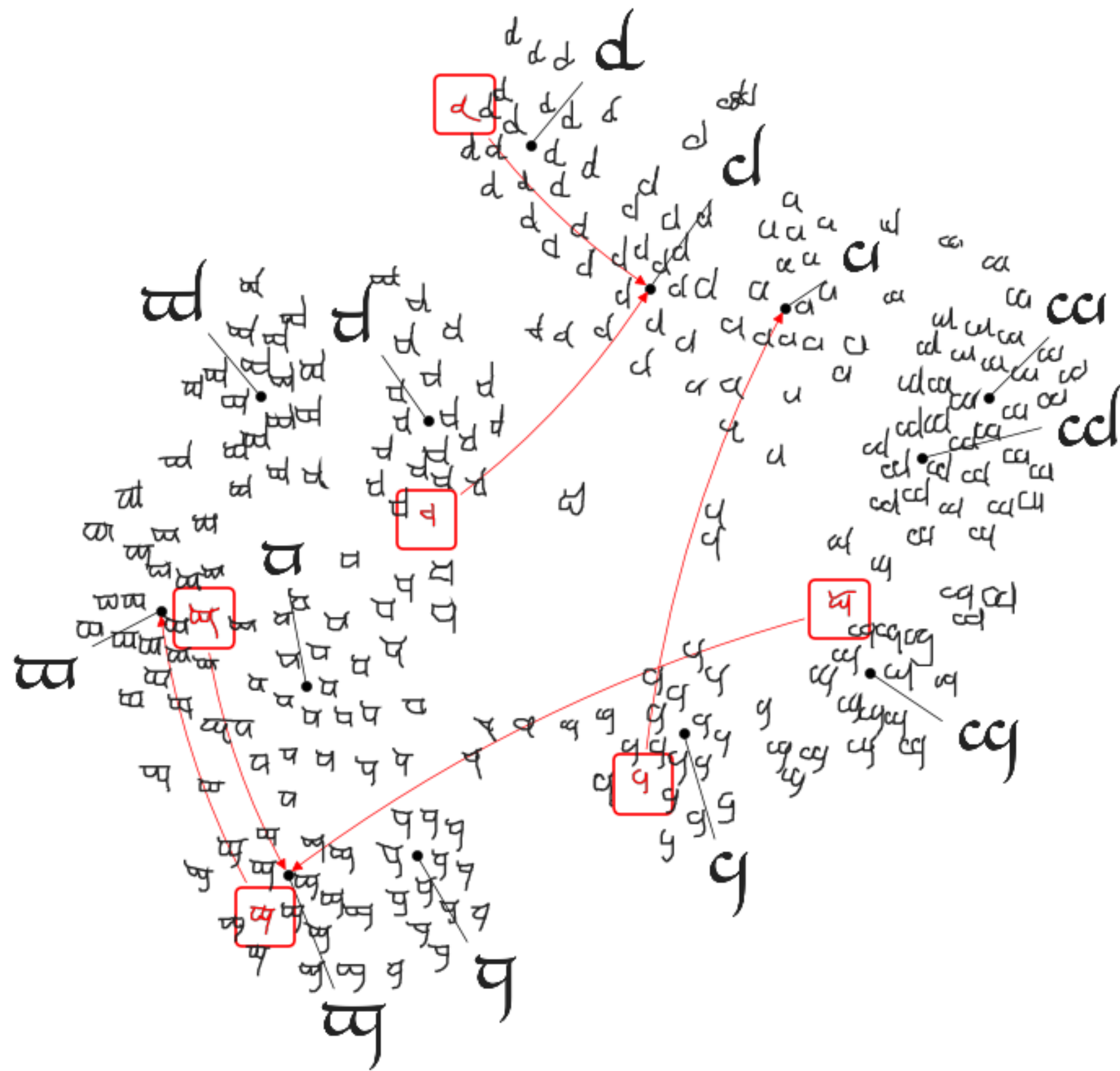
Few-Shot Results

Omniglot

Model	Dist.	Fine Tune	5-way Acc.		20-way Acc.	
			1-shot	5-shot	1-shot	5-shot
MATCHING NETWORKS (Vinyals et al., 2016)	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETWORKS (Vinyals et al., 2016)	Cosine	Y	97.9%	98.7%	93.5%	98.7%
NEURAL STATISTICIAN (Edwards & Storkey, 2017)	-	N	98.1%	99.5%	93.2%	98.1%
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	98.8%	99.7%	96.0%	98.9%

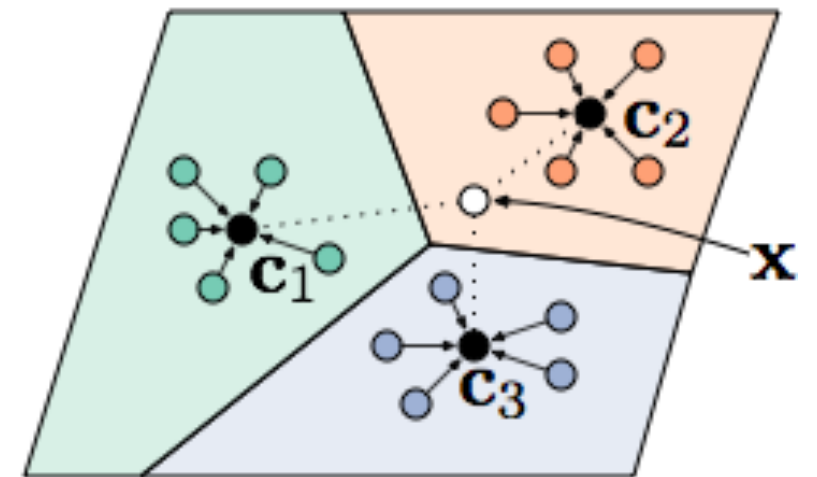
minImageNet

Model	Dist.	Fine Tune	5-way Acc.	
			1-shot	5-shot
BASILINE NEAREST NEIGHBORS*	Cosine	N	28.86 \pm 0.54%	49.79 \pm 0.79%
MATCHING NETWORKS (Vinyals et al., 2016)*	Cosine	N	43.40 \pm 0.78%	51.09 \pm 0.71%
MATCHING NETWORKS FCE (Vinyals et al., 2016)*	Cosine	N	43.56 \pm 0.84%	55.31 \pm 0.73%
META-LEARNER LSTM (Ravi & Larochelle, 2017)*	-	N	43.44 \pm 0.77%	60.60 \pm 0.71%
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	49.42 \pm 0.78%	68.20 \pm 0.66%

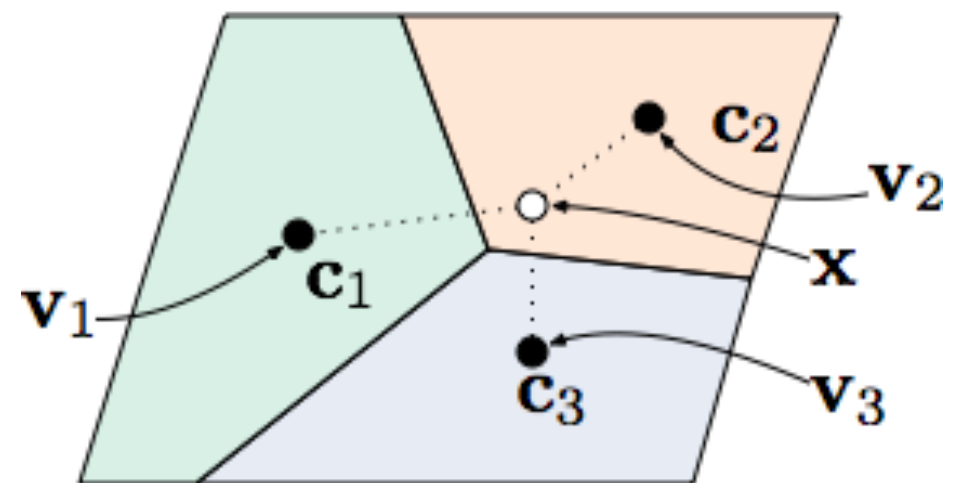


Zero-Shot Learning

- Instead of labeled examples at test, instead have class metadata
 - Text description: e.g. *“It is predominantly blue with a white chest and a blue crest.”*
 - Attribute vector
- Learn metadata embedding to produce prototype representation as function of attribute vector
- Both image and metadata embedding live in same space



Few-shot



Zero-shot

Zero-Shot Results

- CU-Birds dataset
 - Color images with GoogLeNet extracted features
 - 312-dimensional attribute vectors
 - 150 species train, 50 species test
- Linear embedding into shared 1024-dimensional space



Model	Image Features	50-way Acc. 0-shot
ALE [1]	Fisher	26.9%
SJE [2]	AlexNet	40.3%
SAMPLE CLUSTERING [17]	AlexNet	44.3%
SJE [2]	GoogLeNet	50.1%
DS-SJE [22]	GoogLeNet	50.4%
DA-SJE [22]	GoogLeNet	50.9%
PROTO. NETS (OURS)	GoogLeNet	54.6%

Conclusion

- Many other interesting topics in modern CNNs:
 - Fast processing
 - Reduced precision
 - Compressing weights
 - Novel optimization techniques
- Lots of other applications
 - Object detection, tracking, optic flow, 3D vision, egomotion estimation
 - Non-vision ones: text and speech processing, learning to rank

Conclusion

- Still not well understood
 - Learned representations
 - Mini-batch size, design
 - Links to parts-based models; causal models
 - Hyperparameter optimization, particularly wrt architecture
 - Role for non-parametric models, probabilistic models?