ECE 6504: Deep Learning for Perception

Topics:

- (Finish) Backprop
- Convolutional Neural Nets

Dhruv Batra

Virginia Tech

Administrativia

- Presentation Assignments
  - https://docs.google.com/spreadsheets/d/1m76E4mC0wfRjc4HRBWFdAlXKPIzlEwfw1-u7rBw9TJ8/edit#gid=2045905312

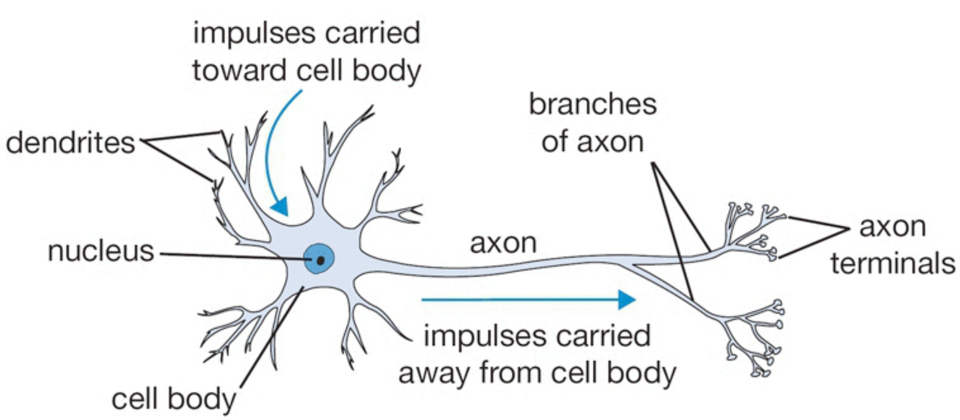
Recap of last time

Last Time

- Notation + Setup
- Neural Networks
- Chain Rule + Backprop

Recall: The Neuron Metaphor

- Neurons
  - accept information from multiple inputs,
  - transmit information to other neurons.
- Artificial neuron
  - Multiply inputs by weights along edges
  - Apply some function to the set of inputs at each node



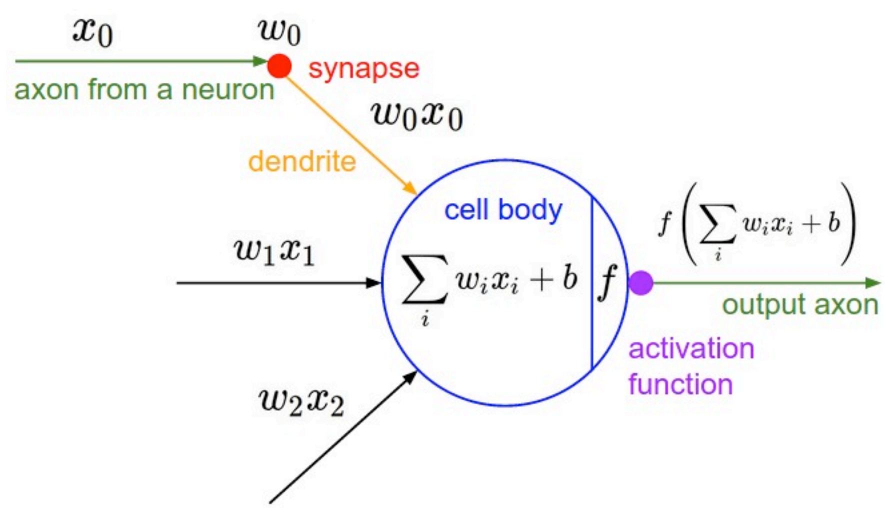
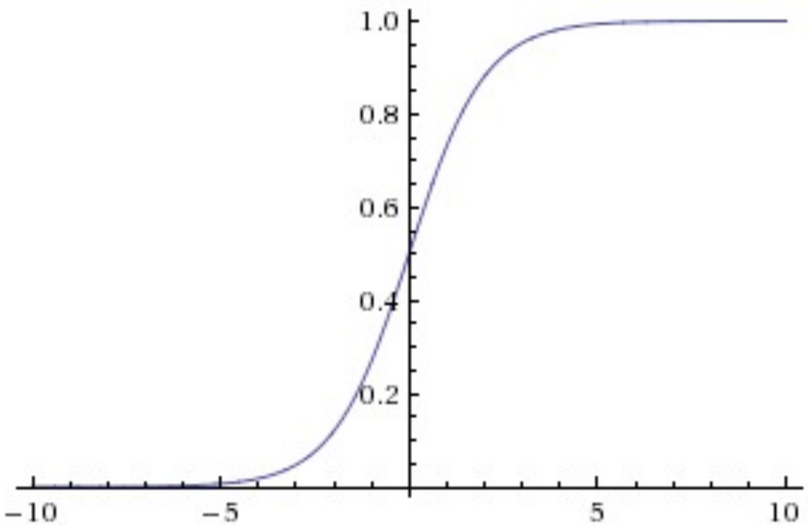
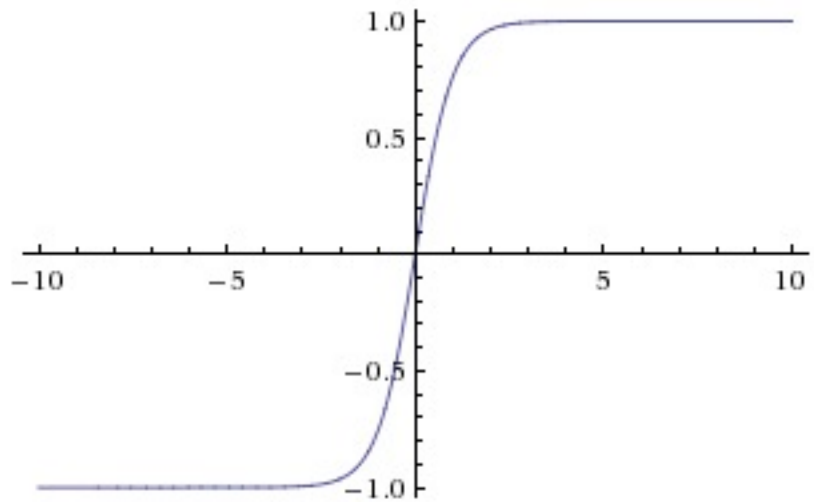


Image Credit: Andrej Karpathy, CS231n

Activation Functions

- sigmoid vs tanh





A quick note

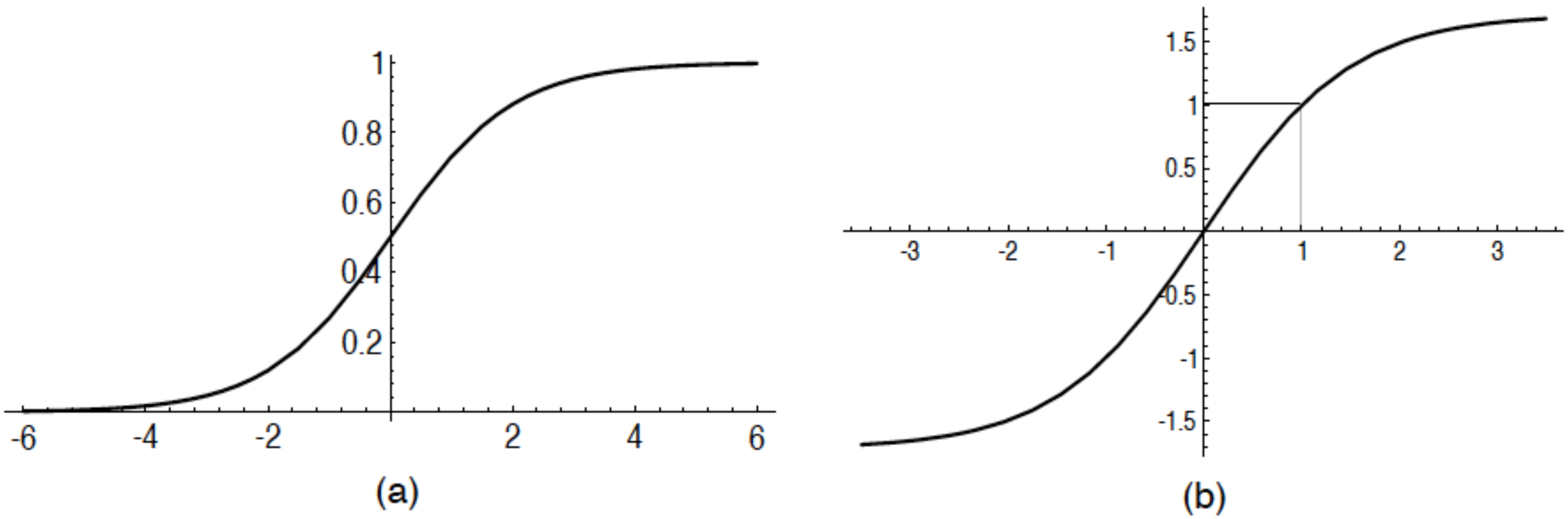
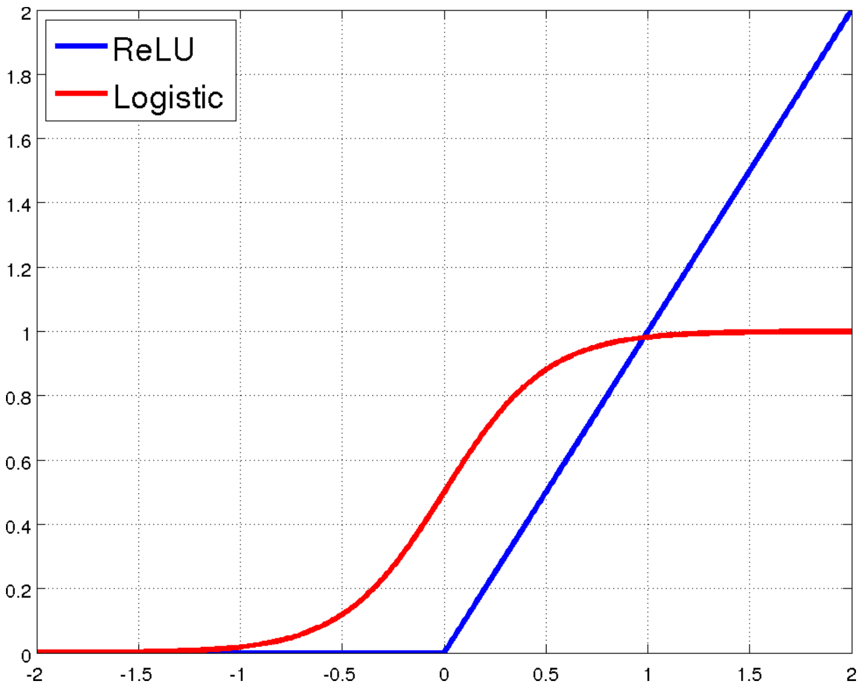
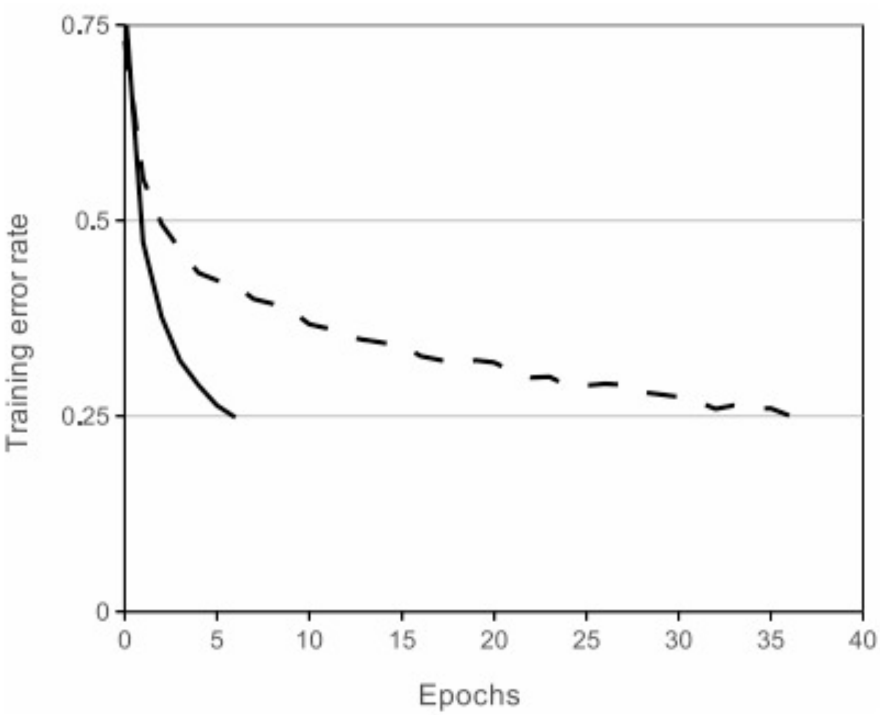


Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1 + e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right)$.

Image Credit: LeCun et al. '98

Rectified Linear Units (ReLU)





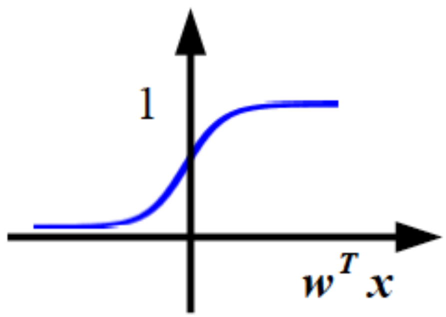
Linear Classifier: Logistic Regression

Input: $\boldsymbol{x} \in R^D$

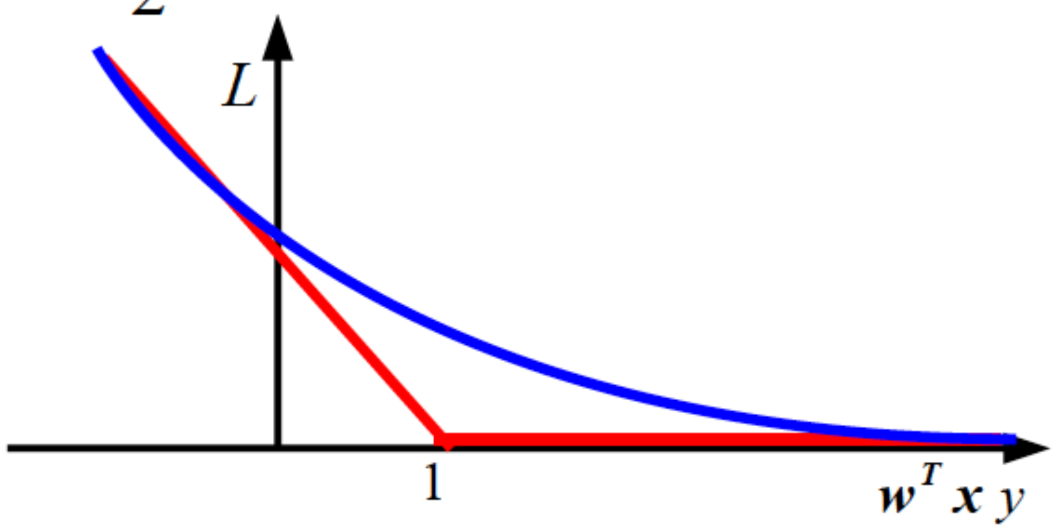
Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $p(y=1|\boldsymbol{x}) = \dfrac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}}}$



Loss: $L = \dfrac{1}{2}\|\boldsymbol{w}\|^2 - \lambda \log(p(y|\boldsymbol{x}))$



Log Loss

Linear Classifier: SVM

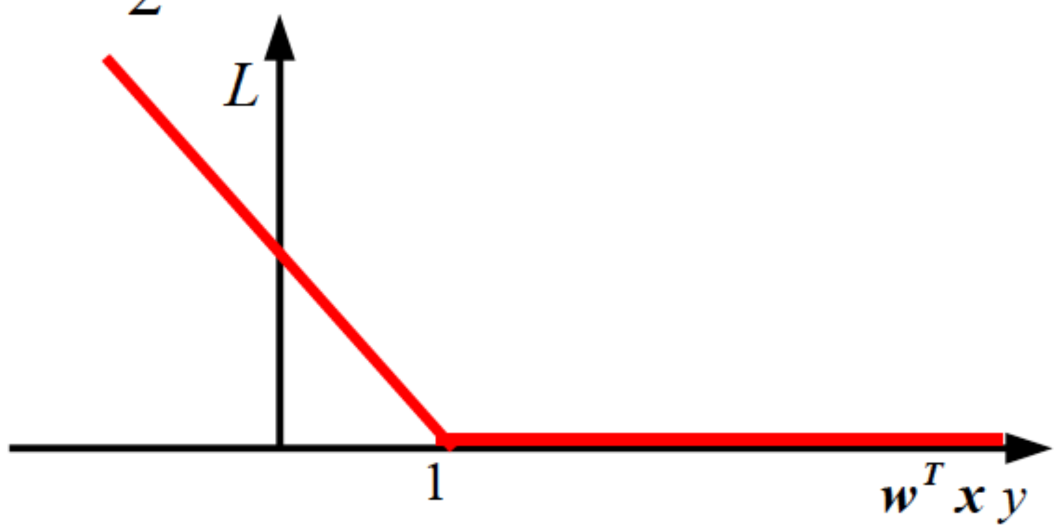
Input: $\boldsymbol{x} \in R^D$

Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $\boldsymbol{w}^T \boldsymbol{x}$

Loss: $L = \frac{1}{2} ||\boldsymbol{w}||^2 + \lambda \, max\,[0, 1 - \boldsymbol{w}^T \boldsymbol{x} \, y]$



Hinge Loss

Visualizing Loss Functions

- Sum of individual losses

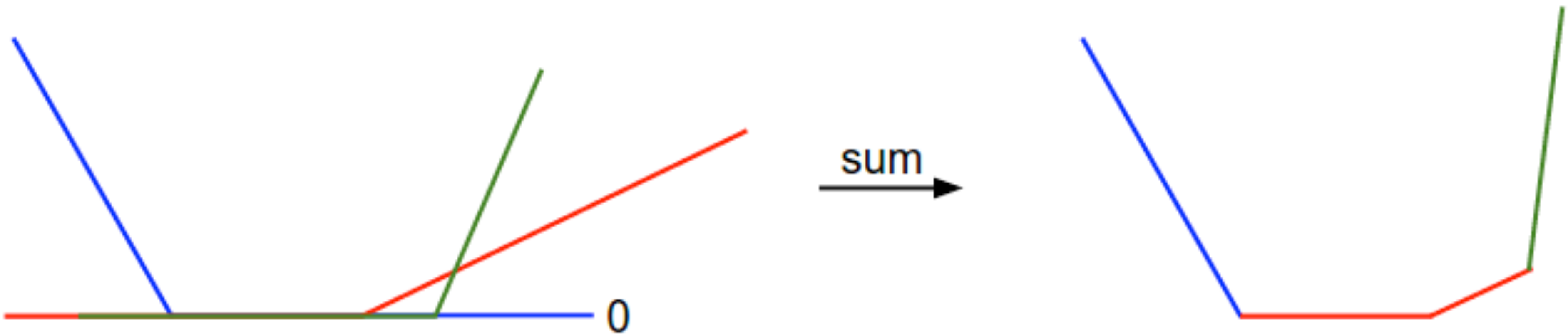
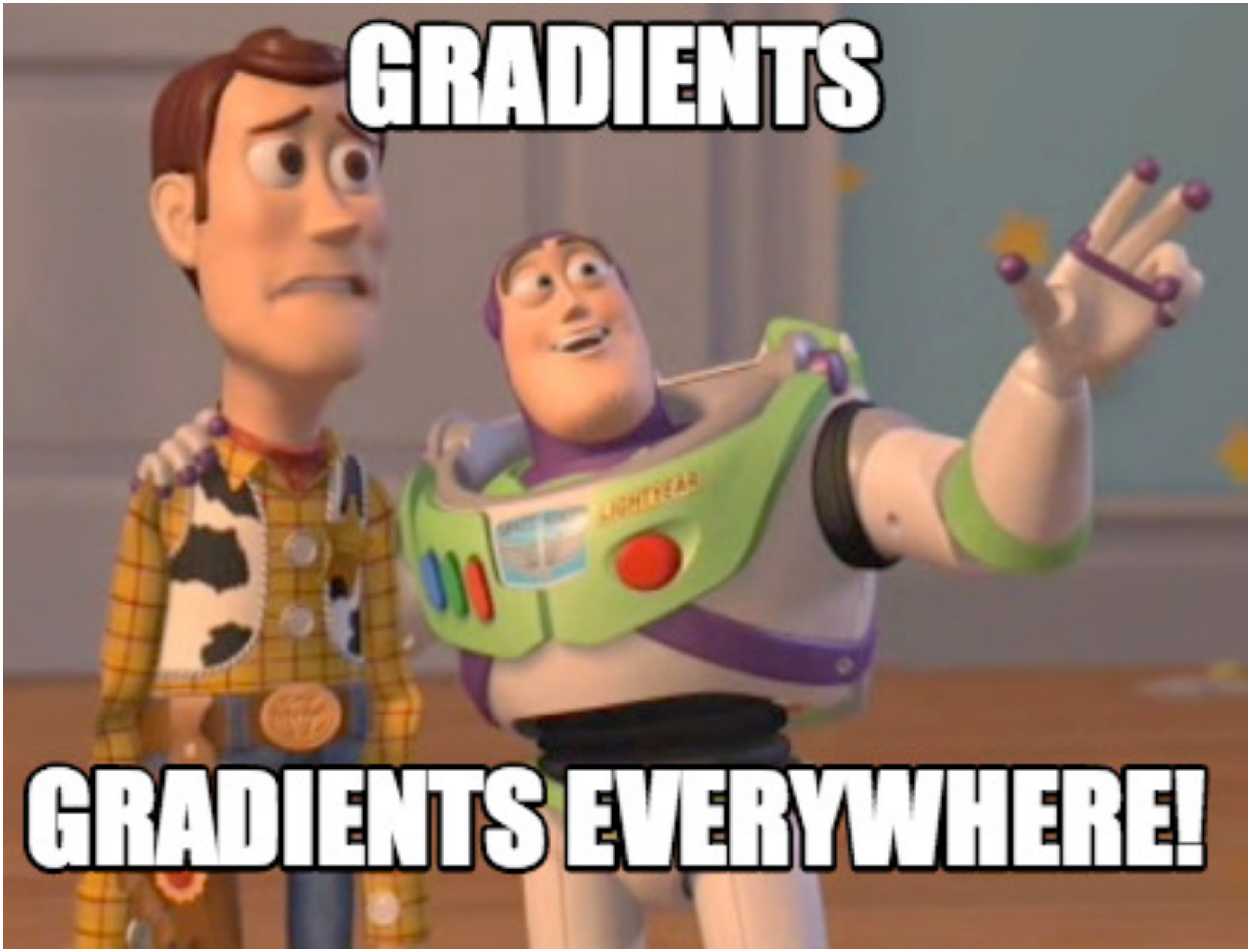
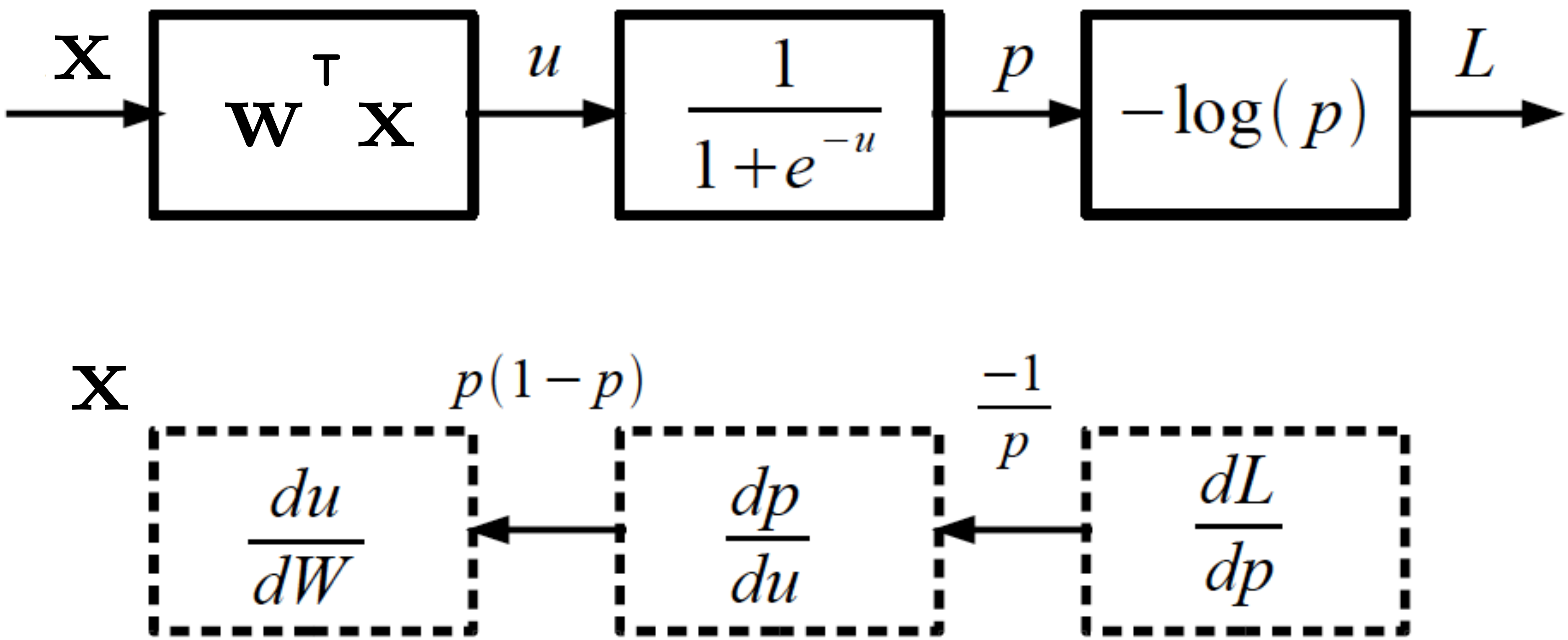


Image Credit: Andrej Karpathy, CS231n

Detour



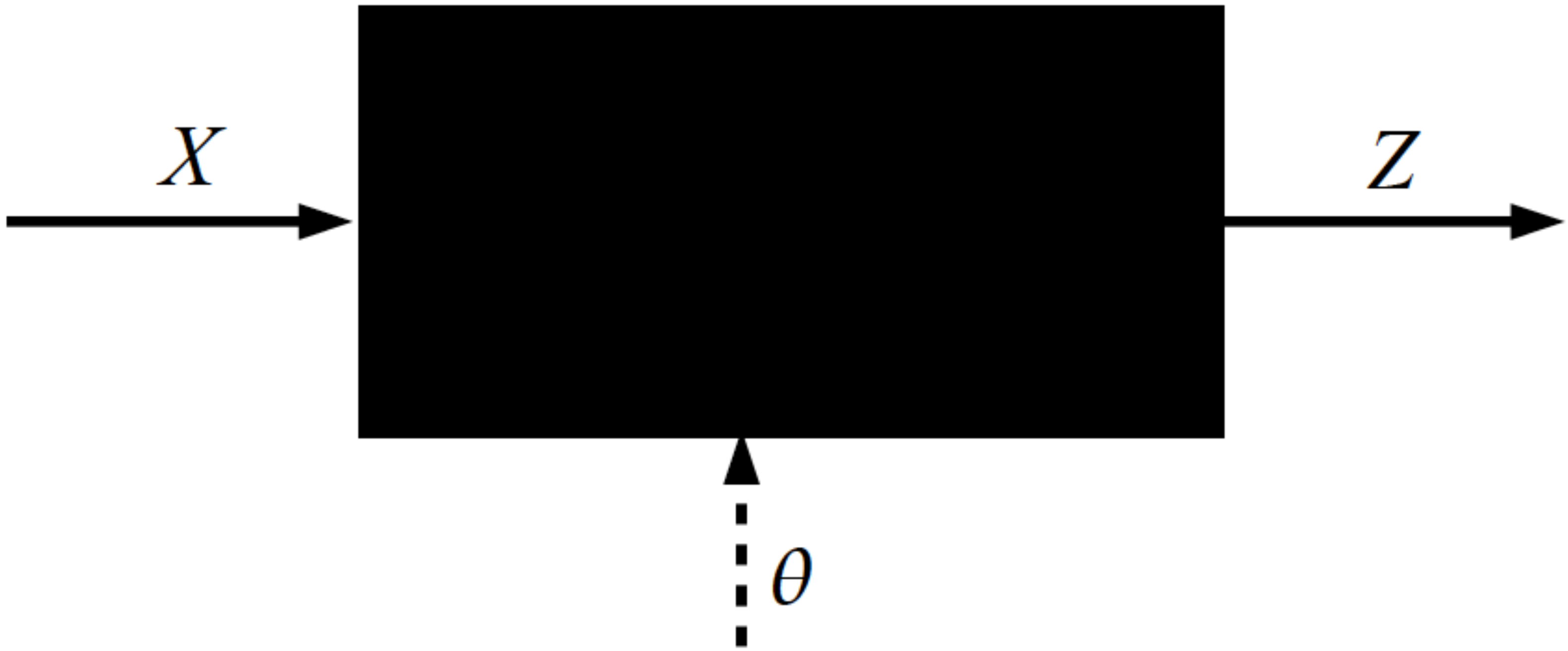
Logistic Regression as a Cascade



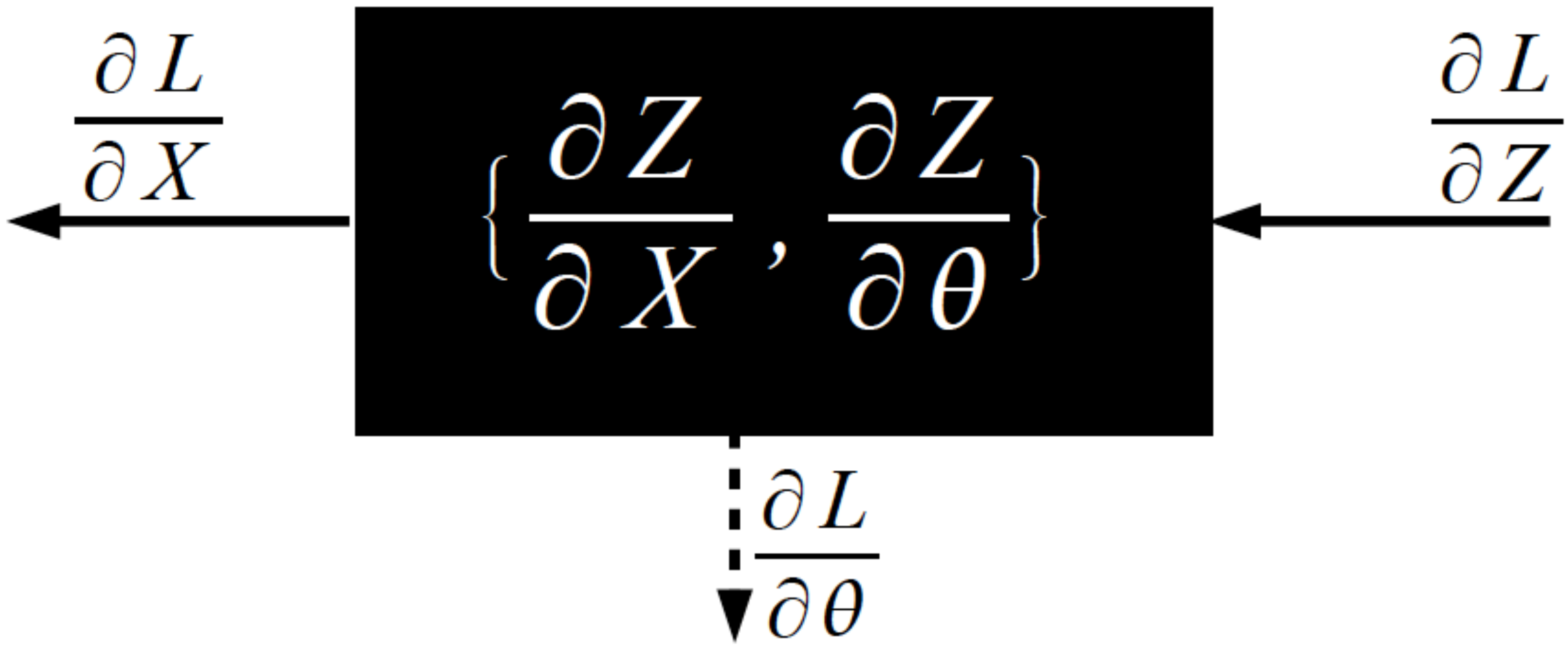
$$\frac{dL}{dW} = \frac{dL}{dp} \cdot \frac{dp}{du} \cdot \frac{du}{dW} = (p-1)\mathbf{x}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Key Computation: Forward-Prop



Key Computation: Back-Prop



Plan for Today

- MLPs
  - Notation
  - Backprop

- CNNs
  - Notation
  - Convolutions
  - Forward pass
  - Backward pass

Multilayer Networks

- Cascade Neurons together
- The output from one layer is the input to the next
- Each Layer has its own sets of weights

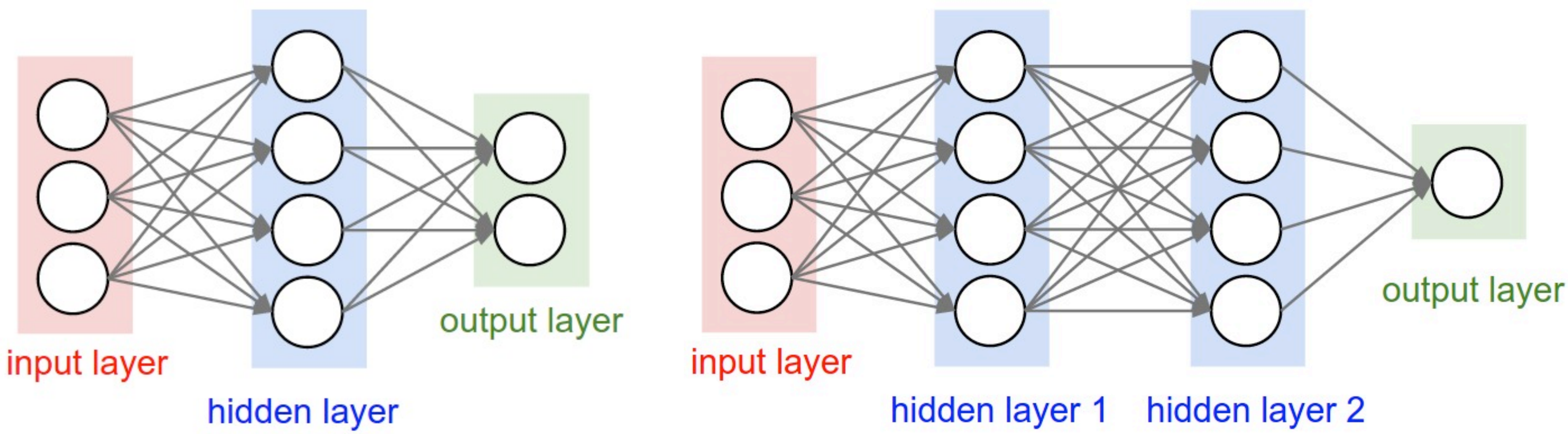
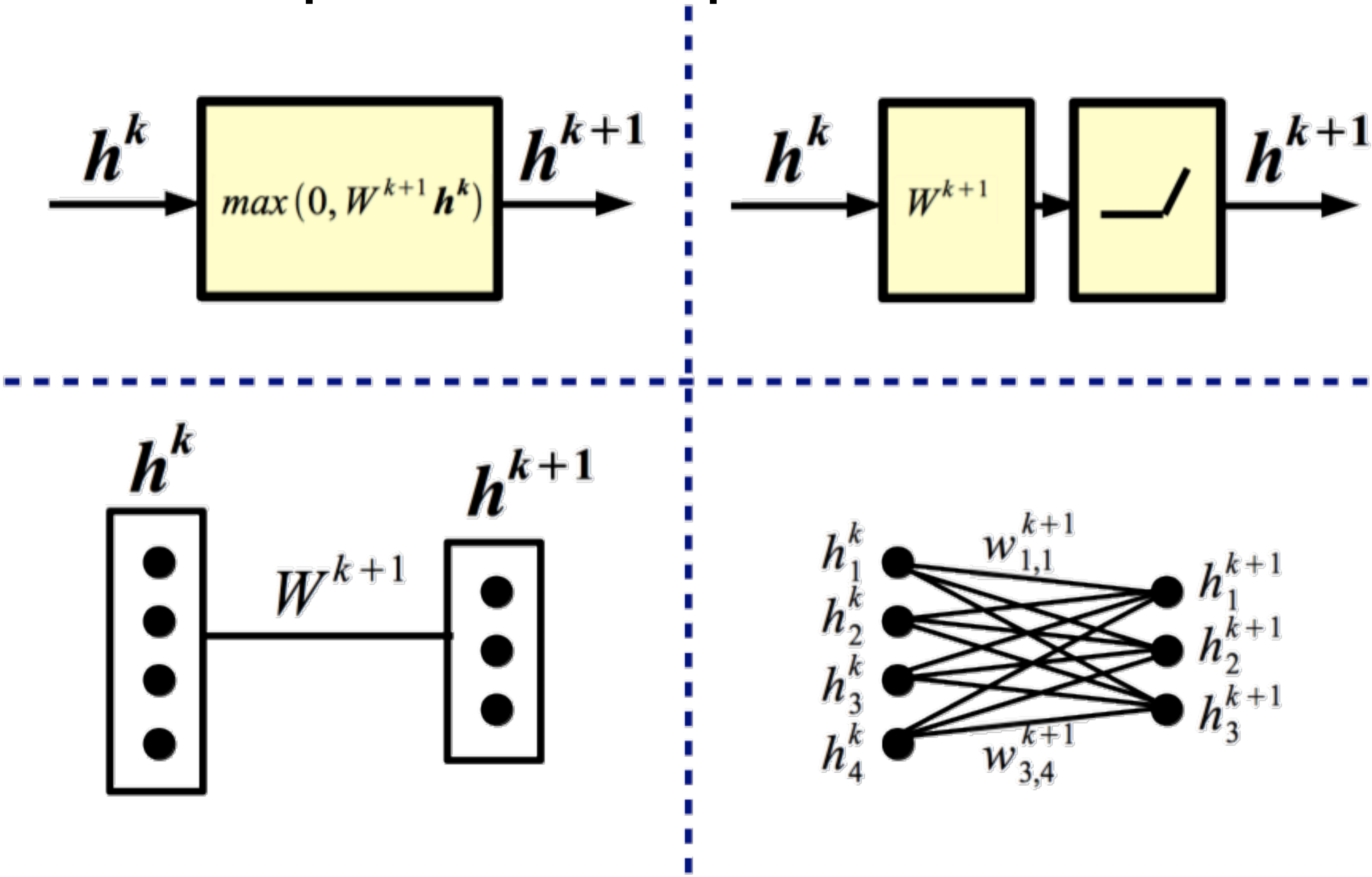


Image Credit: Andrej Karpathy, CS231n

Equivalent Representations



Backward Propagation

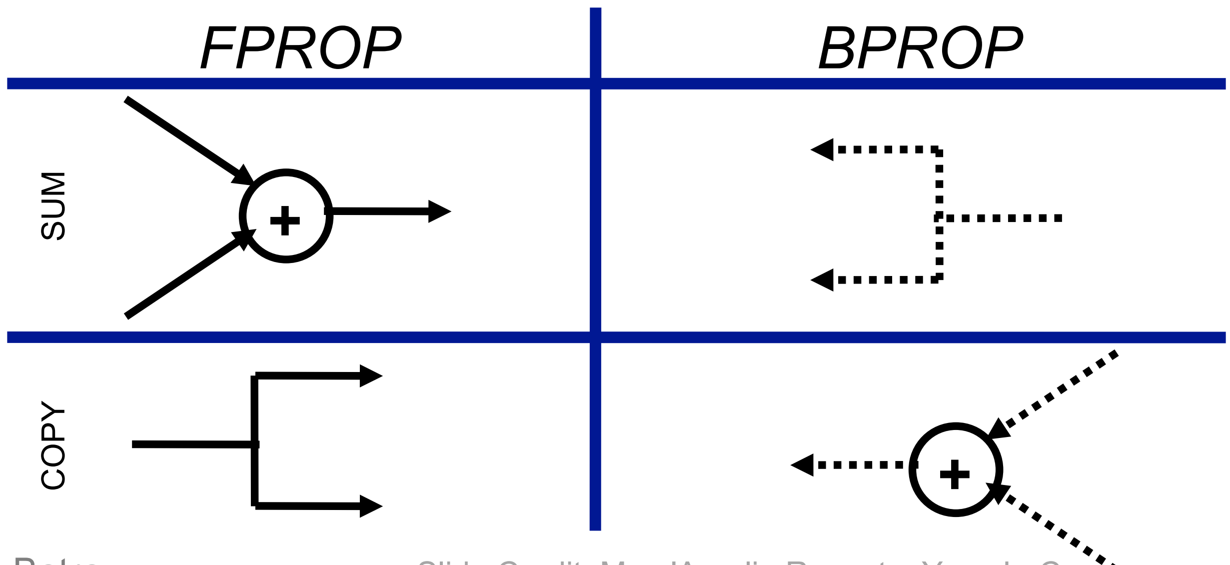
Question: Does BPROP work with ReLU layers only?

Answer: Nope, any a.e. differentiable transformation works.

Question: What's the computational cost of BPROP?

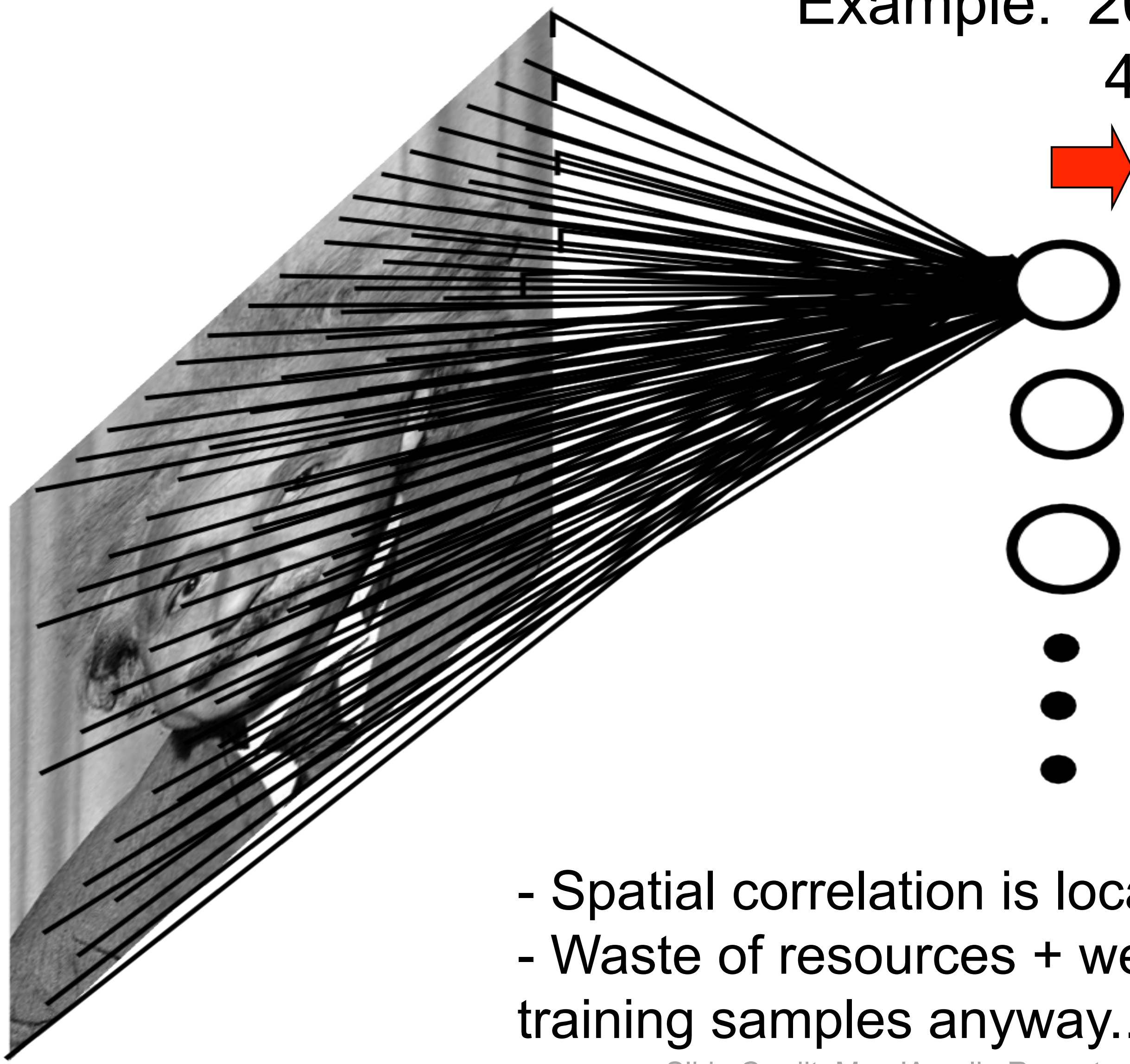
Answer: About twice FPROP (need to compute gradients w.r.t. input and parameters at every layer).

Note: FPROP and BPROP are dual of each other. E.g.,:



Fully Connected Layer

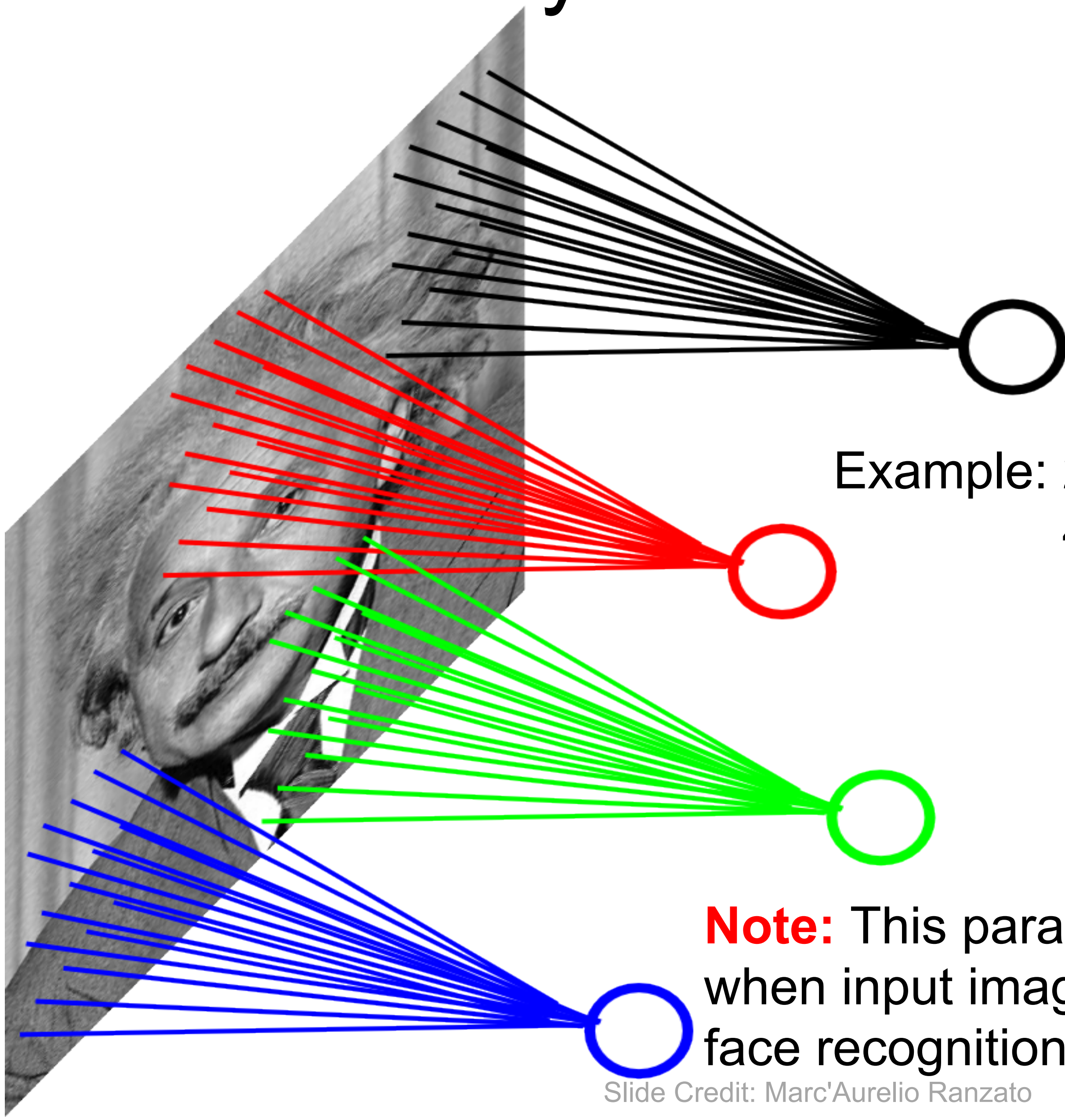
Example: 200x200 image
40K hidden units
~2B parameters!!!



- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer

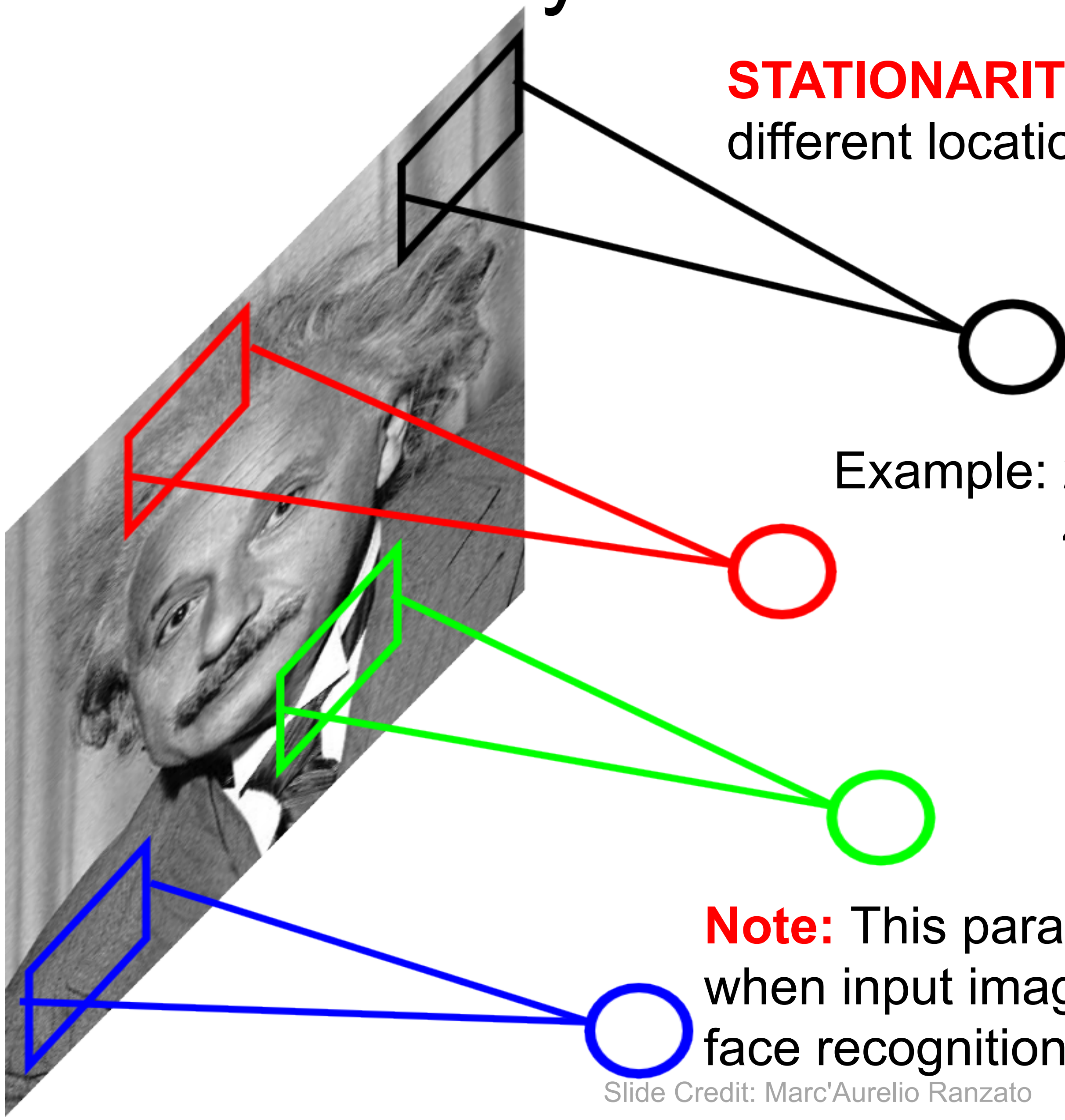


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

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

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer



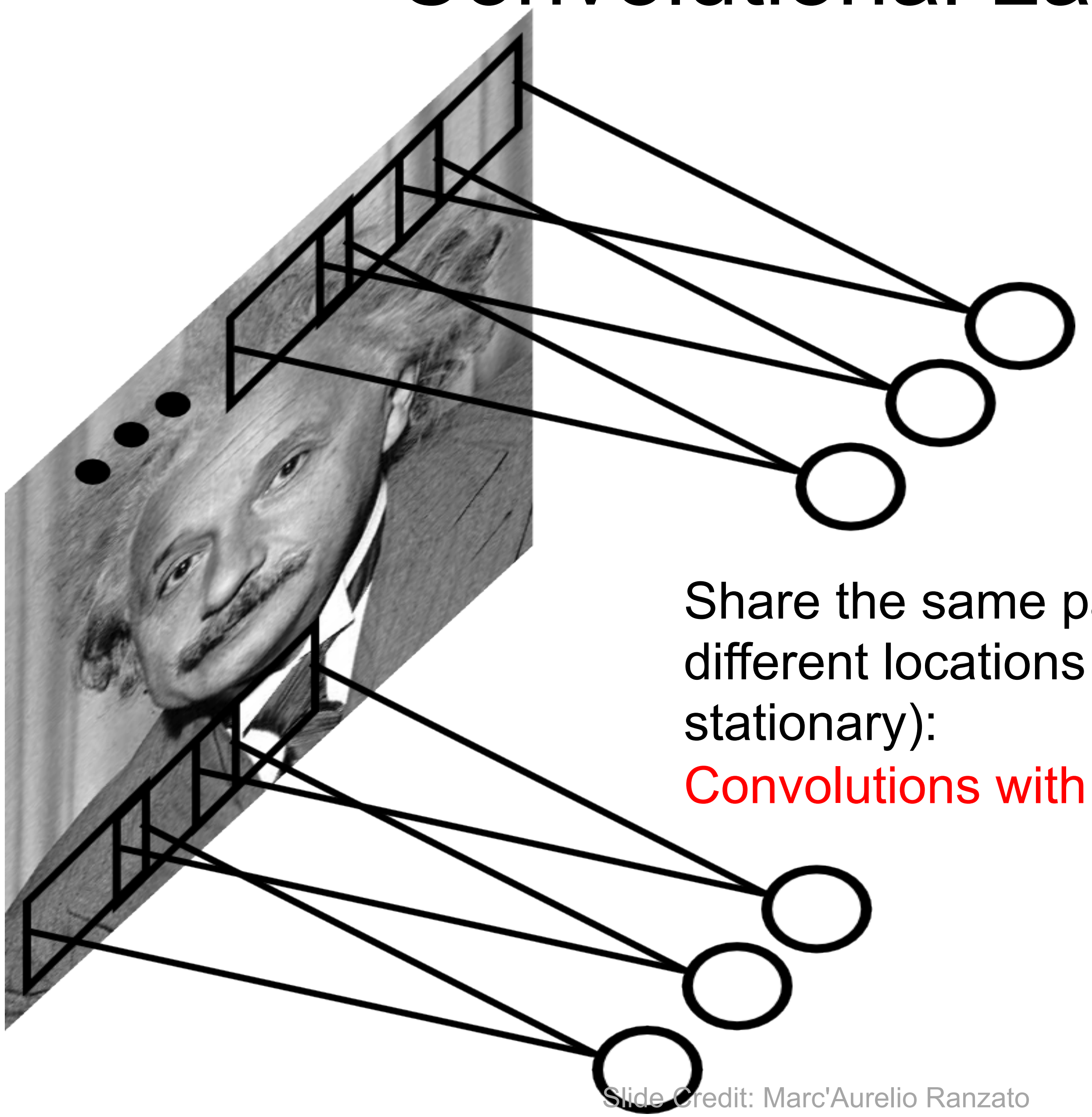
STATIONARITY? Statistics is similar at different locations

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

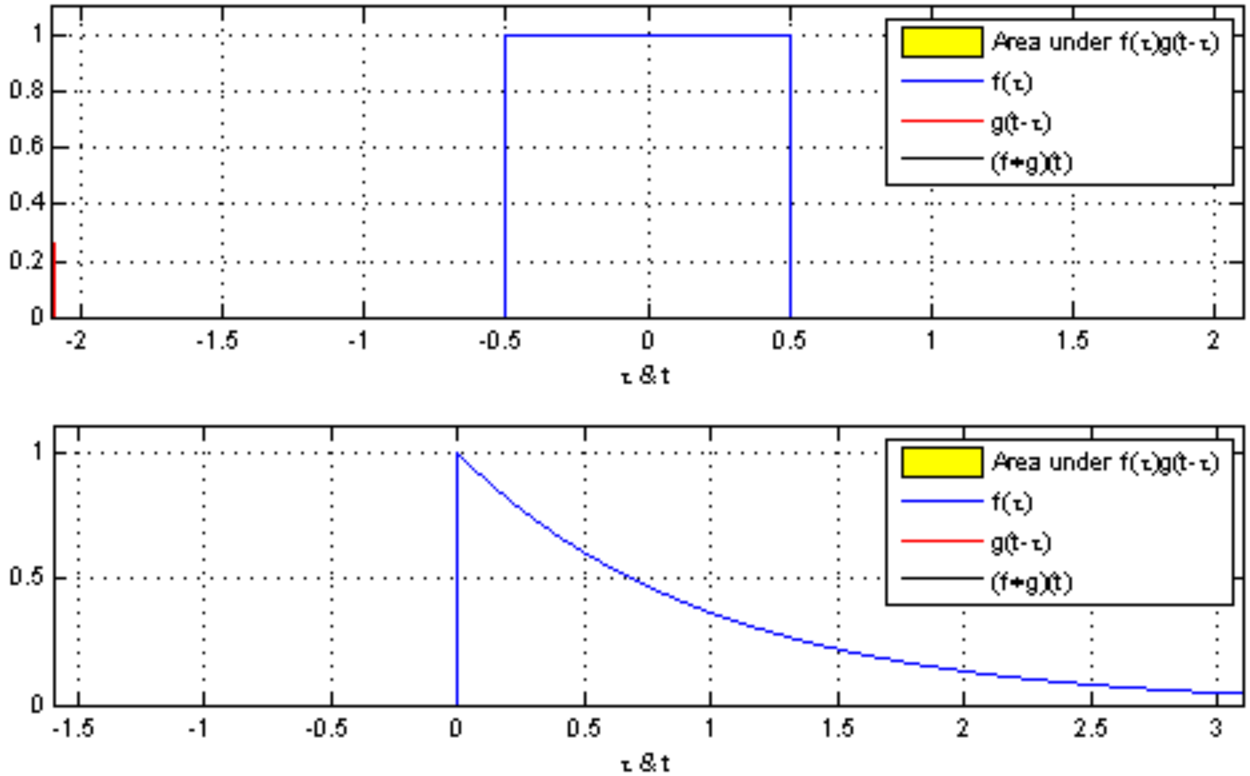
Note: This parameterization is good when input image is registered (e.g., face recognition).

Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

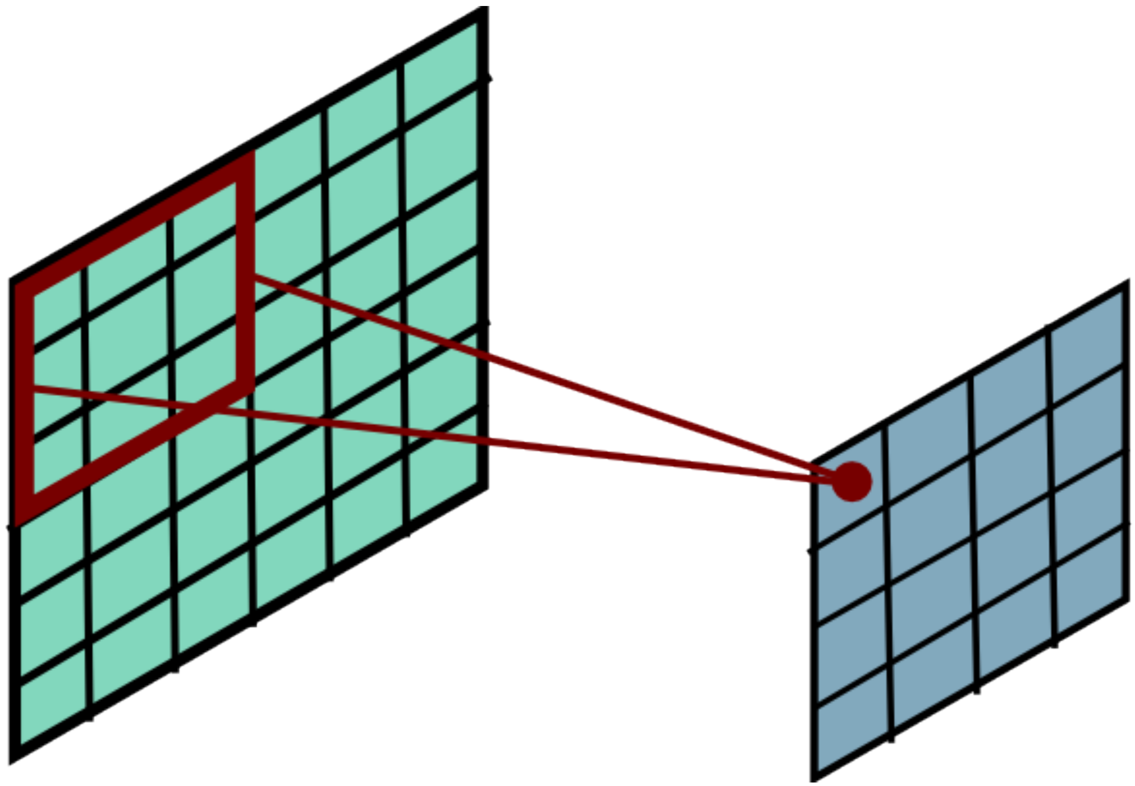


"Convolution of box signal with itself2" by Convolution_of_box_signal_with_itself.gif: Brian Ambergderivative work: Tinos (talk) - Convolution_of_box_signal_with_itself.gif. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Convolution_of_box_signal_with_itself2.gif#/media/File:Convolution_of_box_signal_with_itself2.gif

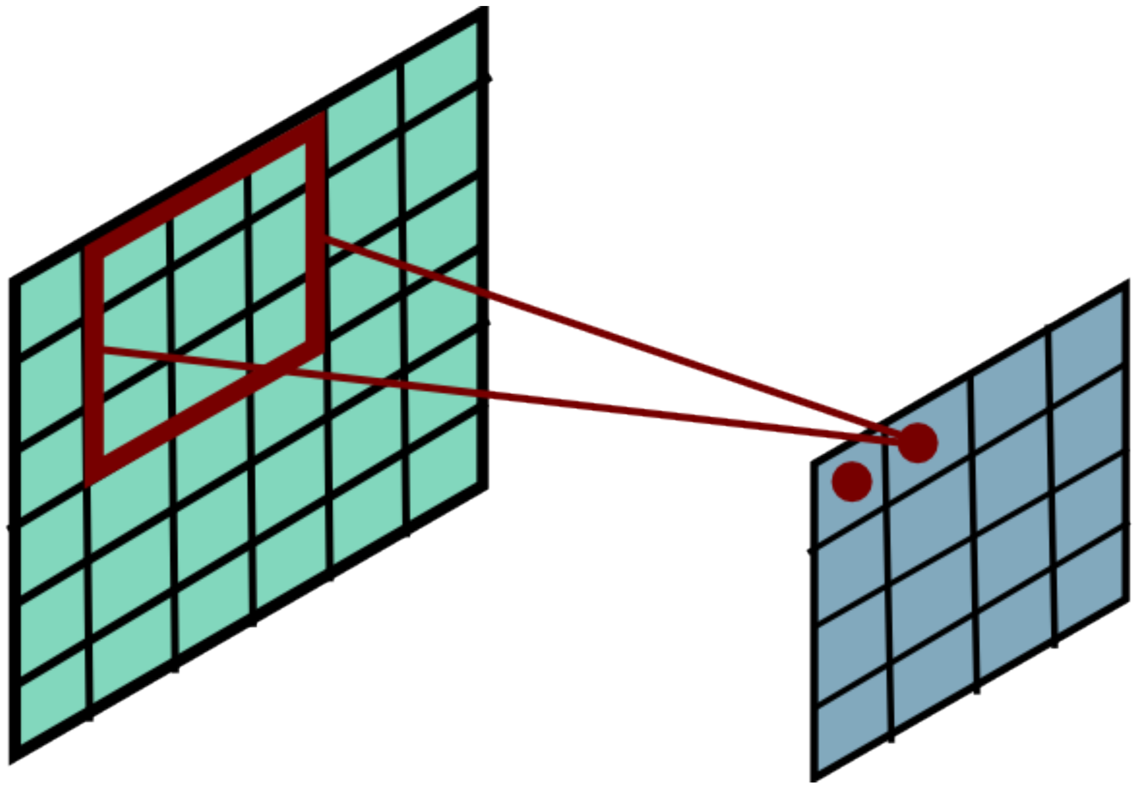
Convolution Explained

- http://setosa.io/ev/image-kernels/
- https://github.com/bruckner/deepViz

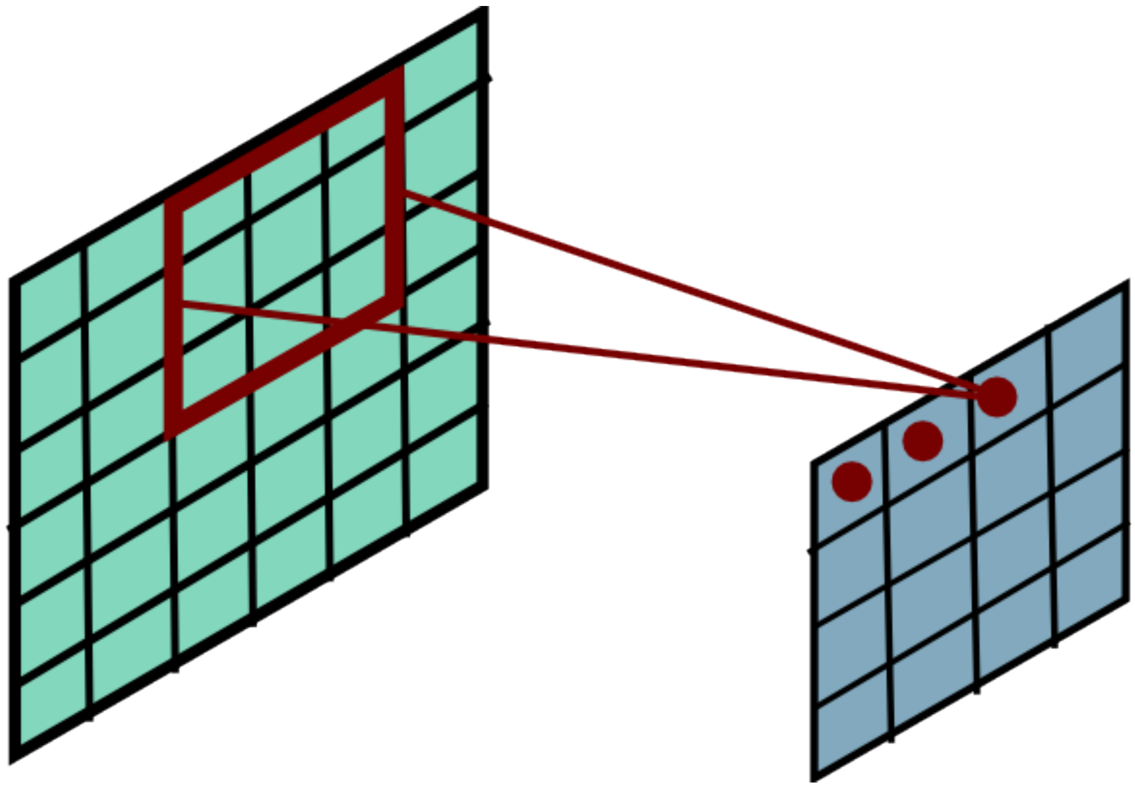
Convolutional Layer



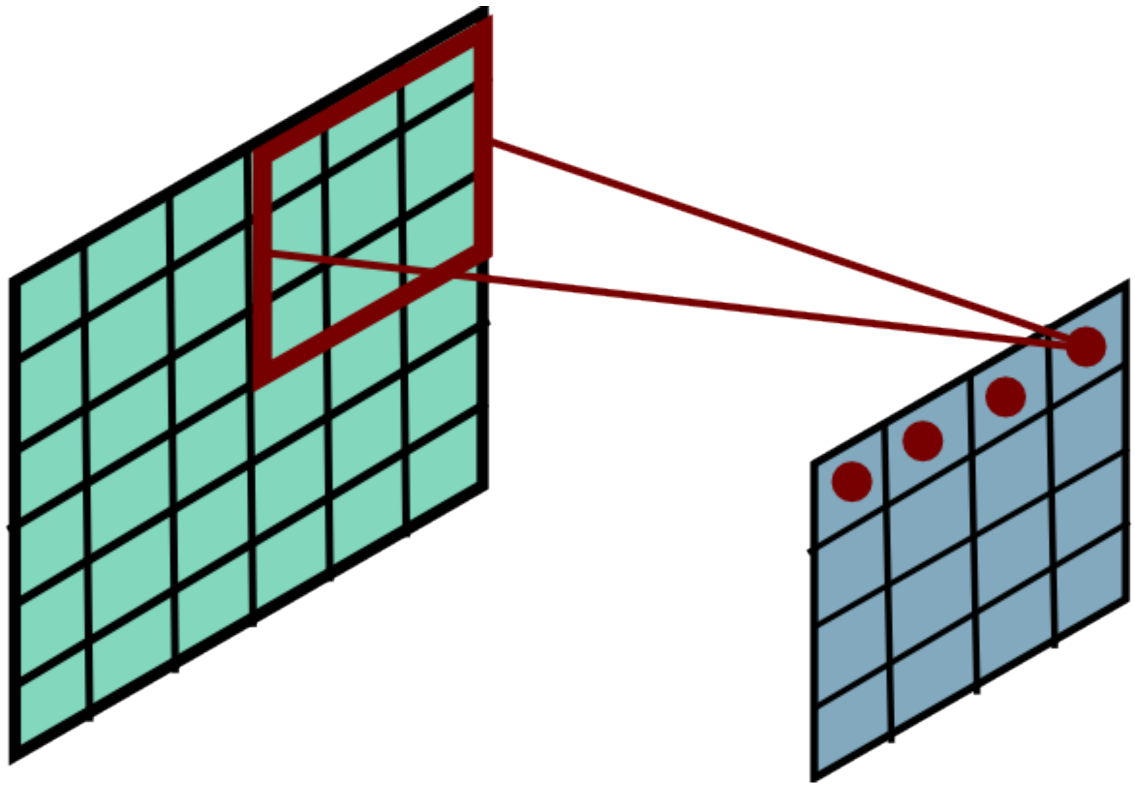
Convolutional Layer



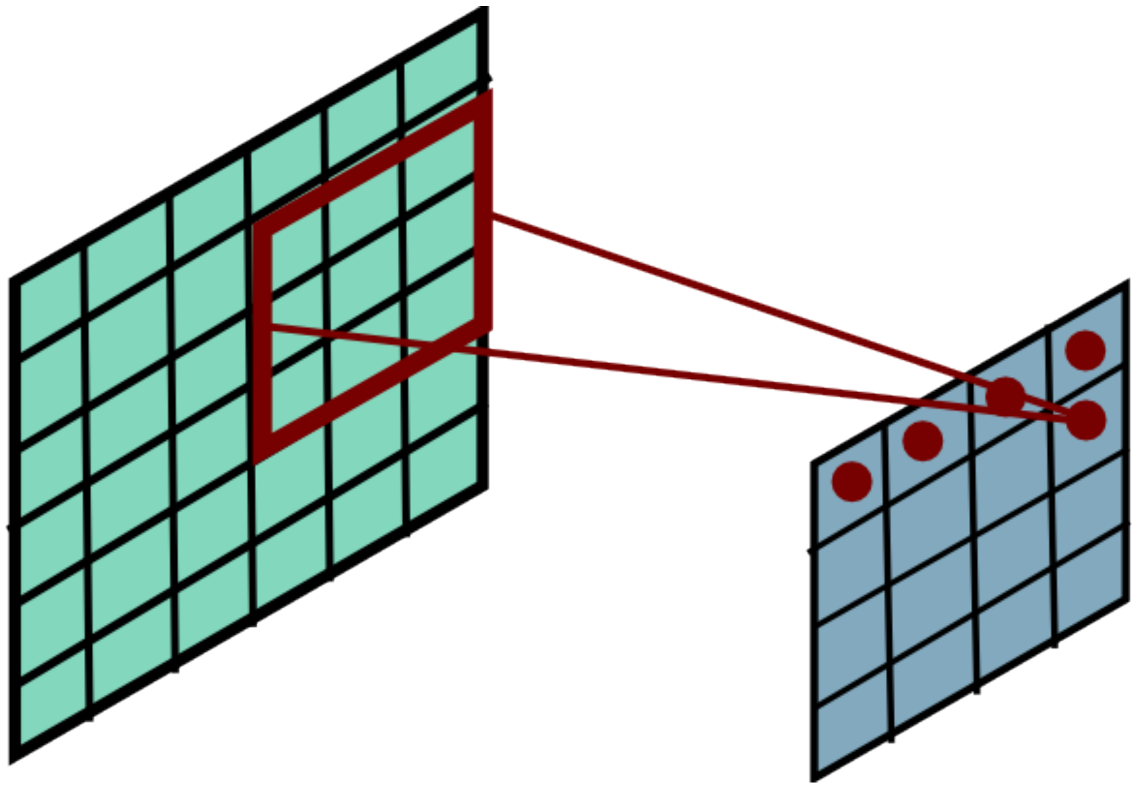
Convolutional Layer



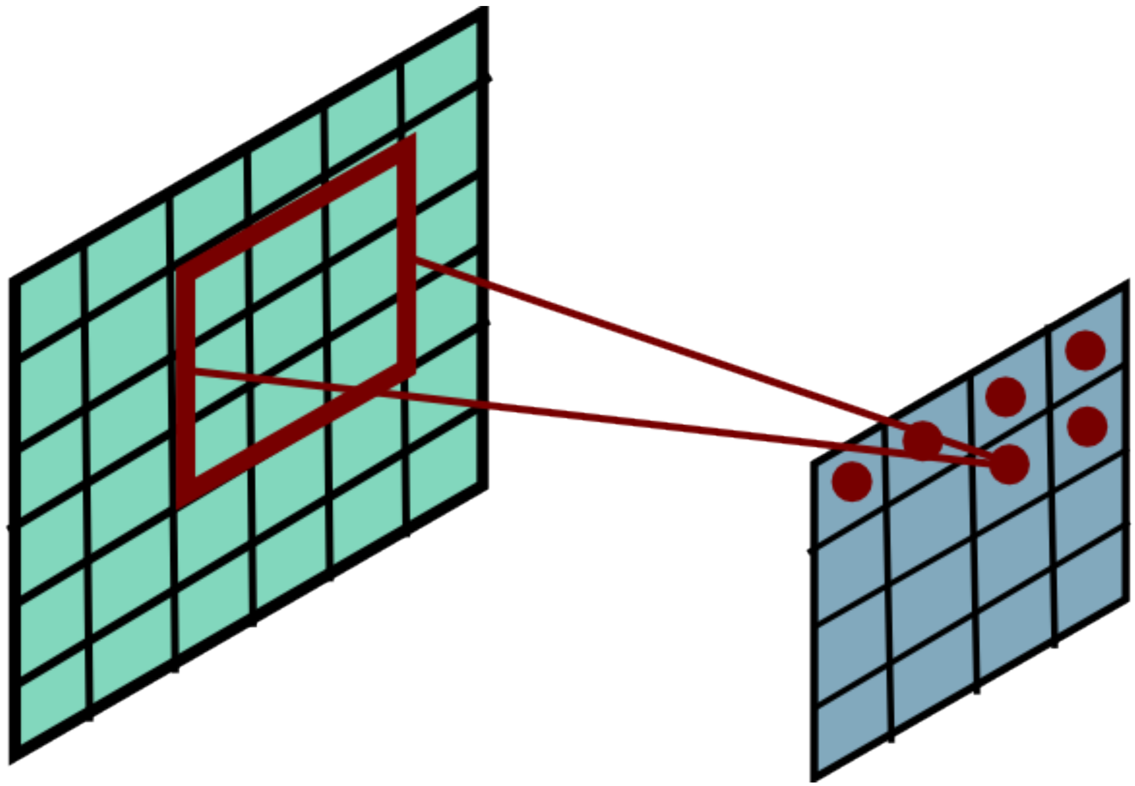
Convolutional Layer



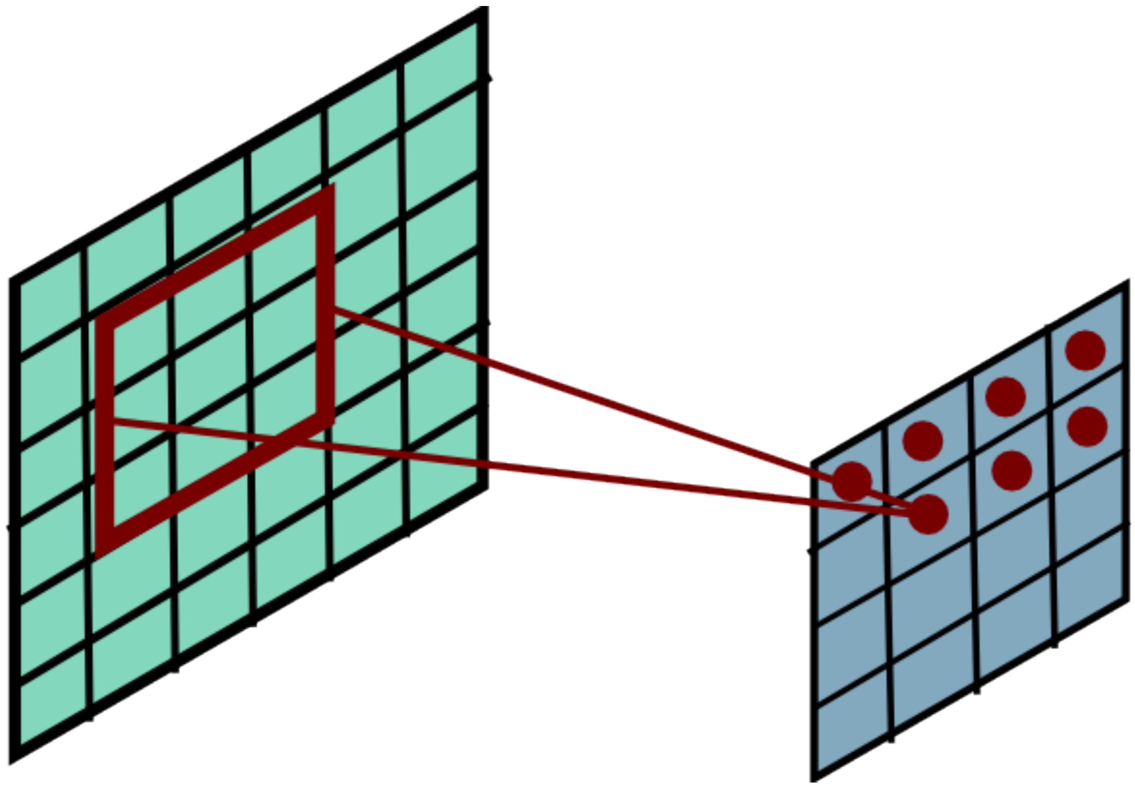
Convolutional Layer



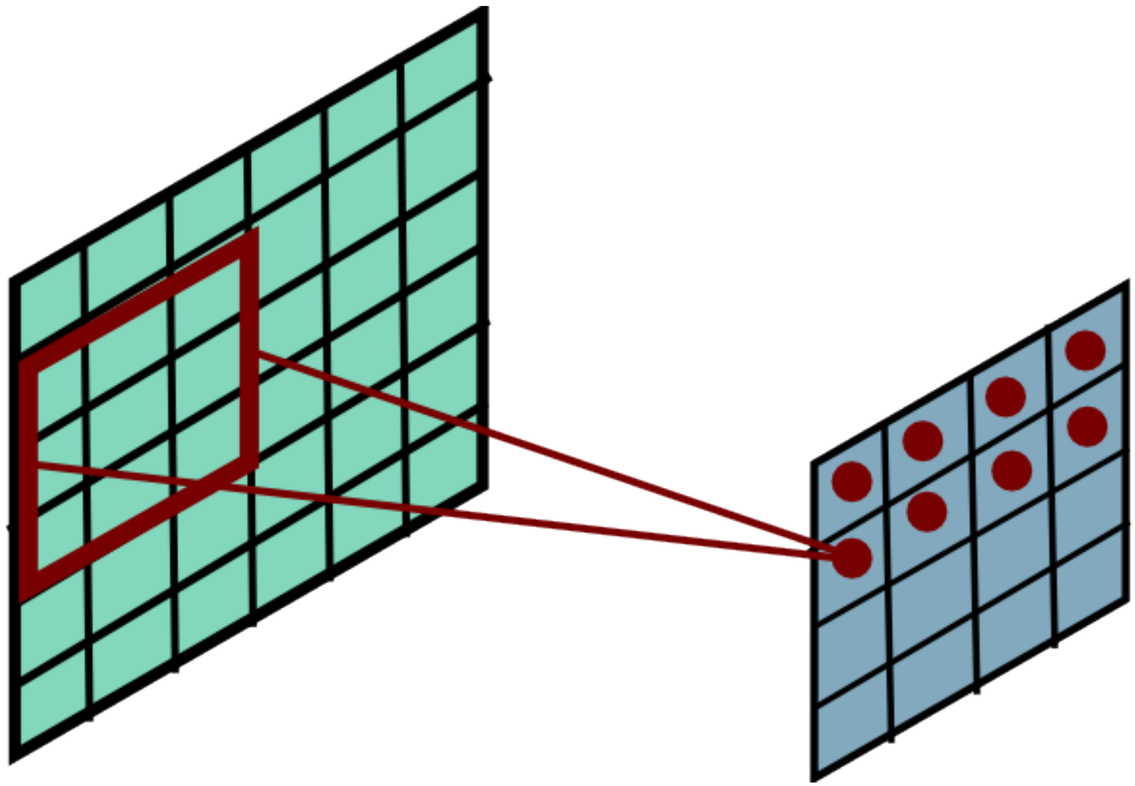
Convolutional Layer



Convolutional Layer

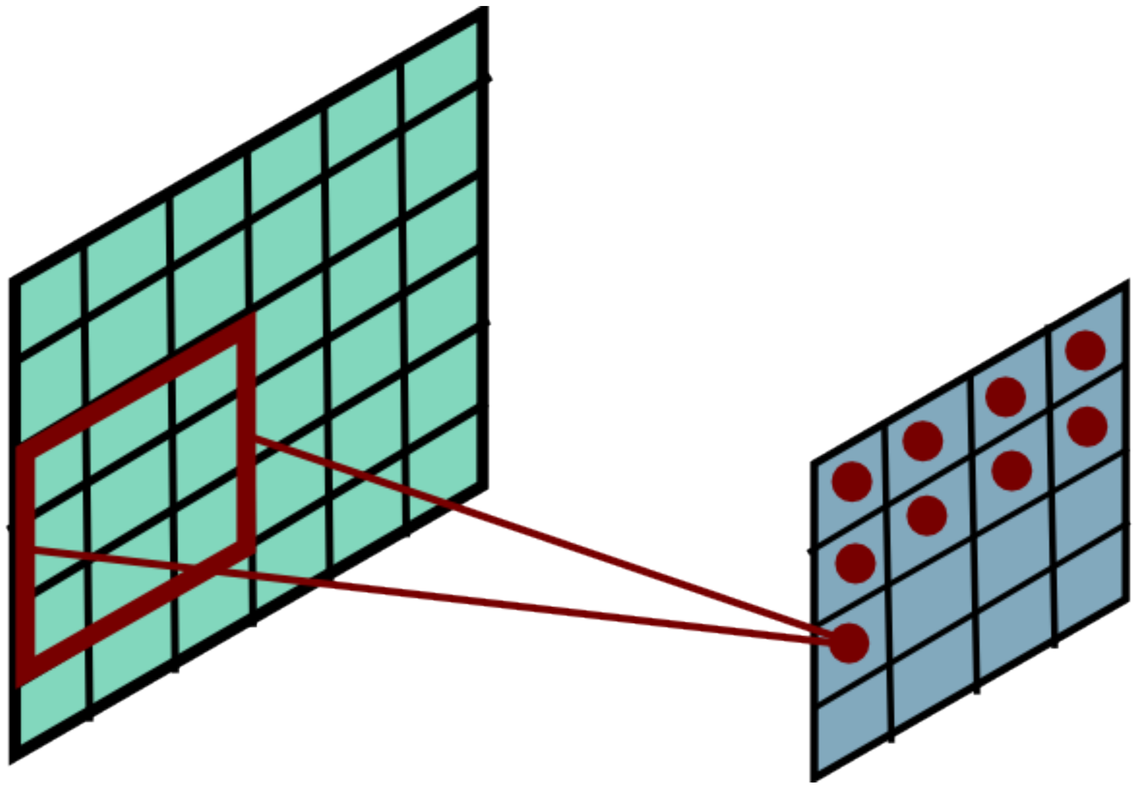


Convolutional Layer

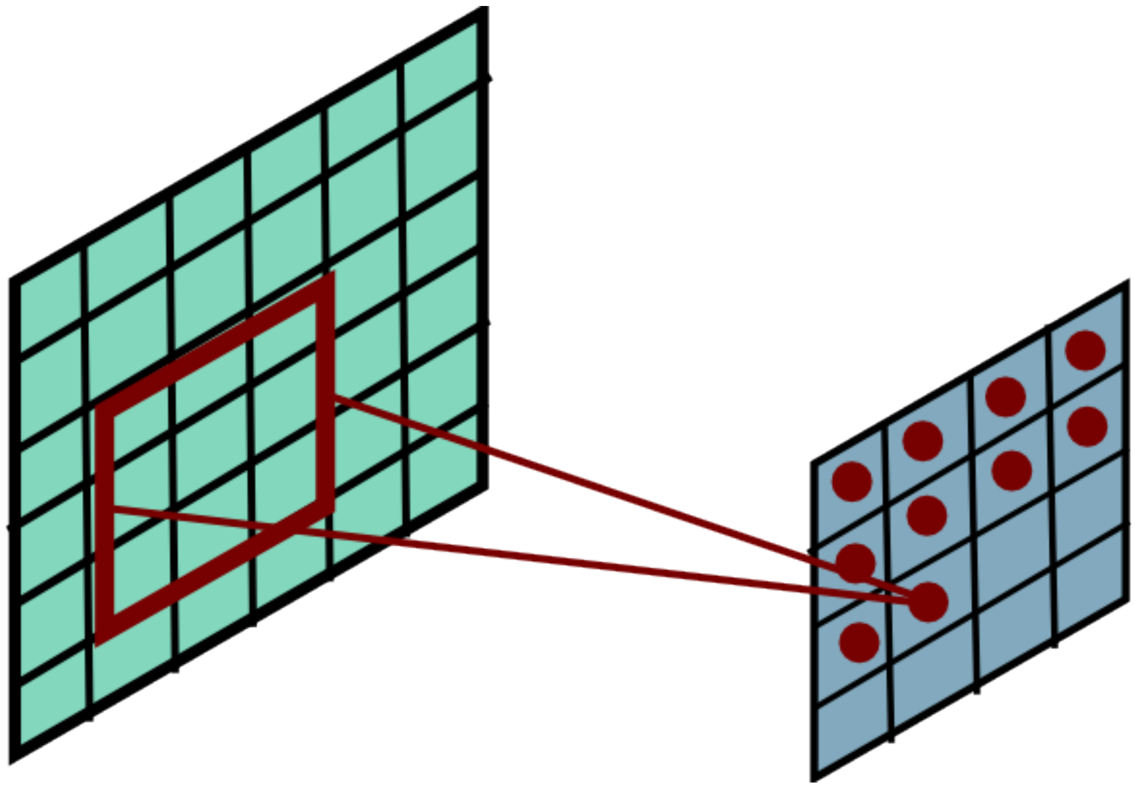


Slide Credit: Marc'Aurelio Ranzato

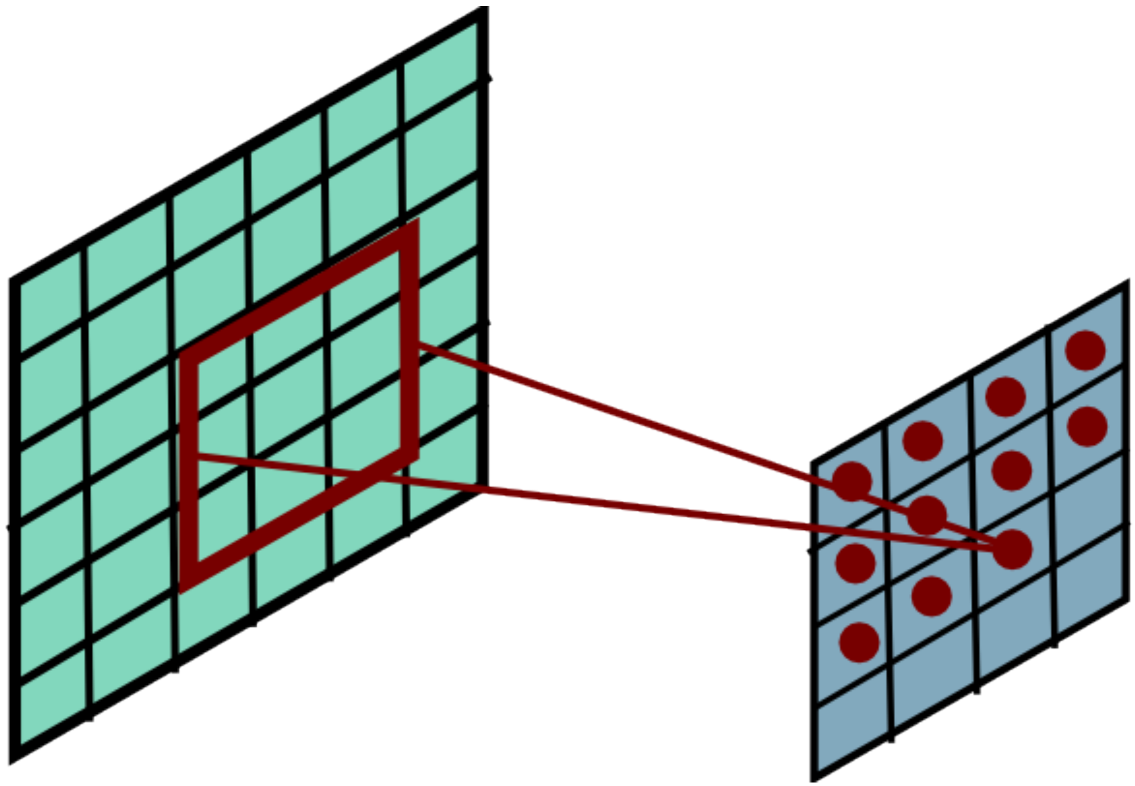
Convolutional Layer



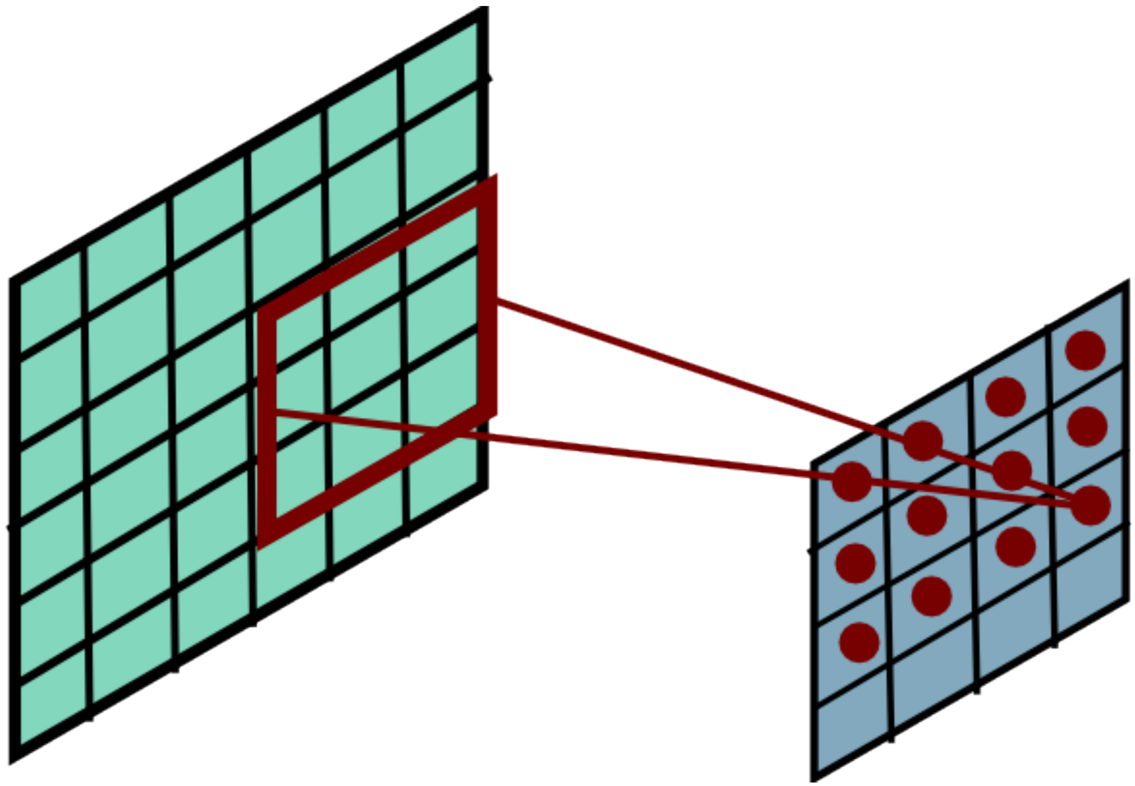
Convolutional Layer



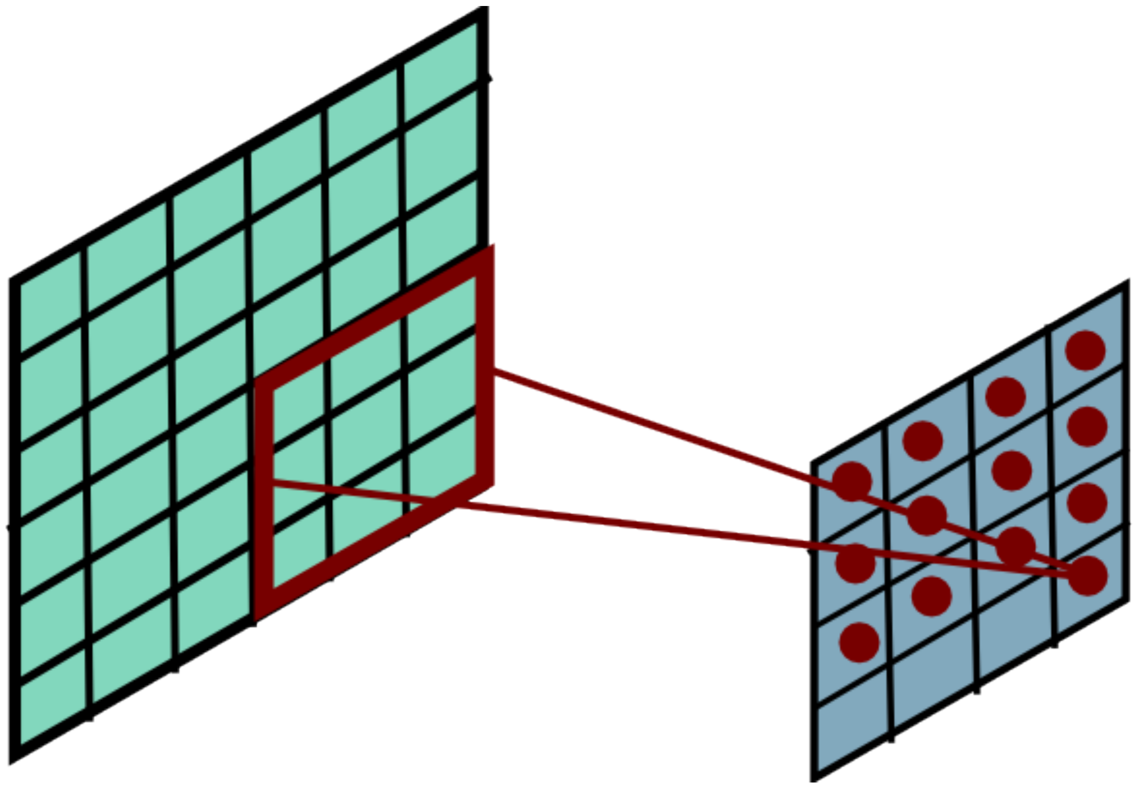
Convolutional Layer



Convolutional Layer

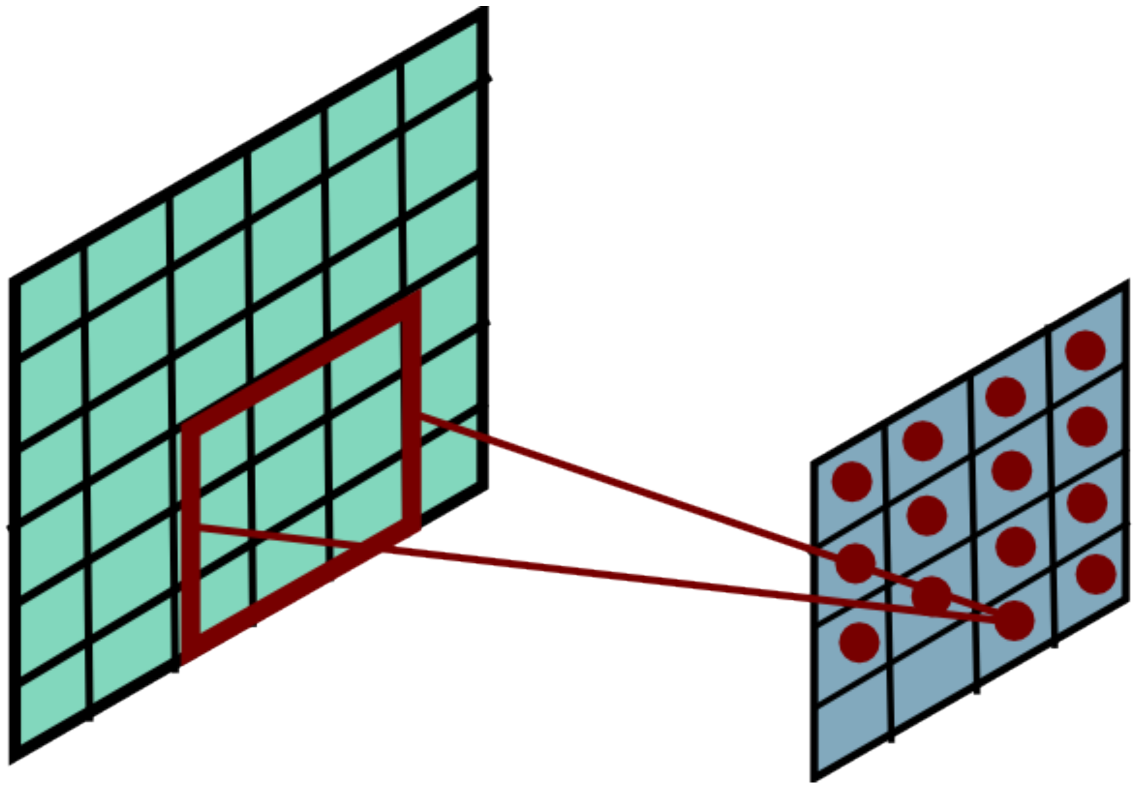


Convolutional Layer

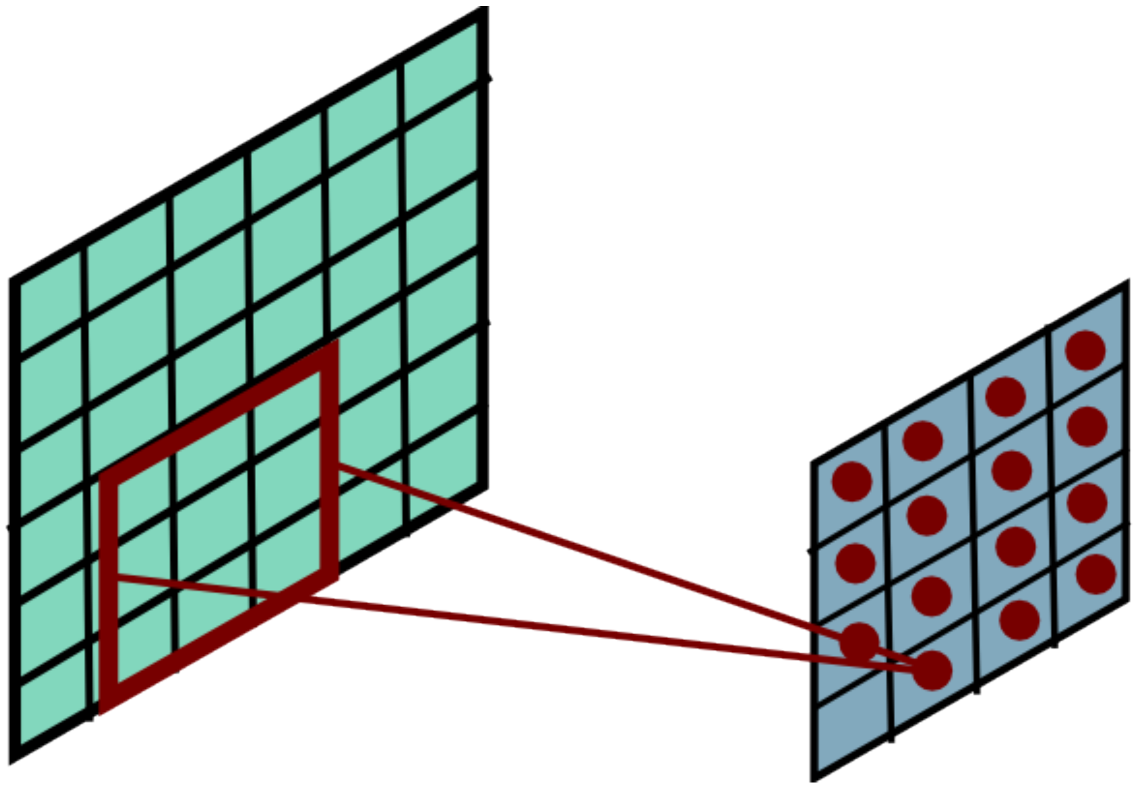


Slide Credit: Marc'Aurelio Ranzato

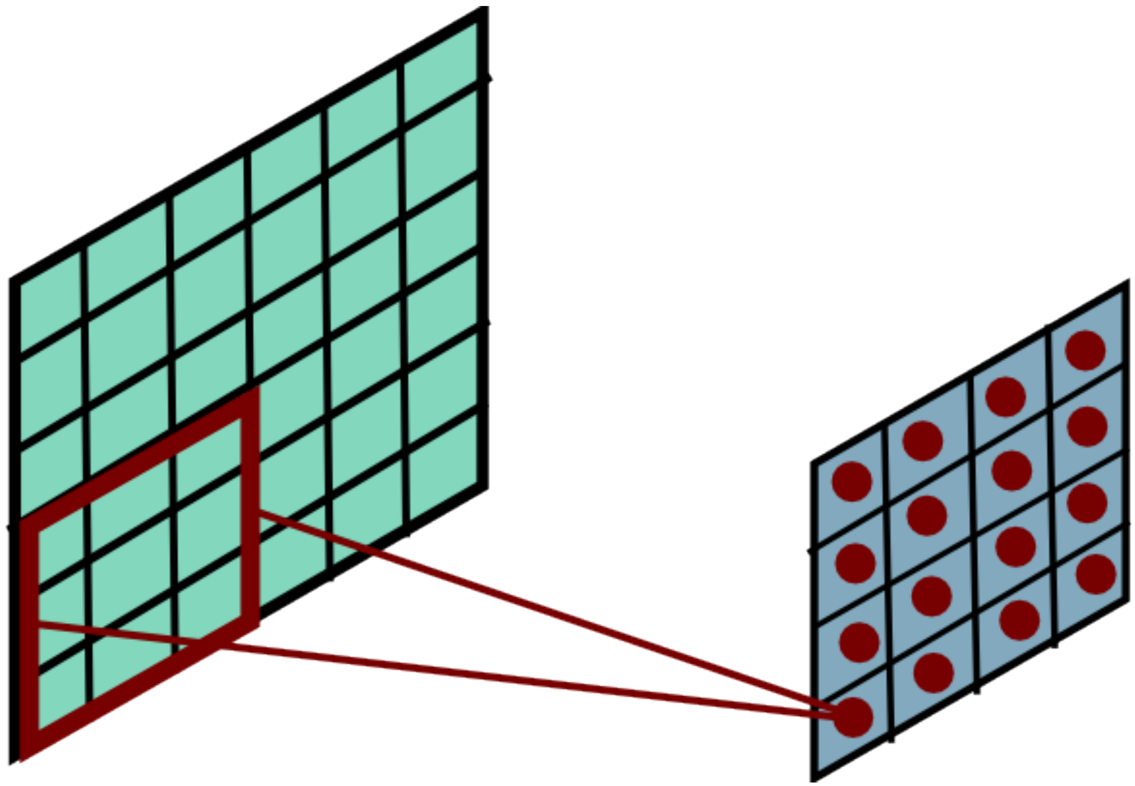
Convolutional Layer



Convolutional Layer



Convolutional Layer

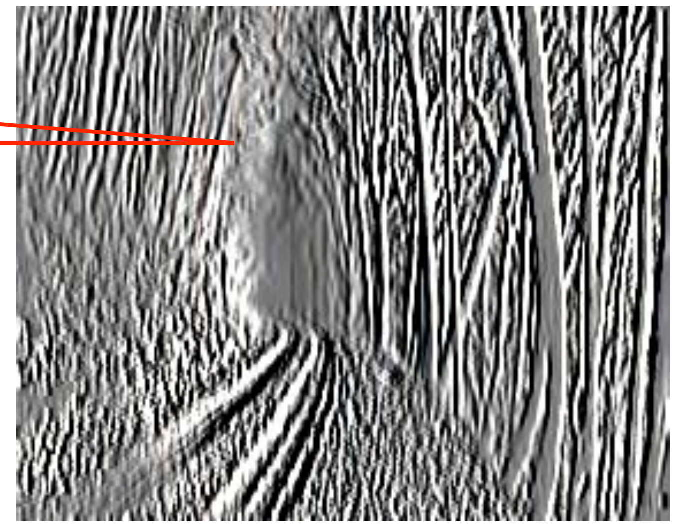


Mathieu et al. "Fast training of CNNs through FFTs" ICLR 2014

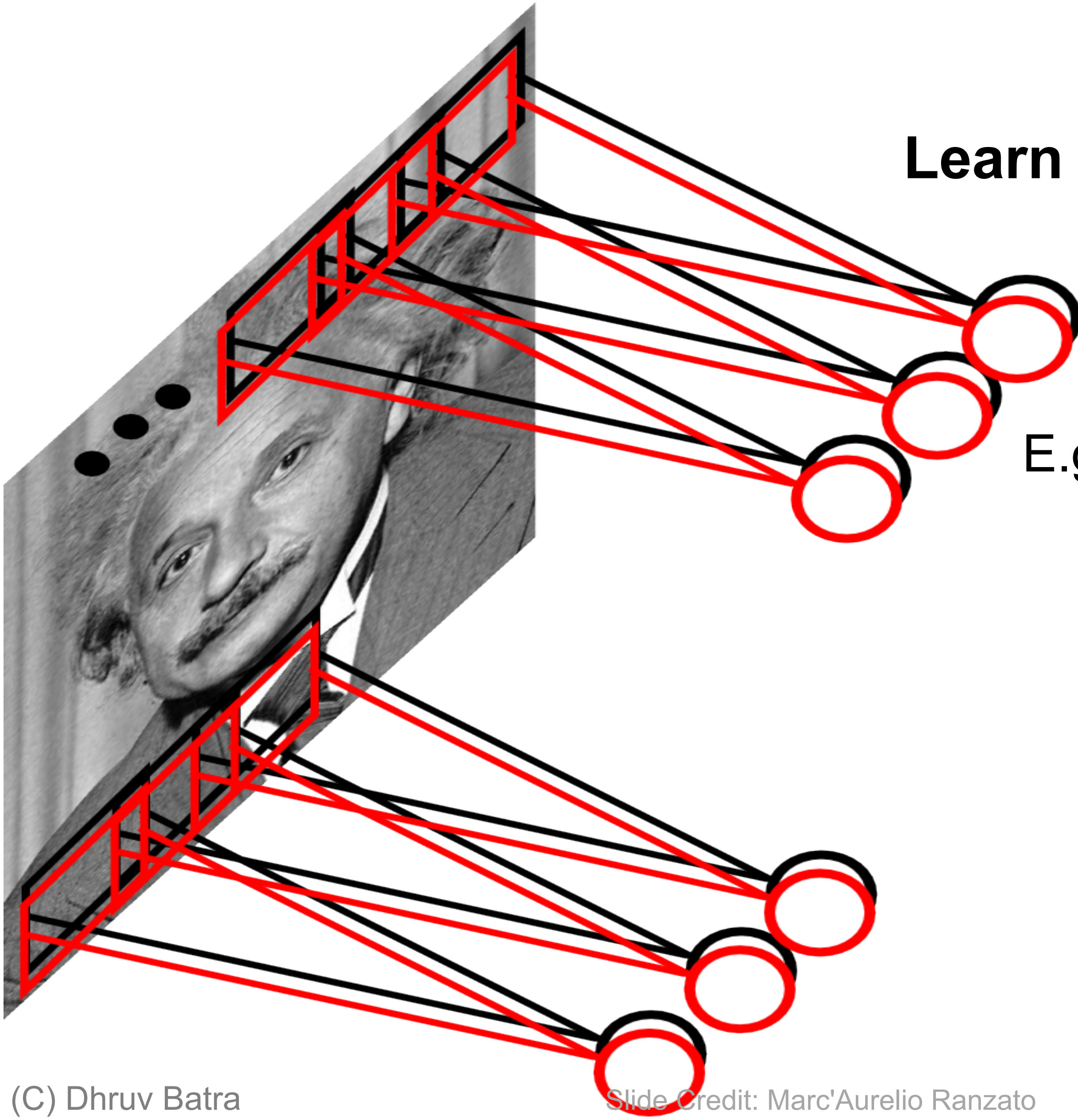
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$



Convolutional Layer

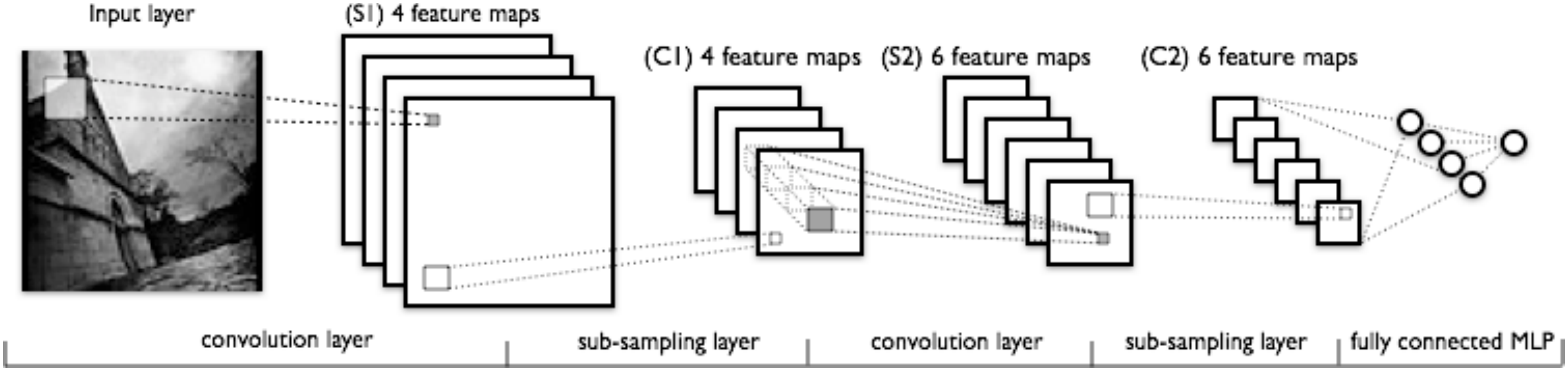


Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Slide Credit: Marc'Aurelio Ranzato

Convolutional Nets



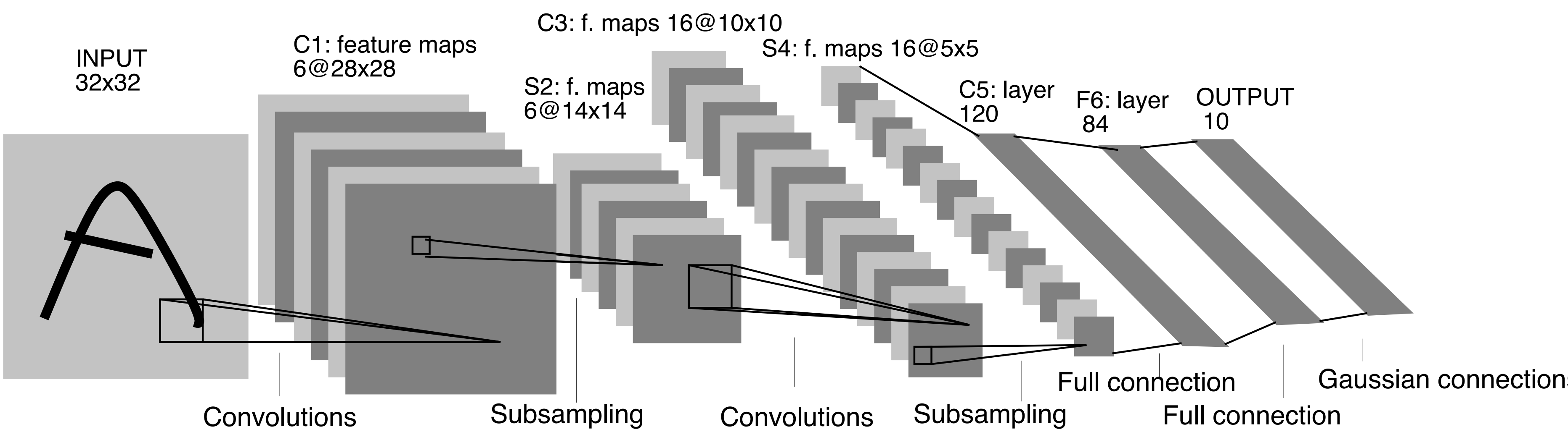


Image Credit: Yann LeCun, Kevin Murphy

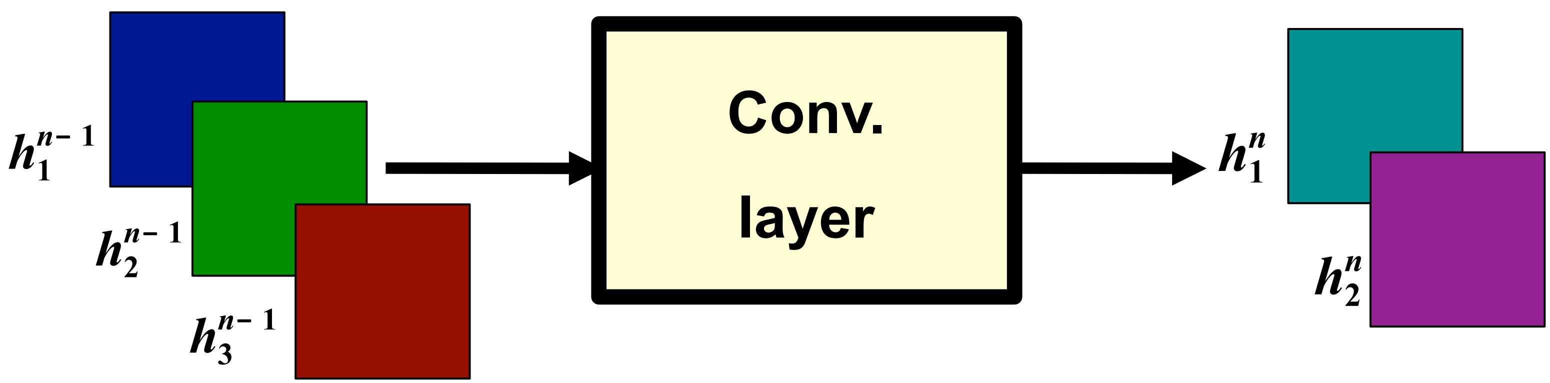
Convolutional Layer

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

output feature map

input feature map

kernel



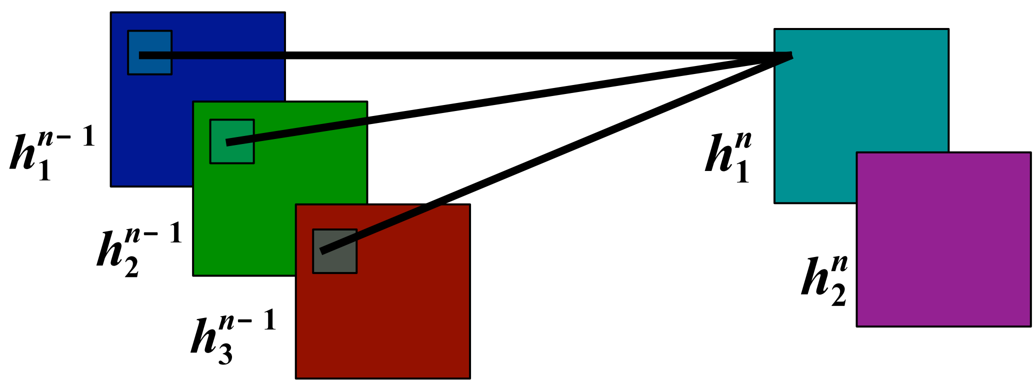
Convolutional Layer

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

output feature map

input feature map

kernel



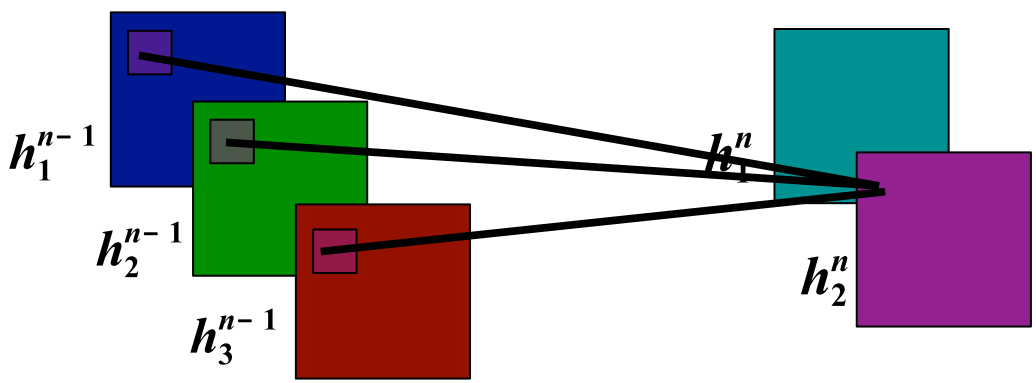
Convolutional Layer

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

output feature map

input feature map

kernel



Convolutional Layer

Question: What is the size of the output? What's the computational cost?

Answer: It is proportional to the number of filters and depends on the stride. If kernels have size KxK, input has size DxD, stride is 1, and there are M input feature maps and N output feature maps then:
- the input has size M@DxD
- the output has size N@(D-K+1)x(D-K+1)
- the kernels have MxNxKxK coefficients (which have to be learned)
- cost: M*K*K*N*(D-K+1)*(D-K+1)

Question: How many feature maps? What's the size of the filters?

Answer: Usually, there are more output feature maps than input feature maps. Convolutional layers can increase the number of hidden units by big factors (and are expensive to compute).
The size of the filters has to match the size/scale of the patterns we want to detect (task dependent).

Key Ideas

A standard neural net applied to images:

- scales quadratically with the size of the input
- does not leverage stationarity

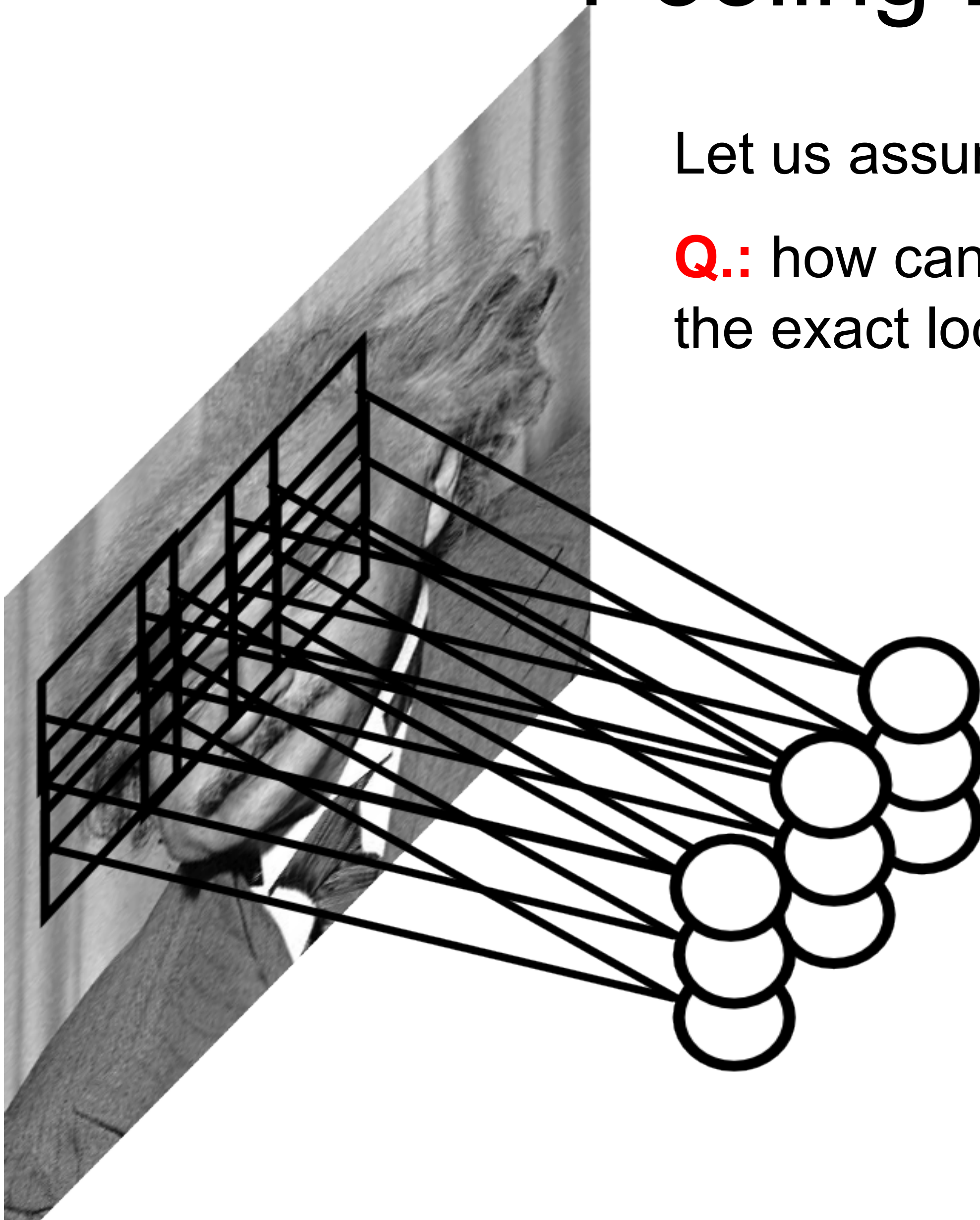
Solution:

- connect each hidden unit to a small patch of the input
- share the weight across space

This is called: **convolutional layer.**
A network with convolutional layers is called **convolutional network.**

LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998

Pooling Layer



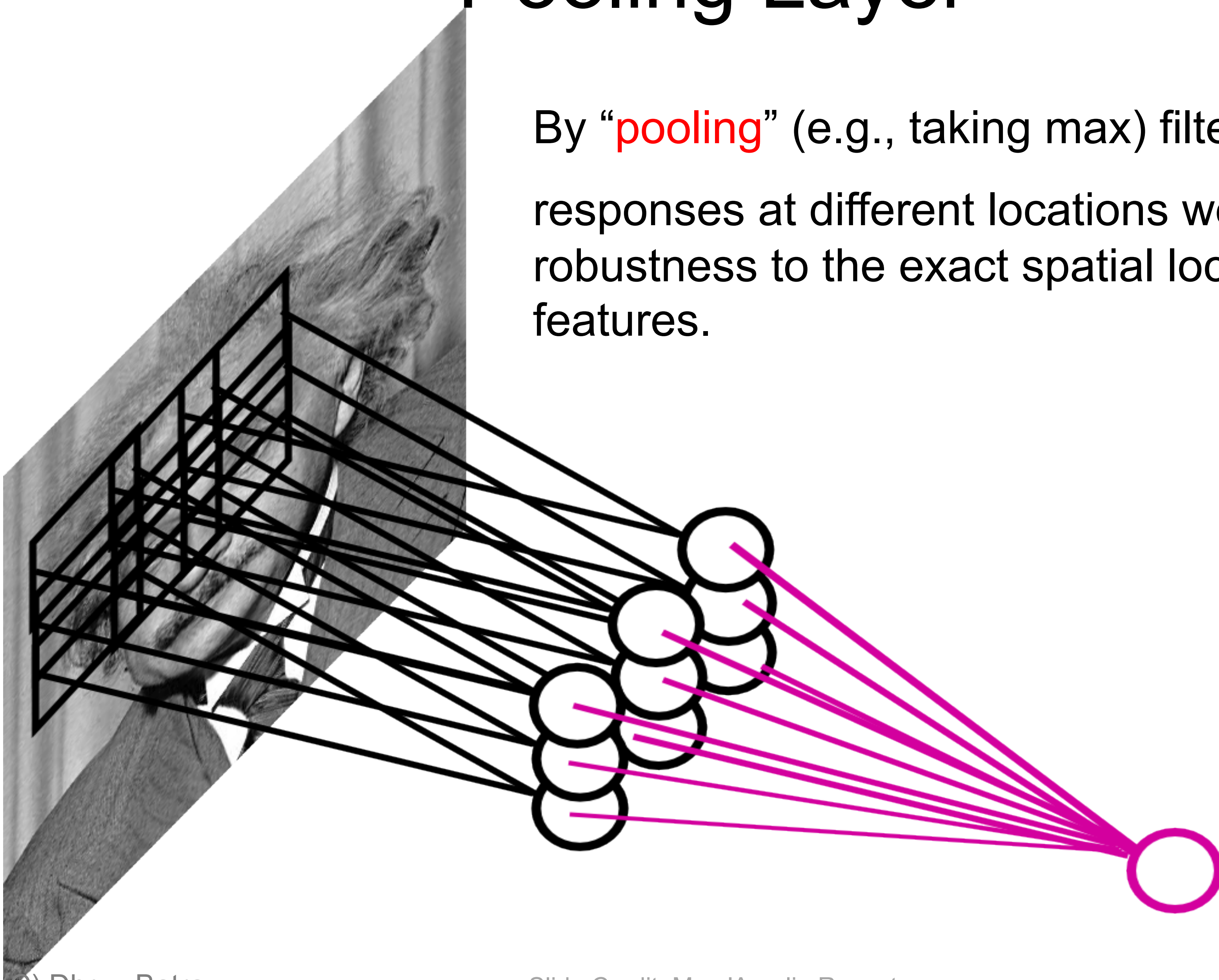
Let us assume filter is an "eye" detector.

Q.: how can we make the detection robust to the exact location of the eye?

Pooling Layer

By “pooling” (e.g., taking max) filter

responses at different locations we gain robustness to the exact spatial location of features.



Slide Credit: Marc'Aurelio Ranzato

Pooling Layer: Examples

Max-pooling:

$$h_i^n(r,c) = \max_{\bar{r} \in N(r),\ \bar{c} \in N(c)} h_i^{n-1}(\bar{r},\bar{c})$$

Average-pooling:

$$h_i^n(r,c) = \operatorname*{mean}_{\bar{r} \in N(r),\ \bar{c} \in N(c)} h_i^{n-1}(\bar{r},\bar{c})$$

L2-pooling:

$$h_i^n(r,c) = \sqrt{\sum_{\bar{r} \in N(r),\ \bar{c} \in N(c)} h_i^{n-1}(\bar{r},\bar{c})^2}$$

L2-pooling over features:

$$h_i^n(r,c) = \sqrt{\sum_{j \in N(i)} h_i^{n-1}(r,c)^2}$$

Pooling Layer

Question: What is the size of the output? What's the computational cost?

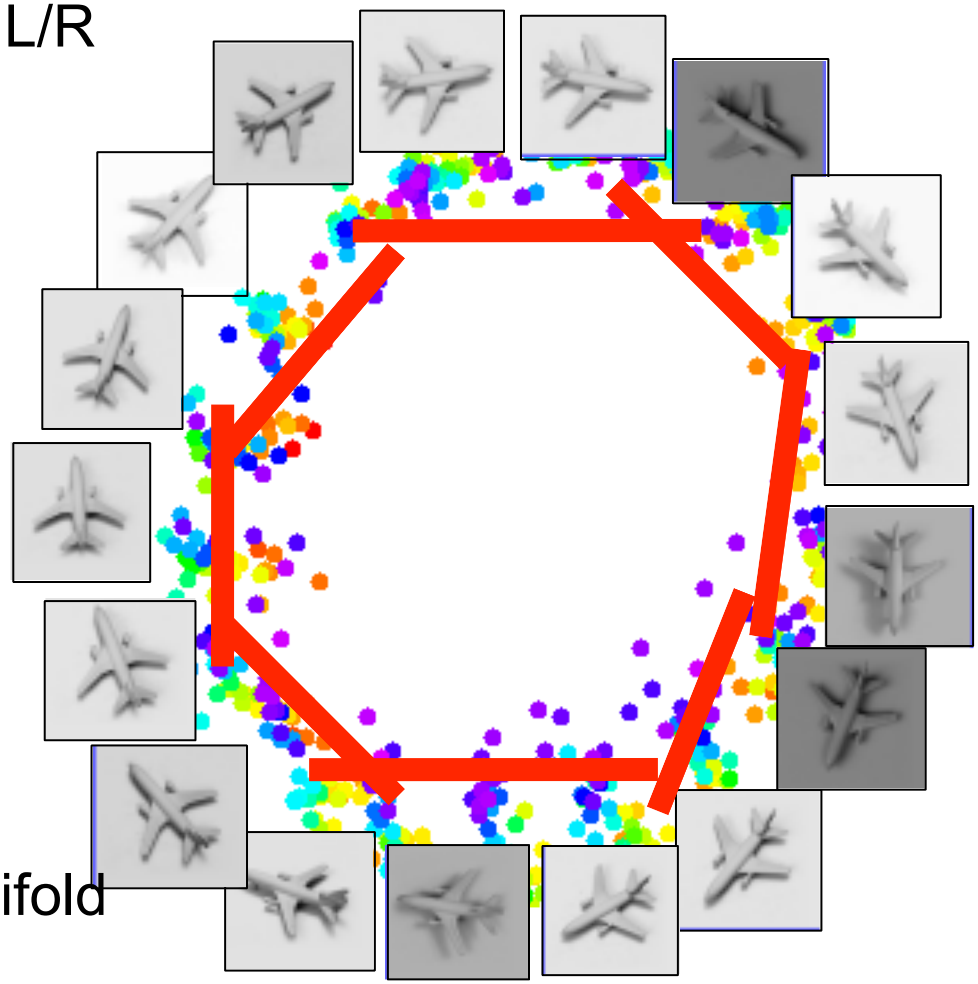
Answer: The size of the output depends on the stride between the pools. For instance, if pools do not overlap and have size KxK, and the input has size DxD with M input feature maps, then:
- output is M@(D/K)x(D/K)
- the computational cost is proportional to the size of the input (negligible compared to a convolutional layer)

Question: How should I set the size of the pools?

Answer: It depends on how much “invariant” or robust to distortions we want the representation to be. It is best to pool slowly (via a few stacks of conv-pooling layers).

Pooling Layer: Interpretation

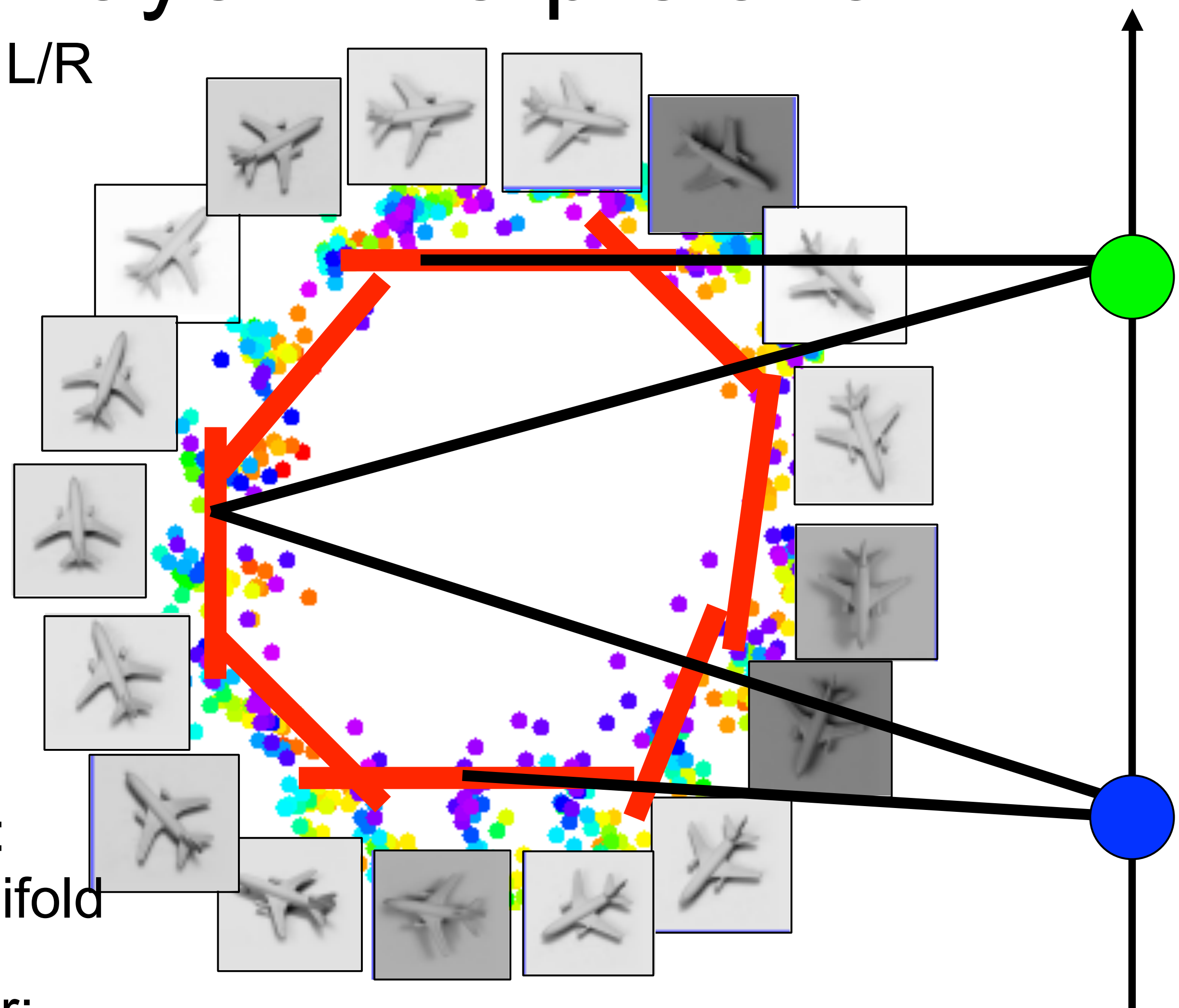
Task: detect orientation L/R



Conv layer:
linearizes manifold

Pooling Layer: Interpretation

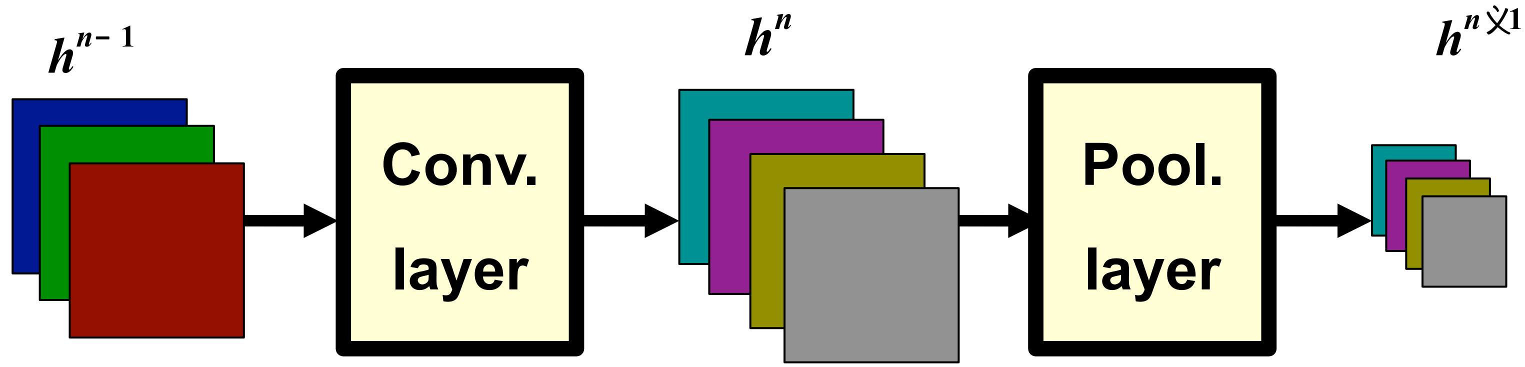
Task: detect orientation L/R



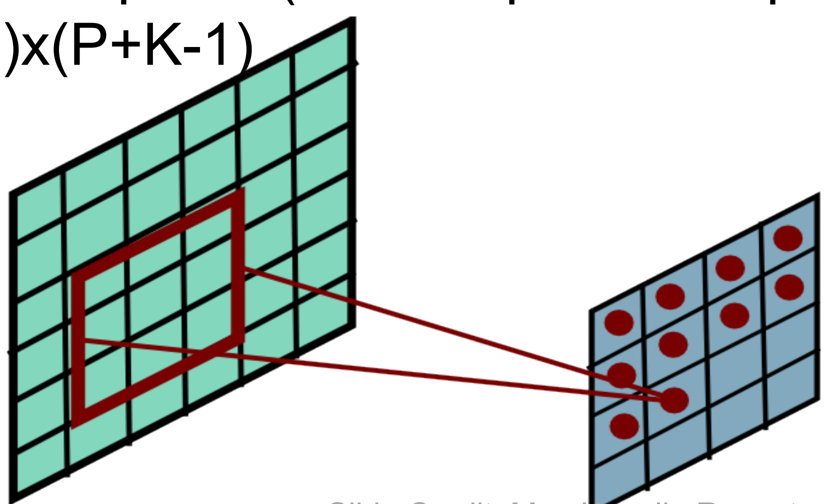
Conv layer:
linearizes manifold

Pooling layer:
collapses manifold

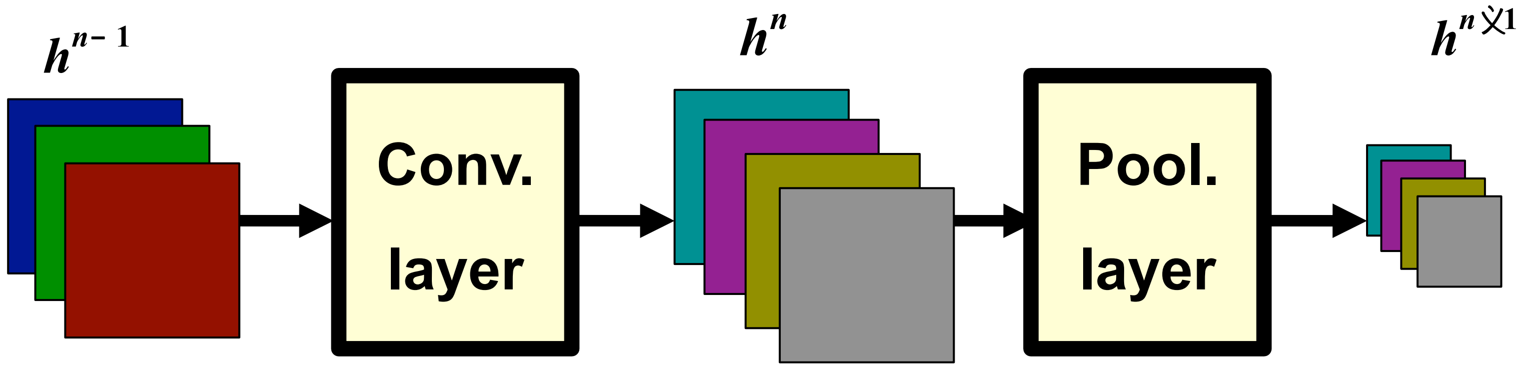
Pooling Layer: Receptive Field Size



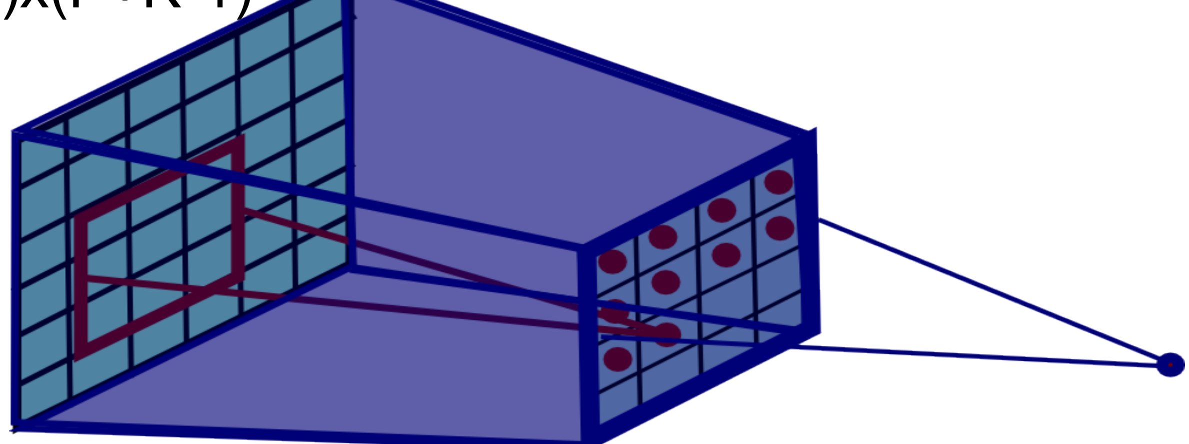
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



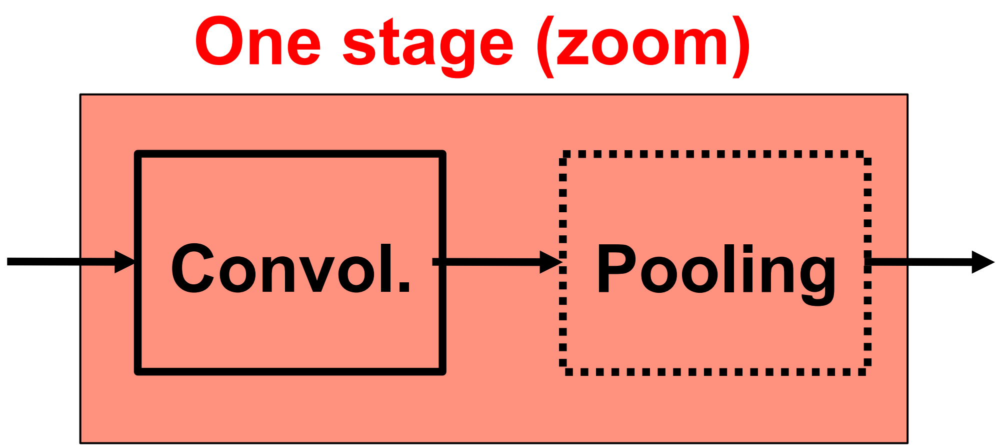
Pooling Layer: Receptive Field Size

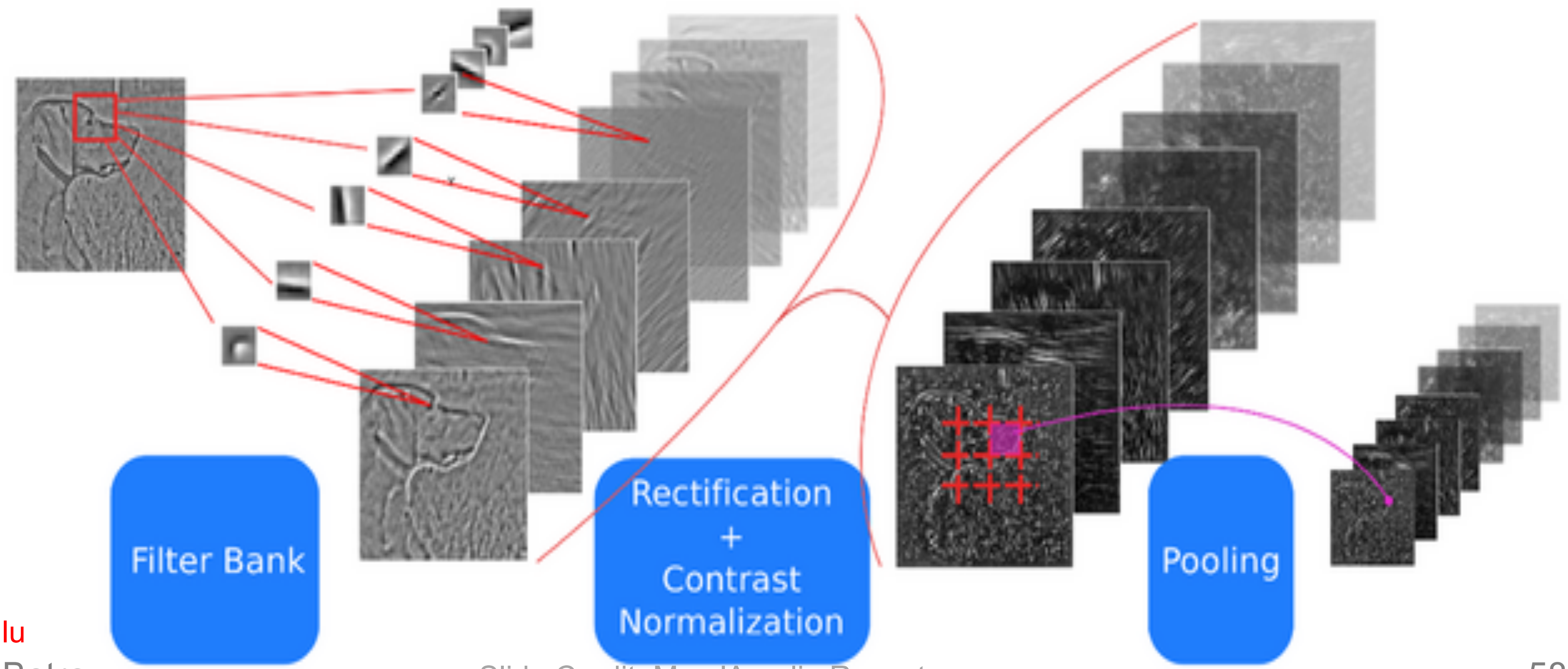


If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



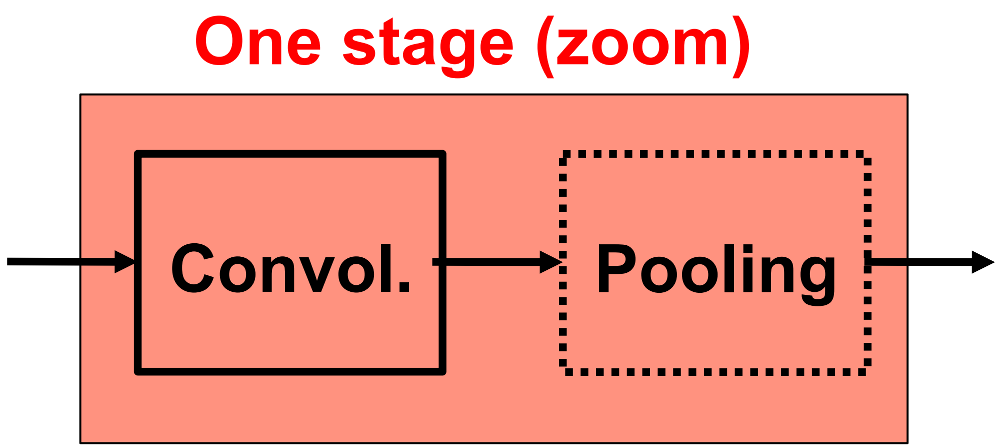
ConvNets: Typical Stage





courtesy of
K. Kavukcuoglu

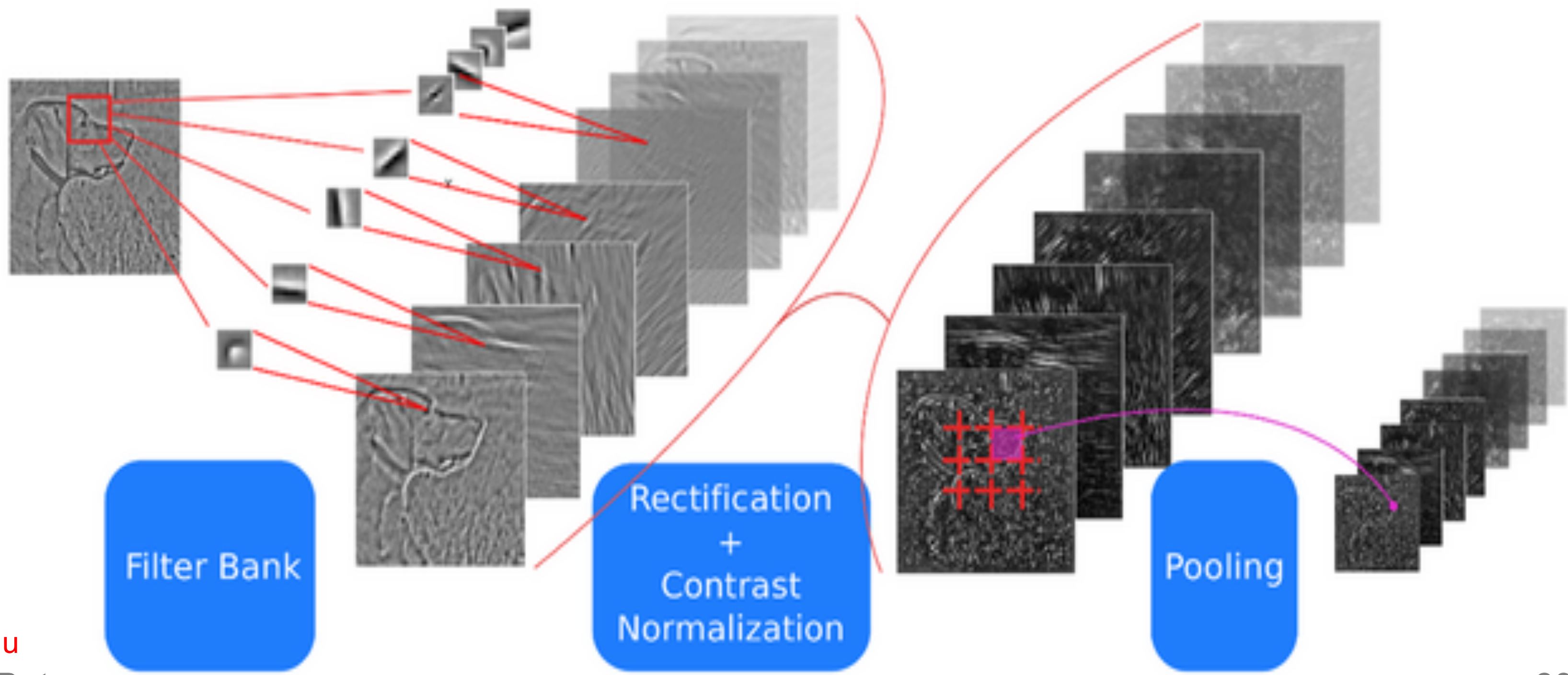
ConvNets: Typical Stage



Conceptually similar to: SIFT, HoG, etc.

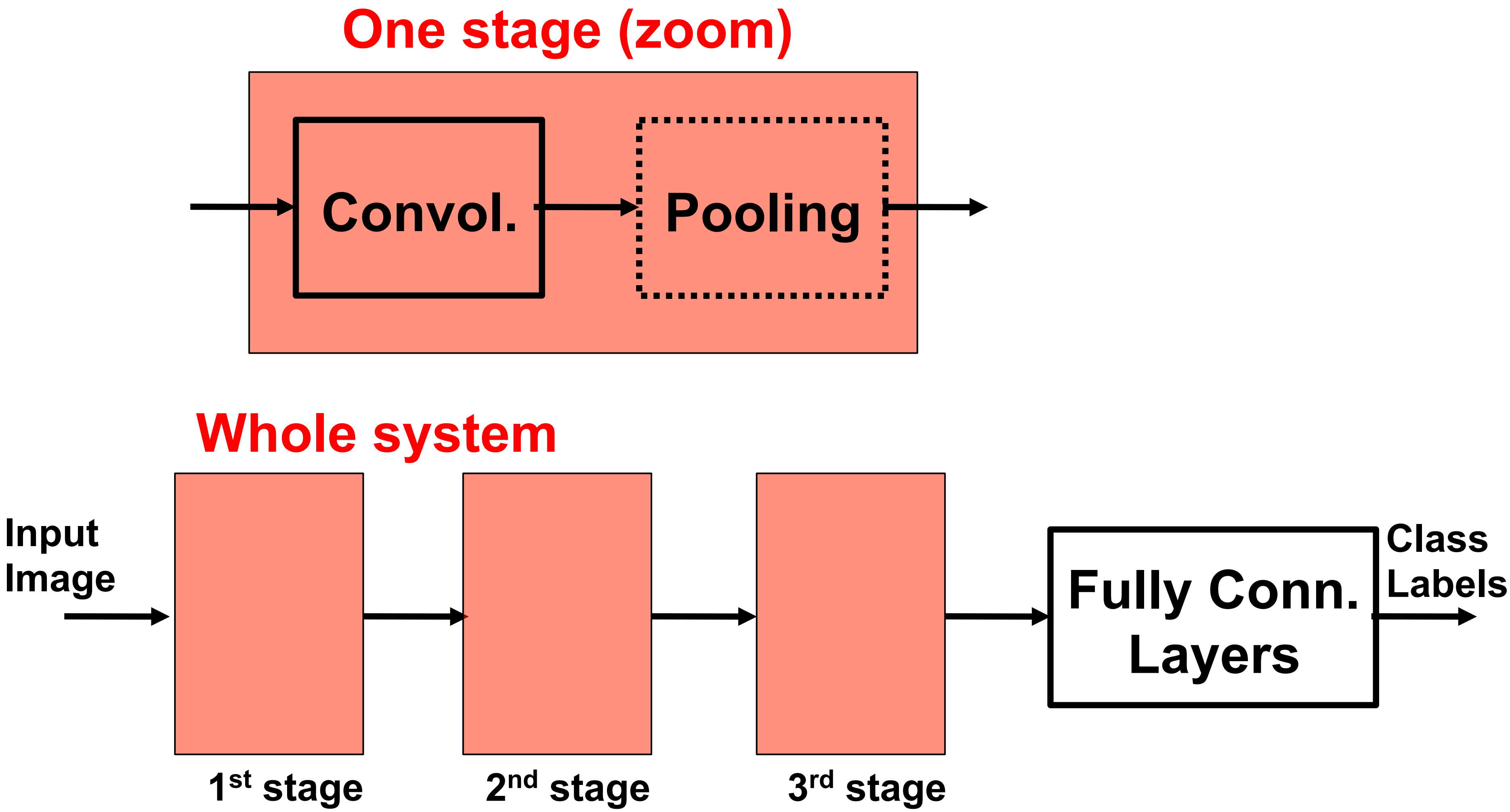
Note: after one stage the number of feature maps is usually increased (conv. layer) and the spatial resolution is usually decreased (stride in conv. and pooling layers). Receptive field gets bigger.

Reasons:
- gain invariance to spatial translation (pooling layer)
- increase specificity of features (approaching object specific units)



courtesy of
K. Kavukcuoglu

ConvNets: Typical Architecture



Visualizing Learned Filters

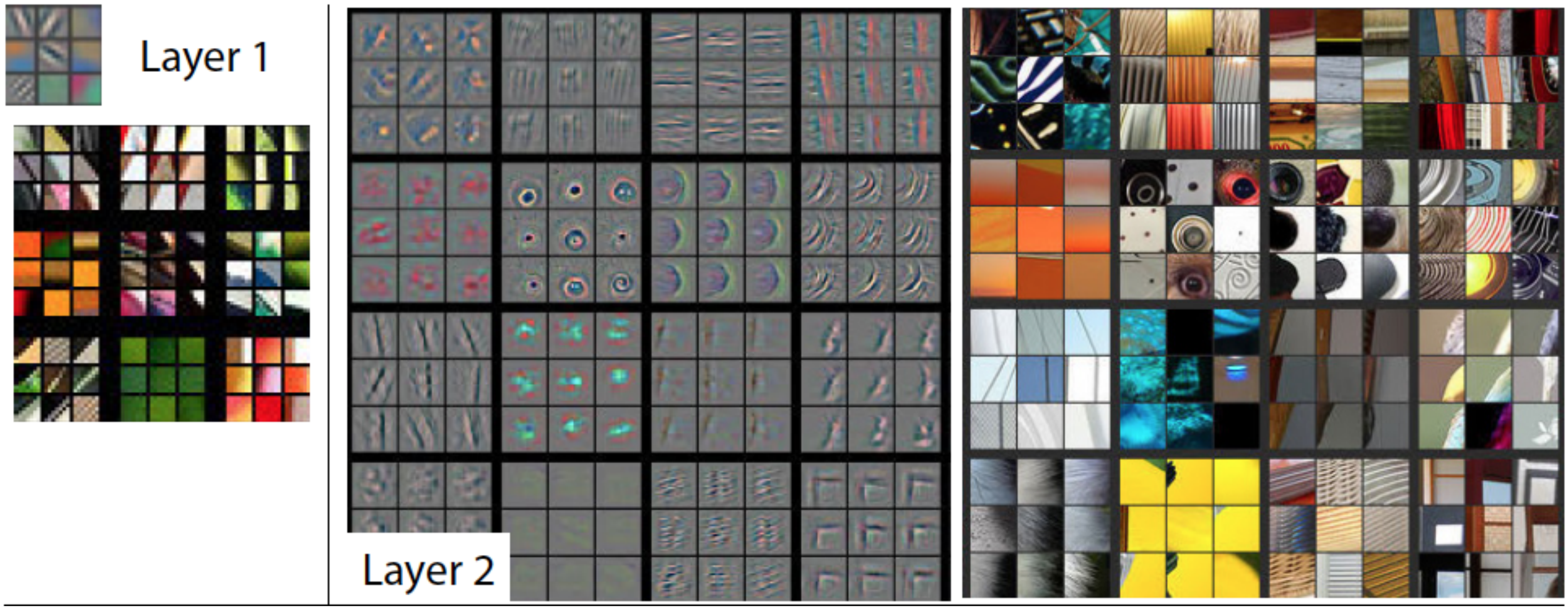
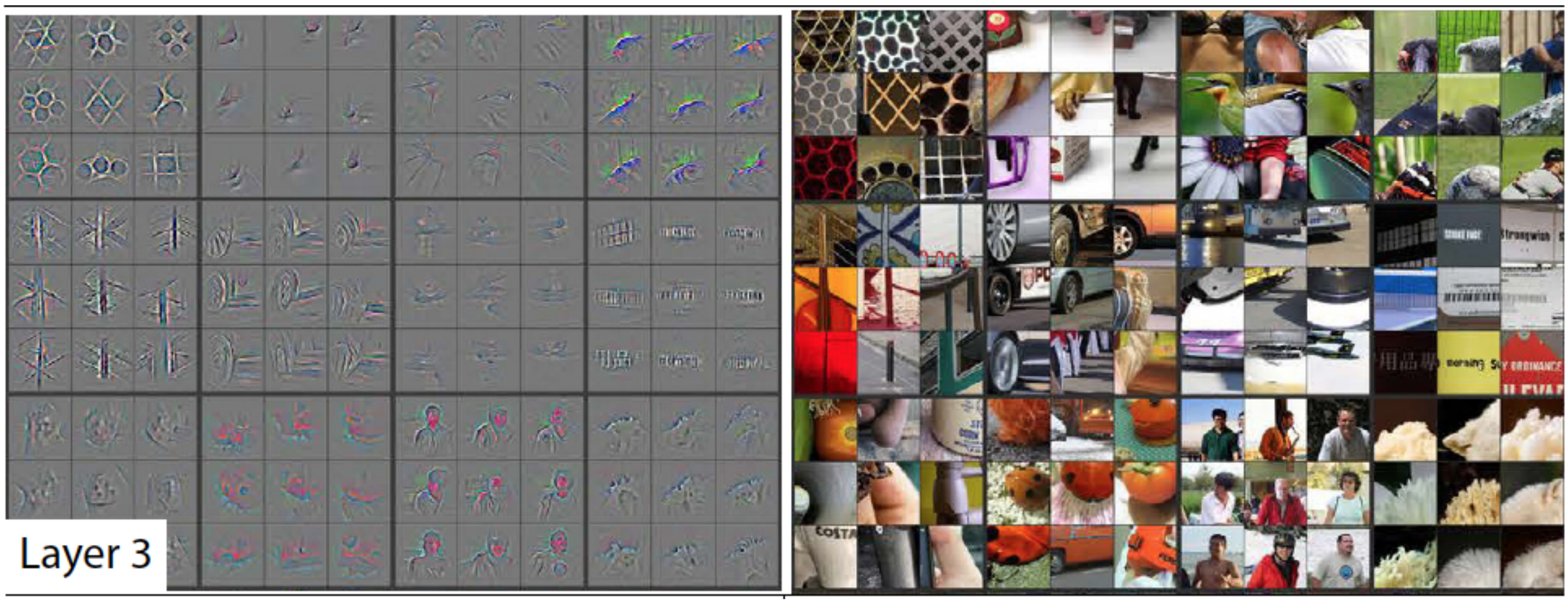
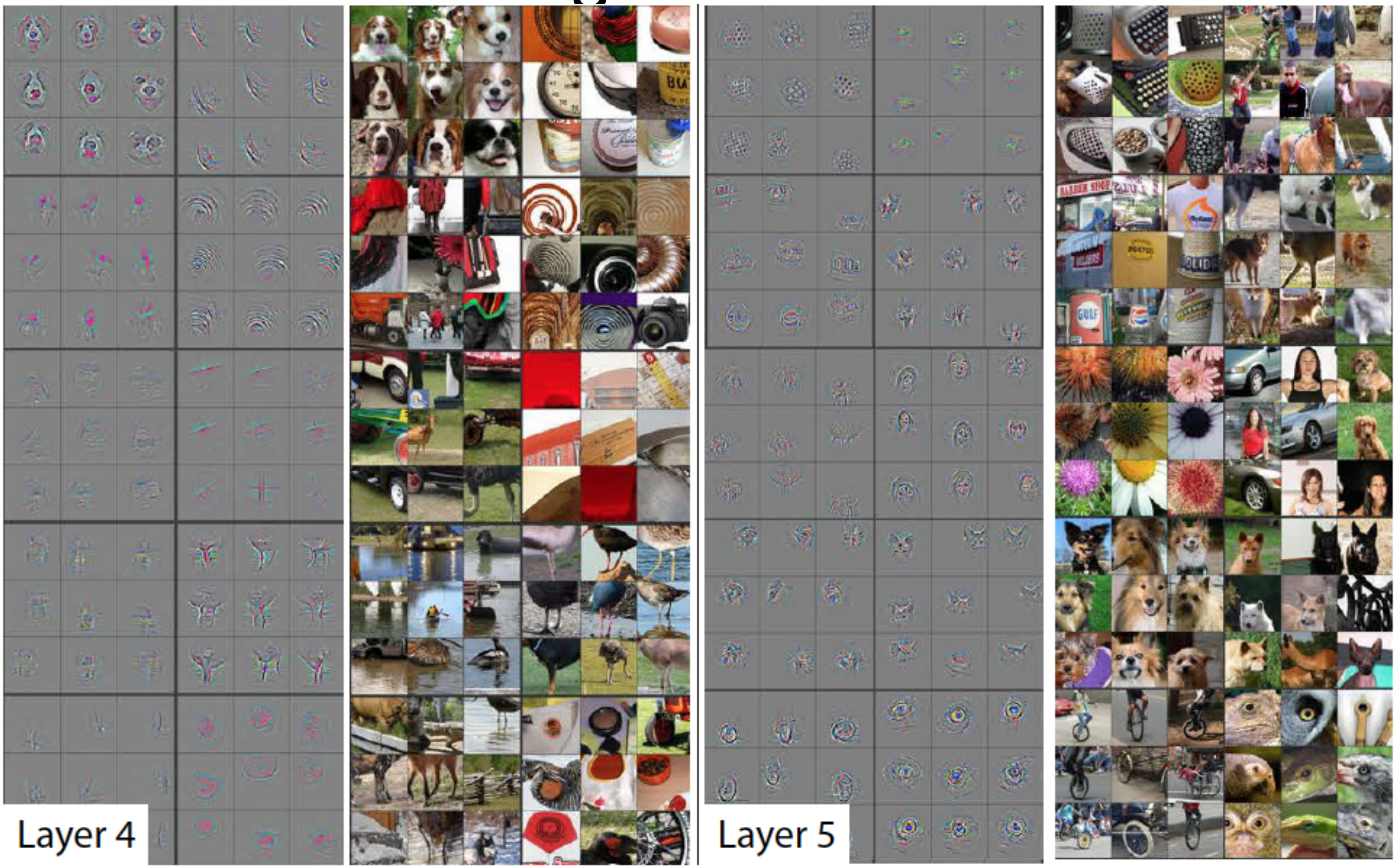


Figure Credit: [Zeiler & Fergus ECCV14]

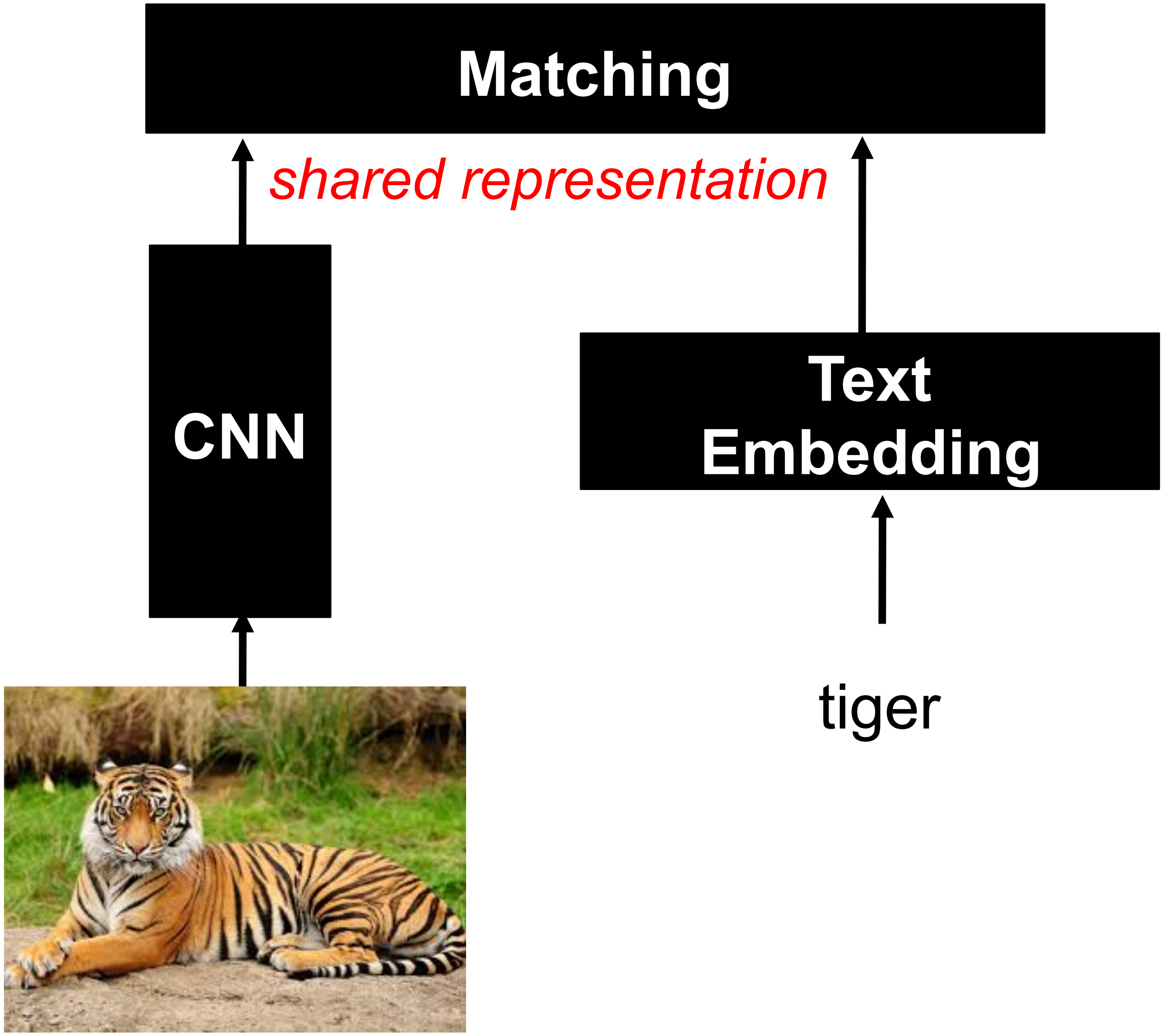
Visualizing Learned Filters



Visualizing Learned Filters

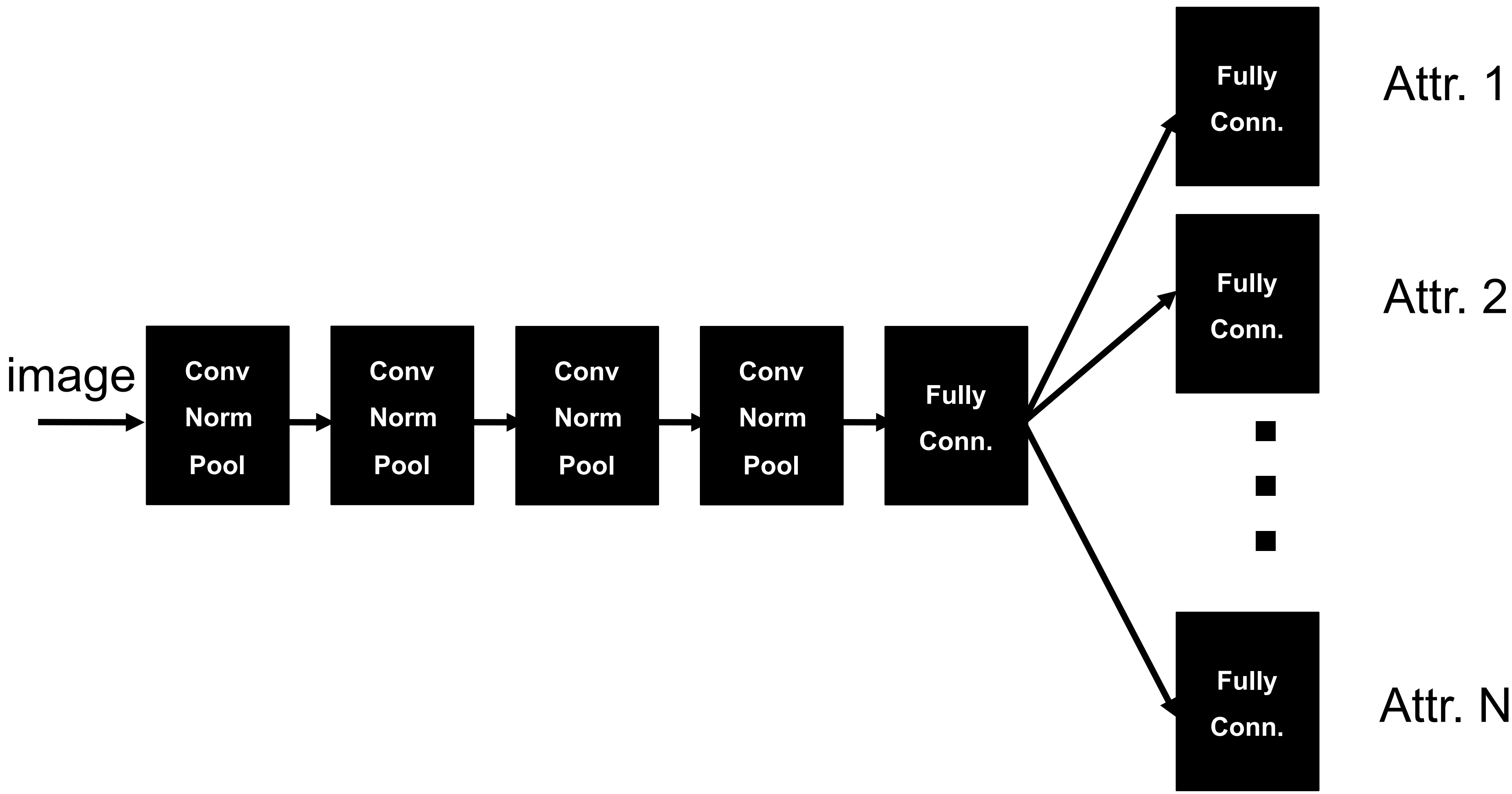


Fancier Architectures: Multi-Modal



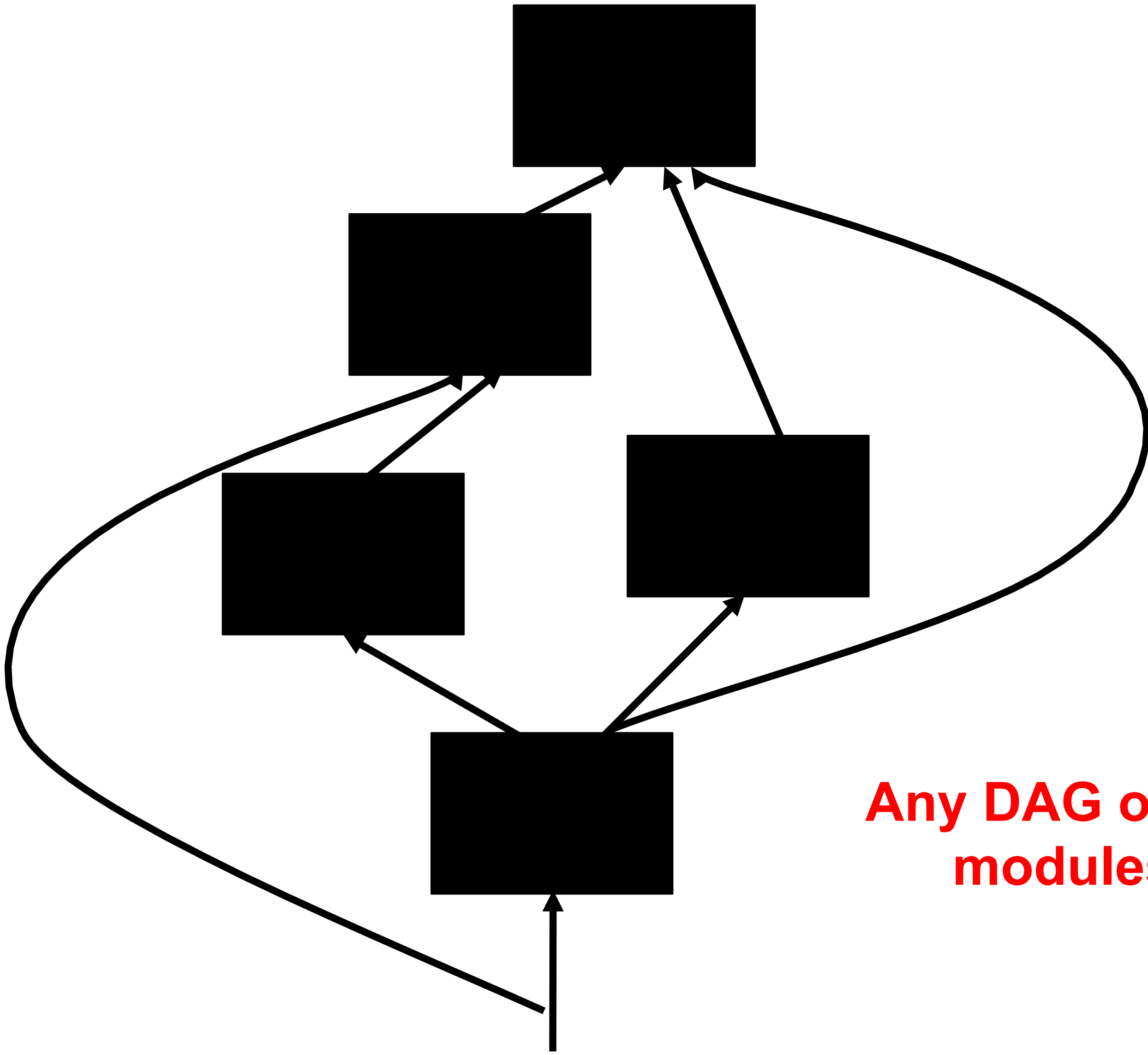
Frome et al. “Devise: a deep visual semantic embedding model” NIPS 2013

Fancier Architectures: Multi-Task



Zhang et al. “PANDA..” CVPR 2014

Fancier Architectures: Generic DAG



Any DAG of differentialble modules is allowed!