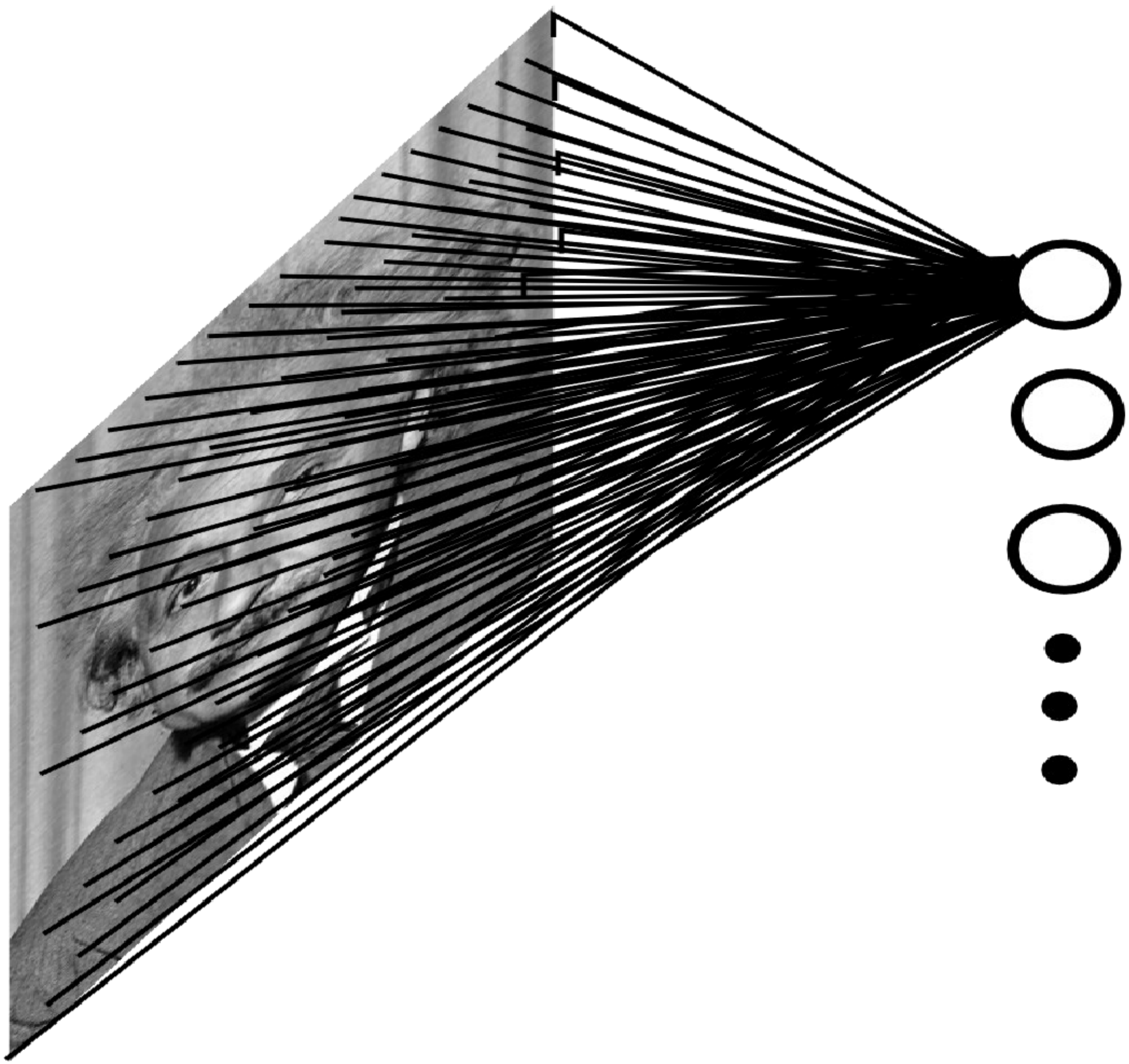
CSE 152: Computer Vision

Hao Su

Convolutional Neural Network

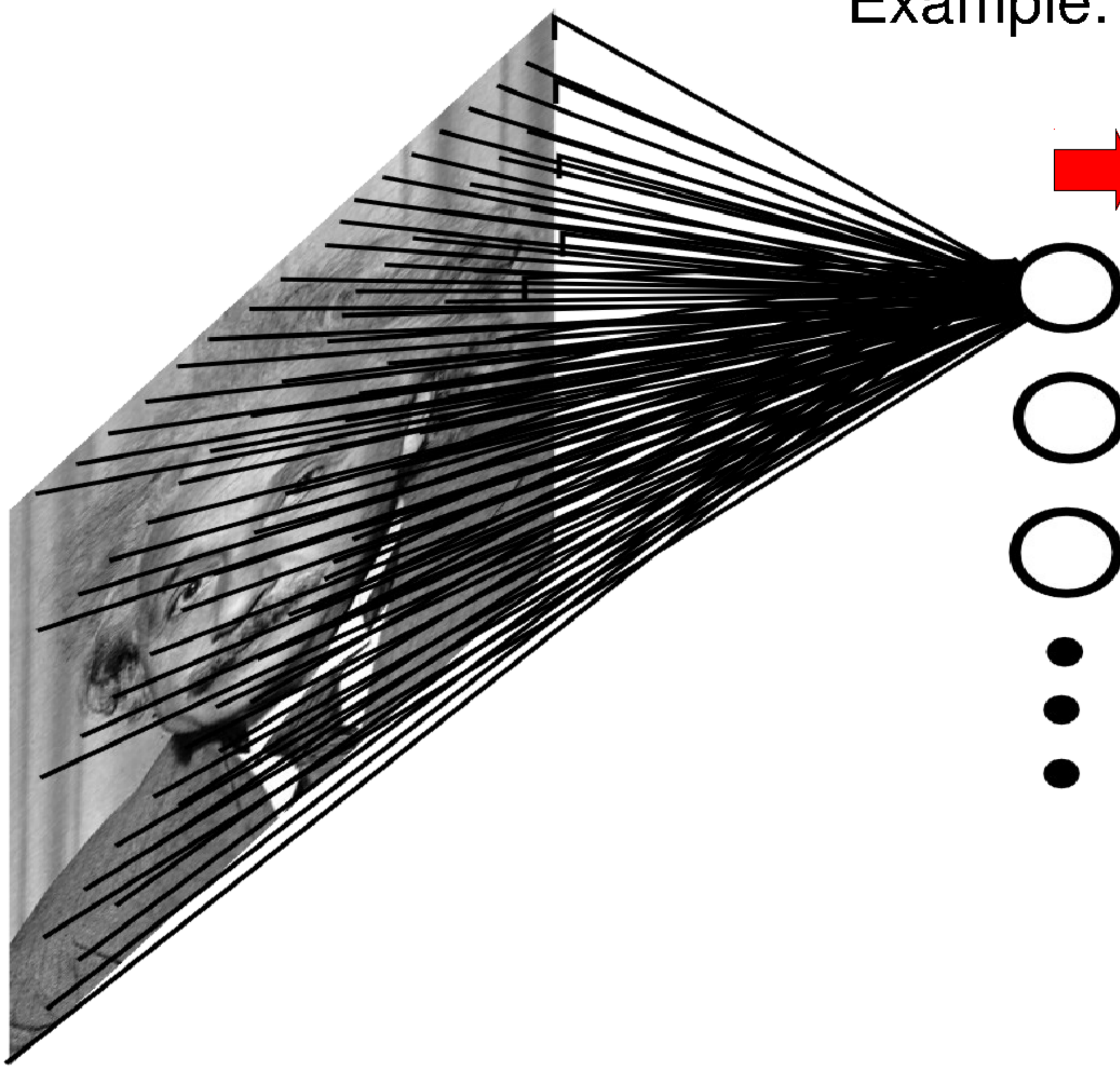
Images as input to neural networks



Ranzato

Images as input to neural networks

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



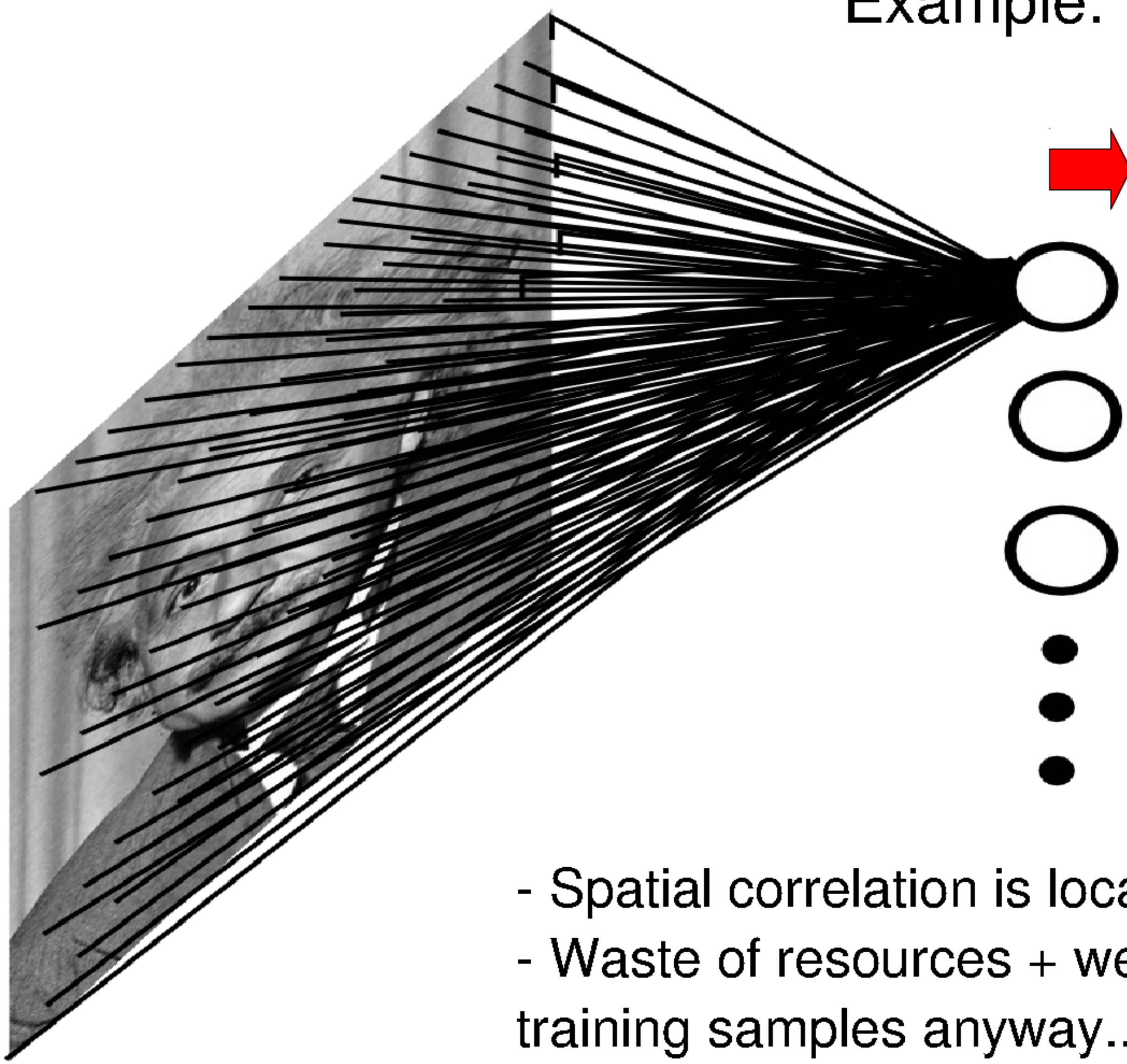
Ranzato

Images as input to neural networks

Example: 200x200 image

40K hidden units

~2B parameters!!!



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

Ranzato

Convolutional Neural Networks

- CNN = a multi-layer neural network with
  - **Local** connectivity:
    - Neurons in a layer are only connected to a small region of the layer before it
  - **Share** weight parameters across spatial positions:
    - Learning shift-invariant filter kernels

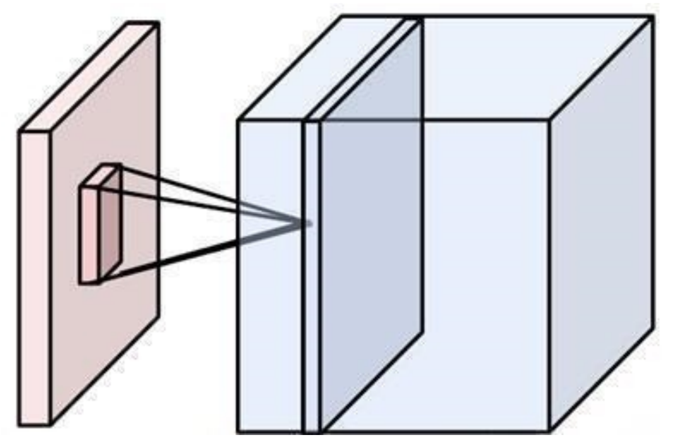
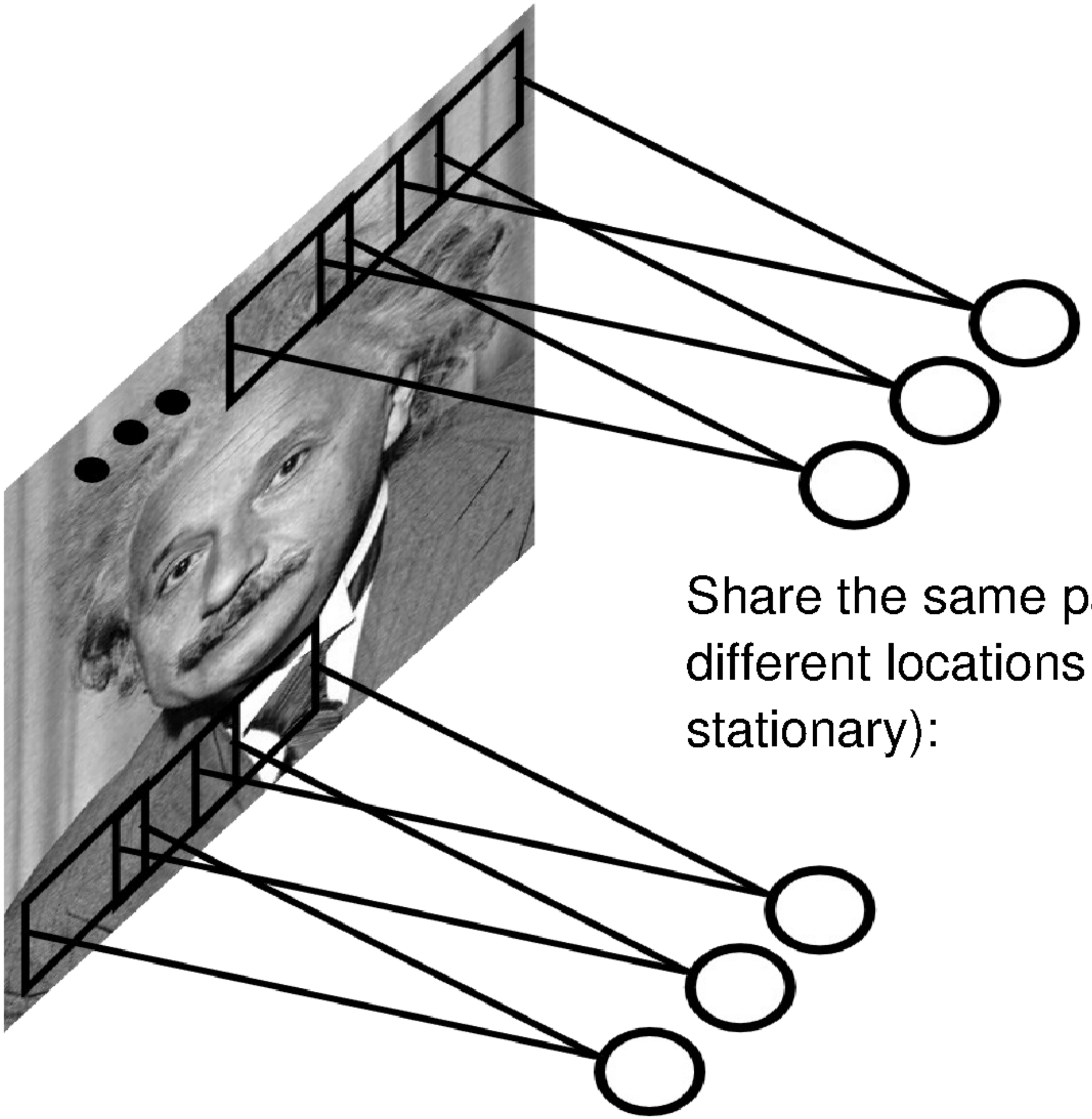
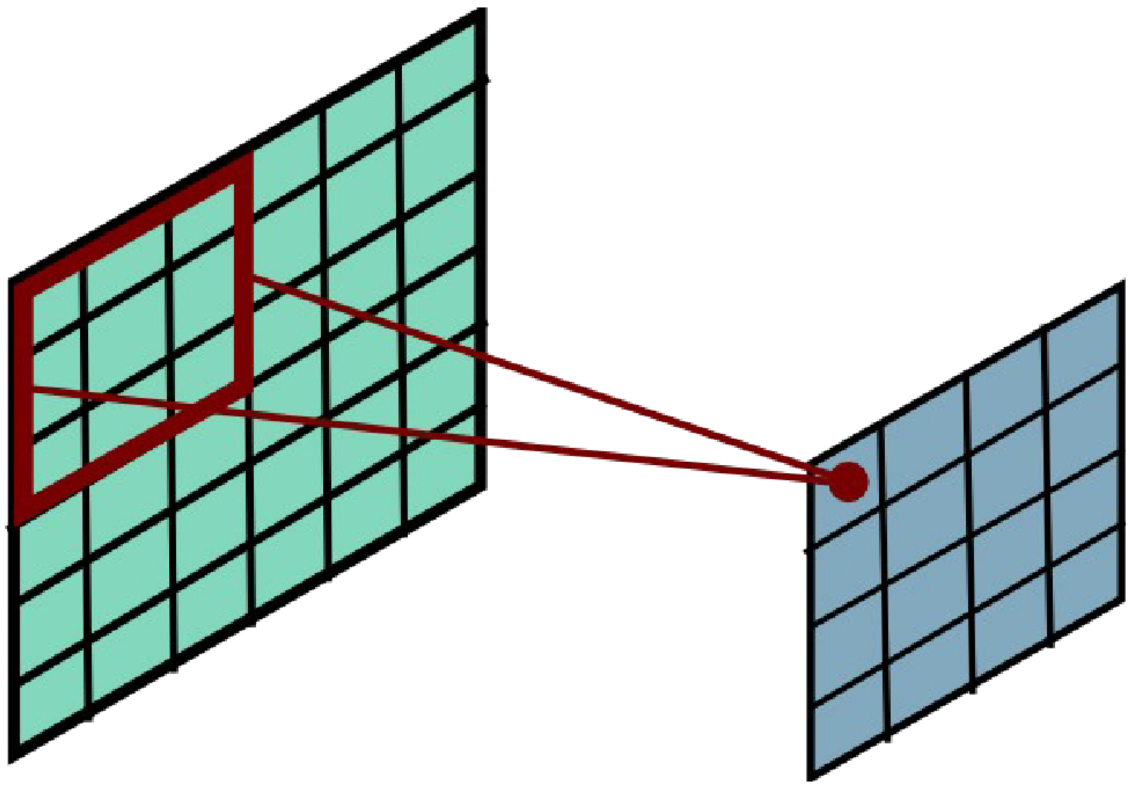


Image credit: A. Karpathy

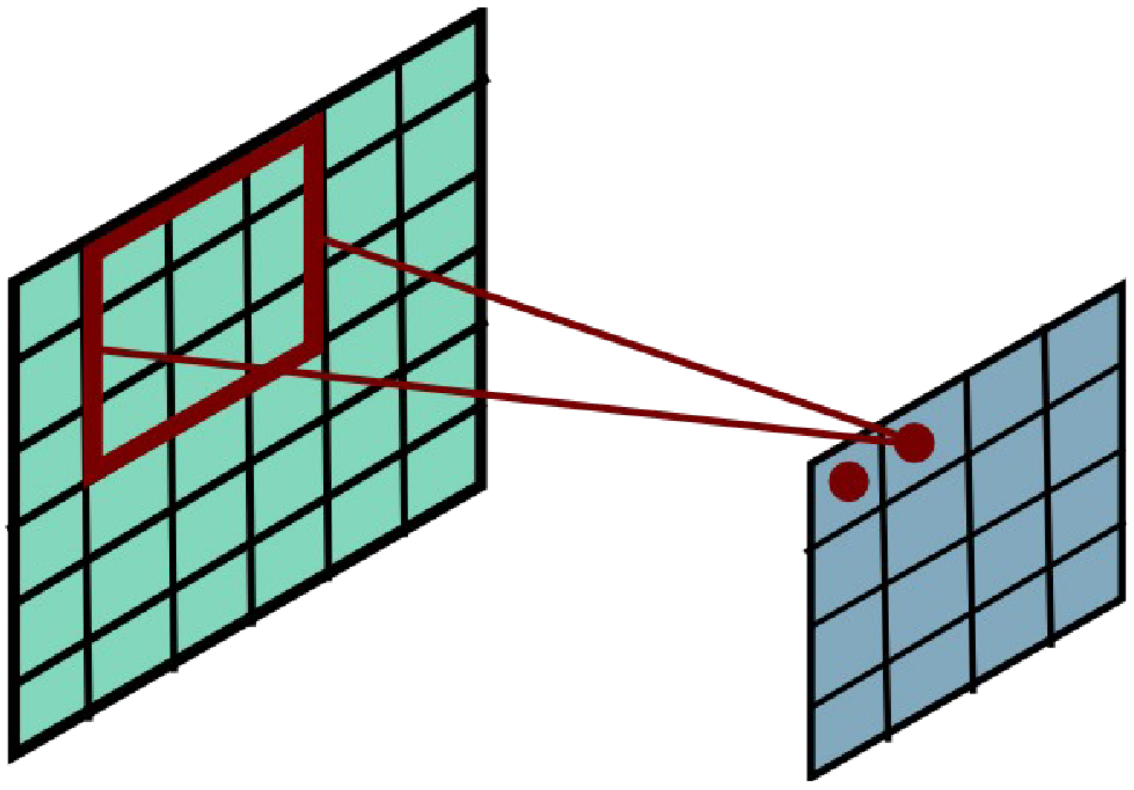


Share the same parameters across different locations (assuming input is stationary):

Convolutional Layer

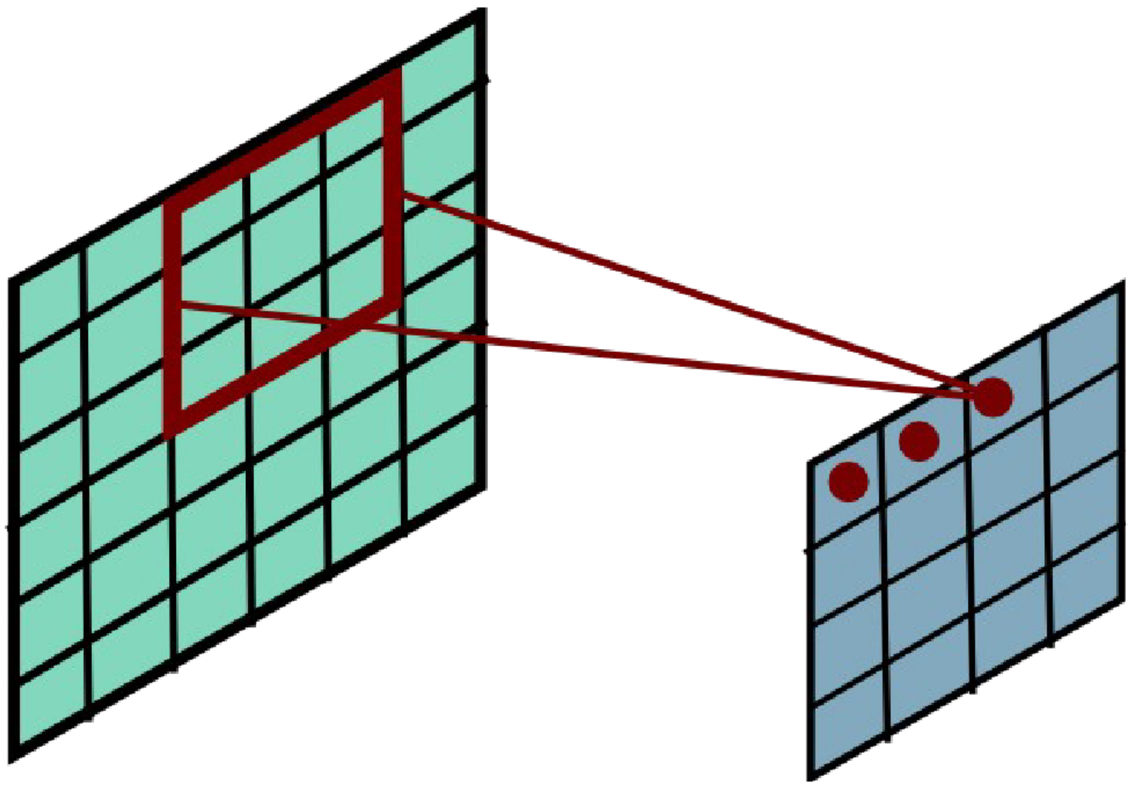


Convolutional Layer

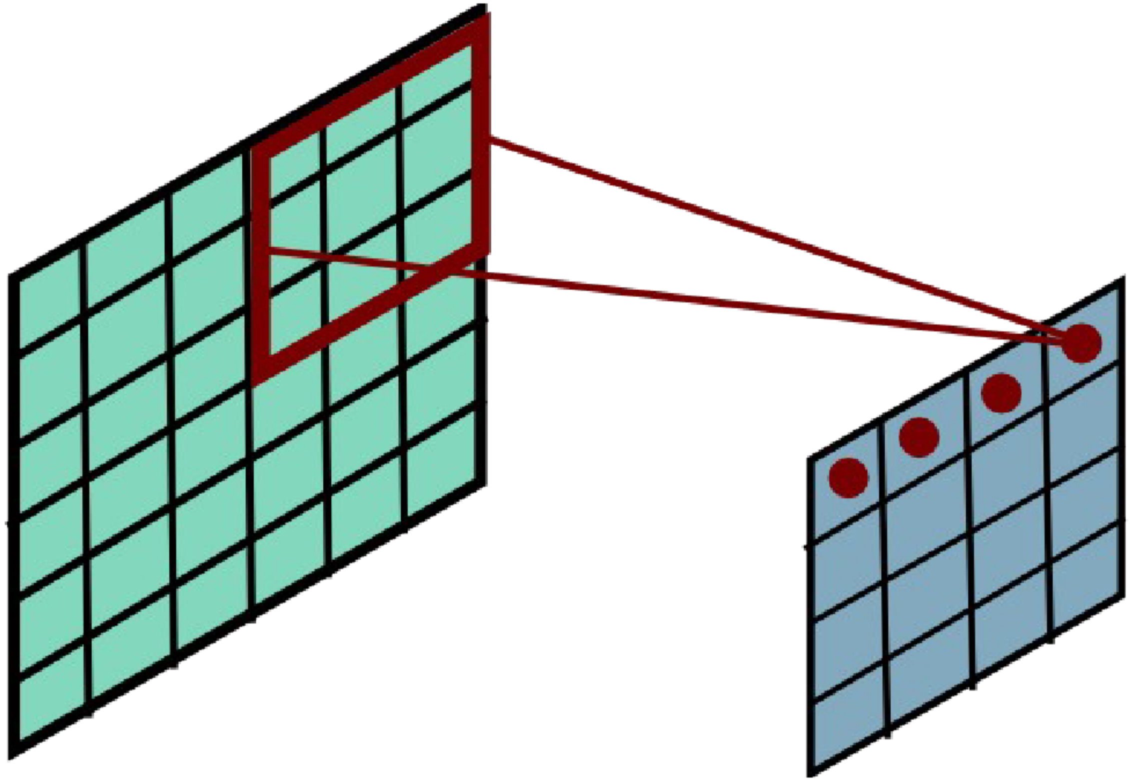


Ranzato

Convolutional Layer

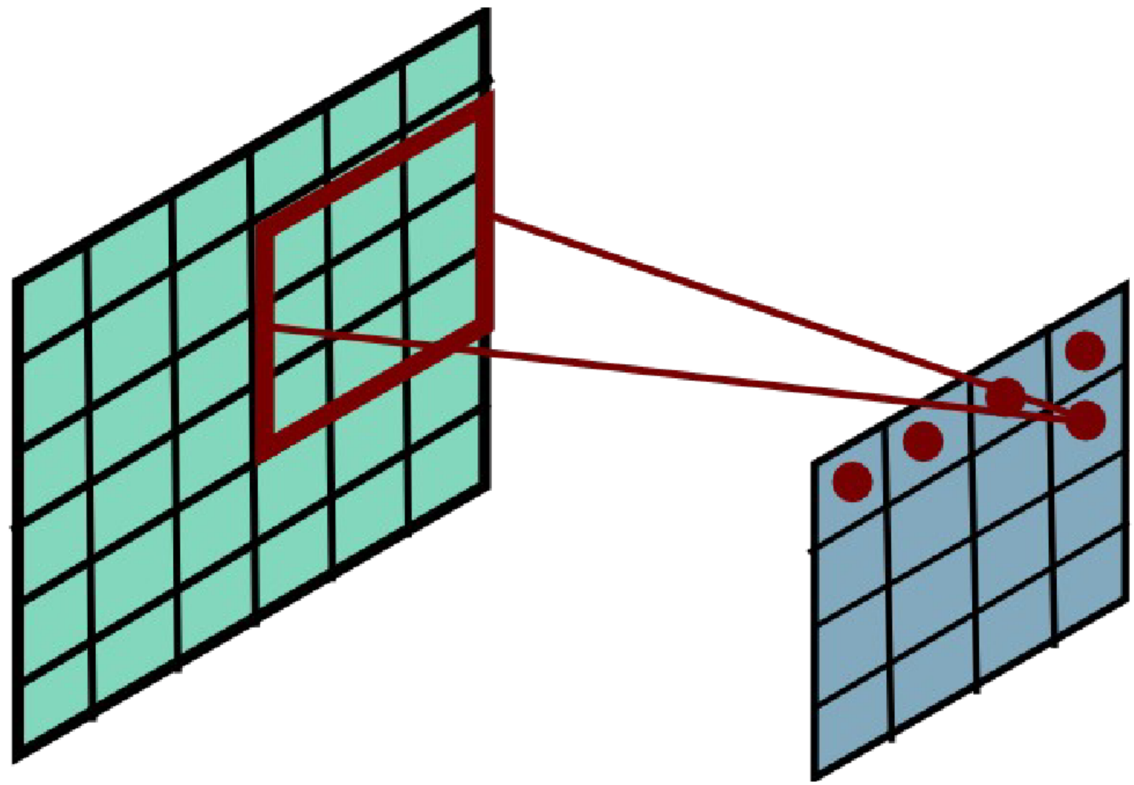


Convolutional Layer

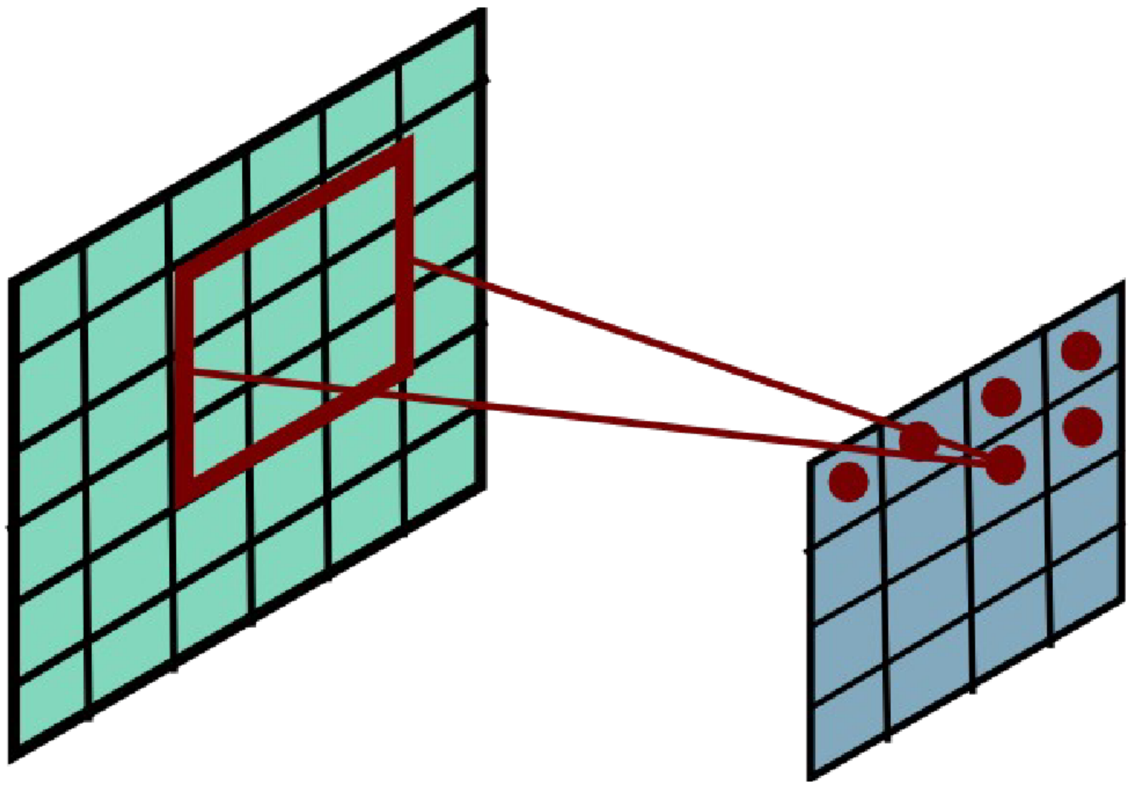


Ranzato

Convolutional Layer

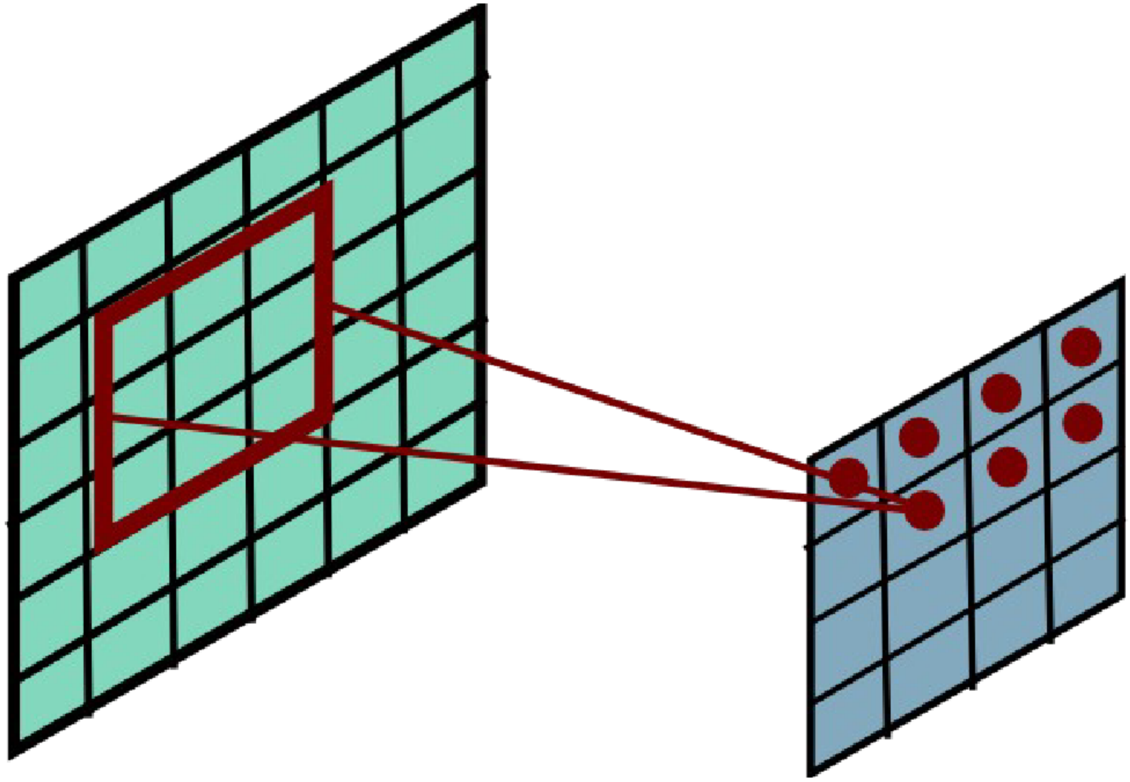


Convolutional Layer



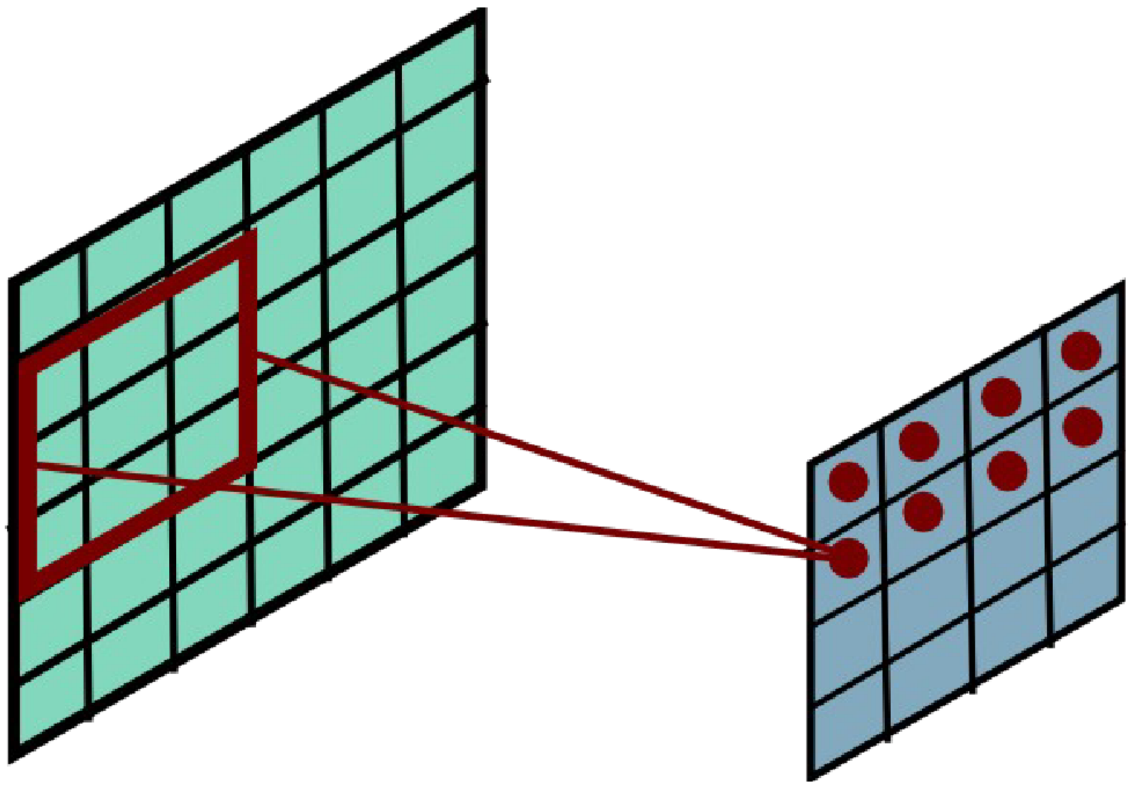
Ranzato

Convolutional Layer

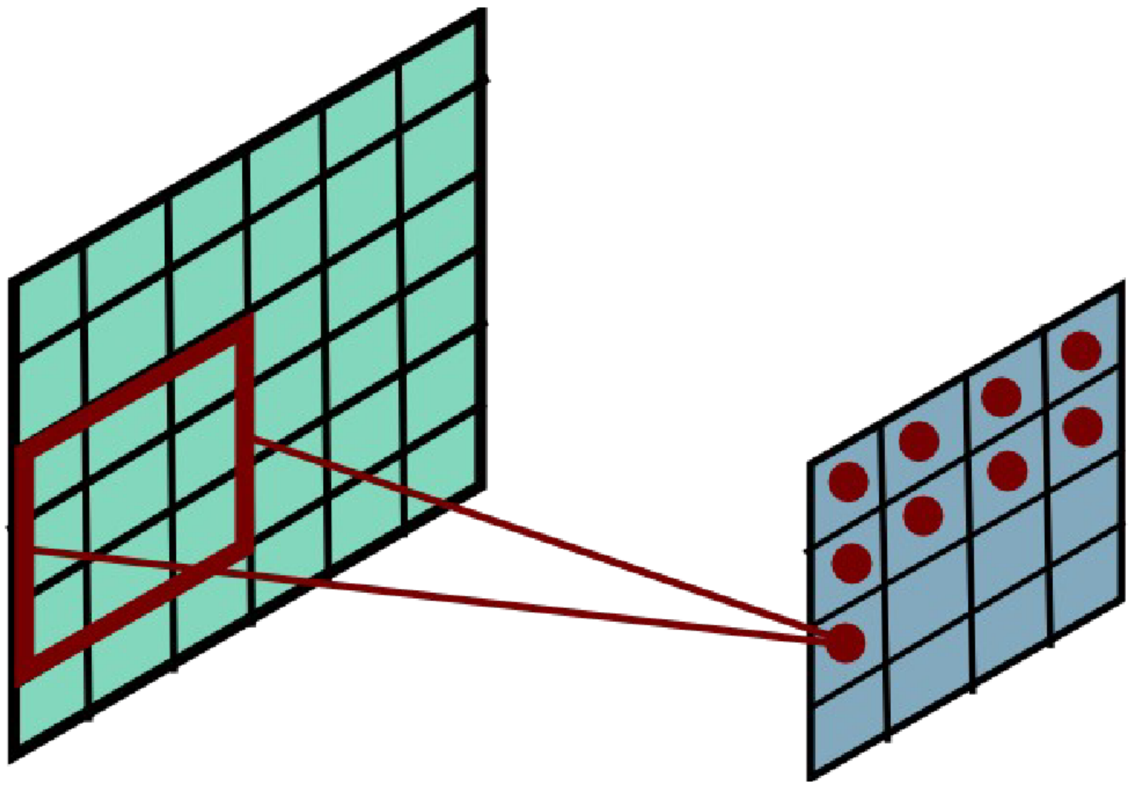


Ranzato

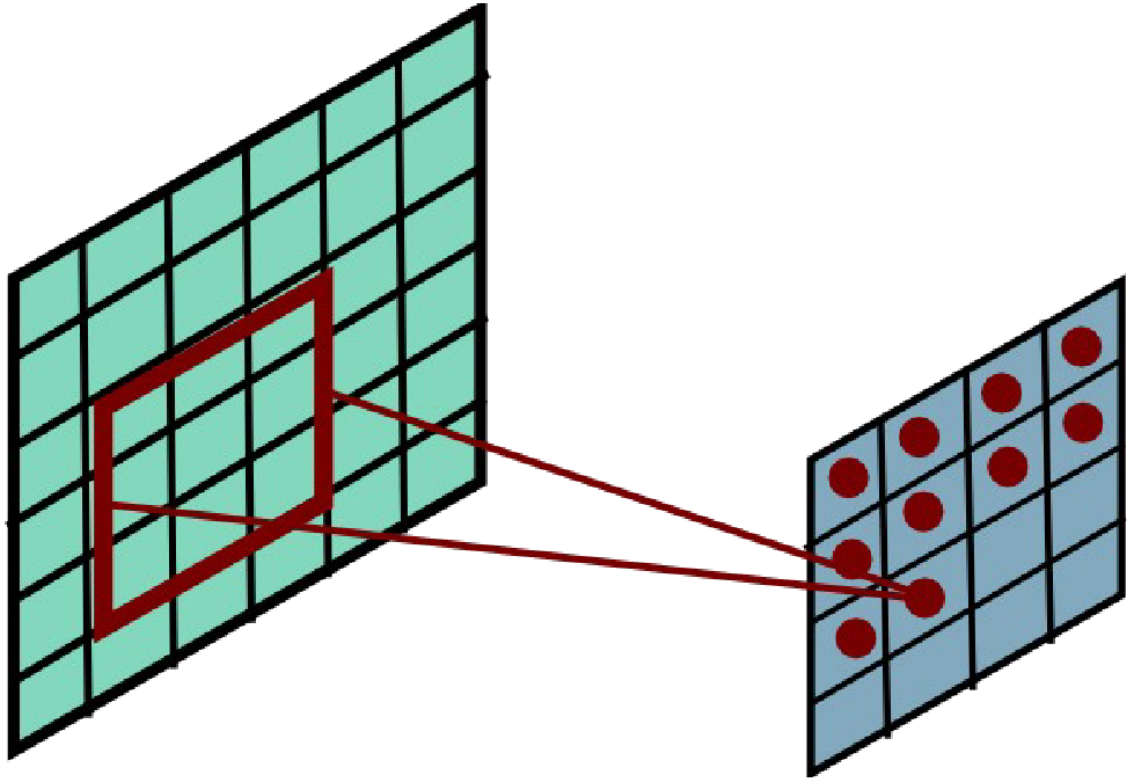
Convolutional Layer



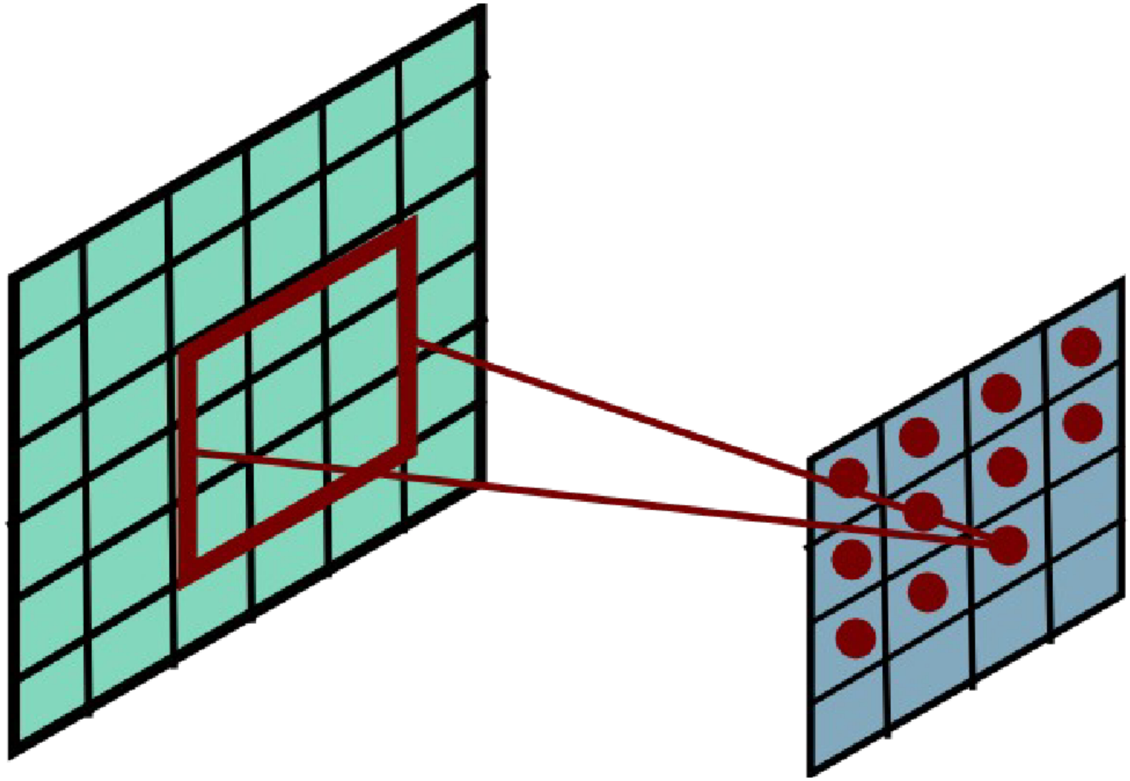
Convolutional Layer



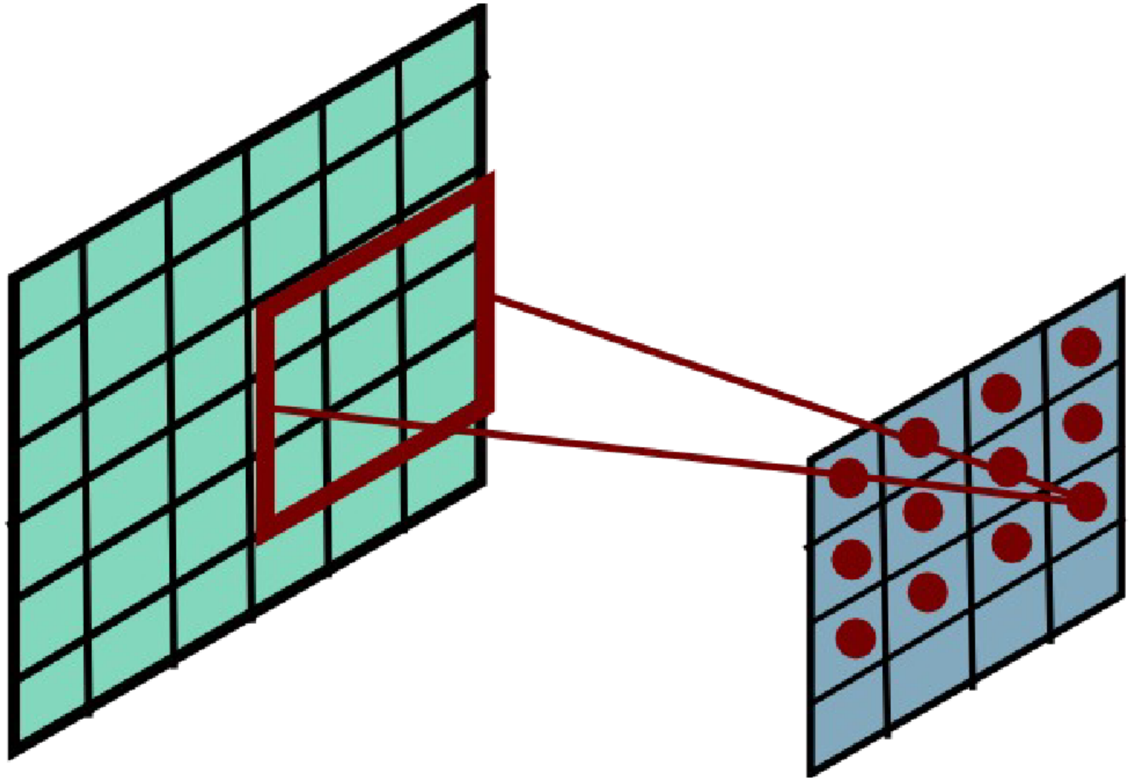
Convolutional Layer



Convolutional Layer

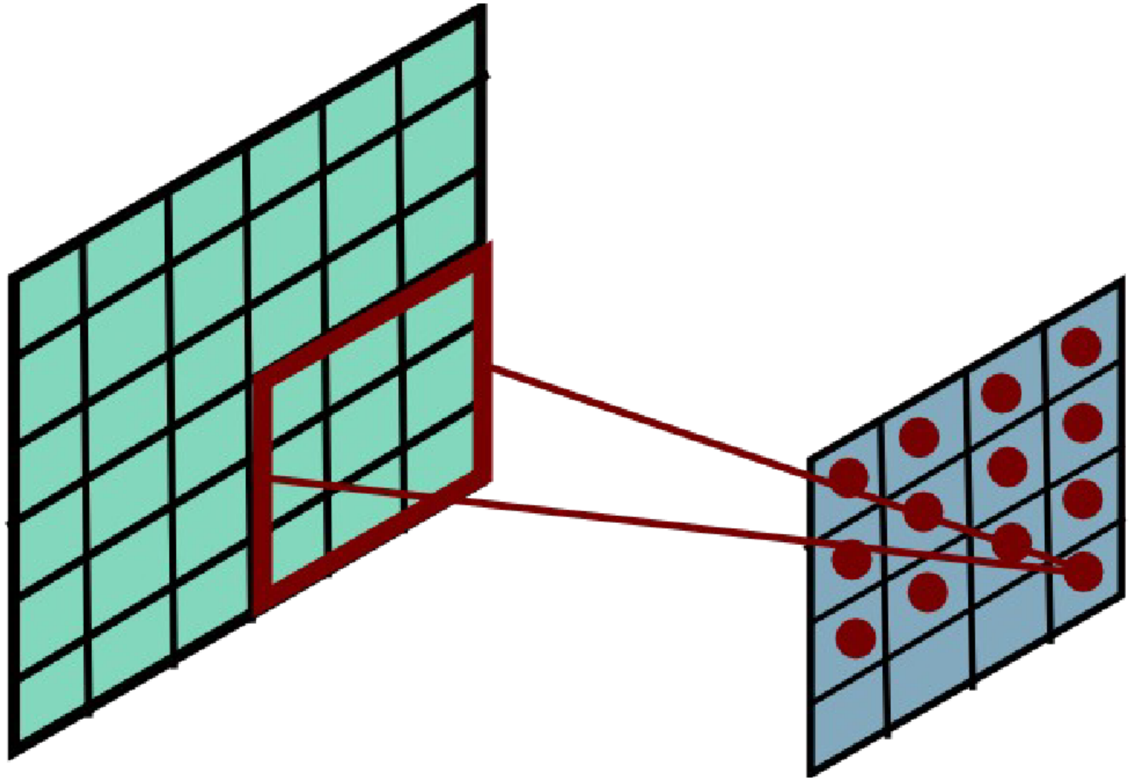


Convolutional Layer



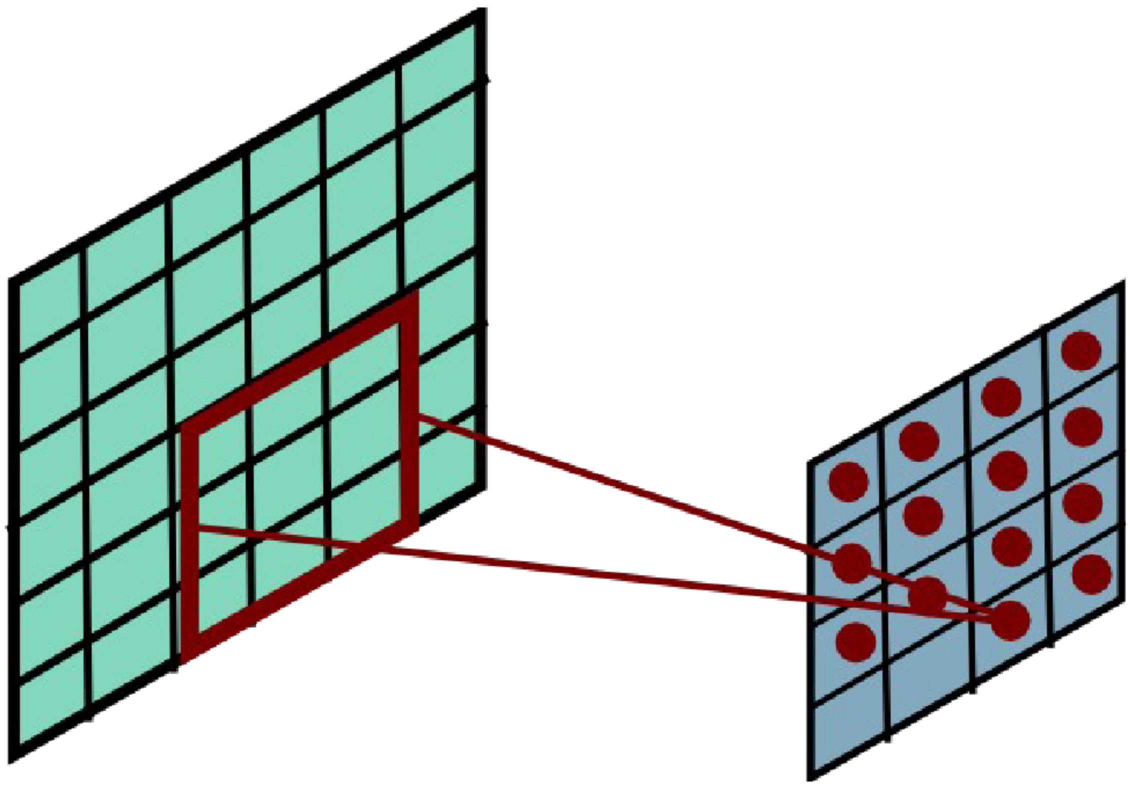
Ranzato

Convolutional Layer

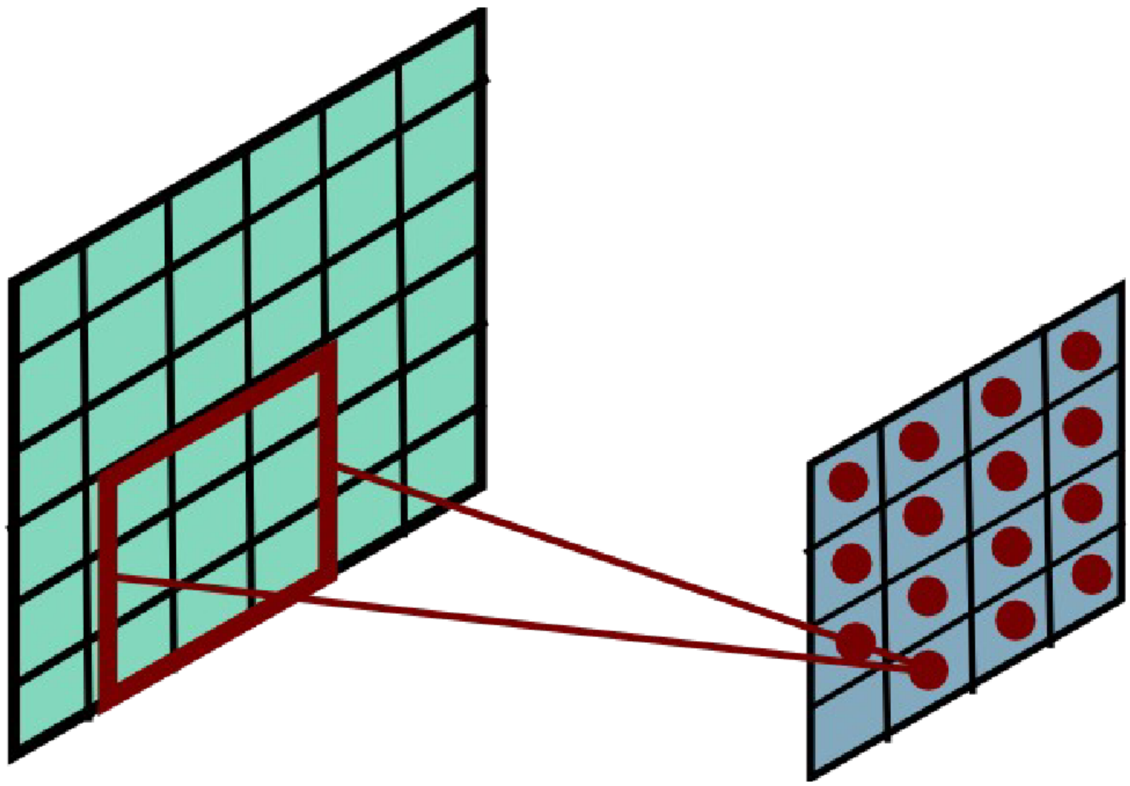


Ranzato

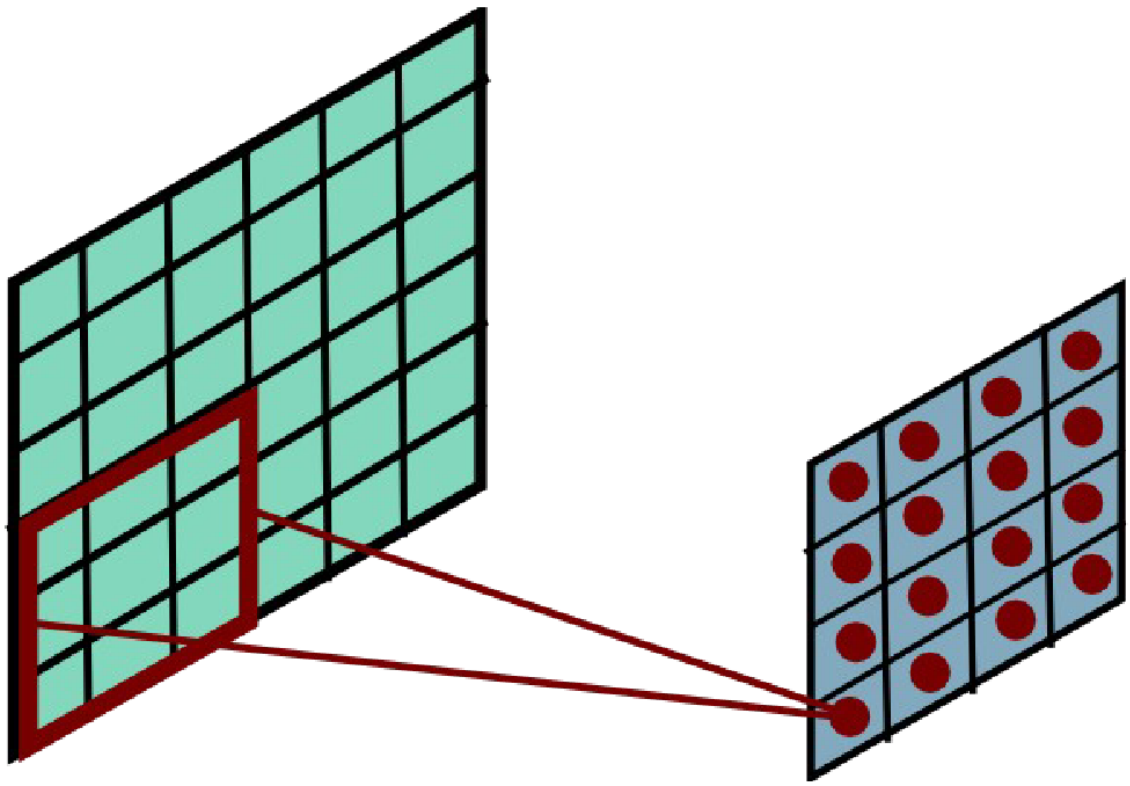
Convolutional Layer



Convolutional Layer

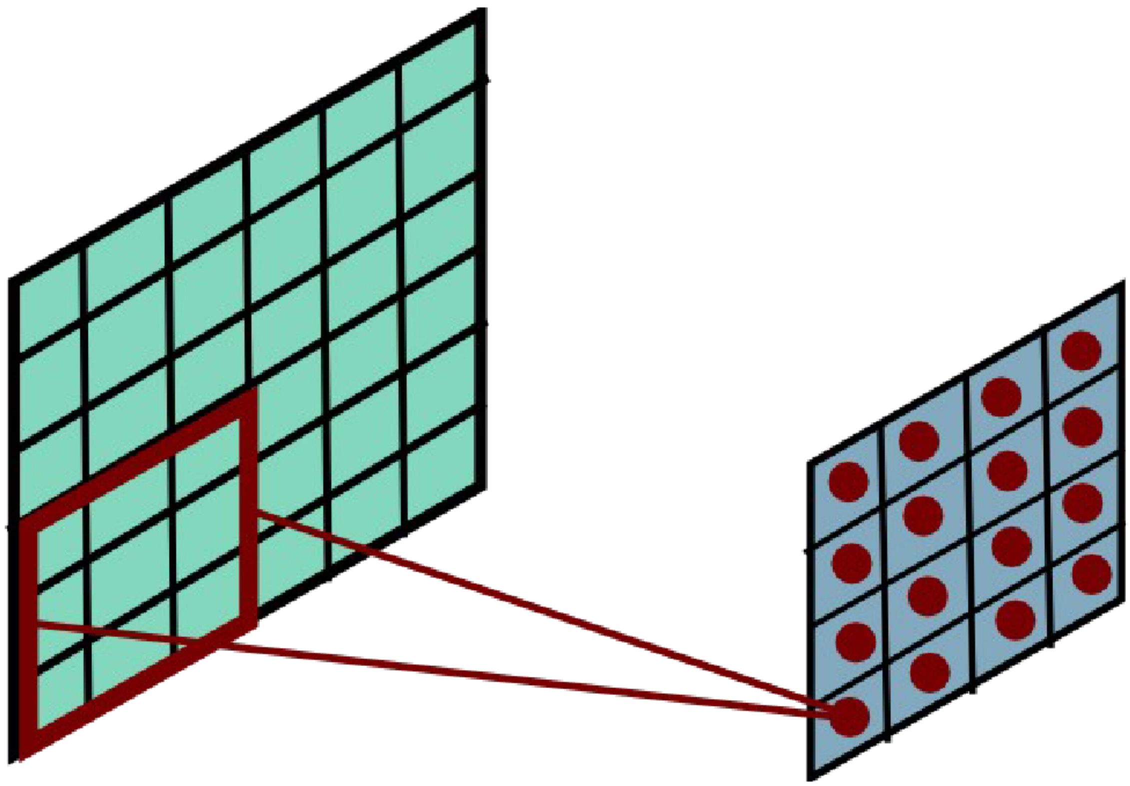


Convolutional Layer

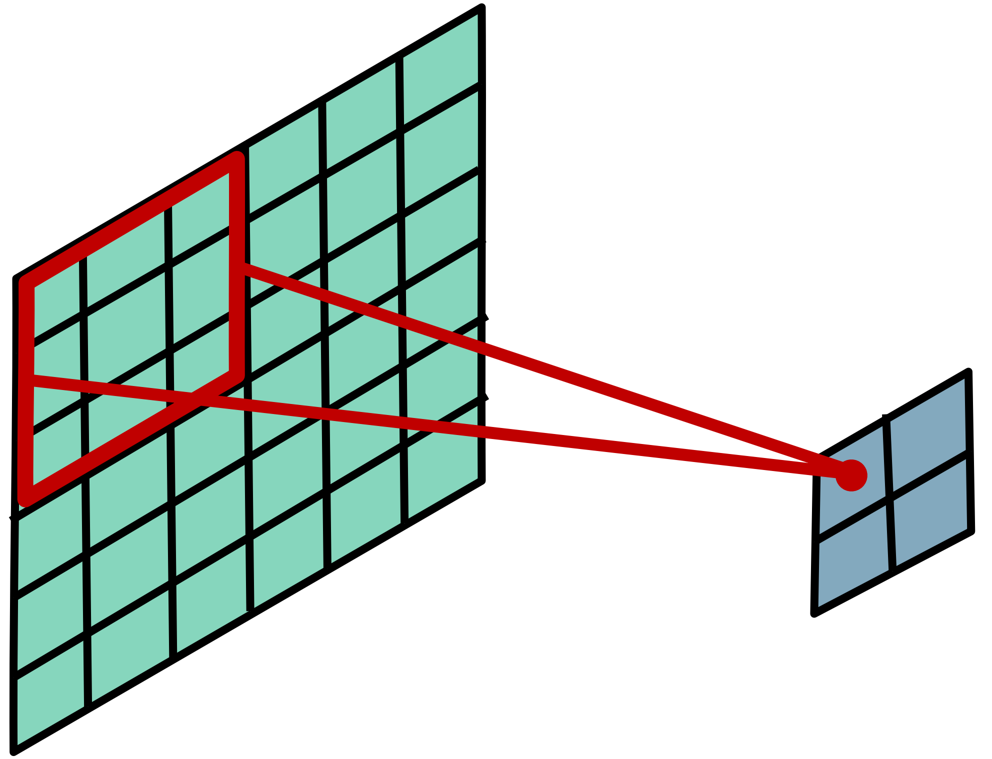


Ranzato

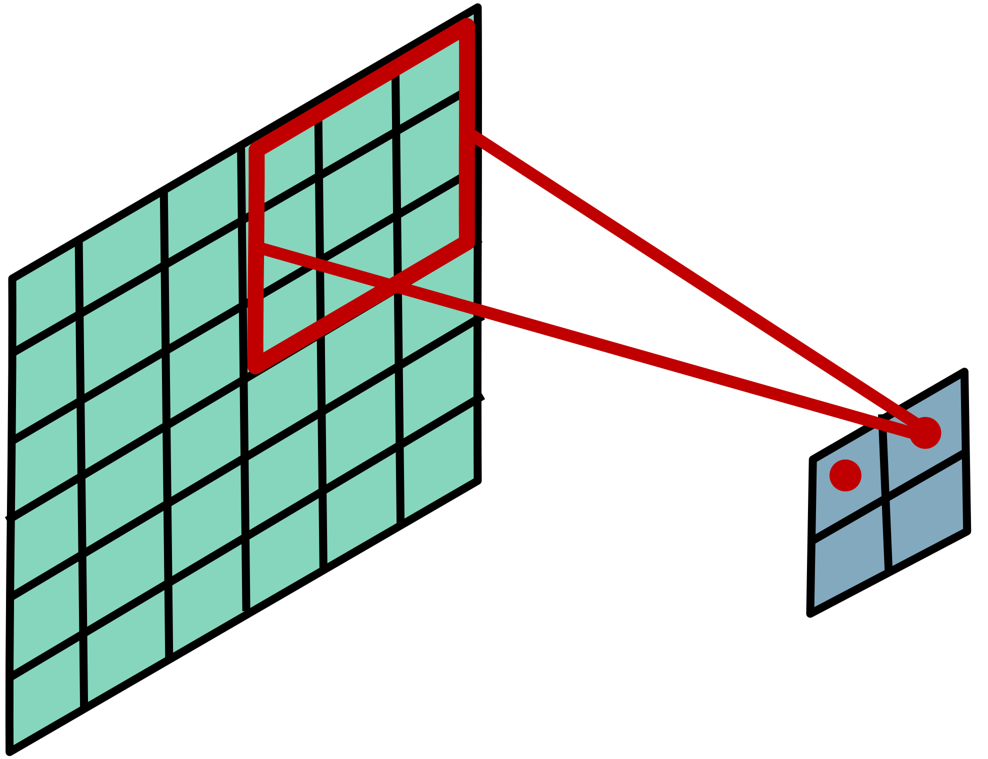
Stride = 3



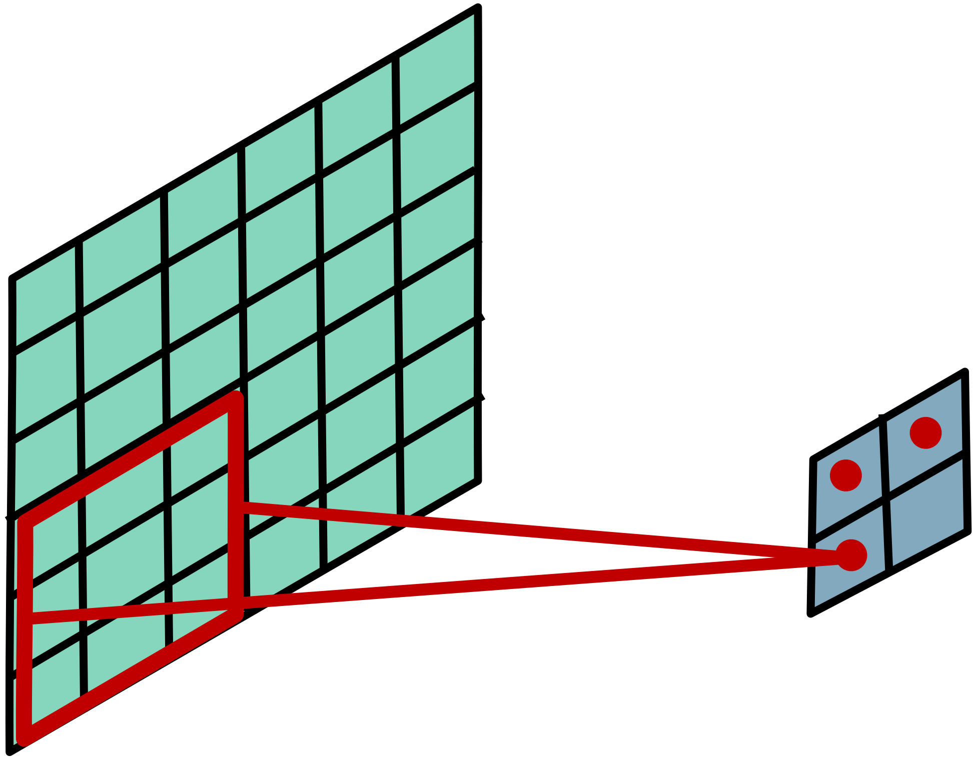
Stride = 3



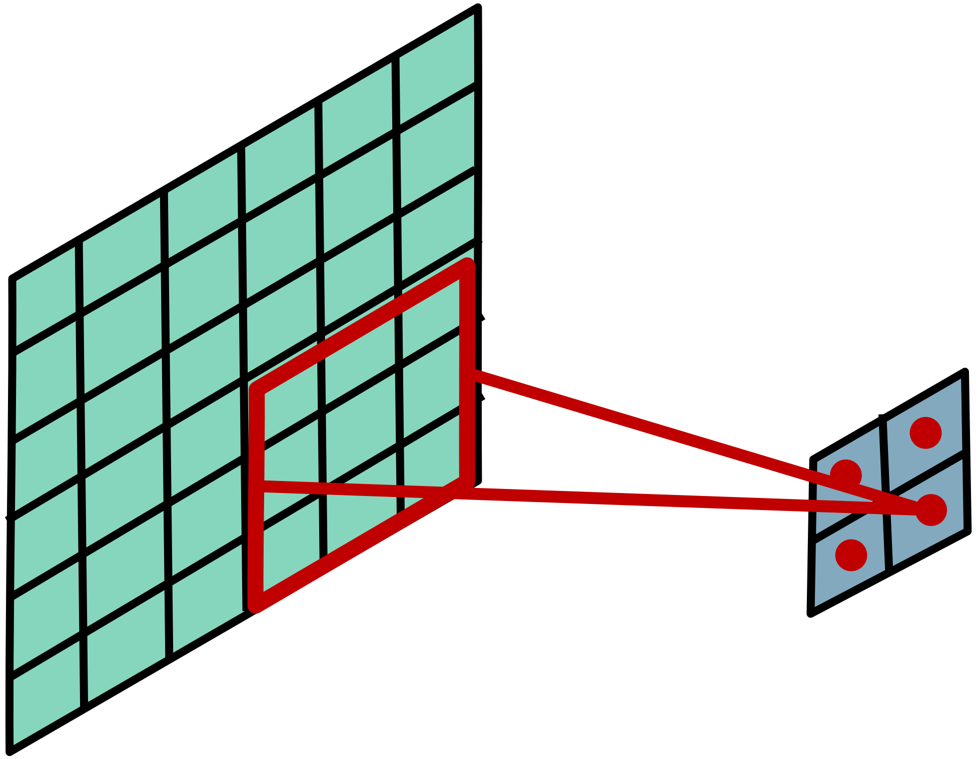
Stride = 3



Stride = 3

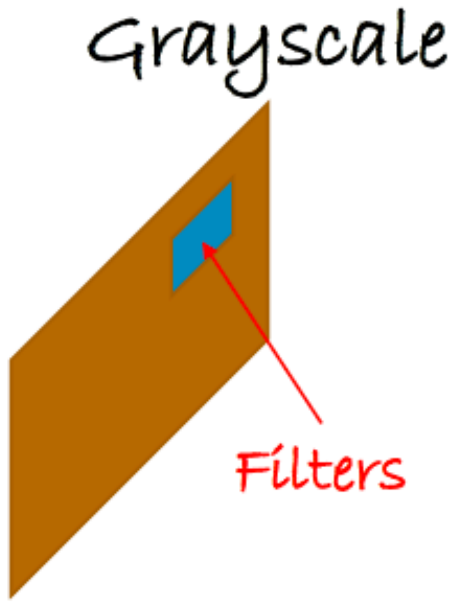


Stride = 3



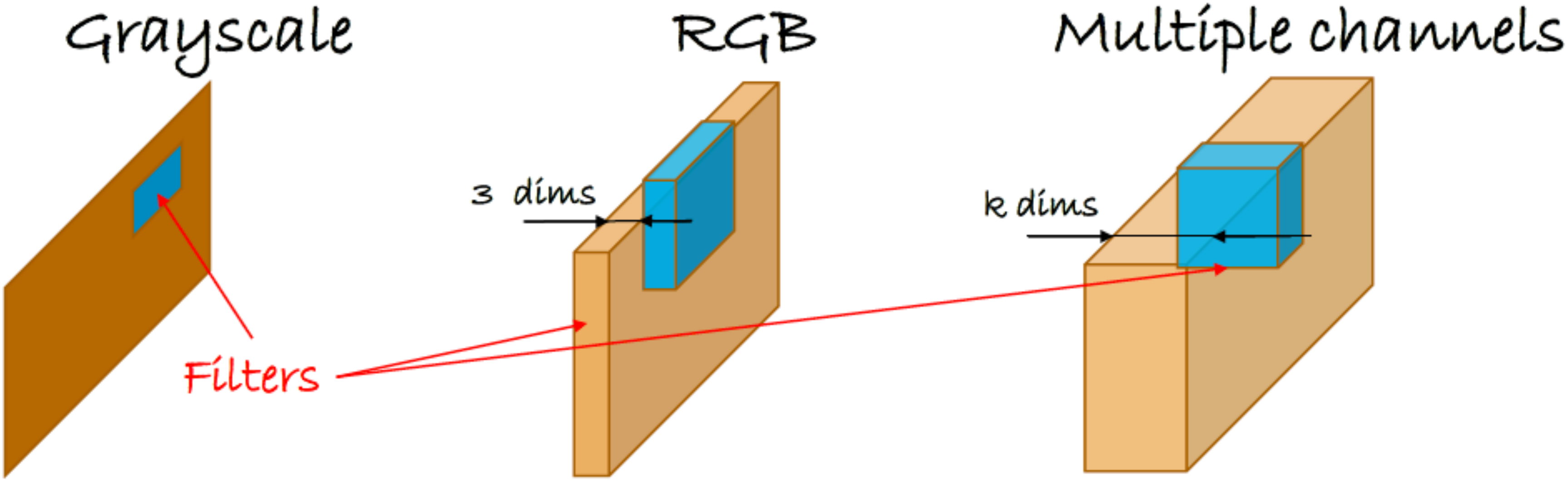
2D spatial filters

- If images are 2-D, parameters should also be organized in 2-D
  - That way they can learn the local correlations between input variables
  - That way they can "exploit" the spatial nature of images

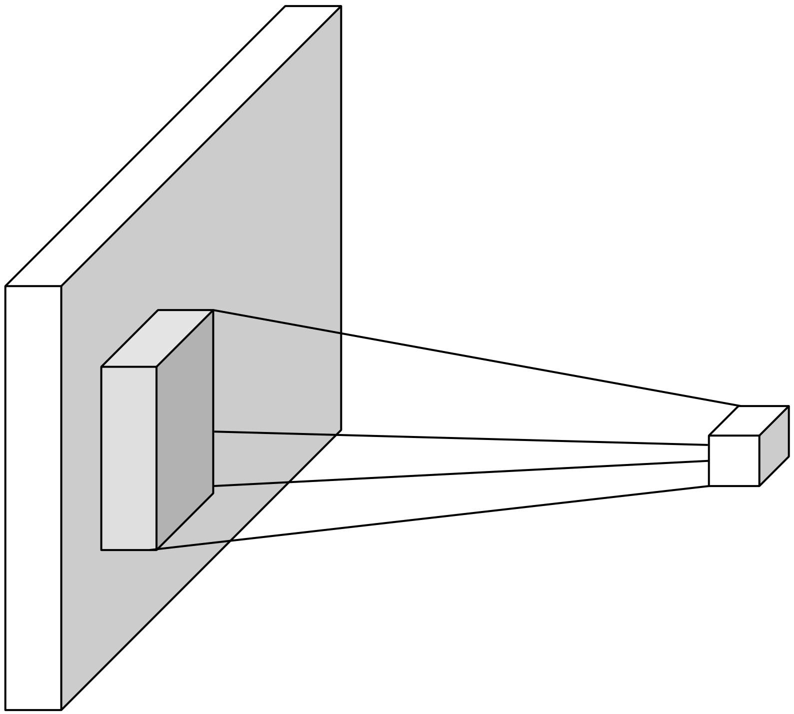


k-D spatial filters

- Similarly, if images are k-D, parameters should also be k-D



Dimensions of convolution

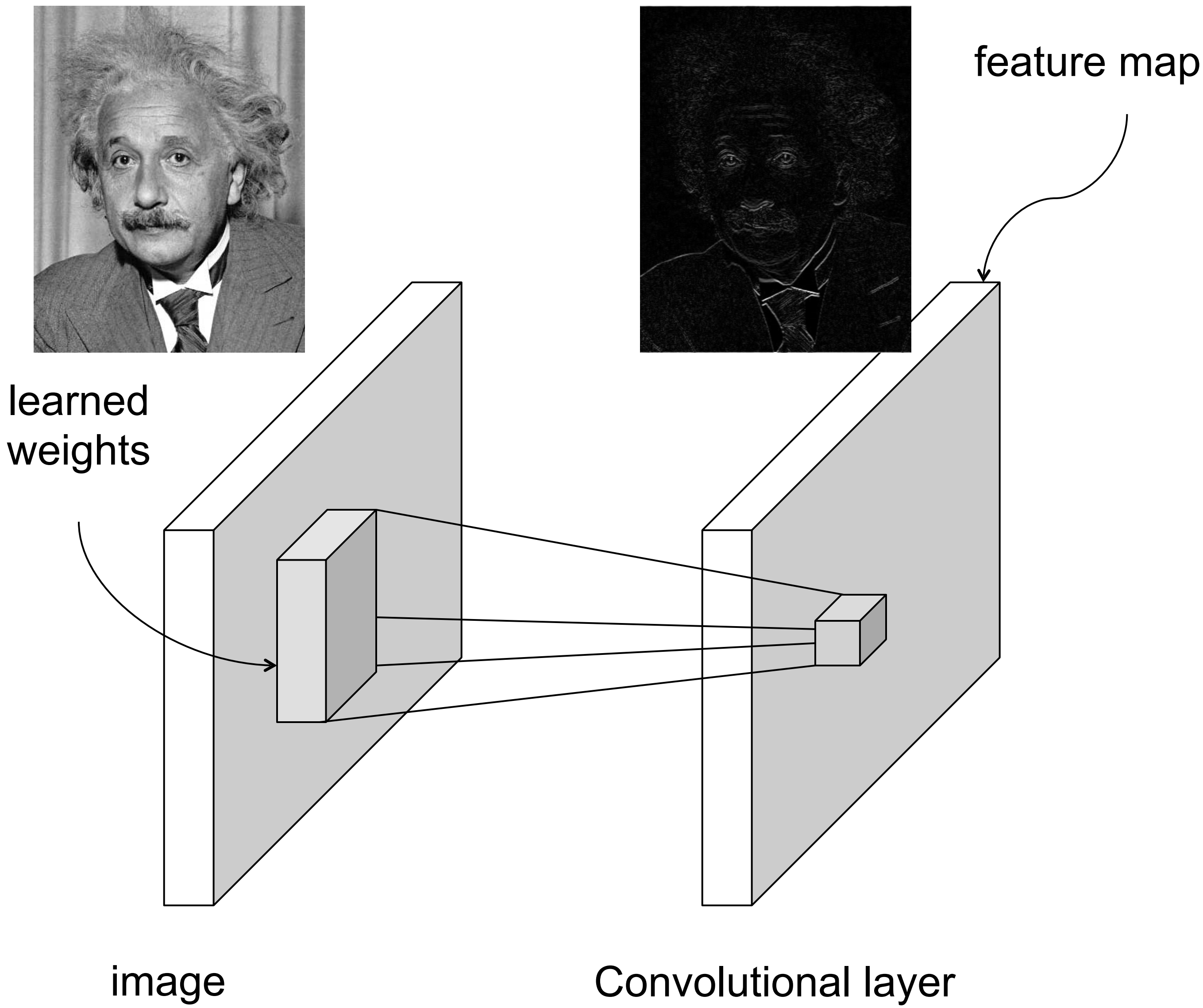


image

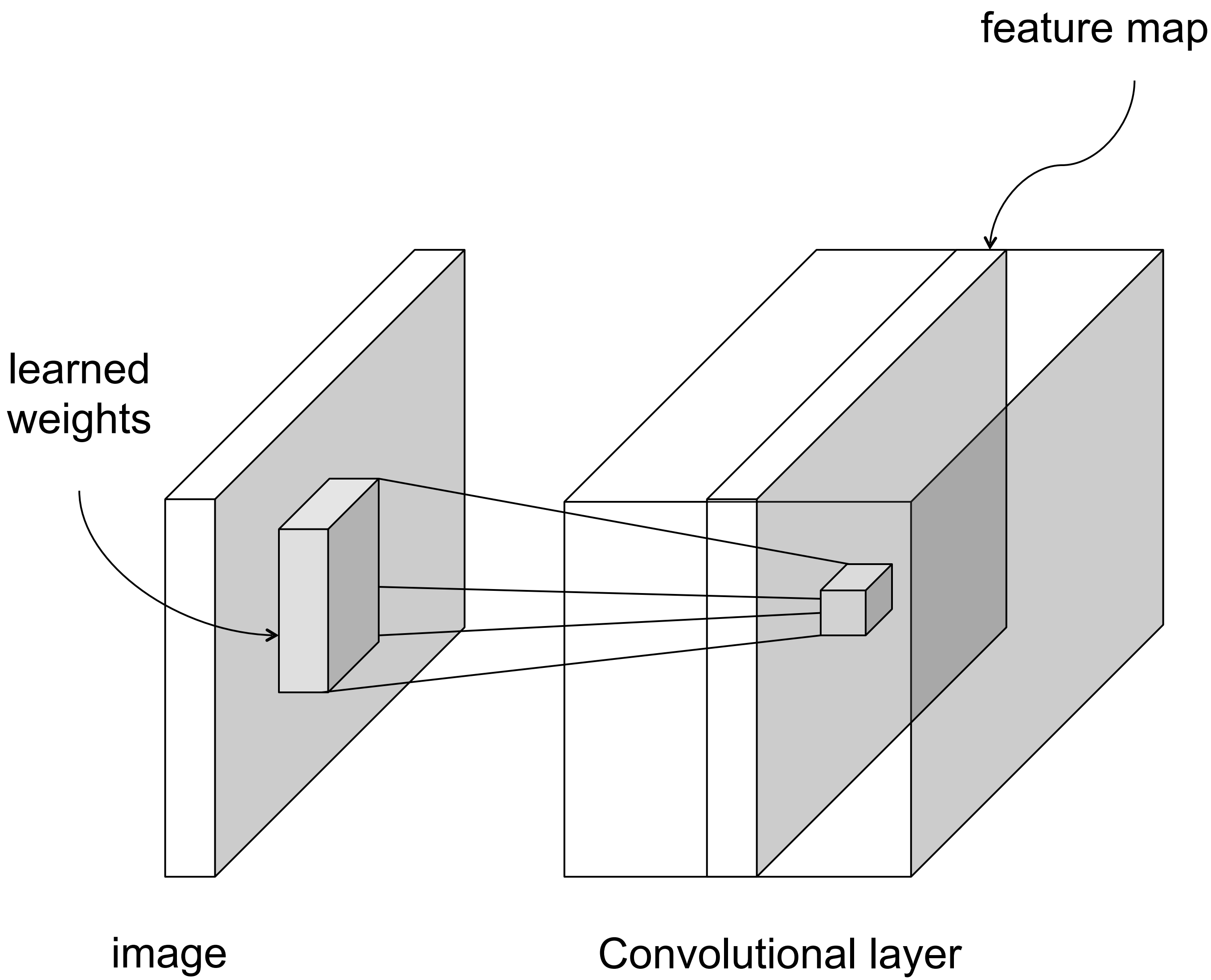
Convolutional layer

Slide: Lazebnik

Dimensions of convolution

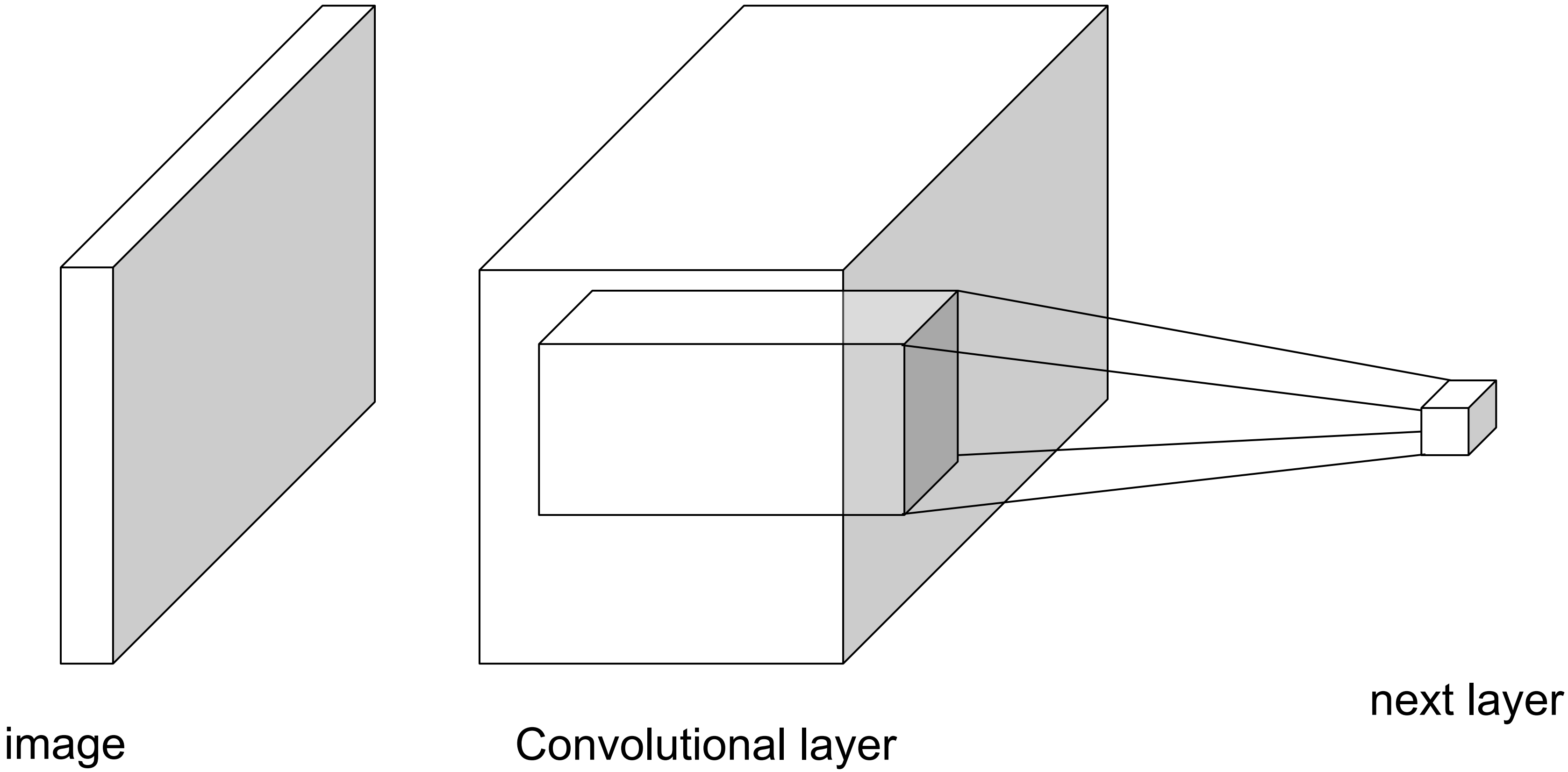


Dimensions of convolution



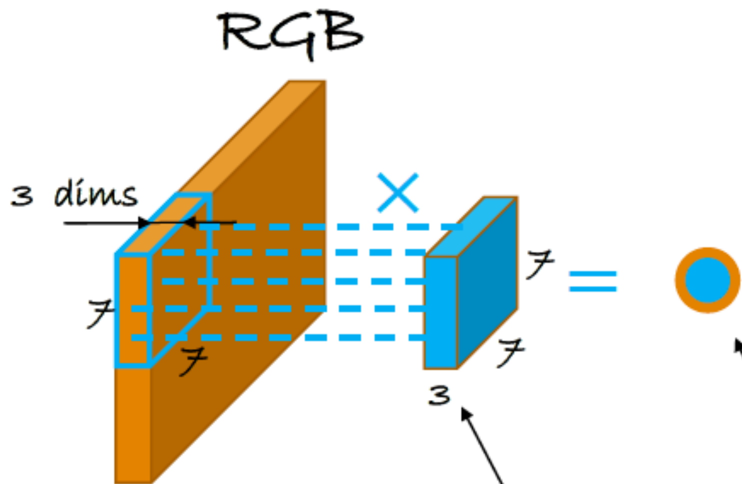
Slide: Lazebnik

Dimensions of convolution



Slide: Lazebnik

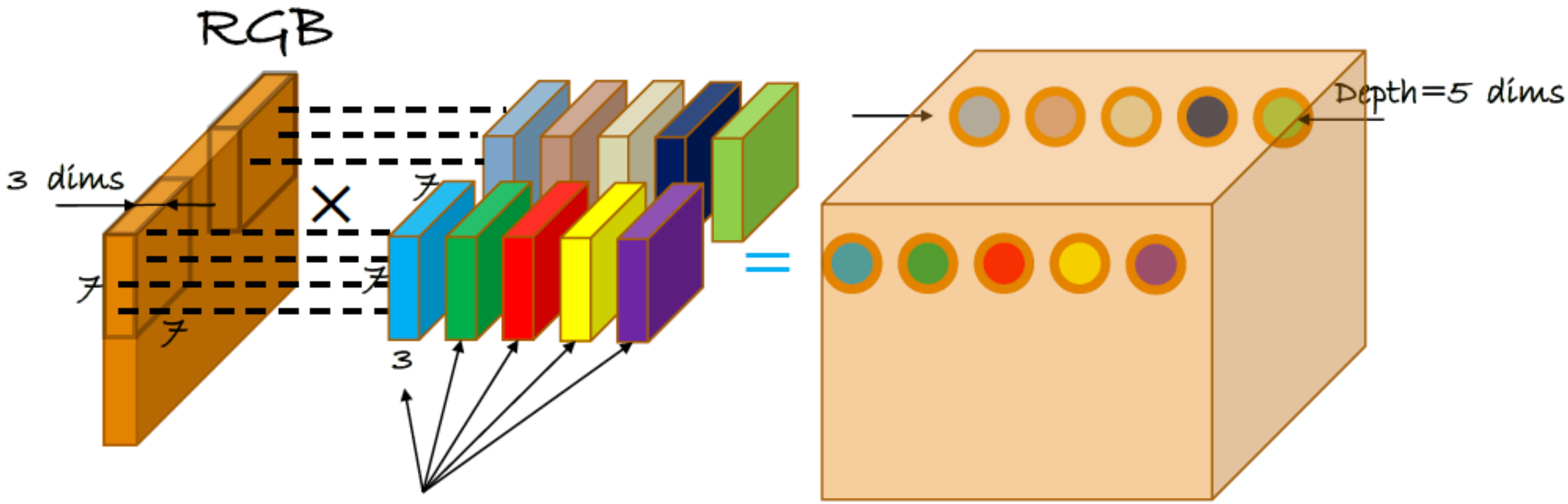
Number of weights



How many weights for this neuron?

$$7 \cdot 7 \cdot 3 = 147$$

Number of weights

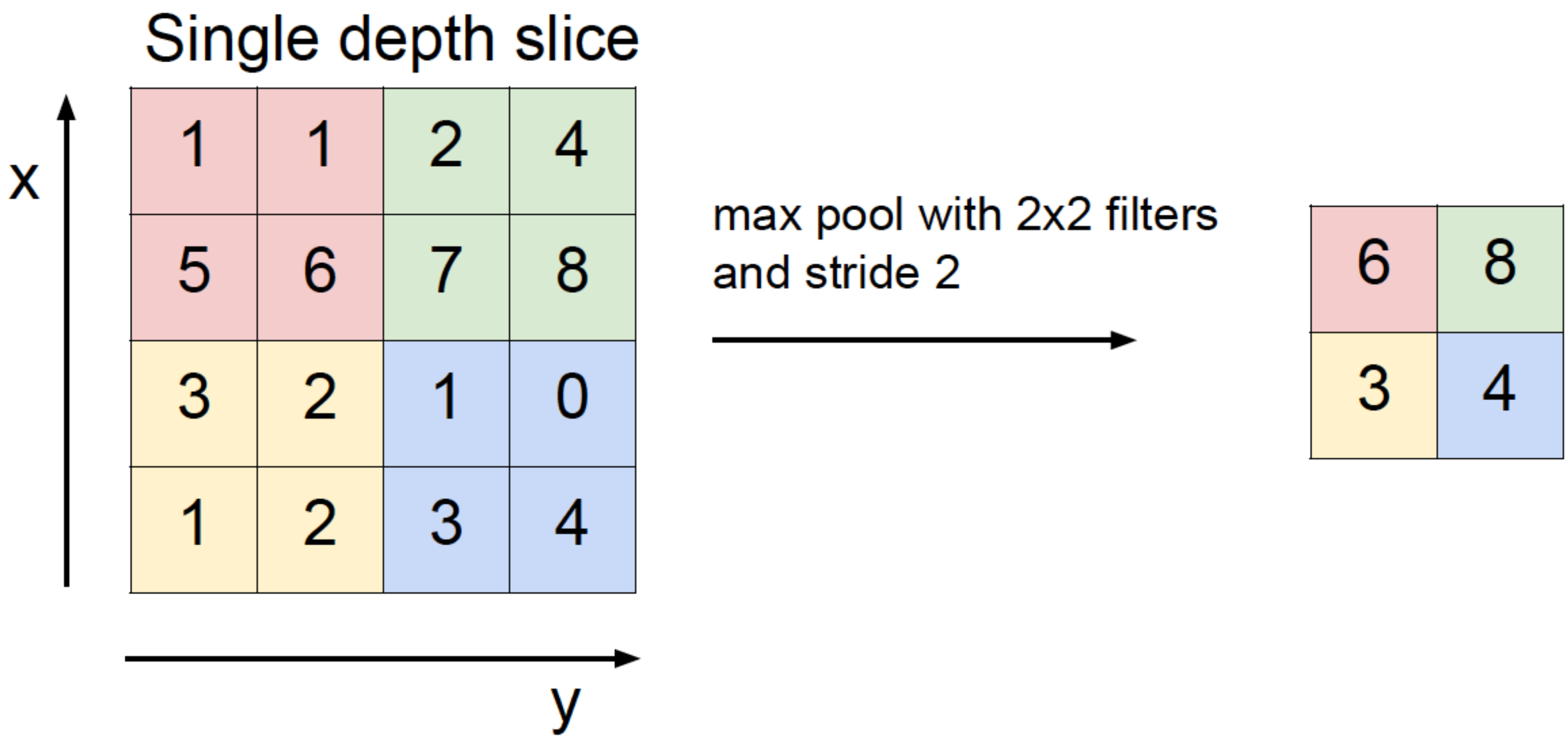


How many weights for these 5 neurons?

$$5 \cdot 7 \cdot 7 \cdot 3 = 735$$

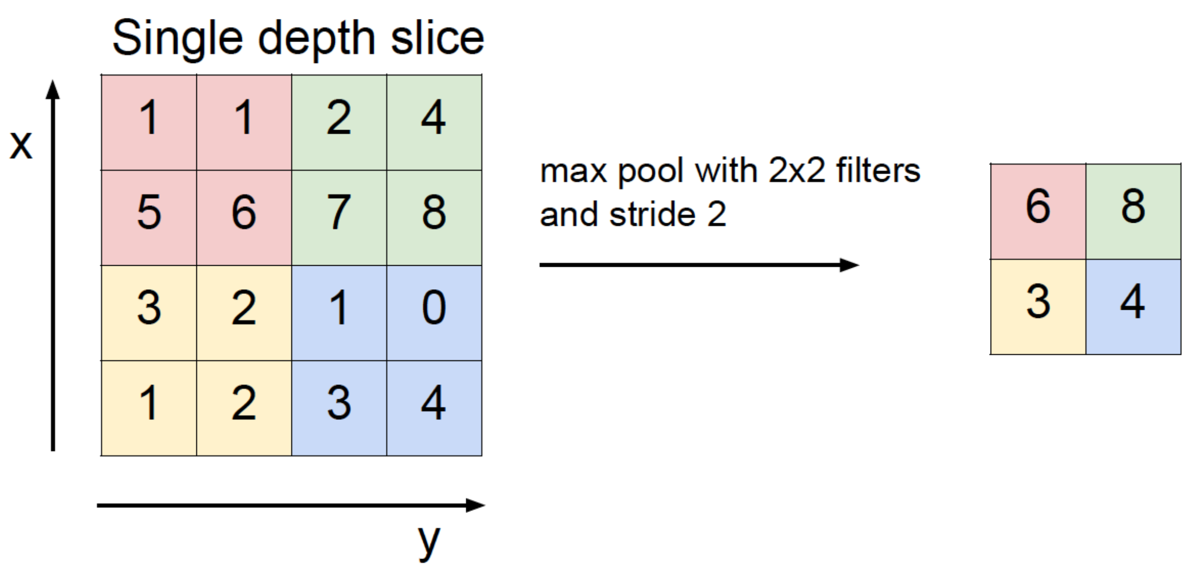
Pooling: Downsample feature maps

- Aggregate multiple values into a single value

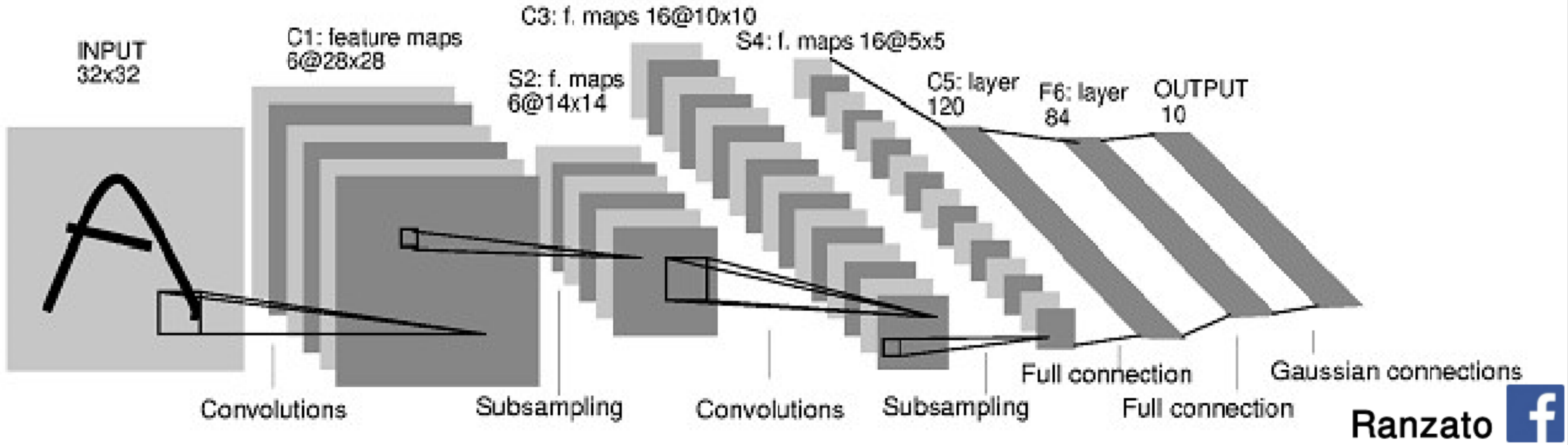


Pooling: Downsample feature maps

- Aggregate multiple values into a single value
- Invariance to small transformations
  - Keep only most important information for next layer
- Reduces the size of the next layer
  - Fewer parameters, faster computations
- Observe larger receptive field in next layer
  - Hierarchically extract more abstract features

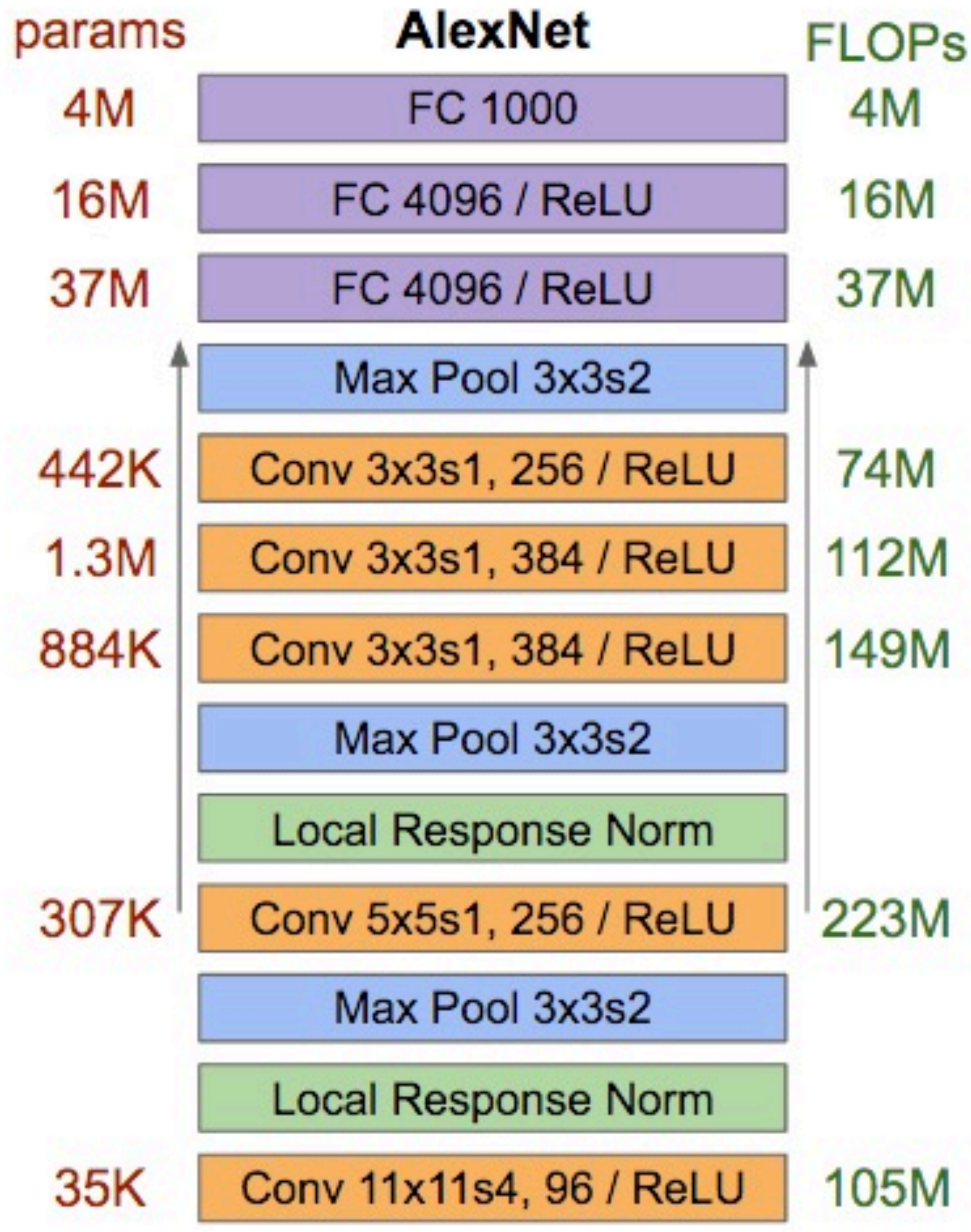


Yann LeCun's MNIST CNN architecture



AlexNet for ImageNet

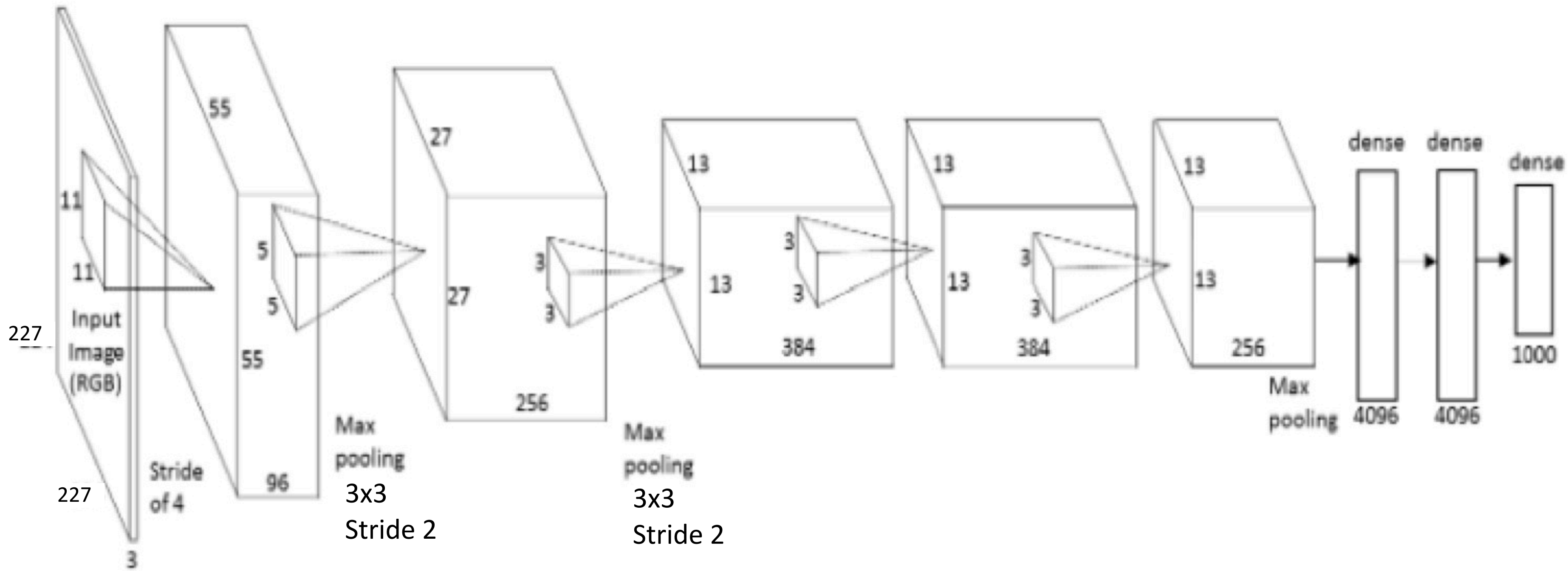
Layers
- Kernel sizes
- Strides
- # channels
- # kernels
- Max pooling



[Krizhevsky et al. 2012]

AlexNet diagram (simplified)

Input size
227 x 227 x 3



Conv 1	**Conv 2**	**Conv 3**	**Conv 4**	**Conv 4**
11 x 11 x 3	5 x 5 x 48	3 x 3 x 256	3 x 3 x 192	3 x 3 x 192
Stride 4	Stride 1	Stride 1	Stride 1	Stride 1
96 filters	256 filters	384 filters	384 filters	256 filters

Convolutional Neural Networks

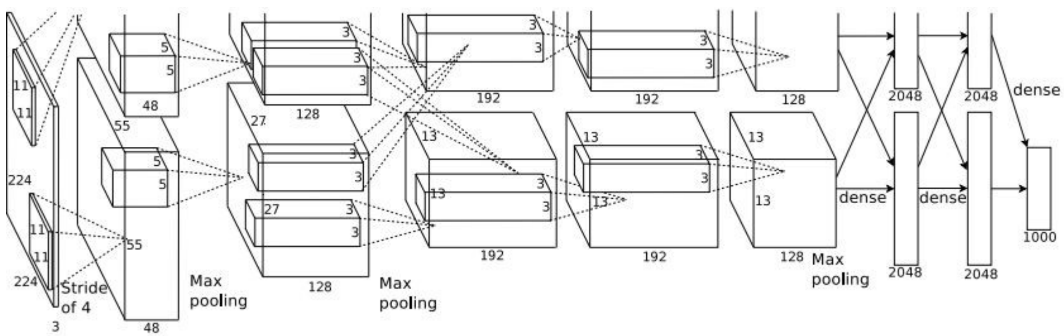
- Question: Spatial structure?
  - Answer: Convolutional filters
- Question: Huge input dimensionalities?
  - Answer: Parameters are shared between filters
- Question: Local variances?
  - Answer: Pooling

[Slides credit: Efstratios Gavves]

What’s going on inside ConvNets?

This image is CC0 public domain

Input Image:
3 x 224 x 224

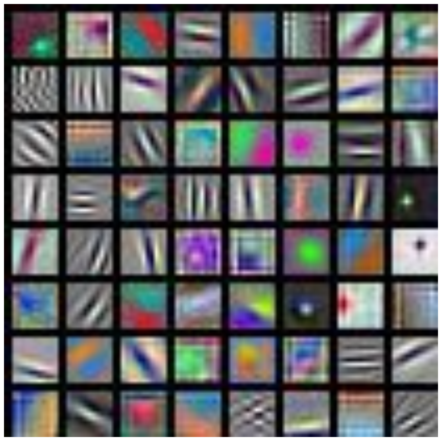


Class Scores:
1000 numbers

What are the intermediate features looking for?

Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012. Figure reproduced with permission.

First Layer: Visualize Filters

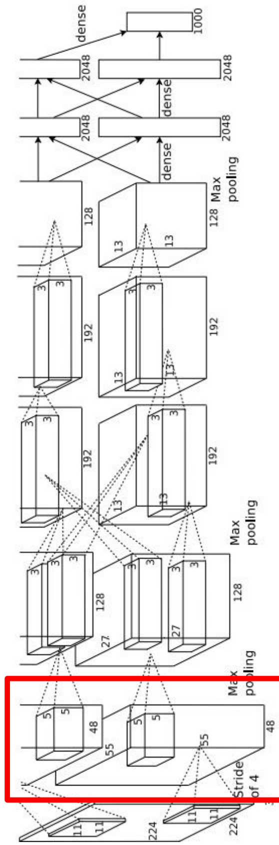


AlexNet: 64 x 3 x 11 x 11

ResNet-18: 64 x 3 x 7 x 7

ResNet-101: 64 x 3 x 7 x 7

DenseNet-121: 64 x 3 x 7 x 7



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014

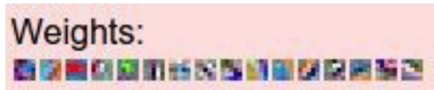
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016

Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Visualize the filters/kernels (raw weights)

layer 1 weights

16 x 3 x 7 x 7



layer 2 weights

20 x 16 x 7 x 7

We can visualize filters at higher layers, but not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

layer 3 weights

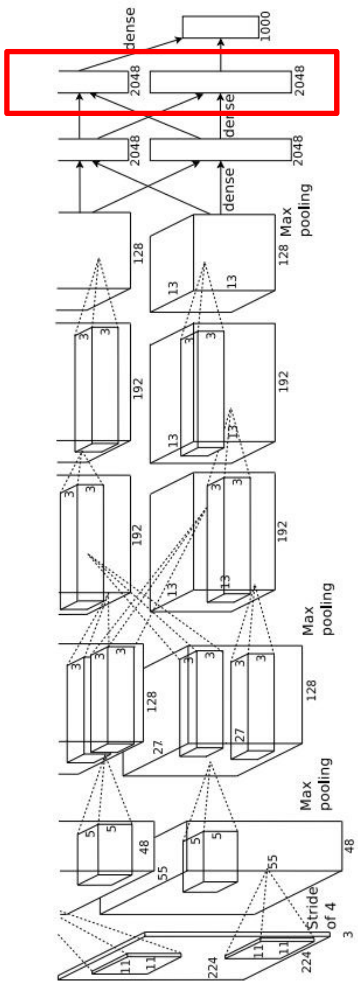
20 x 20 x 7 x 7

Last Layer

FC7 layer

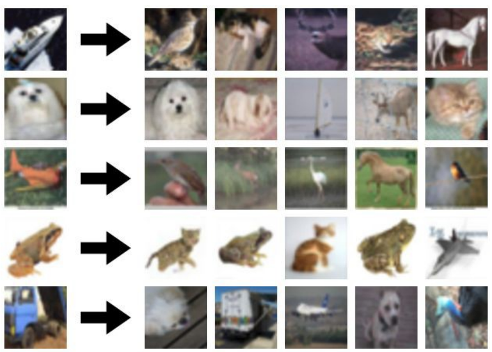
4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors



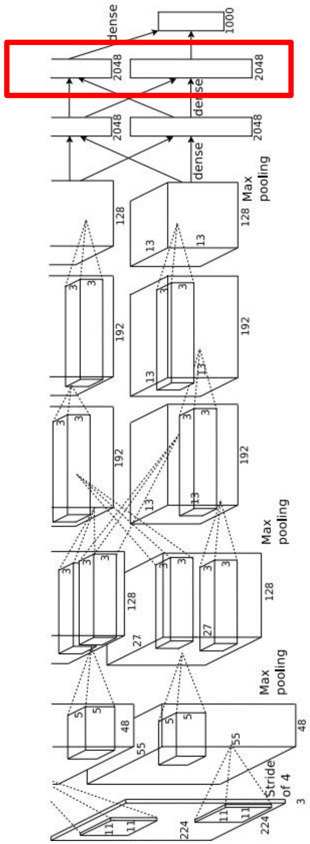
Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space



Test image L2 Nearest neighbors in feature space

4096-dim vector



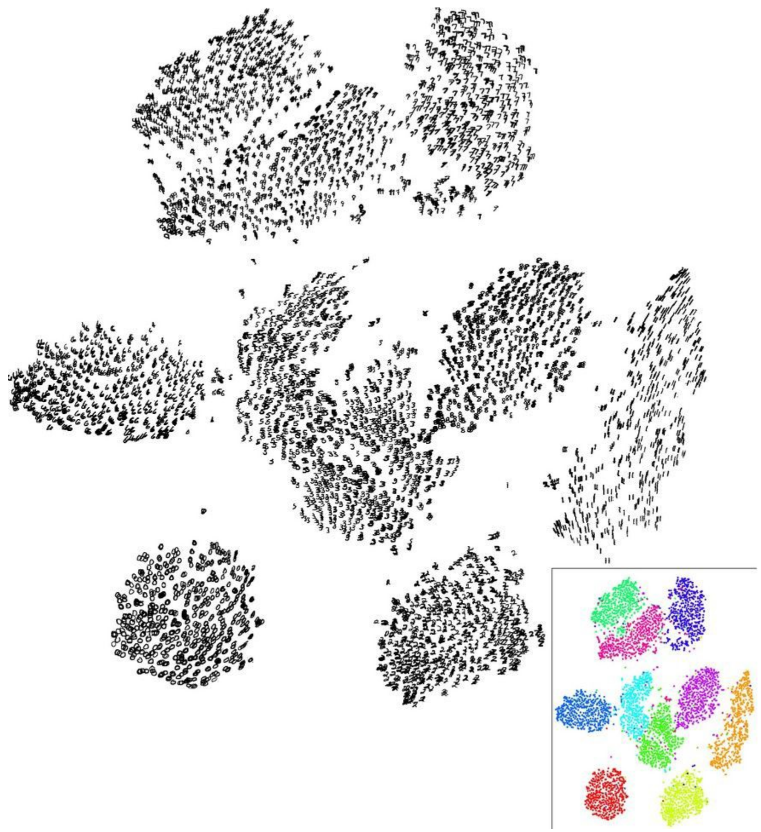
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012. Figures reproduced with permission.

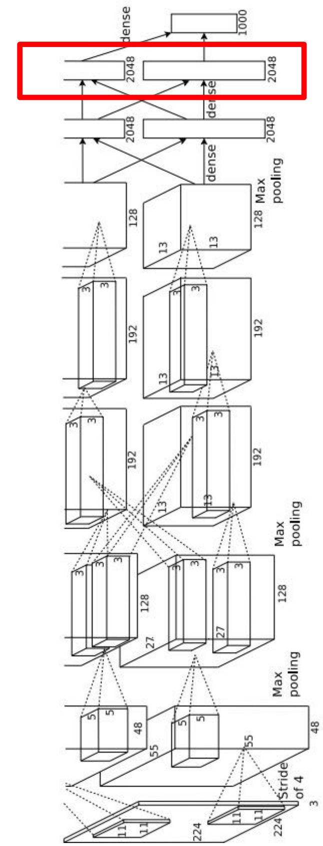
Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principal Component Analysis (PCA)

More complex: **t-SNE**

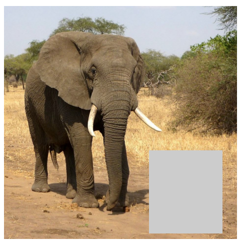


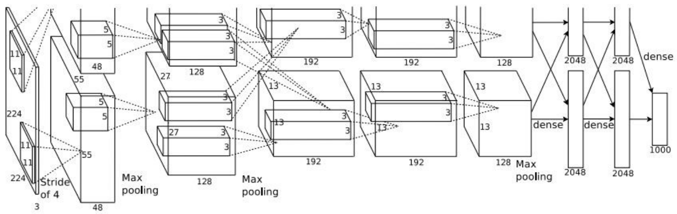


Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

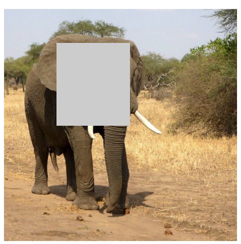
Which pixels matter: Saliency vs Occlusion

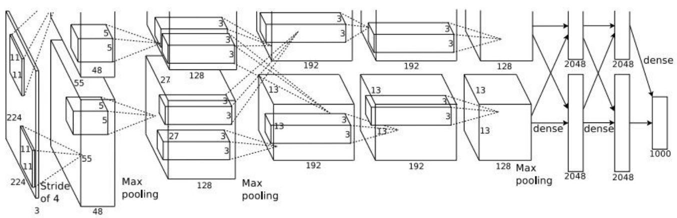
Mask part of the image before feeding to CNN, check how much predicted probabilities change





P(elephant) = 0.95



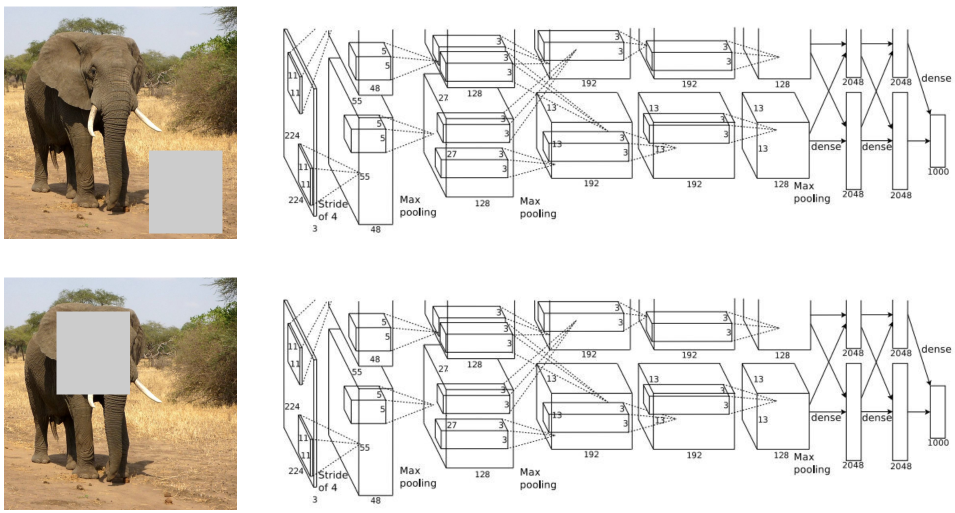


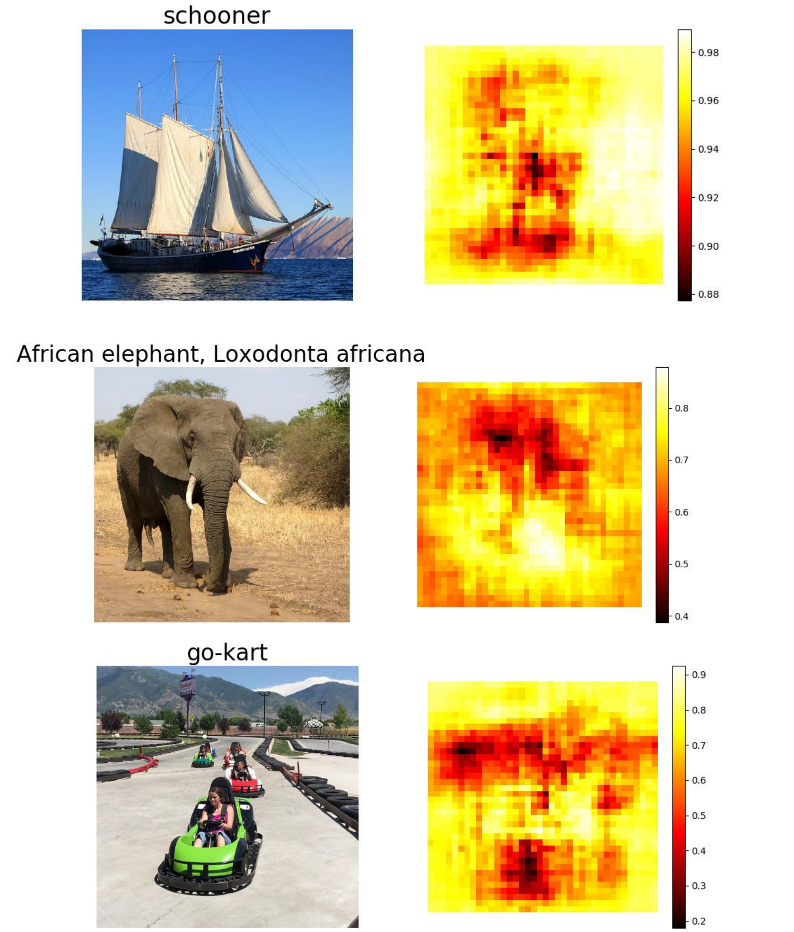
P(elephant) = 0.75

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Which pixels matter: Saliency vs Occlusion

Mask part of the image before feeding to CNN, check how much predicted probabilities change

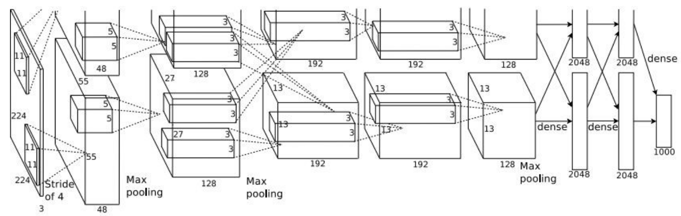




Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

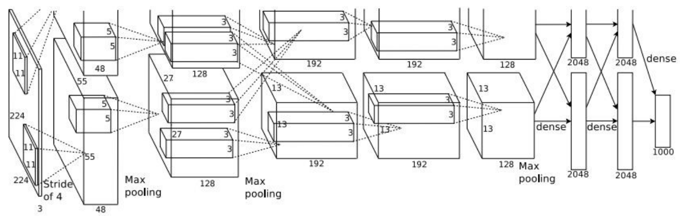


Dog

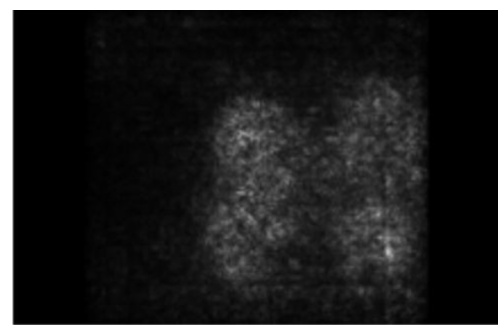
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



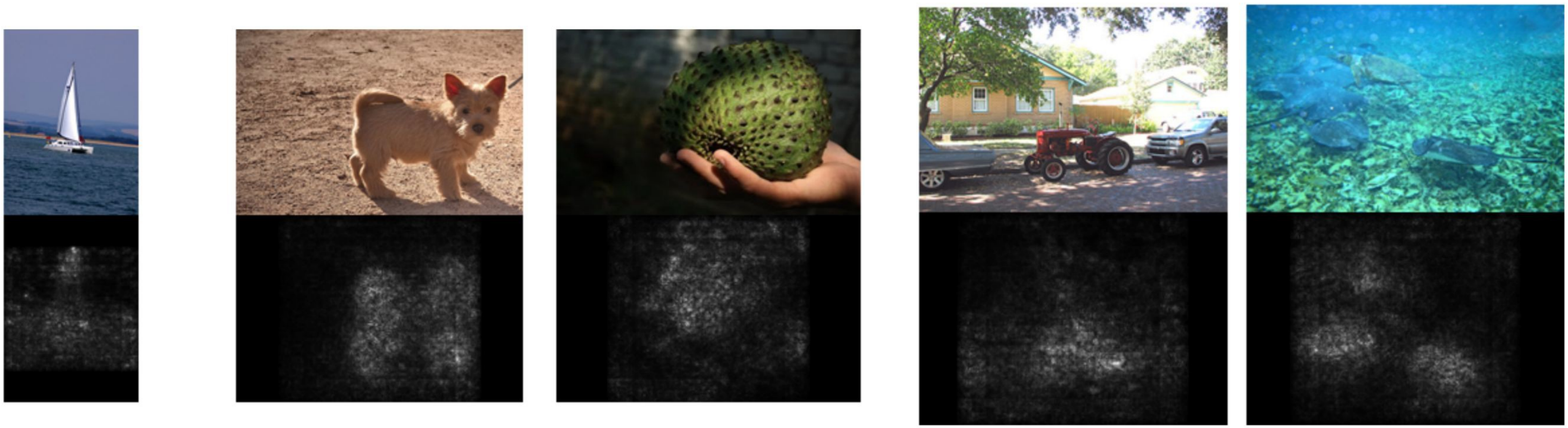
Dog



Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels

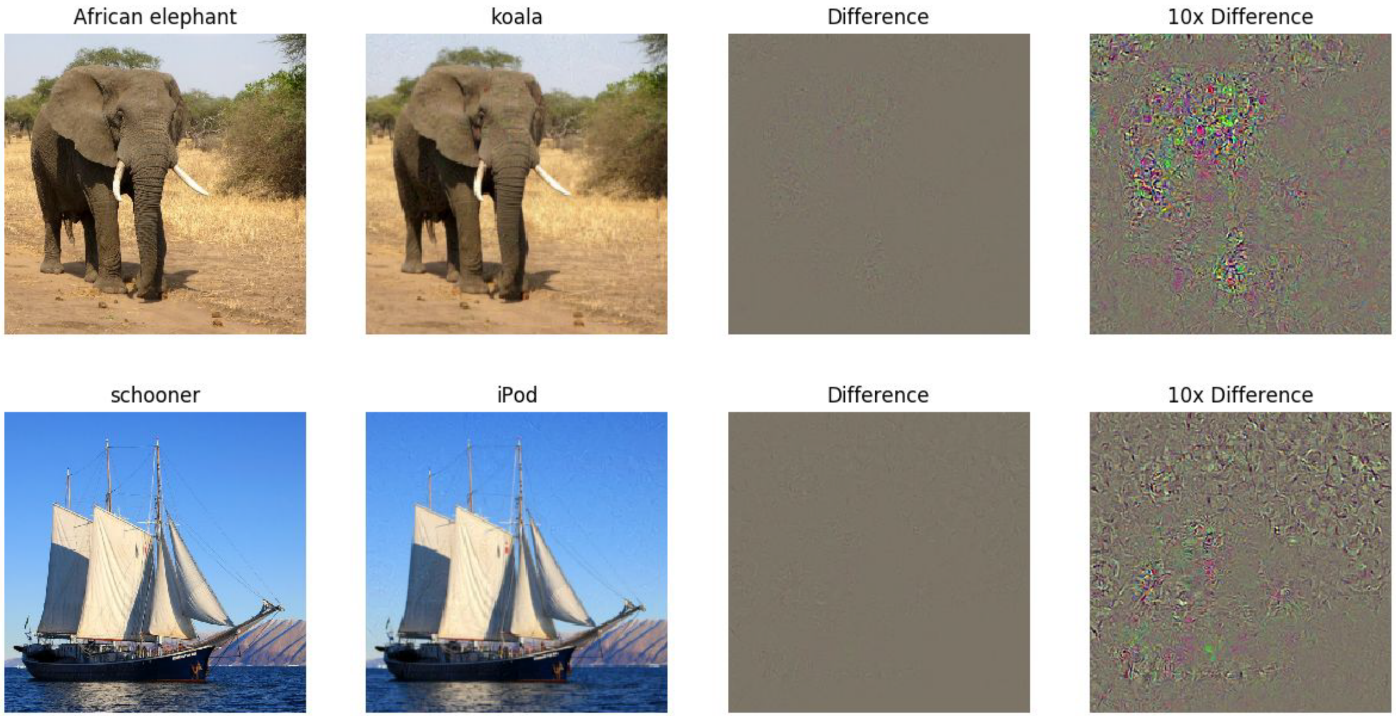
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Saliency Maps



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Fooling Images / Adversarial Examples



Fooling Images / Adversarial Examples

(1) Start from an arbitrary image
(2) Pick an arbitrary class
(3) Modify the image to maximize the class
(4) Repeat until network is fooled

Optimization Formulation

Target class

African elephant

$\mathbf{I}_{ori}$

koala

$\mathbf{I}_{adv}$

$Score_c(\mathbf{I}; \theta)$: the confidence score of an image belonging to class c, using a network of parameters θ

Attack: Modify the image $\mathbf{I}$ to increase $Score_{target\ class}(\mathbf{I}; \theta)$

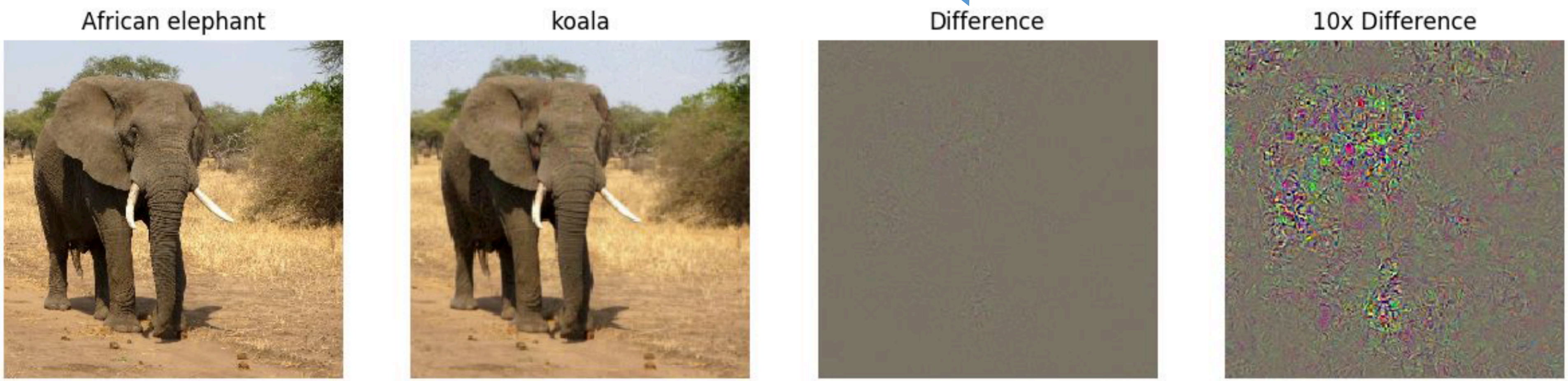
$$\underset{\mathbf{I}_{adv}}{\text{maximize}} \quad Score_{target\ class}(\mathbf{I}_{adv})$$

$$\text{subject to} \quad \|\mathbf{I}_{adv} - \mathbf{I}_{ori}\| \leq \epsilon$$

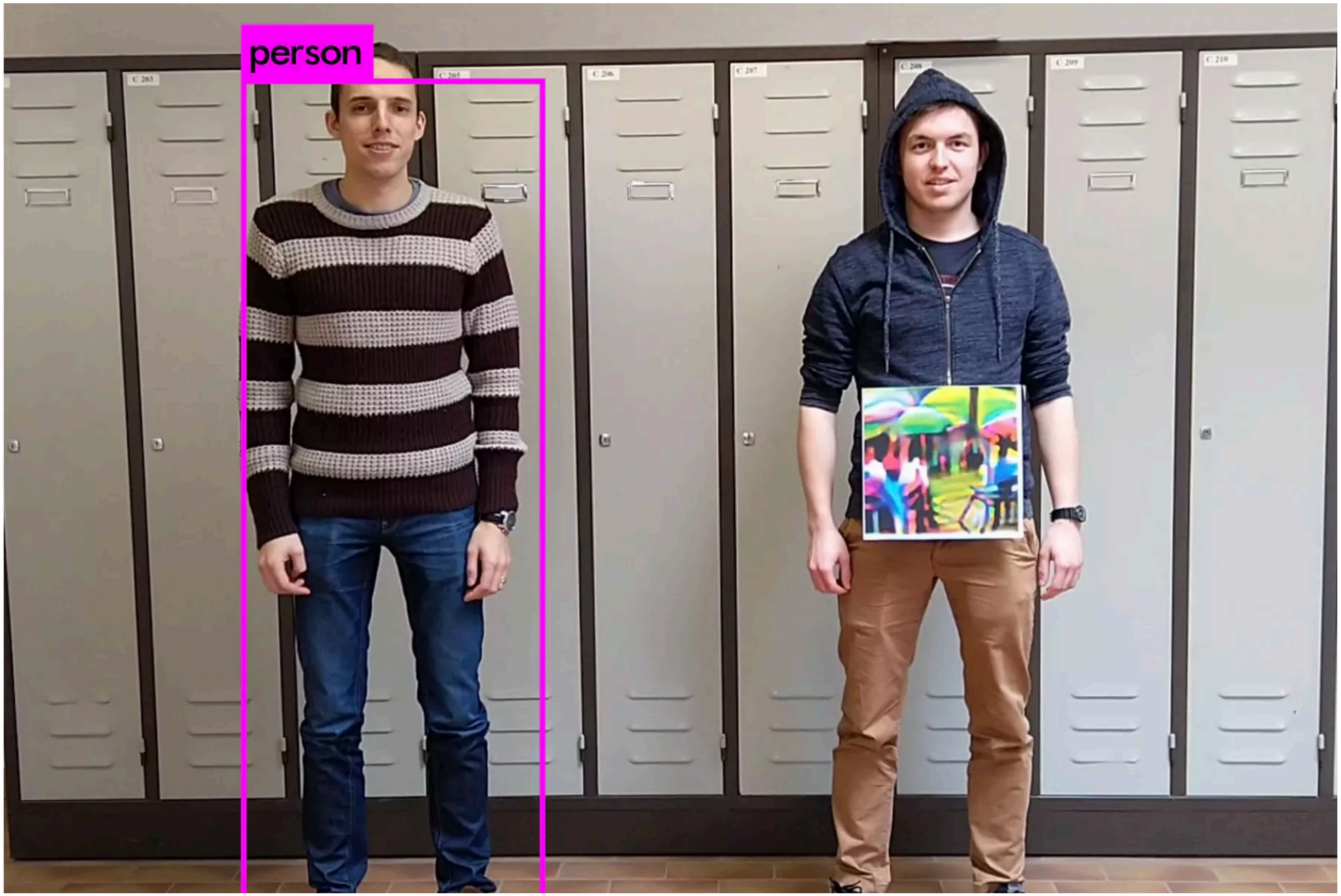
Gradient-based Attack

Fast Gradient Sign Method:

$$\mathbf{I}^{adv} = \mathbf{I} + \boxed{\epsilon \,\mathrm{sign}(\nabla_{\mathbf{I}} Score_{target\ class}(\mathbf{I}))}$$



Patch-based Attack (Spatially Localized)



Dangerous!

(a) (b) (c) (d) (e)

Neural Style Transfer

Content Image

+

Style Image

Gatys, Ecker, and Bethge, “Texture Synthesis Using Convolutional Neural Networks”, NIPS 2015

Neural Style Transfer

Content Image

+

Style Image

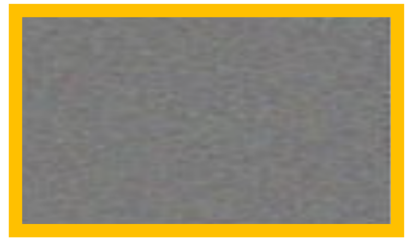
=

Style Transfer!

Gatys, Ecker, and Bethge, “Texture Synthesis Using Convolutional Neural Networks”, NIPS 2015

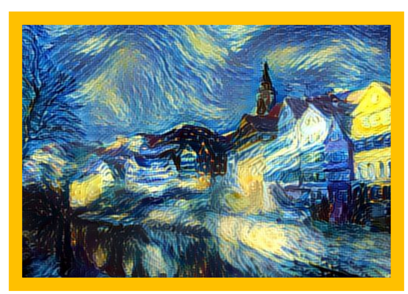
Total Loss

$\mathbf{I}$



$$\mathscr{L}_{total}(\mathbf{I}) = \alpha\,\mathscr{L}_{content}(\mathbf{I}) + \beta\,\mathscr{L}_{style}(\mathbf{I})$$

Minimize total loss



Neural Style Transfer

Example outputs

Gatys, Ecker, and Bethge, “Image style transfer using convolutional neural networks”, CVPR 2016 Figure copyright Justin Johnson, 2015.

Neural Style Transfer

More weight to content loss

More weight to style loss