EECS 498/598: Deep Learning
Lecture 3. Convolutional Neural Networks

Honglak Lee

1/25/2019

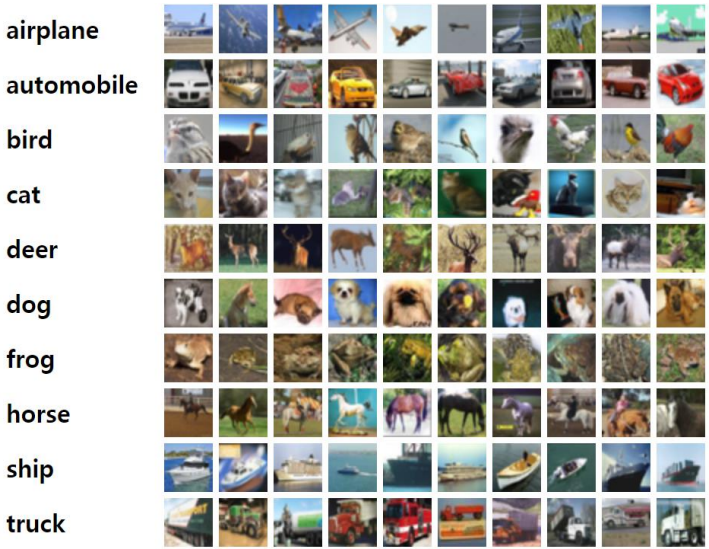
Outline

- **CNN basics**
- Examples of CNN Architectures
- Visualizing and understanding CNNs
- Pre-training/Transfer learning

Image Classification

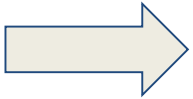
Question: How can we categorize images?

Example: CIFAR-10 dataset



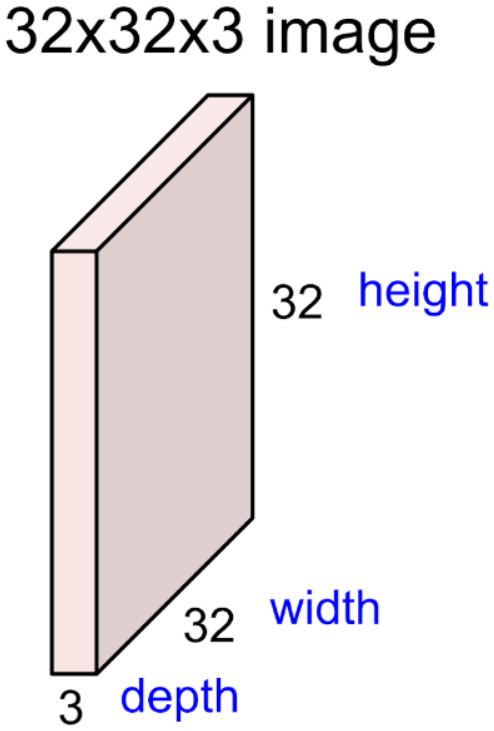
https://www.cs.toronto.edu/~kriz/cifar.html

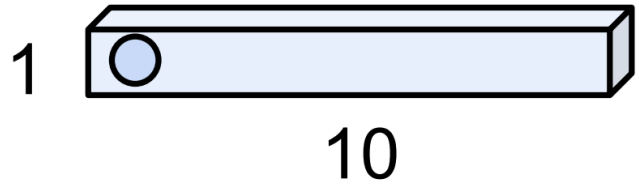
Image Classification



“airplane”

Image Classification

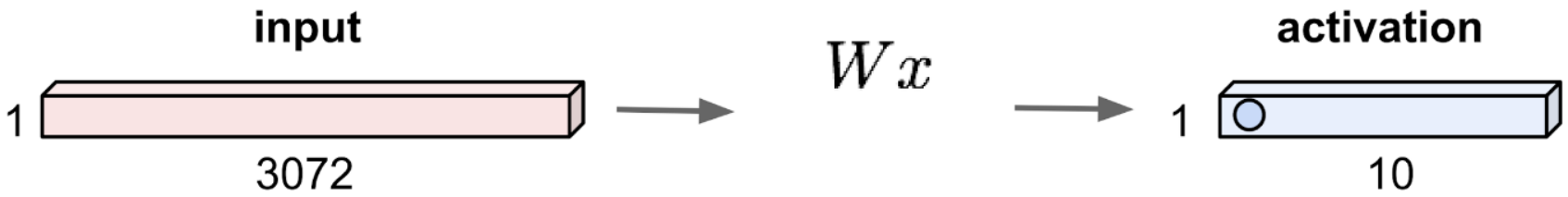




Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Fully Connected Layer

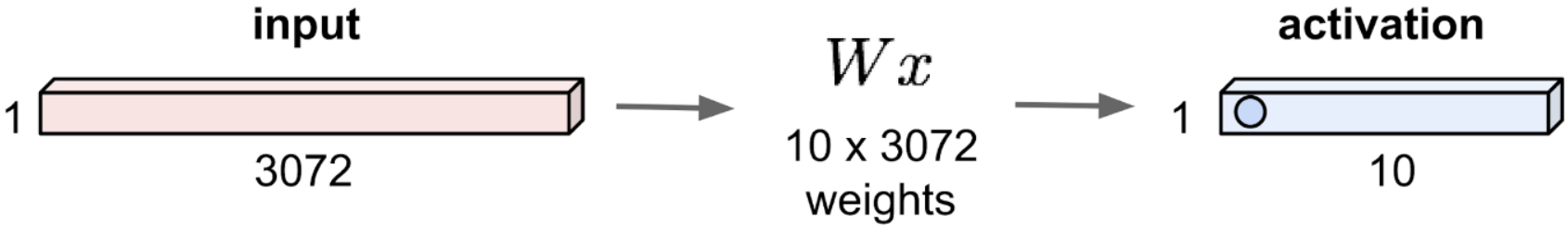
32x32x3 image -> stretch to 3072 x 1



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Fully Connected Layer

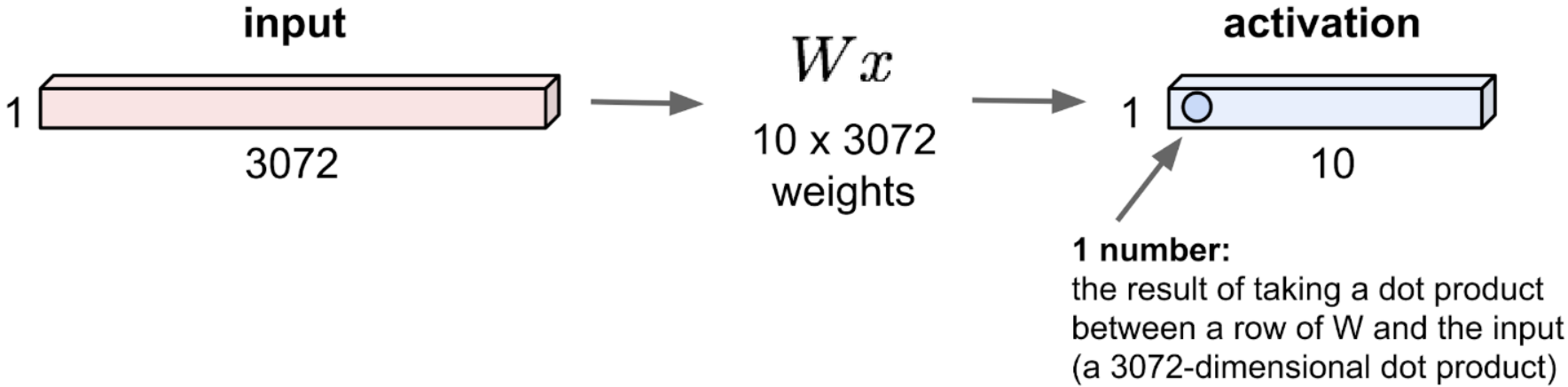
32x32x3 image -> stretch to 3072 x 1



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

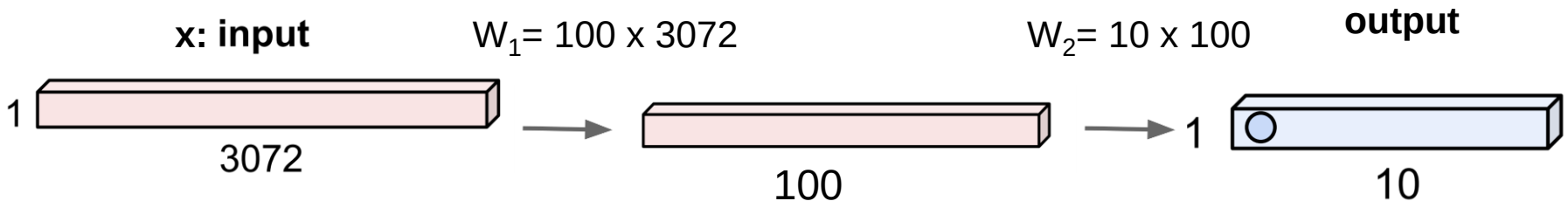
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

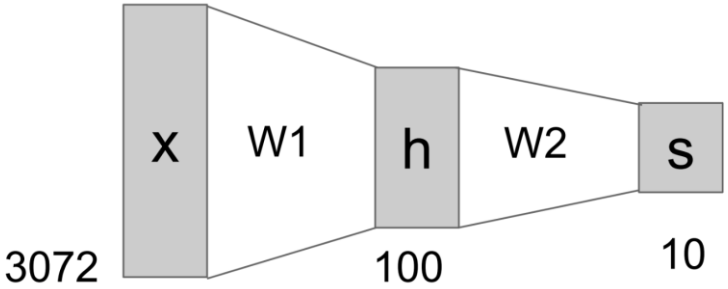


Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Neural Network with 1 hidden layer (with relu)



Alternative diagram:



Shorthand notation

$$f = W_2 \max(0, W_1 x)$$

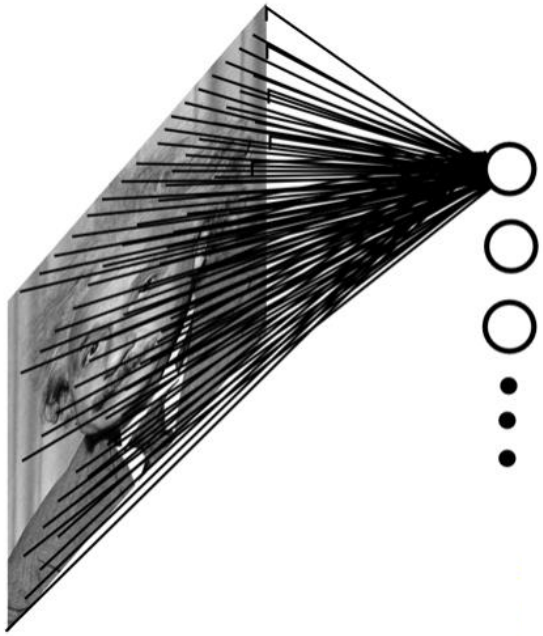
Deeper models: We can add more layers, etc.

Q. Have we solved the problem? Will this be applicable for larger-sized images?

Scaling to larger-sized images

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

Fully conn



Slide credit: Russ Salakhutdinov

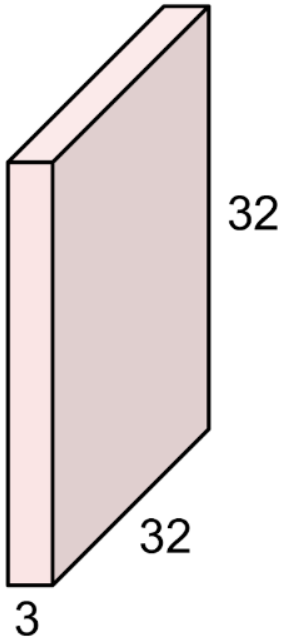
Motivation: Convolutional Neural Networks

- How can we design neural networks that are specifically adapted for classifying large-sized images?
  - Must deal with very **high-dimensional inputs**: 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - Can exploit the **2D topology** of pixels (or 3D for video data)
  - Can build in **invariance** to certain variations: translation, illumination, etc.
- **Convolutional networks** leverage these ideas
  - Local connectivity
  - Convolution, Parameter sharing
  - Pooling / subsampling hidden units

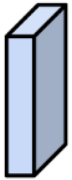
Slide credit: Russ Salakhutdinov

Convolution layer

32x32x3 image



5x5x3 filter

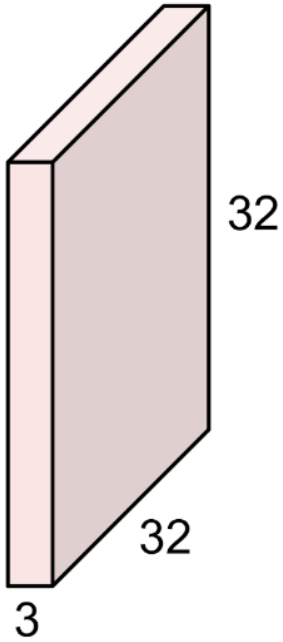


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

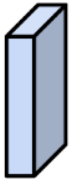
Convolution layer

Filters always extend the full depth of the input volume

32x32x3 image



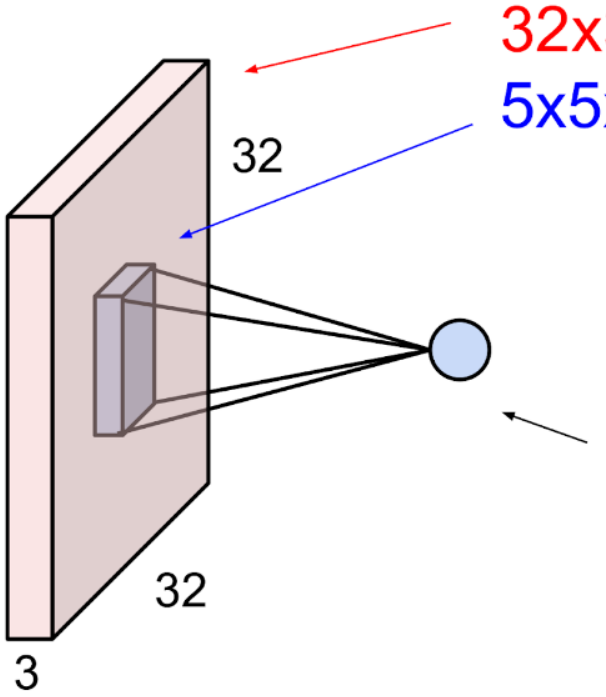
5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Convolution layer

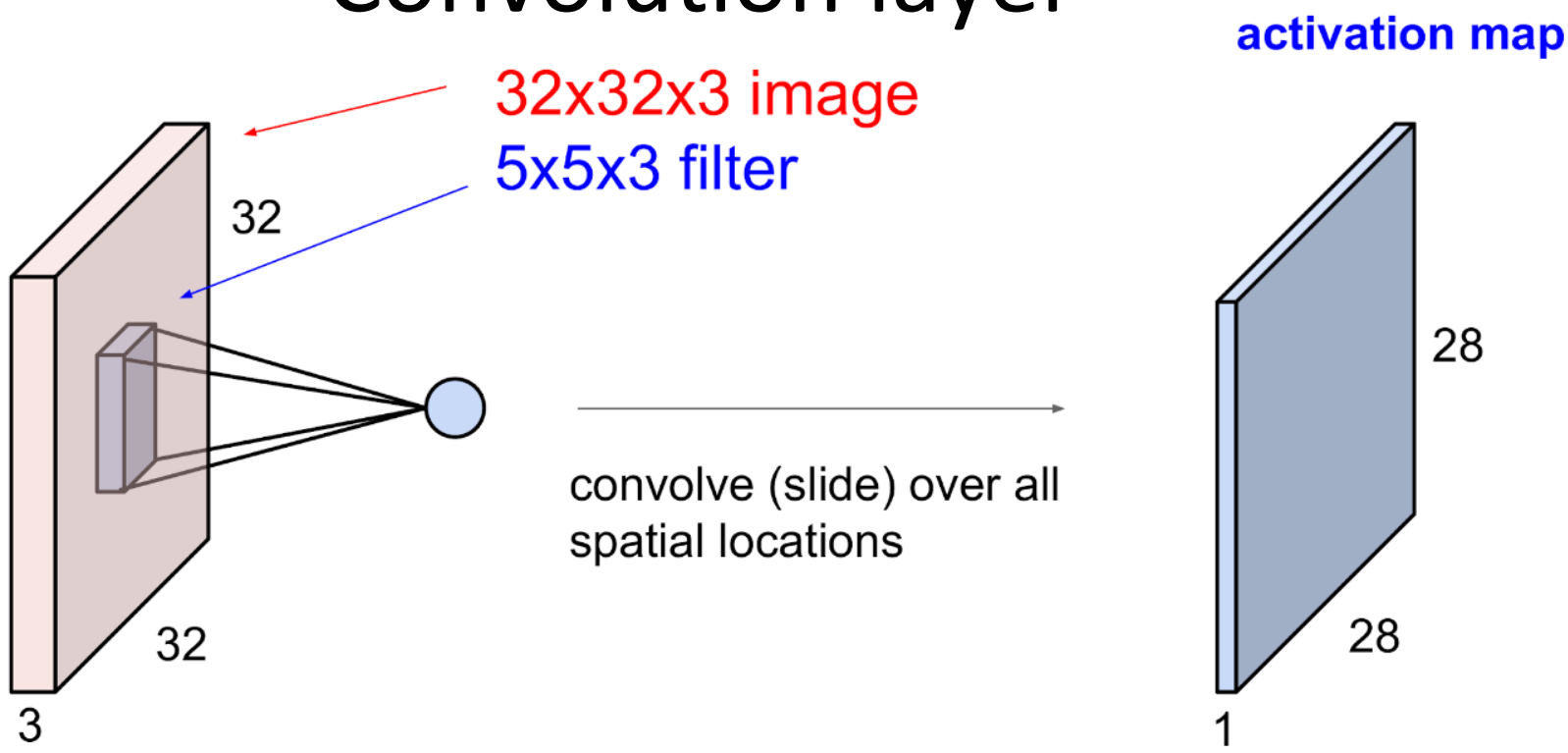


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Convolution layer



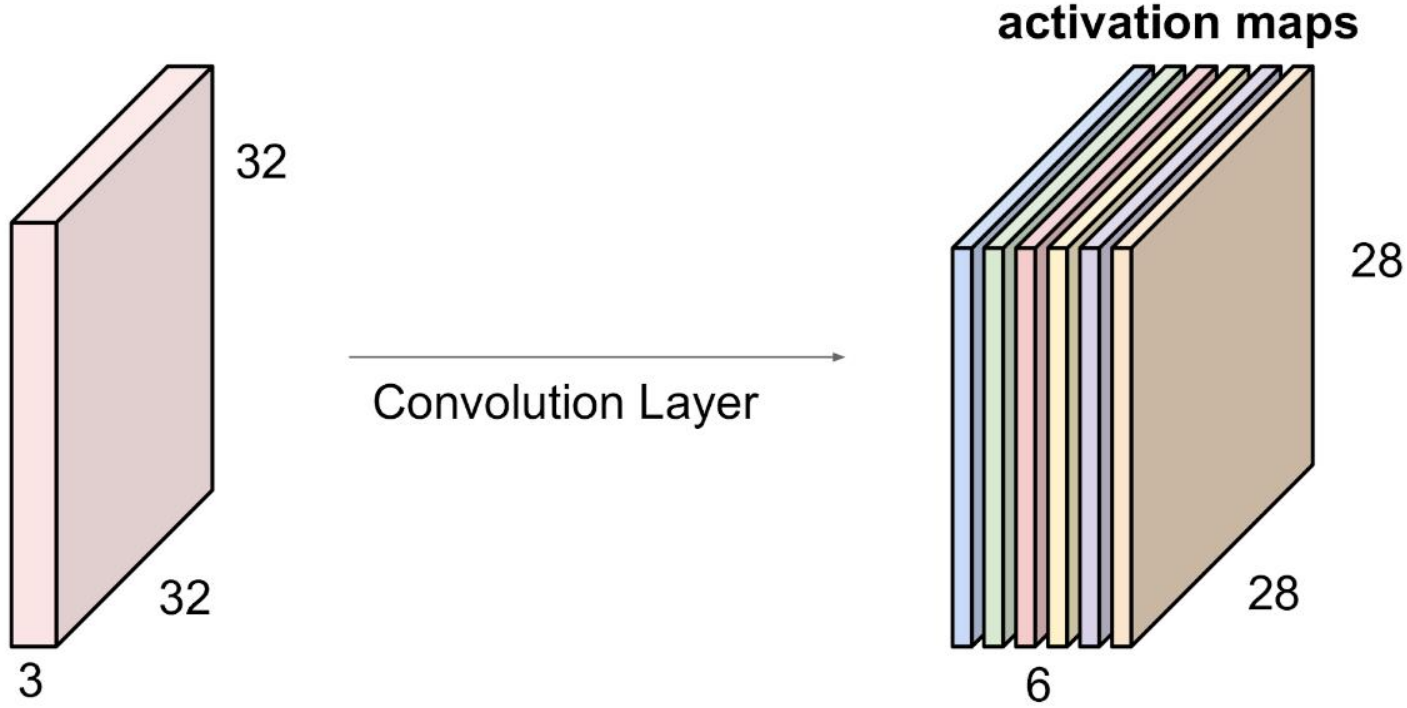
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Convolution layer

consider a second, green filter

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

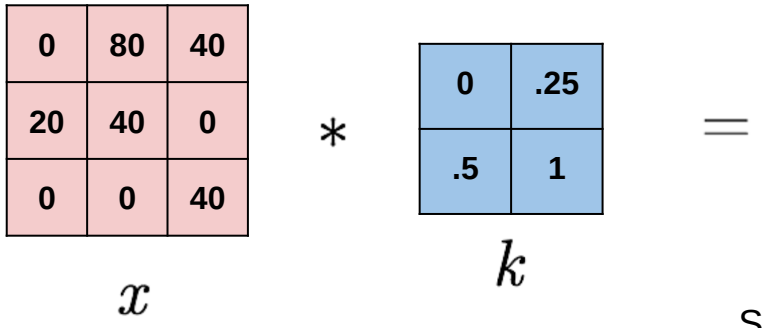
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Discrete Convolution

- The convolution of an image x with kernel k is computed as follows

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

- Example



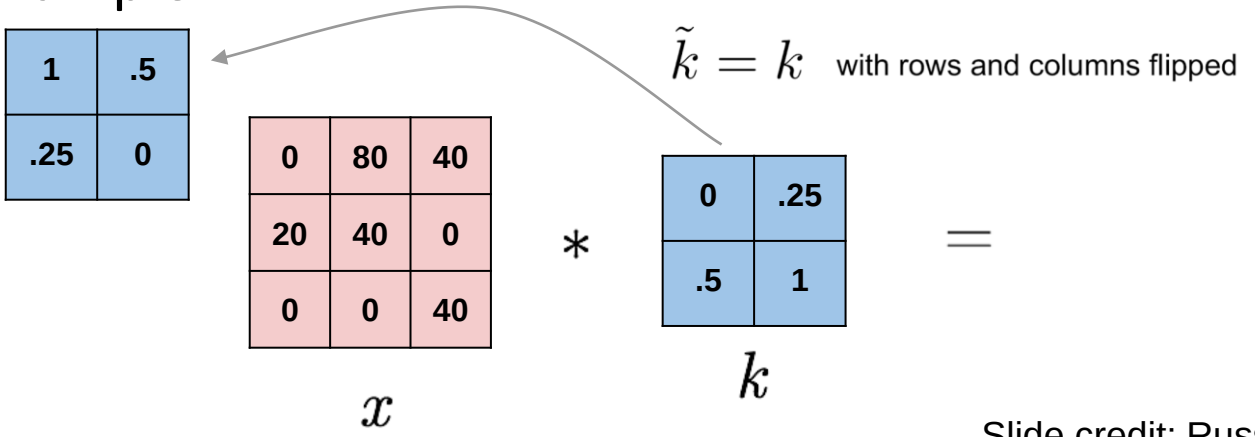
Slide credit: Russ Salakhutdinov

Discrete Convolution

- The convolution of an image x with kernel k is computed as follows

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

- Example



Slide credit: Russ Salakhutdinov

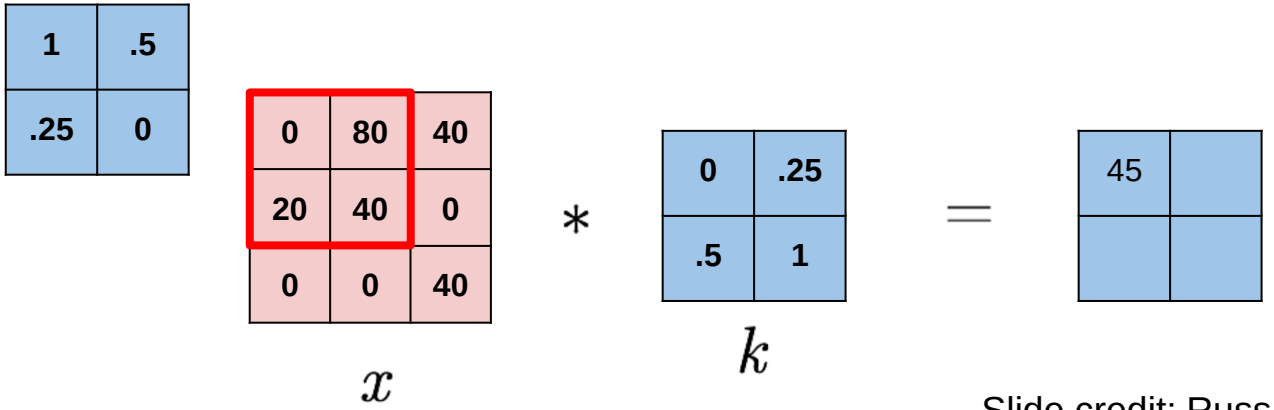
Discrete Convolution

- The convolution of an image x with kernel k is computed as follows

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

- Example

1 x 0 + 0.5 x 80 + 0.25 x 20 + 0 x 40 = 45



Slide credit: Russ Salakhutdinov

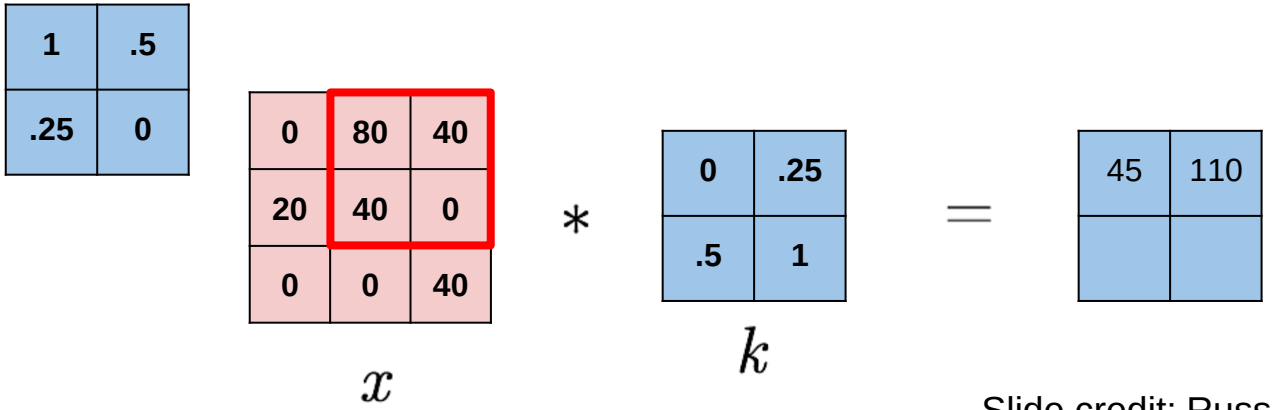
Discrete Convolution

- The convolution of an image x with kernel k is computed as follows

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

- Example

1 x 80 + 0.5 x 40 + 0.25 x 40 + 0 x 0 = 110



Slide credit: Russ Salakhutdinov

Discrete Convolution

- The convolution of an image x with kernel k is computed as follows

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

- Example

1 x 20 + 0.5 x 40 + 0.25 x 0 + 0 x 0 = 40

1	.5
.25	0

0	80	40
20	40	0
0	0	40

x

$*$

0	.25
.5	1

k

$=$

45	110
40	

Slide credit: Russ Salakhutdinov

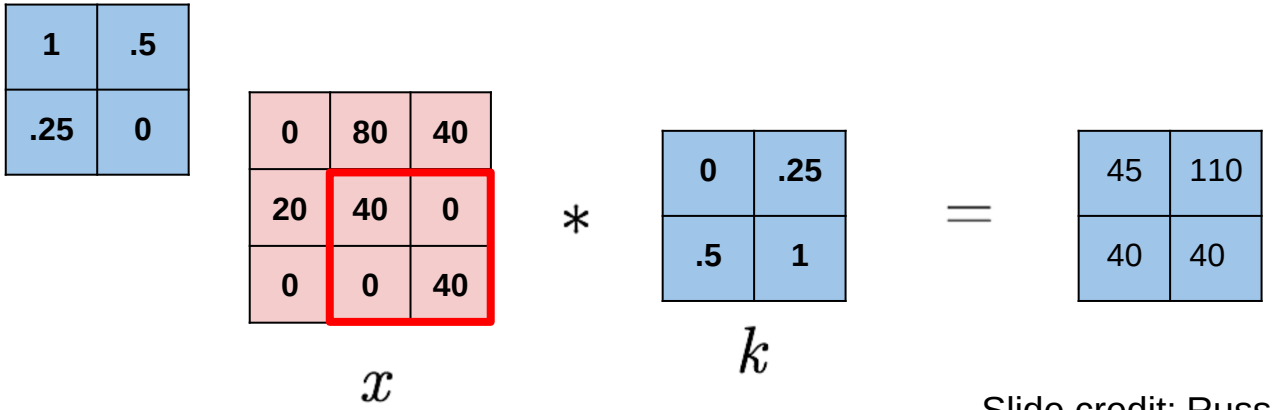
Discrete Convolution

- The convolution of an image x with kernel k is computed as follows

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

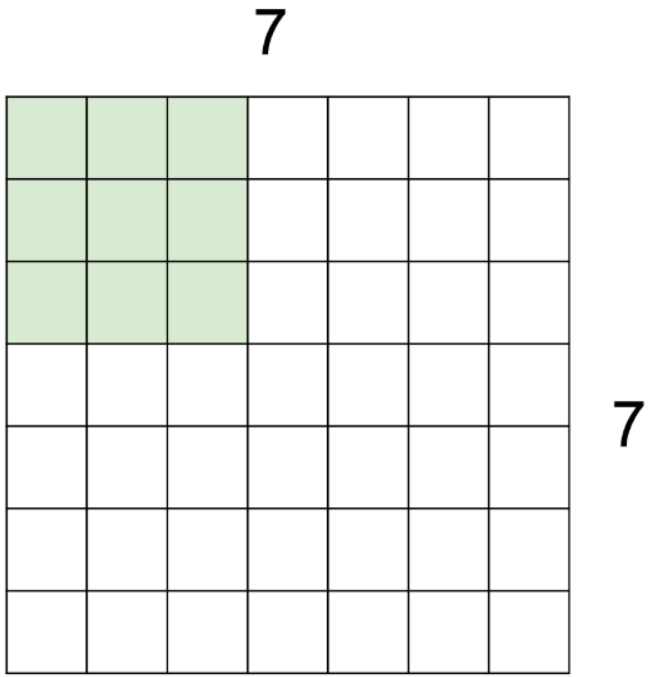
- Example

1 x 40 + 0.5 x 0 + 0.25 x 0 + 0 x 40 = 40



Slide credit: Russ Salakhutdinov

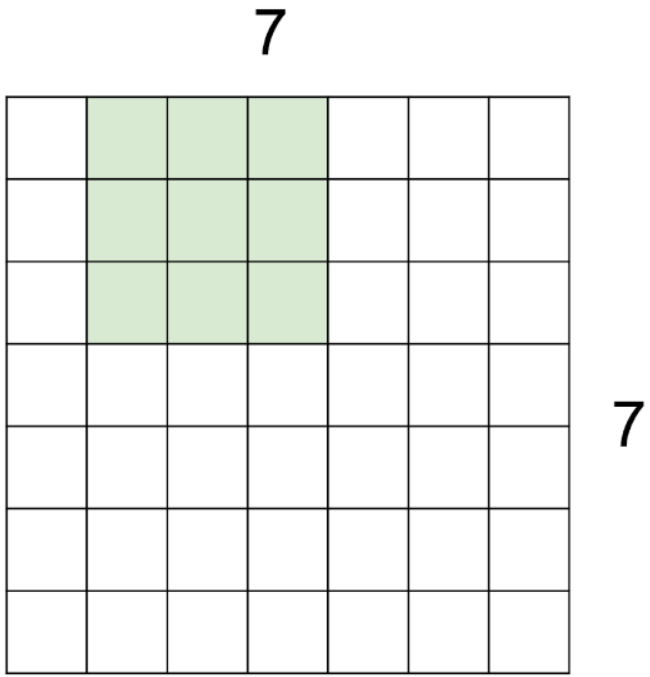
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

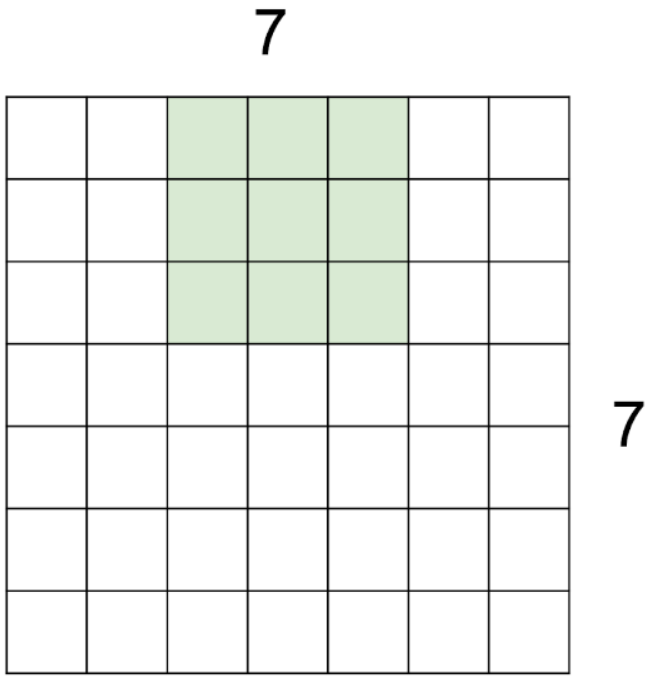
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

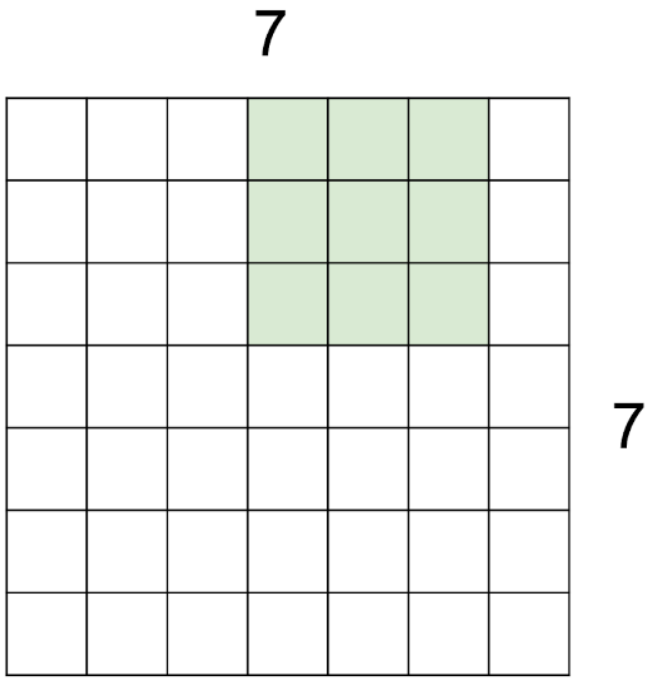
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

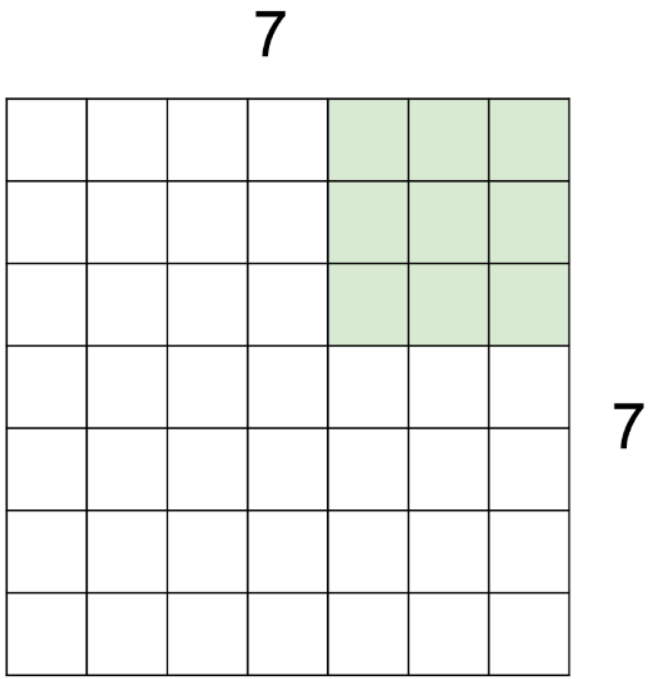
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Closer look at spatial dimensions

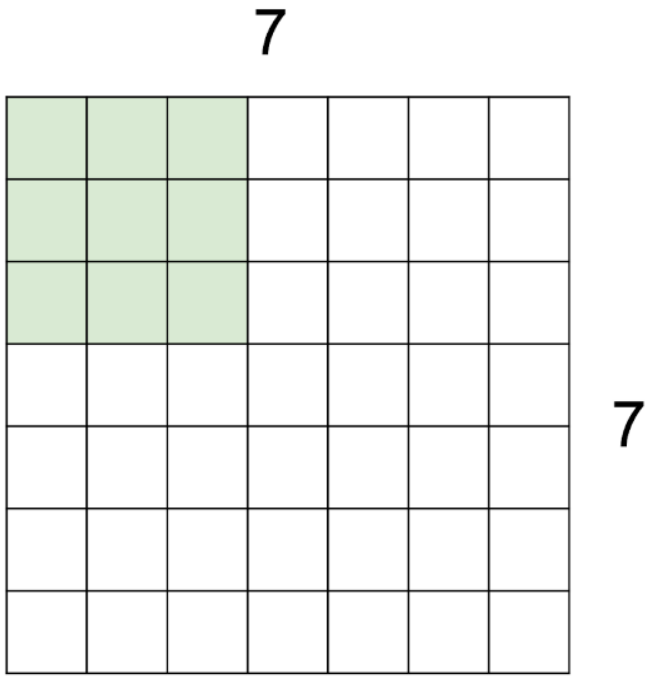


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

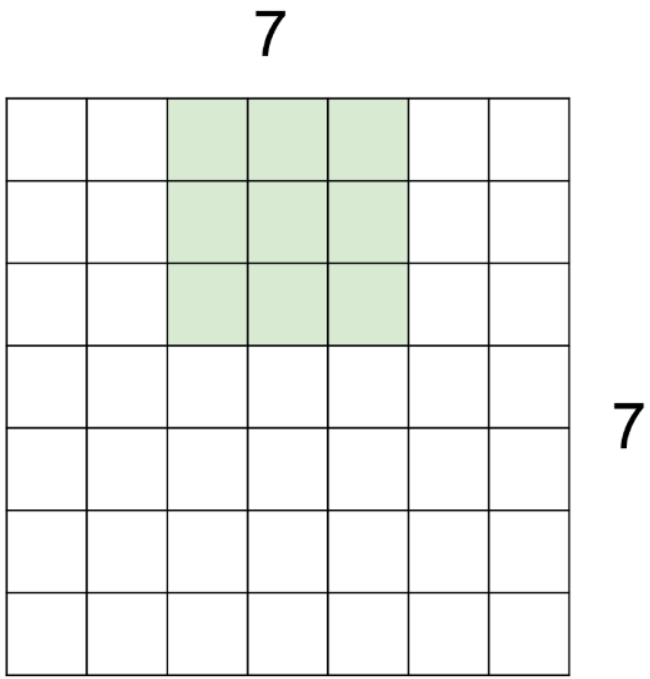
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

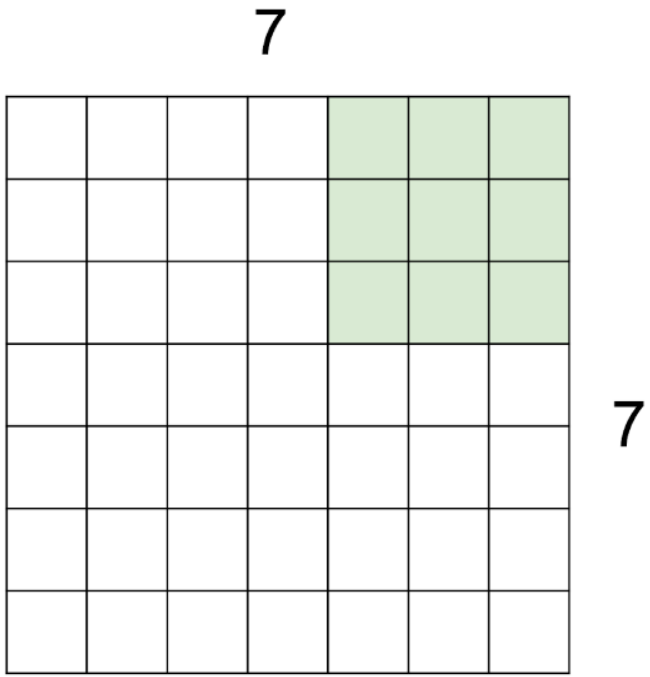
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

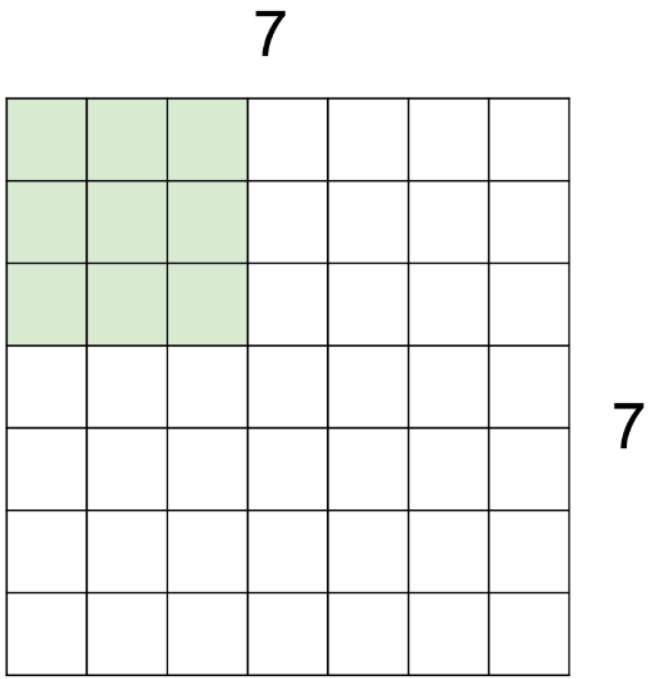
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

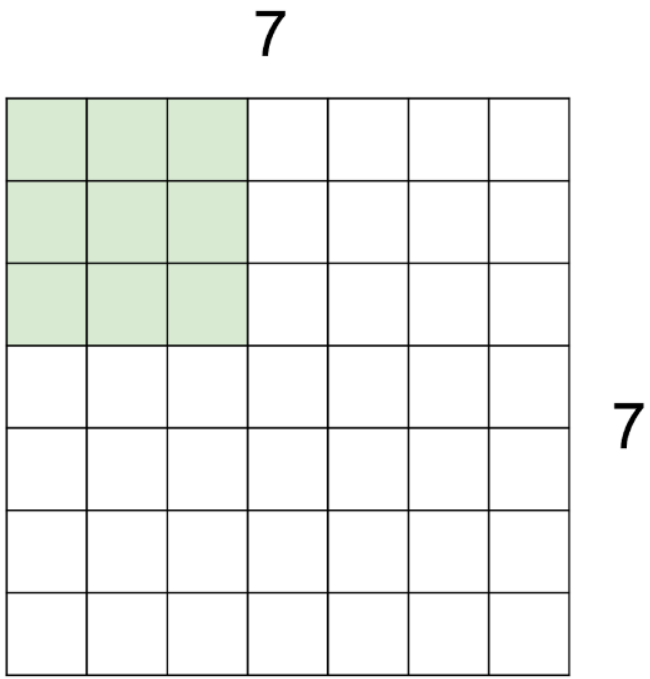
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Closer look at spatial dimensions

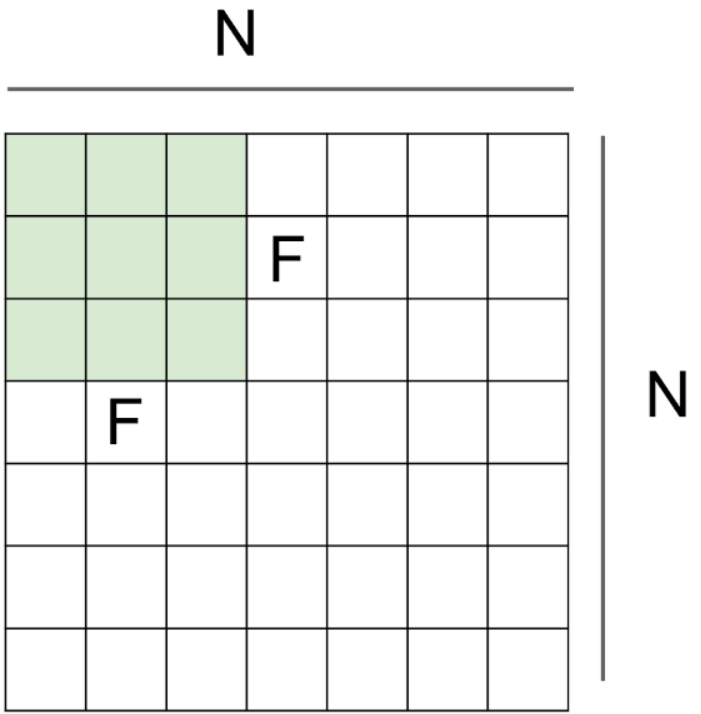


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Closer look at spatial dimensions

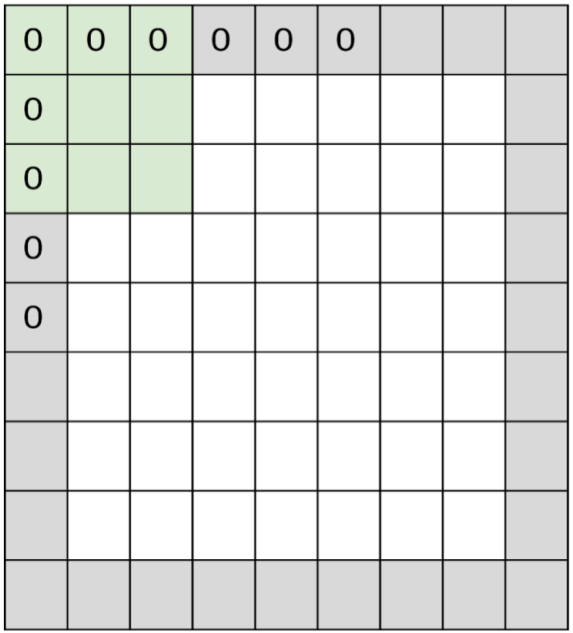


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

In practice: Common to zero pad the border

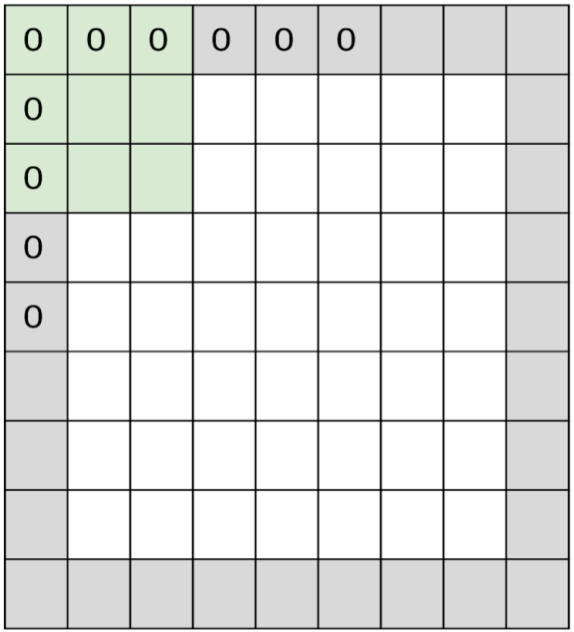


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

In practice: Common to zero pad the border

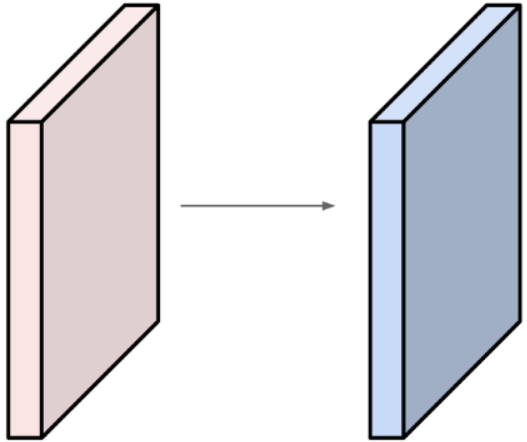


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Example



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Convolution - Summary

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

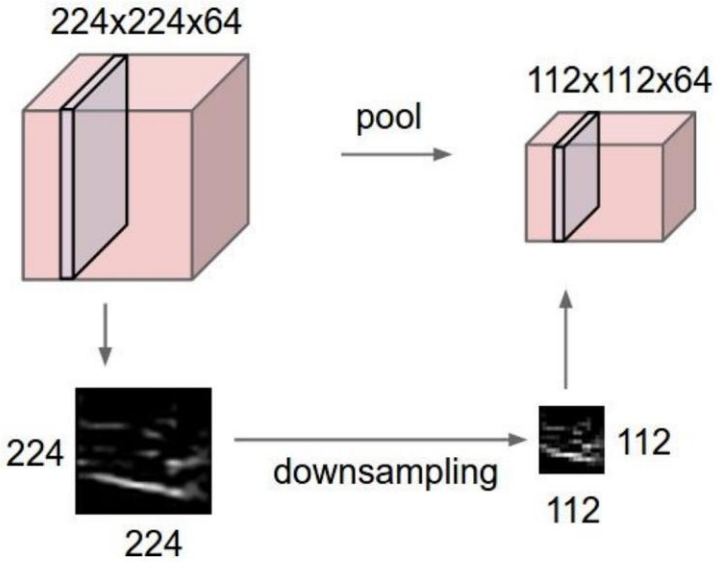
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

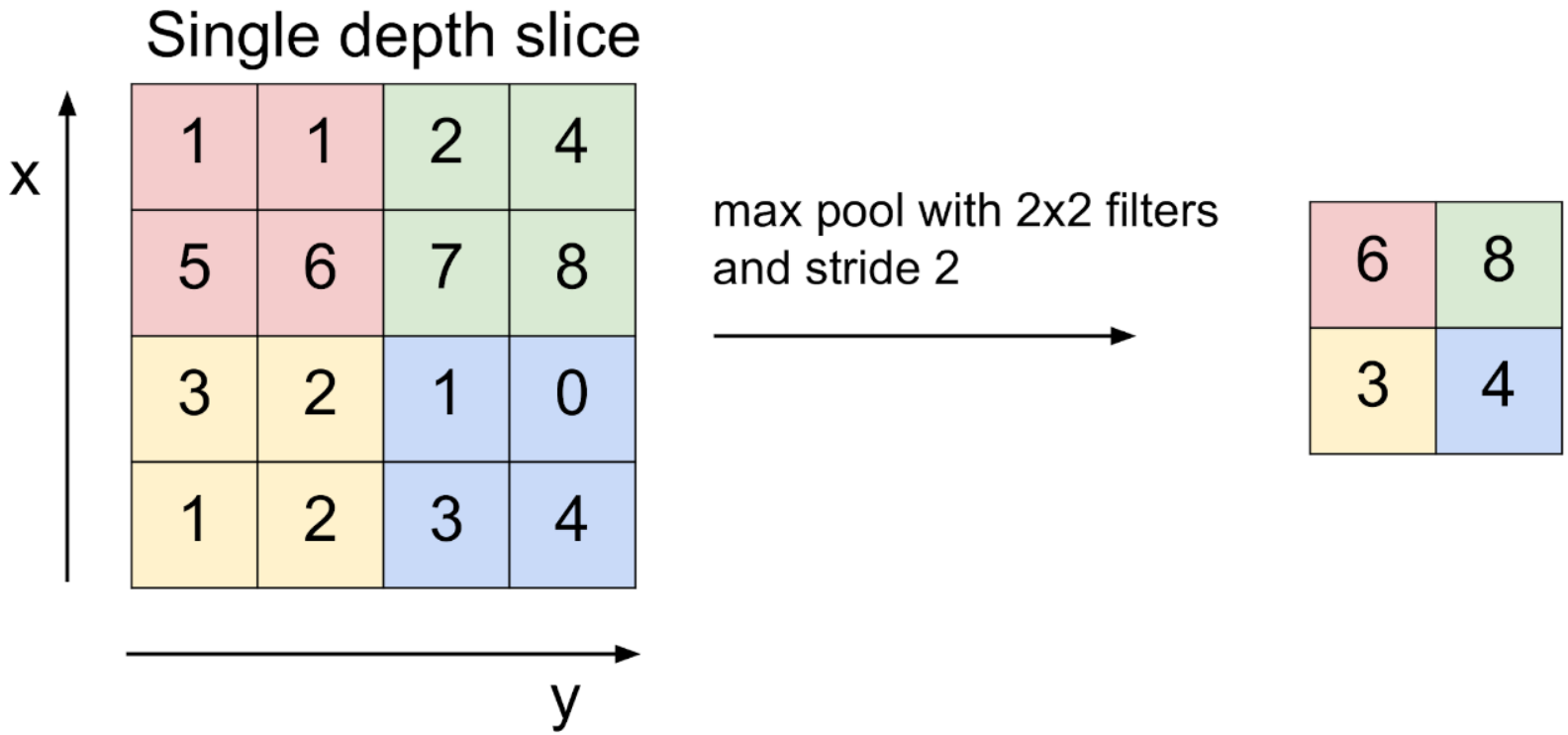
Pooling

- Makes the representations smaller and more manageable
- Operates over each activation map independently



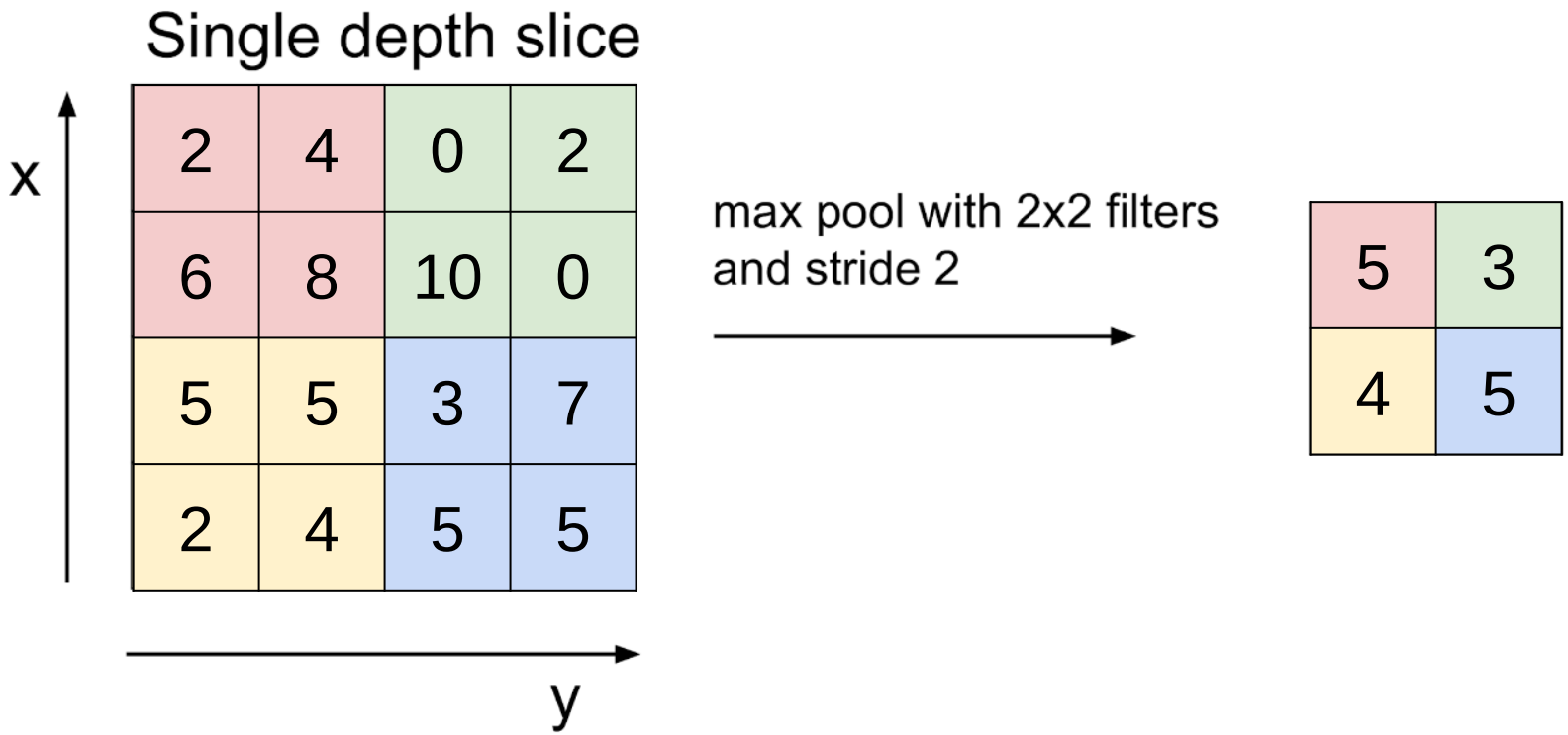
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Max Pooling



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Average Pooling



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Pooling - Summary

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Common settings:

F = 2, S = 2
F = 3, S = 2

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Classic ConvNet Architecture

Input

Conv blocks

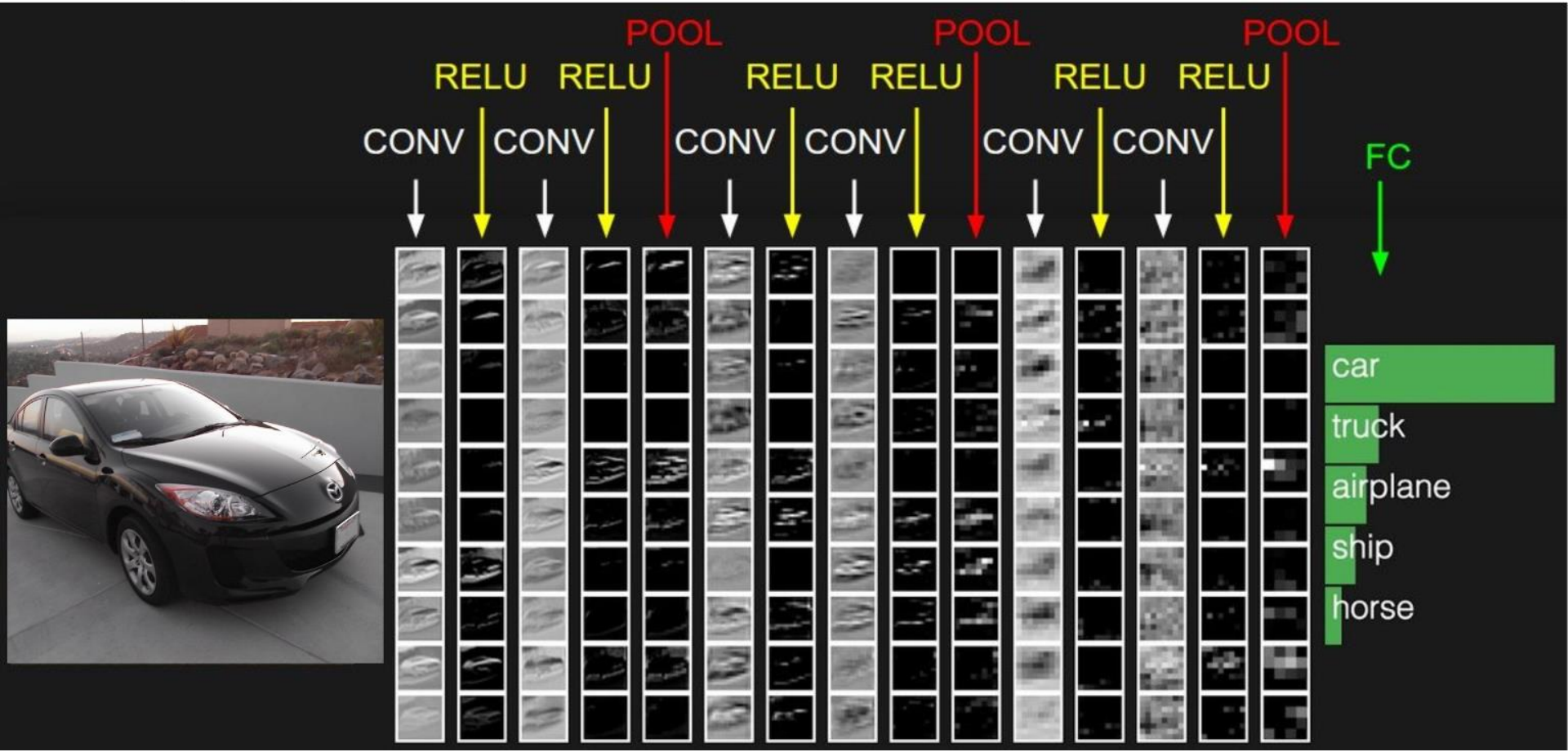
- Convolution + activation (relu)
- Convolution + activation (relu)
- ...
- Maxpooling 2x2

Output

- Fully connected layers
- Softmax

Slide credit: Charles Ollion - Olivier Grisel

two more layers to go: POOL/FC



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

ConvNetJS demo: training on CIFAR-10

ConvNetJS CIFAR-10 demo

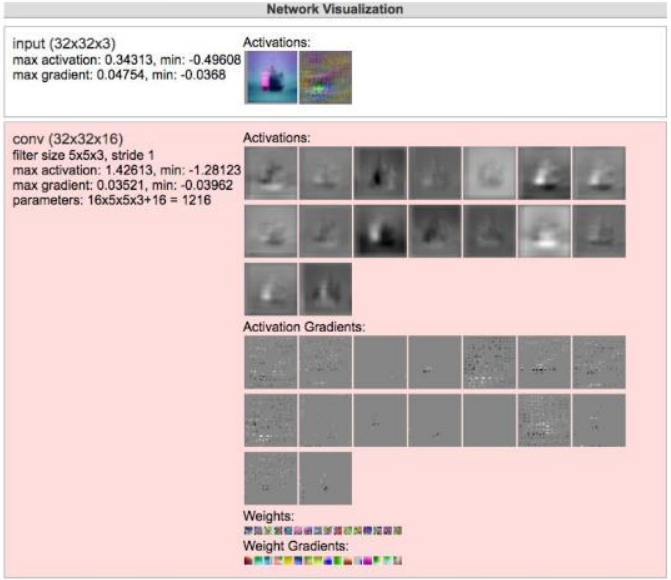
Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.



http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Summary

- ConvNets stack CONV, POOL, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like **[(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX**

  where N is usually up to ~5, M is large, 0 <= K <= 2.
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm

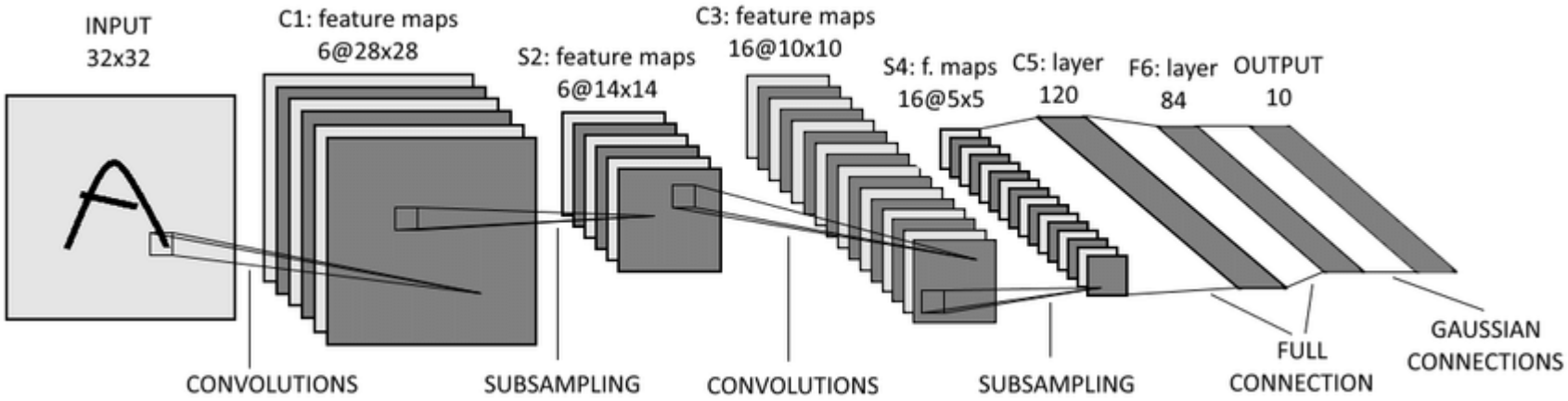
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Outline

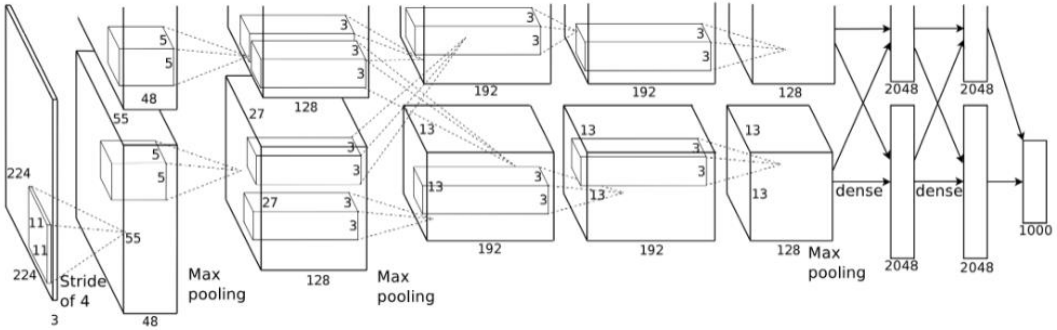
- CNN basics
- **Examples of CNN Architectures**
- Visualizing and understanding CNNs
- Pre-training/Transfer learning

LeCun et al. 1989

Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner, 1989]



AlexNet



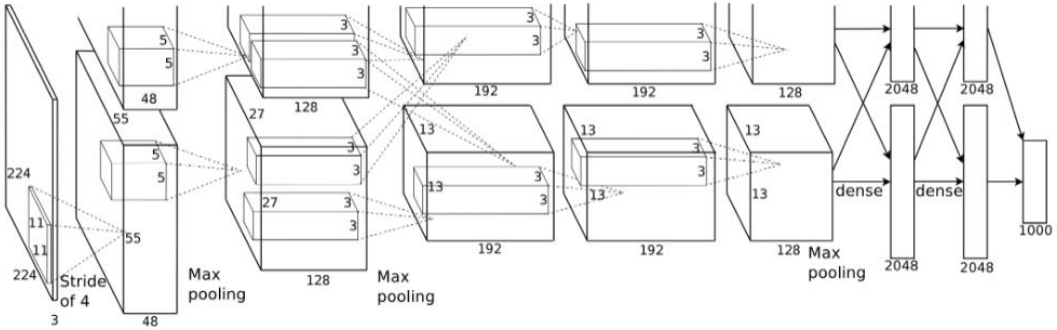
First conv layer: kernel 11x11x3x96 stride 4

- Kernel shape: (11,11,3,96)
- Output shape: (55,55,96)
- Number of parameters: 34,944
- Equivalent MLP parameters: 43.7 x 1e9

Simplified version of Krizhevsky, Alex, Sutskever, and Hinton. "Imagenet classification with deep convolutional neural networks." NIPS 2012

Slide credit: Charles Ollion - Olivier Grisel

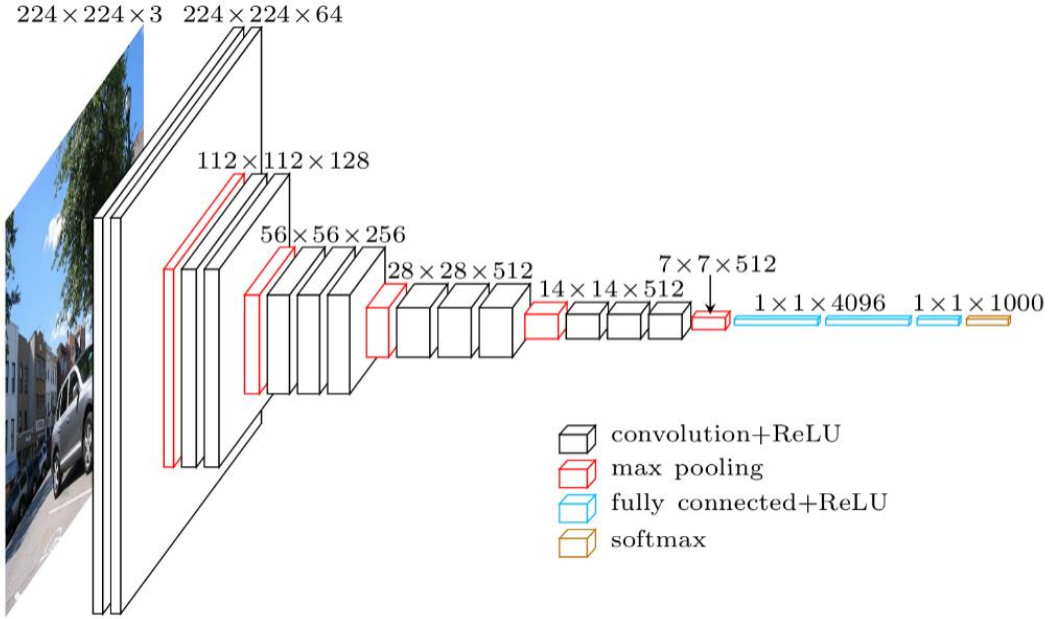
AlexNet




```
INPUT:     [227x227x3]
CONV1:     [55x55x96]   96 11x11 filters at stride 4, pad 0
MAX POOL1: [27x27x96]      3x3   filters at stride 2
CONV2:     [27x27x256] 256 5x5   filters at stride 1, pad 2
MAX POOL2: [13x13x256]     3x3   filters at stride 2
CONV3:     [13x13x384] 384 3x3   filters at stride 1, pad 1
CONV4:     [13x13x384] 384 3x3   filters at stride 1, pad 1
CONV5:     [13x13x256] 256 3x3   filters at stride 1, pad 1
MAX POOL3: [6x6x256]       3x3   filters at stride 2
FC6:       [4096]      4096 neurons
FC7:       [4096]      4096 neurons
FC8:       [1000]      1000 neurons (softmax logits)
```

Slide credit: Charles Ollion - Olivier Grisel

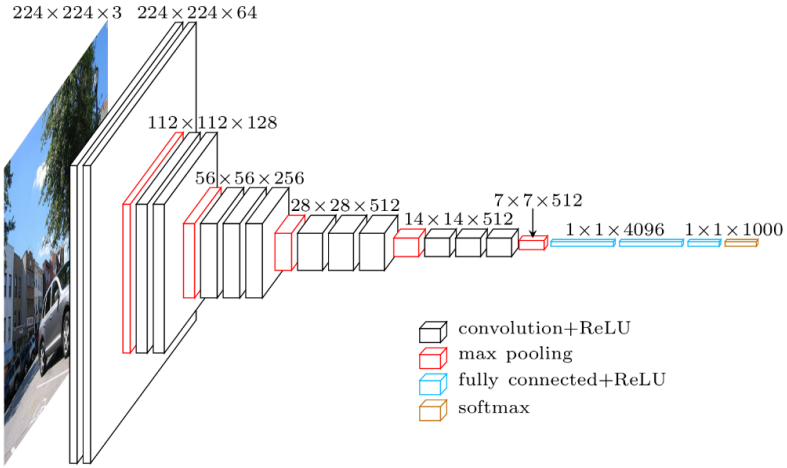
VGG-16



Simonyan, Karen, and Zisserman. "Very deep convolutional networks for large-scale image recognition." (2014)

Slide credit: Charles Ollion - Olivier Grisel

VGG-16



Lots of 3x3 convolution layers: more non-linearity than a single 7x7 conv layer

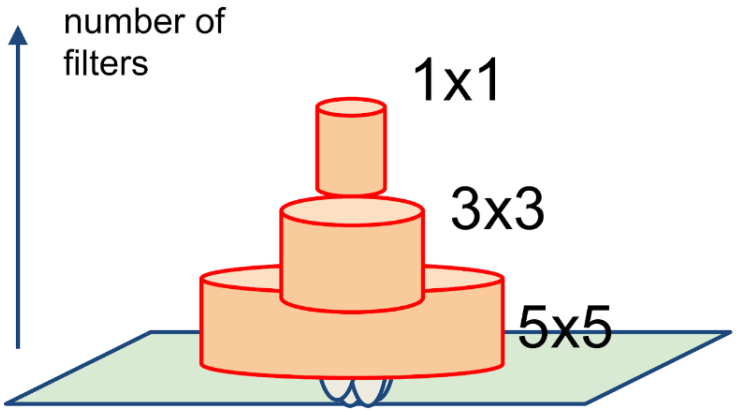
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128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

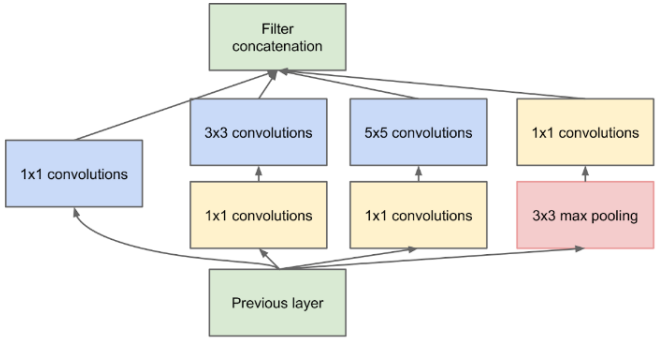
Very Deep Models

Going deep with convolutions, Szegedy et al.

GoogLeNet inception module:

1. Multiple filter scales at each layer
2. Dimensionality reduction to keep computational requirements down

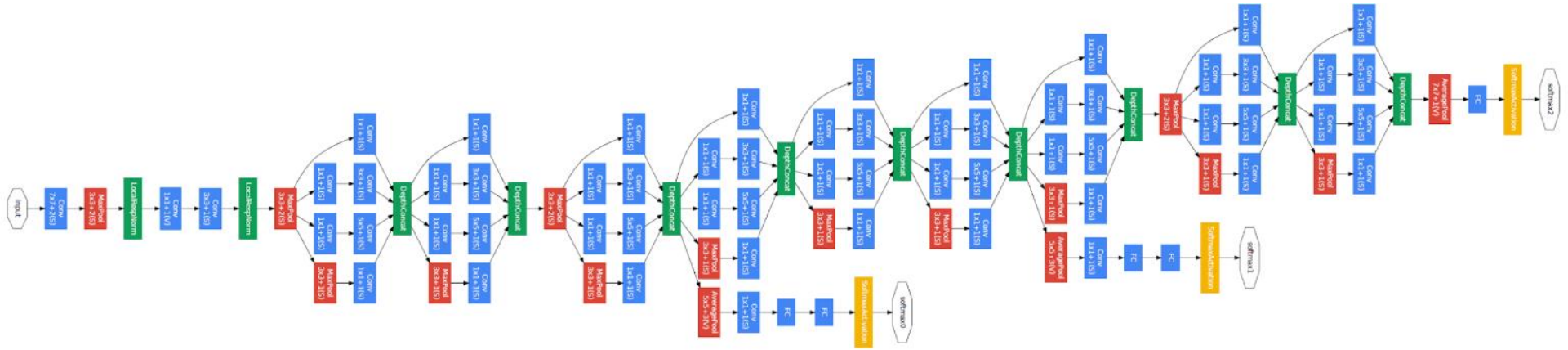




Slide credit: Rob Fergus

Very Deep Models

Going deep with convolutions, Szegedy et al.

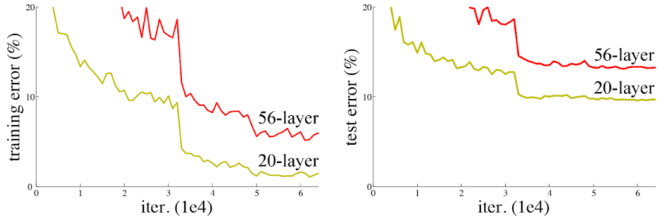


Slide credit: Rob Fergus

Residual Networks

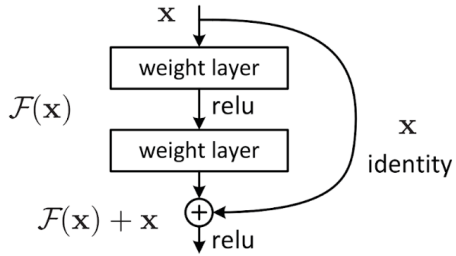
[He, Zhang, Ren, Sun, CVPR 2016]

Really, really deep convnets don't train well,
E.g. CIFAR10:



Key idea: introduce "pass through" into each layer

Thus only residual now needs to be learned

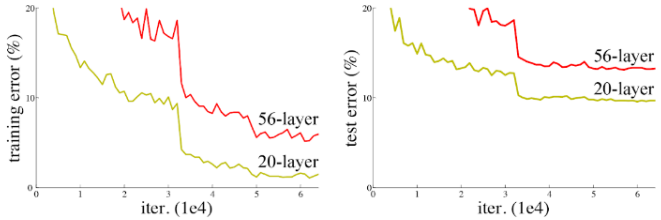


Slide credit
Rob Fergus

Residual Networks

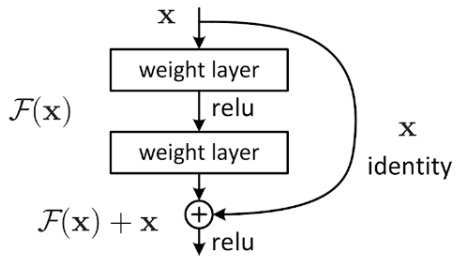
[He, Zhang, Ren, Sun, CVPR 2016]

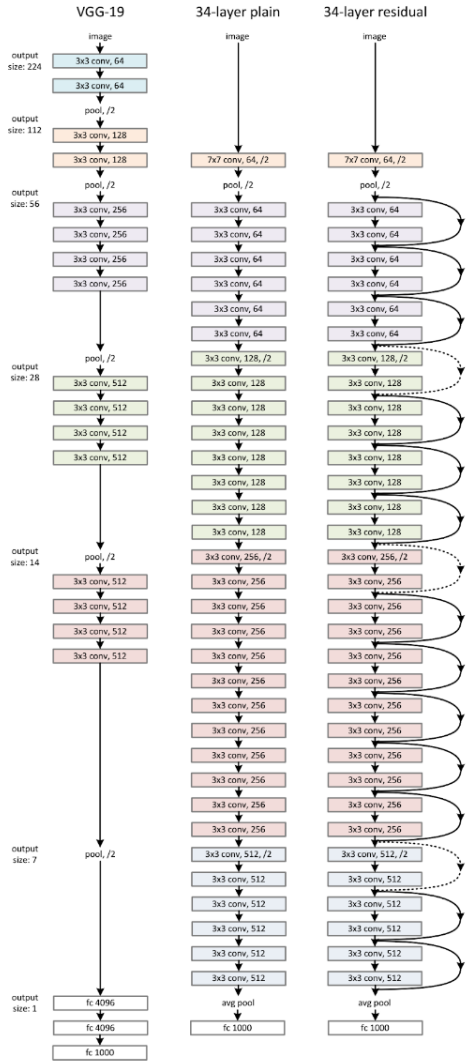
Really, really deep convnets don't train well,
E.g. CIFAR10:



Key idea: introduce "pass through" into each layer

Thus only residual now needs to be learned





Slide credit
Rob Fergus

Residual Networks

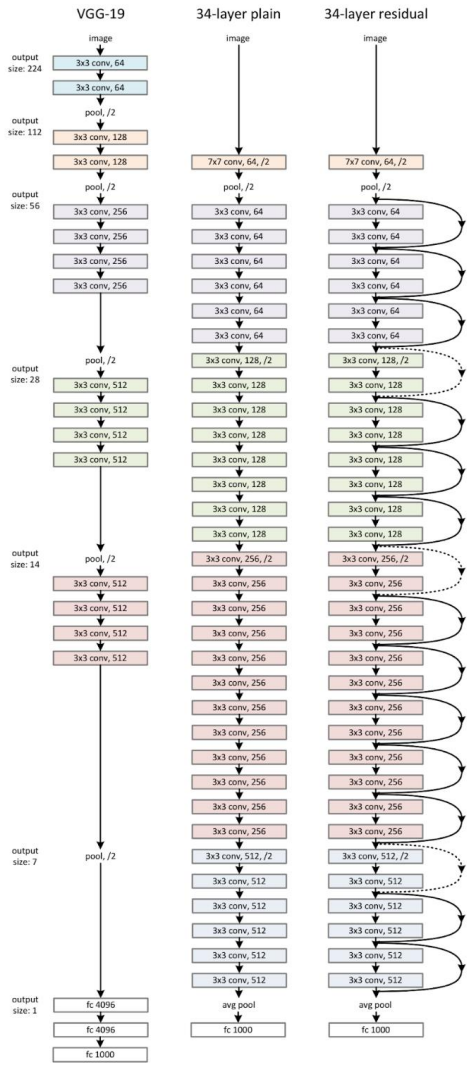
ResNet50 Compared to VGG:

- Superior accuracy in all vision tasks

  5.25% top-5 error vs 7.1%
- Less parameters

  25M vs 138M
- Computational complexity

  3.8B Flops vs 15.3B Flops
- Fully Convolutional until the last layer

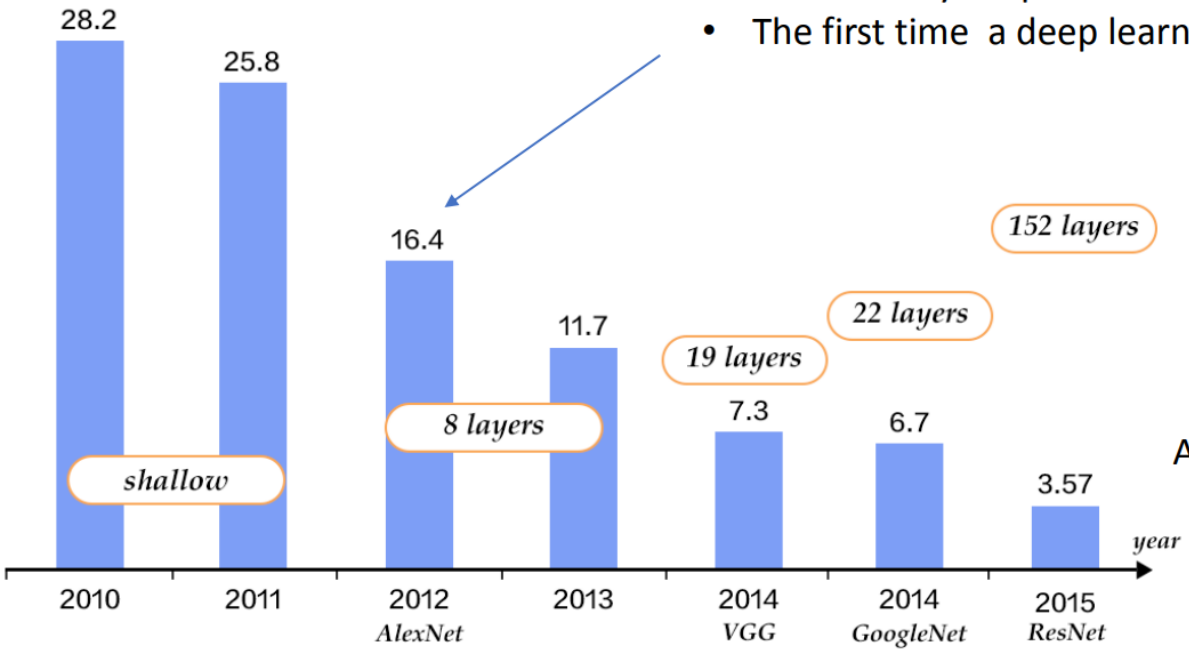


Slide credit: Charles Ollion - Olivier Grisel

Deeper is better

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

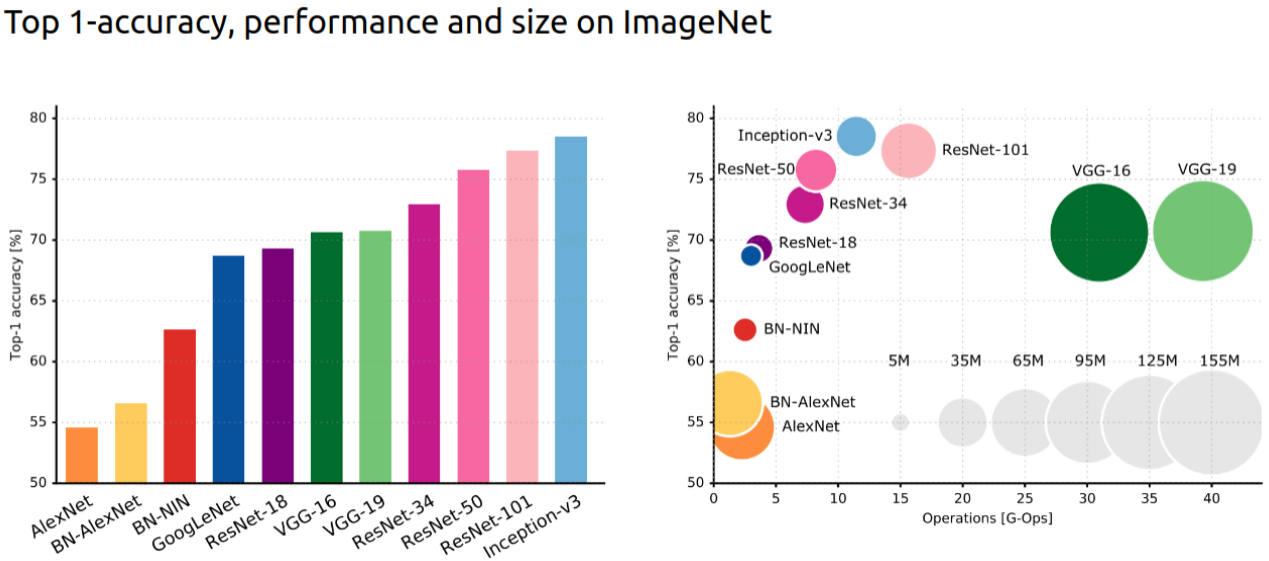
- Dramatically outperformed traditional methods.
- The first time a deep learning method CNN was used.



Average human rate is 5.1%

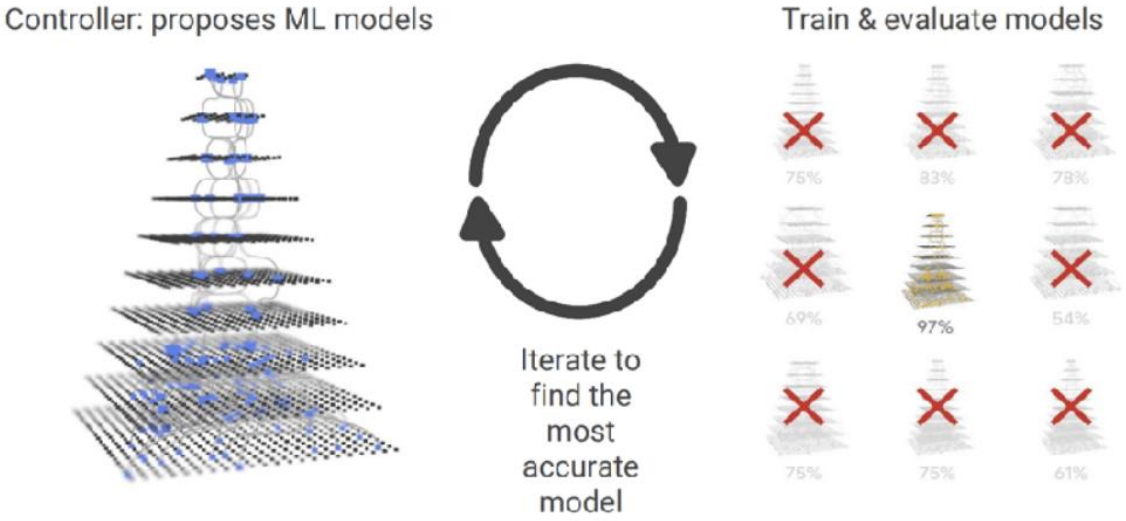
Figure source: http://www.rt-rk.uns.ac.rs/sites/default/files/materijali/predavanja/DL_5.pdf

ImageNet classification



Canziani, Paszke, and Culurciello. "An Analysis of Deep Neural Network Models for Practical Applications." (May 2016).

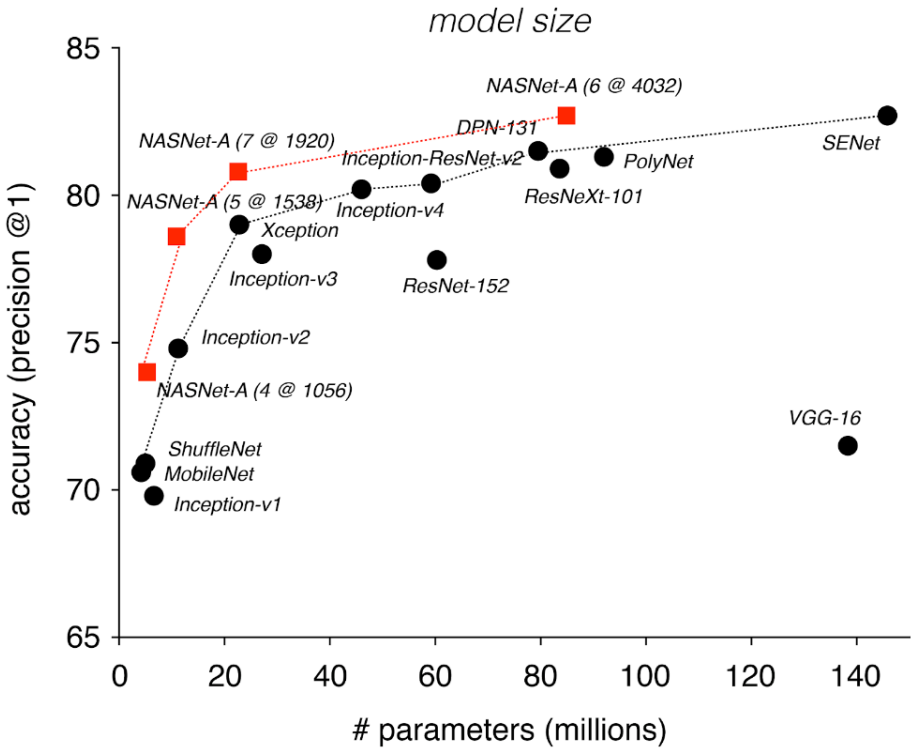
Architecture search



Neural Architecture Search with Reinforcement Learning
B Zoph, Q Le (2016)
Learning Transferable Architectures for Scalable Image Recognition
B Zoph, V Vasudevan, J Shlens, Q Le (2017)
Progressive Neural Architecture Search
C Liu, B Zoph, J Shlens, W Hua, LJ Li, L Fei-Fei, A Yuille, J Huang, K Murphy (2018)
DARTS: Differentiable Architecture Search
H Liu, K Simonyan, Y Yang (2018)

Slide credit: Jon Shlens

Learned architectures surpass human-invented



Learning Transferable Architectures for Scalable Image Recognition
B Zoph, V Vasudevan, J Shlens, Q Le (2017)

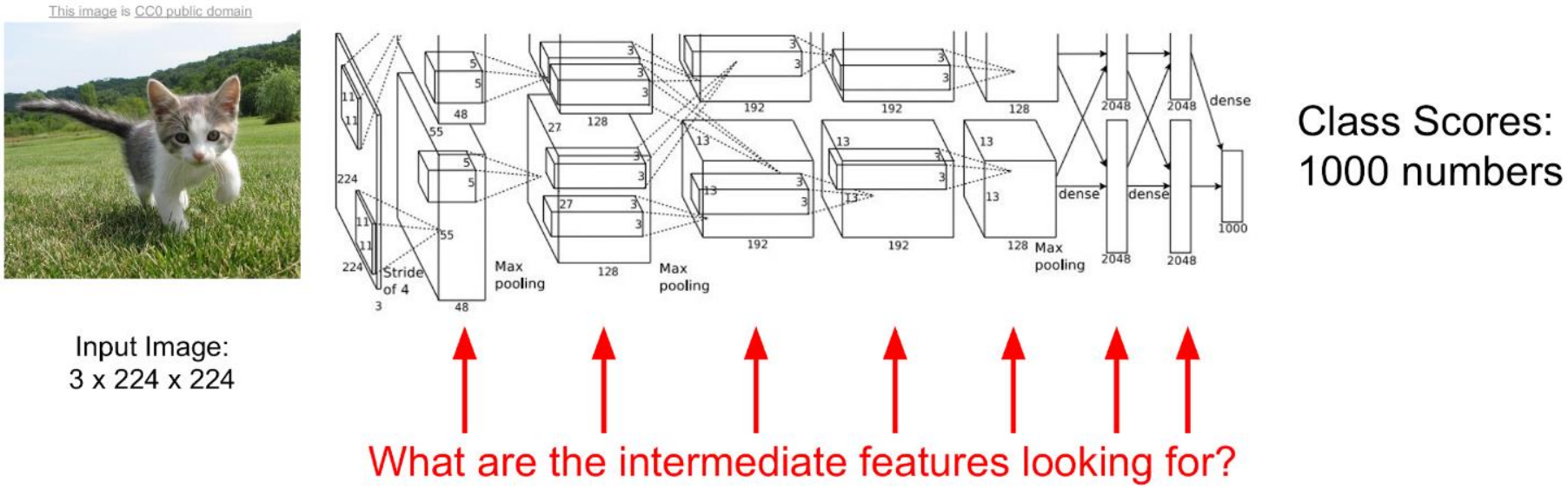
Slide credit: Jon Shlens

Outline

- CNN basics
- Examples of CNN Architectures
- **Visualizing and understanding CNNs**
- Pre-training/Transfer learning

Visualizing and Understanding ConvNets

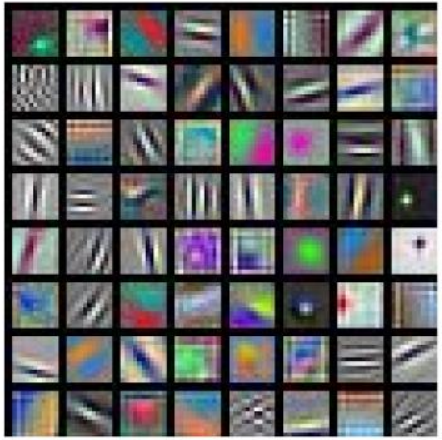
What's going on inside ConvNets?



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

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

First Layer: Visualize filters

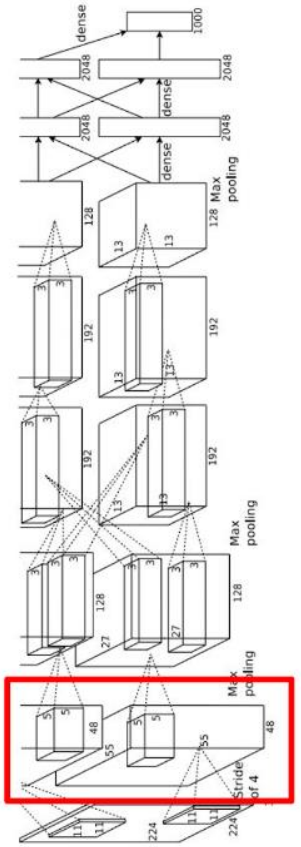


AlexNet:
64 x 3 x 11 x 11

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

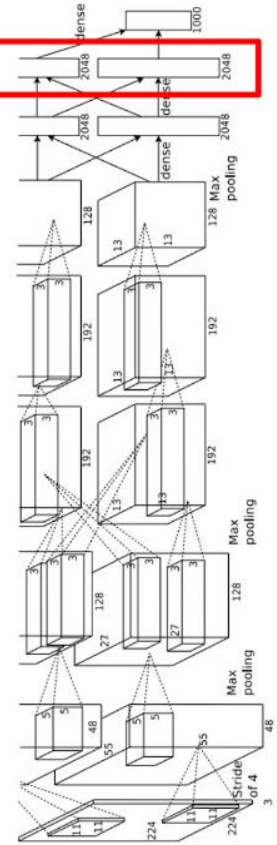


Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Last Layer

FC7 layer



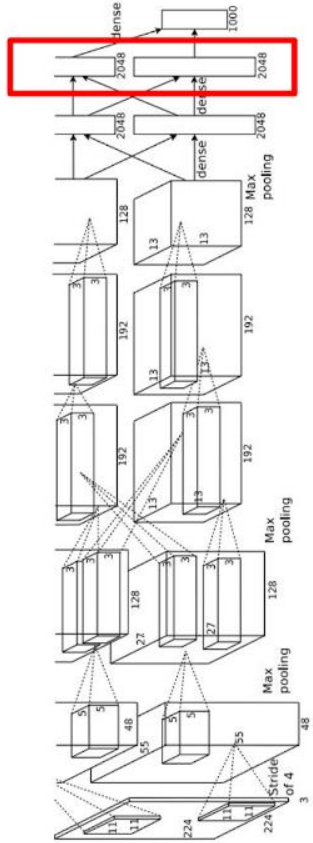
4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

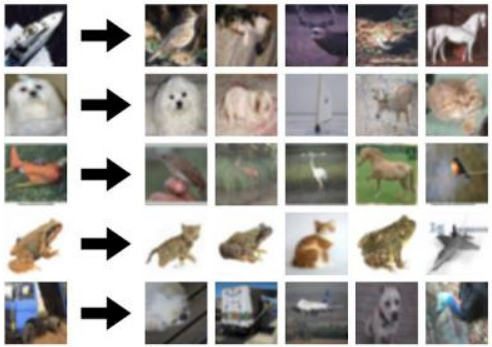
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Last Layer: Nearest Neighbors

4096-dim vector



Recall: Nearest neighbors in pixel space



Test image L2 Nearest neighbors in feature space

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

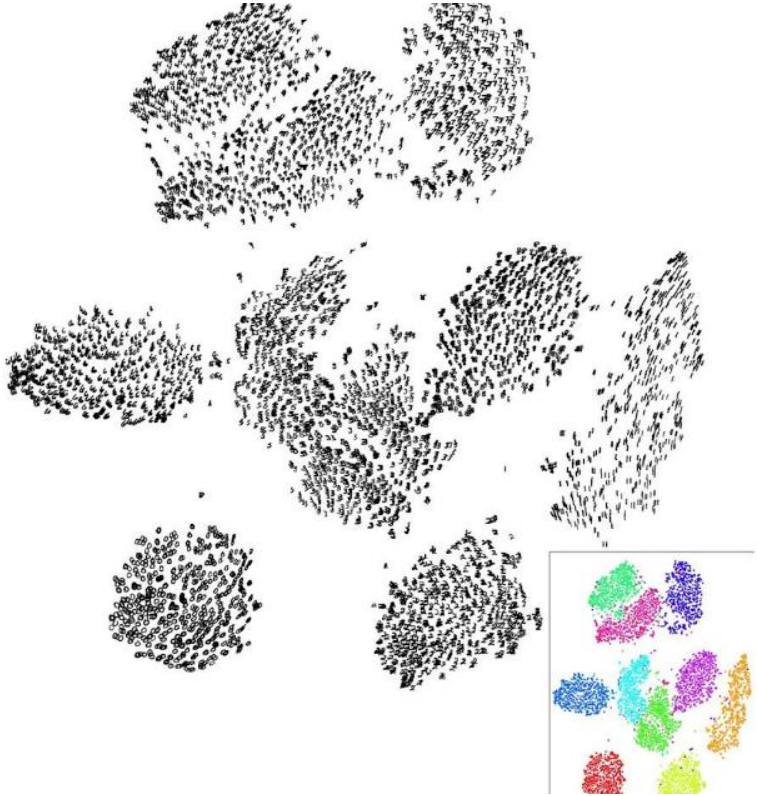
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

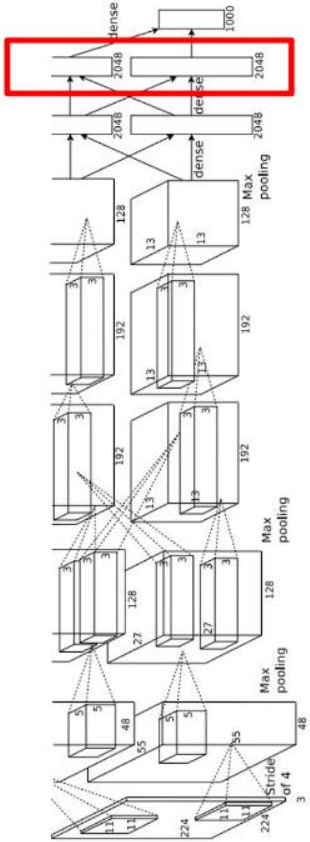
Last Layer: Dimensionality Reduction

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

Simple algorithm: Principle Component Analysis (PCA)

More complex: **t-SNE**

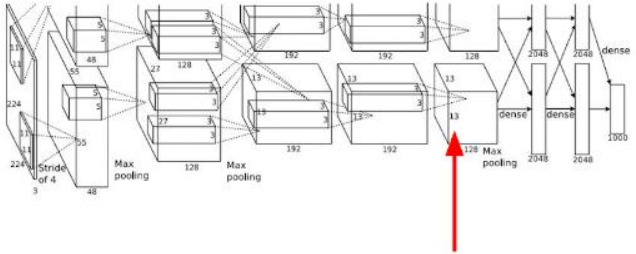




Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Maximally activating patches



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

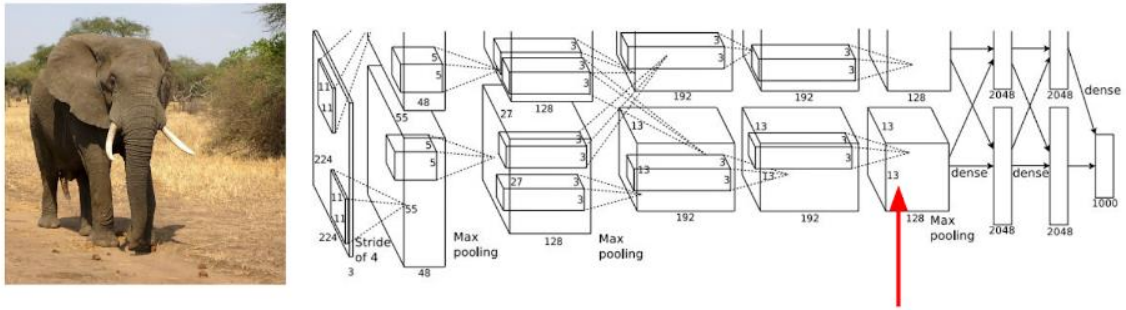
Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015;

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Intermediate features via (guided) backprop



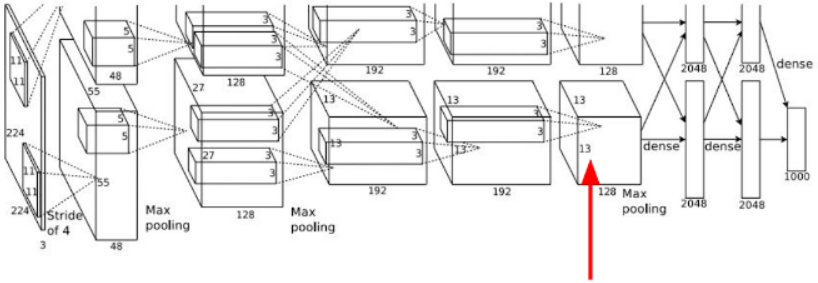
Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

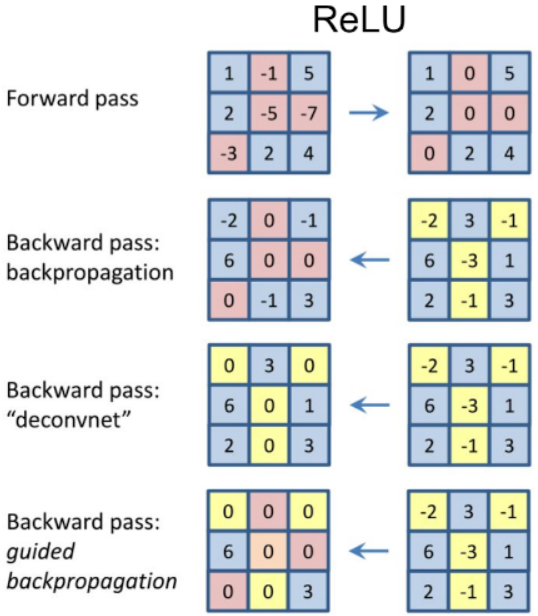
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Intermediate features via (guided) backprop



Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels



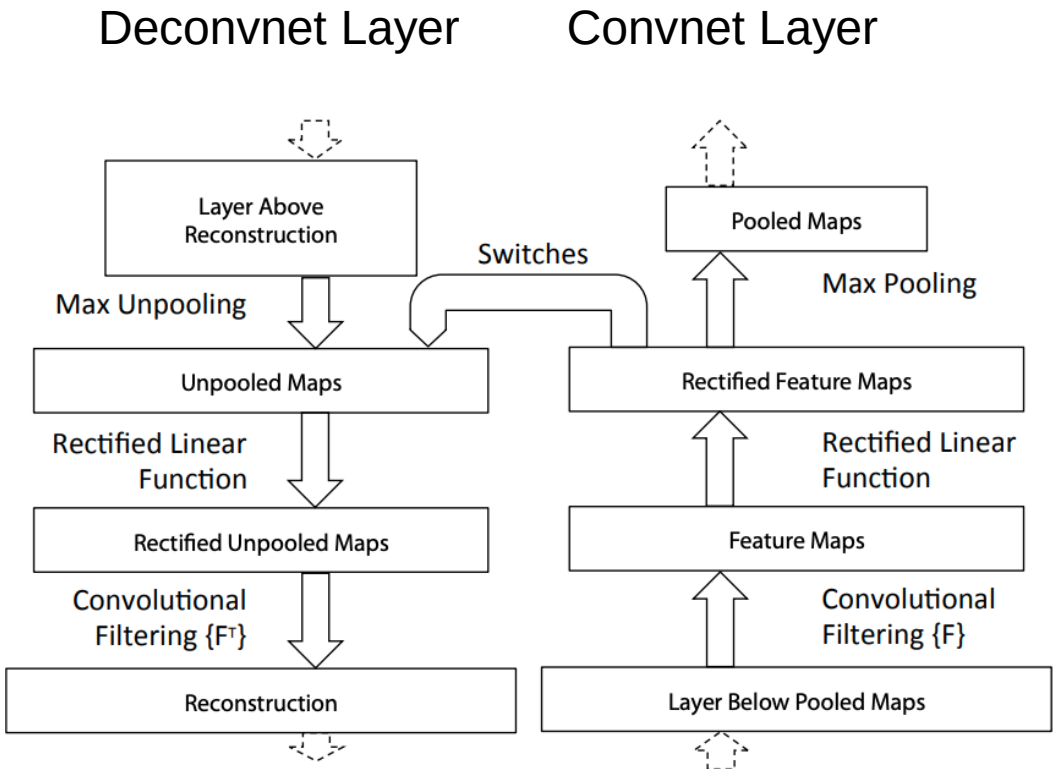
Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

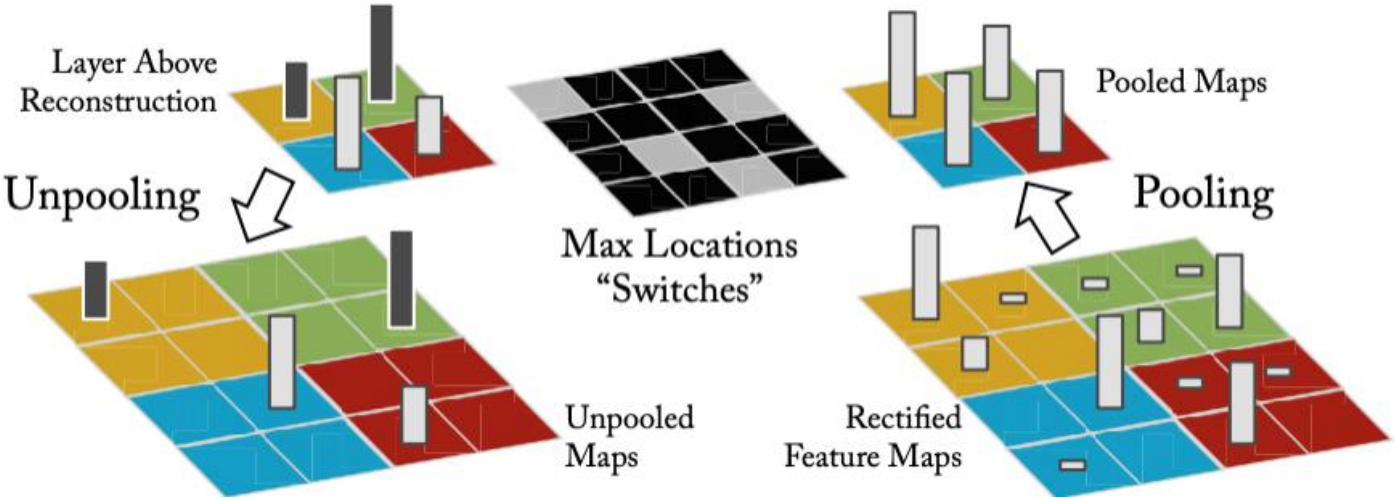
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Visualizing ConvNets



Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler and Rob Fergus

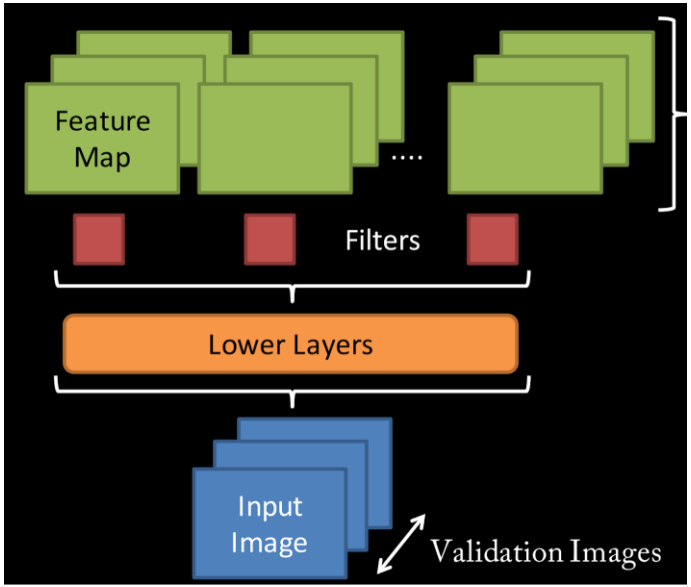
Unpooling operation



Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler and Rob Fergus

Visualizations of Higher Layers

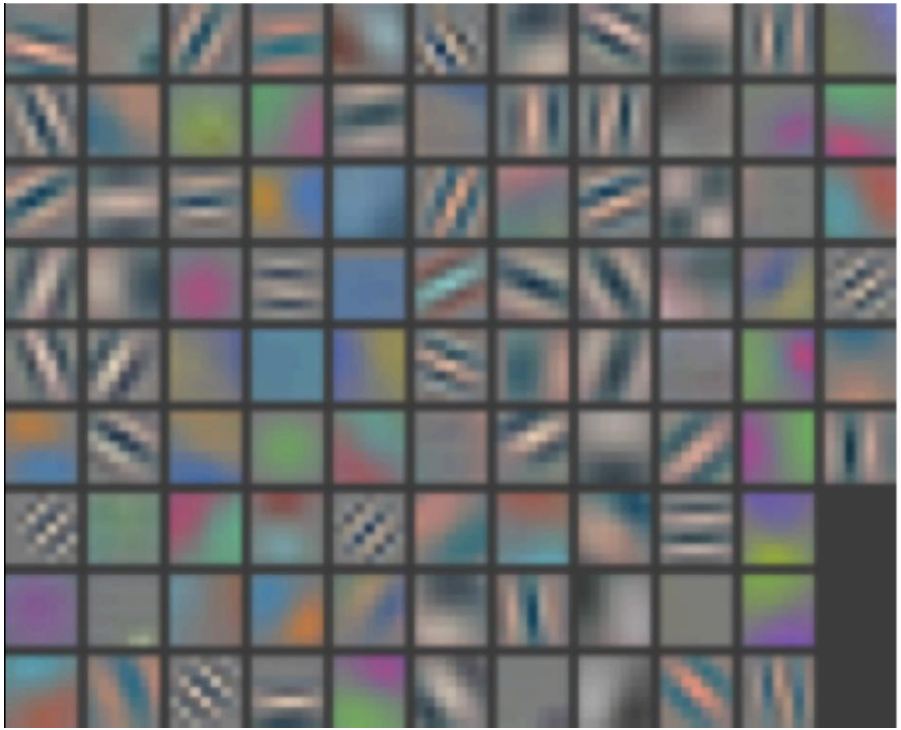
- Use ImageNet 2012 validation set
- Push each image through network



- Take max activation from feature map associated with each filter
- Use Deconvnet to project back to pixel space
- Use pooling “switches” peculiar to that activation

Slide credit: Rob Fergus

Layer 1 filters



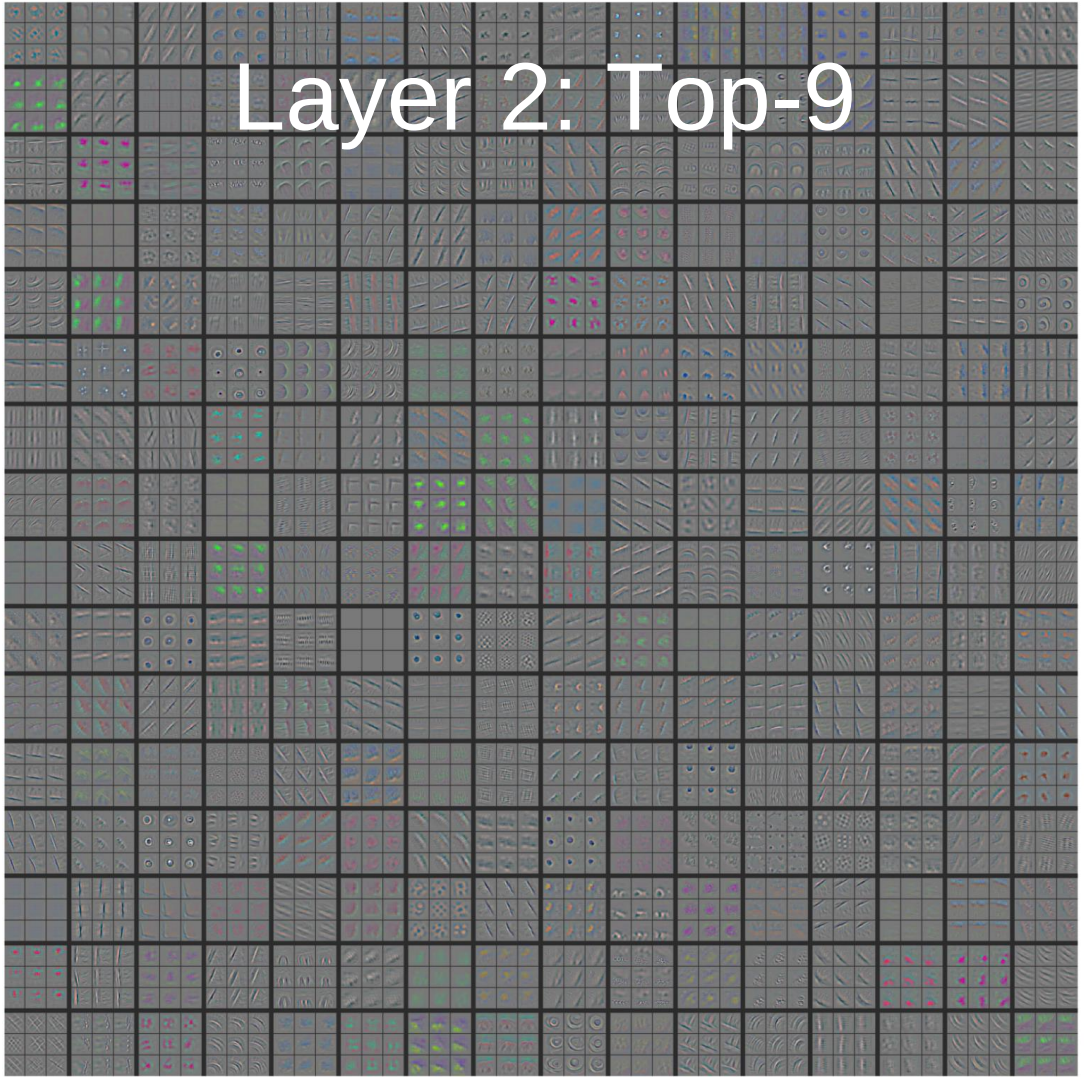
Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler and Rob Fergus

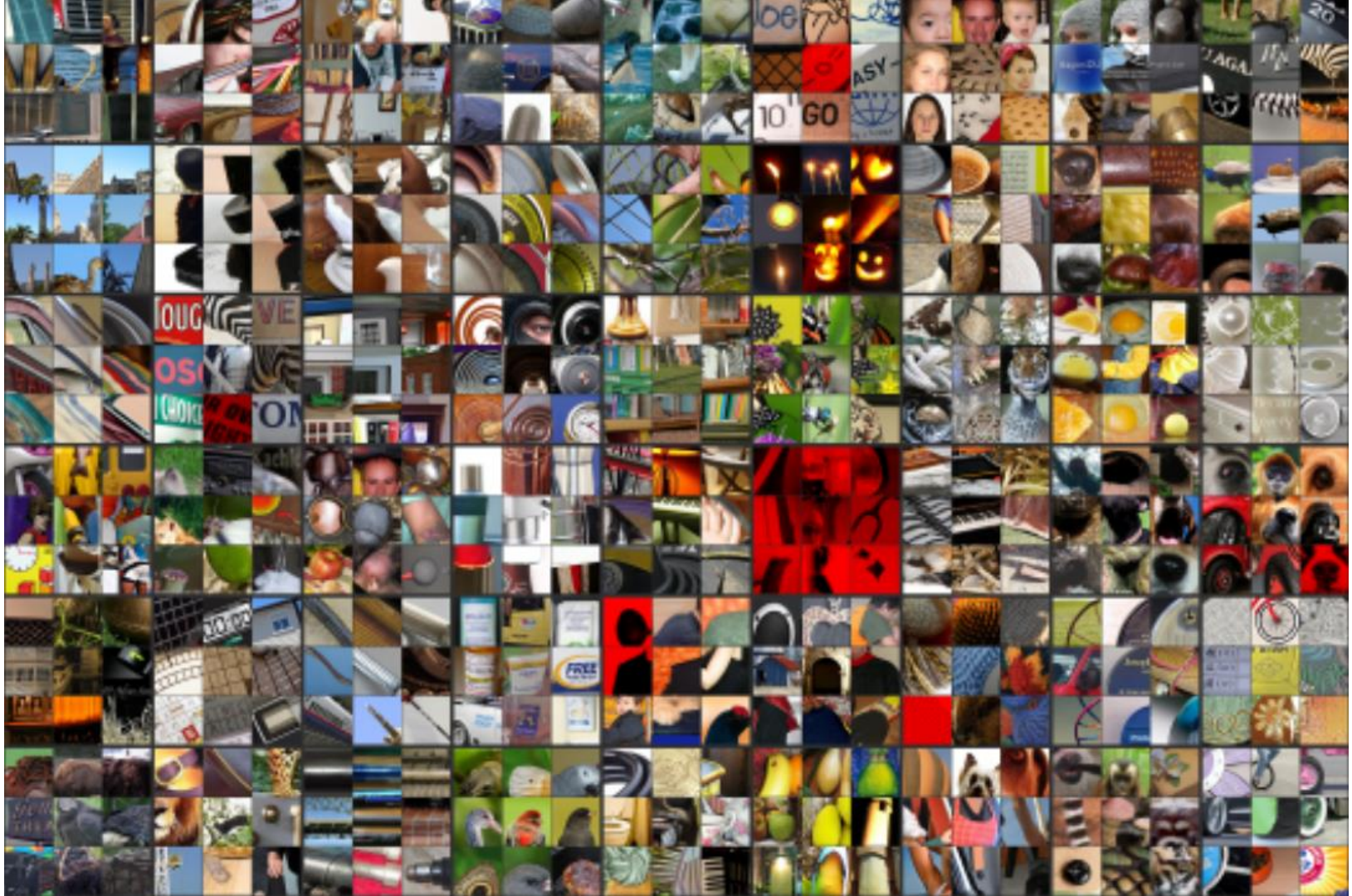
Layer 1: Top-9 Patches

Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler and Rob Fergus

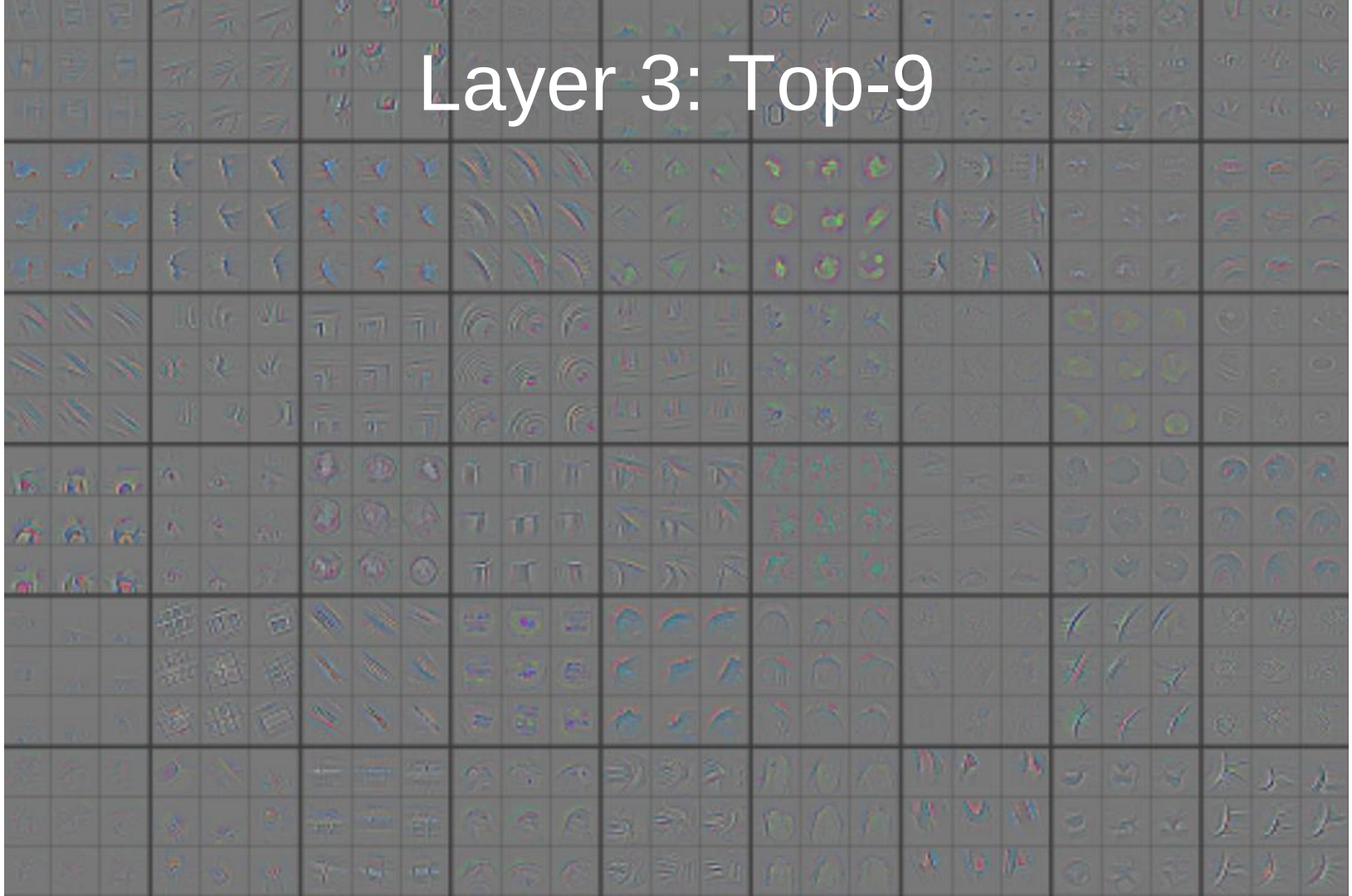
Layer 2: Top-9 Patches

Layer 2: Top-9





Layer 3: Top-9

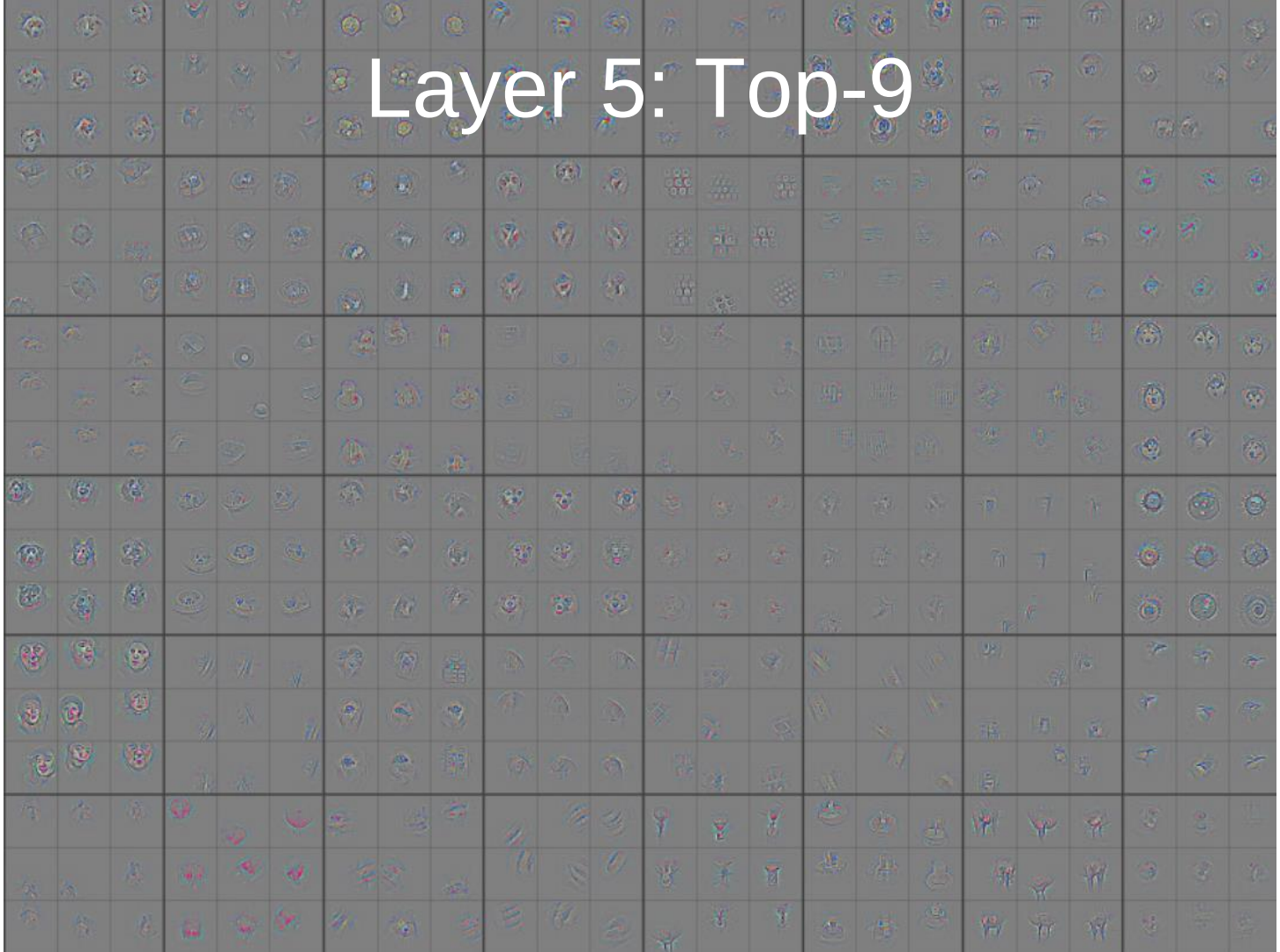


Layer 4: Top-9 Patches

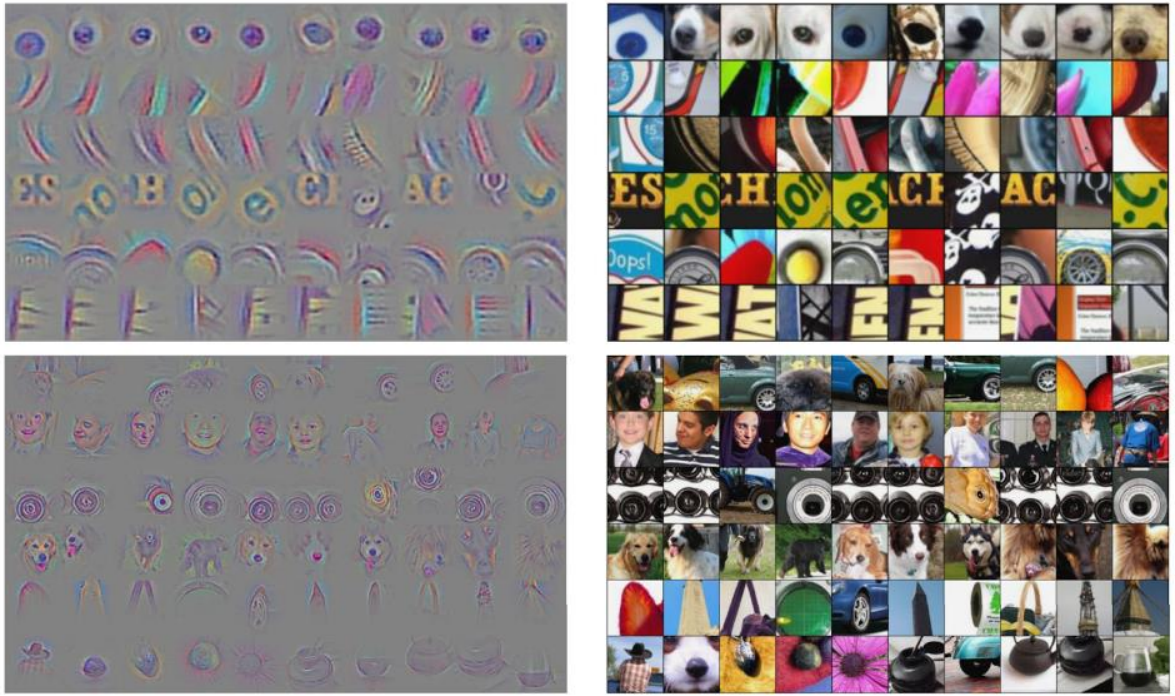
Layer 4: Top-9

Layer 5: Top-9 Patches

Layer 5: Top-9



Intermediate features via (guided) backprop



Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Visualizing CNN features: Gradient Ascent

(Guided) backprop:
Find the part of an image that a neuron responds to

Gradient ascent:
Generate a synthetic image that maximally activates a neuron

$$I^* = \arg\max_I f(I) + R(I)$$

Neuron value

Natural image regularizer

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

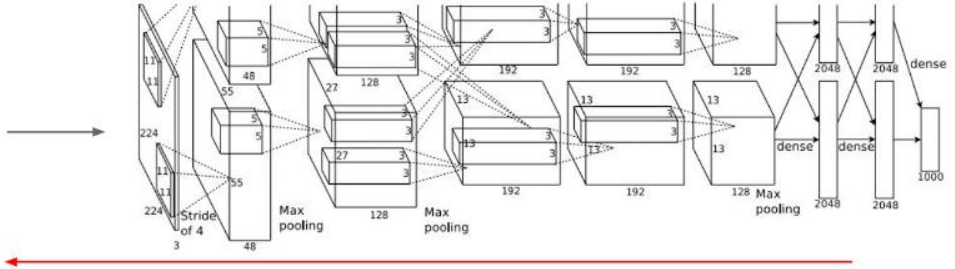
Visualizing CNN features: Gradient Ascent

$$\arg\max_I \boxed{S_c(I)} - \lambda \|I\|_2^2$$

score for class c (before Softmax)

1. Initialize image to zeros

zero image



Repeat:

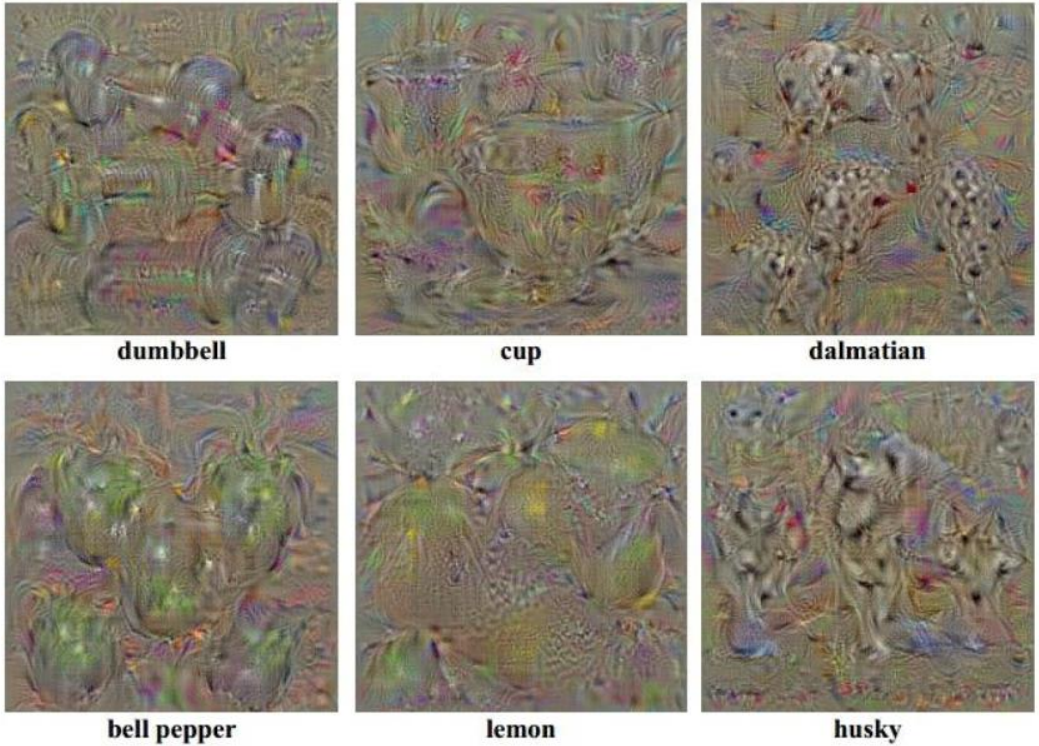
2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Visualizing CNN features: Gradient Ascent

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

Simple regularizer: Penalize L2 norm of generated image



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

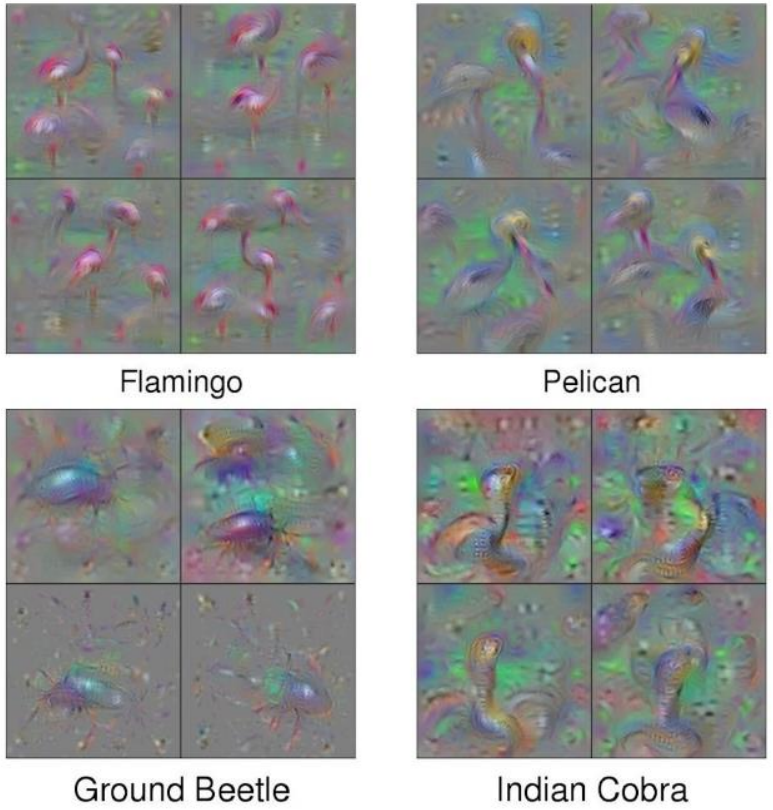
Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Visualizing CNN features: Gradient Ascent

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

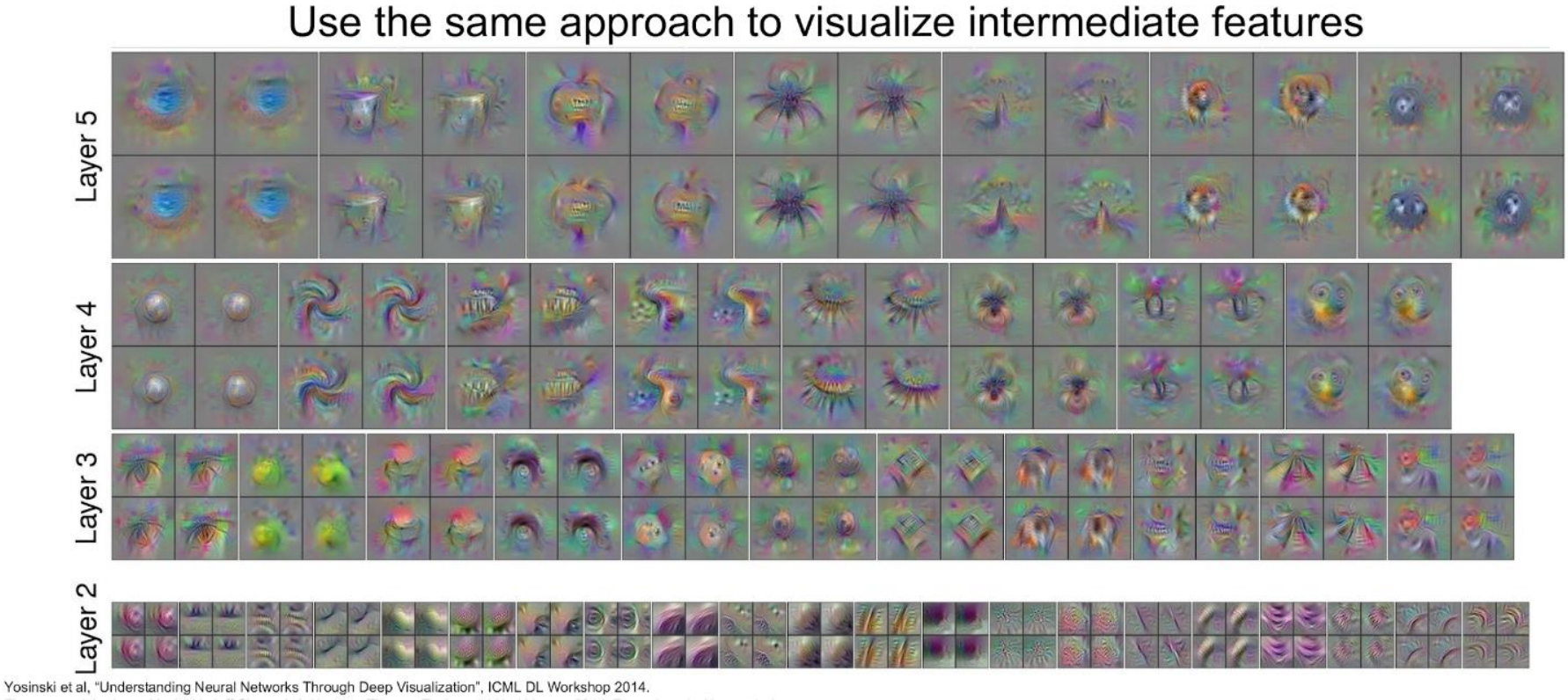
(1) Gaussian blur image
(2) Clip pixels with small values to 0
(3) Clip pixels with small gradients to 0



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Visualizing CNN features: Gradient Ascent

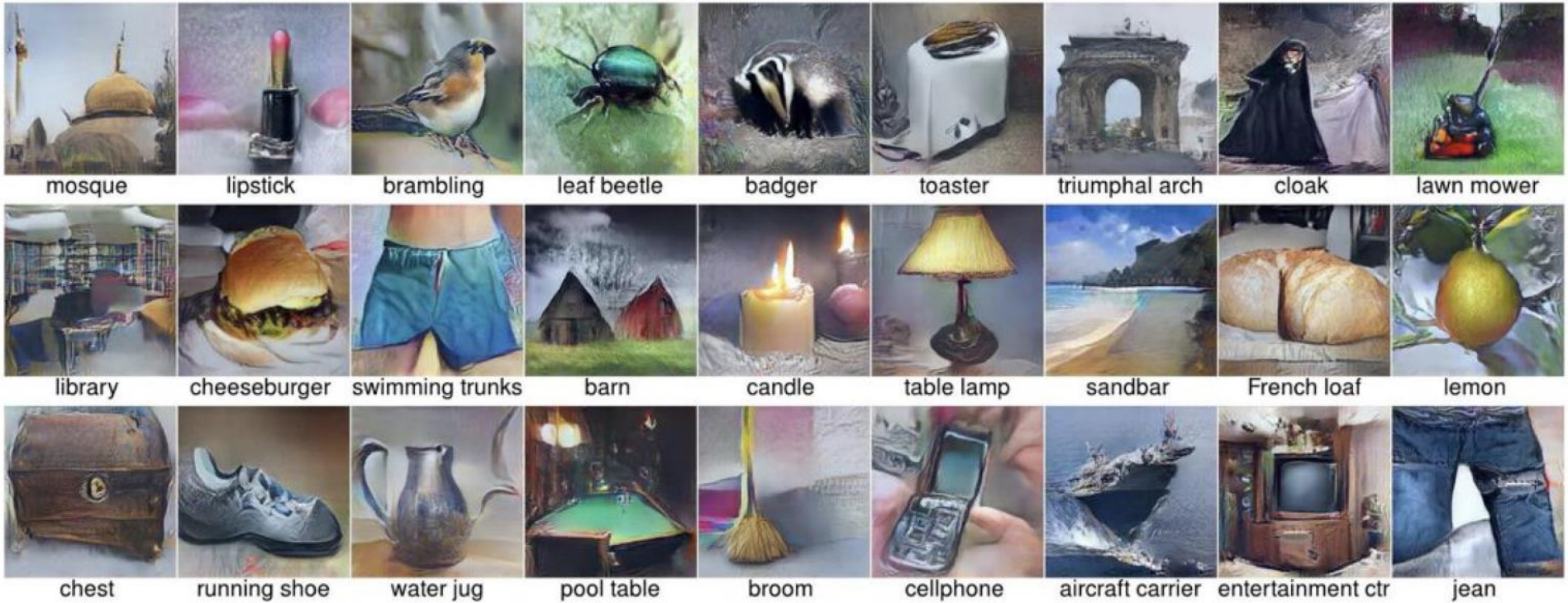


Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Visualizing CNN features: Gradient Ascent

Optimize in FC6 latent space instead of pixel space:



Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Outline

- CNN basics
- Examples of CNN Architectures
- Visualizing and understanding CNNs
- **Pre-training/Transfer learning**

Pre-trained models

- Training a model on ImageNet from scratch takes **days or weeks**.
- Many models trained on ImageNet and their weights are publicly available!

Transfer learning

- Use pre-trained weights, remove last layers to compute representations of images
- Train a classification model from these features on a new classification task
- The network is used as a generic feature extractor
- Better than handcrafted feature extraction on natural images

Pre-trained models

- Training a model on ImageNet from scratch takes **days or weeks**.
- Many models trained on ImageNet and their weights are publicly available!

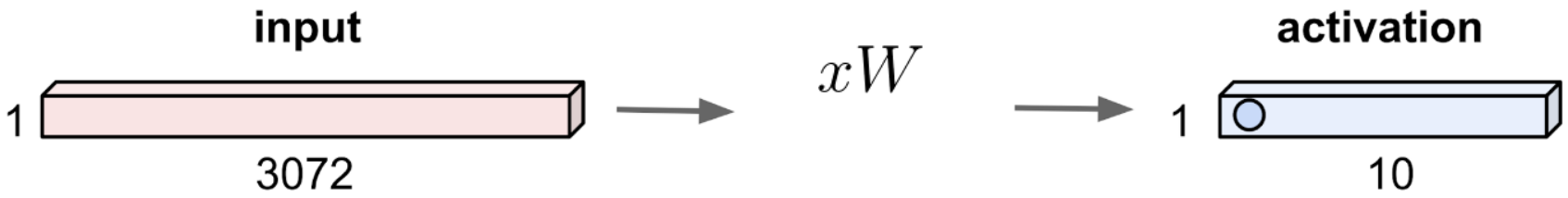
Fine-tuning

Retraining the (some) parameters of the network (given enough data)

- Truncate the last layer(s) of the pre-trained network
- Freeze the remaining layers weights
- Add a (linear) classifier on top and train it for a few epochs
- Then fine-tune the whole network or the few deepest layers
- Use a smaller learning rate when fine tuning

Fully Connected Layer

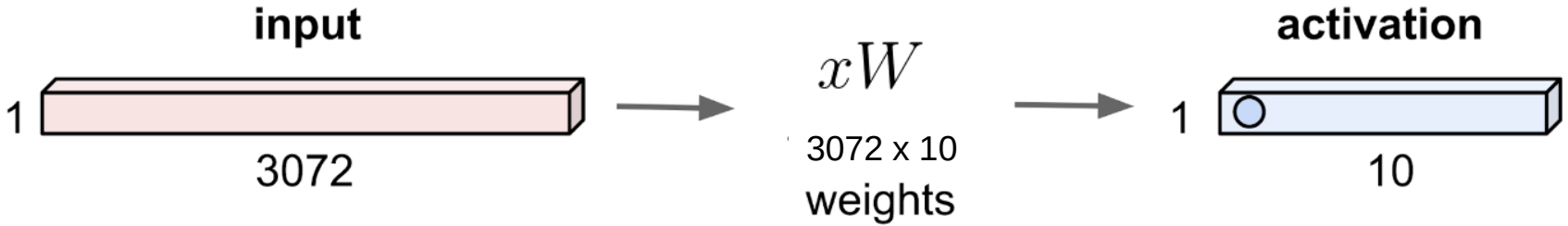
32x32x3 image -> stretch to 3072 x 1



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

Fully Connected Layer

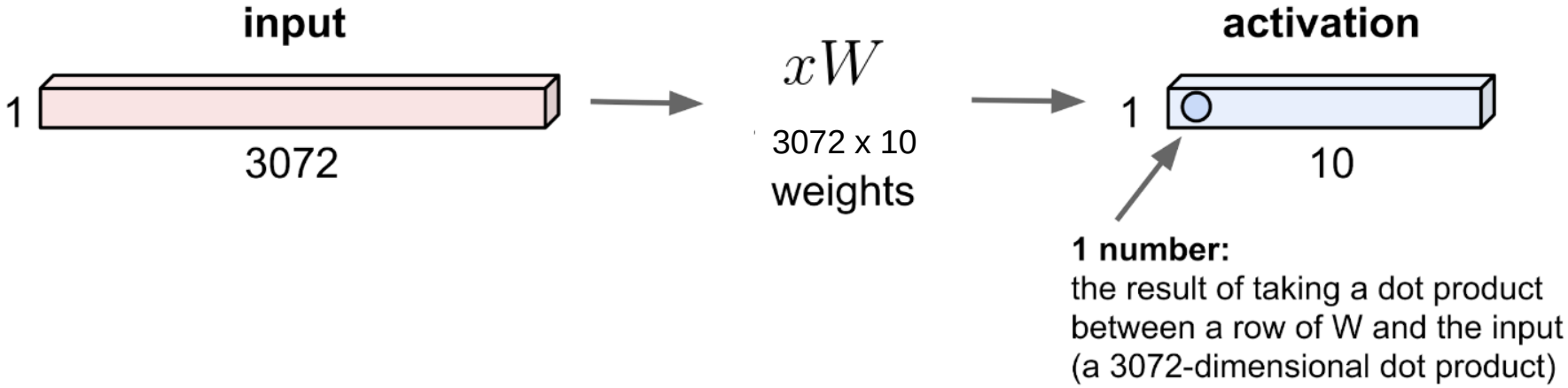
32x32x3 image -> stretch to 3072 x 1



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

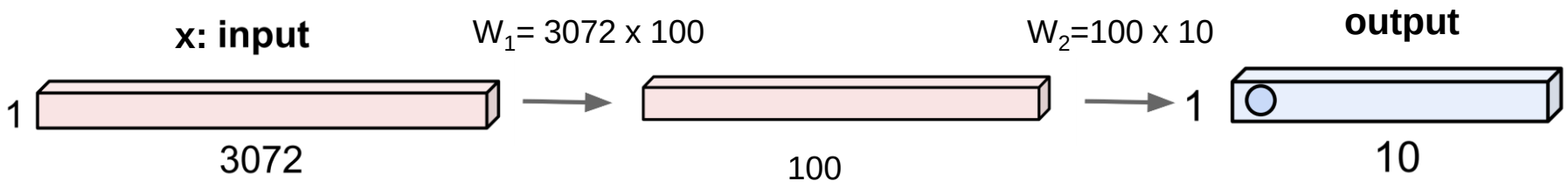
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung

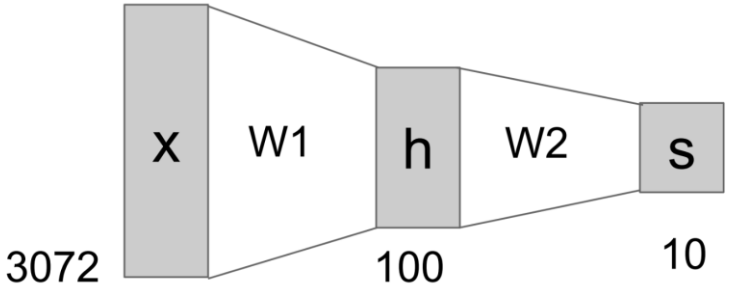
Neural Network with 1 hidden layer (with relu)



Shorthand notation

$$f = \max(0, xW_1)W_2$$

Alternative diagram:



Deeper models: We can add more layers, etc.

Slide credit: Fei-Fei Li, Justin Johnson, Serena Yeung