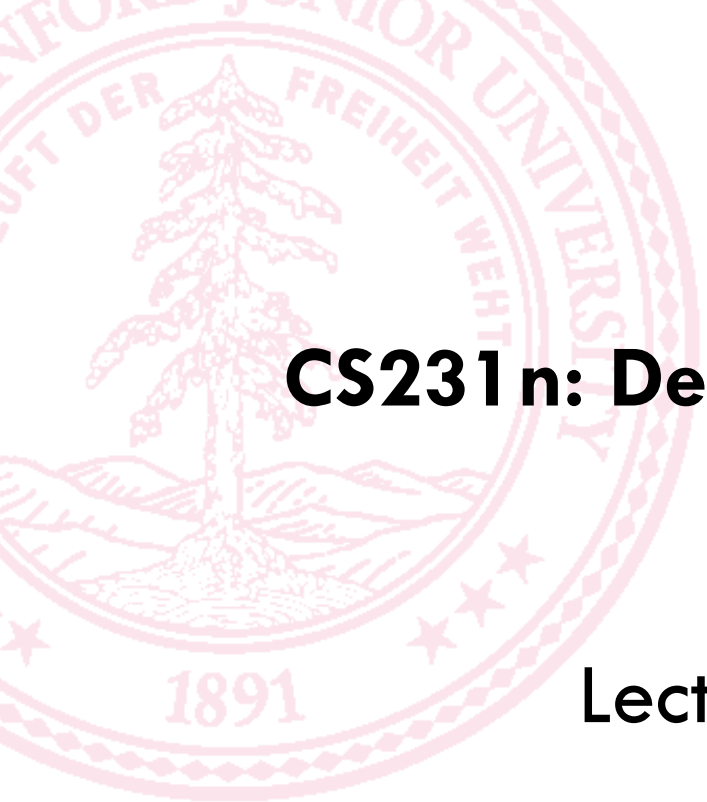


CS231n: Deep Learning for Computer Vision

Lecture 1: Introduction

Computer Vision is everywhere!



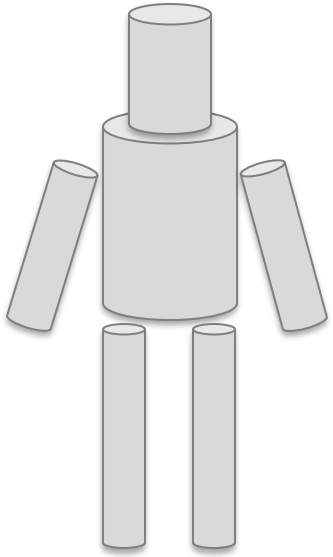




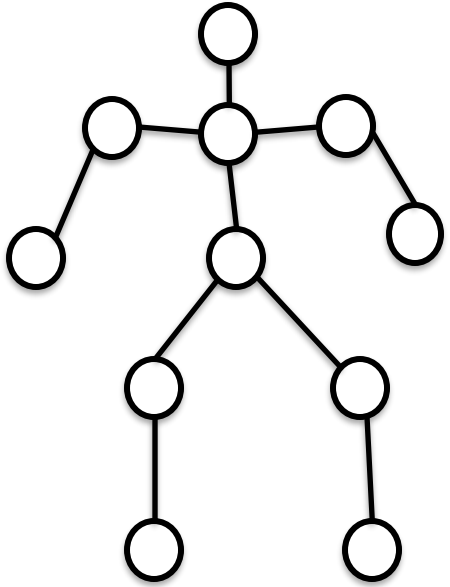



Where did we come from?

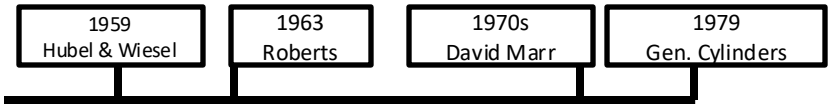
Recognition via Parts (1970s)



Generalized Cylinders,
Brooks and Binford,
1979

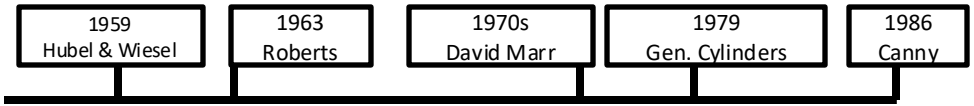


Pictorial Structures,
Fischler and Elshlager, 1973



Slide inspiration: Justin Johnson

Recognition via Edge Detection (1980s)

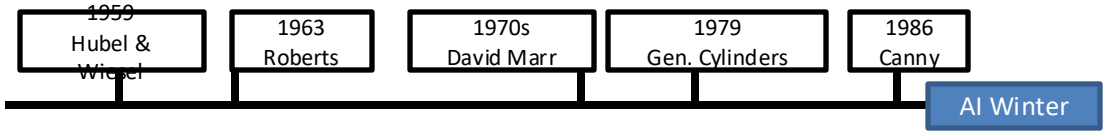


John Canny, 1986
David Lowe, 1987

Slide inspiration: Justin Johnson

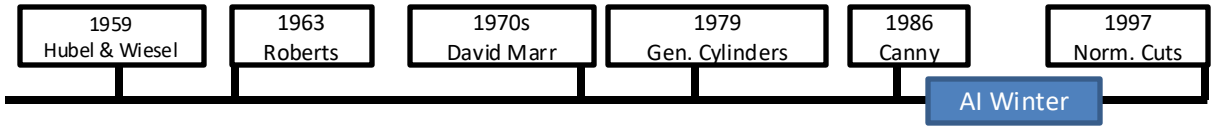
Arriving at an “AI winter”

- Enthusiasm (and funding!) for AI research dwindled
- ”Expert Systems” failed to deliver on their promises
- But subfields of AI continues to grow
    - Computer vision, NLP, robotics, compbio, etc.



Slide inspiration: Justin Johnson

Recognition via Grouping (1990s)



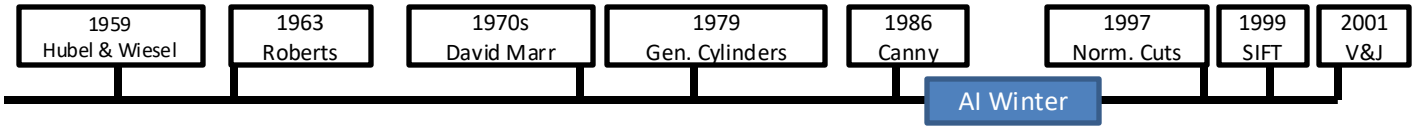
Normalized Cuts, Shi and Malik, 1997

Slide inspiration: Justin Johnson

Face Detection

Viola and Jones, 2001

One of the first successful applications of machine learning to vision



Slide inspiration: Justin Johnson

Caltech 101 images

PASCAL Visual Object Challenge

1959 Hubel & Wiesel
1963 Roberts
1970s David Marr
1979 Gen. Cylinders
1986 Canny
AI Winter
1997 Norm. Cuts
1999 SIFT
2001 V&J
2004, 2007 Caltech101; PASCAL

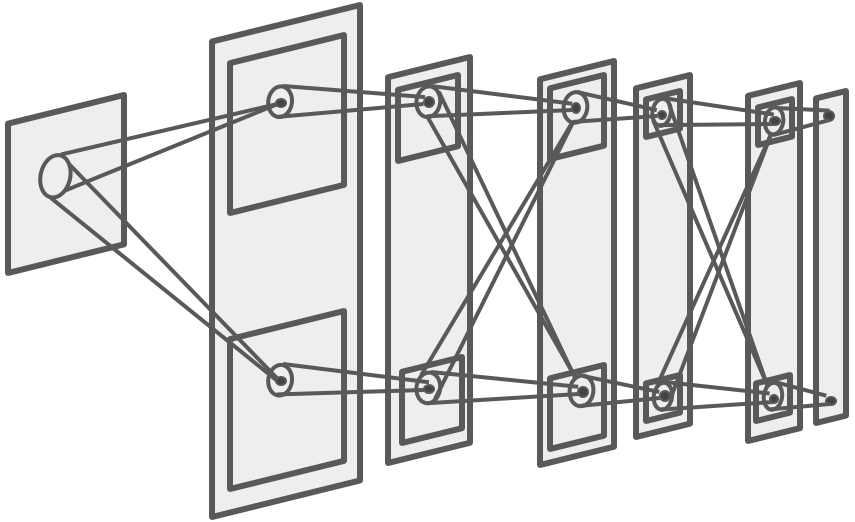
Slide inspiration: Justin Johnson

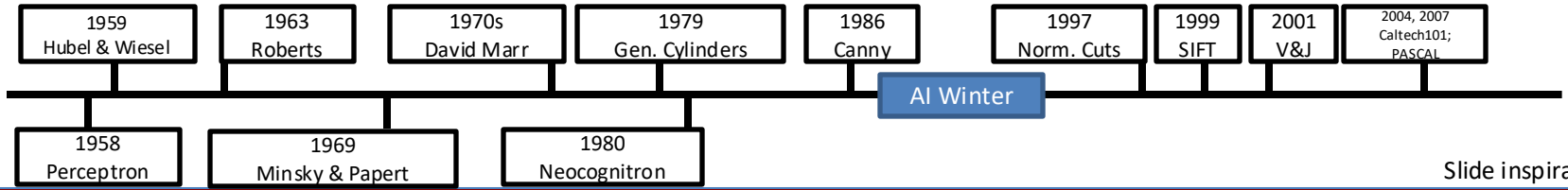
Neocognitron: Fukushima, 1980

Computational model the visual system, directly inspired by Hubel and Wiesel's hierarchy of complex and simple cells

Interleaved simple cells (convolution) and complex cells (pooling)

No practical training algorithm





Slide inspiration: Justin Johnson

Backprop: Rumelhart, Hinton, and Williams, 1986

Introduced backpropagation for computing gradients in neural networks

Successfully trained perceptrons with multiple layers

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}$$

recognizable math

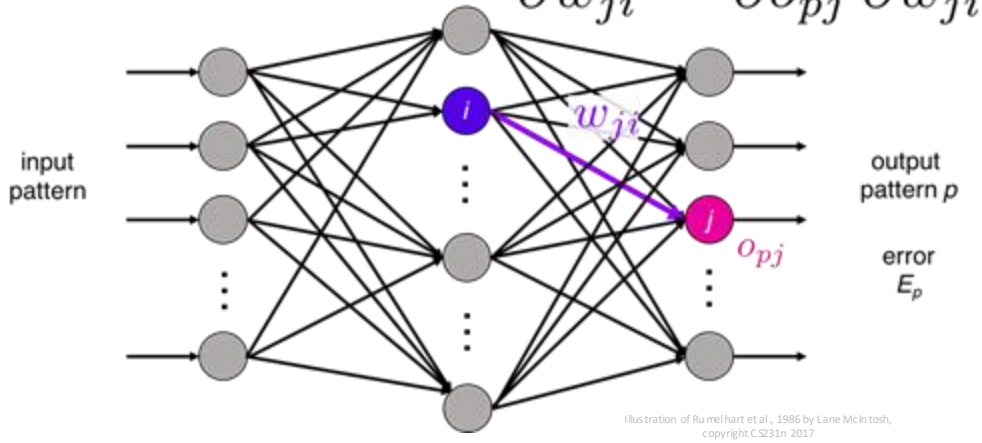
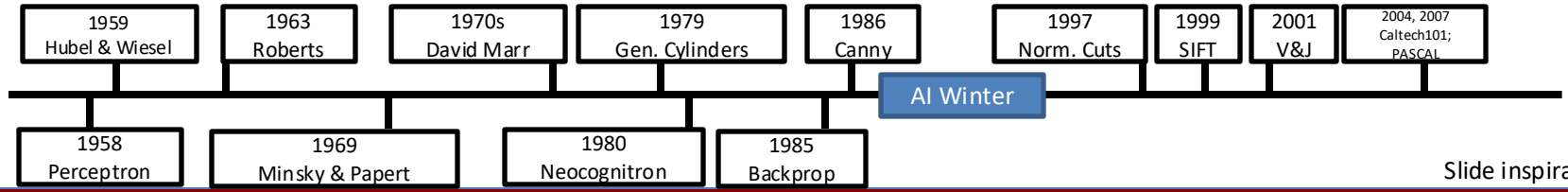
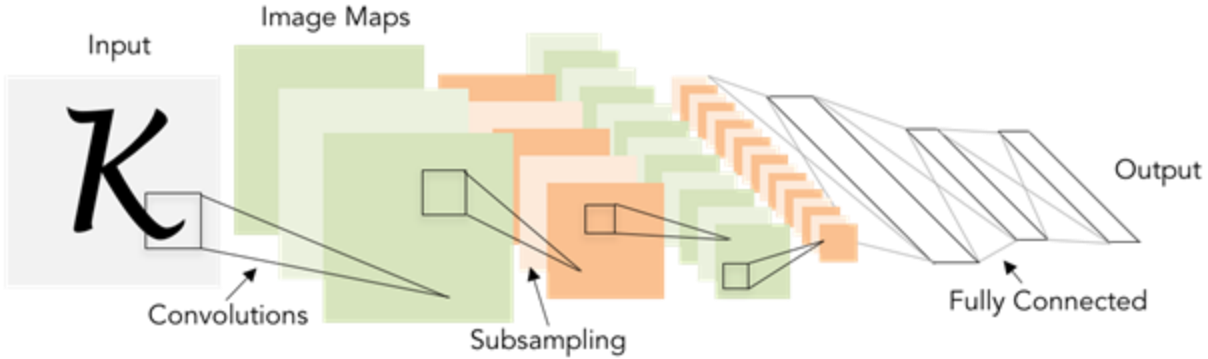


Illustration of Rumelhart et al., 1986 by Lane McIntosh, copyright CS231n 2017

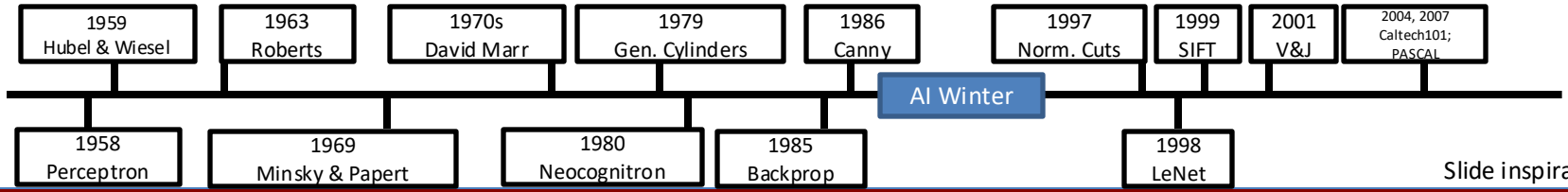


Slide inspiration: Justin Johnson

Convolutional Networks: LeCun et al, 1998



Applied backprop algorithm to a Neocognitron-like architecture
Learned to recognize handwritten digits
Was deployed in a commercial system by NEC, processed handwritten checks
Very similar to our modern convolutional networks!



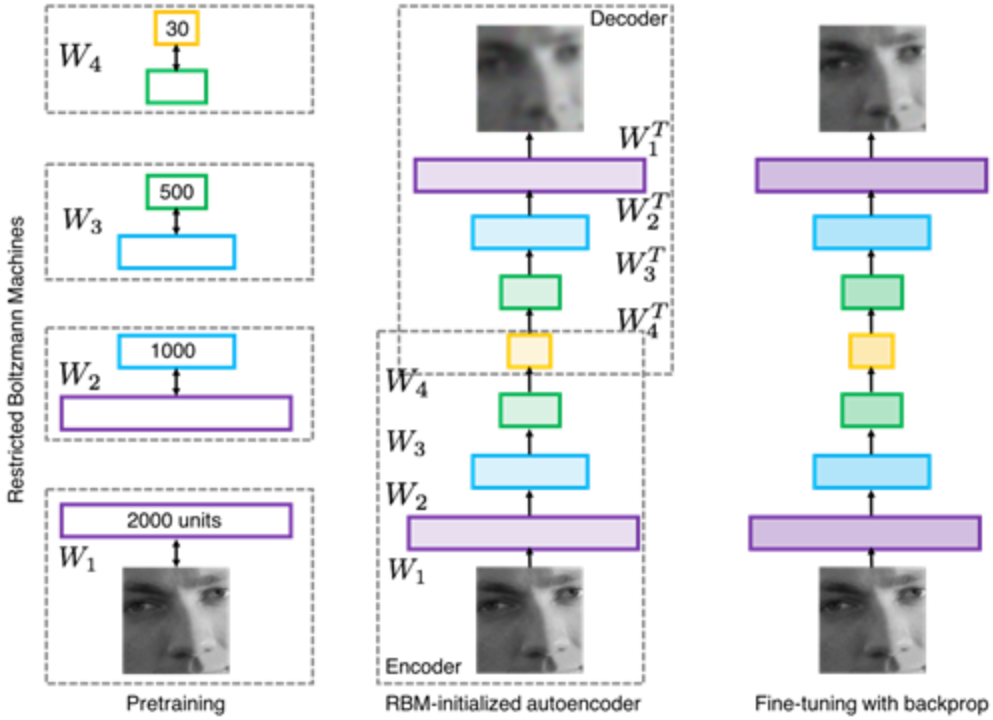
Slide inspiration: Justin Johnson

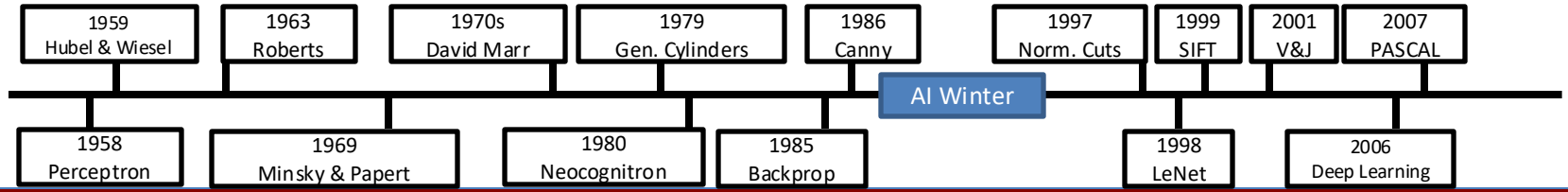
2000s: “Deep Learning”

People tried to train neural networks that were deeper and deeper

Not a mainstream research topic at this time

Hinton and Salakhutdinov, 2006
Bengio et al, 2007
Lee et al, 2009
Glorot and Bengio, 2010





Slide inspiration: Justin Johnson

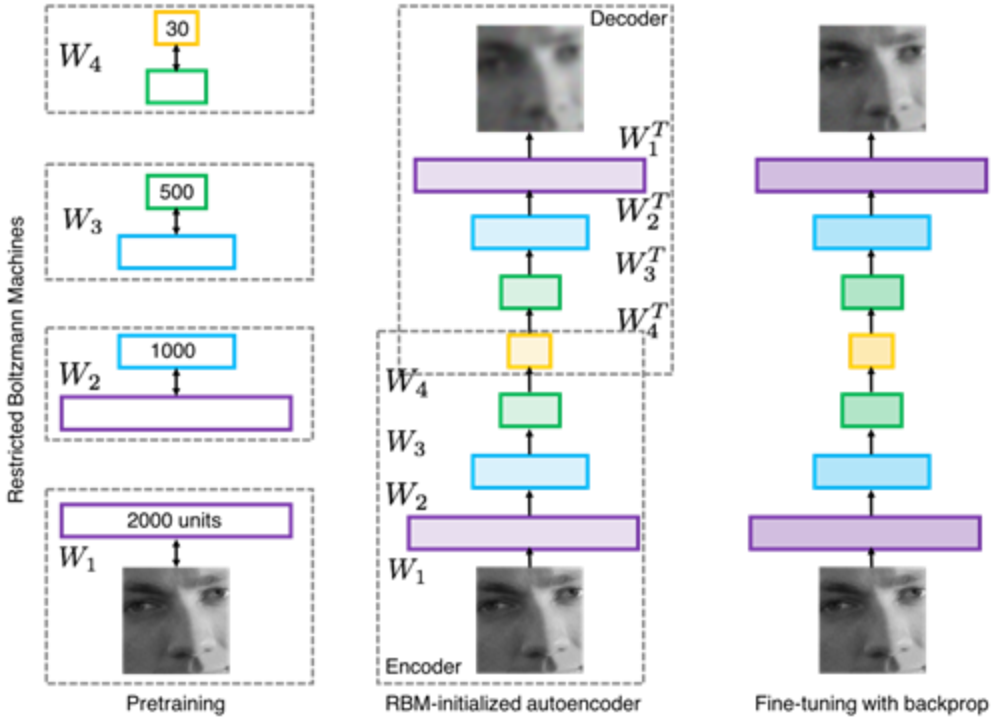
2000s: “Deep Learning”

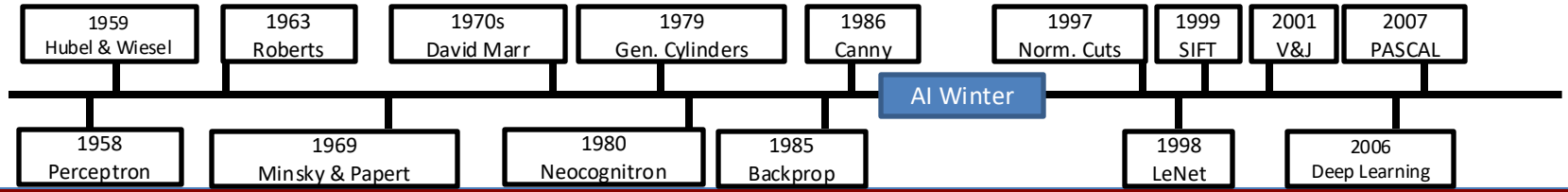
People tried to train neural networks that were deeper and deeper

Not a mainstream research topic at this time

No good dataset to work on

Hinton and Salakhutdinov, 2006
Bengio et al, 2007
Lee et al, 2009
Glorot and Bengio, 2010





Slide inspiration: Justin Johnson

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

1,000 object classes

1,431,167 images

Output:

Scale

T-shirt

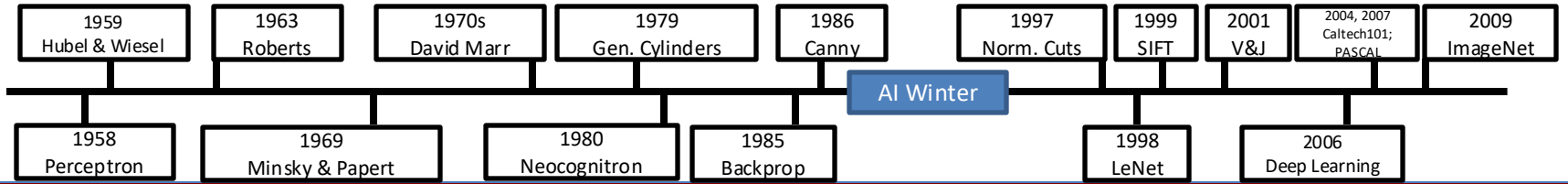
Steel drum

Drumstick

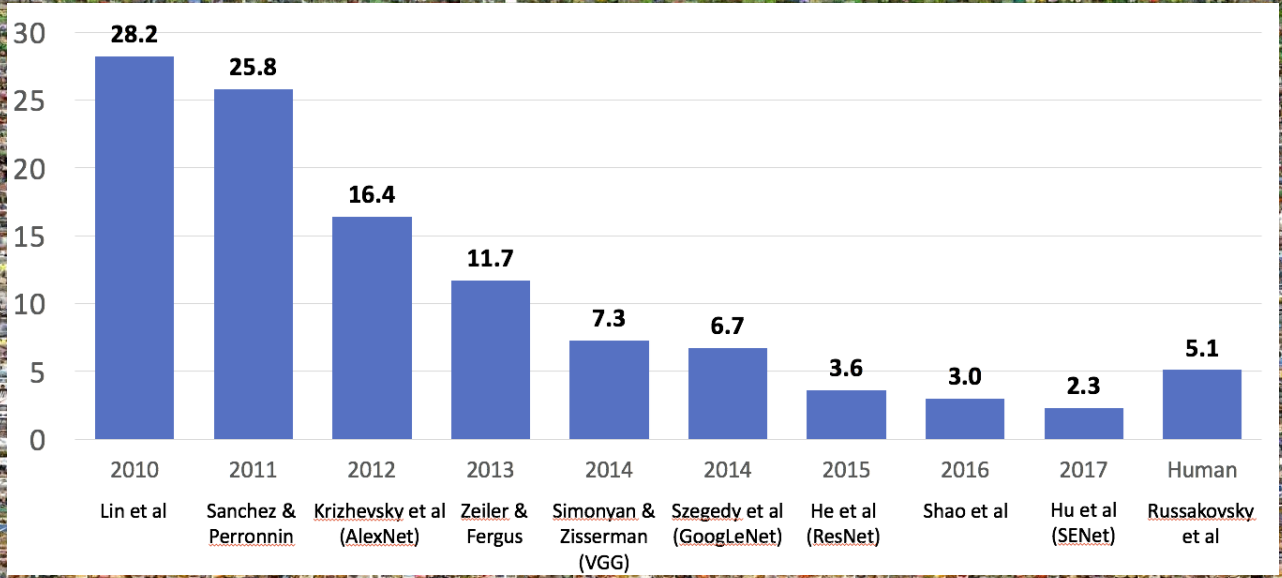
Mud turtle

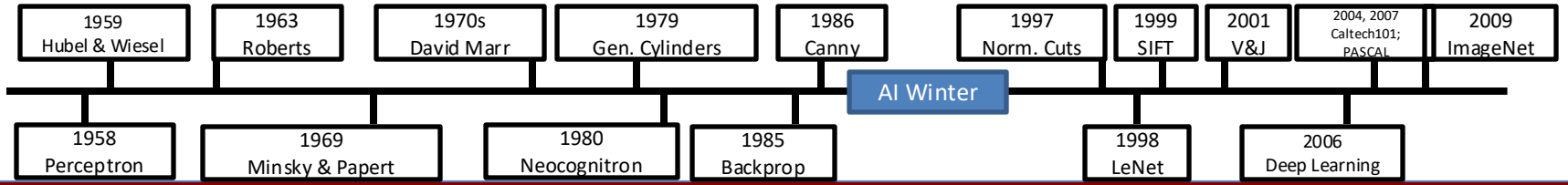
Deng et al, 2009

Russakovsky et al. IJCV 2015

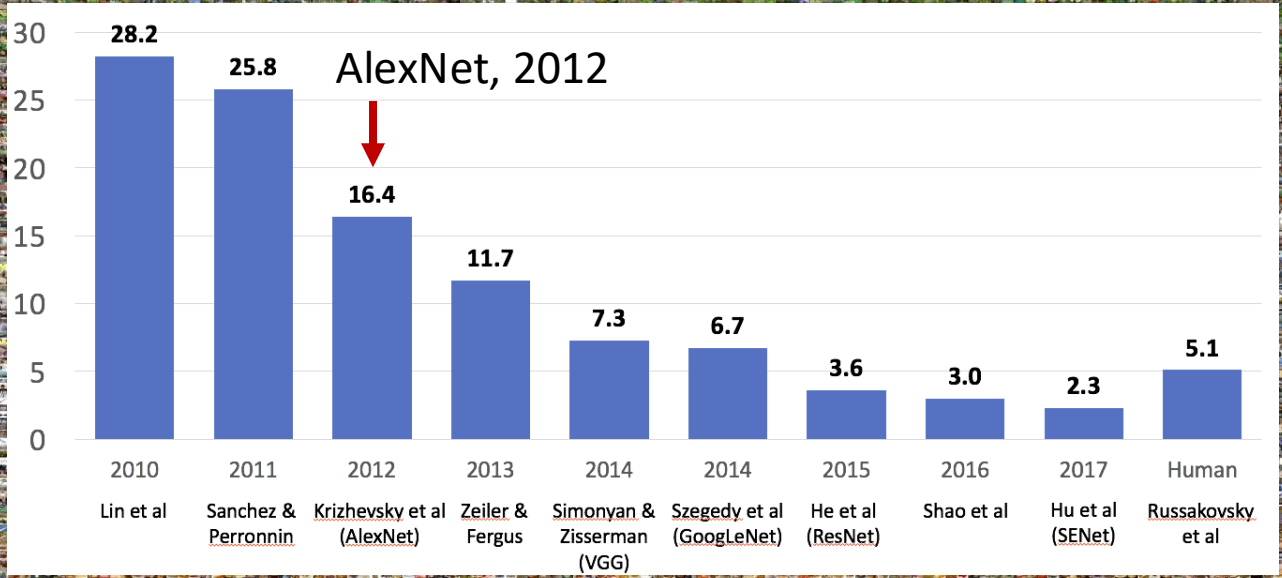


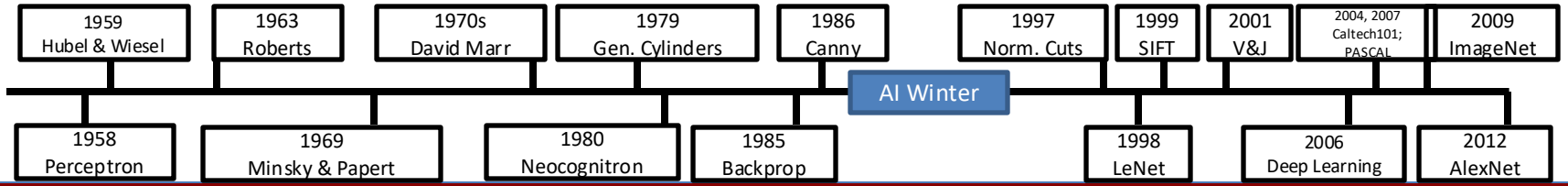
IMAGENET Large Scale Visual Recognition Challenge



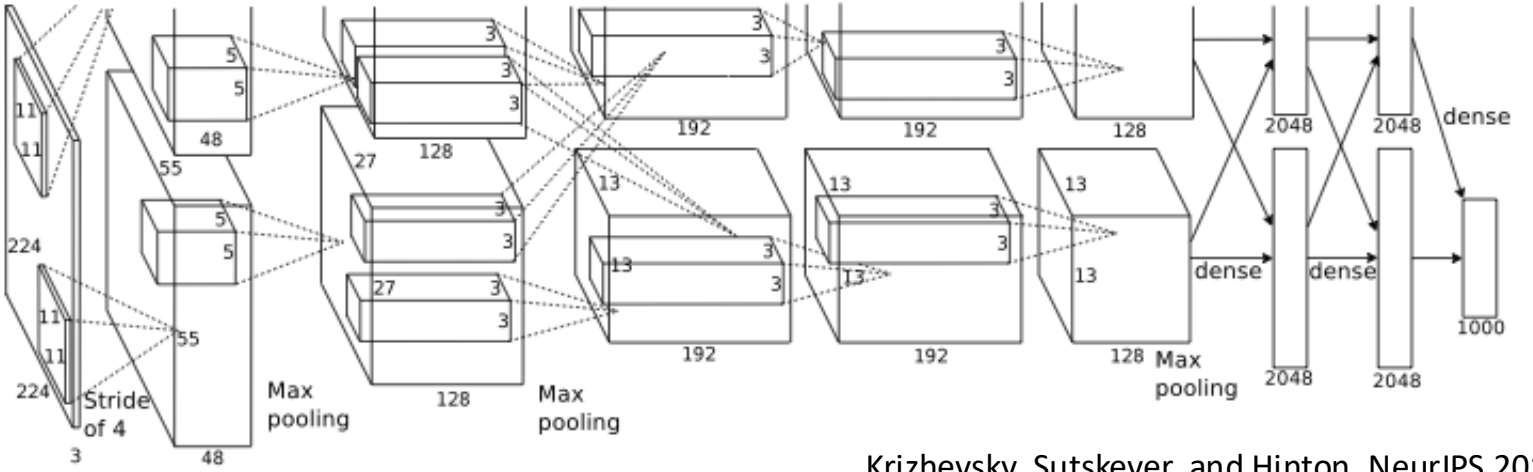


IMAGENET Large Scale Visual Recognition Challenge

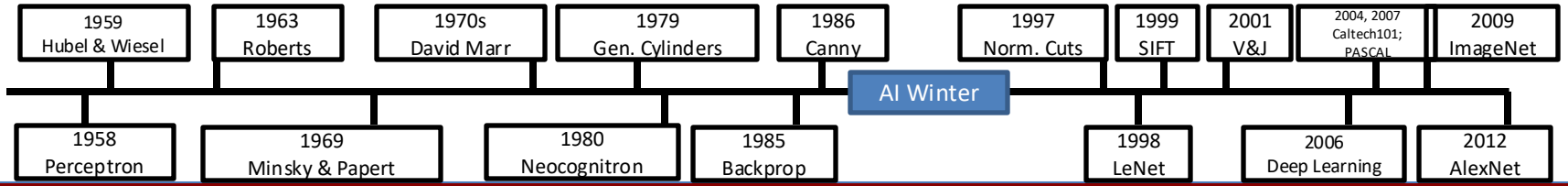




AlexNet: Deep Learning Goes Mainstream

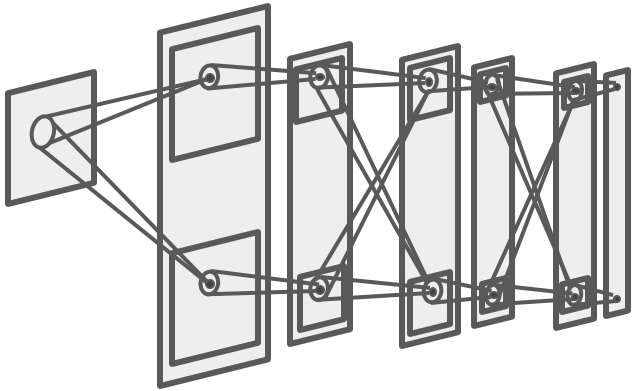


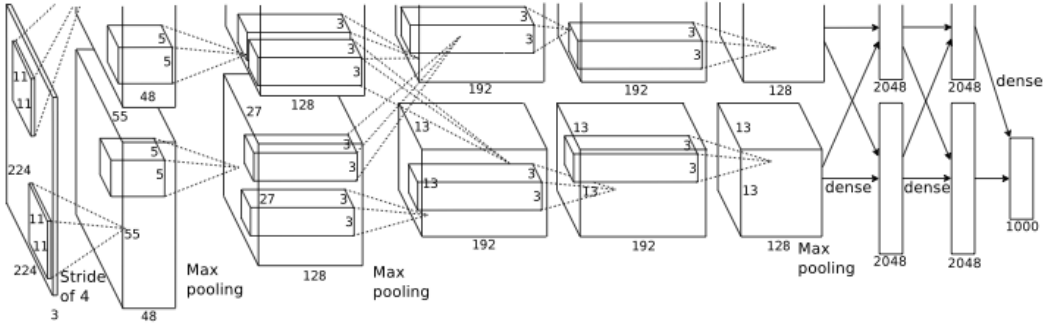
Krizhevsky, Sutskever, and Hinton, NeurIPS 2012

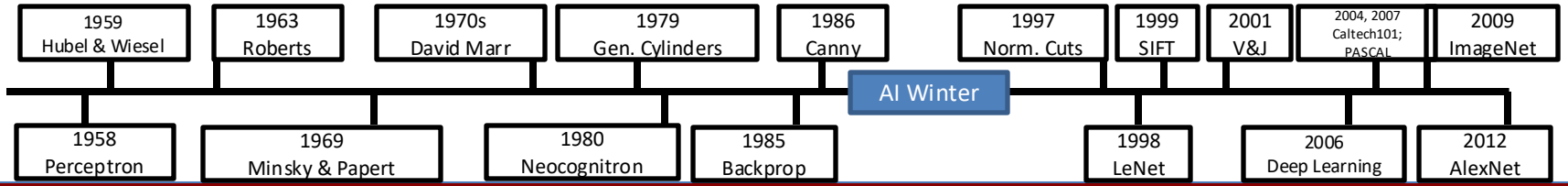


Slide inspiration: Justin Johnson

AlexNet vs. Neocognitron: 32 years apart

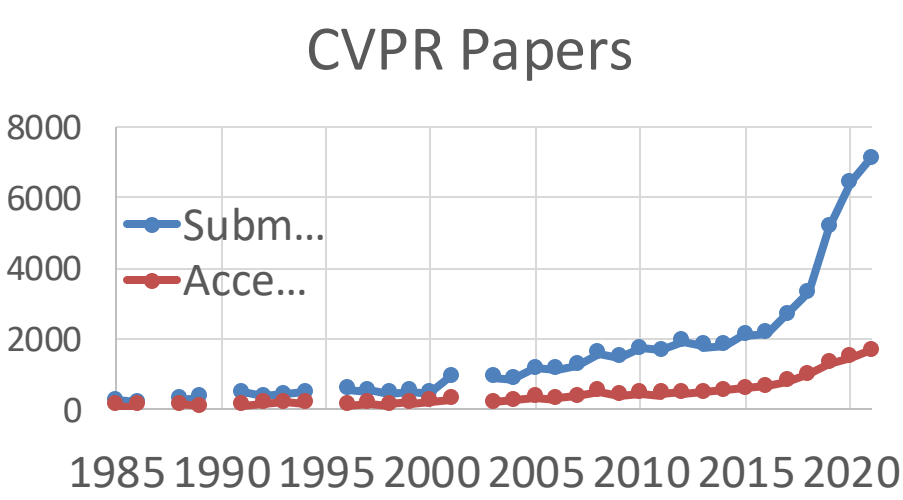




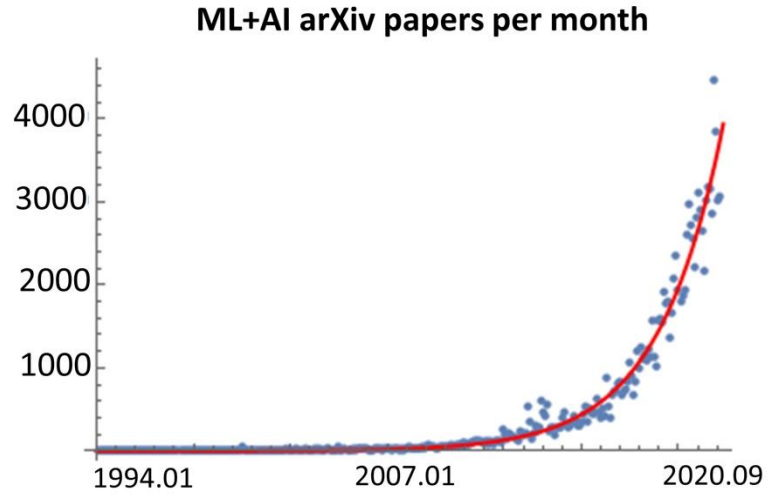


Slide inspiration: Justin Johnson

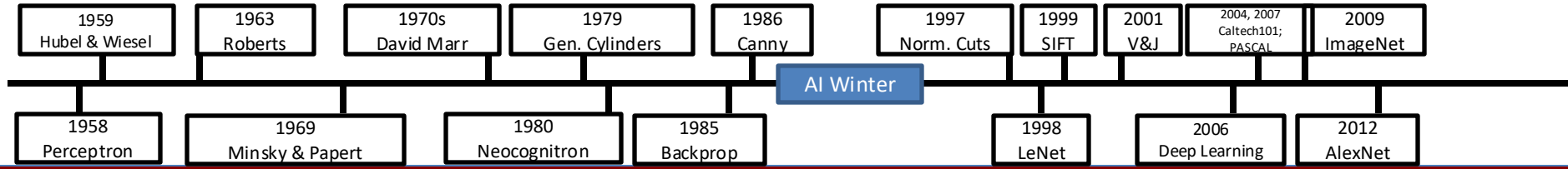
2012 to Present: Deep Learning Explosion



Publications at top Computer Vision conference



arXiv papers per month (source)



Slide inspiration: Justin Johnson

2012 to Present: Deep Learning is Everywhere

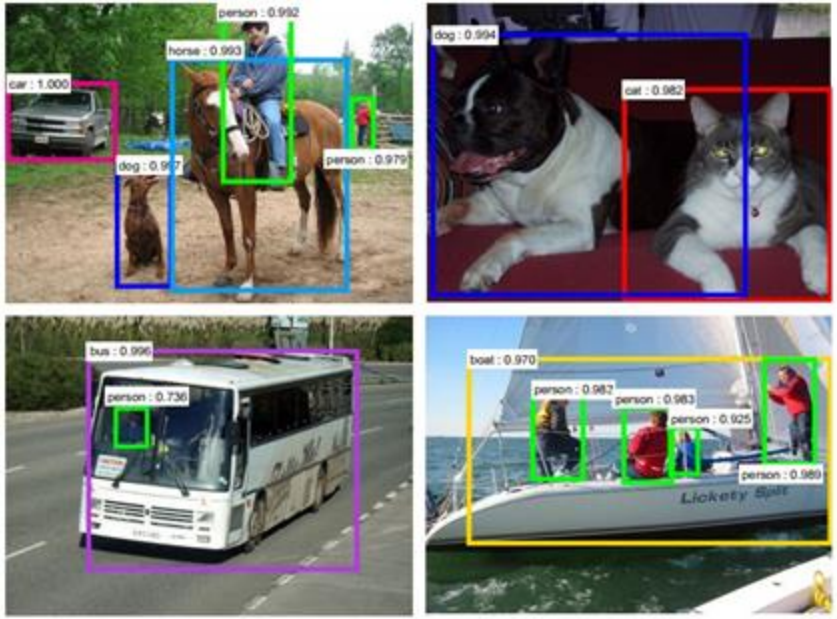
Image Classification

Image Retrieval

Slide inspiration: Justin Johnson

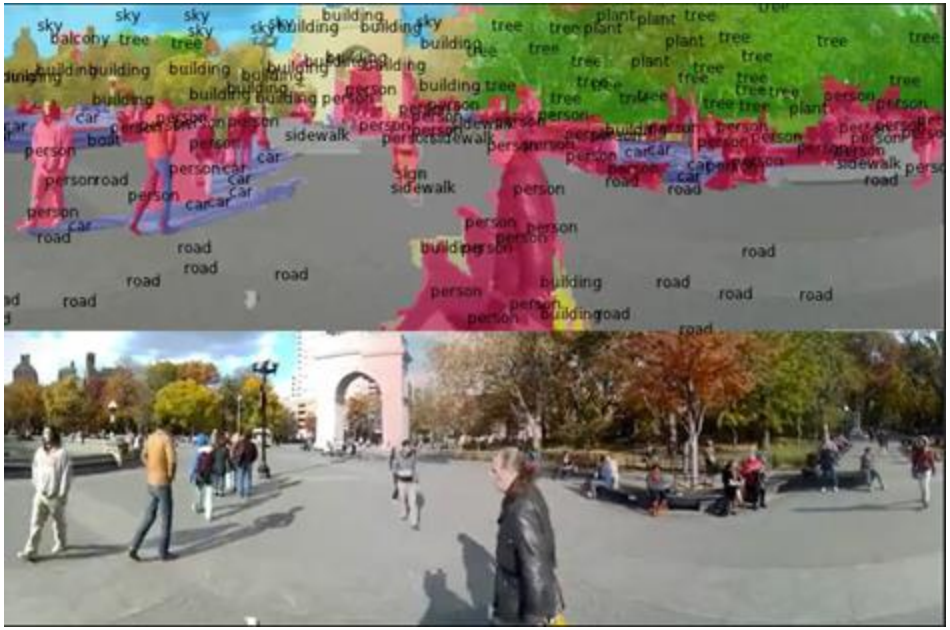
2012 to Present: Deep Learning is Everywhere

Object Detection



Ren, He, Girshick, and Sun, 2015

Image Segmentation

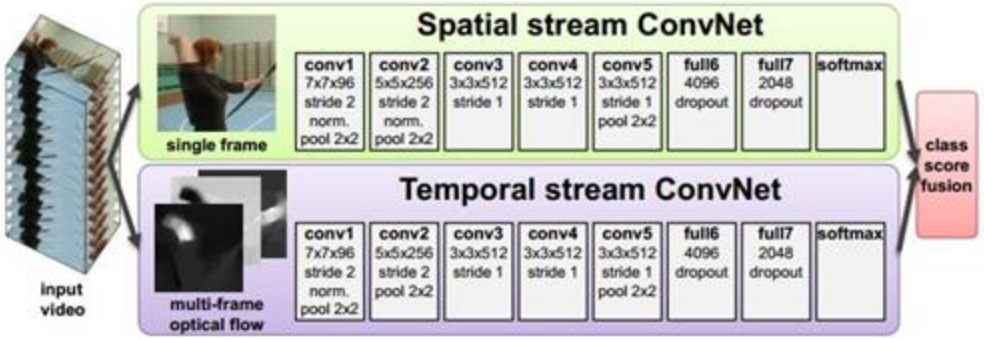


Fabaret et al, 2012

Slide inspiration: Justin Johnson

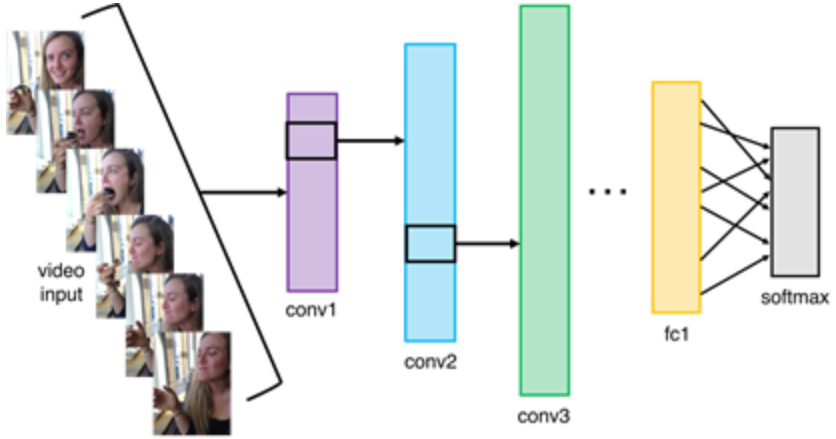
2012 to Present: Deep Learning is Everywhere

Video Classification



Simonyan et al, 2014

Activity Recognition

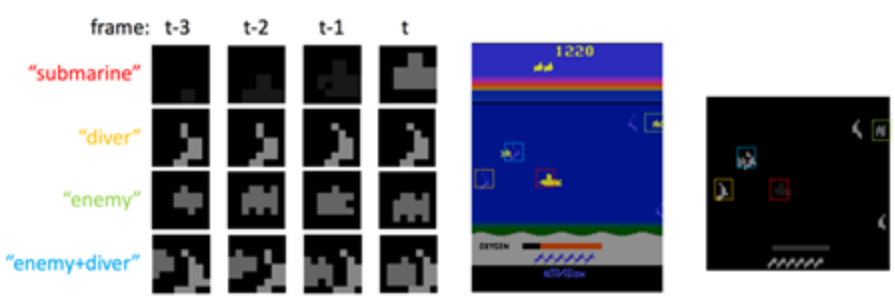


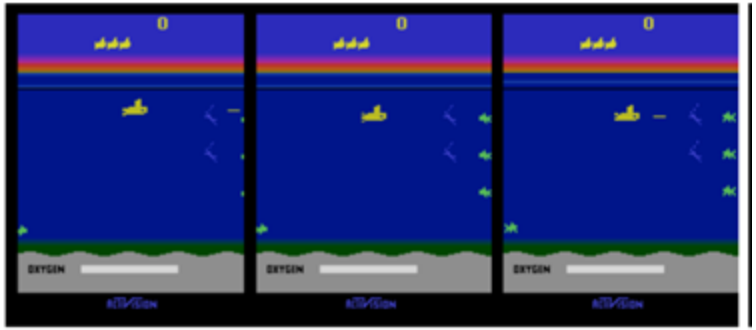
Slide inspiration: Justin Johnson

2012 to Present: Deep Learning is Everywhere

Pose Recognition (Toshev and Szegedy, 2014)

Playing Atari games (Guo et al, 2014)

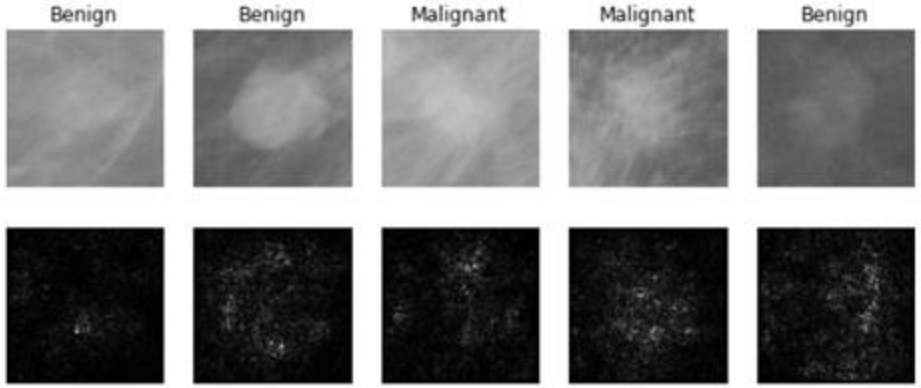




Slide inspiration: Justin Johnson

2012 to Present: Deep Learning is Everywhere

Medical Imaging



Levy et al, 2016 Figure reproduced with permission

Galaxy Classification

Dieleman et al, 2014

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

Whale recognition

Kaggle Challenge

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.

Slide inspiration: Justin Johnson

2012 to Present: Deep Learning is Everywhere

Image Captioning

Vinyals et al, 2015
Karpathy and Fei-Fei, 2015

A white teddy bear sitting in the grass

A man in a baseball uniform throwing a ball

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

Slide inspiration: Justin Johnson

2012 to Present: Deep Learning is Everywhere

TEXT PROMPT

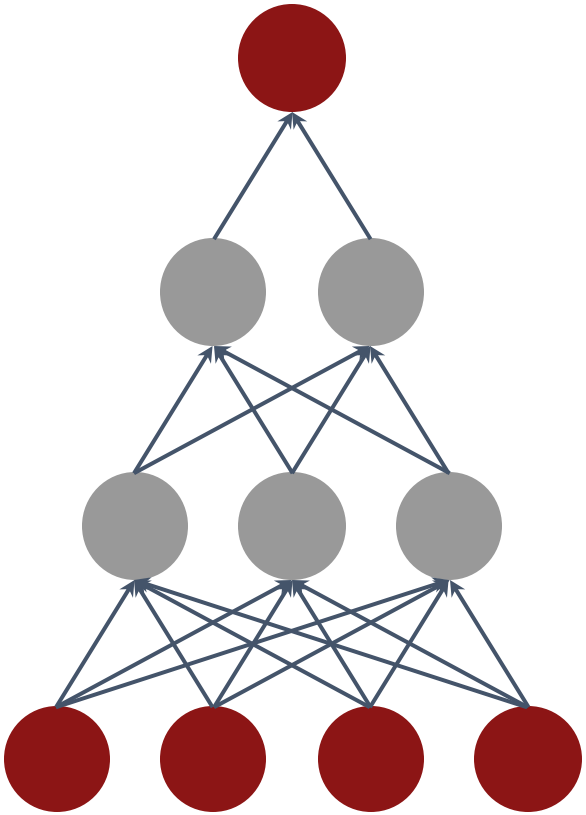
an armchair in the shape of an avocado. an armchair imitating an avocado.

AI-GENERATED IMAGES

Slide inspiration: Justin Johnson

Ramesh et al, "DALL·E: Creating Images from Text", 2021. https://openai.com/blog/dall-e/

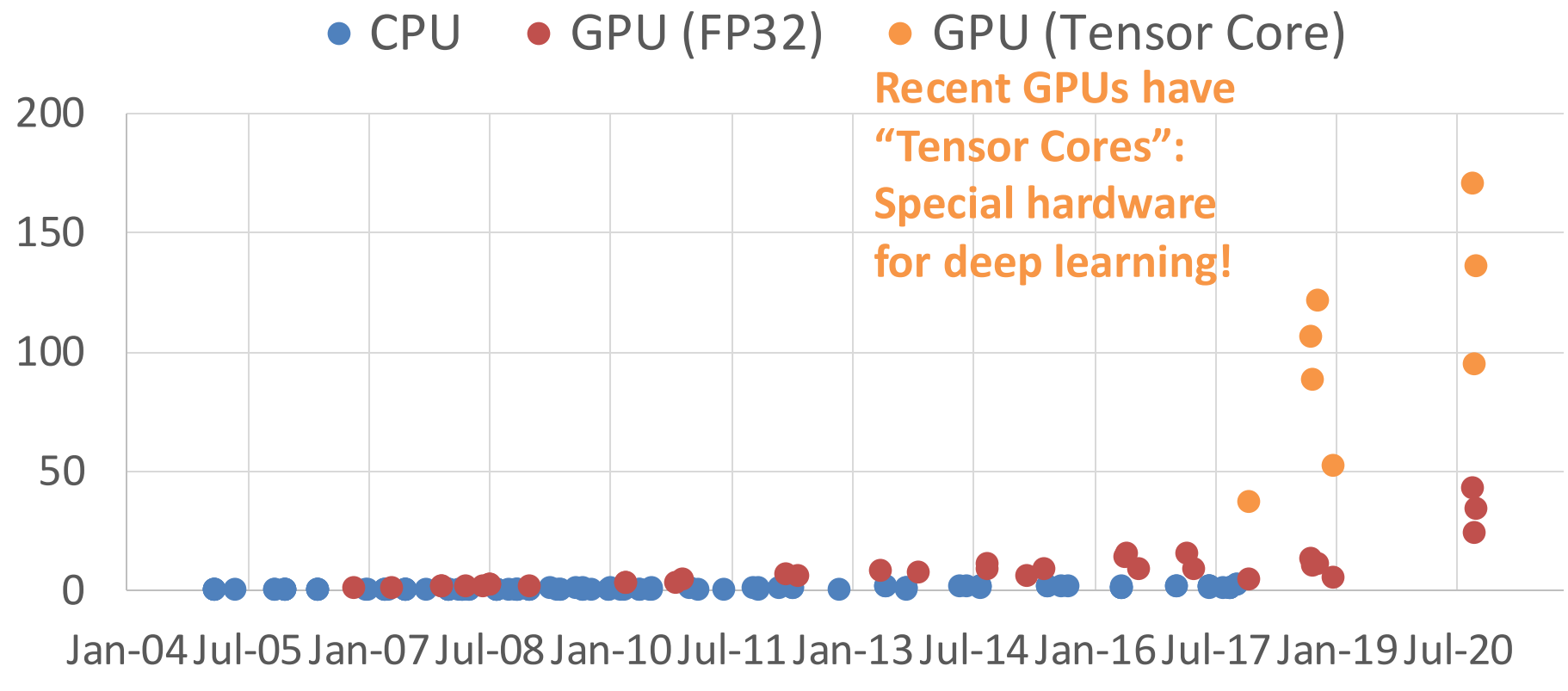
Computation



Algorithms

Data

GFLOP per Dollar



Slide inspiration: Justin Johnson

Despite the successes, computer vision still has a long way to go

Computer Vision Can Cause Harm

Harmful Stereotypes

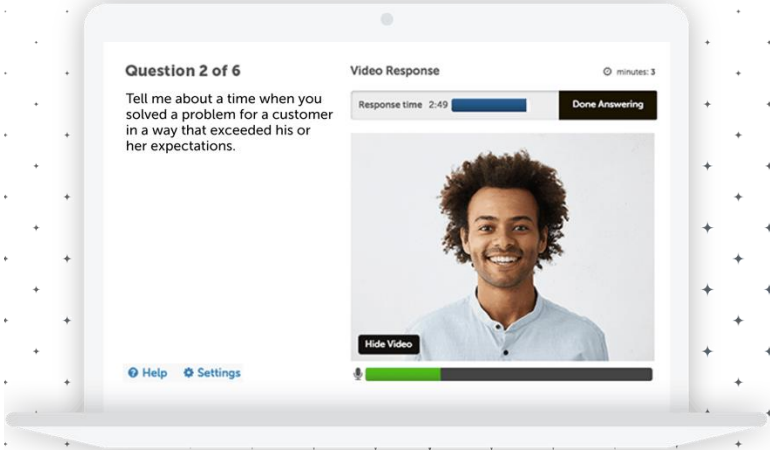
Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%

Affect people's lives

Technology

A face-scanning algorithm increasingly decides whether you deserve the job

HireVue claims it uses artificial intelligence to decide who's best for a job. Outside experts call it 'profoundly disturbing.'



Source: https://www.washingtonpost.com/technology/2019/10/22/ai-hiring-face-scanning-algorithm-increasingly-decides-whether-you-deserve-job/
https://www.hirevue.com/platform/online-video-interviewing-software

Example Credit: Timnit Gebru

Computer Vision Can Save Lives

Slide inspiration: Justin Johnson

And there is a lot we don't know how to do

https://fedandfit.com/wp-content/uploads/2020/06/summer-activities-for-kids_optimized-scaled.jpeg

This image is copyright-free United States government work

Slide inspiration: Andrej Karpathy

Lecture 2:

Image Classification with Linear Classifiers

Image Classification

A Core Task in Computer Vision

Today:

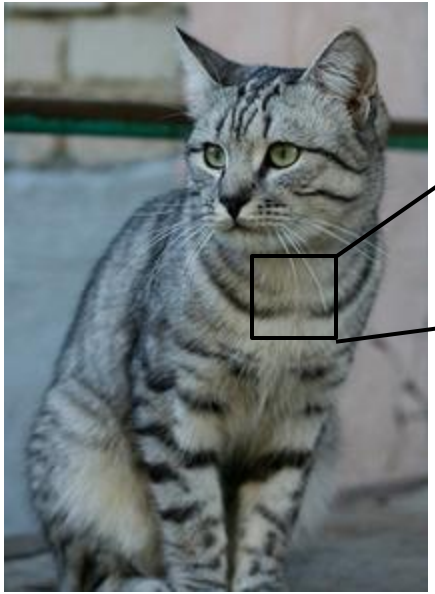
- The image classification task
- Two basic data-driven approaches to image classification
  - K-nearest neighbor and linear classifier

Image Classification: A core task in Computer Vision

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}

cat

The Problem: Semantic Gap



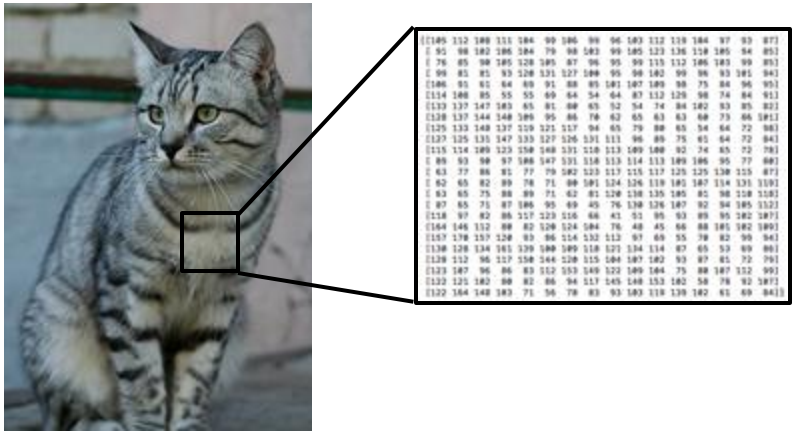

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

What the computer sees

An image is a tensor of integers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Viewpoint variation



All pixels change when the camera moves!

Challenges: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

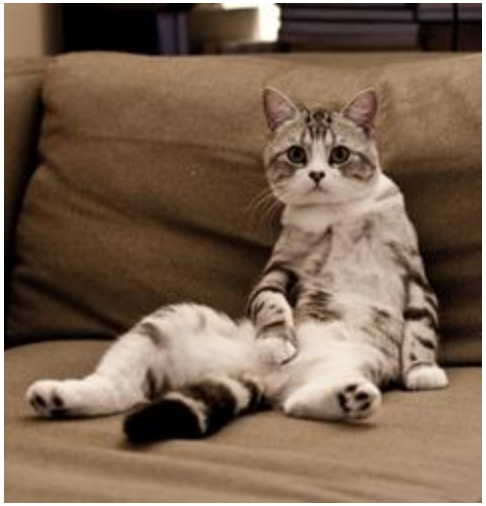
Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Challenges: Deformation



Challenges: Intraclass variation

Challenges: Context

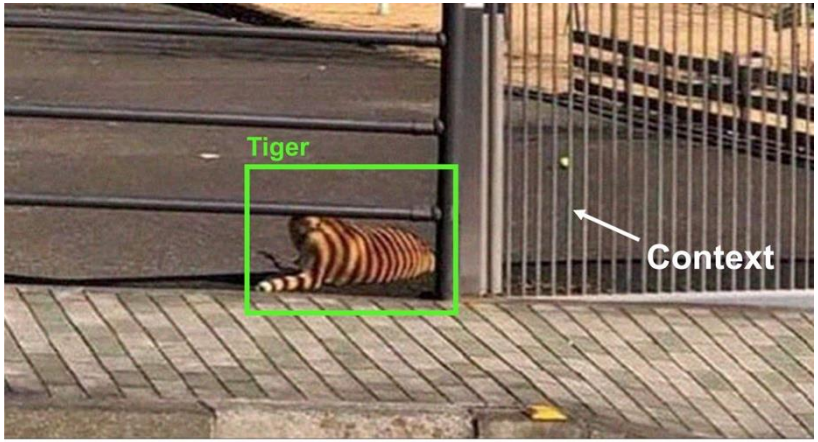


Image source: https://www.linkedin.com/posts/ralph-aboujaoude-diaz-40838313_technology-artificialintelligence-computervision-activity-6912446088364875776-h-Iq?utm_source=linkedin_share&utm_medium=member_desktop_web

An image classifier

```
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Linear Classifier

Parametric Approach: Linear Classifier

$$f(x,W) = Wx + b$$

3072x1 (x)

10x1 (b)

10x1 (f(x,W))

10x3072 (W)

Image

Array of 32x32x3 numbers
(3072 numbers total)

f(x,W)

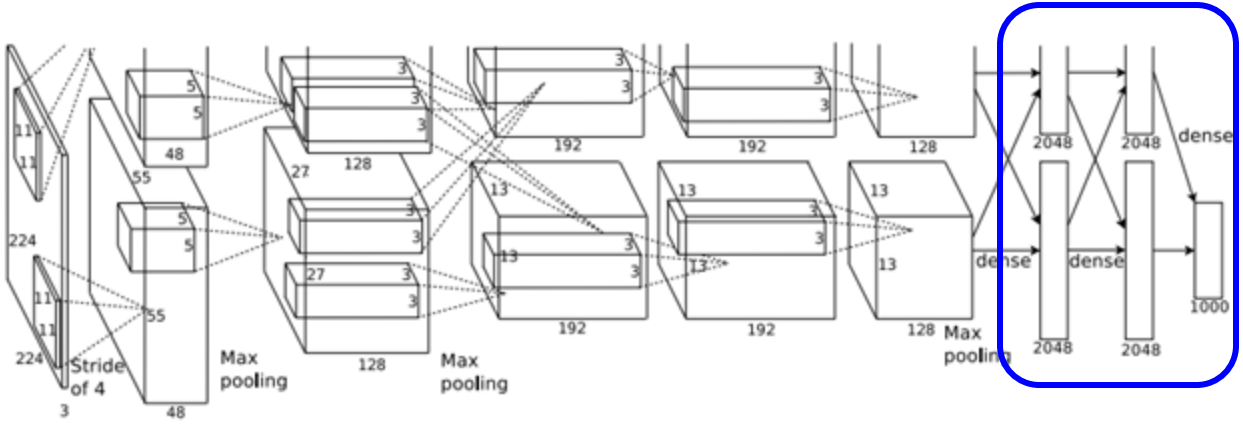
10 numbers giving class scores

W

parameters or weights

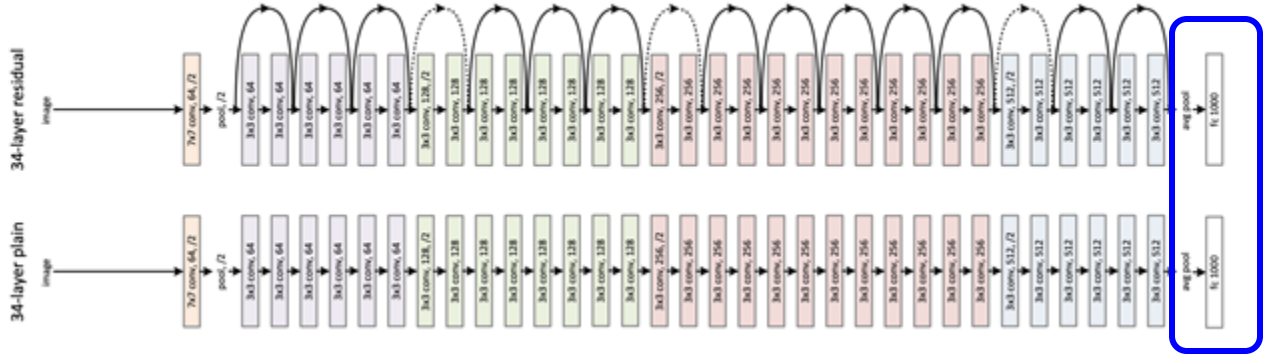
Neural Network

Linear classifiers



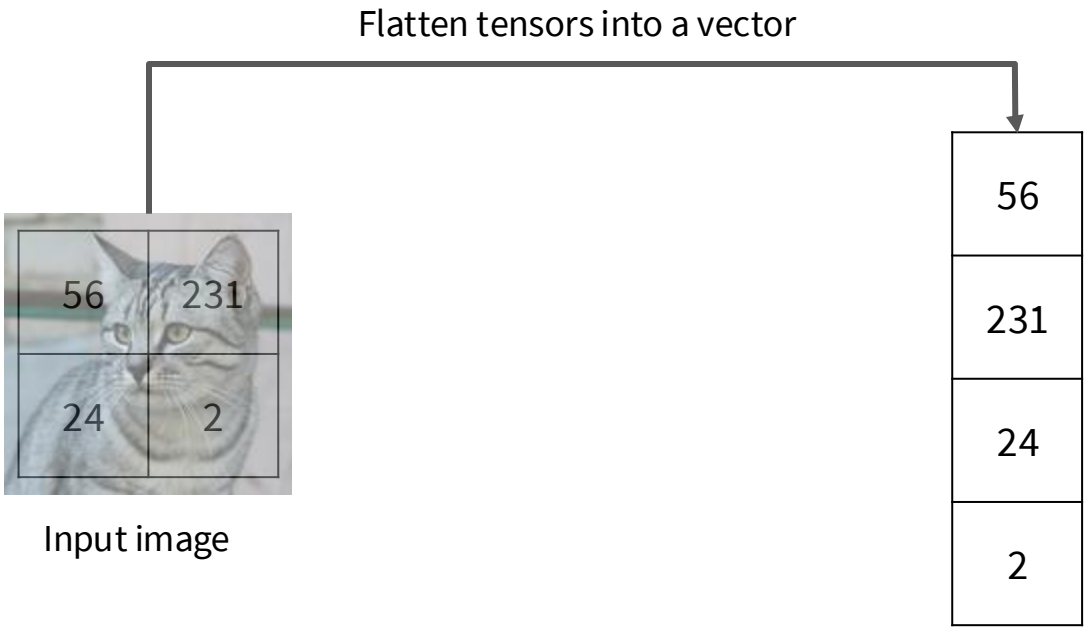
[Krizhevsky et al. 2012]

Linear layers



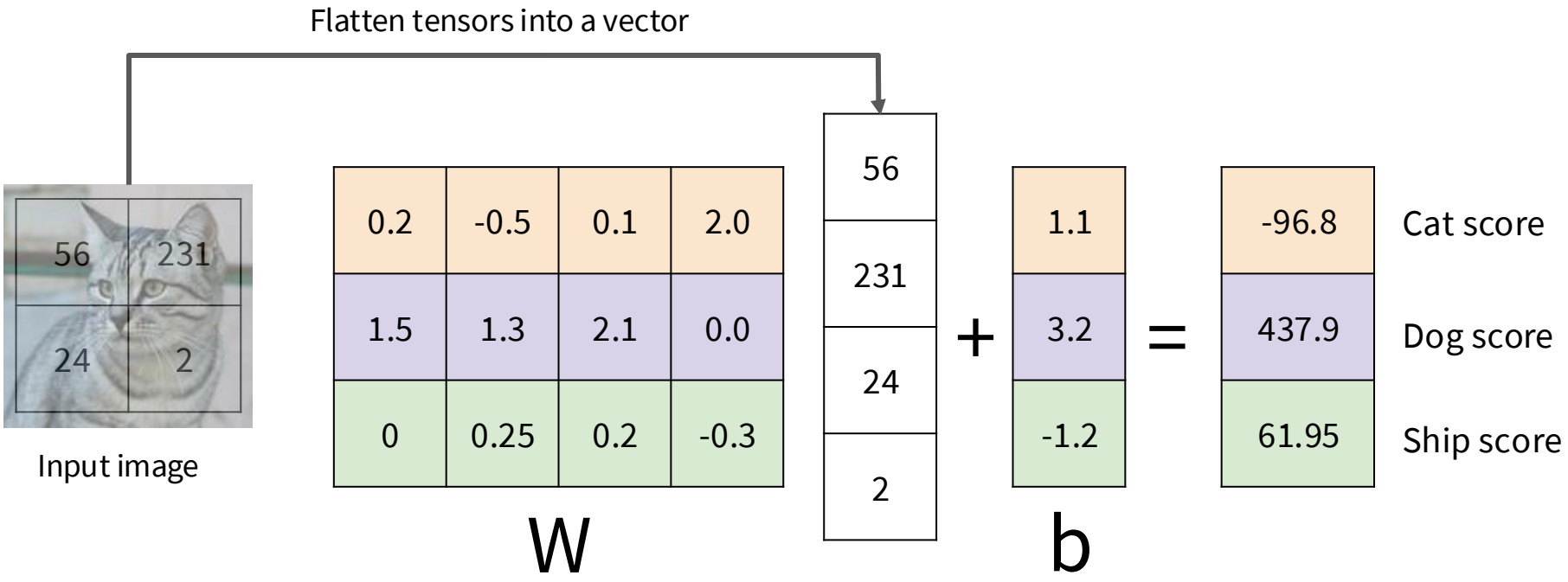
[He et al. 2015]

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

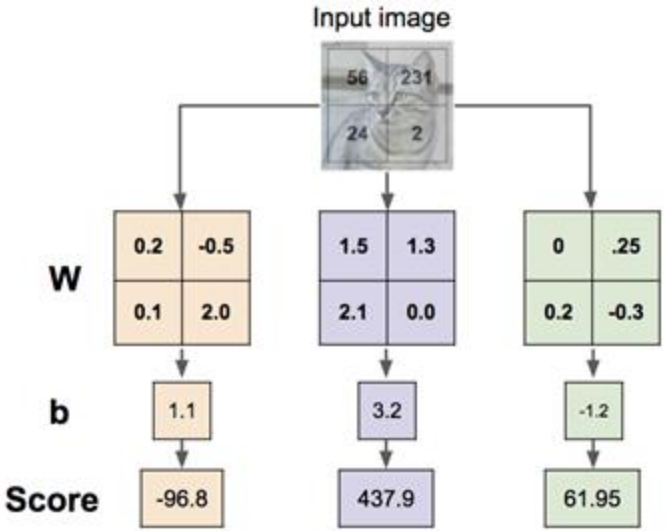


Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

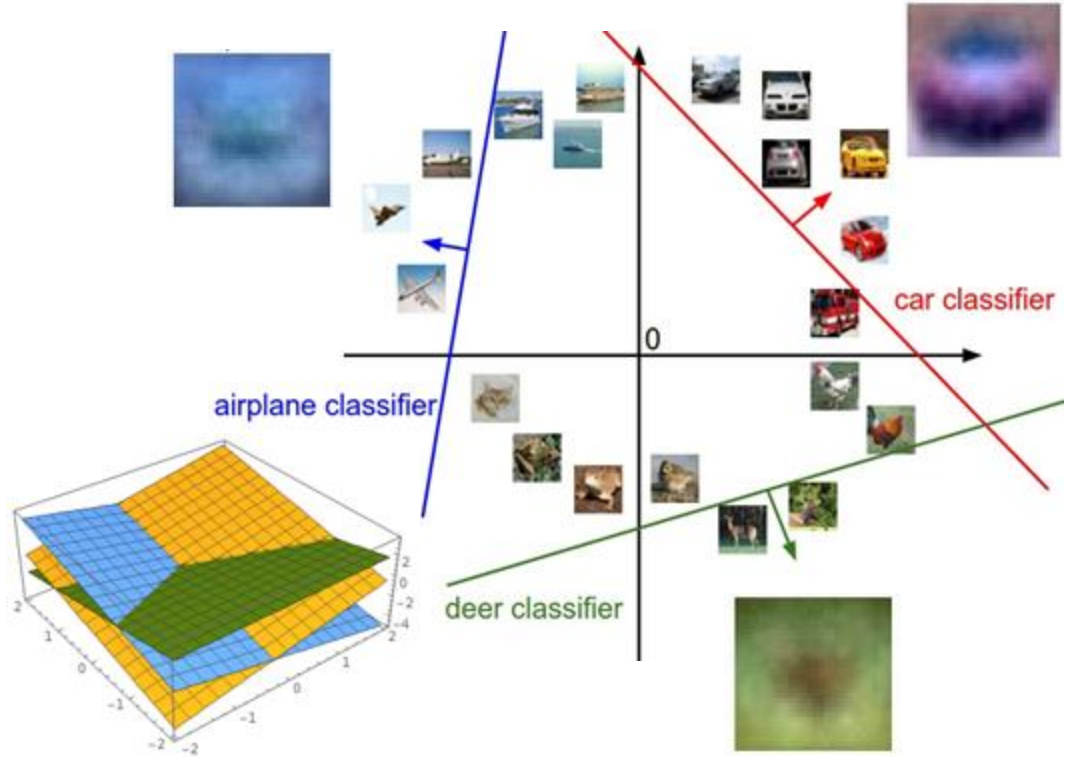
Algebraic Viewpoint



Interpreting a Linear Classifier: Visual Viewpoint



Interpreting a Linear Classifier: Geometric Viewpoint



$$f(x,W) = Wx + b$$

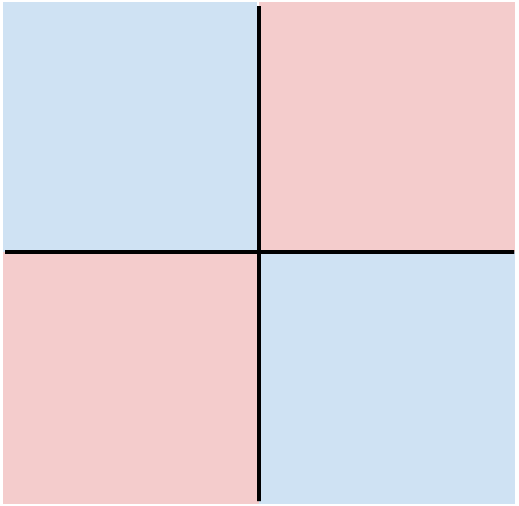
Array of 32x32x3 numbers
(3072 numbers total)

Plot created using Wolfram Cloud

Hard cases for a linear classifier

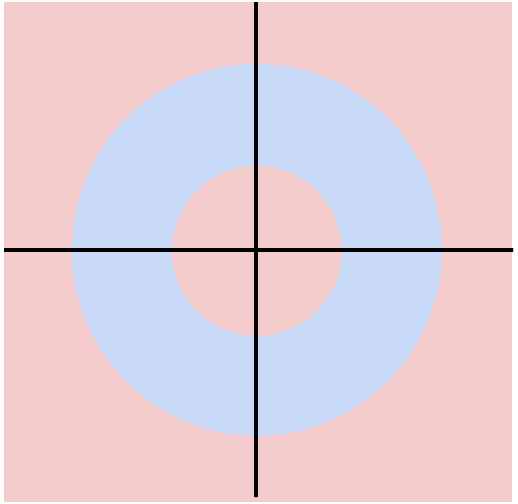
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



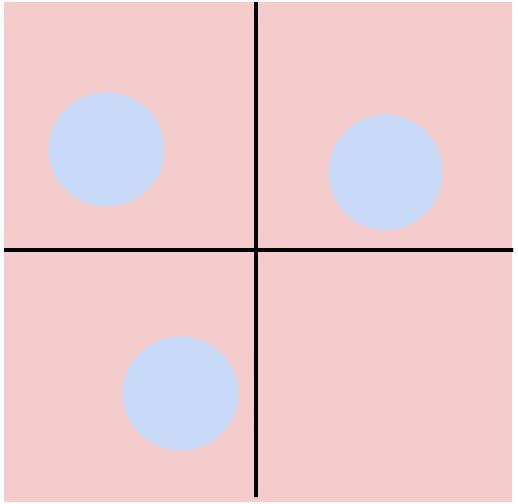
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Softmax classifier

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities
cat	3.2		24.5		0.13
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ L_i = -log(0.13) = 2.04

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data
(See CS 229 for details)

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

$$L_i = -\log P(Y = y_i|X = x_i)$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	3.2		24.5		0.13		1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

$$L_i = -\log P(Y = y_i|X = x_i)$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	3.2		24.5		0.13		1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Kullback–Leibler divergence

$$D_{KL}(P\|Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
	Probabilities must be >= 0				Probabilities must sum to 1		
cat	3.2		24.5		0.13		1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P\|Q)$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Maximize probability of correct class

$$L_i = -\log P(Y = y_i|X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

cat	3.2
car	5.1
frog	-1.7

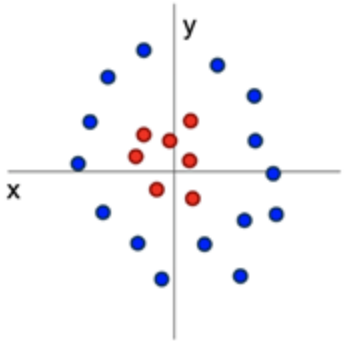
Lecture 5:
Image Classification with CNNs

Problem: Linear Classifiers are not very powerful

Visual Viewpoint

Linear classifiers learn one template per class

Geometric Viewpoint

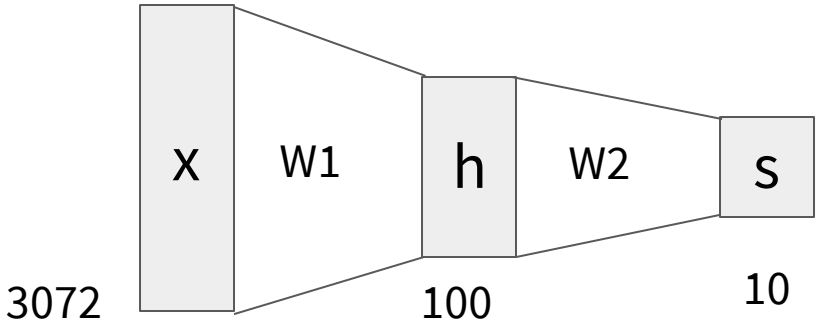


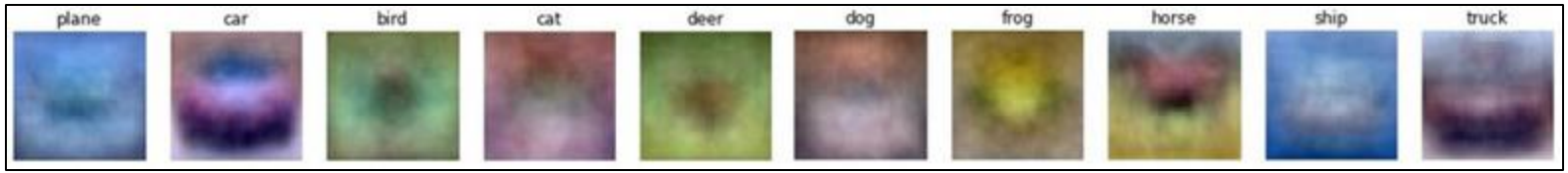
Linear classifiers can only draw linear decision boundaries

Last time: Neural Networks

Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$

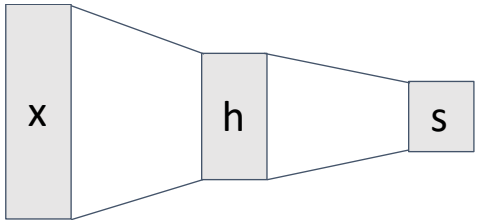




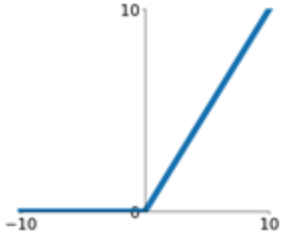
Today: Convolutional Networks

Fully-Connected Layer

We have already seen these



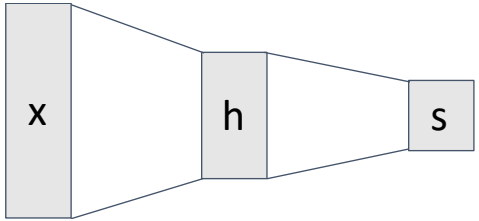
Activation Function



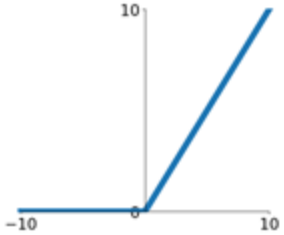
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer

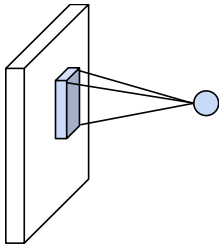


Activation Function

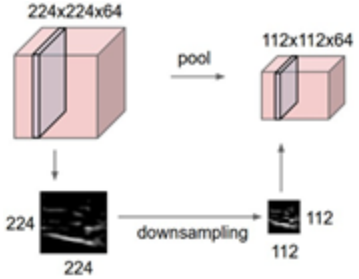


Today: Image-specific operators

Convolution Layer



Pooling Layer

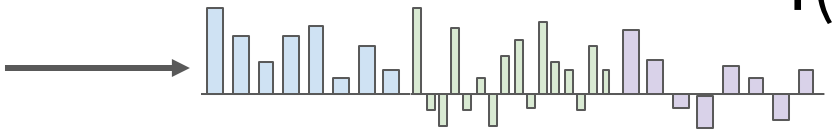


Pixel space

Class scores

$$f(x) = Wx$$

Image features

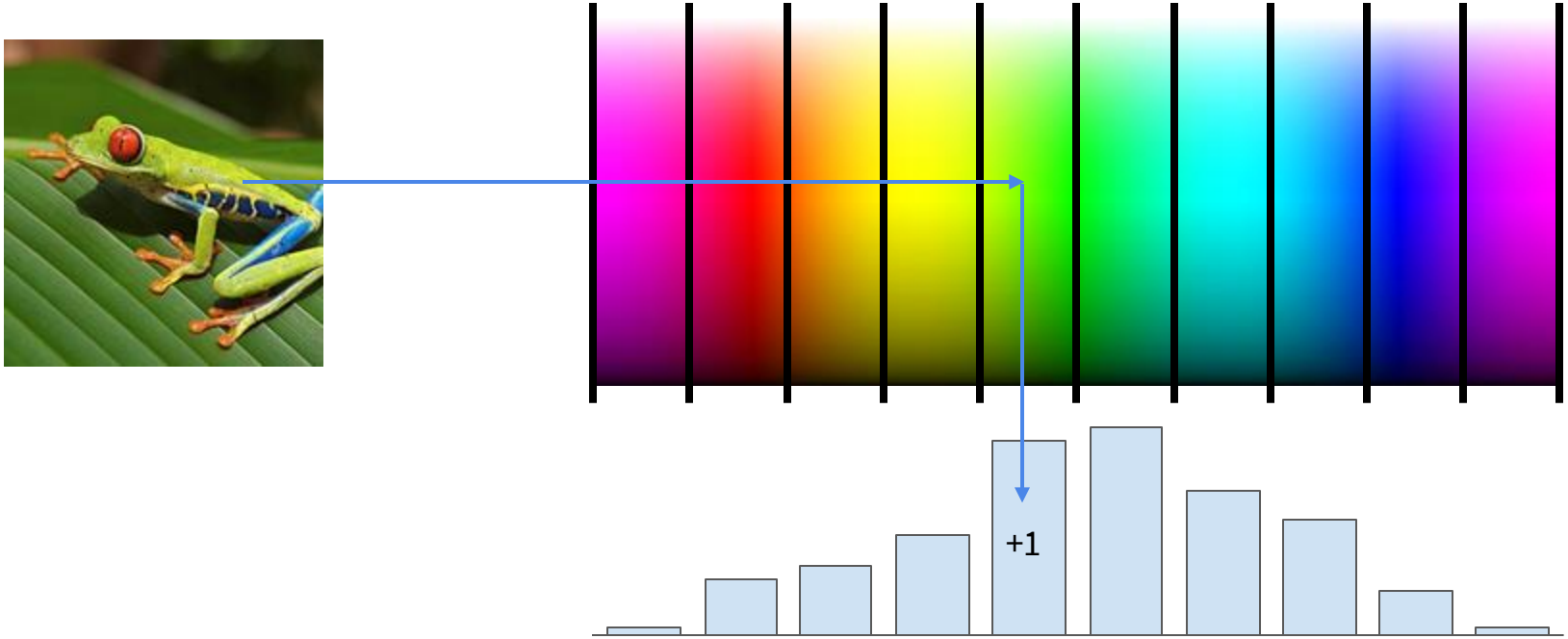


$f(x) = Wx$

Feature Representation

Class scores

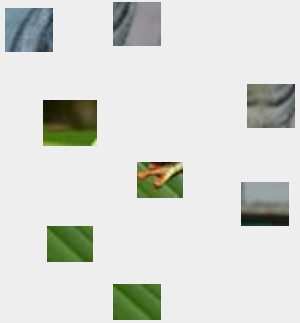
Example: Color Histogram



Example: Bag of Words

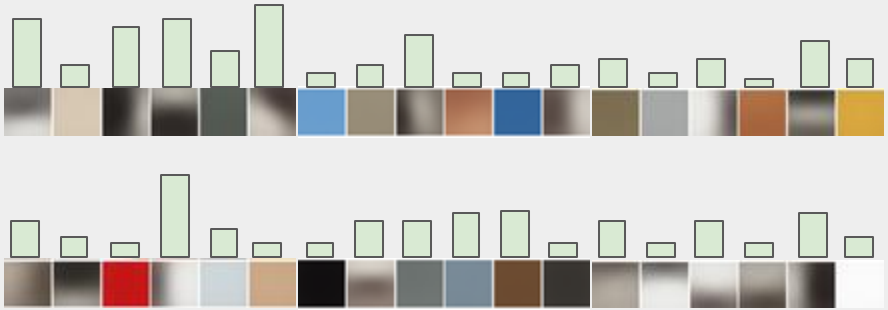
Step 1: Build codebook

Extract random patches



Cluster patches to form “codebook” of “visual words”

Step 2: Encode images



Fei-Fei and Perona, “A bayesian hierarchical model for learning natural scene categories”, CVPR 2005

Image Features

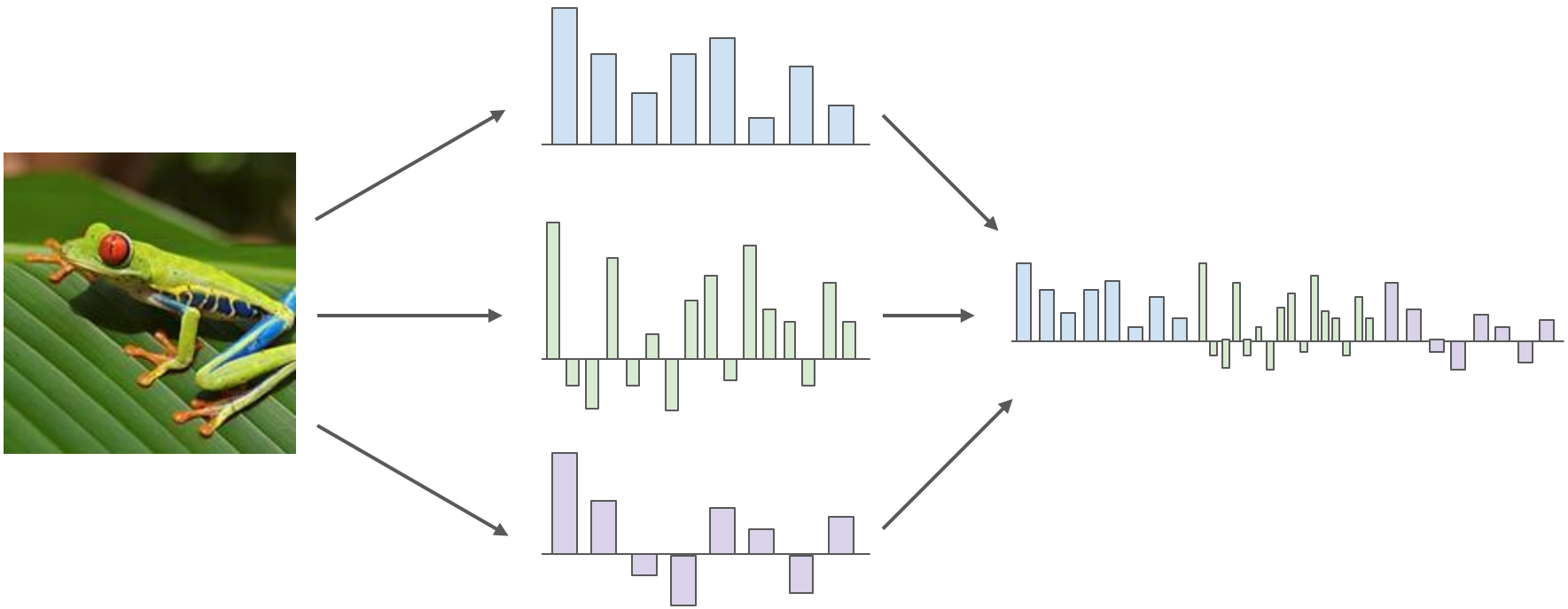


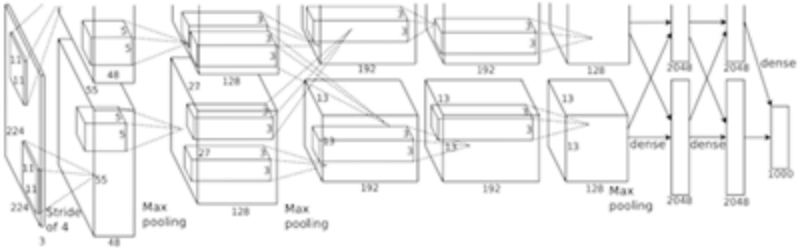
Image features vs. ConvNets

Feature Extraction

f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

10 numbers giving scores for classes

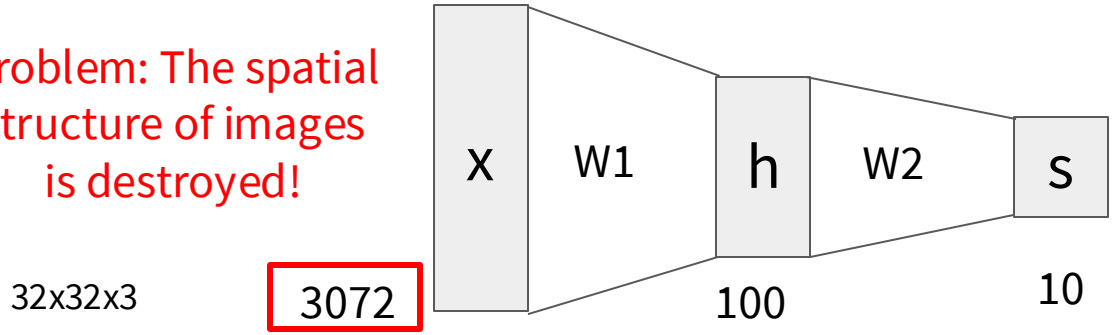
training

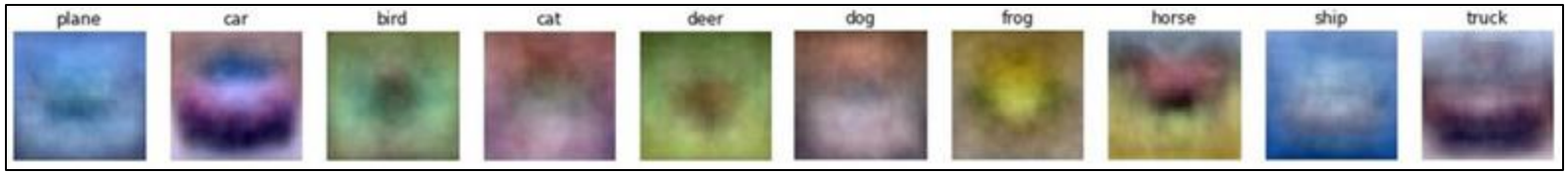
Last Time: Neural Networks

Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$

Problem: The spatial structure of images is destroyed!





Next: Convolutional Neural Networks

Convolution and **pooling** operators extract features while respecting 2D image structure

Fully-Connected layers form an MLP at the end to predict scores

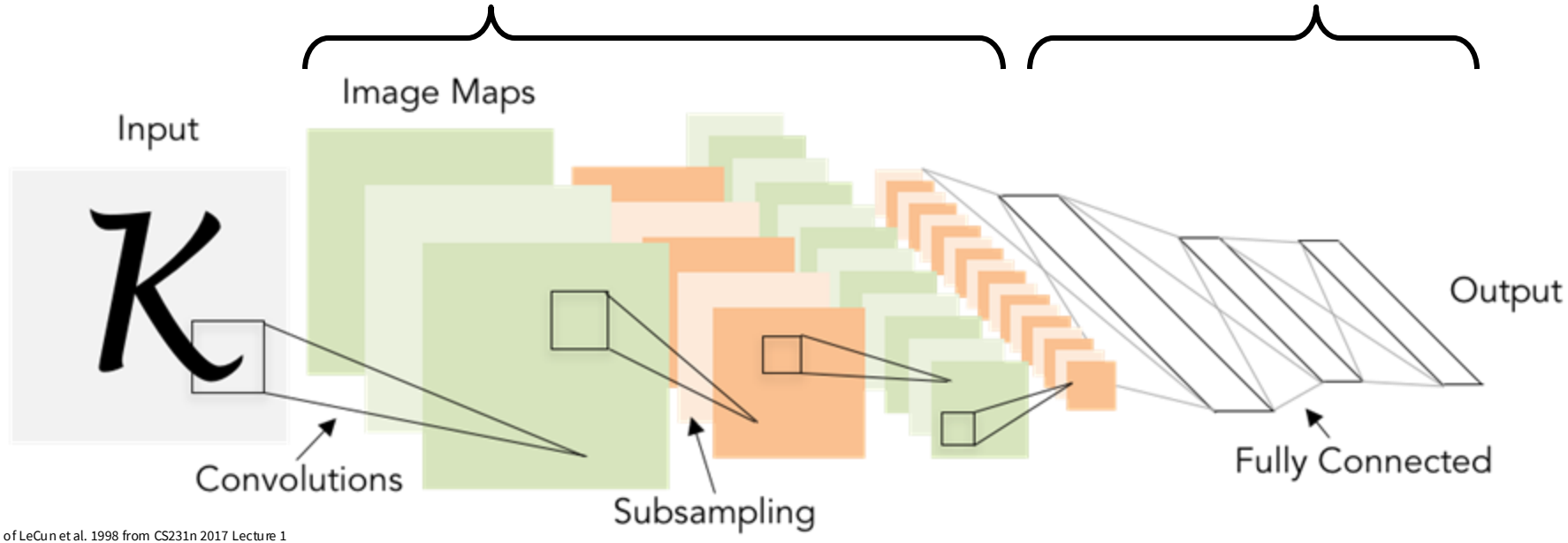


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Next: Convolutional Neural Networks

Trained end-to-end with backprop + gradient descent

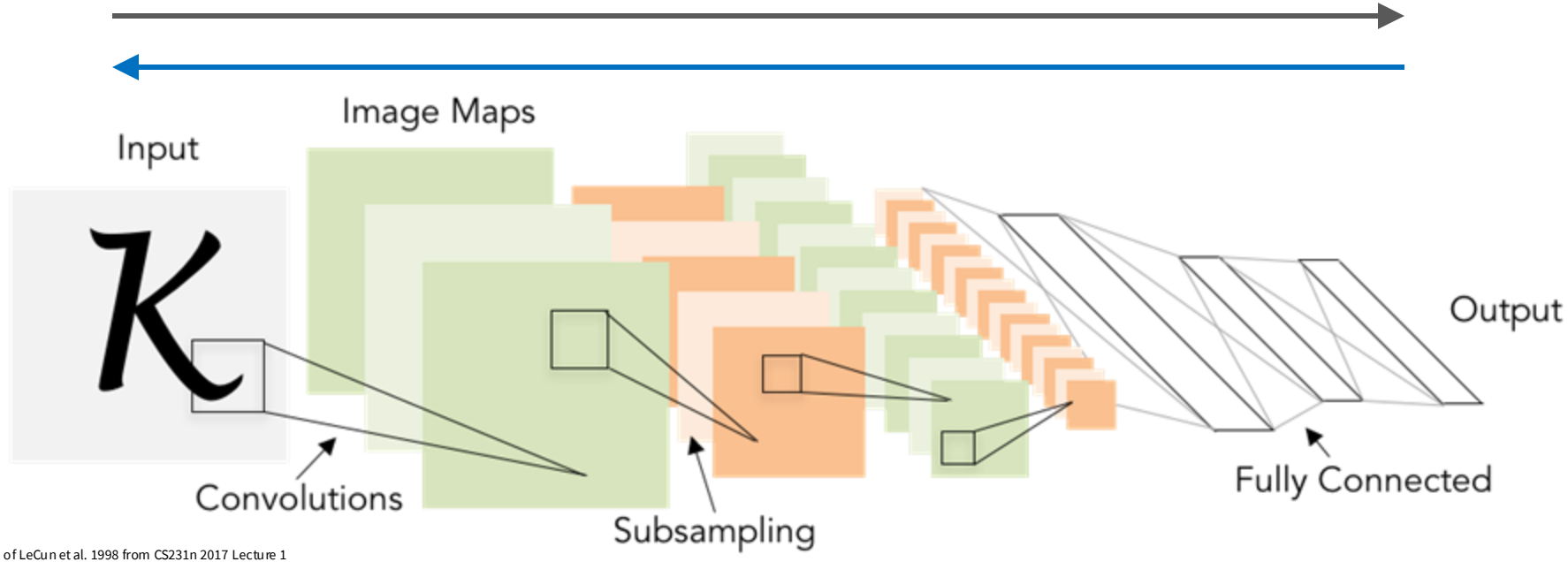
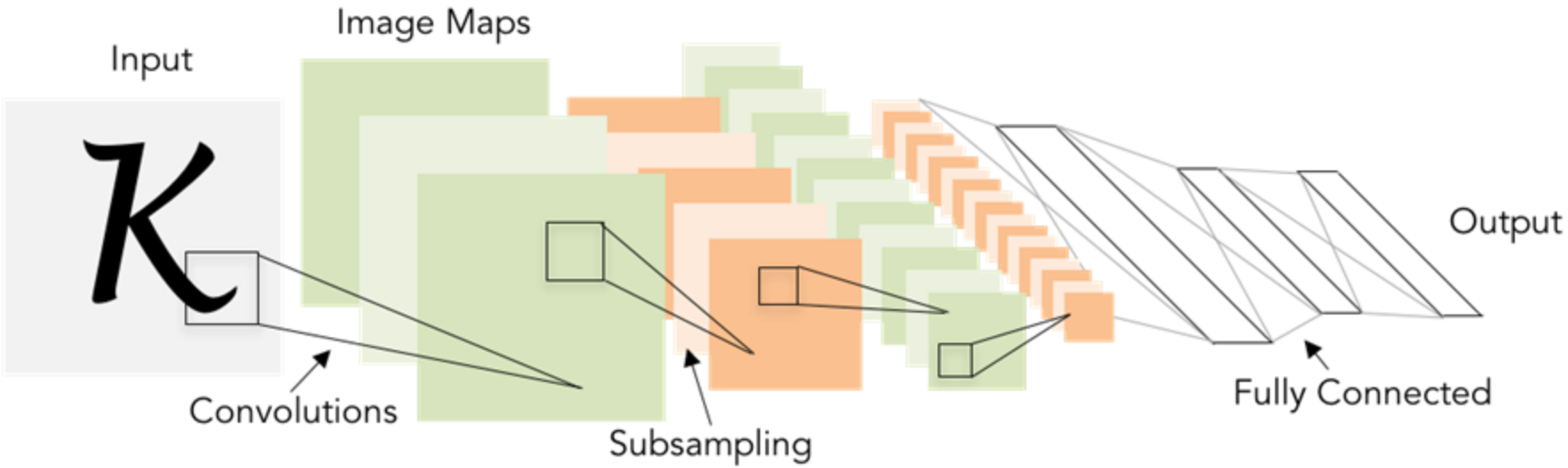


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

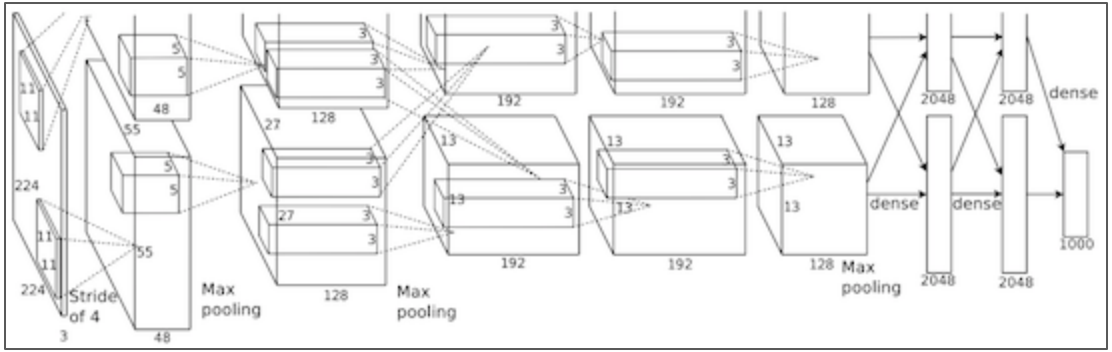
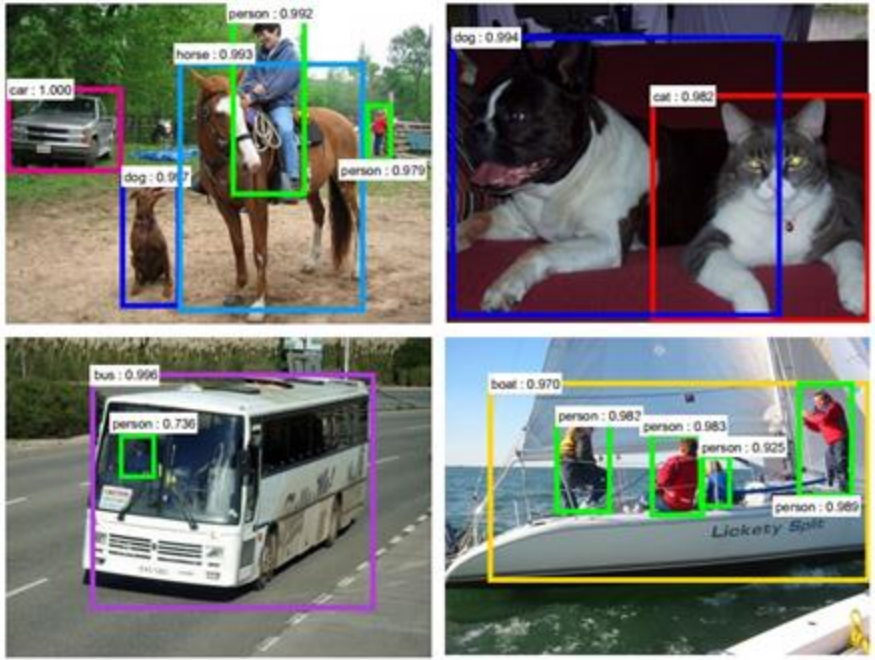


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

~2012 – 2020: ConvNets dominate all vision tasks

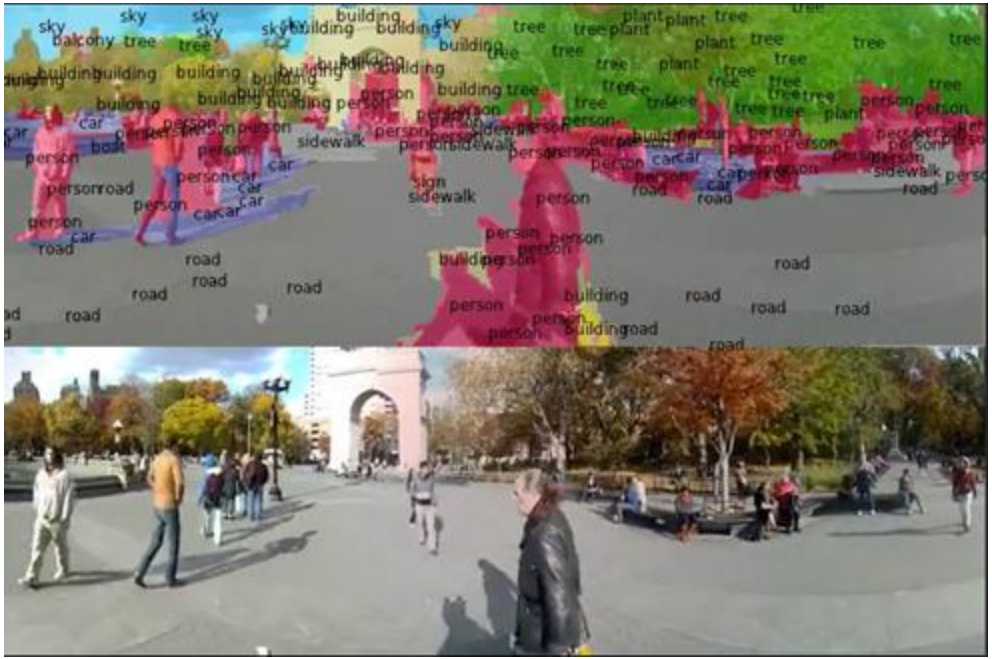
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Detection



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

~2012 – 2020: ConvNets dominate all vision tasks

Image Captioning

A white teddy bear sitting in the grass

A man in a baseball uniform throwing a ball

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

Captions generated by Justin Johnson using Neuraltalk2

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

~2012 – 2020: ConvNets dominate all vision tasks

Text-to-Image Generation

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

A zombie in the style of Picasso

An image of a half mouse half octopus

A painting of a squirrel eating a burger

A watercolor painting of a chair that looks like an octopus

A shirt with the inscription: "I love generative models!"

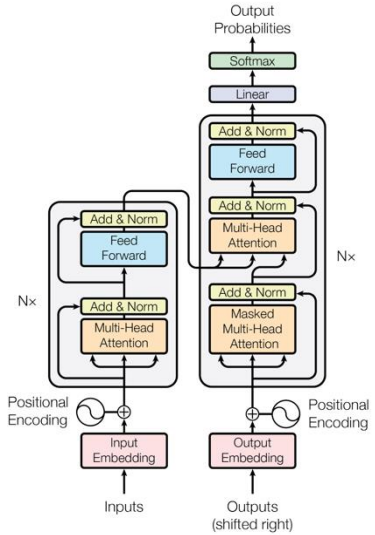
~2012 – 2020: ConvNets dominate all vision tasks

CS231n: Convolutional Neural Networks for Visual Recognition

This class used to be focused on ConvNets!

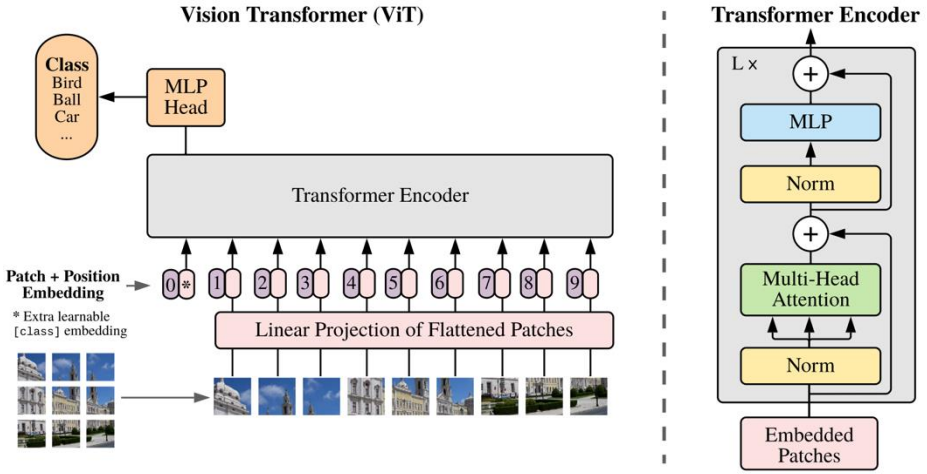
2021 - Present: Transformers have taken over

2017: Transformers for language tasks



Vaswani et al, "Attention is all you need", NeurIPS 2017

2021: Transformers for vision tasks



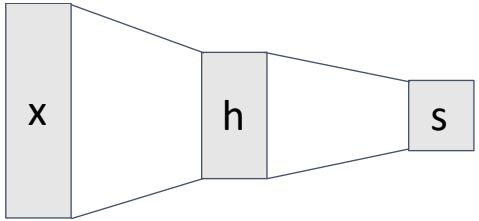
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Convolutional Neural Networks

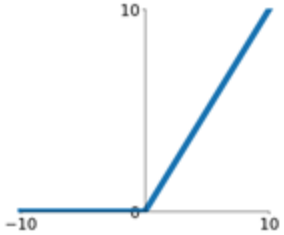
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer

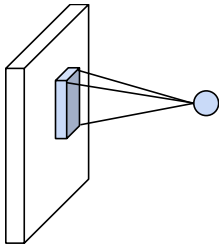


Activation Function

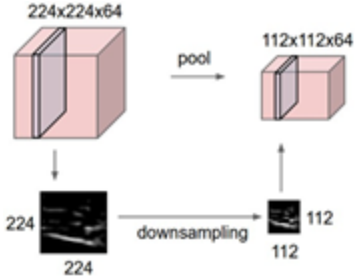


Today: Image-specific operators

Convolution Layer



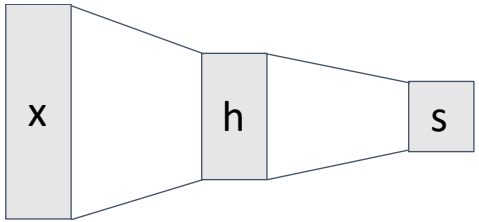
Pooling Layer



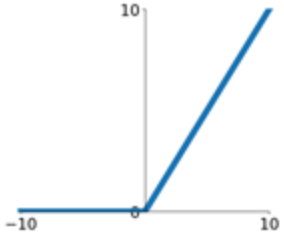
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer

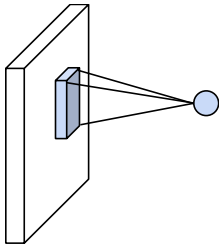


Activation Function



Today: Image-specific operators

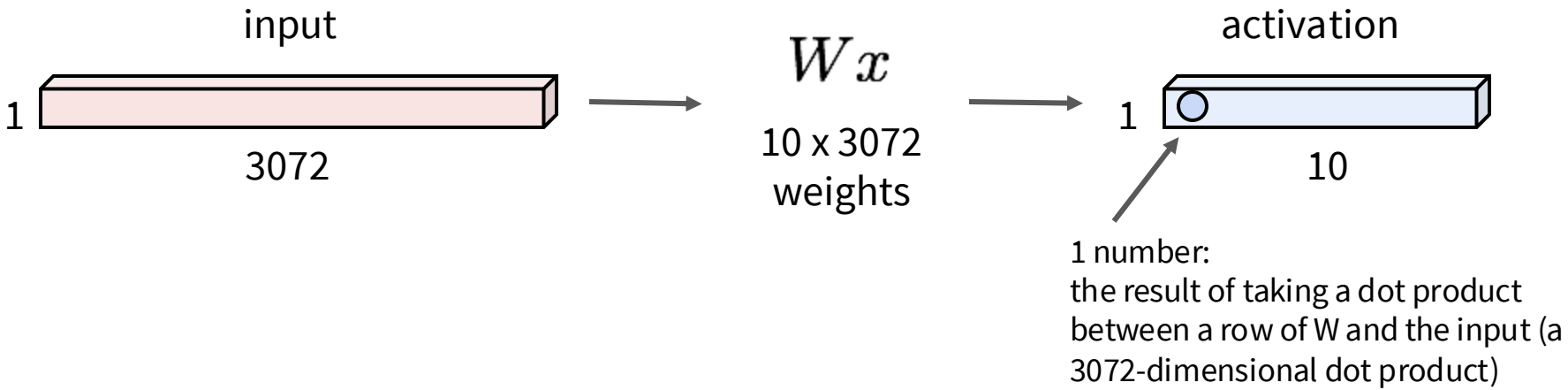
Convolution Layer



Pooling Layer

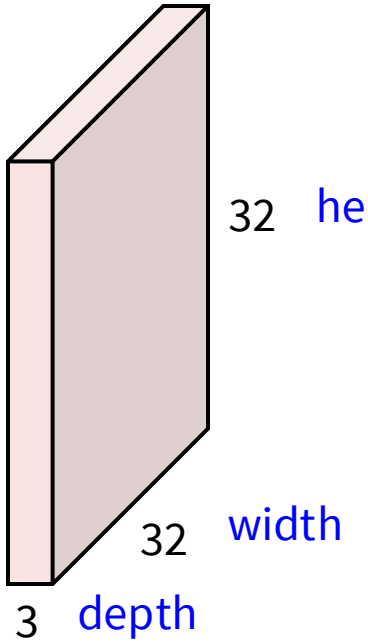
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



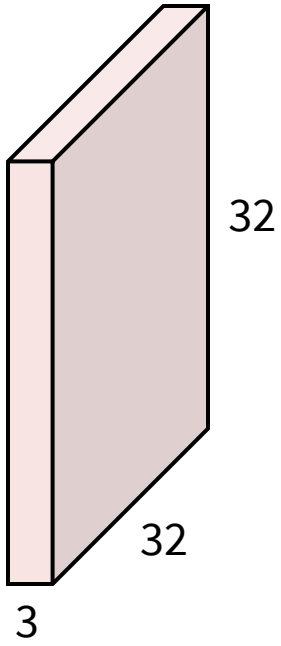
Convolution Layer

32x32x3 image -> preserve spatial structure

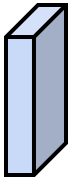


Convolution Layer

32x32x3 image



5x5x3 filter

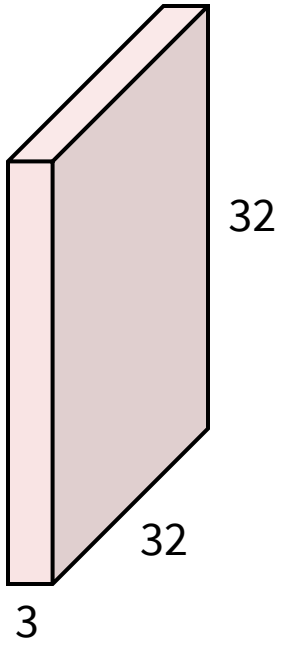


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

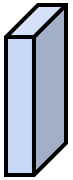
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

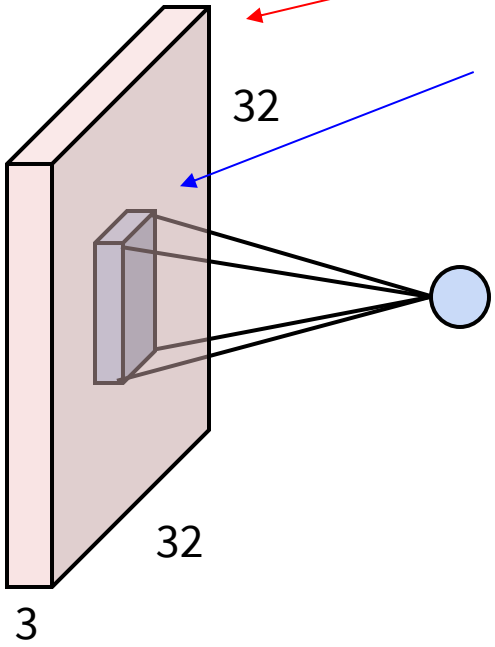


5x5x3 filter



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

Convolution Layer



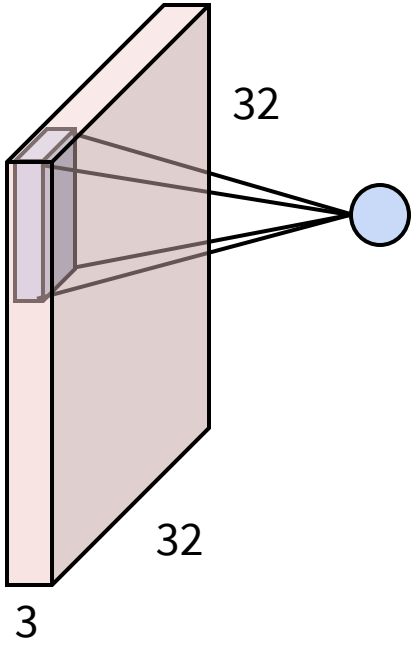
32x32x3 image

5x5x3 filter w

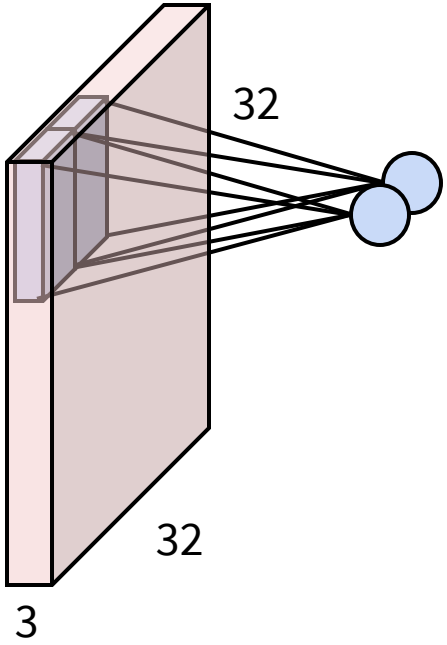
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

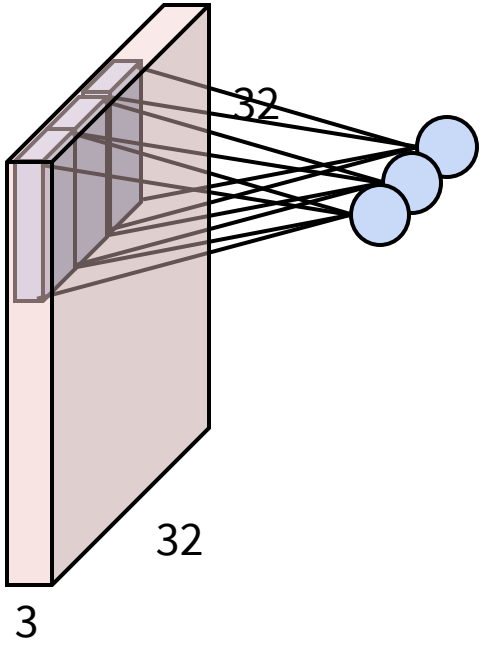
Convolution Layer



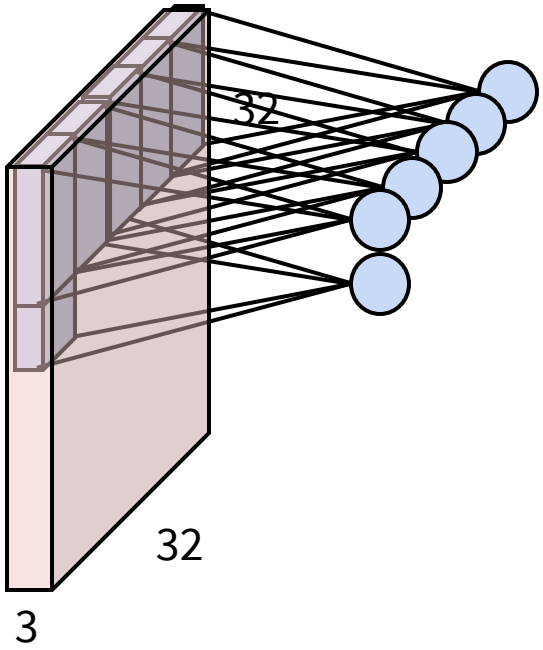
Convolution Layer



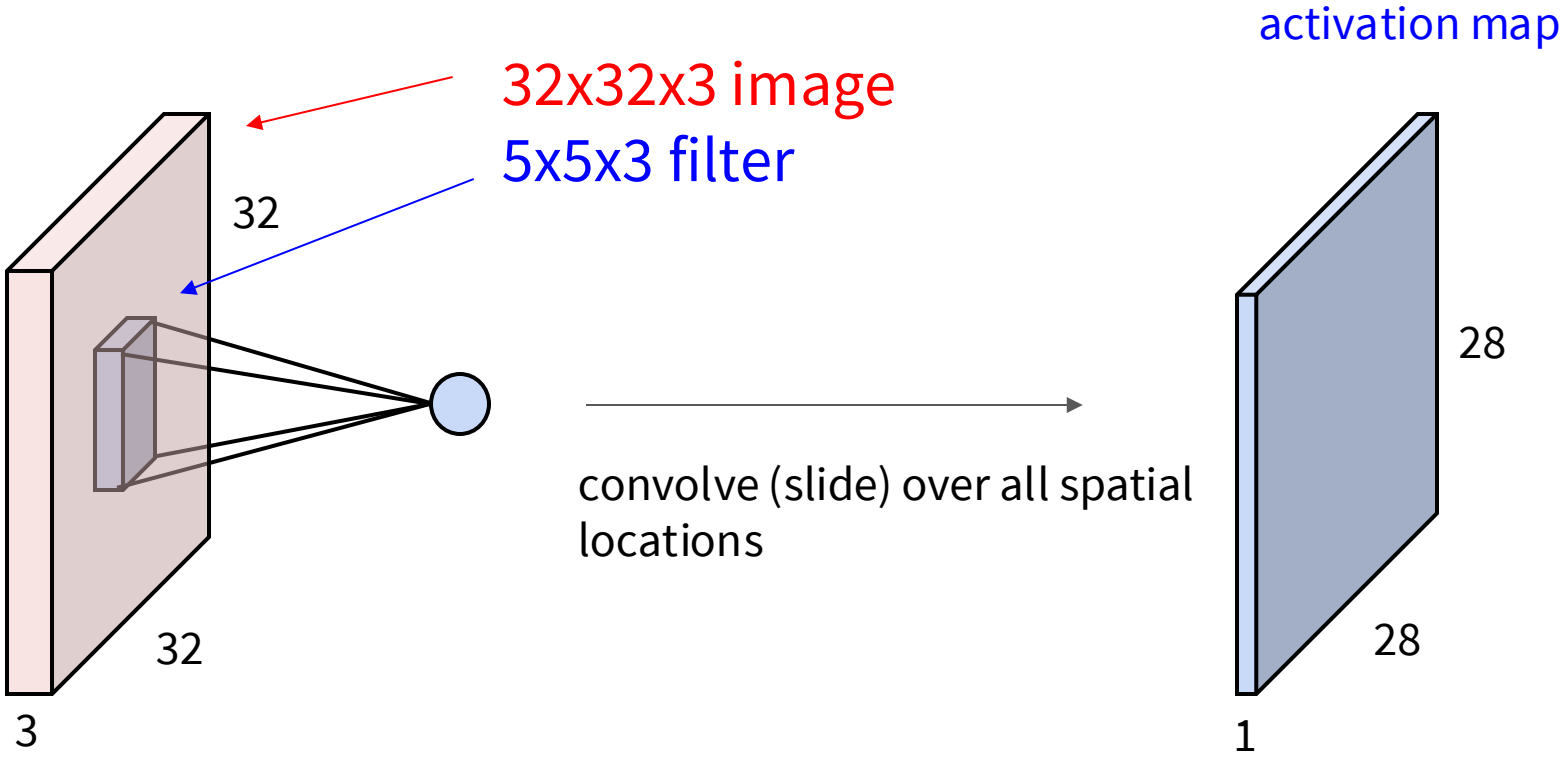
Convolution Layer



Convolution Layer

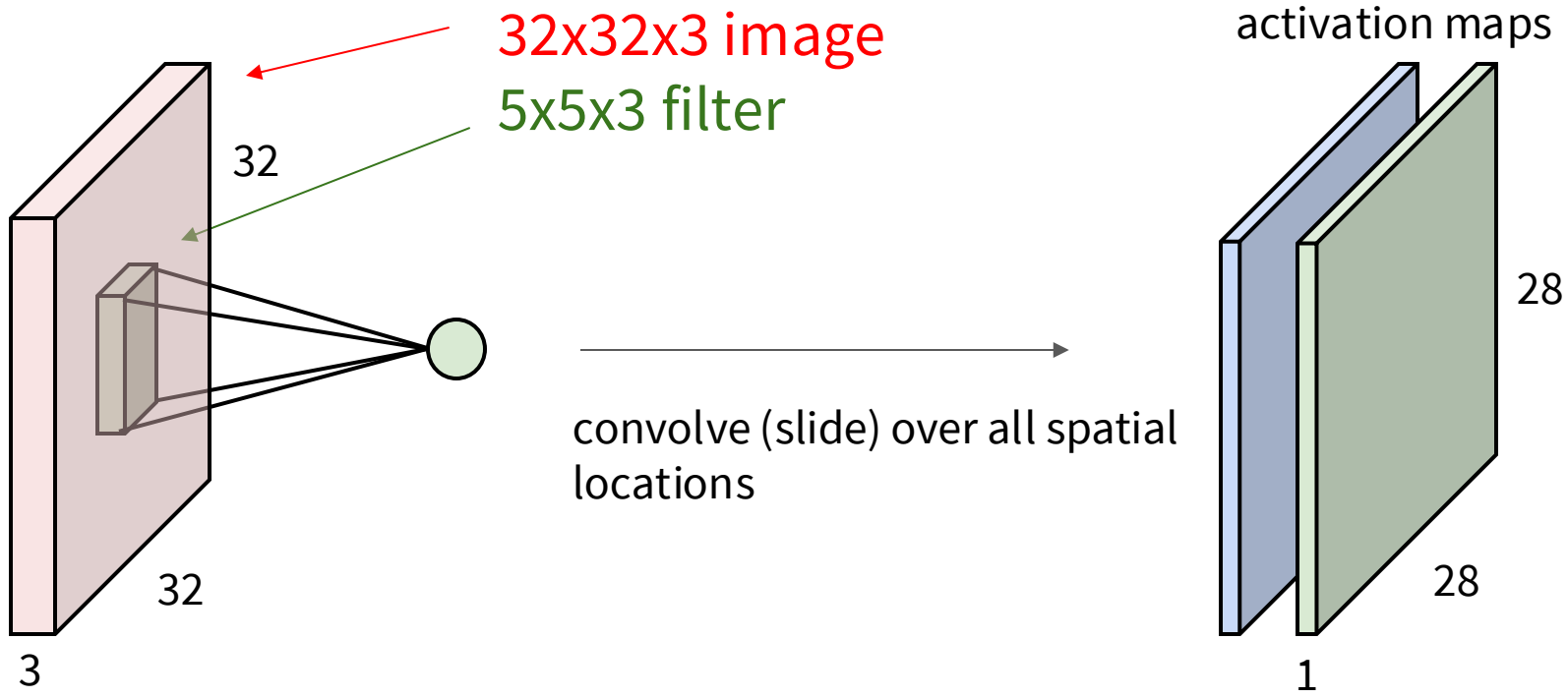


Convolution Layer

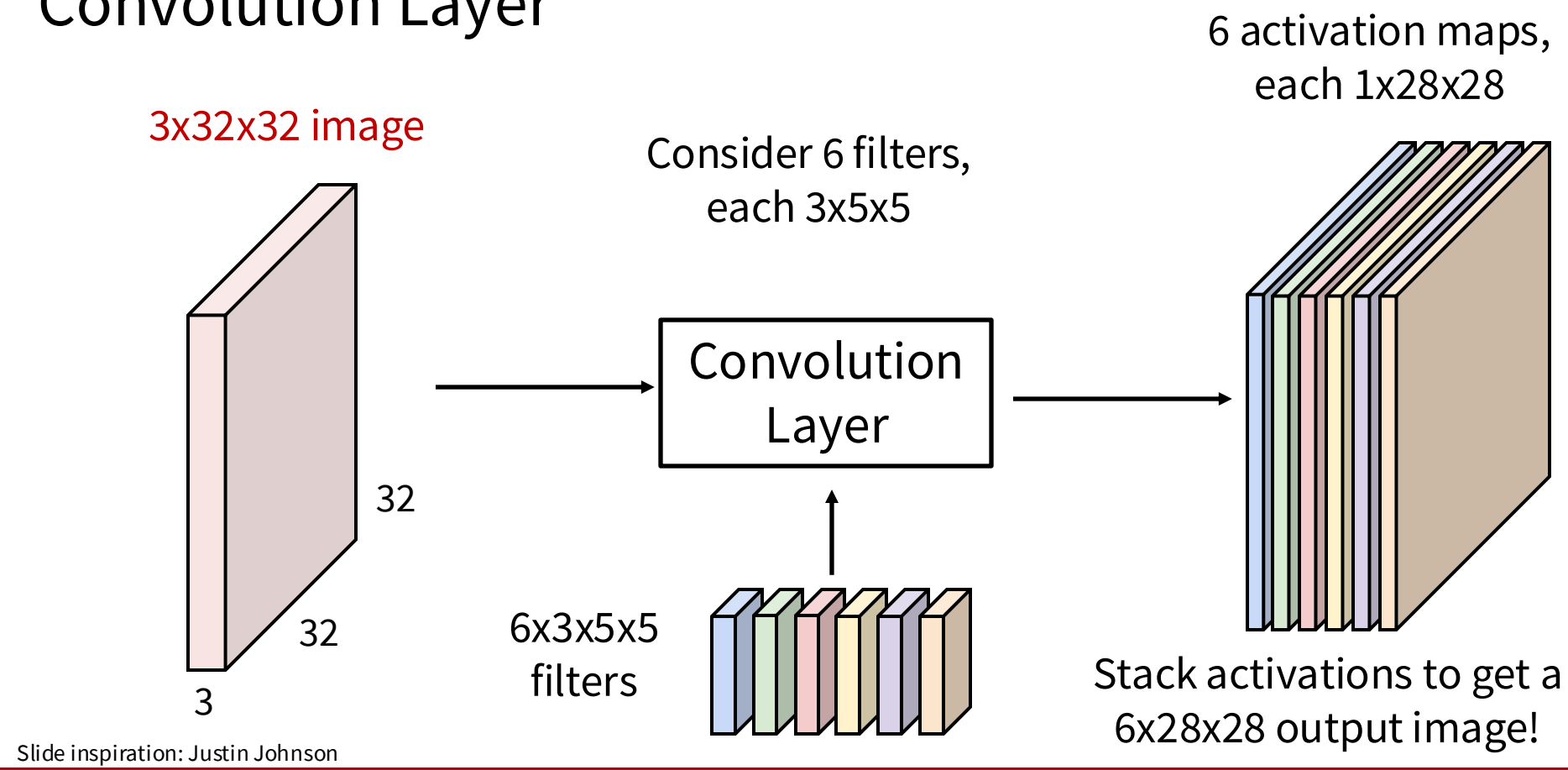


Convolution Layer

consider a second, green filter

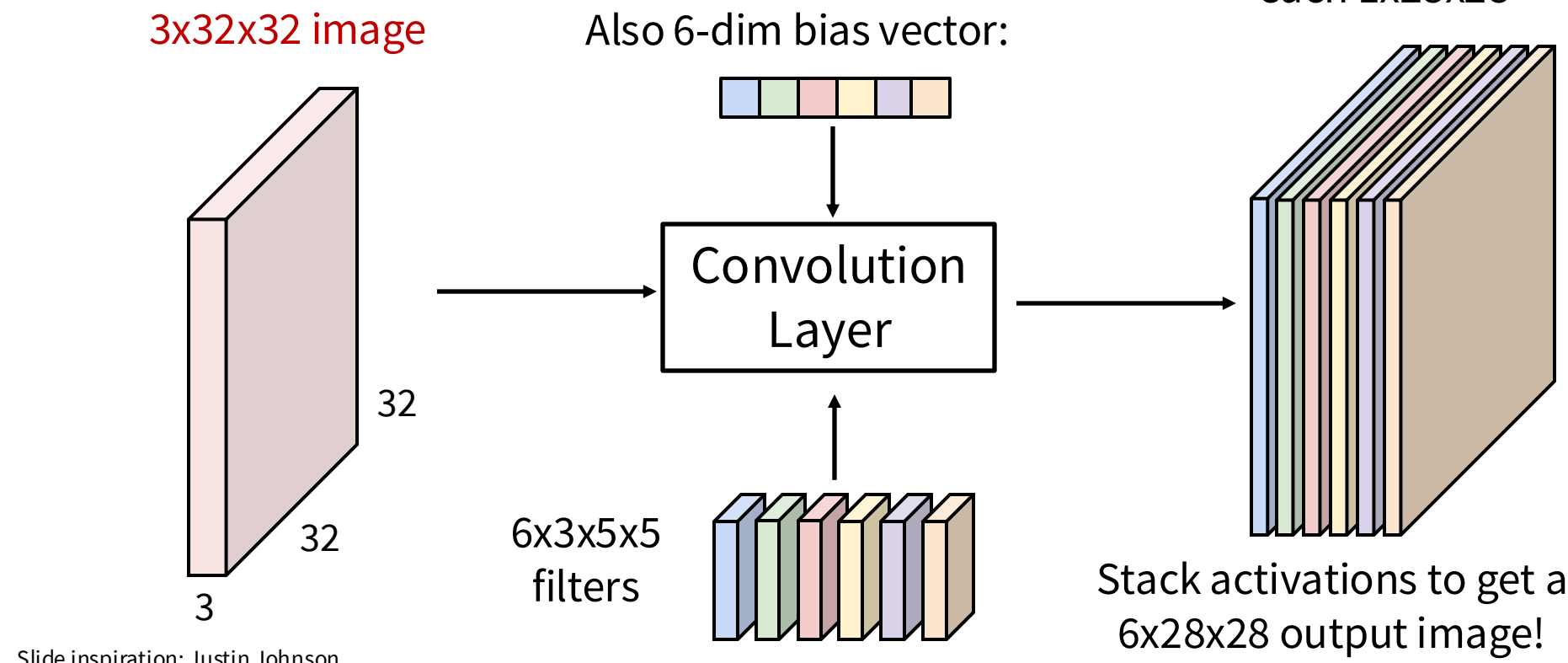


Convolution Layer



Slide inspiration: Justin Johnson

Convolution Layer



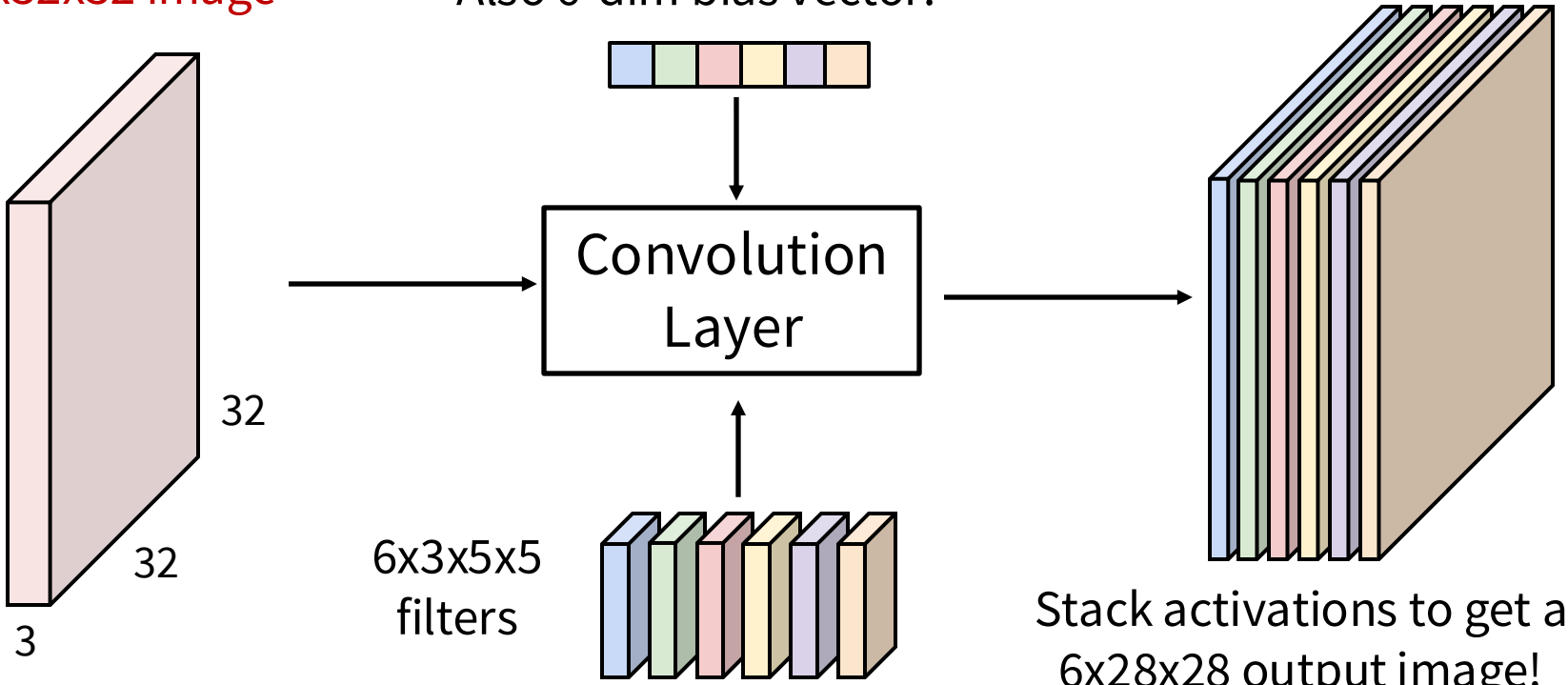
Slide inspiration: Justin Johnson

Convolution Layer

28x28 grid, at each point a 6-dim vector

3x32x32 image

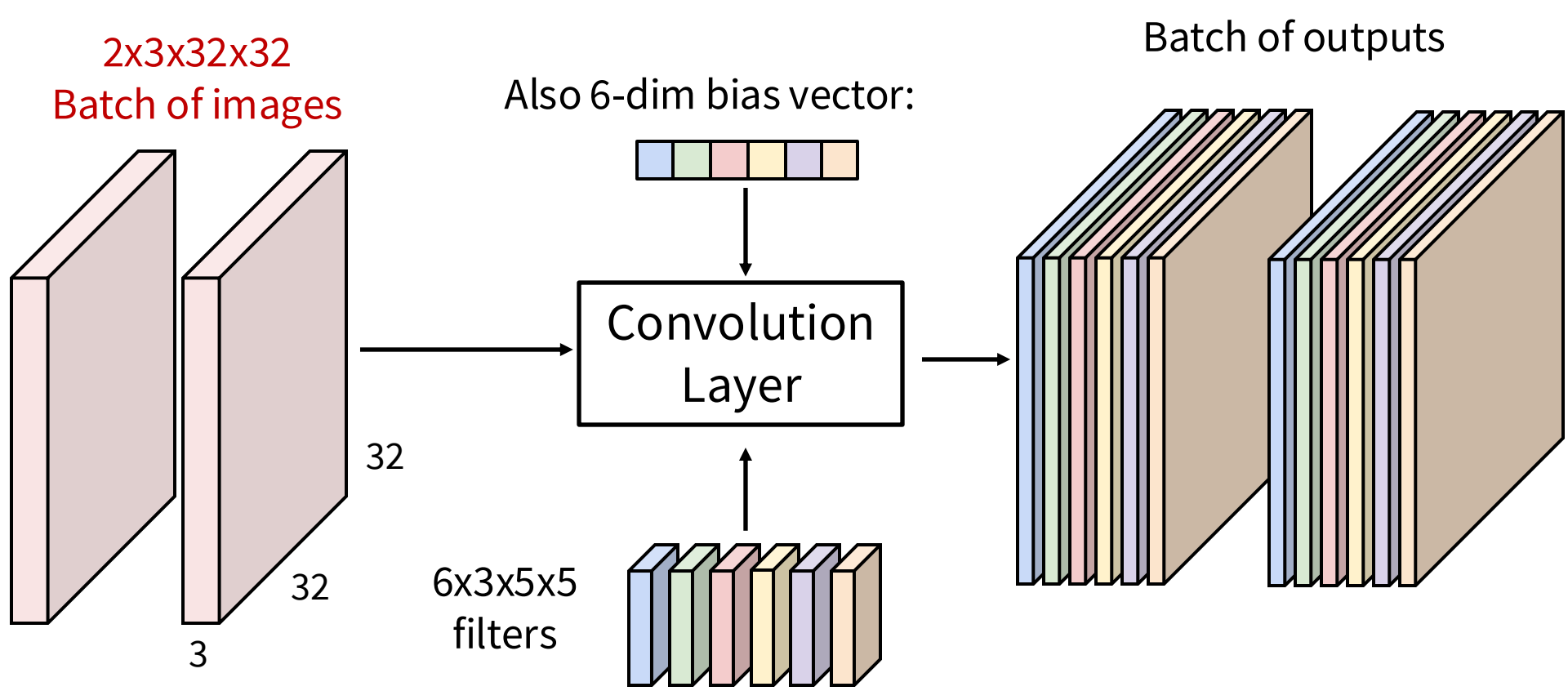
Also 6-dim bias vector:



Stack activations to get a 6x28x28 output image!

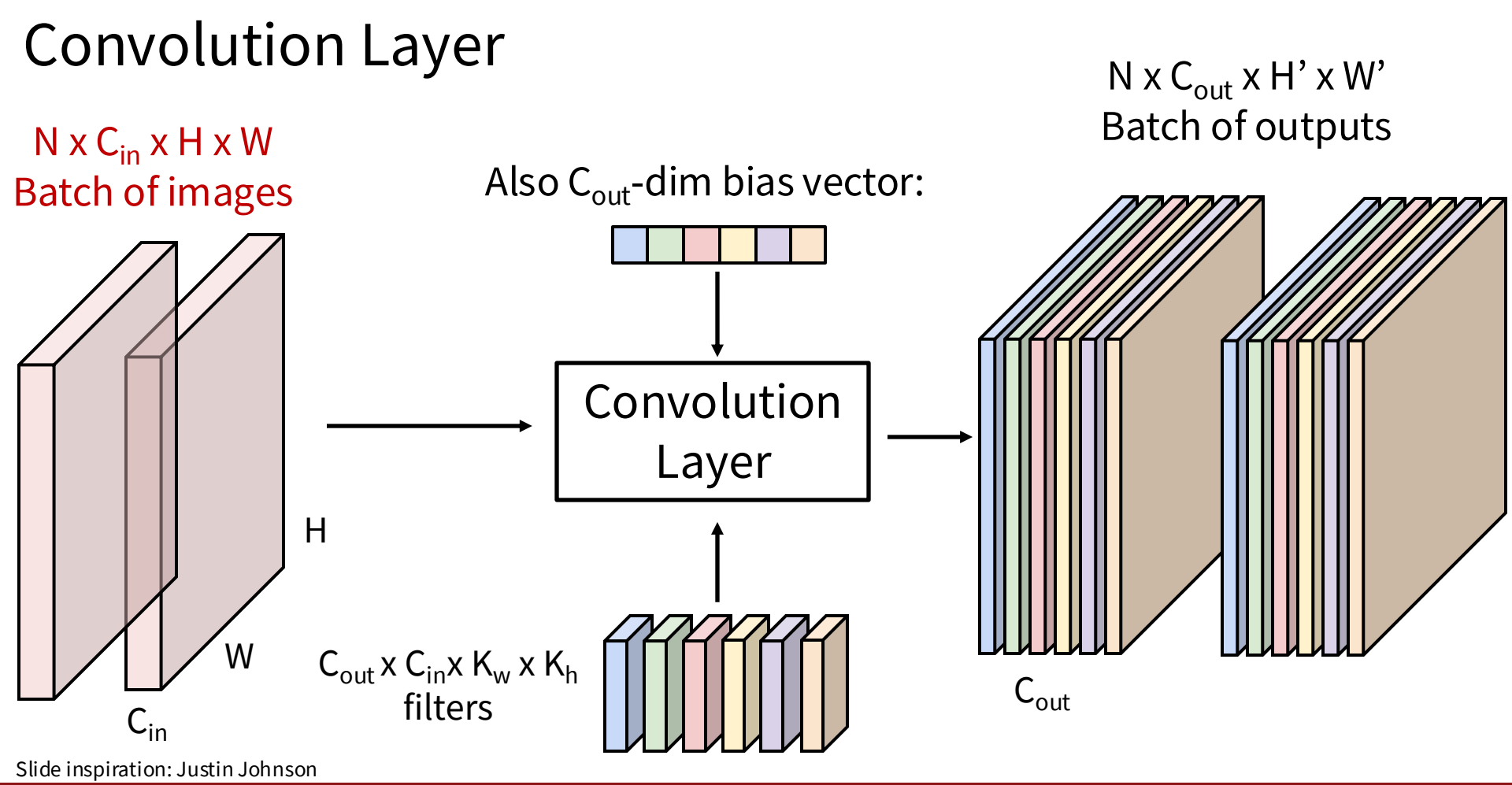
Slide inspiration: Justin Johnson

Convolution Layer



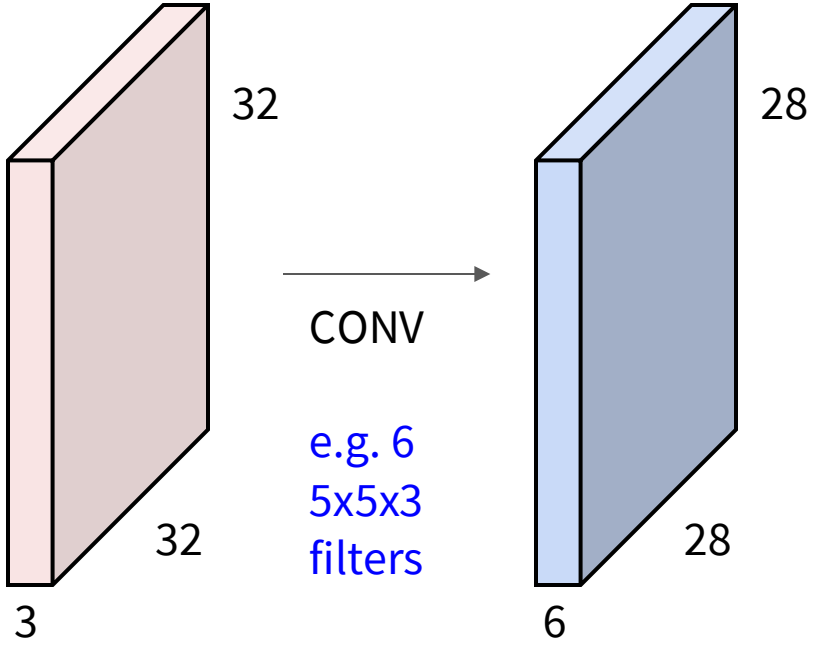
Slide inspiration: Justin Johnson

Convolution Layer

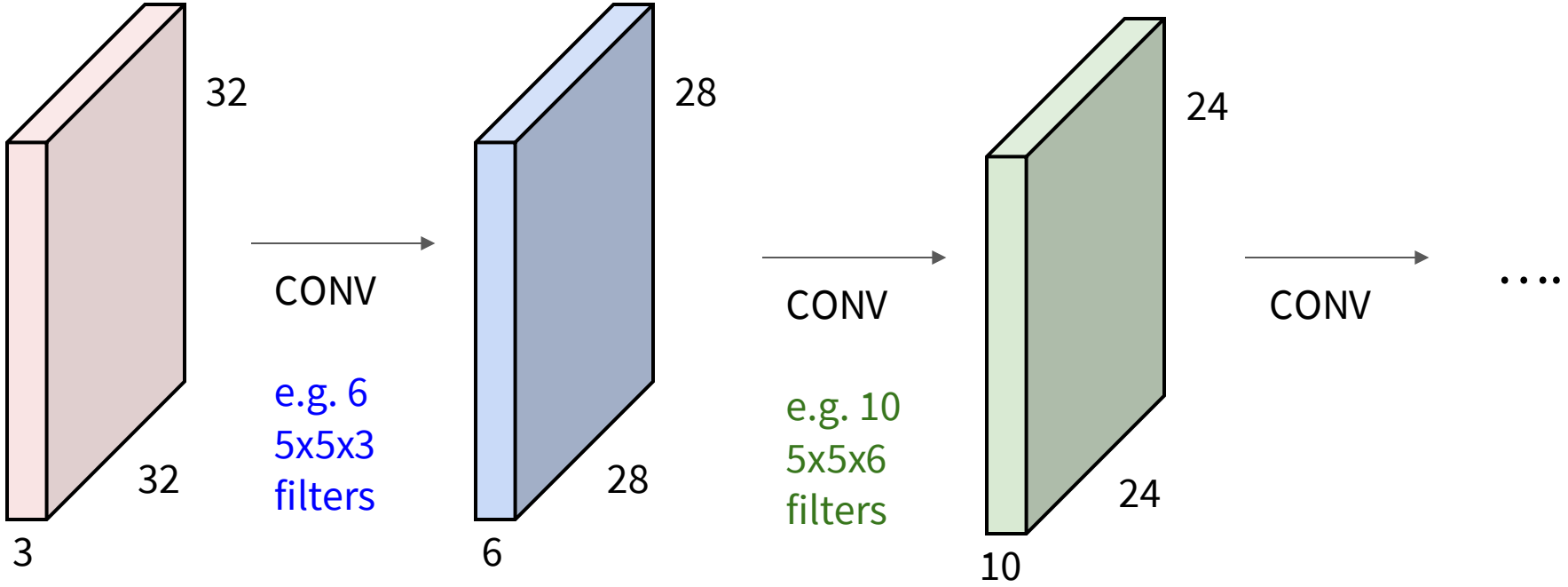


Slide inspiration: Justin Johnson

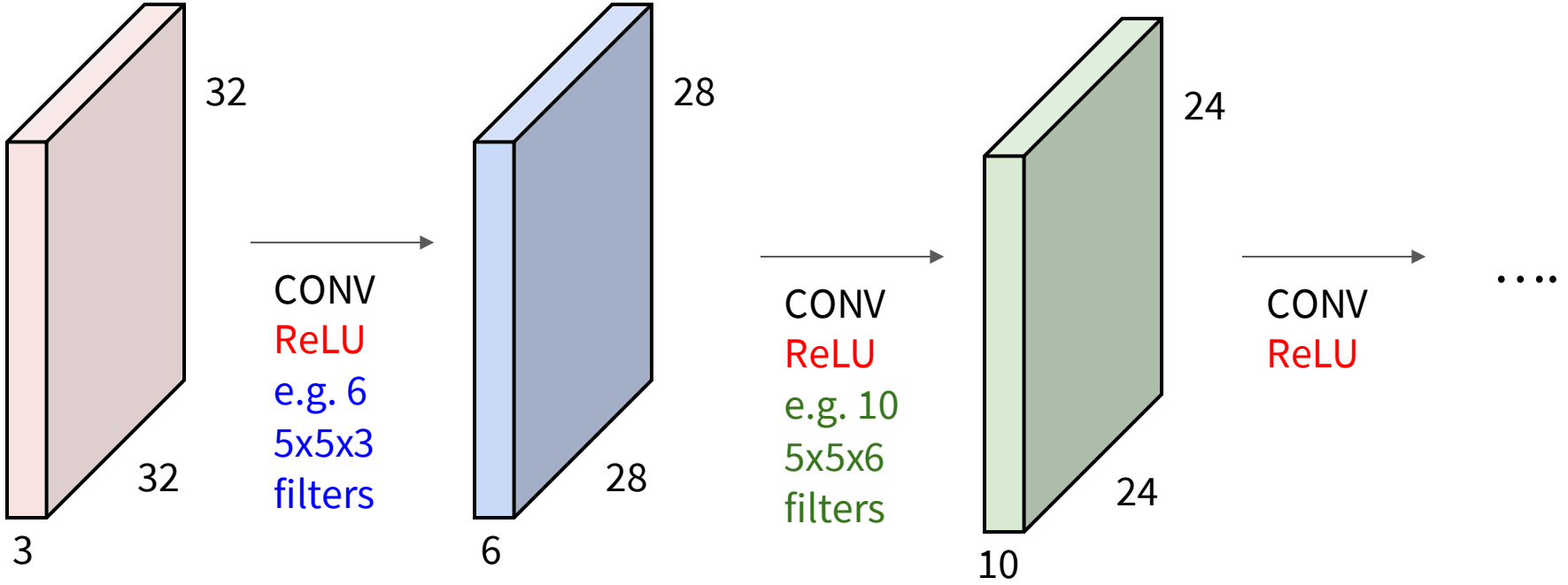
A ConvNet is a neural network with Conv layers



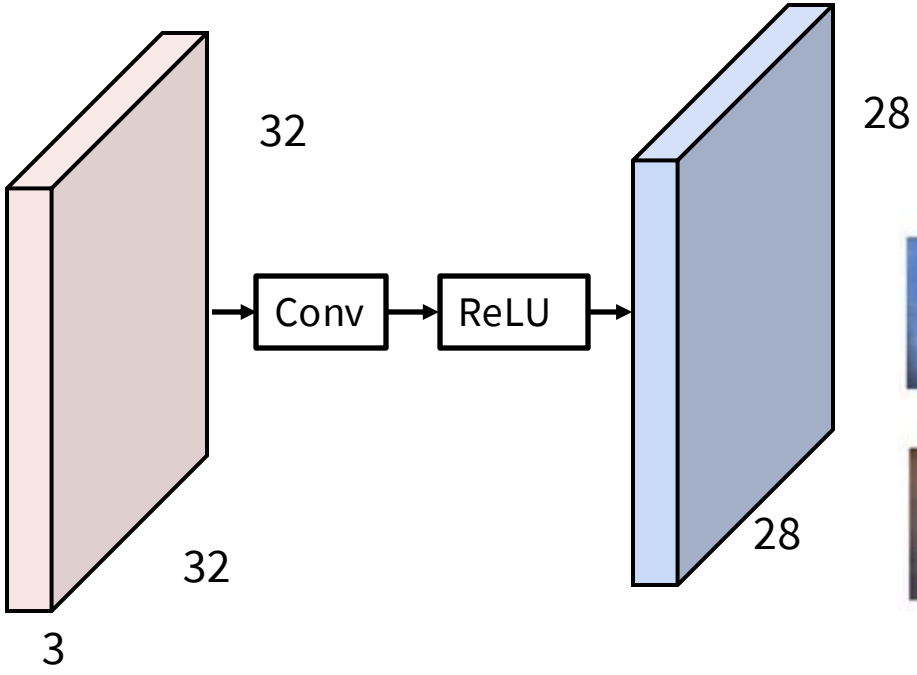
A ConvNet is a neural network with Conv layers



A ConvNet is a neural network with Conv layers with activation functions!

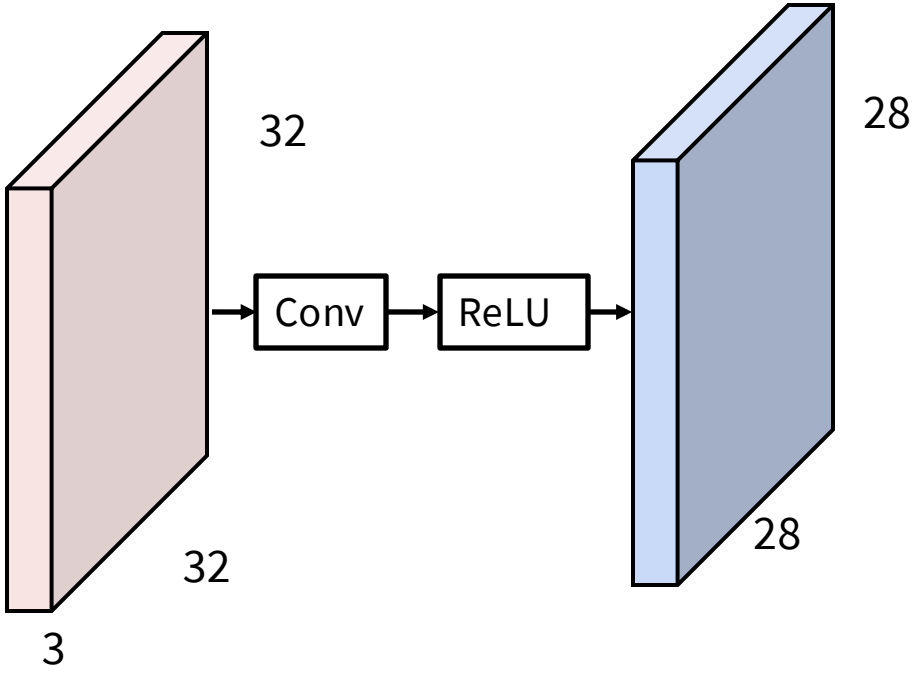


What do Conv filters learn?



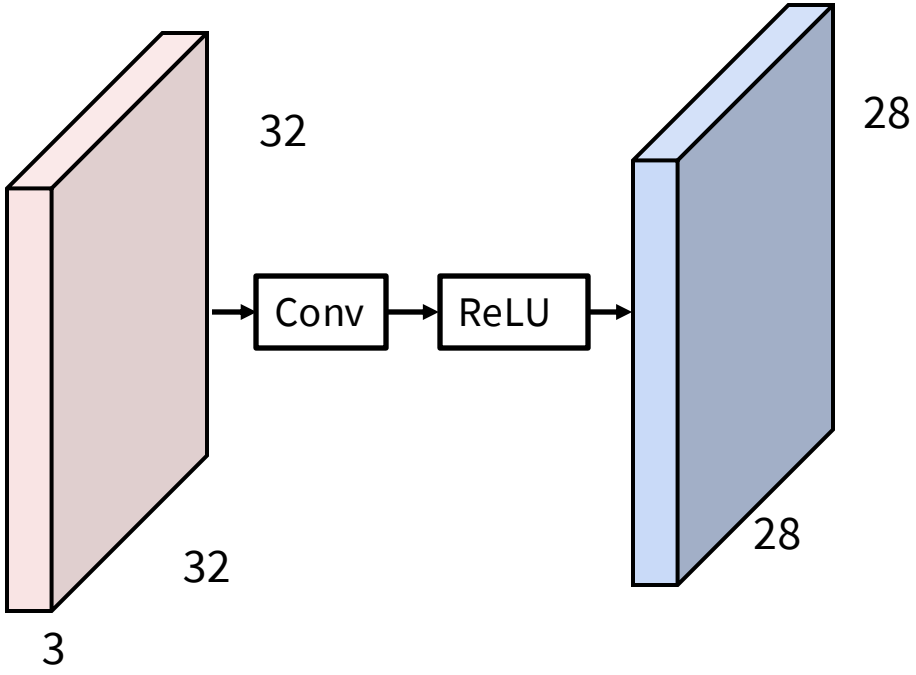
Linear classifier: One template per class

What do Conv filters learn?

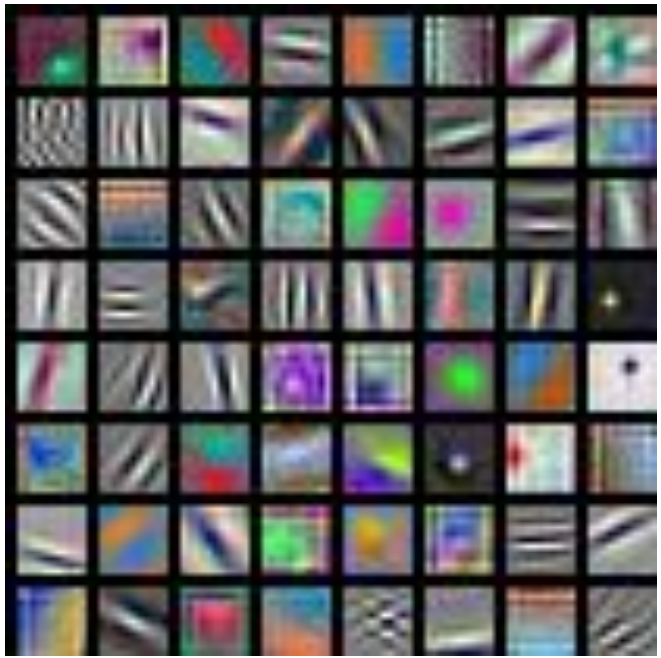


MLP: Bank of whole-image templates

What do Conv filters learn?

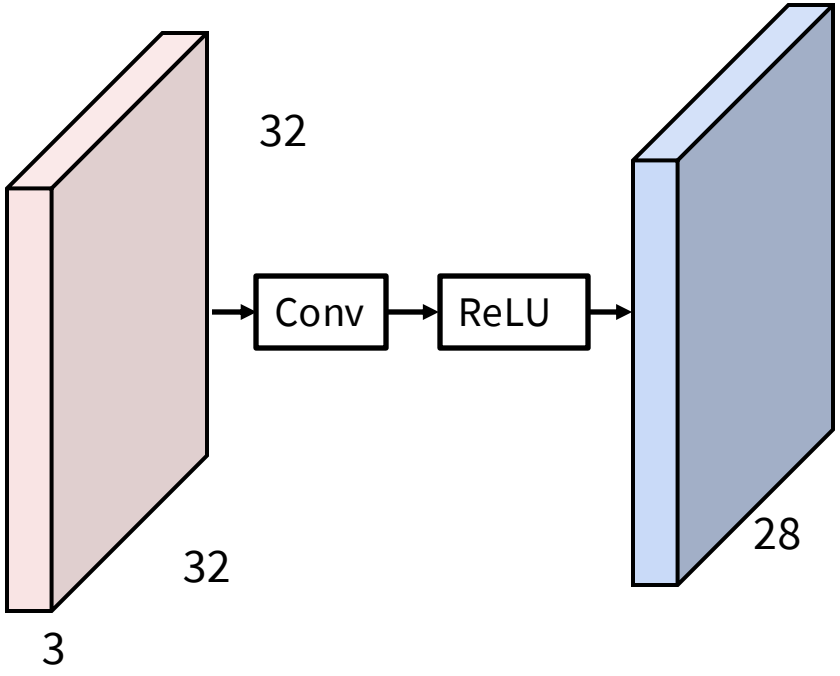


First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)

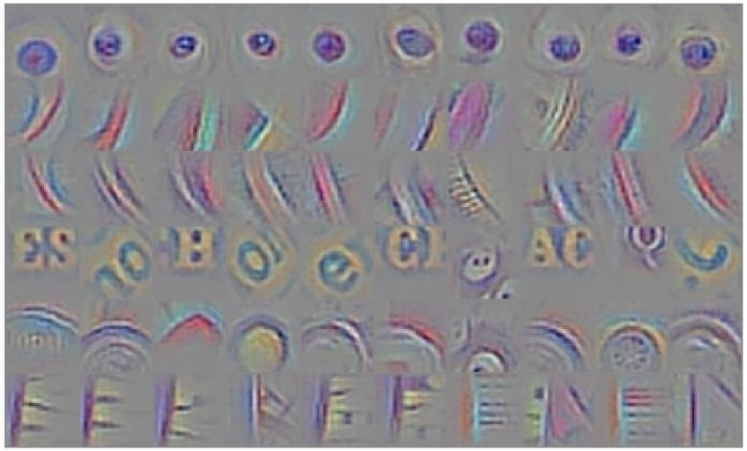


AlexNet: 64 filters, each 3x11x11

What do Conv filters learn?



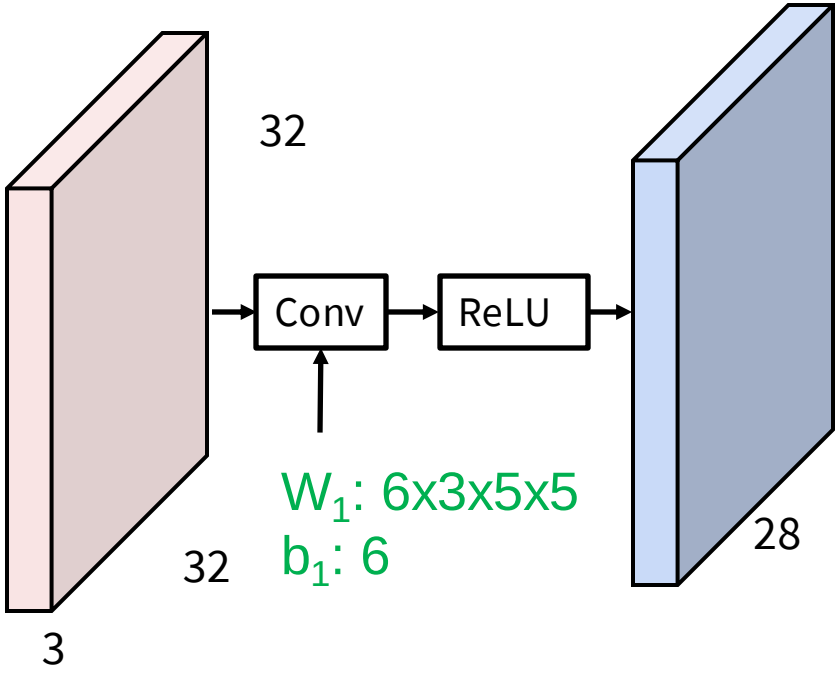
Deeper conv layers: Harder to visualize
Tend to learn larger structures e.g. eyes, letters



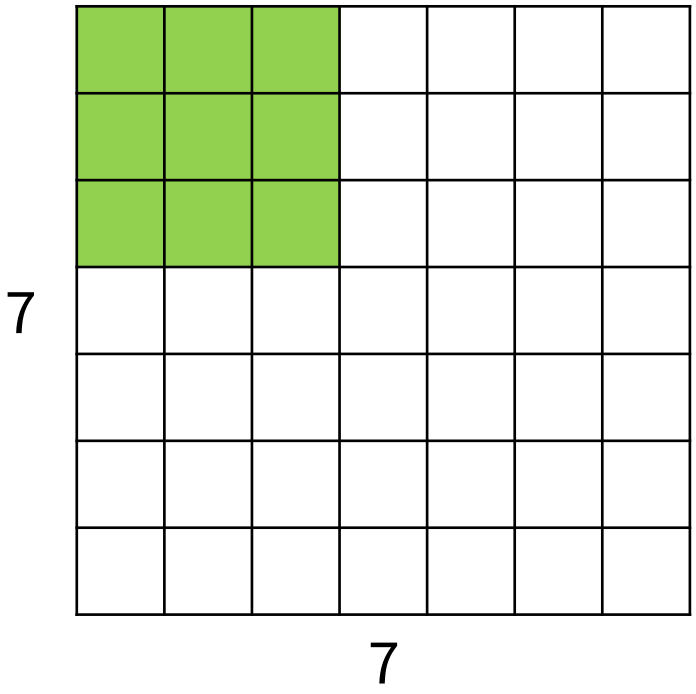
6th layer conv layer from an ImageNet model

Visualization from [Springenberg et al, ICLR 2015]

Convolution: Spatial Dimensions

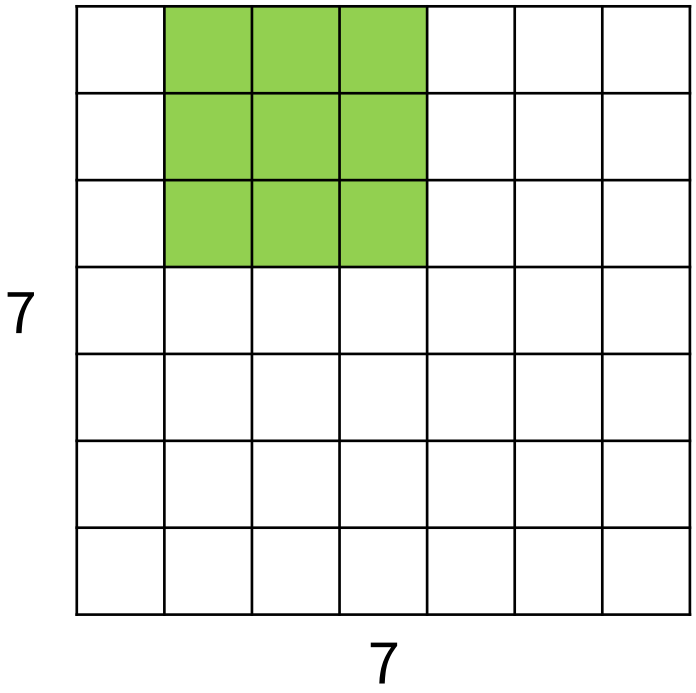


Convolution: Spatial Dimensions



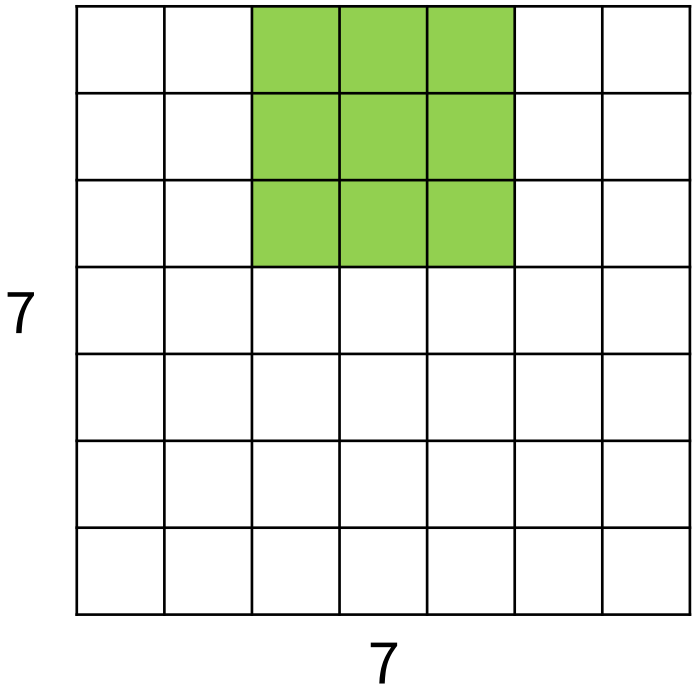
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



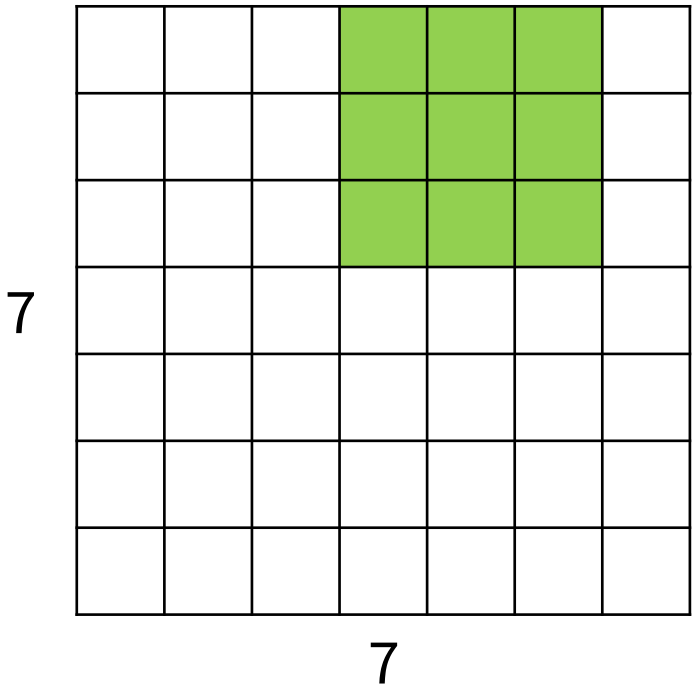
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



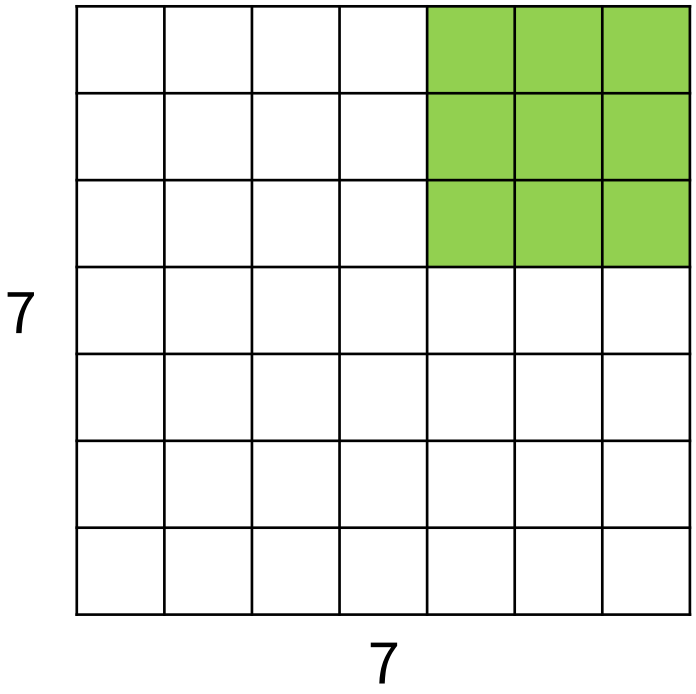
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



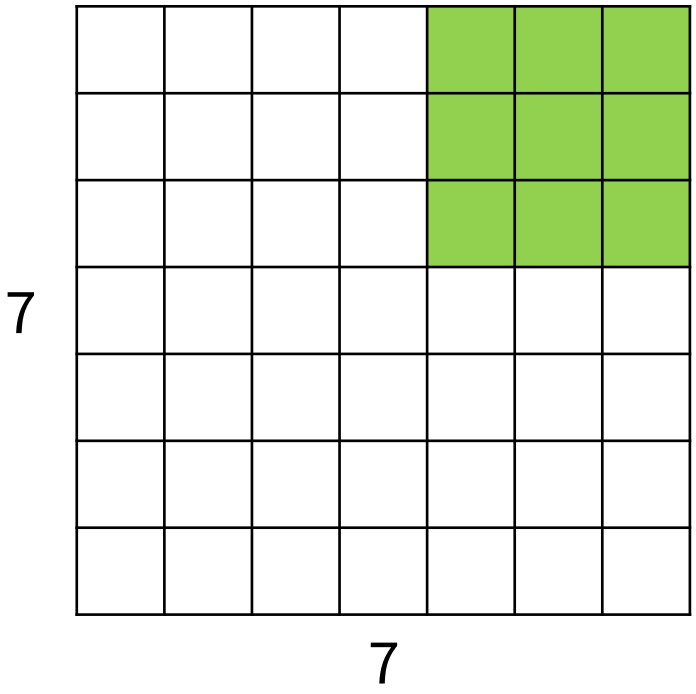
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



Input: 7x7
Filter: 3x3
Output: 5x5

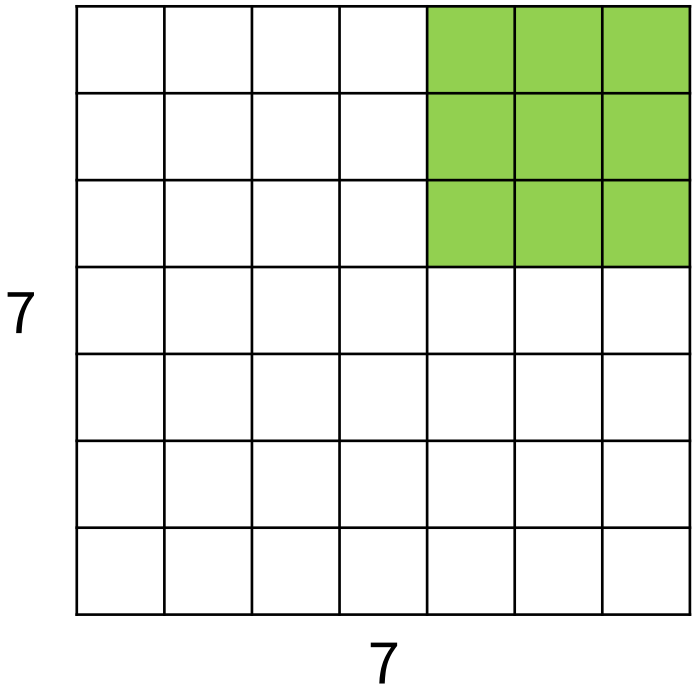
Convolution: Spatial Dimensions



Input: 7x7
Filter: 3x3
Output: 5x5

In general
Input: W
Filter: K
Output: W – K + 1

Convolution: Spatial Dimensions

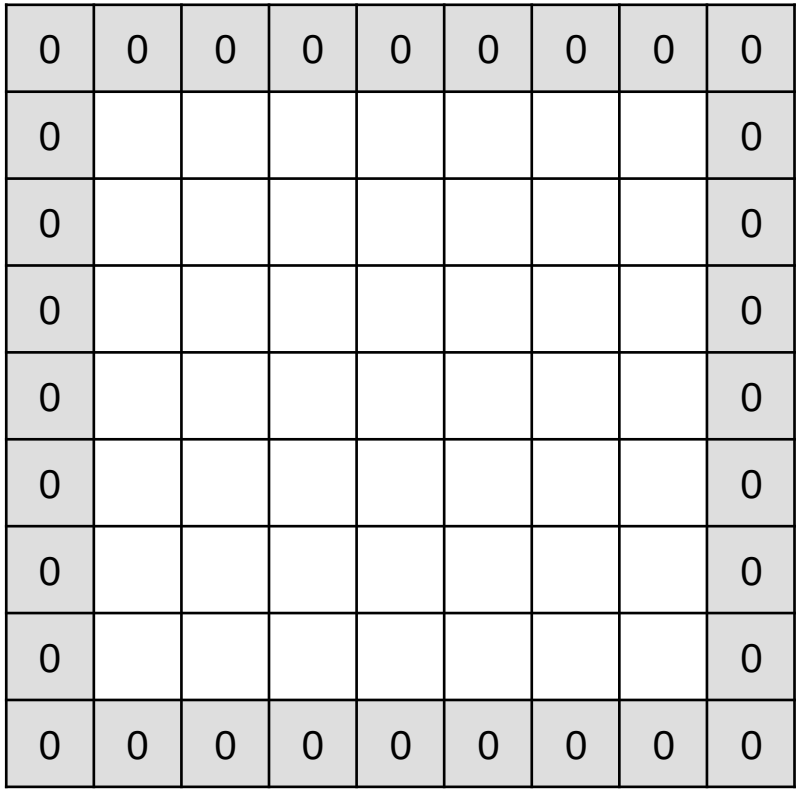


Input: 7x7
Filter: 3x3
Output: 5x5

Problem: Feature maps shrink with each layer!

In general
Input: W
Filter: K
Output: W – K + 1

Convolution: Spatial Dimensions



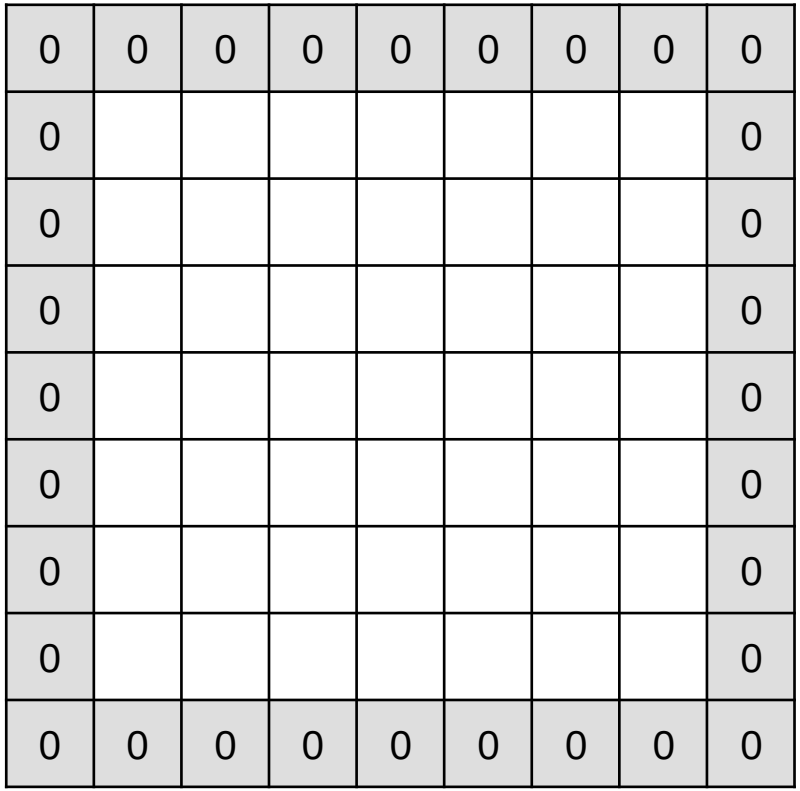
Input: 7x7
Filter: 3x3
Output: 5x5

Problem: Feature maps shrink with each layer!

In general
Input: W
Filter: K
Padding: P
Output: W – K + 1 + 2P

Solution: Add **padding** around the input before sliding the filter

Convolution: Spatial Dimensions



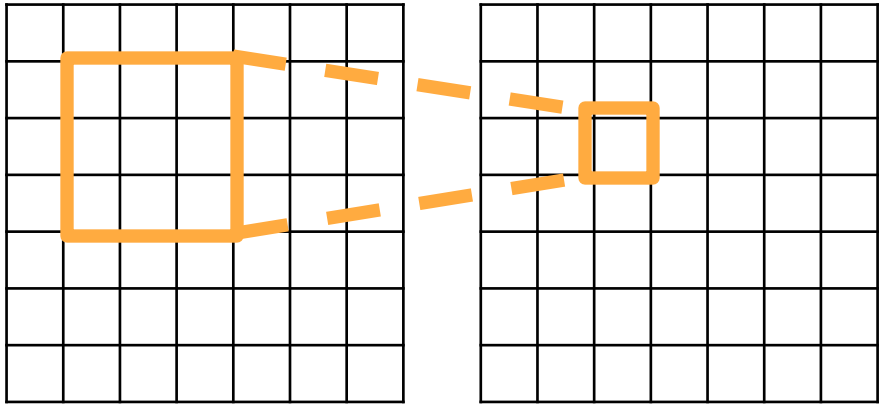
Input: 7x7
Filter: 3x3
Output: 5x5

In general
Input: W
Filter: K
Padding: P
Output: W – K + 1 + 2P

Common setting:
P = (K – 1) / 2
Means output has same size as input

Receptive Fields

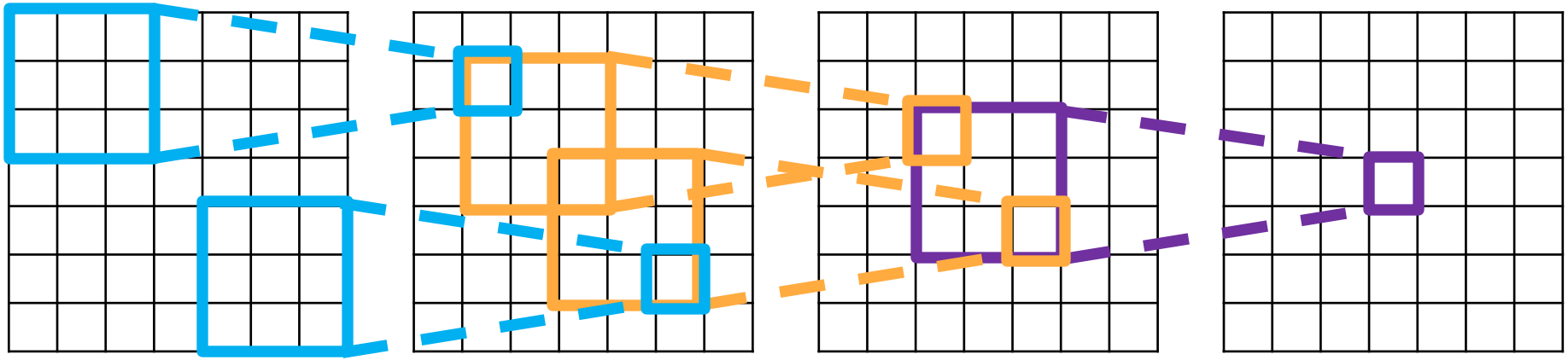
For convolution with **kernel size K**, each element in the output depends on a K x K receptive field in the input



Slide inspiration: Justin Johnson

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)



Input

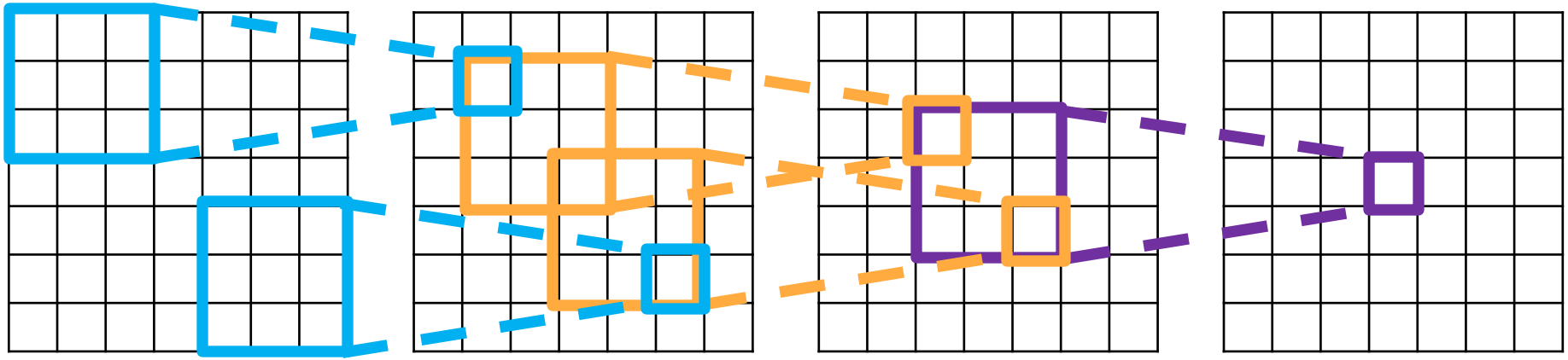
Output

Be careful – “receptive field in the input” vs. “receptive field in the previous layer”

Slide inspiration: Justin Johnson

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)



Input

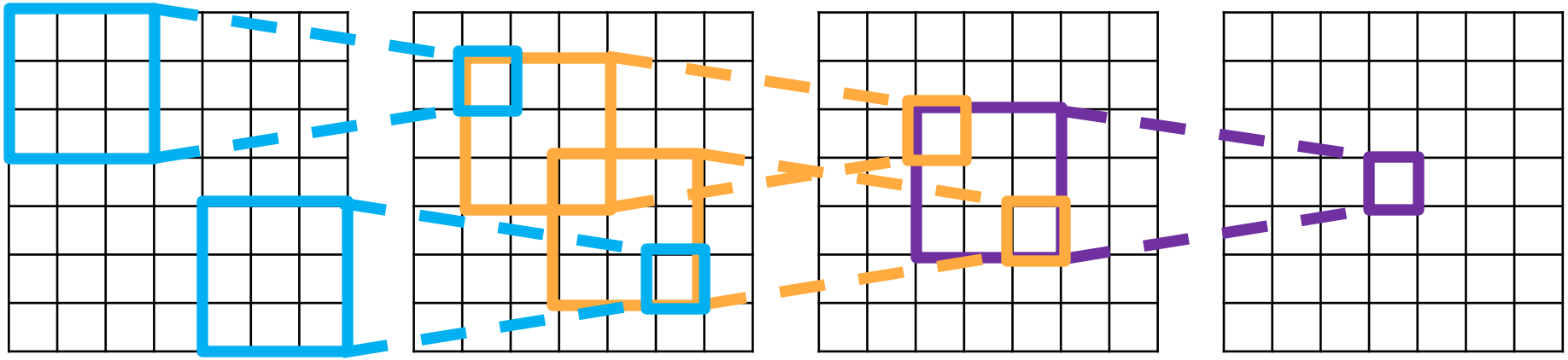
Problem: For large images we need many layers for each output to “see” the whole image image

Output

Slide inspiration: Justin Johnson

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)



Input

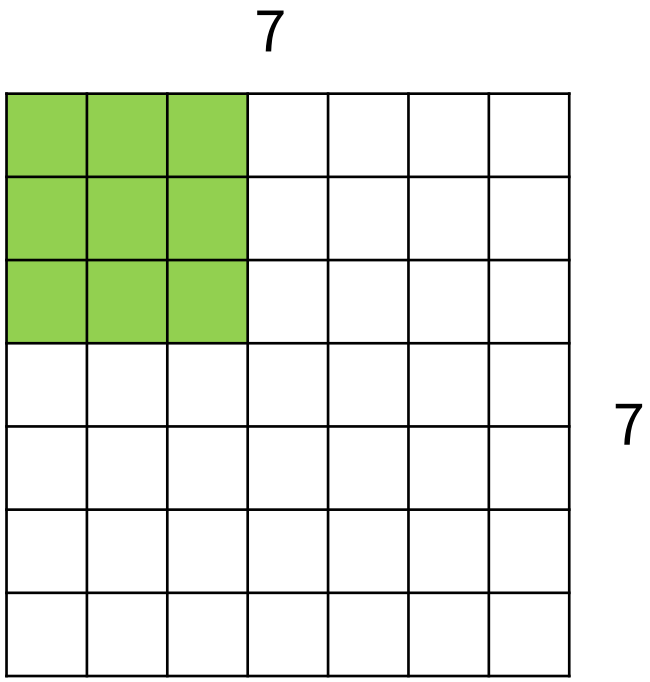
Output

Problem: For large images we need many layers for each output to “see” the whole image image

Solution: Downsample inside the network

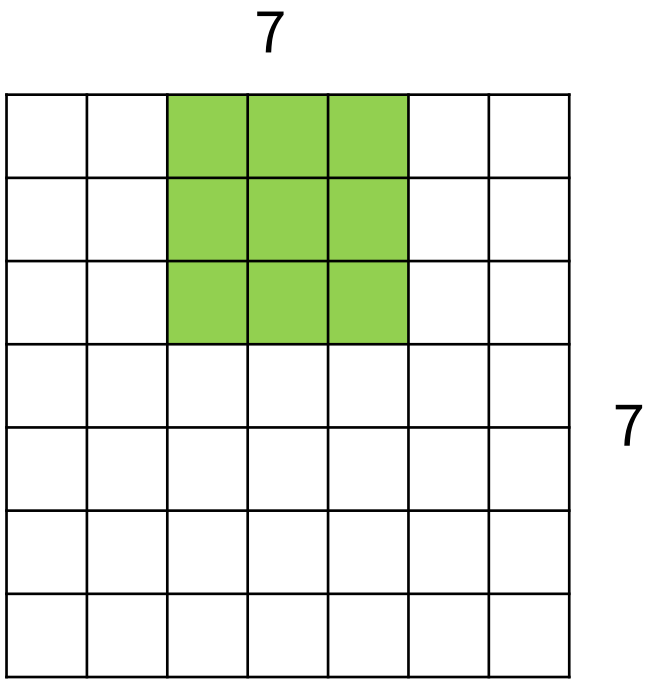
Slide inspiration: Justin Johnson

Strided Convolution



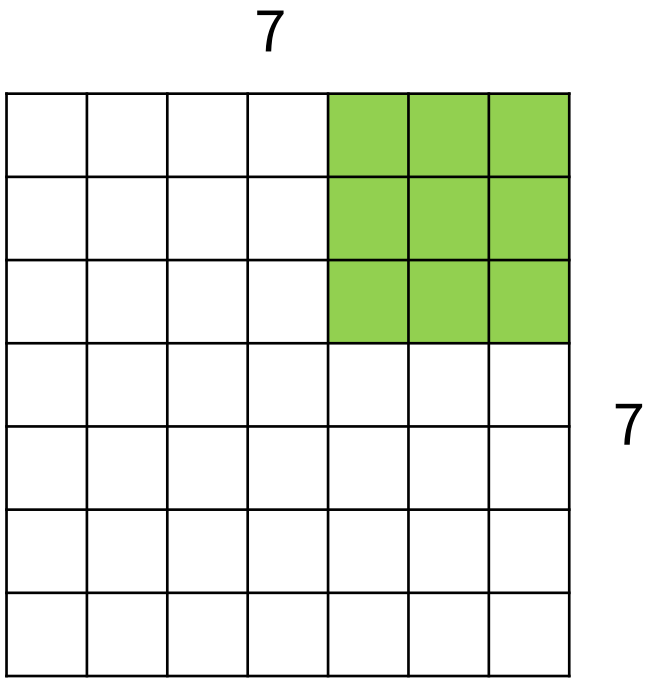
Input: 7x7
Filter: 3x3
Stride: 2

Strided Convolution



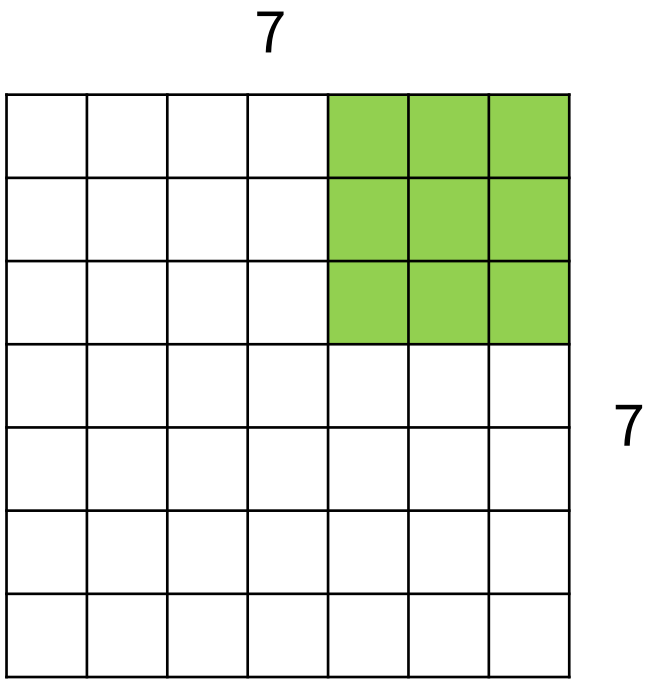
Input: 7x7
Filter: 3x3
Stride: 2

Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2
Output: 3x3

Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2
Output: 3x3

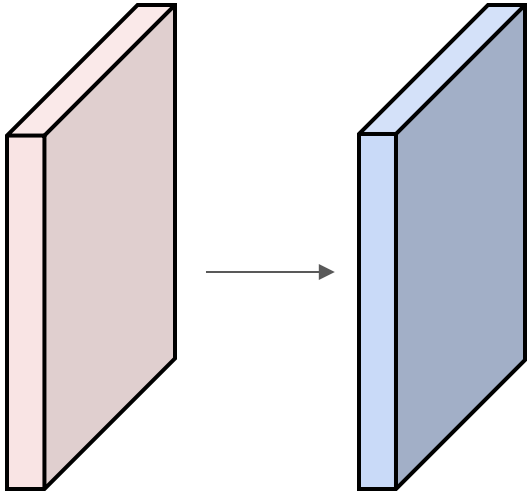
In general:
Input: W
Filter: K
Padding: P
Stride: S

→ Output:
(W – K + 2P) / S + 1

Convolution Example

Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

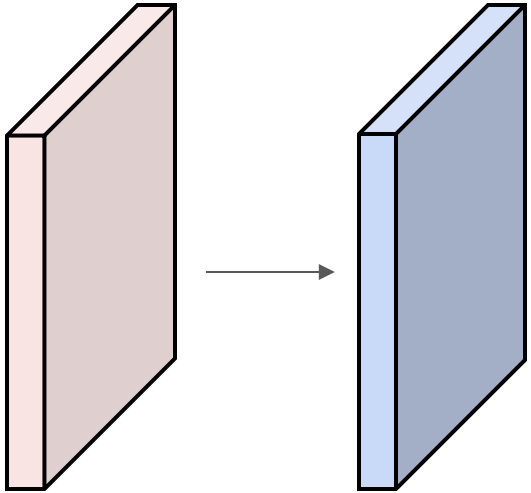
Output volume size: ?



Convolution Example

Input volume: 3 x **32** x **32**
10 **5x5** filters with stride **1**, pad **2**

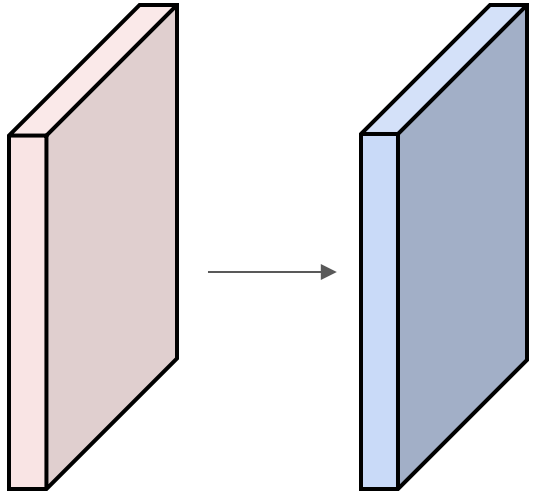
Output volume size: **10** x **32** x **32**
32 = (**32**+2*__2__-**5**)/**1**+1



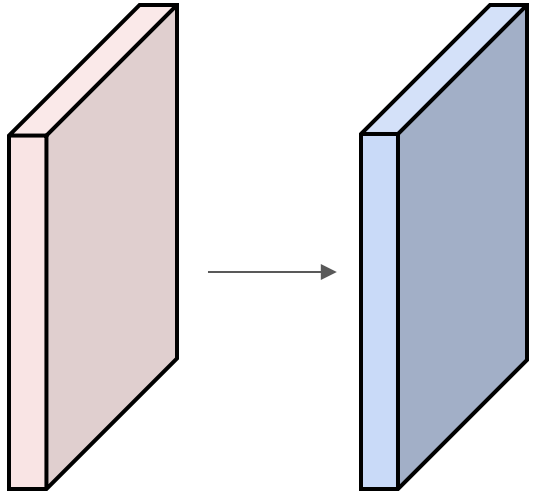
Convolution Example

Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: ?



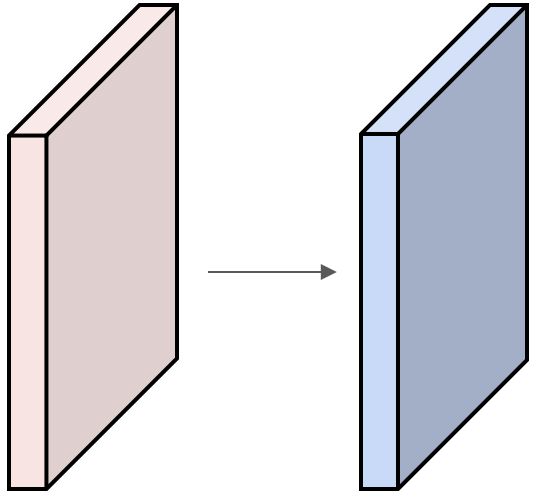
Convolution Example



Input volume: **3** x 32 x 32
10 **5**x**5** filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: 760
Parameters per filter: **3*****5*****5** + 1 (for bias) = **76**
10 filters, so total is **10** * **76** = **760**

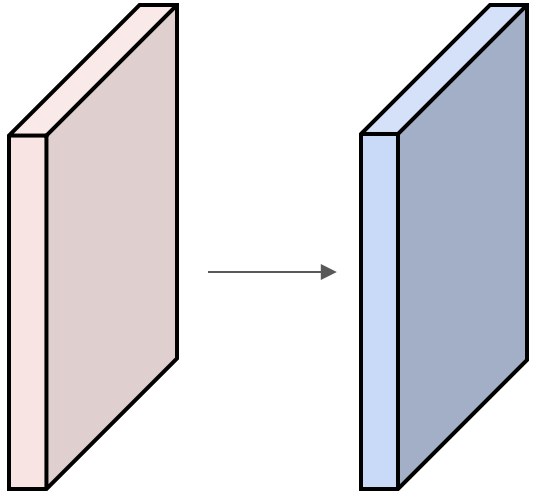
Convolution Example



Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: 760
Number of multiply-add operations?

Convolution Example



Input volume: **3** x 32 x 32
10 **5x5** filters with stride 1, pad 2

Output volume size: **10 x 32 x 32**
Number of learnable parameters: 760
Number of multiply-add operations: **768,000**
10*32*32 = 10,240 outputs
Each output is the inner product of two **3**x**5x5** tensors (75 elems)
Total = 75*10240 = **768K**

Convolution Summary

Input: C_{in} x H x W
Hyperparameters:
- **Kernel size**: K_H x K_W
- **Number filters**: C_{out}
- **Padding**: P
- **Stride**: S

Weight matrix: C_{out} x C_{in} x K_H x K_W
giving C_{out} filters of size C_{in} x K_H x K_W
Bias vector: C_{out}
Output size: C_{out} x H' x W' where:
- H' = (H – K + 2P) / S + 1
- W' = (W – K + 2P) / S + 1

Common settings:
$K_H = K_W$ (Small square filters)
P = (K – 1) / 2 ("Same" padding)
C_{in}, C_{out} = 32, 64, 128, 256 (powers of 2)
K = 3, P = 1, S = 1 (3x3 conv)
K = 5, P = 2, S = 1 (5x5 conv)
K = 1, P = 0, S = 1 (1x1 conv)
K = 3, P = 1, S = 2 (Downsample by 2)

PyTorch Convolution Layer

CLASS torch.nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

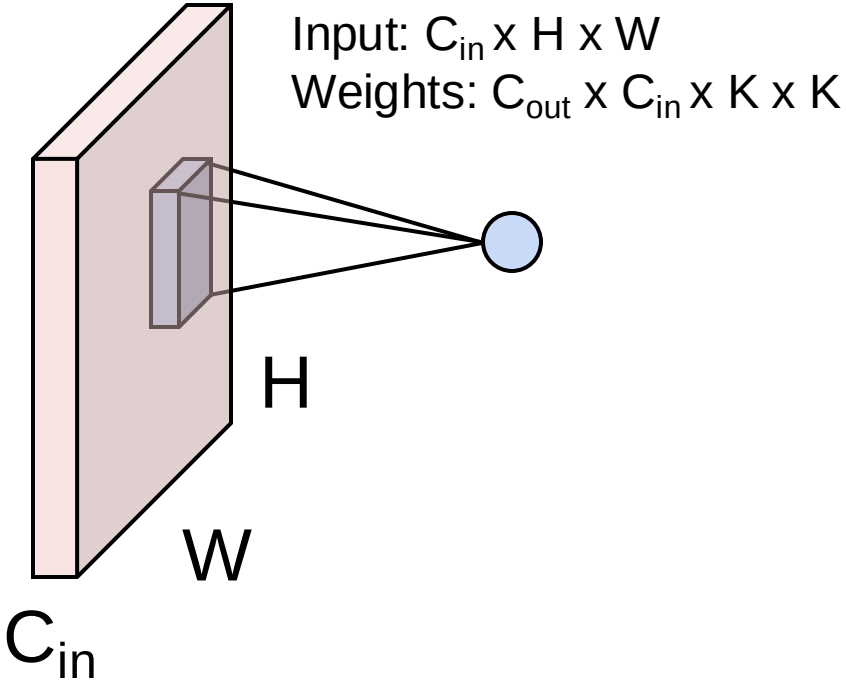
In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

We didn't talk about groups or dilation…

Other Types of Convolution

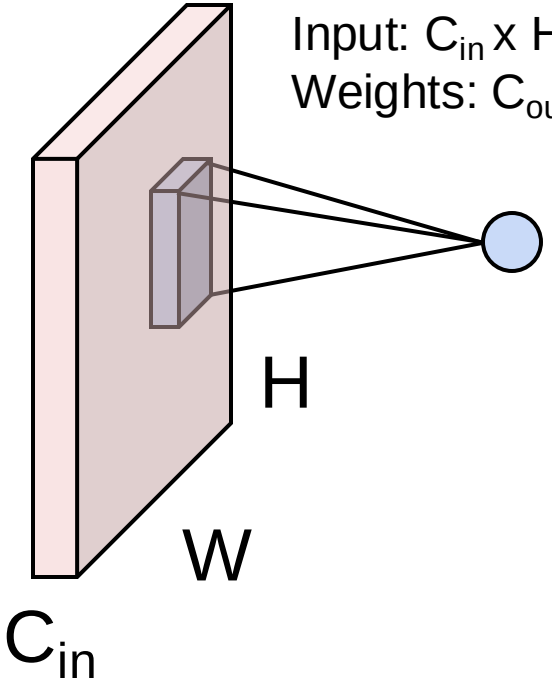
So far: 2D Convolution



Other Types of Convolution

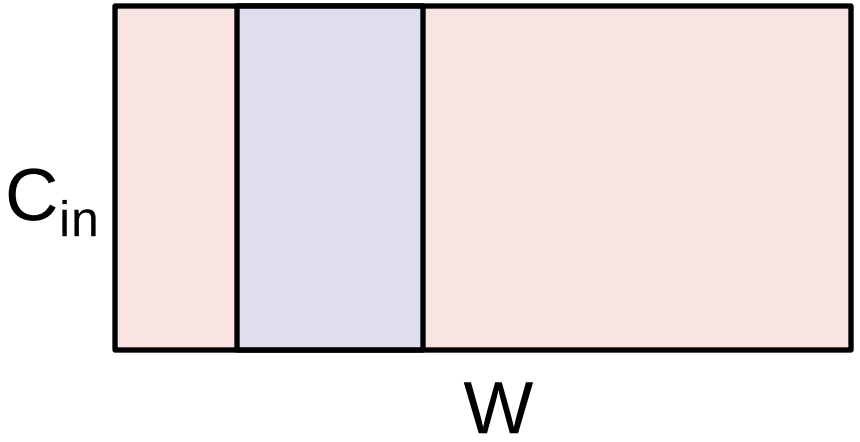
So far: 2D Convolution

Input: C_{in} x H x W
Weights: C_{out} x C_{in} x K x K



1D Convolution

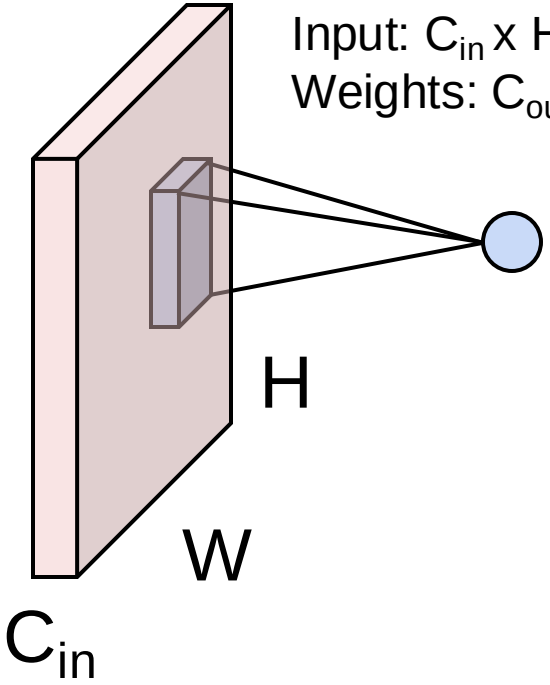
Input: C_{in} x W
Weights: C_{out} x C_{in} x K



Other Types of Convolution

So far: 2D Convolution

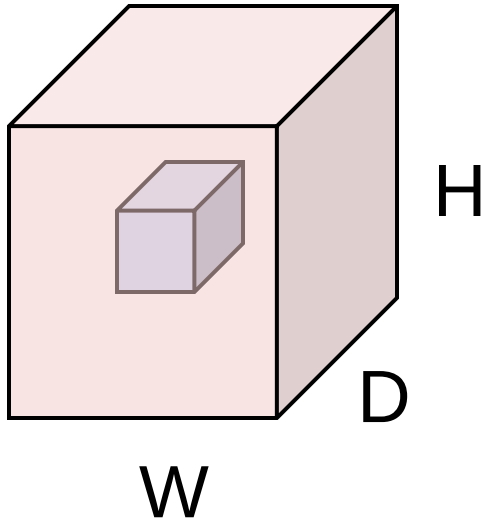
Input: C_{in} x H x W
Weights: C_{out} x C_{in} x K x K



3D Convolution

Input: C_{in} x H x W x D
Weights: C_{out} x C_{in} x K x K x K

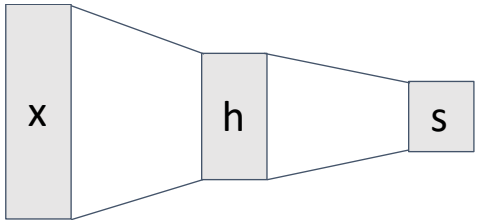
C_{in}-dim vector
at each point
in the volume



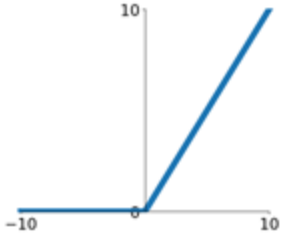
Convolutional Networks

We have already seen these

Fully-Connected Layer

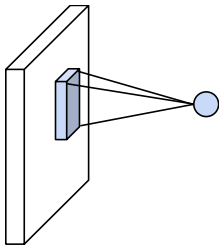


Activation Function



Today: Image-specific operators

Convolution Layer

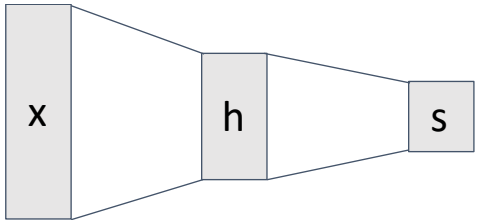


Pooling Layer

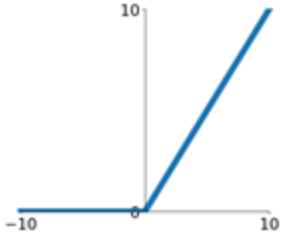
Convolutional Networks

We have already seen these

Fully-Connected Layer

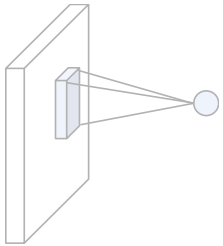


Activation Function

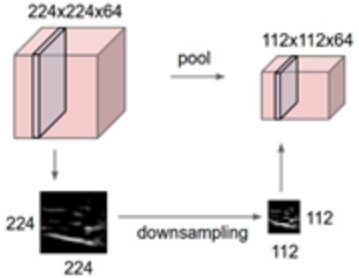


Today: Image-specific operators

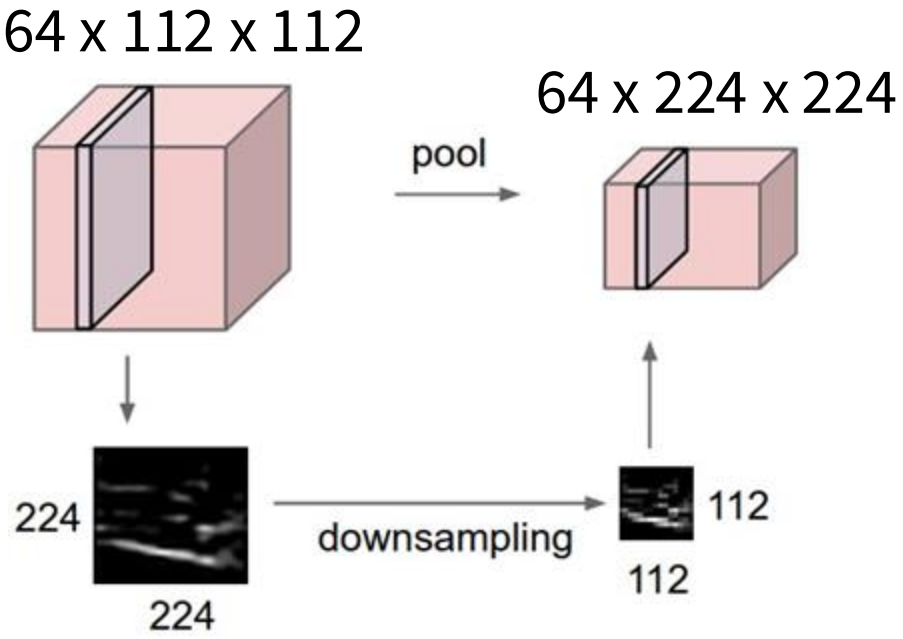
Convolution Layer



Pooling Layer



Pooling Layers: Another way to downsample



Given an input C x H x W,
downsample each 1 x H x W plane

Hyperparameters:
Kernel Size
Stride
Pooling function

Pooling Layers: Another way to downsample

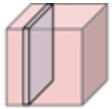
Single depth slice

1	1	2	4
5	**6**	7	**8**
3	2	1	0
1	2	3	**4**

x

y

64 x 224 x 224



Max pooling with 2x2 kernel size and stride 2

6	8
3	4

Gives **invariance** to small spatial shifts. No learnable parameters.

Pooling Summary

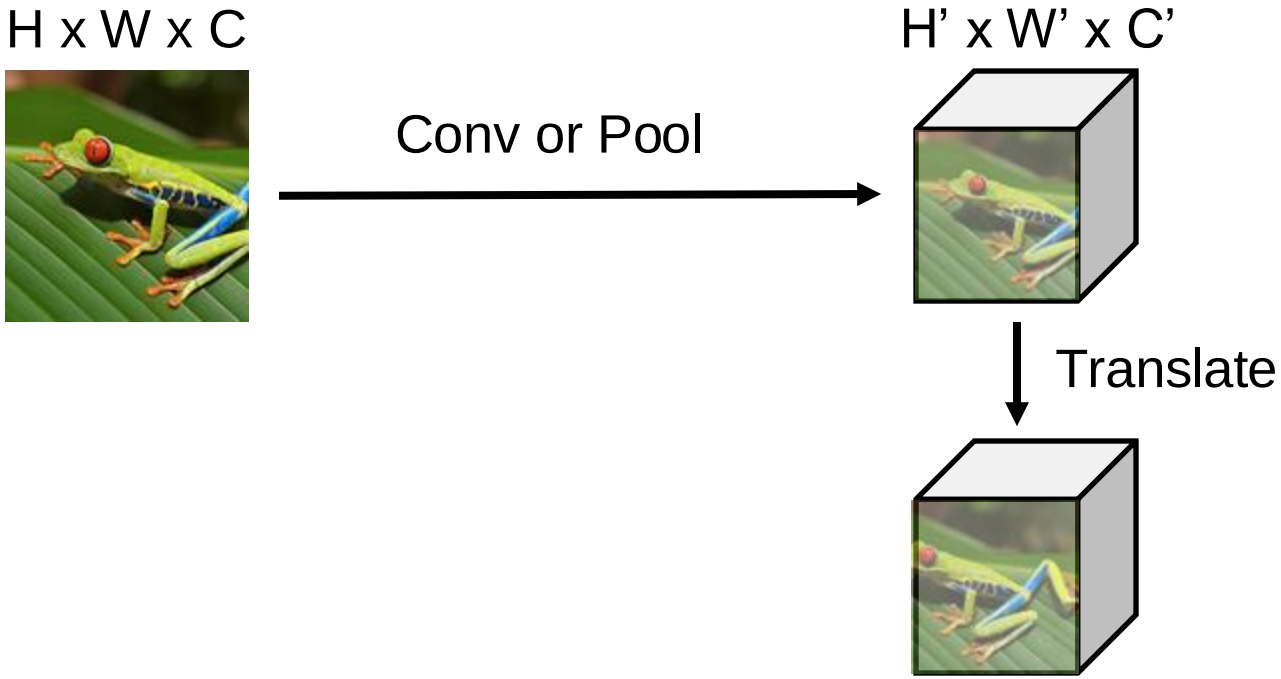
Input: C x H x W
Hyperparameters:
- **Kernel size**: K
- **Stride**: S
- **Pooling function**: max, avg

Output size: C x H' x W' where:
- H' = (H – K) / S + 1
- W' = (W – K) / S + 1

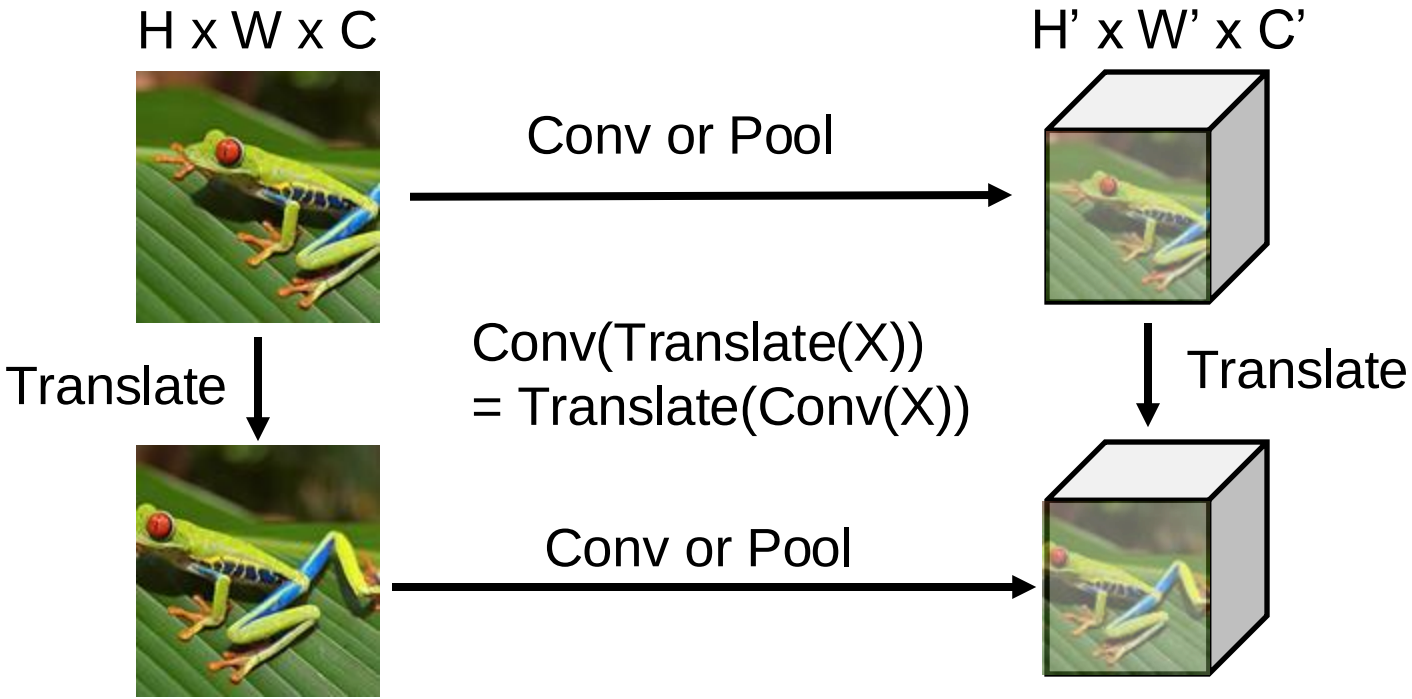
No learnable parameters

Common setting:
max, K=2, S=2 => Gives 2x downsampling

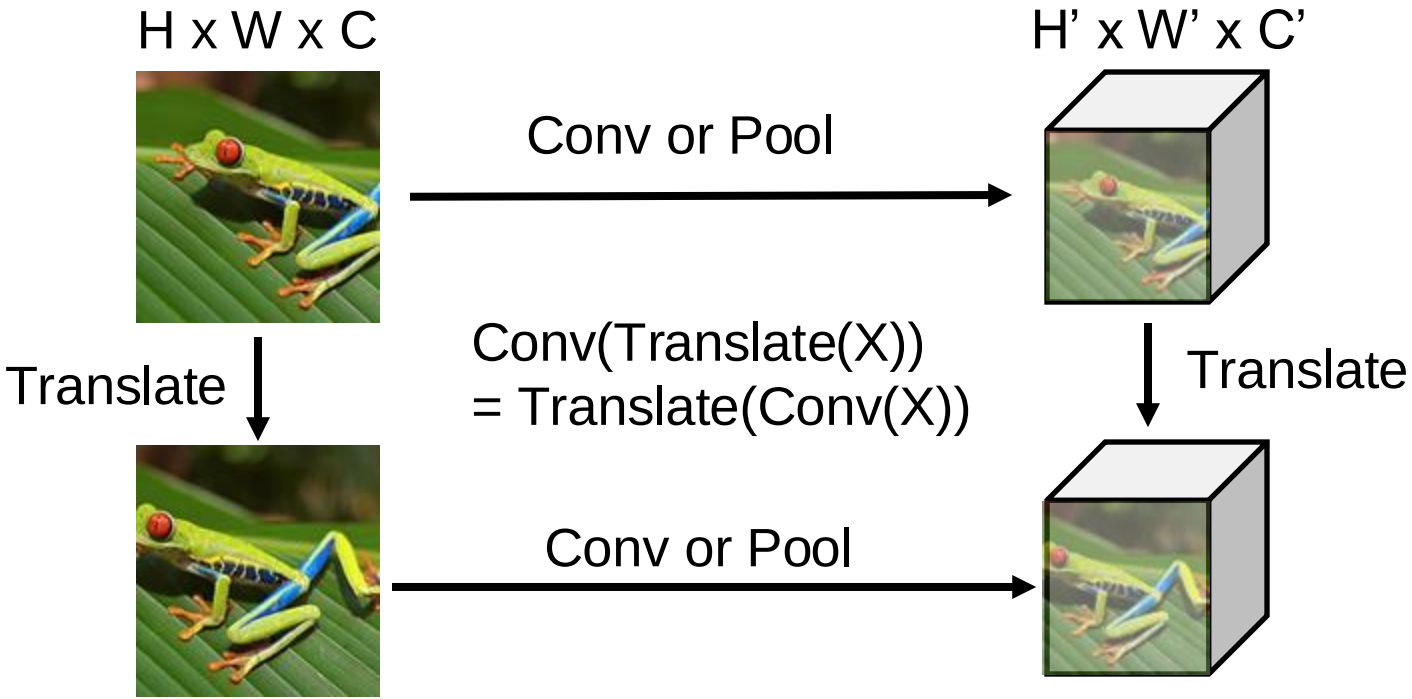
Convolution and Pooling: Translation Equivariance



Convolution and Pooling: Translation Equivariance



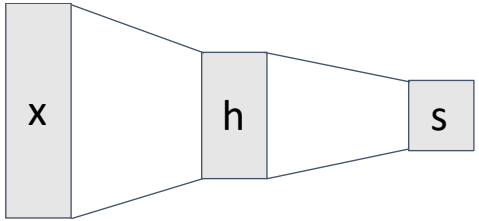
Convolution and Pooling: Translation Equivariance



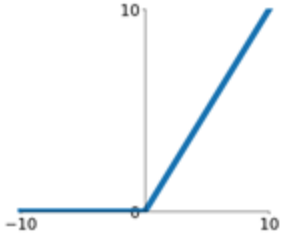
Intuition: Features of images don't depend on their location in the image

Summary: Convolutional Networks

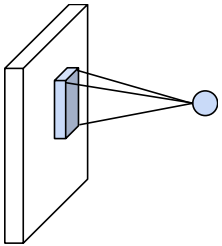
Fully-Connected Layer



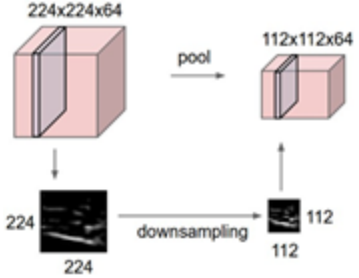
Activation Function



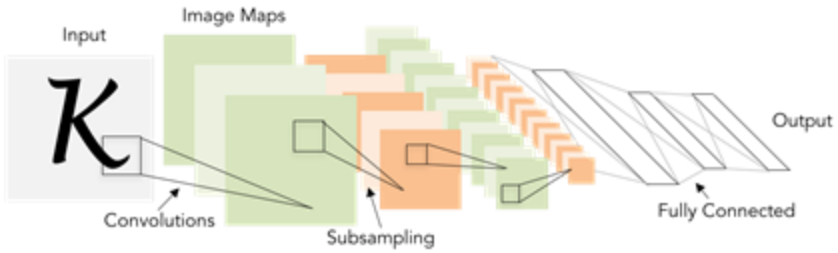
Convolution Layer

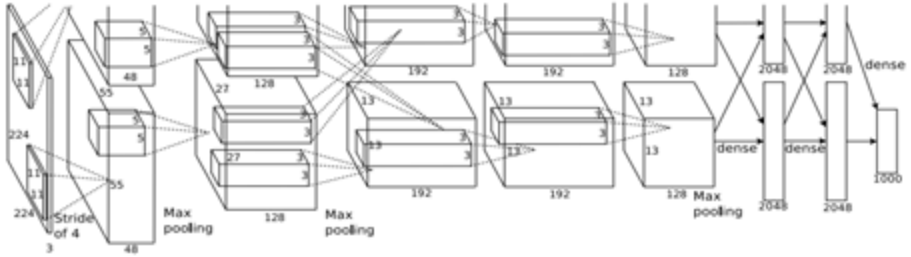


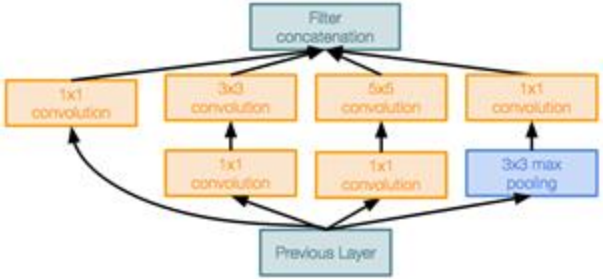
Pooling Layer

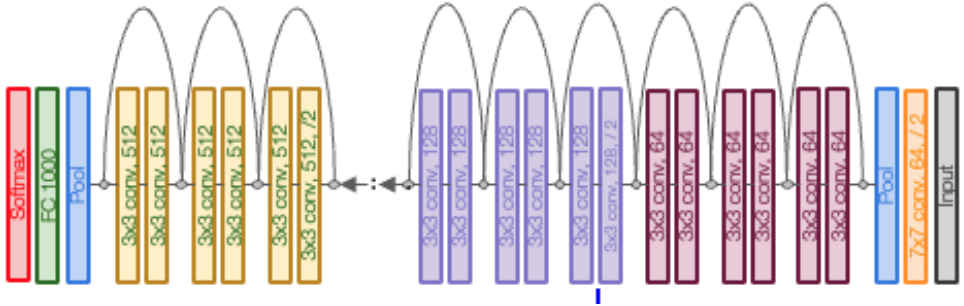


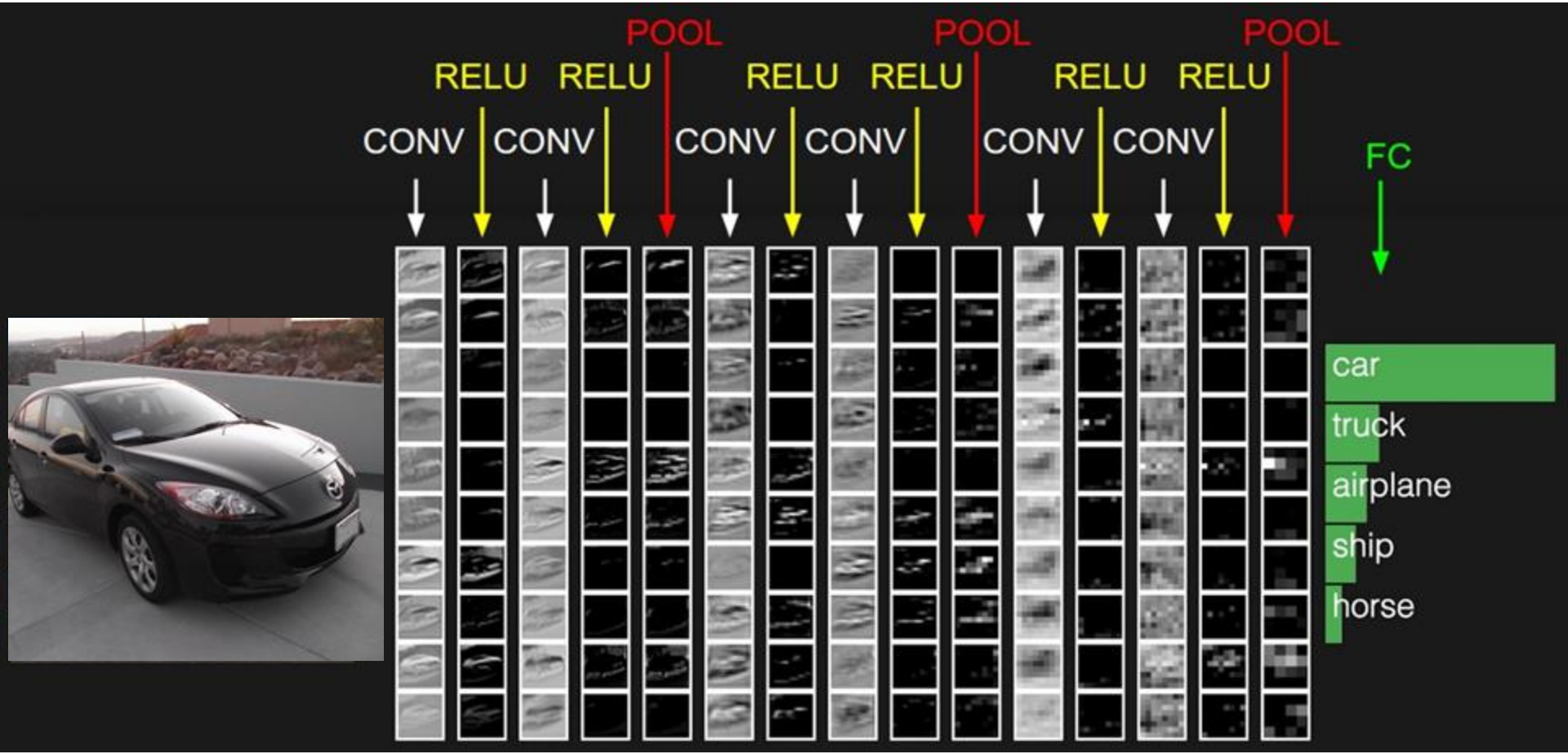
Next time: CNN Architectures










POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

[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