深度学习简介

周晓巍

计算摄影学第七节

Slides adapted from Fei-Fei Li, Justin Johnson, Serena Yeung, Noah Snavely, Jia-Bin Huang

Outline

- 机器学习
- 线性分类器
- 神经网络
- 卷积神经网络
- 训练神经网络
- 深度学习历史及进展

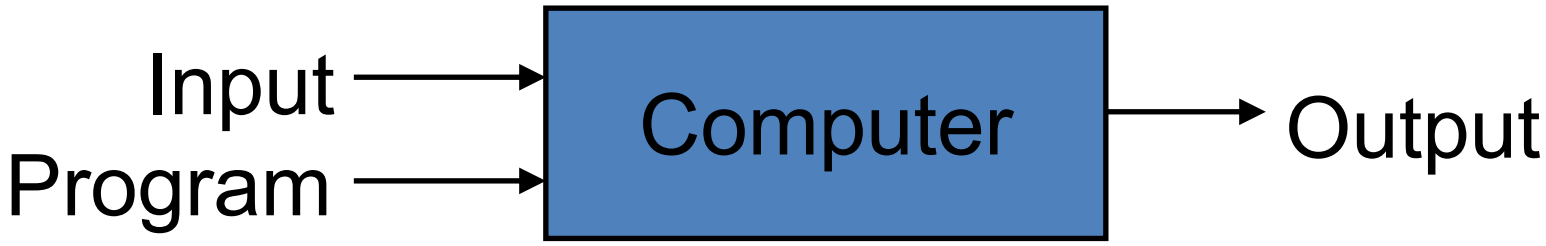
Part I

Machine Learning

机器学习

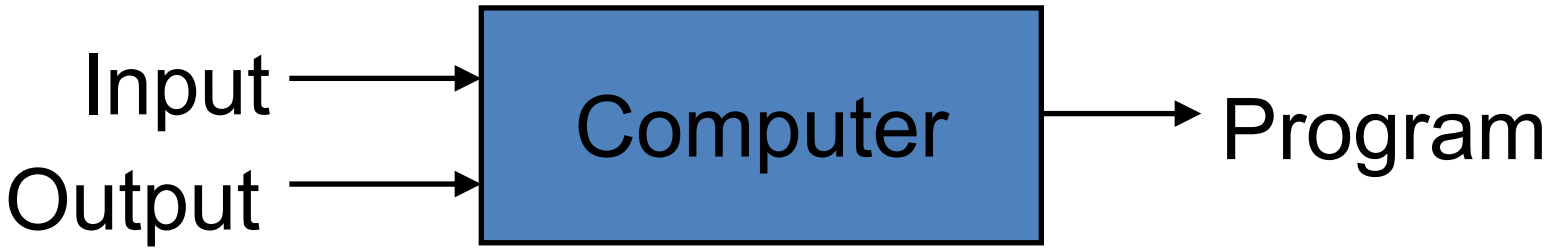
Machine Learning

Traditional Programming



- Examples: image segmentation

Machine Learning

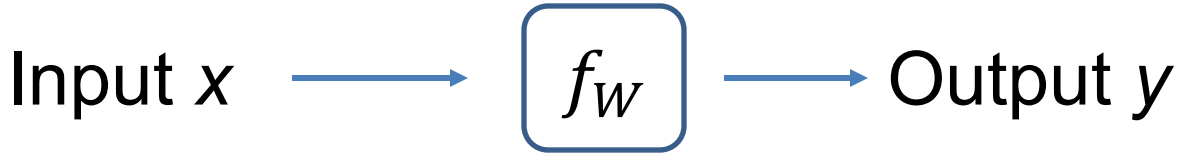


- Development of computer programs that can access data and use it learn for themselves

Slide adapted from Pedro Domingos

Important concepts

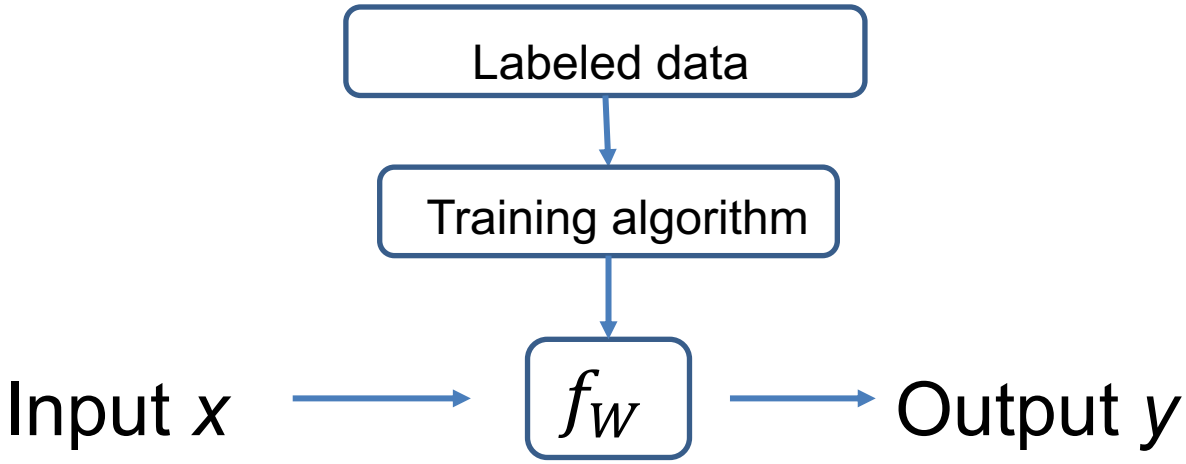
- **Model**: a mathematical description that explains the relation between input x and output y



  - often defined as a function $y = f_w(x)$ with parameters dentoed by w
  - Examples
    - x: parents' height; y: children's height; $y = f_w(x) = w^T x$
    - x: image pixel; y: foreground/background label; $y = f_w(x) = ?$
  - Problem types:
    - **Regression**: y is a real number
    - **Classification**: y is a discrete label

Important concepts

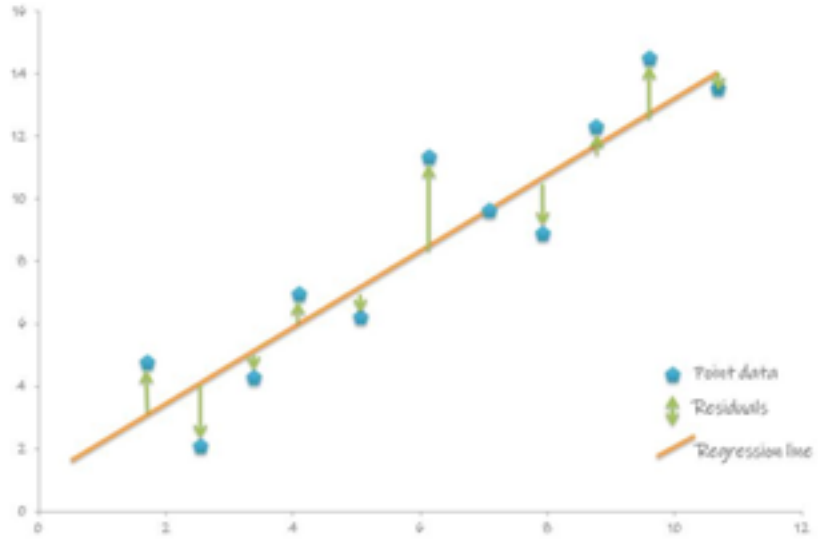
- **Supervised learning**: find f_W from labeled data
  - **labeled data**: exisitng (x,y) pairs, called **training data**, e.g., images already segmented



- Two phases:
  - **Training**: given labeled data, find f_W (**model fitting**)
  - **Testing**: given f_W and a new x, find y (also called **inference**)

A simple example

- Linear regression
  - What is the model?
  - Why is it called regression?
  - What is the training algorithm?



General pipeline for solving a problem with machine learning

- Define the problem
  - What is x and y
- Collect training data: (x,y) pairs
- Design a model
  - What is the form of f_W
- Model training
  - Define a loss function $l(w)$
  - Find optimal w that minimizes $l(w)$ using some optimzation algorithm
- Testing (application)
  - Given a new data x, predict y with the learned f_W

Image classification

- *x: image, y:* class label
- *f:* is called a classifier

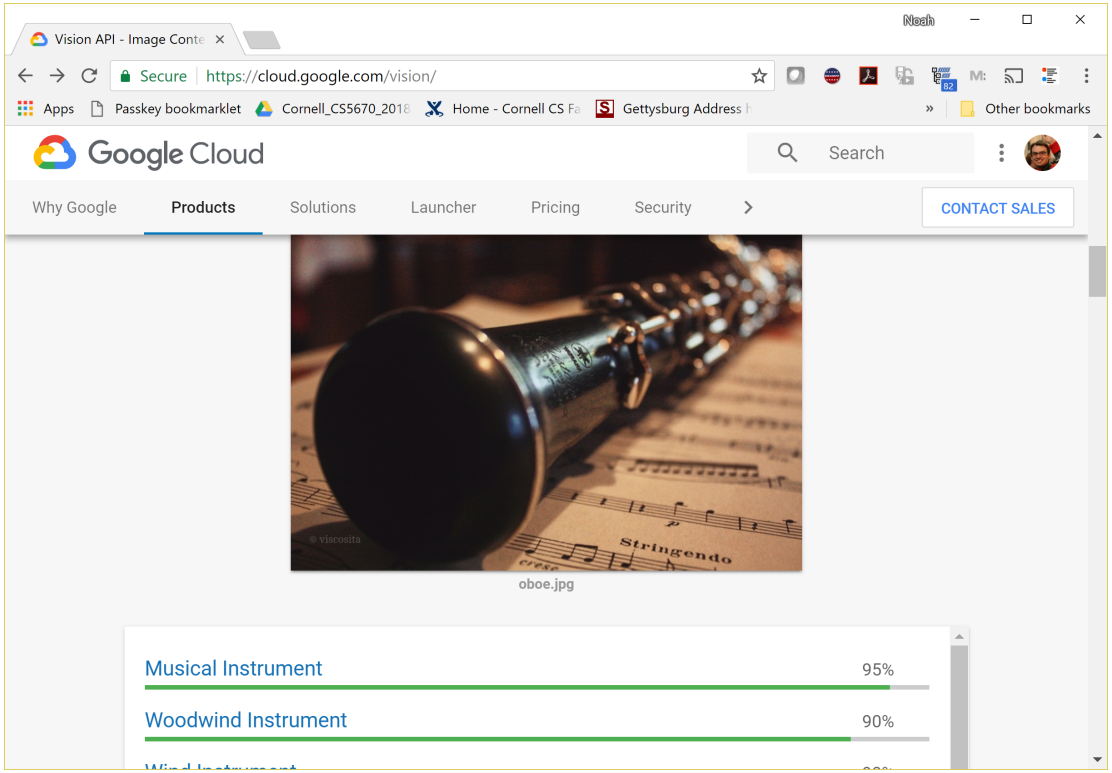
f() = “apple”

f() = “tomato”

f() = “cow”

Dataset: ETH-80, by B. Leibe Slide credit: L. Lazebnik

Image classification demo



https://cloud.google.com/vision/

See also:
https://aws.amazon.com/rekognition/
https://www.clarifai.com/
https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/
…

Image classification

- An easy task for human

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

cat

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Image classification

- But difficult for computer

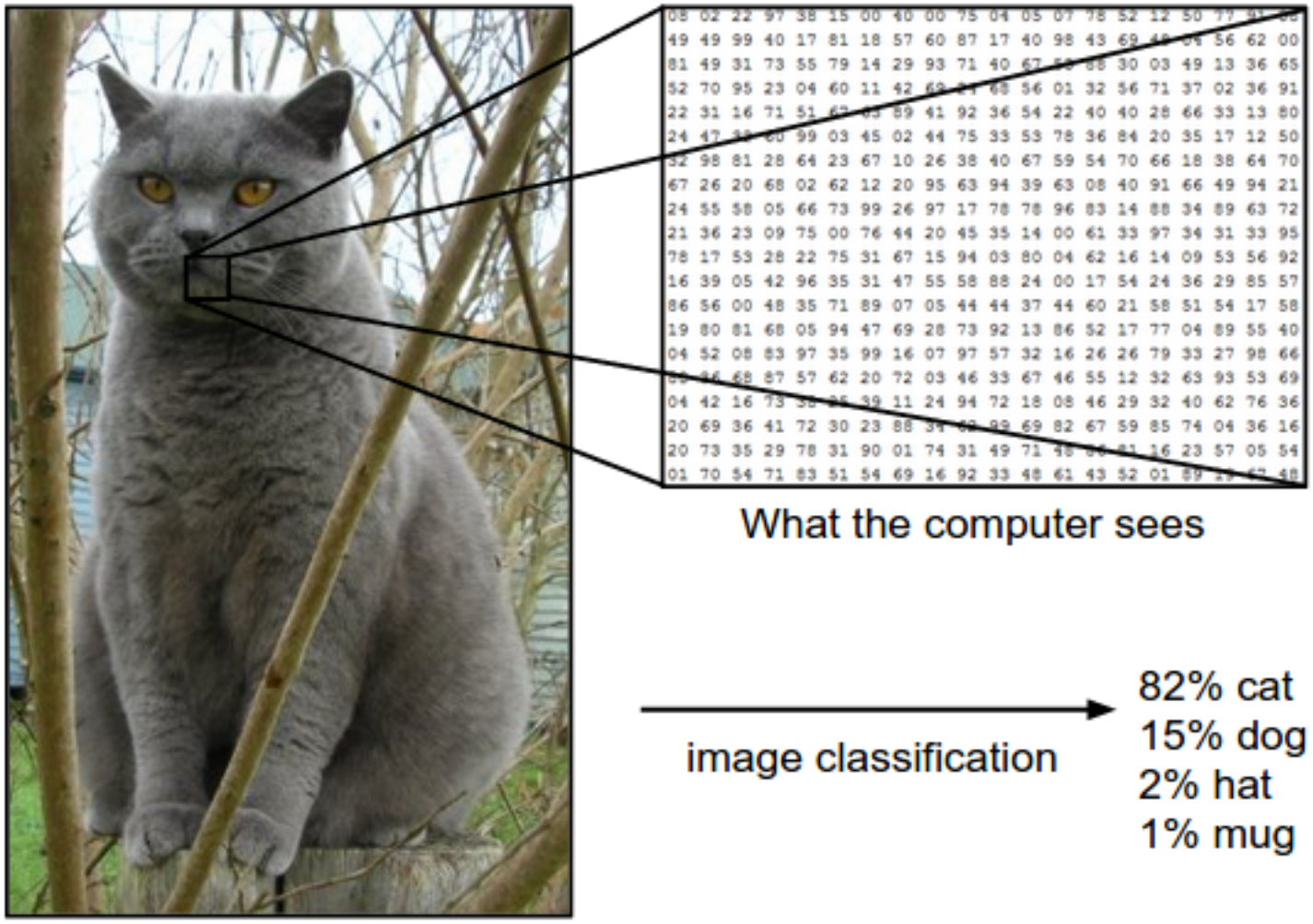
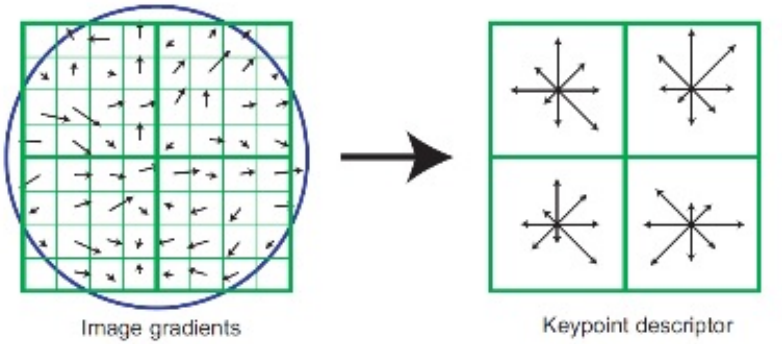
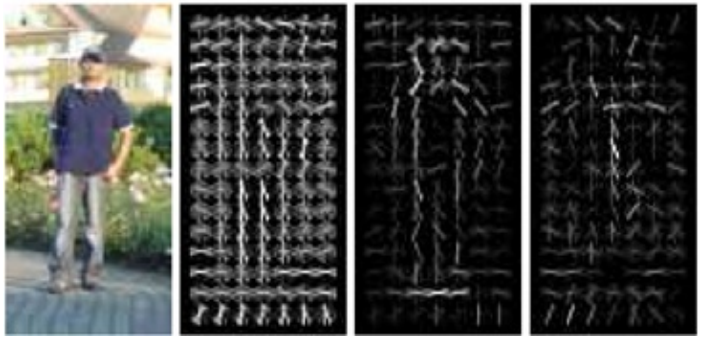


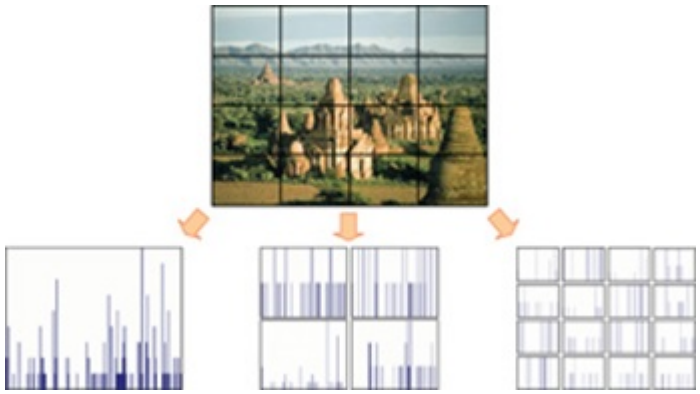
Image features



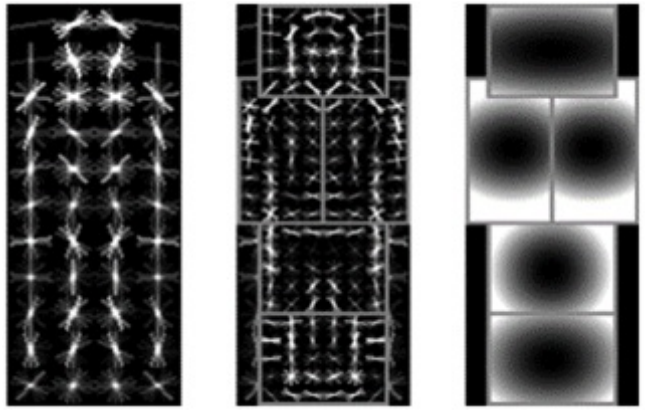
SIFT [Loewe IJCV 04]



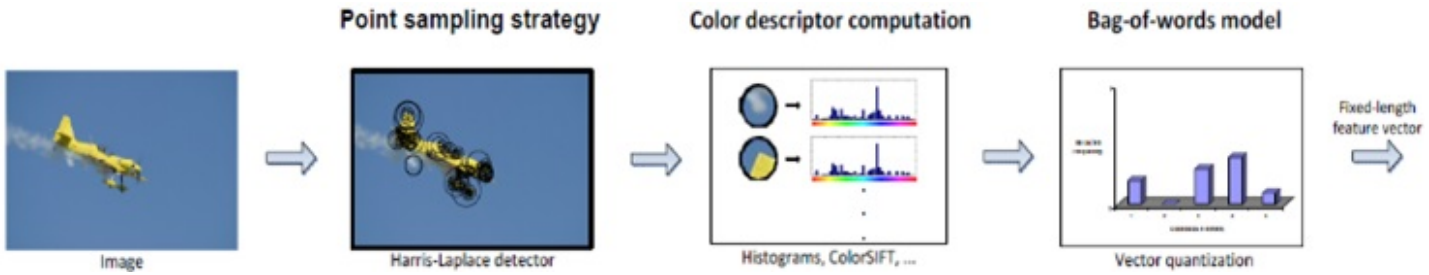
HOG [Dalal and Triggs CVPR 05]



SPM [Lazebnik et al. CVPR 06]



DPM [Felzenszwalb et al. PAMI 10]



Color Descriptor [Van De Sande et al. PAMI 10]

Why use features?
Why not pixels?

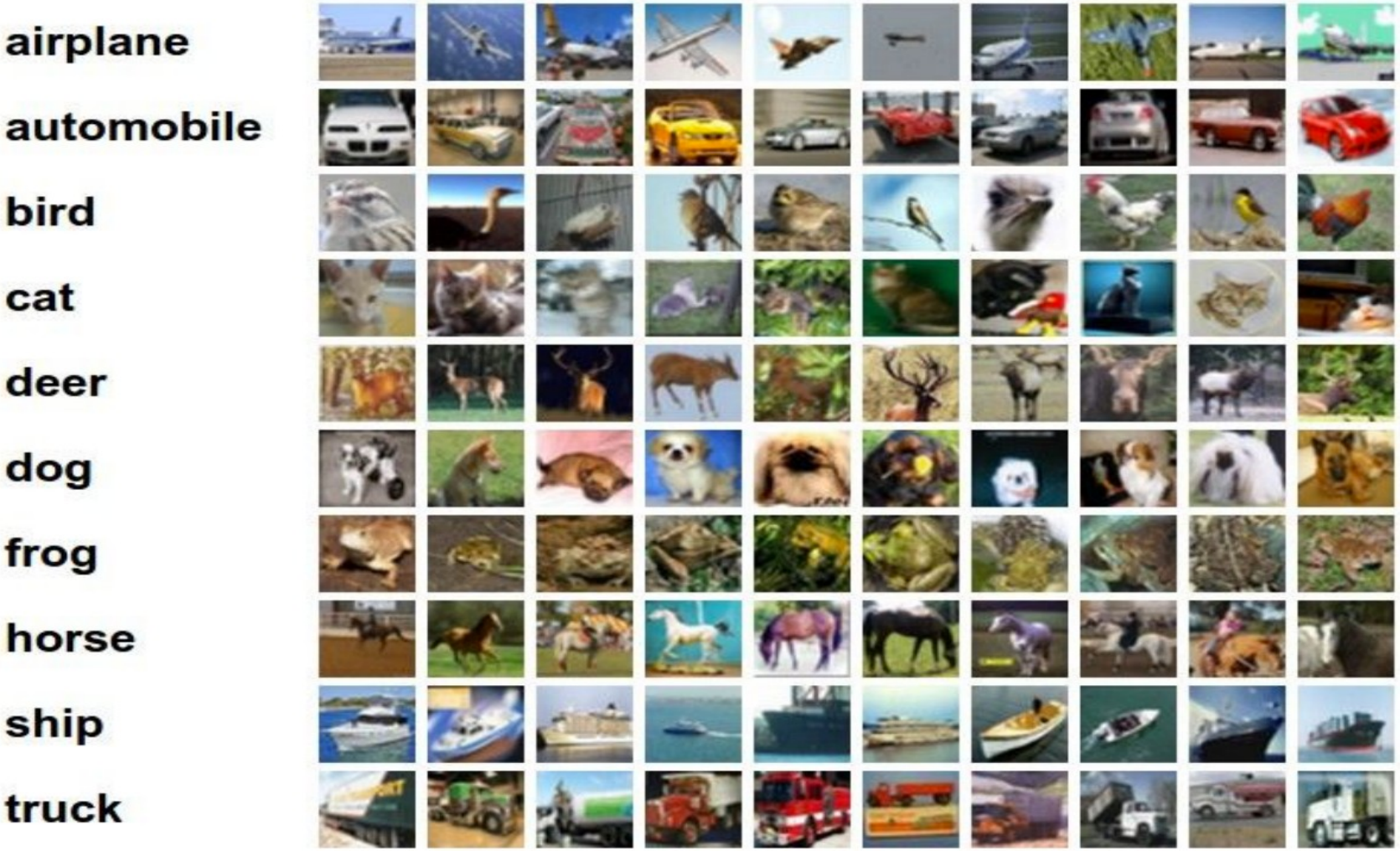


Image classifier

```
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.

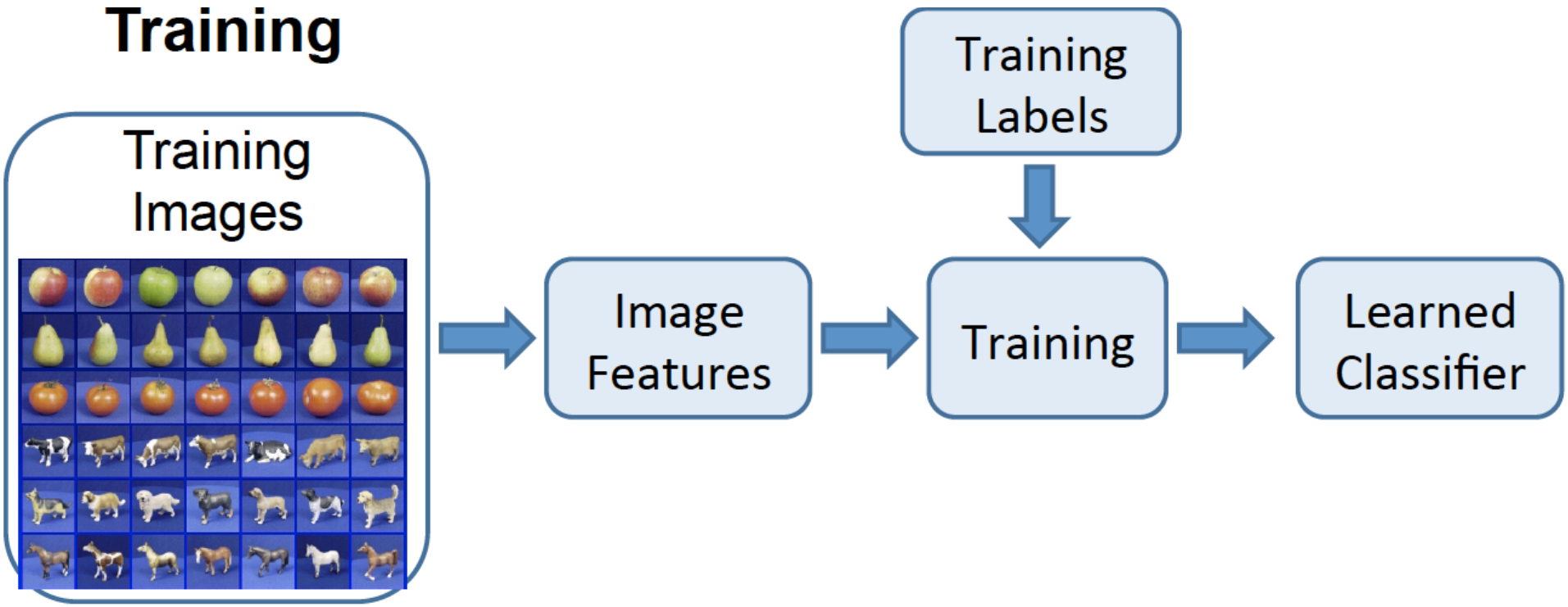
Slide credit: Fei-Fei Li & Justin Johnson & Serena Y

Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier

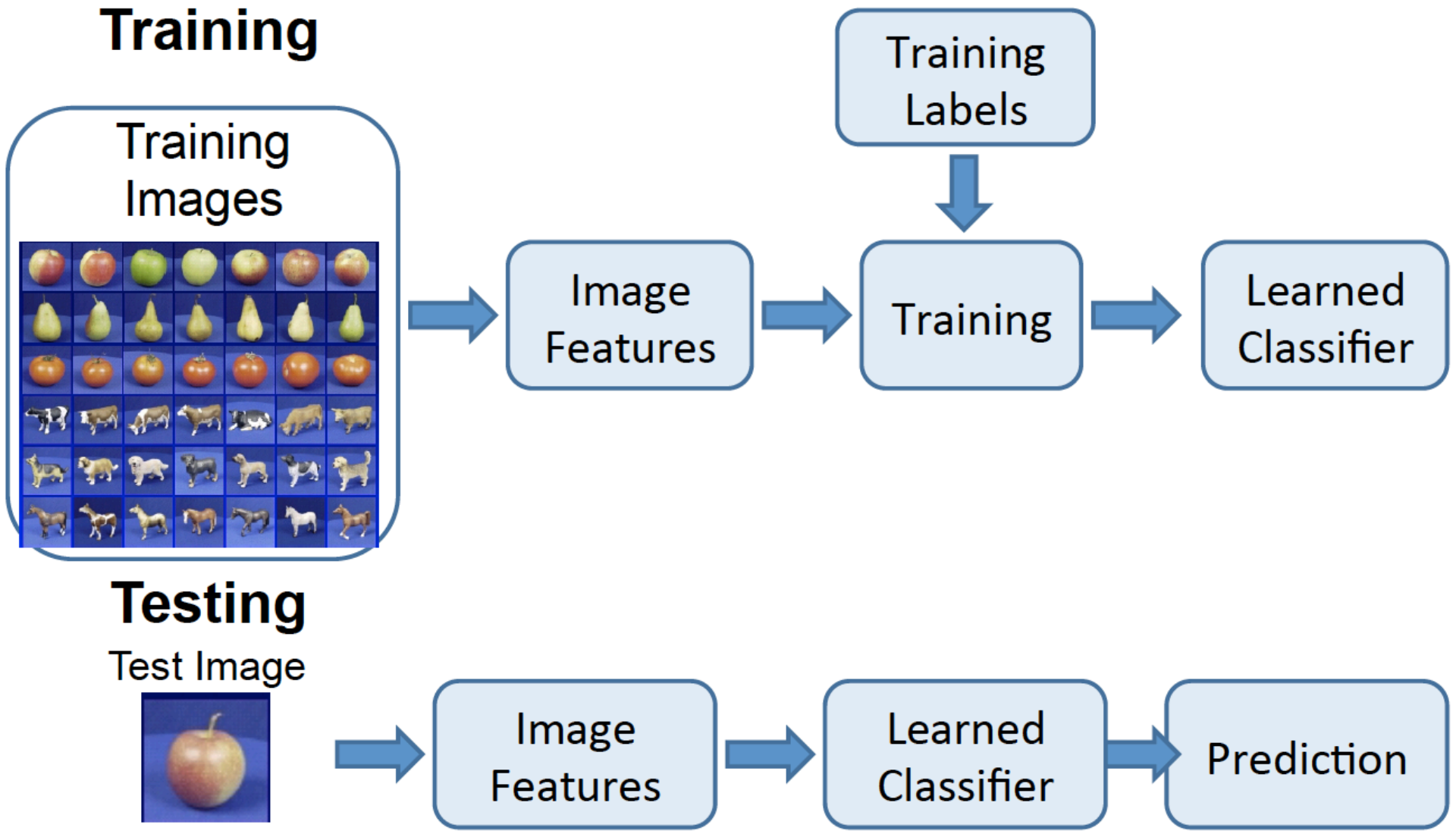
Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Image classification pipeline



Dataset: ETH-80, by B. Leibe Slide credit: D. Hoiem, L. Lazebnik

Image classification pipeline



Dataset: ETH-80, by B. Leibe Slide credit: D. Hoiem, L. Lazebnik

Summary

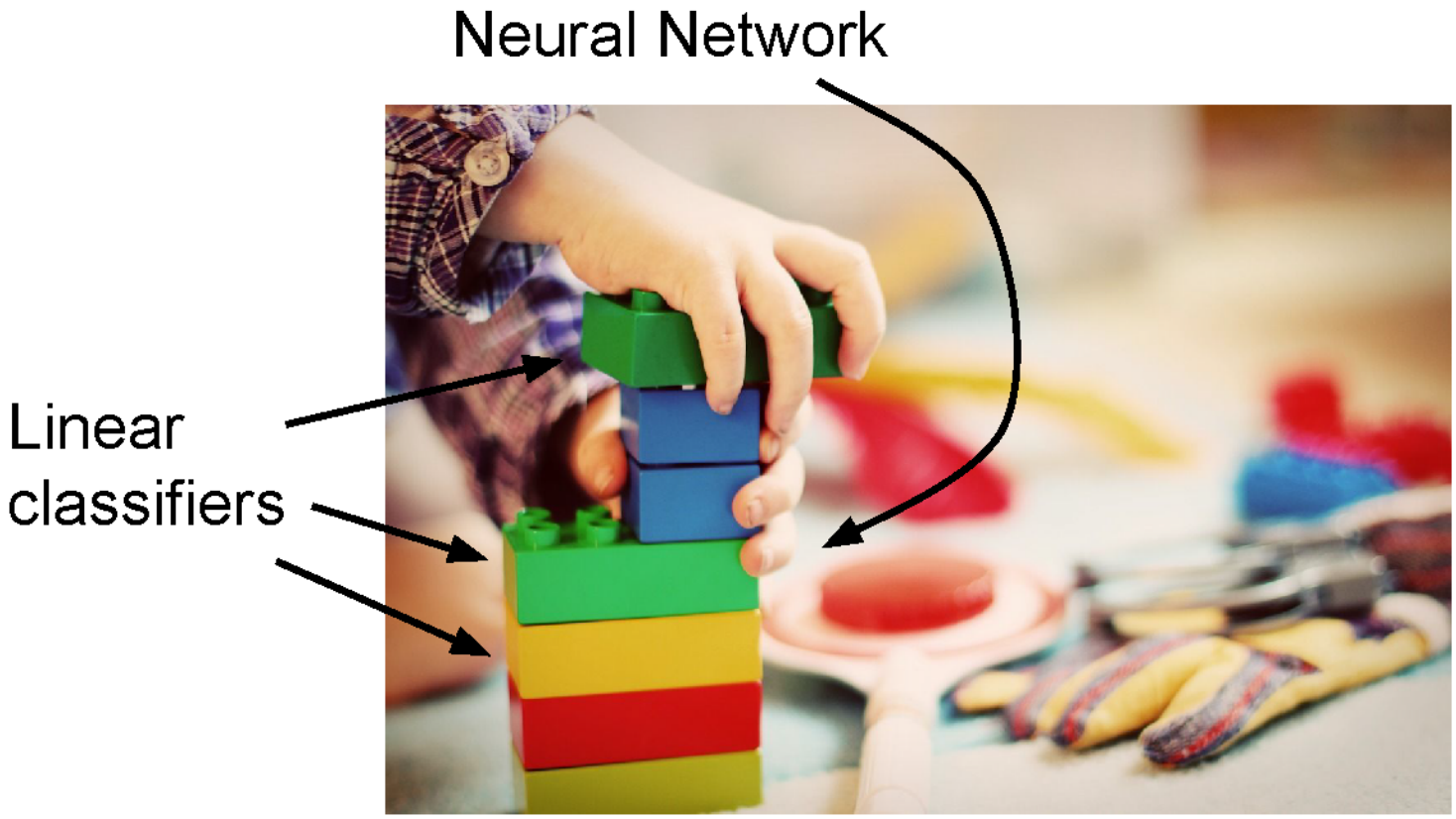
- What is learning
- What is supervised learning
  - Regression
  - Classification
- Image classification
  - Image feature
  - Classifier

Part II

Linear Classifier

线性分类器

Why linear classifiers important?



Classification as a function

- Input x: image
- Output y: class scores

class scores

Classification as a function

- What form should the function have?

image parameters

f(**x**,**W**)

10 numbers, indicating class scores

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Linear classifier

bias vector

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

3072x1

10x1

10x1

10x3072

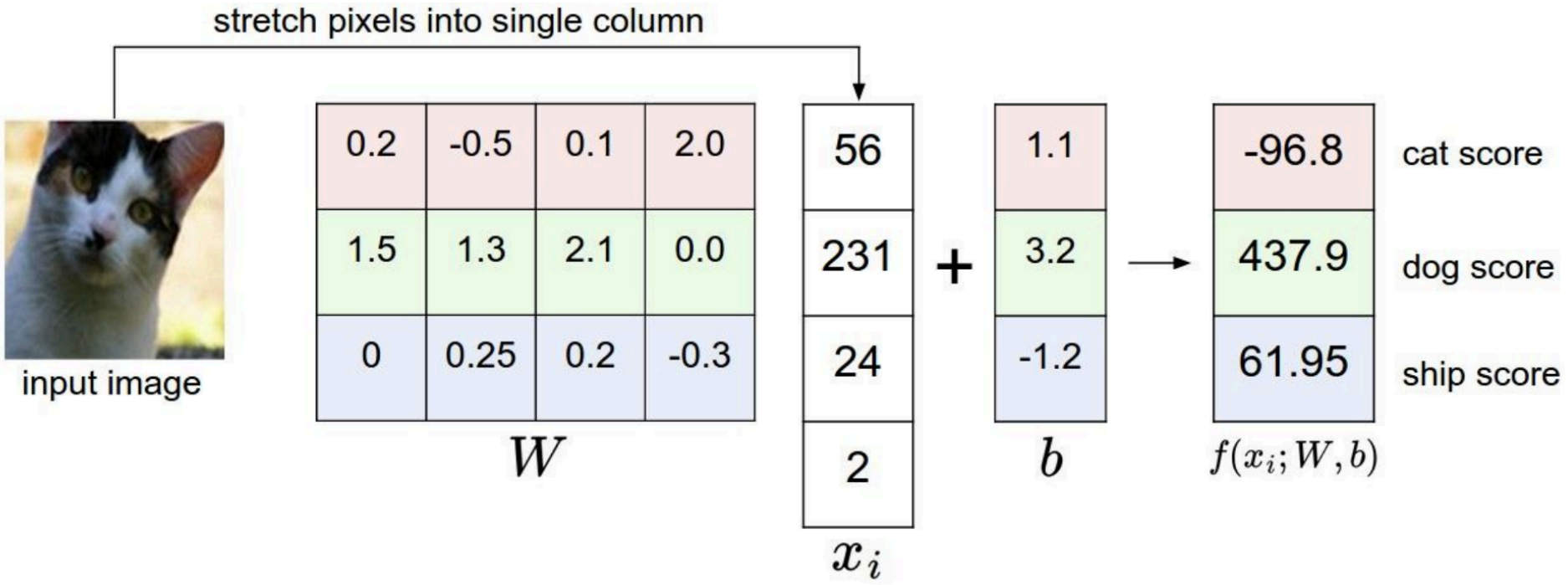
10 numbers, indicating class scores

[32x32x3]
array of numbers 0...1

parameters, or “weights”

Linear classifier

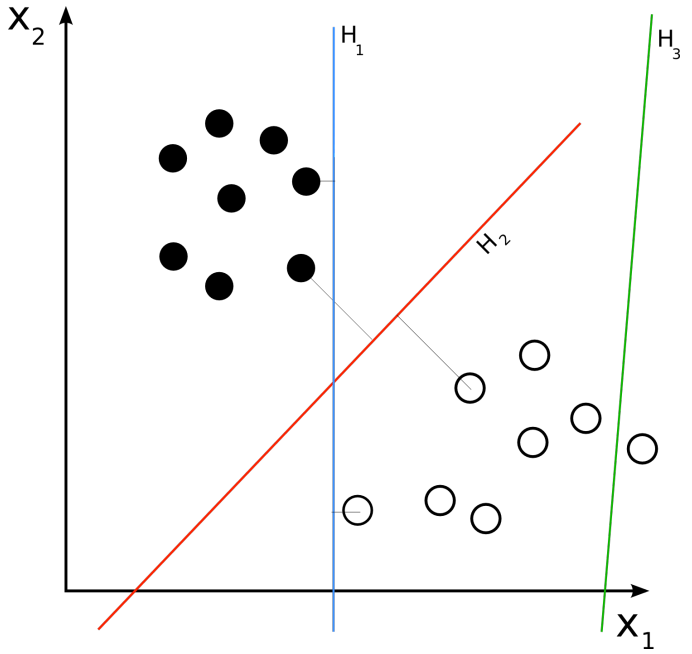
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Visualize linear classifier

- Binary classification
  - Find label by thresholding the score: e.g., it is a "cat" if $w^T x + b > 0$

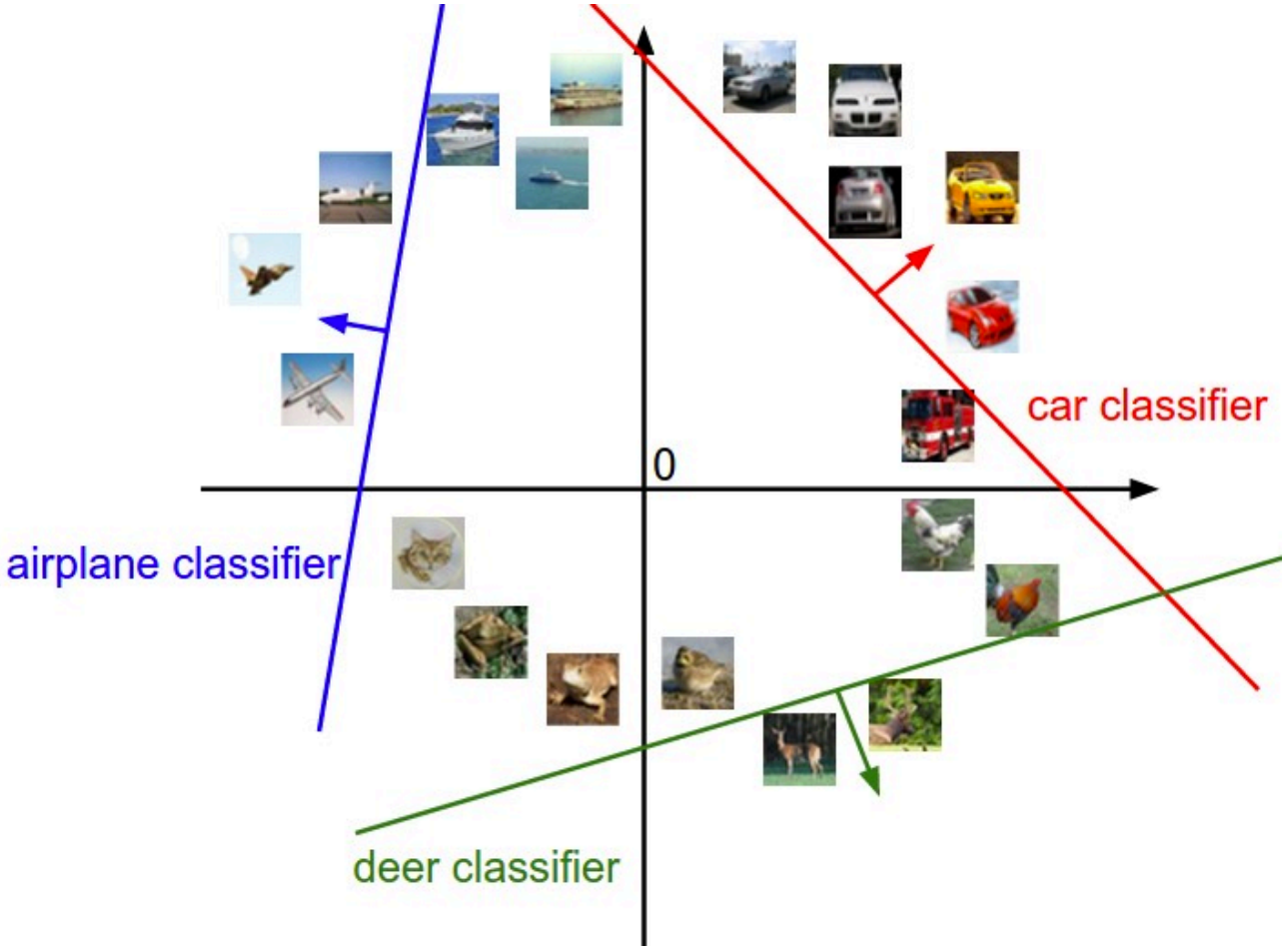
- In 2D, $w^T x + b = 0$ defines a line



Each line H_i correspondes to a set of parameter (w, b)

Visualize linear classifier

- Multi-class



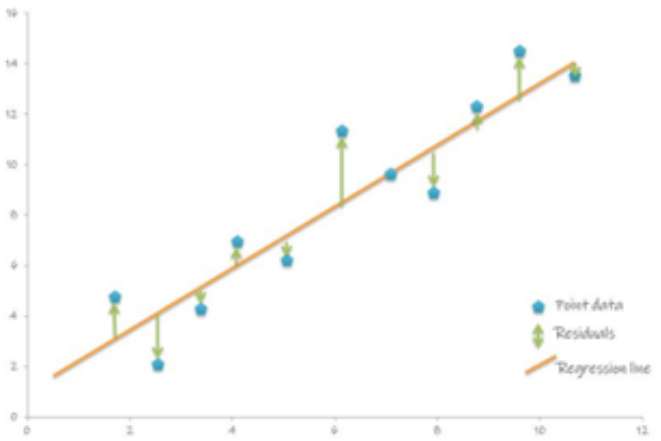
Training: how to find good parameters based on training data?

- Define a loss (objective) function that quantifies the “goodness” of the model parameters
- Solve the optimization problem

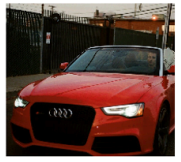
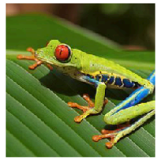
Loss function of classification

- Recap: loss function of regression:

$$\min_{w} \sum_{i} (y_i - f_w(x_i))^2$$



- Can we use the MSE loss for classiciation?
  - No
  - Labels are discrete and scores are real numbers

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

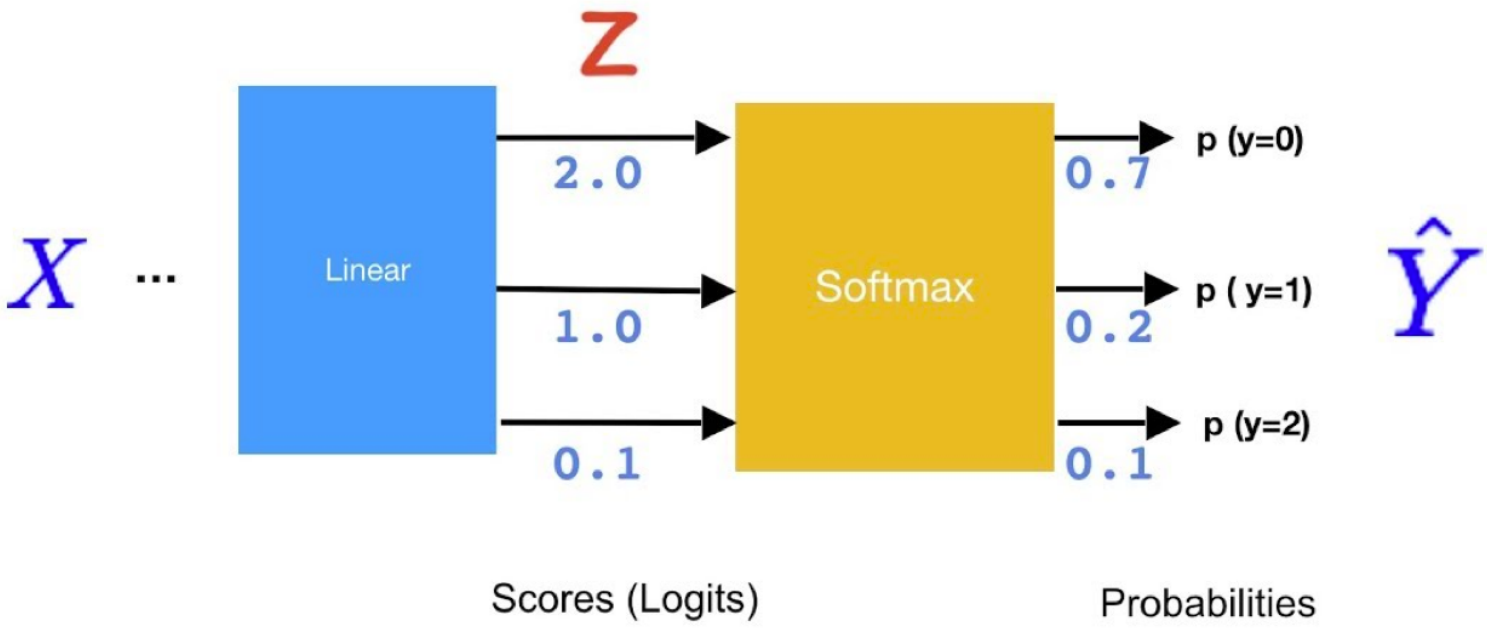
How to define a loss function for predicted scores?

Class labels can be viewed as probabilities

1. Convert scores to probabilities
2. Compute cross entropy between predicted and true probabilities

Covert scores to probabilities

Softmax function: $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.



Why softmax?

- Approximate a max operator
- Statistical interpretability

Image credit: Sung Kim

Cross entropy as loss function

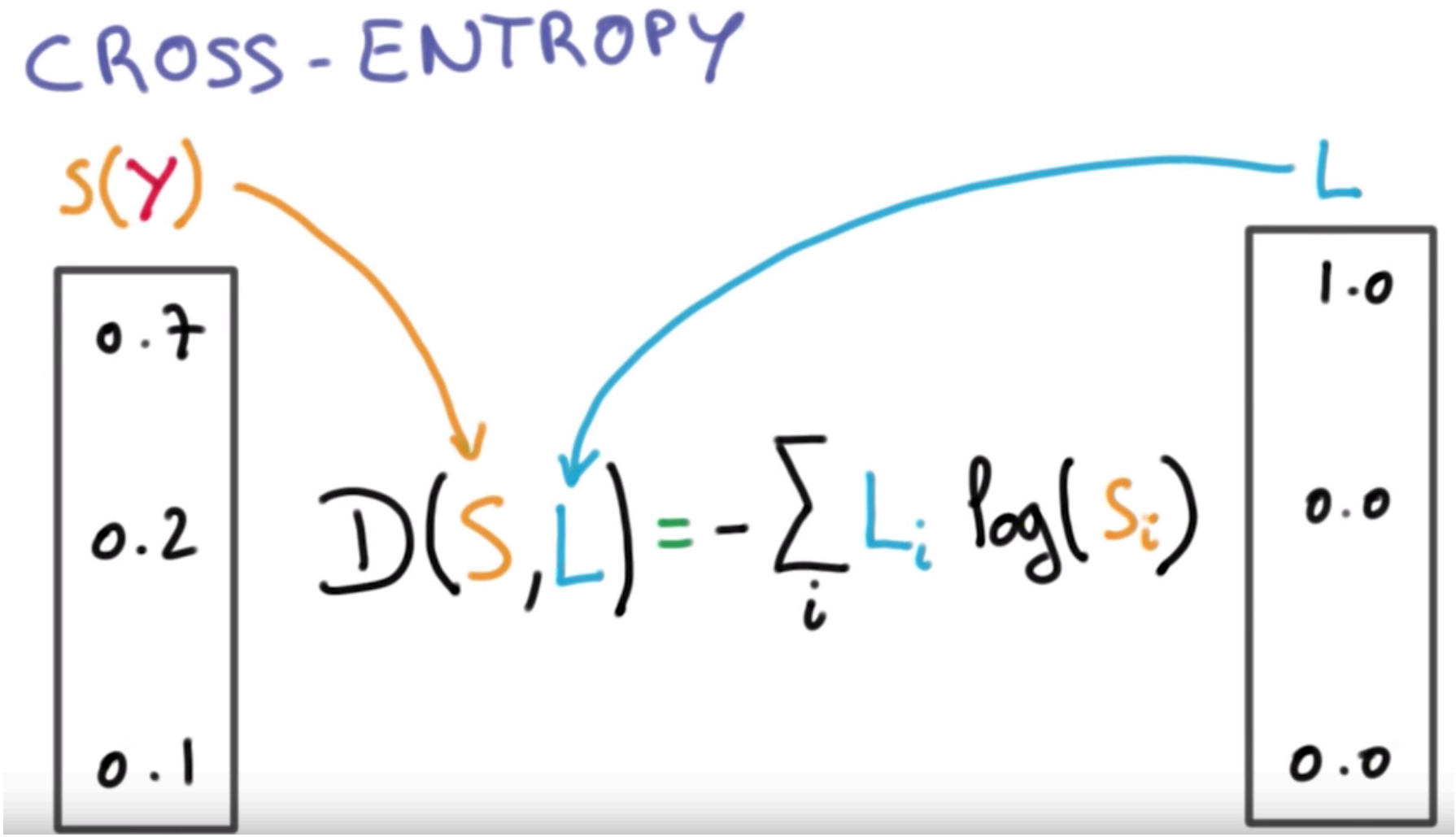


Image credit: Ritchie Ng

Summary

- Linear classifier
- Softmax function
- Cross-entory loss

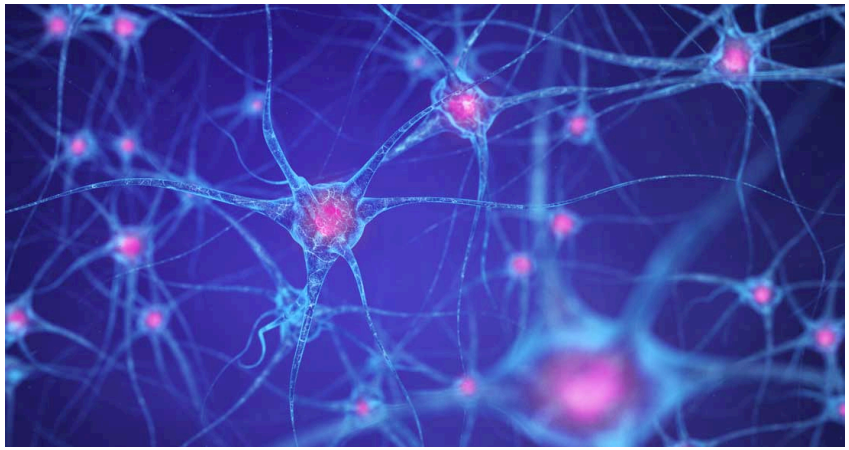
Part III

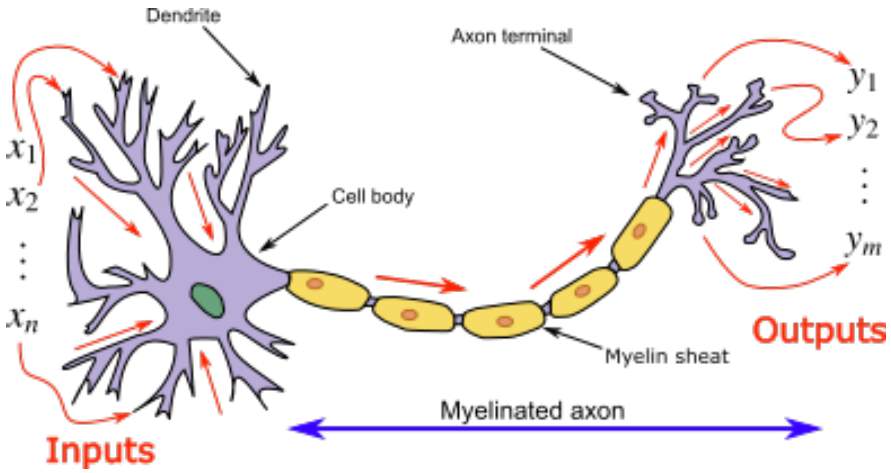
Neural Networks

神经网络

Information processing in human brain

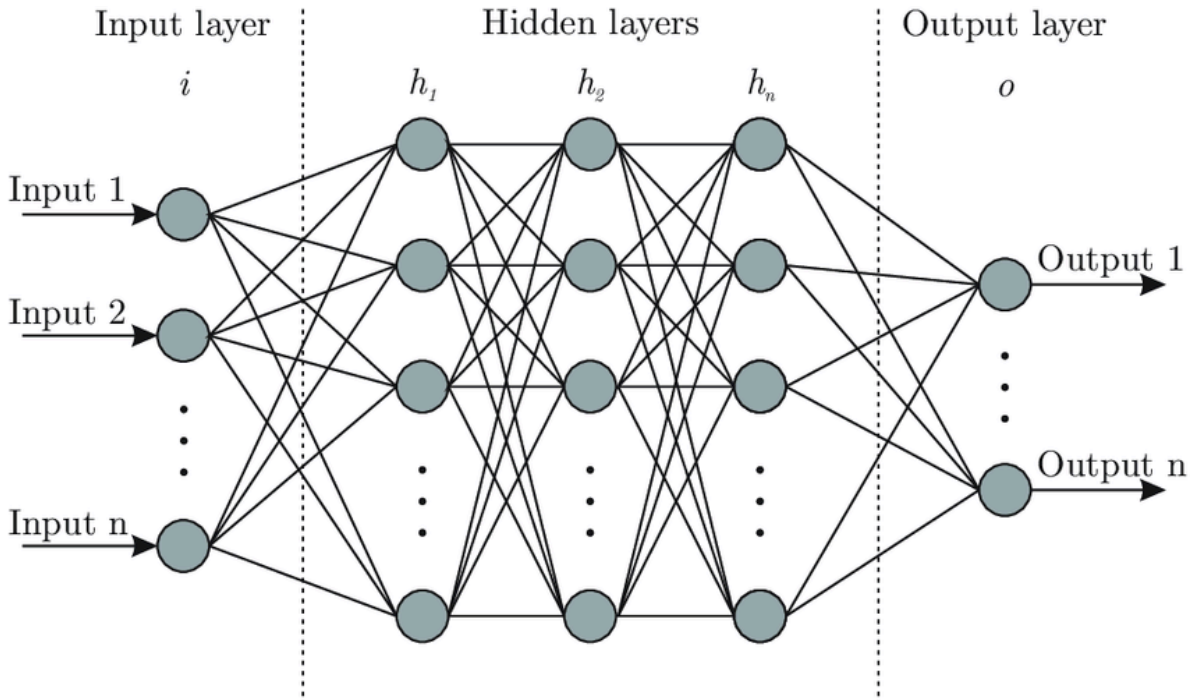
- Very mysterious
- Much complex than linear classifier
- Some facts:
  - Nervous system is composed of many many neurons.
  - A neuron is a nerve cell that carries electrical impulses.



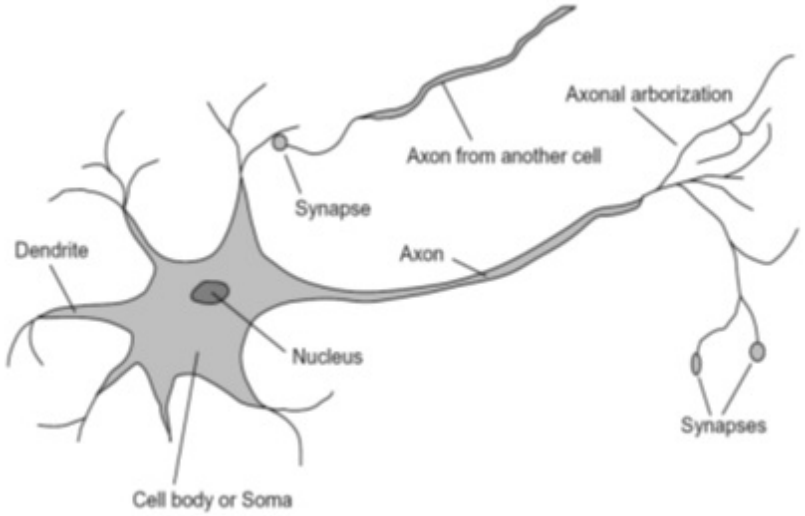


Neural networks

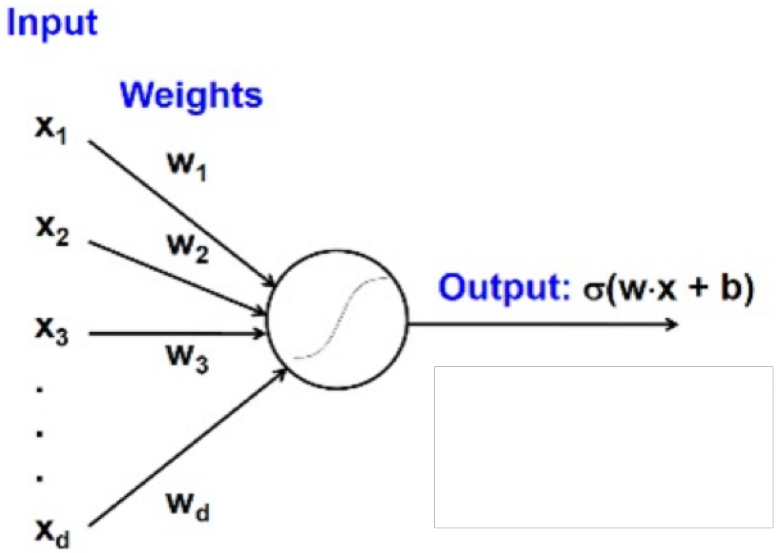
- (Artifical) neural networks – a computational model inspired by human nervous system
- Used to regress the function that maps input to output
- Basic unit – neuron
- Structure – input layer, hidden layers, output layer



Artificial neuron



A biological neuron

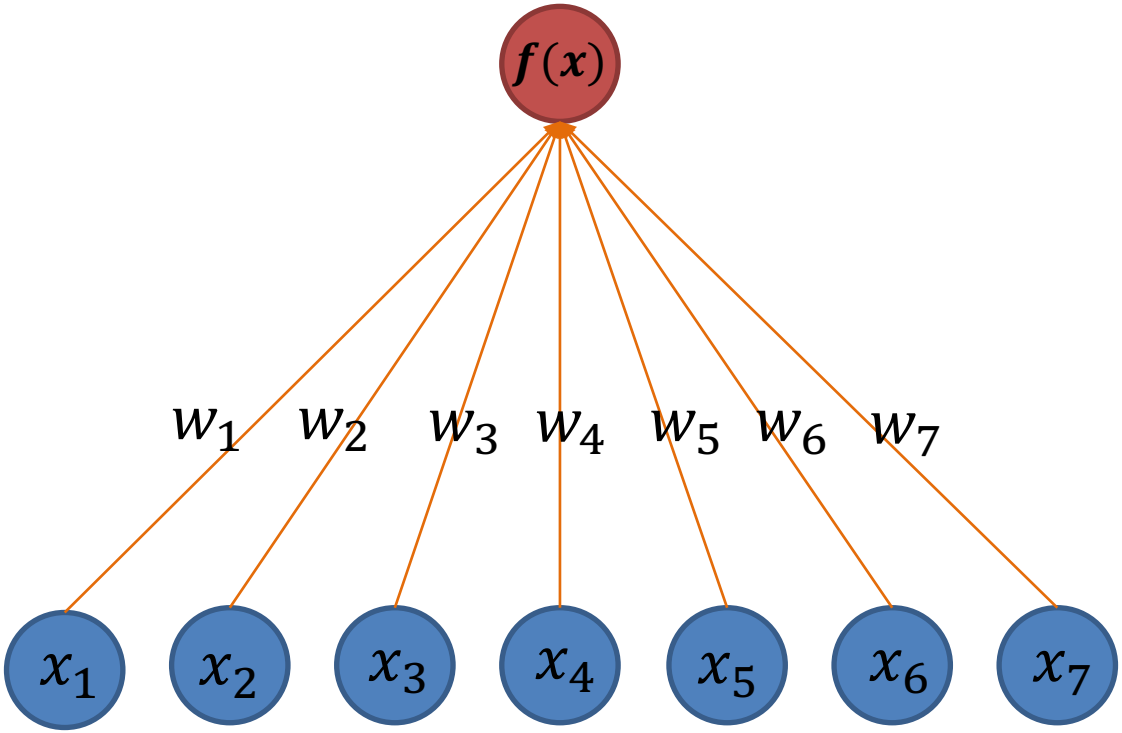


An artificial neuron

Artificial neuron

- Linear model (perceptron) as neuron:

$$f(x) = w^T x + b$$



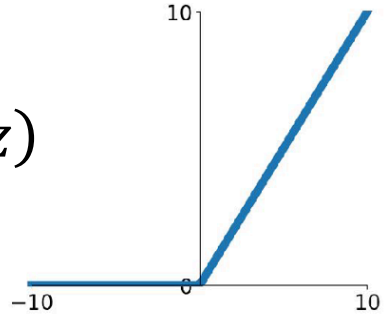
How to model nonlinearity?

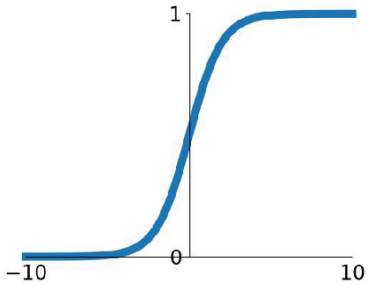
- Adding a nonlinear transform

$$f(x) = \sigma(w^T x + b)$$

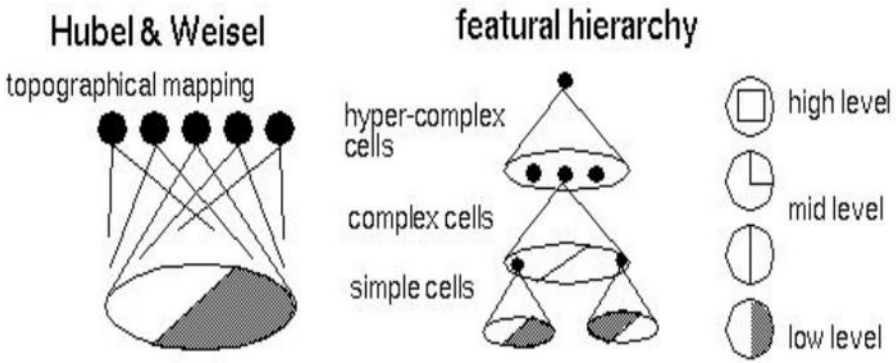
- Named activation functions （激活函数）

- Examples:
  - Rectified Linear Unit (ReLU): $\sigma(z) = \max(0, z)$
    - “Electrical pulse” cannot be negative
  - Sigmoid: $\sigma(z) = 1/(1 + e^{-z})$
    - Map values to probability

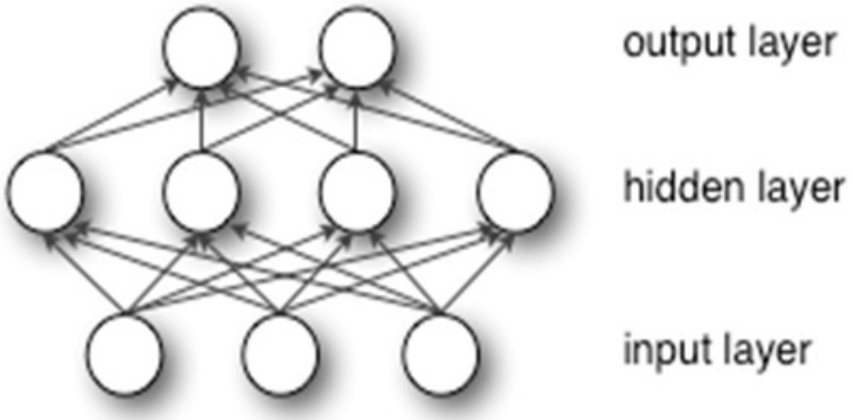




Neural Network: multiple layers of neurons



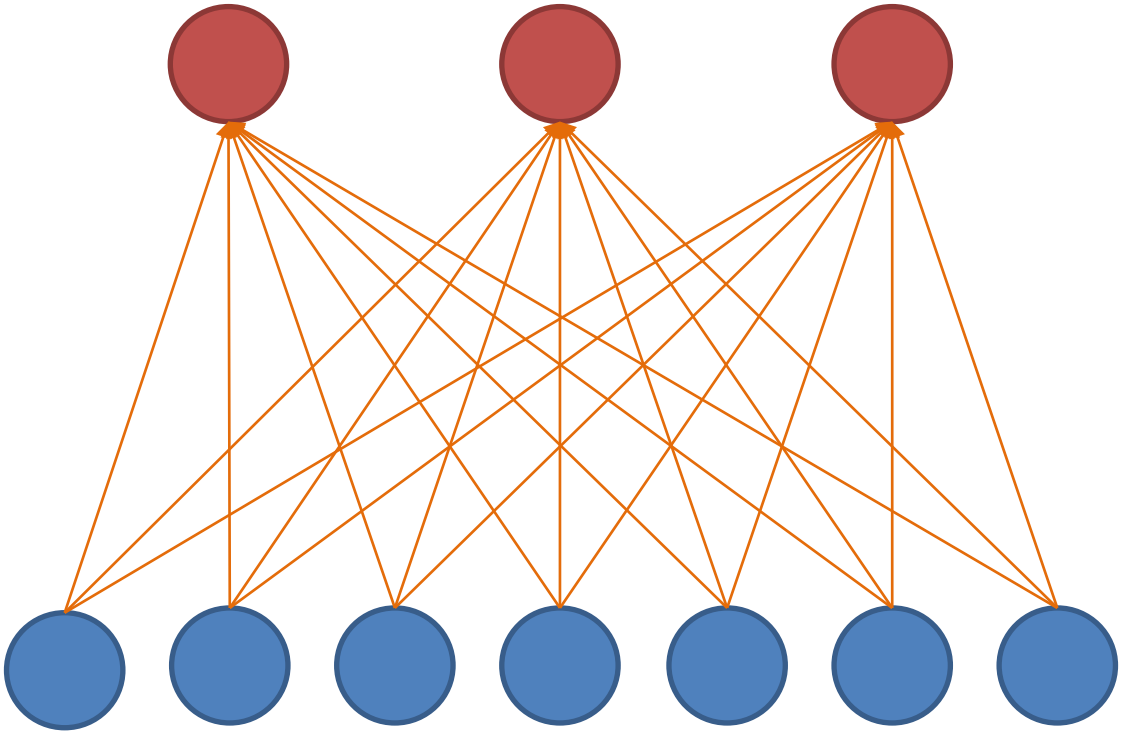
Hubel and Weisel's architecture



Multi-layer Neural Network

Neural networks

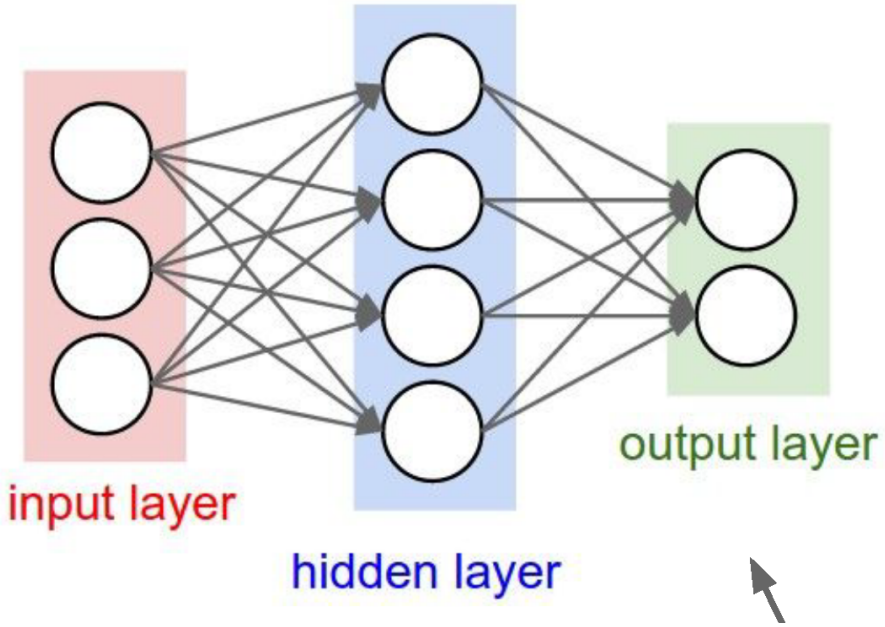
- Extend to multiple outputs



$$f(x) = \sigma(w^T x + b) \longrightarrow f(x) = \sigma(Wx + b)$$

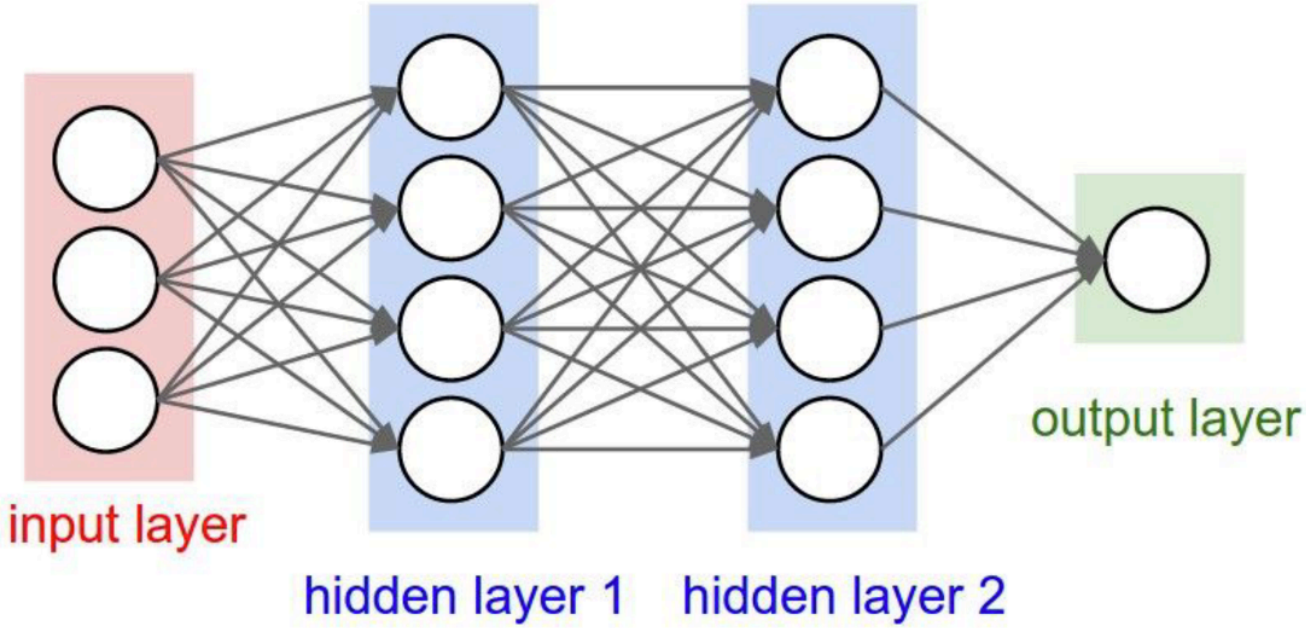
Extend to multiple layers

- $f(x) = \sigma(W_2\sigma(W_1x + b_1) + \mathrm{b}_2)$



Extend to multiple layers

- $f(x) = \sigma(W_3\sigma(W_2\sigma(W_1x + b_1) + \mathrm{b}_2) + \mathrm{b}_3)$



- Neural networks: linear functions chained together and separated by non-linear activation functions
- What will happen if there is no activation functions?

Deep neural networks

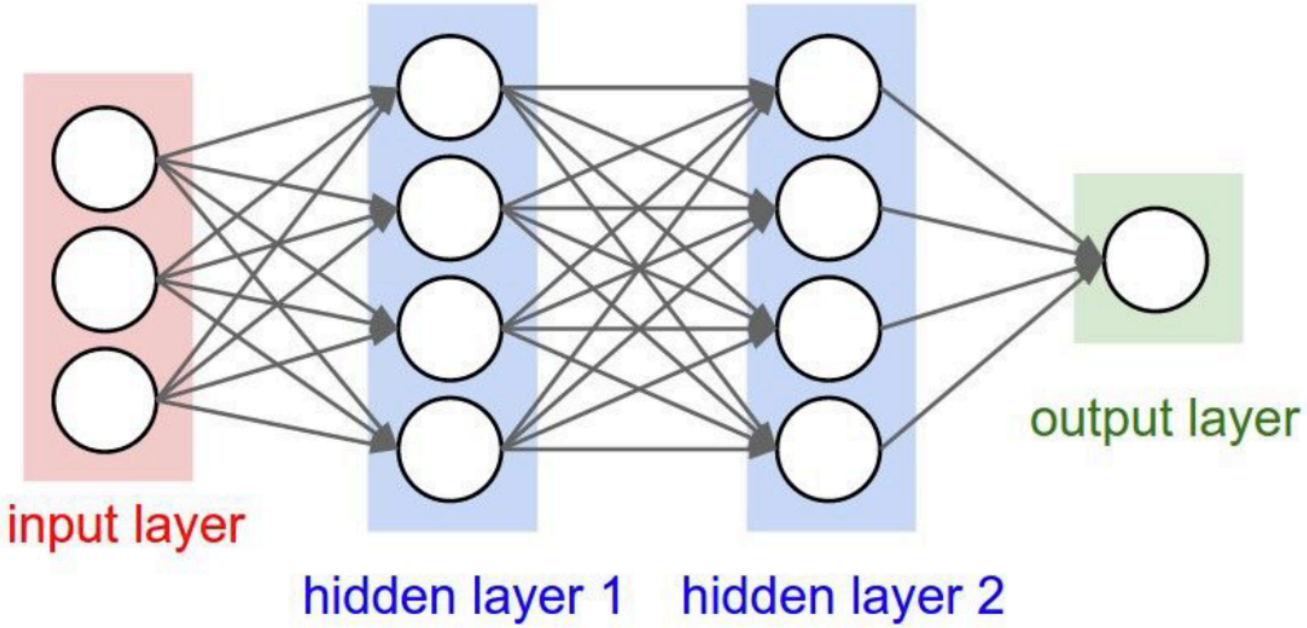
- Deep neural networks
  – neural networks with many layers

- A network with more layers mean is able to represent more complex function

- The more the better?
  - More parameters to learn
    - need more data
  - More difficult to train
    - “gradient vanishing”

Fully connected layer

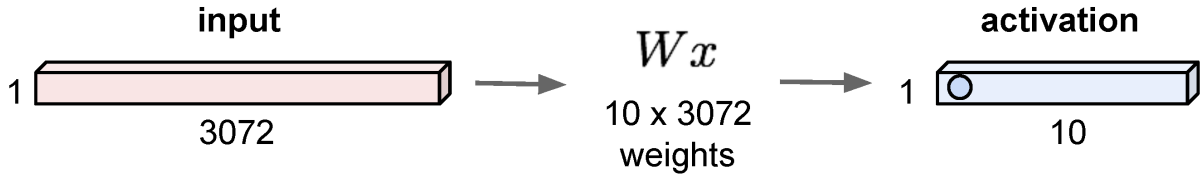
- Nodes are fully connected to the nodes in the previous layer



Fully connected layer

- Fully connected layer has a large number of weights
- Example:

32x32x3 image -> stretch to 3072 x 1



- Deep networks typically have many layers and potentially millions of parameters

- What problem will happen if there are too many parameters?

- How to reduce number of parameters?

Summary

- Perceptron
- Activation function
- Neural networks – multi-layer perceptron
- Mathmatical expression of neural networks
- Fully connected layer

Part IV

Convolutional Neural Networks

卷积神经网络

Convolutional neural networks

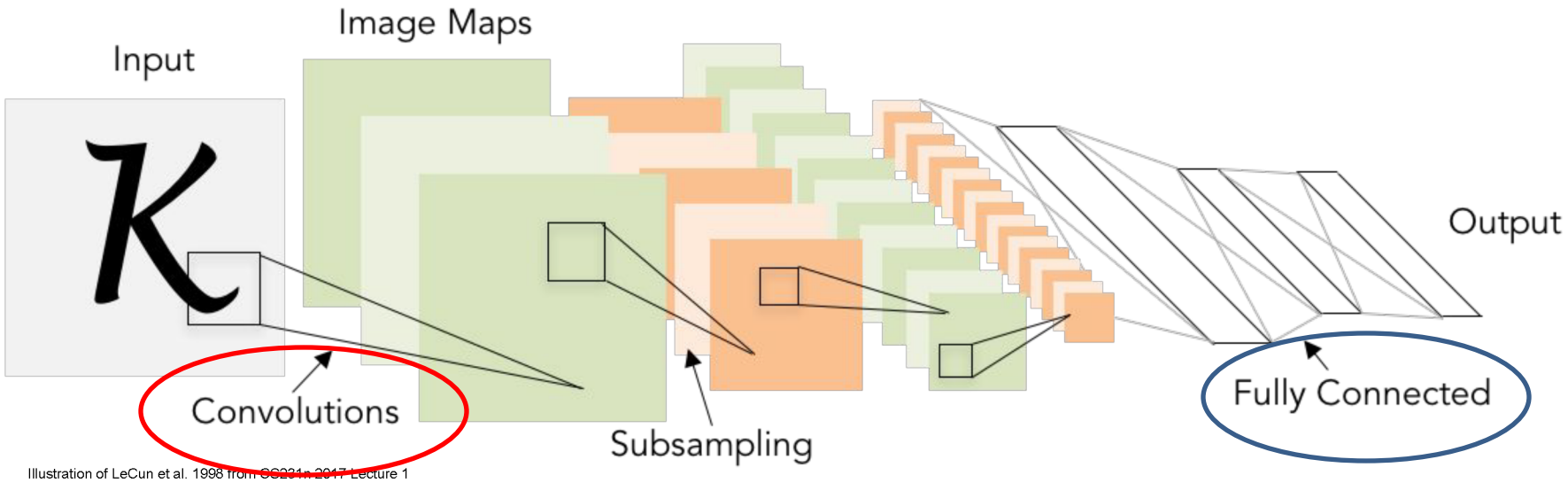
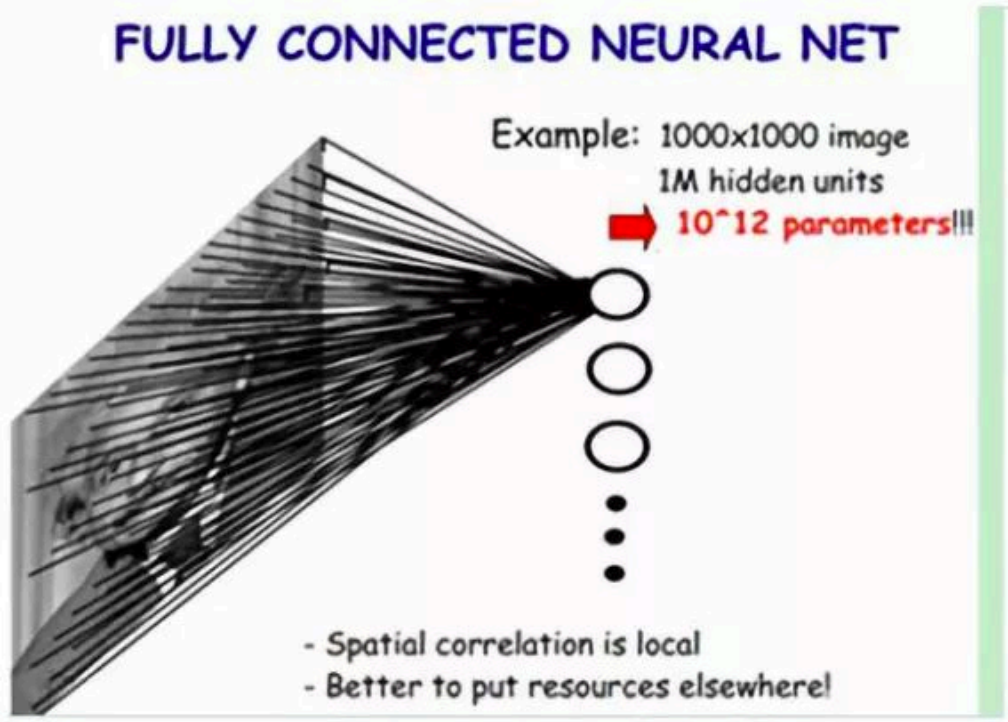
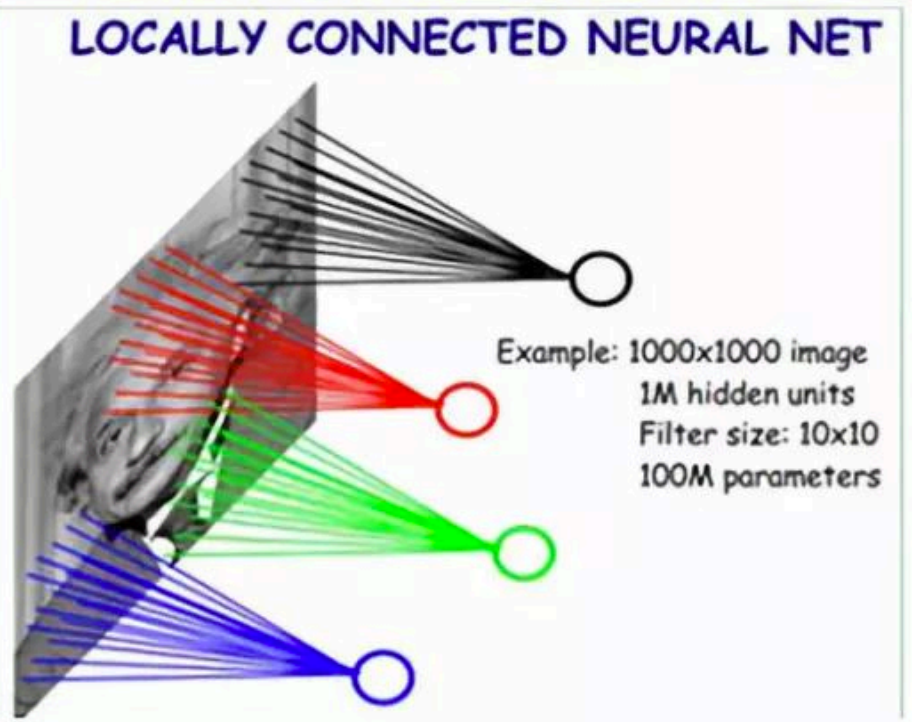


Illustration of LeCun et al. 1998 from ~~CS231n 2017~~ Lecture 1

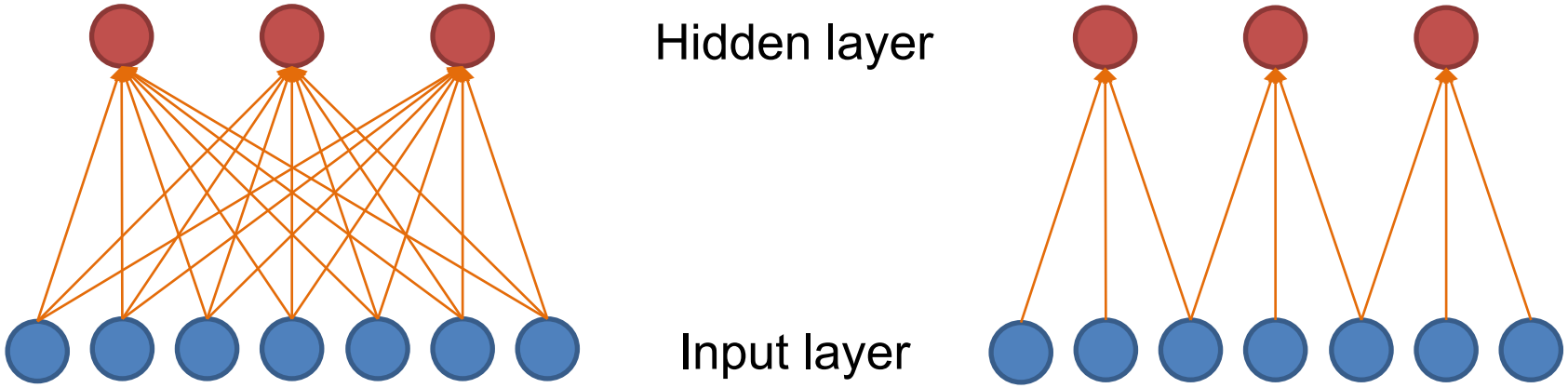
Local patterns are important

Locally connected network





Locally connected network

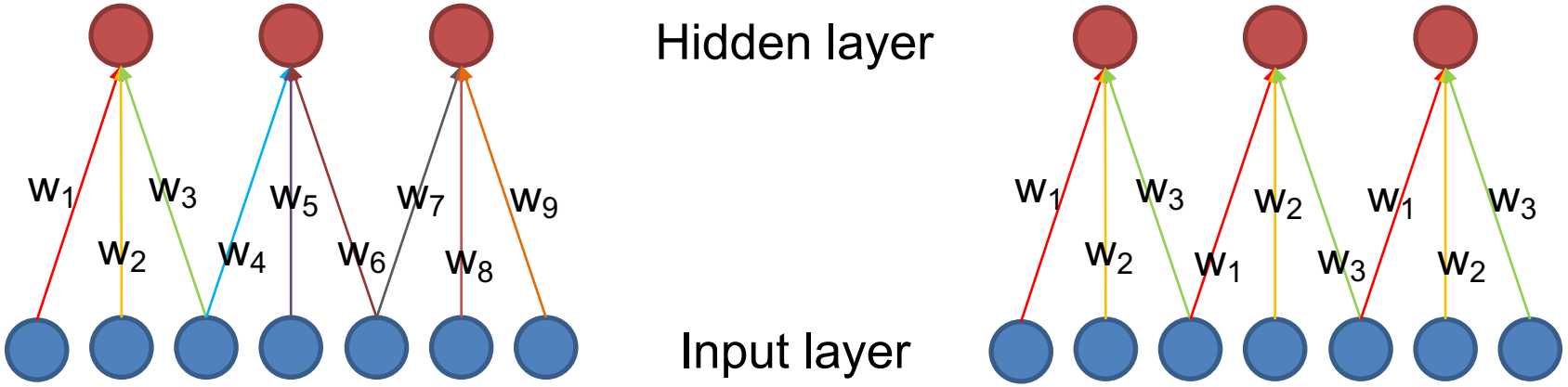


Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Global connectivity: 3 x 7 = 21
  - Local connectivity: 3 x 3 = 9

Weight Sharing



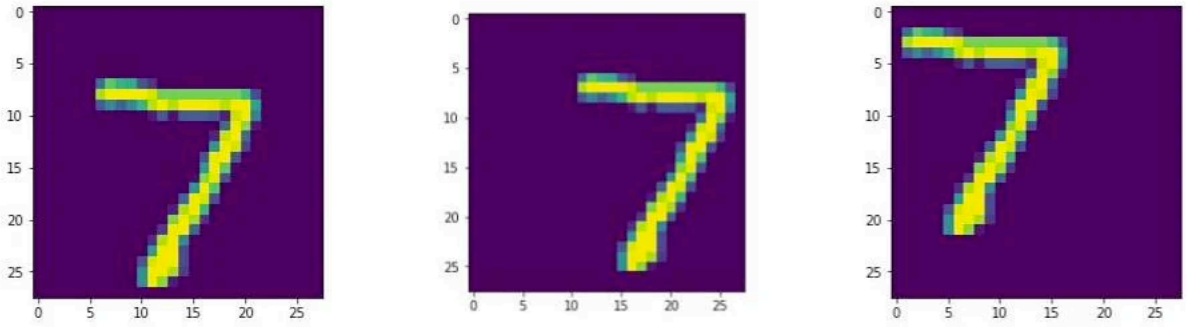
Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing: 3 x 3 = 9
  - With weight sharing : 3 x 1 = 3

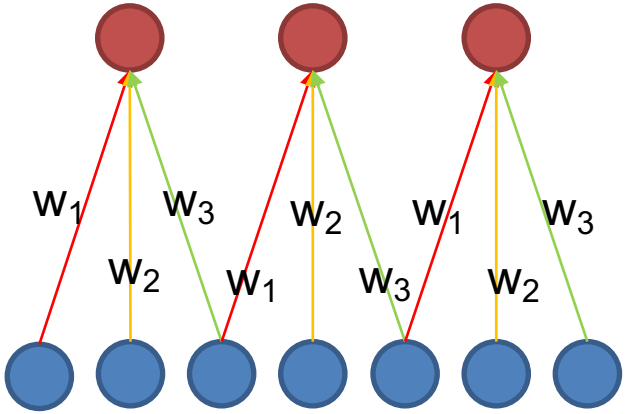
Weight sharing

- Why is weight sharing a valid assumption?
  - Shift invariance property:
    classification output shoud not change with the location of object in the image

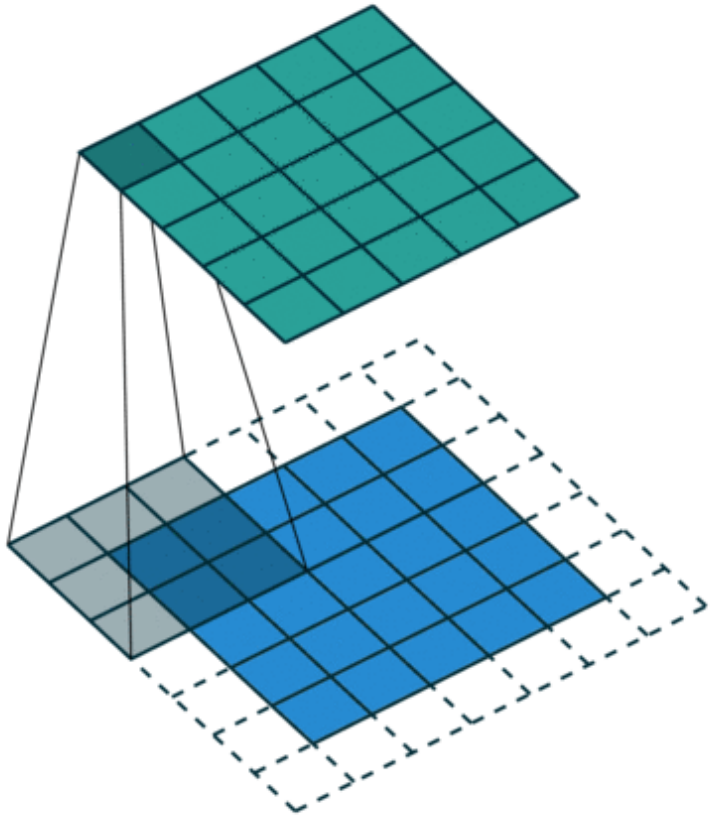


Convolution layer

Local connectivity + weight sharing = convolution!

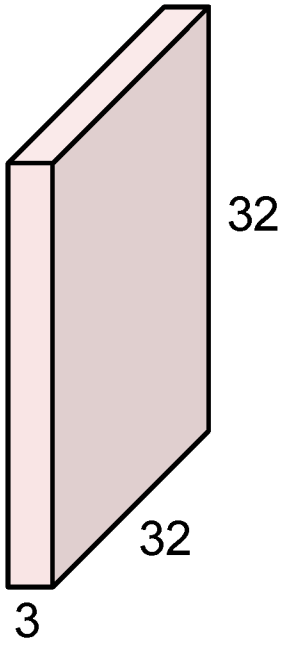


$$f(x) = \sigma(x \otimes w + b)$$



Convolution layer

32x32x3 image



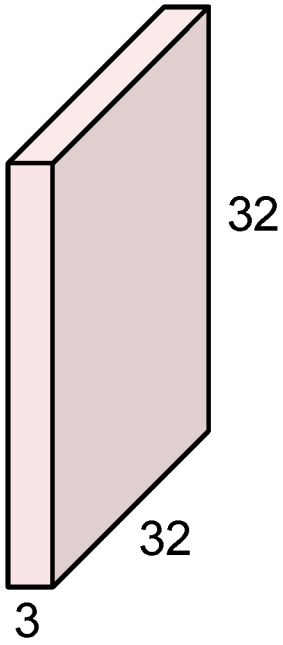
5x5x3 filter

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

Dot product of tensors (“3D matrix”)

Filters always extend the full depth of the input volume

32x32x3 image

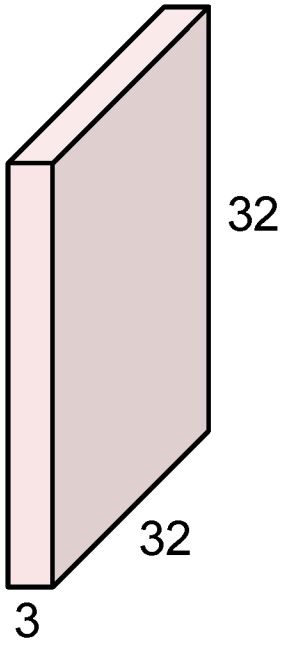


5x5x3 filter

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

Much fewer parameters

32x32x3 image



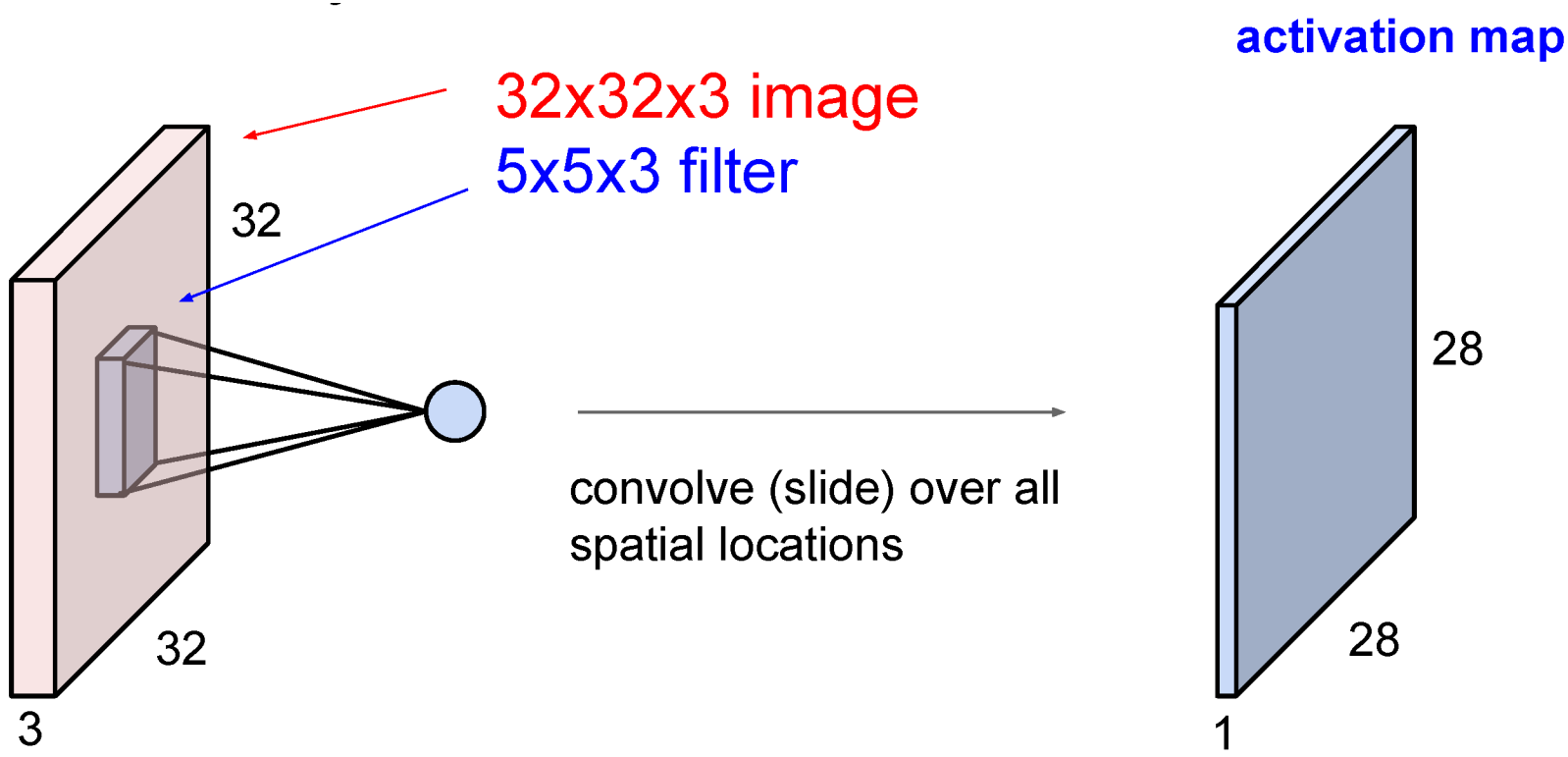
5x5x3 filter

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

Number of weights: 5 x 5 x 3 + 1 = **76**
(vs. 3072 for a fully-connected layer)

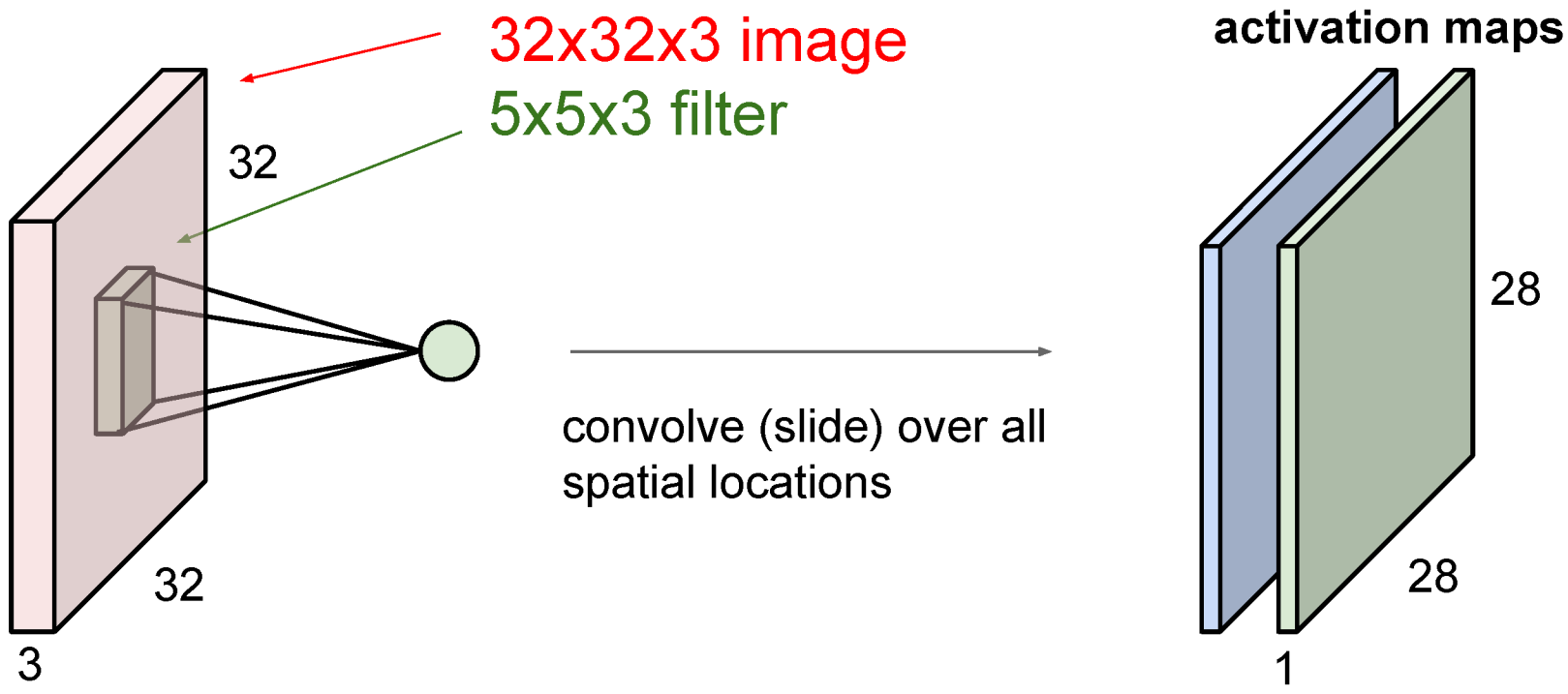
Activation map

- Also called feature map



Activation map

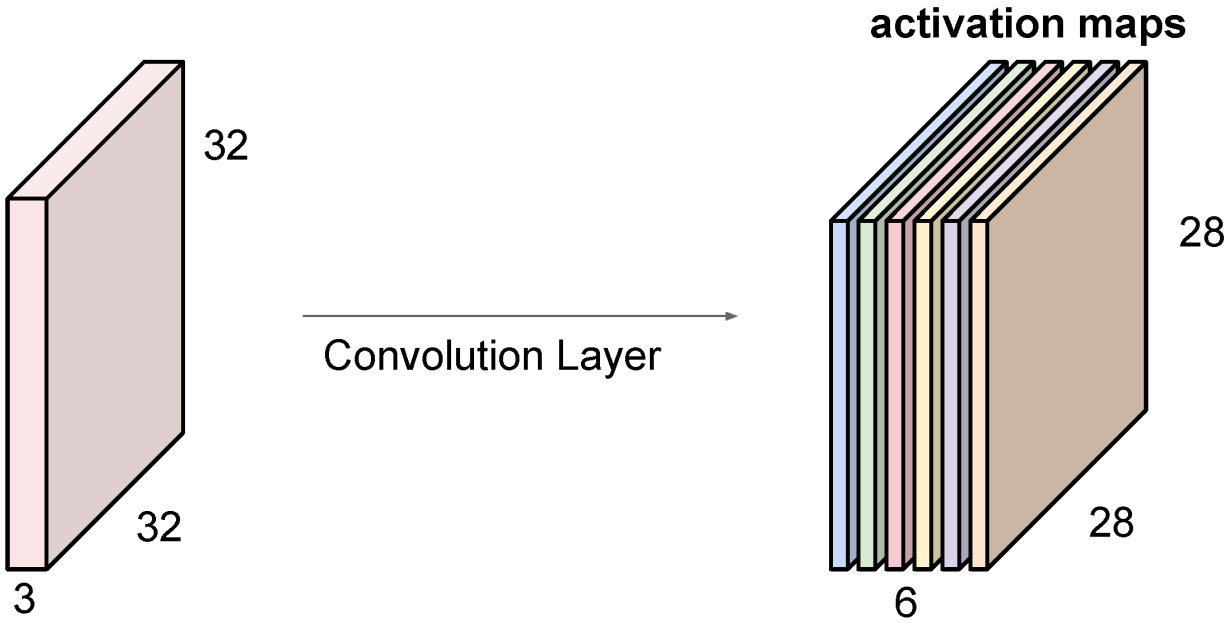
- If there is another filter (green)



Activation map

- Number of filters = number of activation maps

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

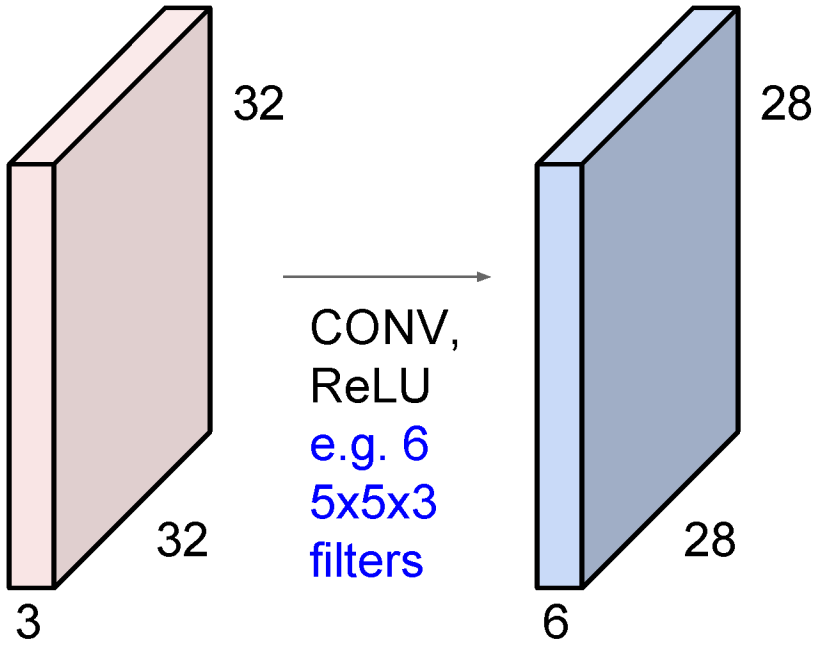


We stack these up to get a "new image" of size 28x28x6!

(total number of parameters: 6 x (75 + 1) = **456**)

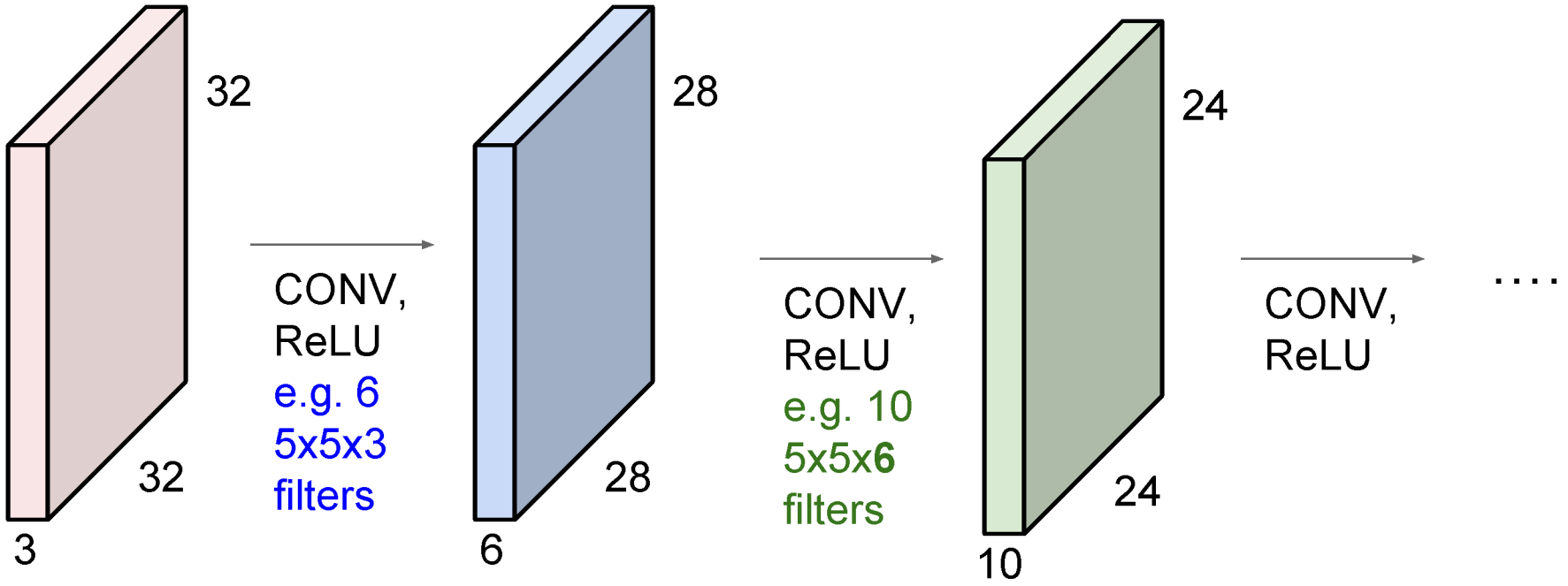
Convolutional neural network (ConvNet)

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



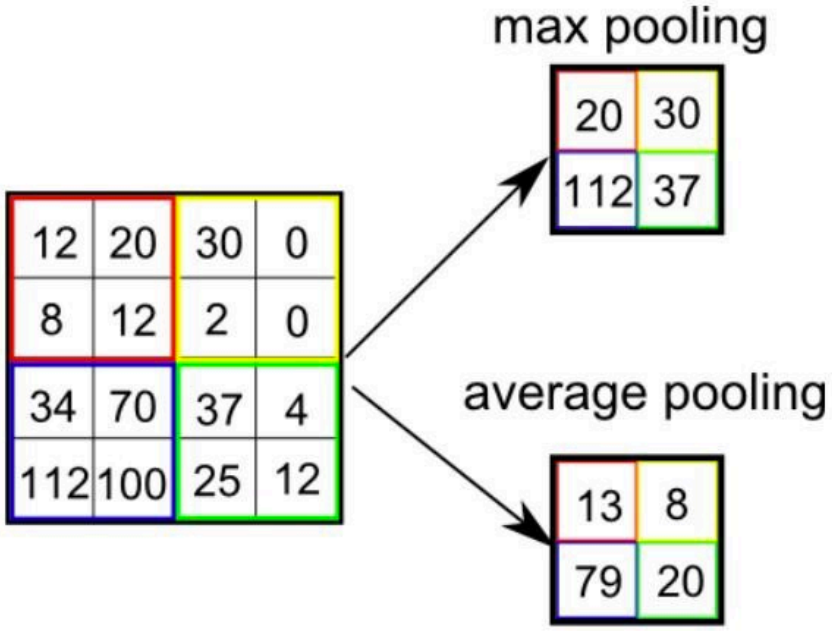
Convolutional neural network (ConvNet)

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



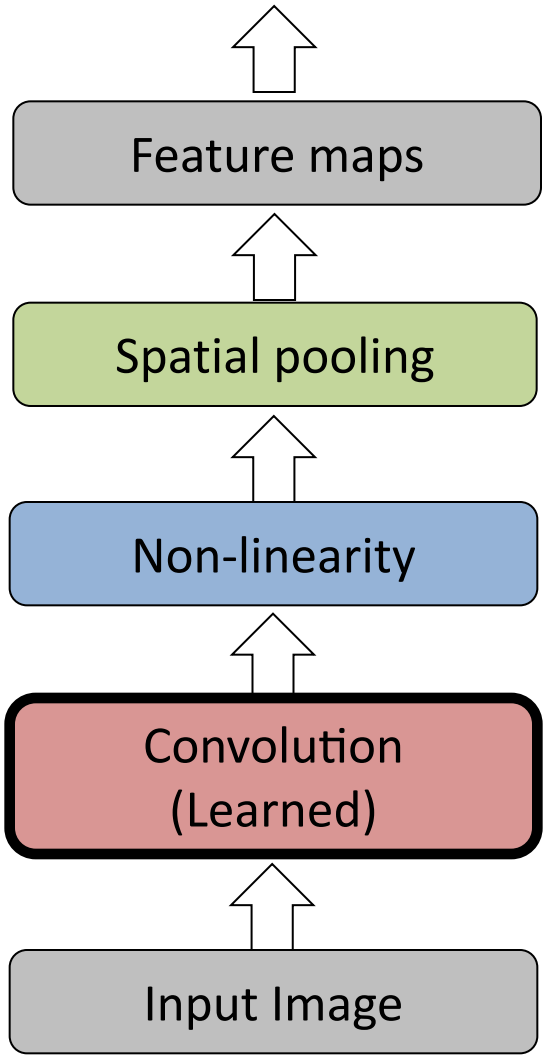
Pooling layer

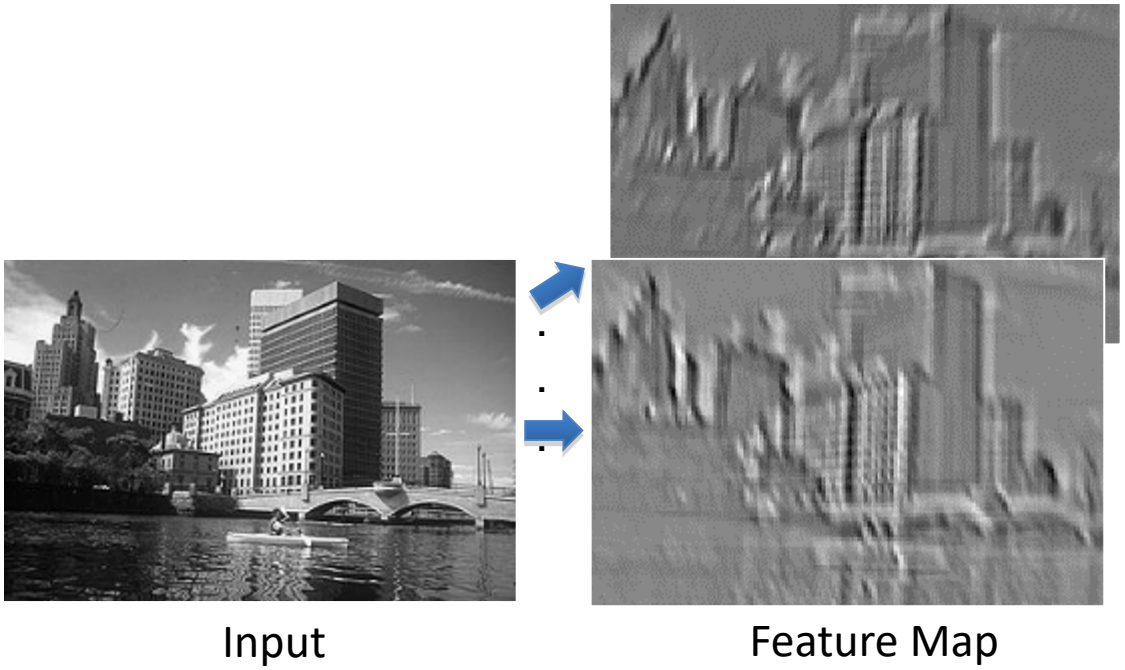
- Pooling layer
  - Max pooling
  - Average pooling



- Why pooling?
  - Makes activation maps smaller and more manageable
  - Aggregate spatial information

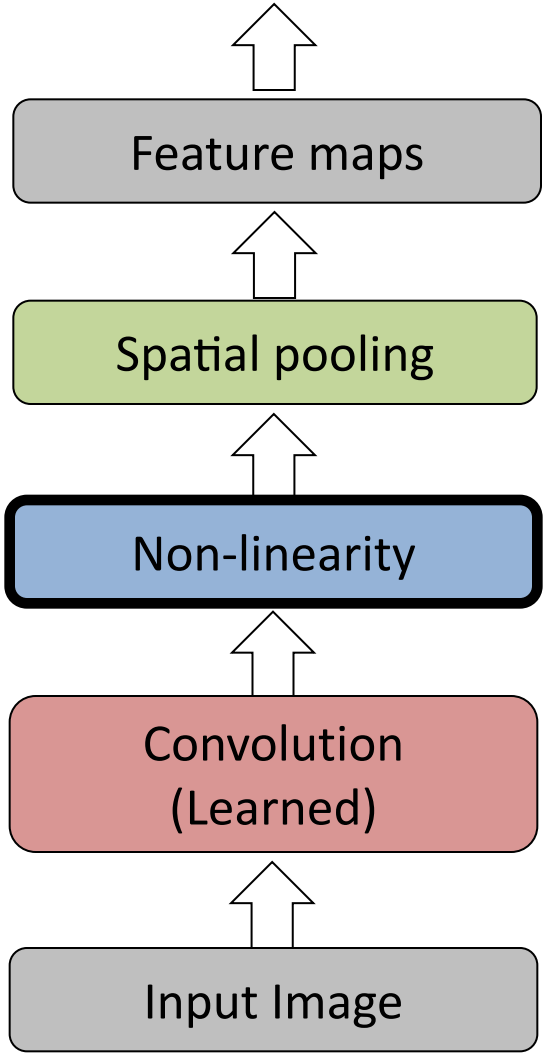
Convolutional Neural Networks



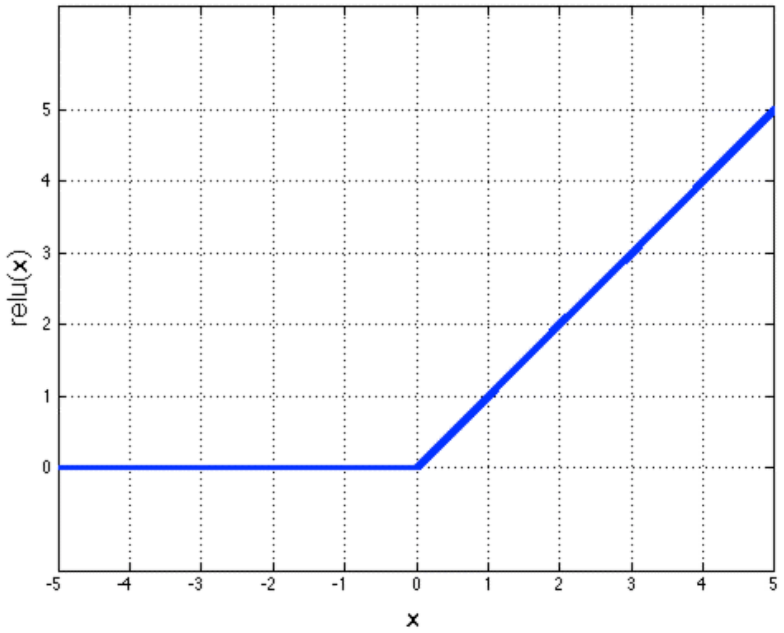


slide credit: S. Lazebnik

Convolutional Neural Networks

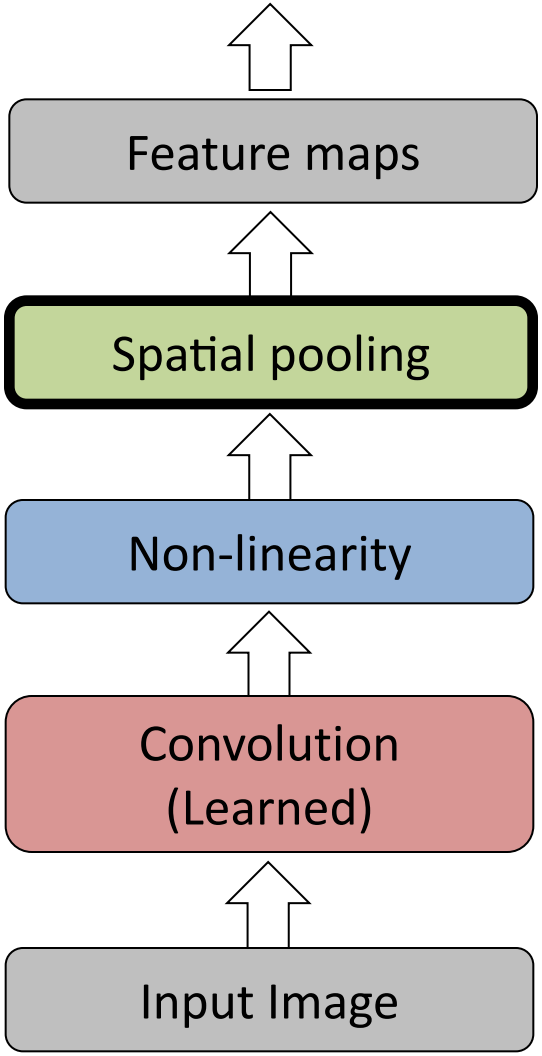


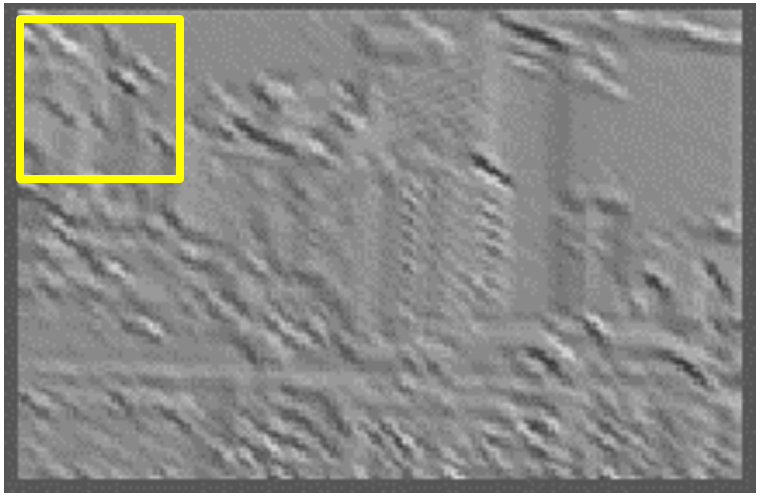
Rectified Linear Unit (ReLU)



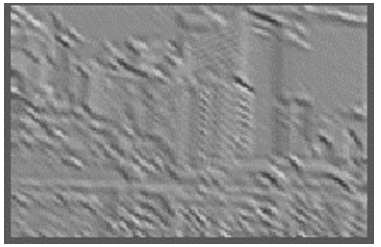
slide credit: S. Lazebnik

Convolutional Neural Networks





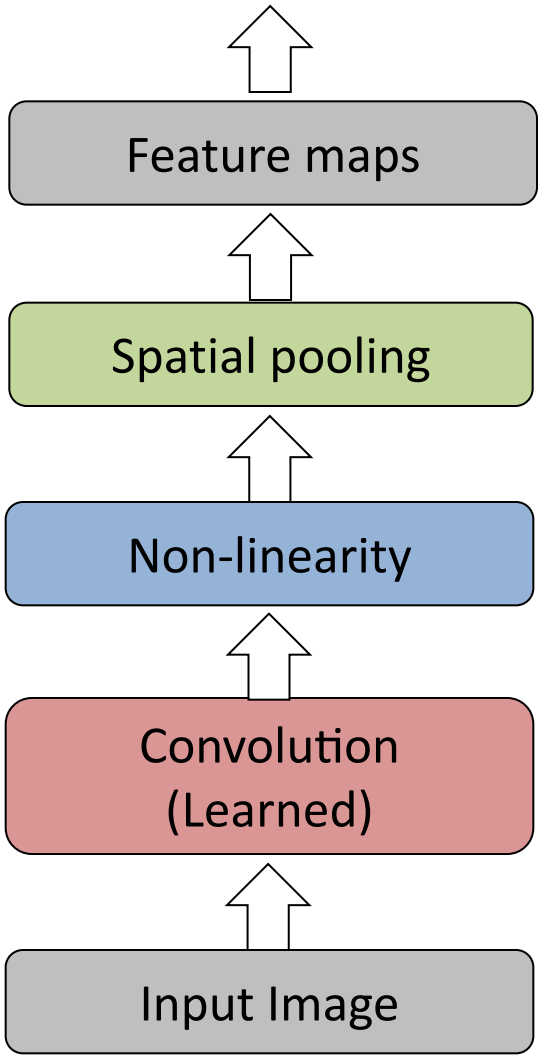
Max pooling

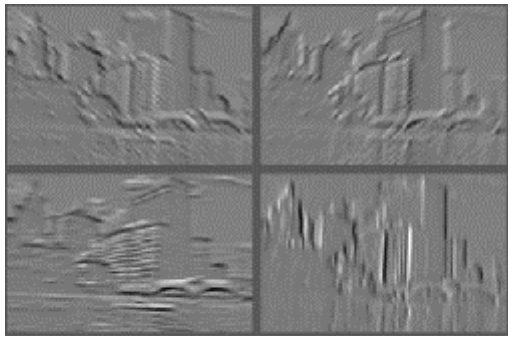


down-sampling and aggregation

slide credit: S. Lazebnik

Convolutional Neural Networks



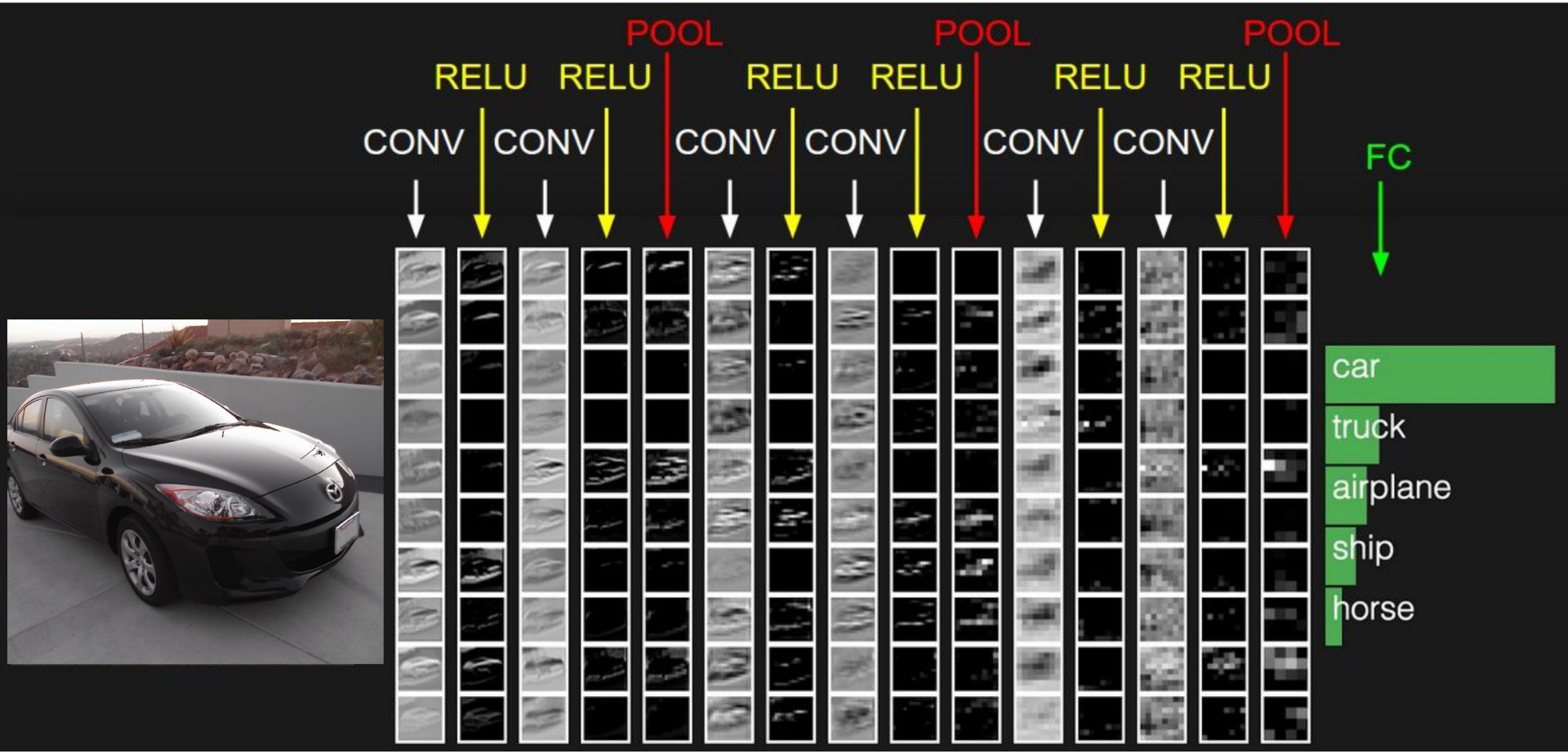


Feature Maps

slide credit: S. Lazebnik

CNN for classifcation

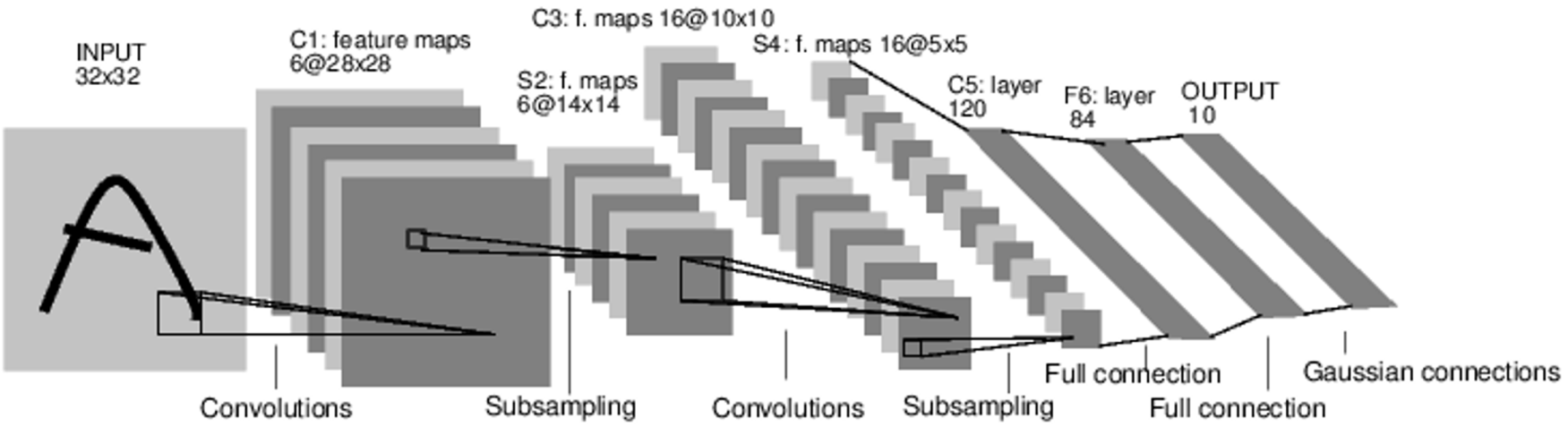
preview:



The final fully connected layer converts the feature maps into class scores

Case Study: LeNet-5

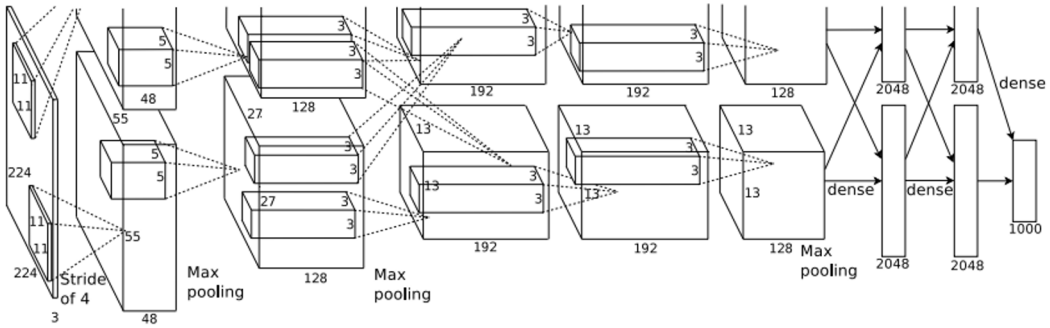
[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

Case Study: AlexNet

[Krizhevsky et al. 2012]



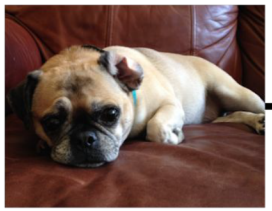
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

What do the filters look like?

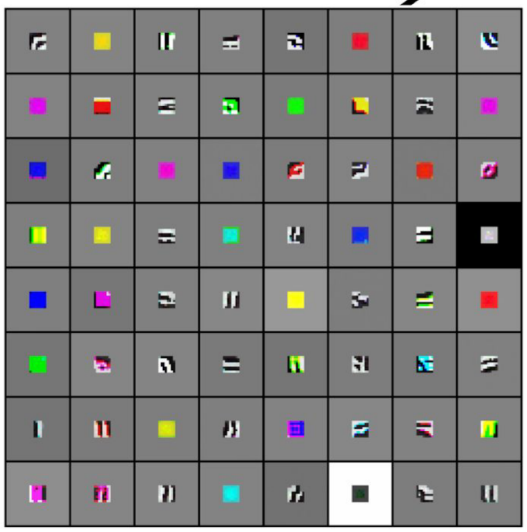
Preview

[Zeiler and Fergus 2013]

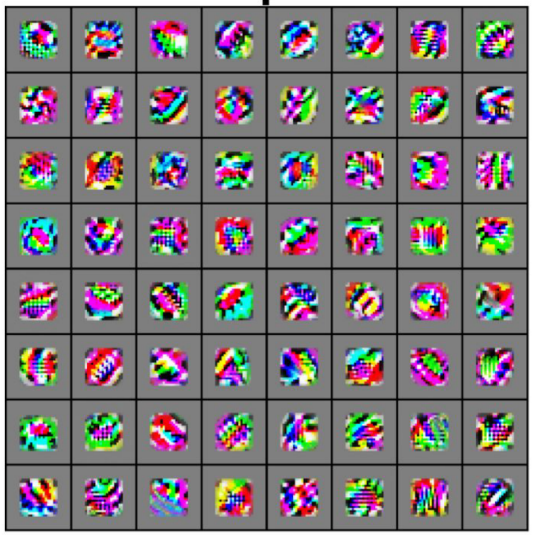
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



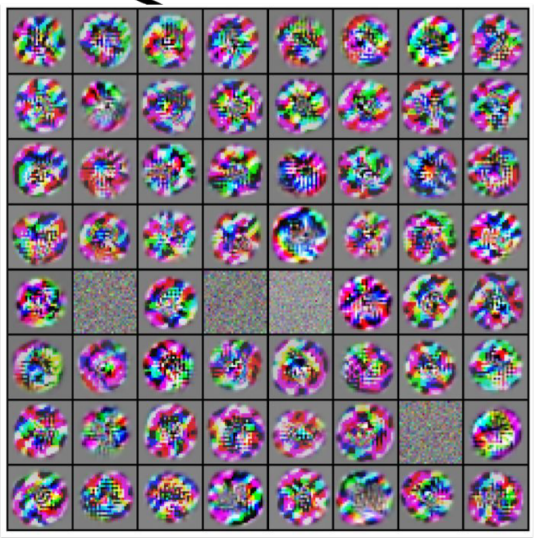
Low-level features → Mid-level features → High-level features → Linearly separable classifier



VGG-16 Conv1_1



VGG-16 Conv3_2



VGG-16 Conv5_3

Summary

- CNN = a multi-layer neural network with local connectivity + weight sharing

- Local connectivity + weight sharing = convolution

- Layer types:
  - Fully-connected layer
  - *Convolutional layer*
  - Pooling layer

Part V

Training Neural Networks

训练神经网络

Multi-layer Neural Network

- A non-linear classifier
- **Training:** find network weights **w** to minimize the error between true training labels y_i and estimated labels $f_{\boldsymbol{w}}(\boldsymbol{x_i})$,

$$L(\boldsymbol{w}) = \frac{1}{n}\sum_i l(y_i, f_{\boldsymbol{w}}(\boldsymbol{x_i}))$$

  For example:
  - L2 loss for regression
  - Cross-entropy loss for classification

- Minimization can be done by gradient descent provided f is differentiable

- This training method is called **back-propagation**

Training CNN with gradient descent

- A CNN as composition of functions

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = f_L(\dots\ (f_2(f_1(\boldsymbol{x}; \boldsymbol{w}_1); \boldsymbol{w}_2)\ \dots; \boldsymbol{w}_L)$$

- Parameters

$$\boldsymbol{w} = (\boldsymbol{w_1}, \boldsymbol{w_2}, \dots \boldsymbol{w_L})$$

- Loss function

$$L(\boldsymbol{w}) = \frac{1}{n}\sum_i l(y_i, f_{\boldsymbol{w}}(\boldsymbol{x_i}))$$

- Gradient descent

$$\boldsymbol{w^{t+1}} = \boldsymbol{w^t} - \eta_t \frac{\partial L}{\partial \boldsymbol{w}}(\boldsymbol{w^t})$$

New weight

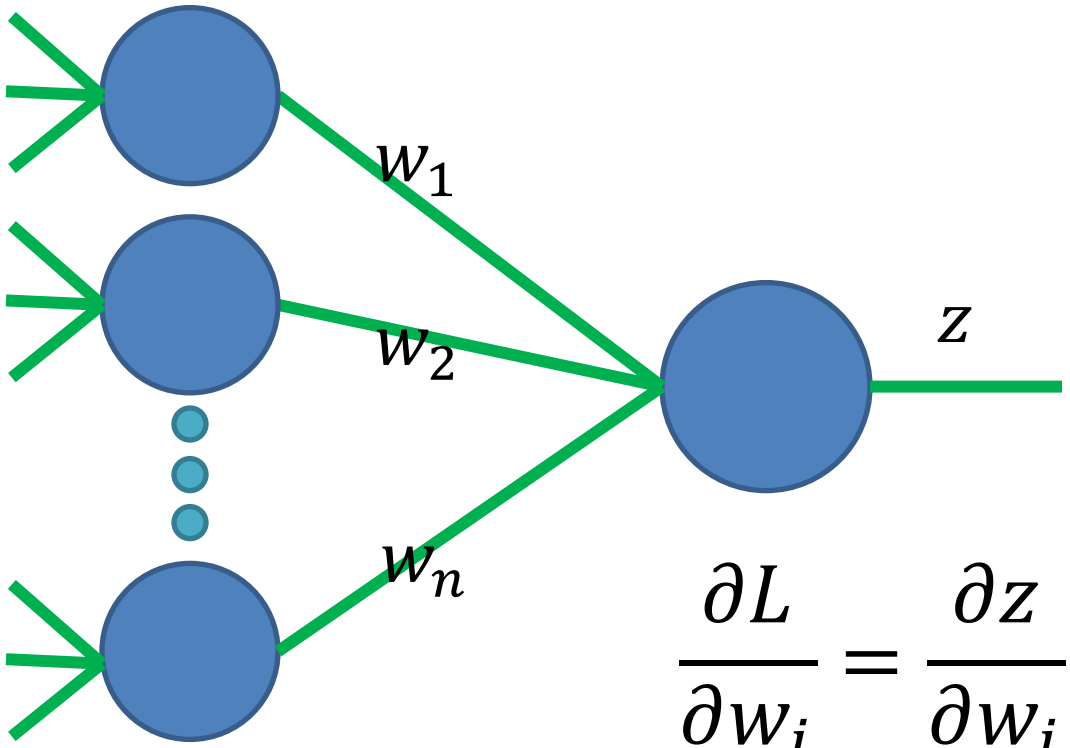
Old weight

Learning rate

Gradient

Backpropagation

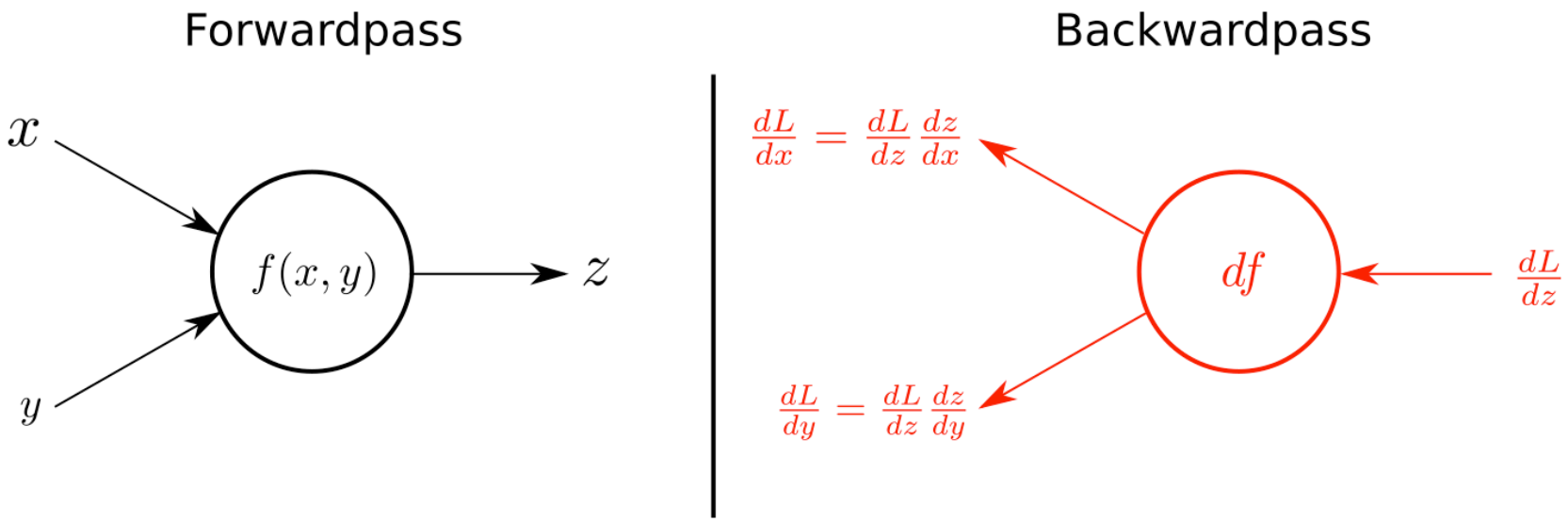
- Use chain rule to calculate gradients of Loss w.r.t. weights



$$\frac{\partial L}{\partial w_i} = \frac{\partial z}{\partial w_i}\frac{\partial L}{\partial z}$$

Backpropagation

- Algorithm
  1. **Forward** data through the network, get loss
  2. **Backprop** to calculate the gradients
  3. **Update** the parameters using the gradient
  4. Go to step 1 if not converged



Stochastic Gradient Descent (SGD)

- Sometimes the training data is too large, e.g., millions of images in ImageNet

- Calculate loss and gradients over all images in each iteration is too expensive

- Stochastic gradient descent: only calcualte loss and gradients on a batch of randomly sampled images

$$\hat{L}(\boldsymbol{w}) = \frac{1}{n}\sum_{i\in\Omega} l(y_i, f_{\boldsymbol{w}}(\boldsymbol{x_i})), \quad SG = \frac{\partial \hat{L}}{\partial \boldsymbol{w}}(\boldsymbol{w^t})$$

- Stochastic gradients approximate the real gradients

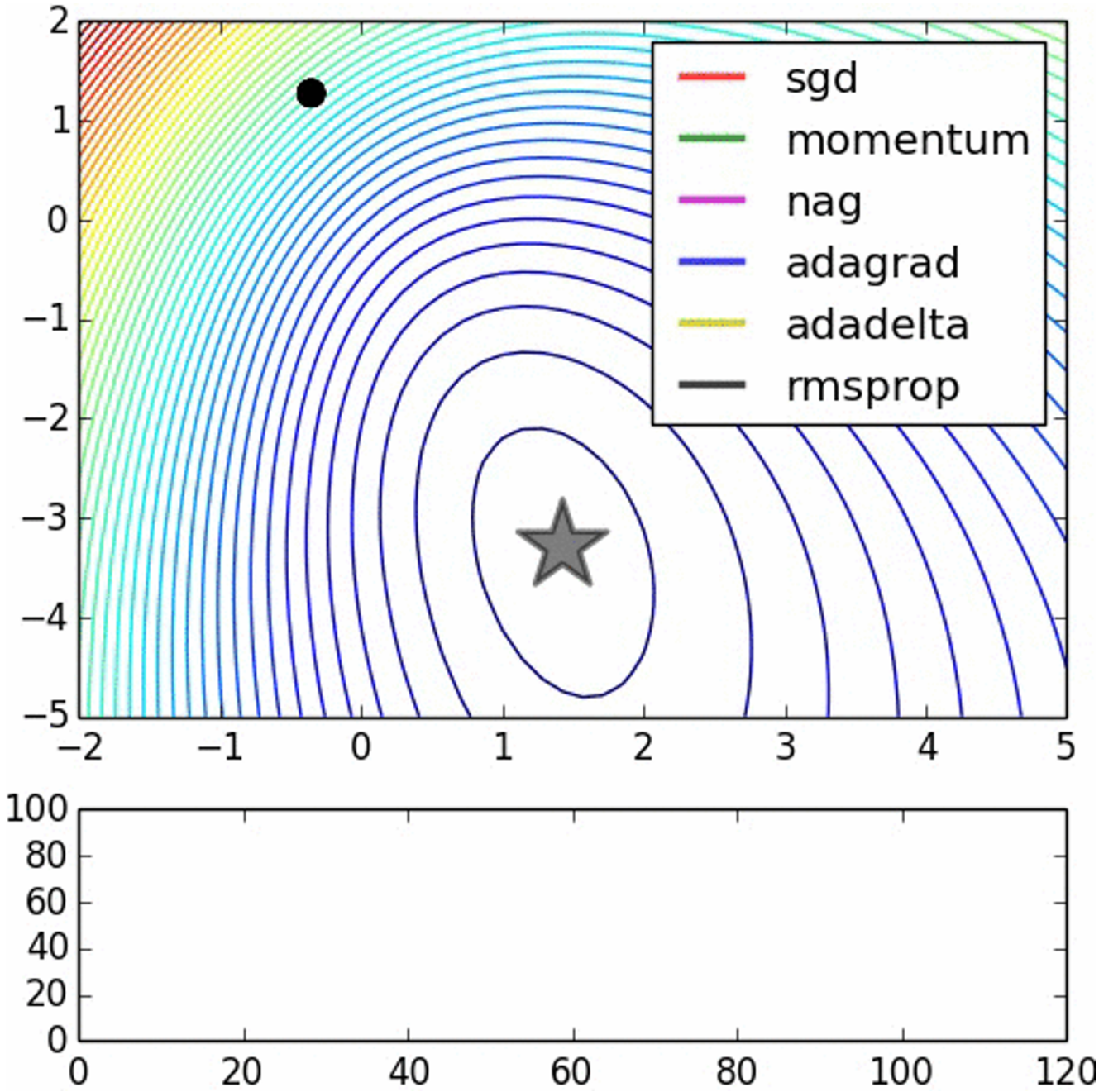


Image credits: Alec Radford

Architecture and hyper-parameters

- Hyper-parameters
  - How many layers to use?
  - How many filters in each layer?
  - What are the best batch size and learning rate?

- How do we set them?
  - One option: try them all and see what works best

Data split

Idea #1: Choose hyperparameters that work best on the data

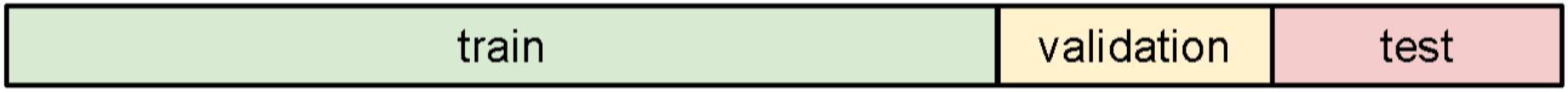
Bad: you can always decrease training loss by using a larger network

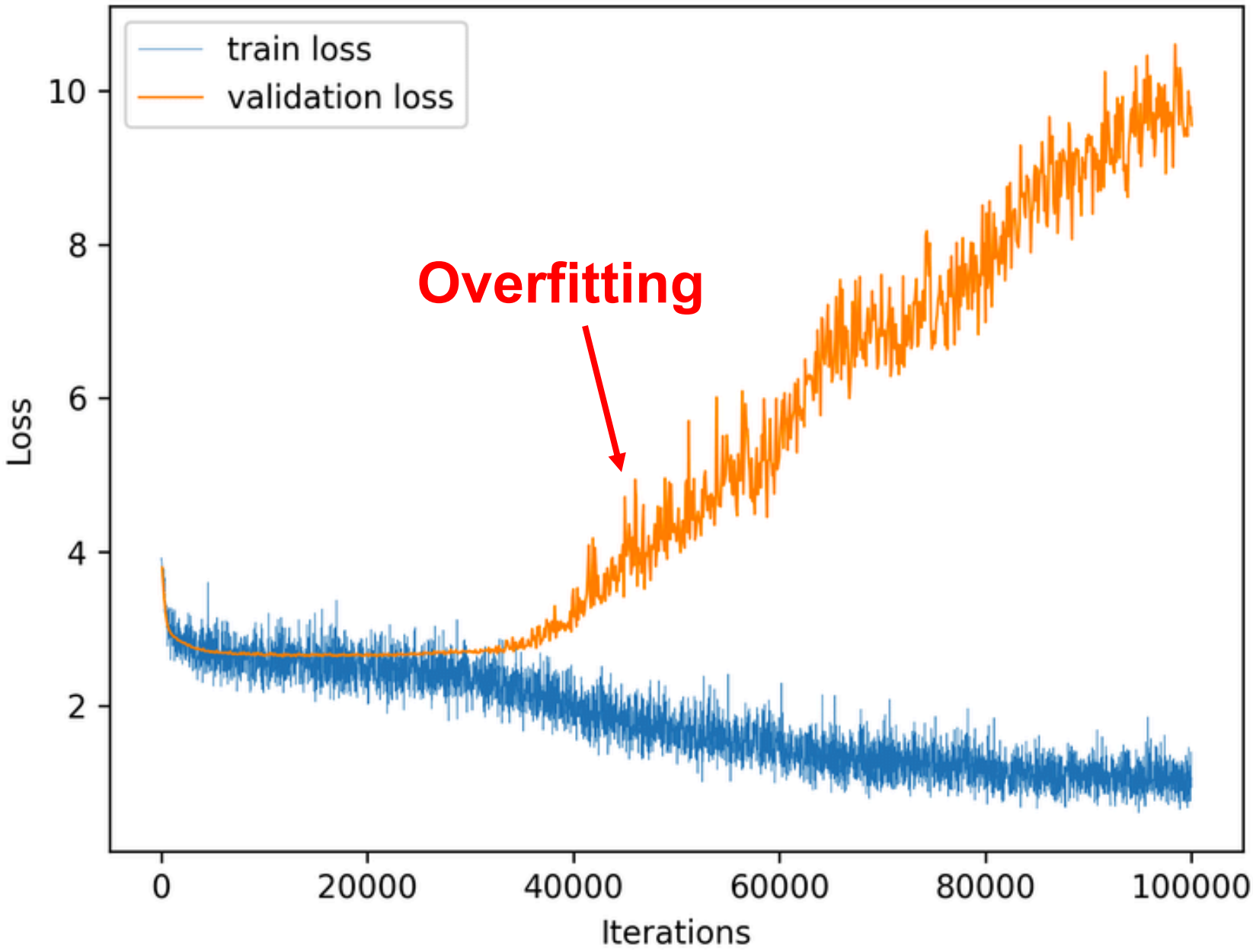
Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

Bad: no idea how it will perform on new data

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Best!

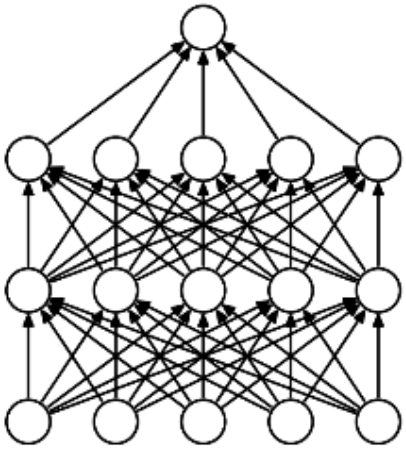



train loss
validation loss
Overfitting
Loss
10
8
6
4
2
0
20000
40000
60000
80000
100000
Iterations

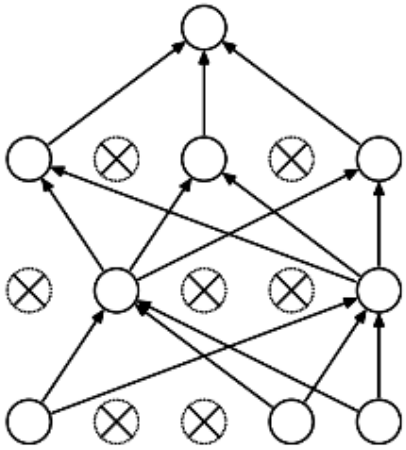
Recap: How to pick hyperparameters?

- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

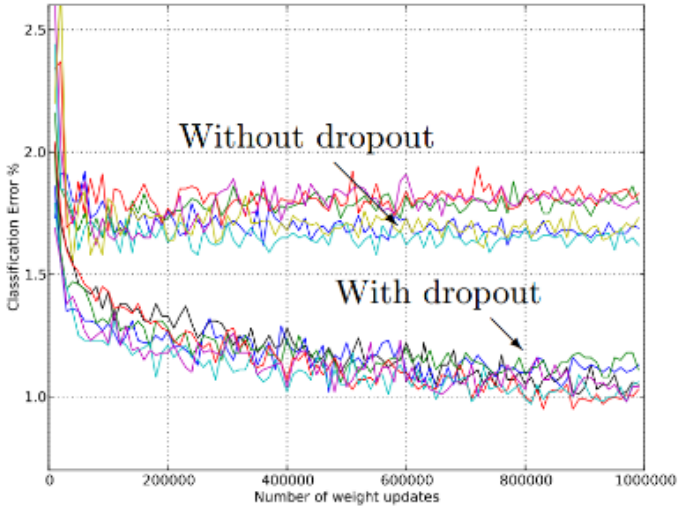
Dropout

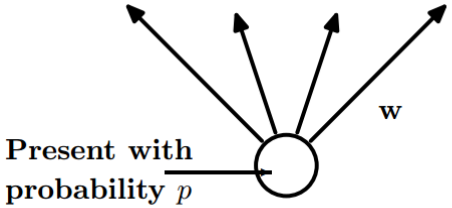


(a) Standard Neural Net

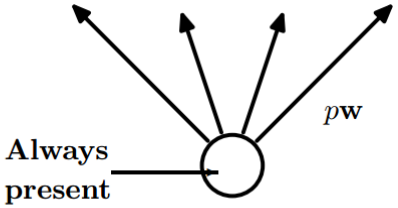


(b) After applying dropout.





(a) At training time



(b) At test time

Main Idea: approximately combining exponentially many different neural network architectures efficiently

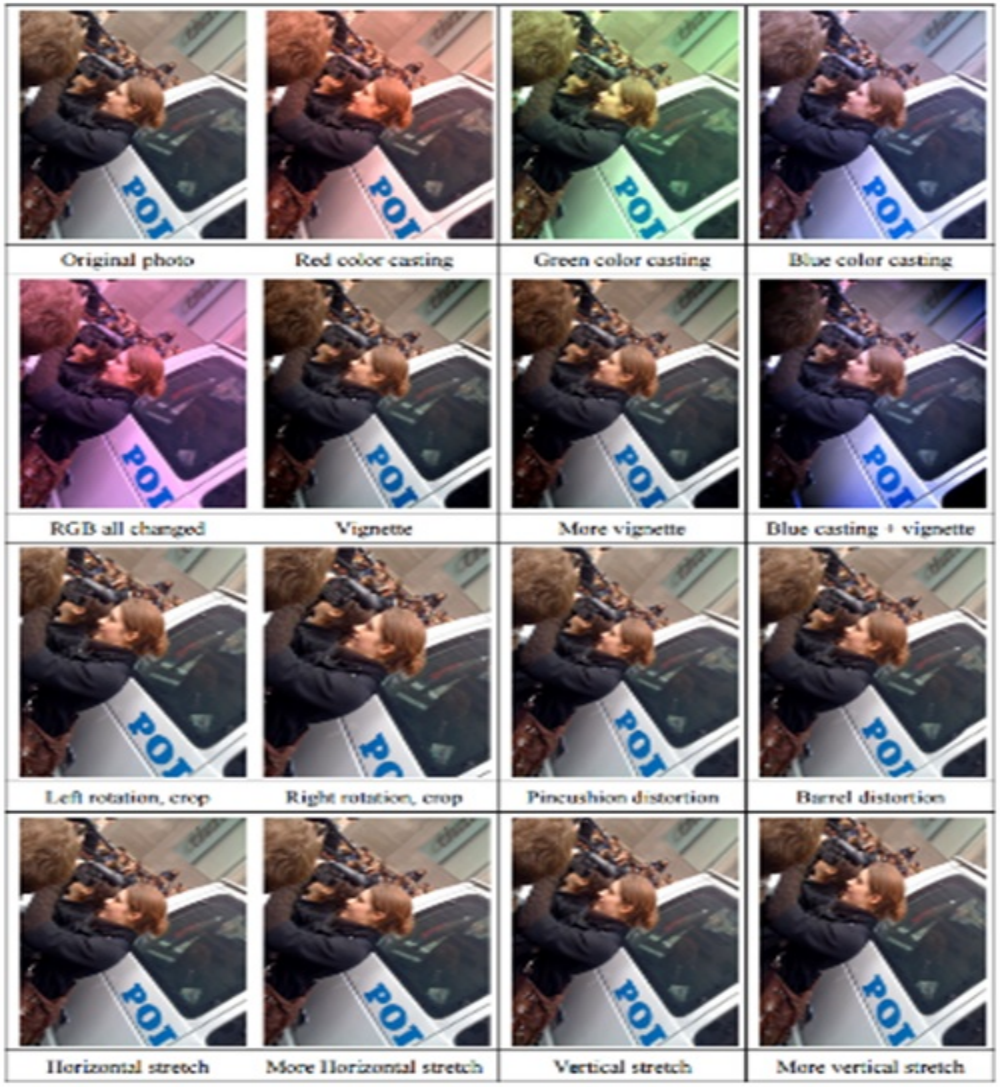
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Data Augmentation (Jittering)

- Create *virtual* training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion



Deep Image [Wu et al. 2015]

Part VI

History and Recent Advances

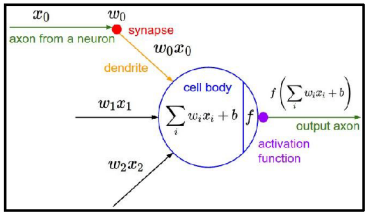
A bit of history...

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

recognized
letters of the alphabet



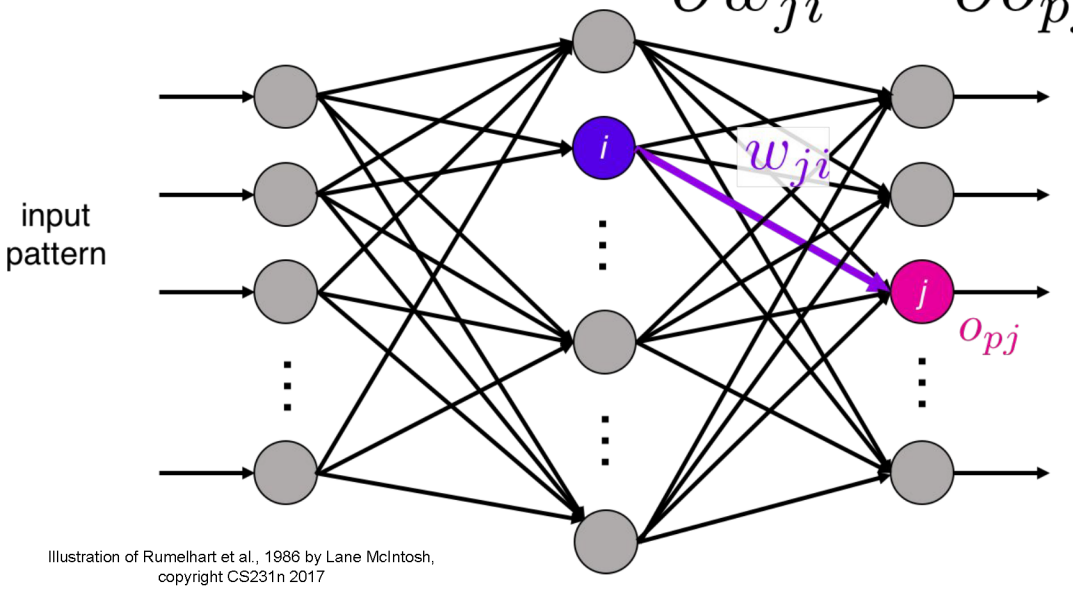
update rule:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

Frank Rosenblatt, ~1957: Perceptron

A bit of history...

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

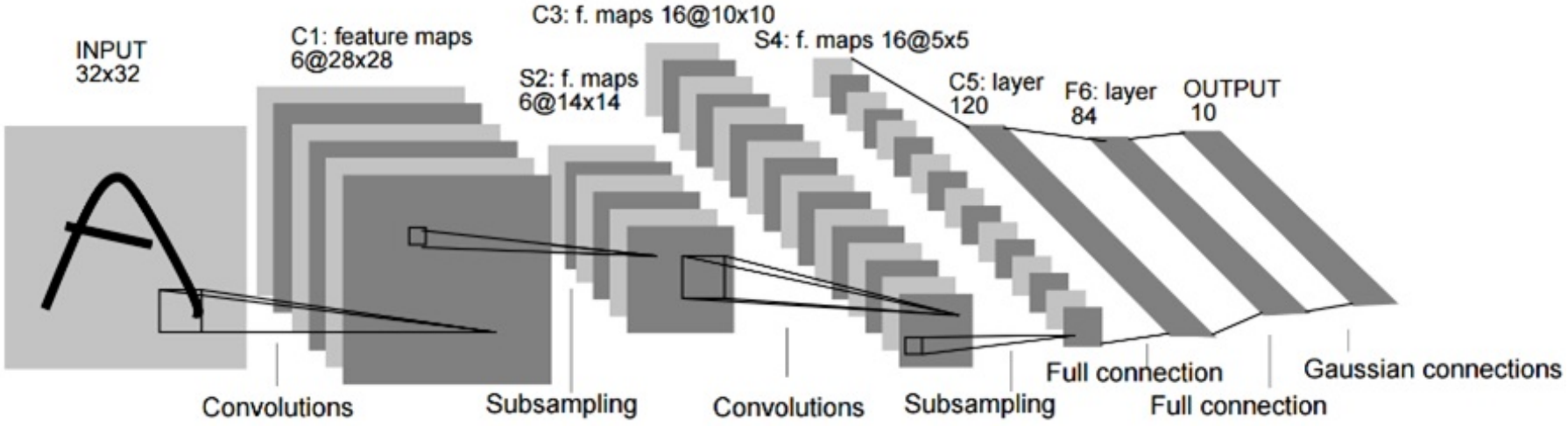


recognizable math

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

Rumelhart et al., 1986: First time back-propagation became popular

LeNet [LeCun et al. 1998]



Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998]

LeNet-1 from 1993

But neural networks were not that successful before

- Why?
  - Small training datasets lead to overfitting
  - Hard to train a deep neural network due to limited computational power

ImageNet dataset

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

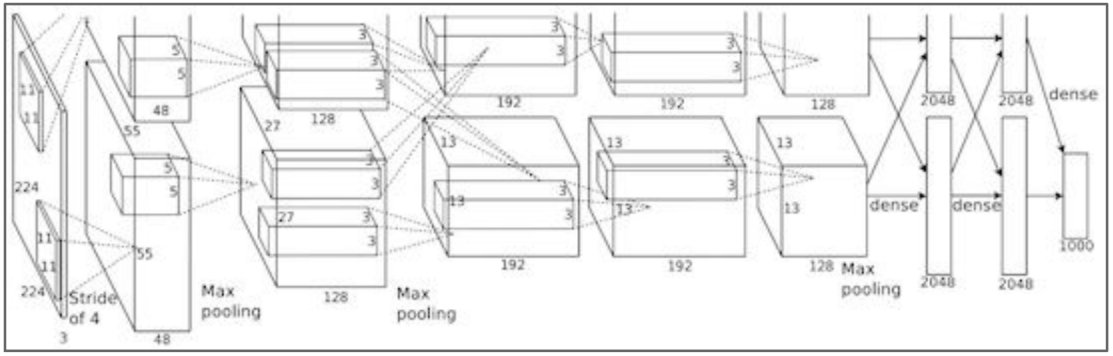
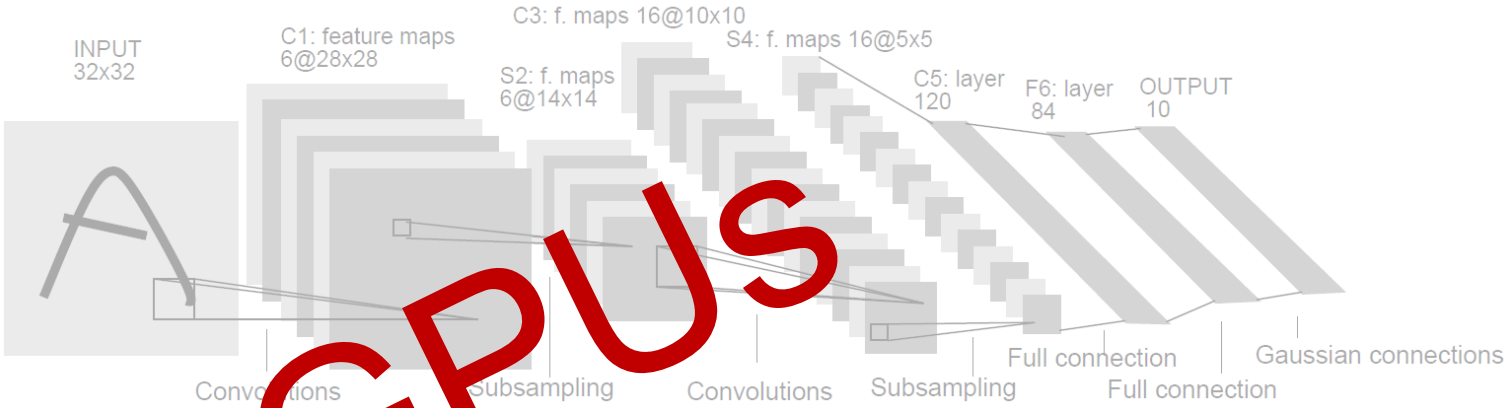


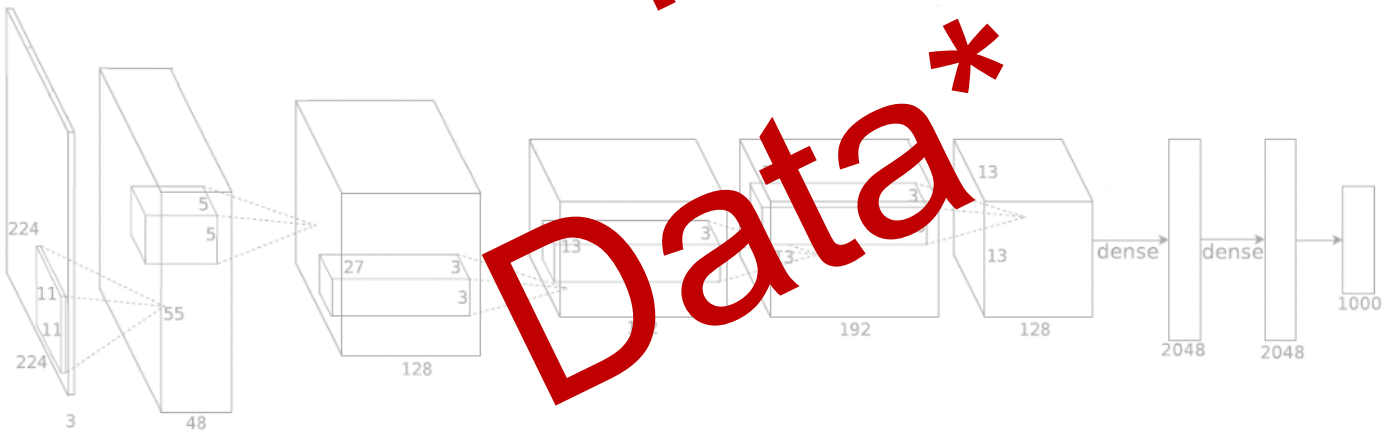
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"



Gradient-Based Learning Applied to Document Recognition, LeCun, Bottou, Bengio and Haffner, Proc. of the IEEE, **1998**

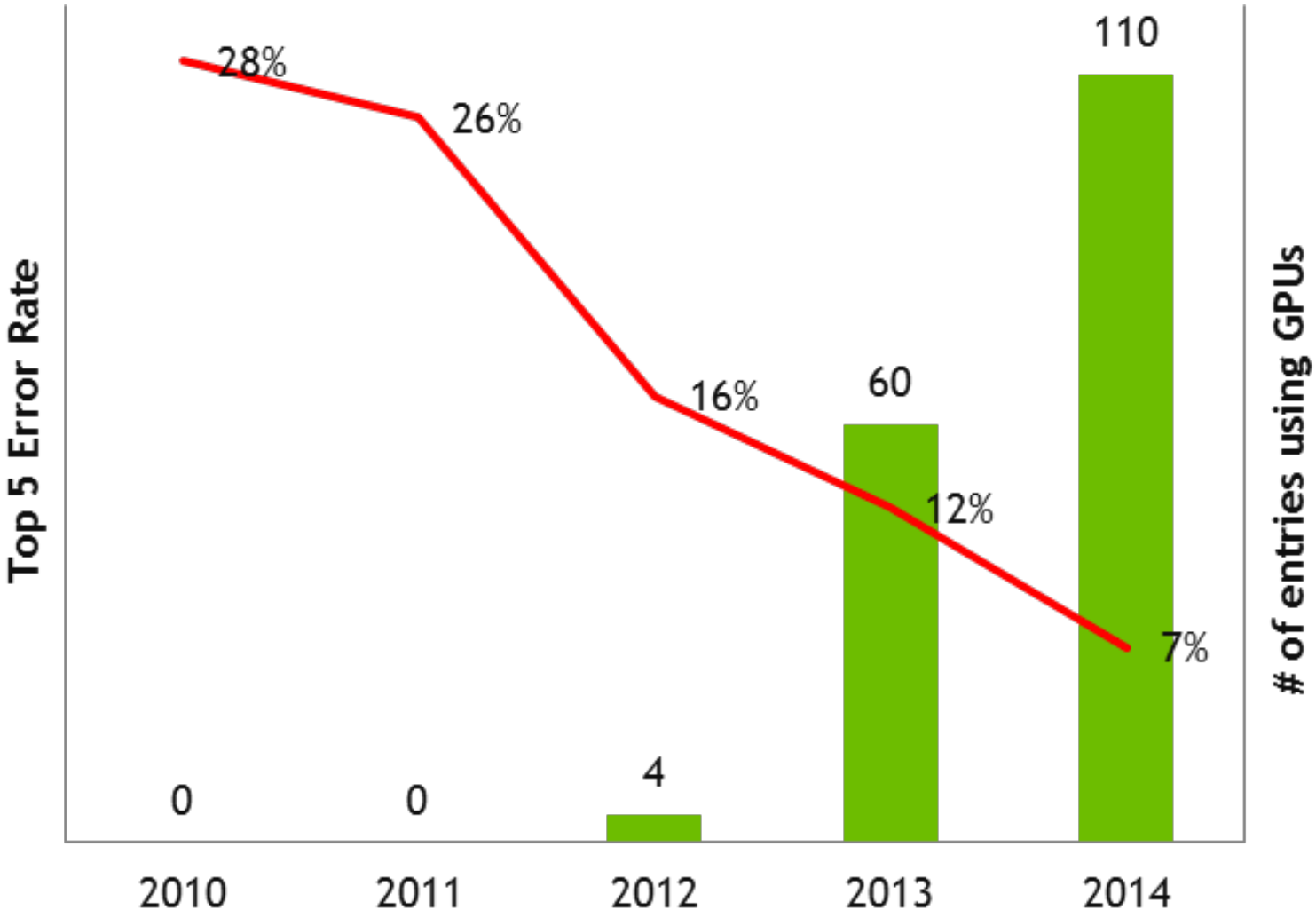
GPUs + Data*



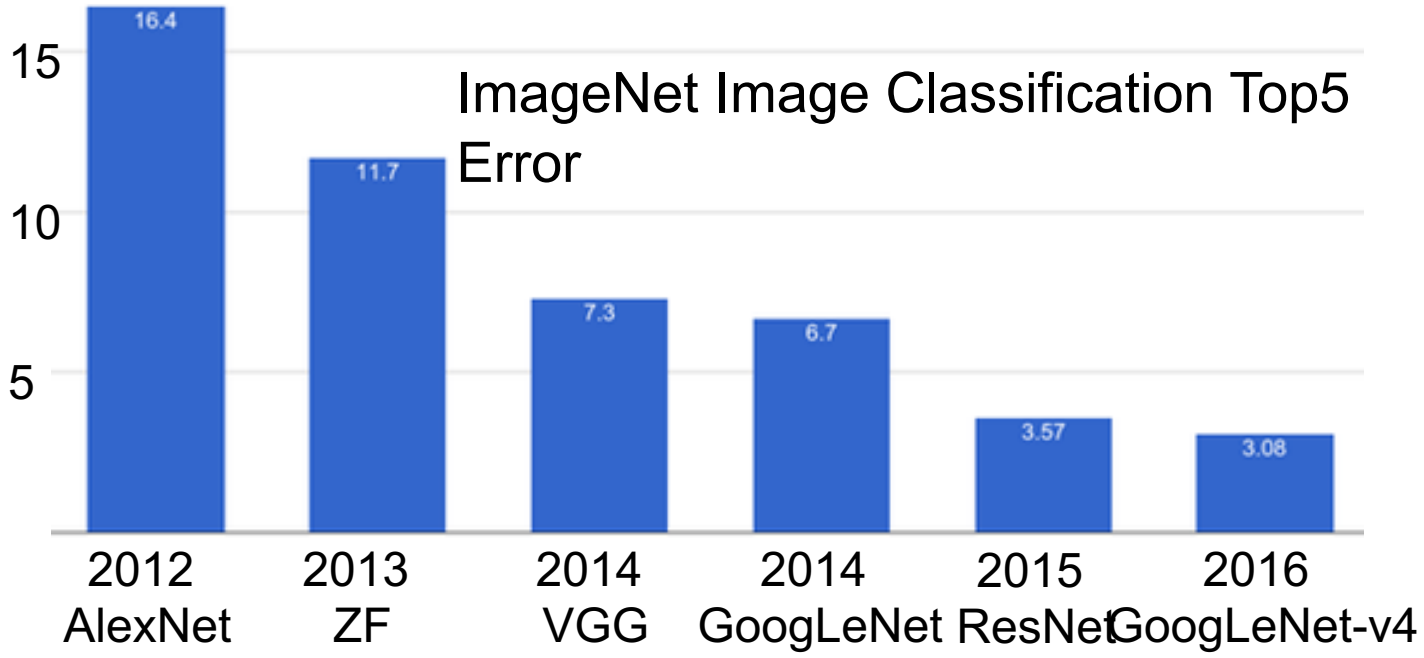
Imagenet Classification with Deep Convolutional Neural Networks, Krizhevsky, Sutskever, and Hinton, NIPS **2012**

Slide Credit: L. Zitnick

IMAGENET

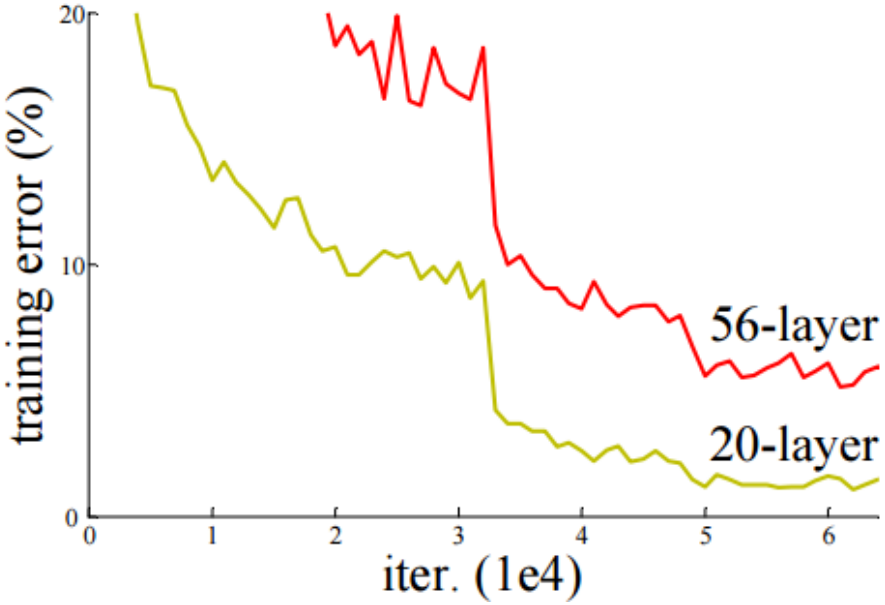

Top 5 Error Rate
# of entries using GPUs
28%
26%
16%
12%
7%
0
0
4
60
110
2010
2011
2012
2013
2014

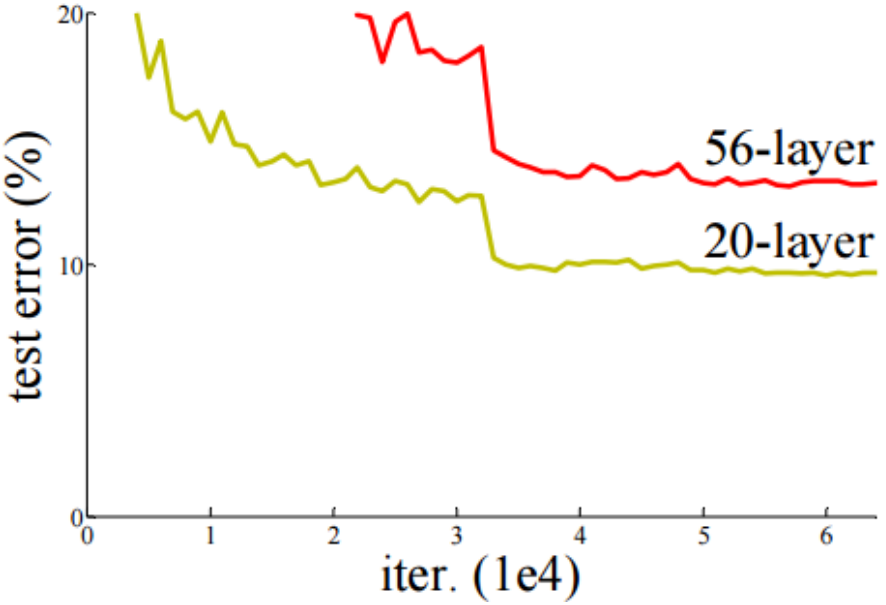
Progress on ImageNet



ResNet

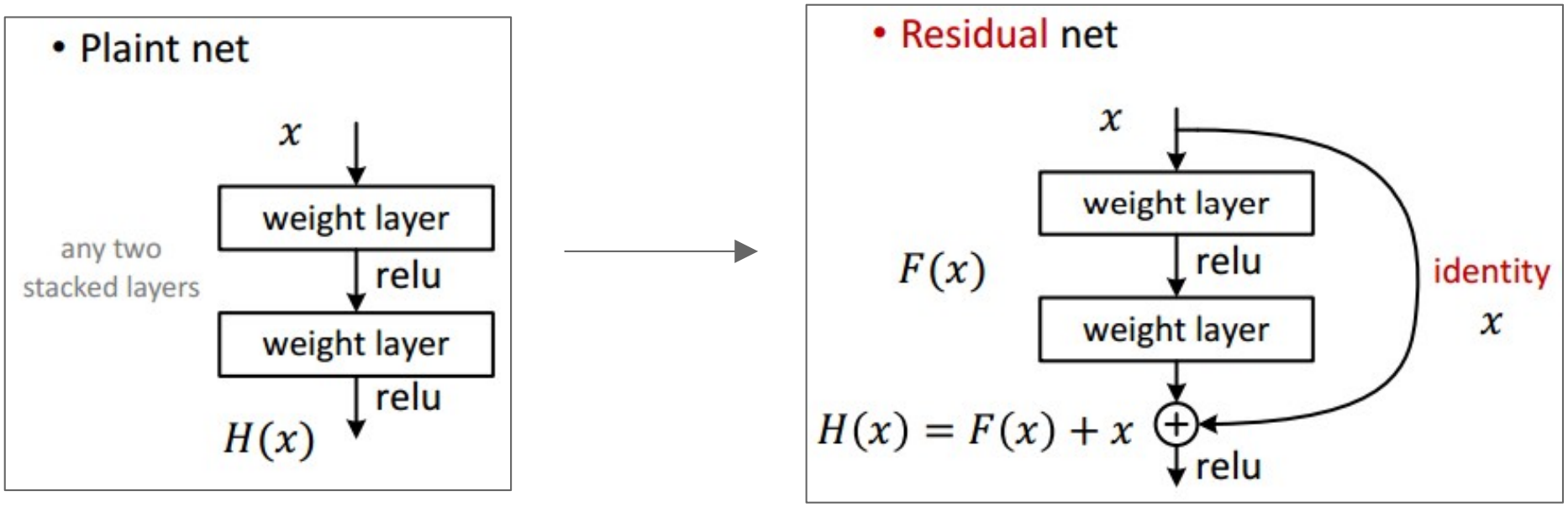
- Can we just increase the #layer?





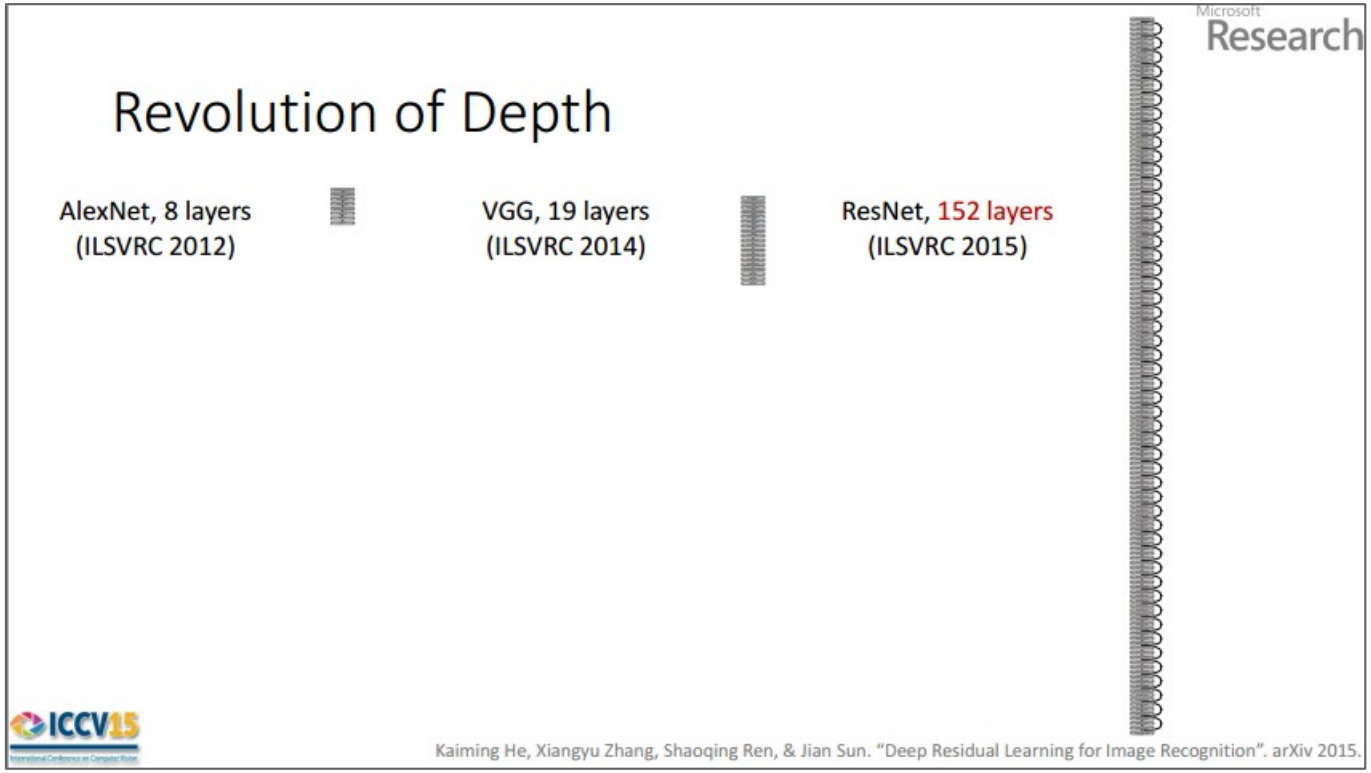
ResNet

- How can we train very deep network?
  - Residual learning



ResNet

ILSVRC 2015 winner (3.6% top 5 error)

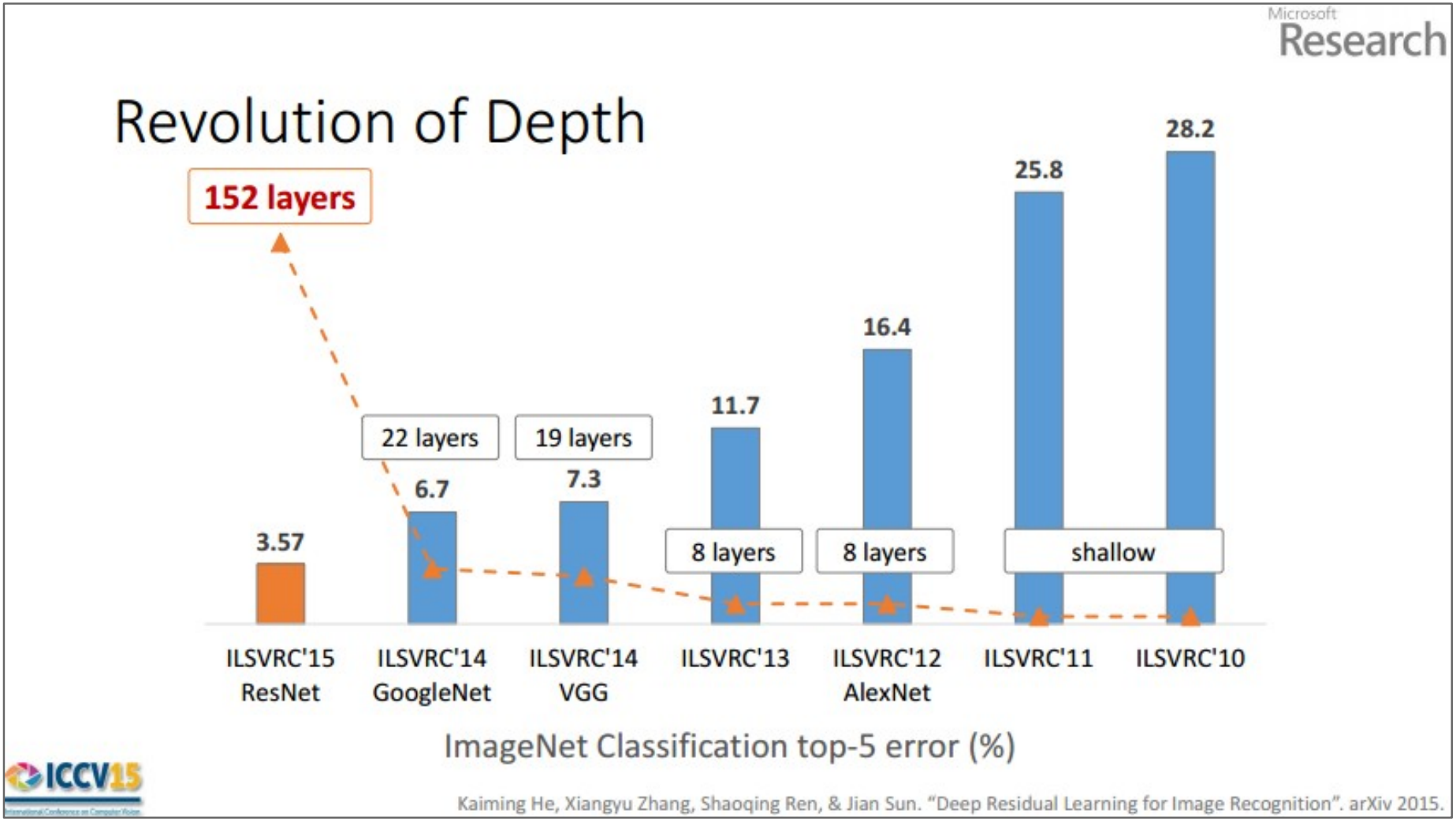


2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

(slide from Kaiming He's recent presentation)

ResNet

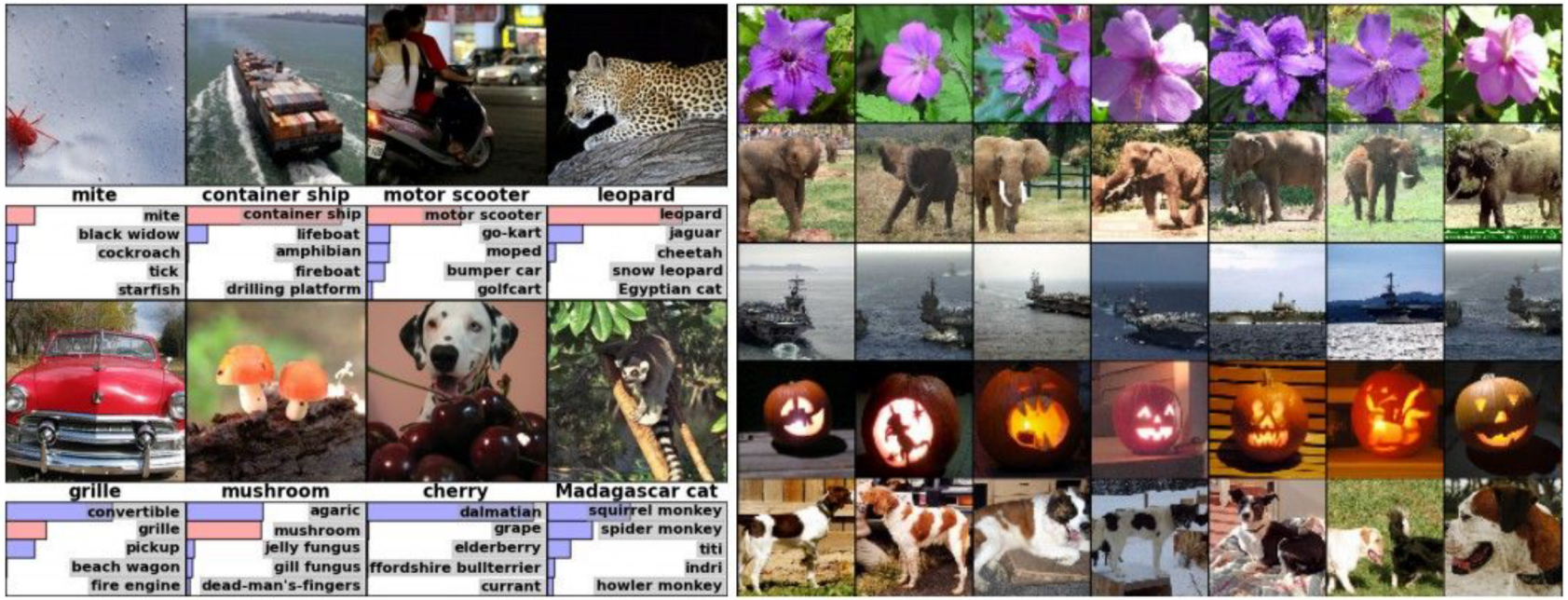


(slide from Kaiming He's recent presentation)

Fast-forward to today: ConvNets are everywhere

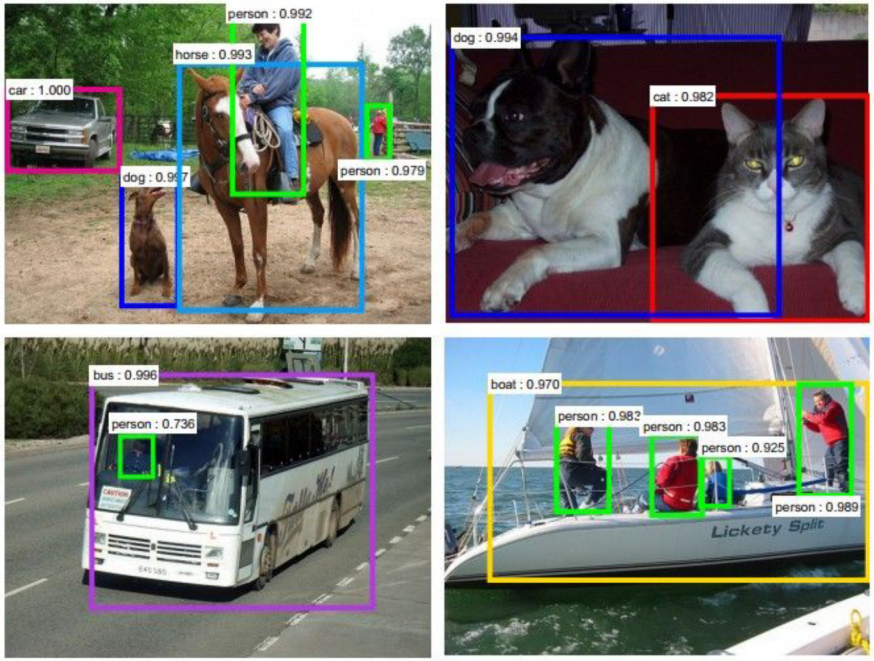
Classification

Retrieval



Fast-forward to today: ConvNets are everywhere

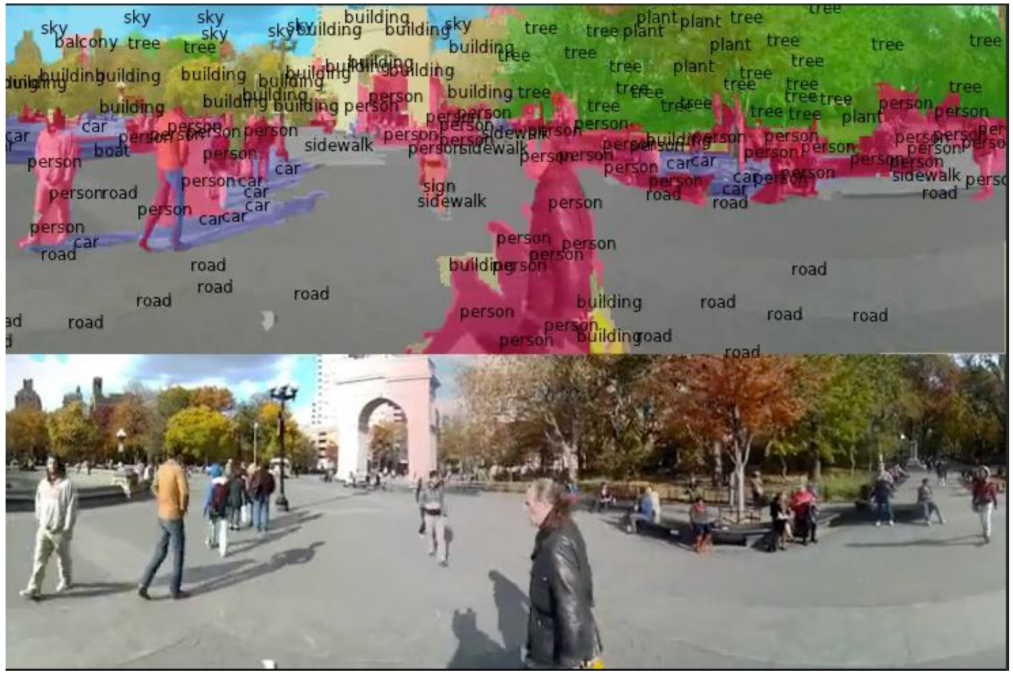
Detection



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

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

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Fast-forward to today: ConvNets are everywhere

Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

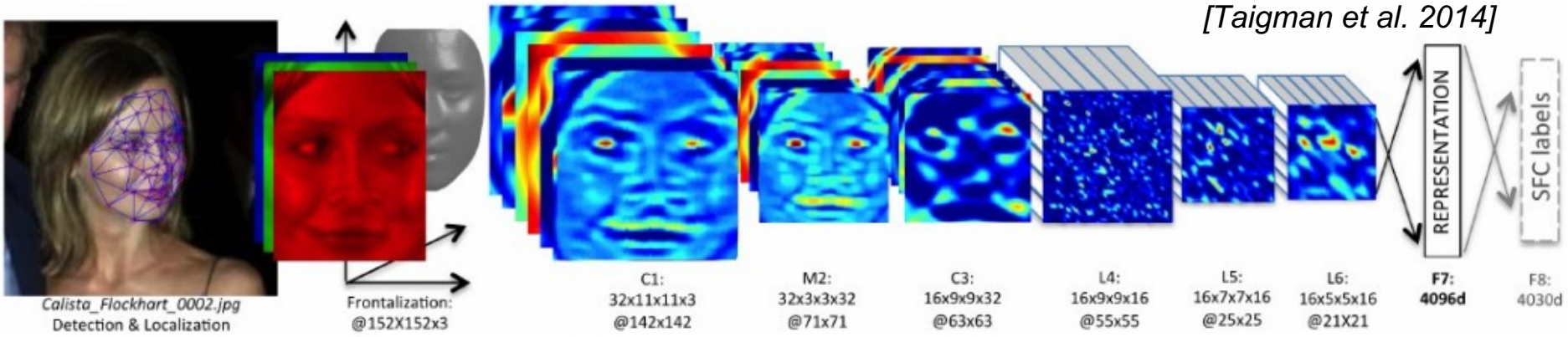
NVIDIA Tesla line

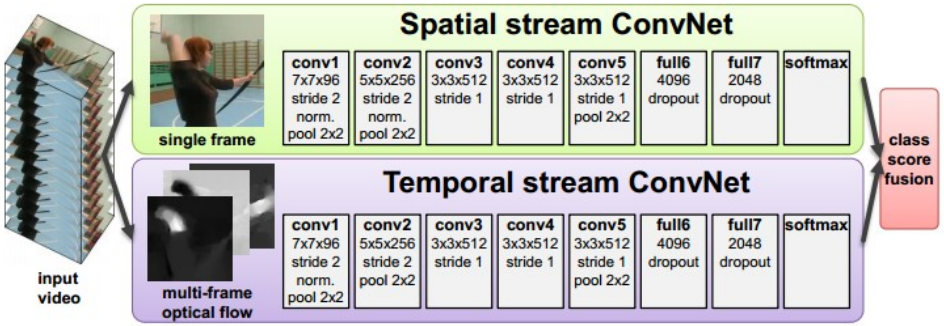
(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Fast-forward to today: ConvNets are everywhere

[Taigman et al. 2014]





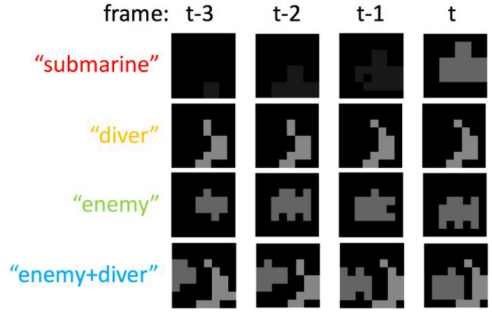
[Simonyan et al. 2014]

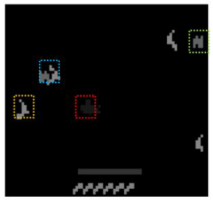
[Goodfellow 2014]

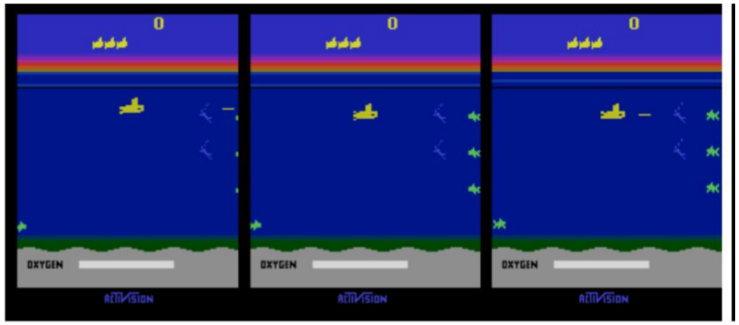
Fast-forward to today: ConvNets are everywhere

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]



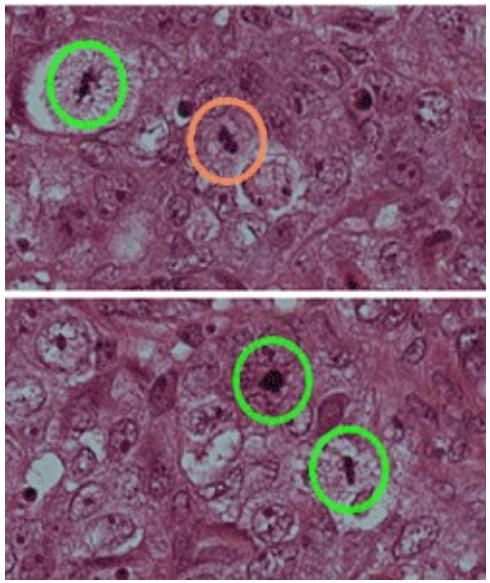




[Guo et al. 2014]

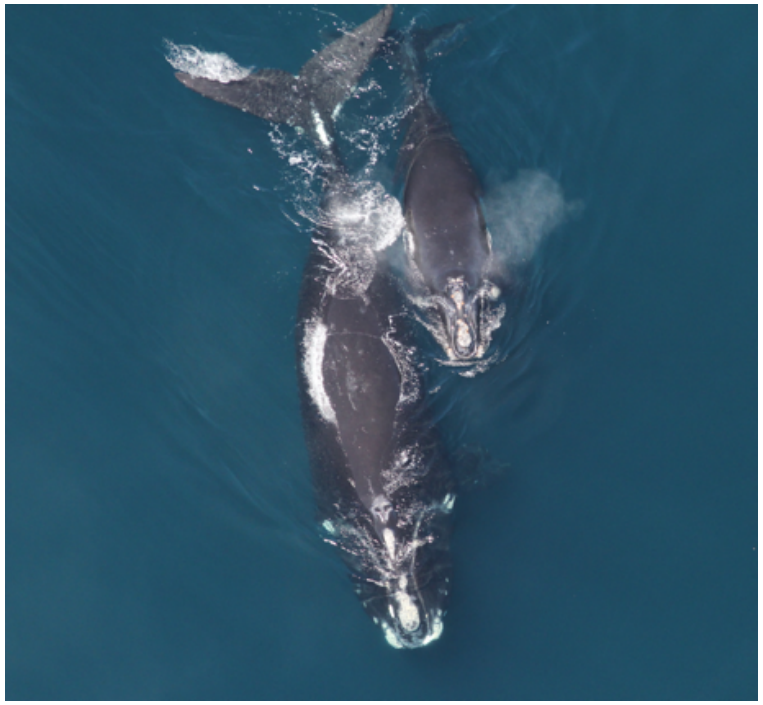
Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

Fast-forward to today: ConvNets are everywhere



[Ciresan et al. 2013]

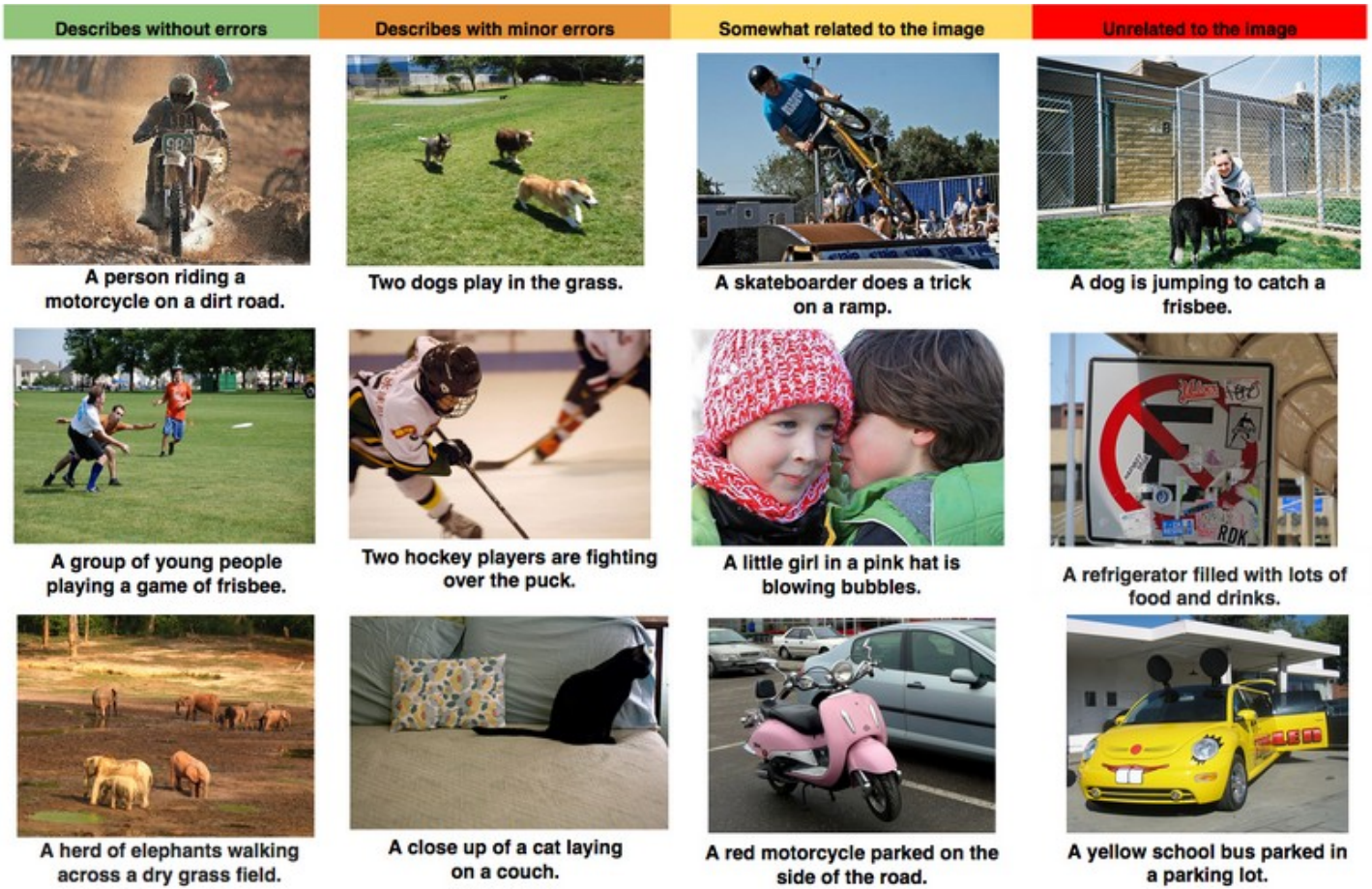
[Sermanet et al. 2011]
[Ciresan et al.]



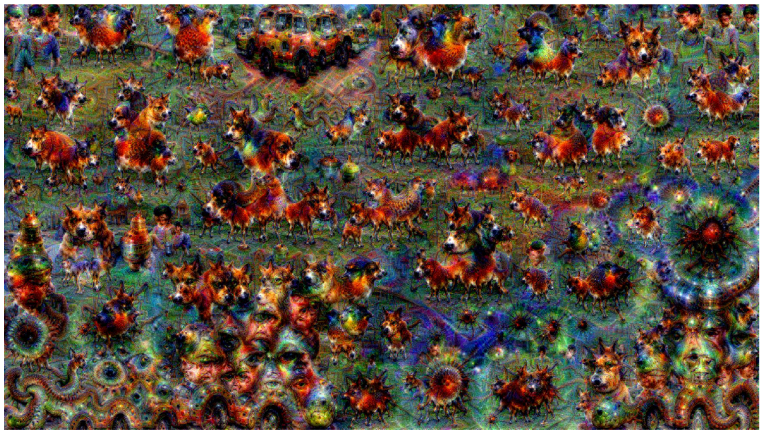
Whale recognition, Kaggle Challenge

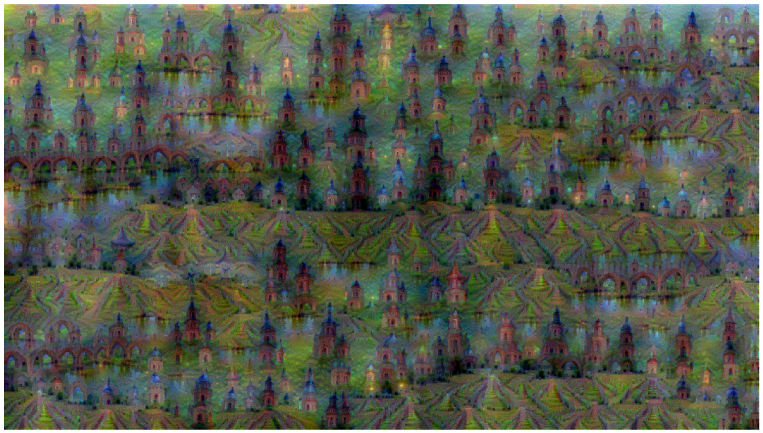
Mnih and Hinton, 2010

Image Captioning

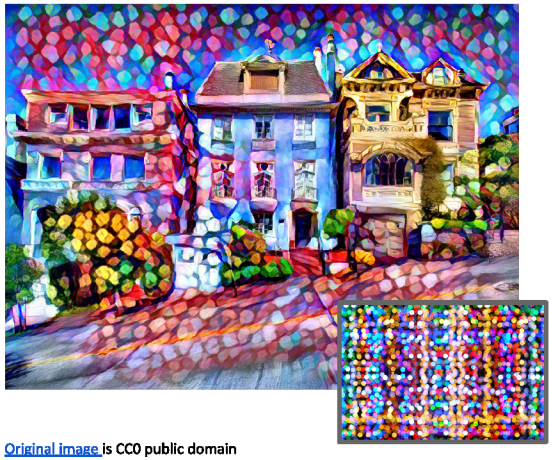


[Vinyals et al., 2015]





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

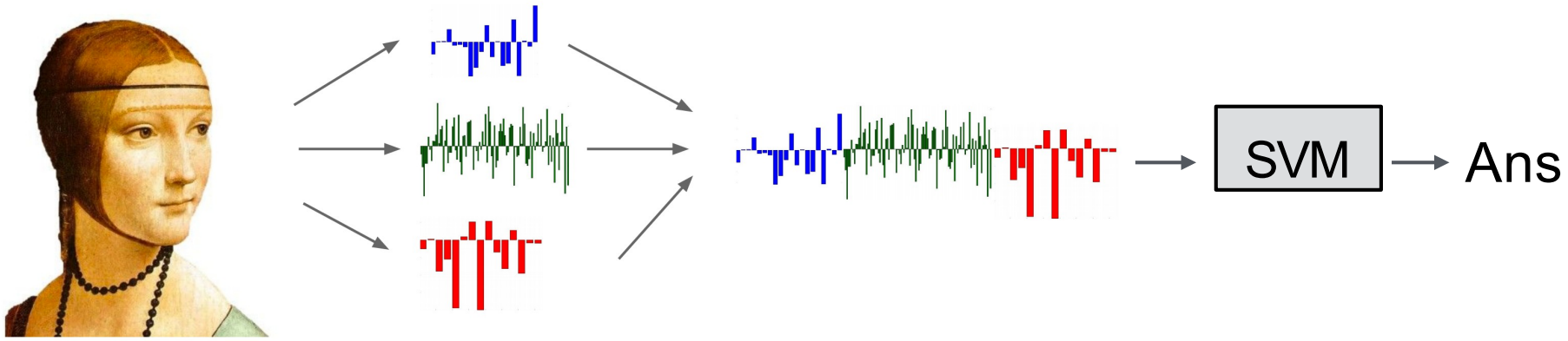


Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Summary

Recap: Life Before Deep Learning



Input Pixels

Extract Hand-Crafted Features

Concatenate into a vector ***x***

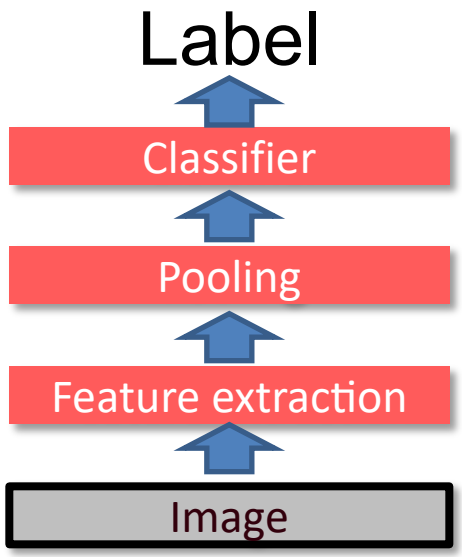
Linear Classifier

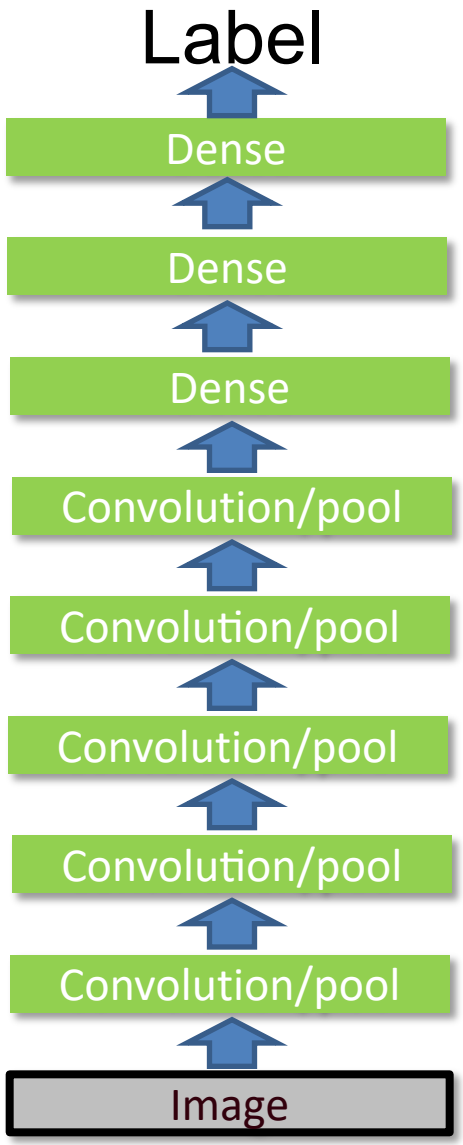
Figure: Karpathy 2016

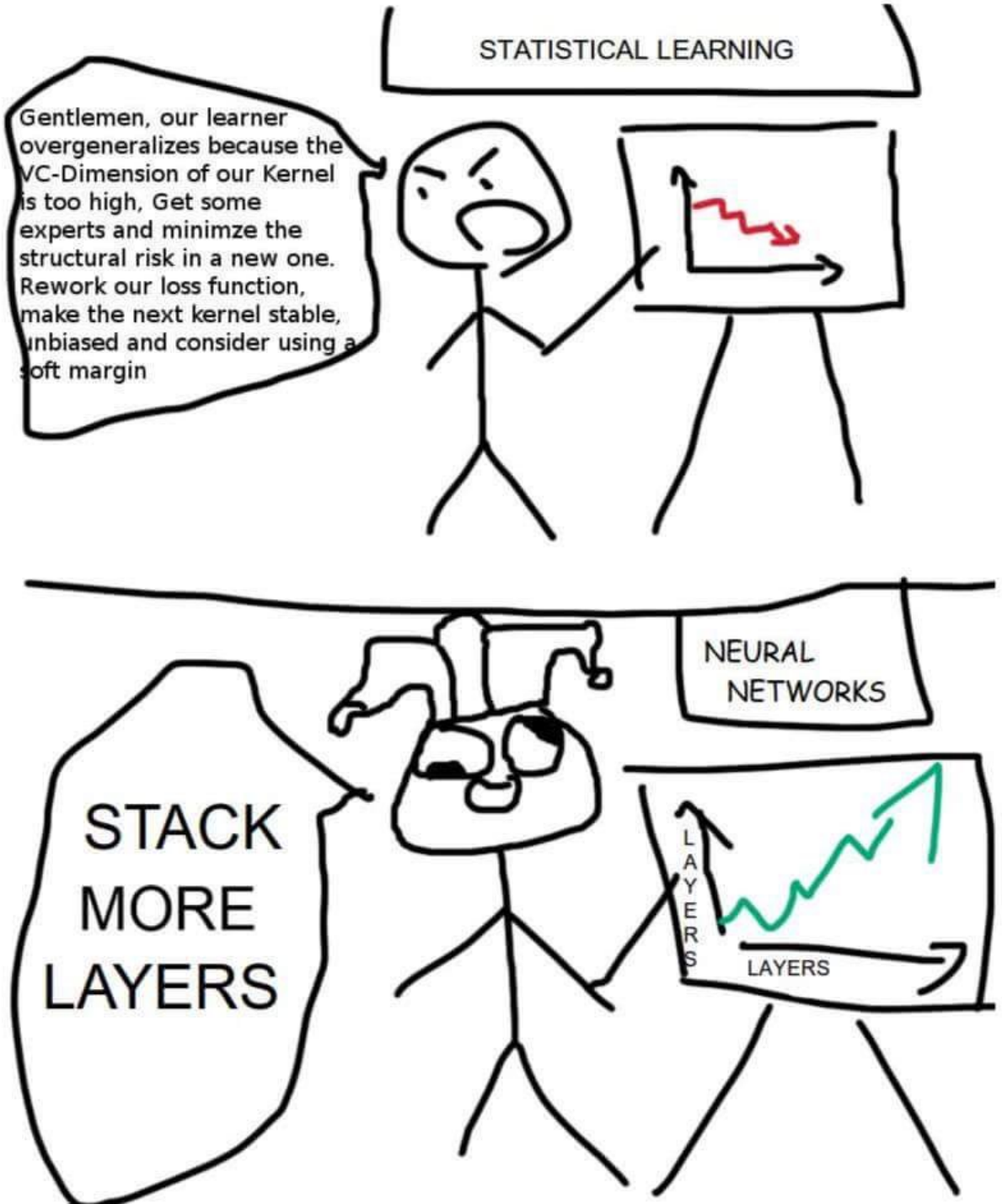
Why deep learning is powerful?

Convolutional filters are trained in a supervised manner by back-propagating classification error

Filters are learned from data instead of hand-crafted!






STATISTICAL LEARNING
Gentlemen, our learner overgeneralizes because the VC-Dimension of our Kernel is too high, Get some experts and minimze the structural risk in a new one. Rework our loss function, make the next kernel stable, unbiased and consider using a soft margin
NEURAL NETWORKS
STACK MORE LAYERS
LAYERS
LAYERS

Deep learning library

- TensorFlow
  - Research + Production

- PyTorch
  - Research

- Caffe2
  - Production

Resources

- http://deeplearning.net/
  - Hub to many other deep learning resources
- https://github.com/ChristosChristofidis/awesome-deep-learning
  - A resource collection deep learning
- https://github.com/kjw0612/awesome-deep-vision
  - A resource collection deep learning for computer vision
- http://cs231n.stanford.edu/syllabus.html
  - Nice course on CNN for visual recognition

Things to remember

- Supervised learning
  - Linear classifier, softmax, cross-entropy loss
- Neural network
  - Linear functions chained together and separated with non-linear activation functions
- Convolutional neural network (CNN)
  - Neural network with local connectivity and weight sharing
  - Convolution, nonlinearity, max pooling
- Training CNN
  - Back propagation, data split, dropout, data augmentation

2019 Turing Awards

Yann LeCun

Geoffrey Hinton

Yoshua