UC San Diego

Lecture 2:
Deep Learning Basics

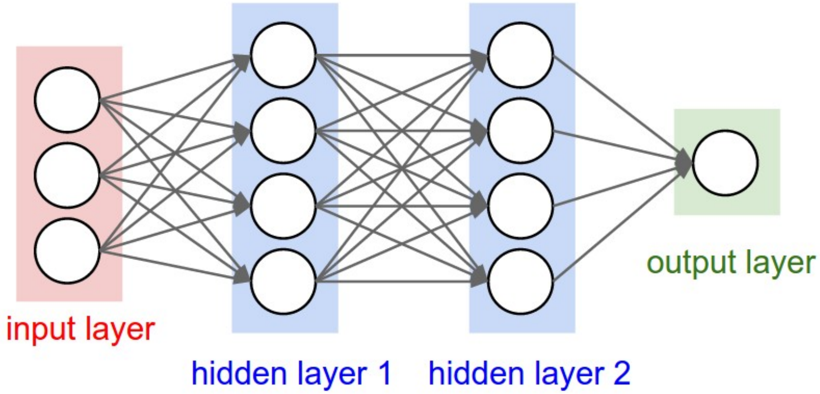
Instructor: Hao Su

Jan 11, 2018

Agenda

- Motivation of Building Deeper Networks
- Ideas in Deep Net Architectures
- Deep Learning Practice (Vignesh Gokul)

Neural Network: A Compositional Function



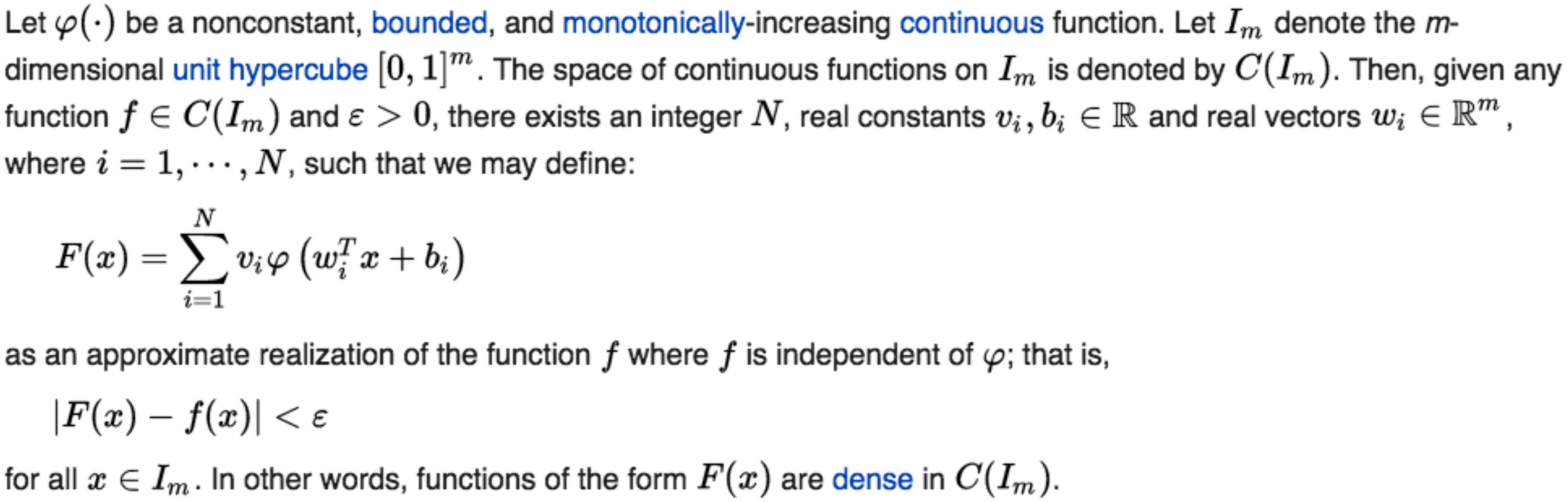
Model: Multi-Layer Perceptron (MLP) $y' = W_3 f(W_2 f(W_1 x + b_1) + b_2) + b_3)$

Loss function: L2 loss $l(y, y') = (y - y')^2$

Optimization: Gradient descent $W = W - \eta \frac{\partial L}{\partial W}$

Universal Approximation Theorem

A three-layer network approximates any continuous function



Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any function $f \in C(I_m)$ and $\varepsilon > 0$, there exists an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$, where $i = 1, \cdots, N$, such that we may define:

$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

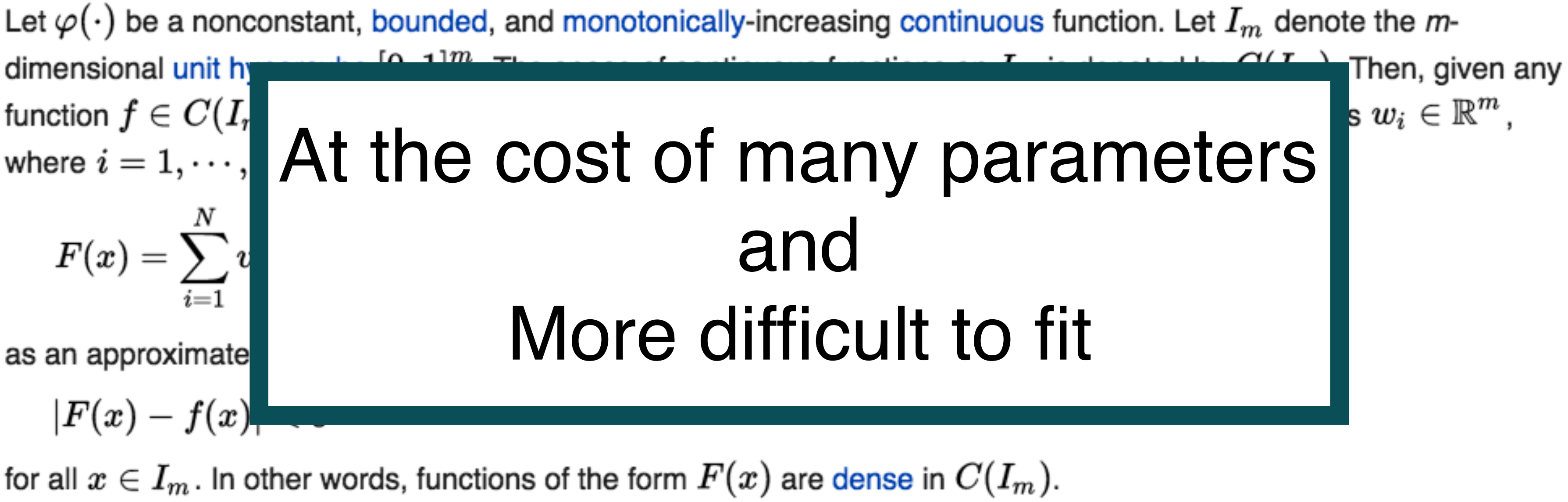
as an approximate realization of the function f where f is independent of φ; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

Universal Approximation Theorem

A three-layer network approximates any continuous function



Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any function $f \in C(I_{m}$ … s $w_i \in \mathbb{R}^m$, where $i = 1, \cdots,$ …

$$F(x) = \sum_{i=1}^{N} v$$

as an approximate …

$$|F(x) - f(x)|$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

At the cost of many parameters
and
More difficult to fit

A Principle in Learning Algorithm Design

- Overfitting is correlated with the complexity of learning model

In exponential family, Bayesian Information Criterion (BIC) for Model Selection

Smaller is better

$$-2 \cdot \ln p(x \mid M) \approx \mathrm{BIC} = -2 \cdot \ln \hat{L} + k \cdot \ln(n) + O(1)$$

- $\hat{L}$ = the maximized value of the likelihood function of the model , i.e. , where $\hat{\theta}$ are the parameter values that maximize the likelihood function;
- x = the observed data;
- n = the number of data points in x, the number of observations, or equivalently, the sample size;
- k = the number of parameters estimated by the model.

A Principle in Learning Algorithm Design

- Overfitting is correlated with the complexity of learning model

In exponential family, Bayesian Information Criterion (BIC) for Model Selection

Smaller is better

$$-2 \cdot \ln p(x \mid M) \approx \mathrm{BIC} = \boxed{-2 \cdot \ln \hat{L}} + k \cdot \ln(n) + O(1)$$

Complex model decreases this term

- $\hat{L}$ = the maximized value of the likelihood function of the model , i.e. , where $\hat{\theta}$ are the parameter values that maximize the likelihood function;
- x = the observed data;
- n = the number of data points in x, the number of observations, or equivalently, the sample size;
- k = the number of parameters estimated by the model.

A Principle in Learning Algorithm Design

- Overfitting is correlated with the complexity of learning model

In exponential family, Bayesian Information Criterion (BIC) for Model Selection

Smaller is better

Complex model increases this term

$$-2\cdot\ln p(x\mid M)\approx \mathrm{BIC}=\boxed{-2\cdot\ln\hat{L}}+\boxed{k}\cdot\ln(n)+O(1)$$

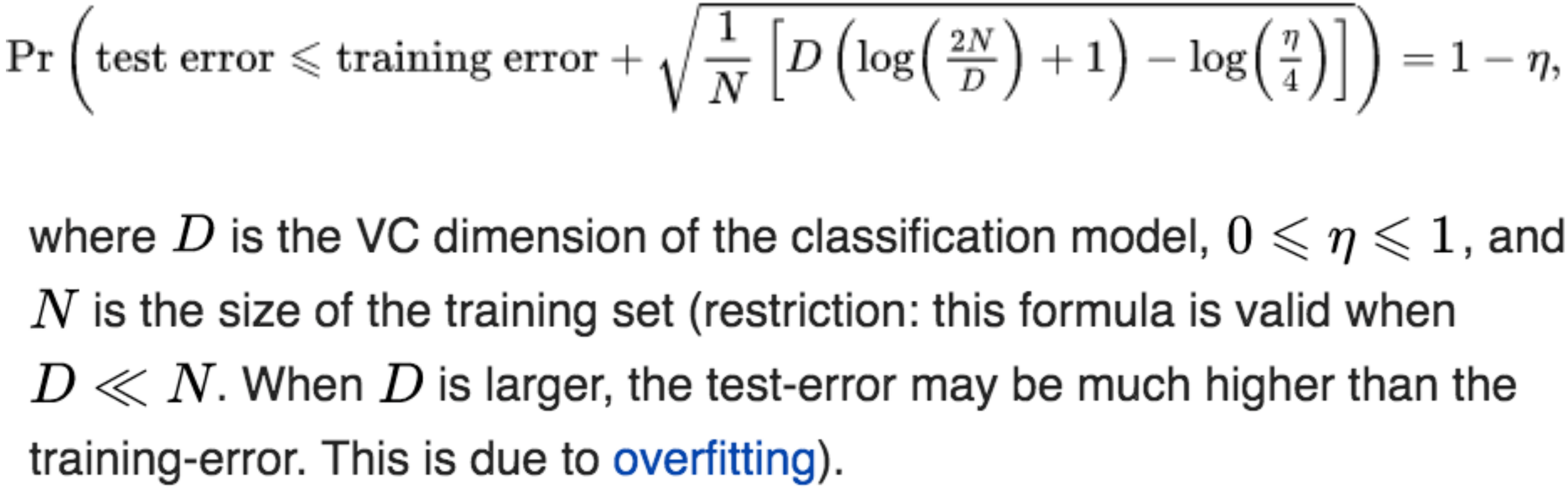
Complex model decreases this term

- $\hat{L}$ = the maximized value of the likelihood function of the model , i.e. , where $\hat{\theta}$ are the parameter values that maximize the likelihood function;
- x = the observed data;
- n = the number of data points in x, the number of observations, or equivalently, the sample size;
- k = the number of parameters estimated by the model.

A Principle in Learning Algorithm Design

- Overfitting is correlated with the complexity of learning model

Probably Approximately Correct (PAC) theory



$$\Pr\left(\text{test error} \leqslant \text{training error} + \sqrt{\frac{1}{N}\left[D\left(\log\left(\tfrac{2N}{D}\right)+1\right)-\log\left(\tfrac{\eta}{4}\right)\right]}\right) = 1-\eta,$$

where D is the VC dimension of the classification model, $0 \leqslant \eta \leqslant 1$, and N is the size of the training set (restriction: this formula is valid when $D \ll N$. When D is larger, the test-error may be much higher than the training-error. This is due to overfitting).

[from Wikipedia]

Occam's Razor Principle

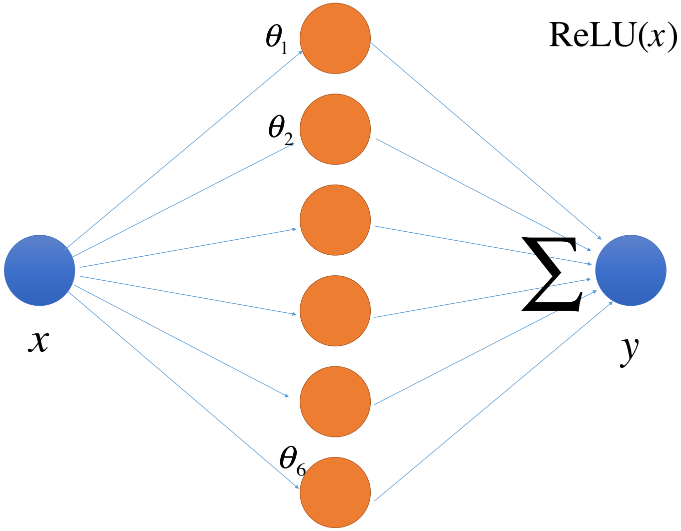
Entia non sunt multiplicanda praeter necessitatem.

William of Ockham, 14th century

Suppose there exist two explanations for an occurrence. In this case the simpler one is usually better.

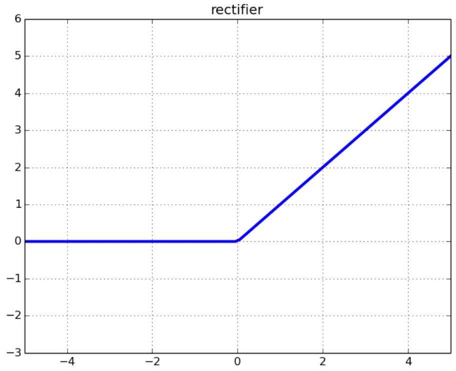
[from Wikipedia]

The Intuition behind Pushing Deep

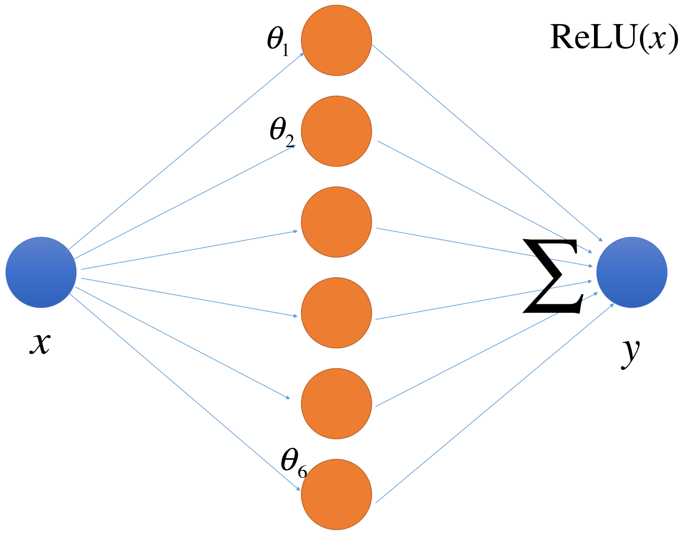


$$a_i = \mathrm{ReLU}(x - \theta_i)$$

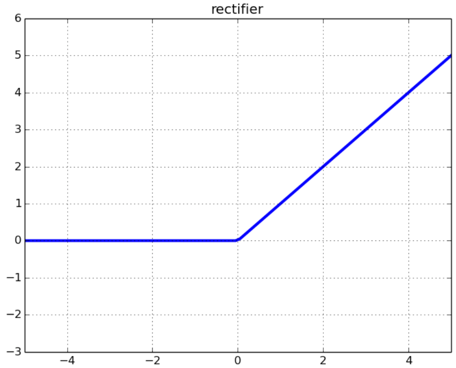
$$\mathrm{ReLU}(x) = \begin{cases} 0 \; if \;\; x < 0 \\ x \; if \;\; x >= 0 \end{cases}$$



The Intuition behind Pushing Deep

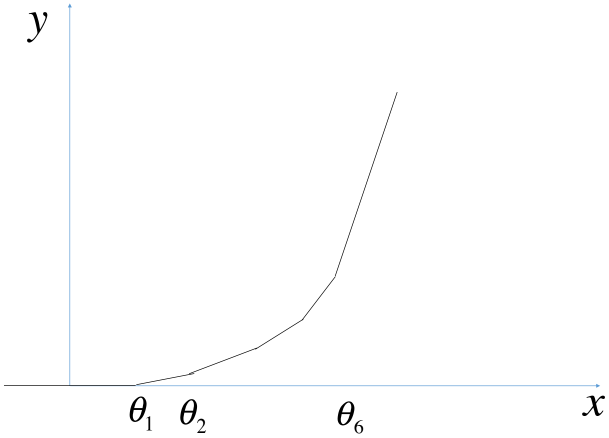


$$\mathrm{ReLU}(x) = \begin{cases} 0 \ \mathit{if} \ \ x < 0 \\ x \ \mathit{if} \ \ x >= 0 \end{cases}$$



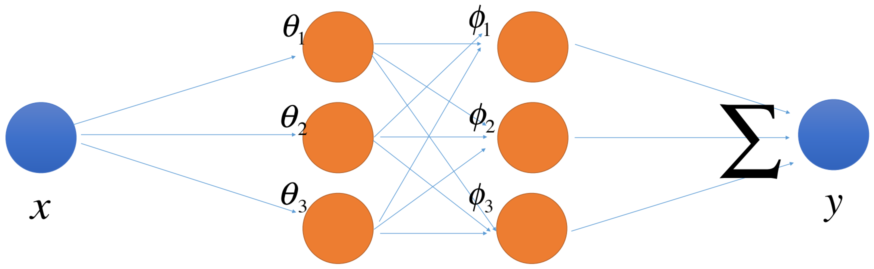
$$a_i = \mathrm{ReLU}(x - \theta_i)$$

$$y = \sum_i \mathrm{ReLU(x-}\theta_i)$$



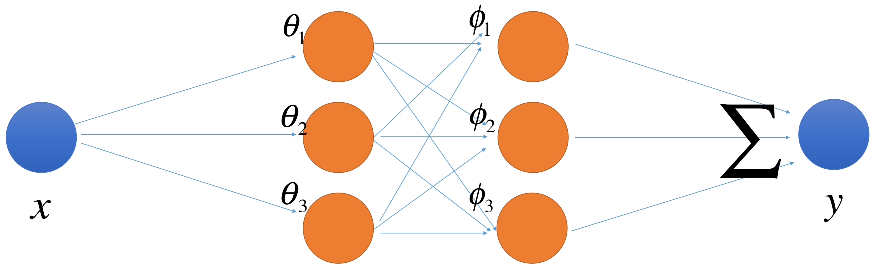
piece-wise linear, 6 knots

The Intuition behind Pushing Deep



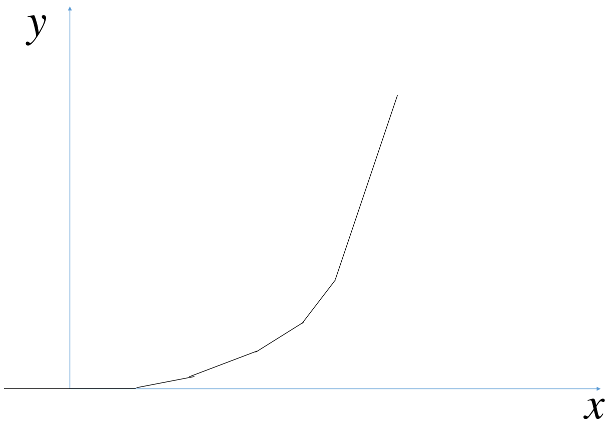
$a_{1,i} = \mathrm{ReLU}(x-\theta_i)$ $a_{2,i} = \mathrm{ReLU}(x-\phi_i)$ y

The Intuition behind Pushing Deep



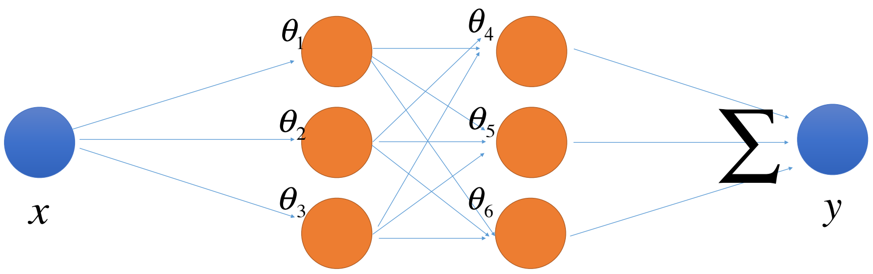
$$a_{1,i} = \mathrm{ReLU}(x - \theta_i) \quad a_{2,i} = \mathrm{ReLU}(x - \phi_i)$$

$$y = \sum_{1\le i\le 3} \mathrm{ReLU}([\sum_{1\le j\le 3} \mathrm{ReLU}(\mathrm{x}\text{-}\theta_j)]\text{-}\phi_i)$$



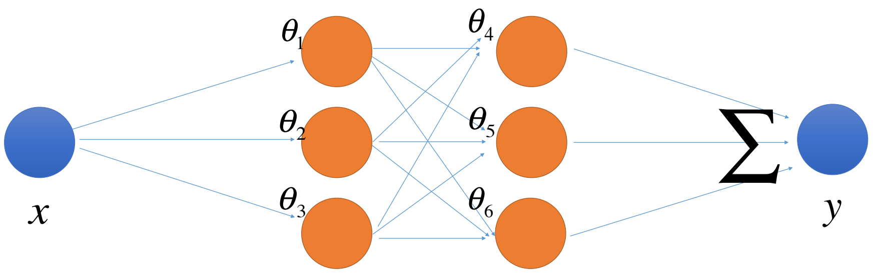
piece-wise linear, can have **9 knots**!

The Intuition behind Pushing Deep



Interpretation I: With the same number of parameters, create combinatorial data flow

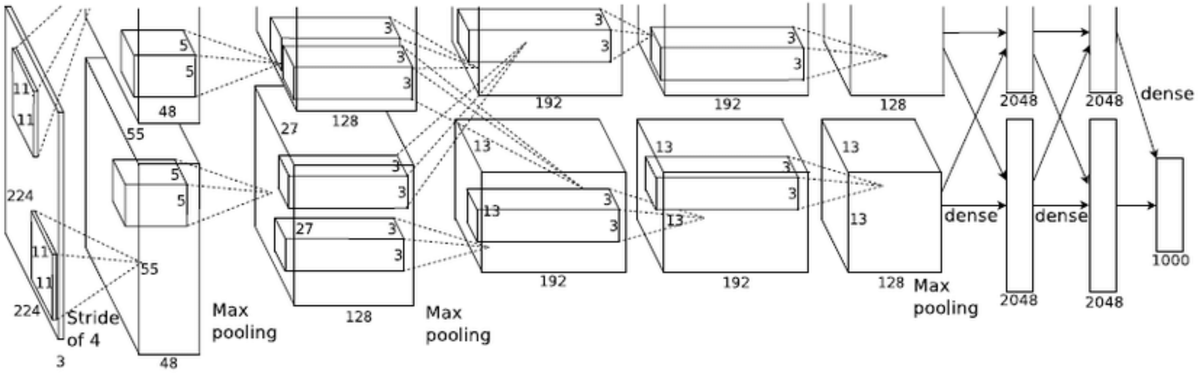
The Intuition behind Pushing Deep



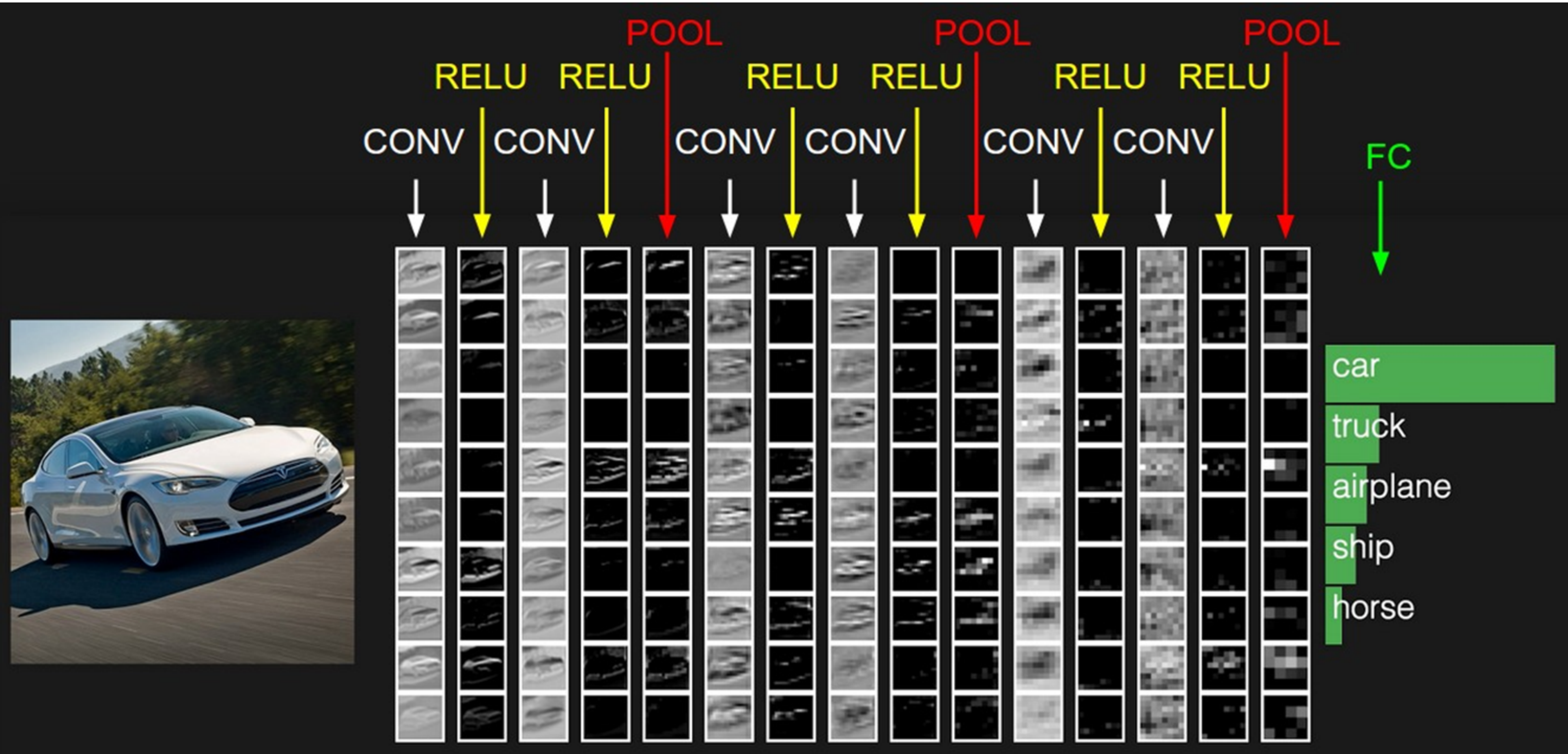
Interpretation I: With the same number of parameters,
create combinatorial data flow

Interpretation II: Abstract data progressively

Alexnet



11
11
224
224
3
Stride of 4
11
11
5
5
55
55
48
48
Max pooling
5
5
3
3
27
27
128
128
Max pooling
3
3
13
13
192
192
3
3
13
13
192
192
13
13
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000



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

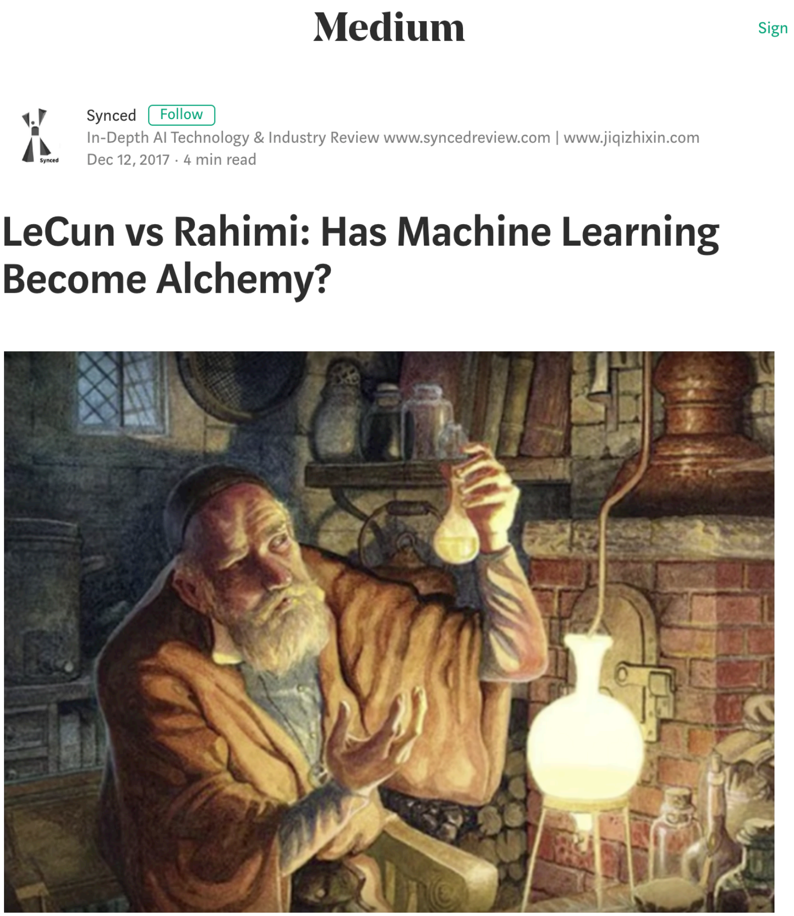
NIPS 2017 Debate

“Machine Learning is the new electricity.”

- Andrew Ng

“Machine Learning has become alchemy.”

- Ali Rahimi (at NIPS 2017)



Medium

Synced

In-Depth AI Technology & Industry Review www.syncedreview.com | www.jiqizhixin.com

Dec 12, 2017 · 4 min read

LeCun vs Rahimi: Has Machine Learning Become Alchemy?

IDEAS IN DEEP NET ARCHITECTURES

What people think I am doing when I “build a deep learning model”

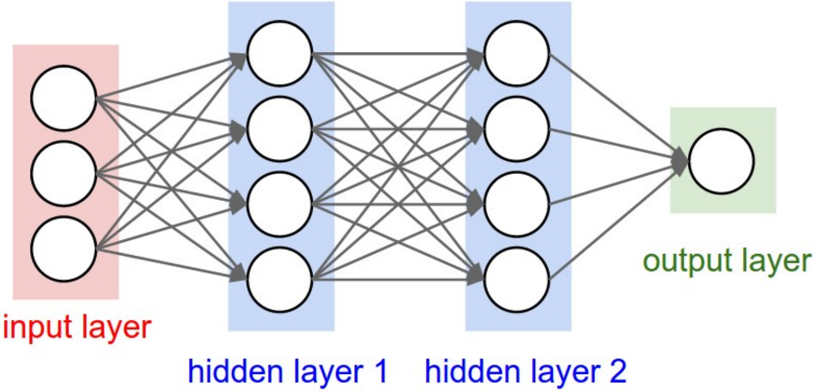
What I actually do...

Contents

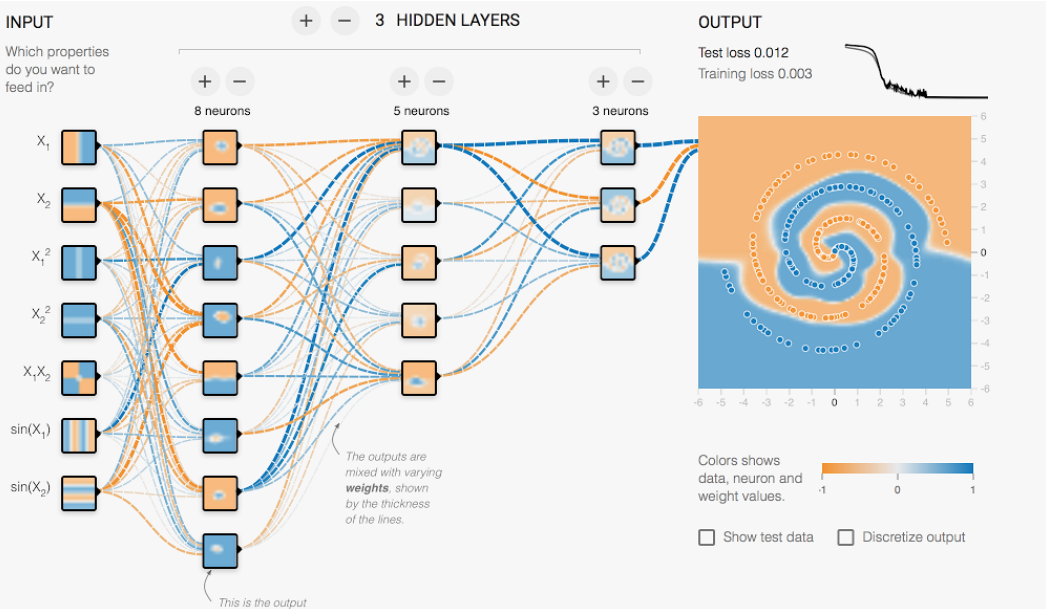
- **Building blocks:** fully connected, ReLU, conv, pooling,
- **Classic architectures:** MLP, LeNet, AlexNet, VGG, ResNet

Multi-Layer Perceptron

Fully Connected



http://playground.tensorflow.org/



Which properties do you want to feed in?

Test loss 0.012

Training loss 0.003

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

This is the output

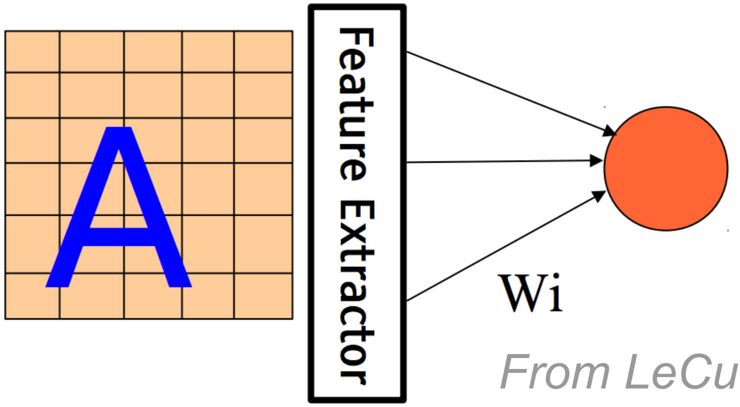
Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

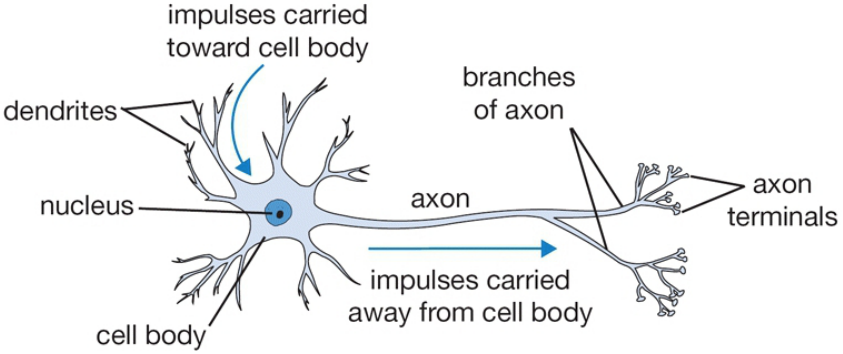
Fully Connected

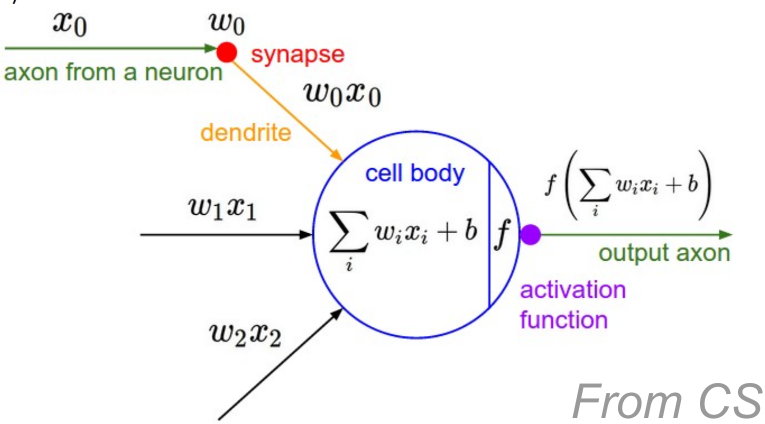
- The first learning machine: the **Perceptron** Built at Cornell in 1960
- The Perceptron was a (binary) linear classifier on top of a simple feature extractor

$$y = sign\left(\sum_{i=1}^{N} W_i F_i(X) + b\right)$$



From LeCun's Slides

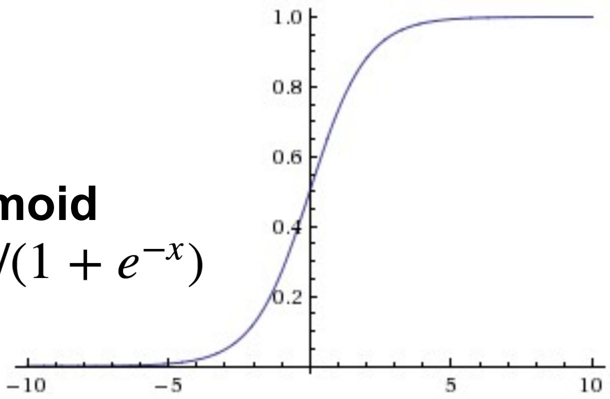

impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body



From CS231N

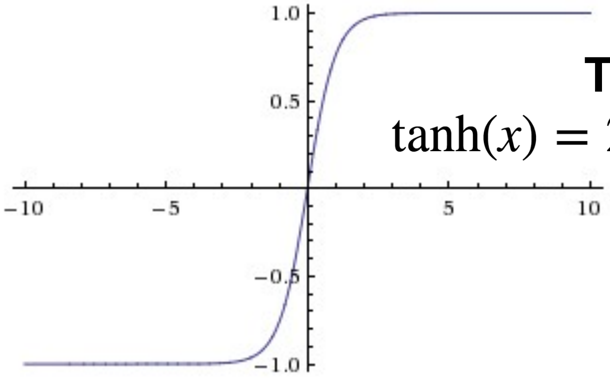
Sigmoid

$\sigma(x) = 1/(1 + e^{-x})$



Tanh

$\tanh(x) = 2\sigma(2x) - 1$

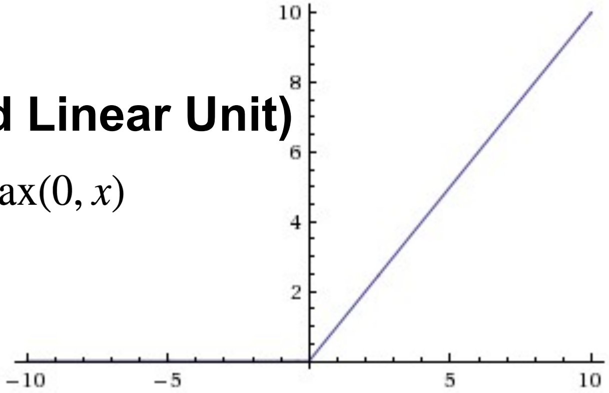


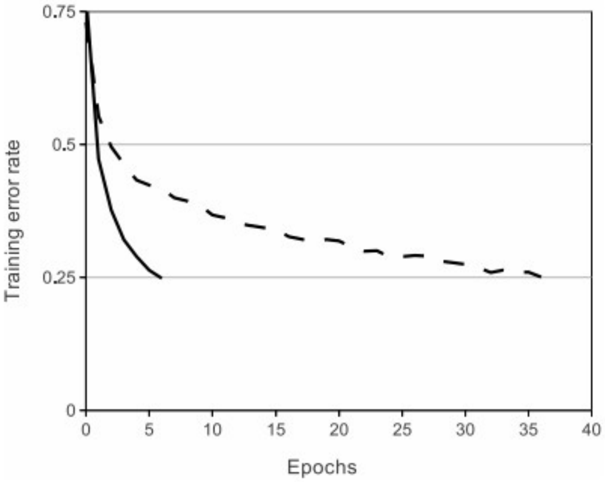
Major drawbacks: Sigmoids saturate and kill gradients

From CS231N

ReLU (Rectified Linear Unit)

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





- + **Cheaper (linear) compared with Sigmoids (exp)**
- + **No gradient saturation, faster in convergence**
- - **“Dead” neurons if learning rate set too high**

A plot from Krizhevsky et al. paper indicating **the 6x improvement in convergence** with the ReLU unit compared to the tanh unit.

Other Non-linear Op:

Leaky ReLU, $f(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x >= 0)(x)$

MaxOut $\max(w_1^T x + b_1, w_2^T x + b_2)$

From CS231N

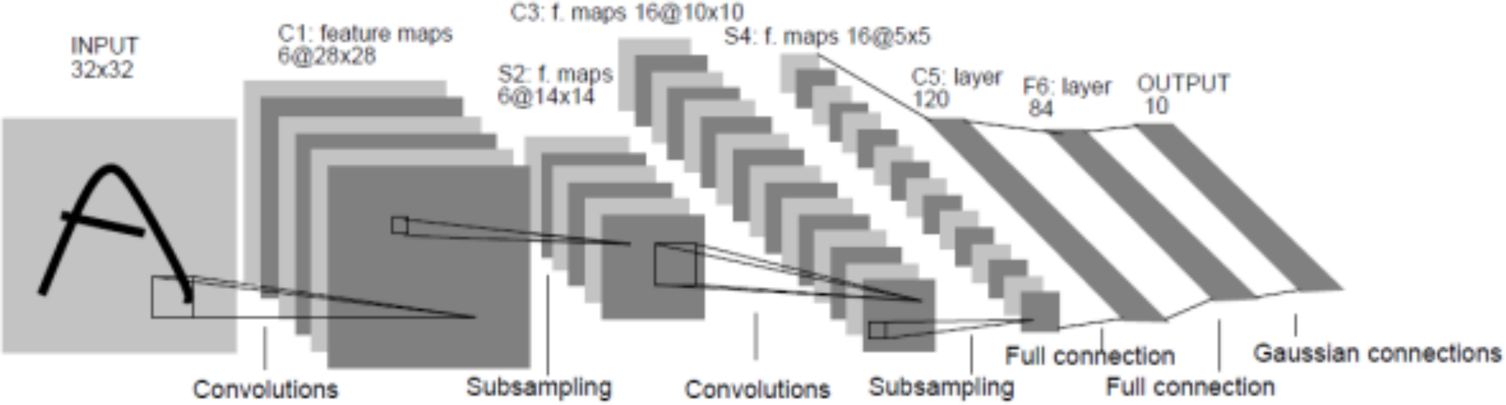
Convolutional Neural Network : LeNet (1998 by LeCun et al.)

Fully Connected

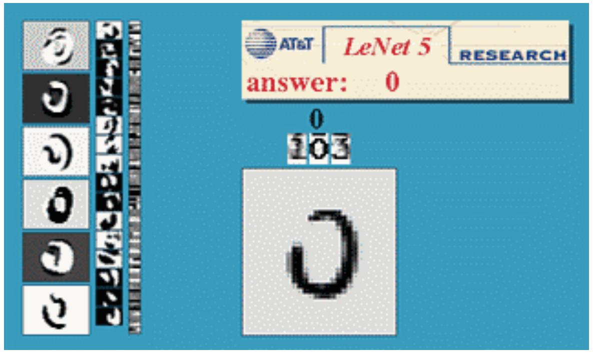
Convolution

Non-linear Op

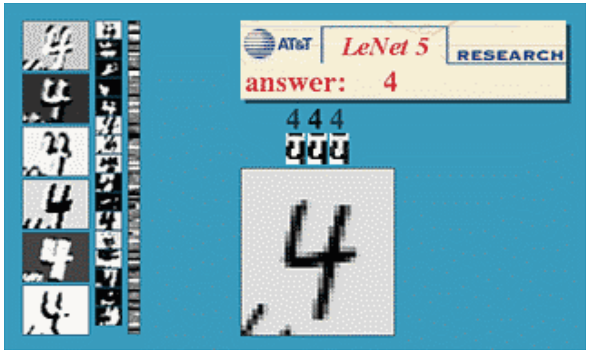
Pooling



(pooling)



answer: 0

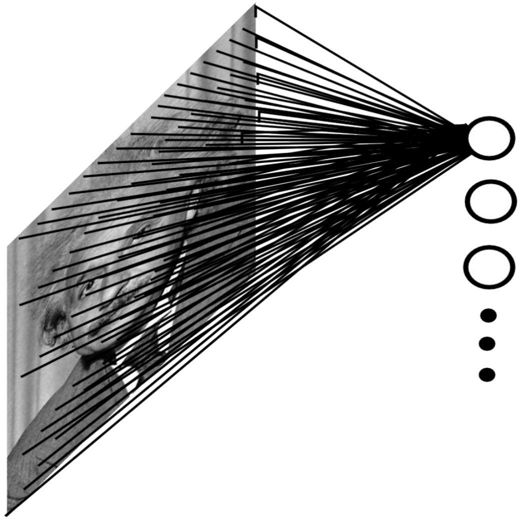


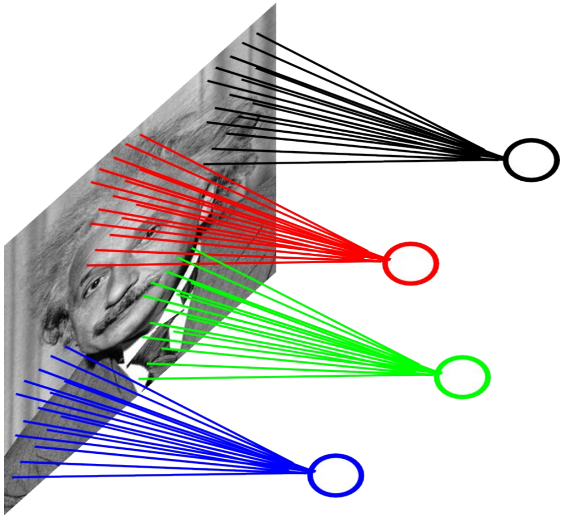
answer: 4

One of the first successful applications of CNN.

Fully Connected NN in high dimension

- Example: 200x200 image
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million params
  - Local connections capture local dependencies

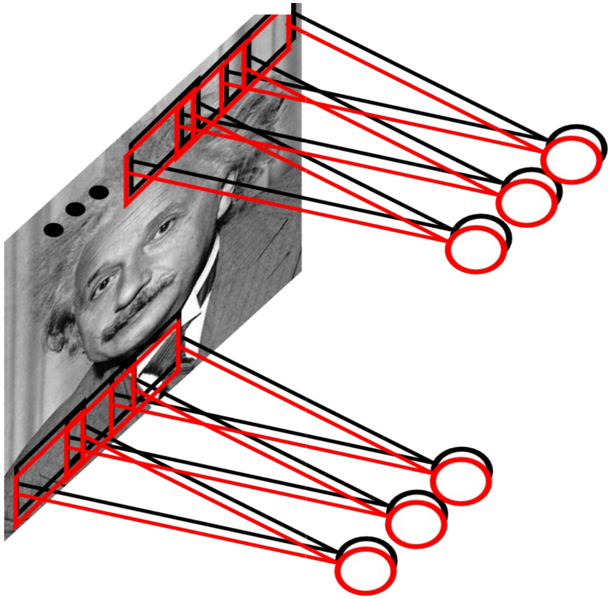




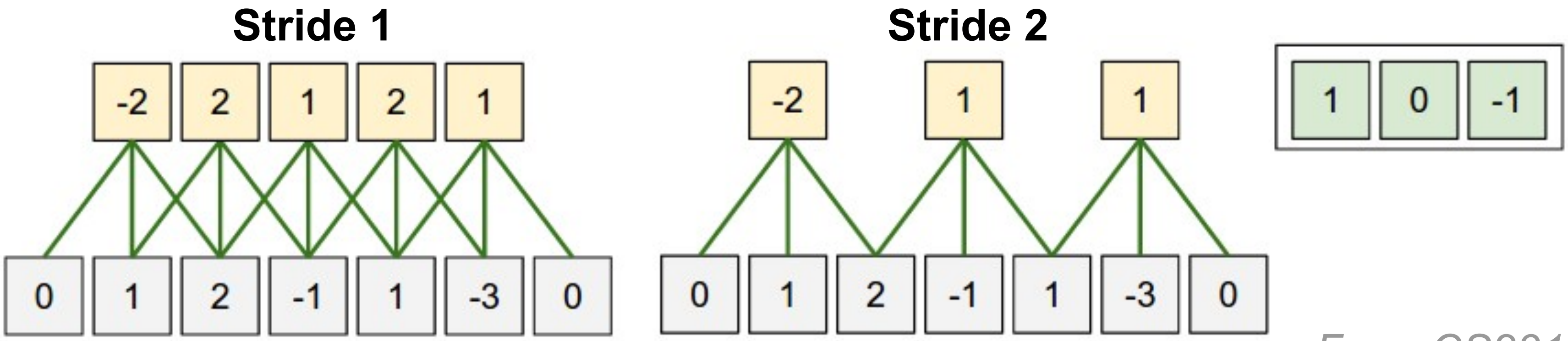
Slide from LeCun

Shared Weights & Convolutions: Exploiting Stationarity

- Example: 200x200 image
  - 400,000 hidden units with 10x10 fields = 1000 params
  - 10 feature maps of size 200x200, 10 filters of size 10x10



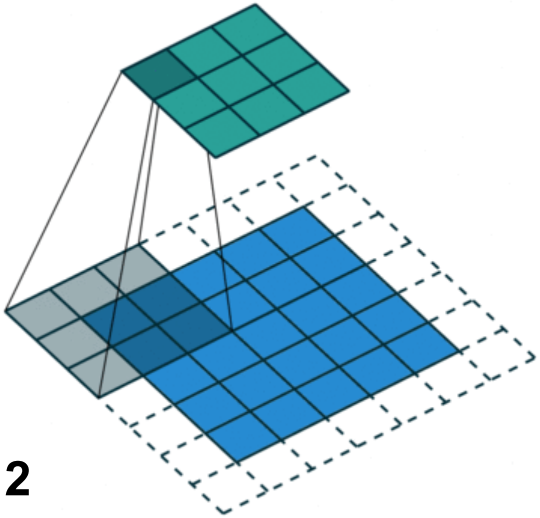
Convolution



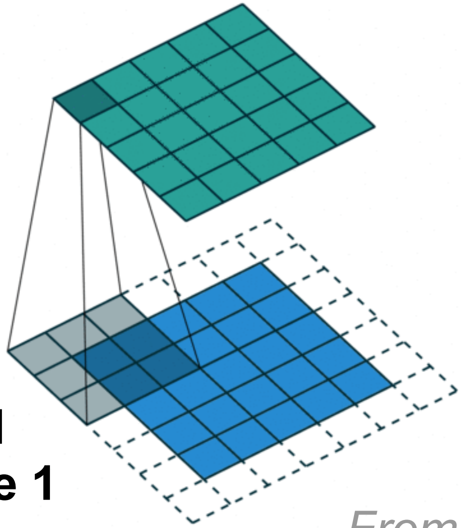
Stride 1

Stride 2

From CS231N



Pad 1
Stride 2

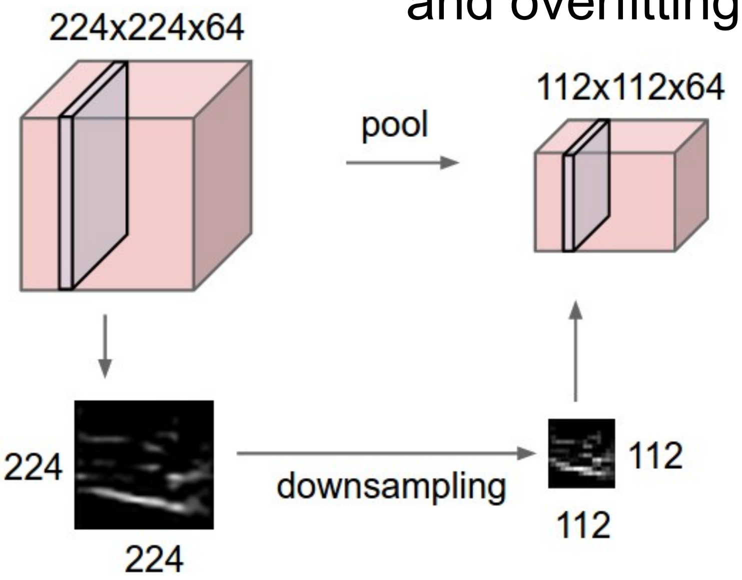


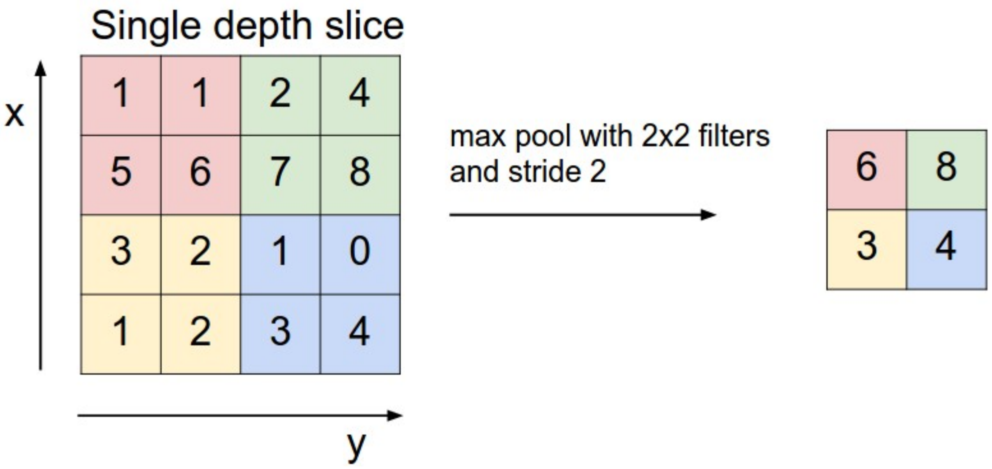
Pad 1
Stride 1

From vdumoulin/ conv_arithmetic

Pooling

Pooling layer (usually inserted in between conv layers) is used to reduce spatial size of the input, thus reduce number of parameters and overfitting.





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

6	8
3	4

Discarding pooling layers has been found to be important in training good generative models, such as variational autoencoders (VAEs) or generative adversarial networks (GANs).
It seems likely that future architectures will feature very few to no pooling layers.

From CS231N

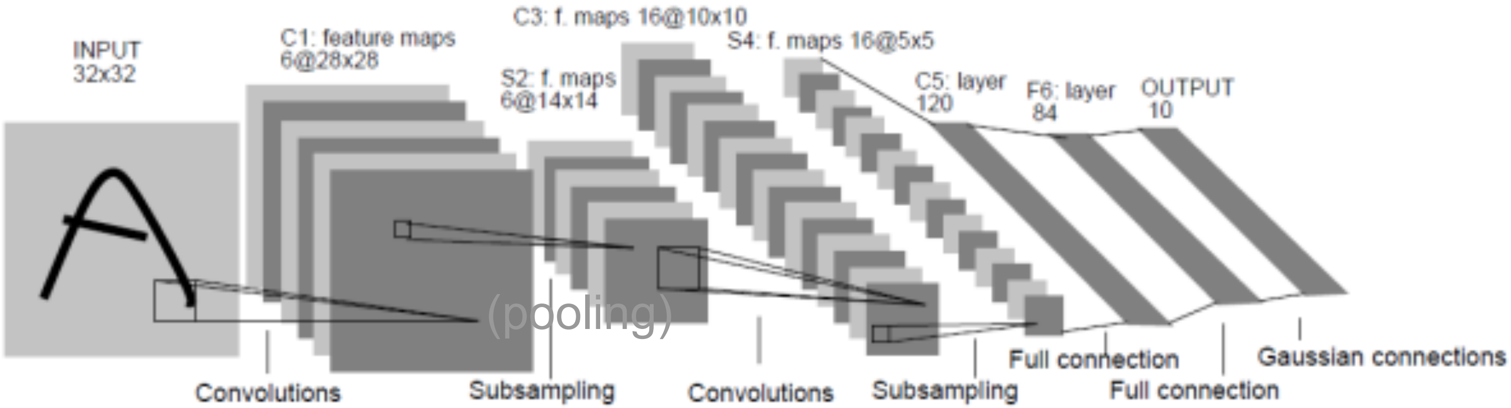
LeNet (1998 by LeCun et al.)

Fully Connected

Convolution

Non-linear Op

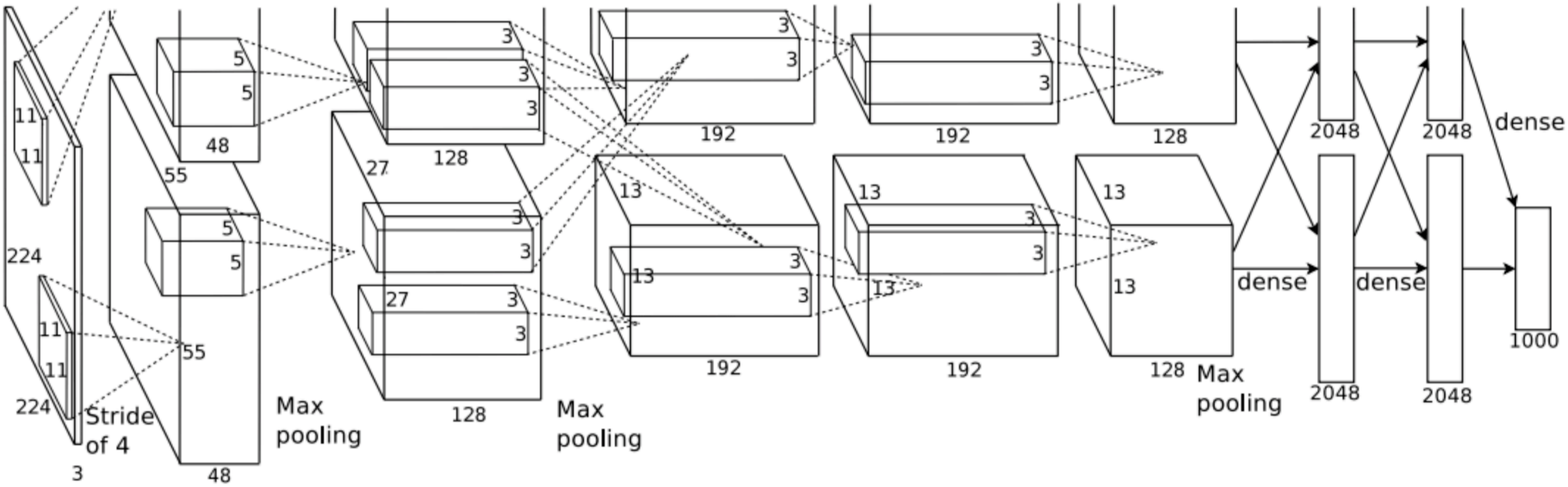
Pooling



AlexNet (2012 by Krizhevsky et al.)

[Donoho et al, STATS385]

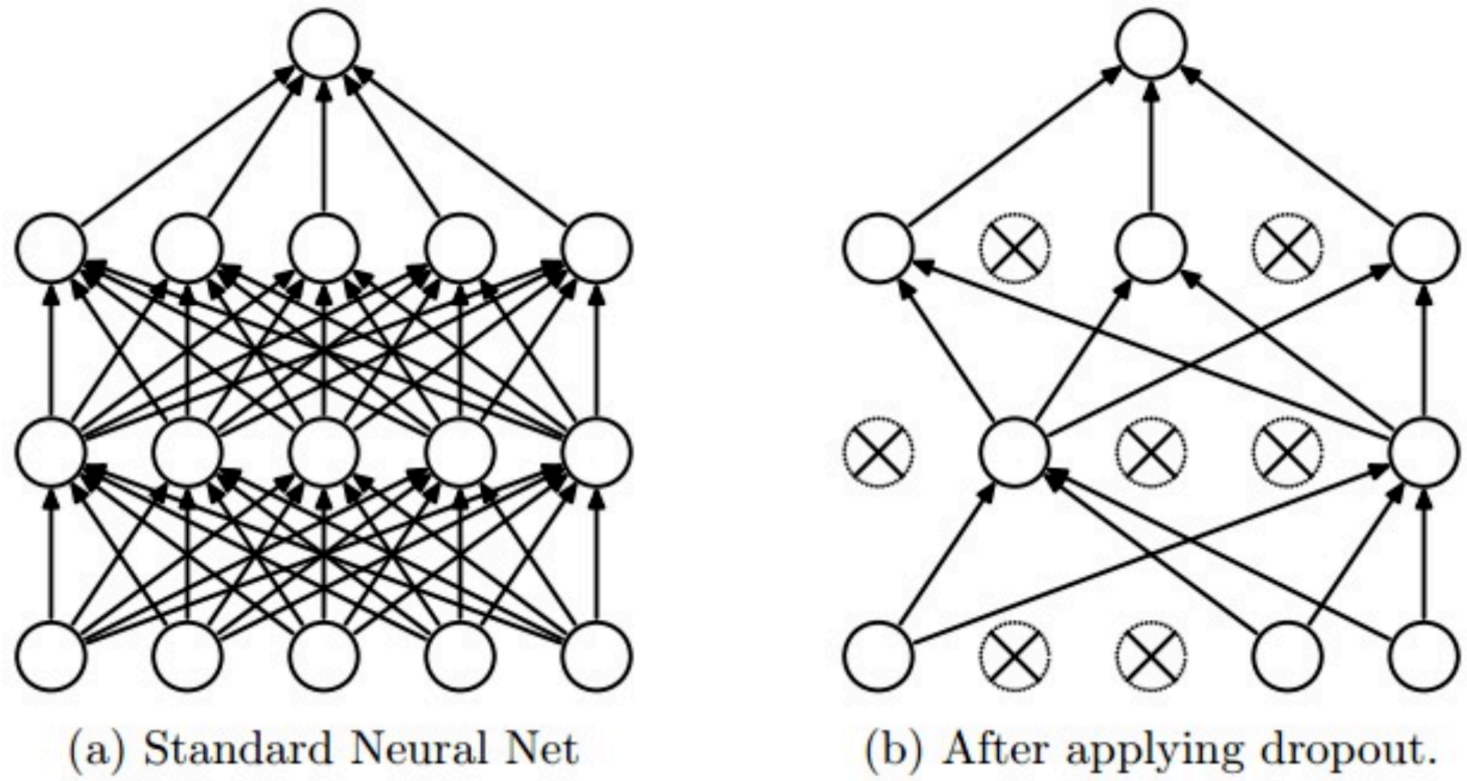
- 8 layers: first 5 convolutional, rest fully connected
- ReLU nonlinearity
- Local response normalization
- Max-pooling
- Dropout



Source: [Krizhevsky et al., 2012]

AlexNet (2012 by Krizhevsky et al.)

[Donoho et al, STATS385]



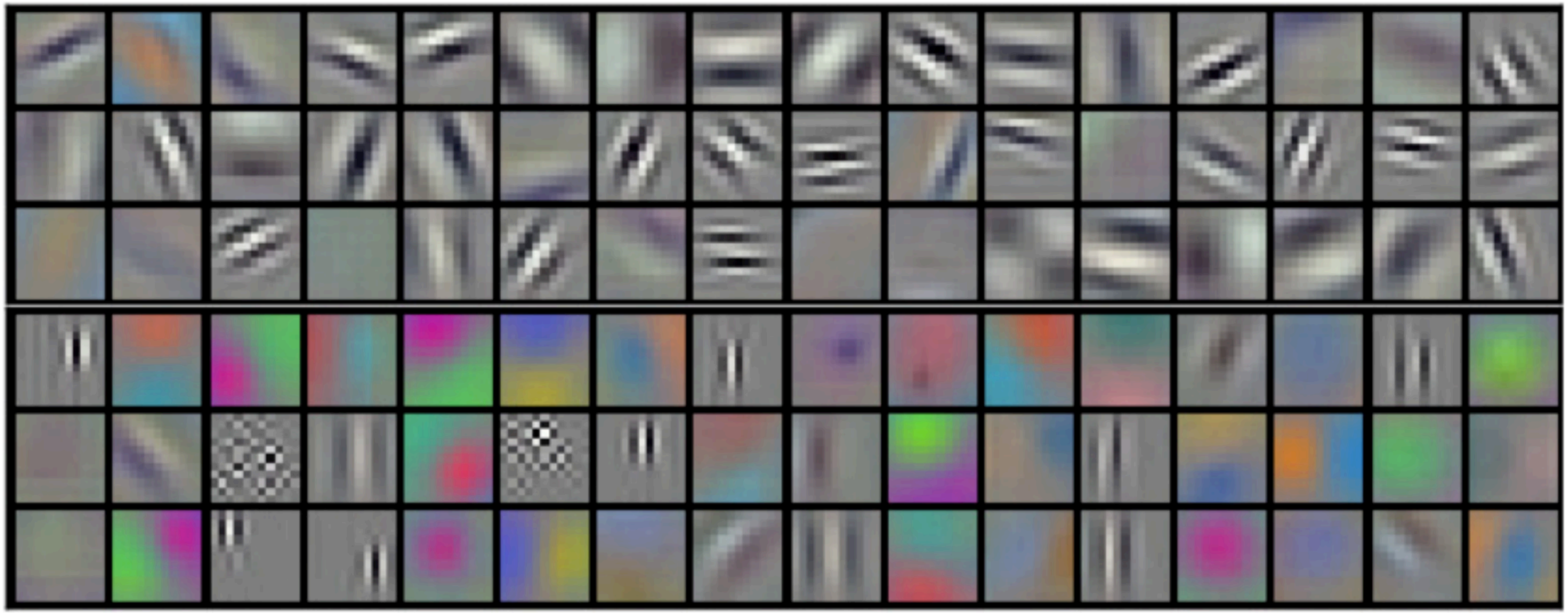
Source: [Srivastava et al., 2014]

- Zero every neuron with probability $1 - p$
- At test time, multiply every neuron by p

AlexNet (2012 by Krizhevsky et al.)

[Donoho et al, STATS385]

- Stochastic gradient descent
- Mini-batches
- Momentum
- Weight decay (ℓ_2 prior on the weights)



Filters trained in the first layer

Source: [Krizhevsky et al., 2012]

AlexNet (2012 by Krizhevsky et al.)

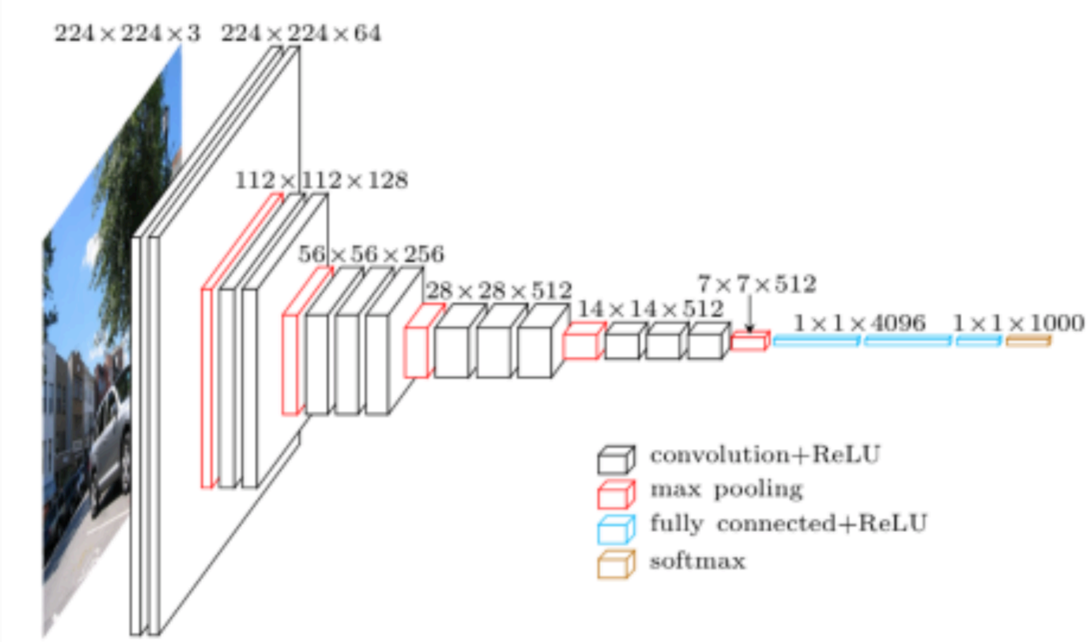
[Donoho et al, STATS385]

- The number of training examples is 1.2 million
- The number of parameters is 5-155 million
- How does the network manage to generalize?

VGG (2014 by Simonyan et al.)

[Donoho et al, STATS385]

- Deeper than AlexNet: 11-19 layers versus 8
- No local response normalization
- Number of filters multiplied by two every few layers
- Spatial extent of filters 3×3 in all layers
- Instead of 7×7 filters, use three layers of 3×3 filters
  - Gain intermediate nonlinearity
  - Impose a regularization on the 7×7 filters



ResNet (2016 by Kaiming He et al.)

[Donoho et al, STATS385]

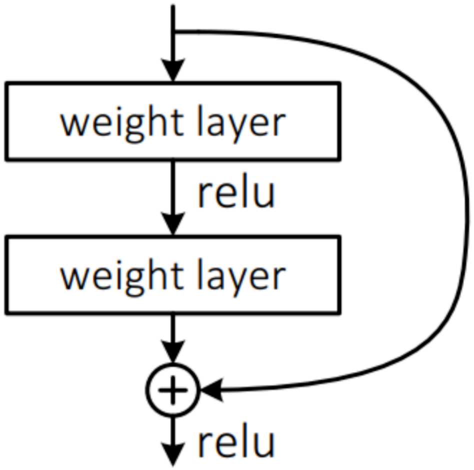
- Formally, deeper networks contain shallower ones (i.e. consider no-op layers)
- **Observation:** Deeper networks not always lower training error
- **Conclusion:** Optimization process can't successfully infer no-op

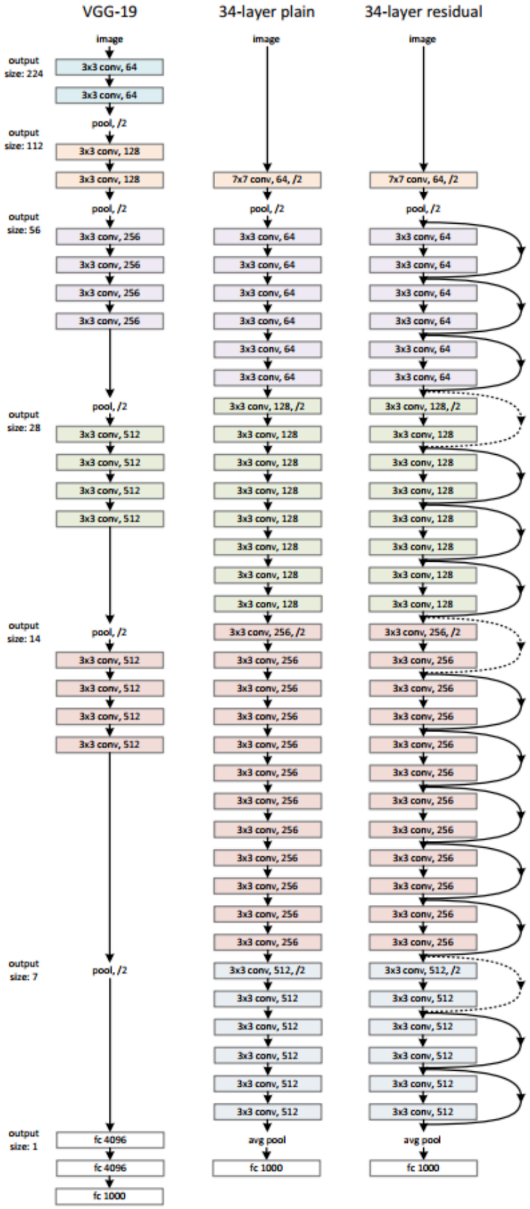
ResNet (2016 by Kaiming He et al.)

[Donoho et al, STATS385]

- Solves problem by adding skip connections
- Very deep: 152 layers
- No dropout

- Batch normalization





Source: Deep Residual Learning for Image Recognition

[Donoho et al, STATS385]

Algorithm 2 Batch normalization [Ioffe and Szegedy, 2015]

Input: Values of x over minibatch $x_1 \dots x_B$, where x is a certain channel in a certain feature vector

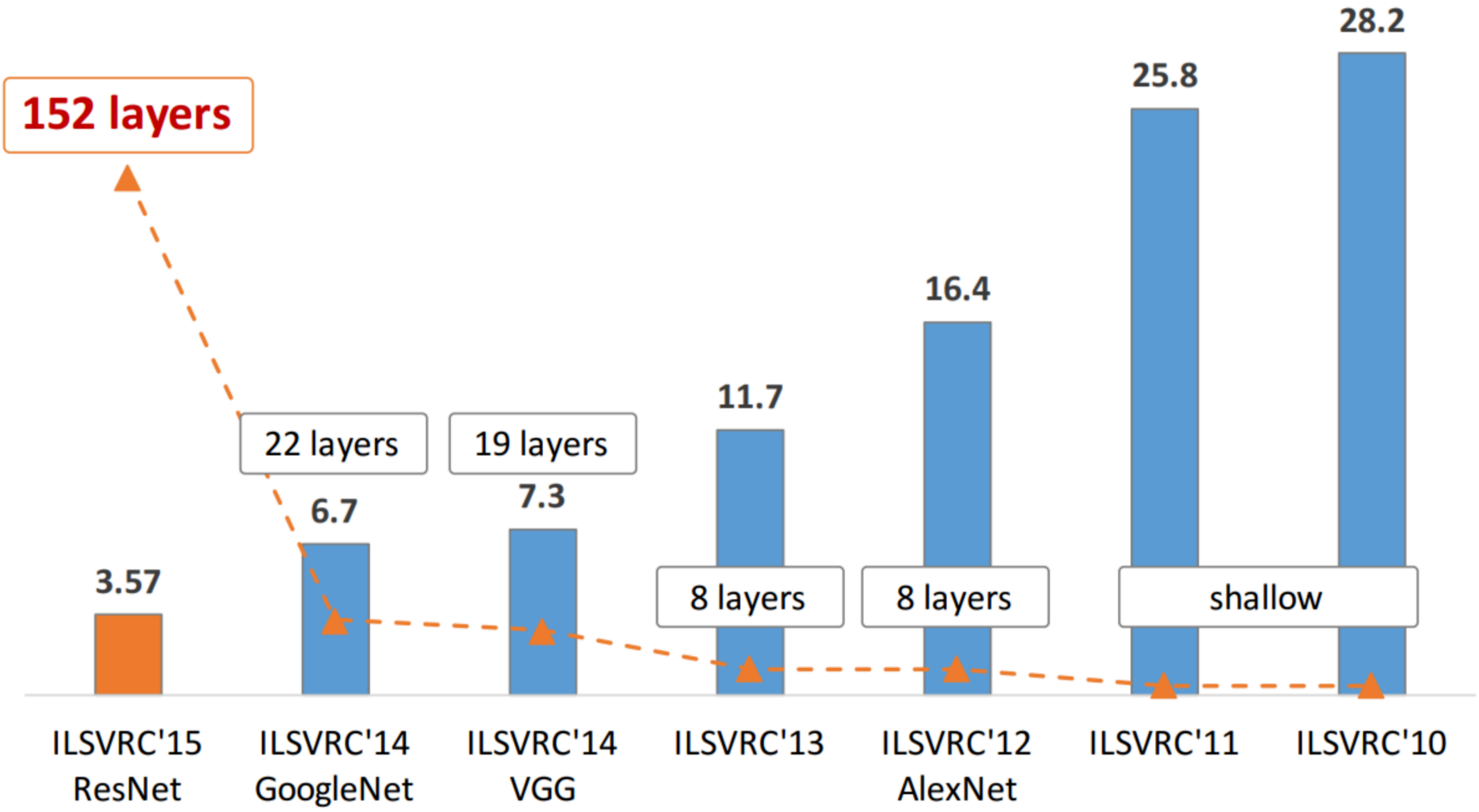
Output: Normalized, scaled and shifted values $y_1 \dots y_B$

```
1: $\mu = \frac{1}{B} \sum_{b=1}^{B} x_b$
2: $\sigma^2 = \frac{1}{B} \sum_{b=1}^{B} (x_b - \mu)^2$
3: $\hat{x}_b = \frac{x_b - \mu}{\sqrt{\sigma^2 + \epsilon}}$
4: $y_b = \gamma \hat{x}_b + \beta$
```

- Accelerates training and makes initialization less sensitive
- Zero mean and unit variance feature vectors

ResNet (2016 by Kaiming He et al.)

[Donoho et al, STATS385]



[He et al., 2016]