

# CAP 6412 Advanced Computer Vision

Website:

<http://www.cs.ucf.edu/~bgong/CAP6412.html>

Jan 14, 2016

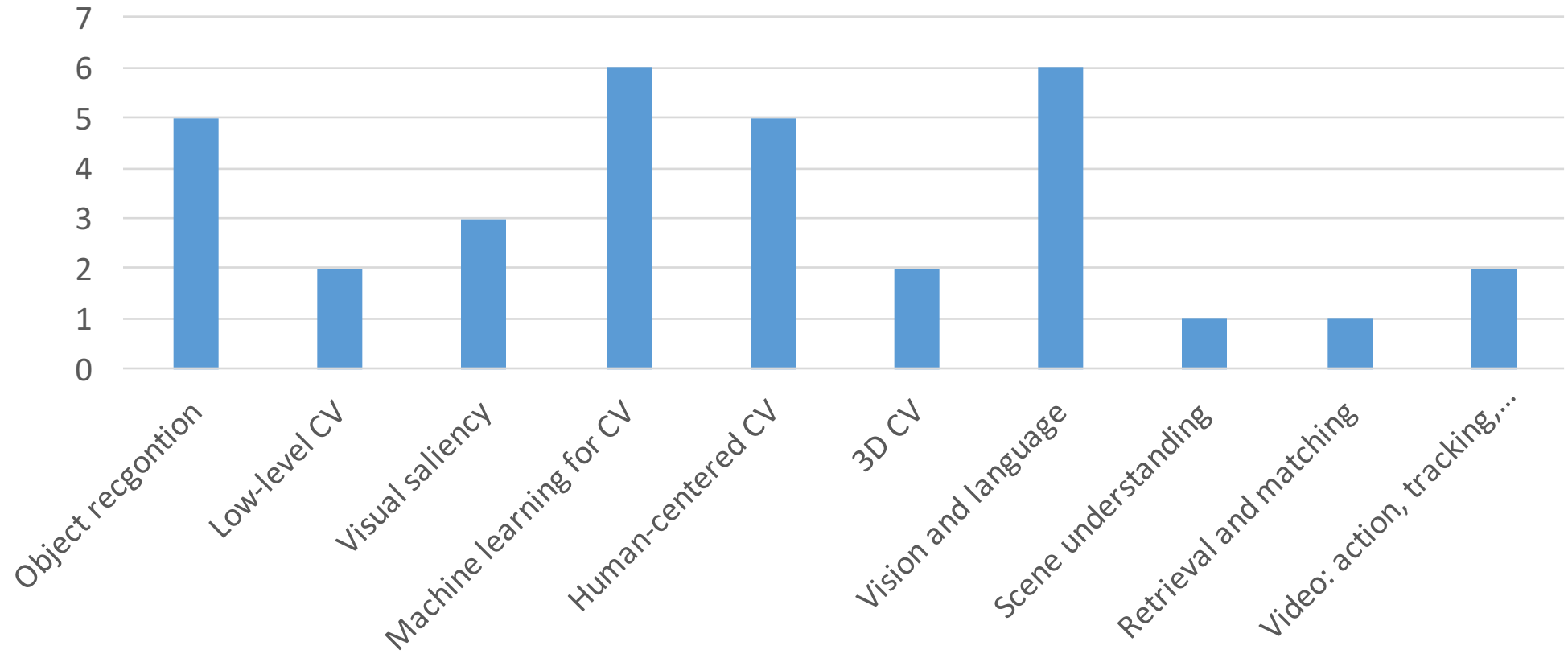
# Today

- Administrivia
- Neural networks & backpropagation (Part I)
- Fundamentals of Convolutional Neural Networks (CNN), by Fareeha

# Webcourse vs. Course homepage

- Webcourse:  
<https://webcourses.ucf.edu/>
  - Announcements
    - Check your UCF email!
  - Homework submission
- Course homepage:  
<http://www.cs.ucf.edu/~bgong/CAP6412.html>
  - All the others
    - Lecture notes, papers, links to resources, syllabus, etc.
    - Bookmark and check regularly

# Topics you have chosen



# Tentative schedule

---

<b>Week 2</b>	<b>CNN visualization &amp; object recognition</b>
Week 3	CNN & object localization
Week 4	CNN & transfer learning
Week 5	CNN & segmentation, super-resolution
Week 6	CNN & videos (optical flow, pose)
Week 7	Image captioning & attention model
Week 8	Visual question answering
Week 9	Attention model, aligning books with movies
Week 10--16	Video: tracking, action, surveillance Human-centered CV 3D CV Low-level CV, etc.

---

# Next week: CNN visualizatin & object recognition

Tuesday (01/19)	<p><b>[ILSVRC]</b> Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. “Imagenet large scale visual recognition challenge.” International Journal of Computer Vision (2014): 1-42.</p> <p><b>[152 layers]</b> He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” arXiv preprint arXiv:1512.03385 (2015).</p>
Thursday (01/21)	<p><b>[Visualization]</b> Zeiler, Matthew D., and Rob Fergus. “Visualizing and understanding convolucional networks.” In Computer Vision–ECCV 2014, pp. 818-833. Springer International Publishing, 2014.</p> <p>Zhou, Bolei, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. “Object detectors emerge in deep scene cnns.” arXiv preprint arXiv:1412.6856 (2014).</p>



# Sign up for presentations in CAP 6412



File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

Comments

Share

\$ % .0+ .00 123
Arial
10
**B**
*I*
~~ABC~~
A
More

*fx*

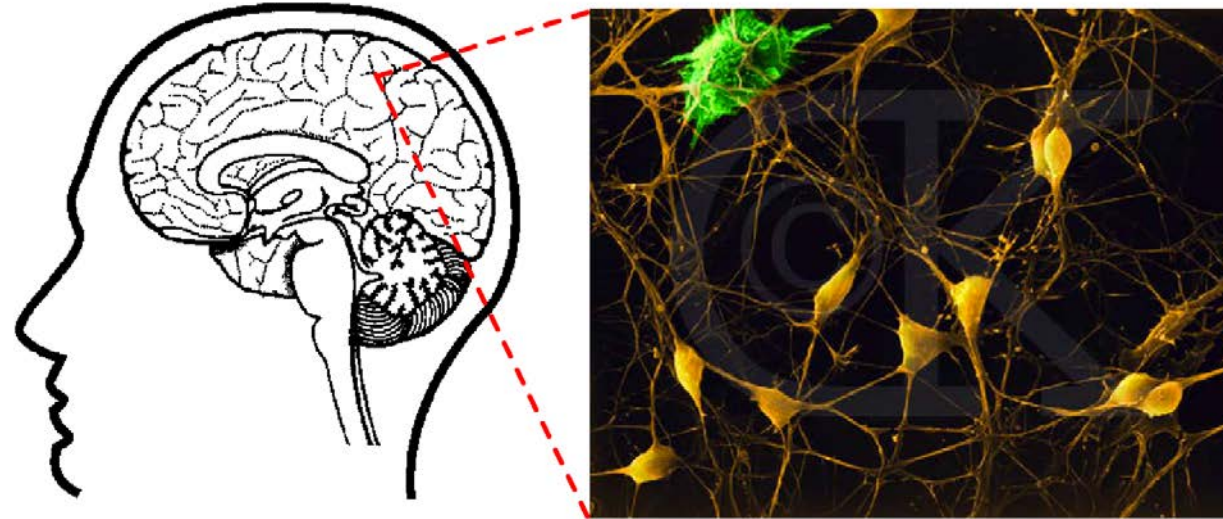
	A	B	C	D
1	Week	Topics	Volunteer for class Tuesday	Volunteer for class Thursday
2	Week 2	CNN & object recognition		
3	Week 3	CNN & object localization		
4	Week 4	CNN& transfer learning		
5	Week 5	CNN& segmentation, super-resolution		
6	Week 6	CNN & videos (optical flow, pose)		
7	Week 7	Image captioning &attention model		
8	Week 8	Visual question answering		
9	Week 9	Attention model, aligningbooks with movies		
10	Week 10--16	TBD		

[Link](#) will be sent to your UCF emails

# Today

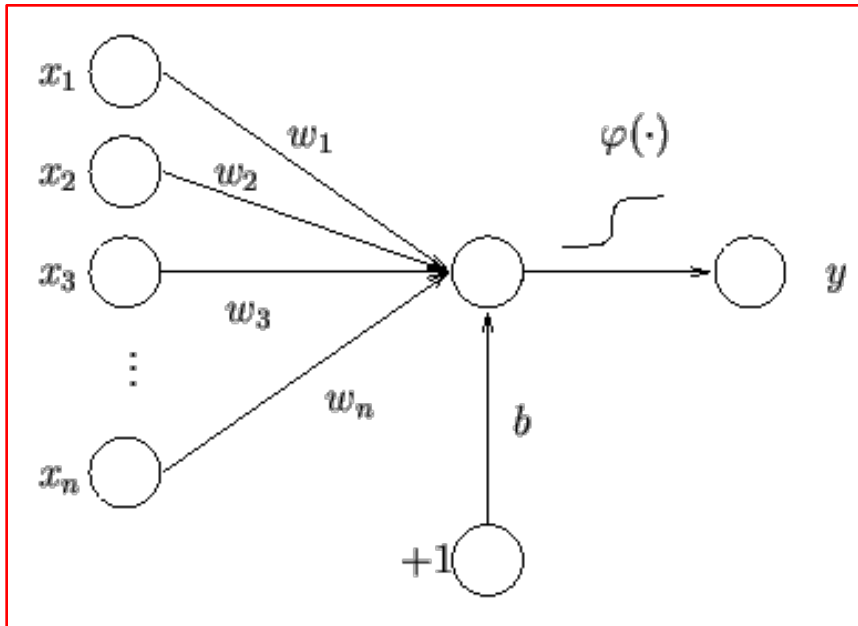
- Administrivia
- **Neural networks & backpropagation (Part I)**
- Fundamentals of Convolutional Neural Networks (CNN), by Fareeha

# Biological neurons



- Human brains has about 10 billion nuerons
- Each connected to 10K other neurons
- A neuron fires if the sum of electrochemical inputs exceeds some threshold

# Artificial neurons --- perceptrons

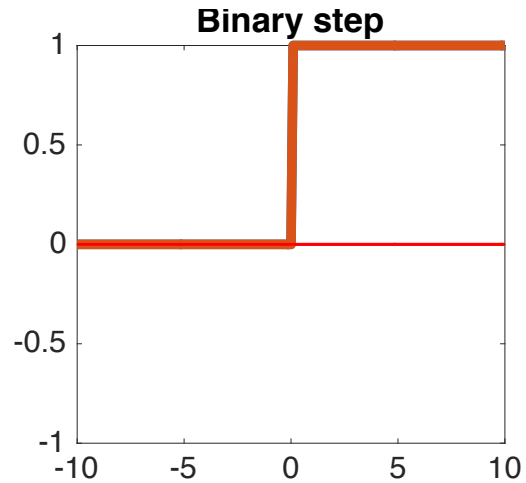


$$y = \varphi\left(\sum_{i=1}^n w_i x_i + b\right)$$
$$= \varphi(\mathbf{w}^T \mathbf{x} + b)$$

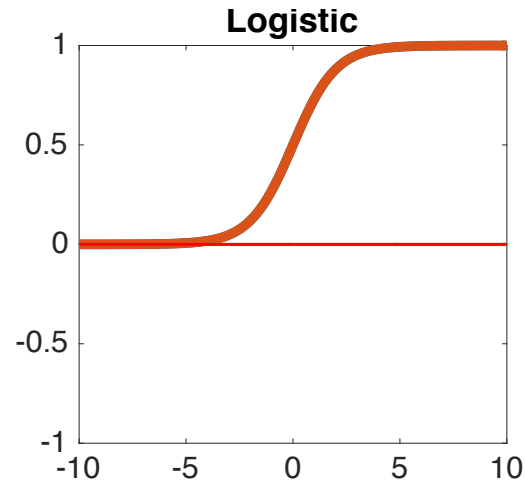
$\varphi(\cdot)$  : activation function

- Introduced by Rosenblatt in 1958
- The **basic building blocks** for (not all) neural networks

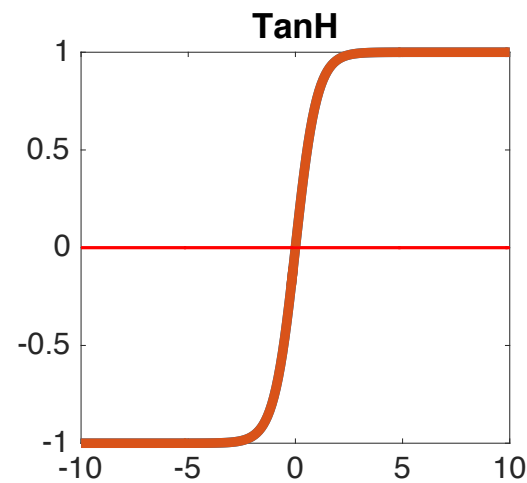
# Popular activation functions



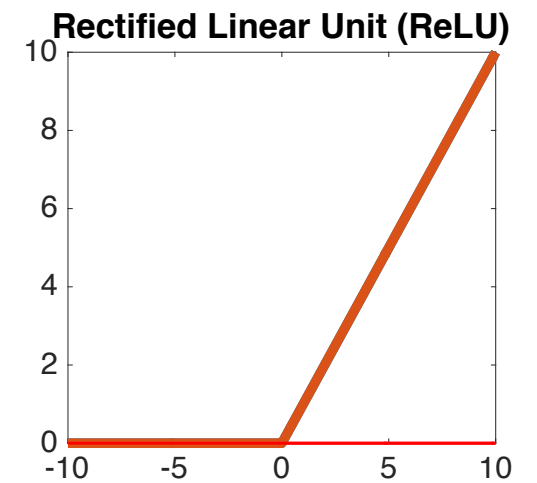
$$\varphi(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$



$$\varphi(x) = \frac{1}{1 + \exp(-x)}$$



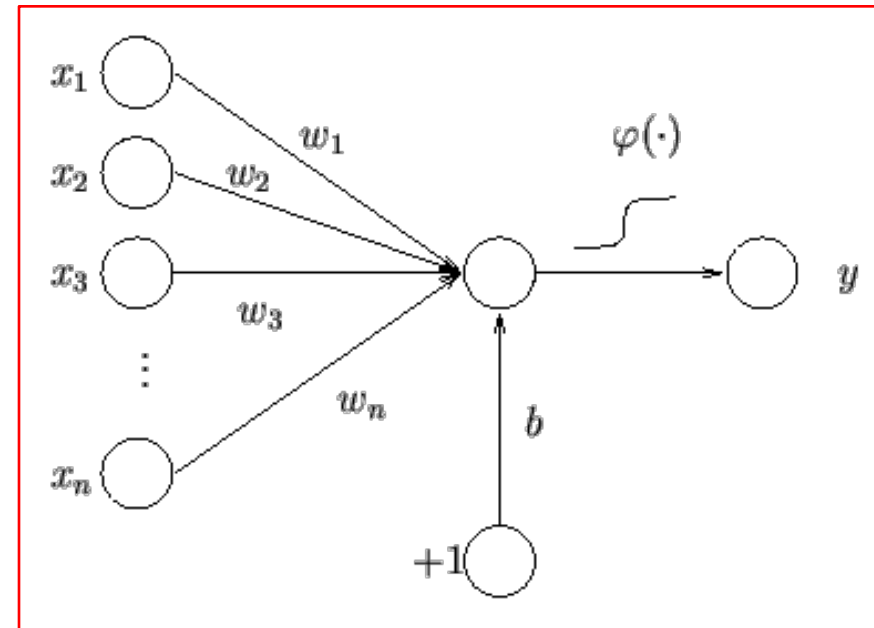
$$\begin{aligned} \varphi(x) &= \tanh(x) \\ &= \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \end{aligned}$$



$$\varphi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

# Artificial neurons --- perceptrons

- Support Vector Machines
- Logistic regression
- AND
- OR
- NOT
- XOR ?
  
- Linear regression



# Building neural networks from perceptrons

- Next Tuesday

# Today

- Administrivia
- Neural networks & backpropagation (Part I)
- **Fundamentals of Convolutional Neural Networks (CNN), by Fareeha**

# Convolutional Neural Networks

Fareeha Irfan

# Outline

- ❑ Background
- ❑ Applications: Convnets for object recognition and language
- ❑ How to design convolutional layers
- ❑ How to design pooling layers
- ❑ How to integrate back-propagation in Convnets
- ❑ How to build convnets in torch
- ❑ AlexNet

# Background

- ❑ Complex classification tasks
- ❑ Object Recognition in Images:
  - ❑ grayscale:  $32 \times 32 = 1024$  pixels
  - ❑ rgb:  $32 \times 32 \times 3 = 3072$  pixels
  - ❑ Fully-connected NN becomes computationally intensive

Algorithm that mimics the brain..

- Neural connections
- Neurons activated during learning

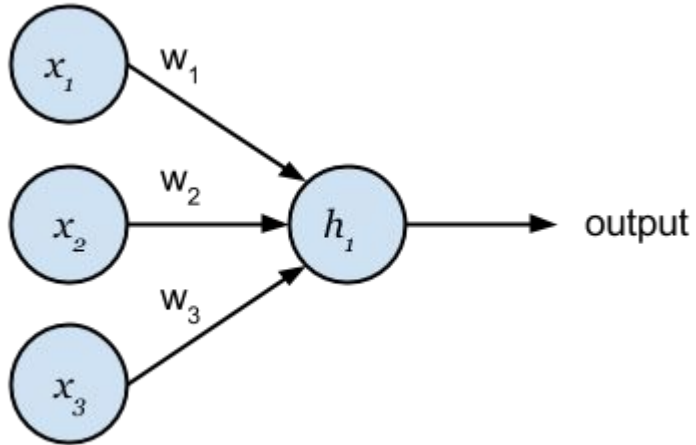
# Convnet Applications

- **Image/Object Recognition:** Can predict who is in the image, what pose are they in.
- **Natural Language Processing:** Predict sentiments about sentences to classify tweets. Extract summaries by finding sentences that are most predictive.
- **Drug Discovery:** Predicting the interaction between molecules and biological proteins can be used to identify potential treatments.

## Some Common Libraries:

- Caffe : Supports both CPU & GPU. Developed in C++
- Torch framework: Written in C
- Cuda-convnet: Implementation in CUDA

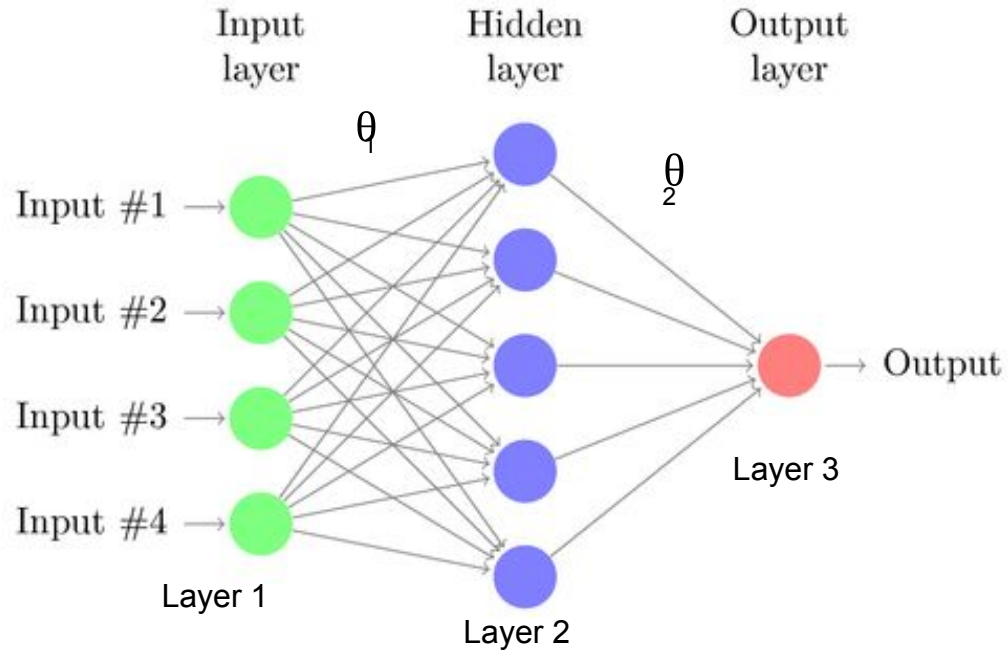
# A Simple Neural Network



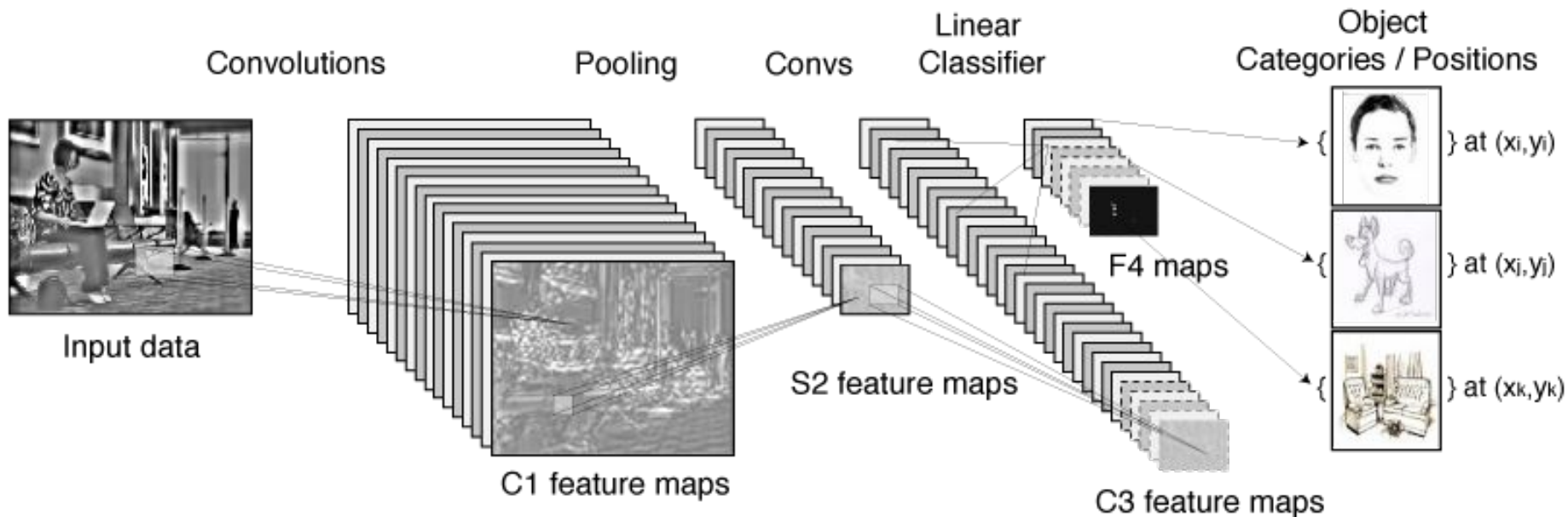
Activation Functions:

- Sigmoid
- Hyperbolic
- Tangent
- ReLU (Rectified Linear Unit)

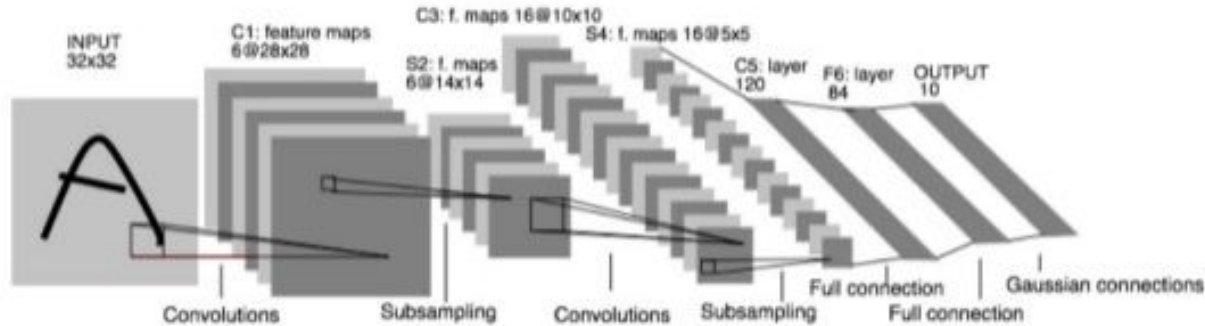
# Neural Network



# Convnet Overview



# Convolutional Neural Nets (CNNs): 1989



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [ LeNet ]

## Neural Network

### Layer 1 (C1)

parameters:

$$(32*32+1)*(28*28+1)*6 \\ = 4827750$$

## ConvNet

### Layer 1 (C1)

parameters:

$$(5*5+1)*6 \\ = 156$$

# Convolutional Layer

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

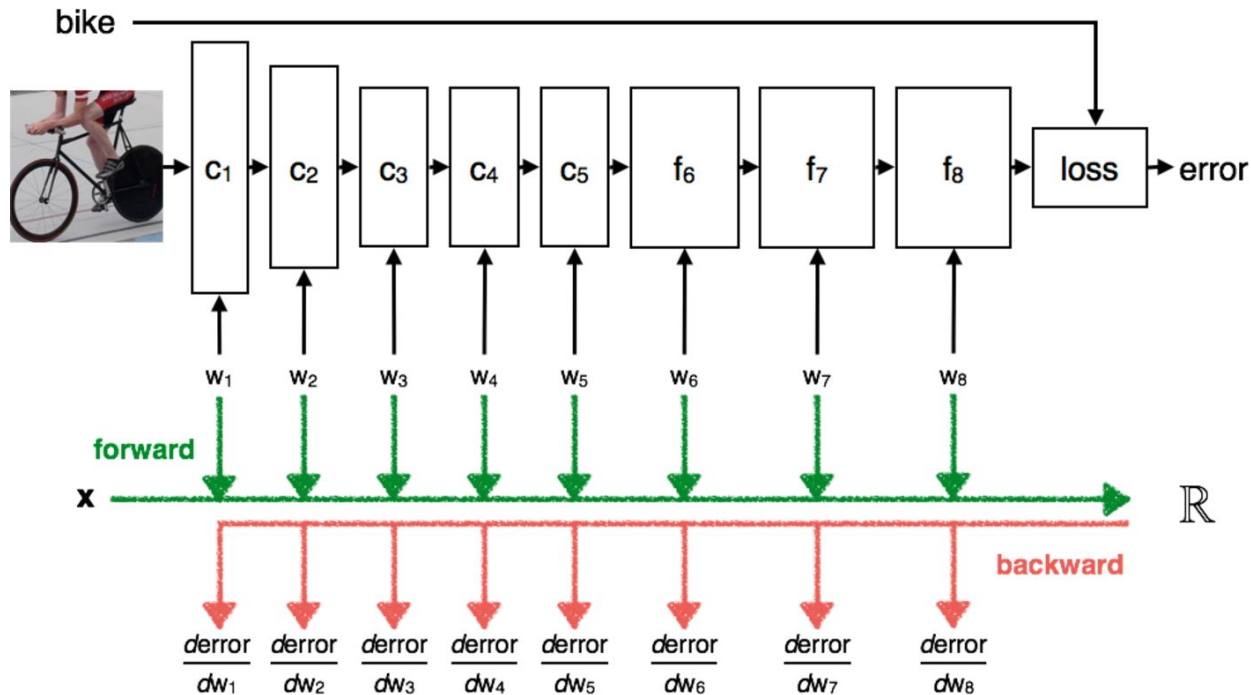
$y$  : Output of the convolution

$x$  : Map with  $K$  channels

$K'$  : Total filters, generating a  $K'$  dimensional map  $y$

$$y_{i'j'k'} = \sum_{ijk} w_{ijkk'} x_{i+i',j+j',k}$$

# Back-propagation



# Back-propagation for Conv Layer

$$\delta_{ijf}^l = \sum_{i'j'f'} \delta_{i'j'f'}^{l+1} \theta_{i-i'+1, j-j'+1, f, f'}$$

# Pooling Layer

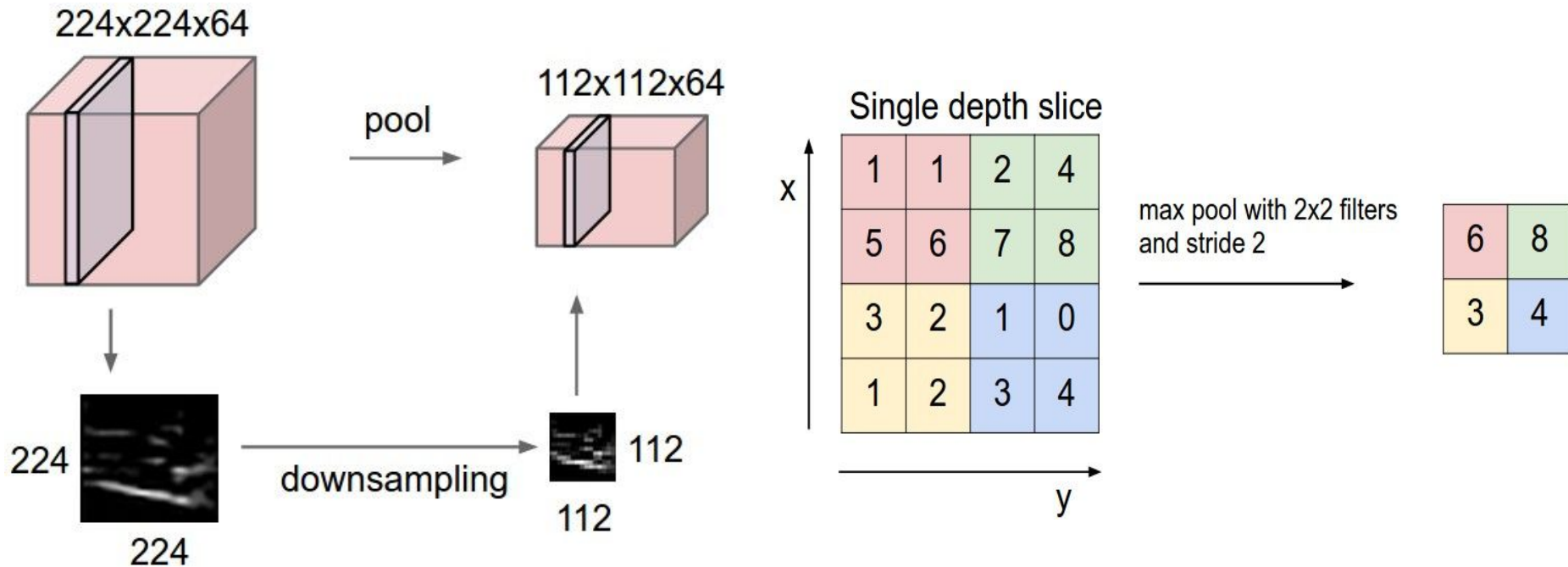
A pooling operator operates on individual feature channels, coalescing nearby feature values into one by the application of a suitable operator.

Common choices include max-pooling (using the max operator) or sum-pooling (using summation).

*Max-pooling* is defined as:

$$y_{i',j'} = \max_{ij \in \Omega(i',j')} x_{ij}$$

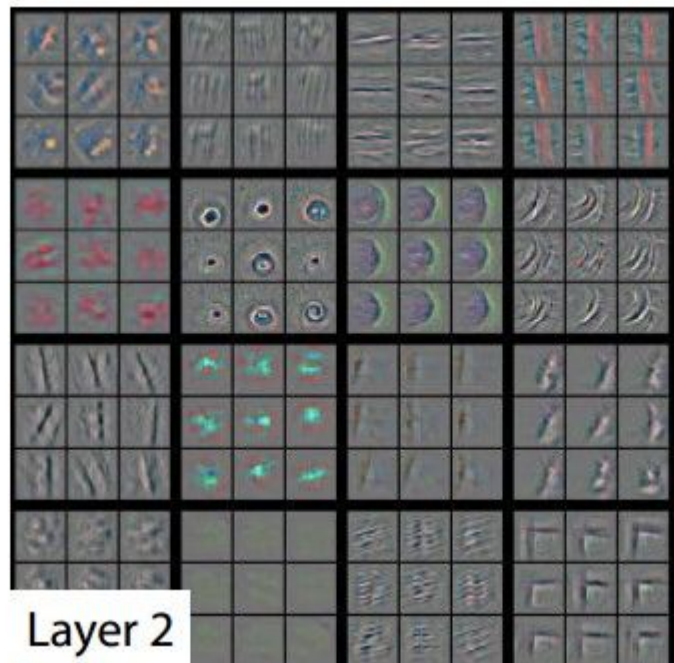
# Pooling Layer



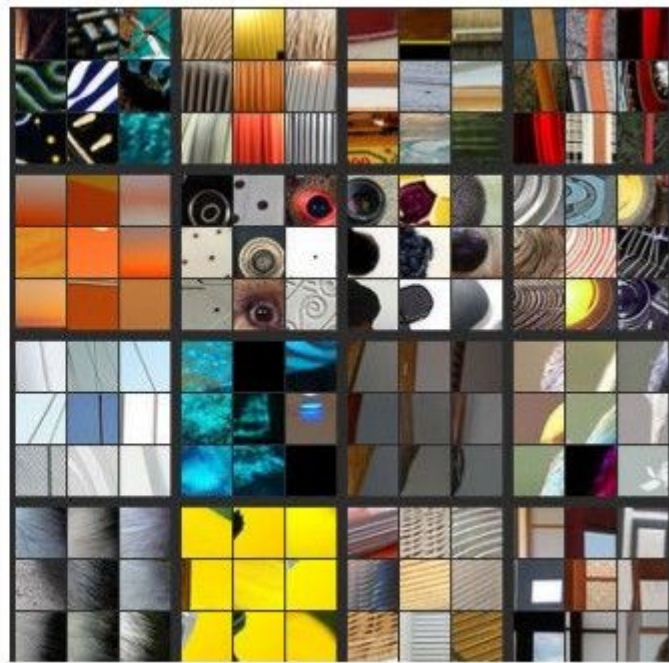
# Convnet

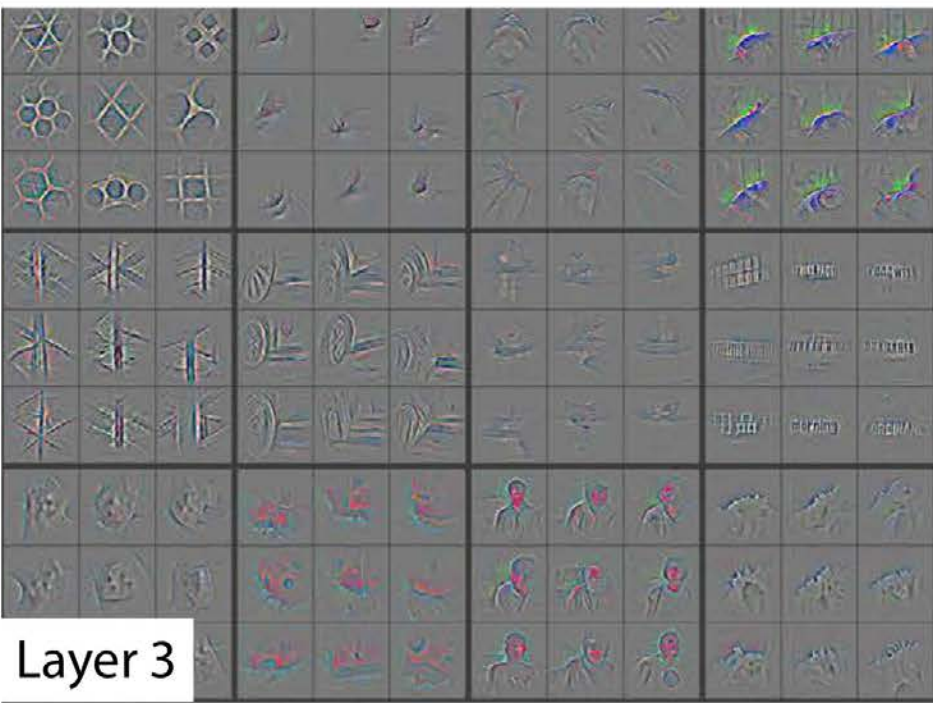


Layer 1



Layer 2





# Convnets in Torch

```
1 | model = nn.Sequential()
2 | model:add(nn.Reshape(1,32,32))
3 | -- layer 1:
4 | model:add(nn.SpatialConvolution(1, 16, 5, 5))
5 | model:add(nn.Tanh())
6 | model:add(nn.SpatialMaxPooling(2, 2, 2, 2))
7 | -- layer 2:
8 | model:add(nn.SpatialConvolution(16, 128, 5, 5))
9 | model:add(nn.Tanh())
10 | model:add(nn.SpatialMaxPooling(2, 2, 2, 2))
11 | -- layer 3, a simple 2-layer neural net:
12 | model:add(nn.Reshape(128*5*5))
13 | model:add(nn.Linear(128*5*5, 200))
14 | model:add(nn.Tanh())
15 | model:add(nn.Linear(200,10))
16 | model:add(nn.LogSoftMax())
```

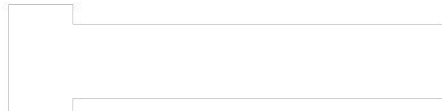
# ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky,  
Ilya Sutskever,  
Geoffrey E. Hinton  
*University of Toronto*

# Paper's contribution

- Trained one of the largest CNNs on subsets of ImageNet
- Achieved the best reported results (by far)
- Publically available GPU implementation
- Describes unusual techniques to improve performance

# Motivation



“Consider for example the problem of **object recognition in computer vision**: we could be interested in building recognizers for at least several thousand categories of objects. Should we have specialized algorithms for each?”

...

“Are we going to have to do this **labor-intensive work for all the possible types of [objects]**? our system will not be very smart if we have to **manually engineer** new patches each time ... new types of object category must be processed. If there exist more **general-purpose learning models**, ... , then searching for them may **save us a considerable amount of labor in the long run.**”

Bengio, **LeCun**. Scaling Learning Algorithms towards AI (2007)



15 million labeled images with 22,000 categories

Labeled by humans using...

Amazon's Mechanical Turk crowdsourcing tool  
pay people to manually label large datasets

# ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

Subset of the full ImageNet dataset

1.2 million training images

50,000 validation, 150,000 test images

**1000 different classes!**

~1000 images in each category (not an unreasonable number for MIA)

amazing **Results** to motivate why we learn about this approach

On test (unseen) data:

top-1 error = 37.5% (prev best was 45.7%)

top-5 error = 15.3% (second-best 26.2%)

# Results

Correct label →

correct predictions



mite

container ship

motor scooter

leopard

mite black widow cockroach tick starfish	container ship lifeboat amphibian fireboat drilling platform	motor scooter go-kart moped bumper car golfcart	leopard jaguar cheetah snow leopard Egyptian cat

Correct label →

incorrect? predictions



grille

mushroom

cherry

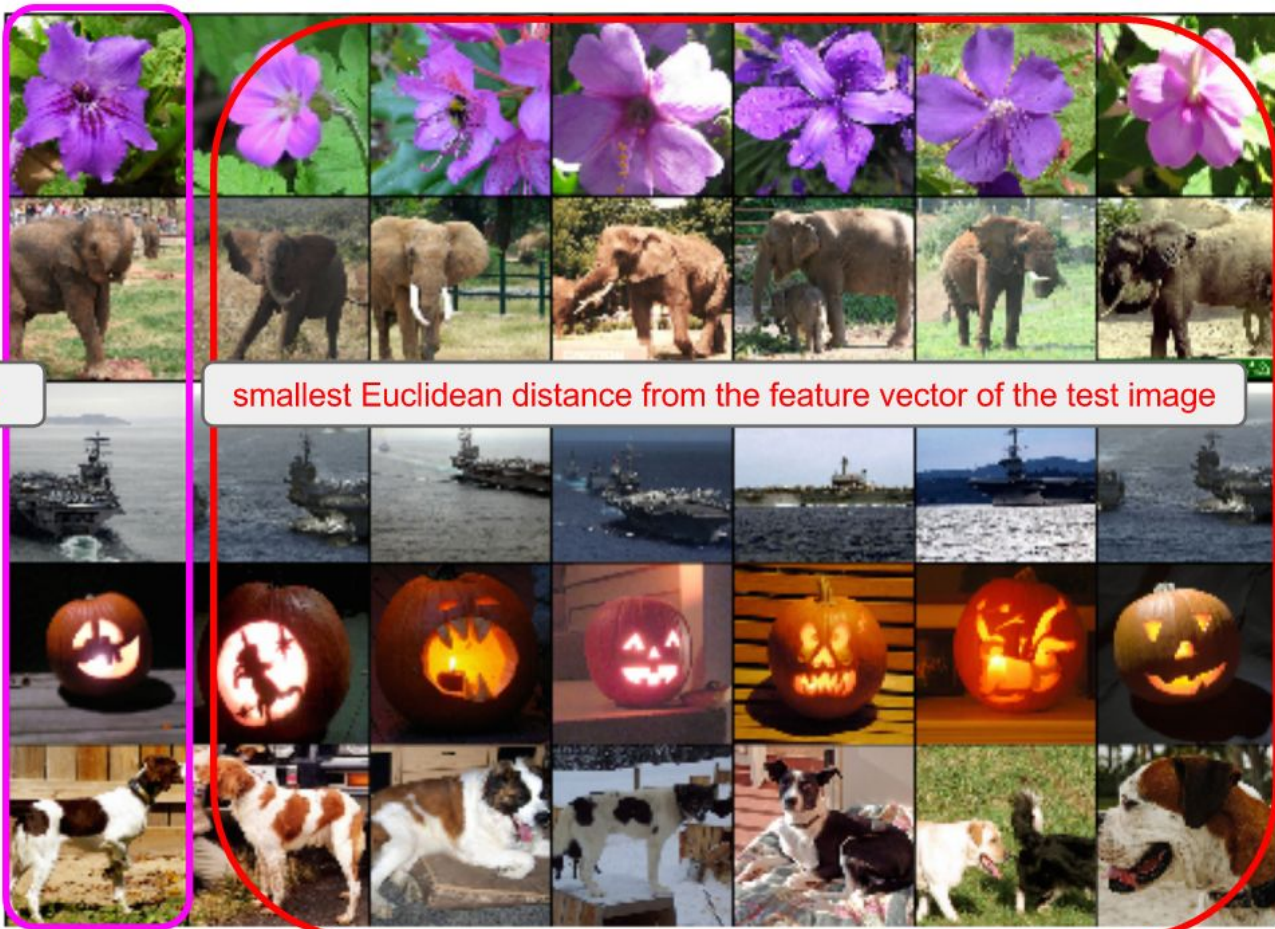
Madagascar cat

convertible grille pickup beach wagon fire engine	agaric mushroom jelly fungus gill fungus dead-man's-fingers	dalmatian grape elderberry ffordshire bullterrier currant	squirrel monkey spider monkey titi indri howler monkey

# Results

test image

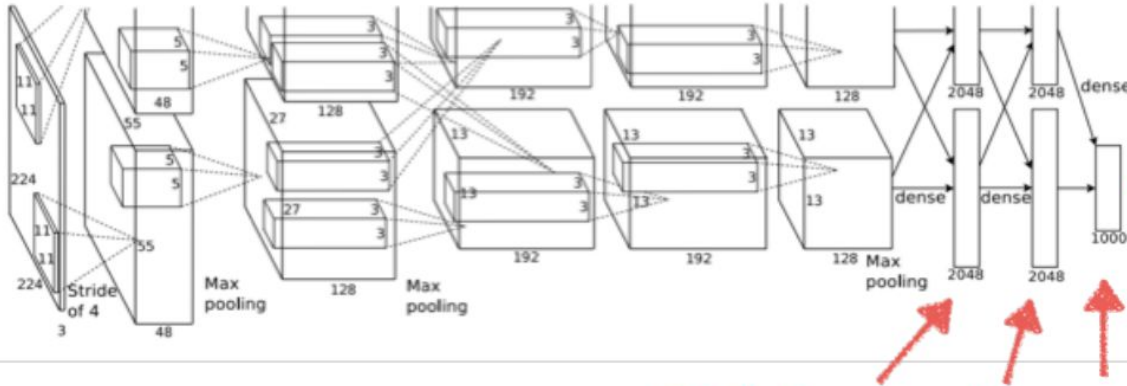
smallest Euclidean distance from the feature vector of the test image



# Architecture

- 60 million parameters
- 650,000 neurons
- 5 convolutional layers ( followed by max-pooling layers)
- 3 fully-connected layers with
- a 1000-way softmax final layer

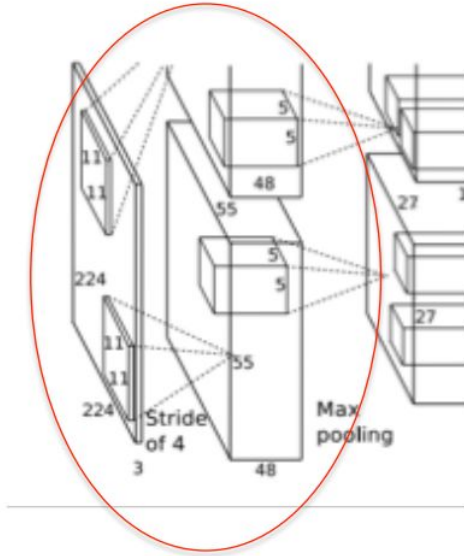
5 Convolutional Layers



1000-way  
softmax

3 Fully Connected Layers

# Layer 1 (Convolutional)



- Images: 227x227x3
- F (receptive field size): 11
- S (stride) = 4
- Conv layer output: 55x55x96

# Architecture

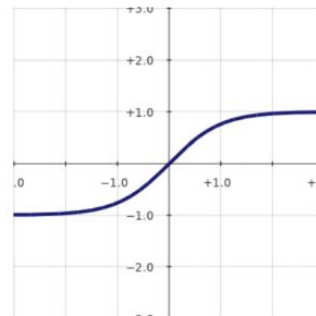
## RELU Nonlinearity

- Standard way to model a neuron

$$f(x) = \tanh(x) \quad \text{or} \quad f(x) = (1 + e^{-x})^{-1}$$

Very slow to train

$$f(x) = \tanh(x)$$

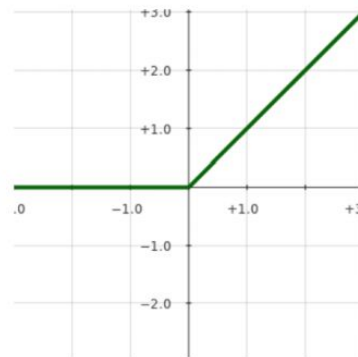


- Non-saturating nonlinearity (RELU)

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

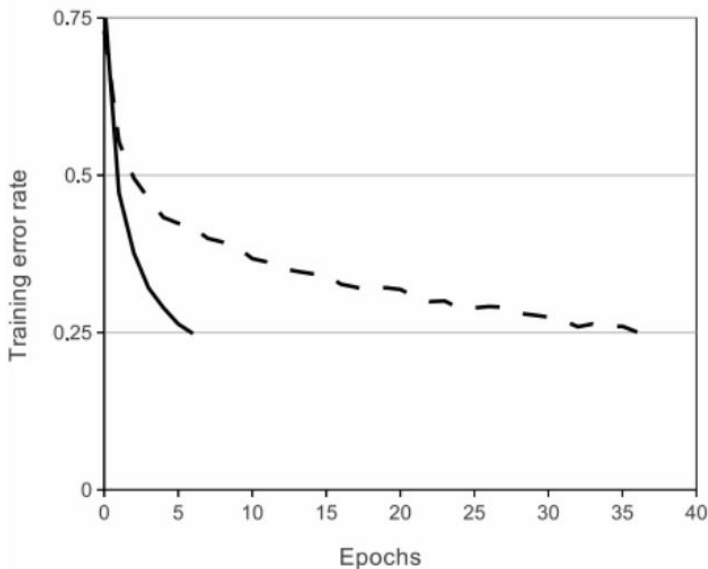
Quick to train

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



# Architecture

## RELU Nonlinearity



A 4 layer CNN with ReLUs (solid line) converges **six times faster** than an equivalent network with tanh neurons (dashed line) on CIFAR-10 dataset

# Local Response Normalization

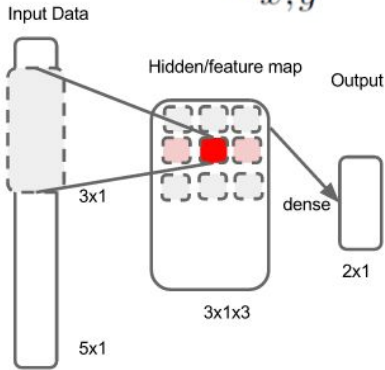
After applying a kernel  $i$  at position  $(x,y)$ , we get an activity  $a$  for a neuron.

We divide the neuron activity  $a$  by the other activity of neurons in the other neighbouring kernel maps.

Reduces error by about 1.3%

sum over  $n$  neighbouring kernel maps

$$b_{x,y}^i = a_{x,y}^i / \left( k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$



$N$  = total number of kernels per layer

Example where  $N = 3$

hyper-parameters:  
 $k = 2$ ,  $\alpha = 10^{-4}$ ,  $n = 5$ ,  $\beta = 0.75$   
 found using the **validation** set

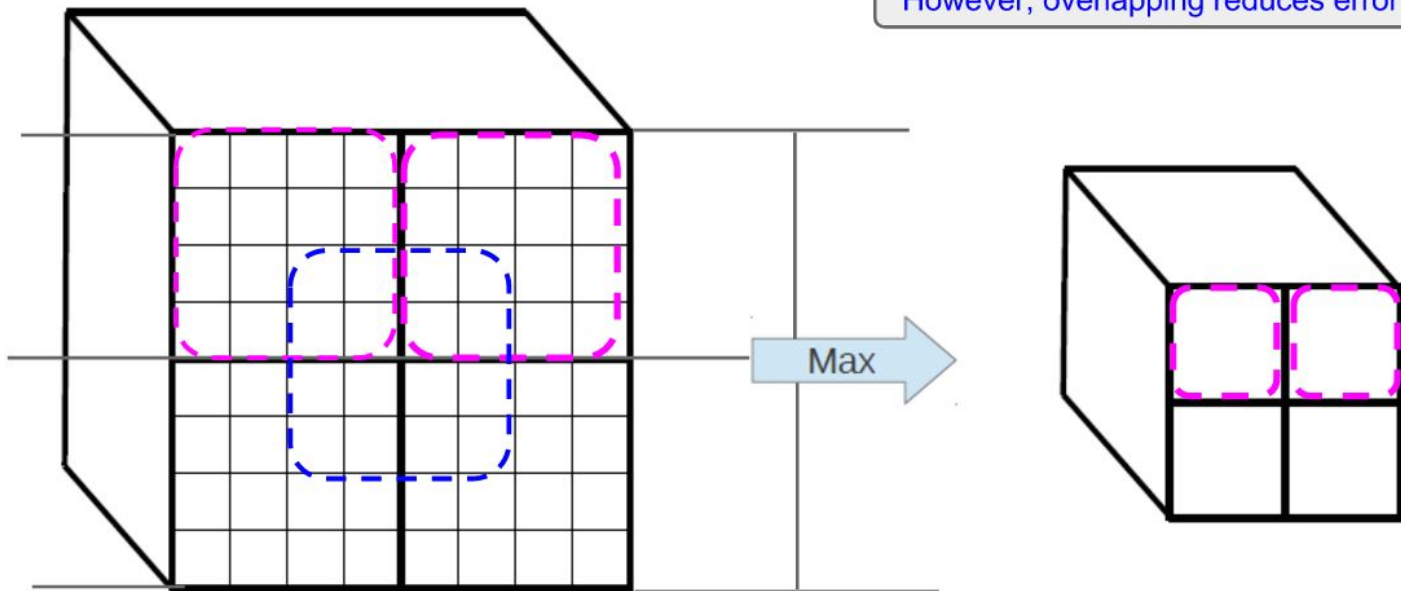
Done to give a form of **lateral inhibition**  
 inspired what is found in real neurons

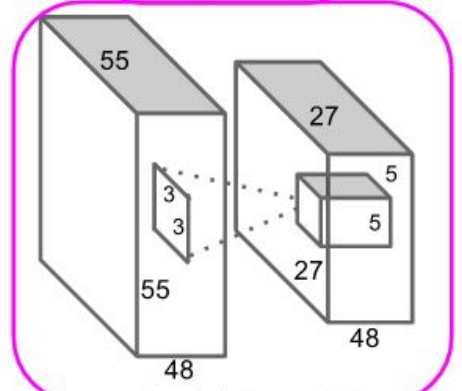
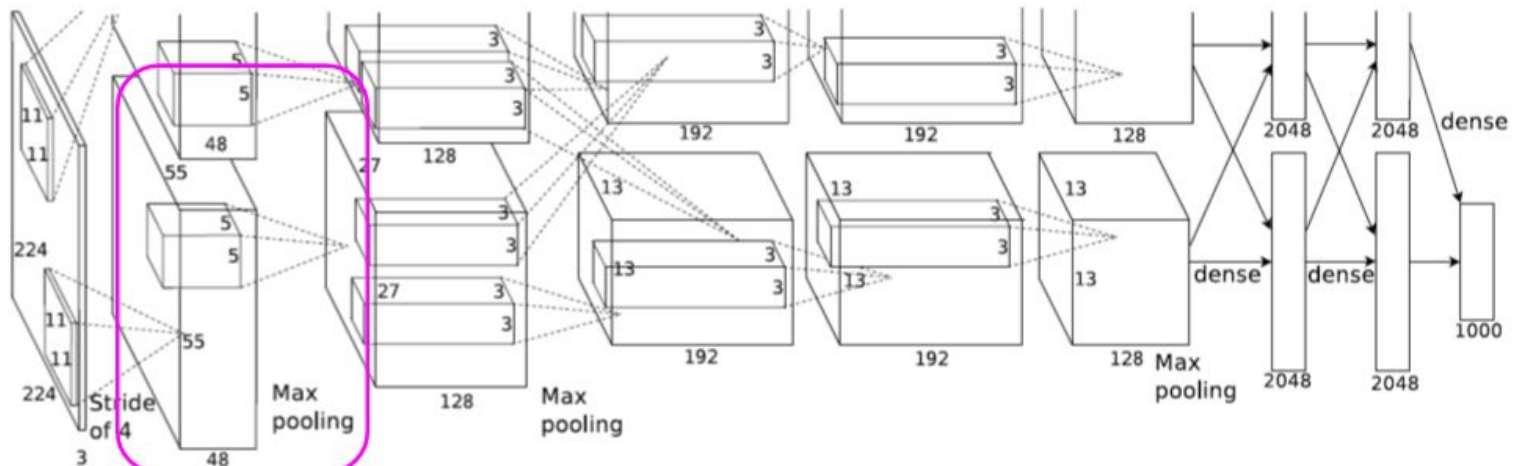
# Max Local Pooling

Summarizes the outputs of neighbouring groups

Traditionally, neighbourhoods don't overlap

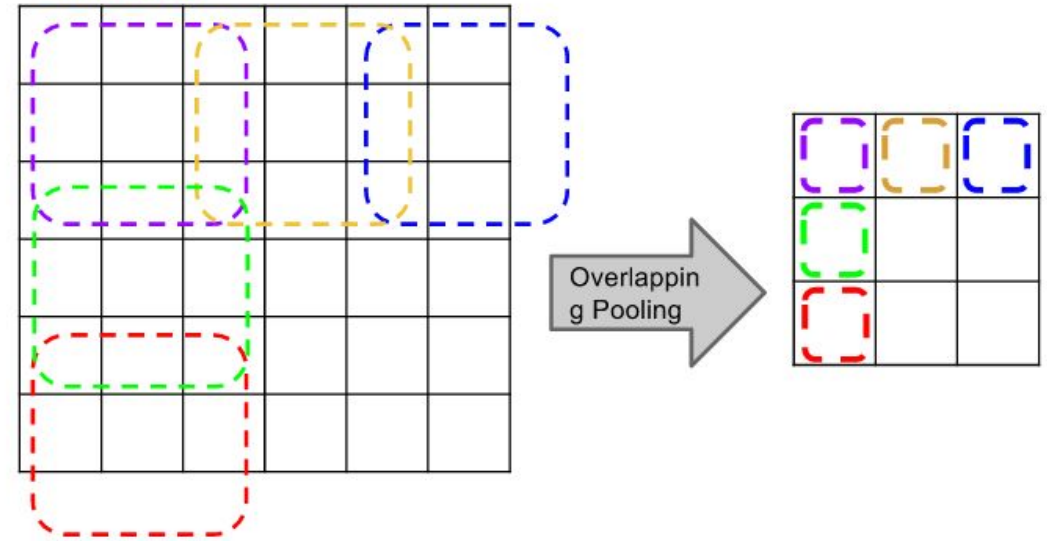
However, overlapping reduces error by ~0.4%





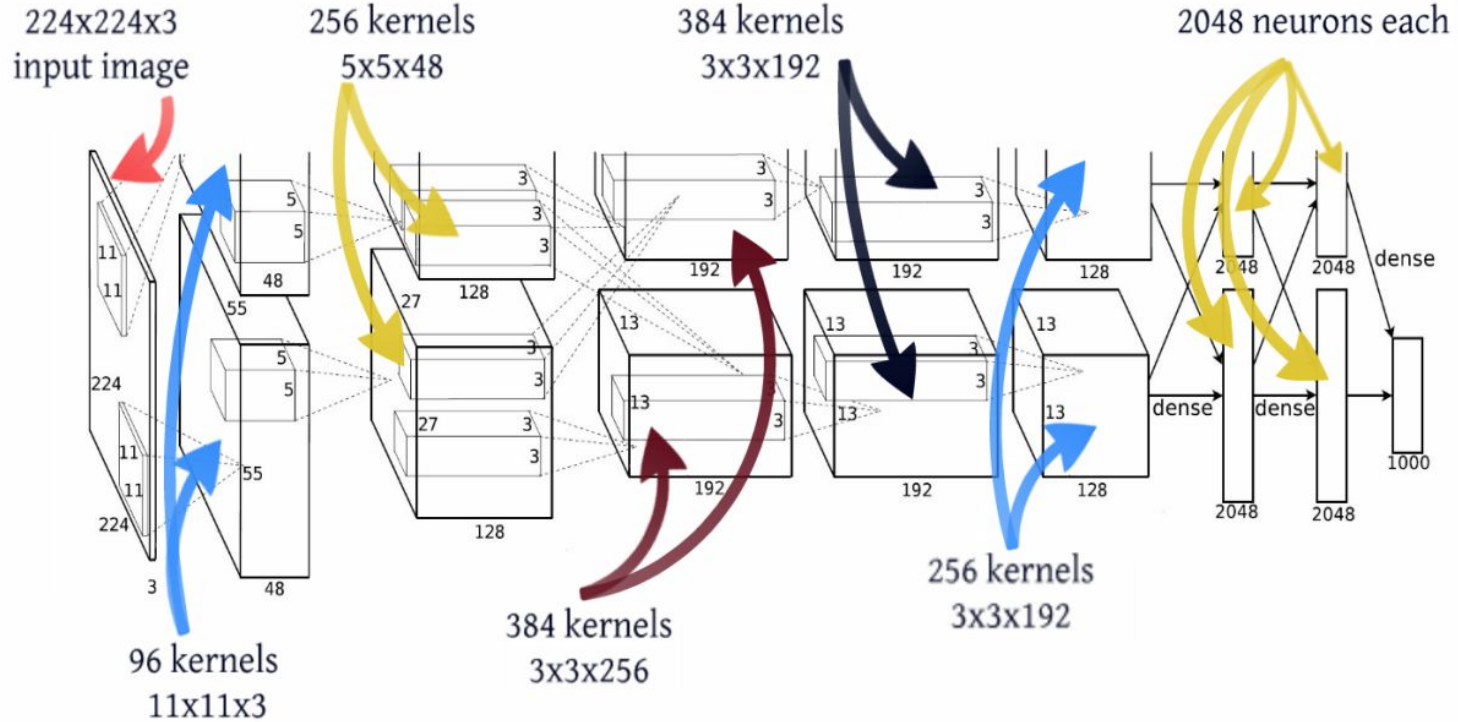
Max-pooling (window = 3x3, stride = 2)

Max-pooling (window = 3x3, stride = 2)



"The second convolutional layer takes as input the (response-normalized and **pooled**) output of the first convolutional layer and filters it with 256 kernels of size 5 x 5 x 48"

# Architecture



# Architecture Overview

4M	<b>FULL CONNECT</b>	4Mflop
16M	<b>FULL 4096/ReLU</b>	16M
37M	<b>FULL 4096/ReLU</b>	37M
	<b>MAX POOLING</b>	
442K	<b>CONV 3x3/ReLU 256fm</b>	74M
1.3M	<b>CONV 3x3ReLU 384fm</b>	224M
884K	<b>CONV 3x3/ReLU 384fm</b>	149M
	<b>MAX POOLING 2x2sub</b>	
	<b>LOCAL CONTRAST NORM</b>	
307K	<b>CONV 11x11/ReLU 256fm</b>	223M
	<b>MAX POOL 2x2sub</b>	
	<b>LOCAL CONTRAST NORM</b>	
35K	<b>CONV 11x11/ReLU 96fm</b>	105M

# Reducing Overfitting

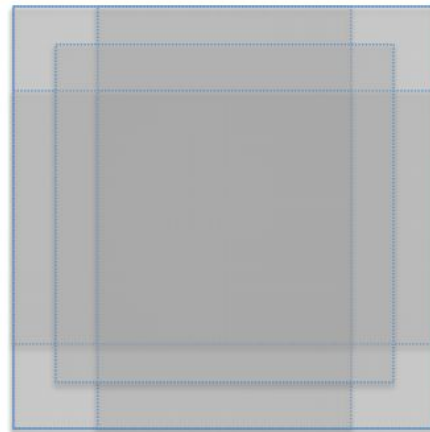
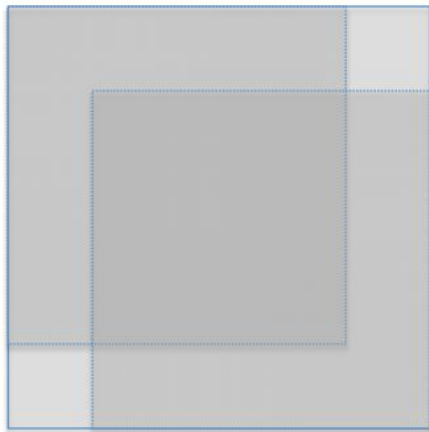
## Data Augmentation

- 60 million parameters, 650,000 neurons  
→ Overfits a lot.
- Crop  $224 \times 224$  patches (and their horizontal reflections.)

# Reducing Overfitting

## Data Augmentation

- At test time, average the predictions on the 10 patches.



# Reducing Overfitting

Reduces the top-1 error rate by over 1%

## Data Augmentation

- Change the intensity of RGB channels
- 

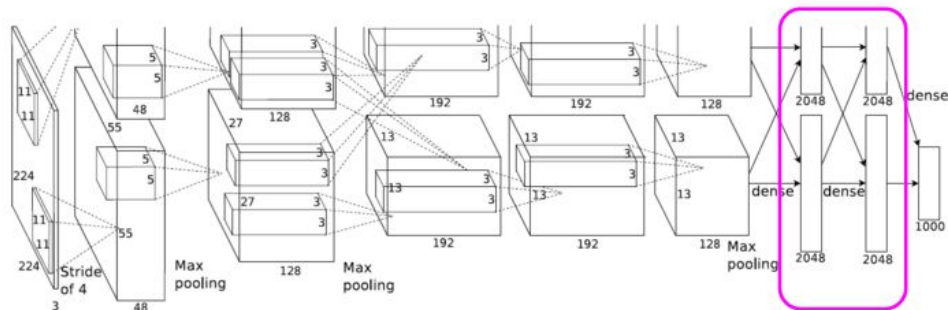
$$I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$$

add multiples of principle components

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1 \lambda_1, \alpha_2 \lambda_2, \alpha_3 \lambda_3]^T$$

$$\alpha_i \sim N(0, 0.1)$$

# Dropout



- Set to 0 the output of a neuron with 0.5 probability
- Reduces complex co-adaption and forces to learn more robust features
- Done in last two layers

A hidden layer's activity on a given training image



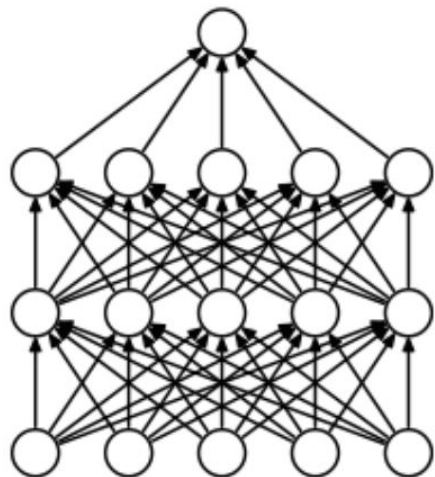
A hidden unit  
turned off by  
dropout



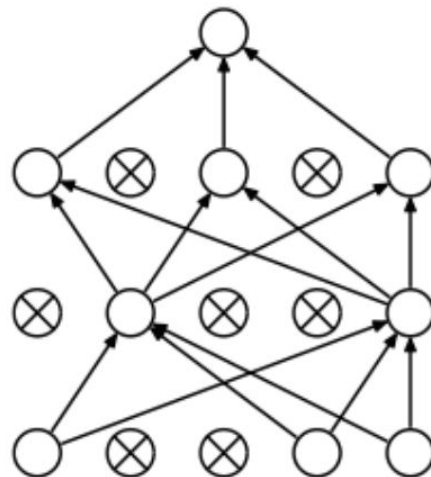
A hidden unit  
unchanged

# Reducing Overfitting

## Dropout



Standard Neural Net



After applying dropout.

- With probability 0.5
- last two 4096 fully-connected layers.

Propagate errors back, and update weights to take a small step in the direction that minimizes the error

# Training

Using stochastic gradient descent and the backpropagation algorithm (repeated application of the chain rule)

One output unit per class

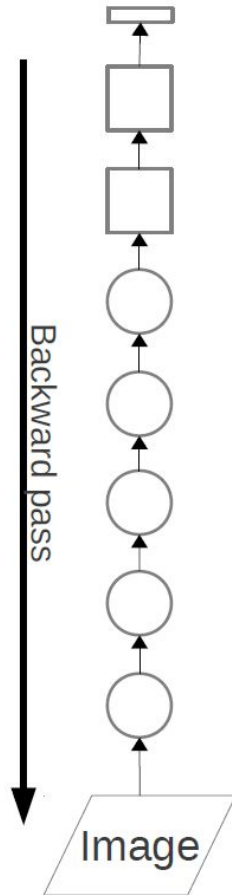
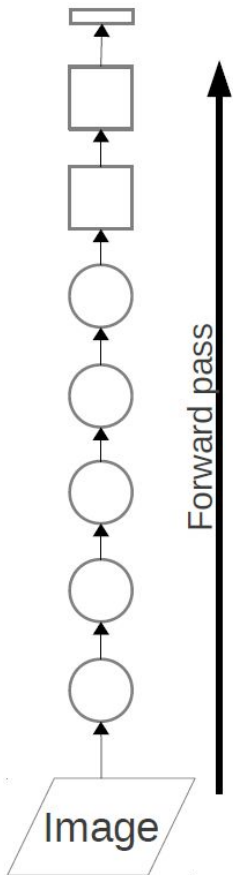
$x_i$  = total input to output unit  $i$

$$f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)}$$

We maximize the log-probability of the correct label,  $\log f(x_t)$

Optimize so the correct label is predicted

Start with some initialized weights



# Reducing Overfitting

- Softmax

$$L = \frac{1}{N} \sum_i -\log \left( \frac{e^{f_{y_j}}}{\sum_j e^{f_j}} \right) + \lambda \sum_k \sum_l W_{k,l}^2$$

$j = 1 \dots 1000$

$P(y_i | x_i; W)$  Likelihood

# Stochastic Gradient Descent Learning

## Updating the weights

**Momentum:** Adds a fraction of the previous weight update to the current one (increases speed of convergence)

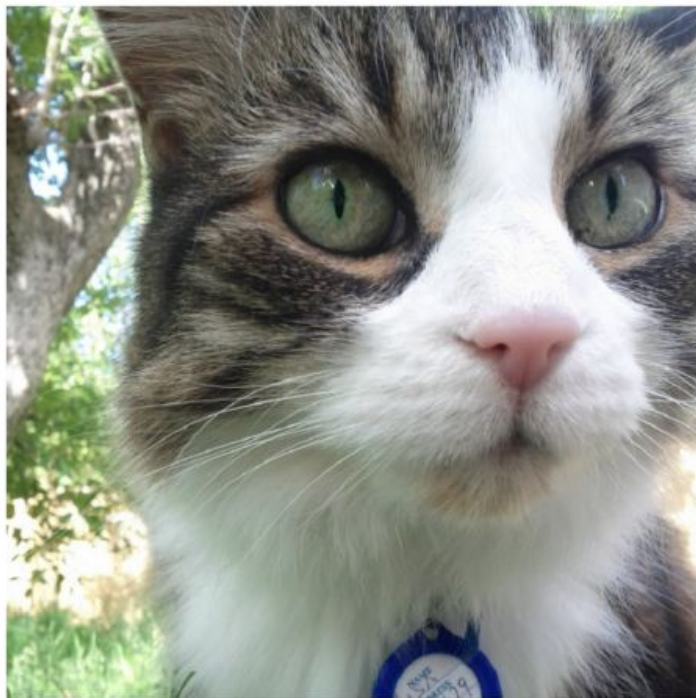
**Batch size** = how many examples in an iteration  $i$  to train the model on = 128 examples

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} \Big|_{w_i} \right\rangle_{D_i}$$
$$w_{i+1} := w_i + v_{i+1}$$

**Weight decay:** penalizes large weights (as weight increases, changes less)

Average over the  $i$ th batch  $D_i$  of the derivative of the objective with respect to  $w$ , evaluated at  $w_i$

# Tech Details - Preprocessing



An input image (256x256)



Minus sign



The mean input image

# Results

Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
CNN	<b>37.5%</b>	<b>17.0%</b>

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Also tried on ImageNet with ~10,000 categories and 8.9 million images  
Top-1 = 67.4%, Top-5 = 40.9%  
(prev best = 78.1% and 60.9%)

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
<i>SIFT + FVs [7]</i>	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	<b>16.4%</b>
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	<b>15.3%</b>

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk\* were “pre-trained” to classify the entire ImageNet 2011 Fall release. See Section 6 for details.