Introduction to Deep Learning and Convolutional Neural Networks

Maryana Alegro, PhD
Grinberg Lab, UCSF
maryana.alegro@ucsf.edu

Deep Learning???

Google | deep learning

All | News | Videos | Books | Images | More | Settings | Tools

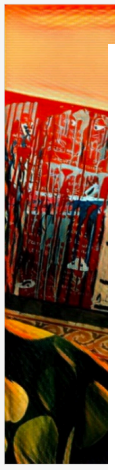
About 18,800,000 results (0.72 seconds)

Deep Learning Makes Driverless Cars Better at Spotting Pedestrians

By Jeremy Hsu
Posted 9 Feb 2016 | 17:00 GMT

Prisma uses AI to turn your photos into graphic novel fodder double quick

Posted Jun 24, 2016 by Natasha Lomas (@riptari)



LG introduces deep learning technology at CES

By . | Publish Date: Jan 9 2017 2:49AM

AdChoices | Online University | Newspapers | Nigeria News

facebook | twitter | Google+ | LinkedIn | (0 Likes)

By Zakariyya Adaramola

ars TECHNICA | BIZ & IT | TECH | SCIENCE | POLICY | CARS | GAMING & CULTURE | FORUMS

GO STONES WILL BREAK MY BONES, BUT AI OVERLORDS WILL NEVER HARM ME —

That mystery Go player crushing the world's best online? It was AlphaGo again

After 50 straight wins, DeepMind's AlphaGo earned a draw—because its connection timed out.

SEBASTIAN ANTHONY (UK) - 1/5/2017, 8:45 AM

JANUARY 25, 2017

Deep learning algorithm does as well as dermatologists in identifying skin cancer

In hopes of creating better access to medical care, Stanford researchers have trained an algorithm to diagnose skin cancer.

BY TAXLOR KUBOTA

It's scary enough making a doctor's appointment to see if a strange mole could be cancerous. Imagine, then, that you were in that situation while also living far away from the nearest doctor, unable to take time off work and unsure you had the money to cover the cost of the visit. In a scenario like this, an option to receive a diagnosis through your smartphone could be lifesaving.

Universal access to health care was on the minds of computer scientists at Stanford when they set out to create an artificially intelligent diagnosis algorithm for skin cancer. They made a database of nearly 130,000 skin disease images and trained their algorithm to visually diagnose potential cancer. From the very first test, it performed with inspiring accuracy.

"We realized it was feasible, not just to do something well, but as well as a human dermatologist," said Sebastian Thrun, an adjunct professor in the Stanford Artificial Intelligence Laboratory. "That's when our thinking changed. That's

A dermatologist uses a dermatoscope, a type of handheld microscope, to look at skin. Computer scientists at Stanford have created an artificially intelligent diagnosis algorithm for skin cancer that matched the performance of board-certified dermatologists. *(Image credit: Matt Young)*

News

Redwood City ready to debut futuristic delivery robots

BOTS | GUEST

Amazon Go, deep learning, and a better retail experience

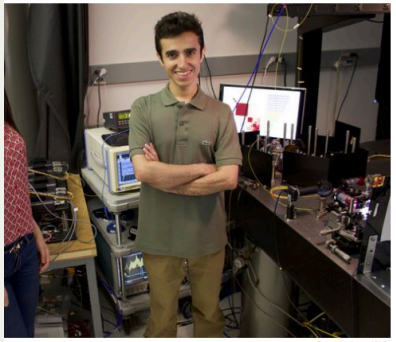
ASHWINI ASOKAN, MAD STREET DEN | JANUARY 12, 2017 2:10 PM

Microscope uses artificial intelligence to find cancer cells more efficiently

Device invented by UCLA professor uses deep learning and photonic time stretch to lion images per second

, 2016

TWEET | SHARE 177 | EMAIL | PRINT



Tunde Akinloye/CNSI

en and Ata Mahjoubfar with the new device, which can image cancer cells without damaging blood samples.

More Science + Technology

Become a citizen scientist and help preserve Californ biodiversity

Experiment resol mystery about wi flows on Jupiter

McLean to receiv American Astronomical Society's Joseph Weber Award

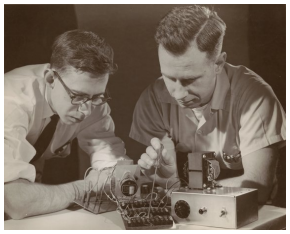
VIEW ALL
Science + Technology

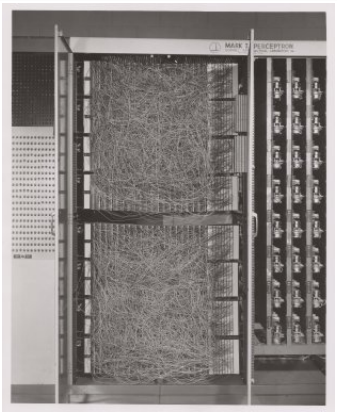
UCLA Top News | Popular

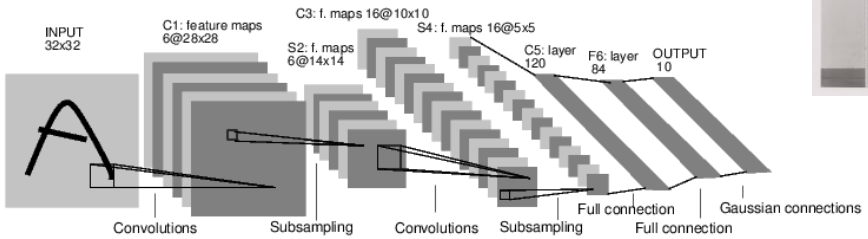
SCIENCE + TECHNOLOGY

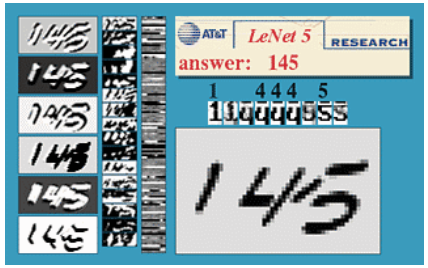
History

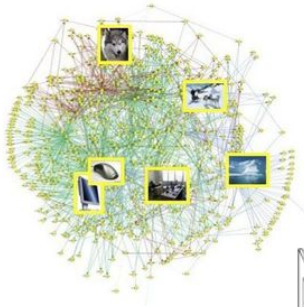
- **1957:** Perceptron
- **1960:** Adaline
- **1969:** Linear models can't learn XOR
- **70's:** not much
- **1980:** Neocognitron (based on V1)
- **80's:** Connectionism; first back propagation studies (modest results)
- **1998:** LeNet (convolutional)
- **2006:** Back propagation improvements
- **2012:** AlexNet



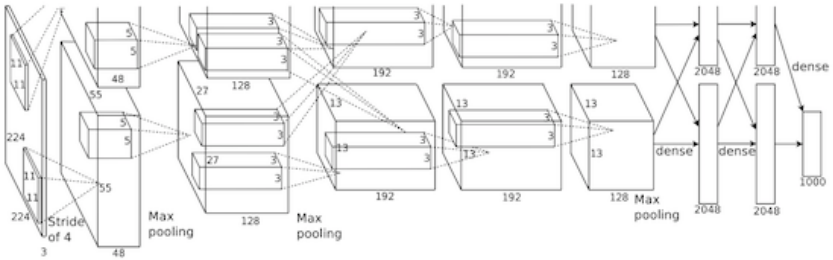




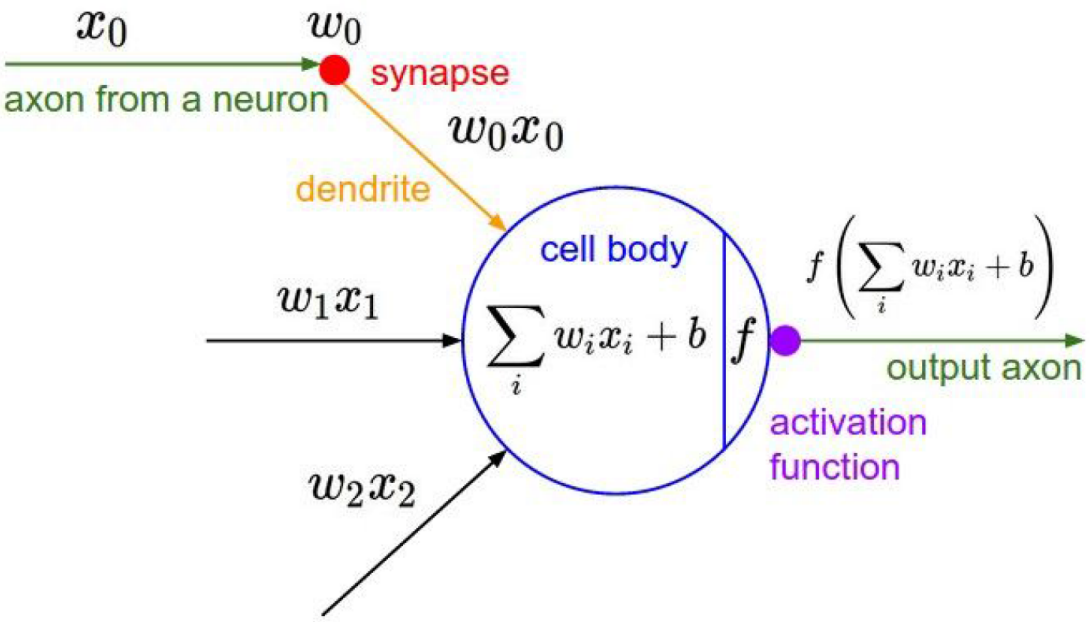


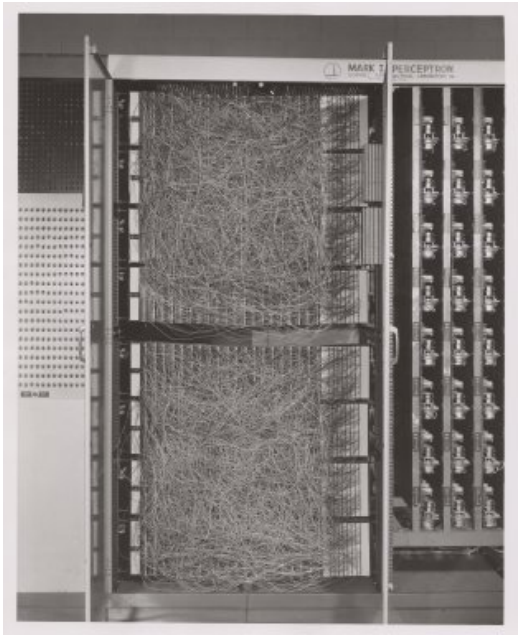


IMAGENET



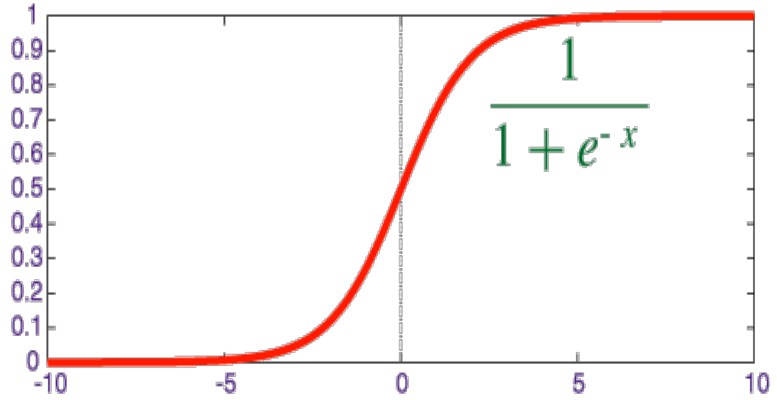
The Perceptron





source: (Andrej, 2016)

- Simple mathematical model of a neuron
- Sigmoid was used to model the a neurons "firing rate"

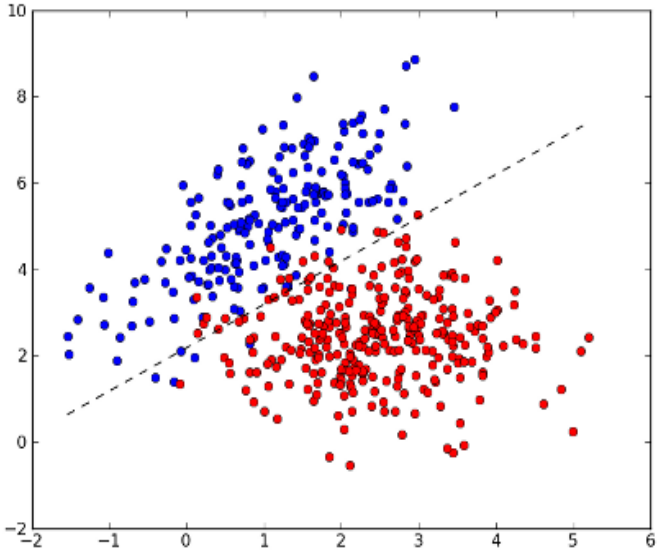


Linear Classifier

- Linear classifier

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

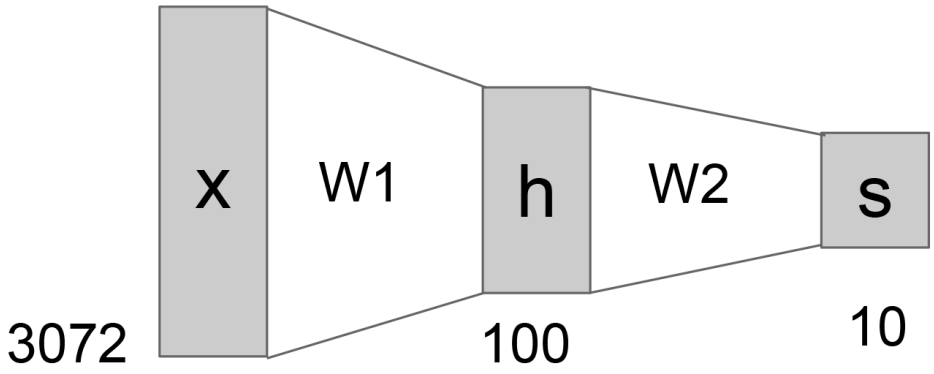
$$f = \sum_i (w_i x_i + b)$$



- x: image
- W: weights
- b: bias, independent from data
- Parametric approach

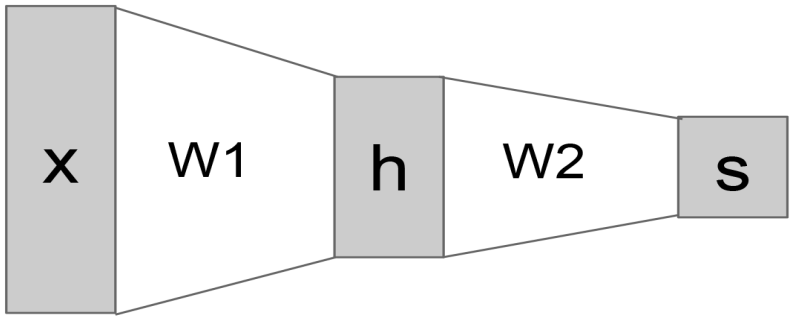
2-Layer Neural Network

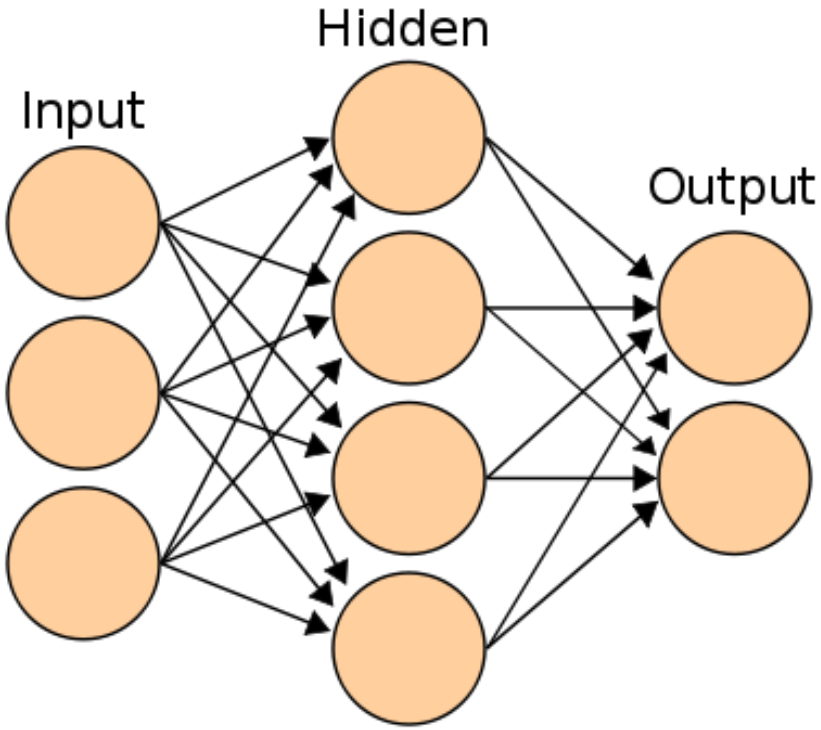
$$f = W_2 max(0, W_1 x)$$



source: (Andrej, 2016)

2-Layer Neural Network





- Neural nets are organized in layers
- Amenable to vectorization
- Can run in parallel
- More efficient

Softmax

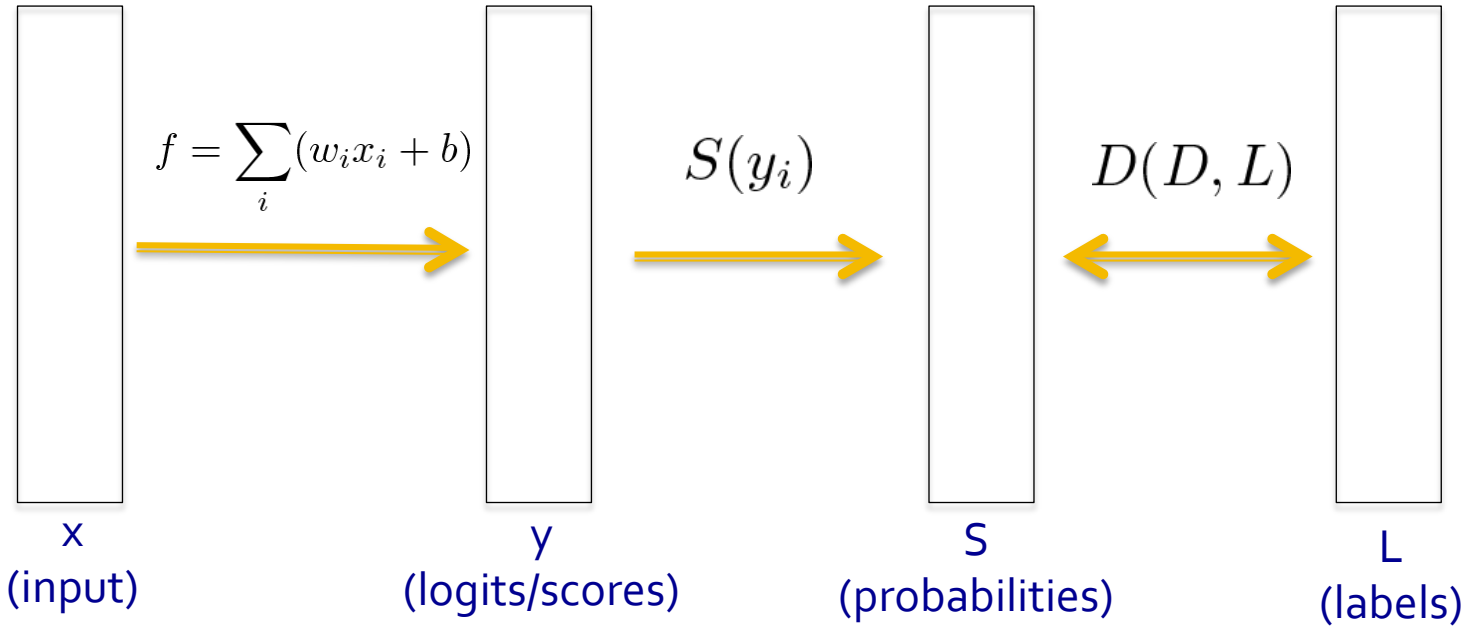
$$S(y_i) = \frac{\exp(y_i)}{\exp(\sum_j \exp(y_j))}$$

- Input: scores from the last layer
- Output: probability distribution over the **j** classes

Loss Function

- Loss functions: compute the distance between classification and training **labels**
- Cross-entropy

$$D(D, L) = -\sum_i L_i log(S_i)$$



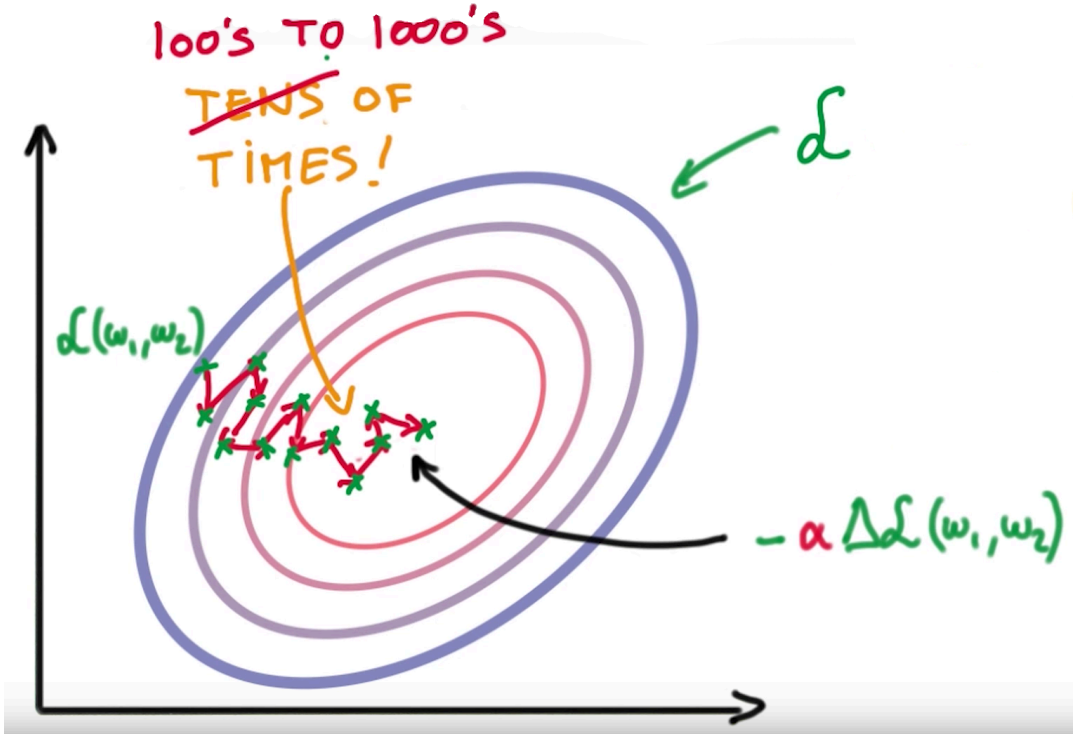
Loss Function & Training

- Loss

our network goes here

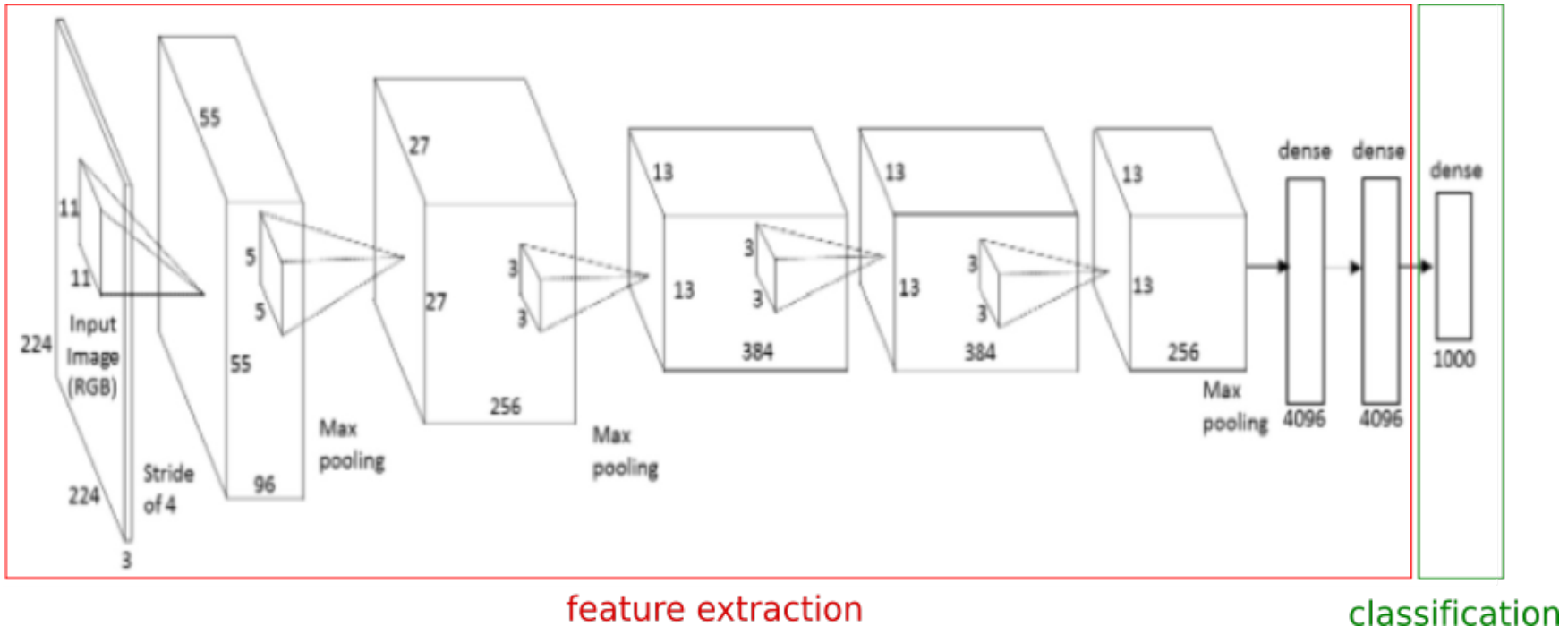
$$\mathcal{L} = \frac{1}{N} \sum_i D(S(Wx_i + b), L_i)$$

- Parameters **W** and **b** can be learned by optimization: **SGD**
- Gradients are computed by **Back Propagation**



source: (Google, 2015)

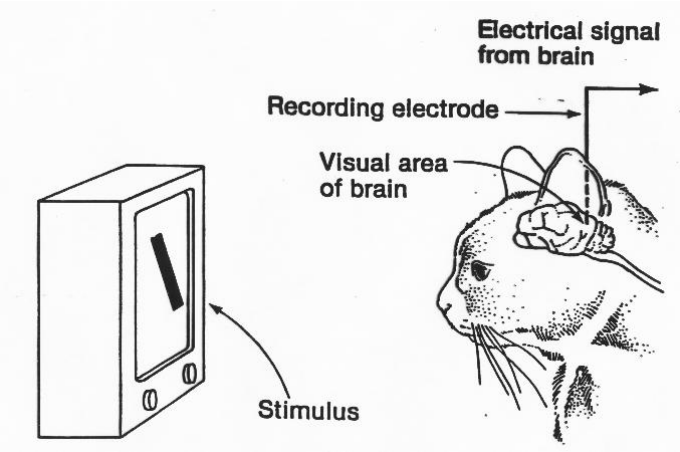
Convolutional Network (ConvNet)

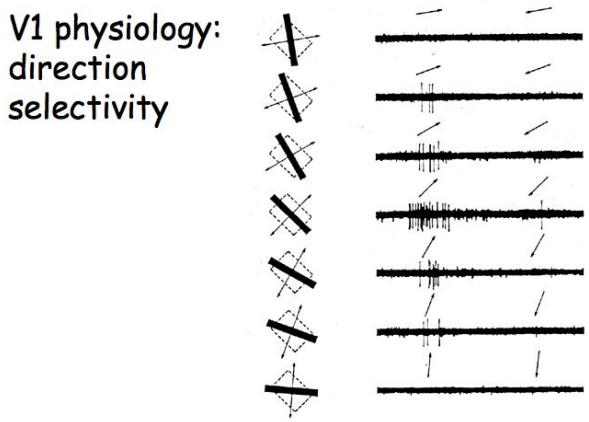


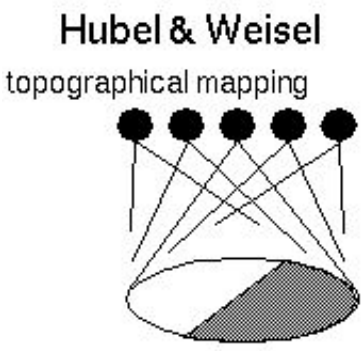
- Computer vision: detection, localization, segmentation
- Convolutional layers + fully connected layers work as a filter bank
  - Representations get more abstract in deeper layers
- Last layer outputs the class scores

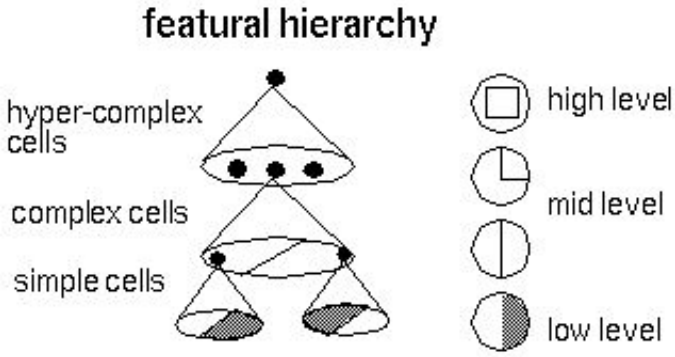
Biological Motivation

- Hubel & Wisel (late 50's)
  - Cat's neurons responded to images projected in specific locations
  - Early visual cortex responded strongly to oriented bars
  - Identified visual cortex hierarchical architecture
  - Nobel prize in 1981



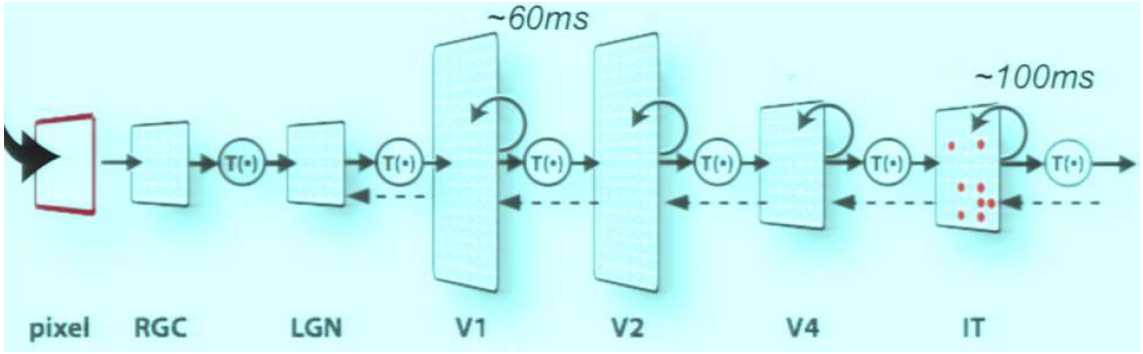


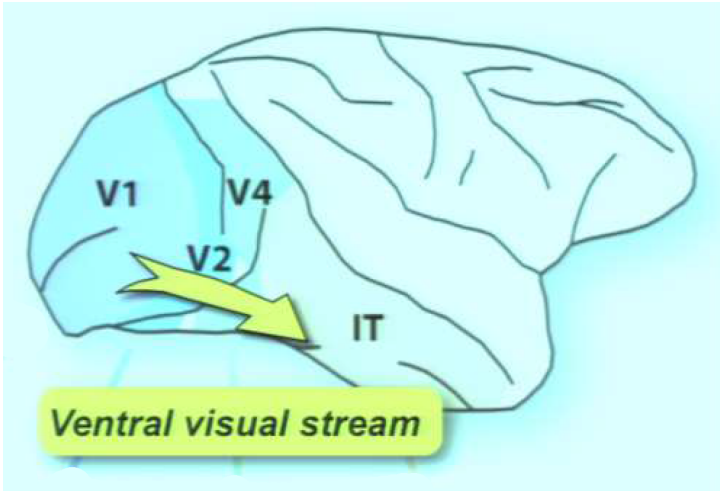




Biological Motivation

- Simplified view of the visual system





source: (DiCarlo,2013)

- RGC and LGN mainly carry signal to V1
- V1 is arranged in a spatial map.
- V1 (in our model) is composed of simple-cells and complex cells.
  - Simple cells: detect spatially localized features
  - Complex cells: detect spatially localized features, somewhat invariant to light changes and shifts in position

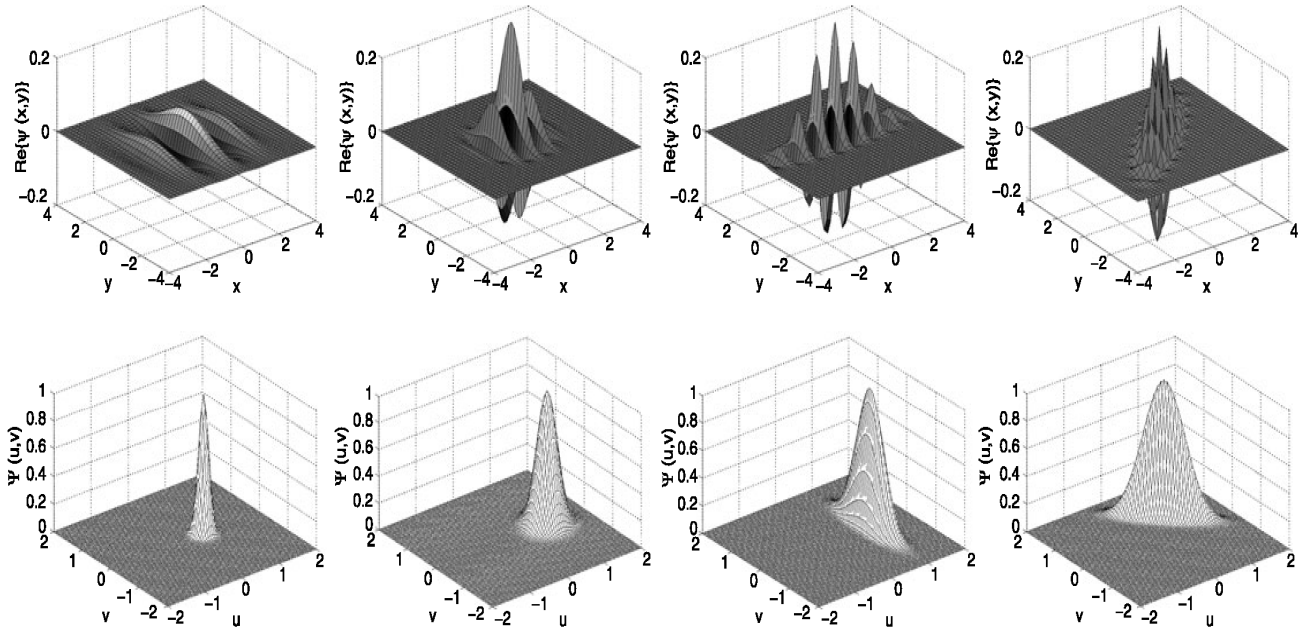
Biological Motivation

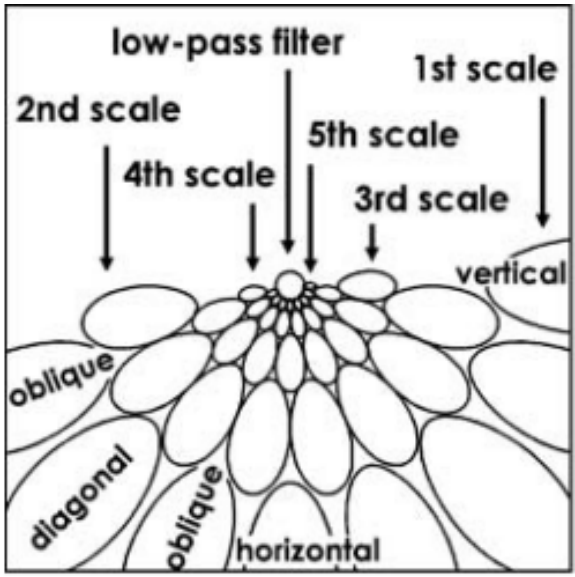
- V1 cell responses can be modeled as Gabor filters

$$w(x, y, \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp(-\beta_x x'^2 - \beta_y y'^2) cos(f x' + \phi)$$

$$x' = (x - x_0)cos(\tau) + (y - y_0)sin(\tau)$$

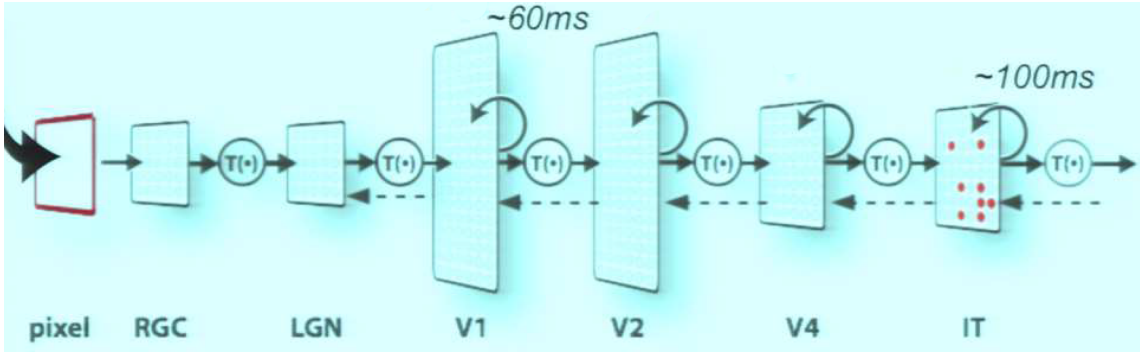
$$y' = -(x - x_0)sin(\tau) + (y - y_0)cos(\tau)$$

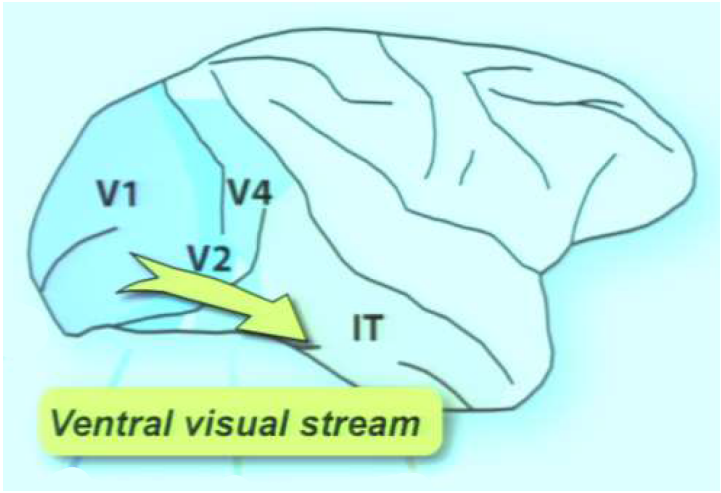




Biological Motivation

- Simplified view of the visual system



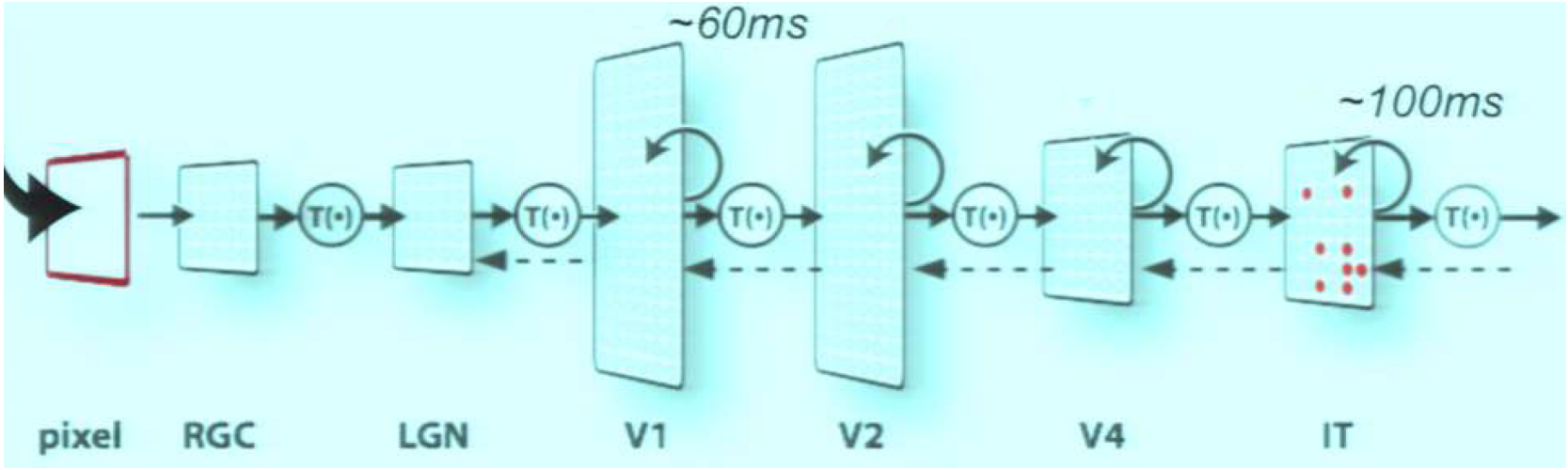


source: (DiCarlo,2013)

- ConvNet are designed to capture 3 V1 properties:
  - Simple cells: ConvNet detectors (kernels) simulate simple cells
  - Complex cells: Poolling units simulate complex cells
  - Spatial mapping: ConvNets also consider input information as 2D maps

Biological Motivation

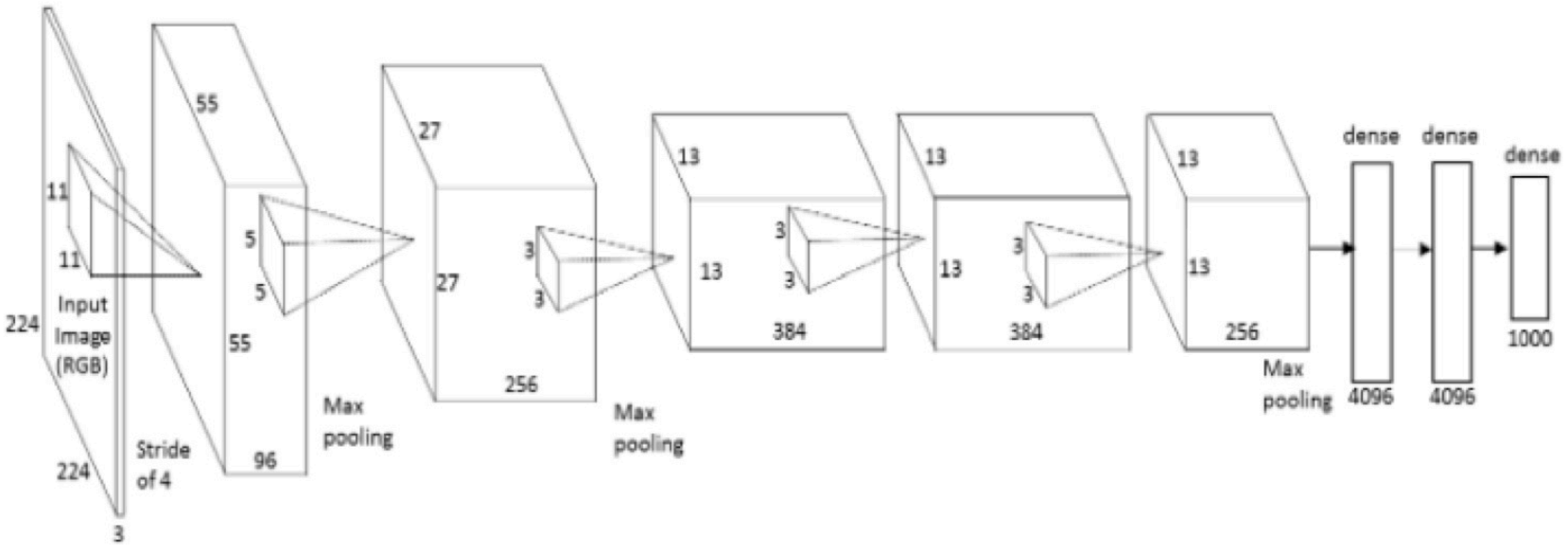
- Simplified view of the visual system



source: (DiCarlo,2013)

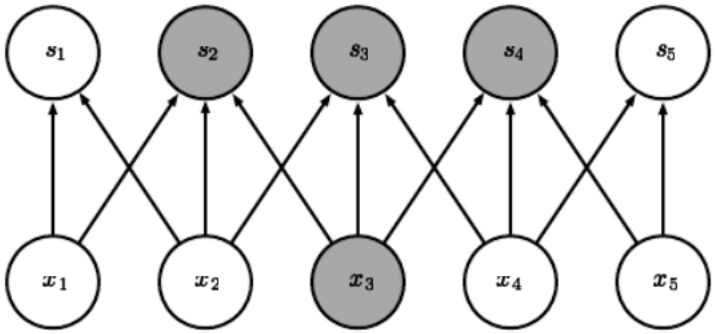
- Convolution + pooling is repeated through the visual cortex layers
- "grandmother cells" fire for specific concepts, located in the medial temporal lobe (in humans).
  - Invariant to input transformations
- Inferotemporal cortex (IT) layer is similar to a ConvNet last layer

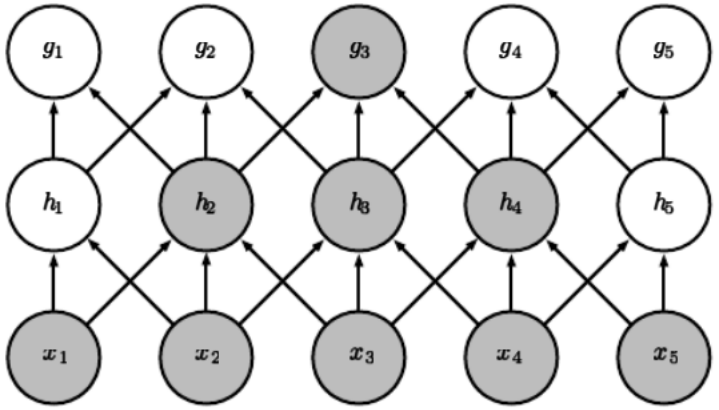
Convolutions



- m(t) = (x*k)(t)
- m = **feature map**
- x = input: multidimensional array (aka **tensor**) of data (image)
- k = **kernel**: multidimensional filter, tuned by the learning algorithm
- Convolution can be seen as a weighted sum

Convolutions





source: (Goodfellow,2016)

- **Receptive field**: portion of a lower layer that affects a neuron on the upper layer
- Parameter sharing and sparse connectivity
  - Less parameters to store (less memory)
  - Smaller number of computations
  - Better statistical efficiency
- Equivariance to translation
  - If the input is translated, the output is translated in the same way

Convolutions

- Activation map size

$$A = [\frac{(N - F)}{S}] + 1$$

N: input size
F: filter size
S: stride

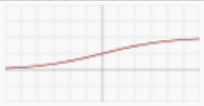
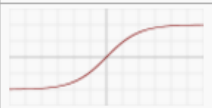
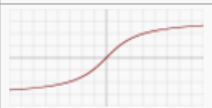
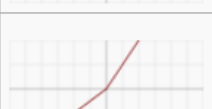
- Padding size

$$P = \frac{(F - 1)}{2}$$

F: filter size

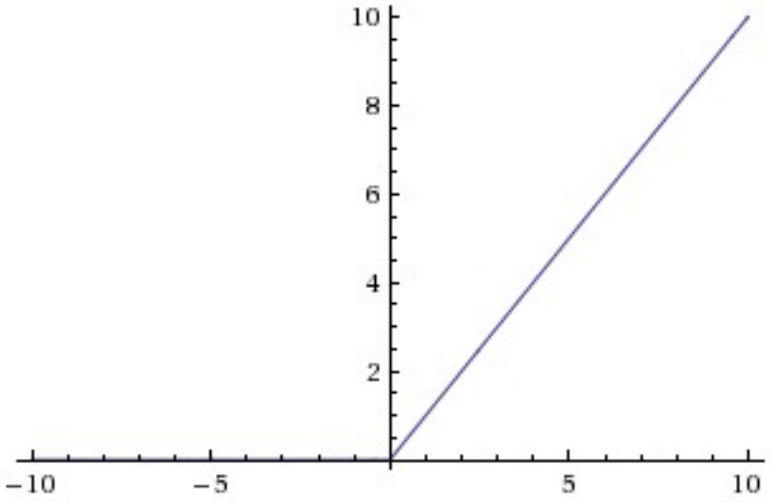
```
def conv_relu(bottom, ks, nout, stride=1, pad=0, group=1,
              param=learned_param,
              weight_filler=dict(type='gaussian', std=0.01),
              bias_filler=dict(type='constant', value=0.1)):
    conv = L.Convolution(bottom, kernel_size=ks, stride=stride,
                         num_output=nout, pad=pad, group=group,
                         param=param, weight_filler=weight_filler,
                         bias_filler=bias_filler)
    return conv, L.ReLU(conv, in_place=True)

n.conv1, n.relu1 = conv_relu(n.data, 11, 96, stride=4, param=param)
```

Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

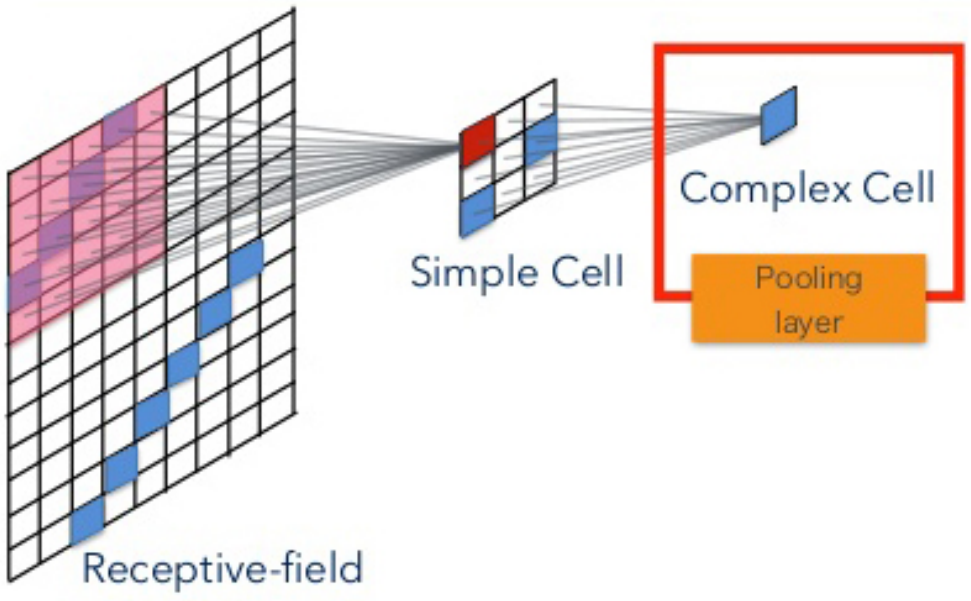
ReLu (Rectified Linear Unit)

- f(x) = max(0,x)



- Nice mathematical properties
  - Easier to optimize
- Final function remains approximately linear
- Large derivatives
- Avoids vanishing gradients
- Not trainable (doesn't have weights)

Pooling



source: (Matsui,2015)

- Computes some kind of summary statistics of the output
- More common: max pooling
- Helps make representation invariant to small translations
- Improves network statistical efficiency
- Not trainable

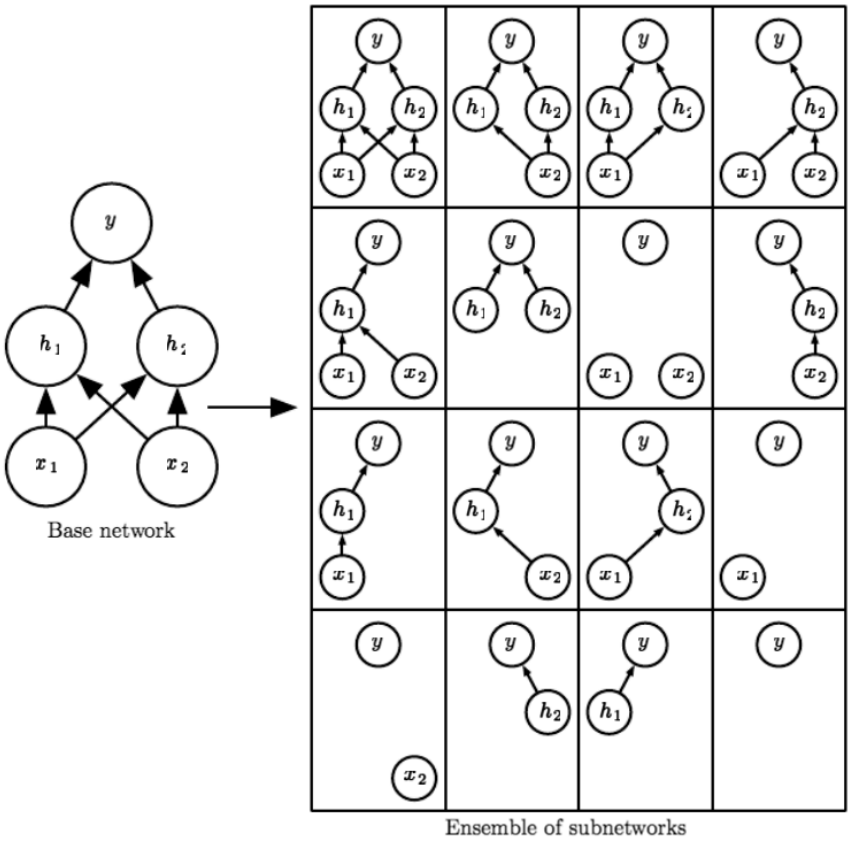
Pooling

- Kernel size
- Stride
- Pooling function

```
def max_pool(bottom, ks, stride=1):
    return L.Pooling(bottom, pool=P.Pooling.MAX, kernel_size=ks, stride=stride)
```

Dropout

- Inexpensive regularization
- Only used in training
- Works by randomly "muting" some connections during training
- Trains ensemble of "subnetworks"
- Computes average of the results
- Not effective for small networks



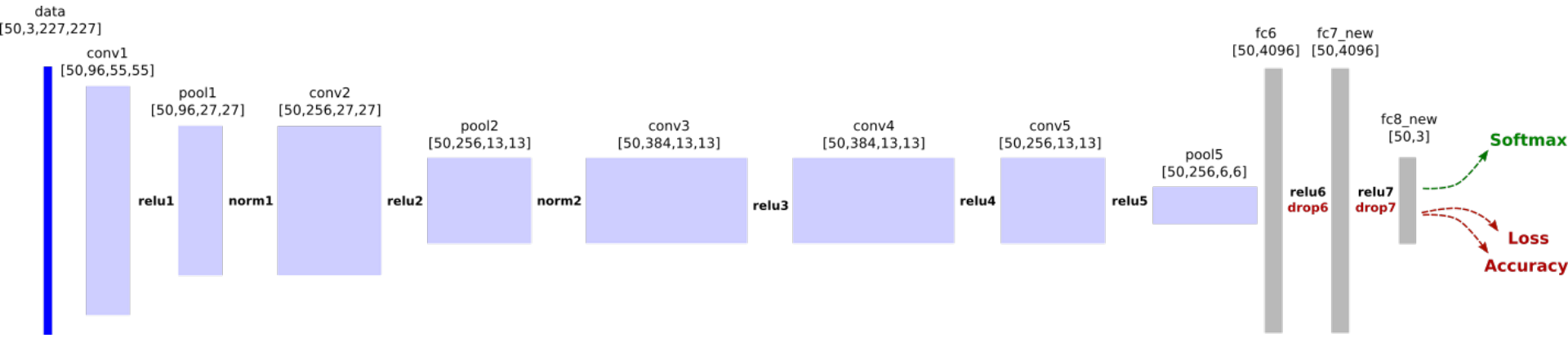
source: (Goodfellow,2016)

```
fc7input = L.Dropout(n.relu6, in_place=True)
```

Architectures

- There's no "best architecture" but several successful ones
- Empirical
- Standard combination o layers:

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
```



References

(Goodfellow,2016): Goodfellow, I.; Bengio, Y.; Couville, A. Deep Learning (book). MIT Press, 2016. (Free online: http://www.deeplearningbook.org/)

(Andrej,2016): Stanford class notes (http://cs231n.stanford.edu/)

(DiCarlo,2013): NIPS 2013 Tutorial (https://www.youtube.com/watch?v=yDvfm7nzIV8)

(Matsui,2015): Presentation slides (http://www.slideshare.net/matsukenbook/deep-learning-chap6-convolutional-neural-net)

(Google,2015): Google's Deep Learning course (https://www.udacity.com/course/deep-learning--ud730)

Caffe: http://caffe.berkeleyvision.org/

Tensorflow: https://www.tensorflow.org/

Torch: http://torch.ch/

Theano: http://deeplearning.net/software/theano/#