Neural Networks:
Deep Learning

Lyle Ungar

Multilevel network: architecture, link functions
CNNs: local receptive fields, max pooling

Regularization: L_2, early stopping, dropout
Gradient Descent (Adagrad again)
Semi-supervised and **transfer learning**
Later: autoencoders, including transformers

All machine learning is optimization

$\hat{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$

$\text{argmin}_{\boldsymbol{\theta}} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|$

So what's new this decade?

- **(Slightly) different loss functions**
- **(Slightly) different optimization methods**
- **GPU instead of CPU**
- **Different, flexible, functional forms for *f***
  - **which require *regularization***

Loss functions

$\hat{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$

$\text{argmin}_{\boldsymbol{\theta}} \, ||\boldsymbol{y} - \hat{\boldsymbol{y}}||$

$||\boldsymbol{y} - \hat{\boldsymbol{y}}||_2$ $||\boldsymbol{y} - \hat{\boldsymbol{y}}||_1$ $||\boldsymbol{y} - \hat{\boldsymbol{y}}||_0$

log-likelihood

hinge, exponential

cross-entropy (KL-divergence)

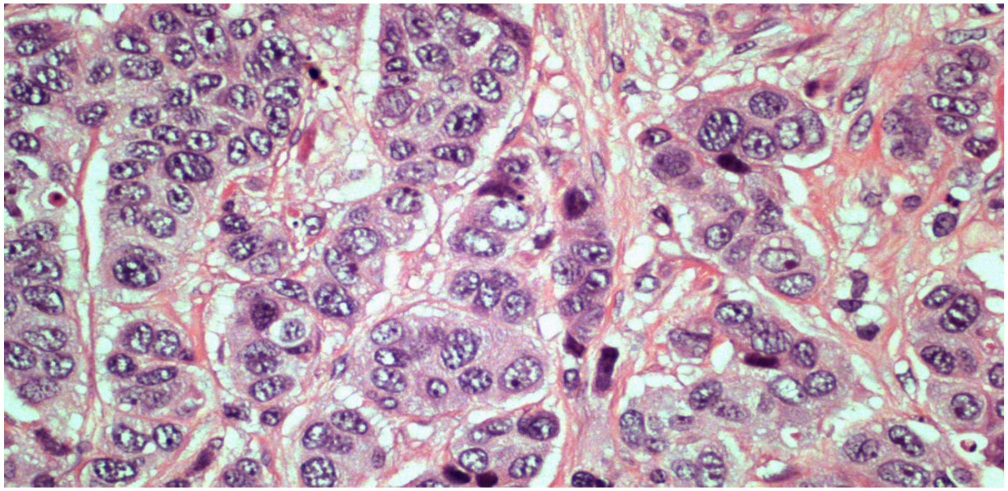
Flexible model forms

$\hat{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$

$\boldsymbol{X}$	$\boldsymbol{y}$
Web page, ad	**Click on ad?**
Past purchases….	**NPV**
Facebook posts	**Age, Sex, Personality, …**

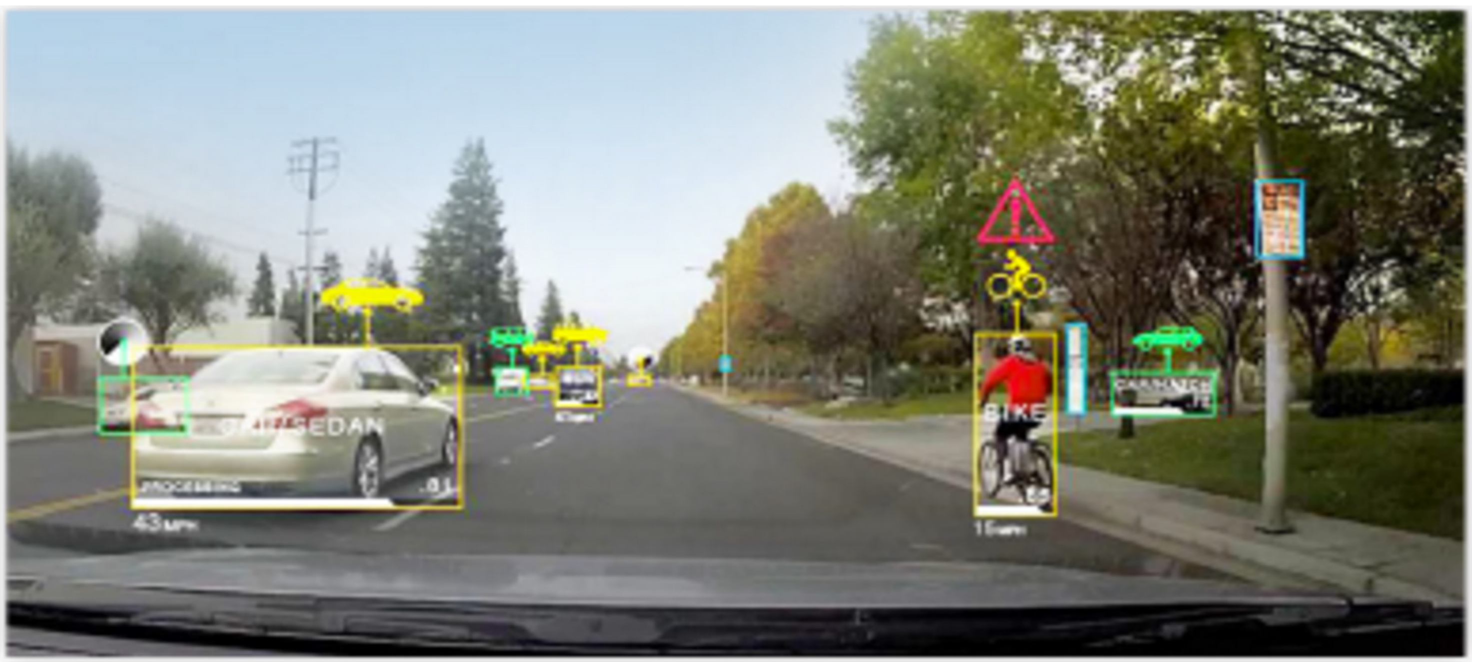
Flexible model forms

X	y
biopsy image	Cancer present?



Flexible model forms

X	**y**
Camera image	**Objects in it**



nvidia

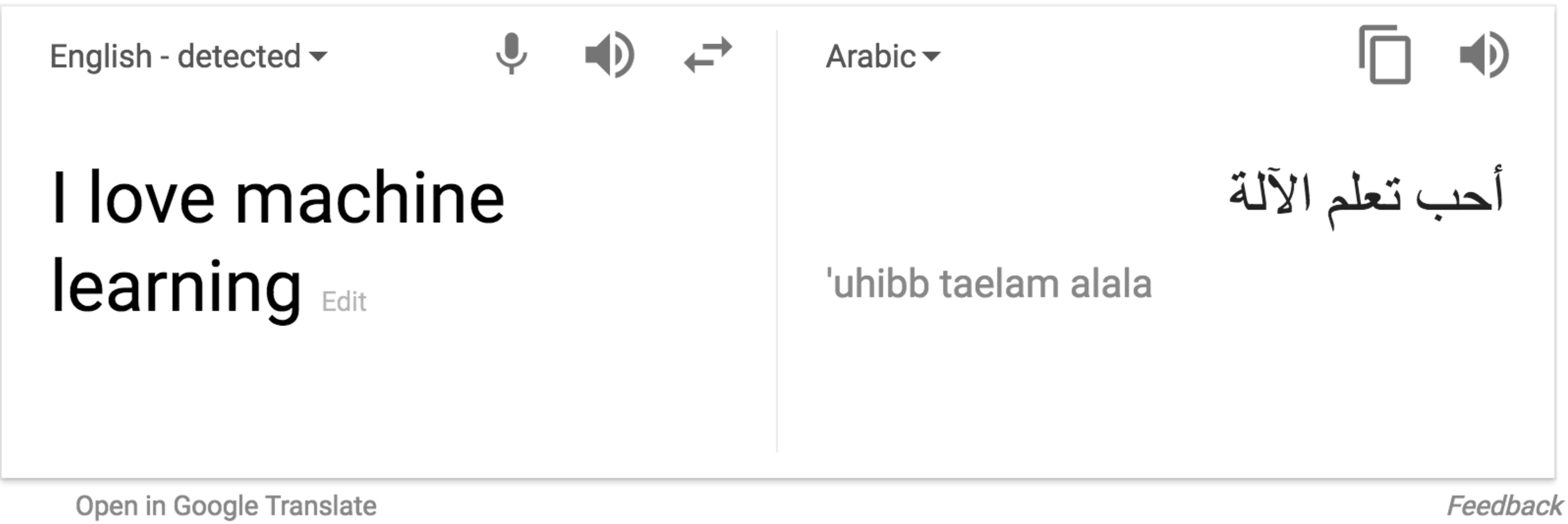
Flexible model forms

X

English sentence

y

Translation



Artificial Neural Nets

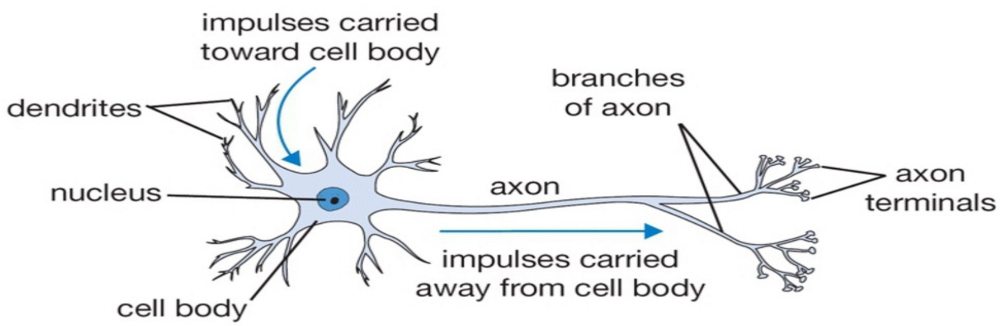
- **Semi-parametric**
  - Flexible model form
- **Used when there are vast amounts of data**
  - Hence popular (again) now
  - But recently with smaller training sets.
- **Deep networks**
  - Idea: representation should have *many* different levels of abstraction

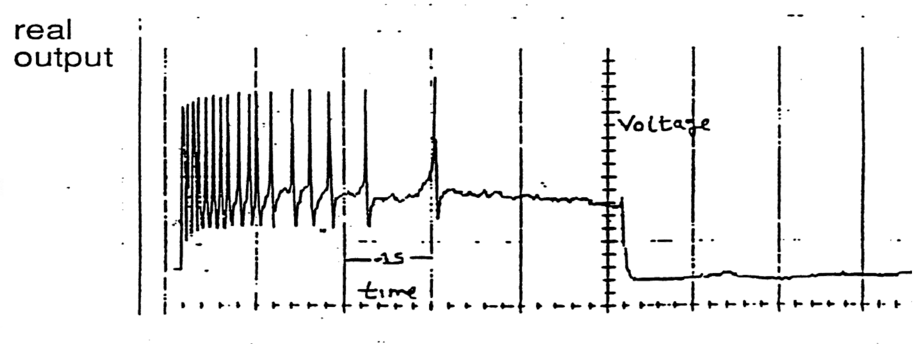
Neural Nets can be

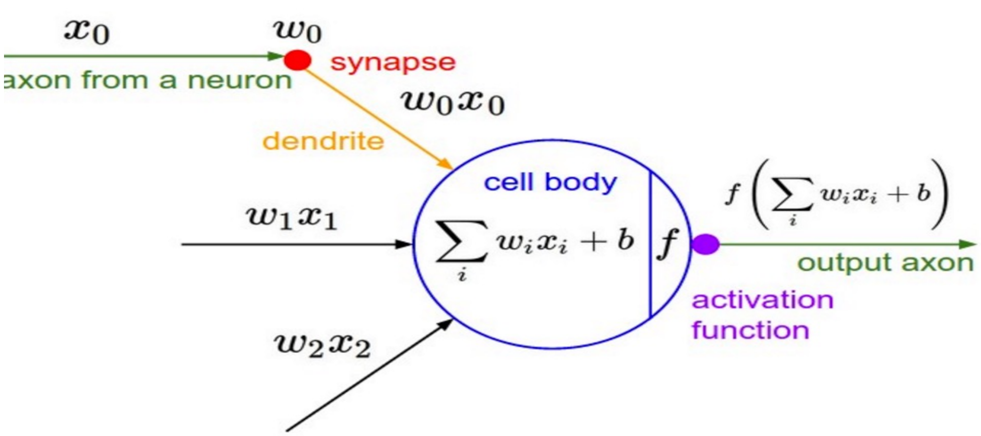
- **Supervised**
  - Generalizes *logistic regression* to a semi-parametric form
- **Unsupervised**
  - Generalizes *PCA* to a semi-parametric form
- **Adversarial (GANS)**
- **Semi-supervised**
- **Reinforcement**

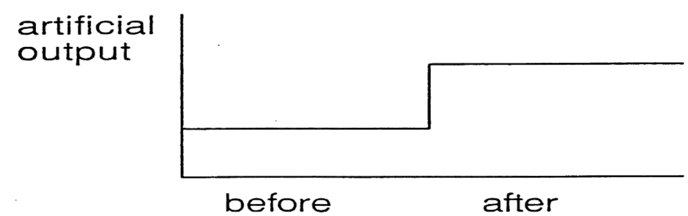
Neural nets often have built-in structure

“Real” and Artificial neuron







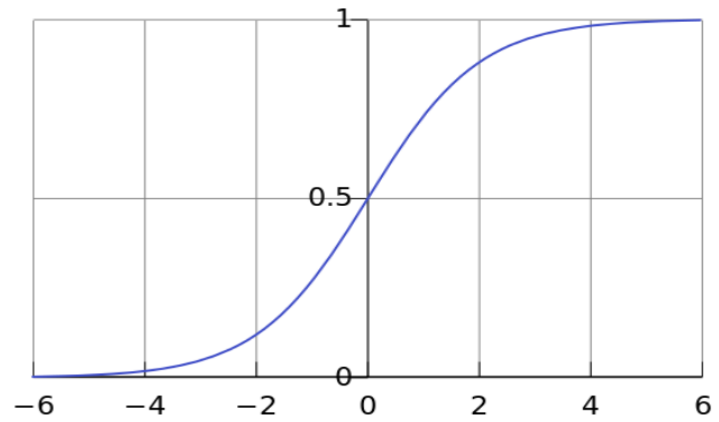


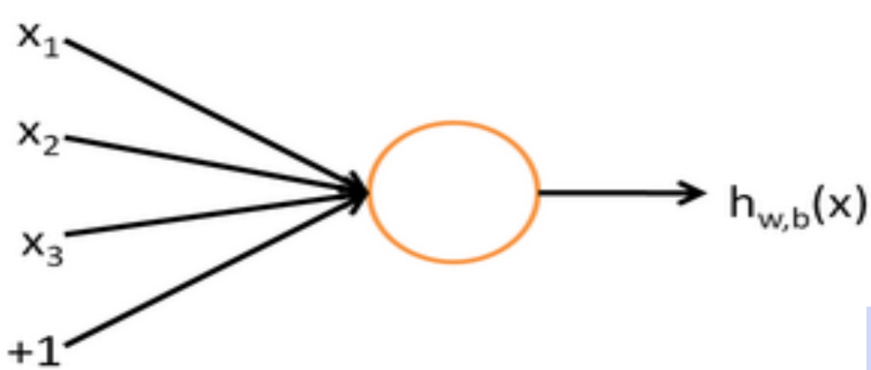
http://cs231n.github.io/neural-networks-1/

One neuron does logistic regression

$$h_{w,b}(x) = f(w^{\mathsf{T}}x + b)$$

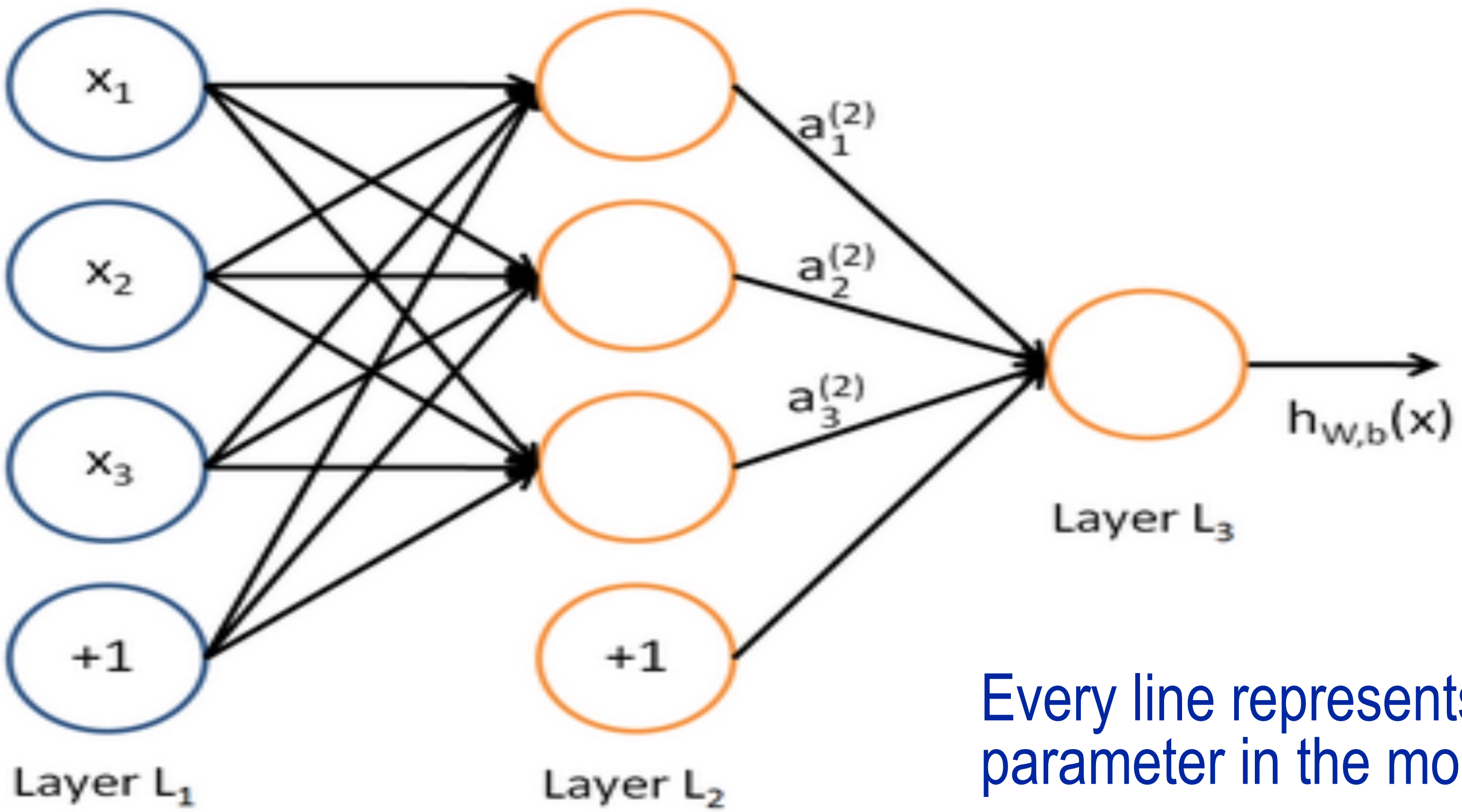
$$f(z) = \frac{1}{1 + e^{-z}}$$





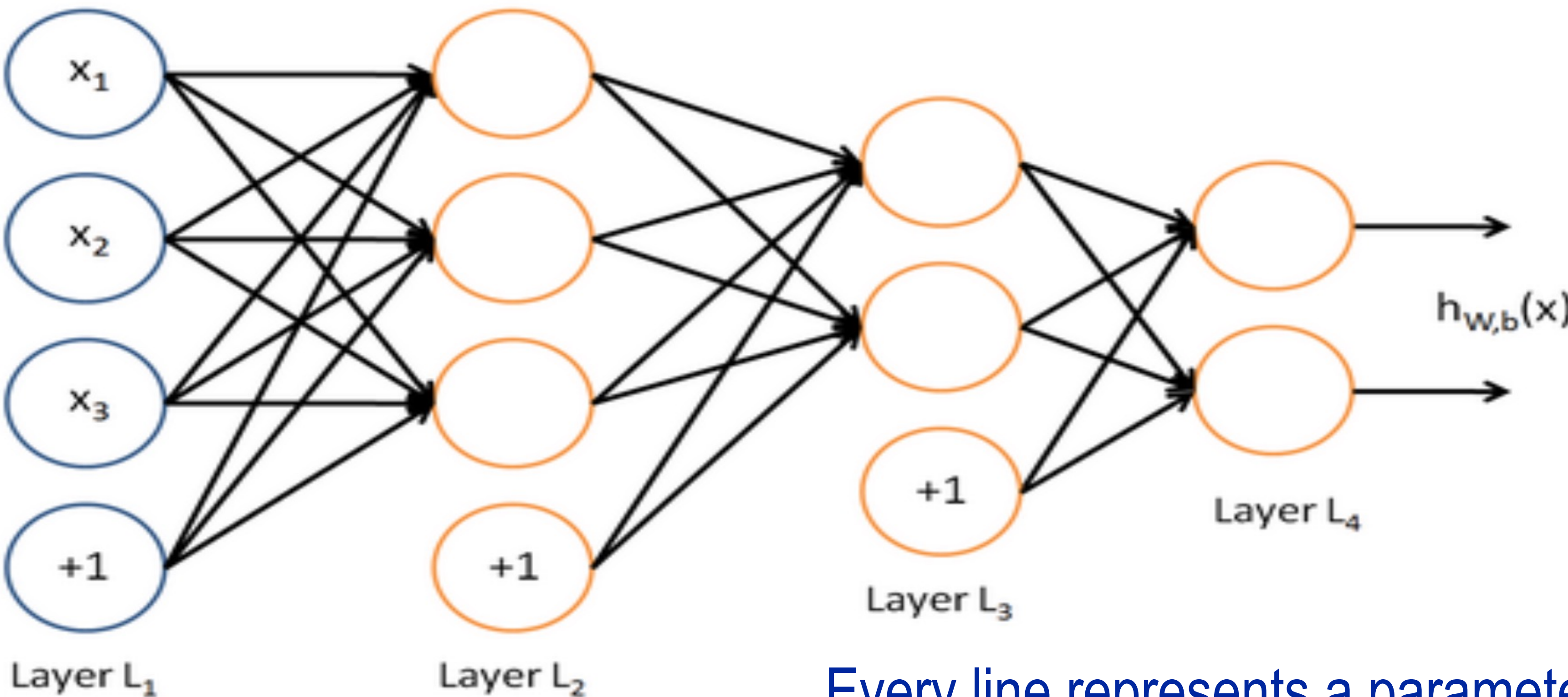
Socher and Manning tutorial

Neural nets stack logistic regressions



Every line represents a parameter in the model

Neural nets stack logistic regressions



Every line represents a parameter in the model

Training

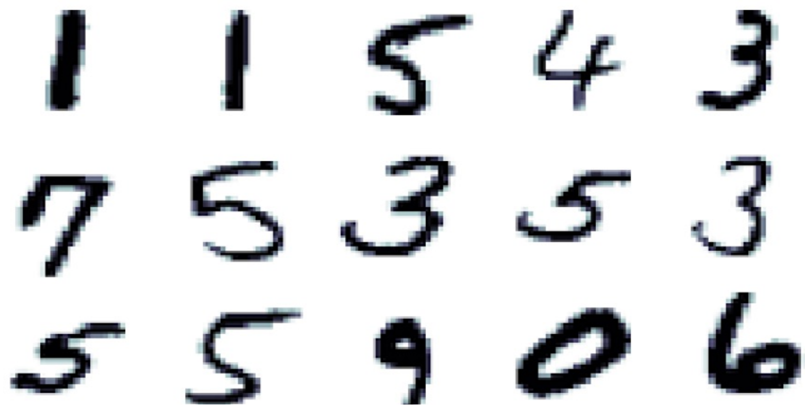
- Mini-batch gradient descent
- “Backpropagate” error derivatives through the model = chain rule

ANNs do pattern recognition

- **Map input “percepts” to output categories or actions**
  - Image of an object $\rightarrow$ what it is
  - Image of a person $\rightarrow$ who it is
  - Picture $\rightarrow$ caption describing it
  - Board position $\rightarrow$ probability of winning
  - A word $\rightarrow$ the sound of saying it
  - Sound of a word $\rightarrow$ the word
  - Sequence of words in English $\rightarrow$ their Chinese translation

MNIST

- Classify 28x28 images of handwritten digits
- **Train:** 50,000
- **Test:** 10,000



Error (%)	Method	Reference
5.0	KNN	Lecun et al. (1998)
3.6	1k RBF + linear classifier	Lecun et al. (1998)
1.6	2-layer NN	Simard et al. (2003)
1.53	boosted stumps	Kegl et al. (2009)
1.4	SVM	Lecun et al. (1998)
0.79	DNN	Srivastava (2013)
0.45	conv-DNN	Goodfellow et al. (2013)
0.21	conv-DNN	Wi et al. (2013)

Street View House Numbers

- Classify 32x32 color images of digits
- Digits taken from housenumbers in Google Street View
- **Train:** 604,388
- **Test:** 26,032

Error (%)	Method	Reference
36.7	WDCH	Netzer et al. (2011)
15	HOG	Netzer et al. (2011)
9.4	KNN	Netzer et al. (2011)
2.47	conv-DNN	Goodfellow et al. (2013)
2	Human	Netzer et al. (2013)
1.92	conv-DNN	Lee et al. (2015)

CIFAR-100

- Classify 32x32 color images into 100 classes
- Images taken from TinyImages dataset at MIT
- **Train:** 50,000
- **Test:** 10,000

Error (%)	Method	Reference
43.77	SVM	Jia et al. (2012)
39.20	OMP	Lin and Kung (2014)
38.57	conv-DNN	Goodfellow et al. (2013)
36.18	DNN	Srivastava and Alakhutdinov (2015)
34.57	conv-DNN	Lee et al. (2015)

ImageNet Classification with Deep Convolutional Neural Networks

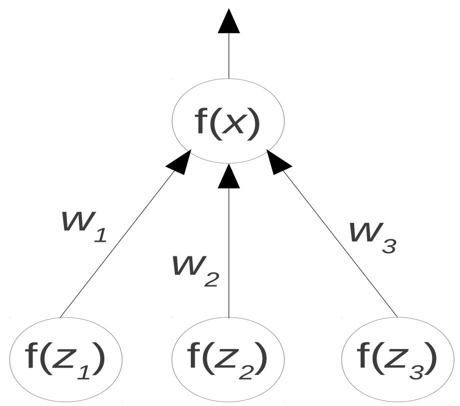
Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada

"AlexNet" 2012

Neural networks

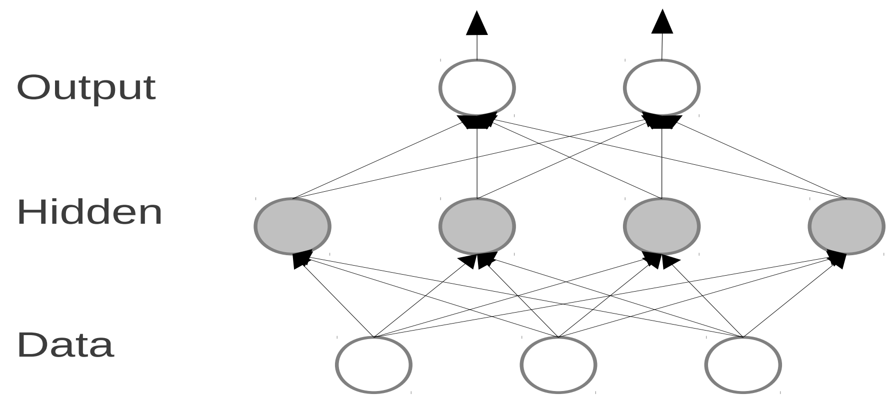
- A neuron



$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

x is called the total input to the neuron, and f(x) is its output

- A neural network

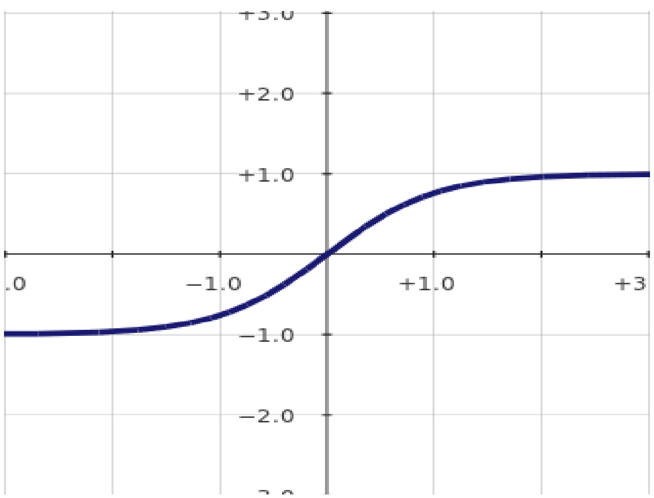


A neural network computes a differentiable function of its input. For example, ours computes: p(label | an input image)

Neurons

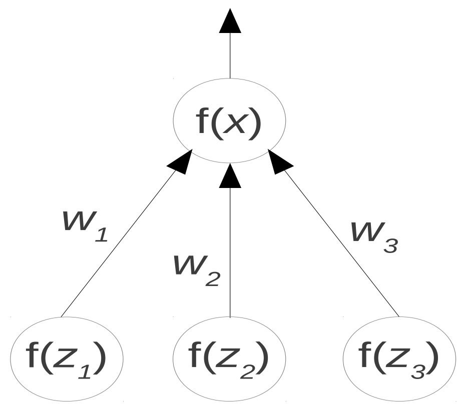
Traditional: sigmoidal e.g. logistic function

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



Hyperbolic tangent

Very bad (slow to train)

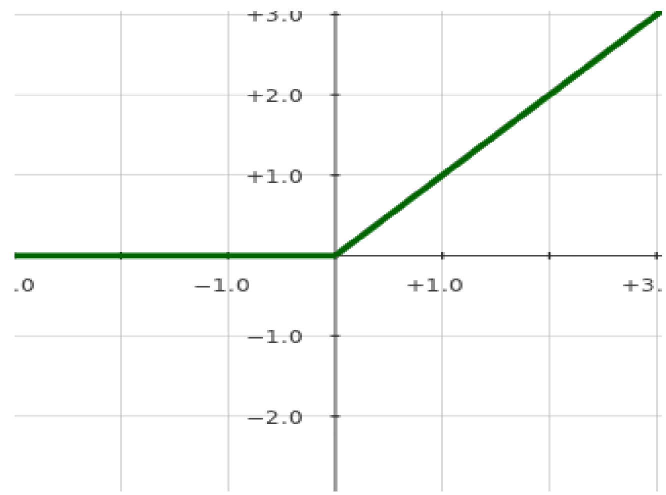


$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

x is called the total input to the neuron, and $f(x)$ is its output

But one can use any nonlinear function

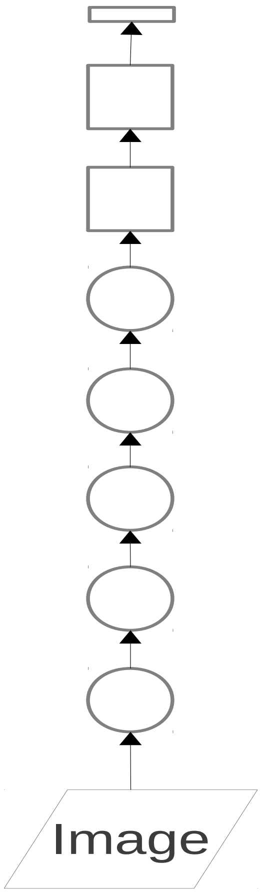
$f(x) = \max(0, x)$



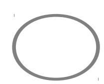
Rectified Linear Unit (ReLU)

Very good (quick to train)

Overview of our model



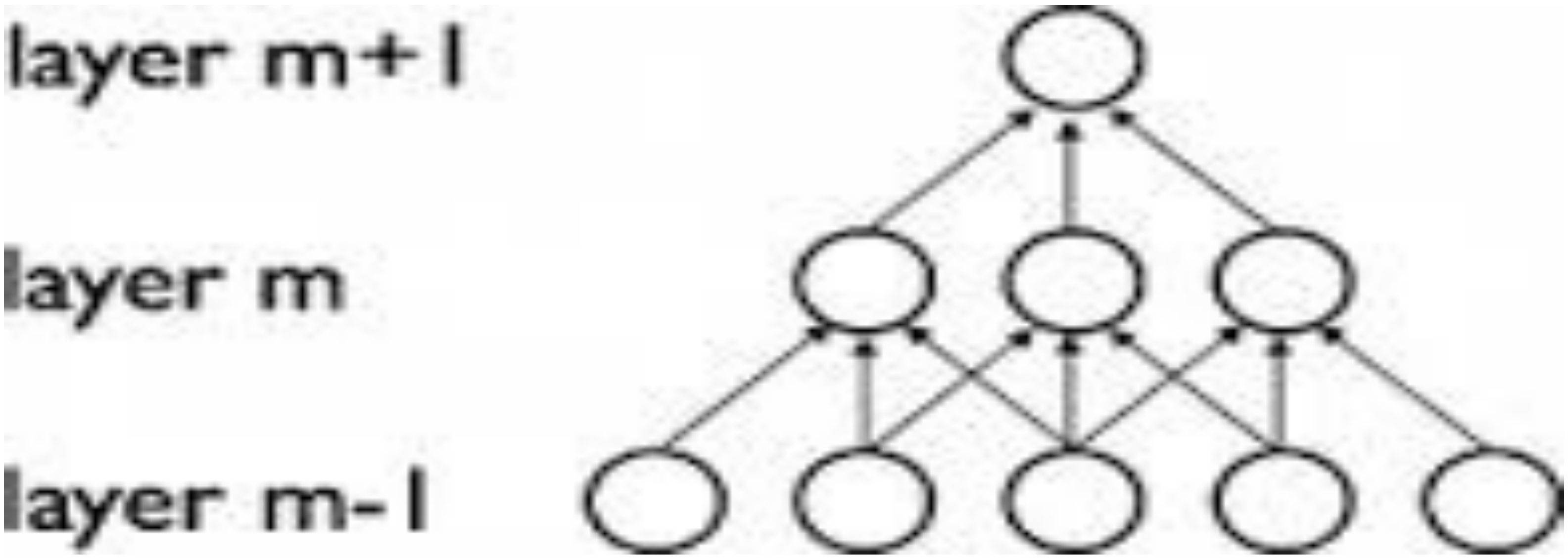
- **Deep**: 7 hidden “weight” layers
- **Learned**: all feature extractors initialized at white Gaussian noise and learned from the data
- Entirely supervised
- **More data = good**



Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

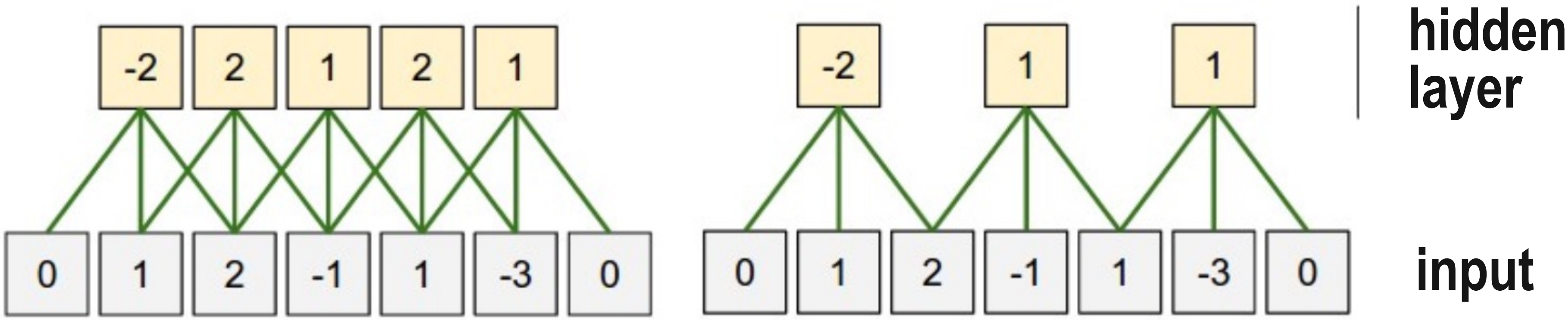
Local receptive fields



Input *x*

In vision, a neuron may only get inputs from a limited set of “nearby” neurons

Local receptive fields

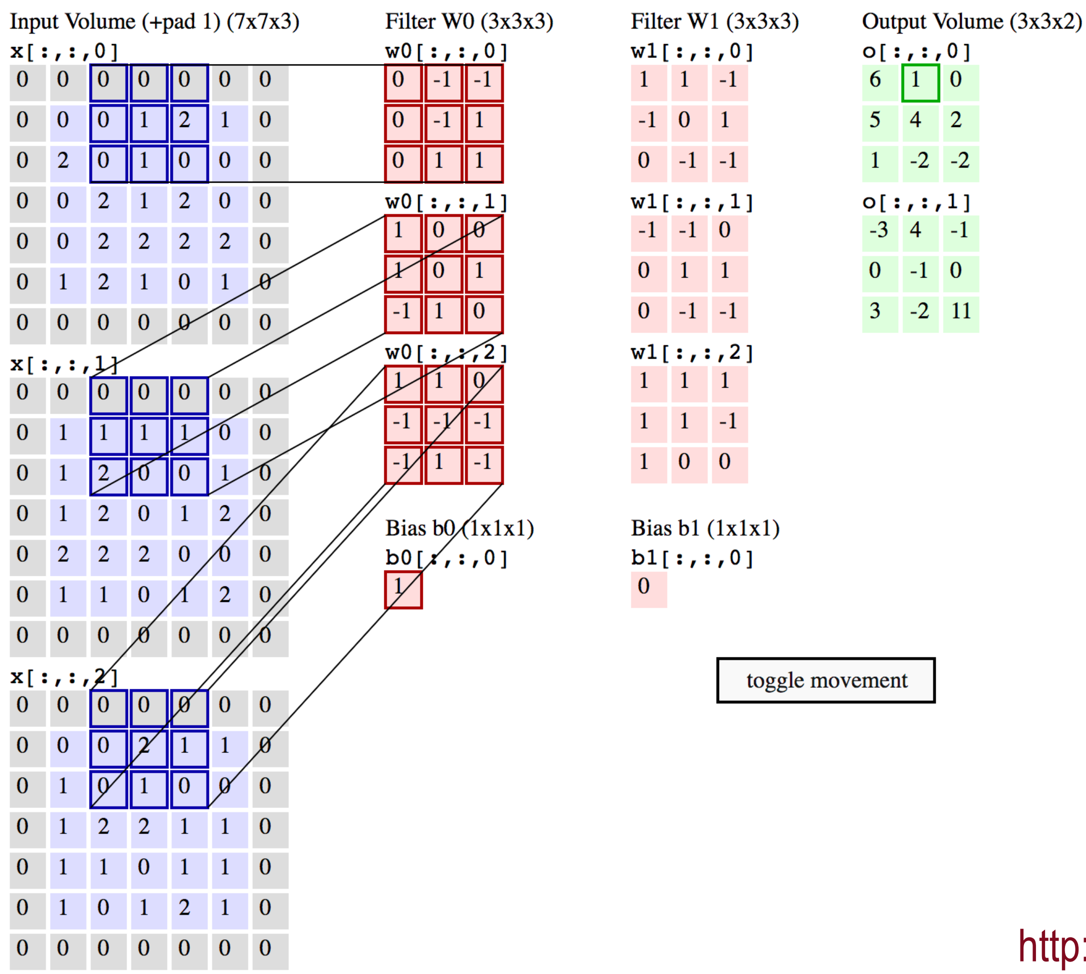


•spatial extent F = 3
•stride S =1

•spatial extent F = 3
•stride S = 2

http://cs231n.github.io/convolutional-networks/

Local receptive fields



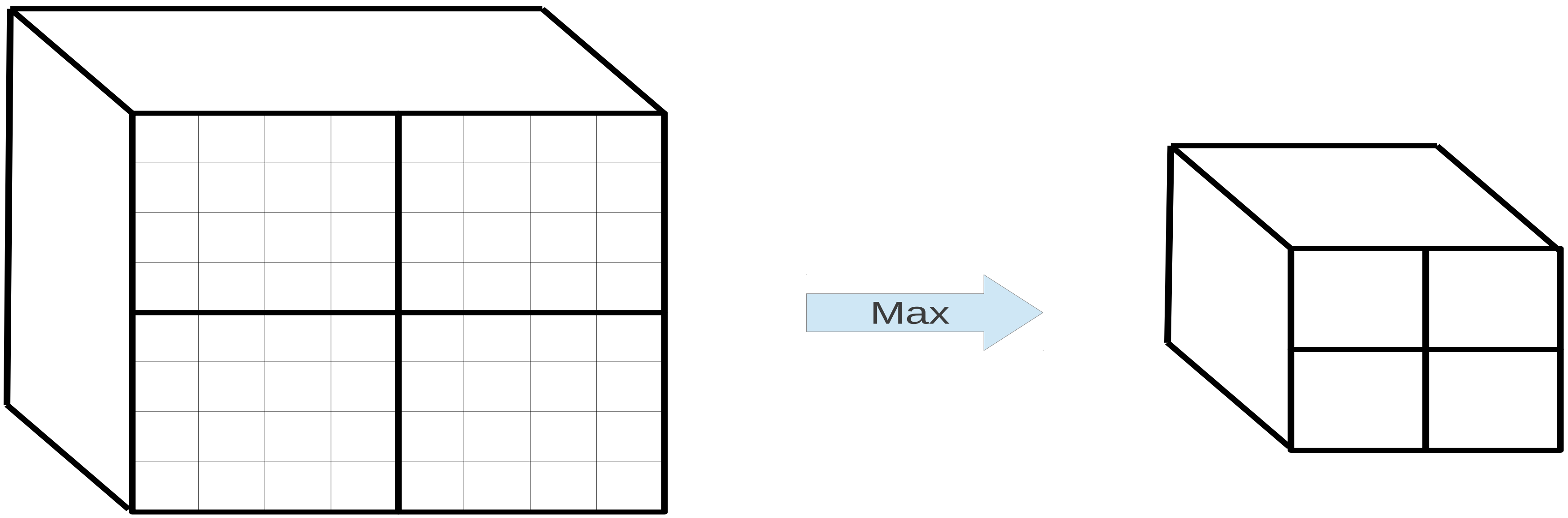
http://cs231n.github.io/convolutional-networks/

Quick tensor background

- **What's a tensor?**
  - As in "tensorflow"
  - Or "Tensor Processing Unit" (TPU)
  - As in the basic data structure in pytorch
    - Aside: there is a worksheet with more than you need to know about pytorch

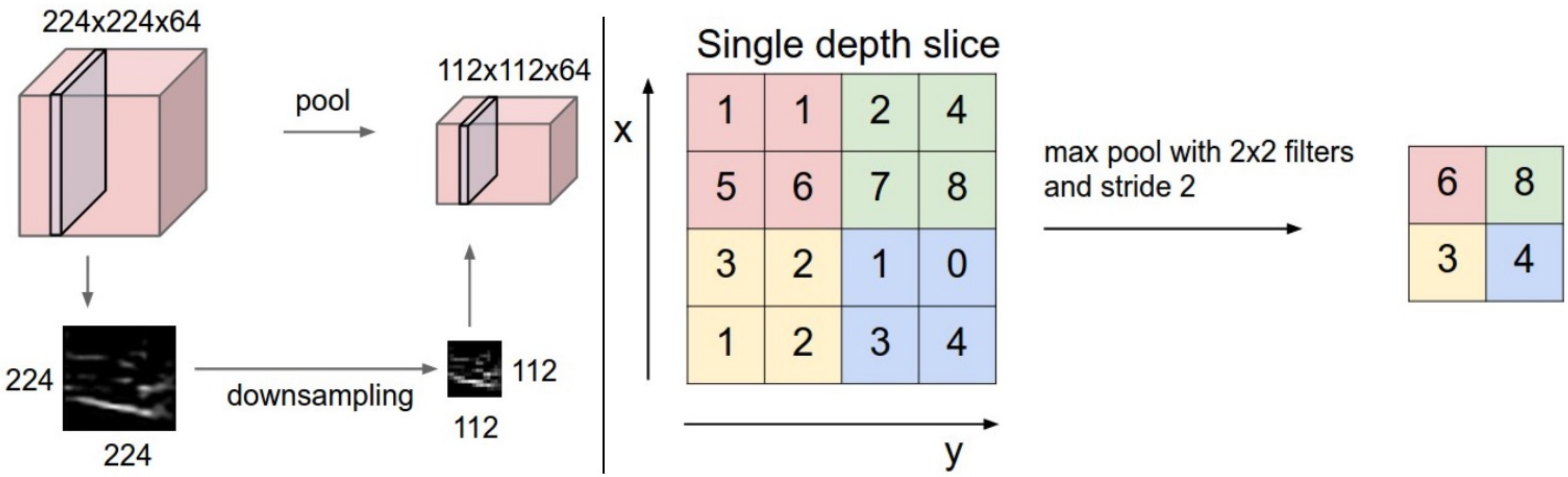
Local pooling

Max-pooling partitions the input image into local regions and outputs the maximum value for each**.**



Reduces the computational complexity
Provides translation invariance.

Max pooling

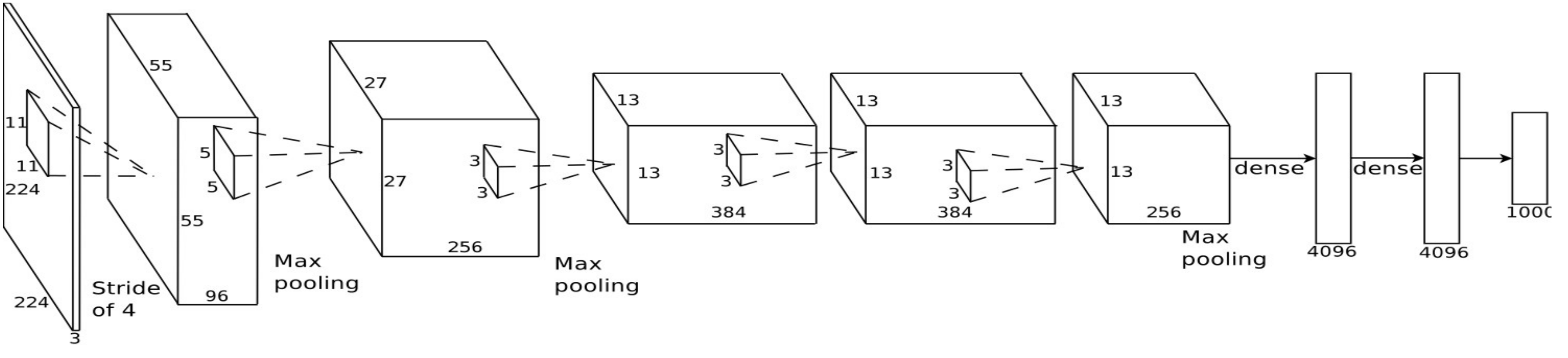


- *Spatial extent F=2*
- *Stride S=2*

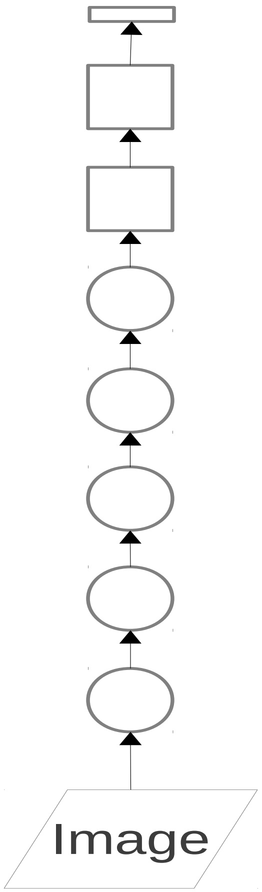
http://cs231n.github.io/convolutional-networks/

Our model

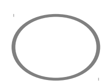
- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000



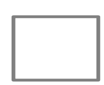
Overview of our model



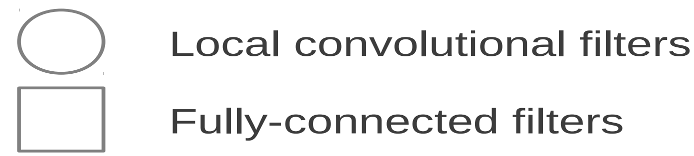
- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer:** 4096-dimensional



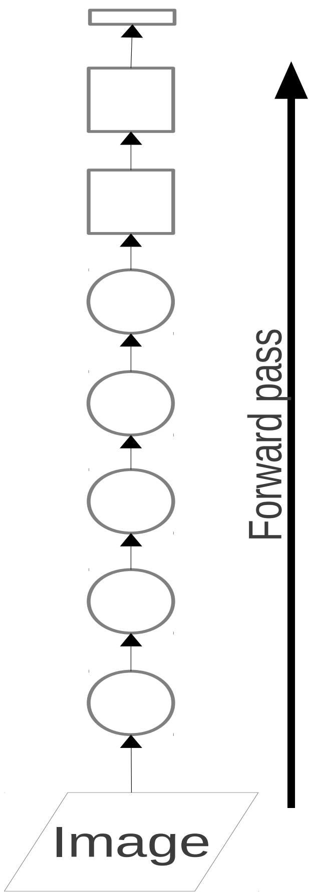
Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity



Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity



Training



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

One output unit per class

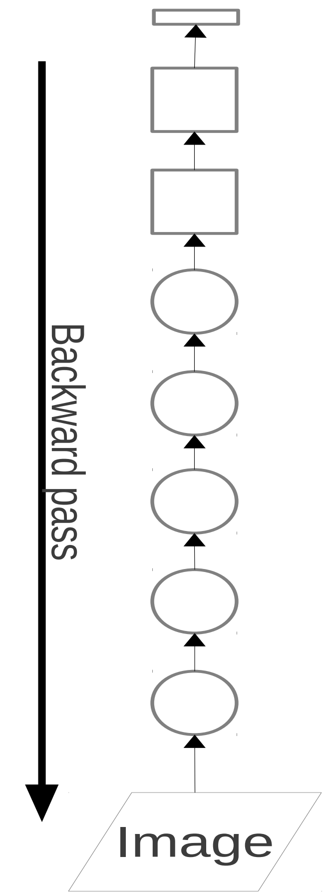
x_i = total input to output unit i

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

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

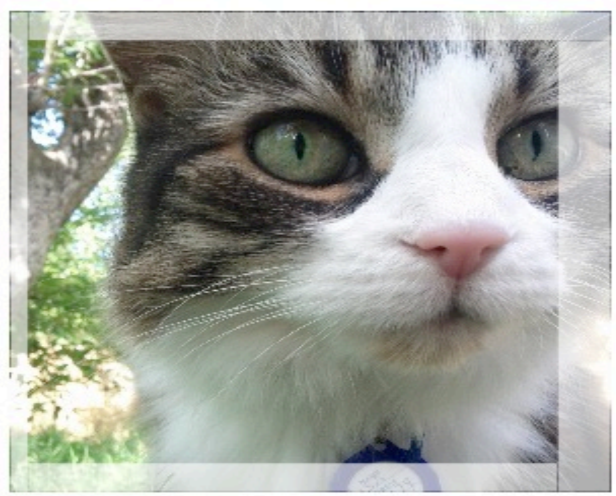
$f(\mathbf{x}_i)$ = **softmax**

$\log(f(\mathbf{x}_t))$ = **cross-entropy**



Data augmentation

- Our neural net has 60M real-valued parameters and 650,000 neurons
- It overfits a lot. Therefore we train on 224x224 patches extracted randomly from 256x256 images, and also their horizontal reflections.



Take advantage of invariances

- **Build into the model (if possible)**
  - The same feature detectors can be used anywhere in the image
- **Use to augment the data**
  - The label doesn't change under mild translation
  - Or under reflection
- **Build into the loss function (if all else fails)**
  - Make the chatbot avoid repetition, or give longer answers or …

"Inductive Bias"