All machine learning is optimization

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

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

So what's new?

- (Slightly) different loss functions
- (Slightly) different optimization methods
- CPU/GPU; SAAS/full stack
- Different, flexible, functional forms for *f*
  - which require *regularization*

Loss functions

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

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

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

log-likelihood

logistic/softmax

hinge

exponential

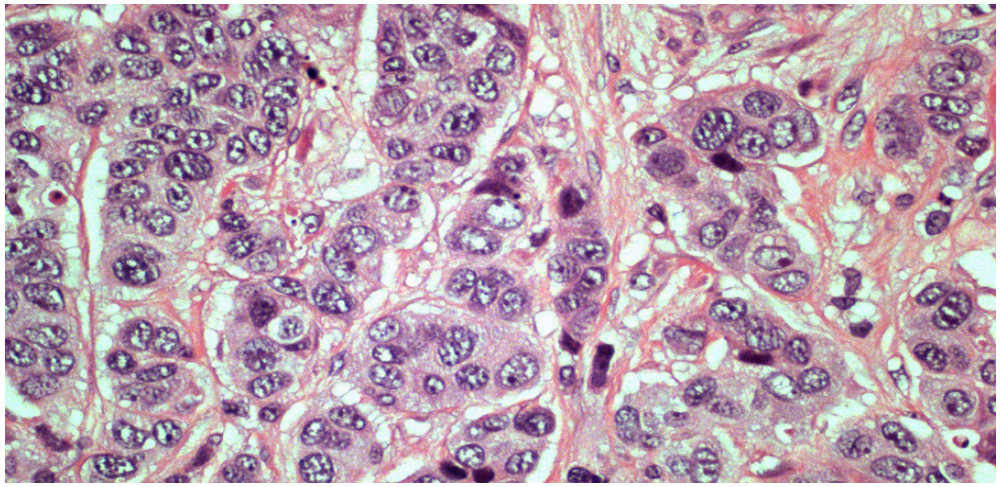
Flexible model forms

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

X	y
Web page, ad	Click on ad?
Past purchases….	NPV
Facebook posts	Age, Sex, Personality, …

Flexible model forms

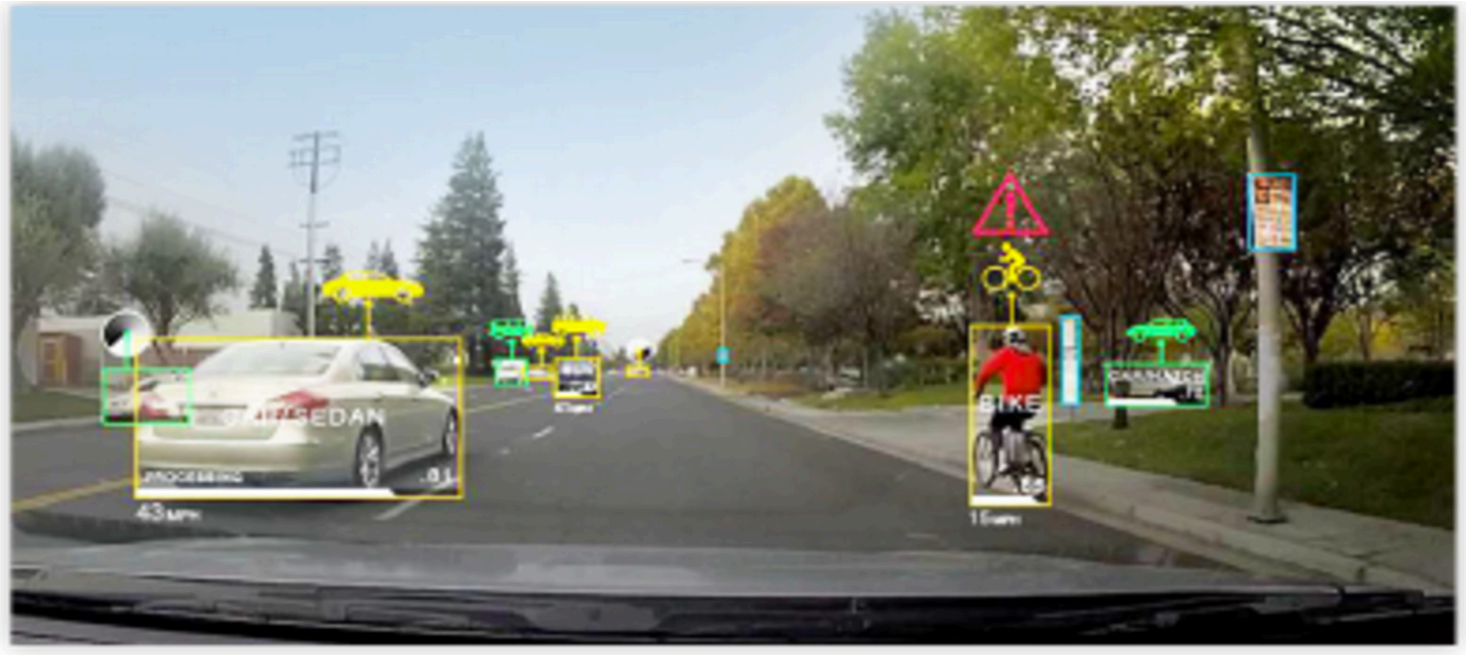
X	y
biopsy image	Cancer present?



Flexible model forms

X **y**

Camera image **Objects in it**



nvidia

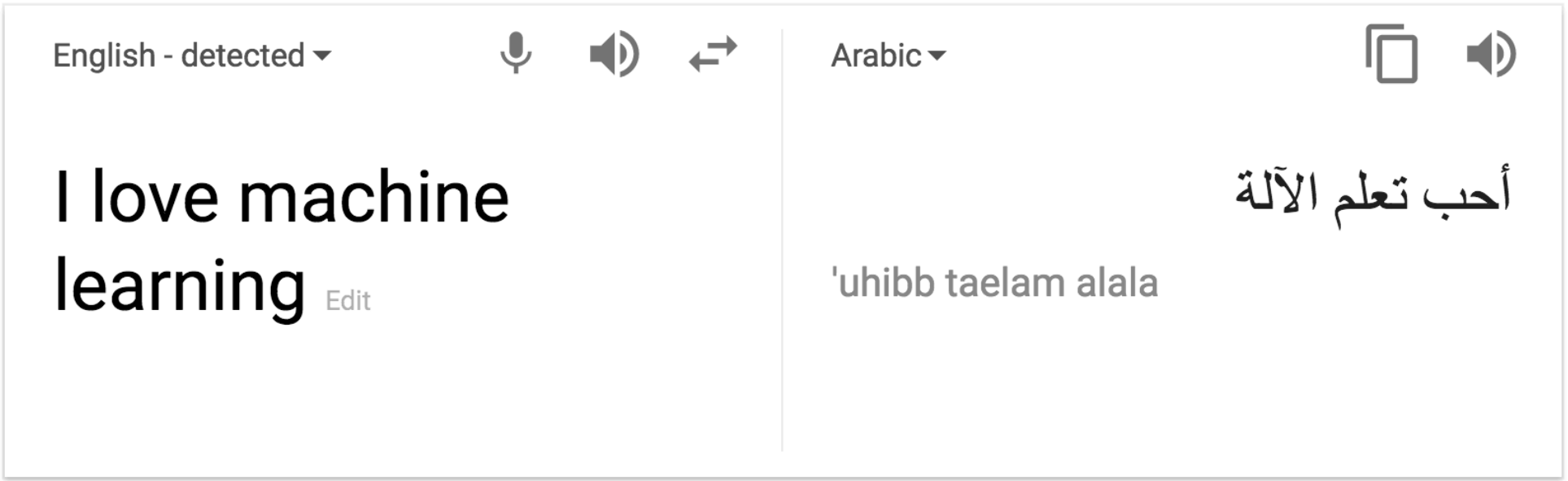
Flexible model forms

X

English sentence

y

Translation



Neural Networks:
Deep Learning

Deep learning is taking over

- **Machine vision**
  - Face/Object/Scene recognition
  - Self driving cars
- **Speech recognition (“speech to text”)**
  - Siri, Alexa …
- **Machine translation**
  - Google translate

Artificial Neural Nets

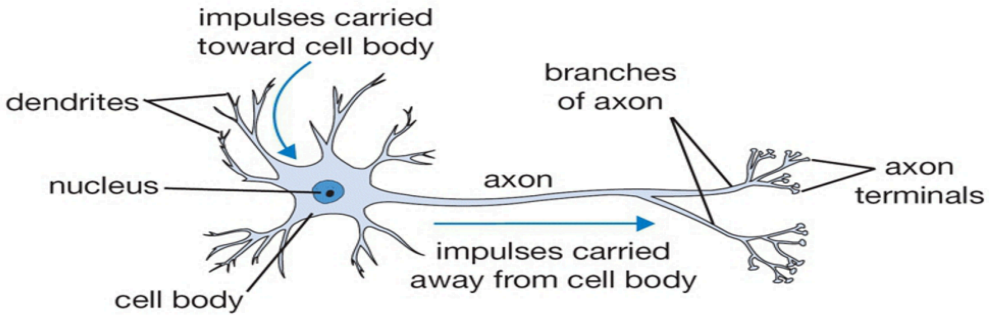
- **Semi-parametric**
  - Flexible model form
- **Used when there are vast amounts of data**
  - Hence popular (again) now
  - But recently with smaller training sets.
- **Deep networks**
  - Idea: representation should have *many* different levels of abstraction

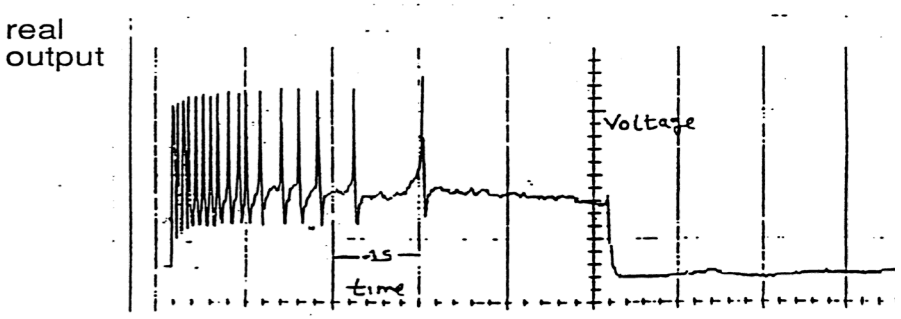
Neural Nets can be

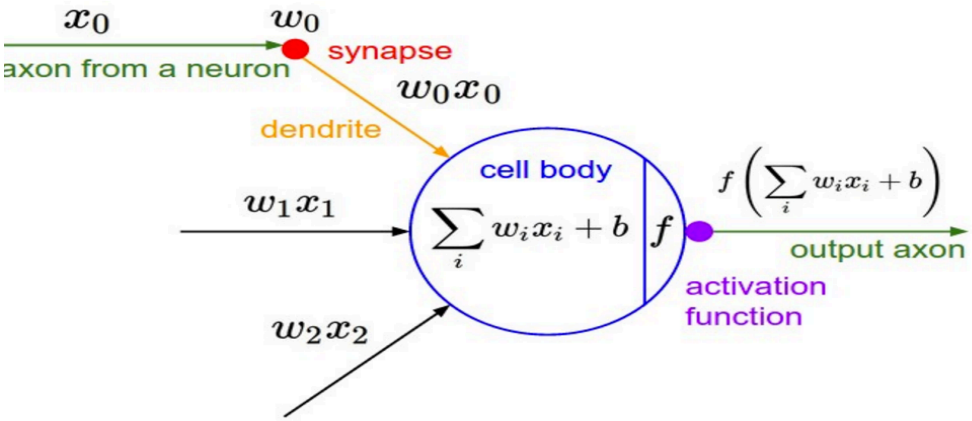
- **Supervised**
  - Generalizes *logistic regression* to a semi-parametric form
- **Unsupervised**
  - Generalizes *PCA* to a semi-parametric form
- **Semi-supervised**
- **Reinforcement**
- **Adversarial**

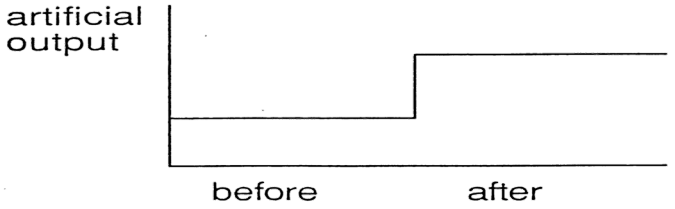
Neural nets often have built in structure

“Real” and Artificial neuron







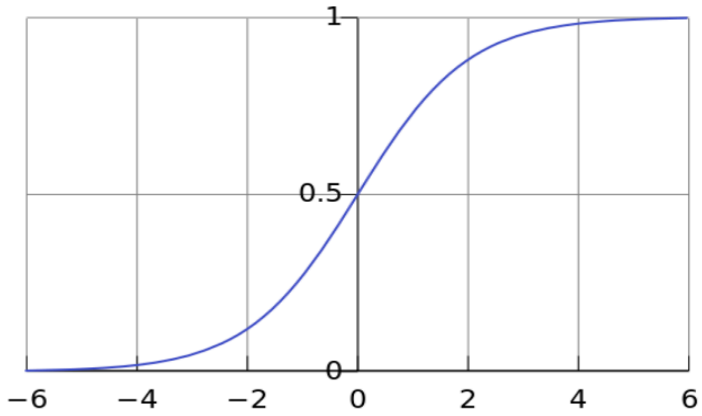


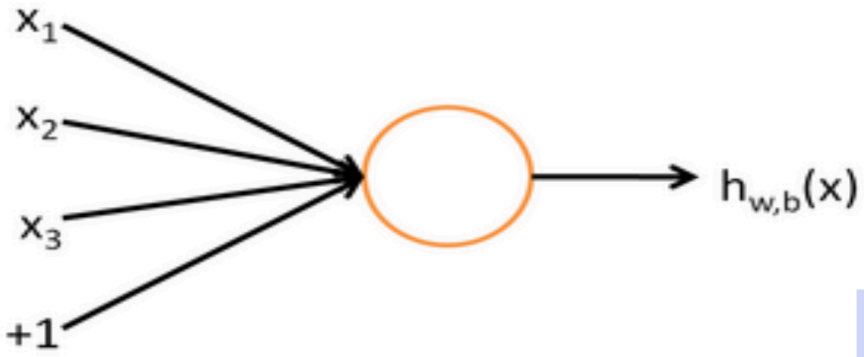
http://cs231n.github.io/neural-networks-1/

One neuron does logistic regression

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

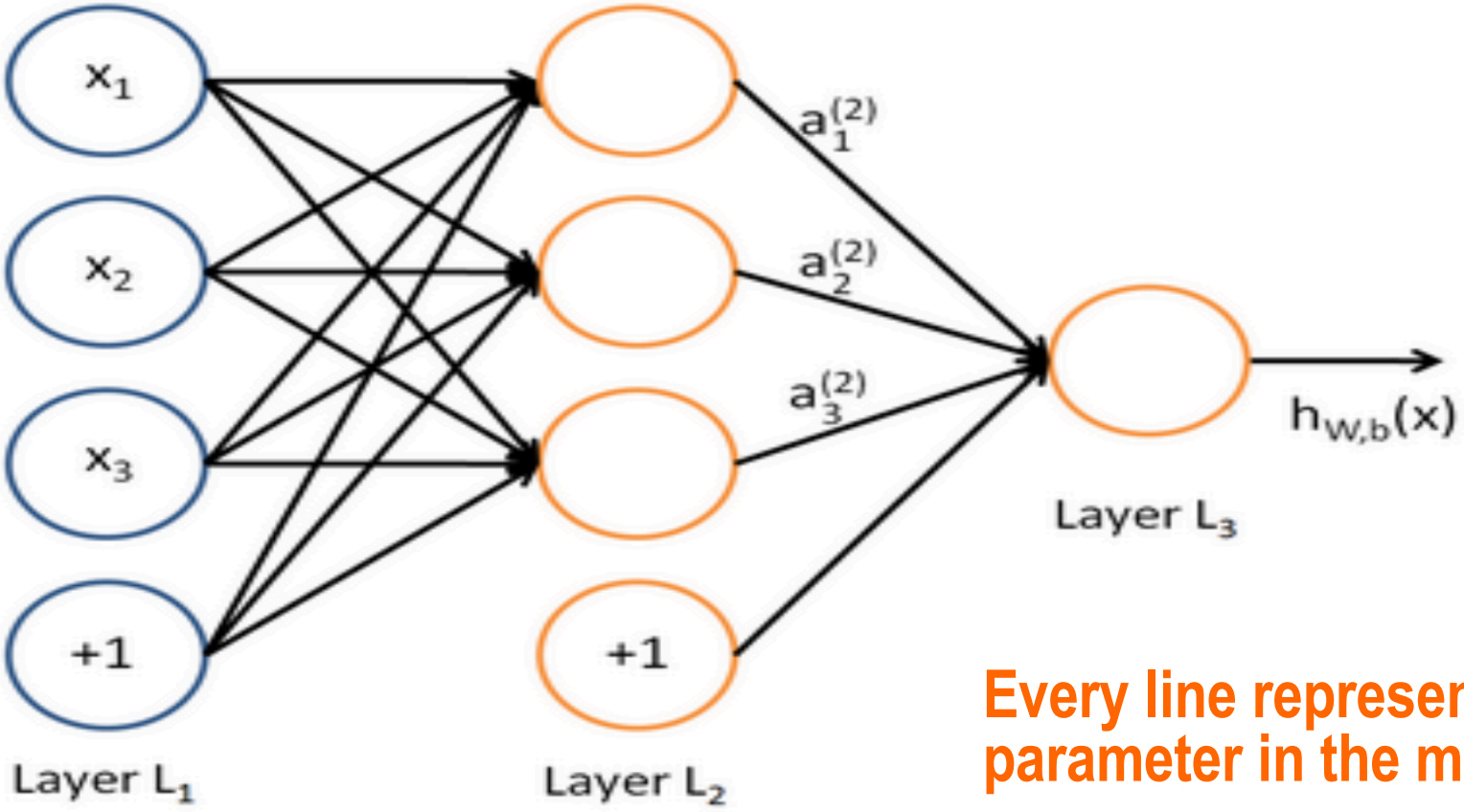
$$f(z) = \frac{1}{1+e^{-z}}$$





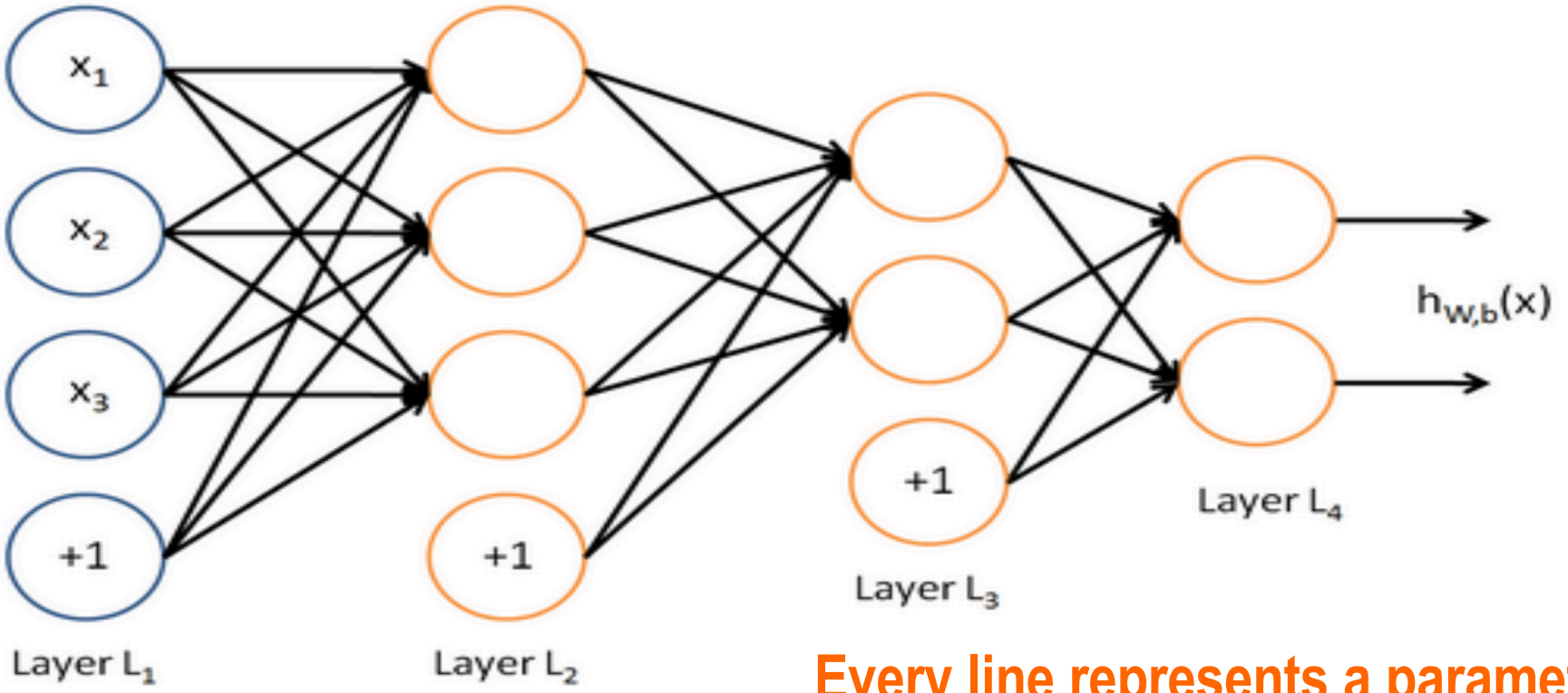
Socher and Manning tutorial

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

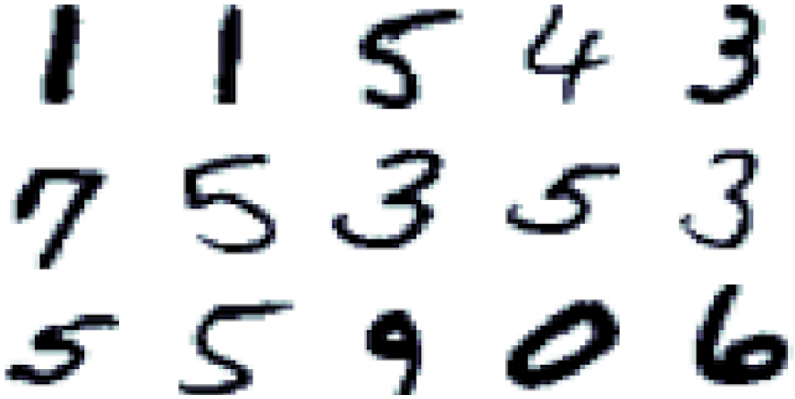
- Mini-batch gradient descent
- “Backpropagate” error derivatives through the model = chain rule

ANNs do pattern recognition

- **Map input "percepts" to output categories or actions**
  - Image of an object → what it is
  - Image of a person → who it is
  - Picture → caption describing it
  - Board position → probability of winning
  - A word → the sound of saying it
  - Sound of a word → the word
  - Sequence of words in English → 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)

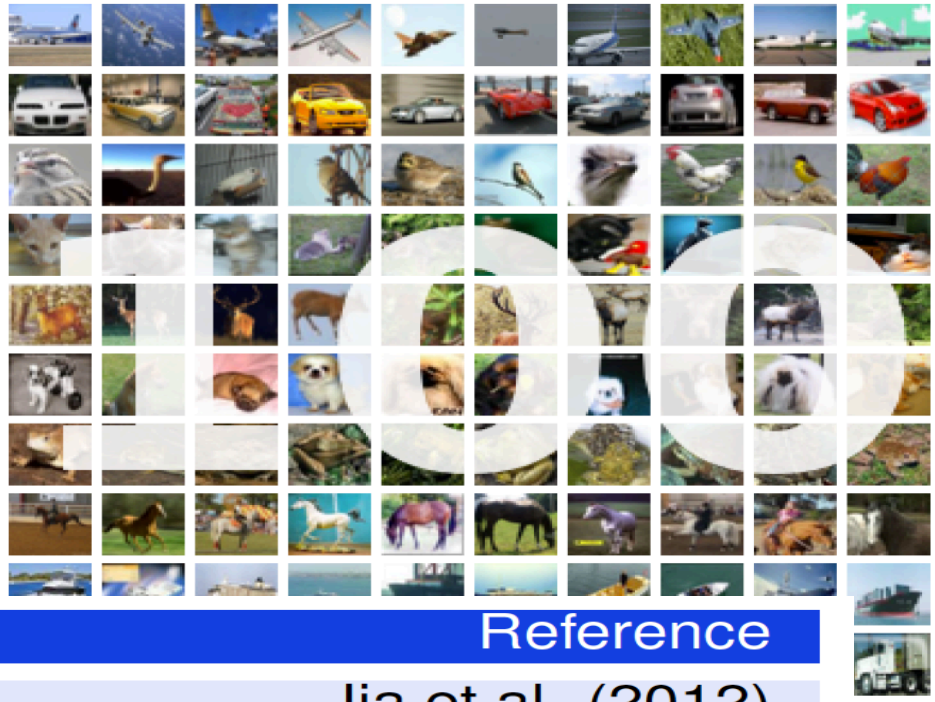
Street View House Numbers

- 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

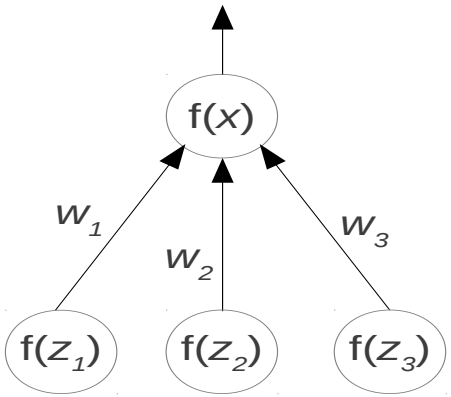
Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada

“AlexNet” 2012

Neural networks

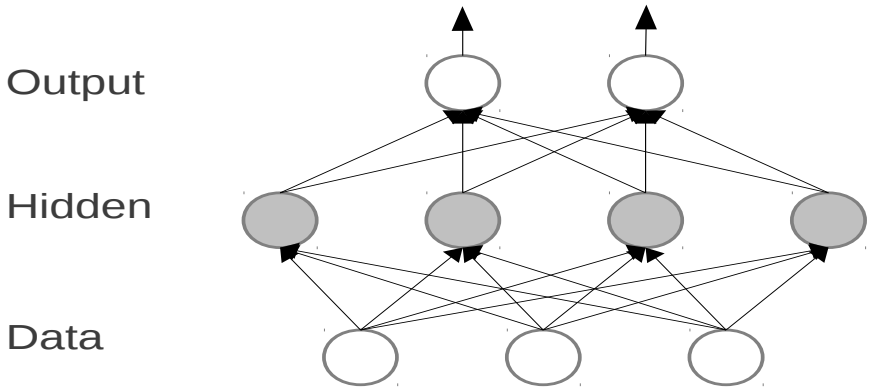
- A neuron



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

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

- A neural network

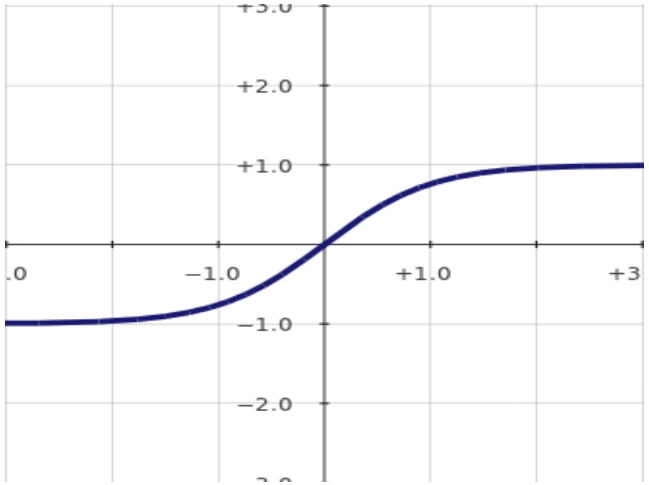


A neural network computes a differentiable function of its input. For example, ours computes: p(label | an input image)

Neurons

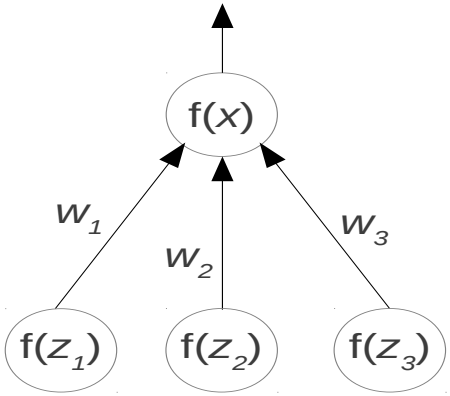
Traditional: sigmoidal e.g. logistic function

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



Hyperbolic tangent

Very bad (slow to train)

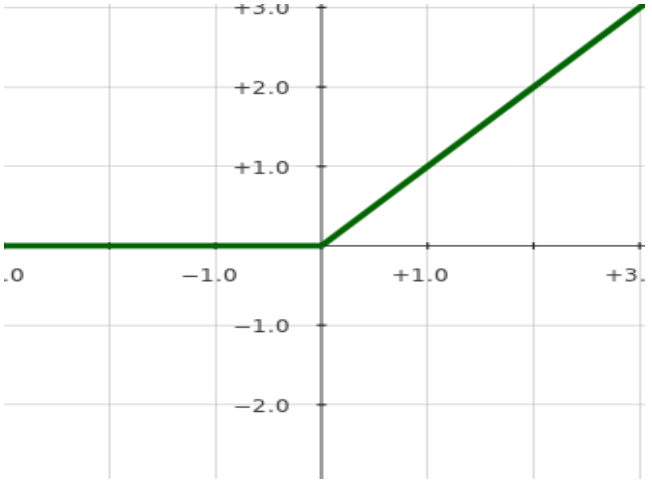


$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

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

But one can use any nonlinear function

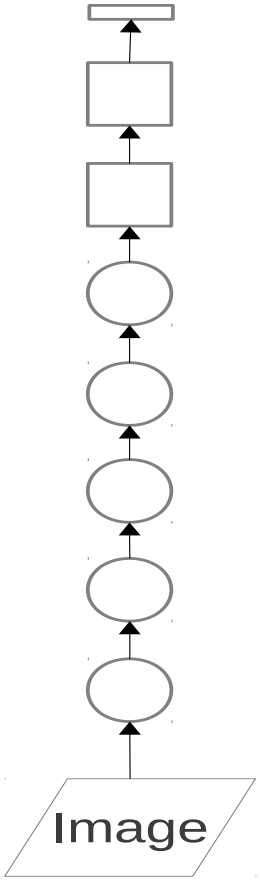
$f(x) = \max(0, x)$



Rectified Linear Unit (ReLU)

Very good (quick to train)

Overview of our model

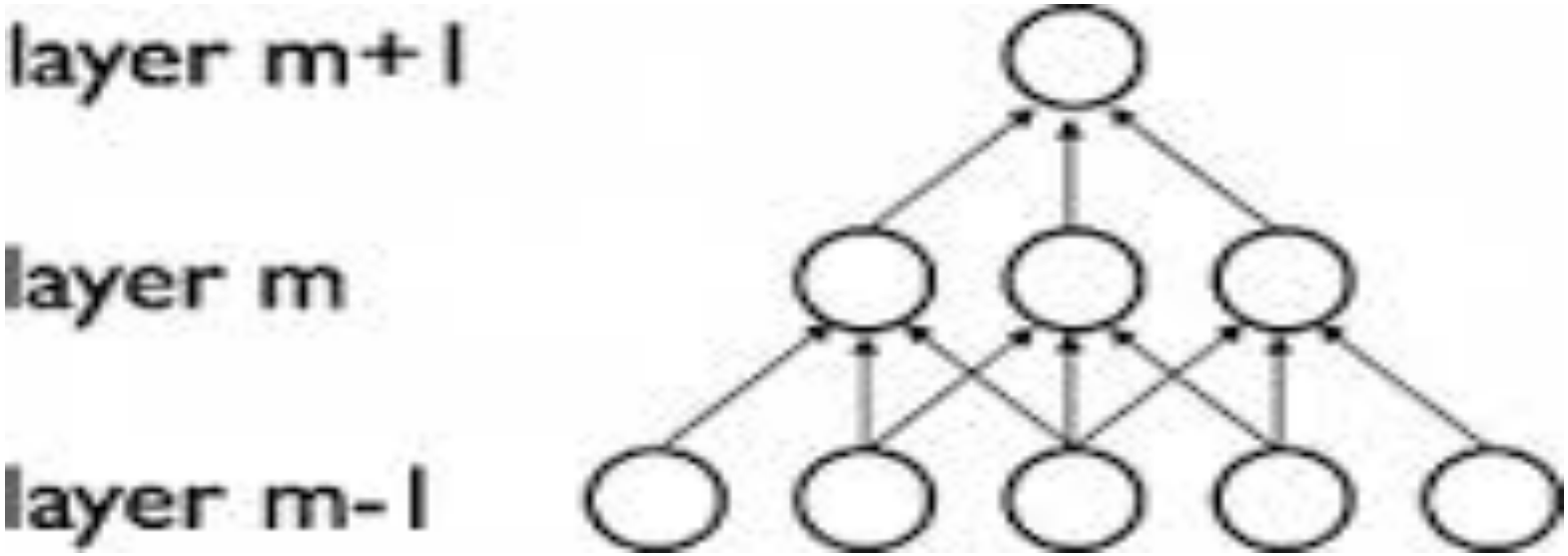


- **Deep**: 7 hidden “weight” layers
- **Learned**: all feature extractors initialized at white Gaussian noise and learned from the data
- Entirely supervised
- **More data = good**

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

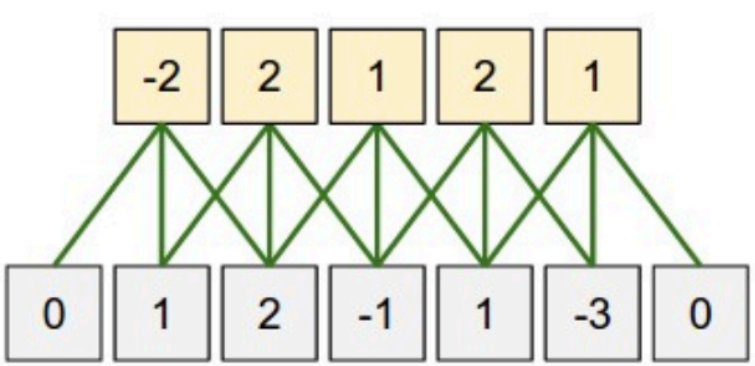
Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

Local receptive fields

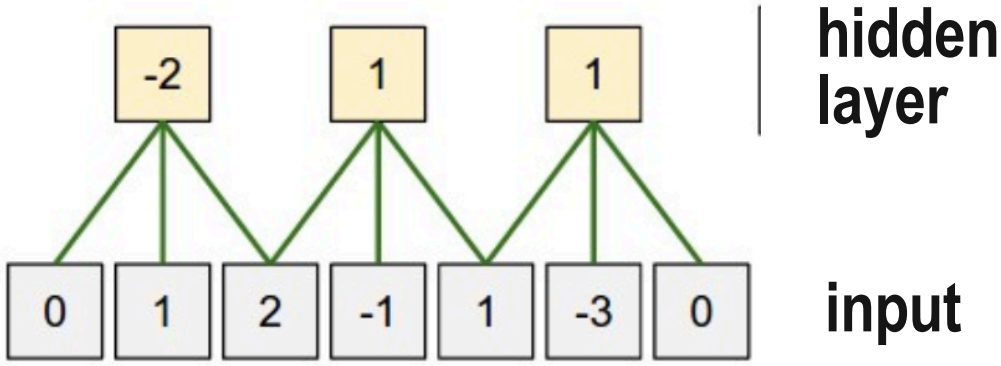


In vision, a neuron may only get inputs from a limited set of “nearby” neurons

Local receptive fields



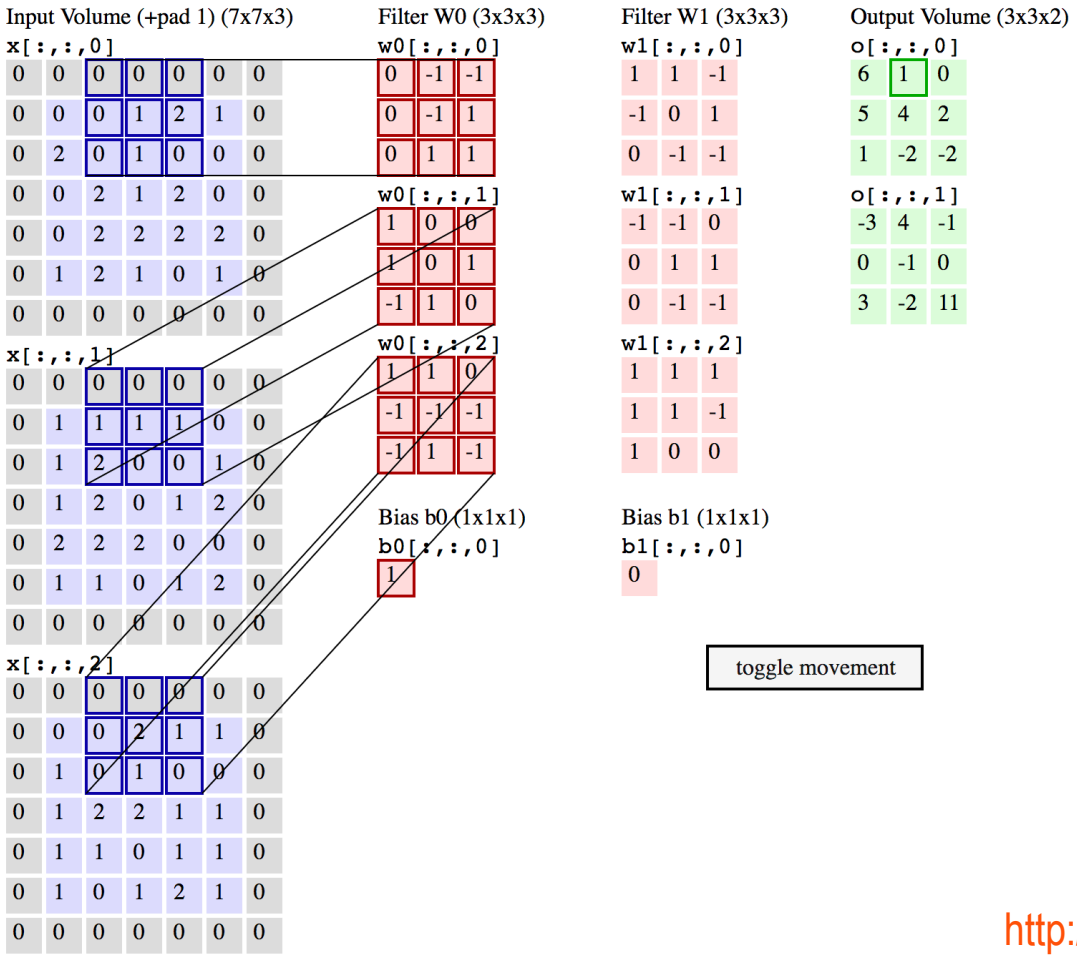
- spatial extent F = 3
- stride S =1



- spatial extent F = 3
- stride S = 2

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

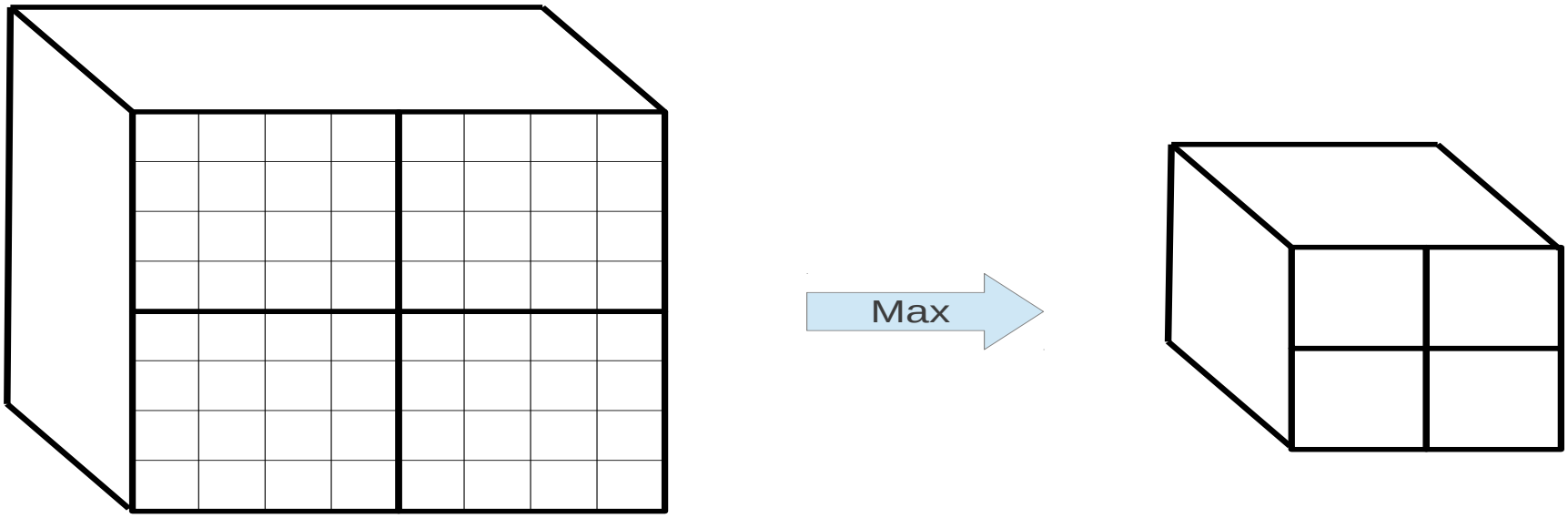
Local receptive fields



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

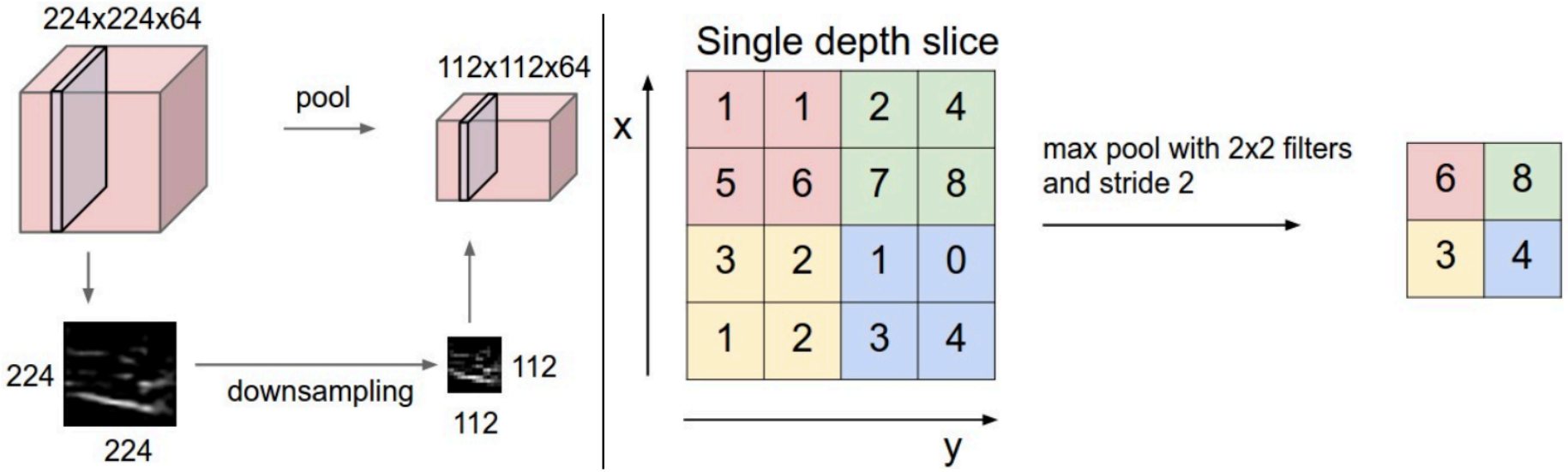
Local pooling

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

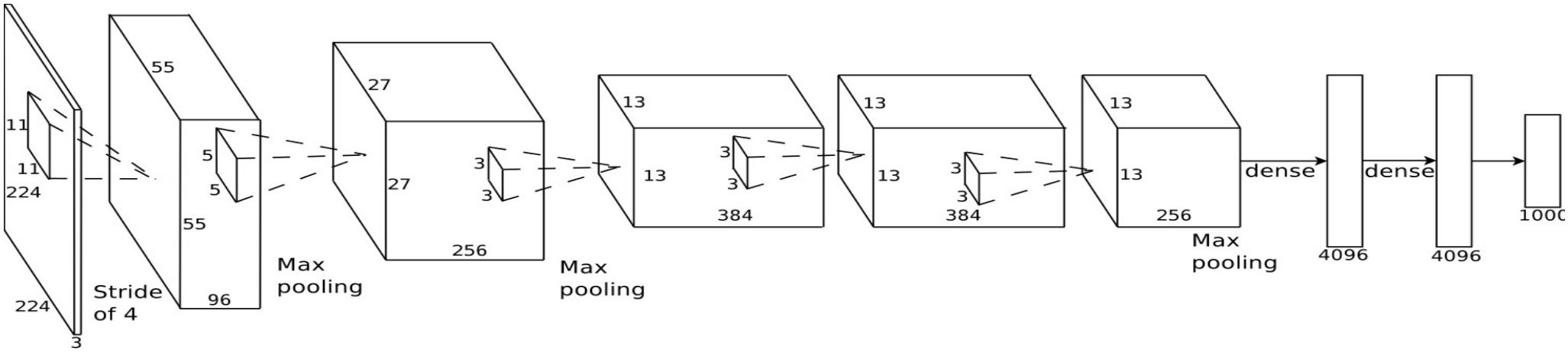


- *Spatial extent F=2*
- *Stride S=2*

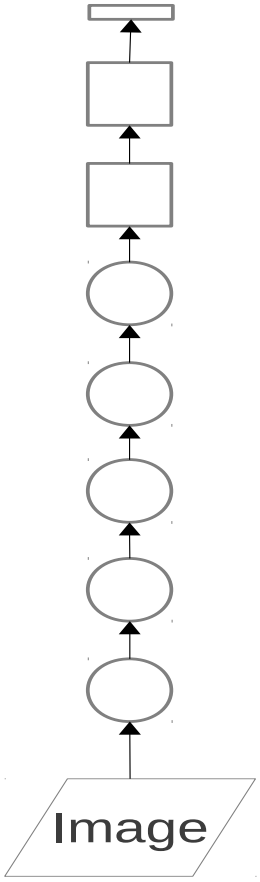
http://cs231n.github.io/convolutional-networks/

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

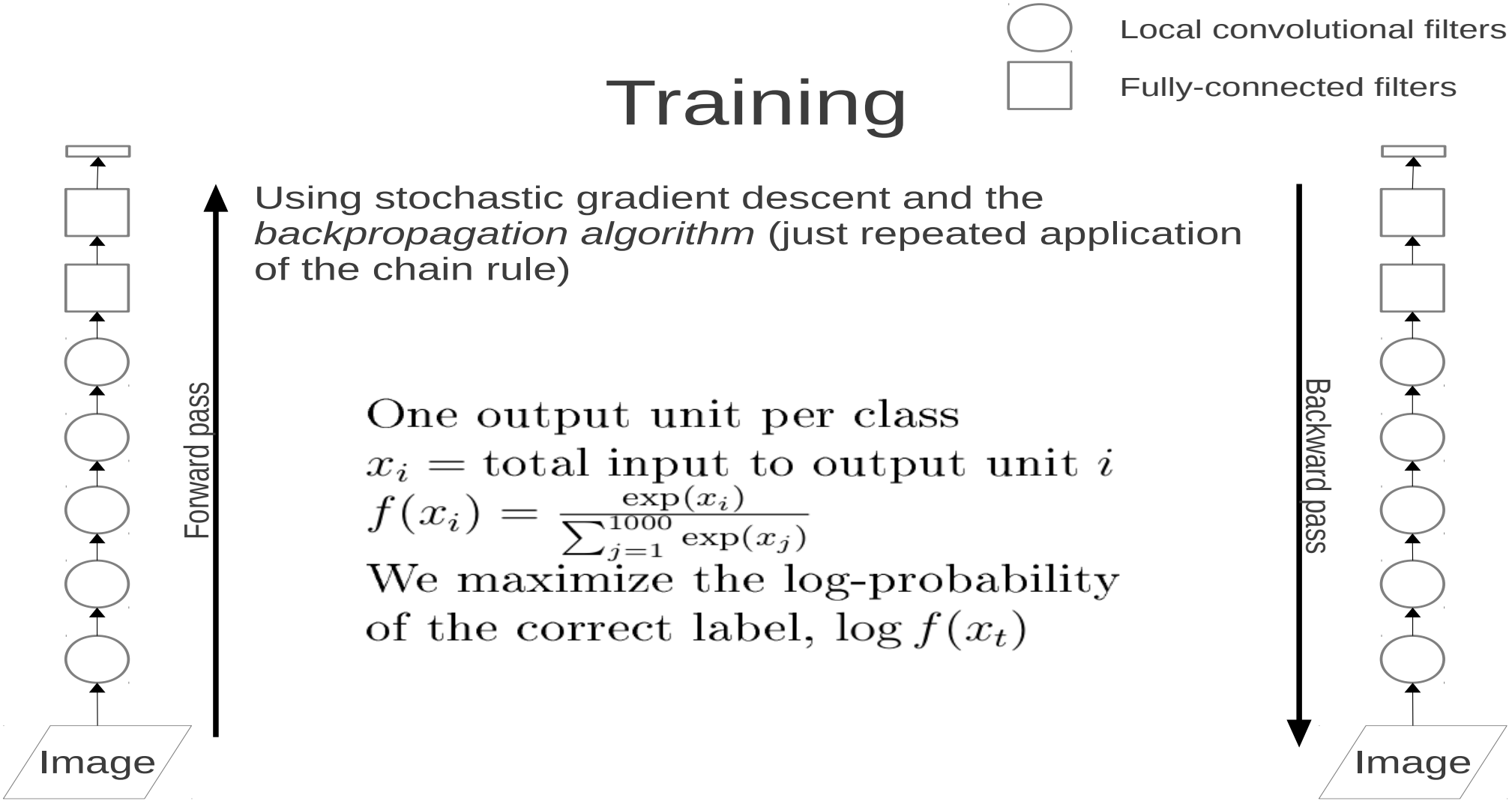


- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer:** 4096-dimensional

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

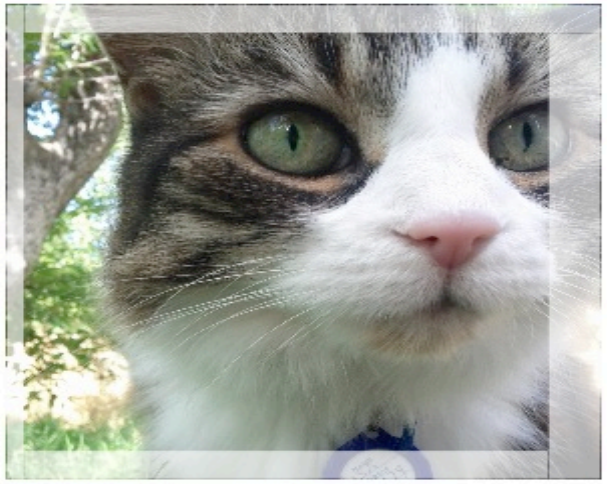
x_i = total input to output unit i

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

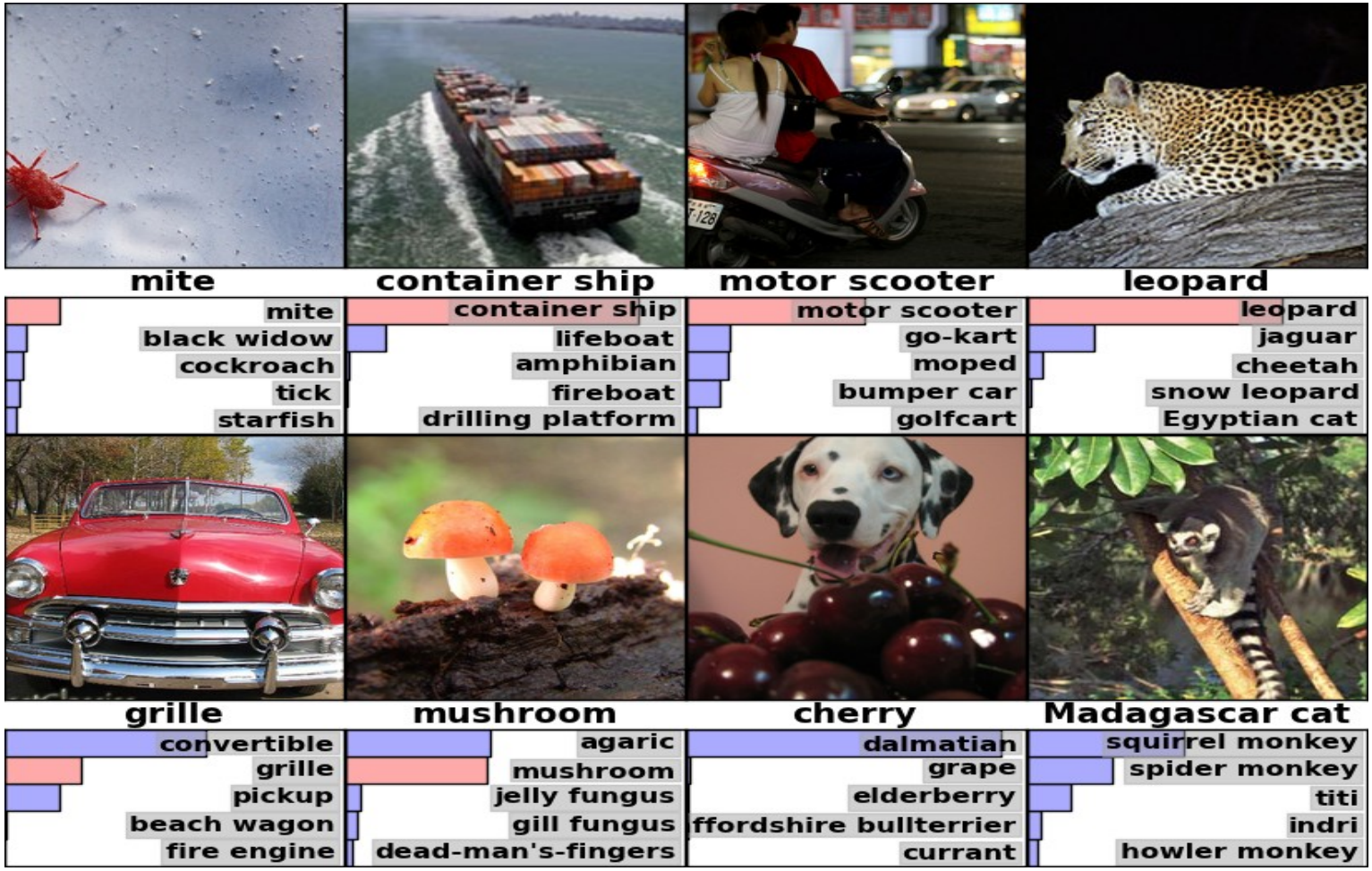
We maximize the log-probability of the correct label, $\log f(x_t)$

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.

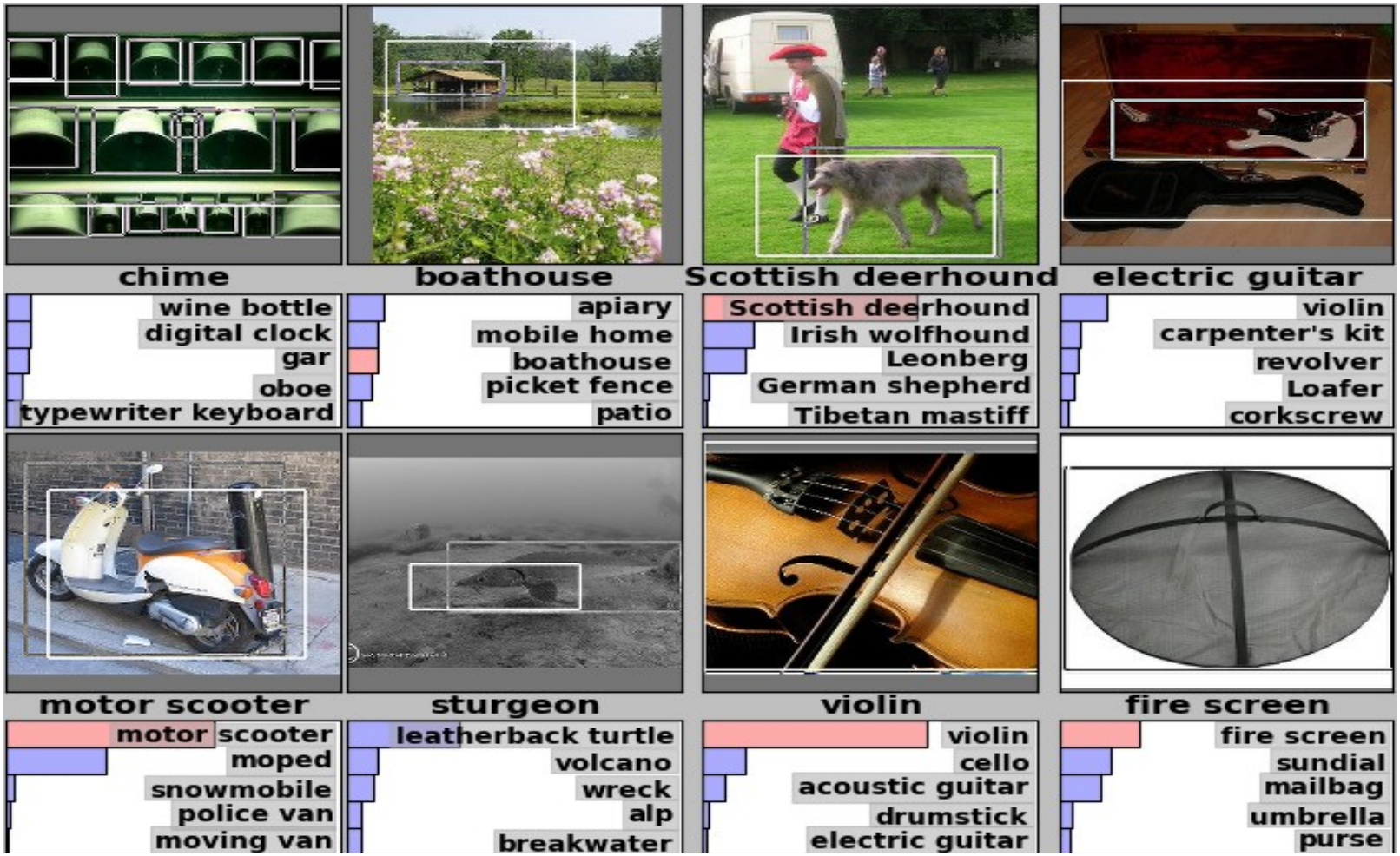


Validation classification



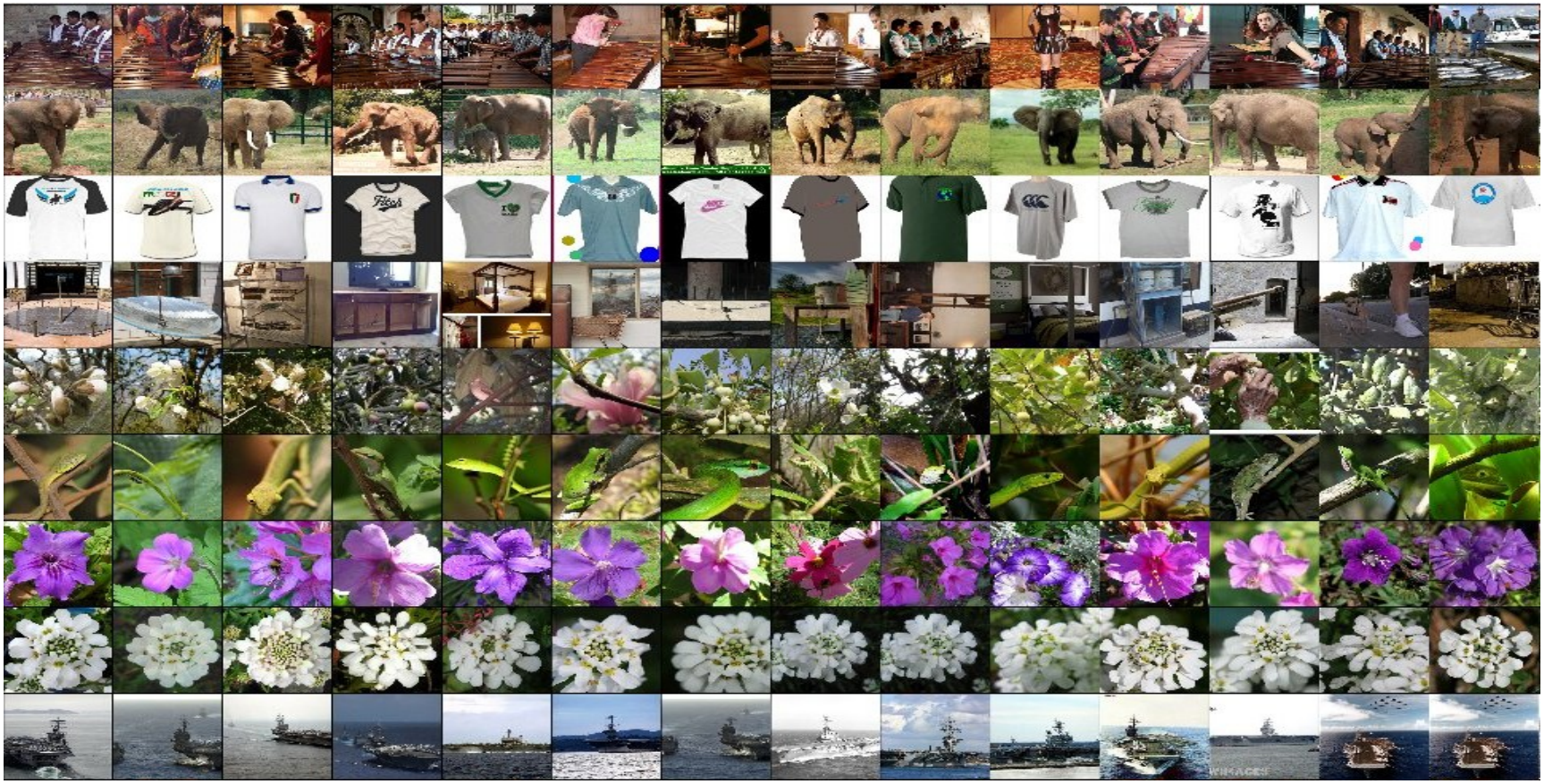
Validation classification

Validation localizations



Retrieval experiments

First column contains query images from ILSVRC-2010 test set, remaining columns contain retrieved images from training set.



Now used for image search; Benefit: Good Generalization

Both recognized as “meal”

Jeff Dean, google

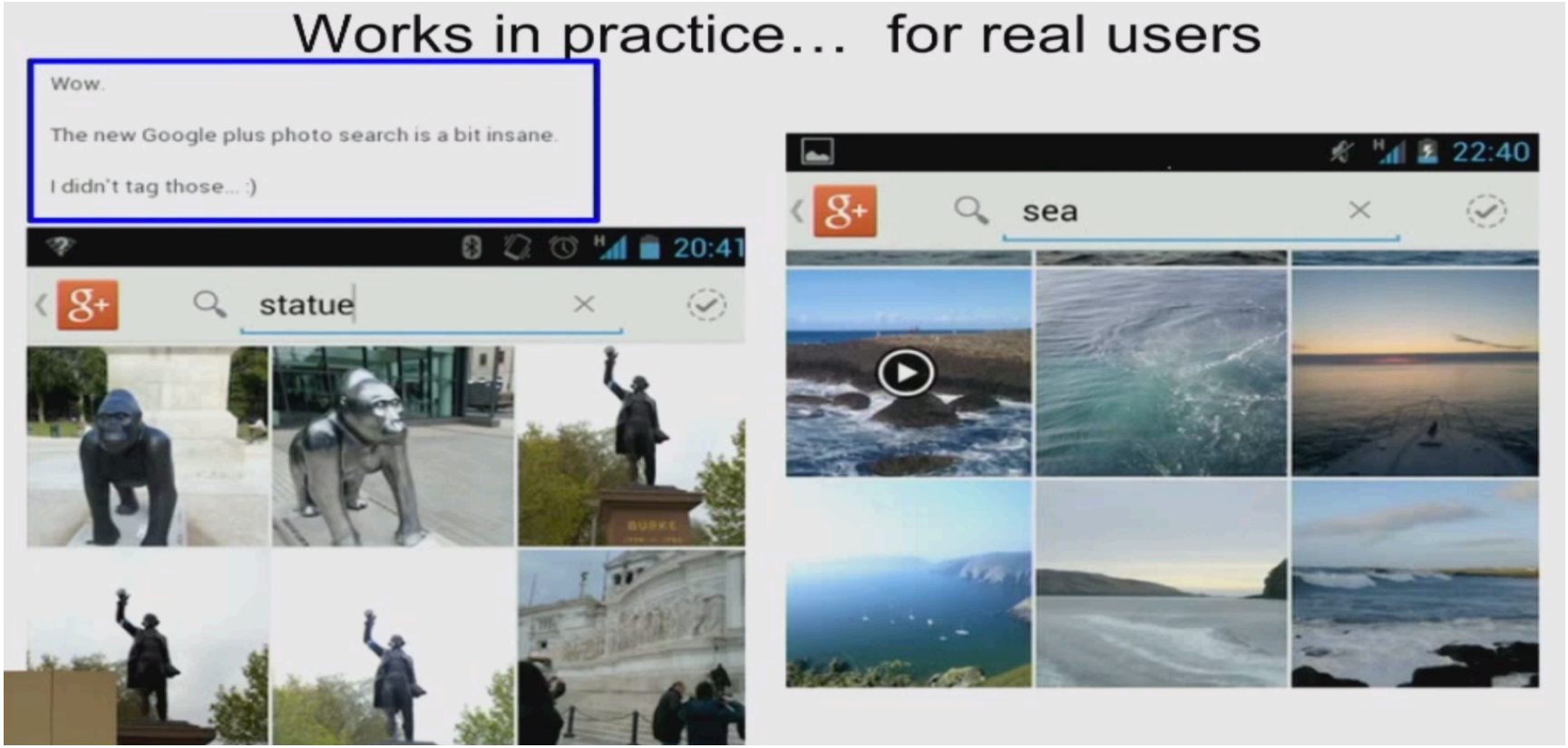
Sensible Errors (sometimes)

"snake"

"dog"

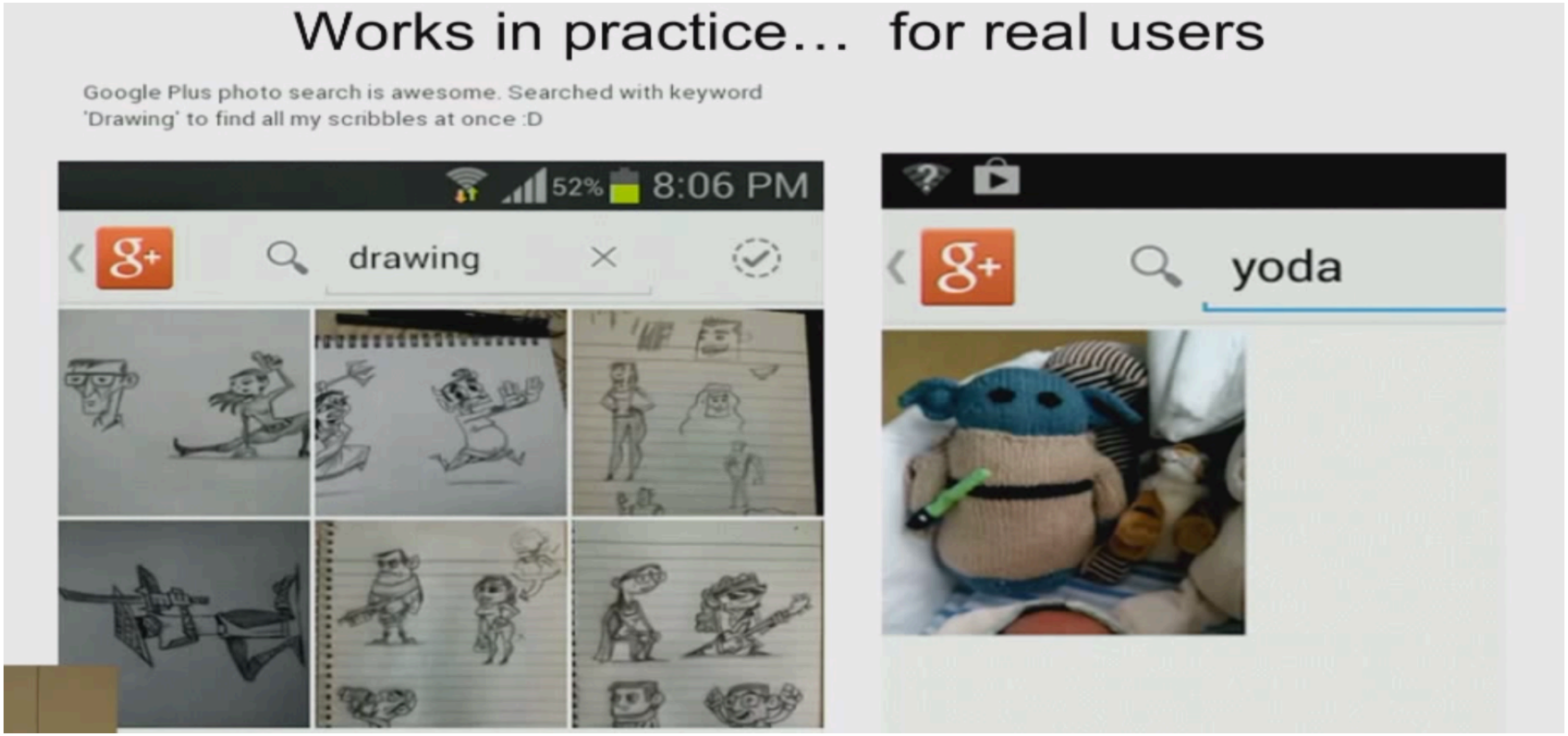
Jeff Dean, google

Now used for image search



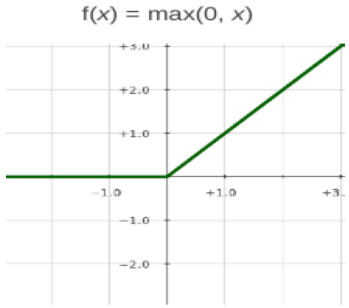
Jeff Dean, google

Now used for image search



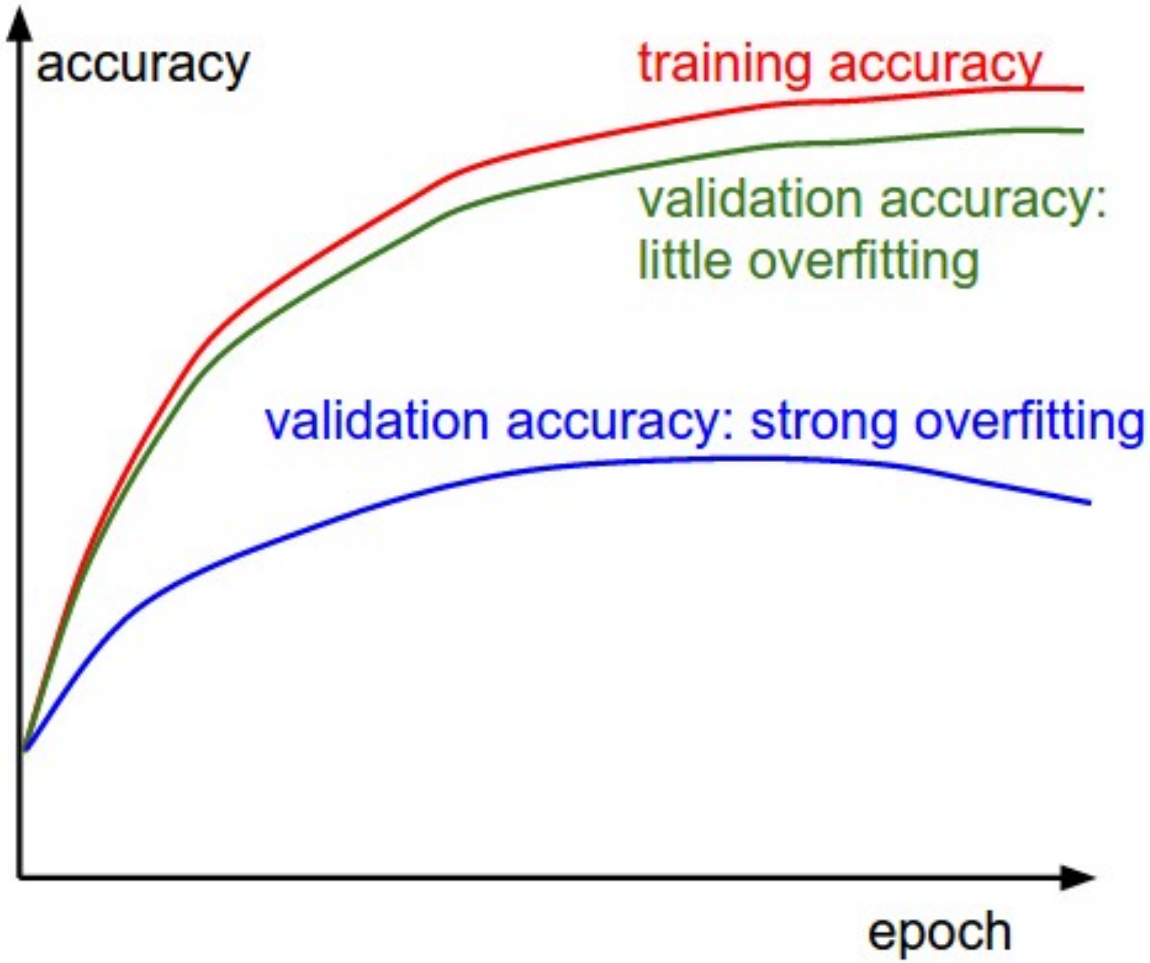
Jeff Dean, google

Modern deep nets



- **Often use rectified linear units (ReLUs)**
  - Less problems of saturation than logistic
- **Use a variety of loss functions**
  - Log likelihood (uses *softmax*) $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.
- **Can be very deep**
- **Solved with mini-batch gradient descent**
- **Regularized using L_2 penalty plus "dropout"**
  - and partial convergence and ..

Regularization



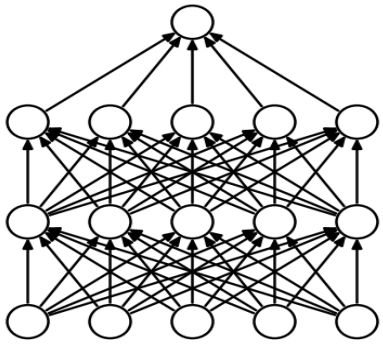
http://cs231n.github.io/neural-networks-3/

Regularization

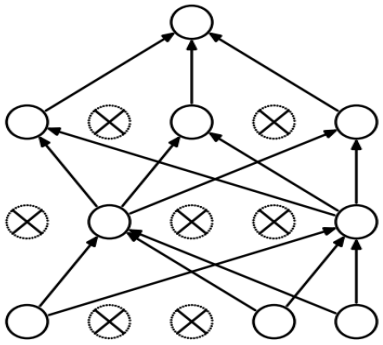
- L_2
- Max norm (L_∞)
- Early stopping
- Dropout

Dropout

- **Randomly (temporarily) remove a fraction *p* of the nodes (with replacement)**
  - Usually p = 1/2
- **Repeatedly doing this samples (in theory) over exponentially many networks**
  - Bounces the network out of local minima
- **For the final network use all the weights but shrink them by *p***



(a) Standard Neural Net



(b) After applying dropout.

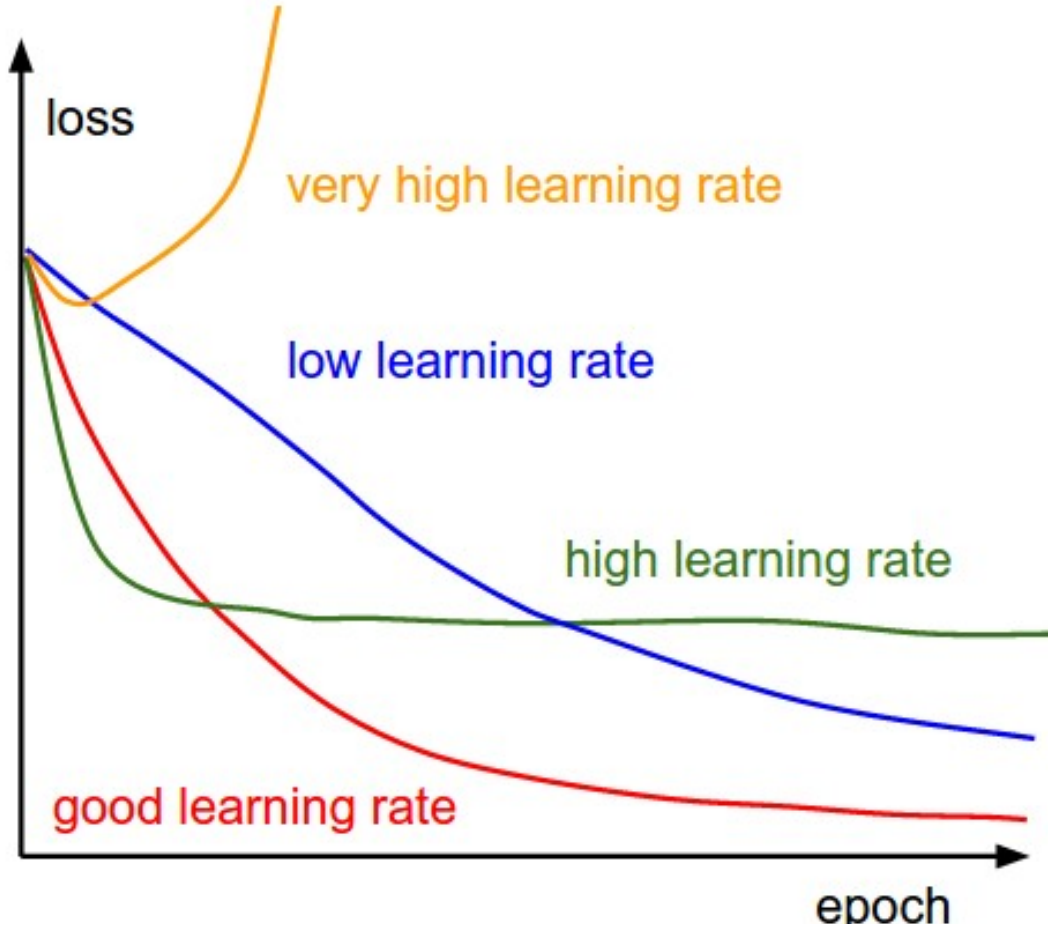
Gradient descent

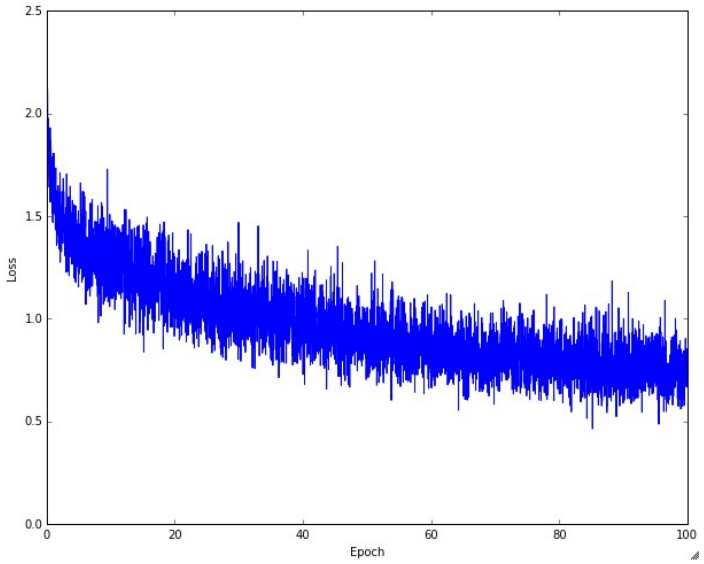
- **Gradient descent**
  - stochastic
  - gradient clipping
- **Minibatch**
- **Momentum**
- **Learning rate adaptation**
  - See the wiki

$$\frac{\delta Err}{\delta w} = \frac{Err(w+h) - Err(w-h)}{2h}$$

$$\Delta w^t = \eta \frac{\delta Err}{\delta w} + m \Delta w^{t-1}$$

Learning rate



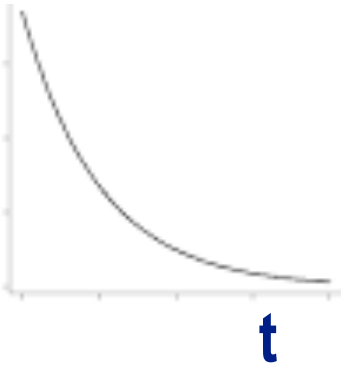


http://cs231n.github.io/neural-networks-3/

Learning rate adaption

- **Adjust the learning rate over time**

$$\Delta w^t = \eta(t)\frac{\delta Err}{\delta w}$$



- **Adagrad: make the learning rate depend on previous changes in *each* weight**
  - increases the learning rate for more sparse parameters and decreases the learning rate for less sparse ones.

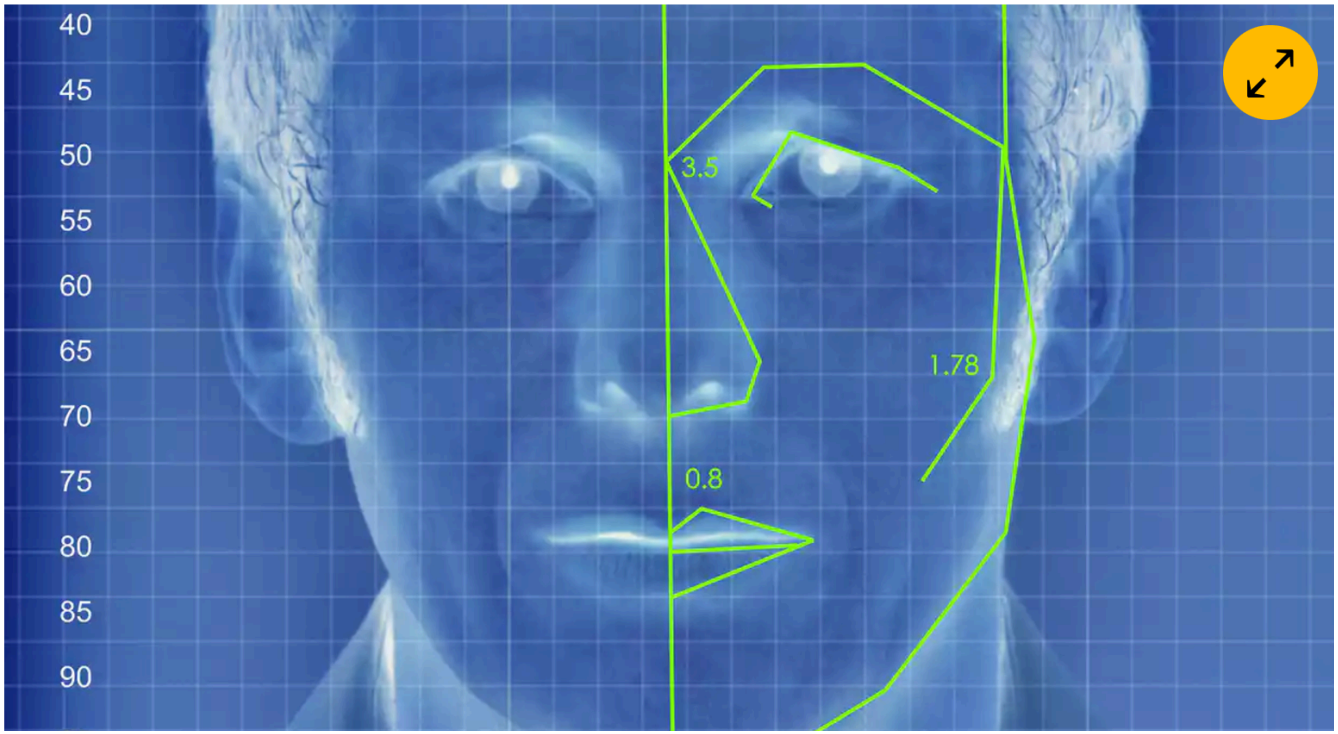
$$\Delta w_j^t = \frac{\eta}{||\delta w_j^\tau||_2}\frac{\delta Err}{\delta w_j}$$

Semi-supervised learning

- How do you learn a model with only a few hundred labeled examples?

New AI can guess whether you're gay or straight from a photograph

An algorithm deduced the sexuality of people on a dating site with up to 91% accuracy, raising tricky ethical questions



https://www.theguardian.com/technology/2017/sep/07/new-artificial-intelligence-can-tell-whether-youre-gay-or-straight-from-a-photograph

Deep neural networks are more accurate than humans at detecting sexual orientation from facial images.

Michal Kosinski & Yilun Wang

We show that faces contain much more information about sexual orientation than can be perceived and interpreted by the human brain. We used deep neural networks to extract features from 35,326 facial images. These features were entered into a logistic regression aimed at classifying sexual orientation. Given a single facial image, a classifier could correctly distinguish between gay and heterosexual men in 81% of cases, and in 74% of cases for women. Human judges achieved much lower accuracy: 61% for men and 54% for women. The accuracy of the algorithm increased to 91% and 83%, respectively, given five facial images per person. Facial features employed by the classifier included both fixed (e.g., nose shape) and transient facial features (e.g., grooming style). Consistent with the prenatal hormone theory of sexual orientation, gay men and women tended to have gender-atypical facial morphology, expression, and grooming styles. …. Additionally, given that companies and governments are increasingly using computer vision algorithms to detect people's intimate traits, our findings expose a threat to the privacy and safety of gay men and women.

https://osf.io/fk3xr/

2017

Deep learning case study

- **Download images and labels from a dating site**
  - where people declare their sexual orientation
- **Only keep images with a single "good" face**
  - Use Face++ to identify faces -- yielded 35,000 faces
- **Use M-turkers to QC & restrict to Caucasians**
- **Use pretrained CNN to compute ~ 4,000 'scores'/image**
  - VGG-Face was trained on 2.6 million faces
- **Use logistic regression on SVD of the 4,000 scores**
  - report cross-validation error predicting gay/straight

What you should know

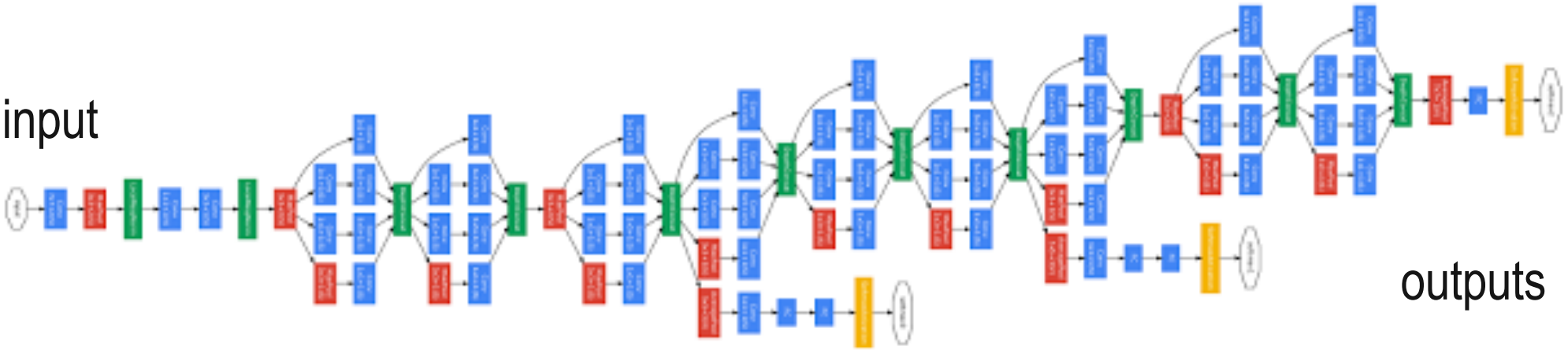
- **CNN**
  - local receptive field
- **Rectified Linear Unit (RLU)**
- **Dropout**
- **Back-propagation**
- **Mini-batch**
- **At least four kinds of regularization**

Visualizing networks

- **Display pattern of hidden unit activations**
  - Just shows they are sparse
- **Show input that maximizes a node's output**
  - Over all inputs in the training set
  - Over the entire range of possible inputs
  - Early layers do feature detection
  - Later layers do object detection
- **Show how occluding parts of an image affect classification accuracy**

http://cs231n.github.io/understanding-cnn/

Lots of fancy network structures

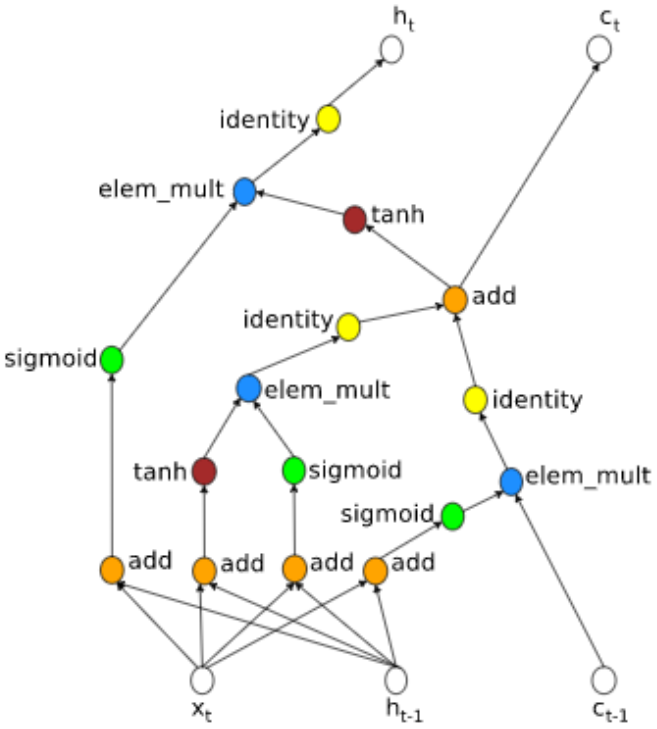


Convolutional (different sizes)
Or fully connected
Maxpool
Concatenation
Softmax

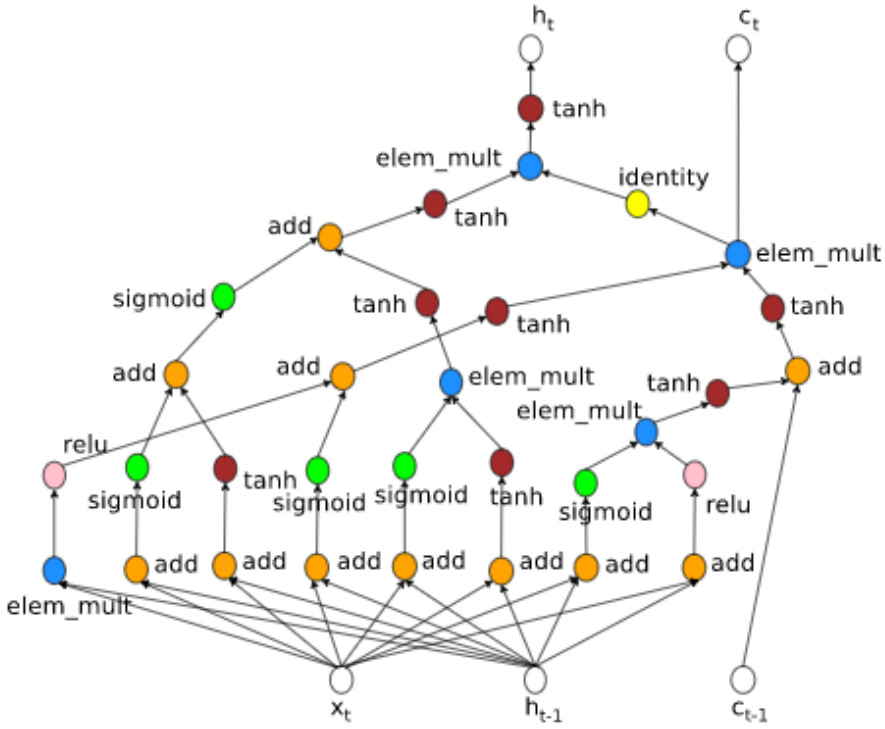
Some layers
use dropout

googlenet

AutoML learns network structure



Human built



Learned by RL

https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html

What do you like best about the class? What can I improve on?