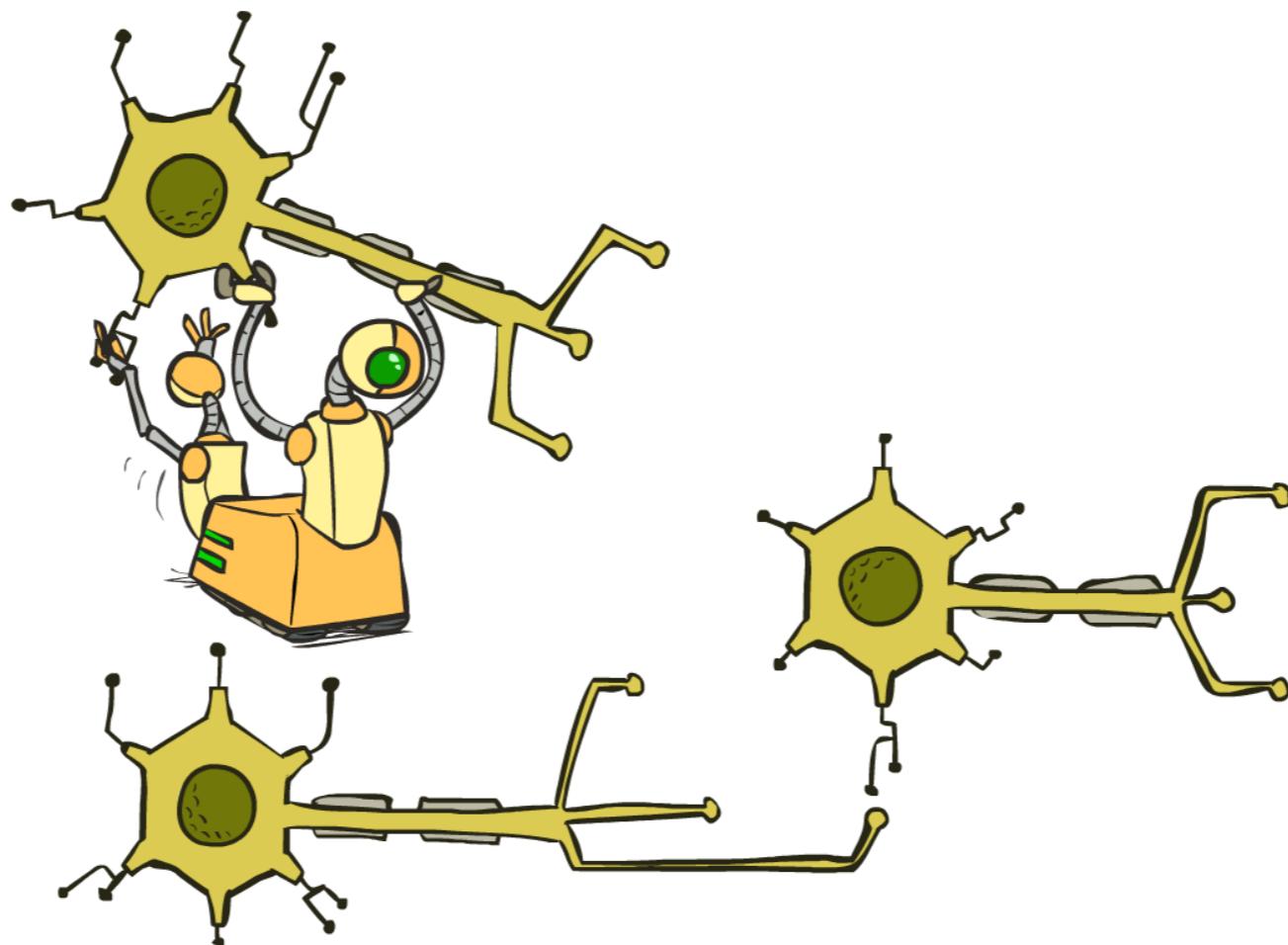


# Ve492: Introduction to Artificial Intelligence

## Neural Nets



Paul Weng

UM-SJTU Joint Institute

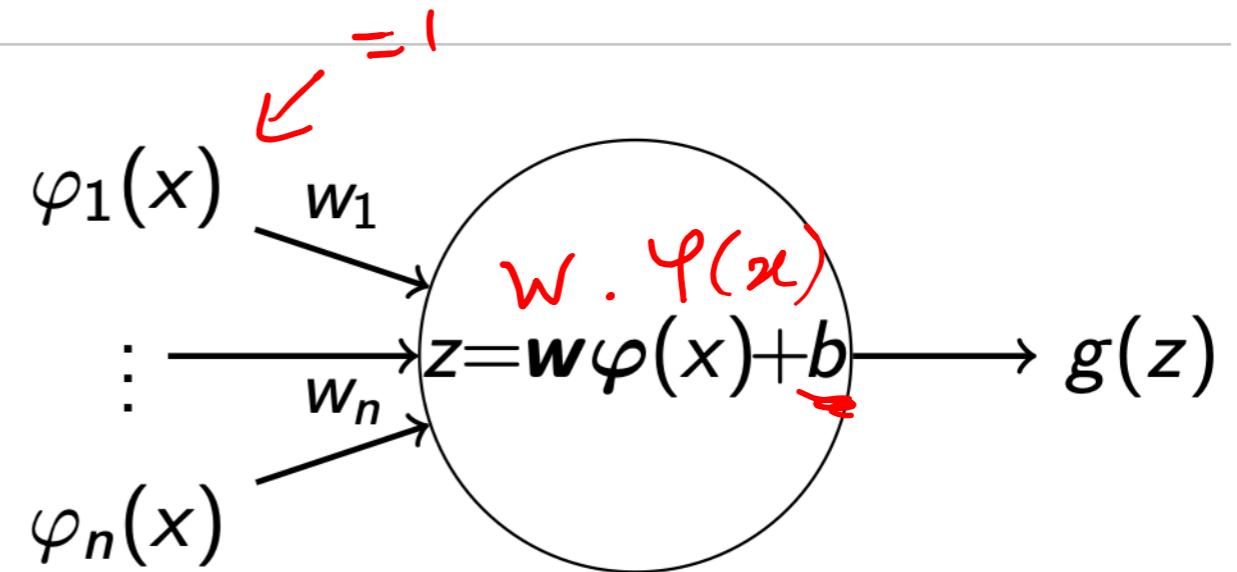
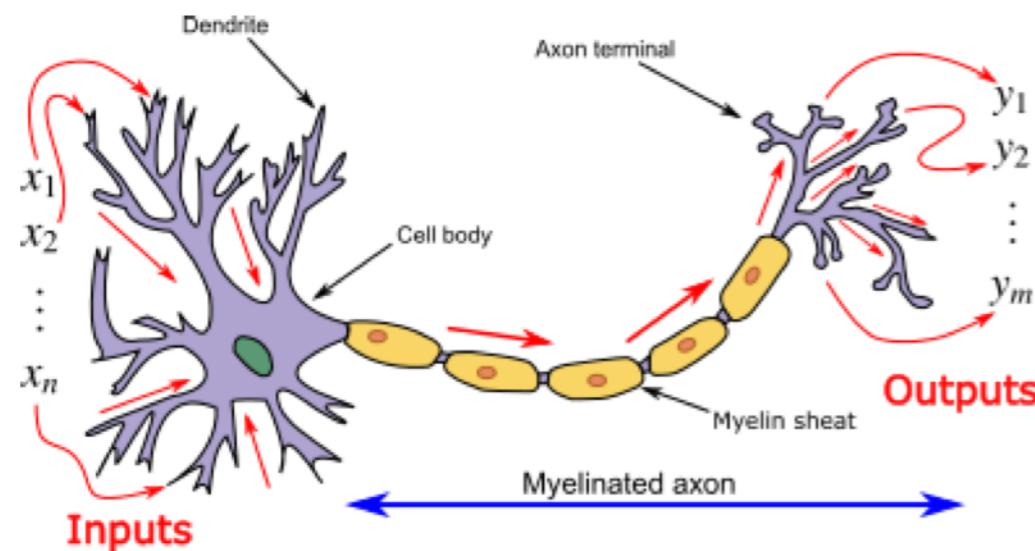
Slides adapted from <http://ai.berkeley.edu>, AIMA, UM

# Today

---

- ❖ (Deep) neural network
- ❖ Applications

# Artificial Neuron



- ❖ Perceptron:  $g(z) = \underline{\text{sign}}(z)$
- ❖ Logistic regression:  $g(z) = \frac{1}{1+e^{-z}}$
- ❖ Linear regression:  $g(z) = z$
- ❖  $g$  is called activation function, usually non-linear (sub)differentiable

# How to Learn the Parameters of the Model?

- ❖ Iterative method that updates unknown weights

- ❖ For perceptron:

$$w \leftarrow w + y^* \varphi(x) \mathbf{1}(w \cdot \varphi(x) \cdot y^* < 0) \quad \checkmark$$

- ❖ For logistic regression:

- ❖ Write likelihood function:  $P(X|w)$

- ❖ Maximize  $\log P(X|w)$  with gradient descent

$$w \leftarrow w + \alpha \nabla_w \log P(X|w) \quad (\text{batch})$$

$$w \leftarrow w + \alpha \nabla_w \log P(x|w) \quad (\text{stochastic})$$

# General Framework: Statistical ML

- ❖ Minimize empirical risk:

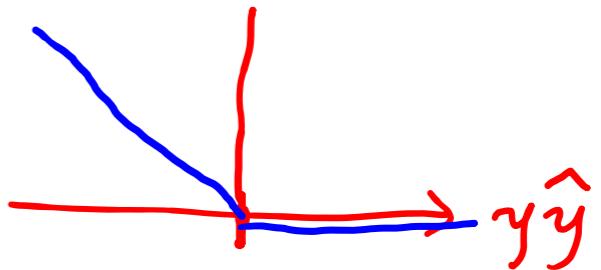
$$\min_w \sum_i \ell(h_w(x^i), y^i)$$

feature vector  $x^i$     label  $y^i$

loss function or measure of error

- ❖ For perceptron:

$$y \in \{-1, 1\} \quad \ell(\hat{y}, y) = \max(0, -y\hat{y})$$



- ❖ For logistic regression:

$$y \in \{0, 1\} \quad \ell(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- ❖ For linear regression:

$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$

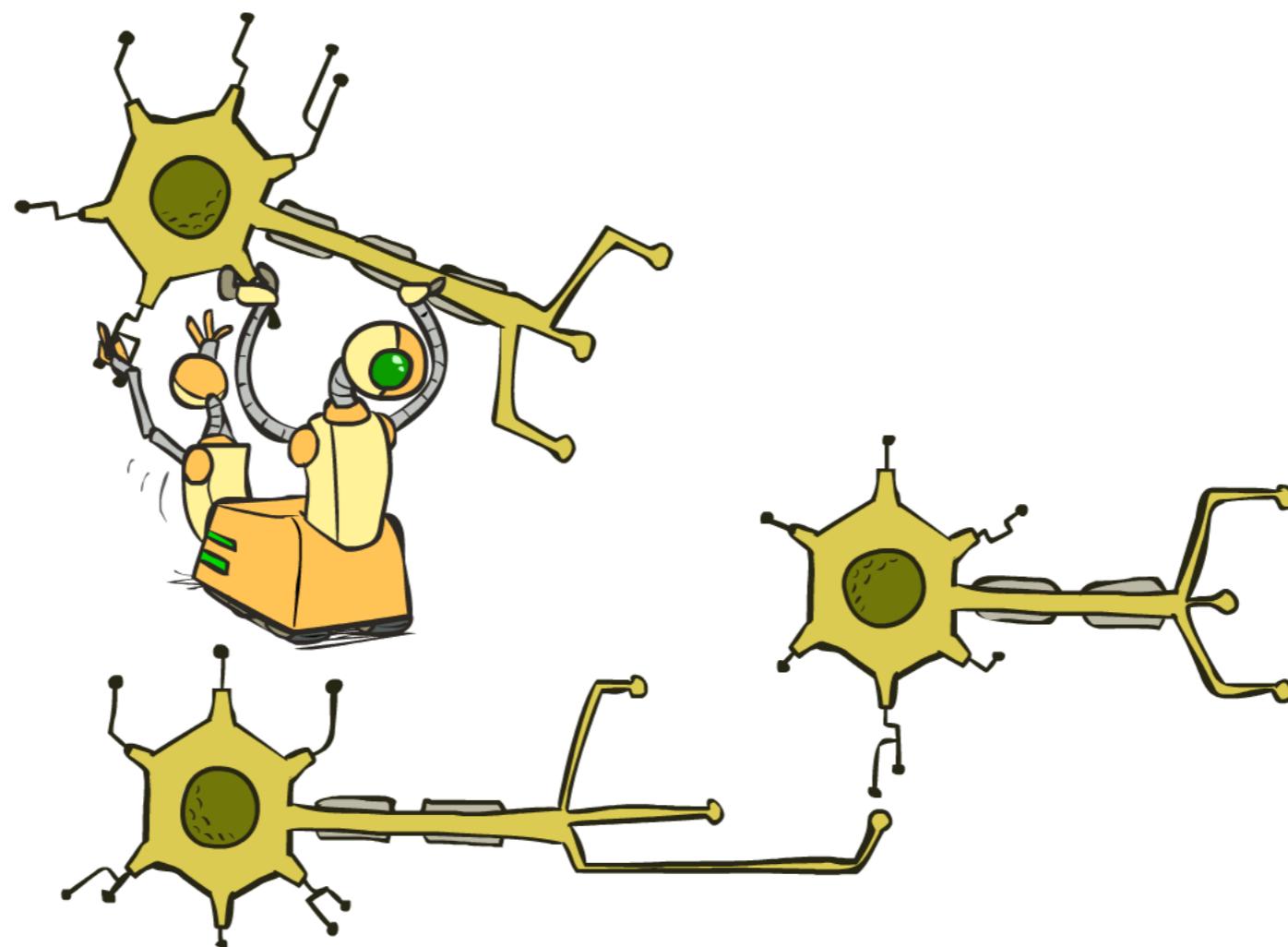
# Mini-Batch Gradient Descent

$$\min_w \sum_i \ell(h_w(x^i), y^i) ]$$

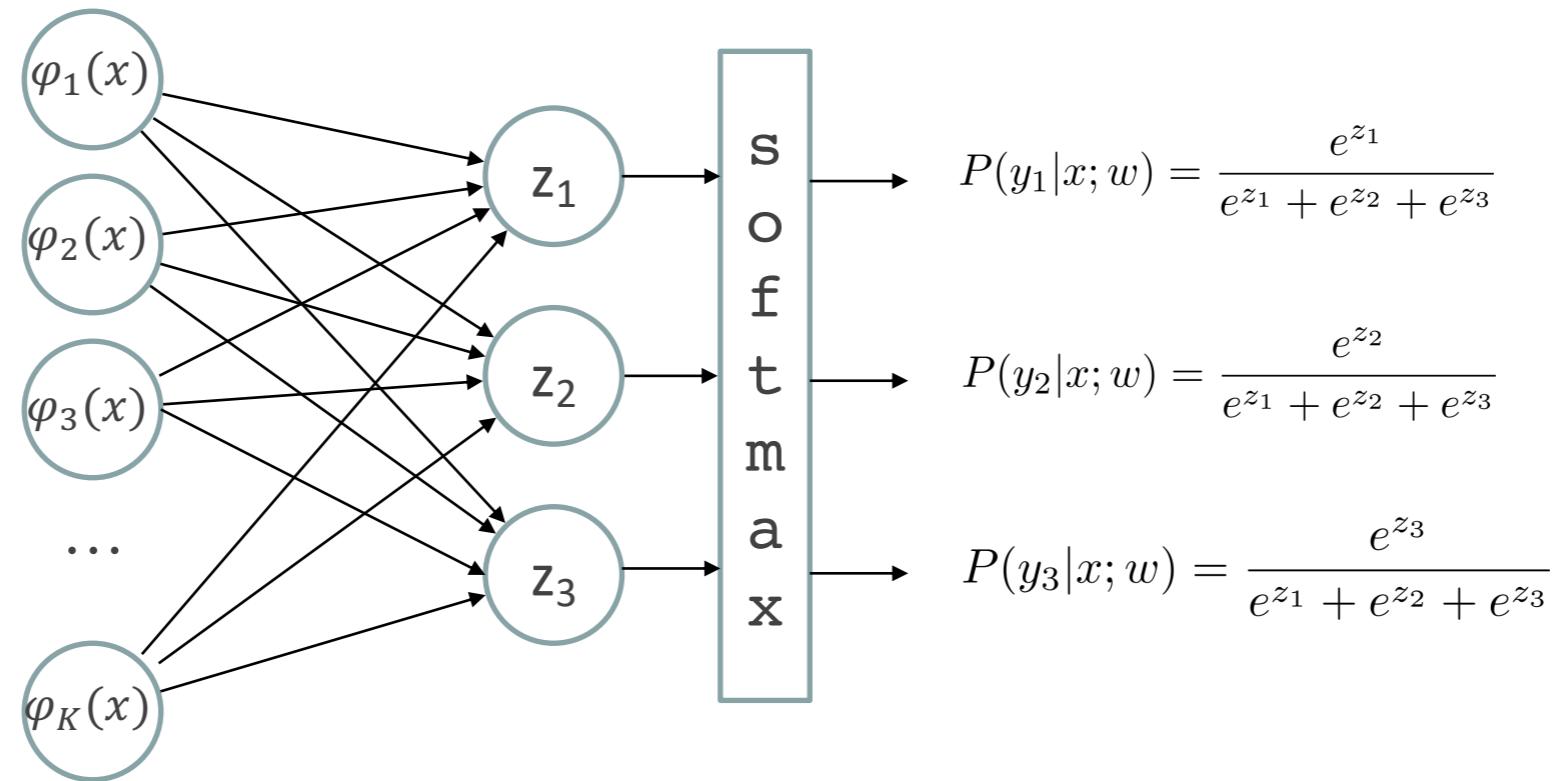
**Observation:** gradient over small set of training examples (=mini-batch) can be computed in parallel, might as well do that instead of a single one

- ❖ init  $w$
- ❖ for iter = 1, 2, ...
  - ❖ pick random subset of training examples  $J$   
 $\underline{\text{mini batch}}$ 
$$w \leftarrow w - \alpha \sum_{j \in J} \nabla_w \ell(h_w(x^j), y^j)$$

# Neural Networks

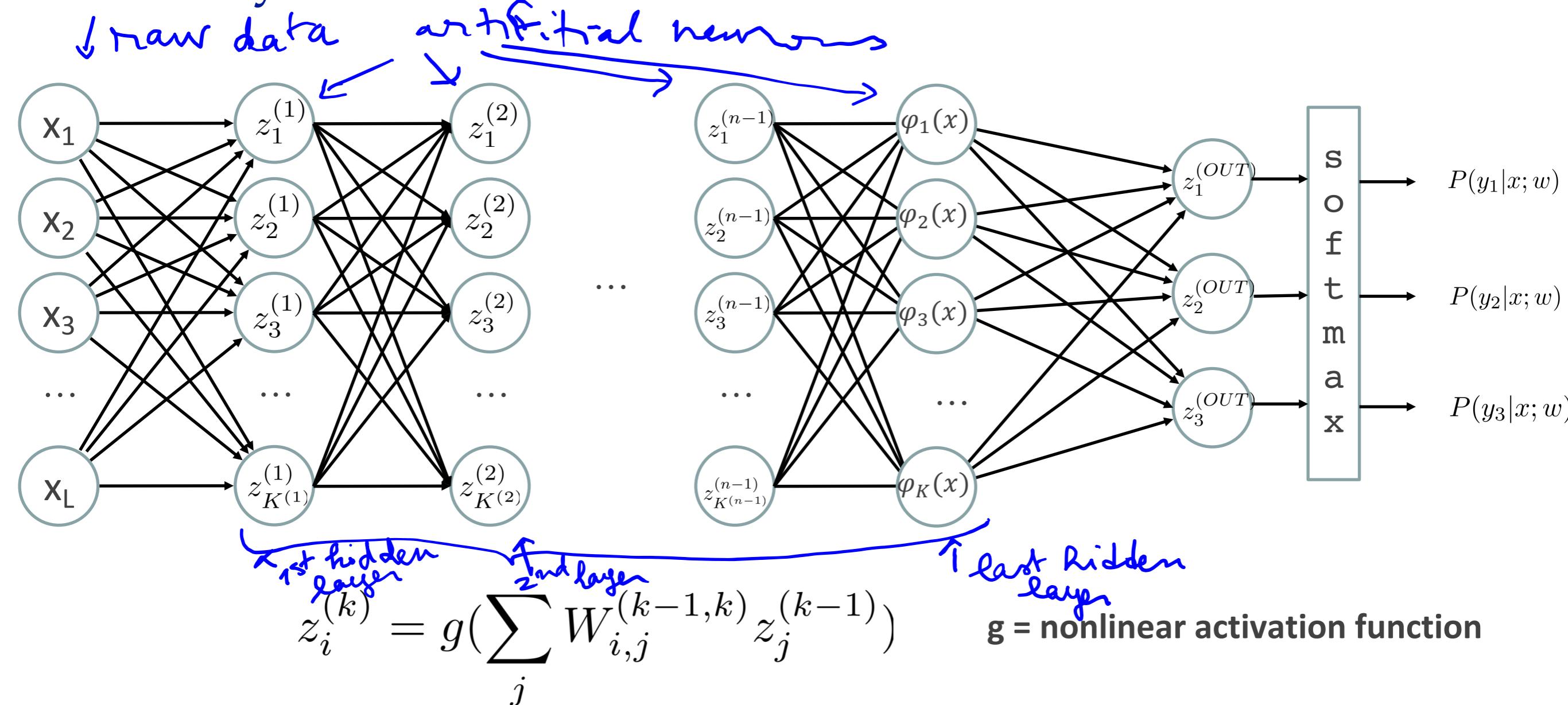


# Multi-class Logistic Regression



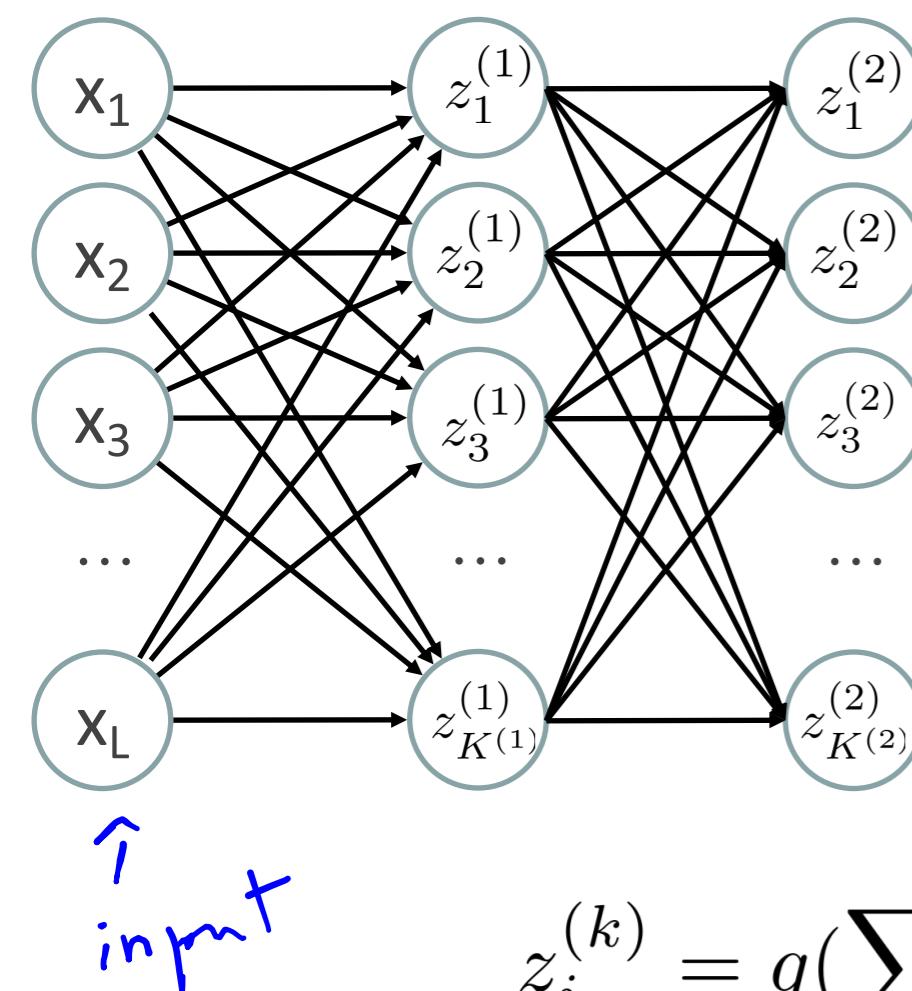
# (Deep) Neural Network

- ❖ Directly learns the features from data

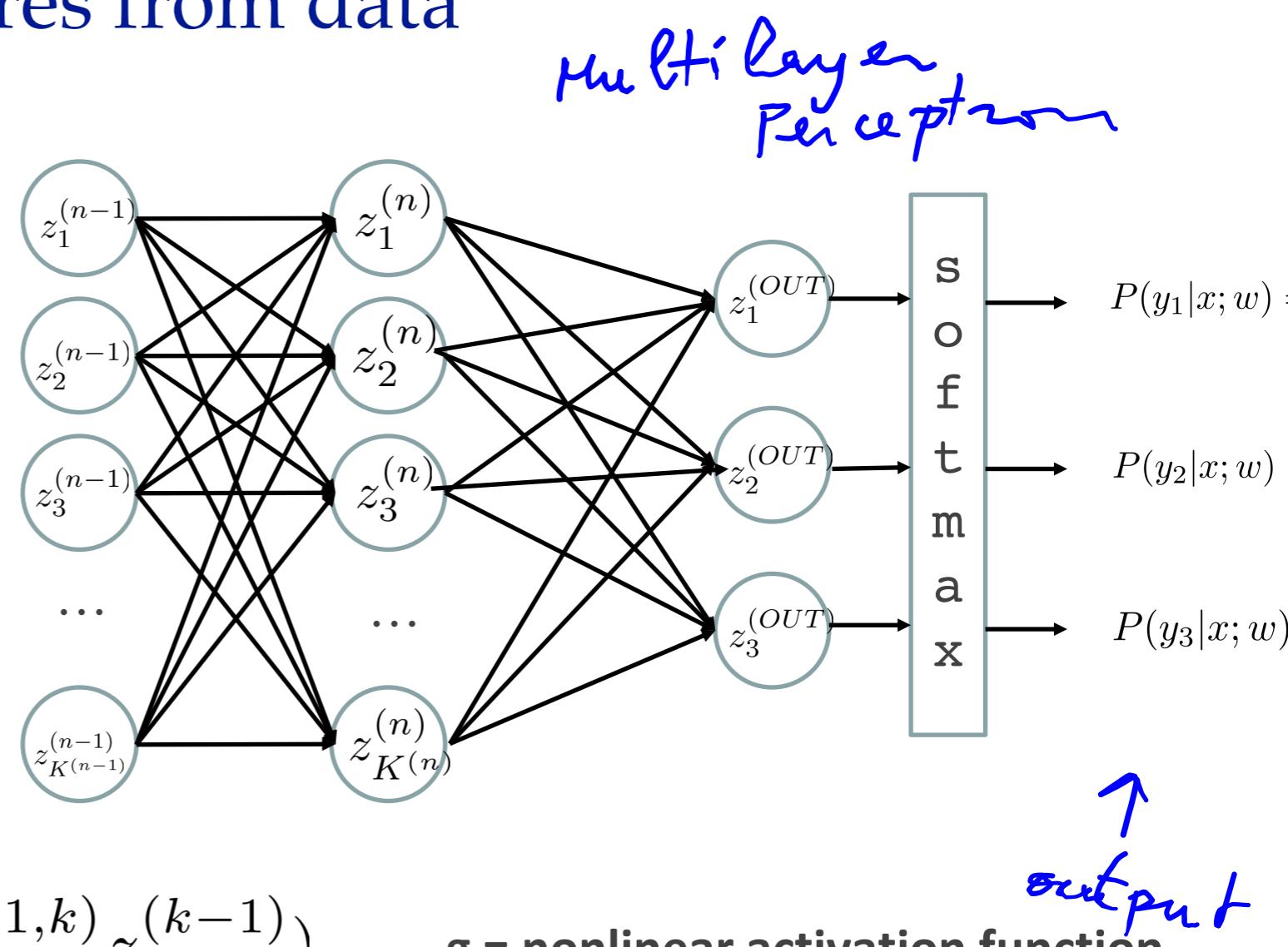


# (Deep) Neural Network

- ❖ Directly learns the features from data



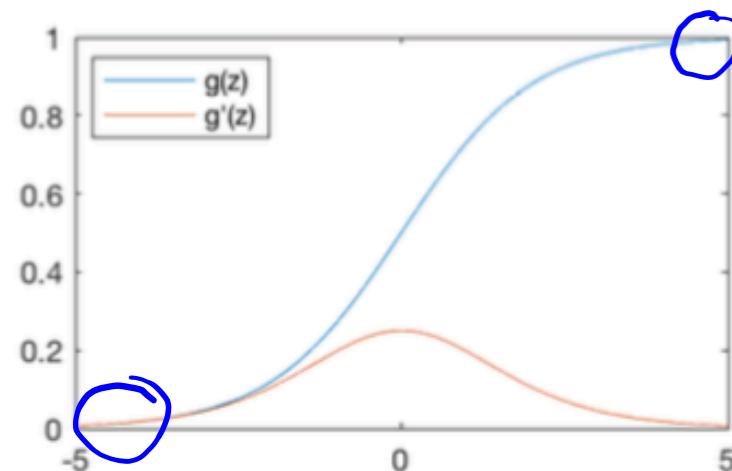
$$z_i^{(k)} = g\left(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)}\right)$$



**g = nonlinear activation function**

# Common Activation Functions

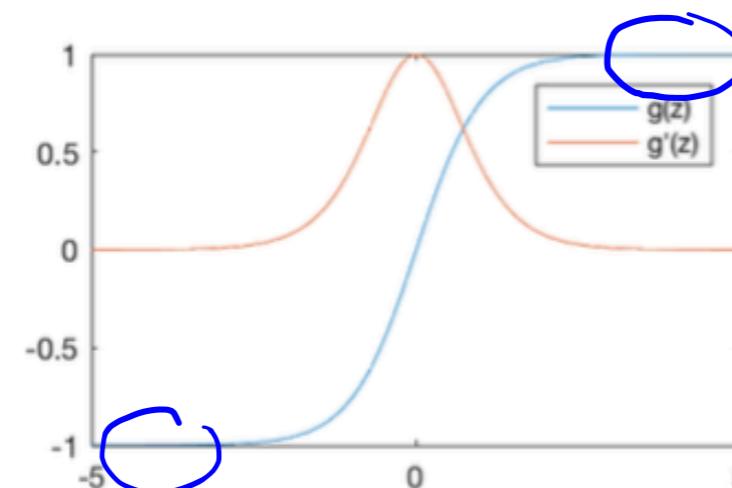
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}} \quad \checkmark$$

$$g'(z) = g(z)(1 - g(z))$$

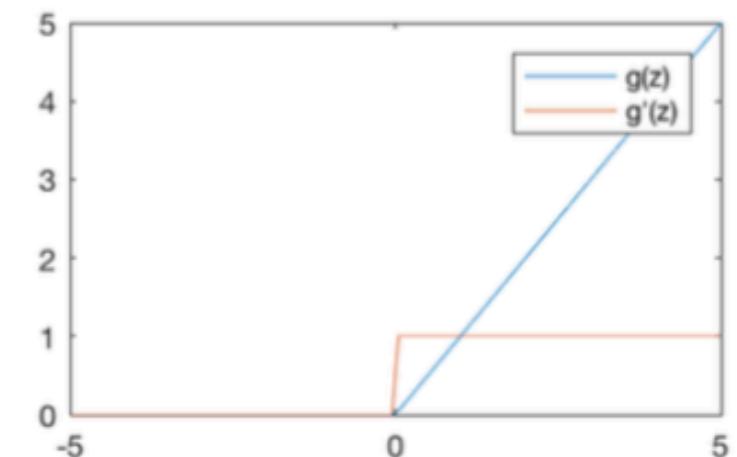
Hyperbolic Tangent



$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad \checkmark$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

[source: MIT 6.S191 introtodeeplearning.com]

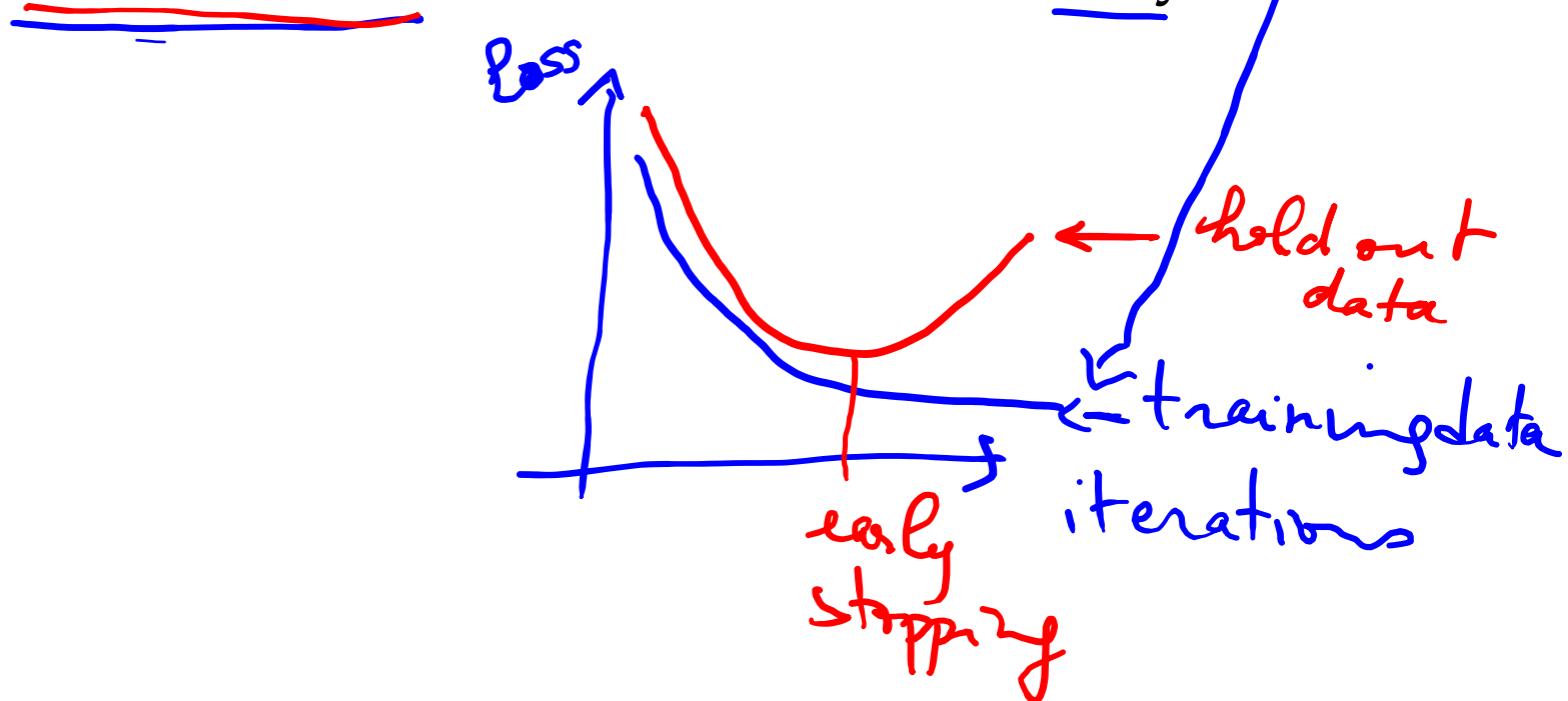
# Training a (Deep) Neural Network

- ❖ Training the deep neural network is just like logistic regression:

$$\min_{\underline{w}} - \underline{ll}(\underline{w}) = \min_{\underline{w}} - \sum_i \log P(y^{(i)} | x^{(i)}; \underline{w})$$

just  $\underline{w}$  tends to be a much, much larger vector

=> just run gradient method  
+ stop when log likelihood of hold-out data starts to decrease (early stopping)



# Quiz: Hyperparameter Tuning

---

- ❖ Can we tune the hyperparameters on the hold-out data used for early stopping?

# Neural Networks Properties

- ❖ **Theorem** (Universal Function Approximators). A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.
- ❖ Practical considerations
  - ❖ Can be seen as learning the features ✓
  - ❖ Large number of neurons
    - ❖ Danger for overfitting
    - ❖ (hence early stopping!)

# Neural Net Demo!

<https://playground.tensorflow.org/>

# How about computing all the derivatives?

- ❖ A neural network is just the composition of many functions:  $h_w(x) = f_n(f_{n-1}(\dots f_1(x) \dots))$
- ❖ Apply chain rule:
  - ❖  $f(x) = g(h(x))$
  - ❖  $f'(x) = h'(x)g'(h(x))$
- ❖  $\Rightarrow$  Derivatives can be computed by following well-defined procedures

# Automatic Differentiation

- ❖ Automatic differentiation software
  - ❖ e.g., Theano, TensorFlow, PyTorch ✓
  - ❖ Only need to program the function  $g(x, w)$  ✓
  - ❖ Can automatically compute all derivatives w.r.t. all entries in  $w$  ✓
  - ❖ This is typically done by caching info during forward computation pass of  $g$ , and then doing a backward pass = “backpropagation”
  - ❖ Autodiff / Backpropagation can often be done at computational cost comparable to the forward pass

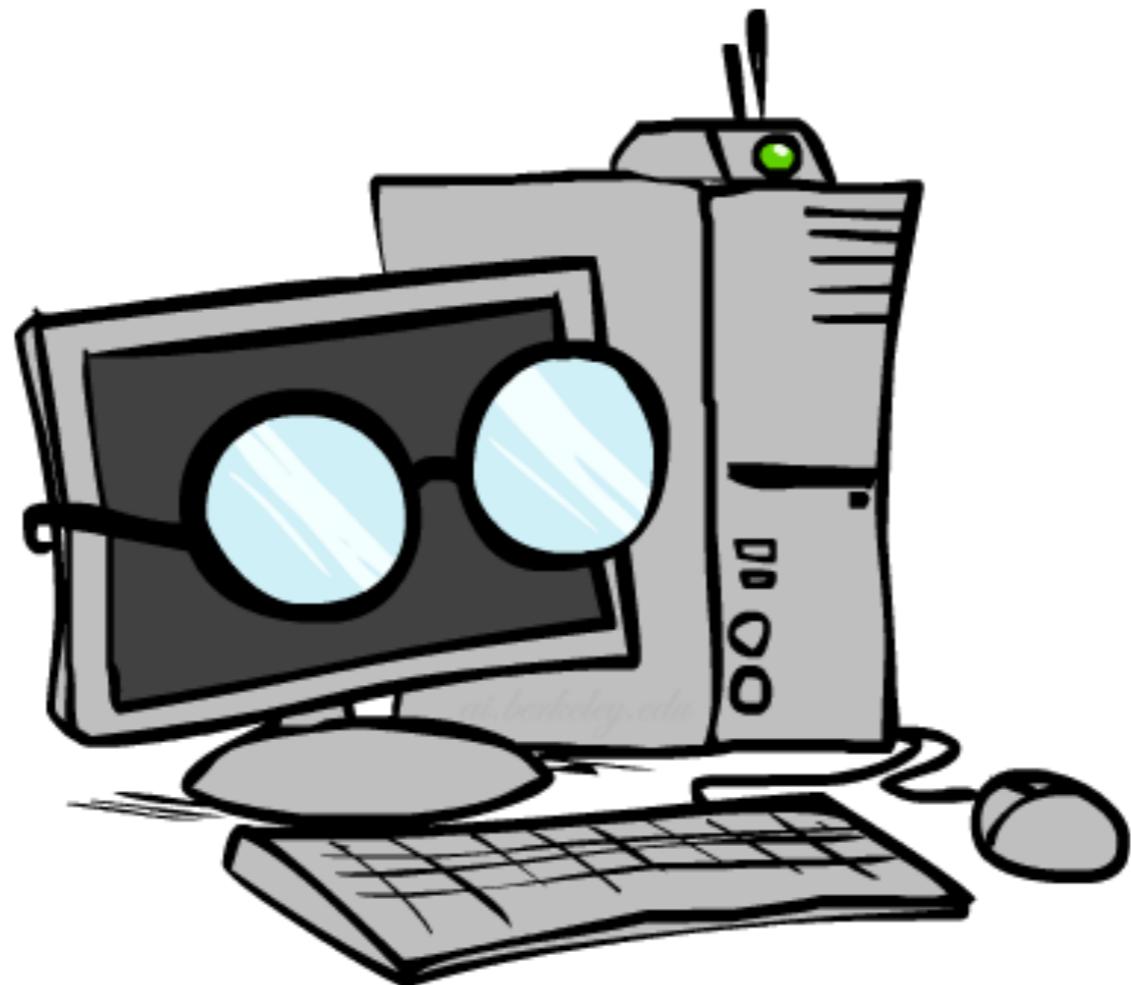
# Summary of Key Ideas

- ❖ Minimize empirical risk:  $\min_w \sum_i \ell(h_w(x^i), y^i)$  ✓
- ❖ Continuous optimization
  - ❖ Gradient descent: ✓
    - ❖ Compute steepest uphill direction = gradient (= just vector of partial derivatives)
    - ❖ Take step in the opposite gradient direction
    - ❖ Repeat (until held-out data accuracy starts to drop = “early stopping”) ✓
- ❖ Deep neural nets
  - ❖ Last layer returns expected output (e.g., probability of classes) ✓
  - ❖ Now also many more layers before this last layer
    - ❖ = computing the features
    - ❖ → the features are learned rather than hand-designed
  - ❖ Universal function approximation theorem ✓
    - ❖ If neural net is large enough
    - ❖ Then neural net can represent any continuous mapping from input to output with arbitrary accuracy
    - ❖ But remember: need to avoid overfitting / memorizing the training data → early stopping!
  - ❖ Automatic differentiation gives the derivatives efficiently ✓

---

# Applications

---



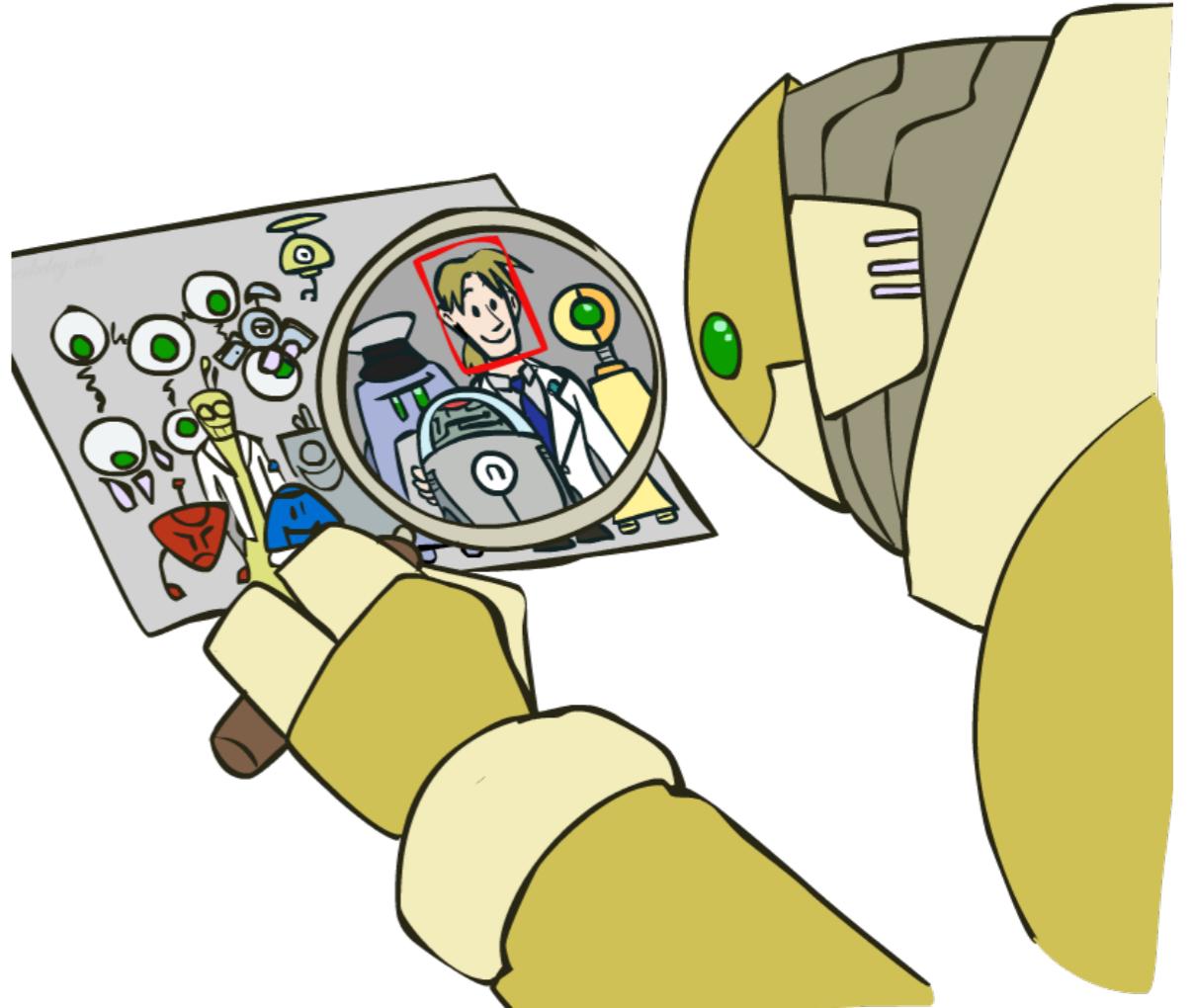
*Deep Neural Networks*

---

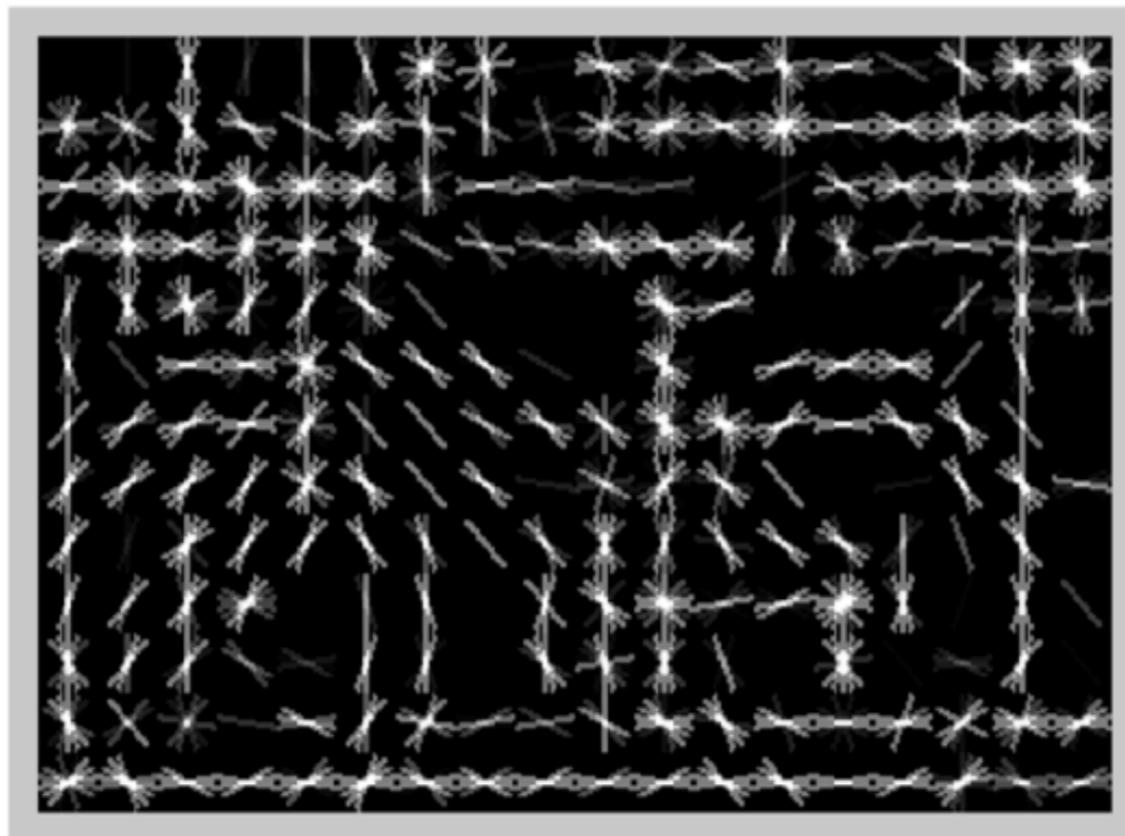
# Applications

---

Computer Vision  
Speech Recognition  
Machine Translation



# Features and Generalization

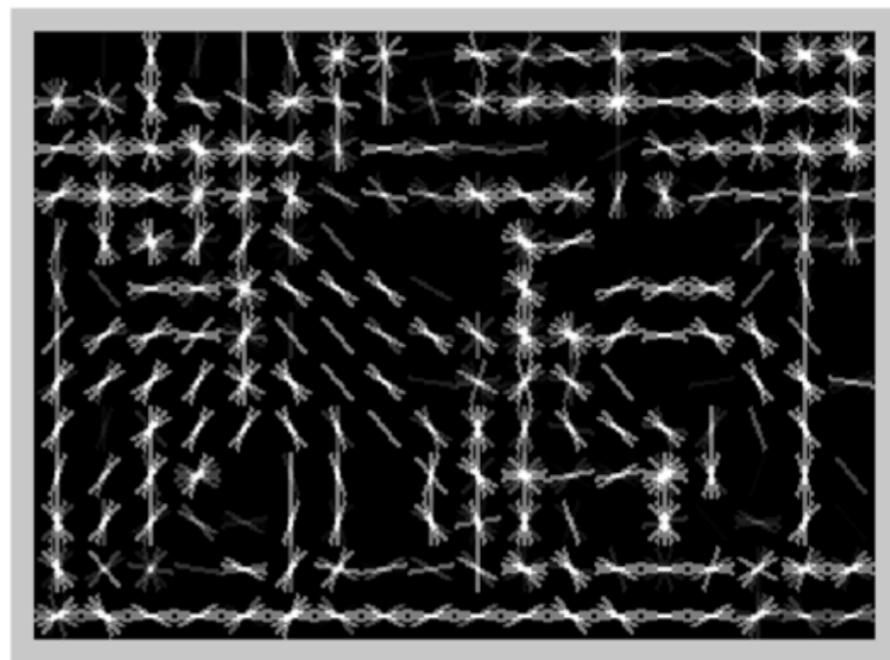


[HoG: Dalal and Triggs, 2005]

# Features and Generalization

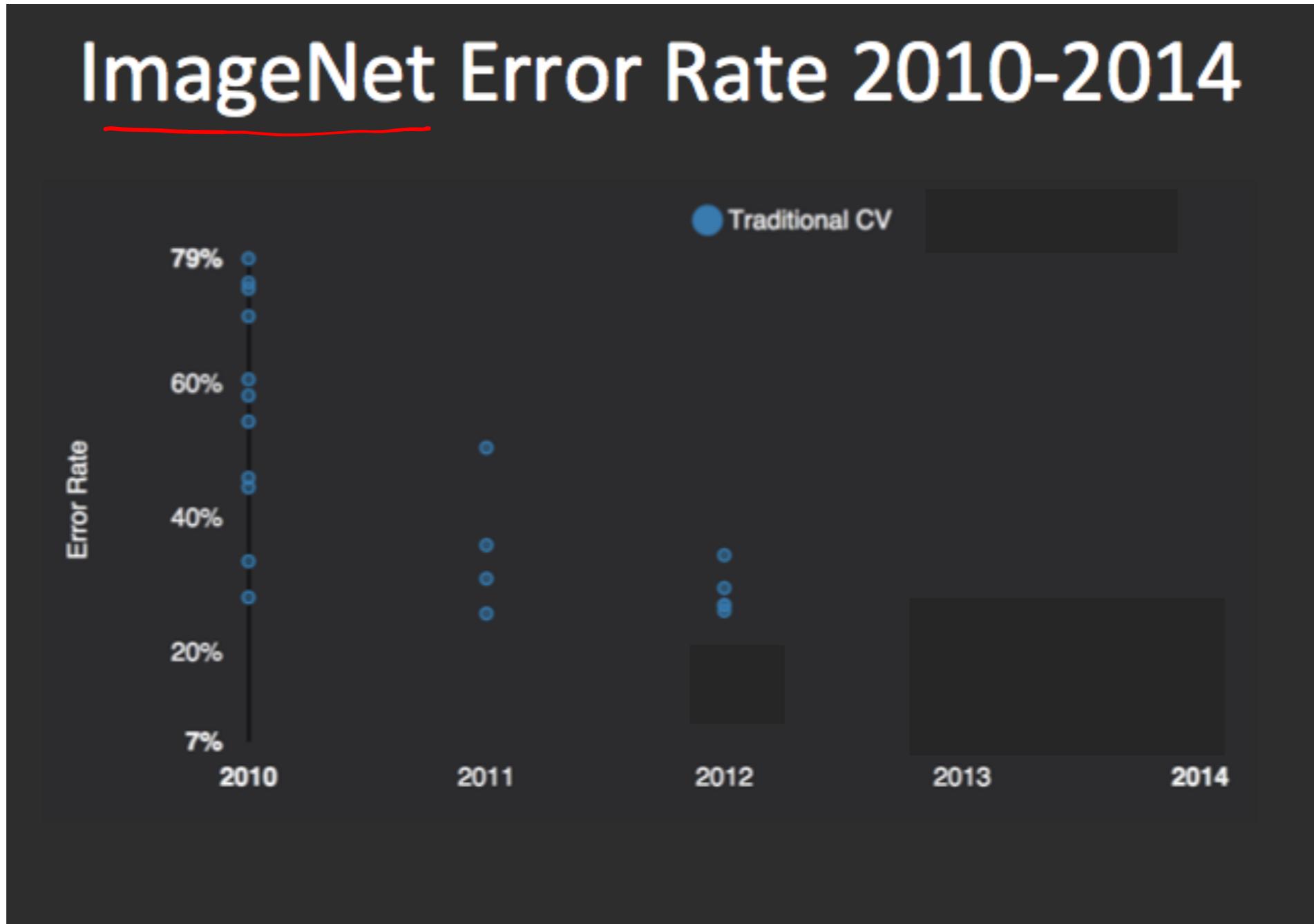


Image

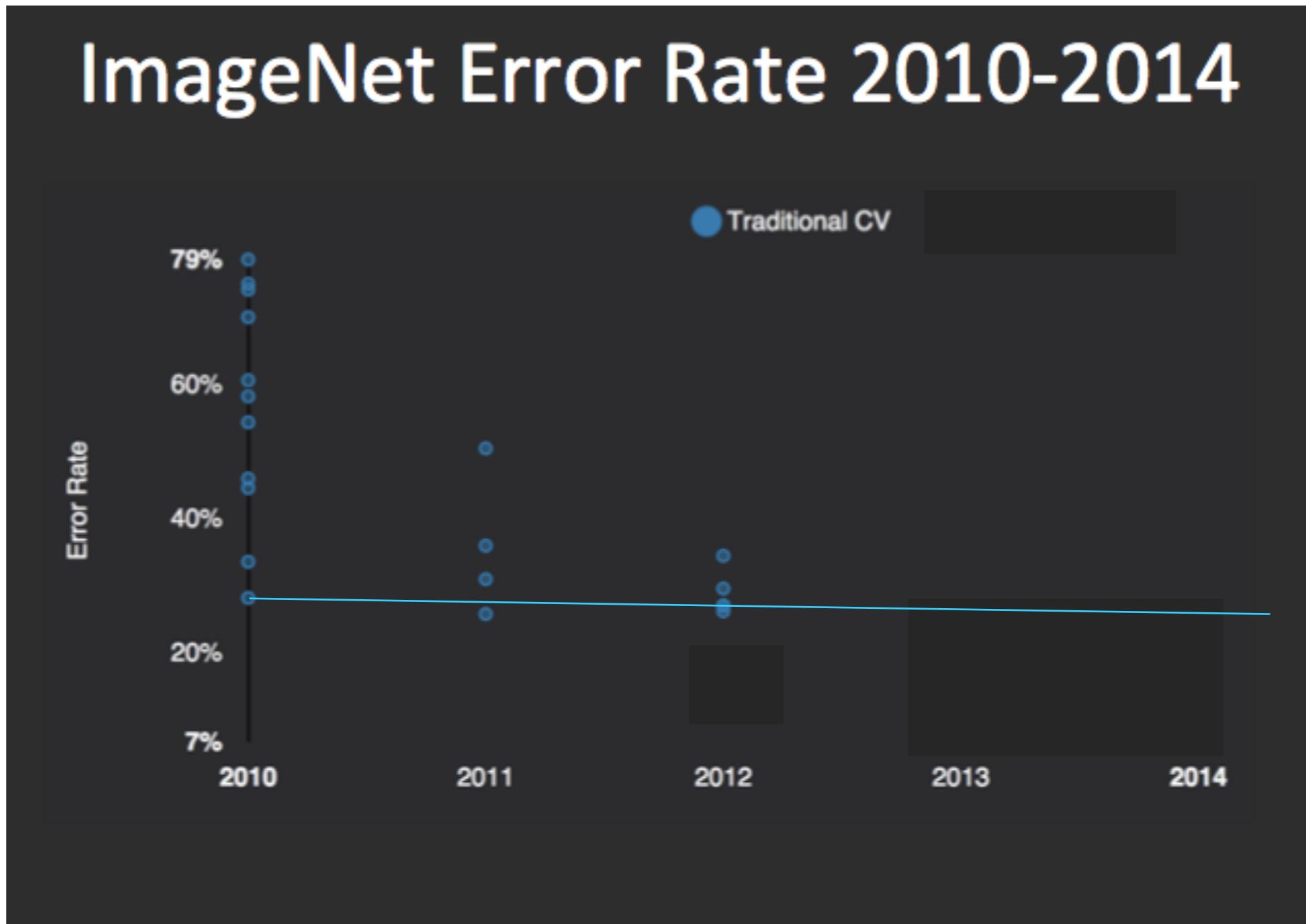


HoG

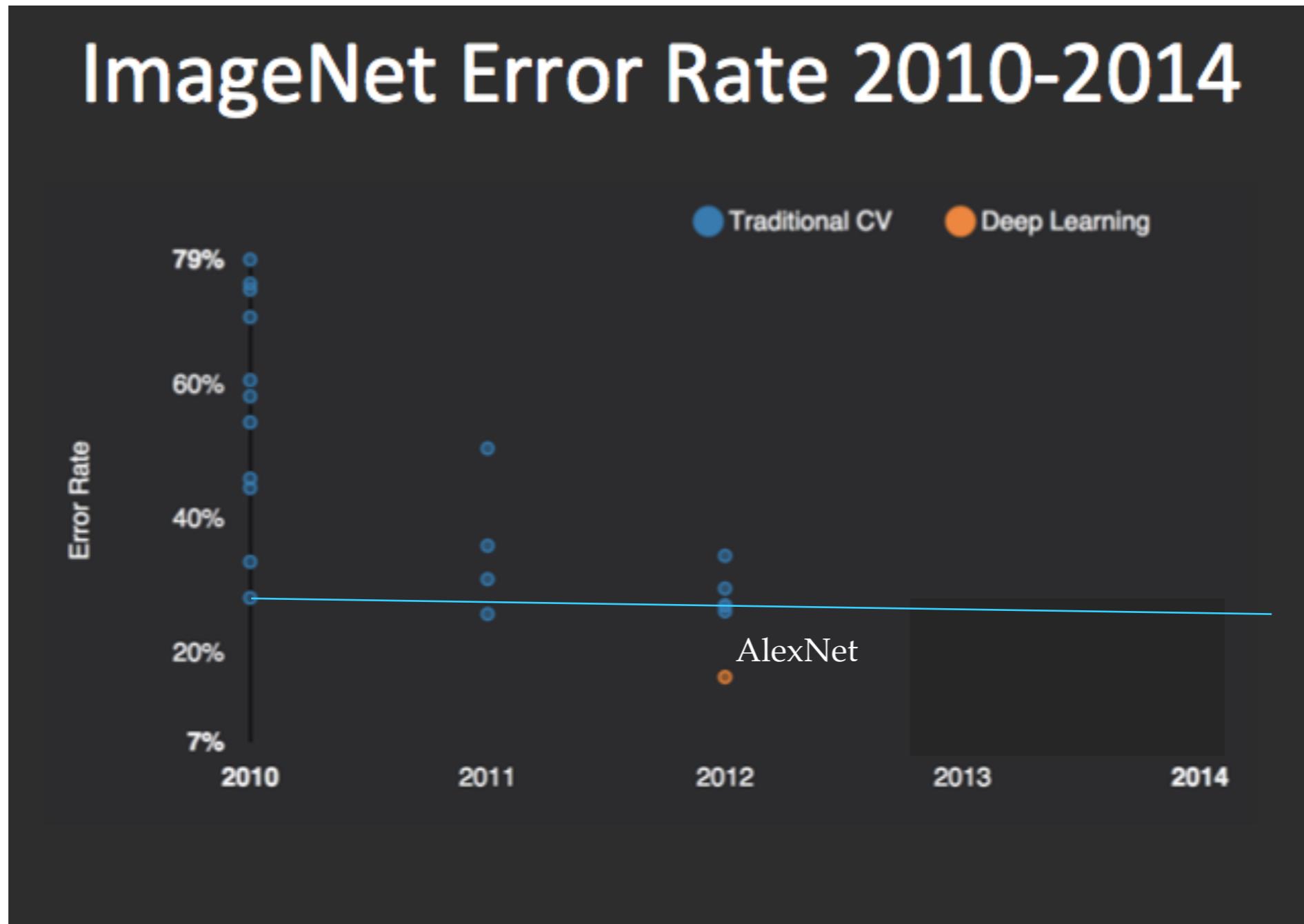
# Performance



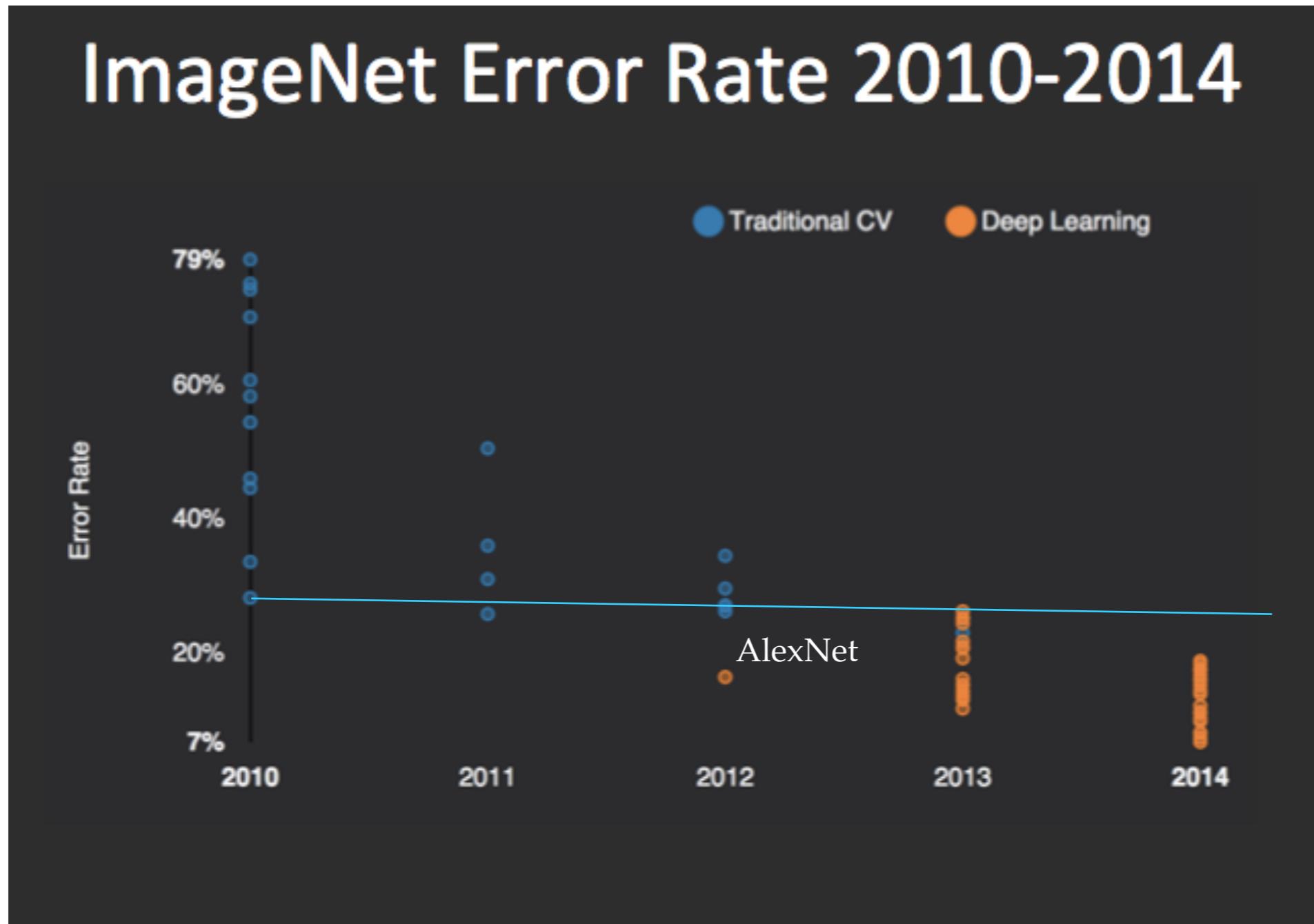
# Performance



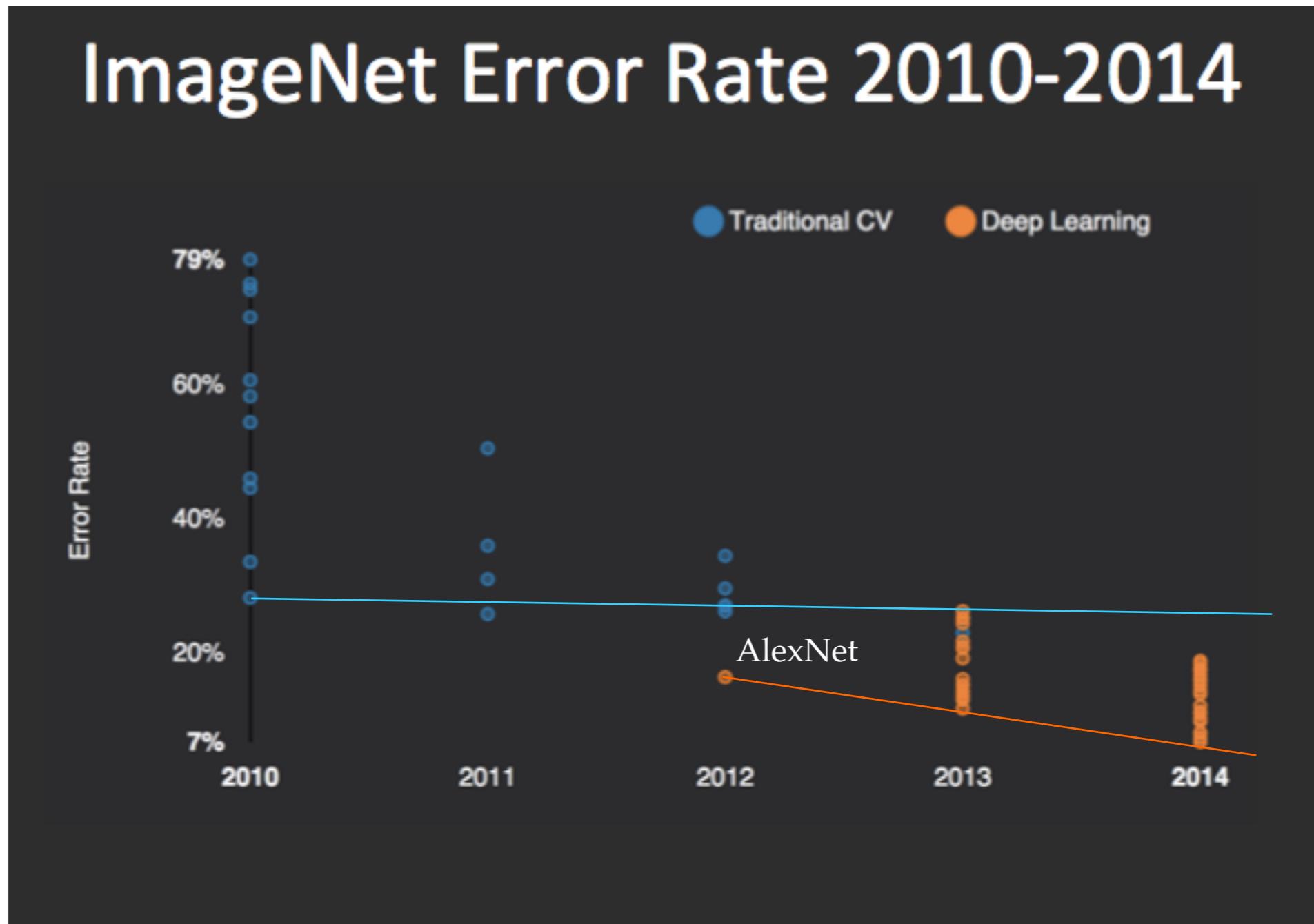
# Performance



# Performance



# Performance

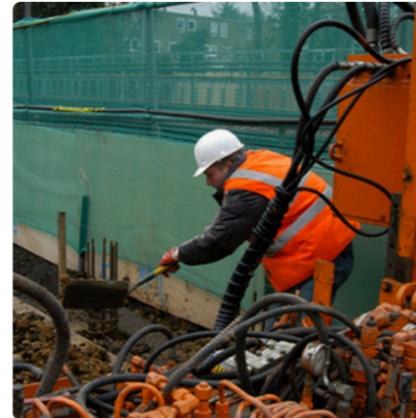


# MS COCO Image Captioning Challenge

$x \rightarrow y$   
sentence  
structured  
prediction



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



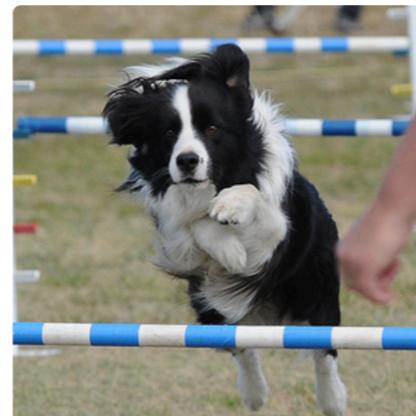
"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."

Karpathy & Fei-Fei, 2015; Donahue et al., 2015; Xu et al, 2015; many more

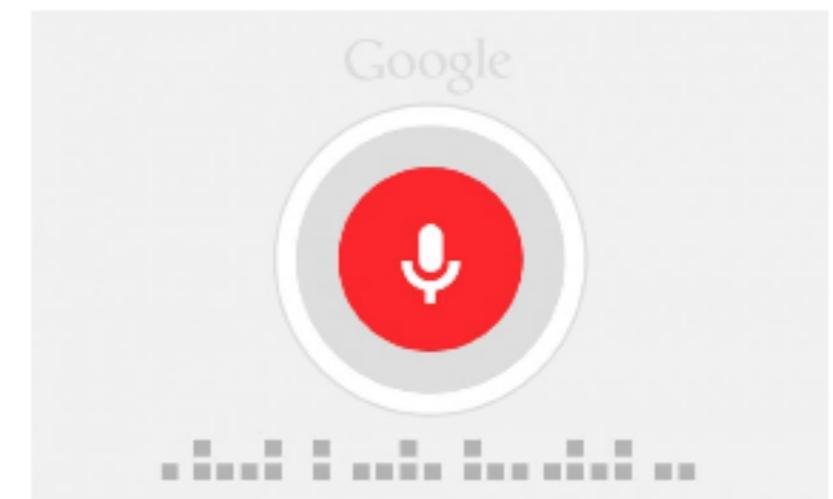
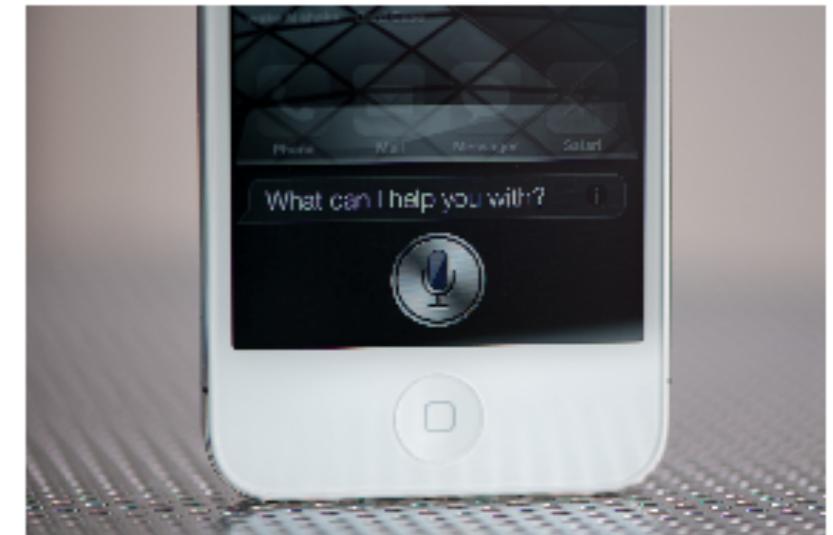
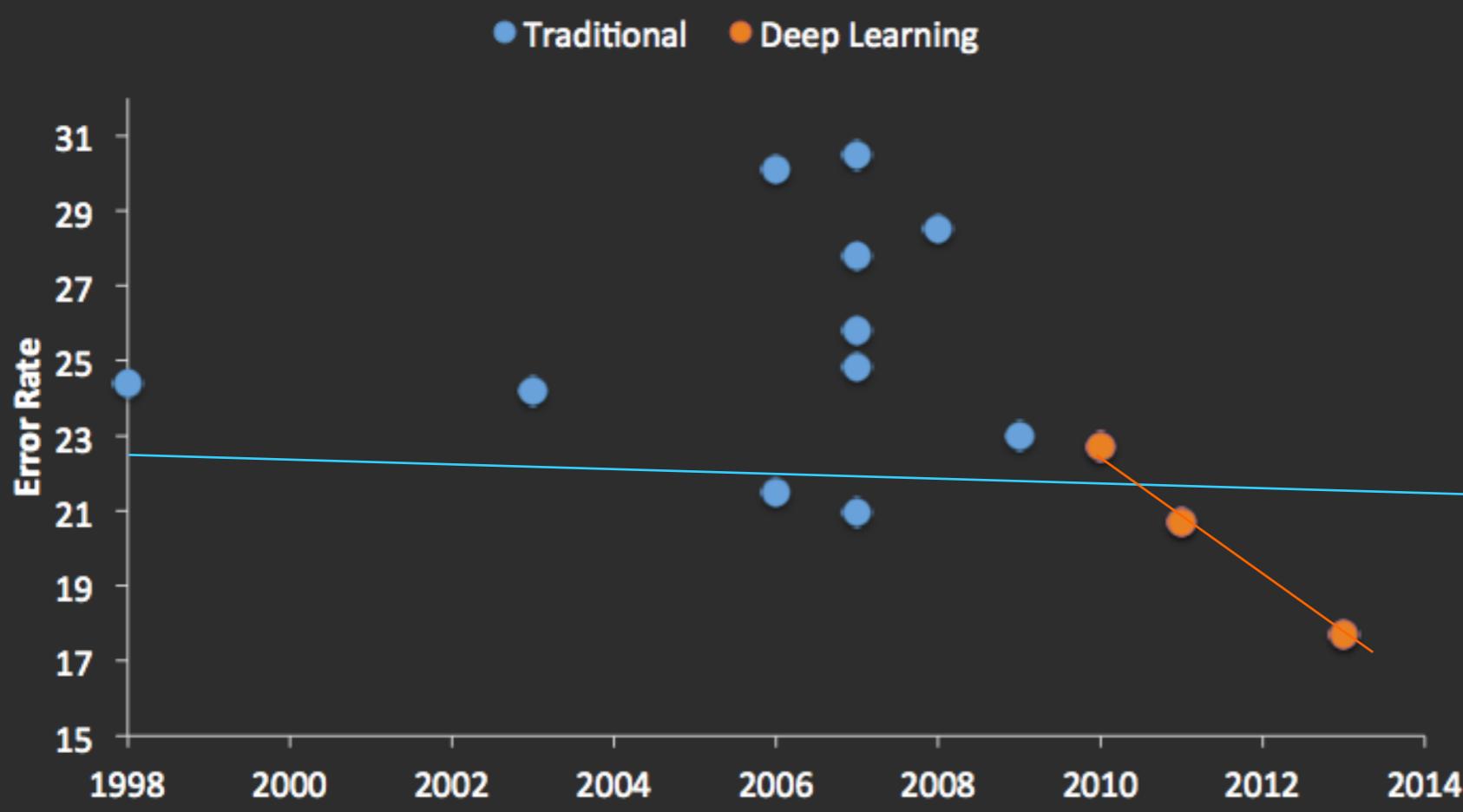
# Visual QA Challenge

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh

 <p>What vegetable is on the plate? Neural Net: <b>broccoli</b> Ground Truth: broccoli</p>	 <p>What color are the shoes on the person's feet ? Neural Net: <b>brown</b> Ground Truth: brown</p>	 <p>How many school busses are there? Neural Net: <b>2</b> Ground Truth: 2</p>	 <p>What sport is this? Neural Net: <b>baseball</b> Ground Truth: baseball</p>
 <p>What is on top of the refrigerator? Neural Net: <b>magnets</b> Ground Truth: cereal</p>	 <p>What uniform is she wearing? Neural Net: <b>shorts</b> Ground Truth: girl scout</p>	 <p>What is the table number? Neural Net: <b>4</b> Ground Truth: 40</p>	 <p>What are people sitting under in the back? Neural Net: <b>bench</b> Ground Truth: tent</p>

# Speech Recognition

## TIMIT Speech Recognition



*graph credit Matt Zeiler, Clarifai*

# Machine Translation

## Google Neural Machine Translation (in production)

