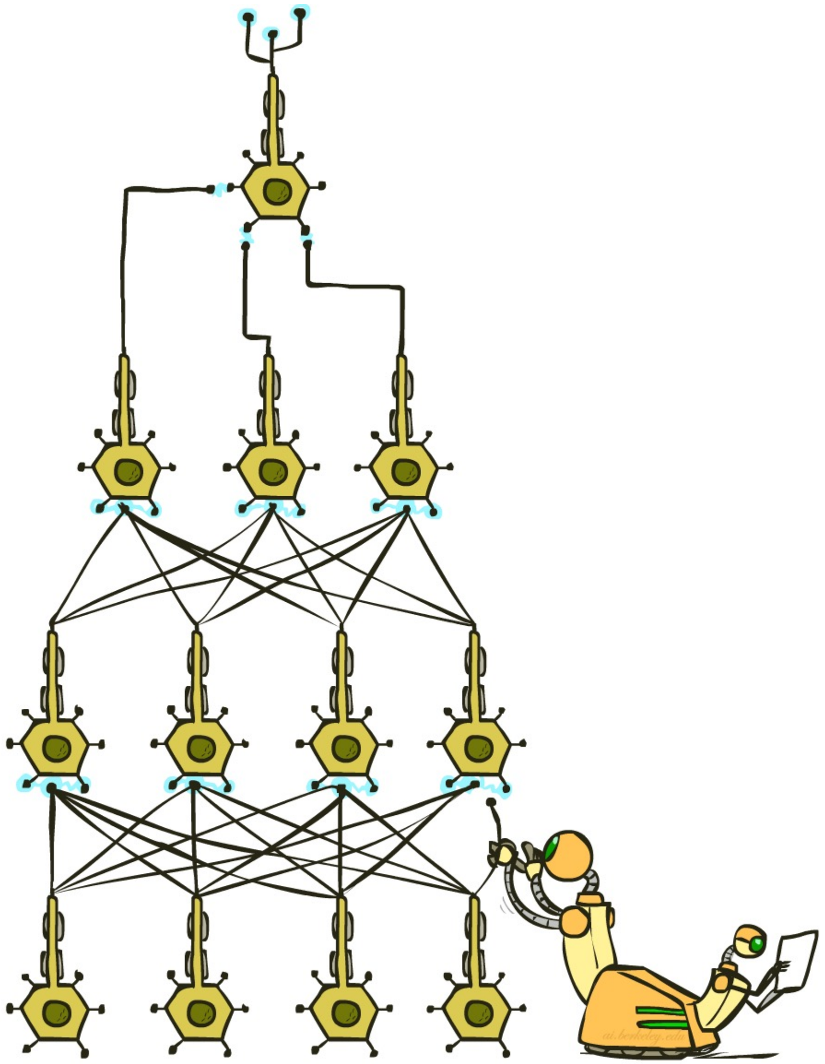


CS 343: Artificial Intelligence

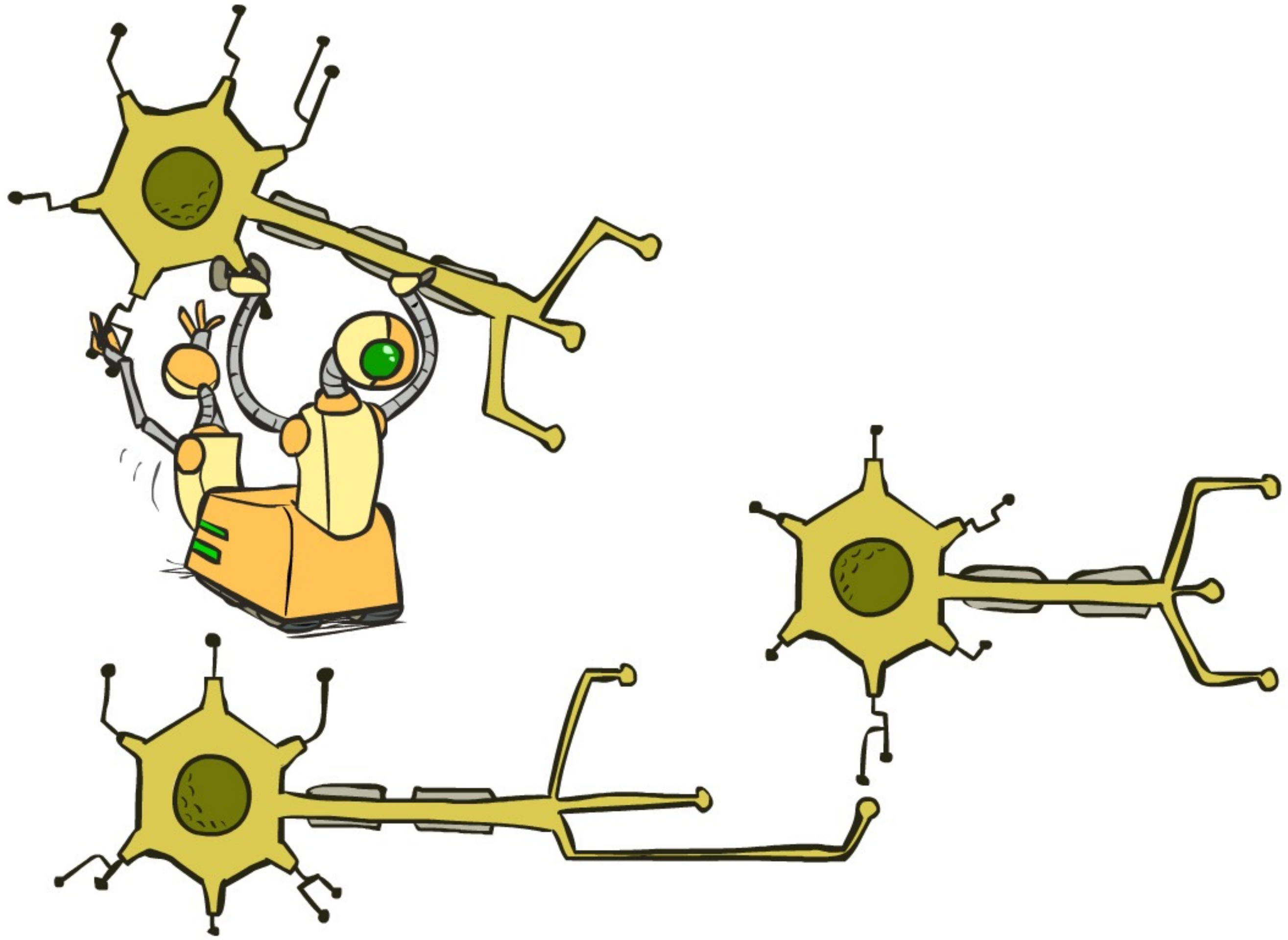
Deep Learning II

Prof. Yuke Zhu — The University of Texas at Austin

Neural Net Demo!

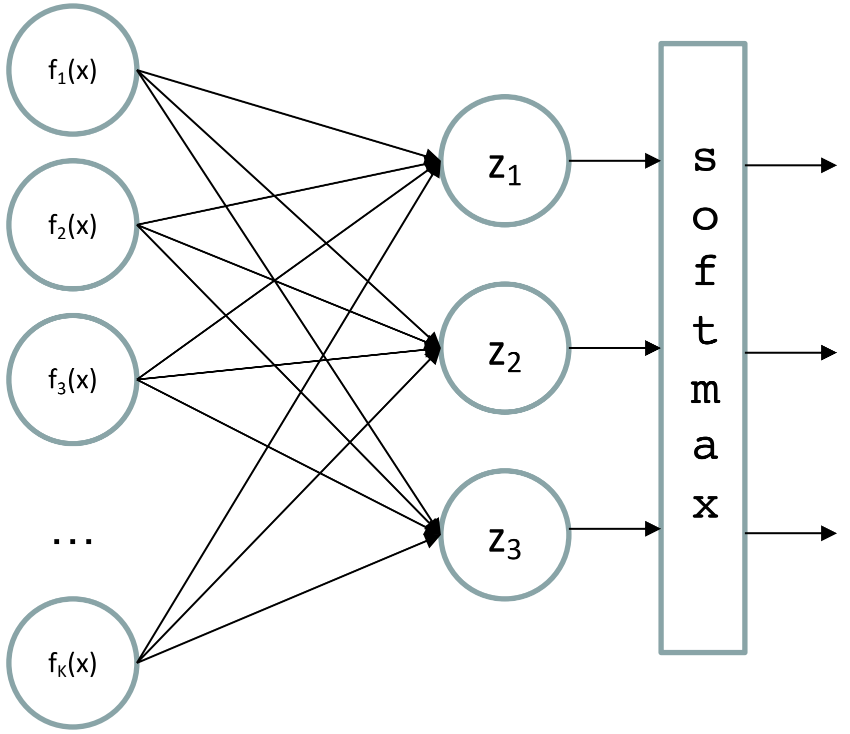
https://playground.tensorflow.org/

Neural Networks



Multi-class Logistic Regression

- = special case of neural network

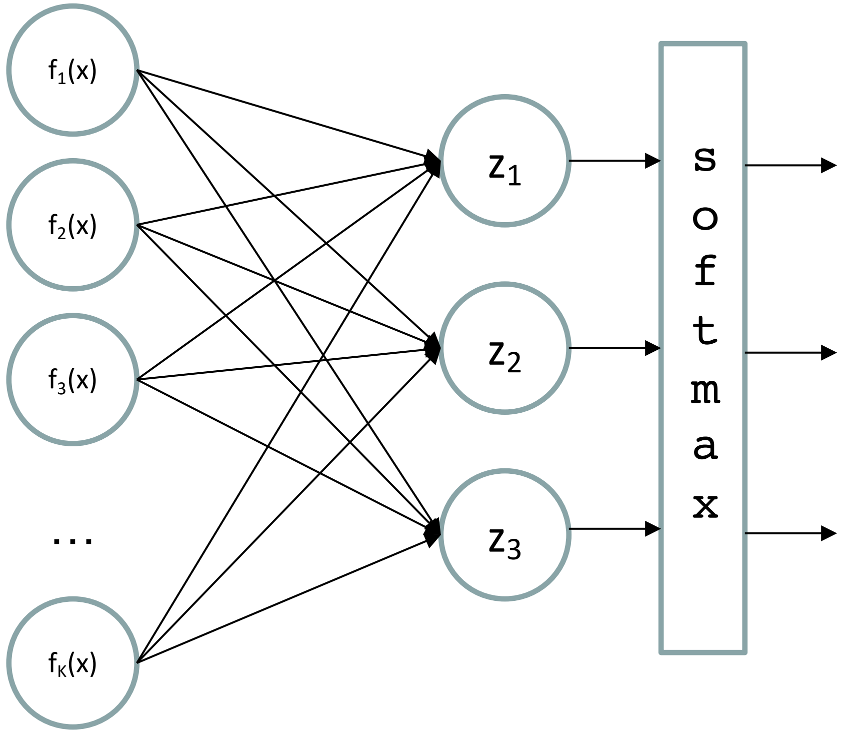


$$P(y_1|x;w) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$P(y_2|x;w) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$P(y_3|x;w) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Deep Neural Network = Also learn the features!

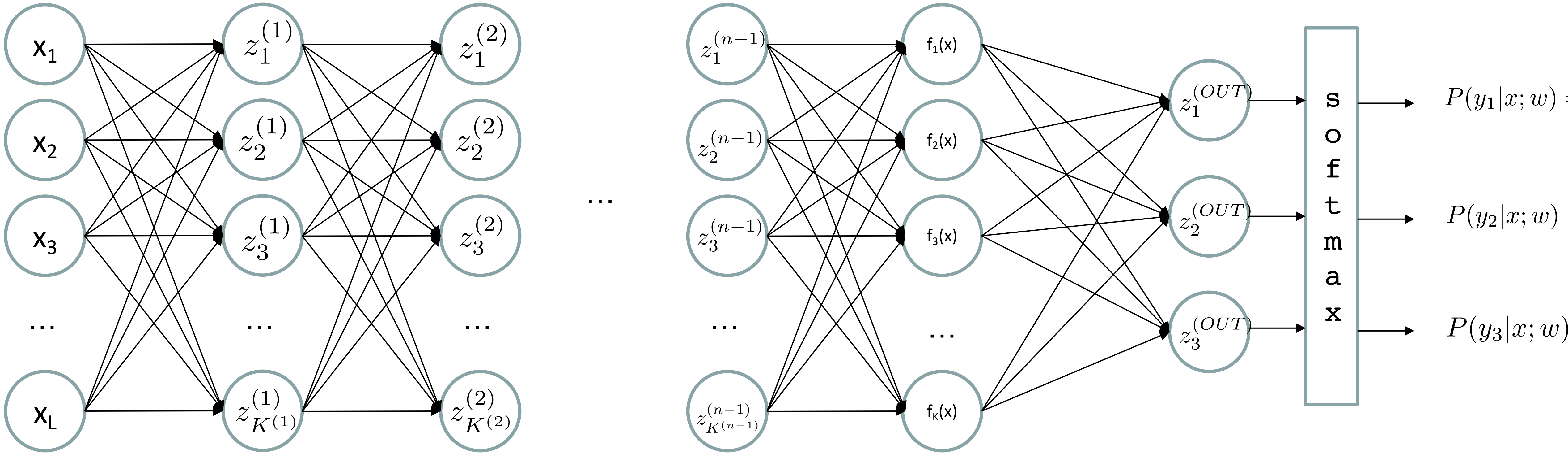


$$P(y_1|x;w) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$P(y_2|x;w) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$P(y_3|x;w) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

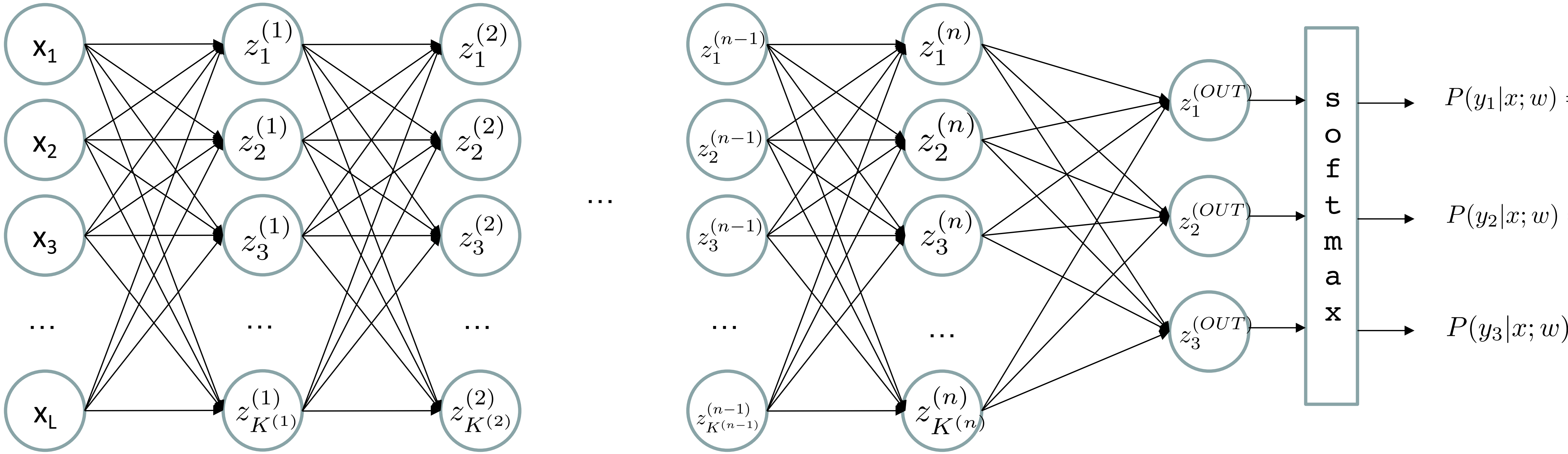
Deep Neural Network = Also learn the features!



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

g = nonlinear activation function

Deep Neural Network = Also learn the features!

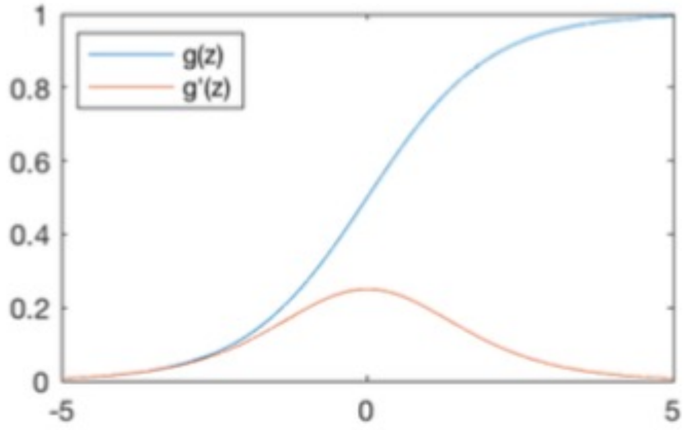


$$z_i^{(k)} = g\left(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)}\right)$$

g = nonlinear activation function

Common Activation Functions

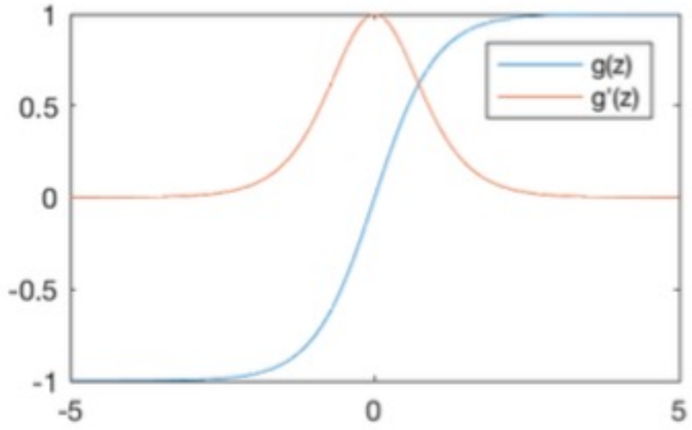
Sigmoid Function



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

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

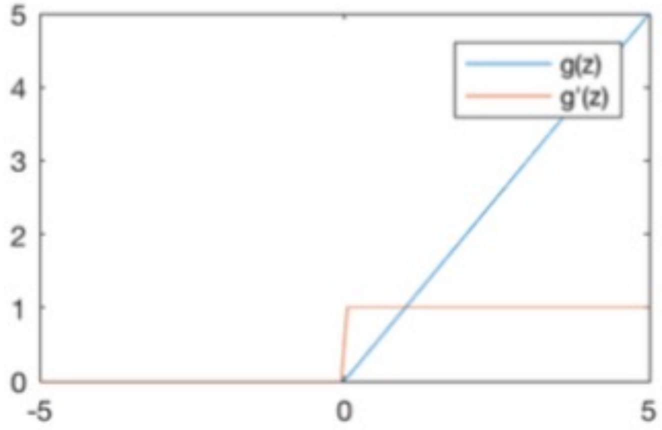
Hyperbolic Tangent



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

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

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]

Deep Neural Network: Also Learn the Features!

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

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

just w tends to be a much, much larger vector ☺

→just run gradient ascent

+ stop when log likelihood of hold-out data starts to decrease

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!)

How about computing all the derivatives?

- Derivatives tables:

$$\frac{d}{dx}(a)=0$$

$$\frac{d}{dx}(x)=1$$

$$\frac{d}{dx}(au)=a\frac{du}{dx}$$

$$\frac{d}{dx}(u+v-w)=\frac{du}{dx}+\frac{dv}{dx}-\frac{dw}{dx}$$

$$\frac{d}{dx}(uv)=u\frac{dv}{dx}+v\frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{u}{v}\right)=\frac{1}{v}\frac{du}{dx}-\frac{u}{v^2}\frac{dv}{dx}$$

$$\frac{d}{dx}(u^n)=nu^{n-1}\frac{du}{dx}$$

$$\frac{d}{dx}(\sqrt{u})=\frac{1}{2\sqrt{u}}\frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{1}{u}\right)=-\frac{1}{u^2}\frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{1}{u^n}\right)=-\frac{n}{u^{n+1}}\frac{du}{dx}$$

$$\frac{d}{dx}[f(u)]=\frac{d}{du}[f(u)]\frac{du}{dx}$$

$$\frac{d}{dx}[\ln u]=\frac{d}{dx}[\log_e u]=\frac{1}{u}\frac{du}{dx}$$

$$\frac{d}{dx}\left[\log_a u\right]=\log_a e\frac{1}{u}\frac{du}{dx}$$

$$\frac{d}{dx}e^u=e^u\frac{du}{dx}$$

$$\frac{d}{dx}a^u=a^u\ln a\frac{du}{dx}$$

$$\frac{d}{dx}\left(u^v\right)=vu^{v-1}\frac{du}{dx}+\ln u\ \ u^v\frac{dv}{dx}$$

$$\frac{d}{dx}\sin u=\cos u\frac{du}{dx}$$

$$\frac{d}{dx}\cos u=-\sin u\frac{du}{dx}$$

$$\frac{d}{dx}\tan u=\sec^2 u\frac{du}{dx}$$

$$\frac{d}{dx}\cot u=-\csc^2 u\frac{du}{dx}$$

$$\frac{d}{dx}\sec u=\sec u\tan u\frac{du}{dx}$$

$$\frac{d}{dx}\csc u=-\csc u\cot u\frac{du}{dx}$$

[source: http://hyperphysics.phy-astr.gsu.edu/hbase/Math/derfunc.html

How about computing all the derivatives?

- But neural net f is never one of those?
  - No problem: CHAIN RULE:

If $$f(x) = g(h(x))$$

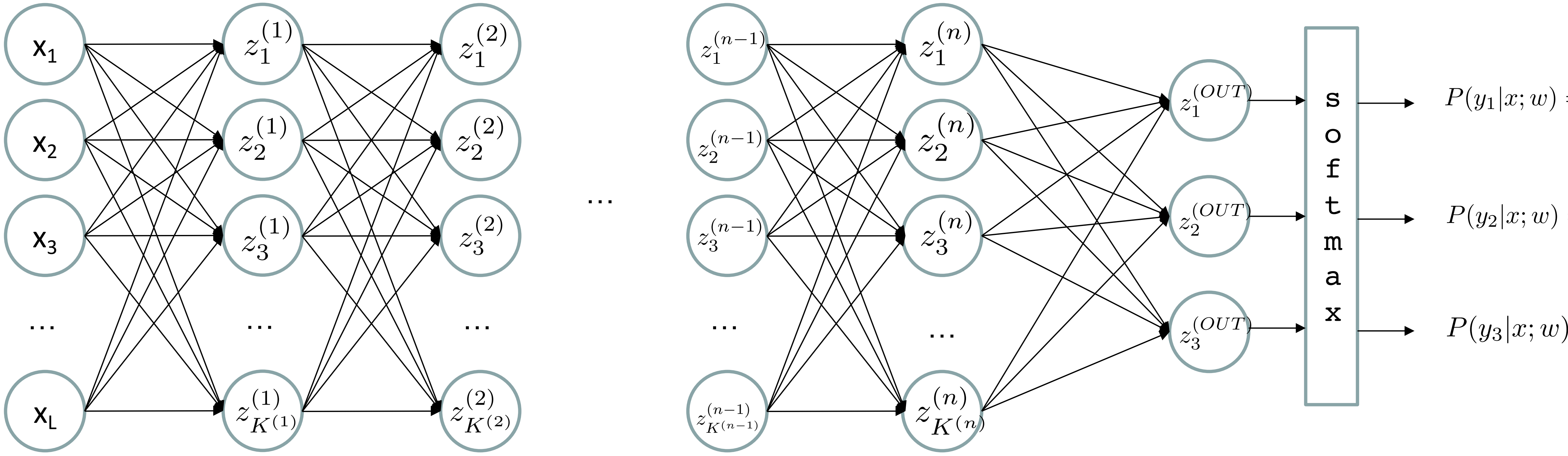
Then $$f'(x) = g'(h(x))h'(x)$$

→ **Derivatives can be computed by following well-defined procedures**

Automatic Differentiation

- Automatic differentiation software
  - e.g. Theano, TensorFlow, PyTorch, Chainer
  - Only need to program the function g(x,y,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 f, and then doing a backward pass = “backpropagation”
  - Autodiff / Backpropagation can often be done at computational cost comparable to the forward pass
- Need to know this exists
- How this is done? -- outside of scope of CS343

Training a Network (setting weights)



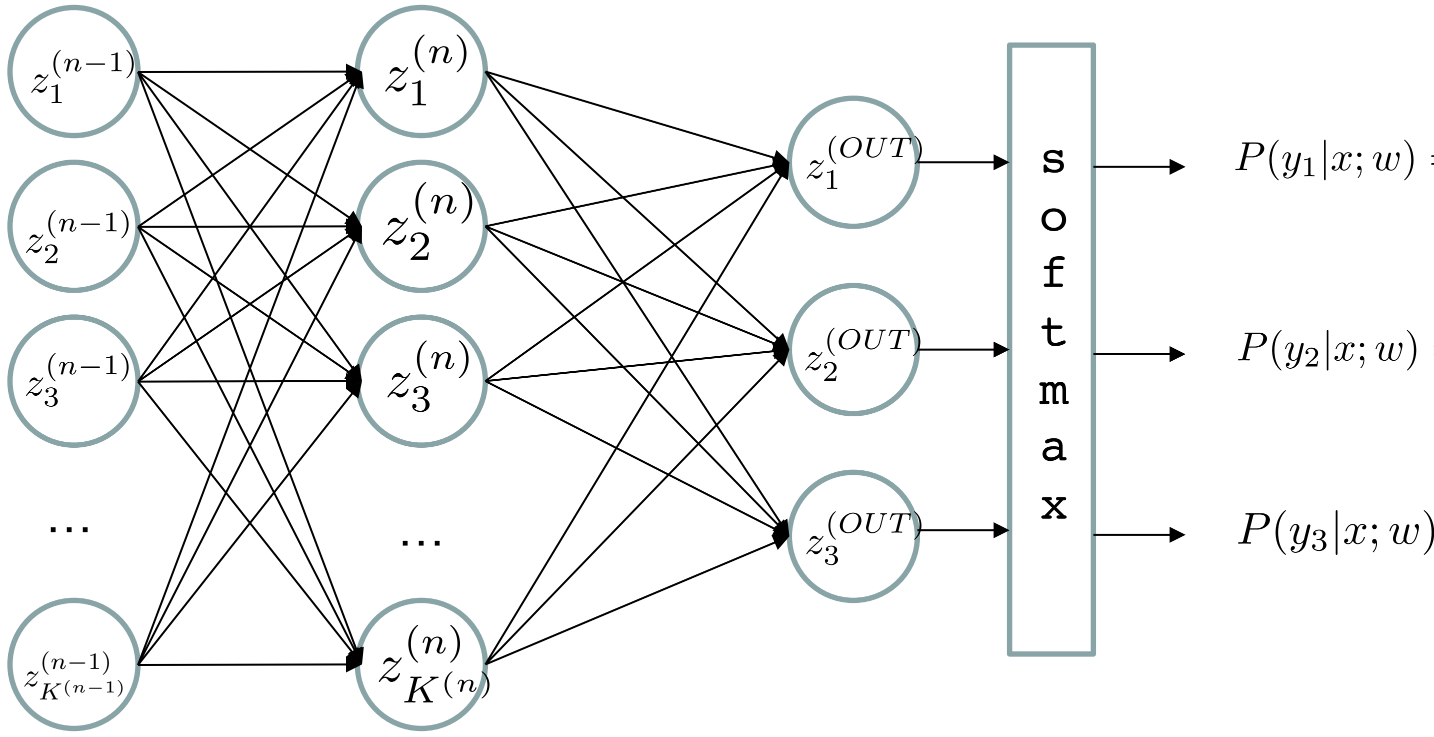
$$z_i^{(k)} = g(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)})$$

g = nonlinear activation function

Training a Network

Key words:

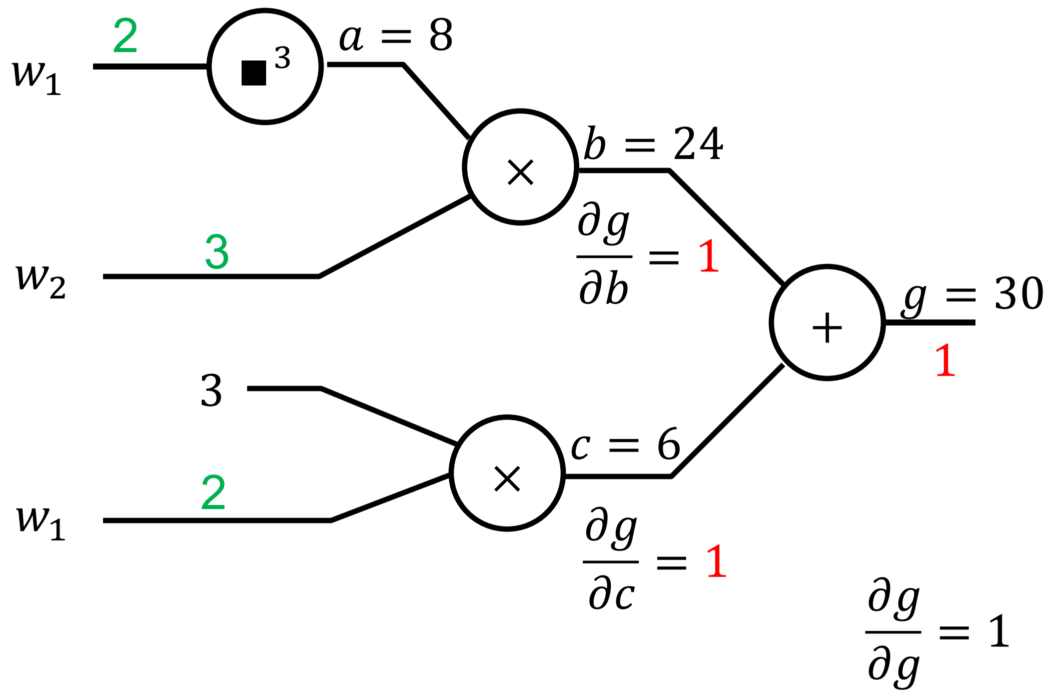
- Forward
- Backwards
- Gradient
- Backprop



g = nonlinear activation function

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

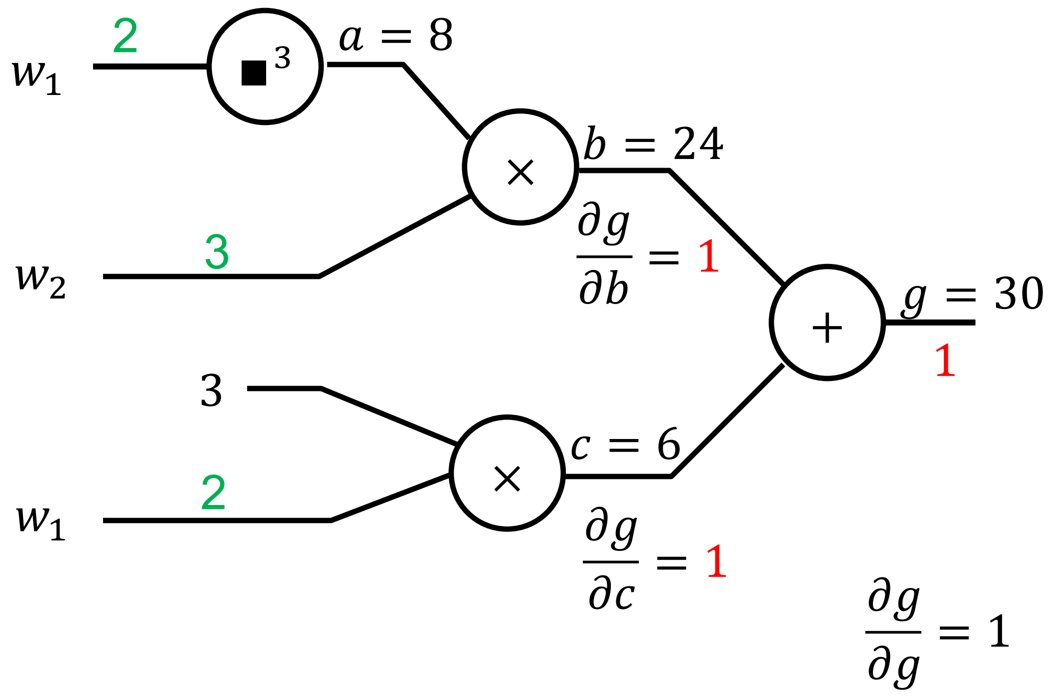
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g / \partial w_1$ and $\partial g / \partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$



w_1
2
$\blacksquare^3$
$a = 8$
$\times$
$b = 24$
w_2
3
$\frac{\partial g}{\partial b} = 1$
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

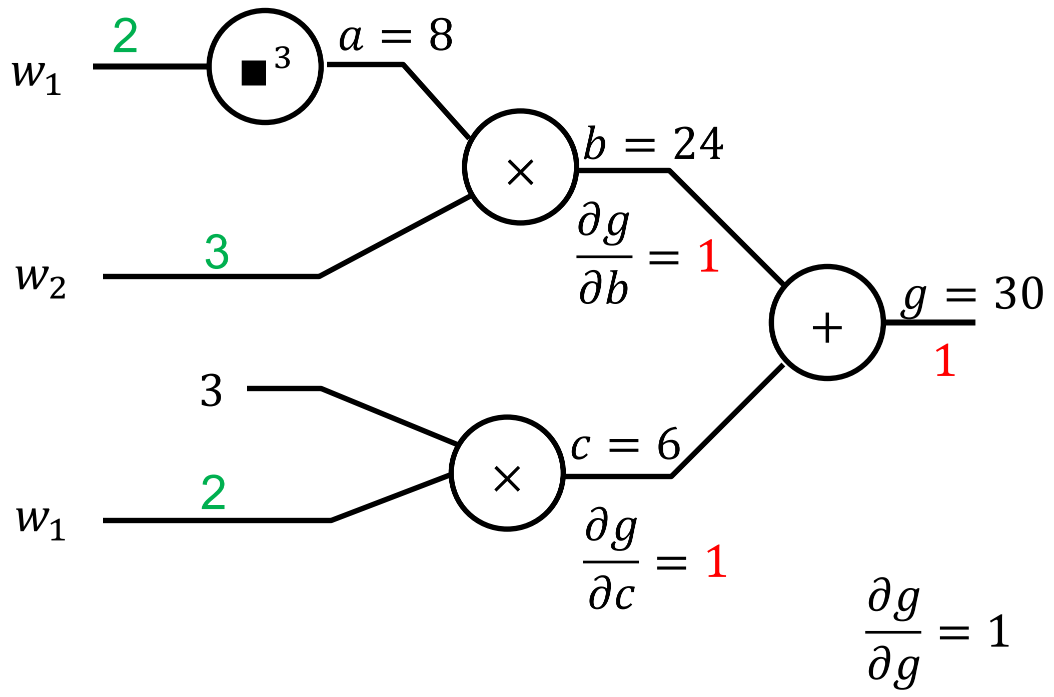
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g/\partial w_1$ and $\partial g/\partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a}$



w_1
2
$\blacksquare^3$
$a = 8$
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

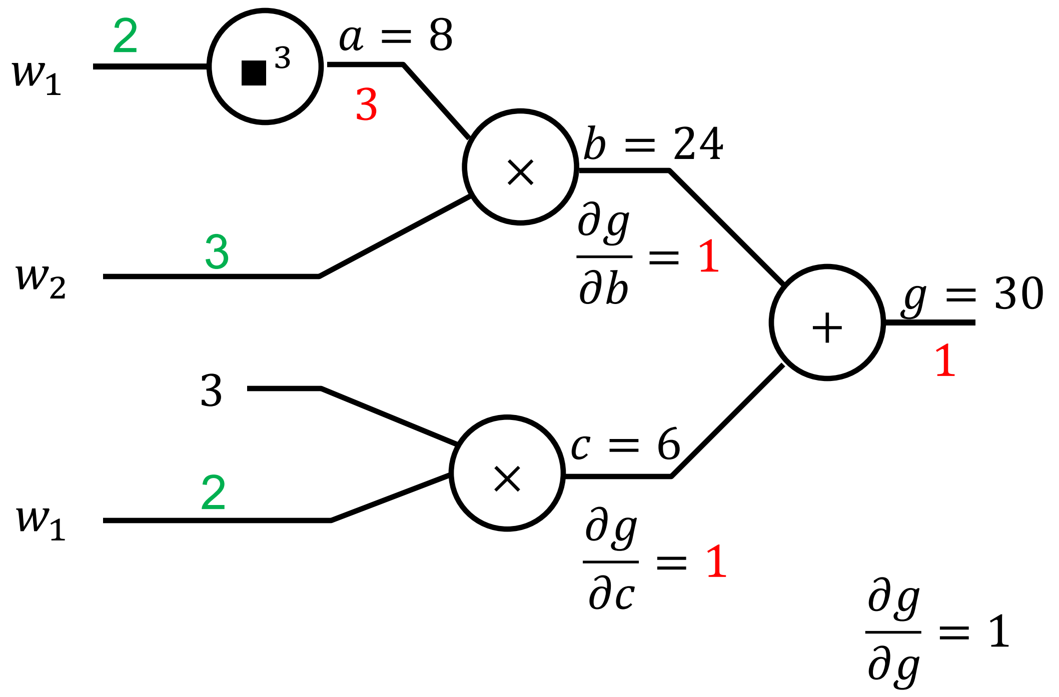
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2,3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g/\partial w_1$ and $\partial g/\partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = ?????$



w_1
2
$\blacksquare^3$
$a = 8$
w_2
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
$+$
$g = 30$
1
3
w_1
2
$\times$
$c = 6$
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

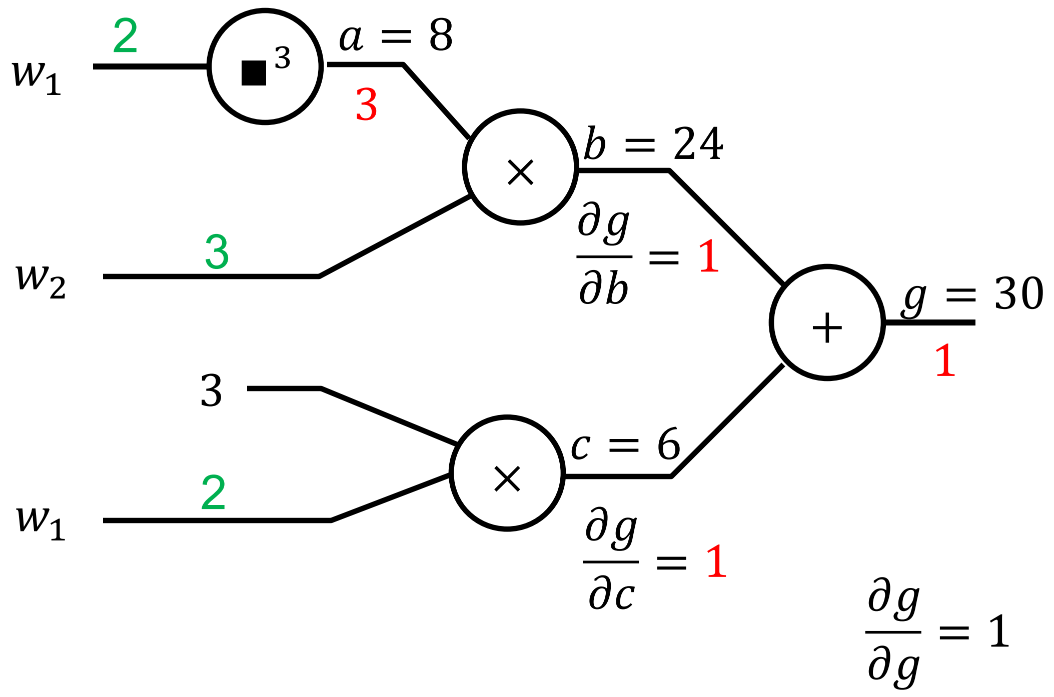
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g / \partial w_1$ and $\partial g / \partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b} \frac{\partial b}{\partial a} = 1 \frac{\partial b}{\partial a} = 1 \cdot 3 = 3$



w_1
2
$\blacksquare^3$
$a = 8$
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

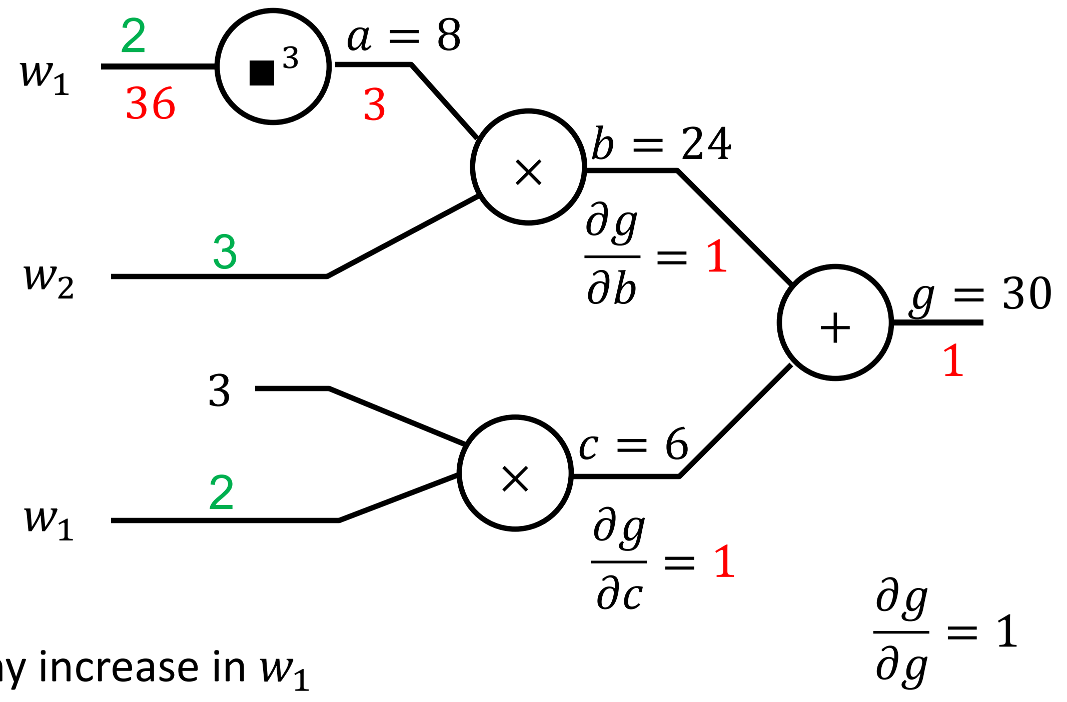
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g / \partial w_1$ and $\partial g / \partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = 1 \frac{\partial b}{\partial a} = 1 \cdot 3 = 3$
- $a = w_1^3$
  - $\frac{\partial g}{\partial w_1} = ?????$



w_1
2
$\blacksquare^3$
$a = 8$
3
w_2
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
$+$
$g = 30$
1
3
w_1
2
$\times$
$c = 6$
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g/\partial w_1$ and $\partial g/\partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = 1\,\frac{\partial b}{\partial a} = 1 \cdot 3 = 3$
- $a = w_1^3$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial a}\frac{\partial a}{\partial w_1} = 3 \cdot 3w_1^2 = 36$

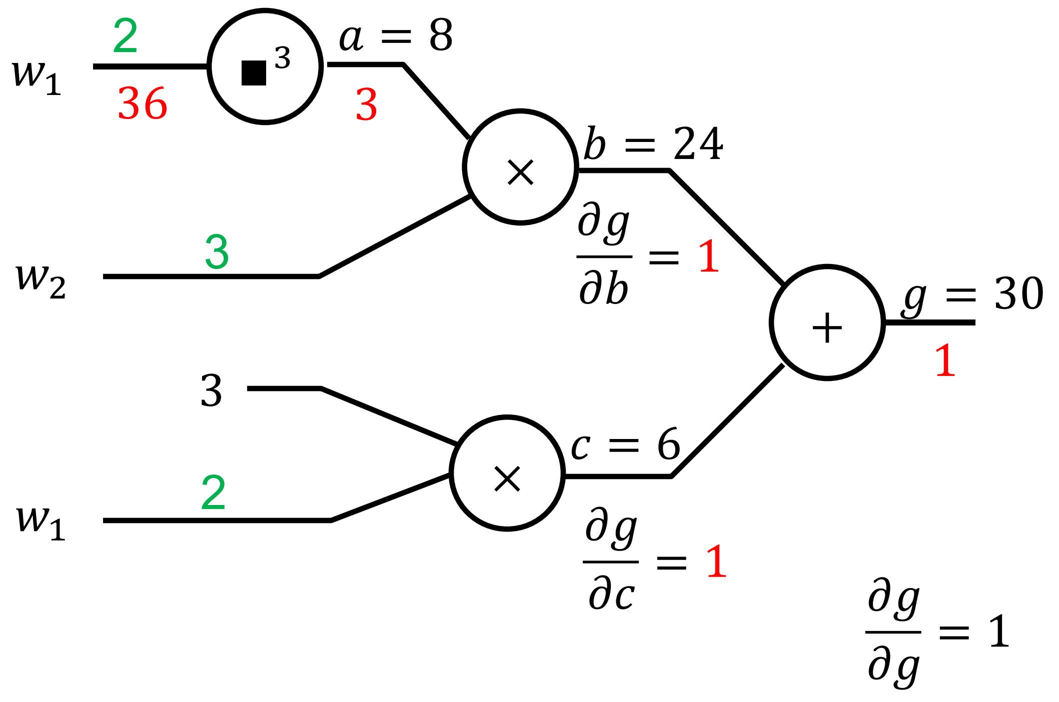


w_1
2
36
$\blacksquare^3$
$a = 8$
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Interpretation: A tiny increase in w_1 will result in an approximately $36w_1$ increase in g due to this cube function.

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

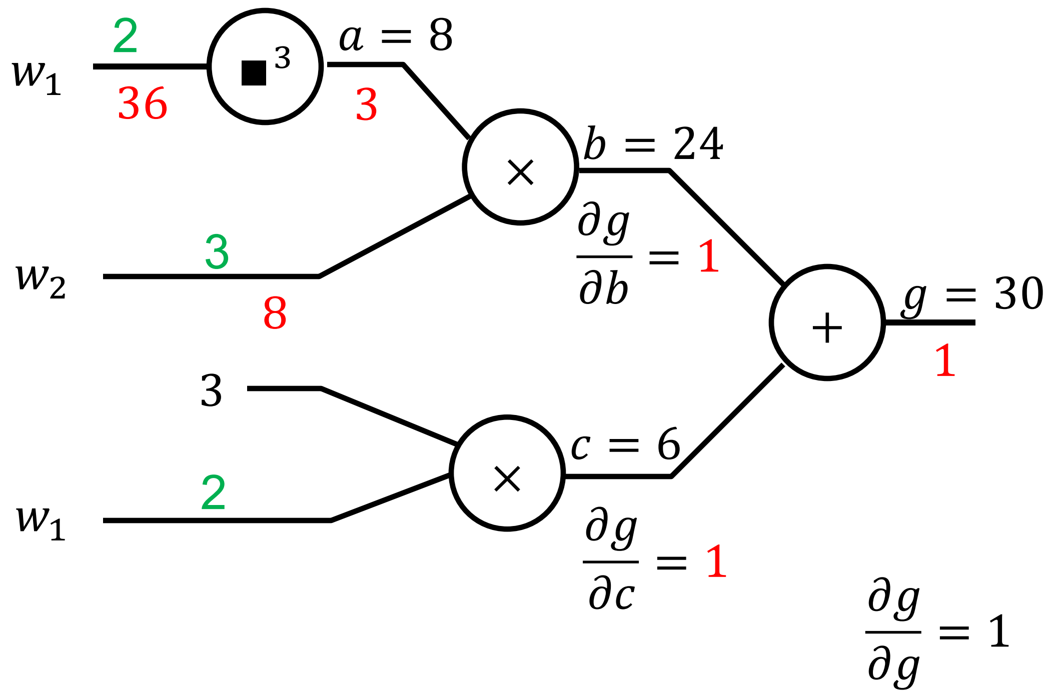
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g / \partial w_1$ and $\partial g / \partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = 1 \frac{\partial b}{\partial a} = 1 \cdot 3 = 3$
- $a = w_1^3$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial a}\frac{\partial a}{\partial w_1} = 3 \cdot 3w_1^2 = 36$
- $\frac{\partial g}{\partial w_2} = ???$ Hint: $b = a \times 3$ may be useful.



w_1
2
36
$\blacksquare^3$
$a = 8$
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
+
$g = 30$
1
3
$c = 6$
$\times$
w_1
2
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

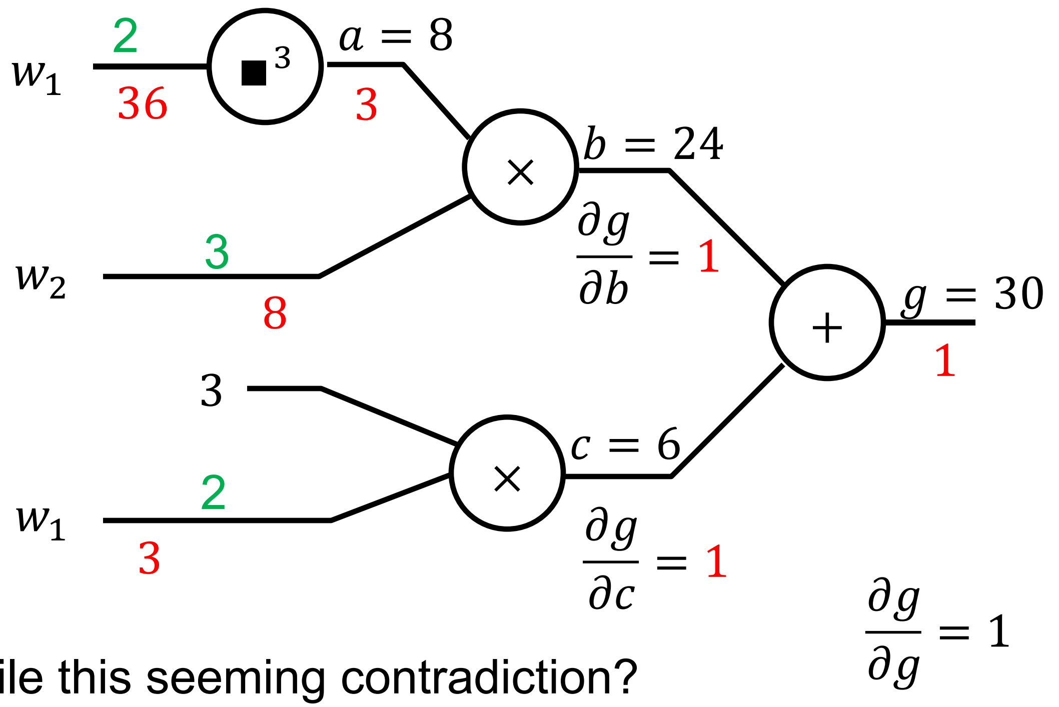
- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
  - Can use derivative chain rule to compute $\partial g / \partial w_1$ and $\partial g / \partial w_2$.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = 1 \frac{\partial b}{\partial a} = 1 \cdot 3 = 3$
  - $\frac{\partial g}{\partial w_2} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial w_2} = 1 \frac{\partial b}{\partial w_2} = 1 \cdot 8 = 8$
- $a = w_1^3$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial a}\frac{\partial a}{\partial w_1} = 3 \cdot 3w_1^2 = 36$



w_1
2
36
$\blacksquare^3$
$a = 8$
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
8
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = 1\frac{\partial b}{\partial a} = 1 \cdot 3 = 3$
  - $\frac{\partial g}{\partial w_2} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial w_2} = 1\frac{\partial b}{\partial w_2} = 1 \cdot 8 = 8$
- $a = w_1^3$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial a}\frac{\partial a}{\partial w_1} = 3 \cdot 3w_1^2 = 36$
- $c = 3w_1$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial c}\frac{\partial c}{\partial w_1} = 1 \cdot 3 = 3$

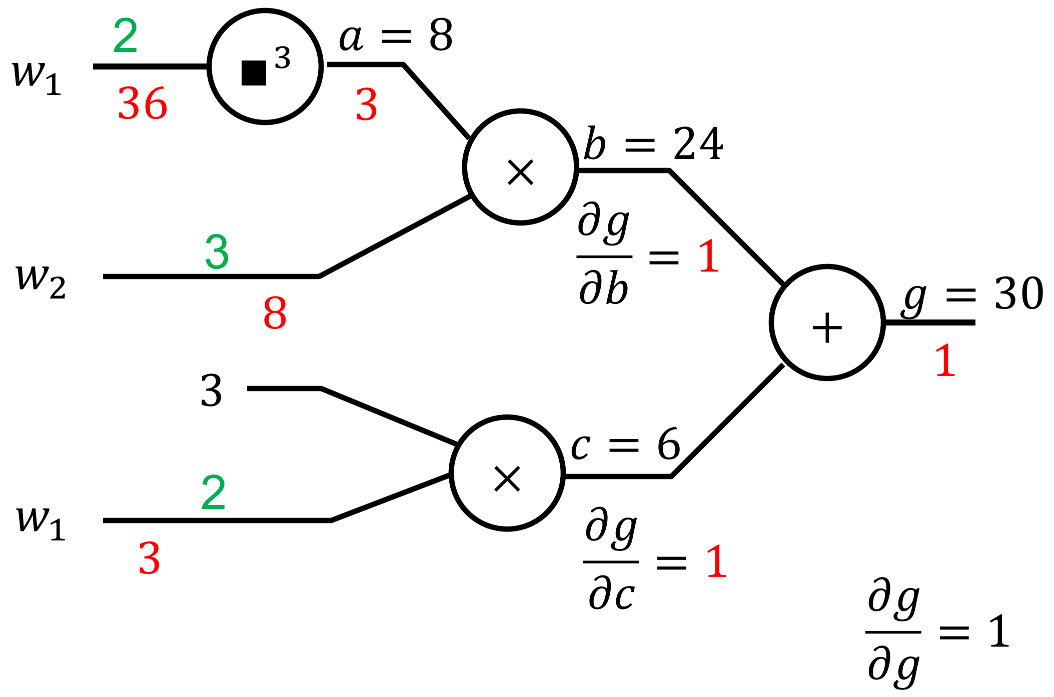


w_1
2
36
$\blacksquare^3$
$a = 8$
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
8
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
3
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

How do we reconcile this seeming contradiction? Top partial derivative means cube function contributes $36w_1$ and bottom p.d. means product contributes $3w_1$ so add them.

Back Propagation: $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$

- Suppose we have $g(\boldsymbol{w}) = w_1^3 w_2 + 3w_1$ and want the gradient at $\boldsymbol{w} = [2, 3]$
- Think of the function as a composition of many functions, use chain rule.
- $g = b + c$
  - $\frac{\partial g}{\partial b} = 1, \frac{\partial g}{\partial c} = 1$
- $b = a \times w_2$
  - $\frac{\partial g}{\partial a} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial a} = 1\,\frac{\partial b}{\partial a} = 1 \cdot 3 = 3$
  - $\frac{\partial g}{\partial w_2} = \frac{\partial g}{\partial b}\frac{\partial b}{\partial w_2} = 1\,\frac{\partial b}{\partial w_2} = 1 \cdot 8 = 8$
- $a = w_1^3$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial a}\frac{\partial a}{\partial w_1} = 3 \cdot 3w_1^2 = 36$
- $c = 3w_1$
  - $\frac{\partial g}{\partial w_1} = \frac{\partial g}{\partial c}\frac{\partial c}{\partial w_1} = 1 \cdot 3 = 3$



w_1
2
36
$\blacksquare^3$
$a = 8$
3
$\times$
$b = 24$
$\frac{\partial g}{\partial b} = 1$
w_2
3
8
$+$
$g = 30$
1
3
$\times$
$c = 6$
w_1
2
3
$\frac{\partial g}{\partial c} = 1$
$\frac{\partial g}{\partial g} = 1$

$$\nabla g = \left[\frac{\partial g}{\partial w_1}, \frac{\partial g}{\partial w_2}\right] = [39, 8]$$

Gradient Ascent

- Punchline: If we can somehow compute our gradient, we can use gradient ascent.
- How do we compute the gradient?
  - Purely analytically.
    - Gives exact symbolic answer. Infeasible for functions of lots of parameters or input values.
  - Finite difference approximation.
    - Gives approximation, very easy to implement.
    - Runtime for ll: $O(NM)$, where N is the number of parameters, and M is number of data points.
  - Back propagation.
    - Gives exact answer, difficult to implement.
    - Runtime for ll: $O(NM)$

$$ll(w) = \sum_{i=1}^{m} \log p(y = y^{(i)} | f(x^{(i)}); w)$$

Summary of Key Ideas

- Optimize probability of label given input $$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$
- Continuous optimization
  - Gradient ascent:
    - Compute steepest uphill direction = gradient (= just vector of partial derivatives)
    - Take step in the gradient direction
    - Repeat (until held-out data accuracy starts to drop = “early stopping”)
- Deep neural nets
  - Last layer = still logistic regression
  - 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 (how? = outside of scope of 343)

Exercises: Deep Learning

Consider the following computation graph for a simple neural network for binary classification. Here x is a single real-valued input feature with an associated class $y^\star$ (0 or 1). There are two weight parameters w_1 and w_2, and non-linearity functions g_1 and g_2 (to be defined later, below). The network will output a value a_2 between 0 and 1, representing the probability of being in class 1. We will be using a loss function $Loss$ (to be defined later, below), to compare the prediction a_2 with the true class $y^\star$.

$$\begin{tikzcd}
w_1 \arrow[dr] & & w_2 \arrow[dr] & & & \\
& * \arrow[r, "z_1 \rightarrow"] & g_1 \arrow[r, "a_1 \rightarrow"] & * \arrow[r, "z_2 \rightarrow"] & g_2 \arrow[r, "a_2 \rightarrow"] & Loss \\
x \arrow[ur] & & & & y^* \arrow[ur] &
\end{tikzcd}$$

1. Perform the forward pass on this network, writing the output values for each node z_1, a_1, z_2 and a_2 in terms of the node's input values:

2. Compute the loss $Loss(a_2, y^\star)$ in terms of the input x, weights w_i, and activation functions g_i:

3. Now we will work through parts of the backward pass, incrementally. Use the chain rule to derive $\frac{\partial Loss}{\partial w_2}$. Write your expression as a product of partial derivatives at each node: i.e. the partial derivative of the node's output with respect to its inputs. (Hint: the series of expressions you wrote in part 1 will be helpful; you may use any of those variables.)

4. Suppose the loss function is quadratic, $Loss(a_2, y^\star) = \frac{1}{2}(a_2 - y^\star)^2$, and g_1 and g_2 are both sigmoid functions $g(z) = \frac{1}{1+e^{-z}}$ (note: it's typically better to use a different type of loss, *cross-entropy*, for classification problems, but we'll use this to make the math easier).
   Using the chain rule from Part 3, and the fact that $\frac{\partial g(z)}{\partial z} = g(z)(1 - g(z))$ for the sigmoid function, write $\frac{\partial Loss}{\partial w_2}$ in terms of the values from the forward pass, $y^\star$, a_1, and a_2:

$$z_1 = x * w_1$$
$$a_1 = g_1(z_1)$$
$$z_2 = a_1 * w_2$$
$$a_2 = g_2(z_2)$$

$$Loss(a_2, y^\star) = Loss(g_2(w_2 * g_1(w_1 * x)), y^\star)$$

$$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

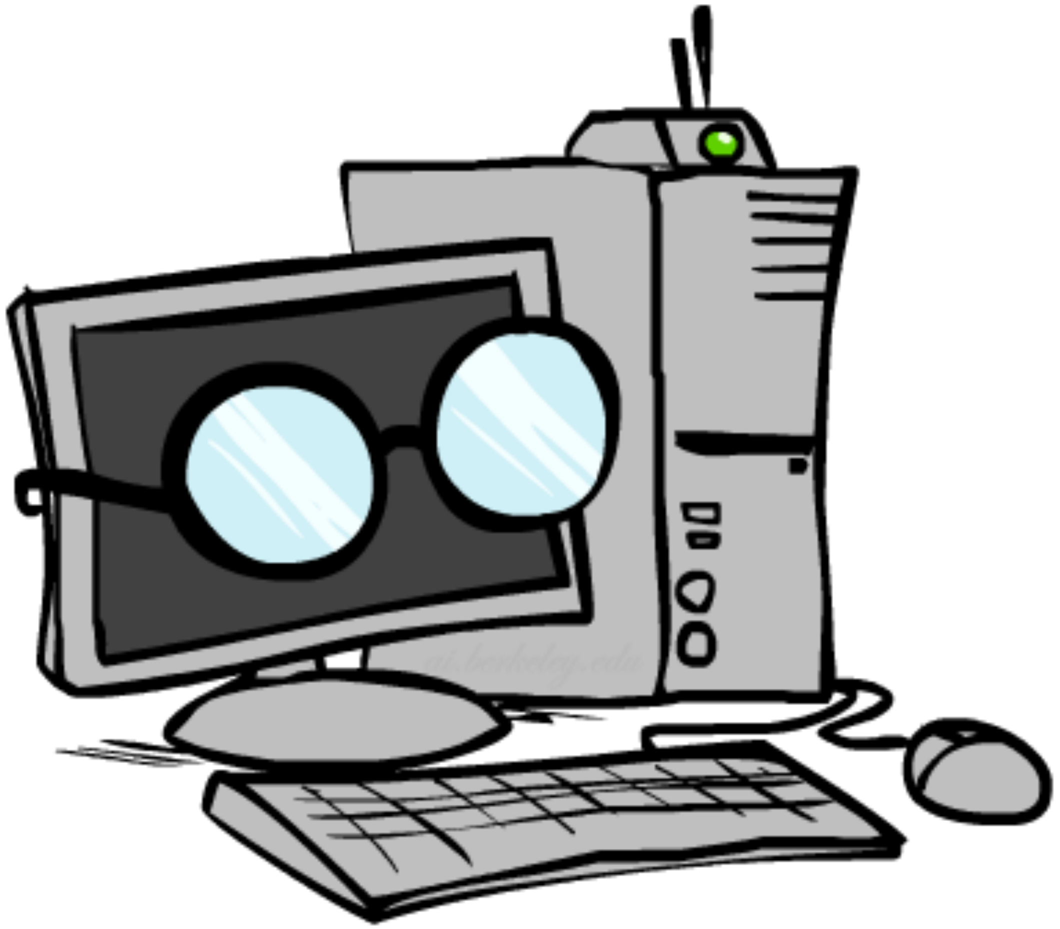
First we'll compute the partial derivatives at each node:

$$\frac{\partial Loss}{\partial a_2} = (a_2 - y^\star)$$
$$\frac{\partial a_2}{\partial z_2} = \frac{\partial g_2(z_2)}{\partial z_2} = g_2(z_2)(1 - g_2(z_2)) = a_2(1 - a_2)$$
$$\frac{\partial z_2}{\partial w_2} = a_1$$

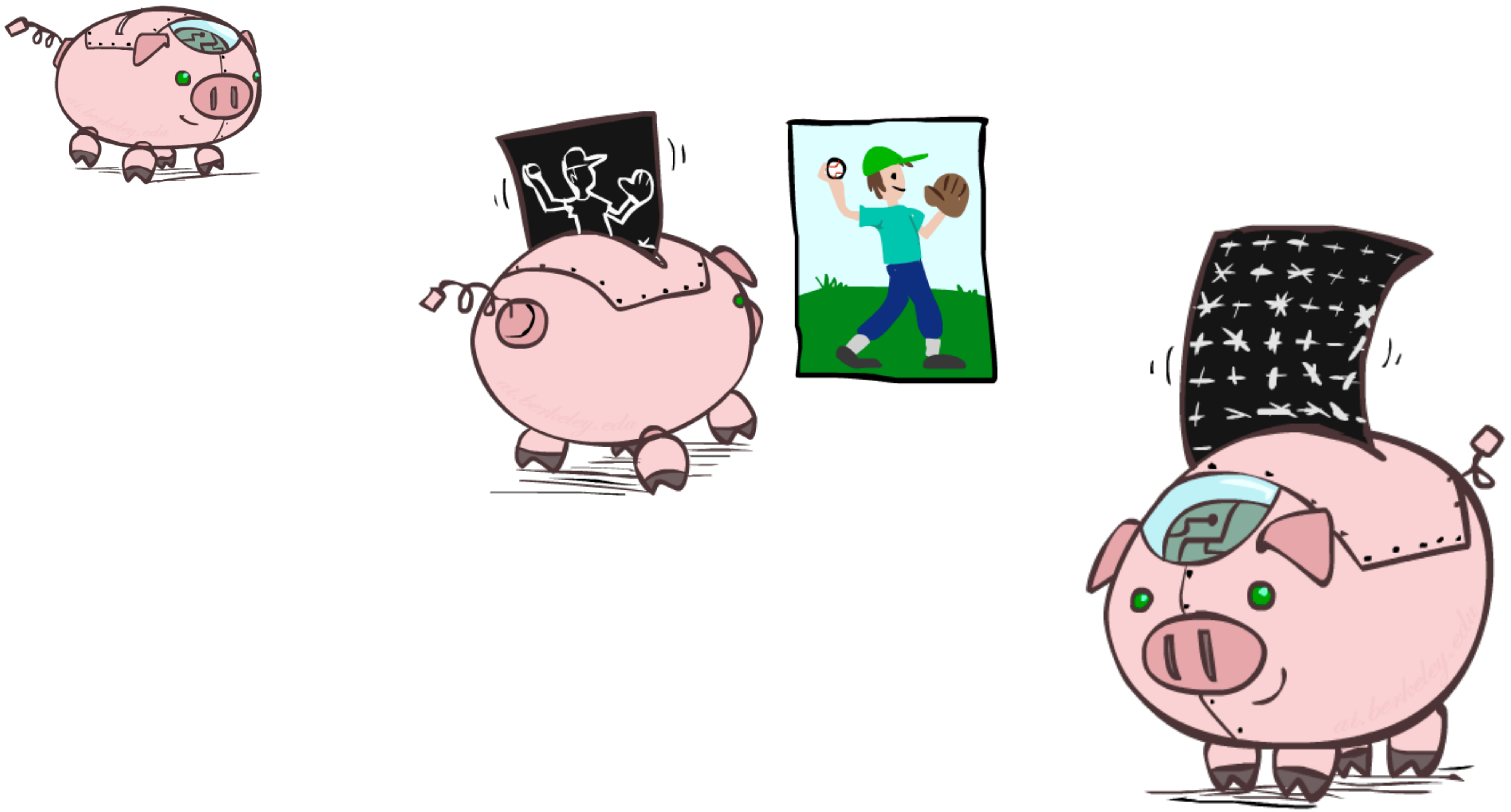
Now we can plug into the chain rule from part 3:

$$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$
$$= (a_2 - y^\star) * a_2(1 - a_2) * a_1$$

Computer Vision

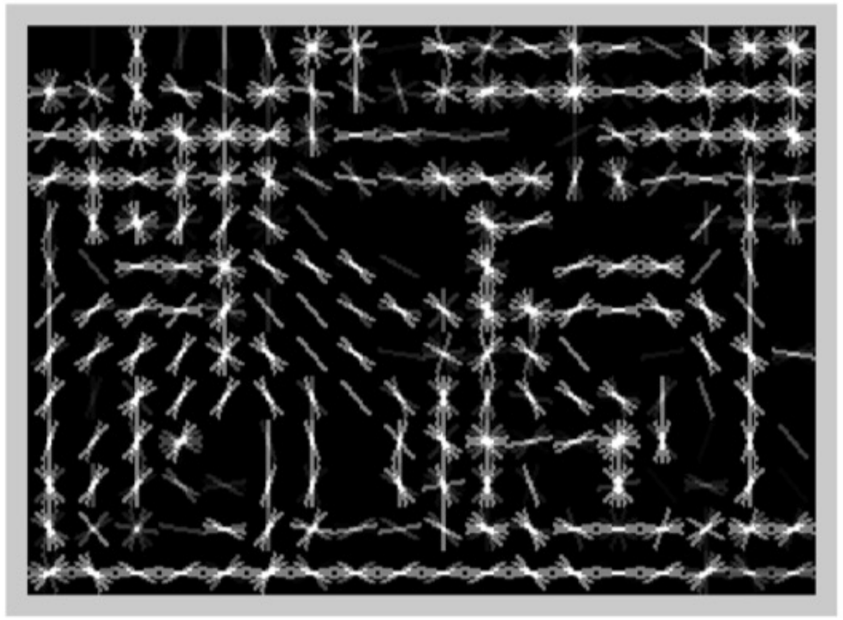


Manual Feature Design

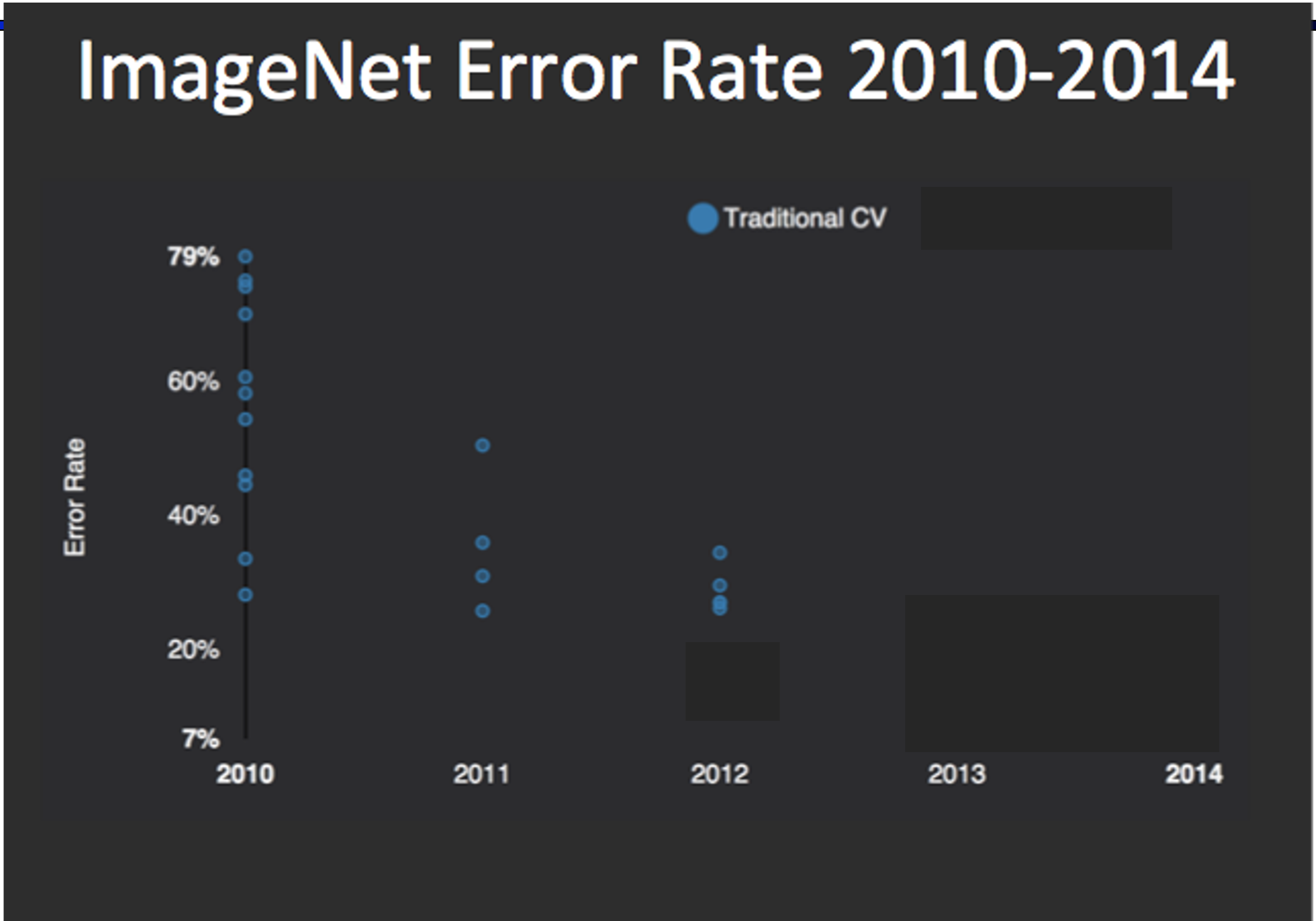


Features and Generalization

Image

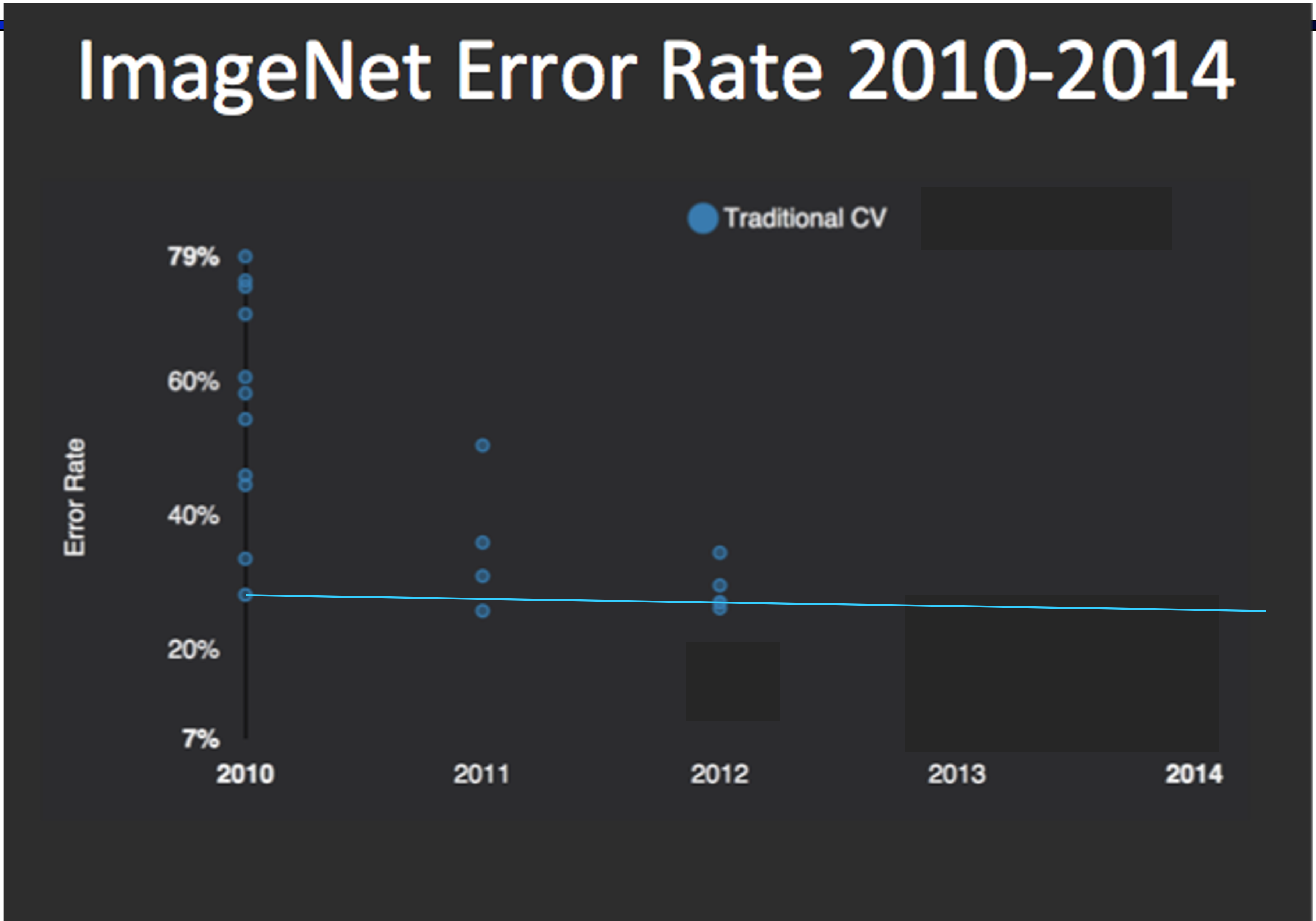

HoG

Performance



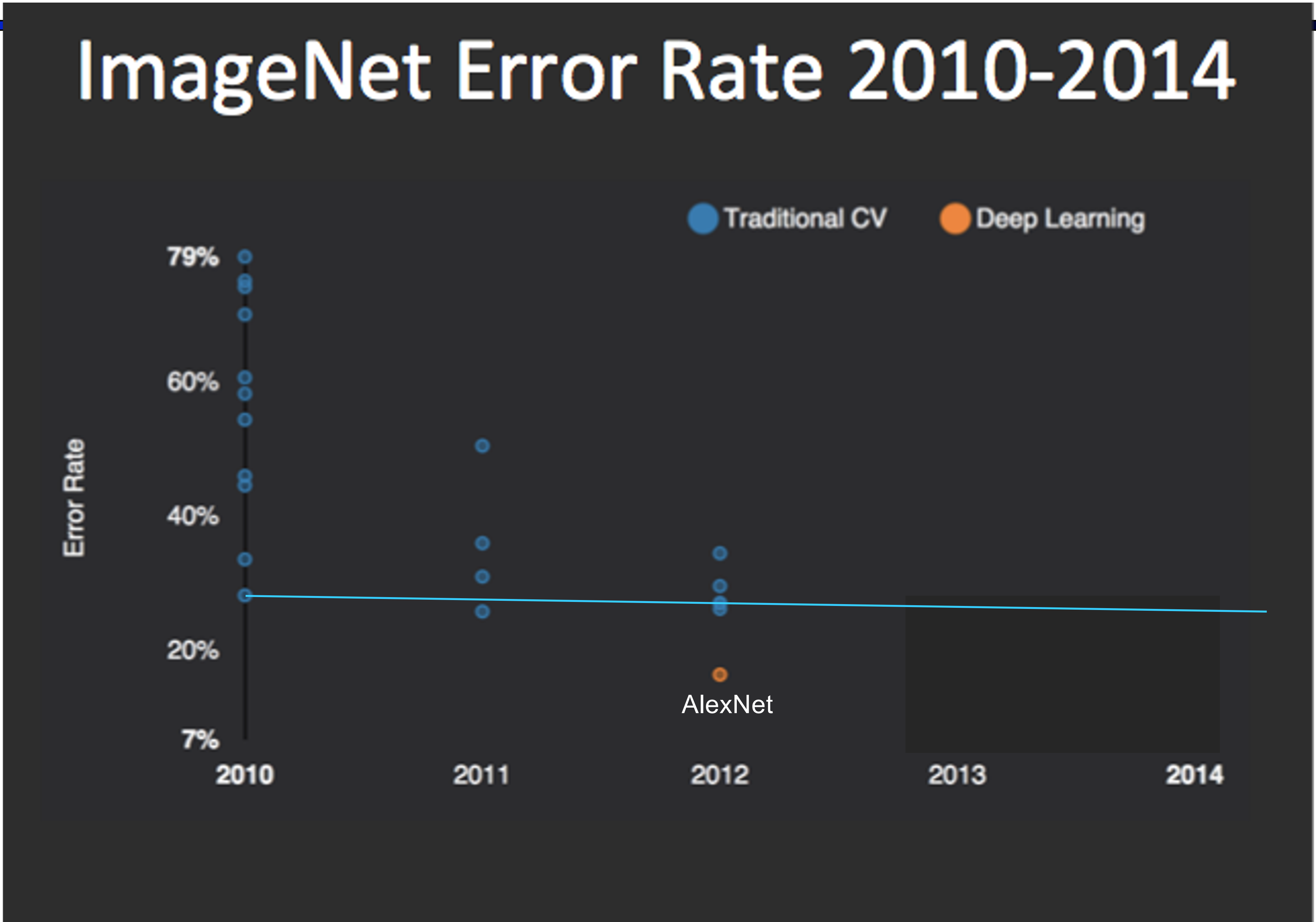
graph credit Matt Zeiler, Clarifai

Performance



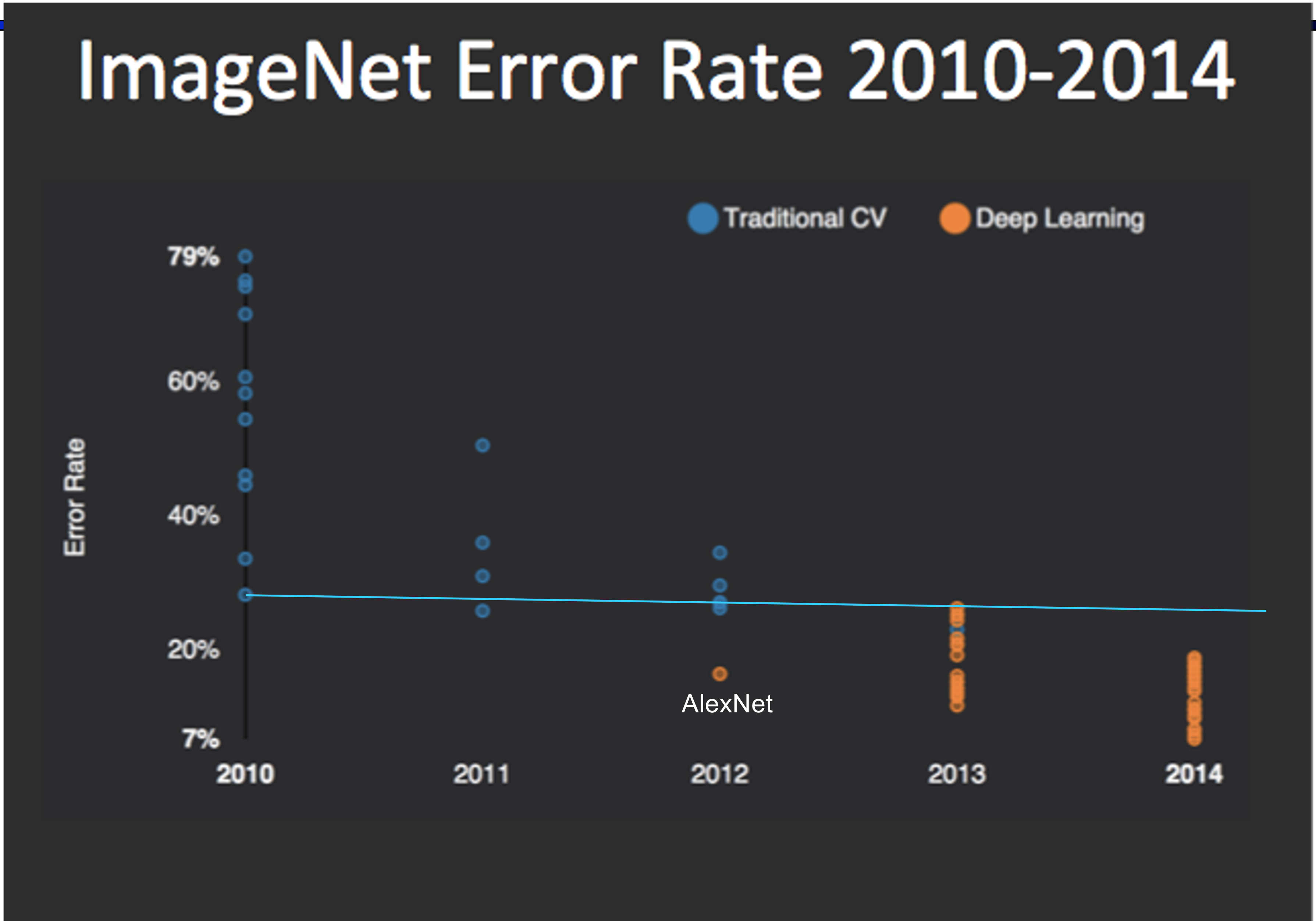
graph credit Matt Zeiler, Clarifai

Performance



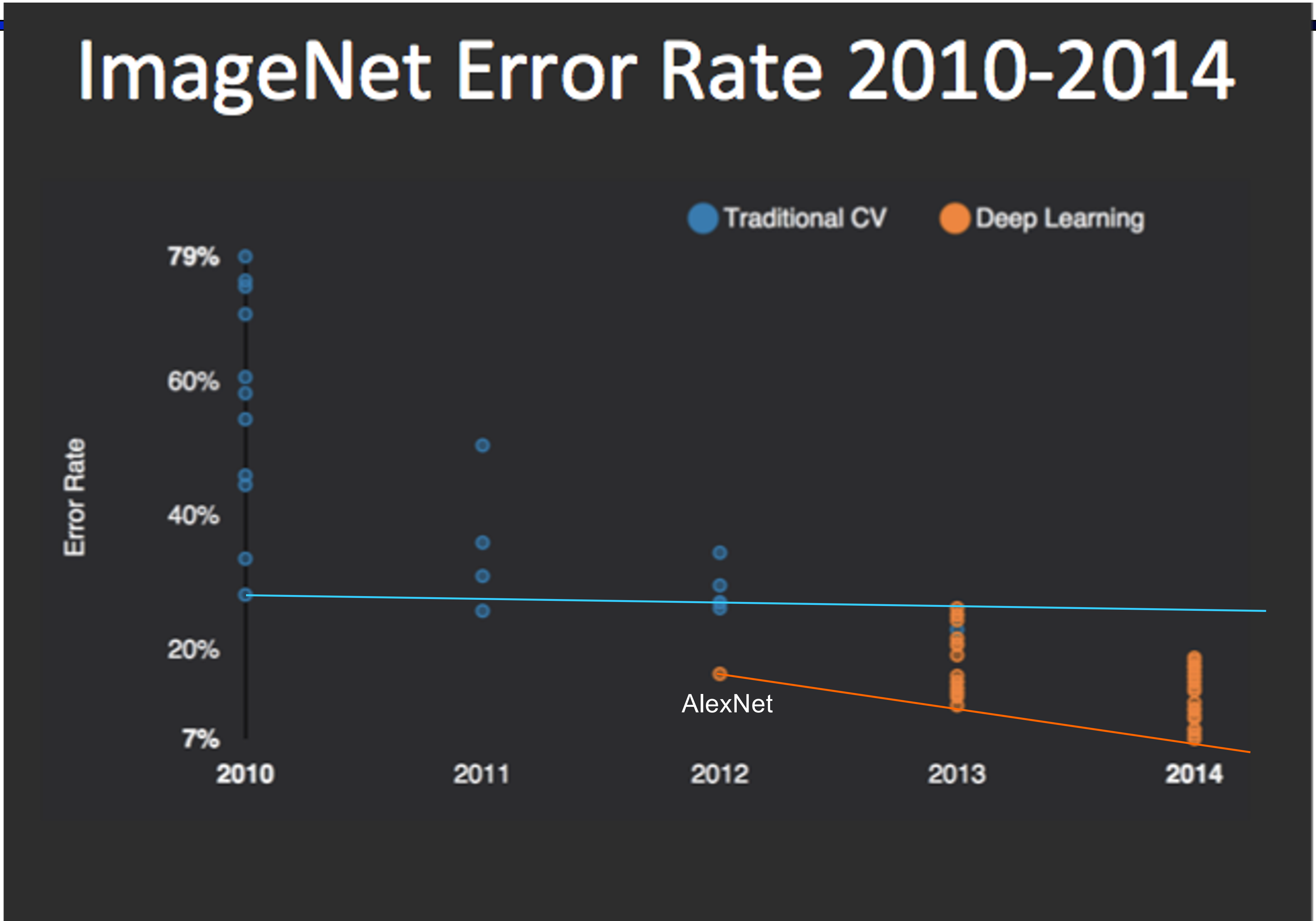
graph credit Matt Zeiler, Clarifai

Performance



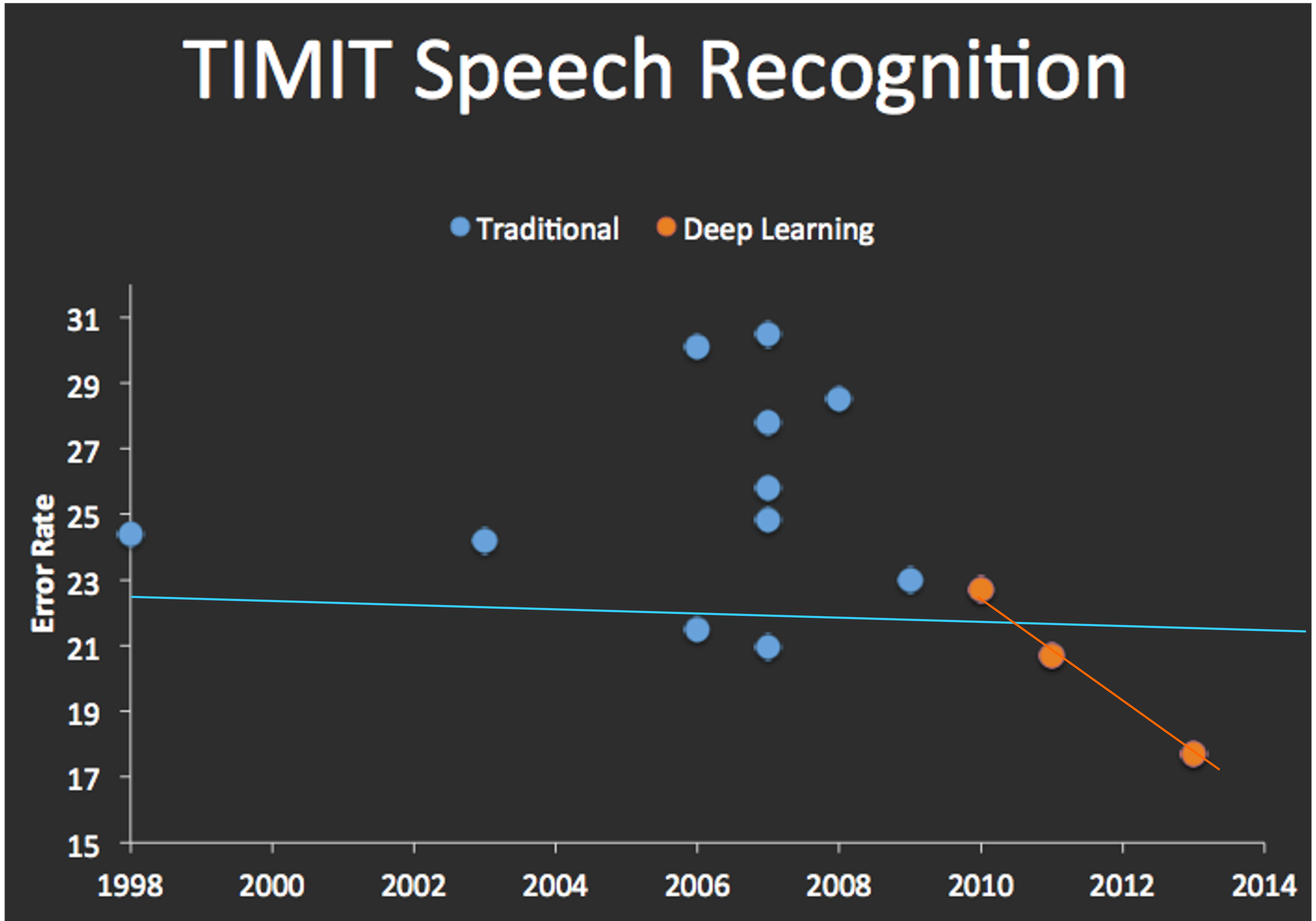
graph credit Matt Zeiler, Clarifai

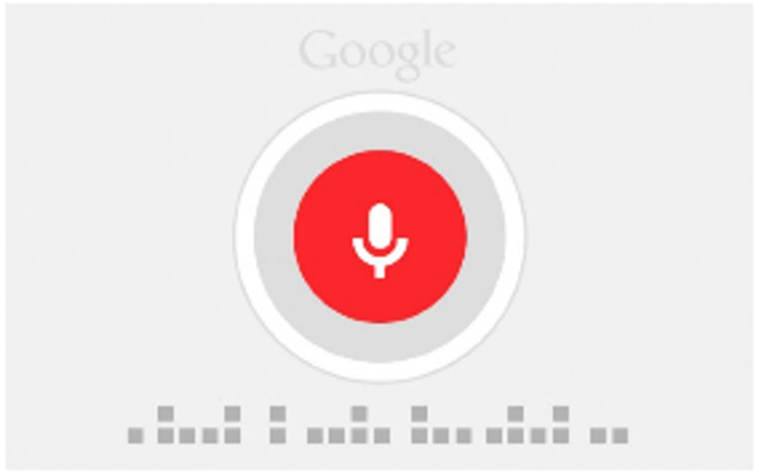
Performance



graph credit Matt Zeiler, Clarifai

Speech Recognition

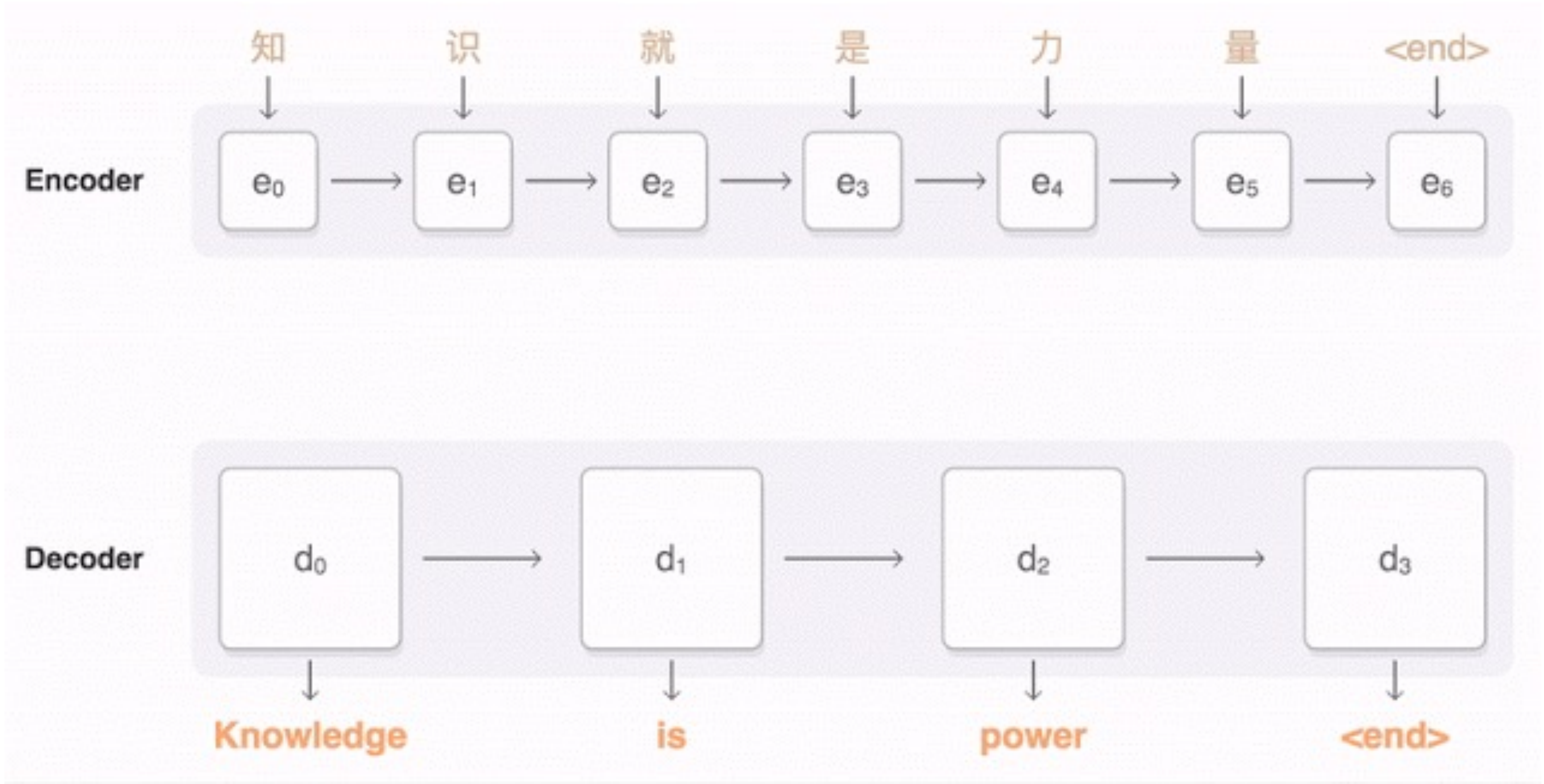




graph credit Matt Zeiler, Clarifai

Machine Translation

Google Neural Machine Translation (in production)


知
识
就
是
力
量
<end>
Encoder
e_0
e_1
e_2
e_3
e_4
e_5
e_6
Decoder
d_0
d_1
d_2
d_3
Knowledge
is
power
<end>