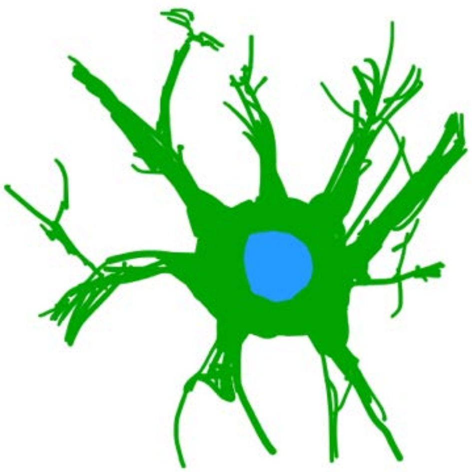
Intro to Neural Networks

Cynthia Rudin

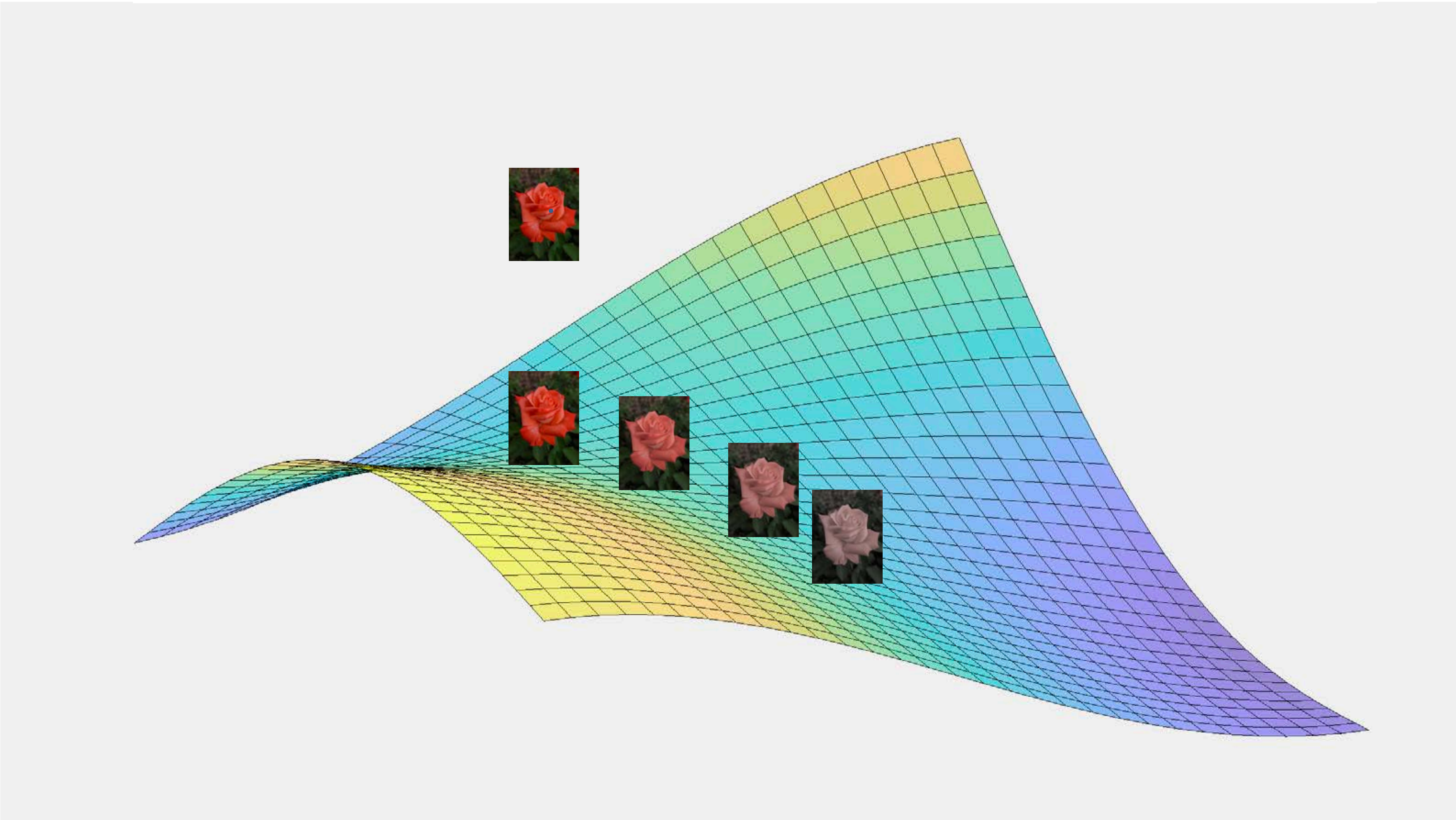
Duke Machine Learning

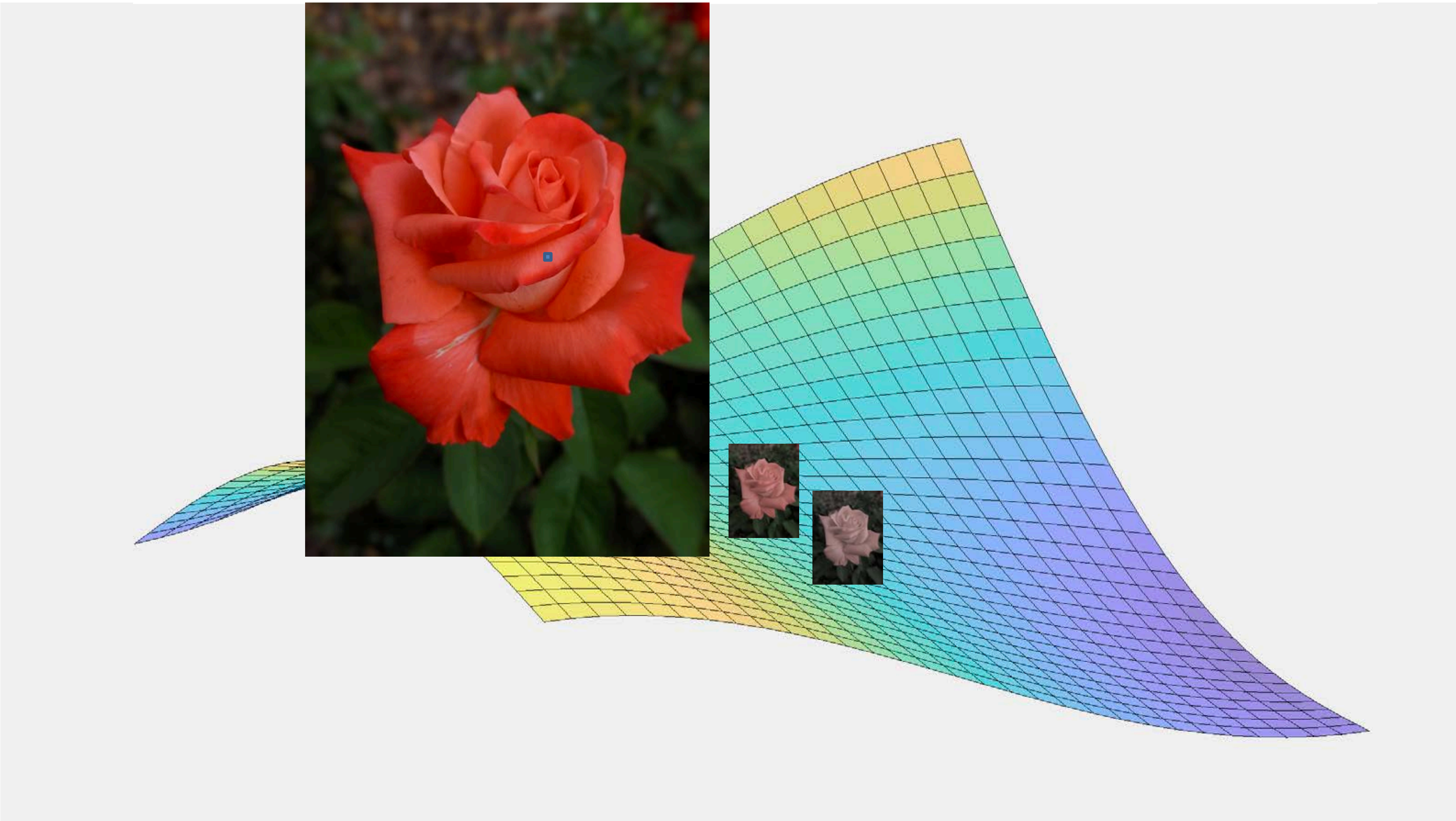
Neurons

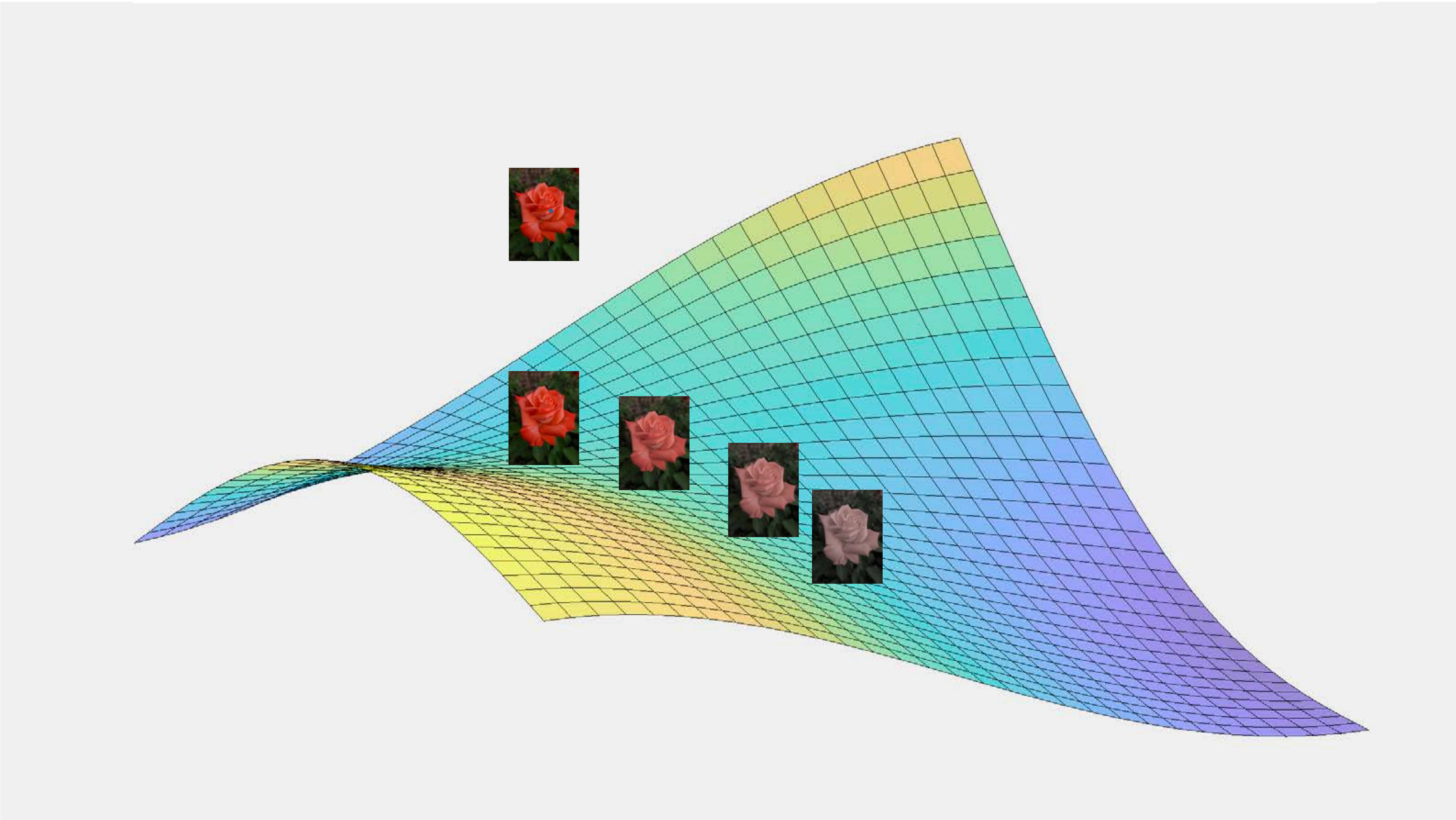
- 10^{11} neurons in a brain, 10^{14} synapses (connections).
- Signals are electrical potential spikes that travel through the network.



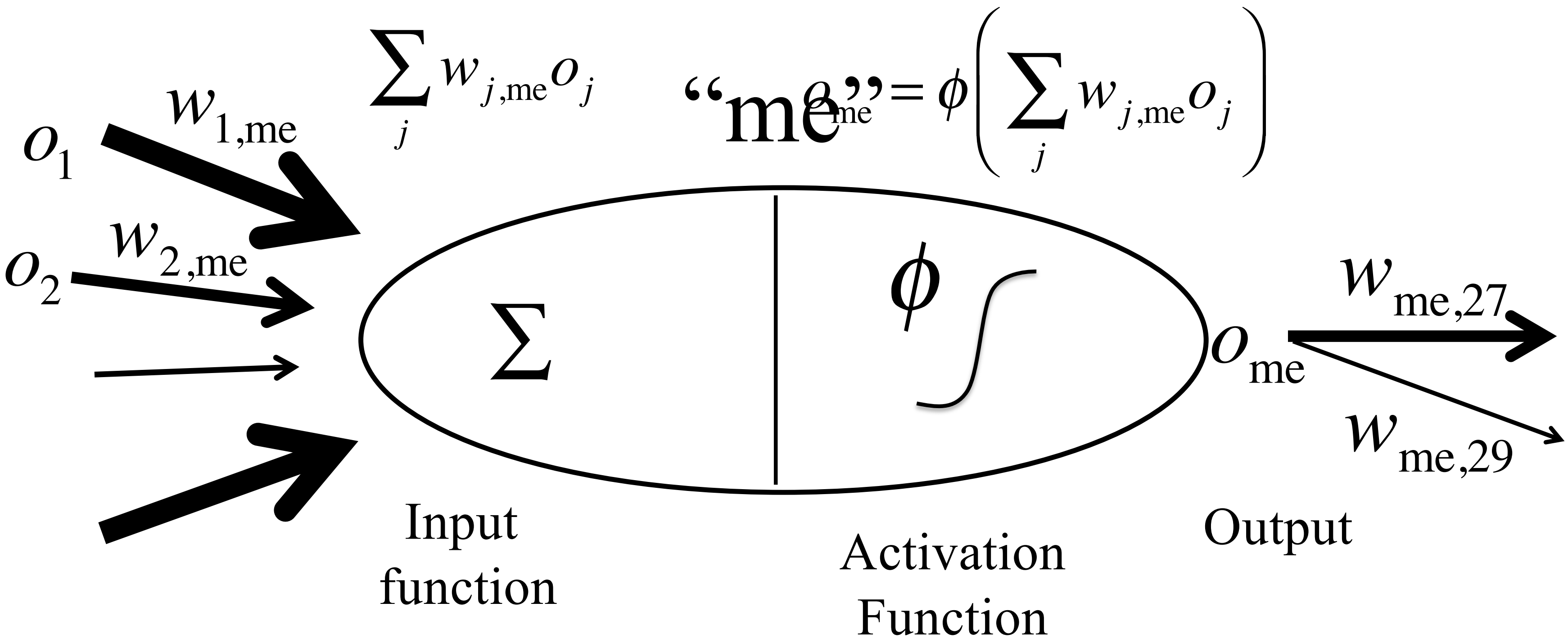
(Credit: Adapted from Russell and Norvig)



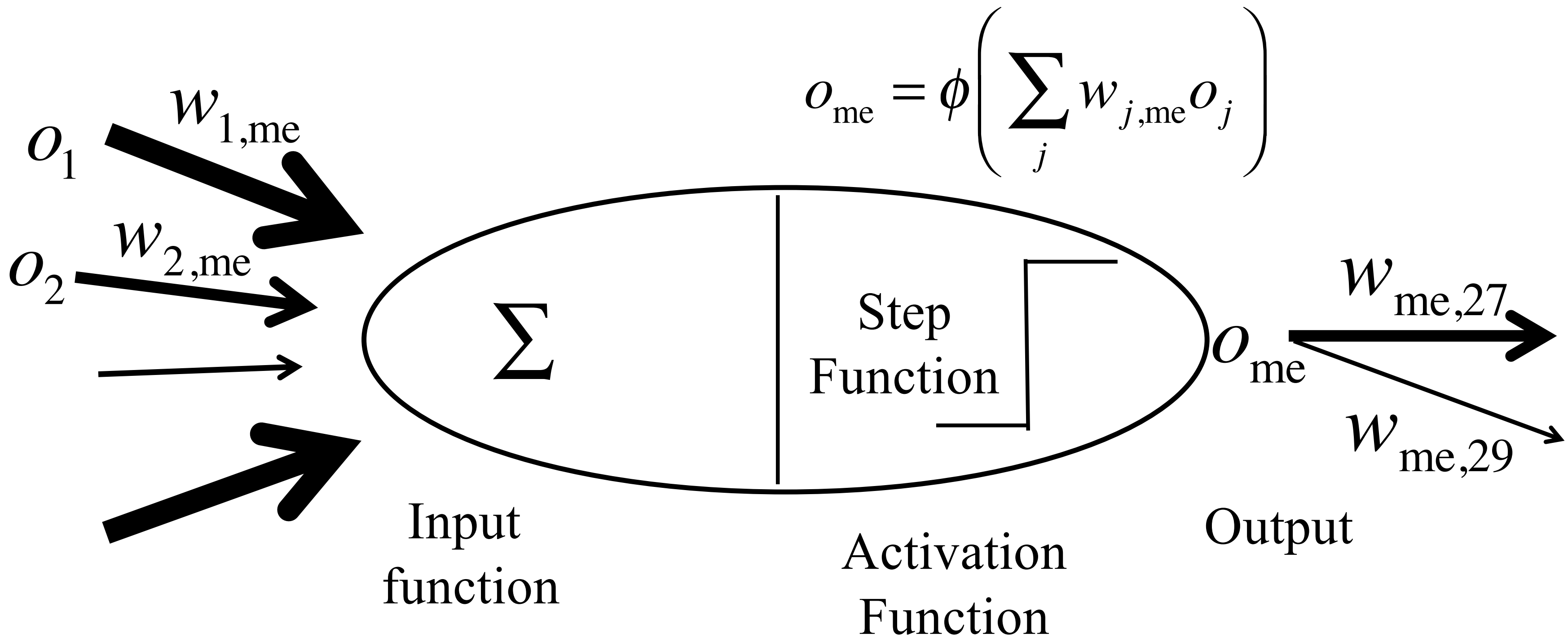




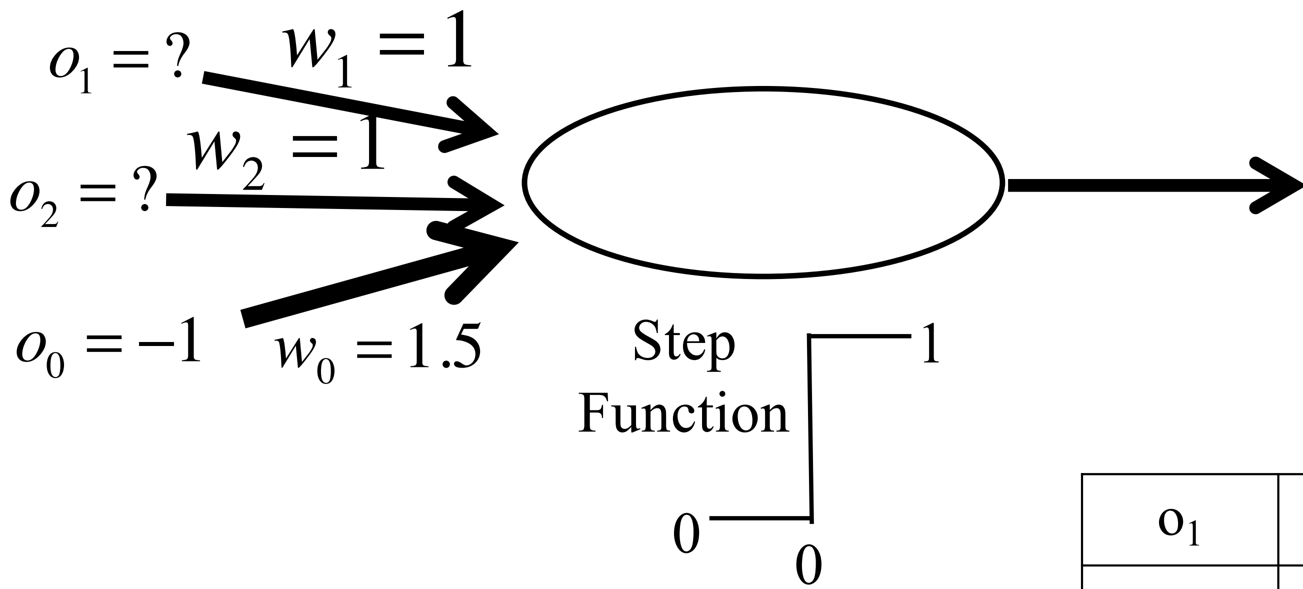
McCulloch-Pitts "Neuron"



McCulloch-Pitts "Neuron"

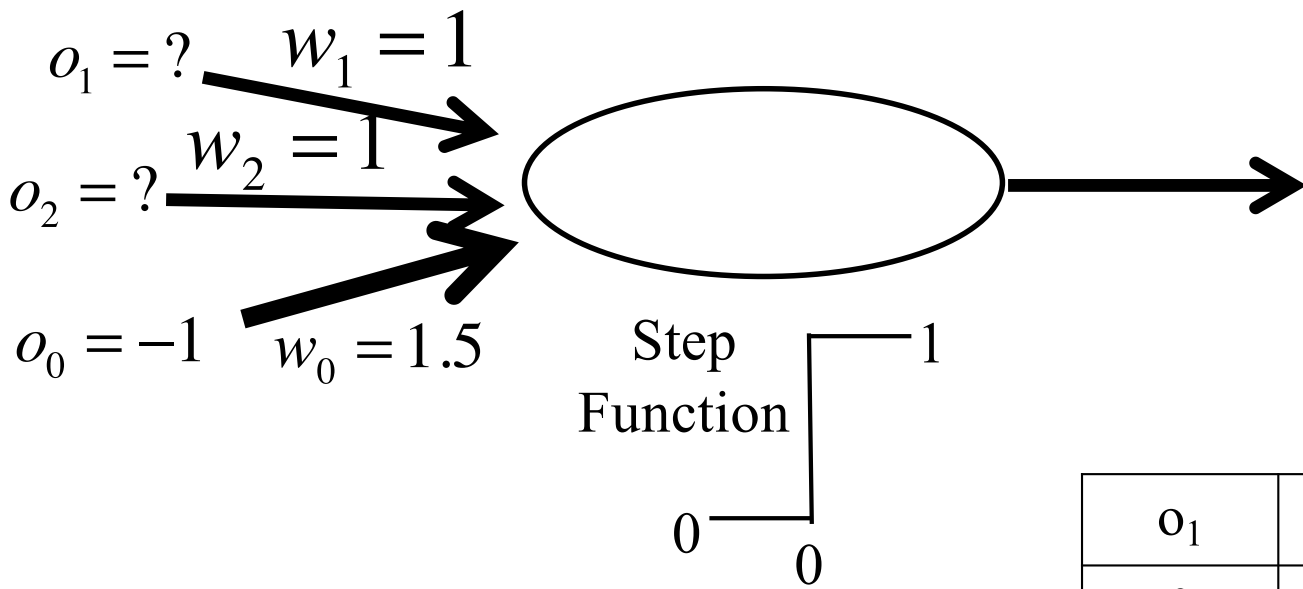


McCulloch-Pitts “Neuron”



o_1	o_2	output

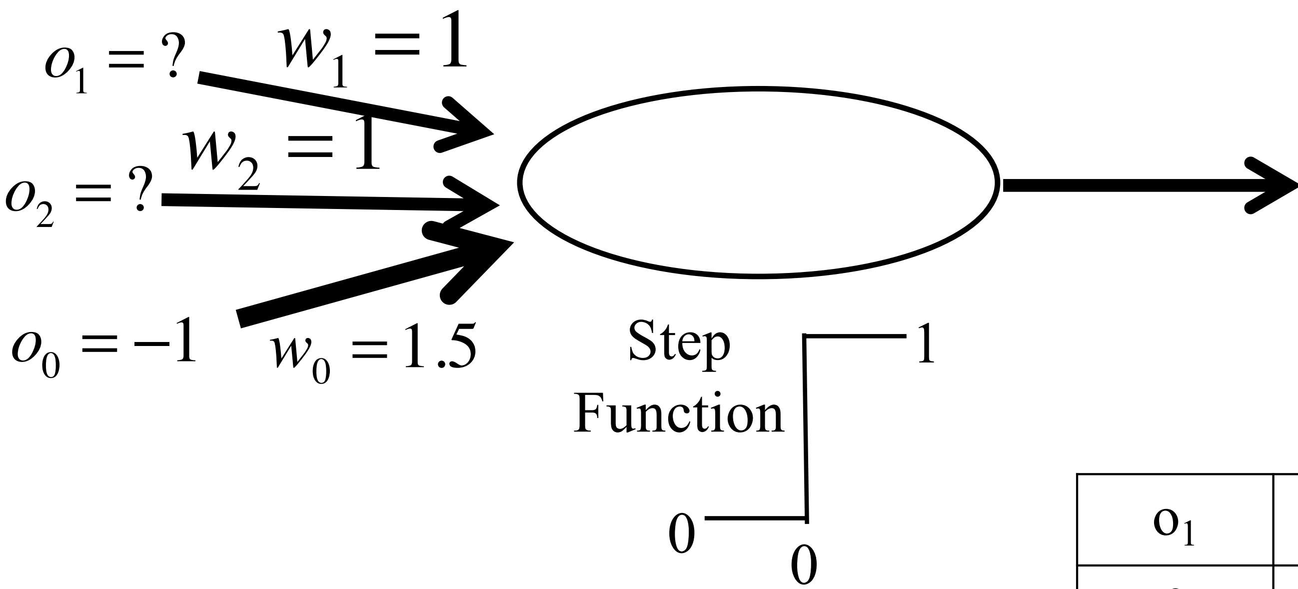
McCulloch-Pitts "Neuron"



o_1	o_2	output
0	0	

0+0-1.5 = -1.5

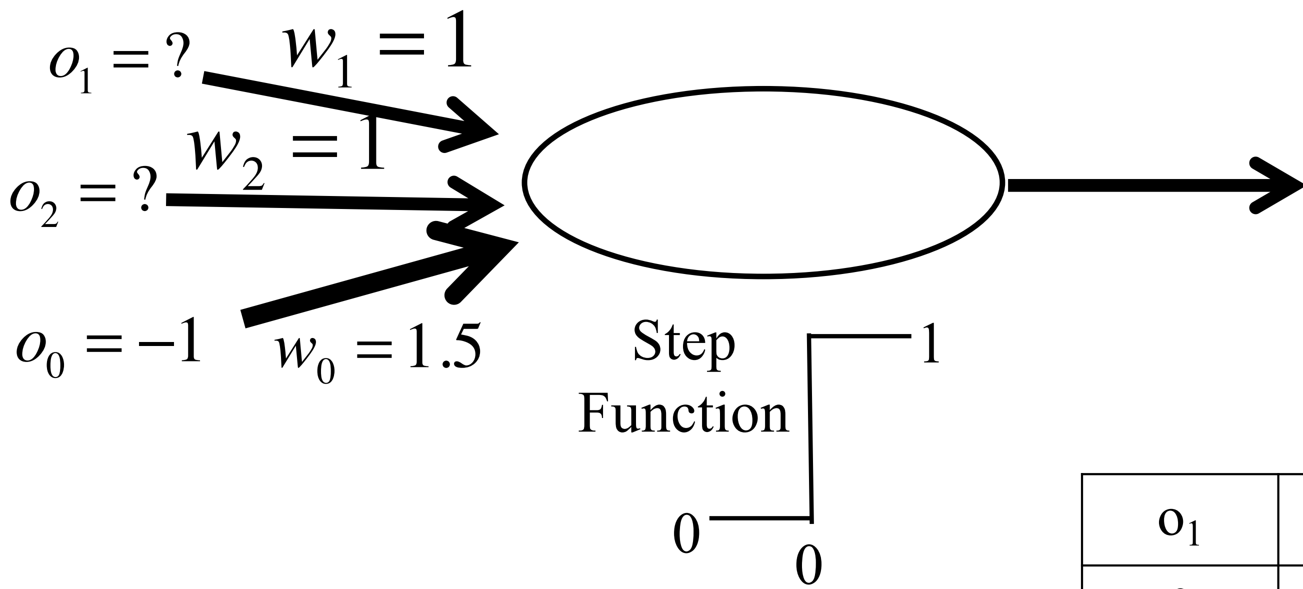
McCulloch-Pitts “Neuron”



o_1	o_2	output
0	0	0

0+0-1.5 = -1.5

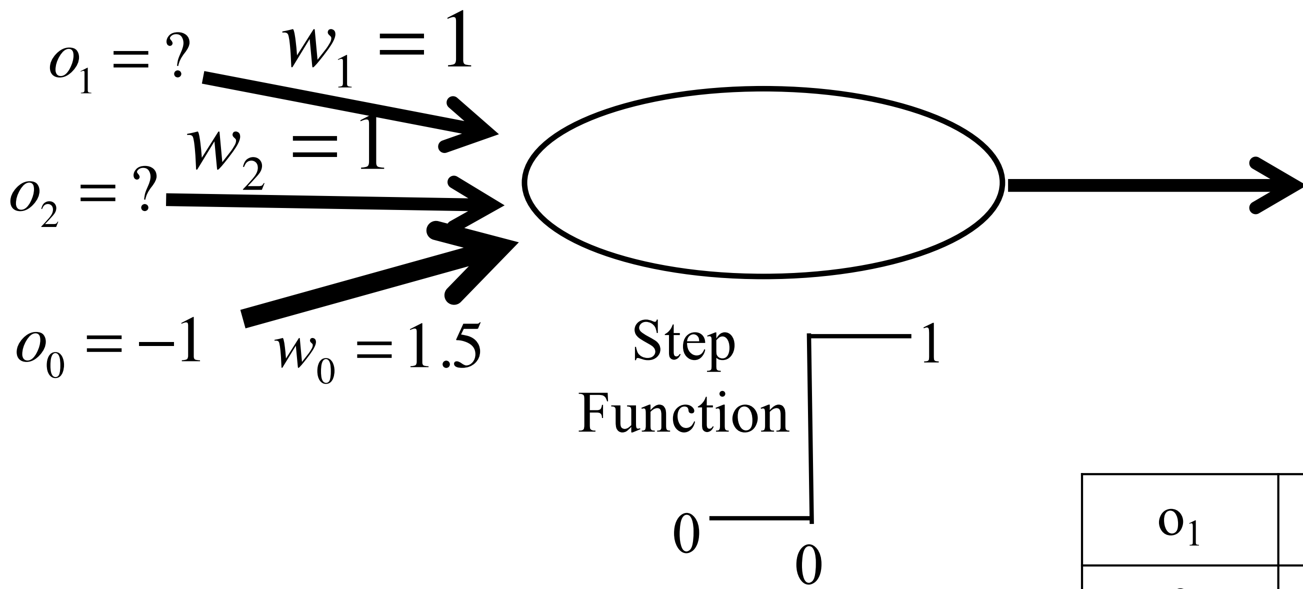
McCulloch-Pitts "Neuron"



o_1	o_2	output
0	0	0
1	0	0

1+0-1.5 = -0.5

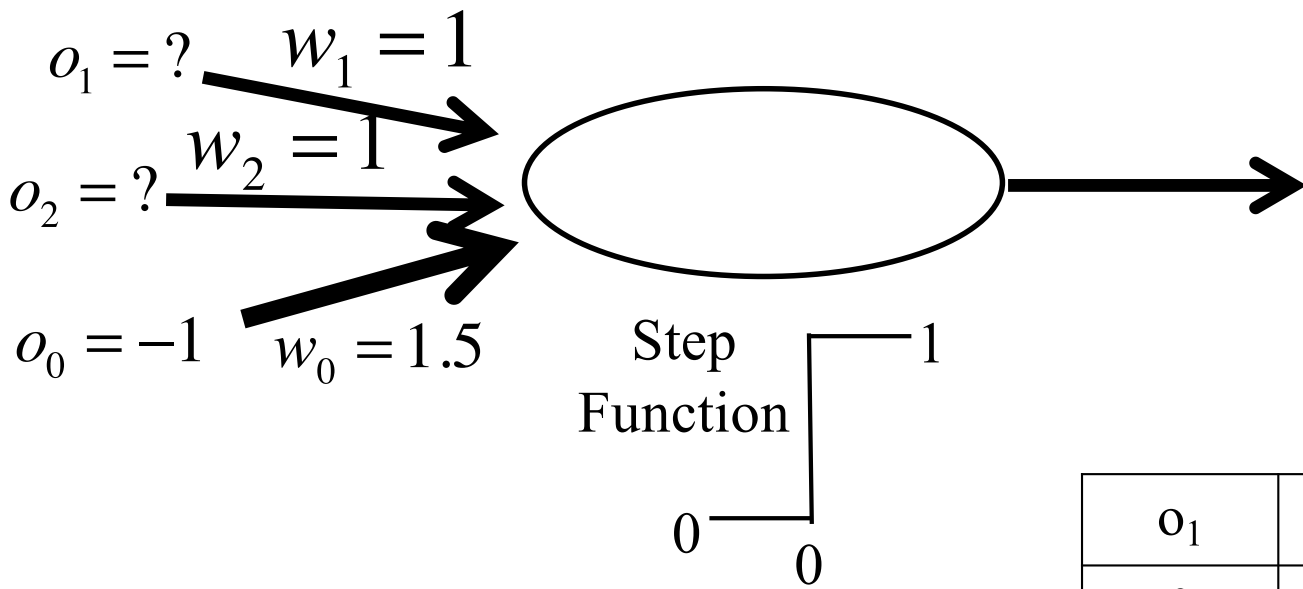
McCulloch-Pitts "Neuron"



o_1	o_2	output
0	0	0
1	0	0
0	1	0

0+1-1.5 = -0.5

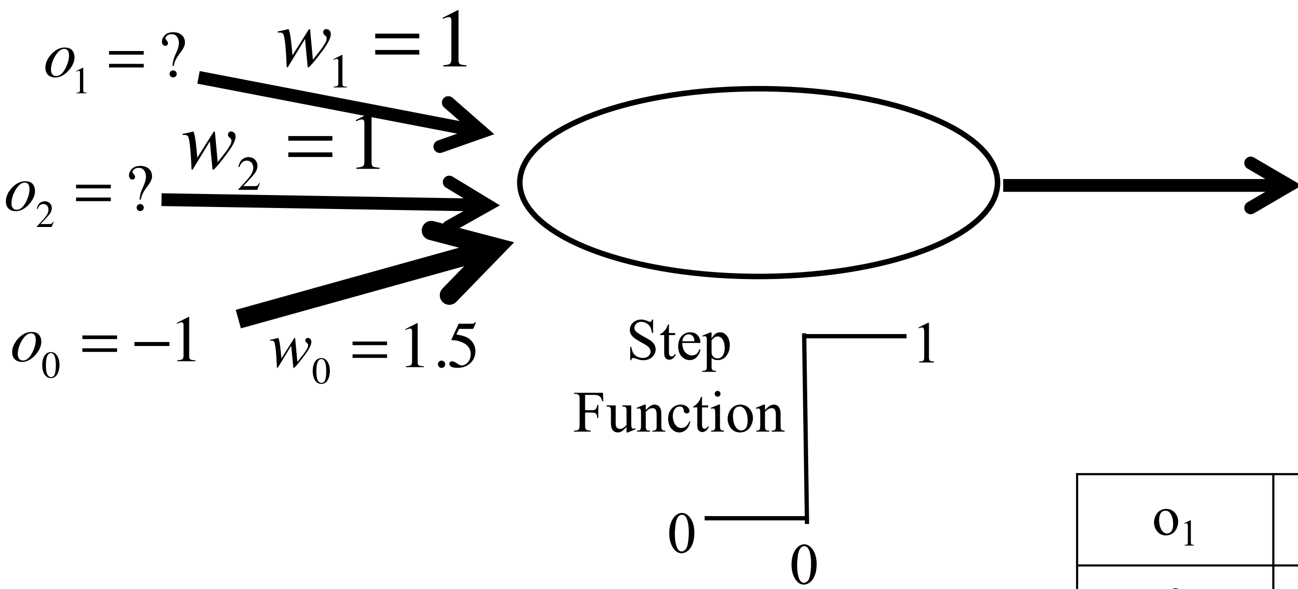
McCulloch-Pitts "Neuron"



o_1	o_2	output
0	0	0
1	0	0
0	1	0
1	1	

1+1-1.5 = 0.5

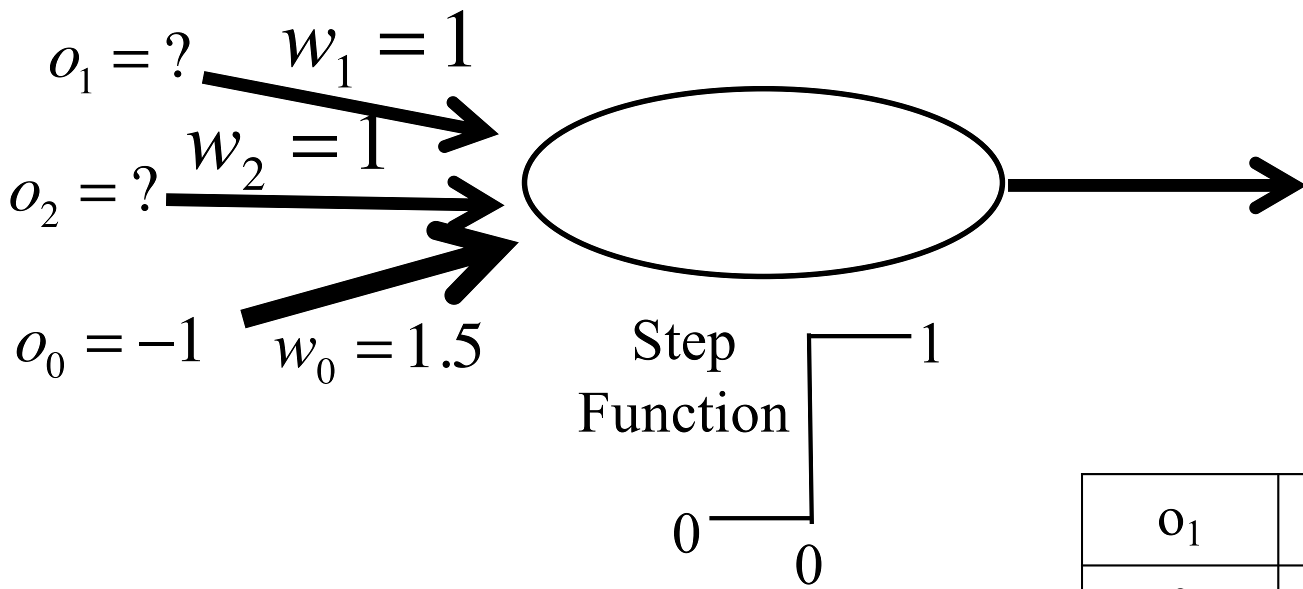
McCulloch-Pitts "Neuron"



1+1-1.5 = 0.5

o_1	o_2	output
0	0	0
1	0	0
0	1	0
1	1	1

McCulloch-Pitts "Neuron"



This neuron computes
the function "and."

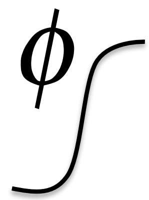
There are "or" and "not" neurons too.

o_1	o_2	output
0	0	0
1	0	0
0	1	0
1	1	1

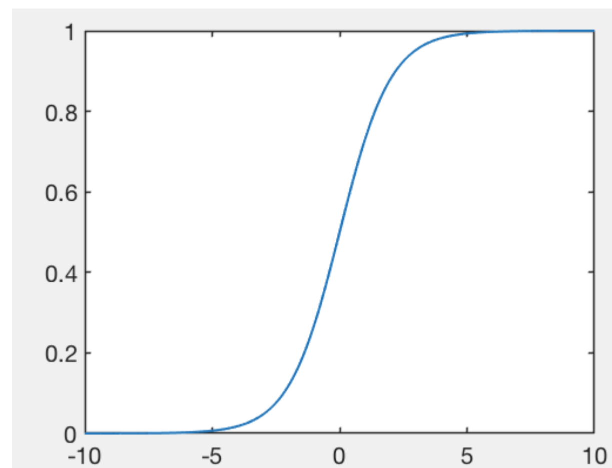
McCulloch-Pitts “Neuron”

$$\phi\left(\sum_j w_{j,\mathrm{me}} a_j\right) = 1/(1+e^{-x})$$

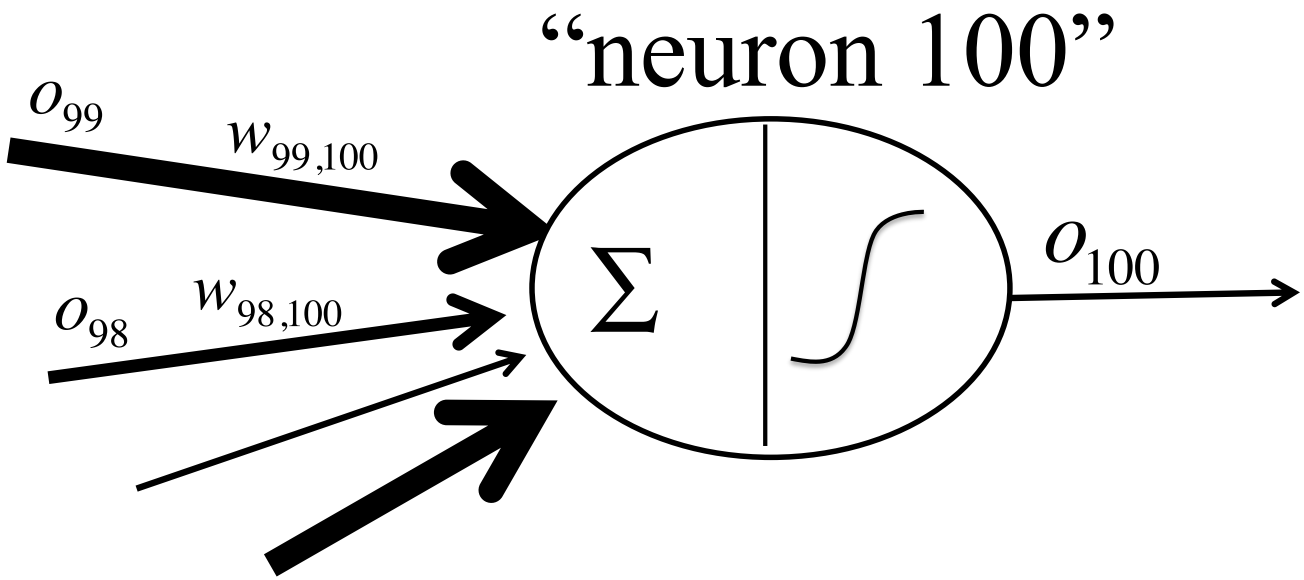
“Sigmoid”



Activation
Function



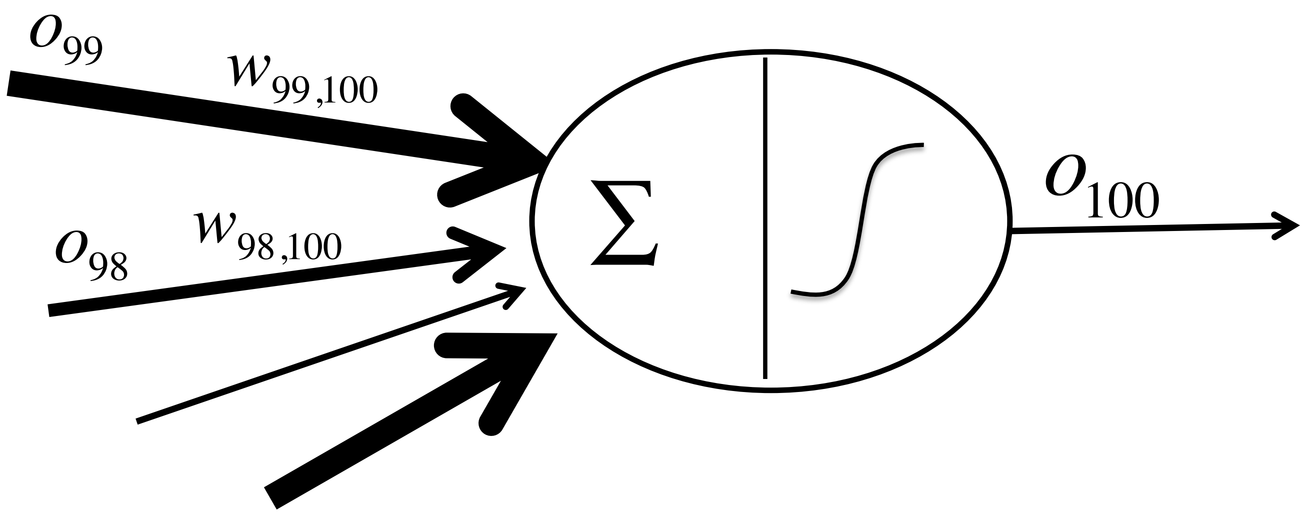
Single Layer



$$E = \frac{1}{2}(y - o_{100})^2$$

Error on one observation

Single Layer



$$E = \frac{1}{2}(y - o_{100})^2$$

Error on one observation

- In a brain, the synapses strengthen and weaken in order to learn.
- Say the same thing happens here.
- How should we set the weights in order to learn (reduce the error)?
- Minimize E with respect to the weights.

Backpropagation

- An algorithm that trains the weights of a neural network
- Requires us to propagate information backwards through the network, then forwards, then backwards, then forwards, etc.
- Propagate backwards = chain rule from calculus.

Backpropagation

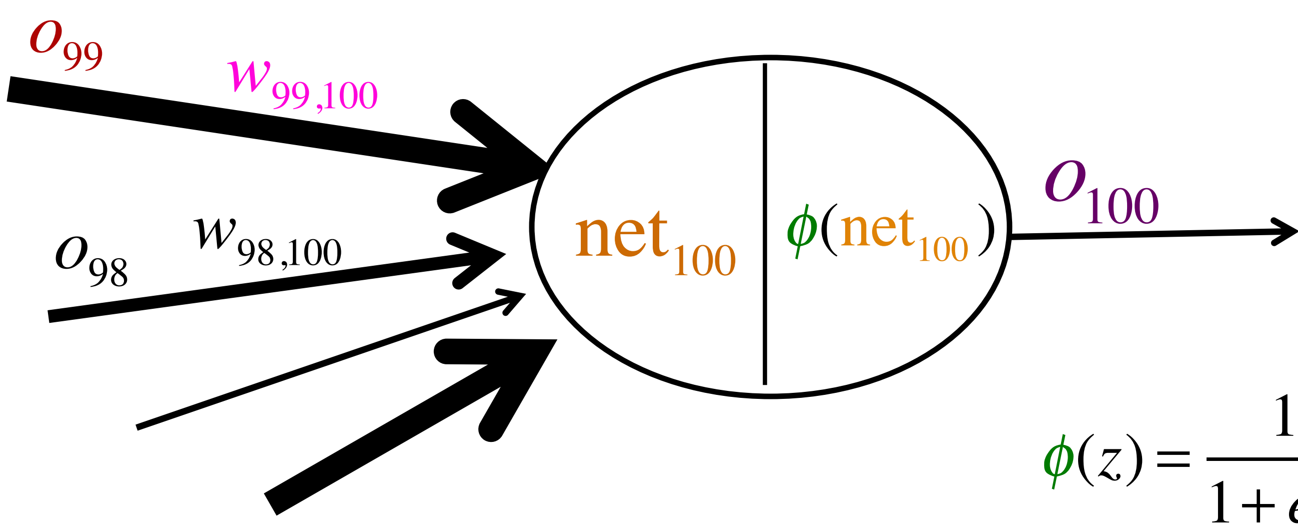
Cynthia Rudin

Duke Machine Learning

Backpropagation

- An algorithm that trains the weights of a neural network
- Requires us to propagate information backwards through the network, then forwards, then backwards, then forwards, etc.
- Propagate backwards = chain rule from calculus.

Single Layer



$$E = \frac{1}{2}(y - o_{100})^2$$

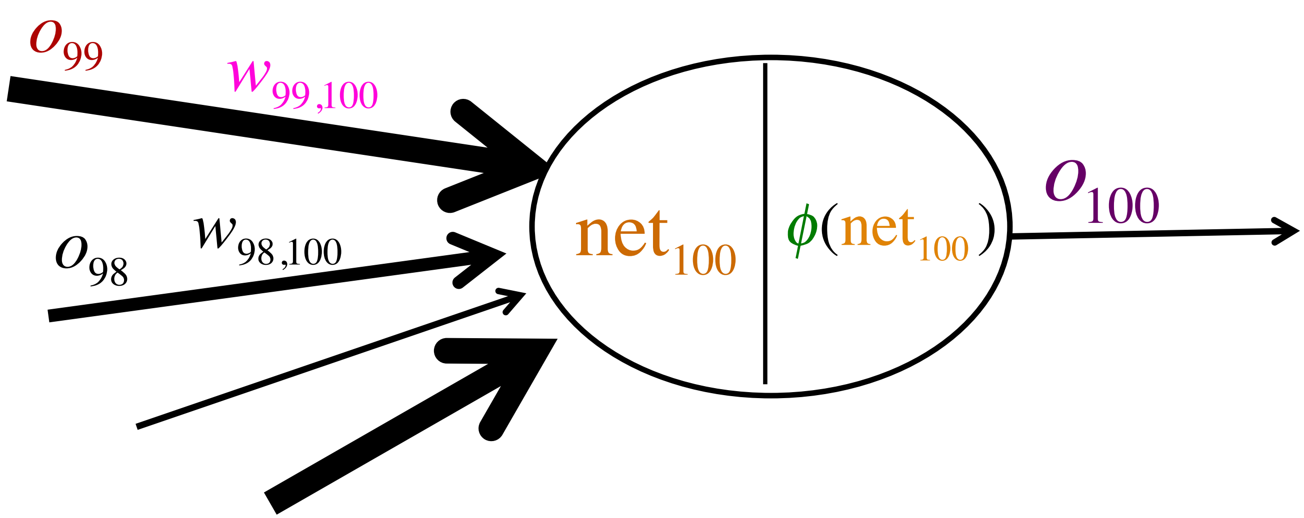
Error on one observation

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

$$\phi'(z) = \frac{d\phi(z)}{dz} = \phi(z)(1-\phi(z))$$

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

Single Layer



$$E = \frac{1}{2}(y - o_{100})^2$$

Error on one observation

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

Single Layer

$$E = \frac{1}{2}(y - o_{100})^2$$

Error on one observation

$$\frac{dE}{do_{100}} = -\frac{1}{2}2(y - o_{100})$$

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

$$-(y - o_{100})$$

Single Layer

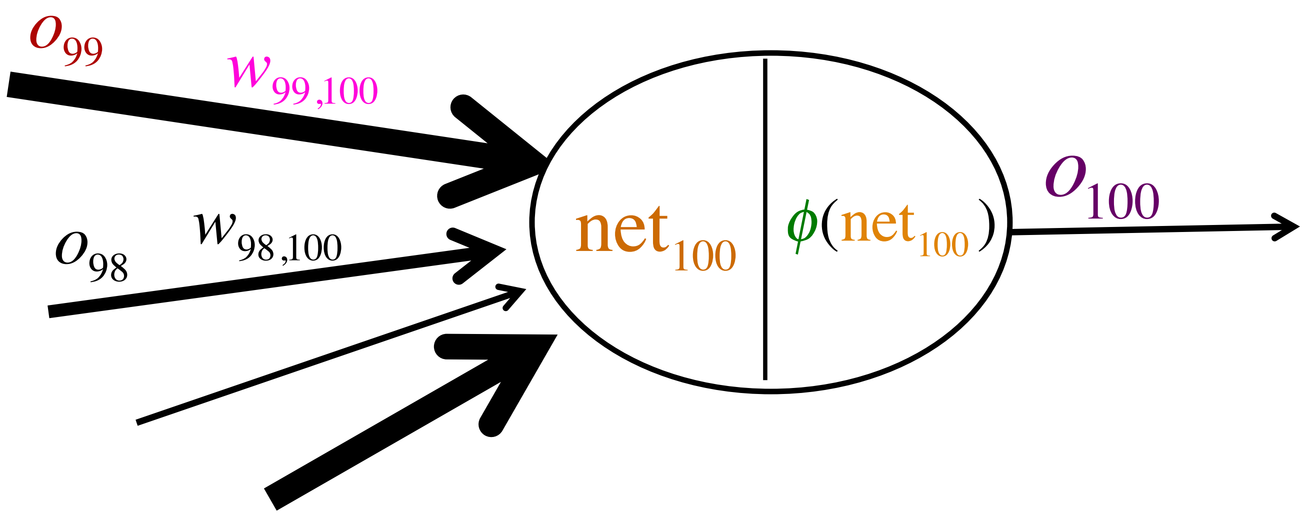
$$E = \frac{1}{2}(y - o_{100})^2$$

Error on one observation

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \mathrm{net}_{100}} \frac{d\ \mathrm{net}_{100}}{dw_{99,100}}$$

$$-(y - o_{100})$$

Single Layer



$$E = \frac{1}{2}(y - o_{100})^2$$

Error on one observation

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$

Single Layer

$$\phi(\text{net}_{100}) \quad o_{100}$$

$$\frac{do_{100}}{d\ \text{net}_{100}} = \frac{d\phi(\text{net}_{100})}{d\ \text{net}_{100}} = \phi'(\text{net}_{100}) = \phi(\text{net}_{100})(1-\phi(\text{net}_{100})) = o_{100}(1-o_{100})$$

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

$$-(y-o_{100})$$

Single Layer

$$\phi(\text{net}_{100}) \qquad o_{100}$$

$$\frac{do_{100}}{d\ \text{net}_{100}} = \frac{d\phi(\text{net}_{100})}{d\ \text{net}_{100}} = \phi'(\text{net}_{100}) = \phi(\text{net}_{100})(1-\phi(\text{net}_{100})) = o_{100}(1-o_{100})$$

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

$$-(y-o_{100}) \qquad\qquad o_{100}(1-o_{100})$$

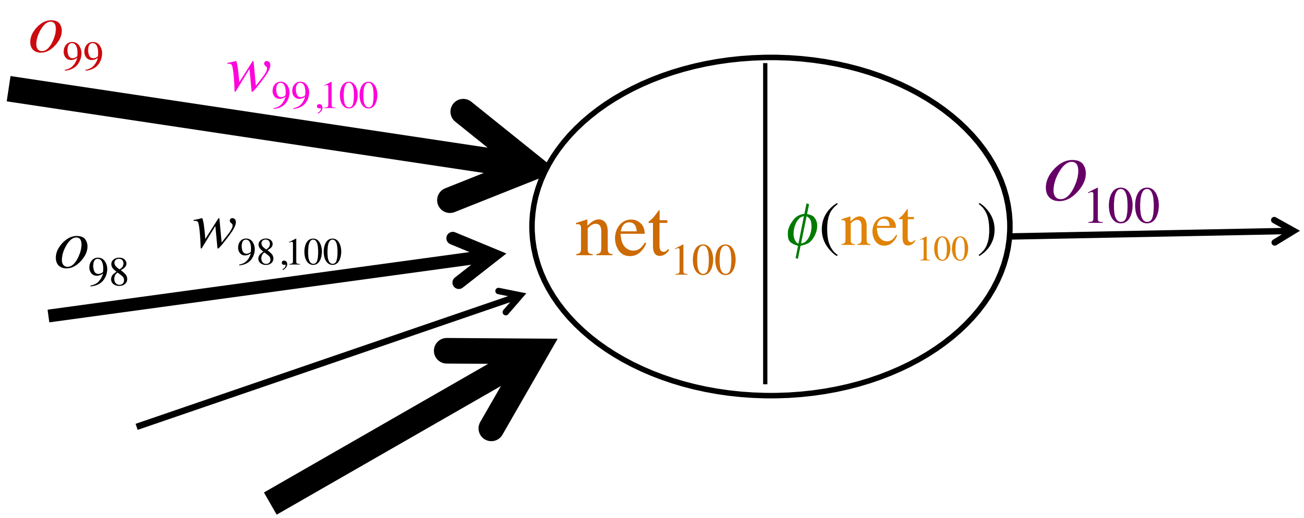
Single Layer

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \mathrm{net}_{100}} \frac{d\ \mathrm{net}_{100}}{dw_{99,100}}$$

$$-(y - o_{100})$$

$$o_{100}(1 - o_{100})$$

Single Layer



$$E = \frac{1}{2}(y - o_{100})^2$$

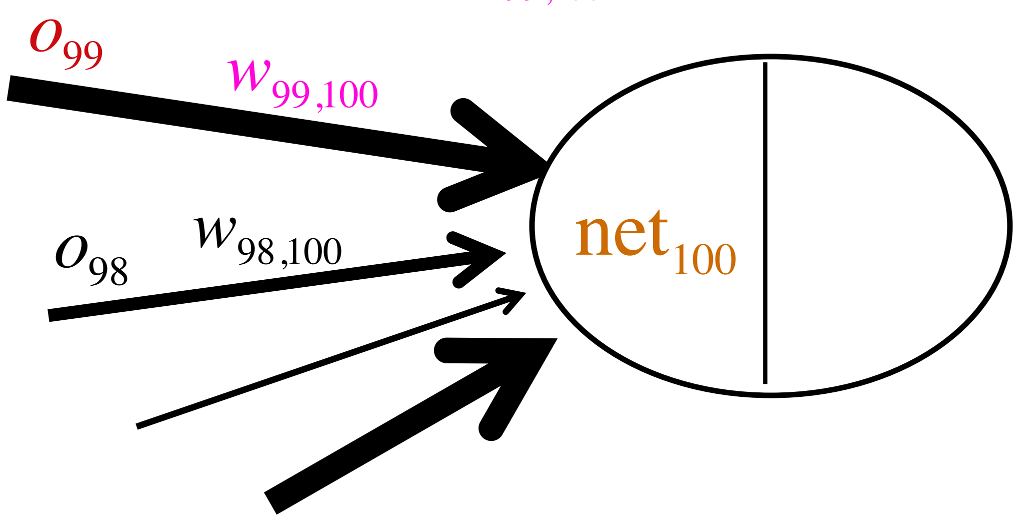
Error on one observation

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \text{net}_{100}} \frac{d\ \text{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$

$o_{100}(1 - o_{100})$

$$\frac{d\ \mathrm{net}_{100}}{dw_{99,100}} = \frac{d\ (w_{99,100}o_{99} + w_{98,100}o_{98} + w_{97,100}o_{97} + \ldots)}{dw_{99,100}} = o_{99}$$



$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \mathrm{net}_{100}} \frac{d\ \mathrm{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$ $\quad$ $o_{100}(1 - o_{100})$ $\quad$ o_{99}

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\ \mathrm{net}_{100}} \frac{d\ \mathrm{net}_{100}}{dw_{99,100}}$$

o_{99}

$-(y - o_{100})$

$o_{100}(1 - o_{100})$

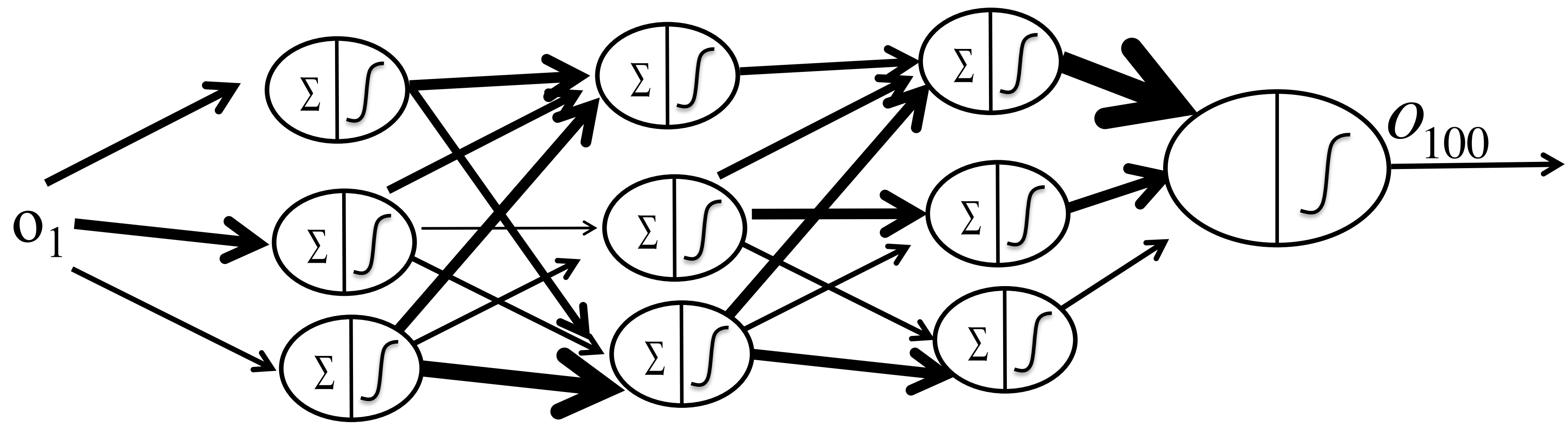
We will need this later – it depends only on node 100

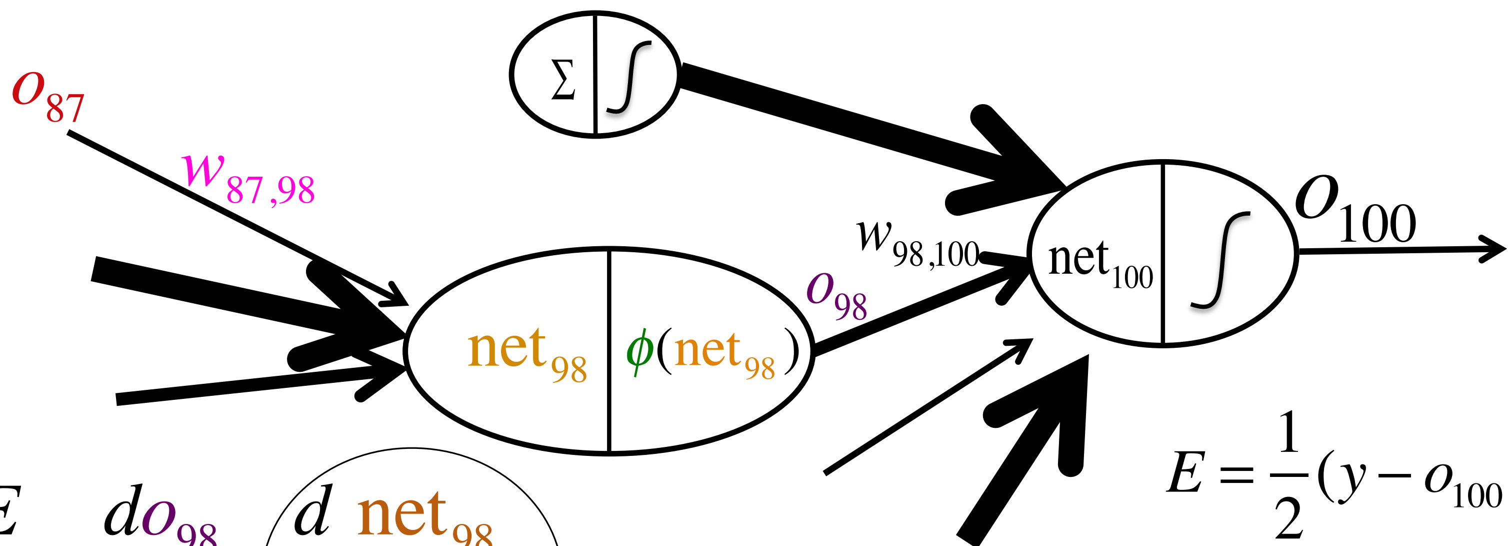
$$\frac{dE}{dw_{99,100}} = \overbrace{\frac{dE}{do_{100}} \frac{do_{100}}{d\ \mathrm{net}_{100}}}^{\delta_{100}} \frac{d\ \mathrm{net}_{100}}{dw_{99,100}}$$

$$= (-(y - o_{100}))o_{100}(1 - o_{100})\,o_{99}$$

Backpropagation

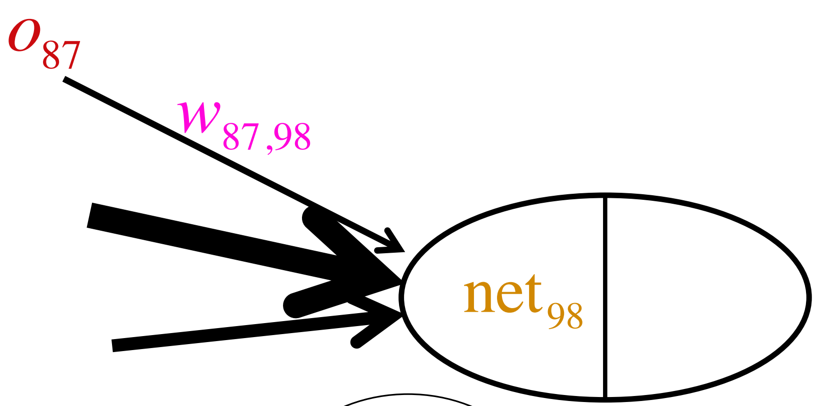
- Go one layer deeper.


o_1
o_{100}



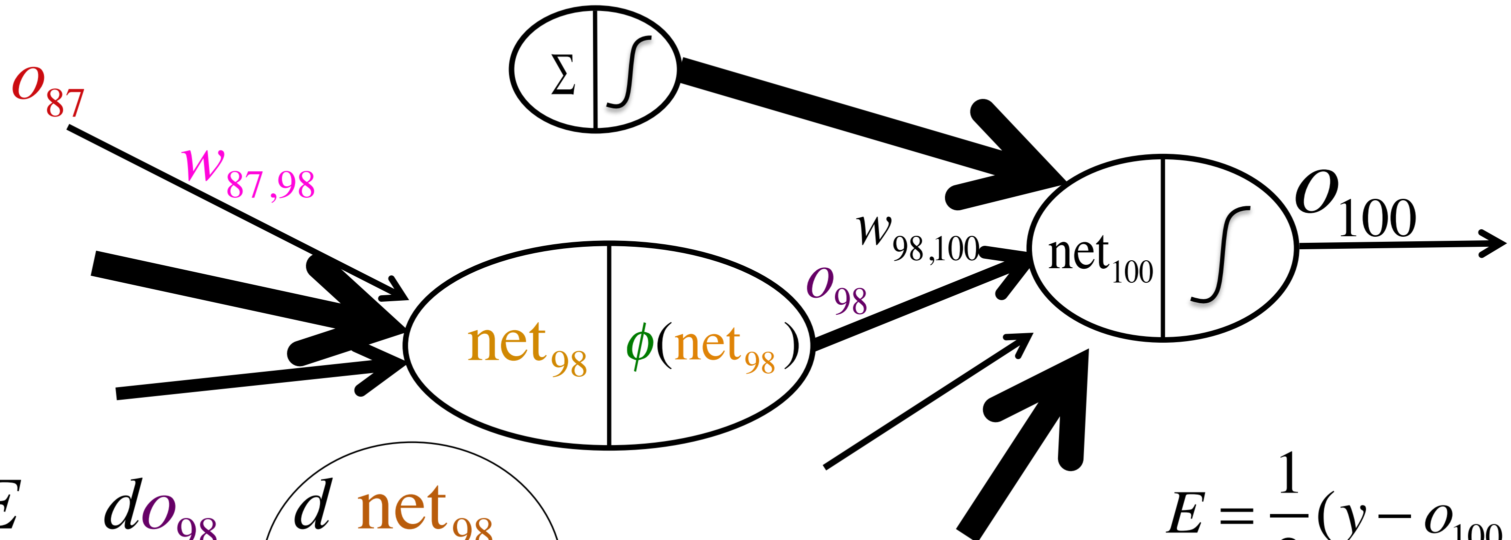
$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}}\frac{do_{98}}{d\ \mathrm{net}_{98}}\frac{d\ \mathrm{net}_{98}}{dw_{87,98}}$$



$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\ \text{net}_{98}} \frac{d\ \text{net}_{98}}{dw_{87,98}}$$

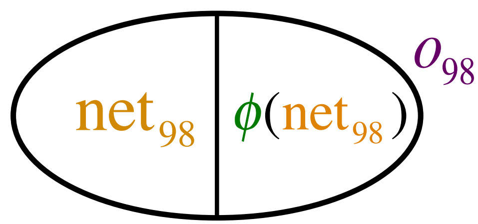
$$\frac{d\ \text{net}_{98}}{dw_{87,98}} = \frac{d\ (w_{87,98}o_{87} + w_{86,98}o_{86} + w_{85,98}o_{85} + \ldots)}{dw_{87,98}} = o_{87}$$



$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\ \text{net}_{98}} \frac{d\ \text{net}_{98}}{dw_{87,98}}$$

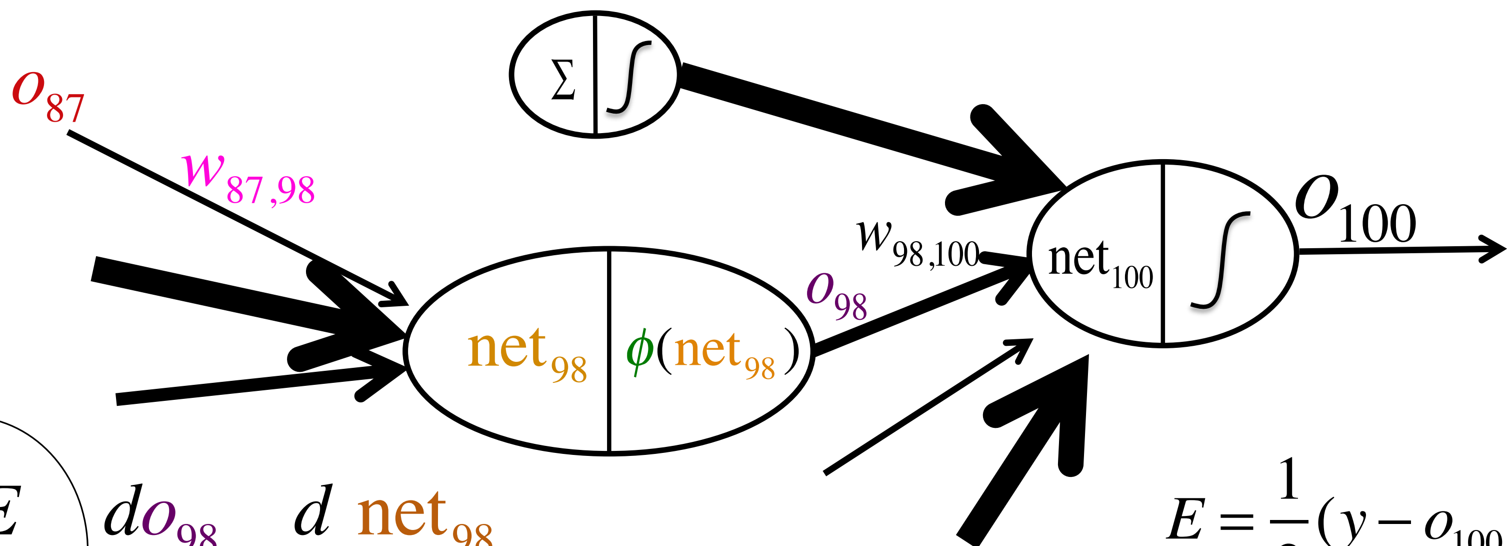
o_{87}



$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\ net_{98}} \frac{d\ net_{98}}{dw_{87,98}}$$

o_{87}

$$\frac{do_{98}}{d\ net_{98}} = \frac{d\phi(net_{98})}{d\ net_{98}} = \phi'(net_{98}) = \phi(net_{98})(1-\phi(net_{98})) = o_{98}(1-o_{98})$$



$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\ \text{net}_{98}} \frac{d\ \text{net}_{98}}{dw_{87,98}}$$

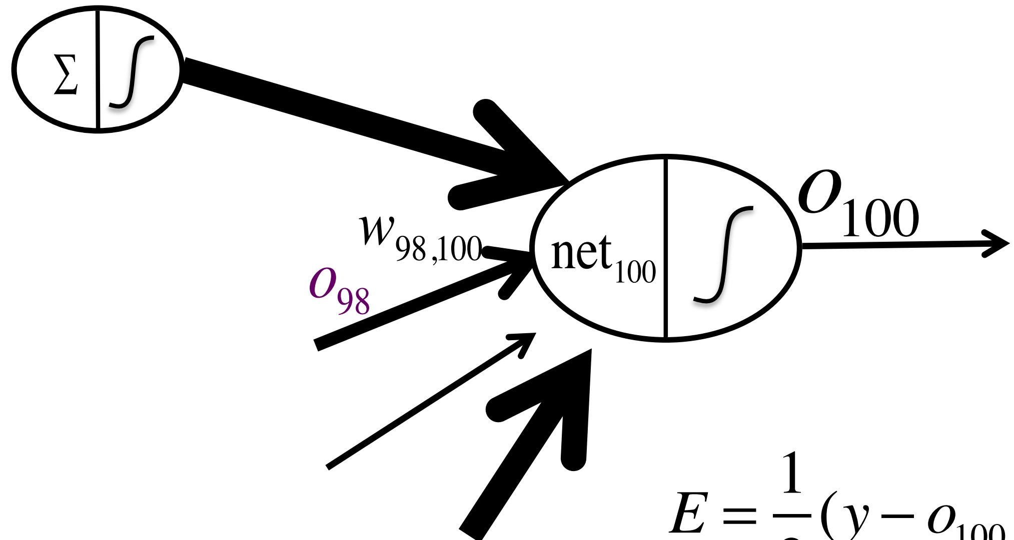
o_{87}

$o_{98}(1 - o_{98})$

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\ \mathrm{net}_{98}} \frac{d\ \mathrm{net}_{98}}{dw_{87,98}}$$

o_{87}

$o_{98}(1 - o_{98})$



$$E = \frac{1}{2}(y - o_{100})^2$$

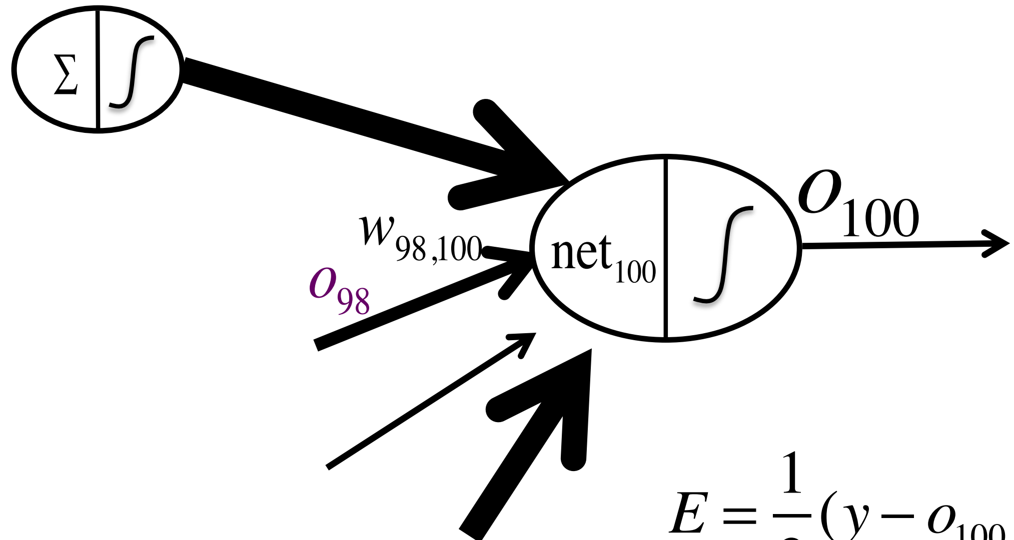
$$\frac{dE}{do_{98}} = \frac{dE}{d\,\text{net}_{100}} \frac{d\,\text{net}_{100}}{do_{98}}$$

δ_{100}

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\,\text{net}_{98}} \frac{d\,\text{net}_{98}}{dw_{87,98}}$$

o_{87}

$o_{98}(1 - o_{98})$

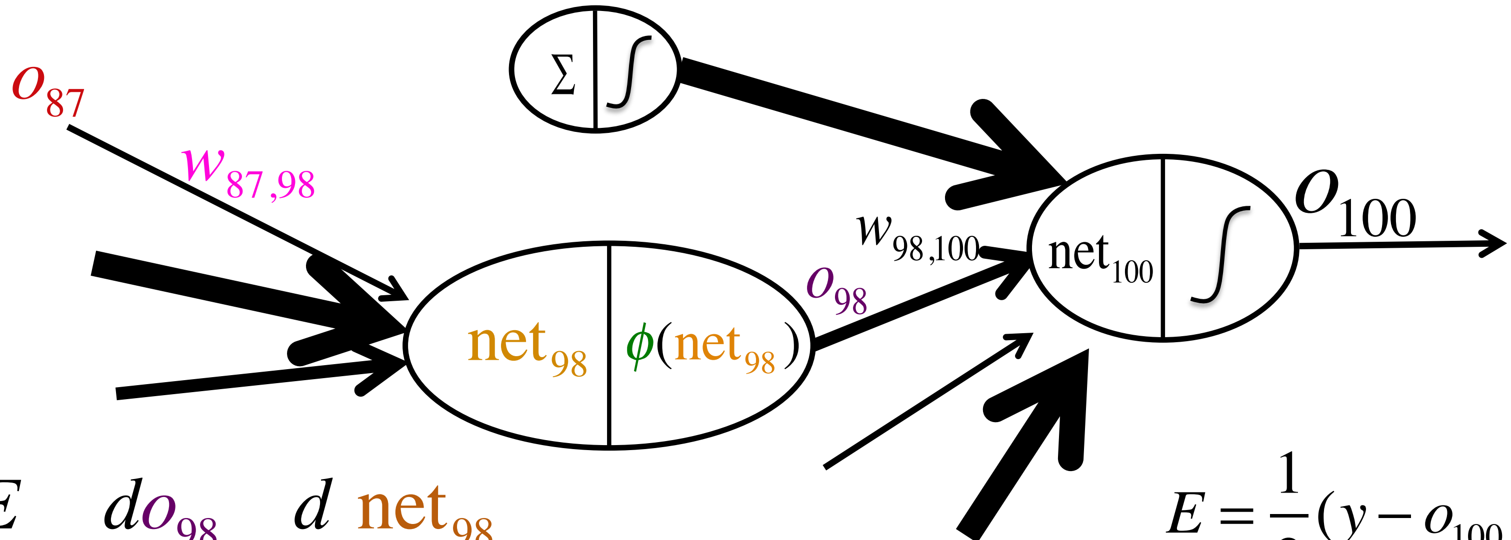


$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{d\,\mathrm{net}_{100}}{do_{98}} = \frac{d(w_{99,100}o_{99} + w_{98,100}o_{98} + \ldots)}{do_{98}} = w_{98,100}$$

$$\frac{dE}{do_{98}} = \underbrace{\frac{dE}{d\,\mathrm{net}_{100}}}_{\delta_{100}} \frac{d\,\mathrm{net}_{100}}{do_{98}}$$

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \underbrace{\frac{do_{98}}{d\ \mathrm{net}_{98}}}_{o_{98}(1-o_{98})} \underbrace{\frac{d\ \mathrm{net}_{98}}{dw_{87,98}}}_{o_{87}}$$



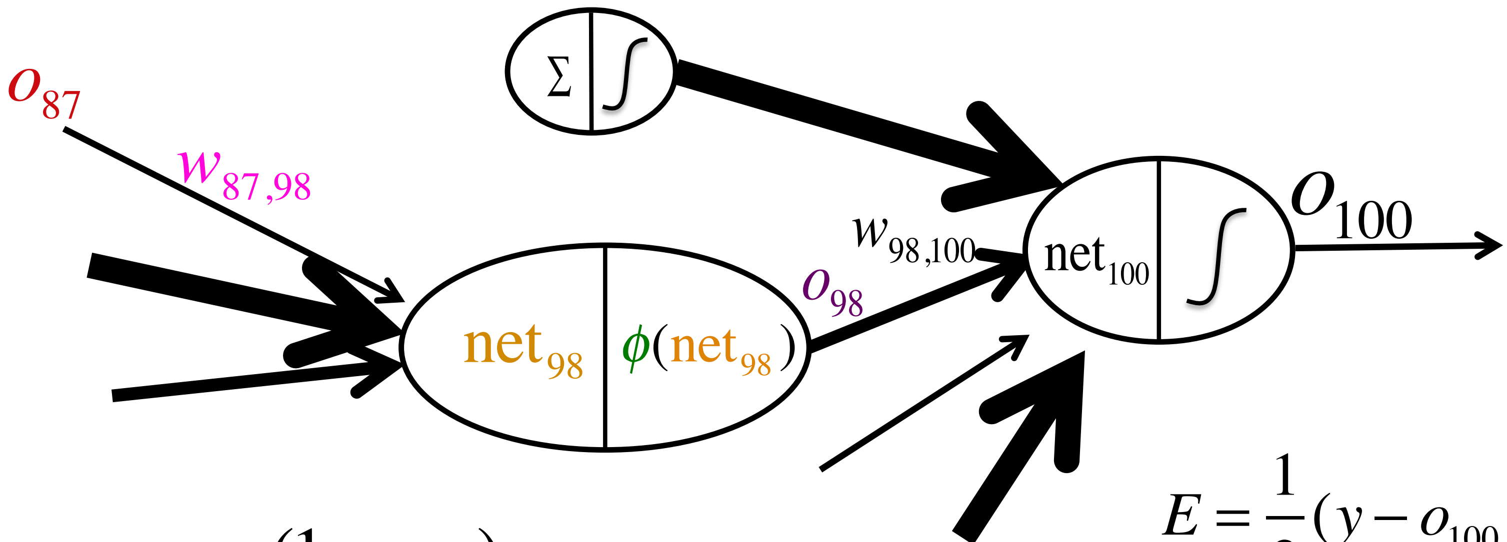
$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\ \text{net}_{98}} \frac{d\ \text{net}_{98}}{dw_{87,98}}$$

o_{87}

$\delta_{100} w_{98,100}$

$o_{98}(1 - o_{98})$

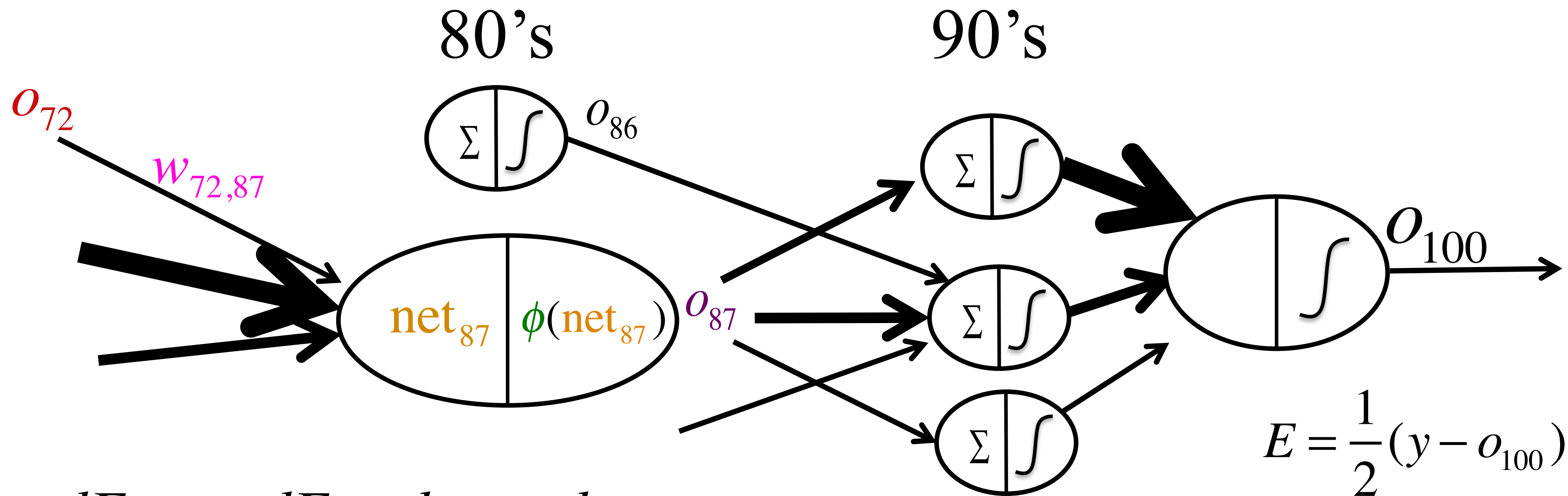


$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{87,98}} = \delta_{100} w_{98,100} o_{98}(1 - o_{98}) o_{87}$$

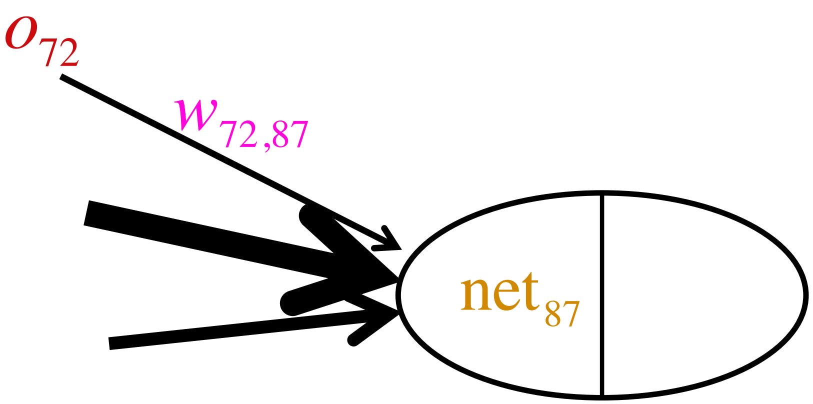
Backpropagation

- Go even one layer deeper.
- Third time is a charm.



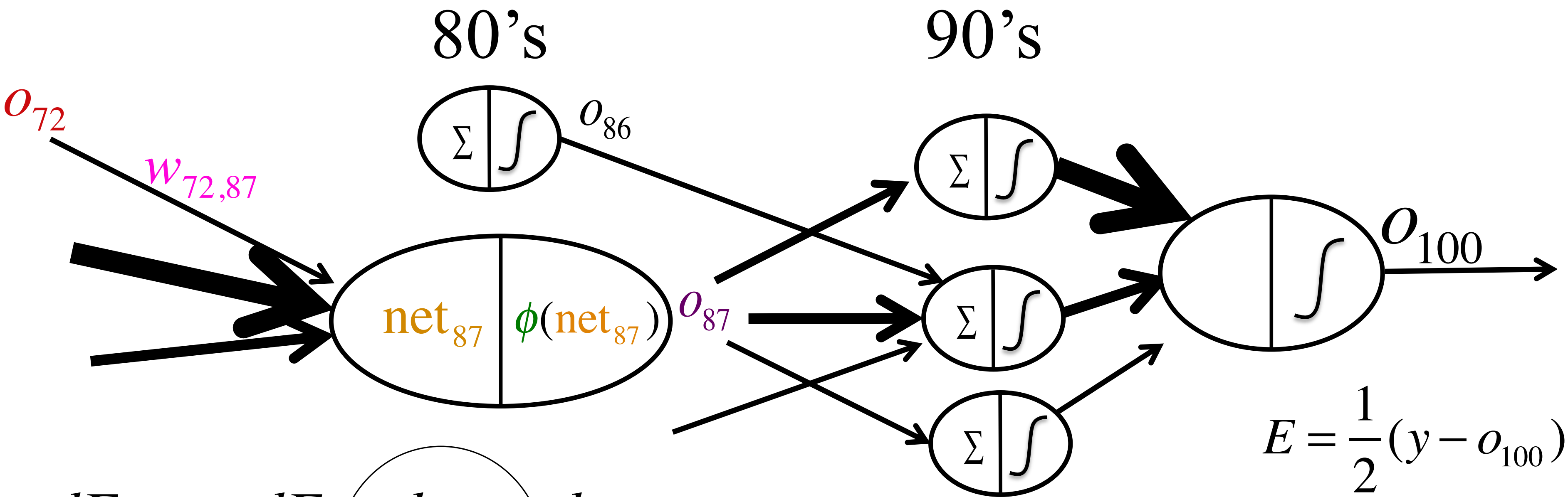
$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \mathrm{net}_{87}} \frac{d\ \mathrm{net}_{87}}{dw_{72,87}}$$



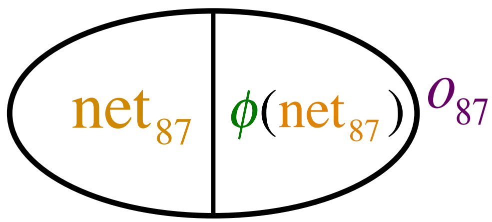
$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \mathrm{net}_{87}} \frac{d\ \mathrm{net}_{87}}{dw_{72,87}}$$

o_{72}



$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \mathrm{net}_{87}} \frac{d\ \mathrm{net}_{87}}{dw_{72,87}}$$

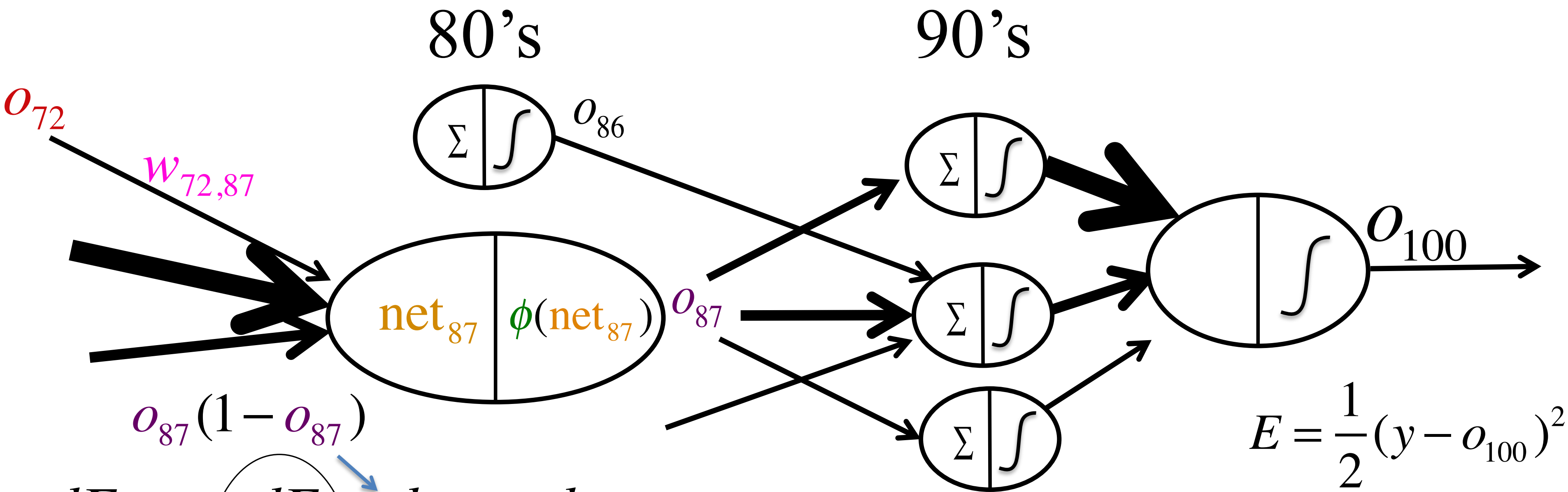
o_{72}



$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \text{net}_{87}} \frac{d\ \text{net}_{87}}{dw_{72,87}}$$

o_{72}

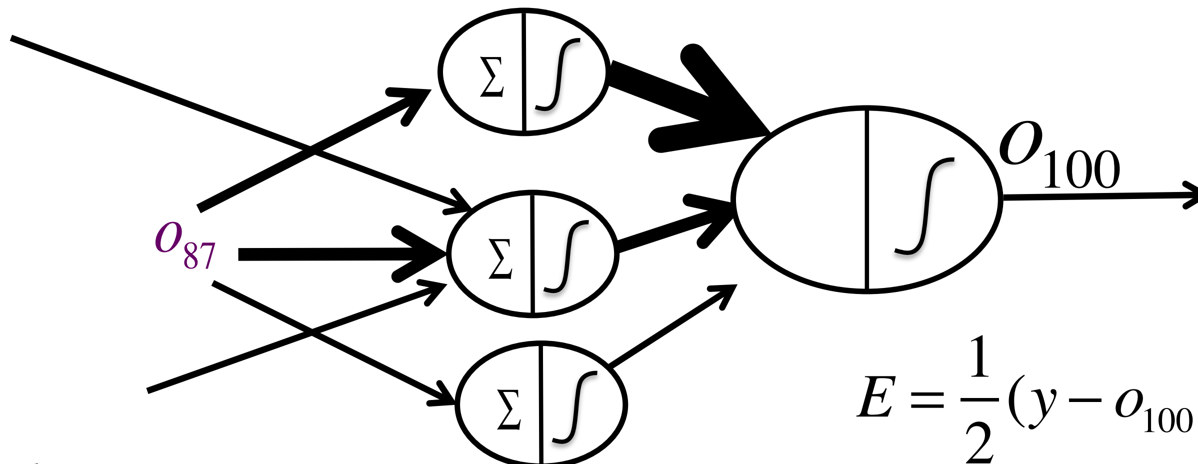
$$\frac{do_{87}}{d\ \text{net}_{87}} = \frac{d\phi(\text{net}_{87})}{d\ \text{net}_{87}} = \phi'(\text{net}_{87}) = \phi(\text{net}_{87})(1-\phi(\text{net}_{87})) = o_{87}(1-o_{87})$$



$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \text{net}_{87}} \frac{d\ \text{net}_{87}}{dw_{72,87}}$$

o_{72}

90's



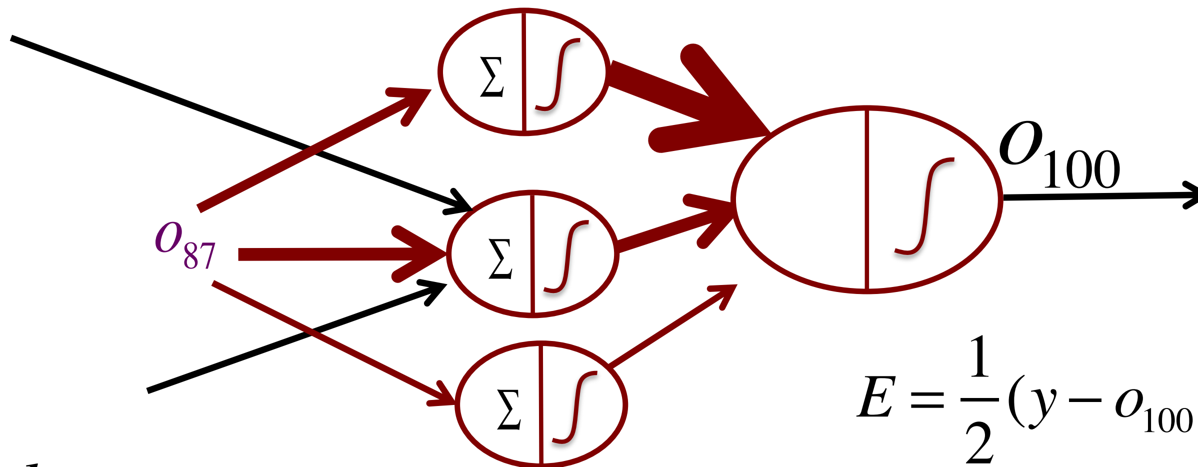
$$E = \frac{1}{2}(y - o_{100})^2$$

$$o_{87}(1 - o_{87})$$

$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \text{net}_{87}} \frac{d\ \text{net}_{87}}{dw_{72,87}}$$

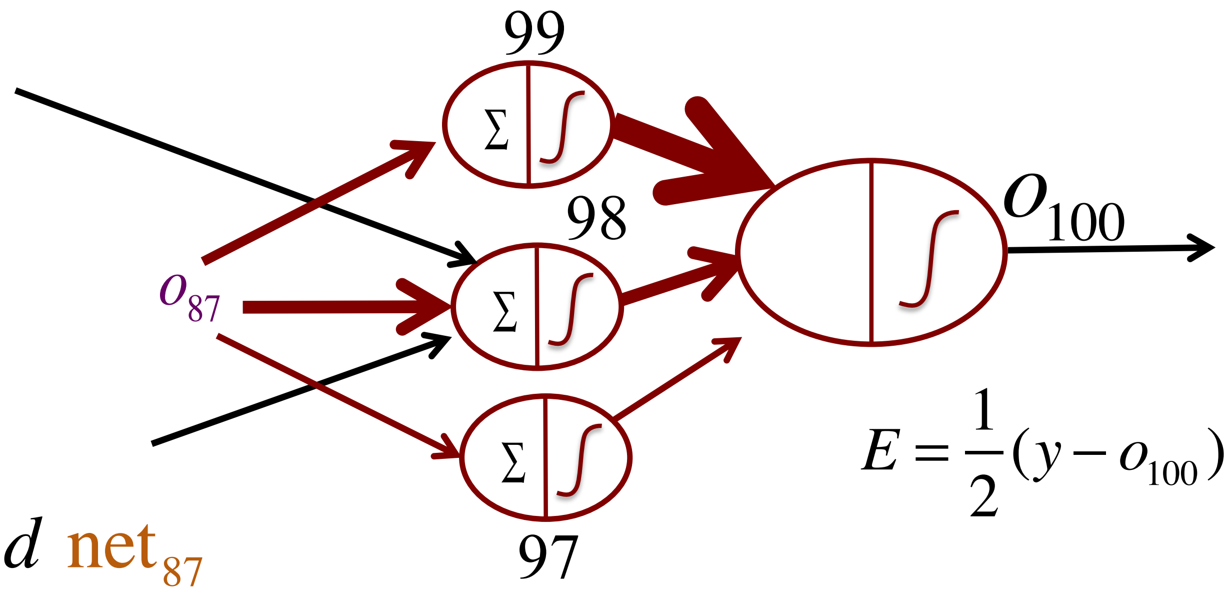
$$o_{72}$$

90's



$$E = \frac{1}{2}(y - o_{100})^2$$

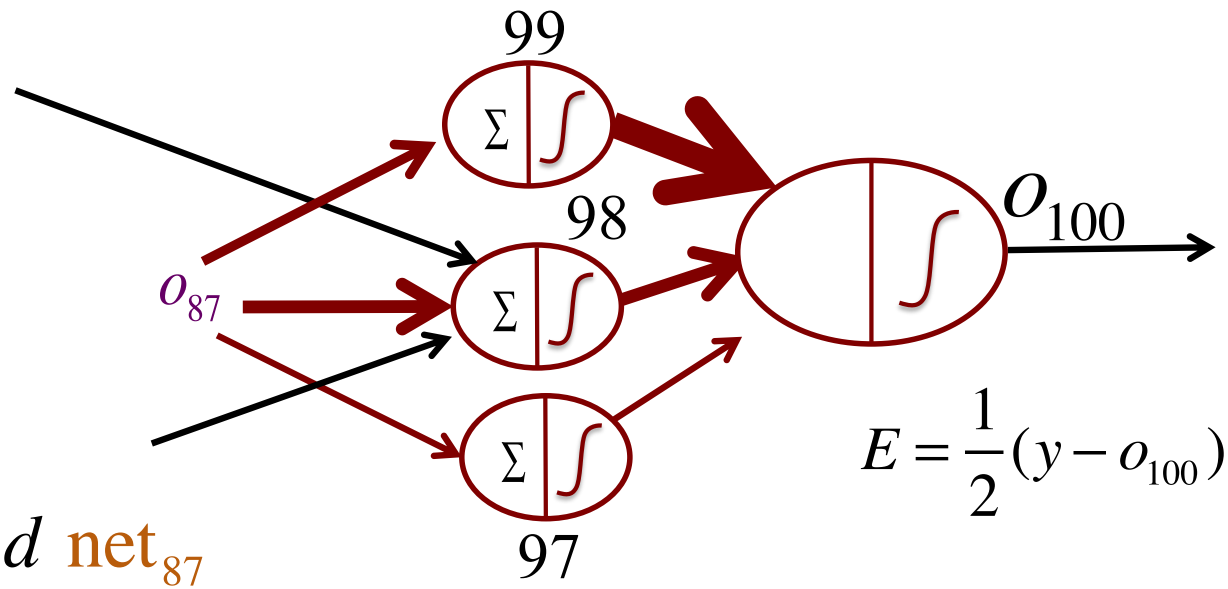
$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \overbrace{\frac{do_{87}}{d\ \mathrm{net}_{87}}}^{o_{87}(1-o_{87})} \underbrace{\frac{d\ \mathrm{net}_{87}}{dw_{72,87}}}_{o_{72}}$$



$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \overbrace{\frac{do_{87}}{d\ \mathrm{net}_{87}}}^{o_{87}(1-o_{87})} \underbrace{\frac{d\ \mathrm{net}_{87}}{dw_{72,87}}}_{o_{72}}$$

$$\frac{dE}{do_{87}} = \frac{dE}{d\ \mathrm{net}_{99}} \frac{d\ \mathrm{net}_{99}}{do_{87}} + \frac{dE}{d\ \mathrm{net}_{98}} \frac{d\ \mathrm{net}_{98}}{do_{87}} + \frac{dE}{d\ \mathrm{net}_{97}} \frac{d\ \mathrm{net}_{97}}{do_{87}} + \ldots$$



$$E = \frac{1}{2}(y - o_{100})^2$$

$o_{87}(1 - o_{87})$

$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d\ \mathrm{net}_{87}} \frac{d\ \mathrm{net}_{87}}{dw_{72,87}}$$

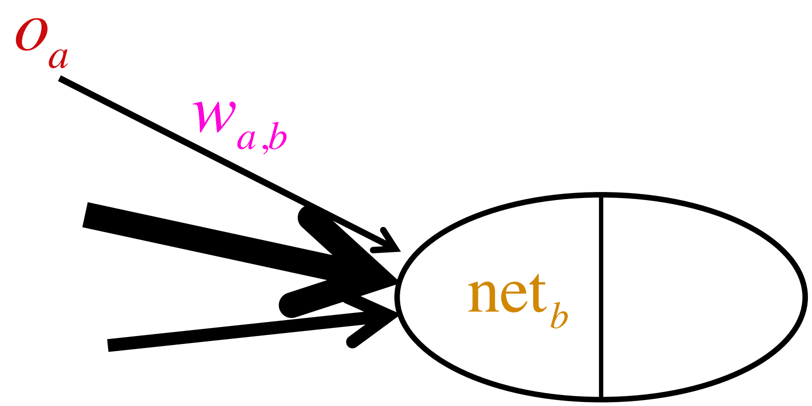
o_{72}

$$\frac{dE}{do_{87}} = \frac{dE}{d\ \mathrm{net}_{99}} \frac{d\ \mathrm{net}_{99}}{do_{87}} + \frac{dE}{d\ \mathrm{net}_{98}} \frac{d\ \mathrm{net}_{98}}{do_{87}} + \frac{dE}{d\ \mathrm{net}_{97}} \frac{d\ \mathrm{net}_{97}}{do_{87}} + \ldots$$

$$= \delta_{99} w_{87,99} + \delta_{98} w_{87,98} + \delta_{97} w_{87,97} + \ldots \qquad = \sum_{\ell \in L} \delta_{87,\ell}, w_{87,\ell}$$

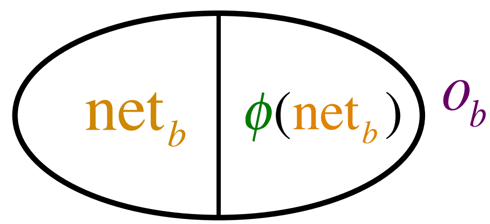
$$\frac{dE}{dw_{a,b}} = \frac{dE}{do_b}\frac{do_b}{d\ \mathrm{net}_b}\frac{d\ \mathrm{net}_b}{dw_{a,b}}$$

$$= \frac{dE}{do_b}\frac{do_b}{d\ \mathrm{net}_b}\, o_a$$



$$\frac{dE}{dw_{a,b}} = \frac{dE}{do_b}\,\frac{do_b}{d\ \mathrm{net}_b}\,\frac{d\ \mathrm{net}_b}{dw_{a,b}}$$

$$= \frac{dE}{do_b}\, o_b(1 - o_b)\, o_a$$

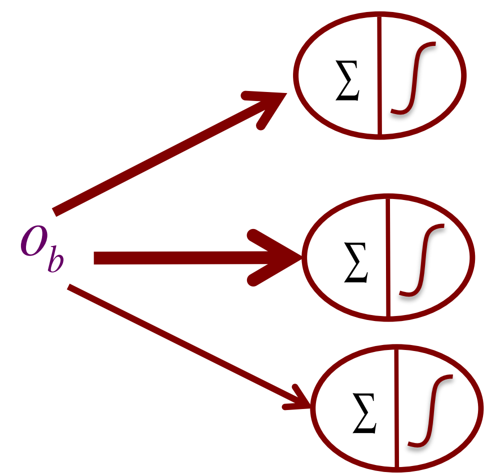


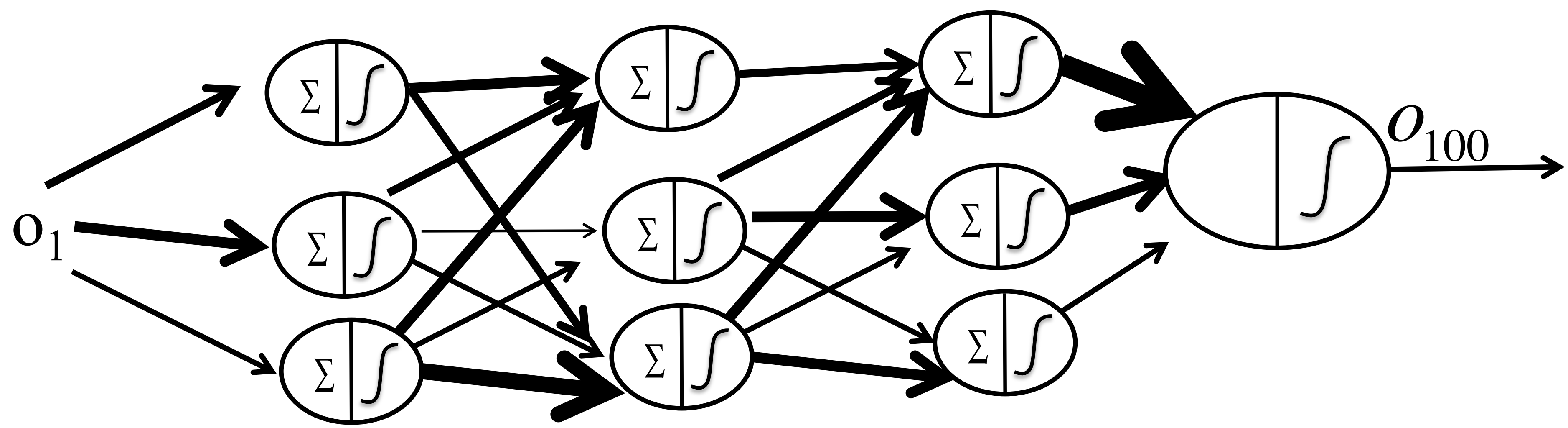
$$\frac{dE}{dw_{a,b}} = \frac{dE}{do_b}\frac{do_b}{d\ \text{net}_b}\frac{d\ \text{net}_b}{dw_{a,b}}$$

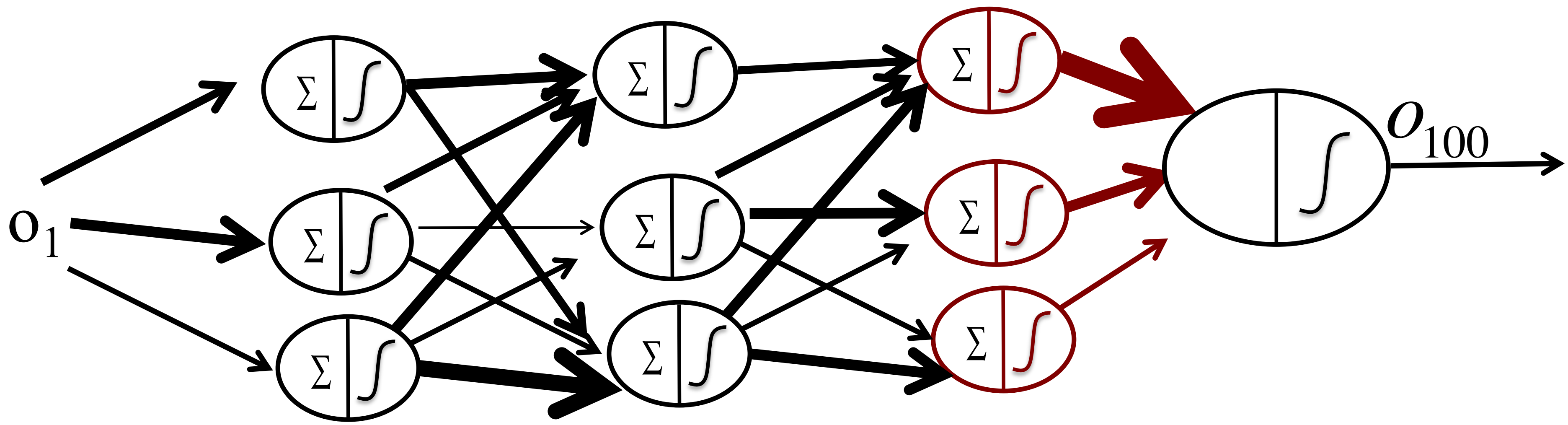
$$= \left(\sum_{\ell \in L} \delta_\ell w_{b,\ell}\right) o_b(1-o_b)o_a$$

The $\ell' s$ are downstream. We must have already computed all the δ_ℓ's ahead of us to compute this.

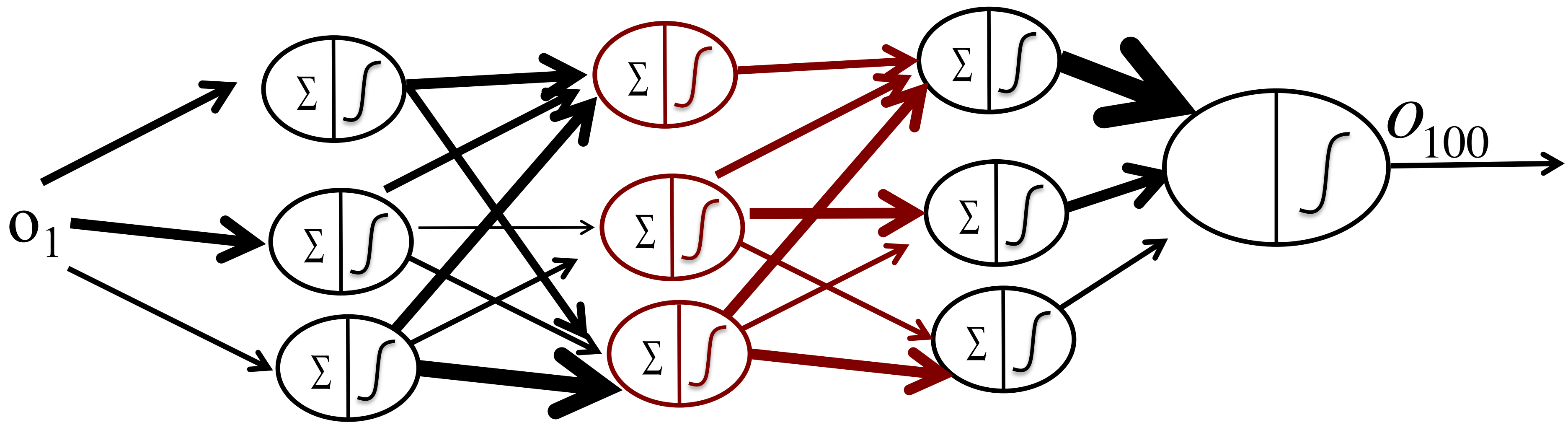
ℓ 's

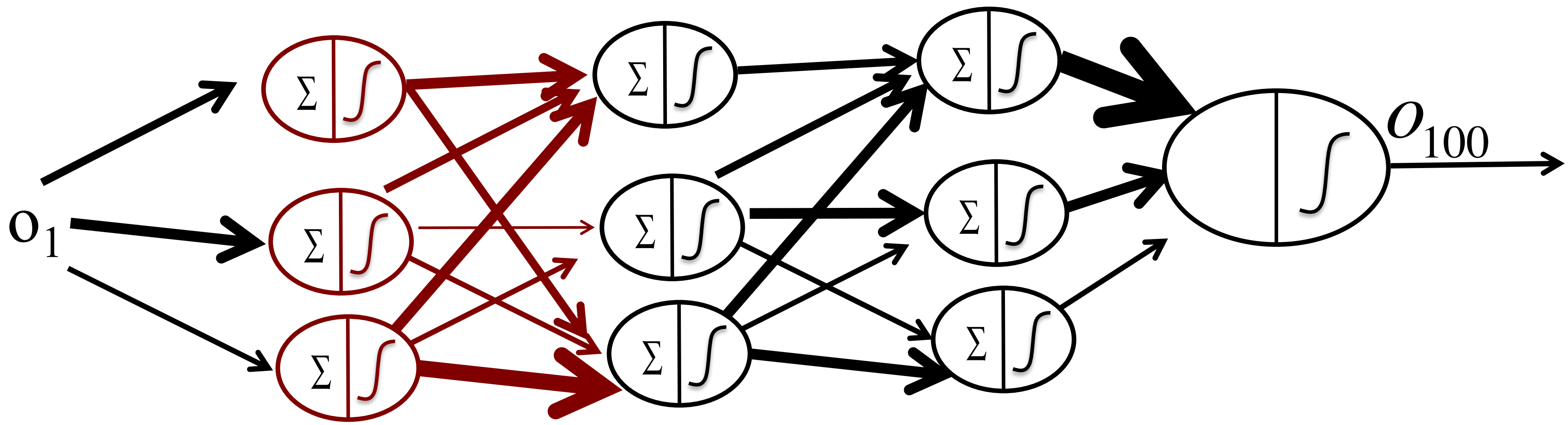



o_1
Σ
o_{100}

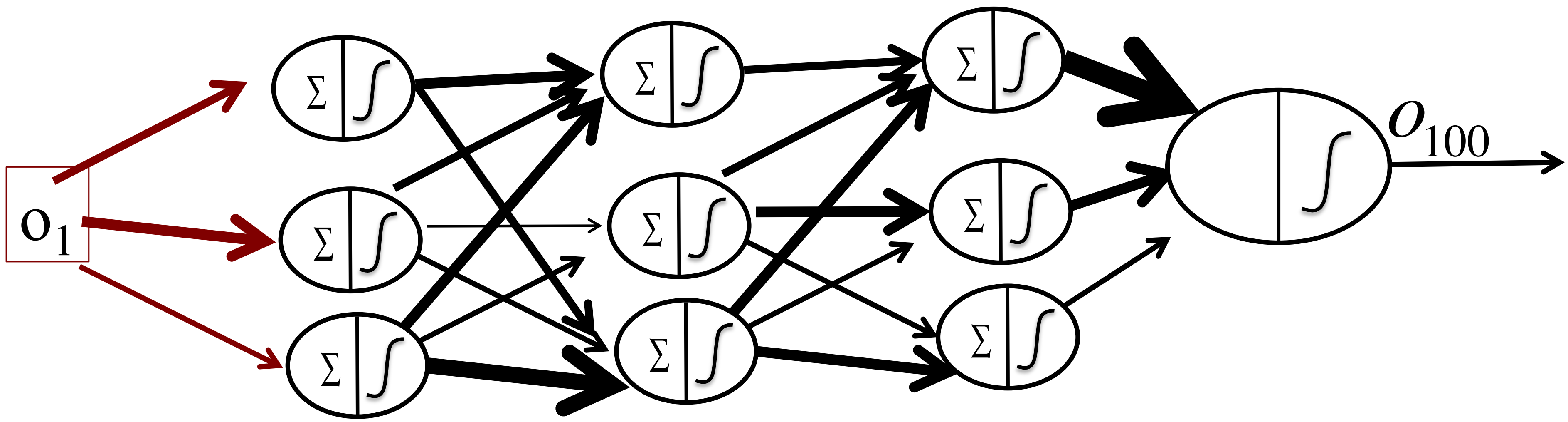


o_1
o_{100}

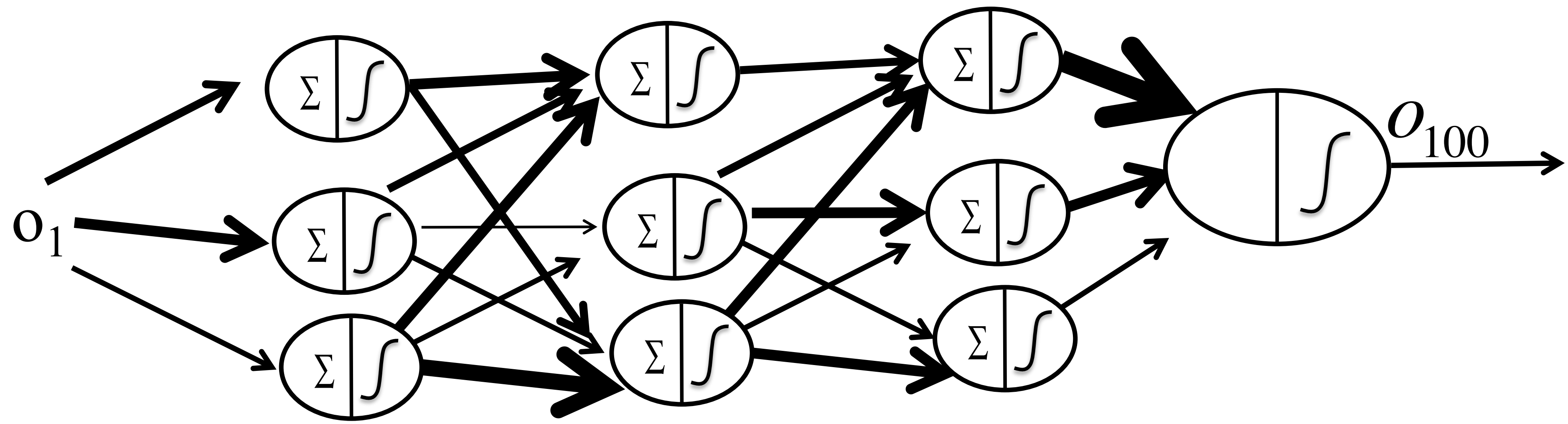

o_1
o_{100}



o_1
o_{100}



o_1
o_{100}

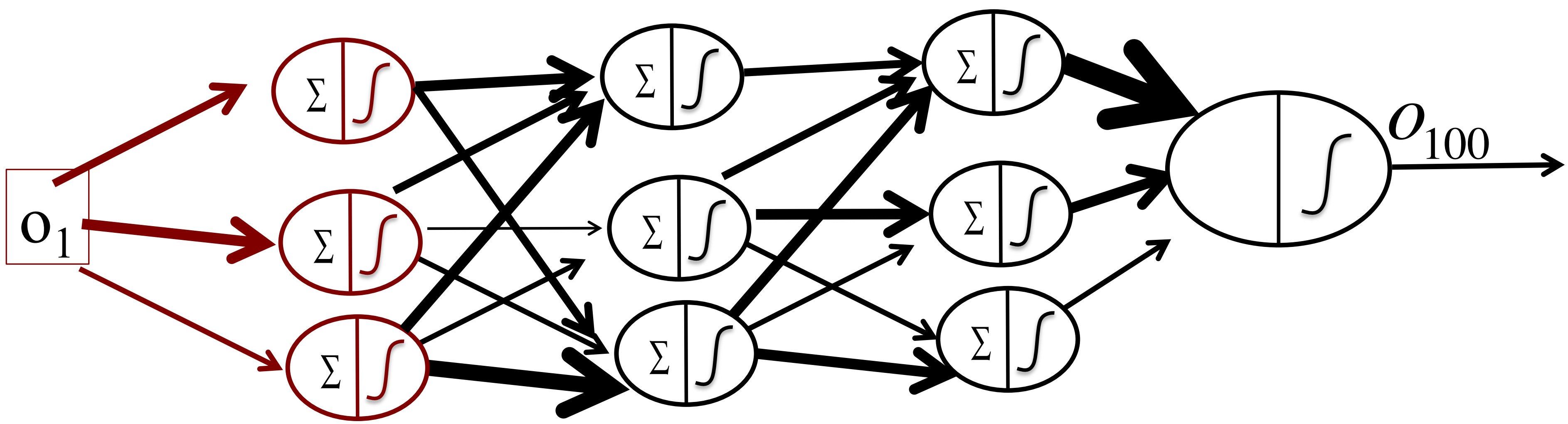

o1
o100

Backpropagation

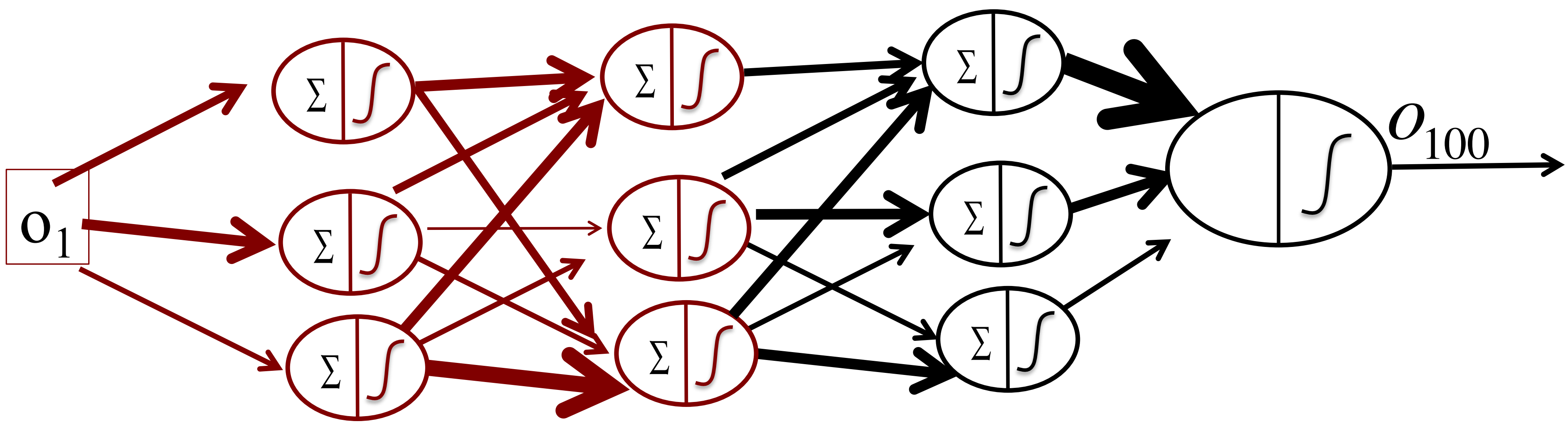
- Now we know how to compute $\dfrac{dE}{dw_{a,b}}$ for all $w_{a,b}$'s.
- Let's do gradient descent.

$$w_{a,b} \longleftarrow w_{a,b} - \alpha \frac{dE}{dw_{a,b}}$$

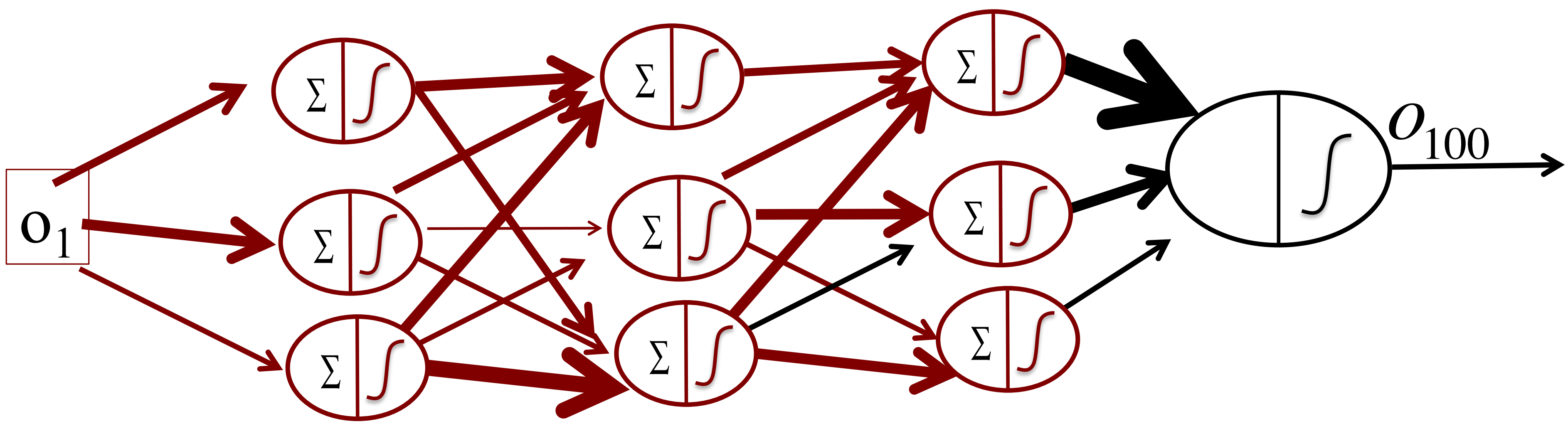
- α is between 0 and 1. Called the "learning rate".
- Now we know how to propagate errors back through the network.
- Remember how to go forward?



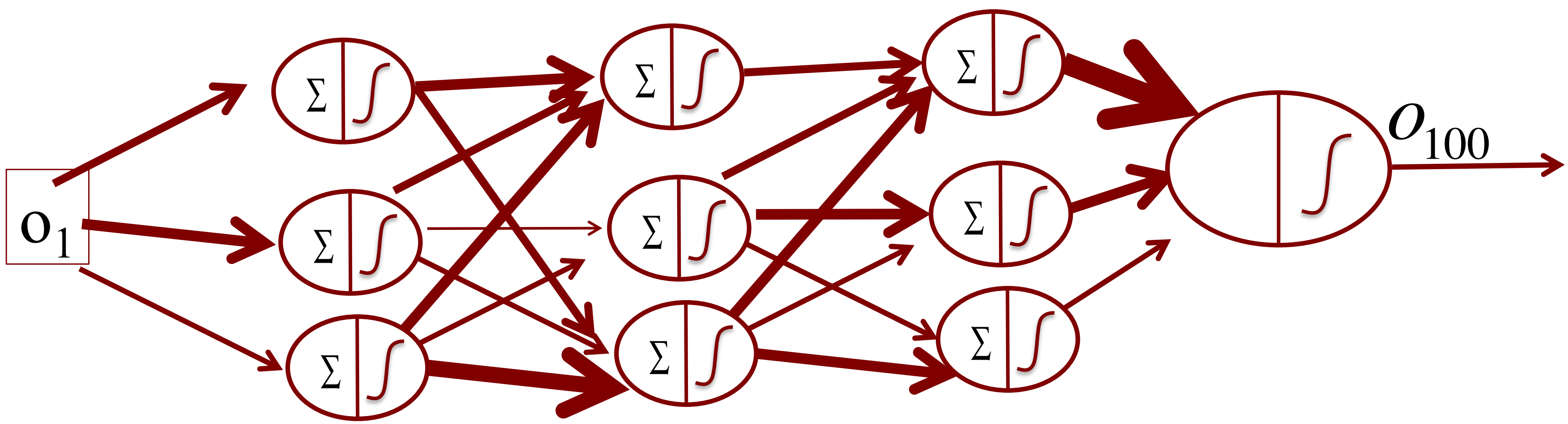
o_1
Σ
o_{100}



o_1
o_{100}



o1
o100



o_1
Σ
Σ
Σ
Σ
Σ
Σ
Σ
Σ
Σ
o_{100}

Backpropagation

- Repeat going backwards (to calculate the gradients), adjusting the weights, and going forwards (to calculate the errors) over and over in order to learn.

Cross-Entropy is Logistic Loss

Cynthia Rudin

Duke Machine Learning

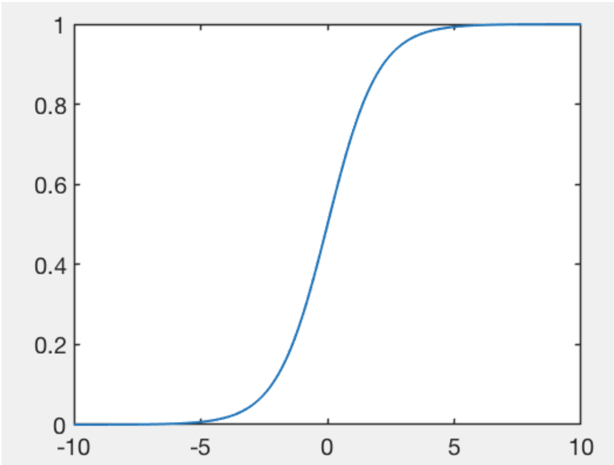
Convergence Problems in Neural Networks

Cynthia Rudin

Duke Machine Learning

Convergence Problems

- NN's have problems with convergence due to vanishing/exploding gradients and saddle points.
- Vanishing gradients come from the flat part of the activation function.
- Exploding gradients happen when we realize that our gradient has vanished and so increase the learning rate and take huge step sizes to compensate (but then mess everything up!)
- Stick to 10^{-5} to 10^{-3} learning rate perhaps?

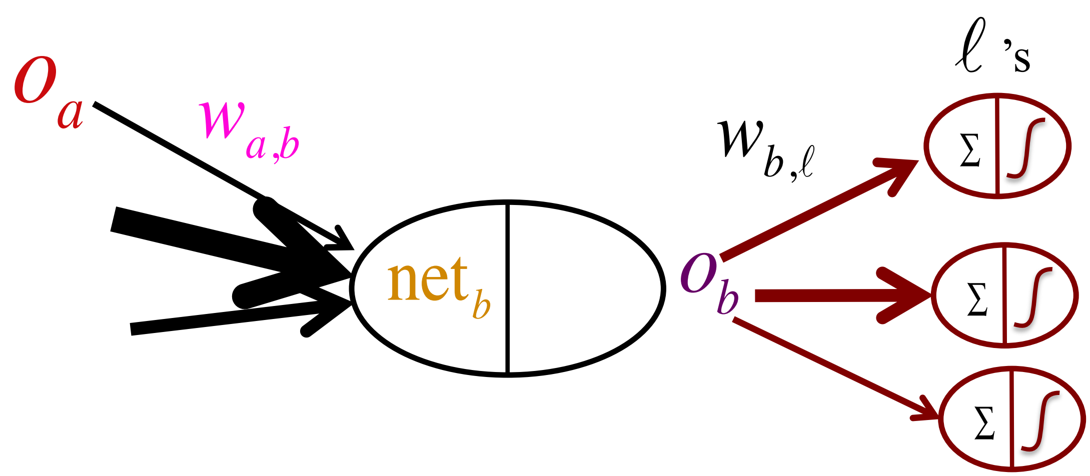


Convergence Problems

- With the sigmoid activation, the derivatives of the input weights for each node are always either all positive or all negative. This is a limitation.

$$\frac{dE}{dw_{a,b}} = \frac{dE}{do_b}\,\frac{do_b}{d\ \mathrm{net}_b}\,\frac{d\ \mathrm{net}_b}{dw_{a,b}}$$

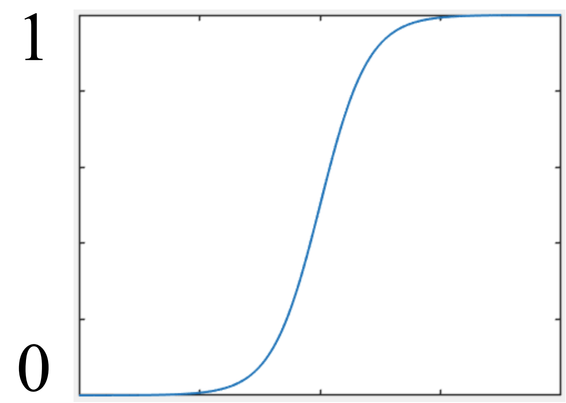
$$= \underbrace{\left(\sum_{\ell\in L}\delta_\ell w_{b,\ell}\right)}_{\text{does not depend on a}}\underbrace{o_b(1-o_b)o_a}_{\text{positive, since all outputs of sigmoid are between 0 and 1.}}$$



Convergence Problems

- Bottom line – most people do not use sigmoid-like activation functions, even though this is more biologically relevant.

Sigmoid $$\sigma(x) = \frac{1}{1+e^{-x}}$$

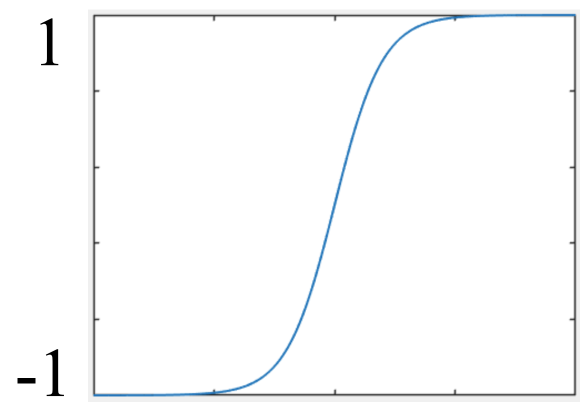


Convergence Problems

- Bottom line – most people do not use sigmoid-like activation functions, even though this is more biologically relevant.

Sigmoid $\sigma(x) = \dfrac{1}{1+e^{-x}}$

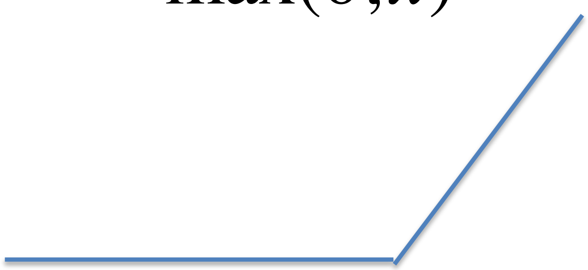
Hyperbolic tangent $\tanh(x)$



Convergence Problems

Rectified Linear Unit (ReLU)

$$\max(0, x)$$



Removes vanishing gradients when nodes are "activated," meaning $x>0$.

(Krizhevsky et al., 2012)

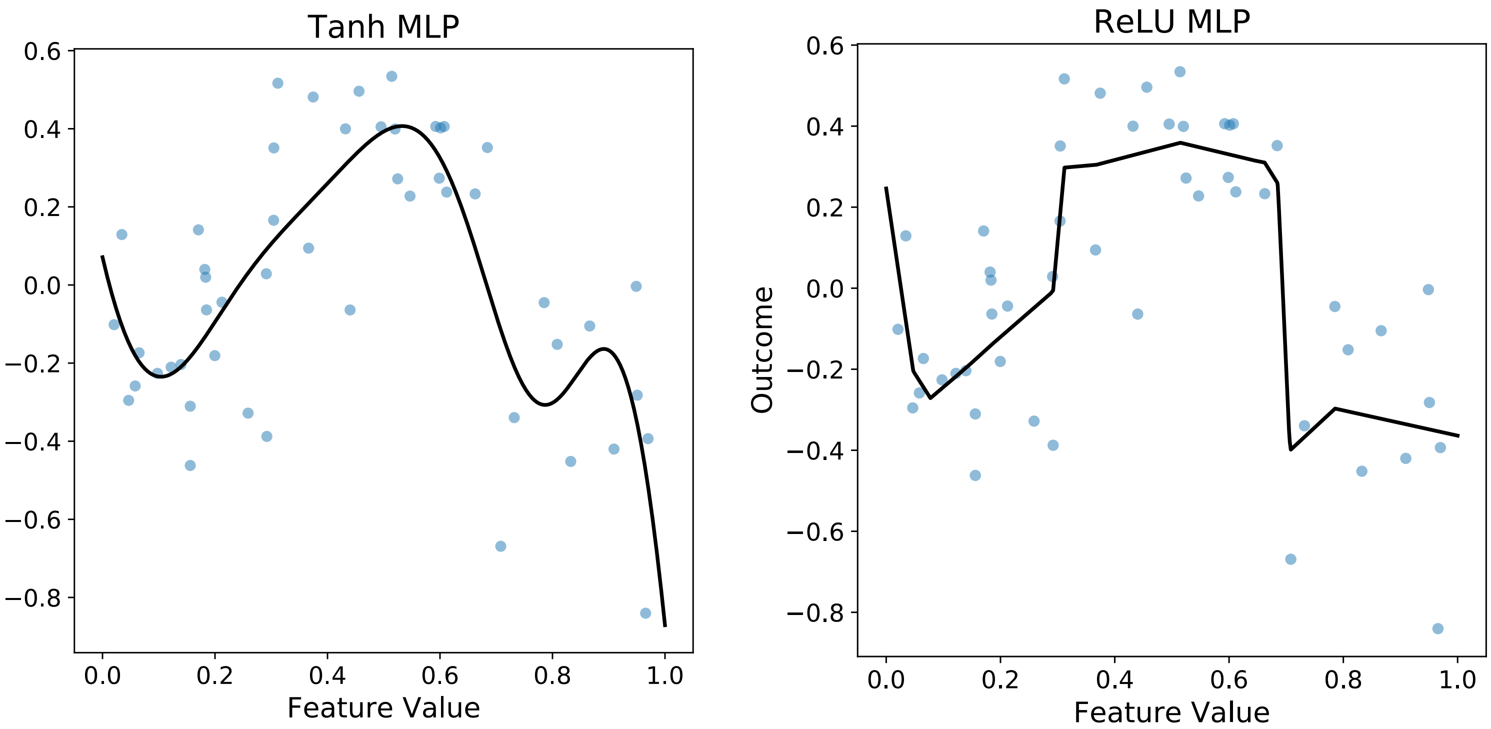
Leaky ReLU

$$\max(0.1x, x)$$

Removes vanishing gradients, but prefers that non-activated nodes be as "non-activated" as possible (doesn't make much sense)

(Mass et al., 2013; He et al., 2015)

Convergence Problems



Rudin and Carlson. The Secrets of Machine Learning: Ten Things You Wish You Had Known Earlier to be More Effective at Data Analysis. INFORMS TutORial, 2019.

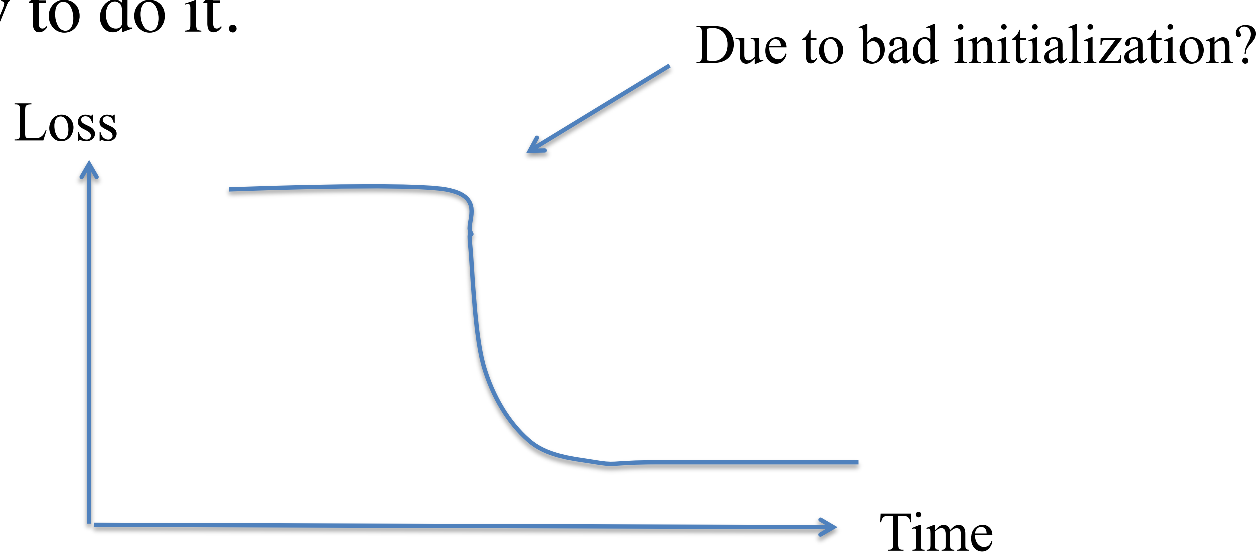
Convergence Problems

Adding momentum to gradients

- adjust gradient to make current gradient similar to previous gradients

Convergence Problems

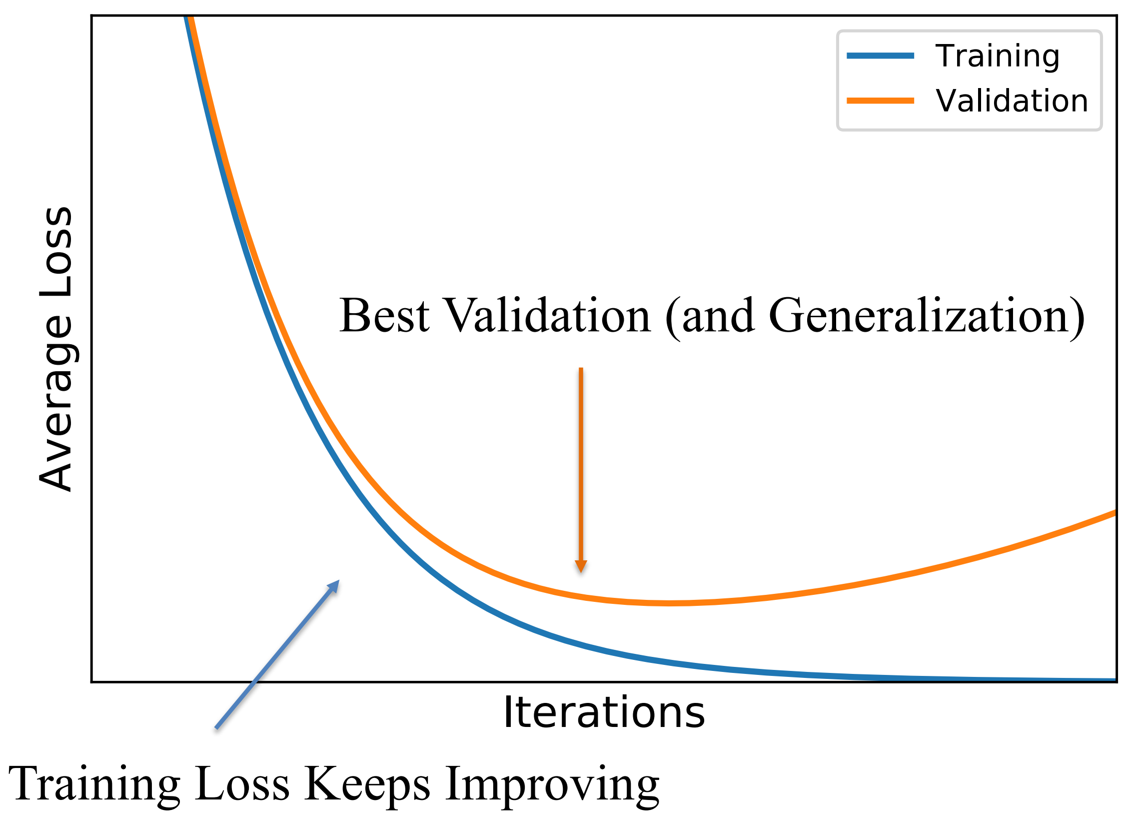
- Initialization of the networks weights is really important. I have no idea how to do it.



Convergence Problems

- Batch Normalization (Ioffe and Szegedy, 2015) is a step that:
  - Normalizes the outputs o_i of several nodes (a "mini-batch") in the same layer. (As usual, subtract the mean of the o_i's divide by their standard deviation).
  - Includes the mean and standard deviation as separate parameters to be learned.
  - Usually the normalization is before the nonlinear activation function.
  - This adds regularization and helps to prevent flat gradients in the network but sometimes it messes things up.

Early stopping via validation set



Rudin and Carlson. The Secrets of Machine Learning: Ten Things You Wish You Had Known Earlier to be More Effective at Data Analysis. INFORMS TutORial, 2019.

Convergence Problems Summary

- There are lots of convergence problems
- vanishing gradients
  - Adjust the learning rate
  - Change the activation function (tanh, ReLU, leaky ReLU, etc.)
  - Use Batch Norm
  - Add Momentum
- bad minima
  - Initialization (somehow…)
- overfitting
  - Stop early using validation set

Convergence Problems Summary

When training a NN, you "become" part of the algorithm because you control its convergence so heavily.

Convolutional neural networks and the intuition behind their architectures

Cynthia Rudin

Duke Machine Learning

Convolutional NN's

- Convolve means to slide the filter over all spatial locations and sum up the filter weights times the inputs.

Convolutional NN's

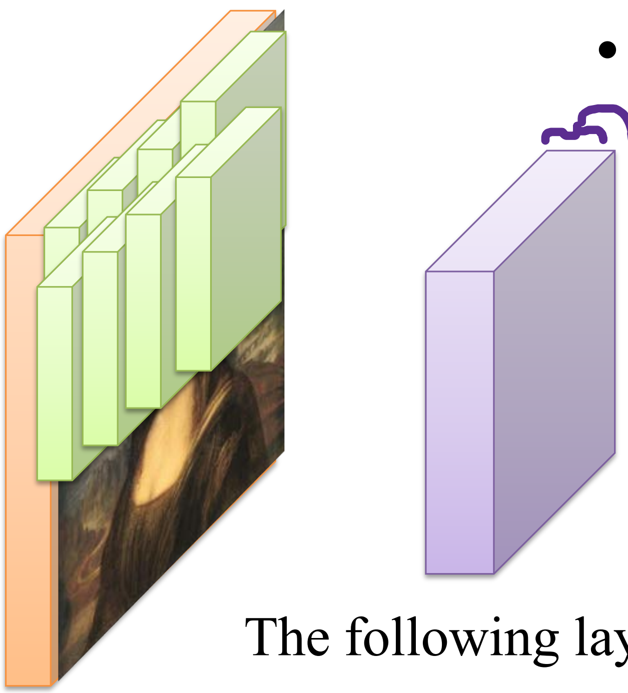
- Convolve means to slide the filter over all spatial locations and sum up the filter weights times the inputs.

$$\begin{matrix} -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{matrix}$$

- An edge filter will detect edges.

Convolutional NN’s

- Convolve means to slide the filter over all spatial locations and sum up the filter weights times the input.



- Stride of 5 means we step by 5’s when we convolve.

The thickness is the number of filters

The following layer is smaller by a factor of 5.

Convolutional NN's

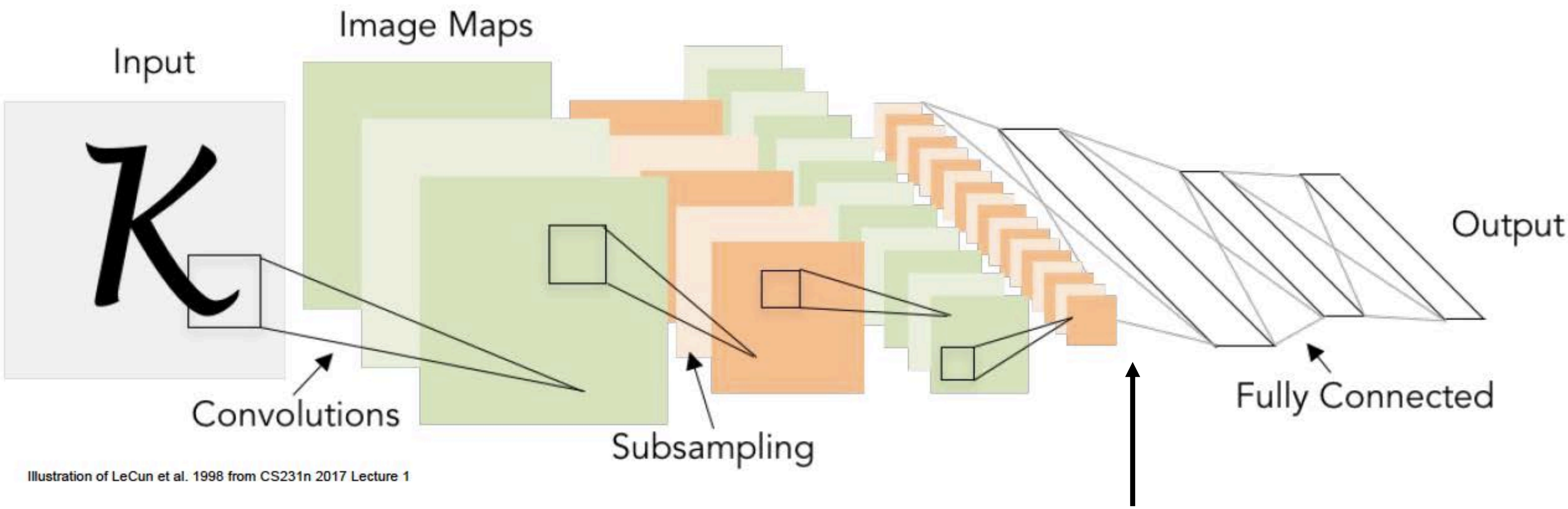
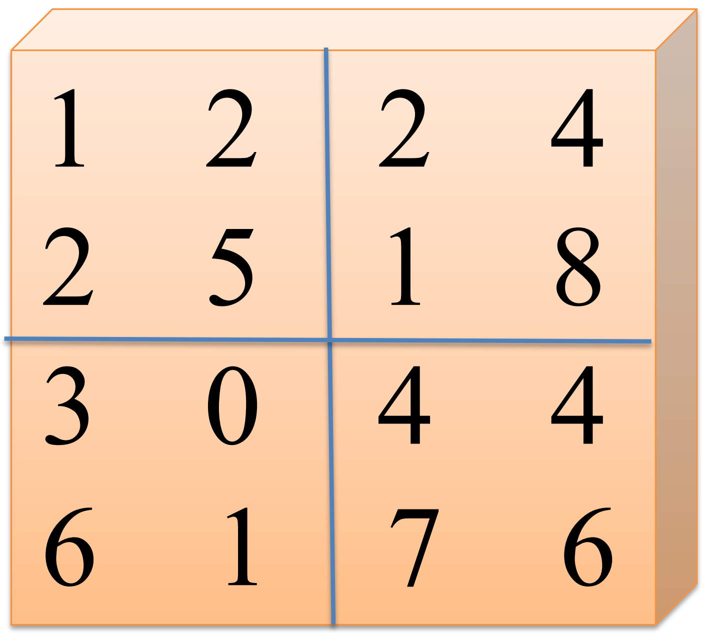


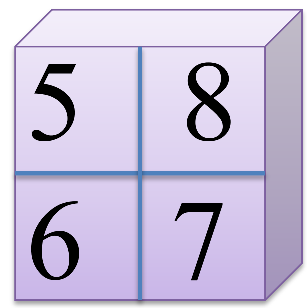
Image from LeCun et al 1998, reproduced in color from Li, Johnson, Yeung, 2017

Convolutional NN’s

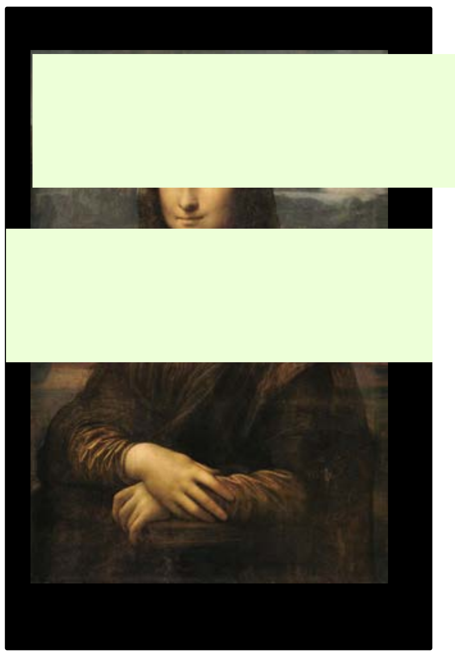
- Max pooling means to convolve with a max function.
- Intuitively keeps track of whether an earlier filter has detected something.



2 x 2 max pool filter and stride 2

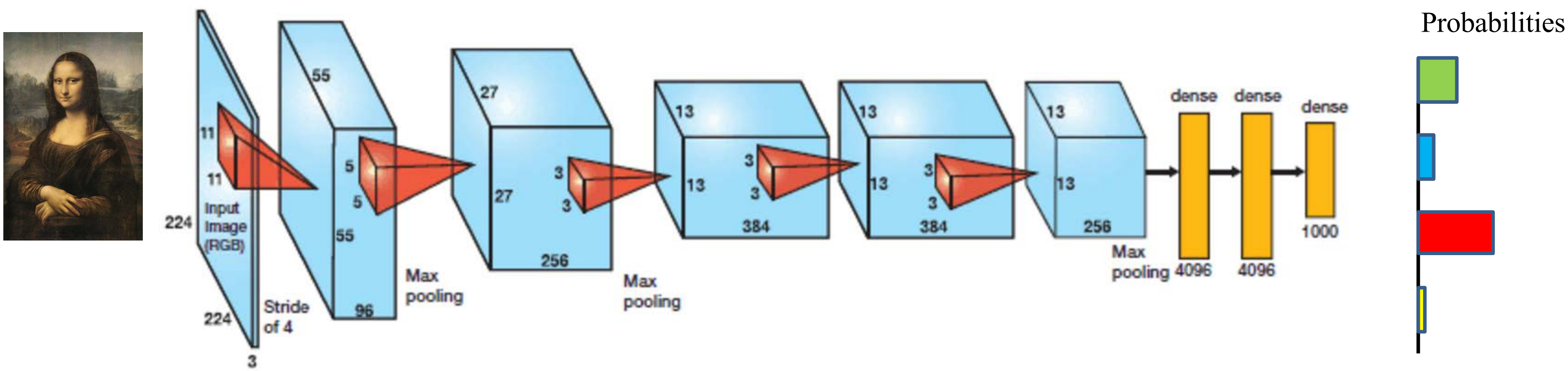


Zero-padding



- Add zeros around the image so that the dimensions work out.

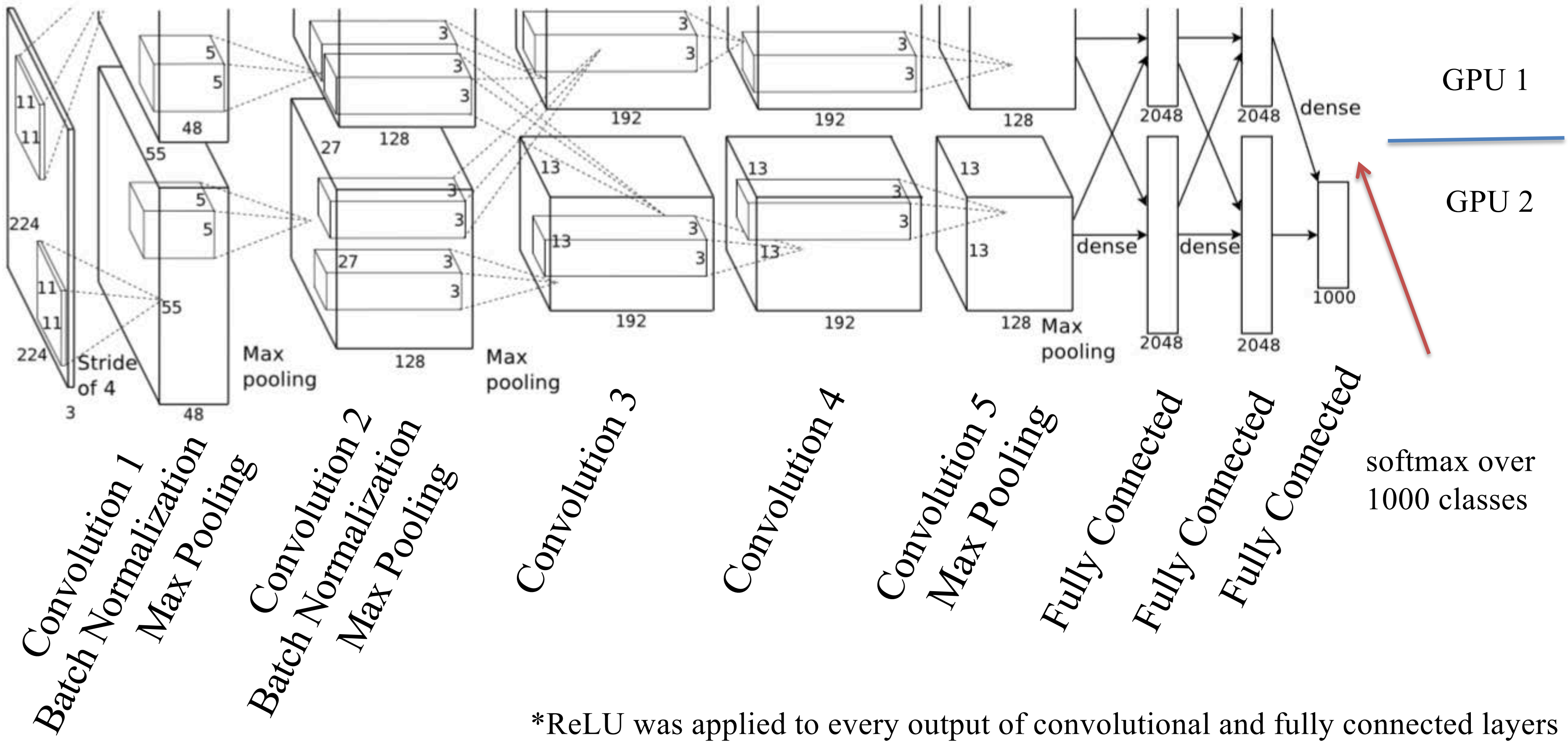
• AlexNet (Krizhevsky et al. 2012)



AlexNet won the ImageNet Large Scale Visual Recognition Challenge in 2012. It achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up.

Image source: unknown

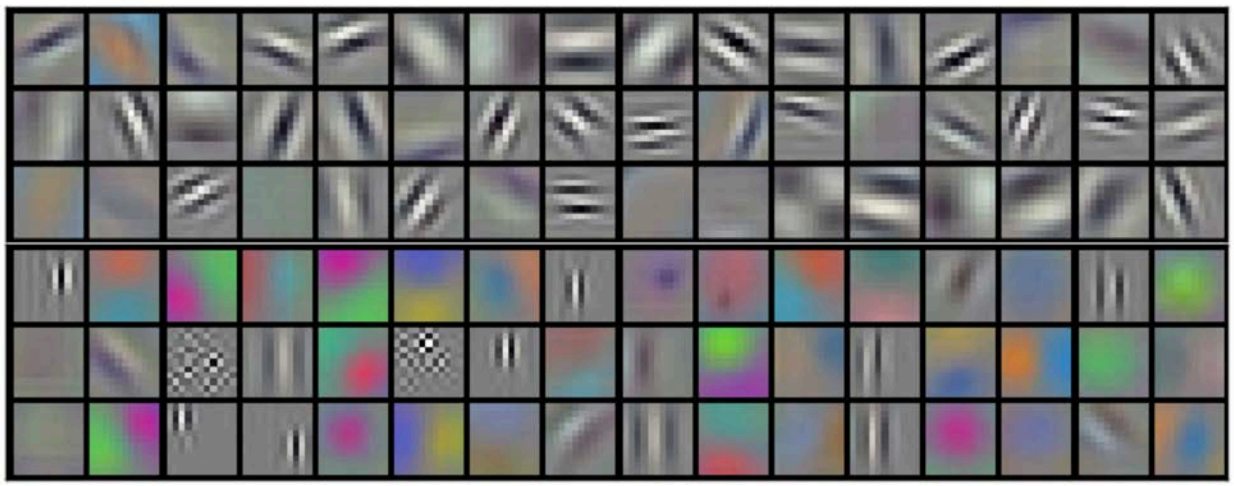
- AlexNet (Krizhevsky et al. 2012)



*ReLU was applied to every output of convolutional and fully connected layers

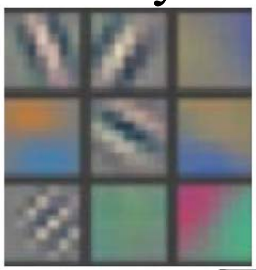
GPU 1

GPU 2

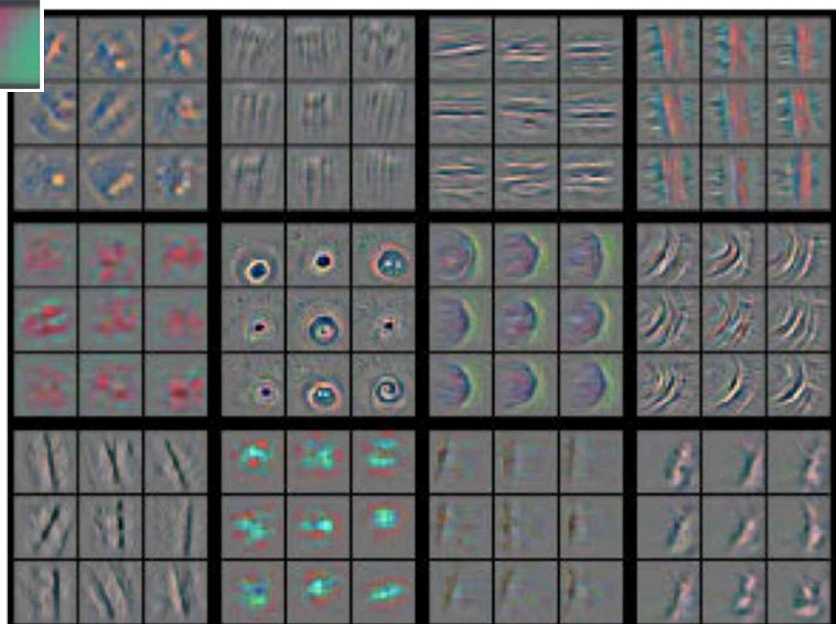


Layer 1 AlexNet filters (Krizhevsky et al. 2012)

from layer 1



from layer 2



from layer 5

Source: Zeiler and Fergus, 2013

Convolutional NN's

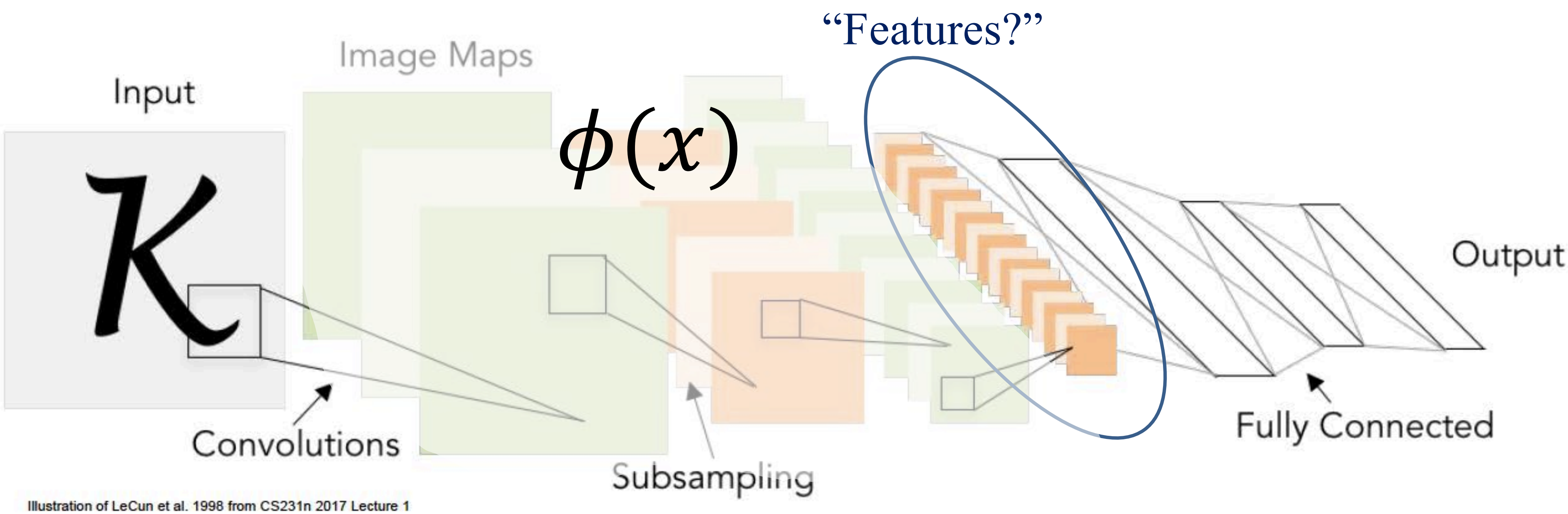
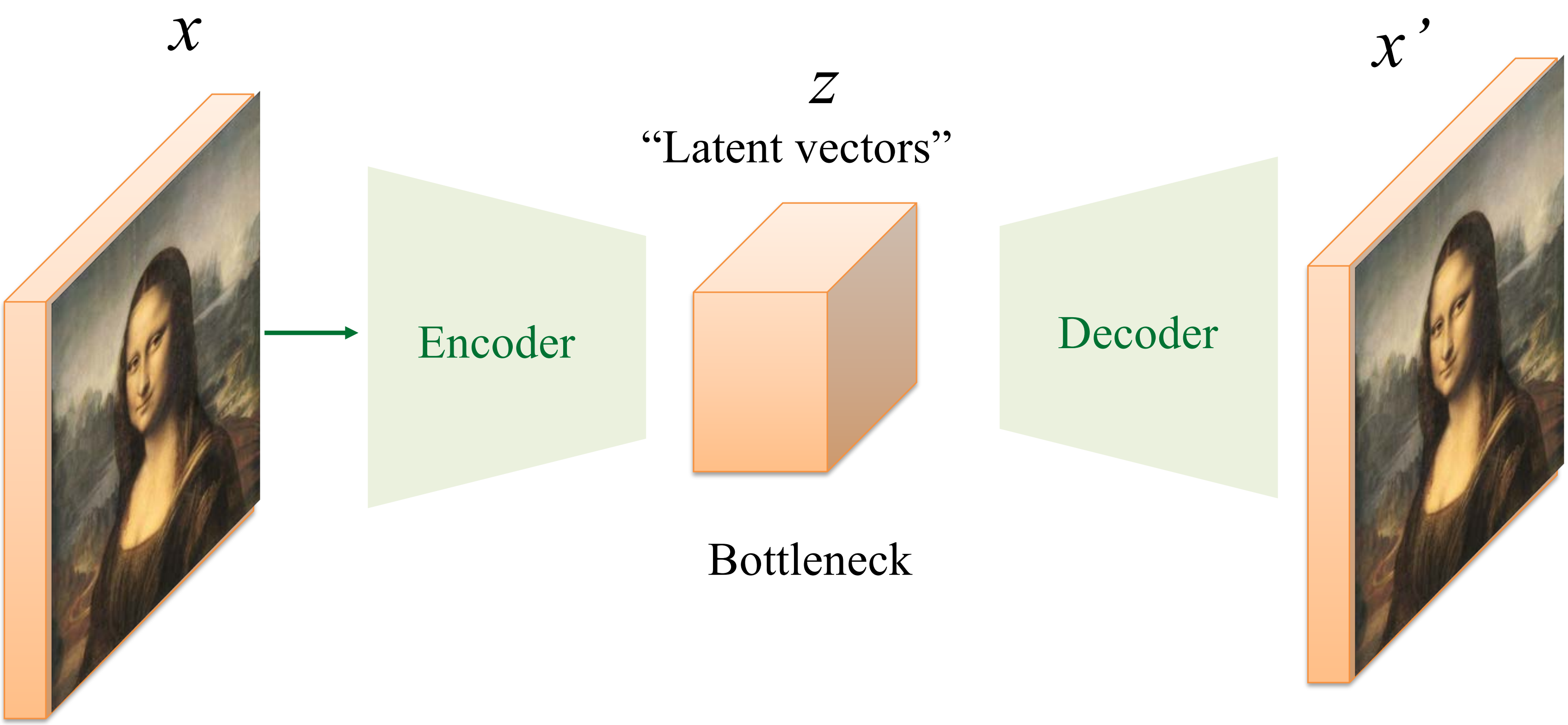


Image from LeCun et al 1998, reproduced also from Li, Johnson, Yeung, 2017

Autoencoders



There has been much work since AlexNet.

Next: Improving performance of CNNs for computer vision.

Improving Performance of Neural Networks

Cynthia Rudin

Duke Machine Learning

Data Augmentation

Chinese Lantern Festival, Cary NC, 2017

Data Augmentation

- include artificial data, such as horizontal flips, rotations, resized, cropped training images, change contrast and brightness, distortion, etc.

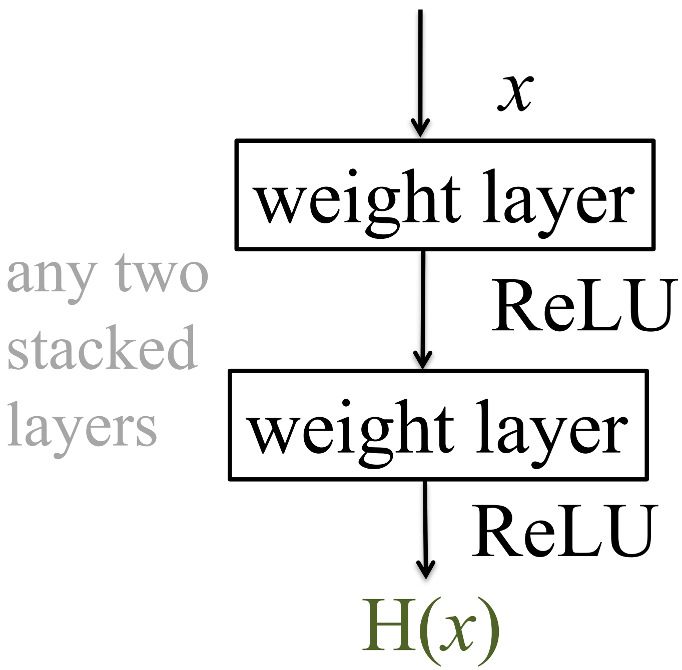
Chinese Lantern Festival, Cary NC, 2017

Data Augmentation

- include artificial data, such as horizontal flips, rotations, resized, cropped training images, change contrast and brightness, distortion, etc.

Chinese Lantern Festival, Cary NC, 2017

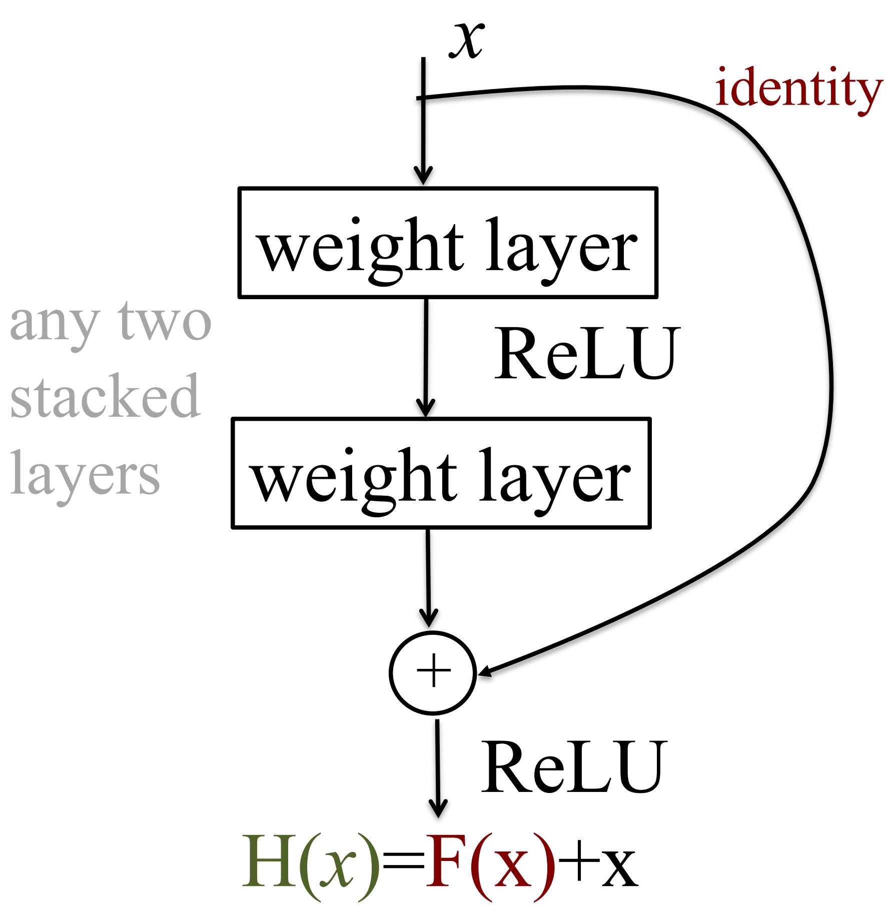
Residual Nets (He et al., 2016)



~~We hope to fit H(x).~~

Slides recreated from Kaiming He's tutorial
http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Residual Nets



~~We hope to fit $H(x)$.~~

We hope to fit $F(x)$.
We are now learning a residual of identity.

http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

Residual Nets

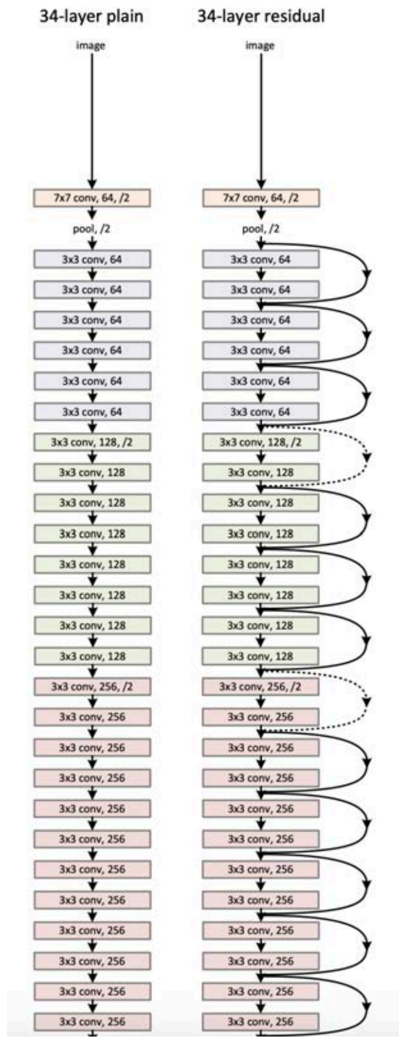
- By adding x, the derivative of the error with respect to x increases by 1. Thus, less vanishing derivatives.

- Allowed networks to go much deeper than before. "From 10 to 1000 layers"

$$H(x)=F(x)+x$$

http://kaiminghe.com/icml16tutorial/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

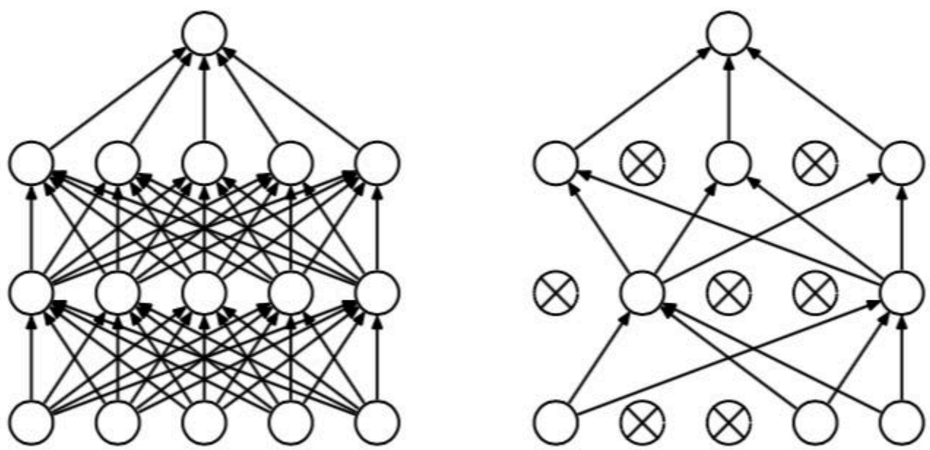
Residual Nets



He et al. Deep Residual Learning for Image Recognition, arXiV2015

Dropout (Srivastava et al., JMLR 2014)

- Forces signal to be “carried” throughout the network
- In each forward pass, for each neuron, with probability p, set all of its output weights to 0.
- p is a hyperparameter, usually p = 0.5.
- During testing, use all nodes.

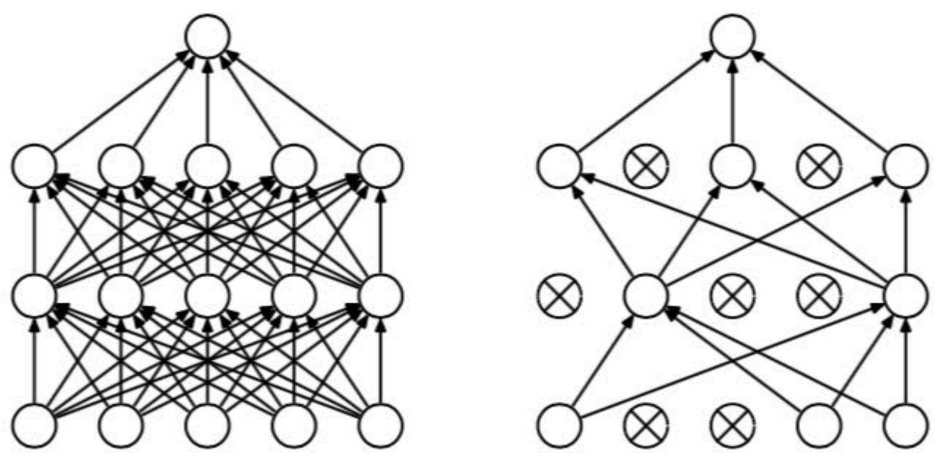


(a) Standard Neural Net (b) After applying dropout.

Image from Srivastava et al JMLR 2014)

Dropout (Srivastava et al., JMLR 2014)

- As if we are training exponentially many “sub” models. Similar idea to bagging (averaging many separately trained models together).
- Creates a redundant encoding.

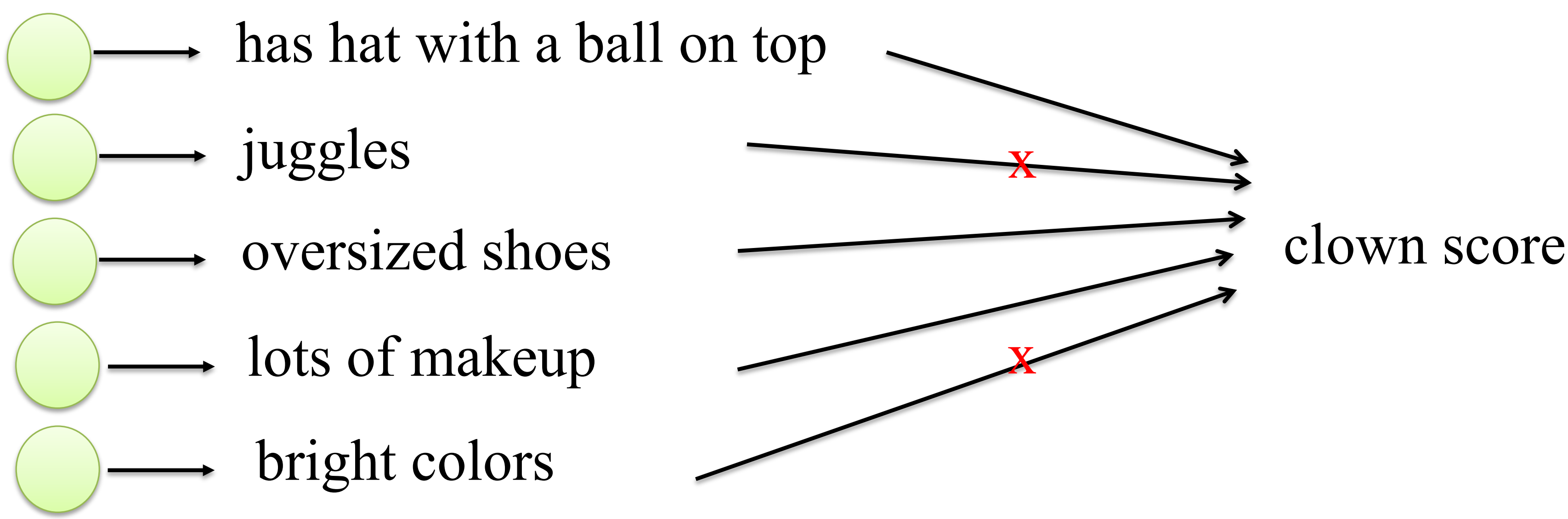


(a) Standard Neural Net (b) After applying dropout.

Image from Srivastava et al JMLR 2014)

Dropout (Srivastava et al., JMLR 2014)

- As if we are training exponentially many “sub” models. Similar idea to bagging (averaging many separately trained models together).
- Creates a redundant encoding.



“Transfer” Learning

- Using information about the solution to one problem to help solve another.
- Use the early layers from a pretrained model in another network. Retrain only the weights from the last few layers.

Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS
FOR LARGE-SCALE IMAGE RECOGNITION

VGG

Karen Simonyan* & Andrew Zisserman+
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

A Big Bag of Tricks

- Dropout
- Batch Normalization
- Data Augmentation
- Residual Networks
- Activation Functions (ReLU, Leaky ReLU)
- Initialization
- Transfer Learning
- :

Other ways to improve neural networks

- Change the dataset. Use fine-grained labels

Is there a fence in this picture?

- Understand the model so you know what’s wrong with it.

Warnings about Neural Networks for Computer Vision

Cynthia Rudin

Duke Machine Learning

CNNs can use the wrong information (confounding)

CNNs can use the wrong information (confounding)

Source: Wikimedia commons, West German soldiers in 1983

Ok, well, that was a bad dataset…

CNNs can use the wrong information (confounding)

npj Digital Medicine

NPJ Digit Med. 2019; 2: 31.
Published online 2019 Apr 30. doi: 10.1038/s41746-019-0105-1
PMCID: PMC6550136
PMID: 31304378

Deep learning predicts hip fracture using confounding patient and healthcare variables

Marcus A. Badgeley,[1,2,3] John R. Zech,[4] Luke Oakden-Rayner,[5] Benjamin S. Glicksberg,[6] Manway Liu,[1] William Gale,[7] Michael V. McConnell,[1,8] Bethany Percha,[2] Thomas M. Snyder,[1] and Joel T. Dudley[2,3]

▸ Author information ▸ Article notes ▸ Copyright and License information Disclaimer

Solution to this? Interpretability? Heavy testing? Massive data augmentation?

Deep fakes are dangerous

Deep fakes are dangerous

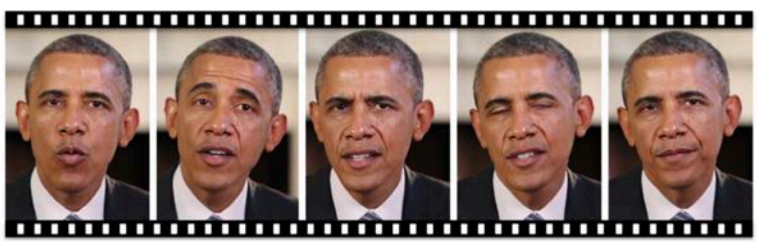
UW NEWS

ENGINEERING | NEWS RELEASES | RESEARCH | SCIENCE | TECHNOLOGY

July 11, 2017

Lip-syncing Obama: New tools turn audio clips into realistic video

Jennifer Langston
UW News



University of Washington researchers have developed new algorithms that solve a thorny challenge in the field of computer vision: turning audio clips into a realistic, lip-synced video of the person speaking those words.

Subscribe | Latest Issues | SCIENTIFIC AMERICAN 175 | Cart 0 | Sign In | Stay Inf

CORONAVIRUS THE SCIENCES MIND HEALTH TECH SUSTAINABILITY VIDEO PODCASTS OPINION PUBLICATIO

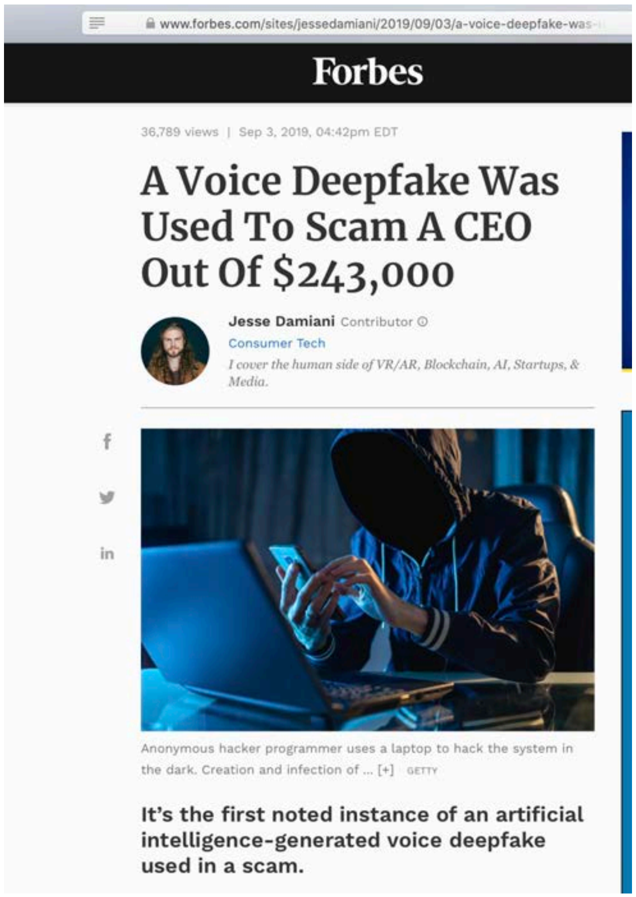
Celebrating 175 Years of Discovery Learn More

POLICY & ETHICS | OPINION

Deepfakes and the New AI-Generated Fake Media Creation-Detection Arms Race

Manipulated videos are getting more sophisticated all the time—but so are the techniques that can identify them

Deep fakes are dangerous


www.forbes.com/sites/jessedamiani/2019/09/03/a-voice-deepfake-was-

Forbes

36,789 views | Sep 3, 2019, 04:42pm EDT

A Voice Deepfake Was Used To Scam A CEO Out Of $243,000

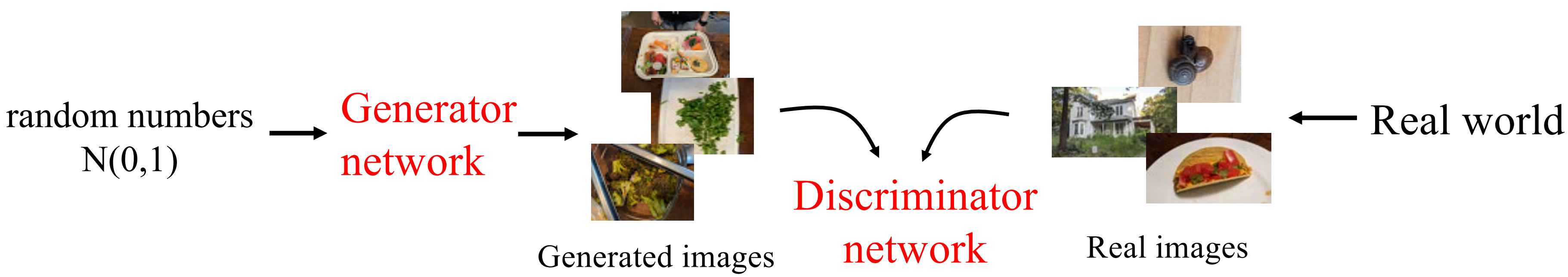
Jesse Damiani Contributor
Consumer Tech
I cover the human side of VR/AR, Blockchain, AI, Startups, & Media.

Anonymous hacker programmer uses a laptop to hack the system in the dark. Creation and infection of ... [+] GETTY

It's the first noted instance of an artificial intelligence-generated voice deepfake used in a scam.

GANS – Generative Adversarial Networks

- GANS are actor-critic models
- They produce realistic-looking images/data
- Used commonly for AI artwork / deep fakes



If the generator creates images that the discriminator can't tell apart, it's good.
(The "arms race" is between the generators and the discriminators.)

(Goodfellow et al 2014)

GANS – Generative Adversarial Networks

From Goodfellow et al 2014:

D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Discriminator maximizes
likelihood of real data

Discriminator minimizes
likelihood of generated data.

Generator maximizes
likelihood of generated data

GANS – Generative Adversarial Networks

From Goodfellow et al 2014:

D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Discriminator maximizes likelihood of real data

Discriminator minimizes likelihood of generated data.

Generator aims to make discriminator not work well.

Generator maximizes likelihood of generated data

GANS – Generative Adversarial Networks

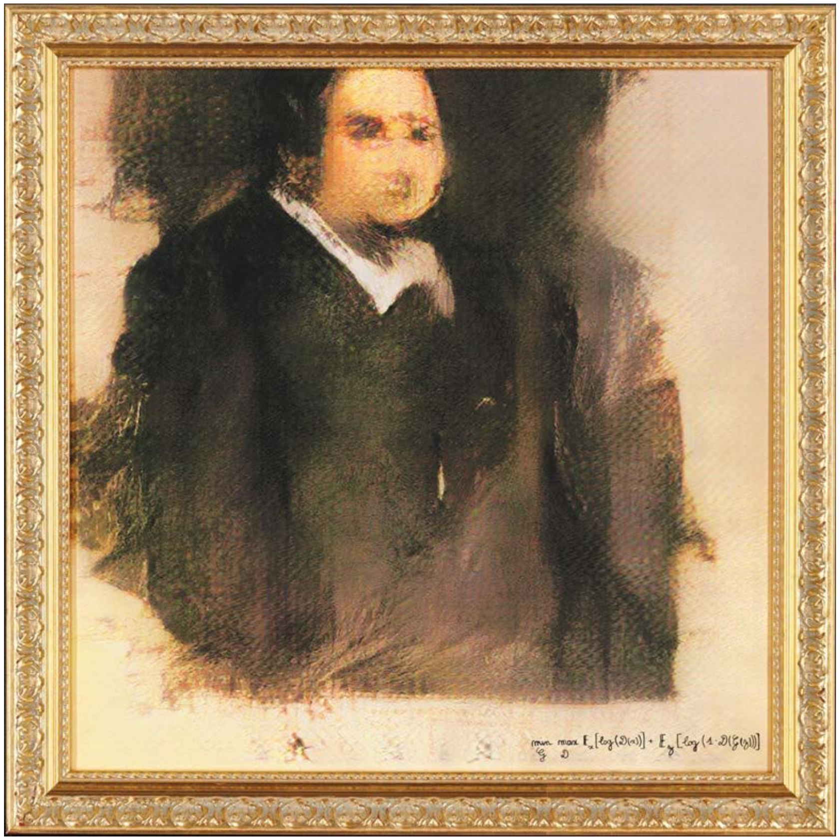
From Goodfellow et al 2014:

D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\max_G \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(\cancel{1 -} D(G(\boldsymbol{z})))].$$

Gradient ascent steps on discriminator
Gradient descent steps on generator



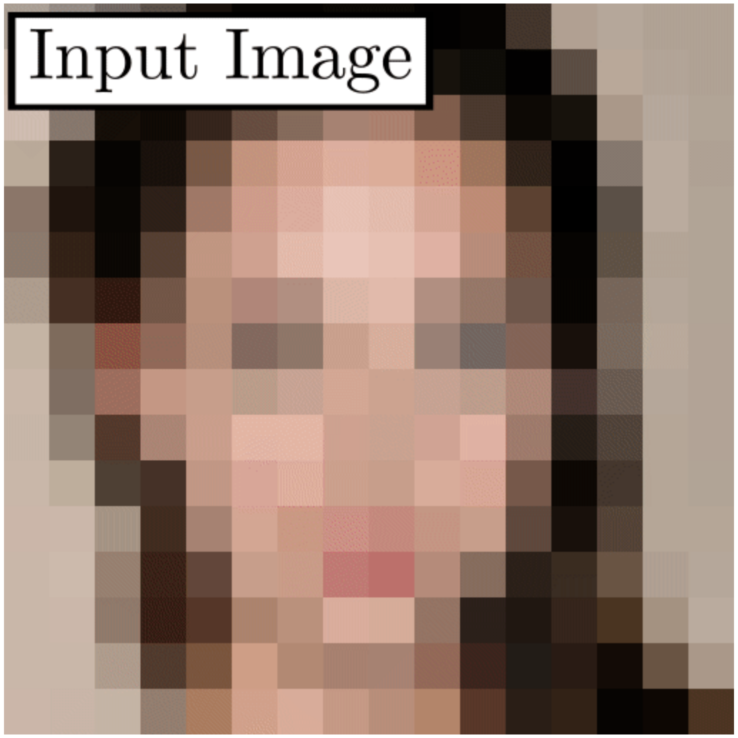
GANs are totally useful for artwork!

Is artificial intelligence set to become art's next medium?

AI artwork sells for $432,500 — nearly 45 times its high estimate — as Christie's becomes the first auction house to offer a work of art created by an algorithm

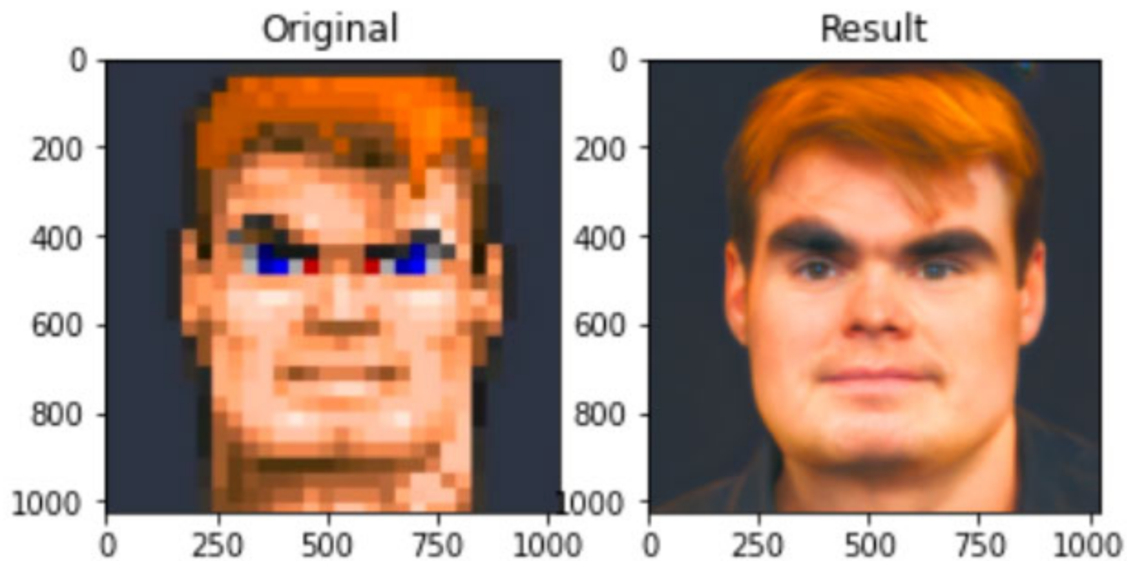
Figure adapted from L. Gatys et al. "A Neural Algorithm of Artistic Style" (2015) by Google AI Blog

GANs are totally useful for artwork!



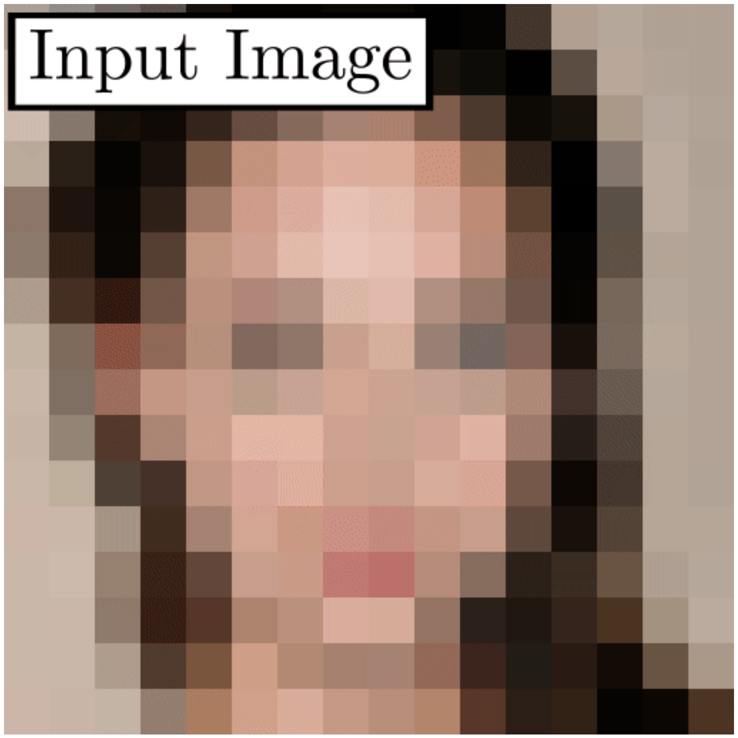
Menon et al. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models, CVPR 2020

Tero Karras et al. A style-based generator architecture for generative adversarial networks. CVPR, 2019.



A twitter user's result from the PULSE algorithm, which uses StyleGAN

GANs are totally useful for artwork!



Menon et al. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models, CVPR 2020

Tero Karras et al. A style-based generator architecture for generative adversarial networks. CVPR, 2019.

PULSE shows us that there is often no hope of identifying someone in a grainy security video.

There could be many high res images corresponding to one low res image.

GANs are totally useful for artwork!

Neural networks can be brittle

- Adversarial attacks show that changing a *single pixel* in an image can change the predicted class in modern ML systems.
- It is easy to fool a computer vision system.

$\overset{?}{=}$

Need better data augmentation…

Eykholt et al., 2018 **Robust Physical-World Attacks on Deep Learning Models,**

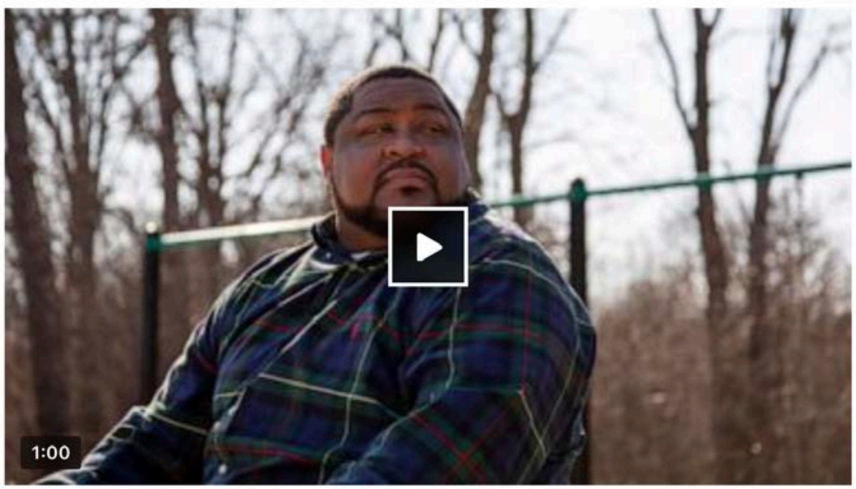
The model will not always be used in the way it is intended

abc NEWS VIDEO LIVE SHOWS 2020 ELECTIONS CORONAVIRUS

Black man wrongfully arrested because of incorrect facial recognition

Robert Williams spent nearly 30 hours in a detention center.

By Ella Torres
June 25, 2020, 2:01 PM • 6 min read



So…

- Much care is needed in many applications of neural networks.
  - – medical image processing (confounding)
  - – automated driving systems (not robust, not perfect)
  - – facial recognition (not perfect, watch for bias)
  - – deep fakes (easily fraudulent)
- Neural networks are great for artwork.