UVA CS 6316 / CS 4501-004
Machine Learning
Fall 2016

Lecture 17:
Neural Networks and Deep Learning

Jack Lanchantin
Dr. Yanjun Qi

UNIVERSITY of VIRGINIA

Neurons

1-Layer Neural Network

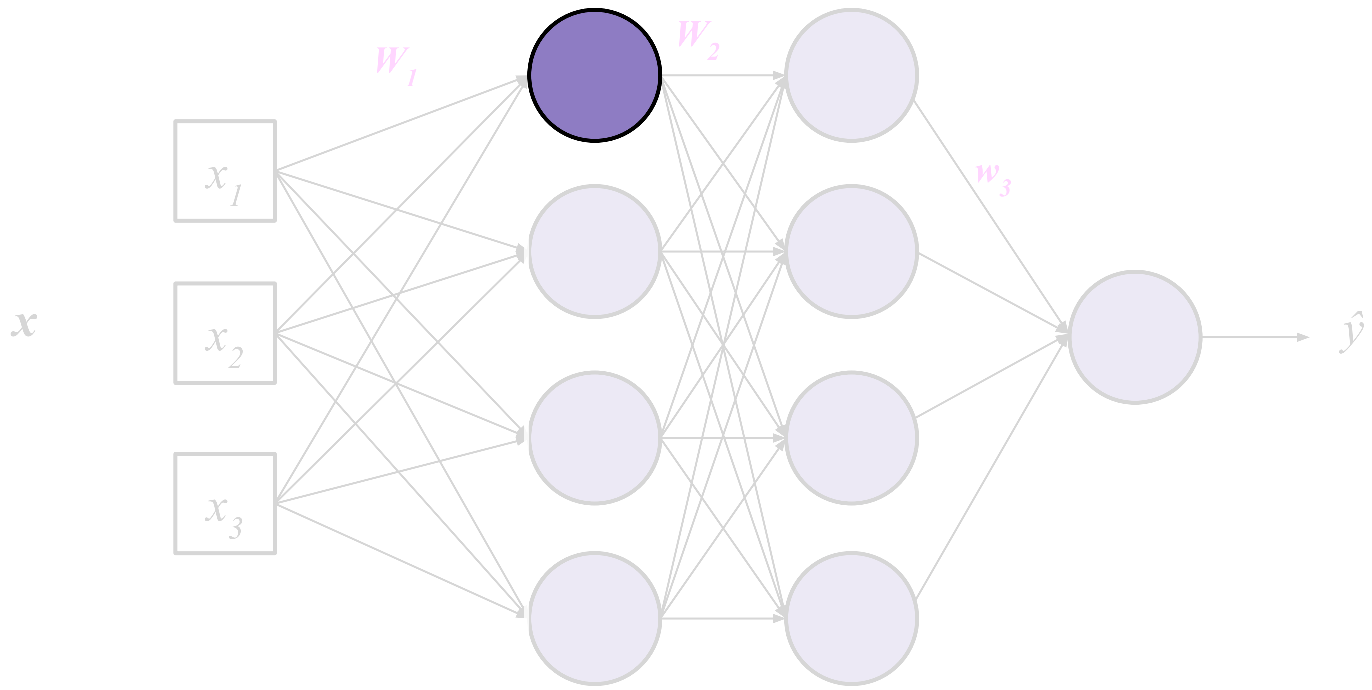
Multi-layer Neural Network

Loss Functions

Backpropagation

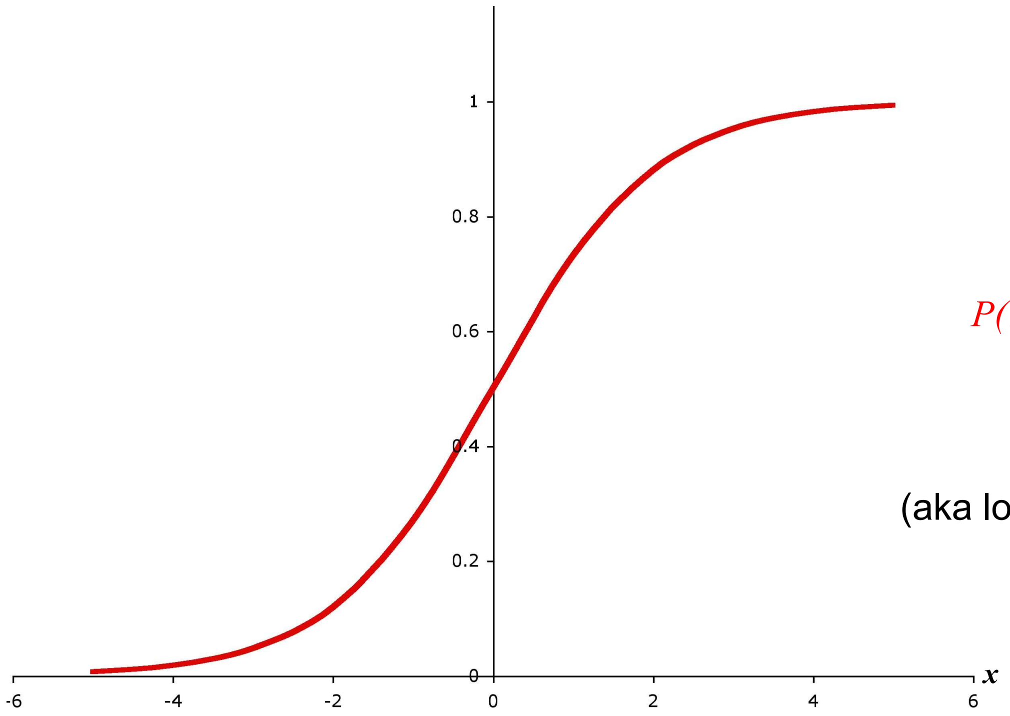
Nonlinearity Functions

NNs in Practice



$\boldsymbol{x}$
x_1
x_2
x_3
W_1
W_2
w_3
$\hat{y}$

Logistic Regression

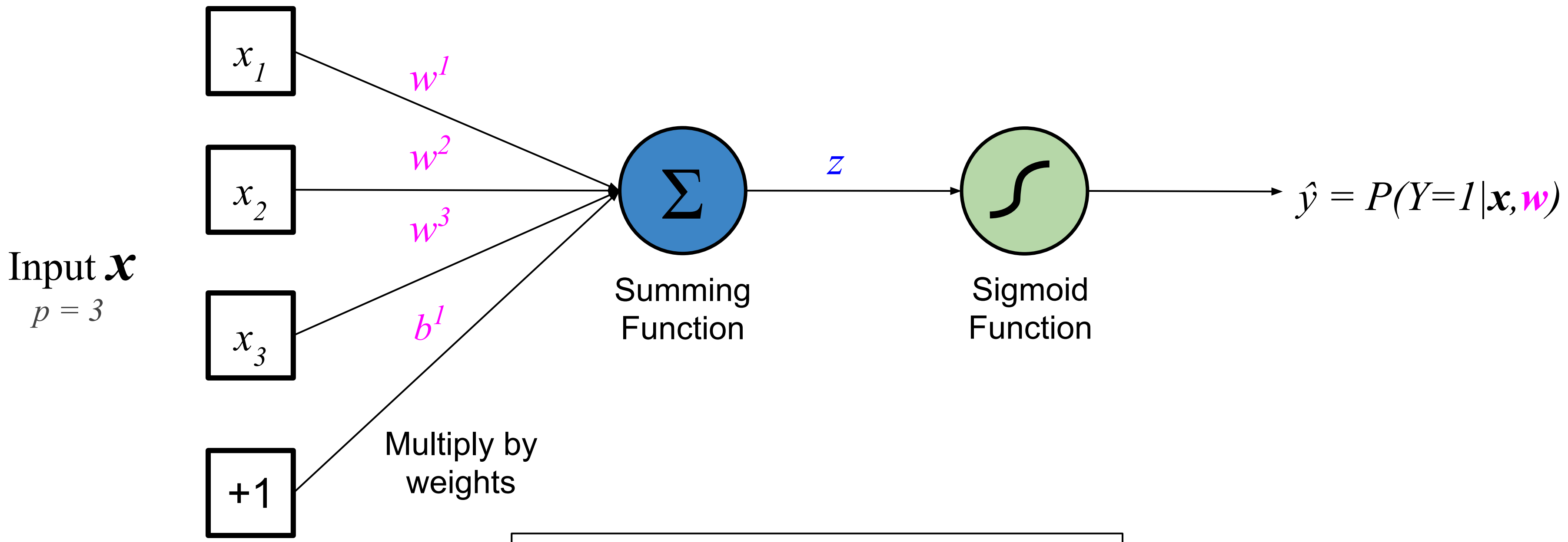


$$P(Y=1|\boldsymbol{x}) = \frac{e^{\boldsymbol{wx}+b}}{1+e^{\boldsymbol{wx}+b}}$$

Sigmoid Function

(aka logistic, logit, “S”, soft-step)

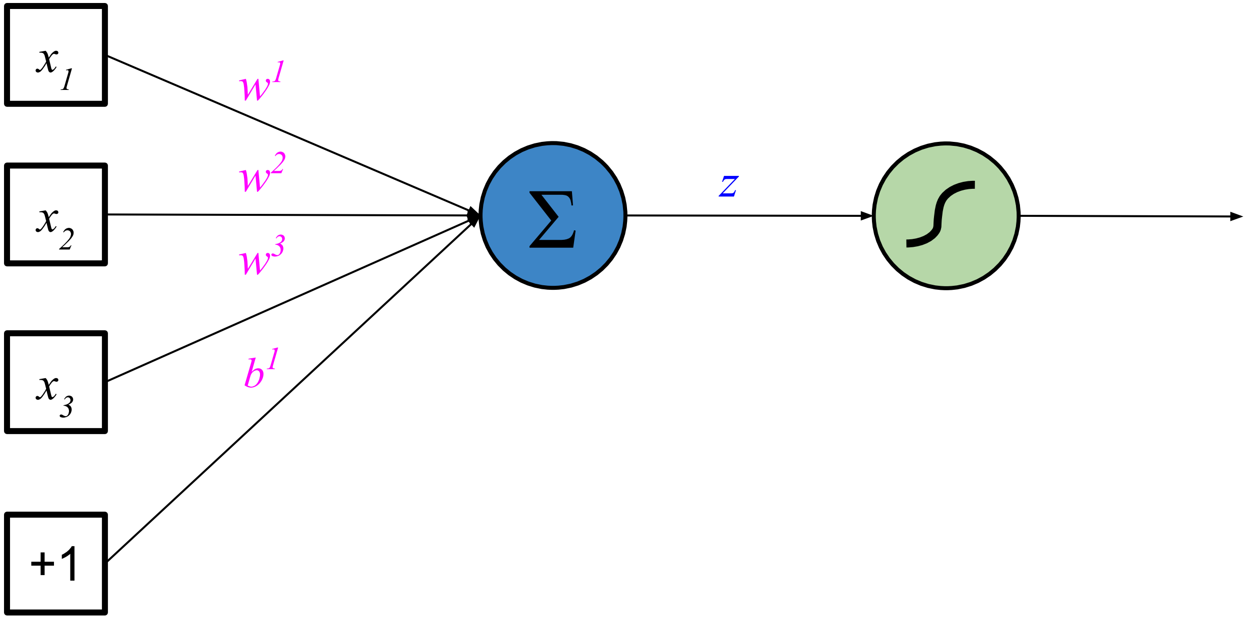
Expanded Logistic Regression



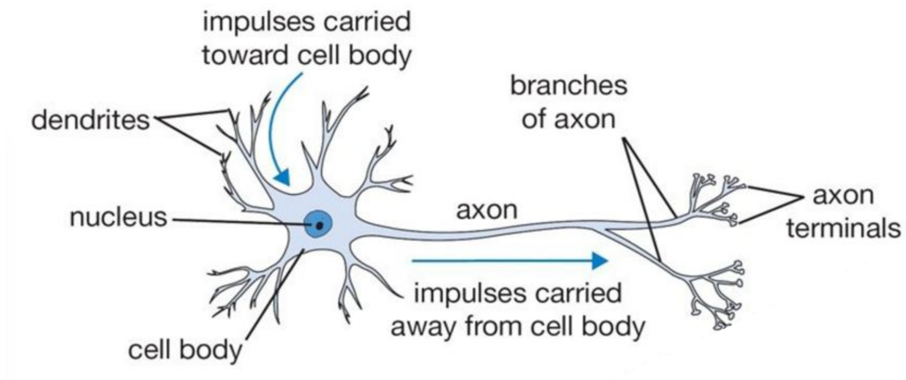
$$\underset{1\times 1}{z} = \underset{1\times p}{\boldsymbol{w}^T} \cdot \underset{p\times 1}{\boldsymbol{x}} + \boldsymbol{b}$$

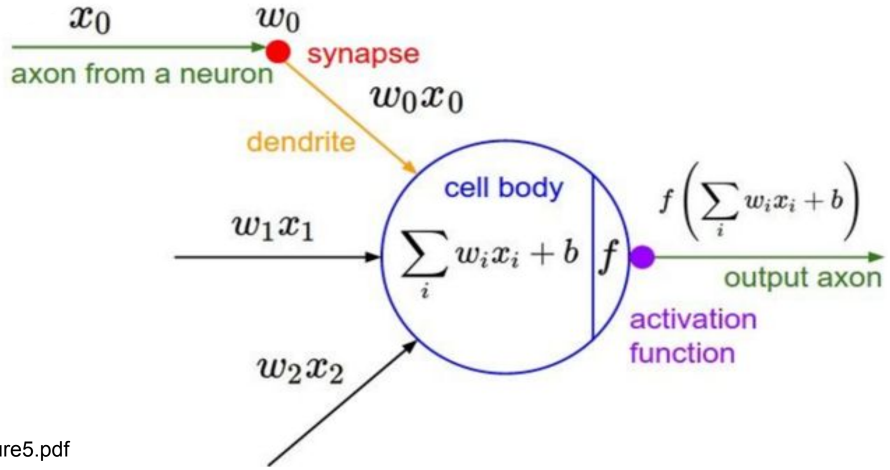
$$y = sigmoid(z) = \frac{e^z}{1 + e^z}$$

“Neuron”



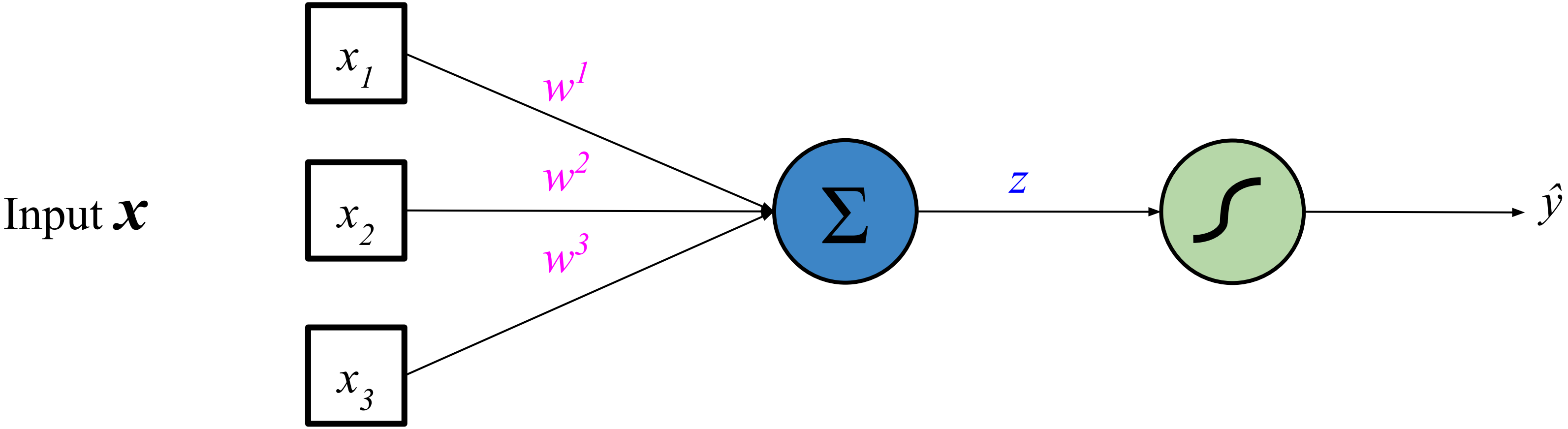
Neurons





http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Neuron

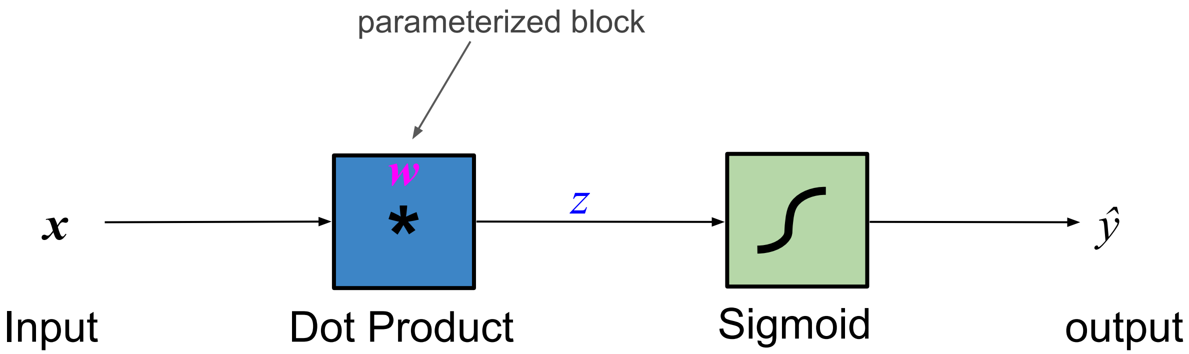


From here on, we leave out bias for simplicity

$$\underset{1\times 1}{z} = \underset{1\times p}{\boldsymbol{w}^T} \cdot \underset{p\times 1}{\boldsymbol{x}}$$

$$\hat{y} = sigmoid(z) = \frac{e^z}{1 + e^z}$$

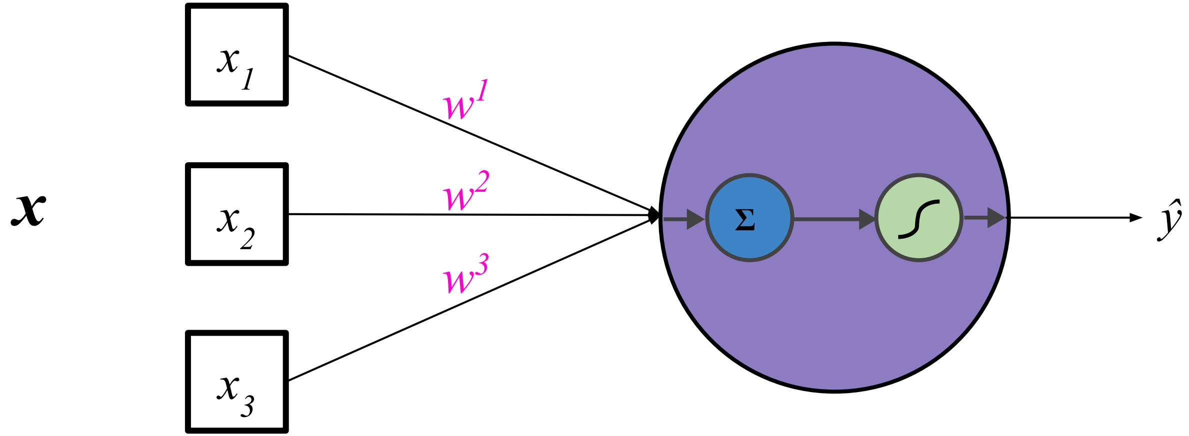
"Block View" of a Neuron



$$\underset{1\times 1}{z} = \underset{1\times p}{\boldsymbol{w}^T} \cdot \underset{p\times 1}{\boldsymbol{x}}$$

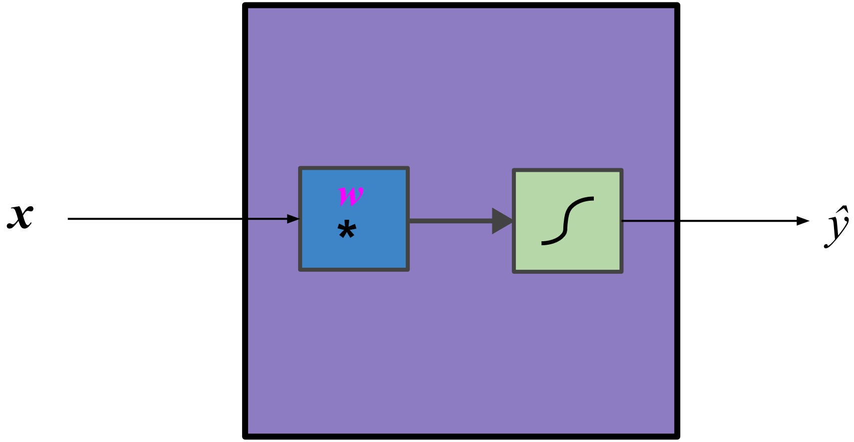
$$\hat{y} = sigmoid(z) = \frac{e^z}{1 + e^z}$$

Neuron Representation



The linear transformation and nonlinearity together is typically considered a single neuron

Neuron Representation



The linear transformation and nonlinearity together is typically considered a single neuron

Neurons

1-Layer Neural Network

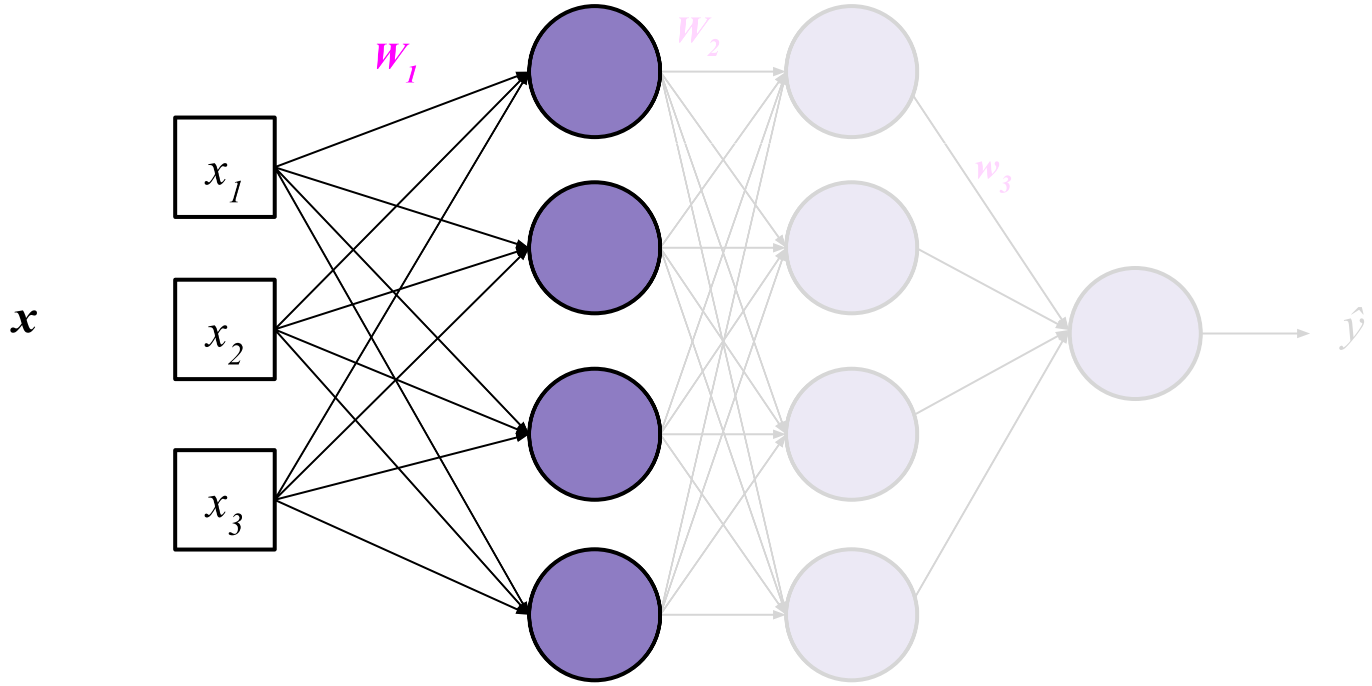
Multi-layer Neural Network

Loss Functions

Backpropagation

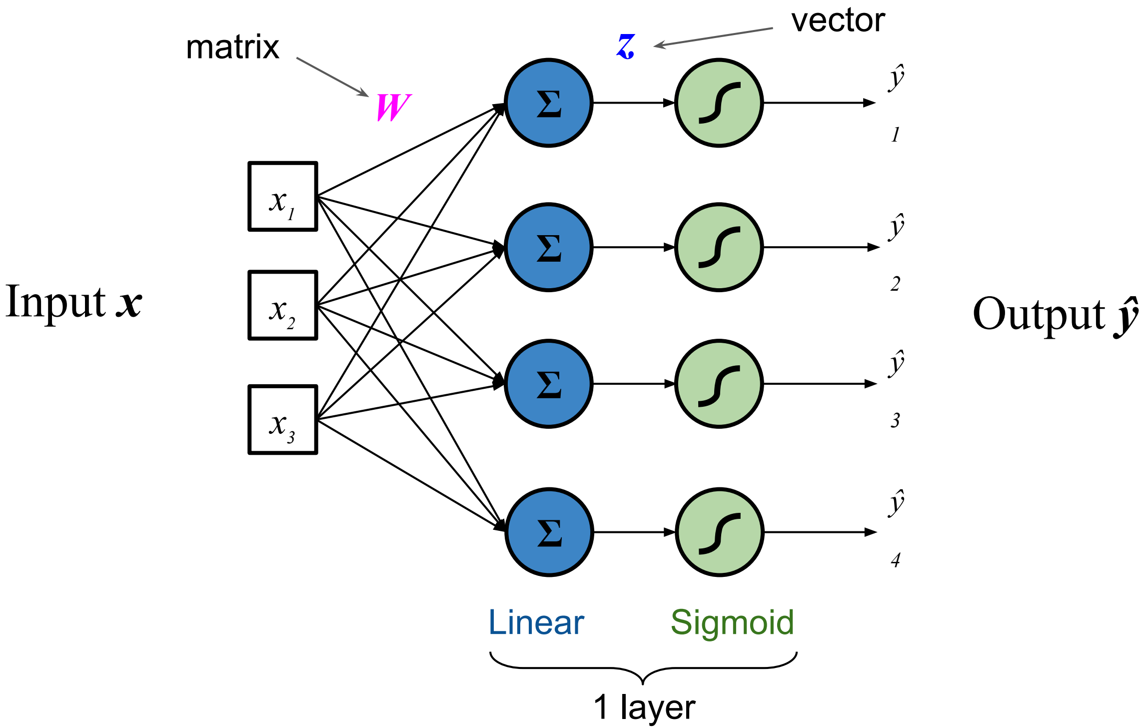
Nonlinearity Functions

NNs in Practice

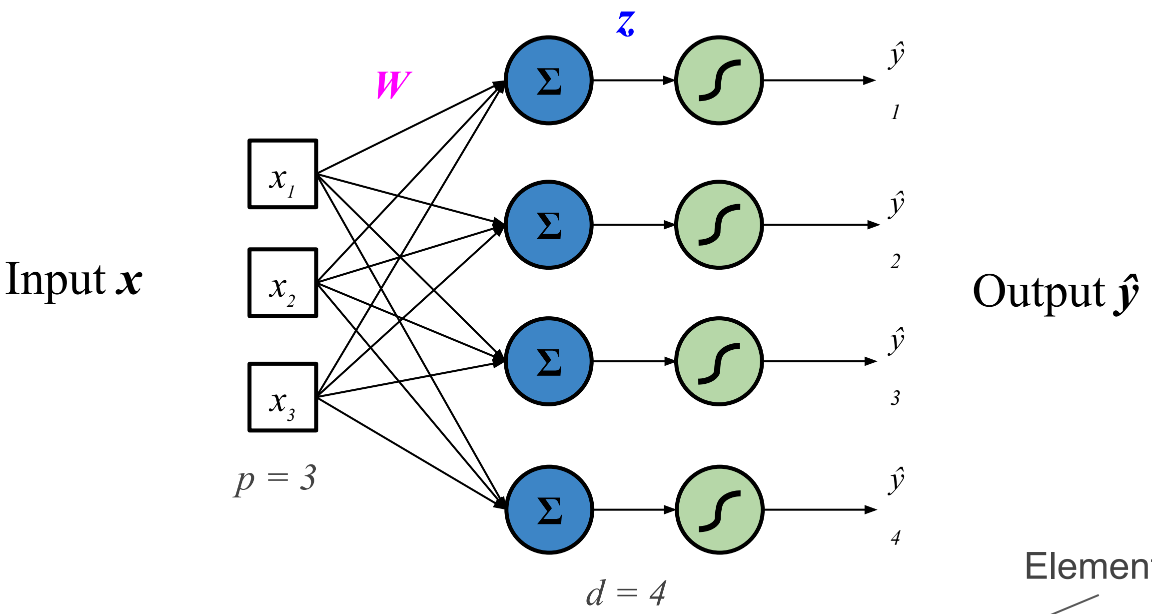


x
x1
x2
x3
W1
W2
w3
ŷ

1-Layer Neural Network (with 4 neurons)



1-Layer Neural Network (with 4 neurons)

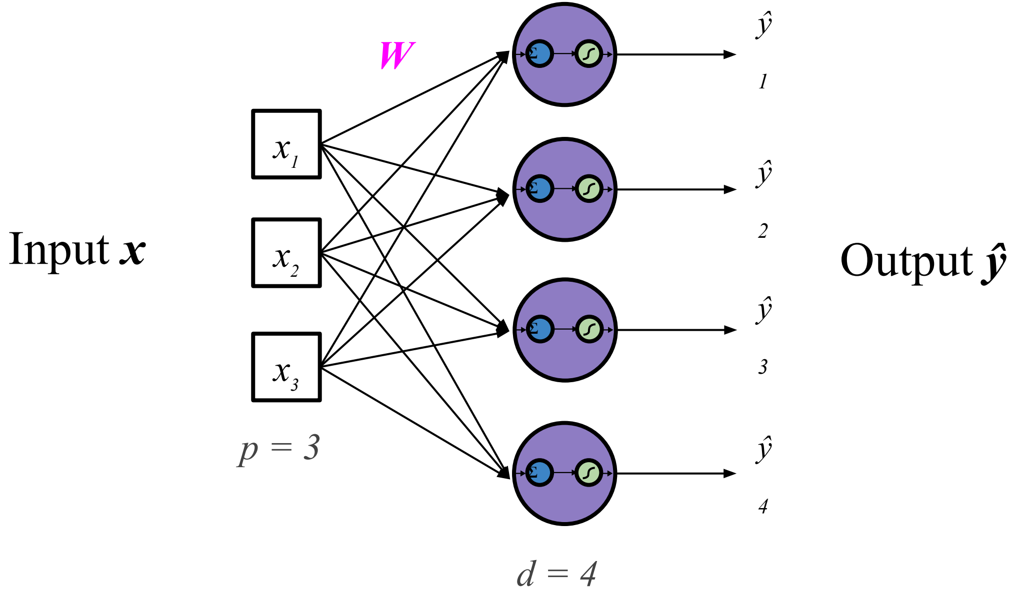


Element-wise on vector z

$$\underset{d\times 1}{\boldsymbol{z}} = \underset{d\times p}{W^T} \underset{p\times 1}{\boldsymbol{x}}$$

$$\underset{d\times 1}{\hat{\boldsymbol{y}}} = sigmoid(\underset{d\times 1}{\boldsymbol{z}}) = \frac{e^{z}}{1 + e^{z}}$$

1-Layer Neural Network (with 4 neurons)

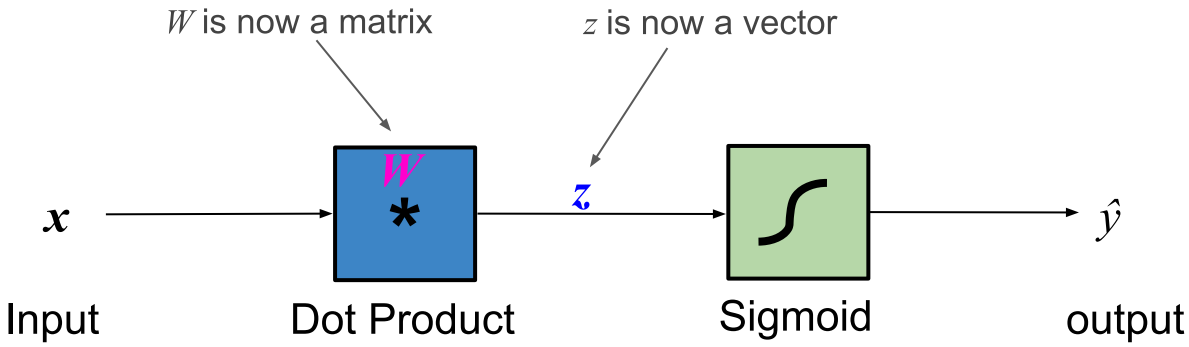


Element-wise on vector z

$$\underset{d\text{x}1}{\boldsymbol{z}} = \underset{d\text{x}p}{\boldsymbol{W}^T} \underset{p\text{x}1}{\boldsymbol{x}}$$

$$\underset{d\text{x}1}{\hat{\boldsymbol{y}}} = sigmoid(\underset{d\text{x}1}{\boldsymbol{z}}) = \frac{e^z}{1 + e^z}$$

“Block View” of a Neural Network



$$\underset{d\text{x}1}{\boldsymbol{z}} = \underset{d\text{x}p}{\boldsymbol{W}^T} \underset{p\text{x}1}{\boldsymbol{x}}$$

$$\underset{d\text{x}1}{\hat{\boldsymbol{y}}} = sigmoid(\underset{d\text{x}1}{\boldsymbol{z}}) = \frac{e^z}{1 + e^z}$$

Neurons

1-Layer Neural Network

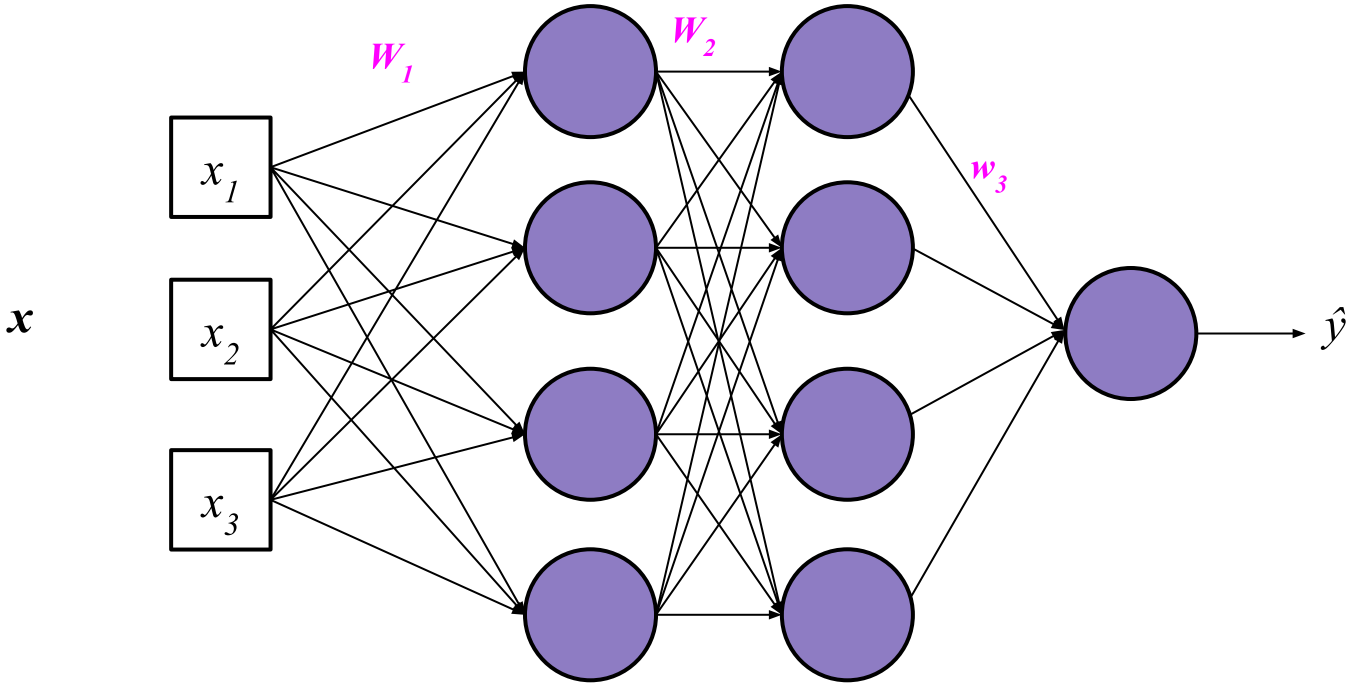
Multi-layer Neural Network

Loss Functions

Backpropagation

Nonlinearity Functions

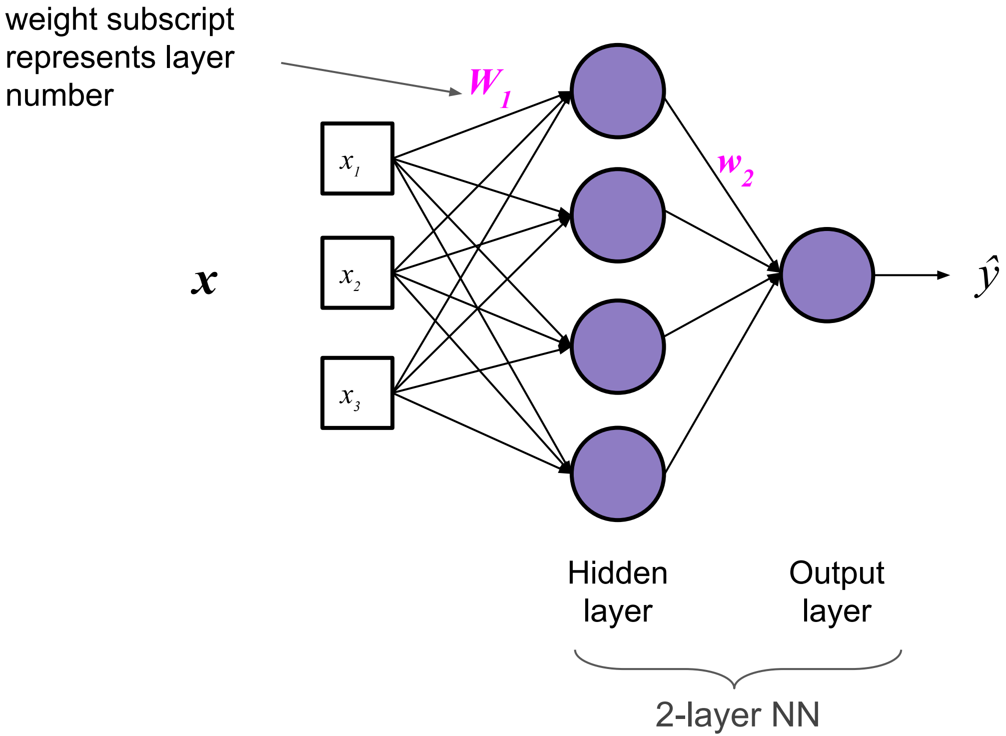
NNs in Practice



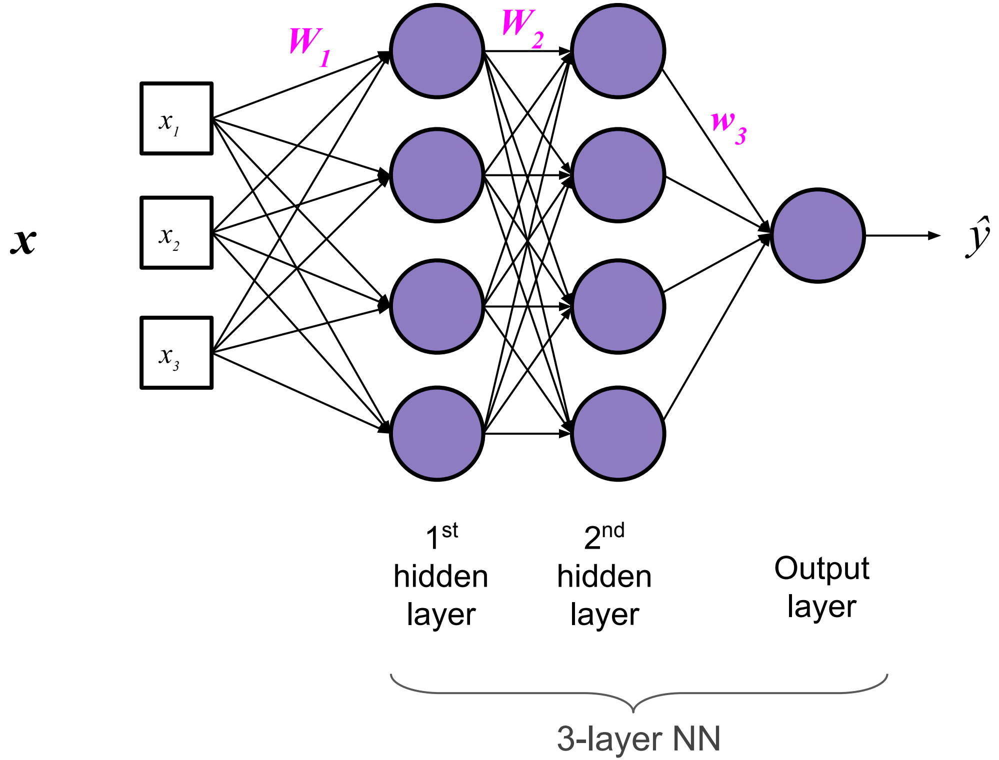
x
x1
x2
x3
W1
W2
w3
ŷ

Multi-Layer Neural Network

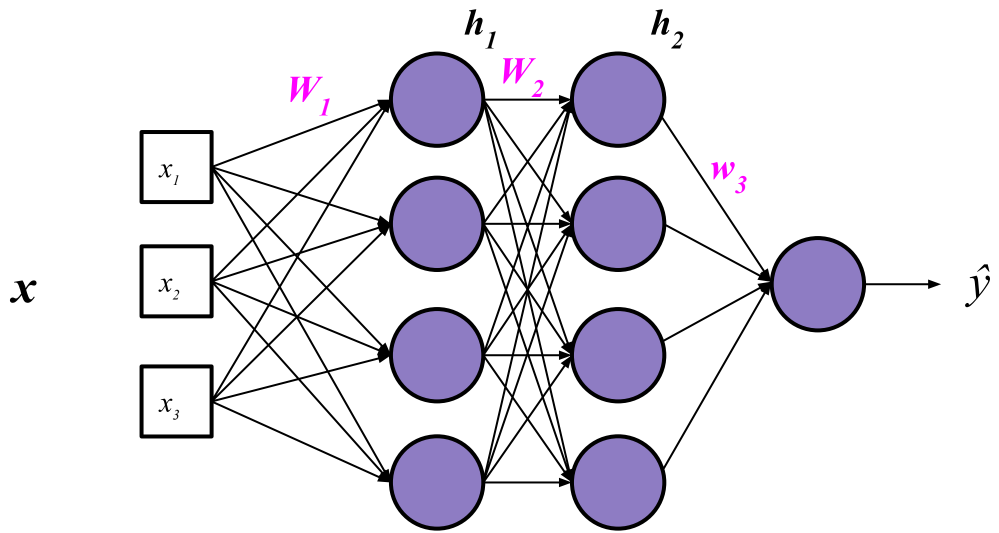
(Multi-Layer Perceptron (**MLP**) Network)



Multi-Layer Neural Network (MLP)



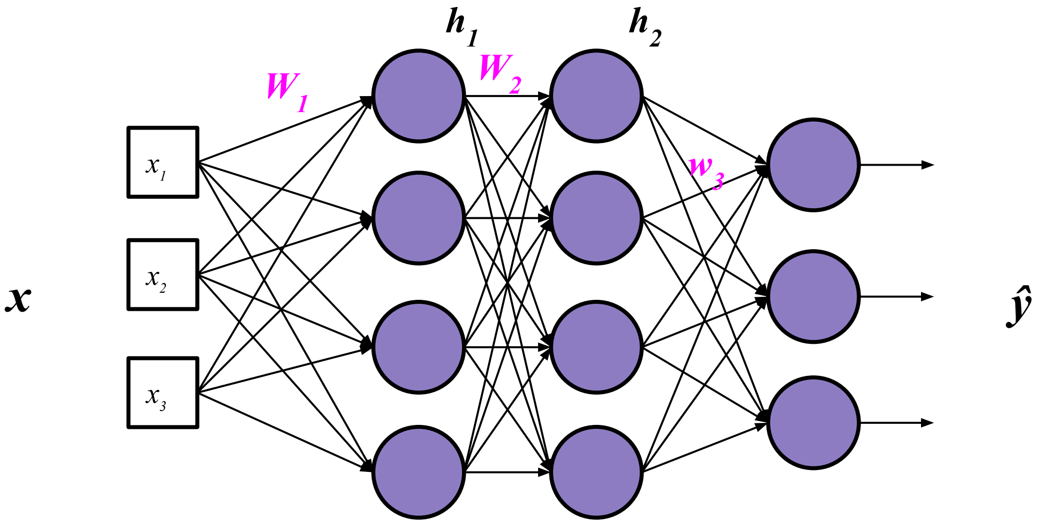
Multi-Layer Neural Network (MLP)



hidden layer 1 output

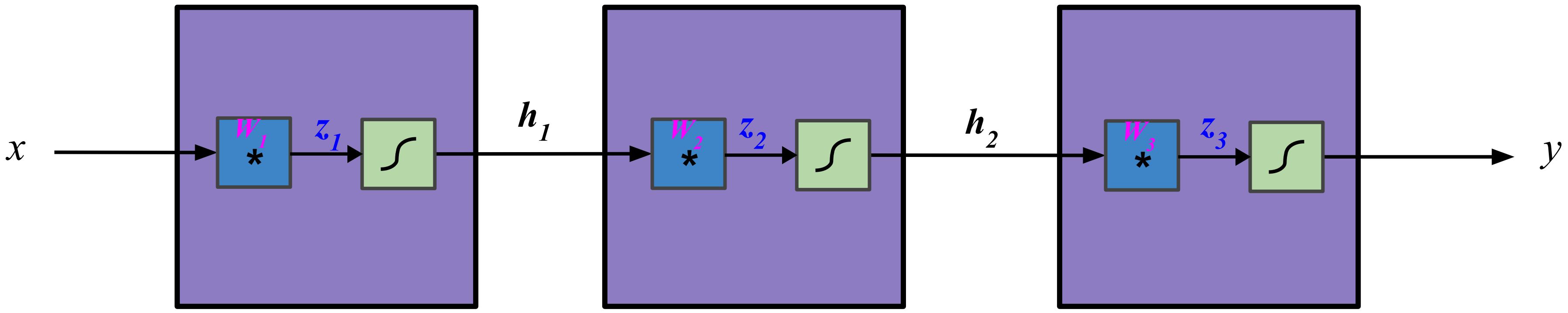
$$\boldsymbol{z_1} = \boldsymbol{W_1^T}\boldsymbol{x}$$
$$\boldsymbol{h_1} = sigmoid(\boldsymbol{z_1})$$
$$\boldsymbol{z_2} = \boldsymbol{W_2^T}\boldsymbol{h_1}$$
$$\boldsymbol{h_2} = sigmoid(\boldsymbol{z_2})$$
$$z_3 = \boldsymbol{w_3^T}\boldsymbol{h_2}$$
$$\hat{y} = sigmoid(z_3)$$

Multi-Class Output MLP



$$
\begin{aligned}
z_1 &= W_1^T x \\
h_1 &= sigmoid(z_1) \\
z_2 &= W_2^T h_1 \\
h_2 &= sigmoid(z_2) \\
z_3 &= W_3^T h_2 \\
\hat{y} &= sigmoid(z_2)
\end{aligned}
$$

"Block View" Of MLP

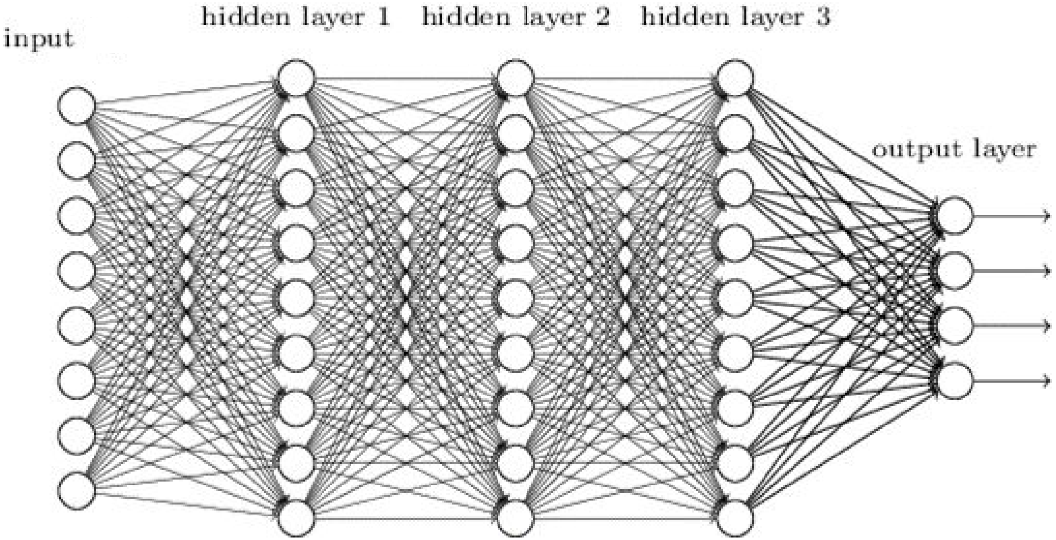


1st
hidden
layer

2nd
hidden
layer

Output
layer

“Deep” Neural Networks (i.e. > 1 hidden layer)



Researchers have successfully used 1000 layers to train an object classifier

Neurons

1-Layer Neural Network

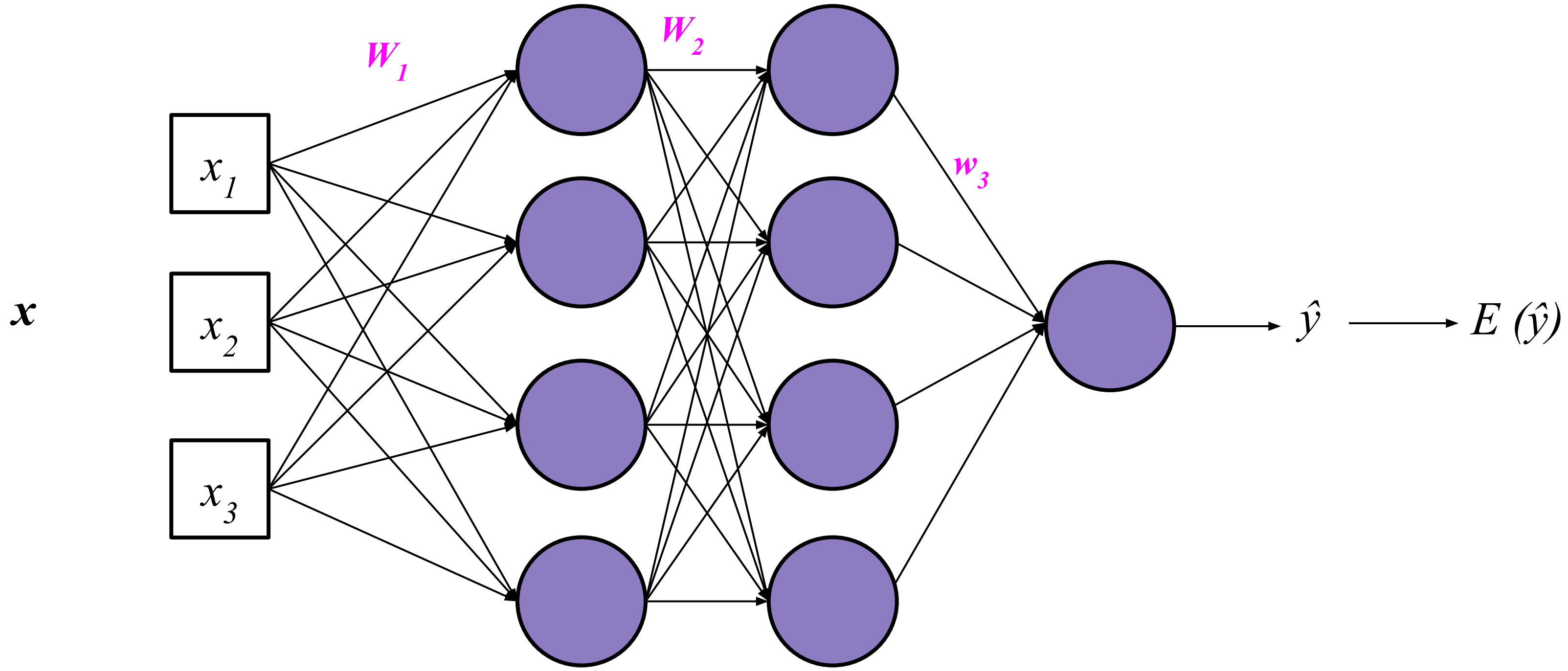
Multi-layer Neural Network

Loss Functions

Backpropagation

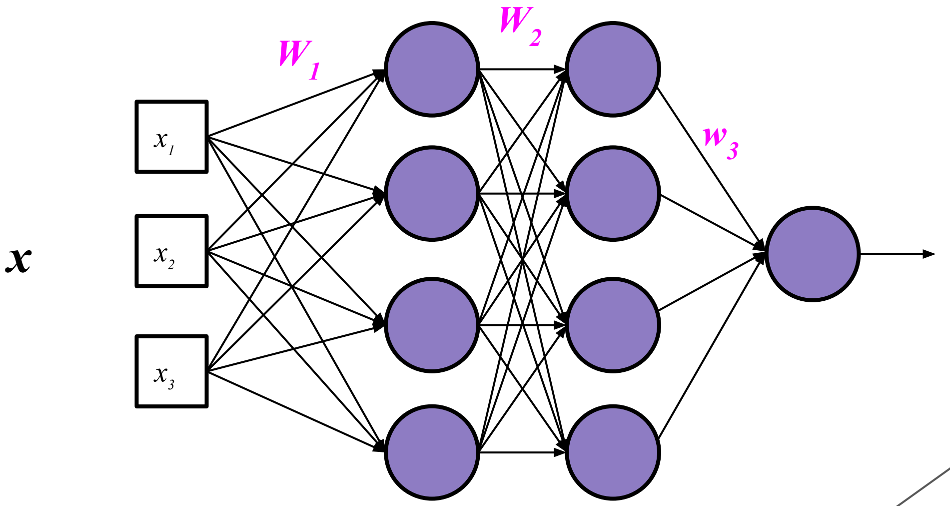
Nonlinearity Functions

NNs in Practice



x
x1
x2
x3
W1
W2
w3
ŷ
E (ŷ)

Binary Classification Loss



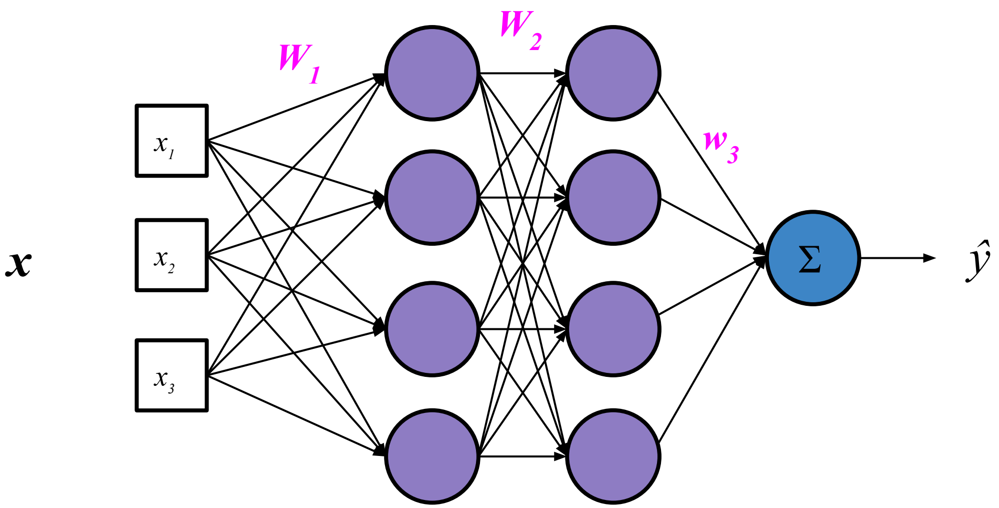
$\hat{y} = P(y=1|X,W)$

this example is for a single sample $\boldsymbol{x}$

$$
\begin{aligned}
E = \text{loss} &= -\log P(\mathrm{Y} = \hat{y} \mid \mathbf{X} = \boldsymbol{x}) \\
&= -\,y \log(\hat{y}) - (1 - y)\log(1-\hat{y})
\end{aligned}
$$

true output

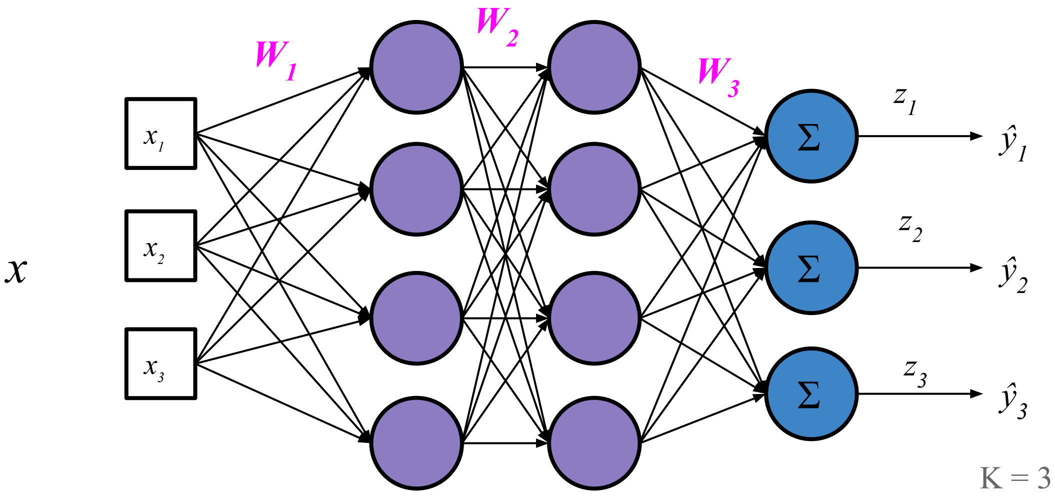
Regression Loss



$$E = \text{loss} = \frac{1}{2}(y - \hat{y})^2$$

true output

Multi-Class Classification Loss



$$\hat{y}_i = \frac{e^{z_i}}{\sum_j e^{z_j}} = P(\hat{y}_i = 1 \mid \boldsymbol{x})$$

"Softmax" function. Normalizing function which converts each class output to a probability.

$$E = \text{loss} = - \sum_{j = 1 \ldots K} y_j \ln \hat{y}_j$$

"0" for all except true class

Neurons

1-Layer Neural Network

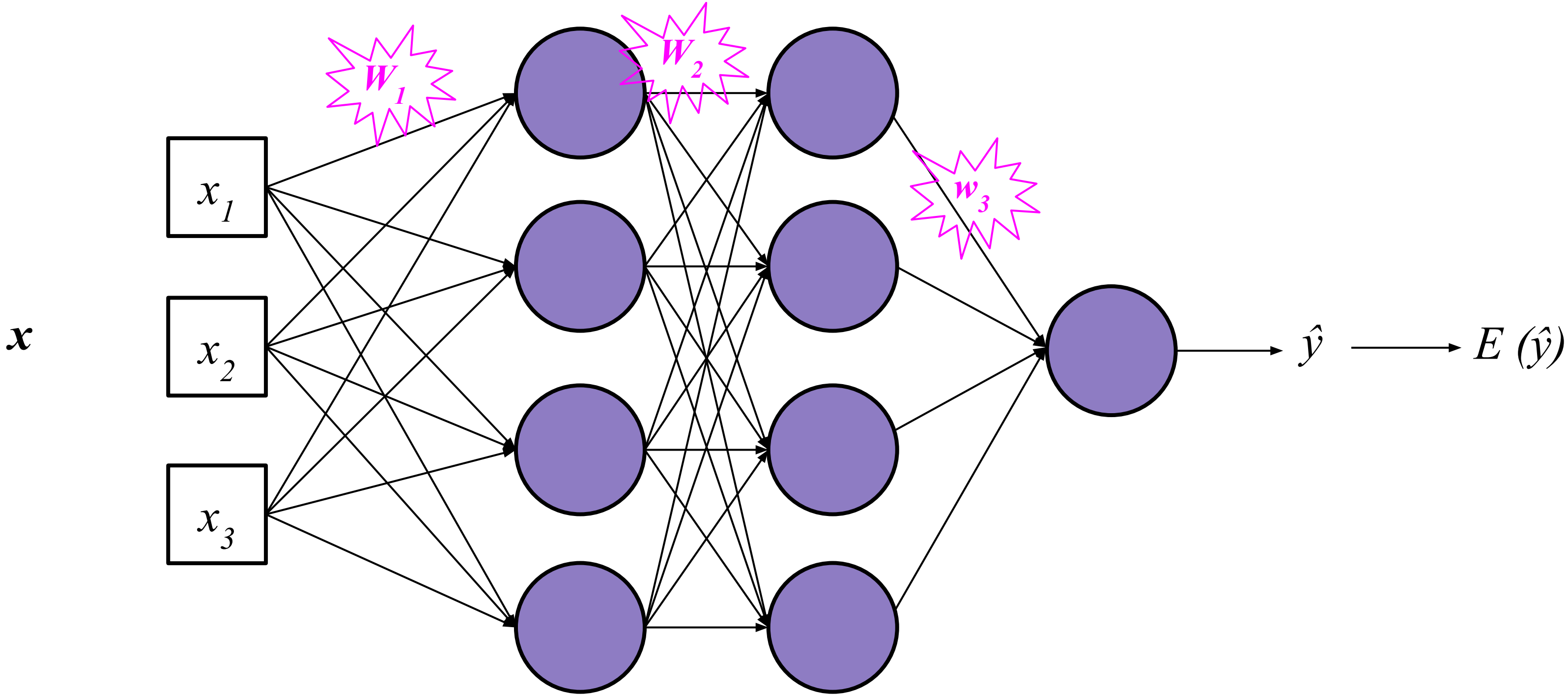
Multi-layer Neural Network

Loss Functions

Backpropagation

Nonlinearity Functions

NNs in Practice



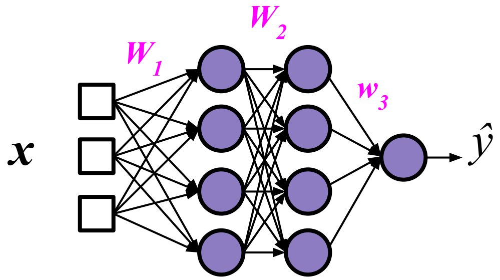
x
x1
x2
x3
W1
W2
w3
ŷ
E (ŷ)

Training Neural Networks

How do we learn the optimal weights W_L for our task??

- **Gradient descent:**

$$W_L(t+1) = W_L(t) - \eta \frac{\partial E}{\partial W_L(t)}$$



But how do we get gradients of lower layers?

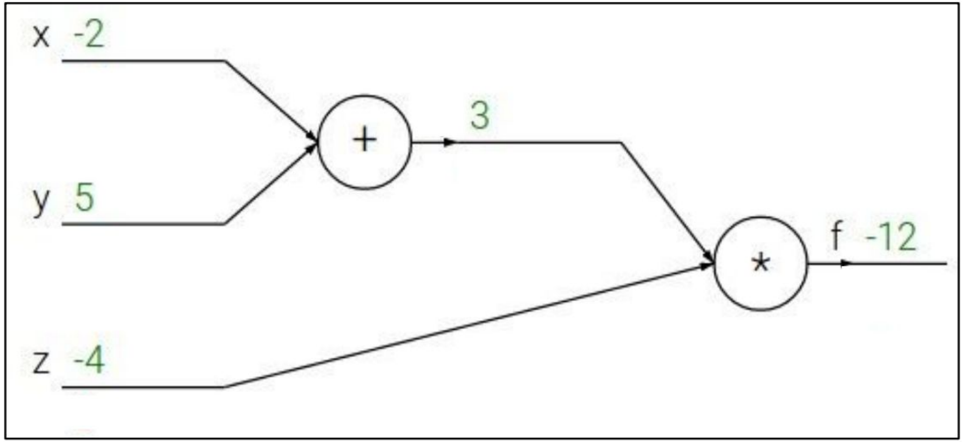
- **Backpropagation!**
  - Repeated application of chain rule of calculus
  - Locally minimize the objective
  - Requires all “blocks” of the network to be differentiable

LeCun et. al. *Efficient Backpropagation*. 1998

Backpropagation Intro

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

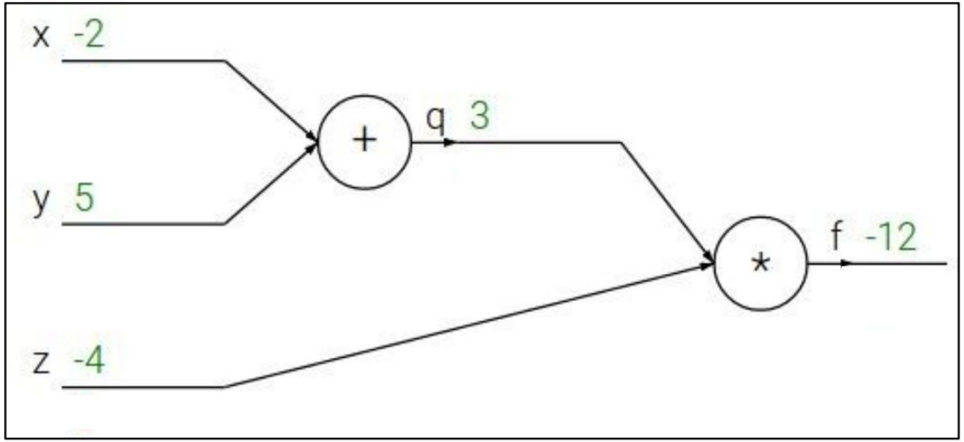


http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation Intro

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

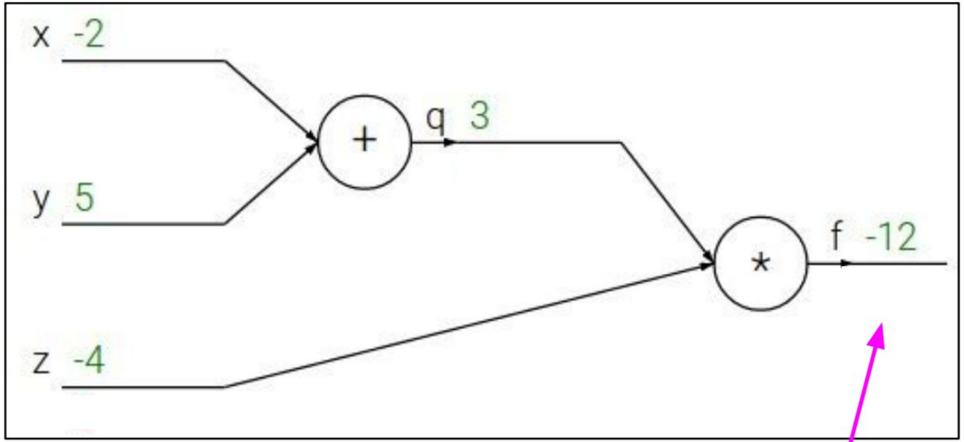
Backpropagation Intro

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



$$\frac{\partial f}{\partial f}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation Intro

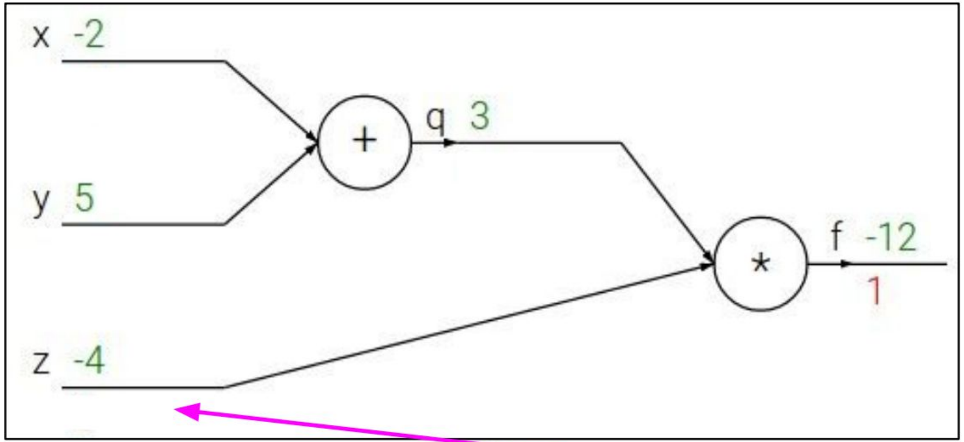
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial z}$

Backpropagation Intro

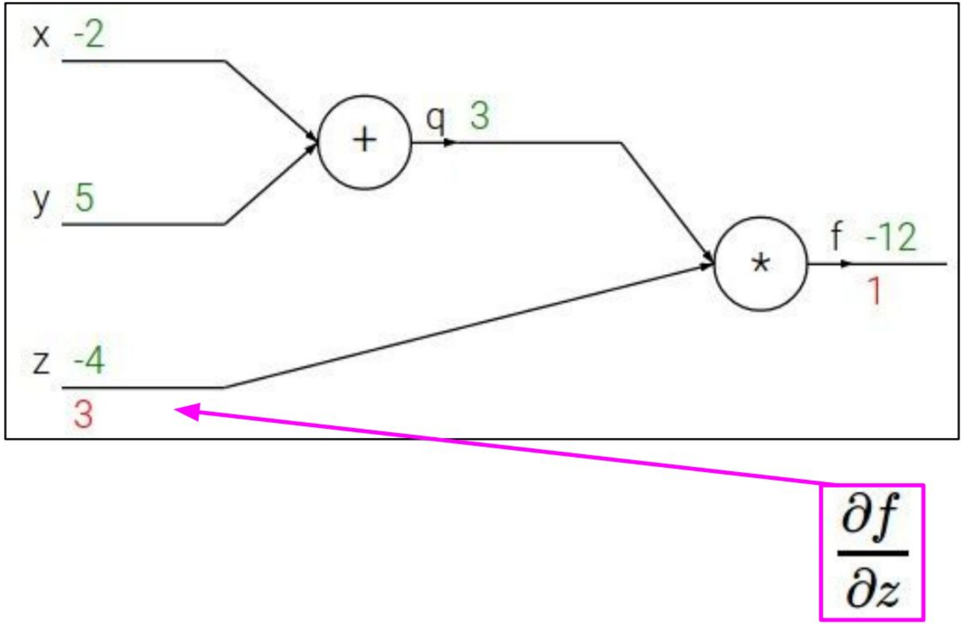
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

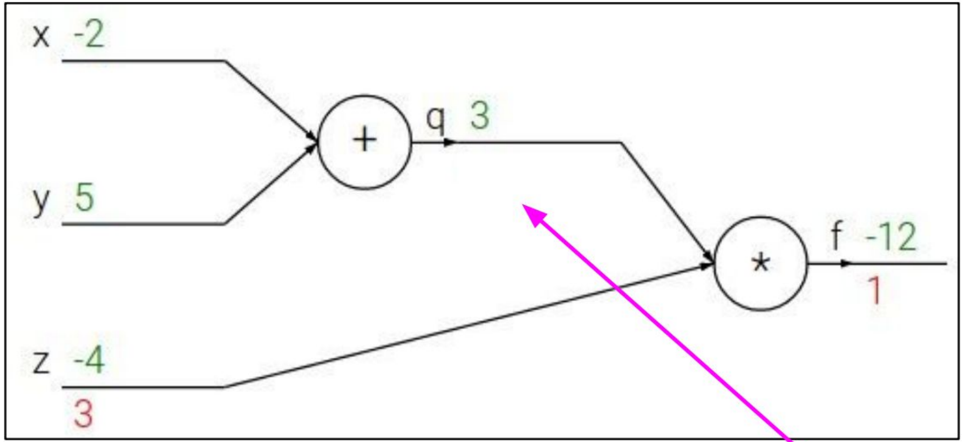
Backpropagation Intro

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



$$\boxed{\frac{\partial f}{\partial q}}$$

Want: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$

Backpropagation Intro

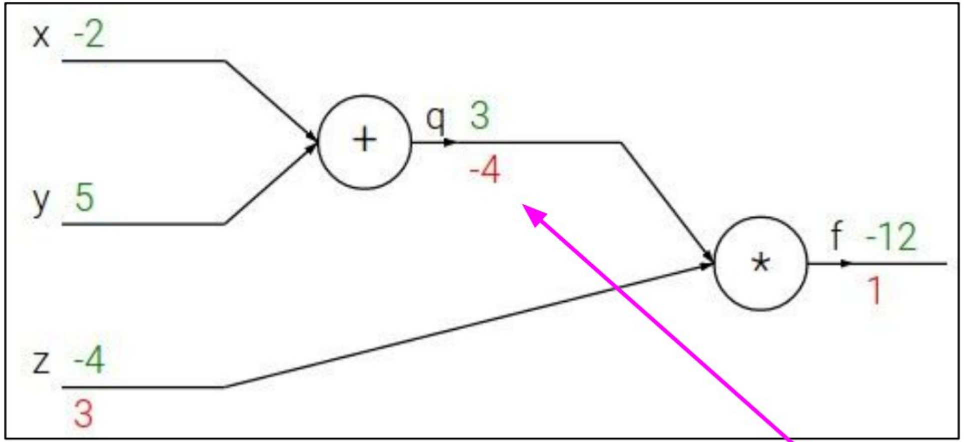
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation Intro

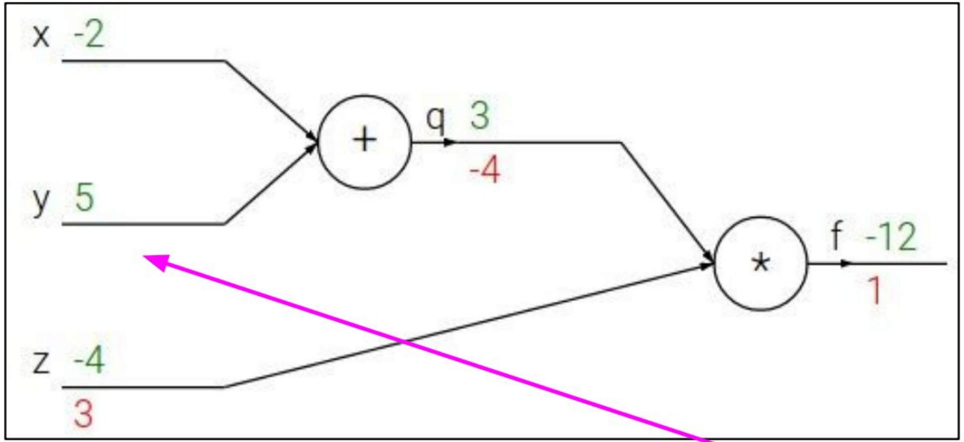
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

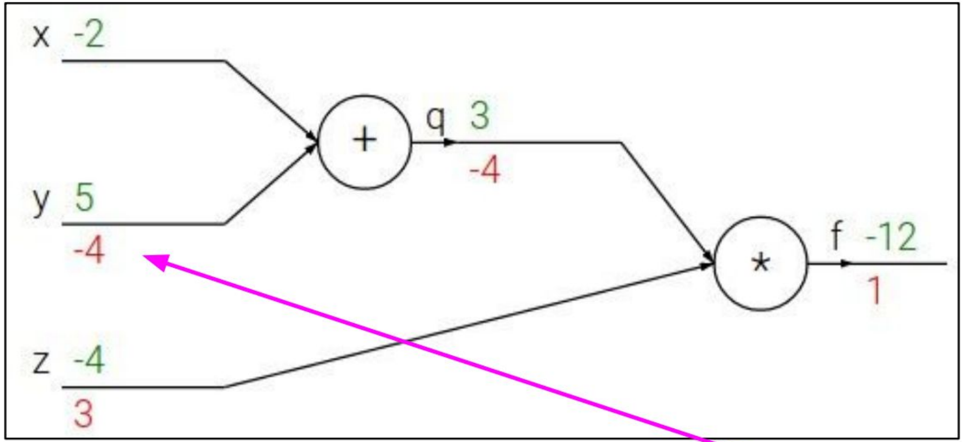


$$\frac{\partial f}{\partial y}$$

Backpropagation Intro

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\boxed{\frac{\partial f}{\partial y}}$$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation Intro

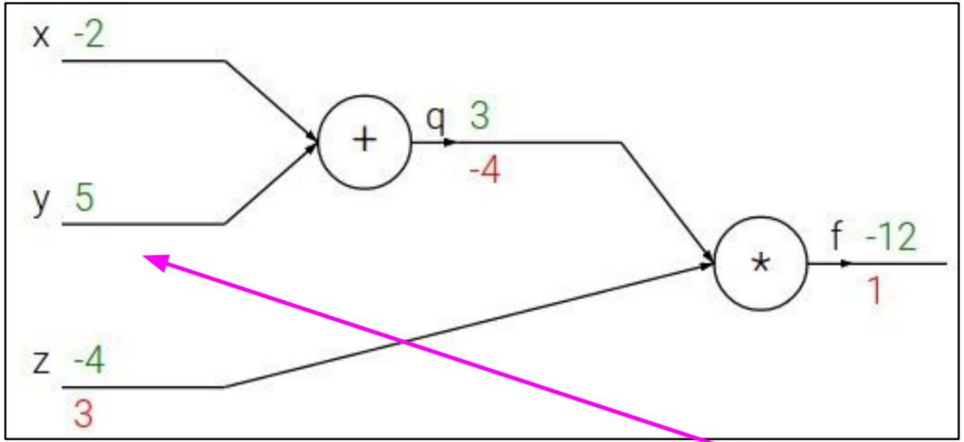
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation Intro

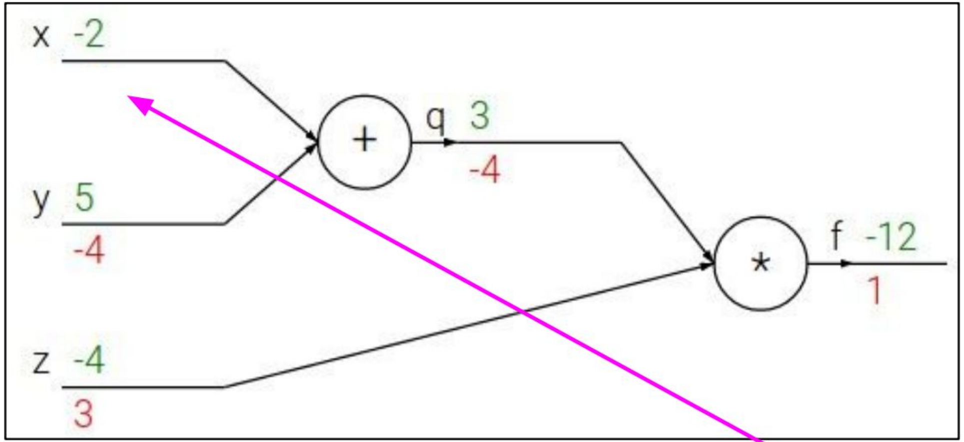
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial x}}$$

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation Intro

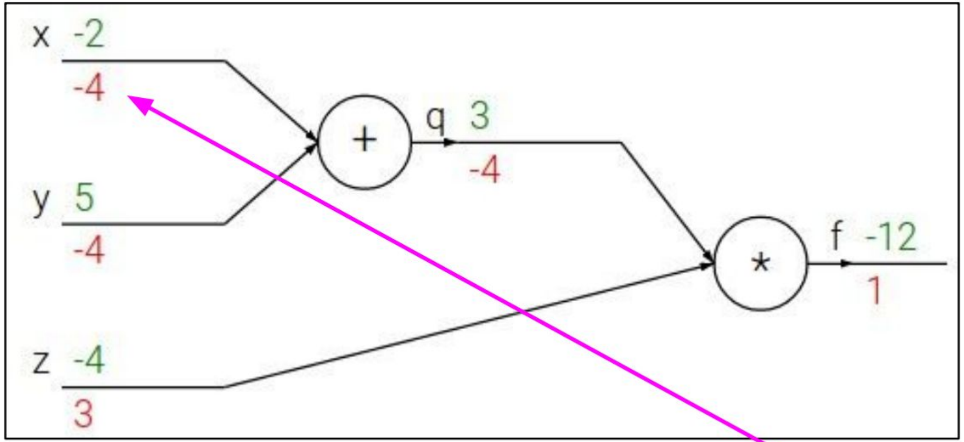
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial x}}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation Intro

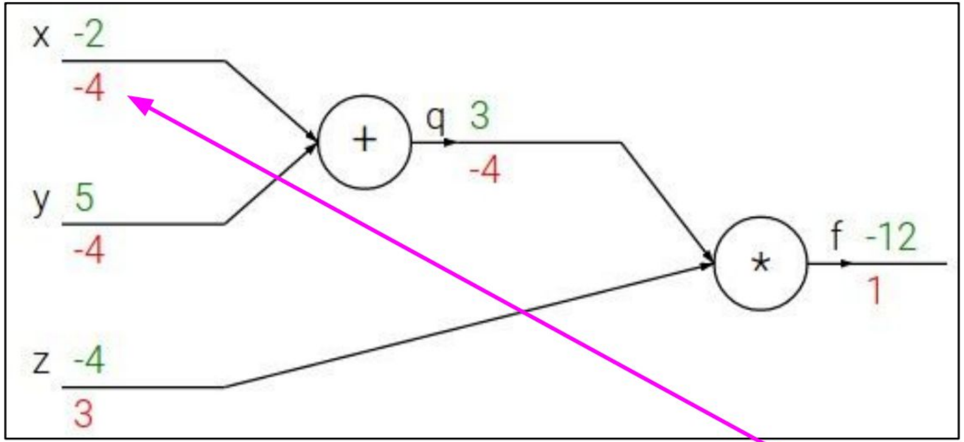
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



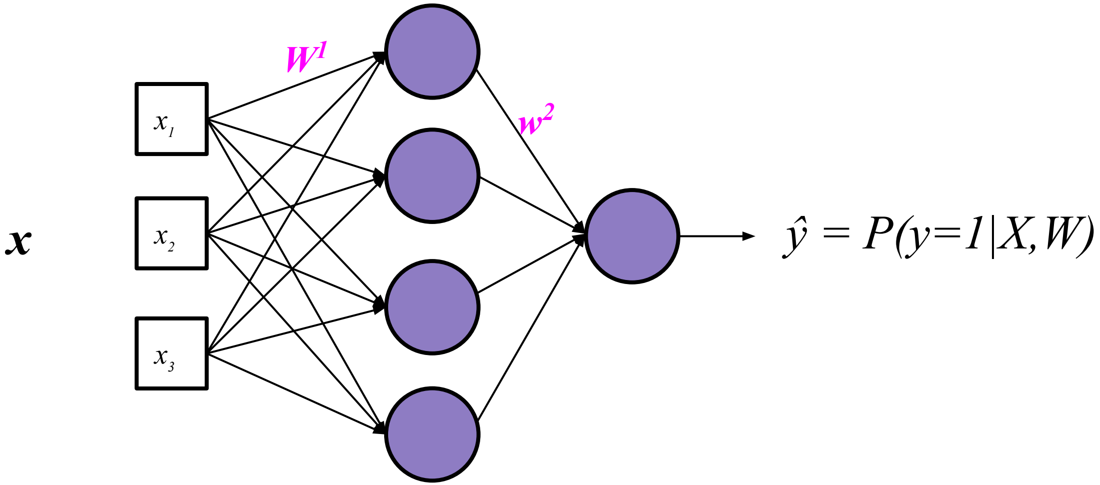
$$\frac{\partial f}{\partial x}$$

Tells us: by increasing x by a scale of 1, we decrease *f* by a scale of 4

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Backpropagation

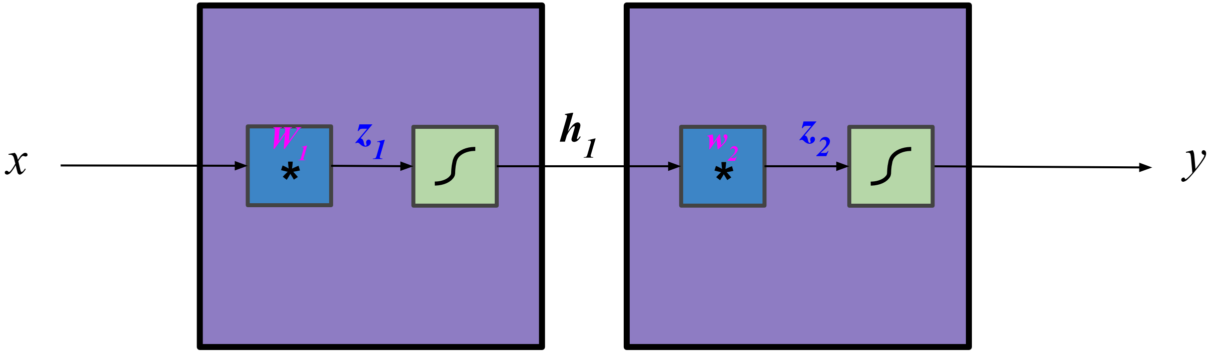
(binary classification example)



Example on 1-hidden layer NN for binary classification

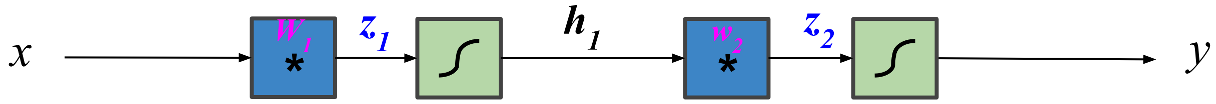
Backpropagation

(binary classification example)



Backpropagation

(binary classification example)



$$E = \text{loss} = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

Gradient Descent to Minimize loss:

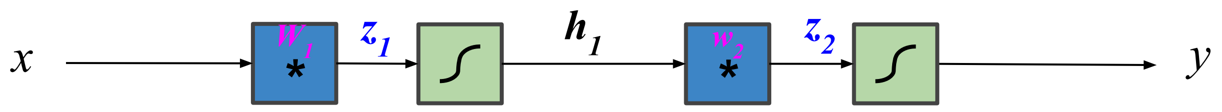
$$\mathbf{w}_2(t+1) = \mathbf{w}_2(t) - \eta\boxed{\frac{\partial E}{\partial \mathbf{w}_2(t)}}$$

$$W_1(t+1) = W_1(t) - \eta\boxed{\frac{\partial E}{\partial W_1(t)}}$$

Need to find these!

Backpropagation

(binary classification example)



$$E = f_4 = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = f_3 = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = f_3 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

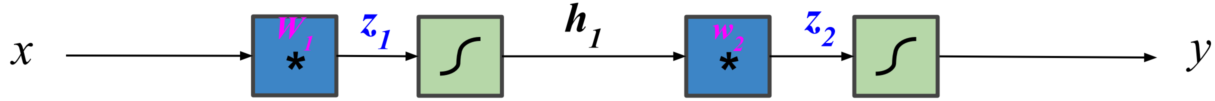
$$\boldsymbol{h}_1 = f_2 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = f_1 = W_1^T \boldsymbol{x}$$

$$E = f_4(f_3(f_2(f_1(x))))$$

Backpropagation

(binary classification example)



$$E = f_4 = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = f_3 = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = f_3 = \boxed{\boldsymbol{w}_2^T}\boldsymbol{h}_1$$

$$\boldsymbol{h}_1 = f_2 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = f_1 = \boxed{W_1^T}\boldsymbol{x}$$

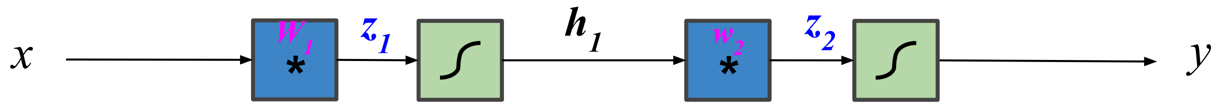
$$\boxed{\frac{\partial E}{\partial \mathbf{w}_2} = ??}$$

$$\boxed{\frac{\partial E}{\partial \mathbf{W}_1} = ??}$$

$$E = f_4(f_3(f_2(f_1(x))))$$

Backpropagation

(binary classification example)



$$E = f_4 = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = f_3 = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = f_3 = \boxed{\boldsymbol{w}_2^T}\boldsymbol{h}_1$$

$$\boldsymbol{h}_1 = f_2 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = f_1 = \boxed{W_1^T}\boldsymbol{x}$$

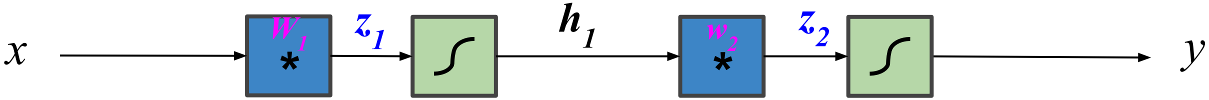
$$\boxed{\frac{\partial E}{\partial \mathbf{w}_2} = ??}$$

$$\boxed{\frac{\partial E}{\partial \mathbf{W}_1} = ??}$$

$$E = f_4(f_3(f_2(f_1(x))))$$ → Exploit the chain rule!

Backpropagation

(binary classification example)



$$E = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = \boxed{\boldsymbol{w}_2^T}\boldsymbol{h}_1$$

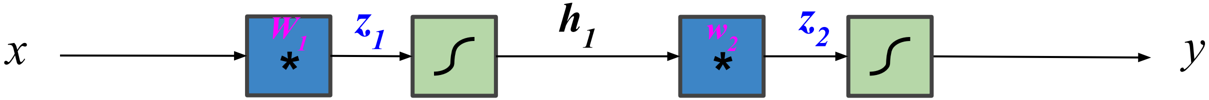
$$\boldsymbol{h}_1 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = W_1^T\boldsymbol{x}$$

$$\boxed{\frac{\partial E}{\partial \boldsymbol{w}_2}} =$$

Backpropagation

(binary classification example)

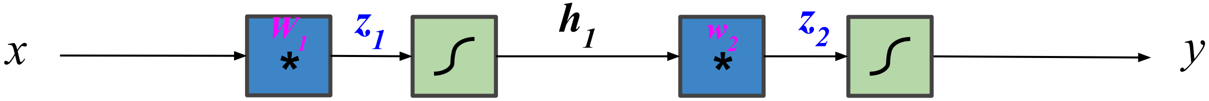


$$
\begin{aligned}
E &= -\,y\ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boxed{\boldsymbol{w}_2^T}\boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T\boldsymbol{x}
\end{aligned}
$$

$$
\boxed{\frac{\partial E}{\partial \boldsymbol{w}_2}} = \overbrace{\frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2}}^{\text{chain rule}}
$$

Backpropagation

(binary classification example)



$$\boxed{\begin{aligned} E = & -y \ln(\hat{y}) \\ & -(1-y)\ln(1-\hat{y}) \end{aligned}}$$

$$\hat{y} = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

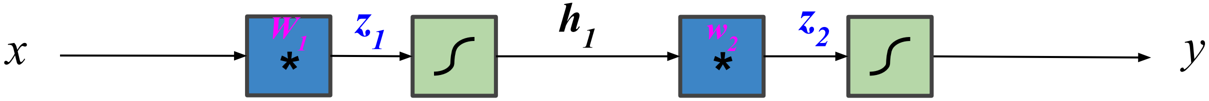
$$\boldsymbol{h}_1 = \frac{e^{\boldsymbol{z}_1}}{1+e^{\boldsymbol{z}_1}}$$

$$\boldsymbol{z}_1 = W_1^T \boldsymbol{x}$$

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{w}_2} &= \boxed{\frac{\partial E}{\partial \hat{y}}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\ &= \boxed{\left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \right)} \cdot \end{aligned}$$

Backpropagation

(binary classification example)



$$
\begin{aligned}
E = &-y \ln(\hat{y}) \\
&-(1-y)\ln(1-\hat{y})
\end{aligned}
$$

$$
\boxed{\hat{y} = \frac{e^{z_2}}{1+e^{z_2}}}
$$

$$
z_2 = \boldsymbol{w}_2^T \boldsymbol{h}_1
$$

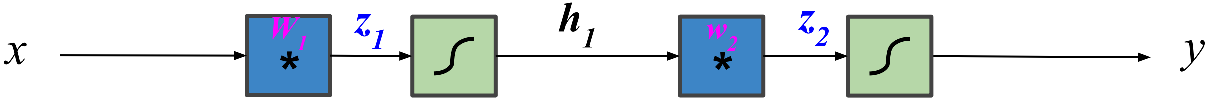
$$
\boldsymbol{h}_1 = \frac{e^{\boldsymbol{z}_1}}{1+e^{\boldsymbol{z}_1}}
$$

$$
\boldsymbol{z}_1 = W_1^T \boldsymbol{x}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}_2} &= \frac{\partial E}{\partial \hat{y}} \cdot \boxed{\frac{\partial \hat{y}}{\partial z_2}} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot \boxed{\left(\frac{e^{z_2}}{1+e^{z_2}}\left(1-\frac{e^{z_2}}{1+e^{z_2}}\right)\right)} \cdot
\end{aligned}
$$

Backpropagation

(binary classification example)



$$E = -\,y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = \frac{e^{z_2}}{1+e^{z_2}}$$

$$\boxed{z_2 = \boldsymbol{w}_2^T \boldsymbol{h}_1}$$

$$\boldsymbol{h}_1 = \frac{e^{z_1}}{1+e^{z_1}}$$

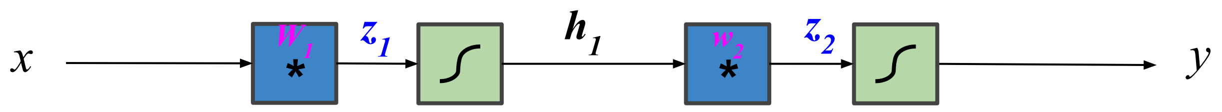
$$\boldsymbol{z}_1 = W_1^T \boldsymbol{x}$$

$$\frac{\partial E}{\partial \boldsymbol{w}_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \boxed{\frac{\partial z_2}{\partial \boldsymbol{w}_2}}$$

$$= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot \left(\frac{e^{z_2}}{1+e^{z_2}}\left(1-\frac{e^{z_2}}{1+e^{z_2}}\right)\right) \cdot \boxed{(\boldsymbol{h}_1)}$$

Backpropagation

(binary classification example)

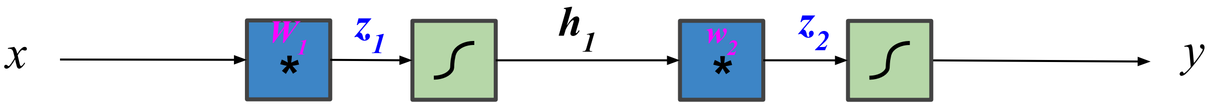


$$
\begin{aligned}
E &= -y\ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \cdot \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \cdot \boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}_2} &= \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot \left(\frac{e^{z_2}}{1+e^{z_2}}\left(1-\frac{e^{z_2}}{1+e^{z_2}}\right)\right) \cdot (\boldsymbol{h}_1) \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot (\hat{y}(1-\hat{y})) \cdot (\boldsymbol{h}_1)
\end{aligned}
$$

Backpropagation

(binary classification example)

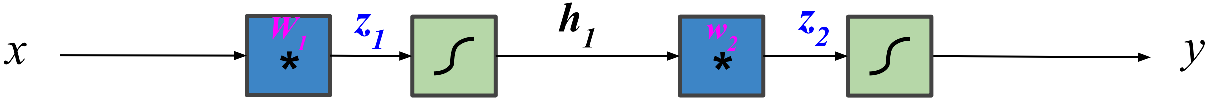


$$
\begin{aligned}
E &= -y\ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= \boxed{W_1^T}\boldsymbol{x}
\end{aligned}
$$

$$\boxed{\frac{\partial E}{\partial \boldsymbol{W}_1}} =$$

Backpropagation

(binary classification example)

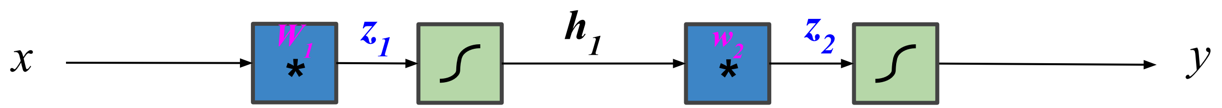


$$
\begin{aligned}
E &= -\,y\ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= \boxed{W_1^T}\boldsymbol{x}
\end{aligned}
$$

$$
\boxed{\frac{\partial E}{\partial \boldsymbol{W}_1}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial W_1}
$$

Backpropagation

(binary classification example)



$$E = -\,y \ln(\hat{y}) \\ \quad -\,(1-y)\ln(1-\hat{y})$$

$$\hat{y} = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

$$\boldsymbol{h}_1 = \frac{e^{z_1}}{1+e^{z_1}}$$

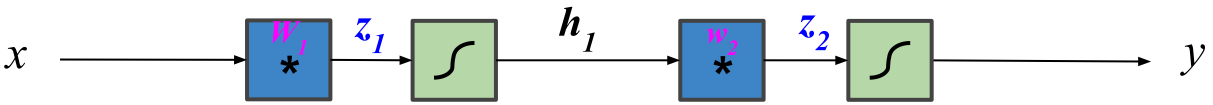
$$\boldsymbol{z}_1 = W_1^T \boldsymbol{x}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial W_1}$$

$$= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot (\hat{y}(1-\hat{y})) \cdot (\boldsymbol{w}) \cdot (\boldsymbol{h}_1(1-\boldsymbol{h}_1)) \cdot (x)$$

Backpropagation

(binary classification example)



$$E = -\,y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

$$\boldsymbol{h}_1 = \frac{e^{z_1}}{1+e^{z_1}}$$

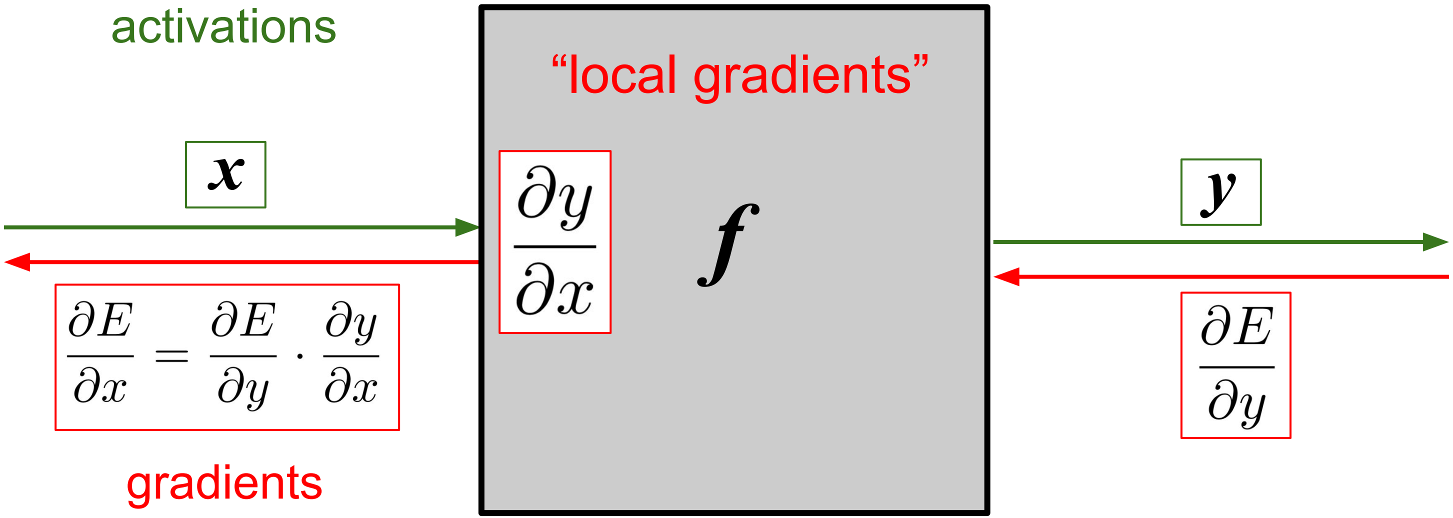
$$\boldsymbol{z}_1 = W_1^T \boldsymbol{x}$$

already computed

$$\frac{\partial E}{\partial \boldsymbol{W}_1} = \boxed{\frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2}} \cdot \frac{\partial z_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial W_1}$$

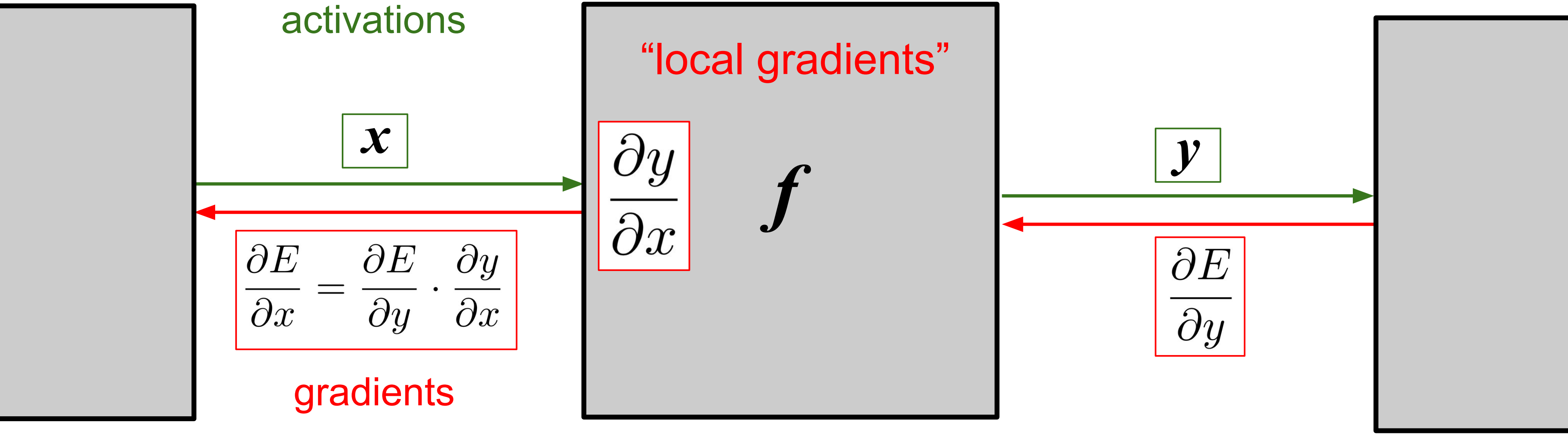
$$= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot (\hat{y}(1-\hat{y})) \cdot (\boldsymbol{w}) \cdot (\boldsymbol{h}_1(1-\boldsymbol{h}_1)) \cdot (x)$$

“Local-ness” of Backpropagation

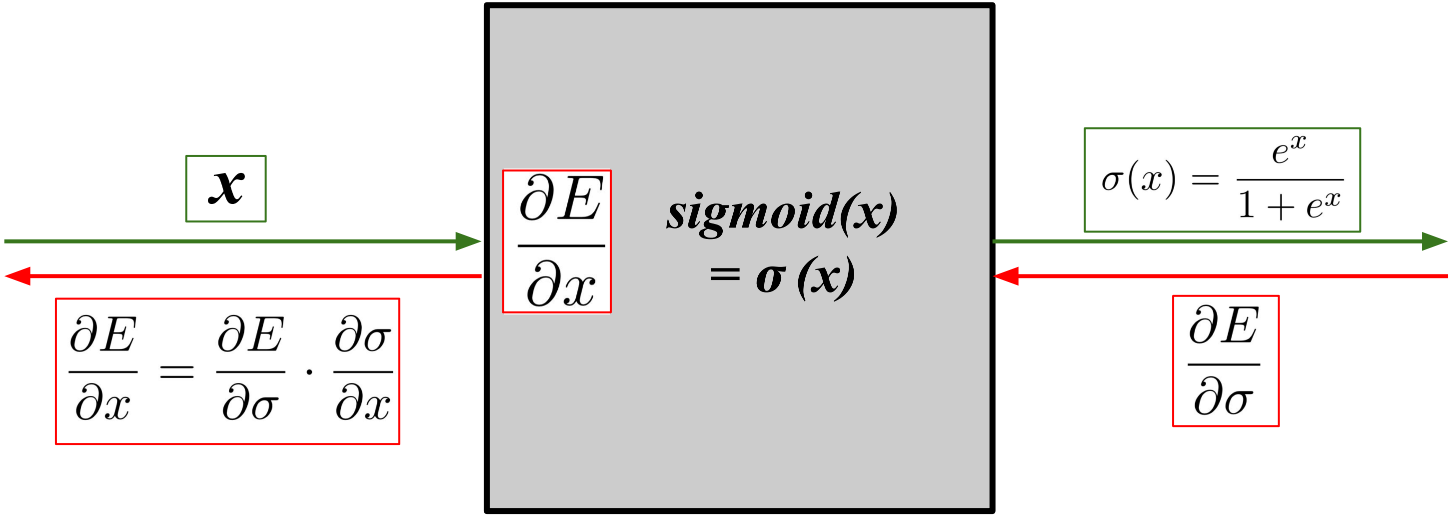


http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

“Local-ness” of Backpropagation

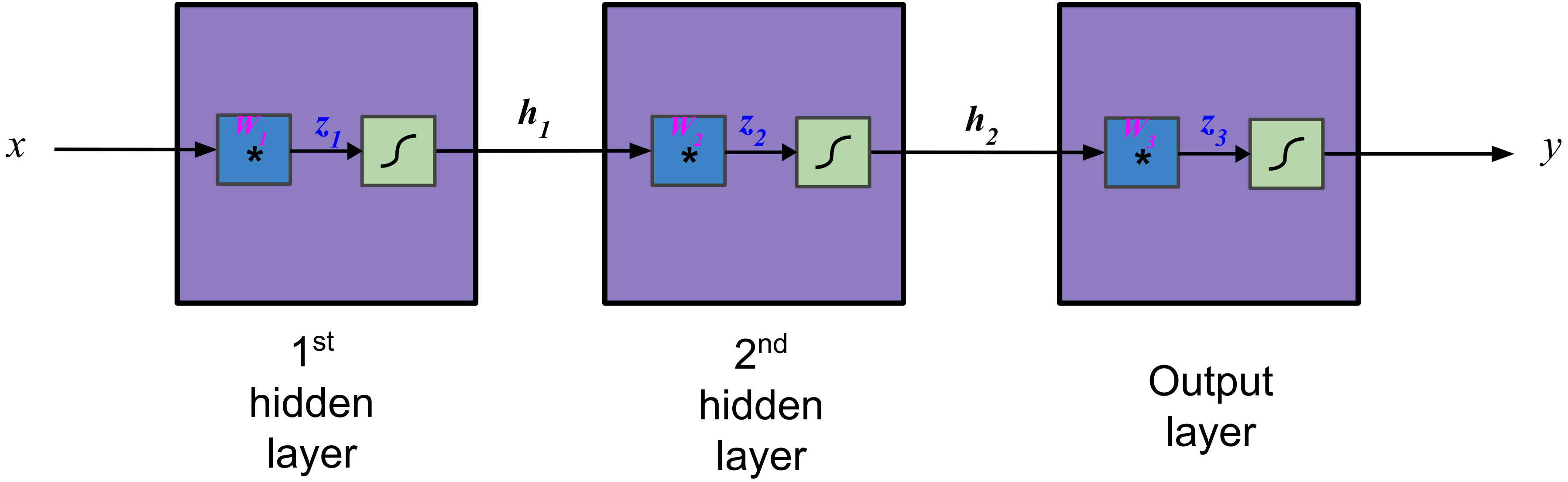


Example: Sigmoid Block



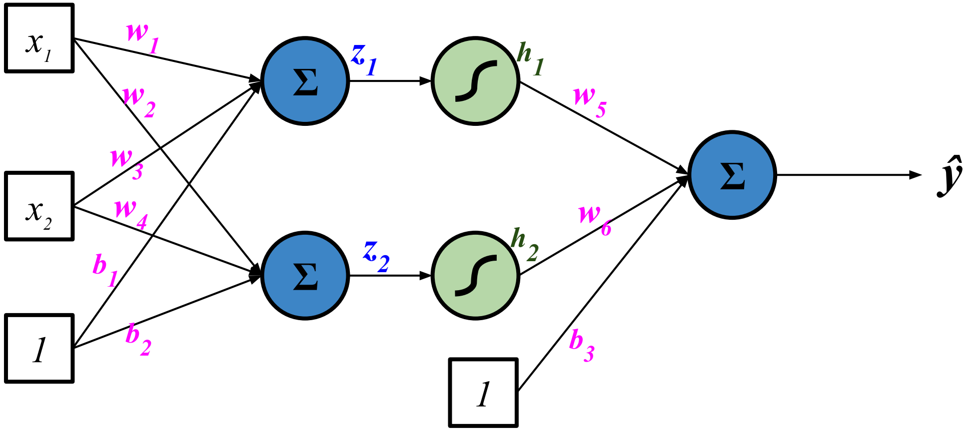
Deep Learning =

Concatenation of Differentiable Parameterized Layers (linear & nonlinearity functions)



Want to find optimal weights W to minimize some loss function E!

Backprop Whiteboard Demo



f_1	$z_1 = x_1 w_1 + x_2 w_3 + b_1$ $z_2 = x_1 w_2 + x_2 w_4 + b_2$
f_2	$h_1 = \frac{exp(z_1)}{1 + exp(z_1)}$ $h_2 = \frac{exp(z_2)}{1 + exp(z_2)}$
f_3	$\hat{y} = h_1 w_5 + h_2 w_6 + b_3$
f_4	$E = (y - \hat{y})^2$

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w(t)}$$

$$\boxed{\frac{\partial E}{\partial w} = ??}$$

Neurons

1-Layer Neural Network

Multi-layer Neural Network

Loss Functions

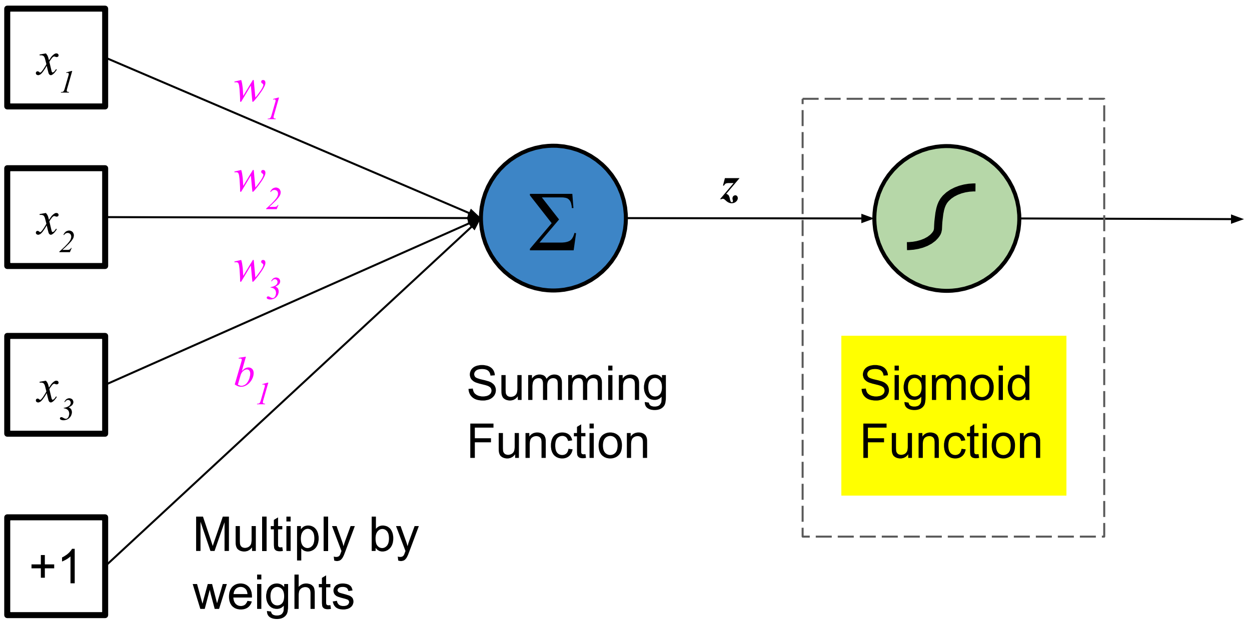
Backpropagation

Nonlinearity Functions

NNs in Practice

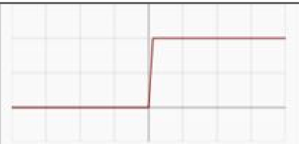
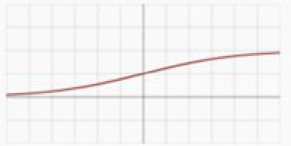
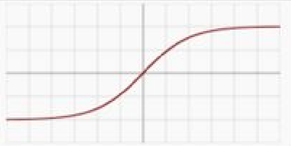
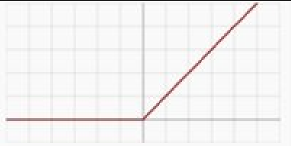
Nonlinearity Functions

(i.e. transfer or activation functions)



Nonlinearity Functions

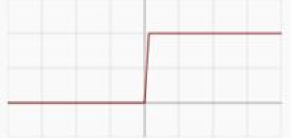
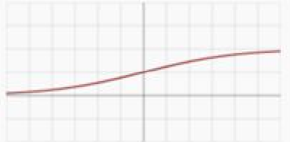
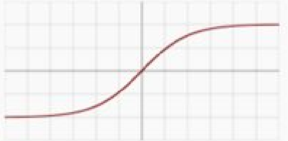
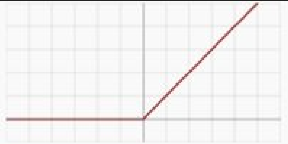
(i.e. transfer or activation functions)

Name	Plot	Equation	Derivative (w.r.t x)
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectifier (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$

https://en.wikipedia.org/wiki/Activation_function#Comparison_of_activation_functions

Nonlinearity Functions

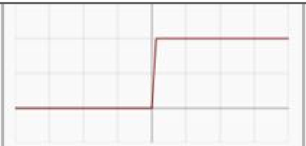
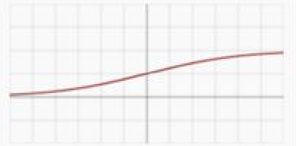
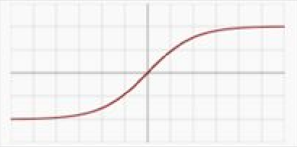
(i.e. transfer or activation functions)

Name	Plot	Equation	Derivative (w.r.t x)
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectifier (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$

https://en.wikipedia.org/wiki/Activation_function#Comparison_of_activation_functions

Nonlinearity Functions

(aka transfer or activation functions)

Name	Plot	Equation	Derivative (w.r.t x)
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectifier (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$

usually works best in practice

https://en.wikipedia.org/wiki/Activation_function#Comparison_of_activation_functions

Neurons

1-Layer Neural Network

Multi-layer Neural Network

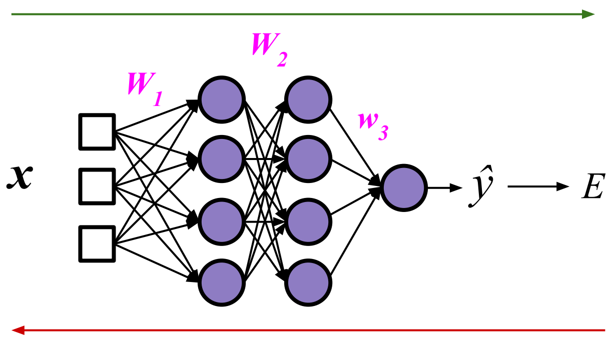
Loss Functions

Backpropagation

Nonlinearity Functions

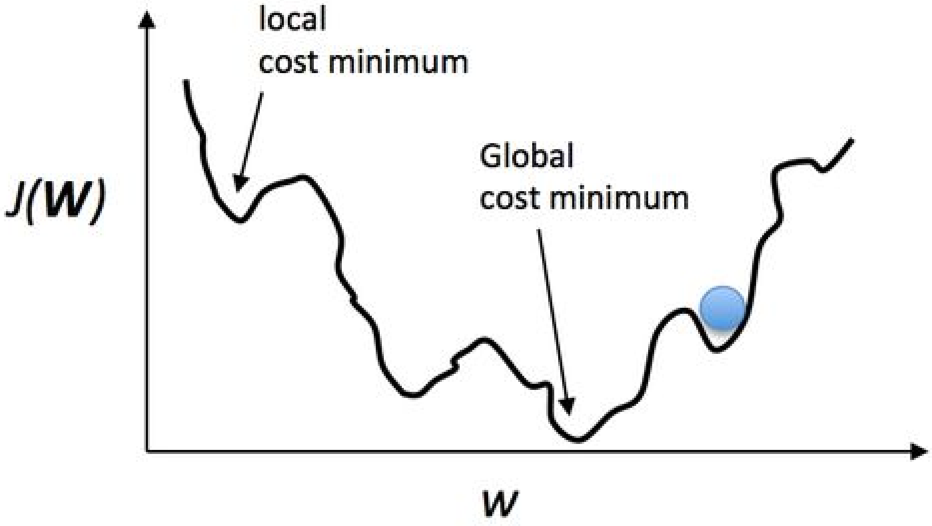
NNs in Practice

Neural Net Pipeline



1. Initialize weights
2. For each batch of input x samples S:
   a. Run the network "Forward" on S to compute outputs and loss
   b. Run the network "Backward" using outputs and loss to compute gradients
   c. Update weights using SGD (or a similar method)
3. Repeat step 2 until loss convergence

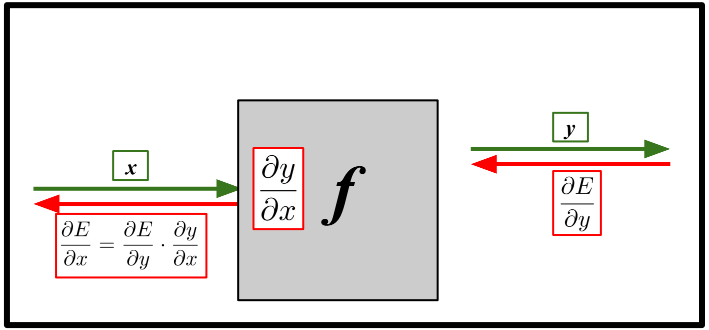
Non-Convexity of Neural Nets



In very high dimensions, there exists many local minimum which are about the same.

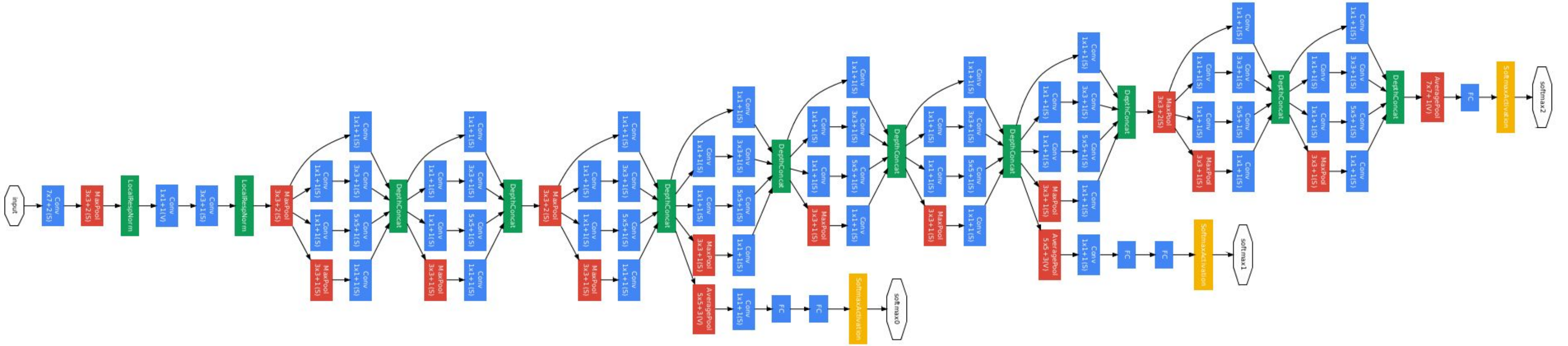
Pascanu, et. al. *On the saddle point problem for non-convex optimization 2014*

Building Deep Neural Nets



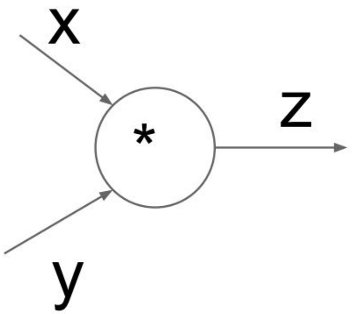
http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Building Deep Neural Nets



“GoogLeNet” for Object Classification

Block Example Implementation

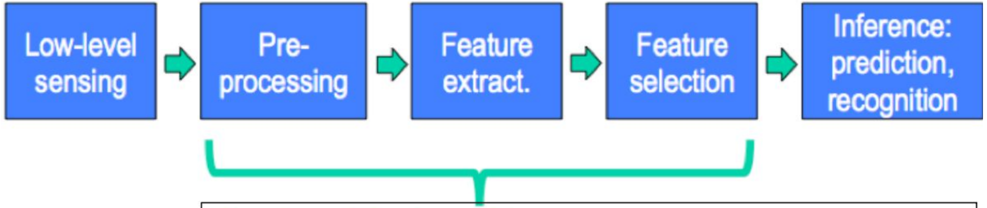



```python
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        self.x = x # must keep these around!
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz # [dz/dx * dL/dz]
        dy = self.x * dz # [dz/dy * dL/dz]
        return [dx, dy]
```

(x,y,z are scalars)

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Advantage of Neural Nets



Feature Engineering

- ✓ Most critical for accuracy
- ✓ Account for most of the computation for testing
- ✓ Most time-consuming in development cycle
- ✓ Often hand-craft and task dependent in practice

Feature Learning

- ✓ Easily adaptable to new similar tasks
- ✓ Layerwise representation
- ✓ Layer-by-layer unsupervised training
- ✓ Layer-by-layer supervised training

As long as it’s fully differentiable, we can train the model to automatically learn features for us.

Advanced Deep Learning Models:

Convolutional Neural Networks

& Recurrent Neural Networks

Most slides from http://cs231n.stanford.edu/

Convolutional Neural Networks
(aka CNNs and ConvNets)

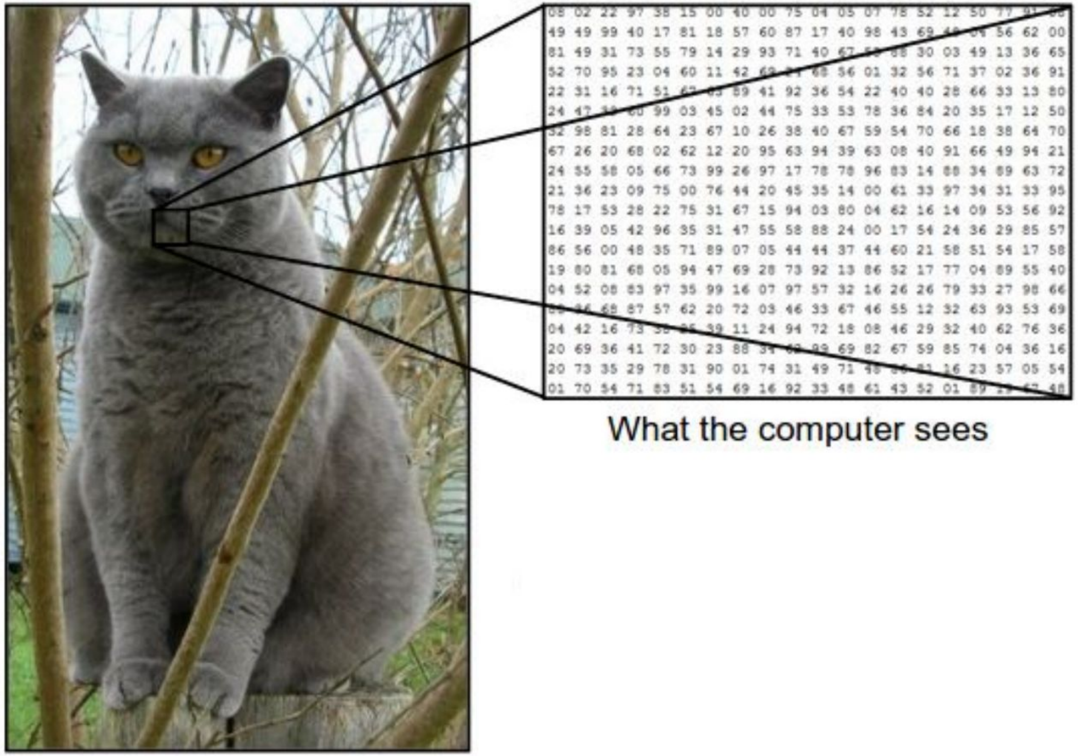
Challenges in Visual Recognition

The problem:
semantic gap

Images are represented as 3D arrays of numbers, with integers between [0, 255].

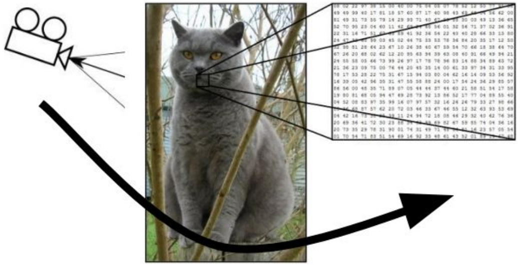
E.g.
300 x 100 x 3

(3 for 3 color channels RGB)



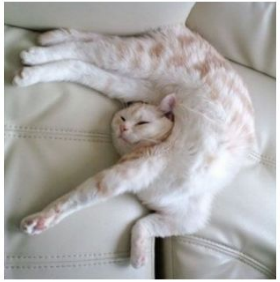
Challenges in Visual Recognition

Camera pose



Illumination

Deformation



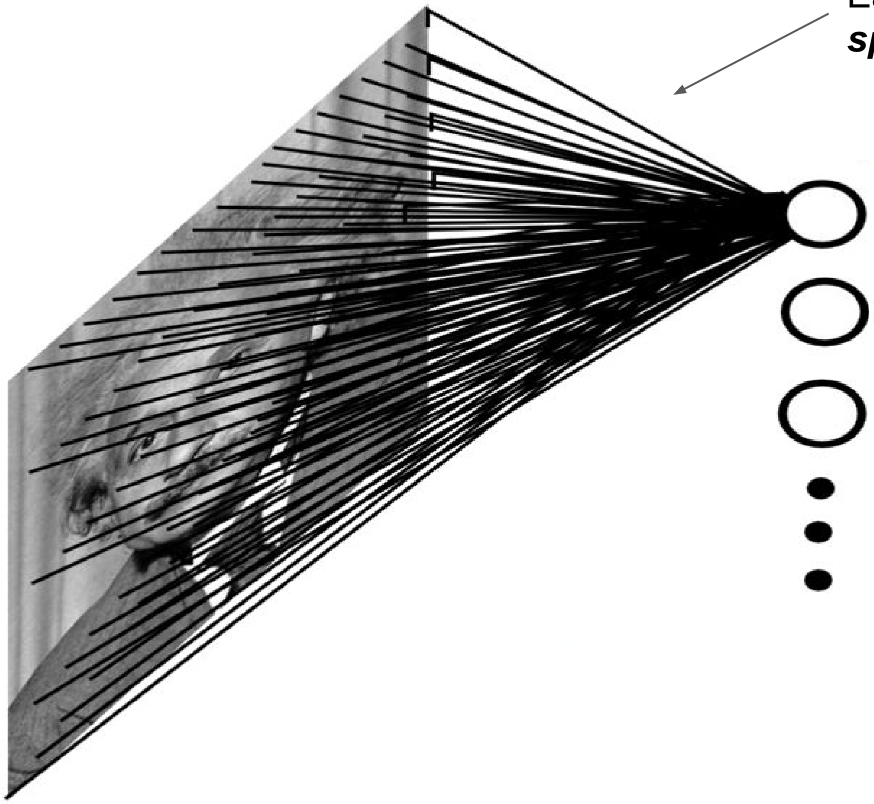
Occlusion

Background clutter

Intraclass variation

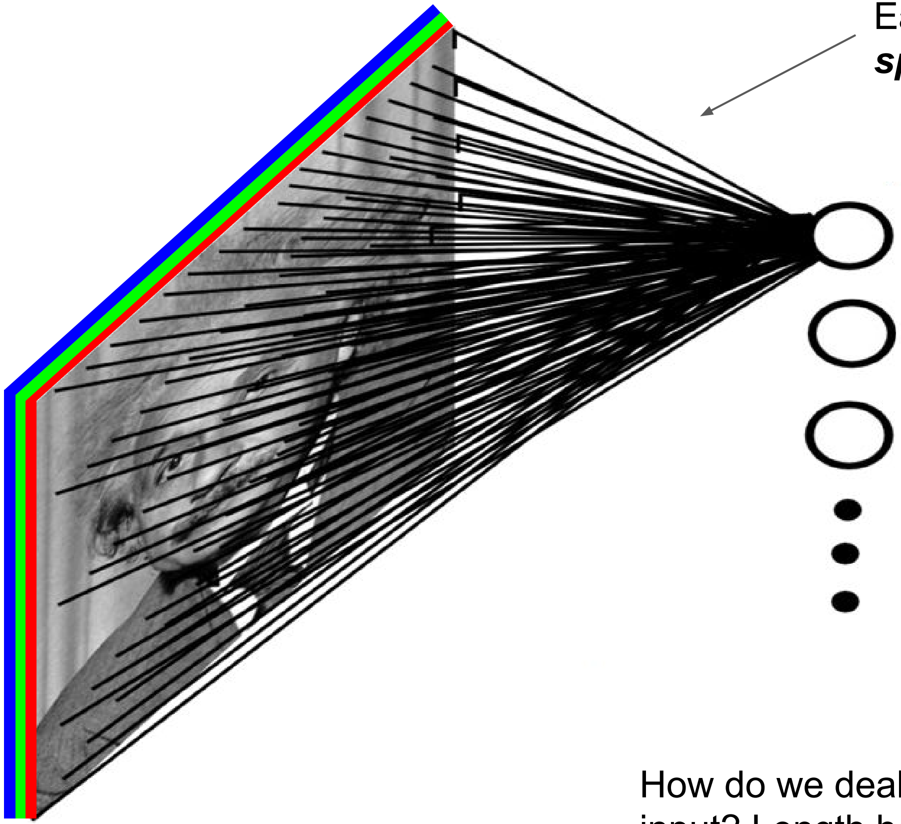
Problems with “Fully Connected Networks” on Images

Each neuron corresponds to a ***specific*** pixel



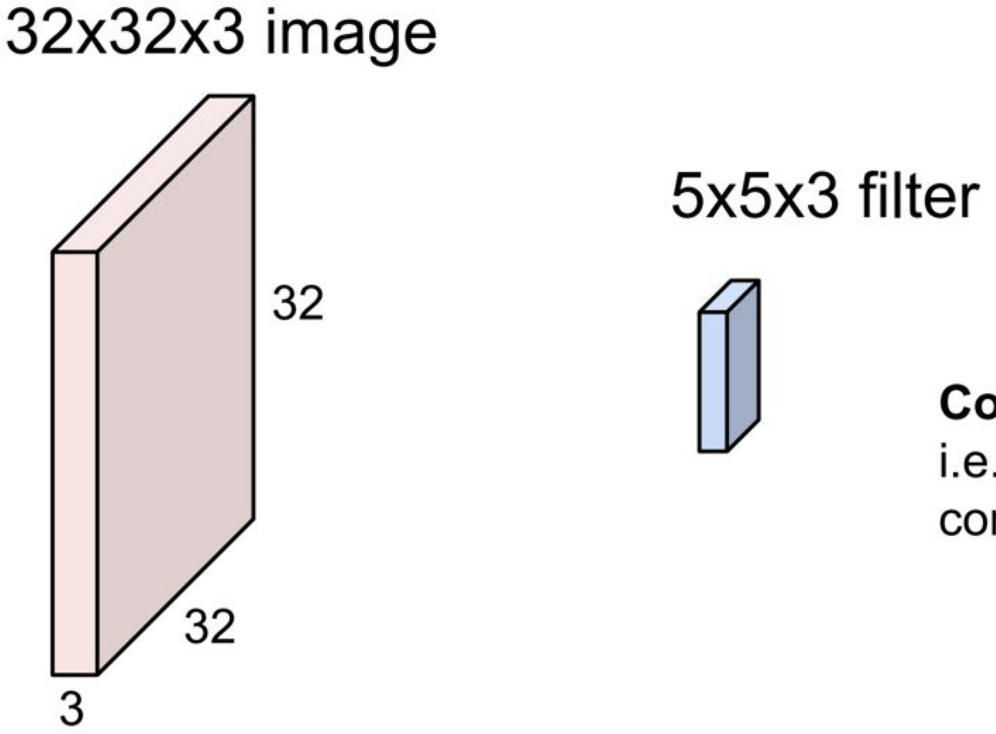
Problems with “Fully Connected Networks” on Images

Each neuron corresponds to a ***specific*** pixel



How do we deal with multiple dimensions in the input? Length,height, channel (R,G,B)

Convolutional Layer

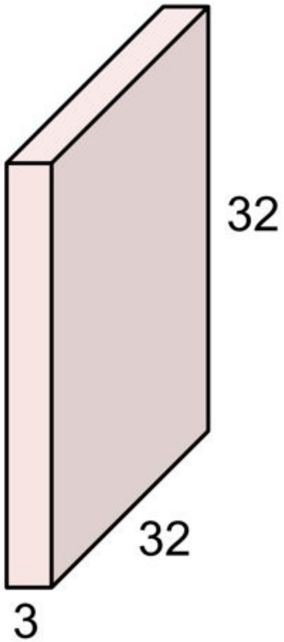


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

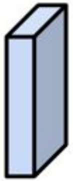
Convolutional Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

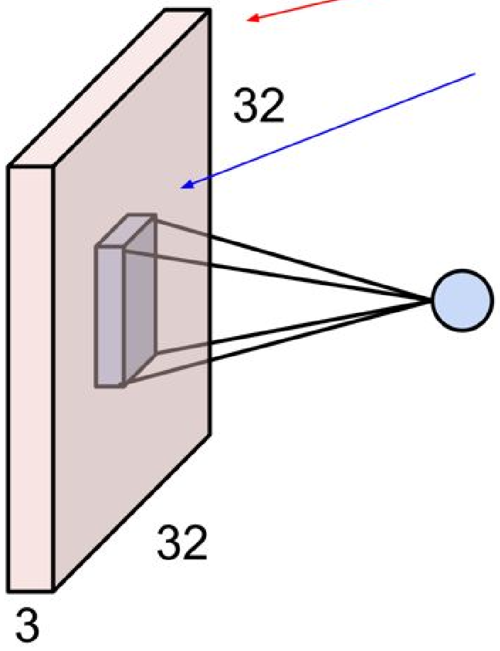


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolutional Layer

32x32x3 image

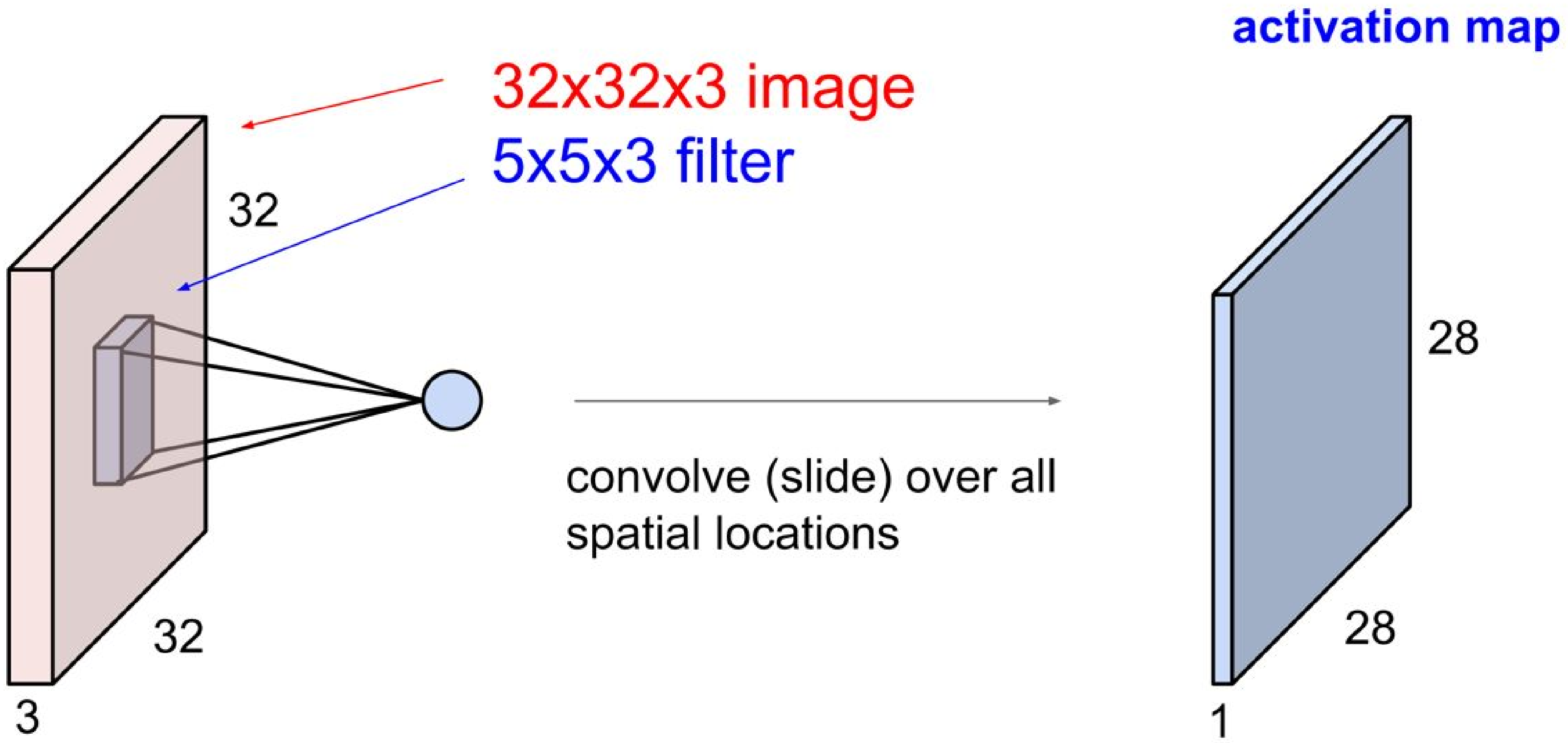
5x5x3 filter w



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolutional Layer



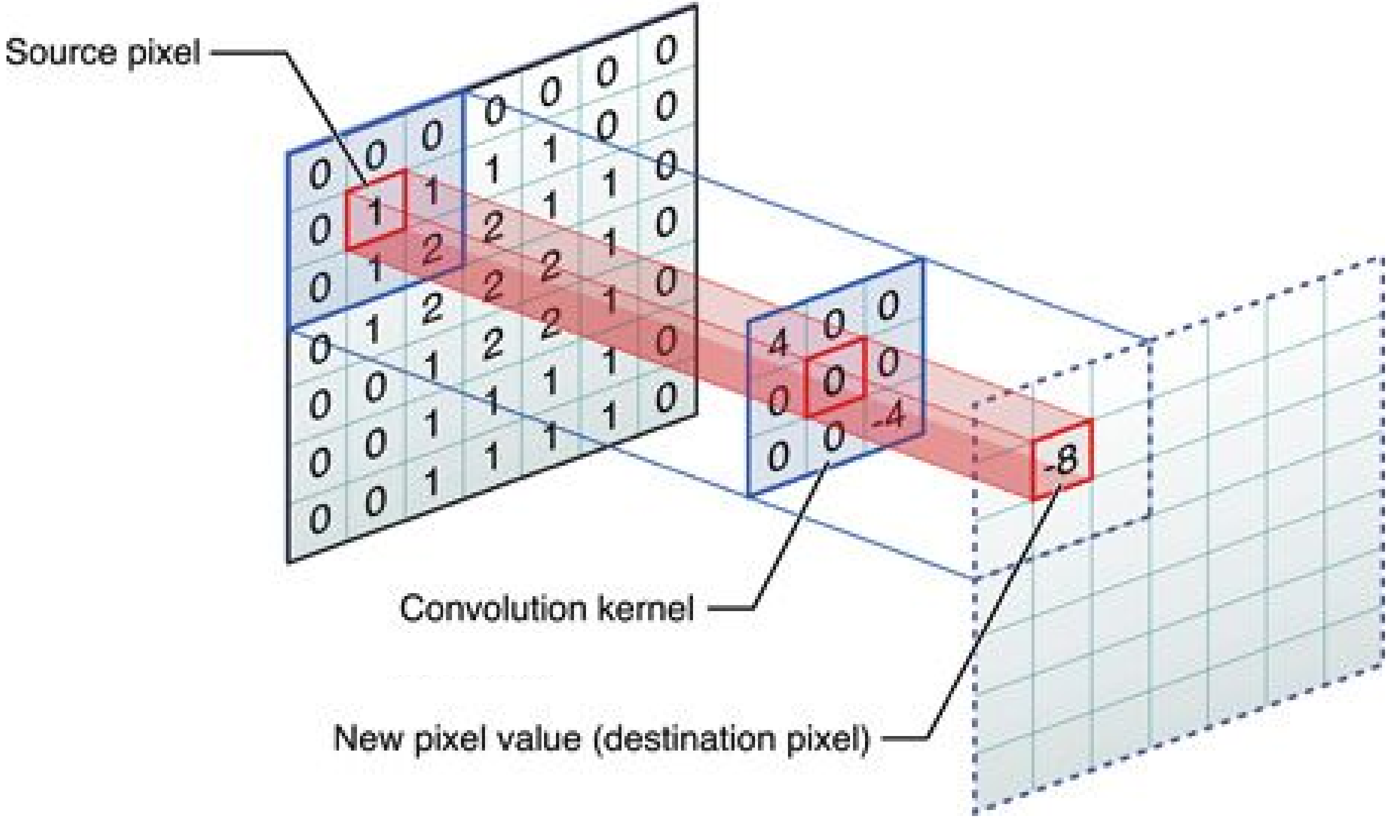
Convolution

We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

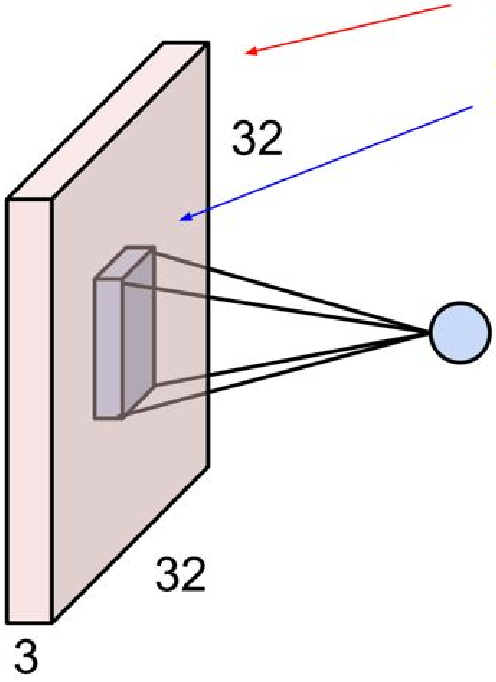
elementwise multiplication and sum of a filter and the signal (image)

Convolution

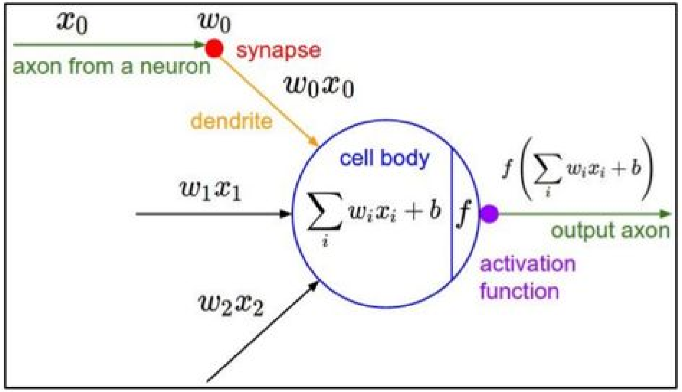


Neuron View of Convolutional Layer

32x32x3 image
5x5x3 filter

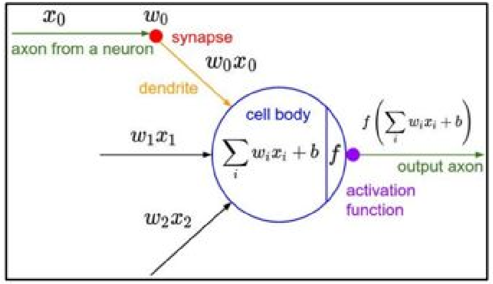


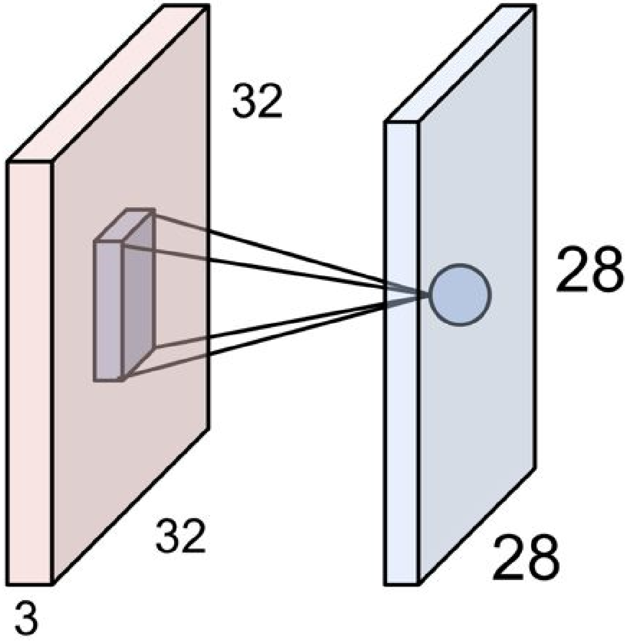
1 number:
the result of taking a dot product between
the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)



It’s just a neuron with local connectivity...

Neuron View of Convolutional Layer





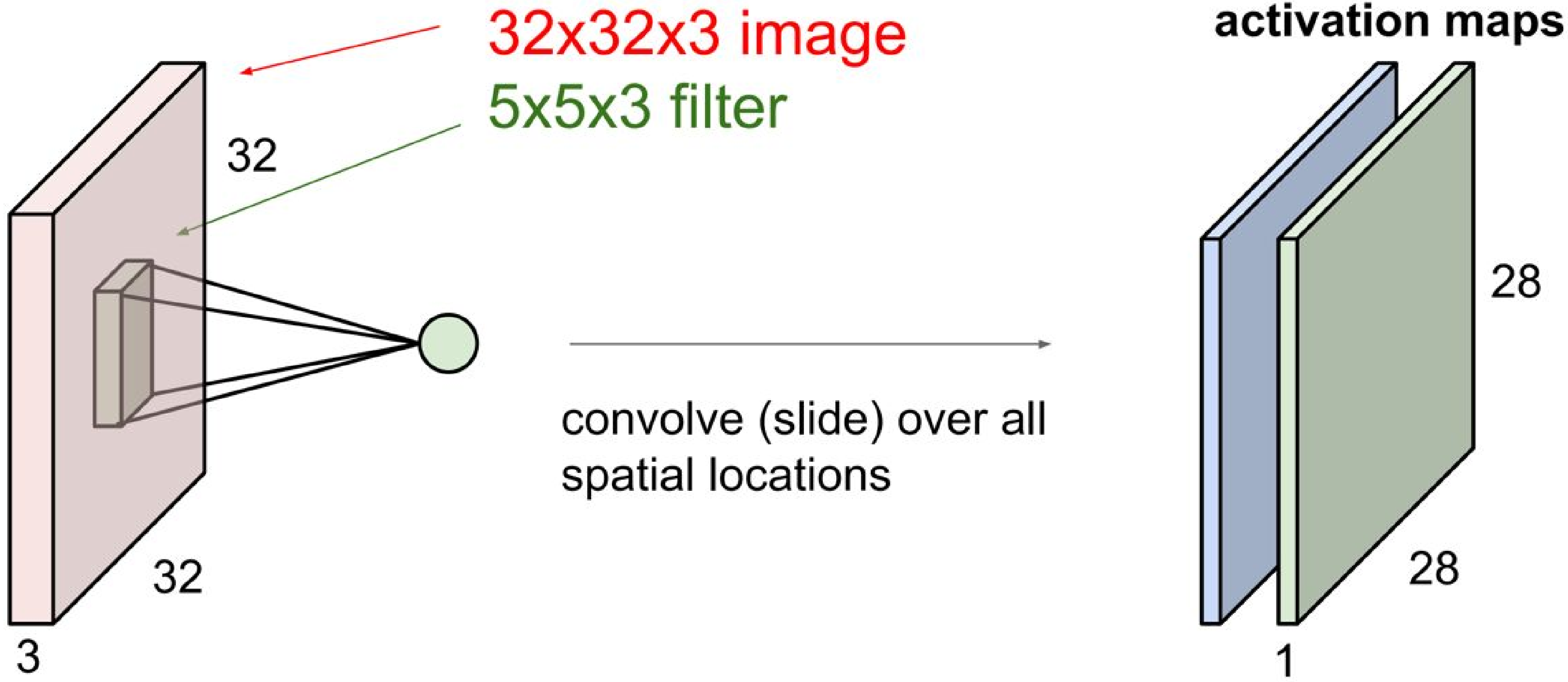
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

“5x5 filter” -> “5x5 receptive field for each neuron”

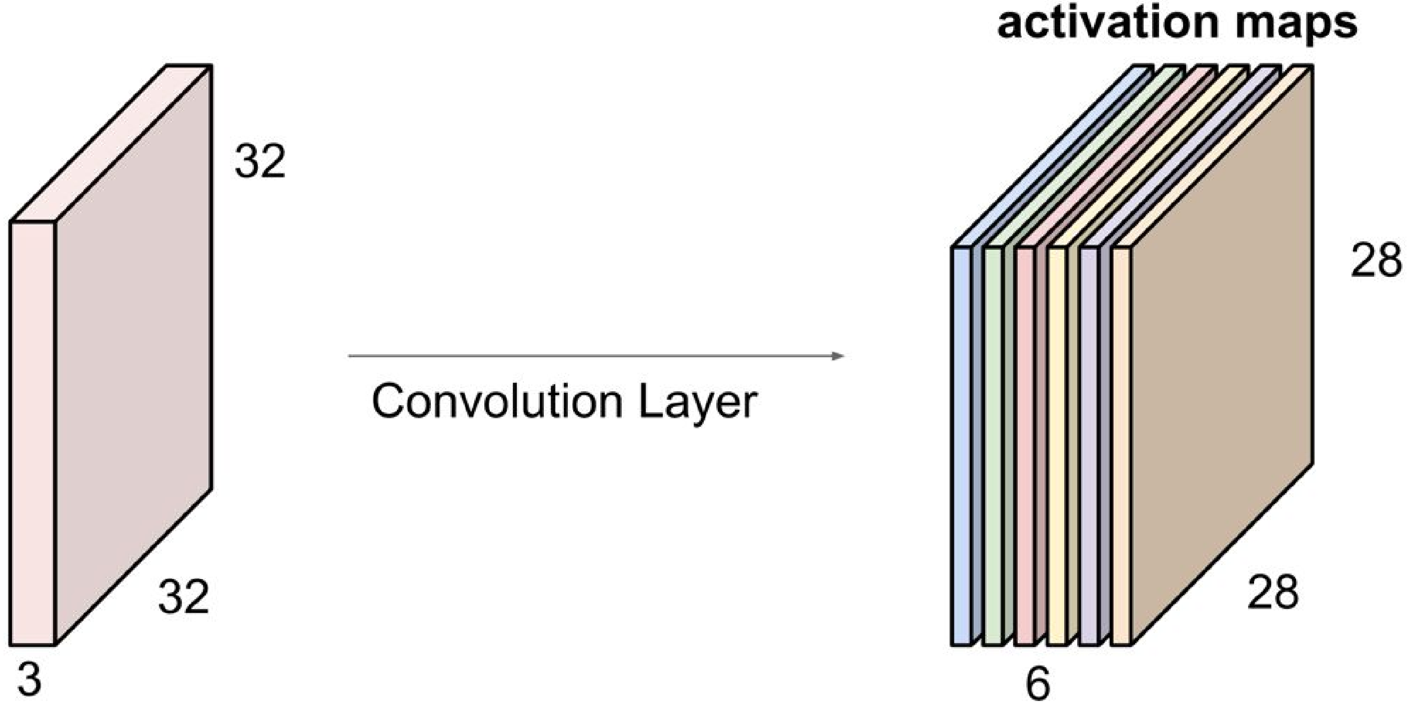
Convolutional Layer

consider a second, green filter



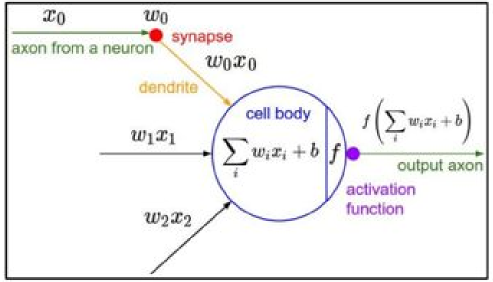
Convolutional Layer

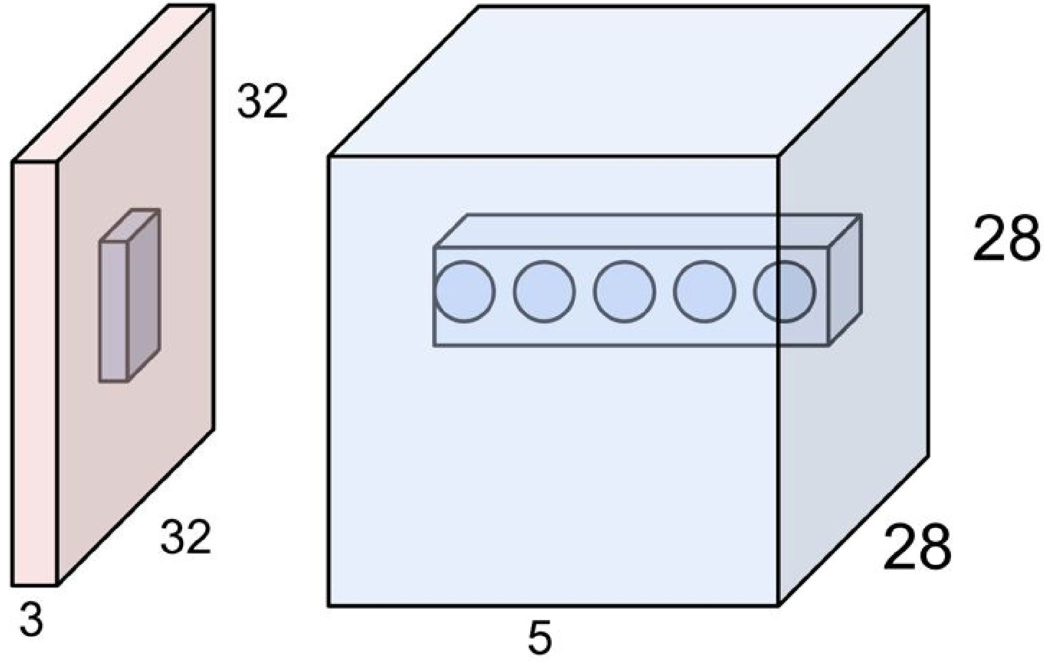
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Neuron View of Convolutional Layer



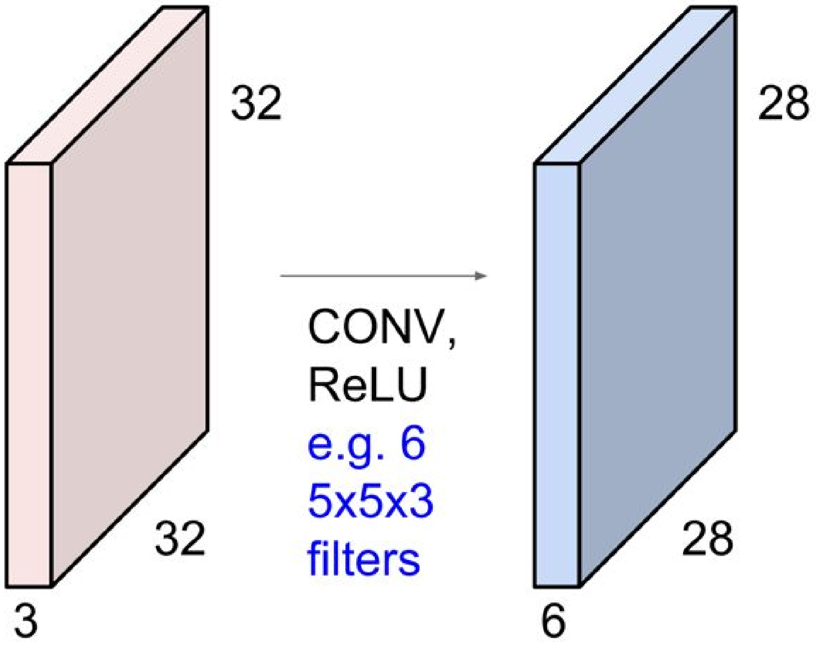


E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

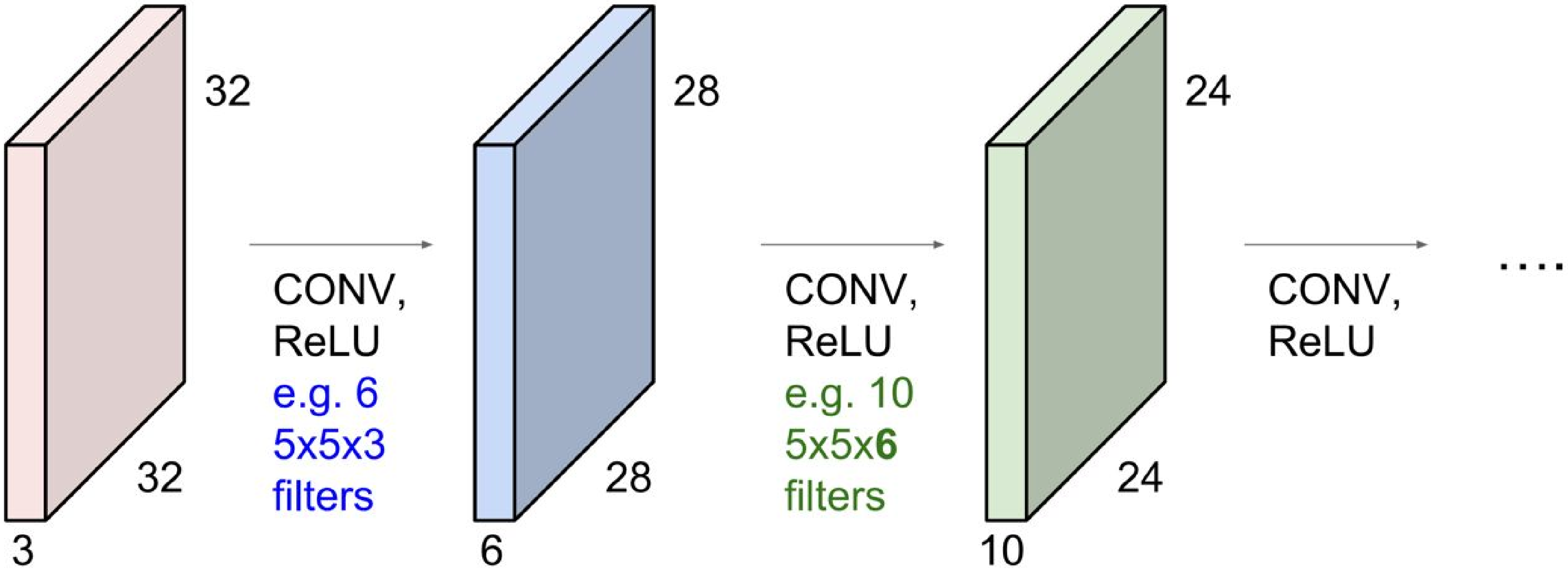
Convolutional Neural Networks

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



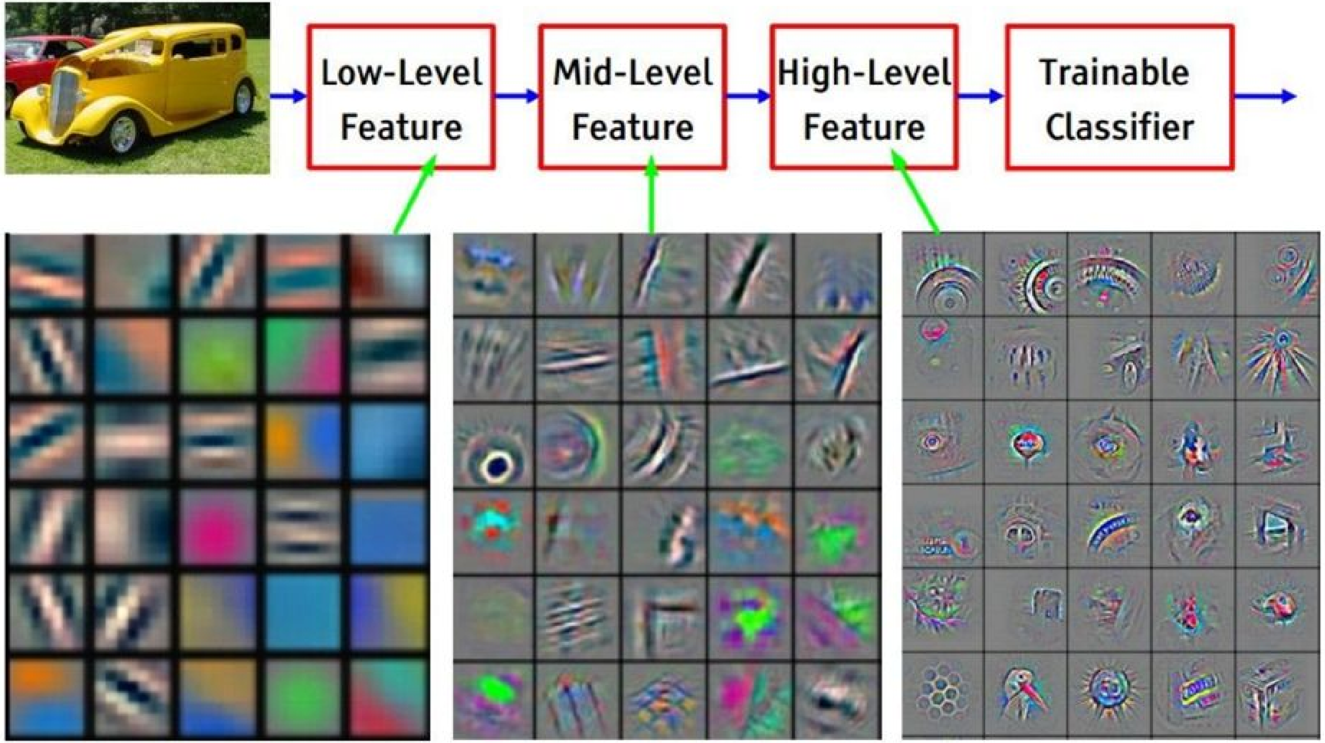
Convolutional Neural Networks

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



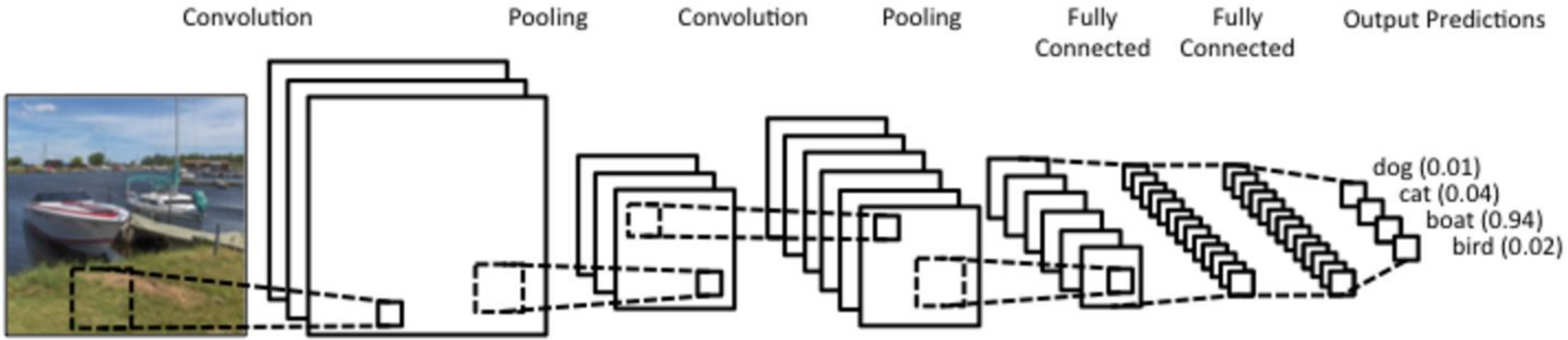
Convolutional Neural Networks

[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

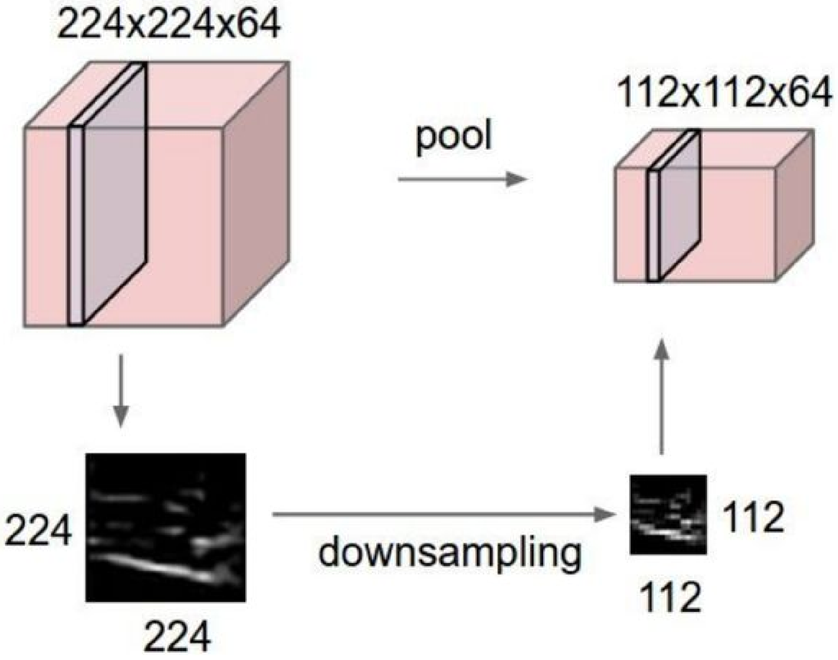
Convolutional Neural Networks



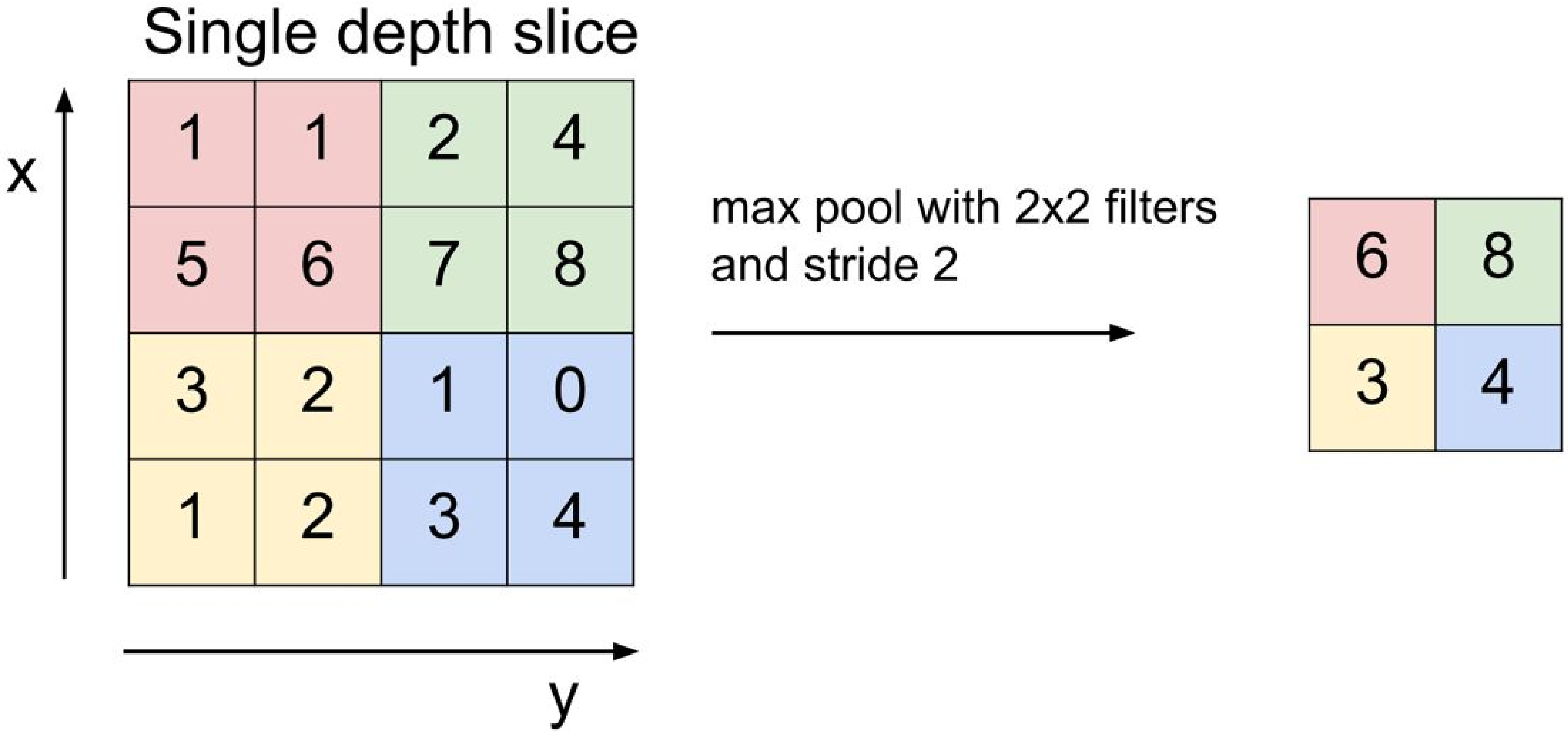
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



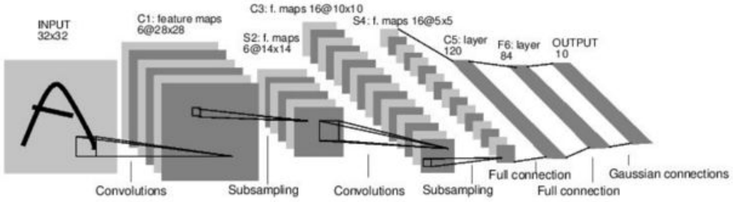
Pooling Layer (**Max Pooling** example)



History of ConvNets

1998

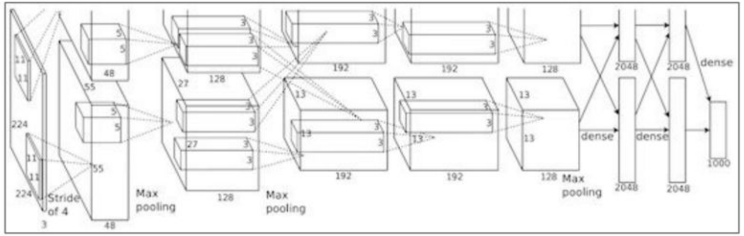
Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner]



LeNet-5

2012

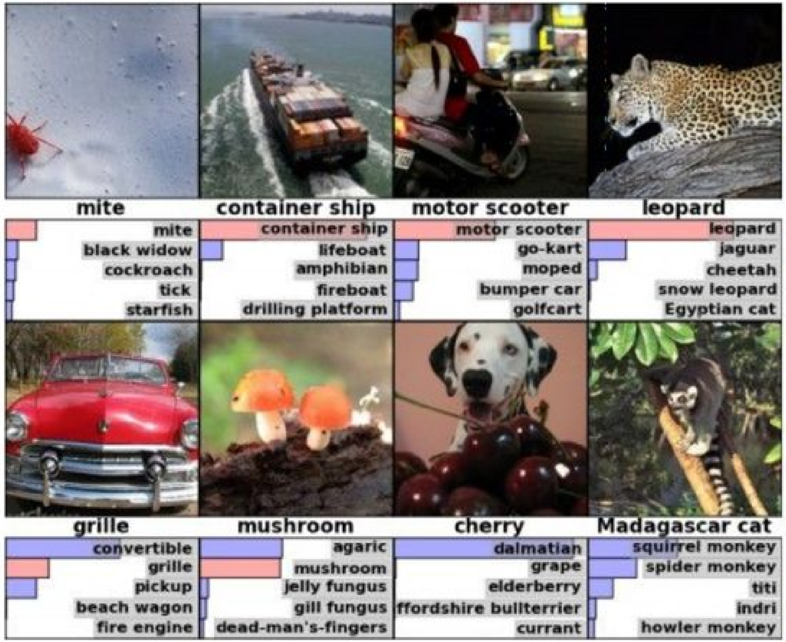
ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]



"AlexNet"

Fast-forward to today: ConvNets are everywhere

Classification

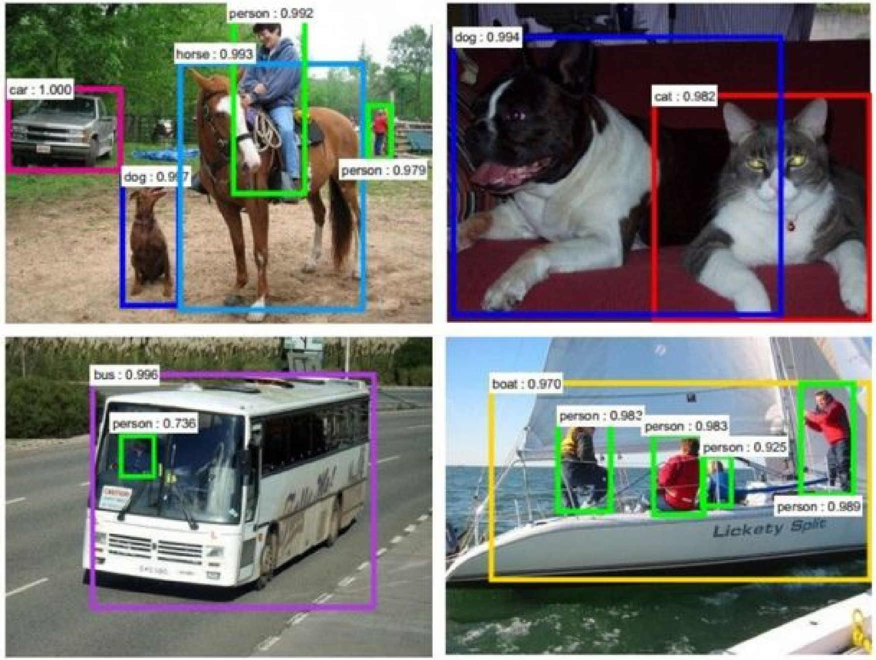


Retrieval

[Krizhevsky 2012]

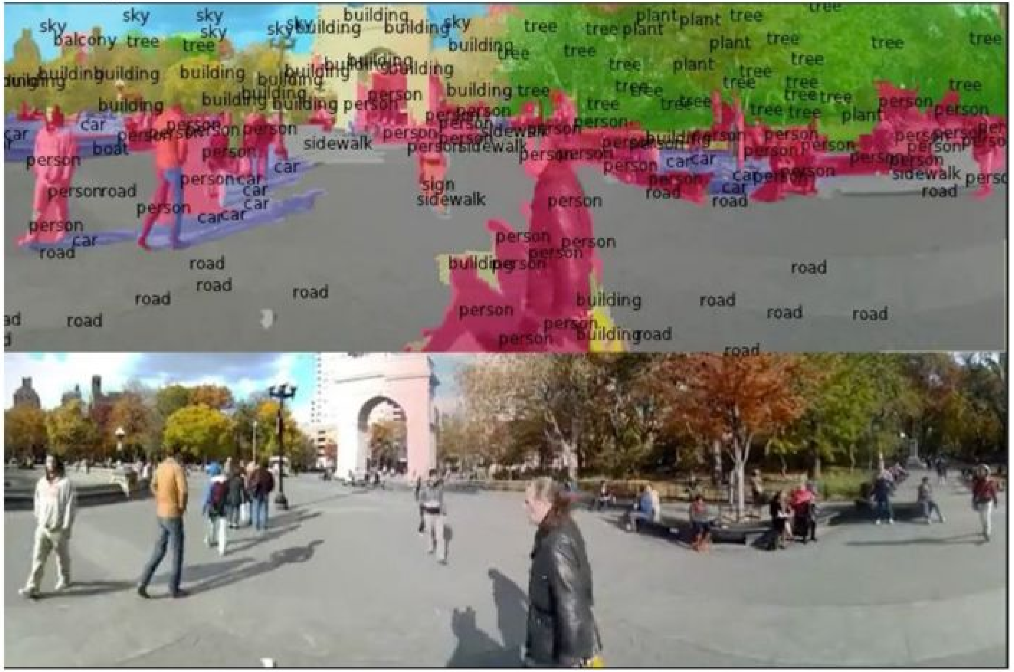
Fast-forward to today: ConvNets are everywhere

Detection



[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



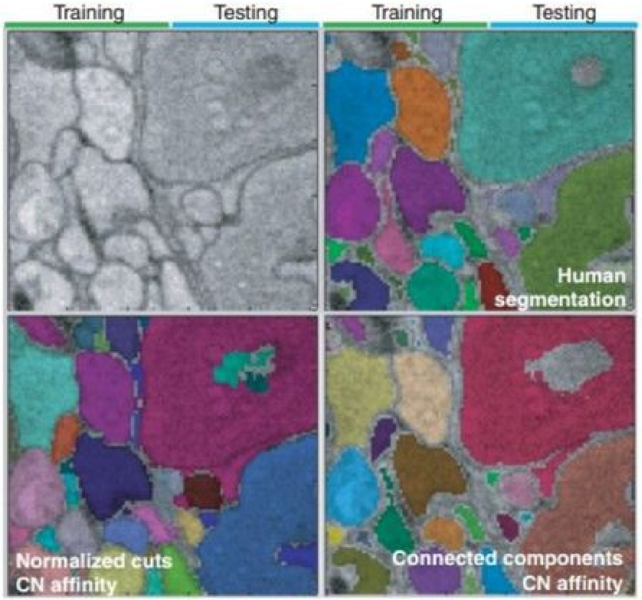
[Farabet et al., 2012]

Fast-forward to today: ConvNets are everywhere

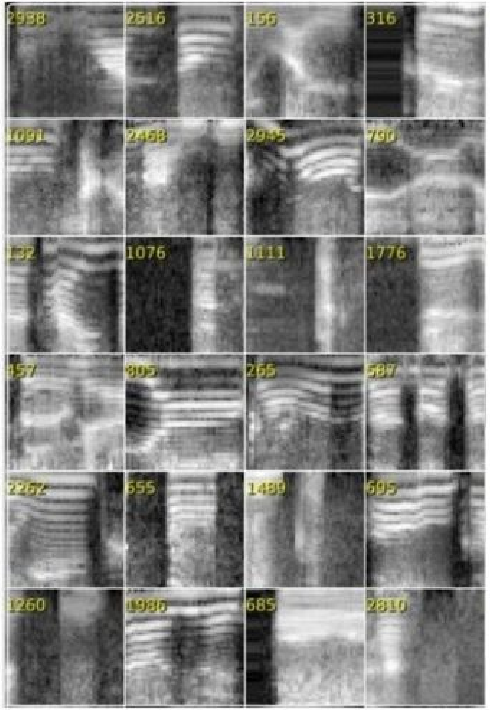
self-driving cars

NVIDIA Tegra X1

Fast-forward to today: ConvNets are everywhere



[Turaga et al., 2010]



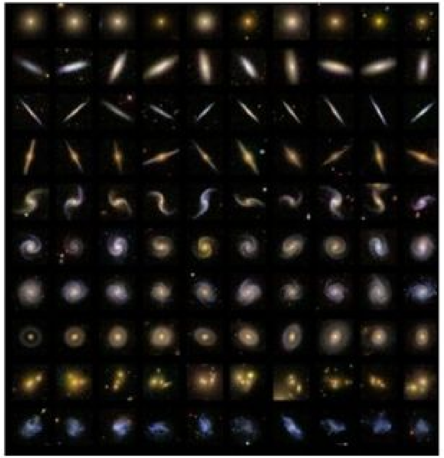
I caught this movie on the Sci-Fi channel recently. It actually turned out to be pretty decent as far as B-list horror/suspense films go. **Two guys (one naive and one loud mouthed a **) take a road trip to stop a wedding but have the worst possible luck when a maniac in a freaky, make-shift tank/truck hybrid decides to play cat-and-mouse with them.** Things are further complicated when they pick up a ridiculously whorish hitchhiker. What makes this film unique is that the combination of comedy and terror actually work in this movie, unlike so many others. The two guys are likable enough and there are some good chase/suspense scenes. Nice pacing and comic timing make this movie more than passable for the horror/slasher buff. **Definitely worth checking out.**

I just saw this on a local independent station in the New York City area. **The cast showed promise but when I saw the director, George Cosmotos, I became suspicious. And sure enough, it was every bit as bad, every bit as pointless and stupid as every George Cosmotos movie I ever saw.** He's like a stupid man's Michael Bey – with all the awfulness that accolade promises. There's no point to the conspiracy, no burning issues that urge the conspirators on. We are left to ourselves to connect the dots from one bit of graffiti on various walls in the film to the next. Thus, the current budget crisis, the war in Iraq, Islamic extremism, the fate of social security, 47 million Americans without health care, stagnating wages, and the death of the middle class are all subsumed by the sheer terror of graffiti. A truly, stunningly idiotic film.

Graphics is far from the best part of the game. **This is the number one best TH game in the series.** Next to Underground. **It deserves strong love. It is an insane game.** There are massive levels, massive unlockable characters... it's just a massive game. **Waste your money on this game. This is the kind of money that is wasted properly.** And even though graphics suck, thats doesn't make a game good. Actually, the graphics were good at the time. Today the graphics are crap. WHO CARES? As they say in Canada, This is the fun game, aye. (You get to go to Canada in THPS3) Well, I don't know if they say that, but they might. who knows. Well, Canadian people do. Wait a minute, I'm getting off topic. This game rocks. Buy it, play it, enjoy it, love it. It's PURE BRILLIANCE.

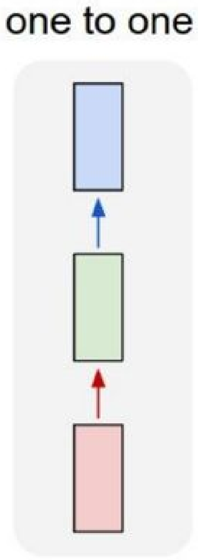
The first was good and original. I was a not bad horror/comedy movie. So I heard a second one was made and I had to watch it . What really makes this movie work is Judd Nelson's character and the sometimes clever script. **A pretty good script for a person who wrote the Final Destination films and the direction was okay.** Sometimes there's scenes where it looks like it was filmed using a home video camera with a grainy - look. Great made - for - TV movie. **It was worth the rental and probably worth buying just to get that nice eerie feeling and watch Judd Nelson's Stanley doing what he does best.** I suggest newcomers to watch the first one before watching the sequel, just so you'll have an idea what Stanley is like and get a little history background.

[Denil et al. 2014]

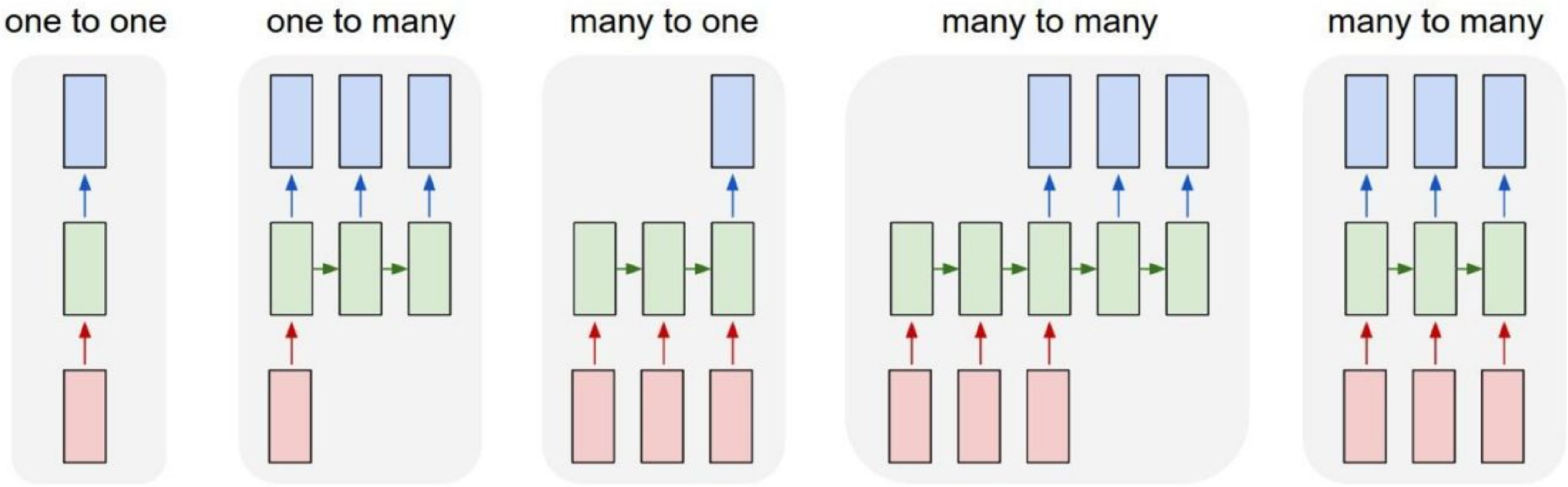


Recurrent Neural Networks

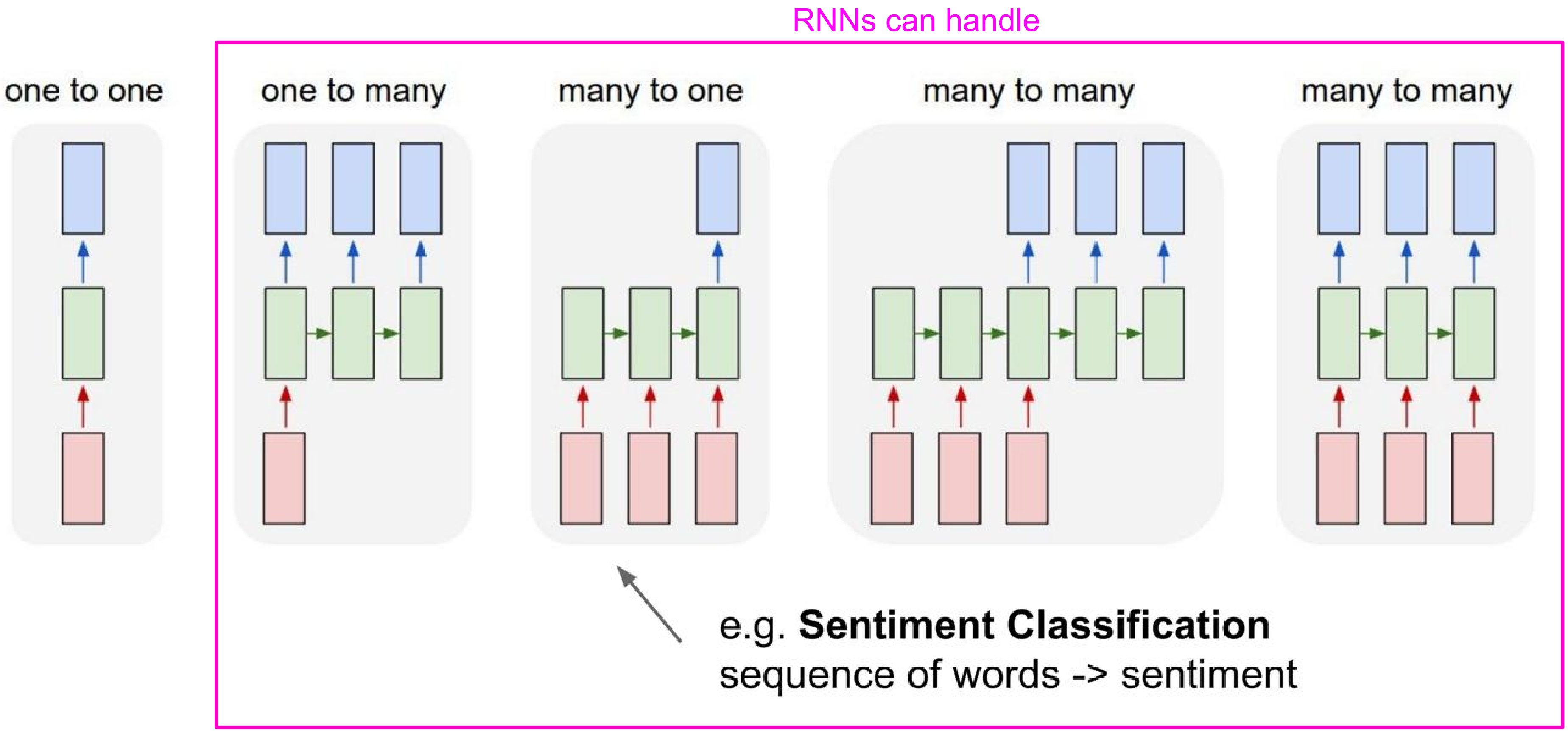
Standard "Feed-Forward" Neural Network



Standard “Feed-Forward” Neural Network

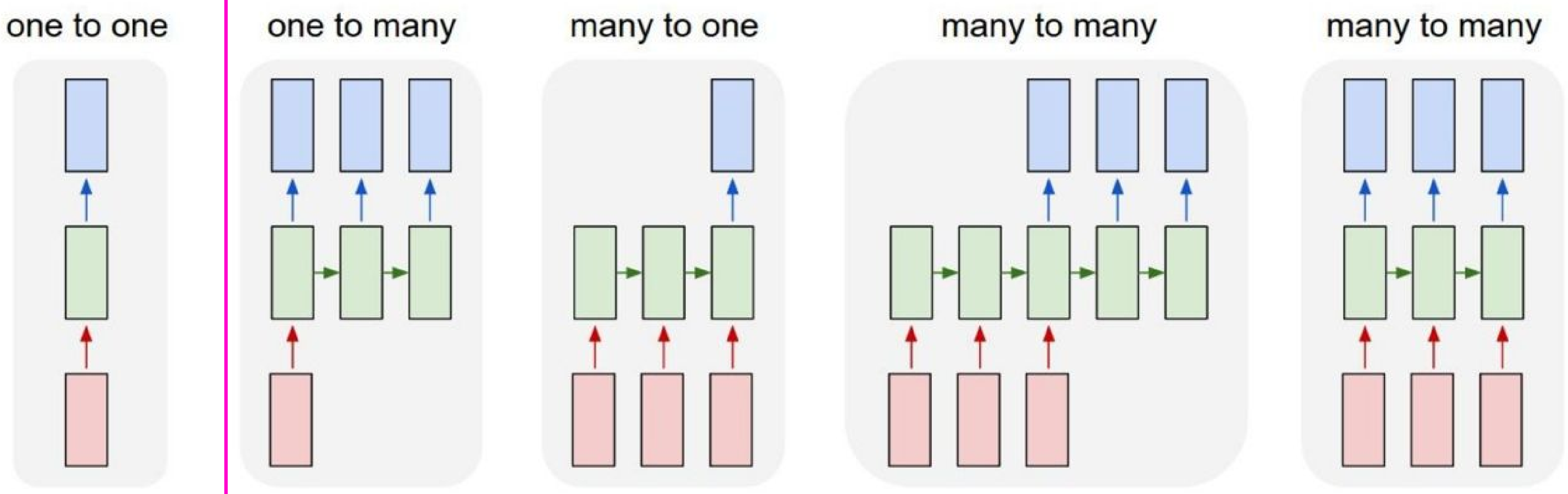


Recurrent Neural Networks (RNNs)



Recurrent Neural Networks (RNNs)

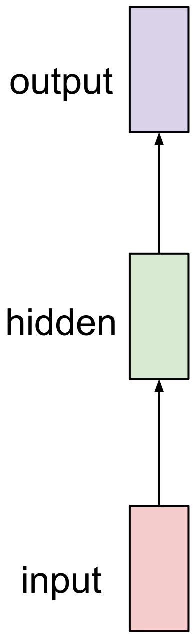
RNNs can handle



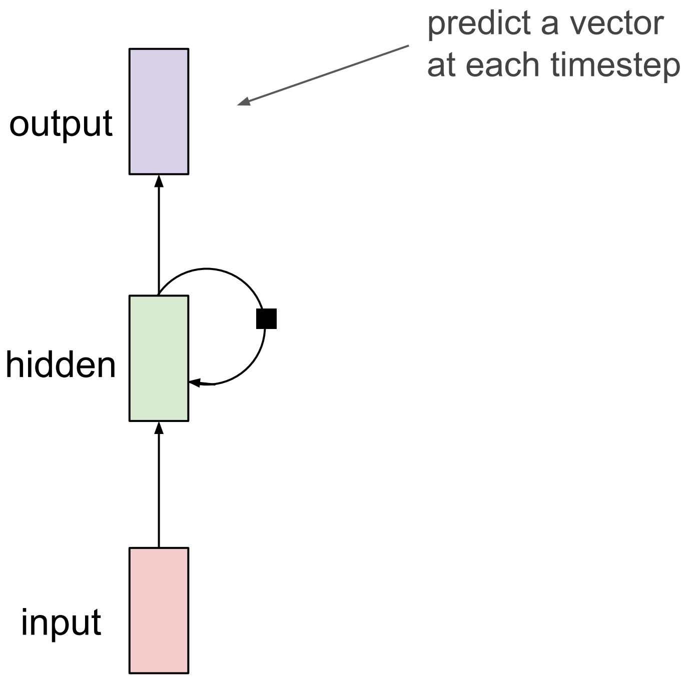
e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks (RNNs)

Traditional "Feed Forward" Neural Network



Recurrent Neural Network



Recurrent Neural Networks

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

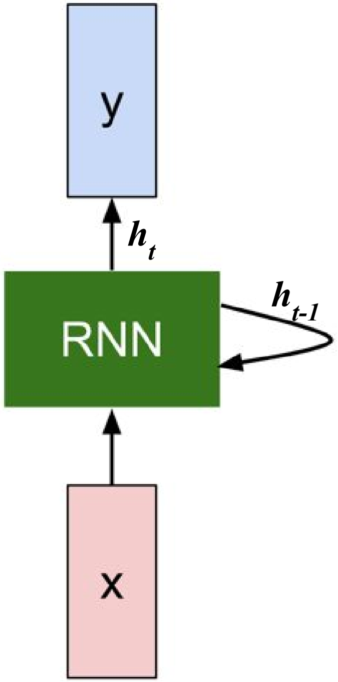
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

old state

input vector at some time step

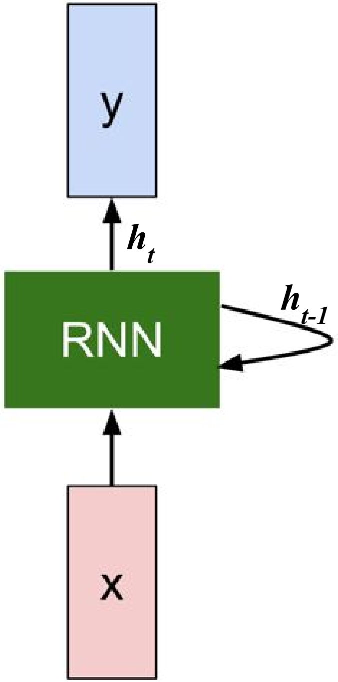


Recurrent Neural Networks

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

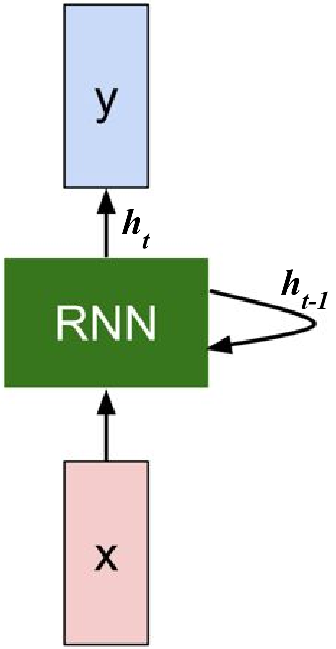
$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.



“Vanilla” Recurrent Neural Network

The state consists of a single *“hidden”* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

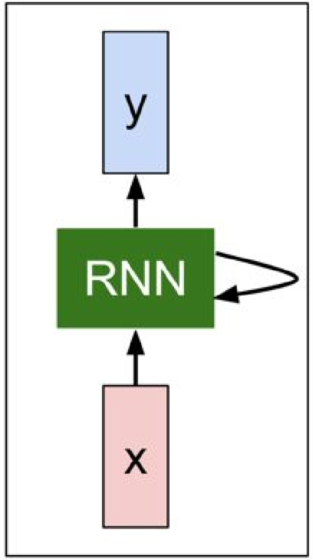
$$y_t = W_{hy}h_t$$

Character Level Language Model with an RNN

Character-level language model example

Vocabulary:
[h,e,l,o]

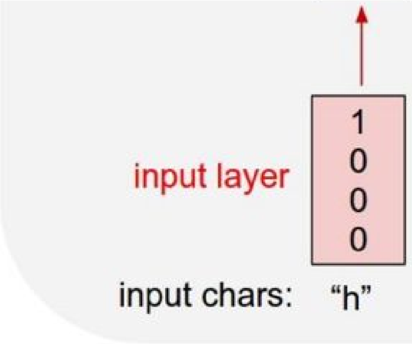
Example training sequence:
“hello”



Character Level Language Model with an RNN

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

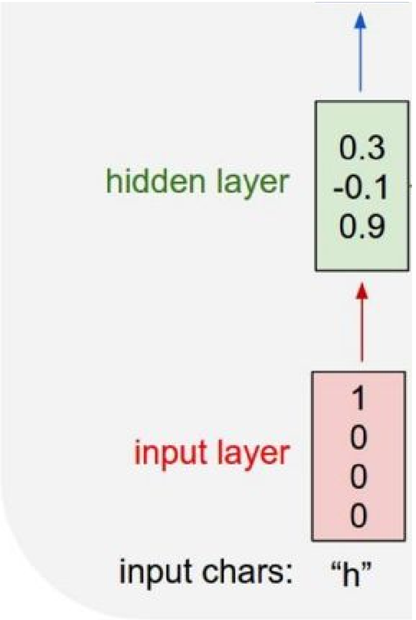


Character Level Language Model with an RNN

$$\boxed{h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)}$$

Vocabulary:
[h,e,l,o]

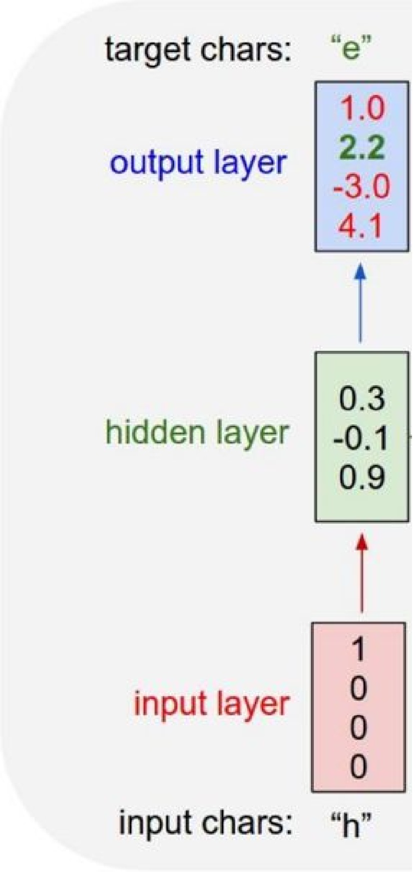
Example training sequence:
“hello”



Character Level Language Model with an RNN

Vocabulary:
[h,e,l,o]

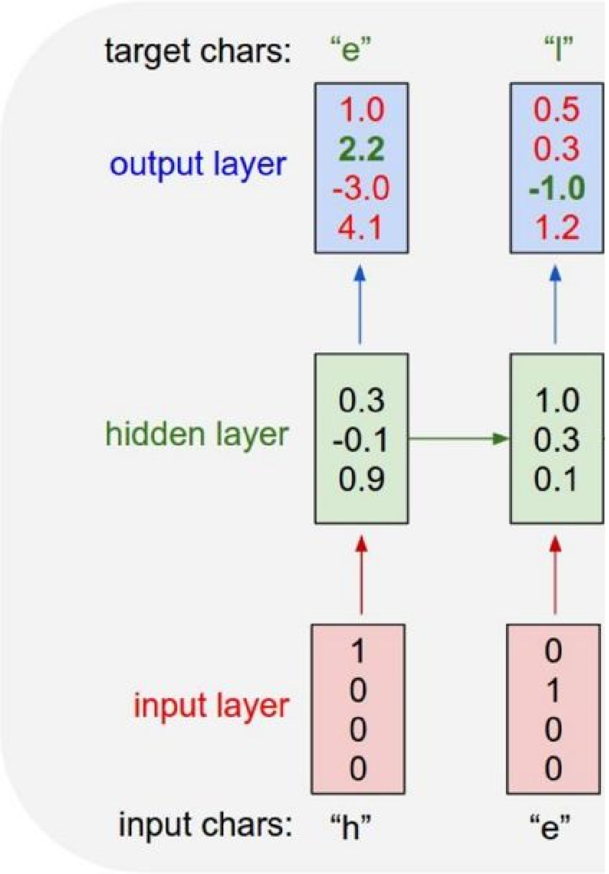
Example training sequence:
“hello”



Character Level Language Model with an RNN

Vocabulary:
[h,e,l,o]

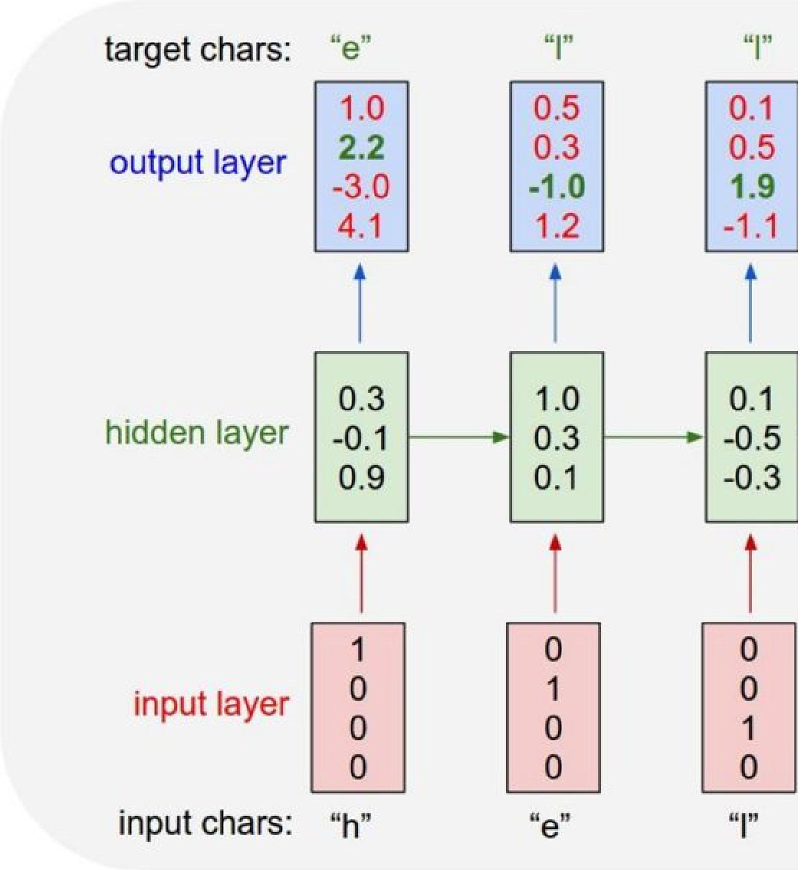
Example training sequence:
“hello”



Character Level Language Model with an RNN

Vocabulary:
[h,e,l,o]

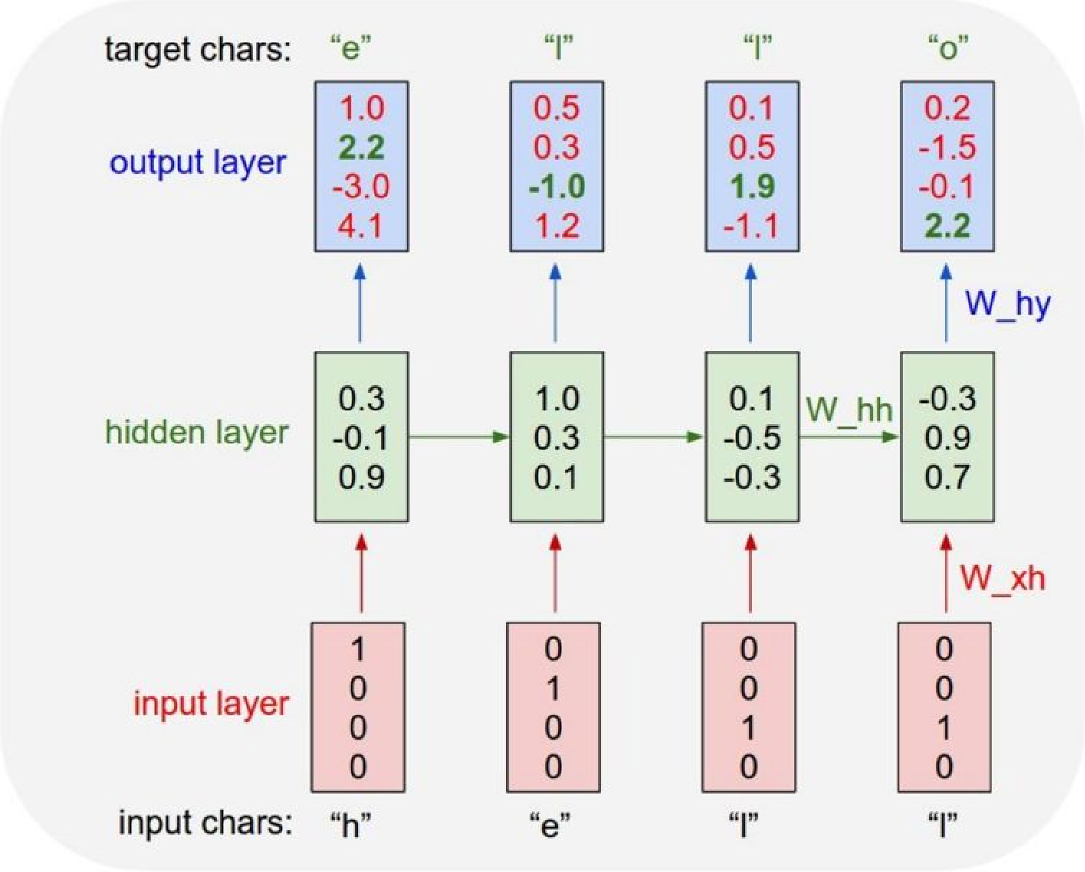
Example training sequence:
“hello”



Character Level Language Model with an RNN

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



Example: Generating Shakespeare with RNNs

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

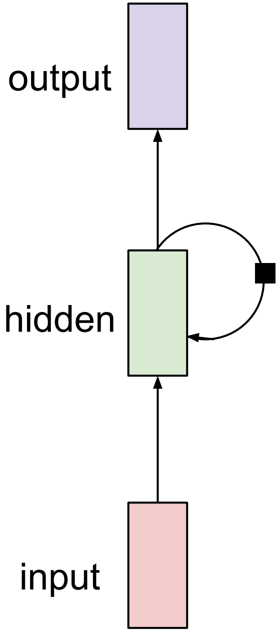
Example: Generating C Code with RNNs

```c
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Long Short Term Memory Networks (LSTMs)

Recurrent networks suffer from the “vanishing gradient problem”

- Aren’t able to model long term dependencies in sequences

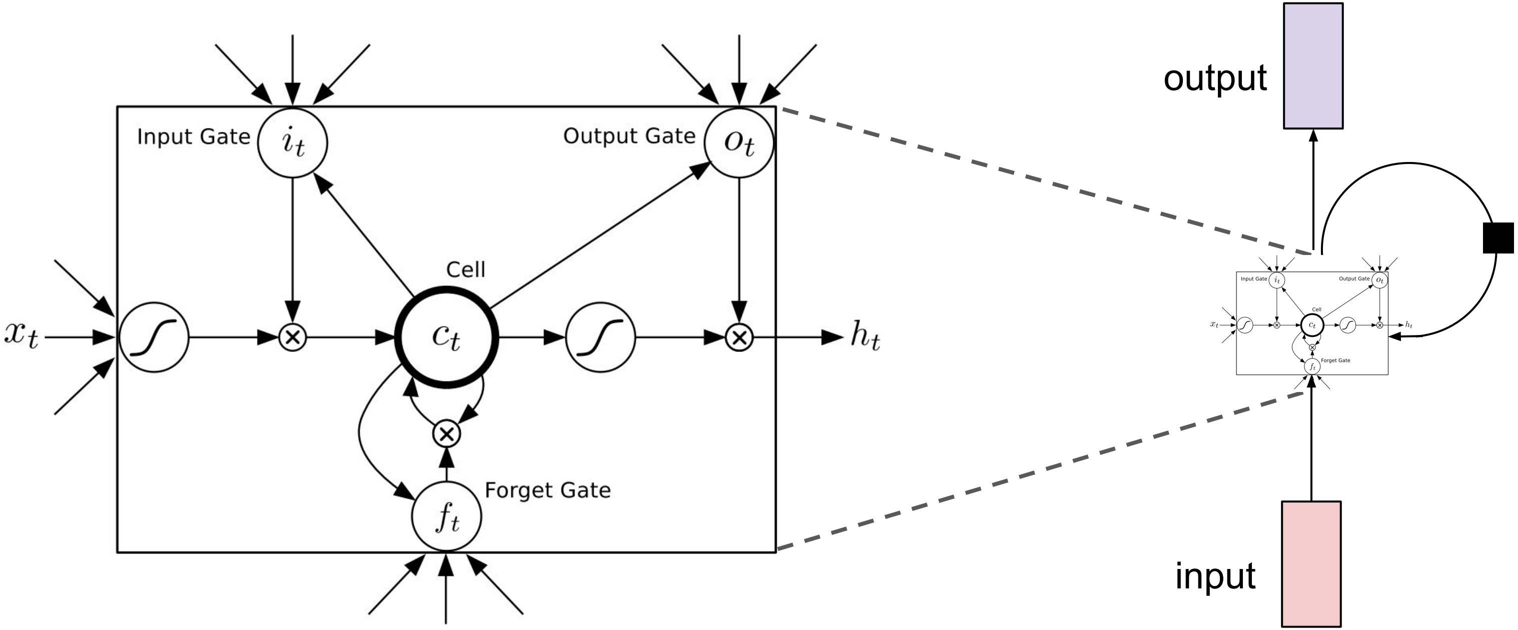


Long Short Term Memory Networks (LSTMs)

Recurrent networks suffer from the “vanishing gradient problem”

- Aren’t able to model long term dependencies in sequences

Use “gating units” to learn when to remember



RNNs and CNNs Together