Neural Network Function Approximation

Map of RL Algorithms

Model-Based

Model-Free

Learn Q	π_θ
SARSA	Policy Gradient
Q-learning	

On Policy

Off-Policy

MLMBTRL
(Learn T,R)

Tabular

This Time

Challenges in Reinforcement Learning:

- Exploration vs Exploitation
- Credit Assignment
- Generalization

Function Approximation

Function Approximation

$$y \approx f_\theta(x)$$

Function Approximation

input

$$y \approx f_\theta(x)$$

Function Approximation

input

$$y \approx f_\theta(x)$$

output

Function Approximation

input

$$y \approx f_\theta(x)$$

output

parameters

Function Approximation

input

$$y \approx f_\theta(x)$$

output parameters

Previously, Linear:

$$f_\theta(x) = \theta^\top \beta(x)$$

Function Approximation

input

$$y \approx f_\theta(x)$$

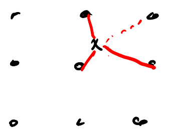
output parameters

Previously, Linear:

$$f_\theta(x) = \theta^\top \beta(x)$$

weights features

interpolation
weight



Function Approximation

input

$$y \approx f_\theta(x)$$

output

parameters

Previously, Linear:

$$f_\theta(x) = \theta^\top \beta(x)$$

weights

features

e.g. $\beta_i(x) = \sin(i\,\pi\,x)$

AI = Neural Nets

Neural Nets are just another function approximator

Neural Network

Neural Network

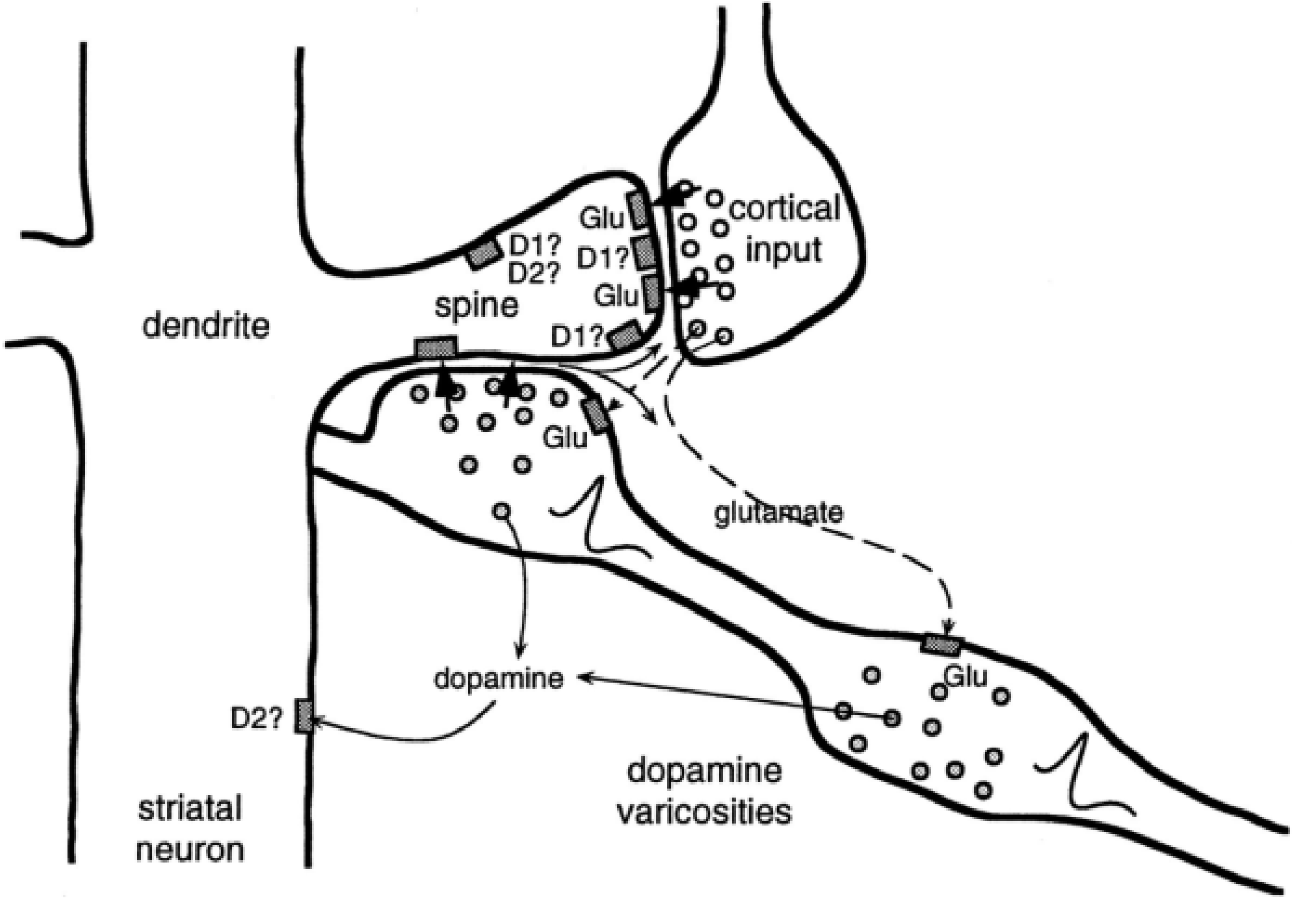
nonlinearity

$$h(x) = \sigma(Wx + b)$$

$$W^{(2)} \sigma\left(W^{(1)} x + b^{(1)}\right) + b^{(2)}$$

Neural Network

$$h(x) = \sigma(Wx + b)$$



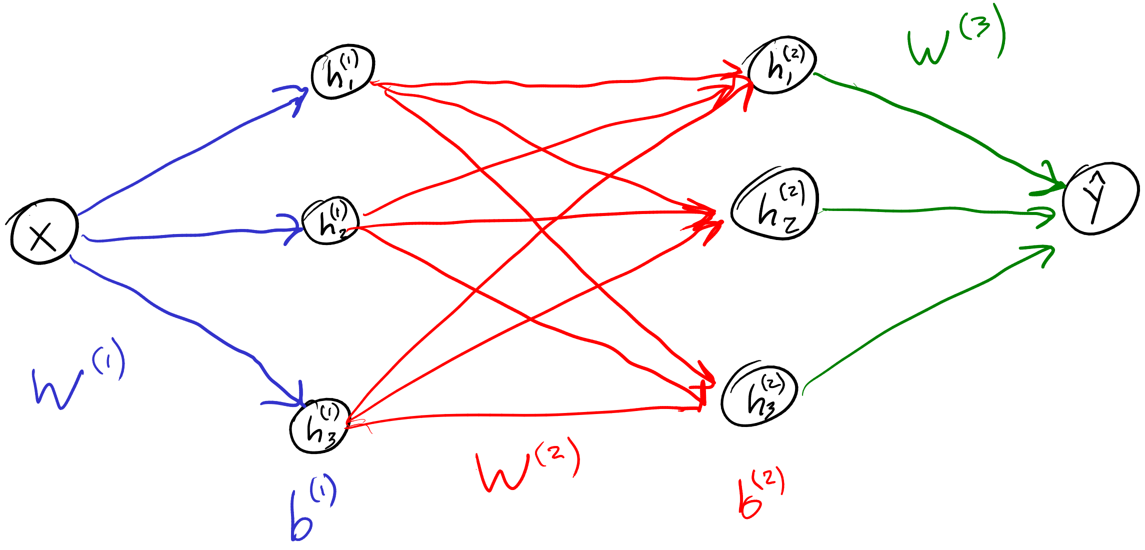
Neural Network

Neural Network

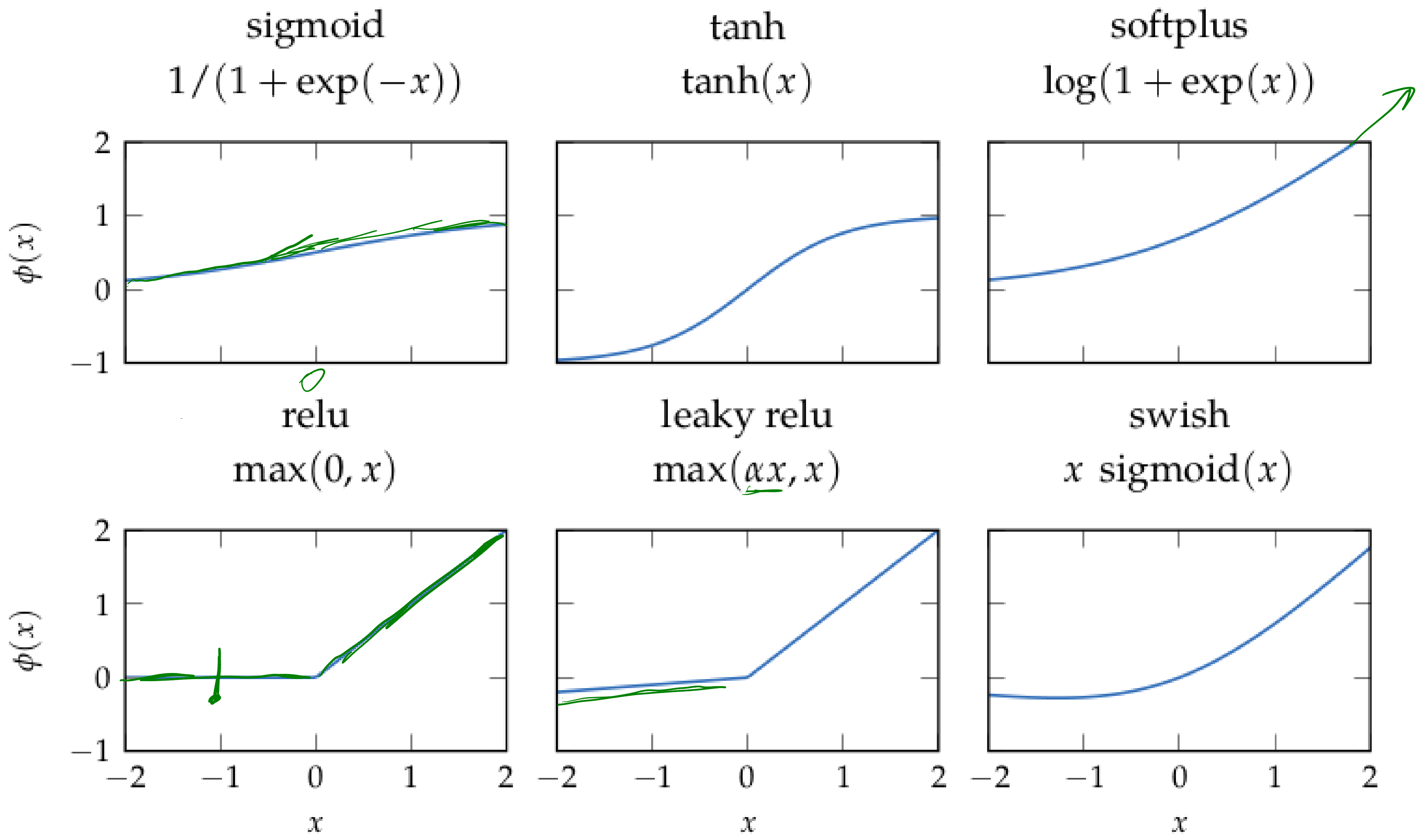
$$\theta = \left(W^{(3)}, W^{(2)}, W^{(1)}, b^{(2)}, b^{(1)}\right)$$

$$h(x) = \sigma(Wx + b)$$

$$f_\theta(x) = h^{(2)}\left(h^{(1)}(x)\right) = W^{(3)} \sigma^{(2)}\left(W^{(2)} \sigma^{(1)}\left(W^{(1)} x + b^{(1)}\right) + b^{(2)}\right)$$

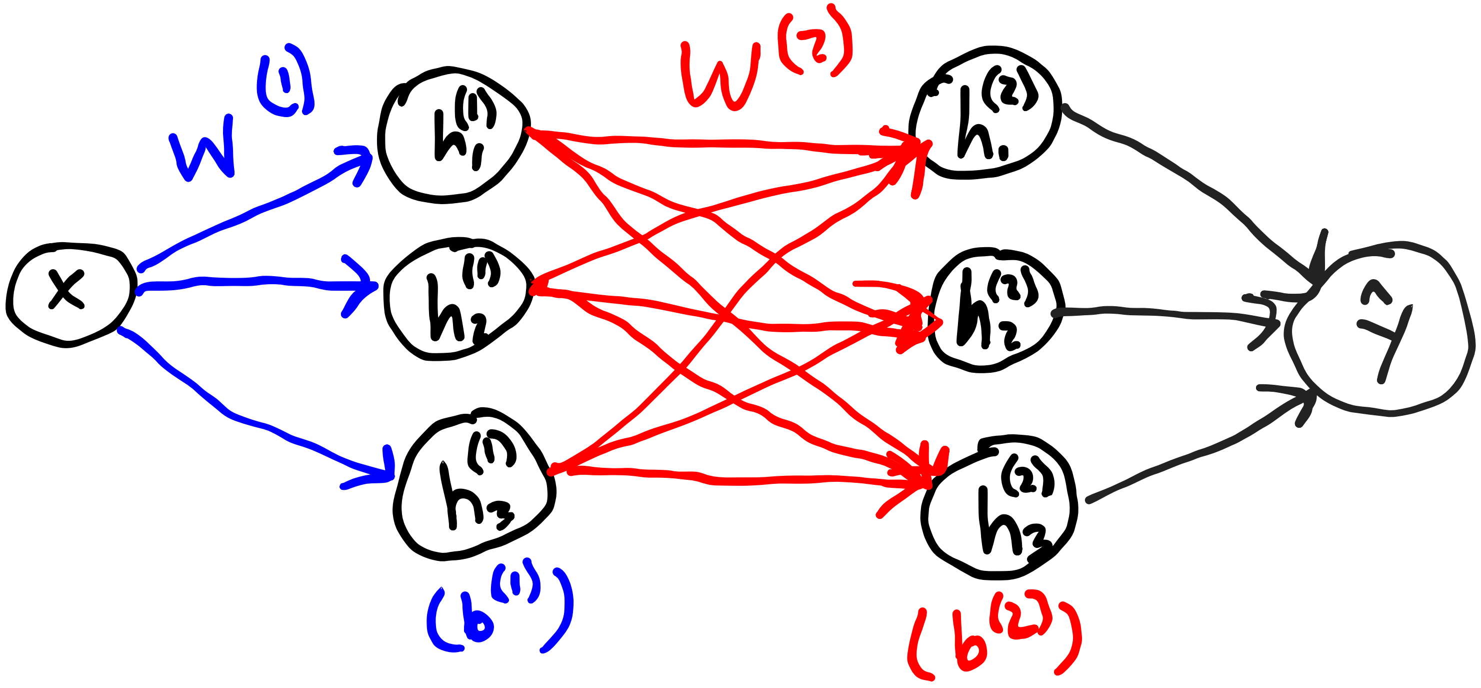


Nonlinearities

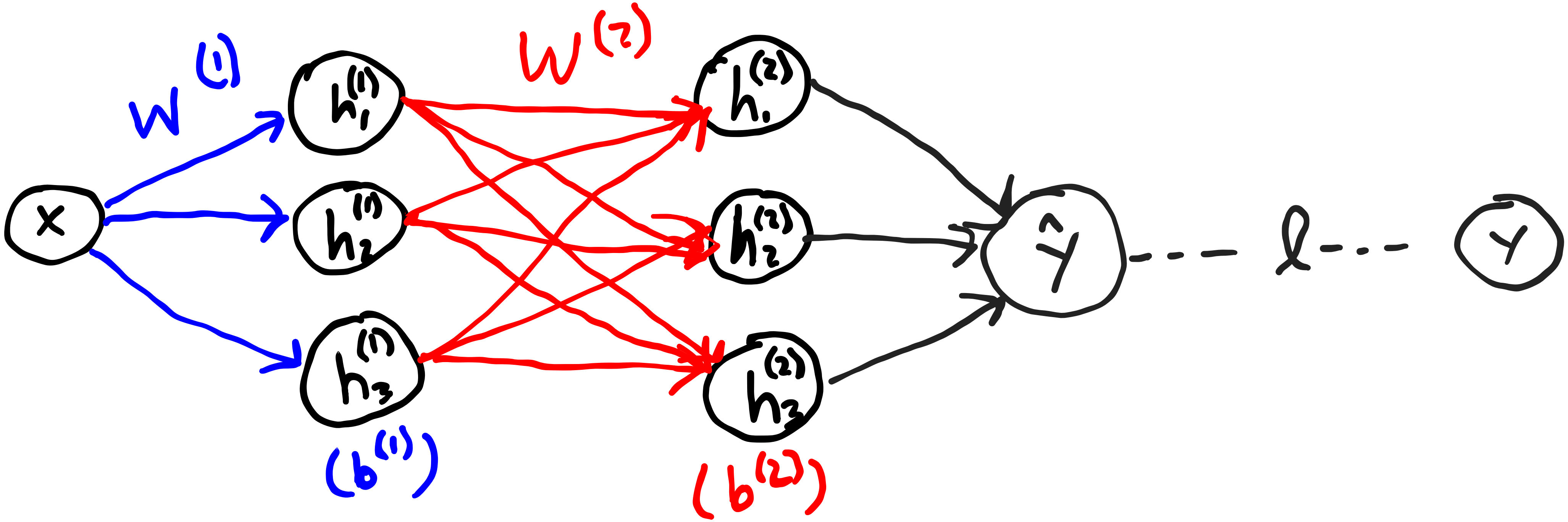


Training

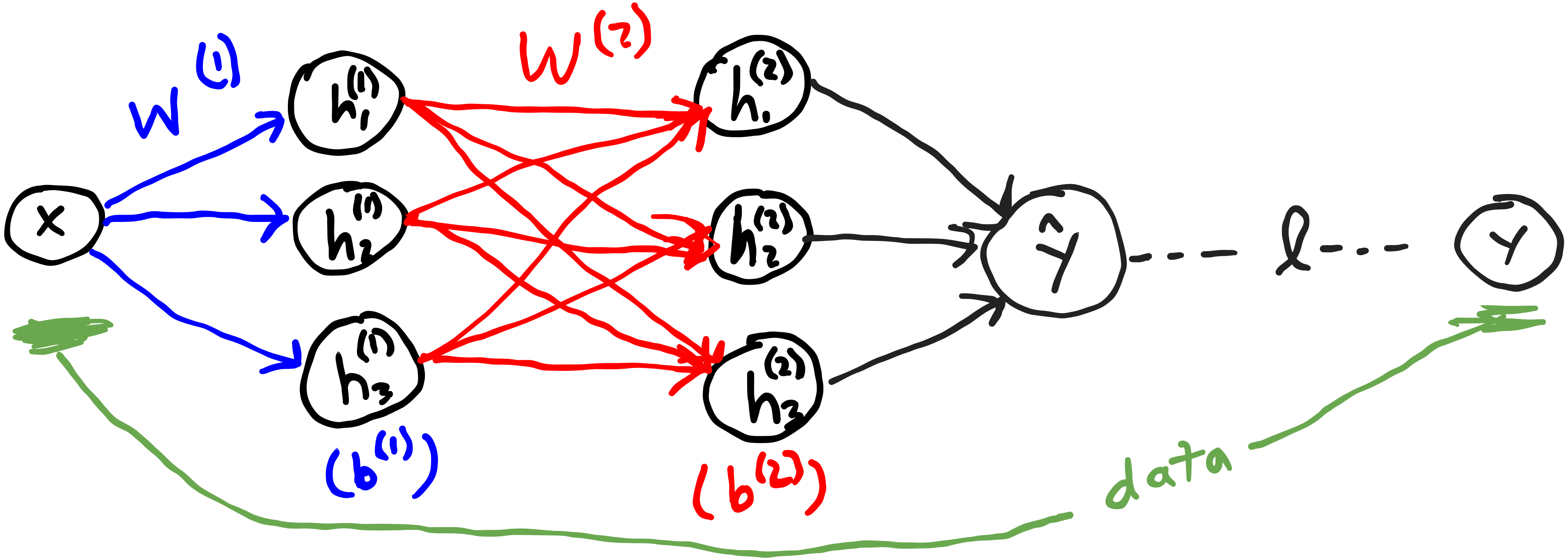
Training



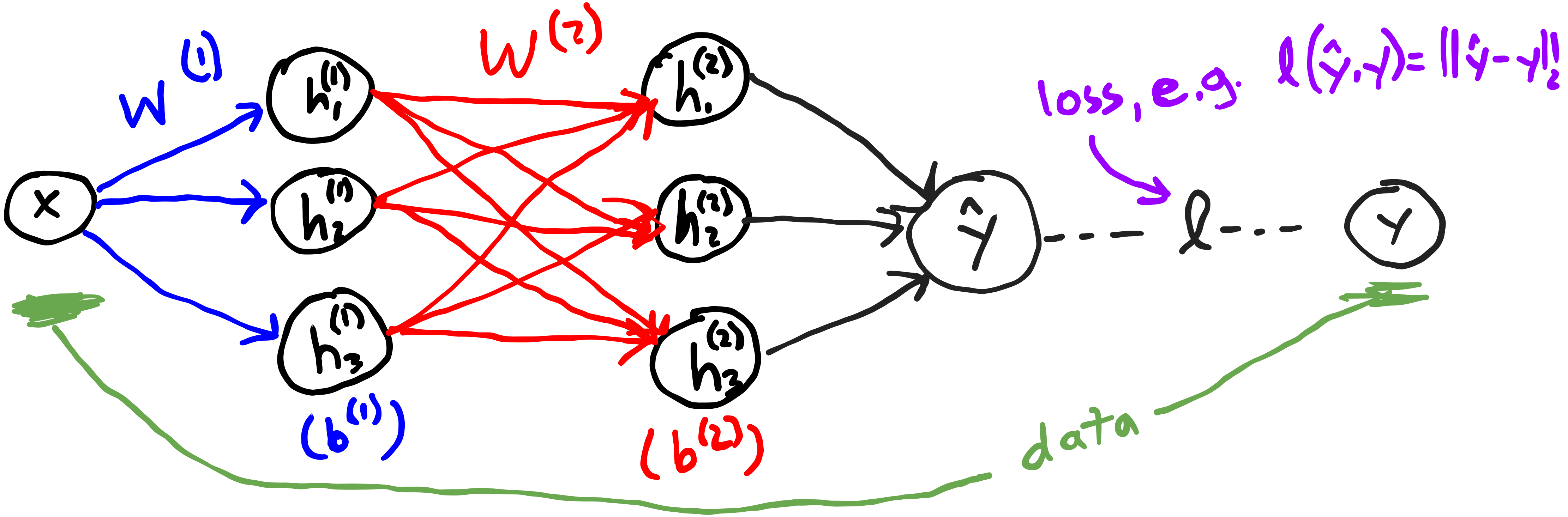
Training



Training

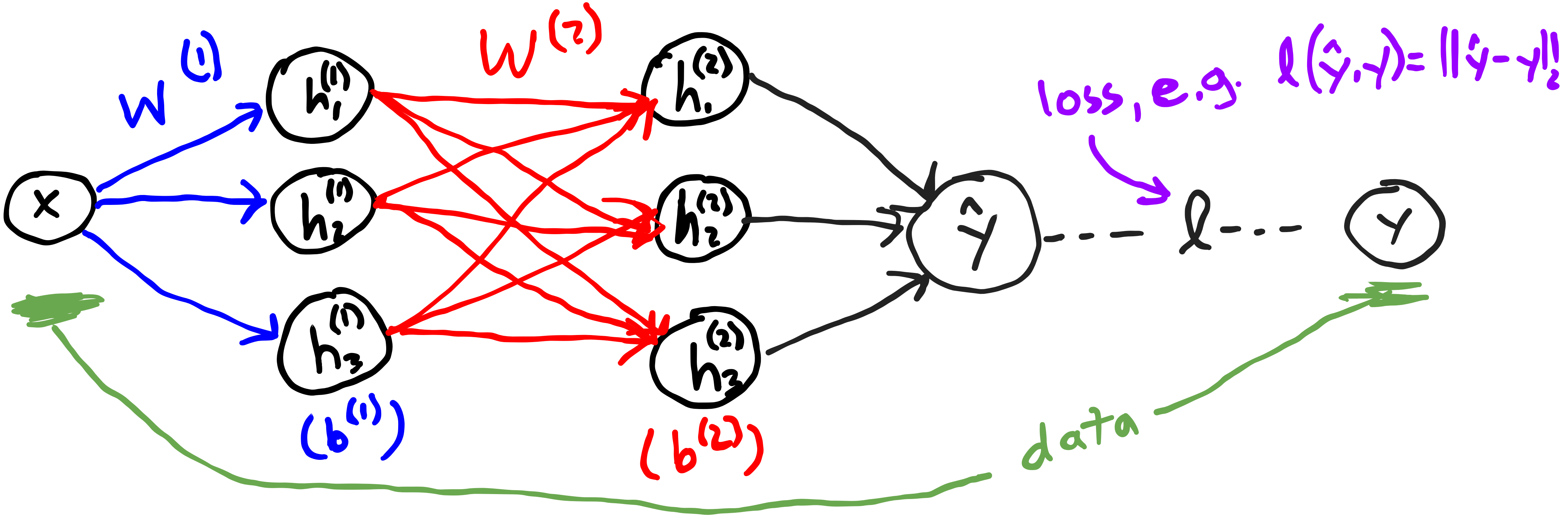


Training



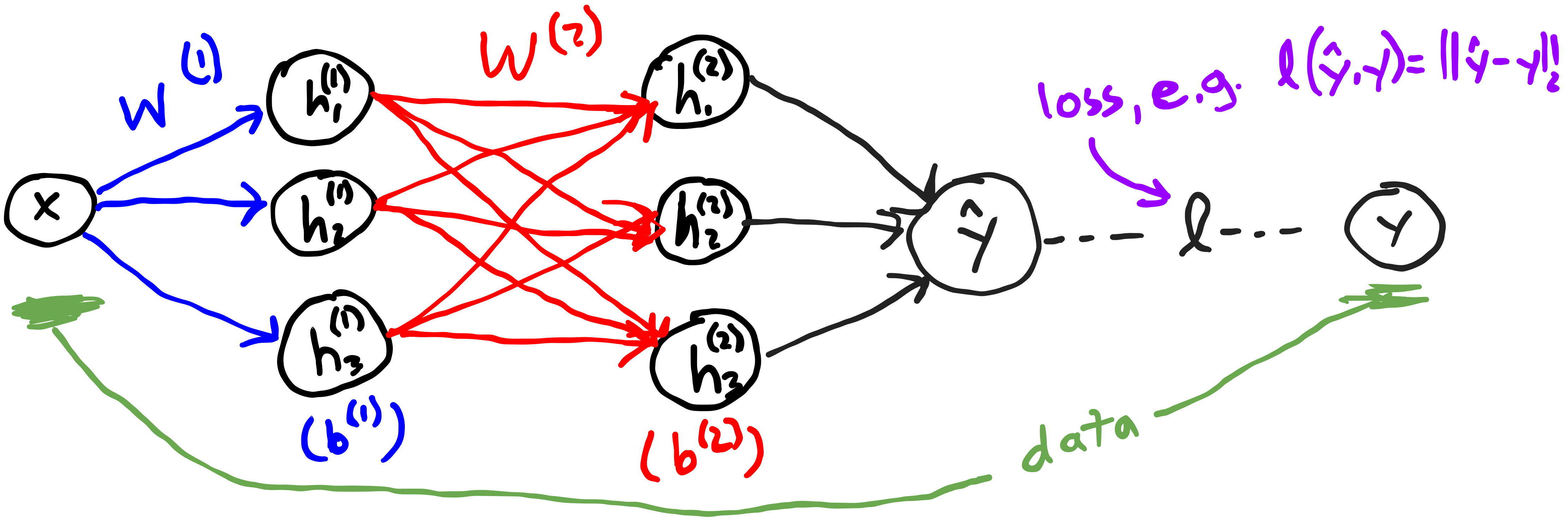
$W^{(1)}$
$W^{(2)}$
x
$h_1^{(1)}$
$h_2^{(1)}$
$h_3^{(1)}$
$(b^{(1)})$
$h_1^{(2)}$
$h_2^{(2)}$
$h_3^{(2)}$
$(b^{(2)})$
$\hat{y}$
ℓ
y
loss, e.g. $\ell(\hat{y}, y) = \|\hat{y} - y\|_2^2$
data

Training



$$\theta^* = \arg\min_\theta \sum_{(x,y)\in\mathcal{D}} l(f_\theta(x), y)$$

Training



$$\theta^* = \arg\min_\theta \sum_{(x,y)\in\mathcal{D}} l(f_\theta(x), y)$$

Stochastic Gradient Descent: $\theta \leftarrow \theta - \alpha \nabla_\theta l(f_\theta(x), y)$

Chain Rule

$$f \circ g \circ h = f(g(h(x)))$$

$$\nabla_\theta \, \ell(f_\theta(x), y)$$

$$\theta = (W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)})$$

$$\frac{\partial f(g(h(x)))}{\partial x}\bigg|_{x_0} = \frac{\partial f(g(h))}{\partial h}\bigg|_{h_0} \frac{\partial h(x)}{\partial x}\bigg|_{x_0} = \frac{\partial f(g)}{\partial g}\bigg|_{g_0} \frac{\partial g(h)}{\partial h}\bigg|_{h_0} \frac{\partial h(x)}{\partial x}\bigg|_{x_0}$$

$$\hat{y} = \overbrace{W^{(2)} \sigma(W^{(1)} x + b^{(1)}) + b^{(2)}}^{f_\theta(x)}$$

$$\ell(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y} - y)^2$$

$$\frac{\partial \ell}{\partial \hat{y}} = -\frac{1}{n} 2(\hat{y} - y)$$

$$\frac{\partial \ell}{\partial W^{(2)}}\bigg|_{x_0} = \frac{\partial \ell}{\partial \hat{y}}\bigg|_{\hat{y}_0} \left(\frac{\partial \hat{y}}{\partial W^{(2)}}\right)\bigg|_{x_0} = \frac{\partial \ell}{\partial \hat{y}}\bigg|_{\hat{y}_0} \left(\sigma(W^{(1)} x_0 + b^{(1)})\right)$$

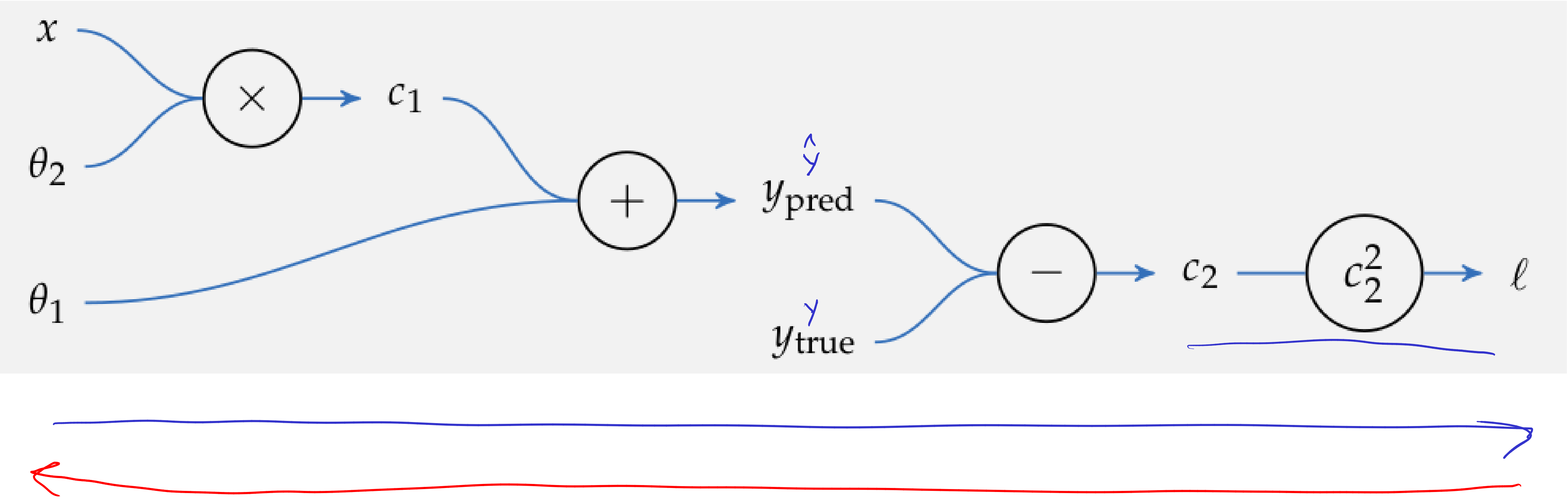
SGD

$$W^{(2)} \leftarrow W^{(2)} - \alpha \widehat{\frac{\partial \ell}{\partial W^{(2)}}}$$

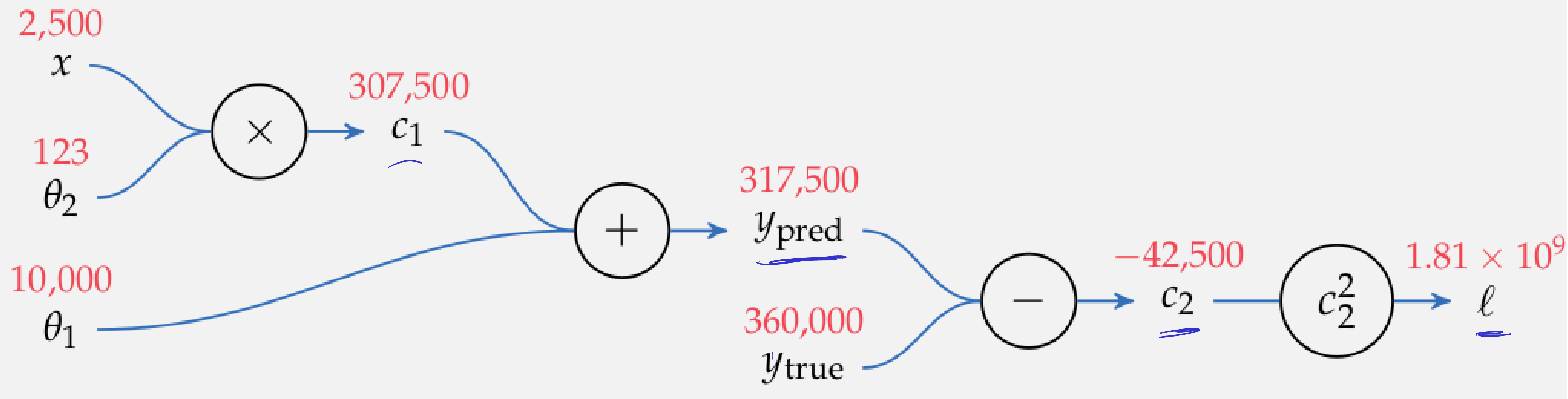
Backprop

Backprop

$\hat{y}$ $\theta_2 x + \theta_1$ $\ell(\hat{y}, y) = (\hat{y} - y)^2$

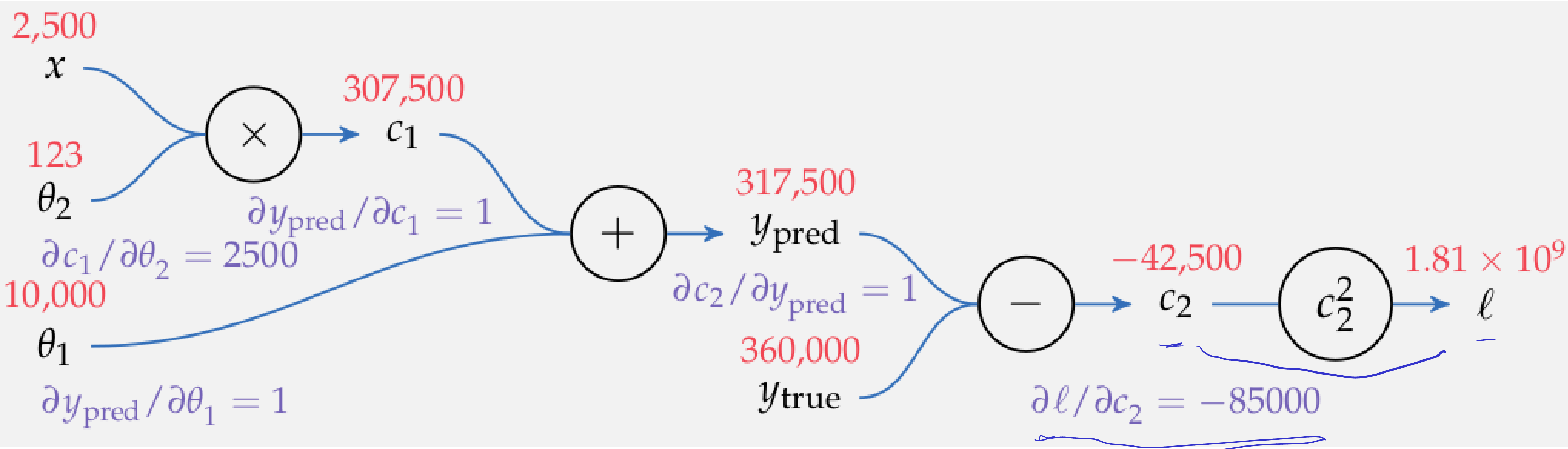


Backprop



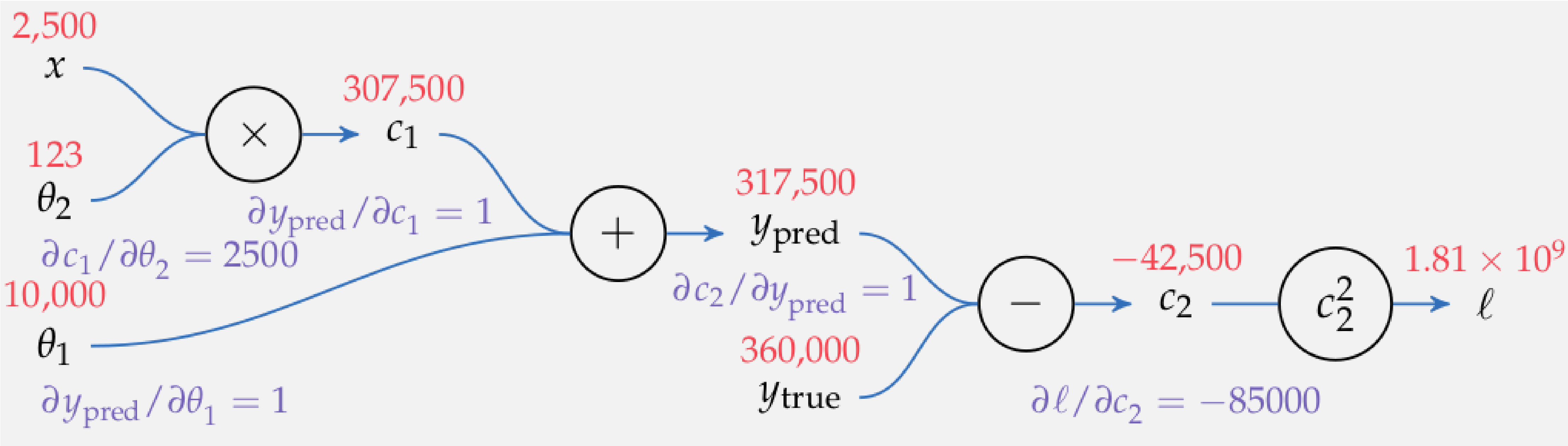
2,500
x
123
θ_2
10,000
θ_1
$\times$
307,500
c_1
$+$
317,500
y_{pred}
360,000
y_{true}
$-$
$-42{,}500$
c_2
c_2^2
1.81×10^9
ℓ

Backprop



$\frac{\partial \ell}{\partial c_2} = 2c_2$

Backprop



$$\frac{\partial \ell}{\partial \theta_1} = \frac{\partial \ell}{\partial c_2}\frac{\partial c_2}{\partial y_{\text{pred}}}\frac{\partial y_{\text{pred}}}{\partial \theta_1} = -85{,}000 \cdot 1 \cdot 1 = -85{,}000$$

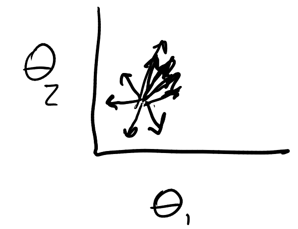
$$\frac{\partial \ell}{\partial \theta_2} = \frac{\partial \ell}{\partial c_2}\frac{\partial c_2}{\partial y_{\text{pred}}}\frac{\partial y_{\text{pred}}}{\partial c_1}\frac{\partial c_1}{\partial \theta_2} = -85{,}000 \cdot 1 \cdot 1 \cdot 2500 = -2.125 \times 10^8$$

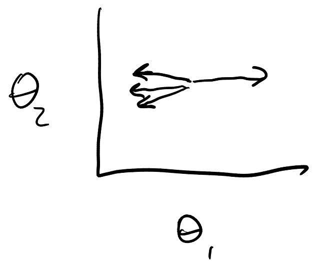
a “fast and furious” approach to training neural networks does not work and only leads to suffering. Now, suffering is a perfectly natural part of getting a neural network to work well, but it can be mitigated by being thorough, defensive, paranoid, and obsessed with visualizations of basically every possible thing. The qualities that in my experience correlate most strongly to success in deep learning are patience and attention to detail.

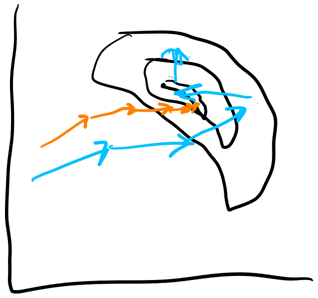
Keep calm and lower your learning rate

- Andrej Karpathy

Adaptive Step Size: RMSProp







$$\theta \leftarrow \theta + \alpha \underbrace{\nabla_\theta f_\theta(x)}_{g^{(k)}}$$

$$\hat{s}^{(k+1)} = \gamma \hat{s}^{(k)} + (1-\gamma)\left(g^{(k)} \odot g^{(k)}\right)$$

element-wise product

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\alpha}{\varepsilon + \sqrt{\hat{s}_i^{(k+1)}}}\, g_i^{(k)}$$

Adaptive Step Size: ADAM

(Adaptive Moment Estimation)

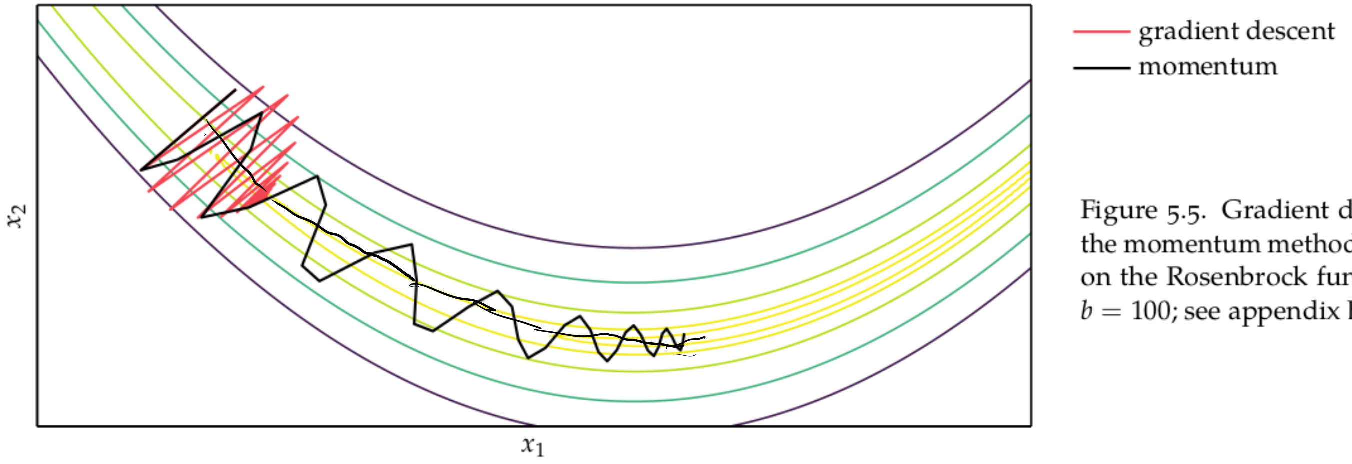


Figure 5.5. Gradient descent and the momentum method compared on the Rosenbrock function with $b = 100$; see appendix B.6.

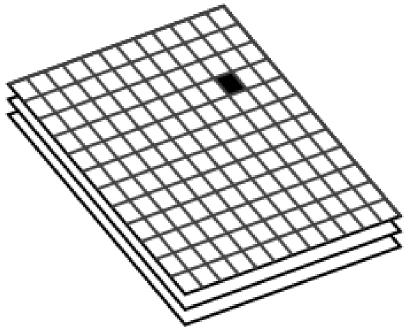
biased decaying momentum $\quad v^{(k+1)} = \gamma_v v^{(k)} + (1-\gamma_v)\, g^{(k)}$

biased decaying sq. grad. $\quad s^{(k+1)} = \gamma_s s^{(k)} + (1-\gamma_s)\left(g^{(k)} \odot g^{(k)}\right)$

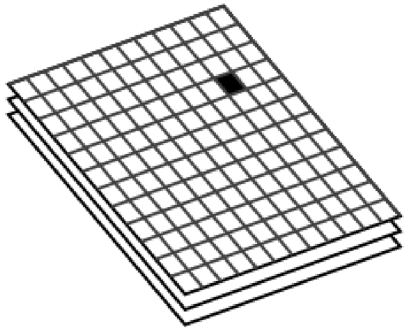
corrected decaying momentum $\quad \hat{v}$

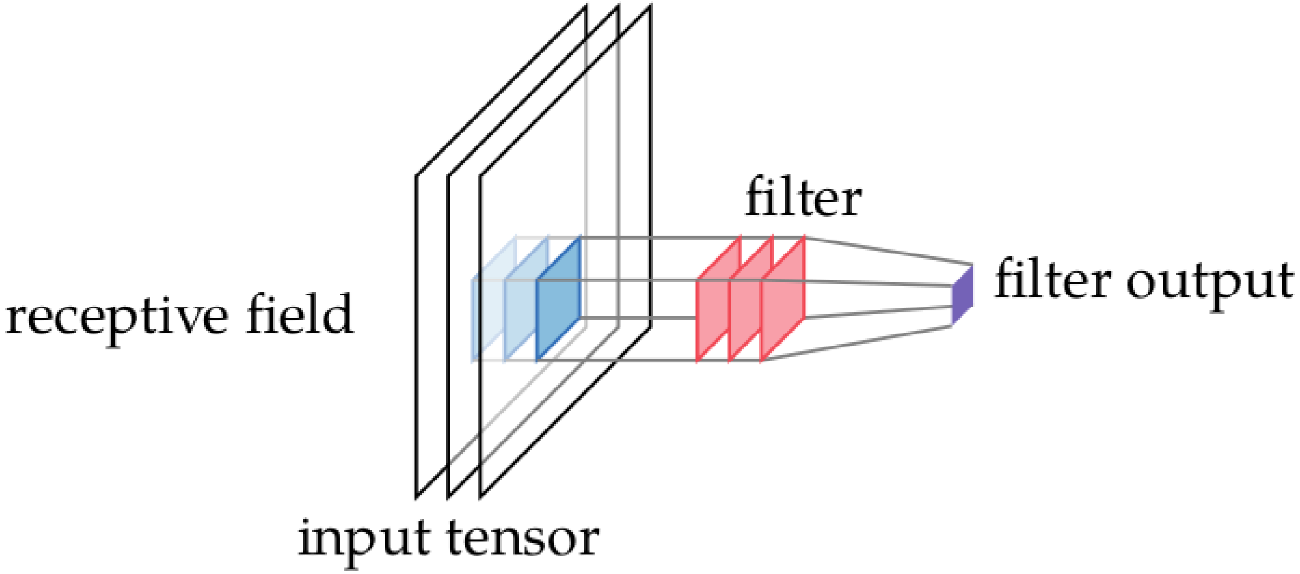
On Your Radar: ConvNets

On Your Radar: ConvNets

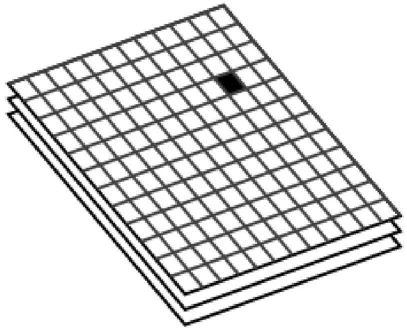


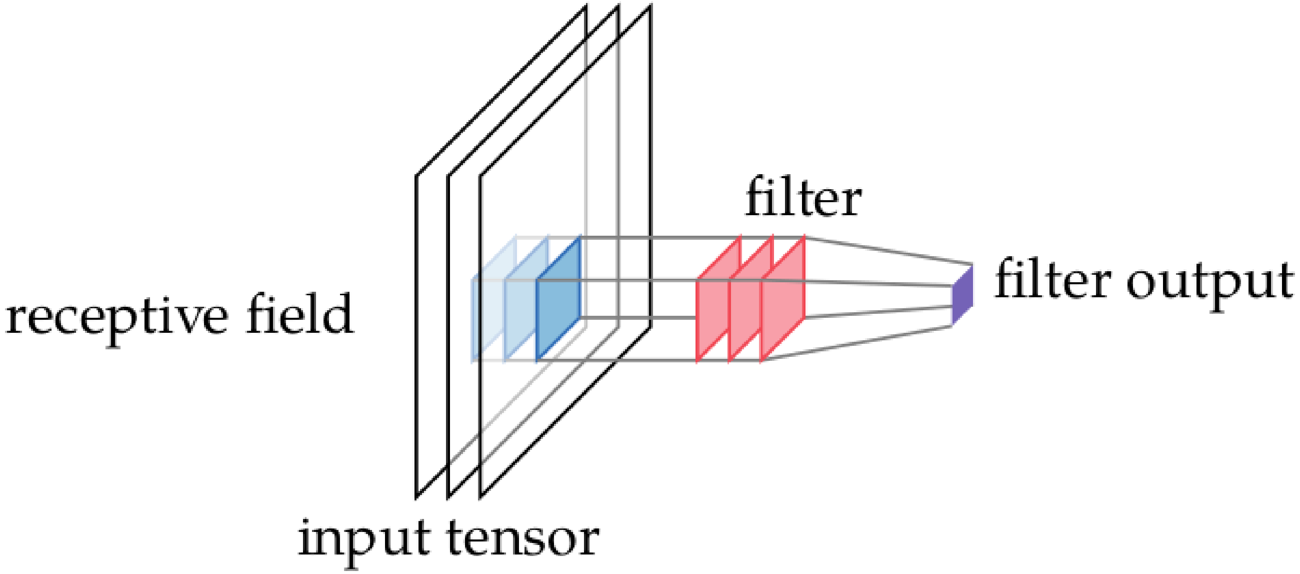
On Your Radar: ConvNets

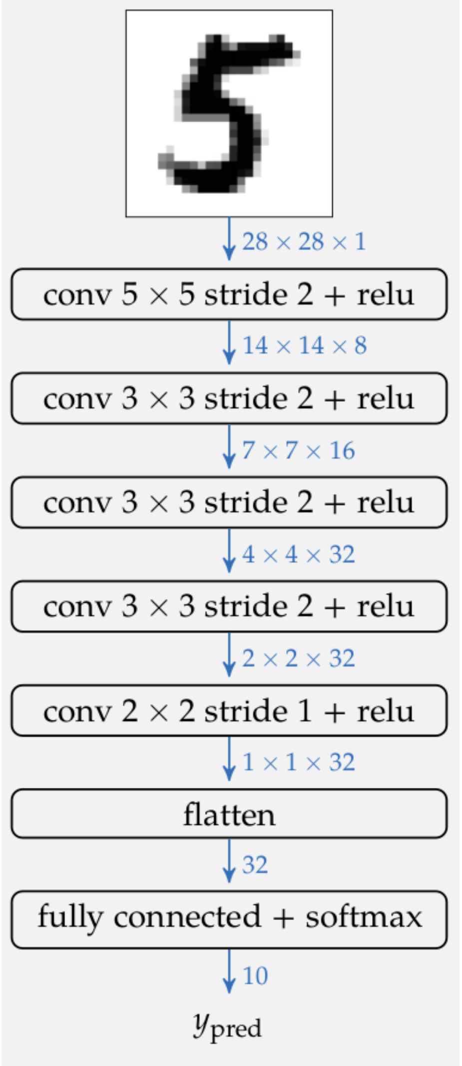




On Your Radar: ConvNets







On Your Radar: Regularization

On Your Radar: Regularization

$$\arg\min_{\theta} \sum_{(x,y)\in \mathbf{D}} \ell(f_{\theta}(x), y) - \beta\|\theta\|^2$$

On Your Radar: Regularization

$$\arg\min_{\theta} \sum_{(x,y)\in \mathbf{D}} \ell(f_{\theta}(x), y) - \beta \|\theta\|^2$$

e.g. Batch norm, layer norm, dropout

On Your Radar: Skip Connections (Resnets)

Resources

OpenAI Spinning up