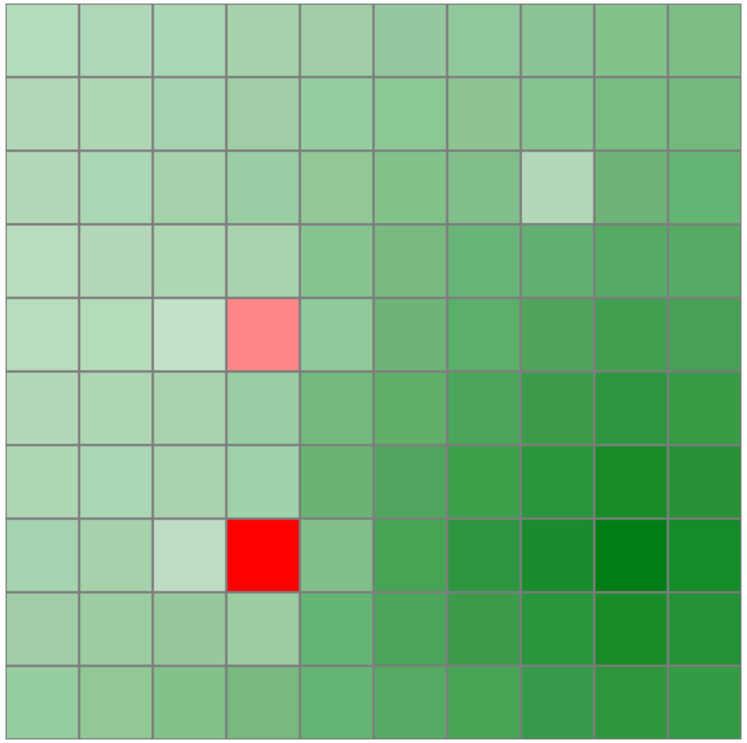
Neural Network Function Approximation



Do we really need to keep track of $U(s)$ for every U separately?

Function Approximation

$$\underset{\text{output}}{y} \approx f_{\underset{\text{parameters}}{\theta}}(\overset{\text{input}}{x})$$

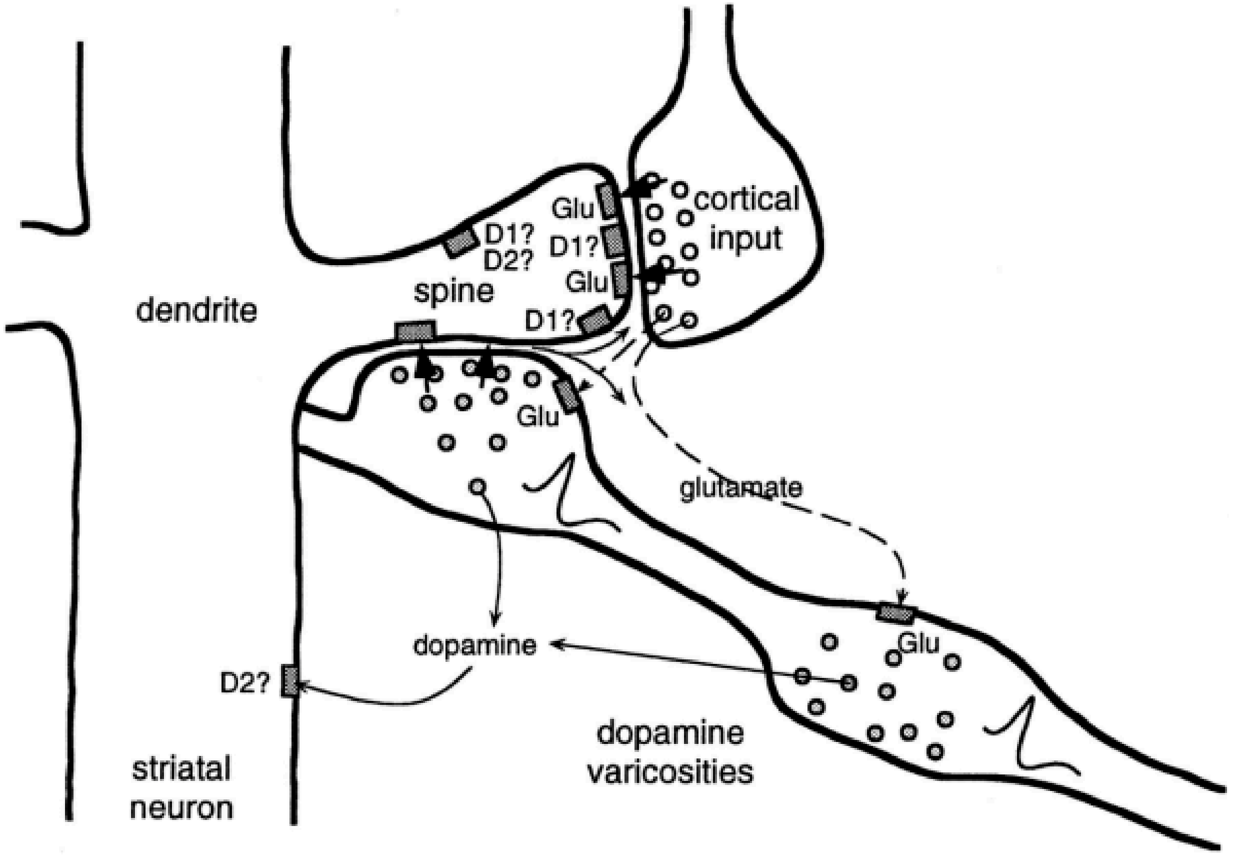
Example: Linear Function Approximation:

$$f_\theta(x) = \underset{\text{weights}}{\theta}^\top \underset{\text{features}}{\beta}(x)$$

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

Neural Network

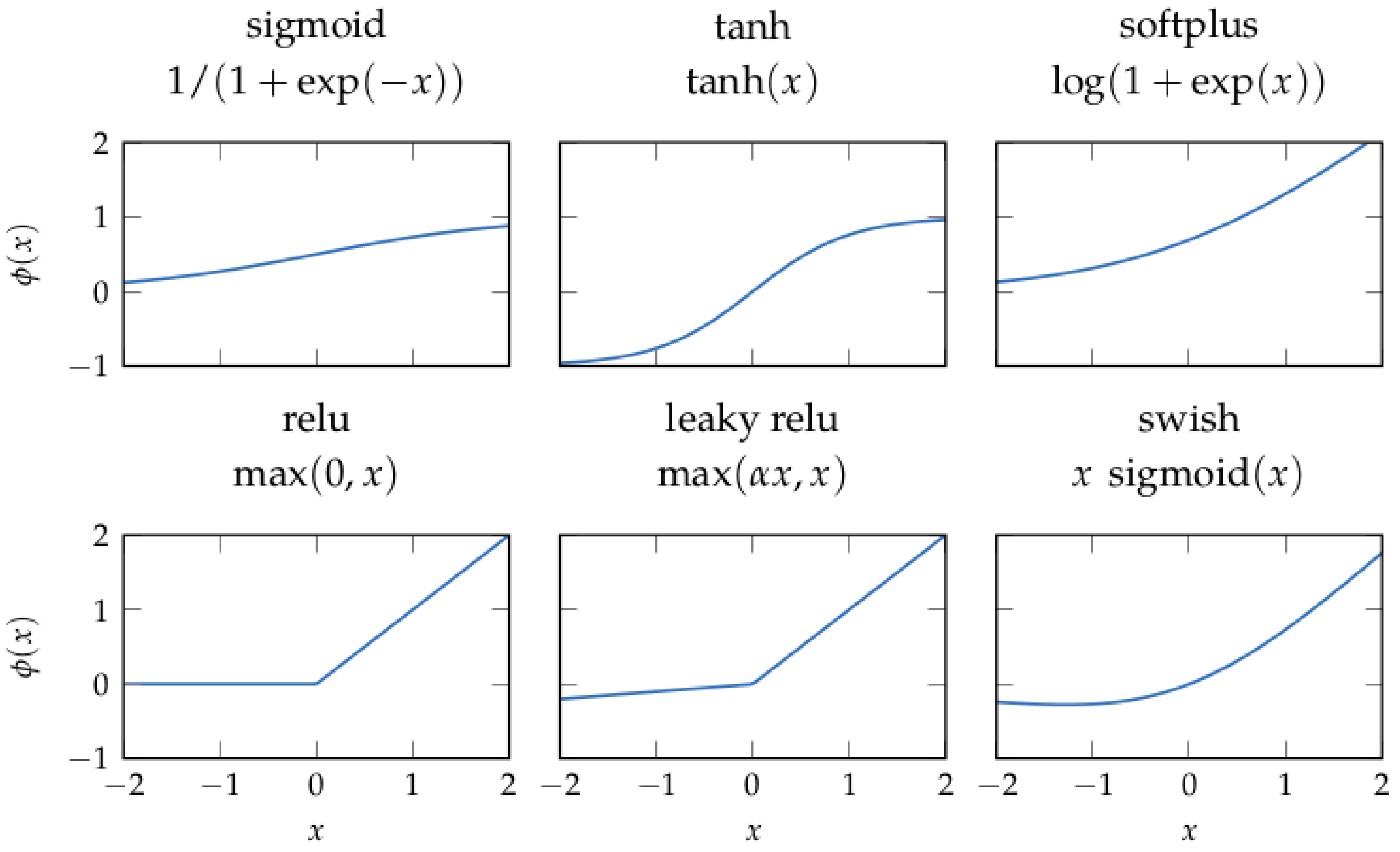
$$h(x) = \sigma(Wx + b)$$



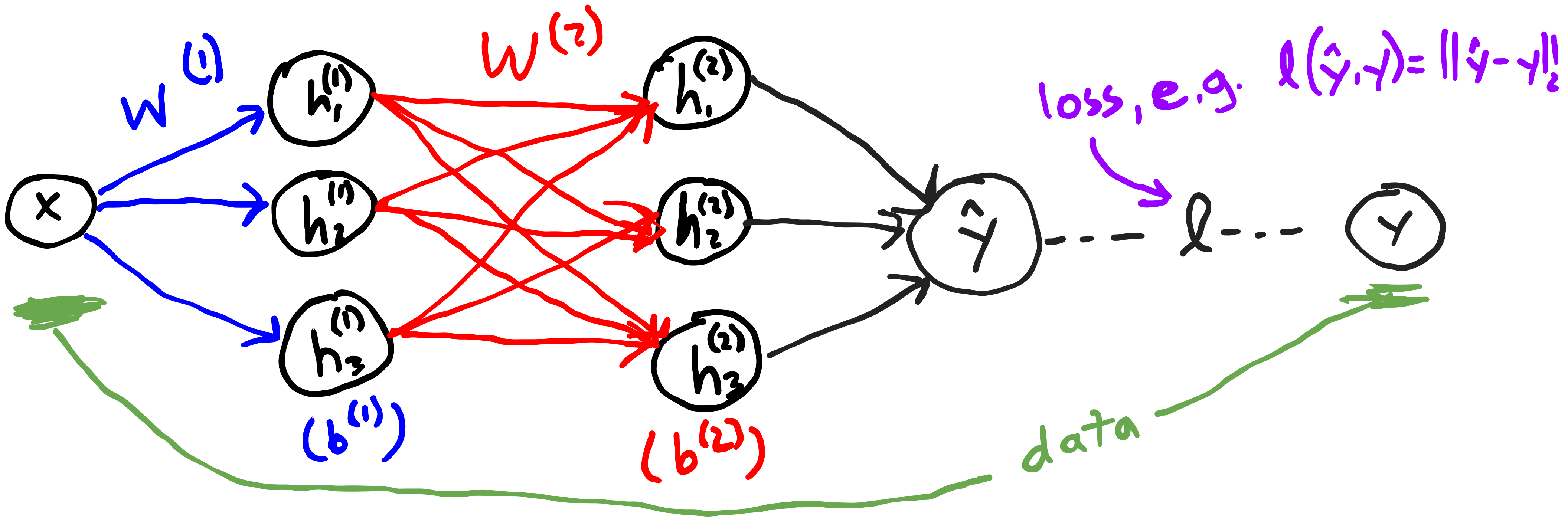
Neural Network

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

Nonlinearities



Training



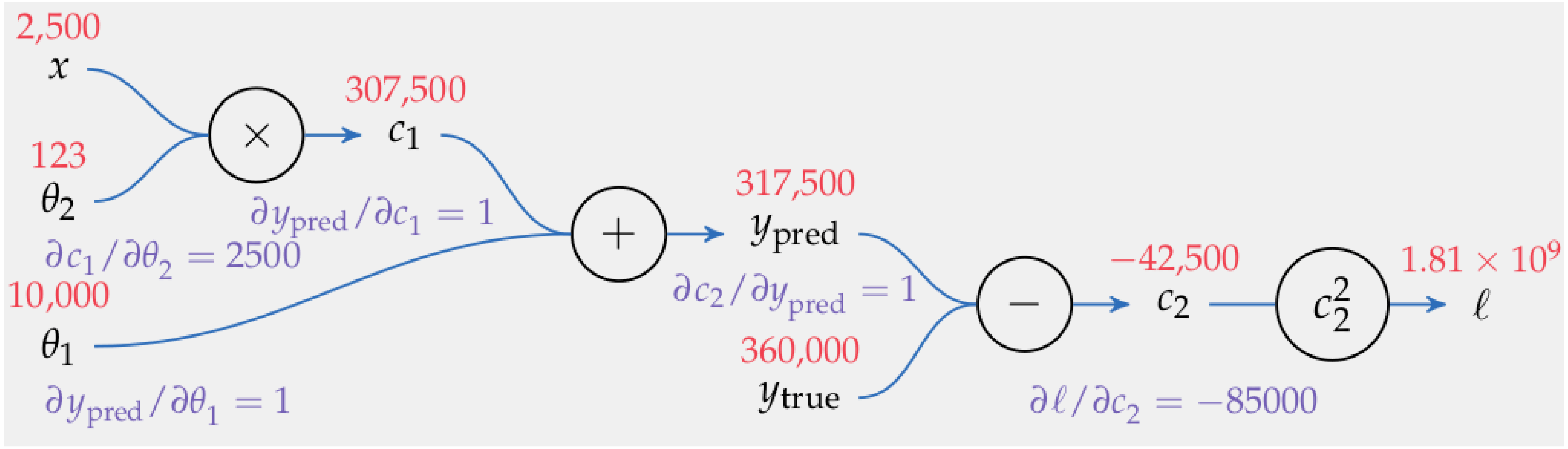
$$\theta^* = \arg\min_\theta \sum_{(x,y)\in\mathcal{D}} l(f_\theta(x), y)$$

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

Chain Rule

Backprop

$$l(x, y_{\text{true}}) = (\theta_2 x + \theta_1 - y_{\text{true}})^2$$


2,500
x
123
θ_2
$\times$
307,500
c_1
$\partial y_{\text{pred}} / \partial c_1 = 1$
$\partial c_1 / \partial \theta_2 = 2500$
10,000
θ_1
$\partial y_{\text{pred}} / \partial \theta_1 = 1$
$+$
317,500
y_{pred}
$\partial c_2 / \partial y_{\text{pred}} = 1$
360,000
y_{true}
$-$
$-42{,}500$
c_2
c_2^2
$\partial \ell / \partial c_2 = -85000$
1.81×10^9
ℓ

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

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

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.

- Andrej Karpathy

Adaptive Step Size: RMSProp

Adaptive Step Size: ADAM

(Adaptive Moment Estimation)

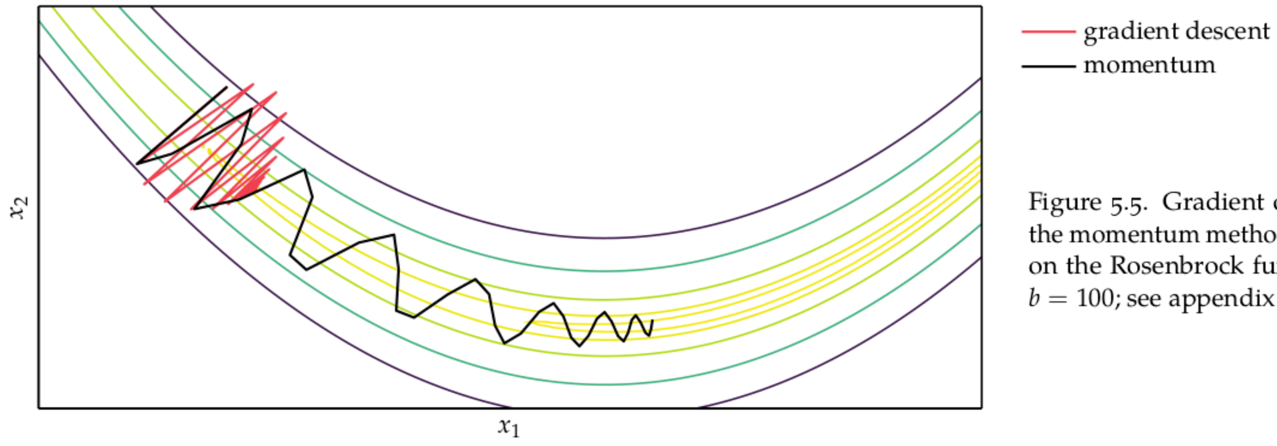


Figure 5.5. Gradient descent and the momentum method compared on the Rosenbrock function with $b = 100$; see appendix B.6.

$$\text{biased decaying momentum: } \mathbf{v}^{(k+1)} = \gamma_v \mathbf{v}^{(k)} + (1-\gamma_v)\mathbf{g}^{(k)} \tag{5.29}$$

$$\text{biased decaying sq. gradient: } \mathbf{s}^{(k+1)} = \gamma_s \mathbf{s}^{(k)} + (1-\gamma_s)\left(\mathbf{g}^{(k)} \odot \mathbf{g}^{(k)}\right) \tag{5.30}$$

$$\text{corrected decaying momentum: } \hat{\mathbf{v}}^{(k+1)} = \mathbf{v}^{(k+1)}/(1-\gamma_v^k) \tag{5.31}$$

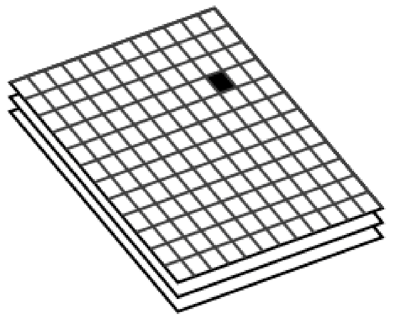
$$\text{corrected decaying sq. gradient: } \hat{\mathbf{s}}^{(k+1)} = \mathbf{s}^{(k+1)}/(1-\gamma_s^k) \tag{5.32}$$

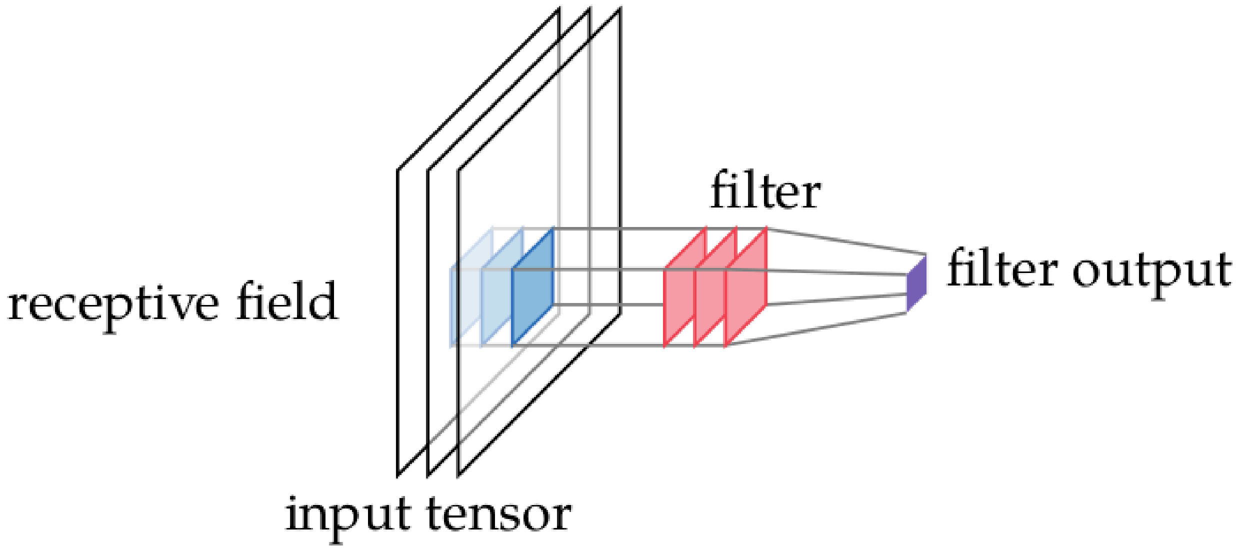
$$\text{next iterate: } \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \hat{\mathbf{v}}^{(k+1)} / \left(\epsilon + \sqrt{\hat{\mathbf{s}}^{(k+1)}}\right) \tag{5.33}$$

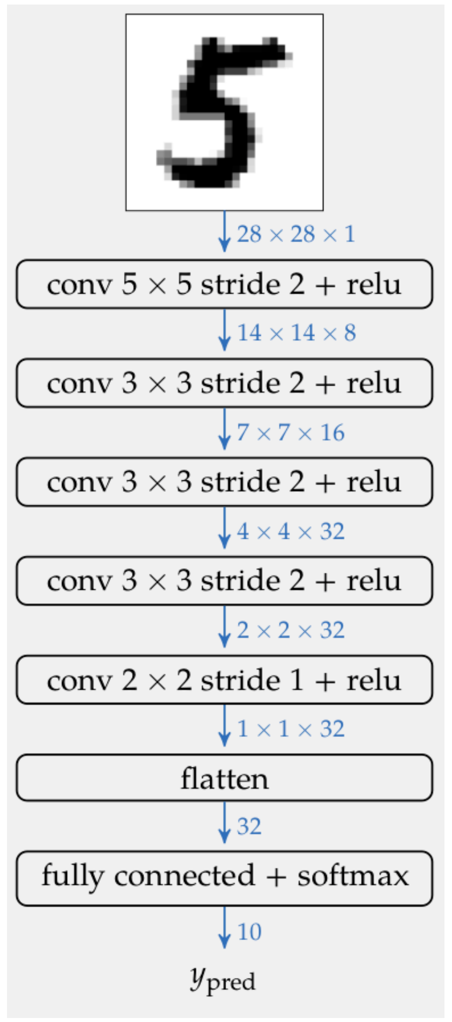
[12] According to the original paper, good default settings are $\alpha = 0.001$, $\gamma_v = 0.9$, $\gamma_s = 0.999$, and $\epsilon = 1 \times 10^{-8}$.

$\odot$ means elementwise multiplication.

On Your Radar: ConvNets






28 × 28 × 1
conv 5 × 5 stride 2 + relu
14 × 14 × 8
conv 3 × 3 stride 2 + relu
7 × 7 × 16
conv 3 × 3 stride 2 + relu
4 × 4 × 32
conv 3 × 3 stride 2 + relu
2 × 2 × 32
conv 2 × 2 stride 1 + relu
1 × 1 × 32
flatten
32
fully connected + softmax
10
y_pred

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