Optimization Methods for Deep Learning

Gradient descent for non-convex optimization

Decsent Lemma: Let $f : \mathbb{R}^d \to \mathbb{R}$ be twice differentiable, and $\|\nabla^2 f\|_2 \leq \beta$. Then setting the learning rate $\eta = 1/\beta$, and applying gradient descent, $x_{t+1} = x_t - \eta \nabla f(x_t)$, we have:

$$f(x_t) - f(x_{t+1}) \geq \frac{1}{2\beta} \|\nabla f(x_t)\|_2^2.$$

Converging to stationary points

Theorem: In $T = O(\frac{\beta}{\epsilon^2})$ iterations, we have $\|\nabla f(x)\|_2 \leq \epsilon$.

Gradient Descent for Quadratic Functions

Problem: $\min_x \frac{1}{2} x^\top A x$ with $A \in \mathbb{R}^{d\times d}$ being positive-definite.

Theorem: Let $\lambda_{\max}$ and $\lambda_{\min}$ be the largest and the smallest eigenvalues of A. If we set $\eta \le \frac{1}{\lambda_{\max}}$, we have

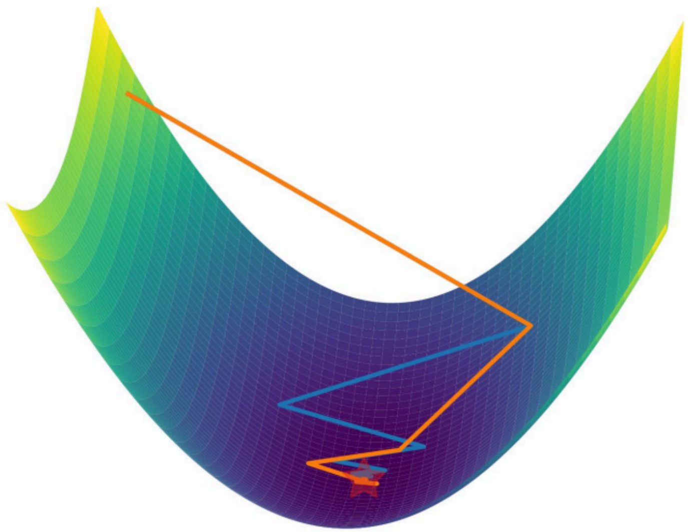
$$\|x_t\|_2 \le \left(1 - \eta\lambda_{\min}\right)^t \|x_0\|_2$$

Momentum: Heavy-Ball Method (Polyak '64)

Problem: $\min_x f(x)$

Method: $v_{t+1} = -\nabla f(x_t) + \beta v_t$

$$x_{t+1} = x_t + \eta v_{t+1}$$

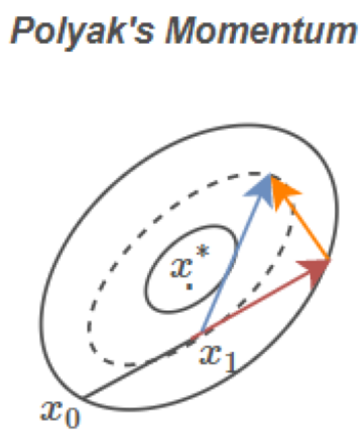


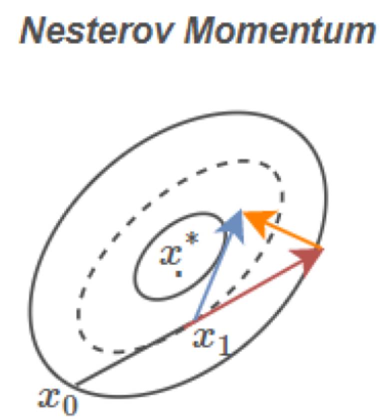
Momentum: Nesterov Acceleration (Nesterov '89)

Problem: $\min_x f(x)$

Method: $v_{t+1} = -\nabla f(x_t + \beta v_t) + \beta v_t$

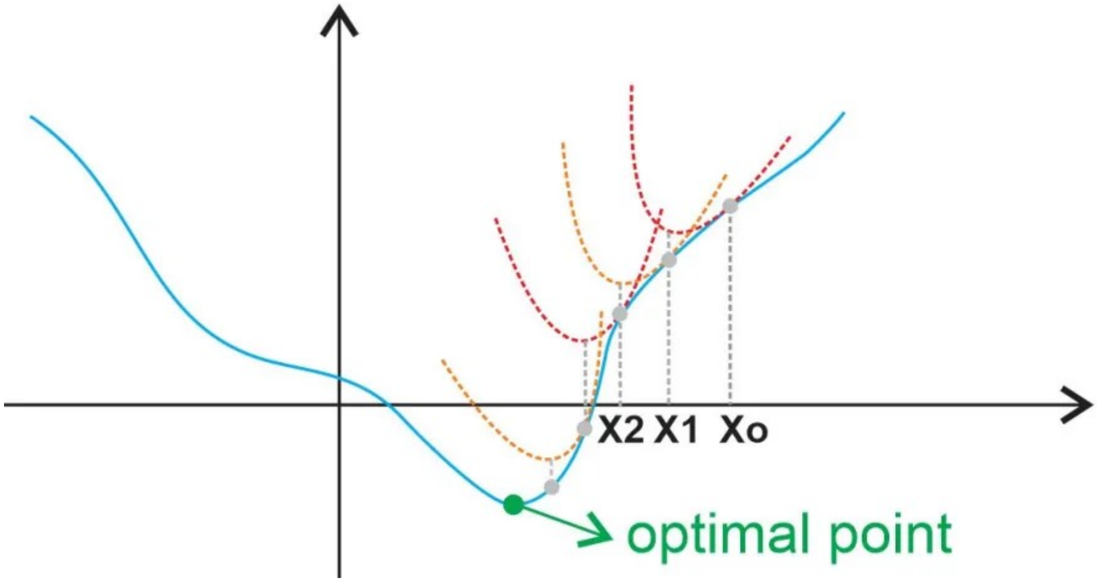
$$x_{t+1} = x_t + \eta v_{t+1}$$





Newton’s Method

Newton’s Method: $x_{t+1} = x_t - \eta(\nabla^2 f(x_t))^{-1} \nabla f(x_t)$



AdaGrad (Duchi et al. '11)

Newton Method: $x_{t+1} = x_t - \eta(\nabla^2 f(x_t))^{-1} \nabla f(x_t)$

AdaGrad: separate learning rate for every parameter

$$x_{t+1} = x_t - \eta(G_{t+1} + \epsilon I)^{-1} \nabla f(x_t),\ (G_t)_{ii} = \sqrt{\sum_{j=1}^{t-1} \left(\nabla f(x_t)_i \right)^2}$$

RMSProp (Hinton et al. ’12)

AdaGrad: separate learning rate for every parameter

$$x_{t+1} = x_t - \eta(G_{t+1} + \epsilon I)^{-1} \nabla f(x_t), \ (G_t)_{ii} = \sqrt{\sum_{j=1}^{t-1} \left(\nabla f(x_t)_i \right)^2}$$

RMSProp: exponential weighting of gradient norms

$$x_{t+1} = x_t - \eta(G_{t+1} + \epsilon I)^{-1/2} \nabla f(x_t),$$
$$(G_{t+1})_{ii} = \beta(G_t)_{ii} + (1 - \beta)(\nabla f(x_t)_i)^2$$

AdaDelta (Zeiler ’12)

RMSProp:

$$x_{t+1} = x_t - \eta(G_{t+1} + \epsilon I)^{-1/2} \nabla f(x_t),$$
$$(G_{t+1})_{ii} = \beta(G_t)_{ii} + (1-\beta)(\nabla f(x_t)_i)^2$$

AdaDelta:

$$x_{t+1} = x_t - \eta \Delta x_t,$$
$$\Delta x_t = \sqrt{u_t + \epsilon} \cdot (G_{t+1} + \epsilon I)^{-1/2} \nabla f(x_t)$$
$$(G_{t+1})_{ii} = \rho(G_t)_{ii} + (1-\rho)(\nabla f(x_t)_i)^2,$$
$$u_{t+1} = \rho u_t + (1-\rho)\|\Delta x_t\|_2^2$$

Adam (Kingma & Ba ’14)

Momentum:

$$v_{t+1} = -\nabla f(x_t) + \beta v_t, \; x_{t+1} = x_t + \eta v_{t+1}$$

RMSProp: exponential weighting of gradient norms

$$x_{t+1} = x_t - \eta (G_{t+1} + \epsilon I)^{-1} \nabla f(x_t),$$
$$(G_t)_{ii} = \beta (G_t)_{ii} + (1-\beta)(\nabla f(x_t)_i)^2$$

Adam:

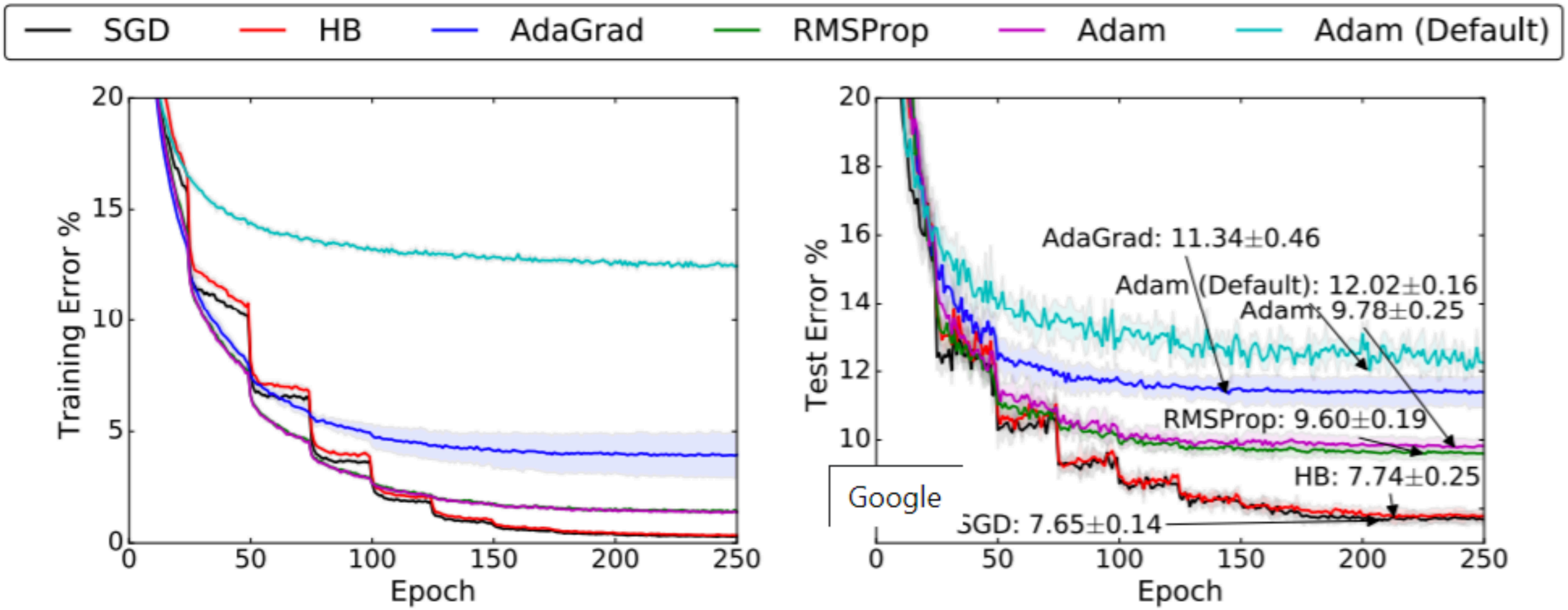
$$v_{t+1} = \beta_1 v_t + (1-\beta_1) \nabla f(x_t)$$
$$(G_{t+1})_{ii} = \beta_2 (G_t)_{ii} + (1-\beta_2)(\nabla f(x_t)_i)^2$$
$$x_{t+1} = x_t - \eta (G_{t+1} + \epsilon I)^{-1/2} v_{t+1}$$

Default choice nowadays.

Other Optimizers

- AdamW
- NAdam
- RAdam
- GGT
- K-FAC
- …

Are these actually useful



(a) CIFAR-10 (Train)

(b) CIFAR-10 (Test)

Figure 1: Training (left) and top-1 test error (right) on CIFAR-10. The annotations indicate where the best performance is attained for each method. The shading represents ± one standard deviation computed across five runs from random initial starting points. In all cases, adaptive methods are performing worse on both train and test than non-adaptive methods.

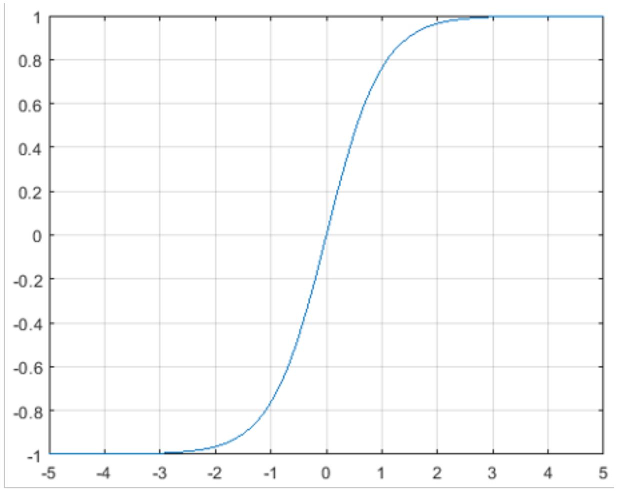
Wilson, Roelofs, Stern, Srebro, Recht '18

Important Techniques in Neural Network Training

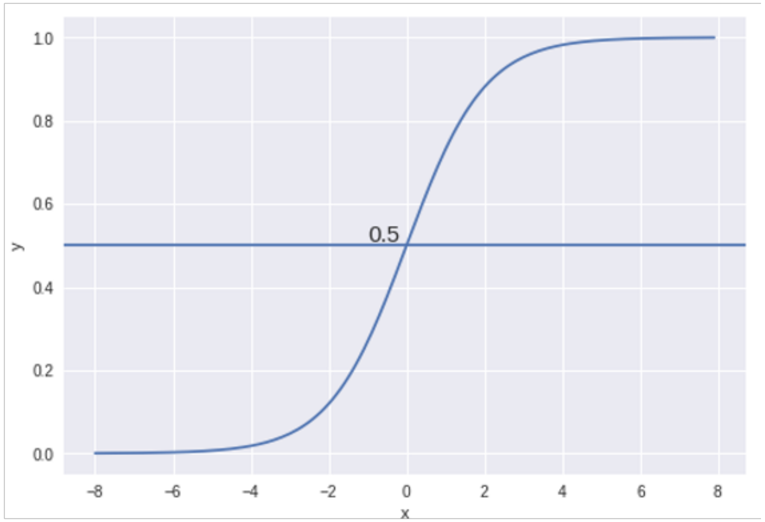
Gradient Explosion / Vanishing

- Deeper networks are harder to train:
    - Intuition: gradients are products over layers
    - Hard to control the learning rate

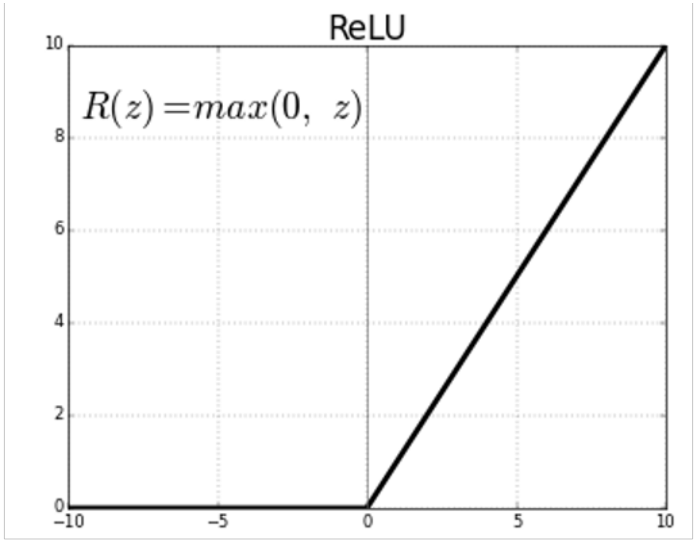
Activation Functions



tanh

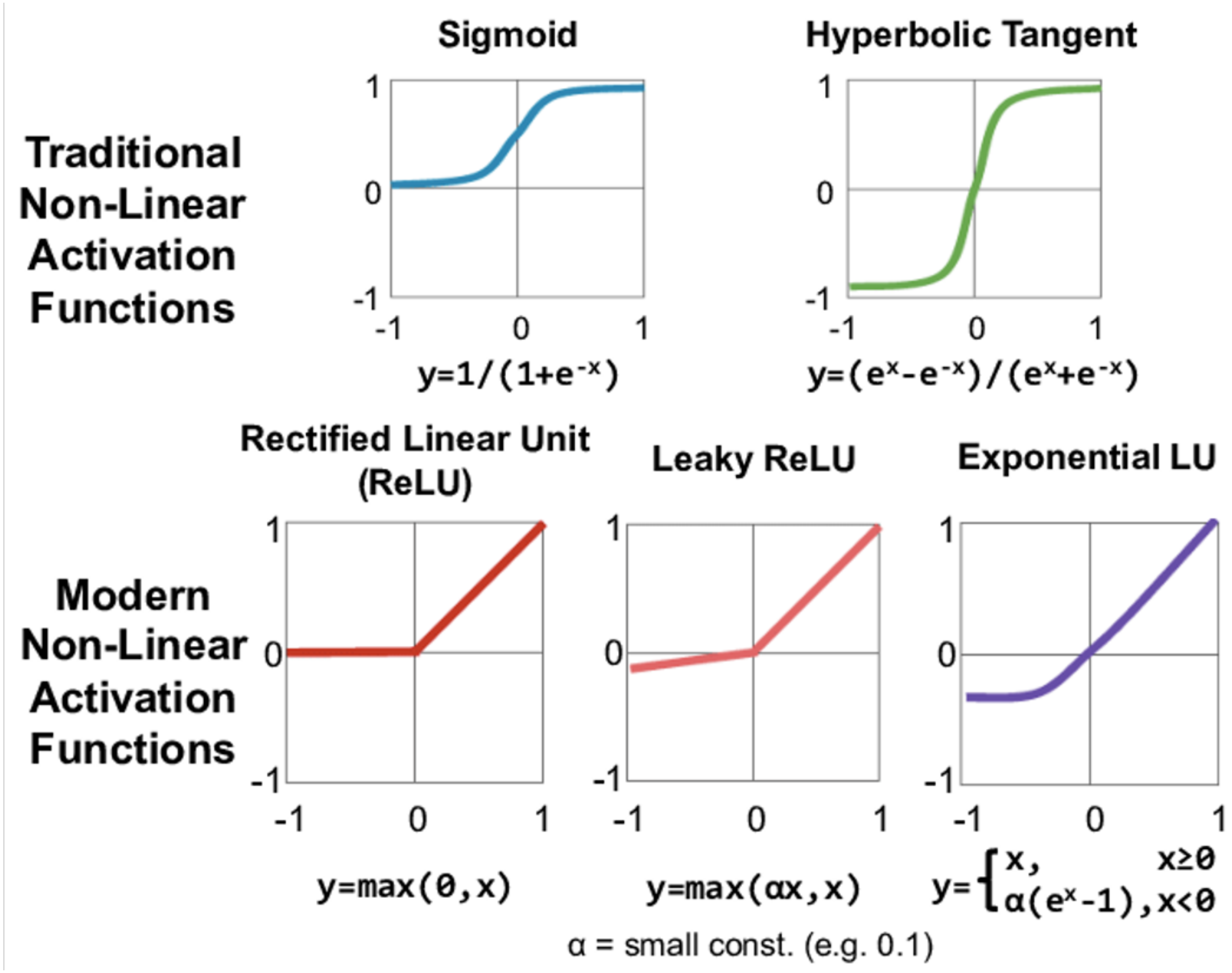


sigmoid



Rectified Linear United

Activation Function



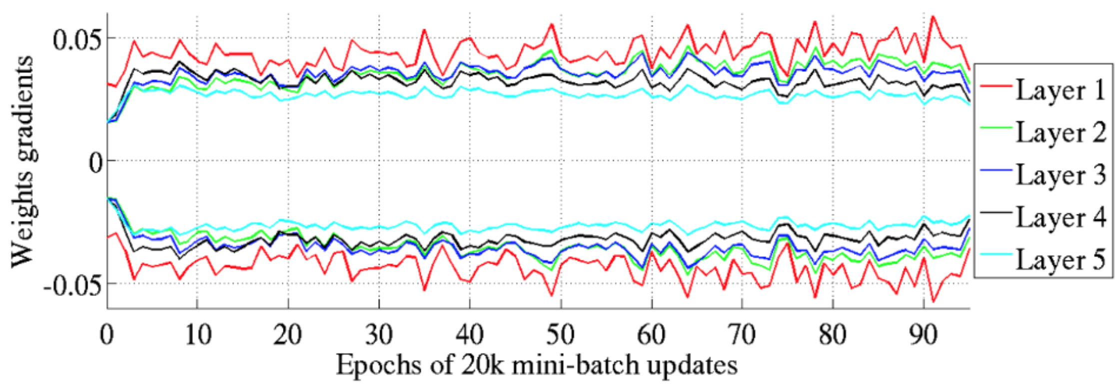
Initialization

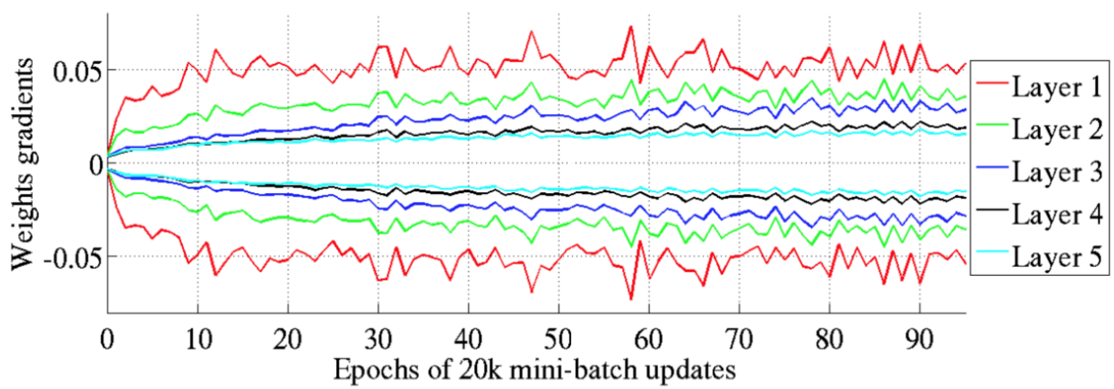
- Zero-initialization
- Large initialization
- Small initialization

- Design principles:
  - Zero activation mean
  - Activation variance remains same across layers

Xavier Initialization (Glorot & Bengio, '10)

- $W_{ij}^{(h)} \sim \text{Unif}\left[-\frac{\sqrt{6}}{\sqrt{d_h + d_{h+1}}}, \frac{\sqrt{6}}{\sqrt{d_h + d_{h+1}}}\right]$
- $b^{(h)} = 0$
- Experiments (tanh activation)





Kaiming Initialization (He et al. '15)

- $W_{ij}^{(h)} \sim \mathcal{N}\left(0, \frac{2}{d_h}\right).$
- $b^{(h)} = 0$
- Designed for ReLU activation
- 30-layer neural network