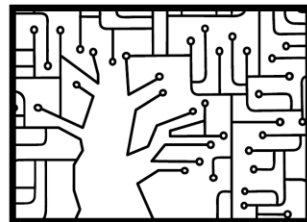


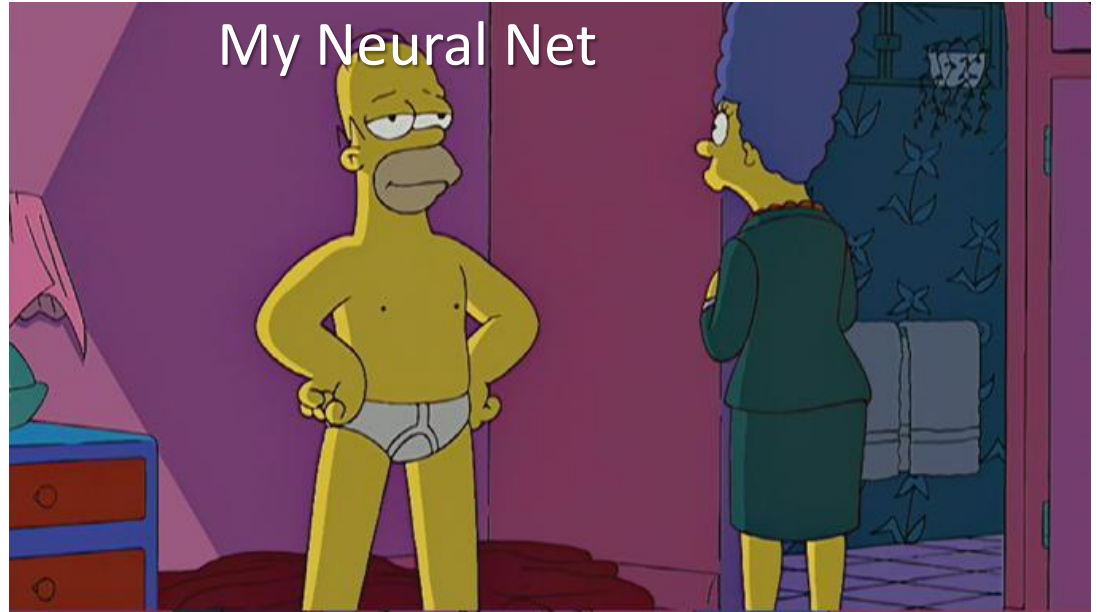
# Deep Learning for Computer Vision: Practical Training

Shai Bagon

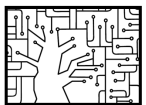


WAIC

My Neural Net

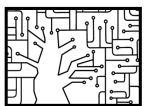


SGD tricks  
Regularization  
BatchNorm  
Initialization  
...



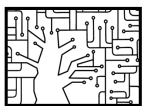
# Agenda

- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



# Gradient Descent

$$\mathcal{L}(\theta; \{(x_i, y_i)\}_{i \in 1..N}) = \frac{1}{|N|} \sum_{i \in \{1..N\}} \mathcal{L}(f(x_i; \theta), y_i)$$

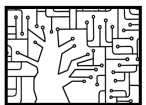


# Stochastic Gradient Descent (SGD)

$$\mathcal{L}(\theta; \{(\mathbf{x}_i, y_i)\}_{i \in 1..N}) \approx \frac{1}{|B|} \sum_{i \in \{B\}} \mathcal{L}(f(\mathbf{x}_i; \theta), y_i)$$

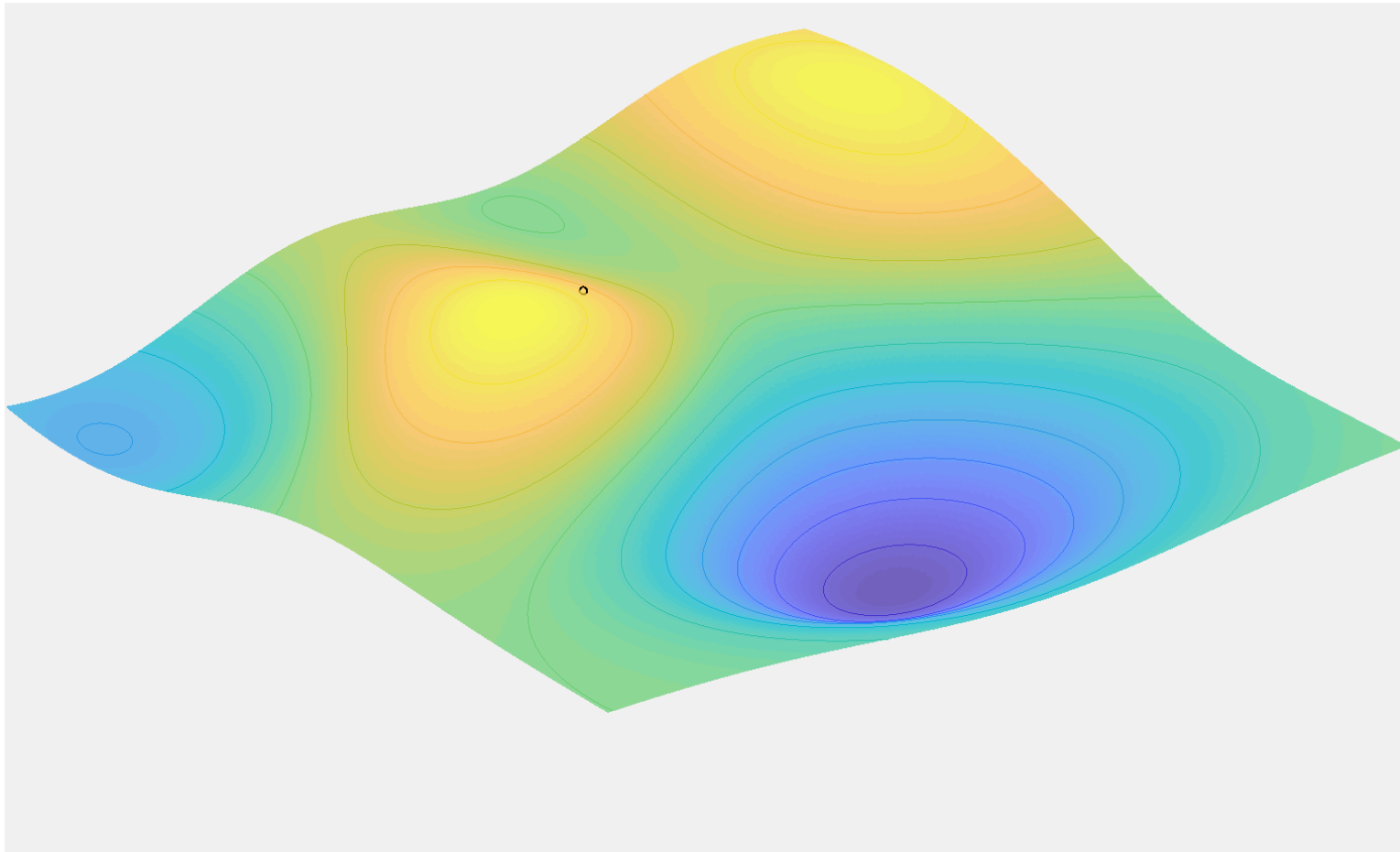
Update rule:

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t; \{(\mathbf{x}_i, y_i)\}_{i \in \{B\}})$$



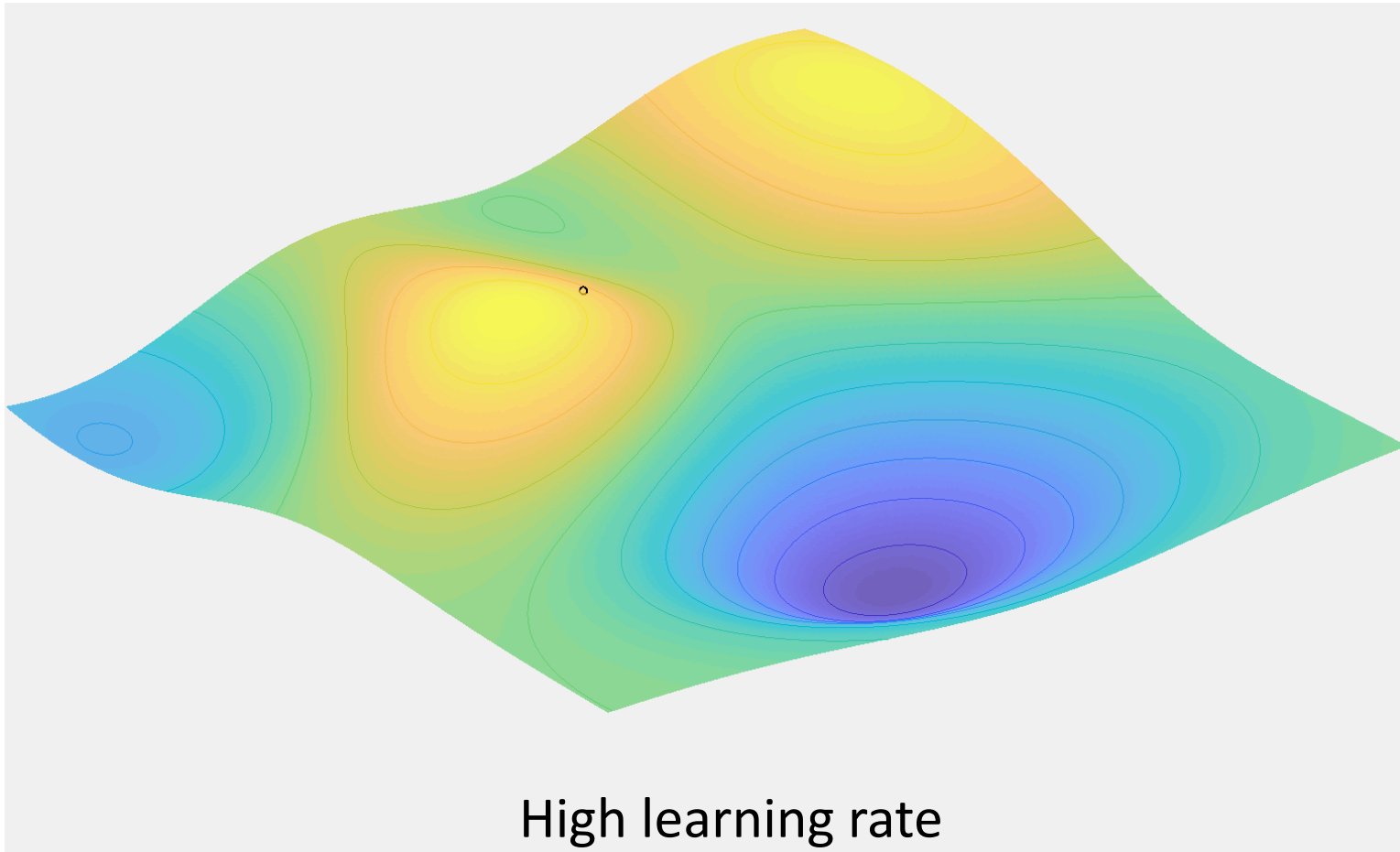
# Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$



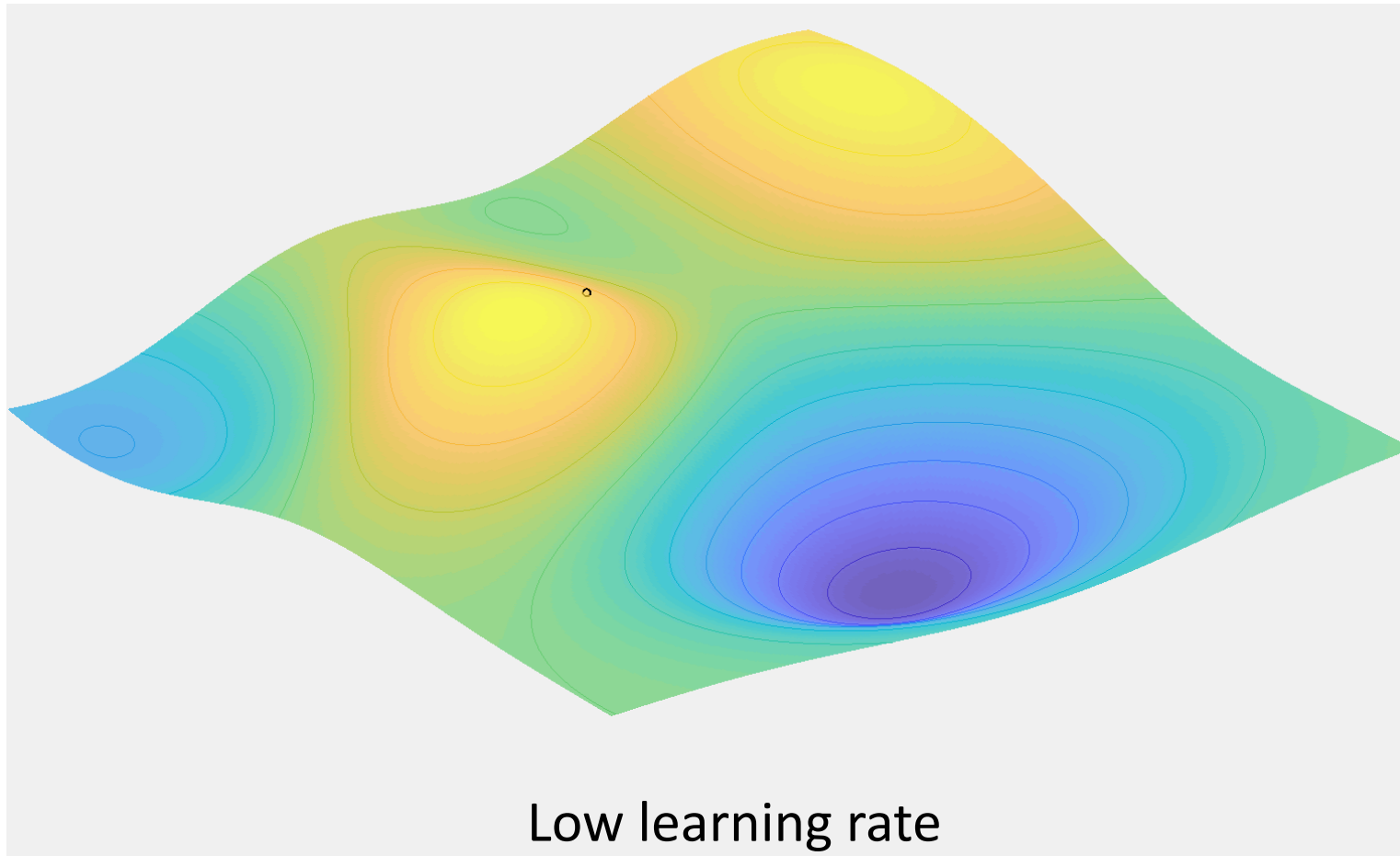
# Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$



# Vanilla SGD

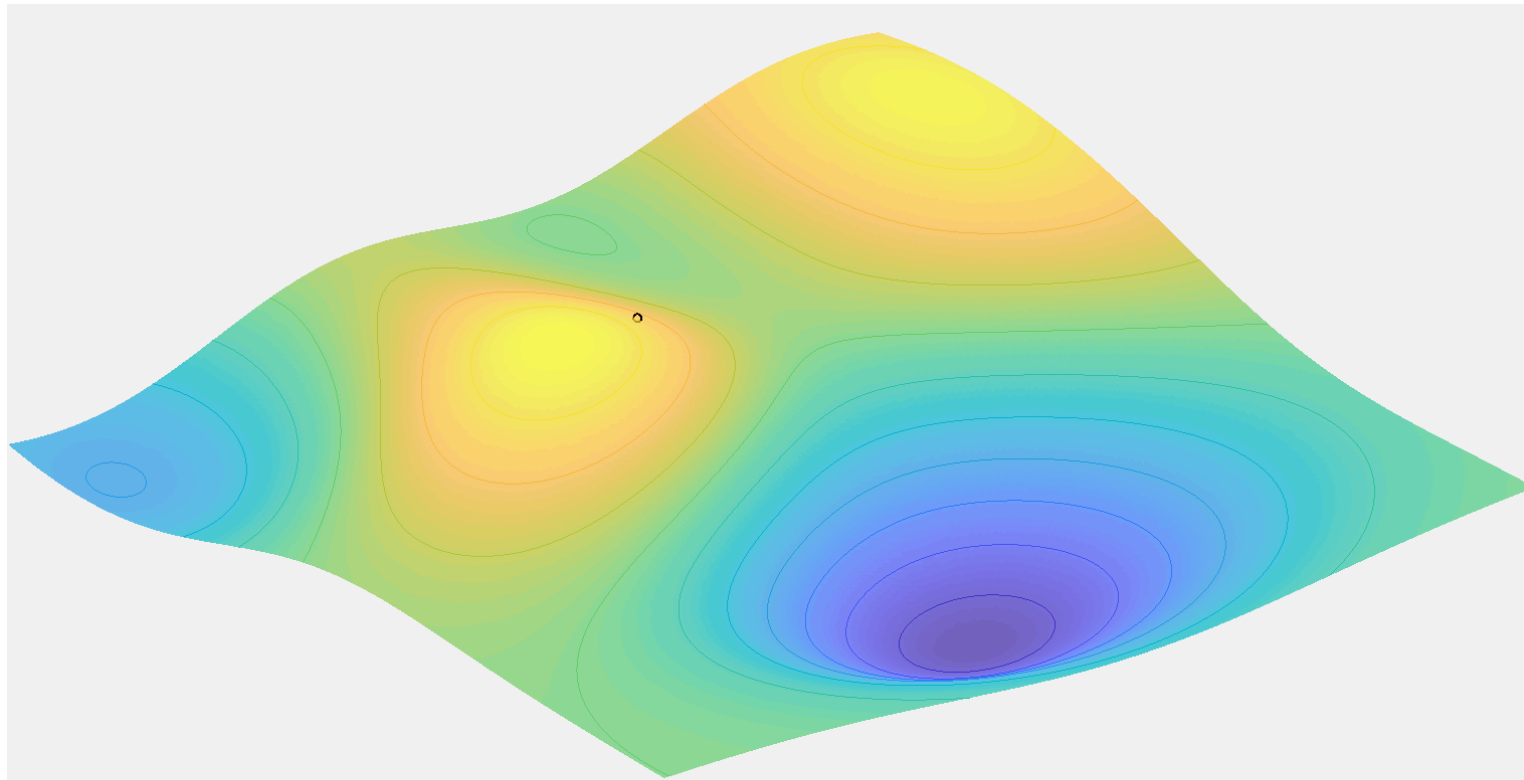
$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$





# Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$



“Ideal” learning rate

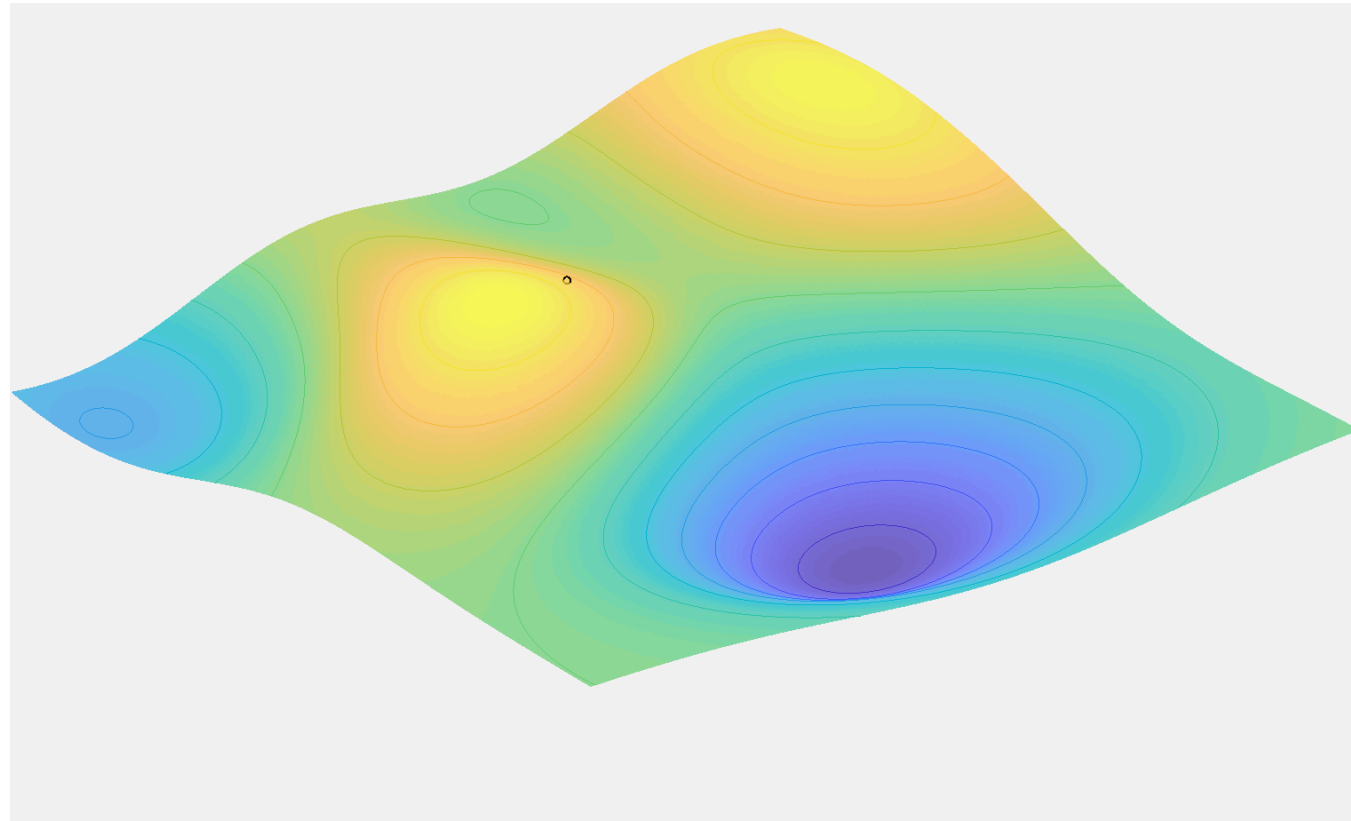
# Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$



# SGD with Momentum

$$\begin{aligned}\theta^{t+1} &= \theta^t + \mu v^t - \eta \nabla \mathcal{L}(\theta^t) \\ v^{t+1} &= \mu v^t - \eta \nabla \mathcal{L}(\theta^t)\end{aligned}$$



# SGD with Momentum

$$\begin{aligned}\theta^{t+1} &= \theta^t + \mu v^t - \eta \nabla \mathcal{L}(\theta^t) \\ v^{t+1} &= \mu v^t - \eta \nabla \mathcal{L}(\theta^t)\end{aligned}$$



Adam

$$v^{t+1} = \beta_1 v^t + (1 - \beta_1) \nabla \mathcal{L}(\theta^t) / (1 - (\beta_1)^t)$$

1<sup>st</sup> moment (like SGD)

Unbiased estimation

$$m^{t+1} = \beta_2 m^t + (1 - \beta_2) \nabla \mathcal{L}(\theta^t)^2 / (1 - (\beta_2)^t)$$

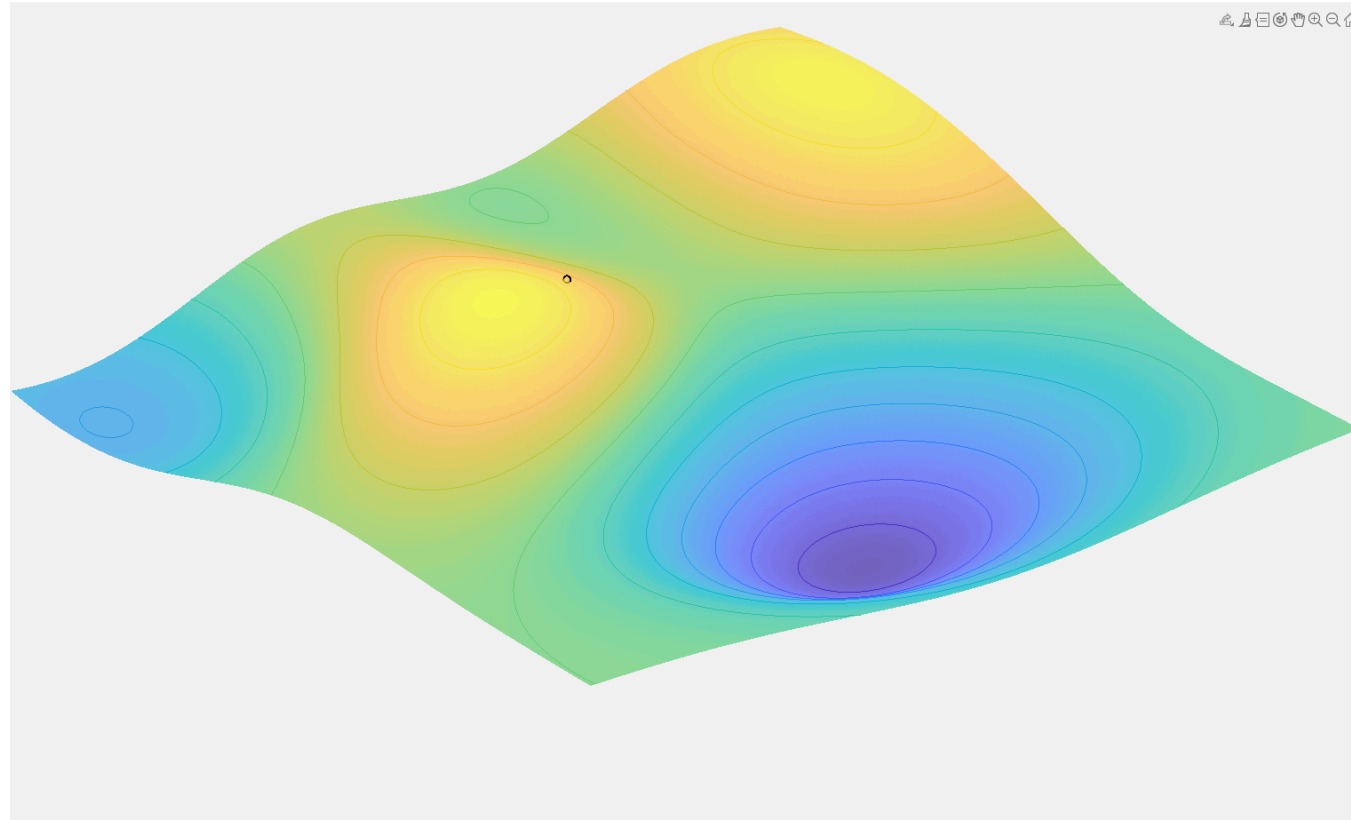
2<sup>nd</sup> moment ("variance")

Weight the change by the variance

$$\theta^{t+1} = \theta^t - \eta \left( v^{t+1} / (\sqrt{m^{t+1}} + \epsilon) \right)$$

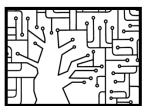
# Adam

$$\theta^{t+1} = \theta^t - \eta \cdot v^{t+1} / (\sqrt{m^{t+1}} + \varepsilon)$$

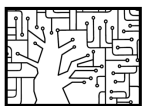
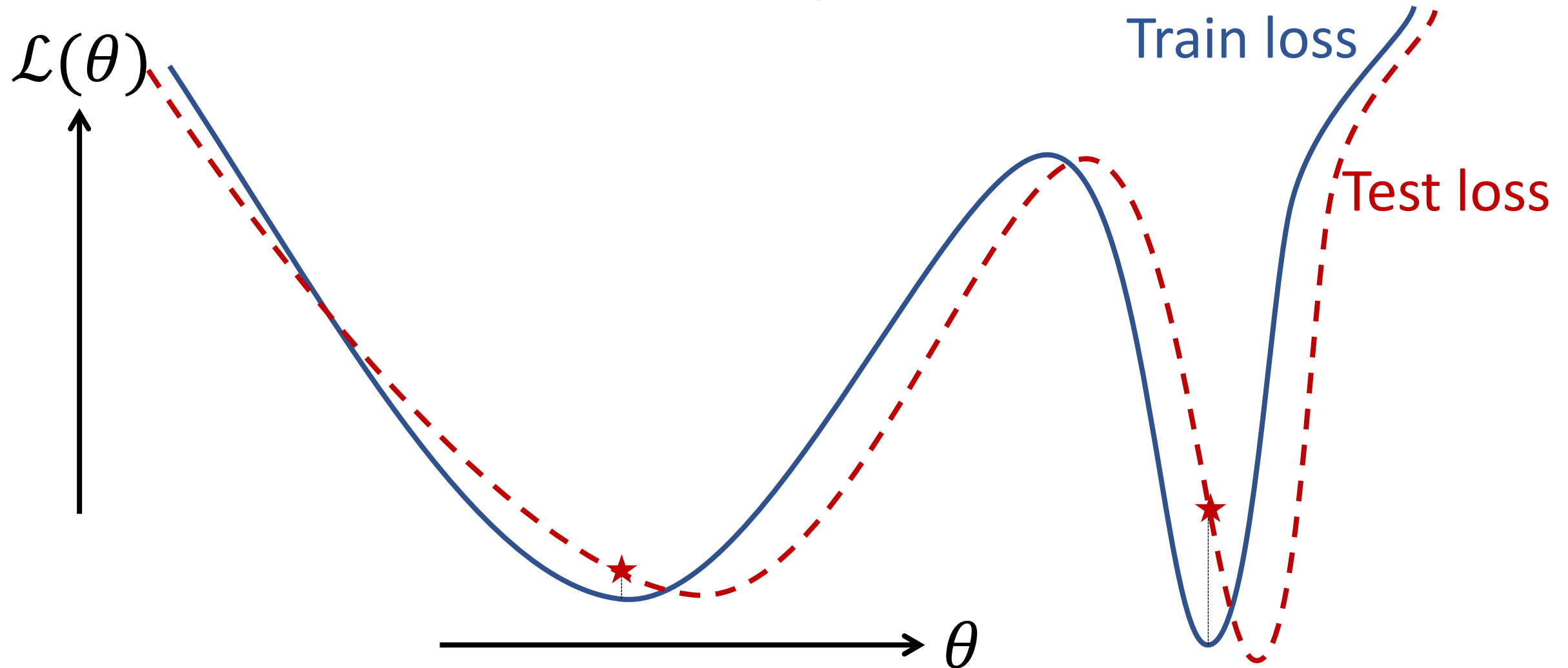


# Other SGD Update Rules

A good review can be found [here](#).



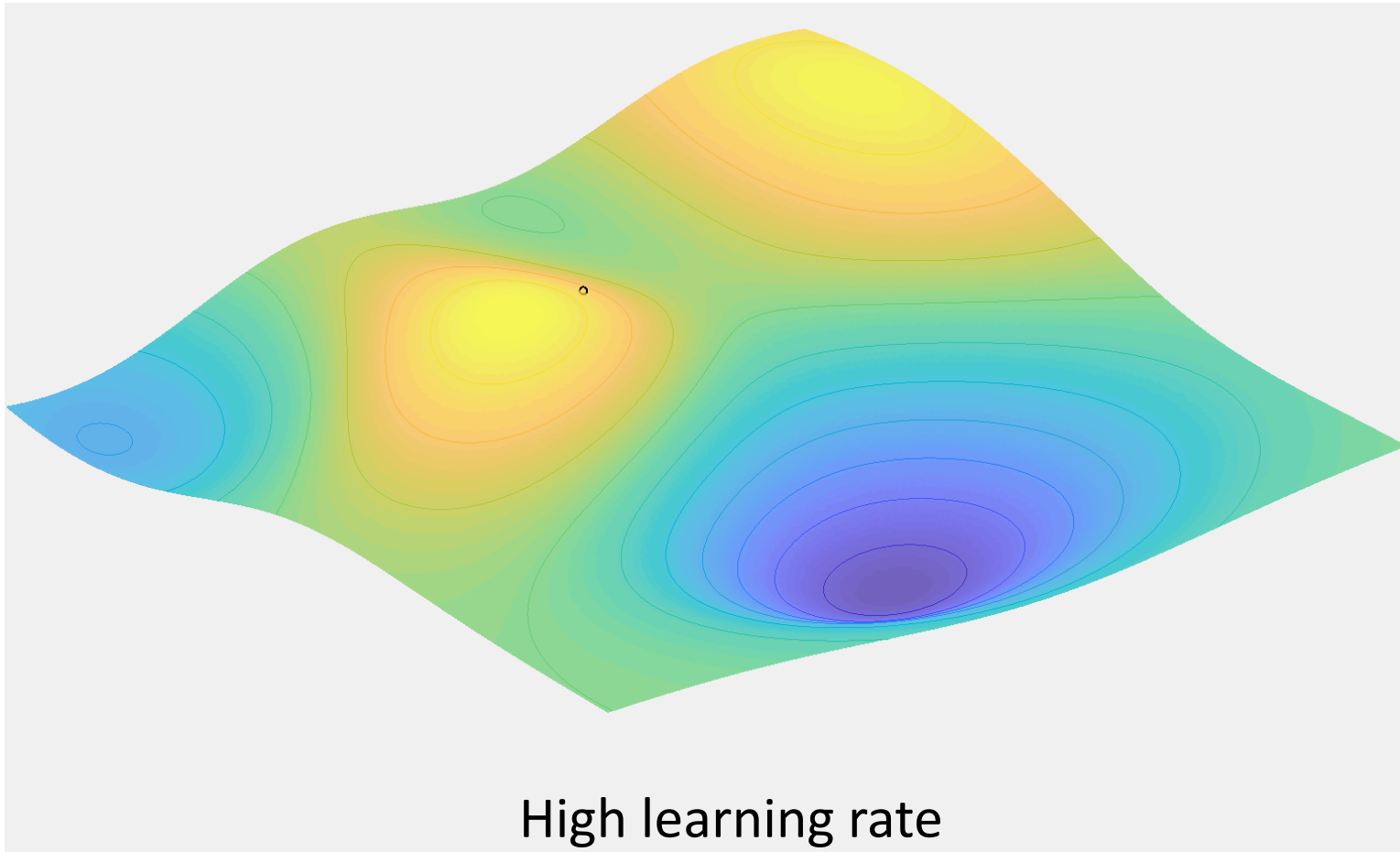
# Do we want fast convergence?





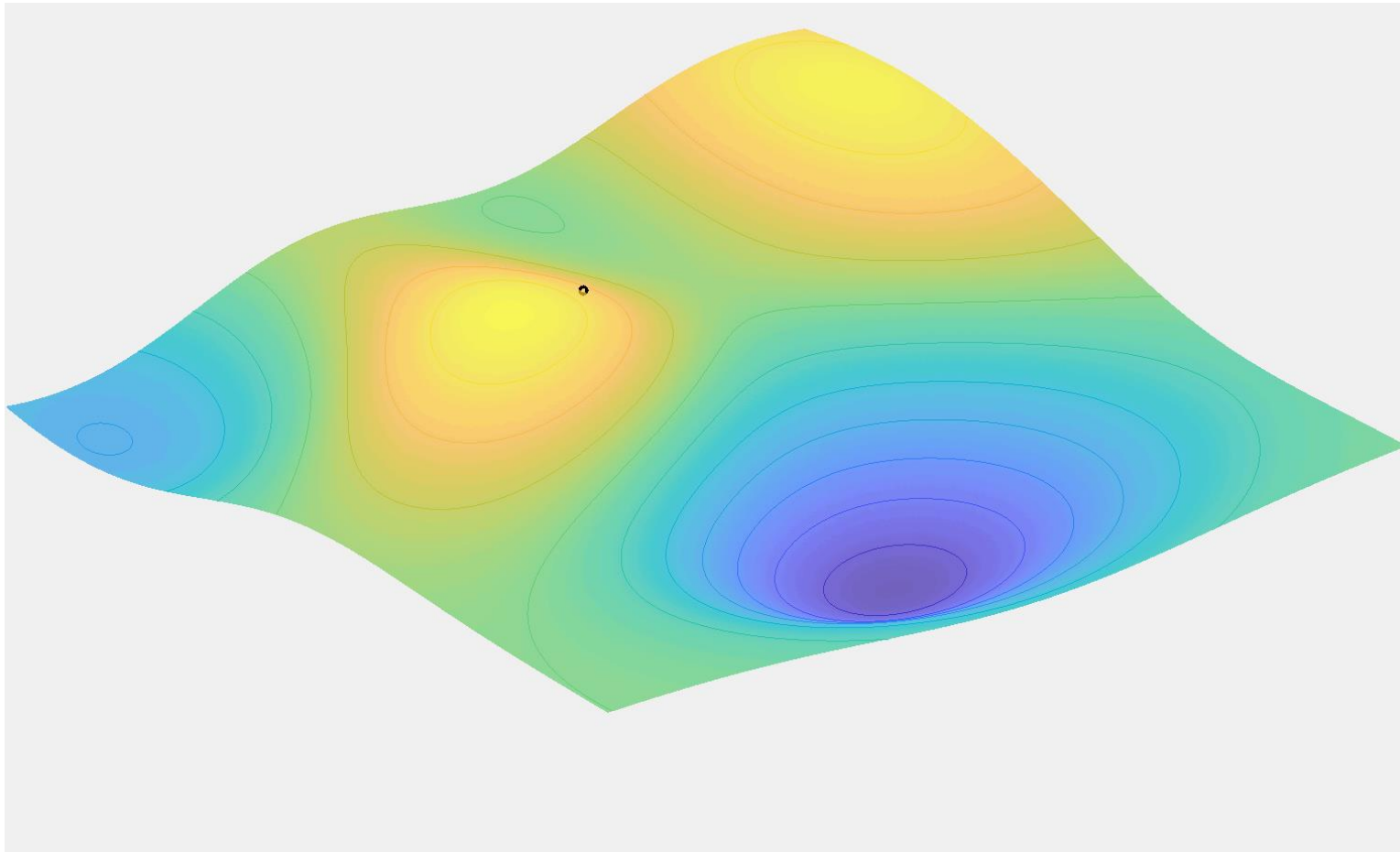
# Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$

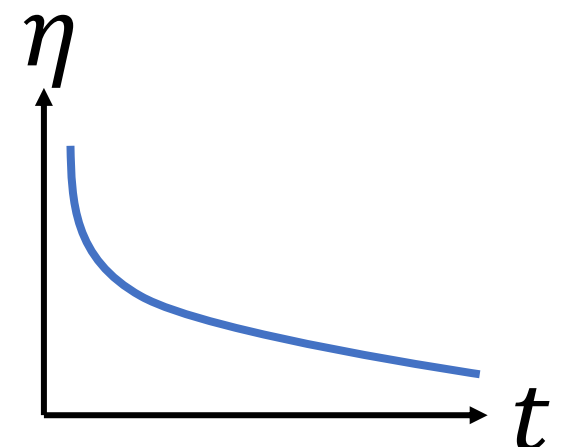
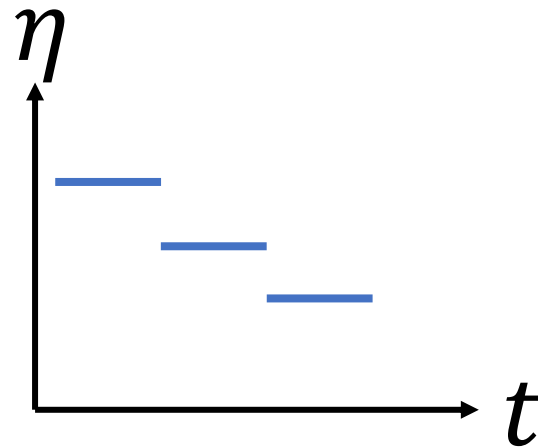
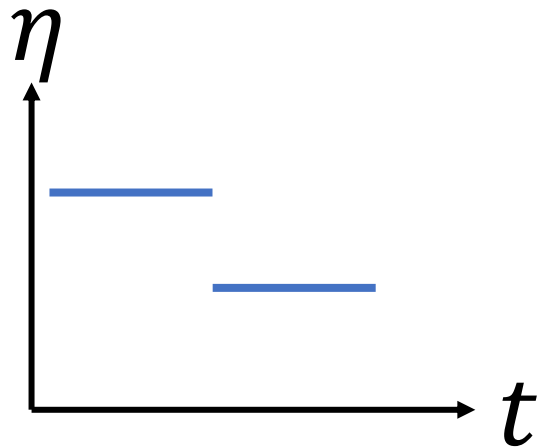


# Learning Rate Decay

$$\theta^{t+1} = \theta^t - \eta_t \nabla \mathcal{L}(\theta^t)$$



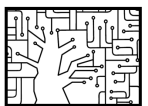
# Learning Rate Decay



# SGD - Remarks

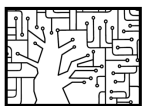
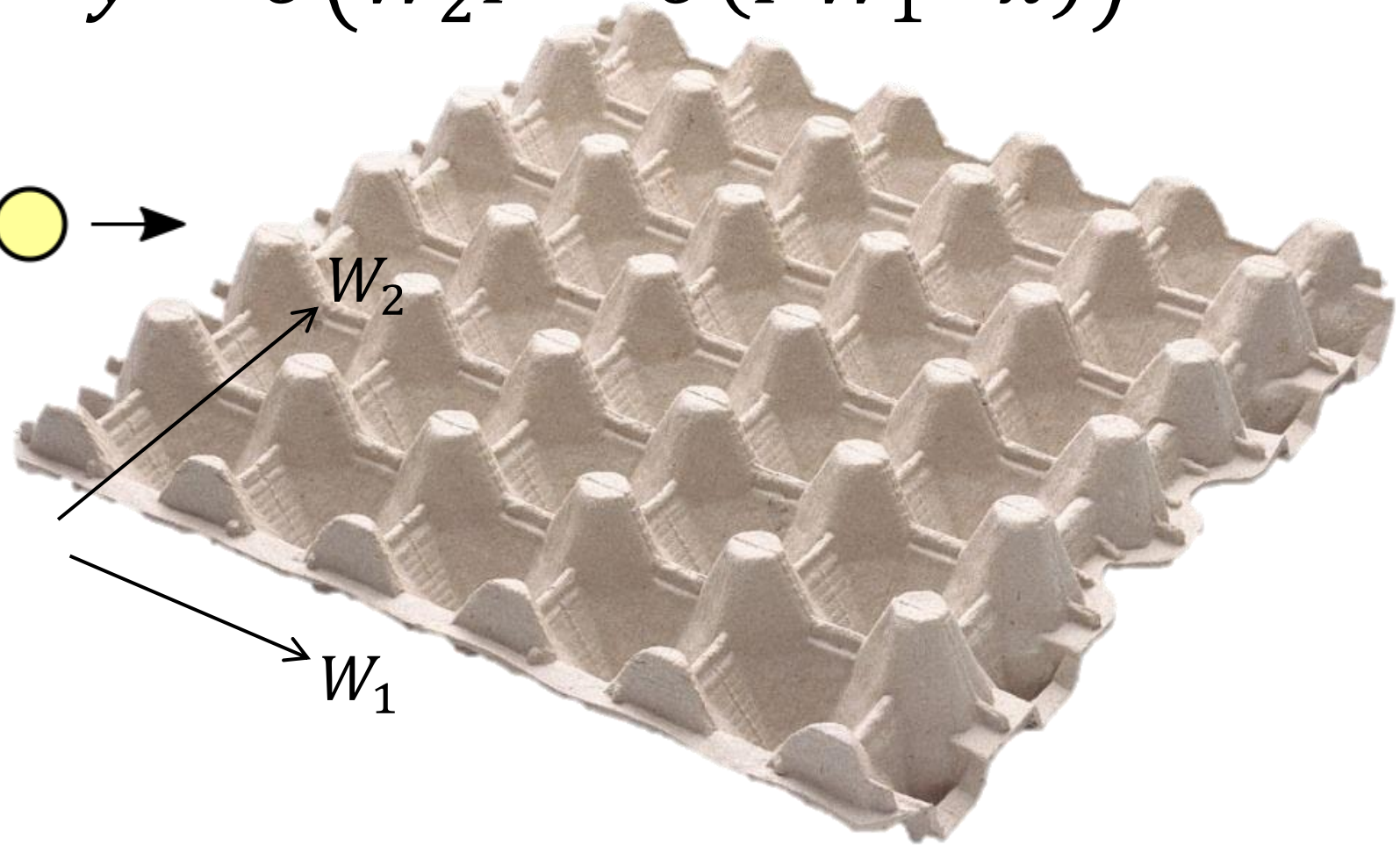
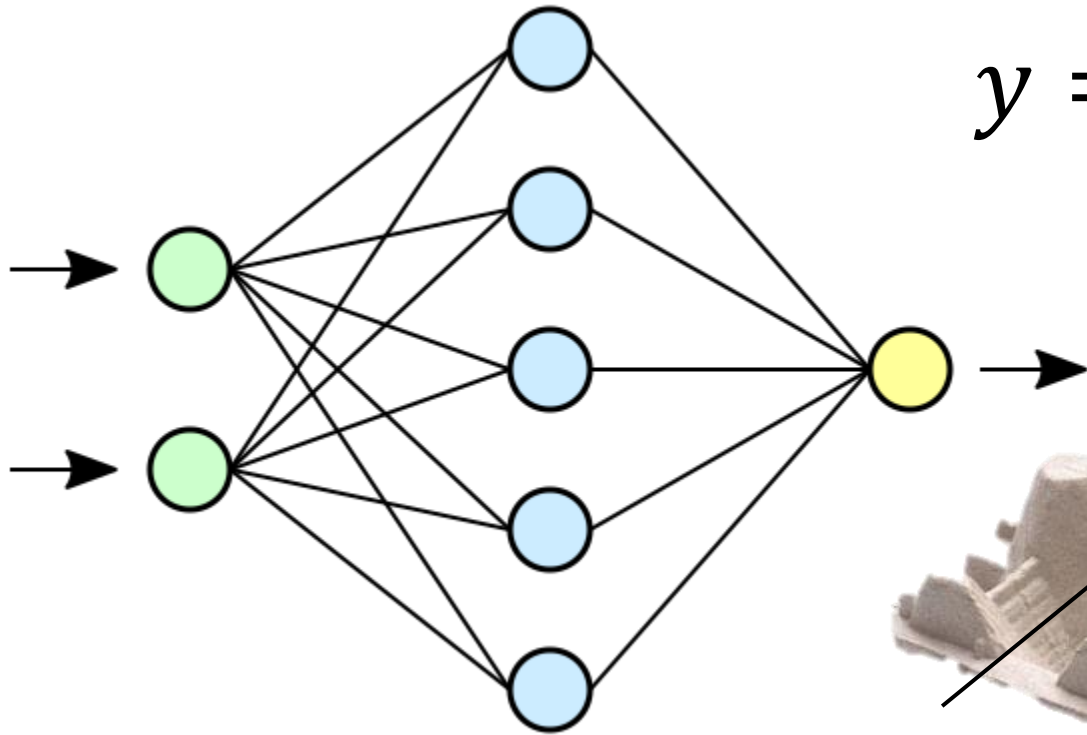
- Guarantees?

**No!**



# SGD - Remarks

$$y = \sigma(W_2 \cdot \sigma(W_1 \cdot x))$$
$$y = \sigma(W_2 P^T \cdot \sigma(PW_1 \cdot x))$$



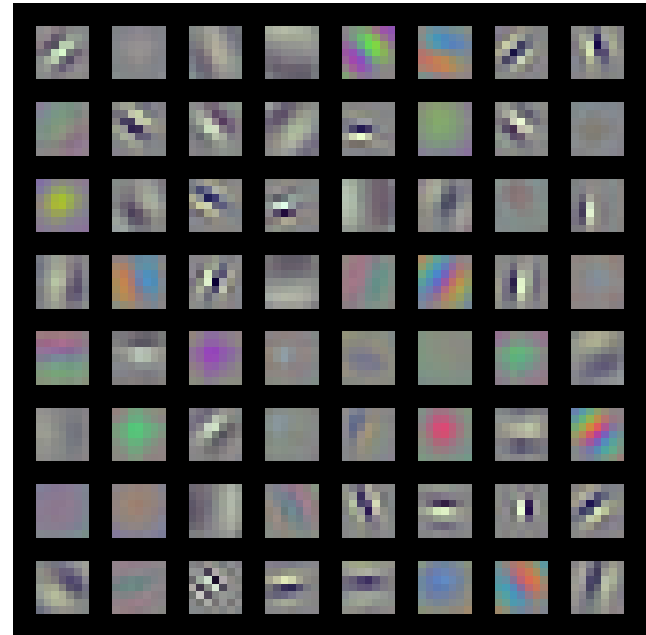
# SGD – Starting point

What if we init  $\theta = 0$ ?

$$y = W^T x$$

$$\frac{\partial y}{\partial x} = W$$

$$\frac{\partial y}{\partial W} = x^T$$

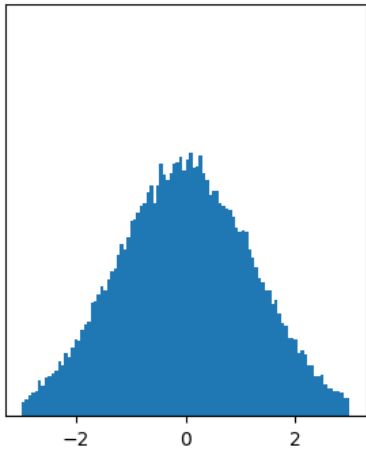


# SGD – Starting point

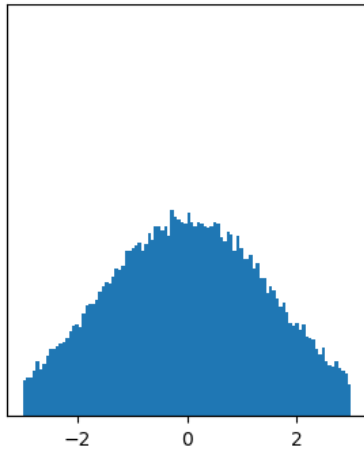
MLP, 6 layers, hidden dim=4096, no activation

$$w_{ij} \sim N(0, 0.02^2)$$

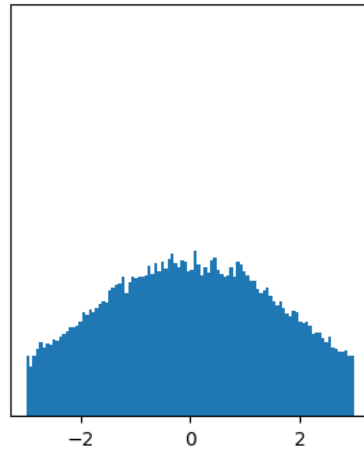
Layer 1  
mean=-0.01  
std=1.28



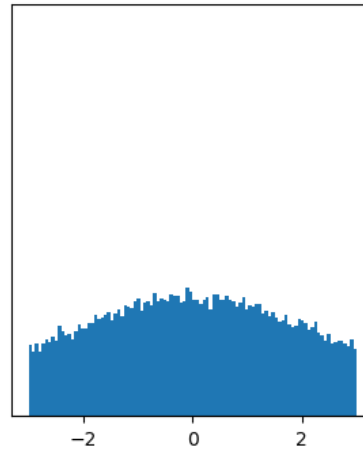
Layer 2  
mean=-0.00  
std=1.64



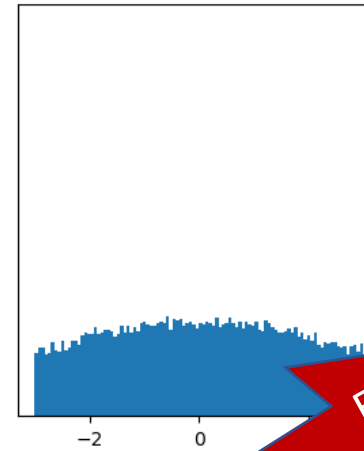
Layer 3  
mean=0.00  
std=2.10



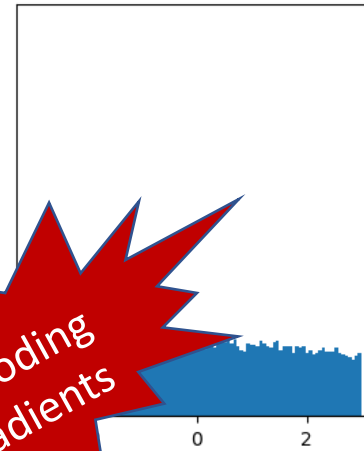
Layer 4  
mean=0.01  
std=2.70



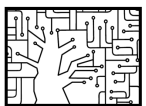
Layer 5  
mean=0.01  
std=3.44



Layer 6  
mean=-0.00  
std=4.40



Exploding  
gradients

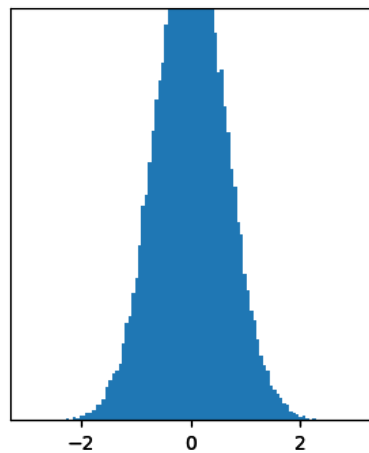


# SGD – Starting point

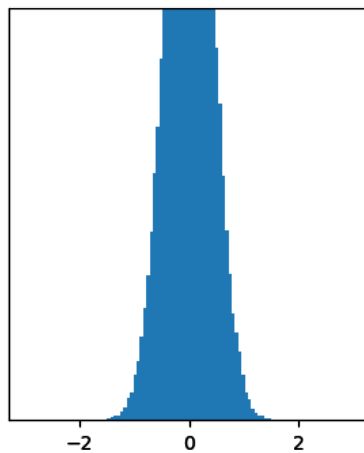
MLP, 6 layers, hidden dim=4096, no activation

$$w_{ij} \sim N(0, 0.01^2)$$

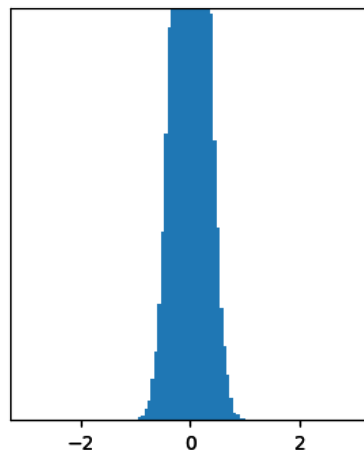
Layer 1  
mean=0.00  
std=0.08



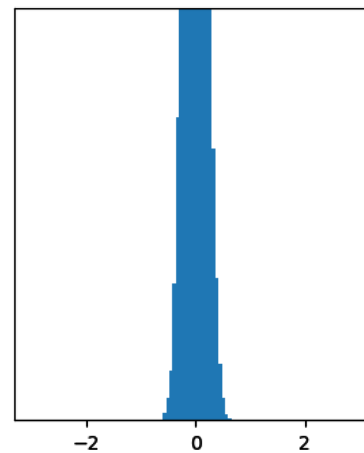
Layer 2  
mean=0.00  
std=0.14



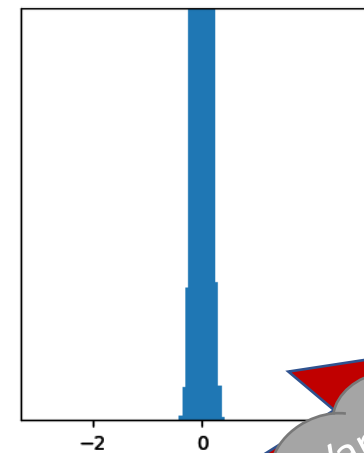
Layer 3  
mean=0.00  
std=0.26



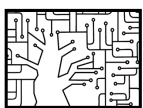
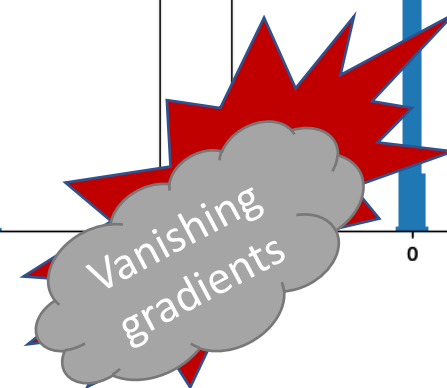
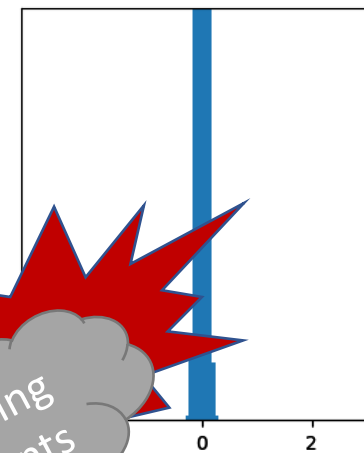
Layer 4  
mean=0.00  
std=0.50



Layer 5  
mean=0.00  
std=1.00



Layer 6  
mean=0.00  
std=2.00





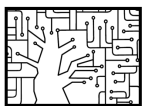
# SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

$$y = W^T x$$

$$\text{var}(y_i) = \text{var}(W_i^T x)$$

$$\text{var}(w) = 1/D_{in} \rightarrow \sigma = \sqrt{1/D_{in}}$$

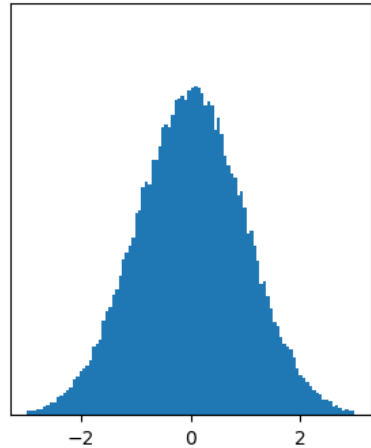


# SGD – Starting point

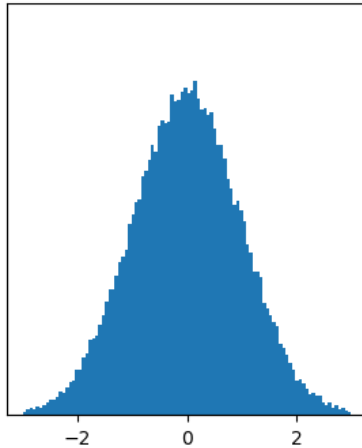
MLP, 6 layers, hidden dim=4096, no activation

$$w_{ij} \sim N\left(0, 1/D_{in}\right)$$

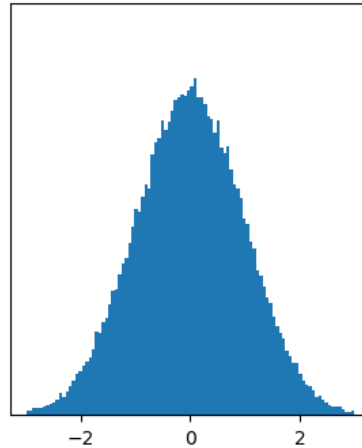
Layer 1  
mean=0.01  
std=0.99



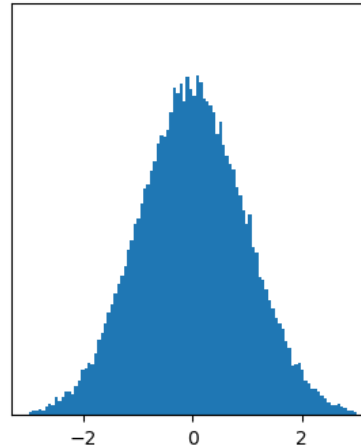
Layer 2  
mean=-0.00  
std=1.00



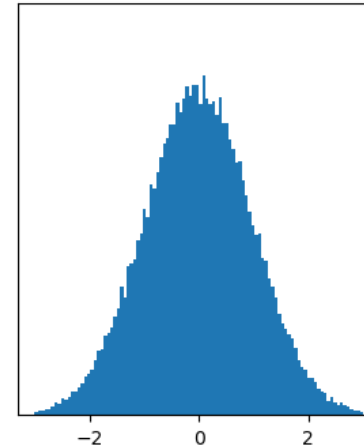
Layer 3  
mean=-0.00  
std=0.99



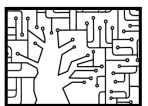
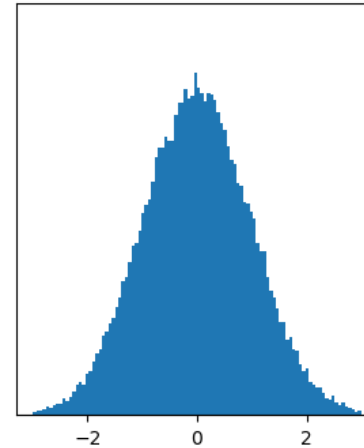
Layer 4  
mean=-0.00  
std=0.99



Layer 5  
mean=-0.01  
std=0.99



Layer 6  
mean=0.00  
std=0.99

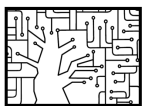
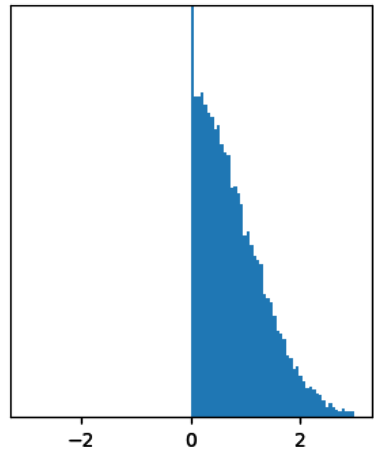


# SGD – Starting point

MLP, 6 layers, hidden dim=4096, **ReLU** activation

$$w_{ij} \sim N\left(0, 1/D_{in}\right)$$

Layer 1  
mean=0.00  
std=0.99

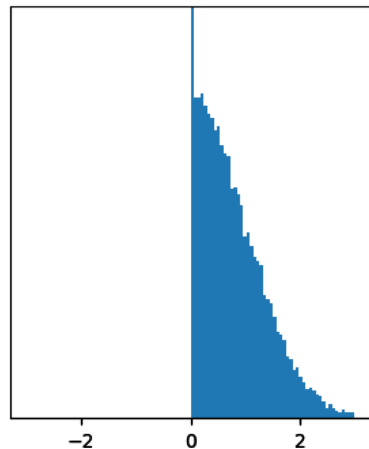


# SGD – Starting point

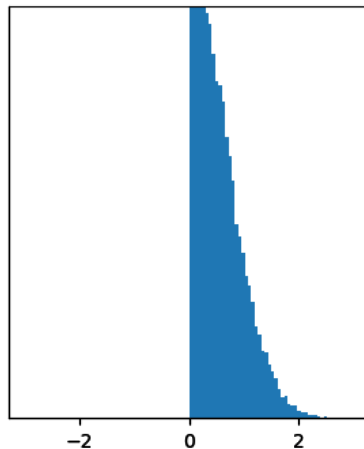
MLP, 6 layers, hidden dim=4096, **ReLU** activation

$$w_{ij} \sim N\left(0, 2/D_{in}\right)$$

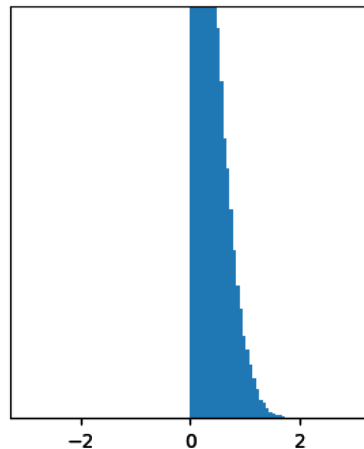
Layer 1  
mean=0.56  
std=0.82



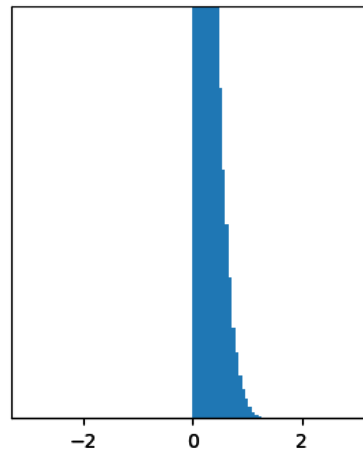
Layer 2  
mean=0.38  
std=0.83



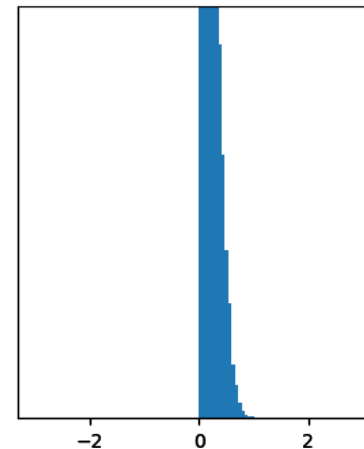
Layer 3  
mean=0.26  
std=0.89



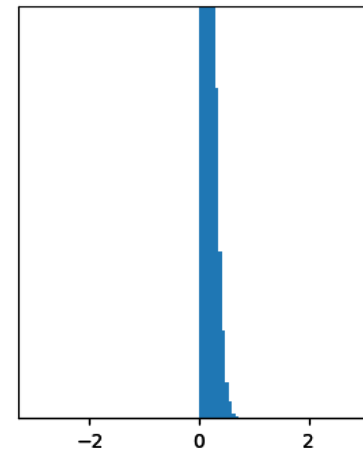
Layer 4  
mean=0.16  
std=0.83



Layer 5  
mean=0.10  
std=0.85

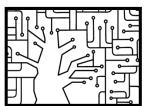


Layer 6  
mean=0.07  
std=0.80



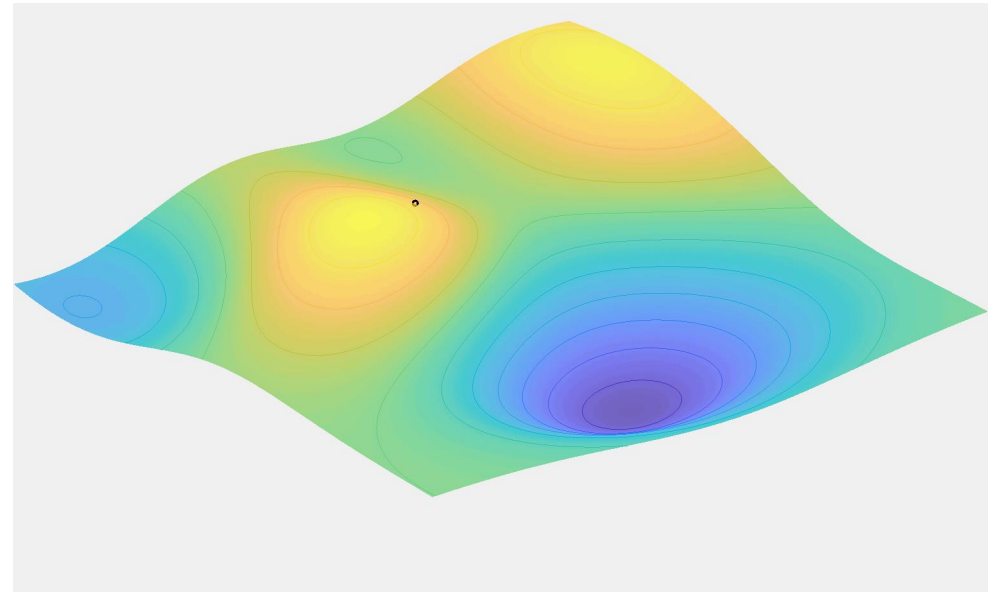
# SGD – Starting point

- Where we start the SGD is crucial
- Init depends on the architecture
- Xavier/Kaiming can be easily extended to Conv layers
- Inputs should be normalized to  $\sim N(0, 1)$

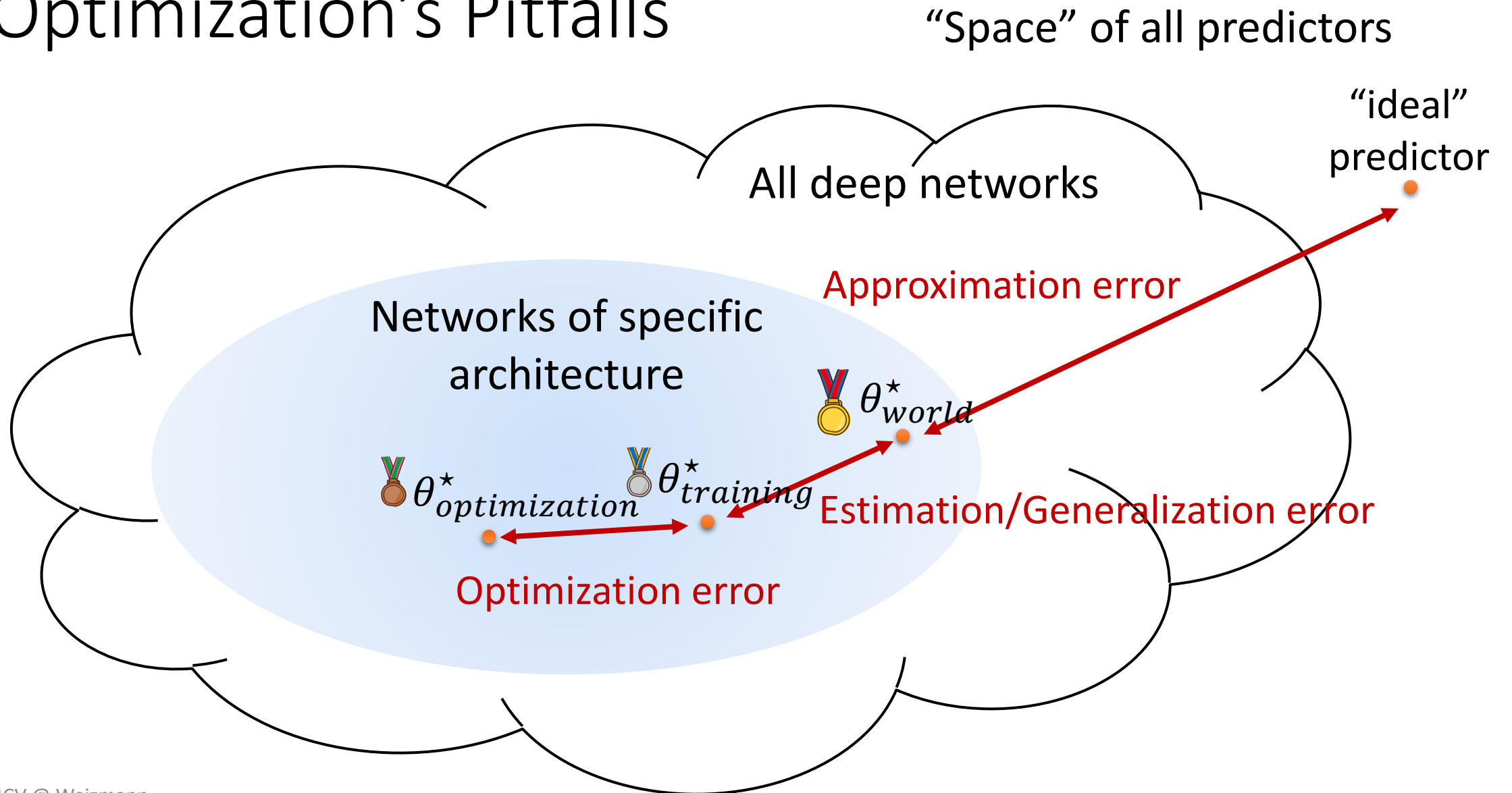


# Agenda

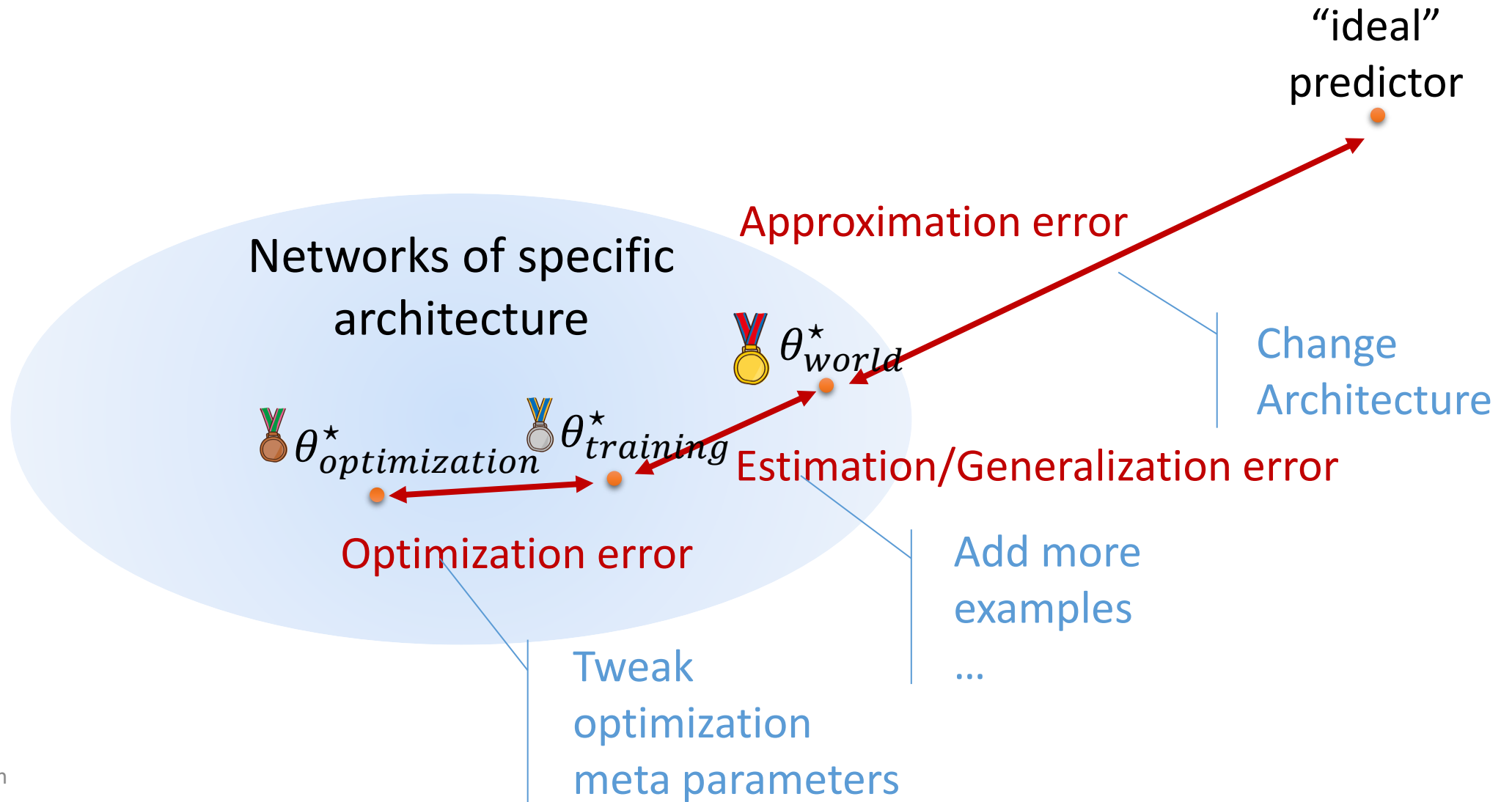
- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



# Optimization's Pitfalls



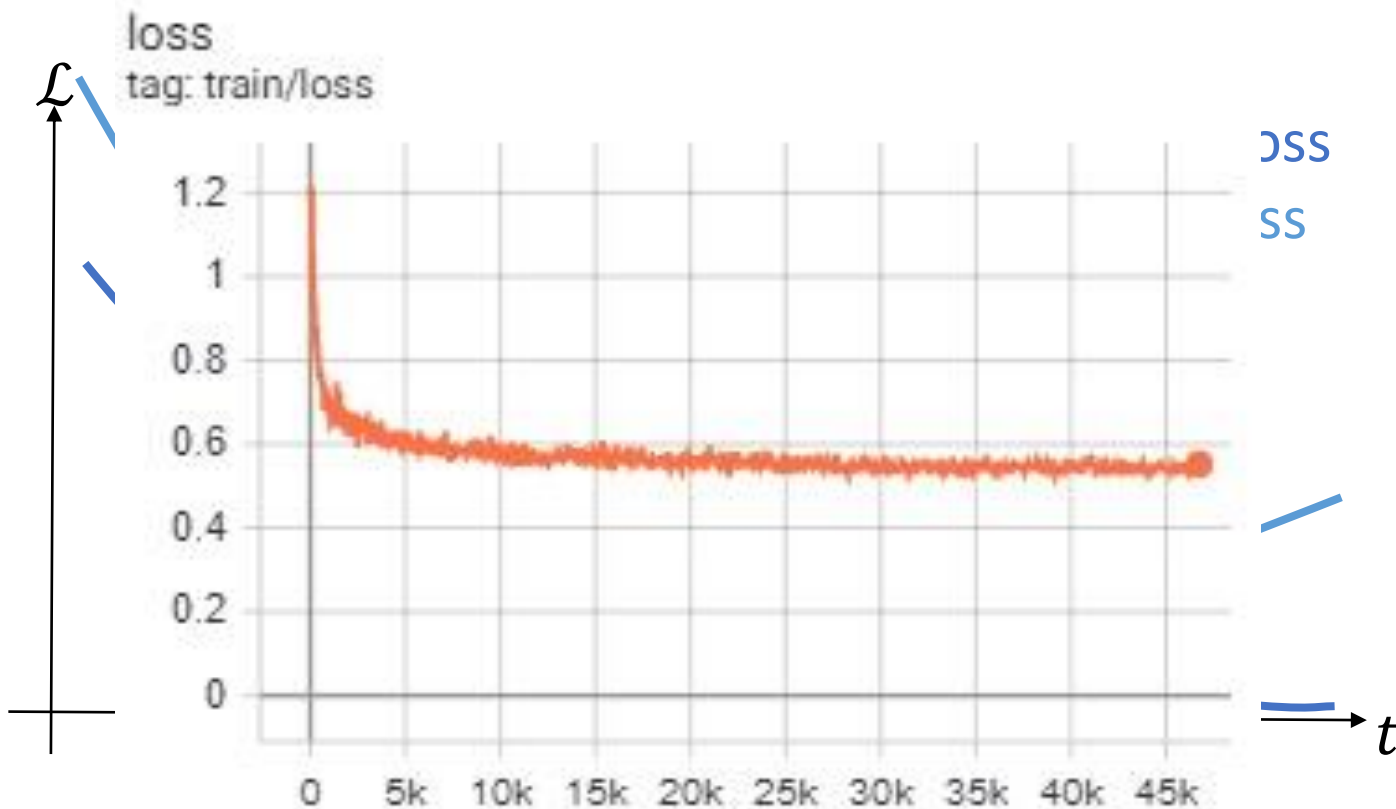
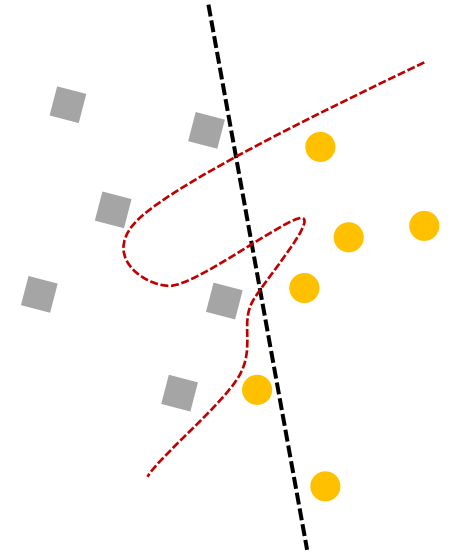
# Optimization's Pitfalls





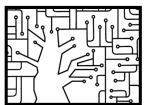
# Optimization's Pitfalls

- How to look at the loss-vs-iterations for train/test set?
- Overfitting



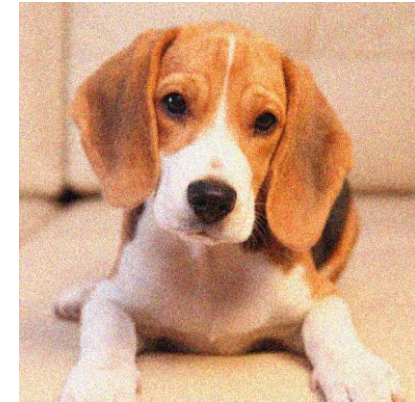
# Regularization

$$\mathcal{L}(\theta; \mathbf{X}) = CE(\theta; \mathbf{X}) + \lambda \|\theta\|_p$$



# Regularization

- Data augmentation  
dog

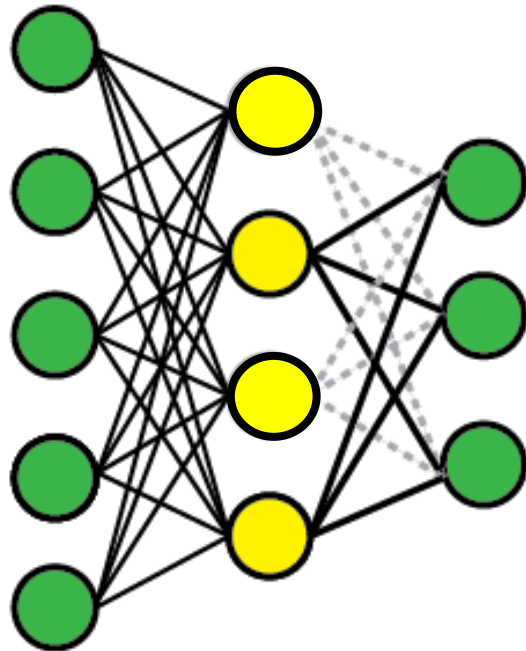


**Torchvision:** `transforms`

**Albumentations:** <https://github.com/albumentations-team/albumentations>

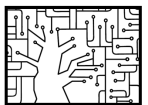
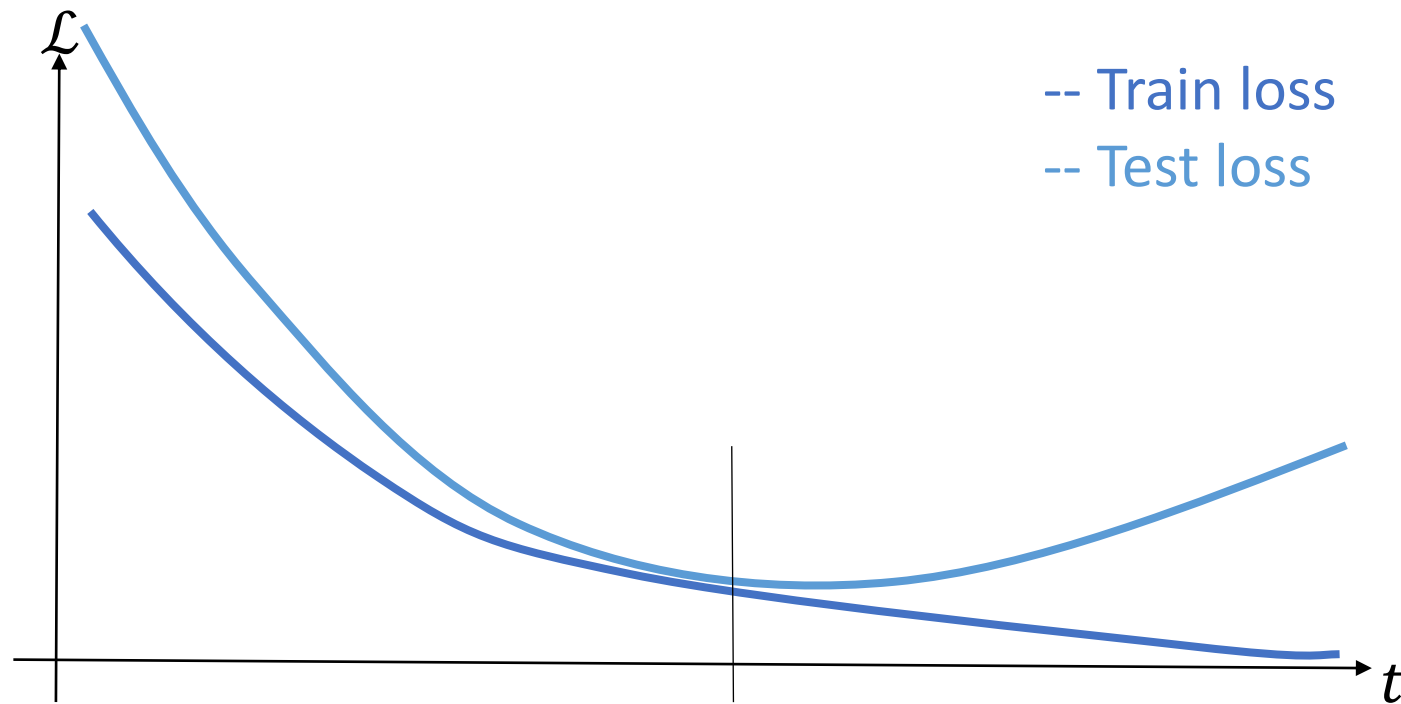
# Regularization

## Dropout



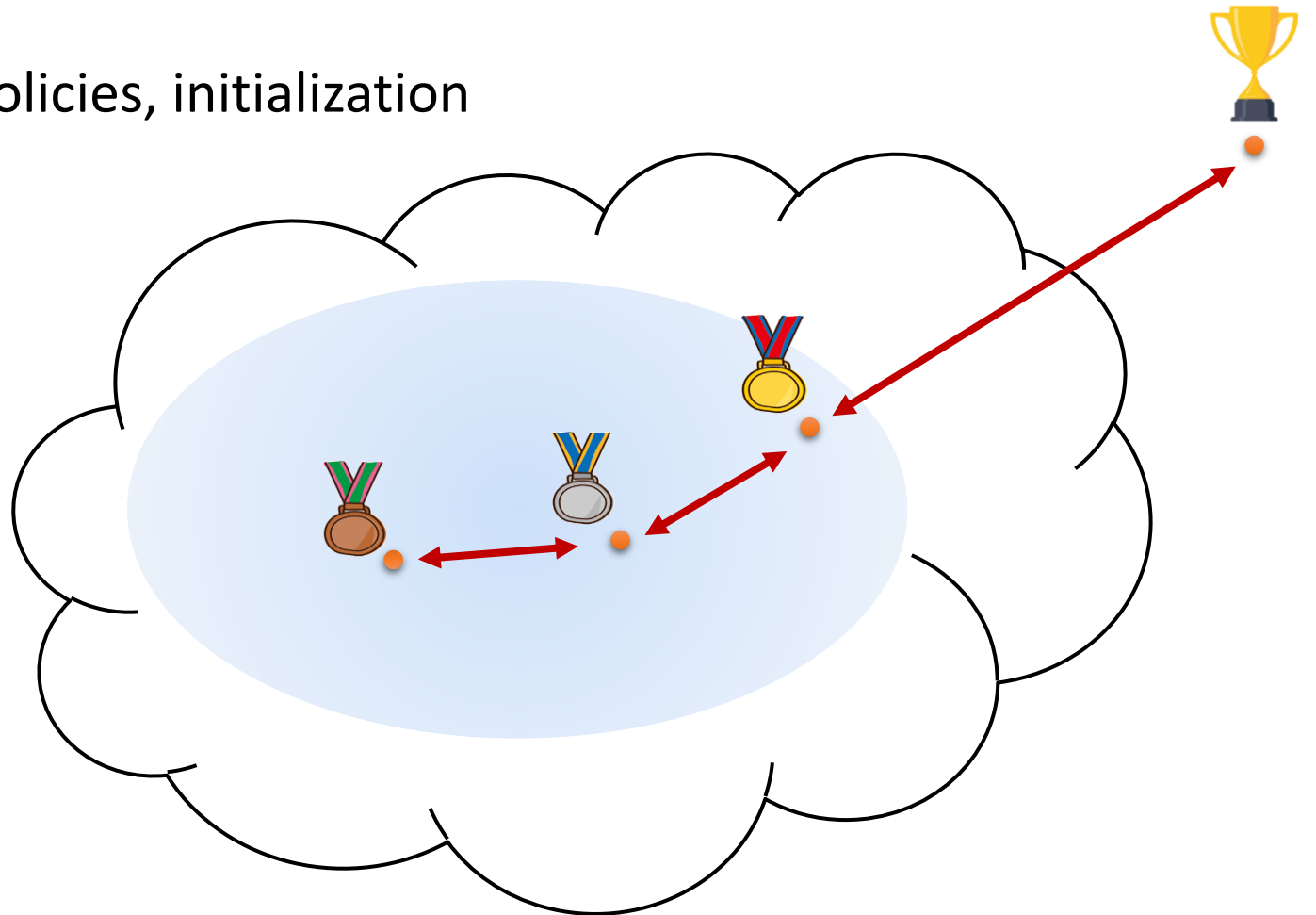
# Regularization

## Early Stopping



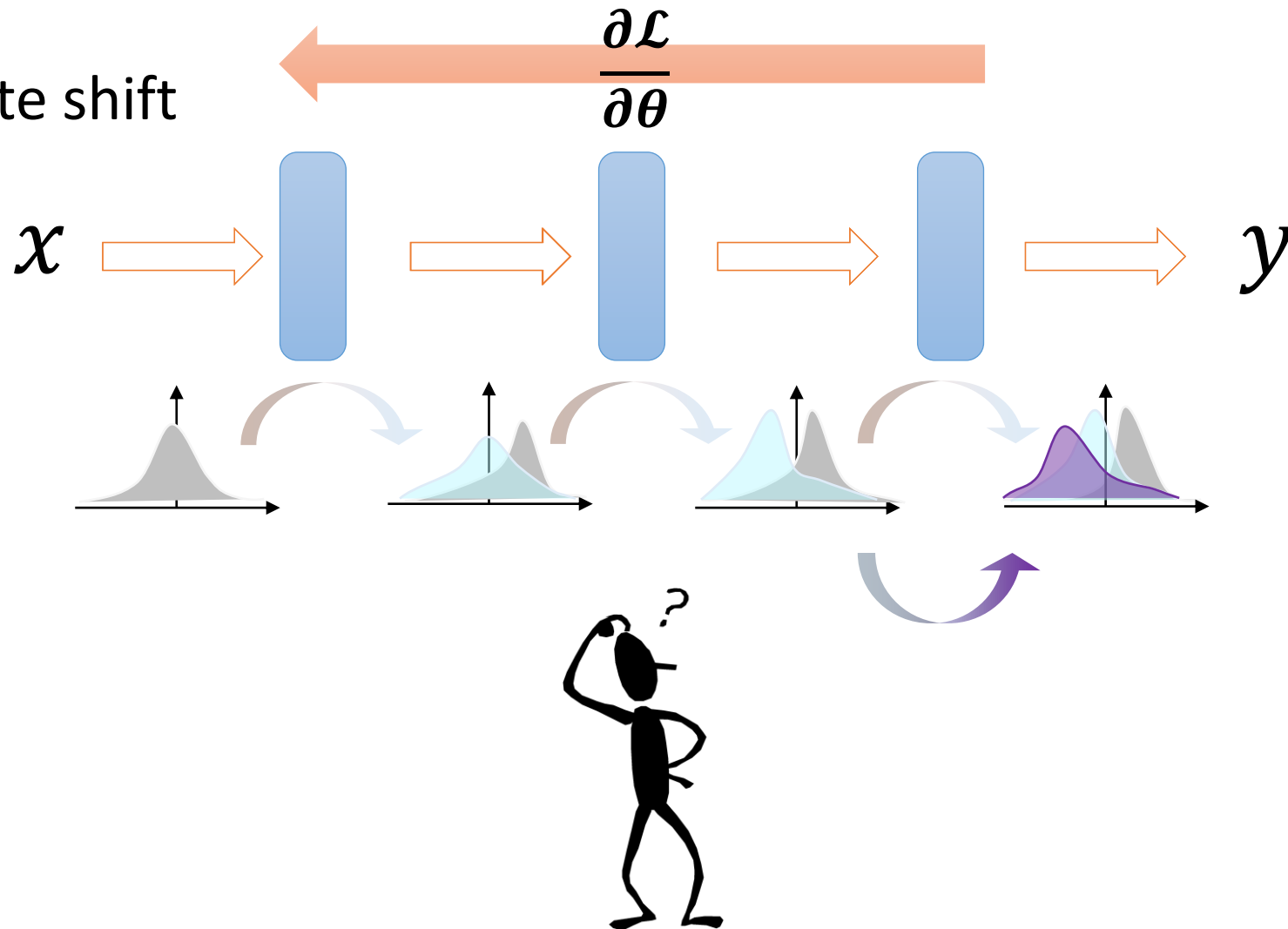
# Agenda

- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



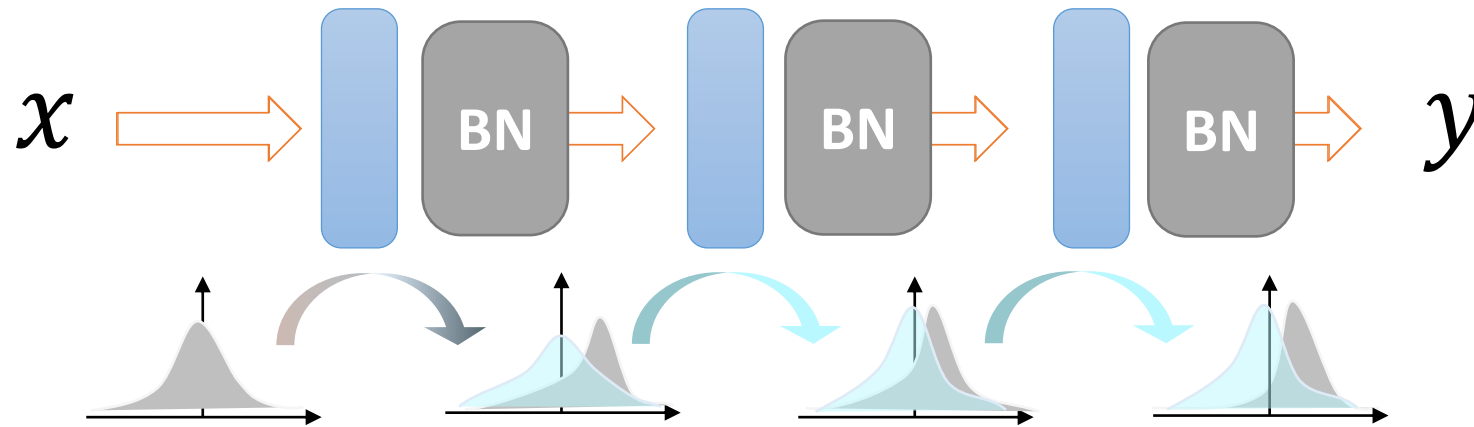
# Normalization Layers

- Covariate shift



# Normalization Layers

- Batch Norm!





# Normalization Layers

- Batch Norm

Input:  $\mathbf{x} \in \mathbb{R}^{N \times C}$

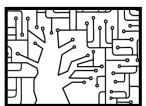
Estimate  $\mu, \sigma \in \mathbb{R}^C$

Per-channel mean  $\mu_c = \frac{1}{N} \sum_b x_{bc}$

Per-channel var  $\sigma^2 = \frac{1}{N} \sum_b (x_{bc} - \mu_c)^2$

Learnable  $\gamma, \beta \in \mathbb{R}^C$

Output:  $\mathbf{y} = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta$



# Normalization Layers

- Batch Norm

Input:  $\mathbf{x} \in \mathbb{R}^{N \times C}$

Estimate  $\mu, \sigma \in \mathbb{R}^C$

Test time?

Per-channel mean  $\mu_c = \frac{1}{N} \sum_b x_{bc}$

Per-channel var  $\sigma^2 = \frac{1}{N} \sum_b (x_{bc} - \mu_c)^2$

Learnable  $\gamma, \beta \in \mathbb{R}^C$

Output:  $\mathbf{y} = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta$

# Normalization Layers

- Batch Norm

Input:  $\mathbf{x} \in \mathbb{R}^{N \times C}$

Estimate  $\mu, \sigma \in \mathbb{R}^C$

Keep track  $\hat{\mu}, \hat{\sigma} \in \mathbb{R}^C$

Learnable  $\gamma, \beta \in \mathbb{R}^C$

Output:  $\mathbf{y} = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta$

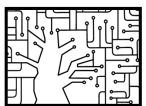
## Training time:

Updated in  
**forward pass**

$$\mu = \frac{1}{N} \sum_b x_{bc}$$
$$\sigma^2 = \frac{1}{N} \sum_b (x_{bc} - \mu_c)^2$$

Updated in  
**backward pass**

$$\hat{\mu} = 0.9\hat{\mu} + 0.01\mu$$



# Normalization Layers

- Batch Norm

Input:  $\mathbf{x} \in \mathbb{R}^{N \times C}$

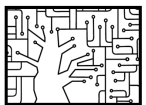
**Test time:**

Keep track  $\hat{\mu}, \hat{\sigma} \in \mathbb{R}^C$

Learnable  $\gamma, \beta \in \mathbb{R}^C$

Output:  $\mathbf{y} = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta$

Output:  $\mathbf{y} = \frac{\mathbf{x} - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon}} \gamma + \beta$



# Normalization Layers

- Batch Norm

Input:  $\mathbf{x} \in \mathbb{R}^{N \times C}$

Estimate  $\mu, \sigma \in \mathbb{R}^C$

Learnable  $\gamma, \beta \in \mathbb{R}^C$

Output:  $\mathbf{y} = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta$

For Conv layers

$\mathbf{x}: N \times C \times H \times W$

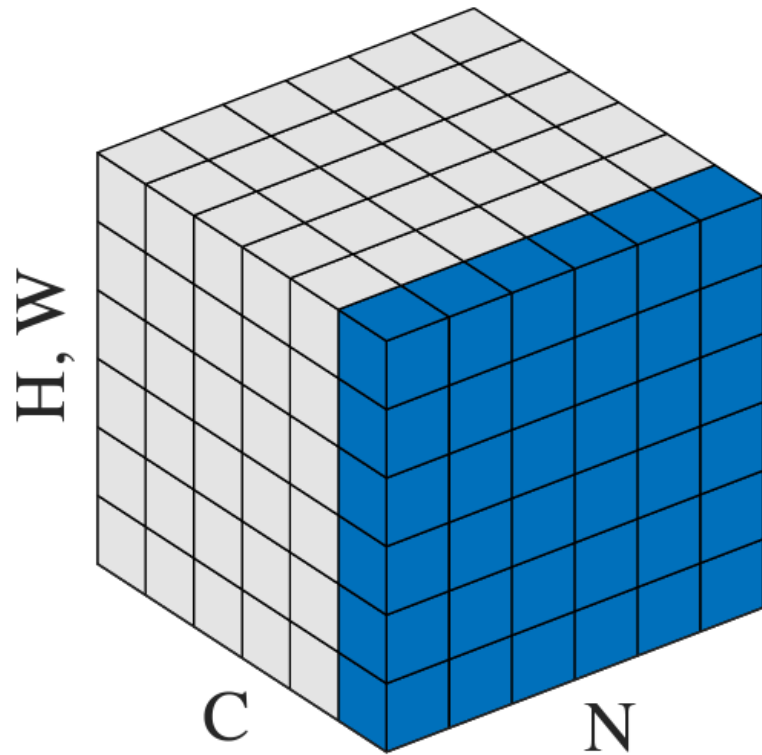


$\mu, \sigma: 1 \times C \times 1 \times 1$

$\gamma, \beta: 1 \times C \times 1 \times 1$

# Normalization Layers

Batch Norm



For Conv layers

$$\mathbf{x}: N \times C \times H \times W$$



$$\mu, \sigma: 1 \times C \times 1 \times 1$$

$$\gamma, \beta: 1 \times C \times 1 \times 1$$

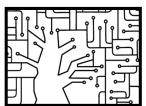
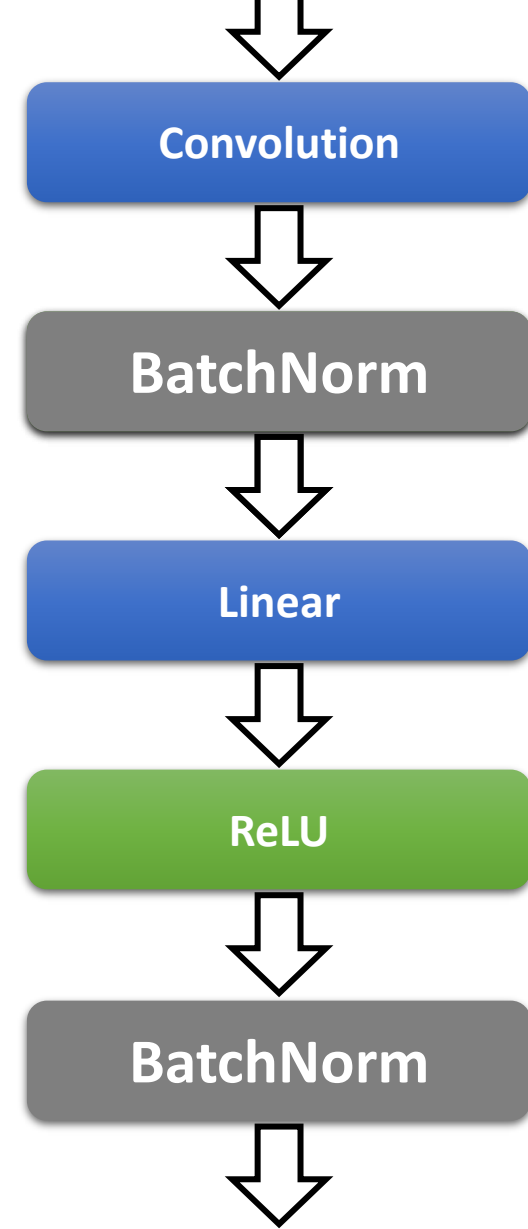
# Normalization Layers

## Batch Norm

- + Training more robust to init
- + Allows for training deeper nets
- + Allows for higher learning rates

H, W

- Different train/test behavior



# Normalization Layers

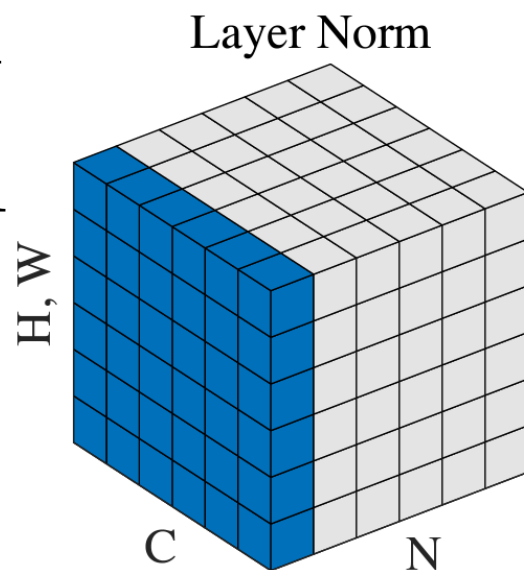
LayerNorm

$$\mathbf{x}: N \times C \times H \times W$$



$$\mu, \sigma: N \times 1 \times 1 \times 1$$

$$\gamma, \beta: 1 \times C \times 1 \times 1$$



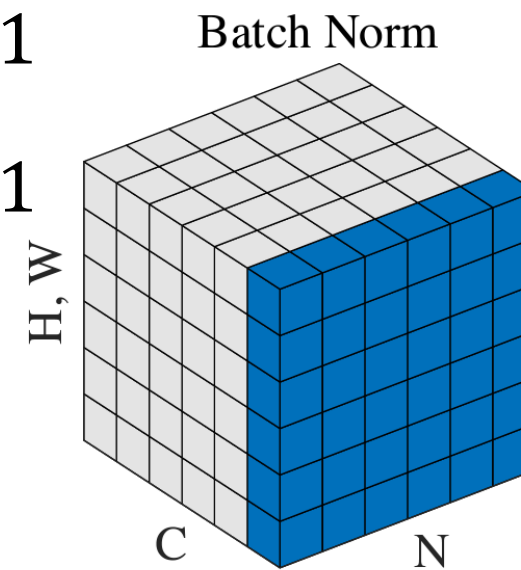
BatchNorm

$$\mathbf{x}: N \times C \times H \times W$$



$$\mu, \sigma: 1 \times C \times 1 \times 1$$

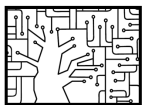
$$\gamma, \beta: 1 \times C \times 1 \times 1$$





# Agenda

- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



# What's coming up...

- Tutorial – CNN Architectures



- Lecture (next week) – Visualization and Understanding

