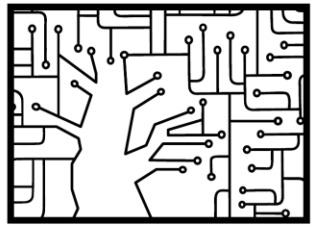
Deep Learning for Computer Vision: **Practical Training**

Shai Bagon




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

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

Stochastic Gradient Descent (SGD)

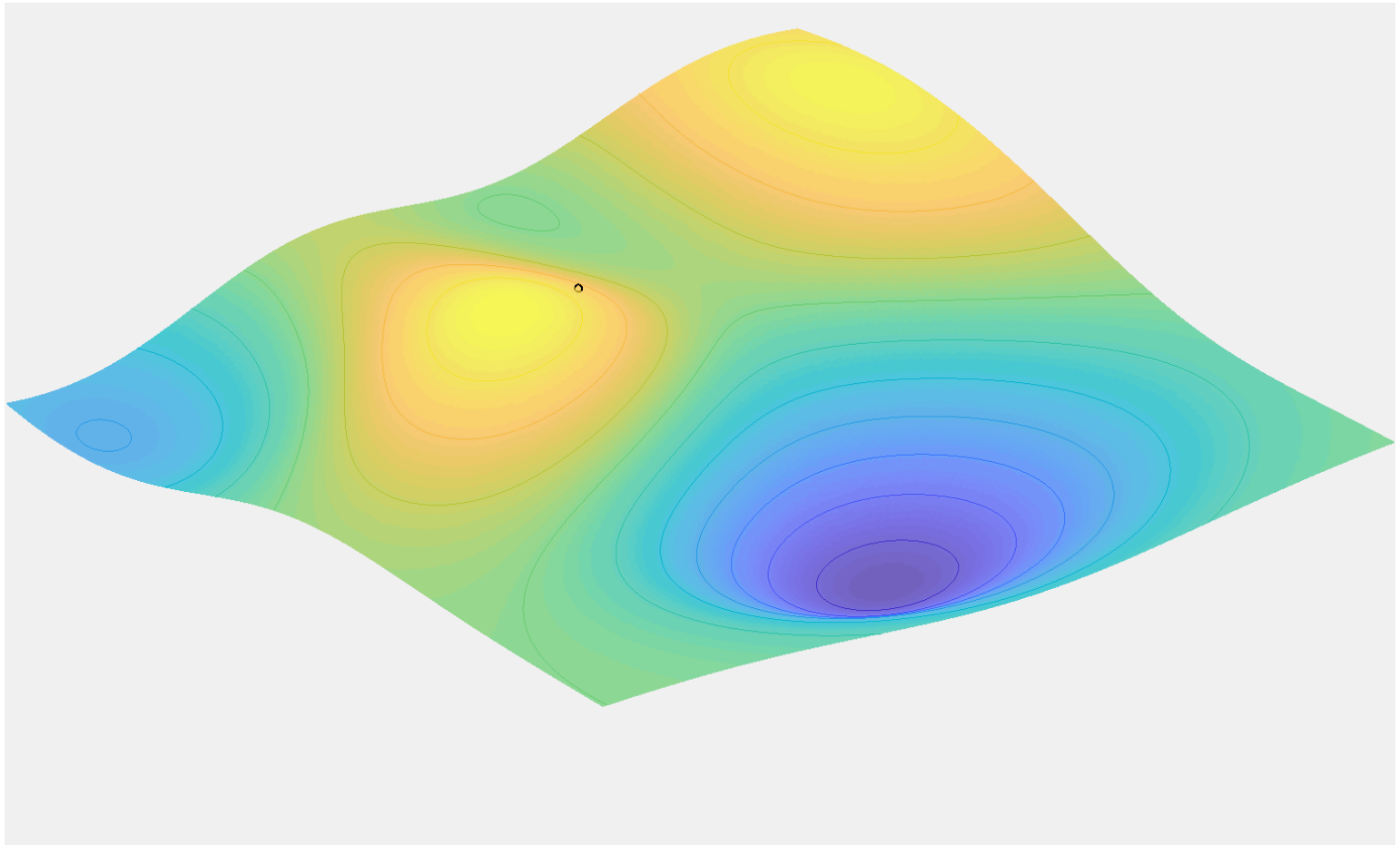
$$\mathcal{L}(\theta; \{(\boldsymbol{x}_i, y_i)\}_{i \in 1..N}) \approx \frac{1}{|B|} \sum_{i \in \{B\}} \mathcal{L}(f(x_i; \theta), y_i)$$

Update rule:

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

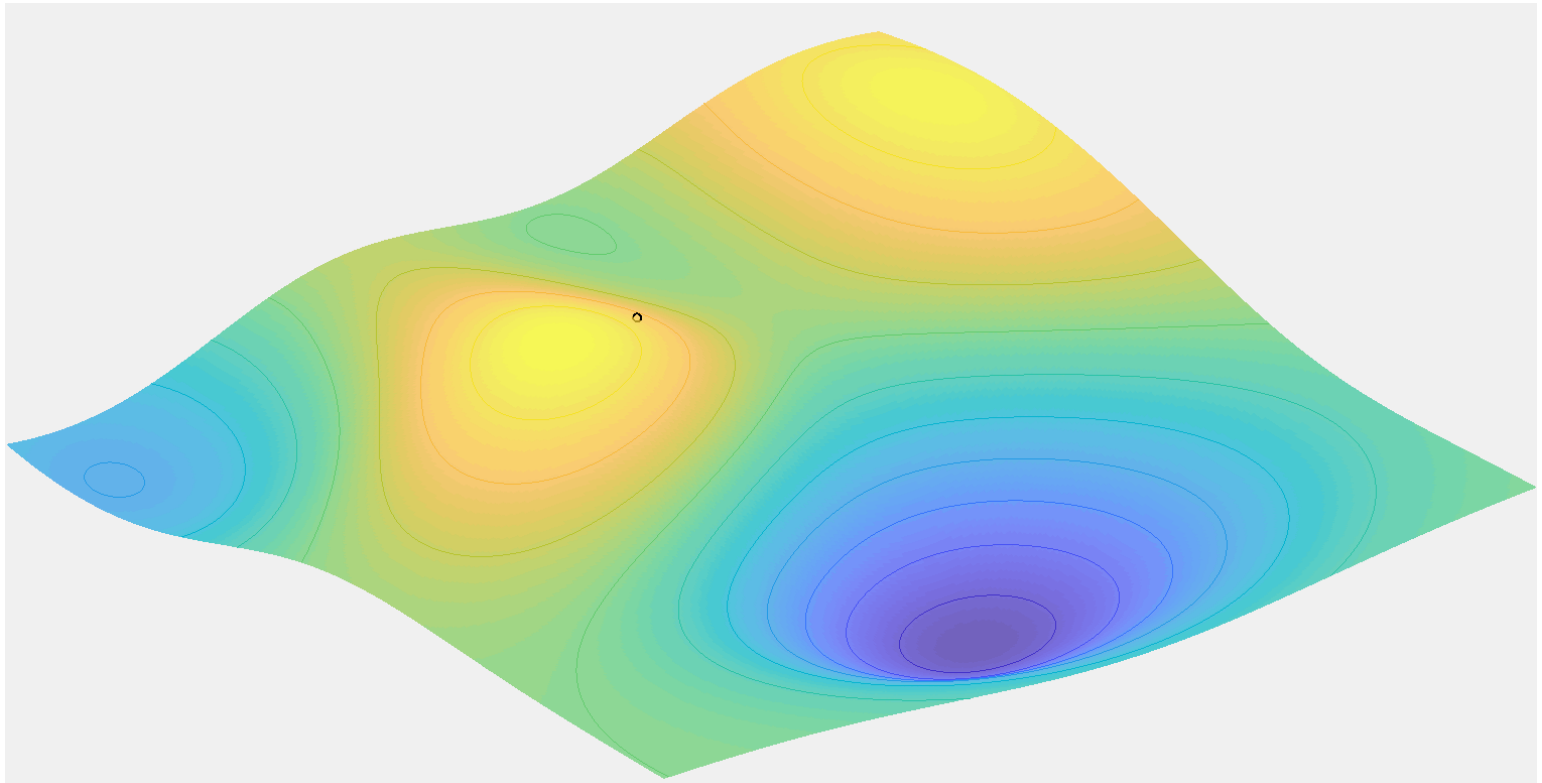
Vanilla SGD

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



Vanilla SGD

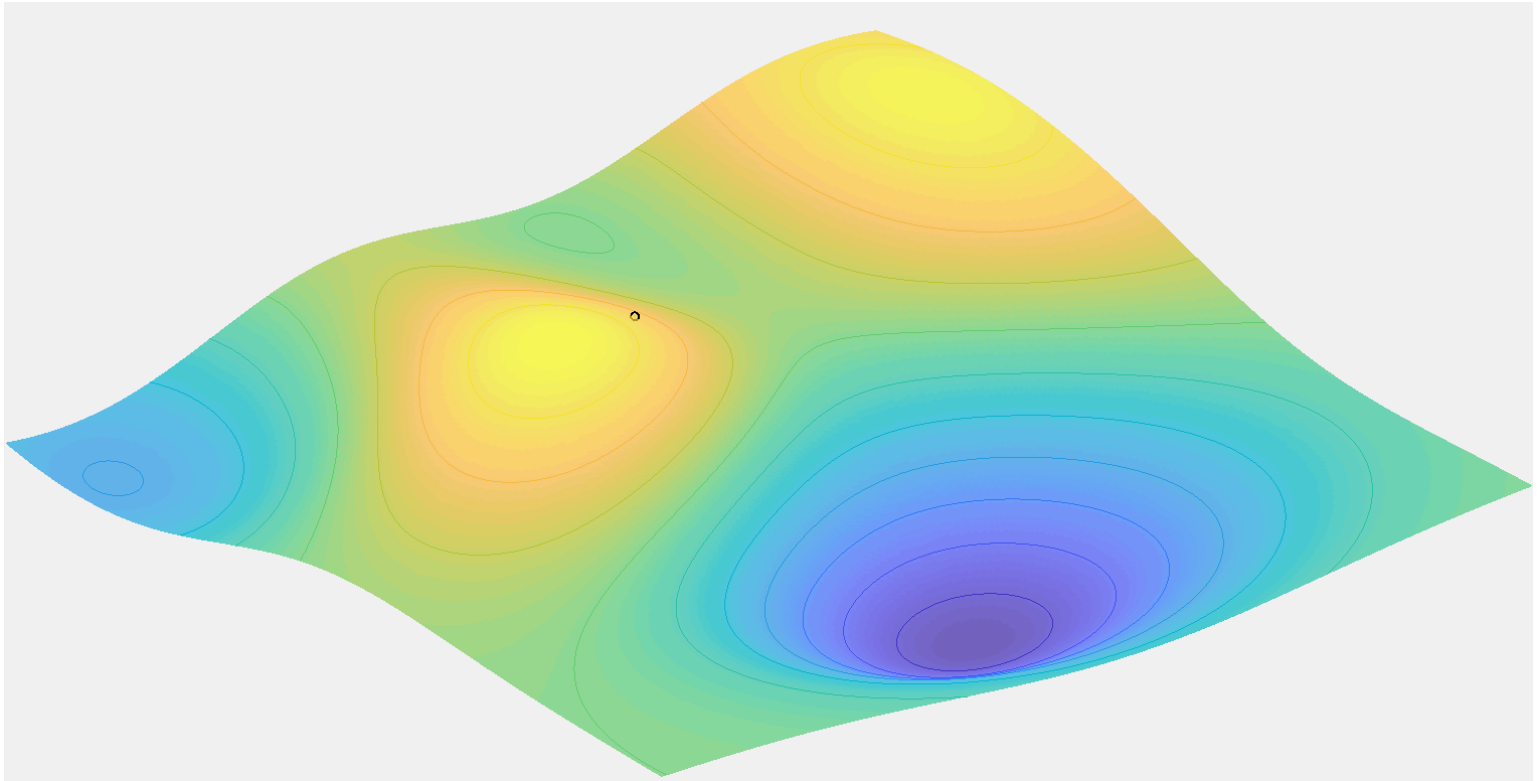
$$\theta^{t+1} = \theta^t - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^t)$$



High learning rate

Vanilla SGD

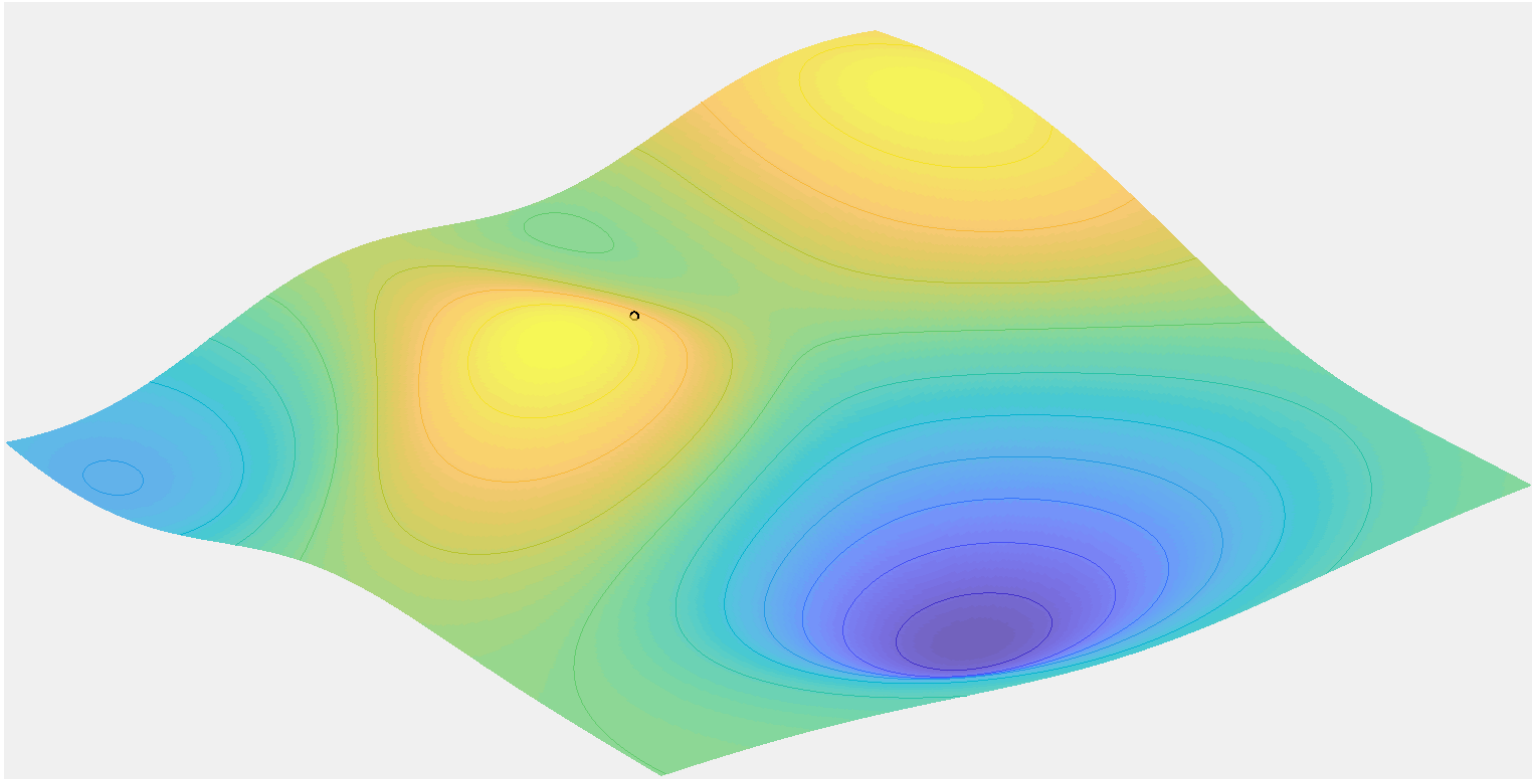
$$\theta^{t+1} = \theta^t - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^t)$$



Low learning rate

Vanilla SGD

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



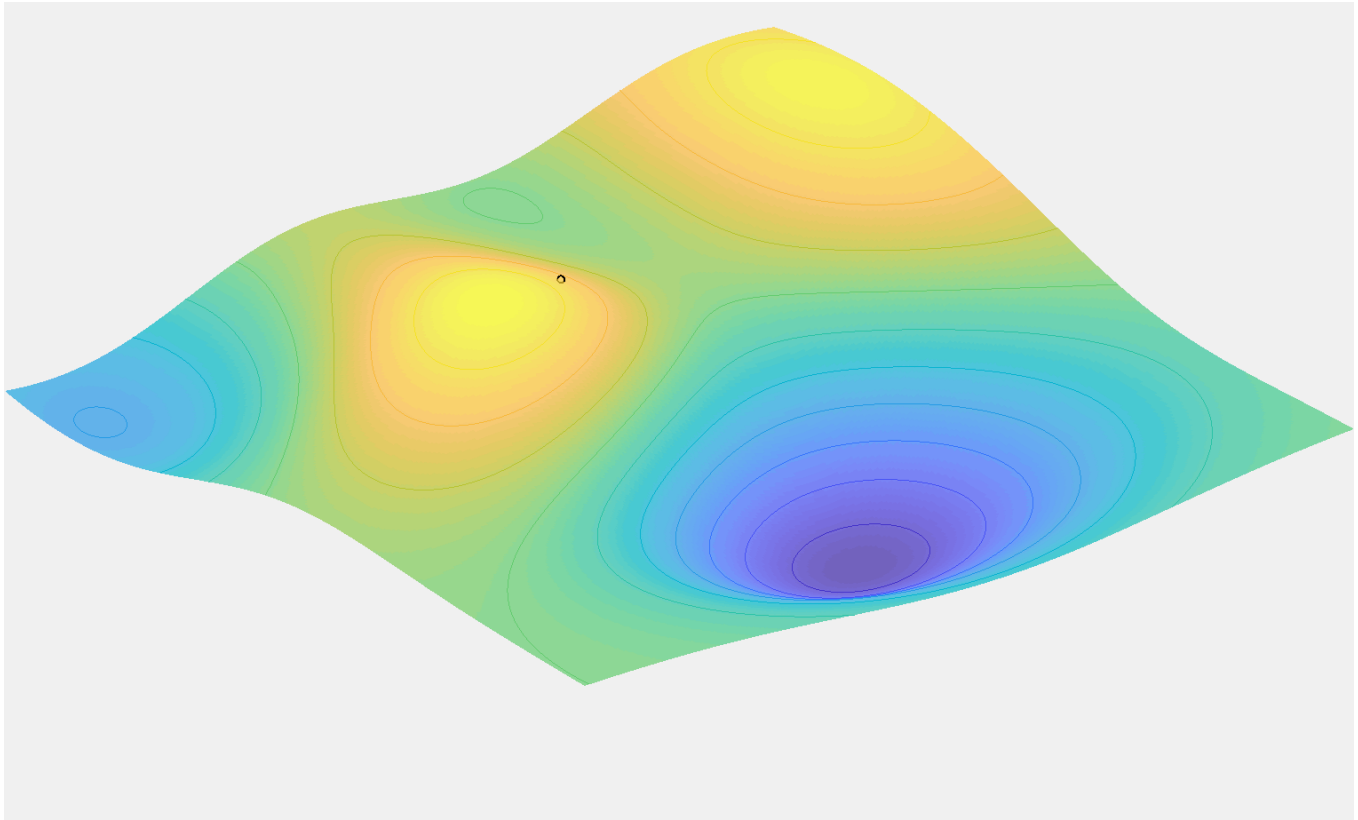
"Ideal" learning rate

Vanilla SGD

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

SGD with Momentum

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



SGD with Momentum

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

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

Adam

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

1st moment (like SGD)

Unbiased estimation

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

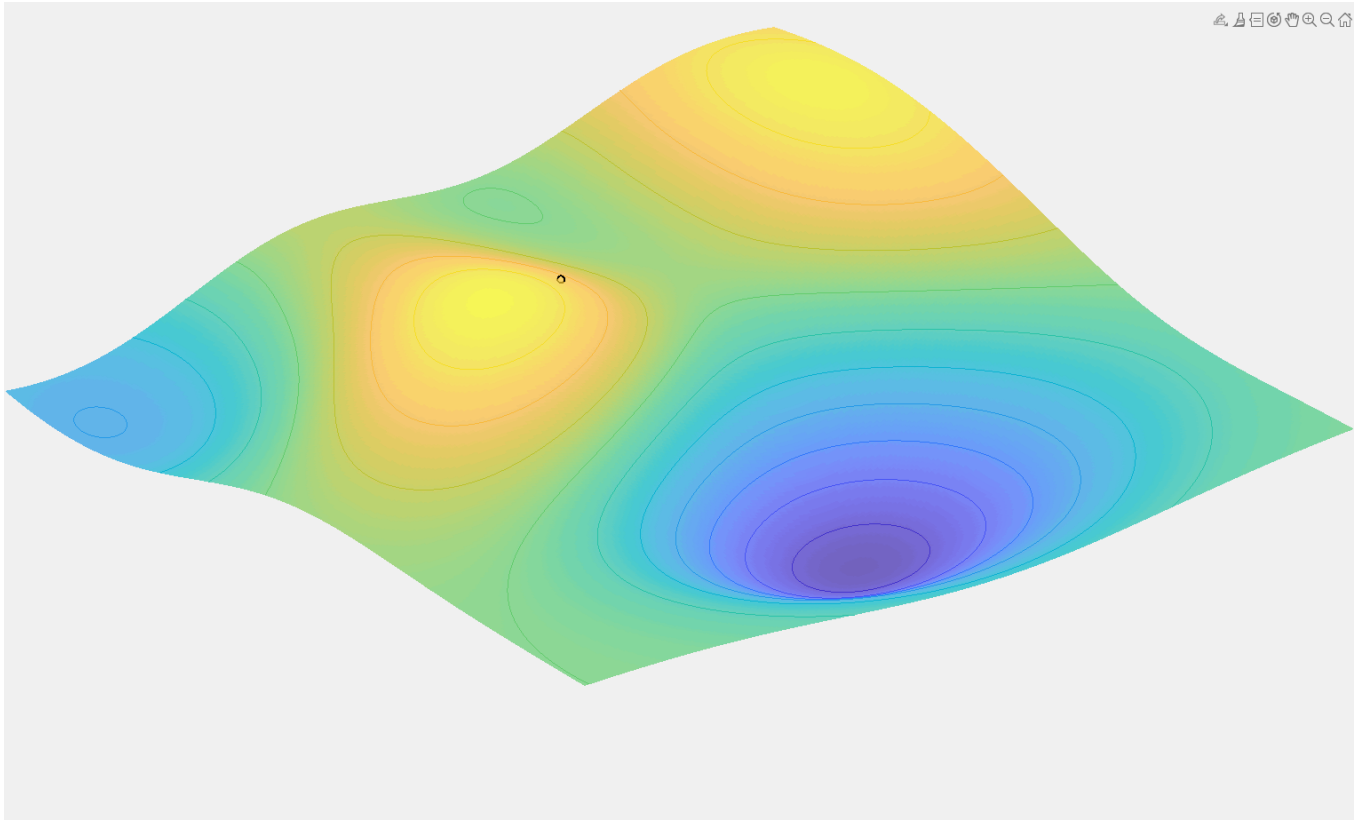
2nd moment ("variance")

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

Weight the change by the variance

Adam

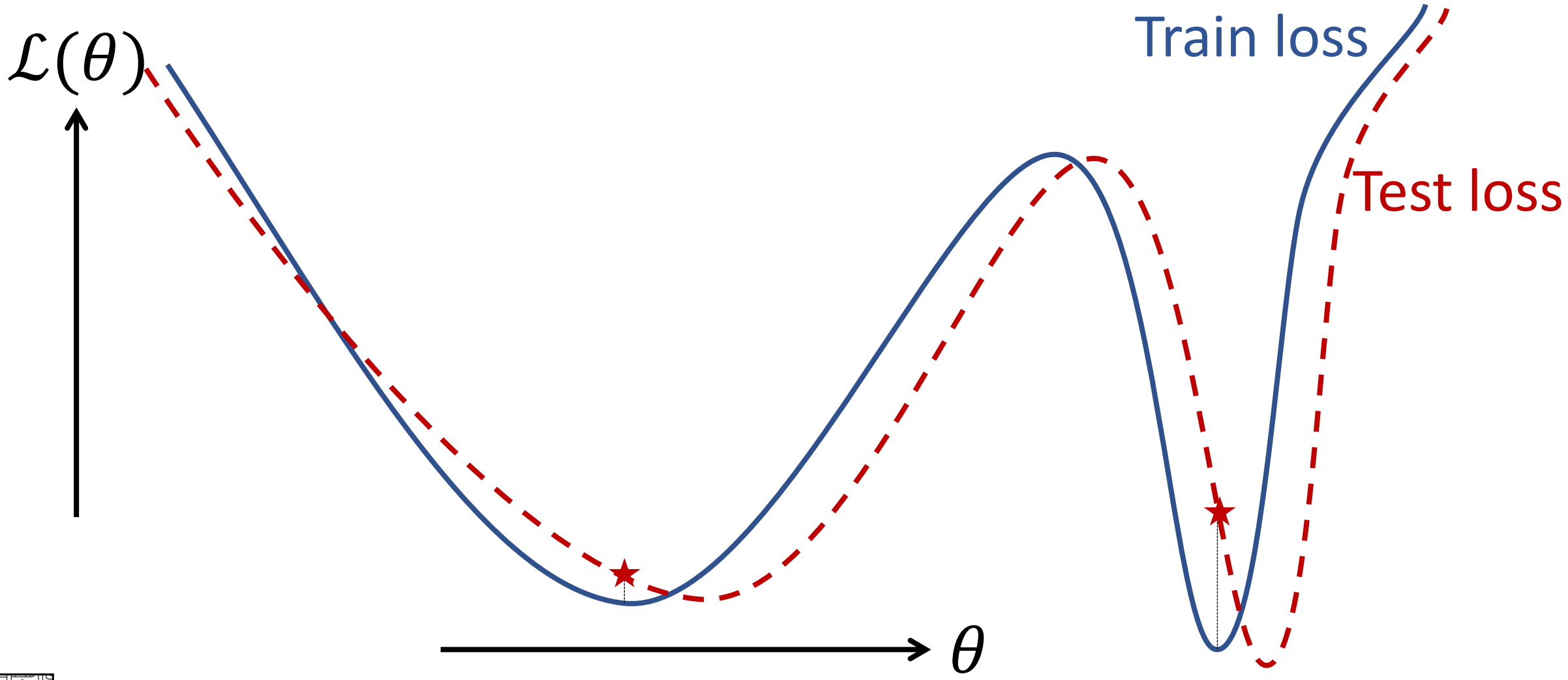
$$\theta^{t+1} = \theta^{t} - \eta \cdot v^{t+1} \Big/ \left(\sqrt{m^{t+1}} + \varepsilon\right)$$



Other SGD Update Rules

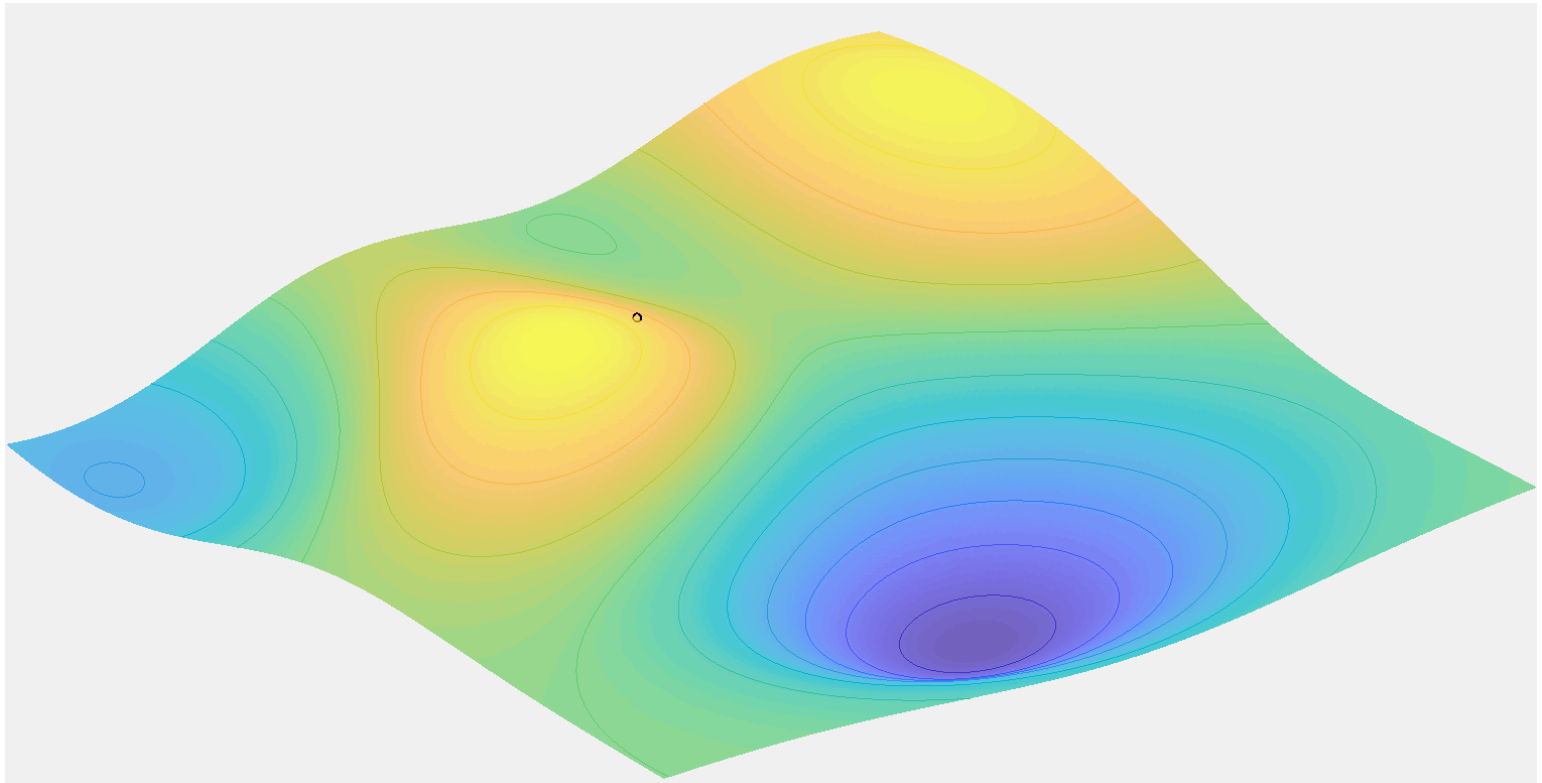
A good review can be found here.

Do we want fast convergence?



Vanilla SGD

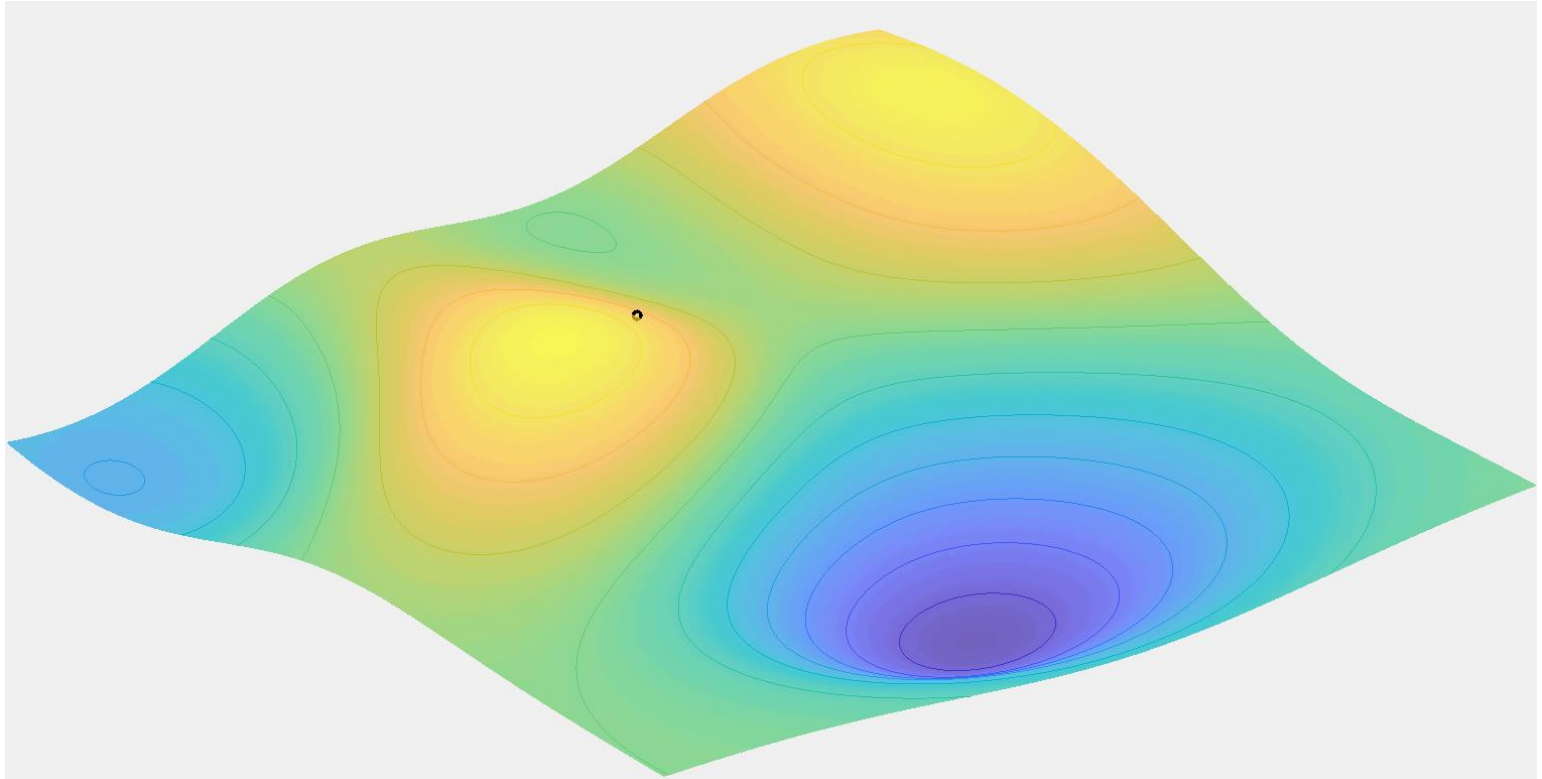
$$\theta^{t+1} = \theta^t - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^t)$$



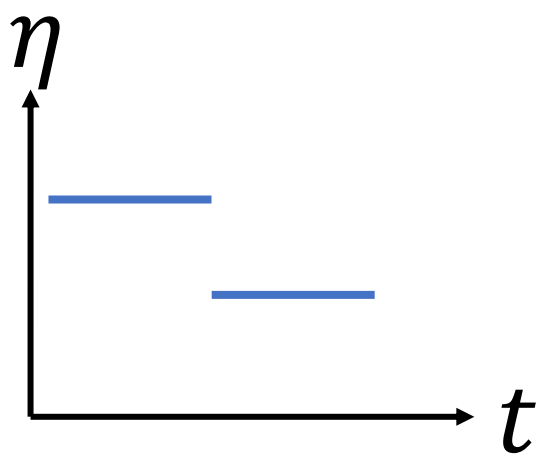
High learning rate

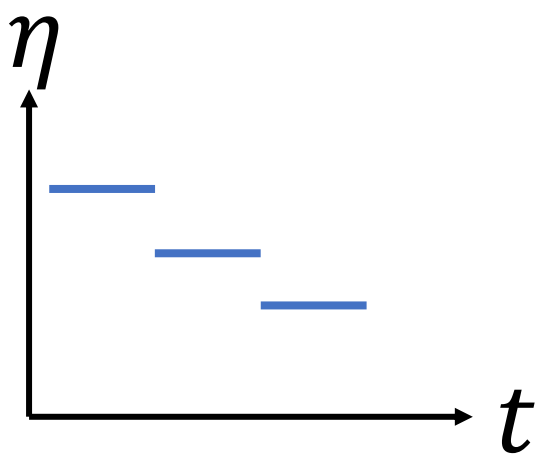
Learning Rate Decay

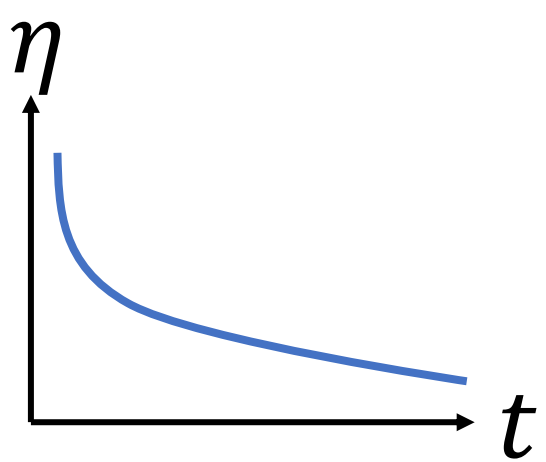
$$\theta^{t+1} = \theta^t - \boldsymbol{\eta_t} \nabla \mathcal{L}(\theta^t)$$



Learning Rate Decay







SGD - Remarks

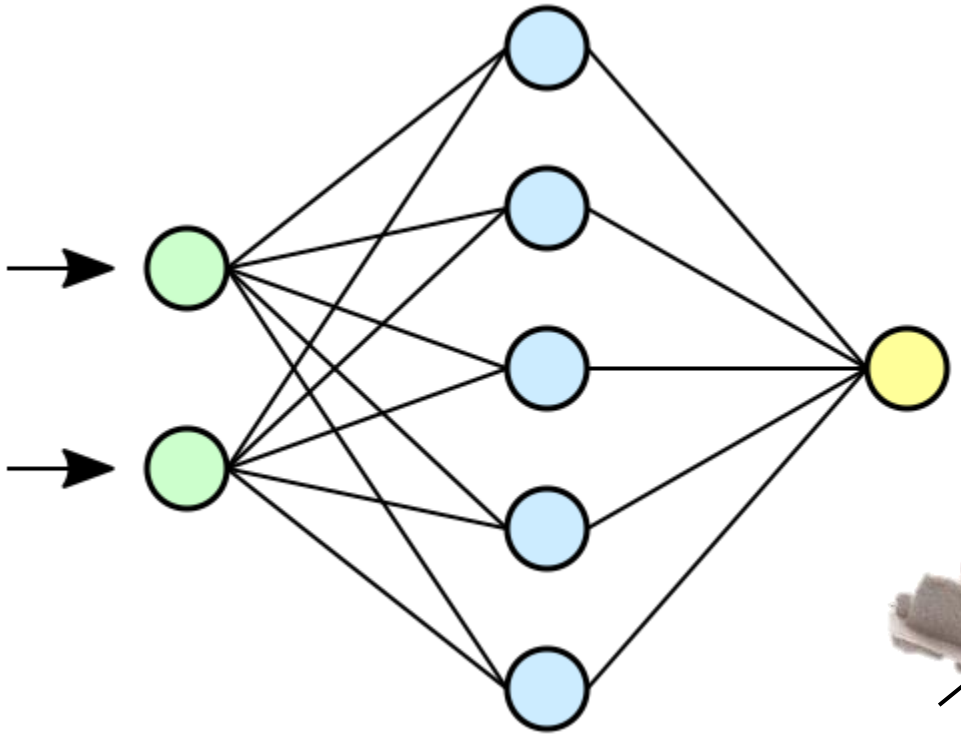
- Guarantees?

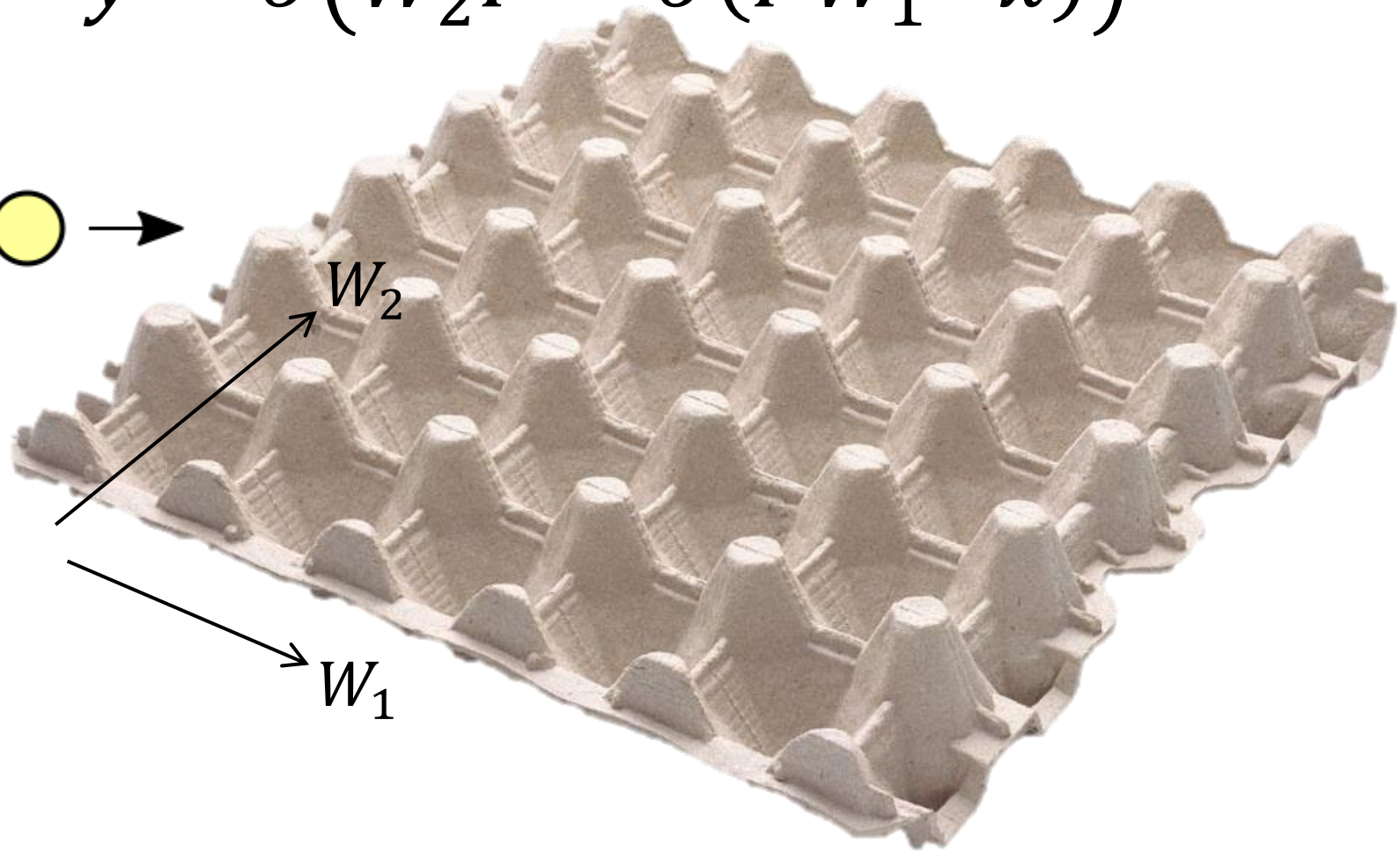
  No!

SGD - Remarks

$$y = \sigma\left(W_2 \cdot \sigma(W_1 \cdot x)\right)$$

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





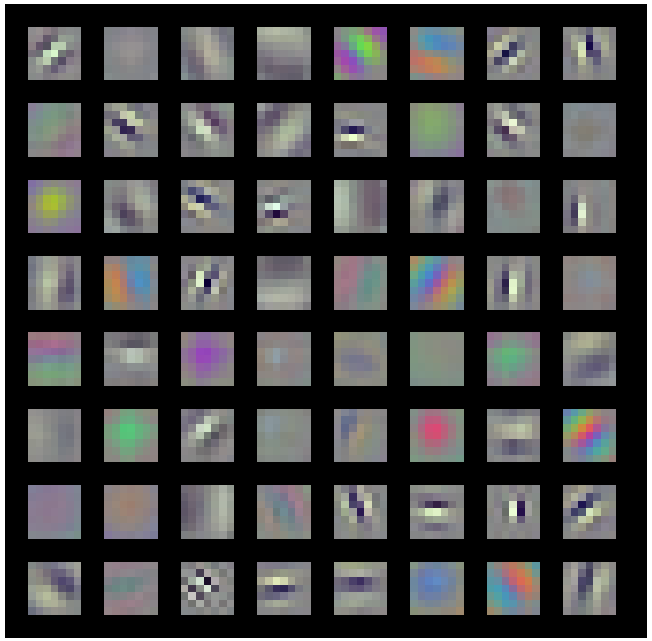
SGD – Starting point

What if we init $\boldsymbol{\theta} = 0$?

$$y = W^T x$$

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

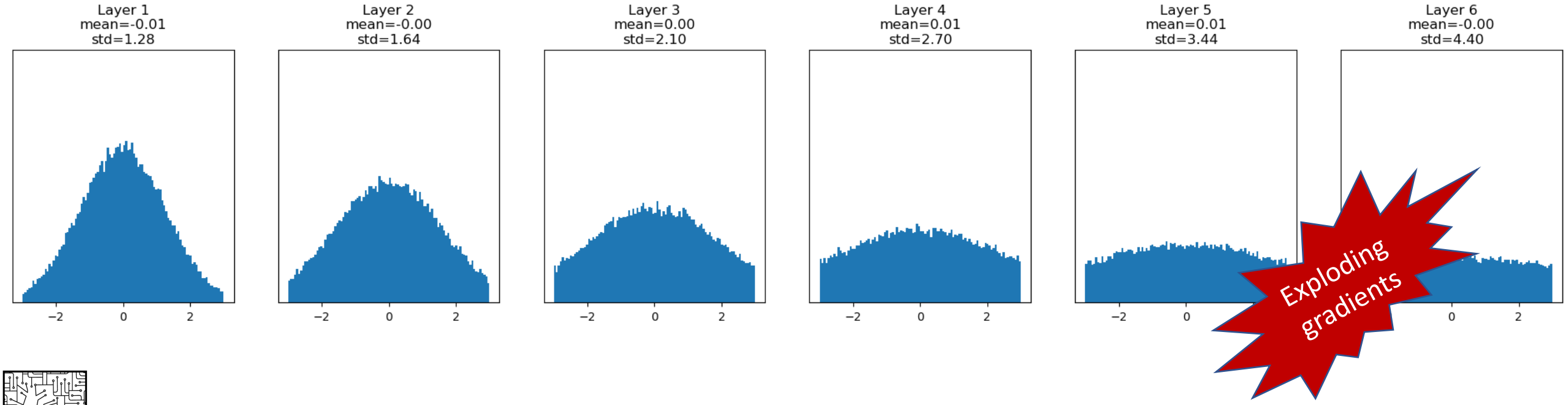
$$\frac{\partial y}{\partial W} = x^T$$



SGD – Starting point

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

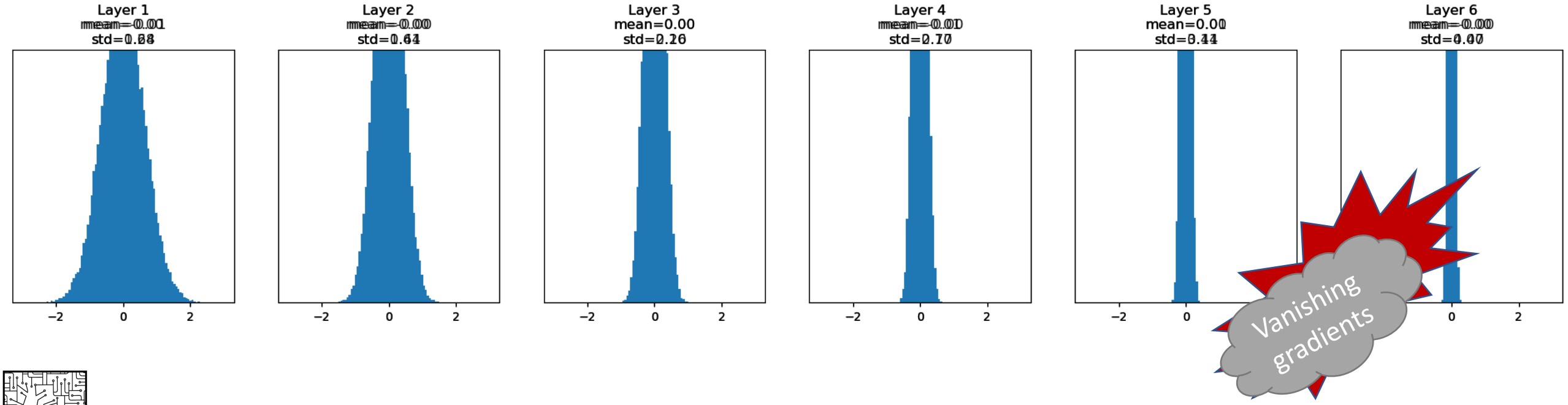
$w_{ij} \sim N(0, 0.02^2)$



SGD – Starting point

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

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



SGD – Starting point

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

$$y = W^T x$$

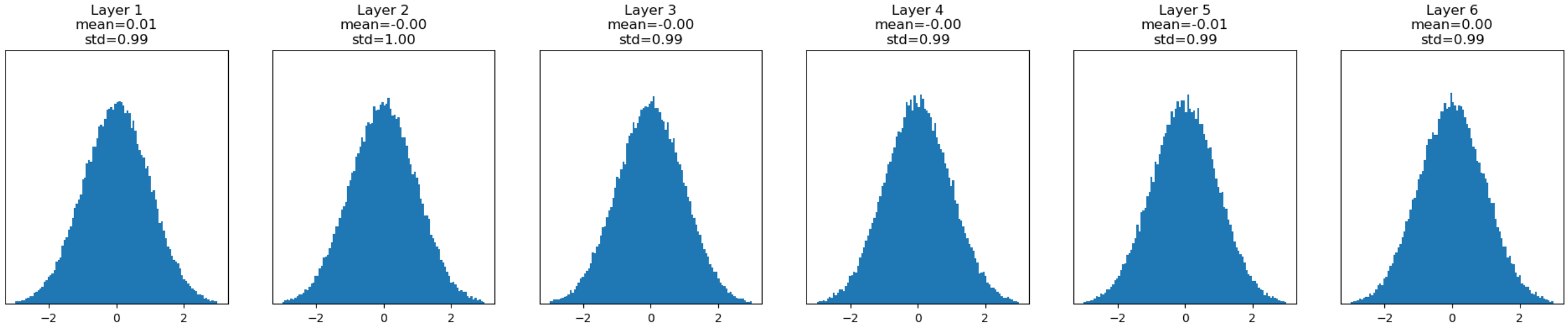
$$\mathrm{var}(y_i) = \mathrm{var}\left(W_i^T x\right)$$

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

SGD – Starting point

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

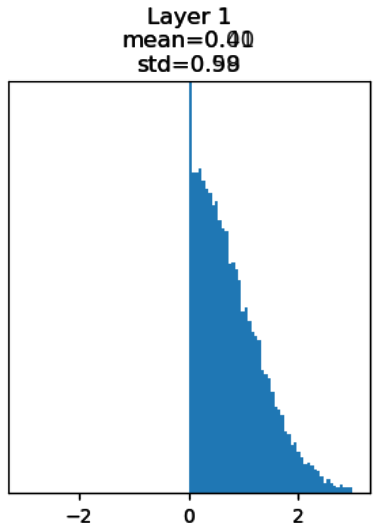
$$w_{ij} \sim N\left(0, {}^{1}/_{D_{in}}\right)$$



SGD – Starting point

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

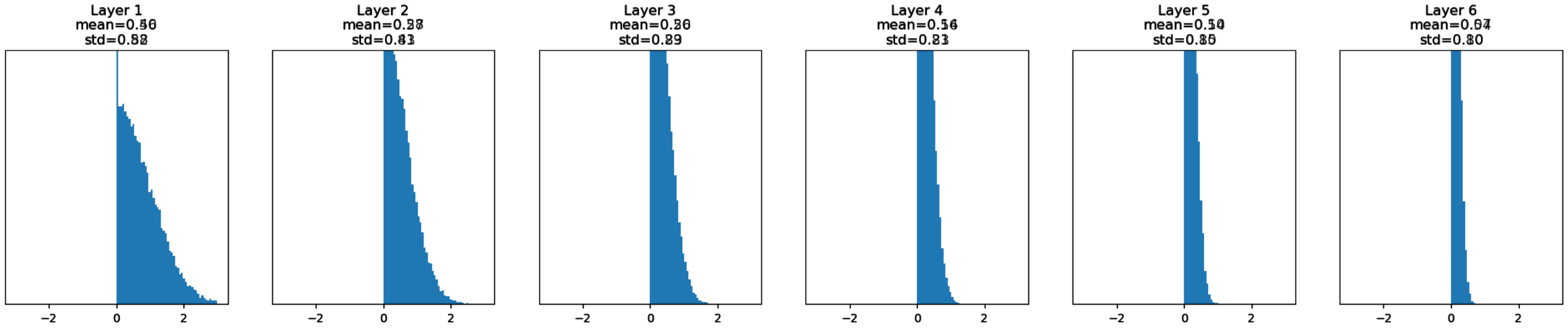
$$w_{ij} \sim N\left(0, {}^{1}/_{D_{in}}\right)$$



SGD – Starting point

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

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



jupyter notebook is available

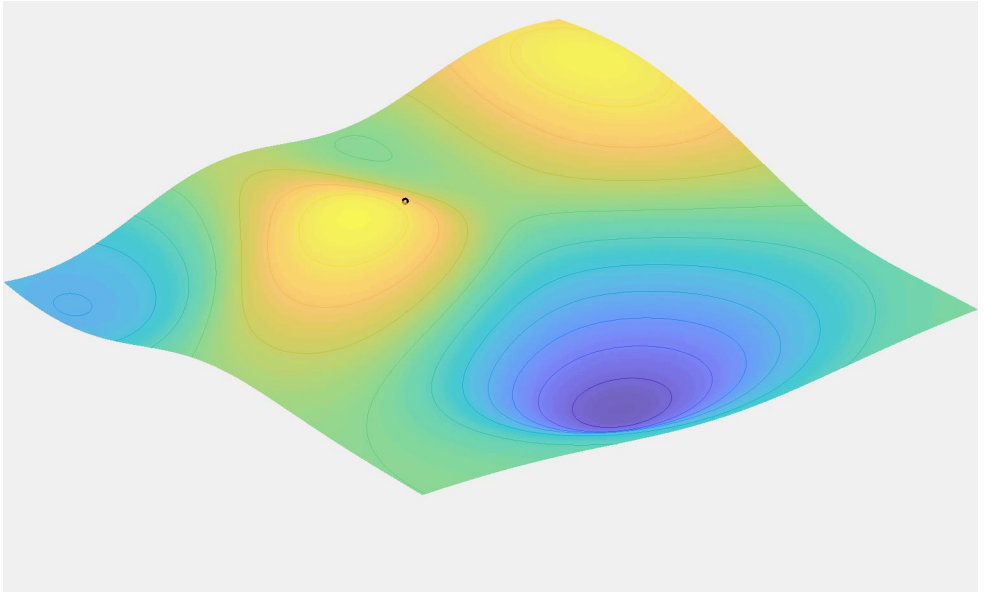
SGD – Starting point

- 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 $\sim N(0,1)$

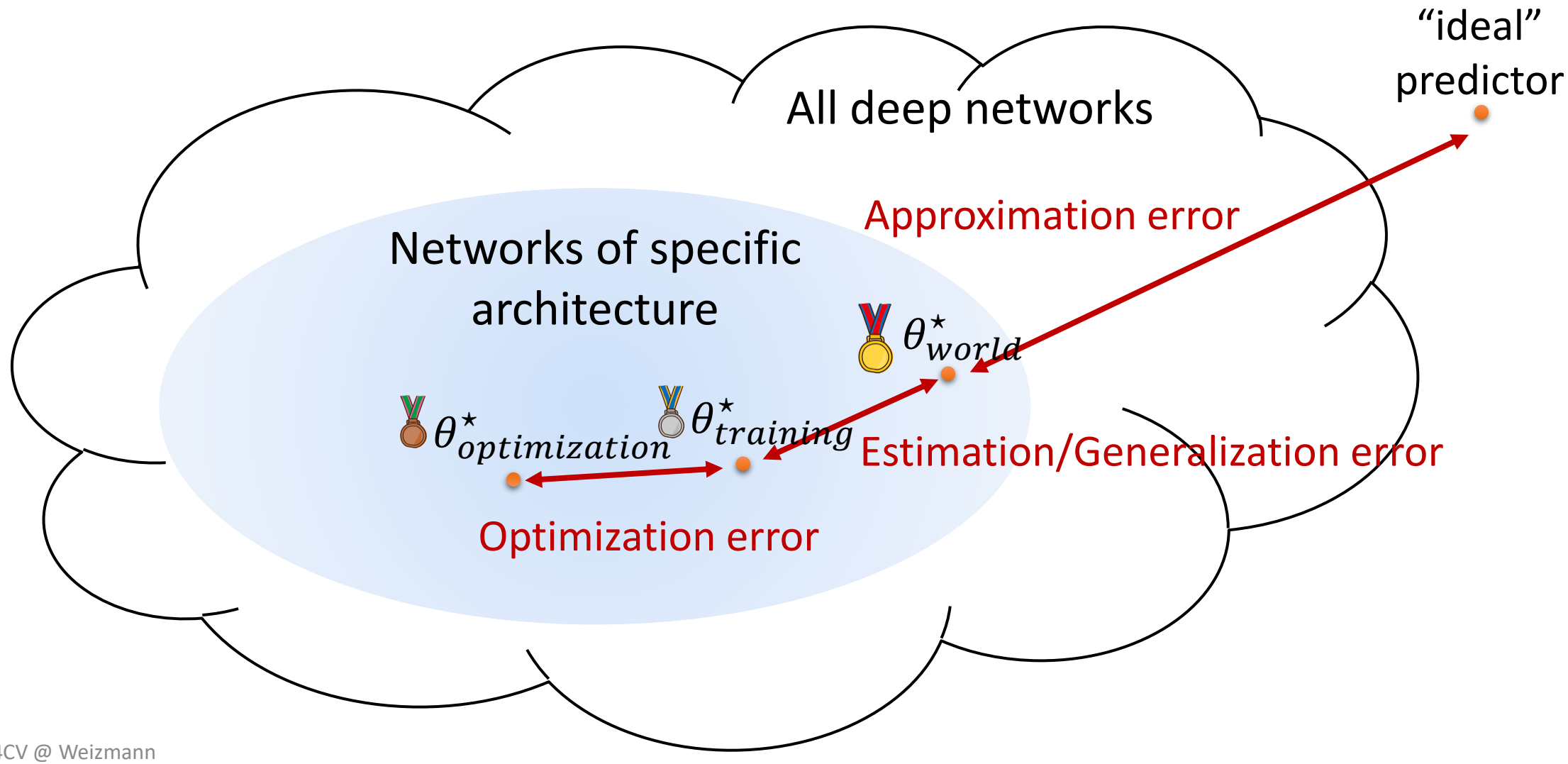
Break

Agenda

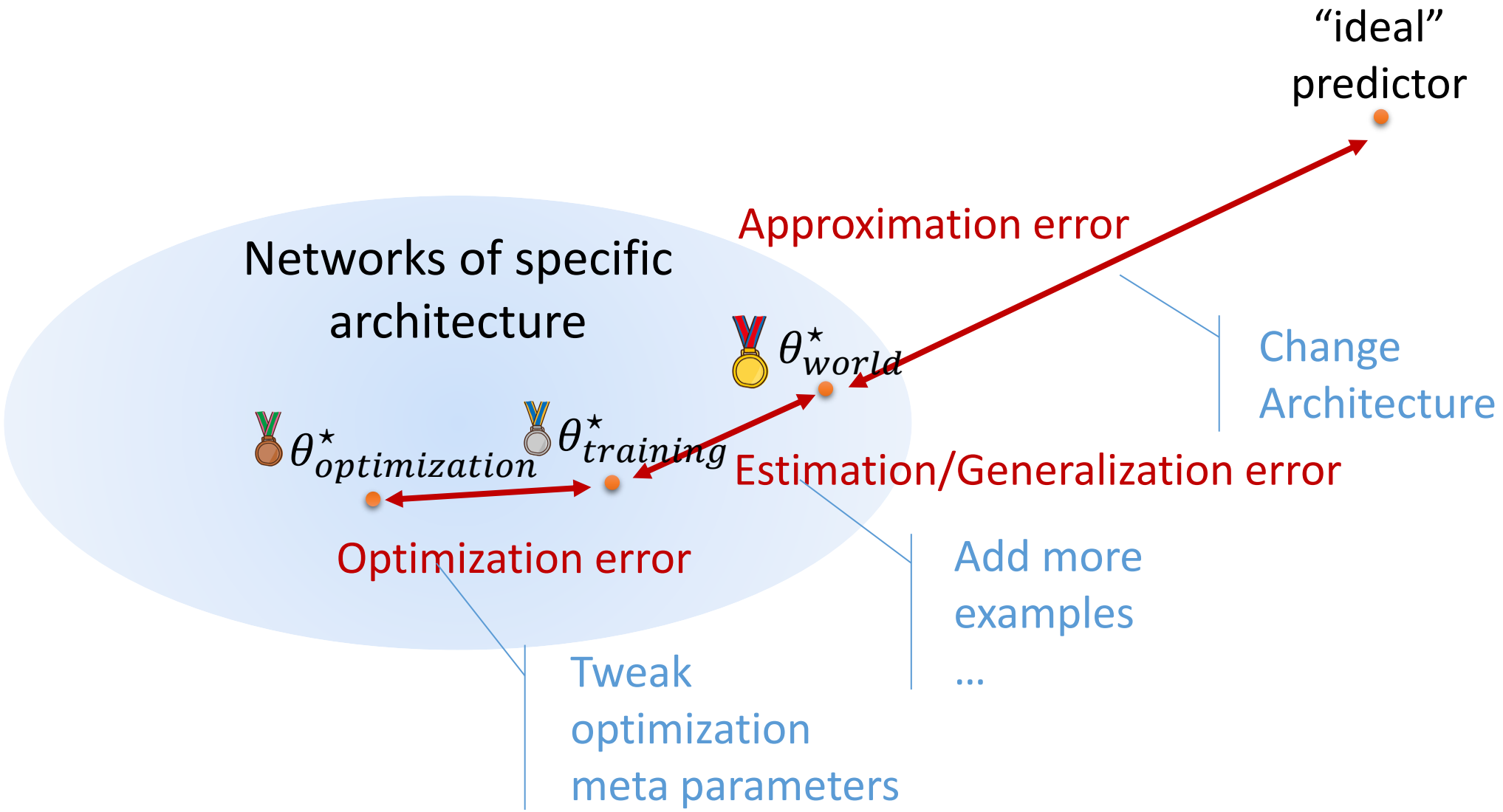
- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



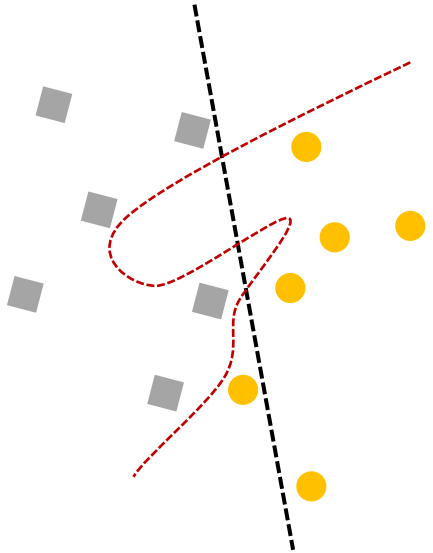
Optimization's Pitfalls



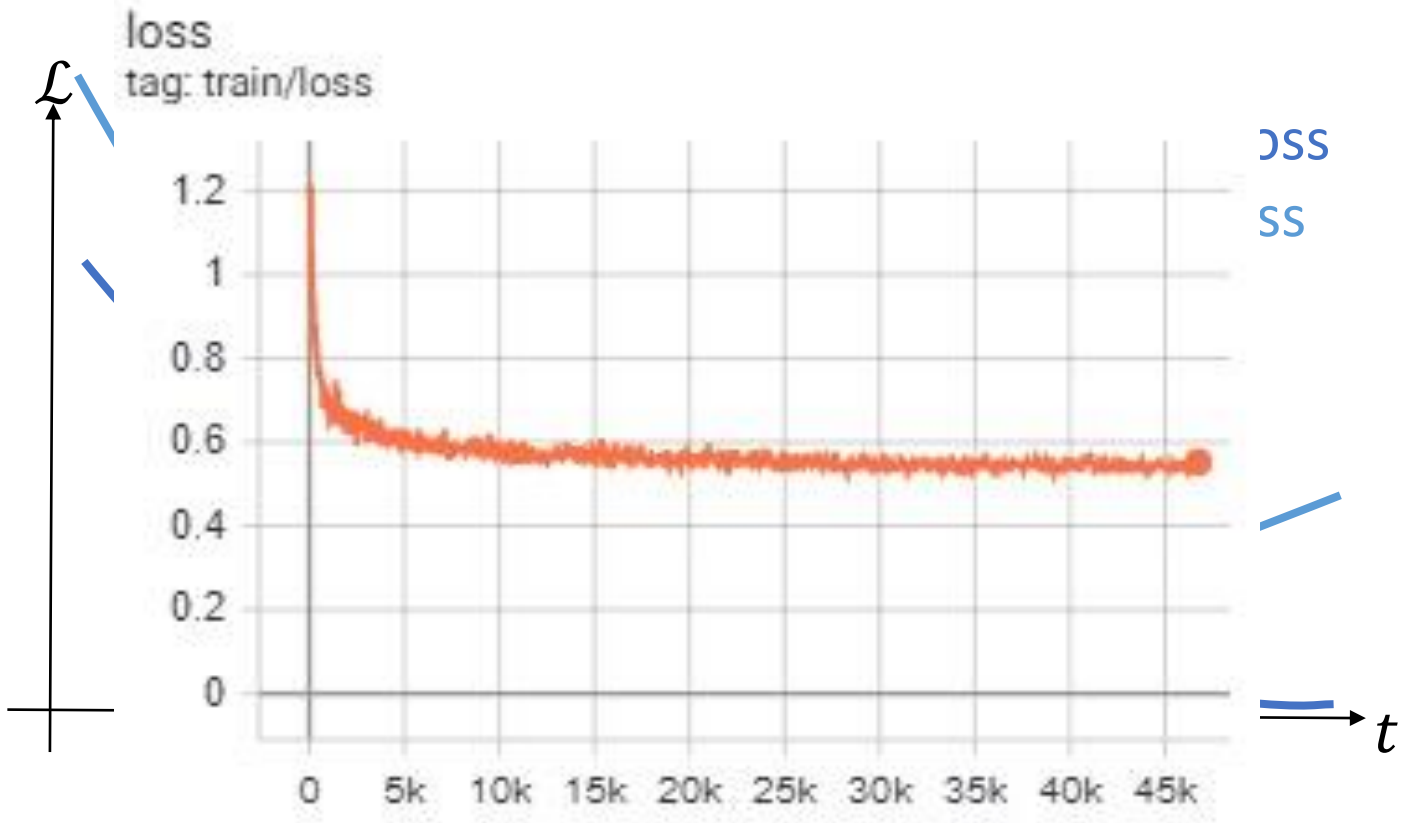
Optimization's Pitfalls



Optimization's Pitfalls



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



Regularization

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

Regularization

- Data augmentation

dog

Torchvision: `transforms`
Albumenations: https://github.com/albumentations-team/albumentations

Regularization

Dropout

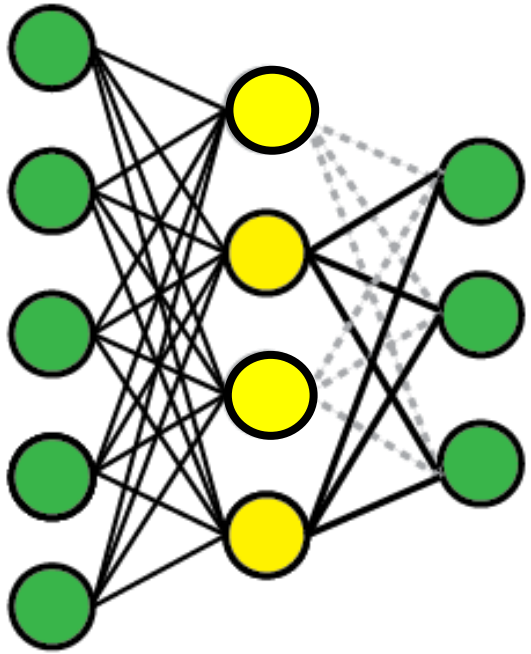
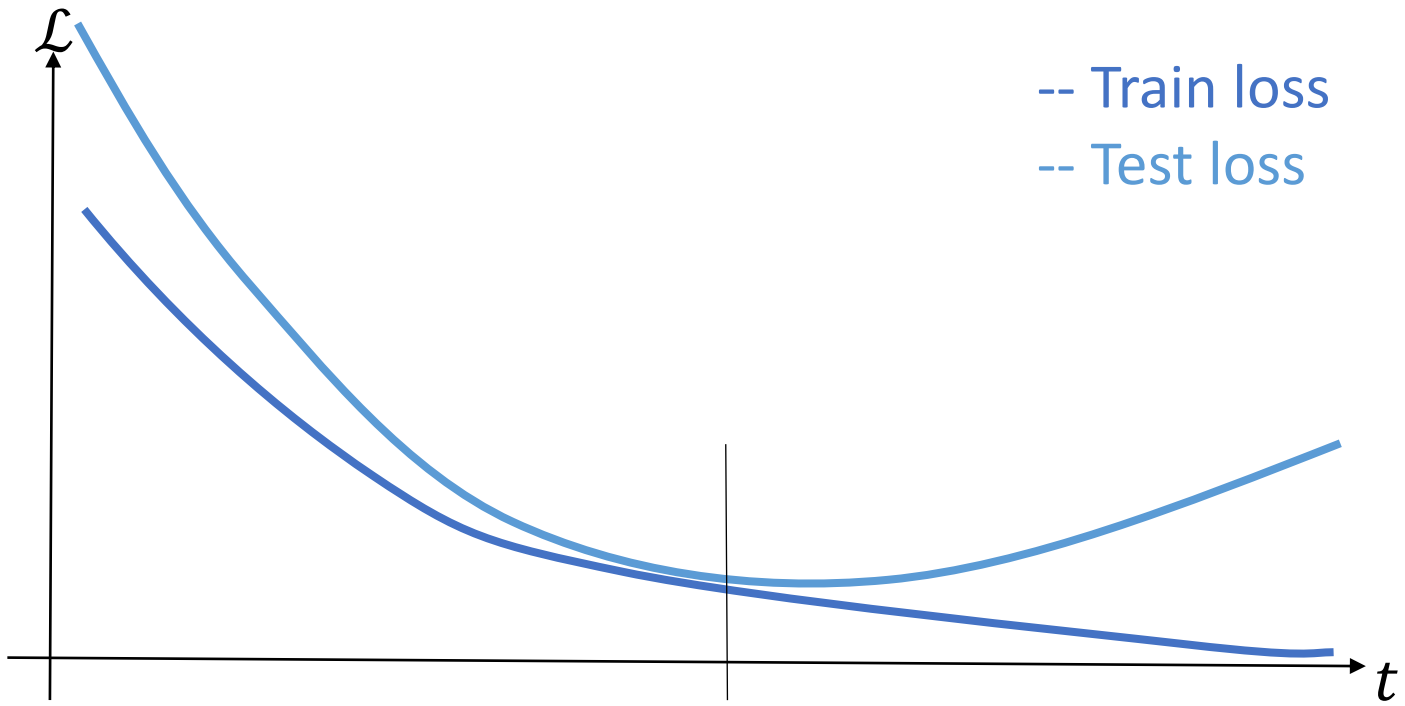


Image source

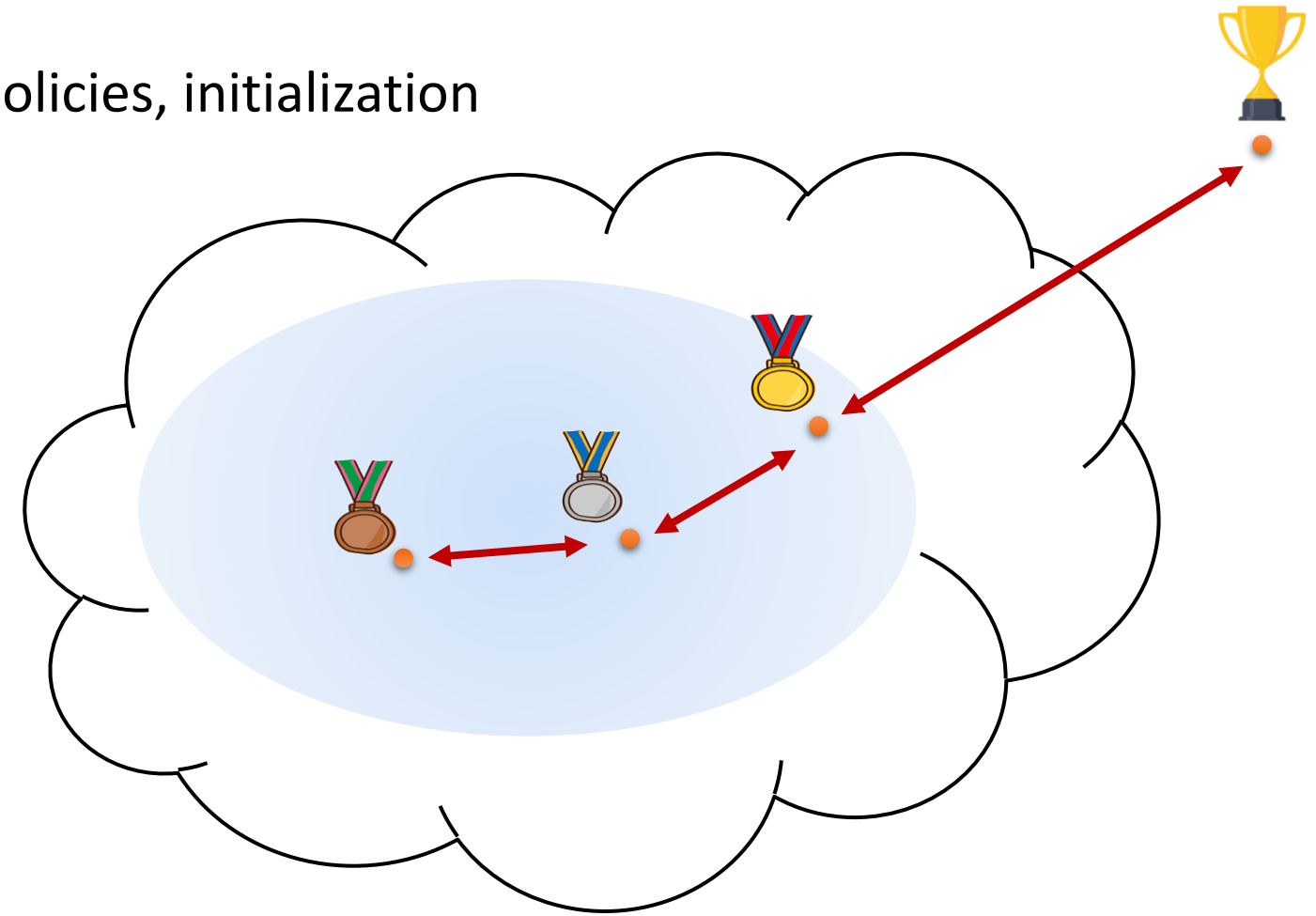
Regularization

Early Stopping



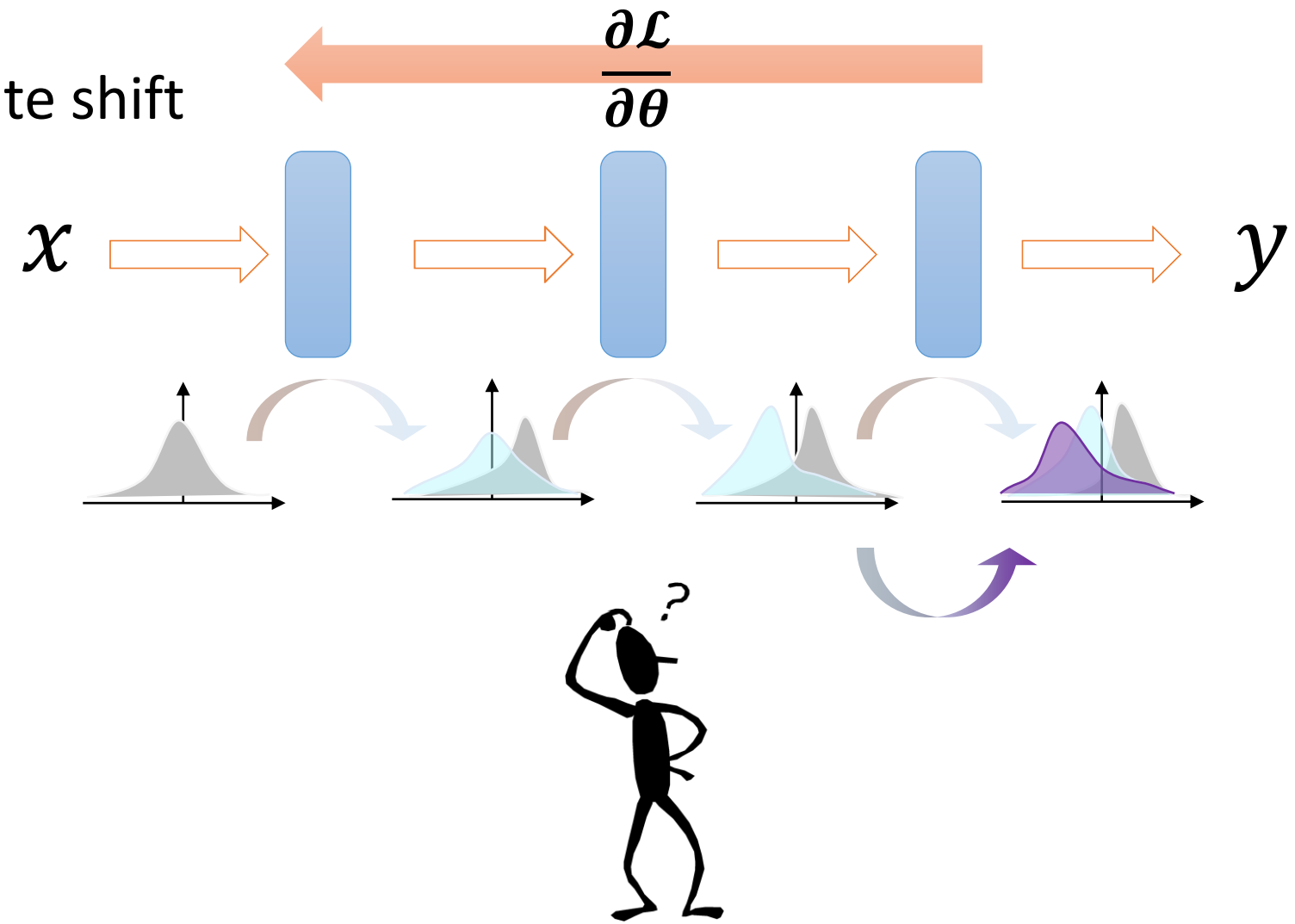
Agenda

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



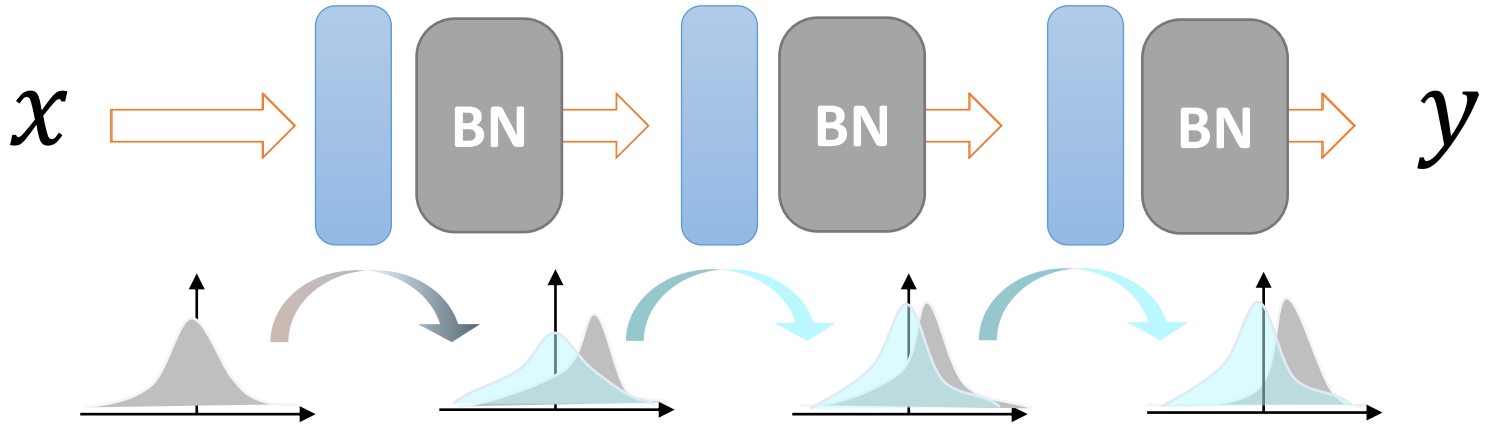
Normalization Layers

- Covariate shift



Normalization Layers

- Batch Norm!



Normalization Layers

- Batch Norm

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

  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: $\boldsymbol{y} = \frac{\boldsymbol{x}-\mu}{\sqrt{\sigma^2+\varepsilon}}\gamma + \beta$

Normalization Layers

- Batch Norm

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

Test time?

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: $\boldsymbol{y} = \frac{\boldsymbol{x}-\mu}{\sqrt{\sigma^2+\varepsilon}}\gamma + \beta$

Normalization Layers

- Batch Norm

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

  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: $\boldsymbol{y} = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}\gamma + \beta$

Training time:

Updated in **forward pass**

$= \frac{1}{N}\sum_b x_{bc}$

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

Updated in **backward pass**

$99\hat{\mu} + 0.01\mu$

Normalization Layers

- Batch Norm

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

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

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

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

Test time:

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

Normalization Layers

- Batch Norm

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

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

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

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

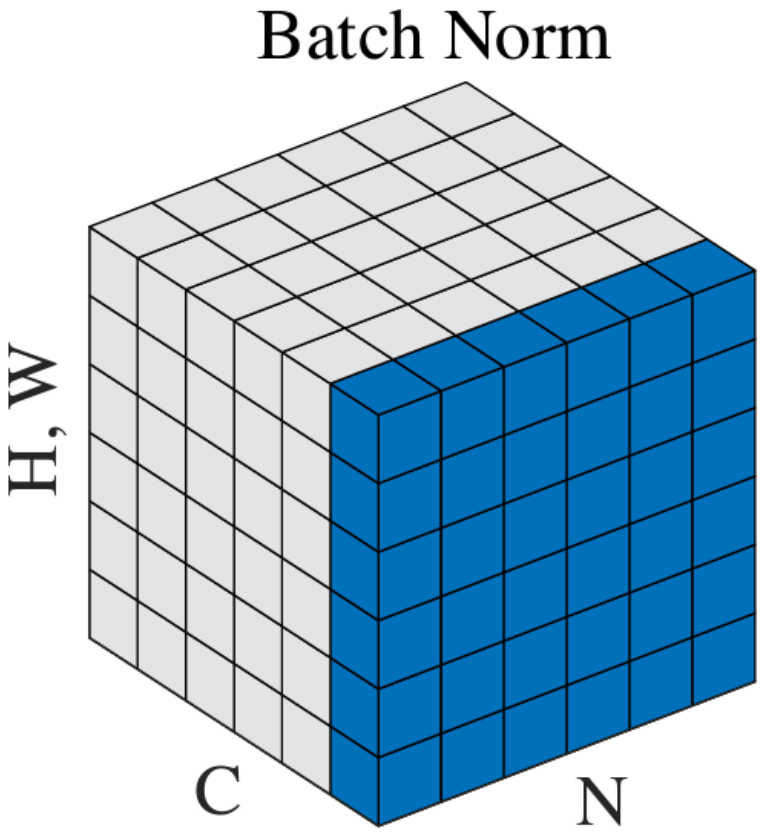
For Conv layers

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

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

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

Normalization Layers



For Conv layers

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

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

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

Yuxin Wu, Kaiming He, **Group normalization**. (ECCV 2018)

Normalization Layers

Batch Norm

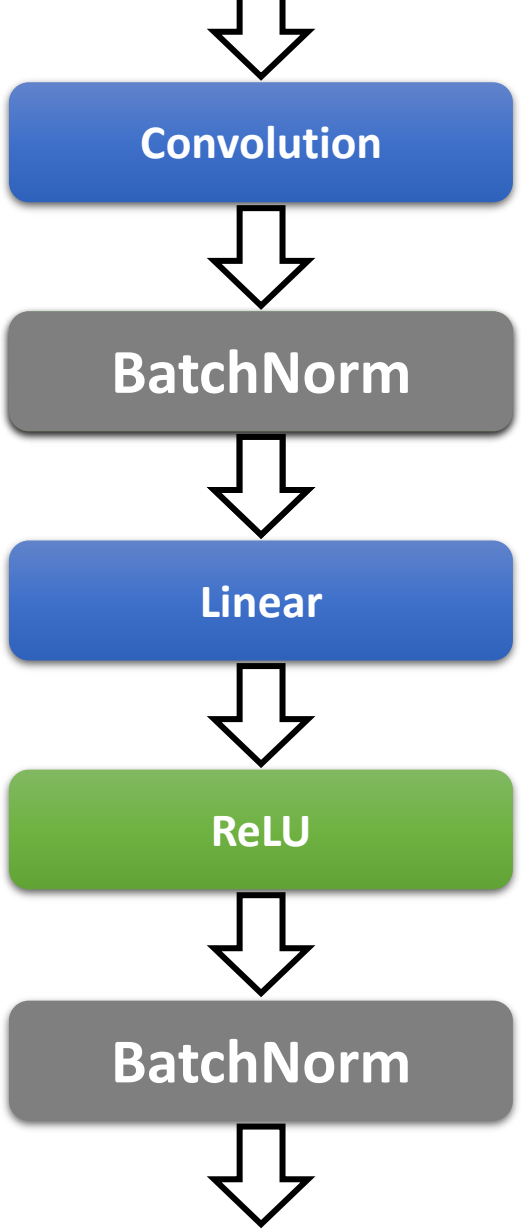
+ Training more robust to init

+ Allows for training deeper nets

+ Allows for higher learning rates

H, W

- Different train/test behavior



Normalization Layers

LayerNorm

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

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

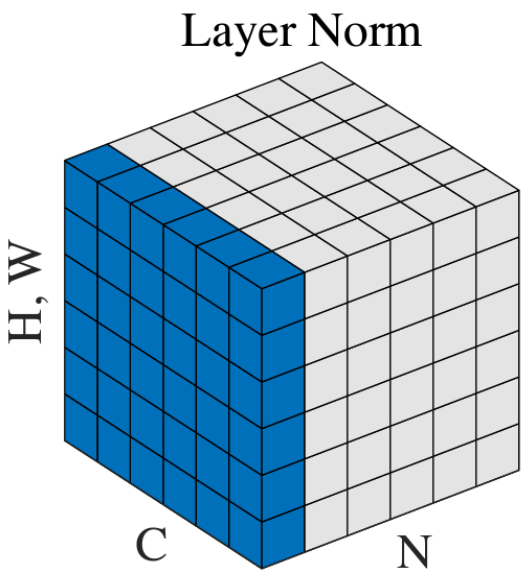
$\gamma, \beta: 1 \times C \times 1 \times 1$

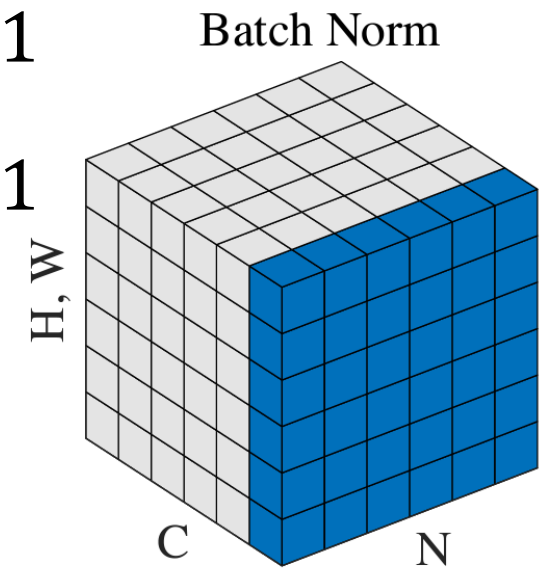
BatchNorm

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

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

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





Yuxin Wu, Kaiming He, **Group normalization**. (ECCV 2018)

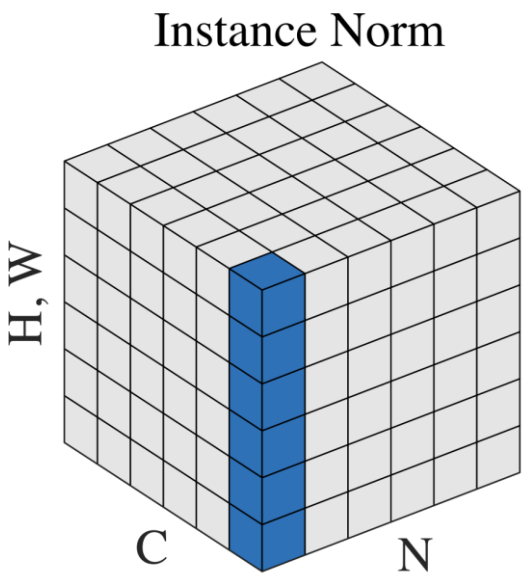
Normalization Layers

InstanceNorm

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

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

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

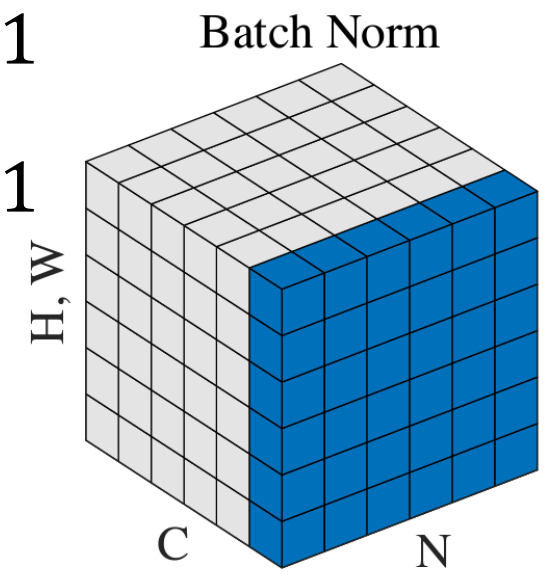


BatchNorm

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

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

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



Yuxin Wu, Kaiming He, **Group normalization**. (ECCV 2018)

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