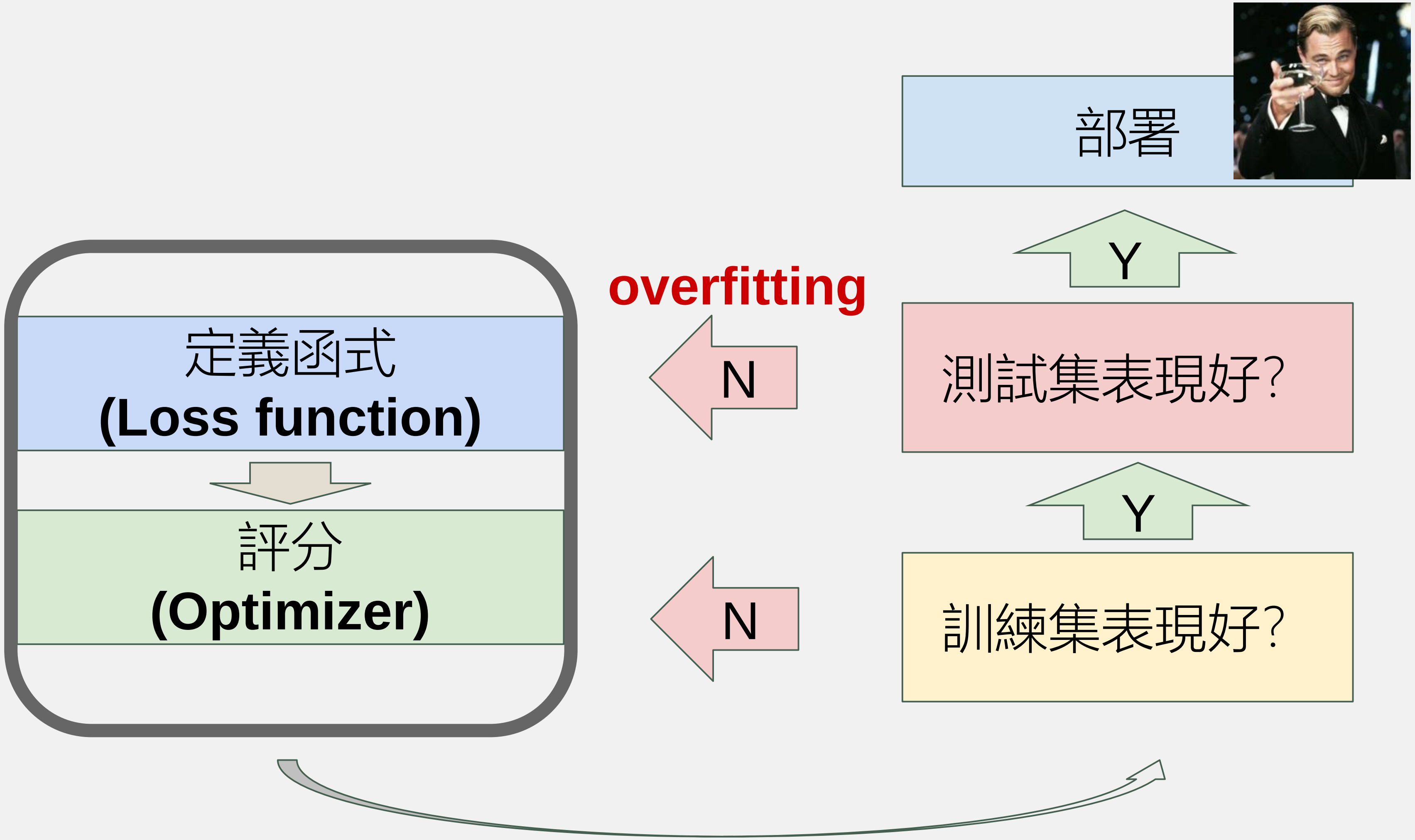
Deep Learning Training Tips

Deep Learning Training Tips

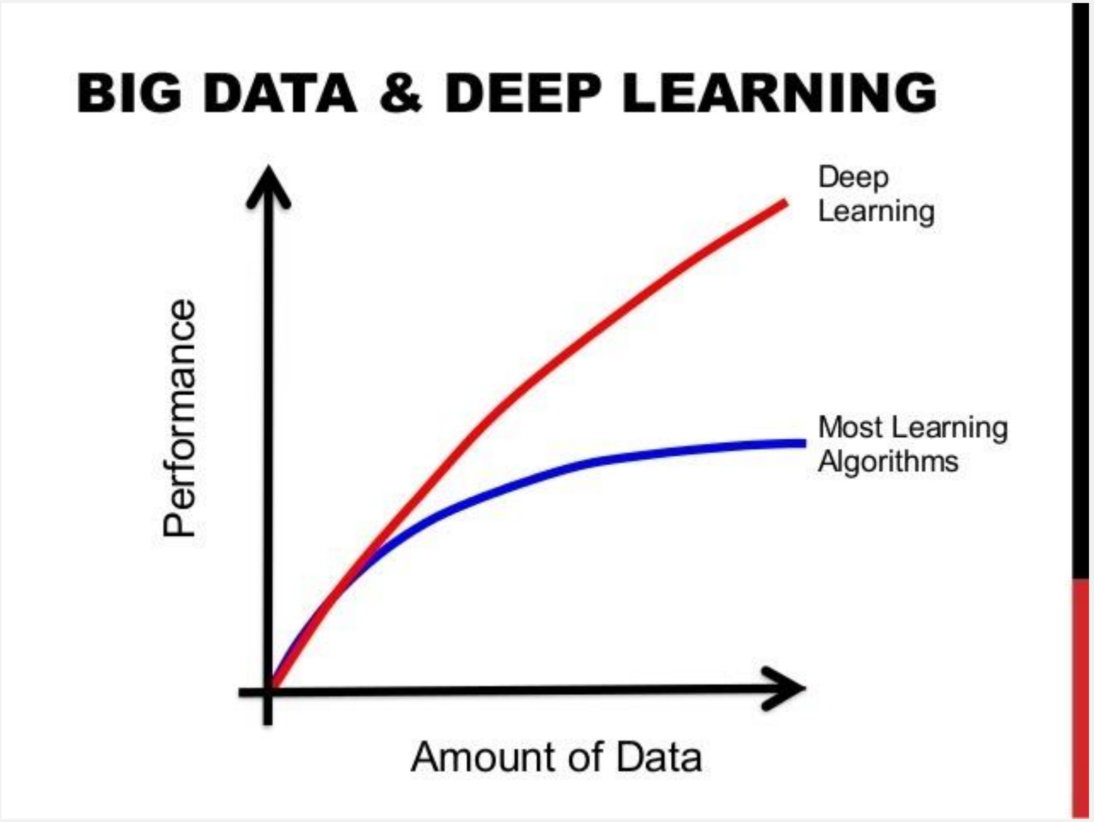
Training Process



Deep Learning Training Tips
Training Process

DL is very strong but…

- It sometimes can't achieve the good performance
- The training time might be long
- etc.



[Reference](#)

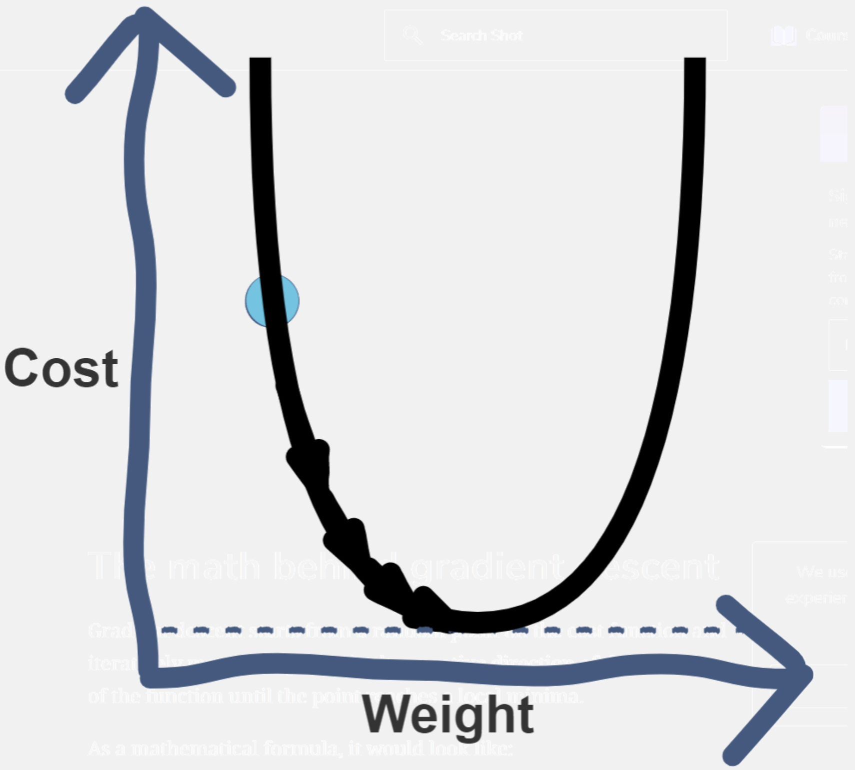
01

Where does the error come?

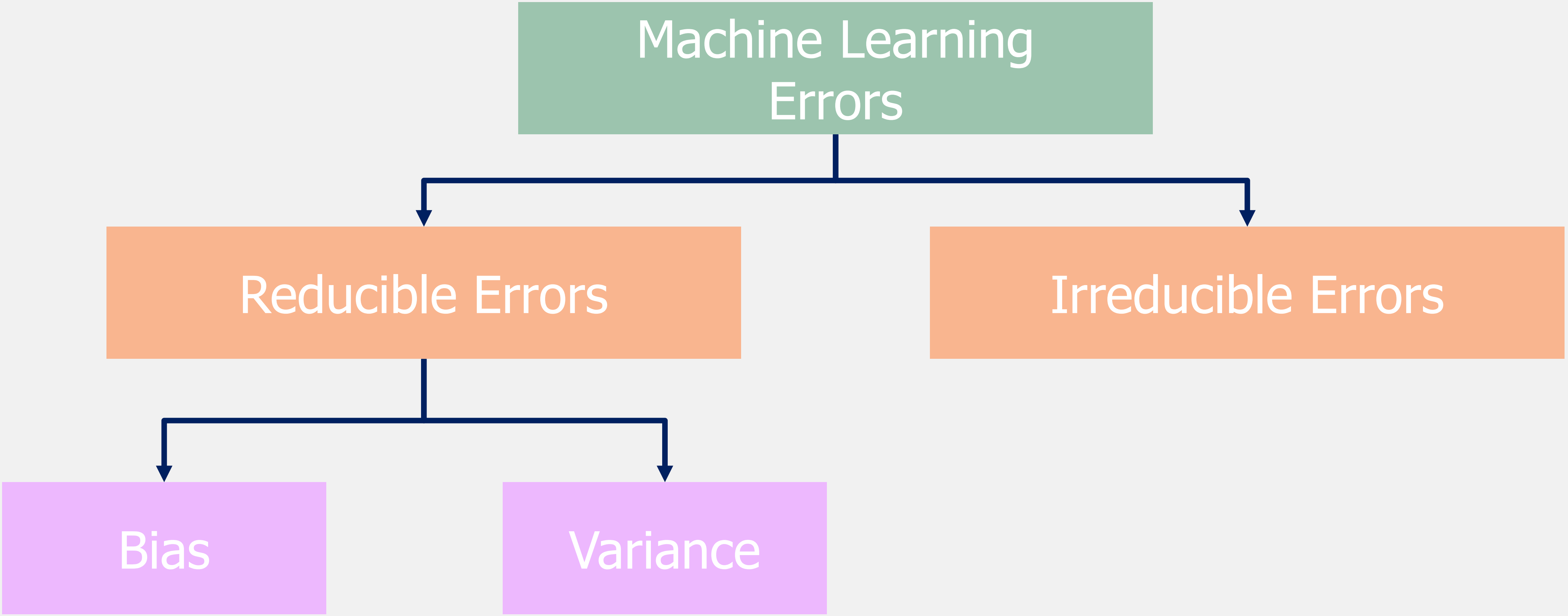
Recap

The success of a machine learning model relies on minimizing the error between the actual and the predicted results. This error/loss is evaluated using special functions called **loss functions**

Gradient descent is a powerful optimization technique used to minimize loss functions. It leverages mathematical concepts to update weights, step by step, in order to gradually reduce model loss and move it towards a local minima

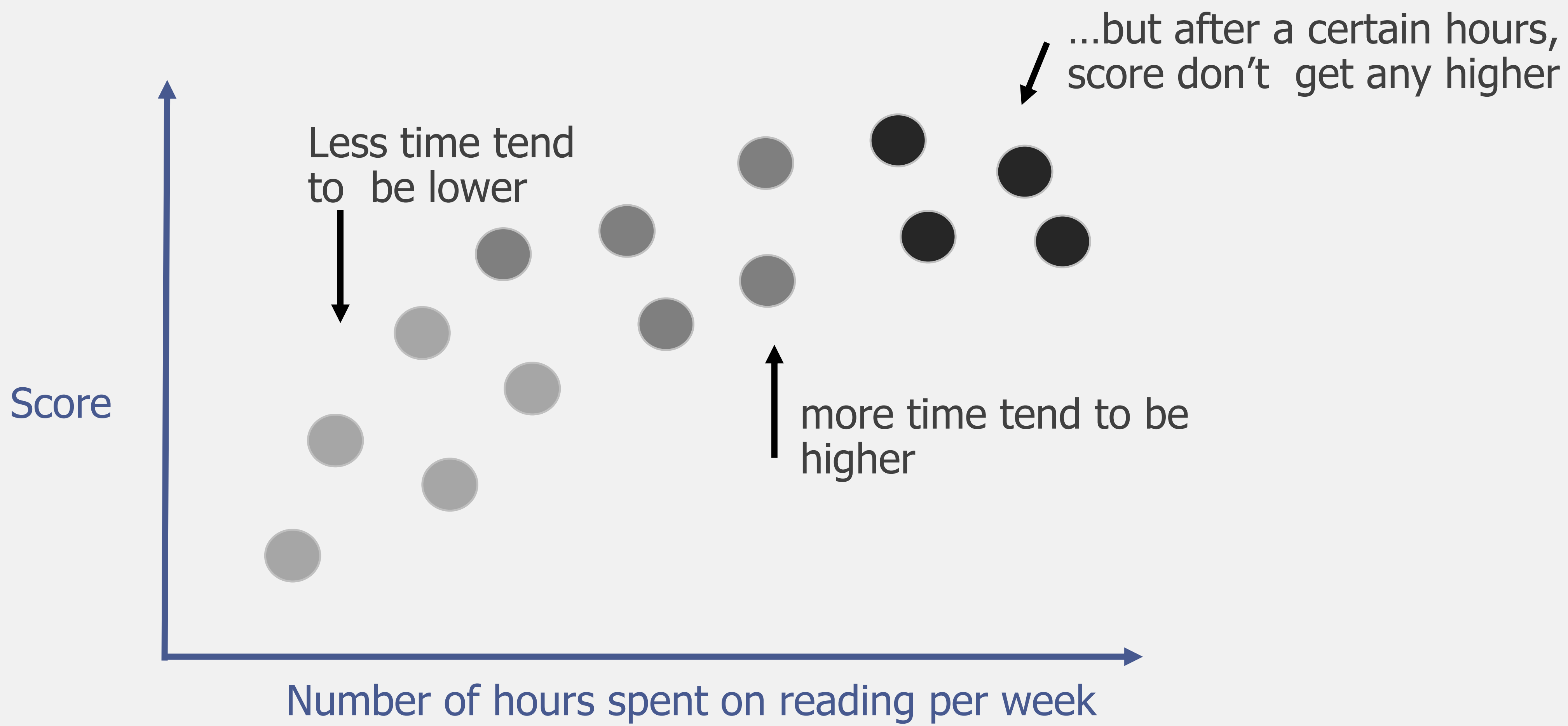


Where does the error come?



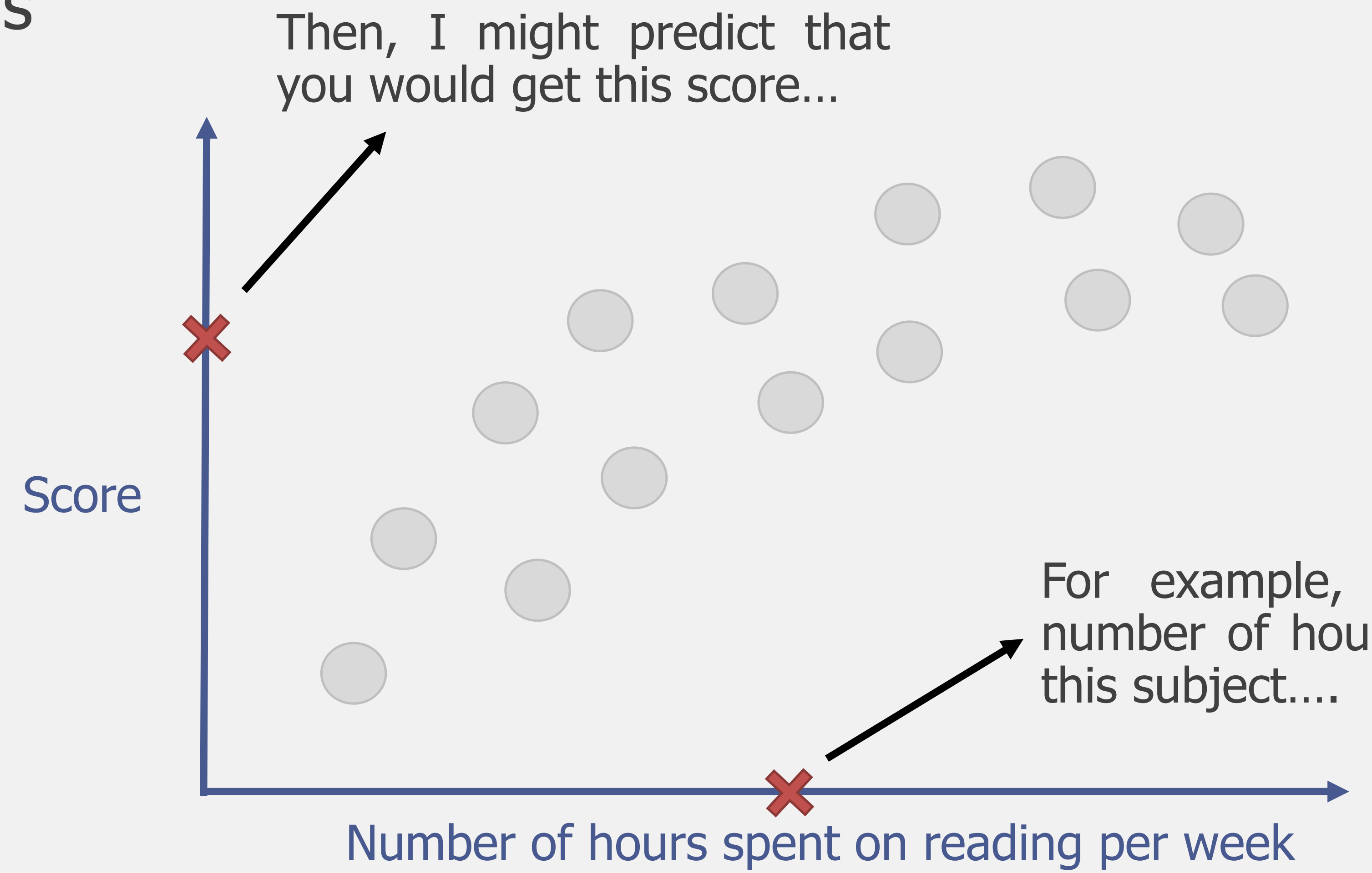
Reference

Where does the error come?



Where does the error come?

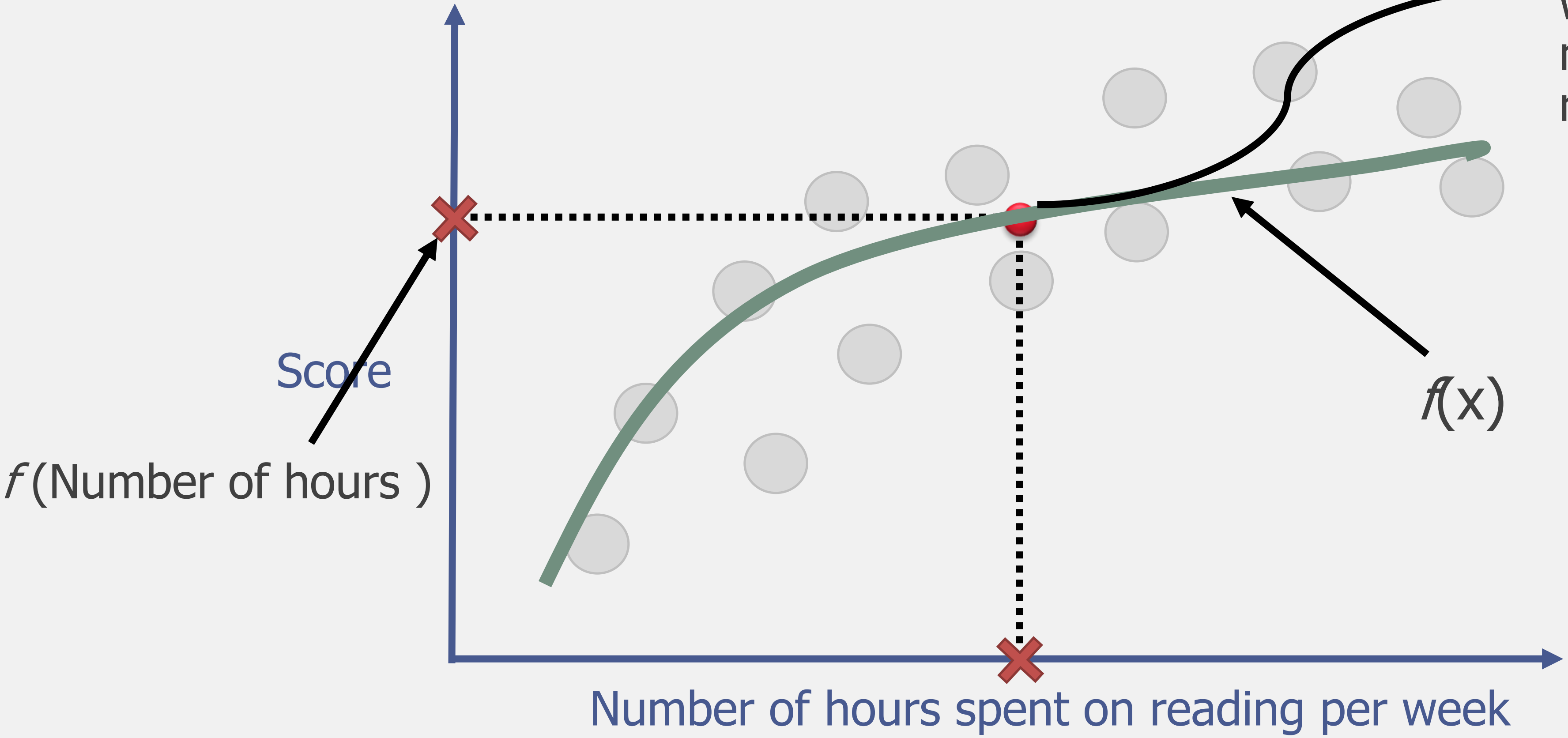
Given this data we would like to predict score when given the number of hours



Where does the error come?

Ideally, we would know the exact mathematical formula that describes the relation between number of hours and score

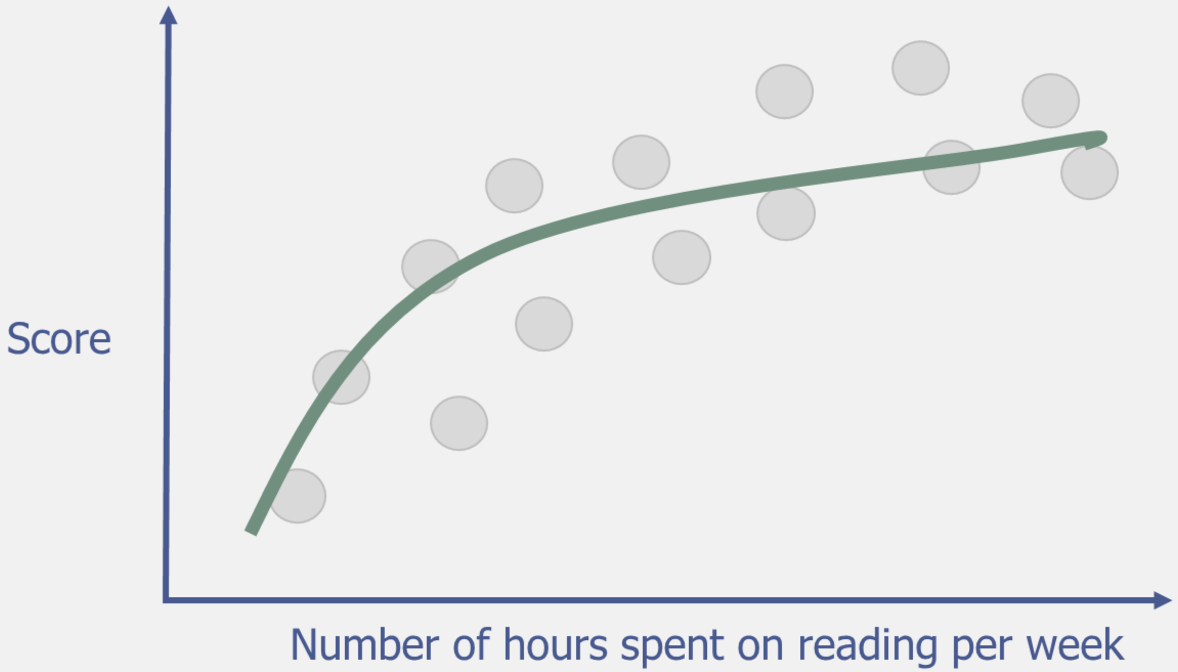
…but, in this case, we don't know the function: f(x), so we're going to use two ML methods to approximate this relationship

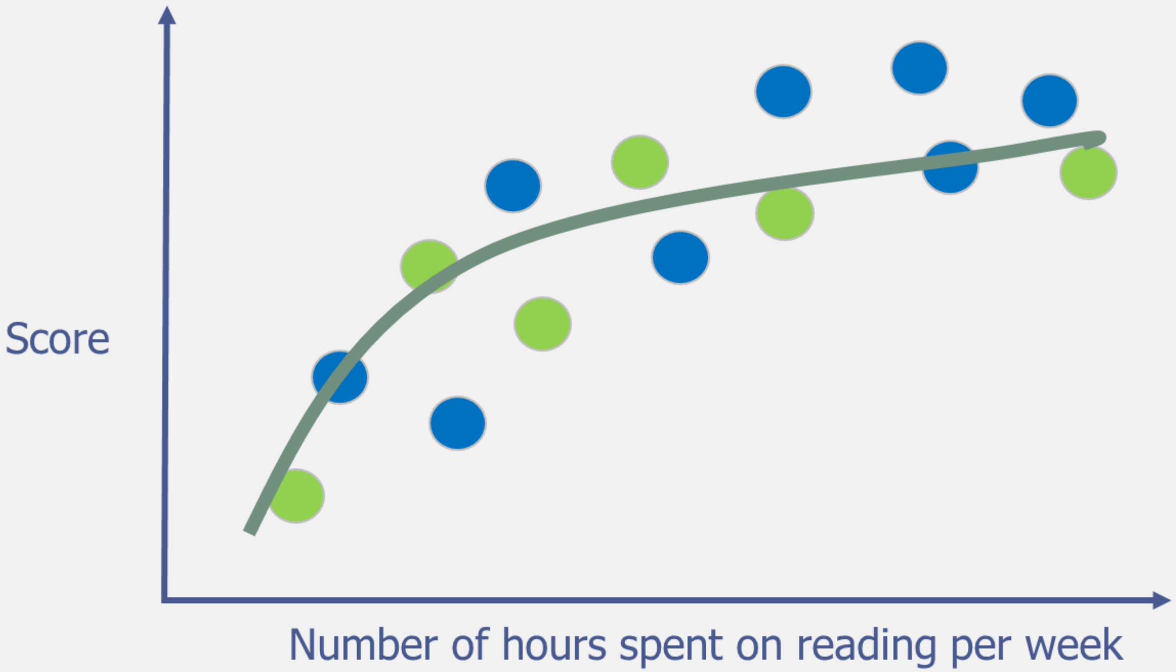


Where does the error come?

Assume the green curve is the “true” relationship for reference

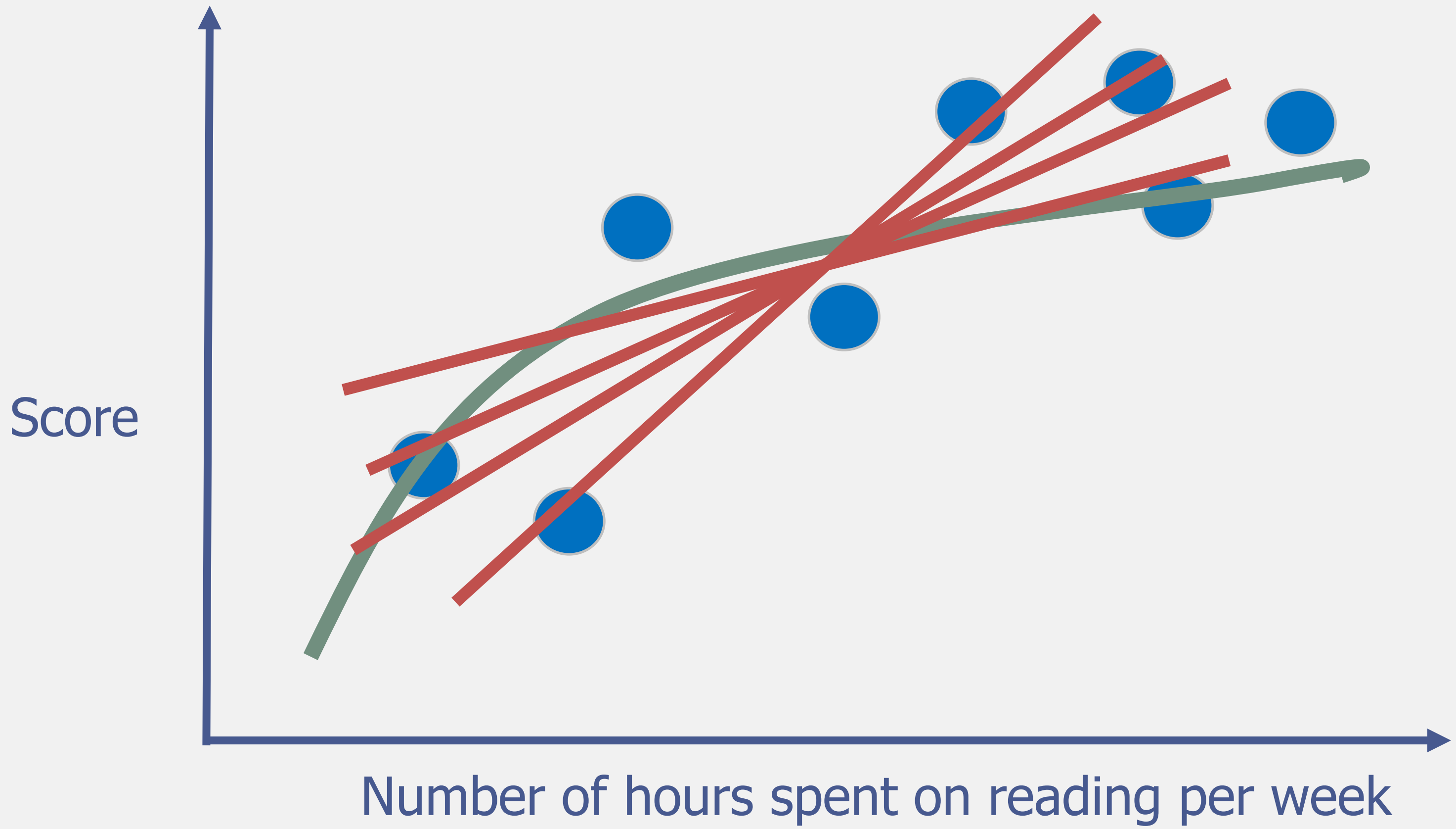
The first step, split the data into two sets, one for training the machine learning algorithms and one for testing them





Where does the error come?

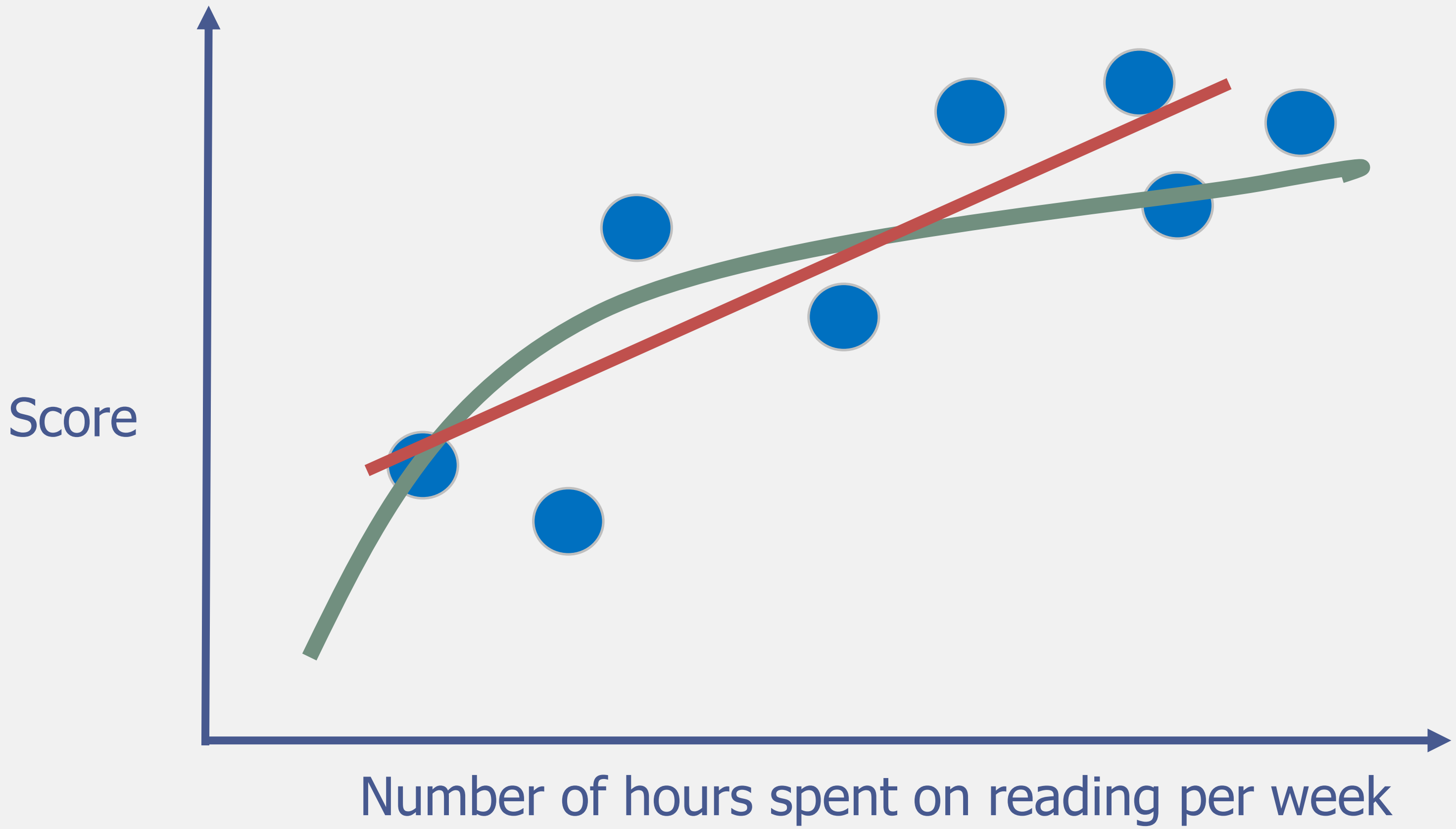
The first algorithm that we will use is Linear Regression



Thus, the **Straight Line** will never reach the true relationship between number of hours and score, no matter how we fit it to the training set or we training how many iterations

Where does the error come?

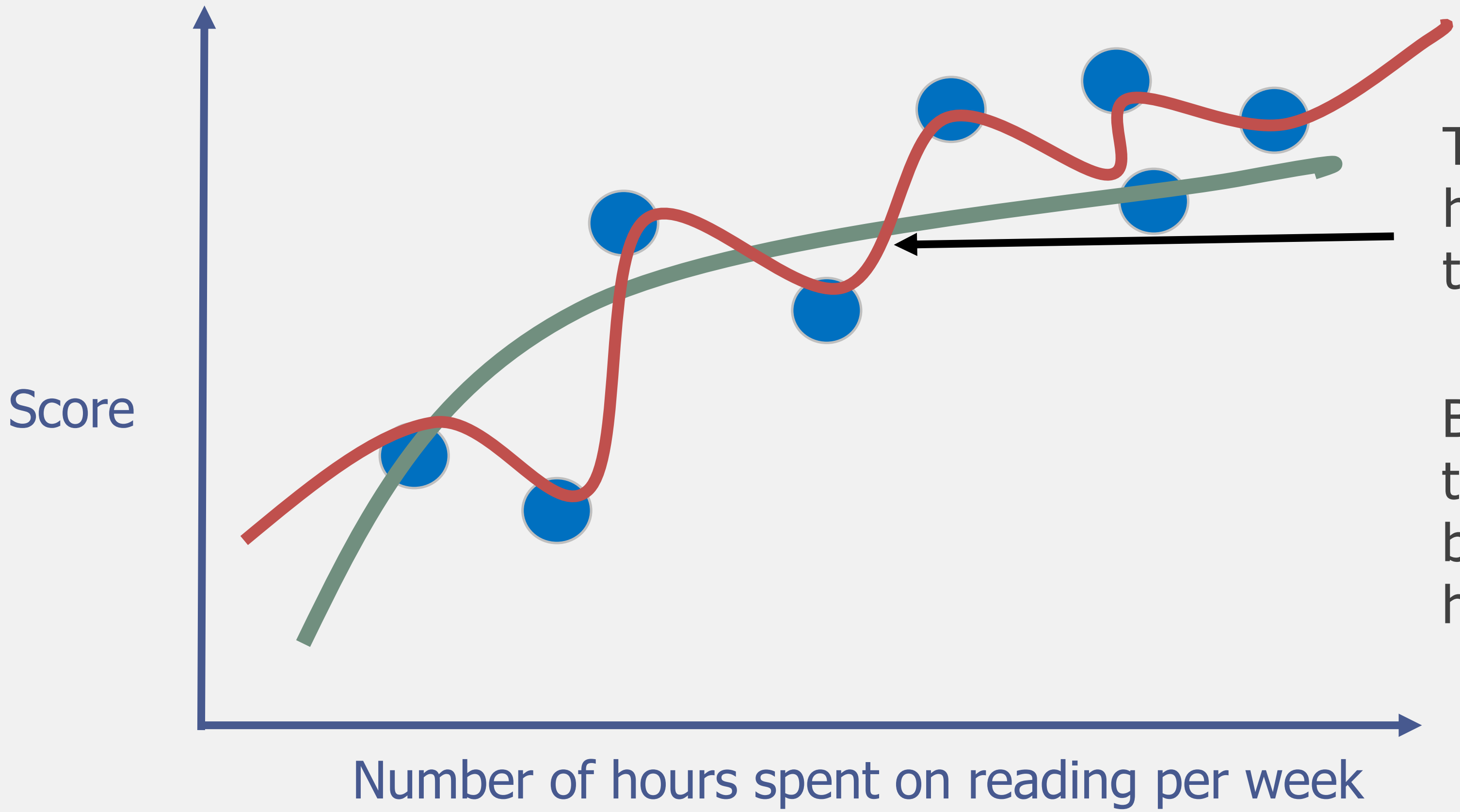
The inability for a machine learning method (like linear regression) to capture the true relationship is called bias



Because the Straight Line couldn't curved like the "true" relationship, it has a relatively large amount of bias

Where does the error come?

Another machine learning algorithm might fit a **Squiggly Line** to the **training set**



The **Squiggly Line** is super flexible and hugs the **training set** along the arc of the “true” relationship

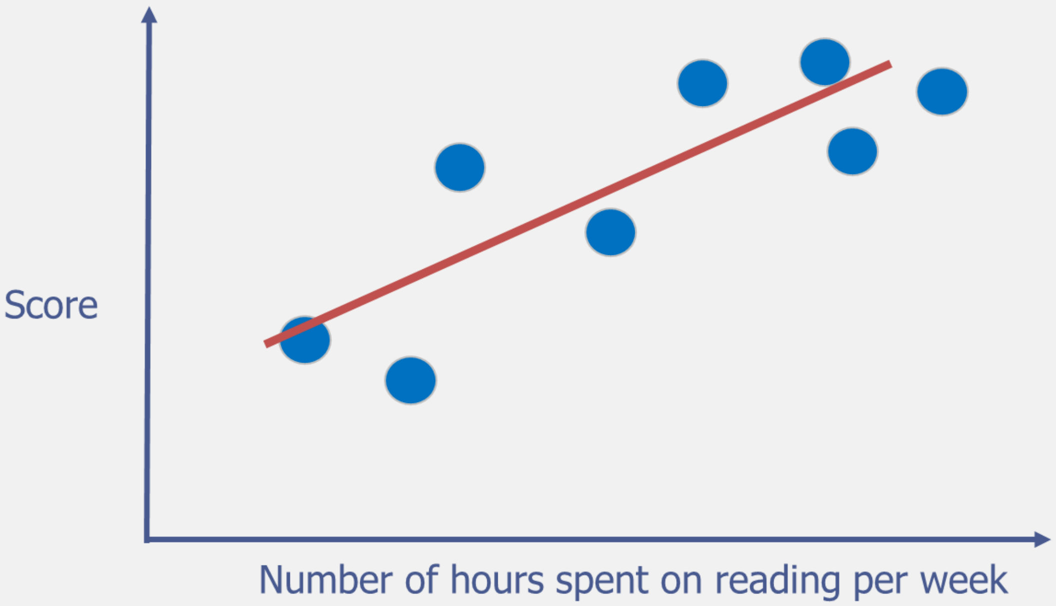
Because the Squiggly Line could handle the arc in the ”true” relationship between number of hours and score, it has vary little bias

Where does the error come?

If we measure the distances from the fit lines to the data, square them and add them up (like mean square error)

In the compare as below, we could see whether the straight line fits the training set better than the squiggly line ……?

The Squiggly Line wins



Where does the error come?

We also have testing set. Now let's calculate the Sums of squares for the testing set

In the contest to see whether the straight Line fits the testing set better than the squiggly Line ......?

The Straight Line wins

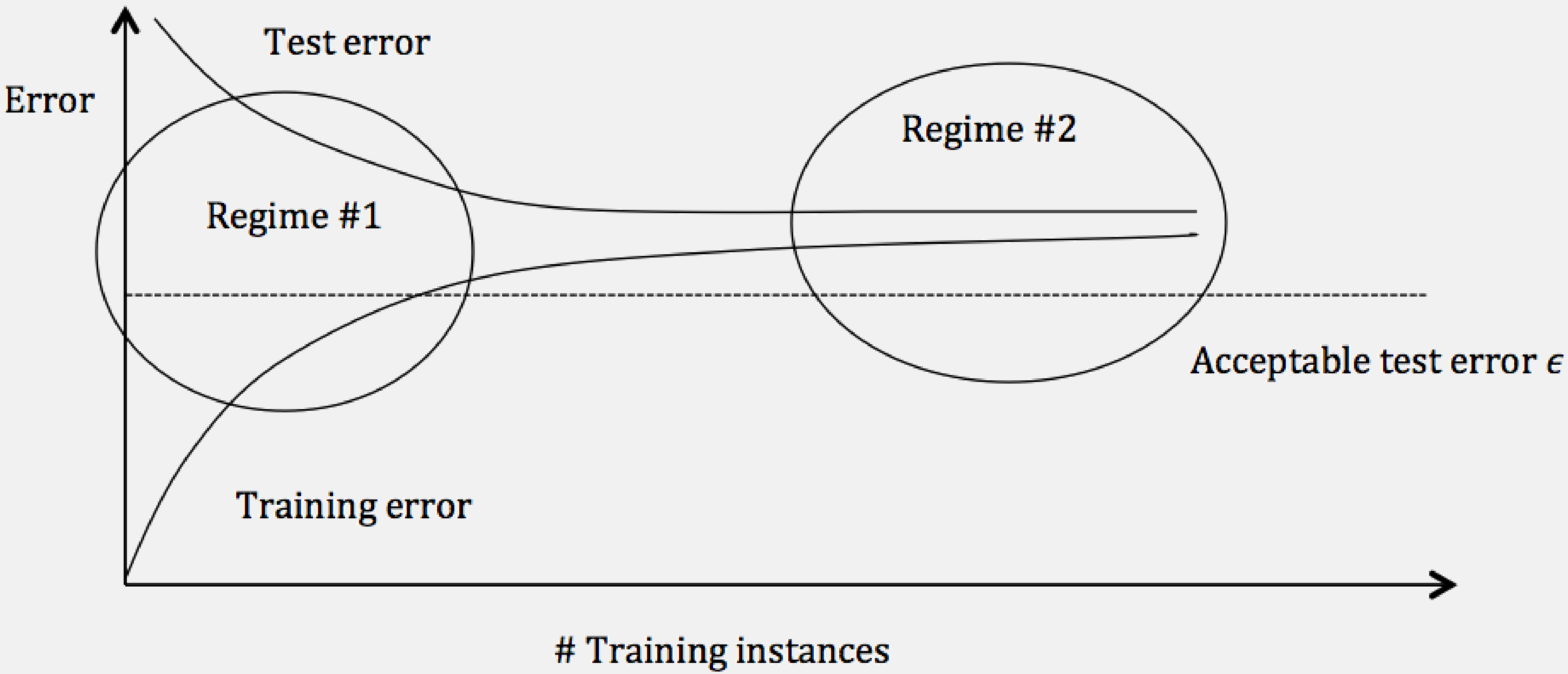
Where does the error come?

Even though Squiggly Line did a great job fitting the training set, it did a terrible job on fitting testing set

In Machine Learning domain, the huge difference error results in fits between training and testing data sets is called Variance

Where does the error come?

High Bias and High Variance



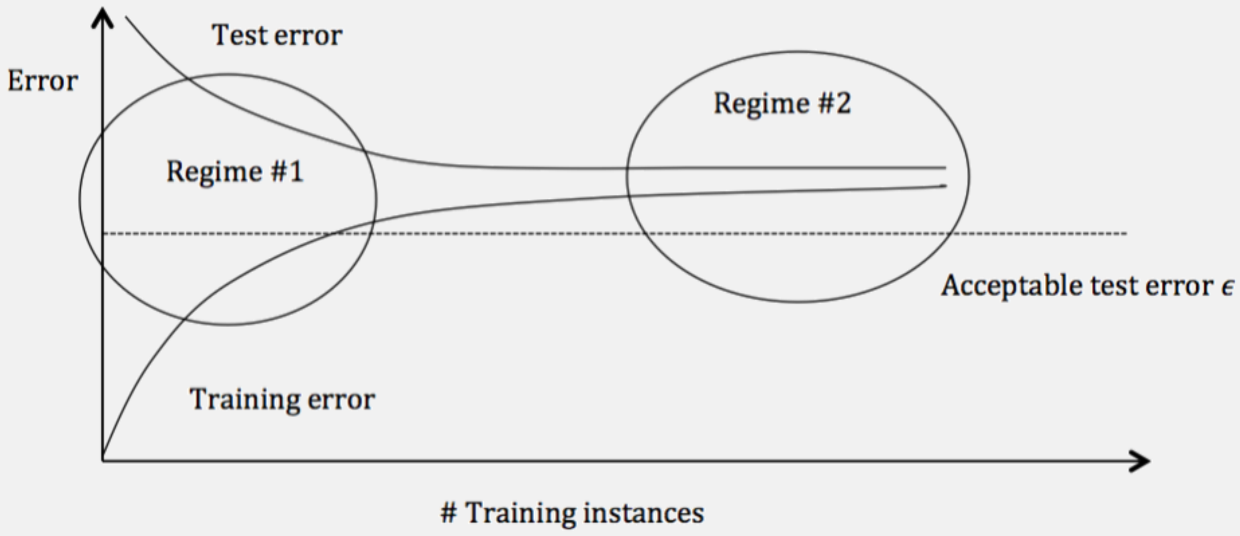
[Reference]

Where does the error come?

High Bias and High Variance

Regime 1 (High variance)

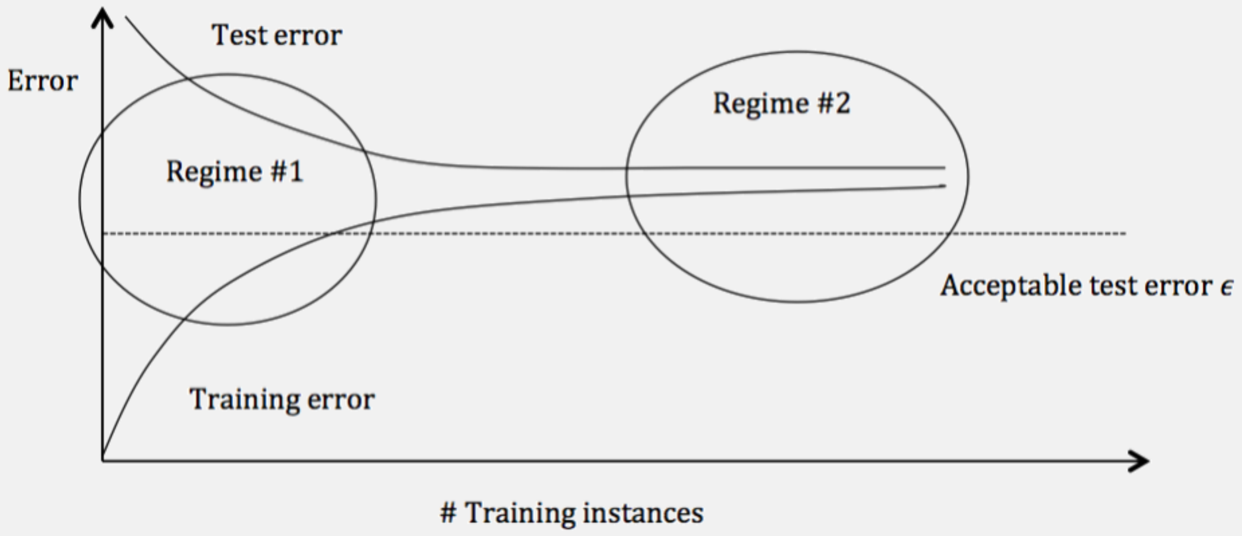
- Symptoms:
  - Training error is much lower than test error
  - Training error is lower than ε
  - Test error is higher than ε
- Remedies:
  - Add more training data
  - Reduce model complexity - complex models are prone to high variance



Where does the error come?
High Bias and High Variance

Regime 2 (High bias)

- Symptoms:
  - Training error is higher than ε
- Remedies:
  - Use more complex model
  - Add features



Where does the error come?

Bias and Variance Tradeoff

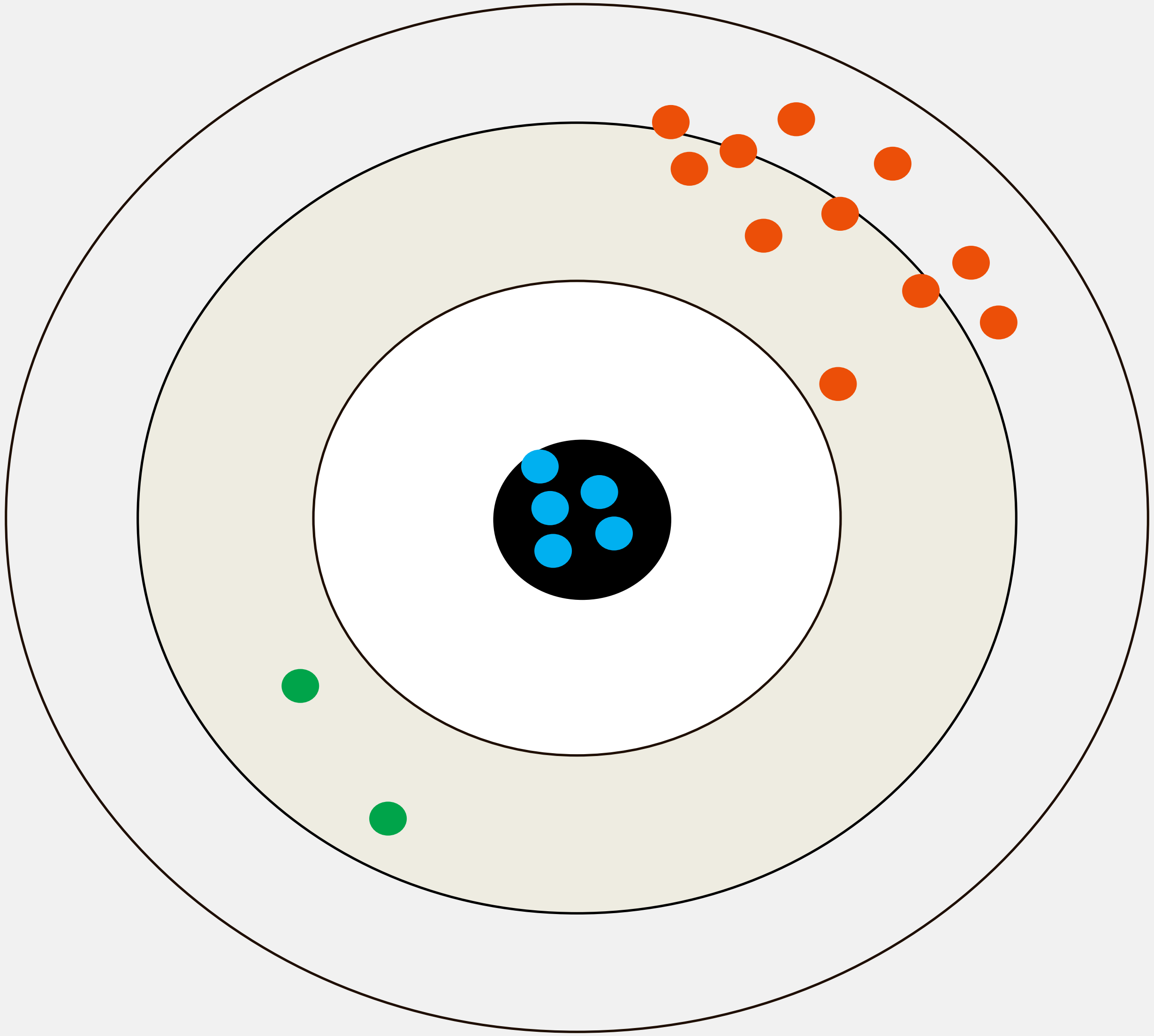
There is usually a bias-variance tradeoff caused by model complexity

Complex models (many parameters) usually have lower bias, but higher variance

Simple models (few parameters) have higher bias, but lower variance

Where does the error come?

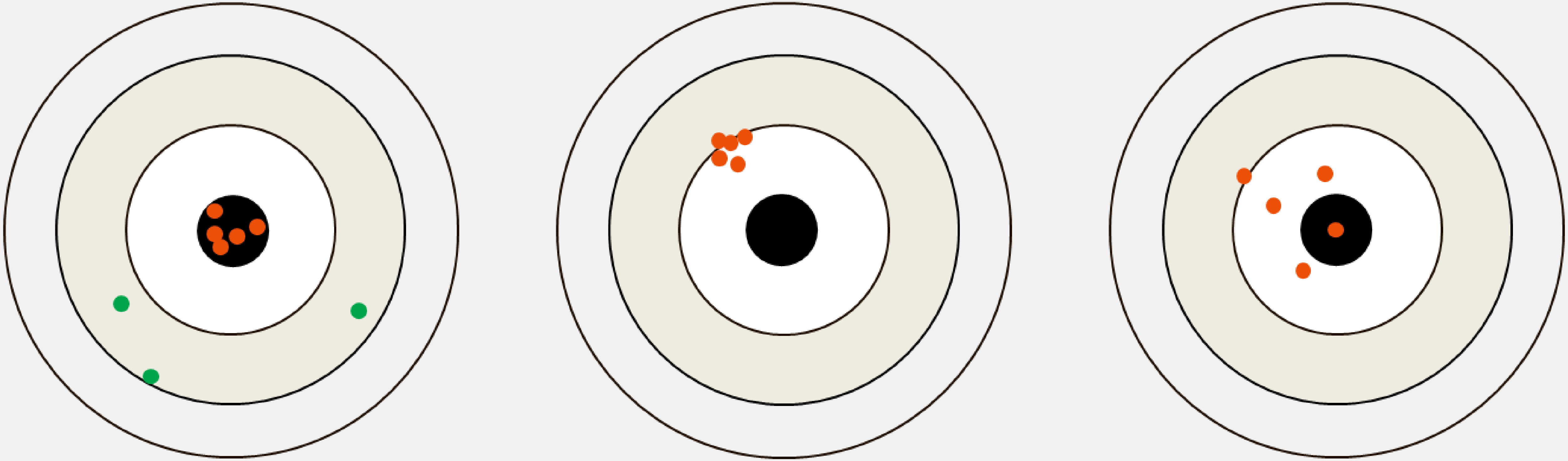
Bias and Variance Tradeoff



Where does the error come?

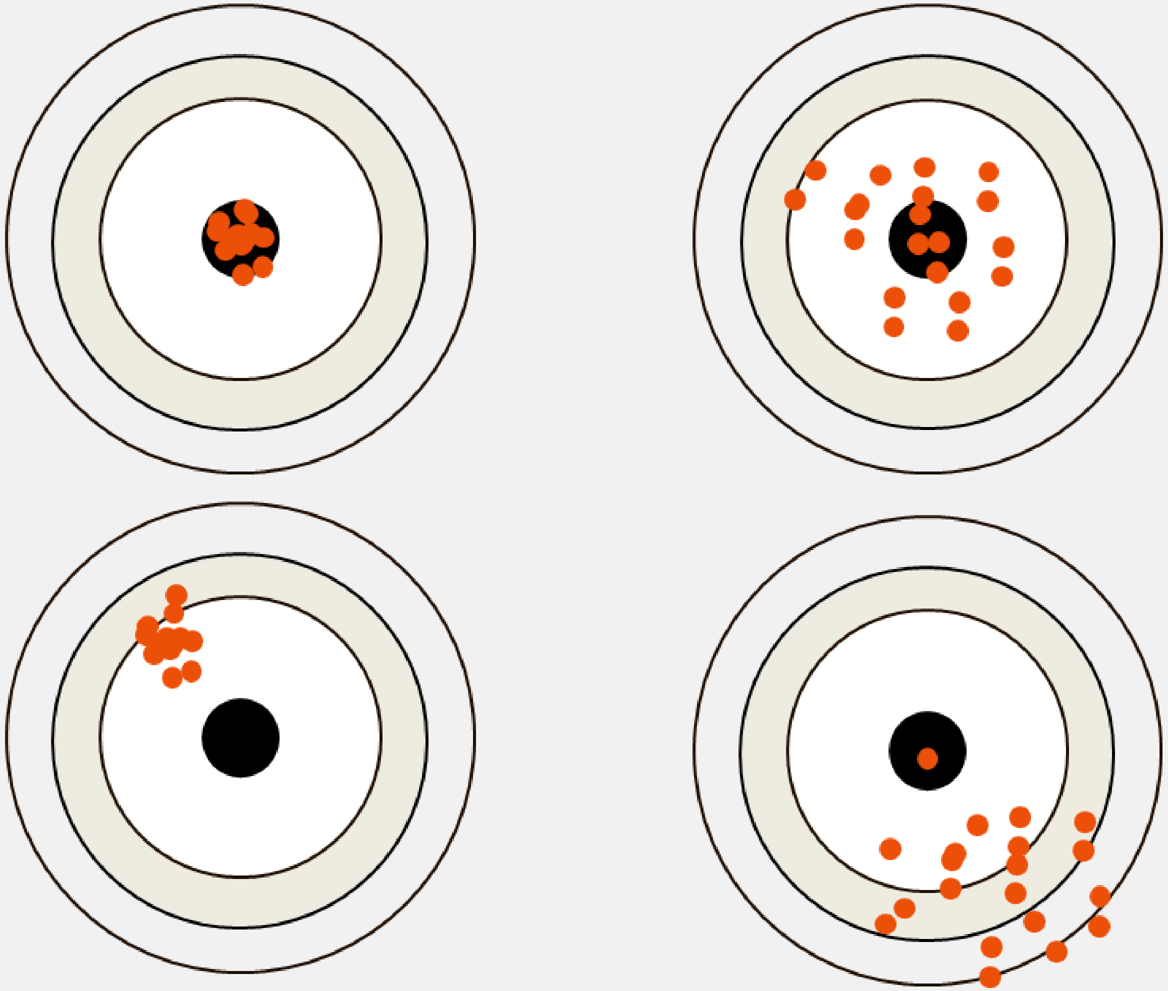
Bias and Variance Tradeoff

Noise Bias Variance



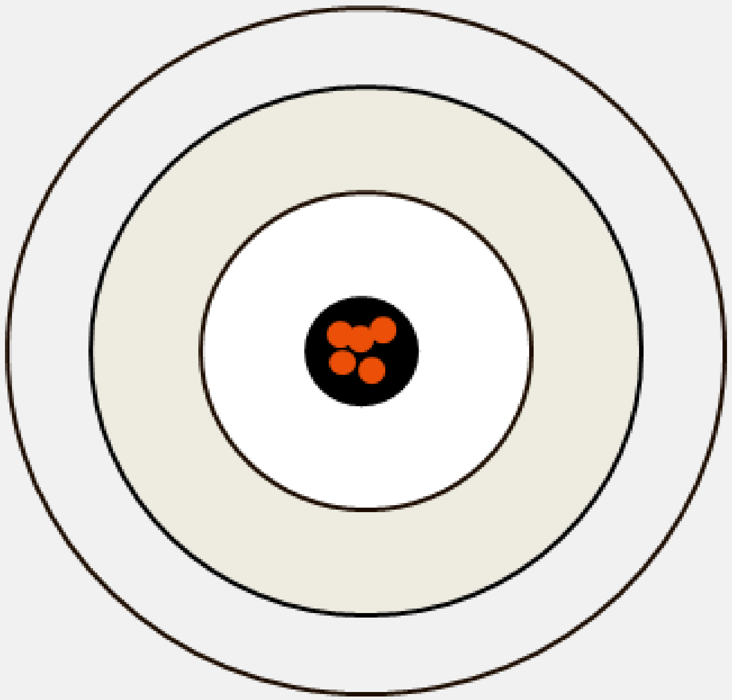
Where does the error come?

Bias and Variance Tradeoff

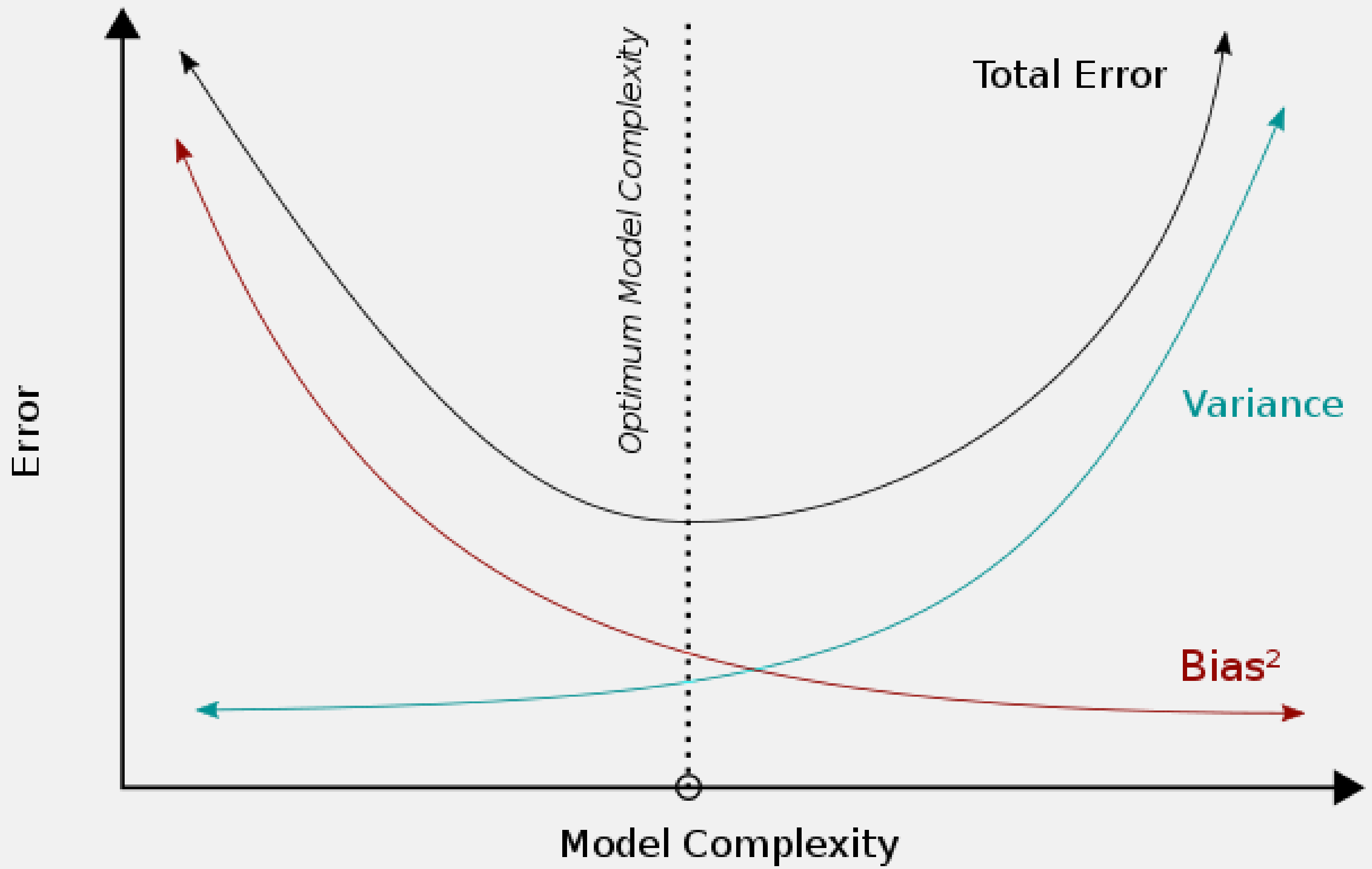


Where does the error come?

Formalizing Bias and Variance



$Bias^2 + Variance$



A champion model should maintain a balance between these two types of errors. This is known as the trade-off management of bias-variance errors

02

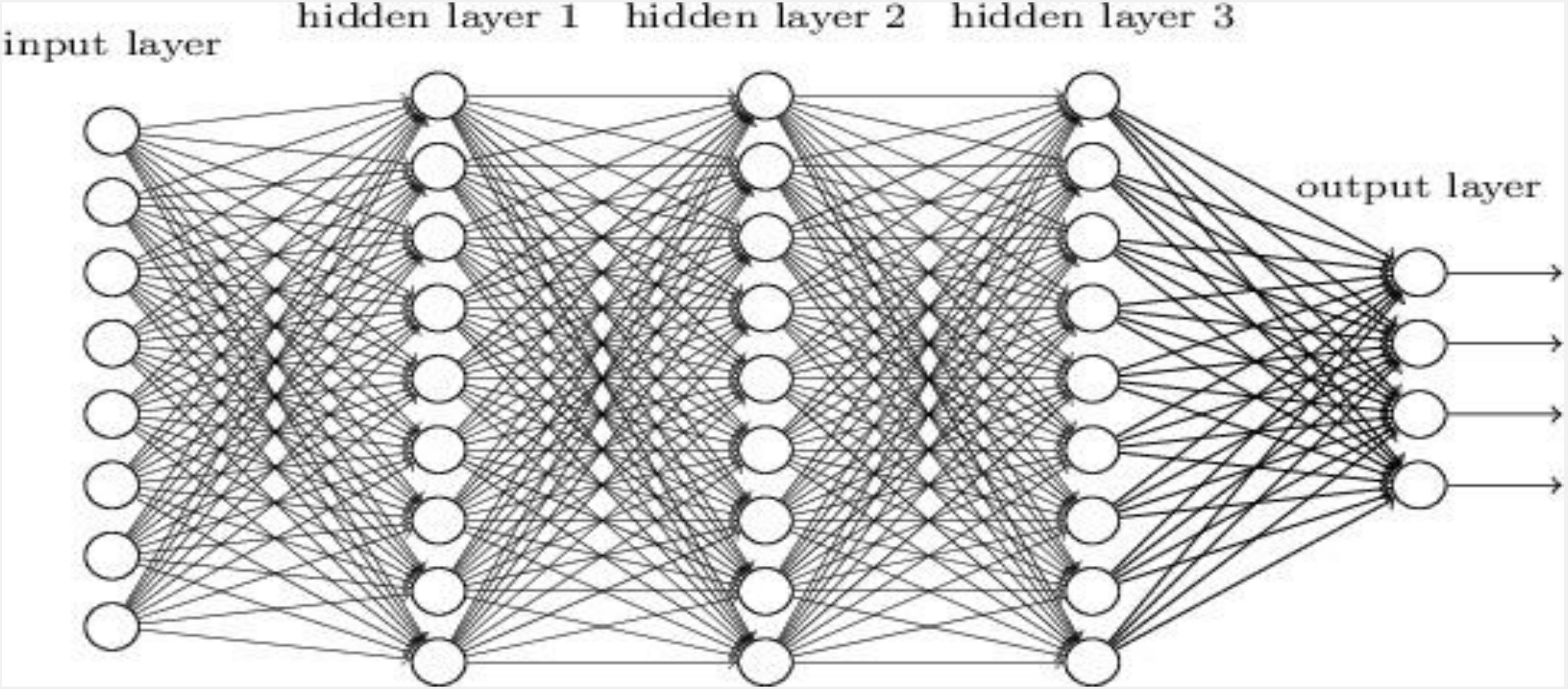
Gradient Unstable

Deep Learning Training Tips

Gradient Unstable

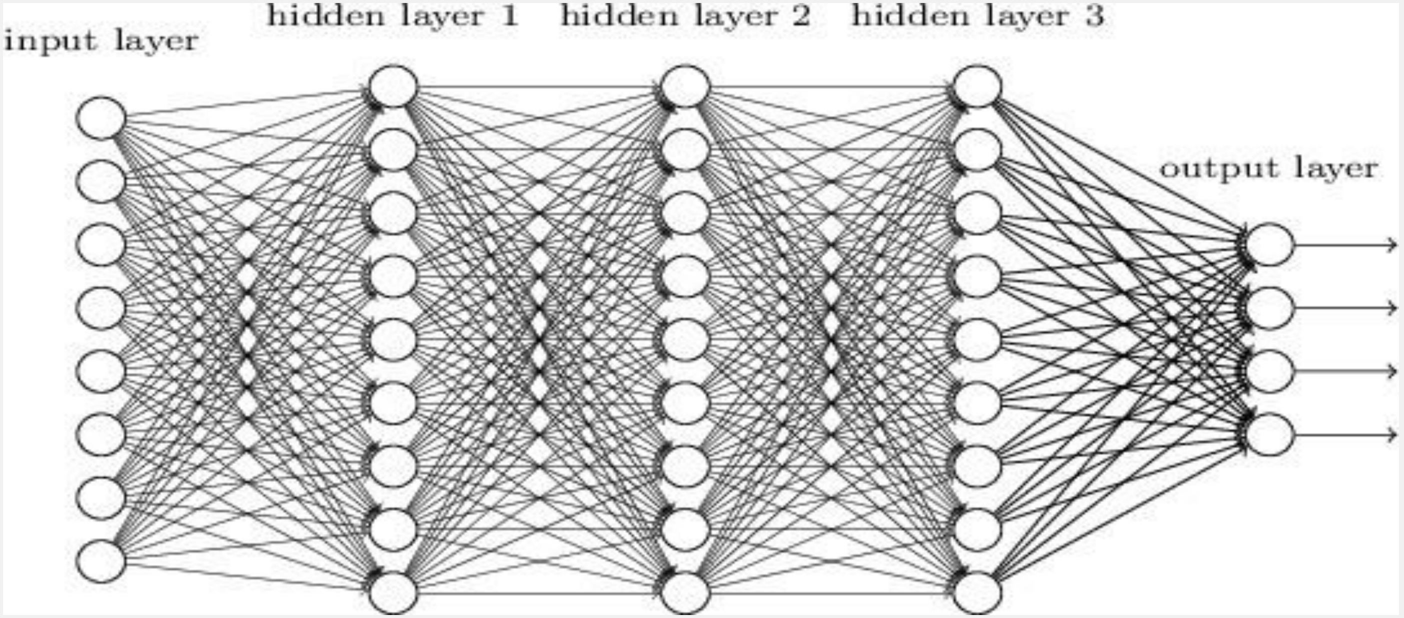
What is gradient unstable?

- ➢ Due to the "deep" layer



Deep Learning Training Tips

Gradient Unstable



$f_{i+1} = f(f_i w_{i+1})$

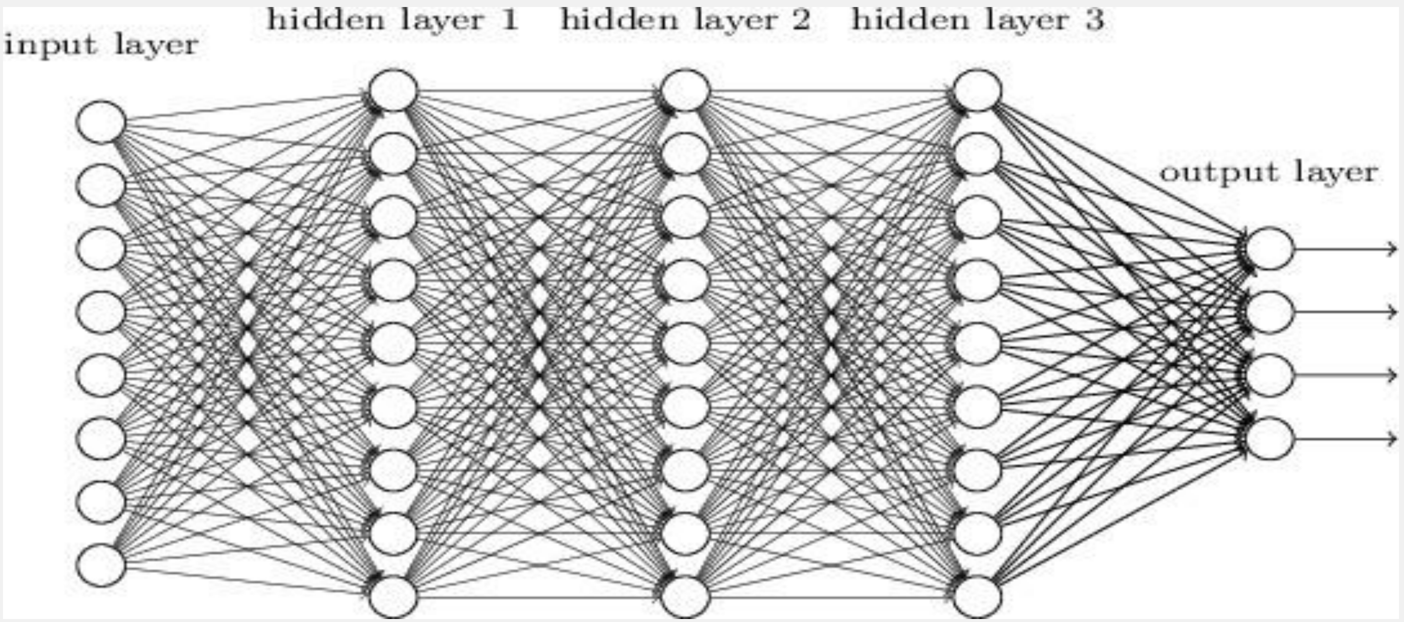
If there are three layers in the hidden layer

$f = f_3(w_3 f_2(w_2 f_1(w_{i+1})))$

$$\frac{\partial f}{\partial w_1} = \frac{\partial f_3}{\partial f_2} w_3 \frac{\partial f_2}{\partial f_1} w_2 \frac{\partial f_1}{\partial w_1}$$

Deep Learning Training Tips

Gradient Unstable

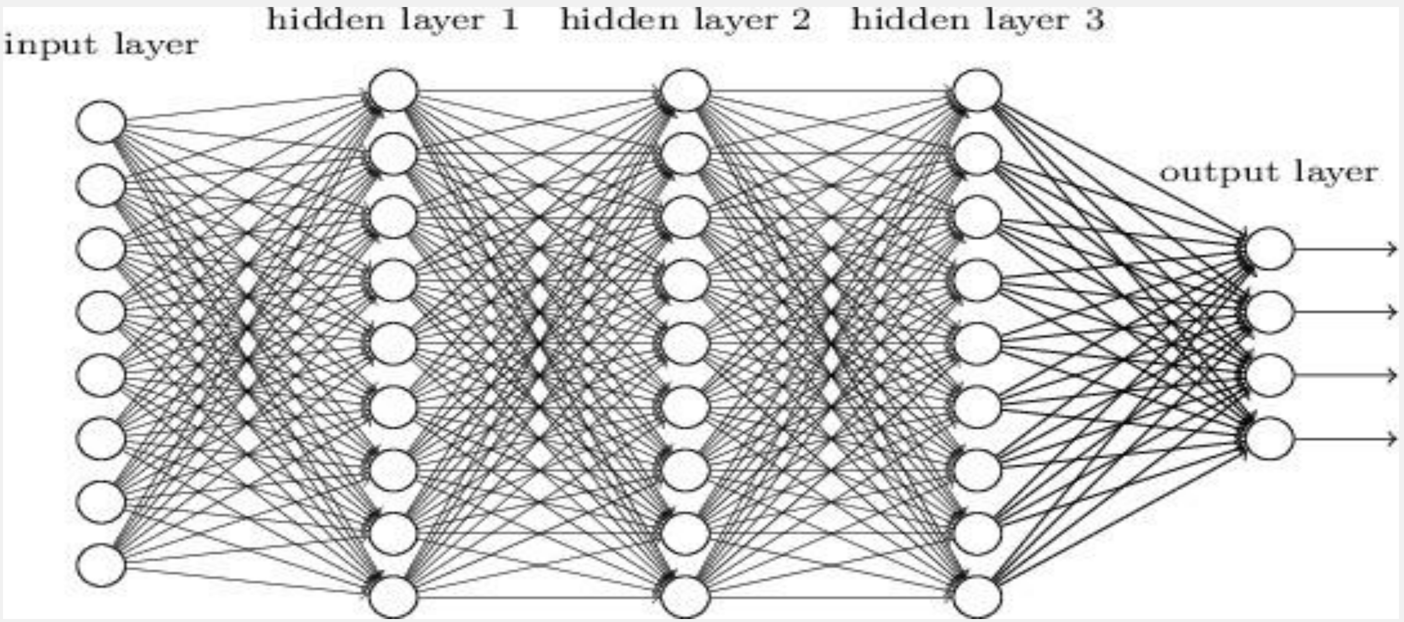


if $0 < w, D_f < 1$

Vanishing gradient

Deep Learning Training Tips

Gradient Unstable



if w, $D_f > 1$

Exploding gradient

03

Optimization

Deep Learning Training Tips

Optimization

Review optimization

- Gradient descent
- SGD (Stochastic gradient descent)
- Adagrad (Adaptive learning rate)
- RMSprop
- Adam

Deep Learning Training Tips

Optimization

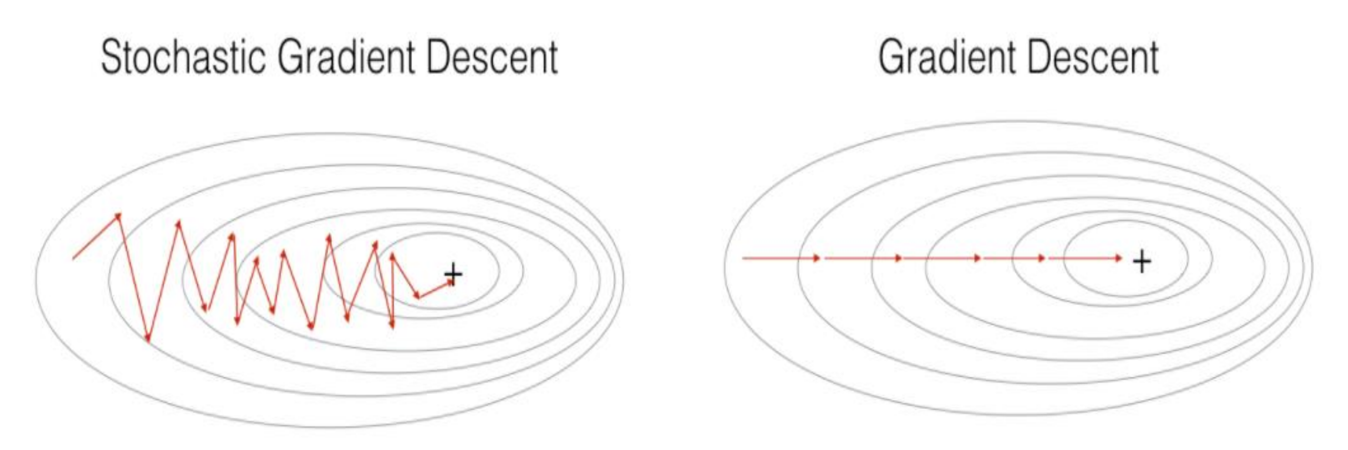
Gradient descent

- Watch all data

$$\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$

SGD

- Random watch a data
- Random watch some data



Deep Learning Training Tips

Optimization

Adagrad

$$\boldsymbol{\theta}_{t+1}^{i} \leftarrow \boldsymbol{\theta}_{t}^{i} - \frac{\eta}{\sigma_t^i} \boldsymbol{g}_t^i \qquad \sigma_t^i = \sqrt{\frac{1}{t+1} \sum_{i=0}^{t} \left(g_t^i\right)^2}$$

RMSProp

$$\boldsymbol{\theta}_{t+1}^{i} \leftarrow \boldsymbol{\theta}_{t}^{i} - \frac{\eta}{\sigma_t^i} \boldsymbol{g}_t^i \qquad \sigma_t^i = \sqrt{\alpha(\sigma_{t-1}^i)^2 + (1-\alpha)(g_t^i)^2}$$

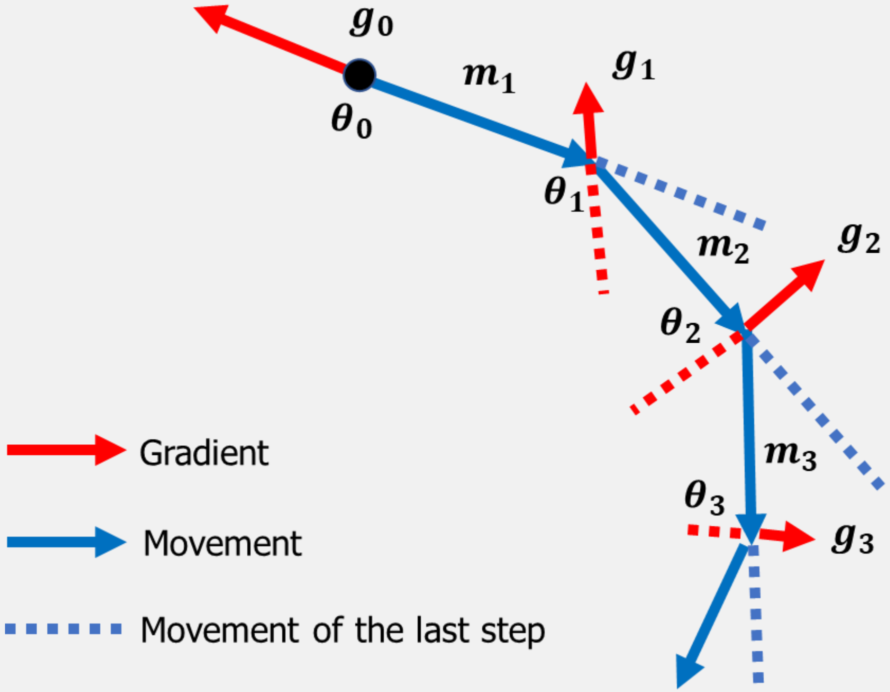
Deep Learning Training Tips

Optimization

Adam

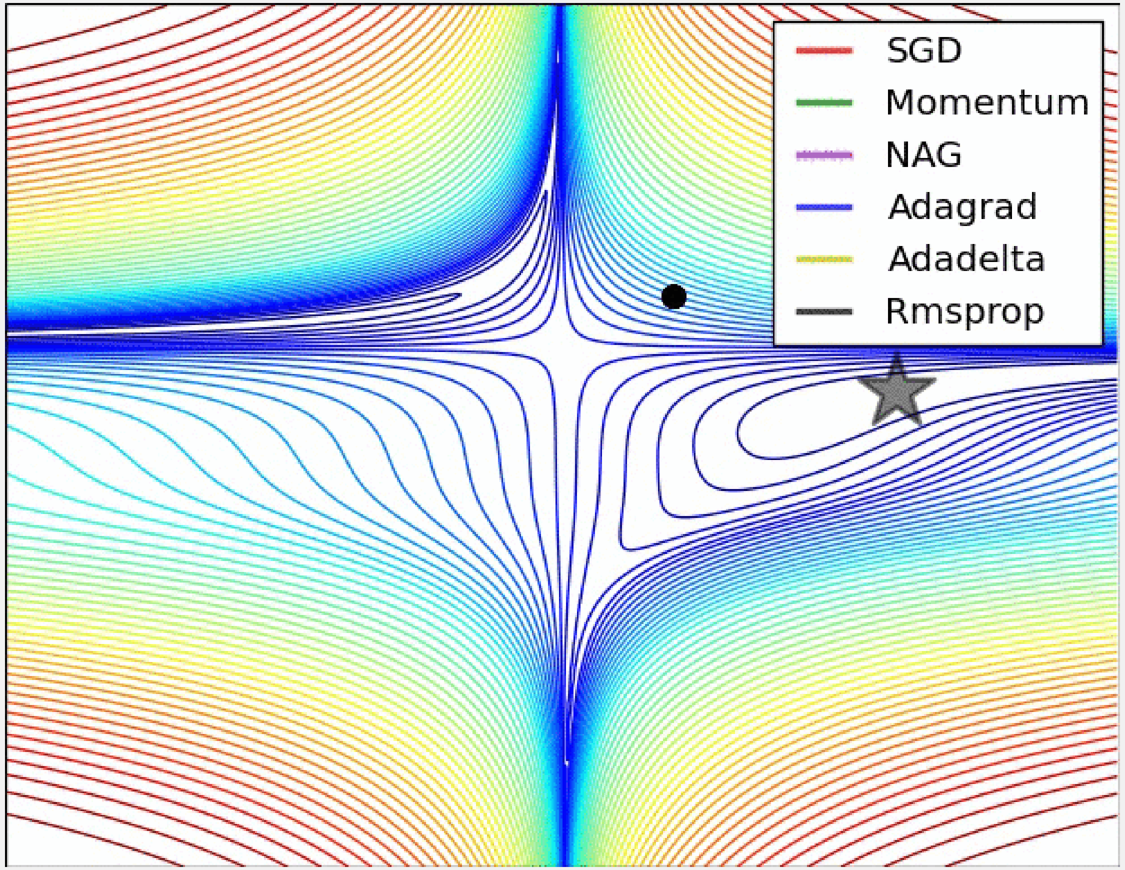
$$\boldsymbol{\theta}_{t+1}^{i} \leftarrow \boldsymbol{\theta}_{t}^{i} - \eta \frac{\widehat{m}_t^i}{\sqrt{\hat{v}_t^i + \varepsilon}} \quad m_0^i = 0 \quad m_t^i = \beta_1 (m_{t-1}^i)^2 + (1-\beta_1)(g_t^i)^2 \quad \widehat{m}_t^i = \frac{m_t^i}{1-\beta_1}$$

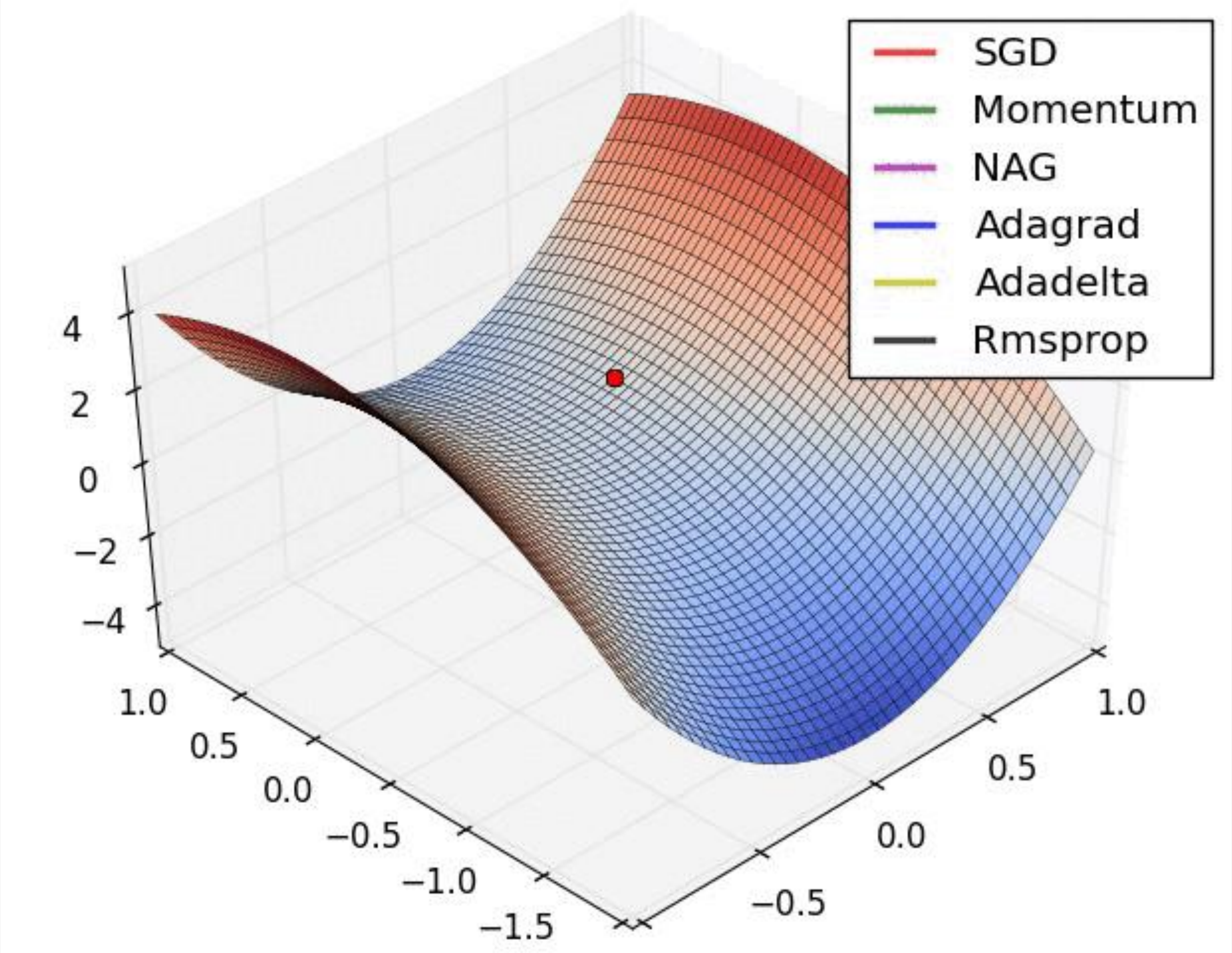
$$v_0^i = 0 \quad v_t^i = \beta_2 (v_{t-1}^i)^2 + (1-\beta_2)(g_t^i)^2 \quad \hat{v}_t^i = \frac{v_t^i}{1-\beta_2}$$



Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1-\beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1-\beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Deep Learning Training Tips
Optimization





Torch.optim

Gradient Descent	optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
SGD	optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
Adagrad	optim.Adagrad(params, lr=0.01, lr_decay=0, weight_decay=0, initial_accumulator_value=0, eps=1e-10)
RMSProp	optim.RMSprop(params, lr=0.01, alpha=0.99, eps=1e-08, weight _decay=0, momentum=0, centered=False)
Adam	optim.Adam([var1, var2], lr=0.0001)

```
optim.SGD([
                {'params': model.base.parameters()},
                {'params': model.classifier.parameters(), 'lr': 1e-3}
            ], lr=1e-2, momentum=0.9)
```

Deep Learning Training Tips

Optimization

PyTorch 的 optimizer 會透過 step() 方法，依據我們設定的學習率還有其他超參數等，來更新我們的參數。

有了優化器，我們也不用再對每一個參數做清零 zero_grad() 了，只需要透過優化器來做就好。

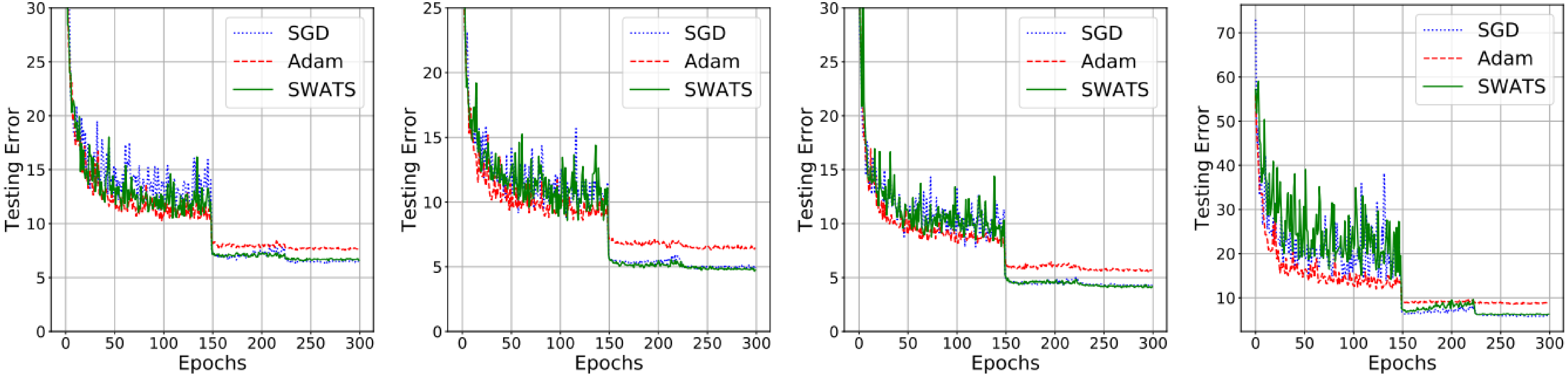
```
optimizer = torch.optim.Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=lr)  # Choose optimize
```

```
if model.training:
    optimizer.zero_grad()
    out_loss.backward()
    optimizer.step()
```

Deep Learning Training Tips
Optimization

Best of both worlds

- ➢ Using two different optimizers



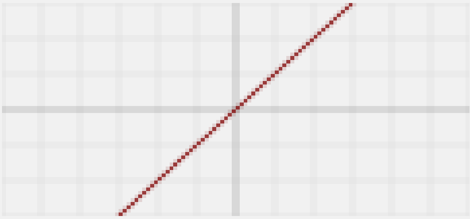
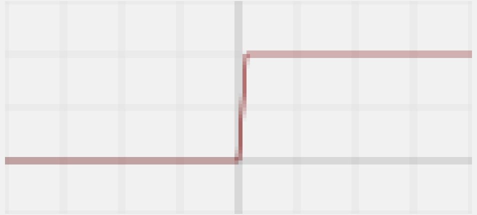
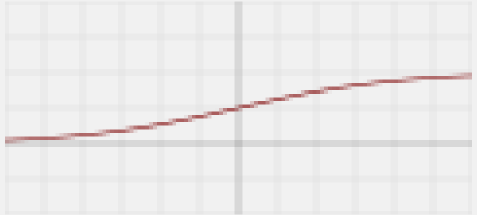
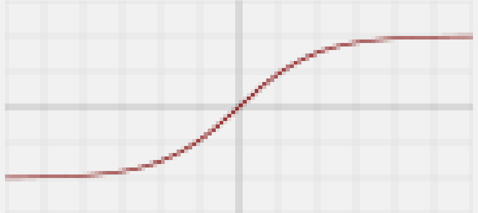
(a) ResNet-32 — CIFAR-10 (b) DenseNet — CIFAR-10 (c) PyramidNet — CIFAR-10 (d) SENet — CIFAR-10

Reference

04

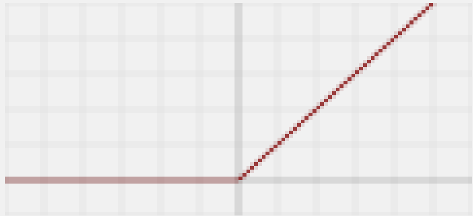
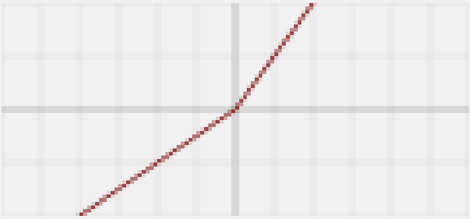
Activation Functions

Deep Learning Training Tips
Activation Functions

Identity		$f(x) = x$	
Step		$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geq 0 \end{cases}$	
Sigmoid		$f(x) = \sigma(x) = \dfrac{1}{1+e^{-x}}$	torch.nn.Sigmoid()
tanh		$f(x) = \tanh(x) = \dfrac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	torch.nn.Tanh()

Deep Learning Training Tips

Activation Functions

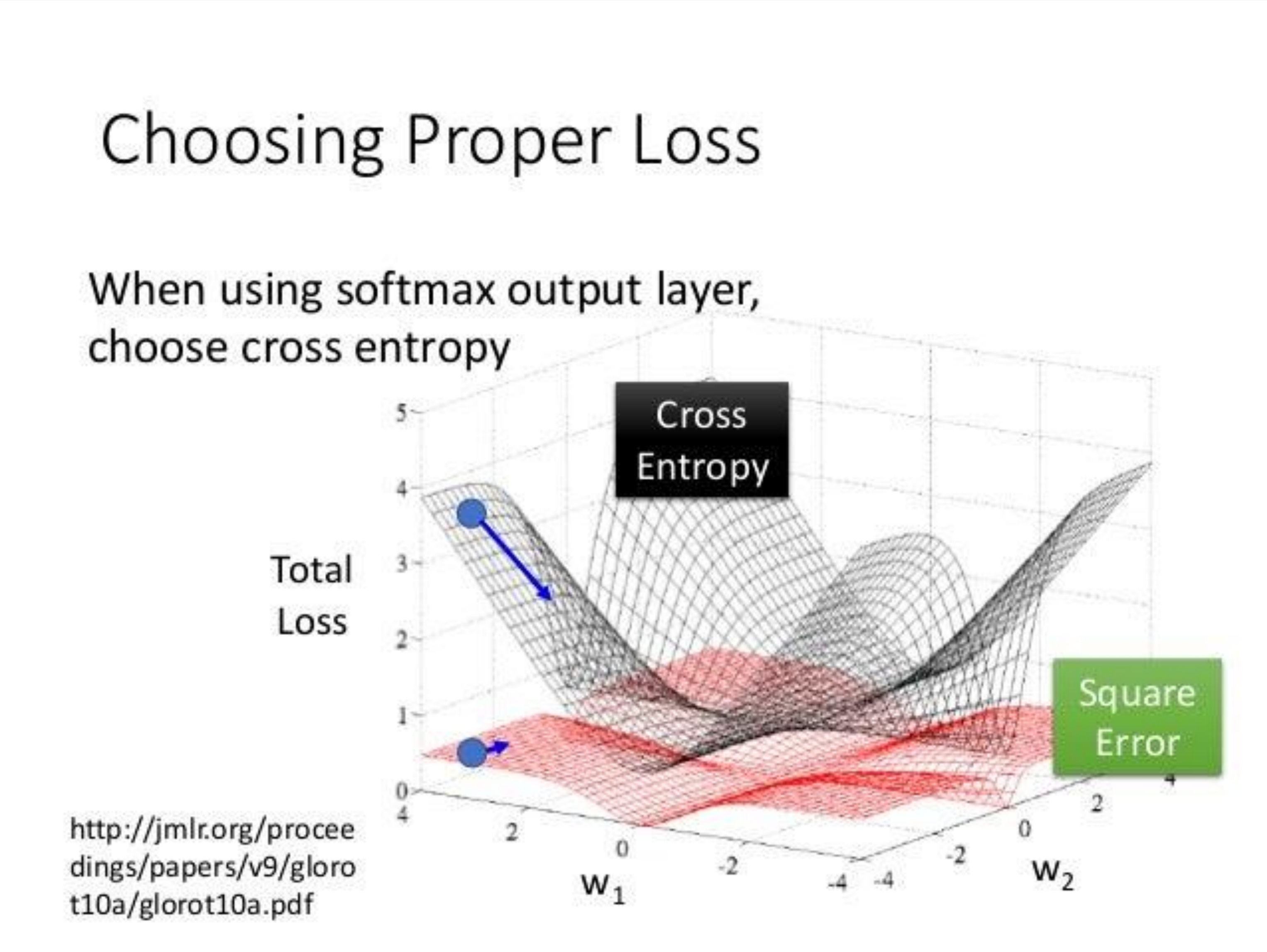
Relu		$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$	torch.nn.ReLU()
Leaky Relu		$f(x) = \begin{cases} 0.01x & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$	torch.nn.LeakyReLU()

05

Loss Functions

Deep Learning Training Tips

Loss Functions

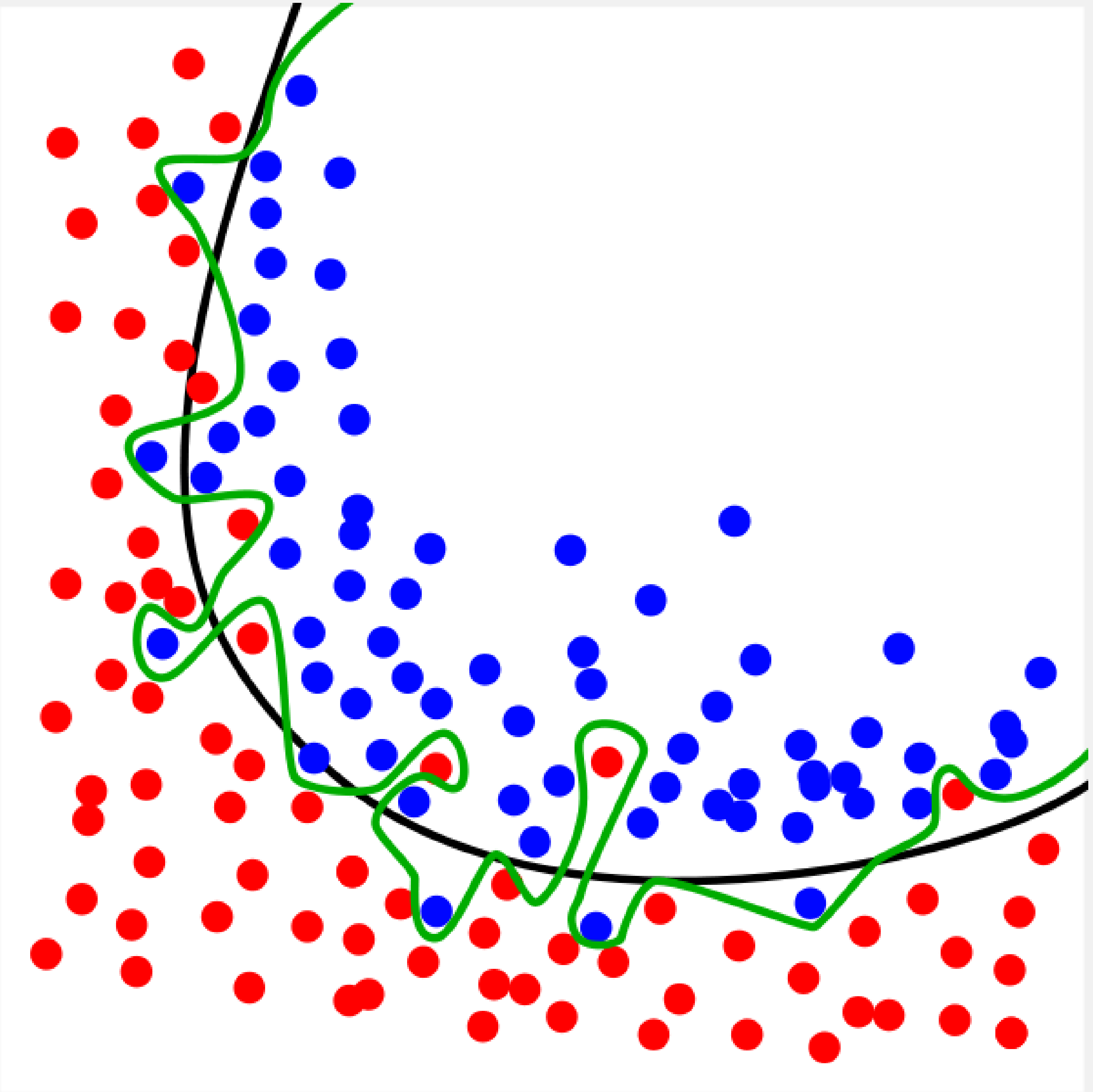


06

Overfitting

Deep Learning Training Tips

Overfitting



Deep Learning Training Tips

Overfitting

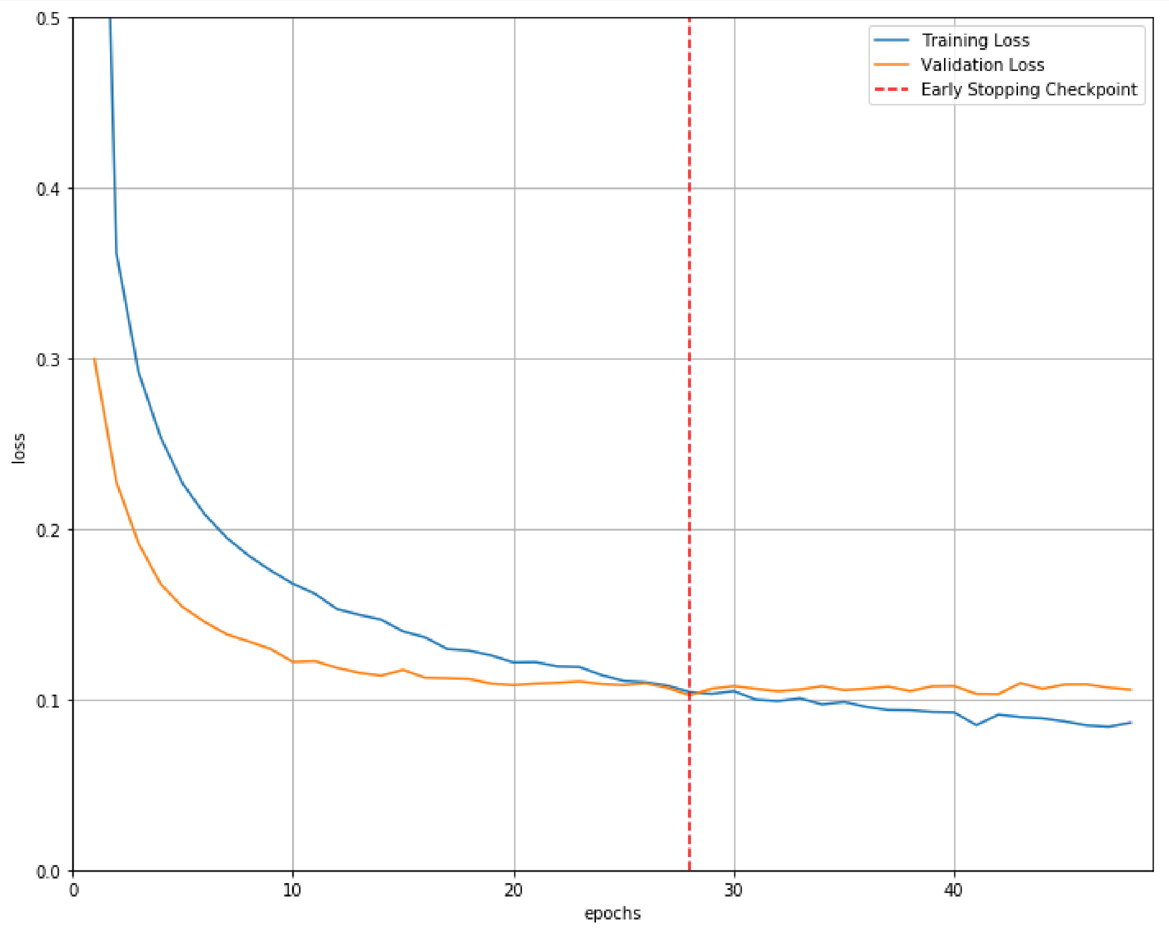
Why overfitting?

- Model complexity
- Data imbalance
- Noise
- Iterations

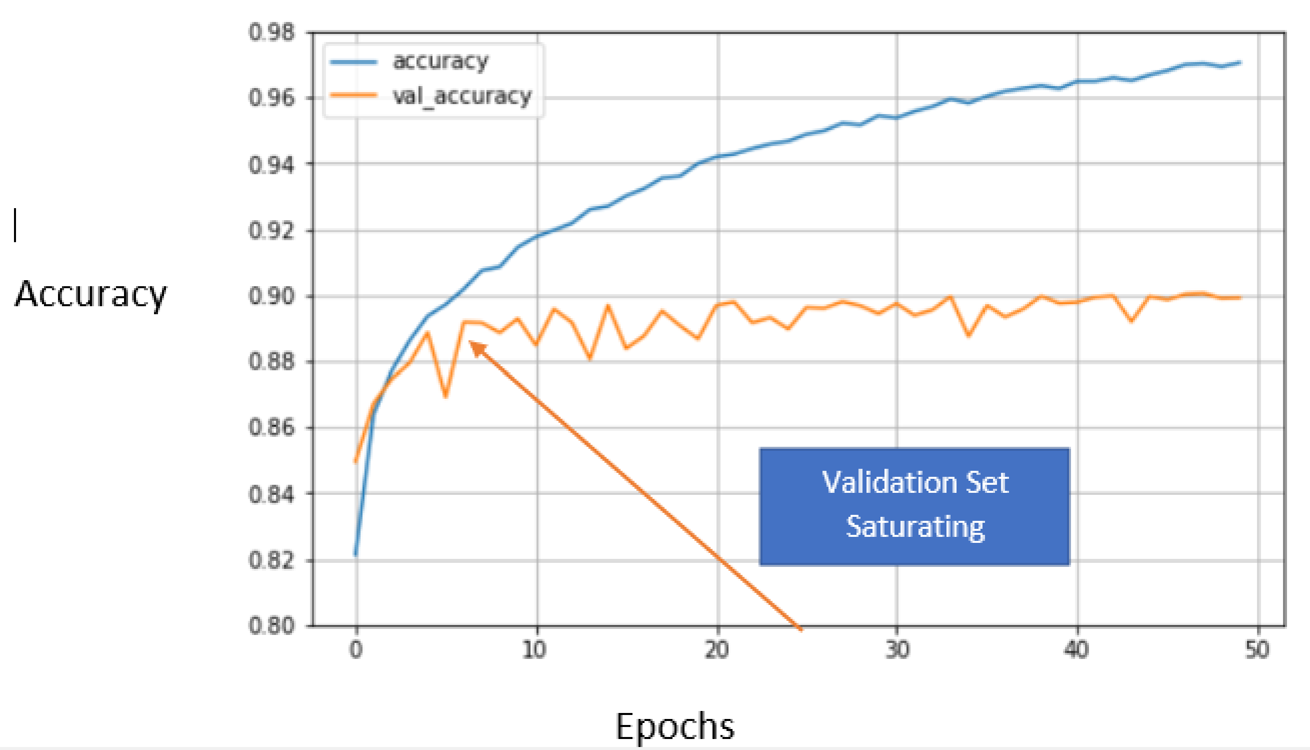
Deep Learning Training Tips
Overfitting

Early stopping

- ➢ Stop training before model overfitting or useless training



[Reference]



Deep Learning Training Tips

Overfitting

Regularization

- Introduce additional information

L1

$$cost = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2

$$cost = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^{M} W_j^{\,2}}_{\text{Regularization}}$$

Loss function

Regularization

Weight Decay

$$\mathrm{L}'(\theta) = \mathrm{L}(\theta) + \lambda \frac{1}{2}||\theta||_2 \quad Gradient: \frac{\partial \mathrm{L}'}{\partial w} = \frac{\partial \mathrm{L}}{\partial w} + \lambda w$$

$$\text{Update}: w^{t+1} \leftarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \frac{\partial \mathrm{L}}{\partial w} + \lambda w^t$$

$$= \underline{(1 - \eta\lambda)w^t} - \eta \frac{\partial L}{\partial w}$$

Close to 0

Deep Learning Training Tips

Overfitting

L1 Regularization

```
regularization_loss = 0
for param in model.parameters():
    regularization_loss += torch.sum(abs(param))

calssify_loss = criterion(pred,target)
loss = classify_loss + lamda * regularization_loss

optimizer.zero_grad()
loss.backward()
optimizer.step()
```

L2 Regularization

weight_decay (float, optional): weight decay (L2 penalty) (default: 0)

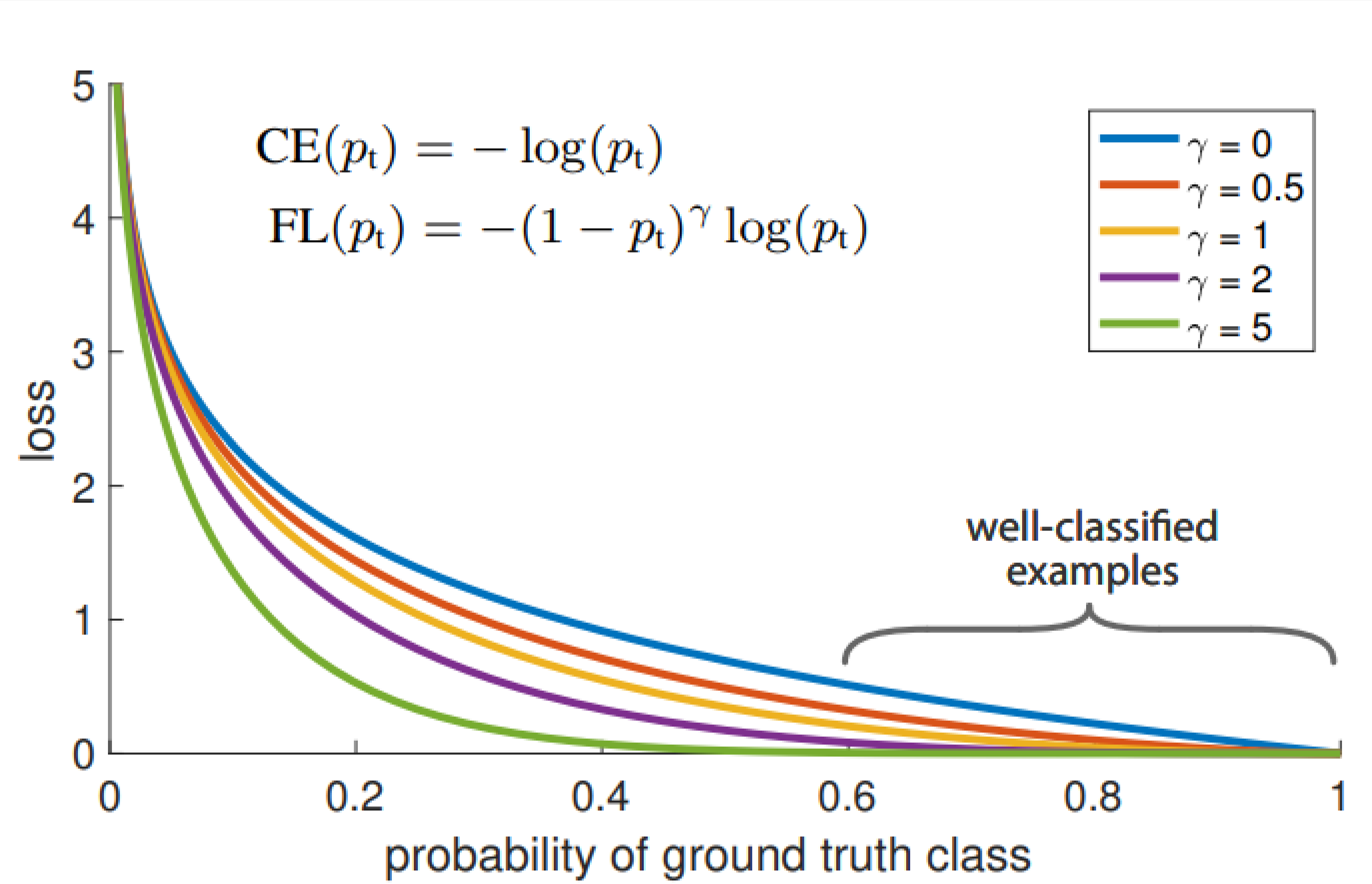
optimizer = optim.Adam(model.parameters(),lr=learning_rate,weight_decay=0.01)

Deep Learning Training Tips
Overfitting

Data imbalance

➢ Using class weights

- Increase the penalty for low number of class



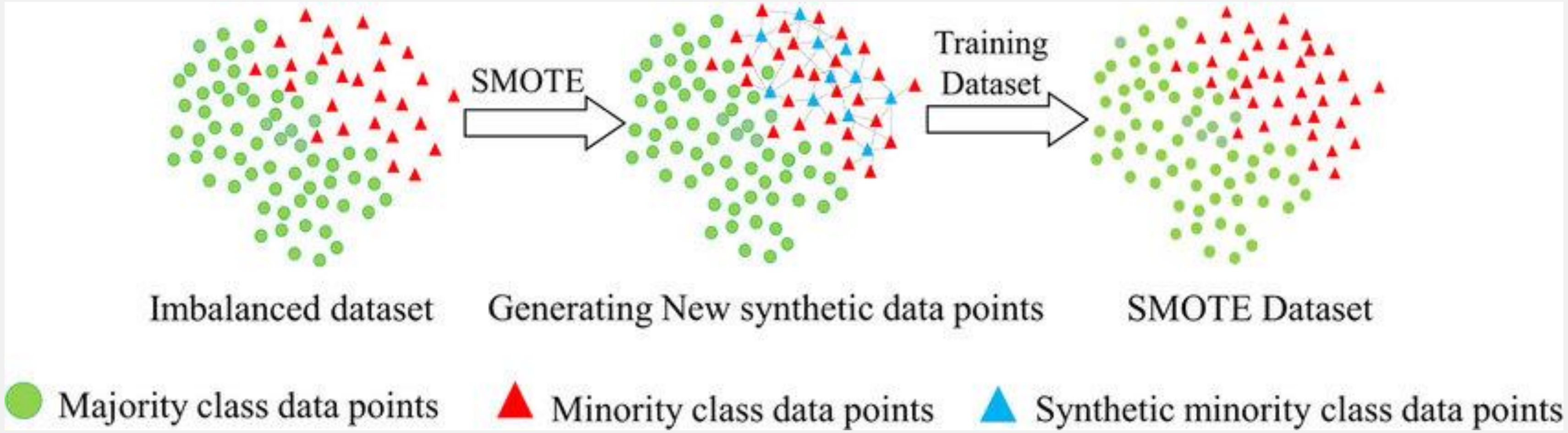
[Focal loss](#)

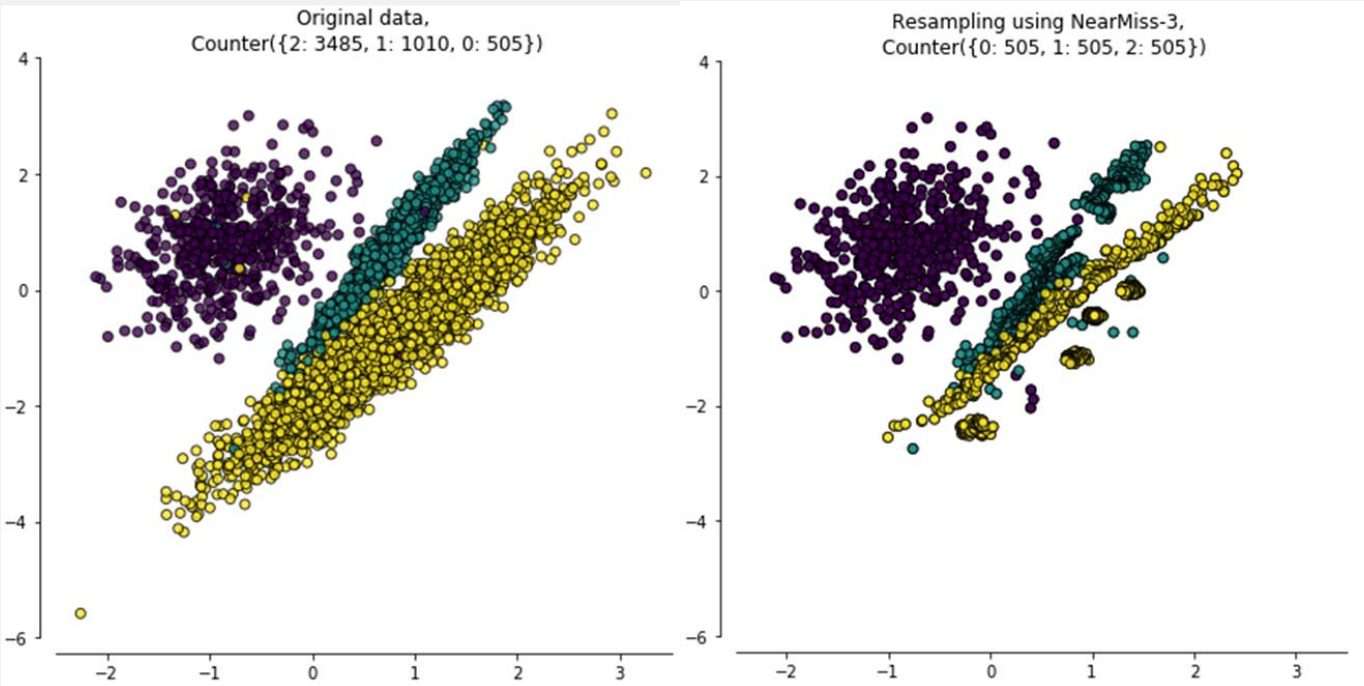
Deep Learning Training Tips

Overfitting

Data imbalance

- ➢ Using sampling
  - [Over-sampling]
  - [Under-sampling]



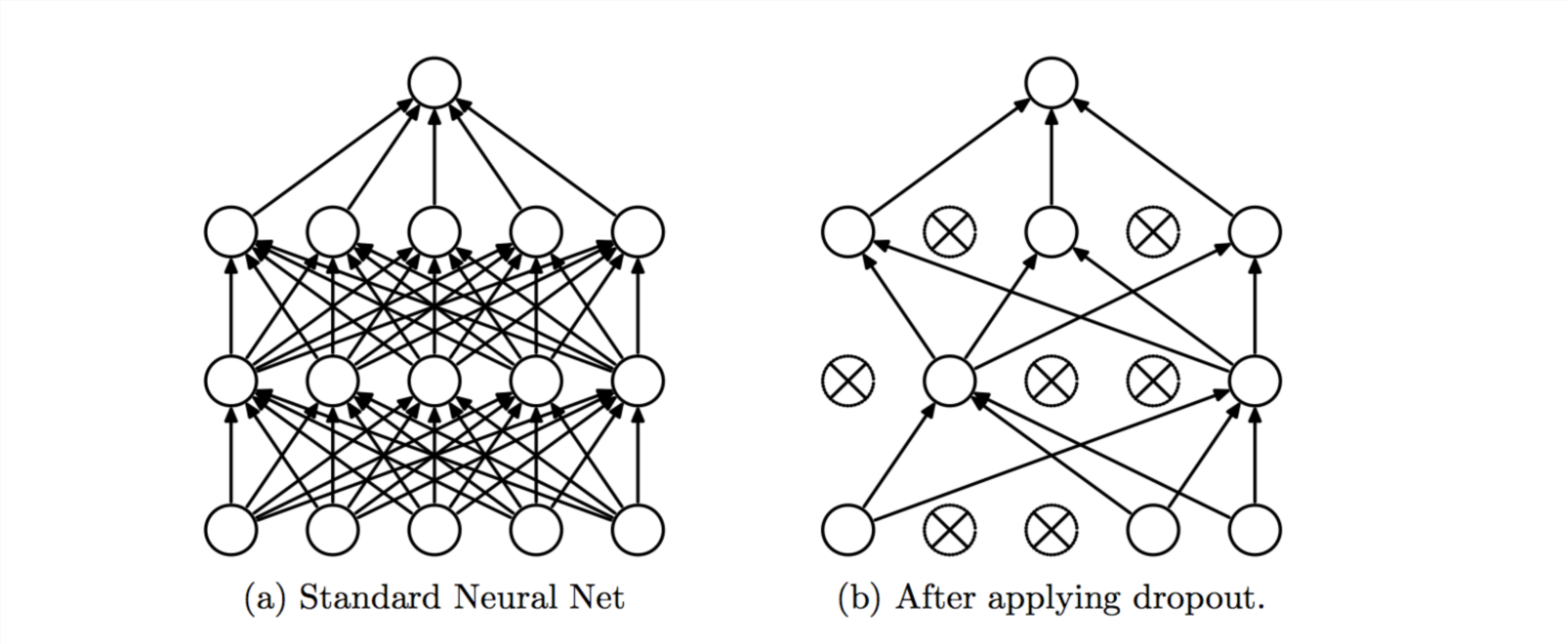


Deep Learning Training Tips

Overfitting

Dropout

- ➢ Random freeze some nodes



torch.nn.Dropout(p=0.5)

Deep Learning Training Tips

Overfitting

Batch normalization

- feature normalization

Advantages

- Slow down the vanishing gradient
- Solve the internal covariate shift
- Eliminate the need of dropout

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching 4.9% top-5 validation error (and 4.8% test error), exceeding the accuracy of human raters.

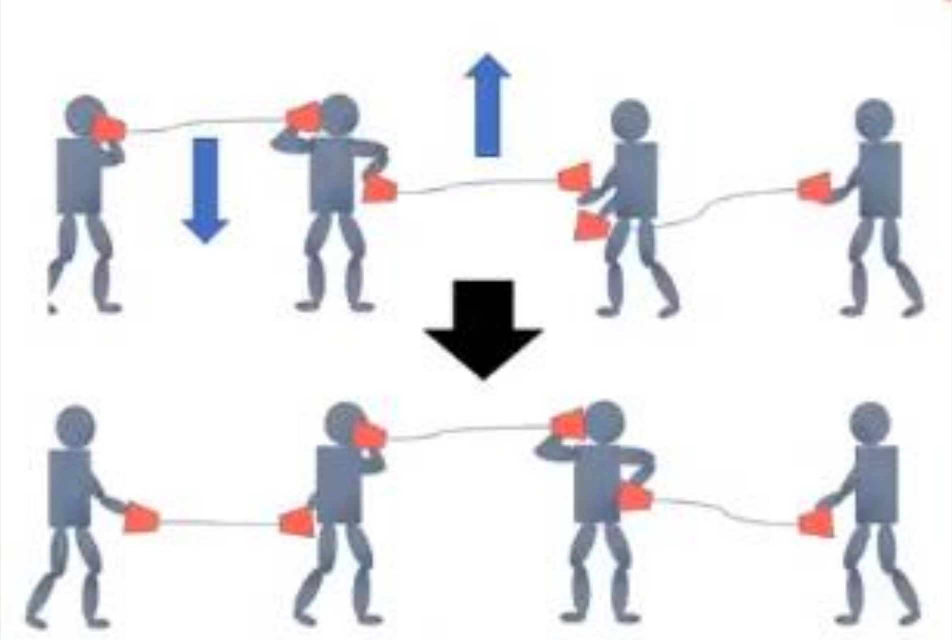
Deep Learning Training Tips
Overfitting

Covariate shift

- The distribution in training set is inconsistent with that in test set

Internal covariate shift

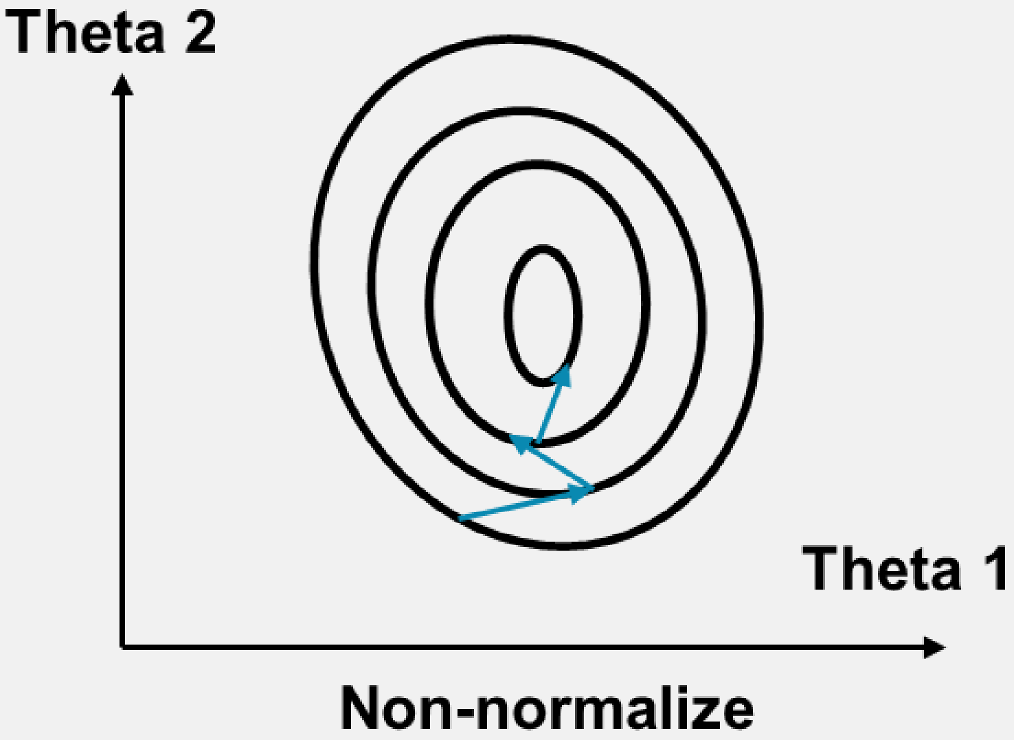
- The distribution is inconsistent in each layer of the neural network

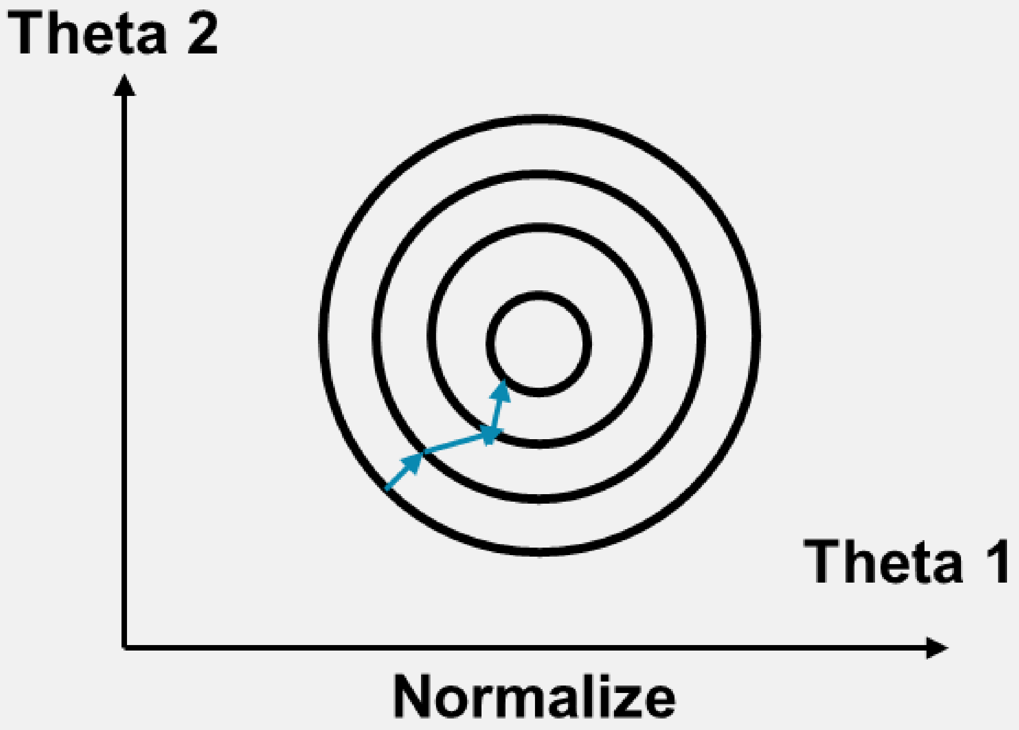


Deep Learning Training Tips

Overfitting

Internal covariate shift

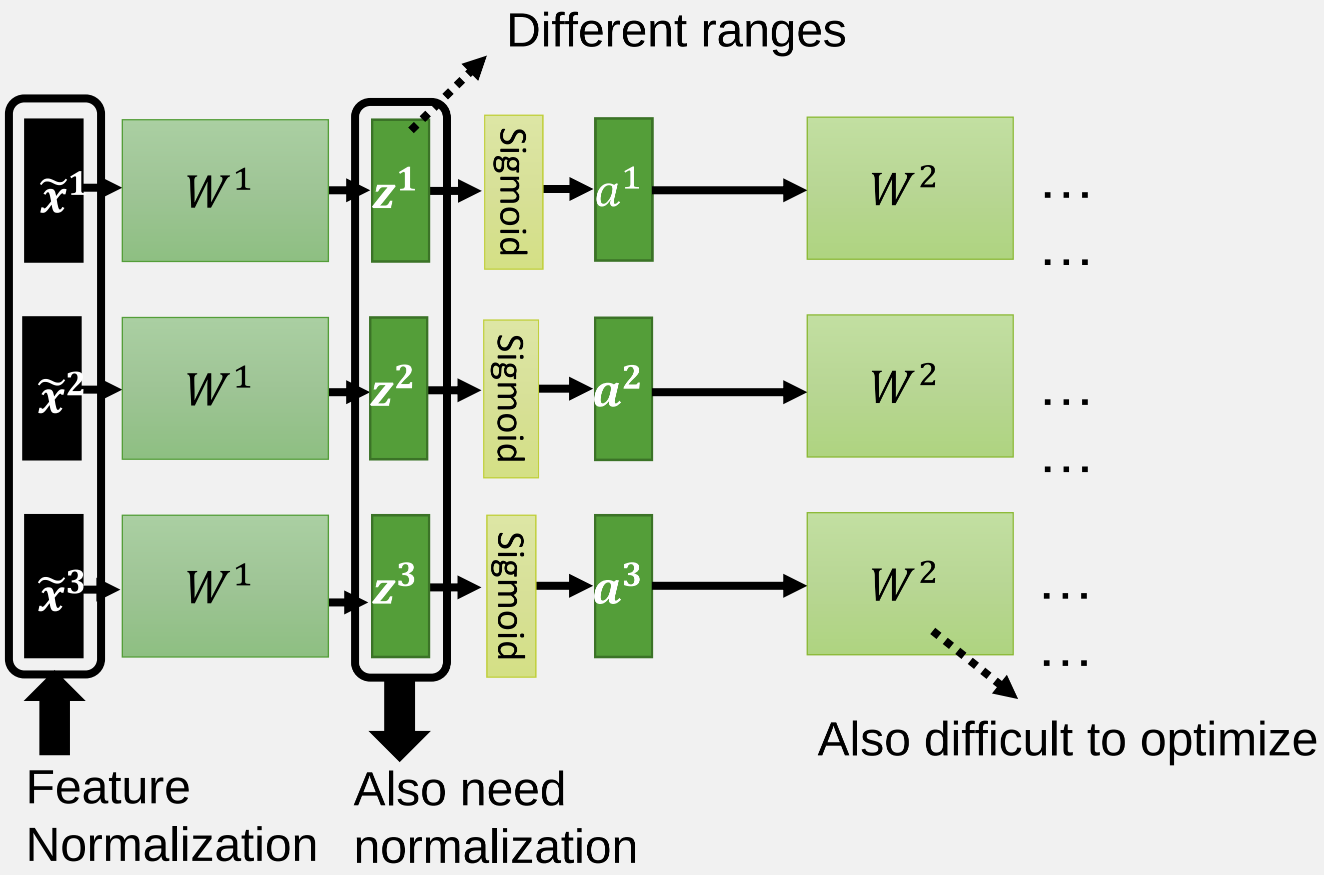




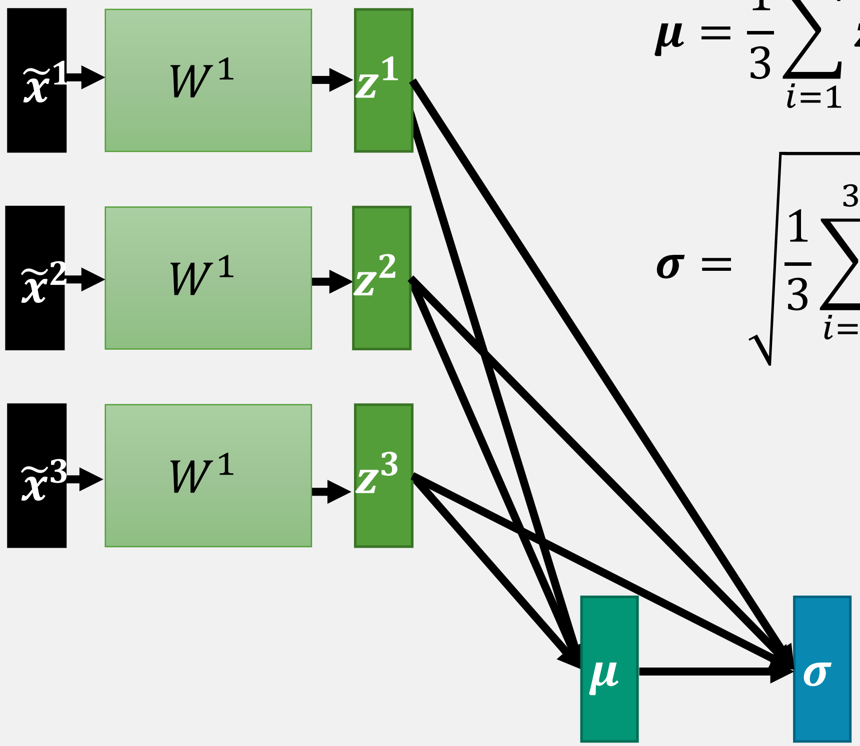
Deep Learning Training Tips

Overfitting

Deep Learning



Deep Learning



$$\boldsymbol{\mu} = \frac{1}{3}\sum_{i=1}^{3} \boldsymbol{z}^i$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{3}\sum_{i=1}^{3} (\boldsymbol{z}^i - \boldsymbol{\mu})^2}$$

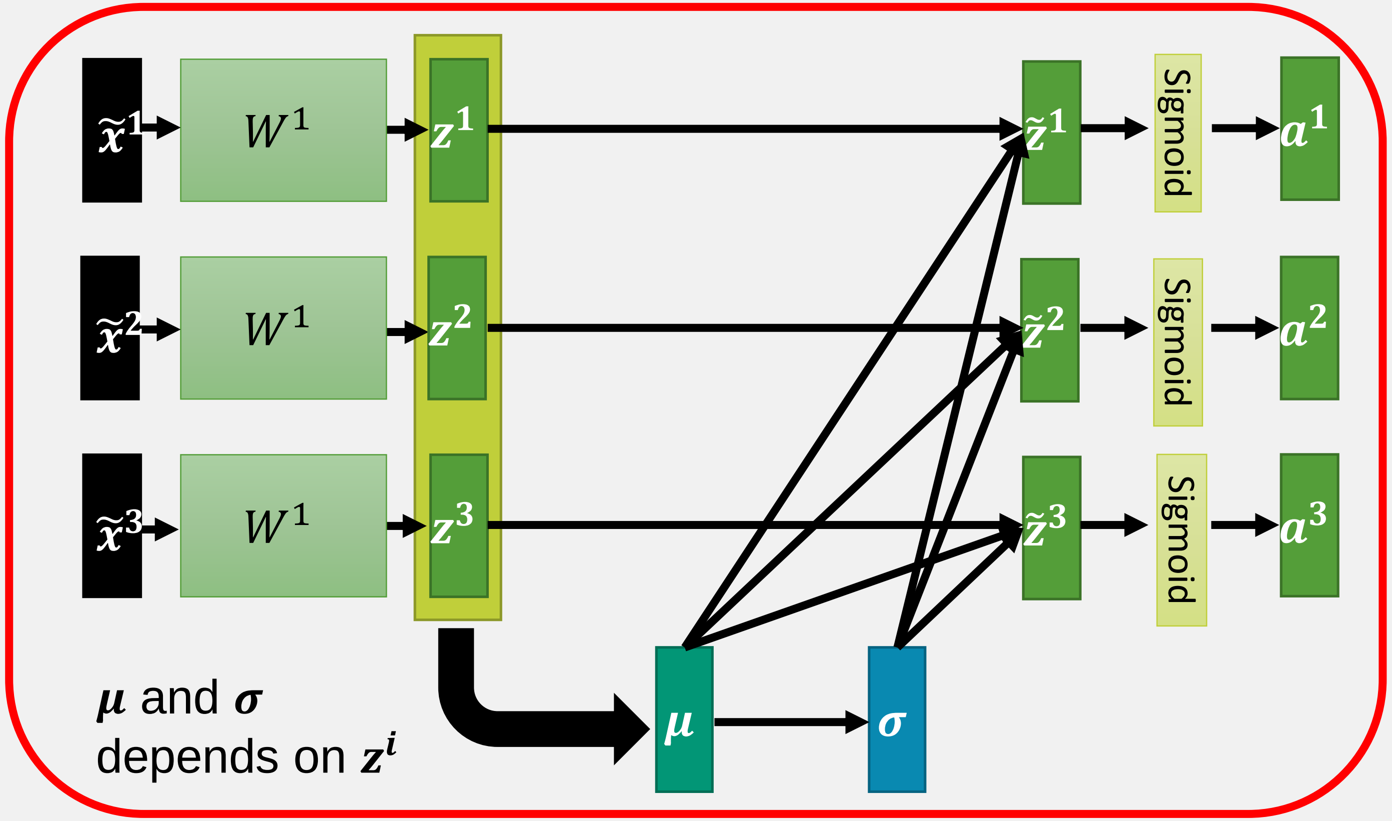
Deep Learning Training Tips

Overfitting

Deep Learning

$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

This is a large network!



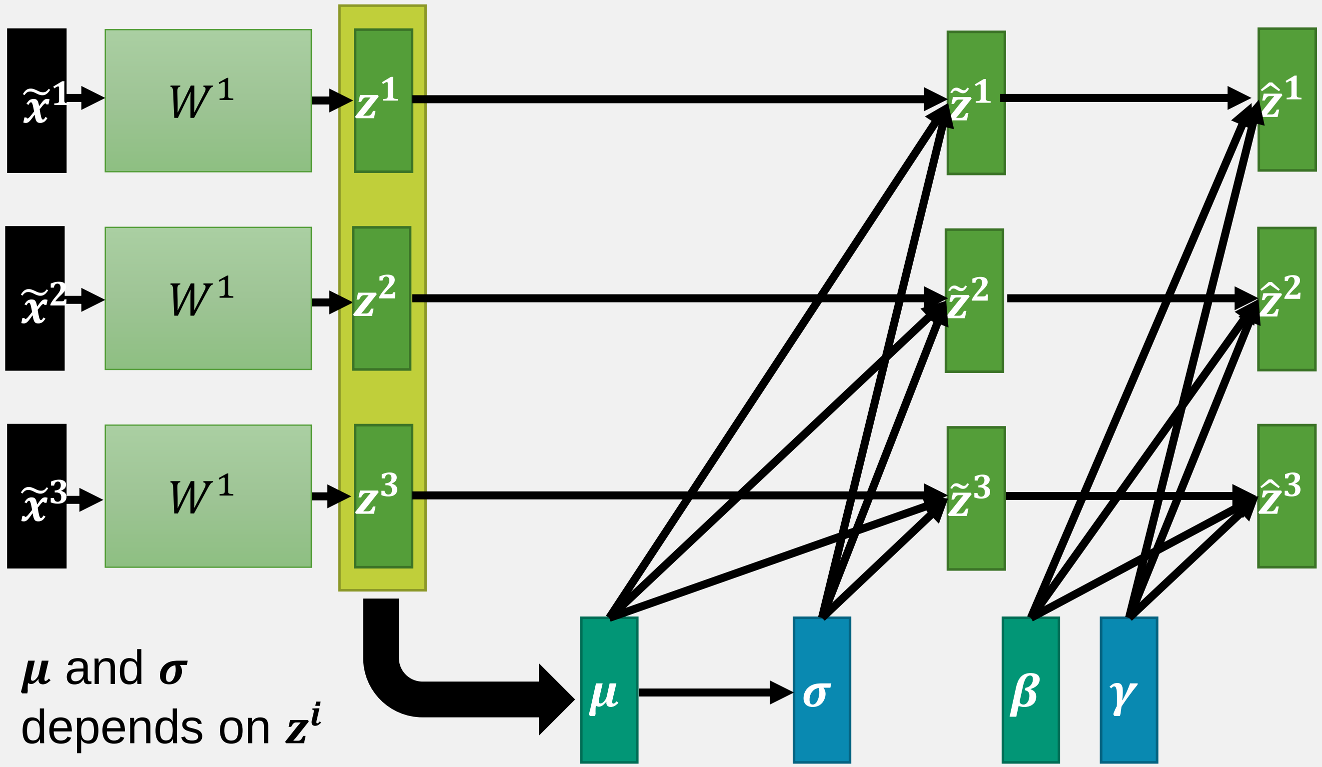
Deep Learning Training Tips

Overfitting

Deep Learning

$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

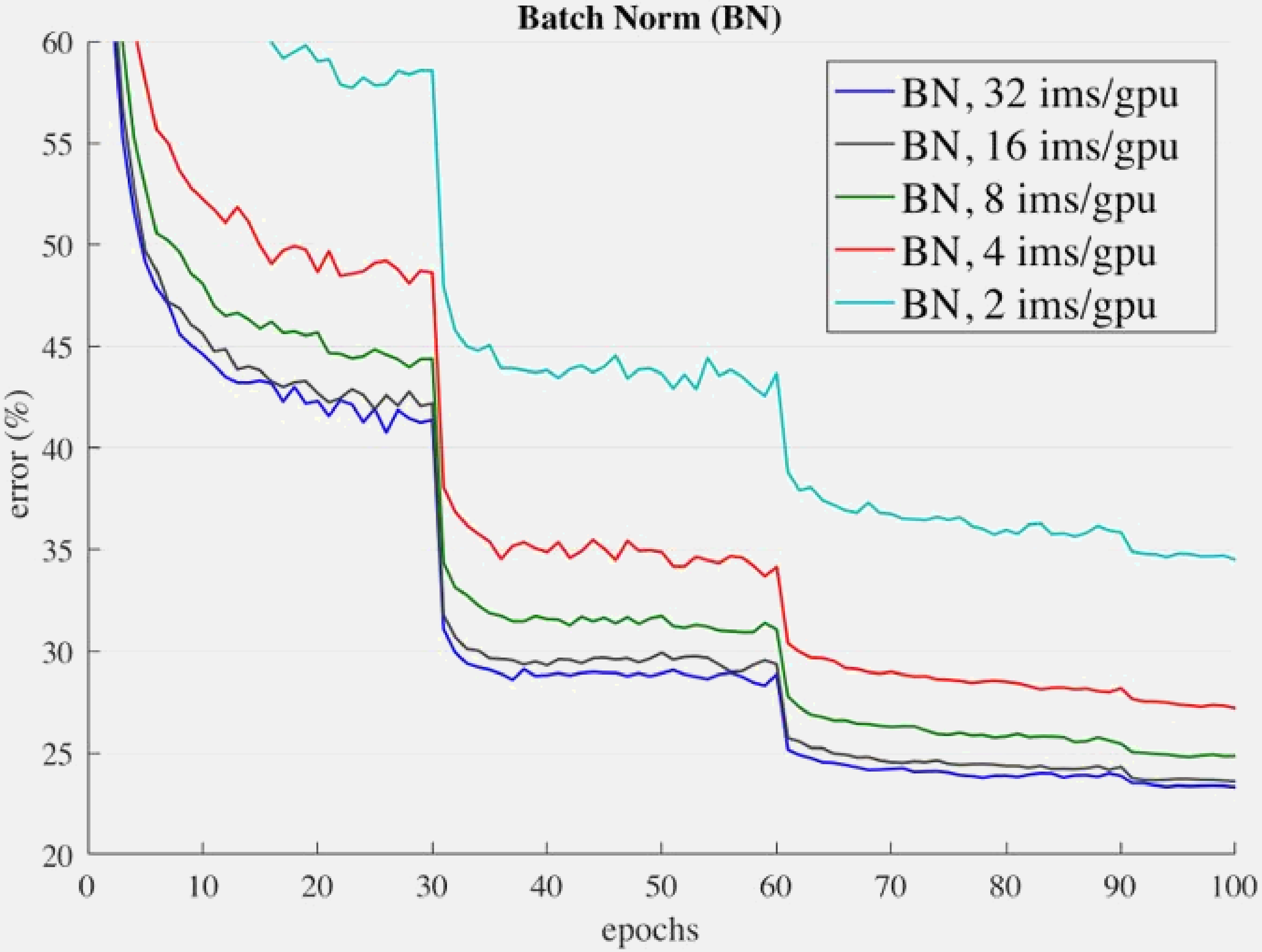
$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$



Deep Learning Training Tips

Overfitting

Deep Learning



Deep Learning

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

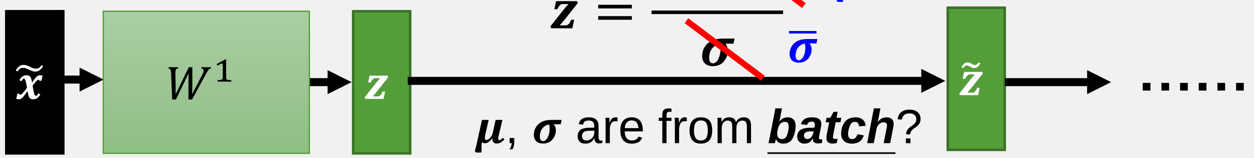
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Deep Learning Training Tips

Overfitting

Deep Learning

$$\tilde{\boldsymbol{z}} = \frac{\boldsymbol{z} - \cancel{\boldsymbol{\mu}}\ \ \bar{\boldsymbol{\mu}}}{\cancel{\boldsymbol{\sigma}}\ \ \bar{\boldsymbol{\sigma}}}$$



$\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ are from ***batch***?

We do not always have ***batch*** at testing stage.

Computing the moving average of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of the batches during training.

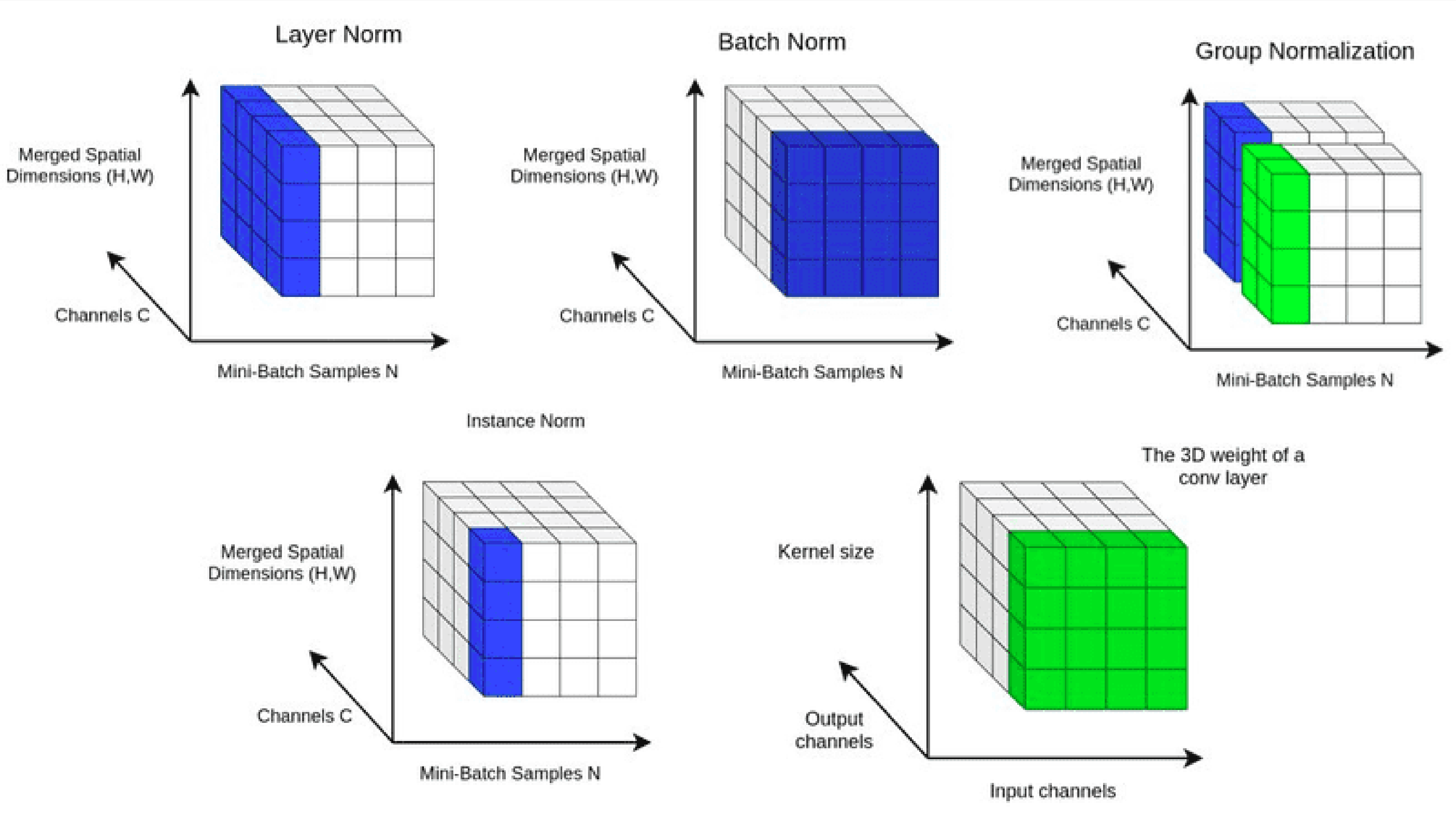
$$\boldsymbol{\mu}^1 \quad \boldsymbol{\mu}^2 \quad \boldsymbol{\mu}^3 \quad \ldots\ldots \quad \boldsymbol{\mu}^t$$

$$\bar{\boldsymbol{\mu}} \leftarrow p\bar{\boldsymbol{\mu}} + (1-p)\boldsymbol{\mu}^t$$

Deep Learning Training Tips

Overfitting

Other normalization



Reference

Deep Learning Training Tips

Overfitting

Other normalization

- Batch Renormalization
  - https://arxiv.org/abs/1702.03275
- Layer Normalization
  - https://arxiv.org/abs/1607.06450
- Instance Normalization
  - https://arxiv.org/abs/1607.08022
- Group Normalization
  - https://arxiv.org/abs/1803.08494
- Weight Normalization
  - https://arxiv.org/abs/1602.07868
- Spectrum Normalization
  - https://arxiv.org/abs/1705.10941

Deep Learning Training Tips
Overfitting

Data Augmentation

Reference

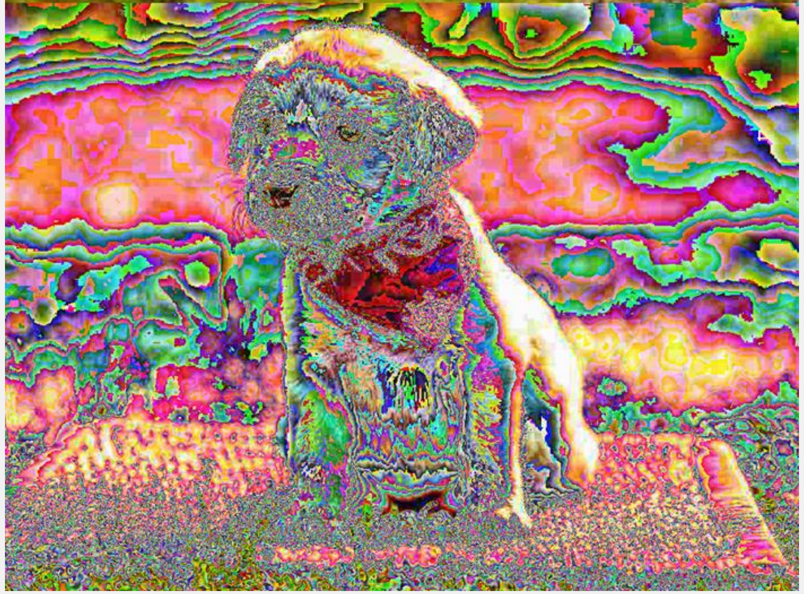
Deep Learning Training Tips

Overfitting

Pytorch Transform

```
mean = [0.5, 0.5, 0.5]
std = [0.1, 0.1, 0.1]
transform = transforms.Compose([
            transforms.ToTensor(),
            transforms.Normalize(mean, std),
            transforms.ToPILImage()
])

img_pil_normal = transform(img_pil)
img_pil_normal
```

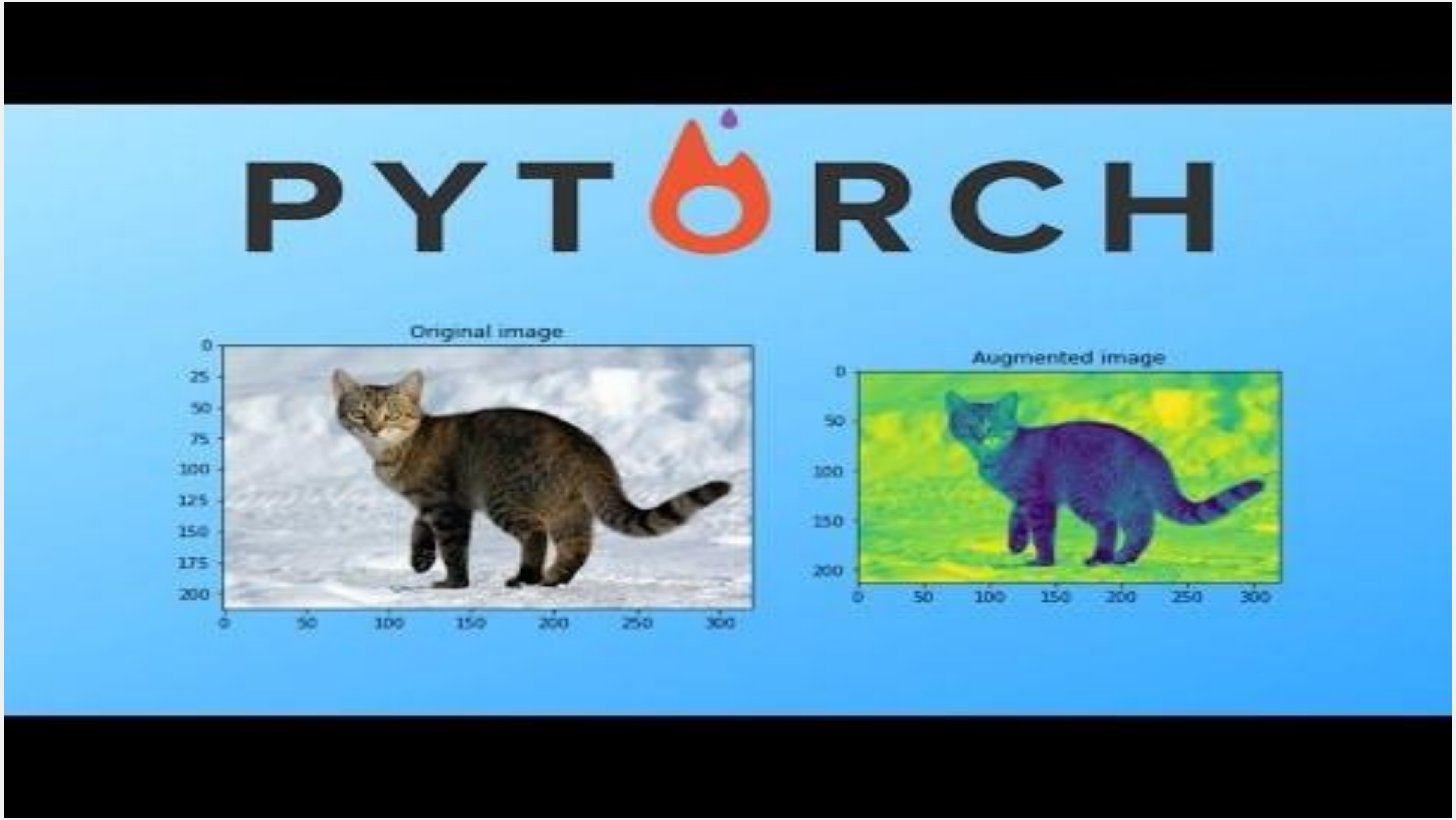


Reference

Deep Learning Training Tips

Overfitting

Pytorch Transform



[Reference](#) 68

Albumentations

```
train_transform = A.Compose(
    [
        A.SmallestMaxSize(max_size=160),
        A.ShiftScaleRotate(shift_limit=0.05, scale_limit=0.05, rotate_limit=15, p=0.5),
        A.RandomCrop(height=128, width=128),
        A.RGBShift(r_shift_limit=15, g_shift_limit=15, b_shift_limit=15, p=0.5),
        A.RandomBrightnessContrast(p=0.5),
        A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
        ToTensorV2(),
    ]
)
```

[Reference](#)

Deep Learning Training Tips

Overfitting

Albumentations

[Reference](#)