Artificial Intelligence: Optimization in Deep Learning

Some Slides from: Goodfellow et al., Deep Learning

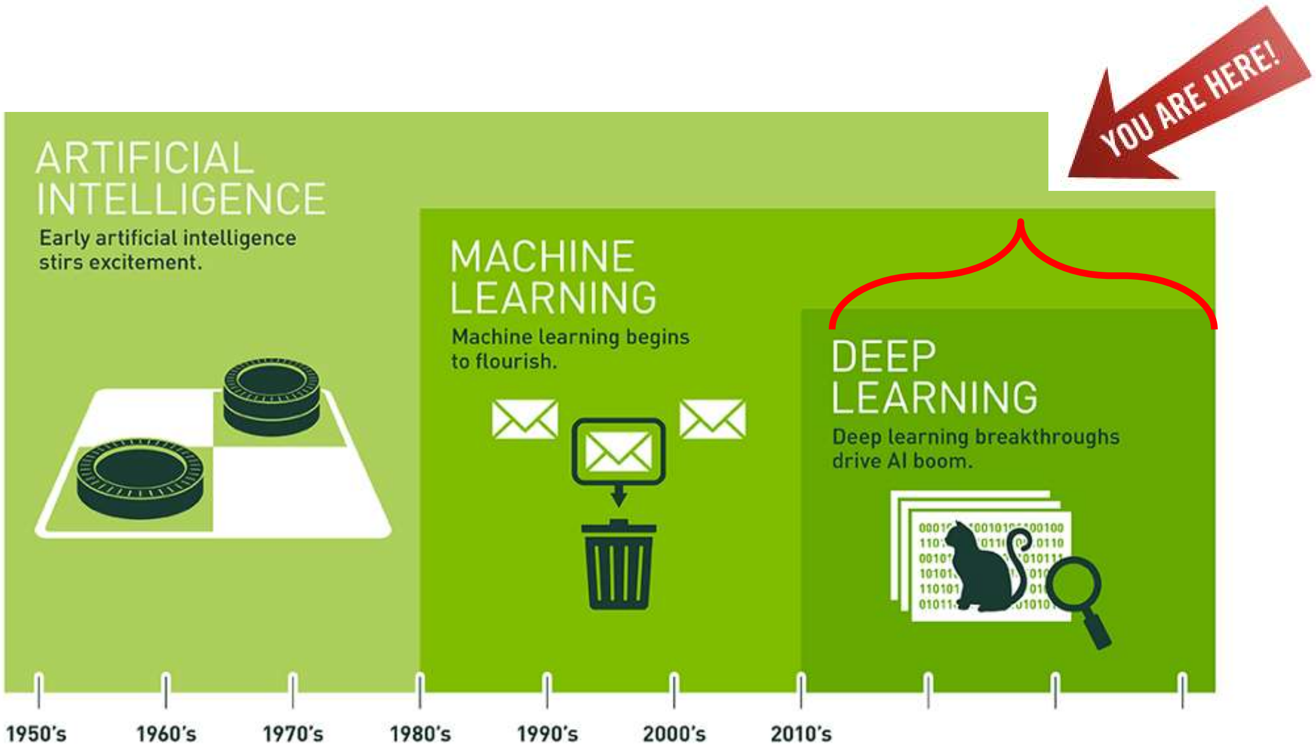
Slides are courtesy of Prof. Mahdi Hosseini.

Today

1. Feedforward in Deep Learning

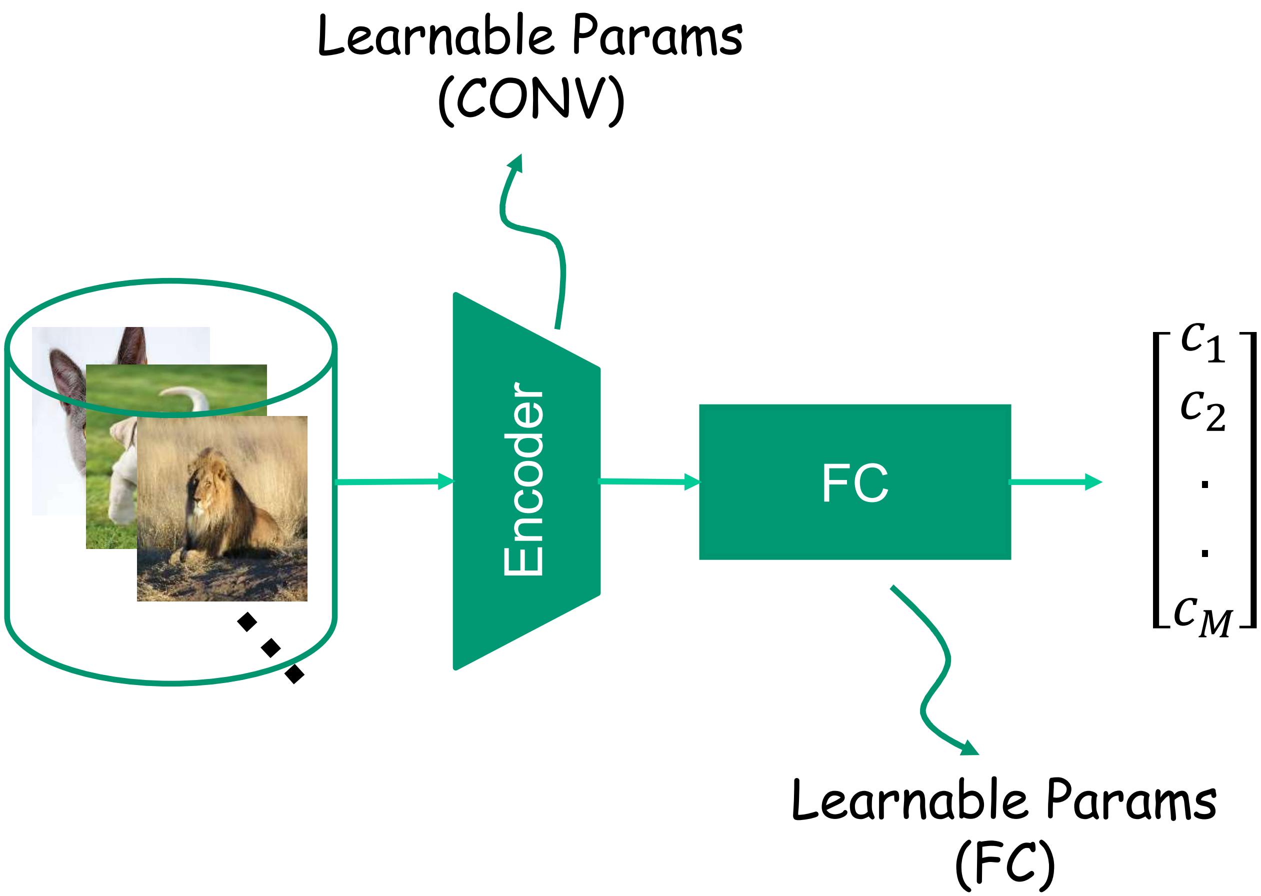
2. Backpropagation in Deep Learning
   - Gradient Descent
   - Stochastic Gradient Decent (SGD)
   - Momentum SGD
   - RMSProp
   - ADAM
3. Scheduled Learning
4. Hyper-Parameter (HP) Tuning

History of AI



https://www.linkedin.com/pulse/goedels-incompleteness-theorem-emergence-ai-eberhard-schoeneburg/

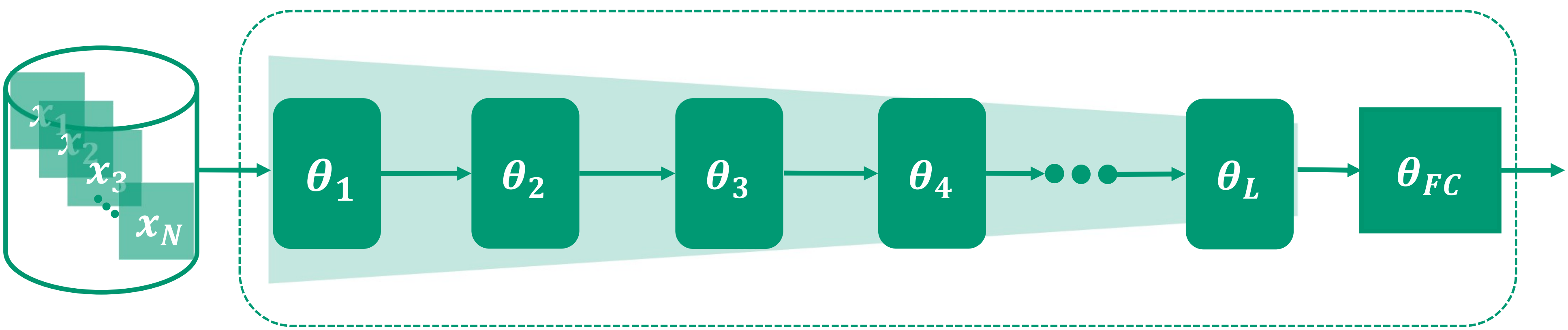
Feed-Forward in Deep Learning



Feed-Forward in Deep Learning

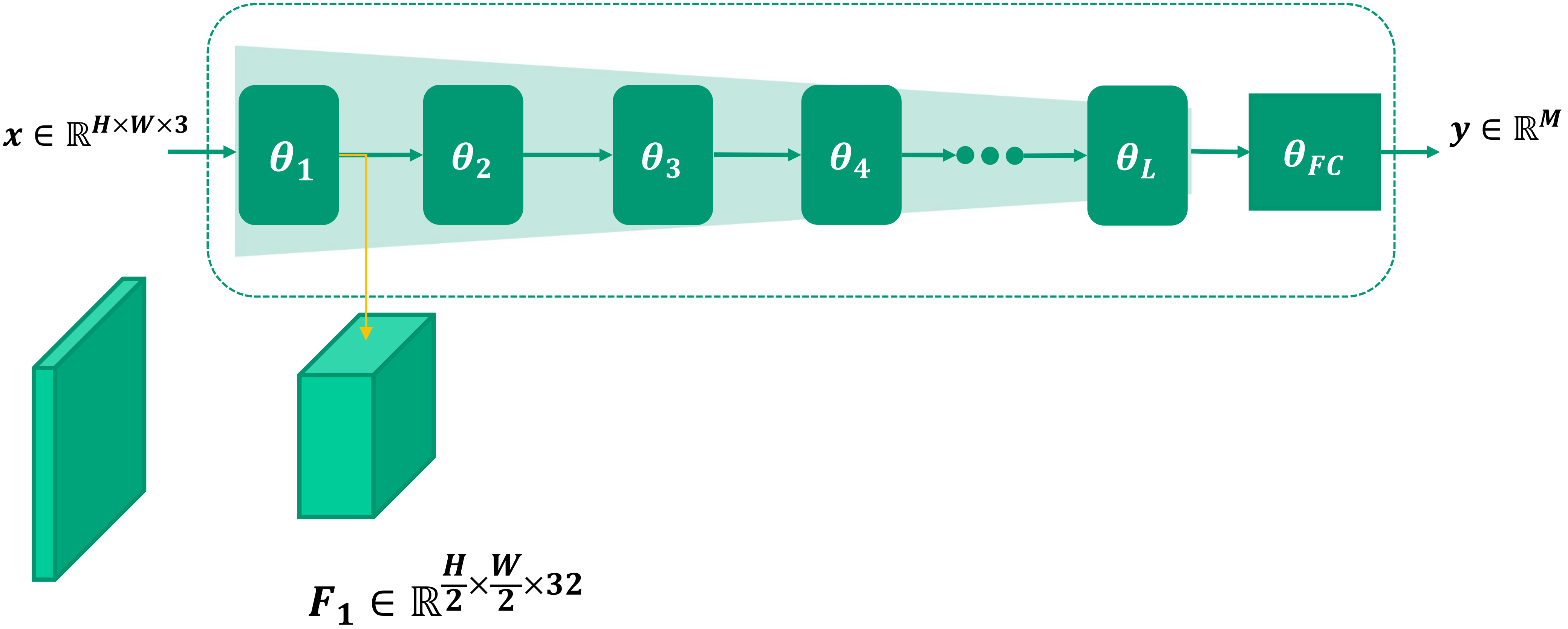
N – Number of images for training
M – Number of Class Images
c – Confidence prediction score
$\boldsymbol{\theta}$ – Learnable parameter
$\boldsymbol{x}$ – Input Image
$\boldsymbol{y}$ – Prediction label
$\boldsymbol{y^{GT}}$ – Ground-truth label

$$y = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_M \end{bmatrix}, e.g.\ y_{cat} = \begin{bmatrix} 0.11 \\ 0.78 \\ 0.06 \\ \cdot \\ 0.23 \end{bmatrix}, y_{cat}^{GT} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \cdot \\ 0 \end{bmatrix}$$



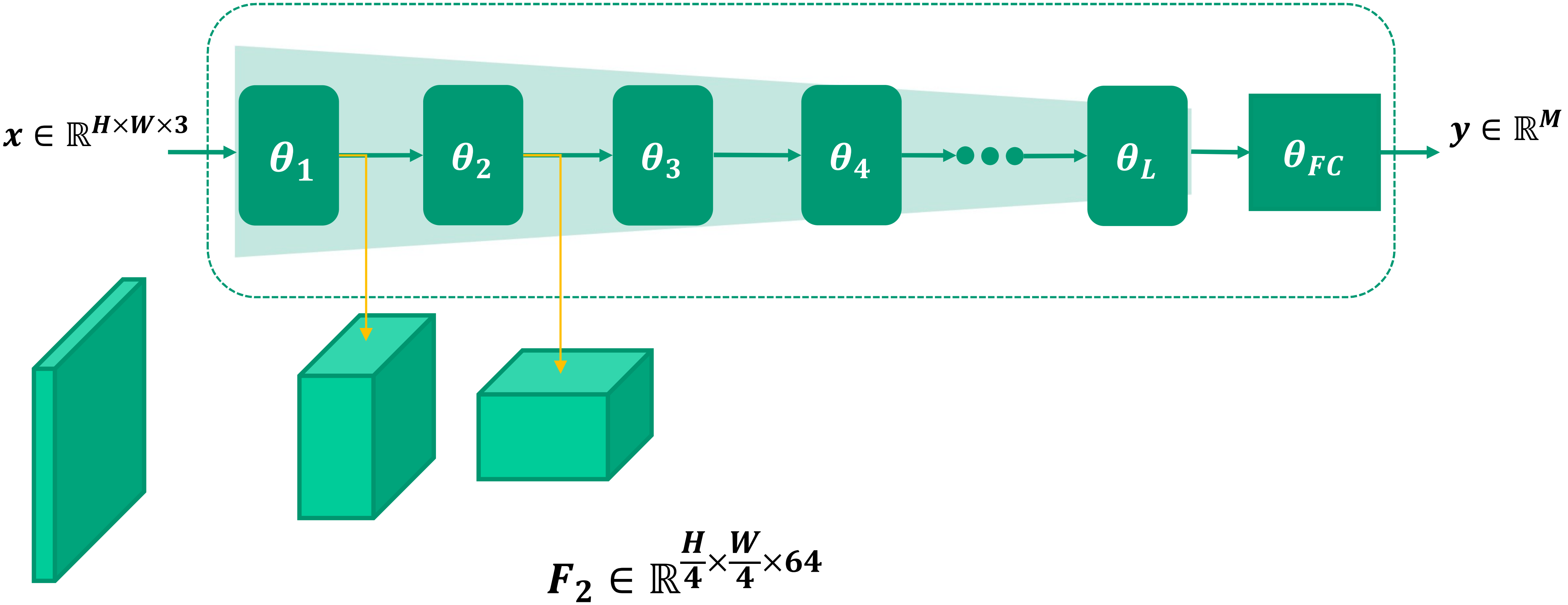
Feed-Forward in Deep Learning

N - N



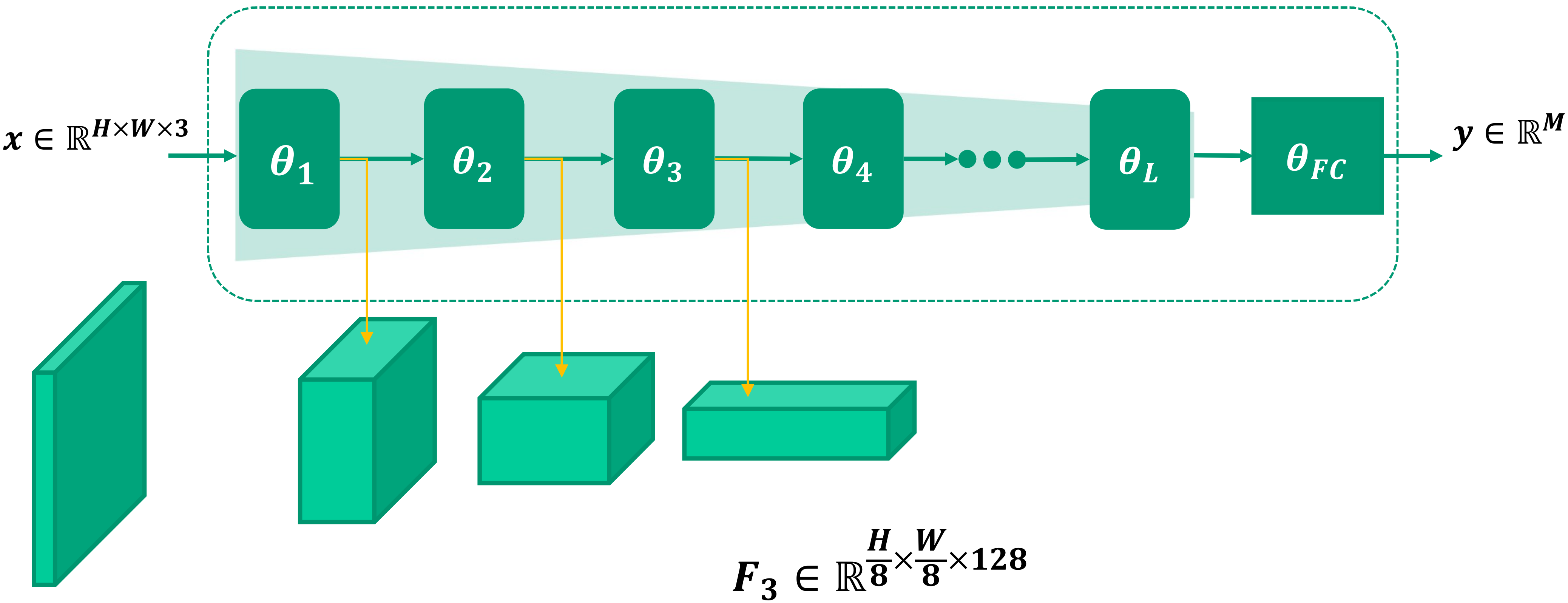
Feed-Forward in Deep Learning

N - N



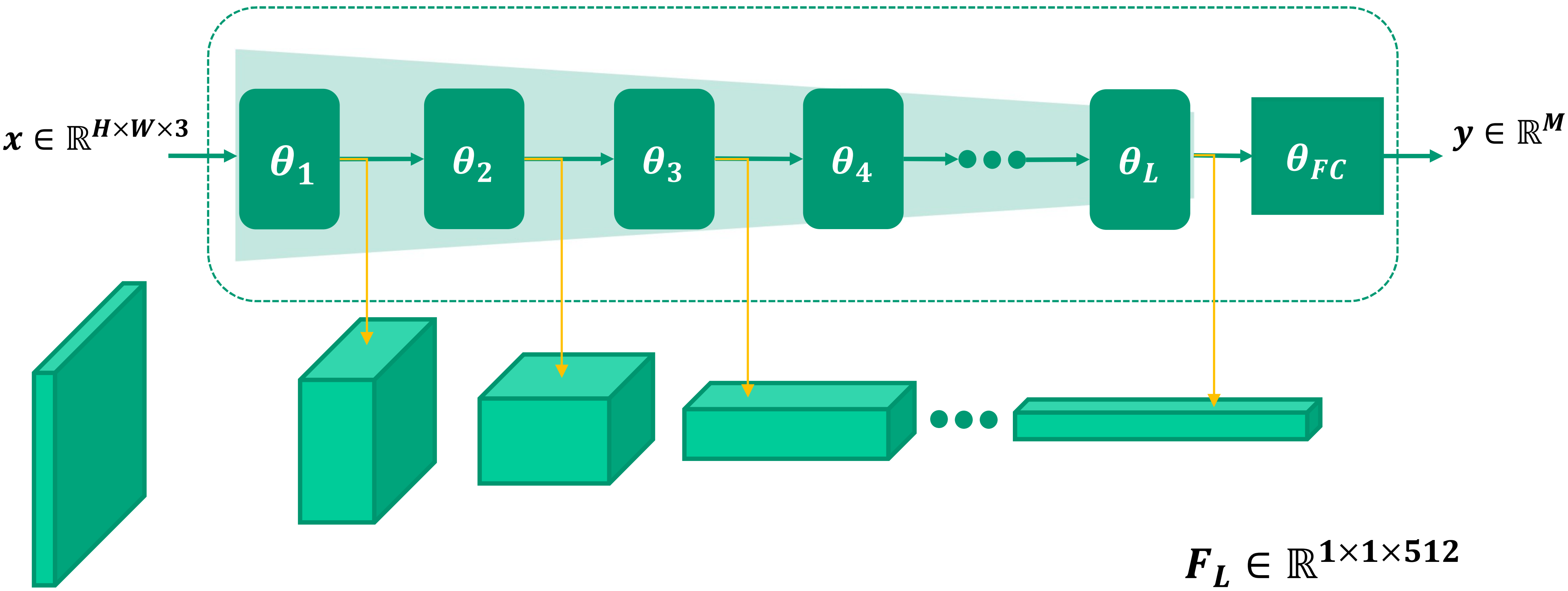
Feed-Forward in Deep Learning

N - N

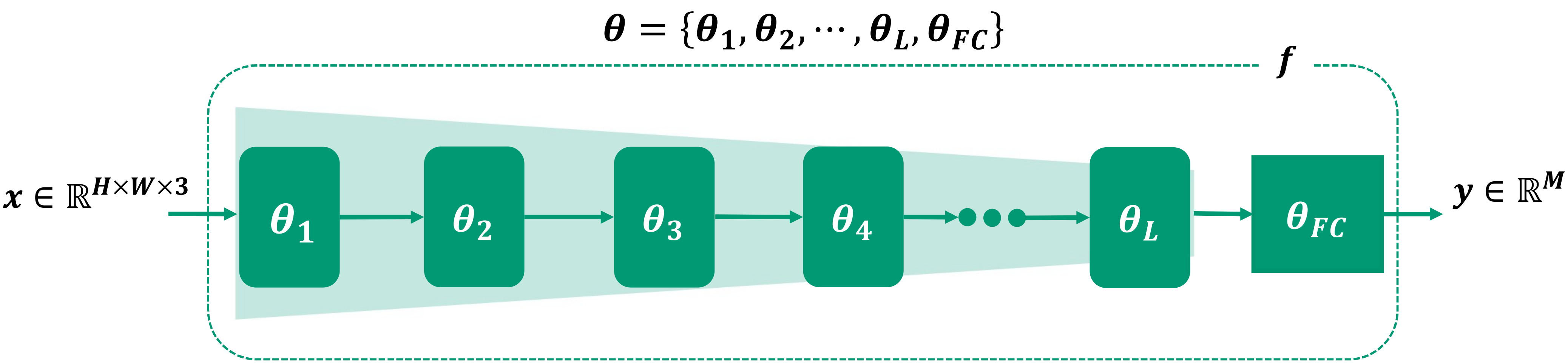


Feed-Forward in Deep Learning

N - N



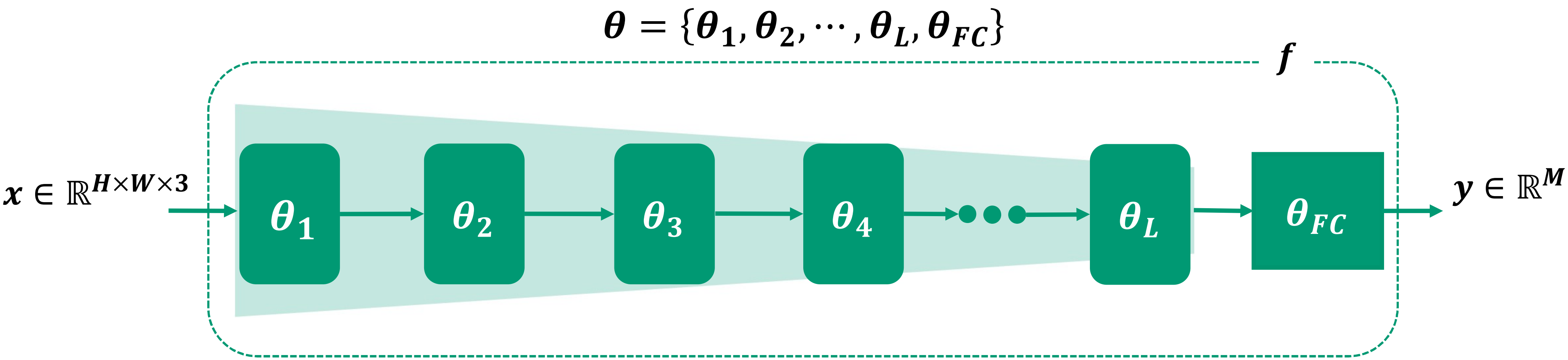
Feed-Forward in Deep Learning



The output prediction label is generated by a function 'f' applied on input image 'x' processed by learnable parameters

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$$

Back-Propagation in Deep Learning



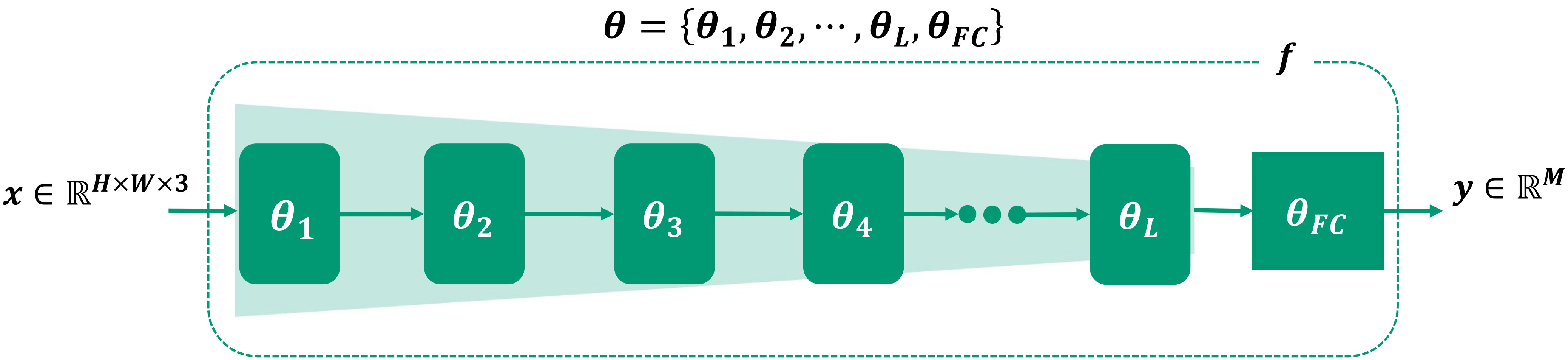
- For each input image $\boldsymbol{x}_i$ there is a corresponding ground-truth label $\boldsymbol{y}^{GT}$ which should be matched with output prediction label $\boldsymbol{y}$

$$\boldsymbol{\epsilon} = \boldsymbol{L}(\boldsymbol{y}, \boldsymbol{y}^{GT})$$

L: Loss-Function

Ideally Speaking: $\boldsymbol{\epsilon} \to \mathbf{0}$

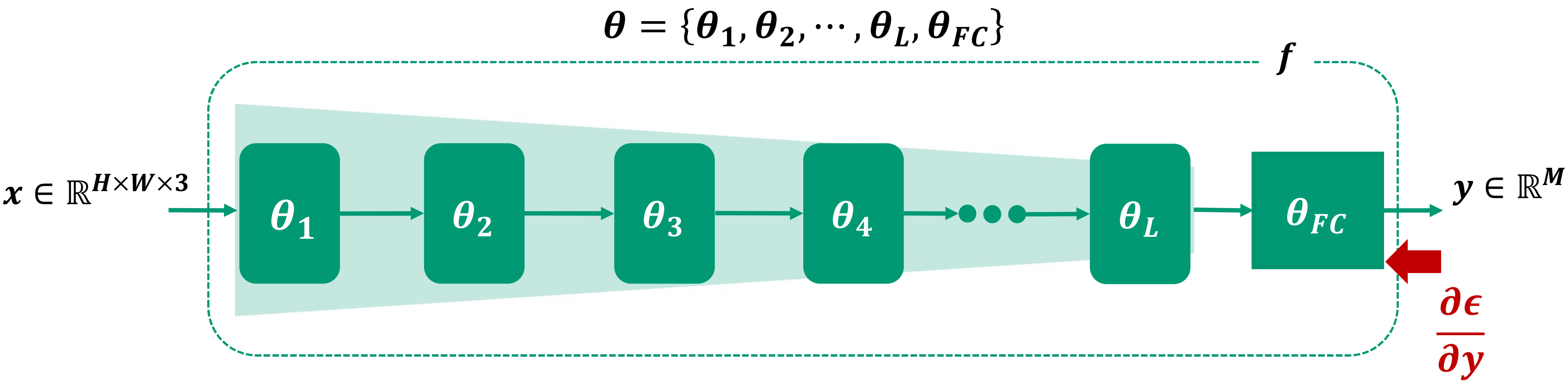
Back-Propagation in Deep Learning



- Calculate the gradient of loss prediction in terms of prediction label

$$\frac{\partial \epsilon}{\partial y} = \frac{\partial L\left(y, y^{GT}\right)}{\partial y}$$

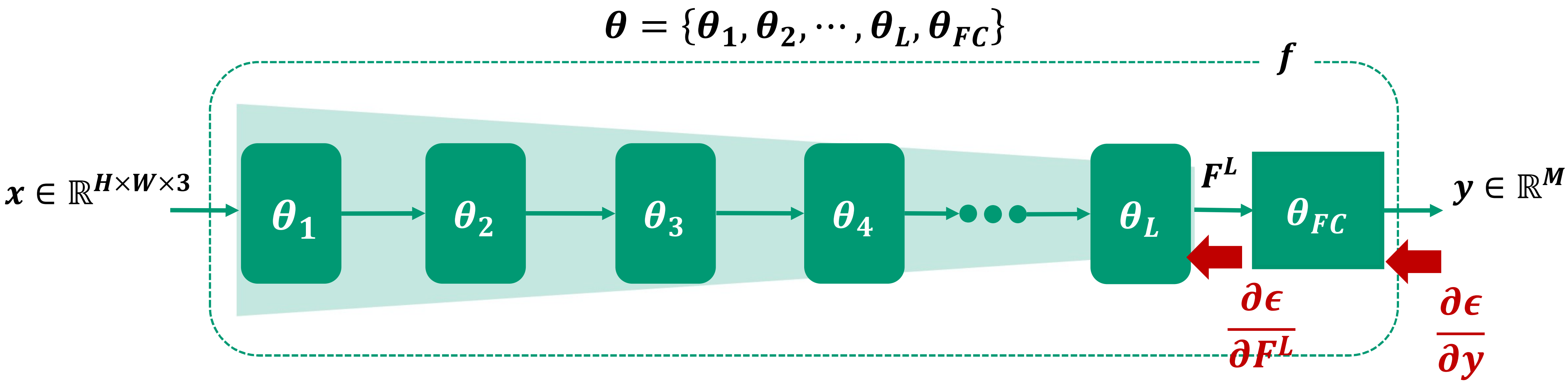
Back-Propagation in Deep Learning



- Calculate the gradient of loss prediction in terms of prediction label

$$\frac{\partial \epsilon}{\partial \boldsymbol{y}} = \frac{\partial L\left(\boldsymbol{y}, \boldsymbol{y}^{GT}\right)}{\partial \boldsymbol{y}}$$

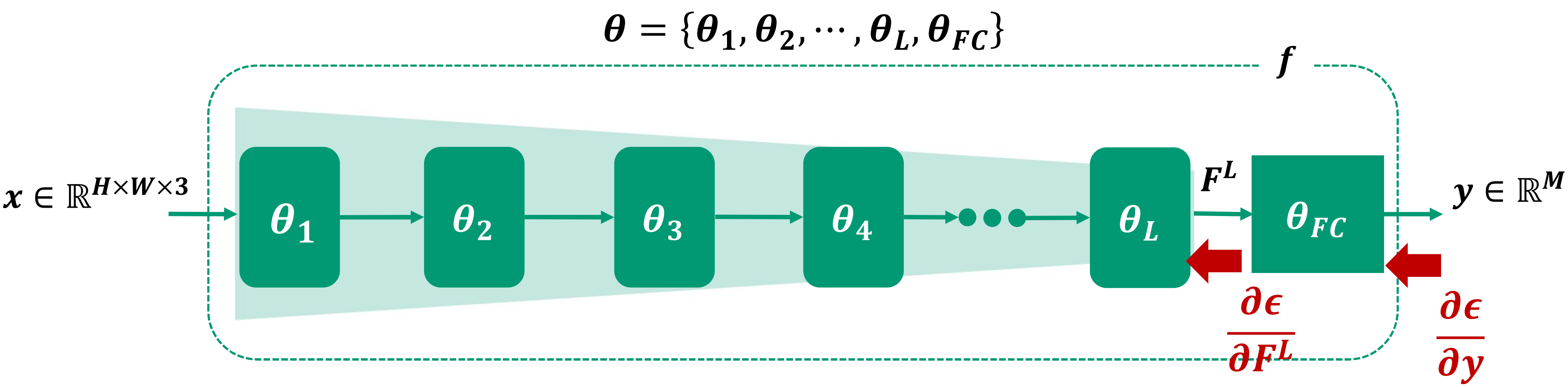
Back-Propagation in Deep Learning



- Back-propagate the gradient of loss-function into inner layers to calculate the gradient of loss-function with respect to the learnable parameter of that particular layer

$$\frac{\partial \epsilon}{\partial F^L} = \frac{\partial \epsilon}{\partial y} \cdot \frac{\partial y}{\partial F^L}$$

Back-Propagation in Deep Learning



- Back-propagate the gradient of loss-function into inner layers to calculate the gradient of loss-function with respect to the learnable parameter of that particular layer

$$\frac{\partial \epsilon}{\partial F^L} = \frac{\partial \epsilon}{\partial y} \cdot \frac{\partial y}{\partial F^L}$$

- We can now update parameter weights using gradient-descent method

$$\underbrace{\theta_L^{k+1}}_{\text{Updated Parameter}} \leftarrow \underbrace{\theta_L^{k}}_{\text{Previous Parameter}} - \underbrace{\alpha}_{\text{Learning Rate}} \cdot \underbrace{\frac{\partial \epsilon}{\partial \theta^L}}_{\text{Error-Gradient}}$$

Back-Propagation in Deep Learning

- Updating on a single image sample introduces noisy gradient direction and we can easily get stuck at local minima
- Select a mini-batch samples (from randomly shuffled data) and average the gradients for updating
- aka we update not for every image but batch-of-images

$$\{x_1, x_2, \cdots, x_B\} \Longleftrightarrow \{\frac{\partial \epsilon}{\partial F_1^l}, \frac{\partial \epsilon}{\partial F_2^l}, \cdots, \frac{\partial \epsilon}{\partial F_B^l}\}$$

- Superimpose all batch gradients to step into average direction

$$\underbrace{\theta_l^{k+1}}_{\text{Updated Parameter}} \leftarrow \underbrace{\theta_l^{k+1}}_{\text{Previous Parameter}} - \underbrace{\rho}_{\text{Learning Rate}} \cdot \underbrace{\frac{1}{B}\sum_{i=1}^{B} \frac{\partial \epsilon}{\partial F_i^l}}_{\text{Stochastic Error-Gradient}}$$

Stochastic Gradient Descent (SGD)

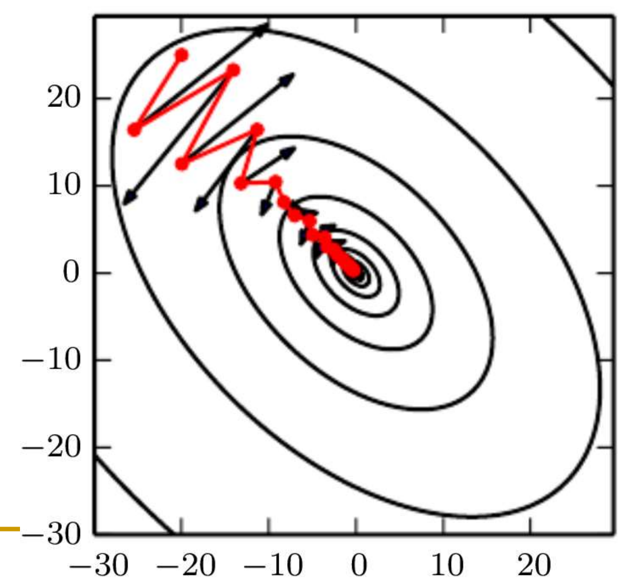
Algorithm 8.1 Stochastic gradient descent (SGD) update

Require: Learning rate schedule $\boldsymbol{\rho}_1\ \boldsymbol{\rho}_2, \ldots$
Require: Initial parameter $\boldsymbol{\theta}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample a minibatch of $\boldsymbol{B}$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\boldsymbol{B})}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{\boldsymbol{B}} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\rho}_k \hat{\boldsymbol{g}}$
    $k \leftarrow k + 1$
  end while

SGD with Momentum

- Learning with SGD can be sometimes slow
- Momentum approach we can be used to accelerate learning in the face of exploring local minima of loss function
  - High curvature
  - Small but consistent gradient
  - Noisy gradient
- Momentum approach accumulates an exponentially decaying moving average of past gradients and continues to move in their direction

The contour lines depicts a quadratic loss function with poor Hessian Matrix. The red path cutting across the contour indicates the path followed by momentum learning rule to minimize the loss function



SGD with Momentum

How to formulate it?

- Introduce a hyper-parameter (i.e. momentum) $\alpha \in [0,1)$
- α determines how quickly the contributions of previous gradients exponentially decay
- The update rule is given by

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \boldsymbol{\rho} \nabla_{\boldsymbol{\theta}} \left(\frac{1}{\boldsymbol{B}} \sum_{i=1}^{\boldsymbol{B}} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}.$$

Momentum-SGD (MSGD)

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate $\boldsymbol{\rho}$, momentum parameter α

Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$

  while stopping criterion not met **do**

    Sample a minibatch of $\boldsymbol{B}$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\boldsymbol{B})}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

    Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{\boldsymbol{B}} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$.

    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \boldsymbol{\rho} \boldsymbol{g}$.

    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$.

  end while

Today

1. Feedforward in Deep Learning
2. Backpropagation in Deep Learning
    - Gradient Descent
    - Stochastic Gradient Decent (SGD)
    - Momentum SGD
    - RMSProp
    - ADAM
3. Scheduled Learning
4. Hyper-Parameter (HP) Tuning

Where to read from rest of topics?

Please refer to Reading Material as well as Class Discussions for the rest of topics.

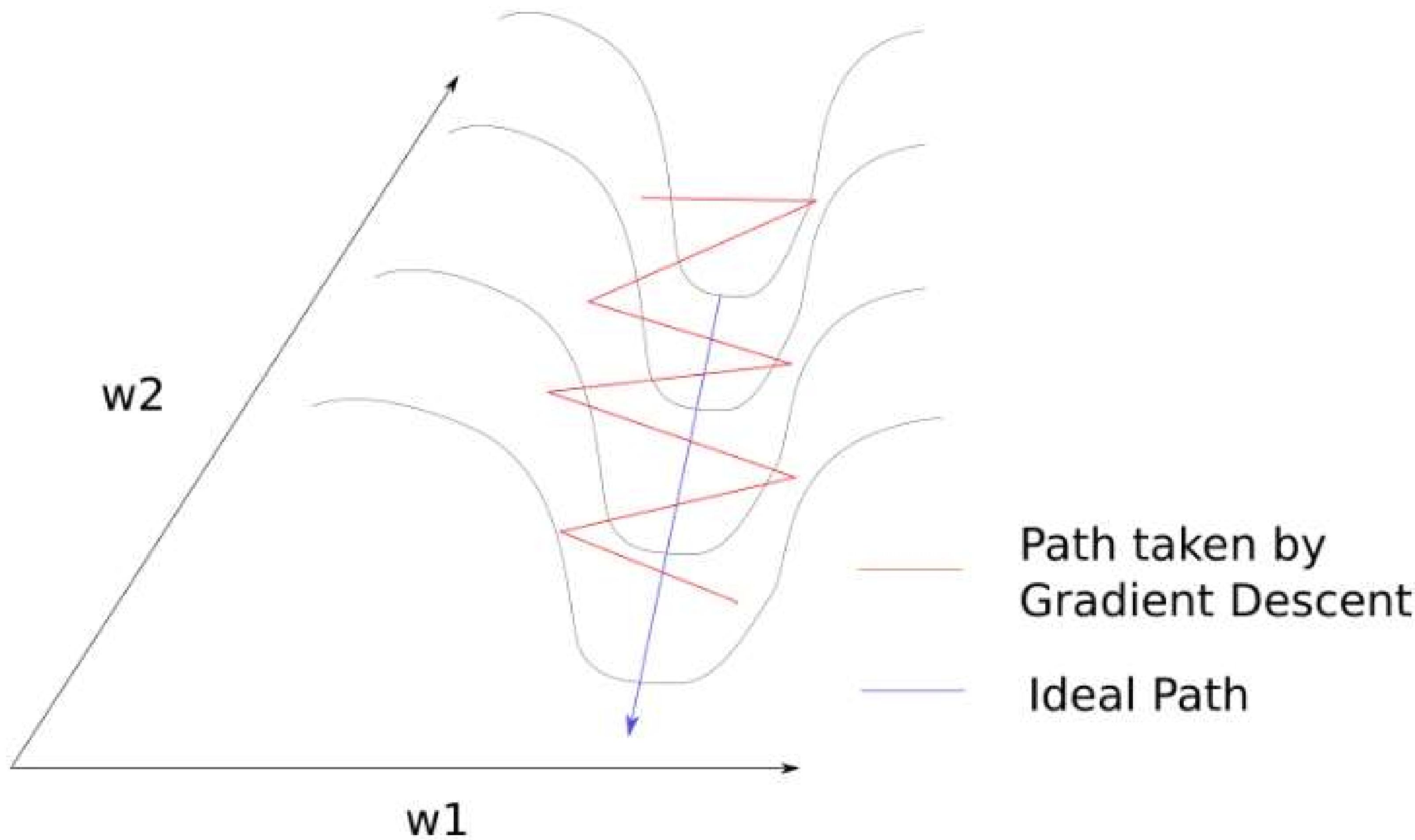
Optimization in Deep Learning Continued!

All Slides from https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

By Ayoosh Kathuria
Intro to optimization in deep learning: Momentum, RMSProp and Adam
https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

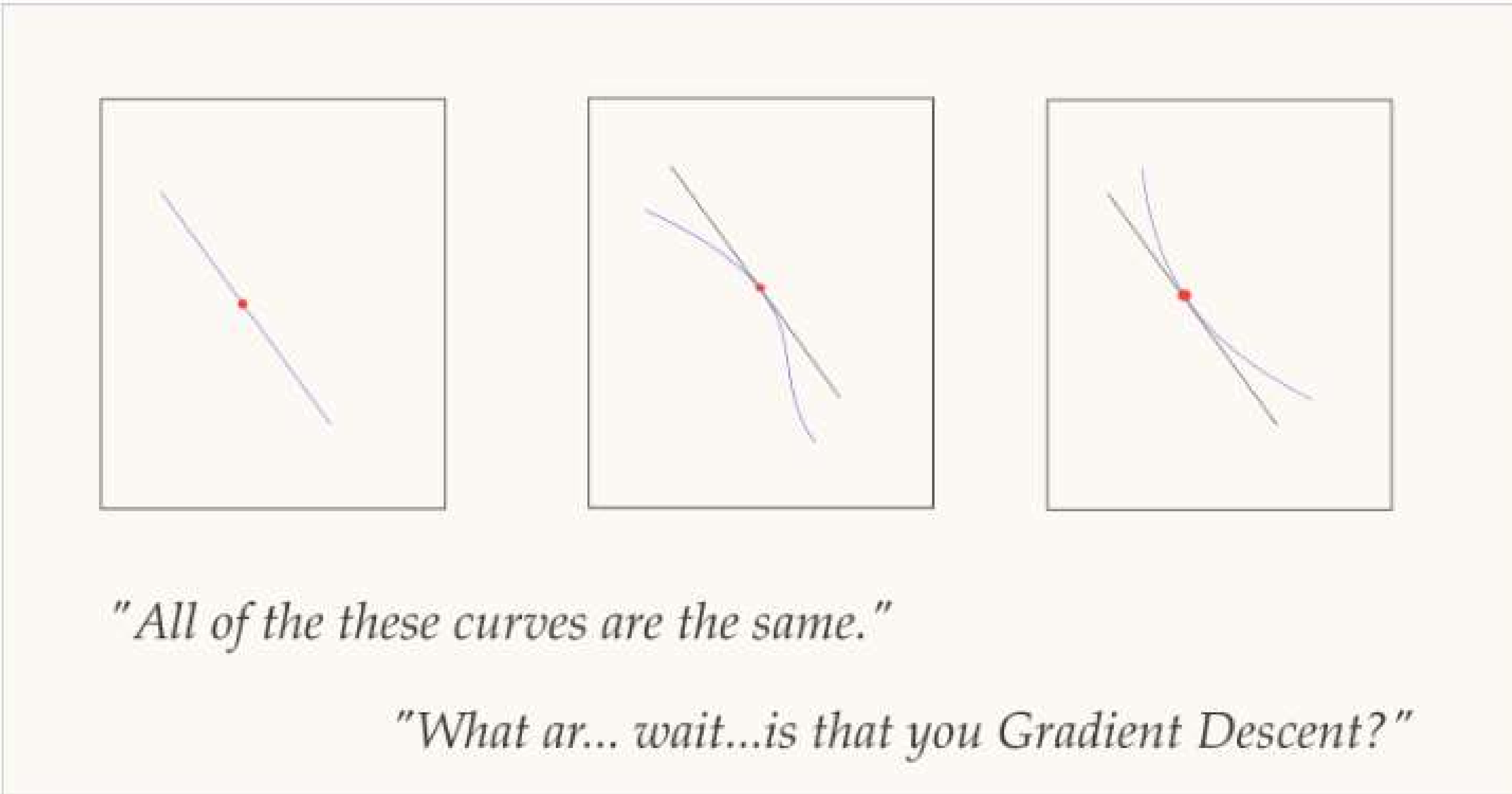
There was no notice prohibiting reproduction on 11 June, 2023.

Pathological Curvature



https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam

First-order optimization (gradient)



- No clue about the curvature of the loss function
  - Second-order optimization (Hessian)
  - Take into account previous gradients

https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam

Momentum

Repeat Until Convergence {

$$\nu_j \leftarrow \boxed{\eta * \nu_j} - \alpha * \nabla_w \sum_{1}^{m} L_m(w)$$

$$\omega_j \leftarrow \nu_j + \omega_j$$

}

$$\underbrace{\eta}_{\text{Coefficient of Momentum}} * \underbrace{\nu_j}_{\text{Retained Gradient}}$$

$$\nu_1 = -G_1$$

$$\nu_2 = -0.9 * G_1 - G_2$$

$$\nu_3 = -0.9 * (0.9 * G_1 - G_2) - G3 = -0.81 * (G_1) - (0.9) * G_2 - G_3$$

The coefficient of momentum is usually initialized at 0.5 and gradually increased to 0.9 over multiple epochs.

RMSProp (Root Mean Square Propogation)

$$For\ each\ Parameter\ w^j$$

$$(j\ subscript\ dropped\ for\ clarity)$$

$$\nu_t = \rho\nu_{t-1} + (1-\rho) * g_t^2$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*

ν_t : *Exponential Average of squares of gradients*

g_t : *Gradient at time t along* ω^j

https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam

RMSProp (Cont'd)

- RMSProp also tries to mitigate the tries the oscillations.
- If average of w1 is larger than w2, the learning step for w1 would be lesser than that of w2, which helps to mitigate oscillations.
- Uses the moving average of the squared gradients to scale the learning rate for each parameter.
- Learning is adjusted separately for each parameter, so gradient *gt* here corresponds to the projection or component of the gradient along the direction represented by the parameter we are updating.
- The hyperparameter p is generally chosen to be around 0.9
- Epsilon is to ensure that we do not end up dividing by zero and is generally chosen to be 1e-10.

ADAM (Adaptive Moment Optimization)

$$For\ each\ Parameter\ w^j$$

$$(j\ subscript\ dropped\ for\ clarity)$$

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*

g_t : *Gradient at time t along* ω^j

ν_t : *Exponential Average of gradients along* ω_j

s_t : *Exponential Average of squares of gradients along* ω_j

β_1, β_2 : *Hyperparameters*

https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam

ADAM (Cont'd)

- ADAM combines the ideas of Momentum and RMSPROP
- Uses the moving average of the gradient and of the squared gradients to scale the learning rate for each parameter.
- Learning is adjusted separately for each parameter, so gradient *gt* here corresponds to the projection or component of the gradient along the direction represented by the parameter we are updating.
- The hyperparameter β1 is generally chosen to be around 0.9 and β2 to be around 0.99
- Epsilon is to ensure that we do not end up dividing by zero and is generally chosen to be 1e-10.

Other Hyper-Parameters and Tuning Options

- Mini-batch size
  - Number of epochs and mini-batch size
- Regularization techniques
  - Structural (number of neurons and hidden layers)
  - Dropout
  - Modified cost function
  - …
- Weight initialization
  - Xavier initialization
  - He initialization
  - …