Carnegie Mellon University
Heinz College

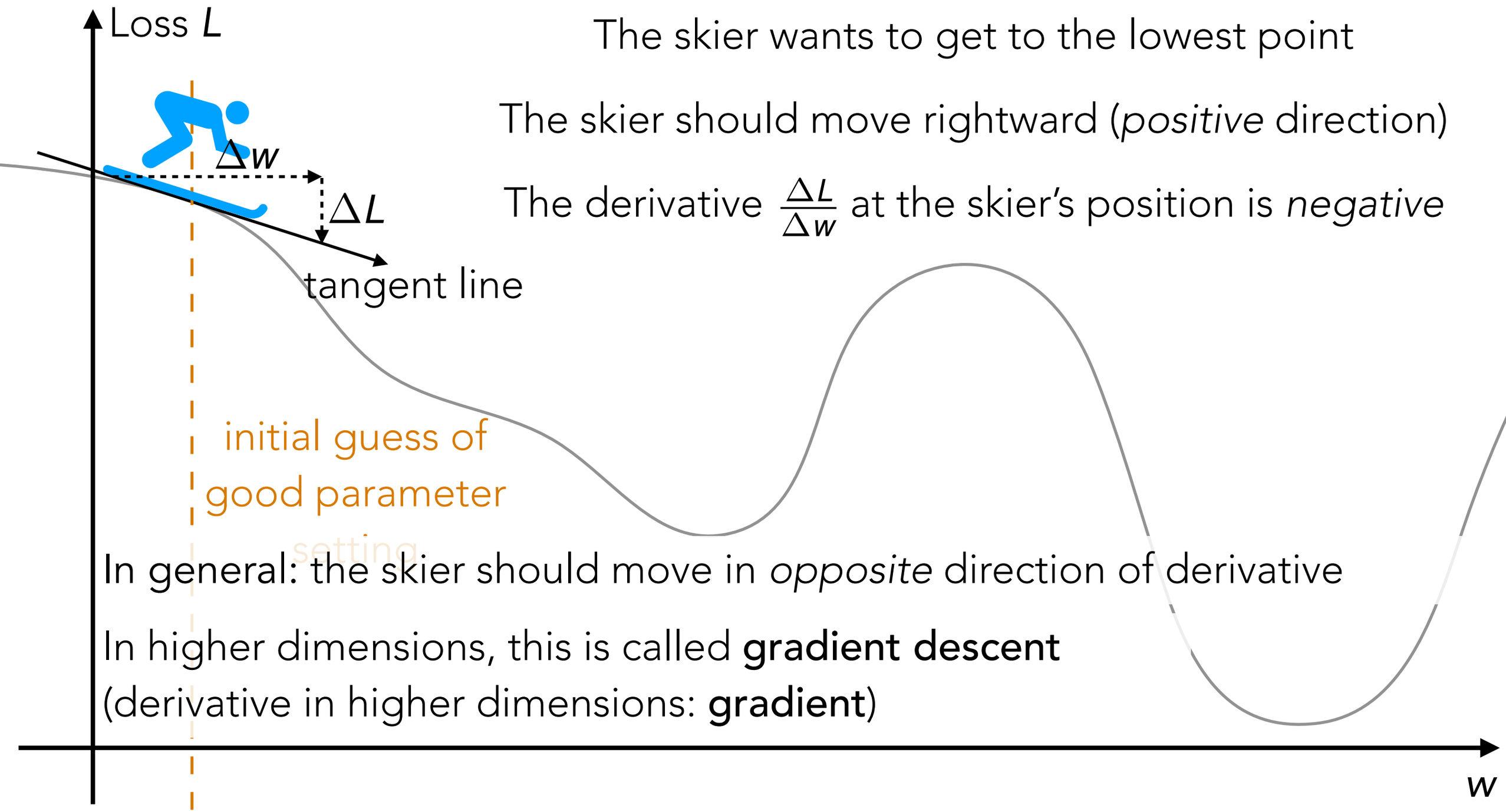
95-865 Unstructured Data Analytics

Some slides for part of recitation: How to train a deep net

Slides by George H. Chen

Training a Deep Net

Suppose the neural network has a single real number parameter *w*



The skier wants to get to the lowest point

The skier should move rightward (*positive* direction)

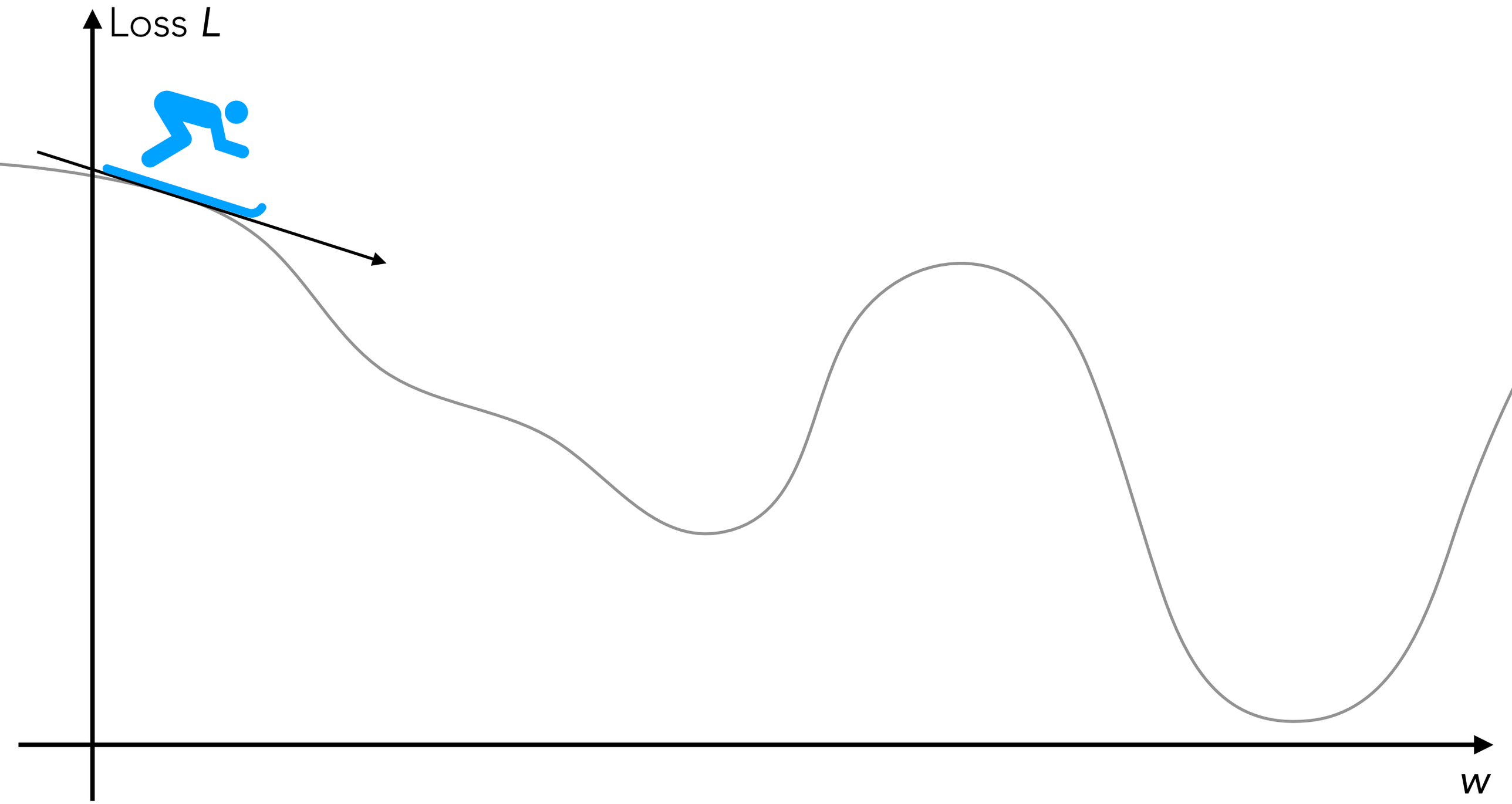
The derivative $\frac{\Delta L}{\Delta w}$ at the skier's position is *negative*

In general: the skier should move in *opposite* direction of derivative

In higher dimensions, this is called **gradient descent**
(derivative in higher dimensions: **gradient**)

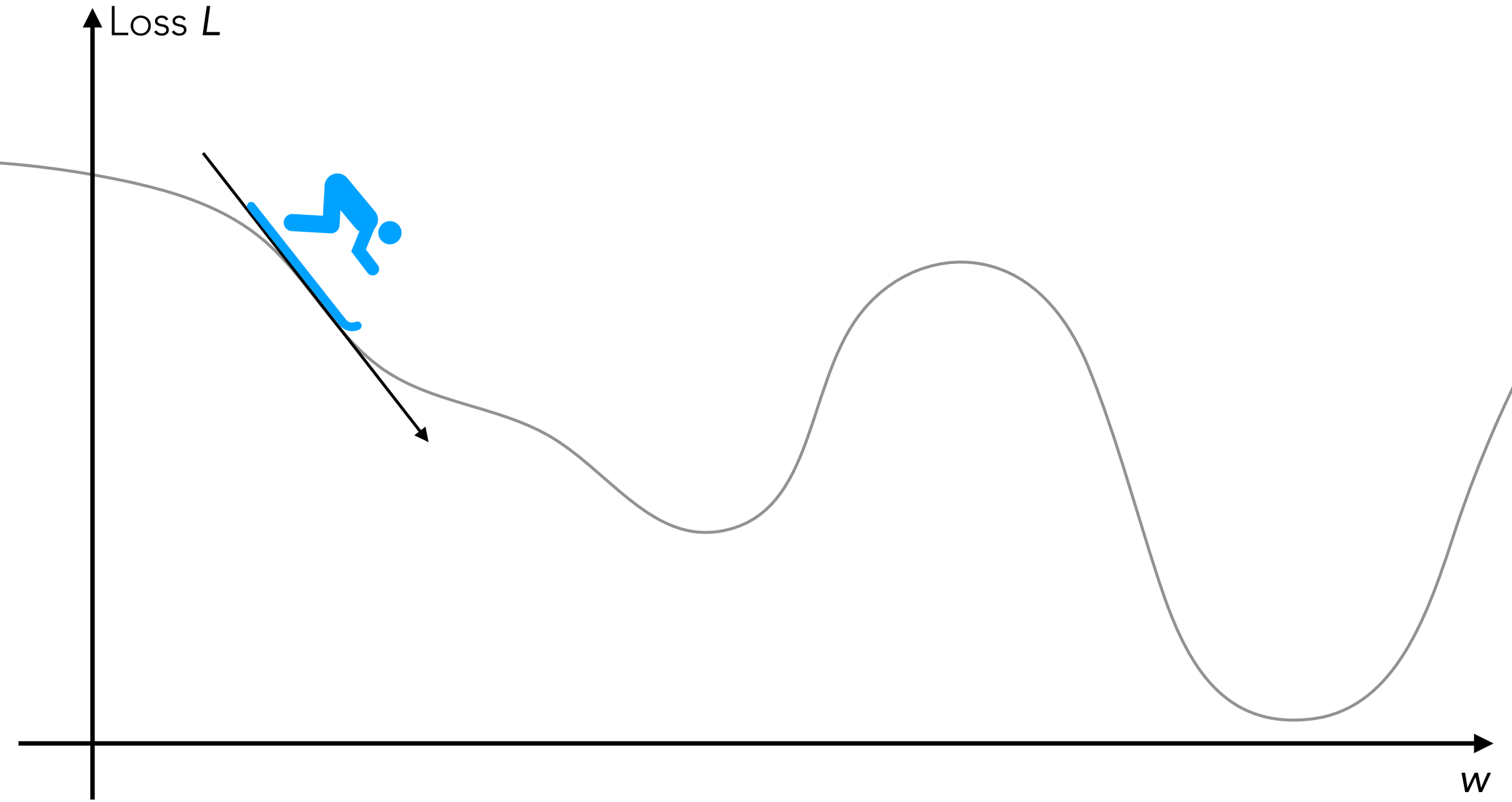
Training a Deep Net

Suppose the neural network has a single real number parameter *w*



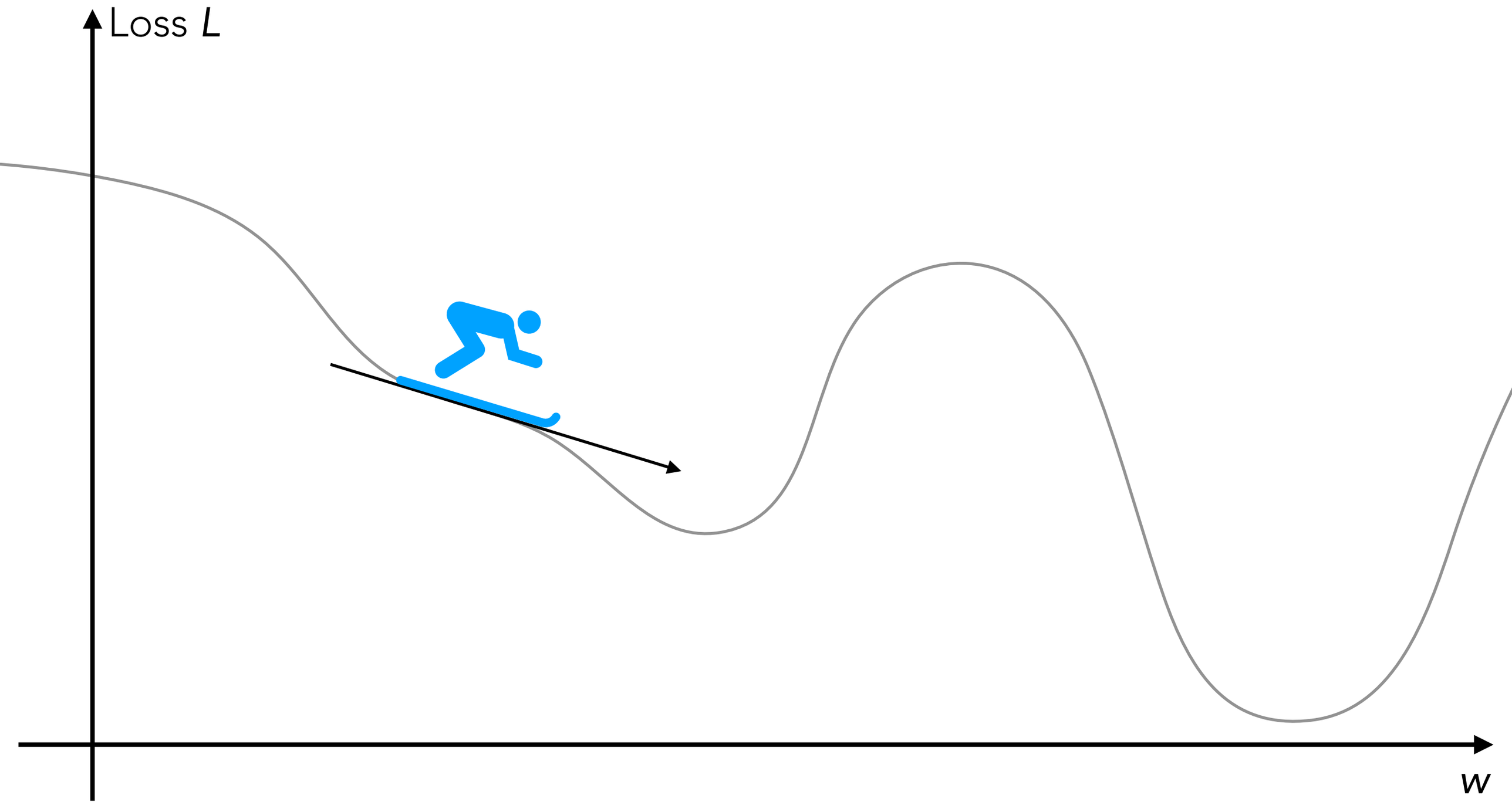
Training a Deep Net

Suppose the neural network has a single real number parameter *w*



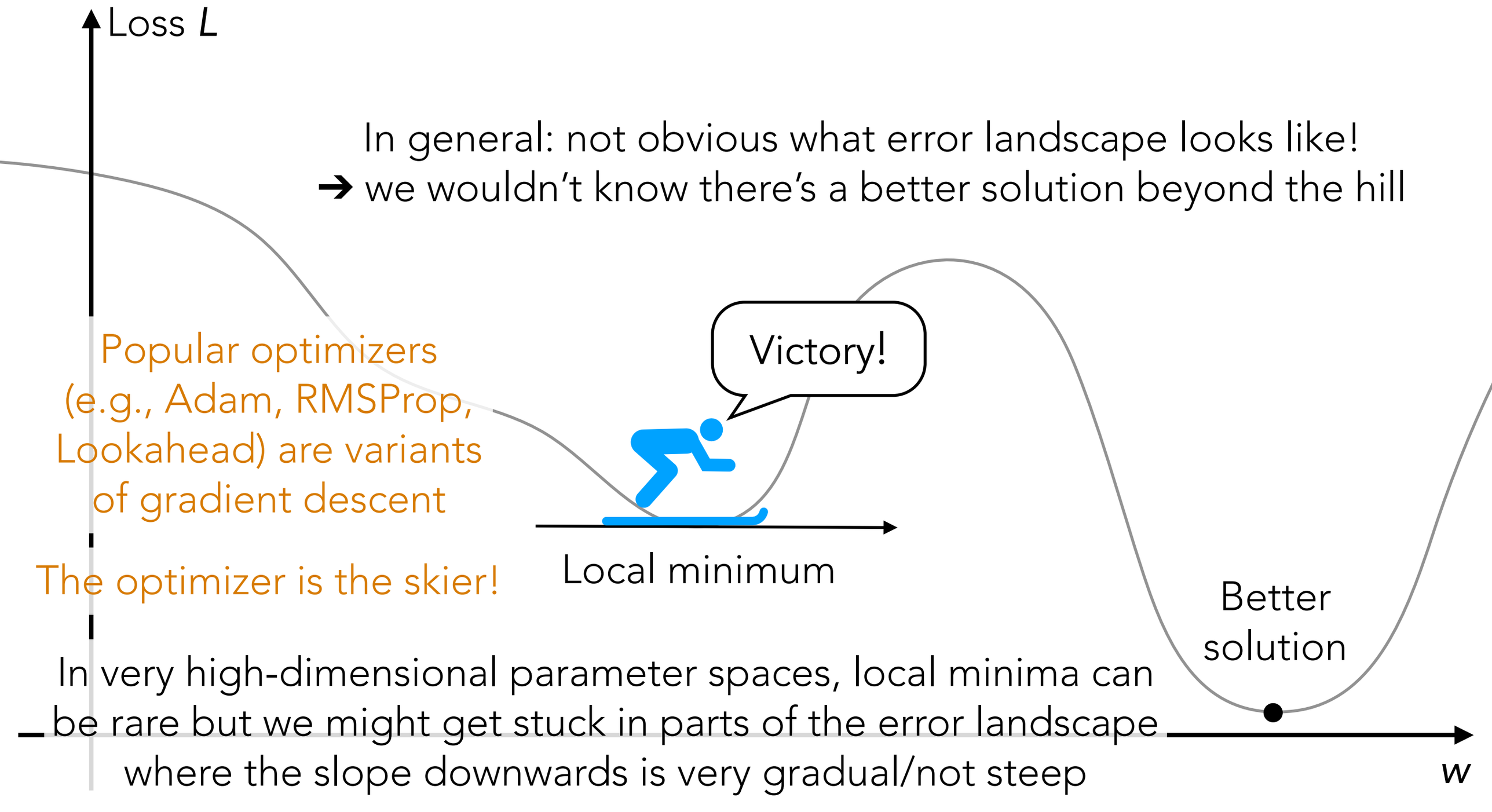
Training a Deep Net

Suppose the neural network has a single real number parameter *w*



Training a Deep Net

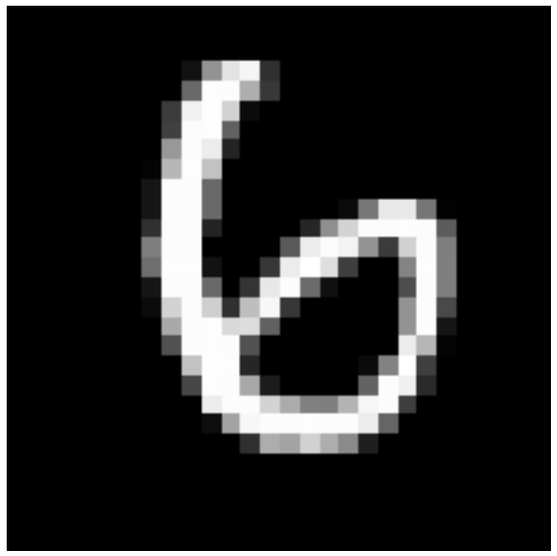
Suppose the neural network has a single real number parameter *w*



Handwritten Digit Recognition

Training label: 6

y_i



28x28 image

x_i

f_1

f_2

$f_1(x_i)$

$f_2(f_1(x_i))$

Loss

L

error

$L(f_2(f_1(x_i)), y_i)$

A neural net does function composition!

Overall loss:

$$\frac{1}{n}\sum_{i=1}^{n} L(f_2(f_1(x_i)), y_i)$$

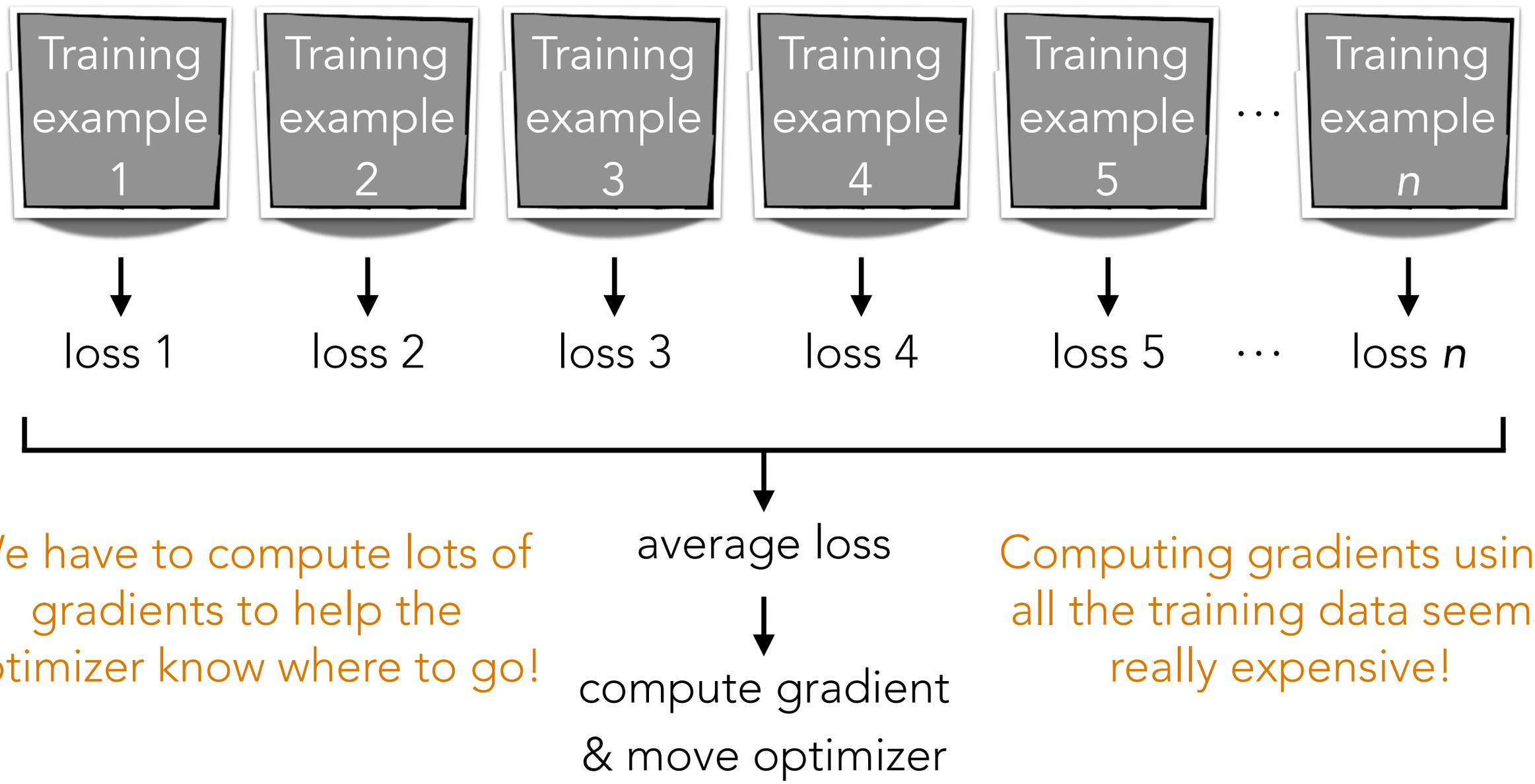
All parameters: θ

Gradient: $\dfrac{\partial \frac{1}{n}\sum_{i=1}^{n} L(f_2(f_1(x_i)), y_i)}{\partial \theta}$

Automatic differentiation is crucial in learning deep nets!

Careful derivative chain rule calculation: **back-propagation**

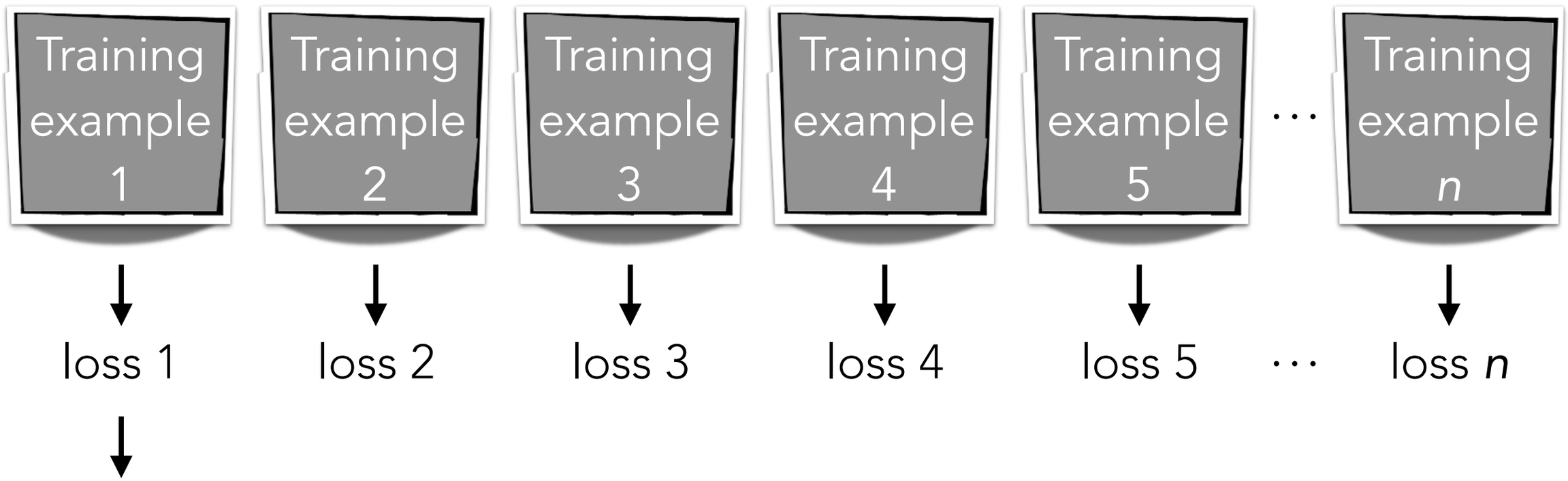
Gradient Descent



We have to compute lots of gradients to help the optimizer know where to go!

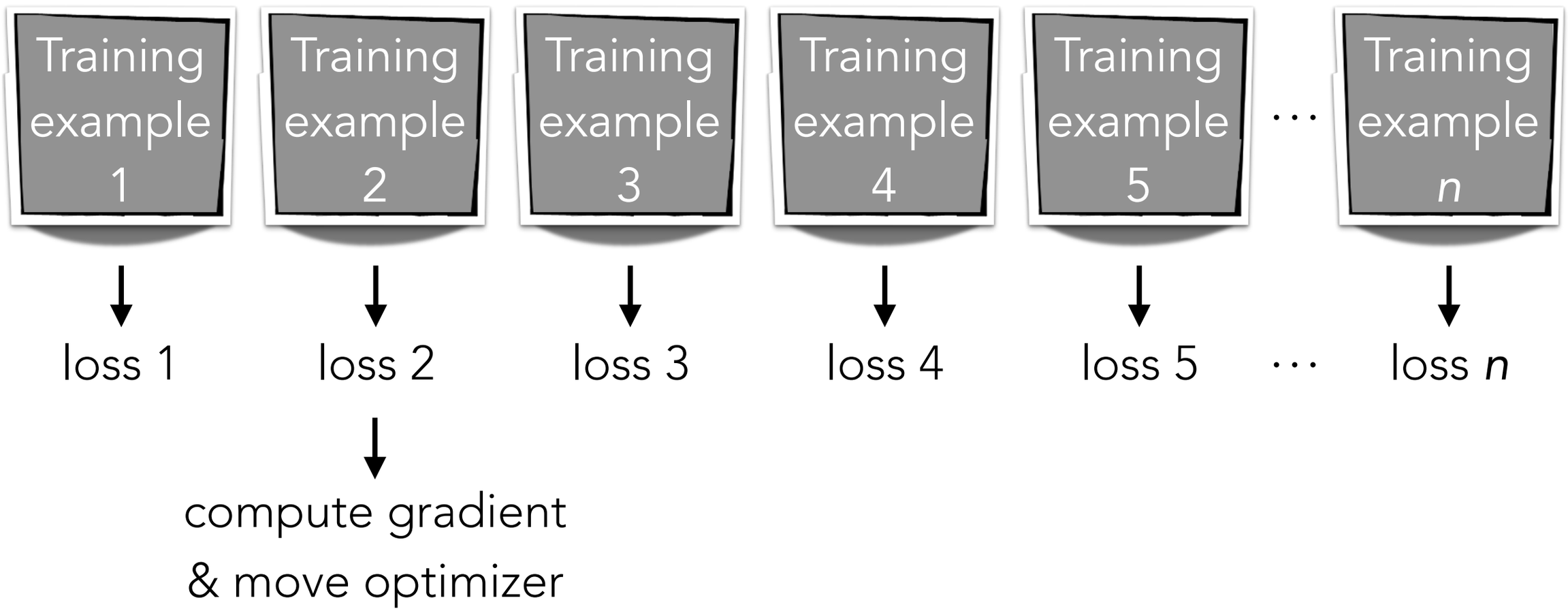
Computing gradients using all the training data seems really expensive!

Stochastic Gradient Descent (SGD)



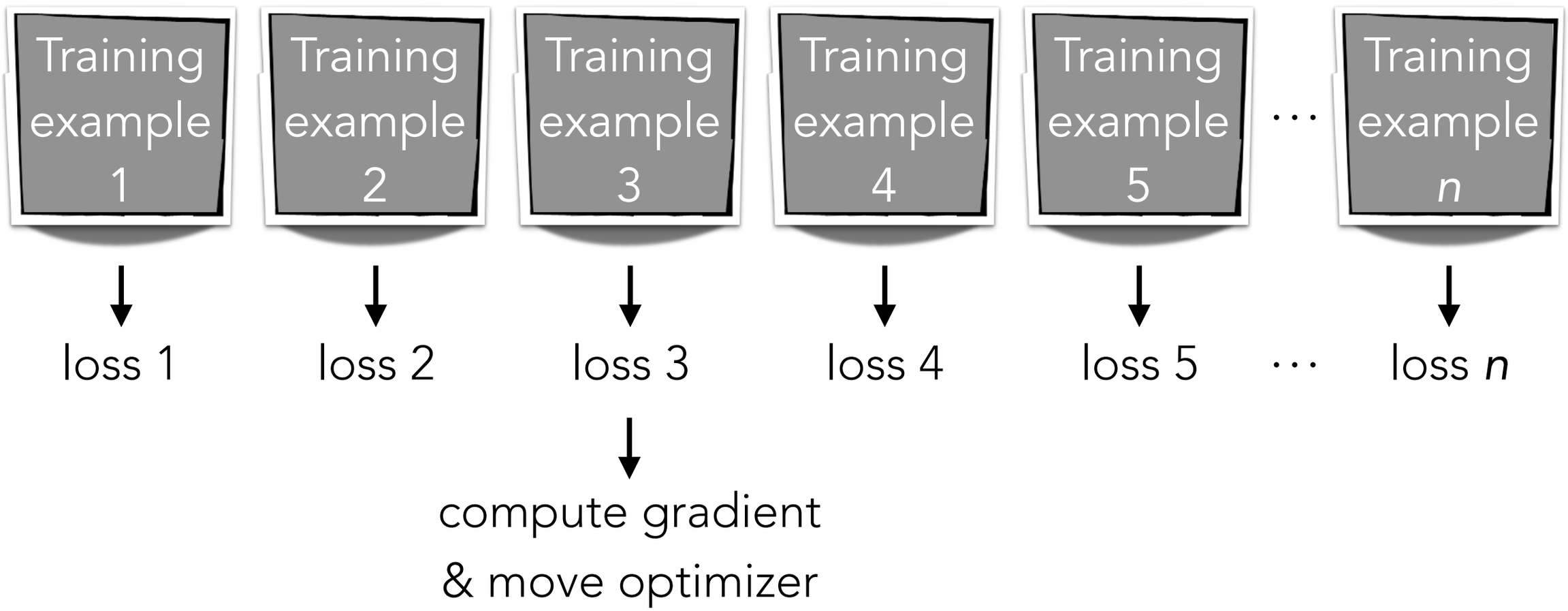
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



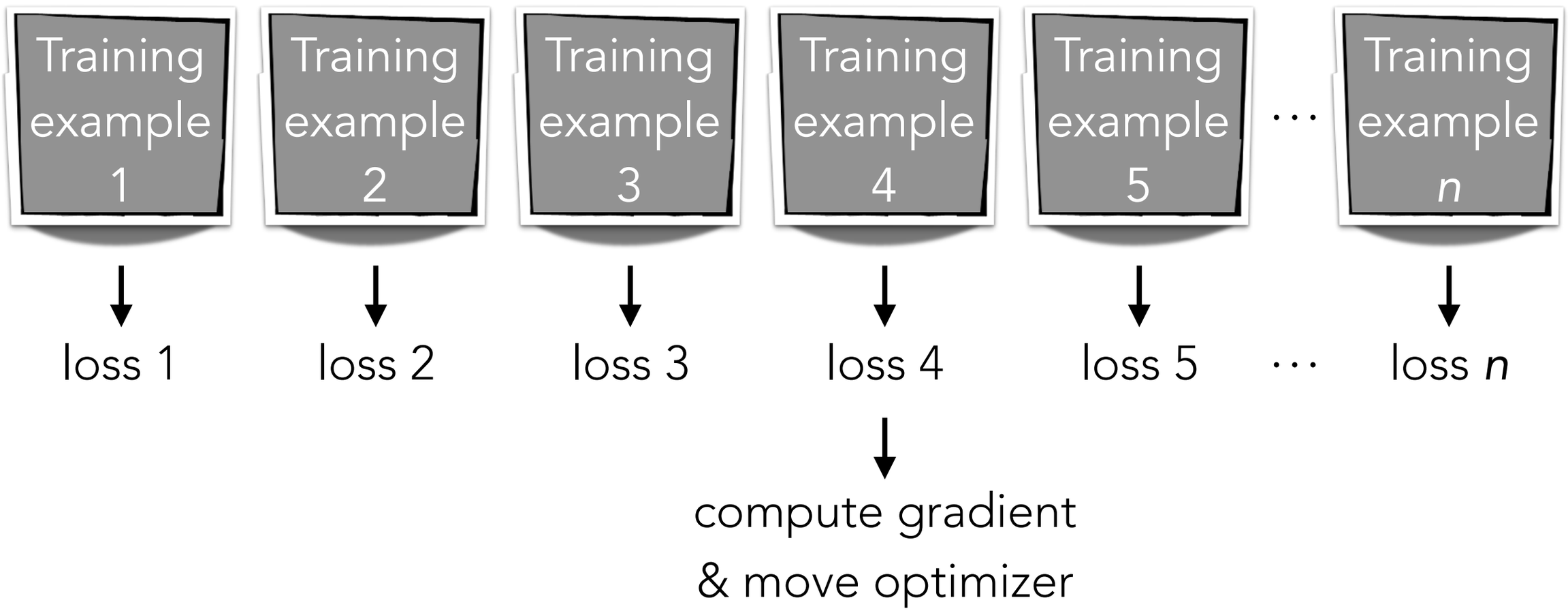
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



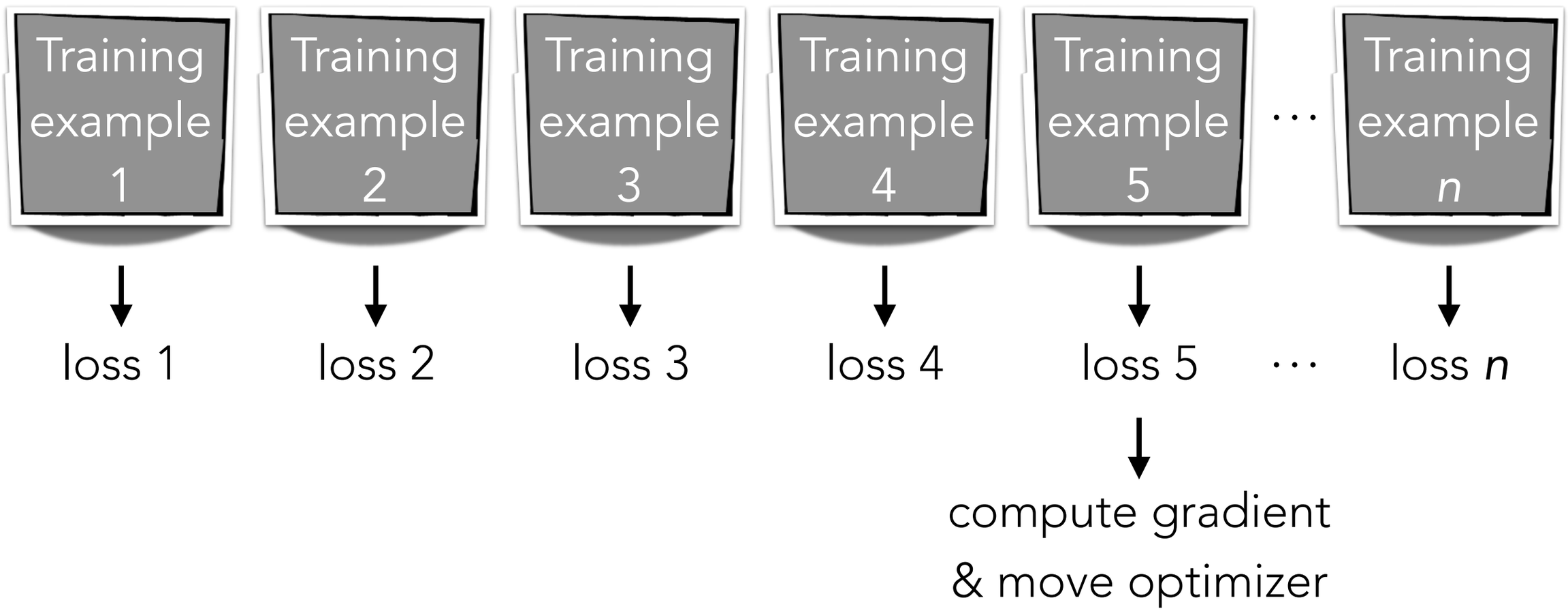
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



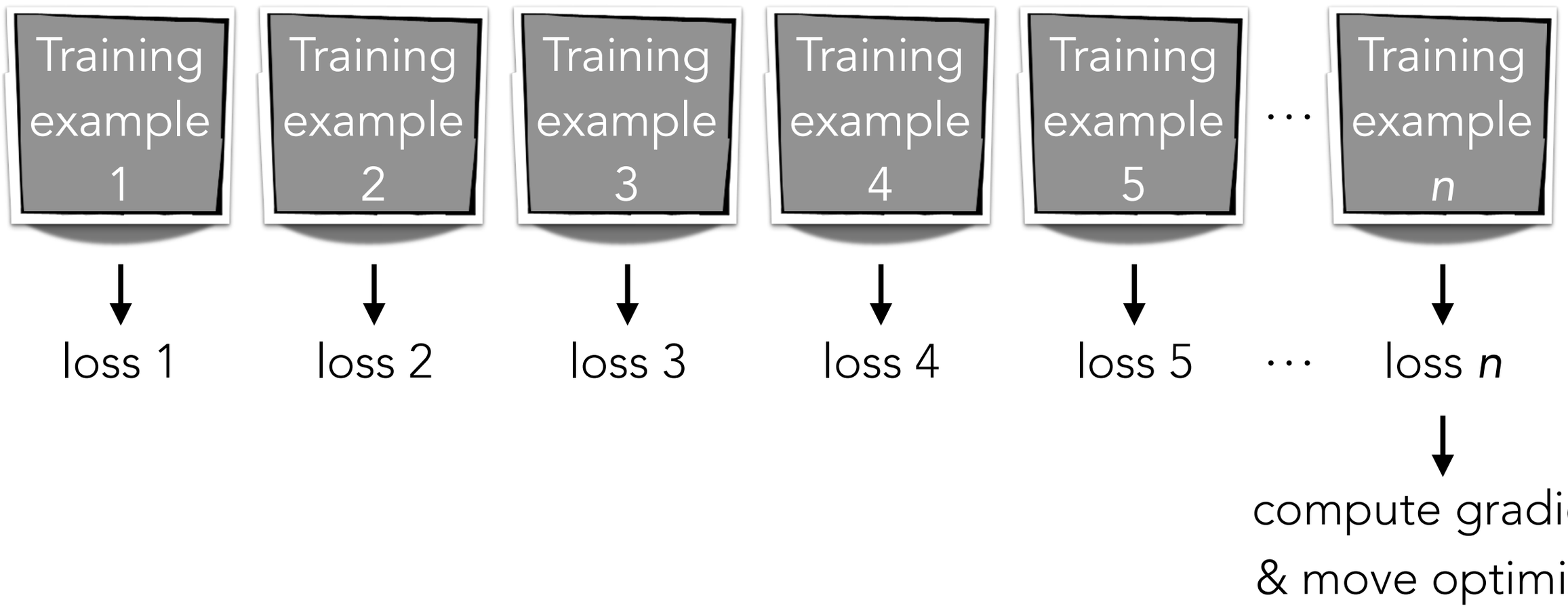
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



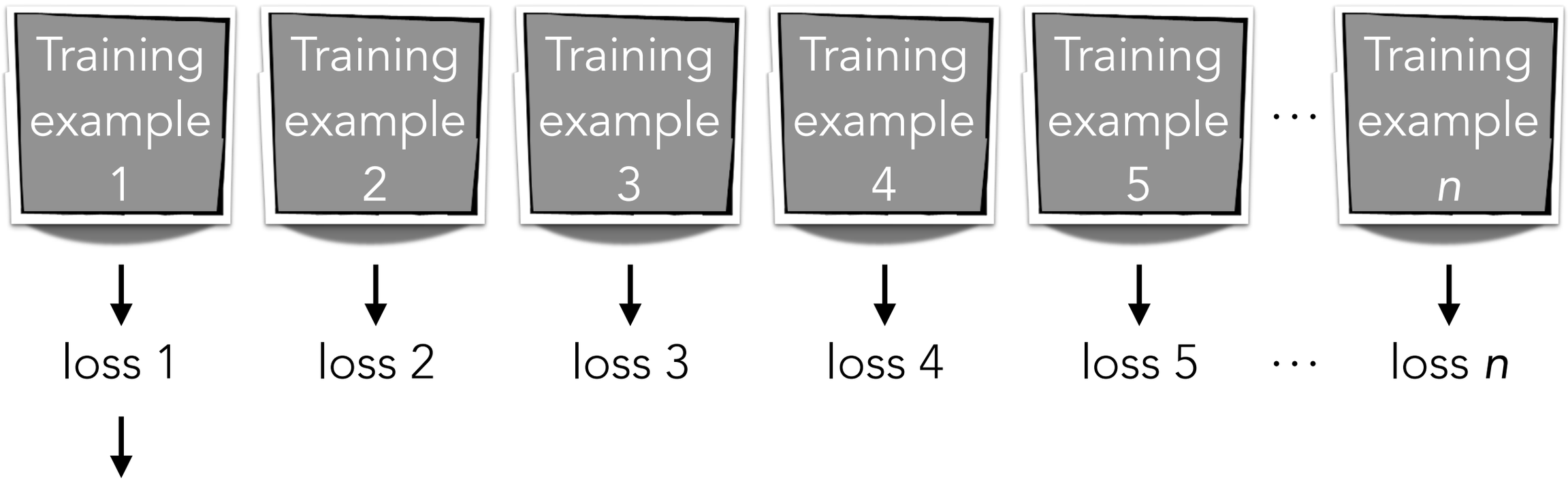
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

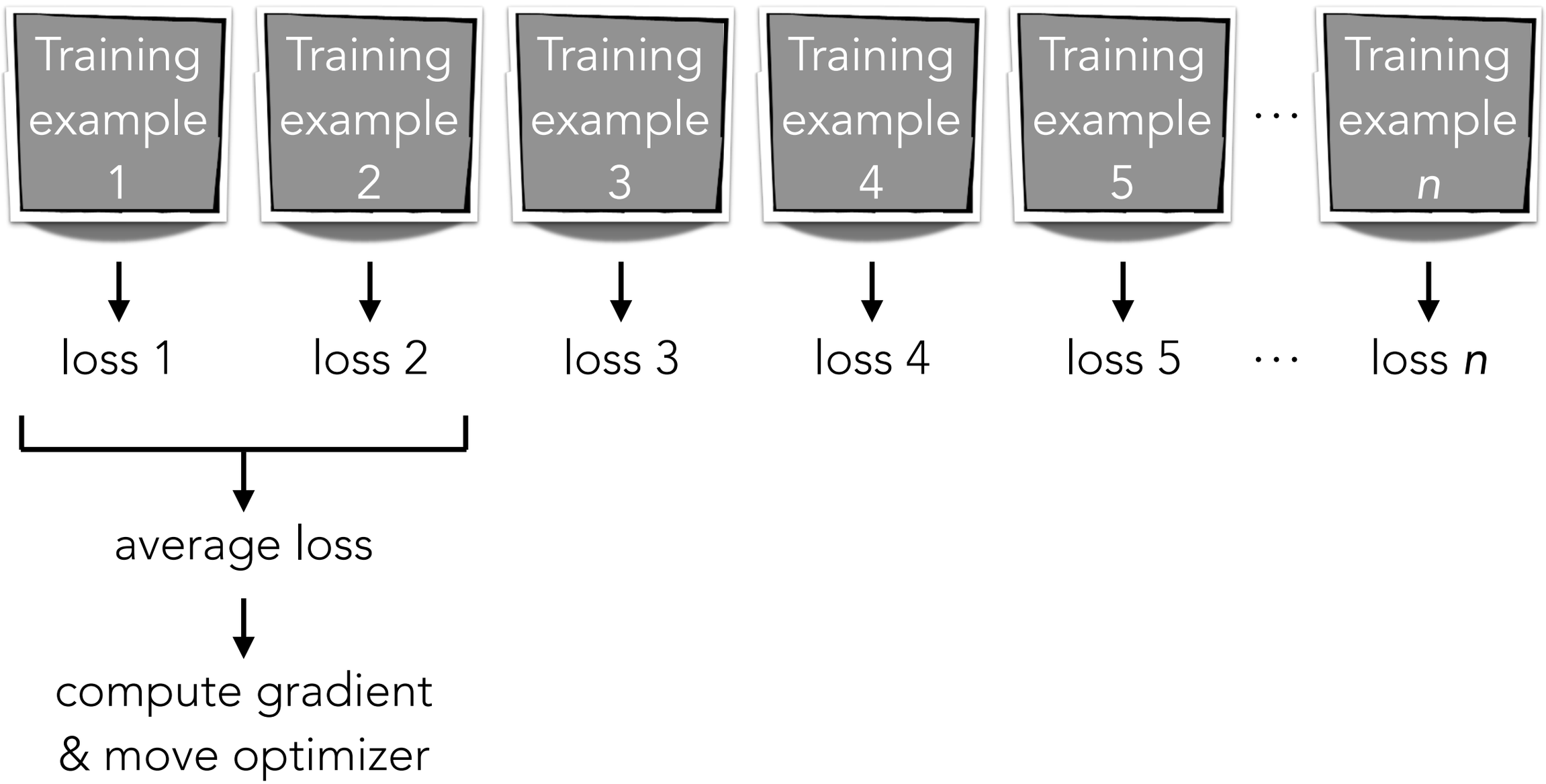
Stochastic Gradient Descent (SGD)



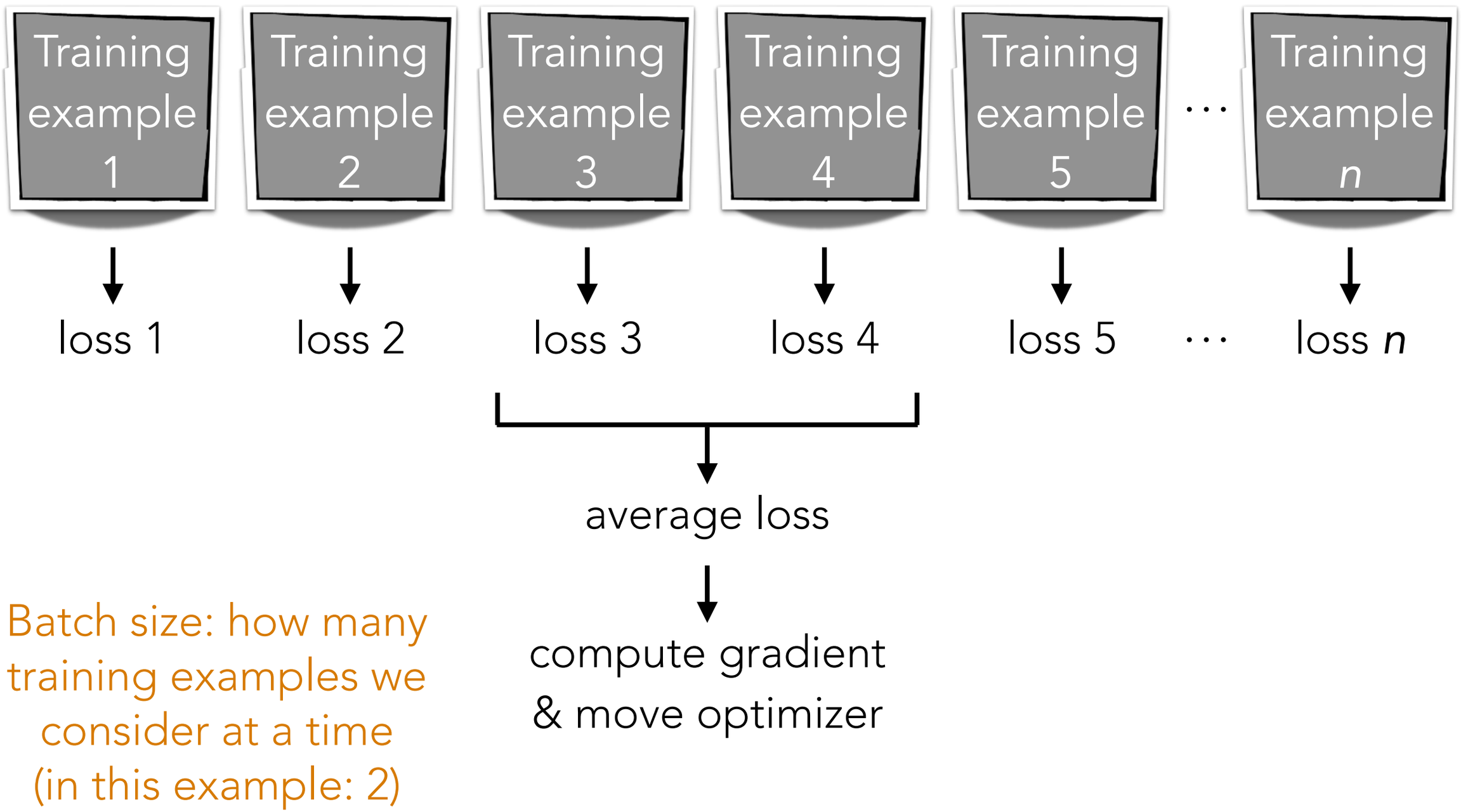
An epoch refers to 1 full pass through all the training data

SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Minibatch Gradient Descent



Minibatch Gradient Descent



Batch size: how many training examples we consider at a time (in this example: 2)

Best optimizer? Best learning rate? Best # of epochs? Best batch size?

Active area of research

Depends on problem, data, hardware, etc

Example: even with a GPU, you can get slow learning (slower than CPU!) if you choose # epochs/batch size poorly!!!

UDA_pytorch_utils.py

A look at `UDA_pytorch_classifier_fit`