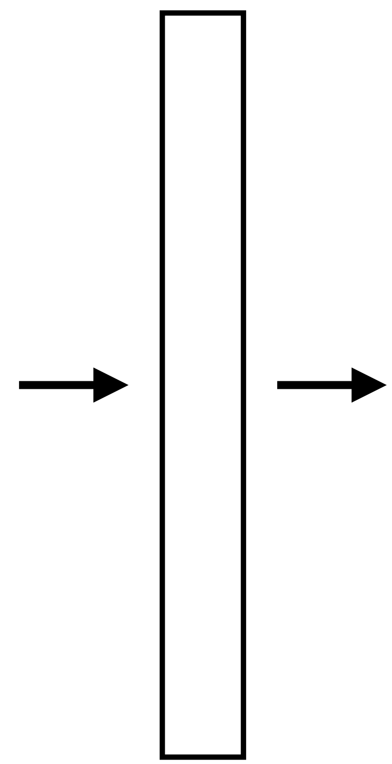
Carnegie Mellon University
Heinz College

95-865 Pittsburgh Lecture 11: Time Series Analysis With Recurrent Neural Nets

George Chen

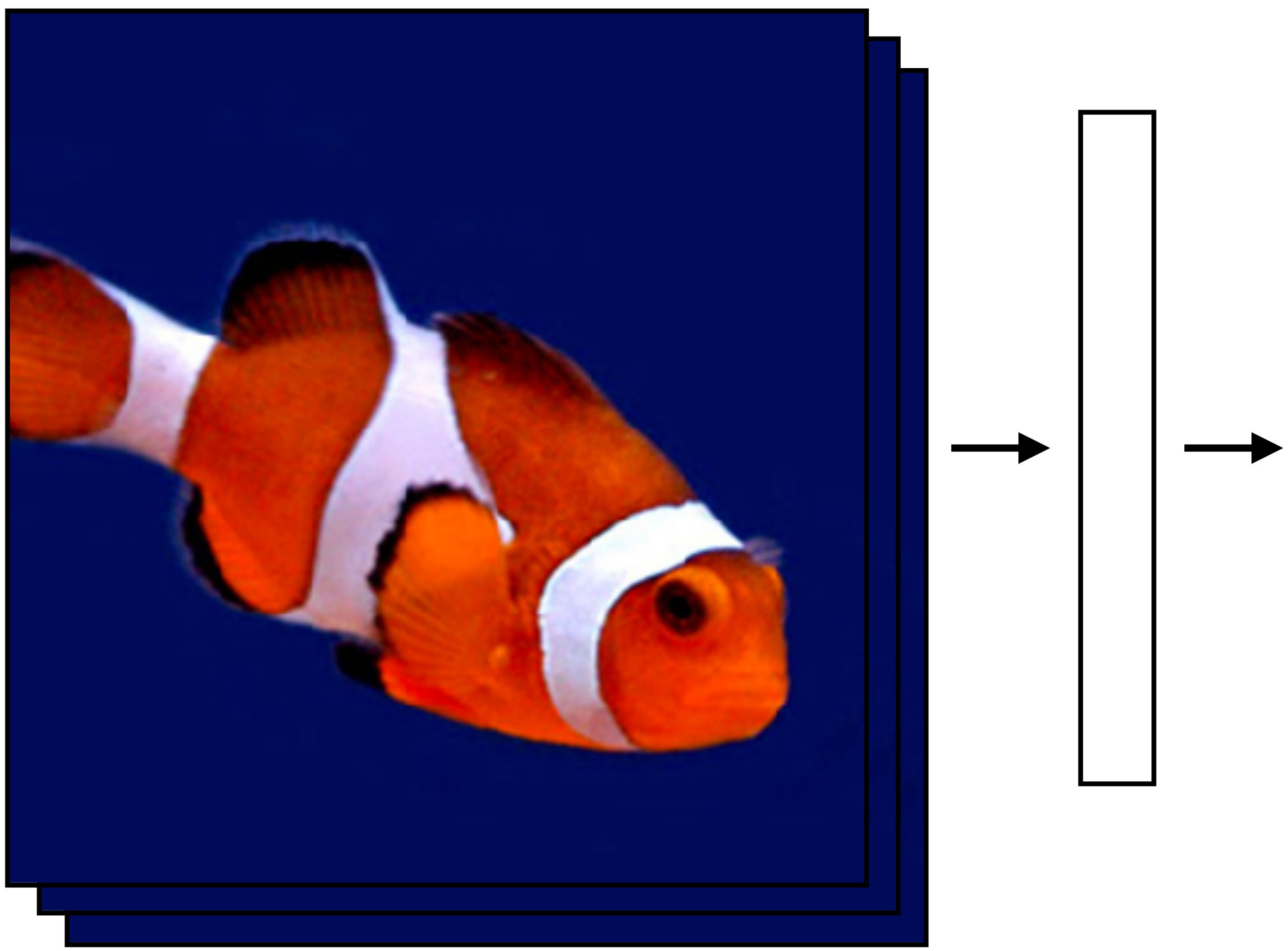
RNNs

What we've seen so far are "feedforward" NNs



RNNs

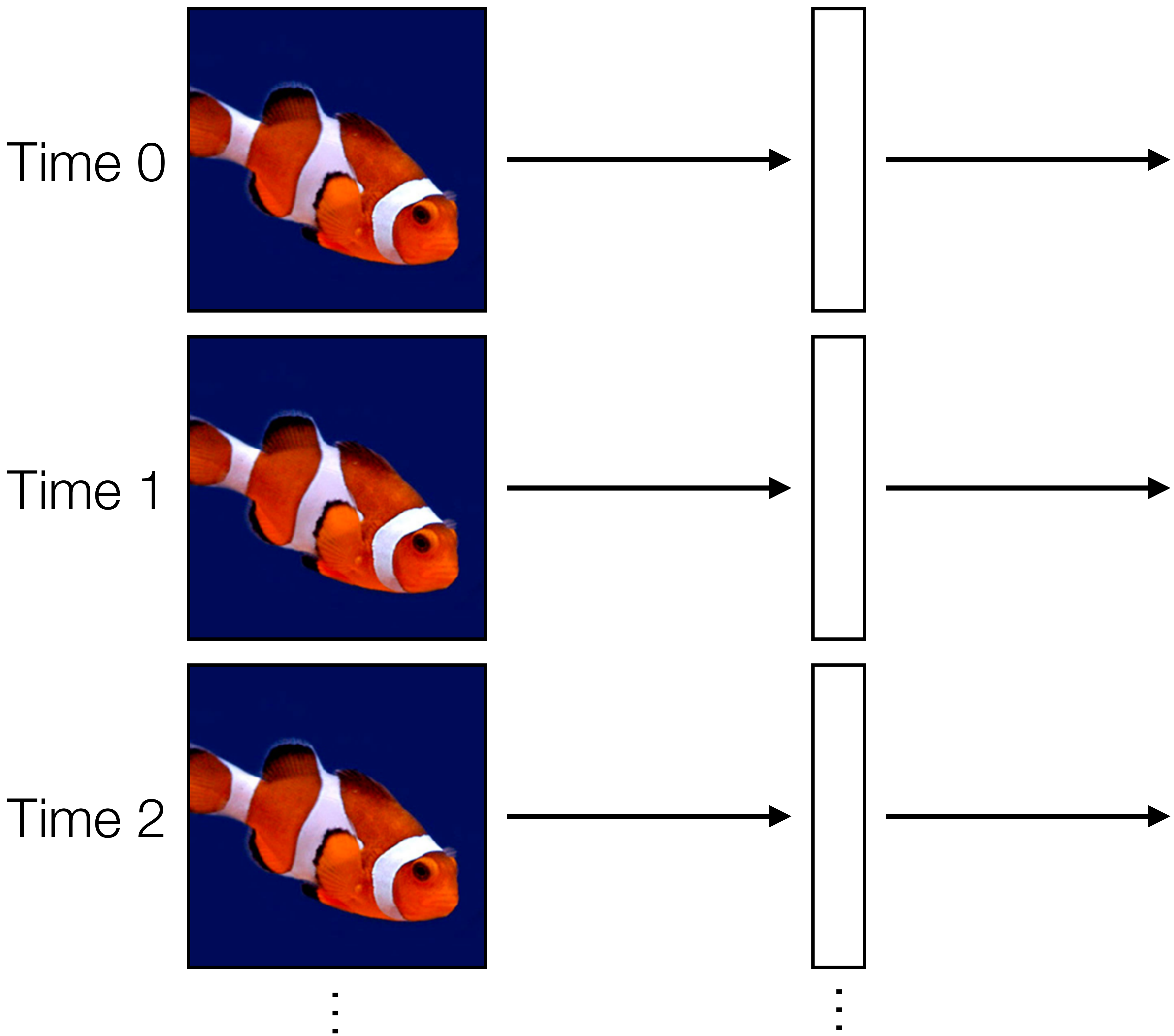
What we've seen so far are "feedforward" NNs



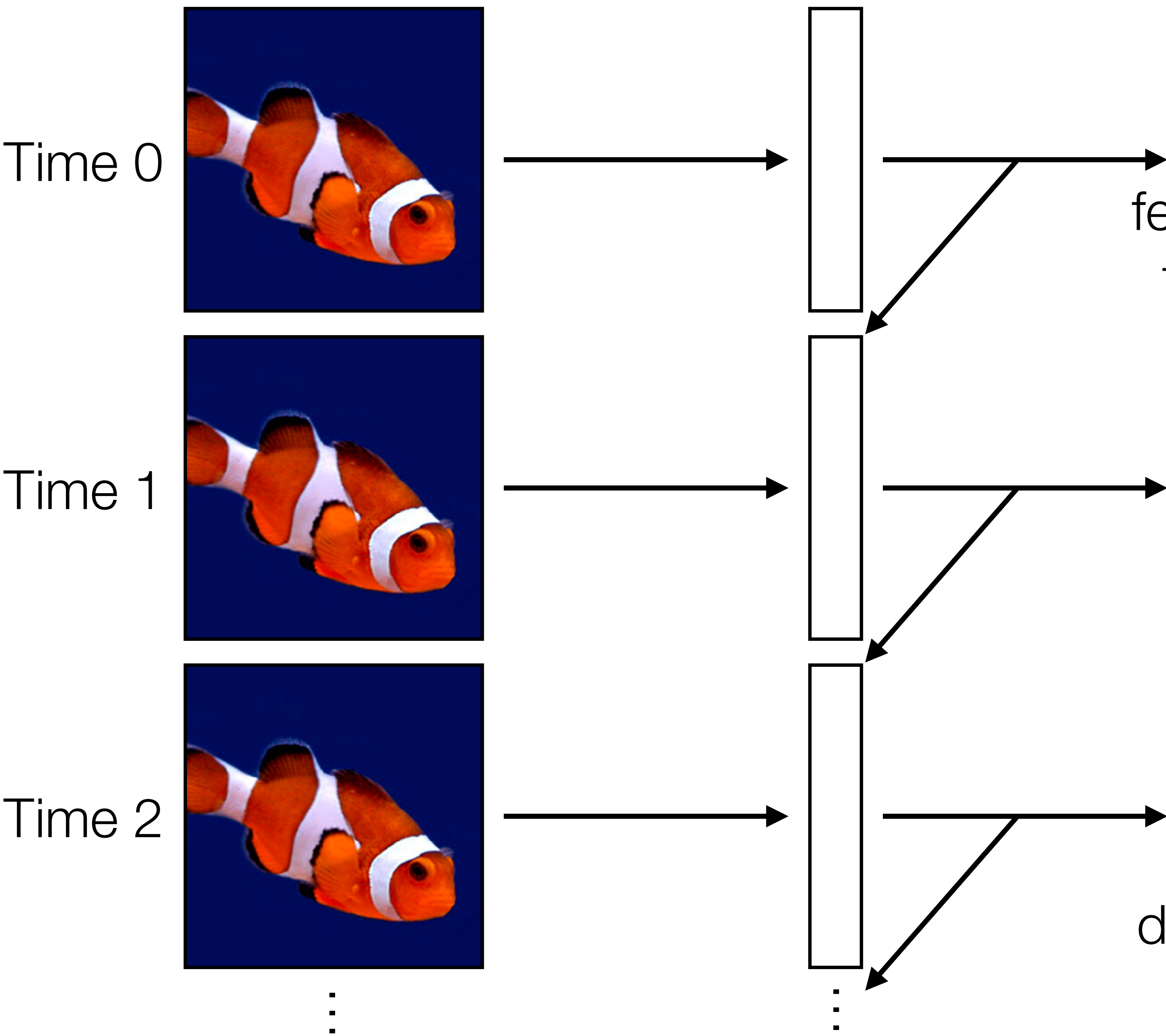
What if we had a video?

RNNs

Feedforward NN's:
treat each video frame separately



RNNs



Feedforward NN's: treat each video frame separately

RNN's: feed output at previous time step as input to RNN layer at current time step

In `keras`, different RNN options: `SimpleRNN`, `LSTM`, `GRU`

Recommendation: don't use `SimpleRNN`

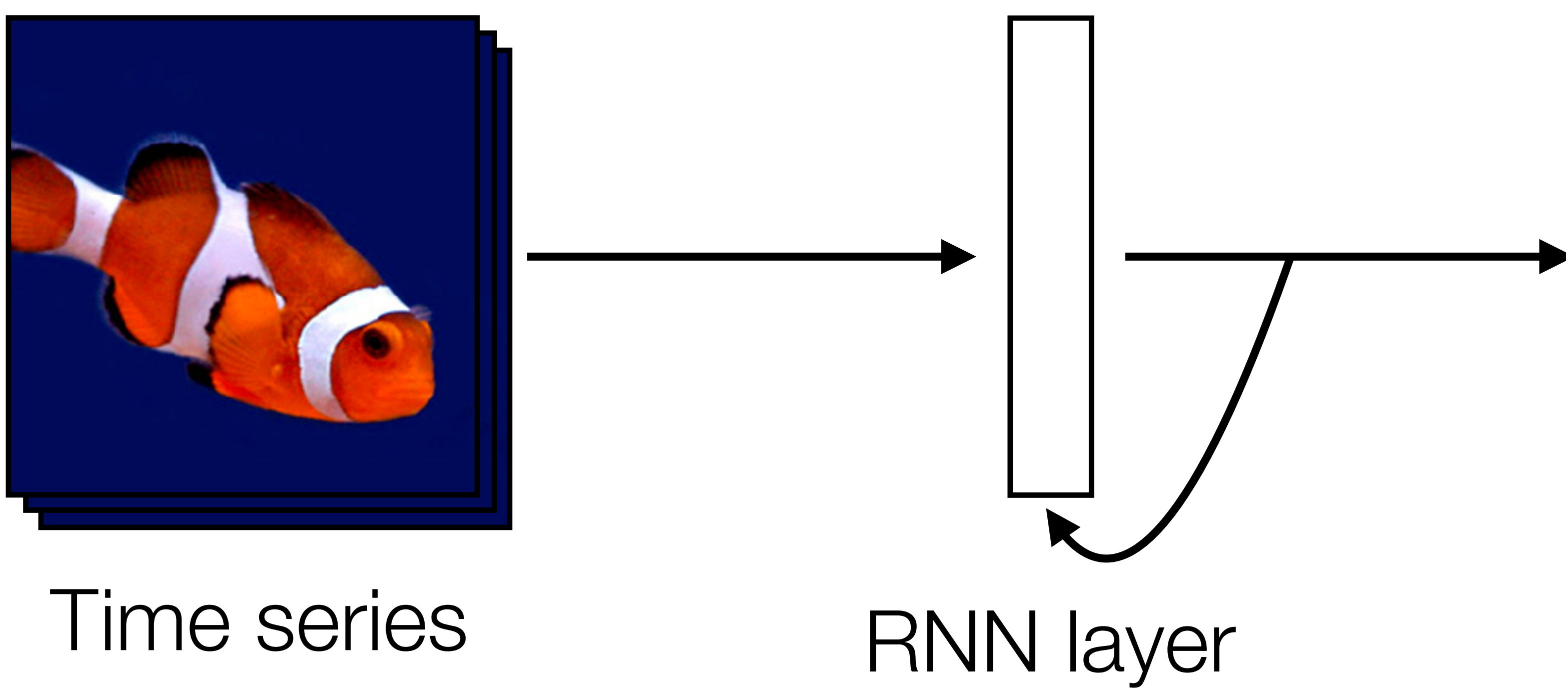
RNNs

Feedforward NN's:
treat each video frame separately

RNN's:
feed output at previous time step as input to RNN layer at current time step

In `keras`, different RNN options:
`SimpleRNN`, `LSTM`, `GRU`

Recommendation:
don't use `SimpleRNN`



Under the Hood

```
current_state = 0

for input in input_sequence:

  output = g(input, current_state)

  current_state = output
```

Different functions g correspond to different RNNs

Example: SimpleRNN

memory stored in `current_state` variable!

```
current_state = 0

for input in input_sequence:

  output = activation(np.dot(W, input)
                      + np.dot(U, current_state)
                      + b)

  current_state = output
```

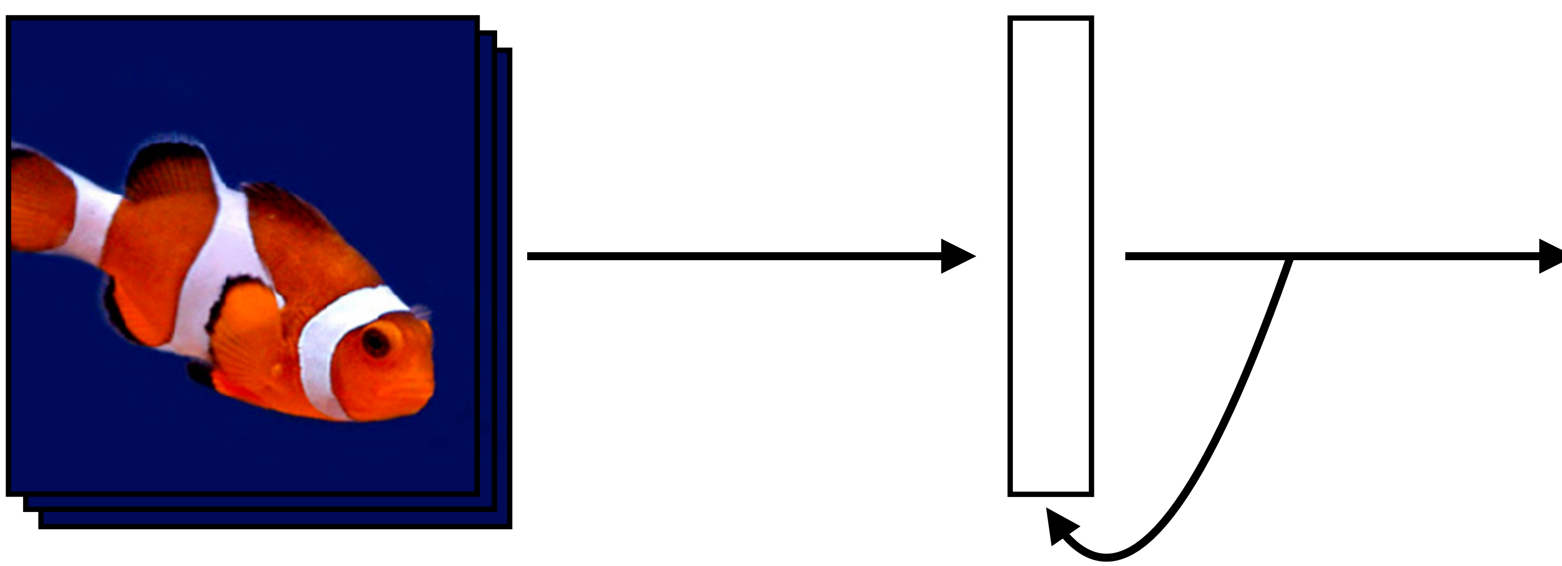
Activation function could, for instance, be ReLU

Parameters: weight matrices `W` & `U`, and bias vector `b`

Key idea: **it's like a dense layer in a `for` loop with some memory!**

RNNs

readily chains together with other neural net layers



Time series

RNN layer

like a dense layer that has memory

Feedforward NN's: treat each video frame separately

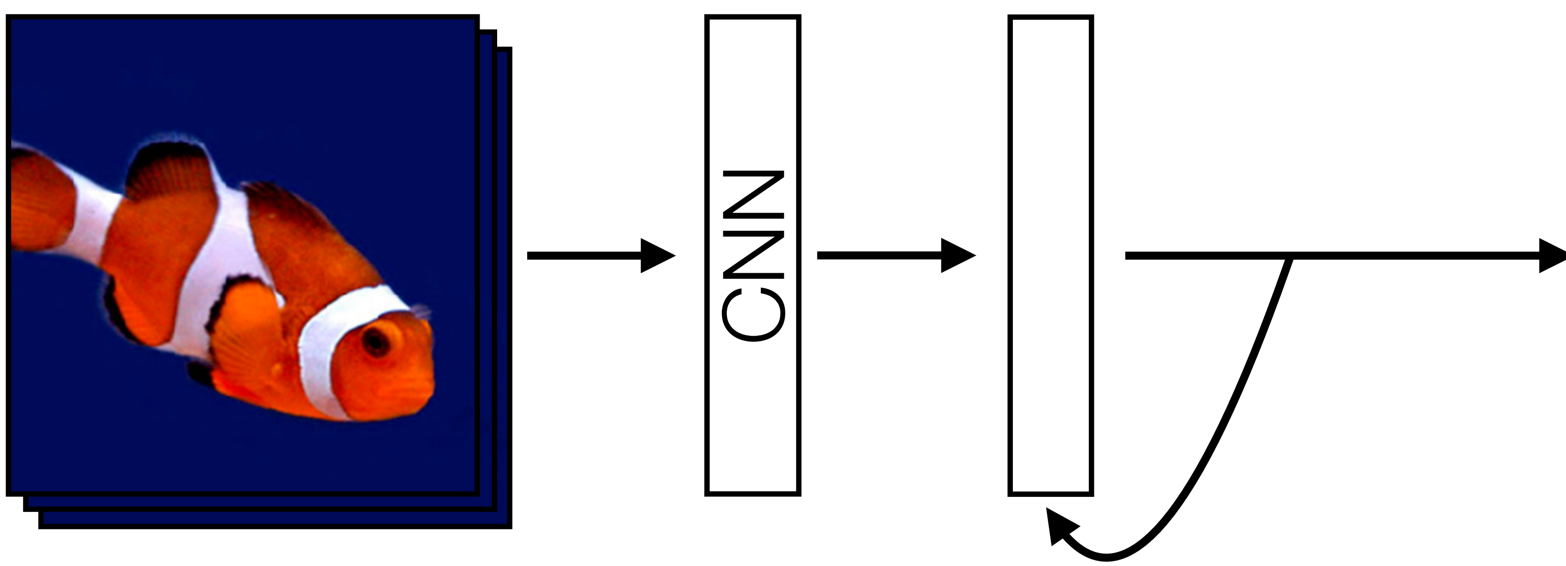
RNN's: feed output at previous time step as input to RNN layer at current time step

In `keras`, different RNN options: `SimpleRNN`, `LSTM`, `GRU`

Recommendation: don't use `SimpleRNN`

RNNs

readily chains together with other neural net layers



Time series

RNN layer

like a dense layer that has memory

Feedforward NN's: treat each video frame separately

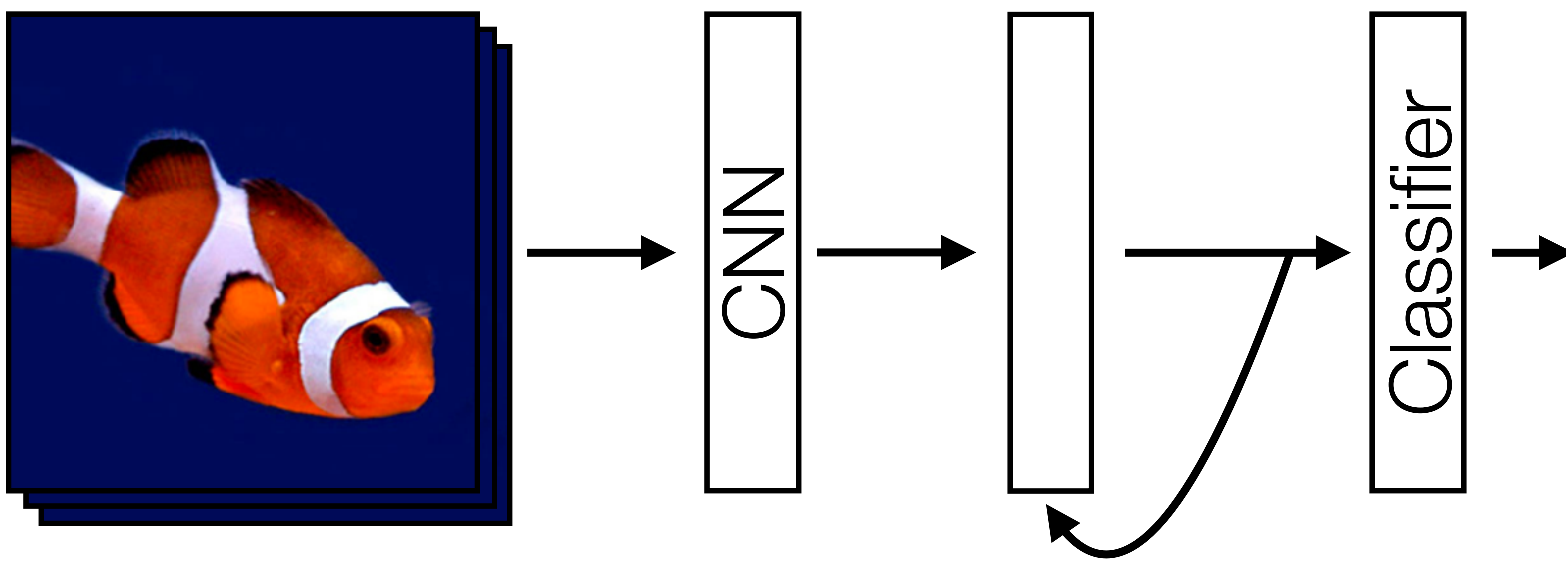
RNN's: feed output at previous time step as input to RNN layer at current time step

In `keras`, different RNN options: `SimpleRNN`, `LSTM`, `GRU`

Recommendation: don't use `SimpleRNN`

RNNs

readily chains together with other neural net layers



Time series

RNN layer

like a dense layer that has memory

Feedforward NN's: treat each video frame separately

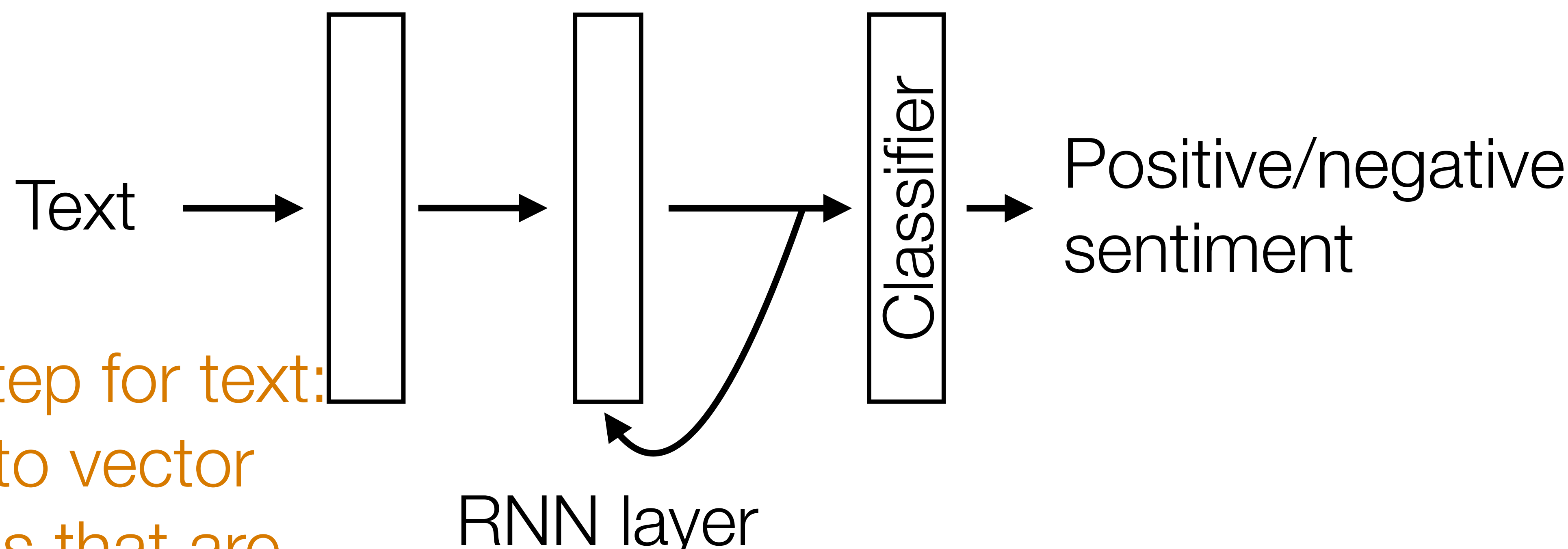
RNN's: feed output at previous time step as input to RNN layer at current time step

In `keras`, different RNN options: `SimpleRNN`, `LSTM`, `GRU`

Recommendation: don't use `SimpleRNN`

RNNs

Example: Given text (e.g., movie review, Tweet), figure out whether it has positive or negative sentiment (binary classification)



Common first step for text: turn words into vector representations that are semantically meaningful

(Flashback) Example Application of PMI: Word Embeddings

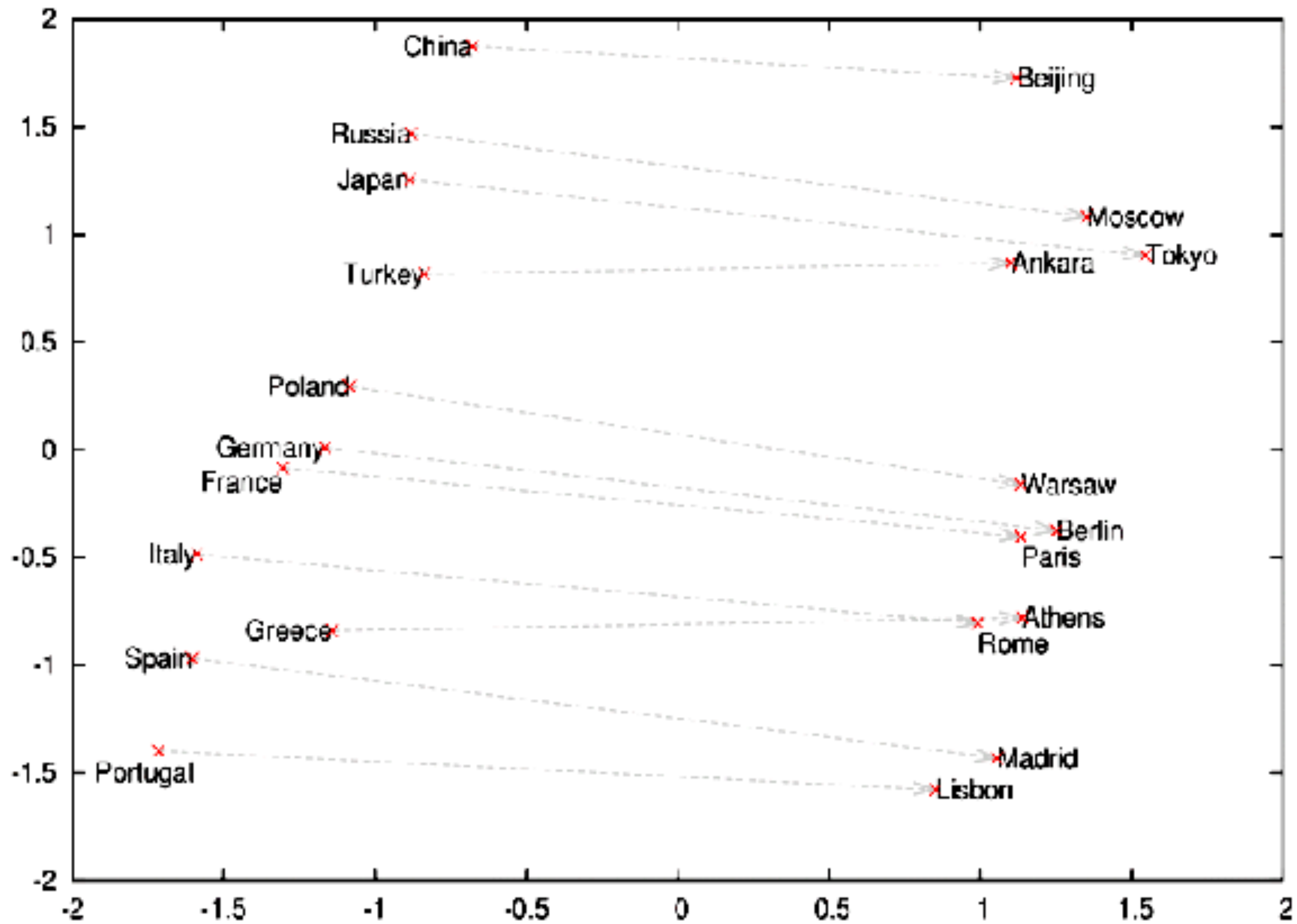


Image source: https://deeplearning4j.org/img/countries_capitals.png

Omer Levy and Yoav Goldberg. Neural word embeddings as implicit matrix factorization. NIPS 2014.

(Flashback) Do Data Actually Live on Manifolds?

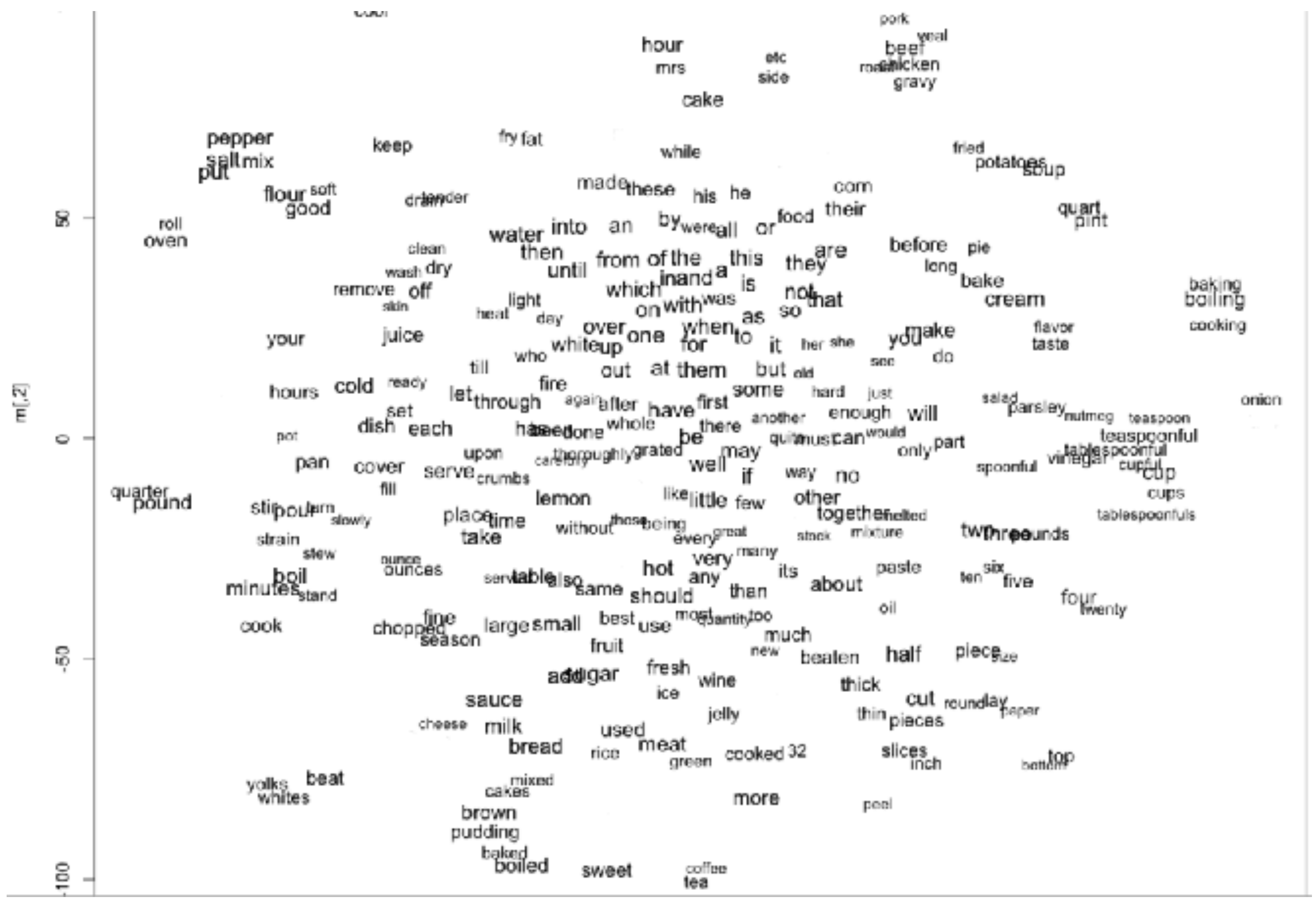
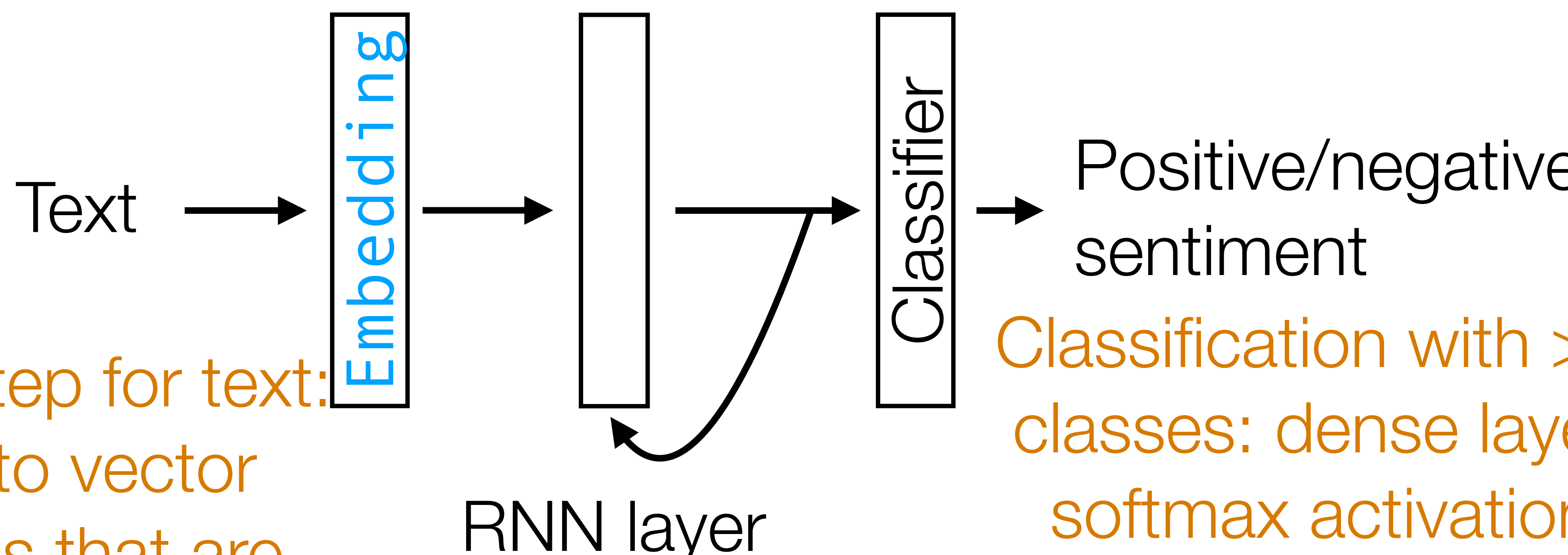


Image source: http://www.adityathakker.com/wp-content/uploads/2017/06/word-embeddings-994x675.png

RNNs

for loss function, replace *category cross entropy* with *binary cross entropy*

Example: Given text (e.g., movie review, Tweet), figure out whether it has positive or negative sentiment (binary classification)



Common first step for text: turn words into vector representations that are semantically meaningful

In `keras`, use the `Embedding` layer

Classification with > 2 classes: dense layer, softmax activation

Classification with 2 classes: dense layer with 2 neurons & softmax equivalent to dense layer with 1 neuron & sigmoid activation (called **logistic regression**)

Word Embeddings

Example of **self-supervised learning**

Even without labels, we can set up a prediction task!

Hide part of training data and try to predict what you've hid!

Word embeddings will be covered in your next recitation
(it's a clever application of predictive data analytics concepts)

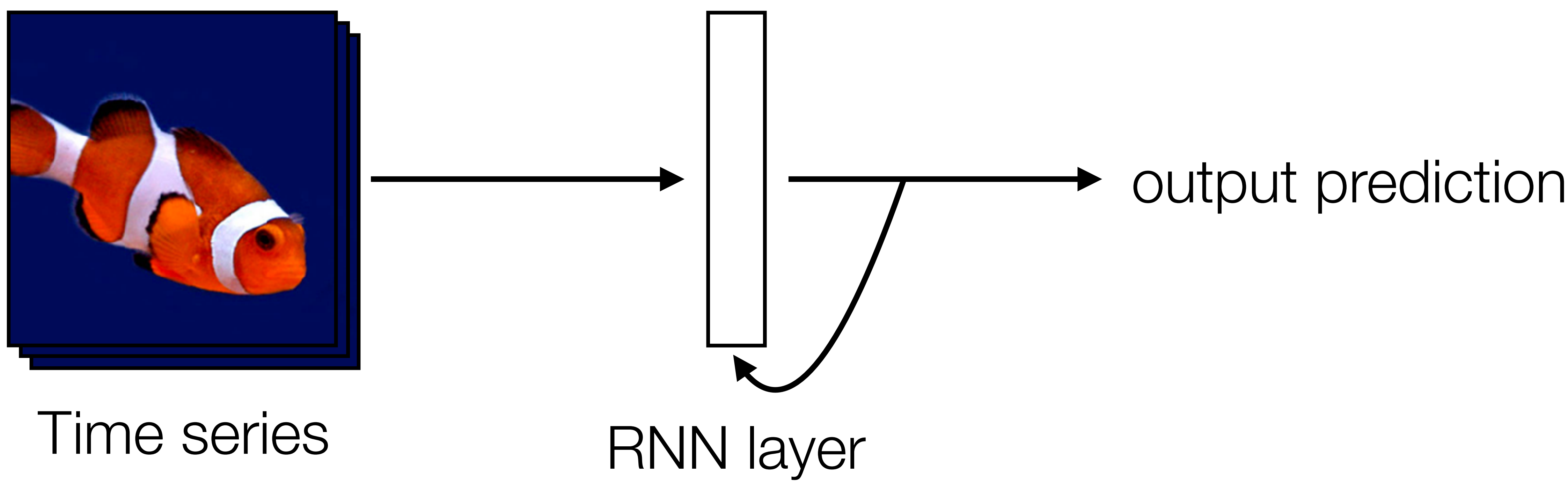
RNNs

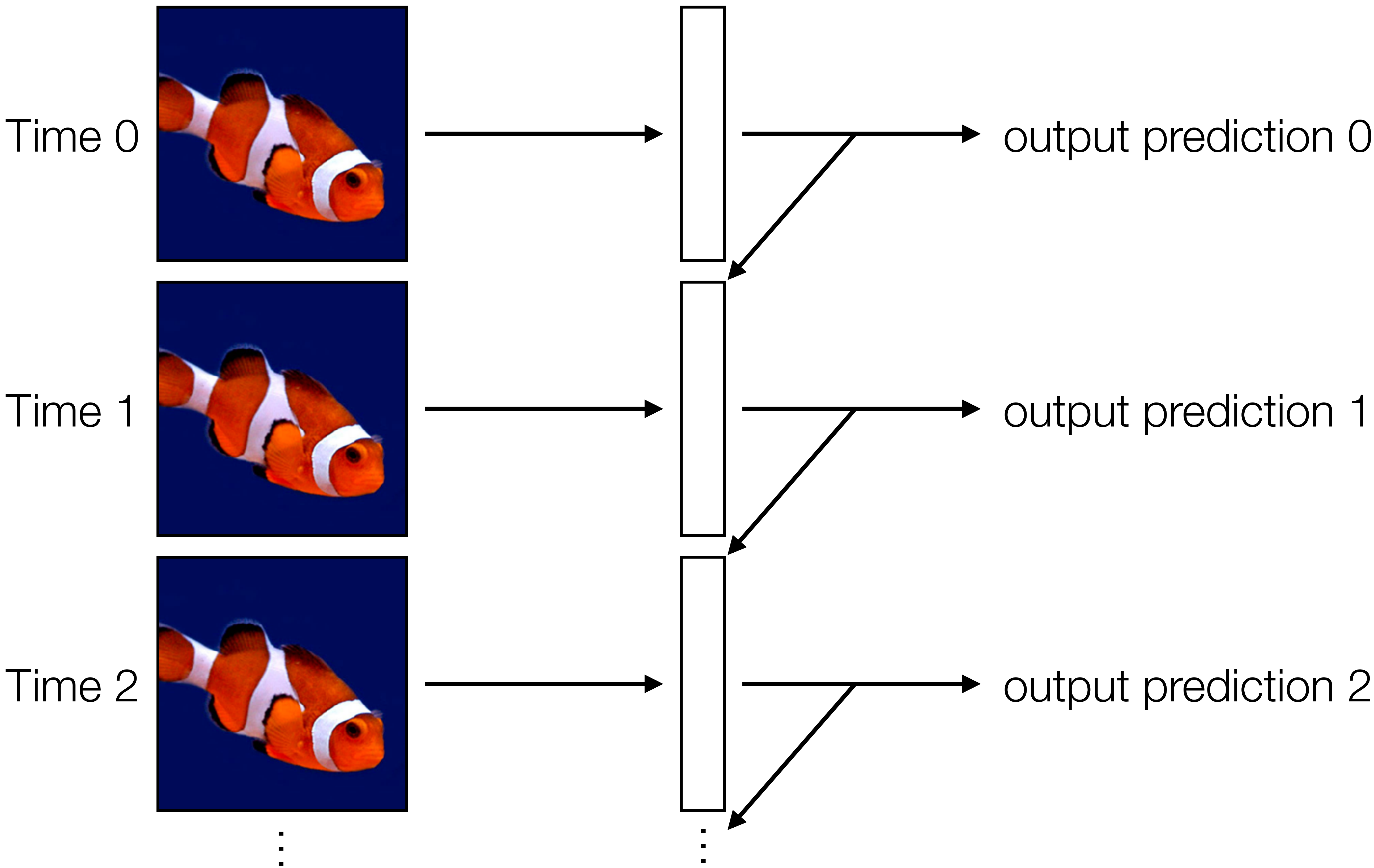
Demo

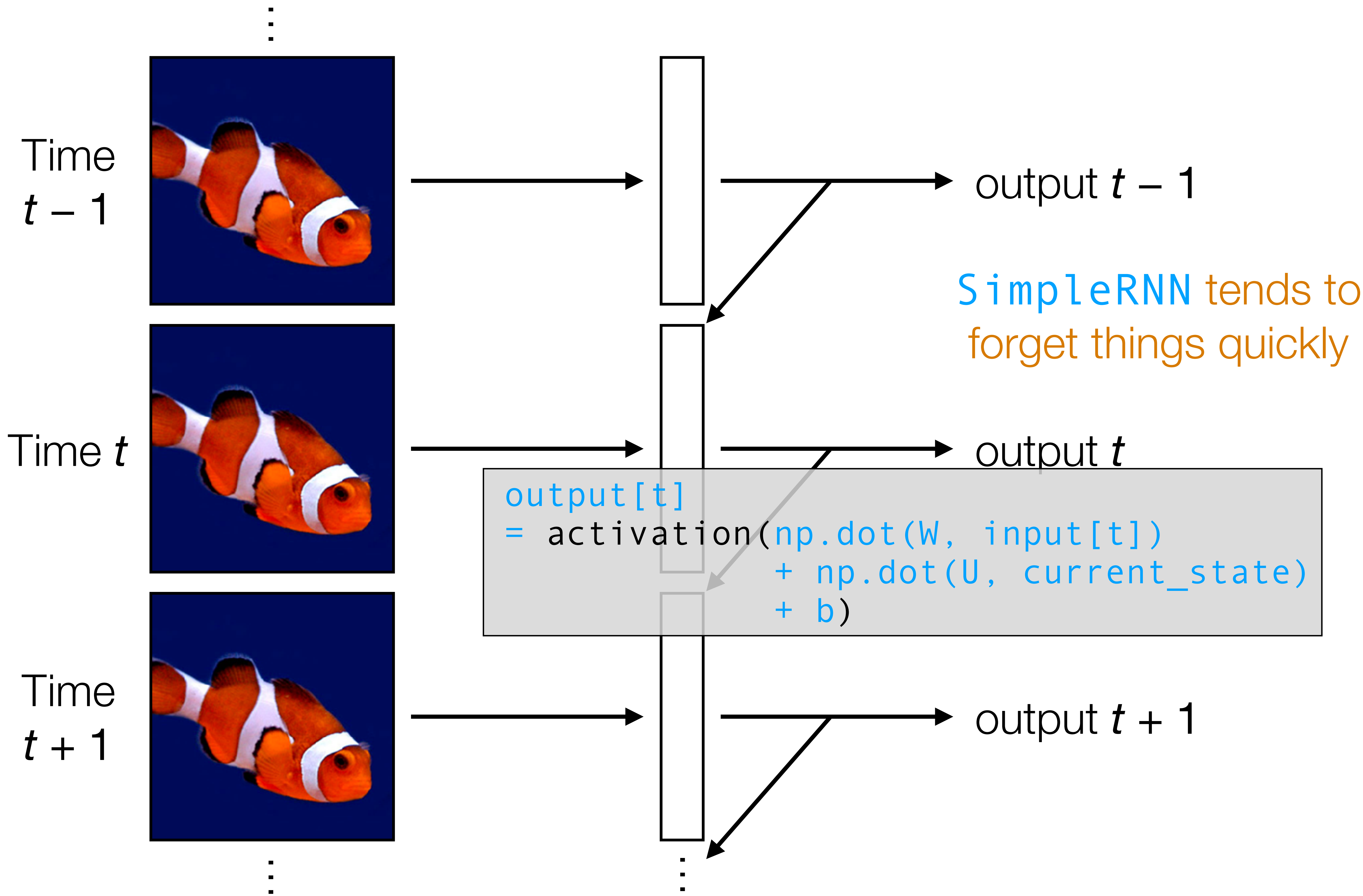
RNNs

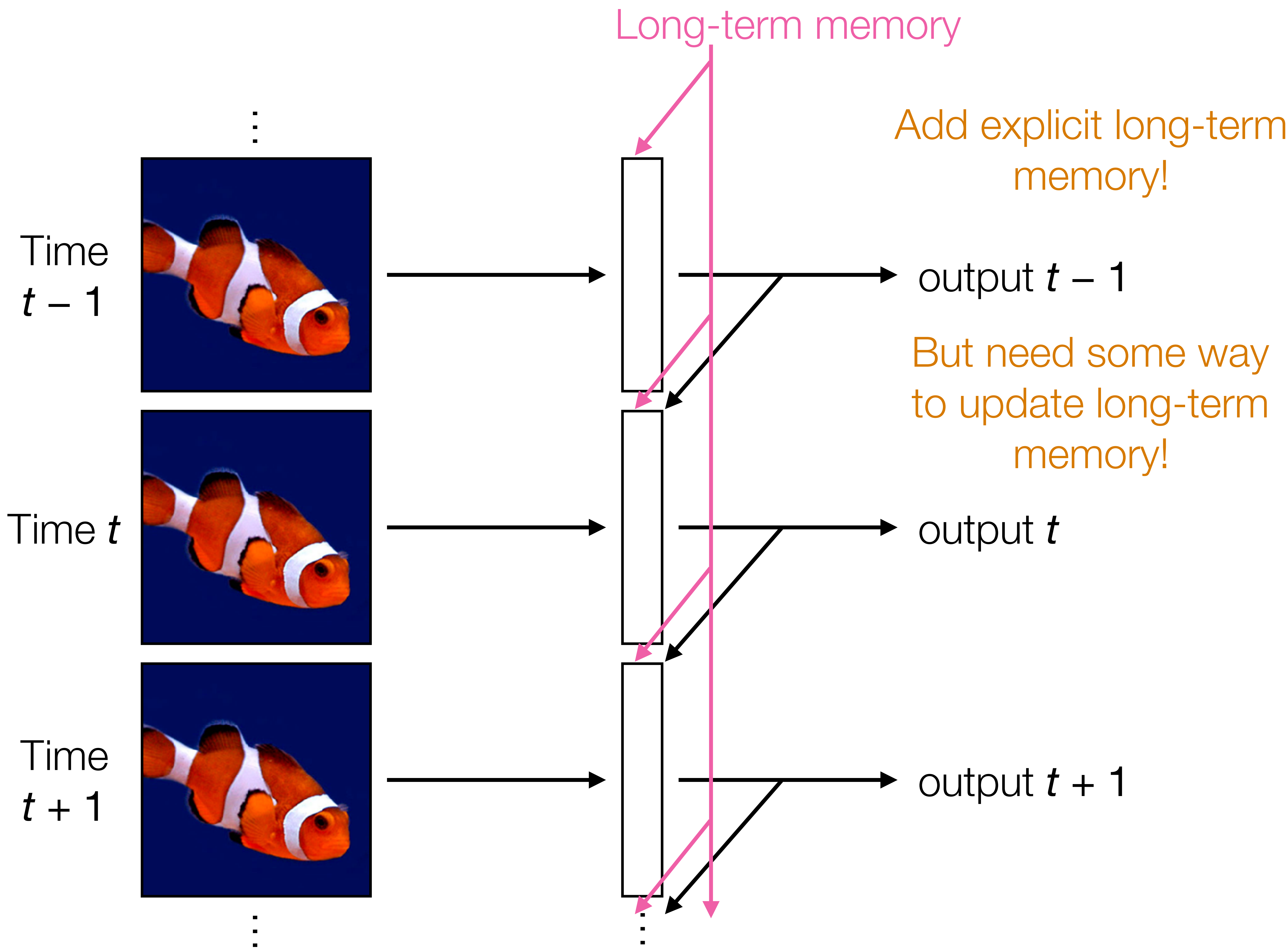
- Neatly handles time series in which there is some sort of global structure, so memory helps
  - If time series doesn't have global structure, RNN performance might not be much better than 1D CNN
- An RNN layer by itself doesn't take advantage of image/text structure!
  - For images: combine with convolution layer(s)
  - For text: combine with embedding layer

A Little Bit More Detail

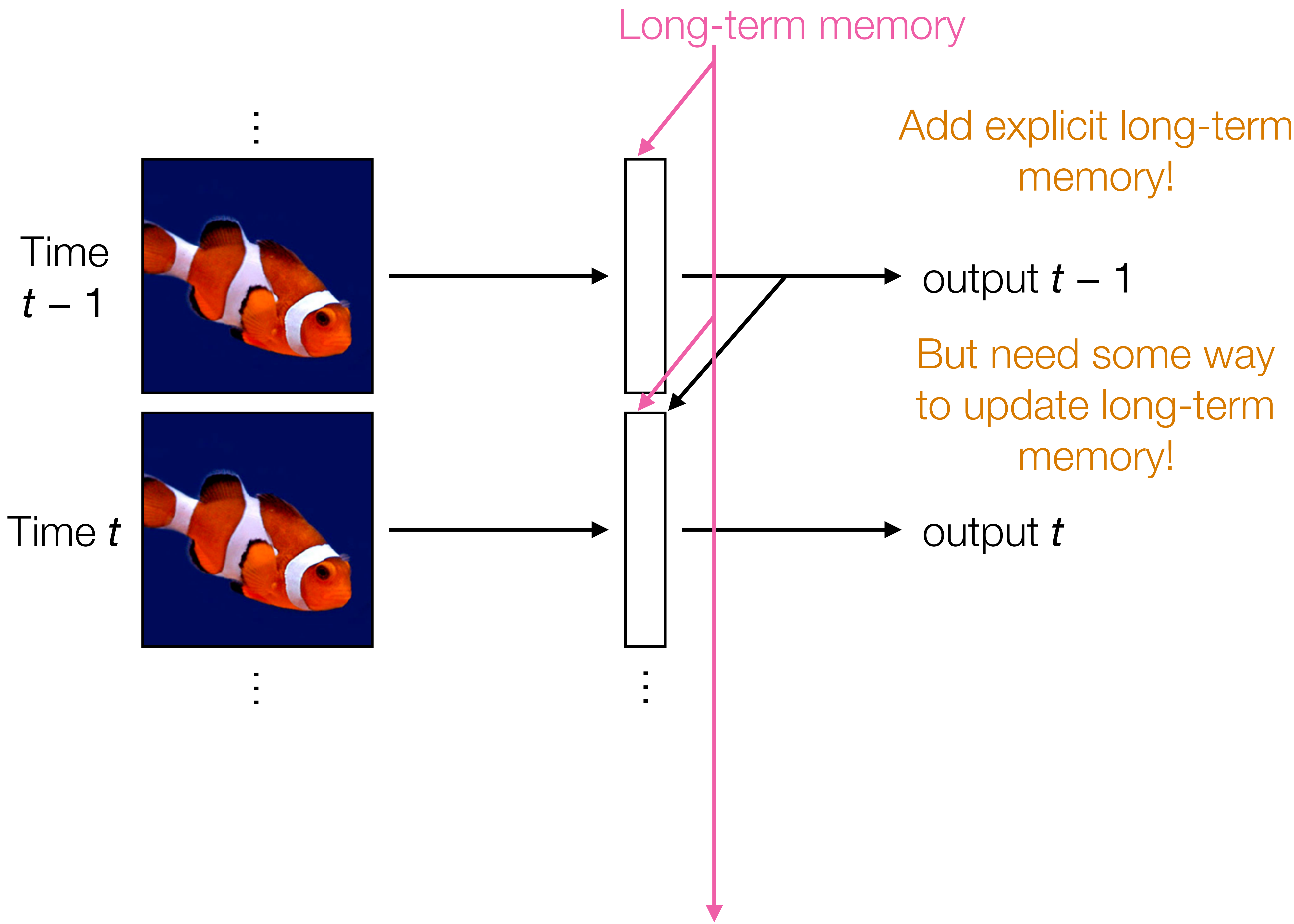


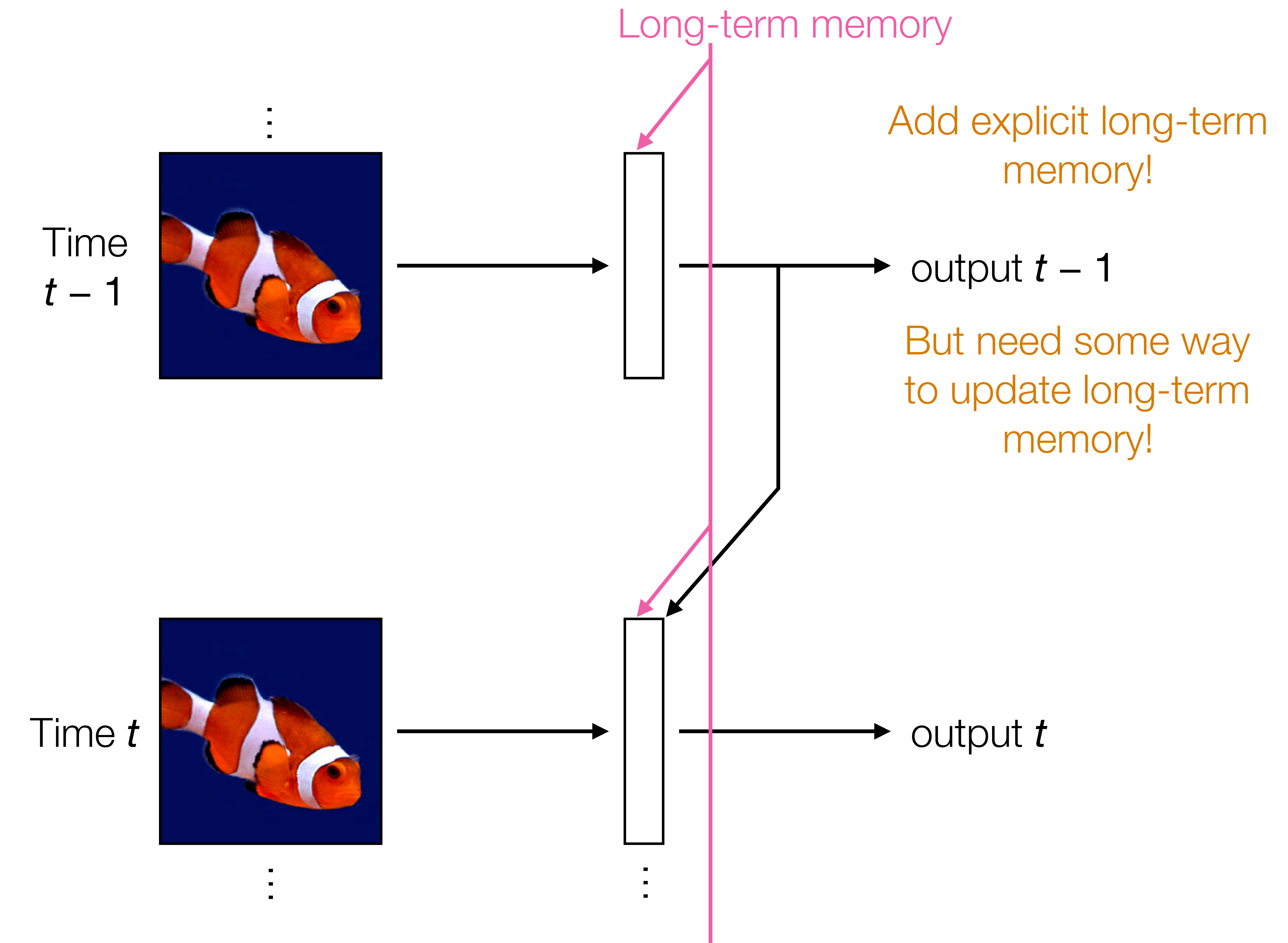

Time 0
Time 1
Time 2
output prediction 0
output prediction 1
output prediction 2

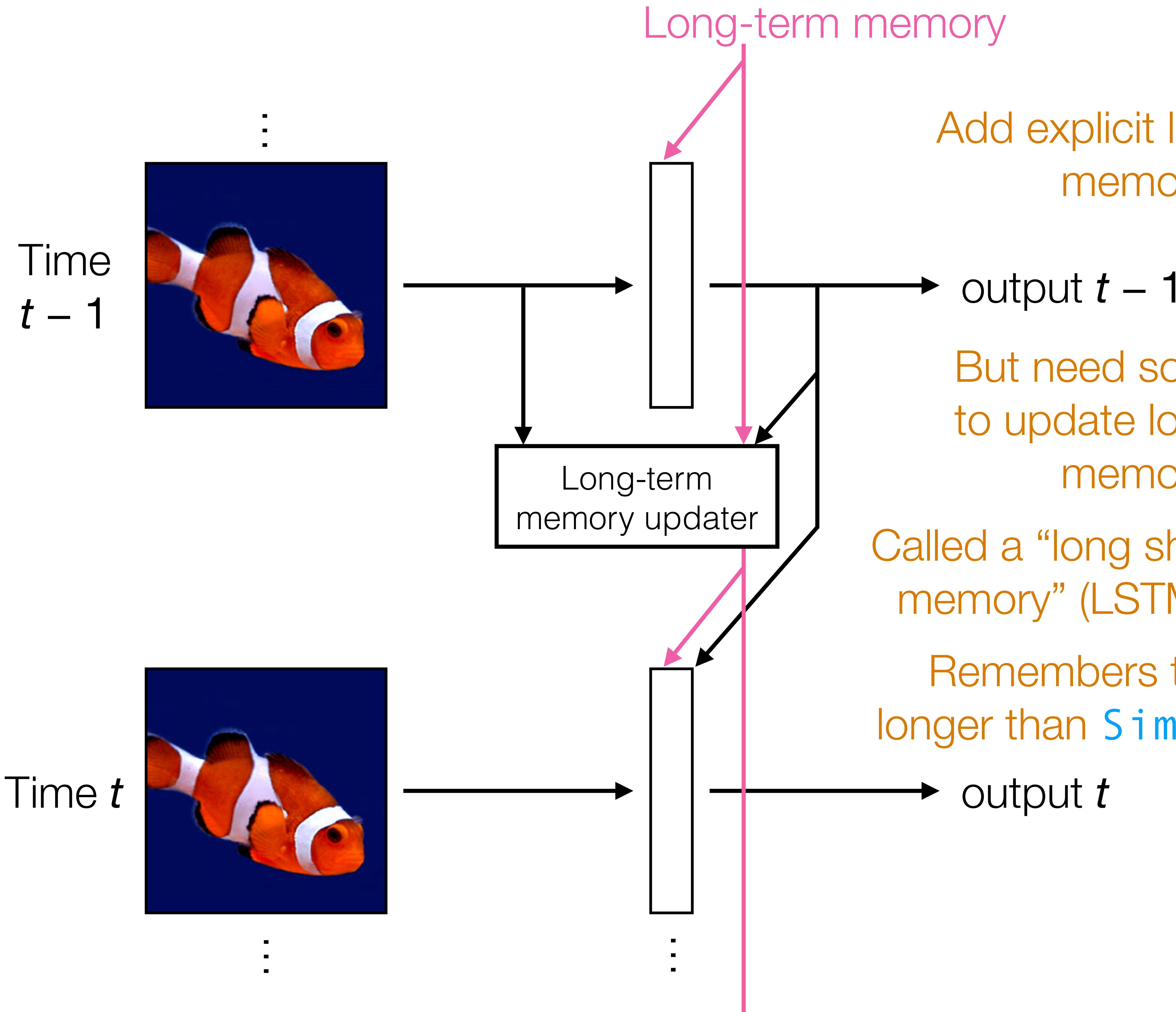

Time
t – 1
Time t
Time
t + 1
output t – 1
output t
output t + 1
SimpleRNN tends to
forget things quickly
output[t]
= activation(np.dot(W, input[t])
+ np.dot(U, current_state)
+ b)



Long-term memory
Add explicit long-term memory!
But need some way to update long-term memory!
Time $t-1$
Time t
Time $t+1$
output $t-1$
output t
output $t+1$


Long-term memory
Add explicit long-term memory!
Time $t-1$
output $t-1$
But need some way to update long-term memory!
Time t
output t


Long-term memory
Add explicit long-term memory!
Time $t-1$
output $t-1$
But need some way to update long-term memory!
Time t
output t



Add explicit long-term memory!

But need some way to update long-term memory!

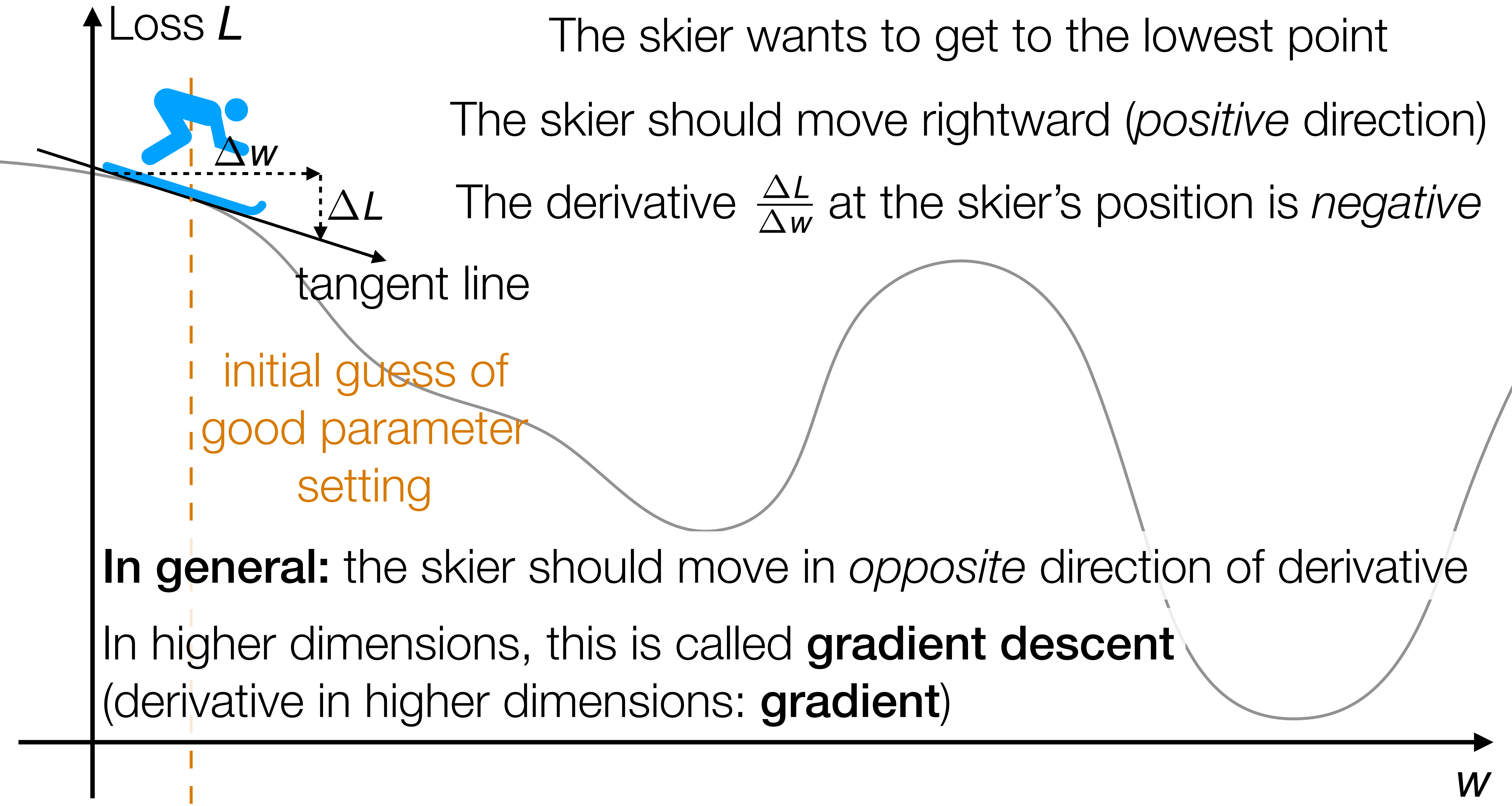
Called a "long short-term memory" (LSTM) RNN

Remembers things longer than `SimpleRNN`

Learning a Deep Net

Gradient Descent

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)

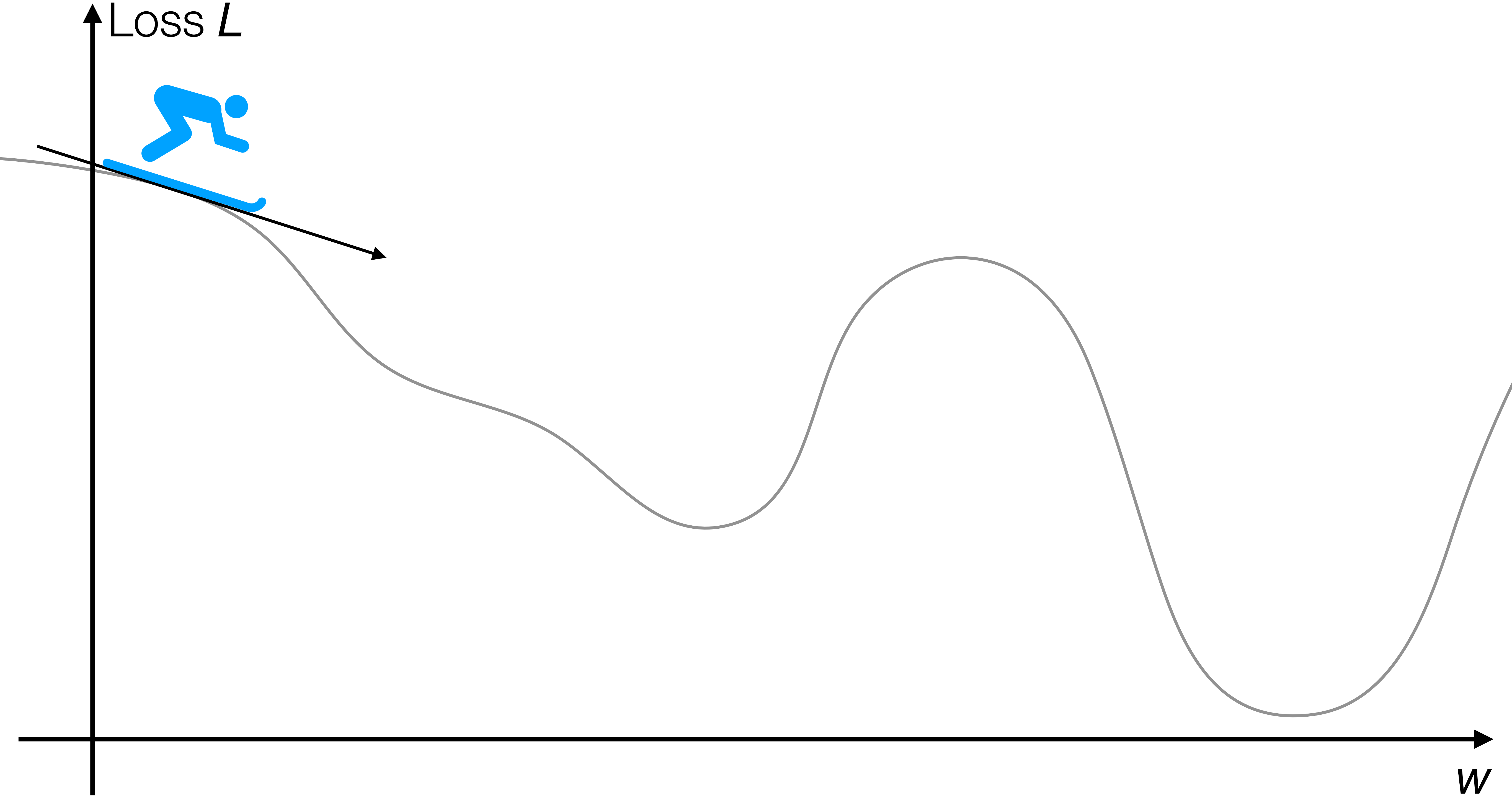
The derivative $\frac{\Delta L}{\Delta w}$ at the skier's position is *negative*

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

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

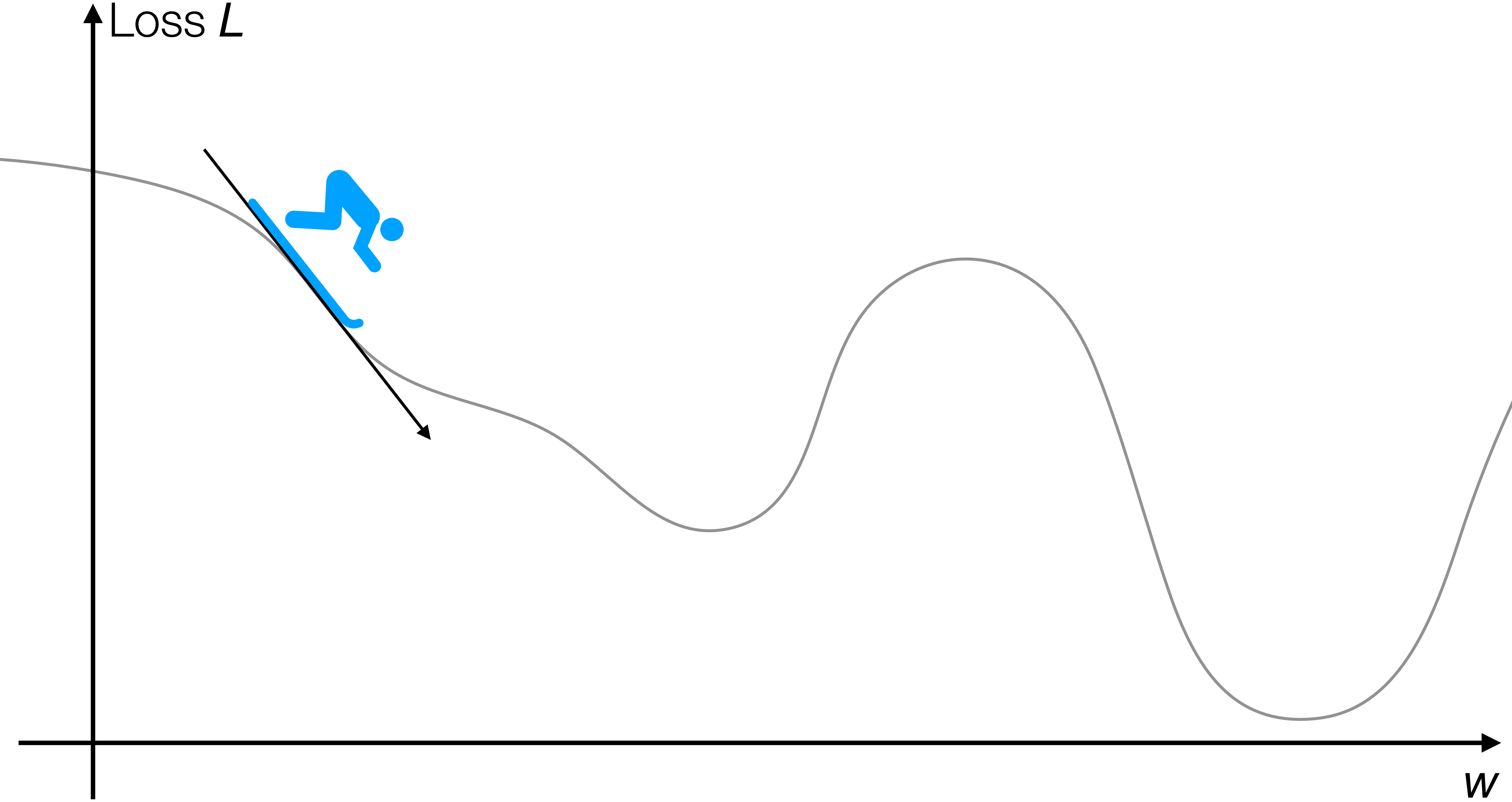
Gradient Descent

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



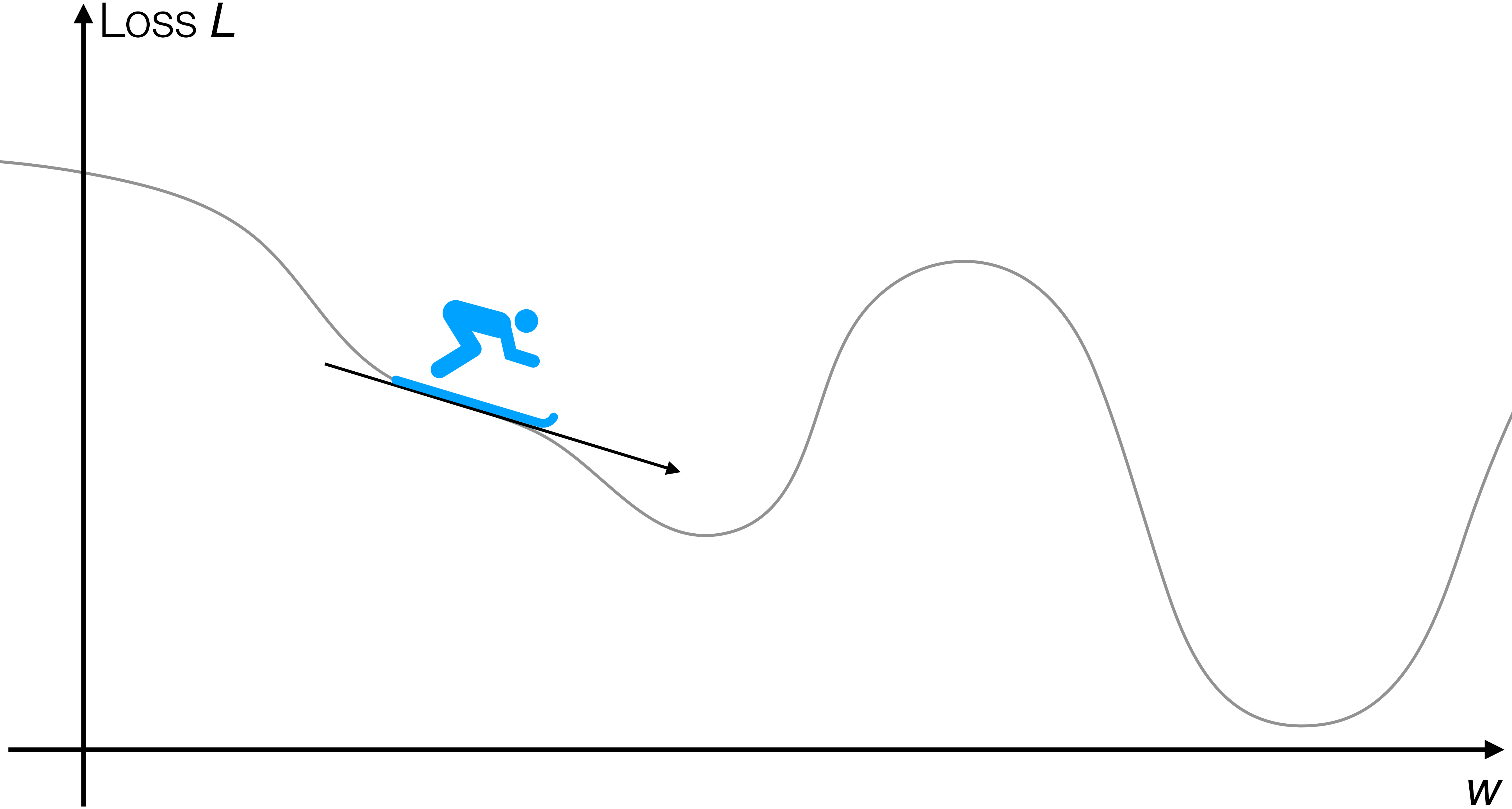
Gradient Descent

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



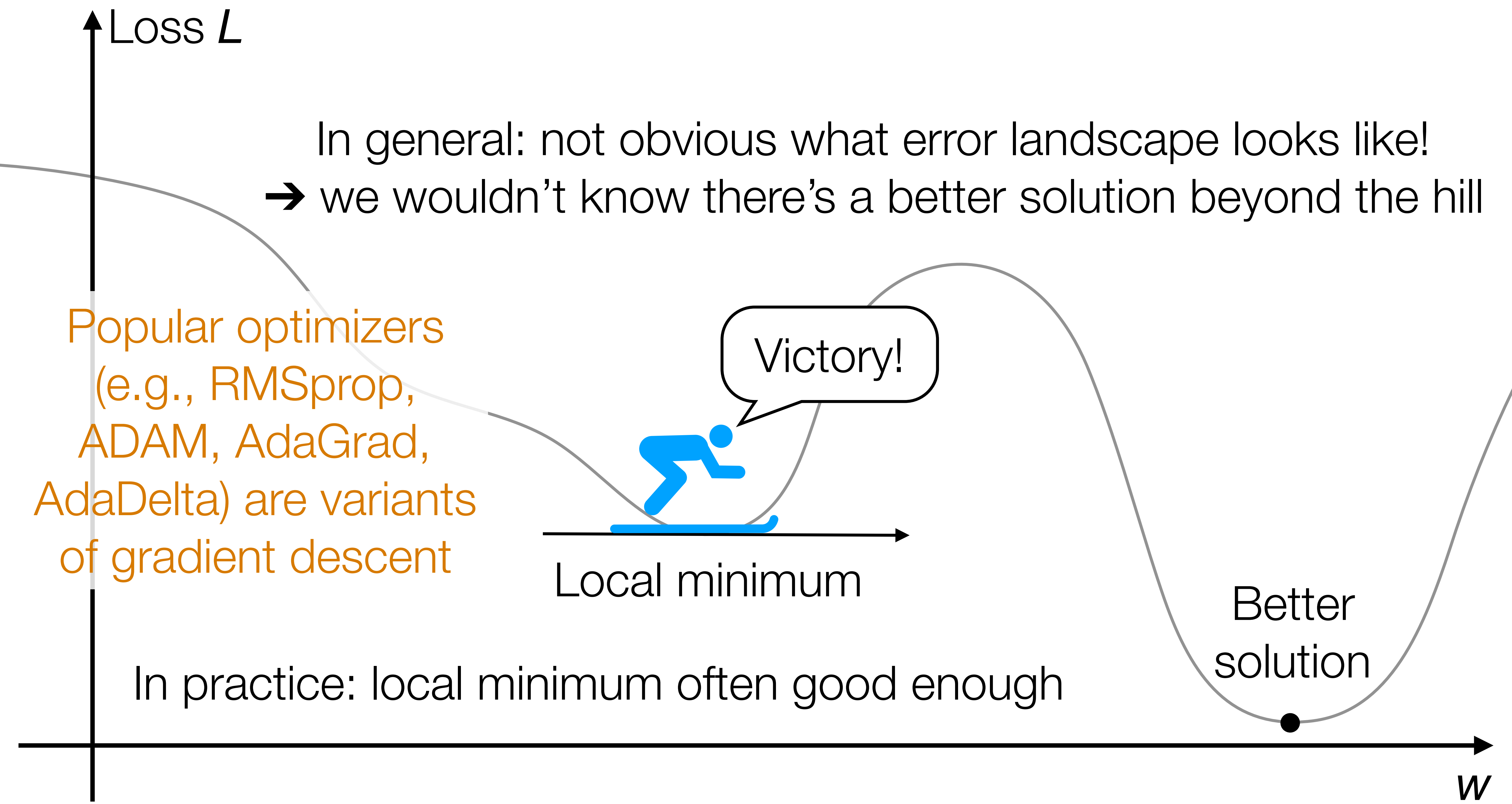
Gradient Descent

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



Gradient Descent

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



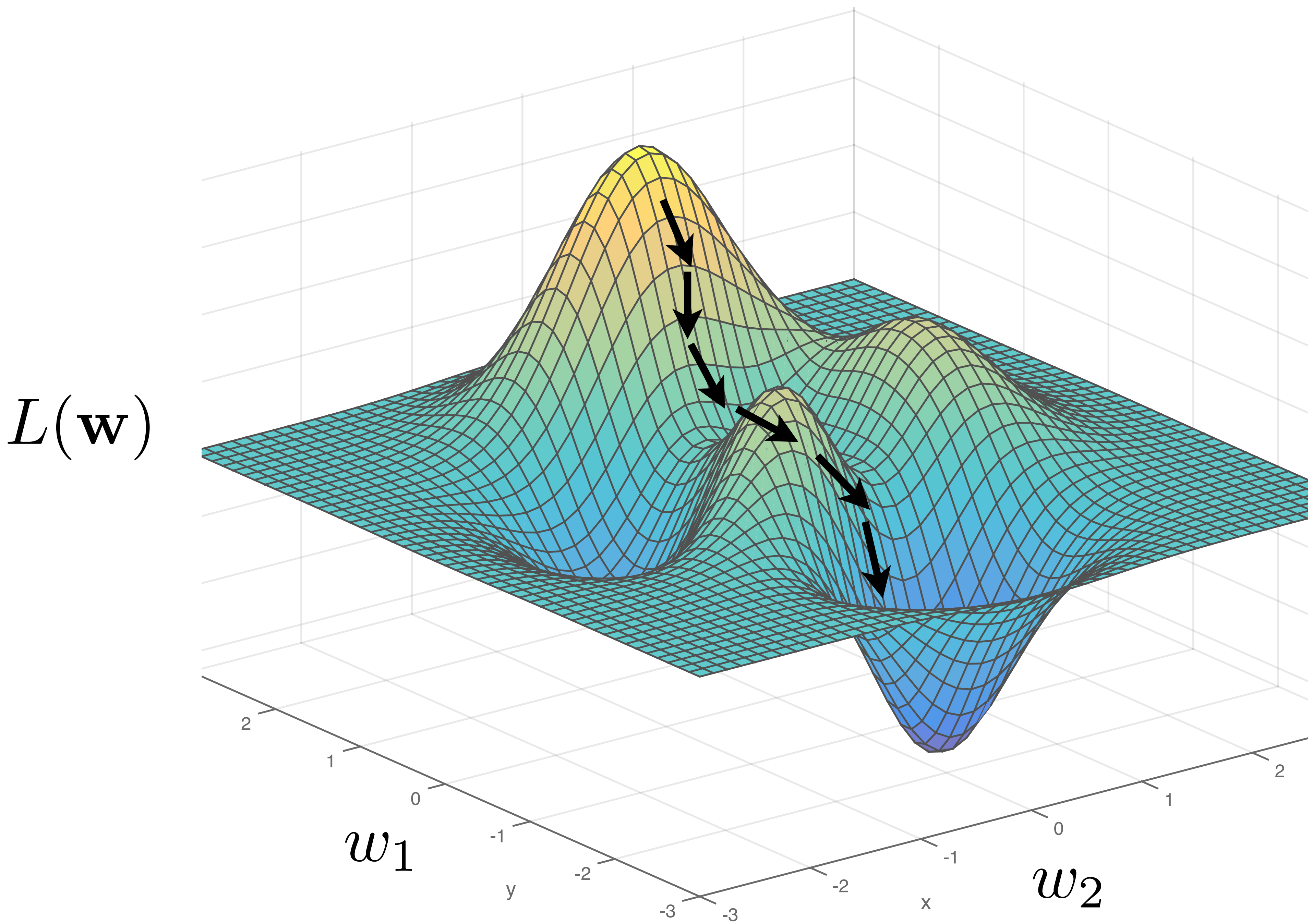
In general: not obvious what error landscape looks like!
➔ we wouldn't know there's a better solution beyond the hill

Popular optimizers (e.g., RMSprop, ADAM, AdaGrad, AdaDelta) are variants of gradient descent

In practice: local minimum often good enough

Gradient Descent

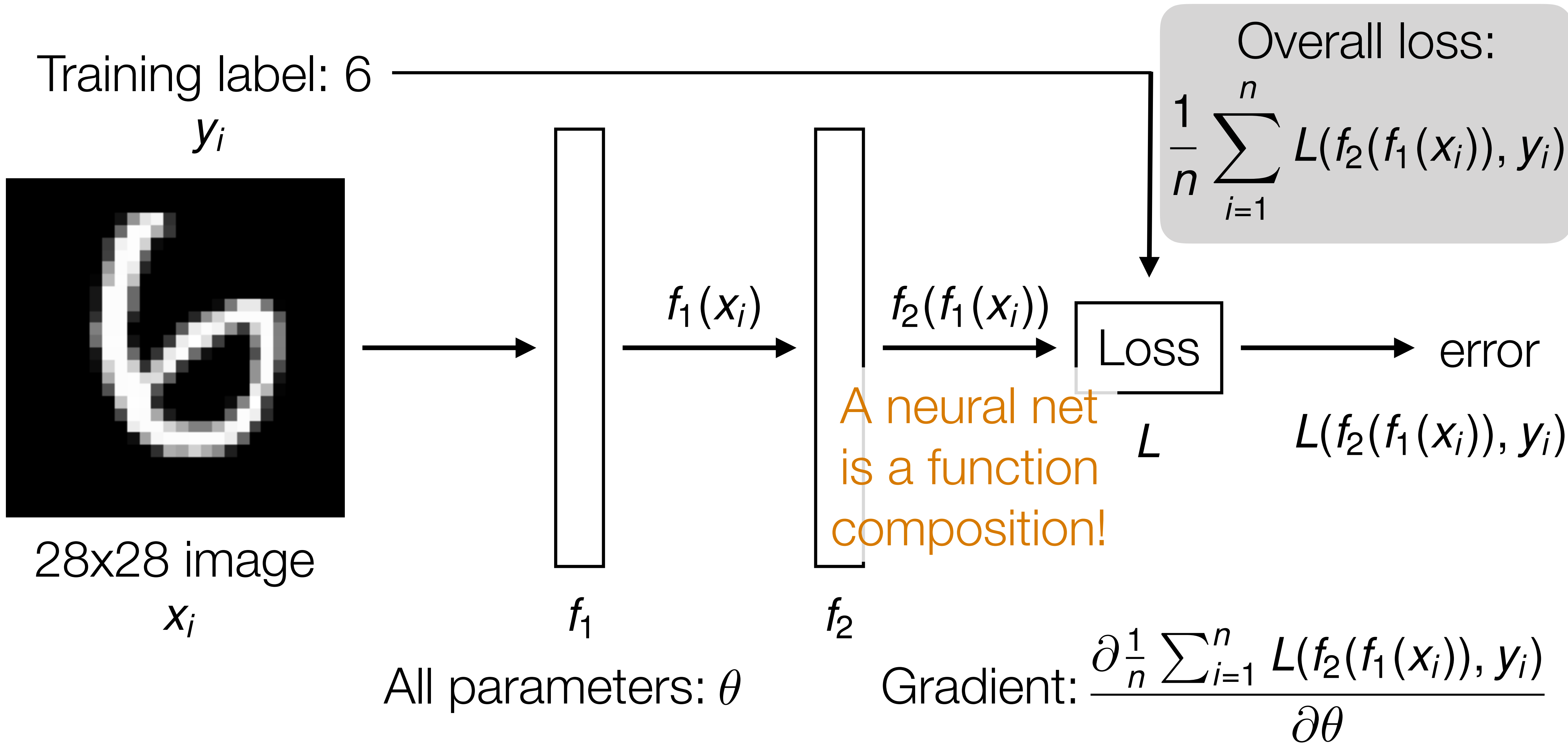
2D example



Slide by Phillip Isola

Remark: In practice, deep nets often have > *millions* of parameters, so *very* high-dimensional gradient descent

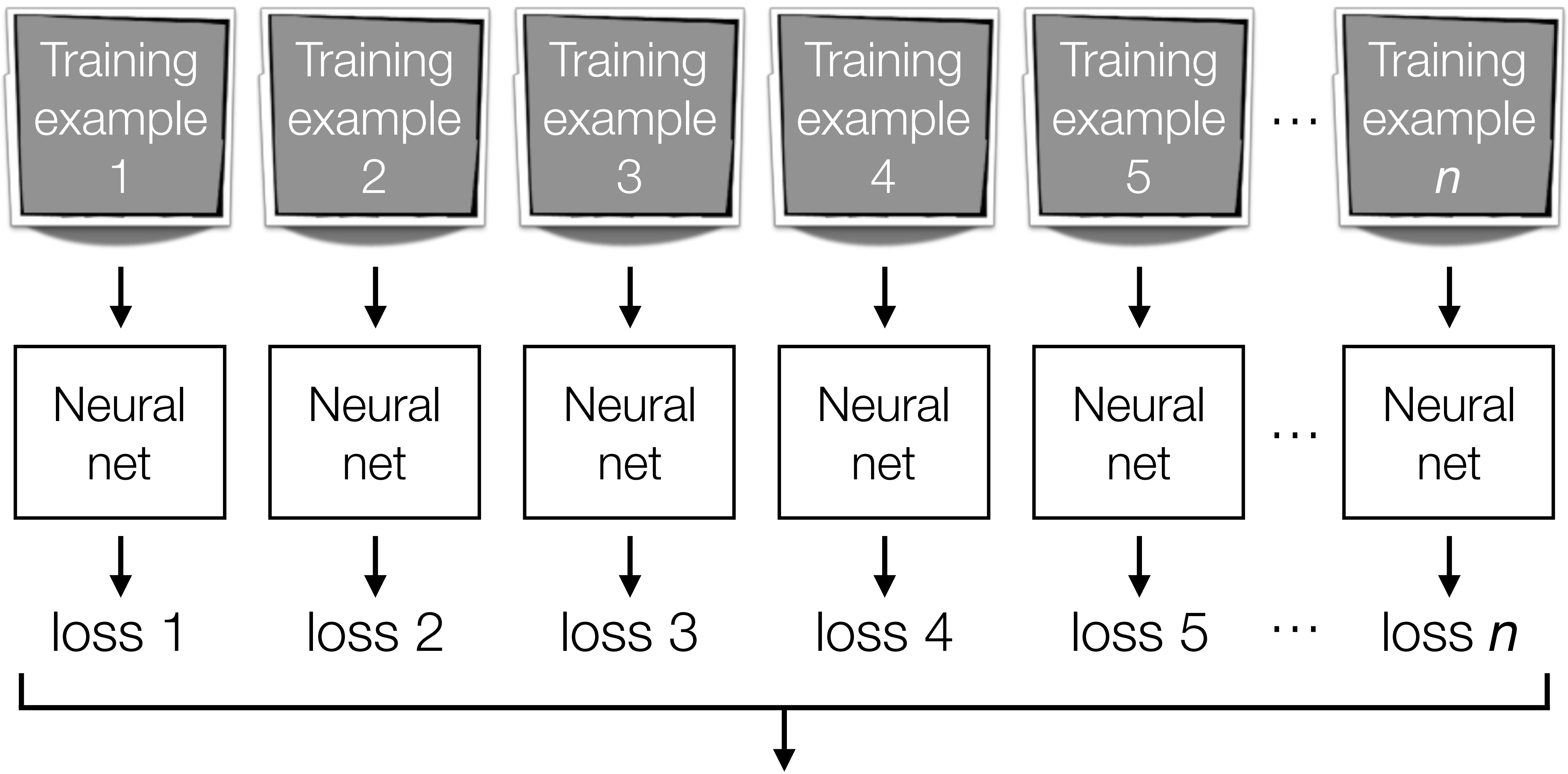
Handwritten Digit Recognition



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 skier know where to go!

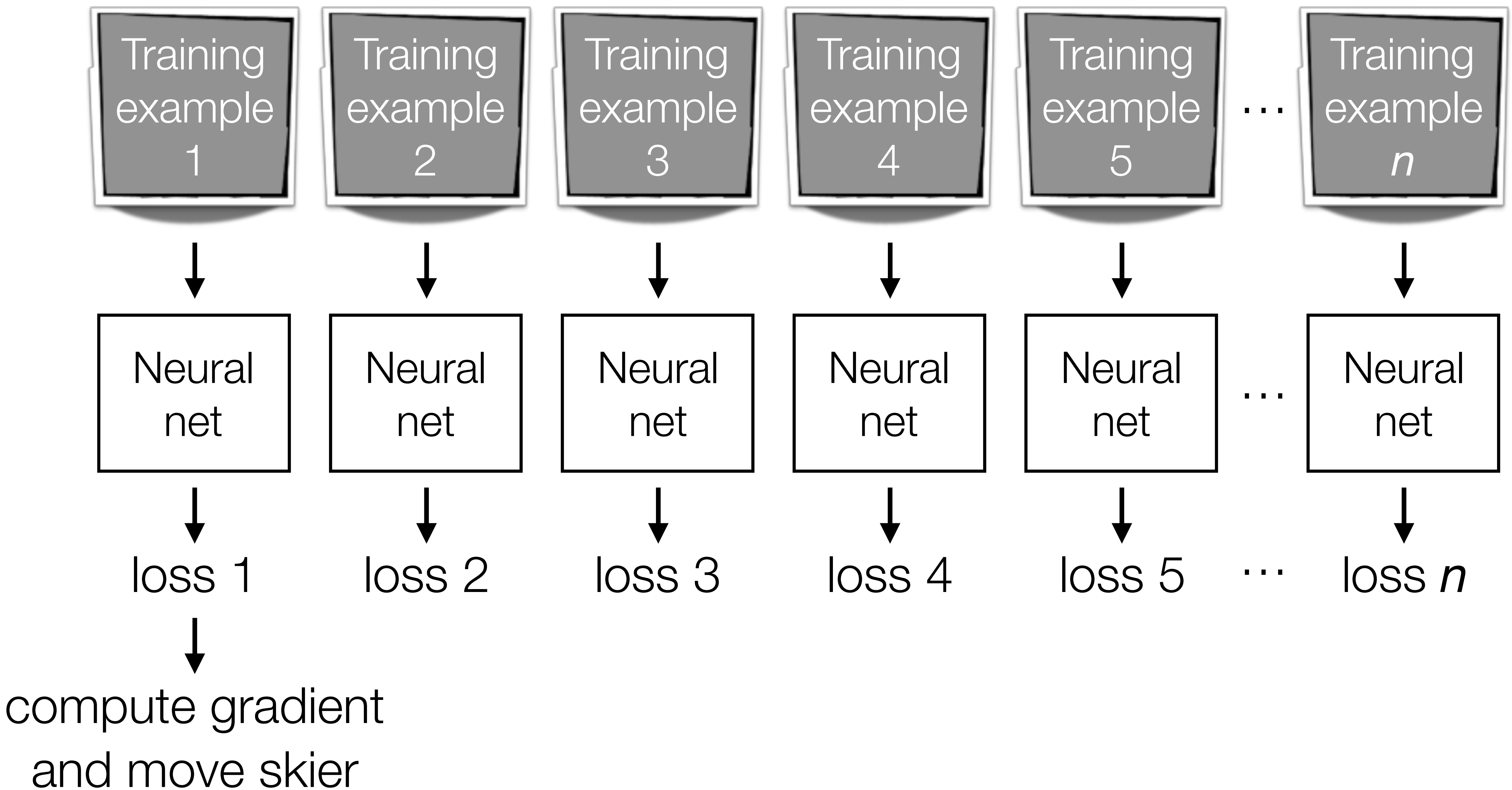
average loss

↓

compute gradient and move skier

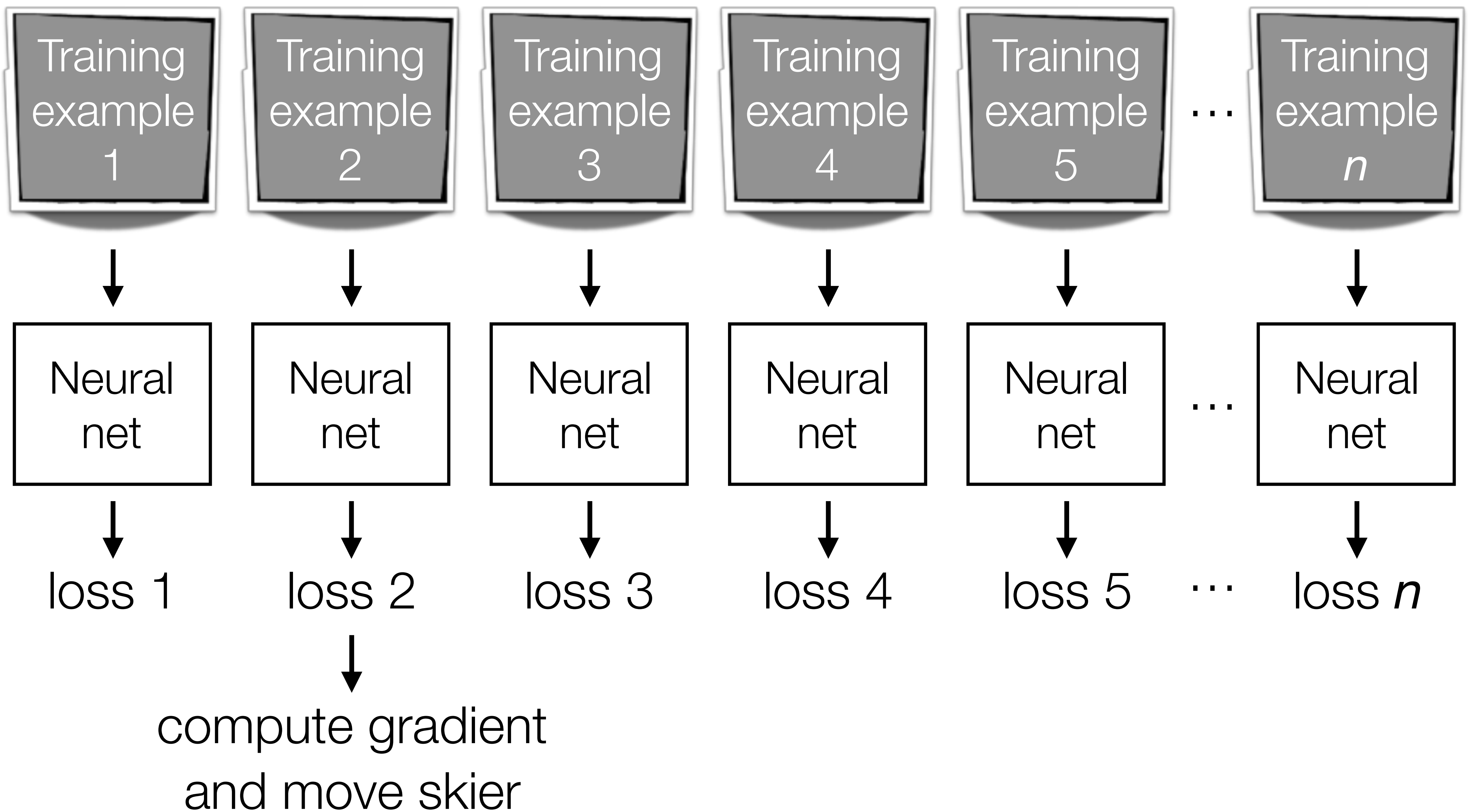
Computing gradients using all the training data seems really expensive!

Stochastic Gradient Descent (SGD)



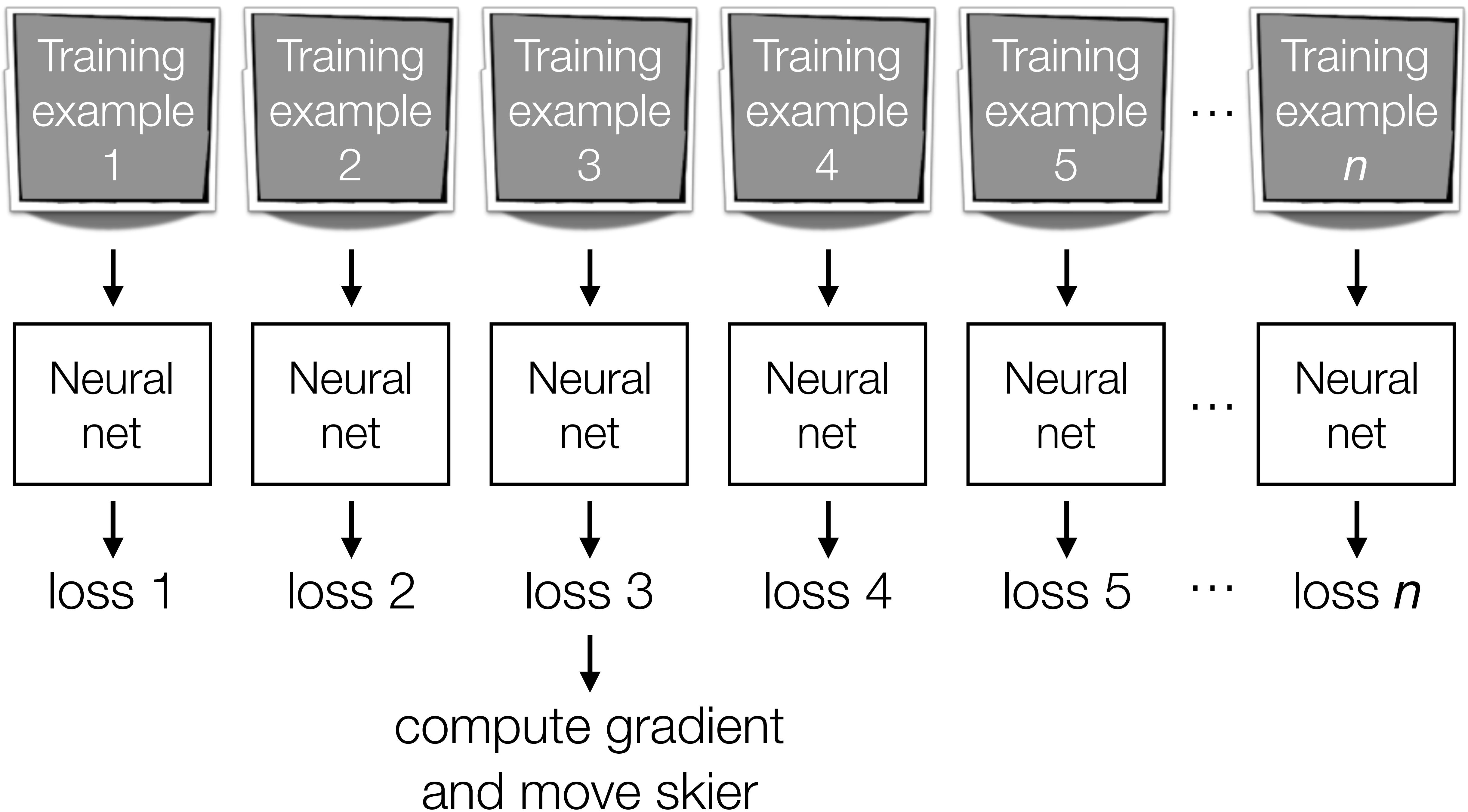
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the “full” gradient)

Stochastic Gradient Descent (SGD)



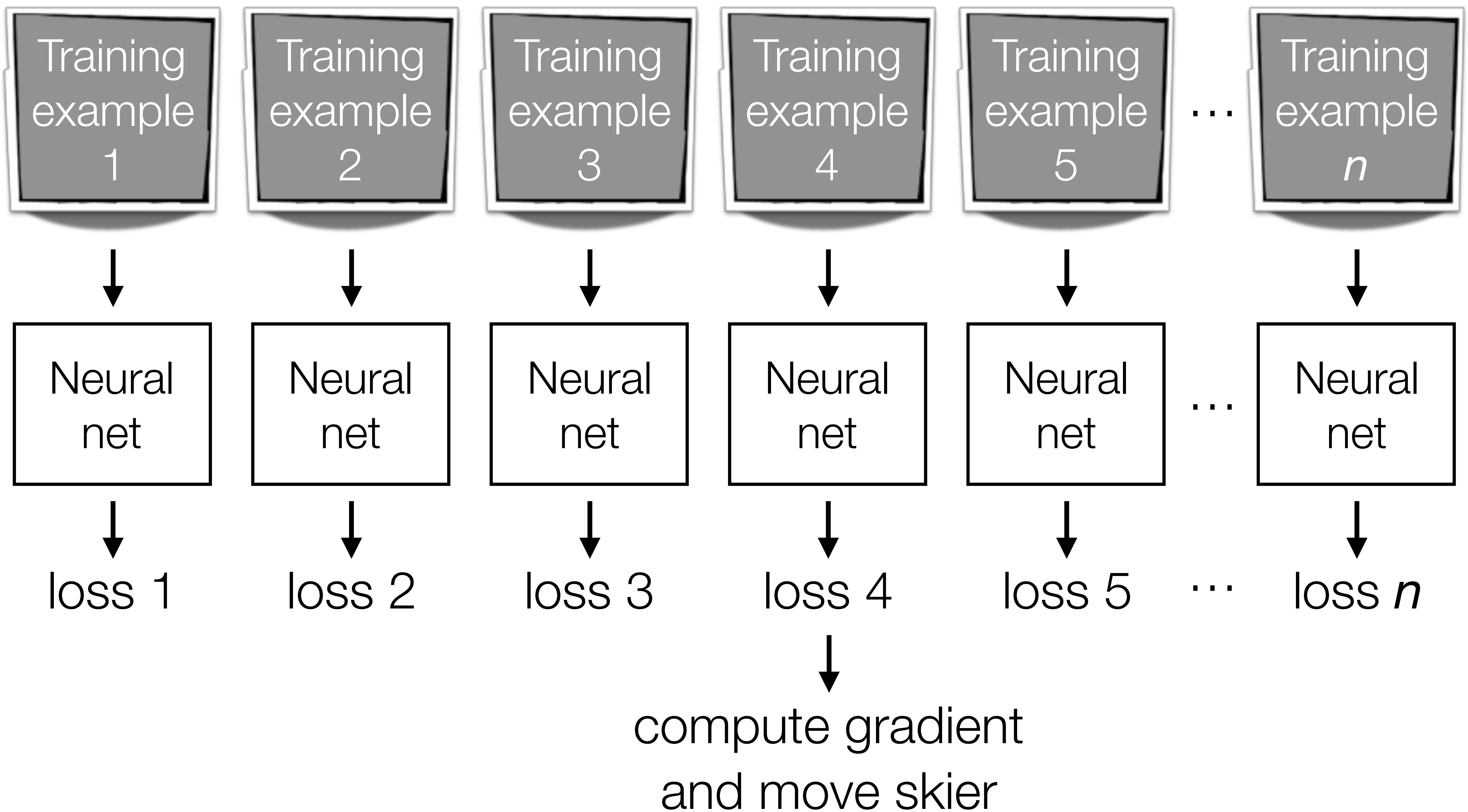
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the “full” gradient)

Stochastic Gradient Descent (SGD)



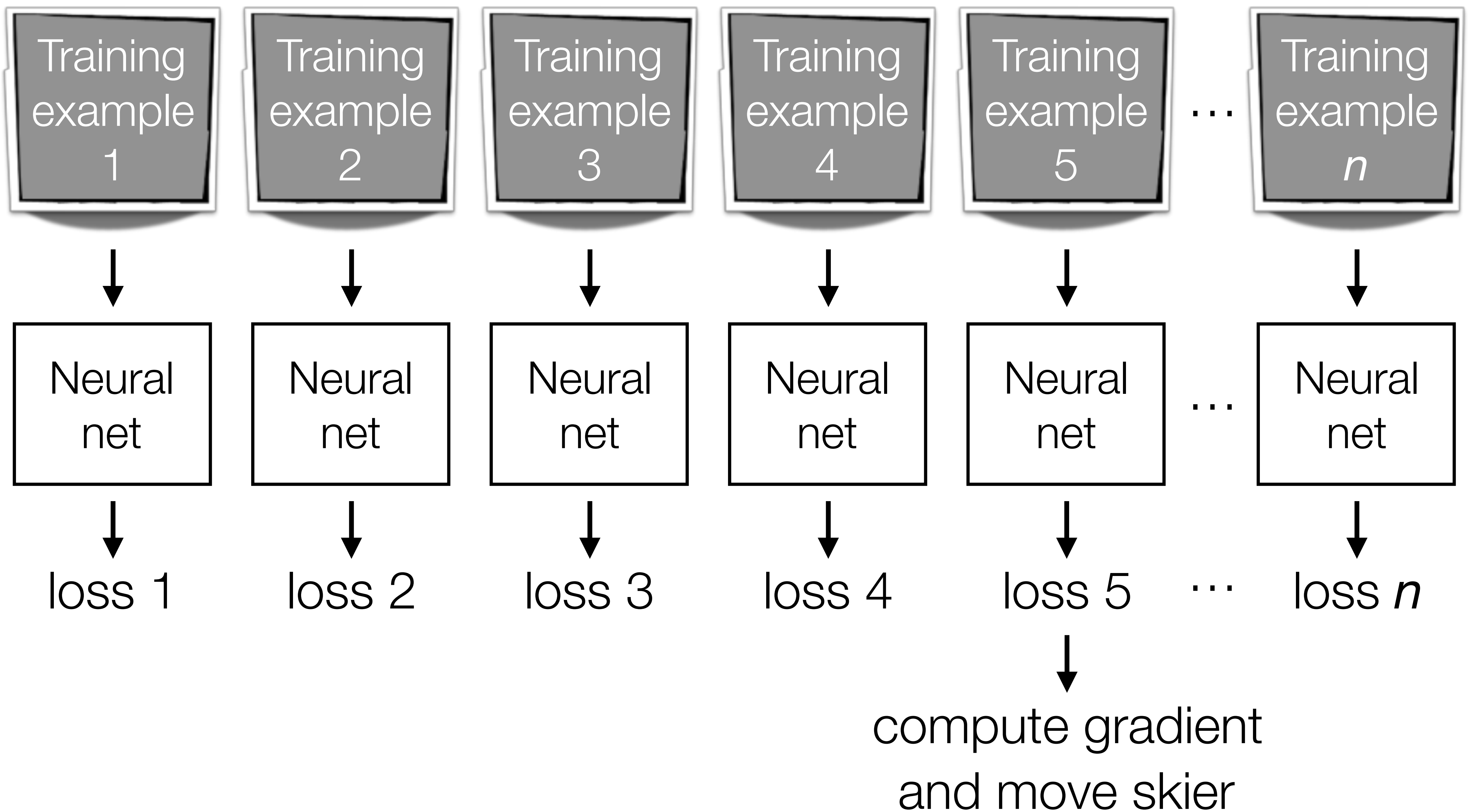
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the “full” gradient)

Stochastic Gradient Descent (SGD)



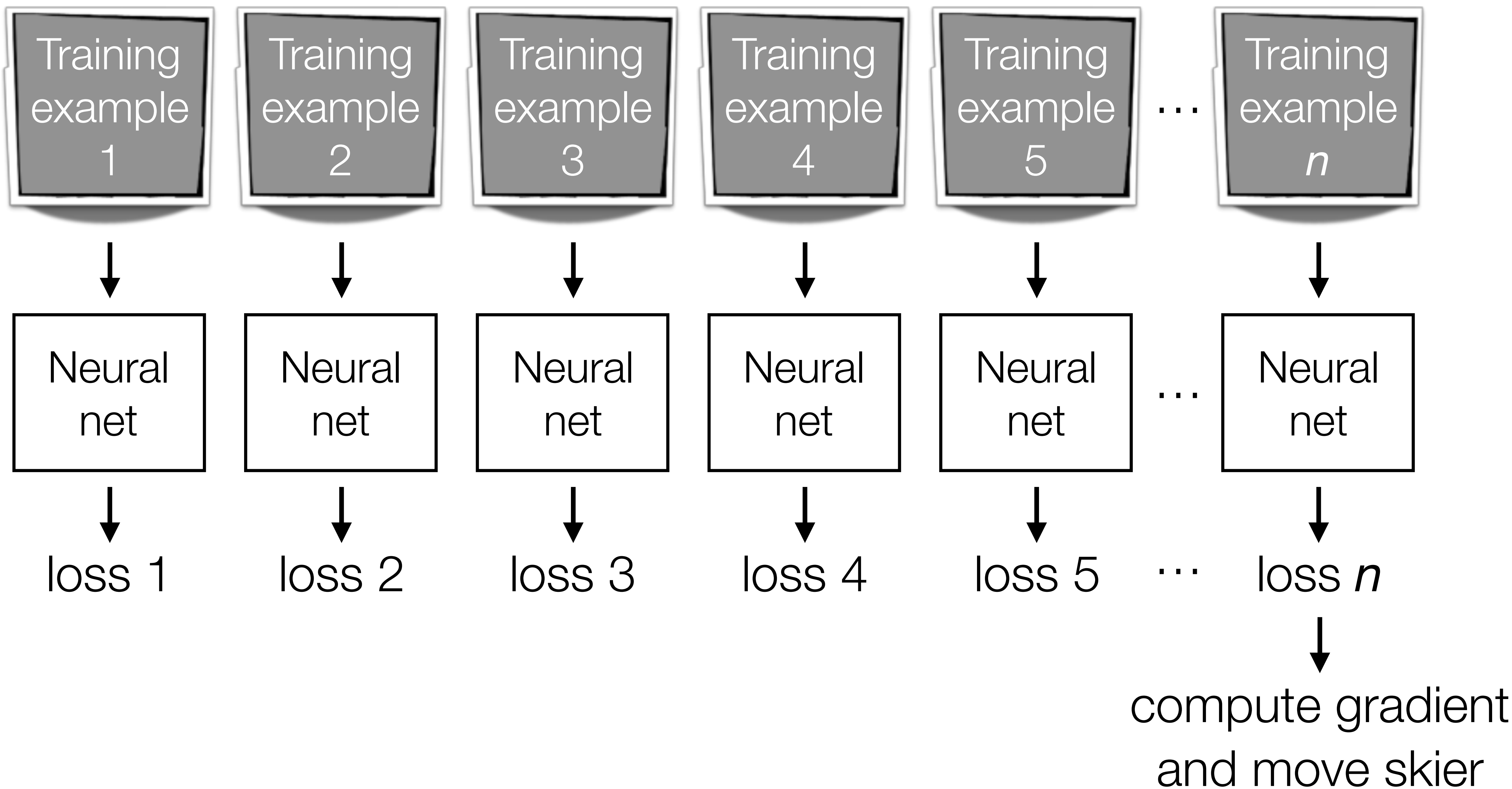
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the “full” gradient)

Stochastic Gradient Descent (SGD)



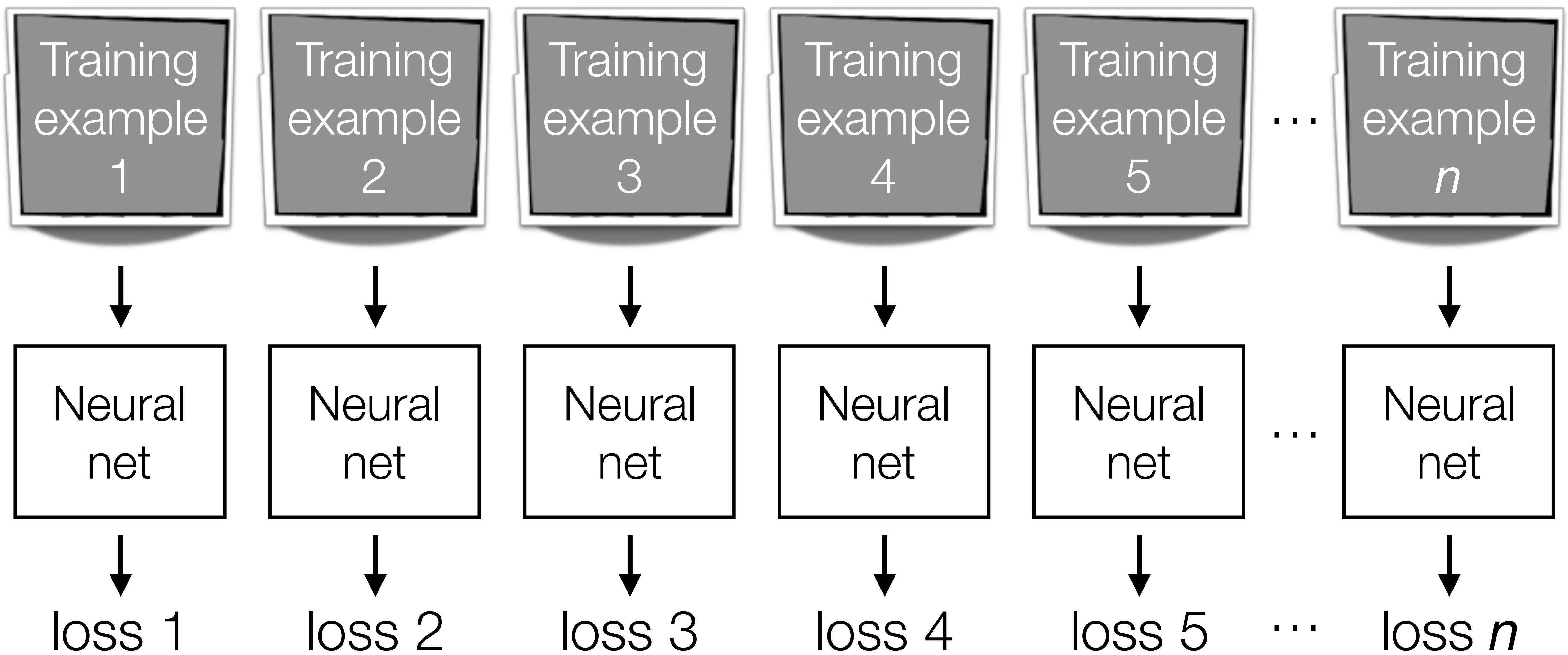
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

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)

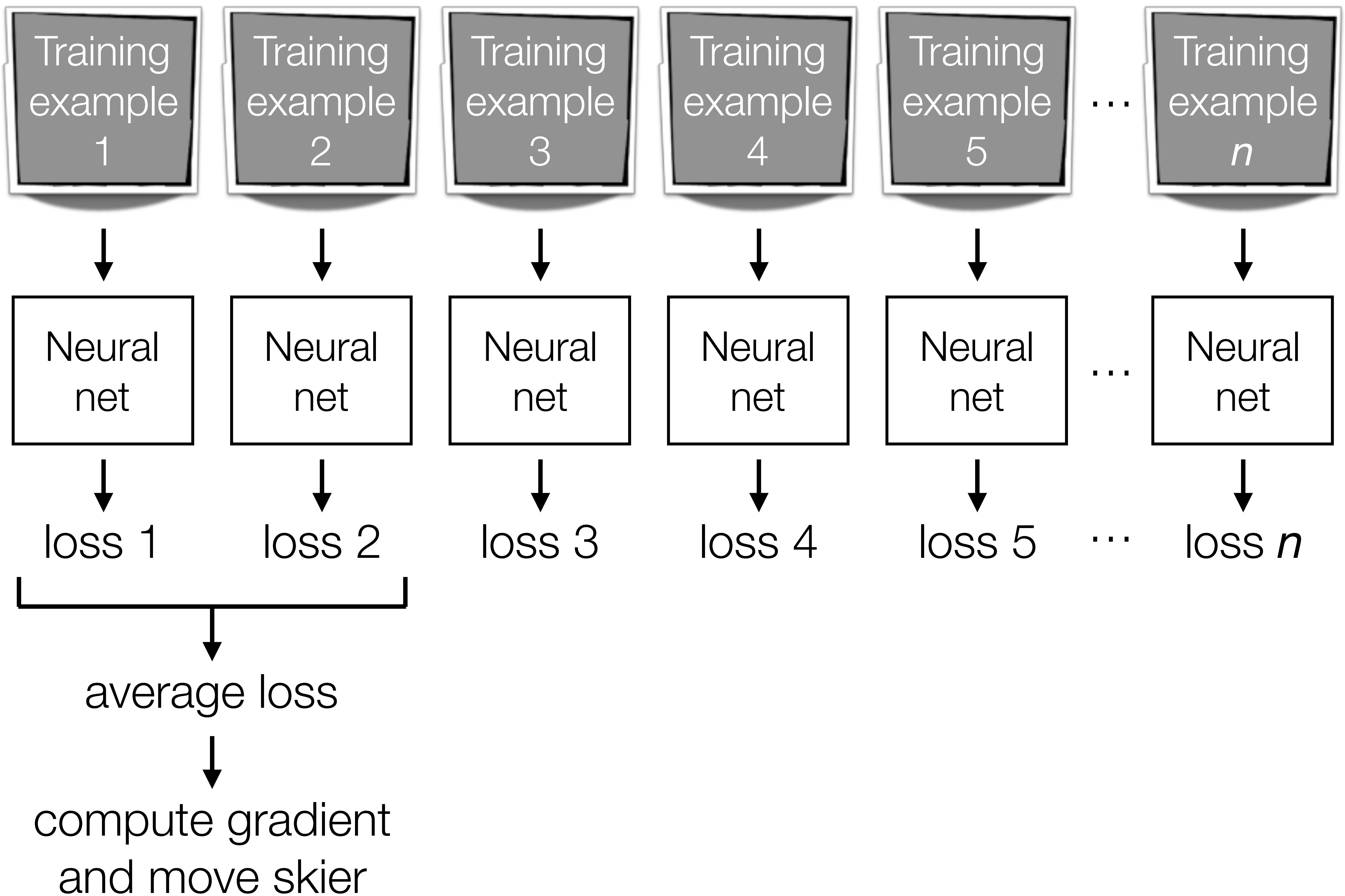


compute gradient
and move skier

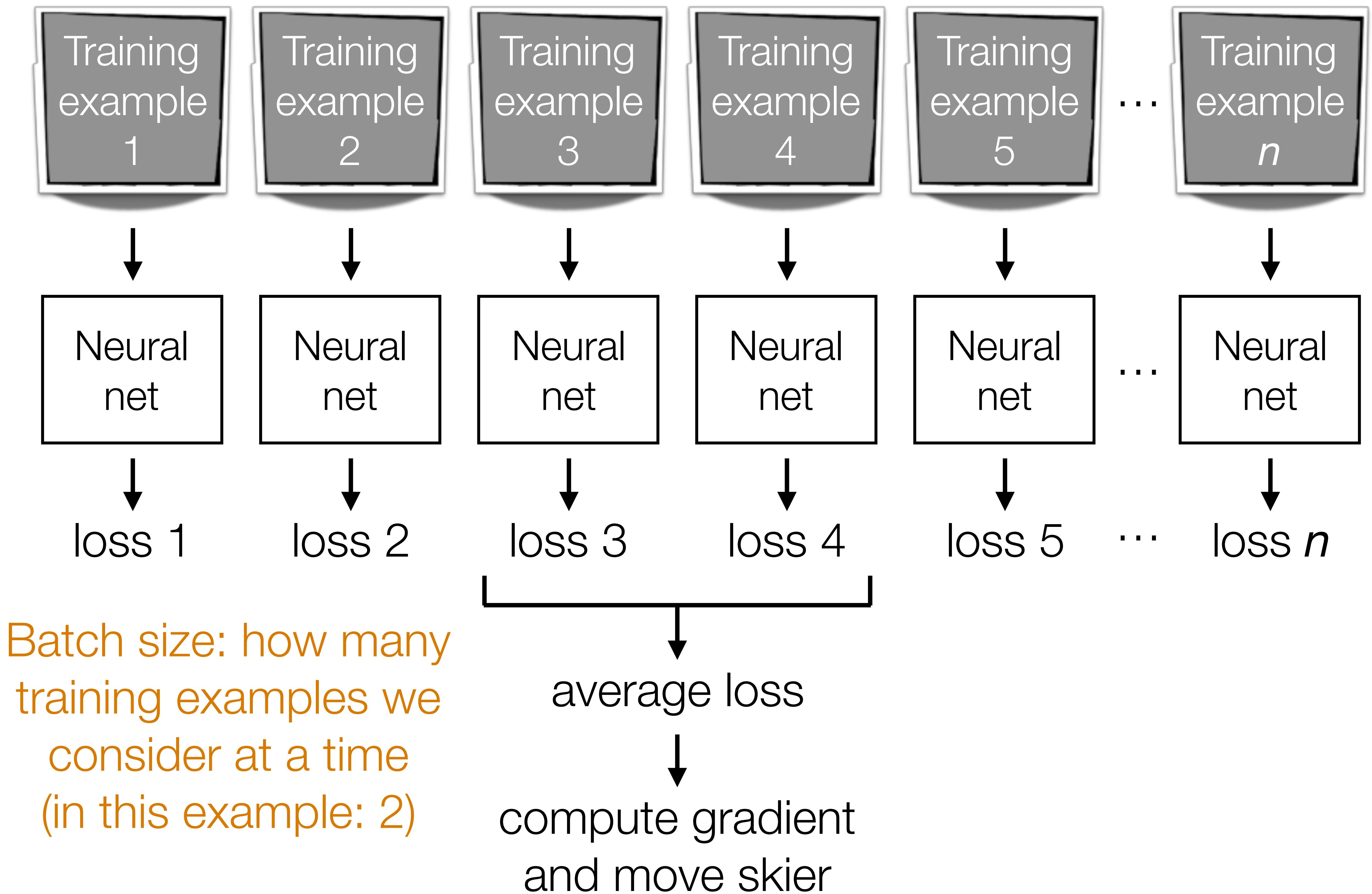
An epoch refers to 1 full pass
through all the training data

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

Mini-Batch Gradient Descent



Mini-Batch Gradient Descent



Best variant of SGD to use?
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!!!