Carnegie Mellon University
HeinzCollege

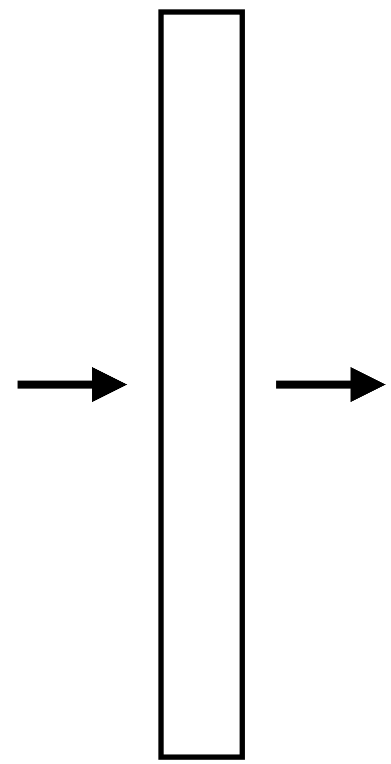
Unstructured Data Analysis

Lecture 13: Time series analysis with recurrent neural nets (RNNs), how learning a neural net works, dealing with small datasets, course wrap-up

George Chen

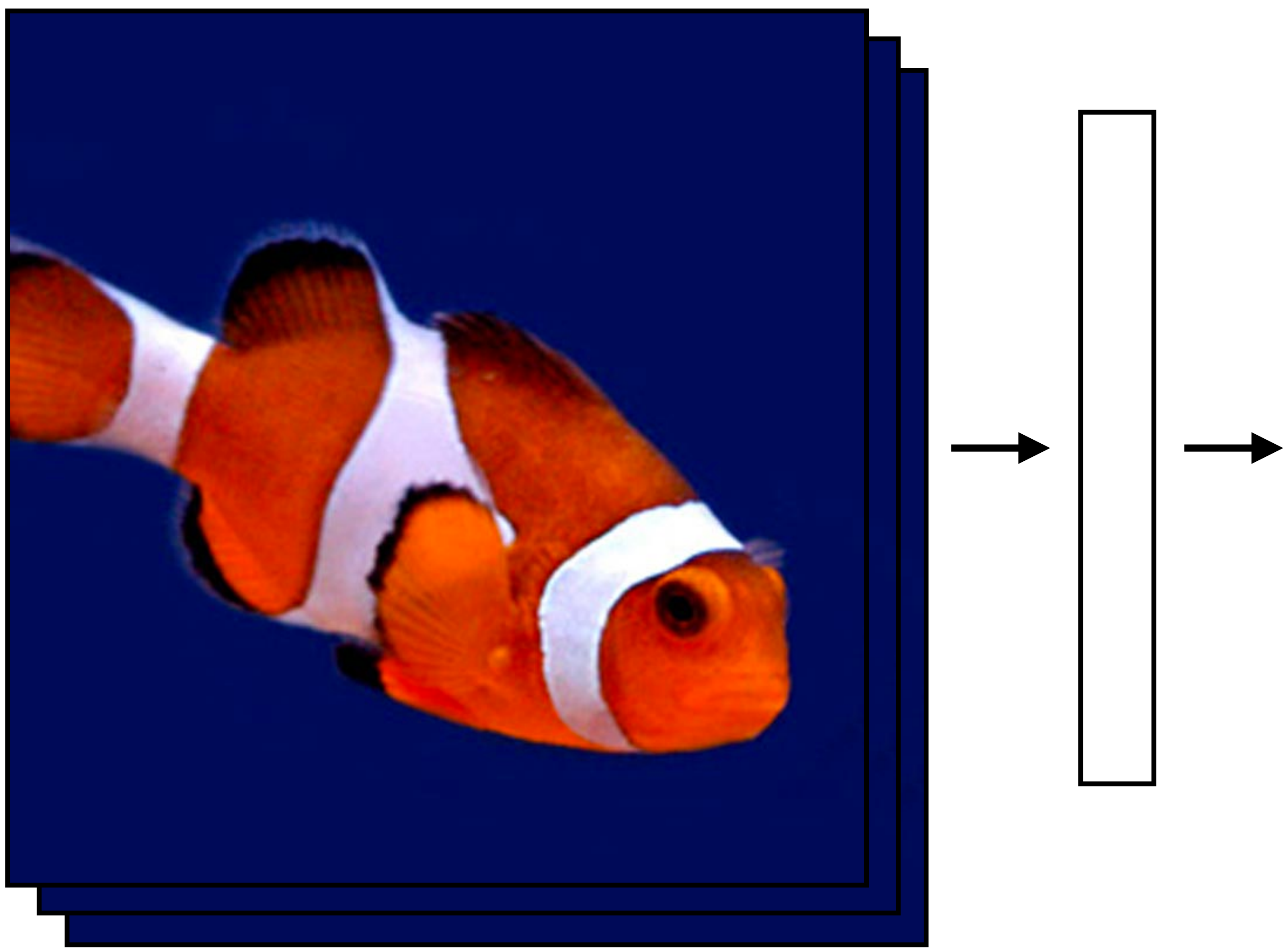
Time Series ("Sequential") Data

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



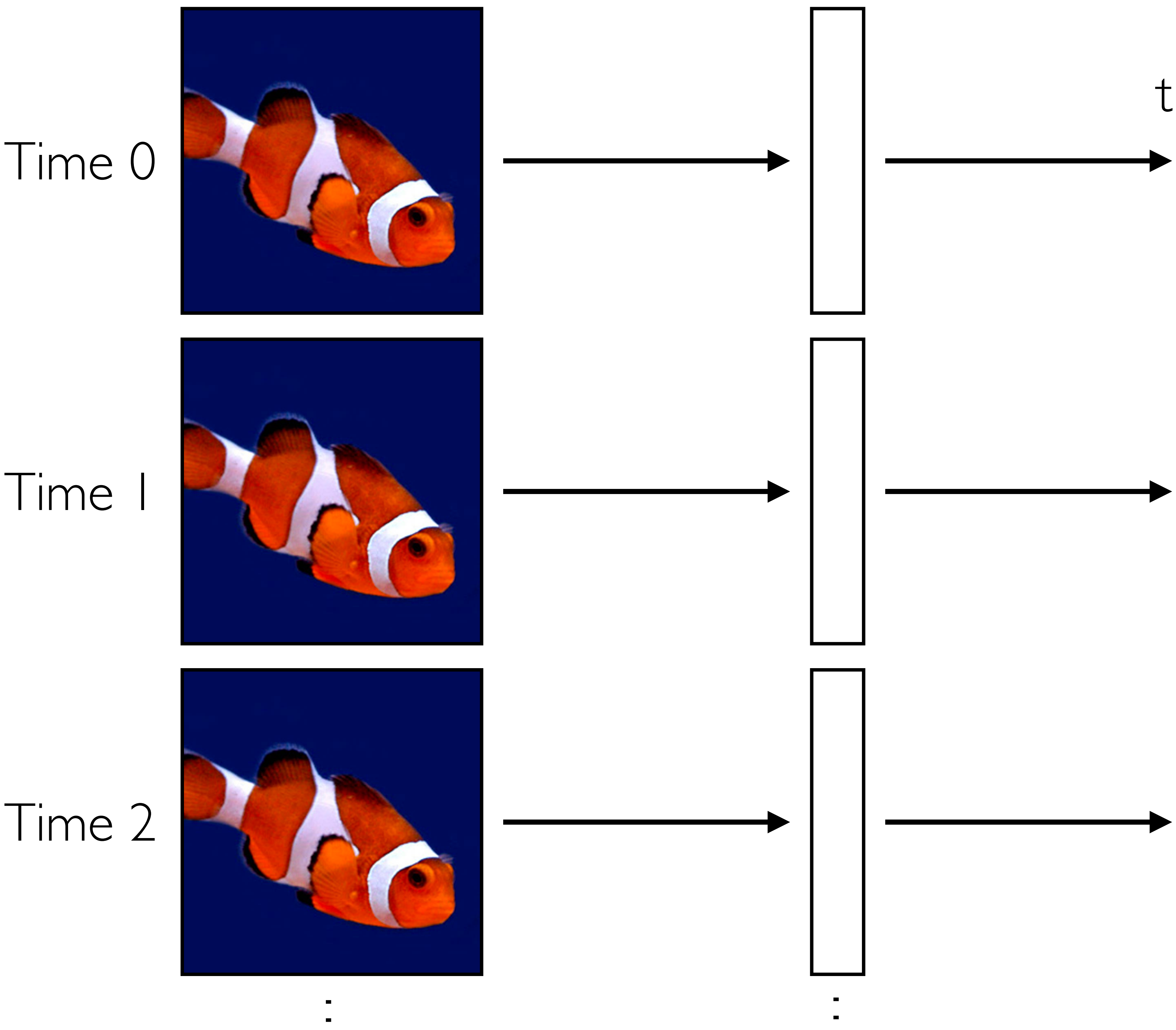
Time Series (“Sequential”) Data

What we’ve seen so far are “feedforward” NNs

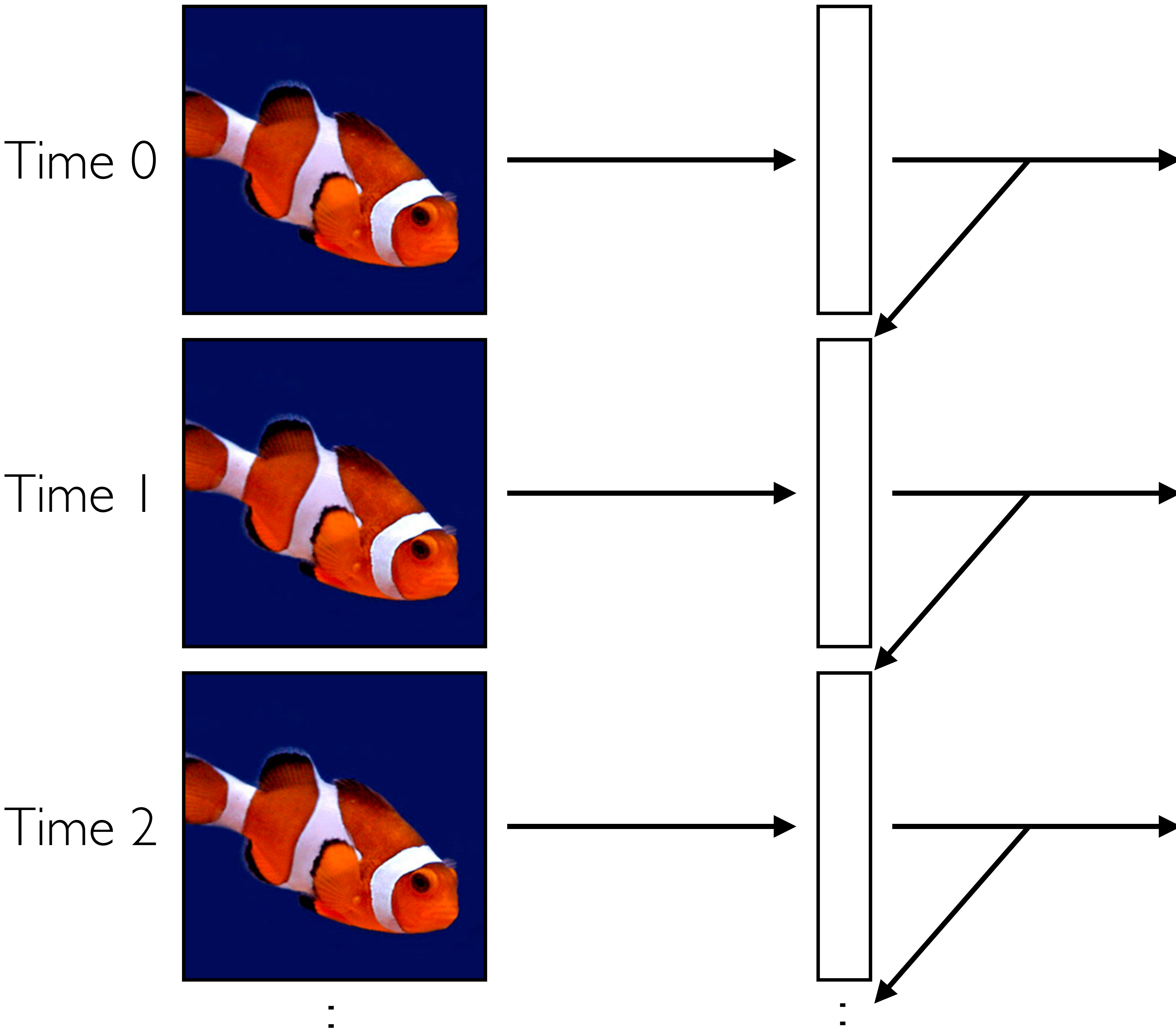


What if we had a video?

Feedforward NN's:
treat each video frame
separately



Recurrent Neural Nets



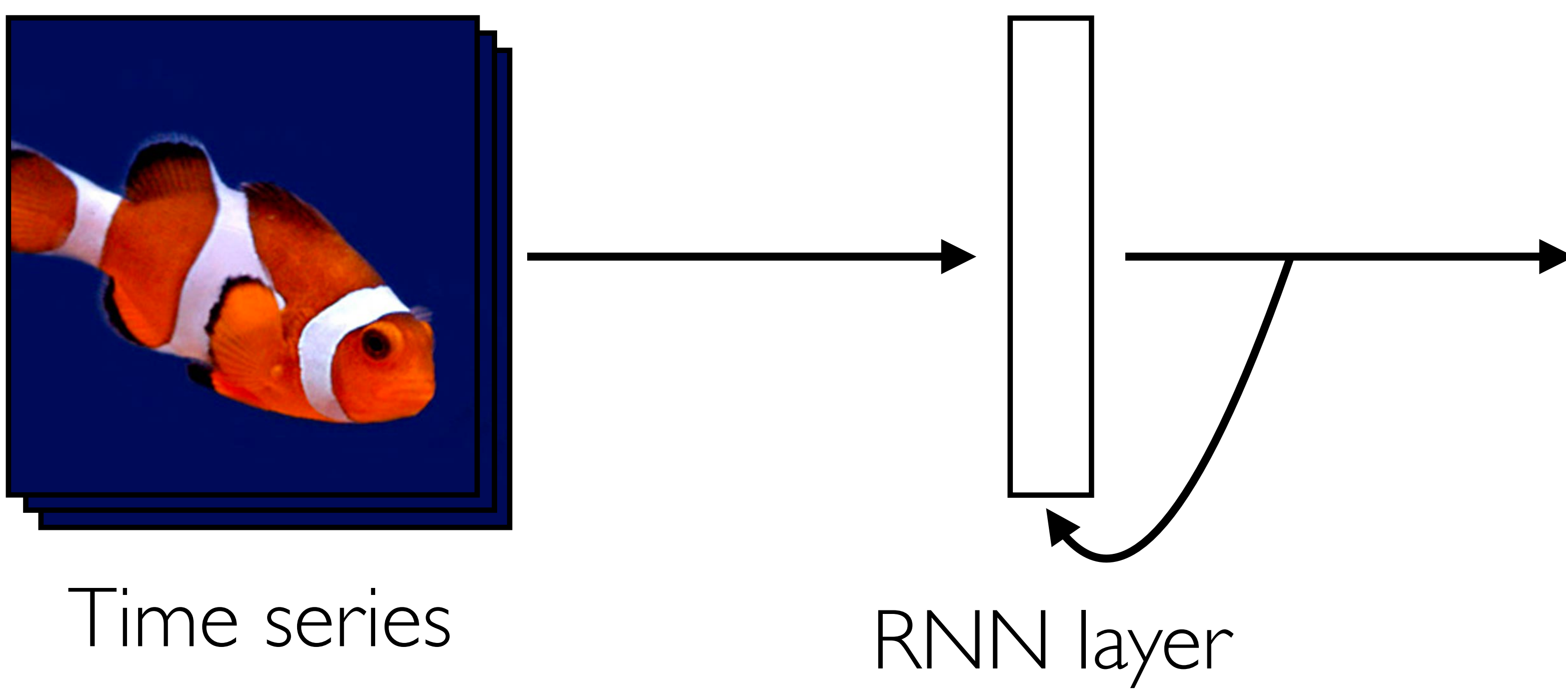
Feedforward NN's:
treat each video frame separately

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

In PyTorch, different RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Recurrent Neural Nets

Feedforward NN's:
treat each video frame separately



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

In PyTorch, different RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Vanilla ReLU RNN

memory that evolves over time; we want to learn how it changes

```
current_state = np.zeros(num_nodes)

for input in input_sequence:

    linear = np.dot(input, W) \
             + np.dot(current_state, U) \
             + b

    output = np.maximum(0, linear) # ReLU

    current_state = output
```

W is a 2D table: # rows: (length of single time step's input), # cols: num_nodes

U is a 2D table: num_nodes by num_nodes

b is a 1D table: num_nodes entries

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

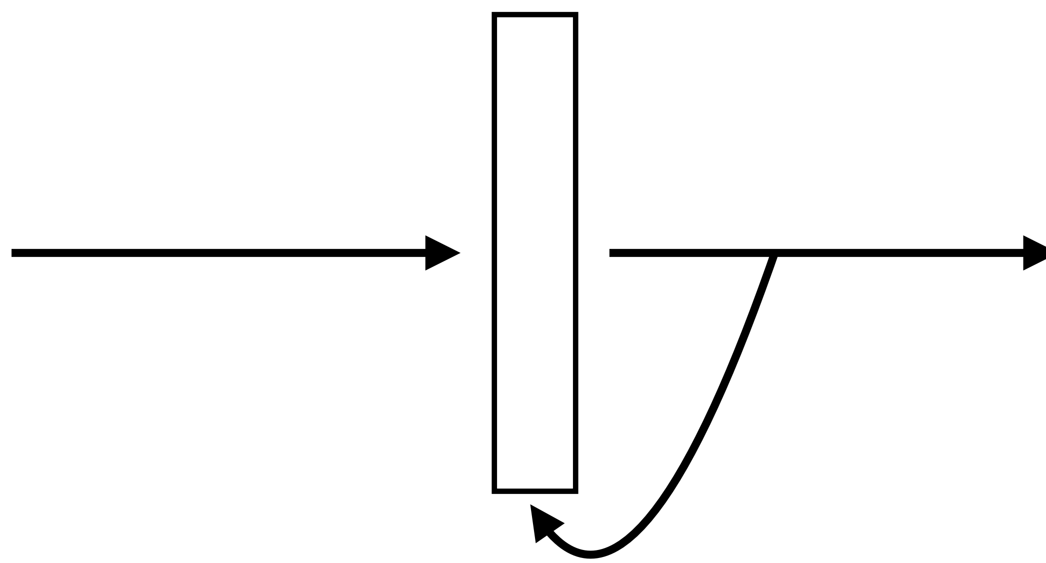
Key idea: **it's like a linear layer in a for loop that tracks how memory changes over time**

Recurrent Neural Nets

Feedforward NN's:
treat each video frame separately

readily chains together with other neural net layers

Time series



RNN layer

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

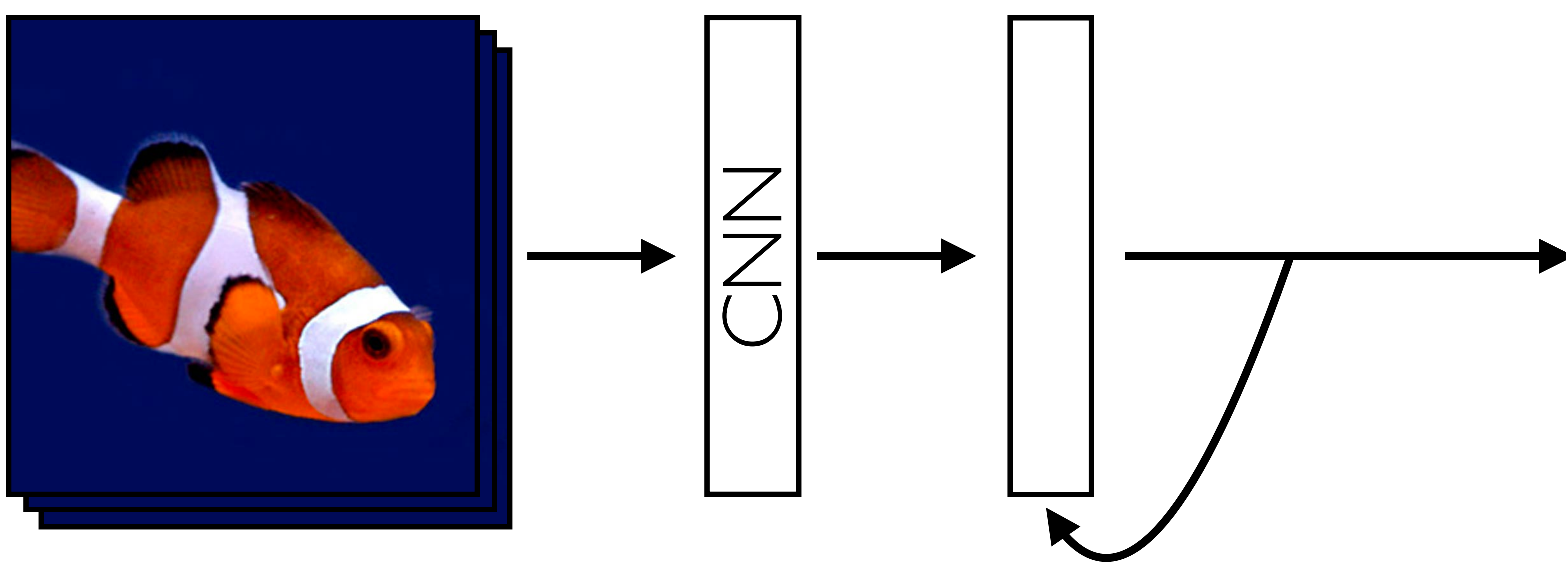
like a linear layer that has memory

does not incorporate image structure!!!

In PyTorch, different RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Recurrent Neural Nets

readily chains together with
other neural net layers



Time series

RNN layer

like a linear layer
that has memory

does not incorporate
image structure!!!

Feedforward NN's:
treat each video frame
separately

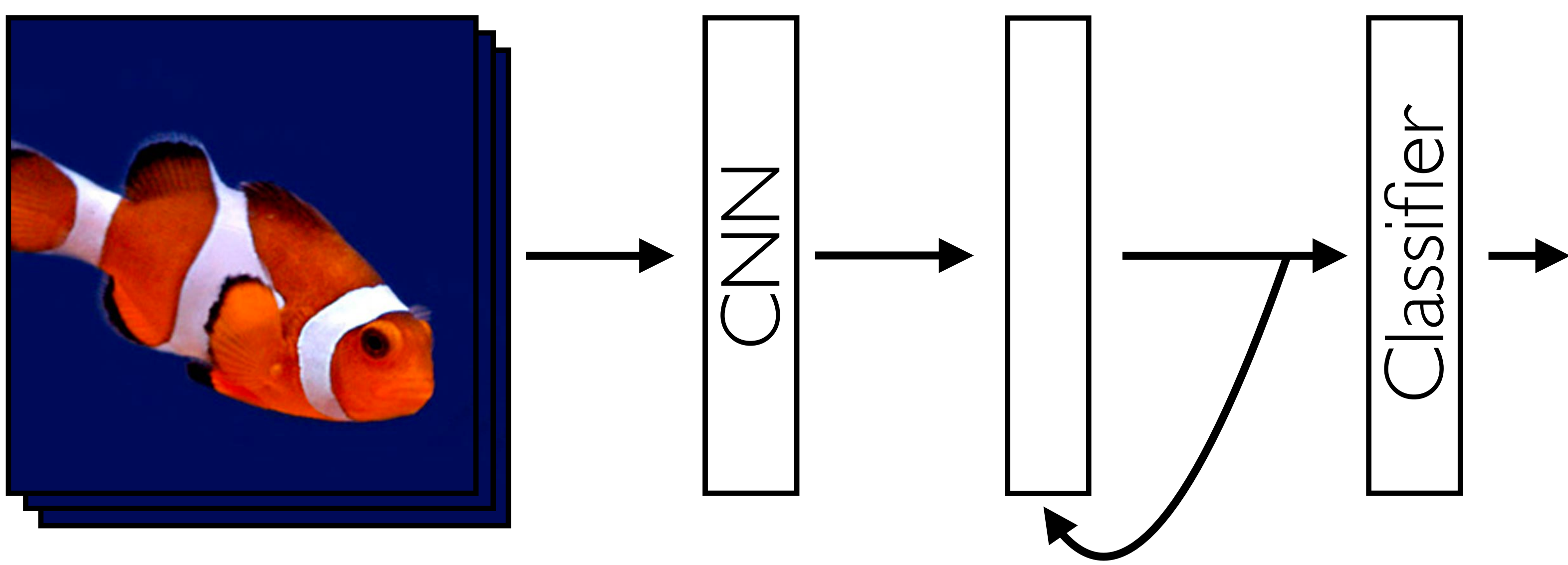
RNNs:
feed output at previous
time step as input to
RNN layer at current
time step

In PyTorch, different
RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Recurrent Neural Nets

Feedforward NN's:
treat each video frame separately

readily chains together with other neural net layers



Time series

RNN layer

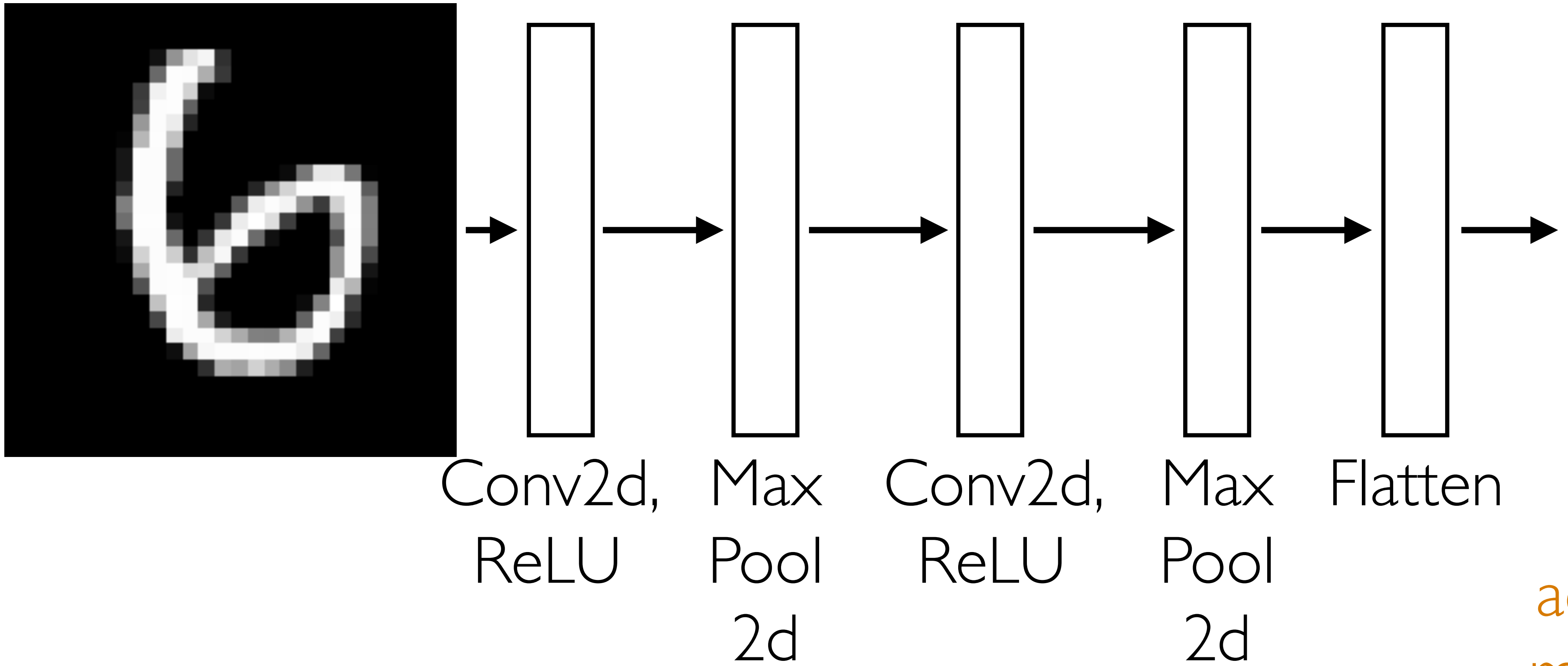
like a linear layer that has memory

does not incorporate image structure!!!

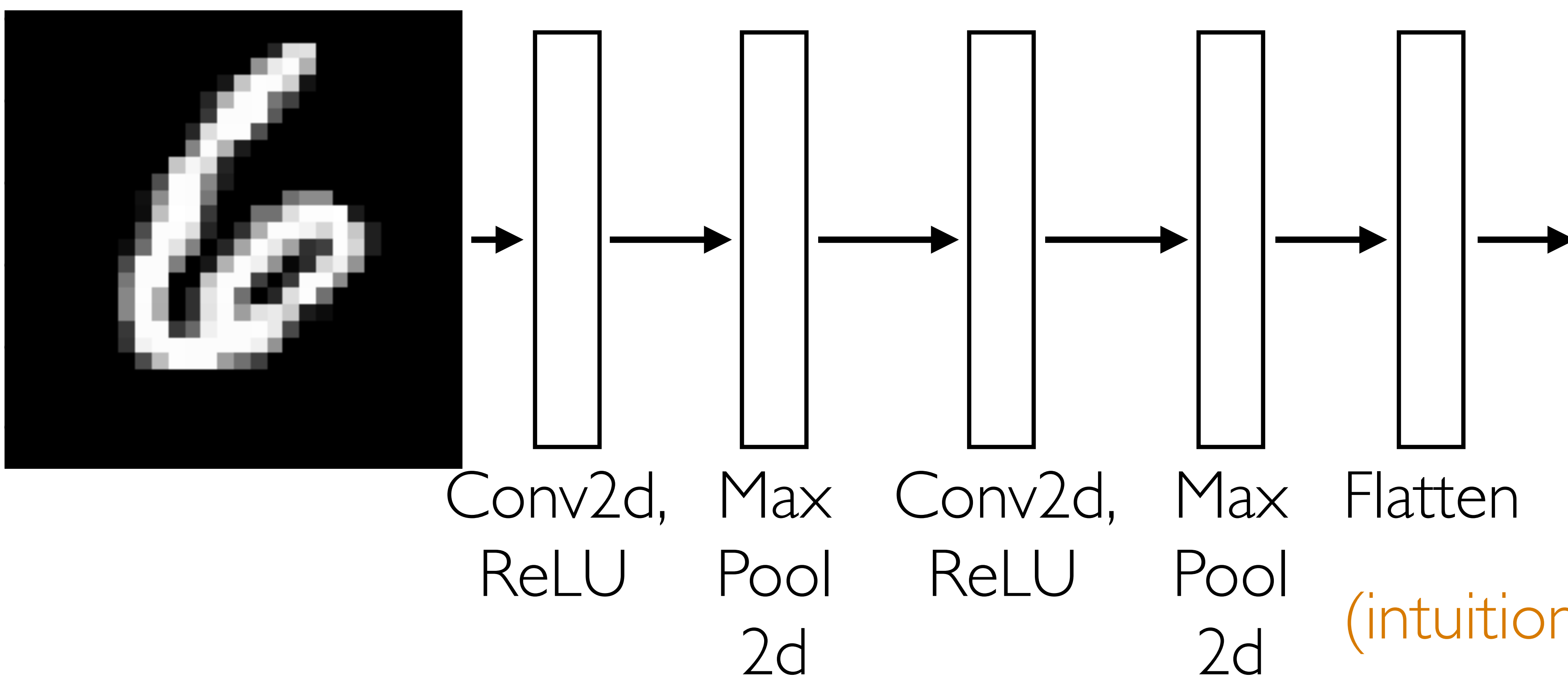
Use CNN to incorporate image structure!

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

In PyTorch, different RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`



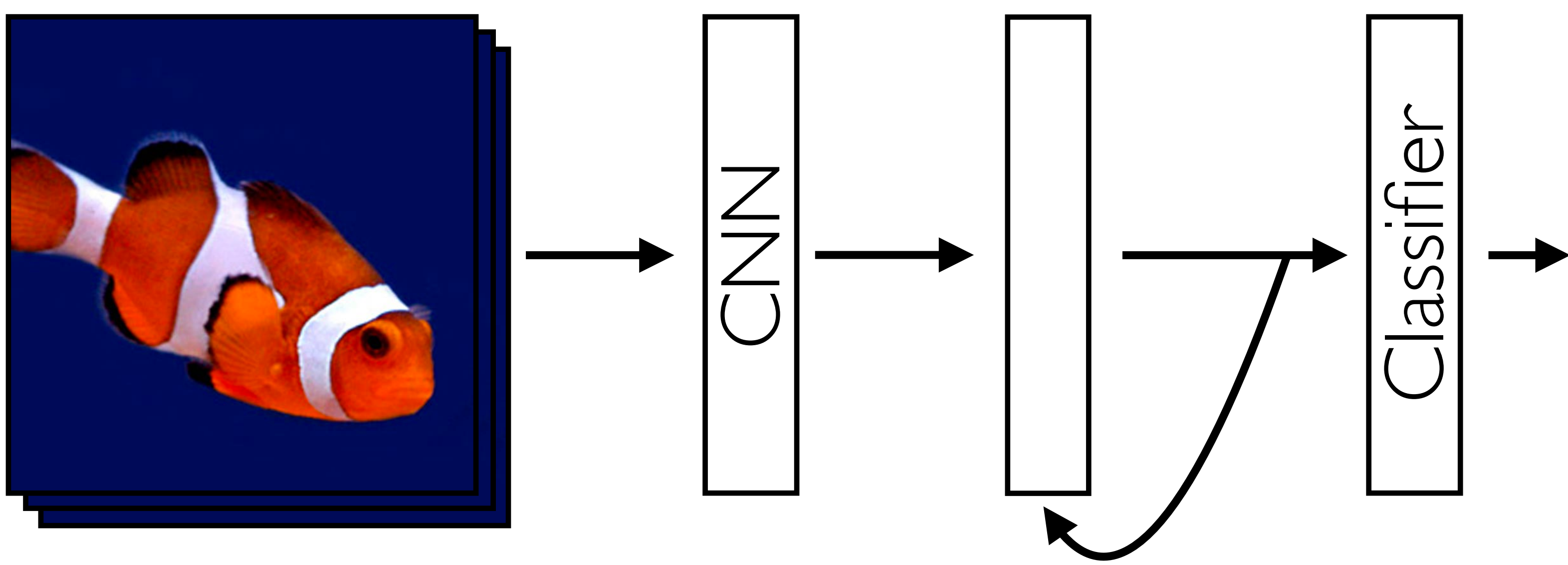
actually, intermediate representations close to the last layer are also similar!



(intuition: recall the crumpled paper analogy!)

Recurrent Neural Nets

readily chains together with
other neural net layers



Time series

RNN layer

Use CNN to
incorporate image
structure!

like a linear layer
that has memory

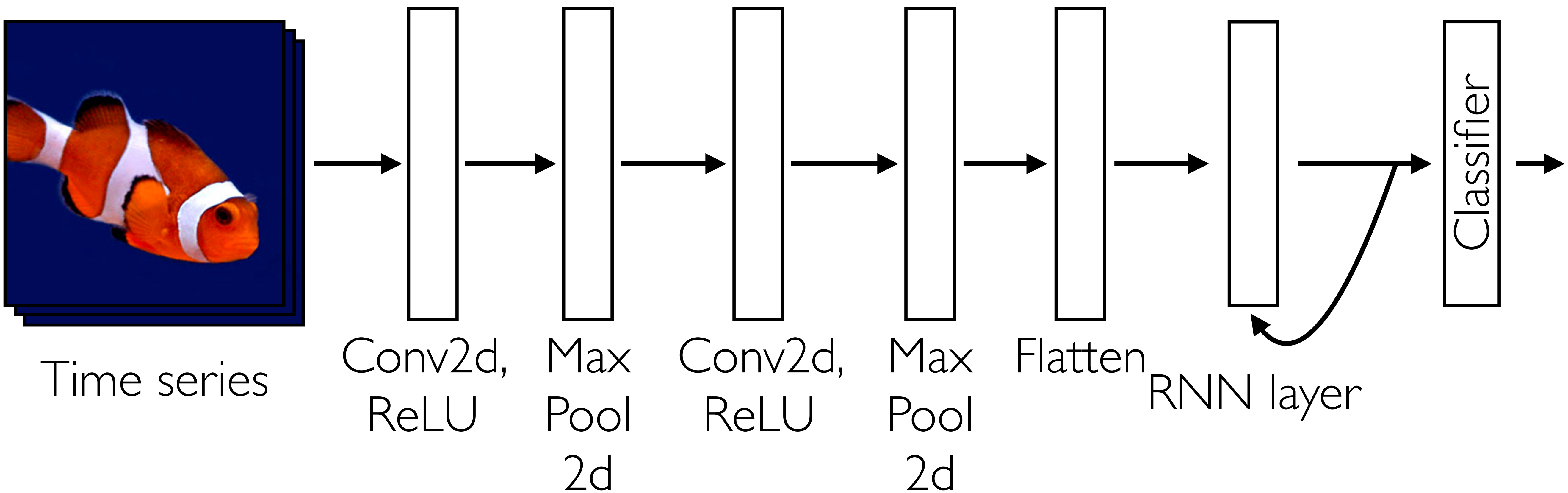
does not incorporate
image structure!!!

Feedforward NN's:
treat each video frame
separately

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

In PyTorch, different
RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Recurrent Neural Nets



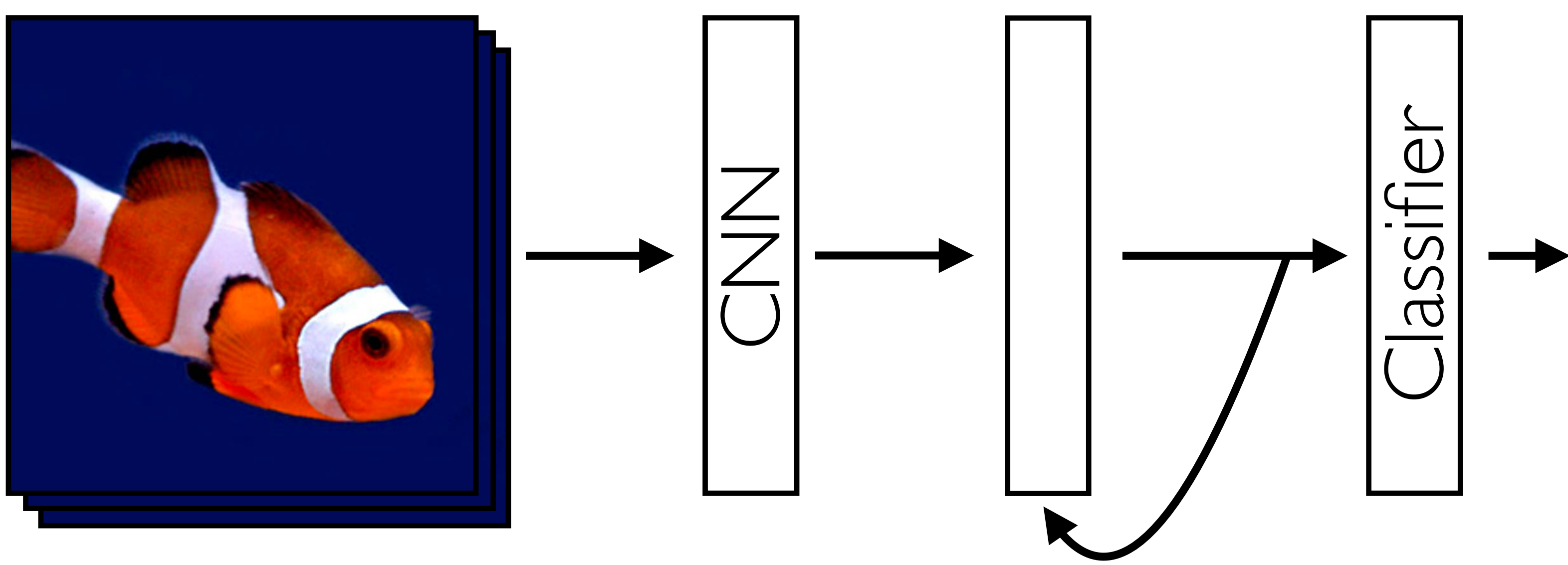
Same CNN applied to each frame separately

Recurrent Neural Nets

Feedforward NN's:
treat each video frame separately

readily chains together with other neural net layers

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



Time series

RNN layer

Use CNN to incorporate image structure!

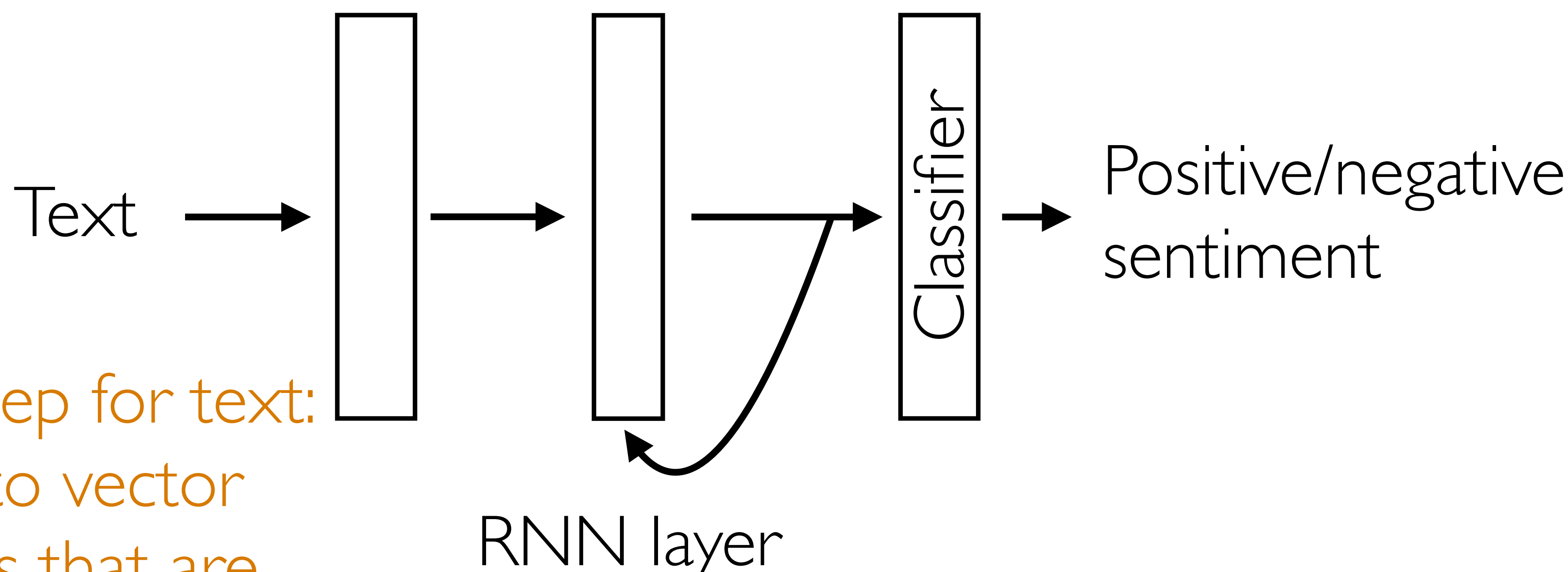
like a linear layer that has memory

does not incorporate image structure!!!

In PyTorch, different RNN options, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Recurrent Neural Nets

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) Do Data Actually Live on Manifolds?

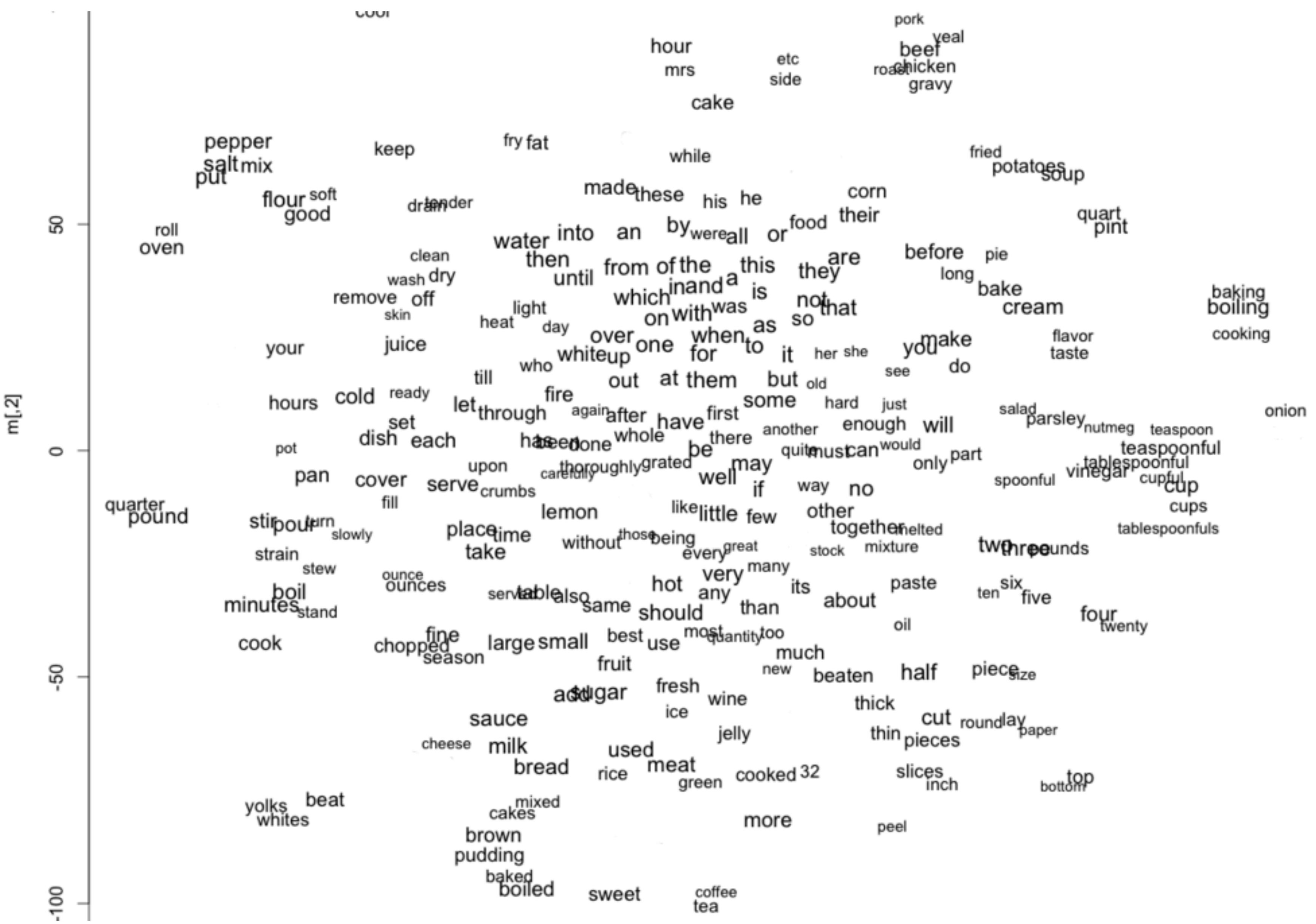
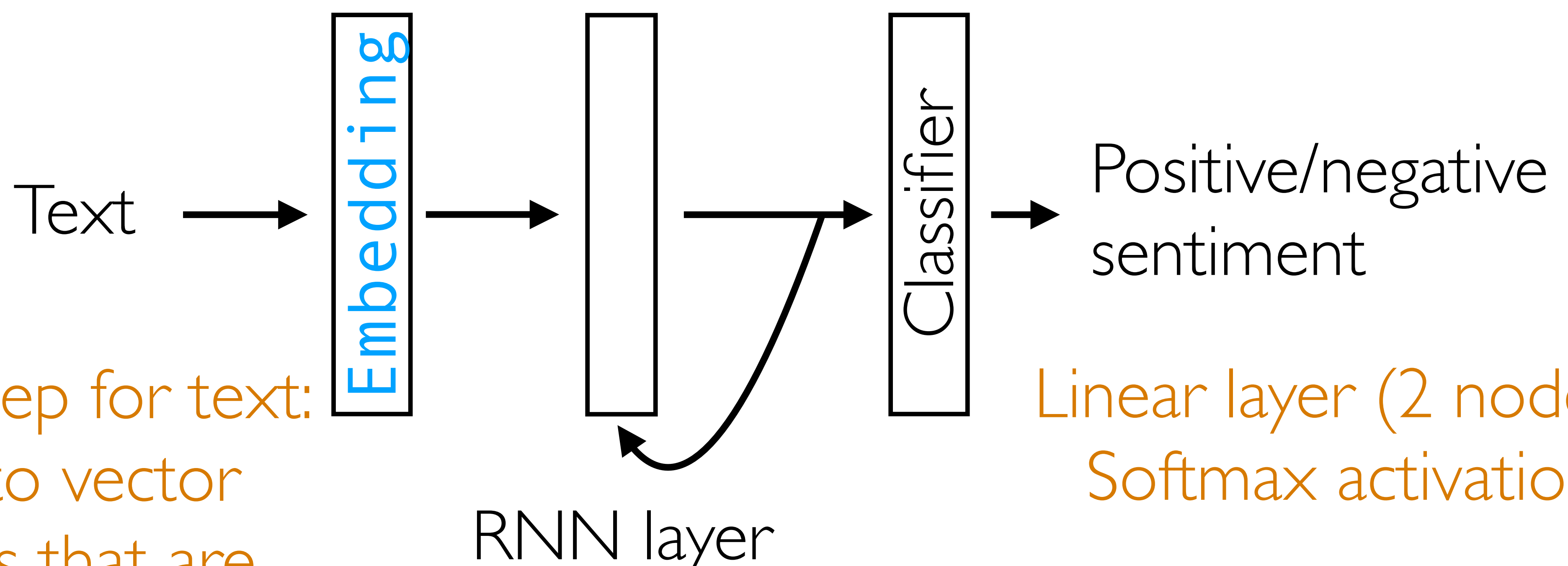


Image source: http://www.adityathakker.com/wp-content/uploads/2017/06/word-embeddings-994x675.png

Recurrent Neural Nets

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

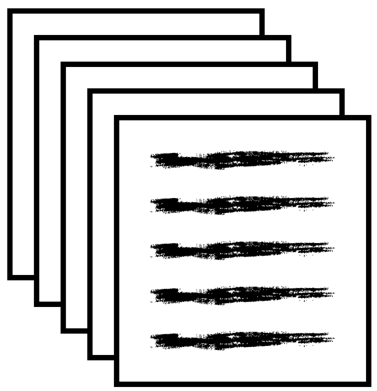


Common first step for text: turn words into vector representations that are semantically meaningful

Linear layer (2 nodes), Softmax activation

In PyTorch, use the `Embedding` layer

Sentiment Analysis with IMDb Reviews



Training reviews

Step 1: Tokenize & build vocabulary

Word index	Word	2D Embedding
0	this	[-0.57, 0.44]
1	movie	[0.38, 0.15]
2	rocks	[-0.85, 0.70]
3	sucks	[-0.26, 0.66]

Step 2: Encode each review as a sequence of word indices into the vocab

Ordering of words matters

Different reviews can have different lengths

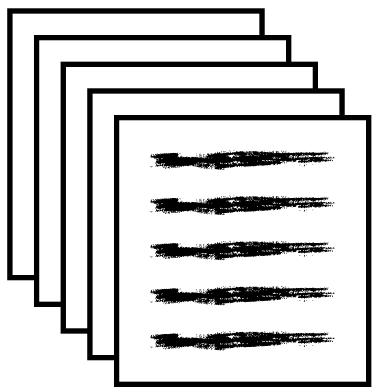
"this movie rocks" → 0 1 2

"this movie sucks" → 0 1 3

"this sucks" → 0 3

Step 3: Use word embeddings to represent each word

Sentiment Analysis with IMDb Reviews



Training reviews

Step 1: Tokenize & build vocabulary

Word index	Word	2D Embedding
0	this	[-0.57, 0.44]
1	movie	[0.38, 0.15]
2	rocks	[-0.85, 0.70]
3	sucks	[-0.26, 0.66]

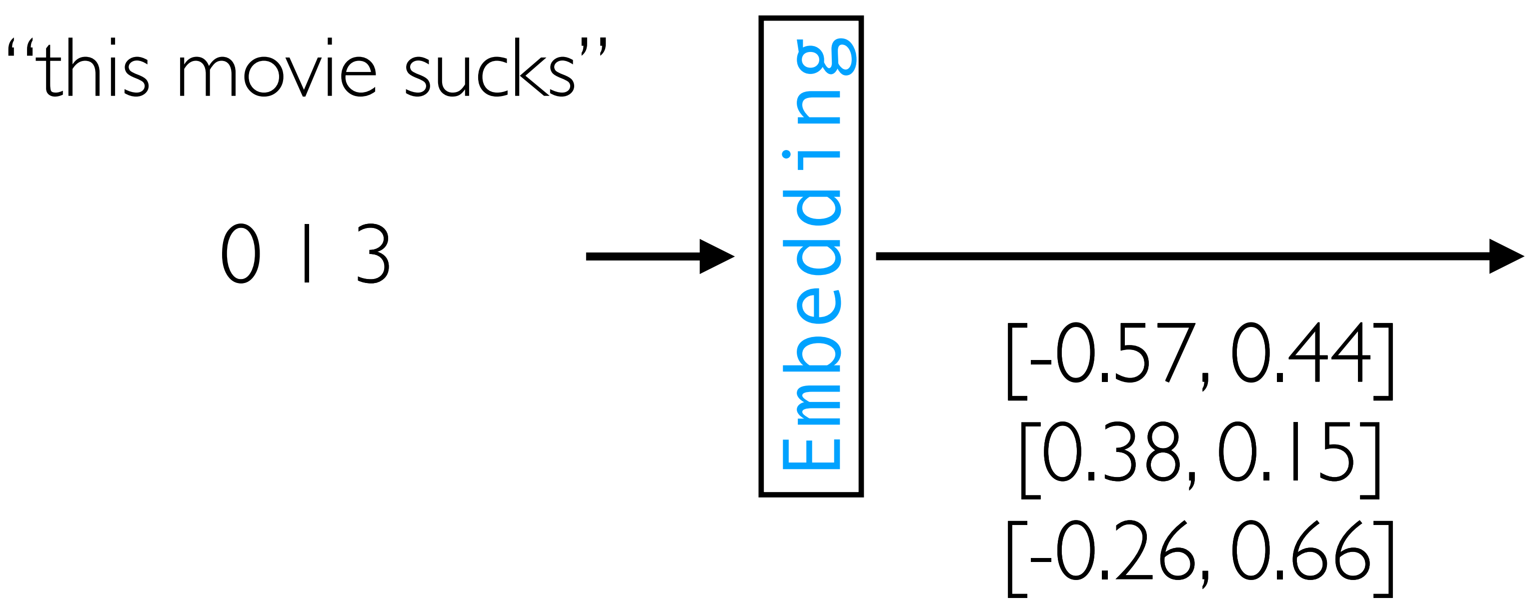
Step 2: Encode each review as a sequence of word indices into the vocab

"this movie sucks" → 0 1 3

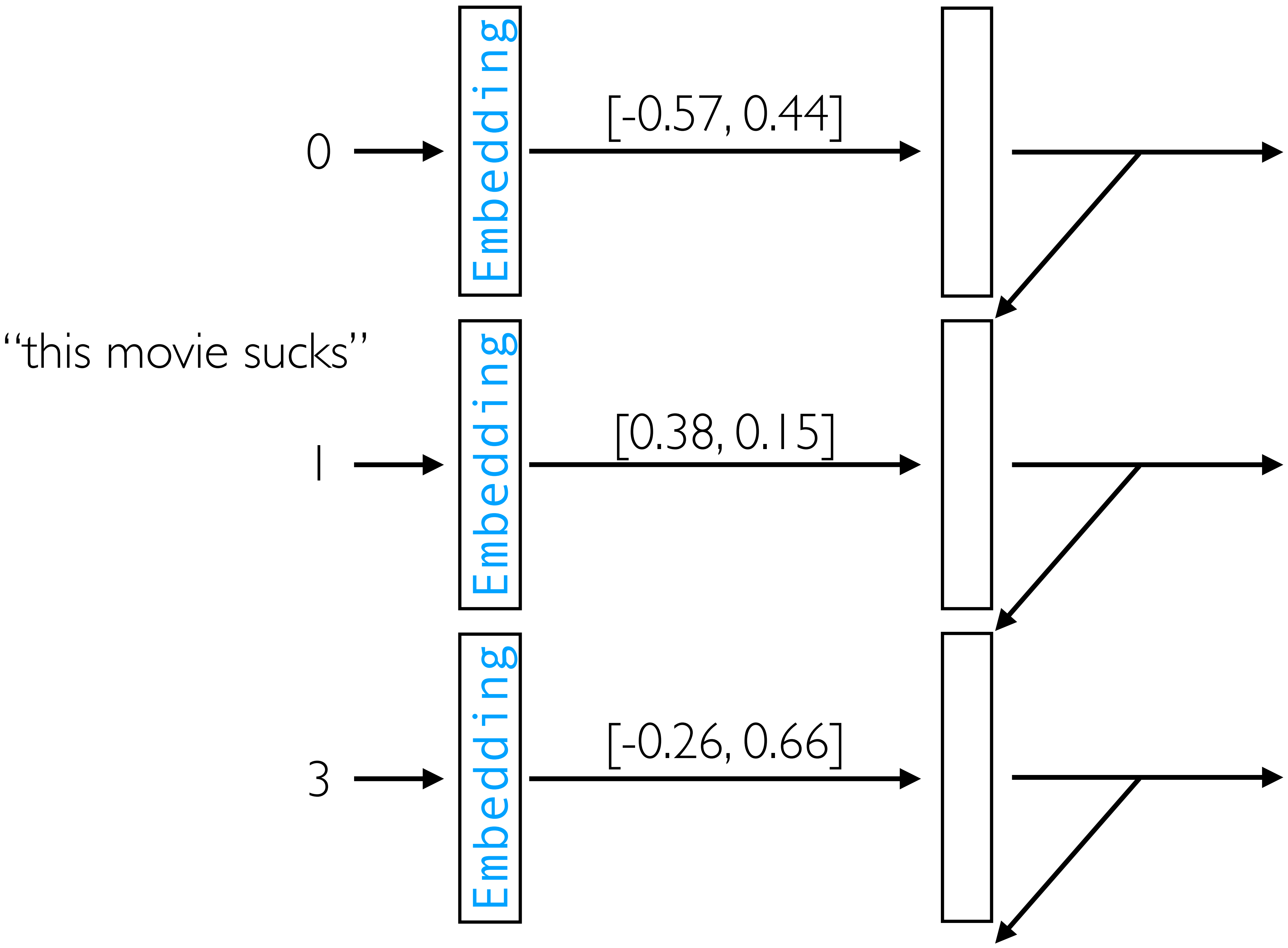
Step 3: Use word embeddings to represent each word

[-0.57, 0.44]
[0.38, 0.15]
[-0.26, 0.66]

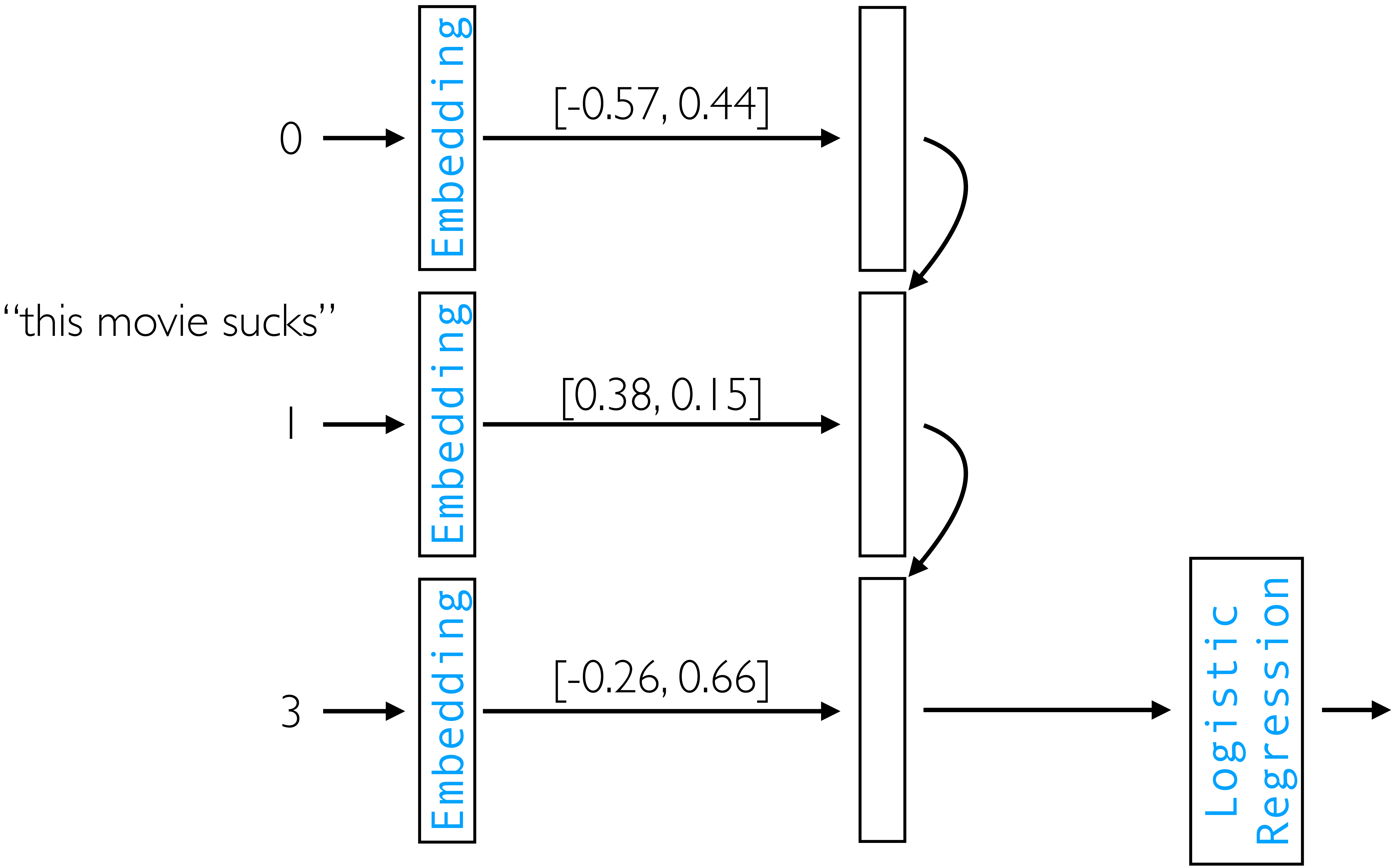
Sentiment Analysis with IMDb Reviews



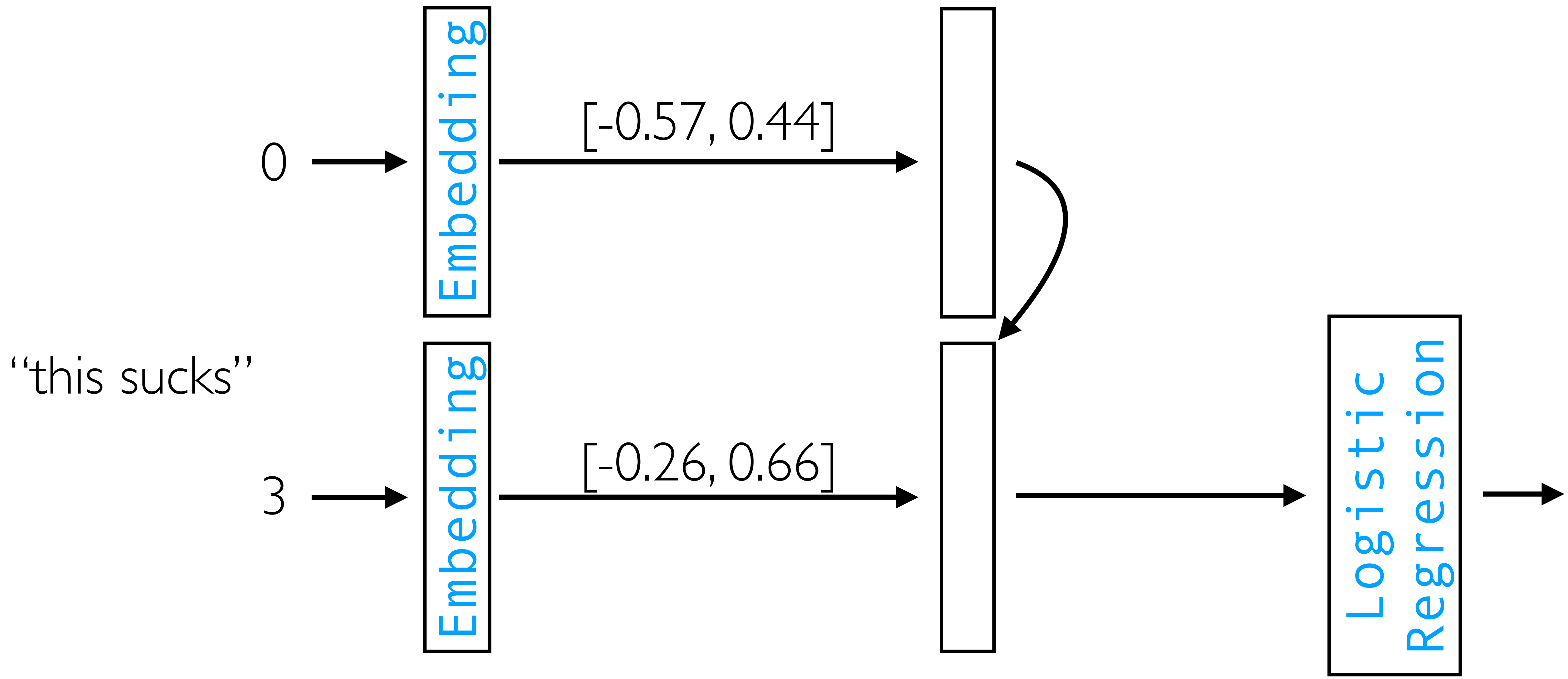
Sentiment Analysis with IMDb Reviews



Sentiment Analysis with IMDb Reviews



Sentiment Analysis with IMDb Reviews

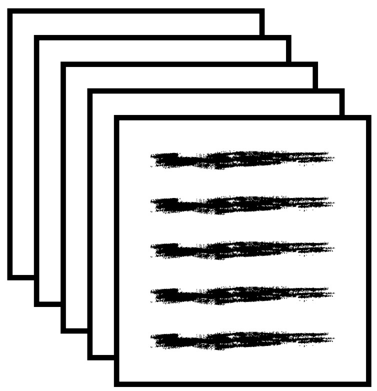


RNN's work with variable-length inputs

Note: Often in text analysis, the word embeddings are treated as fixed, so we do *not* update them during training

What if we didn’t use word embeddings?

Sentiment Analysis with IMDb Reviews



Training reviews

Step 1: Tokenize & build vocabulary

Word index	Word	2D Embedding
0	this	[-0.57, 0.44]
1	movie	[0.38, 0.15]
2	rocks	[-0.85, 0.70]
3	sucks	[-0.26, 0.66]

Step 2: Encode each review as a sequence of word indices into the vocab

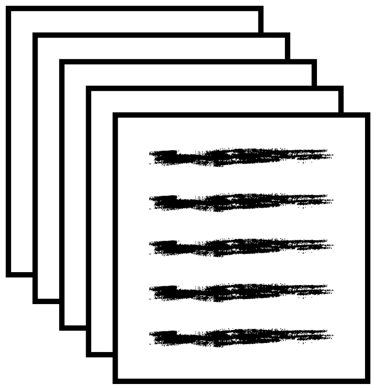
"this movie sucks" → 0 1 3

Step 3: Use word embeddings to represent each word

[-0.57, 0.44]
[0.38, 0.15]
[-0.26, 0.66]

Bad Strategy: One-Hot Encoding

Step 1: Tokenize & build vocabulary



Training reviews

Word index	Word	One-hot encoding
0	this	[1, 0, 0, 0]
1	movie	[0, 1, 0, 0]
2	rocks	[0, 0, 1, 0]
3	sucks	[0, 0, 0, 1]

Step 2: Encode each review as a sequence of word indices into the vocab

"this movie sucks" → 0 1 3

Step 3: Use one-hot encoding to represent each word

[1, 0, 0, 0]
[0, 1, 0, 0]
[0, 0, 0, 1]

This strategy tends to work poorly in practice: distance between every pair of words is the same in one-hot encoding!

Recap/Important Reminder

- Neural nets are *not* doing magic; **incorporating structure is very important to state-of-the-art deep learning systems**

  - Word embeddings encode semantic structure—words with similar meaning are mapped to nearby Euclidean points

  - CNNs encode semantic structure for images—images that are "similar" are mapped to nearby Euclidean points

- An RNN tracks how what's stored in memory changes over time — **an RNN's job is made easier if the memory is a semantically meaningful representation** embedding_weights (100-dimensional GloVe embeddings in the demo)

Sentiment Analysis with IMDb Reviews

In the demo, this part done by creating an instance of the `SpacyEncoder` Python class (`torchnlp` does support other encoders as well in case you don't like spacy/spacy is giving you trouble)

Step 1: Tokenize & build vocabulary

Word index	Word	2D Embedding
0	this	[-0.57, 0.44]
1	movie	[0.38, 0.15]
2	rocks	[-0.85, 0.70]
3	sucks	[-0.26, 0.66]

Step 2: Encode each review as a sequence of word indices into the vocab

"this movie sucks" → 0 1 3

Step 3: Use word embeddings to represent each word

[-0.57, 0.44]
[0.38, 0.15]
[-0.26, 0.66]

Sentiment Analysis with IMDb Reviews

Demo

A special kind of RNN: an “LSTM”

(Flashback) Vanilla ReLU RNN

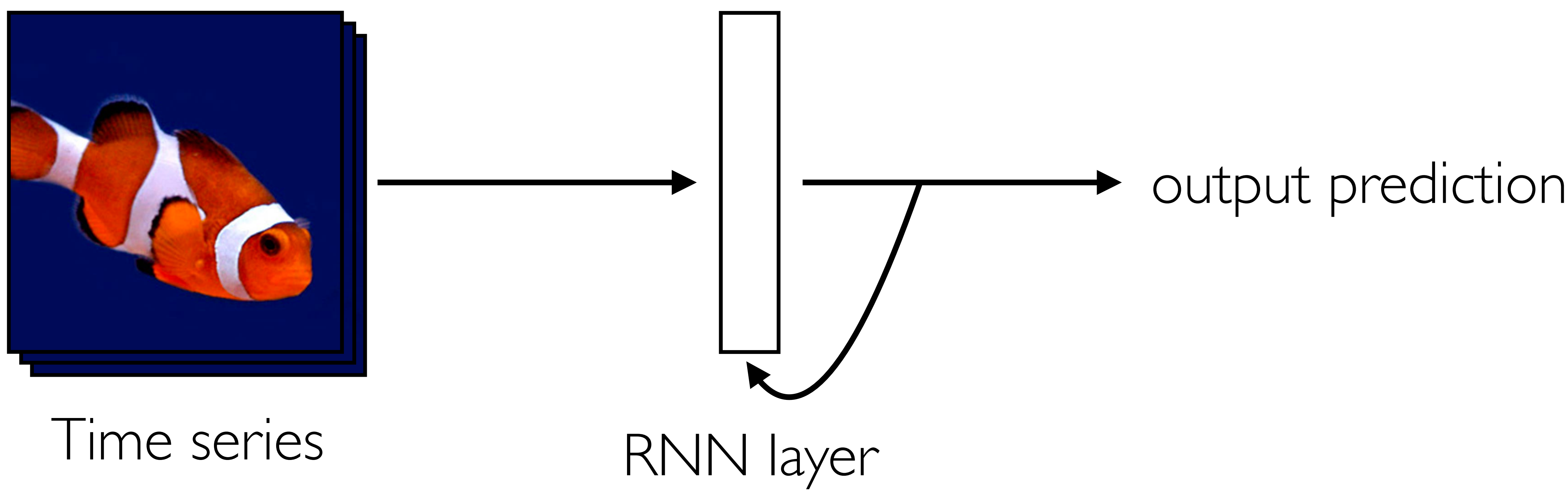
```
current_state = np.zeros(num_nodes)

for input in input_sequence:

  linear = np.dot(input, W) \
           + np.dot(current_state, U) \
           + b

  output = np.maximum(0, linear) # ReLU

  current_state = output
```

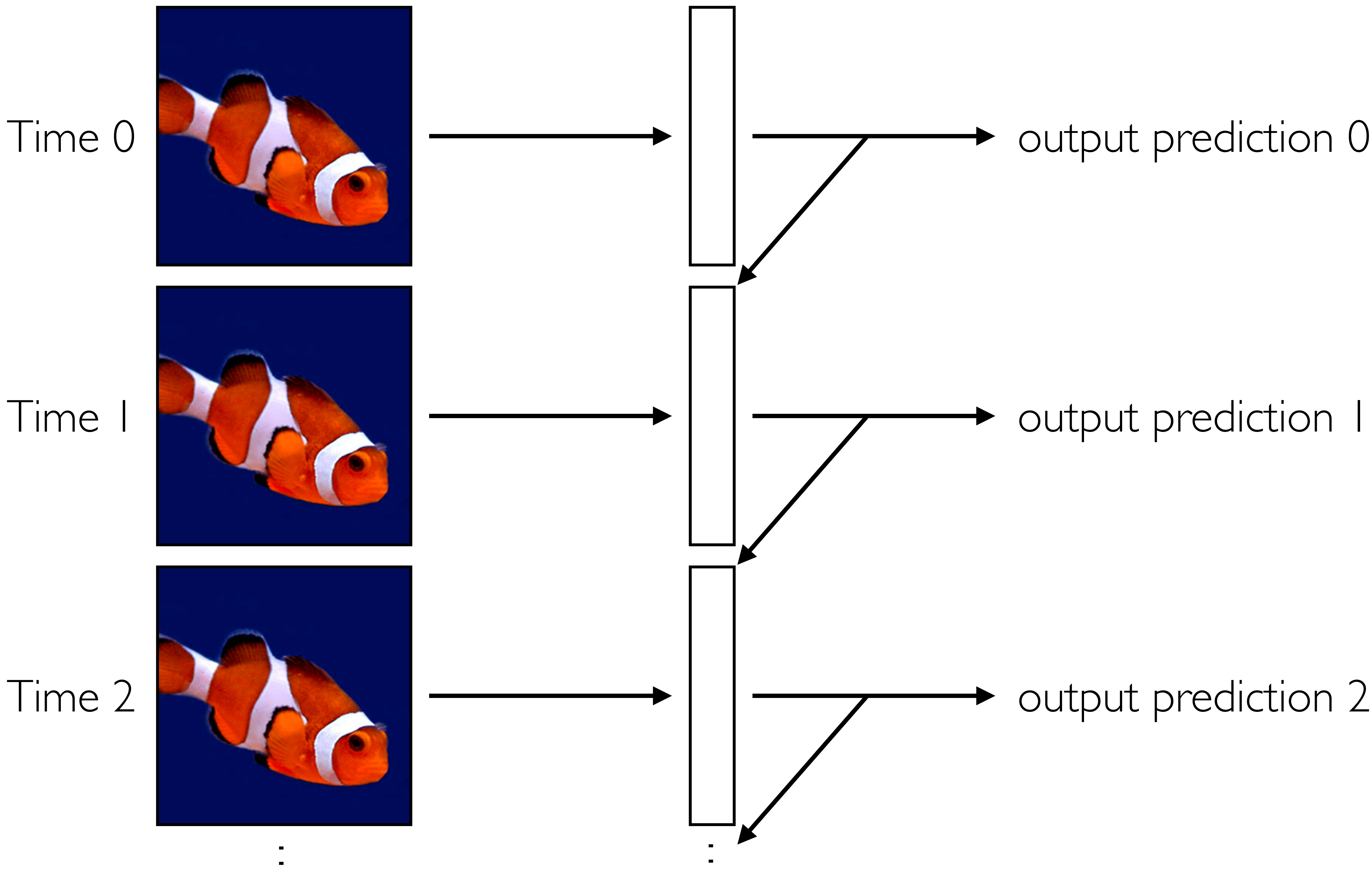
Parameters: weight matrices `W` & `U`, and bias vector `b`

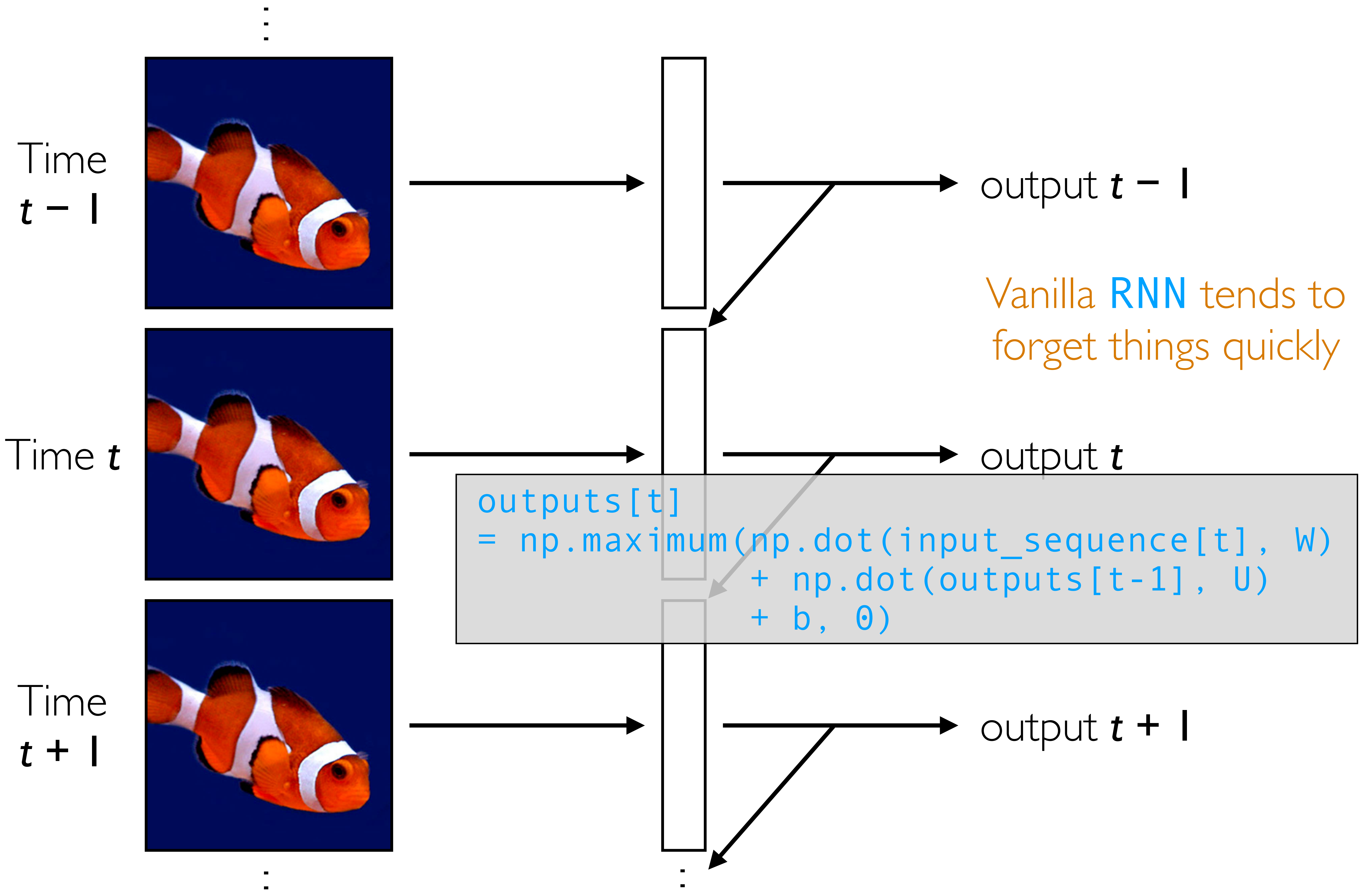
Key idea: **it's like a linear layer in a `for` loop that tracks how memory changes over time**

(Flashback) Vanilla ReLU RNN

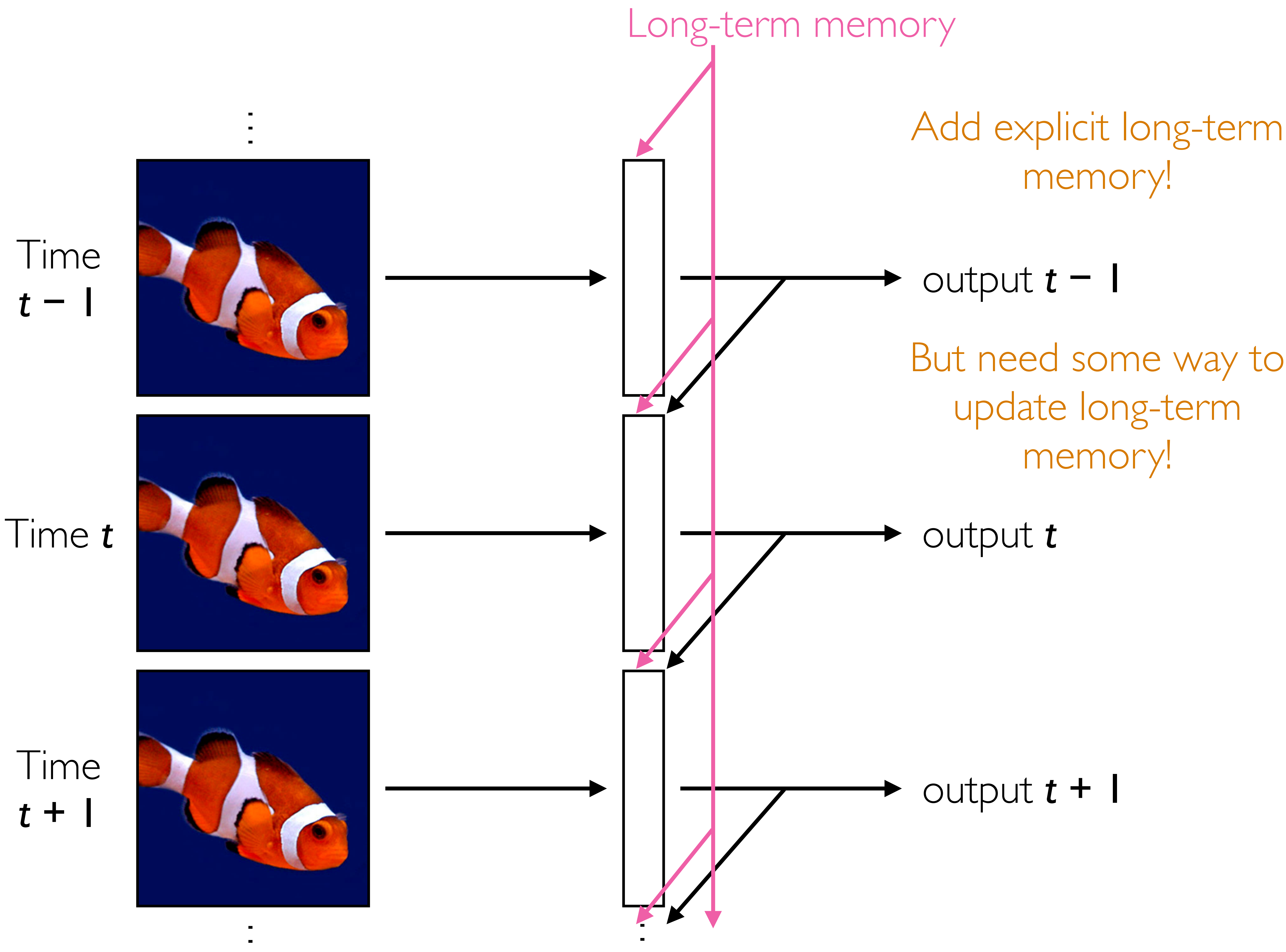
```
current_state = np.zeros(num_nodes)

outputs = []

for input in input_sequence:

  linear = np.dot(input, W) \
           + np.dot(current_state, U) \
           + b

  output = np.maximum(0, linear)  # ReLU

  outputs.append(output)

  current_state = output
```


output prediction
Time series
RNN layer

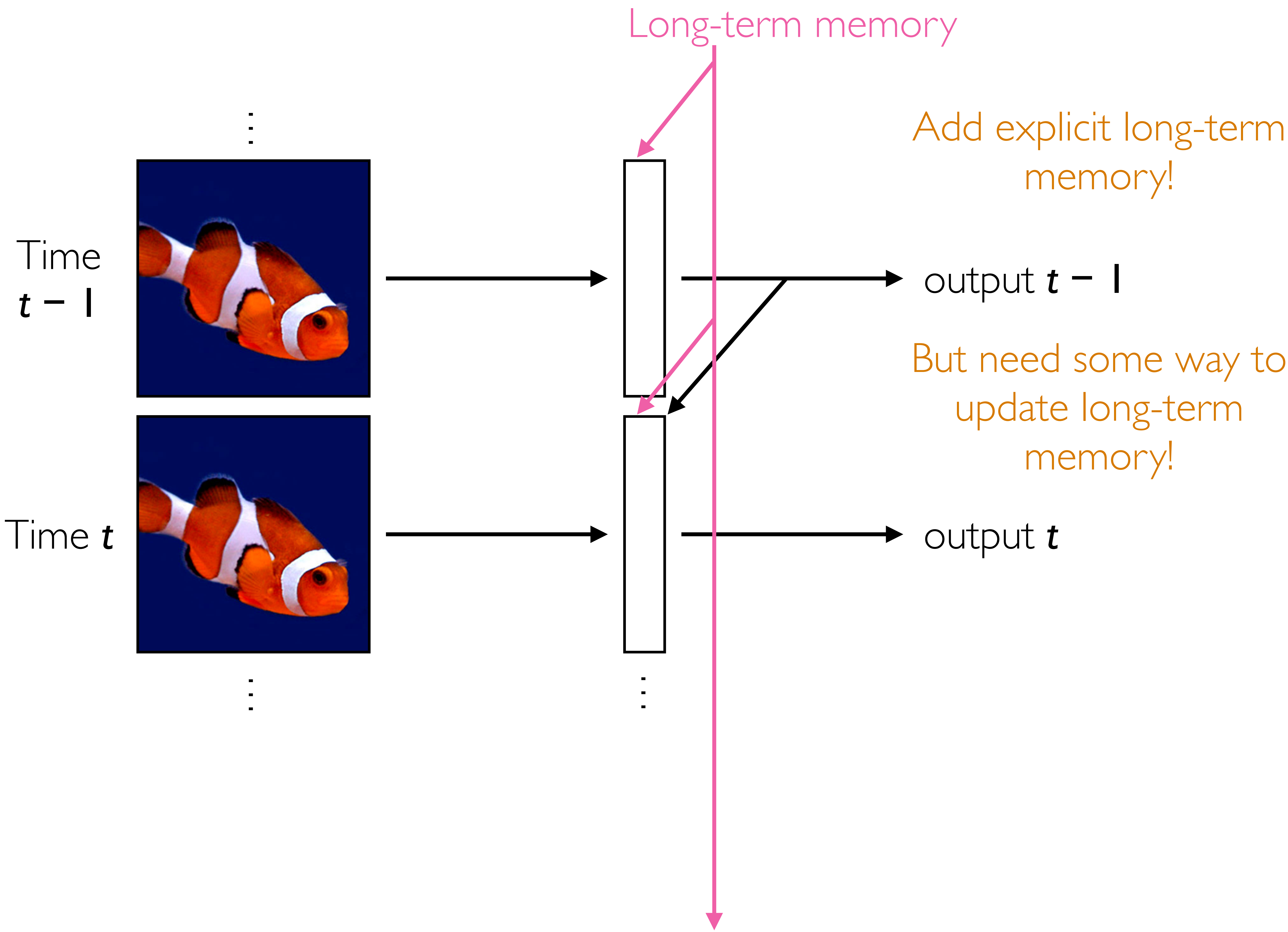

Time 0
Time 1
Time 2
output prediction 0
output prediction 1
output prediction 2

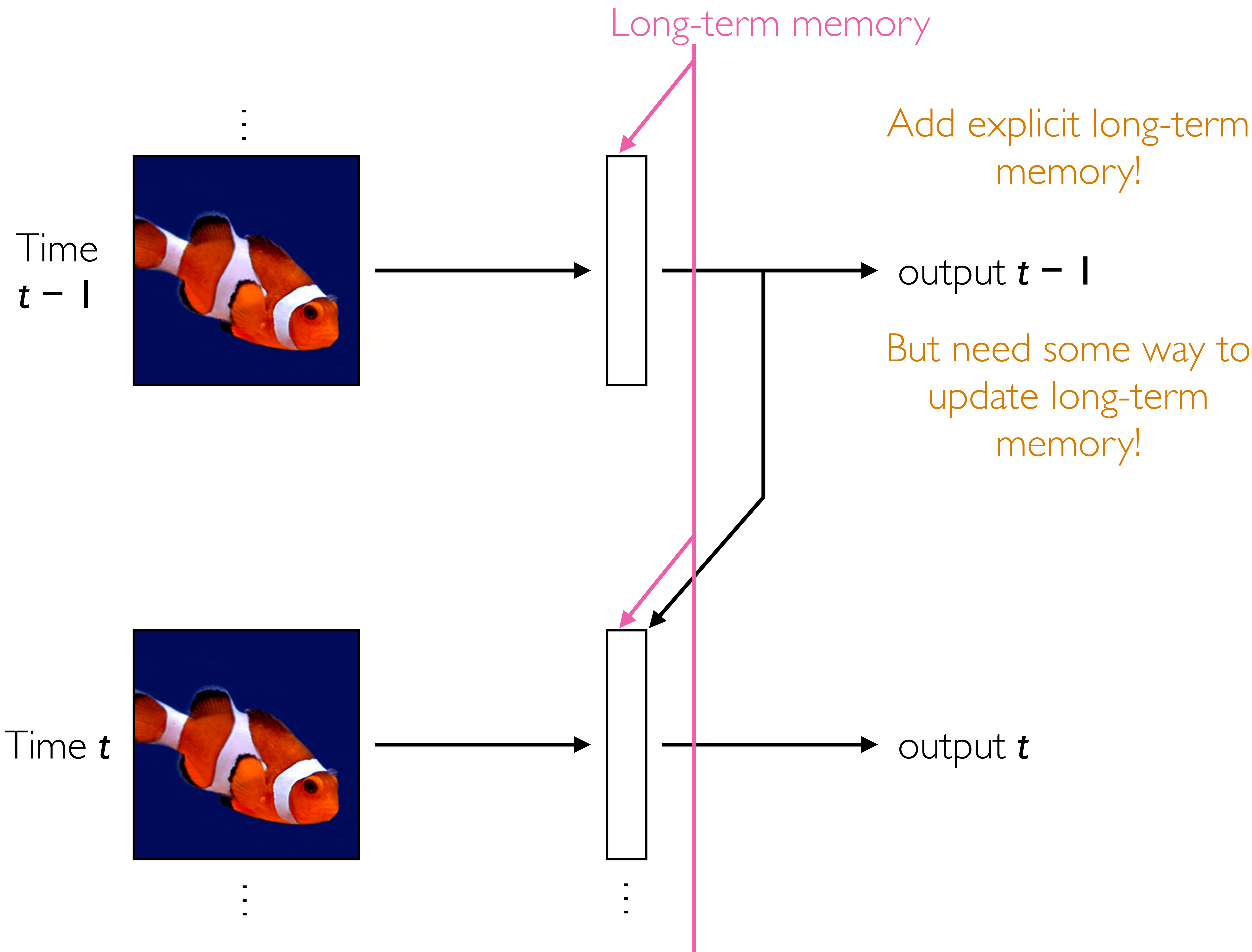


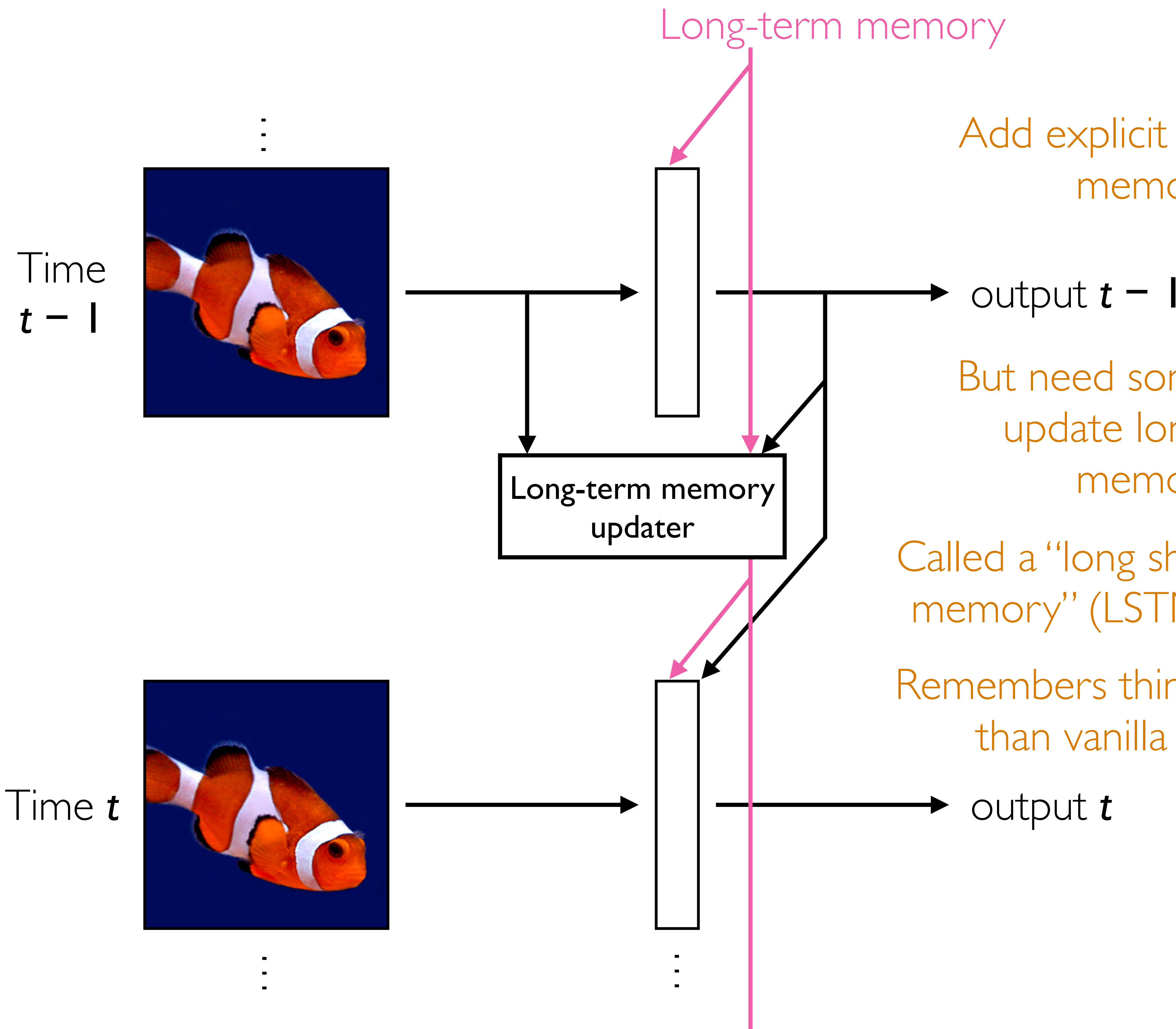

```
outputs[t]
= np.maximum(np.dot(input_sequence[t], W)
             + np.dot(outputs[t-1], U)
             + b, 0)
```



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 vanilla RNN

Analyzing Times Series with CNNs

- Think about an image with 1 column, and where the rows index time steps: this is a time series!

- Think about a 2D image where rows index time steps, and the columns index features: this is a multivariate time series (feature vector that changes over time!)

- CNNs can be used to analyze time series *but inherently the size of the filters used say how far back in time we look*

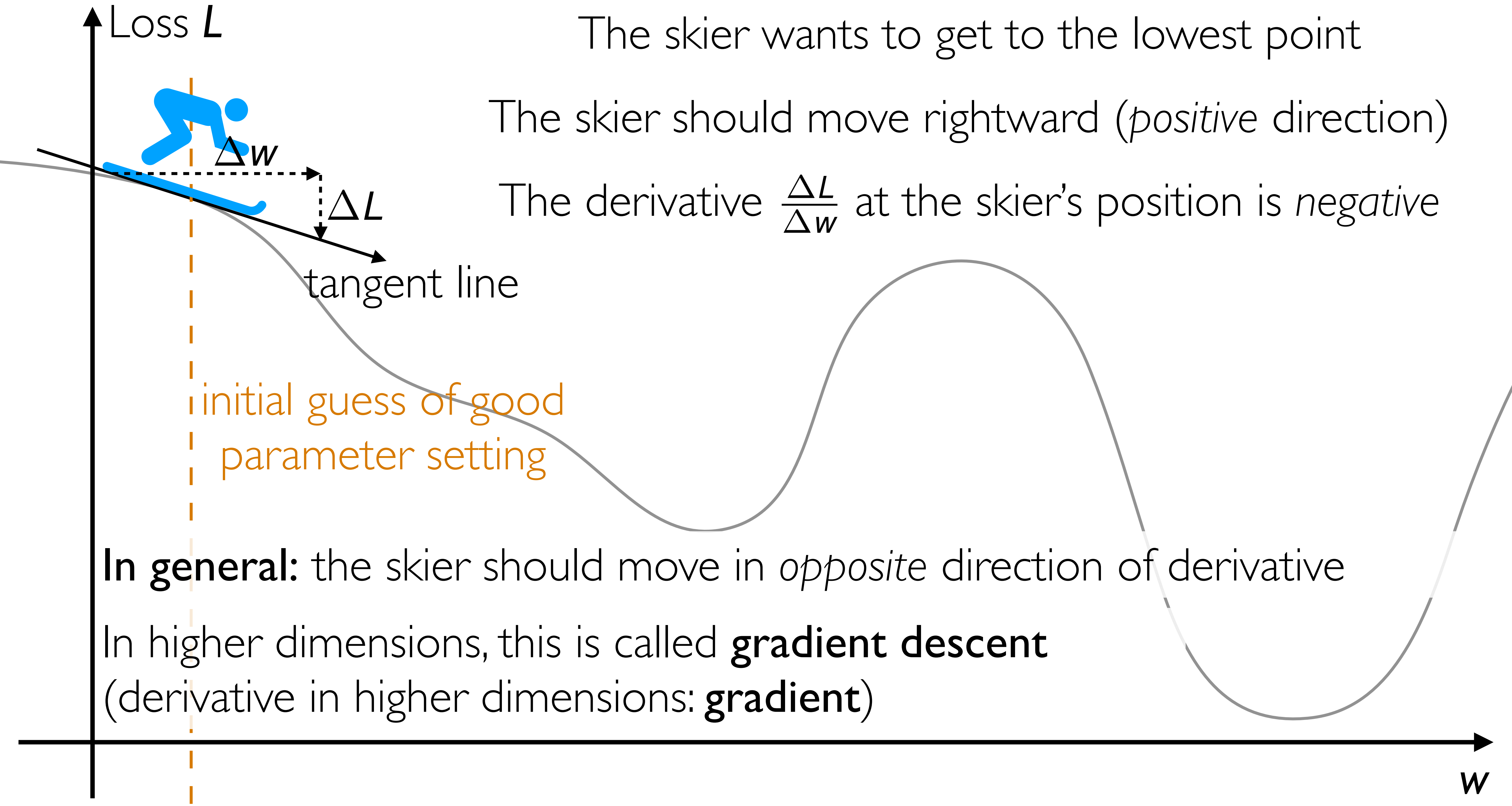
- If your time series does not have long-range dependencies that require long-term memory, CNNs can do well already!

    - If you need long-term memory, use RNNs

Other Deep Learning Topics

Learning 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)

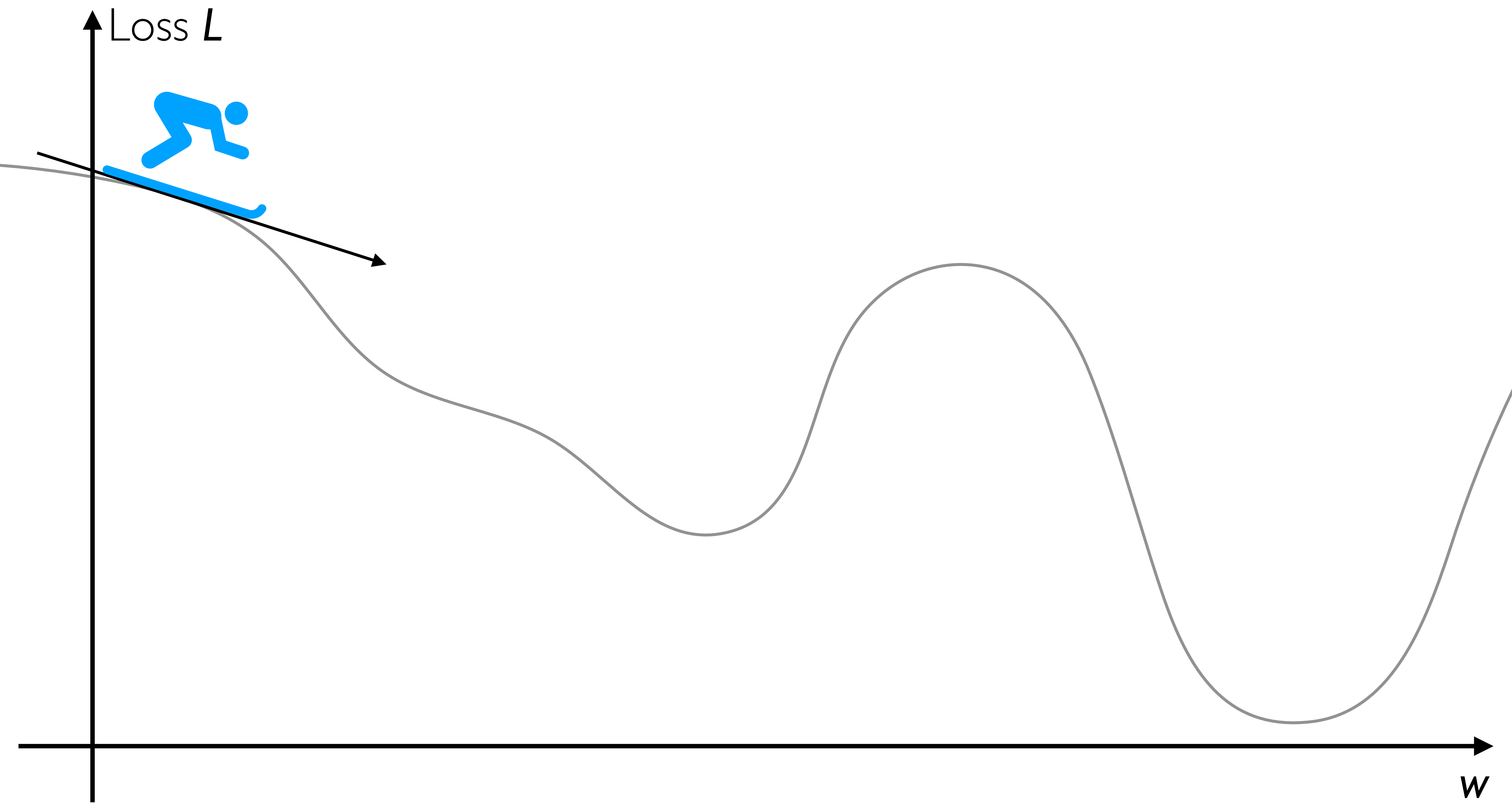
The derivative $\frac{\Delta L}{\Delta w}$ at the skier's position is *negative*

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

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

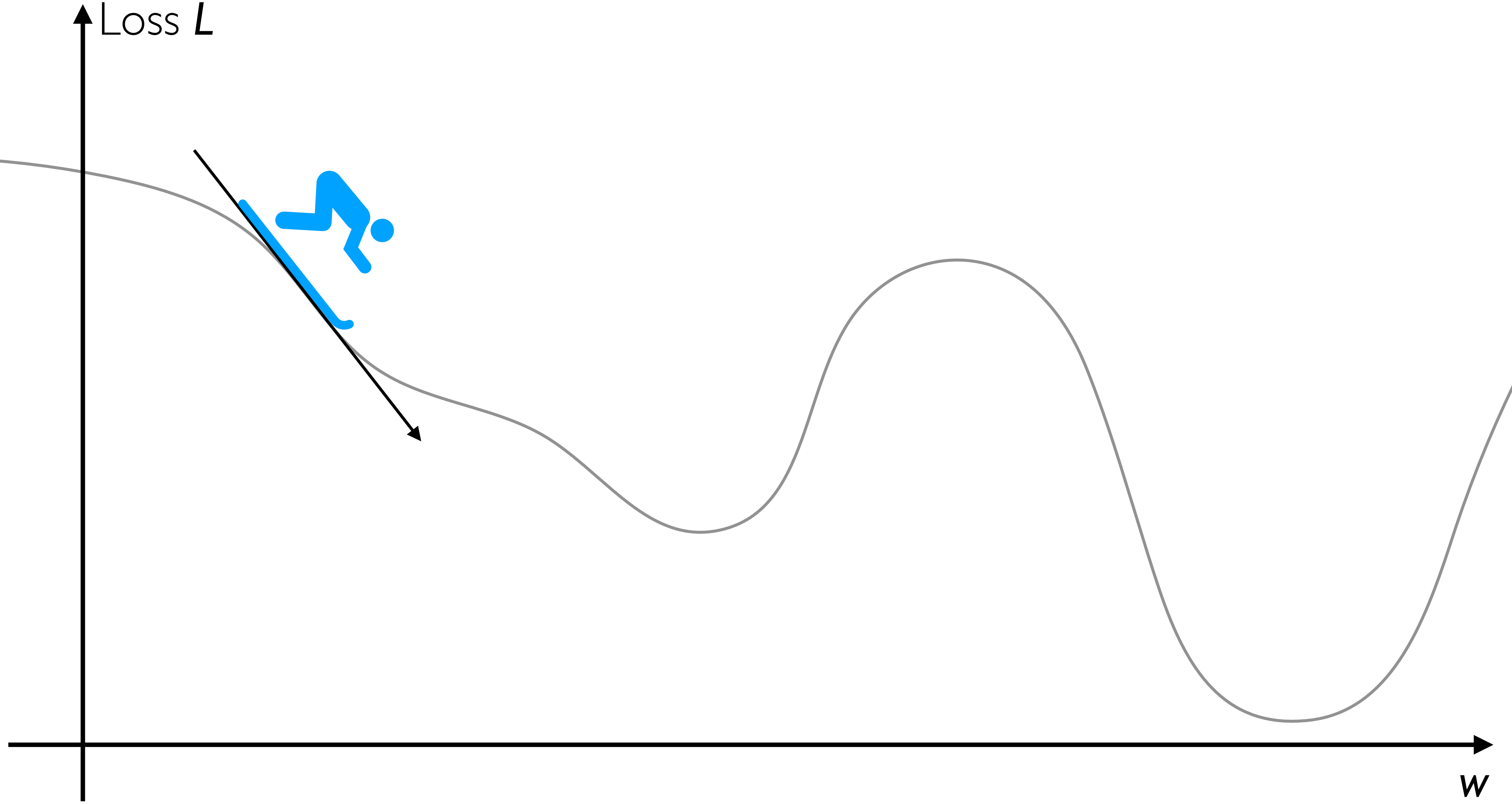
Learning a Deep Net

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



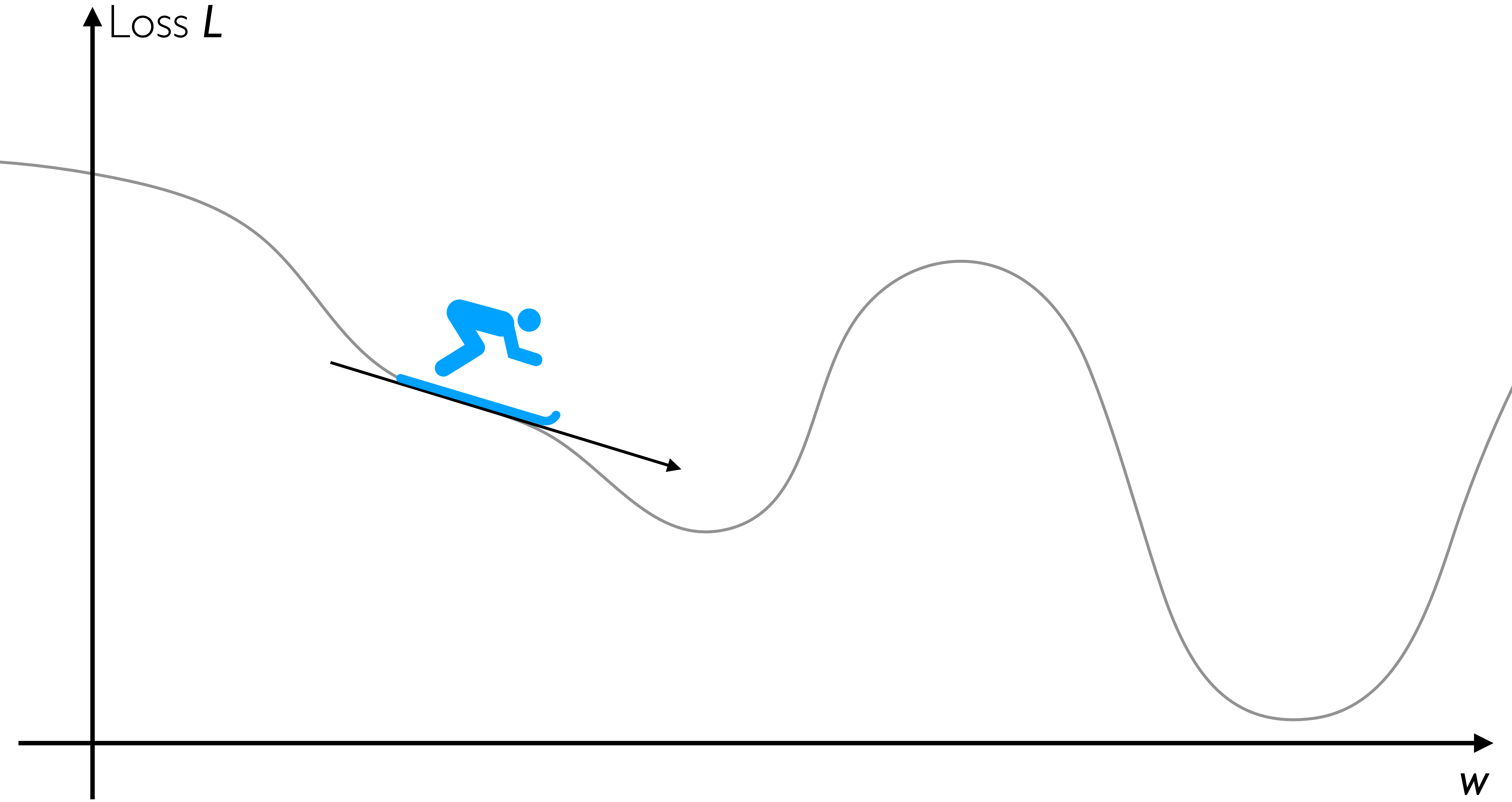
Learning a Deep Net

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



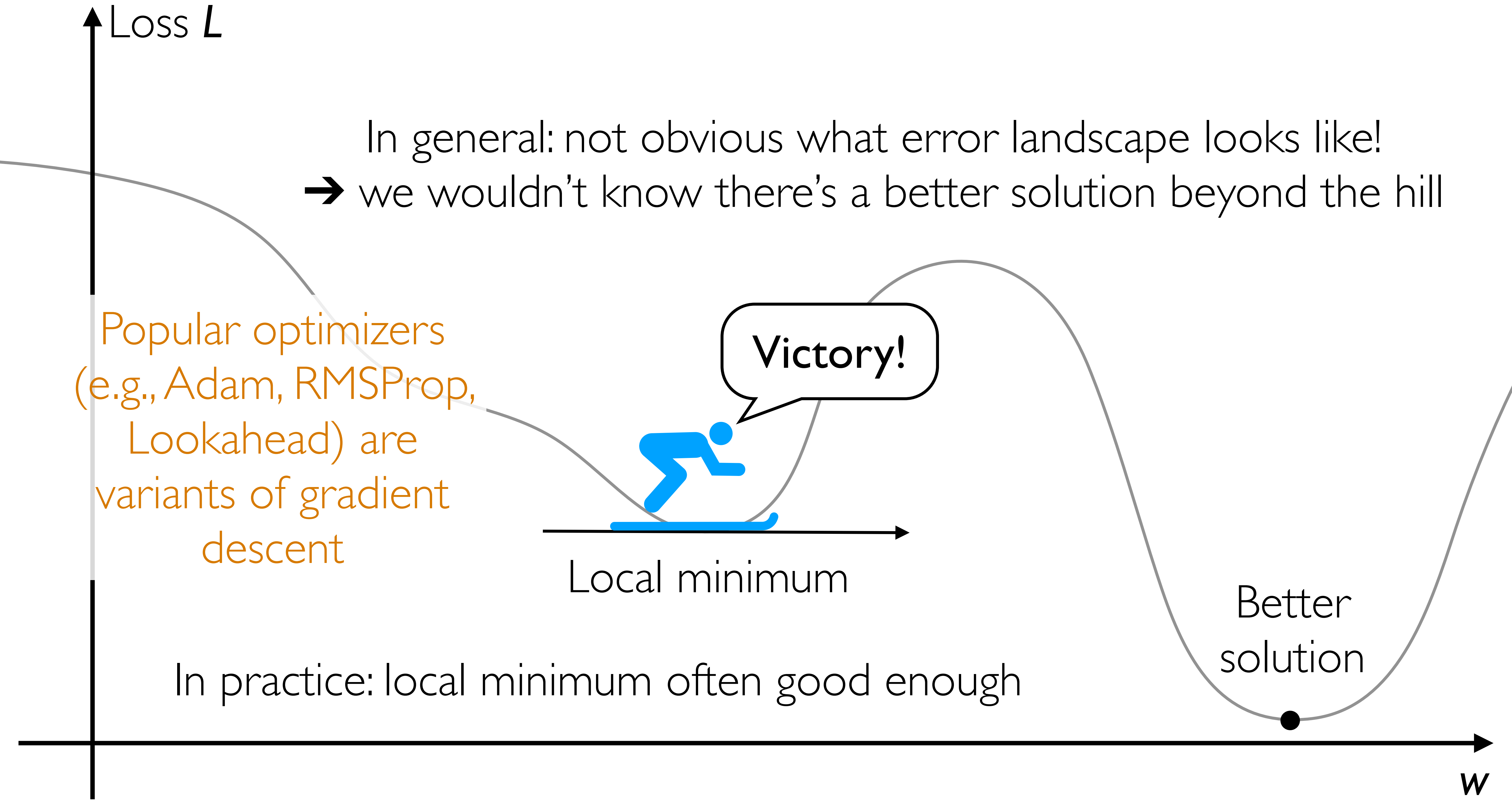
Learning a Deep Net

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



Learning a Deep Net

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

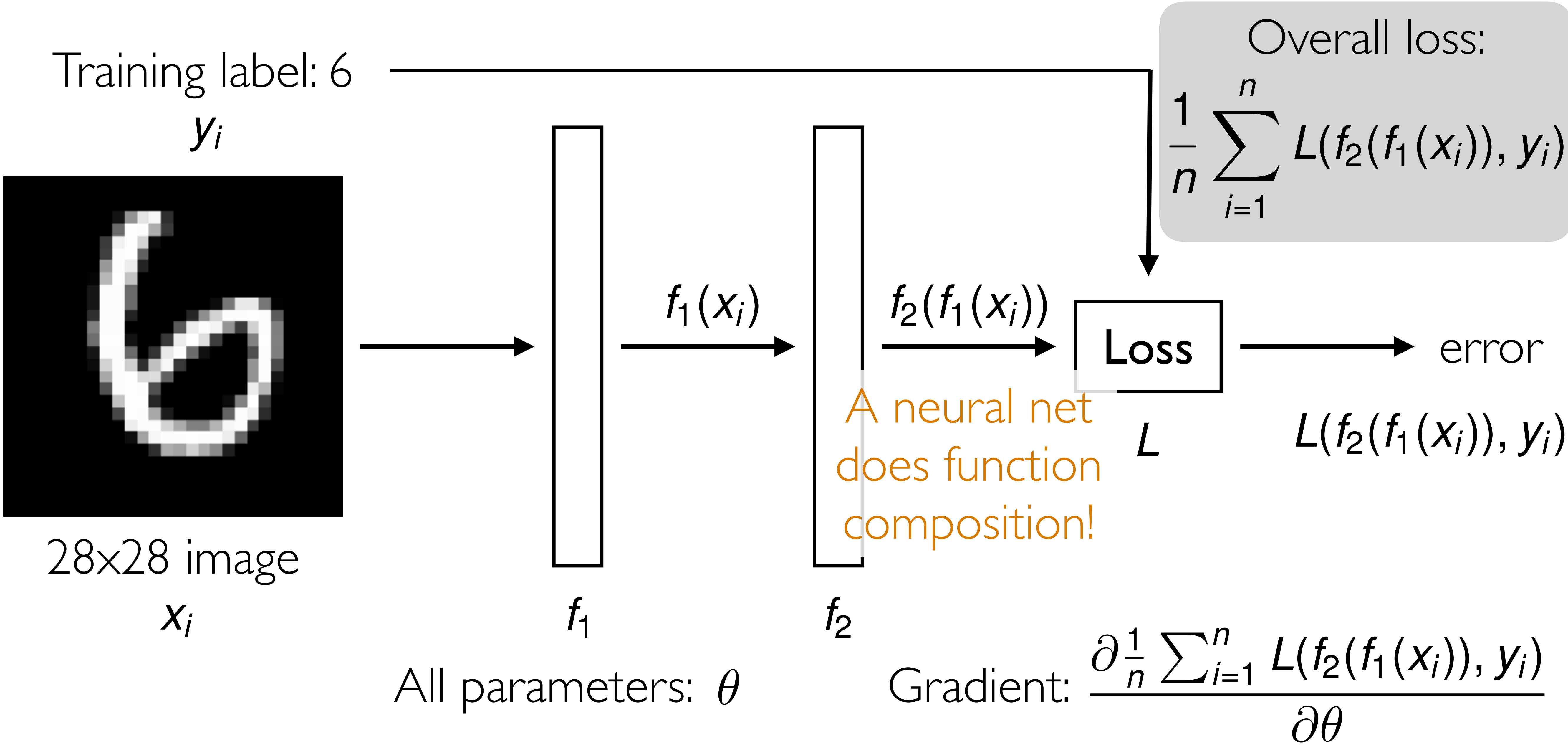


In general: not obvious what error landscape looks like!
➔ we wouldn't know there's a better solution beyond the hill

Popular optimizers (e.g., Adam, RMSProp, Lookahead) are variants of gradient descent

In practice: local minimum often good enough

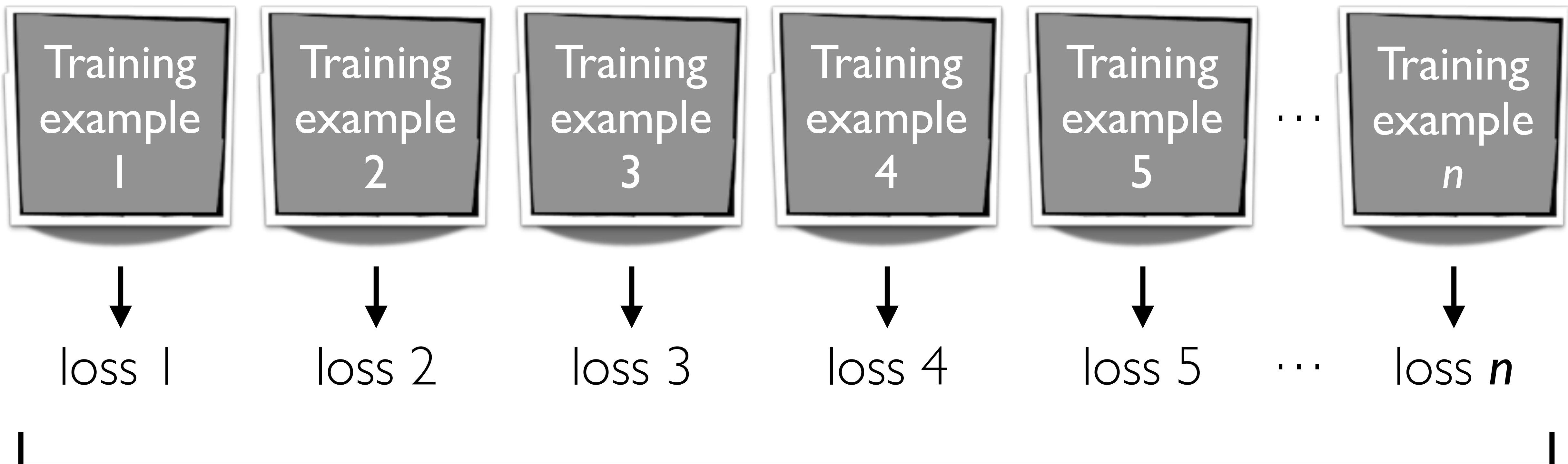
Handwritten Digit Recognition



Automatic differentiation is crucial in learning deep nets!

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

Gradient Descent



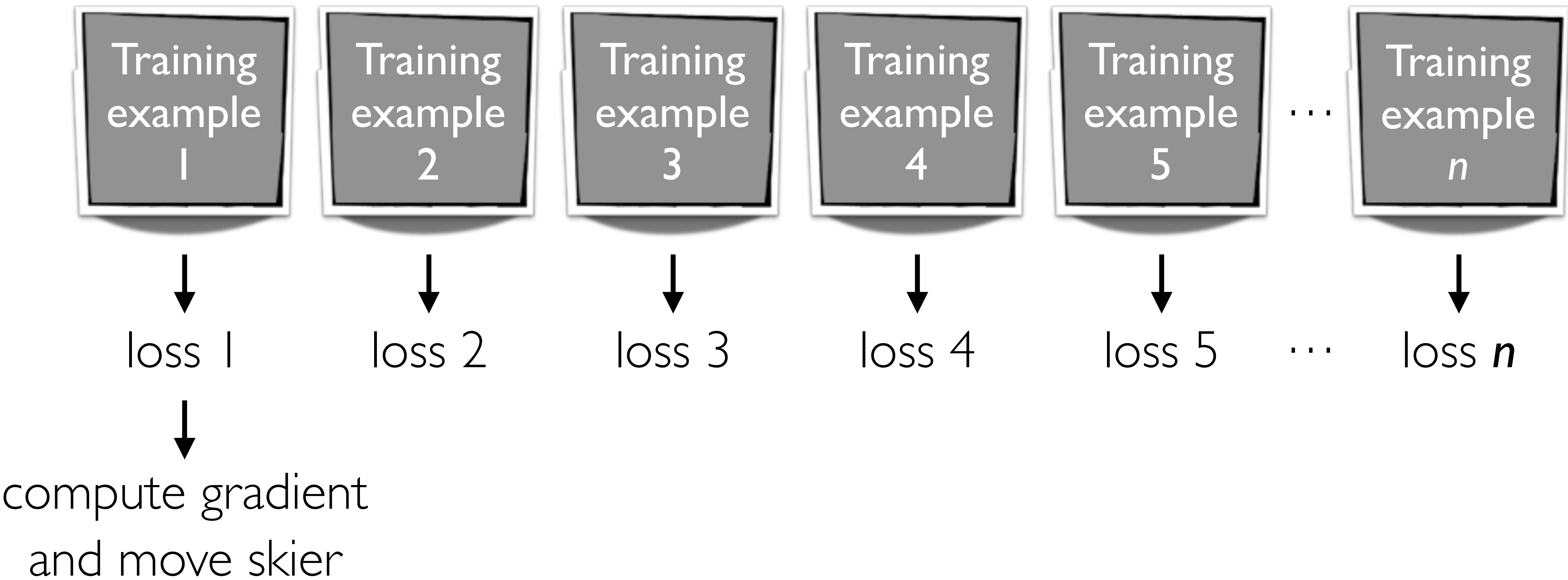
average loss

compute gradient
and move skier

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

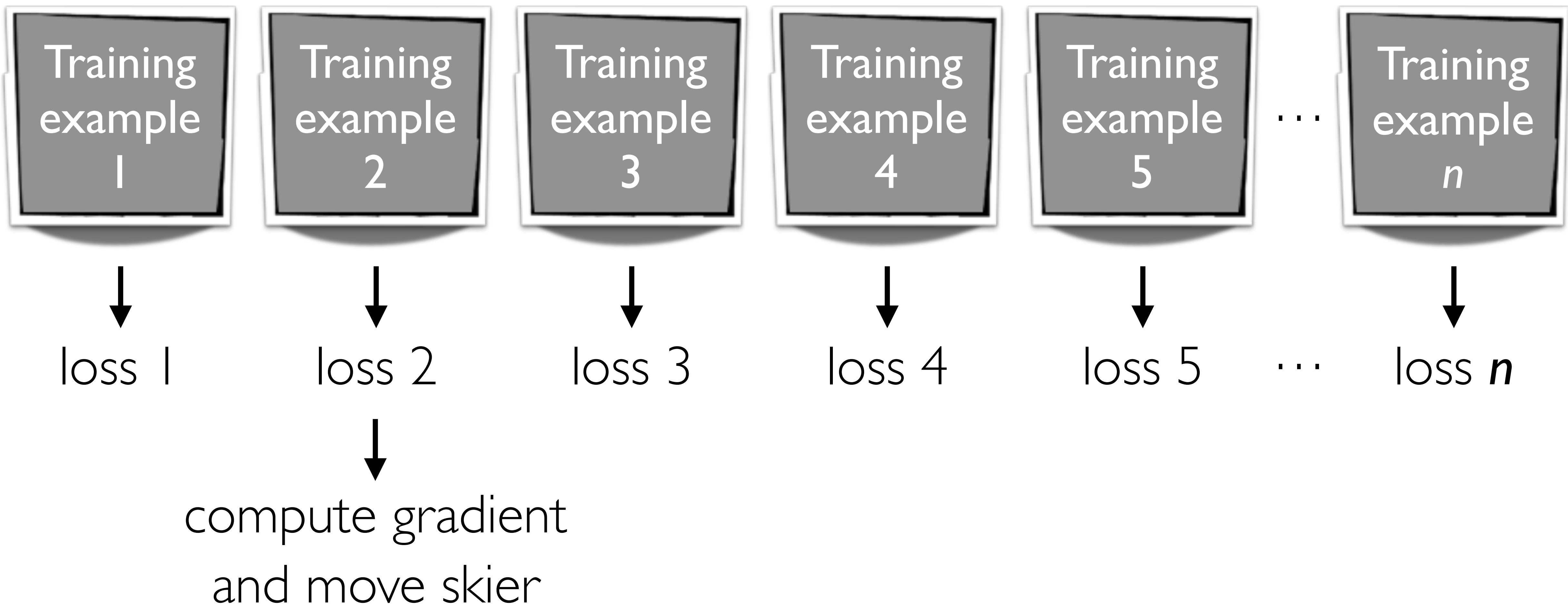
Computing gradients using all the training data seems really expensive!

Stochastic Gradient Descent (SGD)



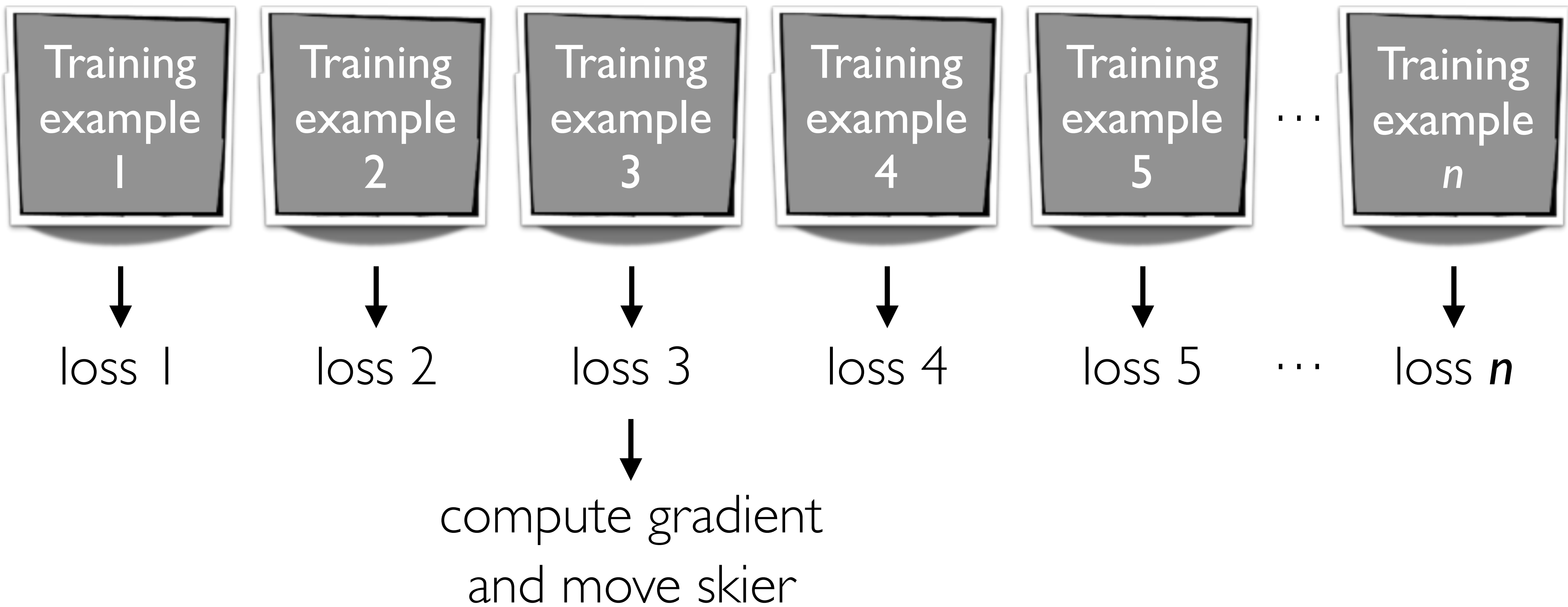
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



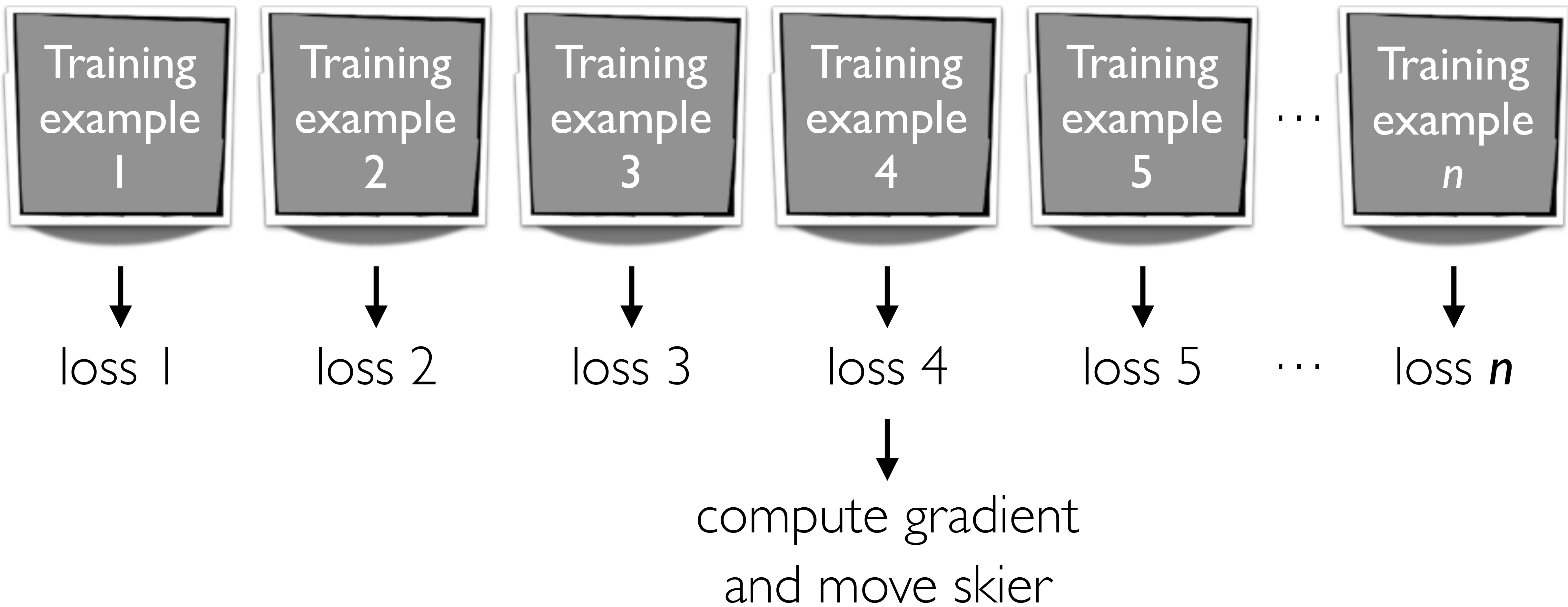
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



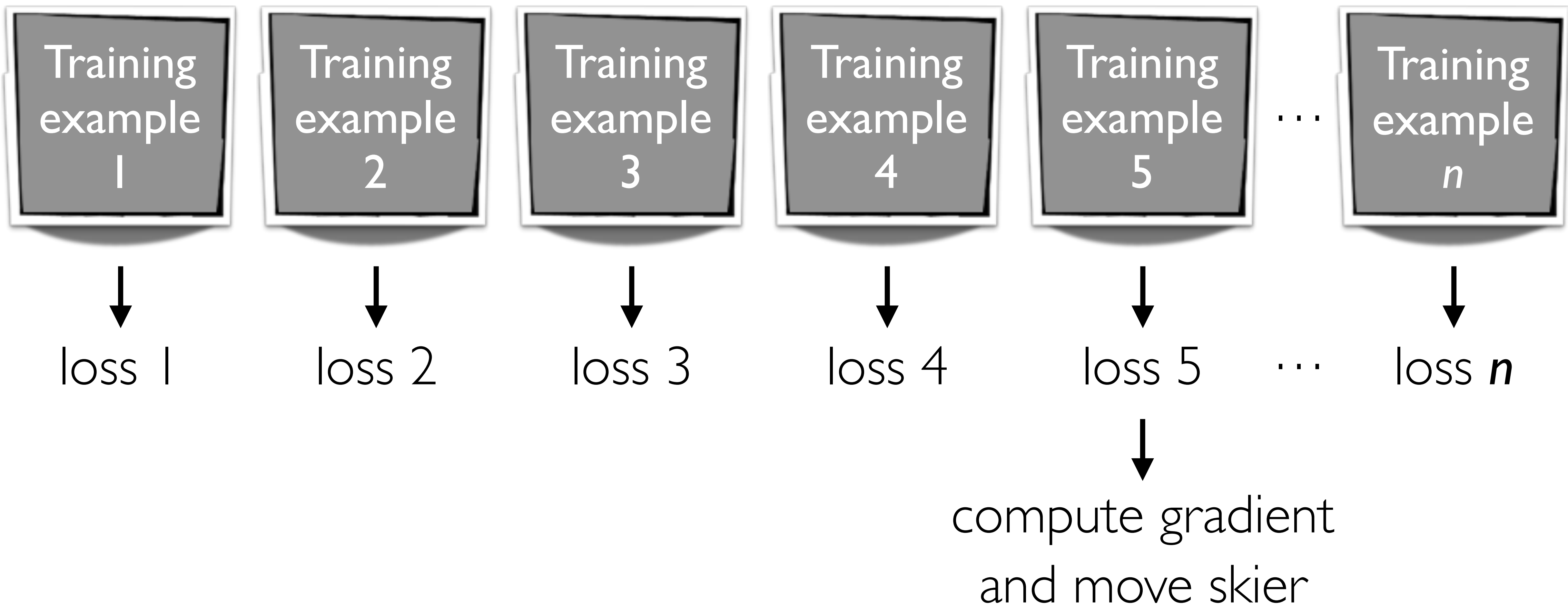
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

Stochastic Gradient Descent (SGD)



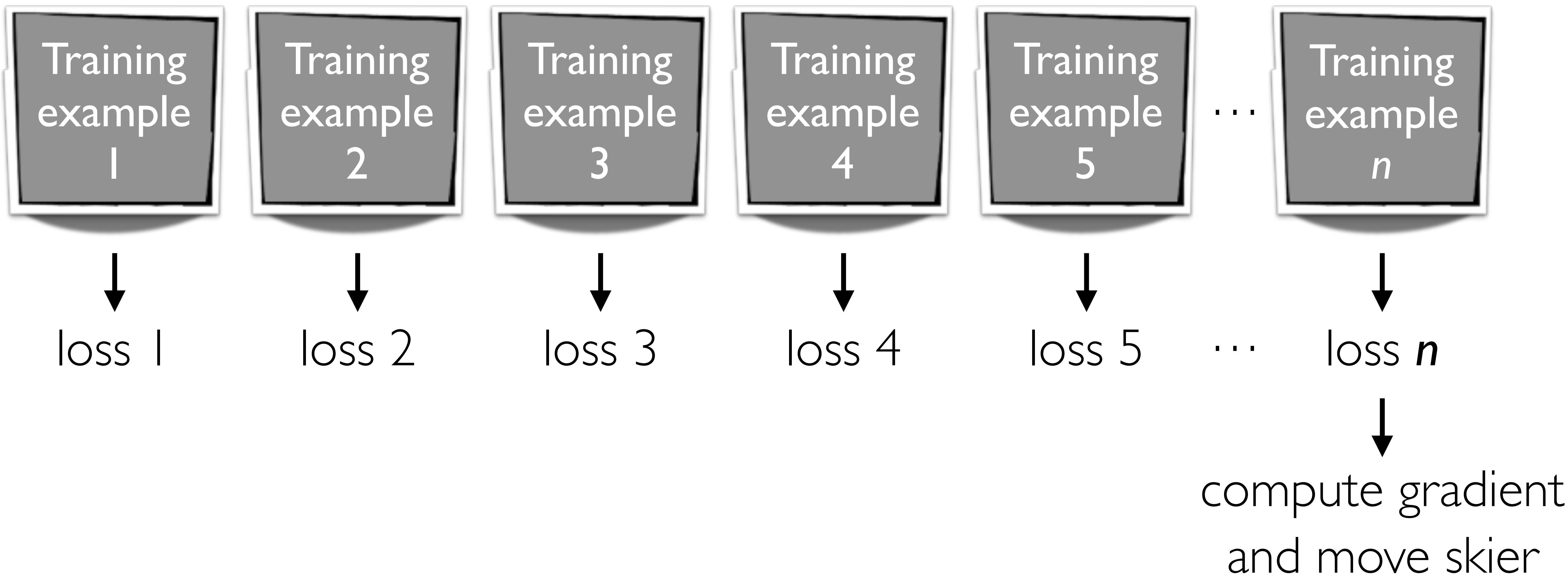
SGD: compute gradient using only 1 training example at a time
(can think of this gradient as a noisy approximation of the "full" gradient)

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)

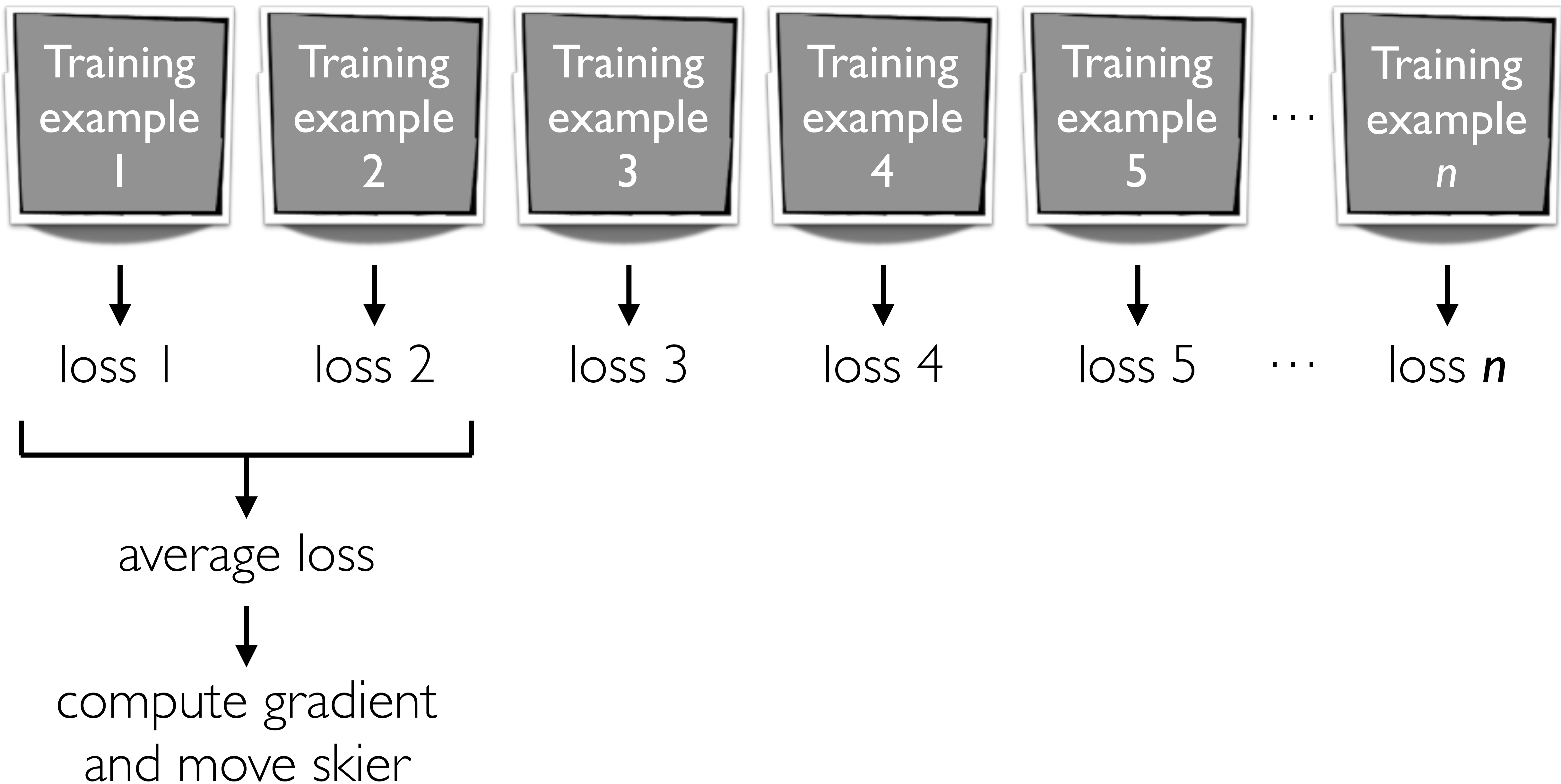
Stochastic Gradient Descent (SGD)

Training example 1	Training example 2	Training example 3	Training example 4	Training example 5	. . .	Training example *n*
↓	↓	↓	↓	↓		↓
loss 1	loss 2	loss 3	loss 4	loss 5	. . .	loss ***n***
↓						
compute gradient and move skier						

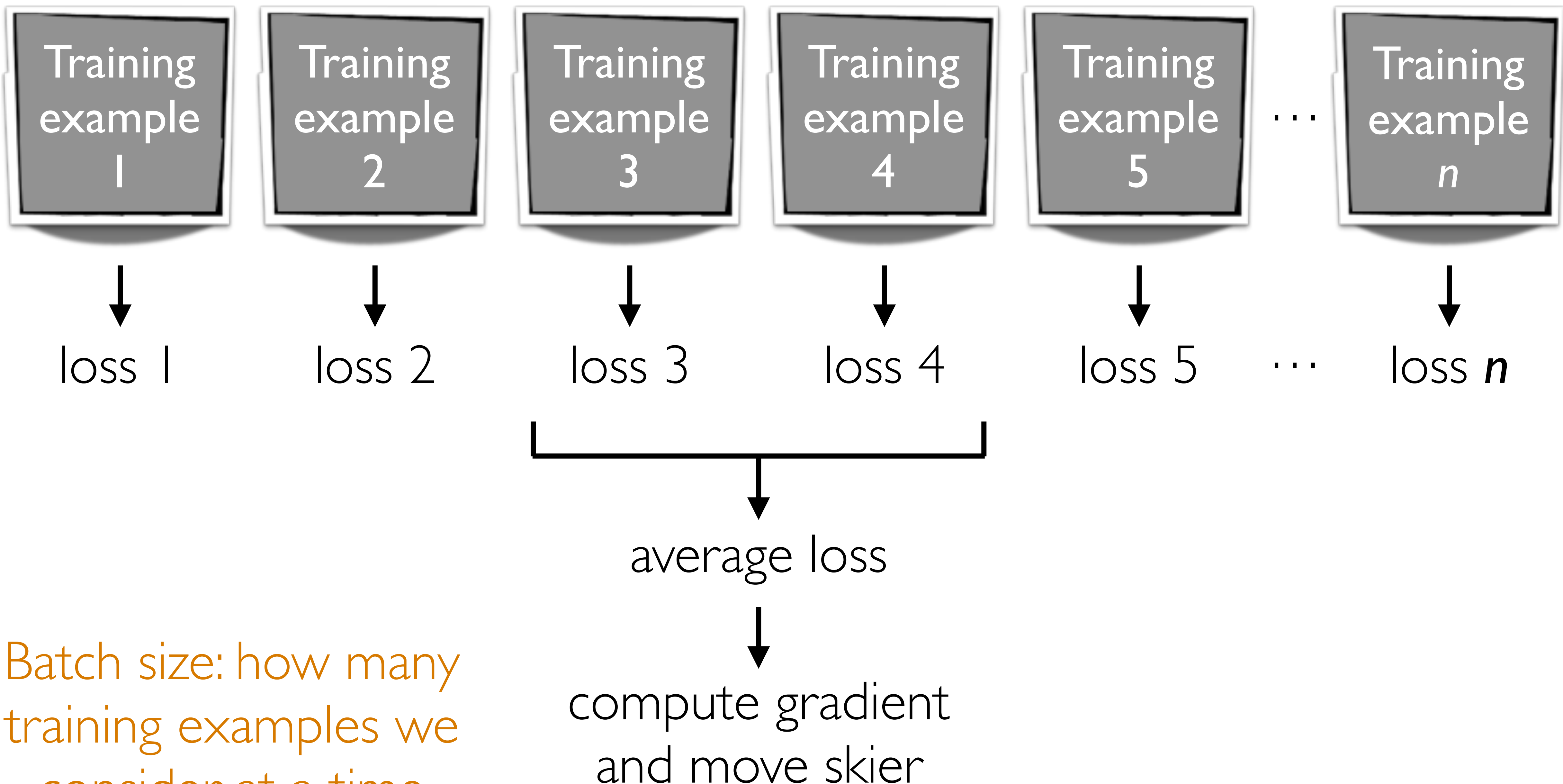
An epoch refers to 1 full pass through all the training data

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

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!!!

Dealing with Small Datasets

Data Augmentation

Generate perturbed versions of your training data to get a larger training dataset

Training image

Training label: cat

Mirrored

Still a cat!

Rotated & translated

Still a cat!

We just turned 1 training example in 3 training examples

Allowable perturbations depend on data (e.g., for handwritten digits, rotating by 180 degrees would be bad: confuse 6's and 9's)

Fine Tuning

If there's an existing pre-trained neural net, you could modify it for your problem that has a small dataset

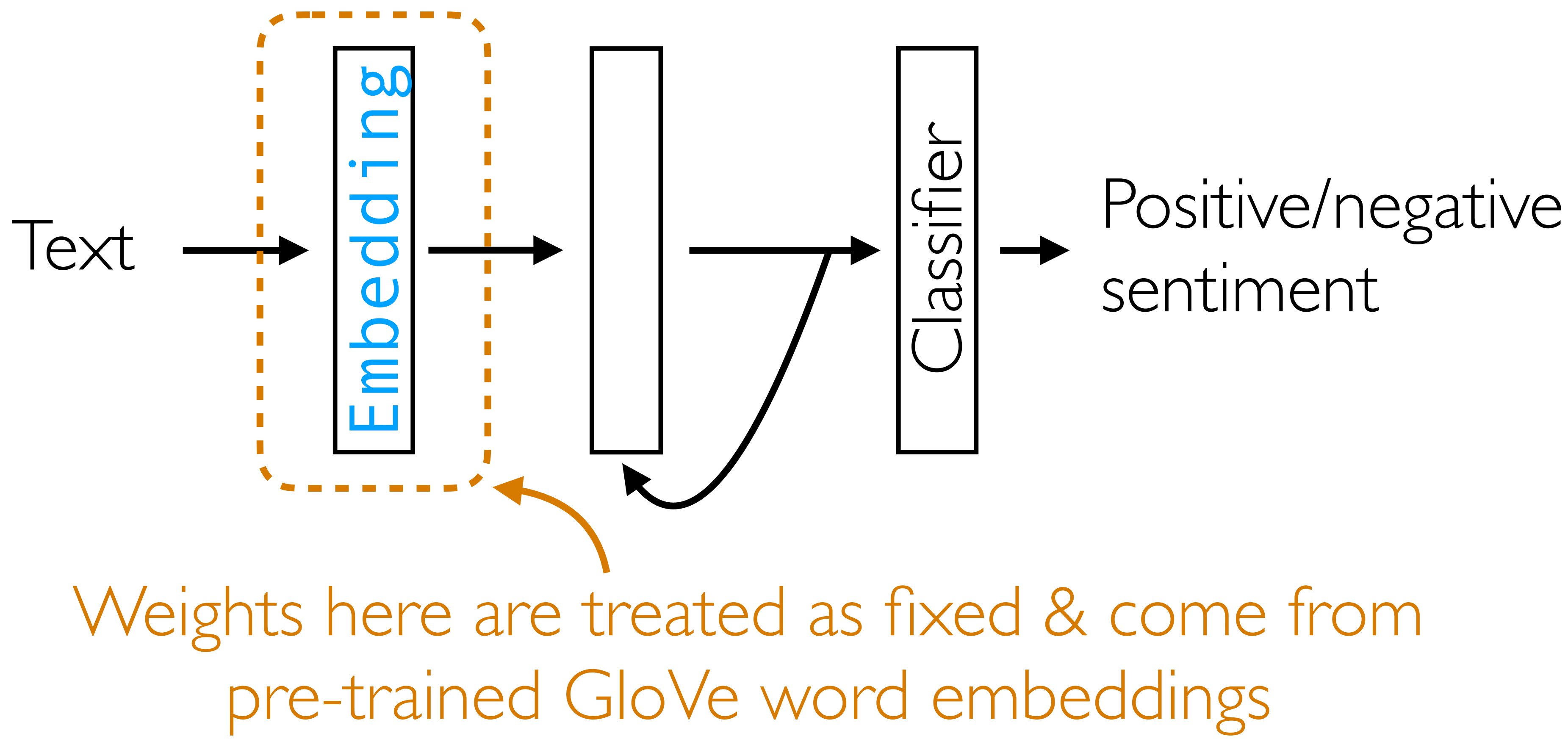
Example: classify between Tesla's and Toyota's

You collect photos from the internet of both, but your dataset size is small, on the order of 1000 images

Strategy: take pre-trained convnet (such as the state-of-the-art ResNet) for ImageNet classification and change final layers to do classification between Tesla's and Toyota's instead of classifying 1000 objects

Fine Tuning

Sentiment analysis RNN demo



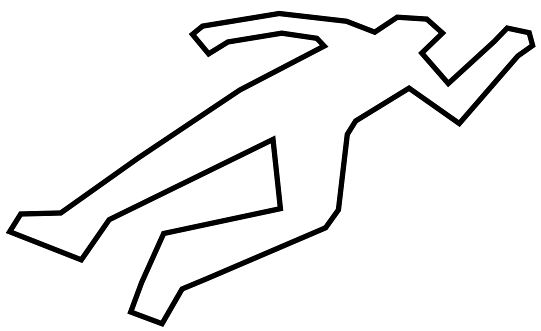
GloVe vectors pre-trained on massive dataset (Wikipedia + Gigaword)

IMDb review dataset is small in comparison

For more, check out the recording of the Pittsburgh lecture next week!

Unstructured Data Analysis

Question



The dead body

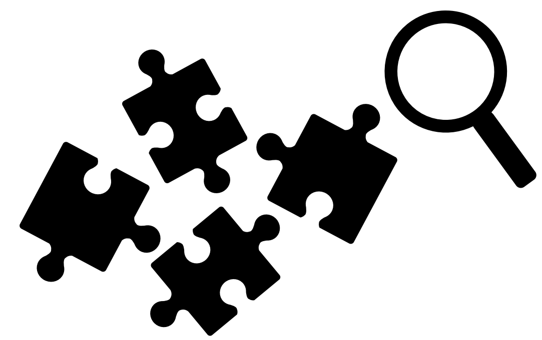
This is provided by a practitioner

Data

The evidence

Some times you have to collect more evidence!

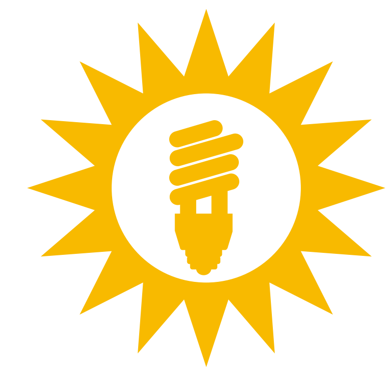
Finding Structure



Puzzle solving, careful analysis

Exploratory data analysis

Insights



When? Where? Why? How? Perpetrator catchable?

Answer original question

<u>There isn't always a follow-up prediction problem to solve</u>

Some Parting Thoughts

- Remember to **visualize steps of your data analysis pipeline**
  - Helpful in debugging & interpreting intermediate/final outputs
- Very often there are *tons* of models/design choices to try
  - Come up with **quantitative metrics** that make sense for your problem, and use these metrics to **evaluate models (think about how we chose hyperparameters!)**
  - But don't blindly rely on metrics without **interpreting results in the context of your original problem!**
- Often times you won't have labels! If you really want labels:
  - Manually obtain labels (either you do it or crowdsource)
  - Set up "self-supervised" learning task (in Pittsburgh last lecture)
- There is a *lot* we did not cover — **keep learning!**

Want to Learn More?

- I posted a Canvas announcement some days ago with follow-up courses that are related to unstructured data analysis

- One of the best ways to learn material is to teach it!

  Apply to be a TA for me next term!