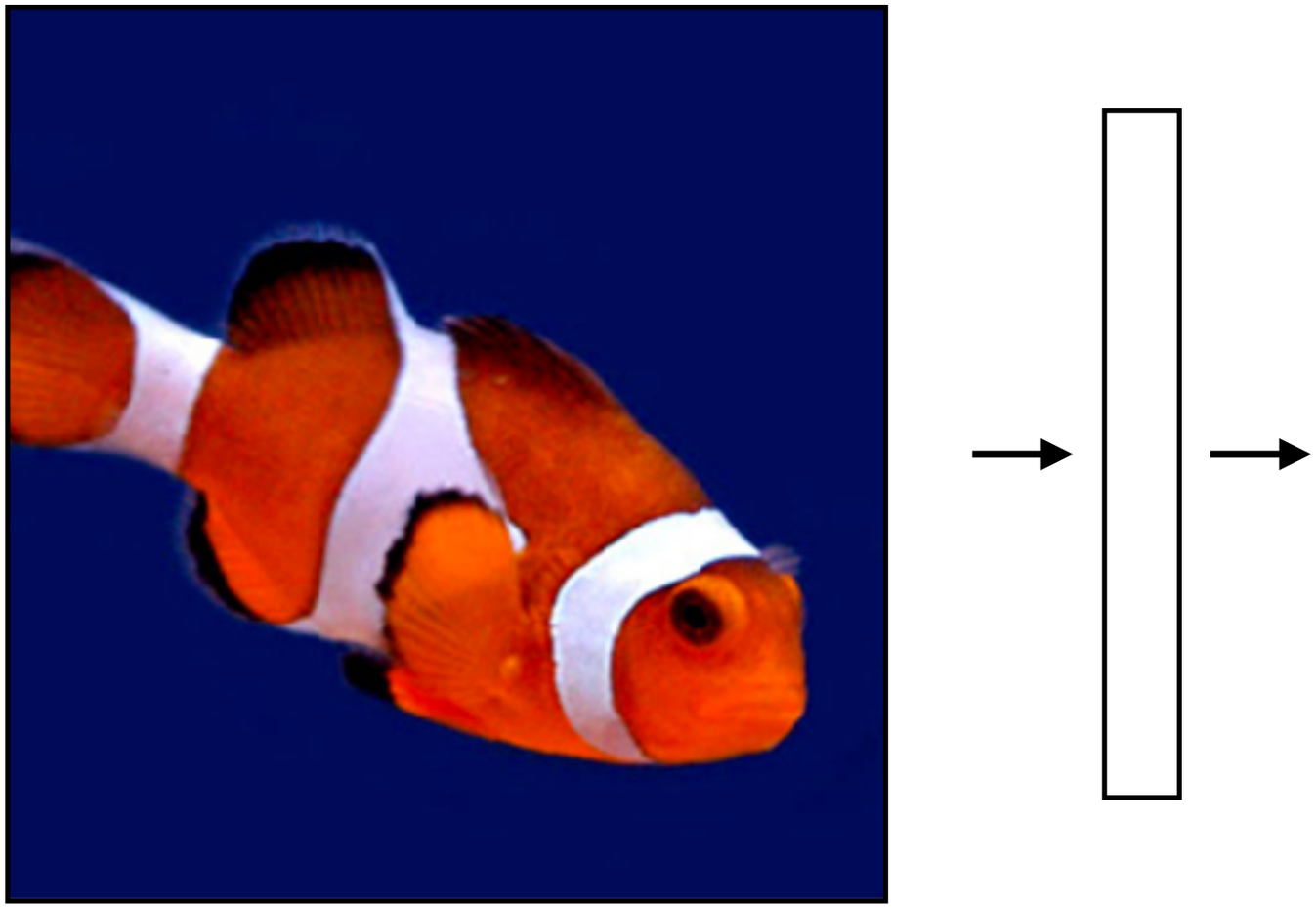
Carnegie Mellon University
Heinz College

94-775/95-865 Lecture 12: Time Series Analysis With Recurrent Neural Nets

George Chen

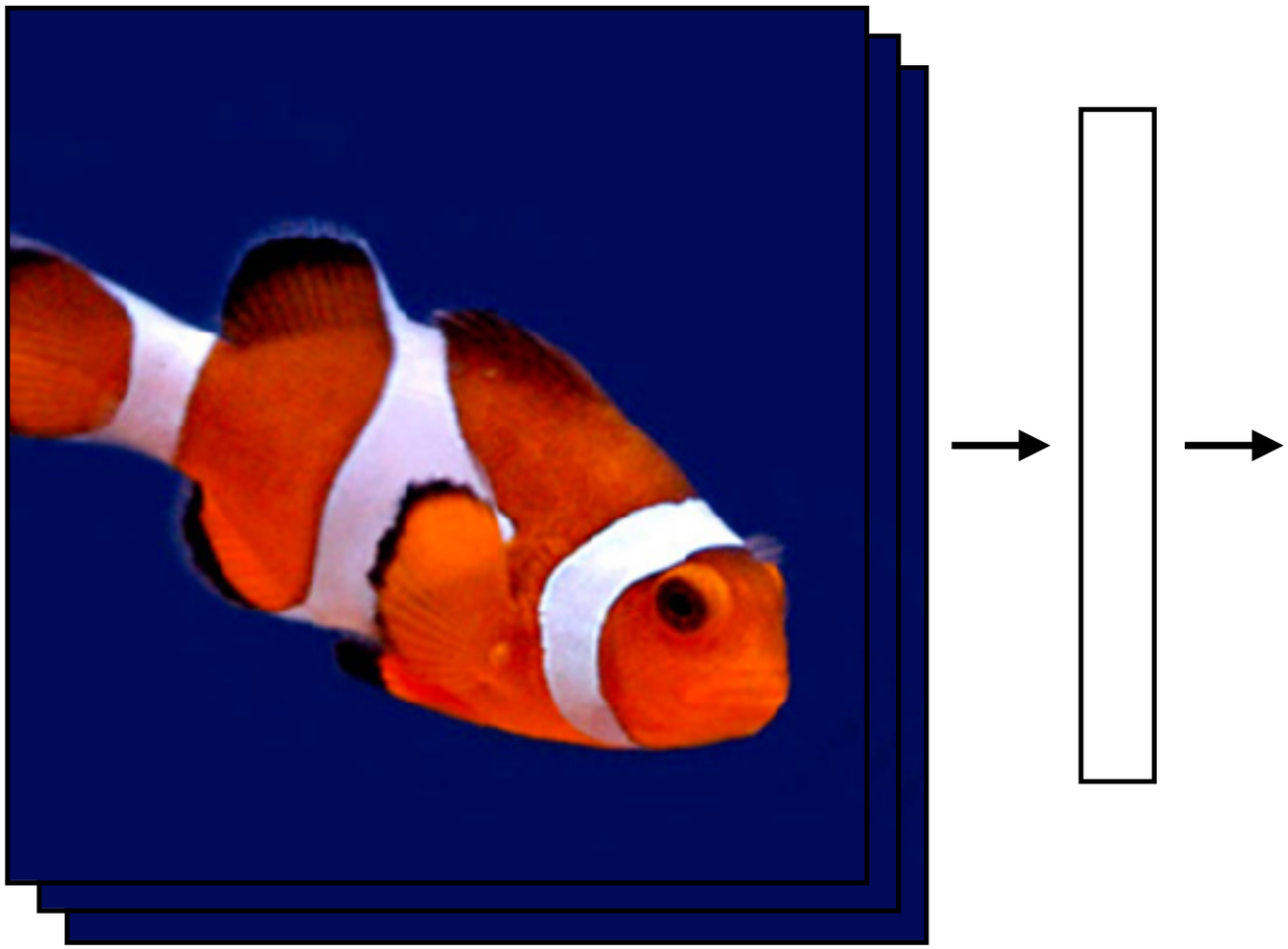
RNNs

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

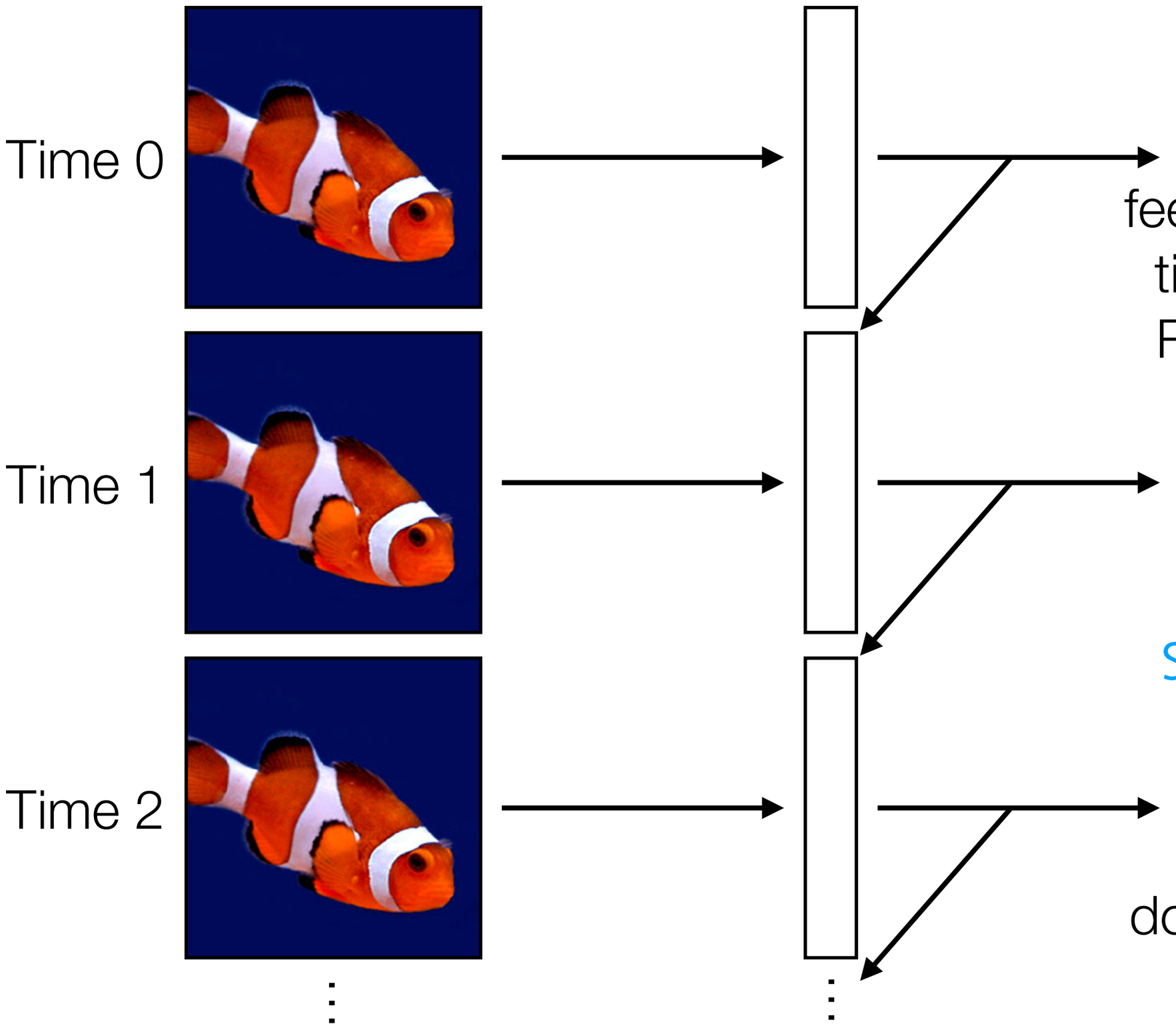
Feedforward NN's: treat each video frame separately

Time 0

Time 1

Time 2

RNNs



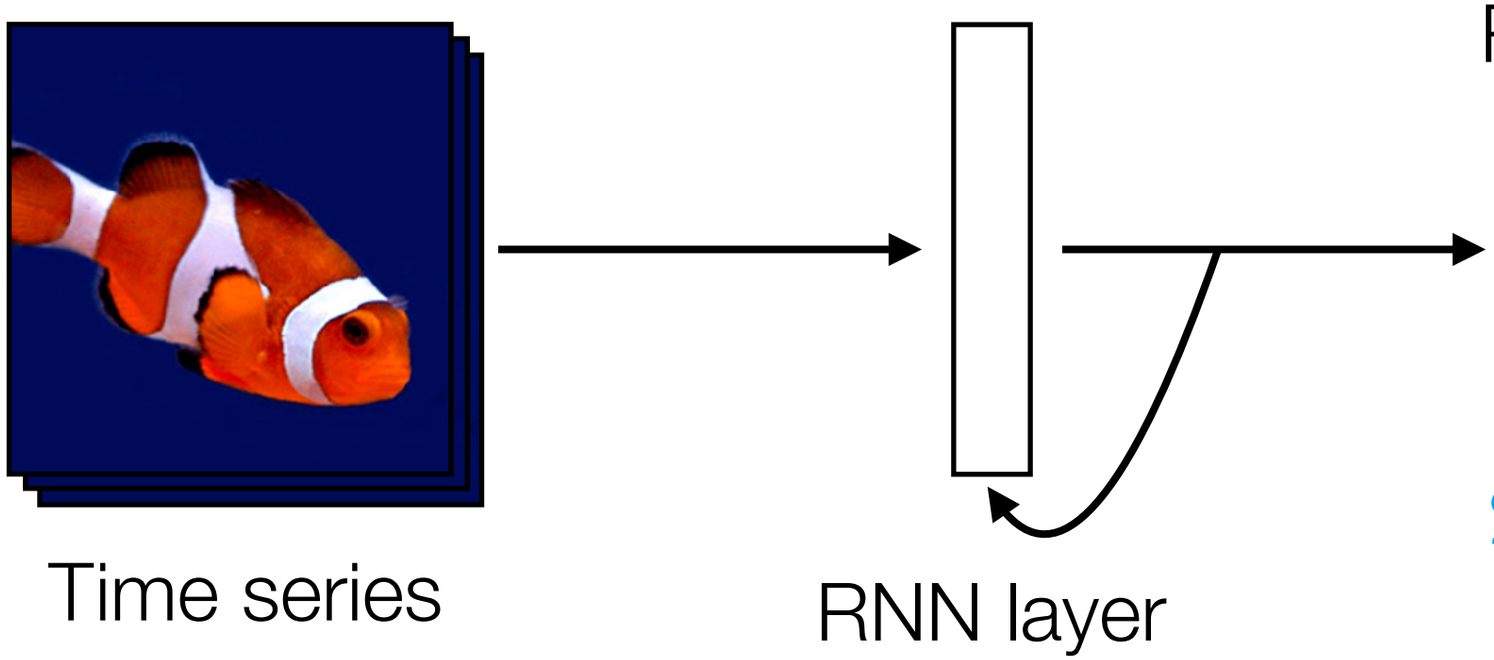
Feedforward NN's: treat each video frame separately

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



Time series

RNN layer

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`

Example: SimpleRNN

memory stored in `current_state` variable!

```
current_state = 0

for input in input_sequence:

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

    current_state = output
```

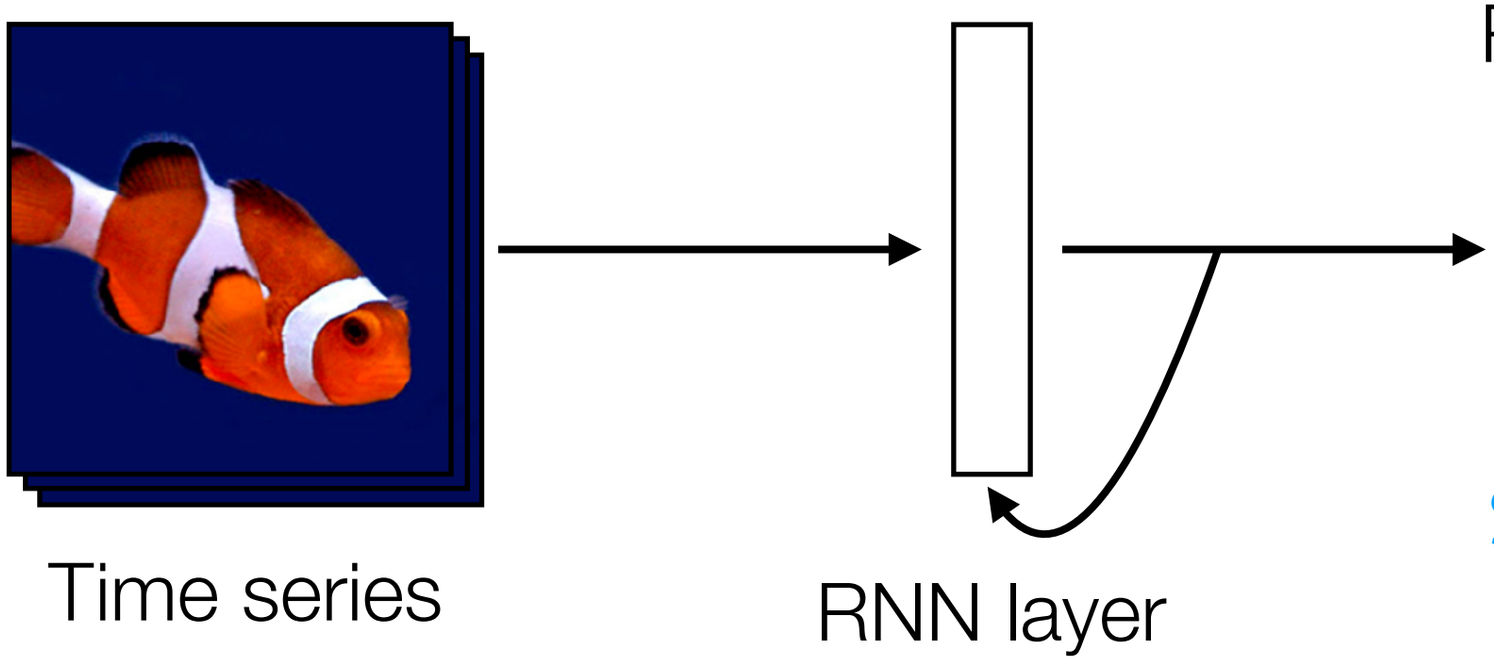
Activation function could, for instance, be ReLU

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

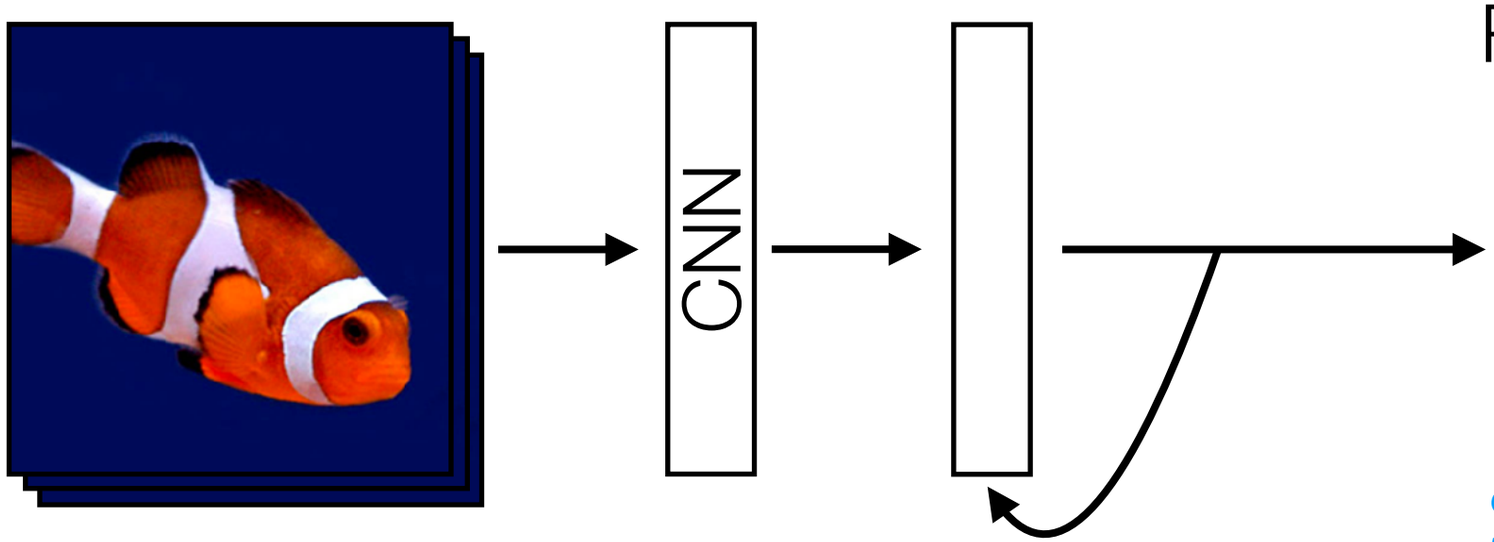
RNN's: feed output at previous time step as input to RNN layer at current time step

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

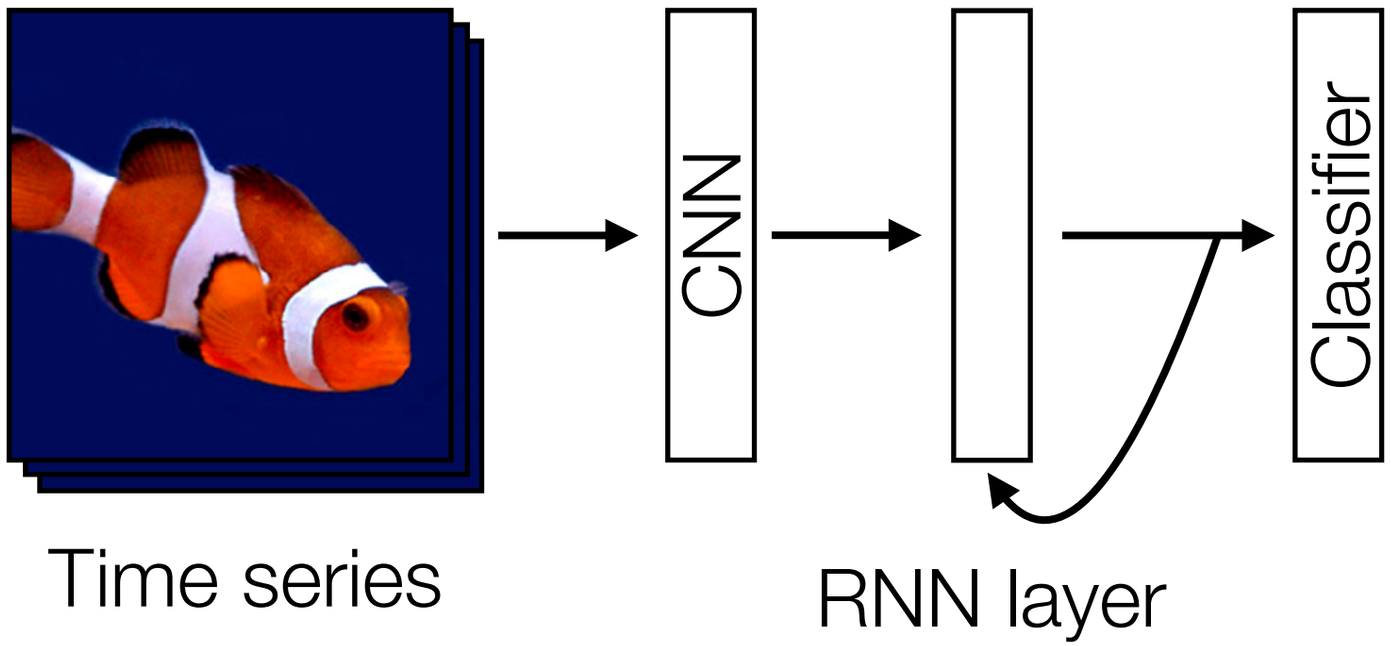
RNN's: feed output at previous time step as input to RNN layer at current time step

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

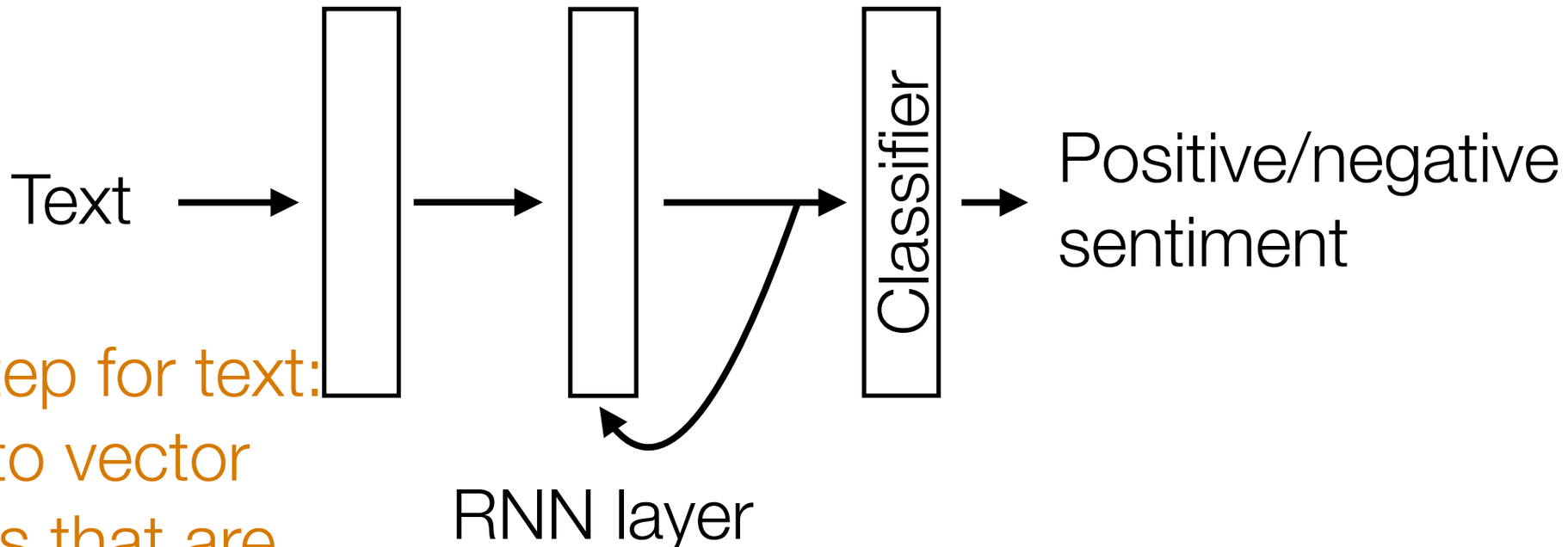
RNN's: feed output at previous time step as input to RNN layer at current time step

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) Do Data Actually Live on Manifolds?

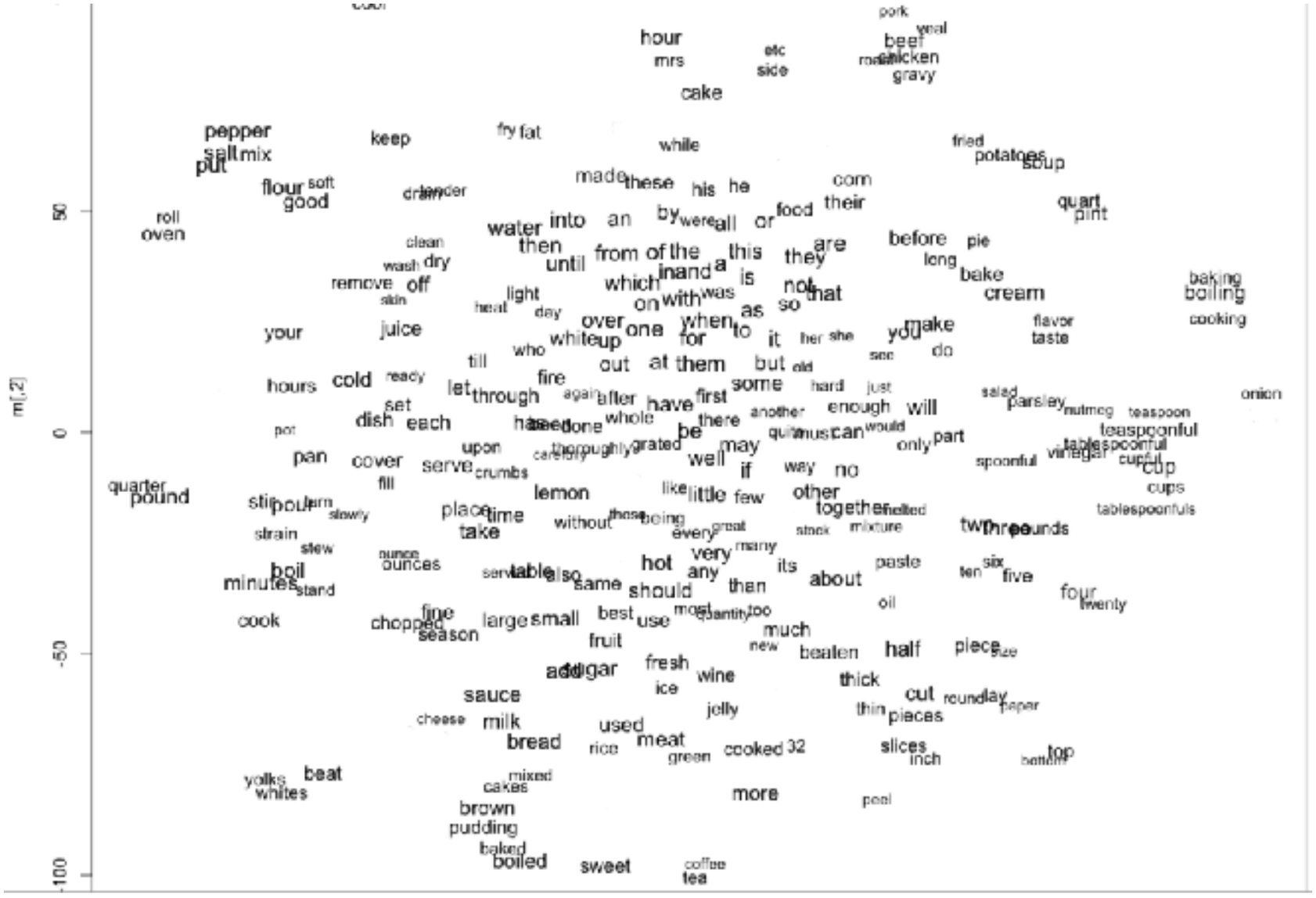
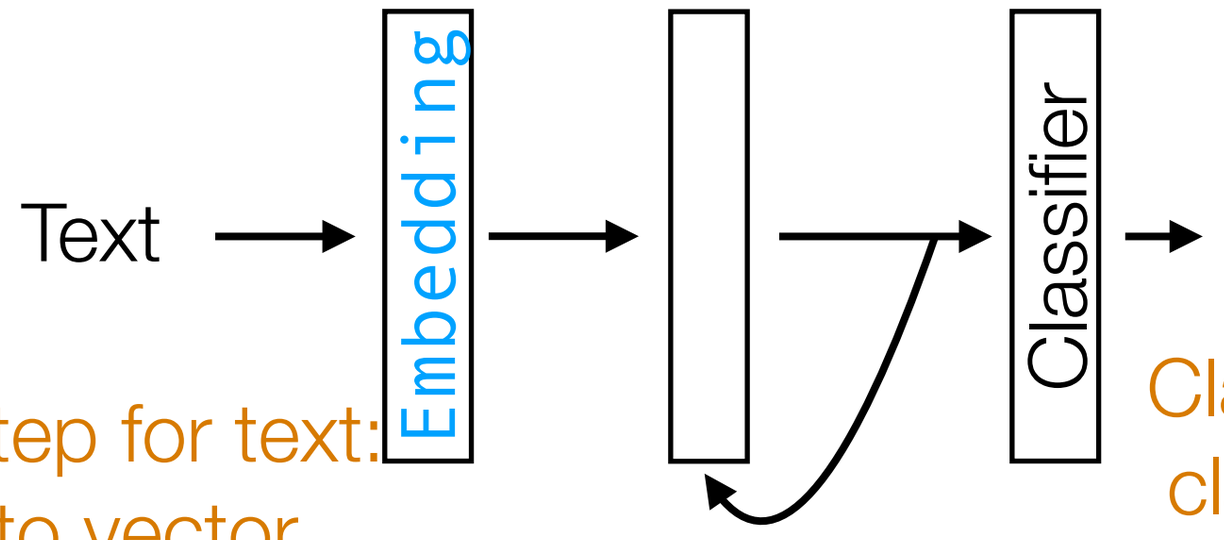


Image source: http://www.adityathakker.com/wp-content/uploads/2017/06/word-embeddings-994x675.png

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!

I'll talk more about self-supervised learning next lecture
(it's a clever application of predictive data analytics concepts)

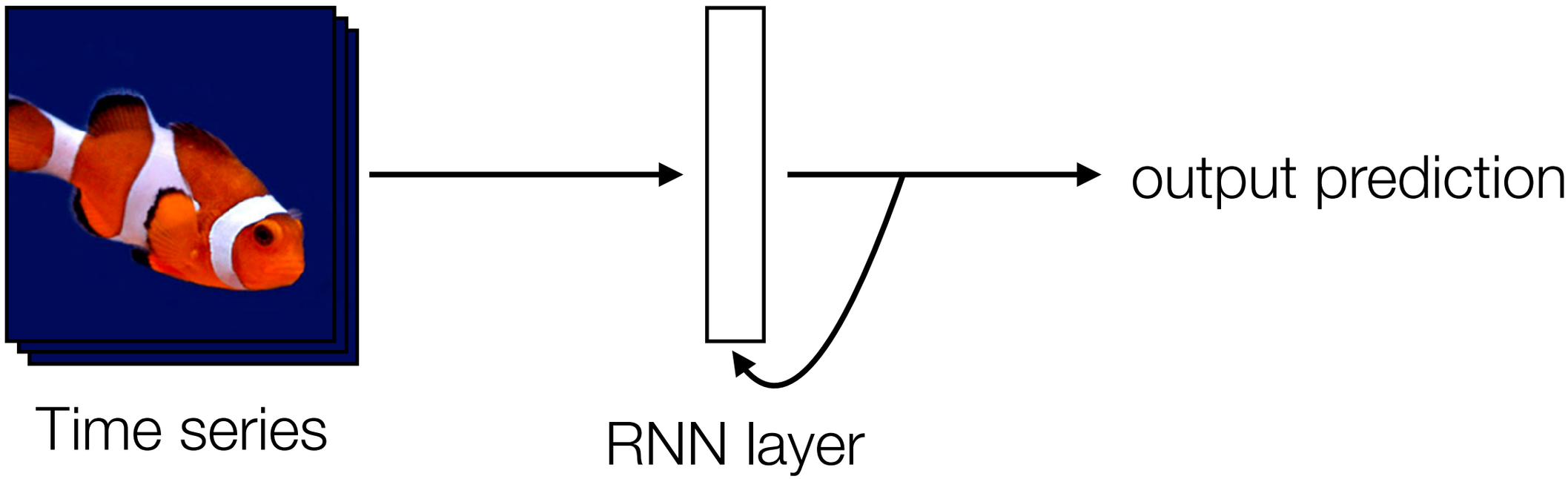
RNNs

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

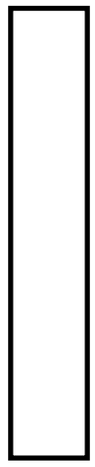


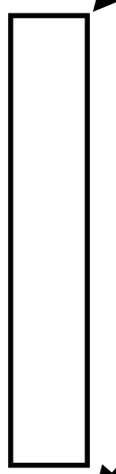
Time 0

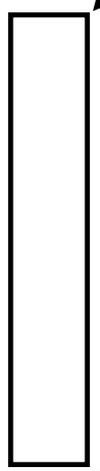
Time 1

Time 2

⋮





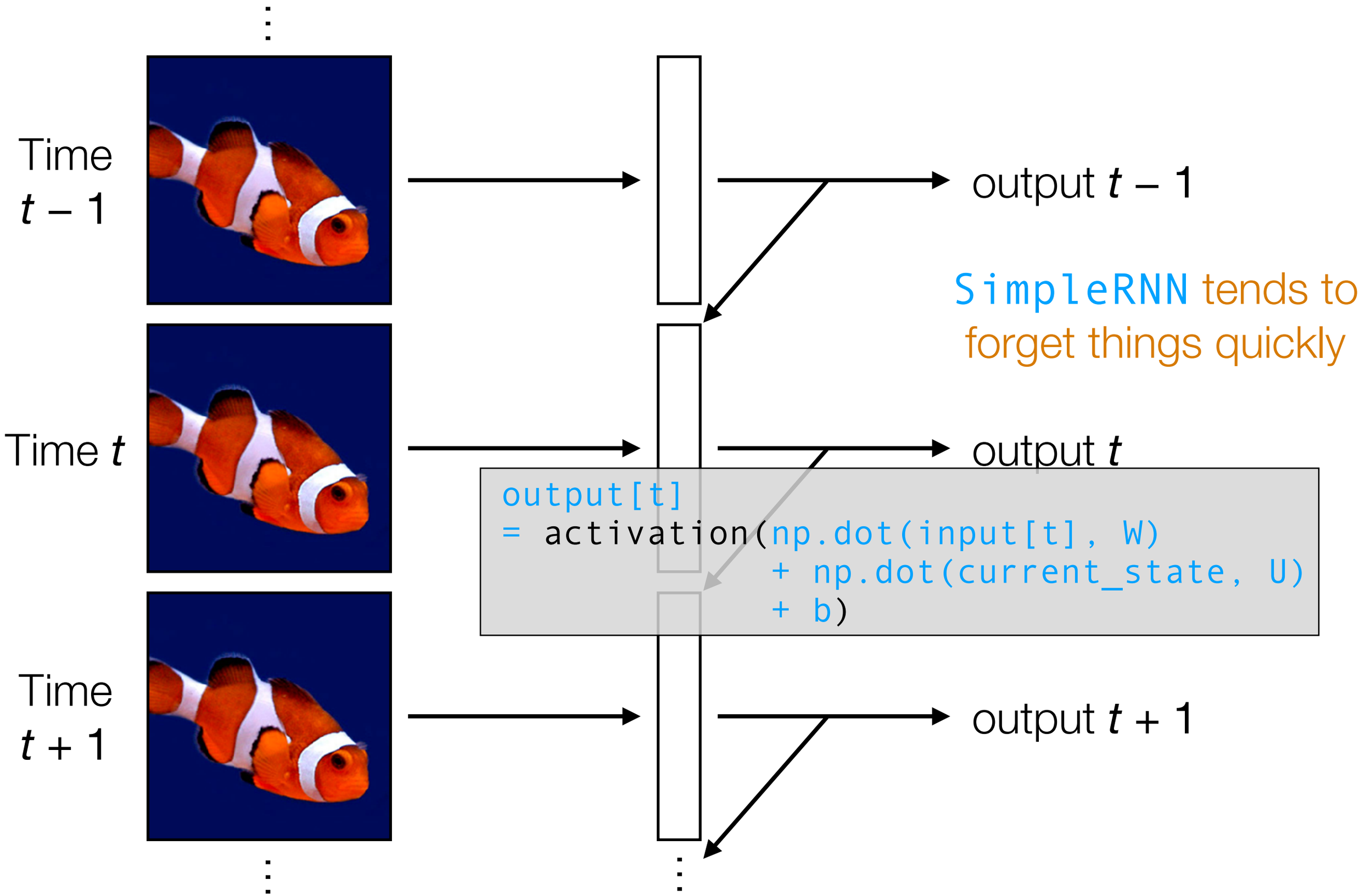


⋮

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(input[t], W)
+ np.dot(current_state, U)
+ b)

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

Time $t + 1$

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

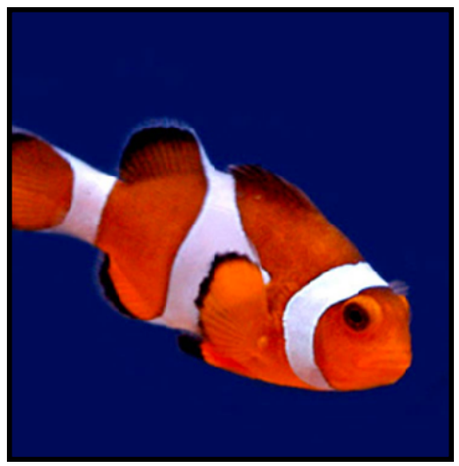
Add explicit long-term memory!

Time $t - 1$

output $t - 1$

But need some way to update long-term memory!

Time t



output t

Long-term memory

Add explicit long-term memory!

Time $t-1$

output $t-1$

But need some way to update long-term memory!

Long-term memory updater

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

Remembers things longer than `SimpleRNN`

Time t

output t