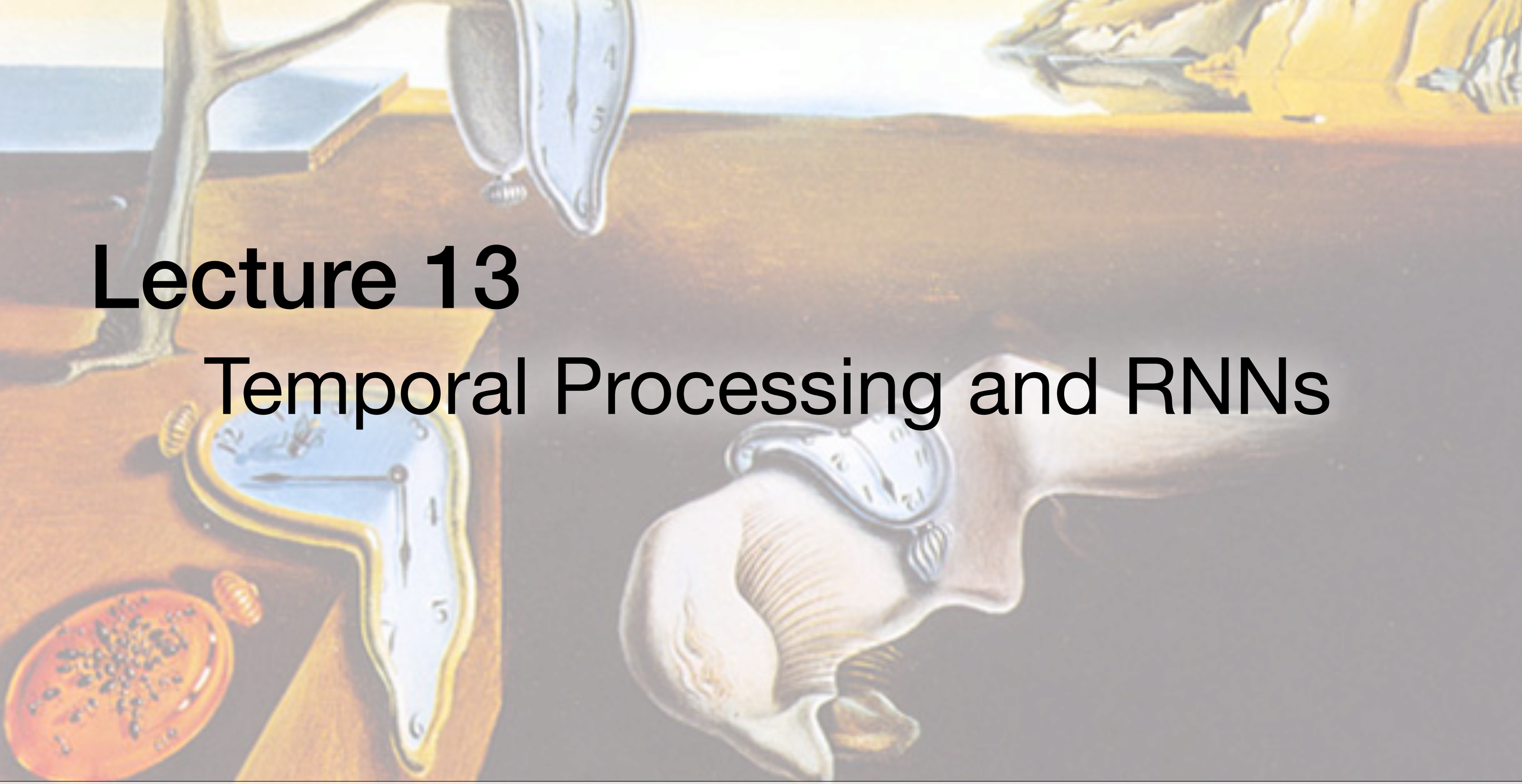


Lecture 13

Temporal Processing and RNNs

Bill Freeman, Phillip Isola

13. Temporal Processing and RNNs

- Sequence problems
- Temporal convnets
- Recurrent Neural Networks (RNNs)
- LSTMs
- Attention
- Example problems:
  - Image captioning
  - Sound prediction

Announcements

- Start thinking about final project ideas
- Information and some suggested topics here: http://6.869.csail.mit.edu/sp22/project.html
- Proposals due March 31st


kindergarden classroom

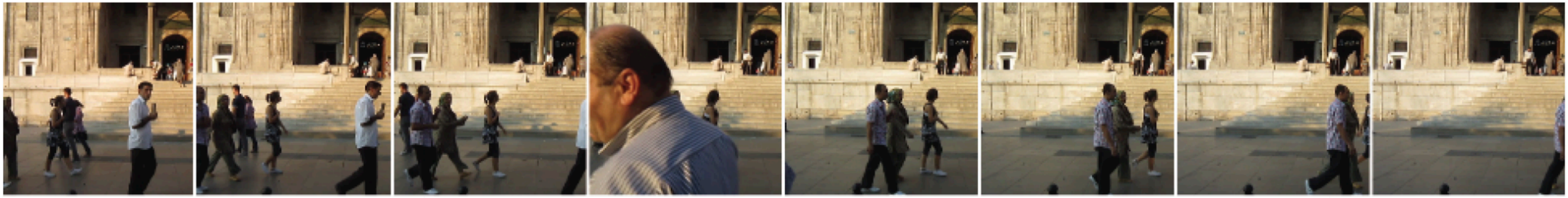

person
television
chair


“What color is the chair?”


“What color is the chair?”
red


“What will the girl do next?”

Sequences



time

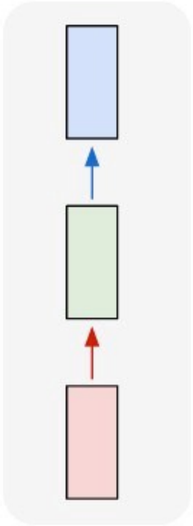
“An”, “evening”, “stroll”, “through”, “a”, “city”, “square”

time

time

How do we model sequences?

one to one

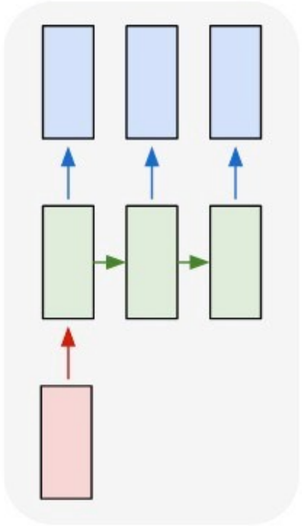


Input: No sequence

Output: No sequence

Example: “standard” classification / regression problems

one to many

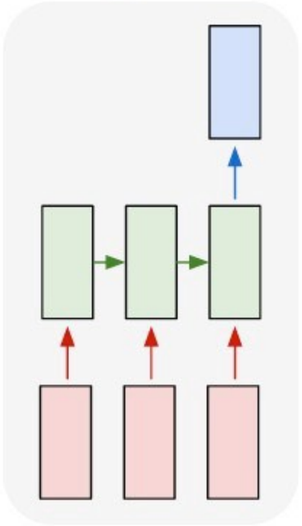


Input: No sequence

Output: Sequence

Example: Im2Caption

many to one

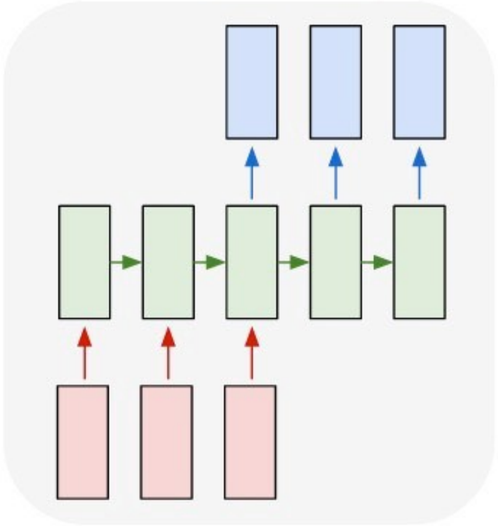


Input: Sequence

Output: No sequence

Example: sentence classification, multiple-choice question answering

many to many

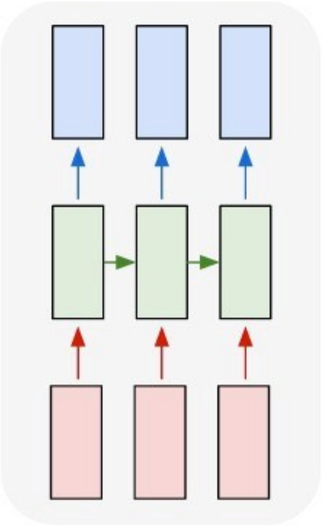


Input: Sequence

Output: Sequence

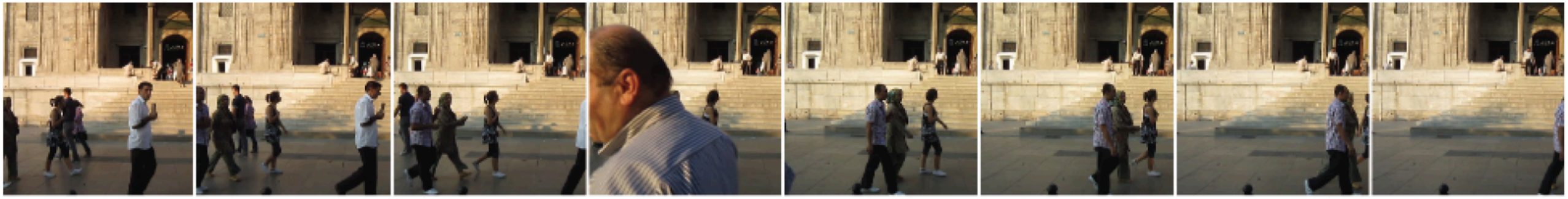
Example: machine translation, video captioning, open-ended question answering, video question answering

many to many

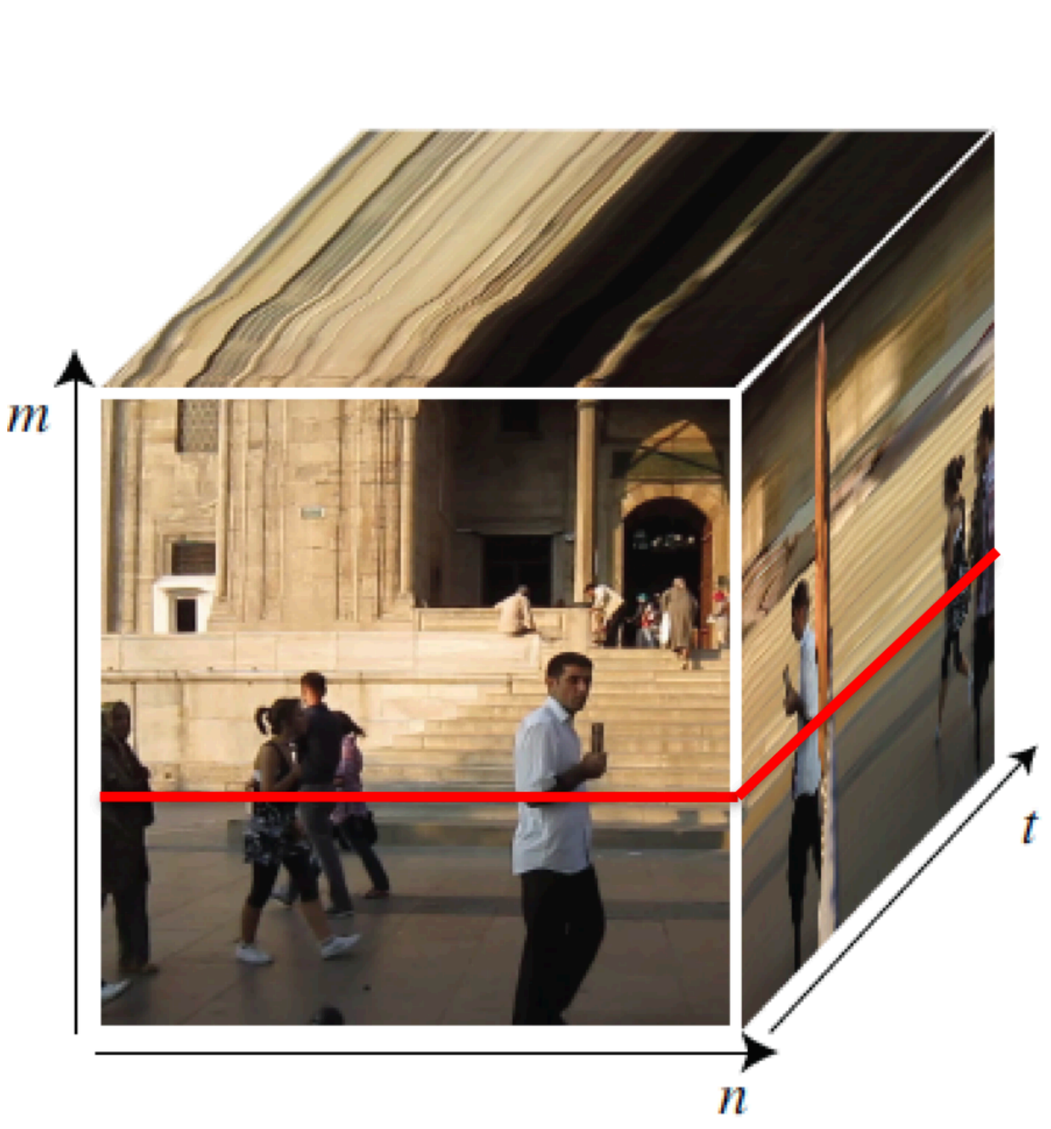


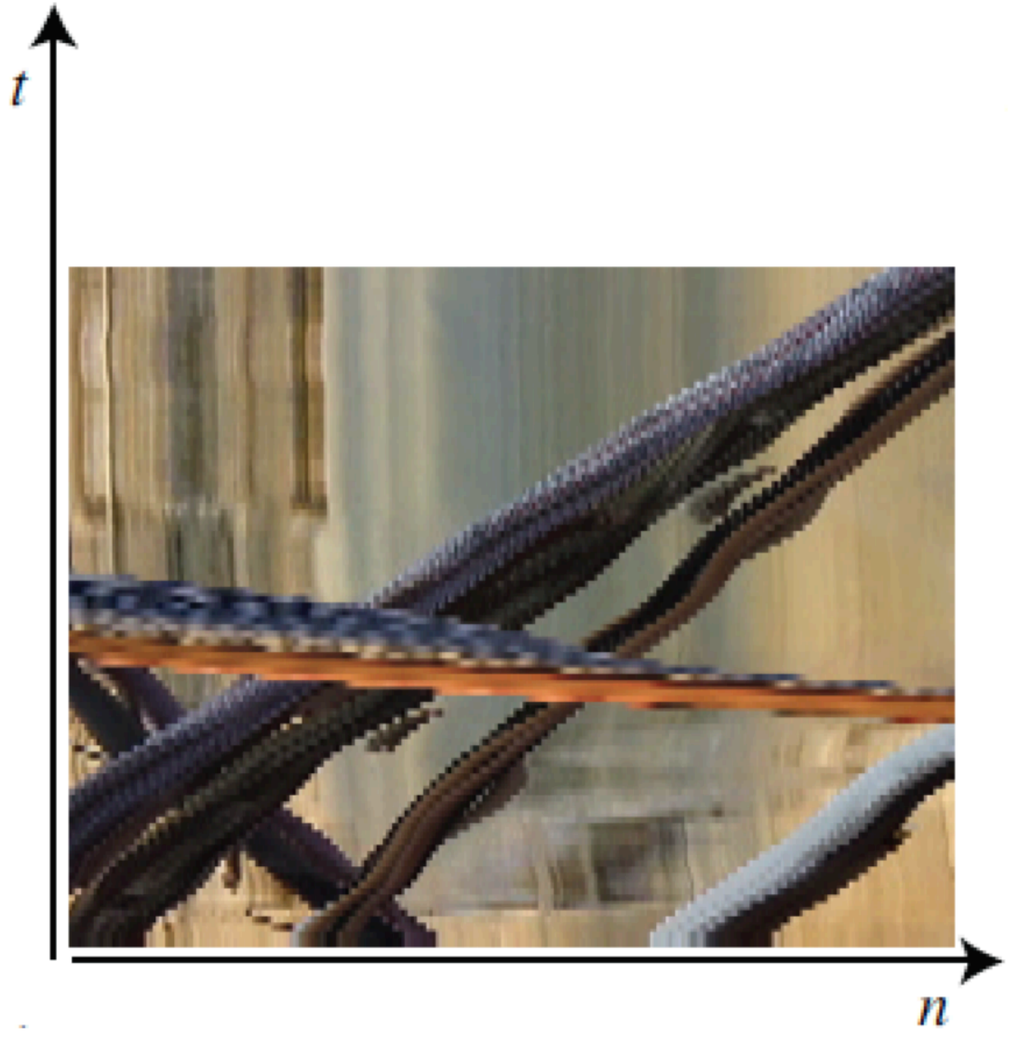
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

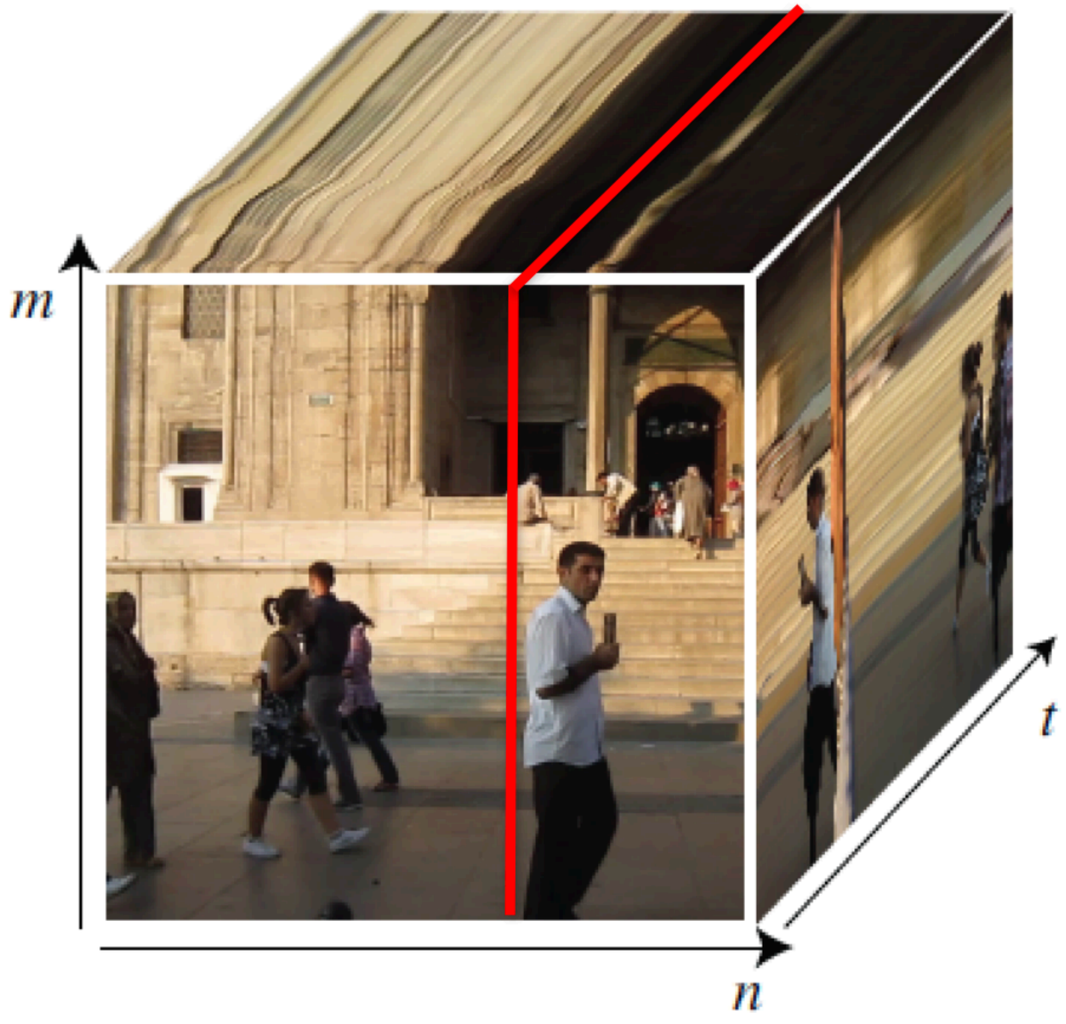
Credit: Dhruv Batra, Andrej Karpathy



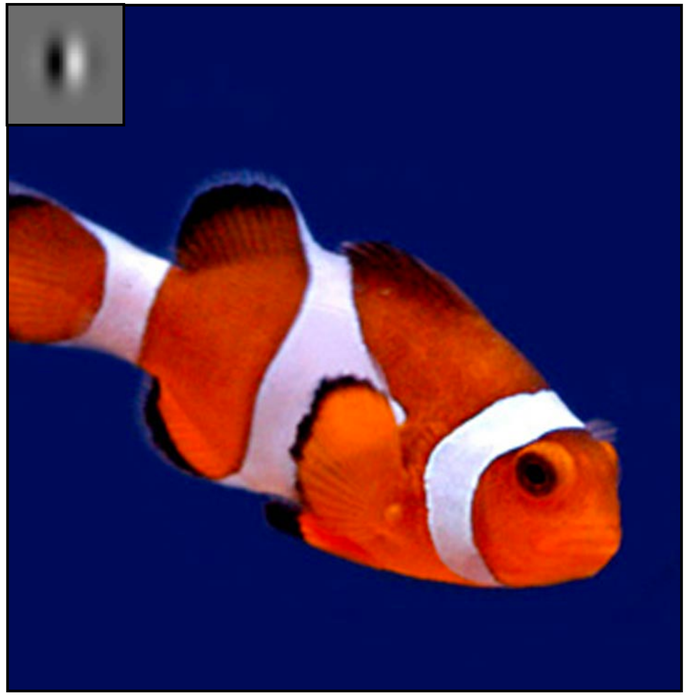

m
n
t


m
n
t

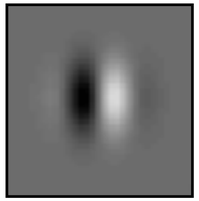

t
n


m
n
t

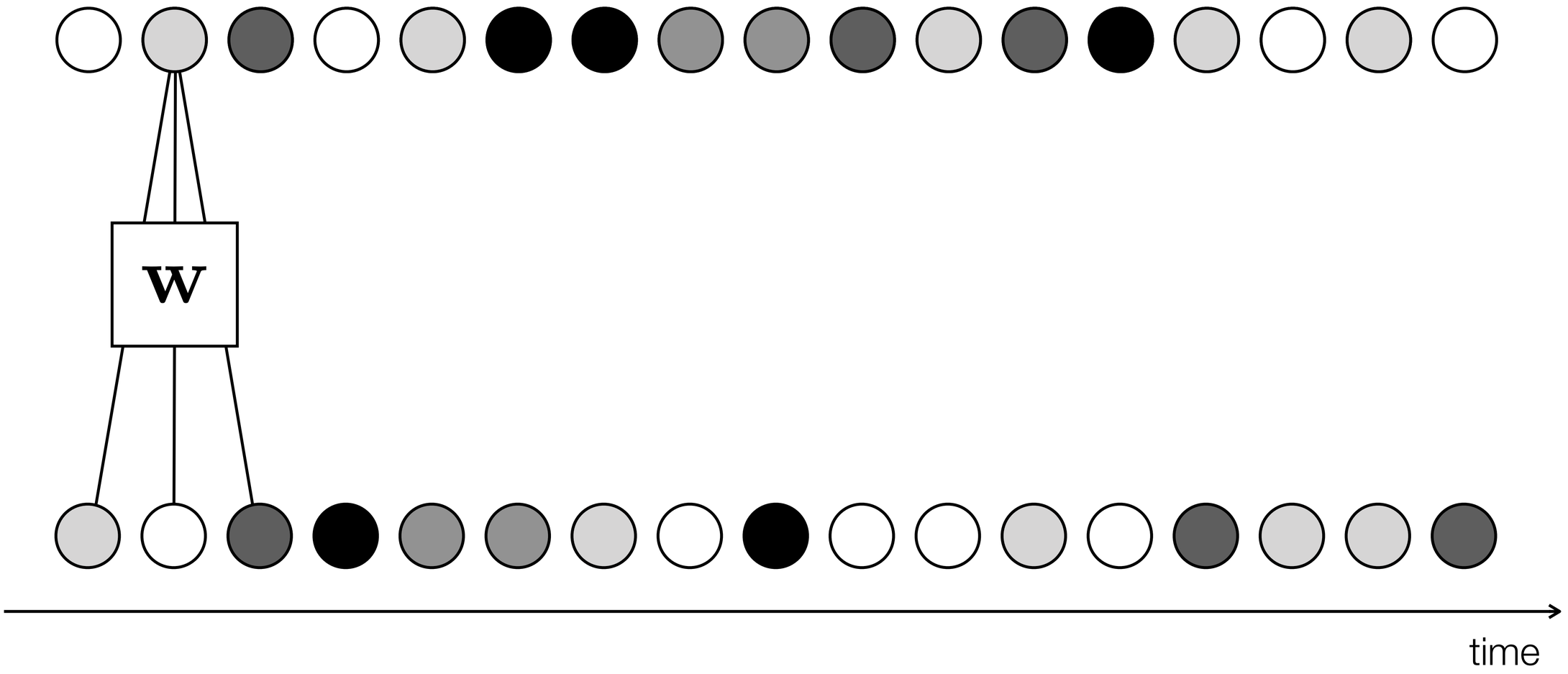

m
t

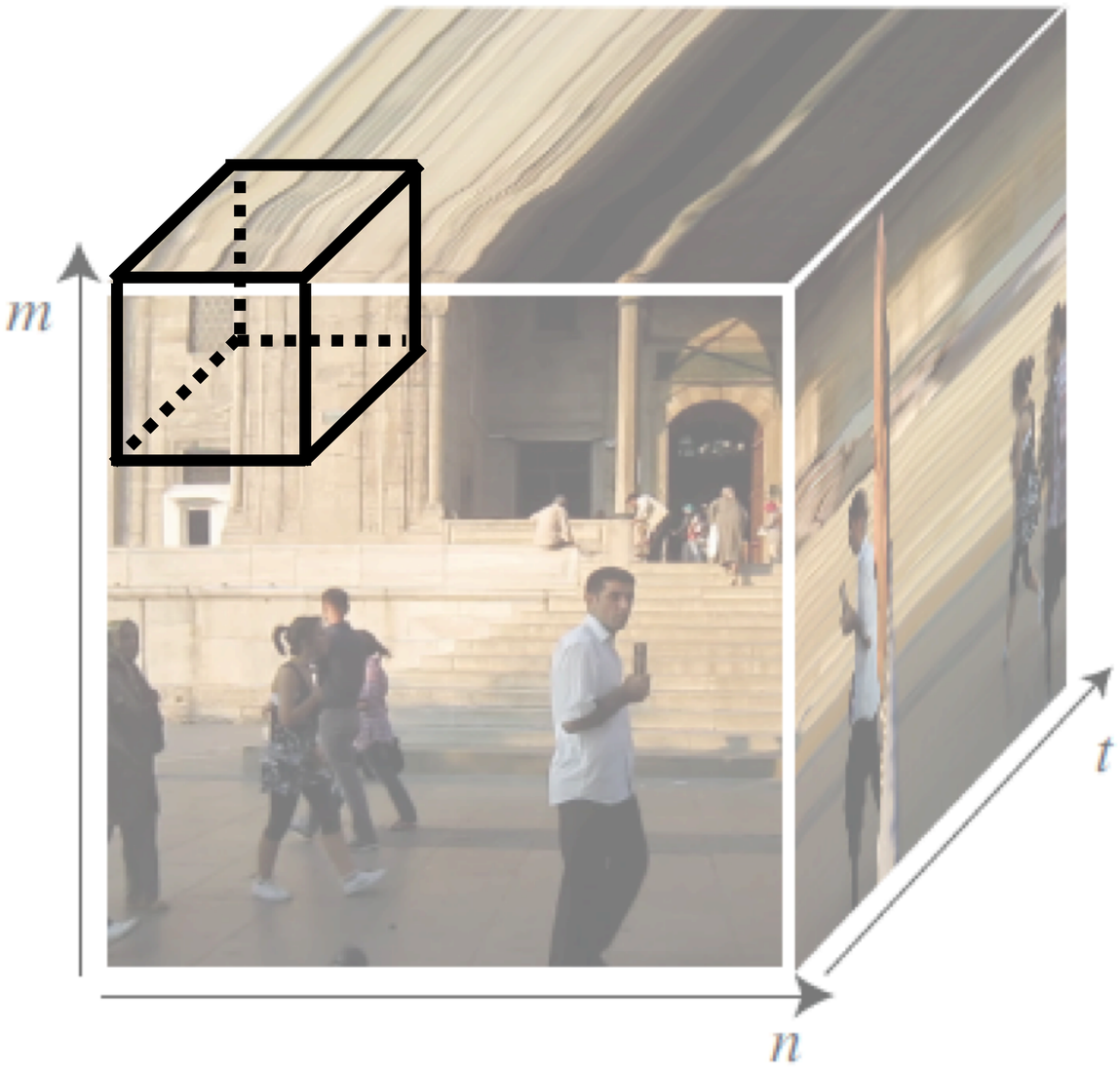


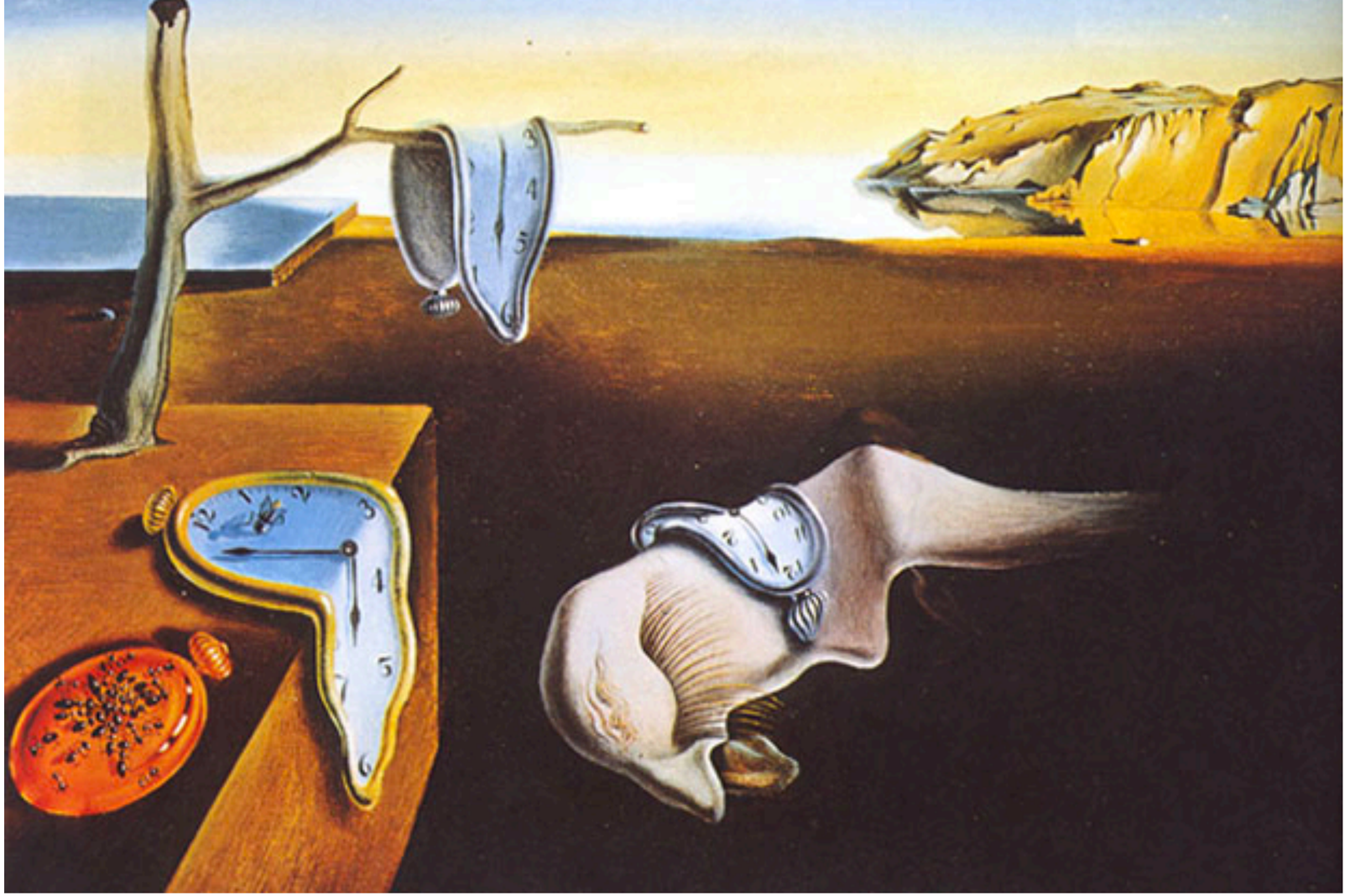
filter



Convolutions in time



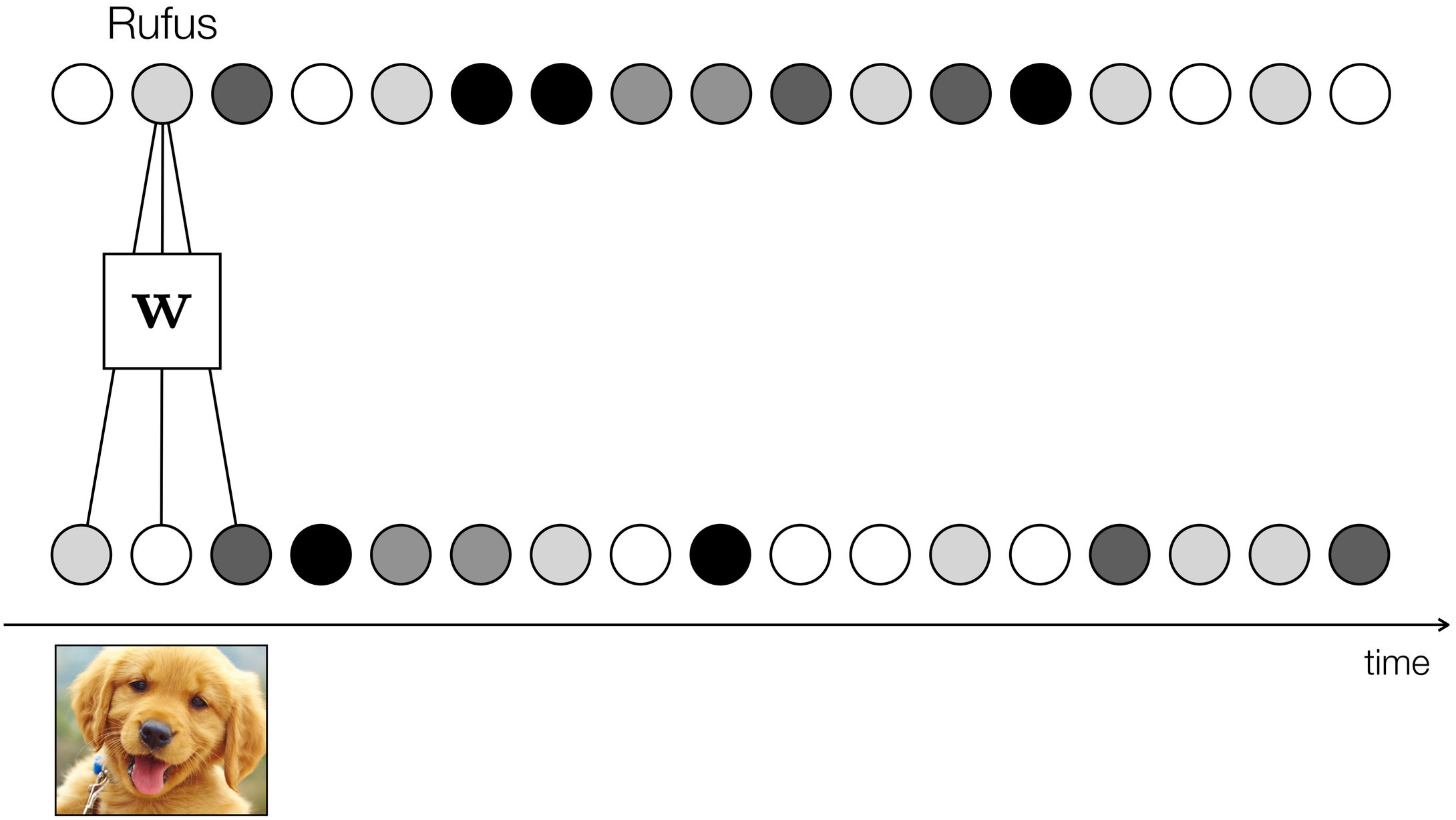

m
n
t

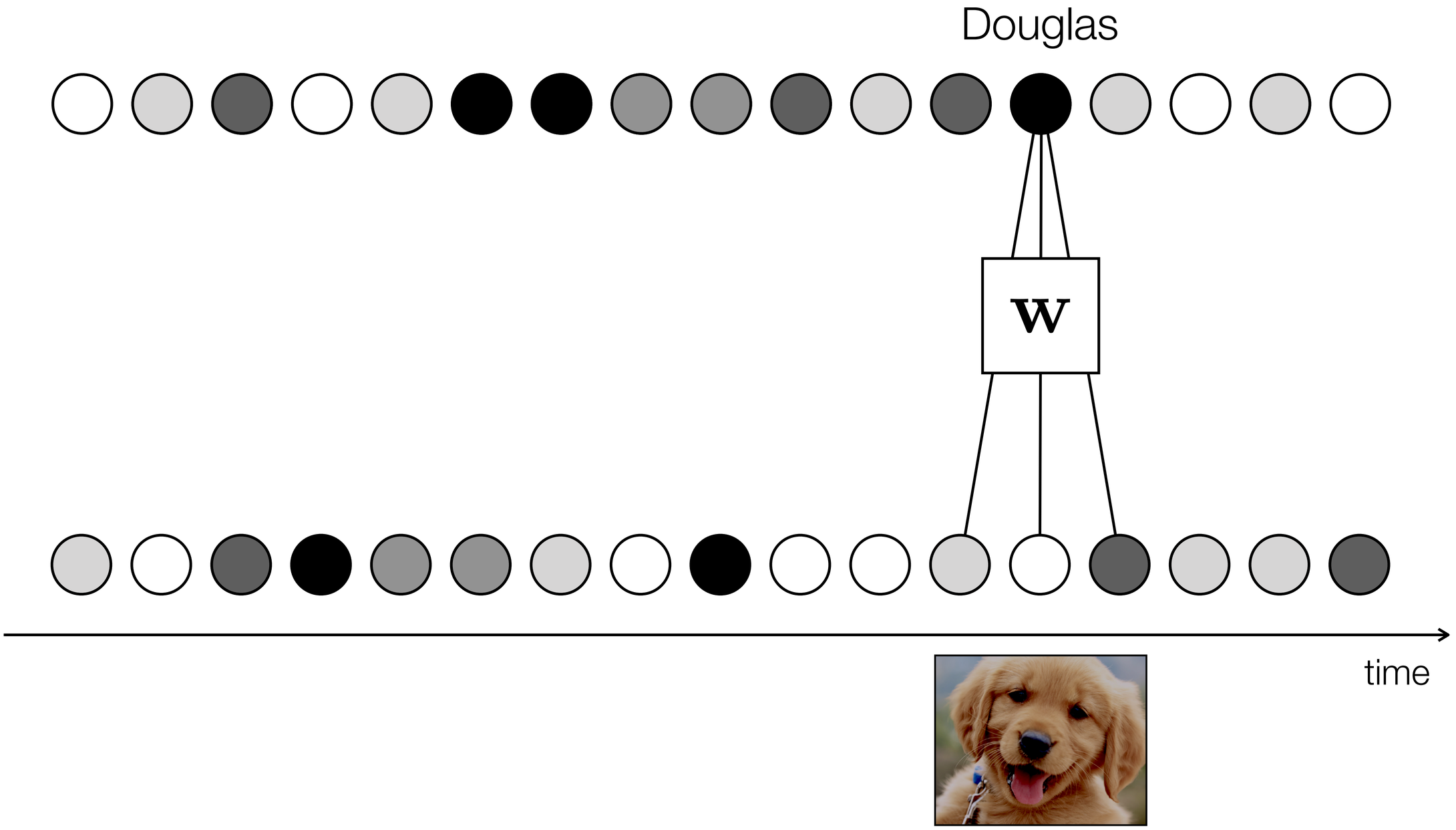


“The Persistence of Memory”, Dali 1931

It bothered him that the dog at three fourteen (seen from the side) should have the same name as the dog at three fifteen (seen from the front).

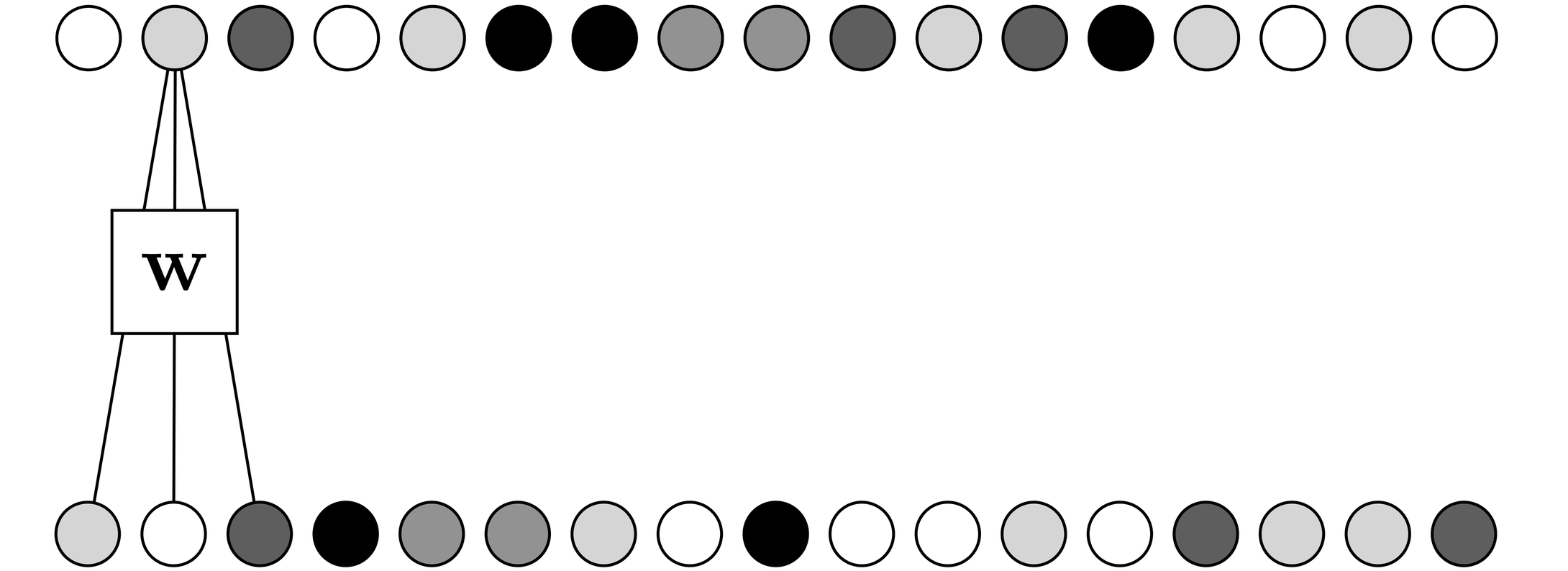
— “Funes the Memorius”, Borges 1962


Rufus
W
time

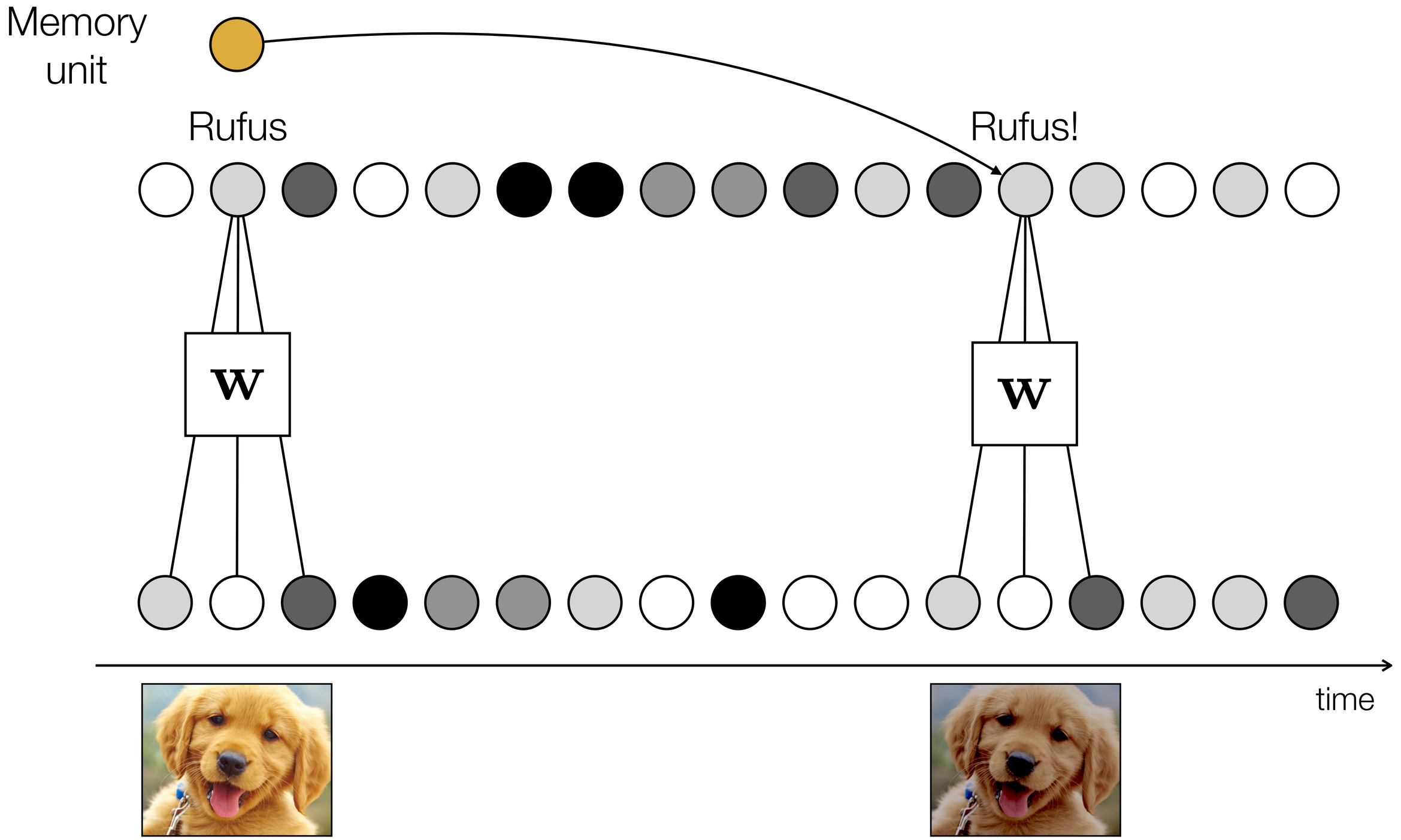

Douglas
W
time

Memory unit

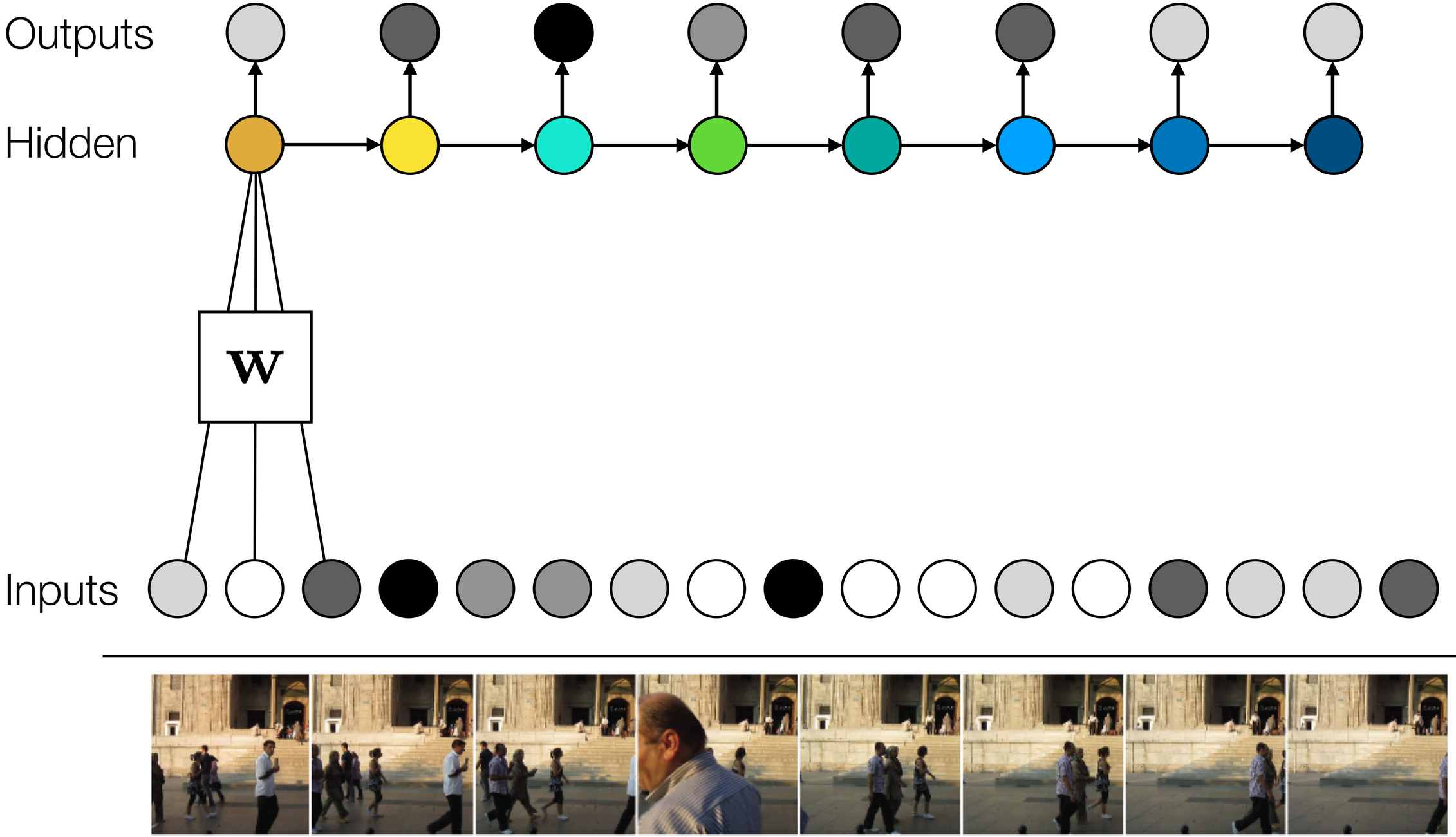
Rufus



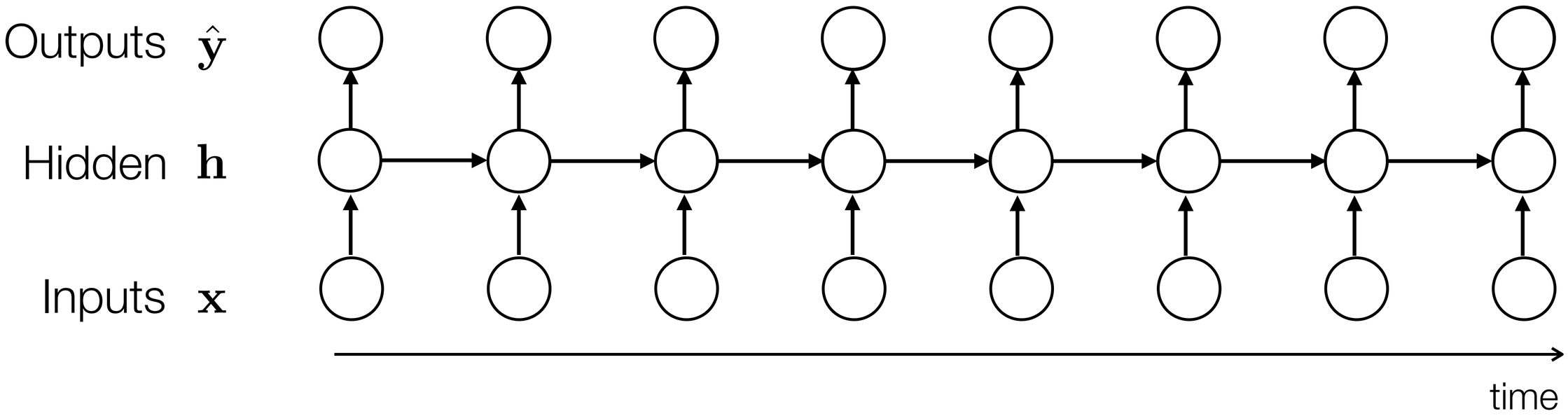
time


Memory unit
Rufus
Rufus!
W
W
time

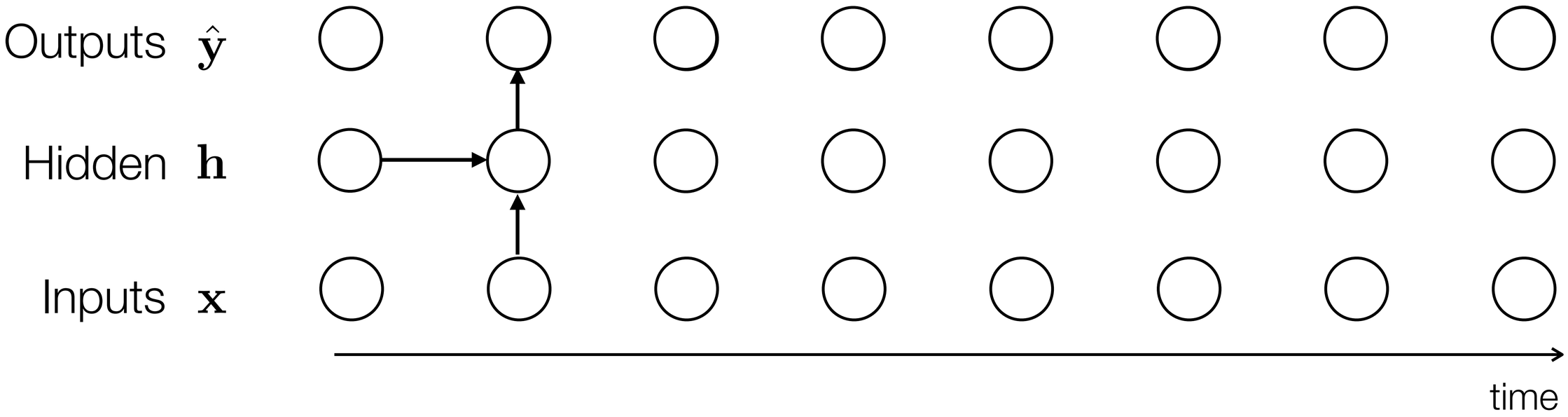
Recurrent Neural Networks (RNNs)



Recurrent Neural Networks (RNNs)



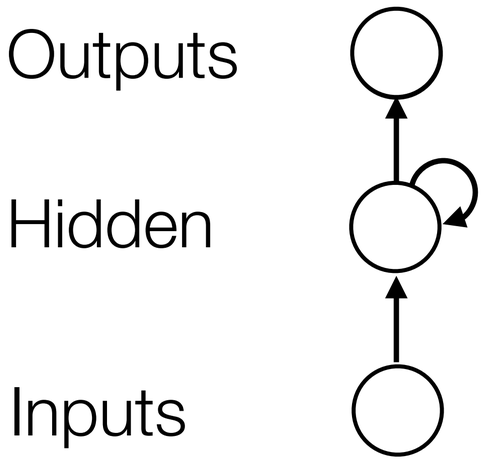
Recurrent Neural Networks (RNNs)



$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

$$\mathbf{y}_t = g(\mathbf{h}_t)$$

Recurrent Neural Networks (RNNs)

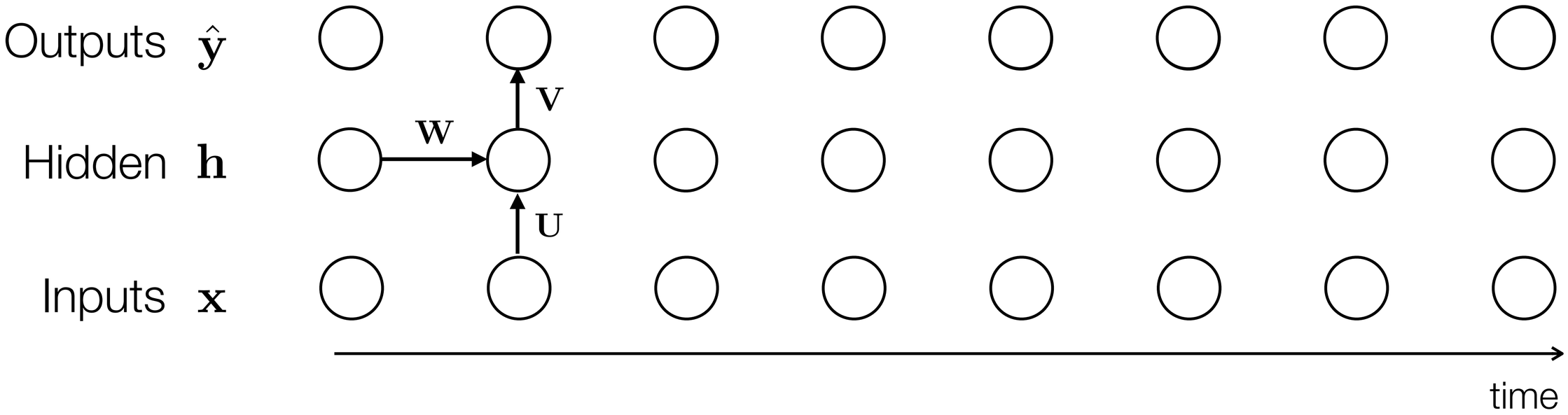


Recurrent!

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

$$\mathbf{y}_t = g(\mathbf{h}_t)$$

Recurrent Neural Networks (RNNs)



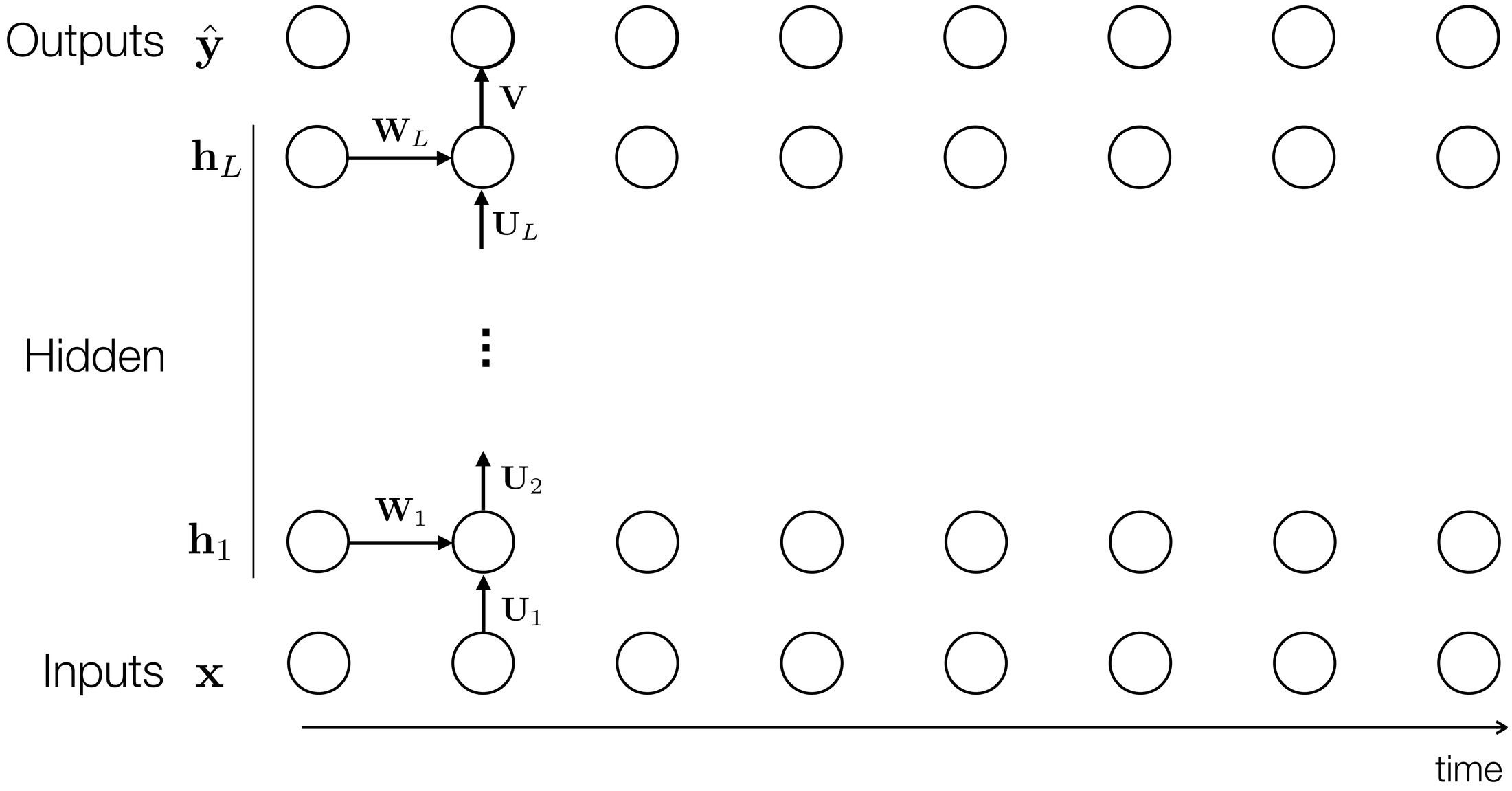
$$\mathbf{a}_t = \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}$$

$$\mathbf{h}_t = \tanh(\mathbf{a}_t)$$

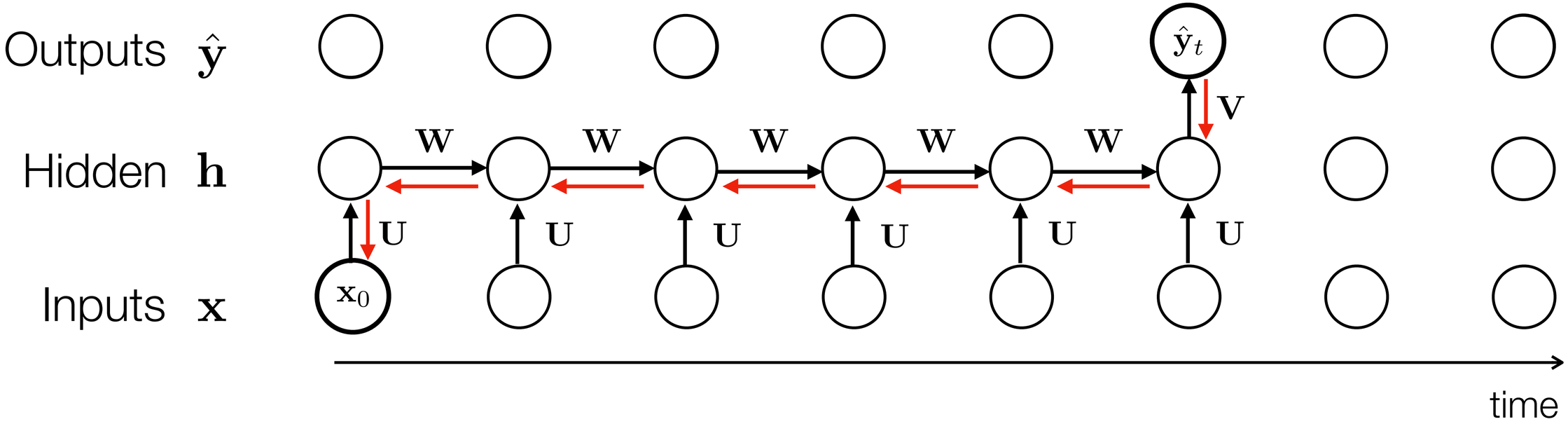
$$\mathbf{o}_t = \mathbf{V}\mathbf{h}_t + \mathbf{c}$$

$$\hat{\mathbf{y}}_t = \texttt{softmax}(\mathbf{o}_t)$$

Deep Recurrent Neural Networks (RNNs)

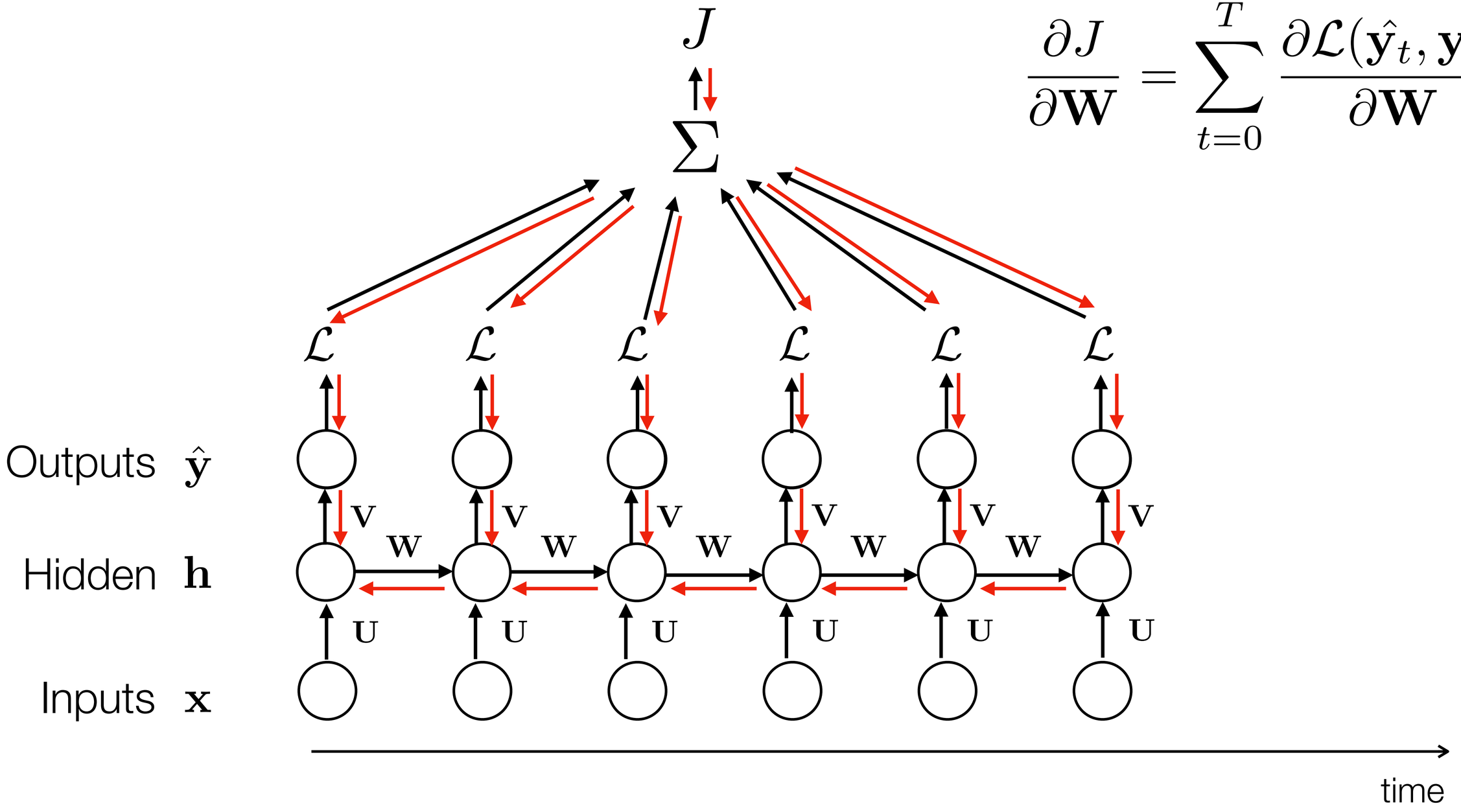


Backprop through time

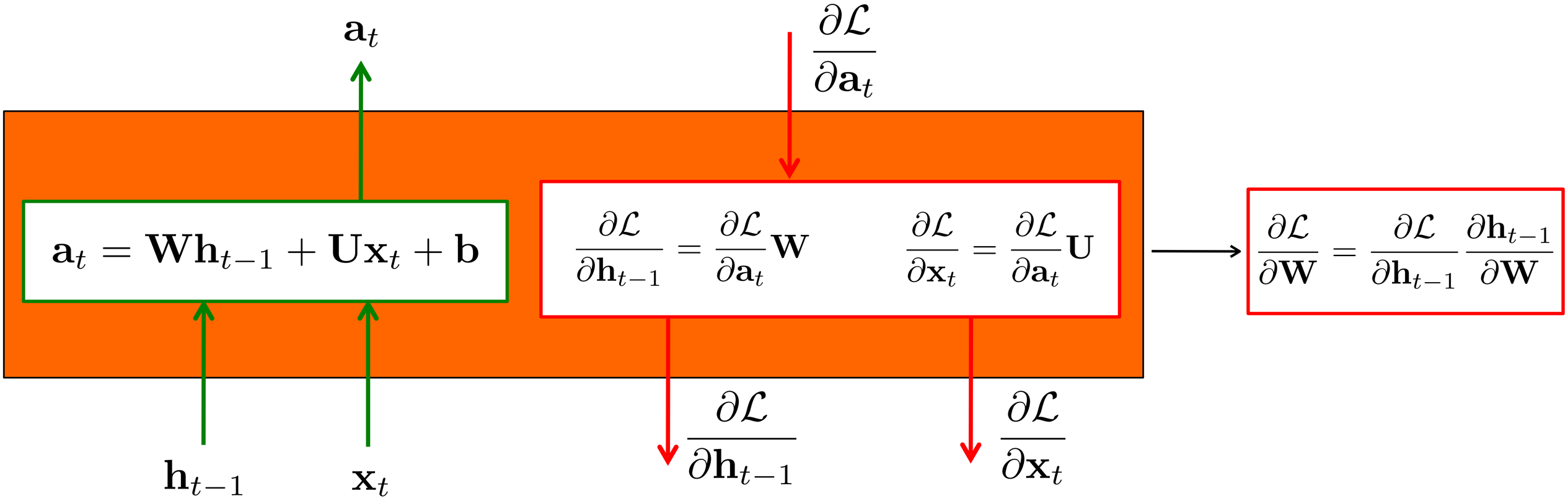


$$\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{x}_0} = \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_1}{\partial \mathbf{h}_0} \frac{\partial \mathbf{h}_0}{\partial \mathbf{x}_0}$$

$$\frac{\partial J}{\partial \mathbf{W}} = \sum_{t=0}^{T} \frac{\partial \mathcal{L}(\hat{\mathbf{y}}_t, \mathbf{y}_t)}{\partial \mathbf{W}}$$



Recurrent linear layer



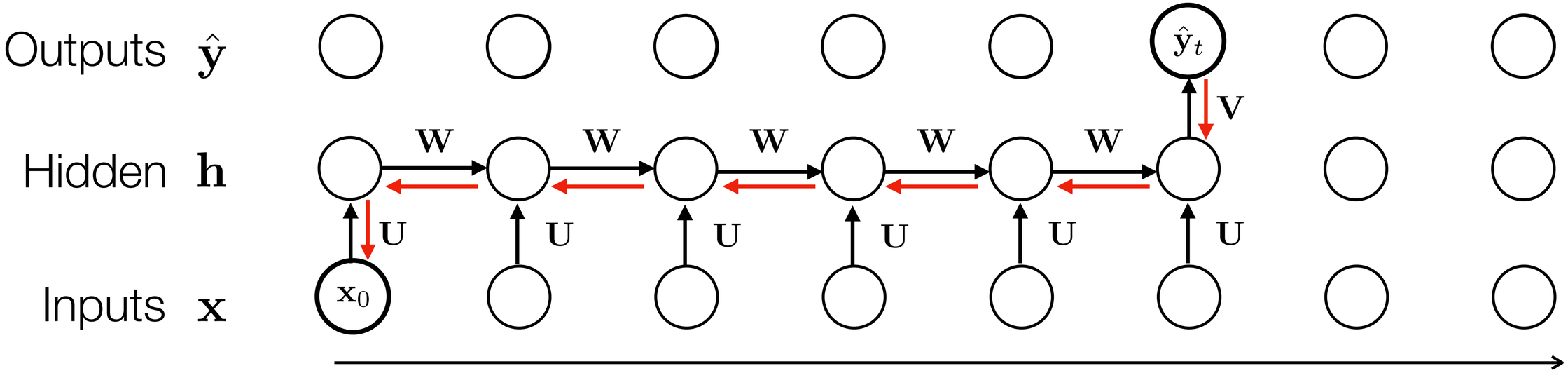
$$\frac{\partial J}{\partial \mathbf{W}} = \sum_{t=0}^{T} \frac{\partial \mathcal{L}(\hat{\mathbf{y}}_t, \mathbf{y}_t)}{\partial \mathbf{W}}$$

The problem of long-range dependences

Why not remember everything?

- Memory size grows with t
- This kind of memory is **nonparametric**: there is no finite set of parameters we can use to model it
- RNNs make a Markov assumption — the future hidden state only depends on the immediately preceding hidden state
- By putting the right info in to the hidden state, RNNs can model depedences that are arbitrarily far apart

The problem of long-range dependences



$$\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{x}_0} = \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_1}{\partial \mathbf{h}_0} \frac{\partial \mathbf{h}_0}{\partial \mathbf{x}_0}$$

- Capturing long-range dependences requires propagating information through a long chain of dependences.
- Old observations are forgotten
- Stochastic gradients become high variance (noisy), and gradients may **vanish** or **explode**

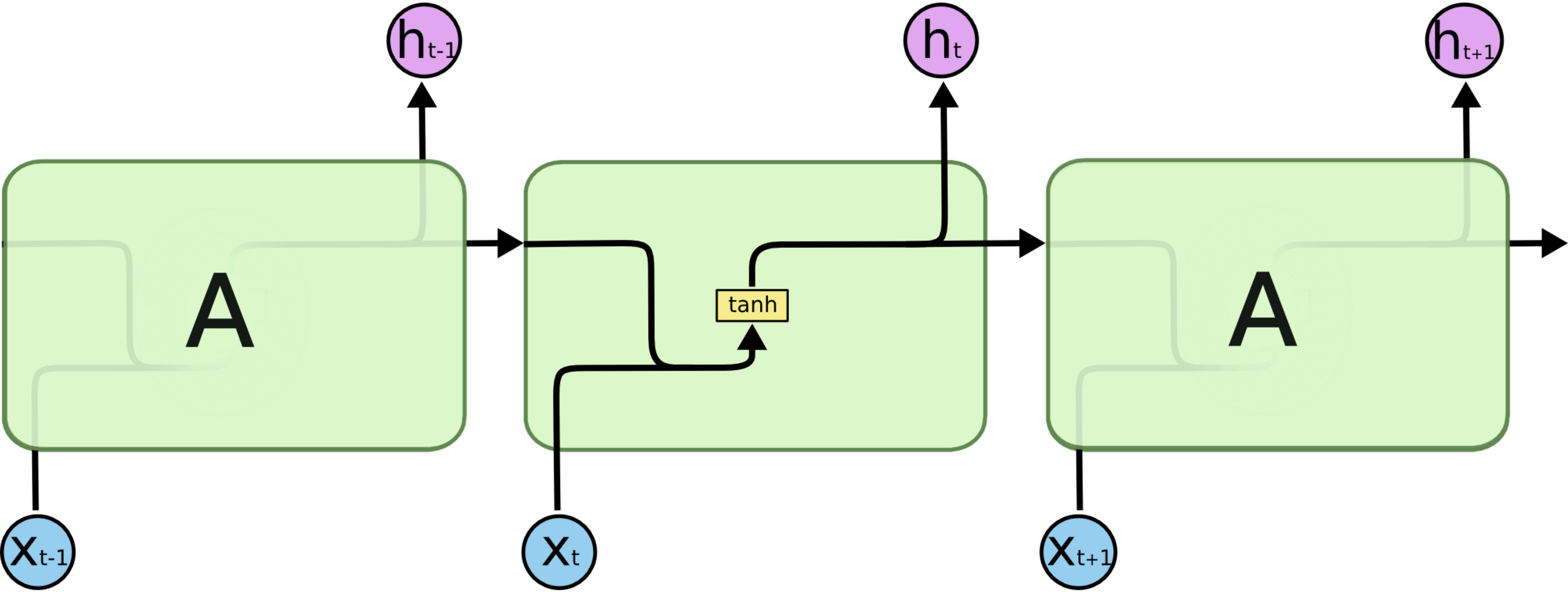
LSTMs

Long Short Term Memory

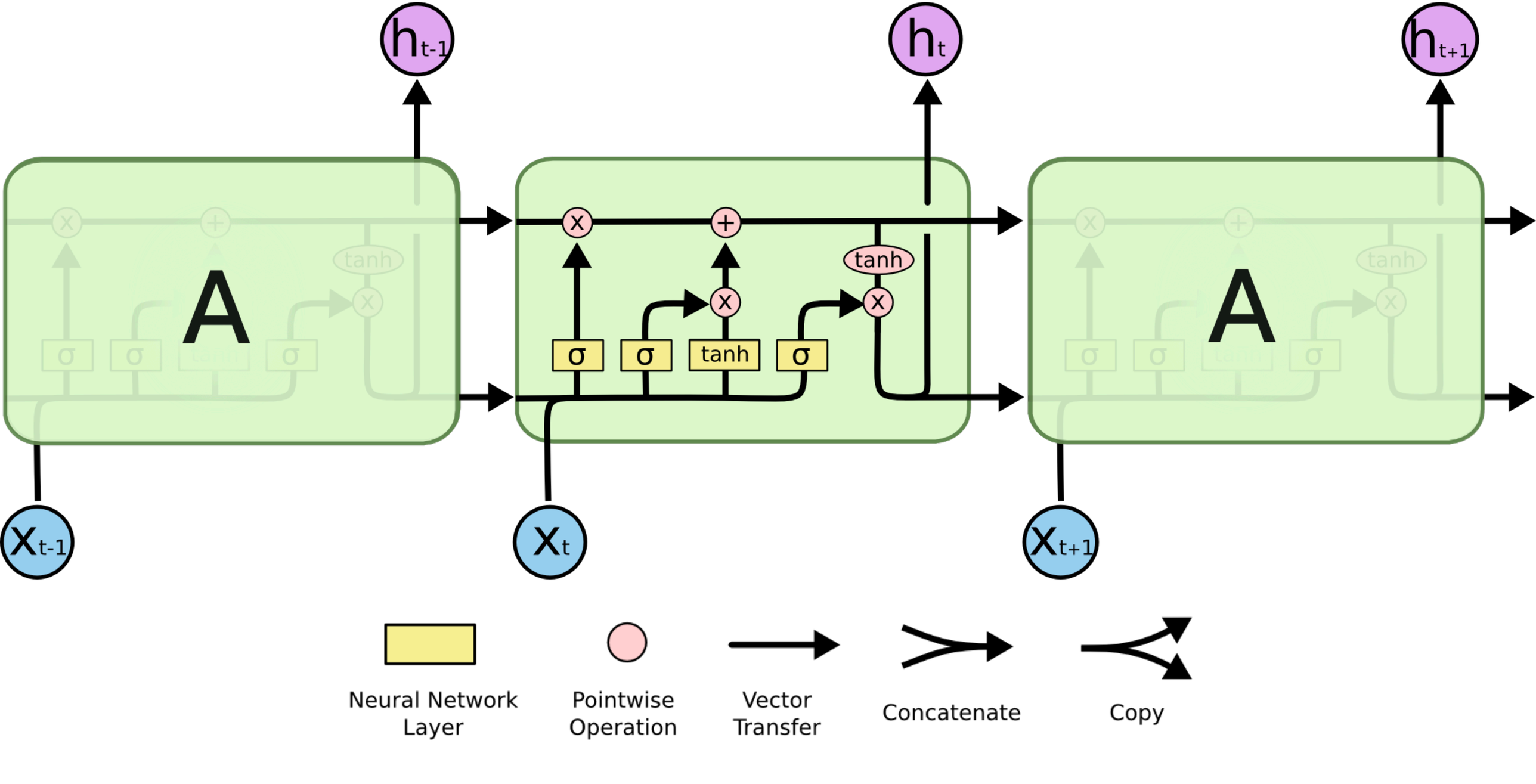
[Hochreiter & Schmidhuber, 1997]

A special kind of RNN designed to avoid forgetting.

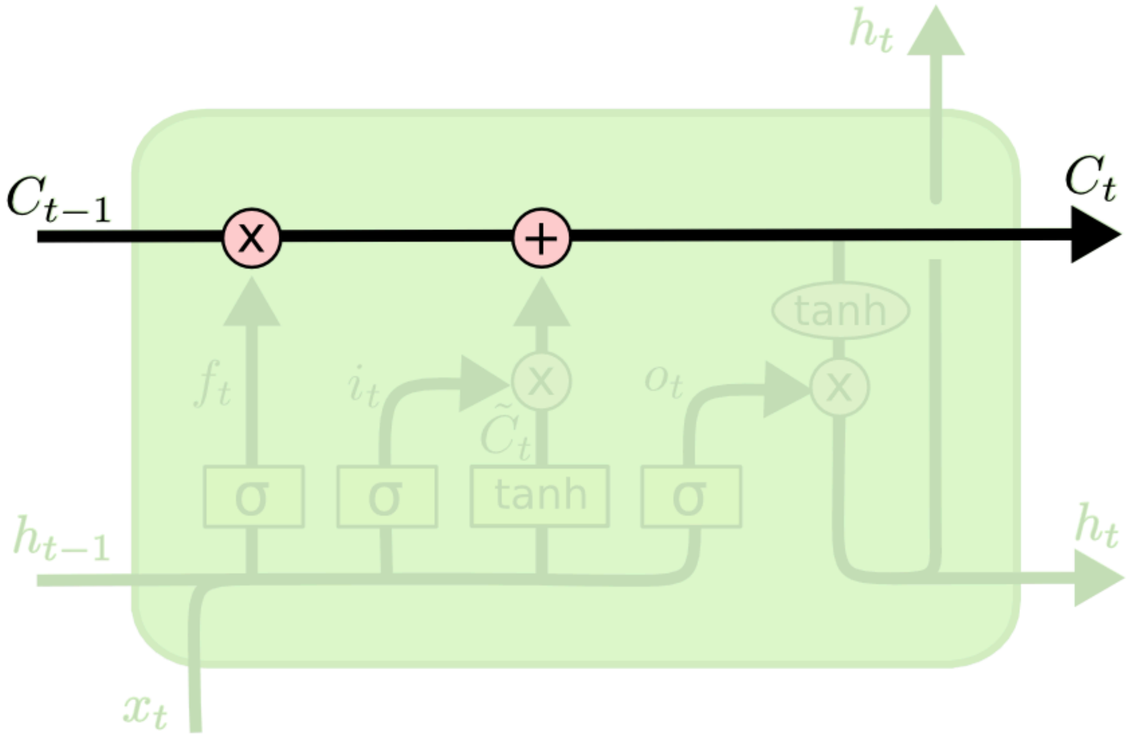
This way the default behavior is not to forget an old state. Instead of forgetting by default, the network has to *learn to forget*.



[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]

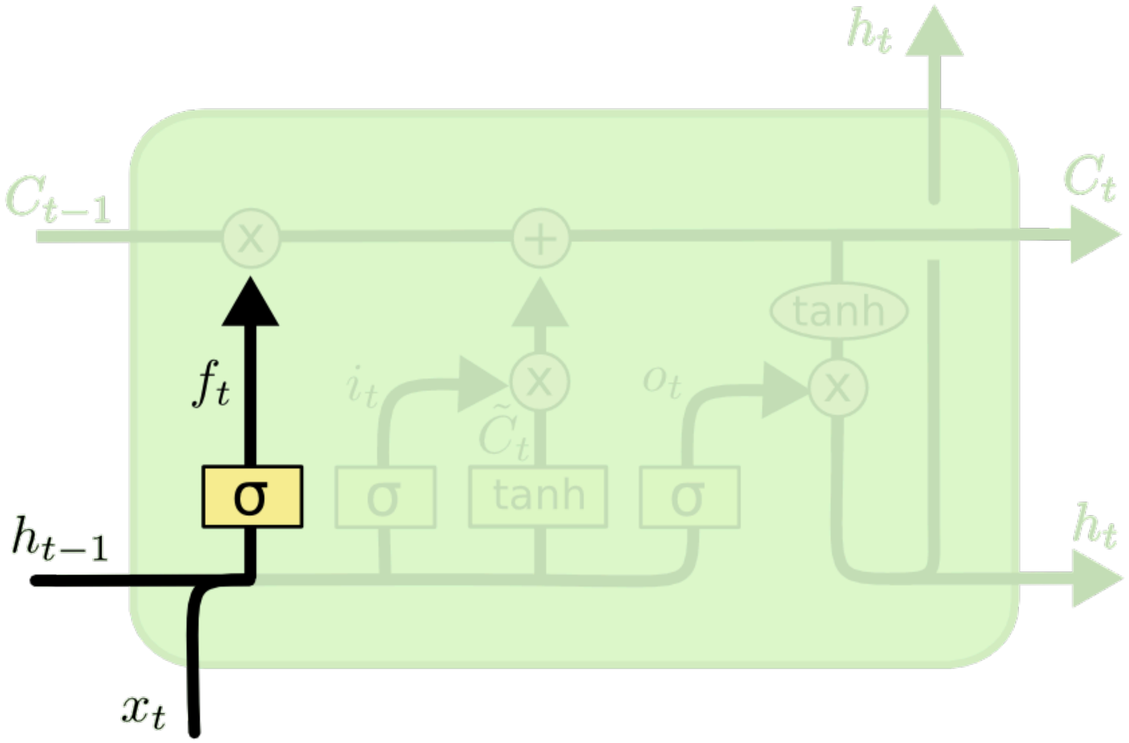


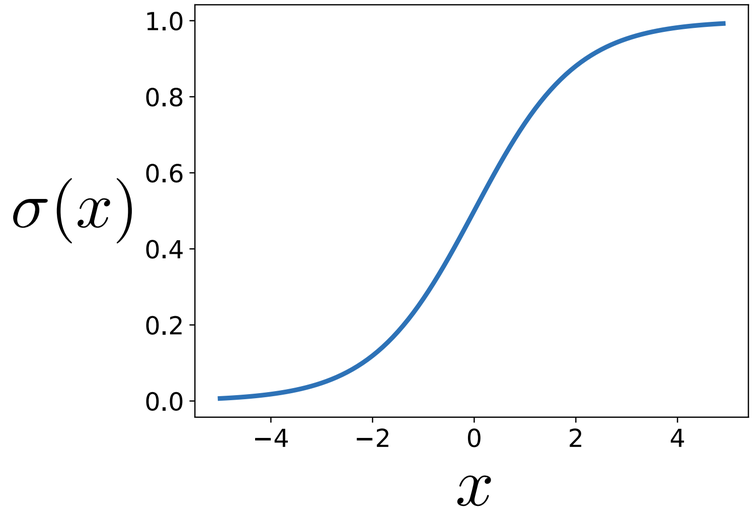
[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]



C_t = Cell state

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]



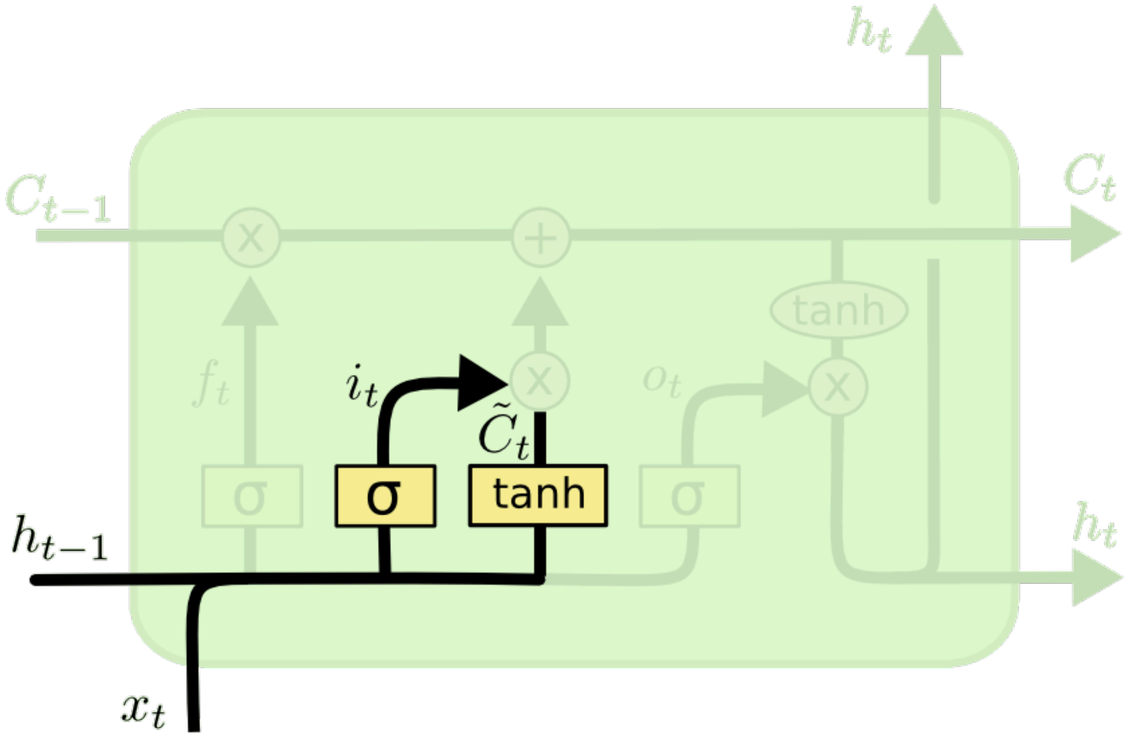


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Decide what information to throw away from the cell state.

Each element of cell state is multiplied by ~1 (remember) or ~0 (forget).

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]



which indices to write to

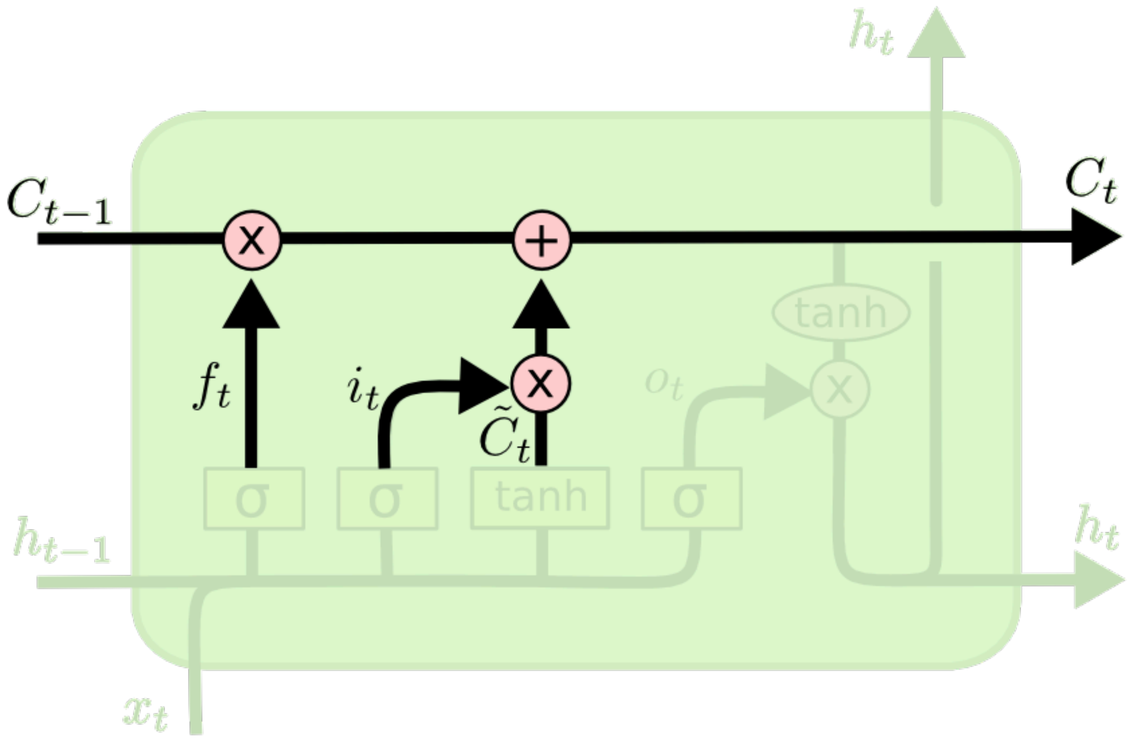
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

what to write to those indices

Decide what new information to add to the cell state.

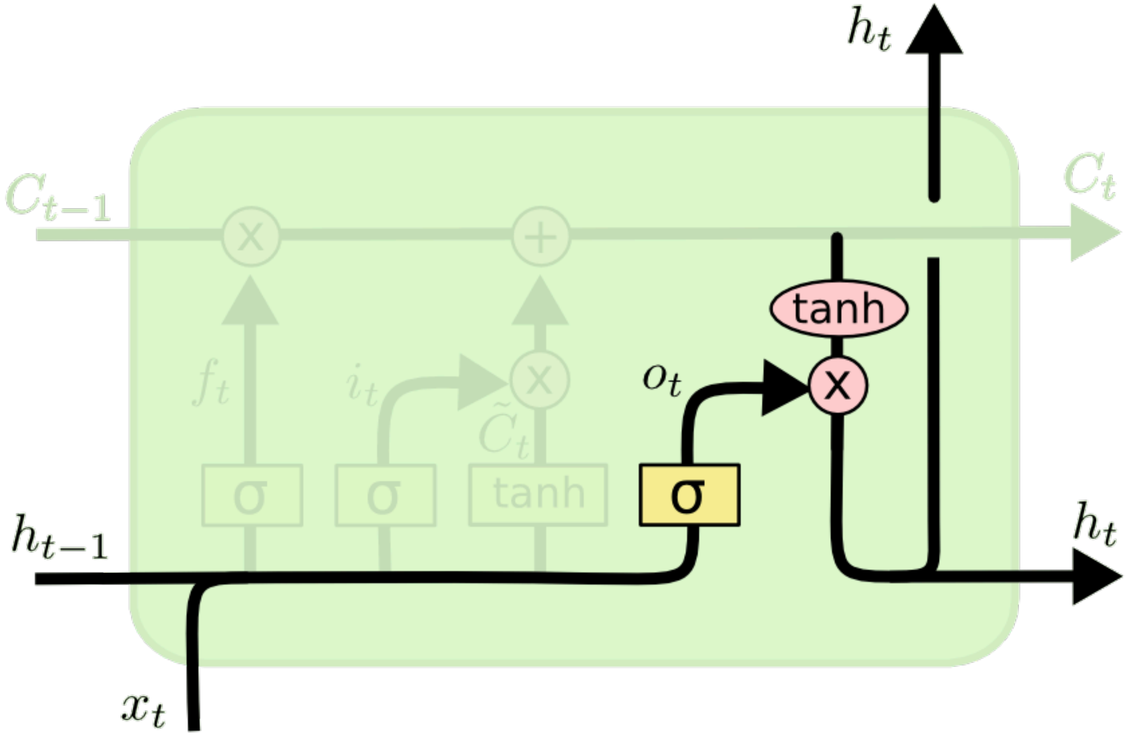
[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Forget selected old information, write selected new information.

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]



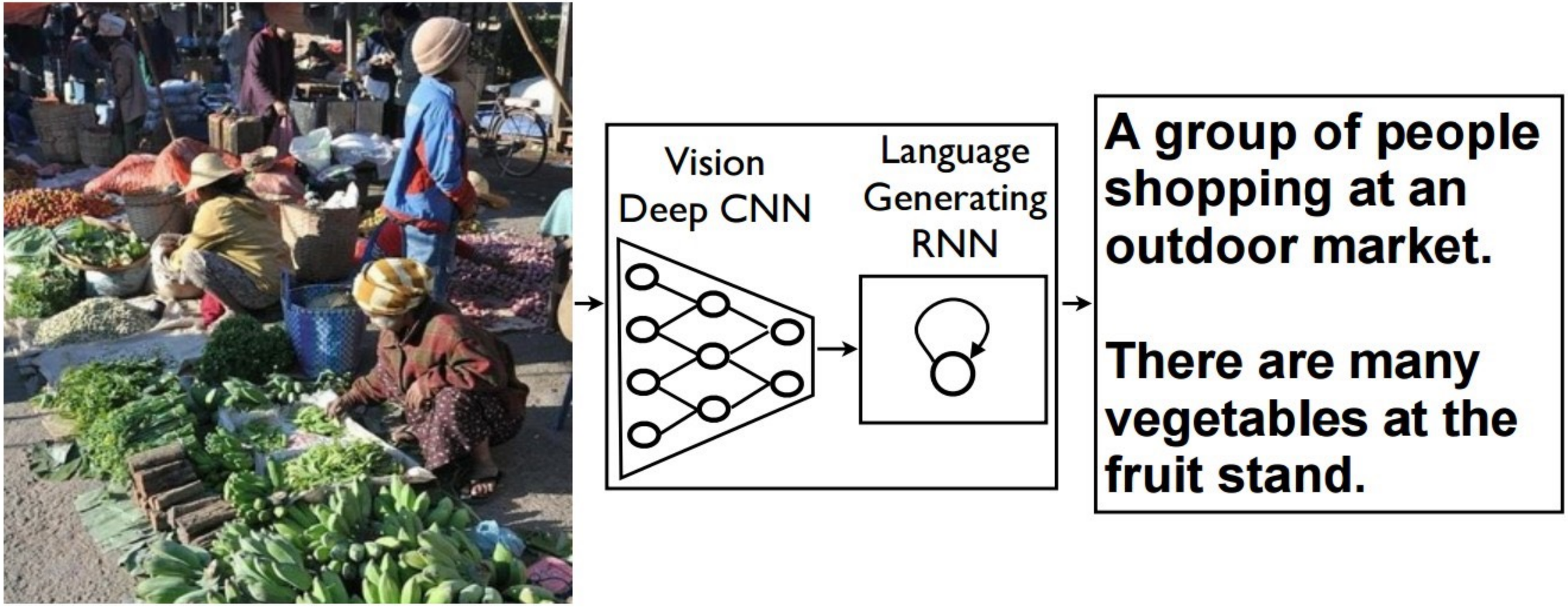
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

After having updated the cell state's information, decide what to output.

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]

Image Captioning



[Example above from: Vinyals, Toshev, Bengio, Erhan, CVPR 2015, https://arxiv.org/abs/1411.4555]

Recipe for deep learning in a new domain

1. Transform your data into numbers (e.g., a vector)

2. Transform your goal into a numerical measure (objective function)

3. #1 and #2 specify the “learning problem”

4. Use a generic optimizer (SGD) and an appropriate architecture (e.g., CNN or RNN) to solve the learning problem

How to represent words as numbers?

One-hot vector

Training data

$\mathbf{x}$	y
	"Fish"
	"Grizzly"
	"Chameleon"
⋮	

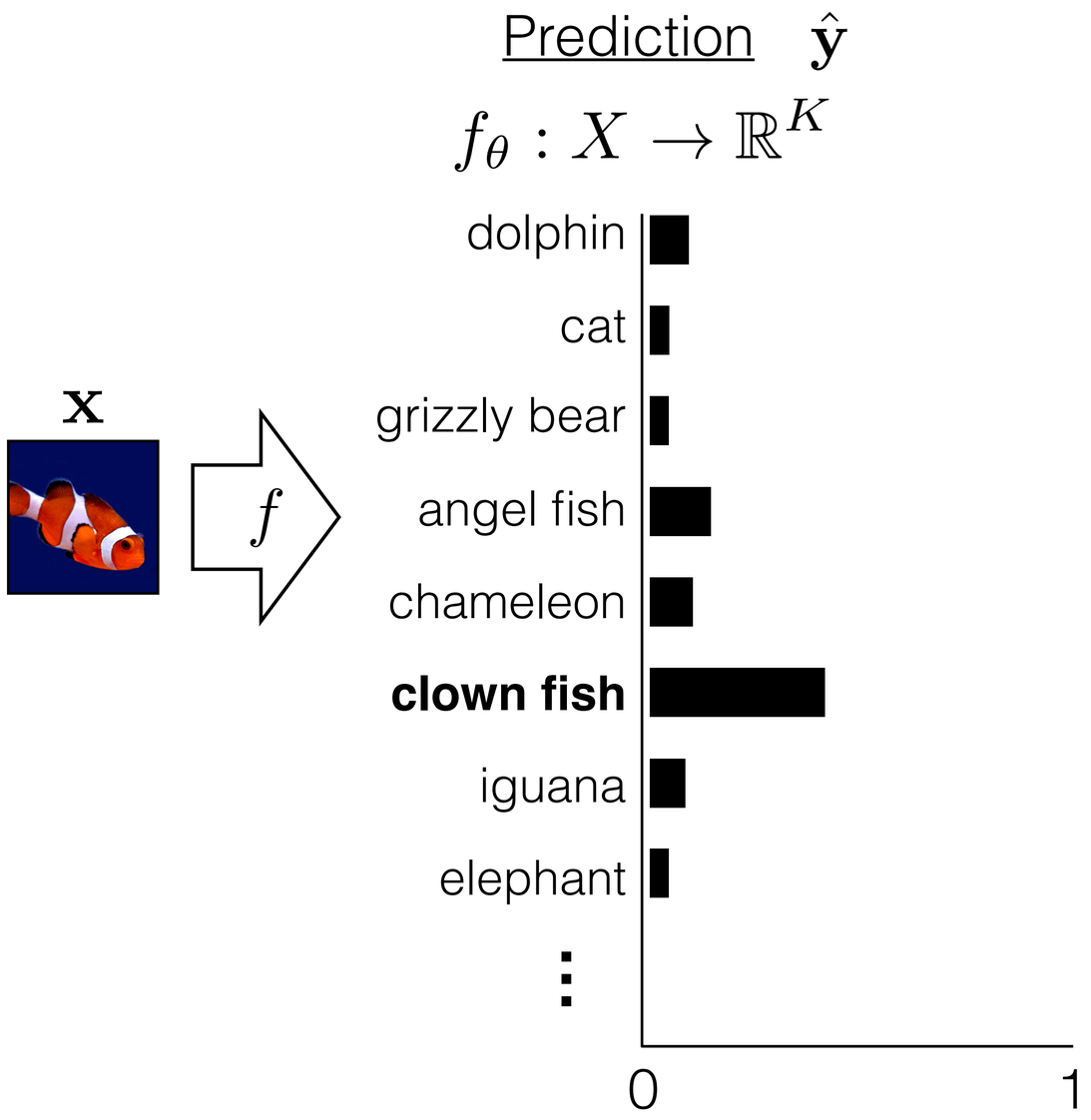
Training data

$\mathbf{x}$	y
	1
	2
	3
⋮	

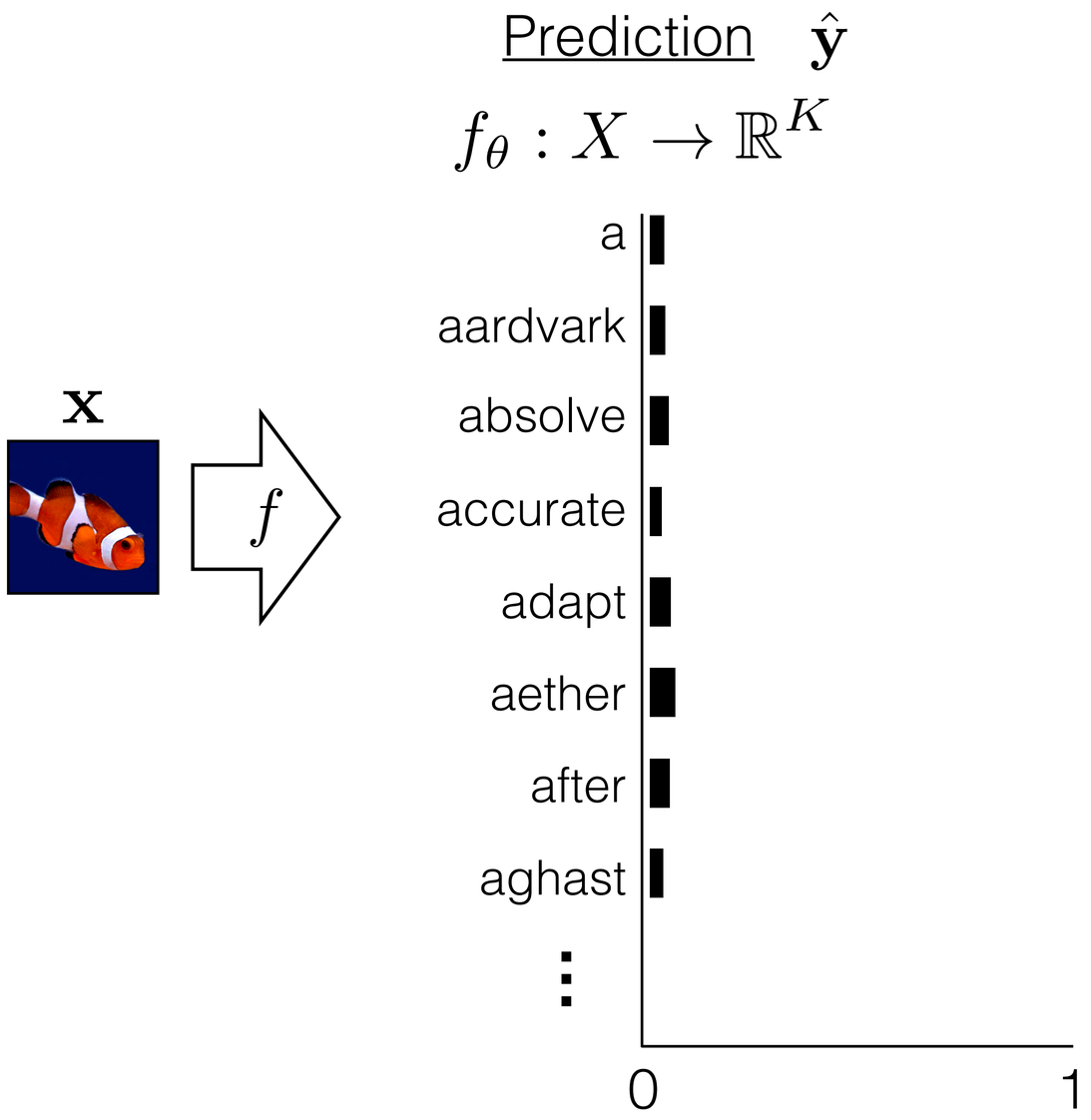
Training data

$\mathbf{x}$	$\mathbf{y}$
	[0,0,1]
	[0,1,0]
	[1,0,0]
⋮	

How to represent words as numbers?

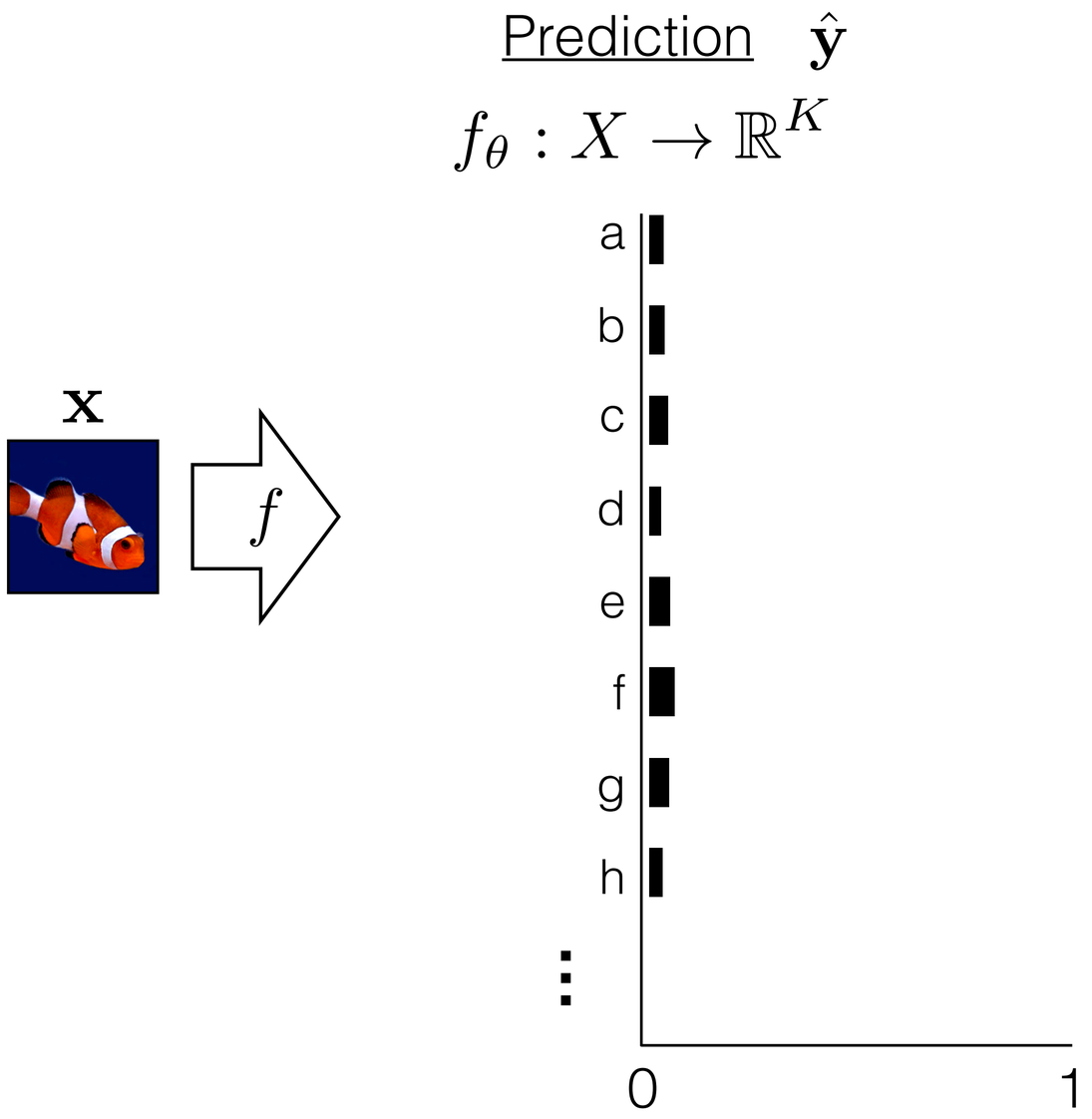


How to represent words as numbers?



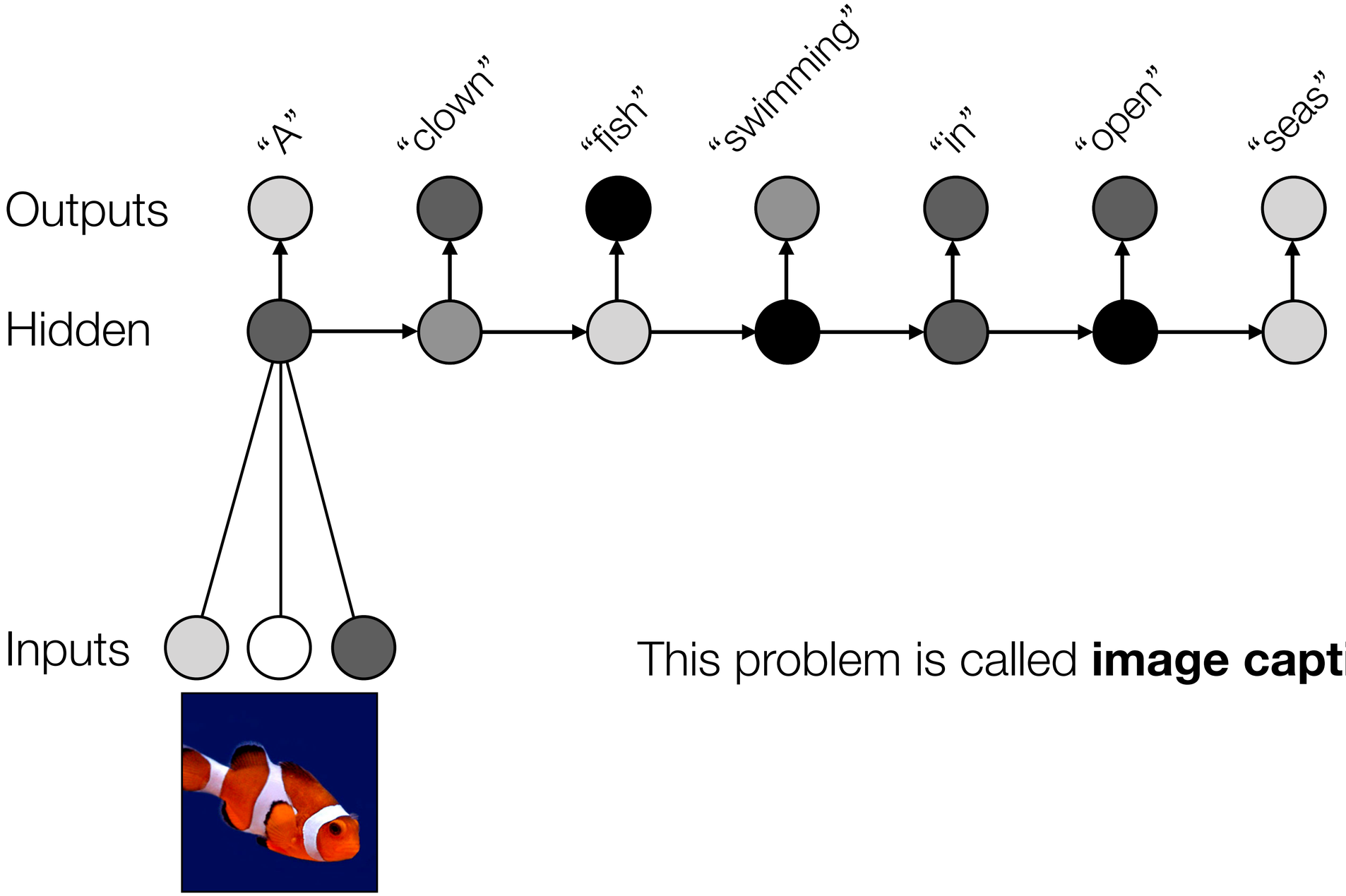
Rather than having just a handful of possible object classes, we can represent all words in a large vocabulary using a very large K (e.g., K=100,000).

How to represent words as numbers?

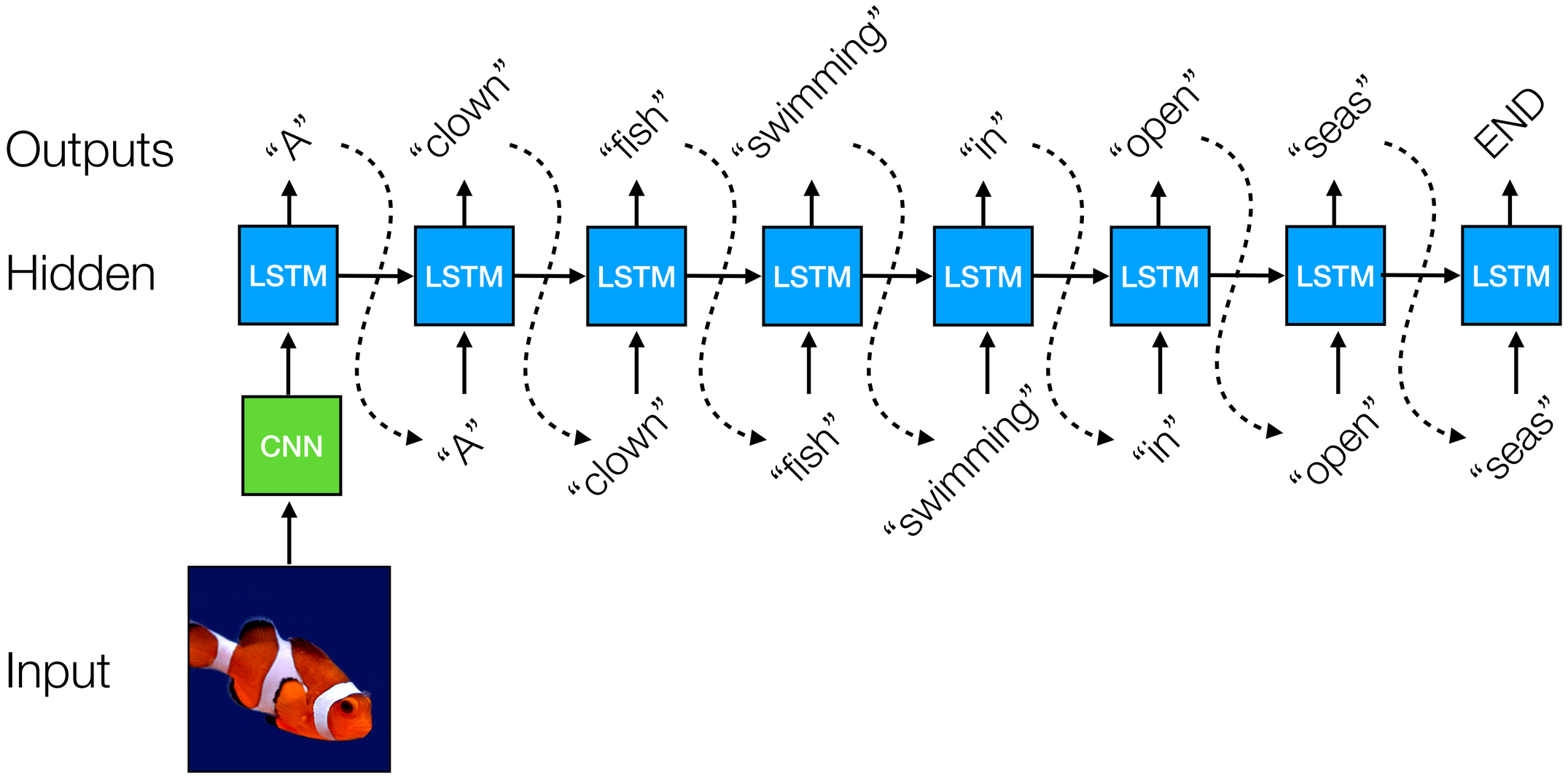


Or, represent each character as a class (e.g., K=26 for English letters),

and represent words as a sequence of characters.



This problem is called **image captioning**


Outputs
Hidden
Input
CNN
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
“A”
“clown”
“fish”
“swimming”
“in”
“open”
“seas”
END
“A”
“clown”
“fish”
“swimming”
“in”
“open”
“seas”

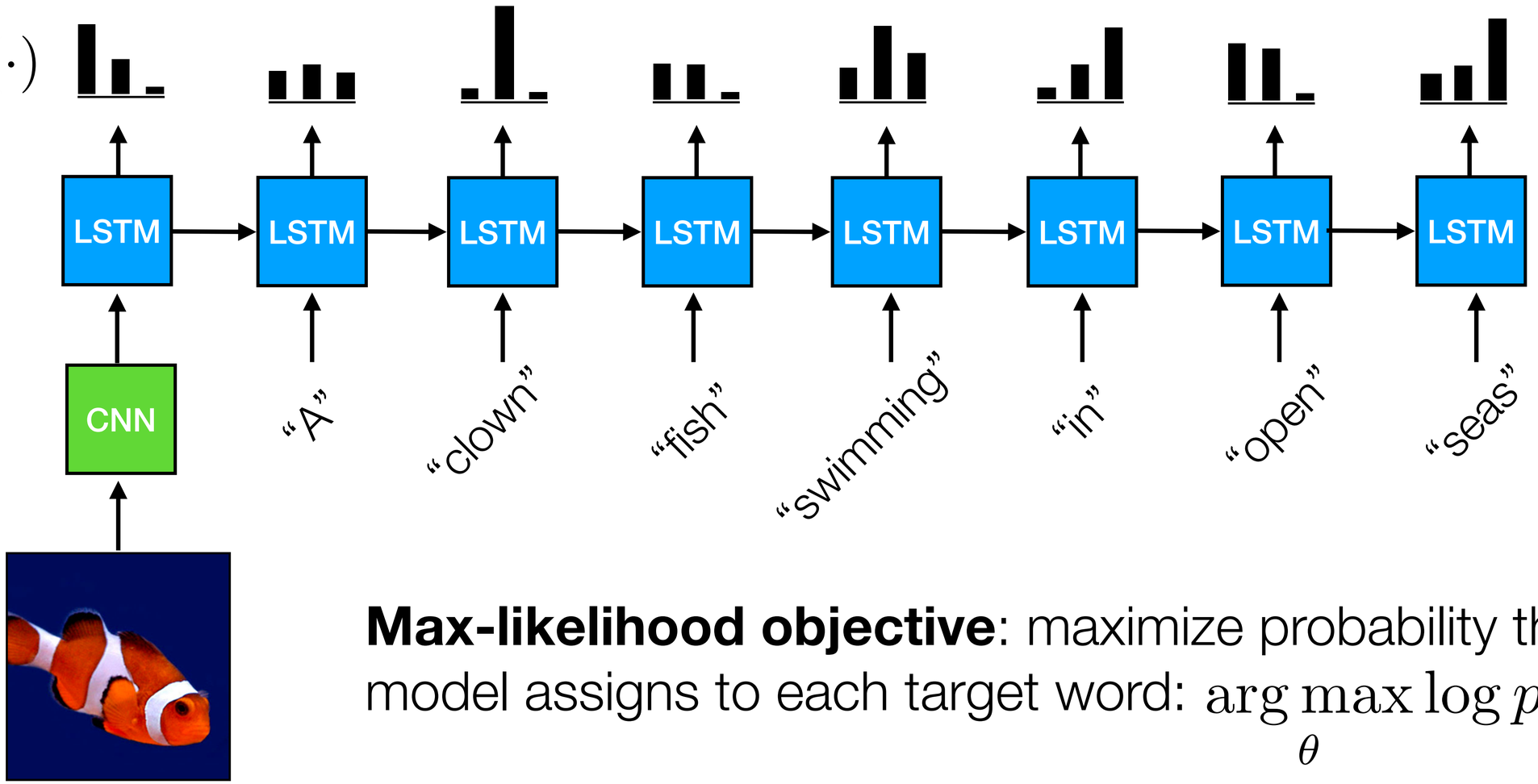
Training

Targets y

Outputs $p_\theta(\cdot)$

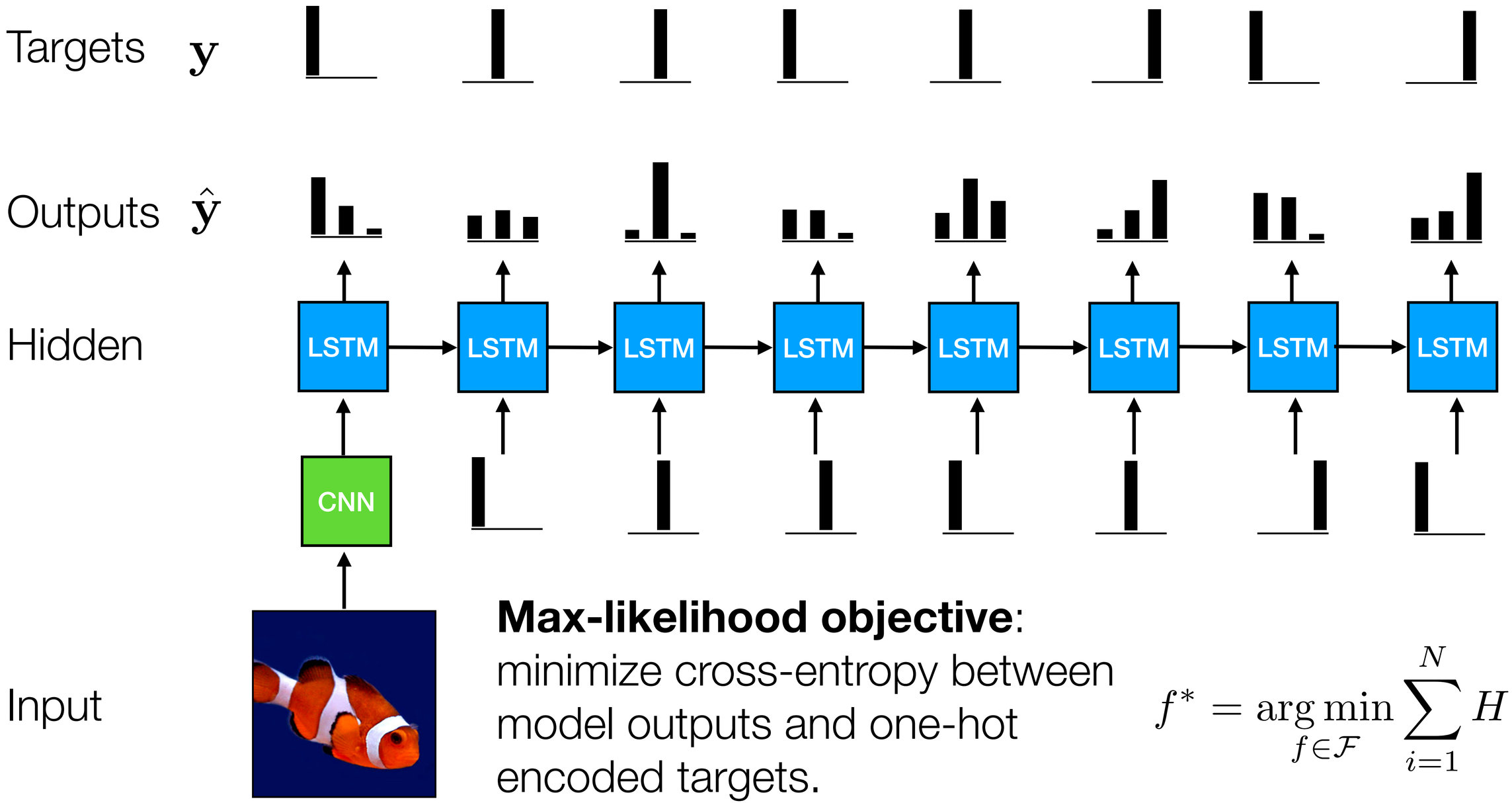
Hidden

Input



Max-likelihood objective: maximize probability the model assigns to each target word: $\arg\max_\theta \log p_\theta(y)$

Training



Max-likelihood objective:
minimize cross-entropy between model outputs and one-hot encoded targets.

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$

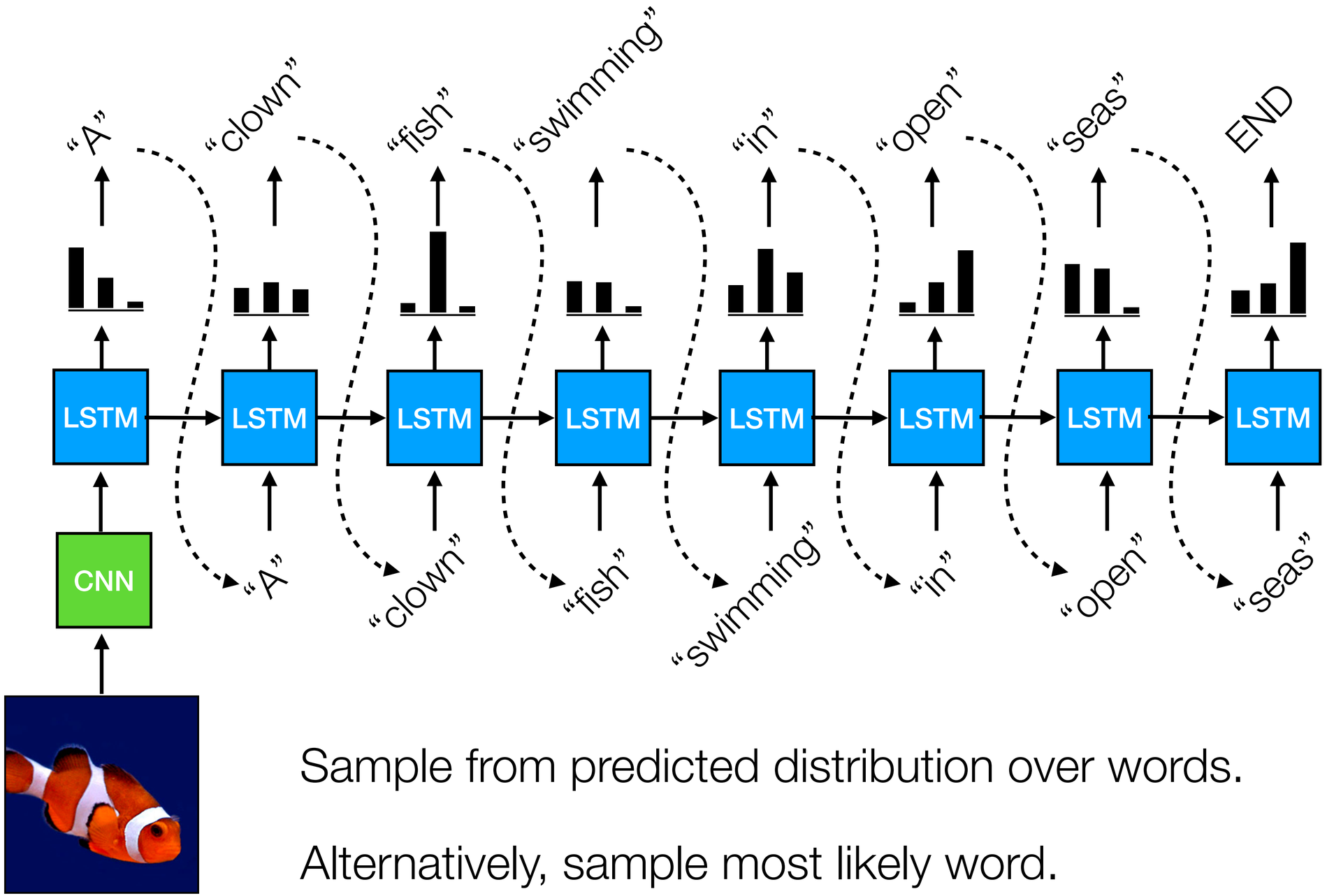
Testing

Samples

Outputs $p_\theta(\cdot)$

Hidden

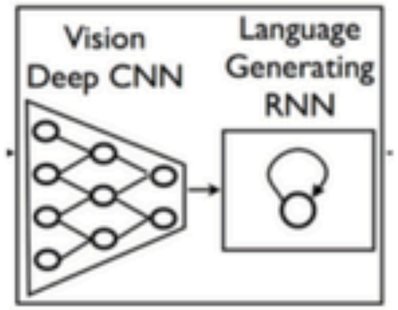
Input



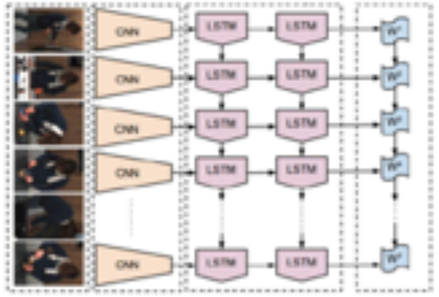
Sample from predicted distribution over words.

Alternatively, sample most likely word.

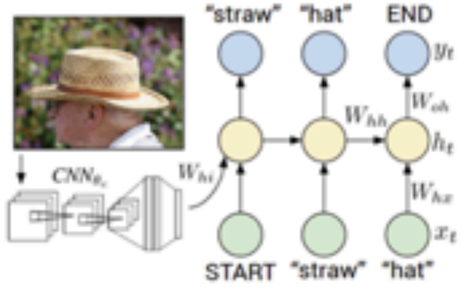
It was very popular a few years ago



Vinyals et al., 2015



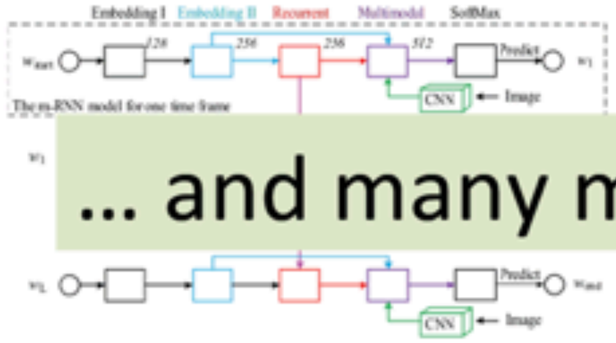
Donahue et al., 2015



Karpathy and Fei-Fei, 2015

Hodosh et al., 2013

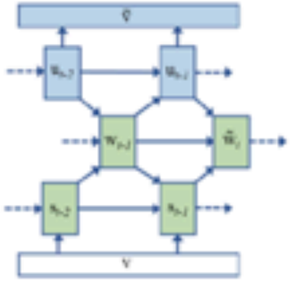
Fang et al., 2015



Mao et al., 2015

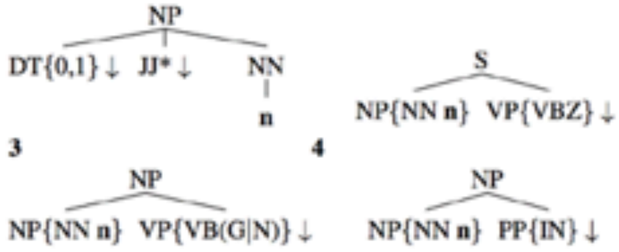
Ordonez et al., 2011

Kulkarni et al., 2011

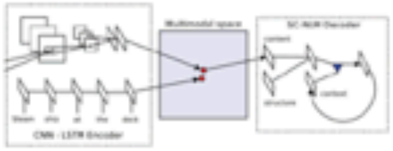


Chen and Zitnick, 2015

Farhadi et al., 2010



Mitchell et al., 2012



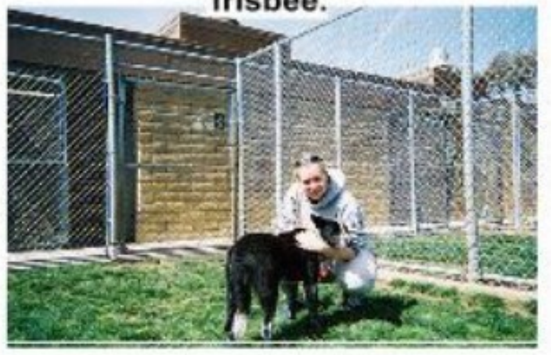
Kiros et al., 2015

A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

A skateboarder does a trick on a ramp.

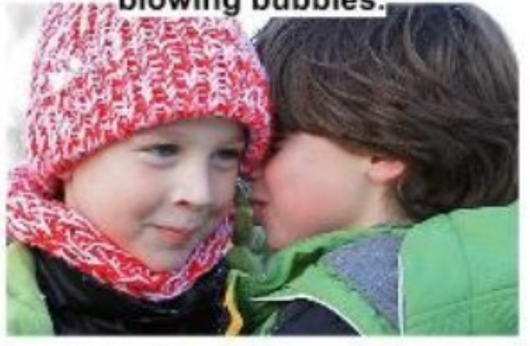
A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

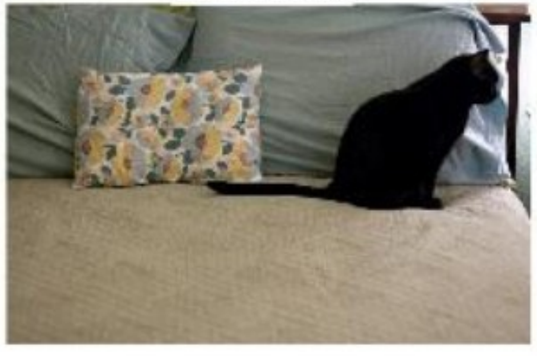
A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.

A herd of elephants walking across a dry grass field.

A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.

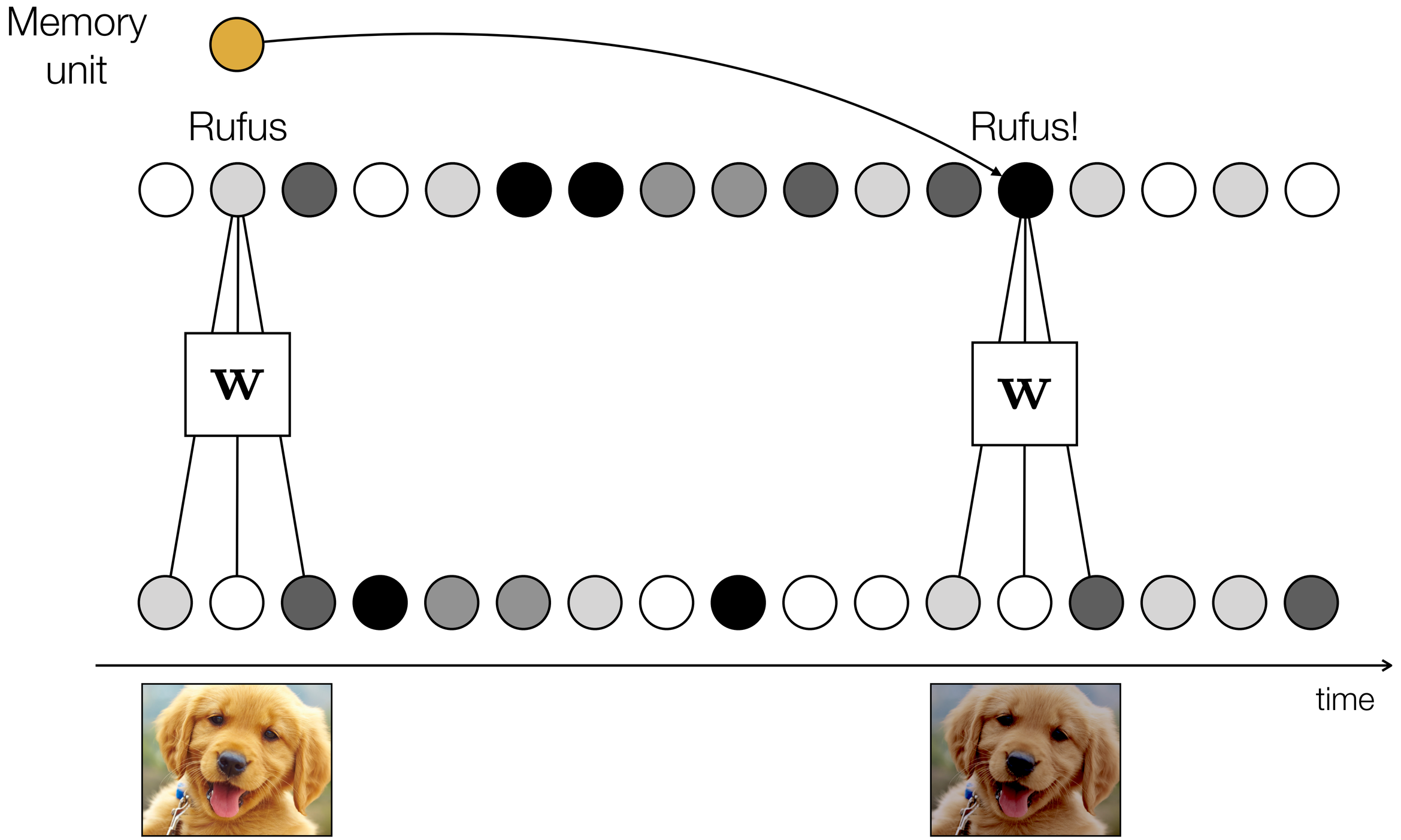
A yellow school bus parked in a parking lot.

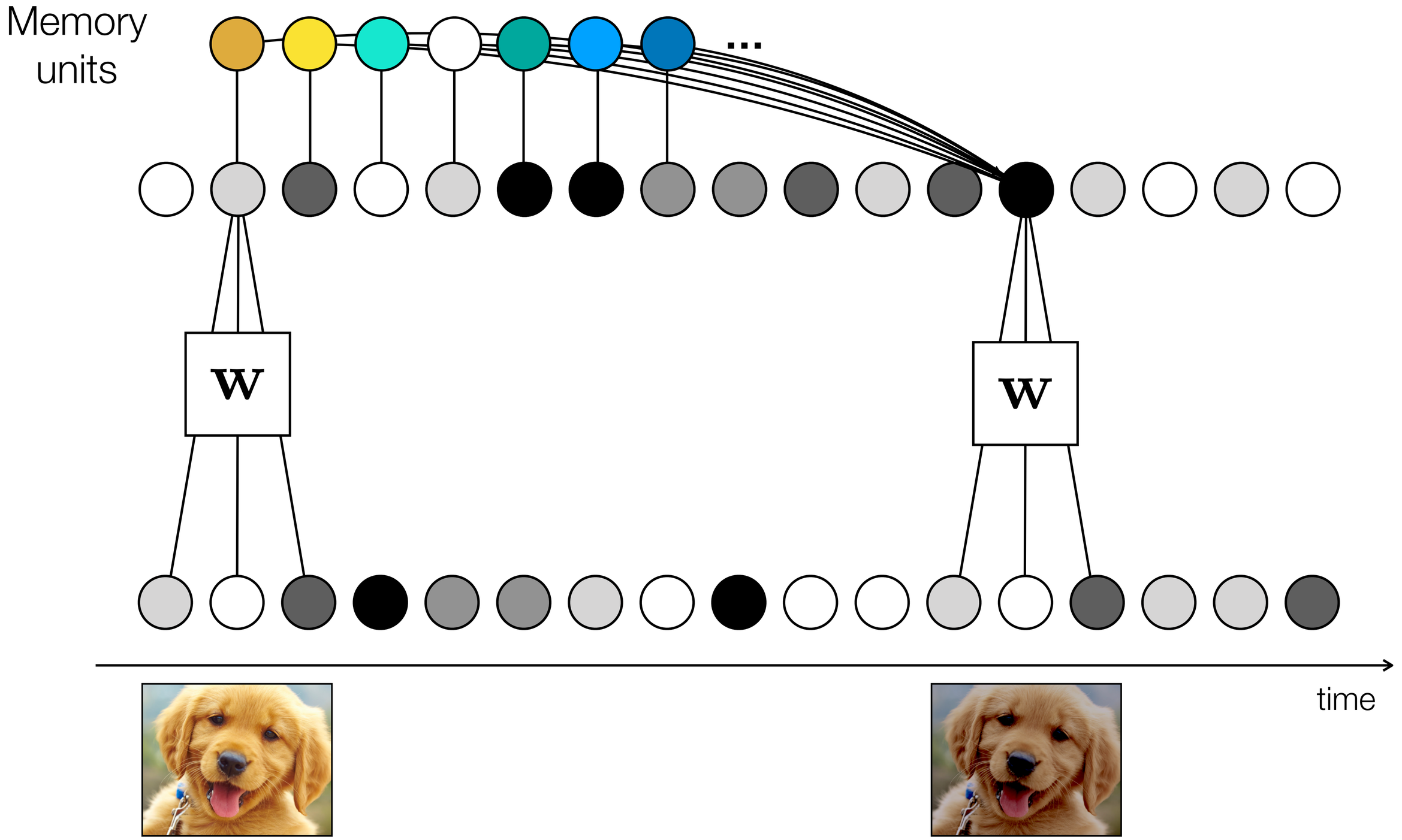
Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image

The problem of long-range dependences

Why not remember everything?

- Memory size grows with t
- This kind of memory is **nonparametric**: there is no finite set of parameters we can use to model it
- RNNs make a Markov assumption — the future hidden state only depends on the immediately preceding hidden state
- By putting the right info in to the hidden state, RNNs can model depedences that are arbitrarily far apart


Memory unit
Rufus
Rufus!
W
W
time

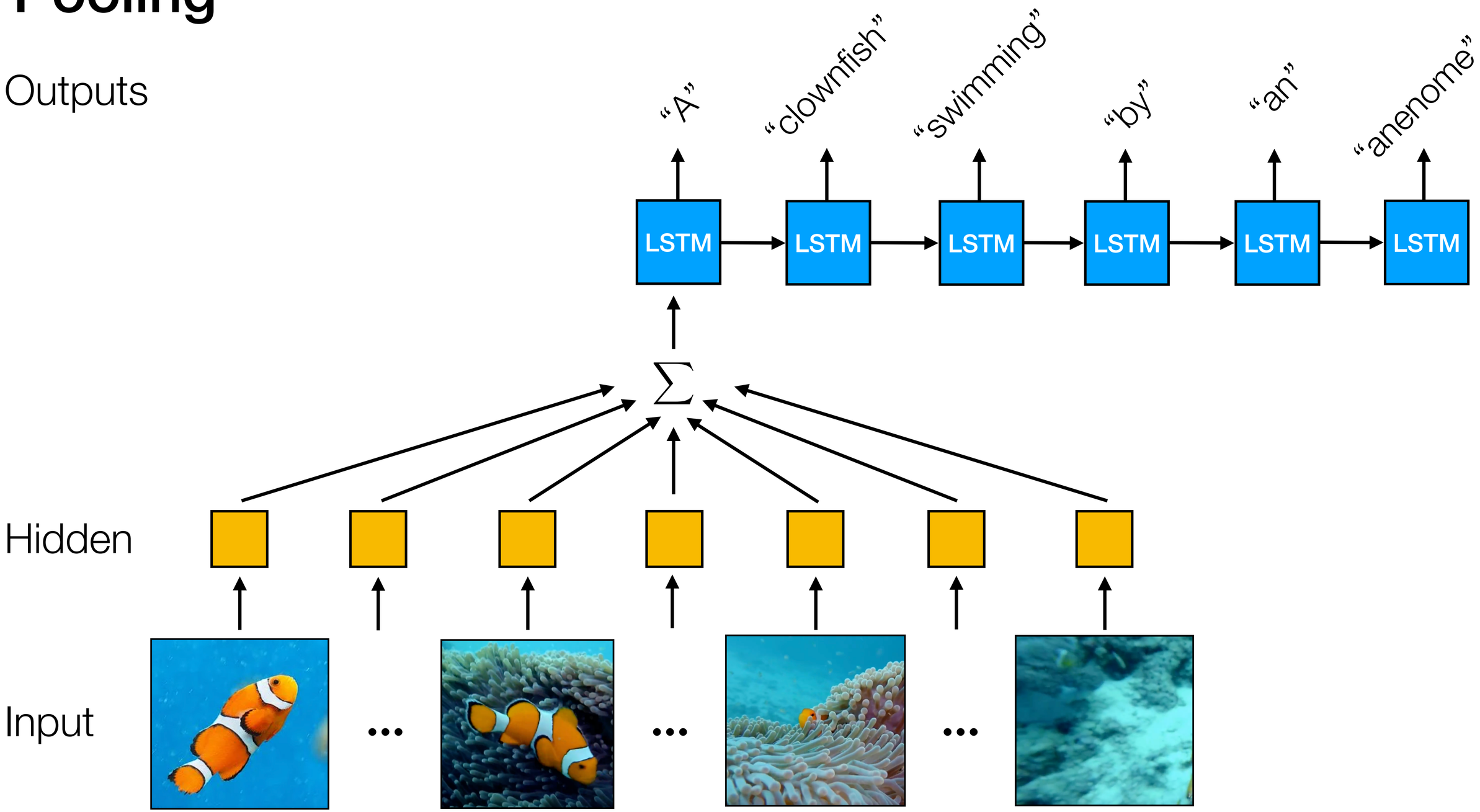

Memory units
...
W
W
time

The problem of long-range dependences

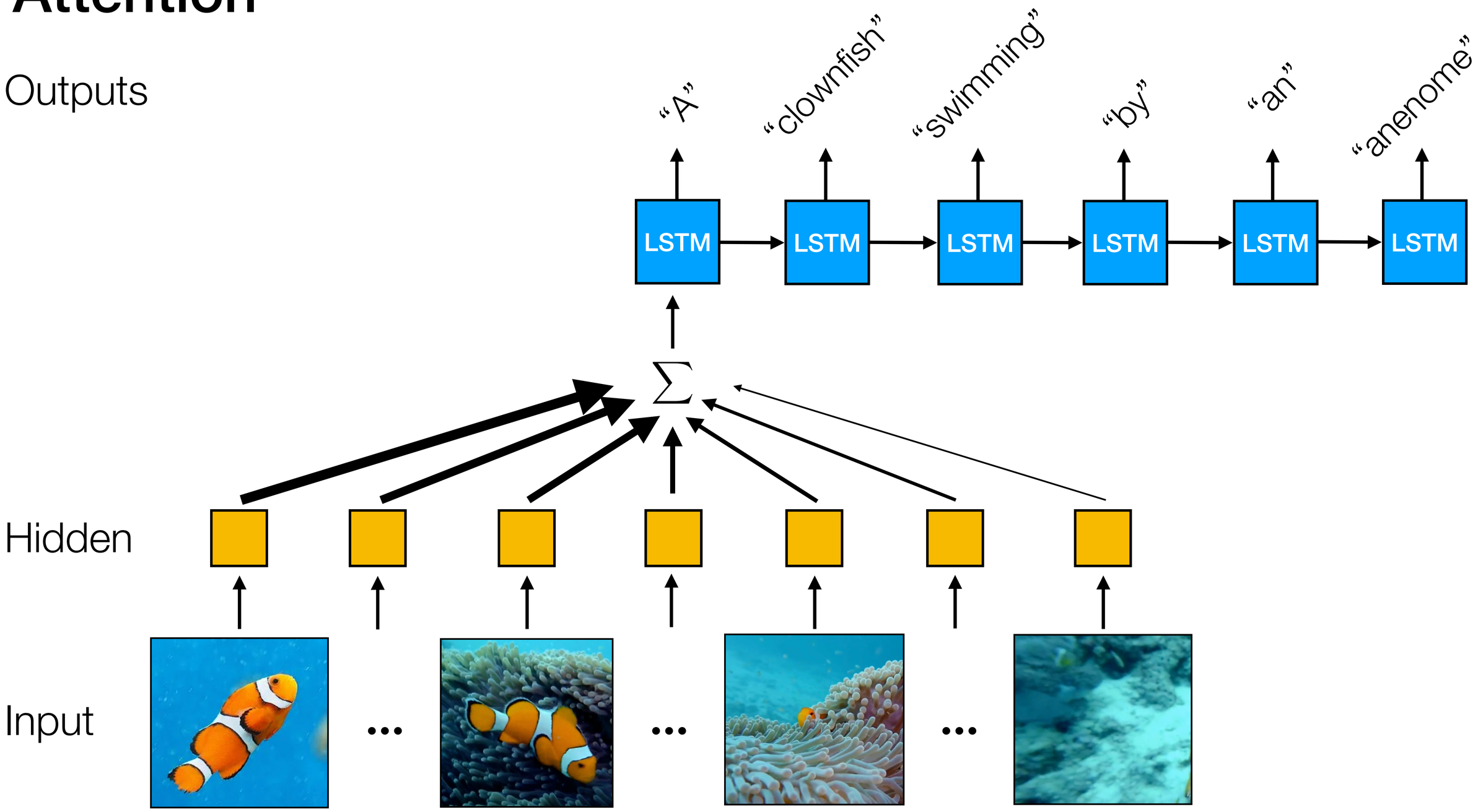
Other methods exist that do directly link old “memories” (observations or hidden states) to future predictions:

- Temporal convolutions
- Attention / Transformers (see https://arxiv.org/abs/1706.03762)
- Memory networks (see https://arxiv.org/abs/1410.3916)

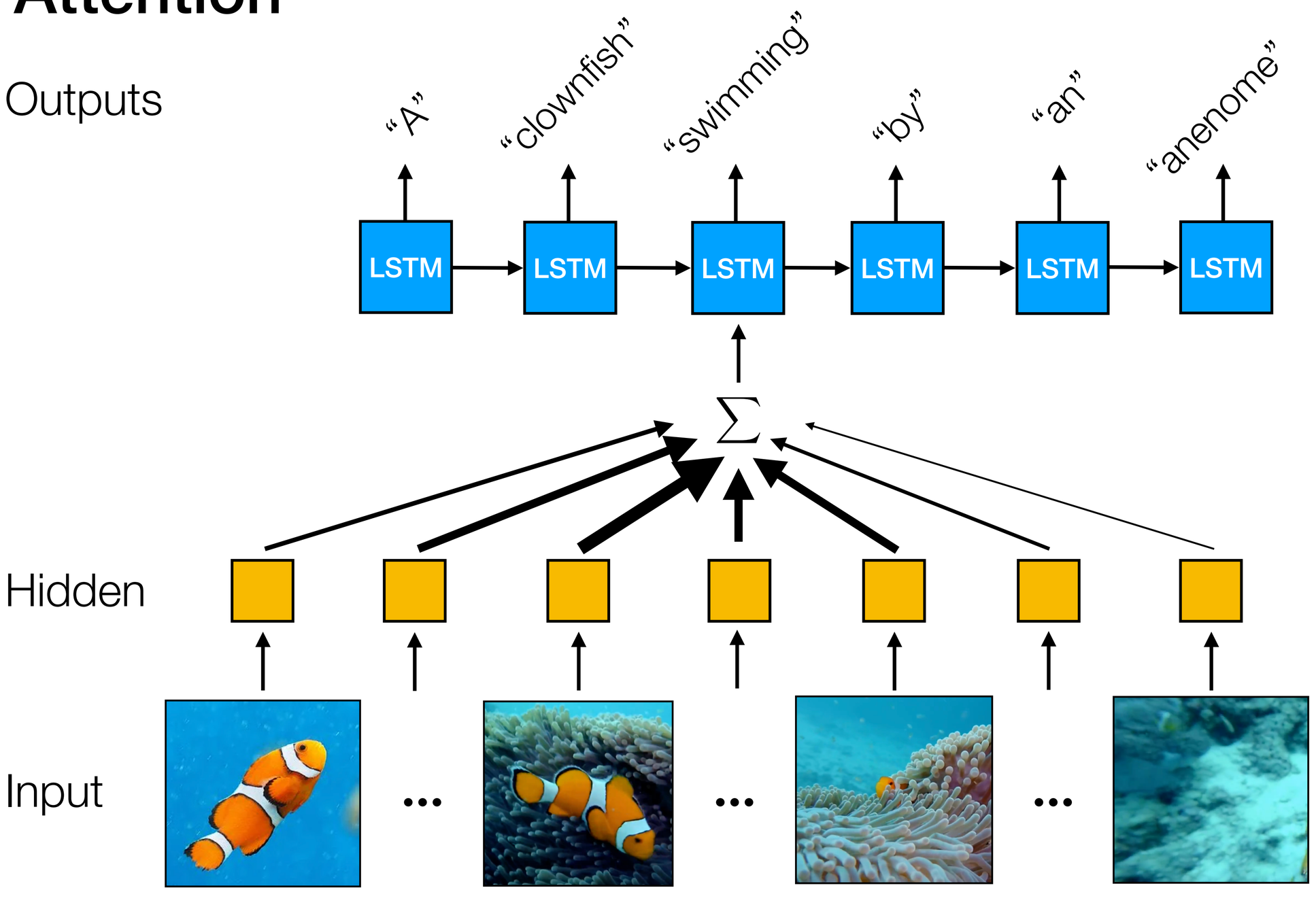
Pooling


Outputs
“A”
“clownfish”
“swimming”
“by”
“an”
“anenome”
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
Σ
Hidden
Input
...
...
...

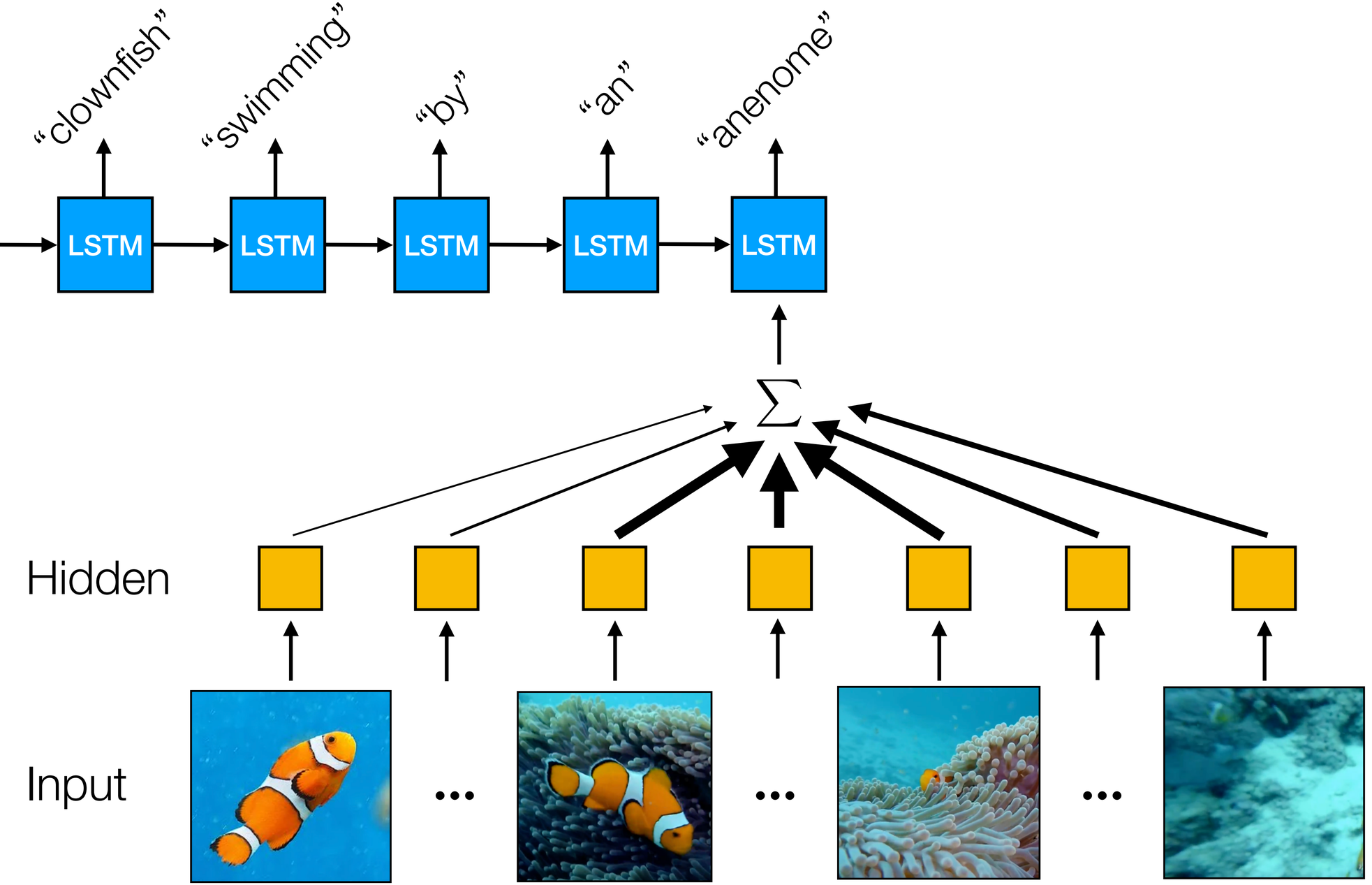
Attention


Outputs
“A”
“clownfish”
“swimming”
“by”
“an”
“anenome”
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
Σ
Hidden
Input
...
...
...

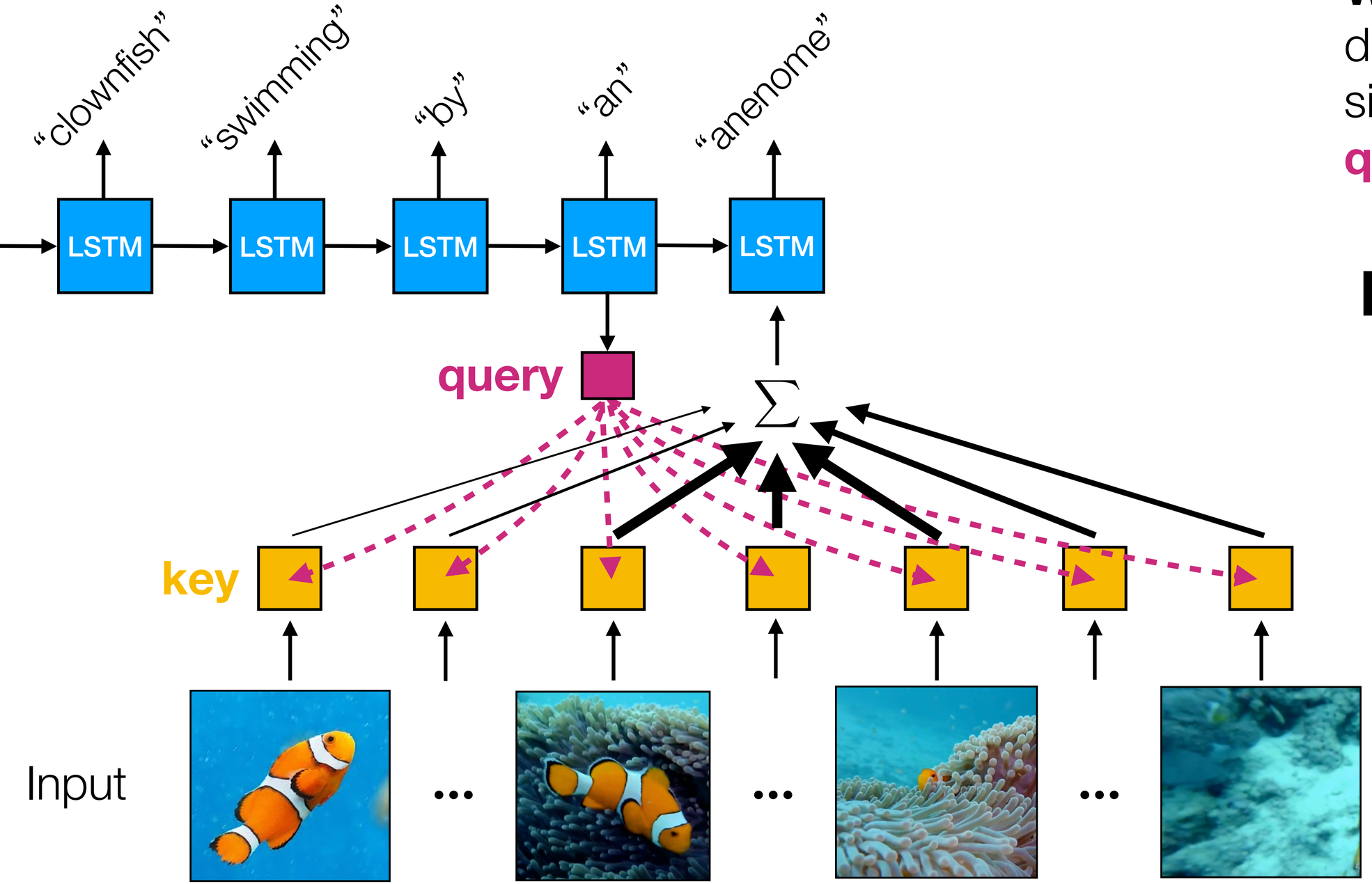
Attention


Outputs
“A”
“clownfish”
“swimming”
“by”
“an”
“anenome”
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
Σ
Hidden
Input
...
...
...

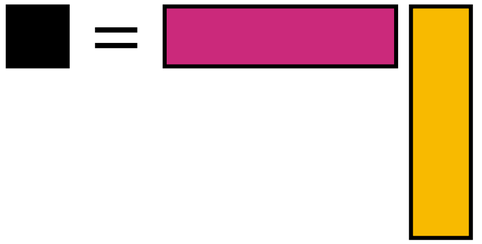
Attention


“clownfish”
“swimming”
“by”
“an”
“anenome”
LSTM
LSTM
LSTM
LSTM
LSTM
Σ
Hidden
Input
...
...
...

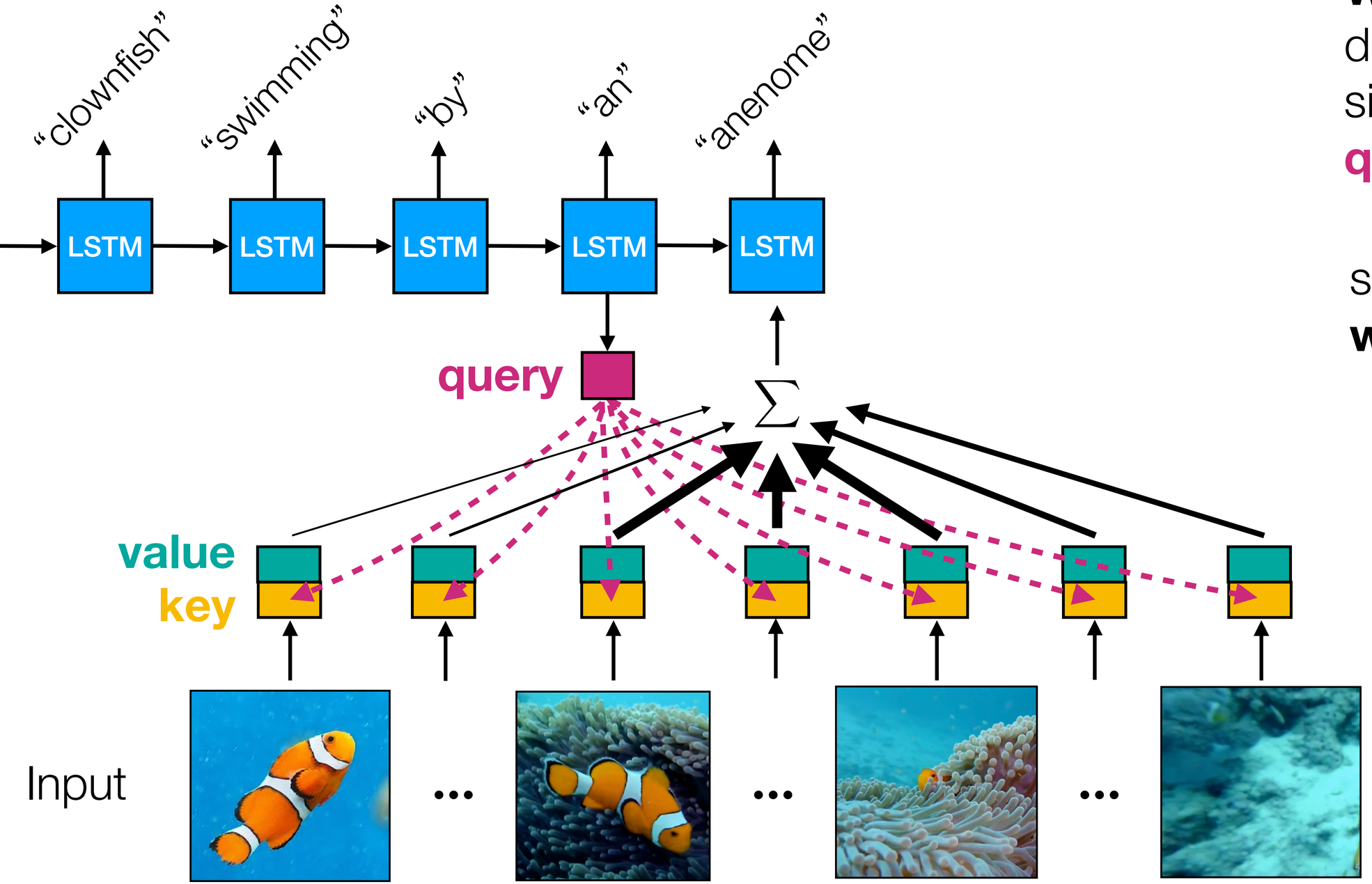
Attention



Weights are determined by similarity between **query** and **key**



Attention

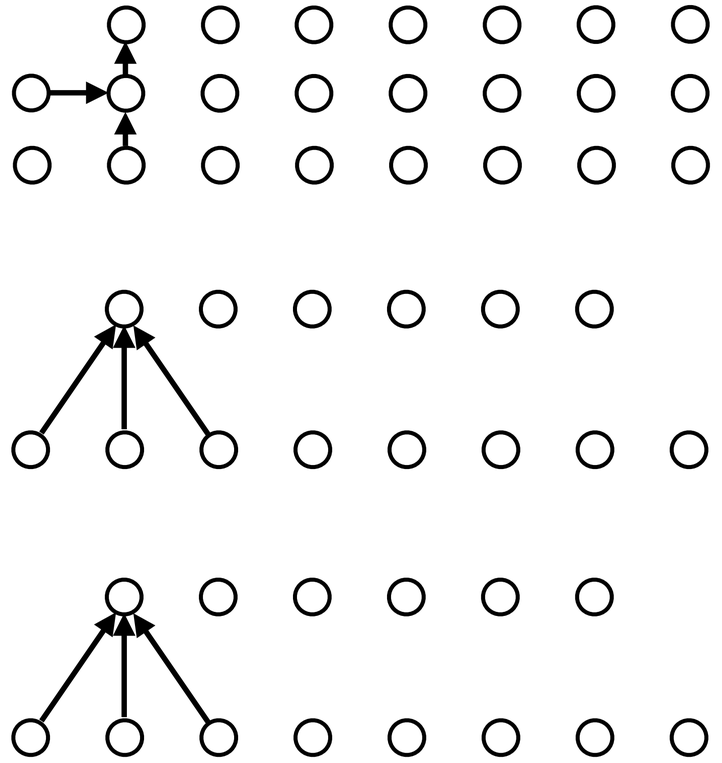


Weights are determined by similarity between **query** and **key**

summation is over **weights** * **value**

Modeling arbitrarily long sequences

- RNNs — recurrent weights are shared across time
- Convolution — conv weights are shared across time
- Attention — weights are dynamically determined



Anything you can do w.r.t. time, you can do w.r.t. space, and vice versa.

Popular right now: treat pixels as a sequence and then apply sequence modeling methods.

Generative Pretraining from Pixels

Mark Chen [1] **Alec Radford** [1] **Rewon Child** [1] **Jeff Wu** [1] **Heewoo Jun** [1] **Prafulla Dhariwal** [1] **David Luan** [1] **Ilya Sutskever** [1]

Abstract

Inspired by progress in unsupervised representation learning for natural language, we examine whether similar models can learn useful representations for images. We train a sequence Transformer to auto-regressively predict pixels, without incorporating knowledge of the 2D input structure. Despite training on low-resolution ImageNet without labels, we find that a GPT-2 scale model learns strong image representations as measured by linear probing, fine-tuning, and low-data classification. On CIFAR-10, we achieve 96.3% accuracy with a linear probe, outperforming a supervised Wide ResNet, and 99.0% accuracy with full fine-tuning, matching the top supervised pre-trained models. An even larger model trained on a mixture of ImageNet and web images is competitive with self-supervised benchmarks on ImageNet, achieving 72.0% top-1 accuracy on a linear probe of our features.

ported strong results using a single layer of learned features (Coates et al., 2011), or even random features (Huang et al., 2014; May et al., 2017). The approach fell out of favor as the state of the art increasingly relied on directly encoding prior structure into the model and utilizing abundant supervised data to directly learn representations (Krizhevsky et al., 2012; Graves & Jaitly, 2014). Retrospective study of unsupervised pre-training demonstrated that it could even hurt performance in modern settings (Paine et al., 2014).

Instead, unsupervised pre-training flourished in a different domain. After initial strong results for word vectors (Mikolov et al., 2013), it has pushed the state of the art forward in Natural Language Processing on most tasks (Dai & Le, 2015; Peters et al., 2018; Howard & Ruder, 2018; Radford et al., 2018; Devlin et al., 2018). Interestingly, the training objective of a dominant approach like BERT, the prediction of corrupted inputs, closely resembles that of the Denoising Autoencoder, which was originally developed for images.

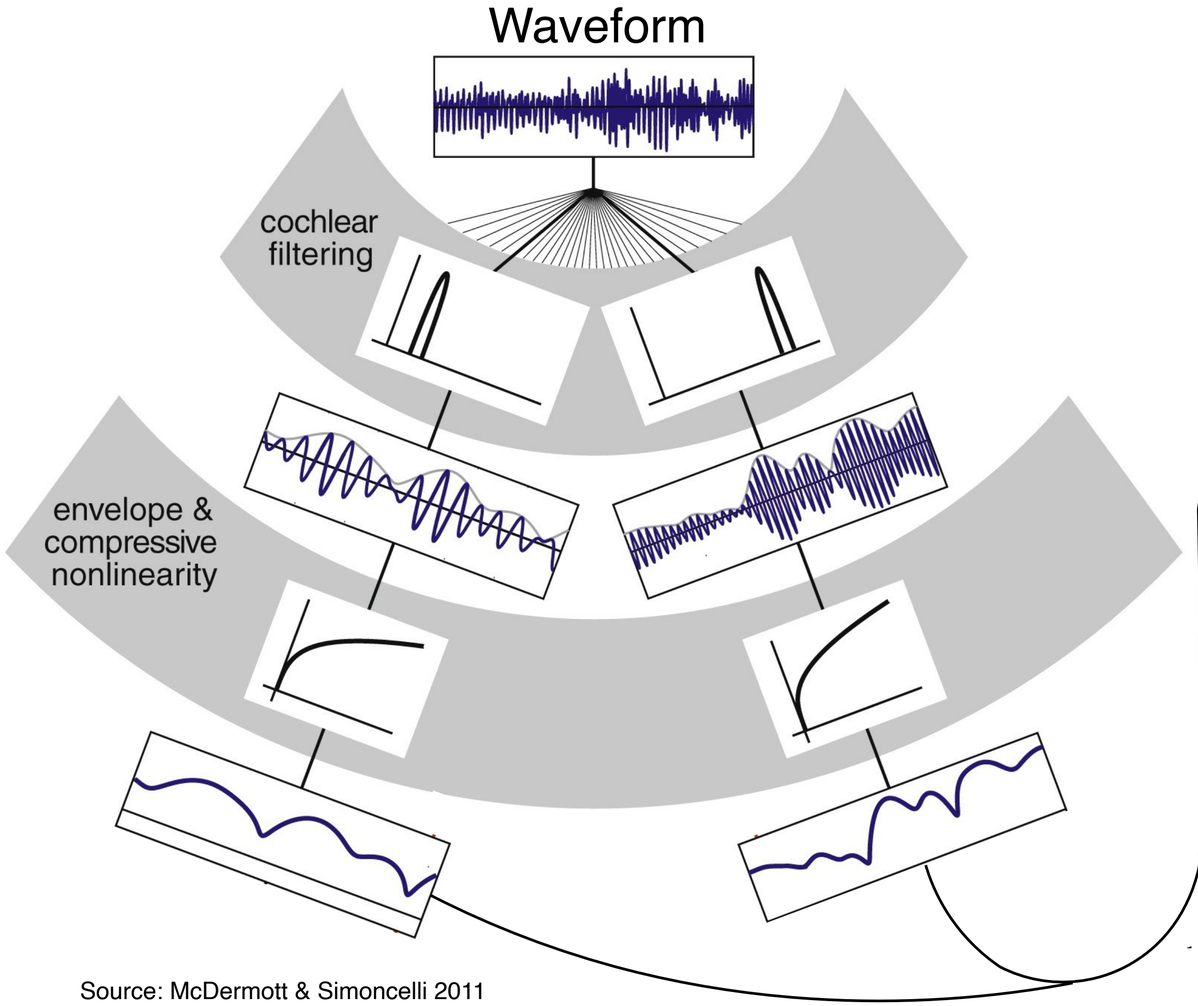

6.867
office hour
RODE

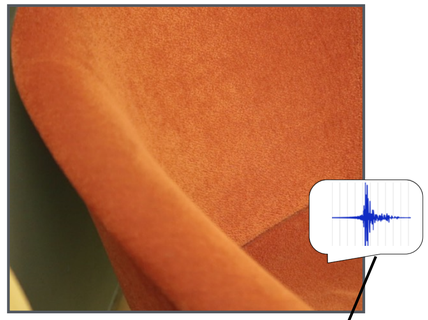
The *Greatest Hits* dataset

The *Greatest Hits* dataset

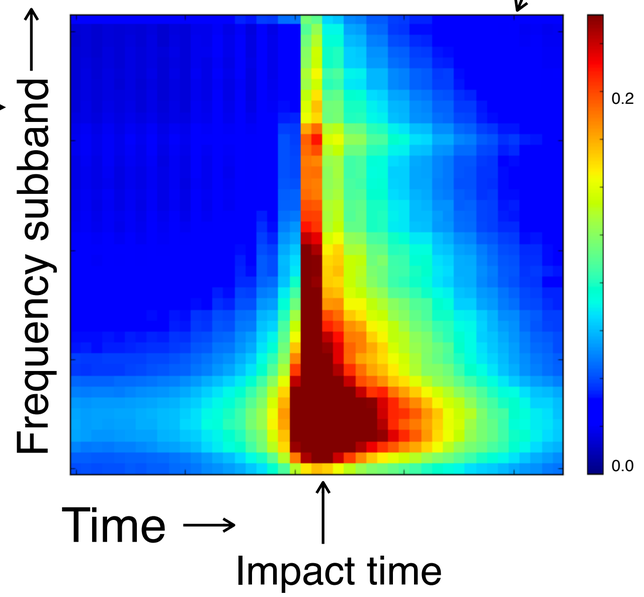
- 978 videos of people probing scenes with a drumstick
- 46,620 hits and scratches
- Material, action, and reaction labels (used for analysis)

Sound and materials



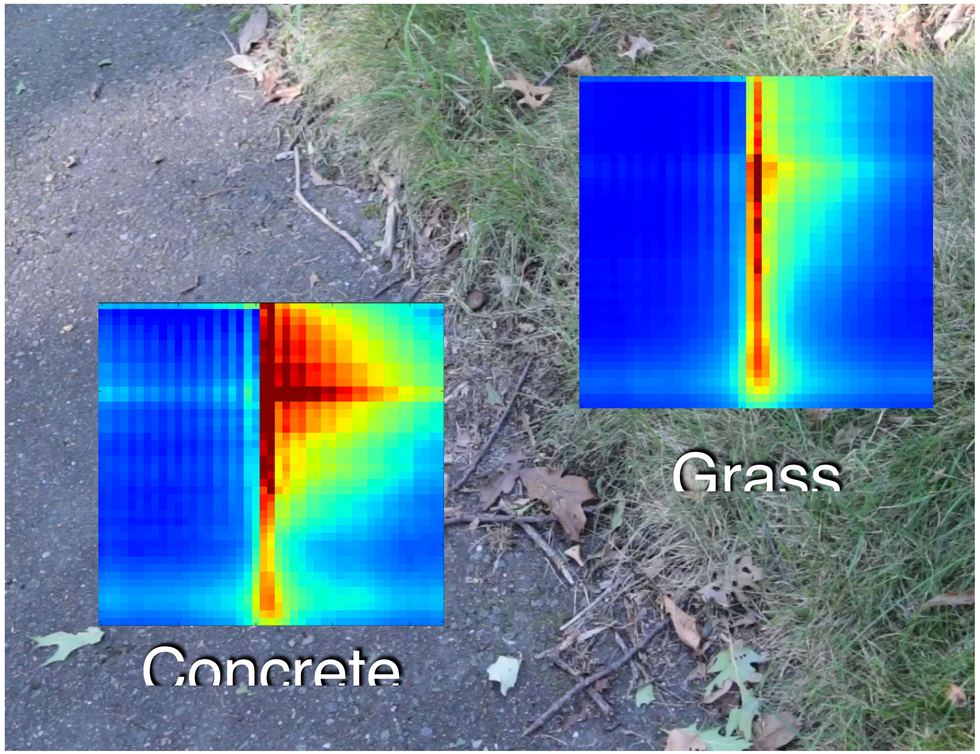


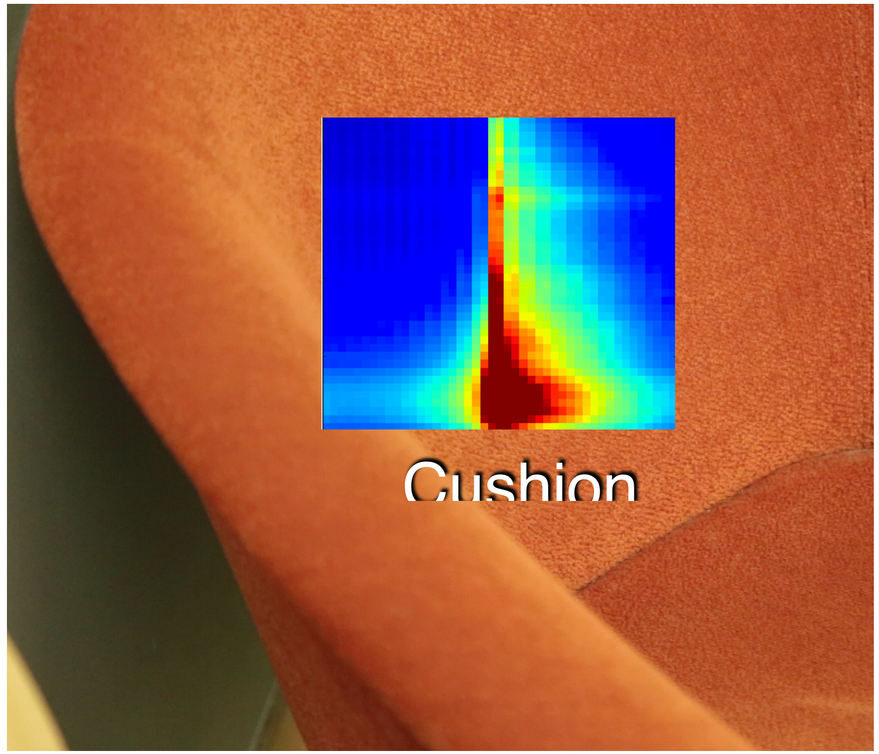
Mean cushion cochleagram



Source: McDermott & Simoncelli 2011

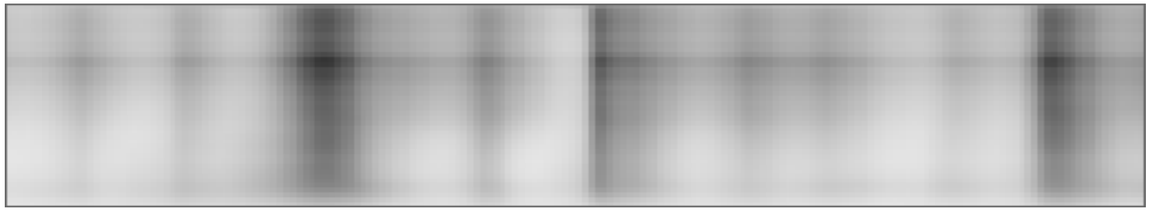
Sound and materials





Predicting sound features

Cochleagram

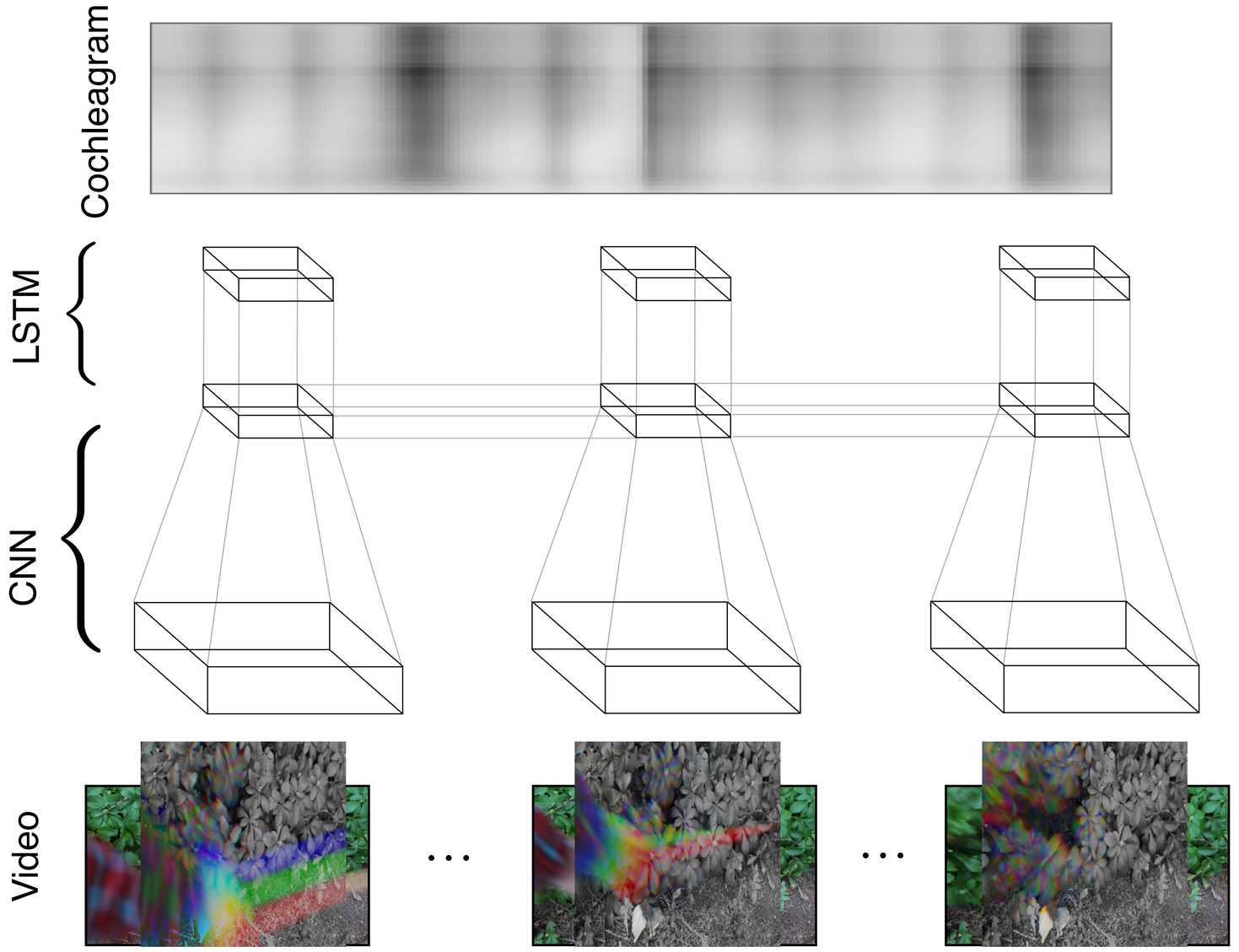


Regression

Video

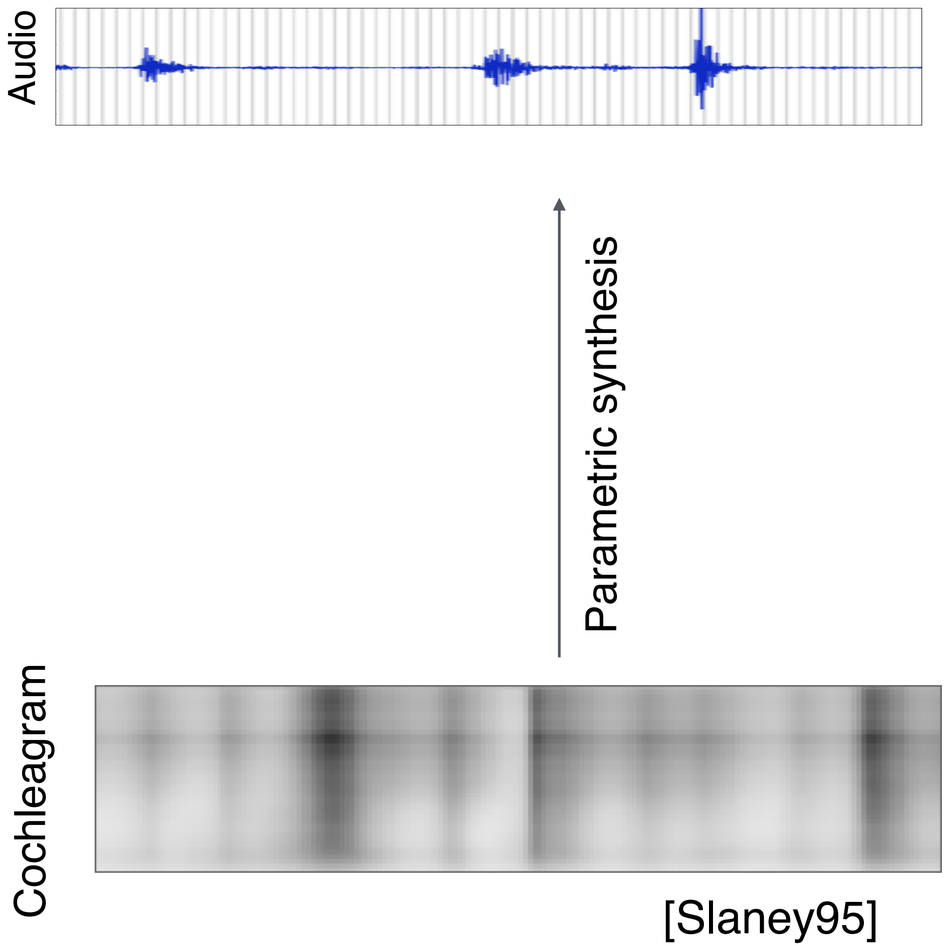
Time ⟶

Predicting sound features

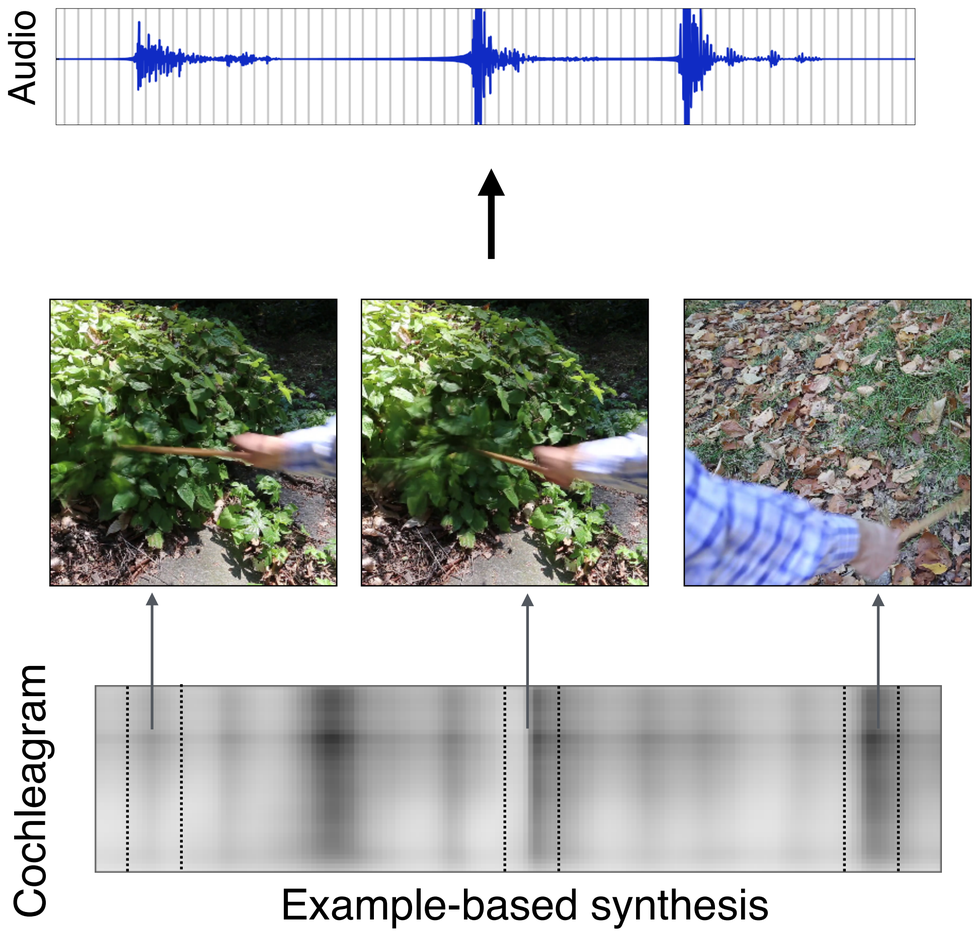


- Two-stream CNN: color + spacetime images

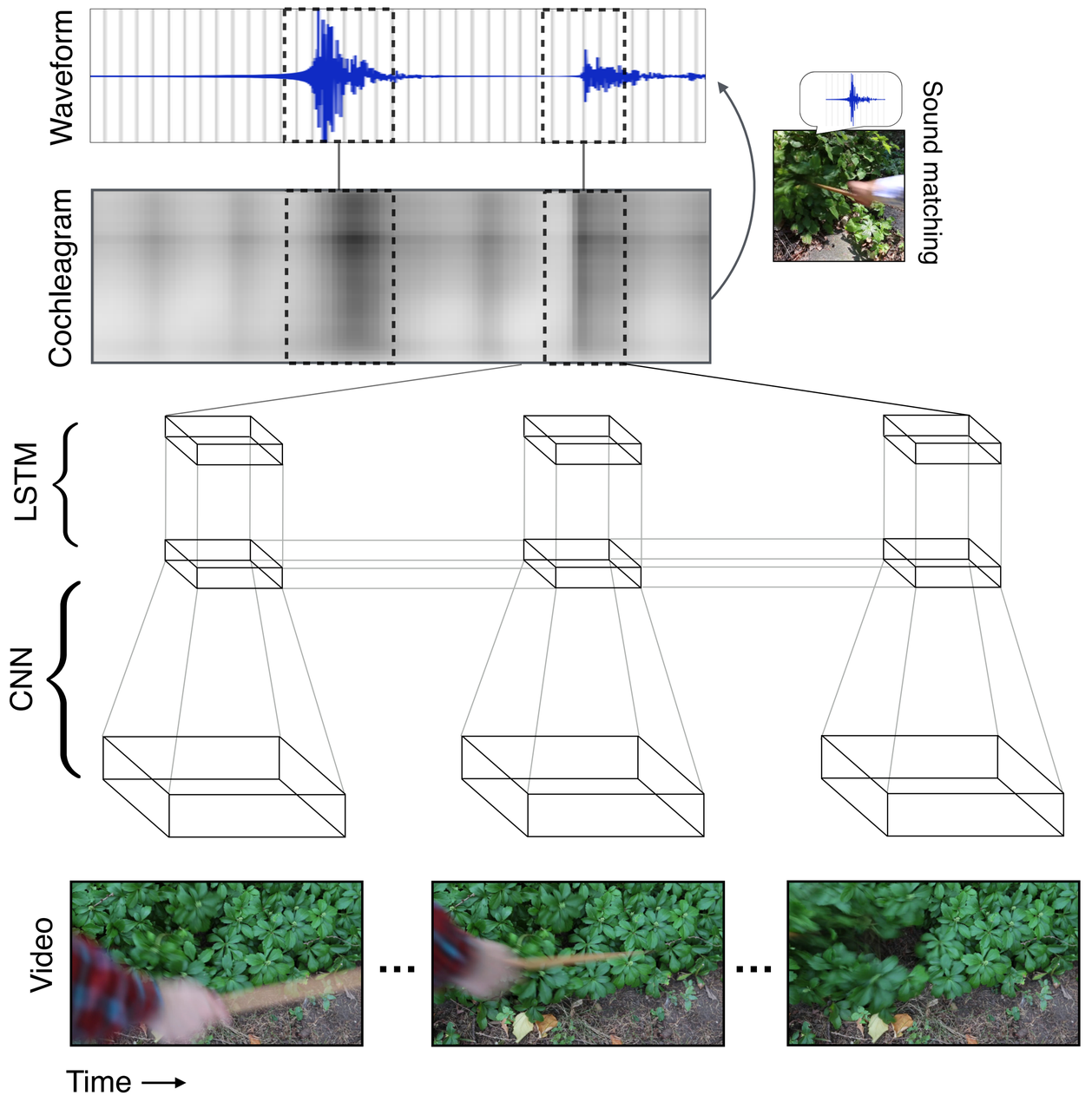
Generating a waveform



Generating a waveform



Recap: full model



Some results

Our output

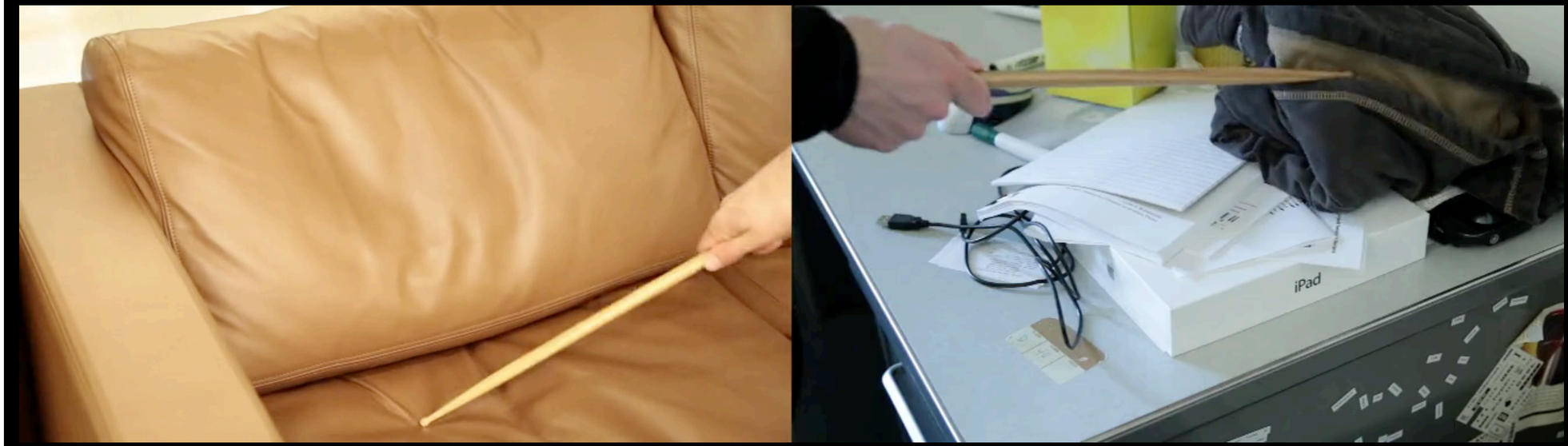
Original sound source

Our output

Original sound source

Our output

Original sound source




Predicted sound


Other actions?


Other actions?


Predicted sound