Lecture 14: Video and temporal models

Source: A. Torralba

Source: A. Torralba

Source: A. Torralba

Source: A. Torralba

Simple video task: action recognition

Making latte art

Jaywalking

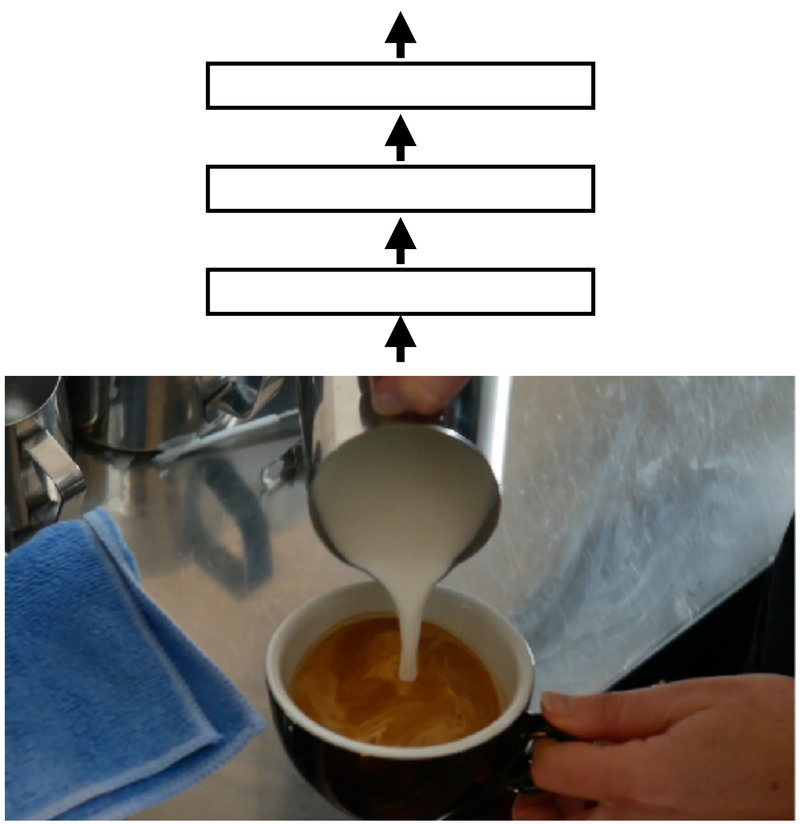
Grooming dog

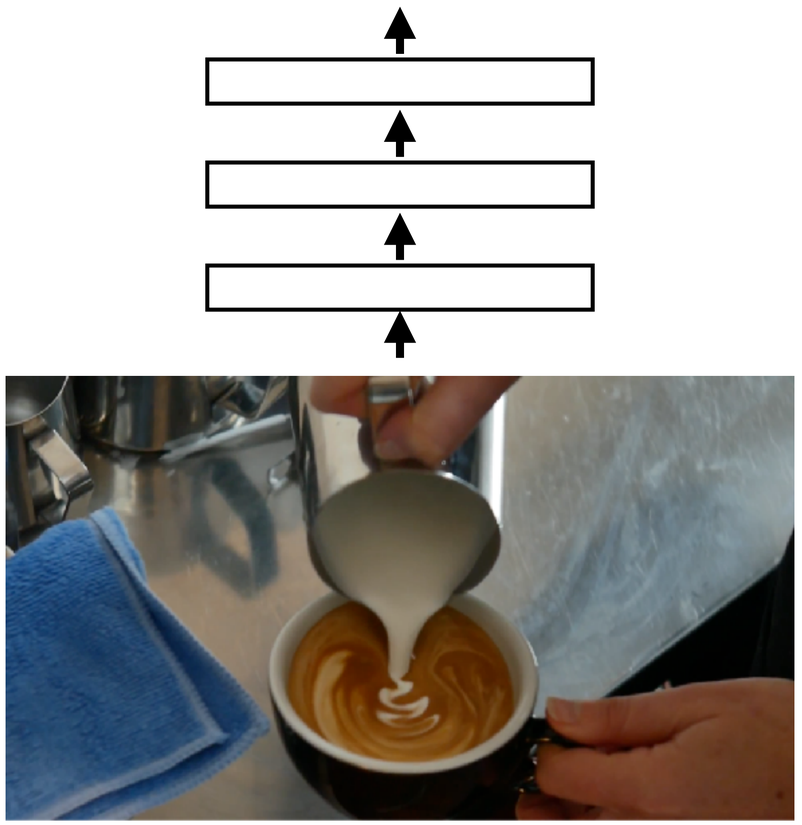
Analogous to image recognition. Useful testbed for designing video models.

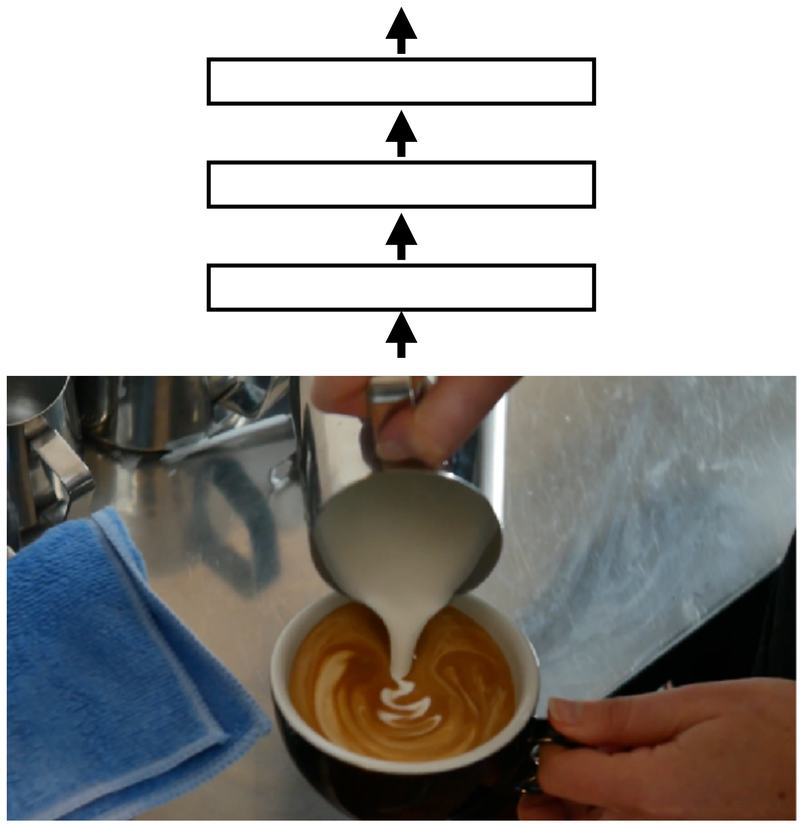
Examples from the Kinetics dataset [Carreira et al. 2017 - 2019]
700 human activity classes, 650K 10-sec videos

Simple model: averaging

$\frac{1}{3}$ p(making latte art | I_1) + $\frac{1}{3}$ p(making latte art | I_2) + $\frac{1}{3}$ p(making latte art | I_3)







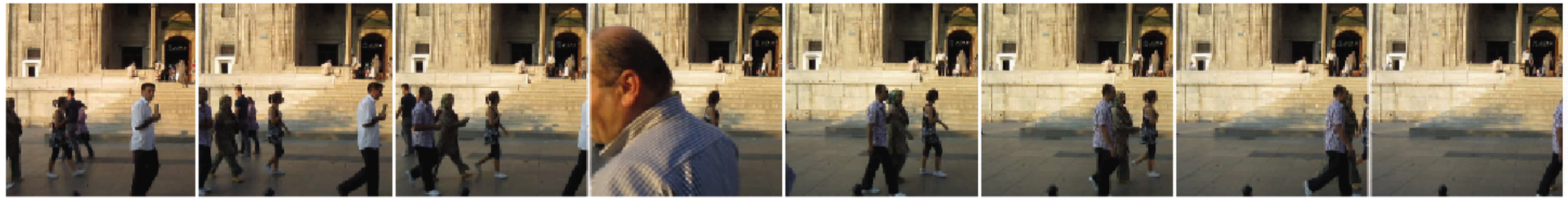
Can’t analyze motion.

Temporal filtering

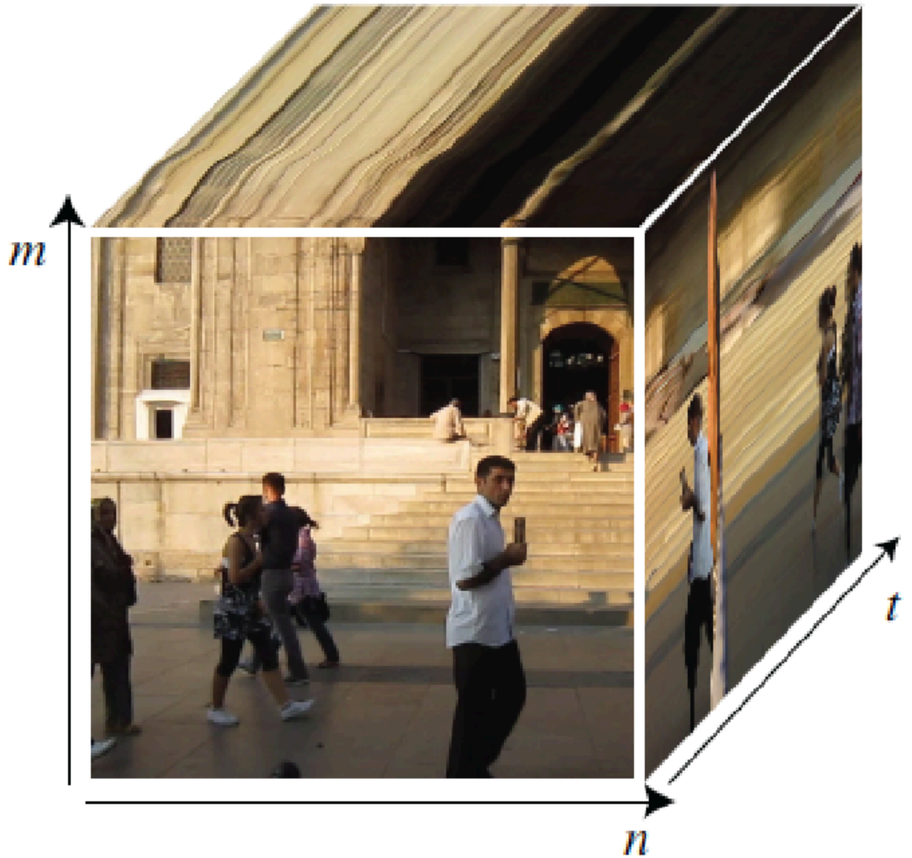
Temporal filtering

Source: Freeman, Torralba, Isola

Videos

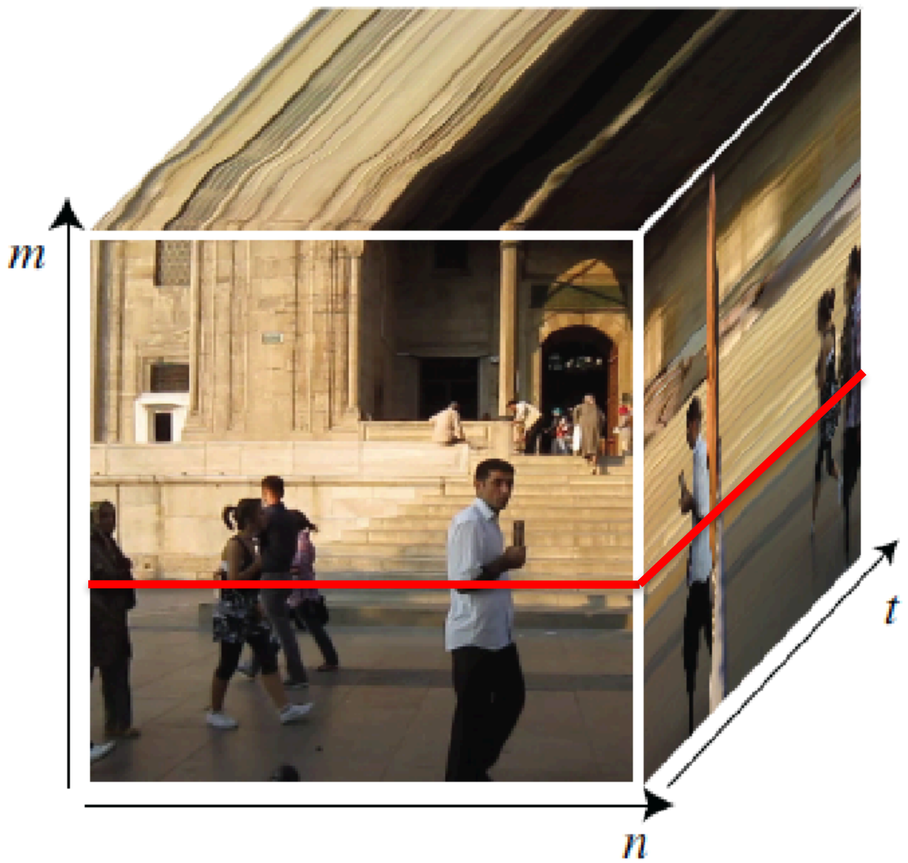


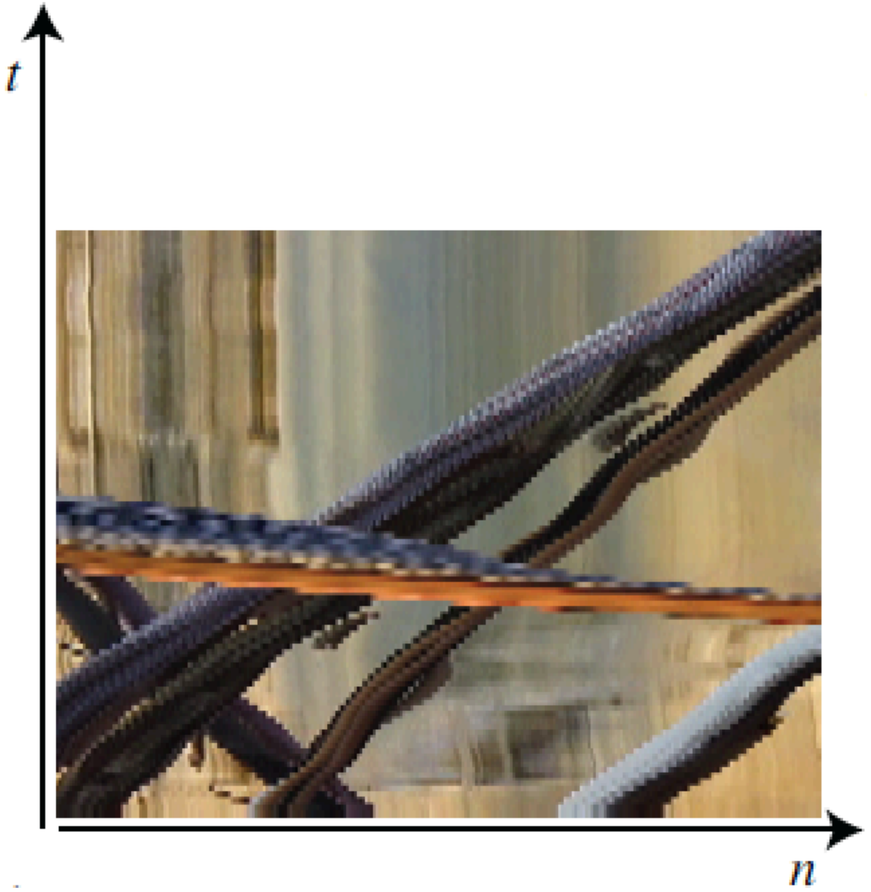
time



Source: Freeman, Torralba, Isola

Videos

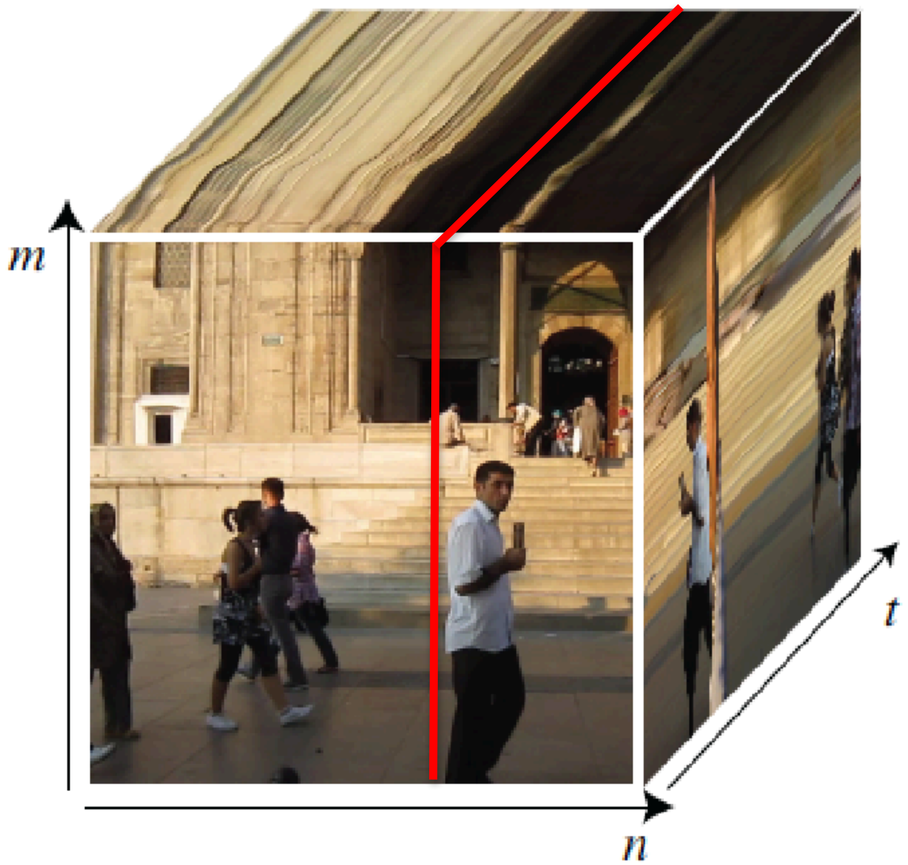


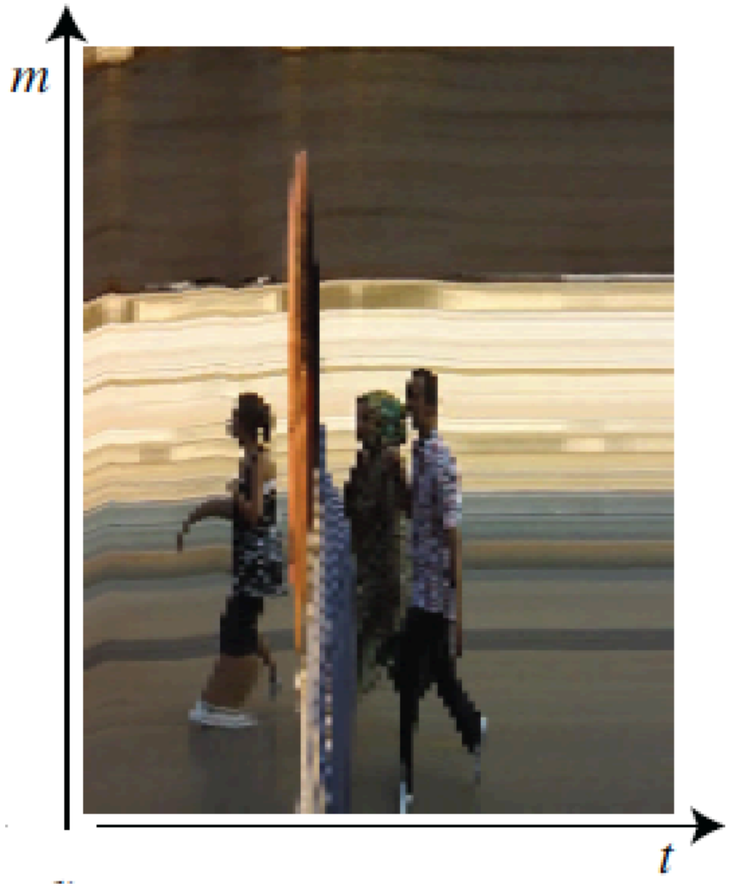


Cube size = 128x128x90

Source: Freeman, Torralba, Isola

Videos





Cube size = 128x128x90

Source: Freeman, Torralba, Isola

Examples of temporal filtering

Temporal median filtering

Source: Alexei Efros

Background subtraction

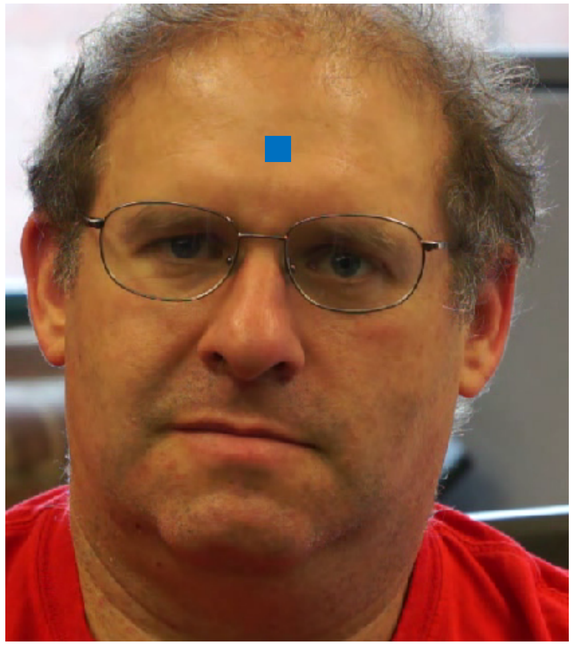
-

=

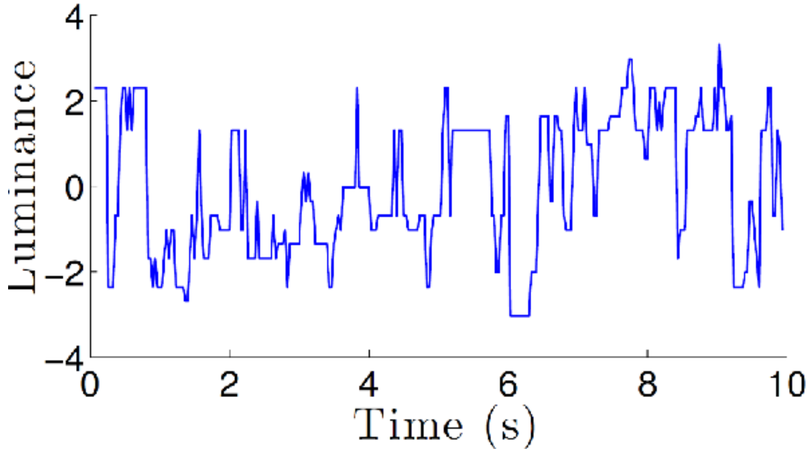
Source: Alexei Efros

Finding subtle color variations

- The face gets slightly redder when blood flows



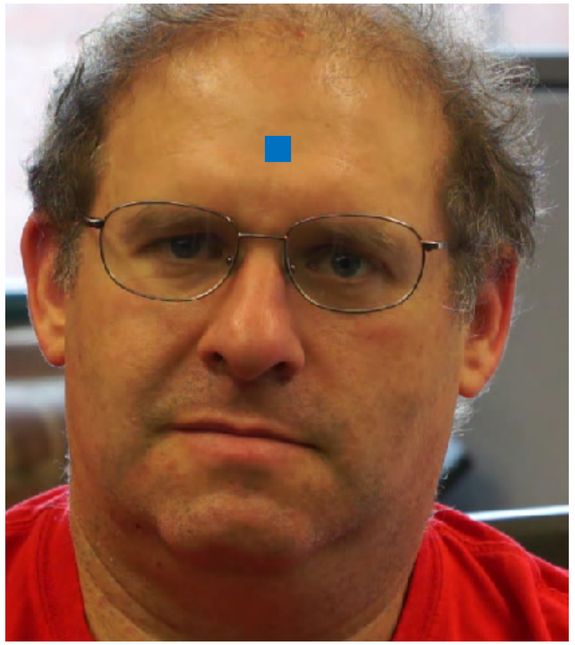
Input frame

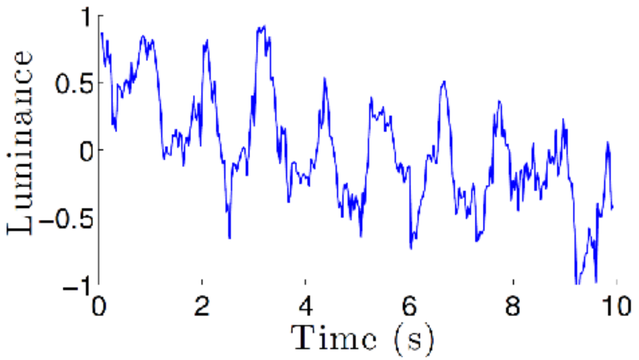


Luminance trace (zero mean)

Source: D. Hoiem

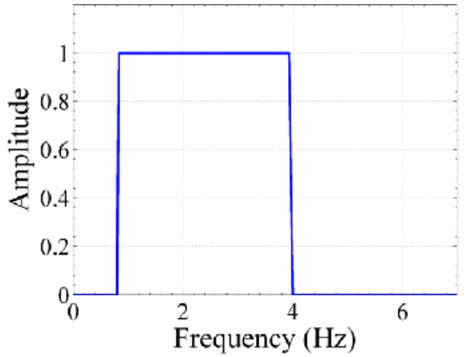
Amplifying Subtle Color Variations





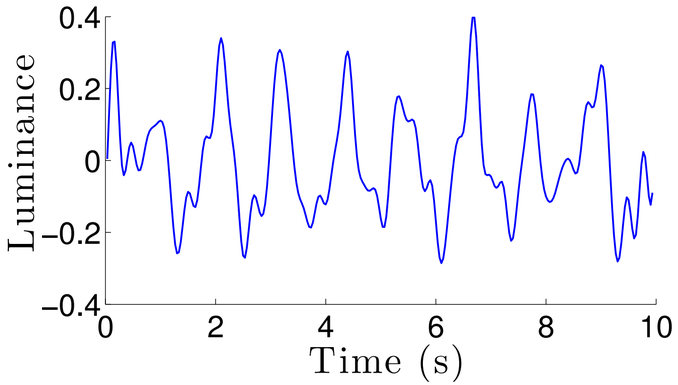
Spatially averaged luminance

⊗



Temporal filter

=

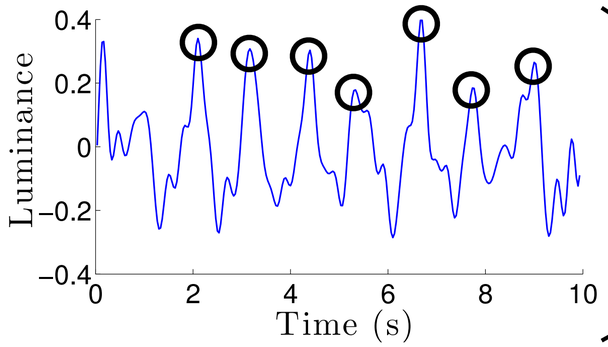


Result

Source: D. Hoiem

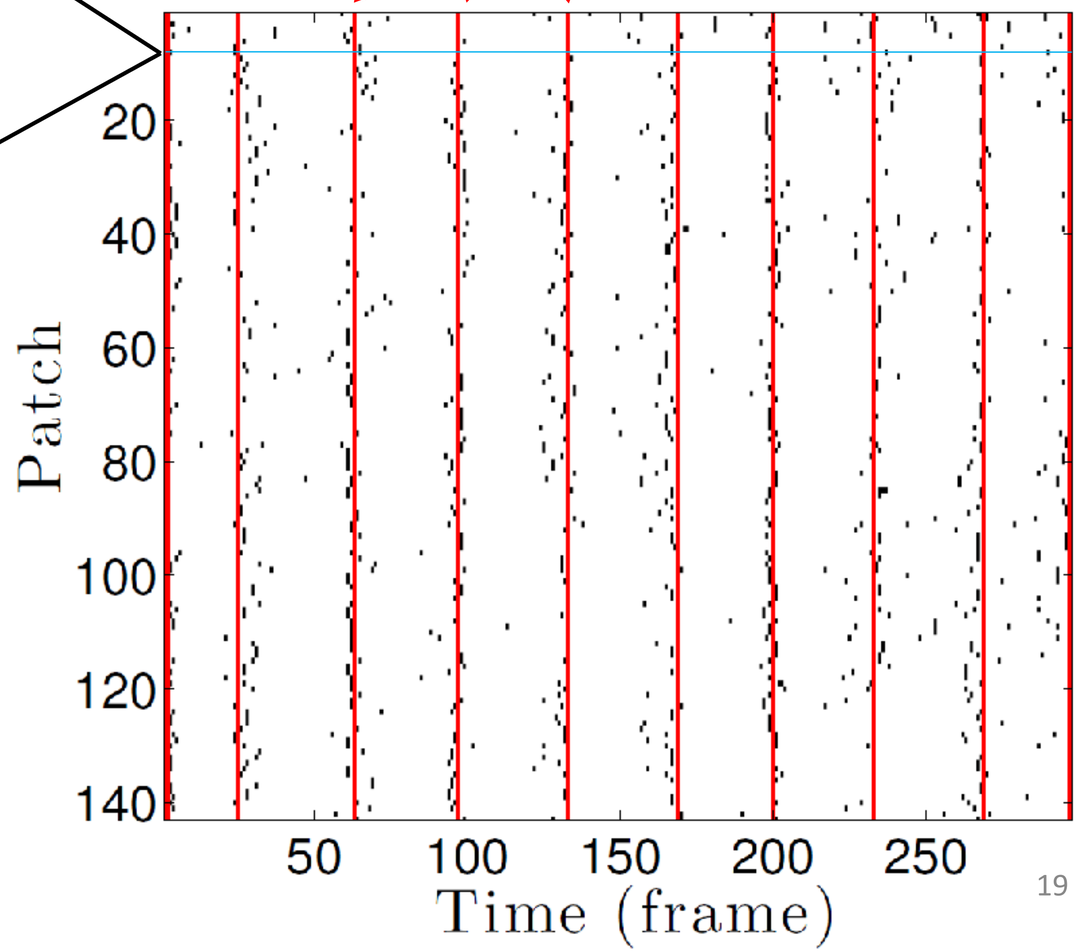
Heart Rate Extraction

Peak detection



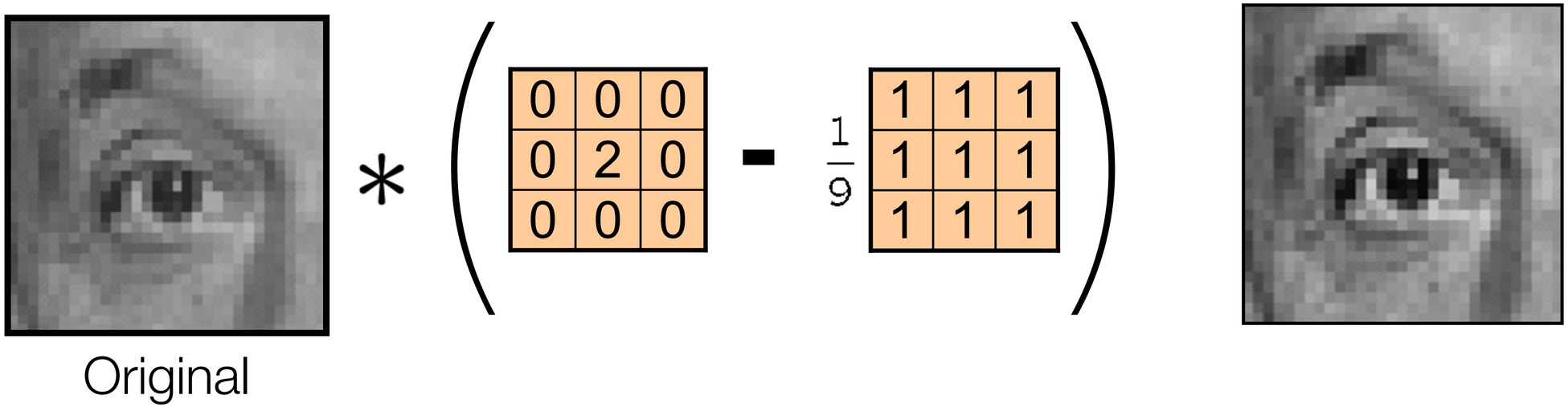
Temporally bandpassed trace
(one pixel)

Pulse locations



Source: D. Hoiem

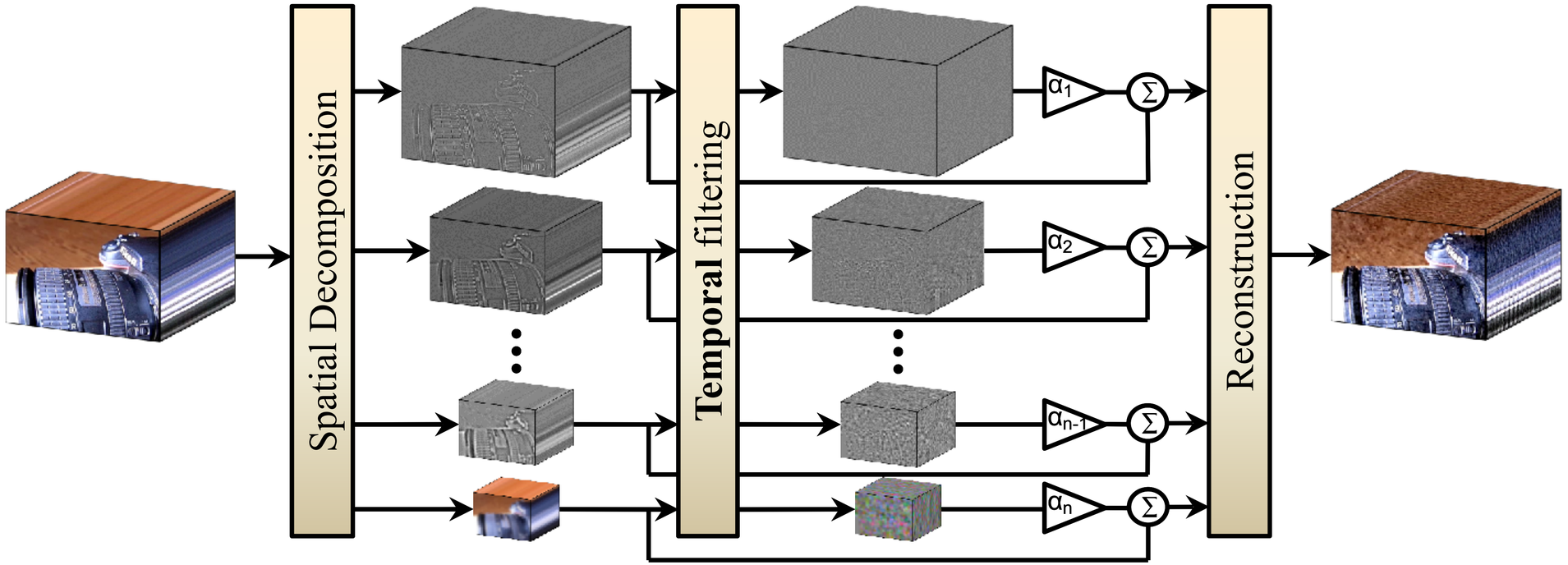
Recall: sharpening filter



Original

Source: D. Lowe

Magnifying tiny motions



Laplacian Pyramid

Bandpass filter intensity at each pixel over time

Amplify bandpassed signal and add back to original

Source: D. Hoiem

Color amplification

Source: [Wadhwa et al., Phase-based Video Motion Processing. SIGGRAPH 2013, Wu et al. SIGGRAPH 2012]

Motion magnification

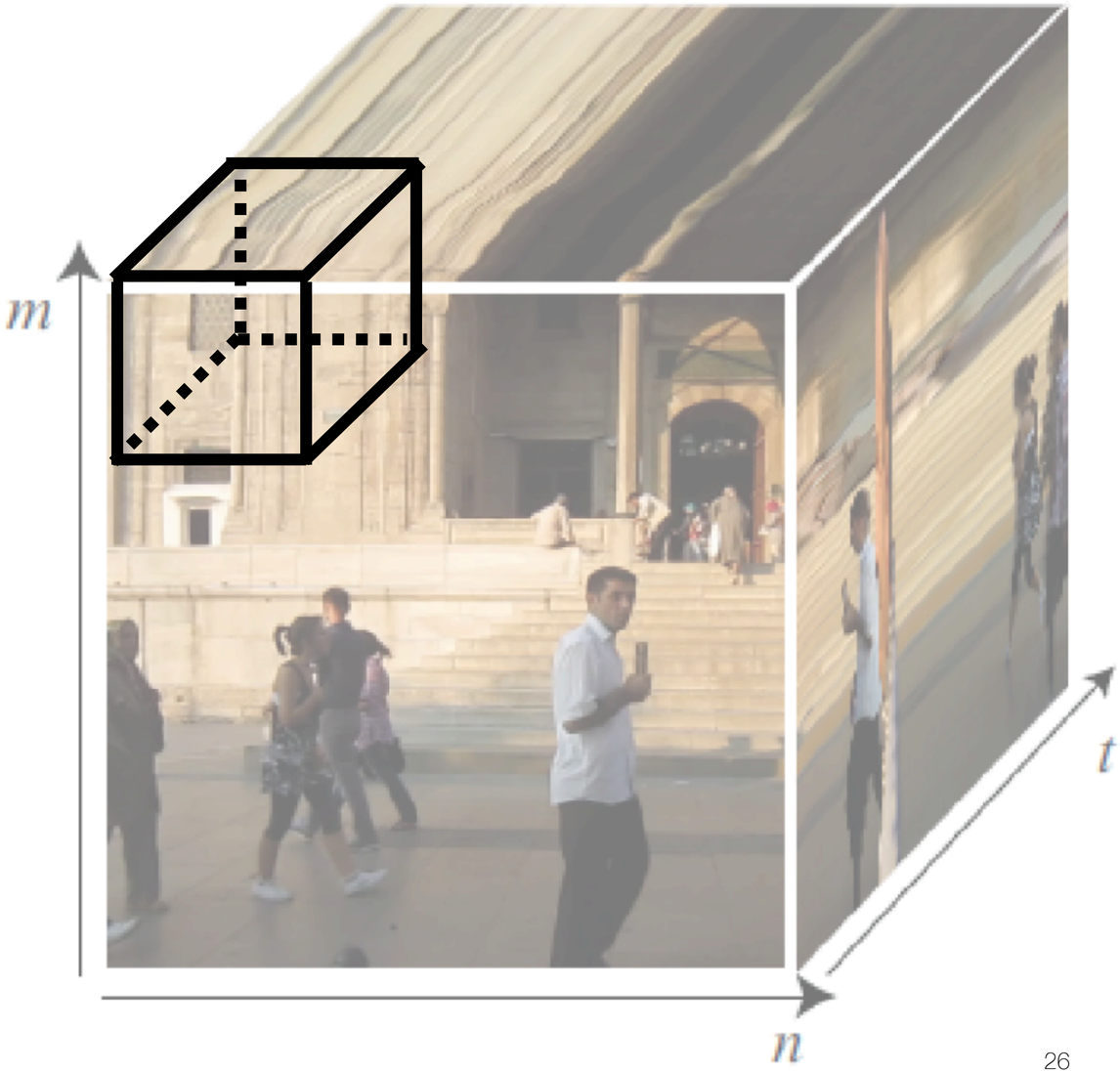
Original

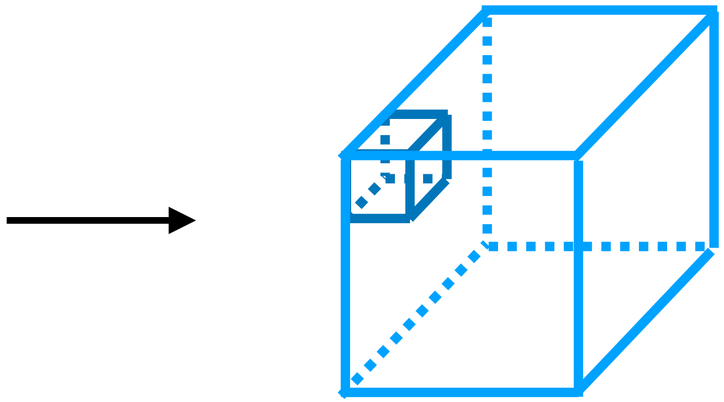
[Wu et al., SIGGRAPH 2012]

[Wadhwa et al., SIGGRAPH 2013]

Space-time convolutions

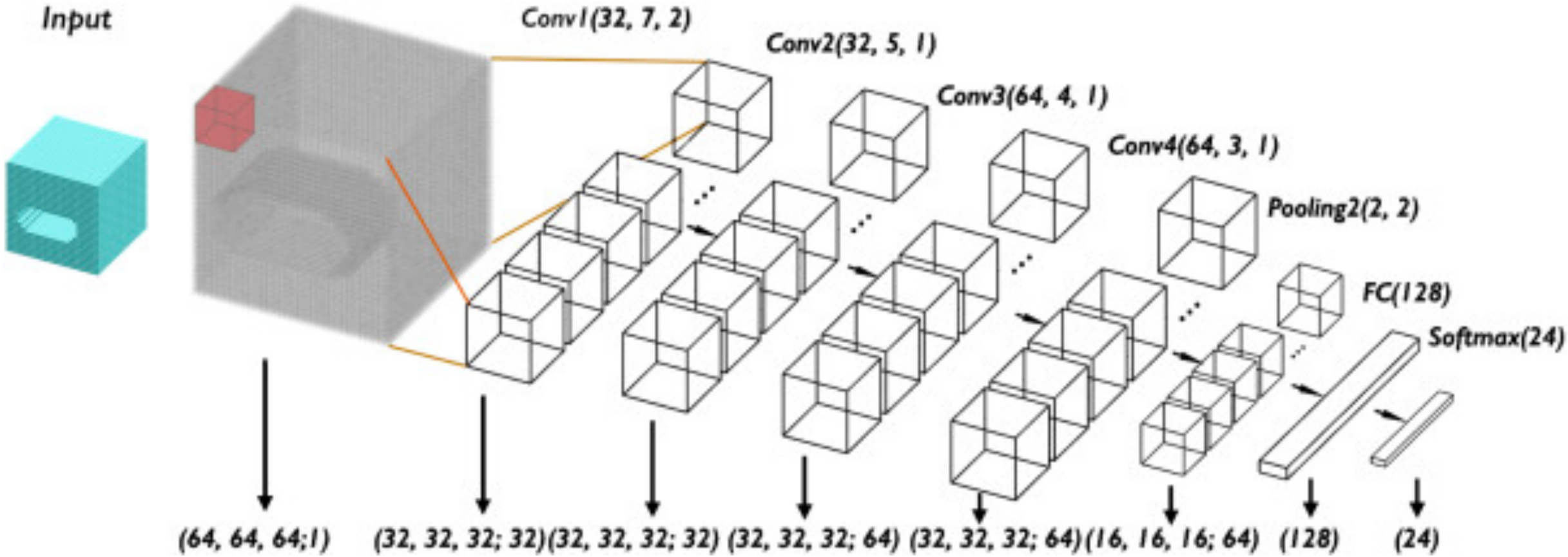
3D space-time convolution





Source: Torralba, Freeman, Isola

3D space-time CNN

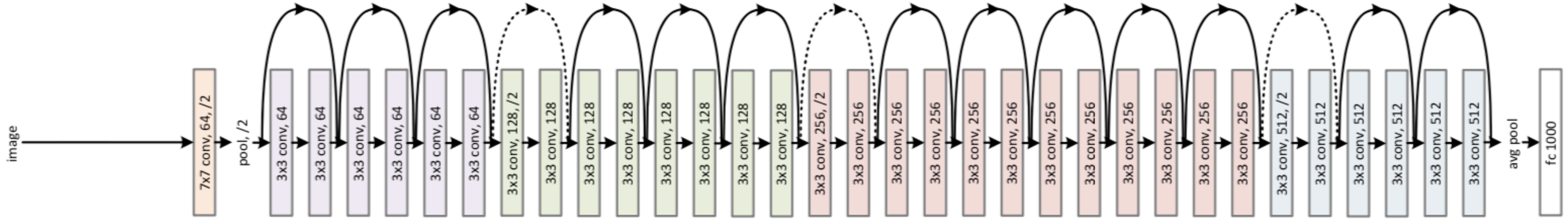


[Source: FeatureNet: Machining feature recognition based on 3D Convolution Neural Network]

Source: Torralba, Freeman, Isola

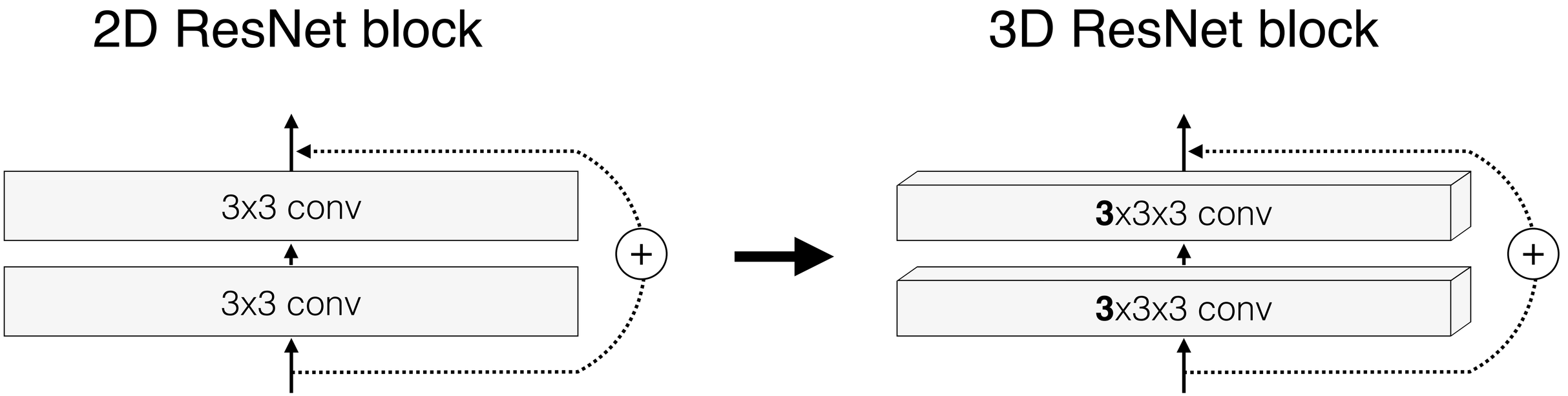
Designing a 3D CNN architecture

Starting point: 2D image CNNs



ResNet [Kaiming He et al. 2016]

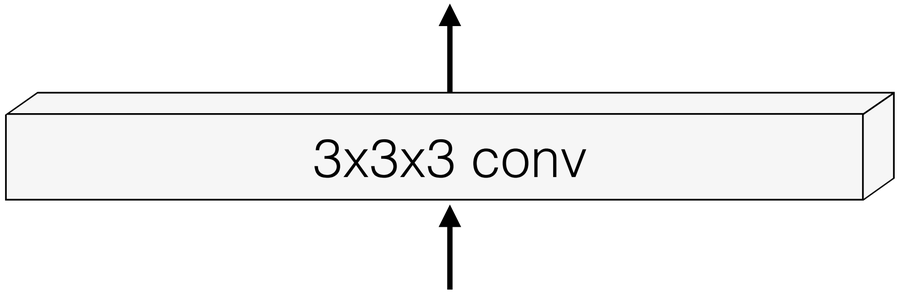
Inflated convolutions



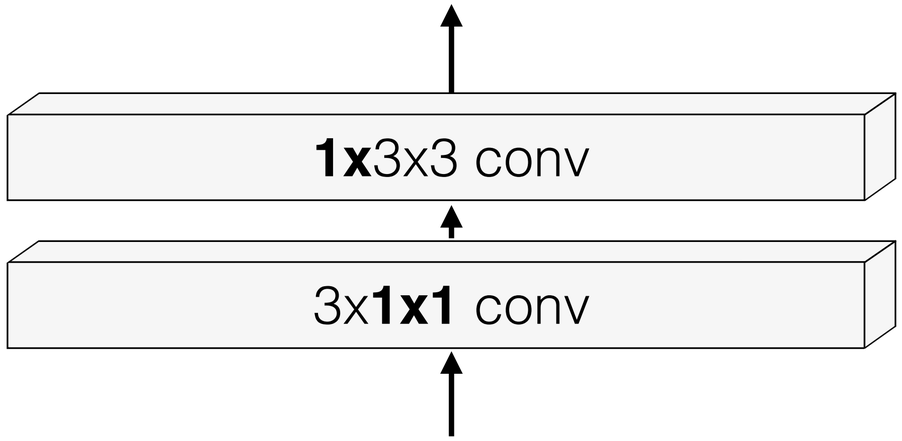
- Can reuse 2D architectures. [Carreira et al. 2017]
- Pretrain with 2D nets (“inflating” 2D filter to 3D)

Separable convolutions

3D convolution



Separate space/time

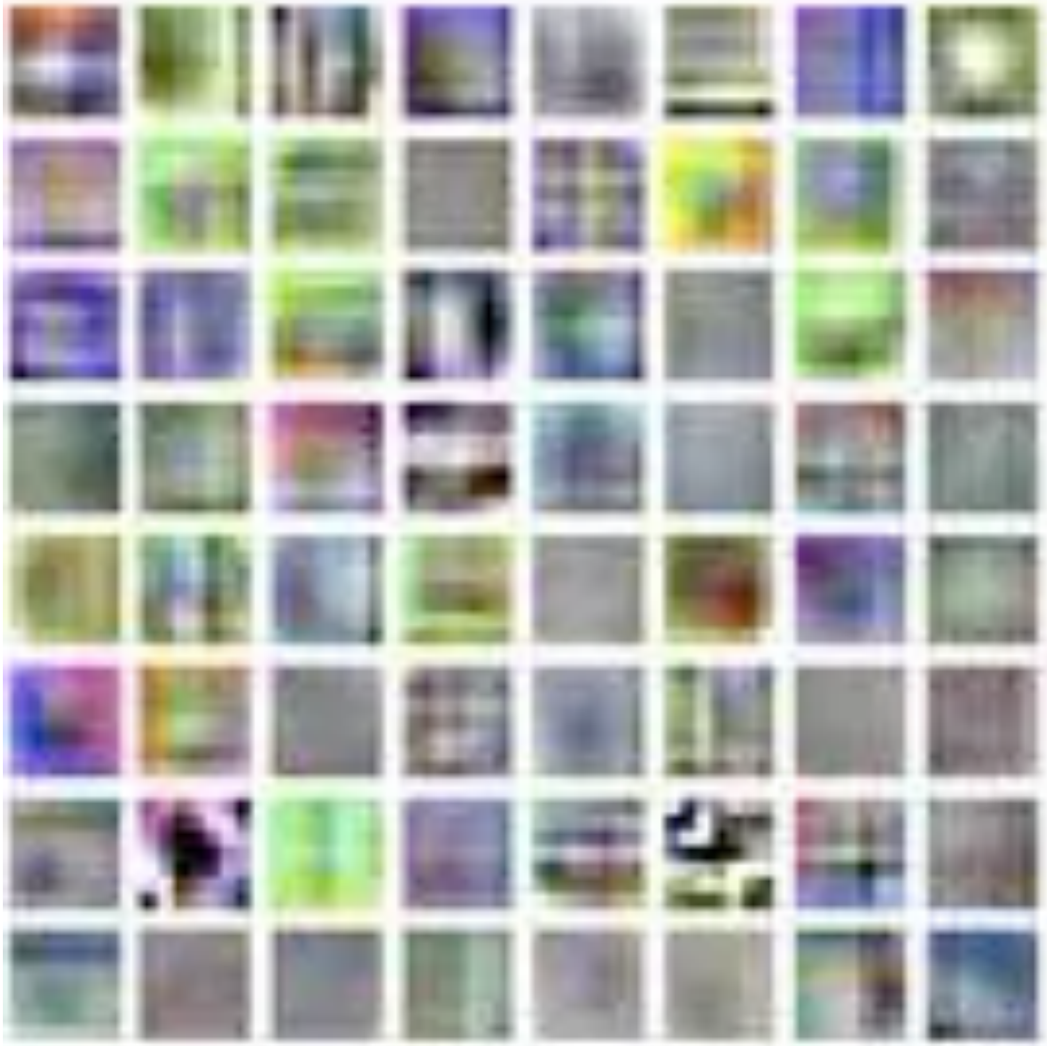


Often works well. Faster and fewer parameters.

$3 \times 3 \times 3 = 27$ vs. $3 \times 3 + 3 = 12$

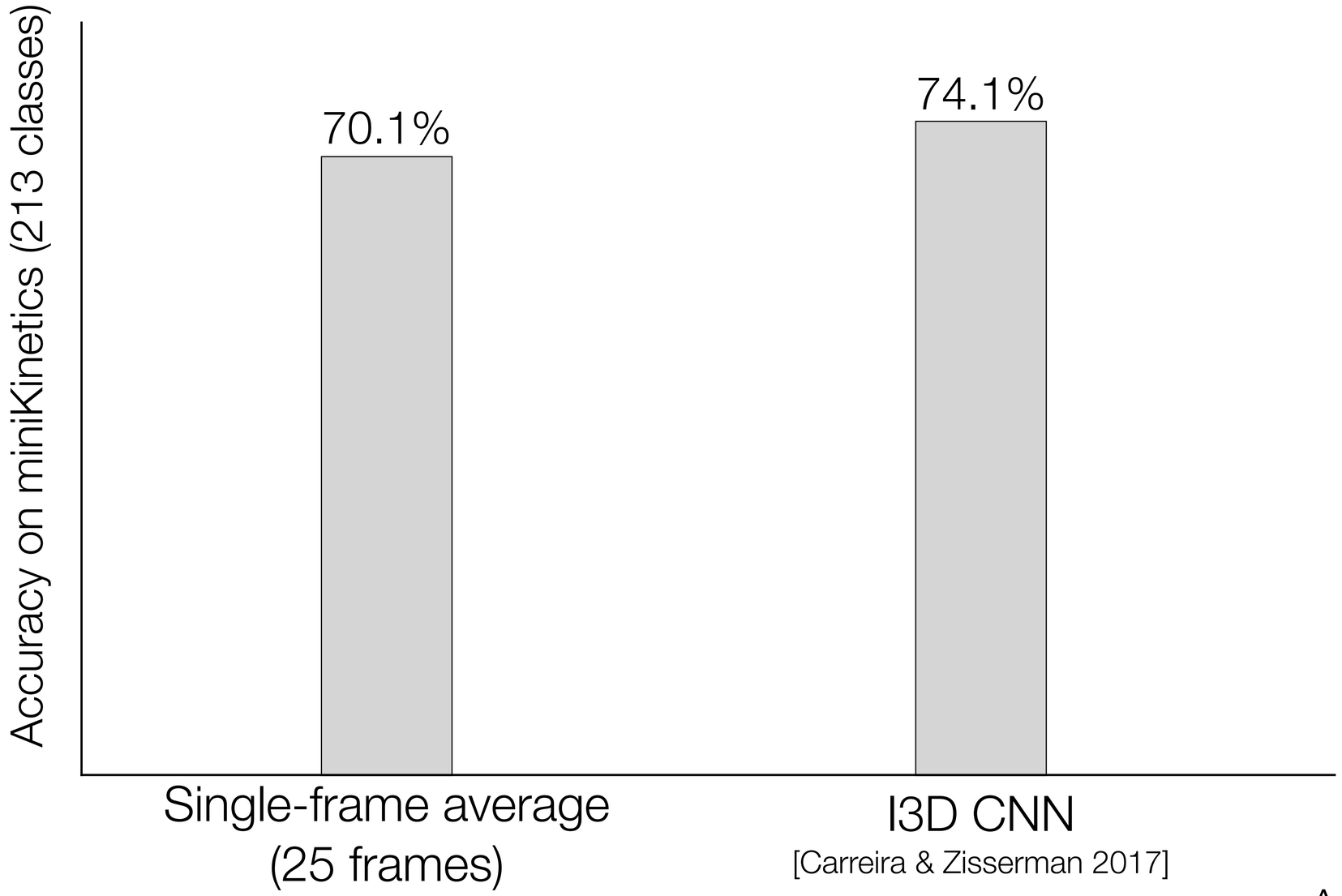
[Tran et al., 2018], [Xie et al., 2018]

Learned space-time filters



$(7 \times 7 \times 7)$ I3D conv1 filters, [Carreira & Zisserman 2017]

When do we actually need motion?



Adapted from David Fouhey

When do we actually need motion?

Let’s look at these again:

Making latte art

Jaywalking

Grooming dog

When do we actually need motion?

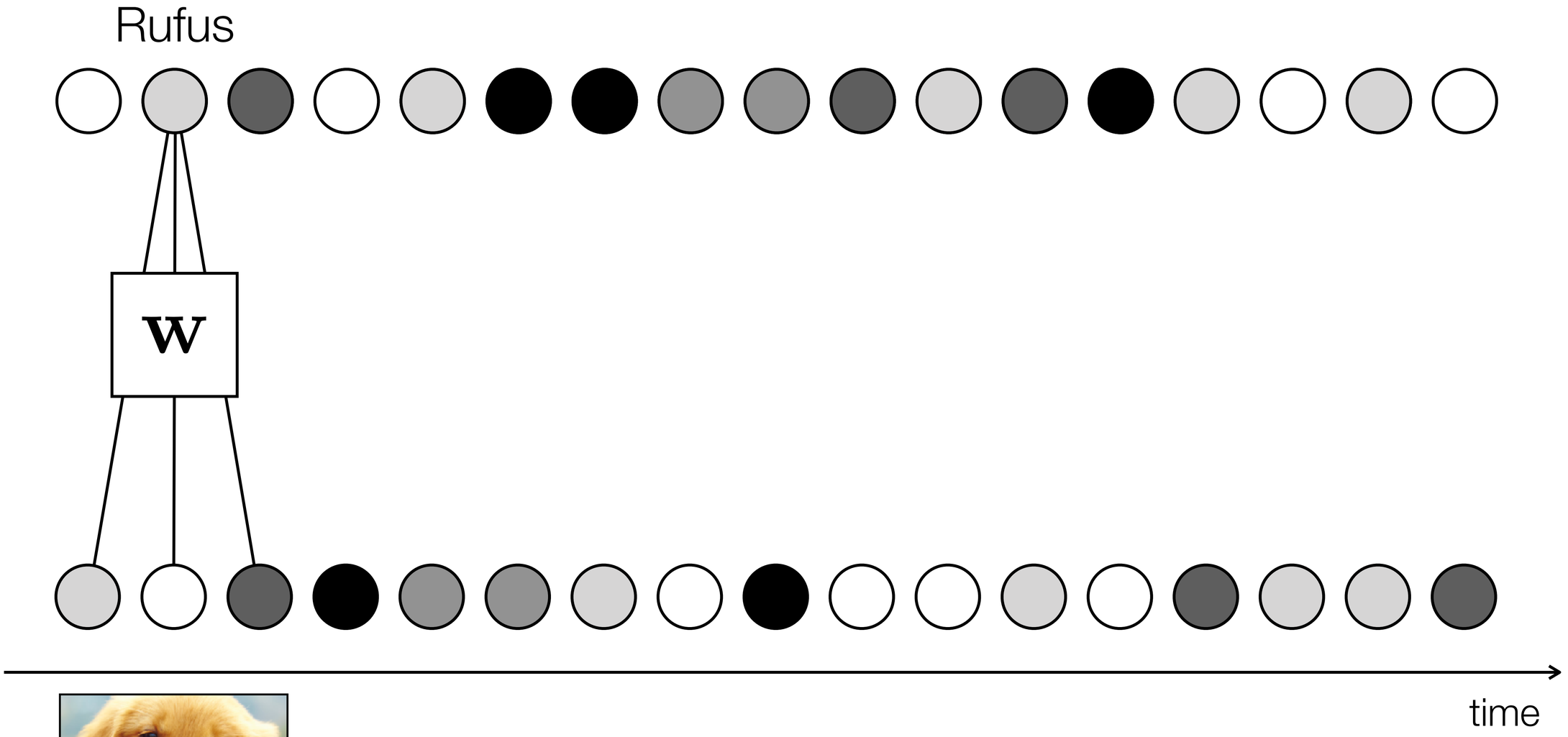
Photo by H. Edgerton

- Single-frame models usually don’t work well.
- But single-image model + temporal pooling often is surprisingly competitive.
- In many tasks that use video, time often provides **extra samples**, rather than motion.
- Later in the course, we’ll see tasks where motion is essential, such as 3D reconstruction.

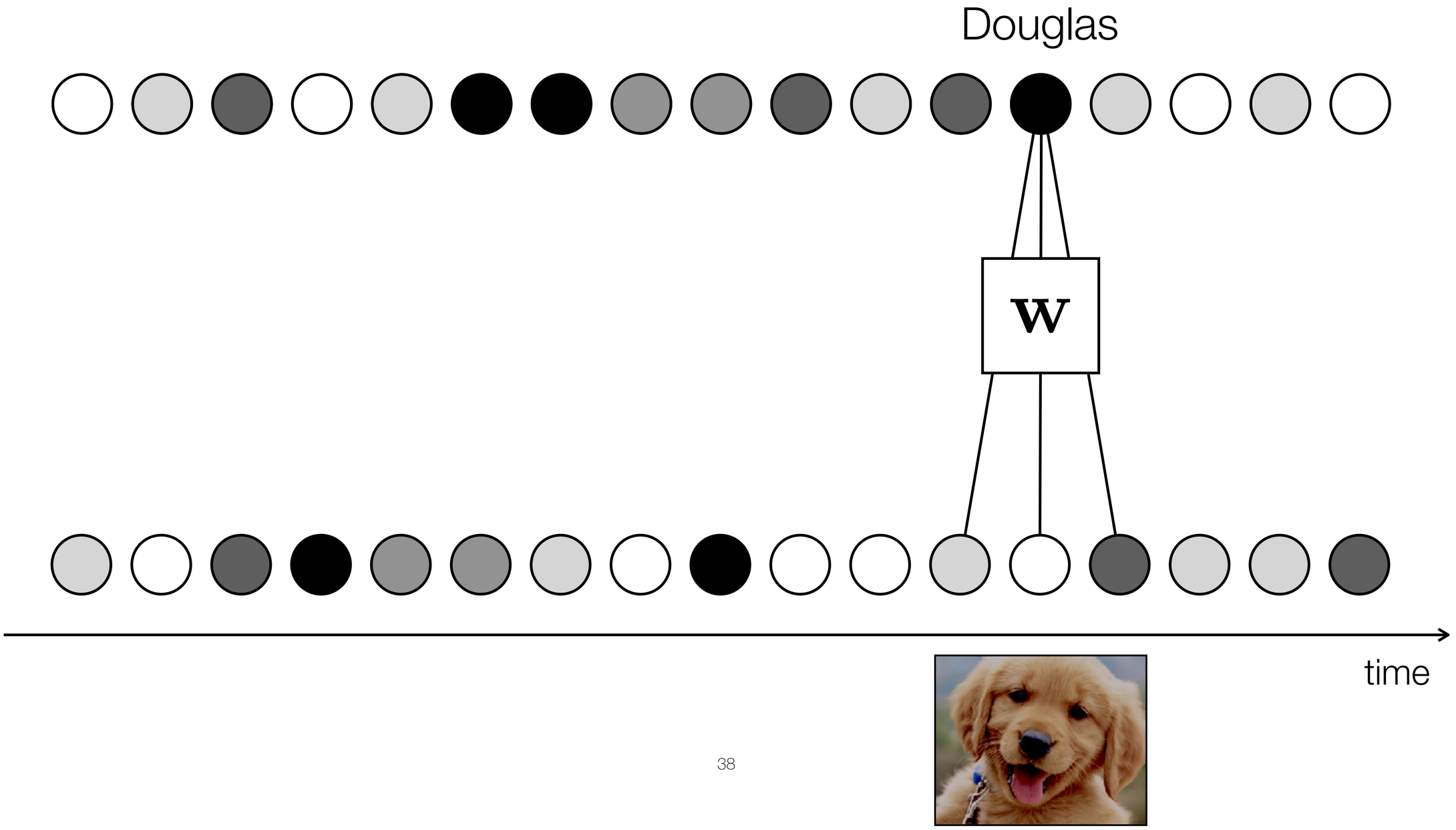
Recurrent networks

[https://www.youtube.com/watch?v=wxfGT-kKxiM]

Source: Torralba, Freeman, Isola



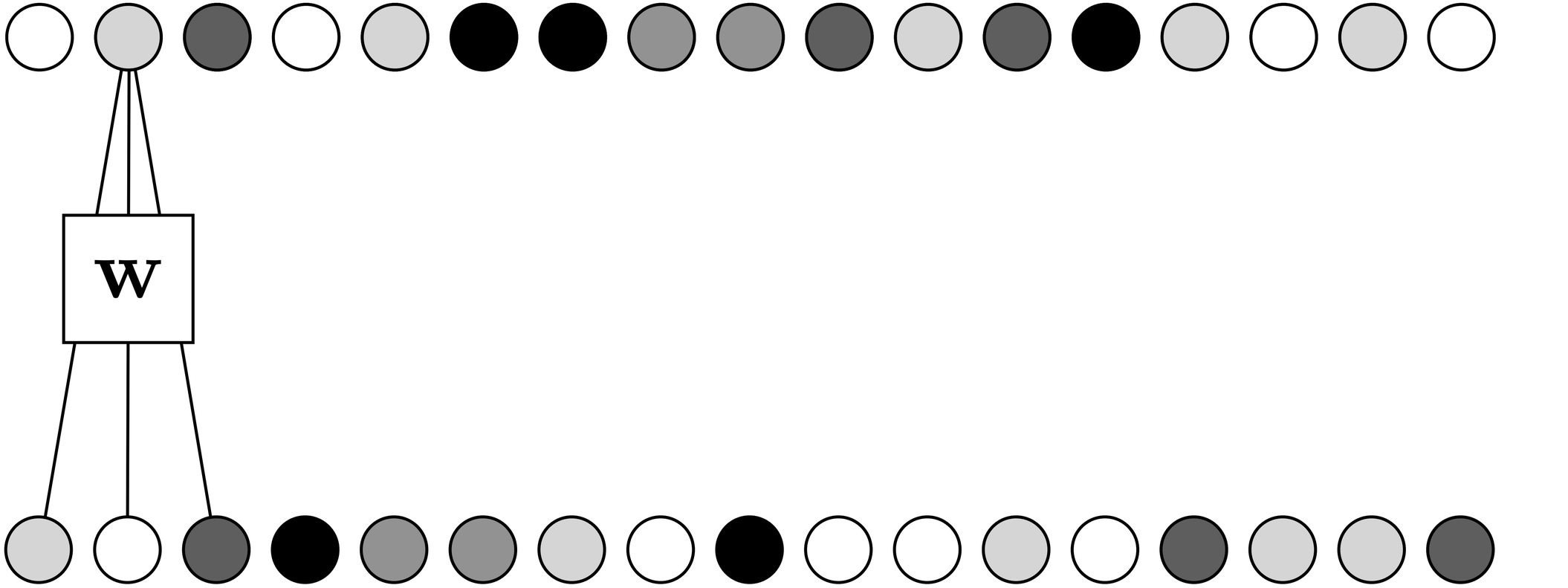
Source: Torralba, Freeman, Isola



Source: Torralba, Freeman, Isola

Memory unit

Rufus



time

Source: Torralba, Freeman, Isola

Memory unit

Rufus

Rufus!

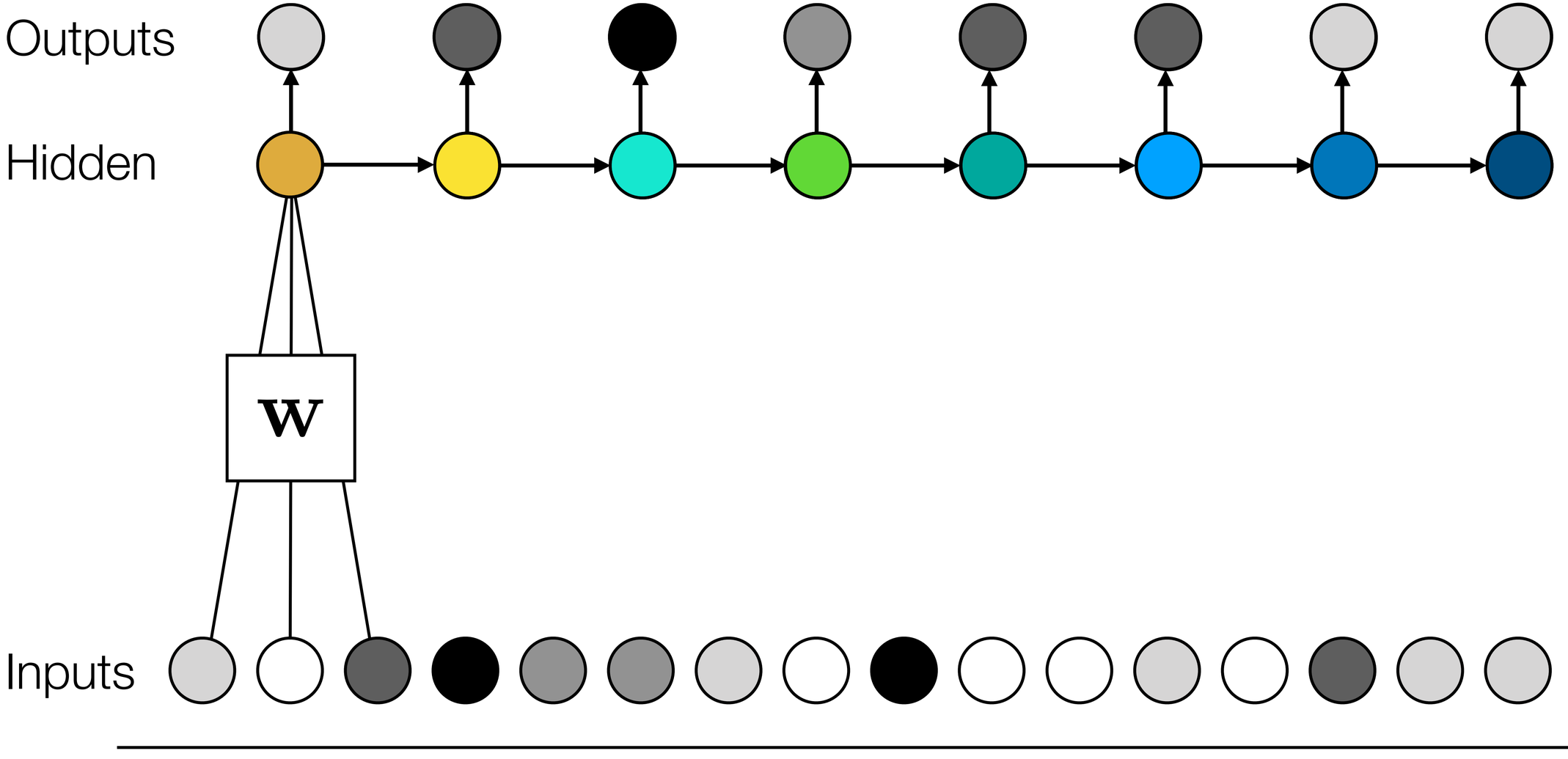
W

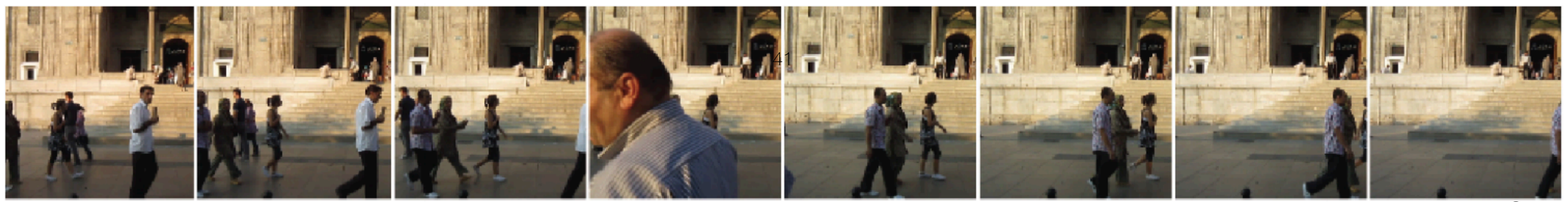
W

time

Source: Torralba, Freeman, Isola

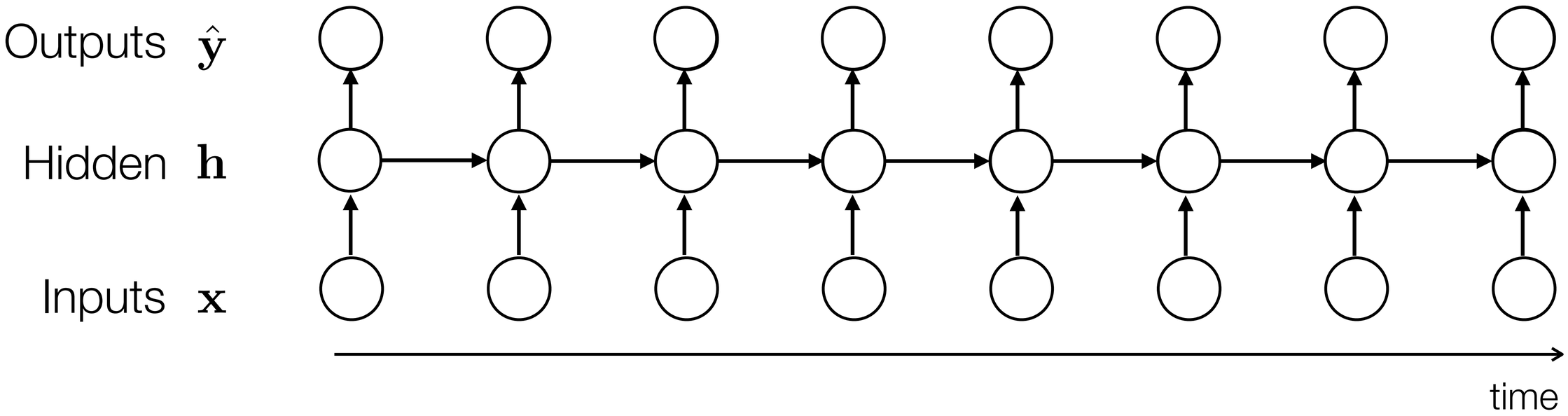
Recurrent Neural Networks (RNNs)





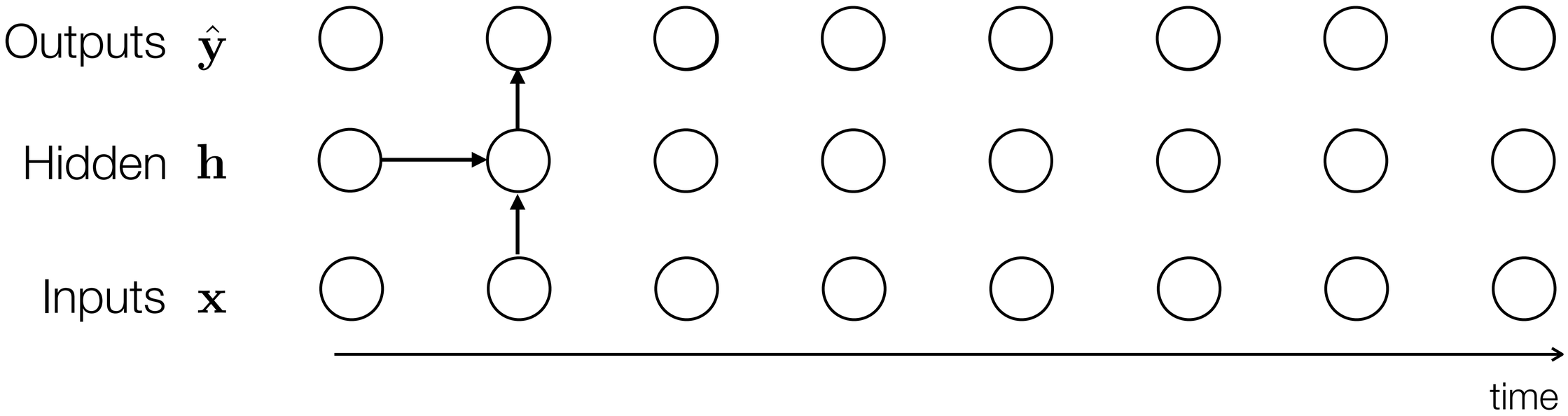
Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)



Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)

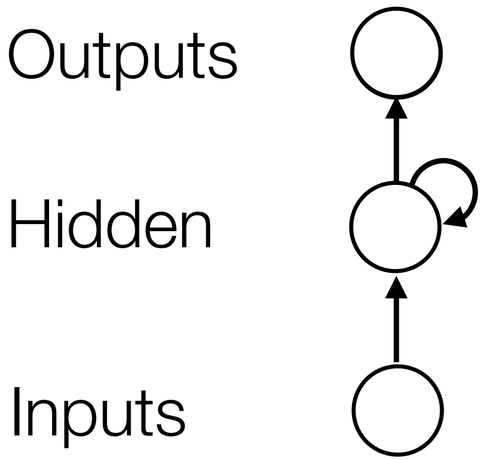


$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$

$$\mathbf{y}^{(t)} = g(\mathbf{h}^{(t)})$$

Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)



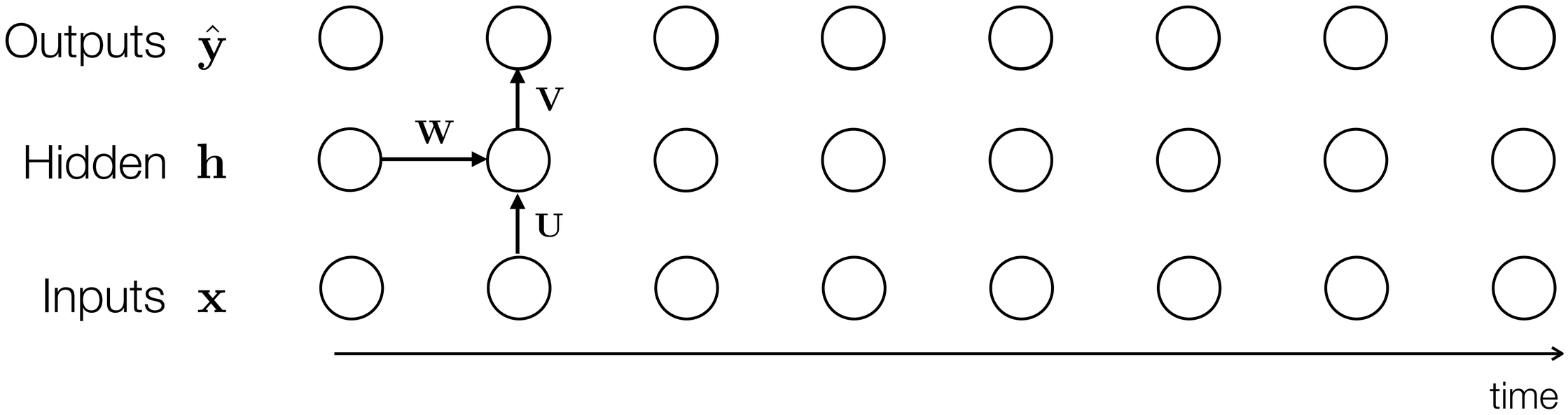
Recurrent!

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

$$\mathbf{y}^{(t)} = g(\mathbf{h}^{(t)})$$

Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)



$$\mathbf{a}^{(t)} = \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b}$$

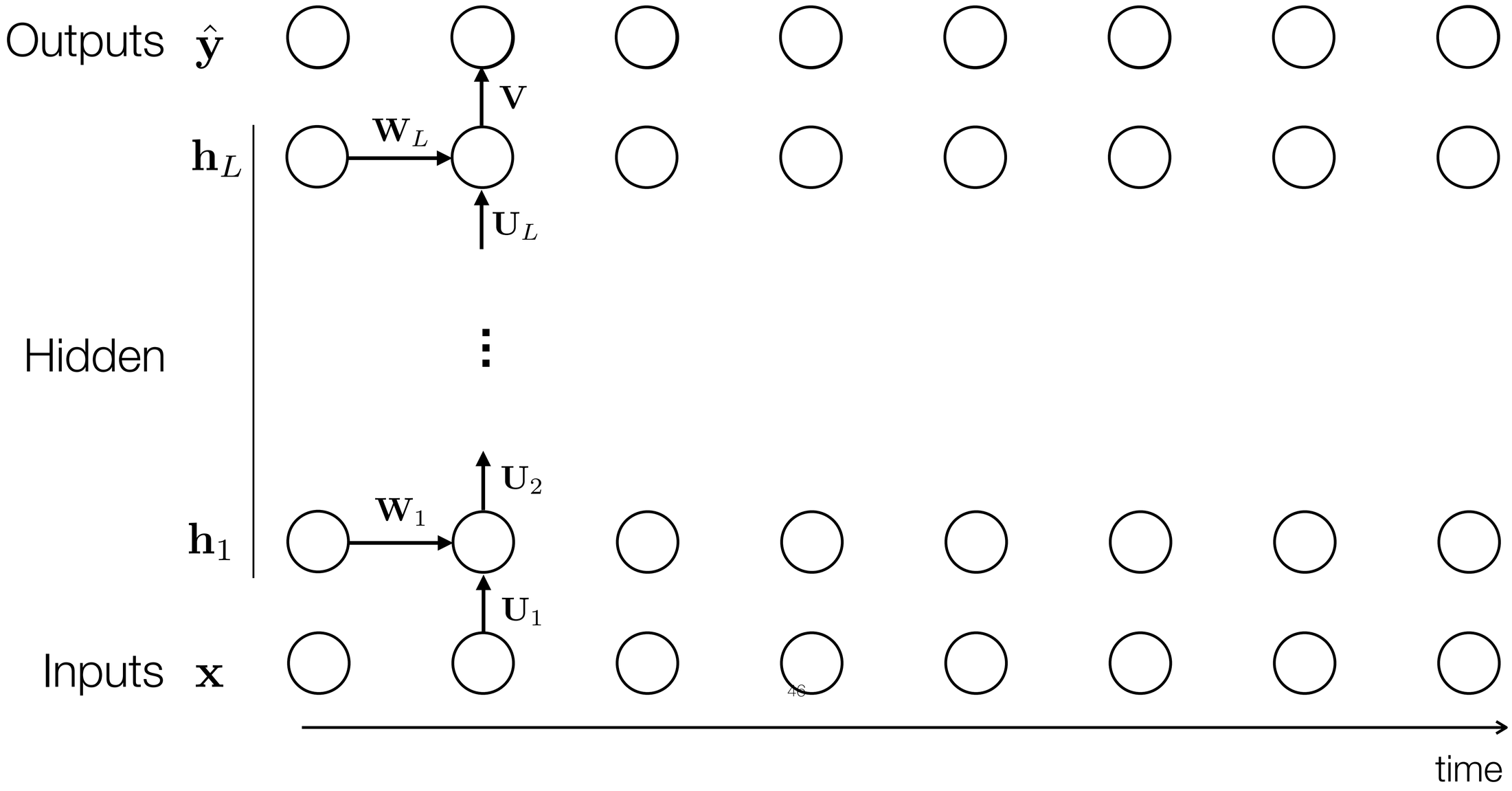
$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)} + \mathbf{c}$$

$$\hat{\mathbf{y}}^{(t)} = \mathtt{softmax}(\mathbf{o}^{(t)})$$

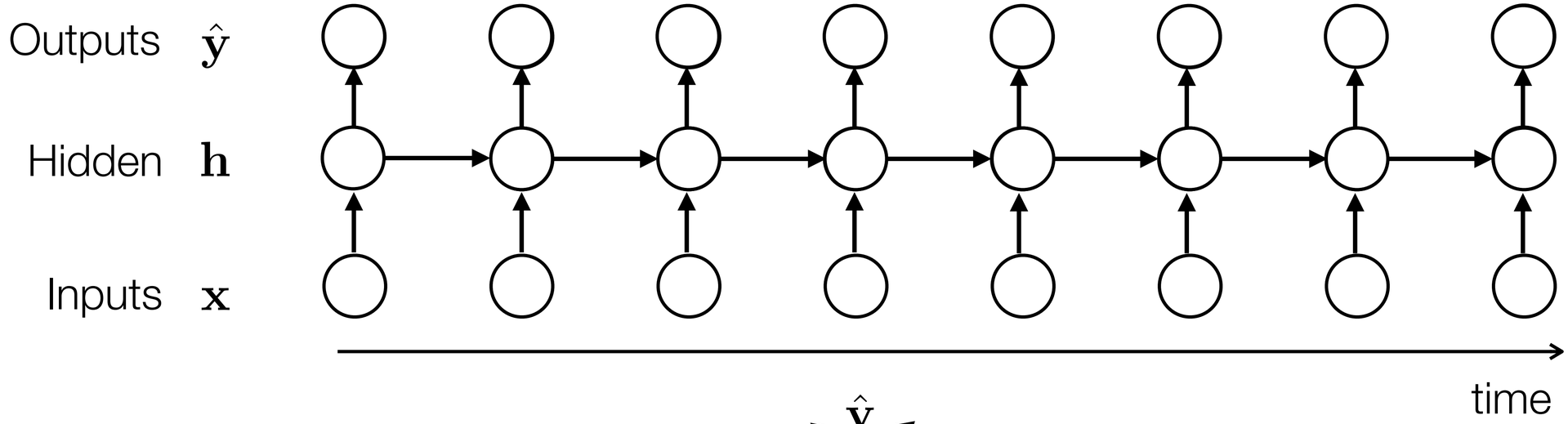
Source: Torralba, Freeman, Isola

Deep Recurrent Neural Networks (RNNs)

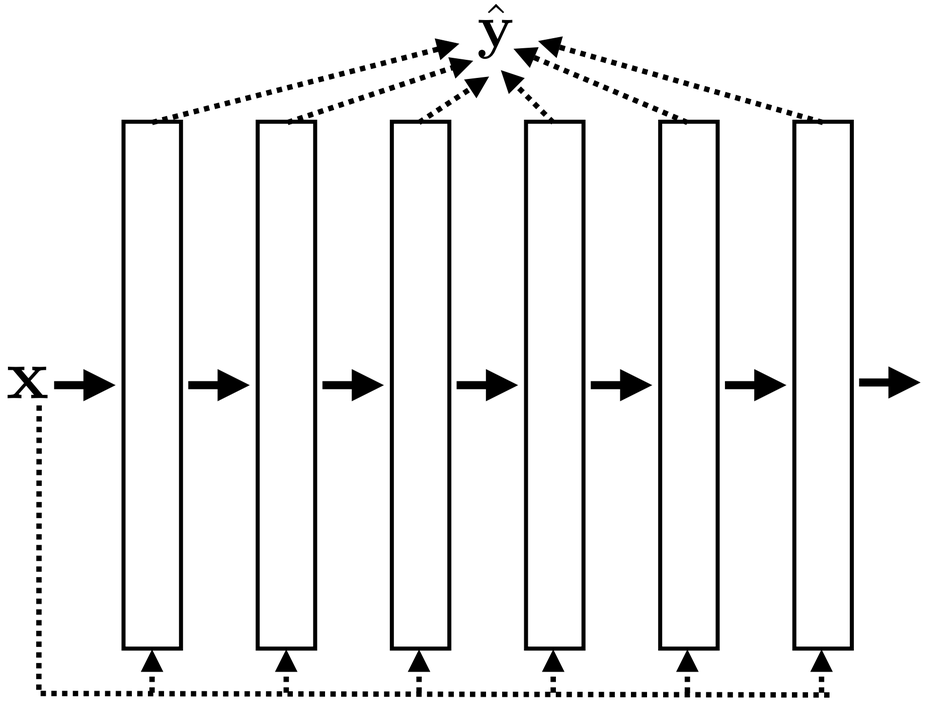


Source: Torralba, Freeman, Isola

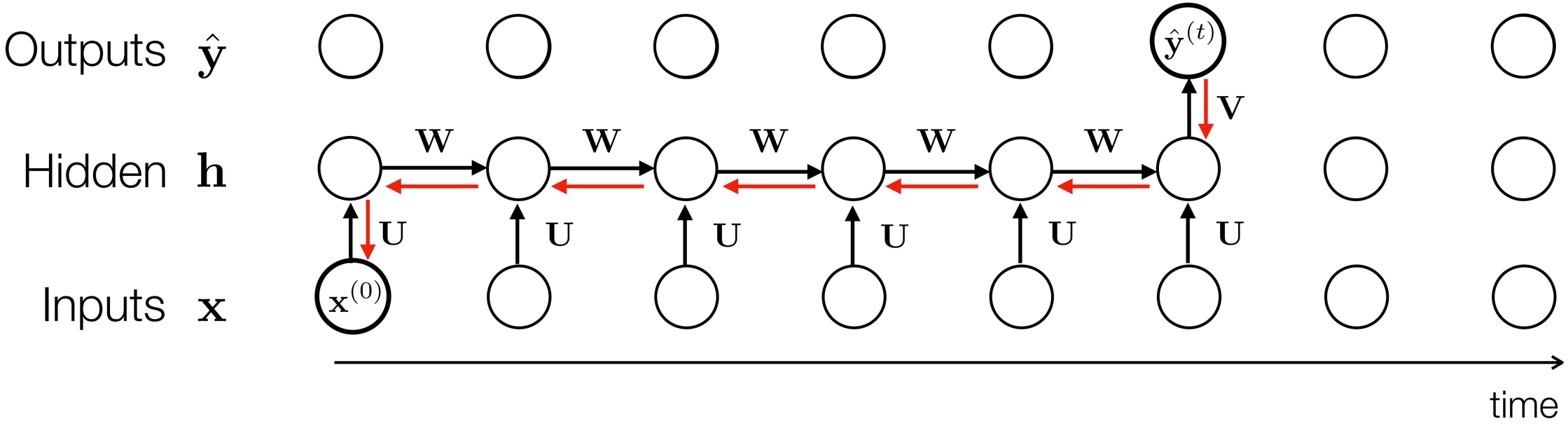
Unrolling an RNN



Equivalent to
"unrolled" network
with shared weights



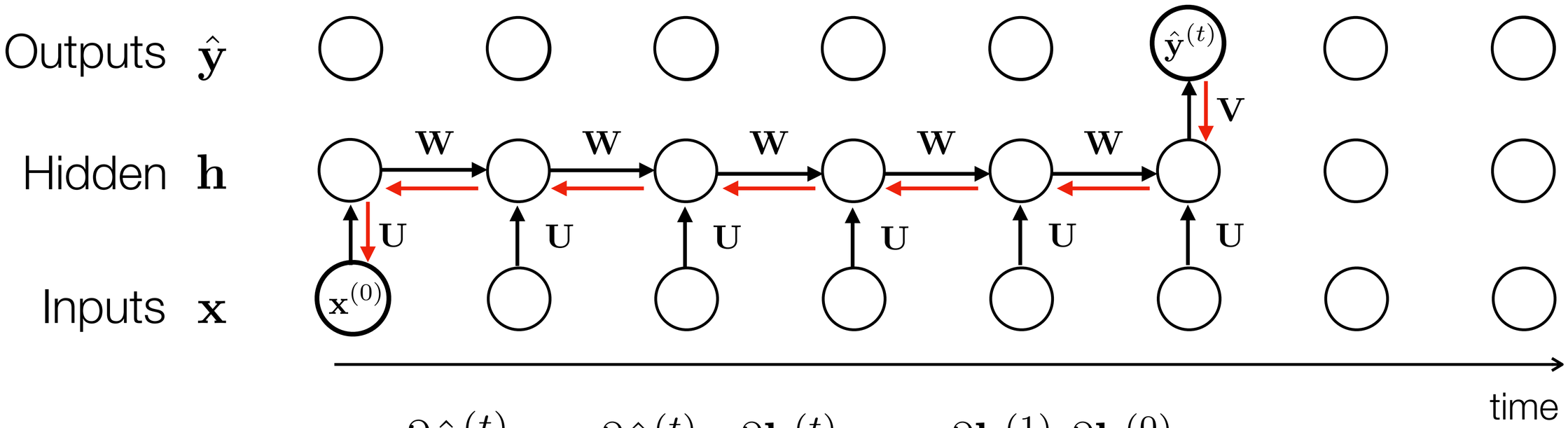
Backpropagation in RNNs



$$\frac{\partial \hat{\mathbf{y}}^{(t)}}{\partial \mathbf{x}^{(0)}} = \frac{\partial \hat{\mathbf{y}}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \cdots \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{h}^{(0)}} \frac{\partial \mathbf{h}^{(0)}}{\partial \mathbf{x}^{(0)}}$$

Source: Torralba, Freeman, Isola

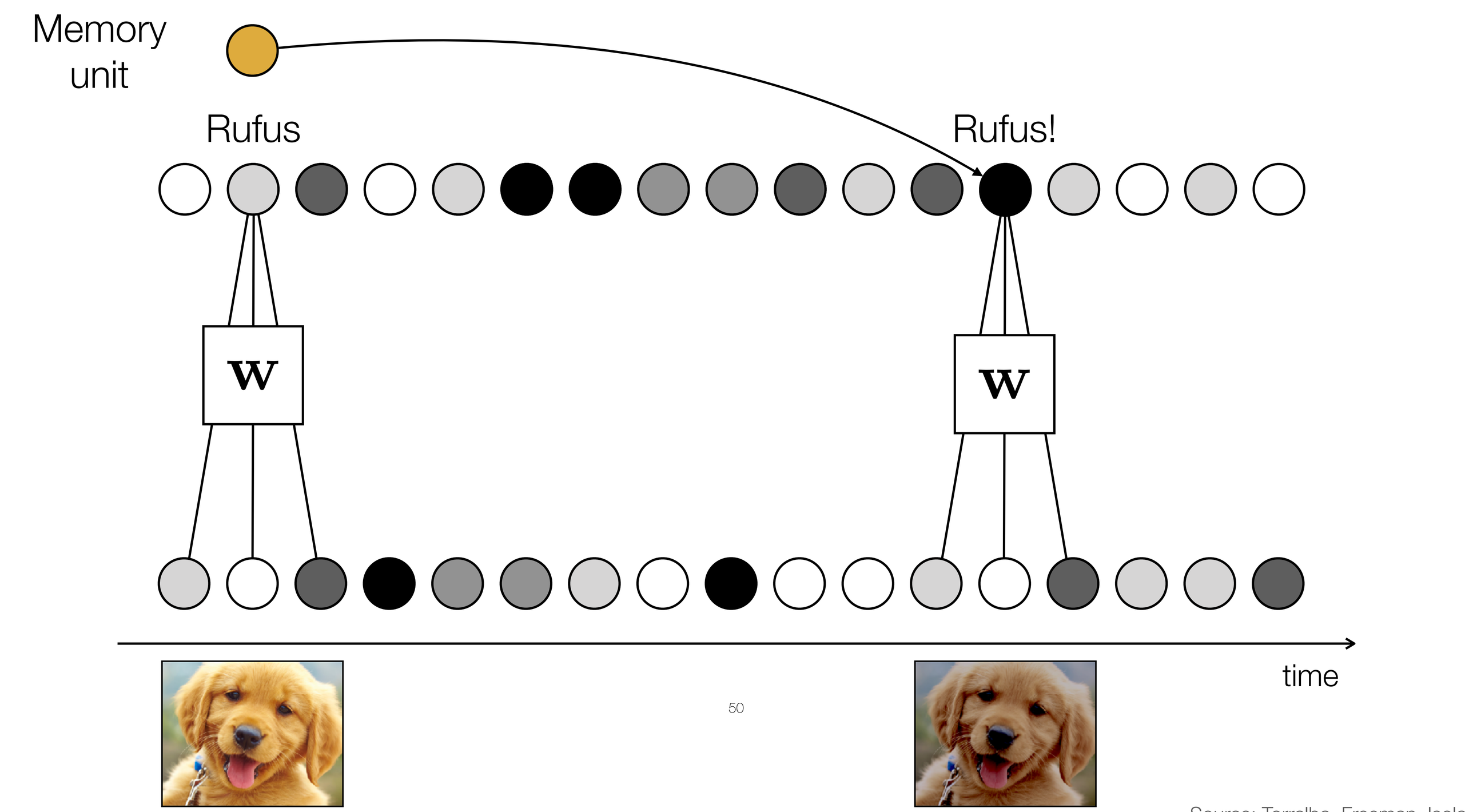
The problem of long-range dependences



$$\frac{\partial \hat{\mathbf{y}}^{(t)}}{\partial \mathbf{x}^{(0)}} = \frac{\partial \hat{\mathbf{y}}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \cdots \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{h}^{(0)}} \frac{\partial \mathbf{h}^{(0)}}{\partial \mathbf{x}^{(0)}}$$

- 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**

Source: Torralba, Freeman, Isola



Source: Torralba, Freeman, Isola

LSTMs

Long Short Term Memory

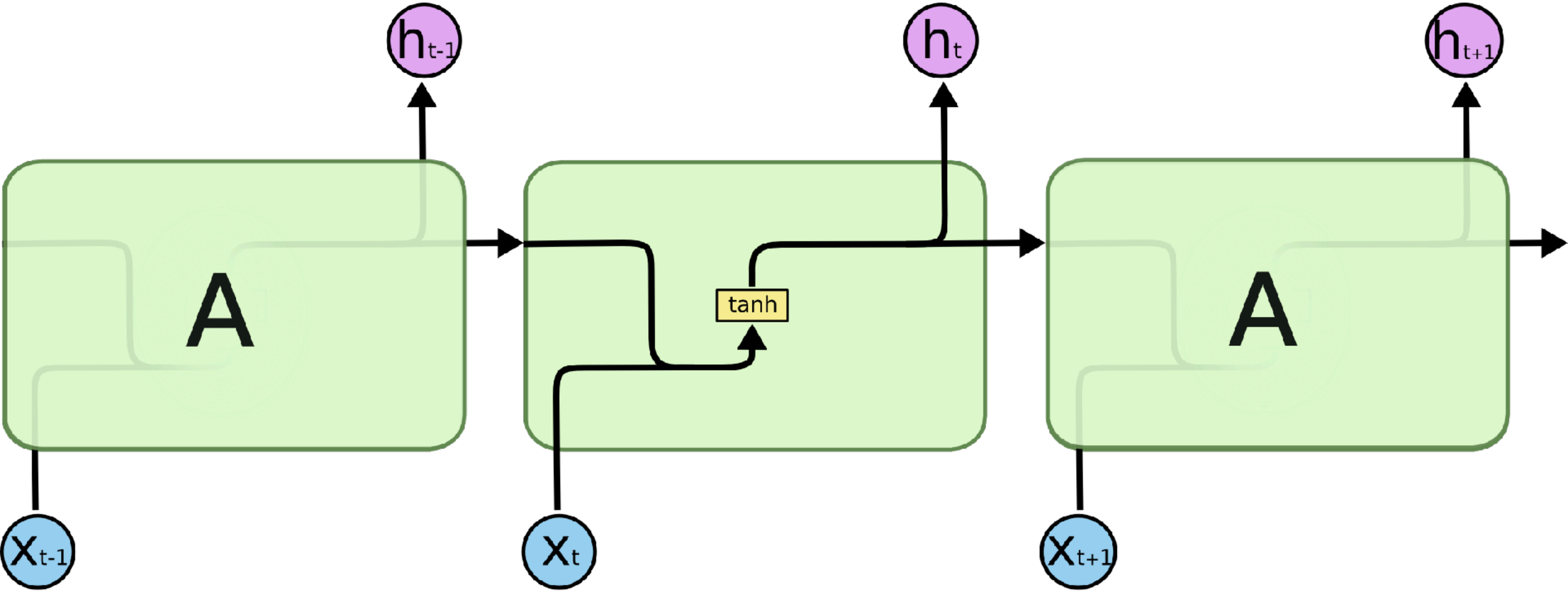
A special kind of RNN designed to avoid forgetting.

Related to ResNets' bias is that state transition is an identity function.

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

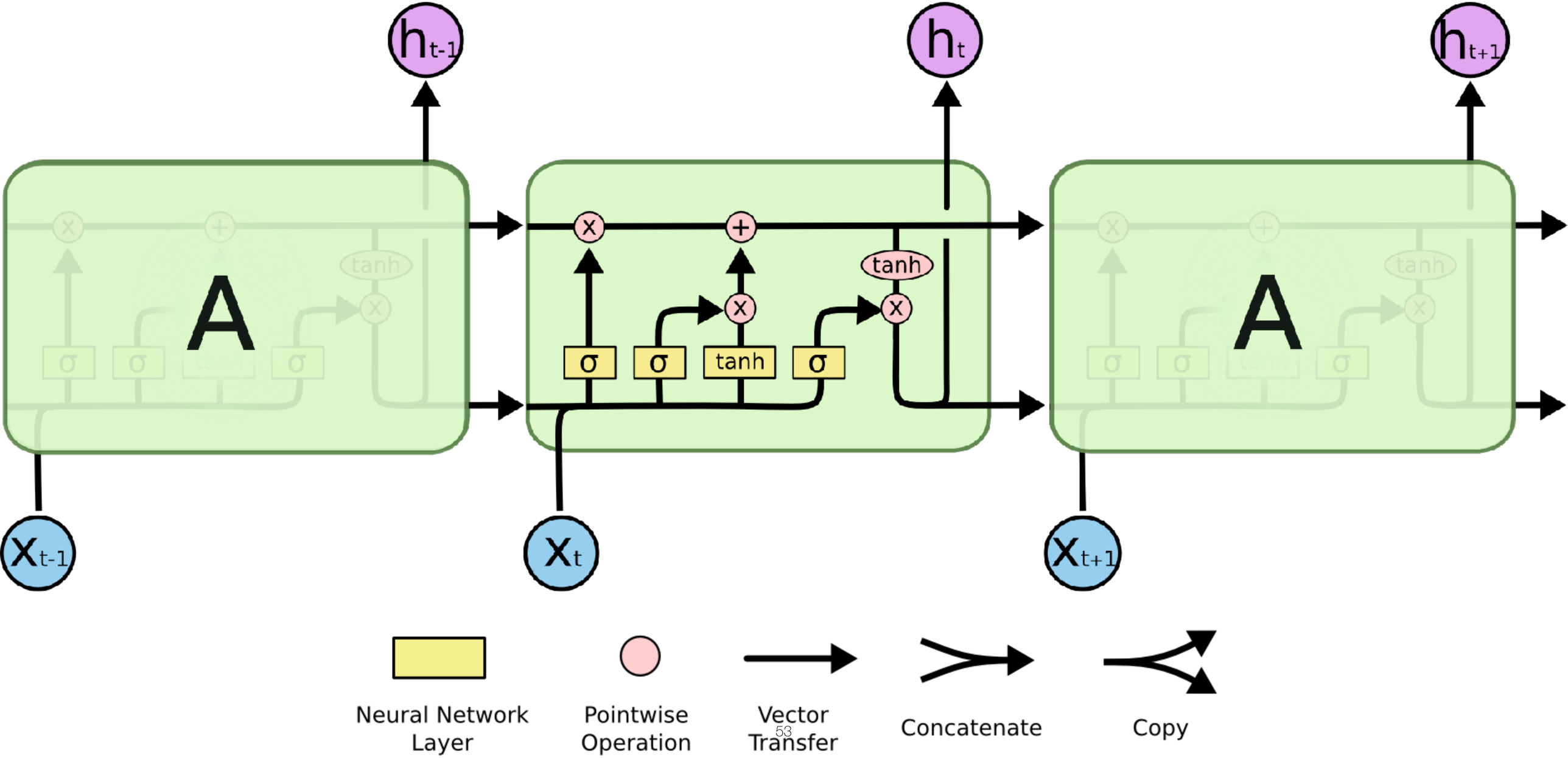
Bit of a complex design. Works well but simpler methods like Gated Recurrent Unit (GRU) are competitive [Jozefowicz[51] et al. 2015].

Source: Torralba, Freeman, Isola



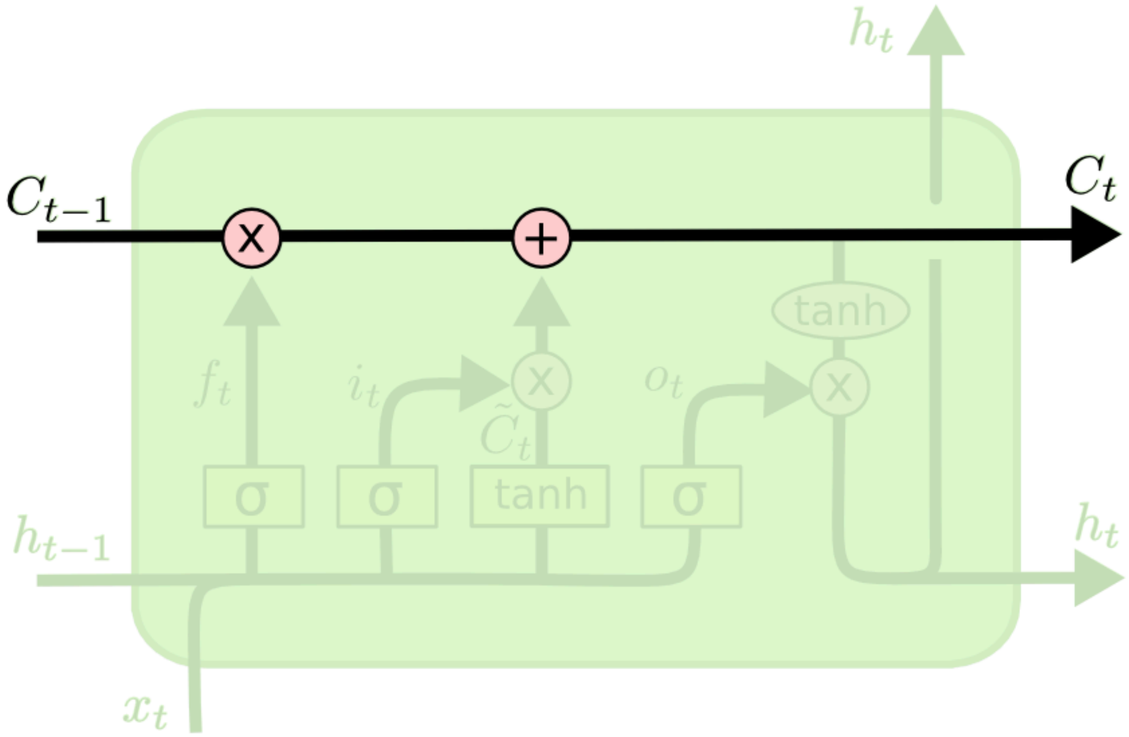
[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]

Source: Torralba, Freeman, Isola



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

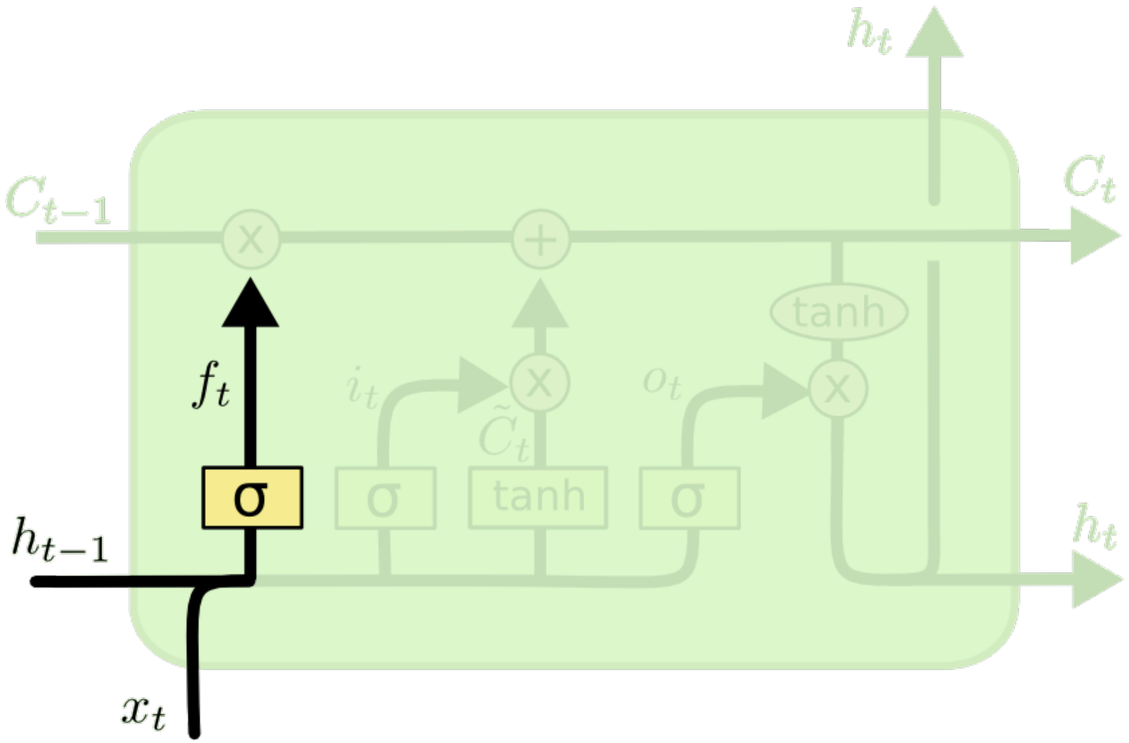
Source: Torralba, Freeman, Isola

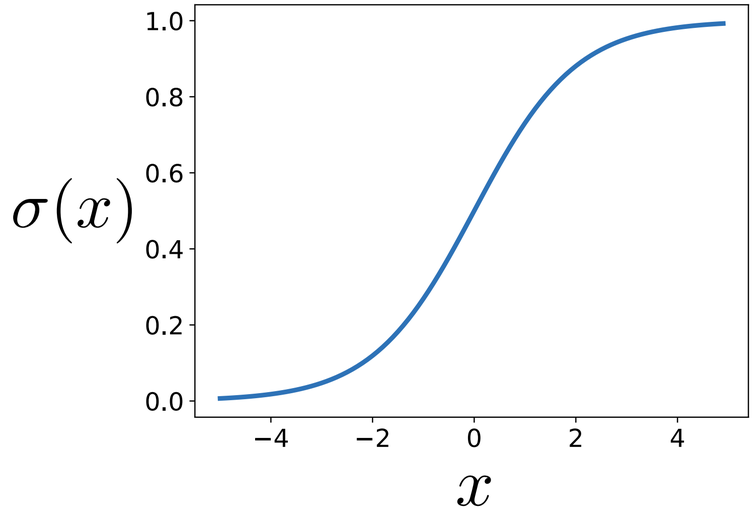


C_t = Cell state

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

Source: Torralba, Freeman, Isola





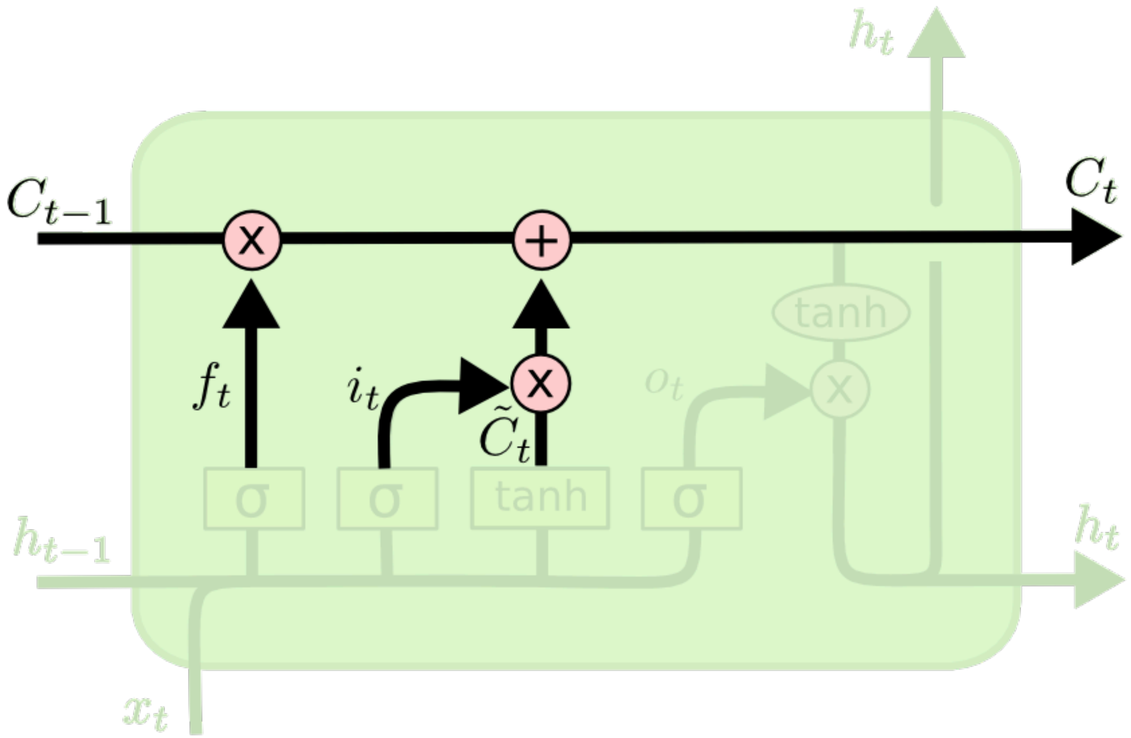
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Decide what information to throw away from the cell state.

Forget gate: 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/]

Source: Torralba, Freeman, Isola



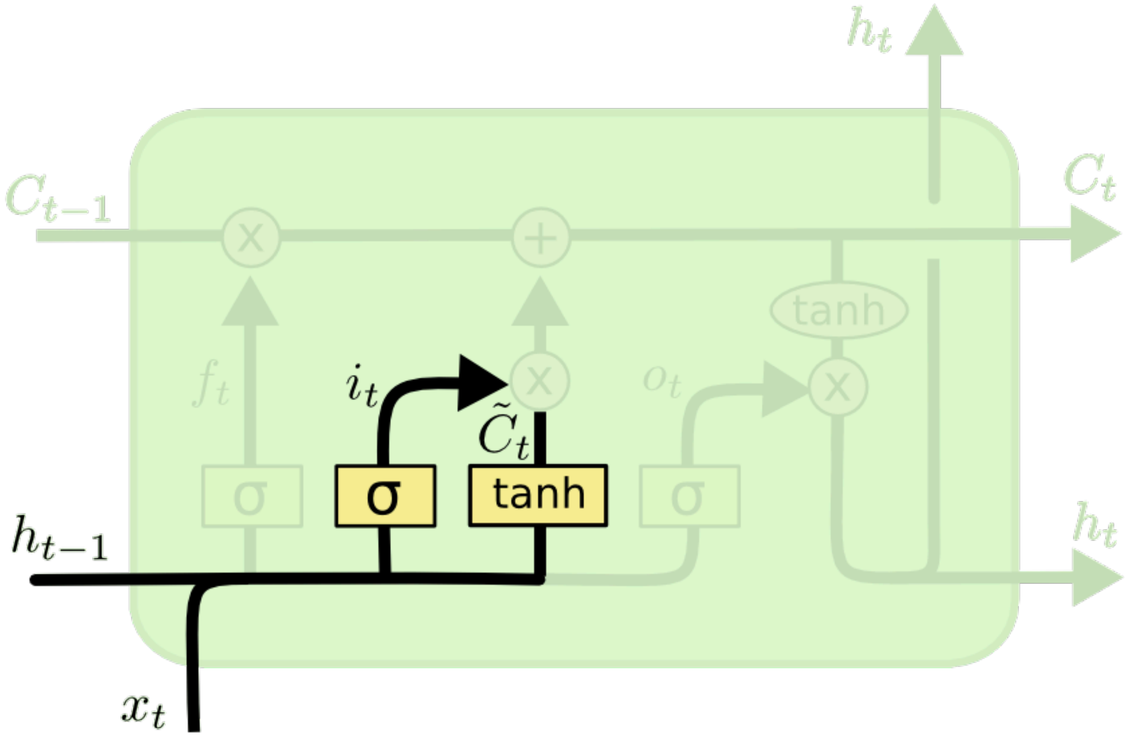
Forget New values to write

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

Forget selected old information, write selected new information.

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

Source: Torralba, Freeman, Isola



which indices to write to

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

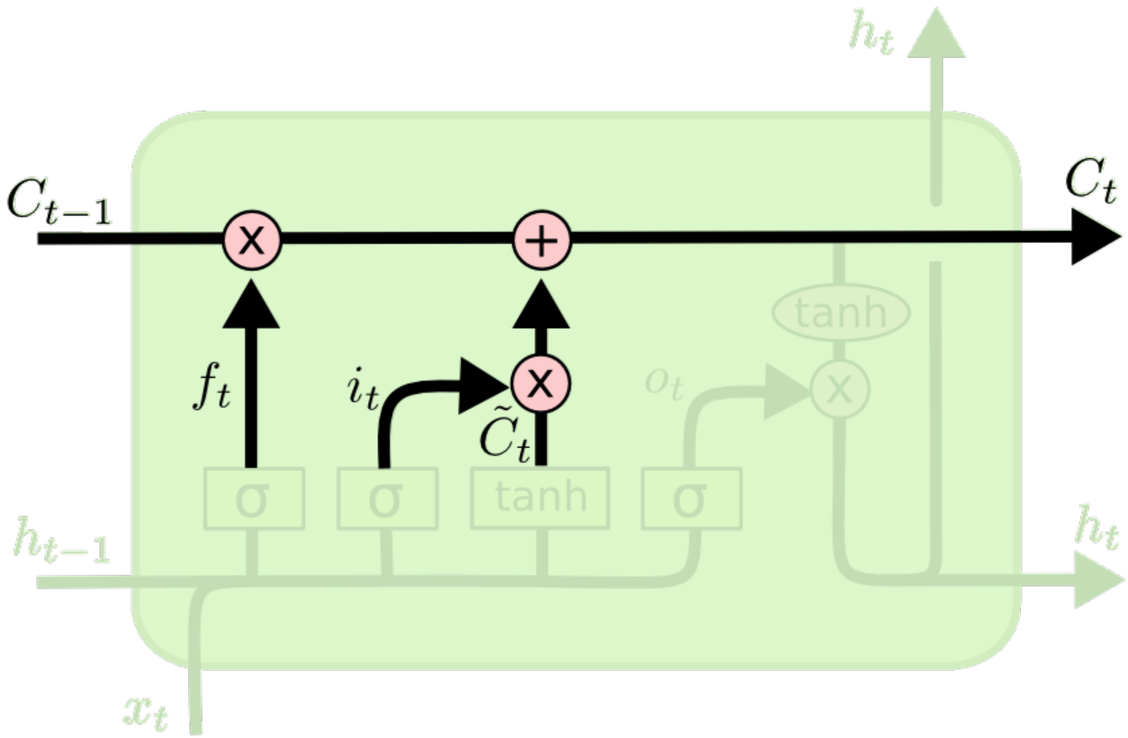
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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/]

Source: Torralba, Freeman, Isola



Forget

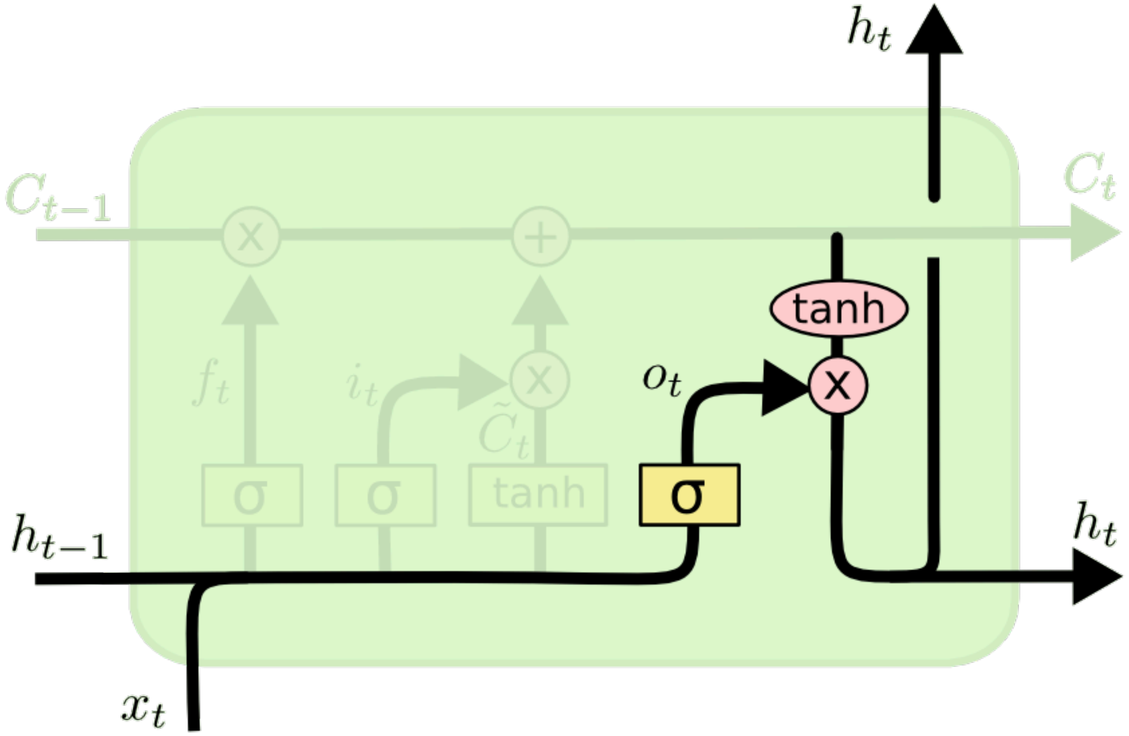
Write new values

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

Forget selected old information, write selected new information.

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

Source: Torralba, Freeman, Isola



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

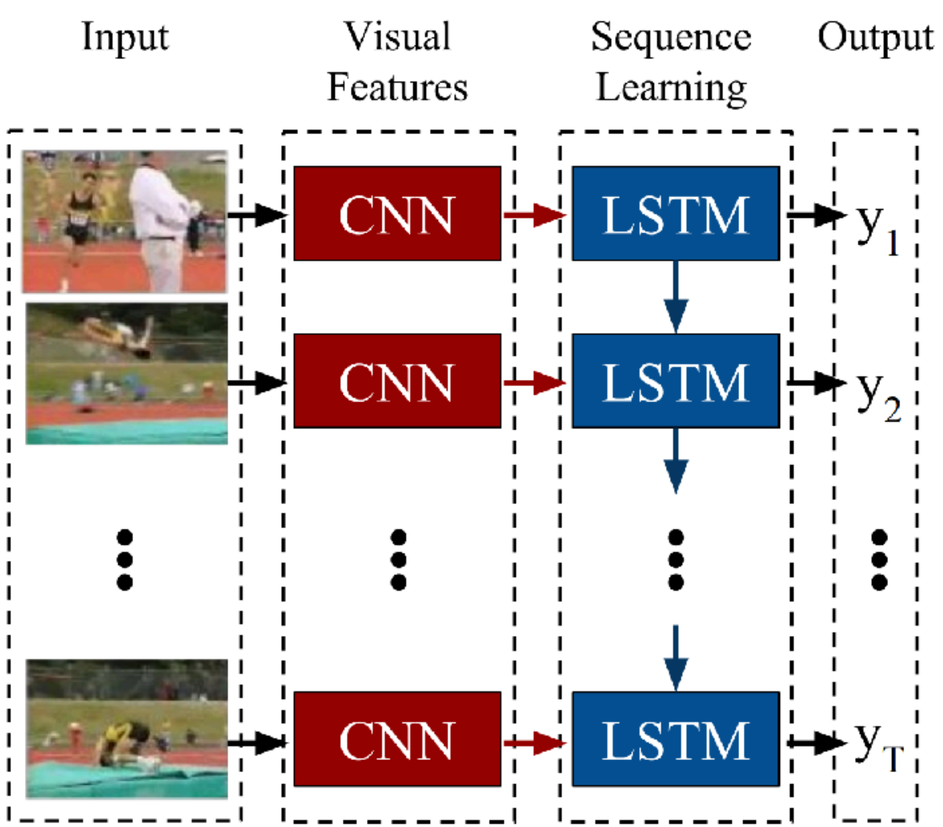
$$h_t = o_t * \tanh\left(C_t\right)$$

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/]

Source: Torralba, Freeman, Isola

Some uses for RNNs



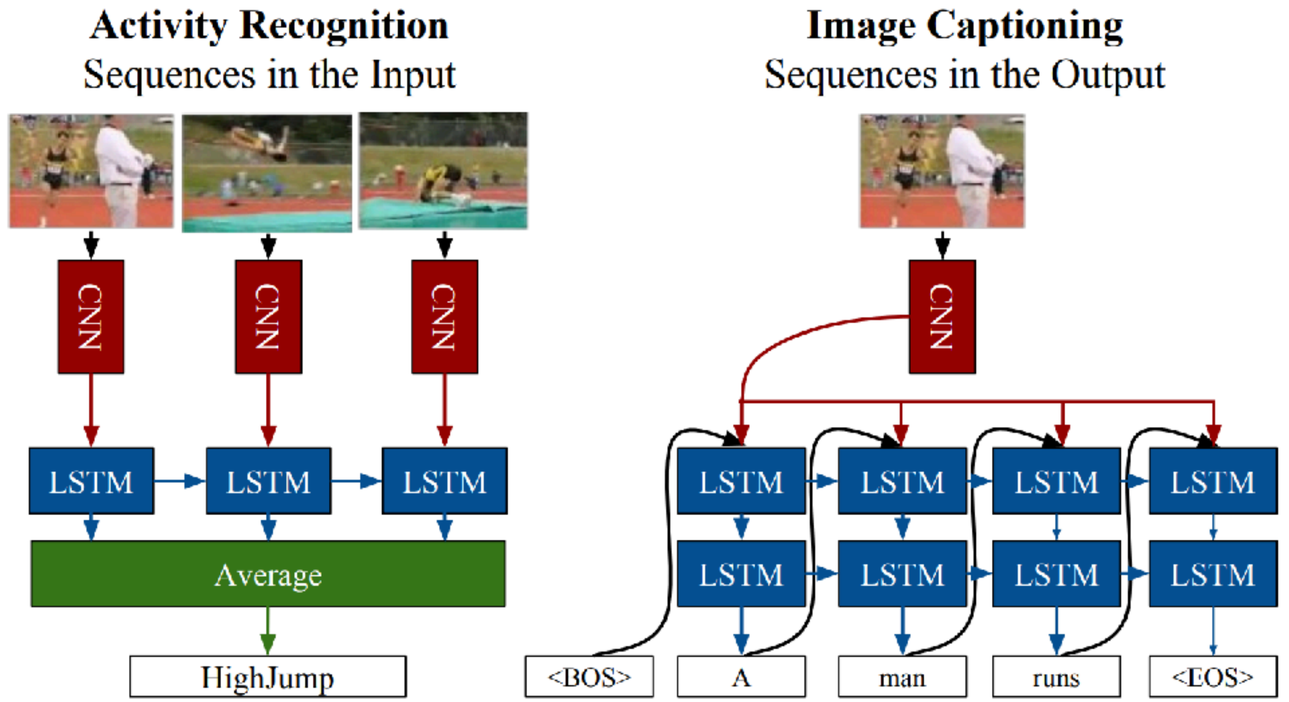


Figure from [Donahue et al. 2016]

Problems with RNNs

- Hard to obtain motion information.
- Results in very deep networks (often slow, hard to train)
- Doesn’t parallelize well
- We’ll see alternative approaches, like **attention** later in class