Lecture 12: Learning temporal models

Announcements

- PS1 regrade requests due today!
- PS2 grades out. Regrades due next Tues.
- PS6 out.
- Project proposal will be next “assignment”. We’ll give you more info next week

Project

- Open-ended! Example projects:
  - Implement and extend a recent computer vision paper
  - Use computer vision in your research
  - We'll also provide a list of project ideas

- Work in small groups (up to 4 people)

- We'll allow updates to project proposal if you change ideas

Source: Torralba, Freeman, Isola

Source: Torralba, Freeman, Isola

Source: Torralba, Freeman, Isola

Source: Torralba, Freeman, Isola

Source: Torralba, Freeman, Isola

“What will the girl do next?”

Source: Torralba, Freeman, Isola

Event prediction

What can happen here?

Source: Torralba, Freeman, Isola

Event prediction

What can happen here?

Video database

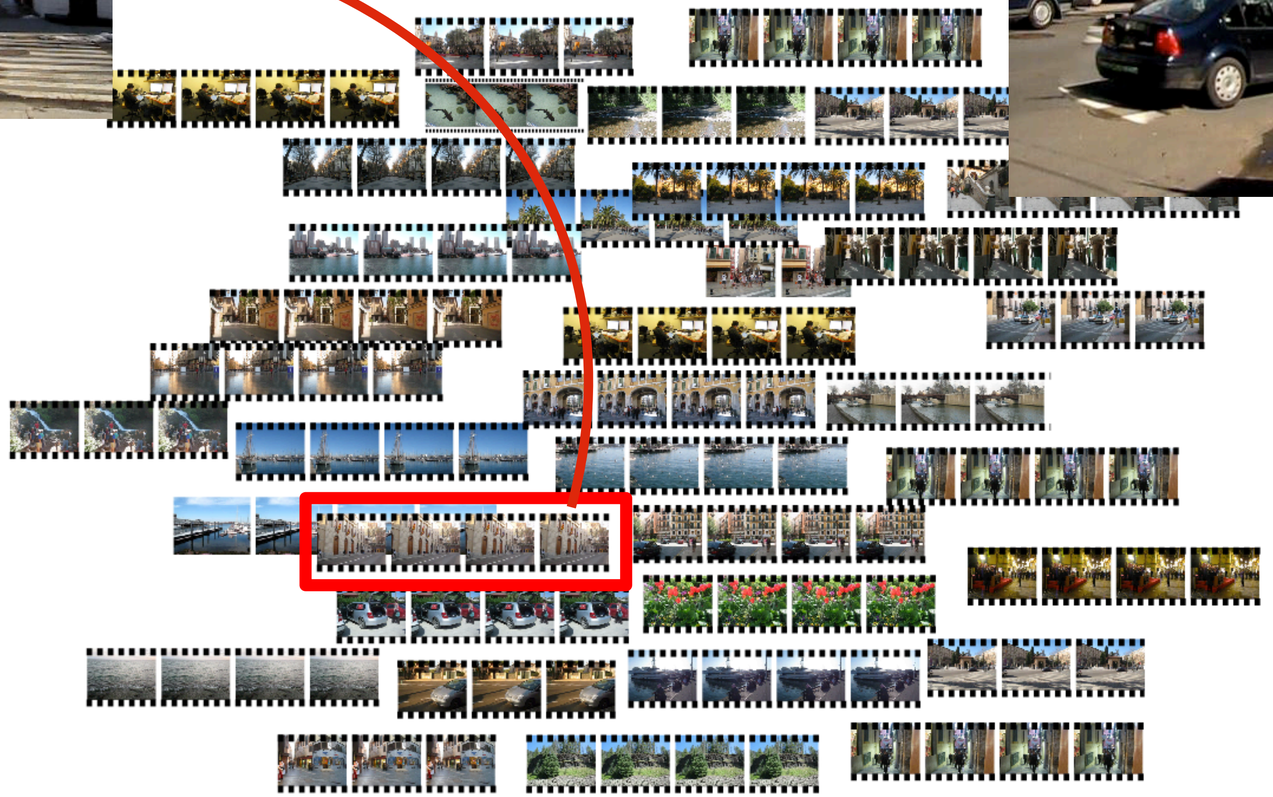
Source: Torralba, Freeman, Isola

Liu, Yuen, Torralba. CVPR 2009. Yuen, Torralba. ECCV 2010

Event prediction

What can happen here?

Video database



Source: Torralba, Freeman, Isola

Liu, Yuen, Torralba. CVPR 2009; Yuen, Torralba. ECCV 2010

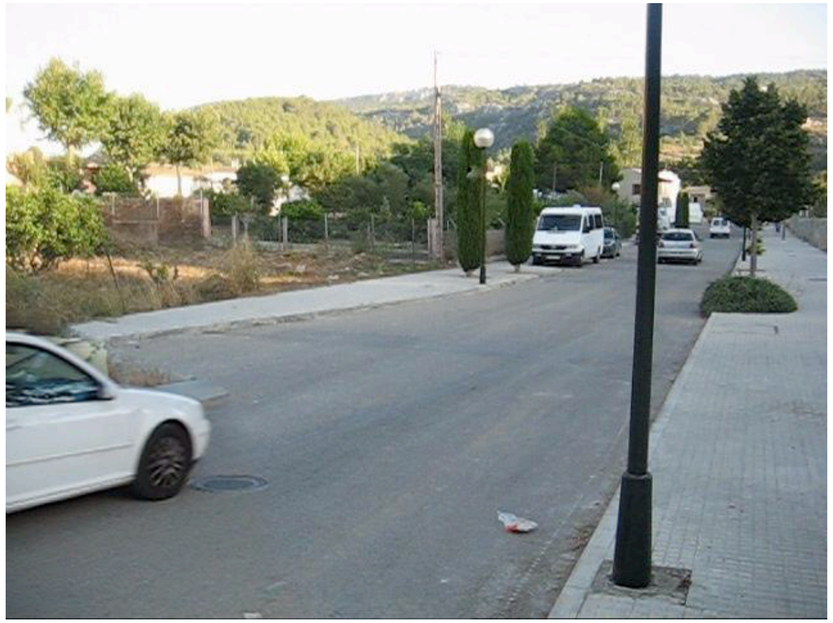
What can happen here?

Source: Torralba, Freeman, Isola

Liu, Yuen, Torralba. CVPR 2009. Yuen, Torralba. ECCV 2010

What can happen here?

Prediction

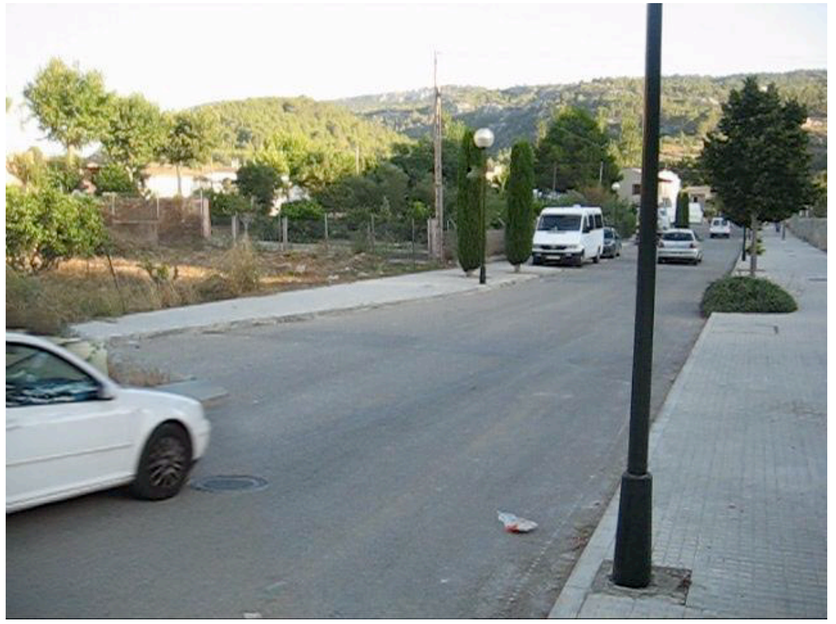


Source: Torralba, Freeman, Isola

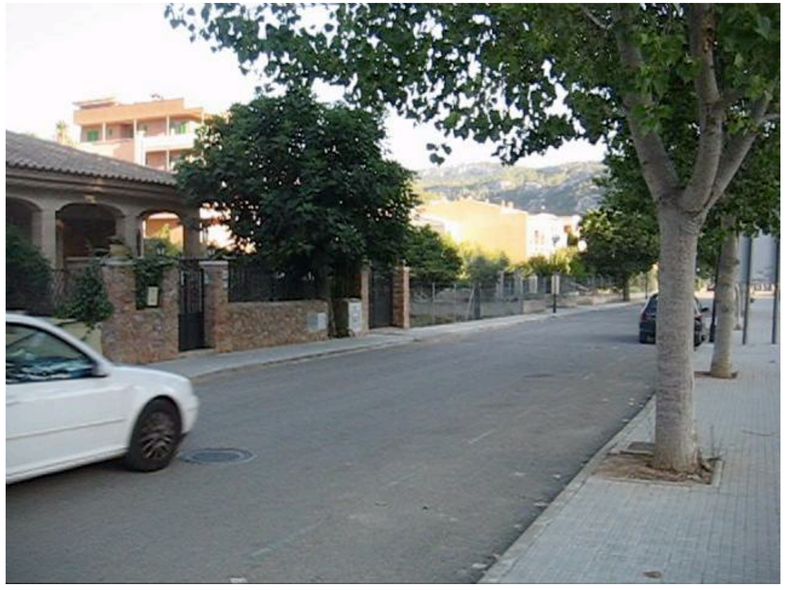
Liu, Yuen, Torralba. CVPR 2009. Yuen, Torralba. ECCV 2010

What can happen here?

Prediction



Nearest neighbor



Source: Torralba, Freeman, Isola

Liu, Yuen, Torralba. CVPR 2009. Yuen, Torralba. ECCV 2010

What can happen here?

Source: Torralba, Freeman, Isola

Liu, Yuen, Torralba. CVPR 2009. Yuen, Torralba. ECCV 2010

What can happen here?

Prediction

Liu, Yuen, Torralba. CVPR 2009. Yuen, Torralba. ECCV 2010

Action recognition

Making latte art

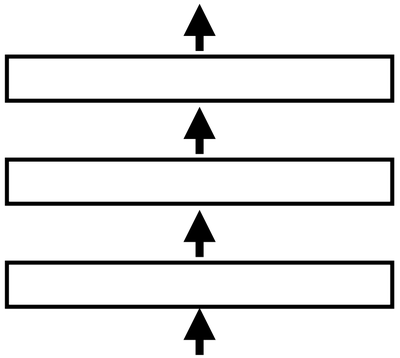
Jaywalking

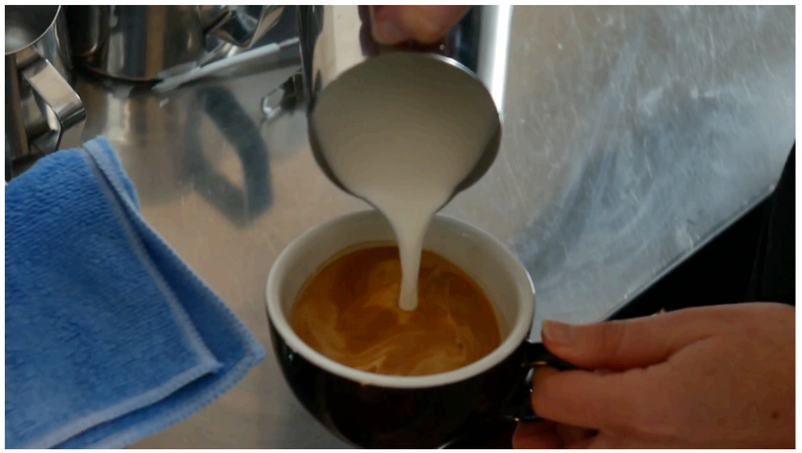
Grooming dog

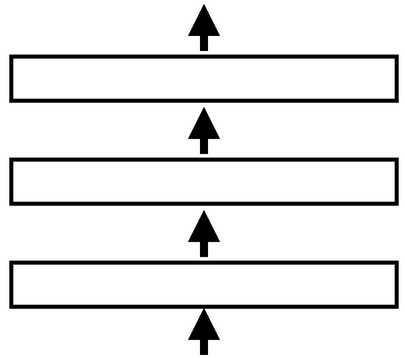
Kinetics dataset [Carreira et al. 2017 - 2019]
700 human activity classes, 650K 10-sec videos

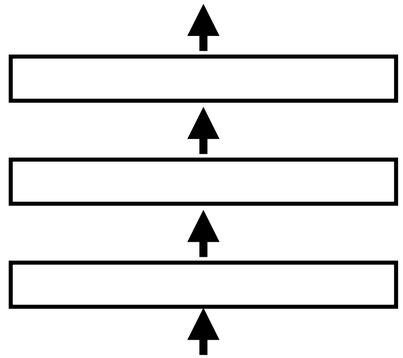
Simple model: averaging

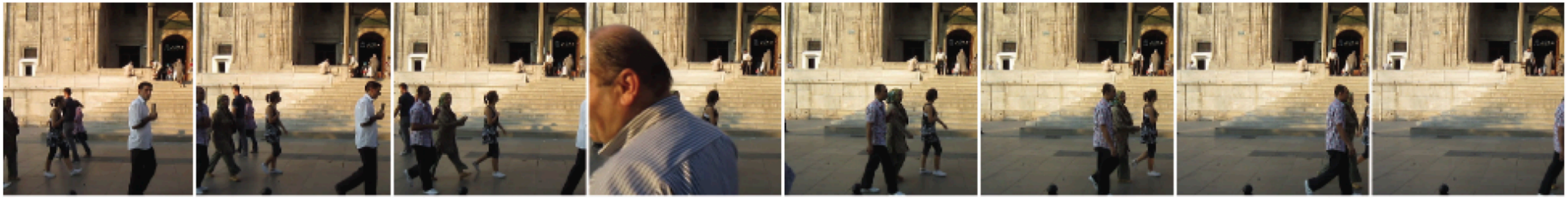
$\frac{1}{3}$ p(making latte art | I_1) + $\frac{1}{3}$ p(making latte art | I_2) + $\frac{1}{3}$ p(making latte art | I_3)



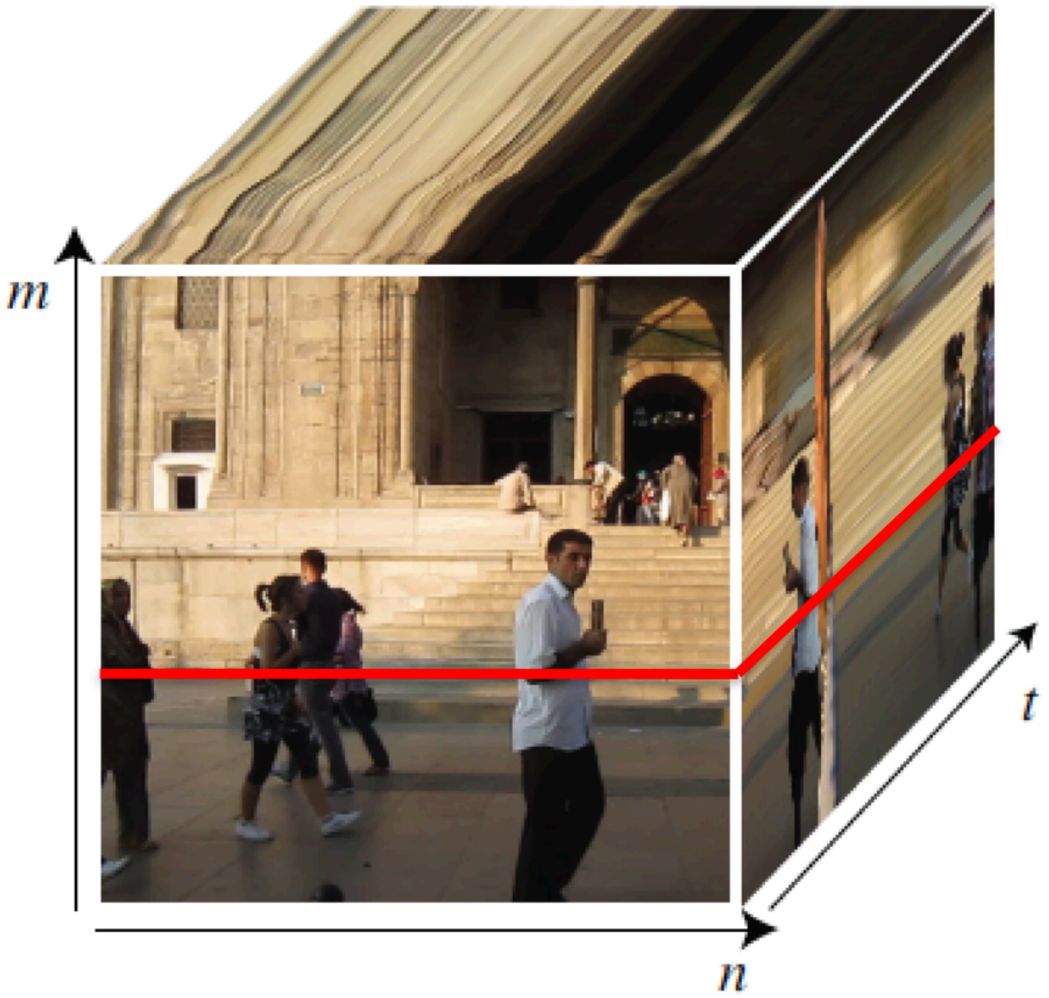


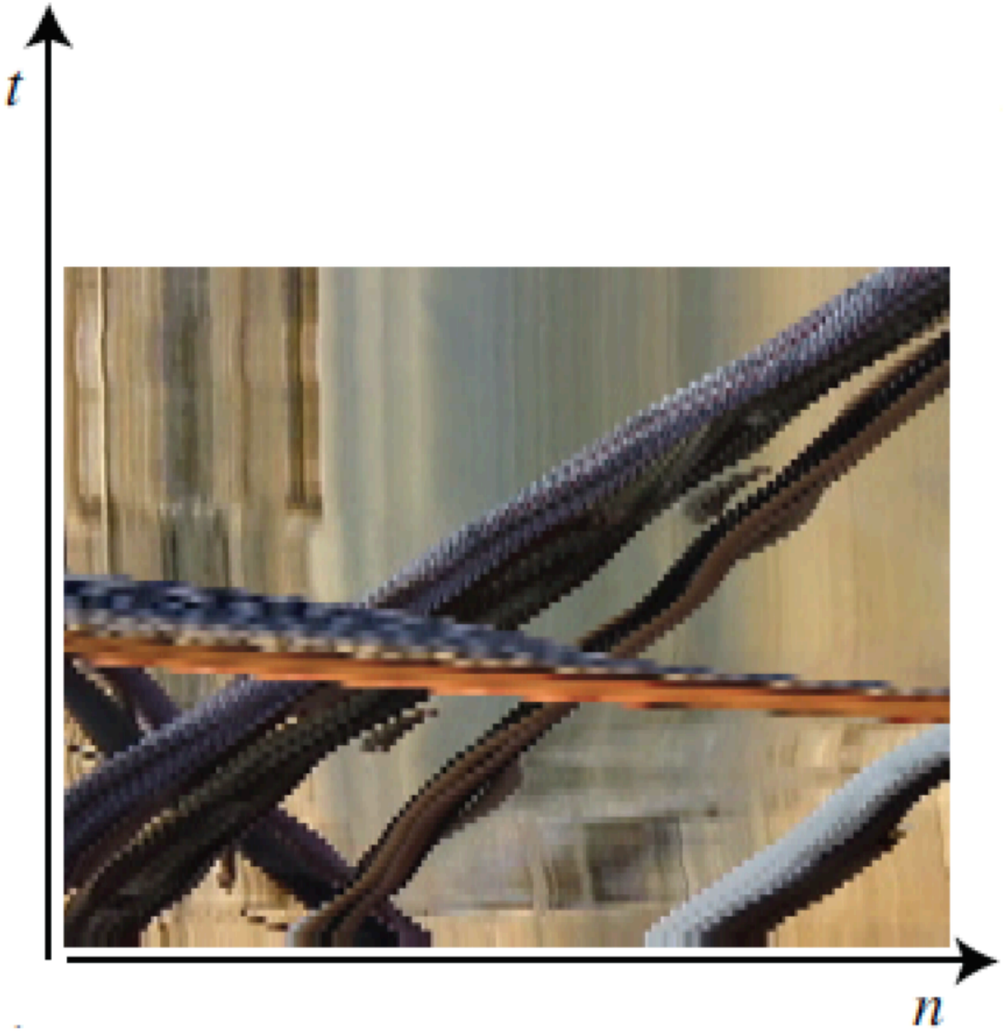




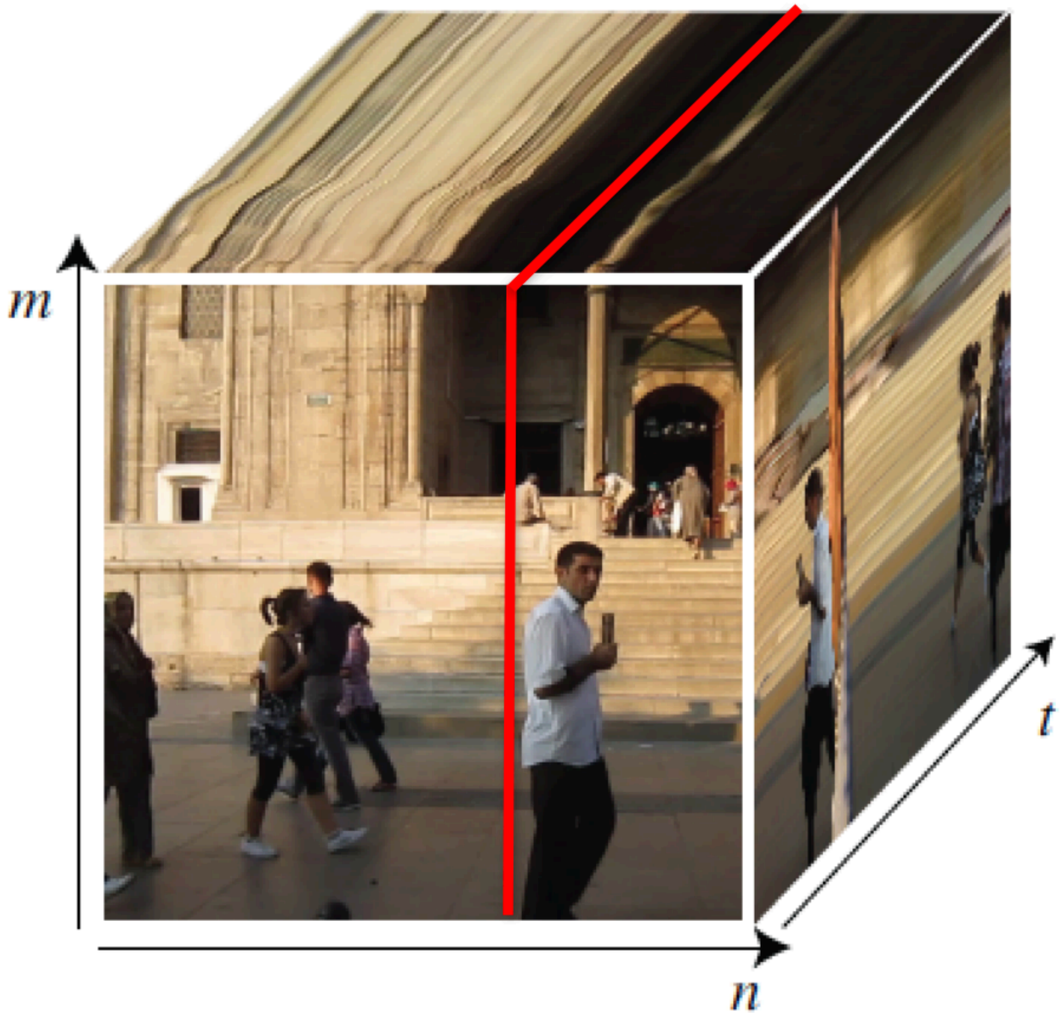


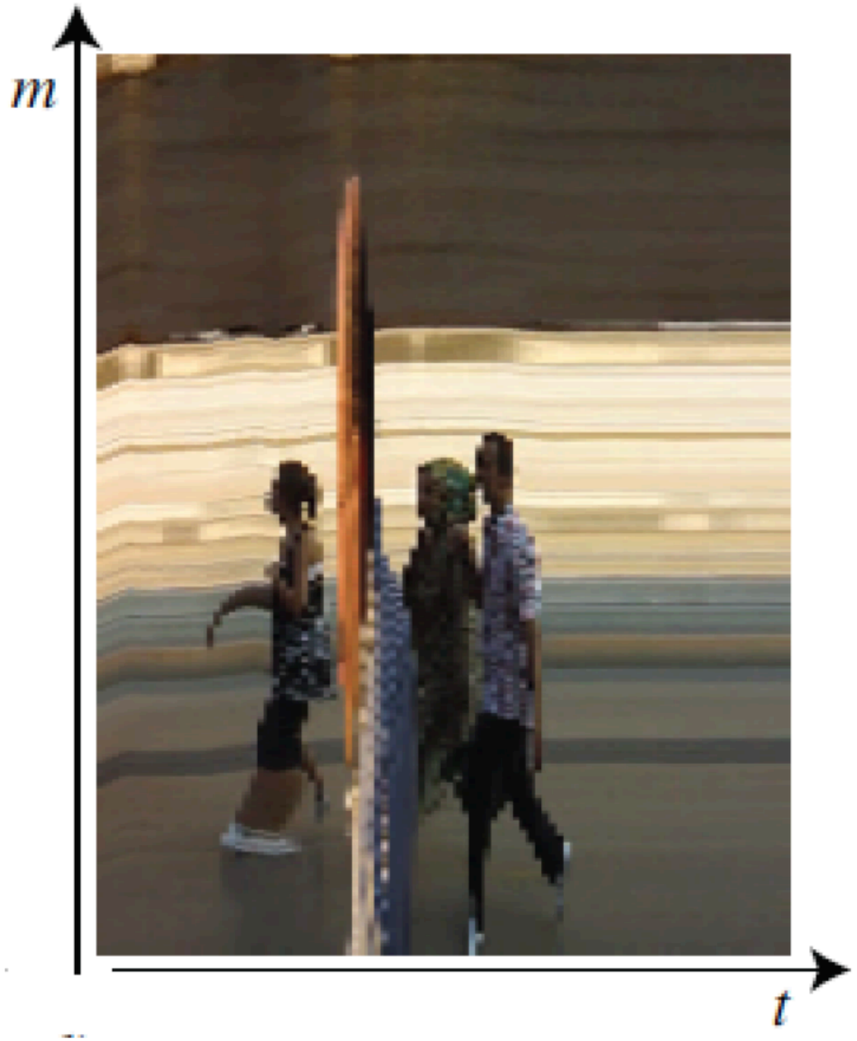
Source: Torralba, Freeman, Isola





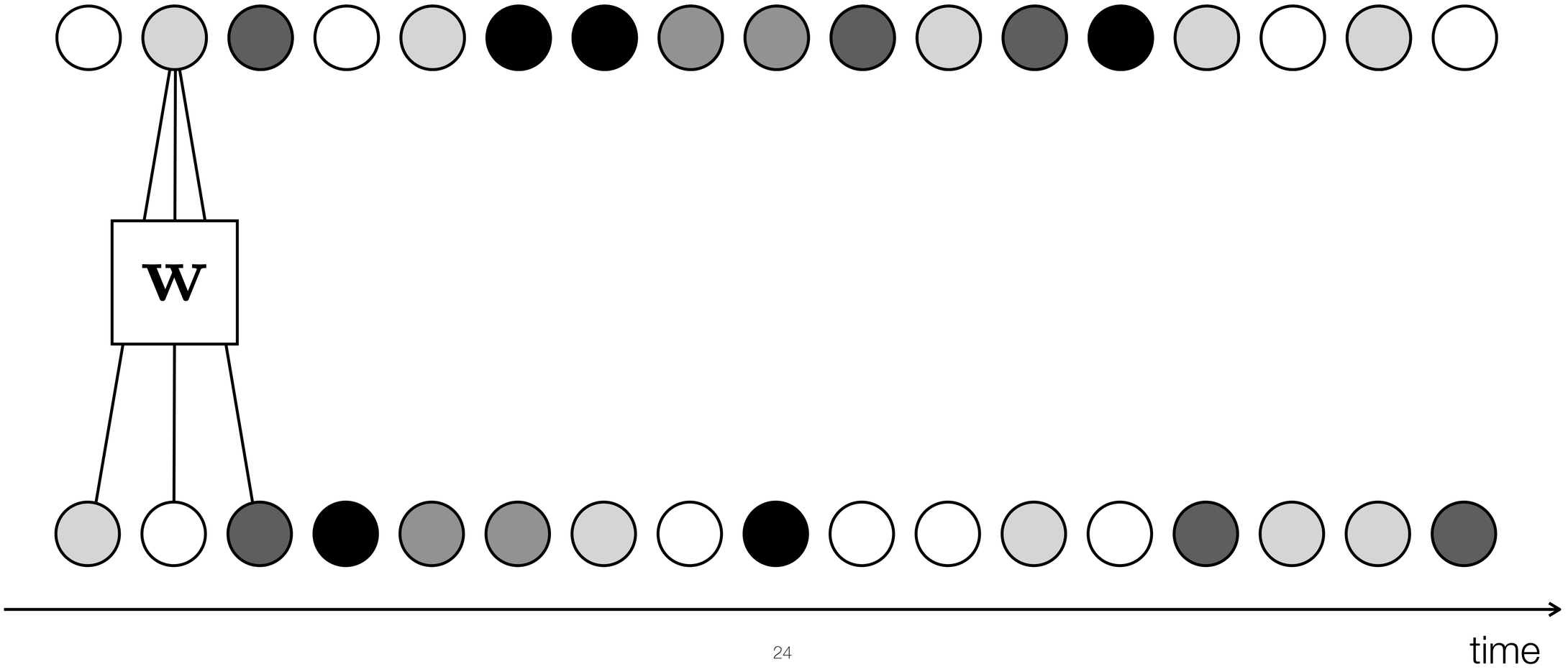
Source: Torralba, Freeman, Isola



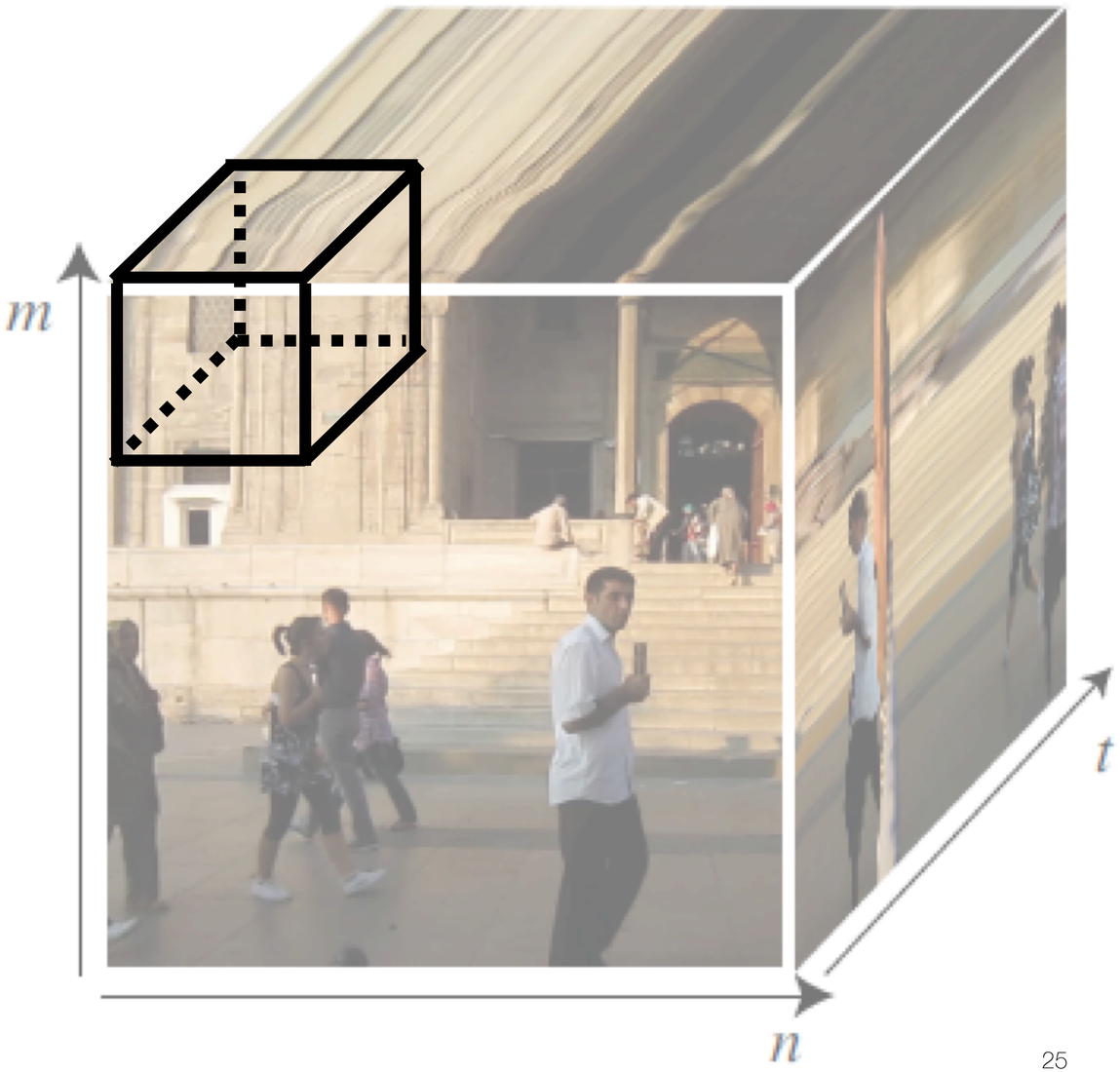


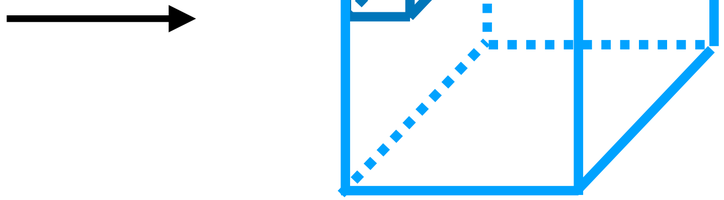
Source: Torralba, Freeman, Isola

Convolutions in time

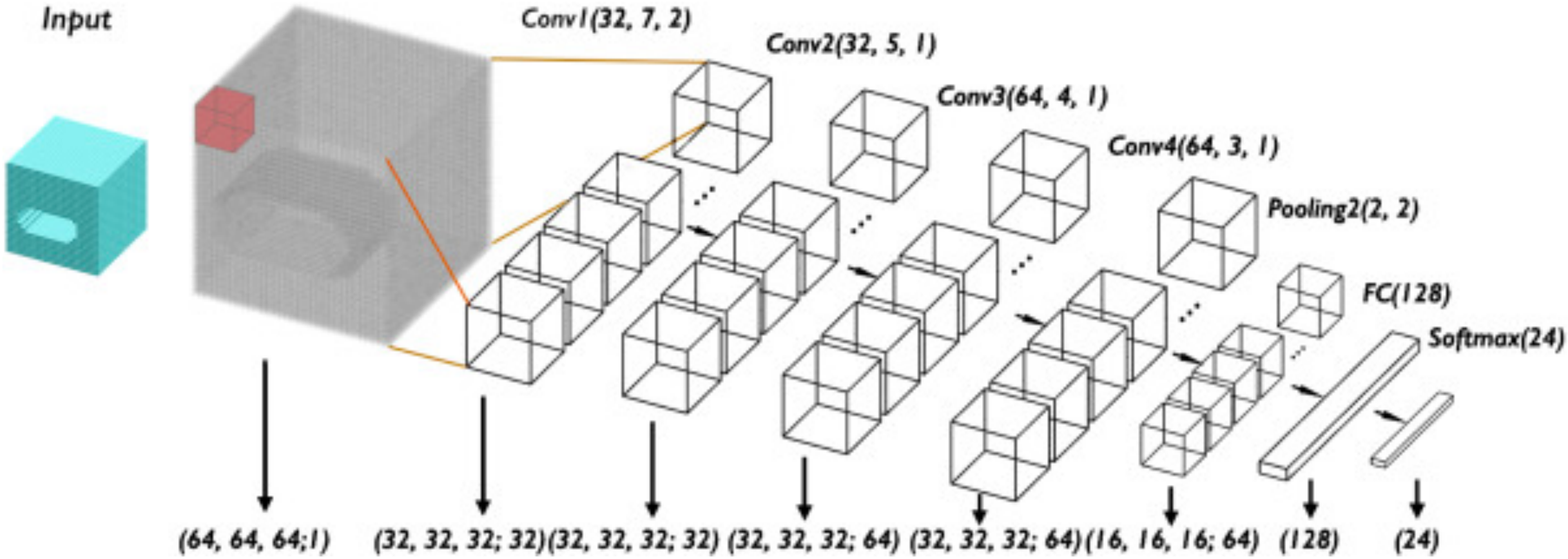


Source: Torralba, Freeman, Isola





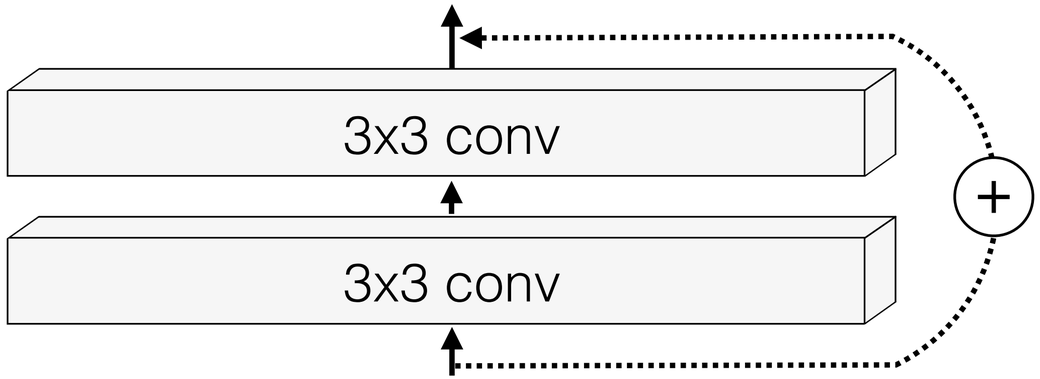
Source: Torralba, Freeman, Isola



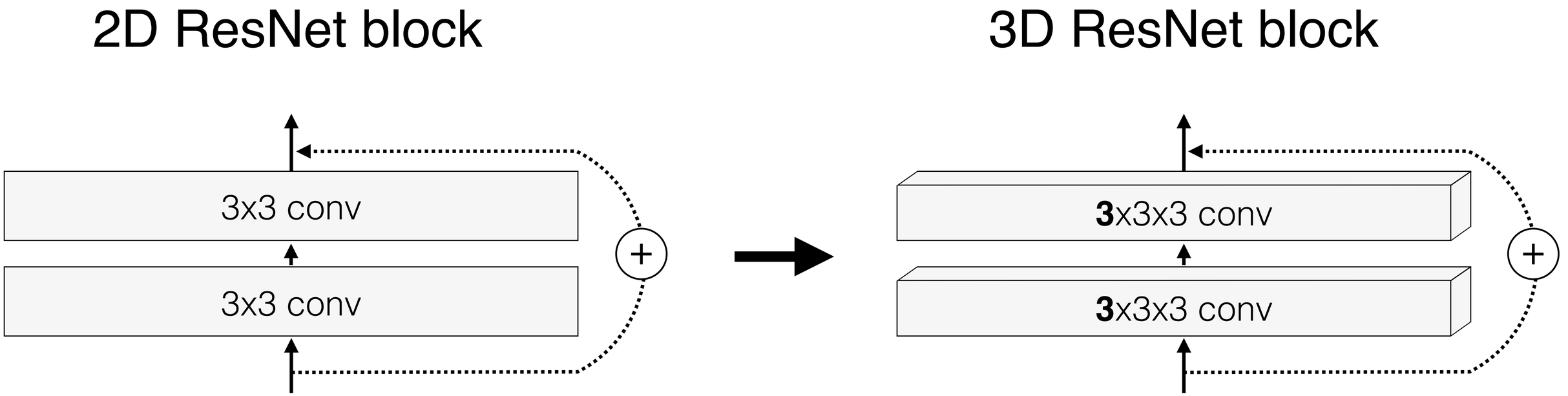
[fig from FeatureNet: Machining feature recognition based on 3D Convolution Neural Network]

Source: Torralba, Freeman, Isola

Inflated convolutions



Inflated convolutions

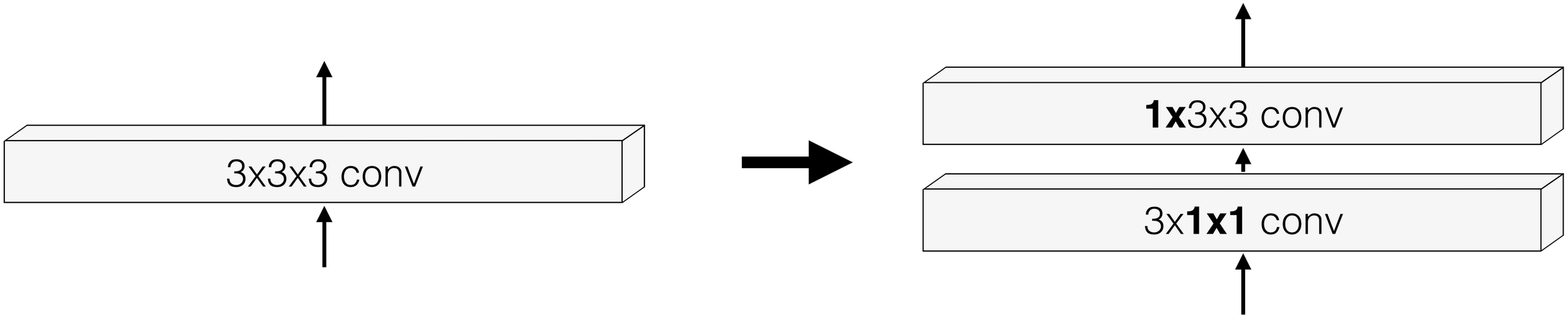


- Can reuse 2D architectures. [Carreira et al. 2017]
- Pretrain with 2D nets (“inflating” 2D filter to 3D)

Separable convolutions

3D convolution

Separate space/time



Faster and fewer parameters

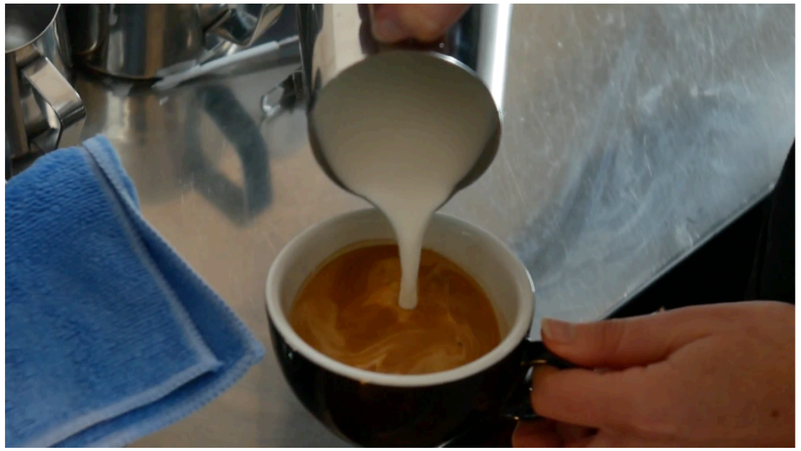
Are we solving the right problem?

Photo by H. Edgerton

Are our models using motion?

- Single-frame models do badly.
- Shuffling frames after training hurts performance
- But 2D convolutions + temporal pooling often is competitive!

When do we actually need motion?



Making latte art

Jaywalking

Grooming dog

Multiple frames often means multiple *samples*

Localizing objects in space and time

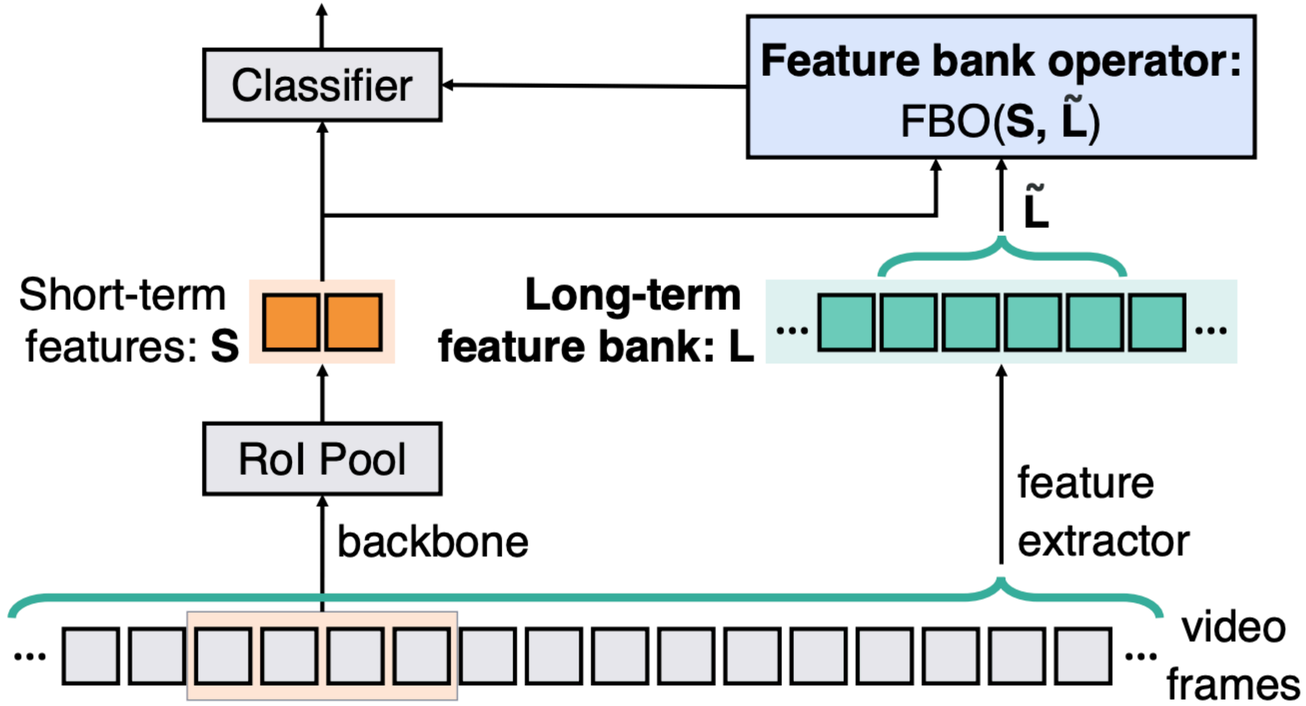
Left: Sit, Talk to, Watch; Right: Crouch/Kneel, Listen to, Watch

Left: Stand, Carry/Hold, Listen to; Middle: Stand, Carry/Hold, Talk to; Right: Sit, Write

Left: Sit, Ride, Talk to; Right: Sit, Drive, Listen to

Left: Stand, Watch; Middle: Stand, Play instrument; Right: Sit, Play instrument

AVA dataset [Gu et al. 2018]

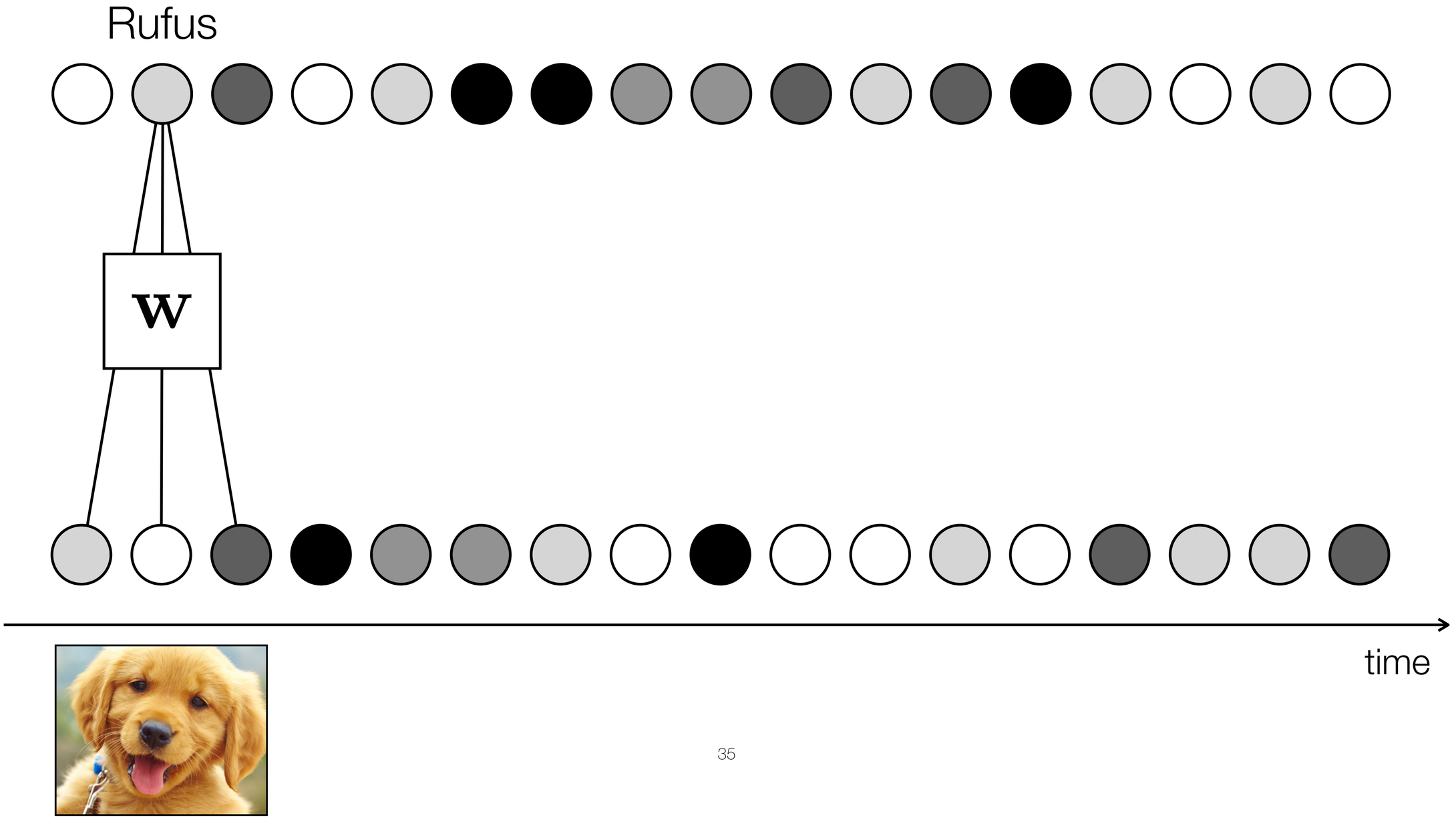


[Wu et al. 2019]

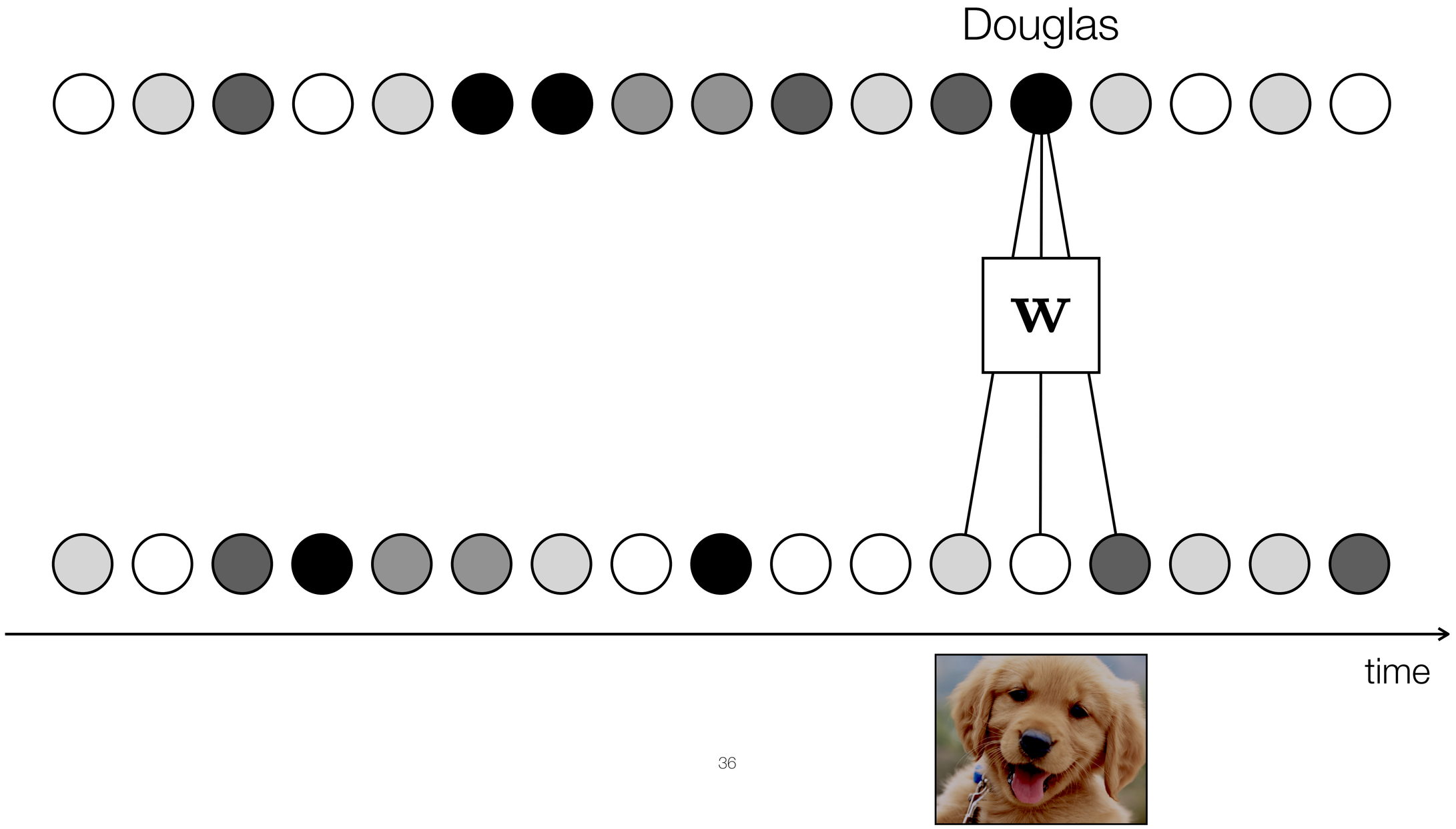
Recurrent networks

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

Source: Torralba, Freeman, Isola



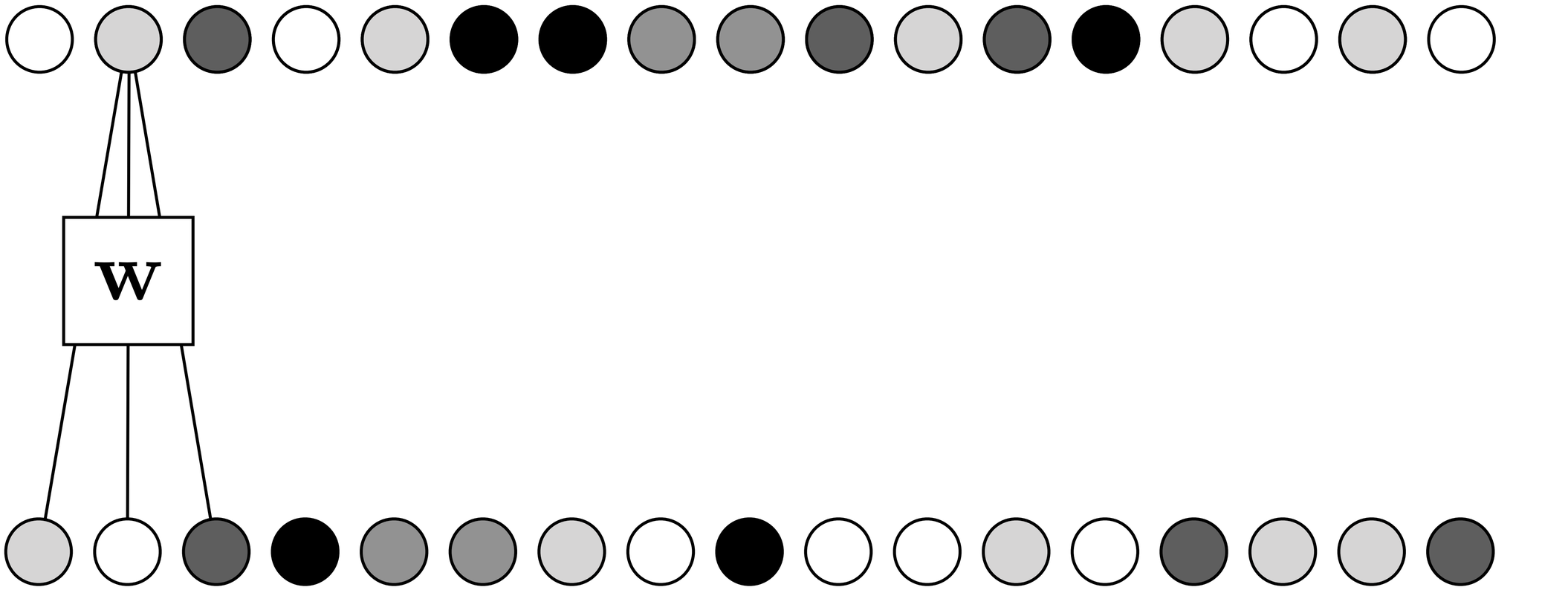
Source: Torralba, Freeman, Isola



Source: Torralba, Freeman, Isola

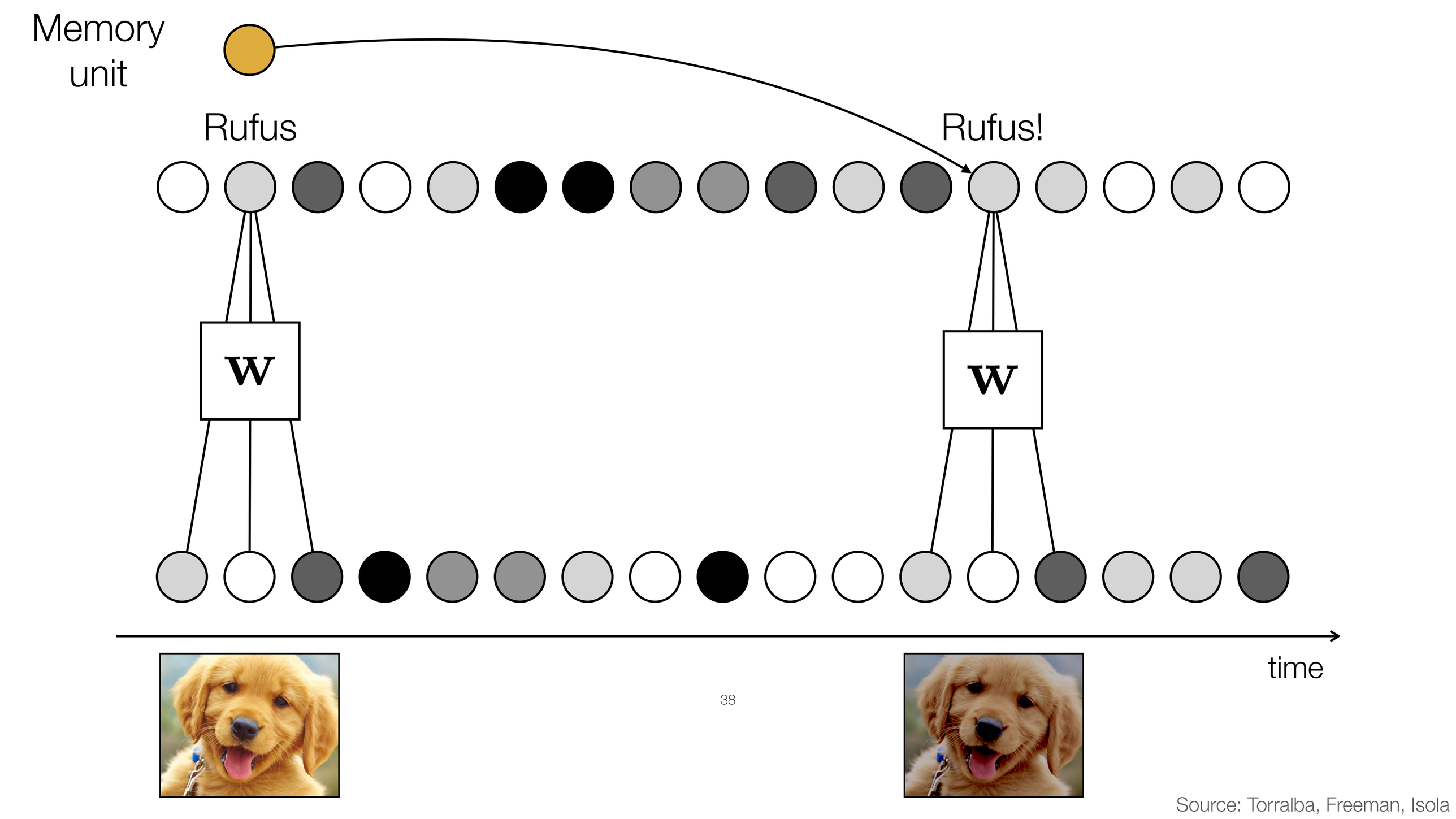
Memory unit

Rufus



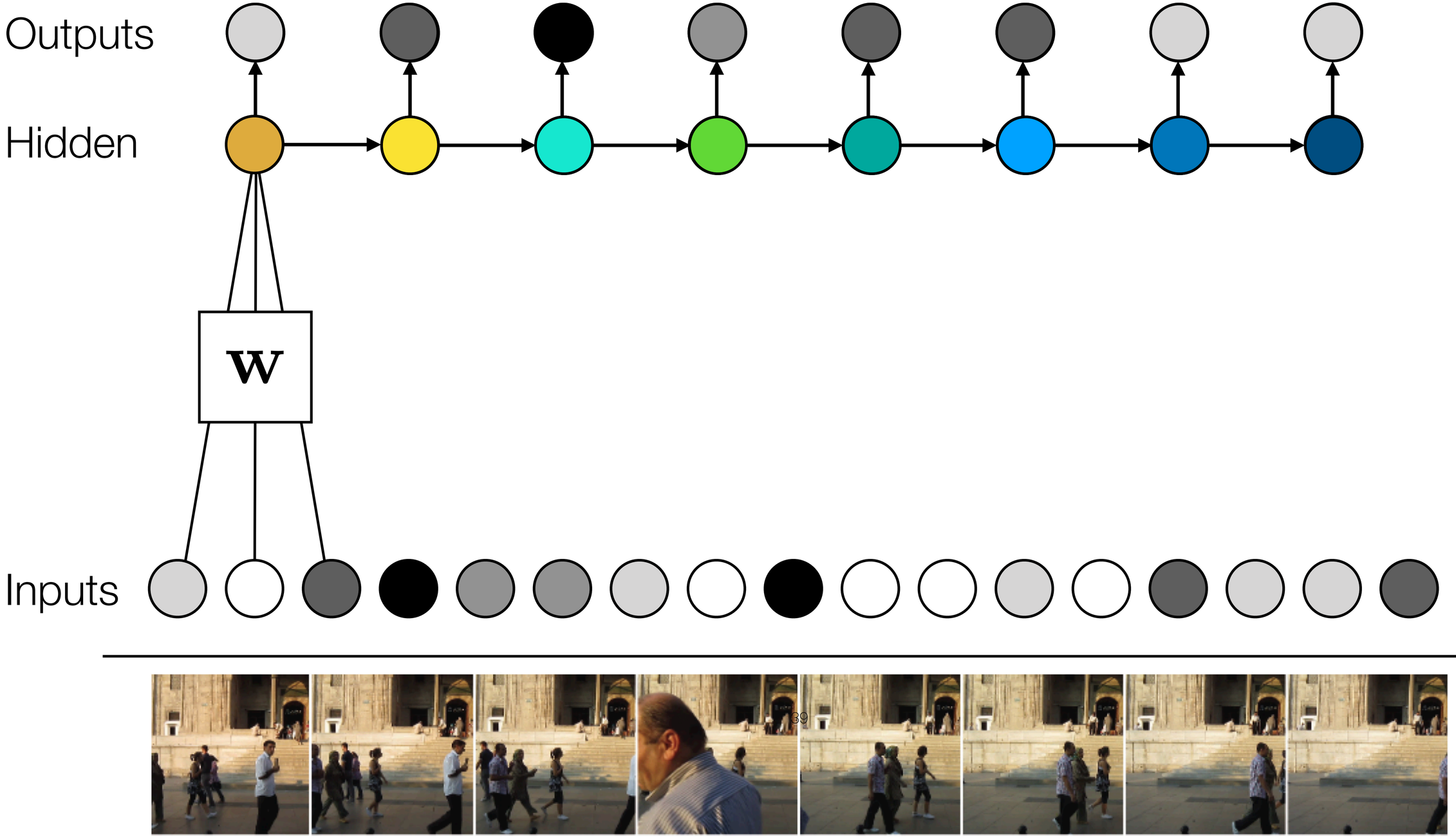
time

Source: Torralba, Freeman, Isola

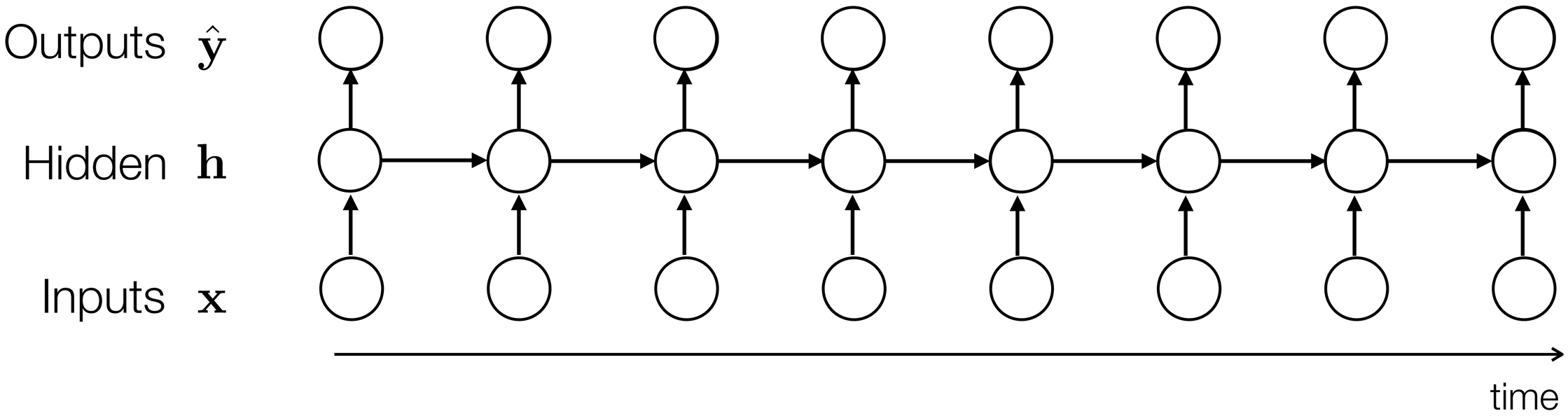


Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)

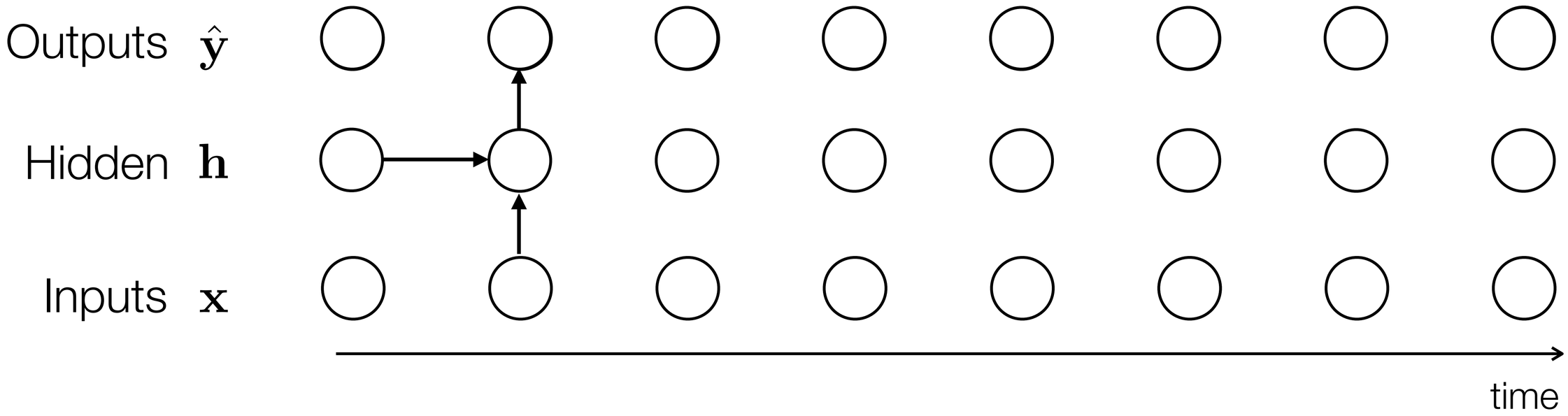


Recurrent Neural Networks (RNNs)



Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)

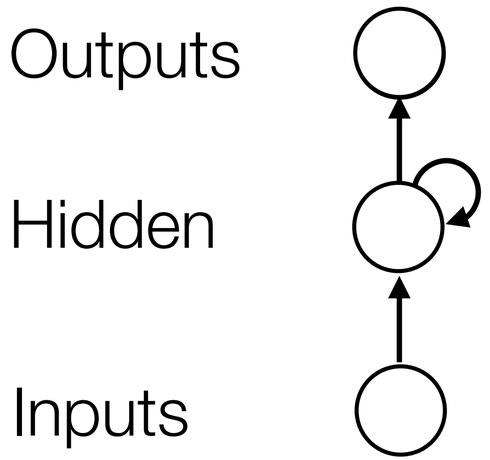


$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$

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

Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)



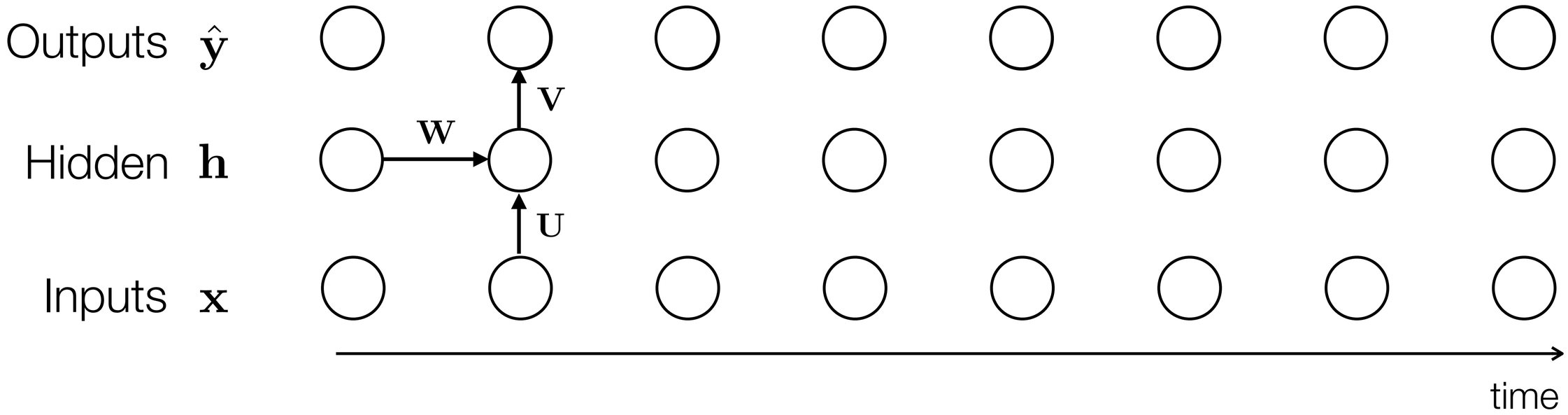
Recurrent!

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

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

Source: Torralba, Freeman, Isola

Recurrent Neural Networks (RNNs)



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

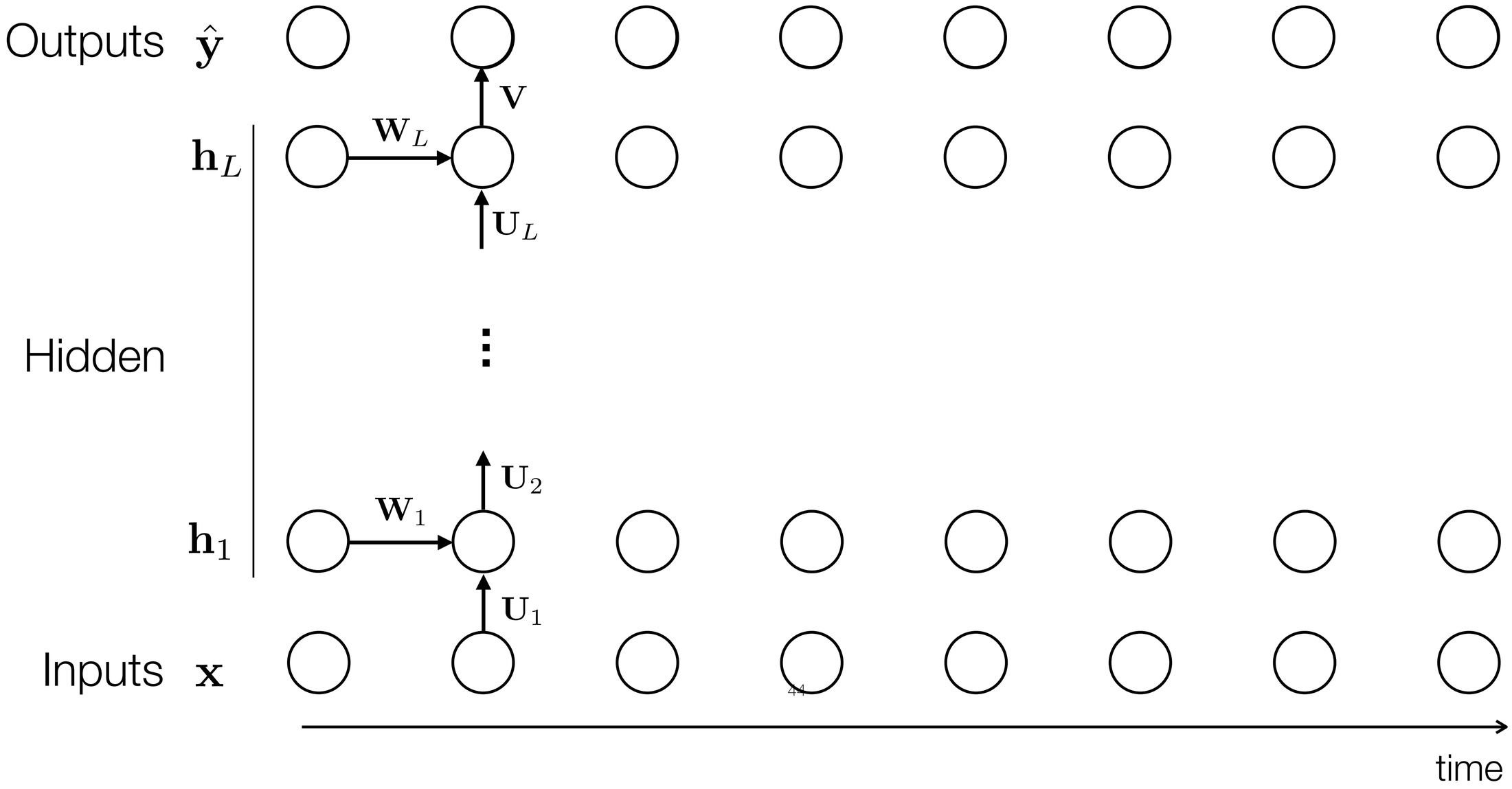
$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

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

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

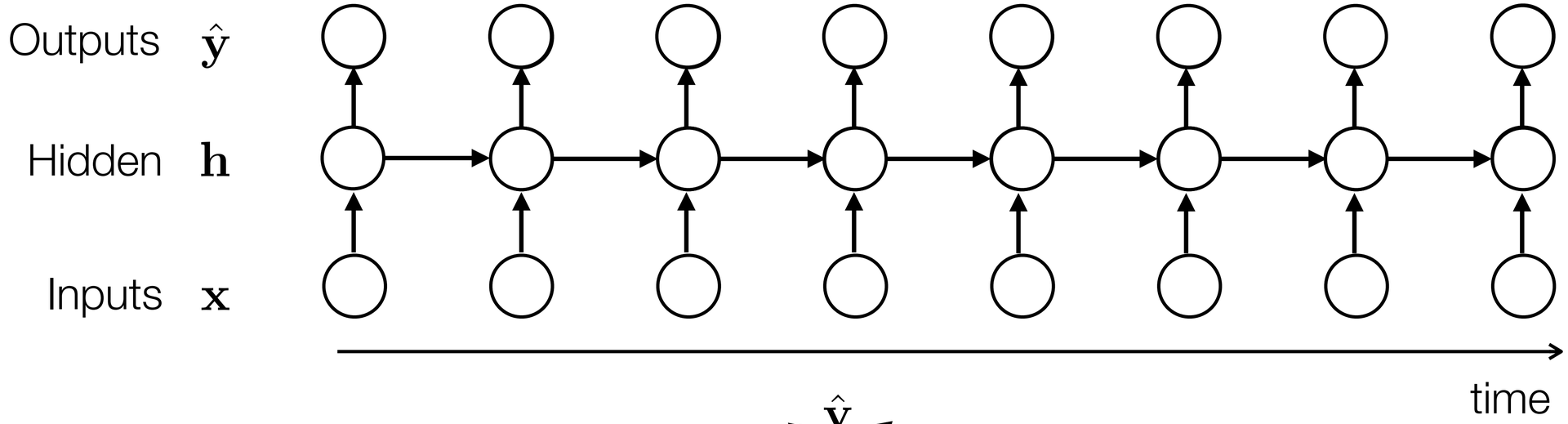
Source: Torralba, Freeman, Isola

Deep Recurrent Neural Networks (RNNs)

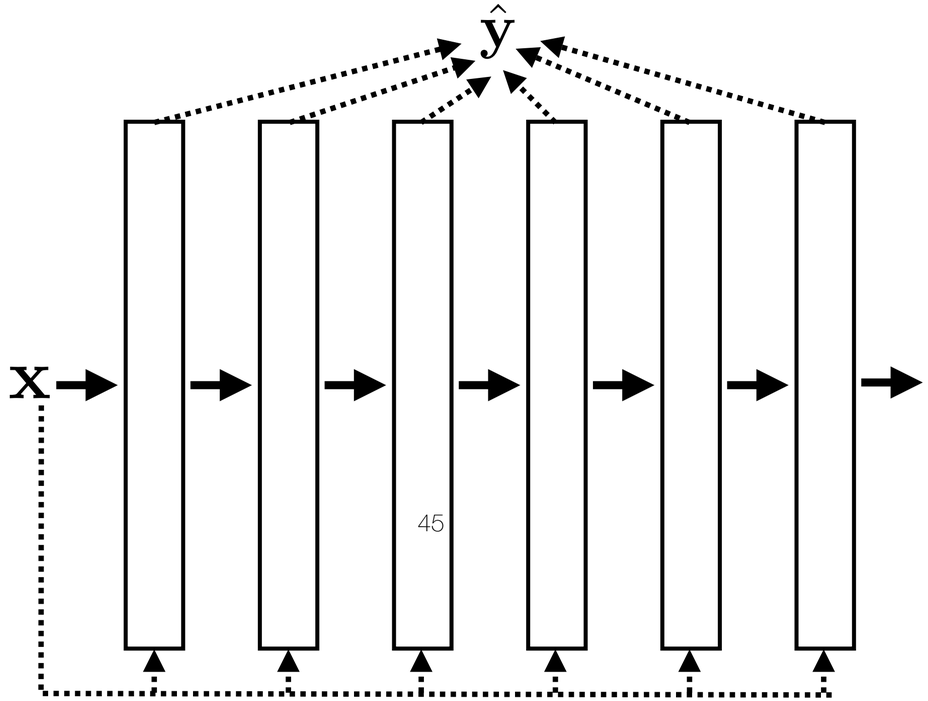


Source: Torralba, Freeman, Isola

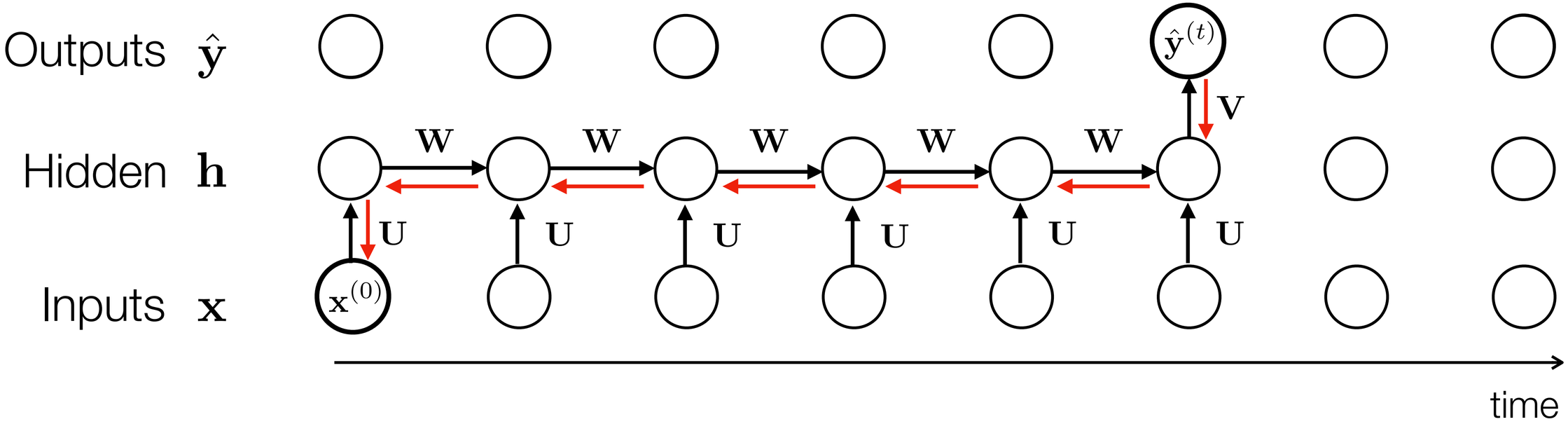
Unrolling an RNN



Equivalent to "unrolled" network with lots of layers



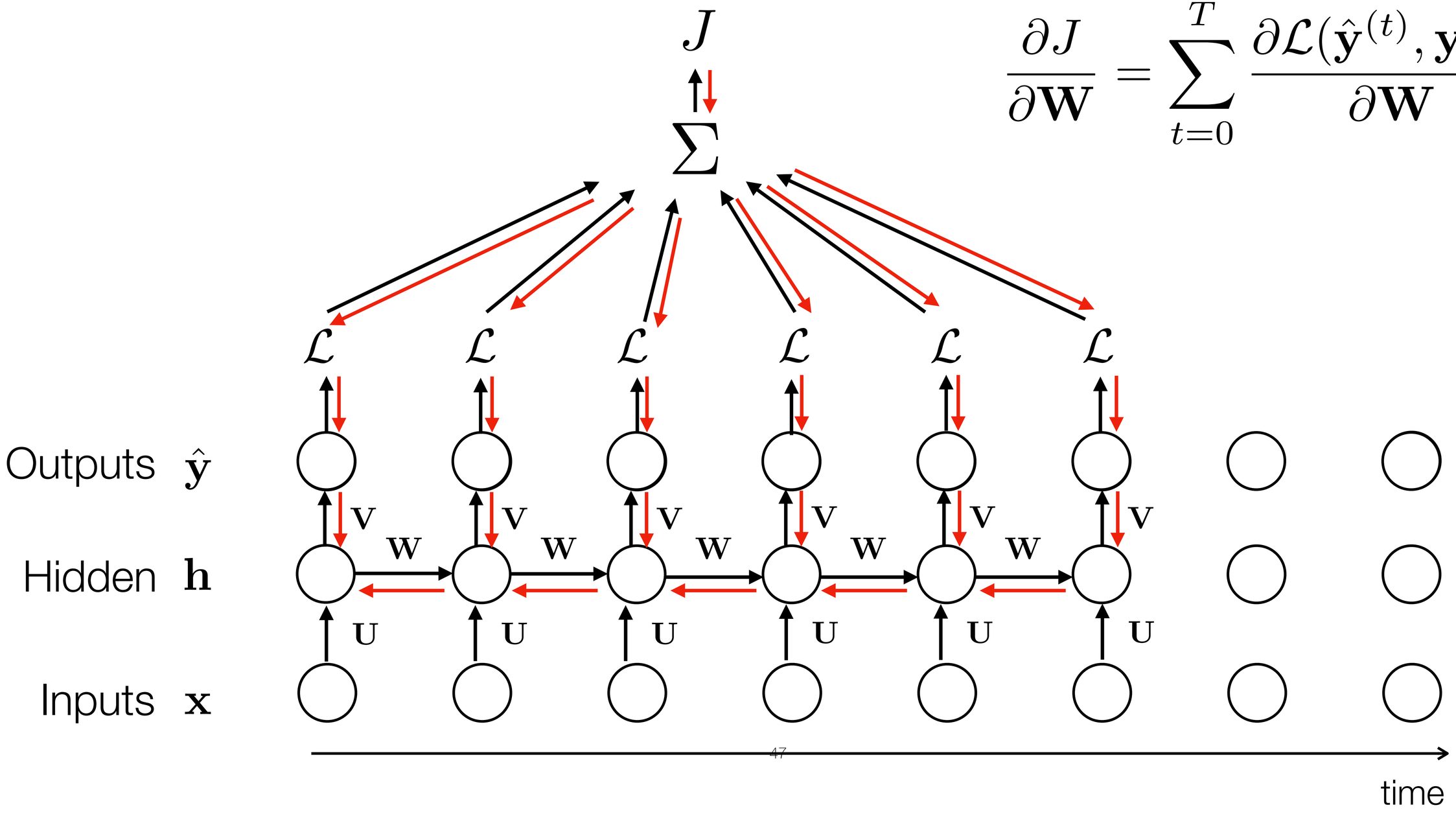
Backprop through time



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

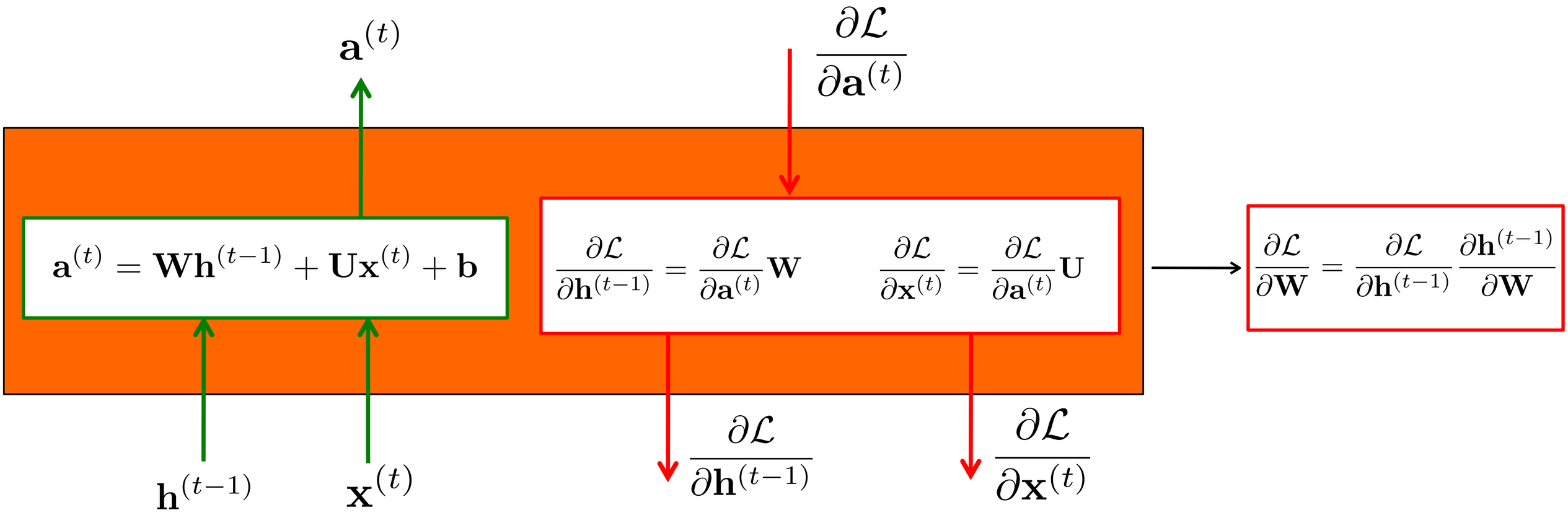
Source: Torralba, Freeman, Isola

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



Source: Torralba, Freeman, Isola

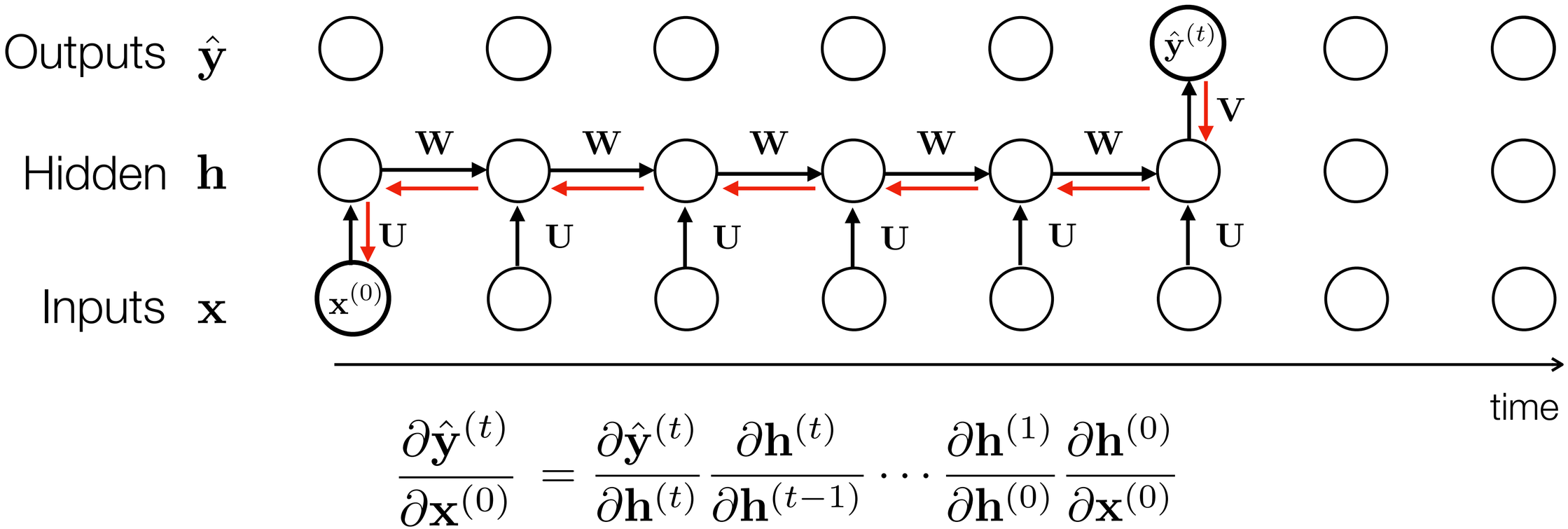
Recurrent linear layer



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

Source: Torralba, Freeman, Isola

The problem of long-range dependences



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

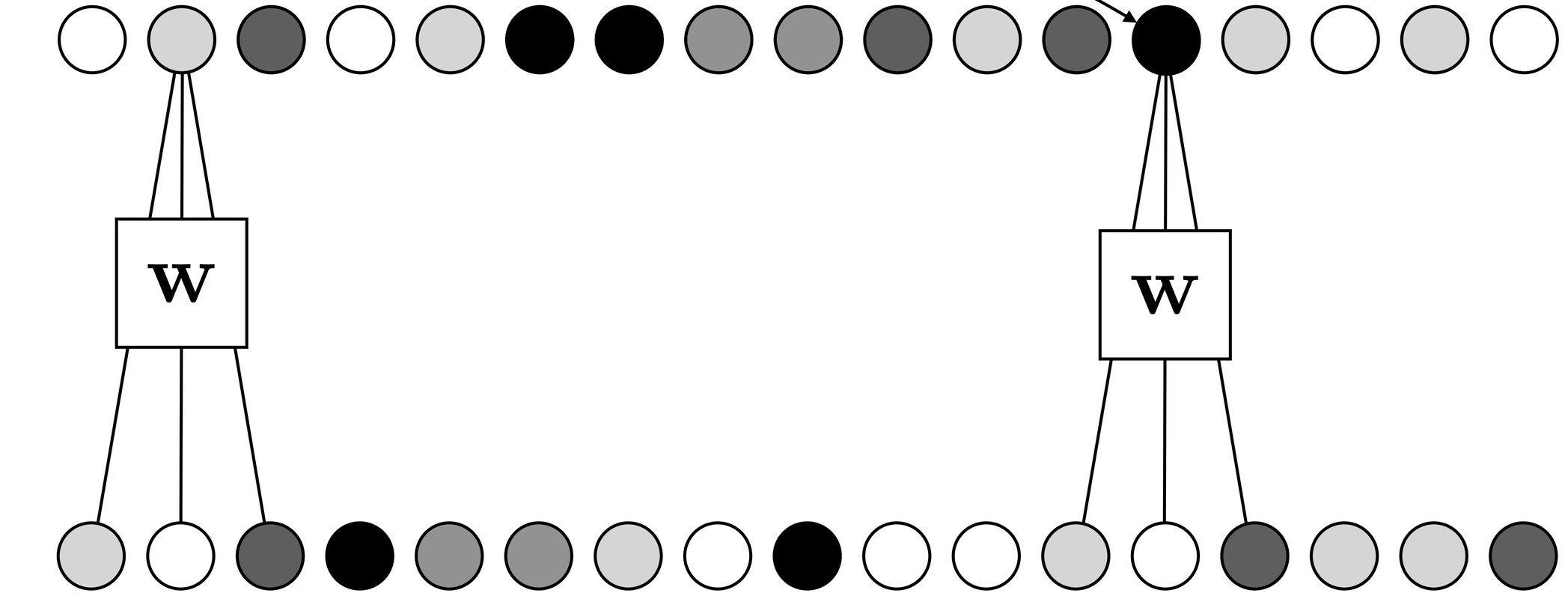
- Capturing long-range dependences requires propagating information through a long chain of dependences.
- Old observations are forgotten
- Stochastic gradients become high variance (noisy), and gradients may **vanish** or **explode**

Source: Torralba, Freeman, Isola

Memory unit

Rufus

Rufus!



time

Source: Torralba, Freeman, Isola

Memory units

...

W

W

time

Source: Torralba, Freeman, Isola

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
- Markov-like assumption: future state only dependent on preceding hidden state. Very easy to forget things.
- By putting the right info in to the hidden state, RNNs can model dependencies that are arbitrarily far apart

Source: Torralba, Freeman, Isola

LSTMs

Long Short Term Memory

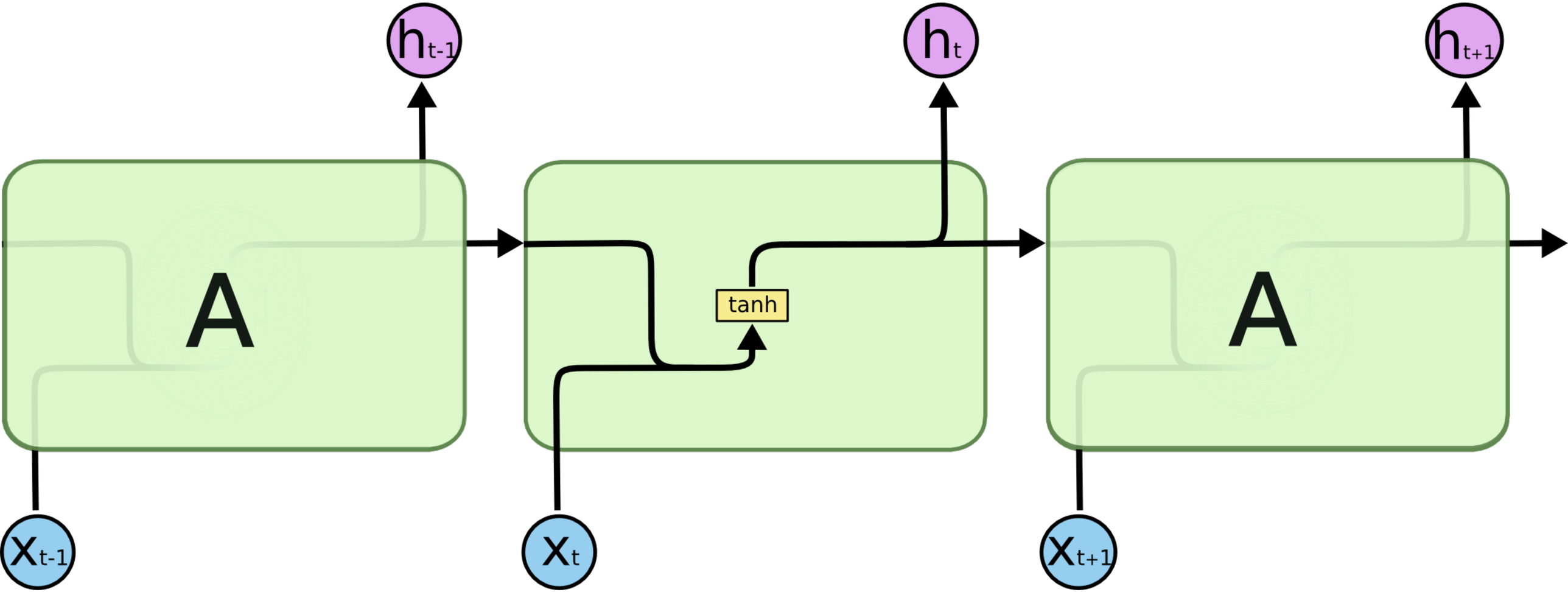
A special kind of RNN designed to avoid forgetting.

Related to ResNets inductive 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*.

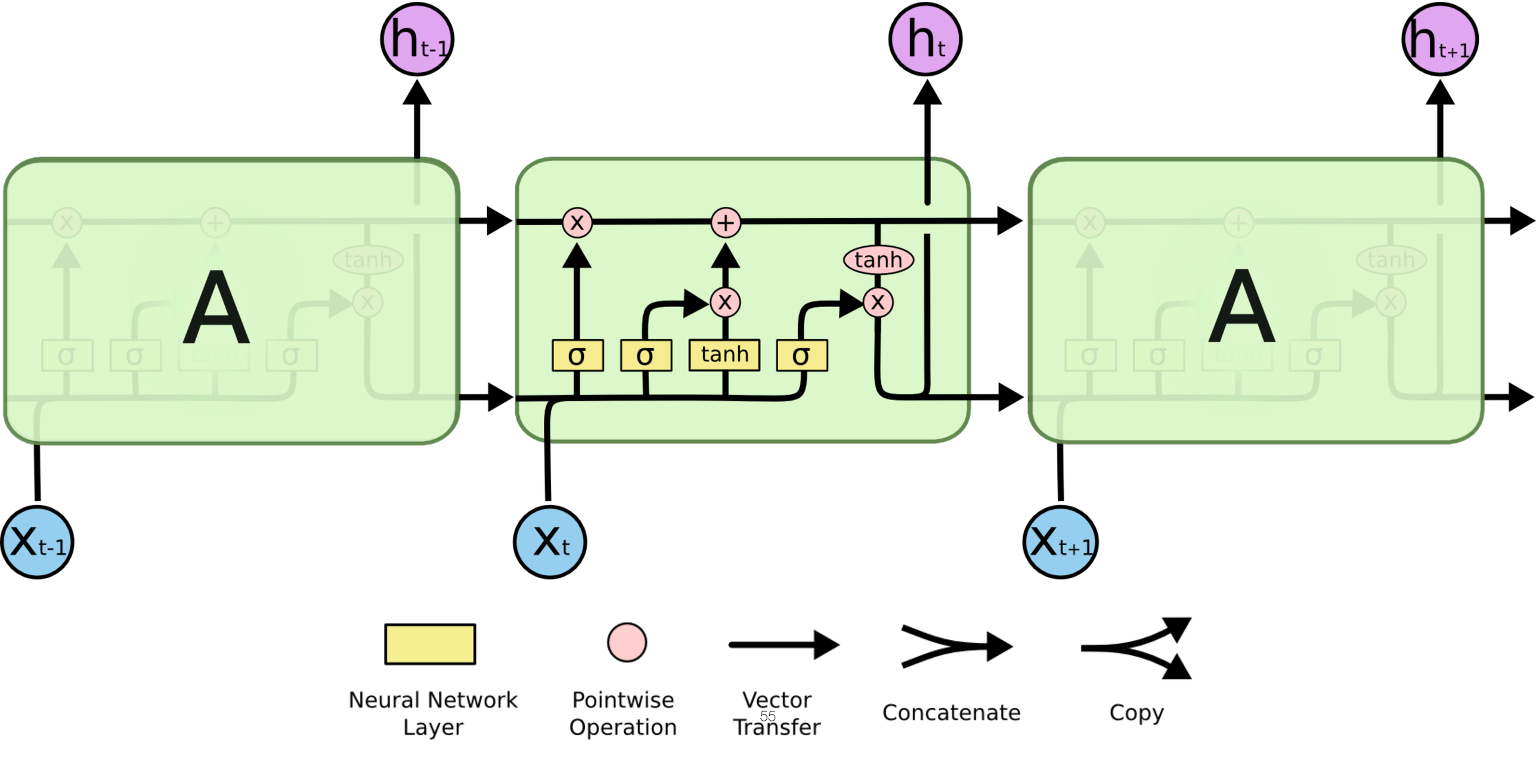
Bit of a complex design. Works well but simpler methods like Gated Recurrent Unit (GRU) are competitive [Jozefowicz et al. 2015].

Source: Torralba, Freeman, Isola



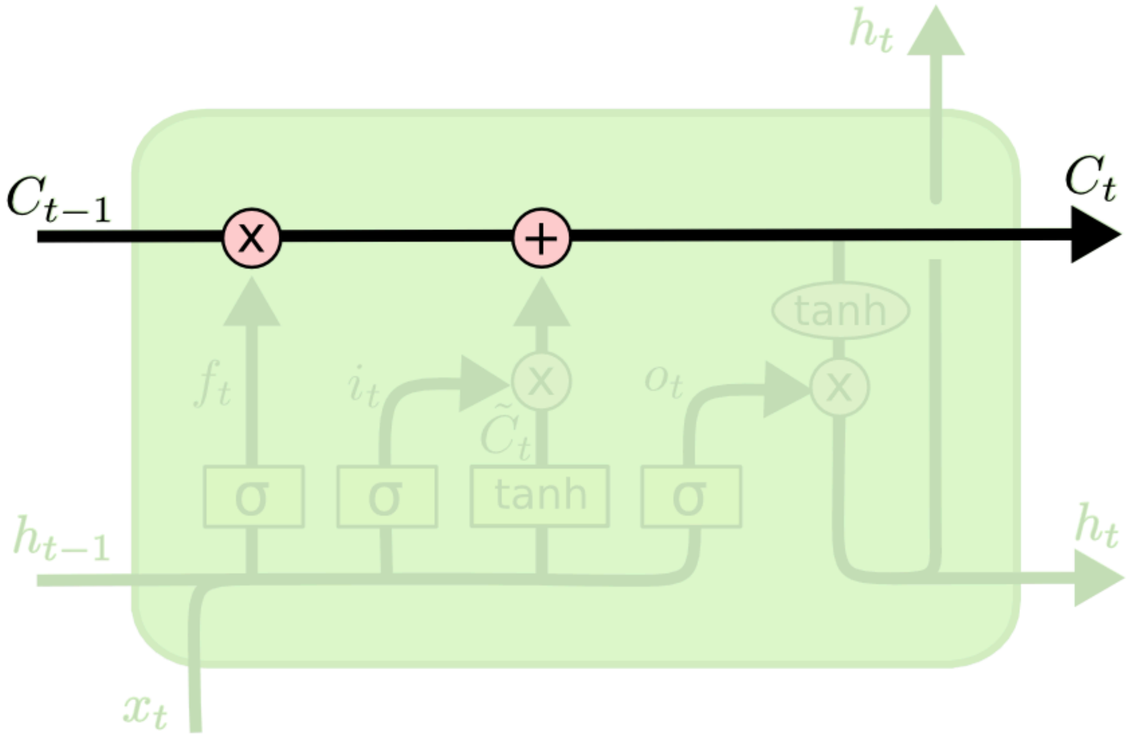
[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]

Source: Torralba, Freeman, Isola



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

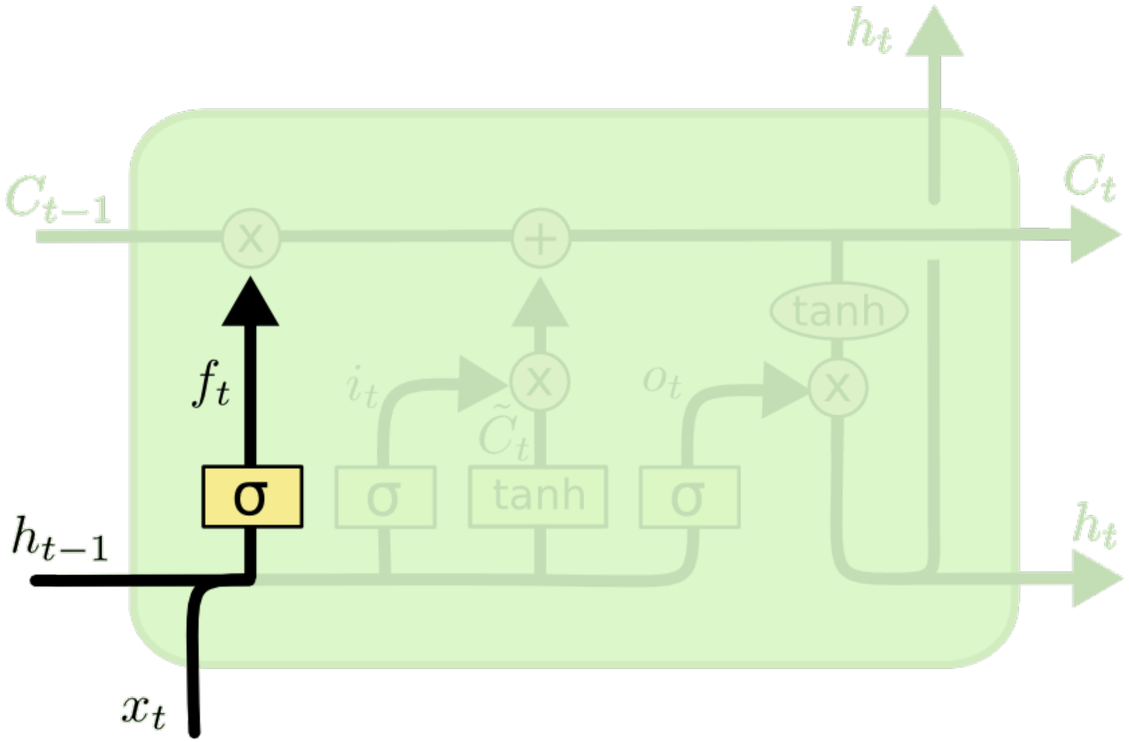
Source: Torralba, Freeman, Isola

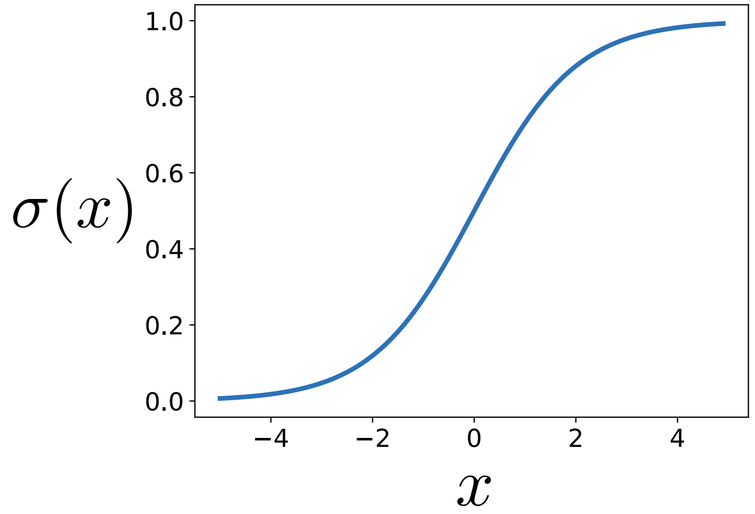


C_t = Cell state

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

Source: Torralba, Freeman, Isola





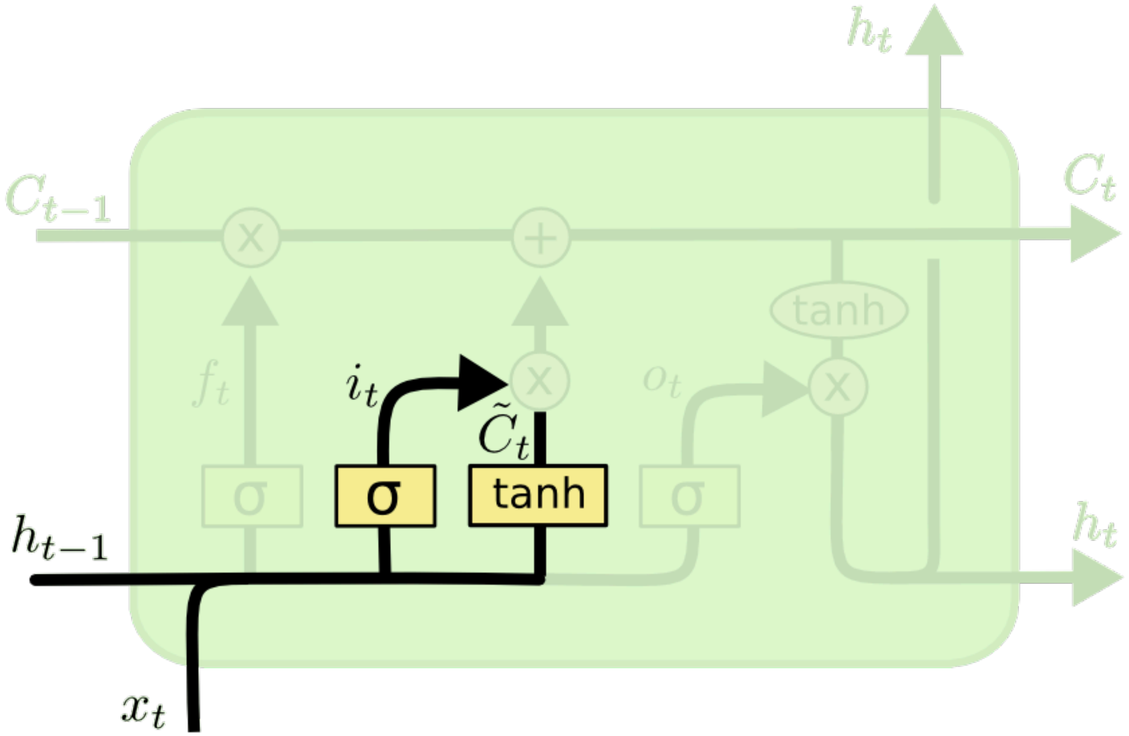
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Decide what information to throw away from the cell state.

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



which indices to write to

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

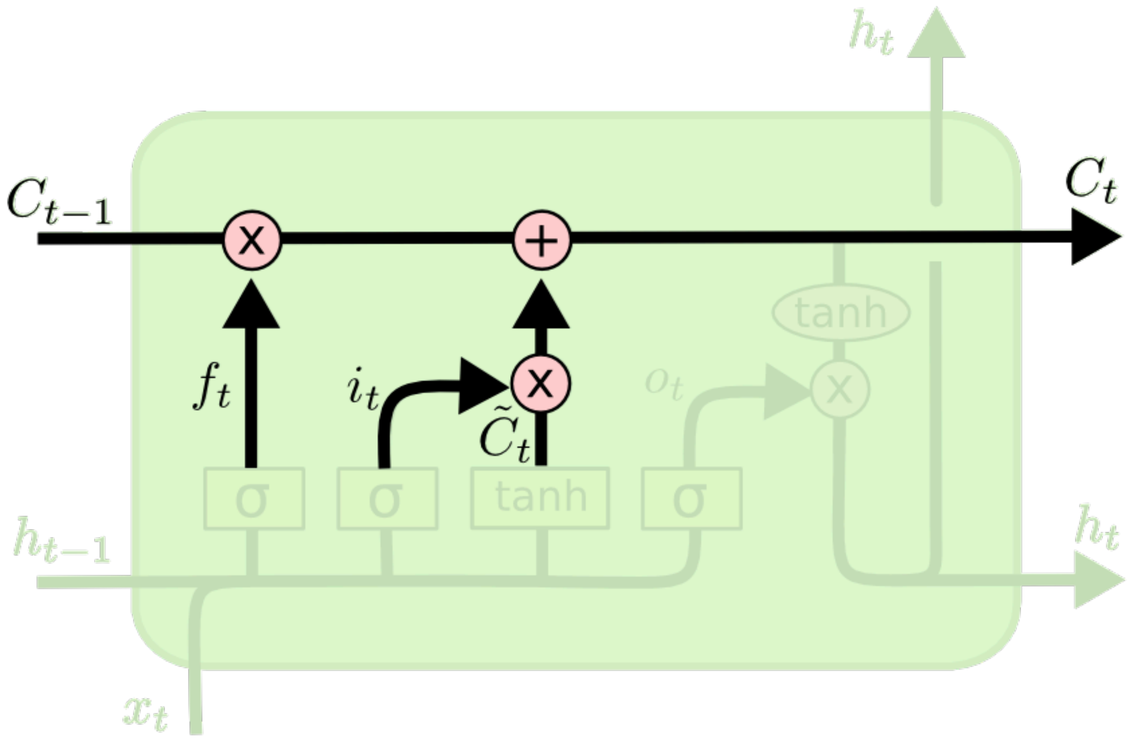
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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

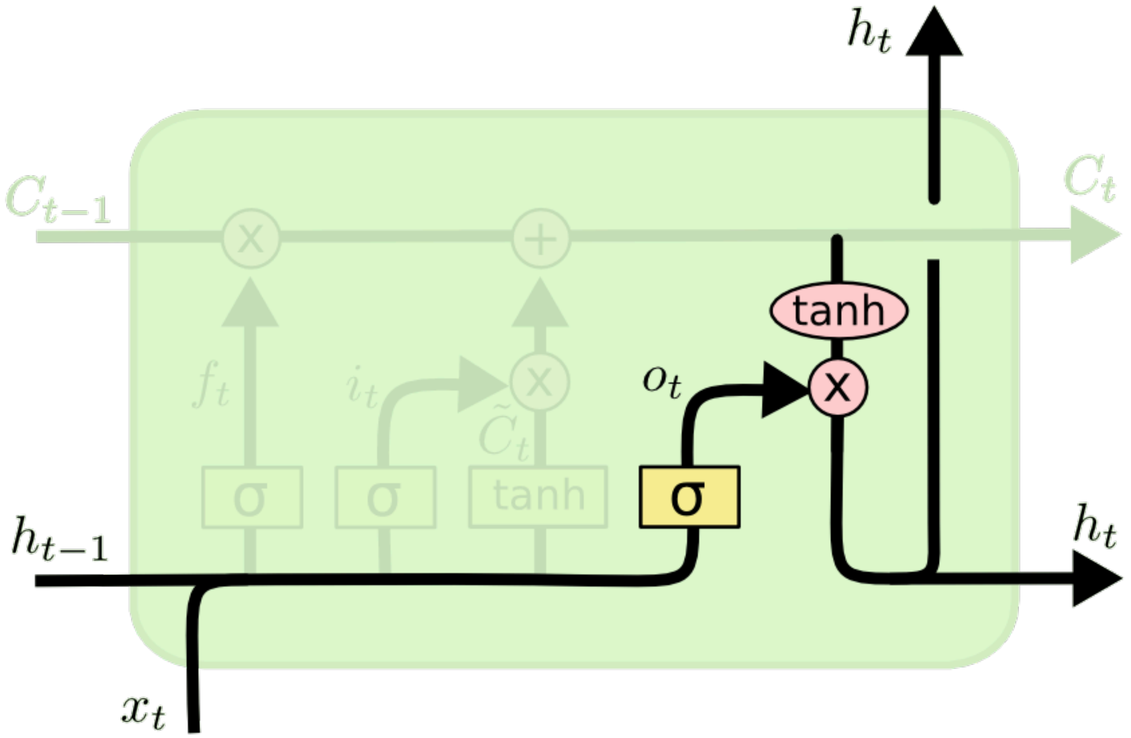


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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



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

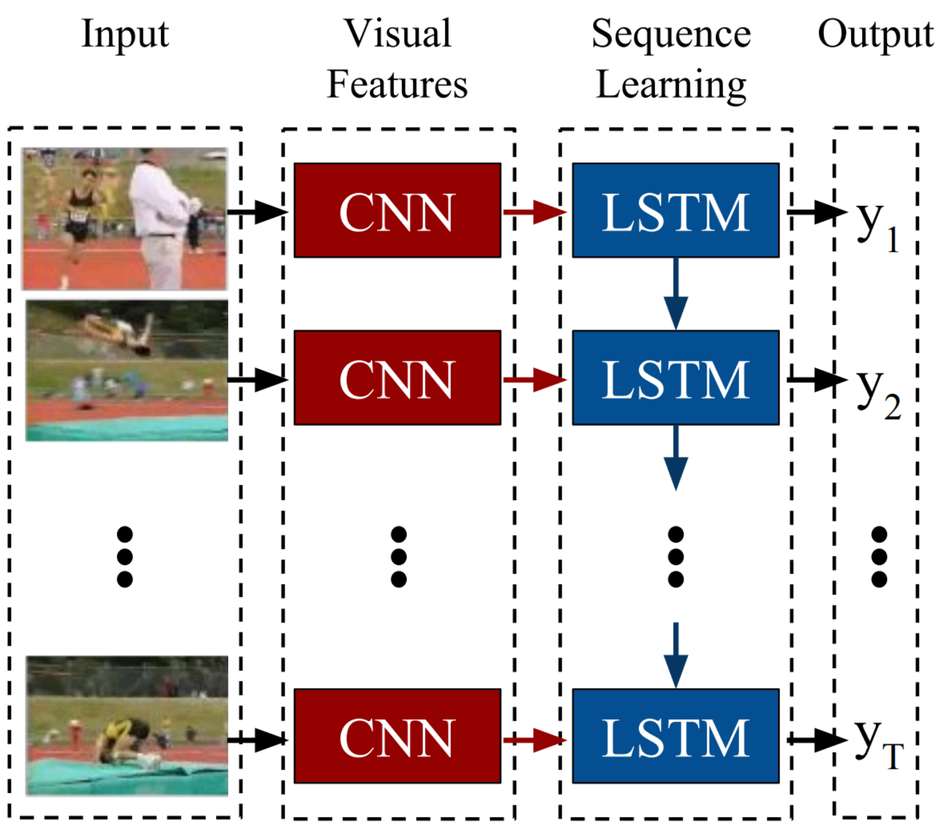
$$h_t = o_t * \tanh\left(C_t\right)$$

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 LSTMs



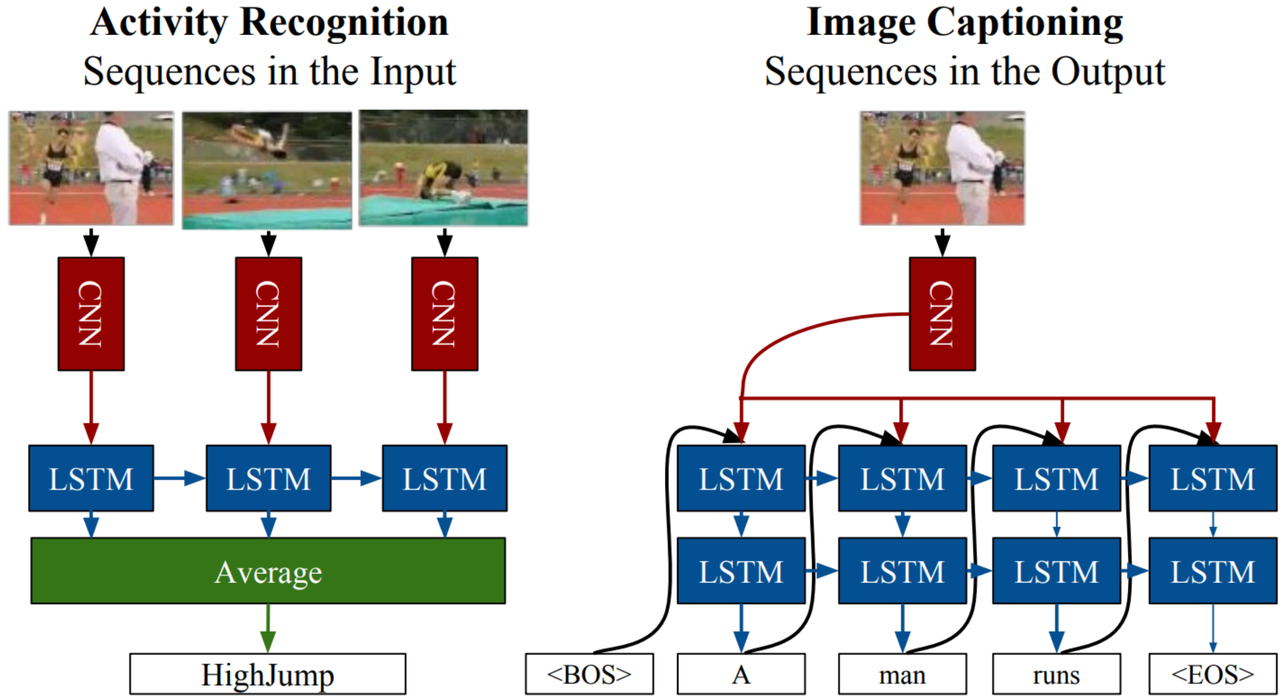


Figure from [Donahue et al. 2016]

The problem of long-range dependences

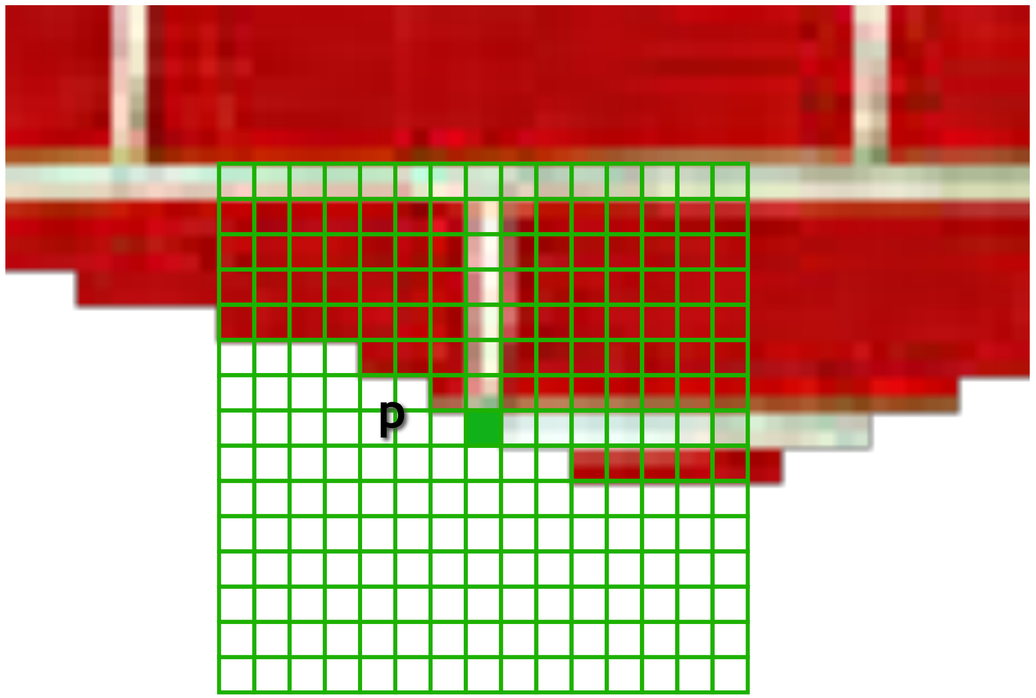
Other methods exist that do directly link old "memories" (observations or hidden states) to future predictions:

- Temporal convolutions (but temporal horizon often short)
- Attention (see https://arxiv.org/abs/1706.03762)
- Memory networks (see https://arxiv.org/abs/1410.3916)

Source: Torralba, Freeman, Isola

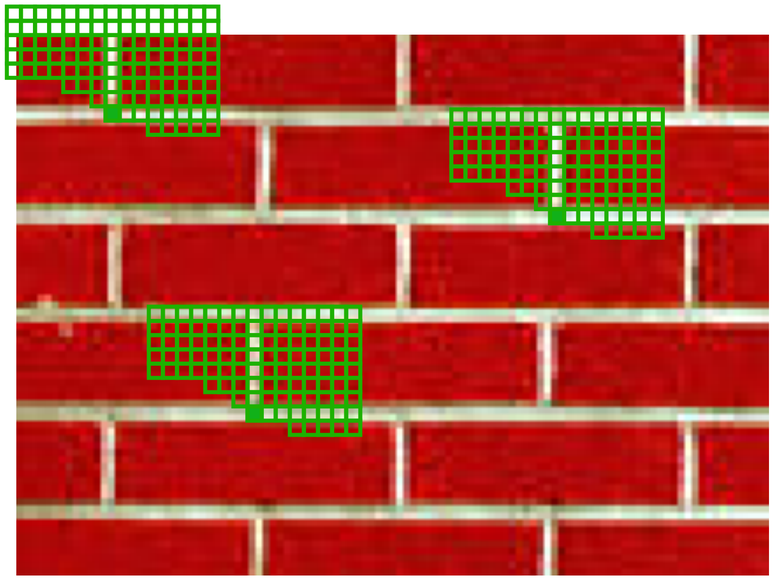
Other uses for sequence models

Texture synthesis by non-parametric sampling



Synthesizing a pixel

non-parametric sampling



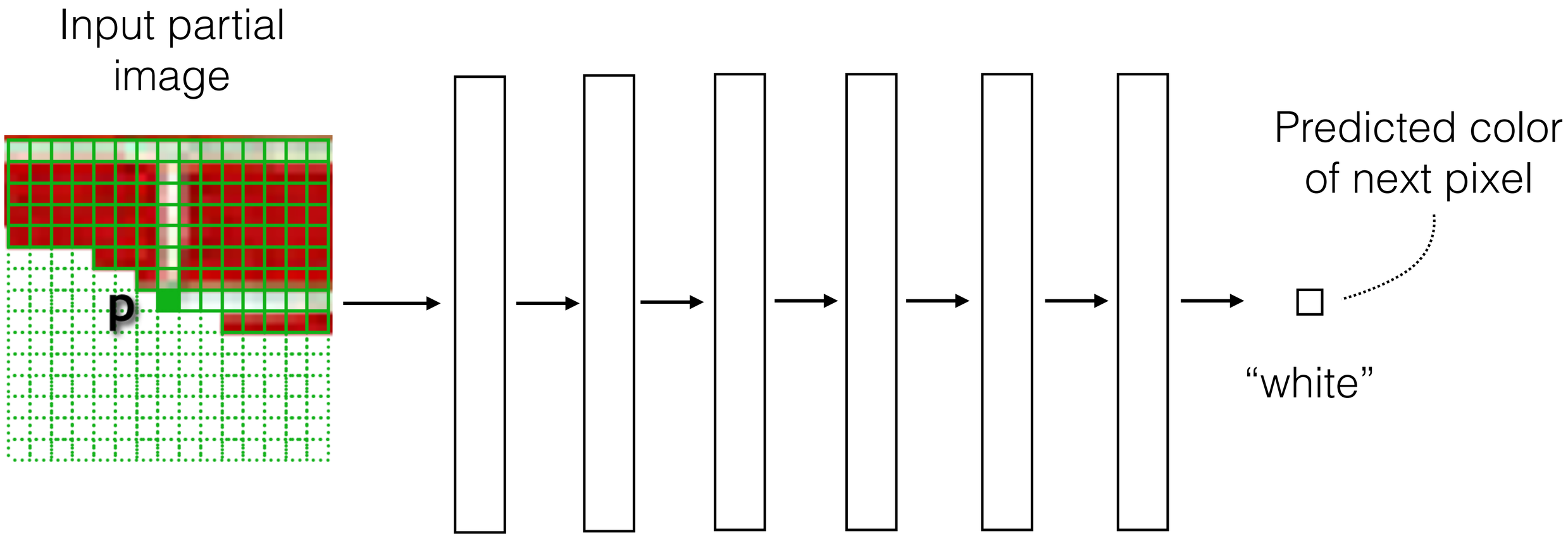
Input image

Models $P(p|N(p))$

[Efros & Leung 1999]

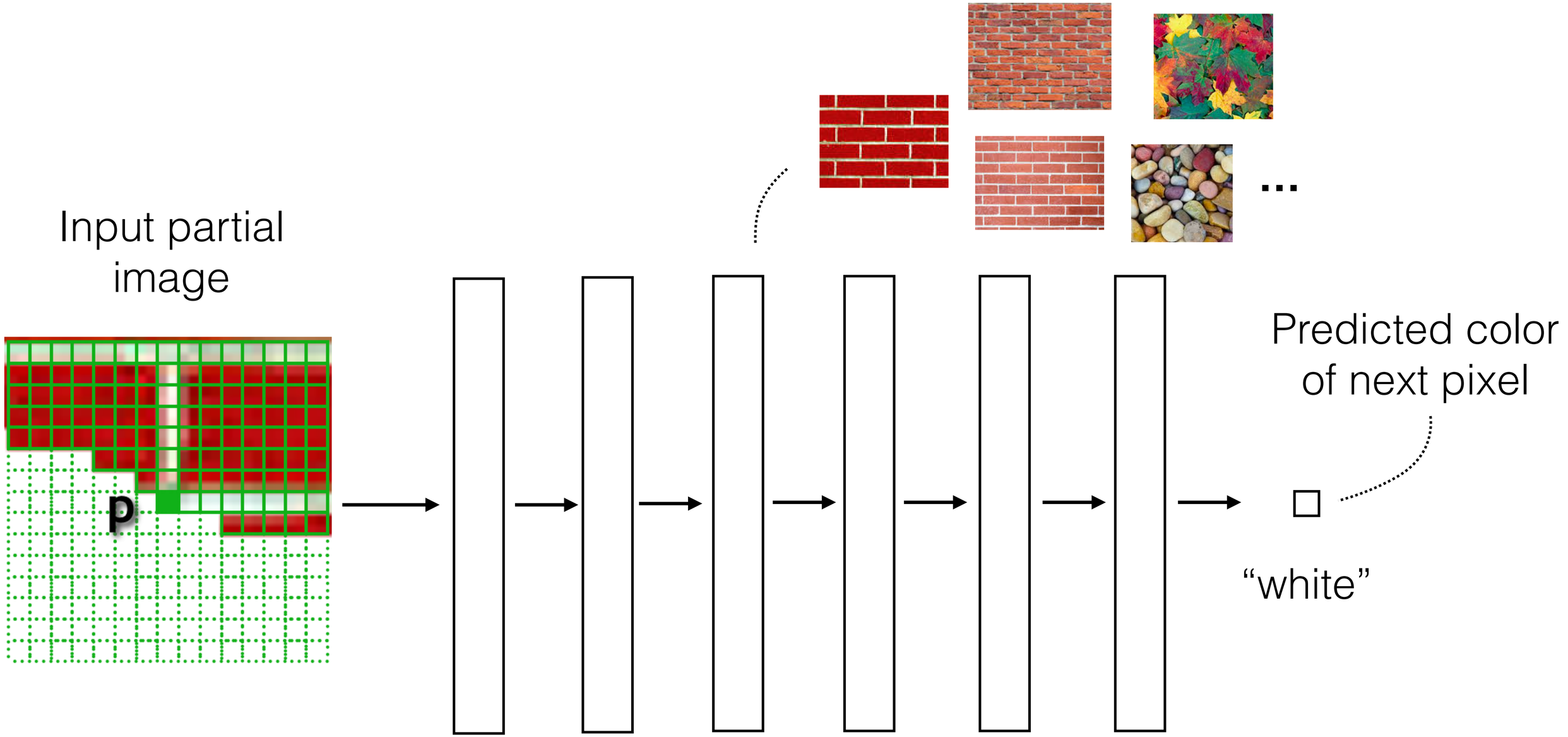
Source: Torralba, Freeman, Isola

Texture synthesis with a deep net



[PixelRNN, PixelCNN, van der Oord et al. 2016]

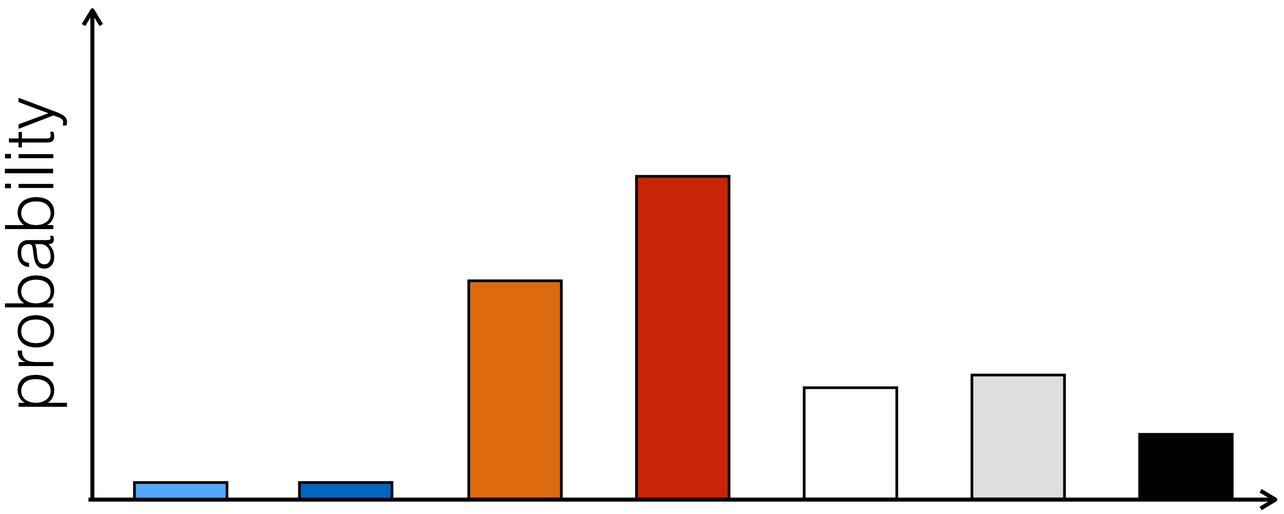
Source: Torralba, Freeman, Isola



[PixelRNN, PixelCNN, van der Oord et al. 2016]

Source: Torralba, Freeman, Isola

Represent colors as discrete categories



- One class for each intensity value, e.g. 256 classes
- One label per color channel

Source: Torralba, Freeman, Isola

And we can interpret the learner as modeling P(next pixel | previous pixels):

Softmax regression (a.k.a. multinomial logistic regression)

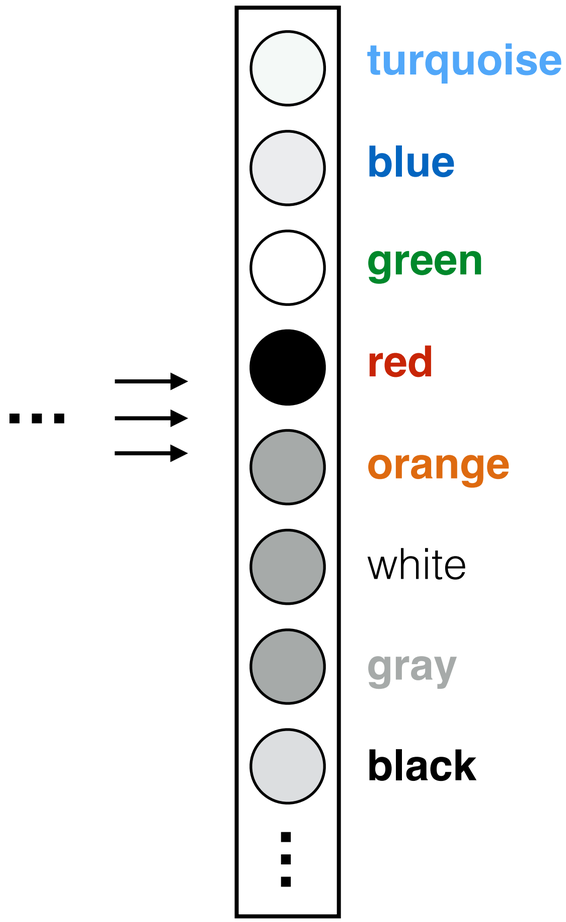
$$\hat{\mathbf{y}} \equiv [P_\theta(Y = 1|X = \mathbf{x}), \ldots, P_\theta(Y = K|X = \mathbf{x})]$$ ← predicted probability of each class given input **x**

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$ ← picks out the -log likelihood of the ground truth class $\mathbf{y}$ under the model prediction $\hat{\mathbf{y}}$

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

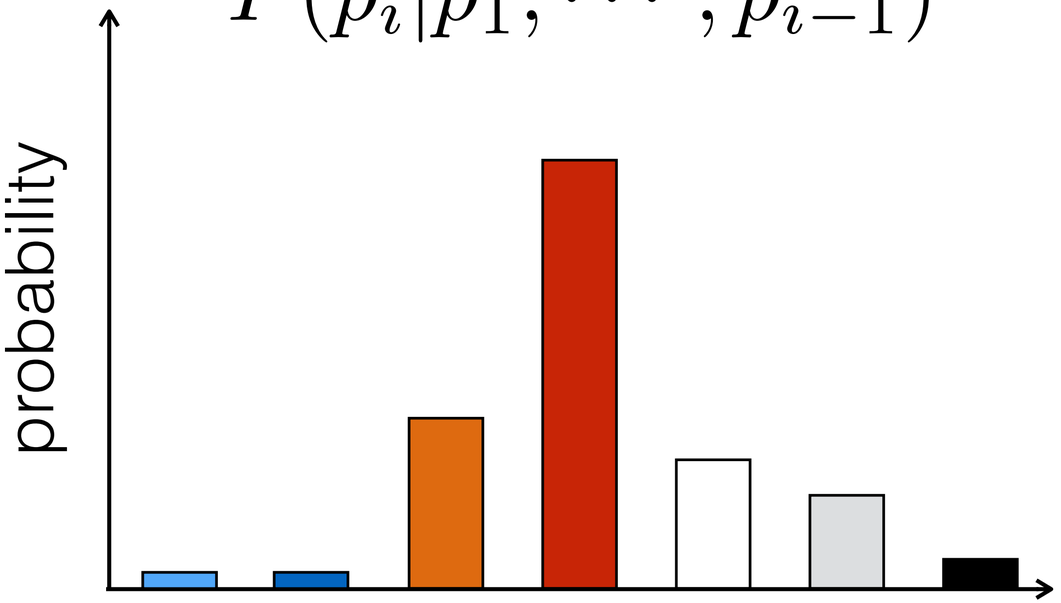
Source: Torralba, Freeman, Isola

Network output



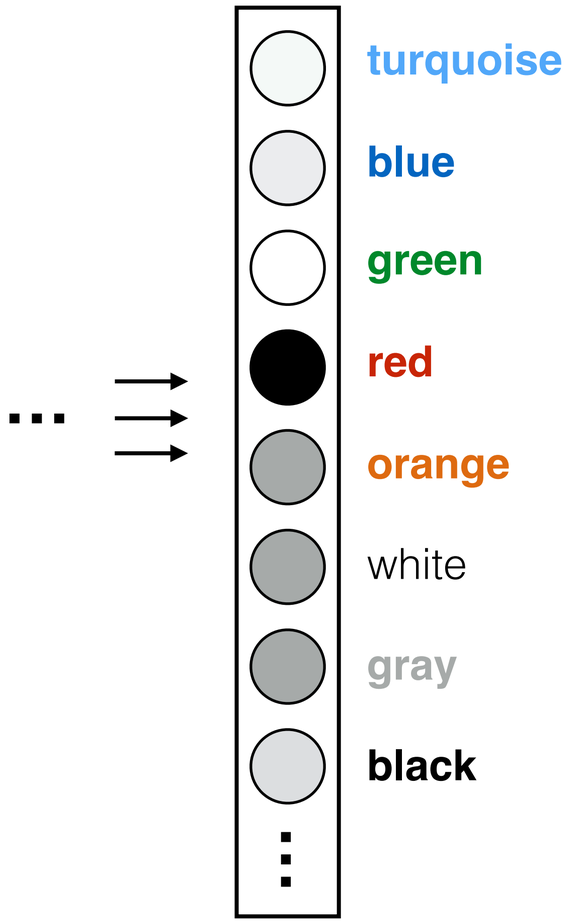
P(next pixel | previous pixels)

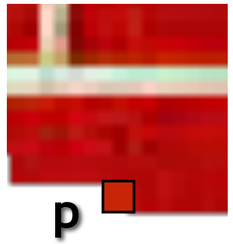
$$P(p_i | p_1, \cdots, p_{i-1})$$



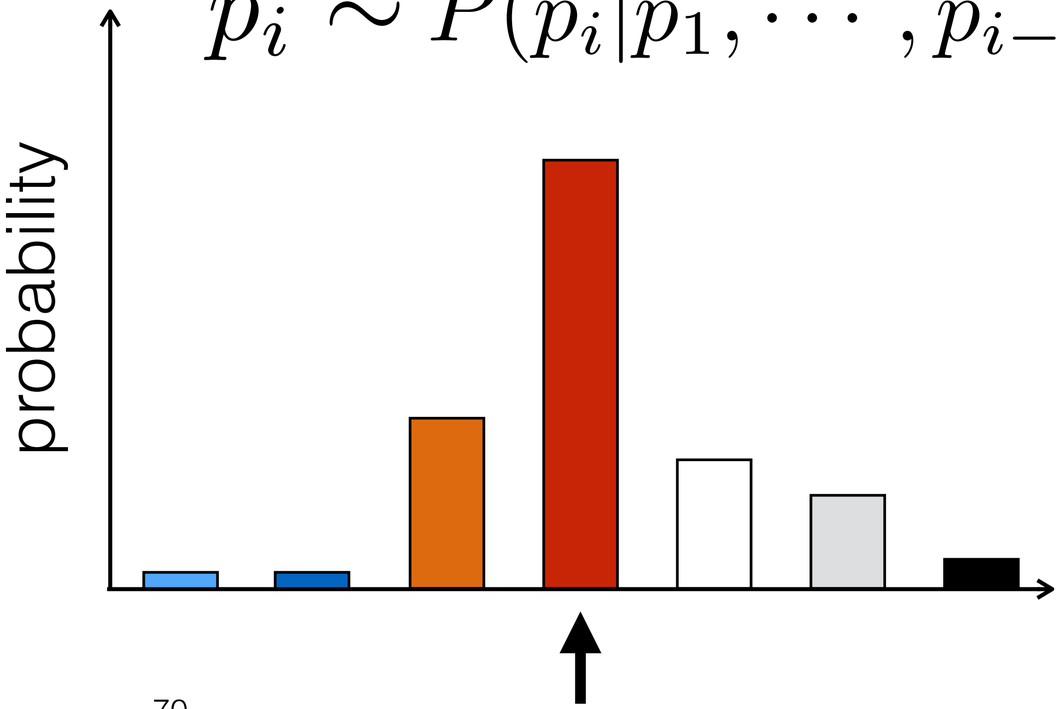
Source: Torralba, Freeman, Isola

Network output



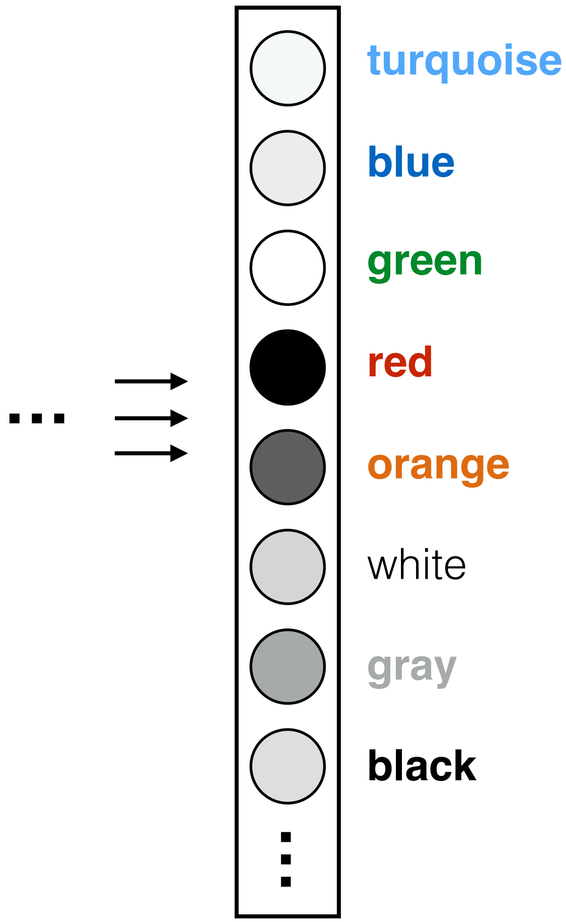


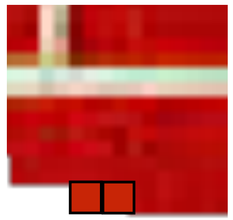
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



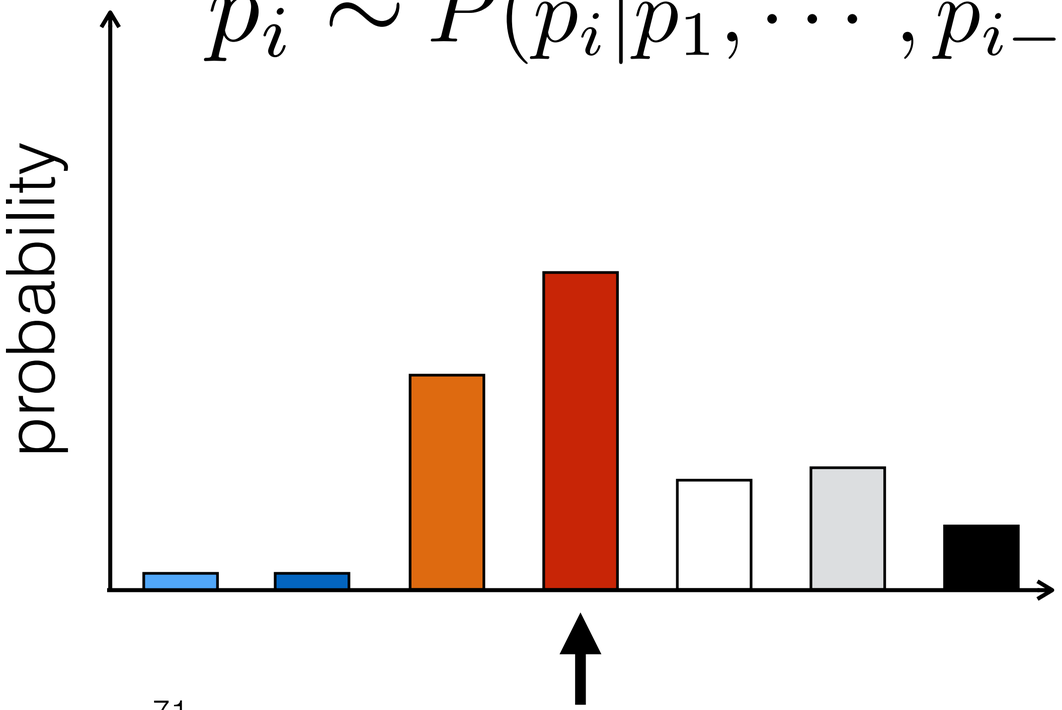
Source: Torralba, Freeman, Isola

Network output



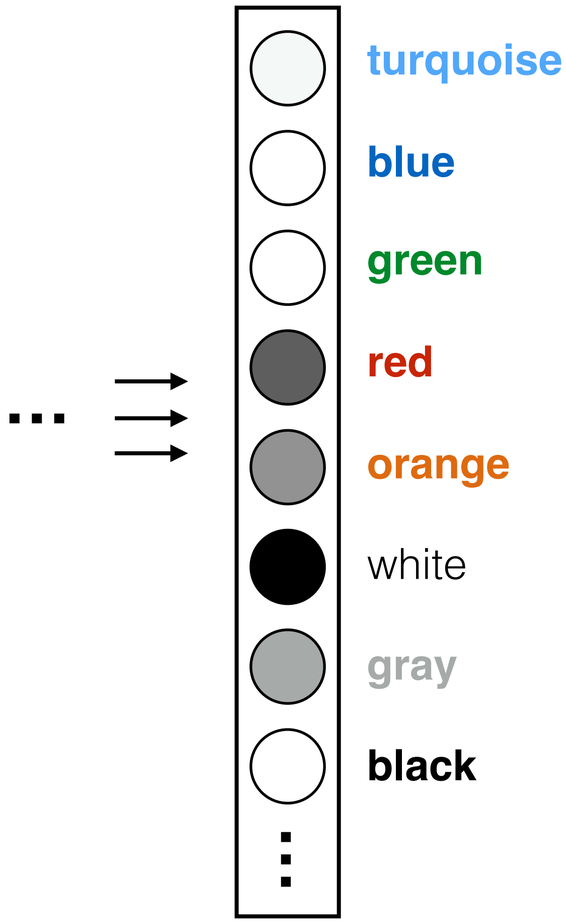


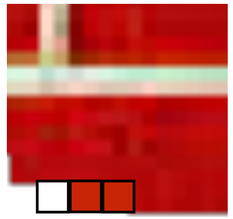
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



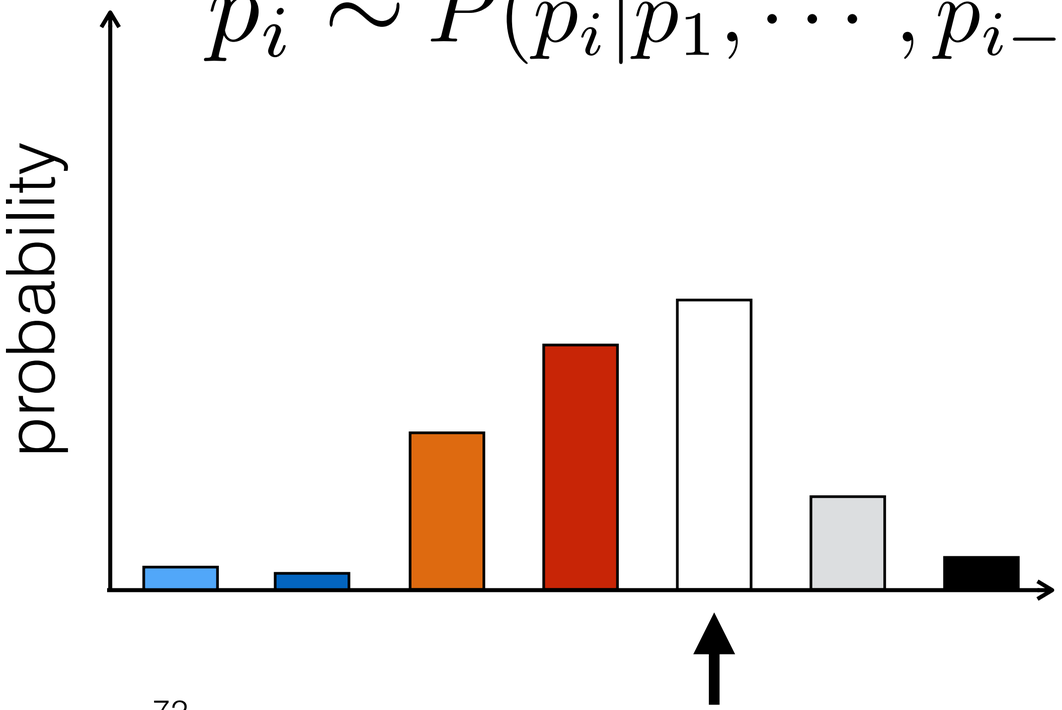
Source: Torralba, Freeman, Isola

Network output



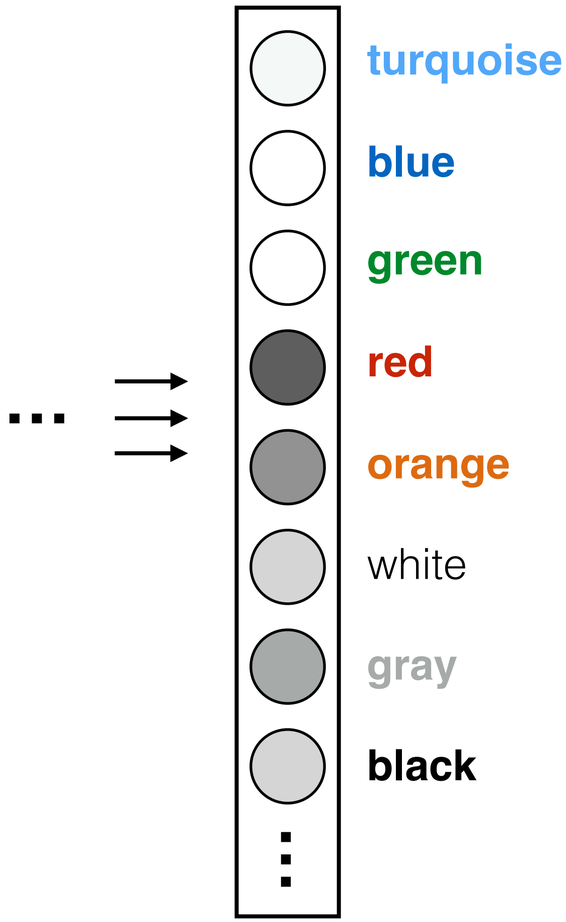


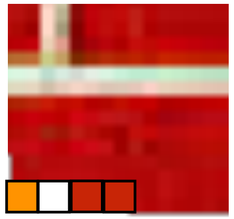
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



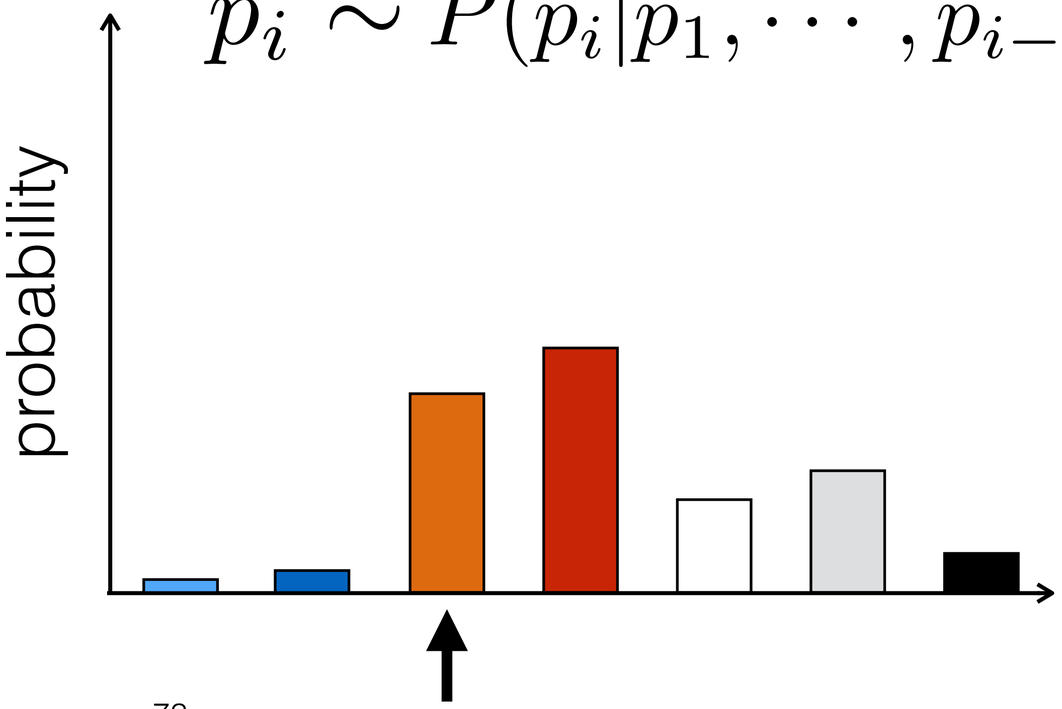
Source: Torralba, Freeman, Isola

Network output





$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



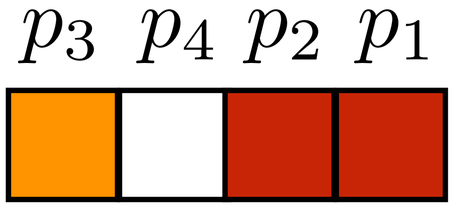
Source: Torralba, Freeman, Isola

$$p_1 \sim P(p_1)$$

$$p_2 \sim P(p_2|p_1)$$

$$p_3 \sim P(p_3|p_1, p_2)$$

$$p_4 \sim P(p_4|p_1, p_2, p_3)$$



$$\{p_1, p_2, p_3, p_4\} \sim P(p_4|p_1, p_2, p_3)P(p_3|p_1, p_2)P(p_2|p_1)P(p_1)$$

$$p_i \sim P(p_i|p_1, \ldots, p_{i-1})$$

$$\mathbf{p} \sim \prod_{i=1}^{N} P(p_i|p_1, \ldots, p_{i-1})$$

Source: Torralba, Freeman, Isola

Autoregressive probability model

$$\mathbf{p} \sim \prod_{i=1}^{N} P(p_i | p_1, \ldots, p_{i-1})$$

$$P(\mathbf{p}) = \prod_{i=1}^{N} P(p_i | p_1, \ldots, p_{i-1}) \quad \longleftarrow \quad \textbf{General product rule}$$

The sampling procedure we defined above takes exact samples from the learned probability distribution (pmf).

Multiplying all conditionals evaluates the probability of a full joint configuration of pixels.

Source: Torralba, Freeman, Isola

Autoregressive probability model

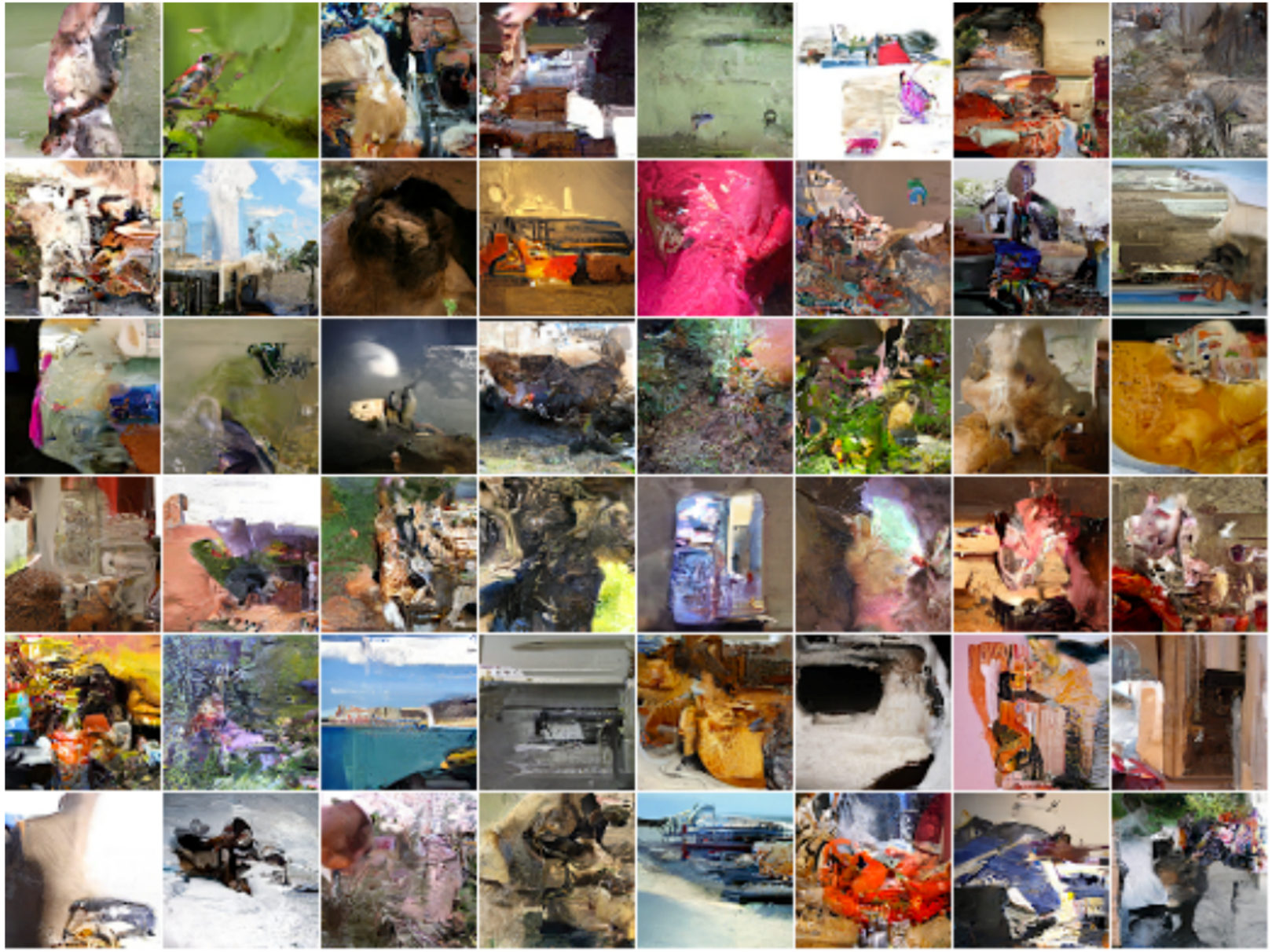
$$\mathbf{p} \sim P(\mathbf{p})$$

Models that allow us to sample, i.e. *generate*, images from scratch are called **generative models**.

We will see more examples in a future lecture.

Source: Torralba, Freeman, Isola

Samples from PixelRNN



[PixelRNN, van der Oord et al. 2016]

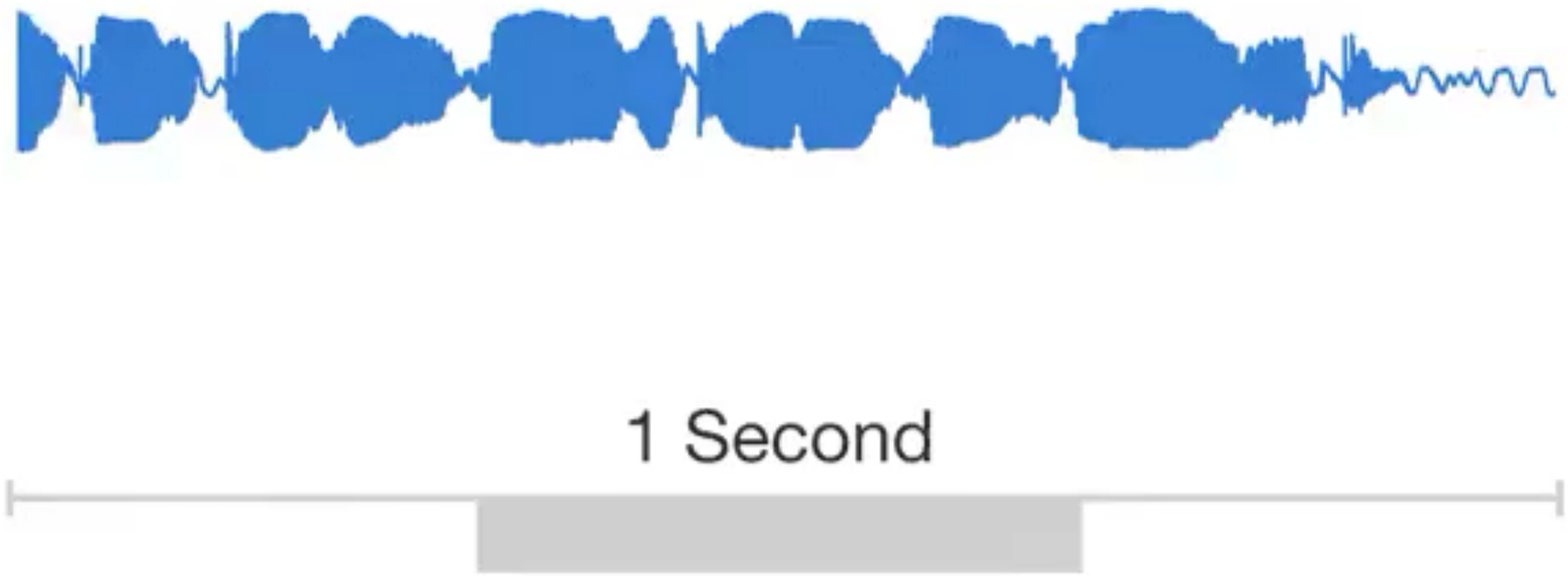
Source: Torralba, Freeman, Isola

Image completions (conditional samples) from PixelRNN

[PixelRNN, van der Oord et al. 2016]

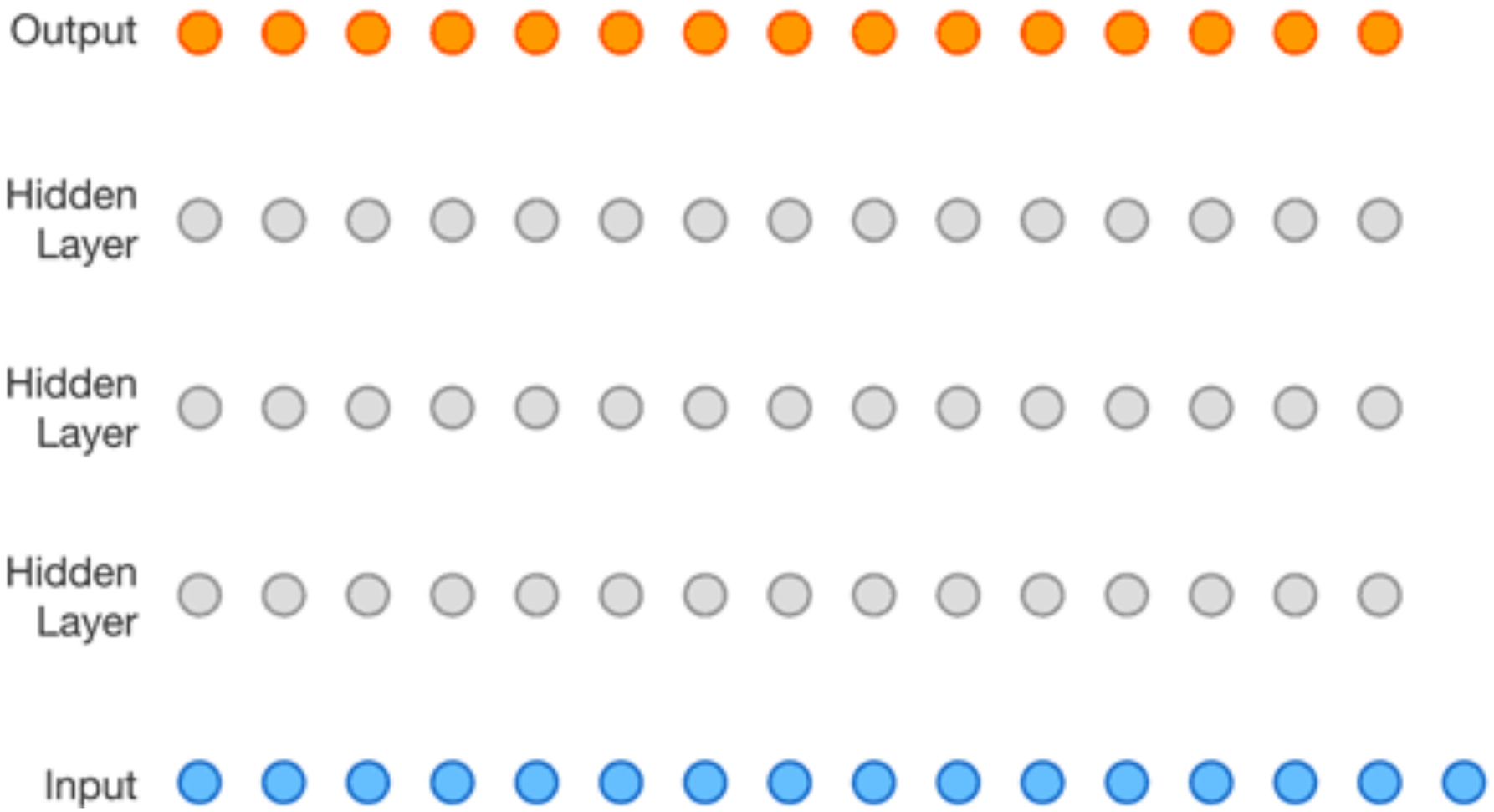
Source: Torralba, Freeman, Isola

Synthesize audio with the same model



[**Wavenet**, https://deepmind.com/blog/wavenet-generative-model-raw-audio/]

Synthesize audio with the same model



Text-to-speech

[**Wavenet**, https://deepmind.com/blog/wavenet-generative-model-raw-audio/]

Source: Torralba, Freeman, Isola

Next week: more image synthesis