Wulfram Gerstner
EPFL, Lausanne, Switzerland

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

Objectives for today:

- Why are sequences important?
- Long-term dependencies in sequence data
- Sequence processing with feedforward models
- Sequence processing with recurrent models
- Vanishing Gradient Problem
- Long-Short-Term Memory (LSTM)
- Application: Music generation

Reading for this lecture:

Goodfellow et al.,**2016** *Deep Learning*

- **Ch. 10** (except 10.6 and 10.8)

Further Reading for this Lecture:

Paper:

- F.A. Gers and J. Schmidhuber and F. Cummins (2000)
  Learning to Forget: Continual Prediction with LSTM
  Neural Computation, 12, 2451–2471
- Xu et al. (2015),
  Show, attend and tell: Neural image caption generation…, ICML

review: Artificial Neural Networks for classification

Given: Training data set

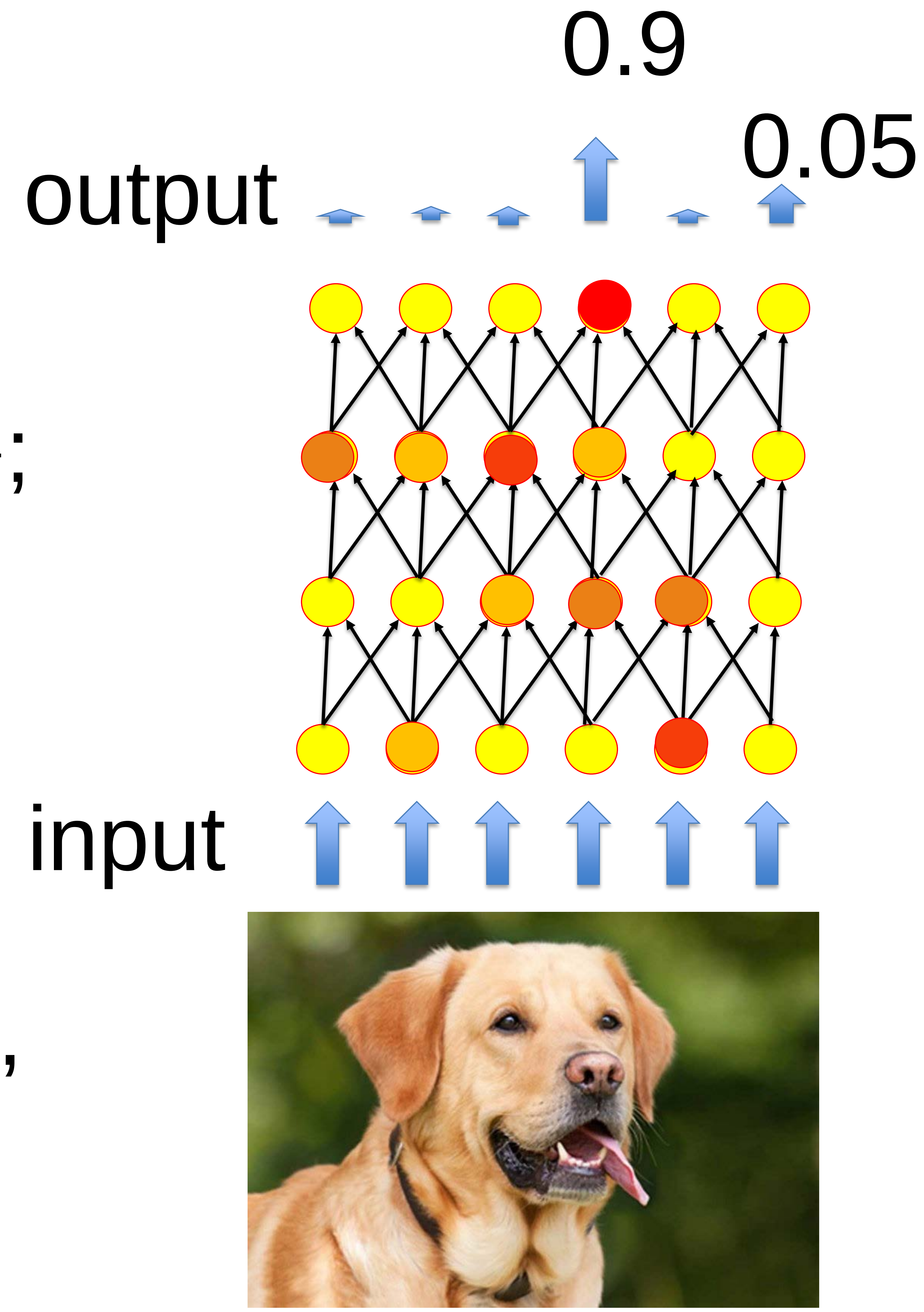
$$\{ \quad (\boldsymbol{x}^{\mu}, \boldsymbol{t}^{\mu}) \quad , \qquad 1 \leq \mu \leq P \quad \};$$

Aim of learning:

Adjust connections such that output $\boldsymbol{y}^{\mu}$ is correct

$$\boldsymbol{y}^{\mu} = \boldsymbol{t}^{\mu}$$

(for each static input image, $\boldsymbol{x}^{\mu}$)



review: Artificial Neural Networks for classification

Given: Training data set $\{ \quad (\boldsymbol{x}^{\mu}, t^{\mu}) \quad , \qquad 1 \leq \mu \leq P \quad \};$

Question:
is this really the most frequent situation
in practice ?

No, for several reasons:
- difficult to get the labeled data!
- data is rarely static!

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

1. Sequences

1. Sequences: first example = video sequence

You have seen the past *n* frames, what is the next frame?

'video frame prediction'

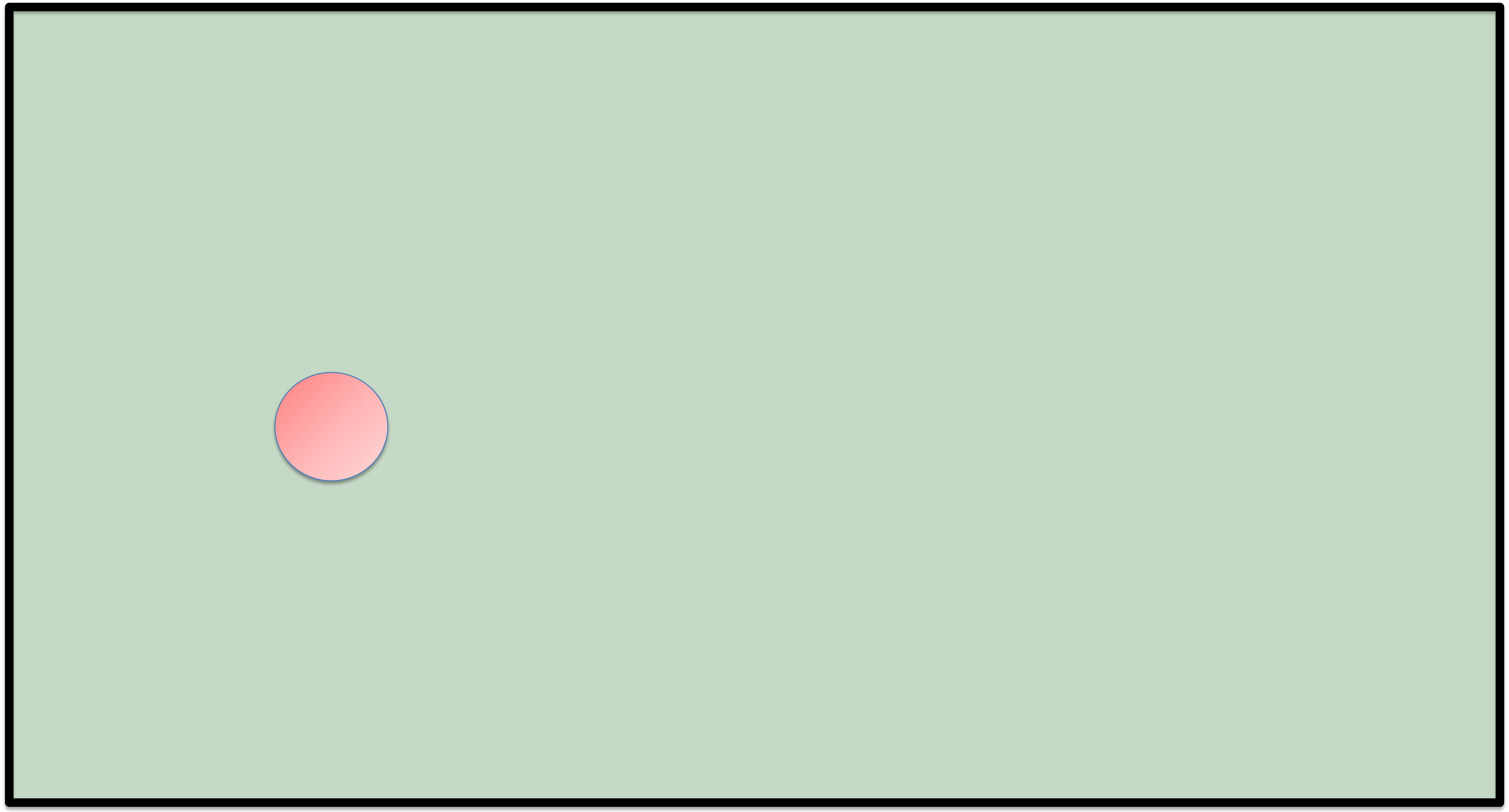
Screenshot from 'Casablanca'

1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

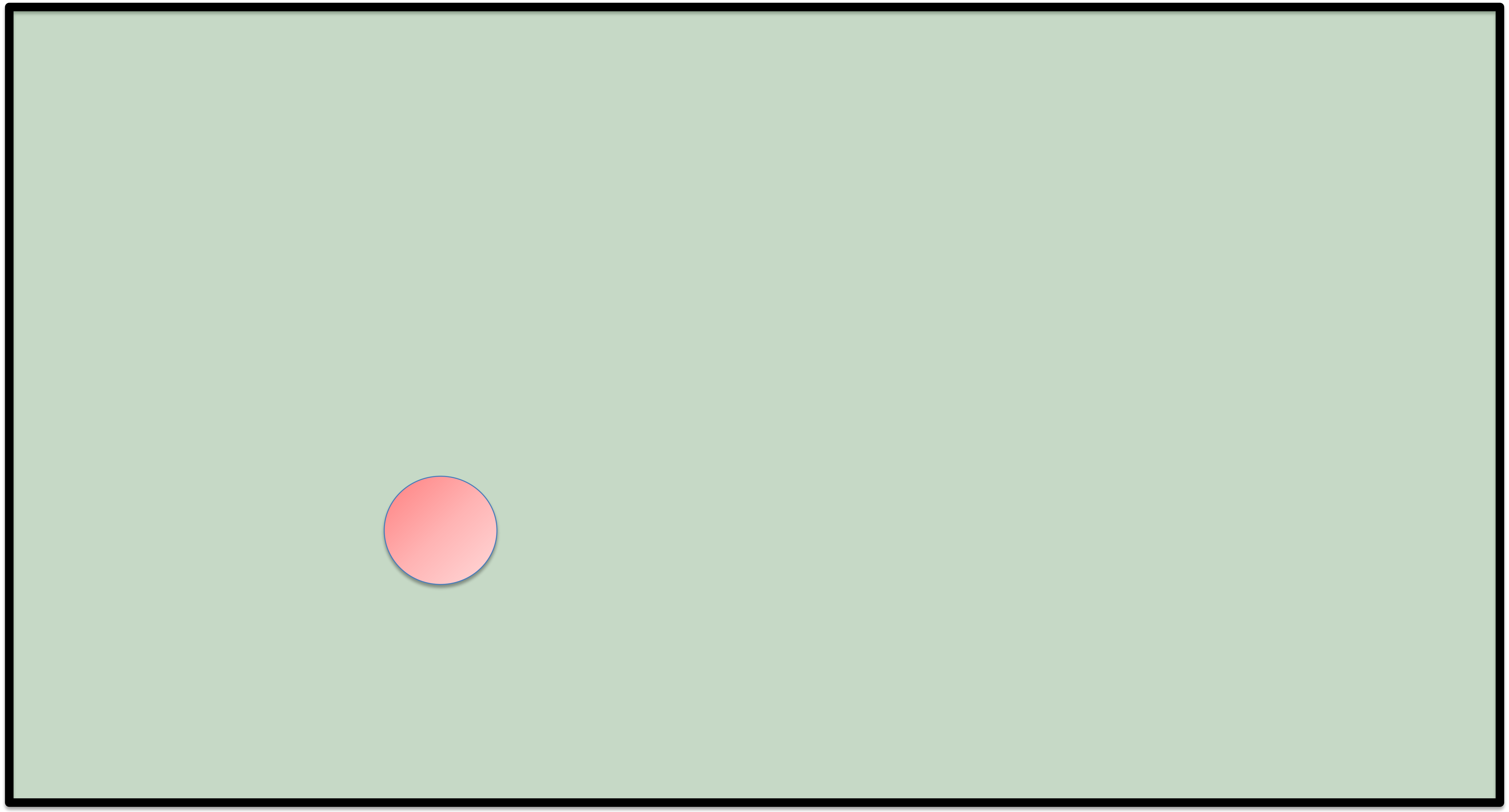
1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



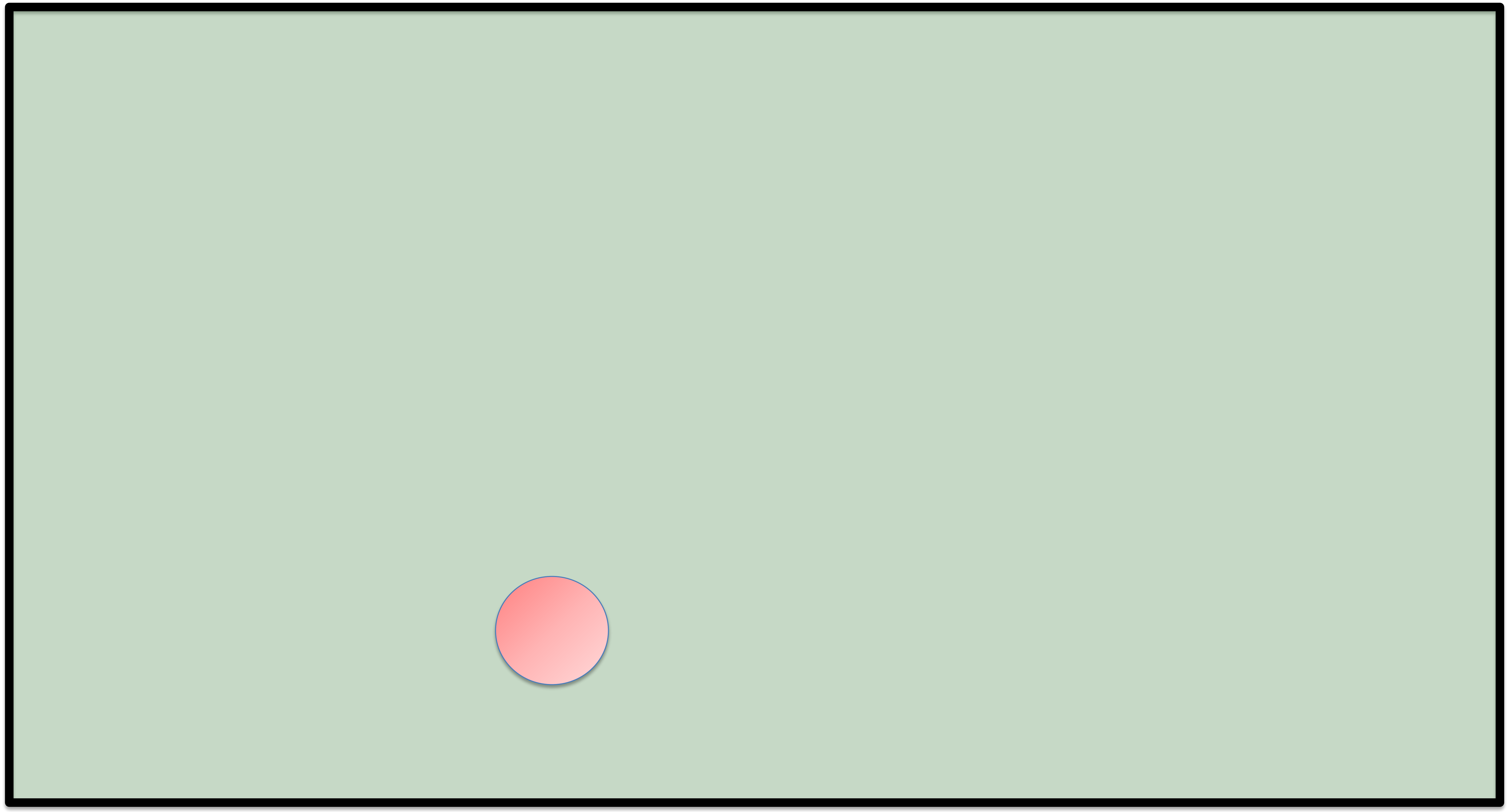
1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



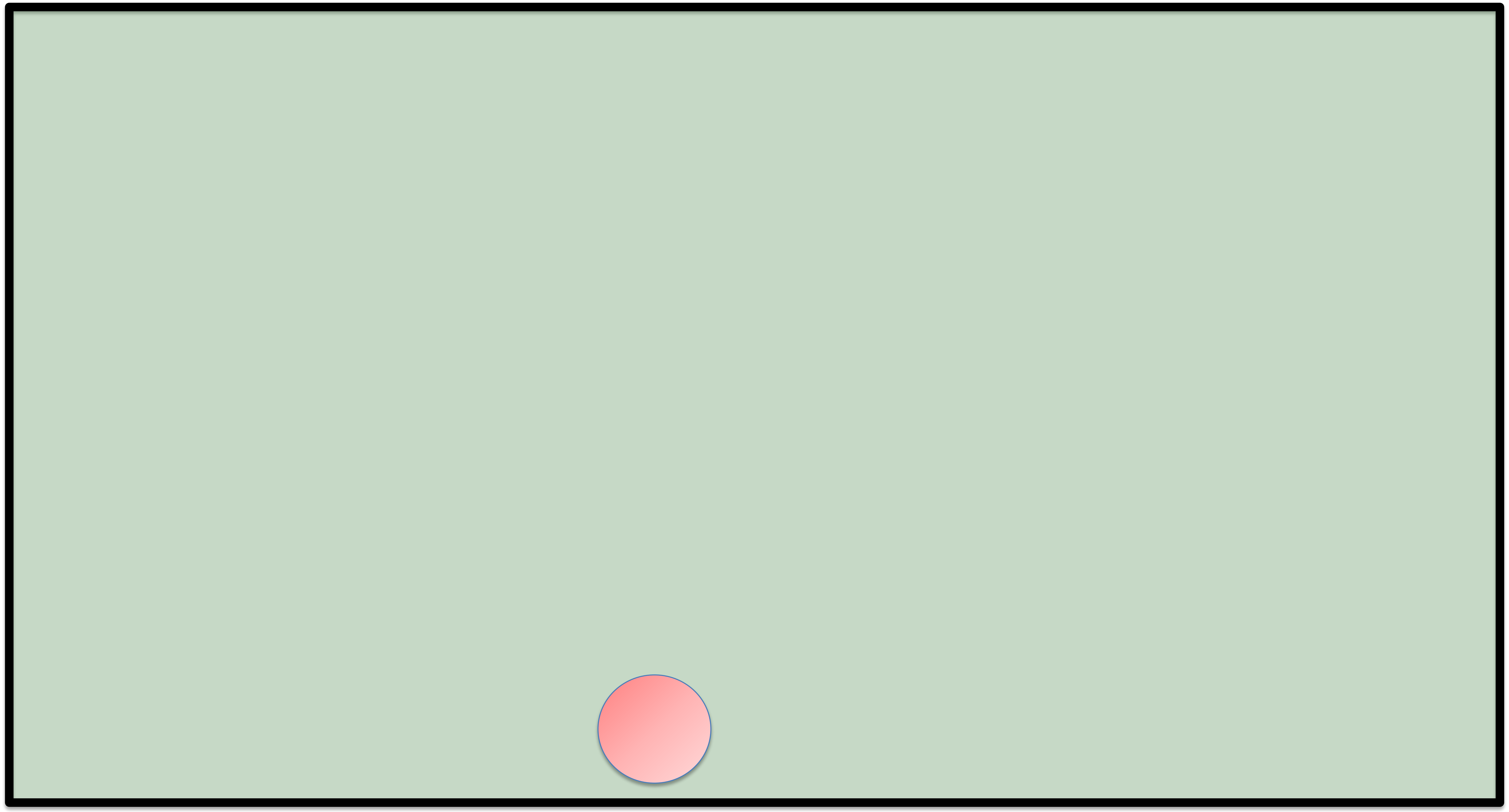
1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



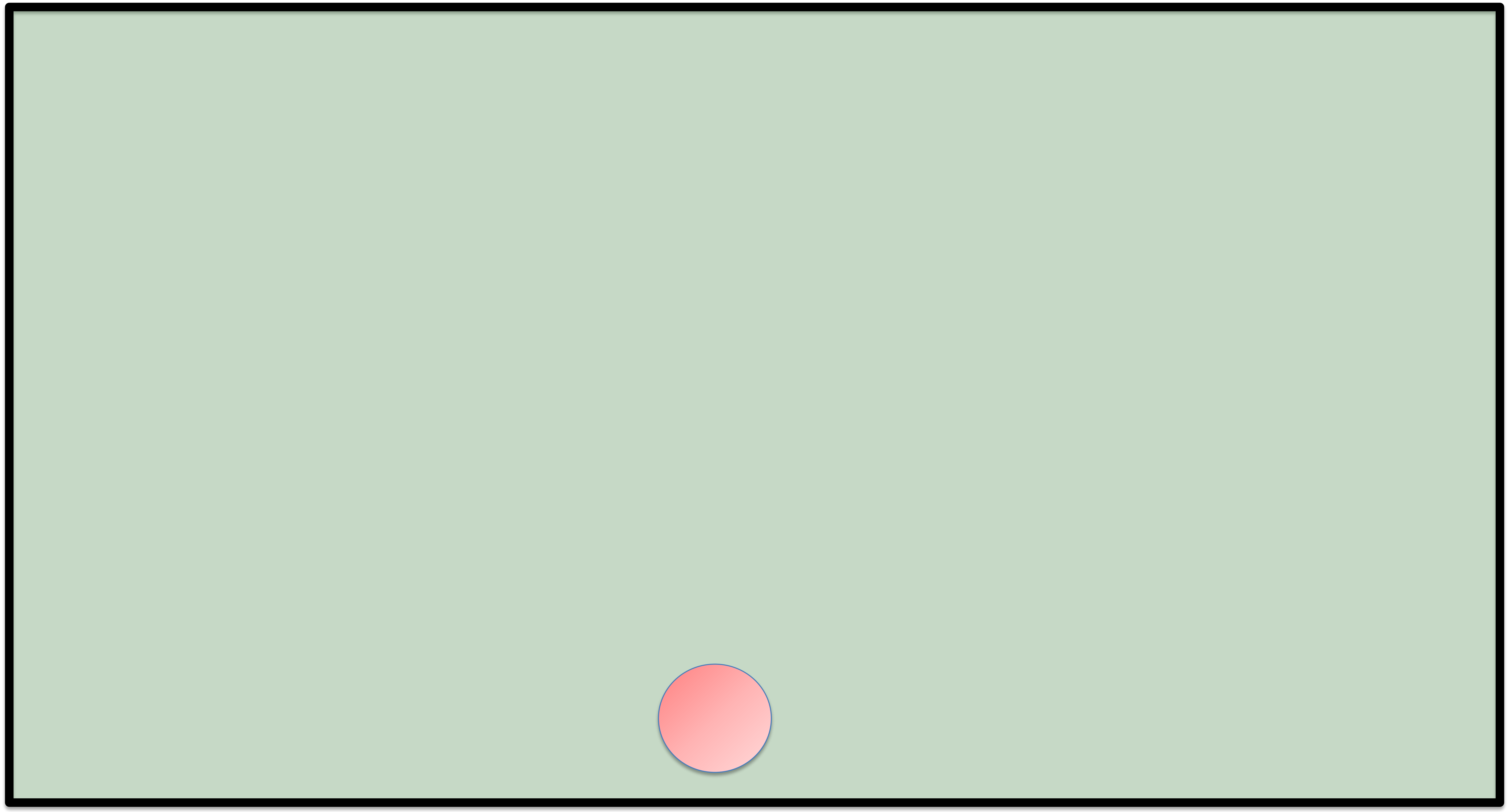
1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



1. Video Sequences/Video Frame Prediction

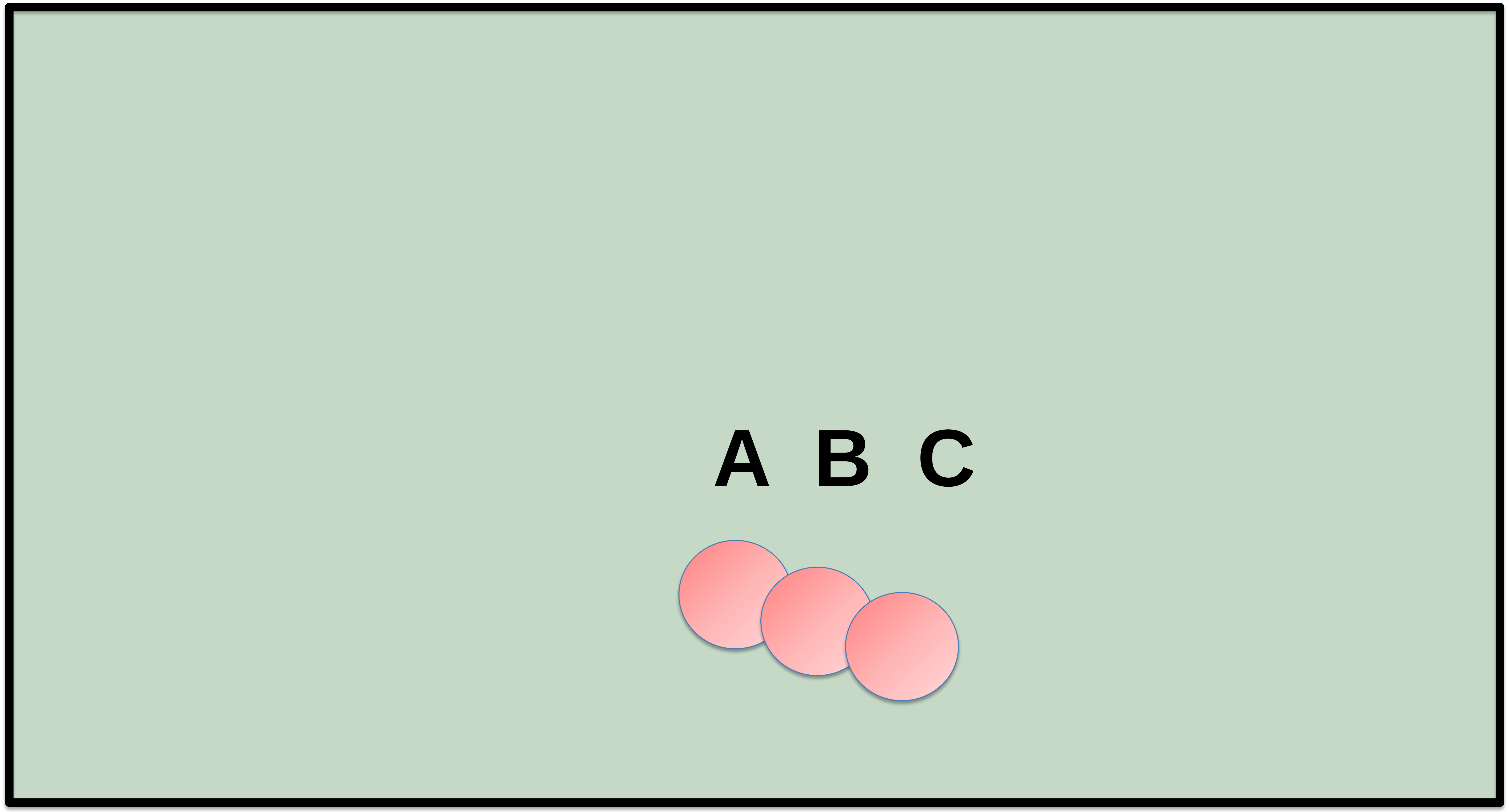
Bouncing Billiard Ball



1. Video Sequences/Video Frame Prediction

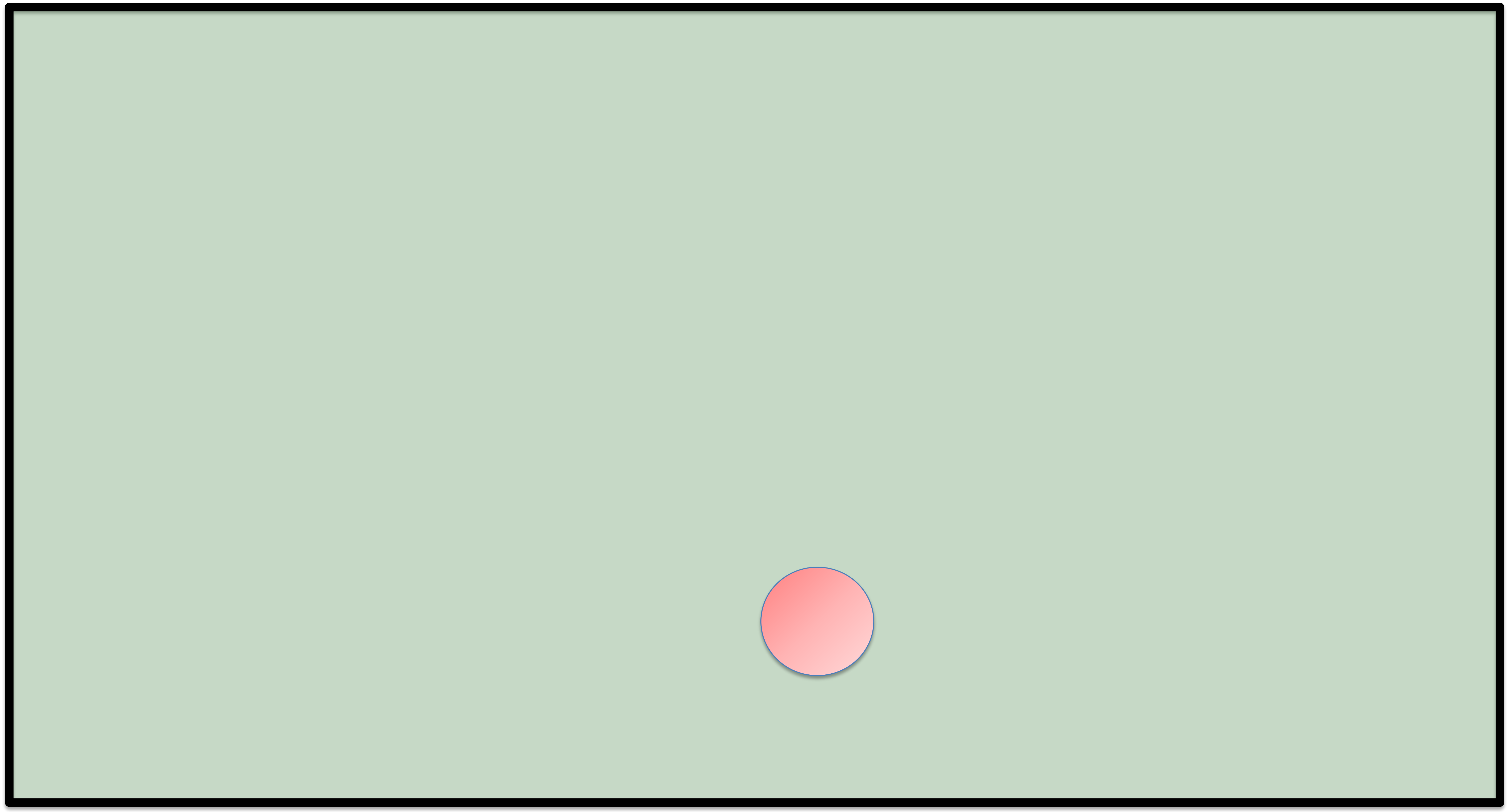
Bouncing Billiard Ball

Predict position in next frame



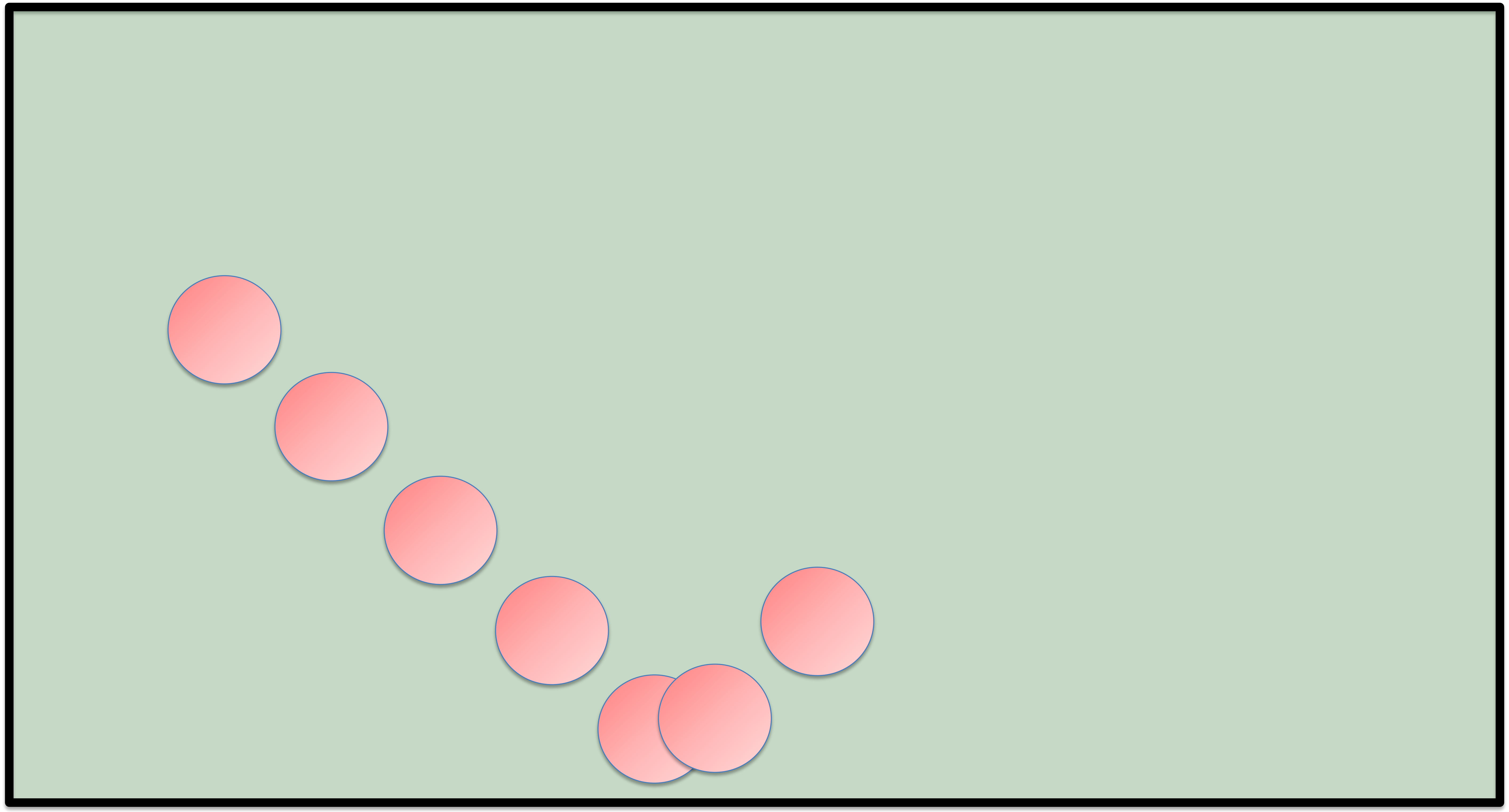
1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



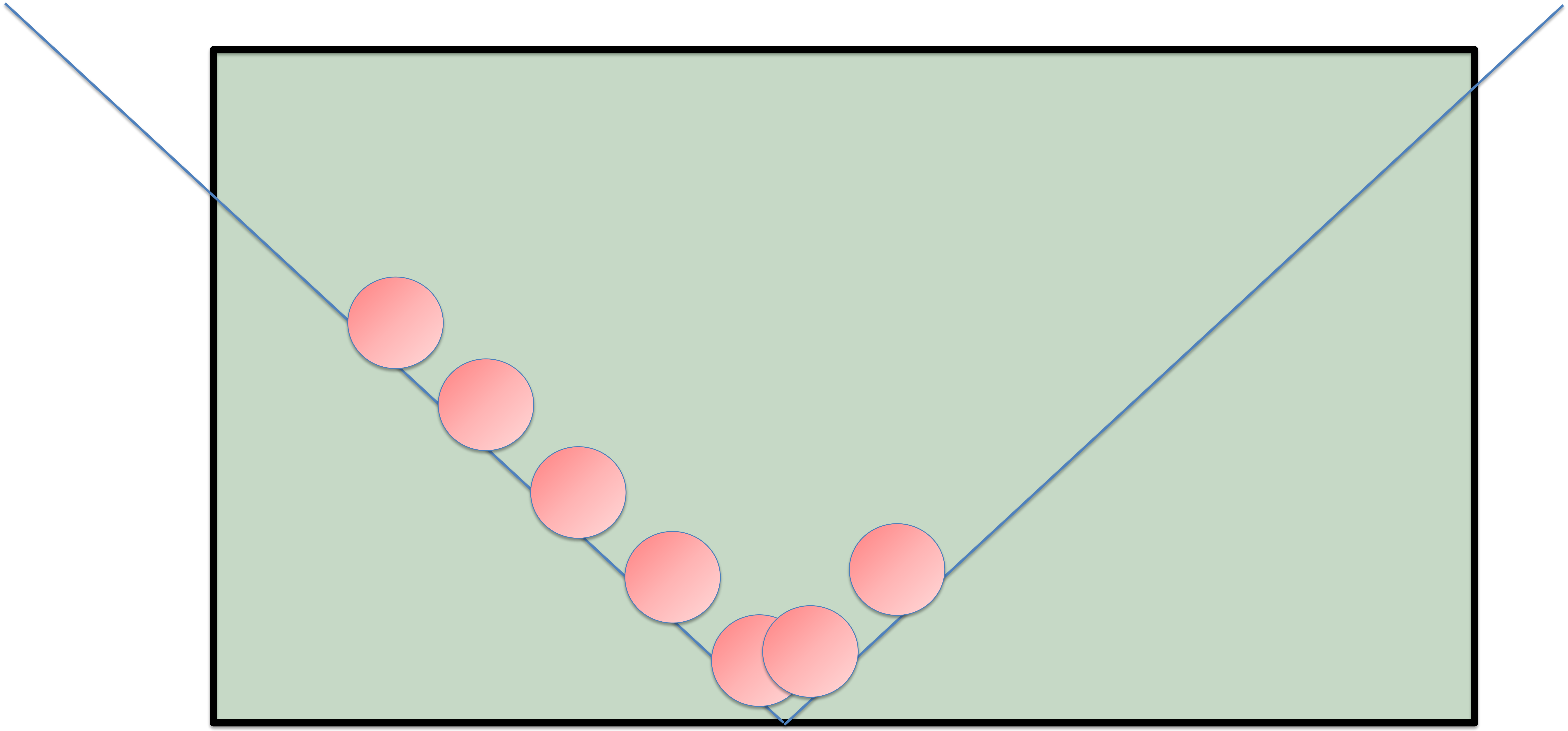
1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



1. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



1^{st} example: video frame prediction

- Target of training is the next frame
  → lots of training data!!!

- Data consists of a temporal sequence,
  prediction needs more than 1 frame in the past
  → not the standard static input scenario

- Output is high-dimensional (pixels in one frame)

1. Sequences

- 1st example: video frame prediction

  Analogous: - move your arm while watching
  - observe movements of your neighbor and predict next move

- 2nd example: text prediction

1. Sequences: 2nd example - text prediction

Similar to Caltech, MIT, and GeorgiaTech which are considered top-level technical universities in the US, TUMunich, ETHZurich and …

1. Sequences: text prediction

2nd example: Text prediction

- Target of training is the next word
  → lots of training data!!!

- Data consists of a temporal sequence,
  prediction needs more than 1 word in the past
  → not the standard static input-output scenario

- Output is high-dimensional
  (ten-thousands of potential words)

1. Sequences

- 1st example: video frame prediction

- 2nd example: text prediction

  analogous: - text translation
  - speech (or phoneme) prediction
  - music prediction

- 3rd example: action planning

1. Sequences: 3rd example – action planning and navigation

- Close your eyes

- Imagine how you would go
  to the library in the ‘learning center’

1. Sequences

Summary:

- Sequences are everywhere

films, text, speech, body movement, action planning, navigation

- more common in reality than static input-output paradigms

We don’t look at static photos in normal live

- target data (needed for supervised learning) is often cheap

e.g., target is next frame in video / next word in text/
next action in movement:
– all easy to observe

1. Sequences: Aim

First Question for today

how can we model and learn sequences in artificial neural networks?

Artificial Neural Networks: Lecture 6

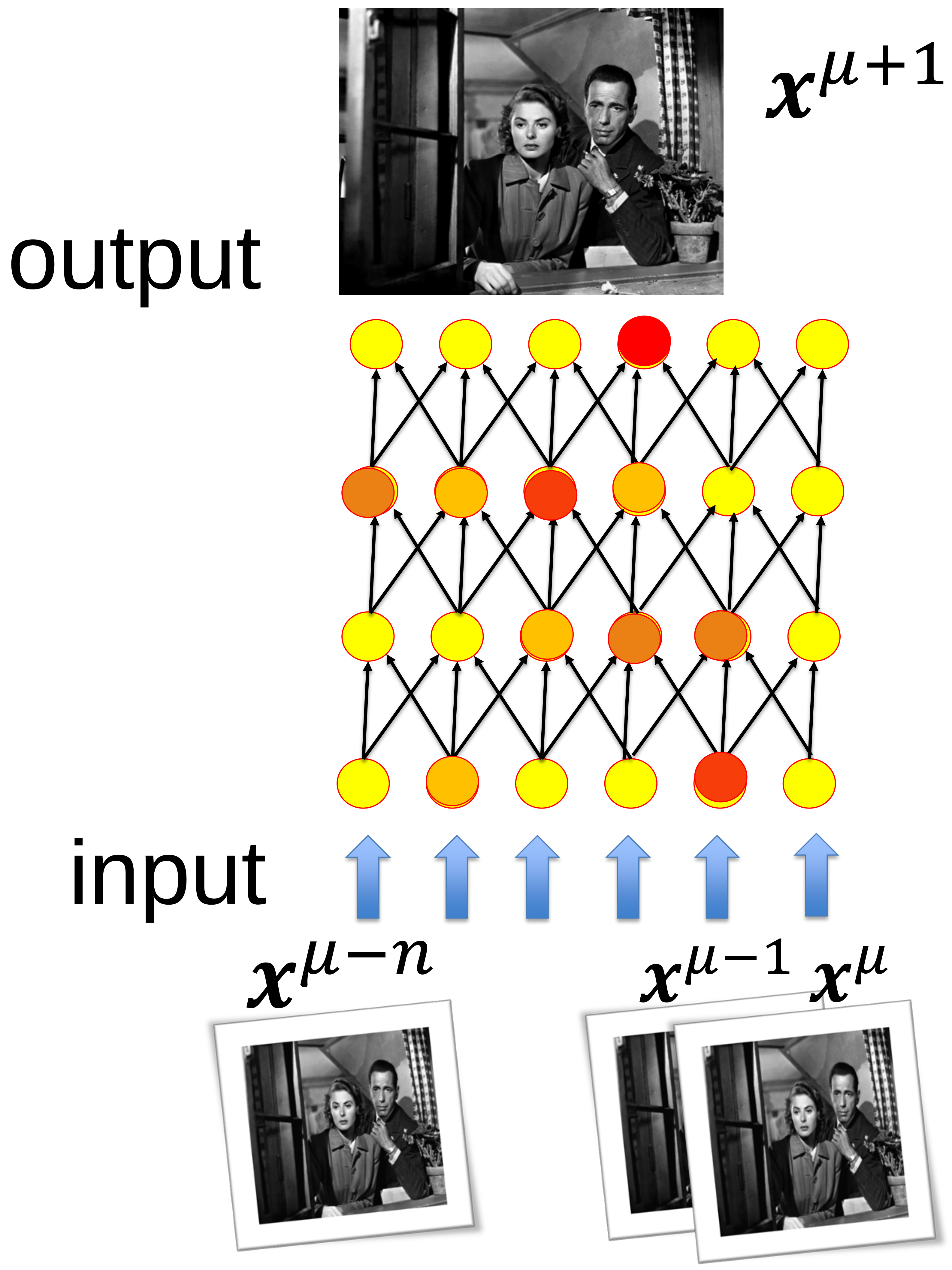
Sequences and Recurrent Networks

1. Sequences
2. Naïve Neural Network implementation:
   increase number of inputs

2. Naïve solution: increase number of inputs

predict next output

take *n* frames as input

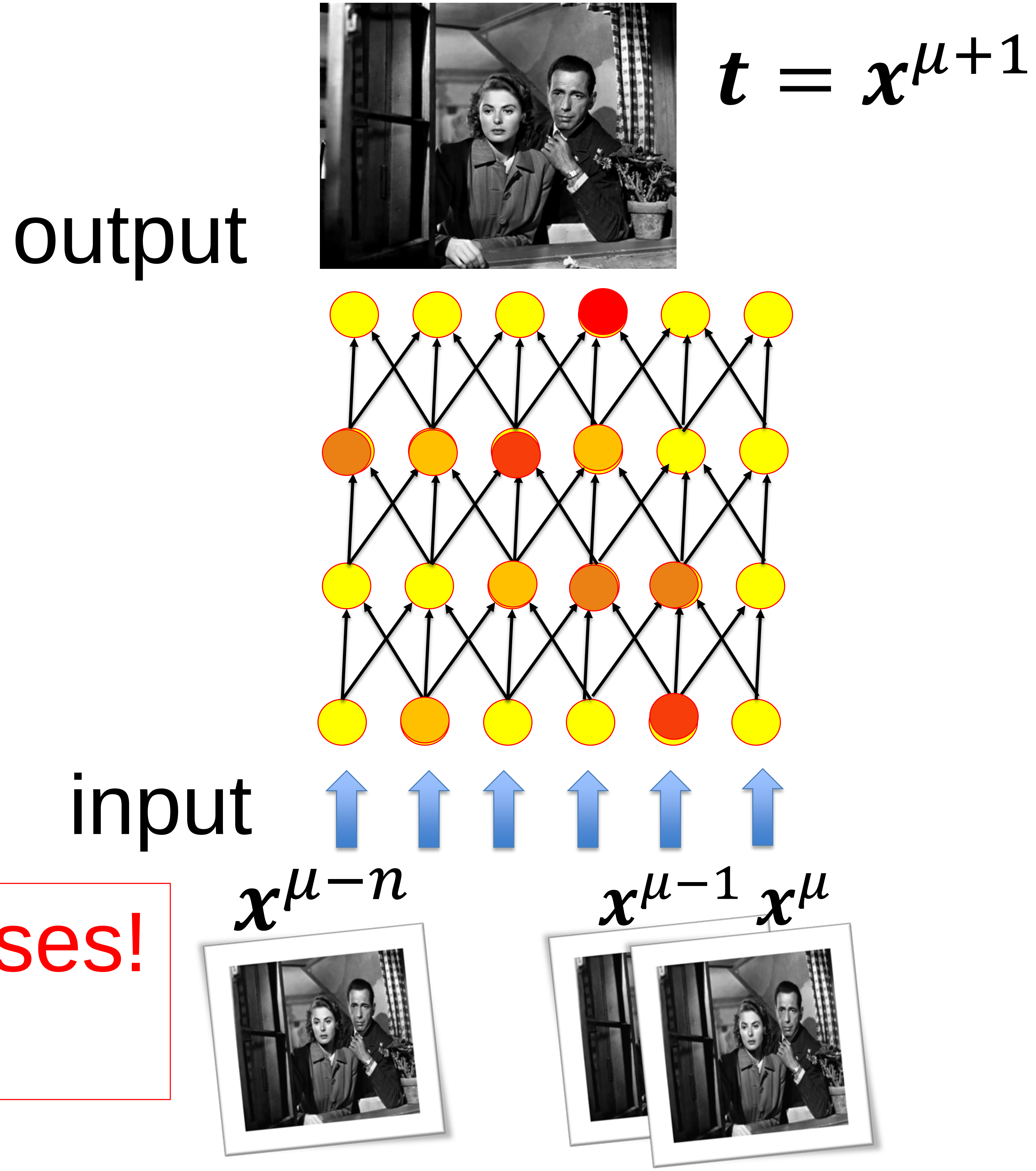


2. Naïve solution: Problems

predict next output

take *n* frames as input

BUT - dimensionality increases!
- what is best *n* ?



2. Naïve solution: Problems

The naïve solution corresponds to implementing **n-grams** with a neural network, but

- dimensionality increases!
- what is best *n* ?

→ What is the relevant time scale?
(number of frames necessary for good prediction)

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

1. Sequences
2. Naïve solution: increase number of inputs
3. Long-term Dependencies

3. Dependencies in Video

Bouncing Billiard Ball

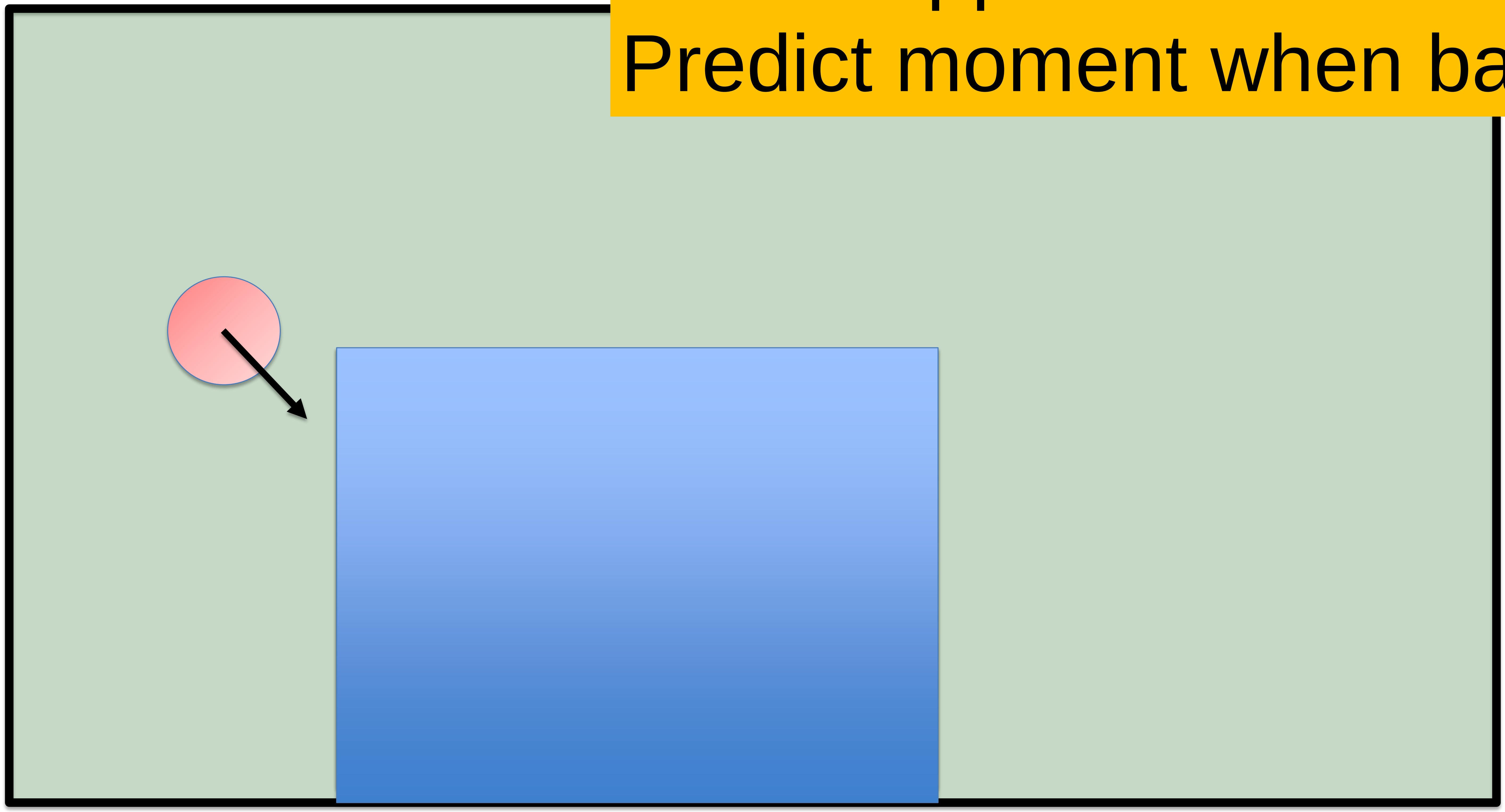
time = 1

3. Dependencies in Video

Bouncing Billiard Ball

Ball disappears behind blue screen.
Predict moment when ball **reappears**!

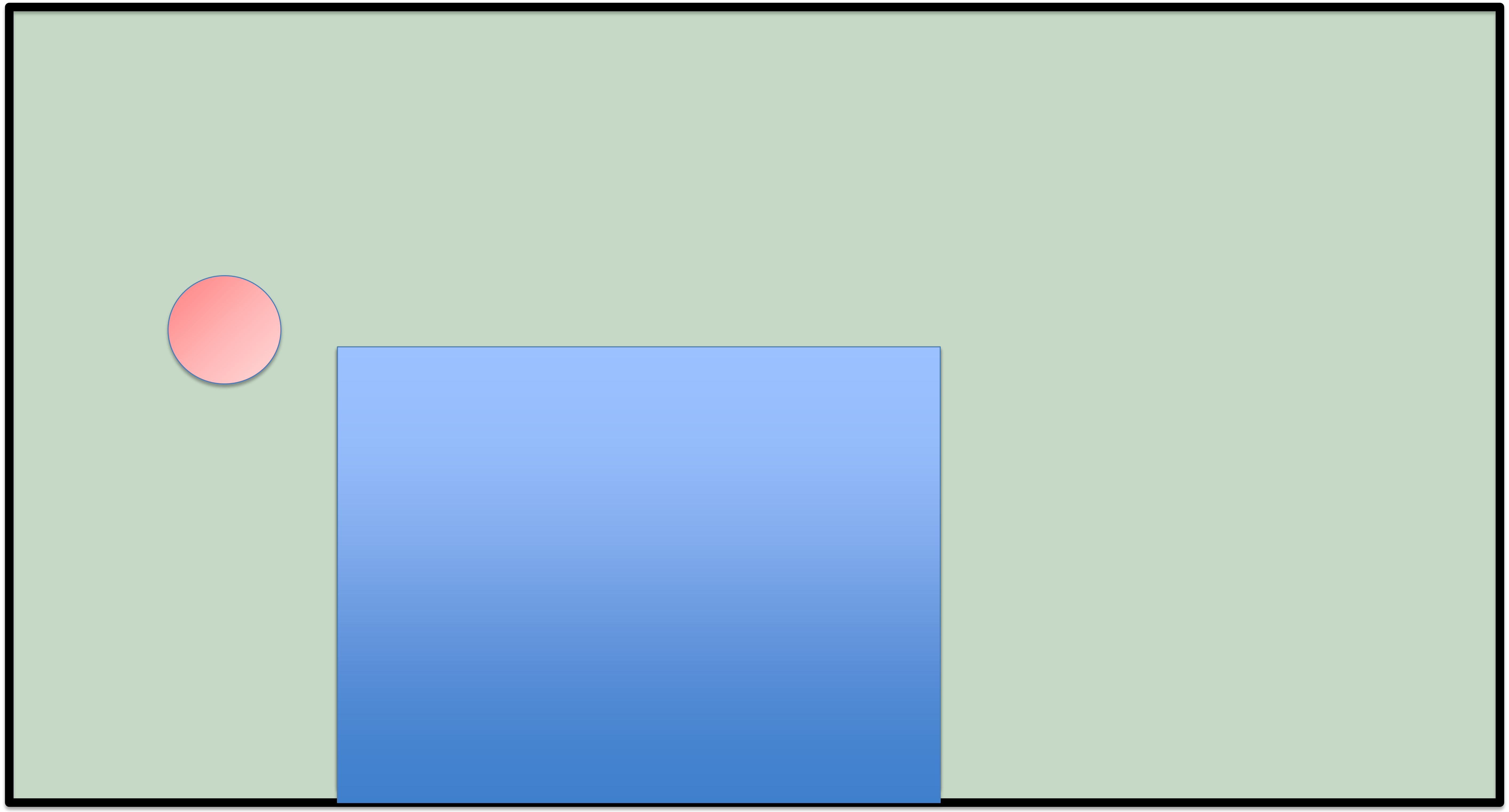
time = 1



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

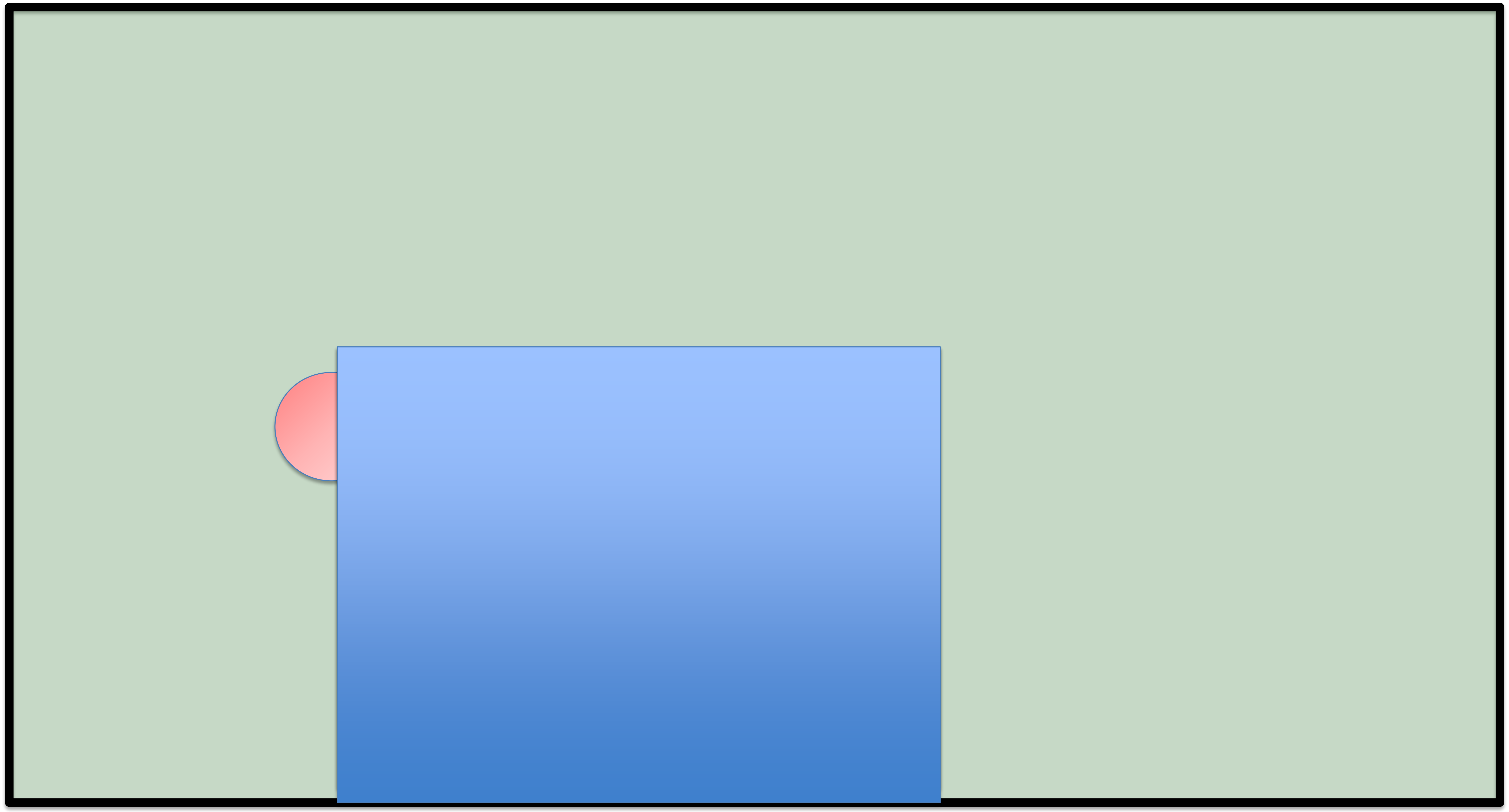
time = 1



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

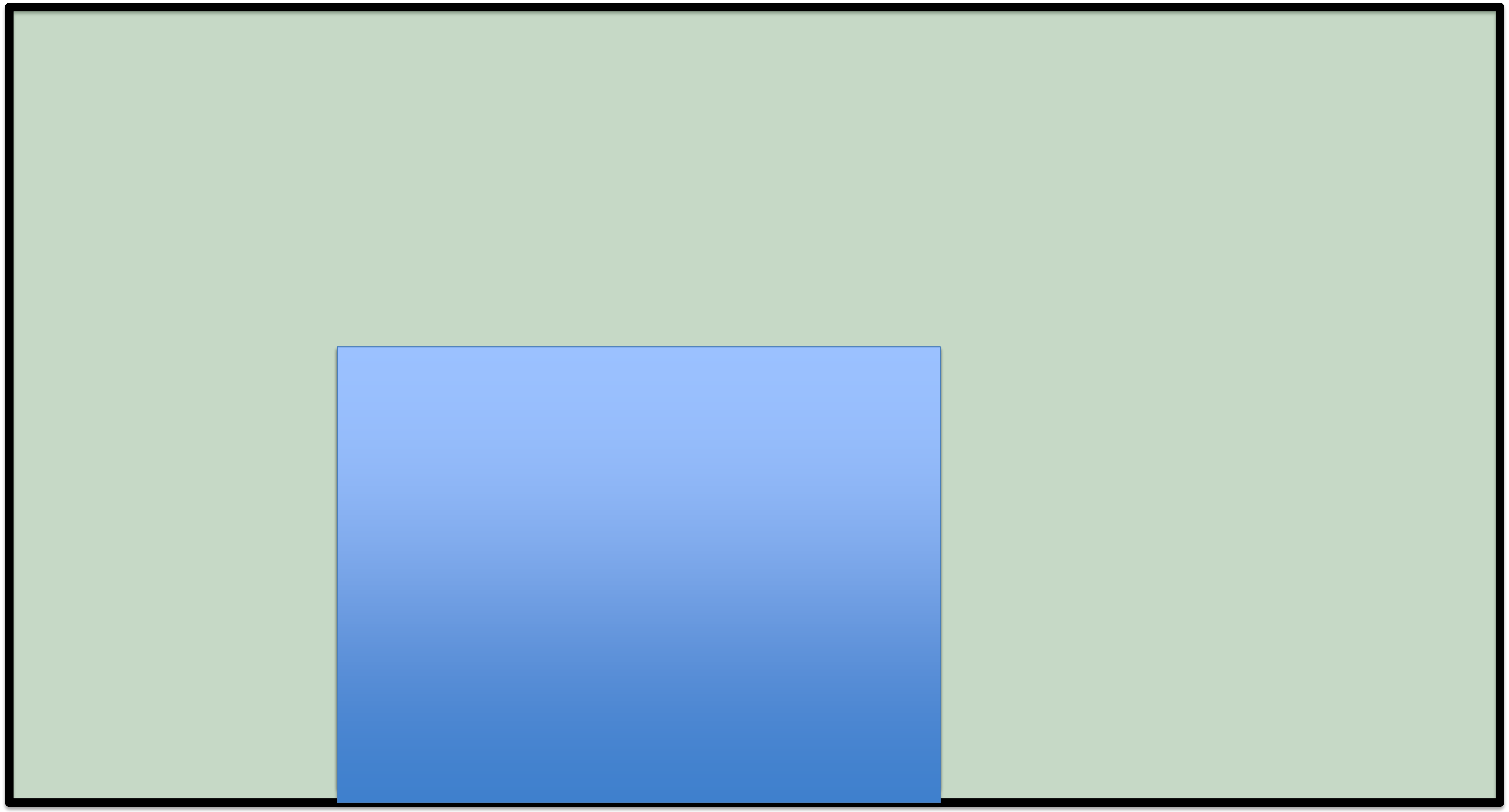
time = 2



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

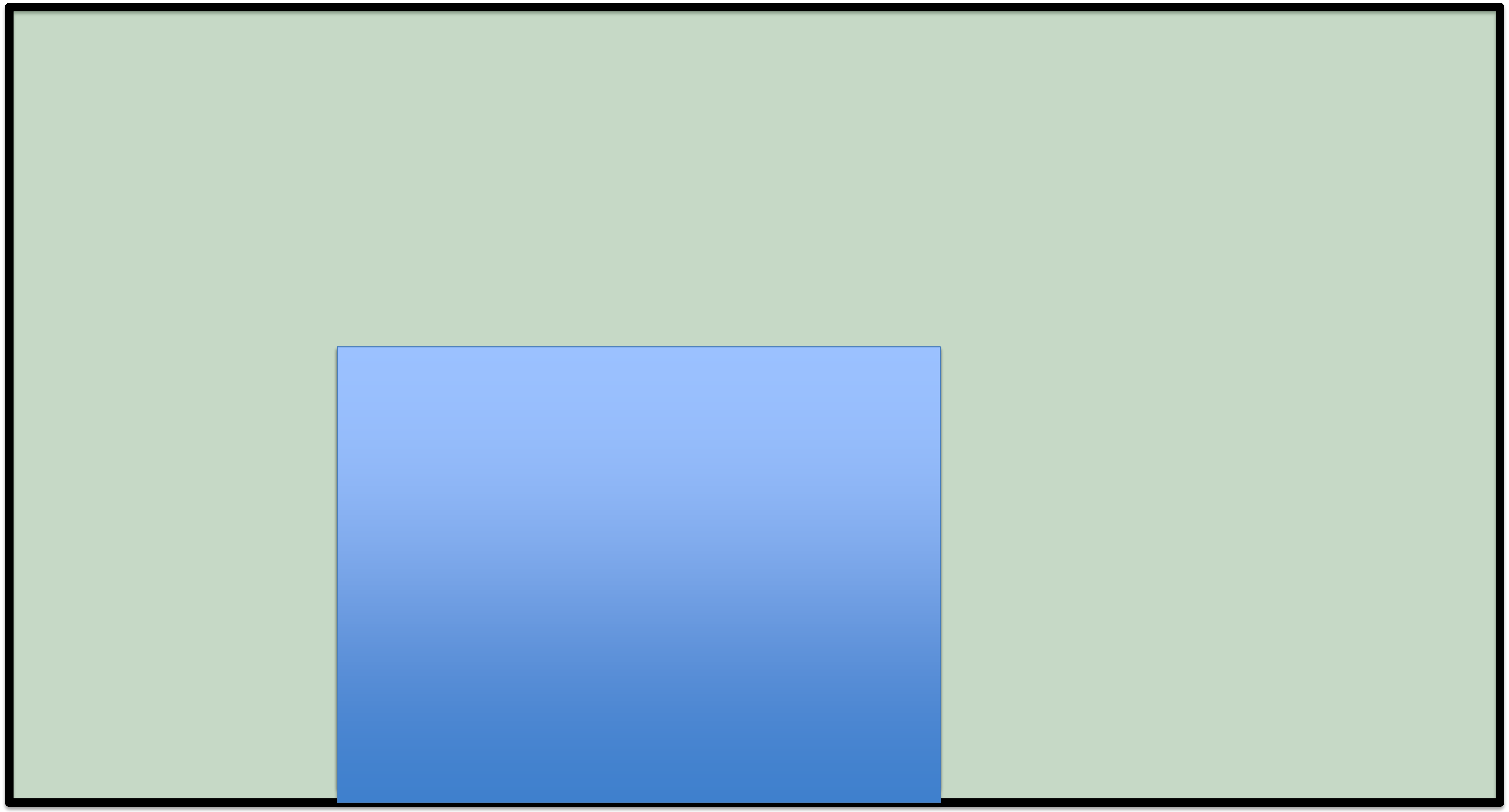
time = 3



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

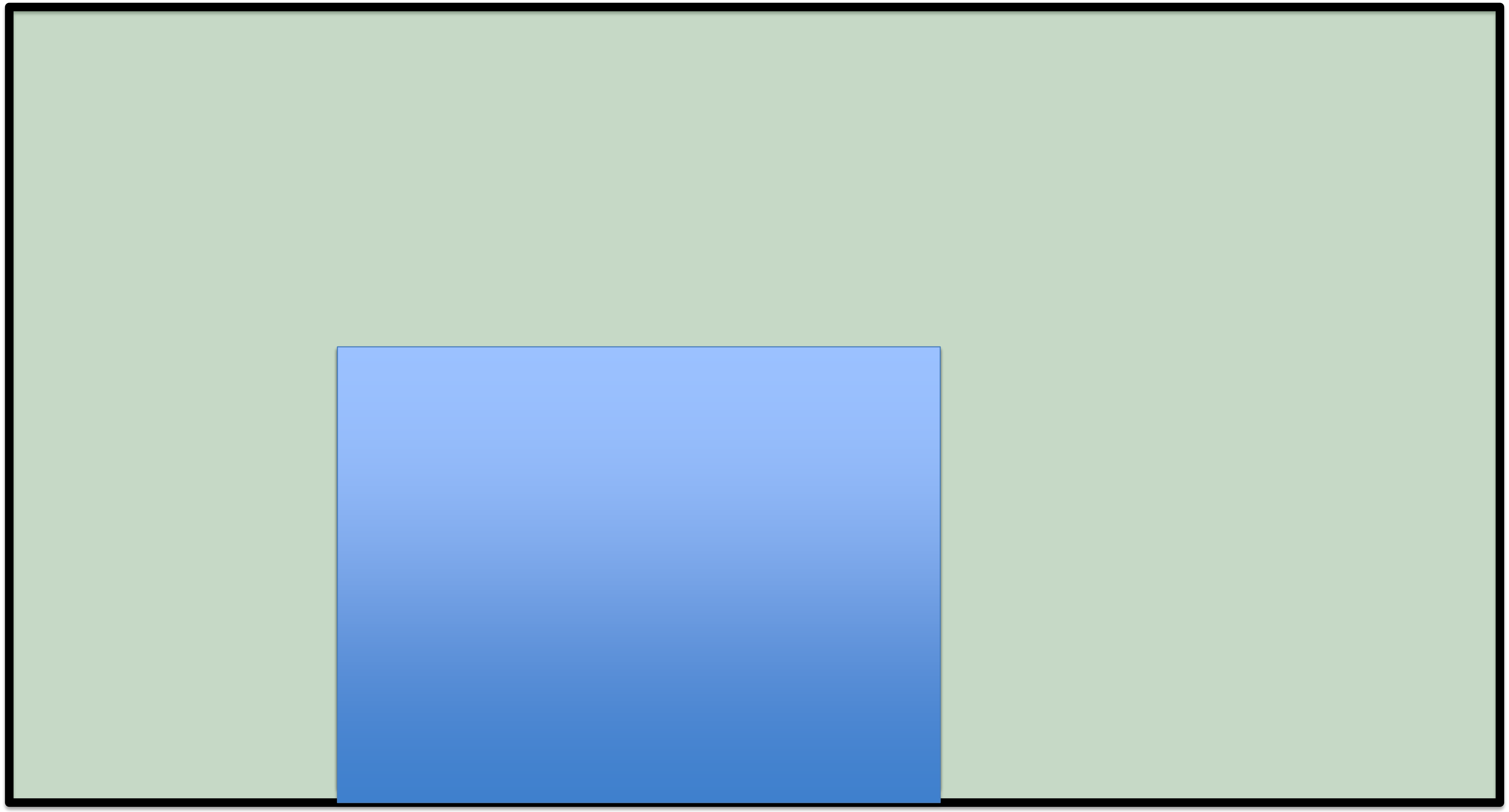
time = 4



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

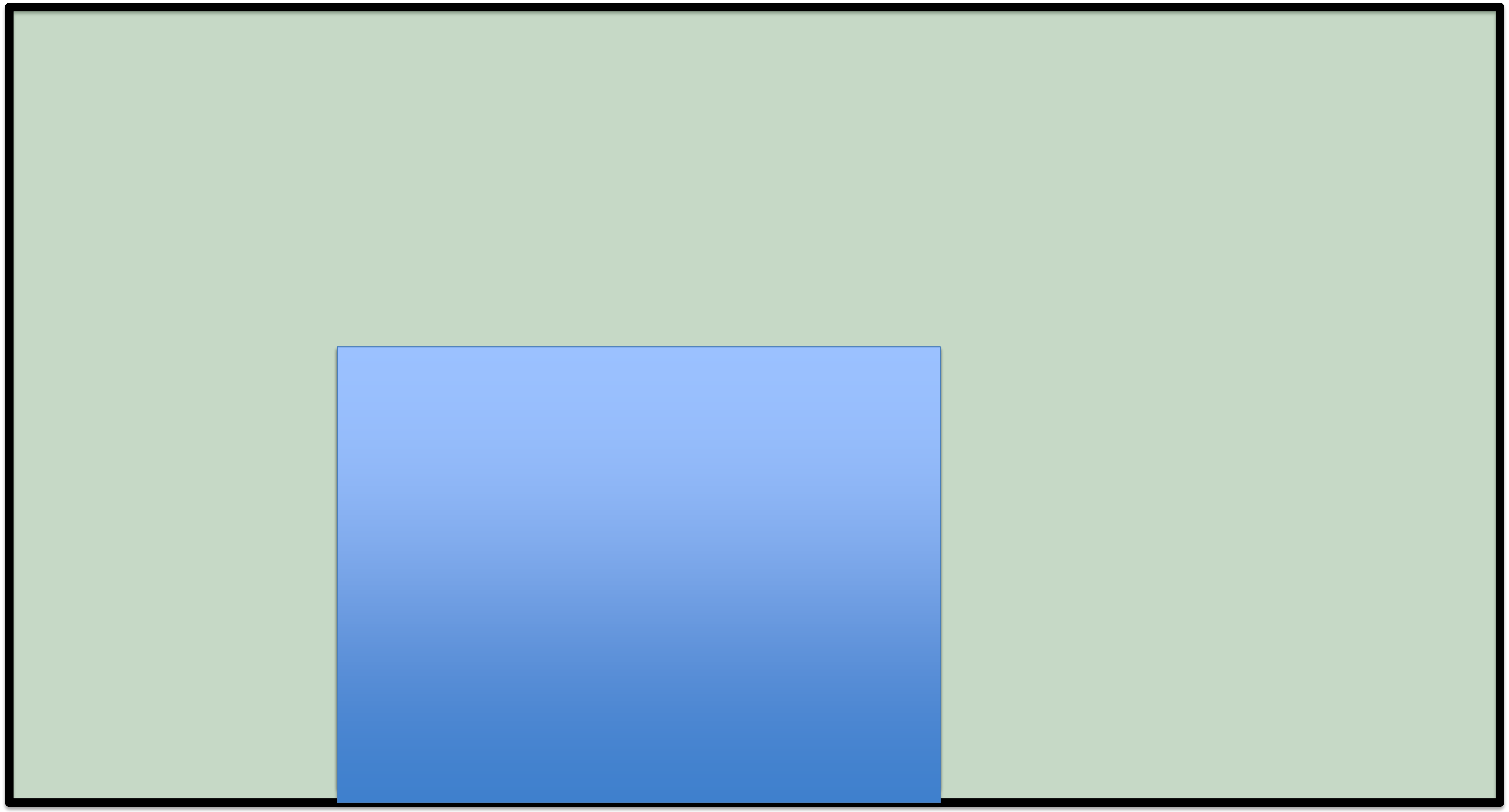
time = 5



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

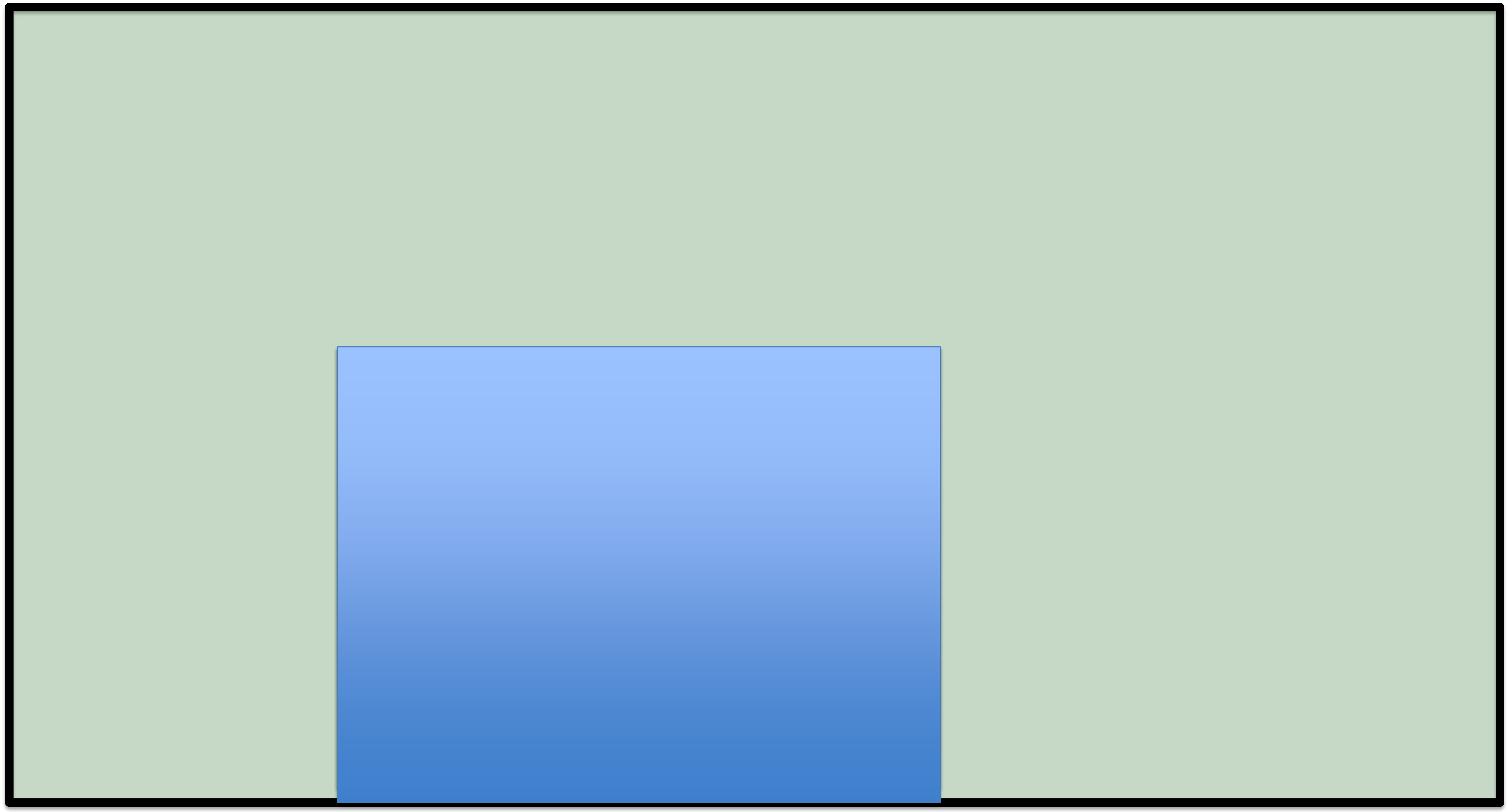
time = 6



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball

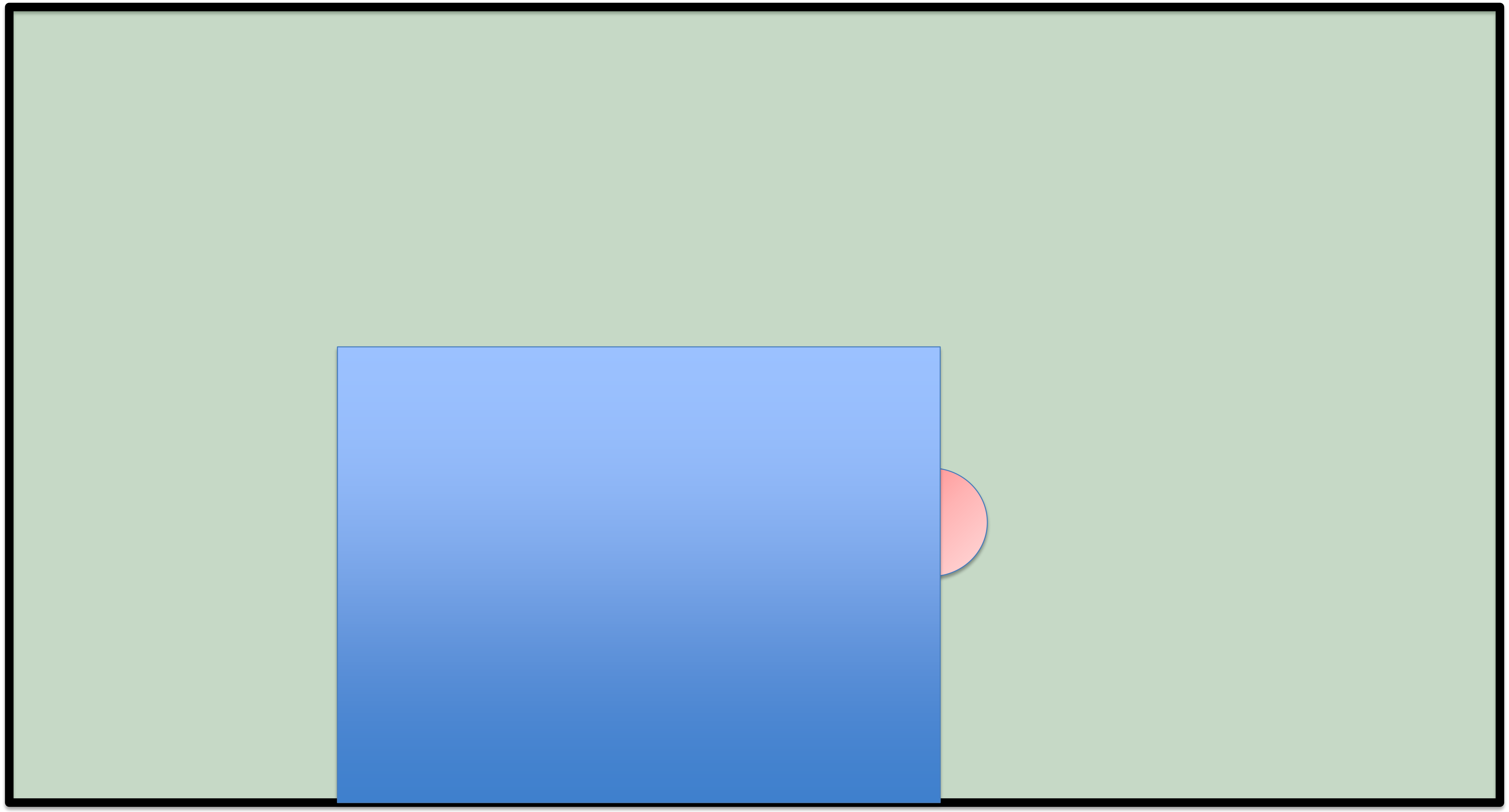
time = 7



3. Video Sequences/Video Frame Prediction

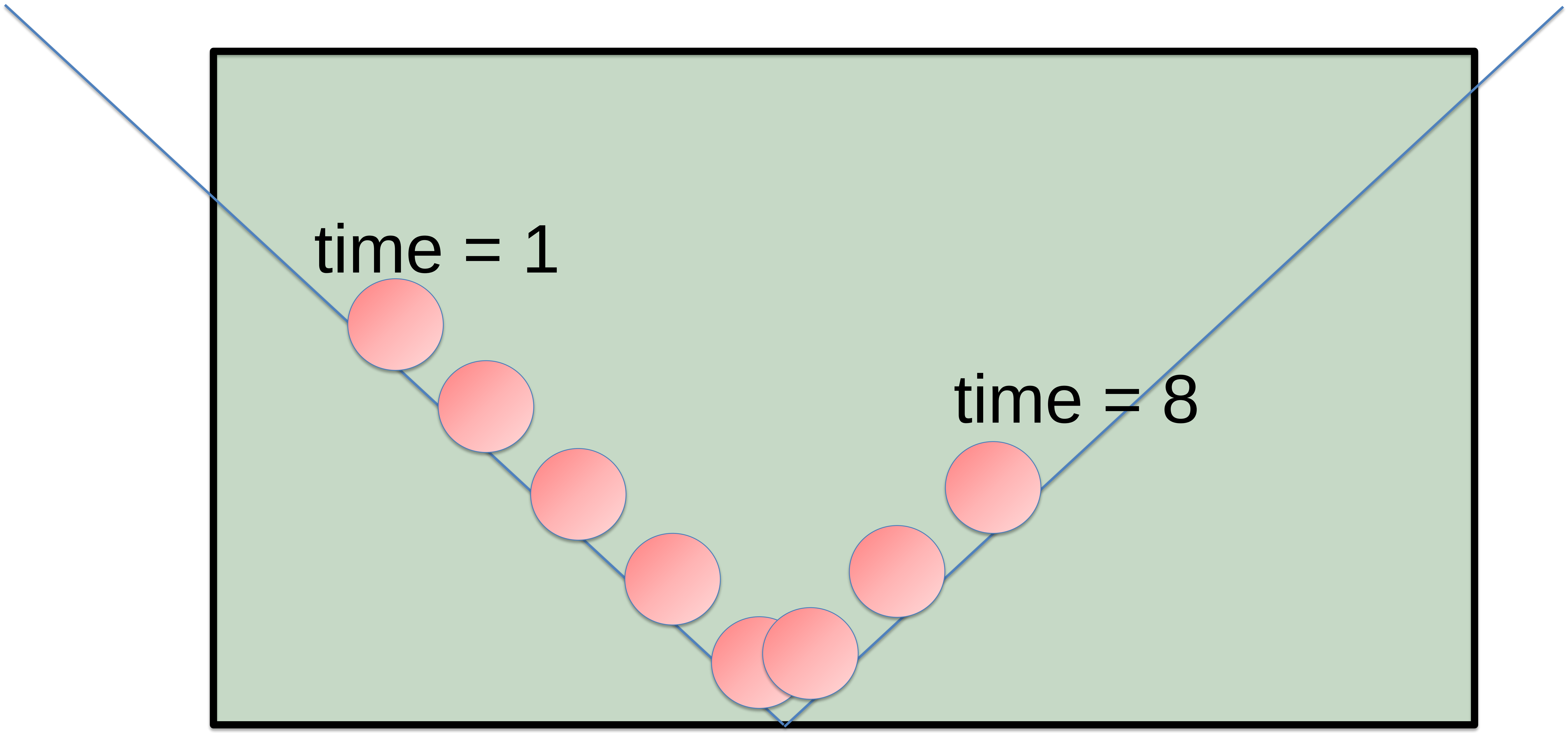
Bouncing Billiard Ball

time = 8



3. Video Sequences/Video Frame Prediction

Bouncing Billiard Ball



3. Long-term dependencies in sequences

1st example: video frame prediction -
you potentially need a memory over MANY frames!

Extreme example:
- memory over a whole story, since entrance scene turns out to be important to predict the end
  → long time scale!!!
- but movements within one scene are on a fast time scale

Example: Actor with red shoes

→ You never know in advance how many frames you need
→ There might be several relevant time scales!

3. Long-term dependencies in sequences

1st example: video frame prediction

2nd example: text prediction and text translation

3. Long-term dependencies in text sequences

We are in 2013 and hear on the radio:

The international press writes that Mr. Obama who is starting today his second term as president of the United States is praised as one of the most influential world leaders.

We are in 2019 and remember:

In 2013 many international journals wrote that Mr. Obama who was then starting his second term as president of the United States was praised as one of the most influential world leaders.

3. Long-term dependencies in text sequences

Grammar rules create long-term dependencies

The international press writes that Mr. Obama who is starting today his second term as president of the United States is praised by the World Economic Forum as one of the most influential world leaders.

In 2013 many international journals wrote that Mr. Obama who was then starting his second term as president of the United States was praised by World Economic forum as one of the most influential world leaders.

3. Long-term dependencies in text sequences

Grammar rules create long-term dependencies

→ important for text translation

3. Long-term dependencies in text sequences

Ambiguities:

Tank as army vehicle
Tank as liquid container

Question: how can we disambiguate?

3. Long-term dependencies in text sequences

army vehicle / liquid container

There are a hundred liter of water in the tank.

3. Long-term dependencies in text sequences

Grammar rules create long-term dependencies
→ important for text translation

Context resolves ambiguities
→ creates long-term dependencies
→ important for text translation

3. Long-term dependencies in text sequences

Depuis le mois de mars le nombre de vols à l’aèroport de Genève a augmenté par 20 pourcent.

3. Long-term dependencies in sequences

1st example: video frame prediction

2nd example: text prediction and text translation

→ You never know in advance how many words you need
→ There might be several relevant time scales!

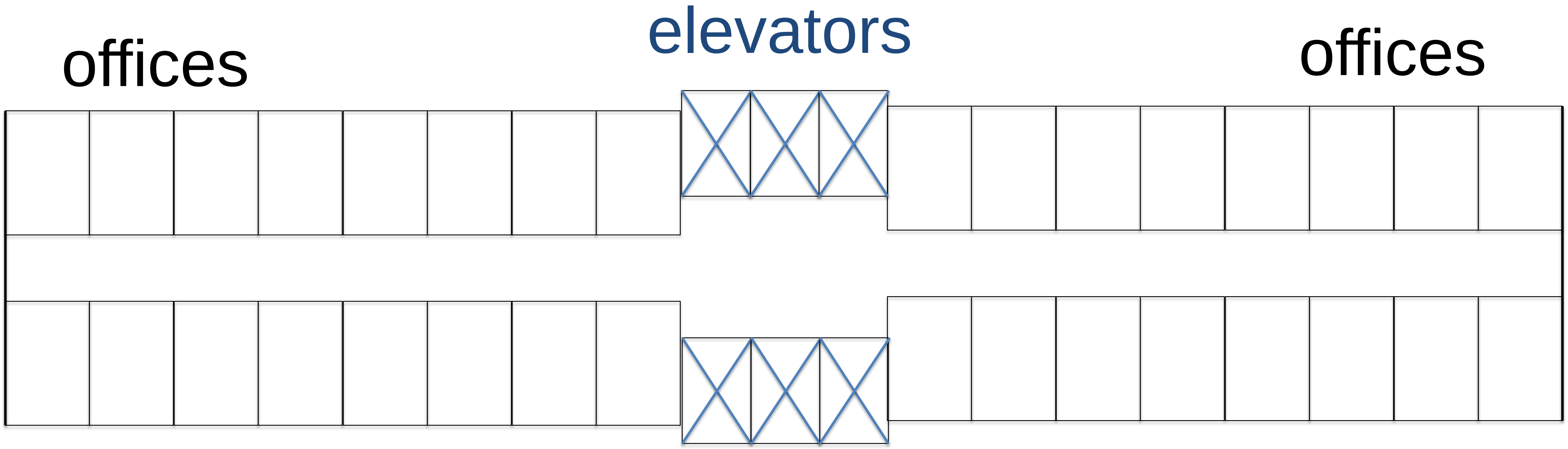
3. Long-term dependencies in sequences

1st example: video frame prediction

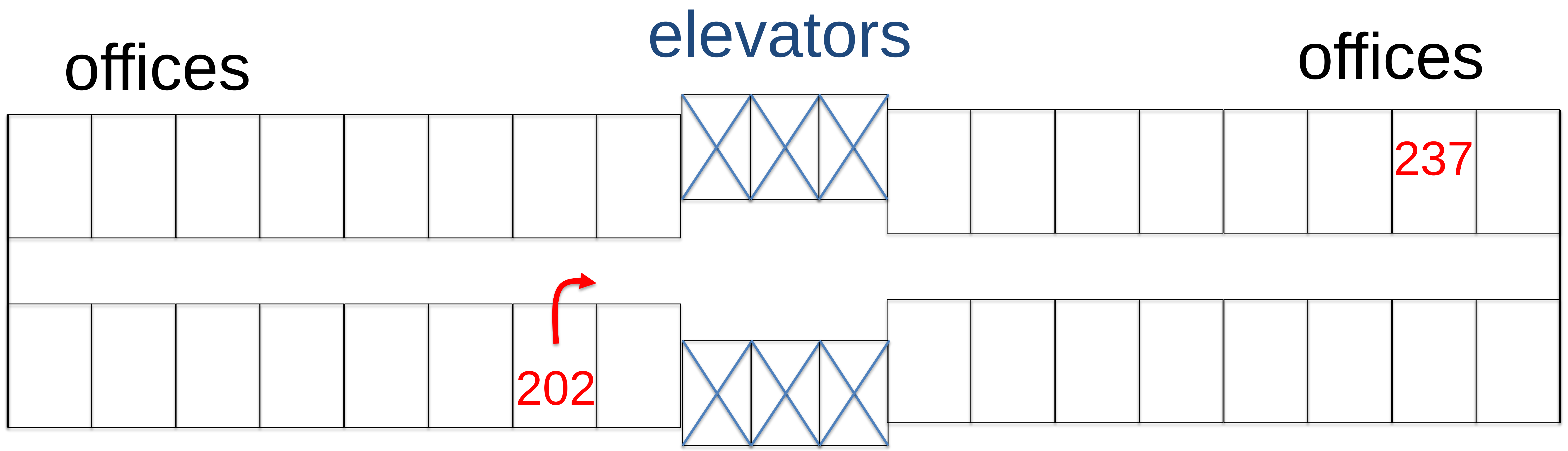
2nd example: text prediction and text translation

3rd example: action planning and navigation

3. Long-term dependencies in action sequences



3. Long-term dependencies in action sequences



start on floor 2, room 202

meeting on floor 8, room 837

3. Long-term dependencies in action sequences

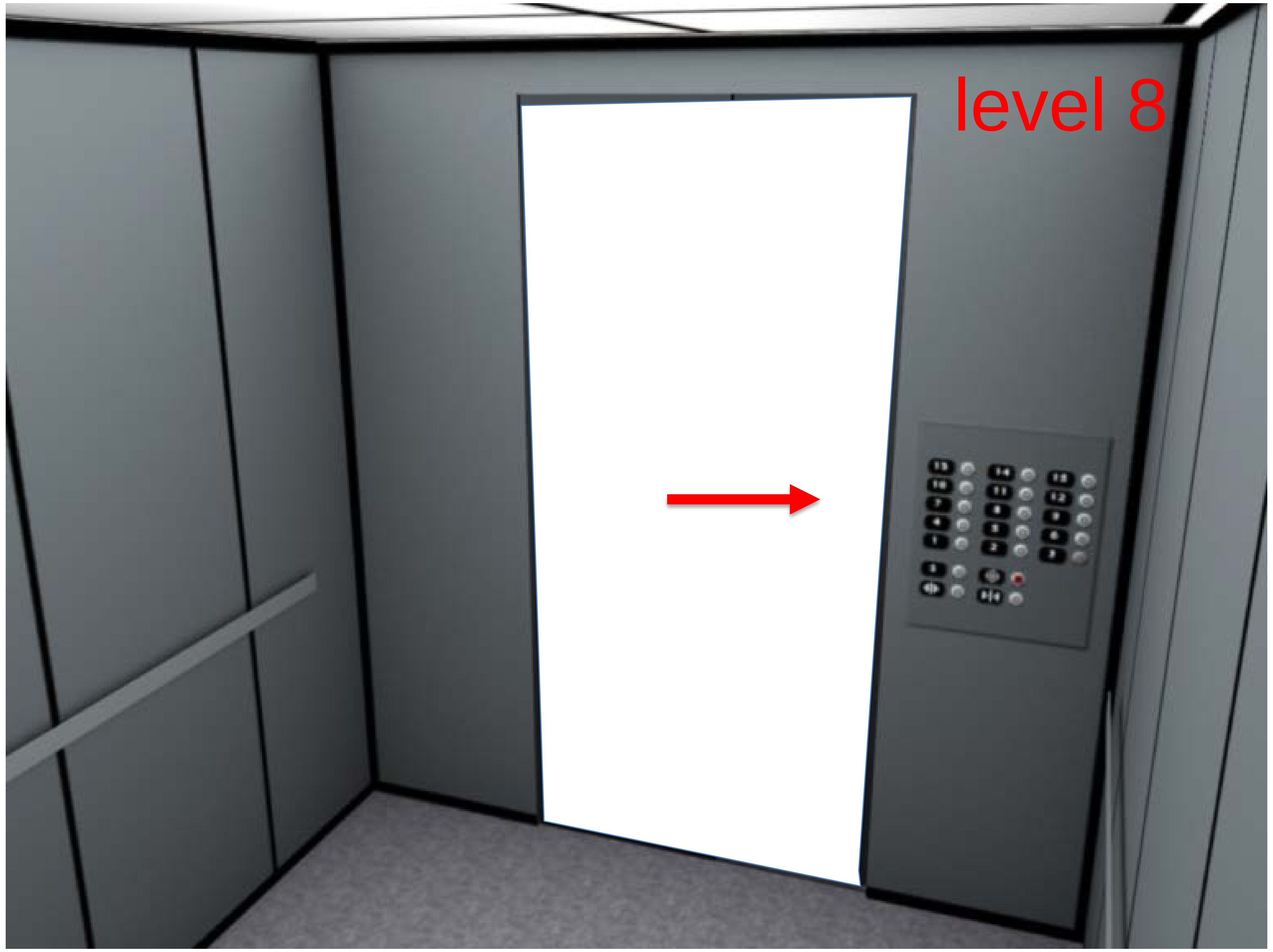
3. Long-term dependencies in action sequences

3. Long-term dependencies in action sequences

3. Long-term dependencies in action sequences

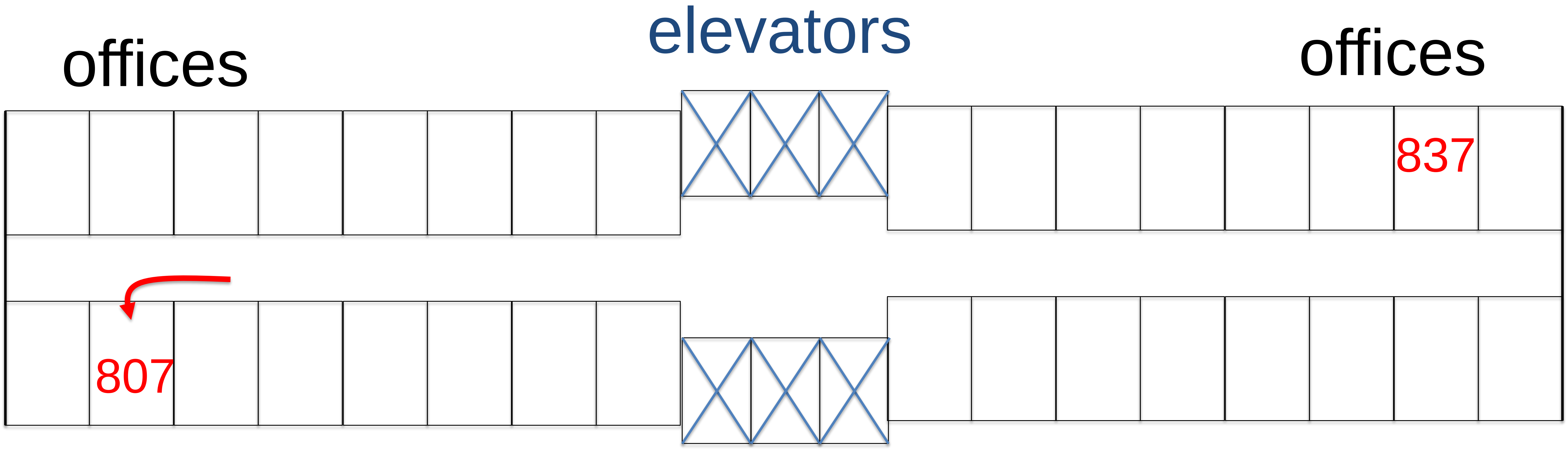
3. Long-term dependencies in action sequences

3. Long-term dependencies in action sequences



3. Long-term dependencies in action sequences

3. Long-term dependencies in action sequences



meeting on floor 8, room 837

3. Long-term dependencies in sequences

1st example: video frame prediction

2nd example: text prediction and text translation

3rd example: action planning and navigation

Symmetries create ambiguities in space
Whether you should turn left or right depends on which elevator you took
→ Long-term dependencies
→ You do not know the time scale of dependency a priori

Quiz: Sequences

[ ] In texts, the longest temporal dependence is about 10-20 words.
[ ] Training data for text sequences is scarce and costly because it needs labeling.
[ ] Training data for video frame prediction is cheap, because there are thousands of videos on the internet and no labeling is needed
[ ] Target values in sequence tasks are always high-dimensional.
[ ] In video frame prediction, if I take the last 1000 frames as input, I am sure to be on the safe side (I am sure to cover all potential temporal dependencies)

3. Long-term dependencies in Sequences

Summary:

- Sequences are everywhere
- more common in reality than static input-output paradigms
- sequences contain dependencies on several time scales
  (fast as well as slow)
- Maximum time scale is hard to know at the beginning
  (or even impossible)

→ We need a memory in the model

2. Long-term dependencies in sequences: Aim

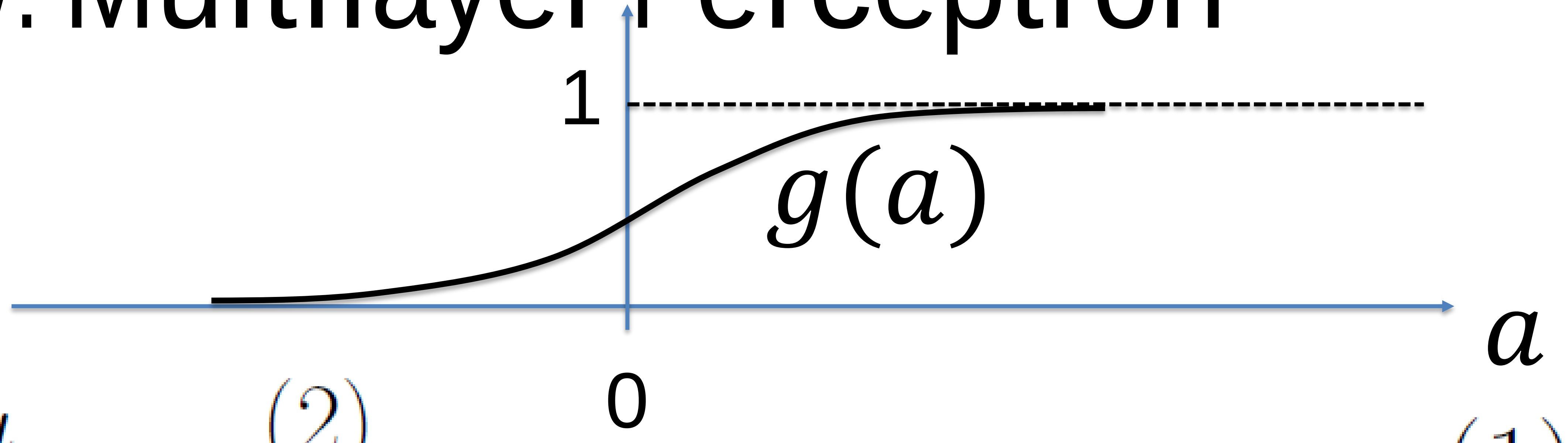
Second Question for Today

how can we keep a memory of past events in artificial neural networks?

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

1. Sequences
2. Naïve solution: increase number of inputs
3. Long-term Dependencies
4. Recurrent Neural Networks

Review: Multilayer Perceptron



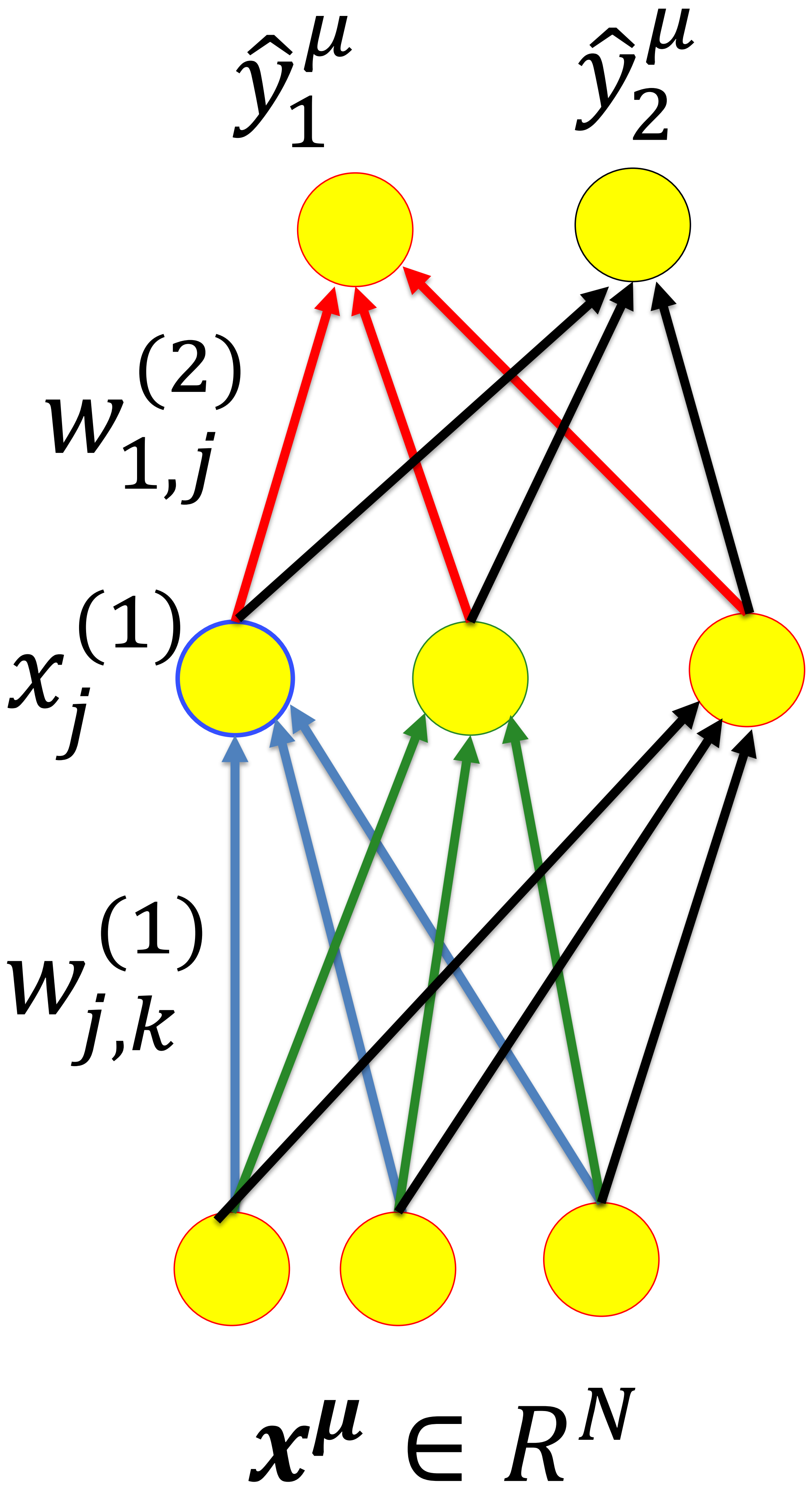
$$\hat{y}_i^\mu = x_i^{(2)} \quad (1)$$

$$= g^{(2)}[a_i^{(2)}] \quad (2)$$

$$= g^{(2)}[\sum_j w_{ij}^{(2)} x_j^{(1)}] \quad (3)$$

$$= g^{(2)}[\sum_j w_{ij}^{(2)} g^{(1)}(a_j^{(1)})] \quad (4)$$

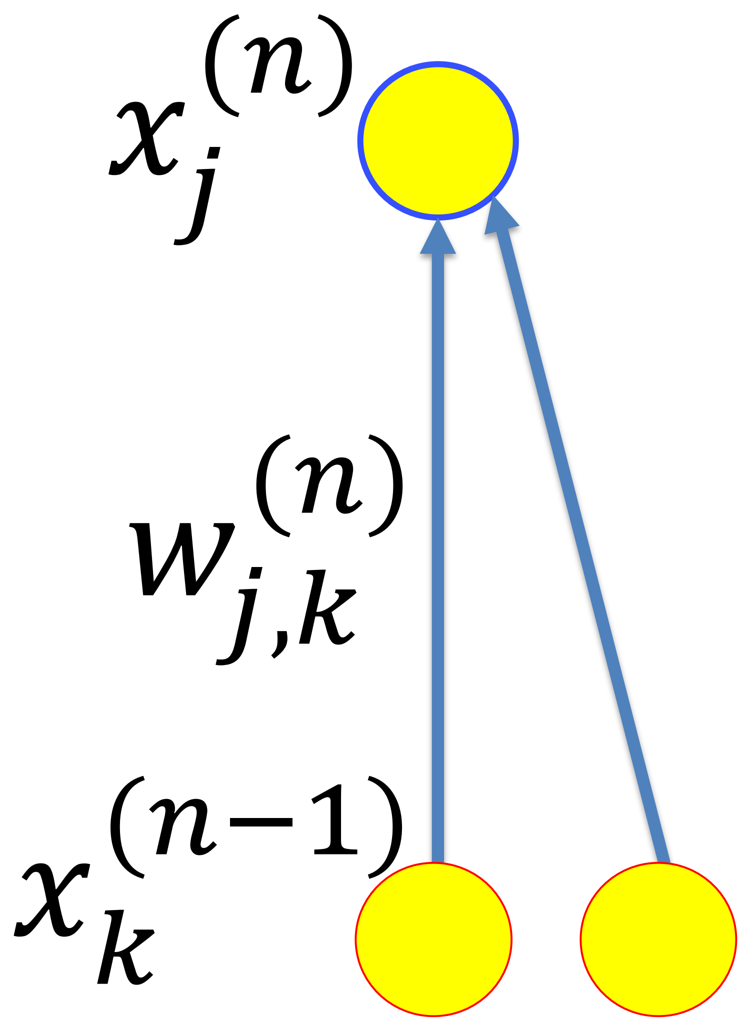
$$= g^{(2)}[\sum_j w_{ij}^{(2)} g^{(1)}(\sum_k w_{jk}^{(1)} x_k^\mu)] \quad (5)$$



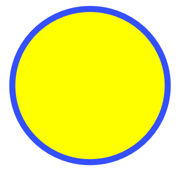
Review: graphical representation

threshold can be removed

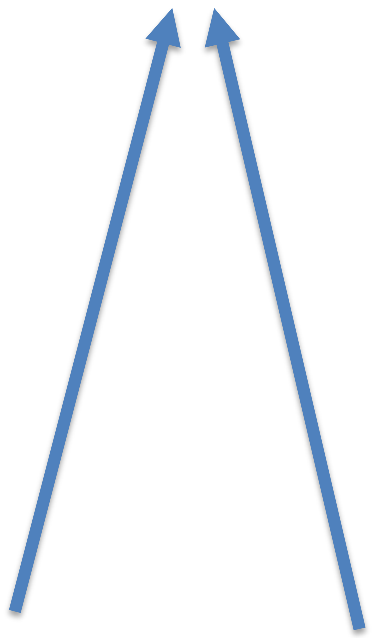
$$x_j^{(n)} = g\left(\sum_k w_{jk}\, x_k^{(n-1)} \cancel{-\vartheta}\right)$$



circle

 $= g(\,.\,)$

converging arrows

 $\sum_k$

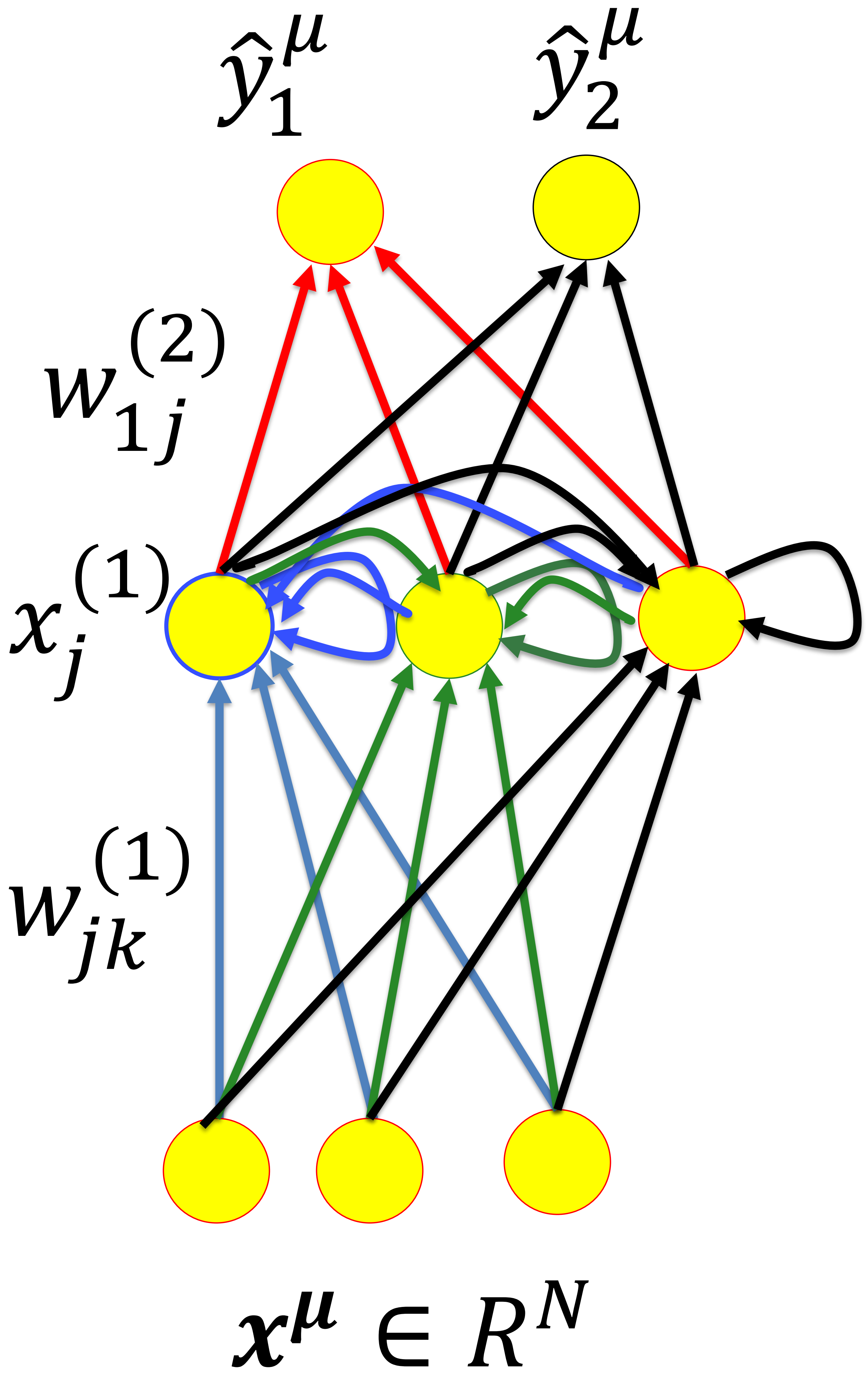
4. Recurrent Neural Networks

neurons in hidden layer
have lateral connections

$$x_j^{(1)} \leftarrow g\left(\sum_k w_{jk}^{(1)} x_k^{(0)} + \sum_i w_{ji}^{(lat)} x_i^{(1)} \right)$$

(formula can be read off from graph)

Blackboard 1



4. Update in Recurrent Neural Network

Include timing information:

Discrete big time steps t=1,2, …

Update rule for state of neuron

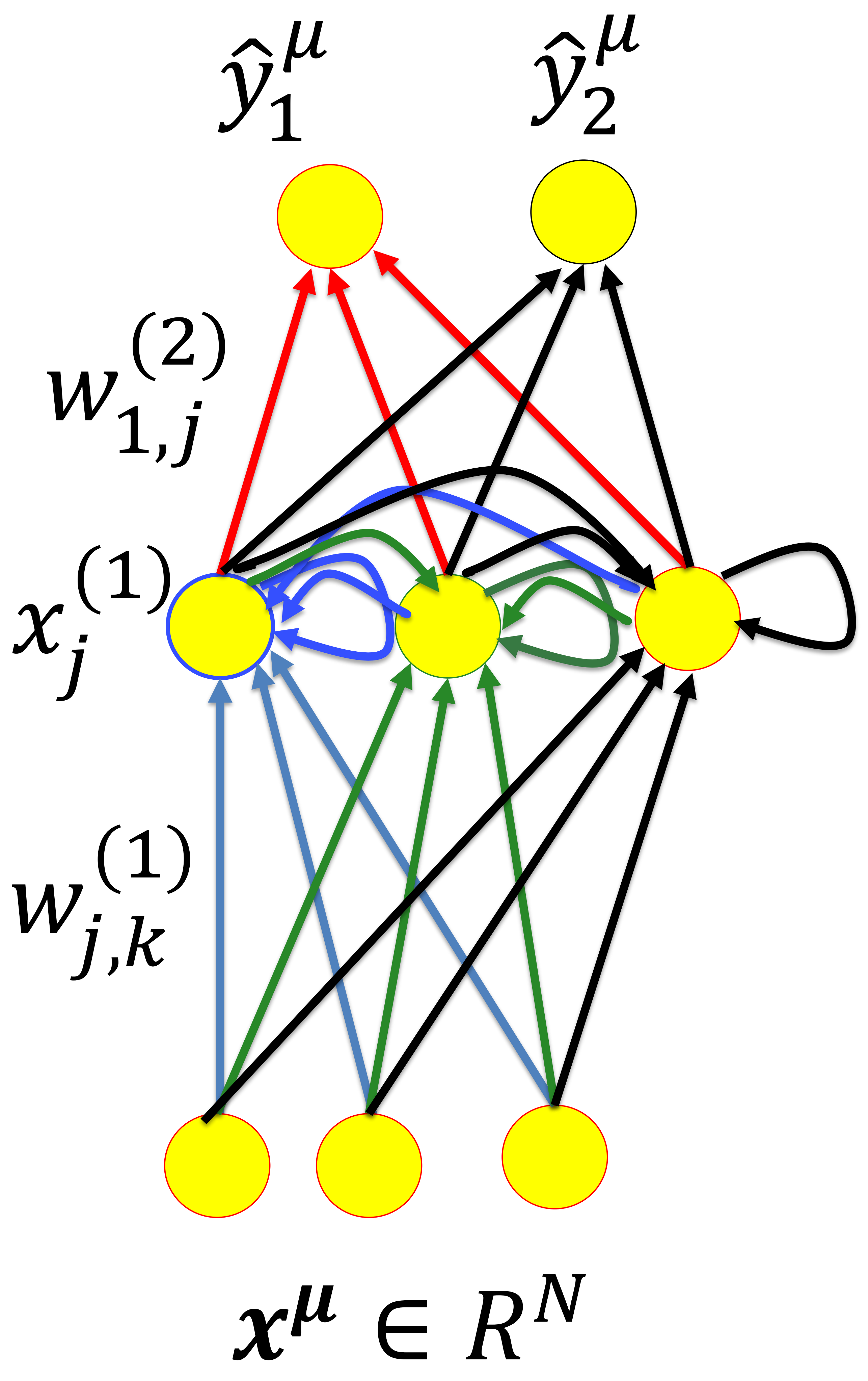
$$x_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1)}\, x_k^{(0)}(t) \;+ \sum_i w_{ji}^{(lat)}\, x_i^{(1)}(t-1)\right)$$

Input at time t:

$\boldsymbol{x}^{\mu}$ with index $\mu = t$

component

$$x_k^{(0)}(t) = x_k^t$$



4. Training data for Recurrent Neural Network

input

$\{\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3 \ \ ..., \boldsymbol{x}^T\}$ single sequence of length T

target vector for output

$\{\boldsymbol{t}^1, \boldsymbol{t}^2, \boldsymbol{t}^3 \ \ ..., \boldsymbol{t}^{T-1}\}$

one example is: predict next input (e.g. video frame)

$\boldsymbol{t}^1 = \boldsymbol{x}^2$

$\boldsymbol{t}^2 = \boldsymbol{x}^3$

$\boldsymbol{t}^3 = \boldsymbol{x}^4$ ‘target at time step 3 is the input at time step 4’

4. Training data for Recurrent Neural Network (text example)

'The grammar book of my friend. The first sentence often begins with a three-letter word, because the word 'the' is quite common. However much longer words are also possible as a first word of a sentence. Therefore this is just a rule of thumb. … '

$\boldsymbol{x}^1$ = character T in 1-hot coding

input

$$\{\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3 \;\; \ldots, \boldsymbol{x}^T\}$$

target vector for output

$$\{\boldsymbol{t}^1, \boldsymbol{t}^2, \boldsymbol{t}^3 \;\; \ldots, \boldsymbol{t}^{T-1}\}$$

aim is: predict end of word symbol (text processing)

$$\boldsymbol{t}^1 = 0$$
$$\boldsymbol{t}^2 = 0$$
$$\boldsymbol{t}^3 = 1$$

'target at time step 3 is the 'blank' at time step 4'

4. Update in Recurrent Neural Network (details)

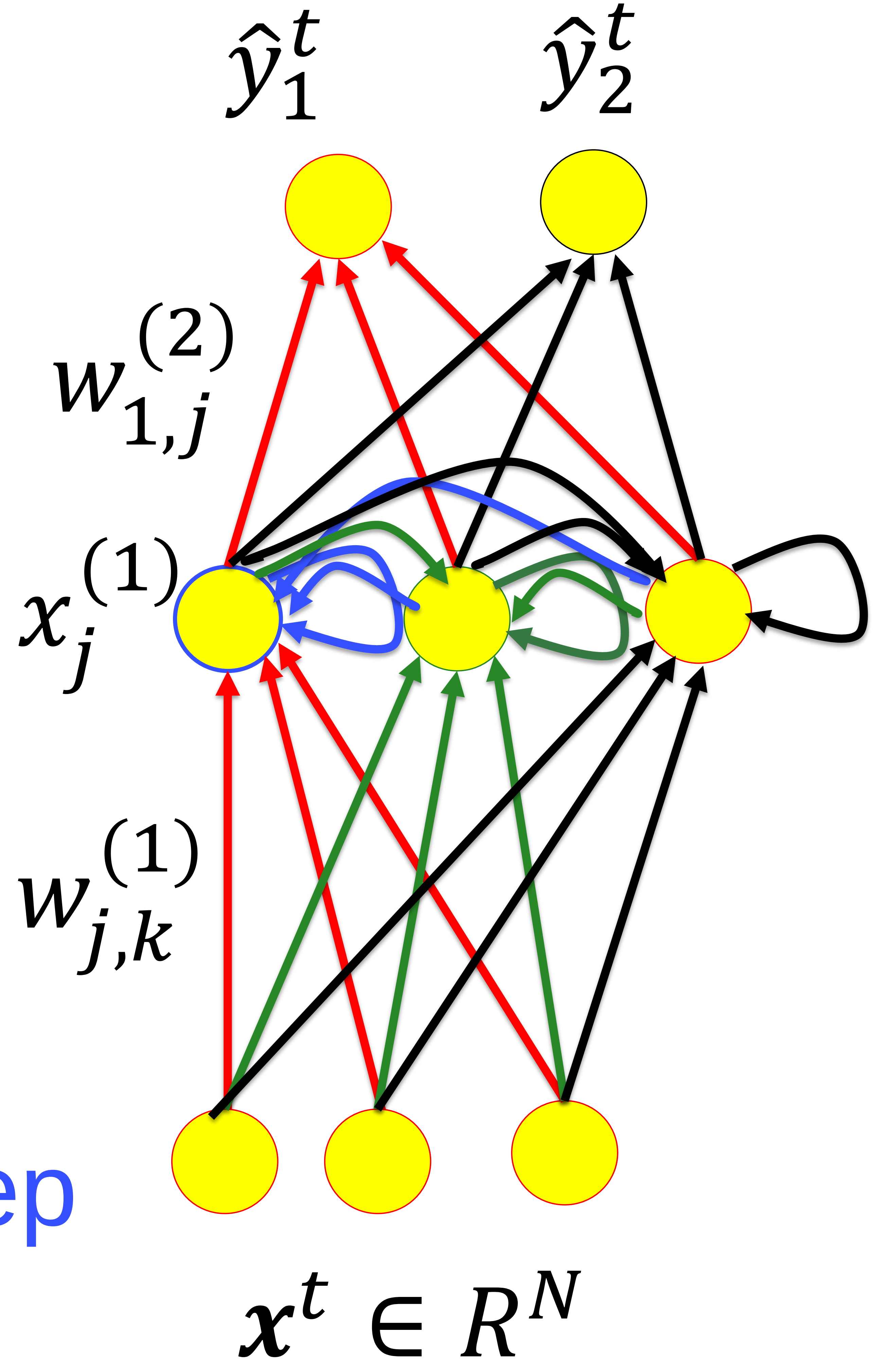
Discrete big time steps t=1,2, …

$$\hat{y}_i^t = \hat{y}_i\ (t) = g\left(\sum w_{ij}^{(2)} x_j^{(1)}(t)\right)$$

$$x_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1)}\ x_k^{(0)}(t)\ \ + \sum_i w_{ji}^{(lat)}\ x_i^{(1)}(t-1)\ \right)$$

Feedforward processing
within the same time step
(feedforward pass)

Lateral input
from previous step

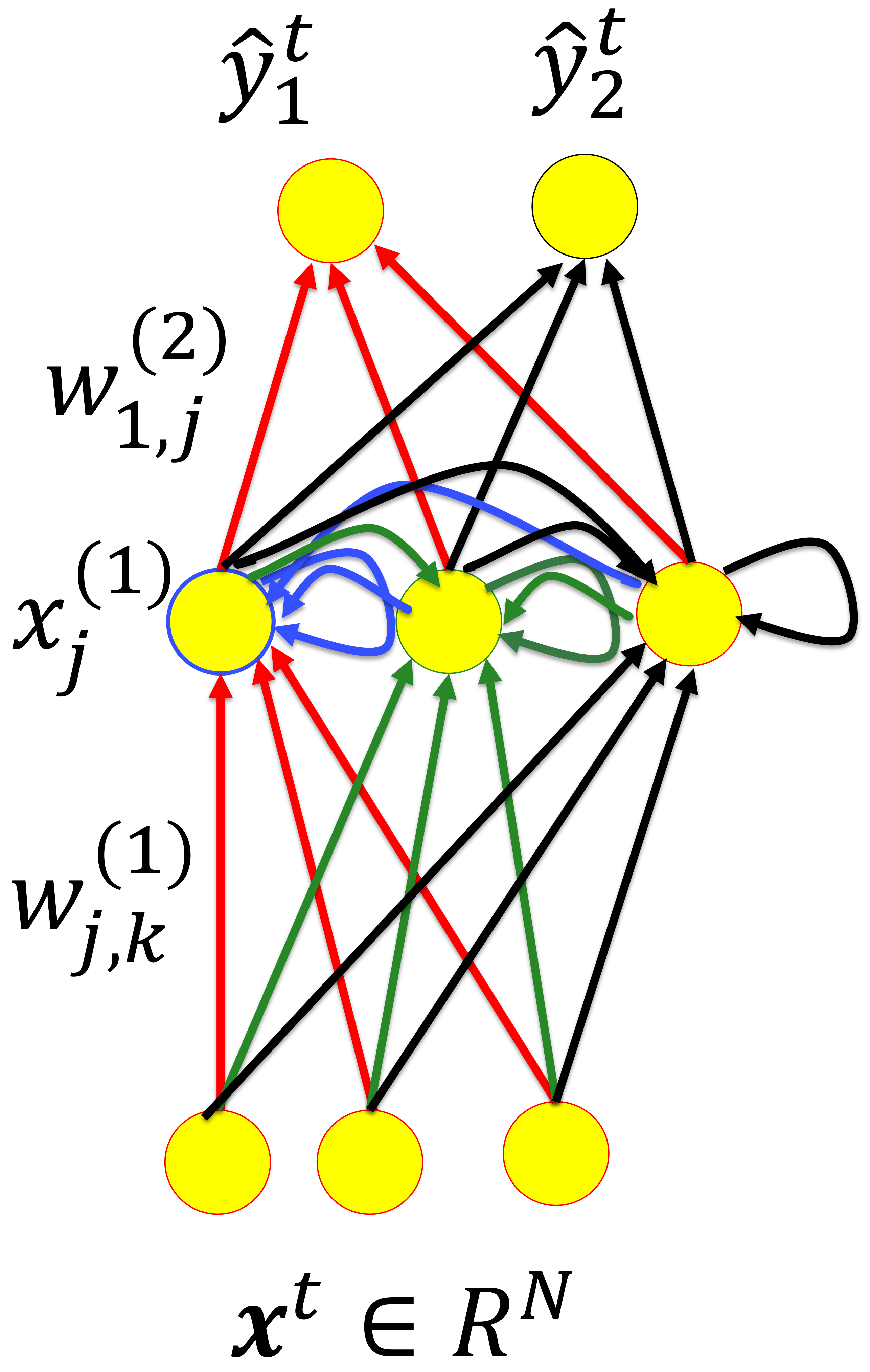


4. Update in Recurrent Neural Network (details)

Update scheme looks complicated.

Question:

How does this work in practice?



Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

1. Sequences
2. Naïve solution: increase number of inputs
3. Long-term Dependencies
4. Recurrent Neural Networks
5. Unfolding the network in time

5. Update in Recurrent Neural Network

Discrete big time steps $t=1,2,\ldots$

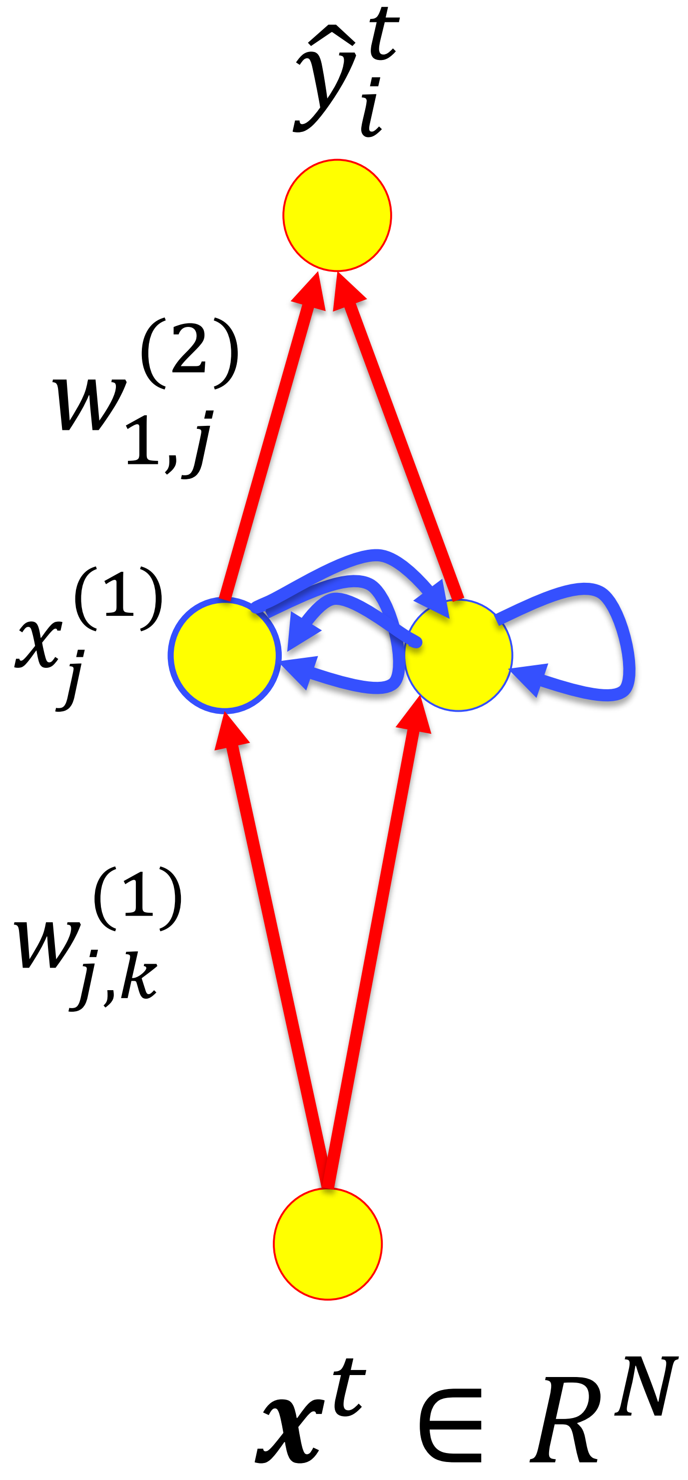
Exercise 1
In Class (8min)

$$\hat{y}_i^t = \hat{y}_i\ (t) = g\left(\sum w_{ij}^{(2)} x_j^{(1)}(t) - \vartheta\right)$$

$$x_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1)}\ x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat)}\ x_i^{(1)}(t-1) - \vartheta_j\right)$$

Feedforward processing within one big time step

Lateral input from previous step



5. Unfolding in time

Blackboard 2

5. Compact graphics for Recurrent Neural Network

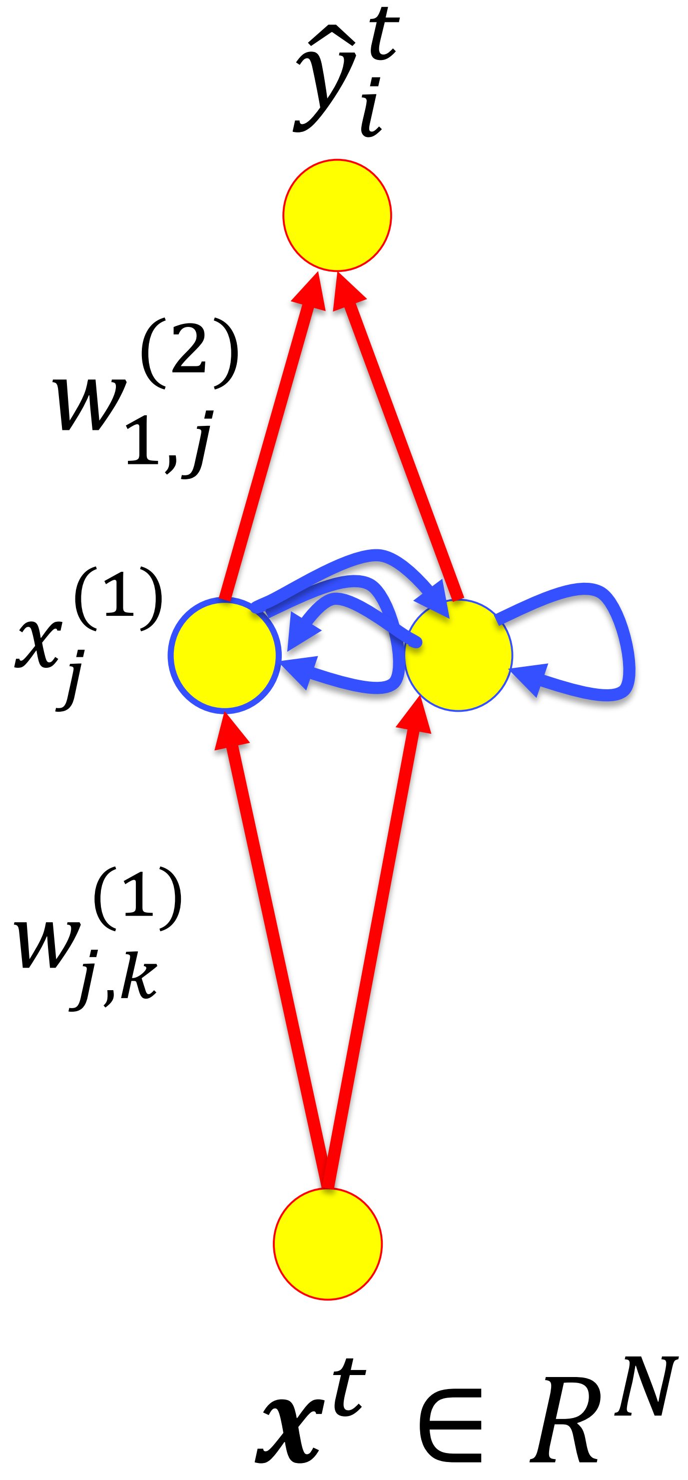
Discrete big time steps $t=1,2,\ldots$

$$\hat{y}_i^t = \hat{y}_i\ (t) = g\left(\sum w_{ij}^{(2)} x_j^{(1)}(t) - \vartheta\right)$$

$$x_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1)}\ x_k^{(0)}(t)\ \ + \sum_i w_{ji}^{(lat)}\ x_i^{(1)}(t-1) - \vartheta_j\right)$$

Feedforward processing within one big time step

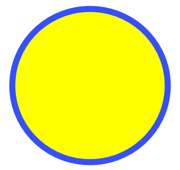
Lateral input from previous step



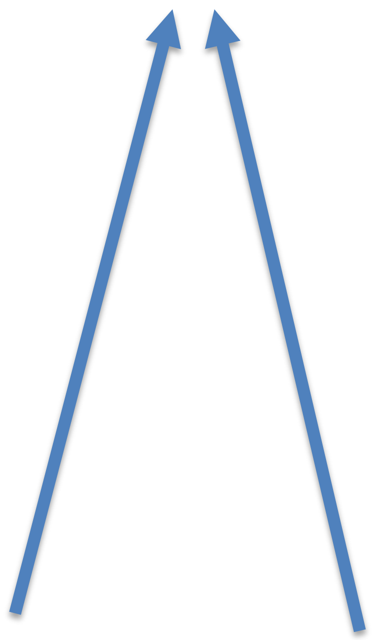
Review: graphical representation

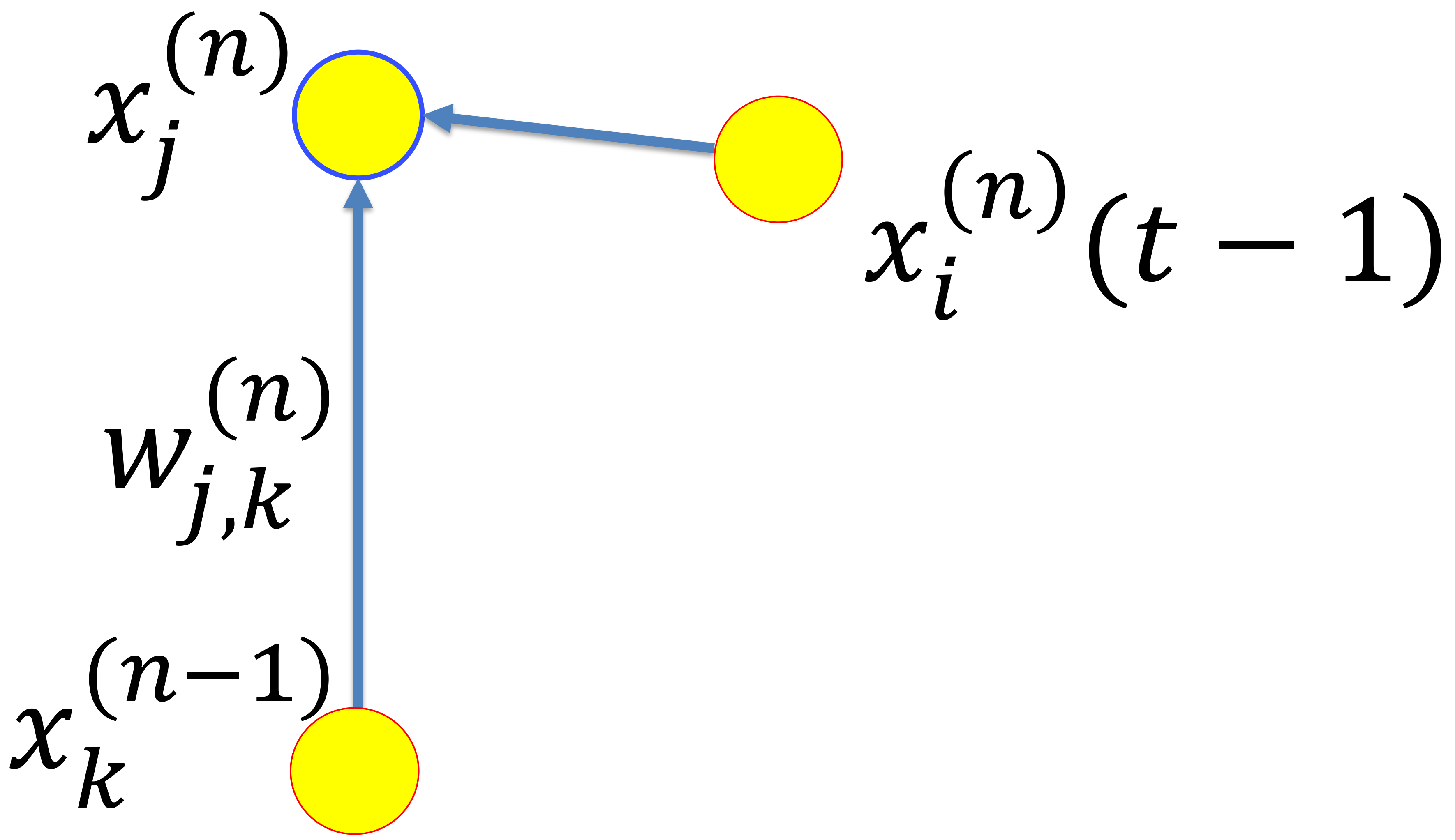
$$x_j^{(n)}(t) = g\left(\sum_k w_{jk}\, x_k^{(n-1)}(t) \quad + \sum_i w_{ji}\, x_i^{(n)}(t-1) \quad \right)$$

circle

 $= g(\,.\,)$

converging arrows

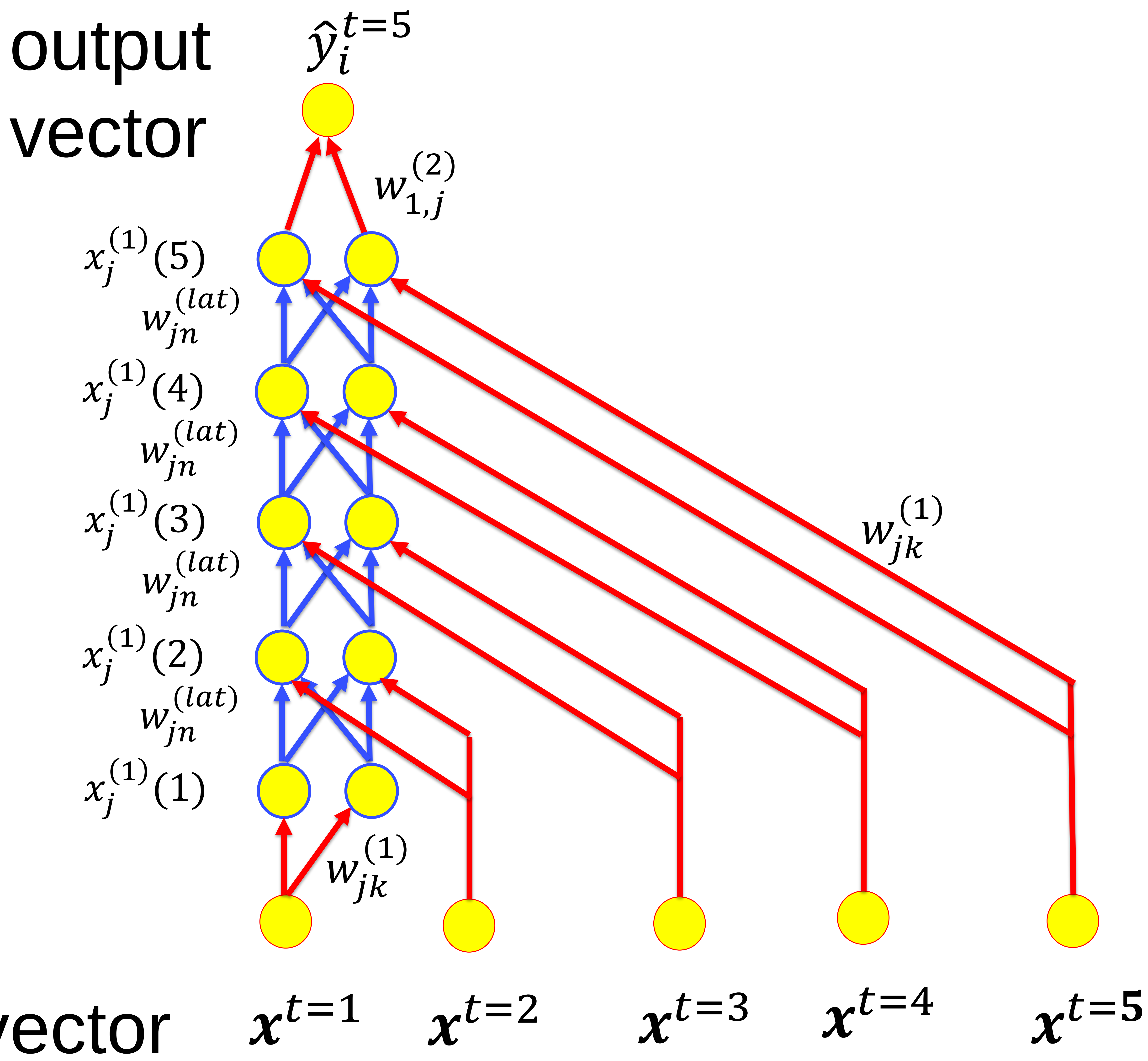
 $\sum$



5. unfolded graphics for Recurrent Neural Network

Discrete big time steps
t=1,2,3,4,5

equivalent
feedforward network
for 5 time steps

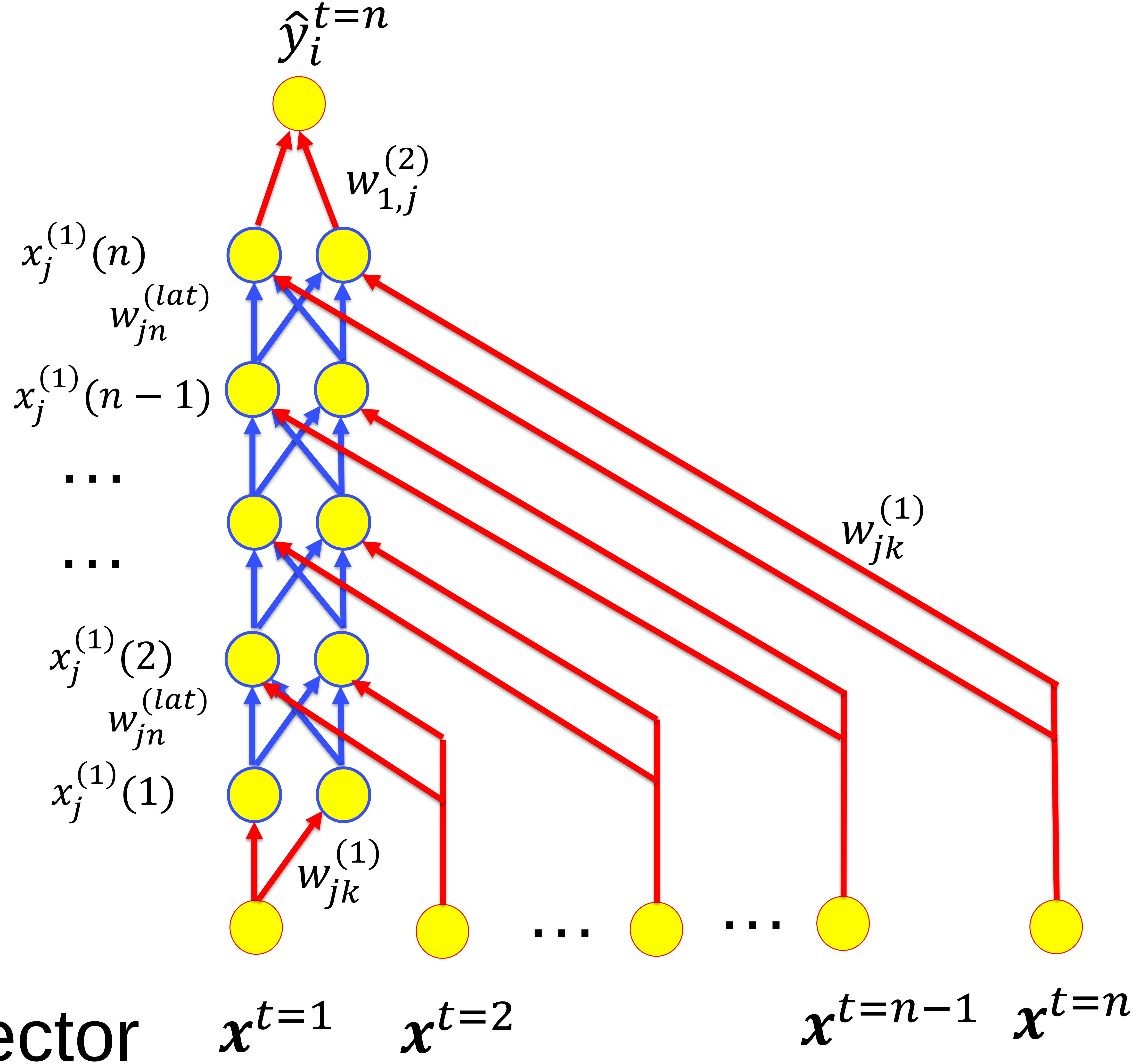


5. unfolded graphics for Recurrent Neural Network

Discrete big time steps

$t=1,2,3,4,5, \ldots , n$

equivalent
feedforward network
for *n* time steps
→
n hidden layers with
identical feedforward
weights



Quiz: Unfolding of Recurrent Networks

We process a **sequence of length *T*.**

[ ] When processing a sequence of length *T*, a recurrent network with one hidden layer can always be reformulated as a deep feedforward network.

[ ] A recurrent network with one hidden layer of *n* neurons leads to an unfolded feedforward network with *n* layers of *n* neurons each.

[ ] A recurrent network with one hidden layer of *n* neurons leads to an unfolded feedforward network with *T* hidden layers

[ ] The unfolded network corresponds to a feedforward network with weight sharing.

[ ] The unfolded network corresponds to a feedforward network where inputs have direct short-cut connections to all hidden layers.

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

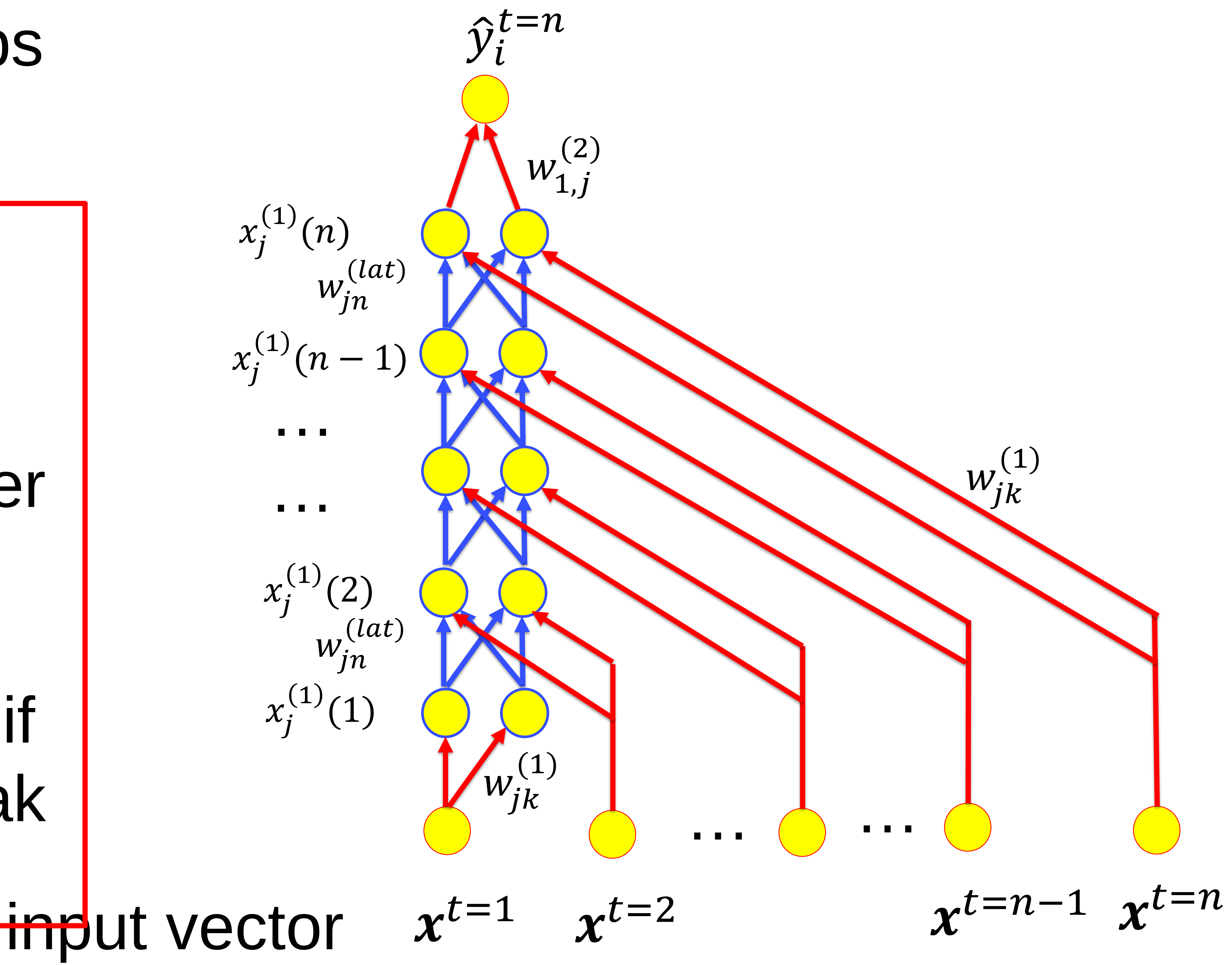
1. Sequences
2. Naïve solution: increase number of inputs
3. Long-term Dependencies
4. Recurrent Neural Networks
5. Unfolding the network in time
6. Backpropagation through time

6. Backpropogation through time

Discrete big time steps

$t=1,2,3,4,5,\ldots,n$

- take the unfolded equivalent network
- apply backprop after each time step
- cut backward path if signal gets too weak



BackProp

0. **Initialization of weights**

1. **Choose pattern $\mathbf{x}^\mu$**

   input $x_k^{(0)} = x_k^\mu$

2. **Forward propagation of signals** $x_k^{(n-1)} \longrightarrow x_j^{(n)}$

$$x_j^{(n)} = g^{(n)}(a_j^{(n)}) = g^{(n)}(\sum w_{jk}^{(n)} x_k^{(n-1)}) \quad (1)$$

   output $\hat{y}_i^\mu = x_i^{(n_{\max})}$

3. **Computation of errors in output**

$$\delta_i^{(n_{\max})} = g'(a_i^{(n_{\max})})\ [t_i^\mu - \hat{y}^\mu] \quad (2)$$

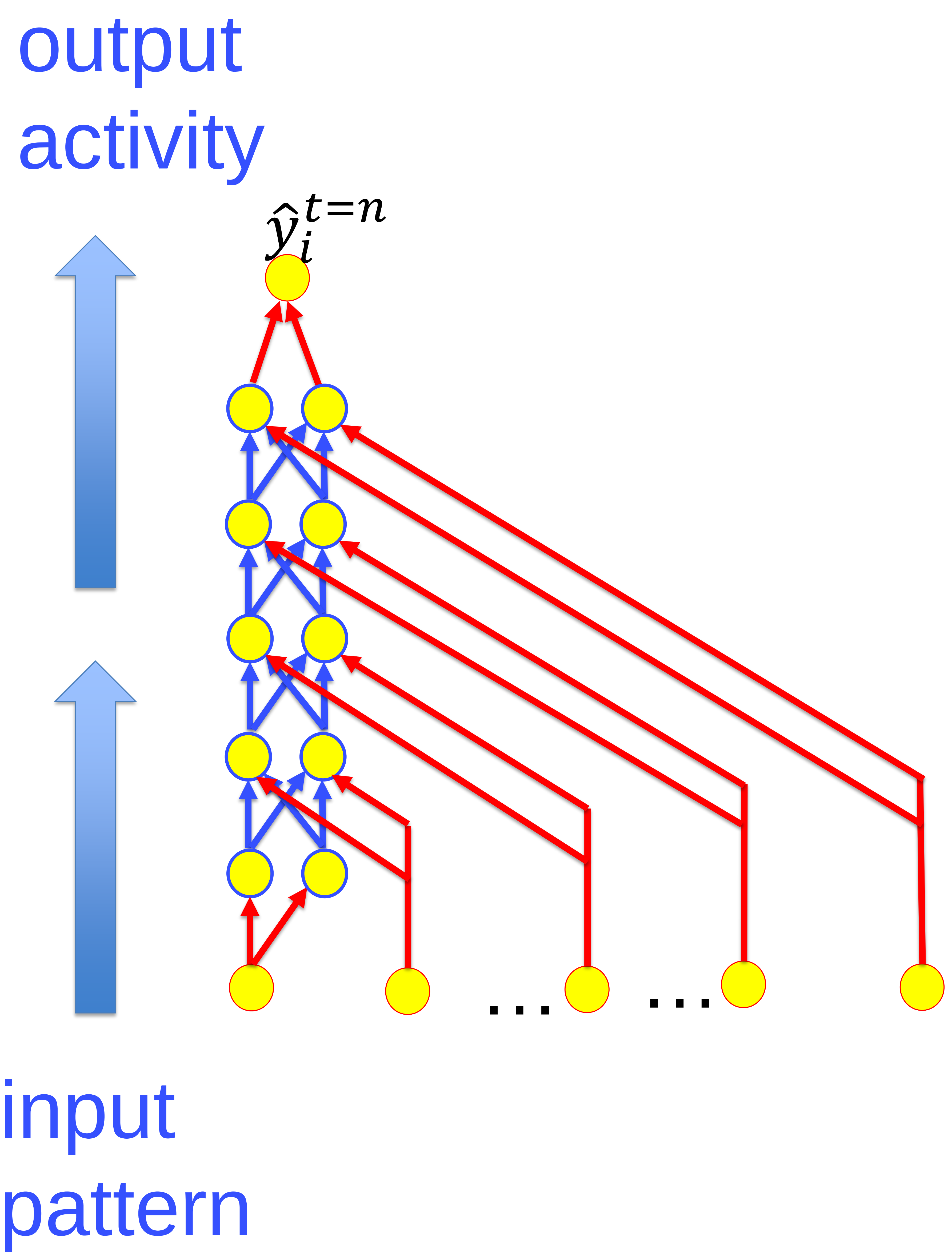
4. **Backward propagation of errors** $\delta_i^{(n)} \longrightarrow \delta_j^{(n-1)}$

$$\delta_j^{(n-1)} = g'^{(n-1)}(a^{(n-1)}) \sum_i w_{ij}\, \delta_i^{(n)} \quad (3)$$

5. **Update weights (for each (i,j) and all layers (n))**

$$\Delta w_{ij}^{(n)} = \eta\, \delta_i^{(n)}\, x_j^{(n-1)} \quad (4)$$

6. **Return to step 1.**



BackProp

0. **Initialization of weights**

1. **Choose pattern** $\mathbf{x}^\mu$

   input $x_k^{(0)} = x_k^\mu$

2. **Forward propagation of signals** $x_k^{(n-1)} \longrightarrow x_j^{(n)}$

$$x_j^{(n)} = g^{(n)}(a_j^{(n)}) = g^{(n)}(\sum w_{jk}^{(n)} x_k^{(n-1)}) \tag{1}$$

   output $\hat{y}_i^\mu = x_i^{(n_{\max})}$

3. **Computation of errors in output**

$$\delta_i^{(n_{\max})} = g'(a_i^{(n_{\max})})\ [t_i^\mu - \hat{y}^\mu] \tag{2}$$

4. **Backward propagation of errors** $\delta_i^{(n)} \longrightarrow \delta_j^{(n-1)}$

$$\delta_j^{(n-1)} = g'^{(n-1)}(a^{(n-1)}) \sum_i w_{ij}\, \delta_i^{(n)} \tag{3}$$

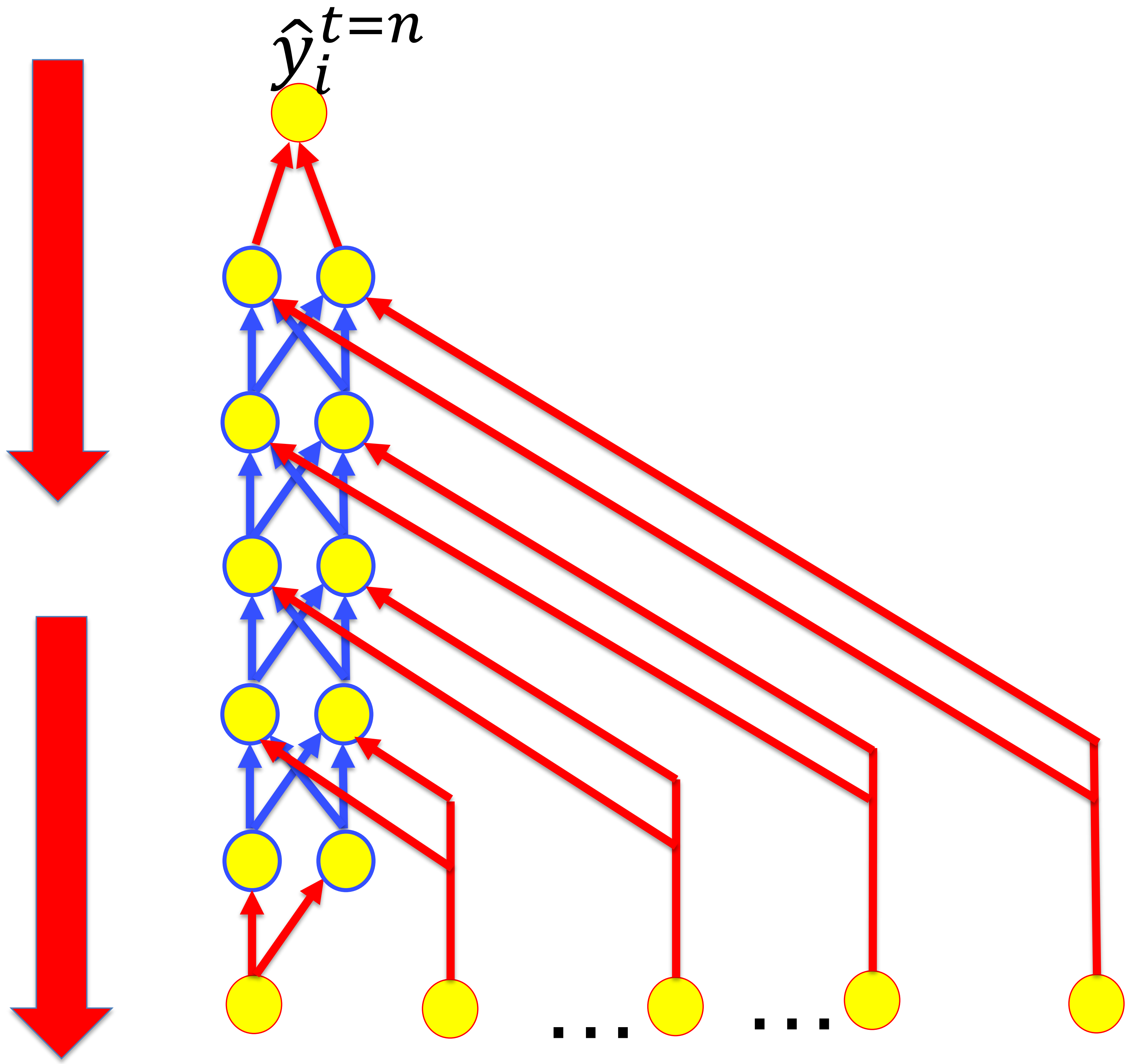
5. **Update weights (for each** (i,j) **and all layers** (n)**)**

$$\Delta w_{ij}^{(n)} = \eta\, \delta_i^{(n)}\, x_j^{(n-1)} \tag{4}$$

6. **Return to step 1.**

Calculate output error

δ



BackProp

0. Initialization of weights
1. Choose pattern $\mathbf{x}^\mu$

   input $x_k^{(0)} = x_k^\mu$

2. Forward propagation of signals $x_k^{(n-1)} \longrightarrow x_j^{(n)}$

$$x_j^{(n)} = g^{(n)}(a_j^{(n)}) = g^{(n)}(\sum w_{jk}^{(n)} x_k^{(n-1)}) \qquad (1)$$

   output $\hat{y}_i^\mu = x_i^{(n_{\max})}$

3. Computation of errors in output

$$\delta_i^{(n_{\max})} = g'(a_i^{(n_{\max})})\ [t_i^\mu - \hat{y}^\mu] \qquad (2)$$

4. Backward propagation of errors $\delta_i^{(n)} \longrightarrow \delta_j^{(n-1)}$

$$\delta_j^{(n-1)} = g'^{(n-1)}(a^{(n-1)})\ \sum_i w_{ij}\, \delta_i^{(n)} \qquad (3)$$

5. Update weights (for each (i,j) and all layers (n))

$$\boxed{\Delta w_{ij}^{(n)} = \eta\, \delta_i^{(n)}\, x_j^{(n-1)} \qquad (4)}$$

6. Return to step 1.

update **all** weights

$$\Delta w_{i,j}^{(n)} = \delta_i^{(n)} x_j^{(n-1)}$$

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

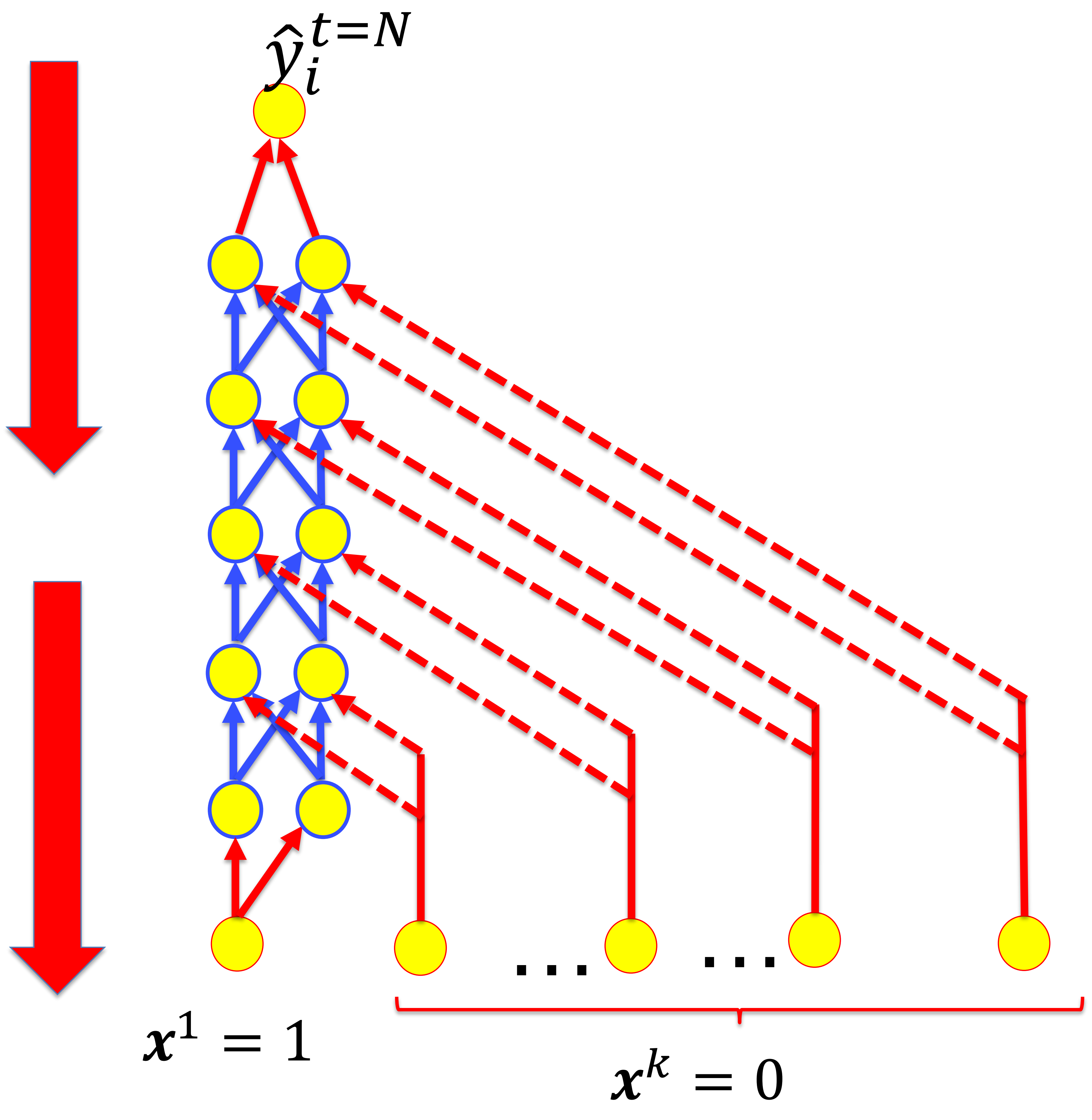
1. Sequences
2. Naïve solution: increase number of inputs
3. Long-term Dependencies
4. Recurrent Neural Networks
5. Unfolding the network in time
6. Backpropagation through time
7. The vanishing Gradient Problem

7. Vanishing gradient problem

- Assume strong input at time *t=1*
- Assume no further input up to time $t=N$
- Calculate error in output
- Backpropagate over N layers to find the effect of earlier input on the output now

Calculate output error

δ



BackProp

0. Initialization of weights

1. Choose pattern $\mathbf{x}^\mu$

   input $x_k^{(0)} = x_k^\mu$

2. Forward propagation of signals $x_k^{(n-1)} \longrightarrow x_j^{(n)}$

$$x_j^{(n)} = g^{(n)}(a_j^{(n)}) = g^{(n)}(\sum w_{jk}^{(n)} x_k^{(n-1)}) \tag{1}$$

   output $\hat{y}_i^\mu = x_i^{(n_{\max})}$

3. Computation of errors in output

$$\delta_i^{(n_{\max})} = g'(a_i^{(n_{\max})})\ [t_i^\mu - \hat{y}^\mu] \tag{2}$$

4. Backward propagation of errors $\delta_i^{(n)} \longrightarrow \delta_j^{(n-1)}$

$$\delta_j^{(n-1)} = g'^{(n-1)}(a^{(n-1)}) \sum_i w_{ij}\, \delta_i^{(n)} \tag{3}$$

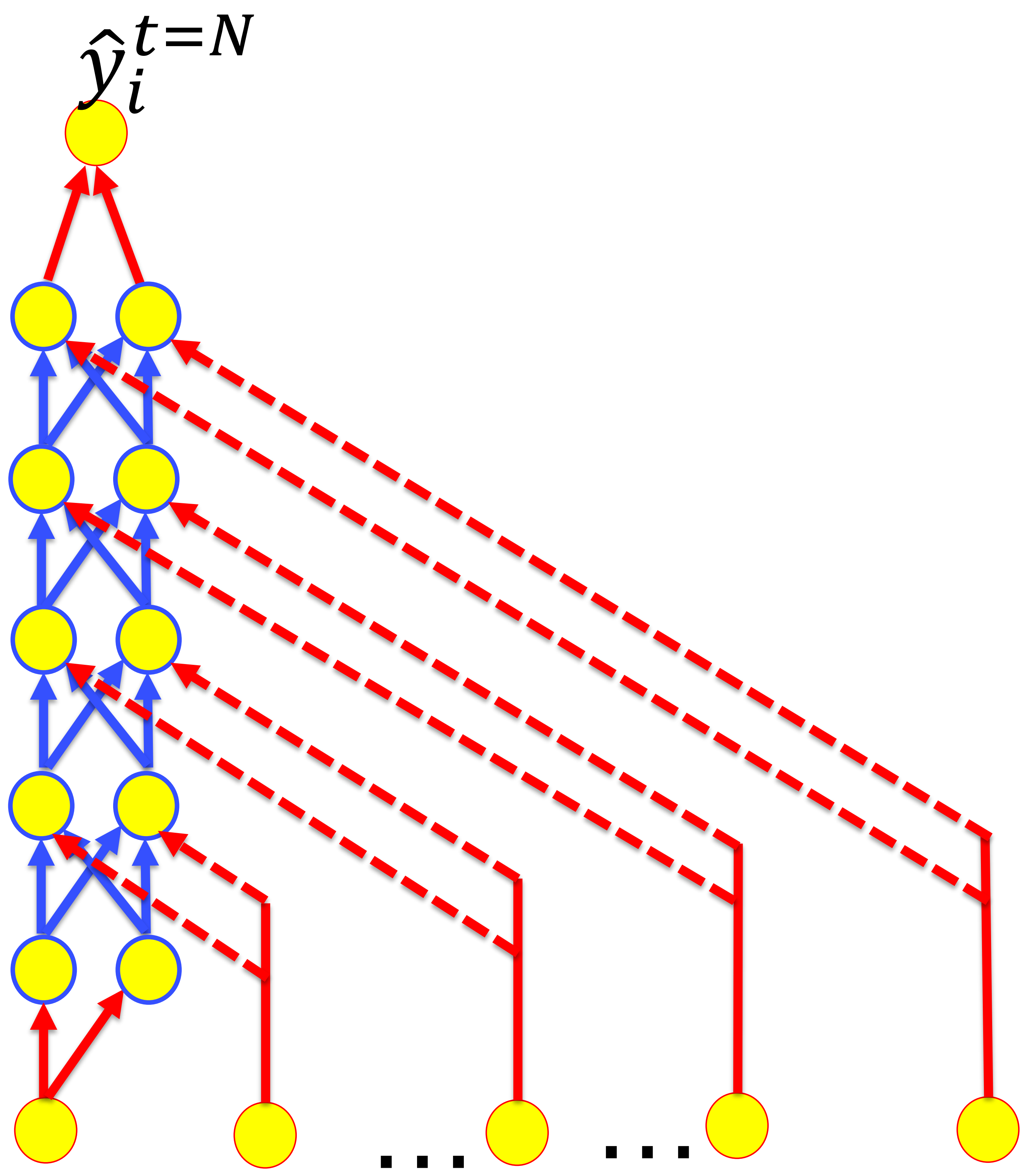
5. Update weights (for each (i,j) and all layers (n))

$$\Delta w_{ij}^{(n)} = \eta\, \delta_i^{(n)}\, x_j^{(n-1)} \tag{4}$$

6. Return to step 1.

Calculate output error

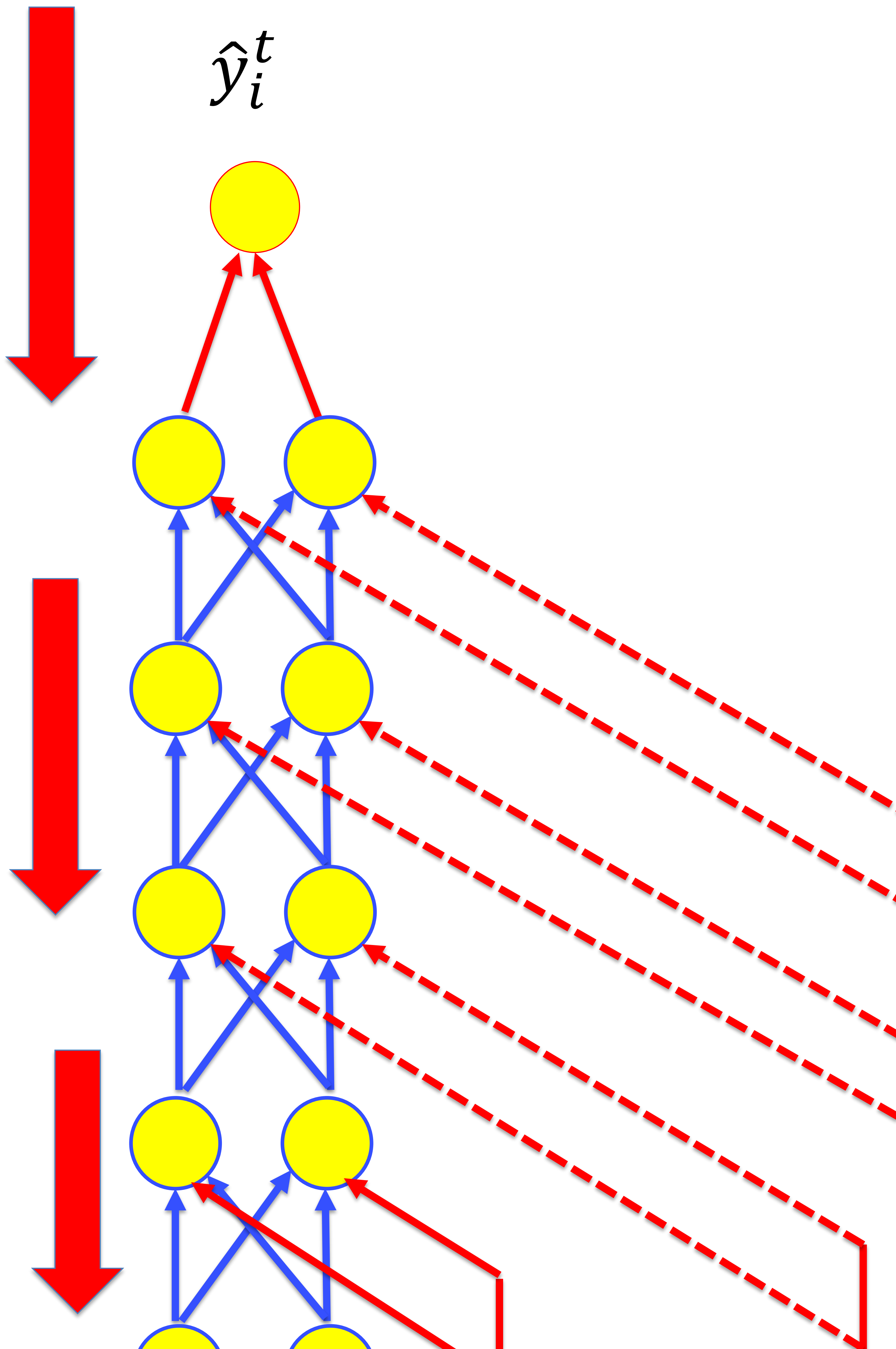
δ



7. Vanishing gradient problem

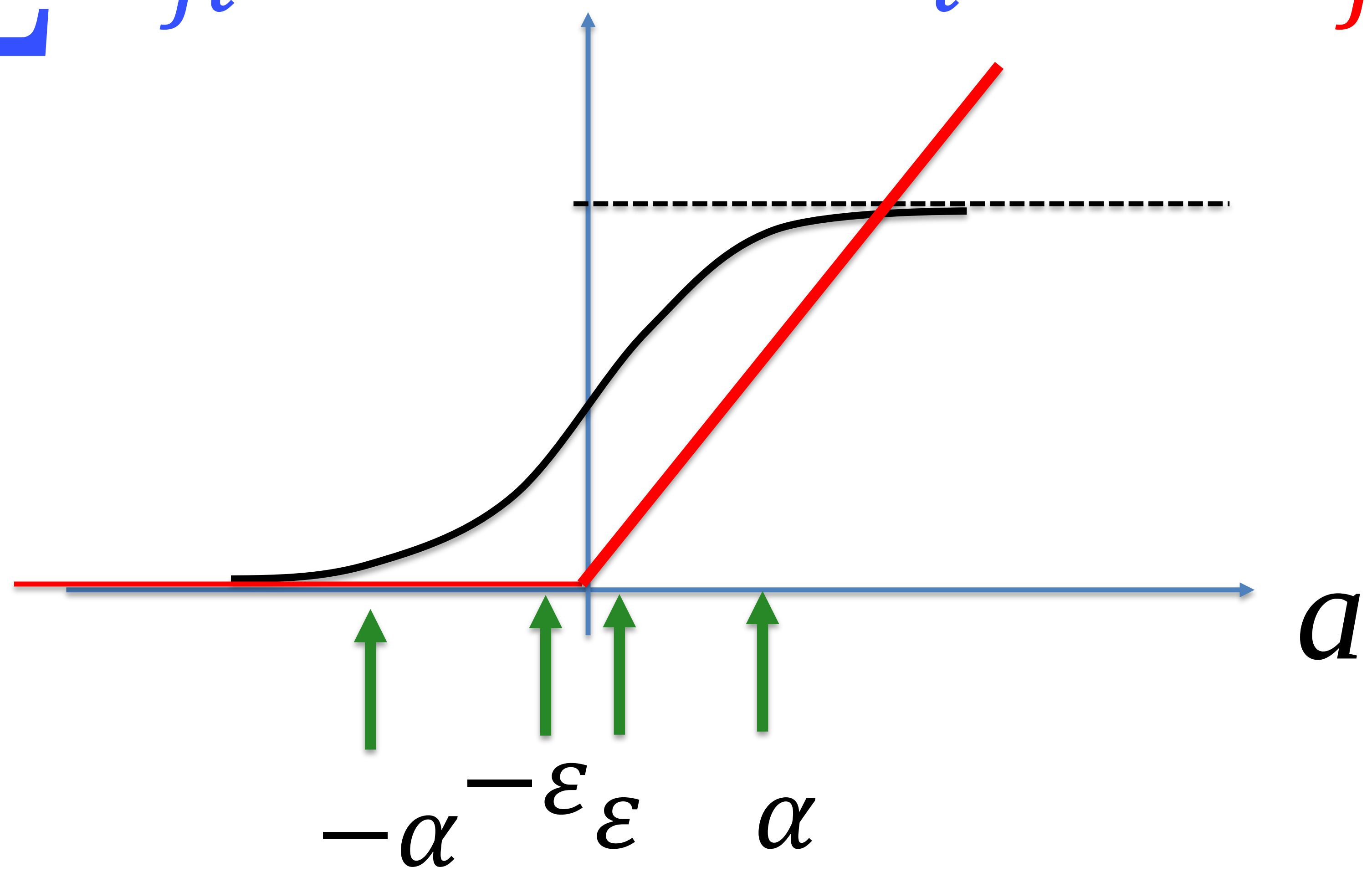
- Assume strong input at time *t-N*,
- Assume no further input up to time t
- Calculate error in output
- Backpropagate over N layers to find the effect of input

$$\delta_i^{(n-1)} = \sum_j w_{ji}^{(lat)} g'^{(n-1)}(a_i^{(n-1)}) \delta_j^{(n)}$$



7. Vanishing gradient problem

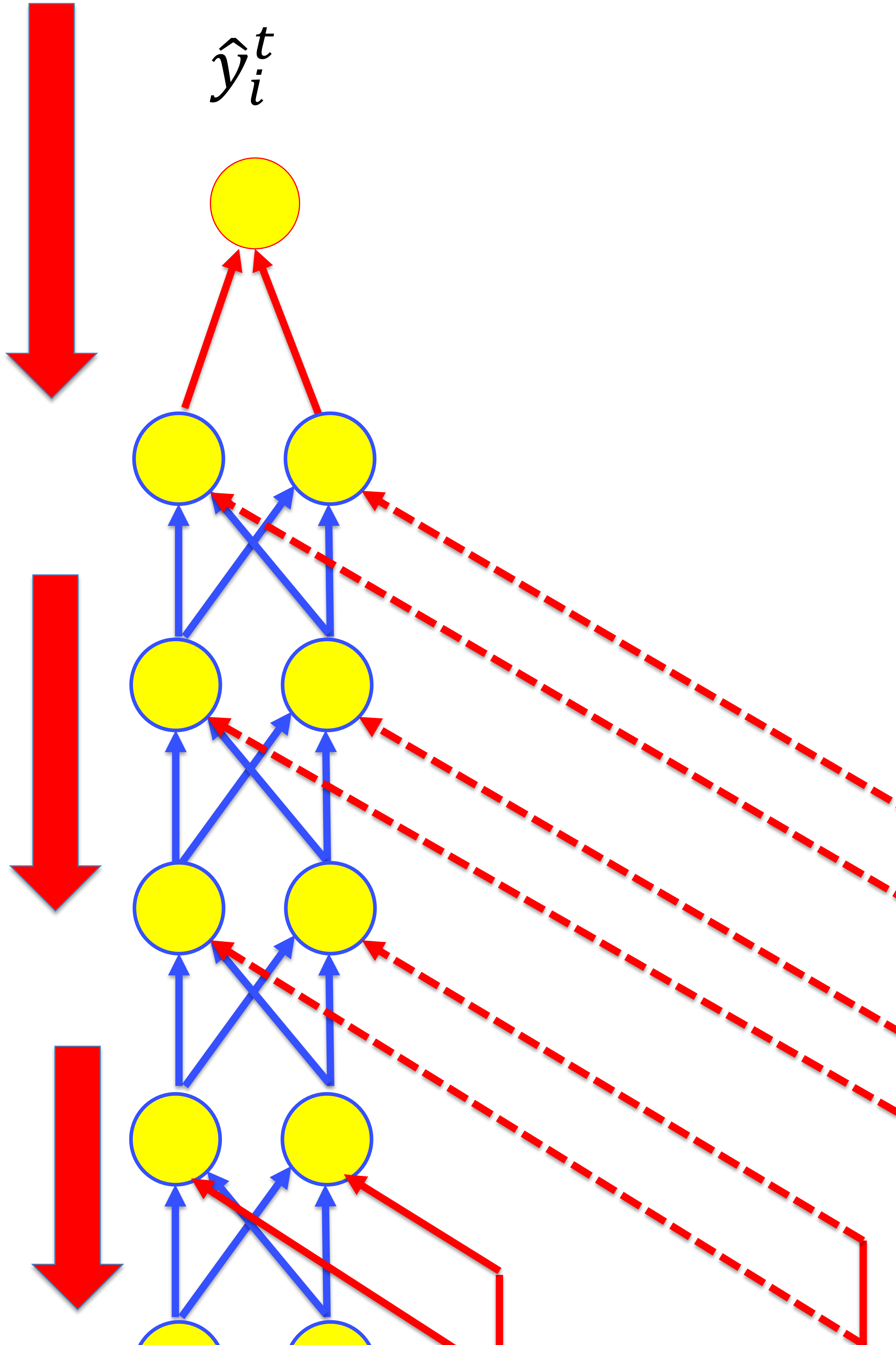
$$\delta_i^{(n-1)} = \sum_j w_{ji}^{(lat)} g'^{(n-1)}(a_i^{(n-1)})\delta_j^{(n)}$$



After N layers: each path contributes

$$\delta_i^{(t-N)} \sim g'^{(1)} w_{ji}^{(lat)} g'^{(2)} w_{ji}^{(lat)} \ldots g'^{(N-1)} w_{ji}^{(lat)} \delta_j^{(N)}$$

Many terms to be summed,
but most terms vanish if |g’w| < 1



Quiz: Vanishing Gradient Problem

The vanishing gradient problem of recurrent network means that

[ ] the derivative of the gain function vanishes: $g' = 0$

[ ] that the output error at time t contains only very little information about input at an earlier time step t-k if $k>10$

[ ] that $|g' \, w_{ji}^{(lat)}|^k \approx 0$ for $k>10$

7. Summary: Vanishing Gradient Problem

It is hard to learn long-term dependencies of sequence data
with a (normal) recurrent neural network
using backpropagation.

Artificial Neural Networks: Lecture 6
Sequences and Recurrent Networks

1. Sequences
2. Naïve solution: increase number of inputs
3. Long-term Dependencies
4. Recurrent Neural Networks
5. Unfolding the network in time
6. Backpropagation through time
7. The vanishing Gradient Problem
8. Long Short-Term Memory (LSTM)

8. Long short-term memory (LSTM)

Two basic ideas

(i) Hard to keep memory in a recurrent network
→ define explicit memory units

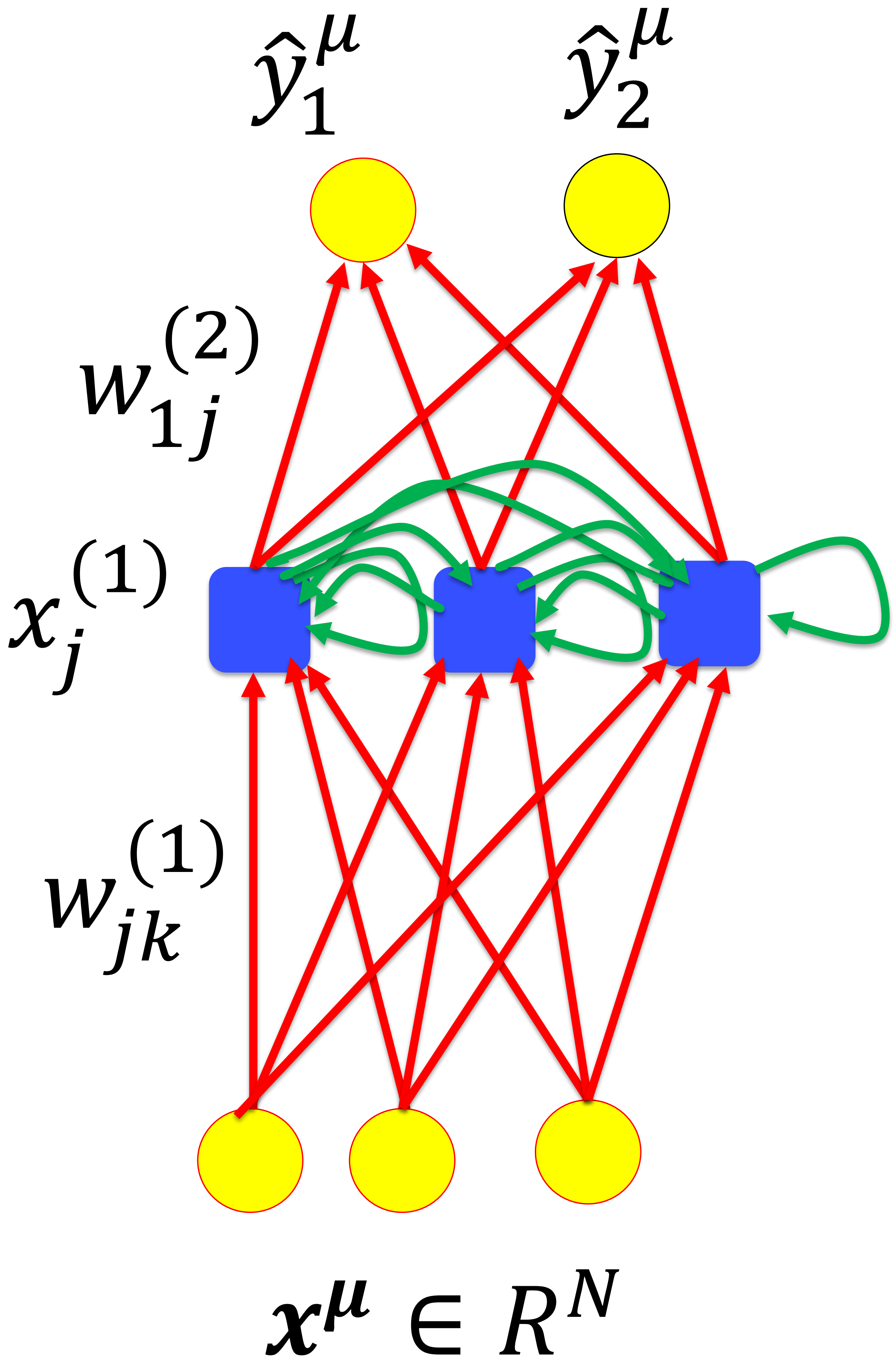
(ii) Avoid the vanishing gradient problem

→ make sure that $g'^{(1)} w_{ji}^{(lat)} = 1$

8. Long short-term memory (LSTM)

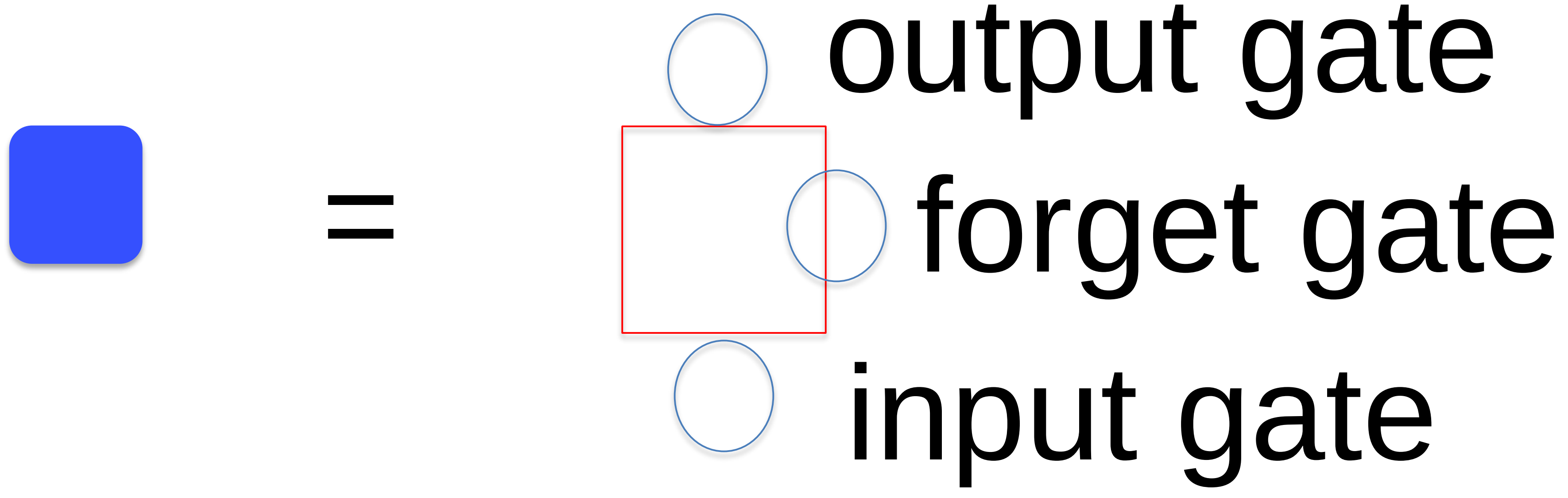
Replace neurons in hidden layer
by memory units

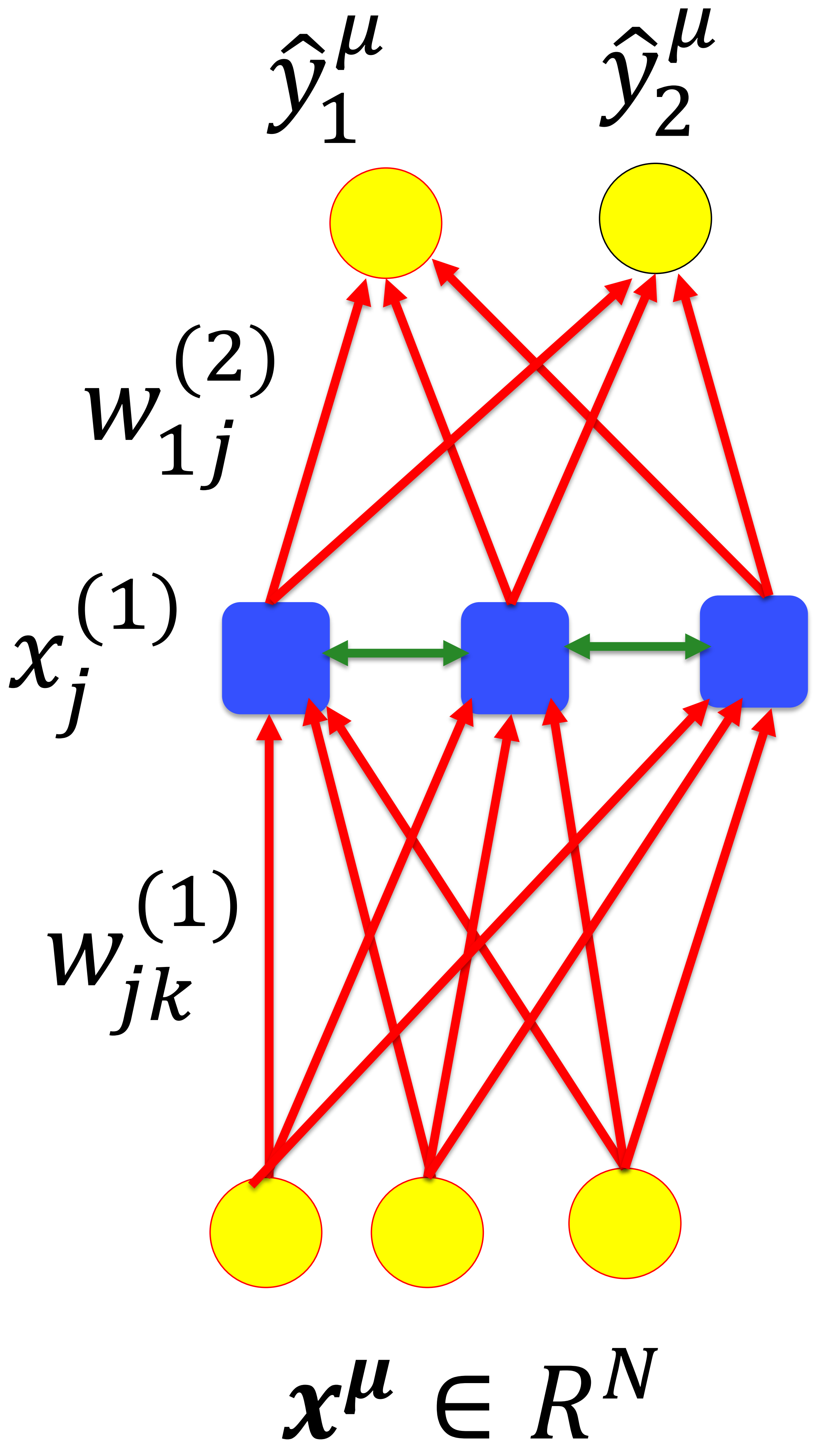
= 1 memory unit
= 1 LSTM unit



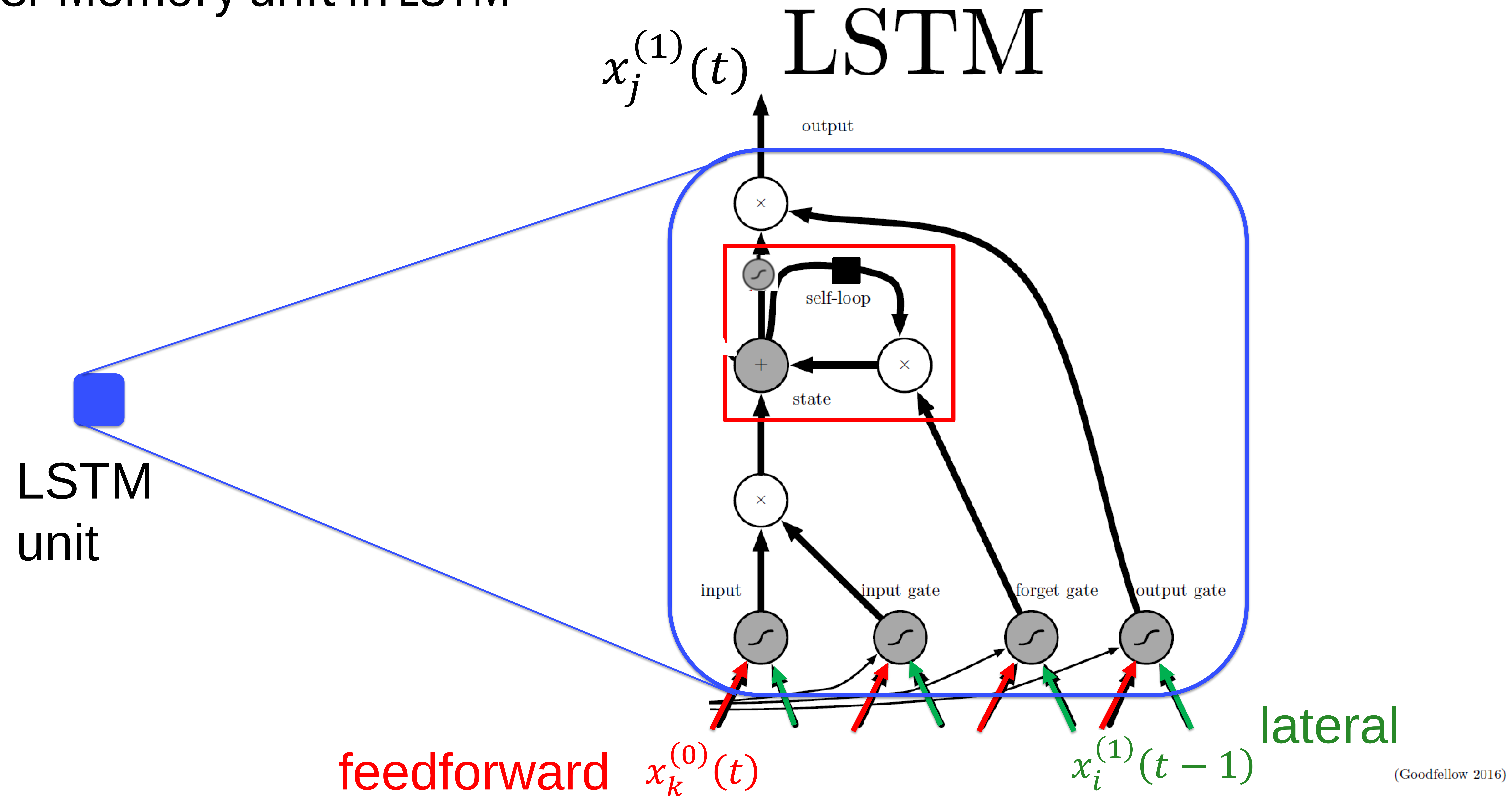
8. Long short-term memory (LSTM)

Replace neurons in hidden layer by LSTM units

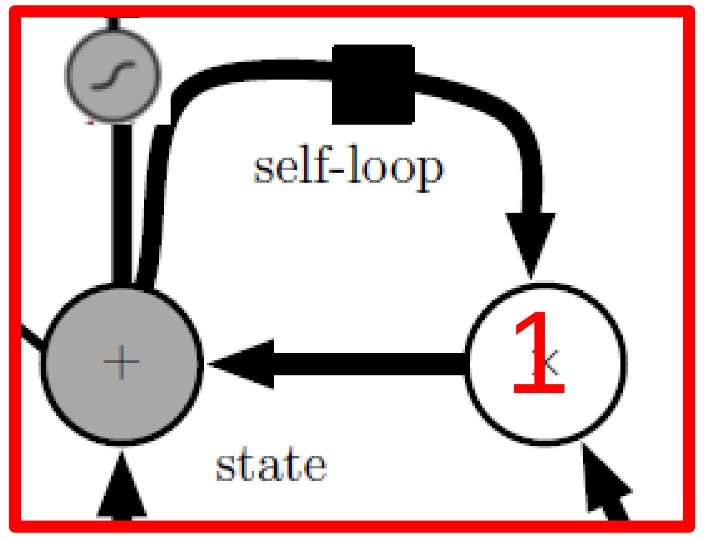




8. Memory unit in LSTM



8. Memory unit in LSTM



Internal state s of memory

$$s_j^{(1)}(t) = 1 \cdot s_j^{(1)}(t-1)$$

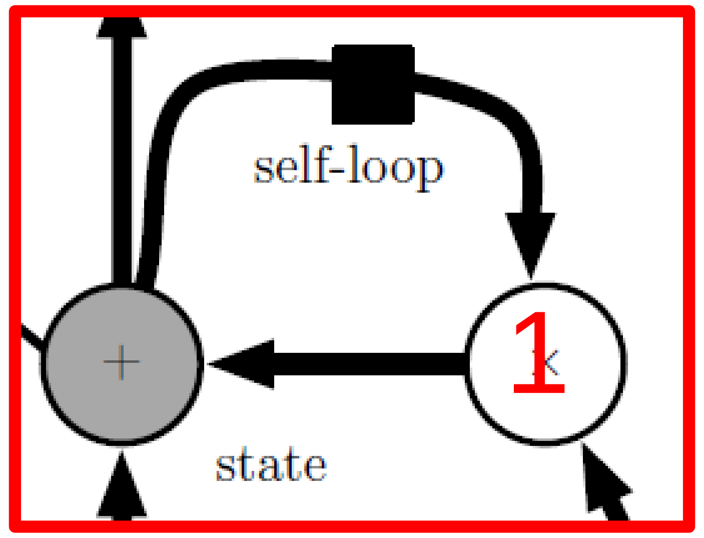
Compare: $x_j^{(1)}(t) = g[w \cdot s_j^{(1)}(t-1)]$

set *g(a)=a*
and *w=1*

8. Memory unit in LSTM: writing into memory

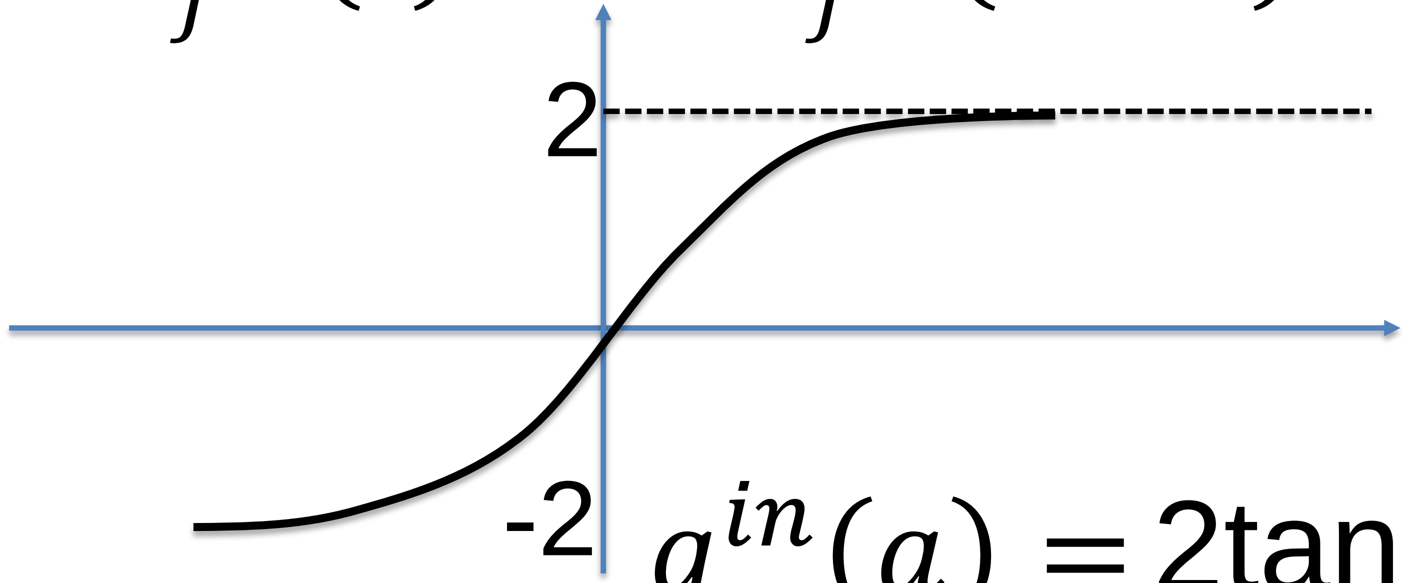
‘write in memory when useful for the task’

Internal state s of memory



$s_j^{(1)}(t) = 1 \cdot s_j^{(1)}(t-1)$ + (gated) ***input***

$s_j^{(1)}(t) = 1 \cdot s_j^{(1)}(t-1)$ + (gated) $g^{in}[\sum_k w_{jk}^{(1)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat)}\, x_i^{(1)}(t-1) - \vartheta_j]$



$g^{in}(a) = 2\tanh(a)$

$x_k^{(0)}(t)$

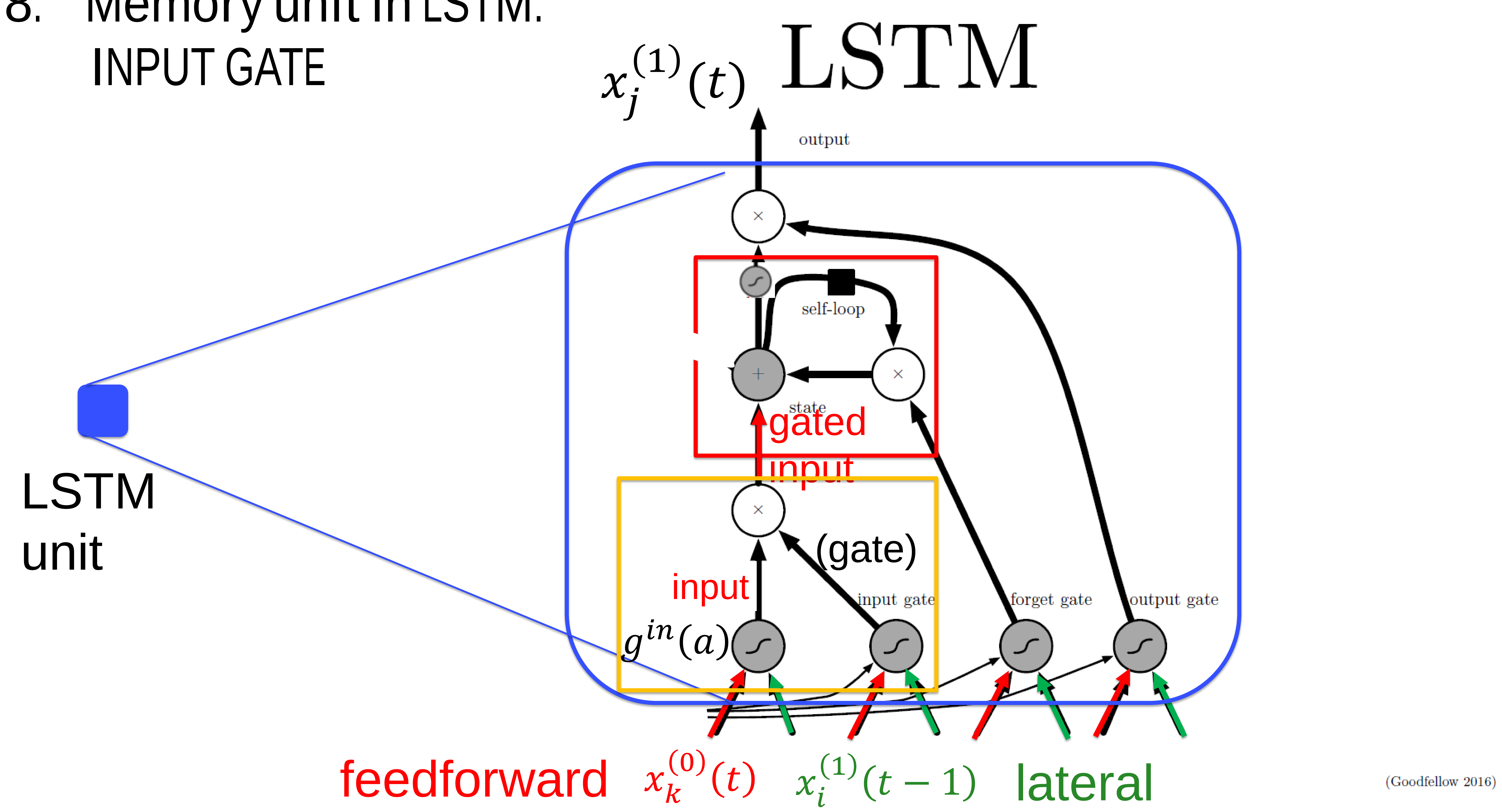
feedforward

input

$x_i^{(1)}(t-1)$

lateral

8. Memory unit in LSTM:
INPUT GATE



8. LSTM – input gate

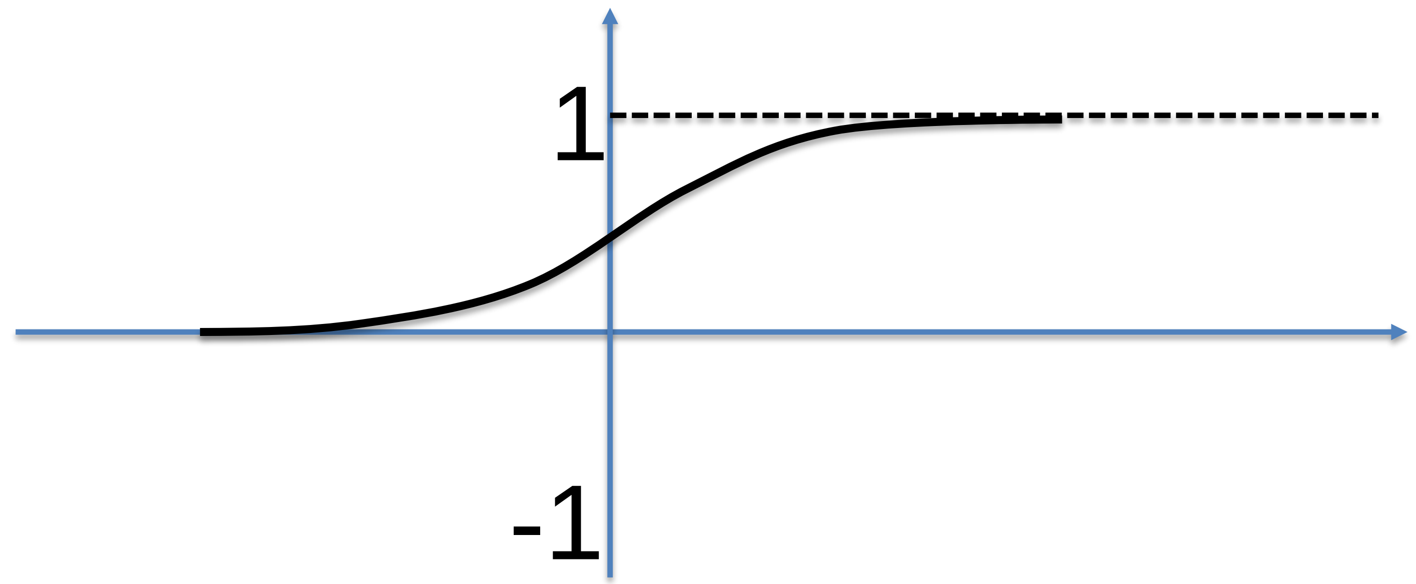
Internal state s of memory

$$s_j^{(1)}(t) = 1 \cdot s_j^{(1)}(t-1) + \text{(gated) } g^{in}\underbrace{[\textstyle\sum_k w_{jk}^{(1)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat)}\, x_i^{(1)}(t-1) - \vartheta_j]}_{\text{input}}$$

$$s_j^{(1)}(t) = 1 \cdot s_j^{(1)}(t-1) + \quad Y_j^{(1)} g^{in}[\textstyle\sum_k w_{jk}^{(1)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat)}\, x_i^{(1)}(t-1) - \vartheta_j]$$

Gating variable Y of input

$$Y_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1,Y)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat,Y)}\, x_i^{(1)}(t-1) - \vartheta_j^{(1,Y)} - \mathbf{1}\right)$$

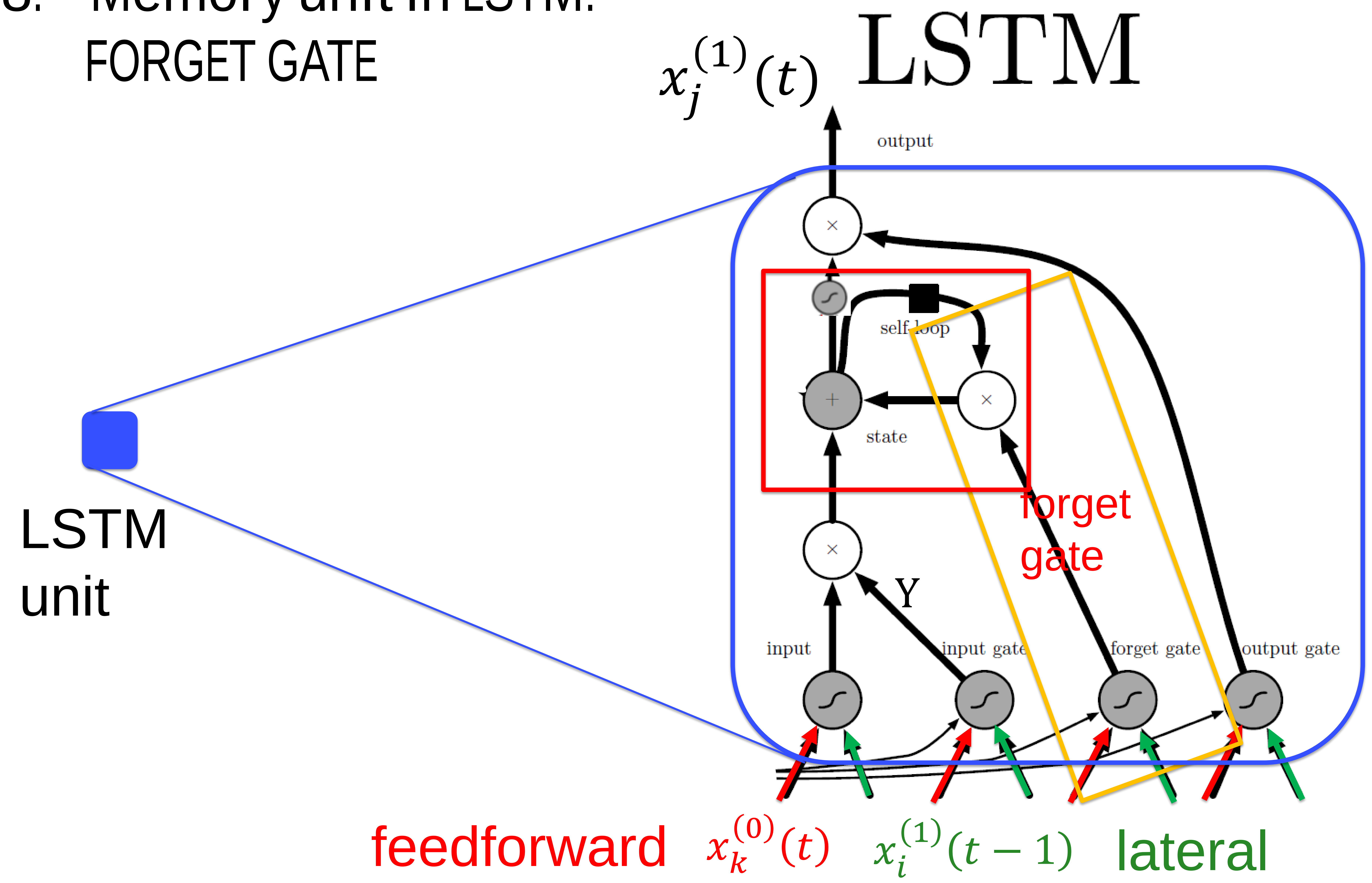


a

$x_k^{(0)}(t)$ ***input*** $x_i^{(1)}(t-1)$

feedforward lateral

8. Memory unit in LSTM:
FORGET GATE



(Goodfellow 2016)

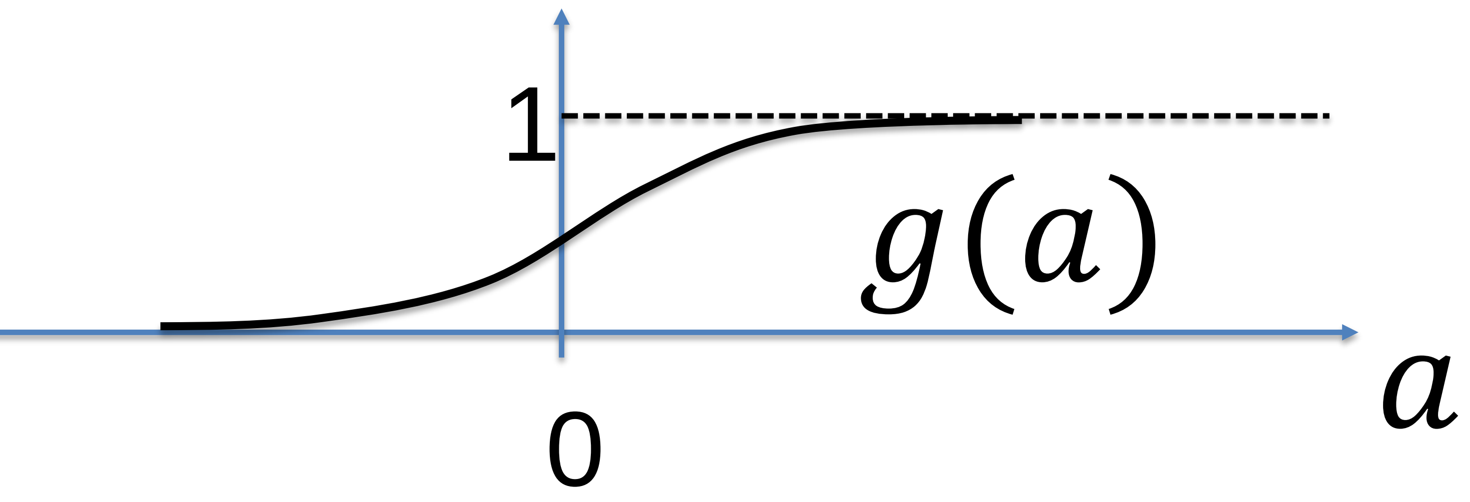
8. LSTM – Forgetting gate (initialize at 1 or close to 1)

$$s_j^{(1)}(t) = 1 \cdot s_j^{(1)}(t-1) + \quad Y_j^{(1)}\, g[\textstyle\sum_k w_{jk}^{(1)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat)}\, x_i^{(1)}(t-1) - \vartheta_j]$$

$$s_j^{(1)}(t) = f \cdot s_j^{(1)}(t-1) + \quad Y_j^{(1)}\, g[\textstyle\sum_k w_{jk}^{(1)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat)}\, x_i^{(1)}(t-1) - \vartheta_j]$$

Gating variable *f* for forgetting

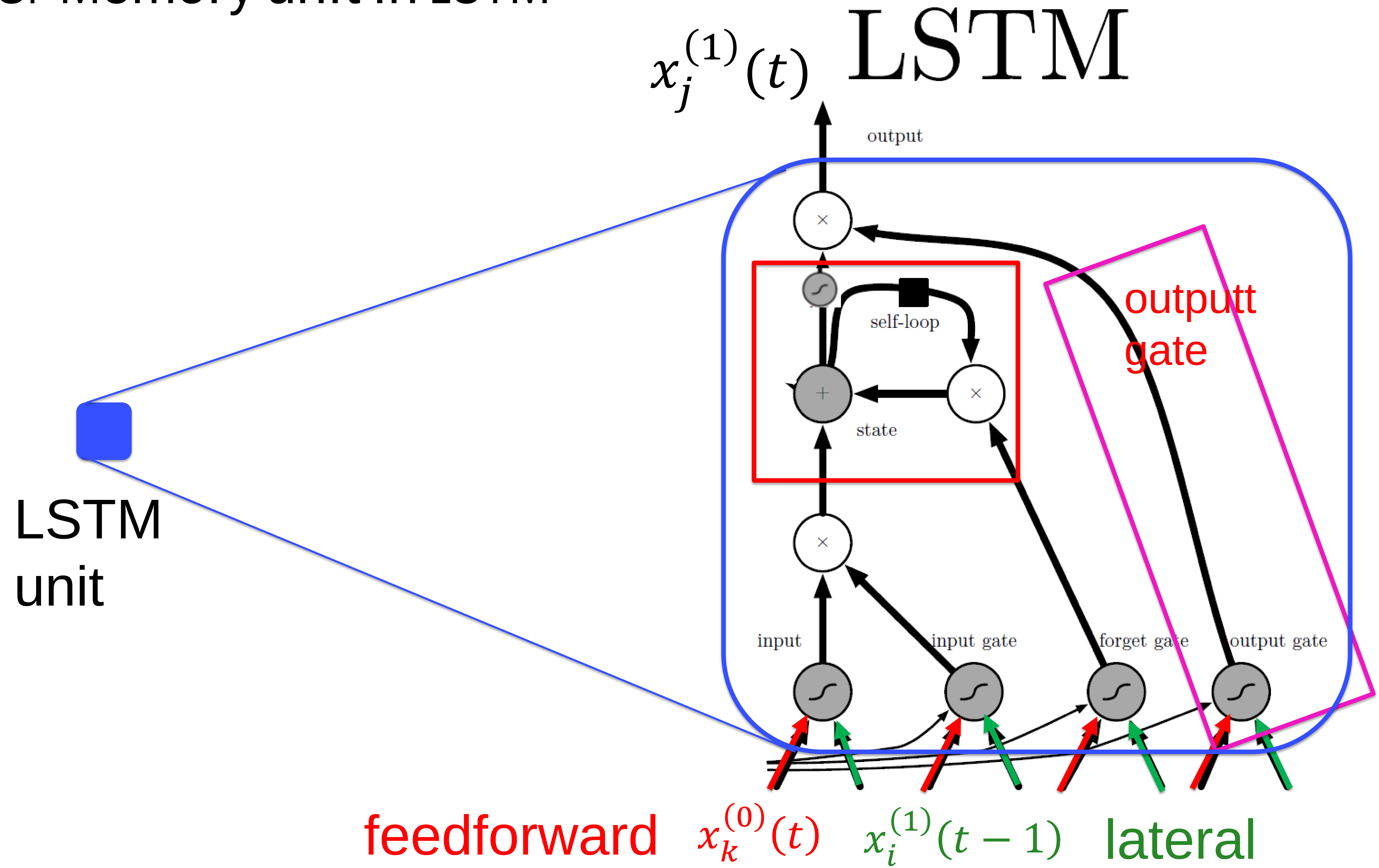
$$f_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1,f)}\, x_k^{(0)}(t) \quad + \sum_i w_{ji}^{(lat,f)}\, x_i^{(1)}(t-1) - \vartheta_j^{(1,f)} + \mathbf{1} \right)$$



$x_k^{(0)}(t)$ ***input*** $x_i^{(1)}(t-1)$

feedforward lateral

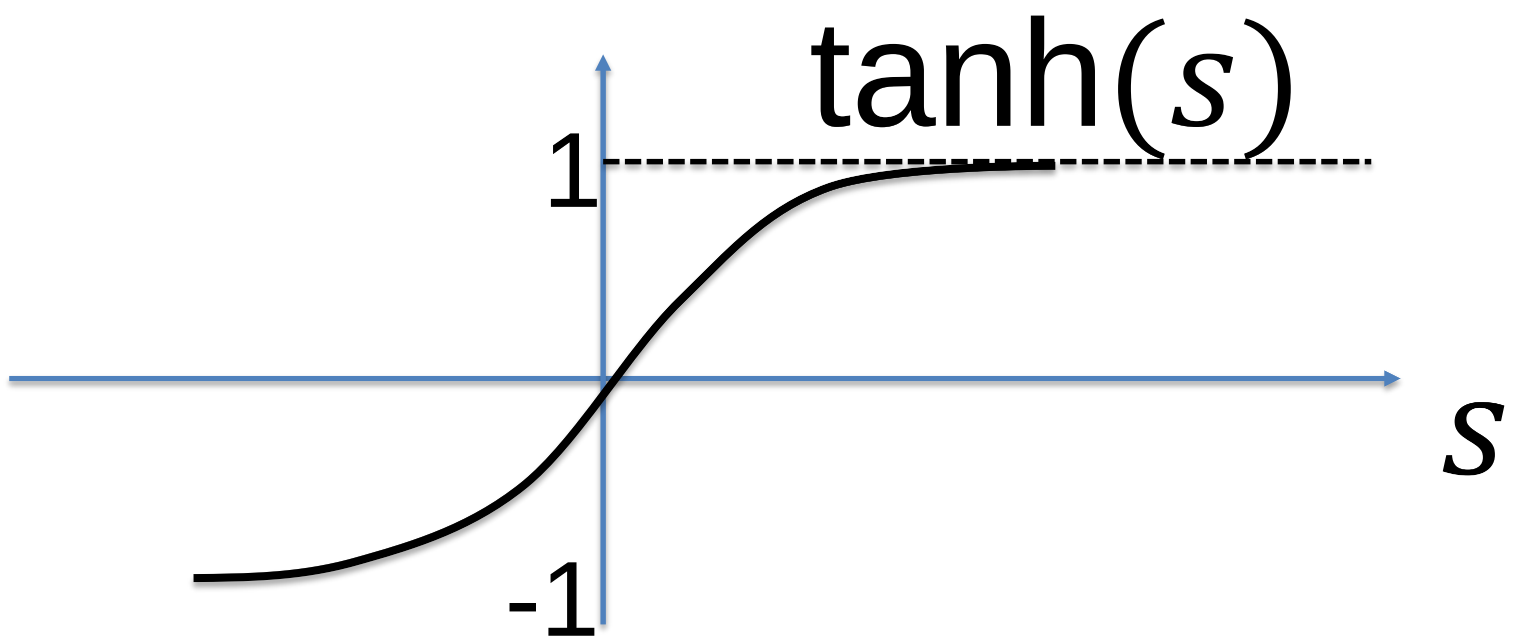
8. Memory unit in LSTM



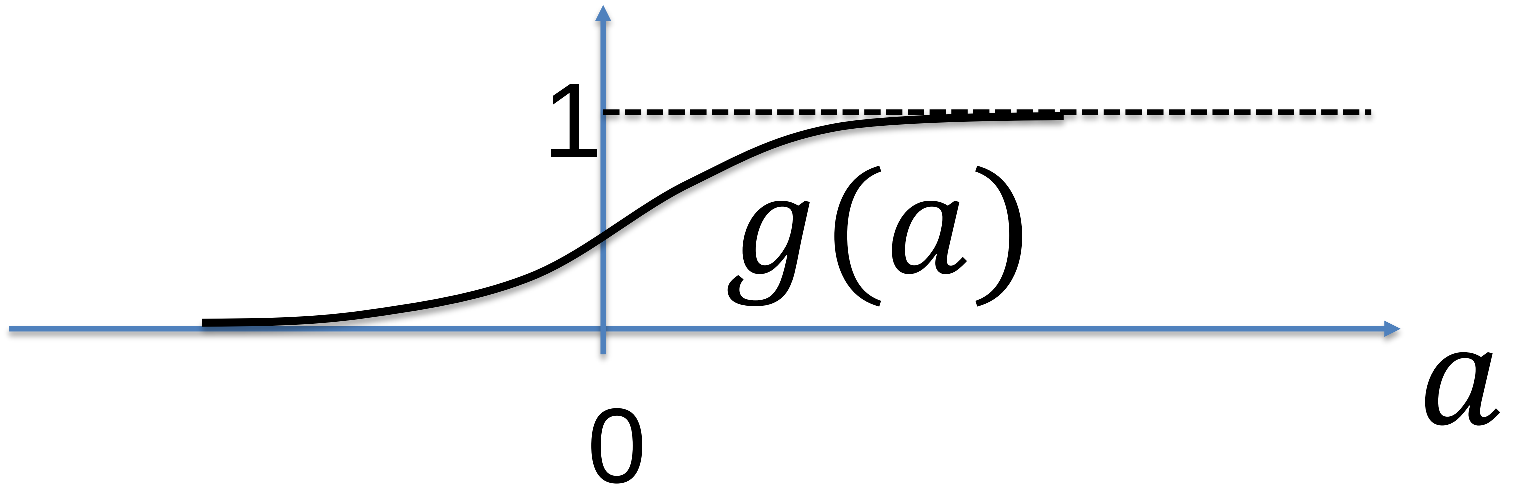
(Goodfellow 2016)

8. LSTM –output gate

$$x_j^{(1)}(t) = \quad q_j^{(1)} \tanh[s_j^{(1)}(t)]$$



Gating variable q for output $\quad g(a) = 0.5[1 + tanh(a)]$



$$q_j^{(1)}(t) = g\left(\sum_k w_{jk}^{(1,q)} \; x_k^{(0)}(t) \;\; + \sum_i w_{ji}^{(lat,q)} \; x_i^{(1)}(t-1) - \vartheta_j^{(1,f)} - 1 \right)$$

$x_k^{(0)}(t)$ ***input*** $x_i^{(1)}(t-1)$

feedforward lateral

8. Memory unit in LSTM

Remark (memory block):
1 LSTM unit can have several state variables, controlled by a shared gates

LSTM
unit

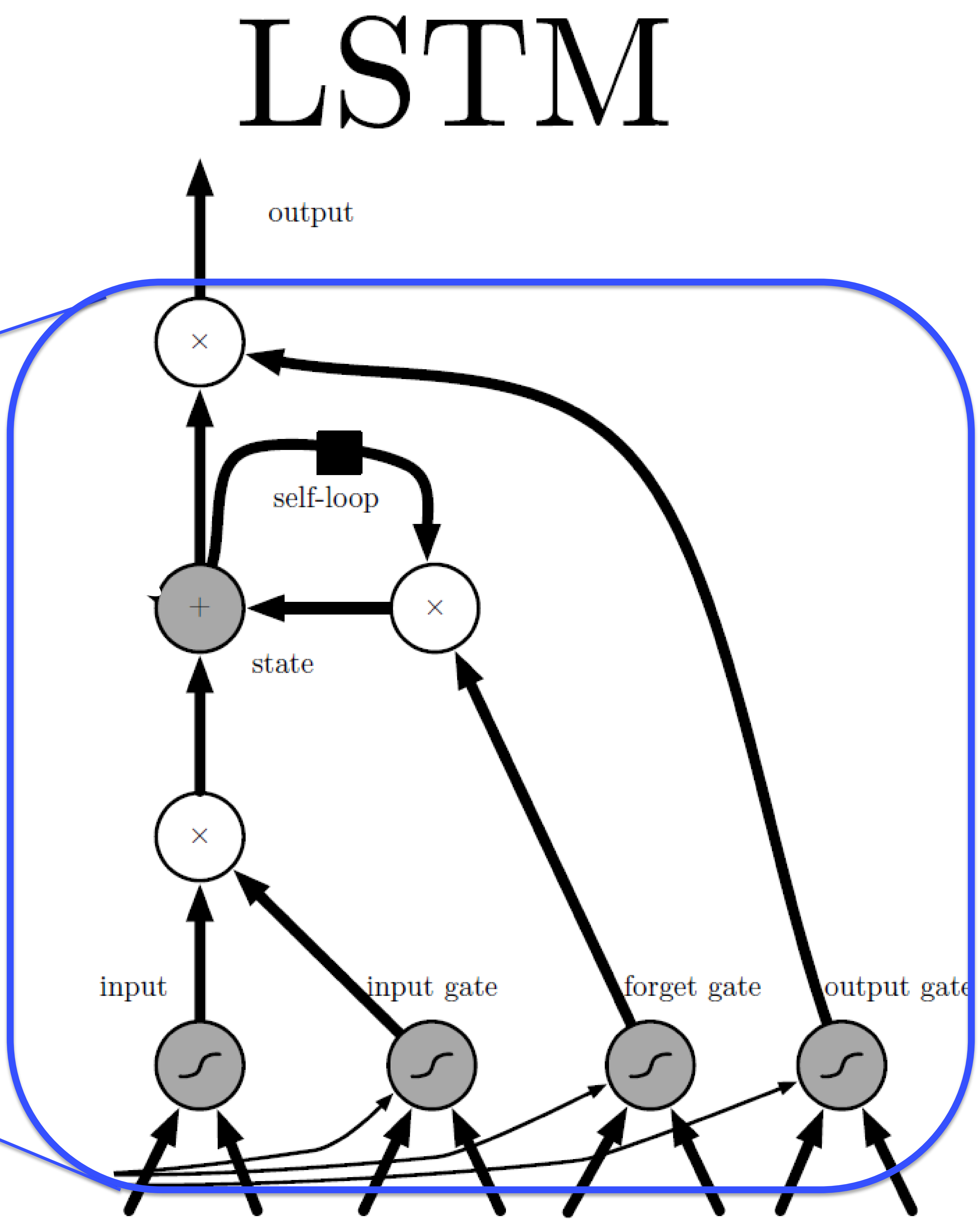


Figure 10.16

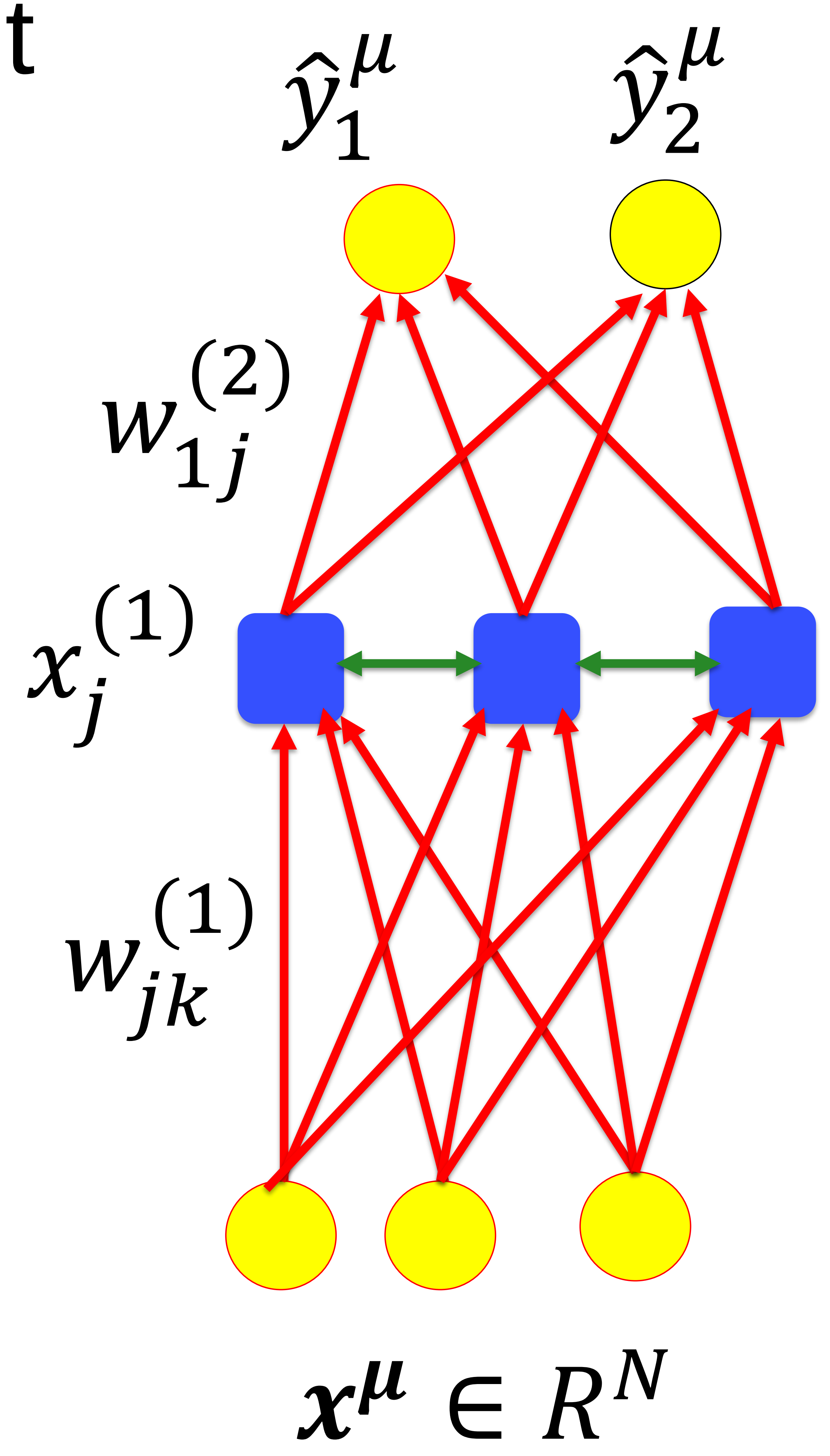
8. LSTM networks

■ = 1 memory unit
= 1 LSTM unit

Trained with BackProp

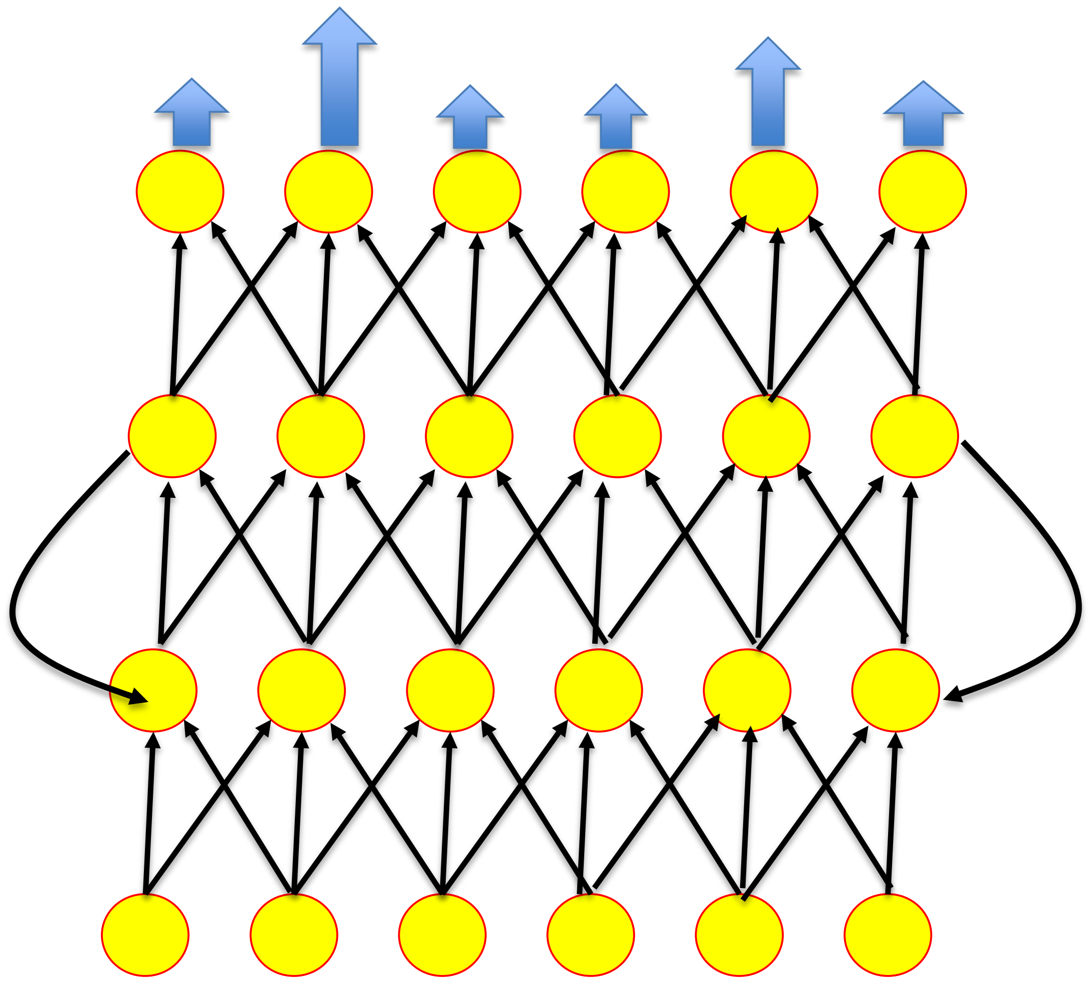
State of the art for
- text translation by machines
- handwriting recognition
- speech recognition
- image captioning

e.g. Xu et al. 2015



Deep networks with recurrent connections (Lecture 1)

‘a man sitting on a couch with a dog’



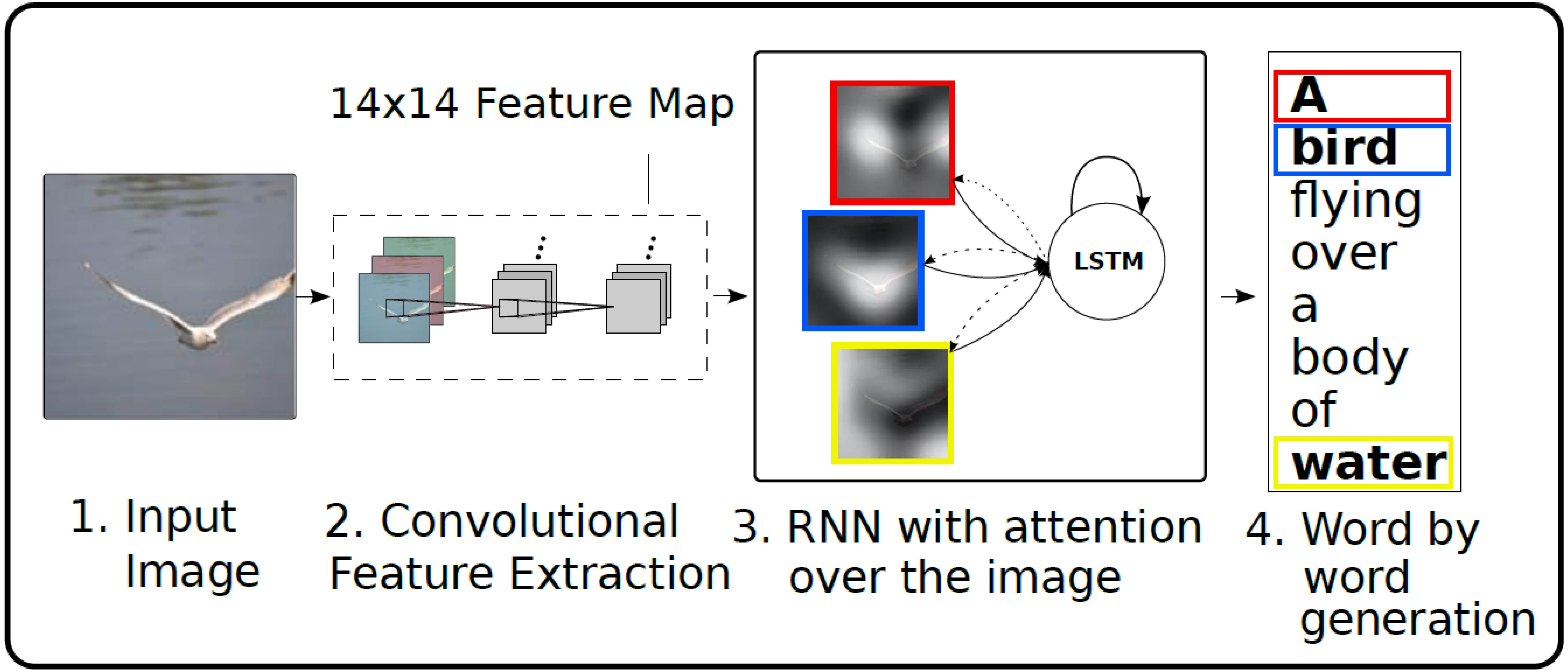
Network desribes the image with the words:

‘a man sitting on a couch with a dog’

(Fang et al. 2015)

8. LSTM in Deep networks with recurrent connections

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in Sections 3.1 & 5.4



Xu et al. (2015), *Show, attend and tell: Neural image caption generation…, ICML*

Artificial Neural Networks: Lecture 6

Sequences and Recurrent Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Objectives for today:

- Why are sequences important?
  they are everywhere; labeling is (mostly) for free
- Long-term dependencies in sequence data
  unknown time scales, fast and slow
- Sequence processing with feedforward models
  corresponds to n-gram=finite memory
- Sequence processing with recurrent models
  potentially unlimited memory, but:
- Vanishing Gradient Problem
  error information does not travel back beyond a few steps
- Long-Short-Term Memory (LSTM)
  explicit memory units keep information beyond a few steps
- Application: Music generation

The end