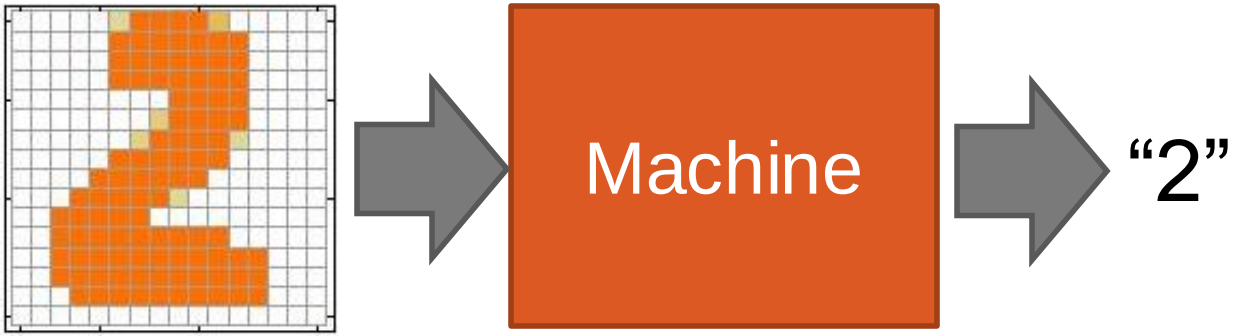
Deep Learning

Deep Learning

- Deep learning is kind of hard. Why bother with it?
- Amazing results… in speech, NLP, vision/multimodal work
- Does its own feature selection!
- The big players (Google, Facebook, Baidu, Microsoft, IBM…) are doing a lot of this
- The hot new thing?
- Actually, many of the architectures that we'll talk about were invented in the 1980s and 1990s
- What's new is hardware that can use these architectures at scale.

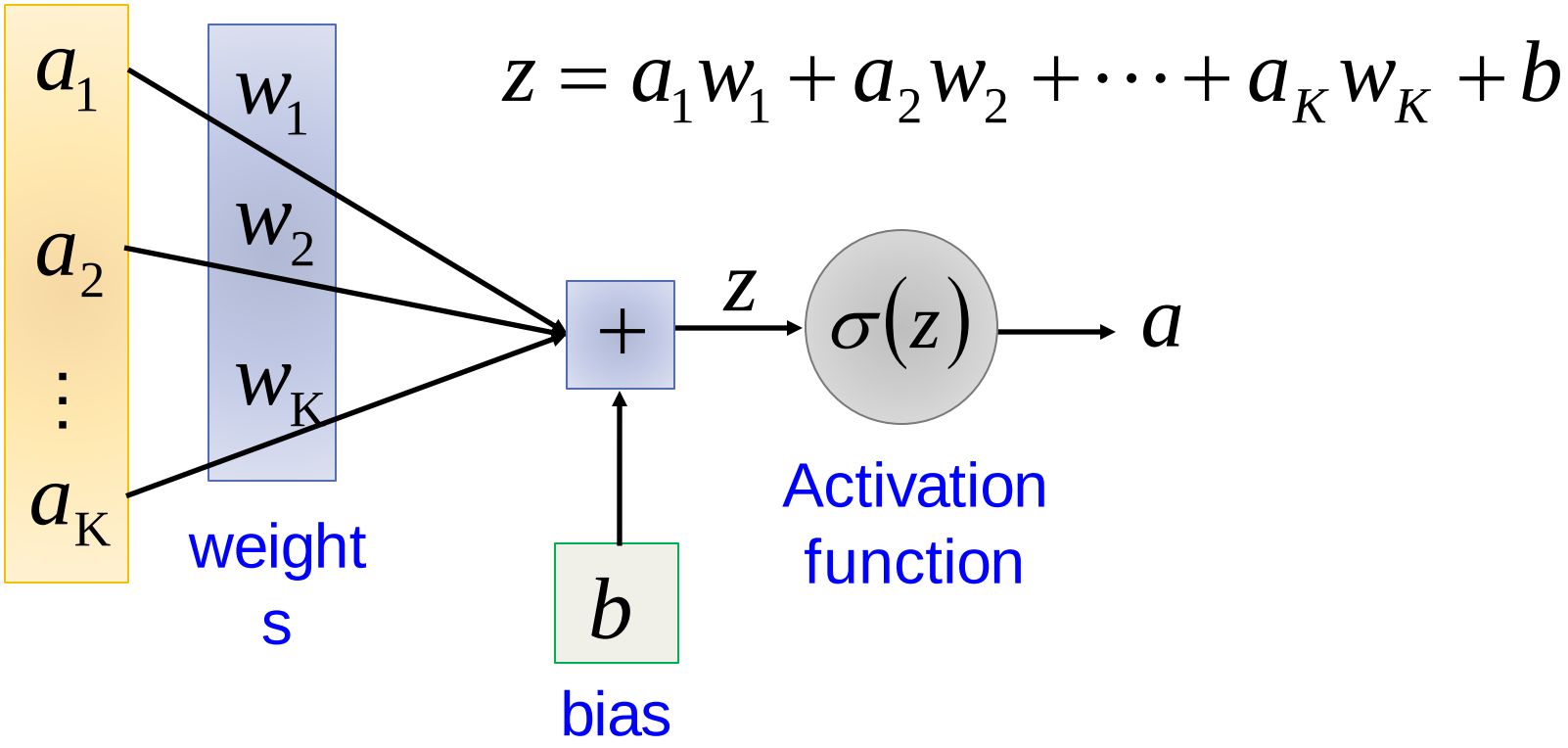
Example Application

- Handwriting Digit Recognition

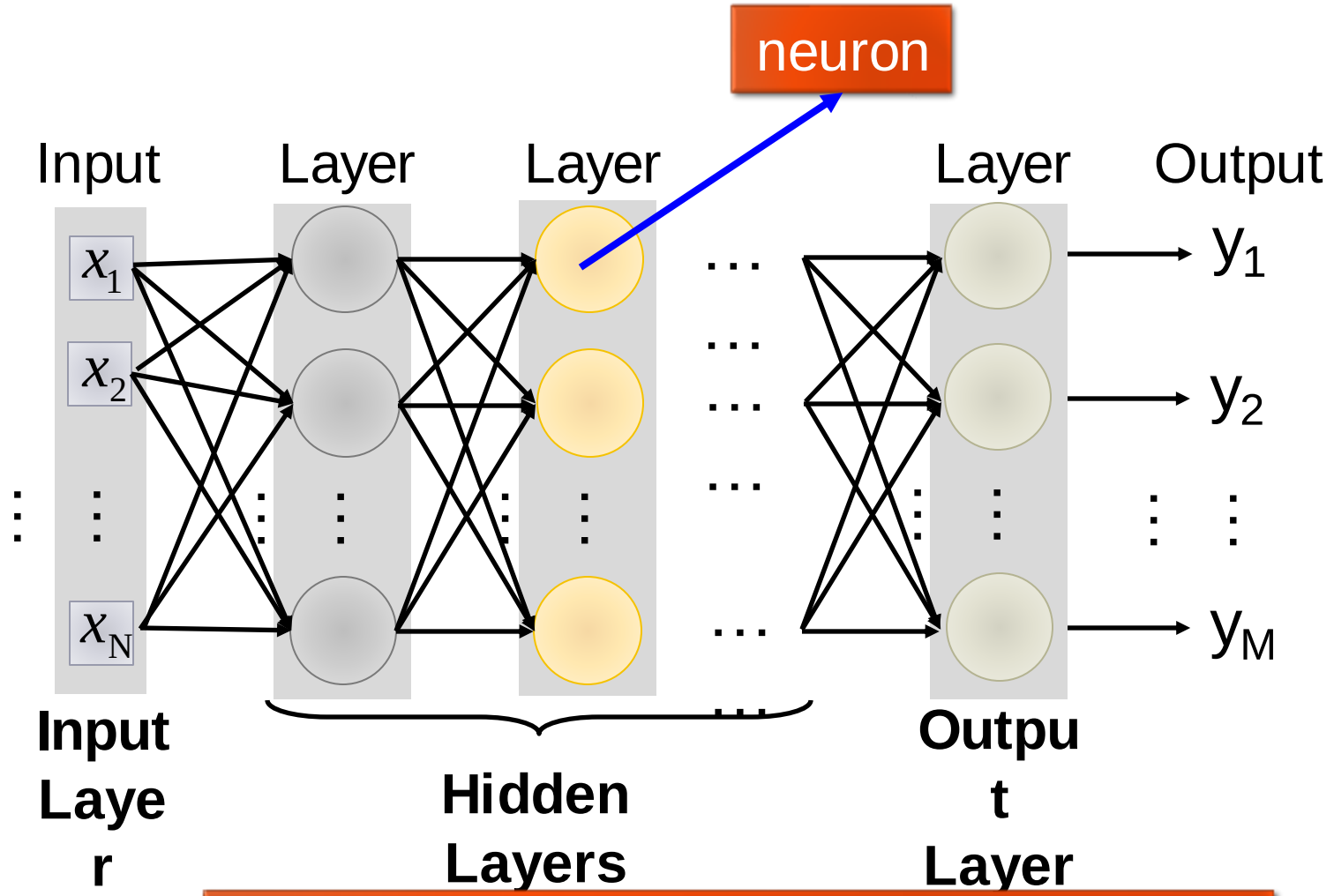


Element of Neural Network

Neuron $f: R^K \rightarrow R$

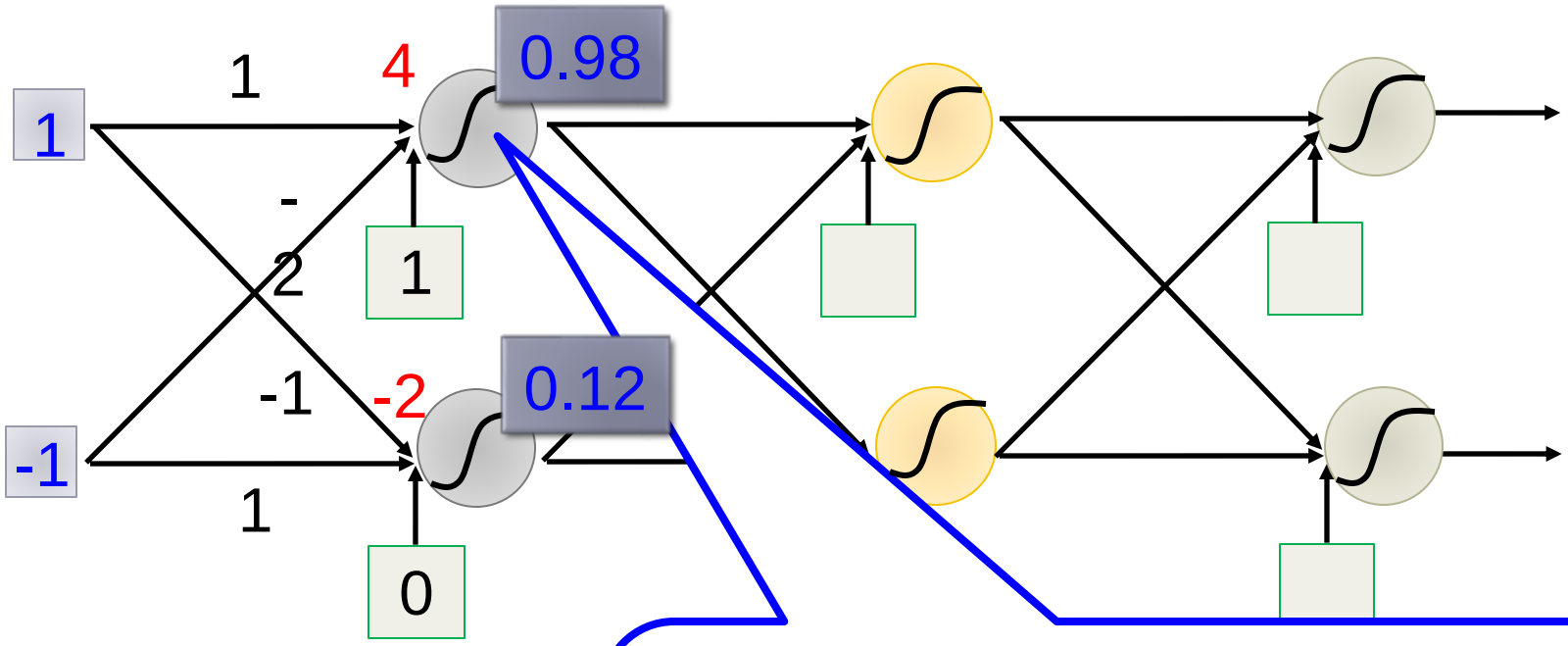


Neural Network

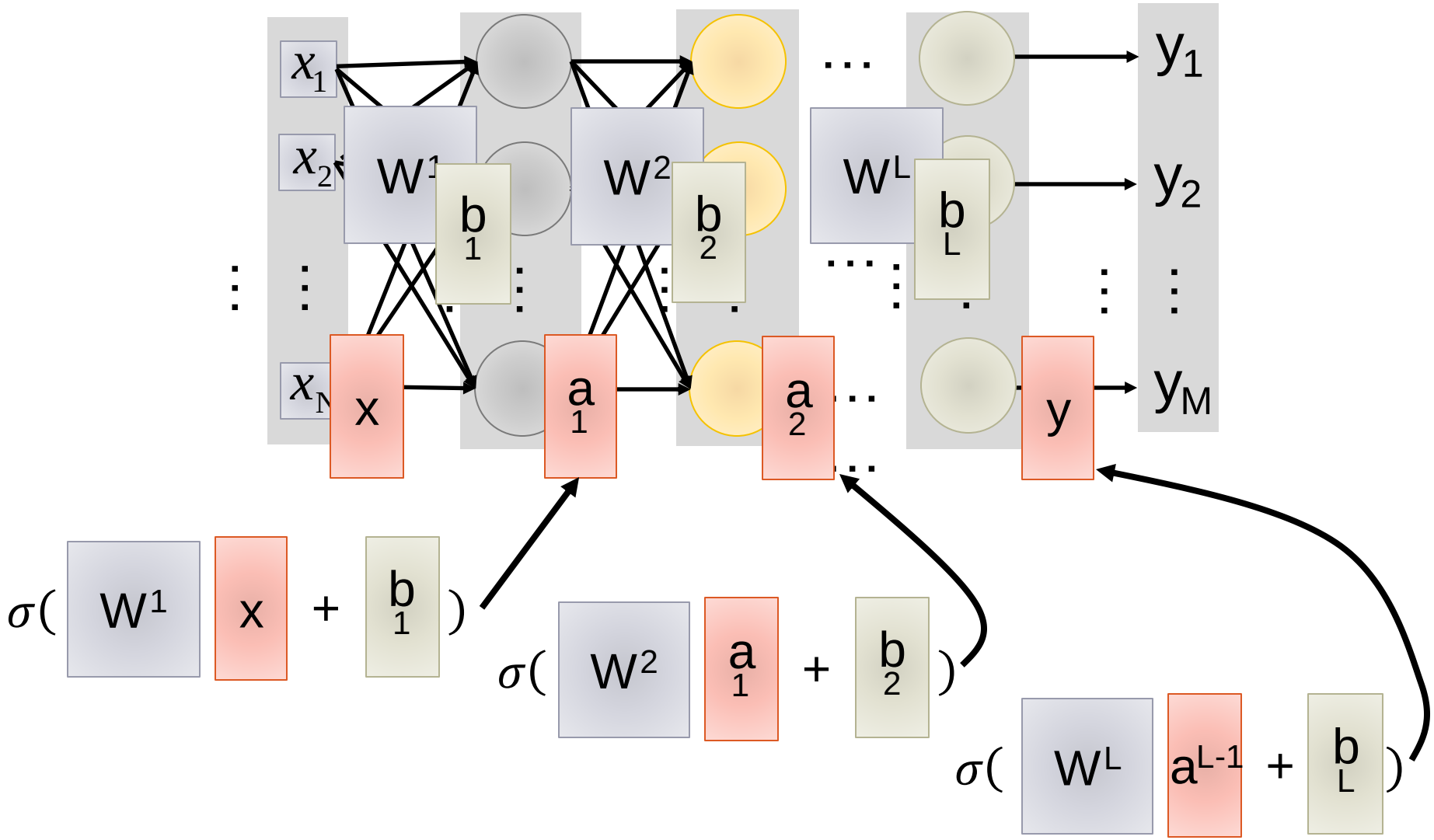


Deep means many hidden layers

Neural Network



Neural Network

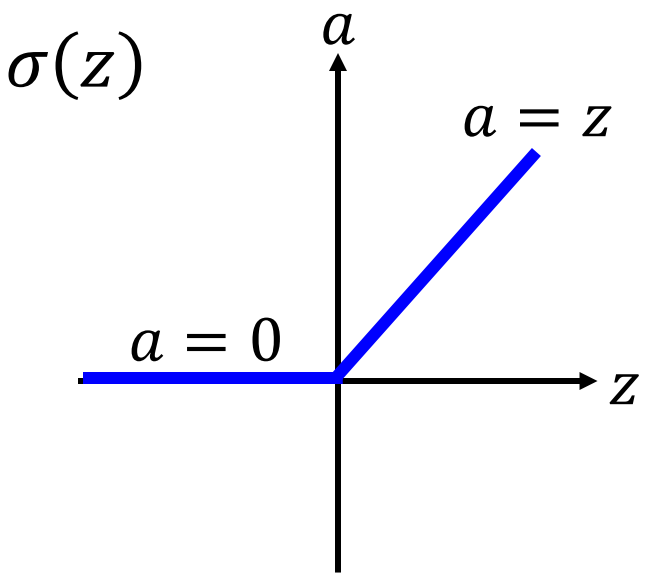


Training DNN

New Activation Function

ReLU

- Rectified Linear Unit (ReLU)



Reason:

1. Fast to compute

2. Vanishing gradient problem

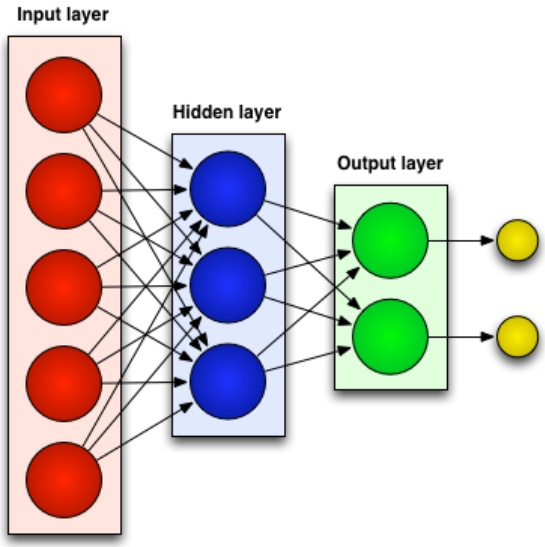
$$f'(x) = \begin{cases} 1 if\ x >= 0 \\ 0 if\ x < 0 \end{cases}$$

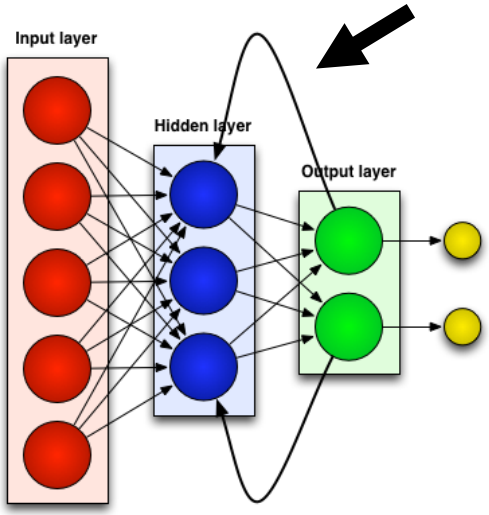
$$\sigma'(z) = \sigma(z) * (1 - \sigma(z))$$

Neural Networks

- **Generally there are two kinds of neural networks:**

  - **Feedforward Neural Networks:**
    - ✓ **connections between the units do not form a cycle**

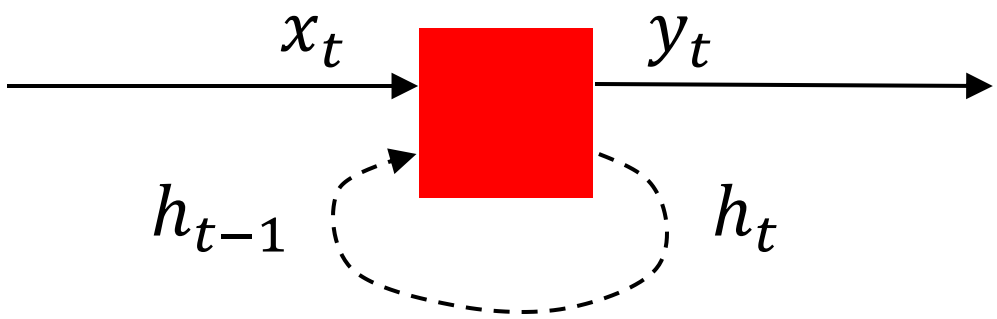




  - **Recurrent Neural Network:**
    - ✓ **connections between units form cyclic paths**

Recurrent Neural Networks

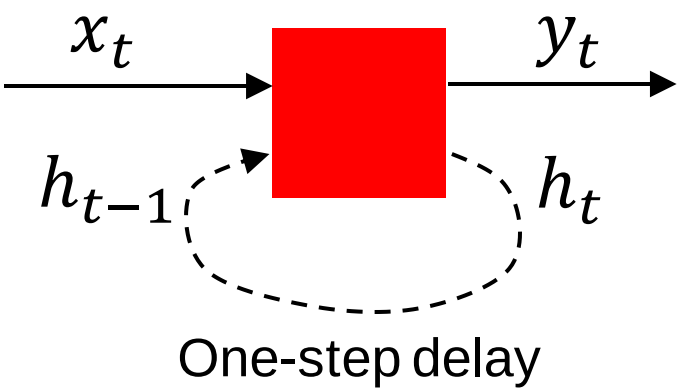
Recurrent networks introduce cycles and a notion of time.

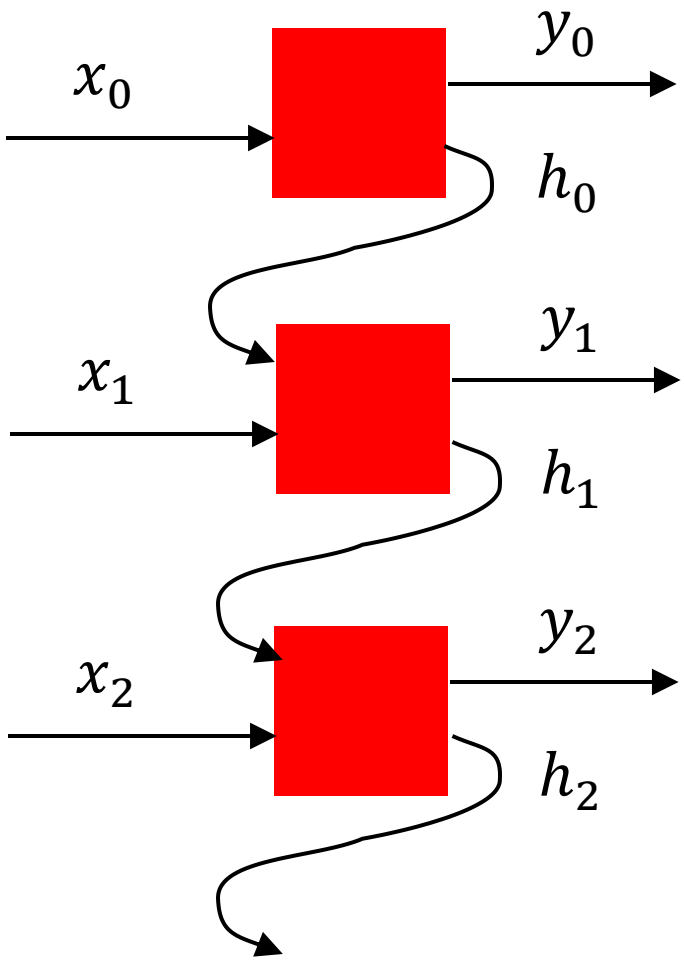


- They are designed to process sequences of data $x_1, \ldots, x_n$ and can produce sequences of outputs $y_1, \ldots, y_m$.

Unrolling RNNs

RNNs can be unrolled across multiple time steps.



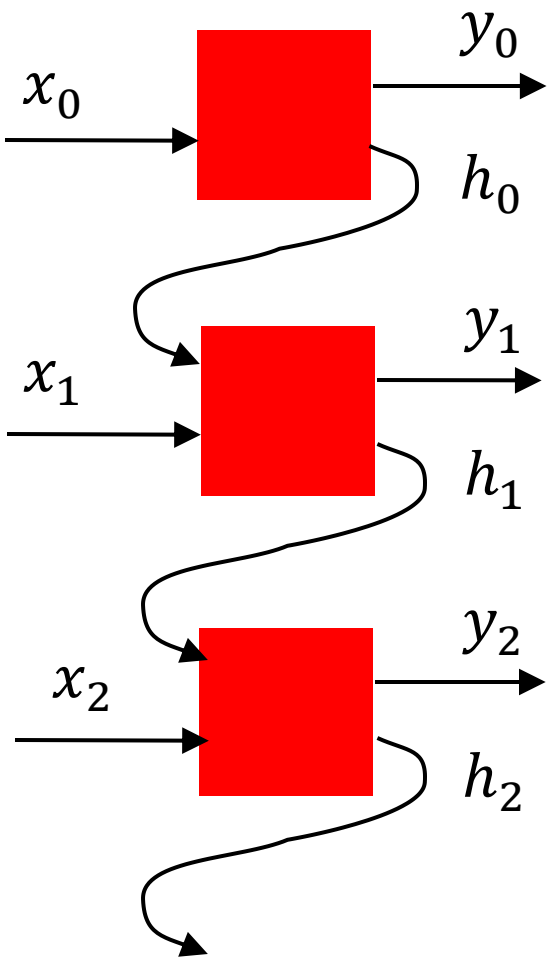


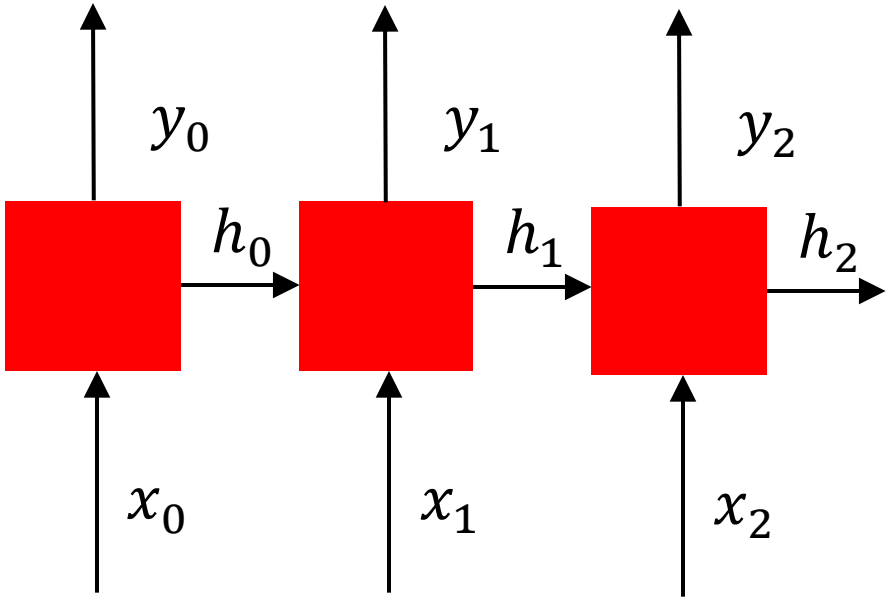
This produces a DAG which supports backpropagation.

But its size depends on the input sequence length.

Unrolling RNNs

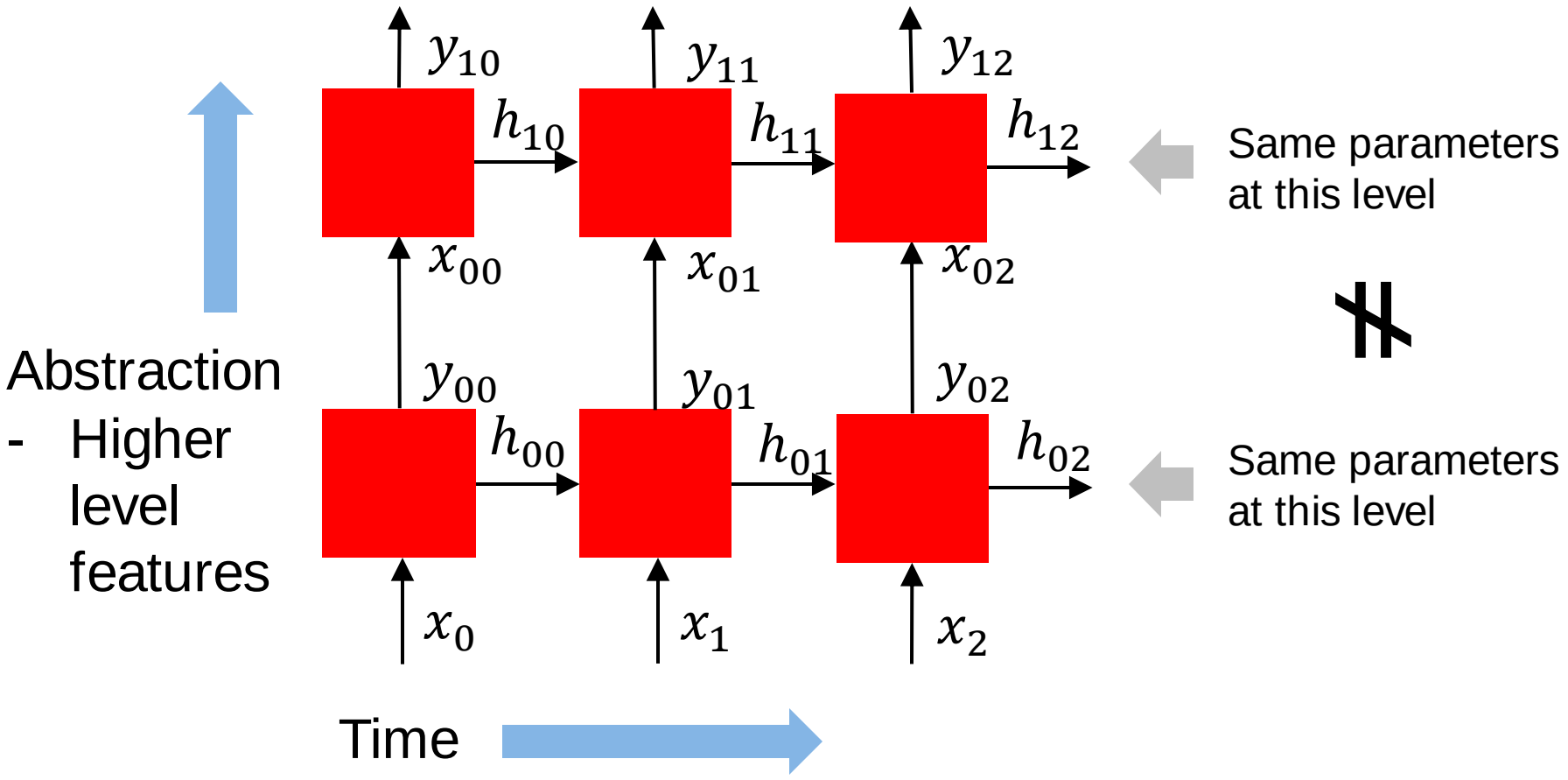
Usually drawn as:





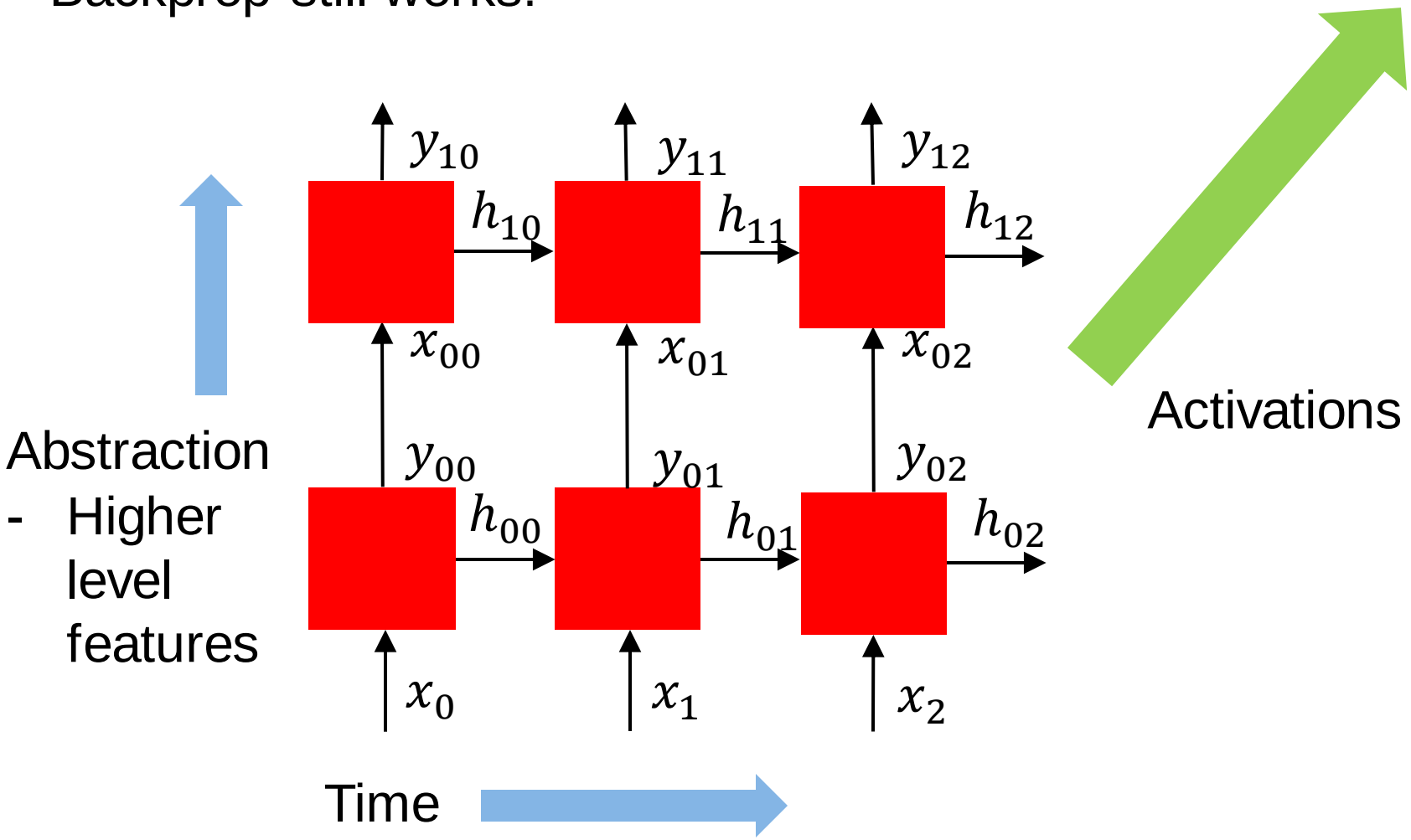
RNN structure

Often layers are stacked vertically (deep RNNs):



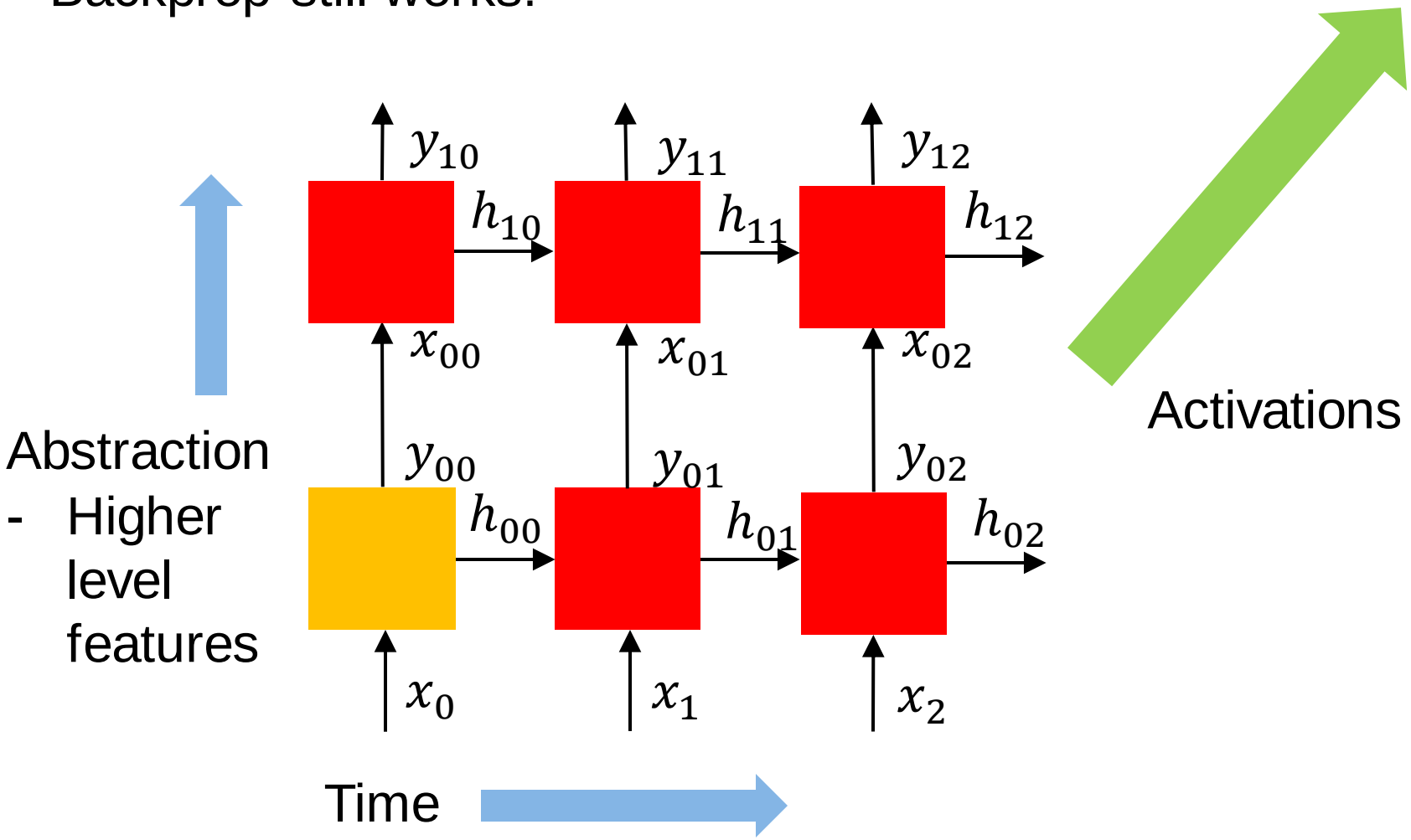
RNN structure

Backprop still works:



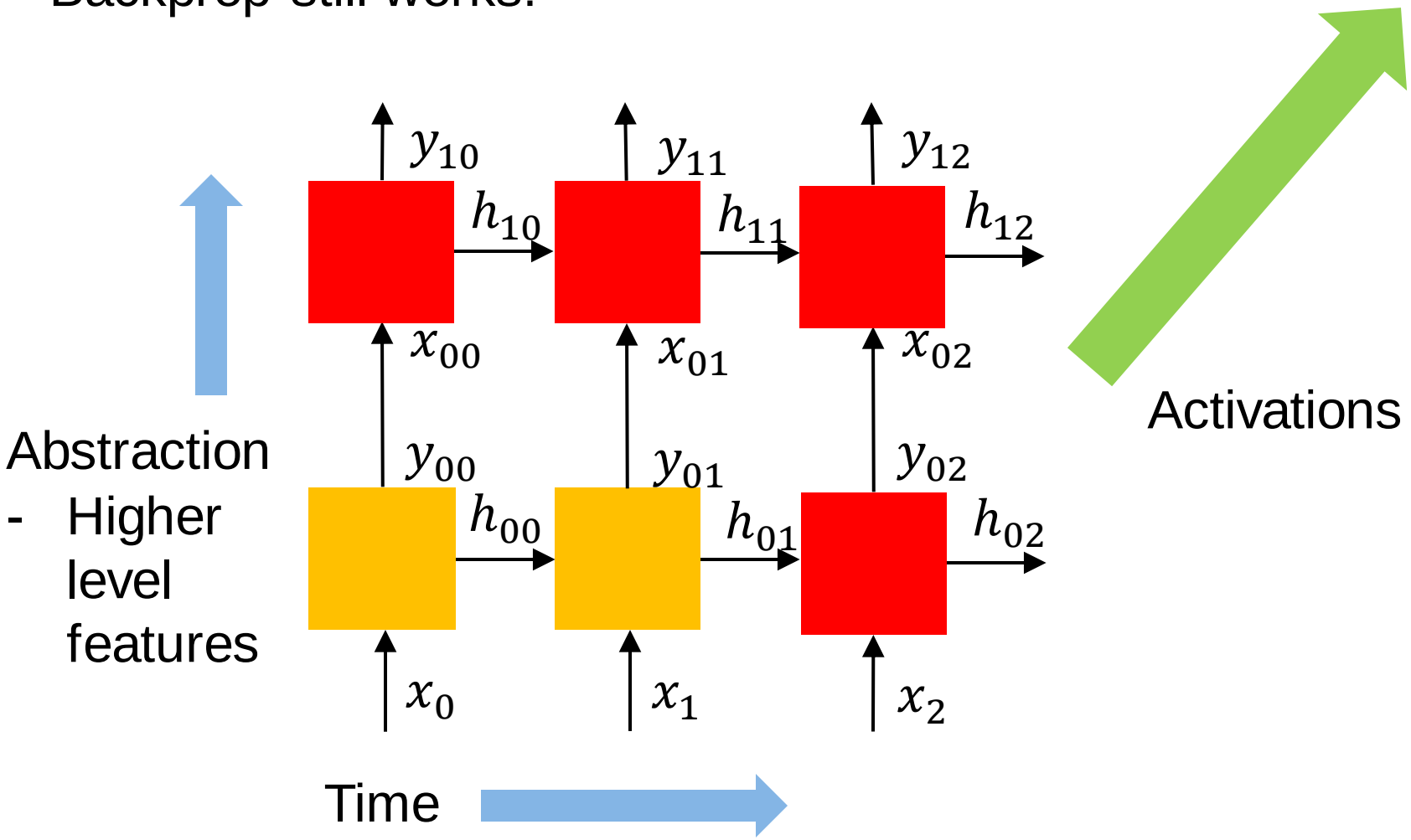
RNN structure

Backprop still works:



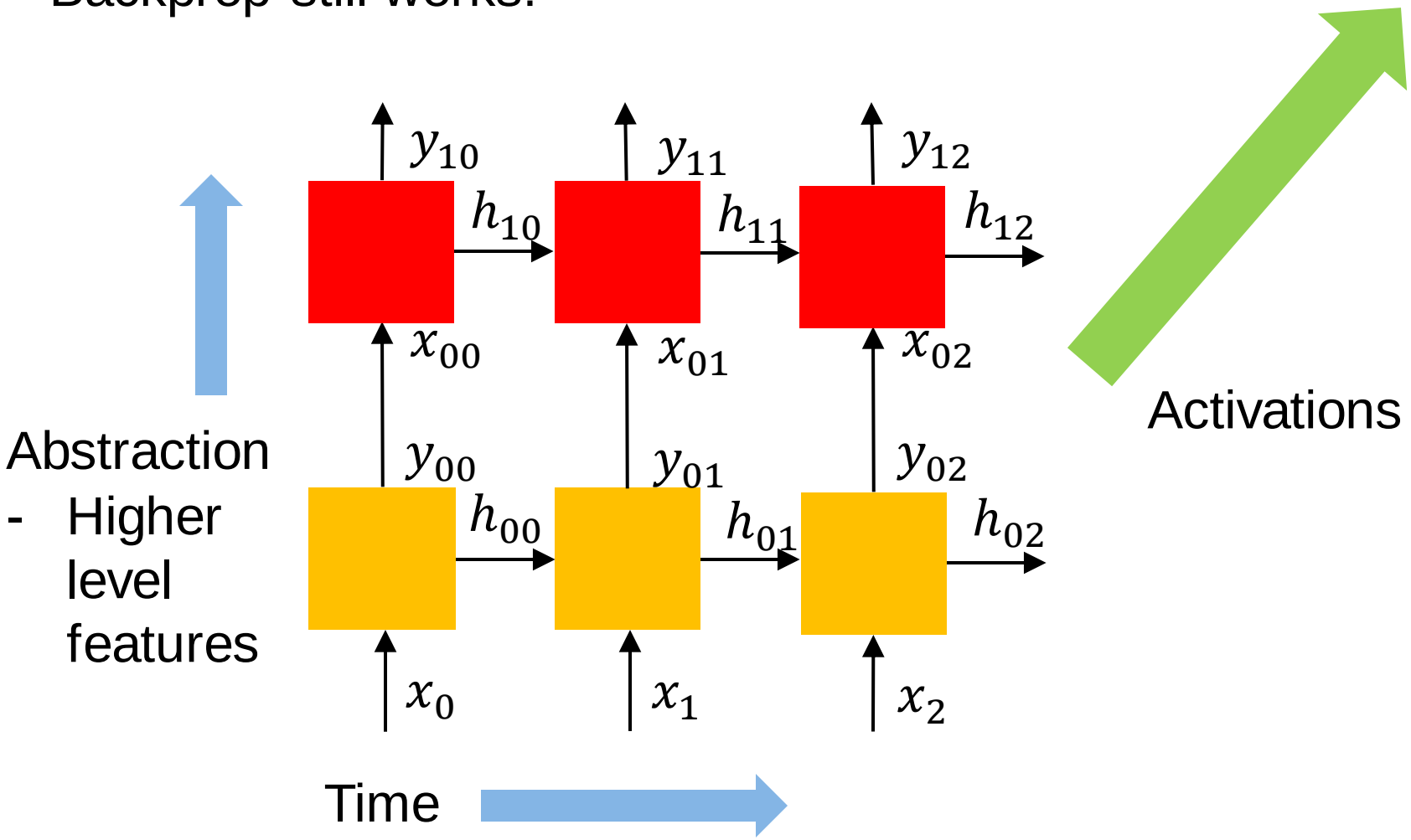
RNN structure

Backprop still works:



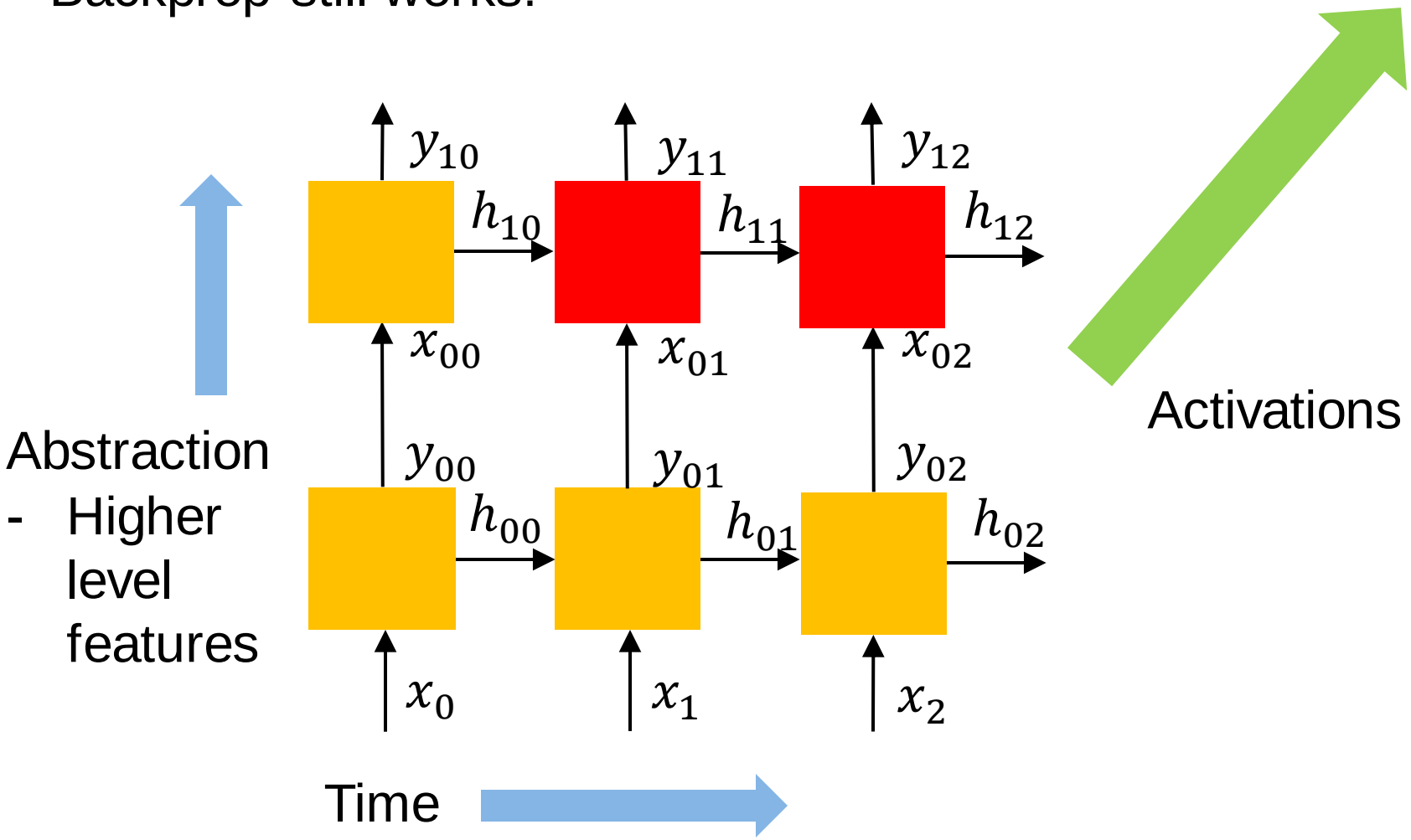
RNN structure

Backprop still works:



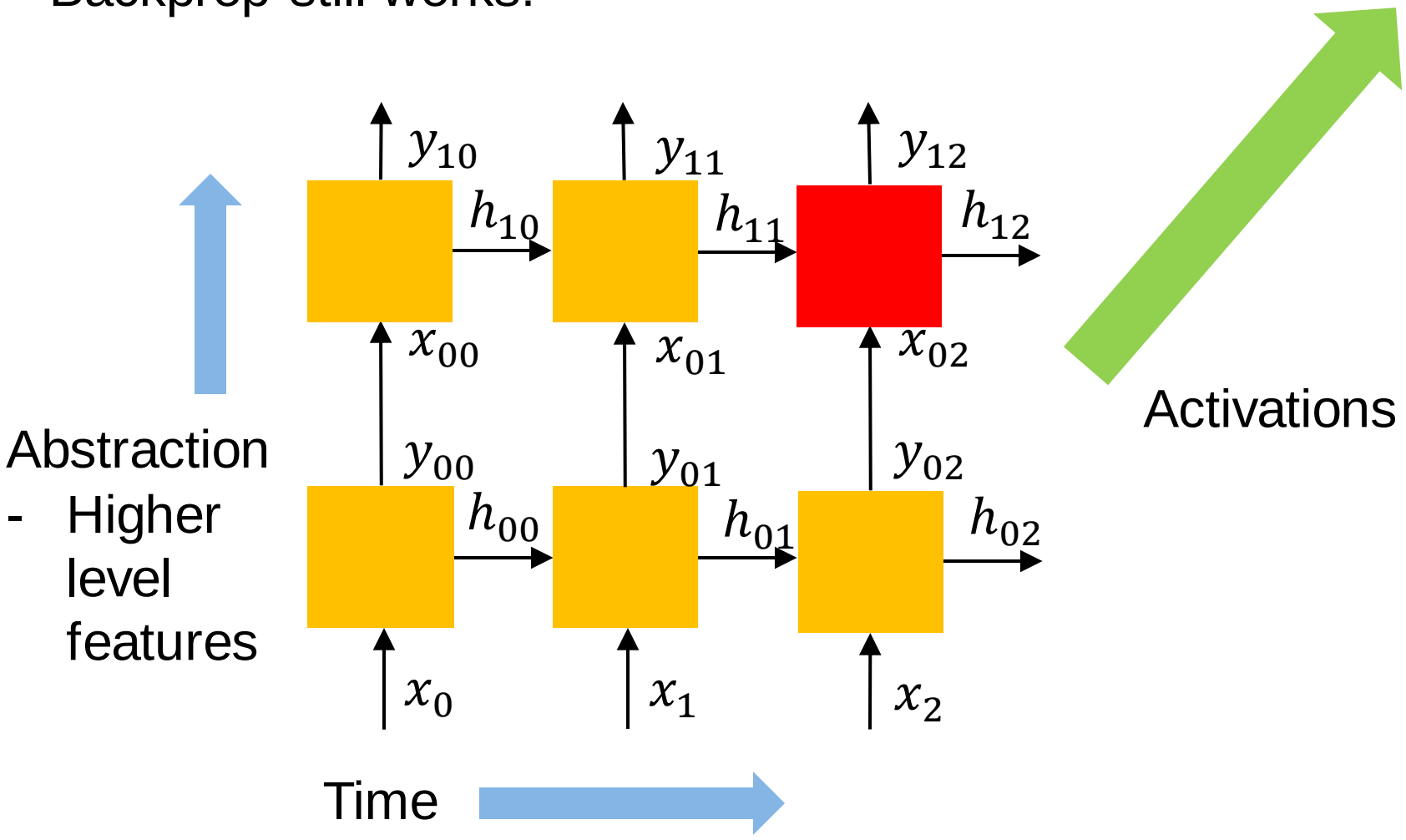
RNN structure

Backprop still works:



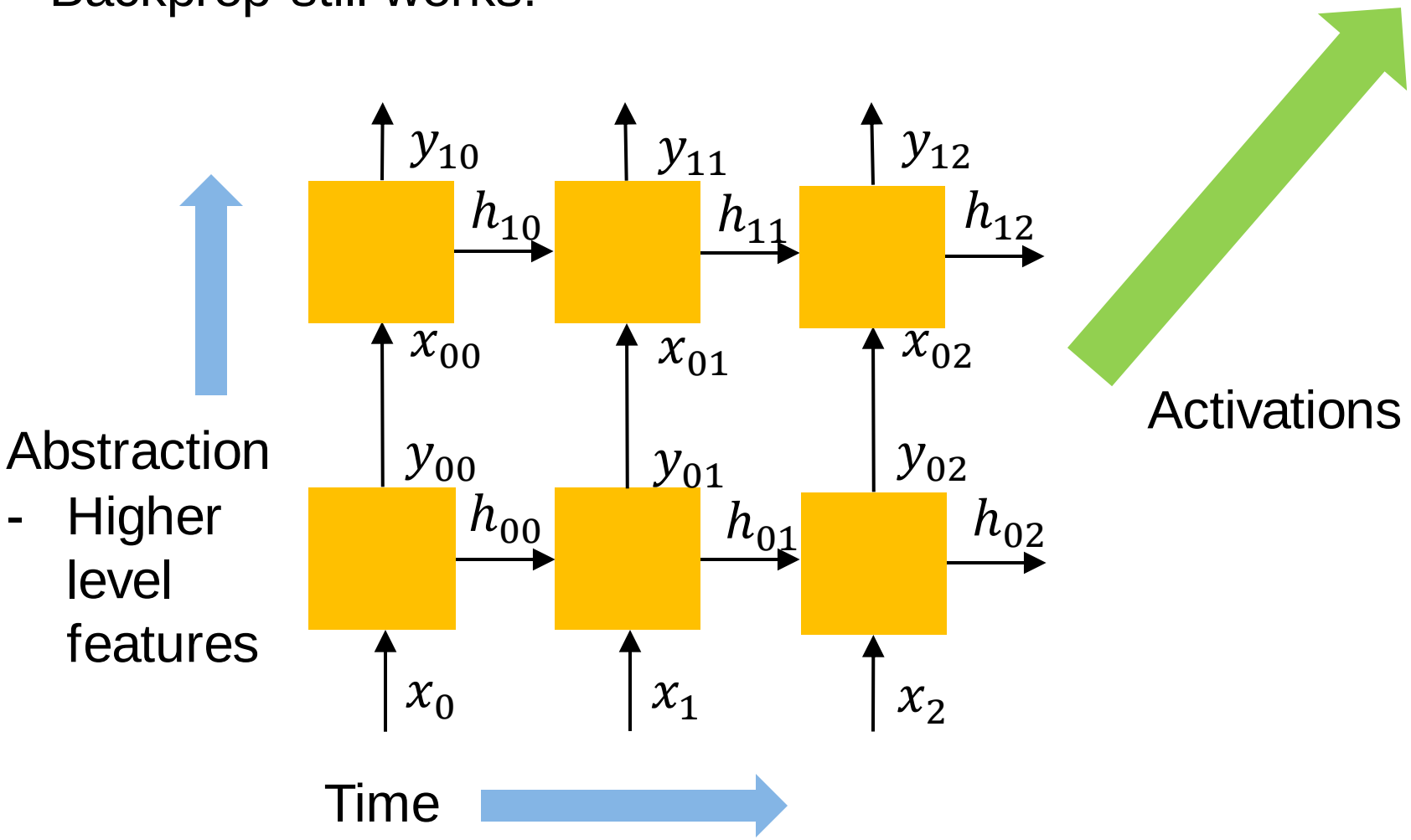
RNN structure

Backprop still works:



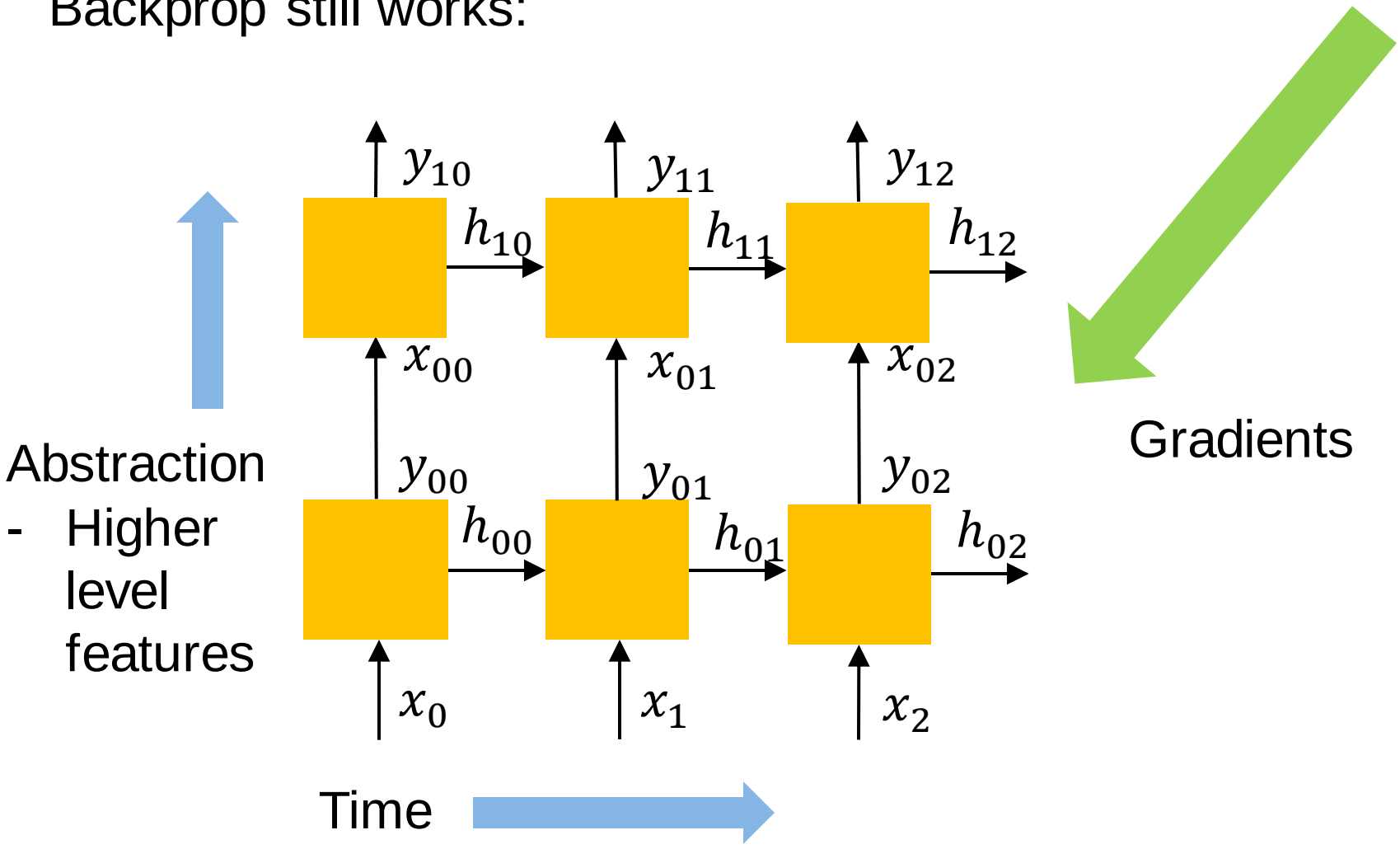
RNN structure

Backprop still works:



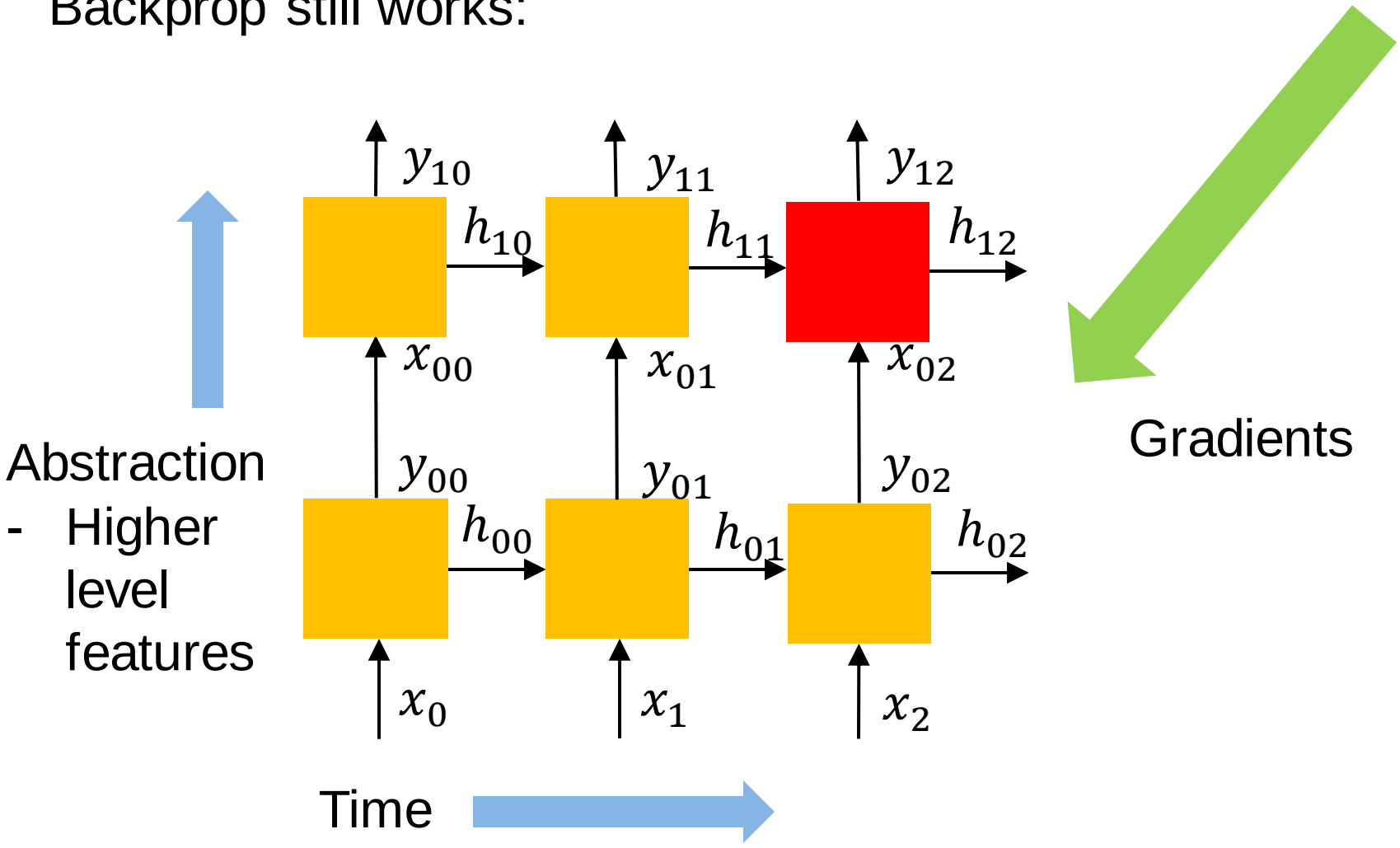
RNN structure

Backprop still works:



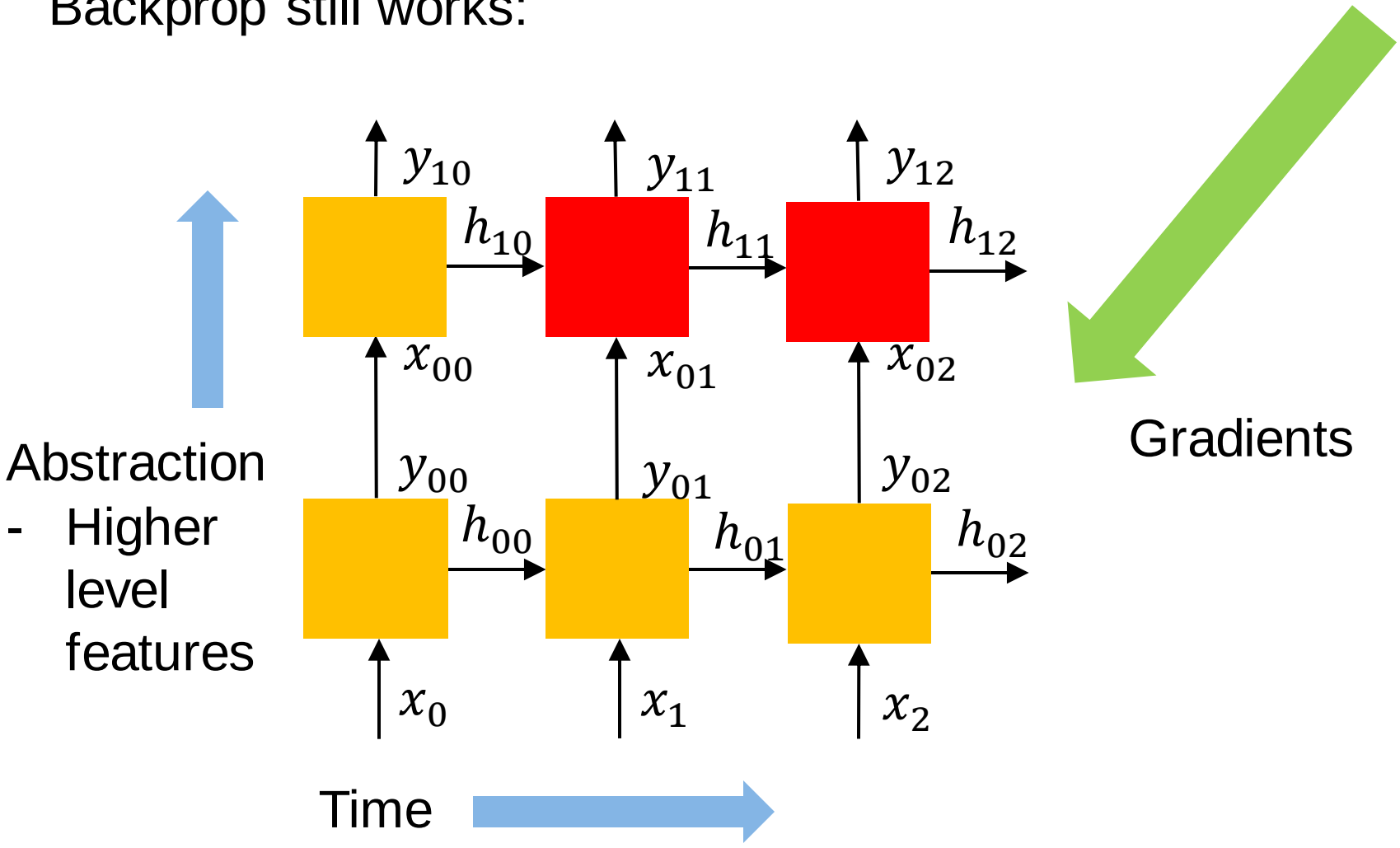
RNN structure

Backprop still works:



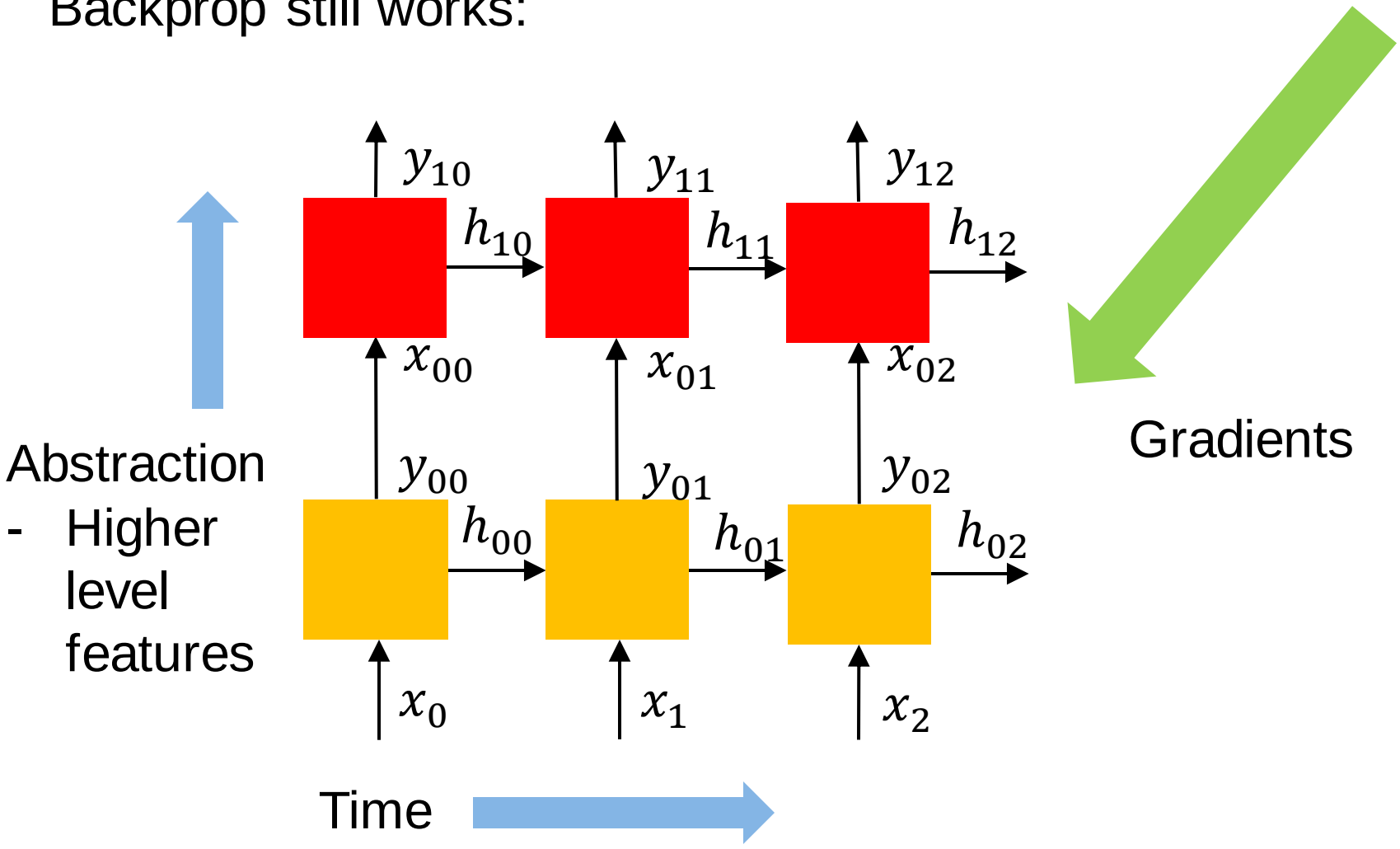
RNN structure

Backprop still works:



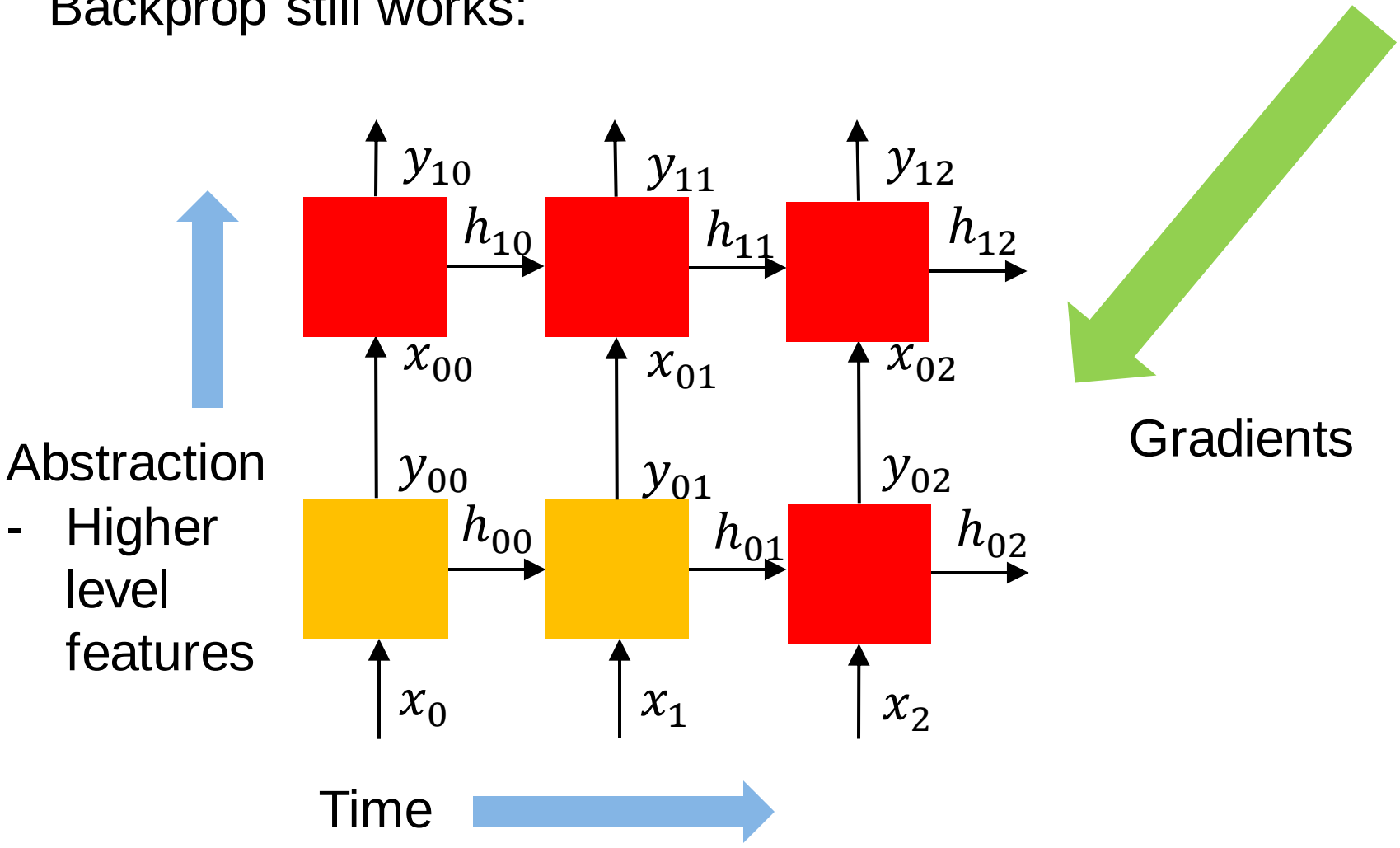
RNN structure

Backprop still works:



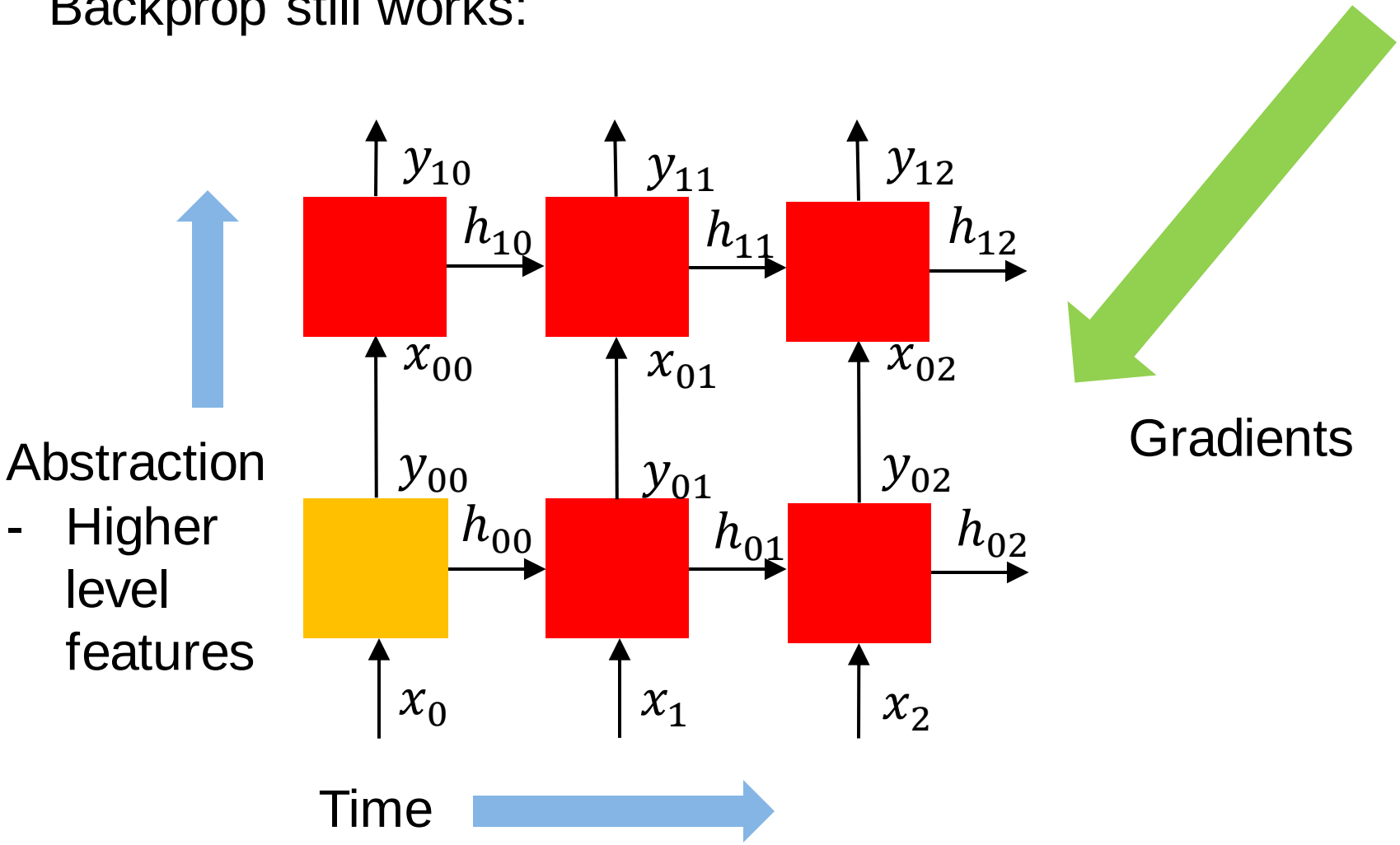
RNN structure

Backprop still works:



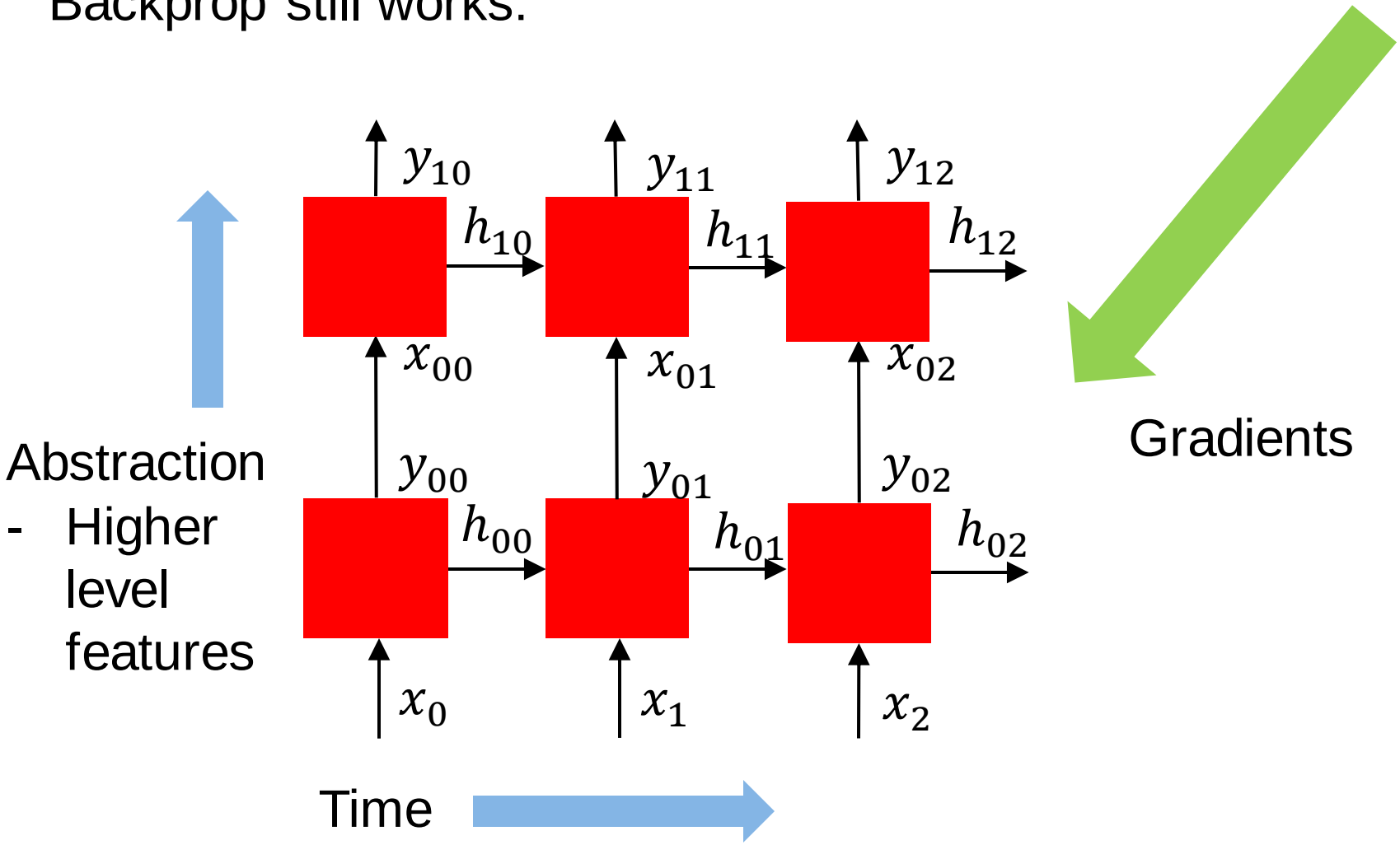
RNN structure

Backprop still works:



RNN structure

Backprop still works:



Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

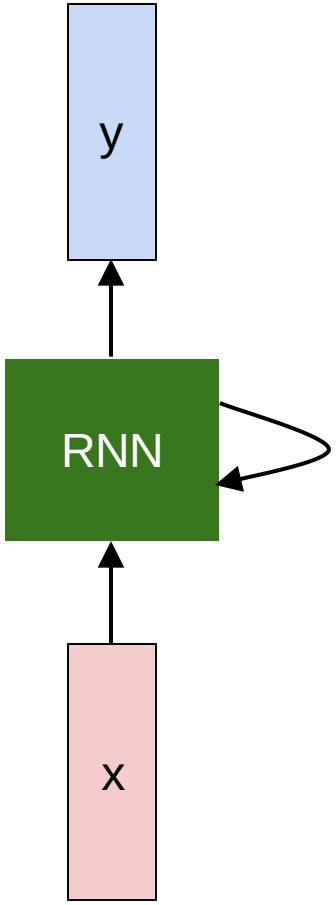
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

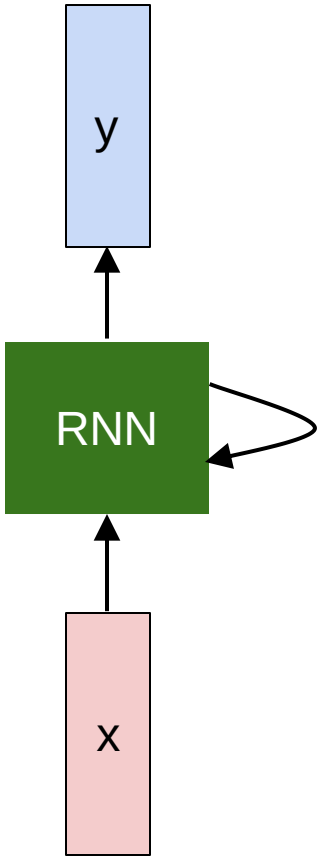
old state

input vector at some time step



Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:

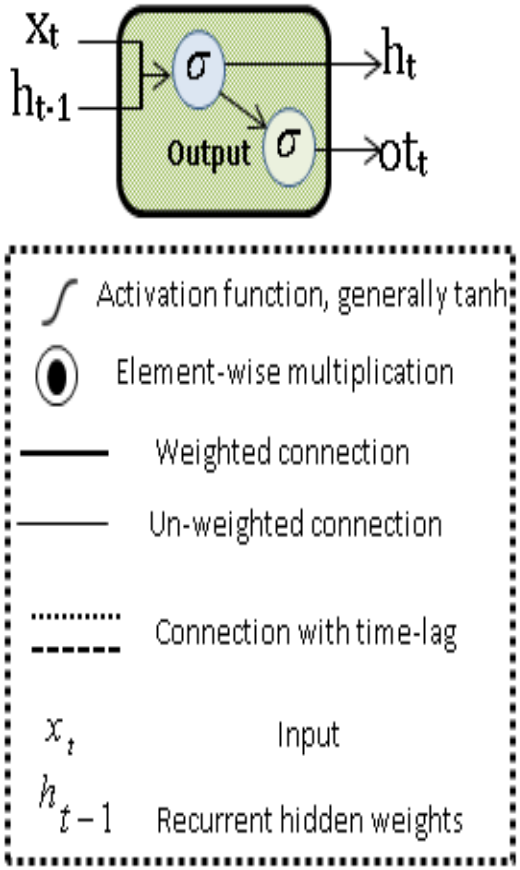


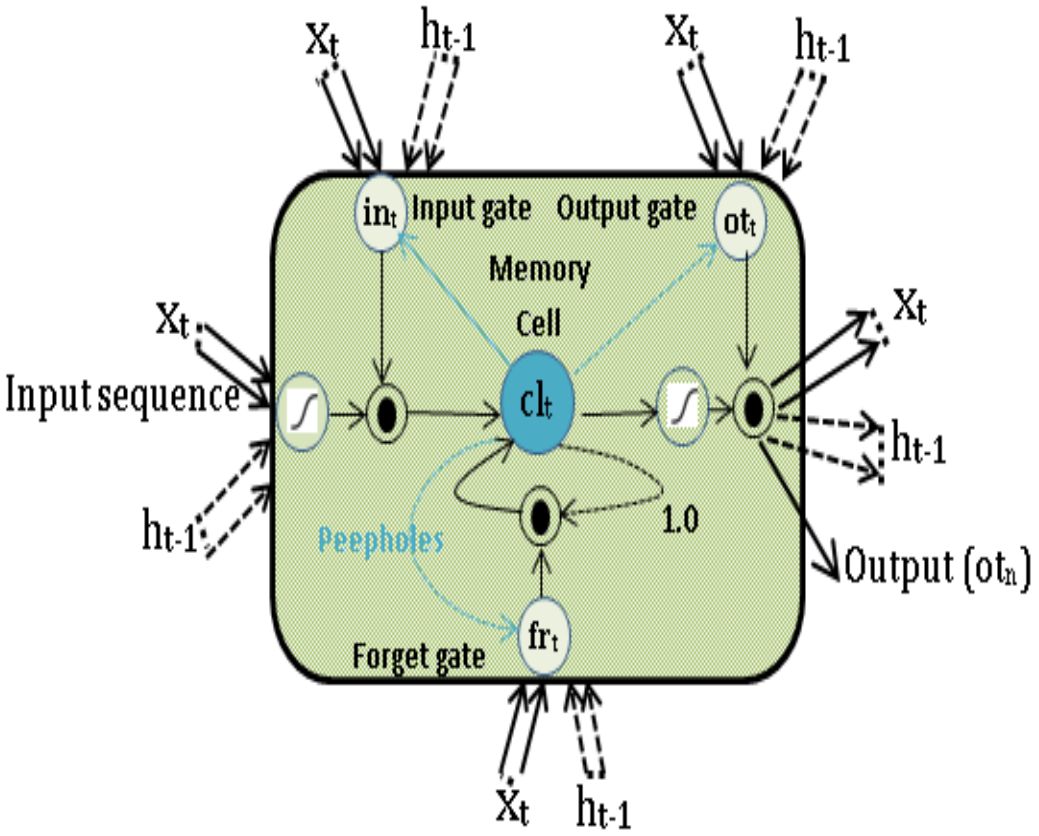
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Long short-term memory





Long short-term memory

$$x_t, h_{t-1}, cl_{t-1} \rightarrow h_t, cl_t$$

$$in_t = \sigma(w_{xin} x_t + w_{hin} h_{t-1} + w_{clin} cl_{t-1} + b_{in})$$

$$fr_t = \sigma(w_{xfr} x_t + w_{hifr} h_{t-1} + w_{clfr} cl_{t-1} + b_{fr})$$

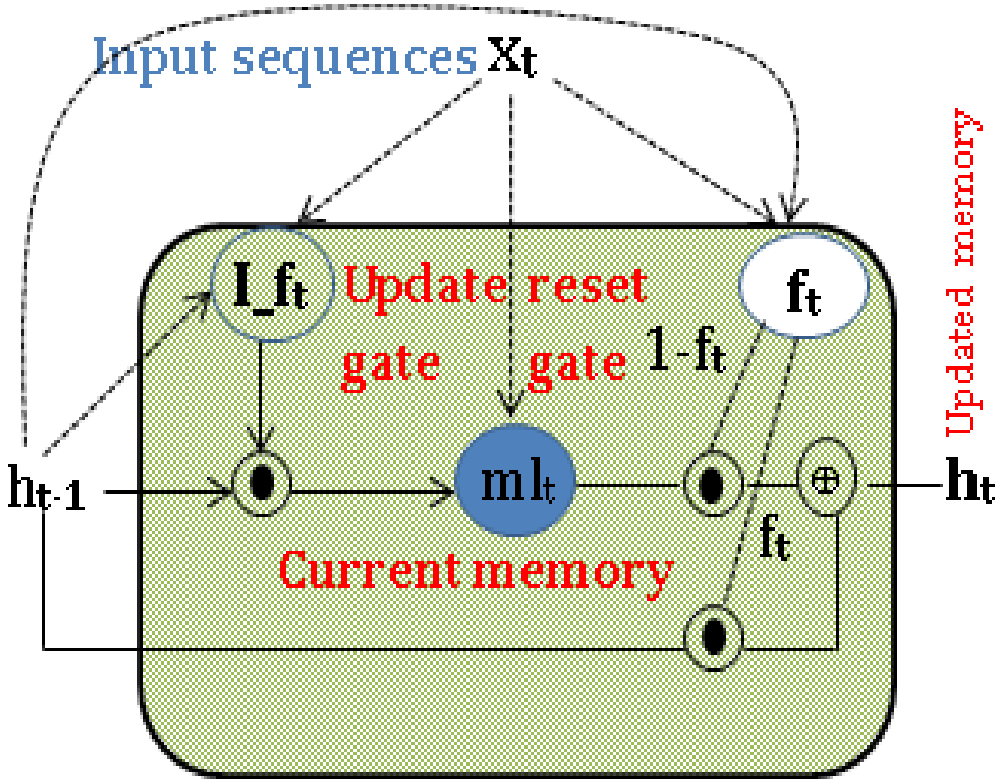
$$cl_t = fr_t \odot cl_{t-1} + in_t \odot \tanh(w_{xcl} x_t + w_{hcl} hi_{t-1} + b_{cl})$$

$$ot_t = \sigma(w_{xot} x_t + w_{hot} hi_{t-1} + w_{clot} cl_t + b_{ot})$$

$$h_t = ot_t \odot \tanh(cl_t)$$

Gated Recurrent Unit

Gated recurrent unit (GRU) is an alternative to LSTM networks.
Formulae shows, unlike LSTM memory cell with a list of gates (input, output and forget), GRU only consist of gates (update and forget) that are collectively involve in balancing the interior flow of information of the unit.



Gated Recurrent Unit

$$x_t, h_{t-1} \rightarrow h_t$$

$$in\_fr_t = \sigma(w_{xin\_fr} x_t + w_{hiin\_fr} h_{t-1} + b_{in\_fr}) \quad \text{(Update gate)}$$

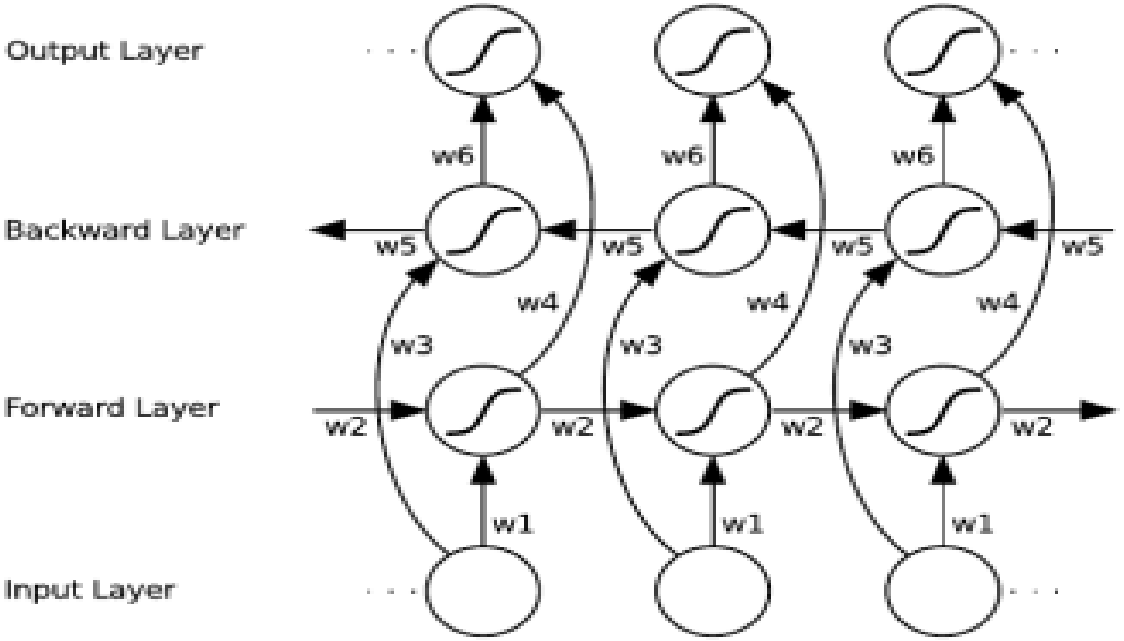
$$fr_t = \sigma(w_{xfr} x_t + w_{hifr} h_{t-1} + b_{fr}) \quad \text{(Forget or reset gate)}$$

$$cl_t = \tanh(w_{xcl} x_t + w_{hcl}(fr \odot hi_{t-1}) + b_{cl}) \quad \text{(Current memory)}$$

$$h_t = f \odot h_{t-1} + (1-f) \odot cl \quad \text{(Updated memory)}$$

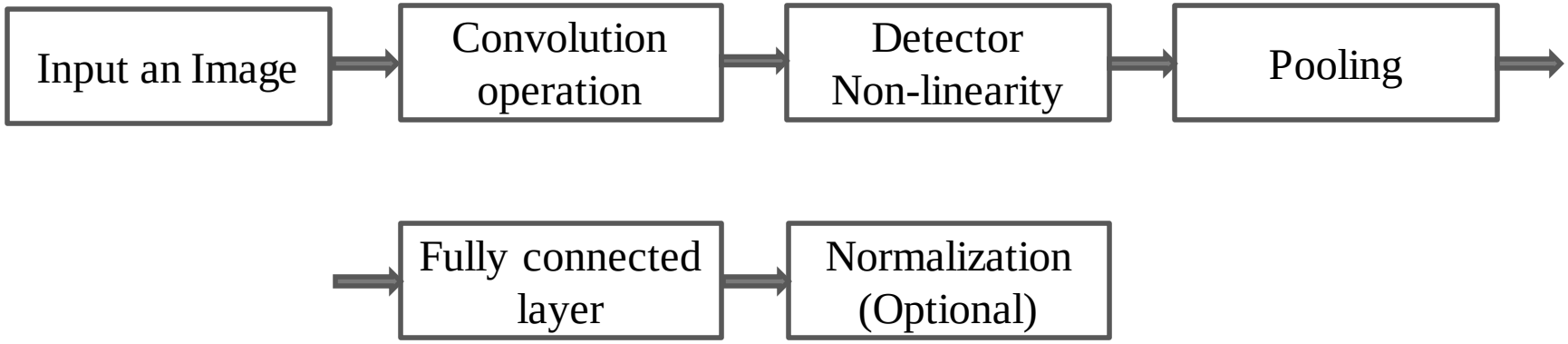
Extensions to LSTM architecture: Bidirectional RNN, LSTM, GRU

- Only the past information is taken into account in the training of a unidirectional RNN/LSTM
- Bidirectional architecture enables the use of future information
- Implementation with separate Forward-pass and Backward-pass specific layer weights
- Final output computed as the sum of forward and backward layer outputs



Convolutional Neural Network

- Neural Network with a convolution operation instead of matrix multiplication in at least one of the layers



Convolutional Neural Network

Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

Filter (Kernel)

0.2	0.7
-0.5	0.7

Convolutional Neural Network

- Input, e.g. an image

1	3	5	2	4
6	0	2	1	3
6	3	1	3	6
7	3	2	1	3
5	3	0	0	2

$$c_1 = \mathrm{f}\,(0.2 * 1 + 0.7 * 3 - 0.5 * 6 + 0.7 * 0) = \mathrm{f}(-0.7)$$

$$c_2 = \mathrm{f}(0.2 * 3 + 0.7 * 5 - 0.5 * 0 + 0.7 * 2) = \mathrm{f}(5.5)$$

Note: Bias terms omitted!

f represents some non-linear activation function

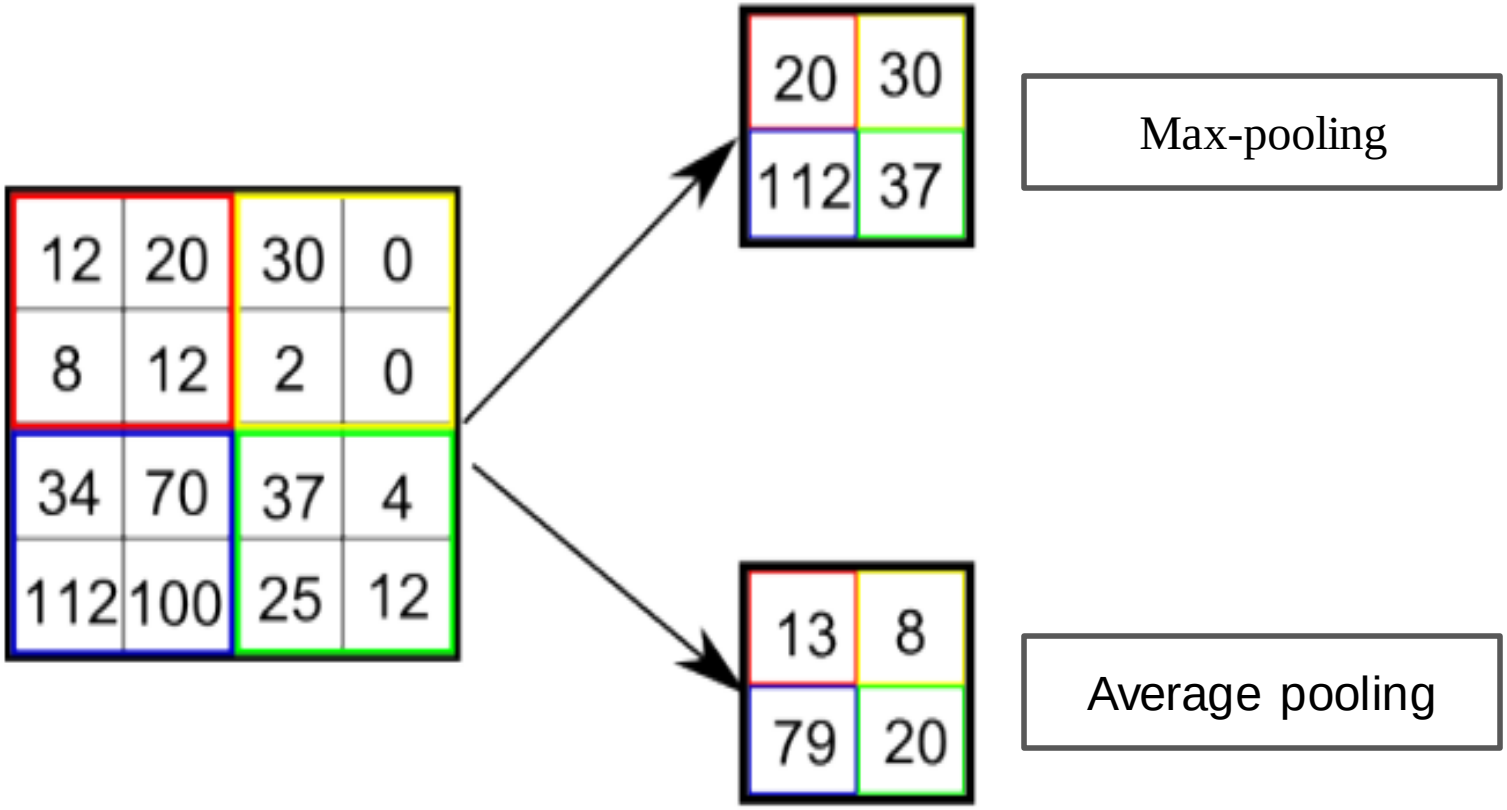
Filter (Kernel)

0.2	0.7
-0.5	0.7

Feature map

f(-0.7) f(5.5) …

Convolutional Neural Network



Software Installation

- sudo apt-get install libatlas-base-dev gfortran python-dev
- sudo apt-get install python-pip
- sudo pip install --upgrade pip
- sudo pip install numpy
- sudo pip install scipy
- sudo pip install matplotlib
- Sudo pip install seaborn
- sudo pip install scikit-learn
- sudo pip install tensorflow
- sudo pip install theano
- sudo pip install keras
- sudo pip install pandas
- sudo pip install h5py
- sudo pip install jupyter
- sudo pip install ipython

Artificial Intelligence (AI) toolkits

Scikit-learn - Python library that implements a comprehensive range of machine learning algorithms.

- easy-to-use, general-purpose toolbox for machine learning in Python.
- supervised and unsupervised machine learning techniques.
- Utilities for common tasks such as model selection, feature extraction, and feature selection.
- Built on NumPy, SciPy, and matplotlib.
- Open source, commercially usable - BSD license.

Artificial Intelligence (AI) toolkits

TensorFlow - library for numerical computation using data flow graphs / deep learning.

- Open source
- By Google
- used for both research and production
- Used widely for deep learning/neural nets
- But not restricted to just deep models
- Multiple GPU Support

Artificial Intelligence (AI) toolkits

Keras – It is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Runs seamlessly on CPU and GPU.

Supporting Libraries

NumPy

Base N-dimensional array package

SciPy library

Fundamental library for scientific computing

Matplotlib

Comprehensive 2D Plotting

IP[y]: IPython

IPython

Enhanced Interactive Console

Sympy

Symbolic mathematics

pandas

Data structures & analysis