# Deep Learning for Cyber Security use cases

**Vinayakumar R**

PhD Student,

Centre for Computational Engineering and Networking,
Amrita Vishwa Vidyapeetham,

Coimbatore

https://vinayakumarr.github.io/

# Agenda

AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

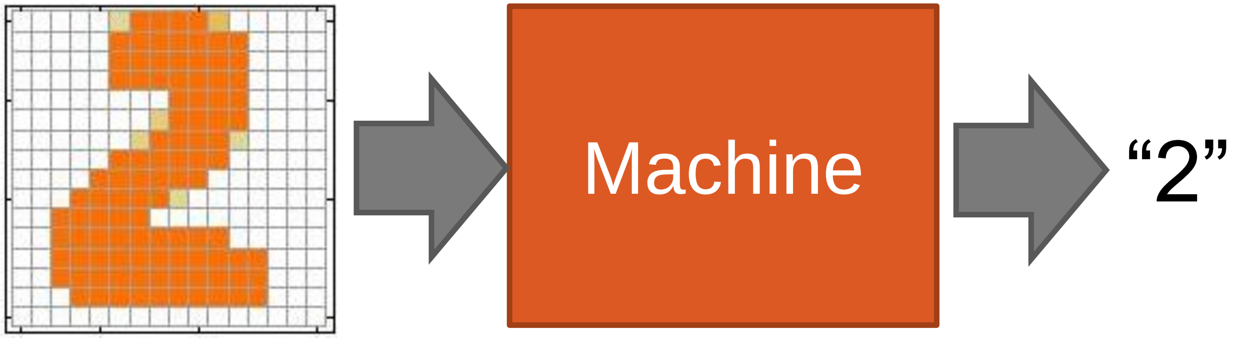
- Why deep learning?
- Deep Neural Networks
- Recurrent structures – RNN, LSTM, GRU
- Bidirectional recurrent structures
- Cyber Security use case: Intrusion Detection
- Hands on tutorial on python – numpy, scipy, matplotlib, pandas
- Hands on tutorial on TensorFlow and Keras

# 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

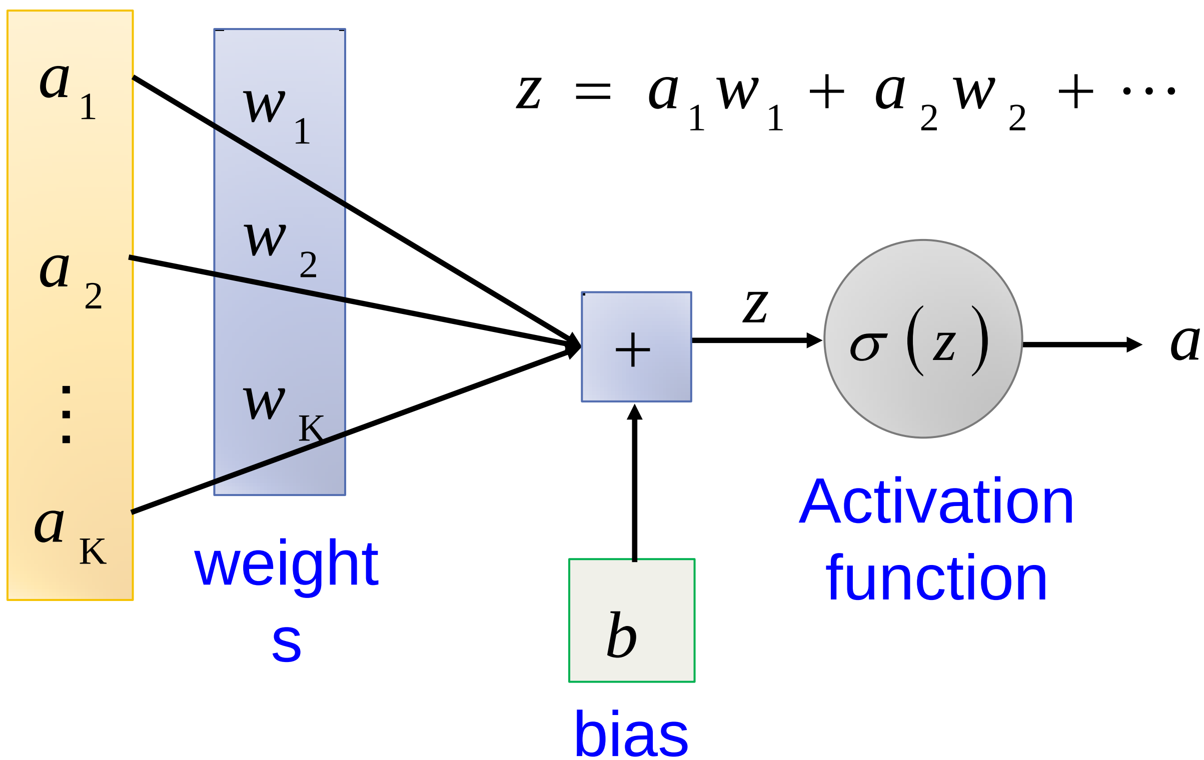
- Handwriting Digit Recognition



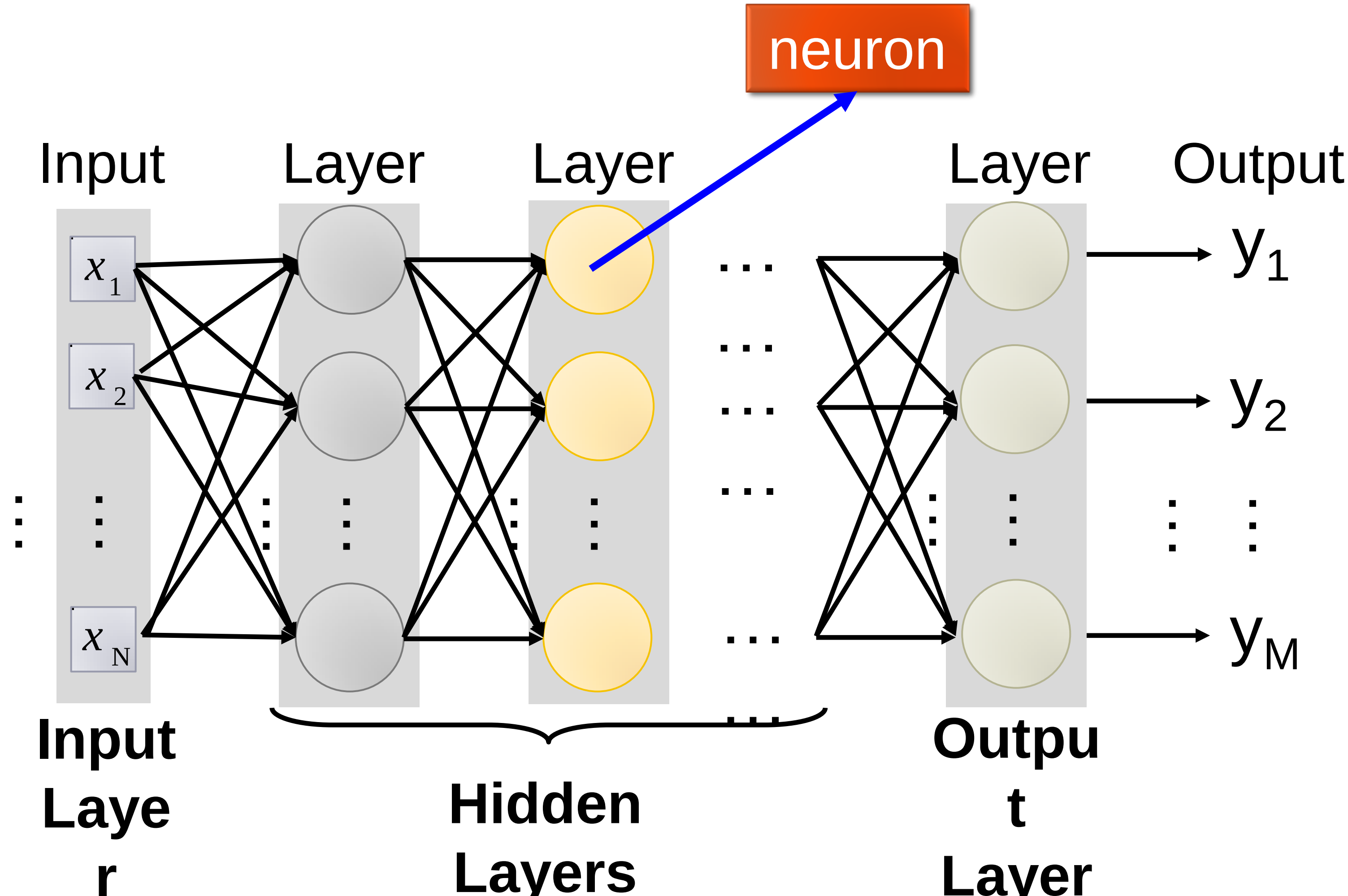
# Element of Neural Network

**Neuron** $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

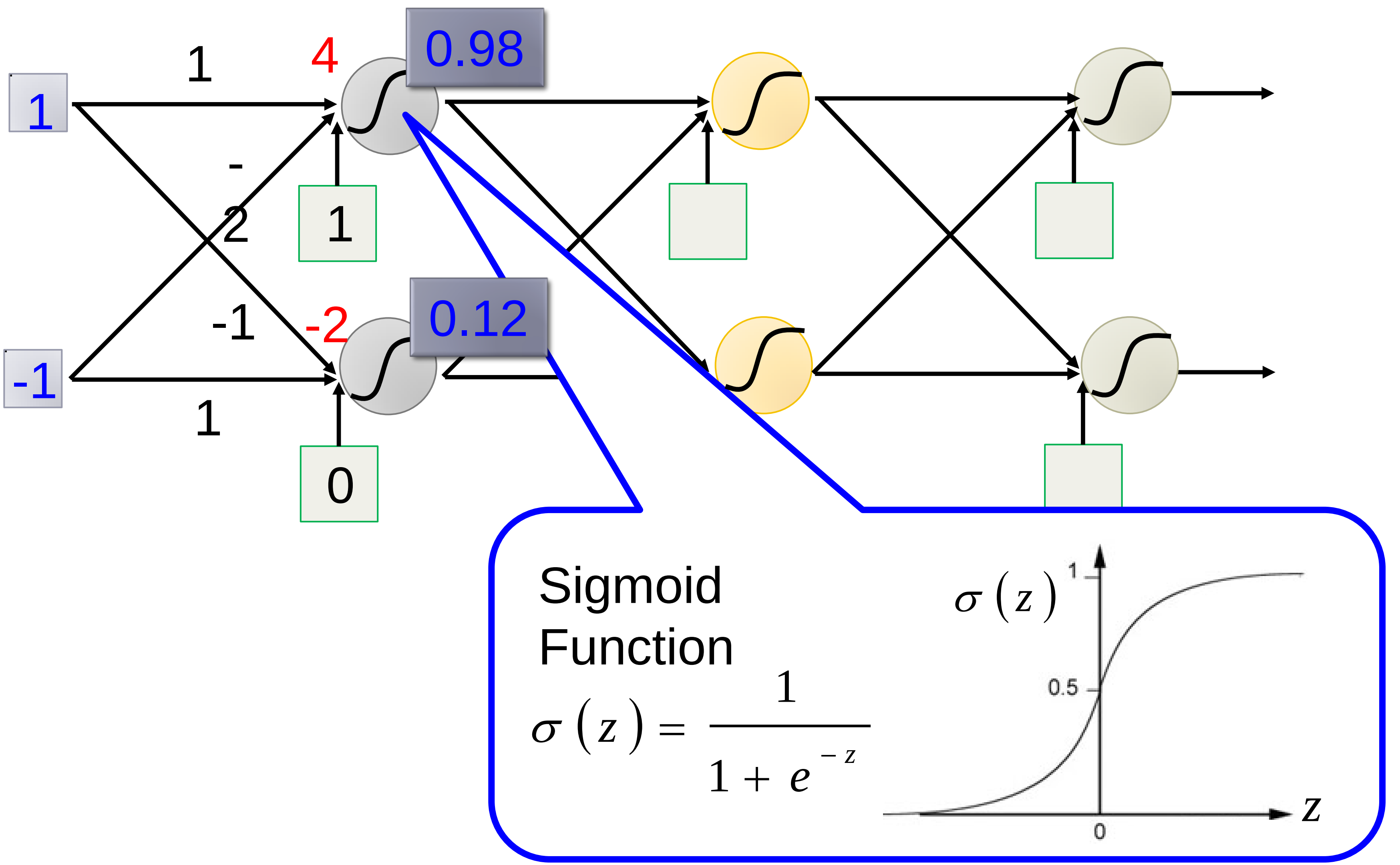


# Neural Network

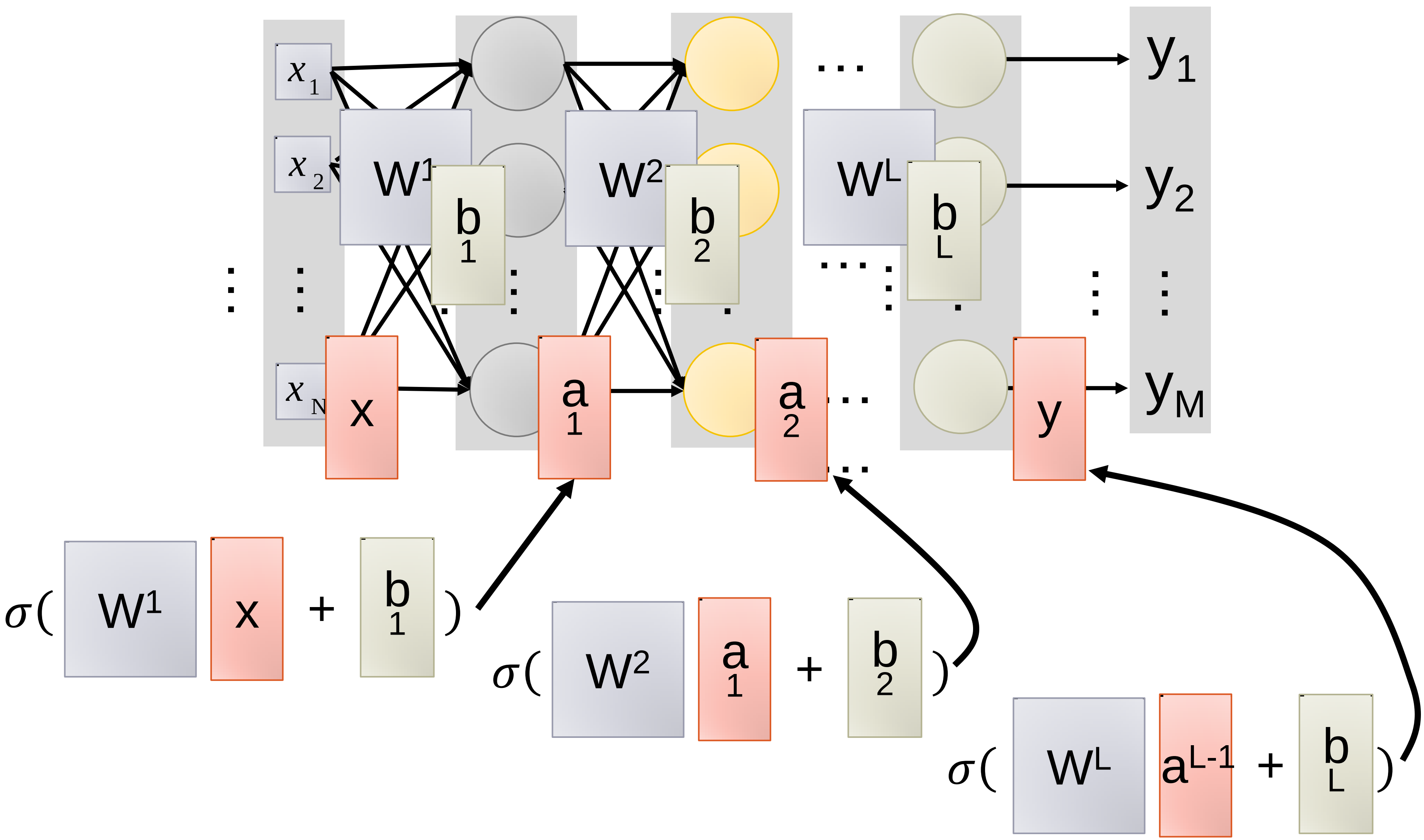



Deep means many hidden layers

# Neural Network



# Neural Network

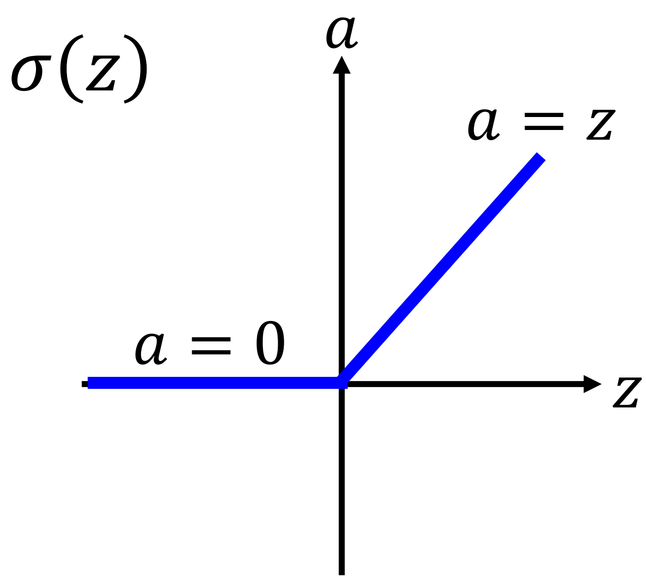


# Training DNN

# New Activation Function

# ReLU

- Rectified Linear Unit (ReLU)




***Reason:***

1. Fast to compute

2. Vanishing gradient problem

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

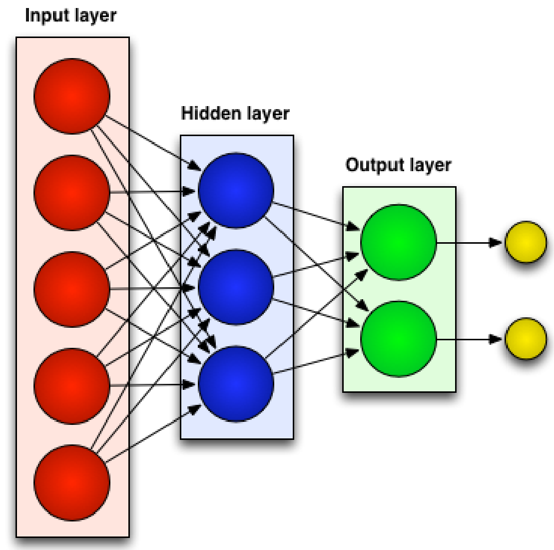
$$\sigma'(z) = \sigma(z) * (1 - \sigma(z))$$

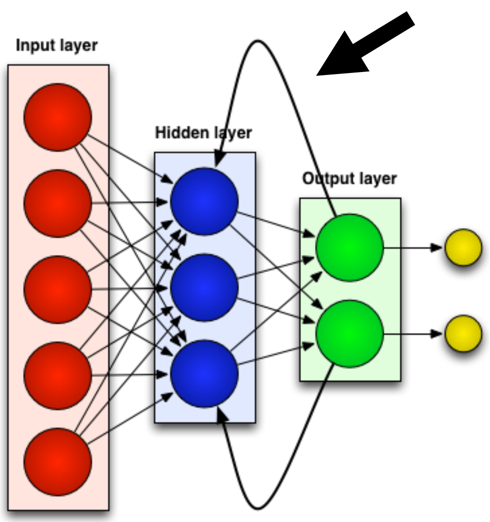
# Neural Networks

AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

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

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

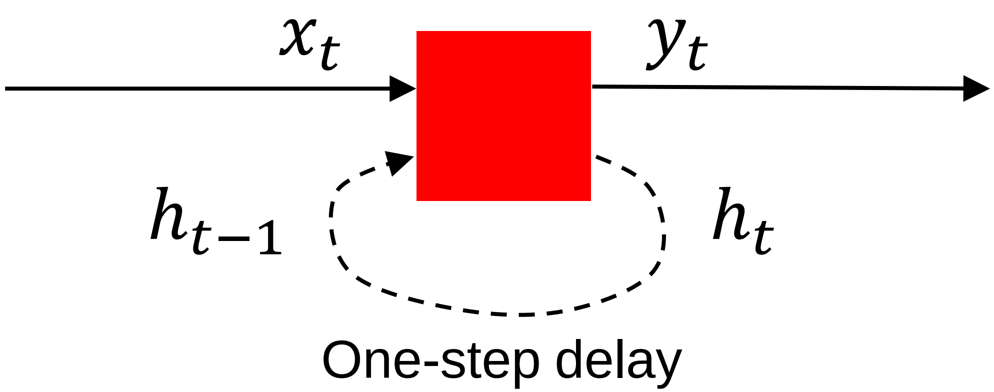



  - ➢ **Recurrent Neural Network:**
    - ✓ **connections between units form cyclic paths**

# Recurrent Neural Networks

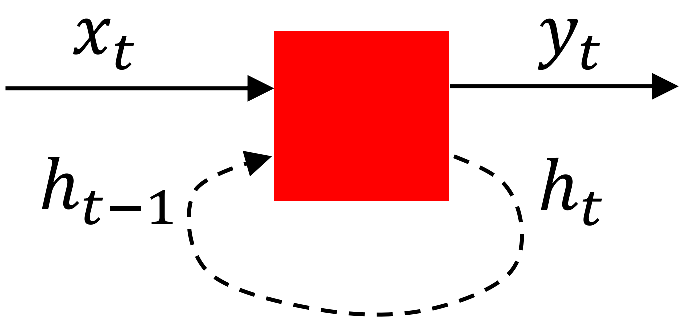
Recurrent networks introduce cycles and a notion of time.

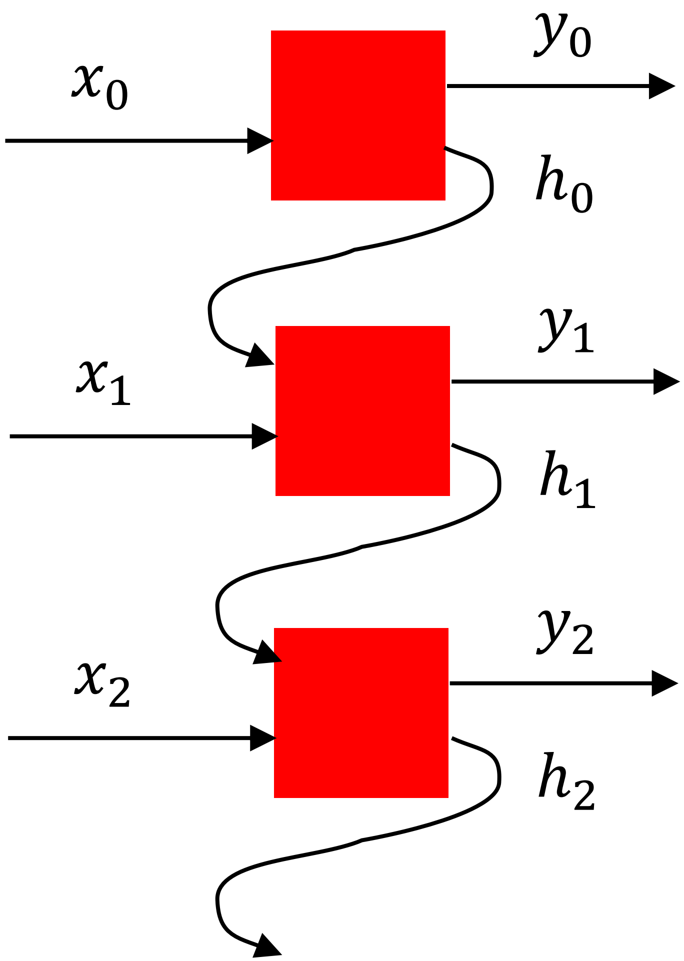



- They are designed to process sequences of data $x_1, \dots, x_n$ and can produce sequences of outputs $y_1, \dots, y_m$.

# Unrolling RNNs

RNNs can be unrolled across multiple time steps.



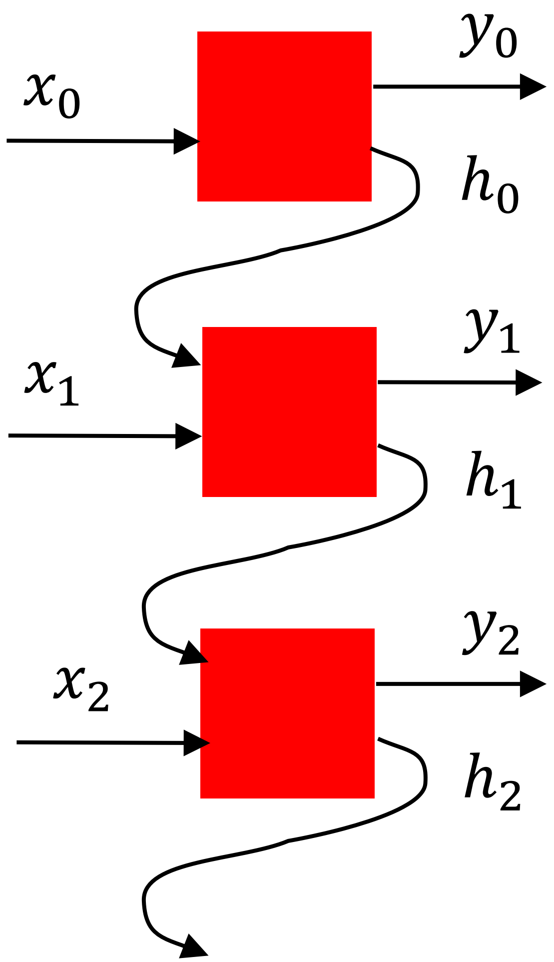



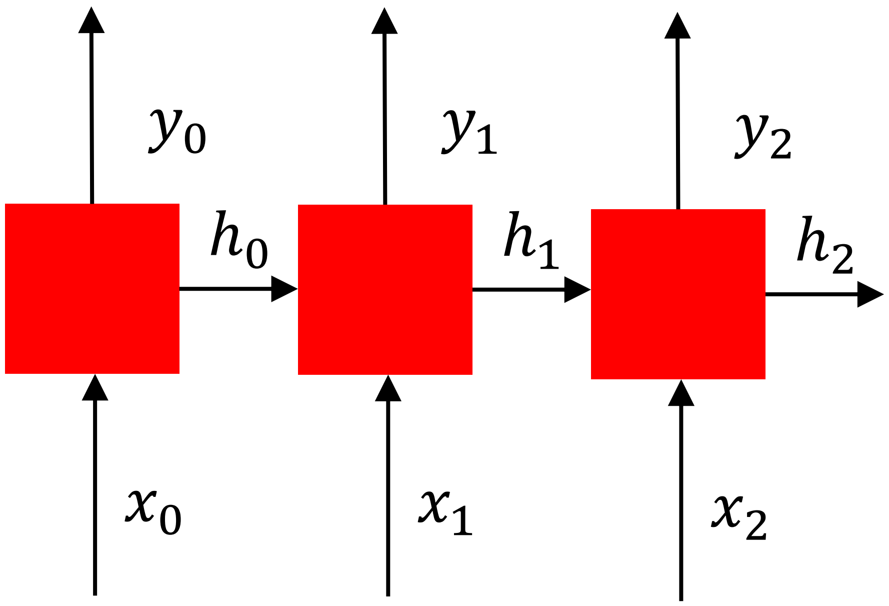

This produces a DAG which supports backpropagation.

But its size depends on the input sequence length.

# Unrolling RNNs

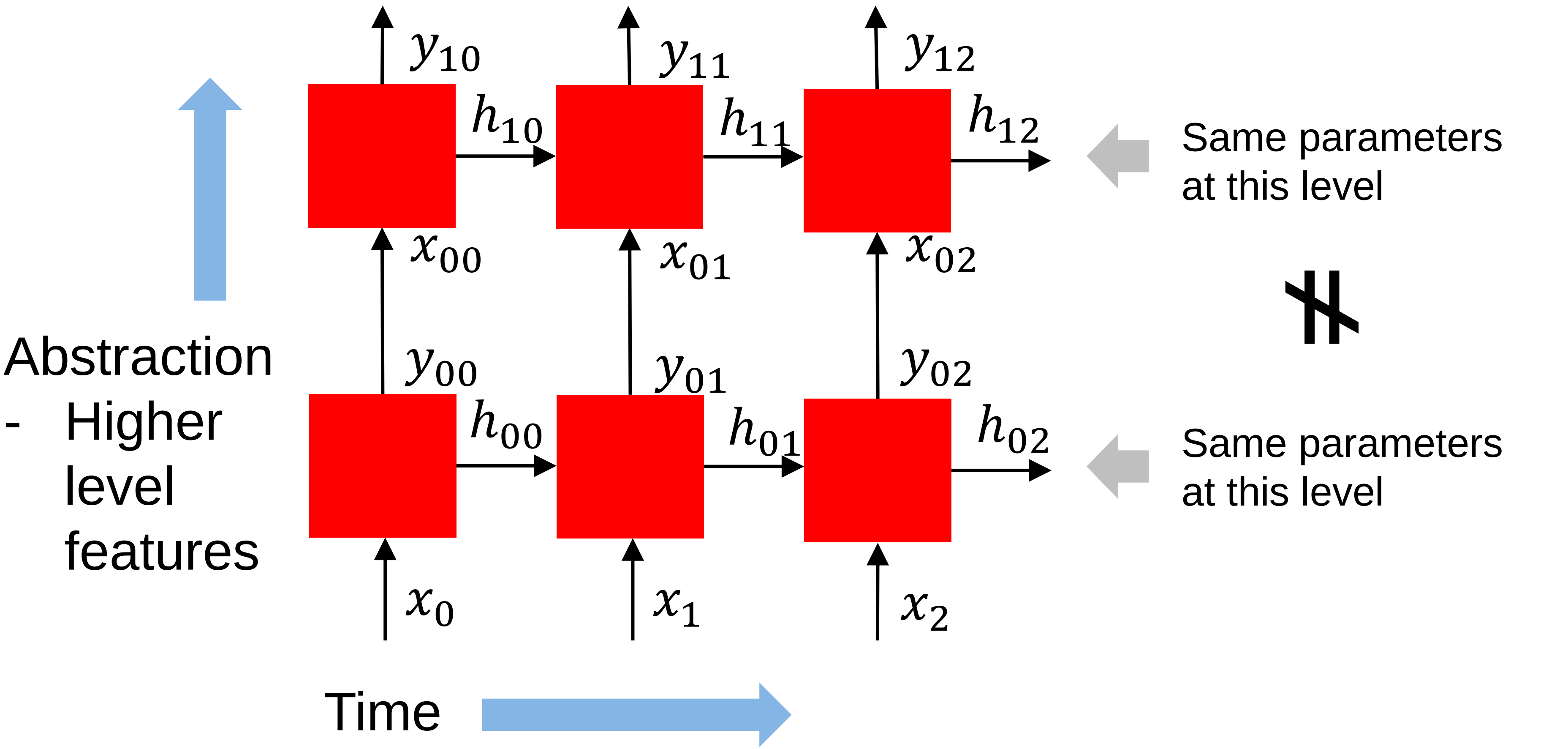
Usually drawn as:



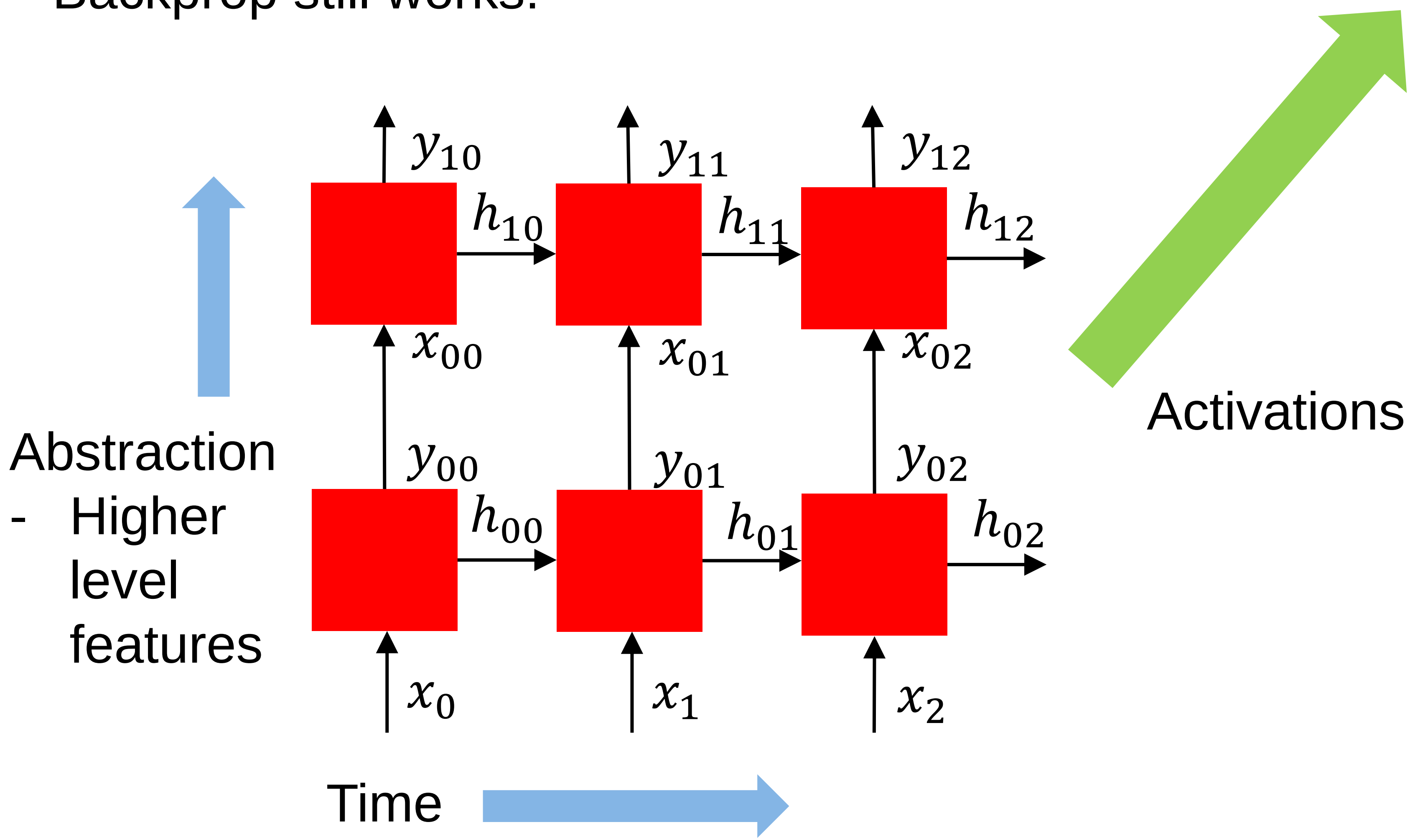
# RNN structure

Often layers are stacked vertically (deep RNNs):



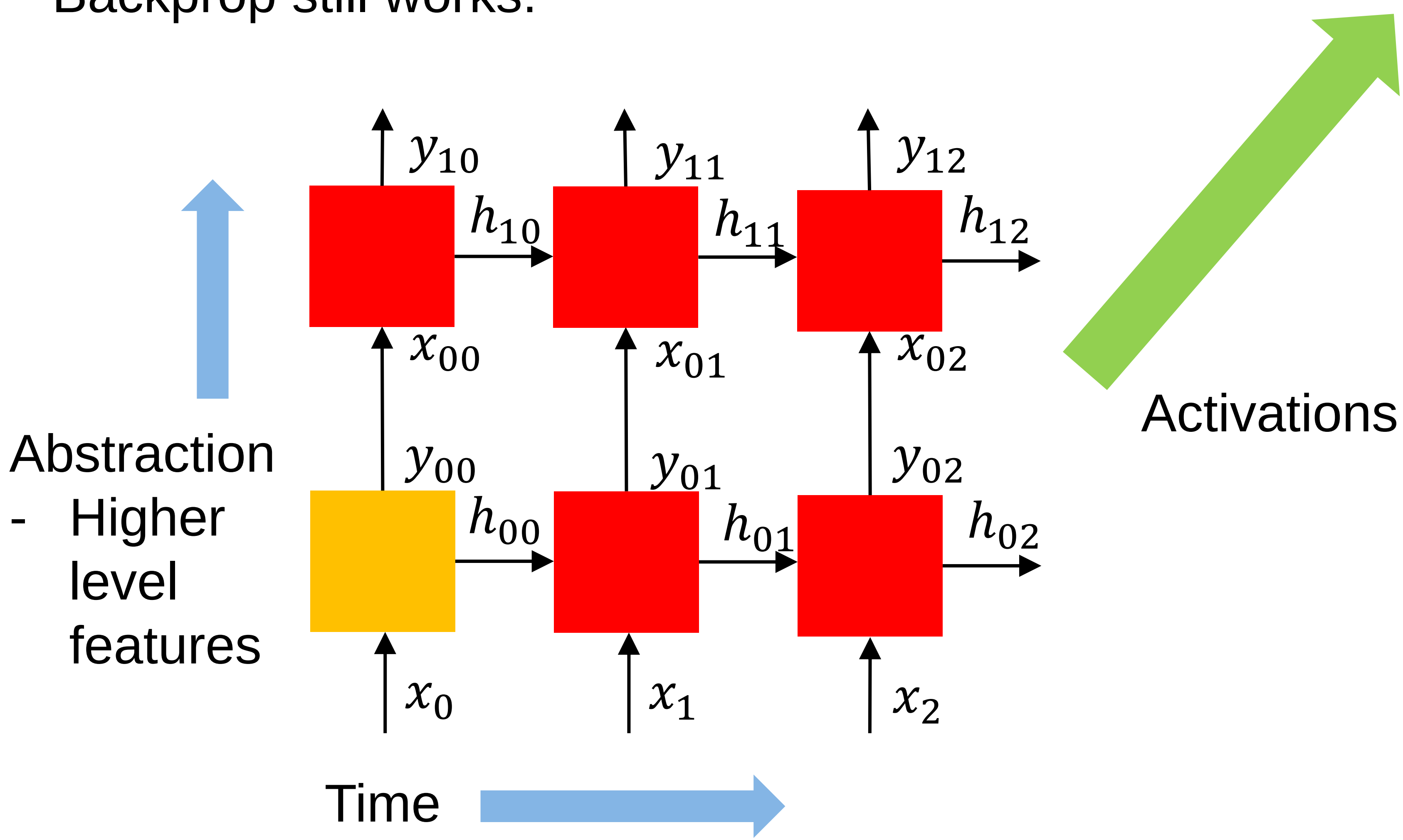
# RNN structure

Backprop still works:



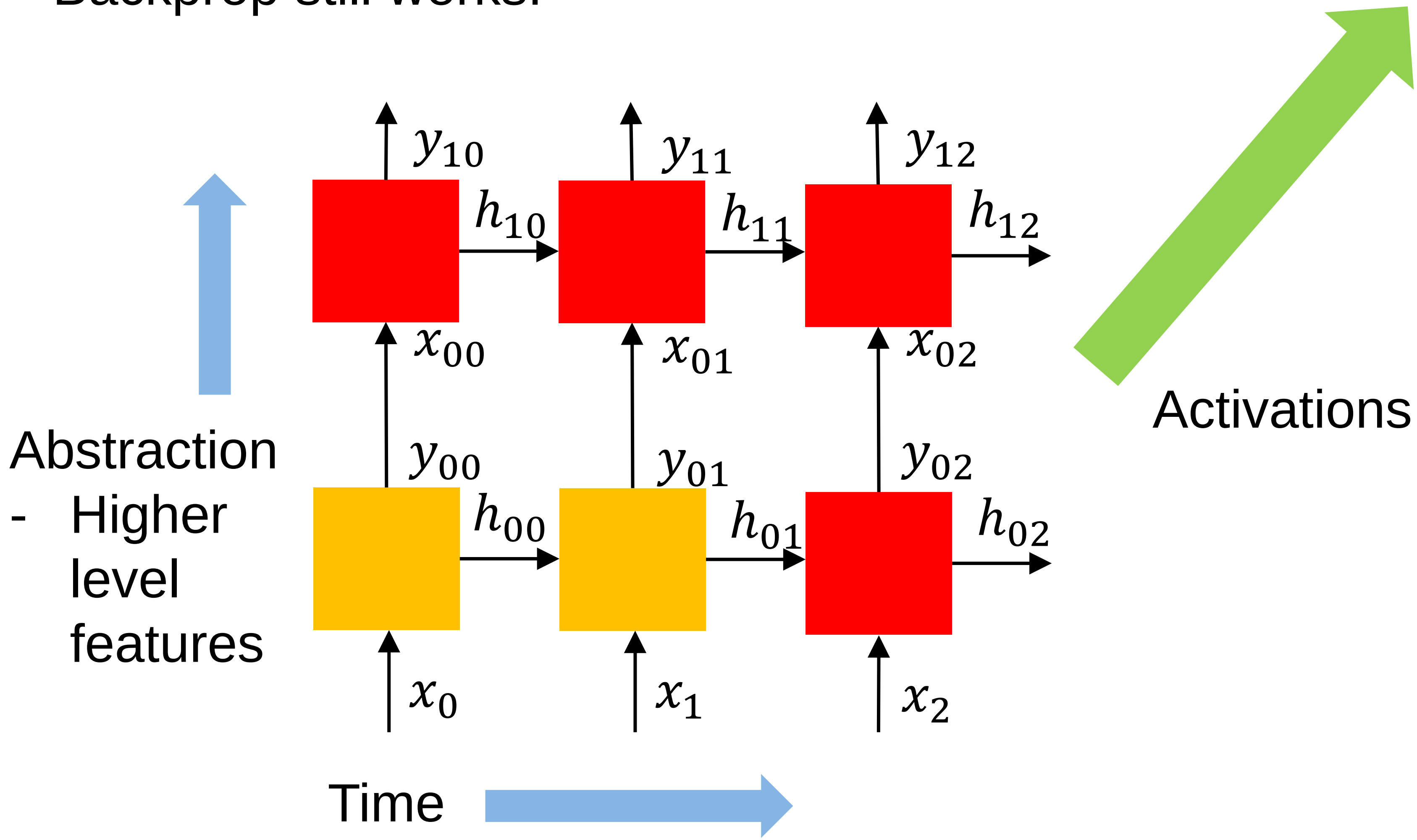
# RNN structure

Backprop still works:



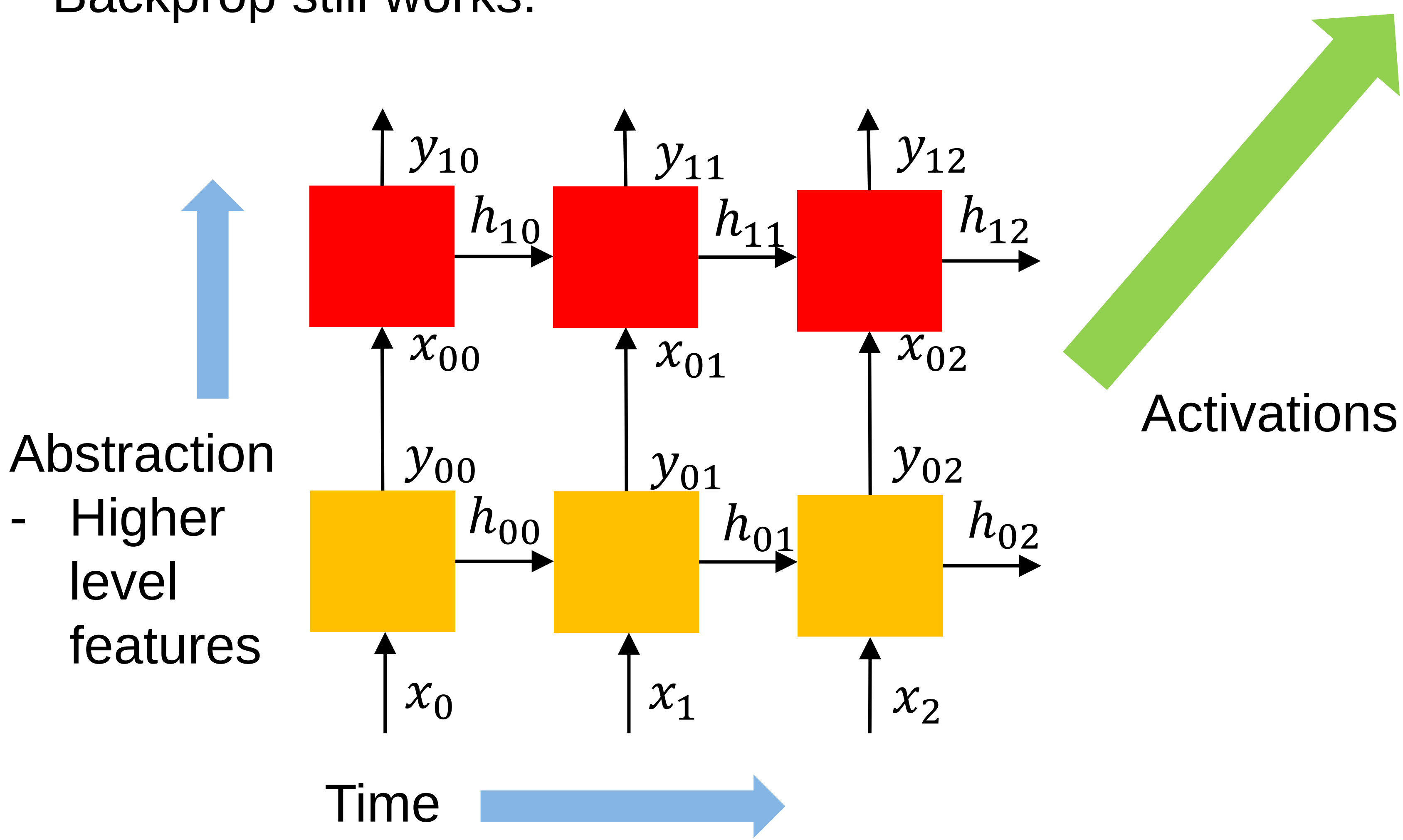
# RNN structure

Backprop still works:



# RNN structure

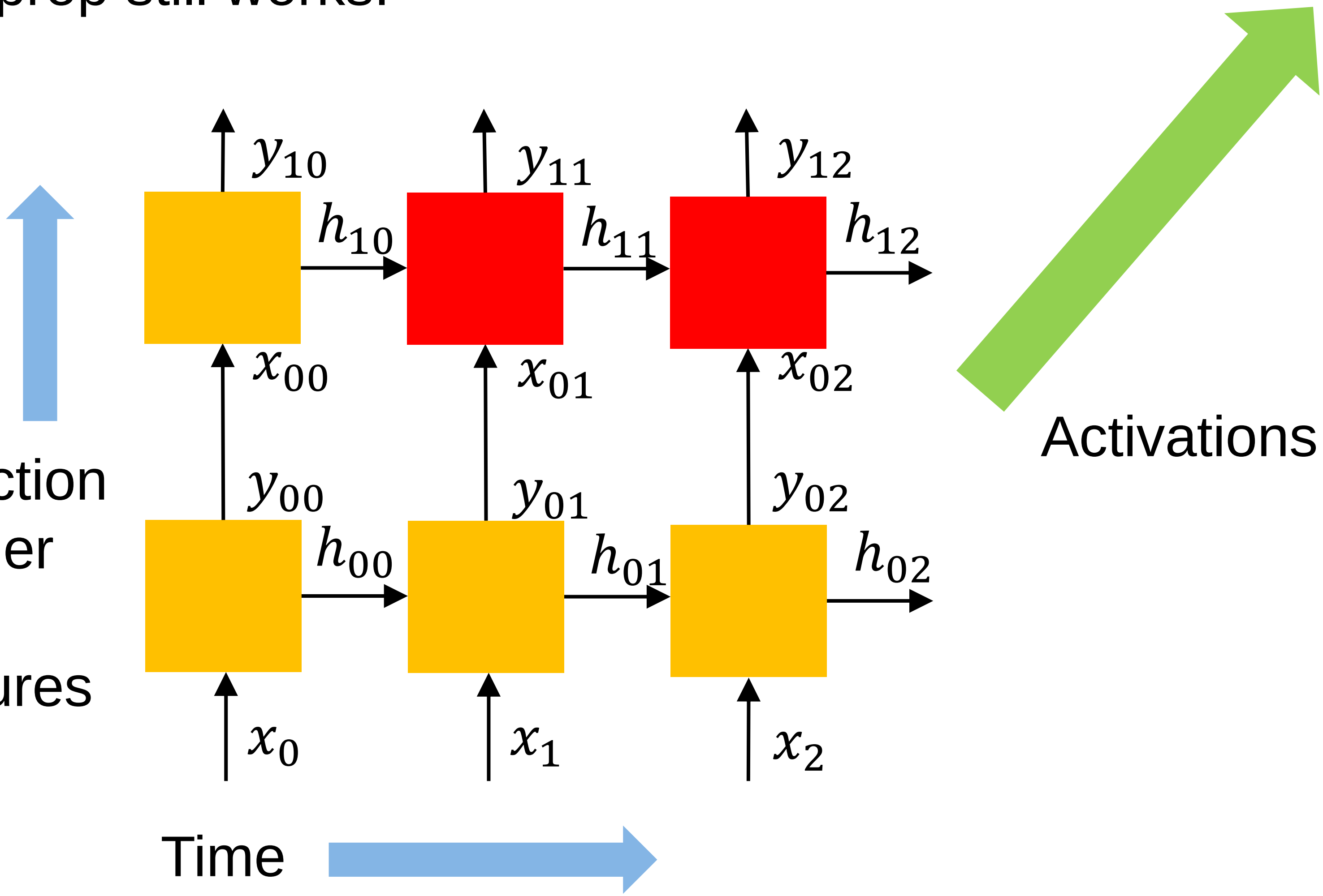
Backprop still works:



# RNN structure

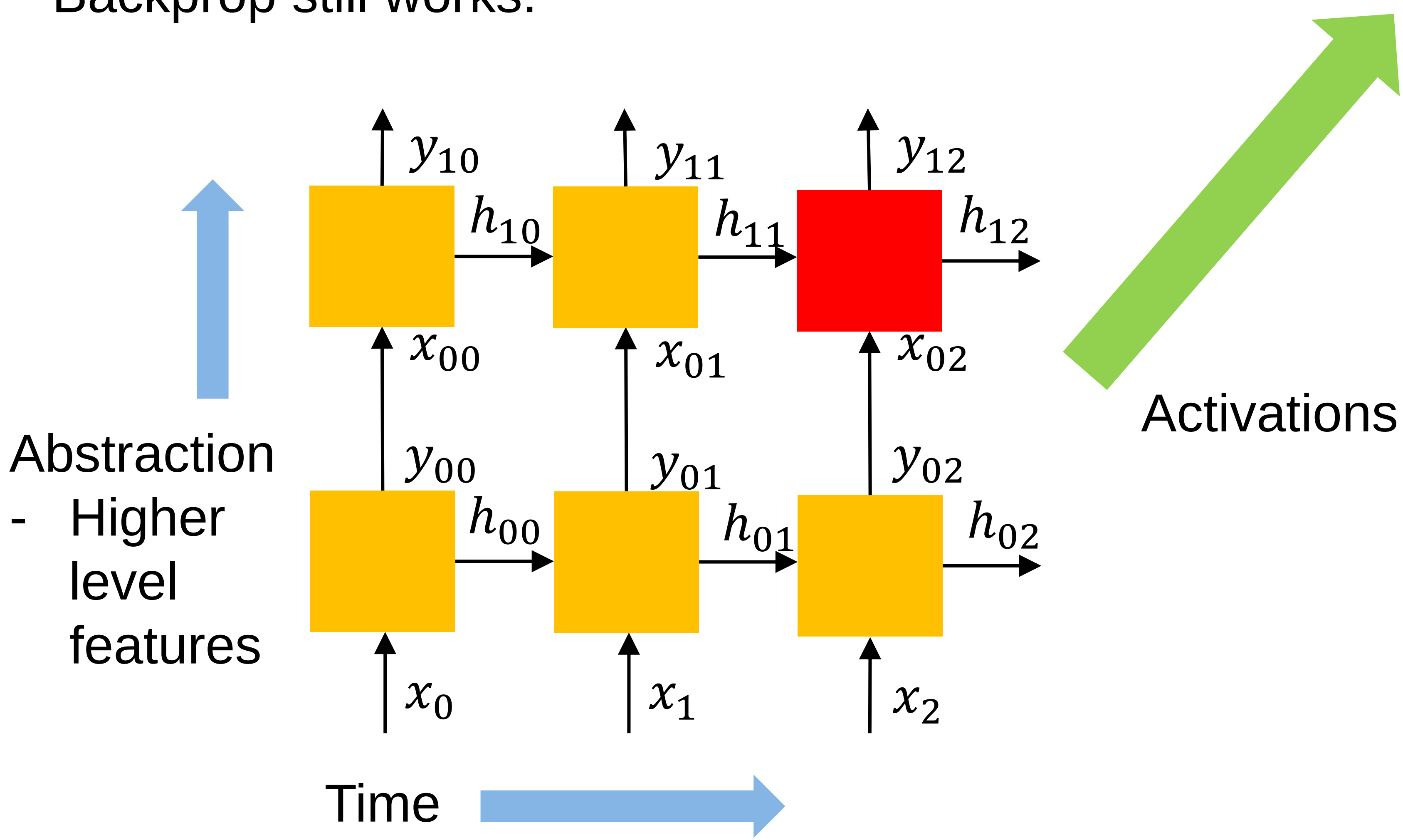
Backprop still works:

Abstraction
- Higher level features



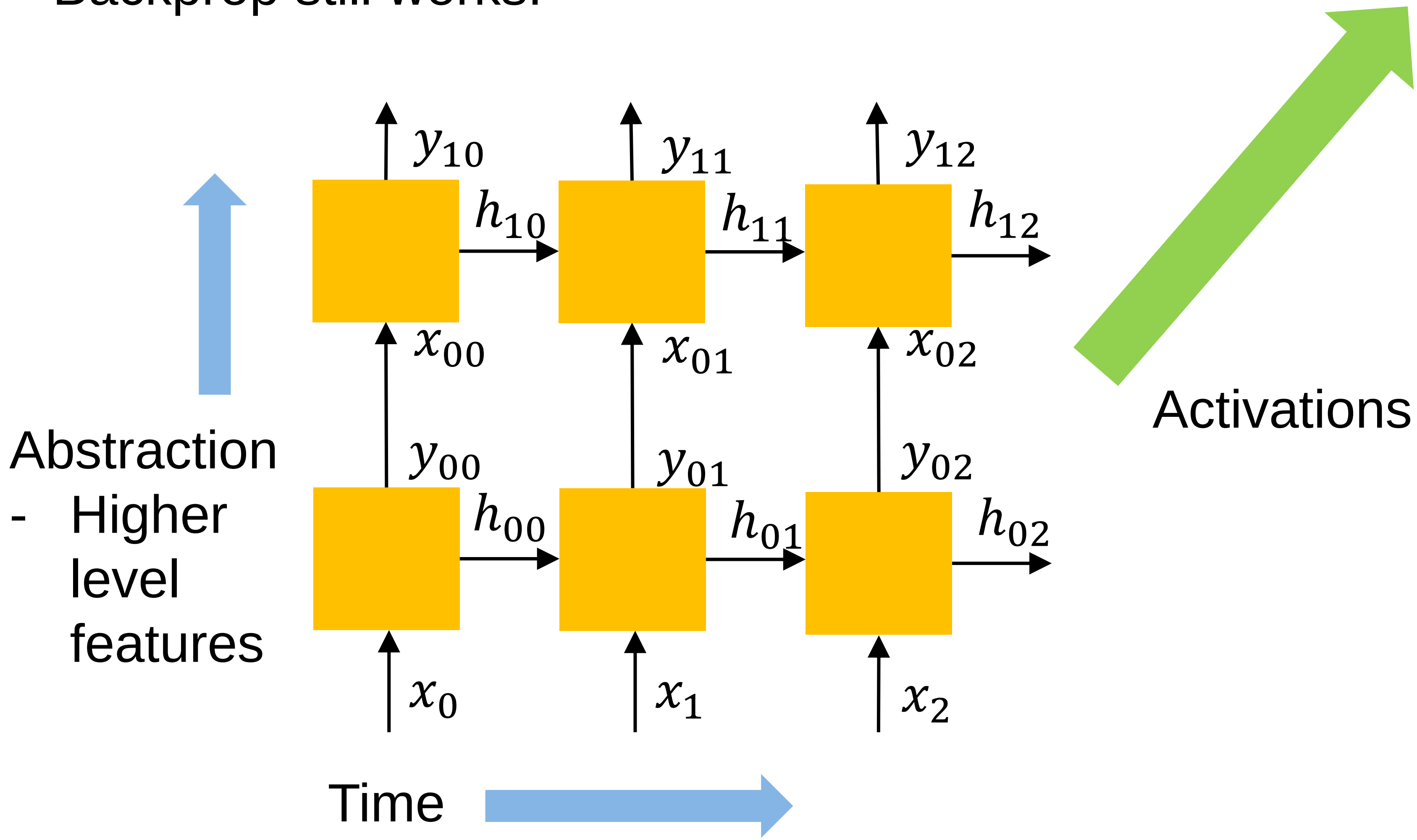
# RNN structure

Backprop still works:



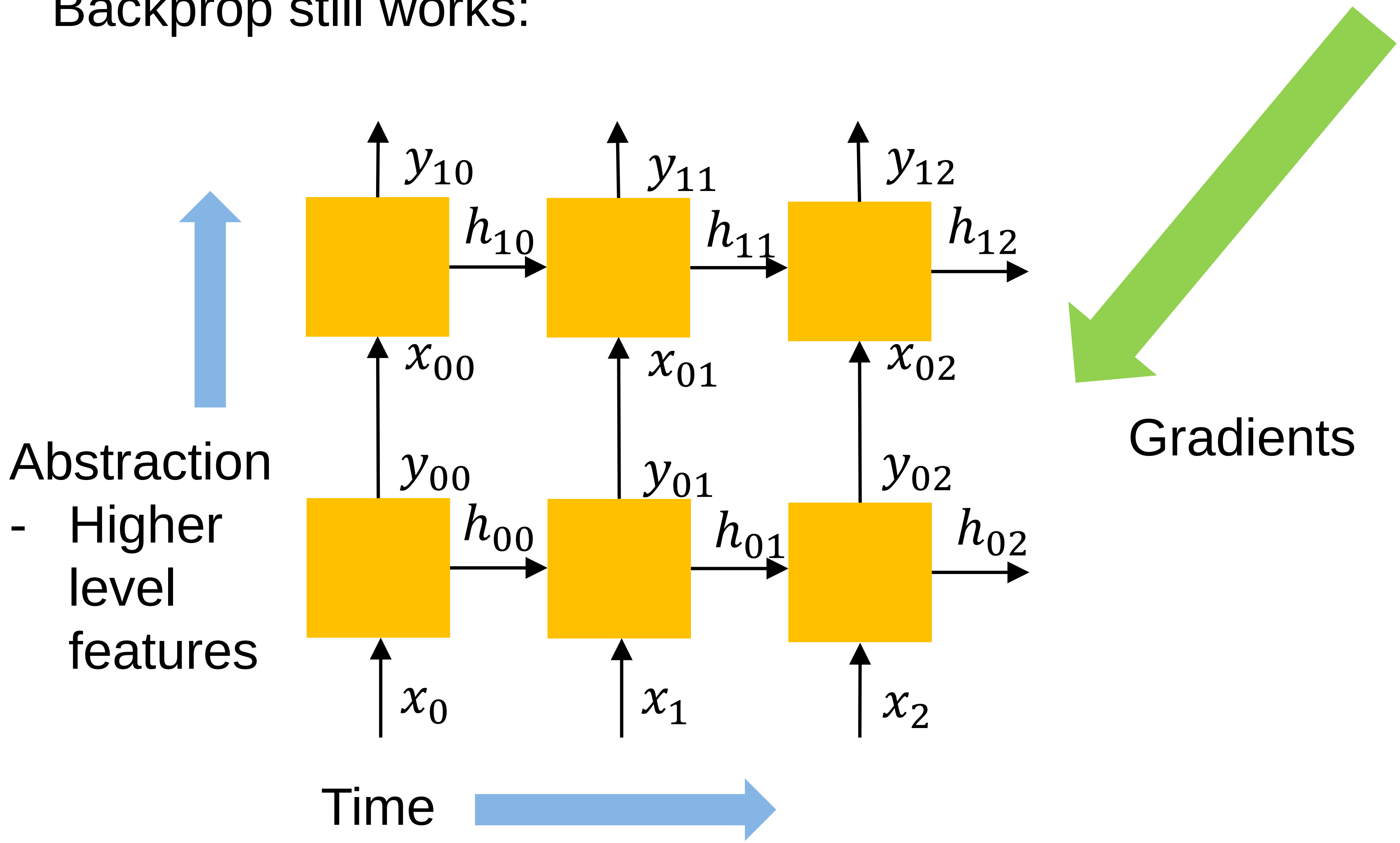
# RNN structure

Backprop still works:



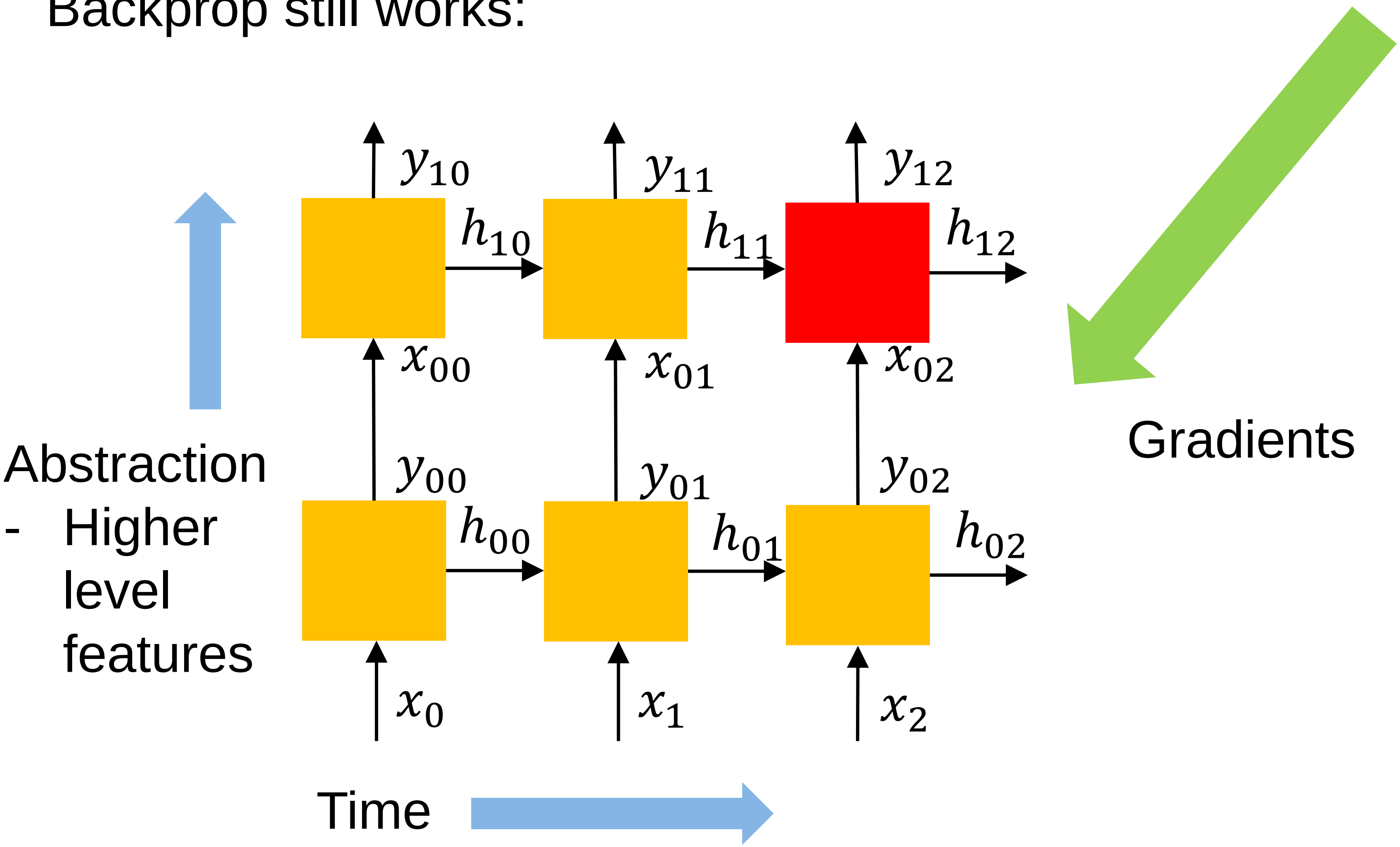
# RNN structure

Backprop still works:



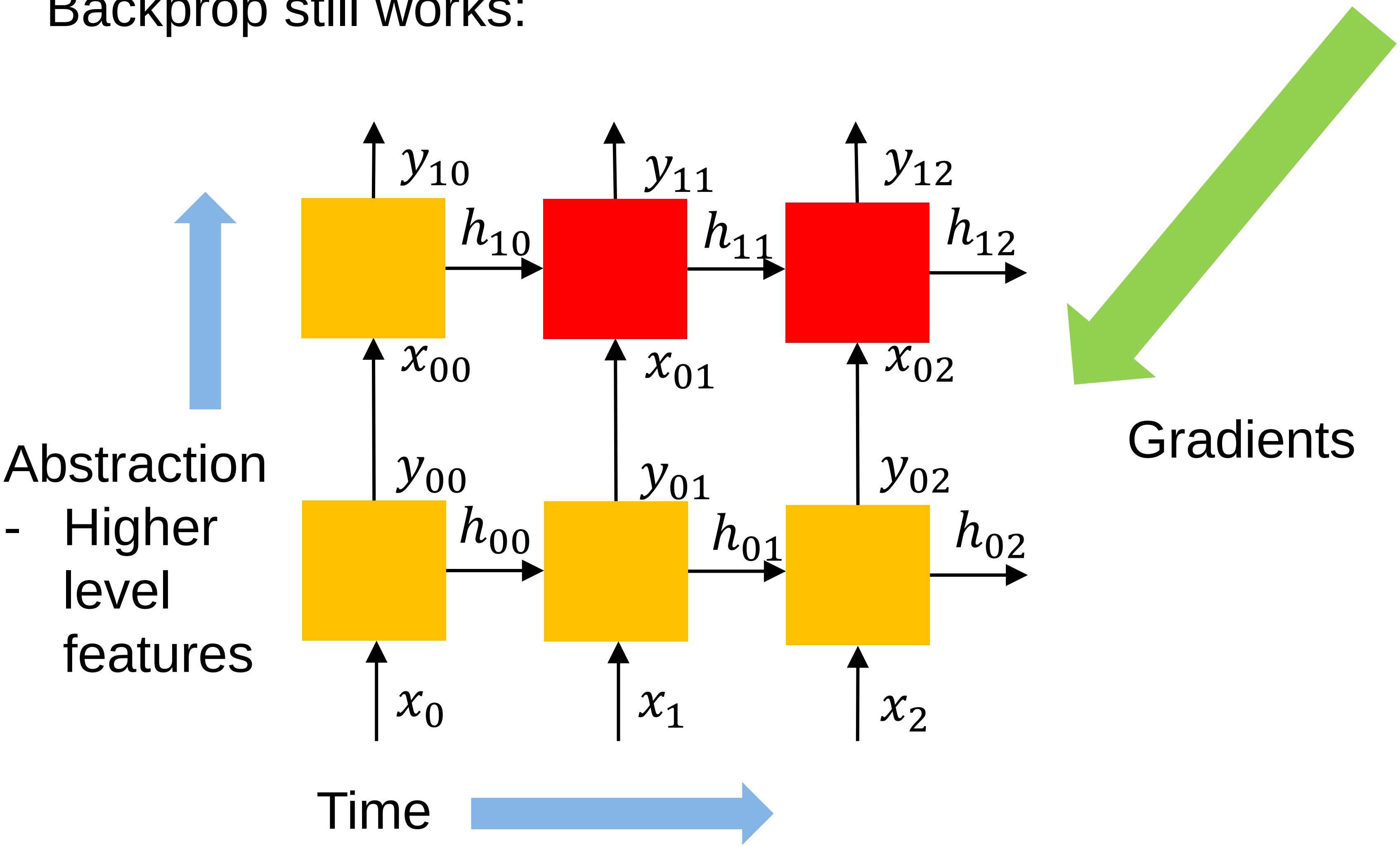
# RNN structure

Backprop still works:



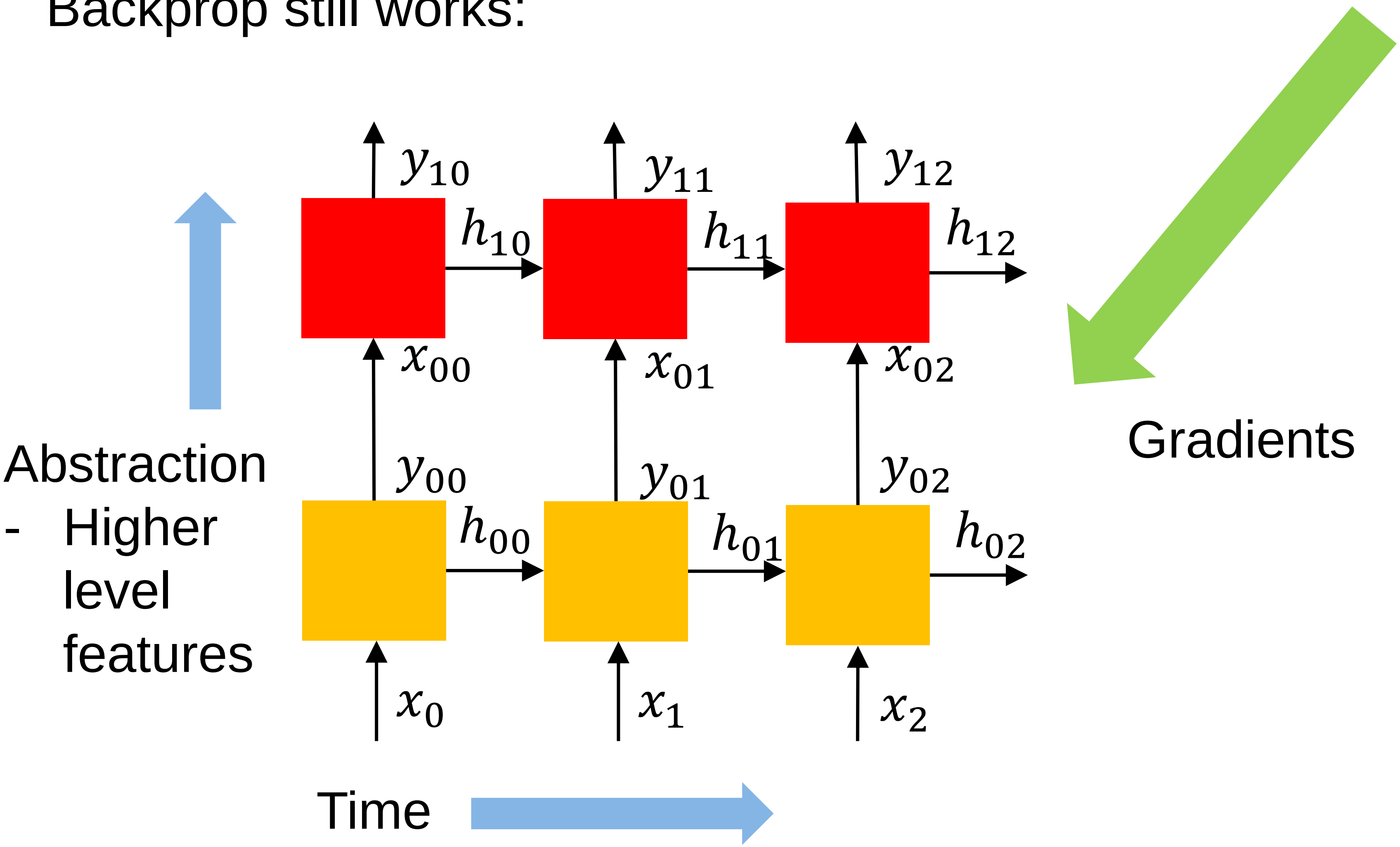
# RNN structure

Backprop still works:



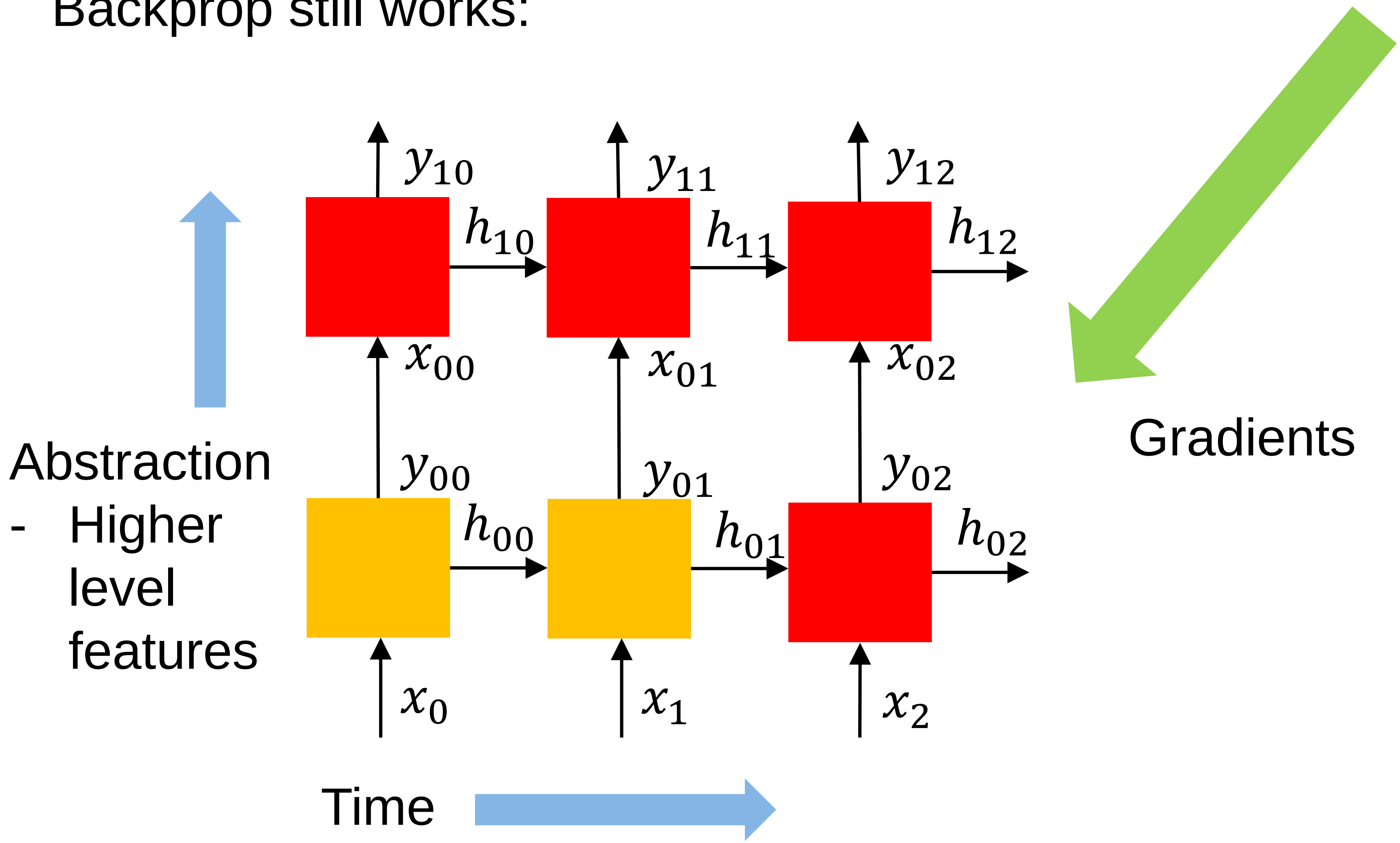
# RNN structure

Backprop still works:



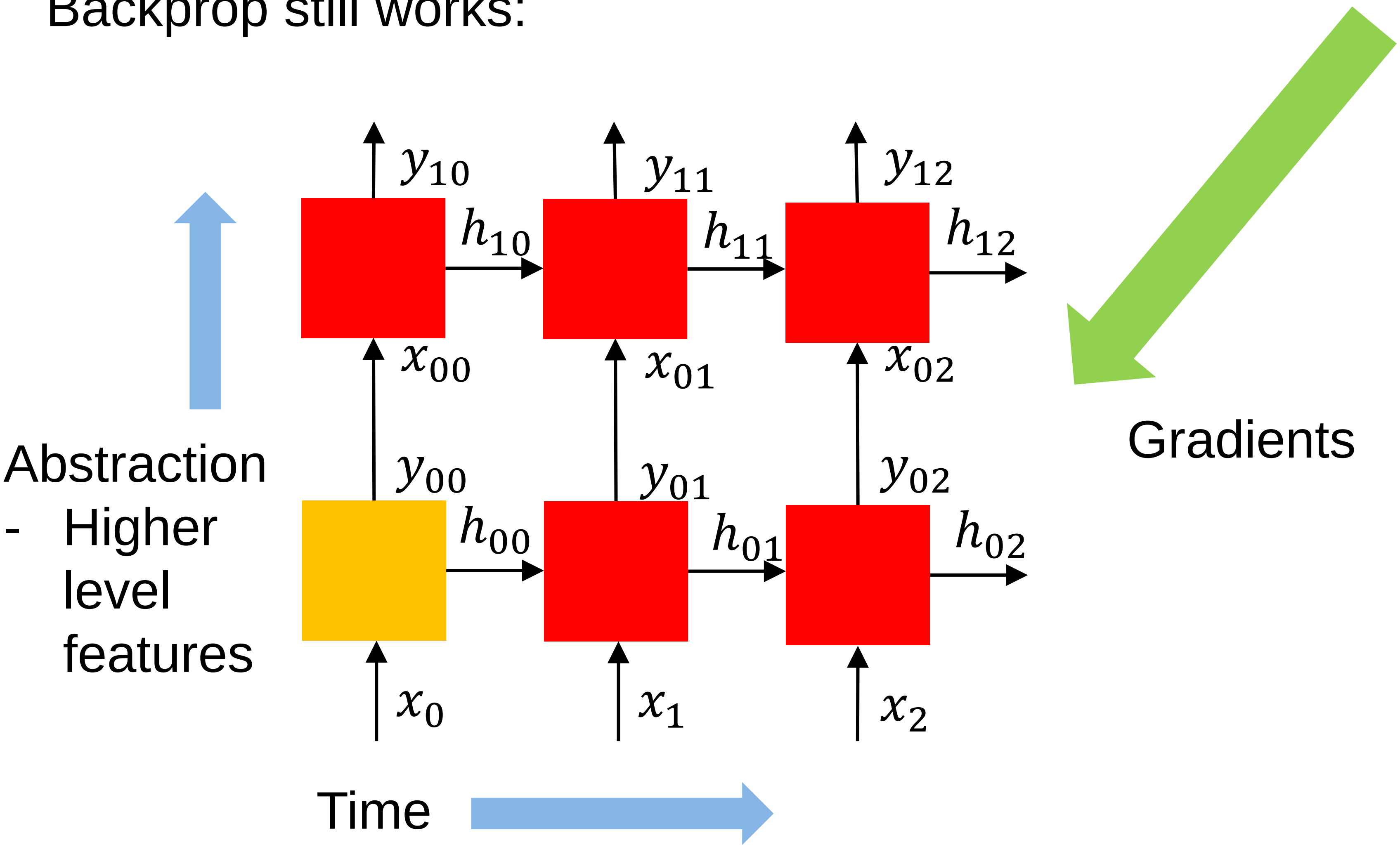
# RNN structure

Backprop still works:



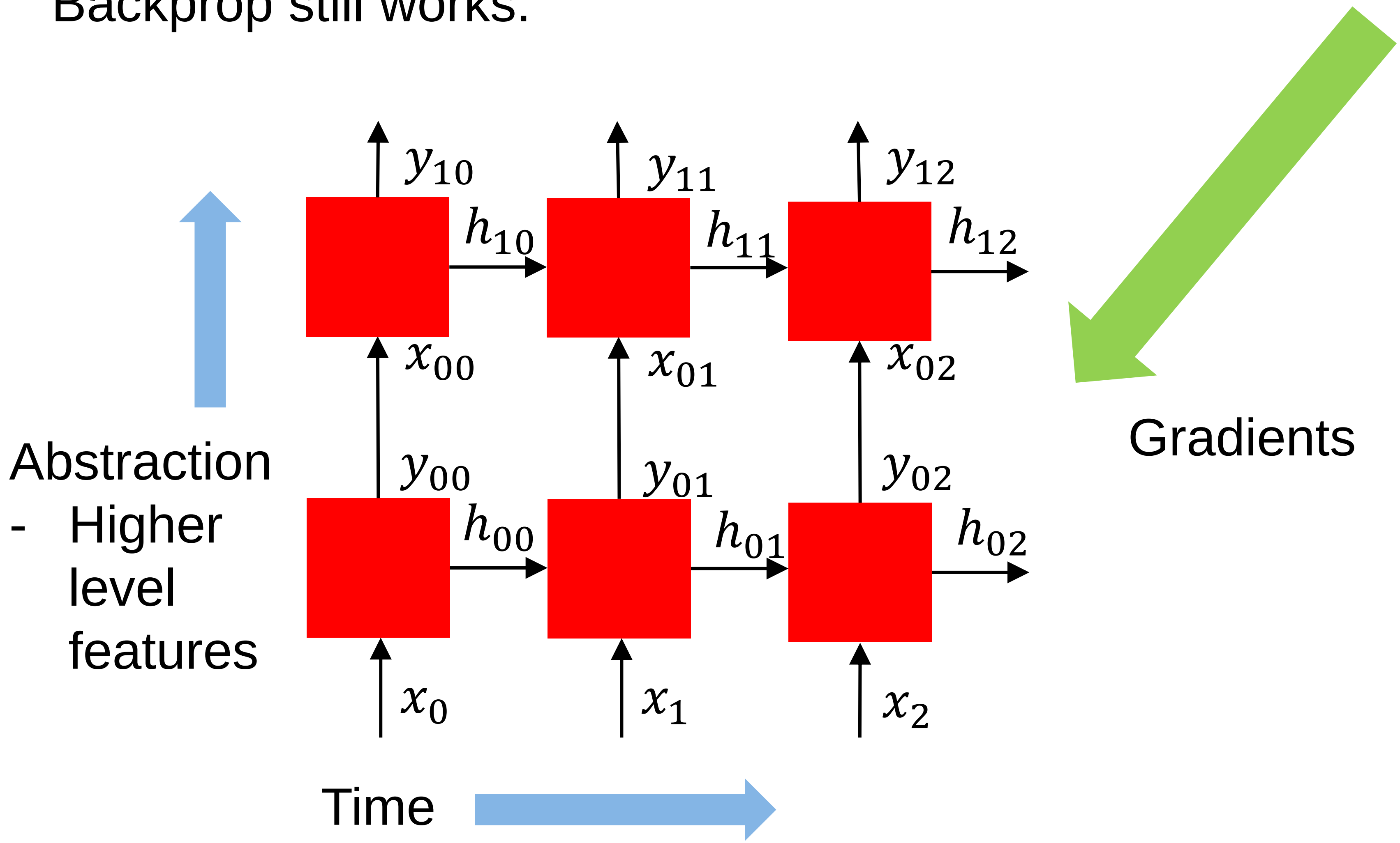
# RNN structure

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:

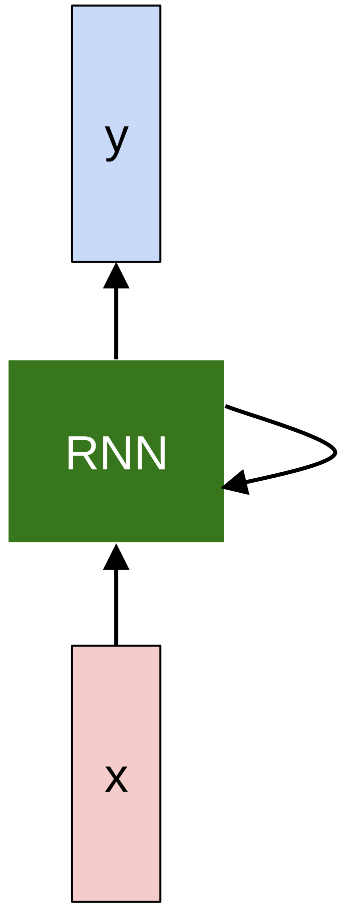
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

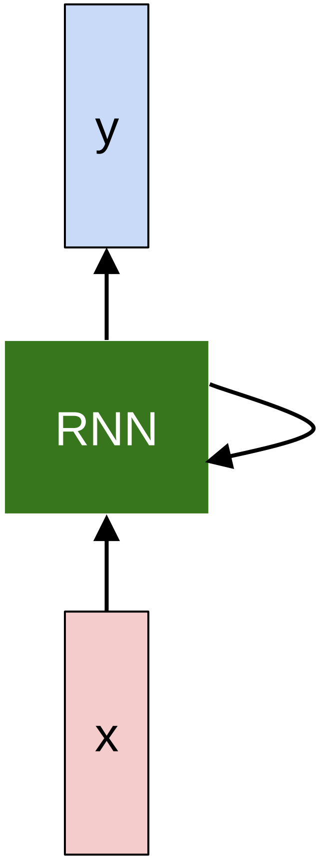
input vector at some time step

some function with parameters W



# Recurrent Neural Network

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

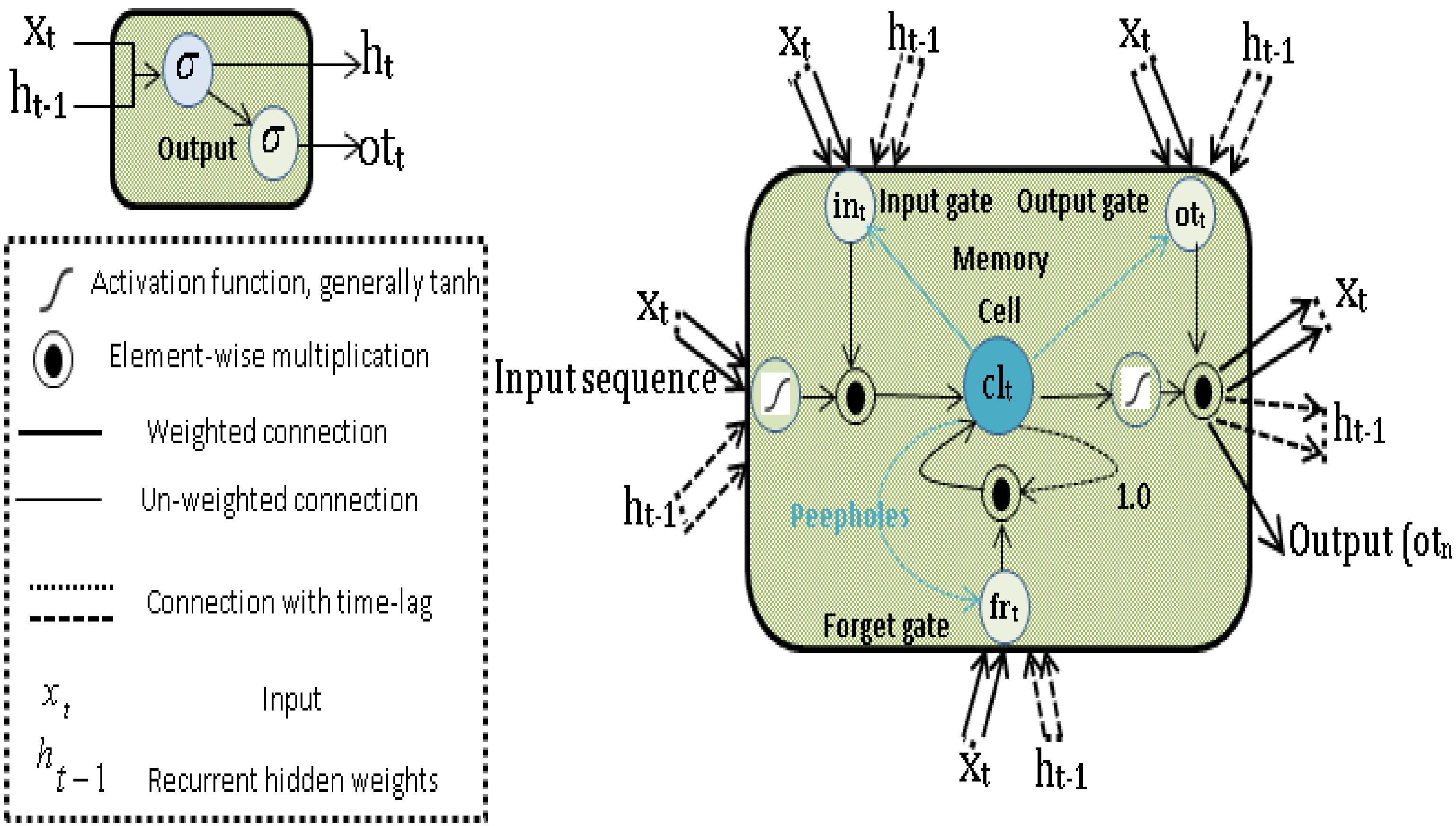



$$h_t = f_W(h_{t-1}, x_t)$$

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

$$y_t = W_{hy} h_t$$

# Long short-term memory



# Long short-term memory

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

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

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

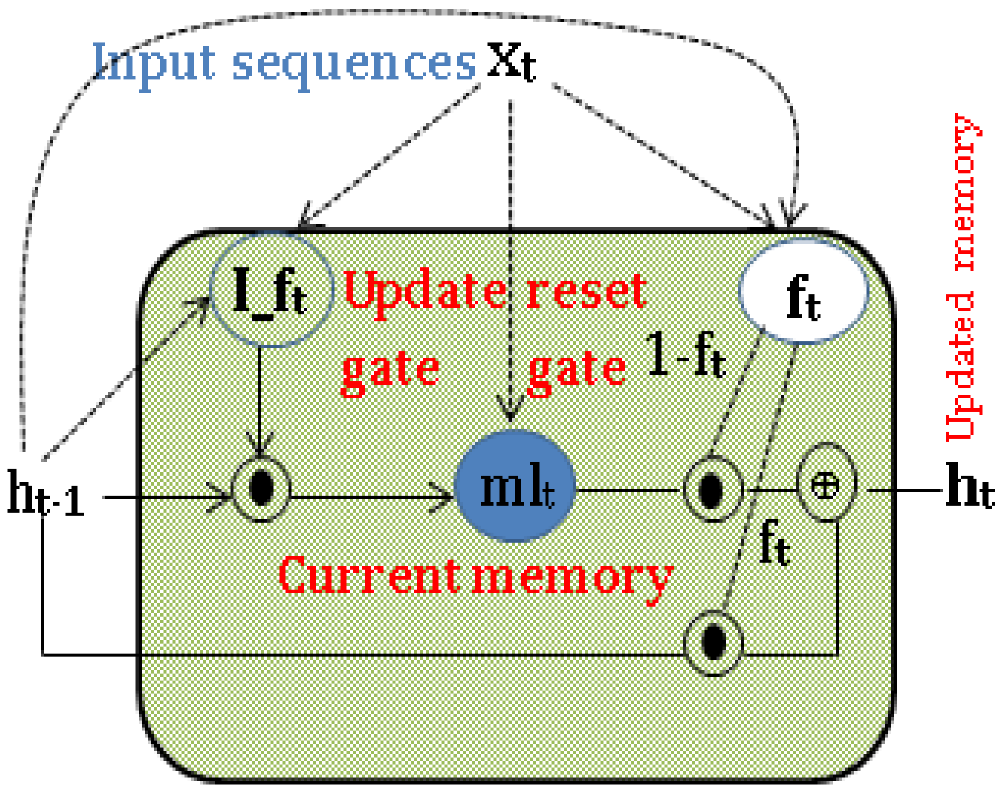
$$cl_t = fr_t \odot cl_{t-1} + in_t \odot \tanh(w_{xcl} x_t + w_{hcl} hi_{t-1} + b_{cl})$$

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

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

# 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

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

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

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

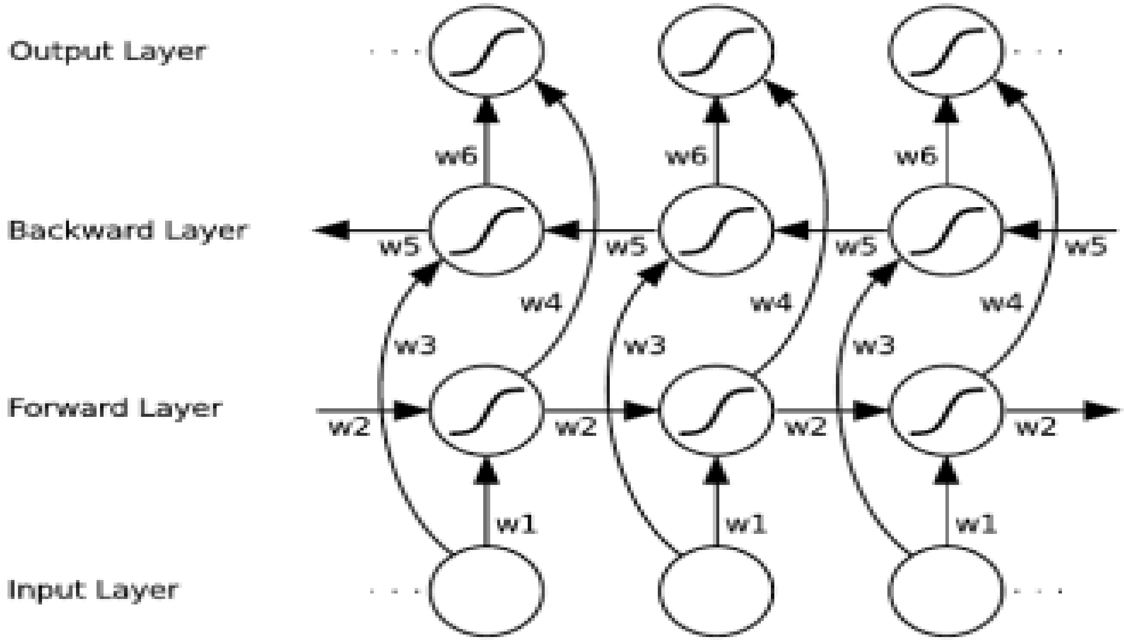
$$cl_t = \tanh(w_{xcl} x_t + w_{hcl}(fr \odot hi_{t-1}) + b_{cl})$$ (Current memory)

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



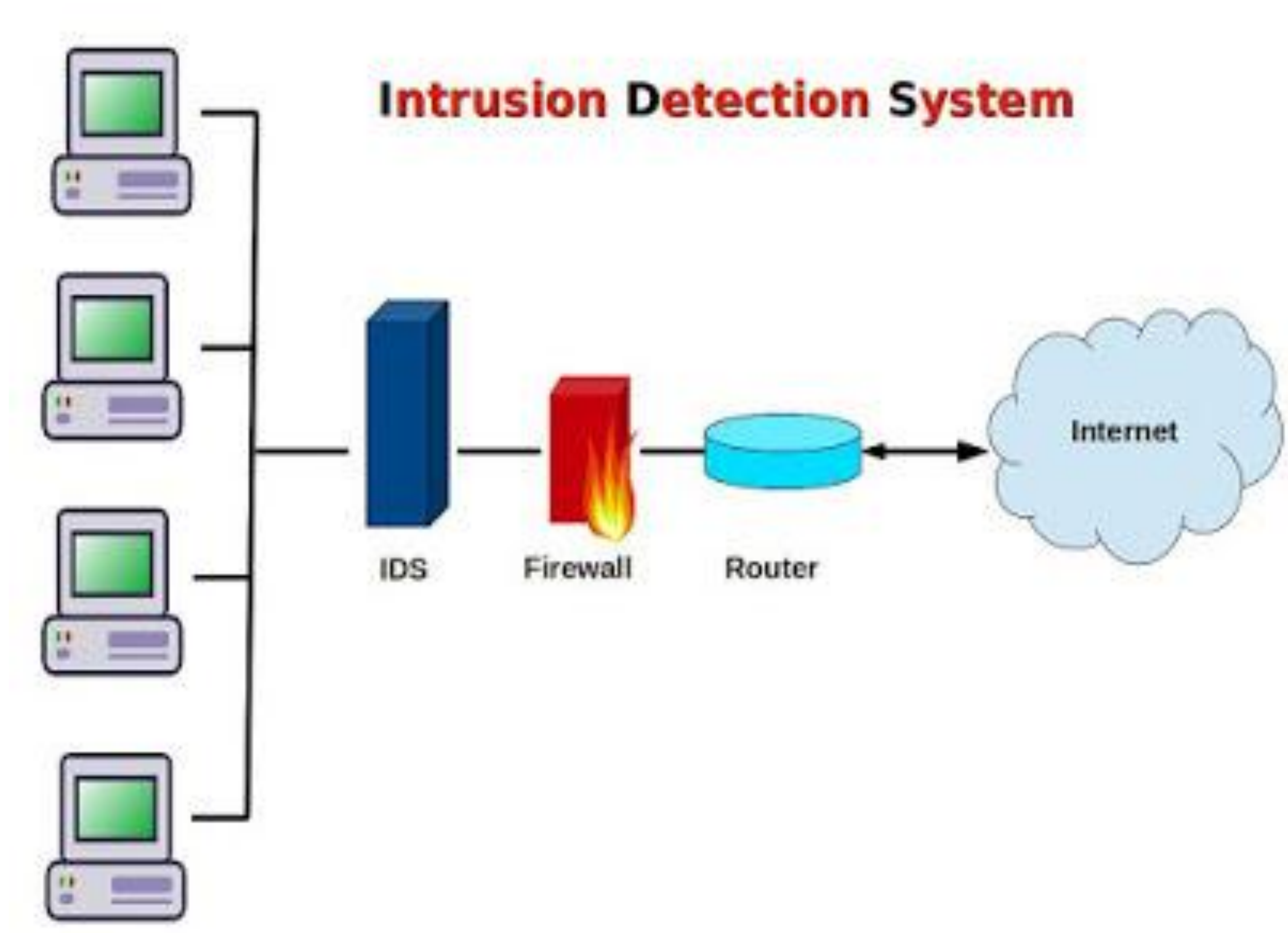
# Summary

- RNNs allow a lot of flexibility in architecture design
- RNNs are simple but don’t work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed.

# Case studies

Cyber Security use case: Intrusion detection

# Case studies



# Intrusion detection

- Intrusion : Attempting to break into or misuse your system.
- Intruders may be from outside the network or legitimate users of the network.
- Intrusion can be a physical, system or remote intrusion.
- Intrusion Detection Systems look for attack signatures, which are specific patterns that usually indicate malicious or suspicious intent.
- anomaly detection, signature based, host based and network based are different IDS.
- There are various approaches to attack malicious activities, namely (1) static approaches: firewalls, encryption and decryption techniques of cryptography, software updates and many others and (2) dynamic approaches: anomaly and intrusion detection (ID).

# Intrusion detection

- Intrusion detection is categorized in to 2 types based on the network behavior and network type.
- Network-based IDS (N-IDS): Most commonly used in both academia and industries, it analyzes all the network traffic by looking in packet level information to find the suspicious activity.
- Host-based IDS (H-IDS): focuses on the information of each particular system or host, heavily depends for data on the sources of log files such as sensors, system logs, software logs, file systems, disk resources and many more. Mostly, organizations used combination of both of them to get benefited largely in real-time IDS deployment.
- Lincoln Labs - raw TCP dump data collected from a local-area network (LAN).

# Intrusion detection

- The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

- A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

- **DOS** - Intruder aims at making network resources down and consequently, resources are inaccessible to authorized users, e.g. syn flood.
- **Probe** - acquiring statistics about the computer System or network e.g., port scanning.
- **R2L** - Illegitimate access from remote computer , e.g. guessing password.

# Intrusion detection

| Attack category | Full data set | 10% data set | |
|---|---|---|---|
| | KDDCup '99' | KDDCup '99' | |
| | Train | Train | Test |
| Normal | 972780 | 97278 | 60593 |
| DoS | 3883370 | 391458 | 229853 |
| Probe | 41102 | 4107 | 4166 |
| R2L | 1126 | 1126 | 16189 |
| U2R | 52 | 52 | 228 |
| Total | | **494021** | **311029** |

Description of Data set

# Intrusion detection

Demo

Thank you

Questions ?

vinayakumarr77@gmail.com

https://vinayakumarr.github.io/

Hands on session on “Machine learning and Deep learning using Scikit-learn, Tensorflow and Keras”

# 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

**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.

**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

**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

**IPython**
Enhanced Interactive Console

**Sympy**
Symbolic mathematics

**pandas**
Data structures & analysis

# Hands – on tutorial on supporting libraries

# Hands – on tutorial on classical machine learning algorithms using scikit-learn

# Hands – on tutorial on Tensorflow with Keras

# References

1. R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran: Applying convolutional neural network for network intrusion detection. ICACCI 2017: 1222-1228

2. R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran: Evaluating effectiveness of shallow and deep networks to intrusion detection system. ICACCI 2017: 1282-1289

3. R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran: Evaluation of Recurrent Neural Network and its variants for Intrusion Detection System (IDS)" has accepted in Special Issue On Big Data Searching, Mining, Optimization & Securing (BSMOS) Peer to Peer Cloud Based Networks in IJISMD (Accepted)