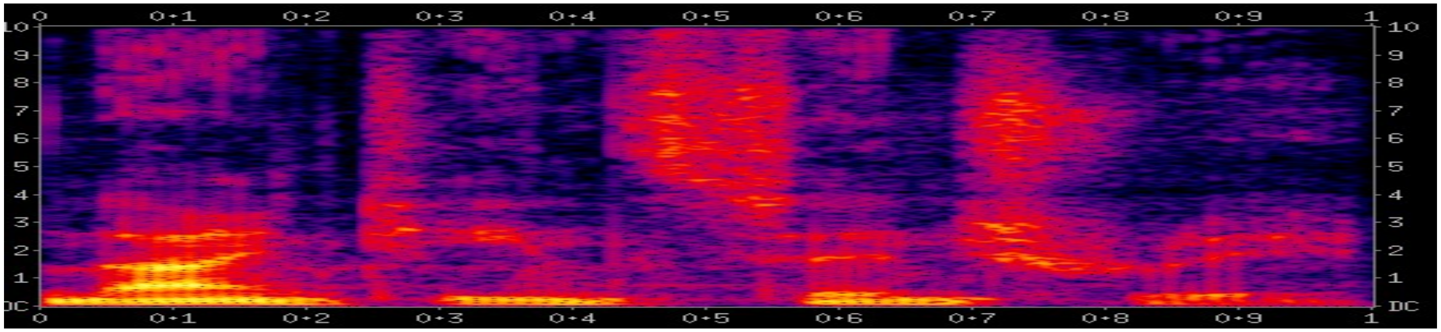
Modelling Series

- In many situations one must consider a *series* of inputs to produce an output
  - Outputs too may be a series

- Examples: ..

What did I say?

“To be” or not “to be”??



- Speech Recognition
  - Analyze a series of spectral vectors, determine what was said
- Note: Inputs are sequences of vectors. Output is a classification result

What is he talking about?

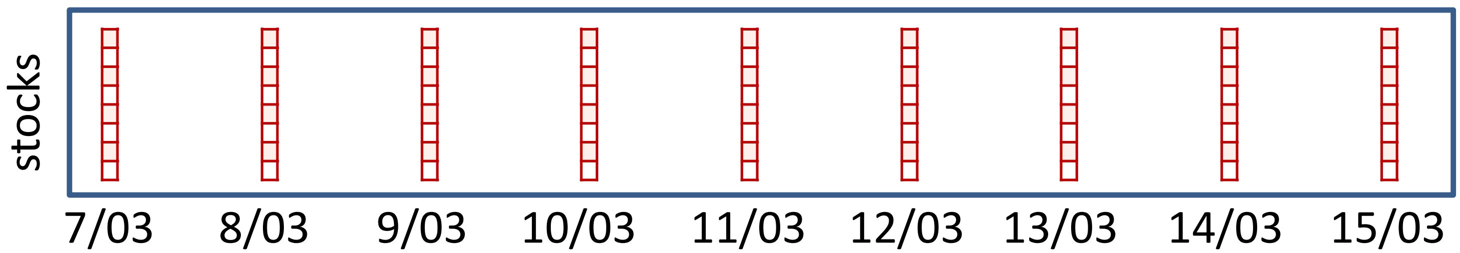
"Football" or "basketball"?

The Steelers, meanwhile, continue to struggle to make stops on defense. They've allowed, on average, 30 points a game, and have shown no signs of improving anytime soon.

- Text analysis
  - E.g. analyze document, identify topic
    - Input series of words, output classification output
  - E.g. read English, output French
    - Input series of words, output series of words

Should I invest..

To invest or not to invest?



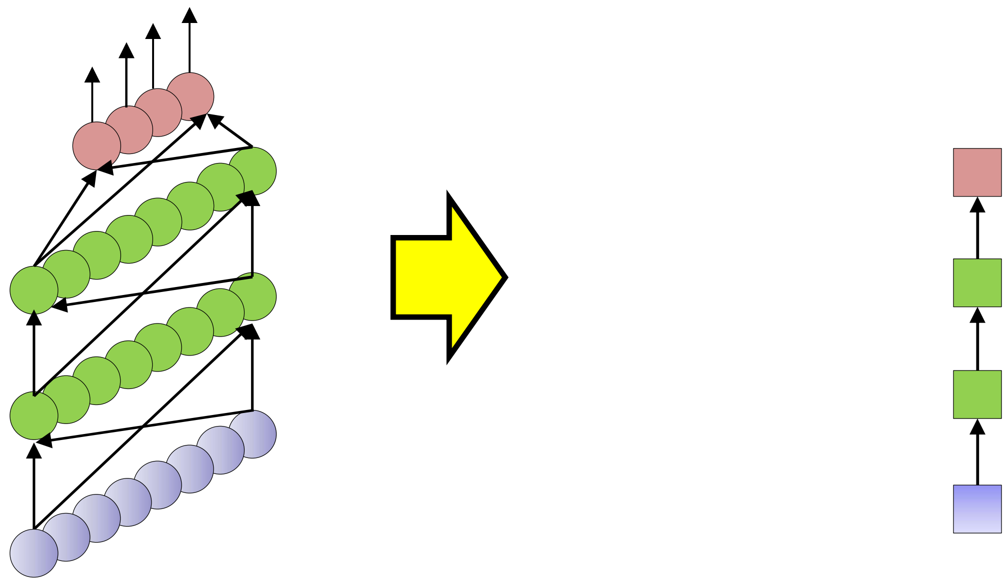
- Note: Inputs are sequences of vectors. Output may be scalar or vector
  - Should I invest, vs. should I not invest in X?
  - Decision must be taken considering how things have fared over time

These are classification and prediction problems

- Consider a sequence of inputs
  - Input vectors
- Produce one or more outputs

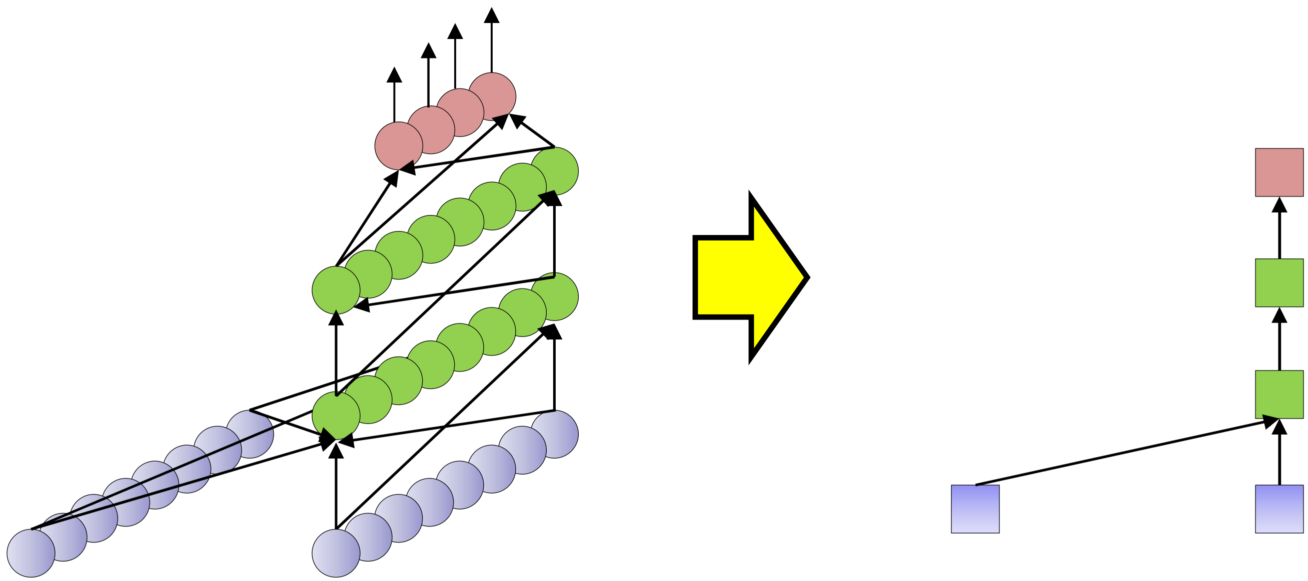
- This can be done with neural networks
  - Obviously

Representational shortcut



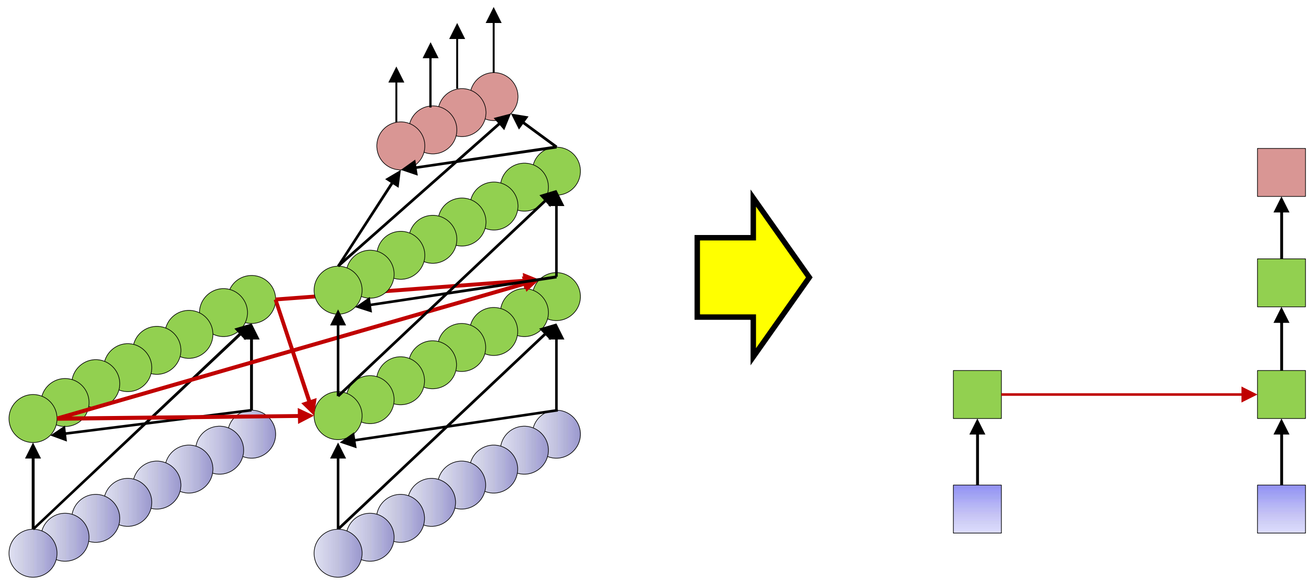
- Input at each time is a *vector*
- Each layer has many neurons
  - Output layer too may have many neurons
- But will represent everything by simple boxes
  - Each box actually represents an entire *layer with many units*

Representational shortcut



- Input at each time is a *vector*
- Each layer has many neurons
  - Output layer too may have many neurons
- But will represent everything by simple boxes
  - Each box actually represents an entire *layer with many units*

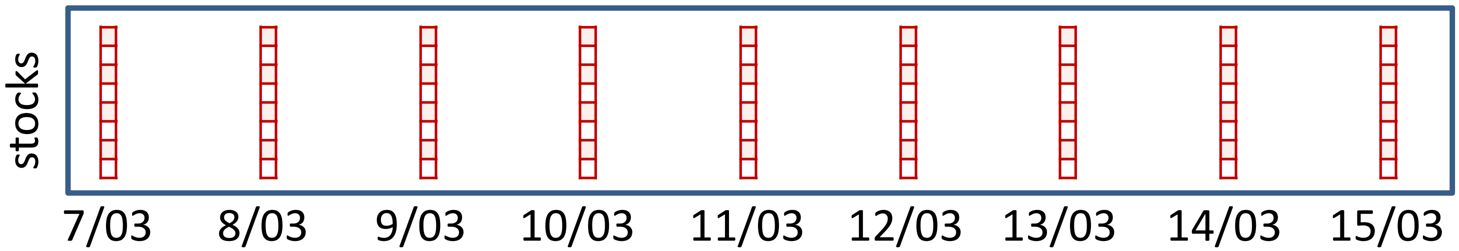
Representational shortcut



- Input at each time is a *vector*
- Each layer has many neurons
  - Output layer too may have many neurons
- But will represent everything as simple boxes
  - Each box actually represents an entire *layer with many units*

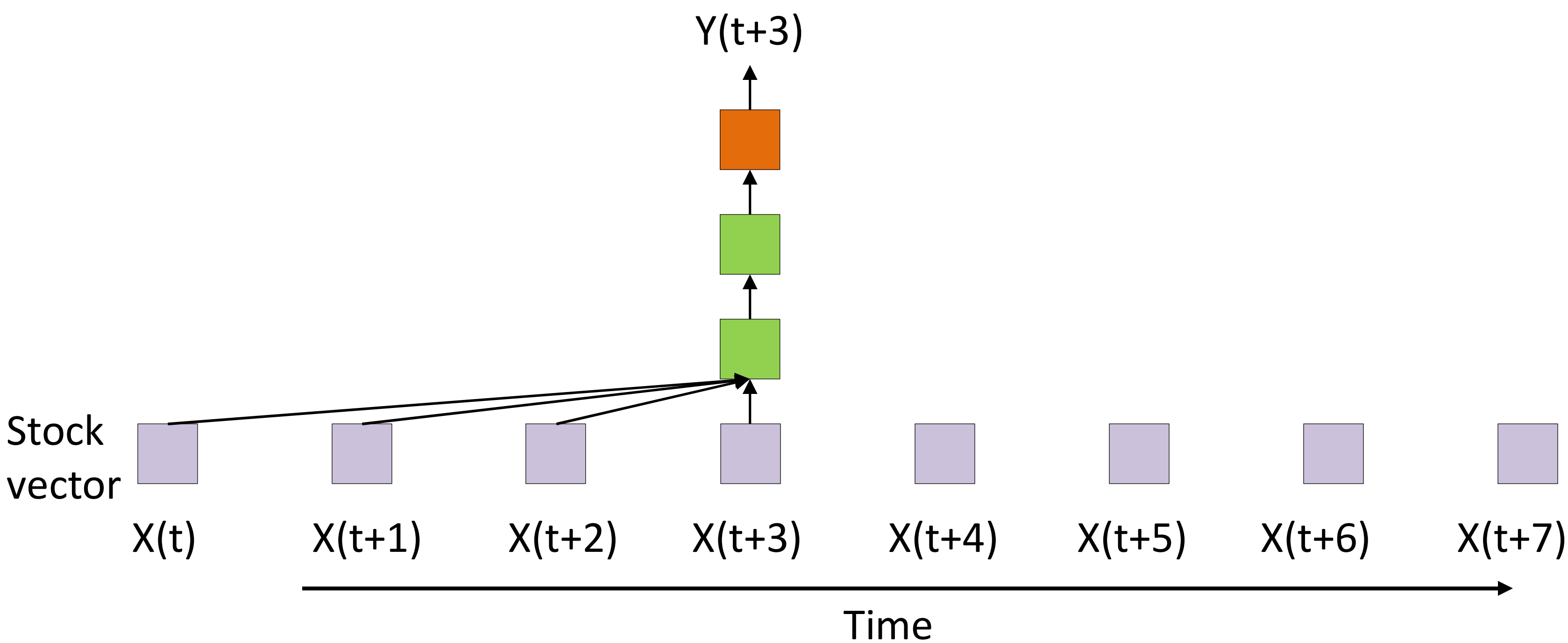
The stock prediction problem…

To invest or not to invest?



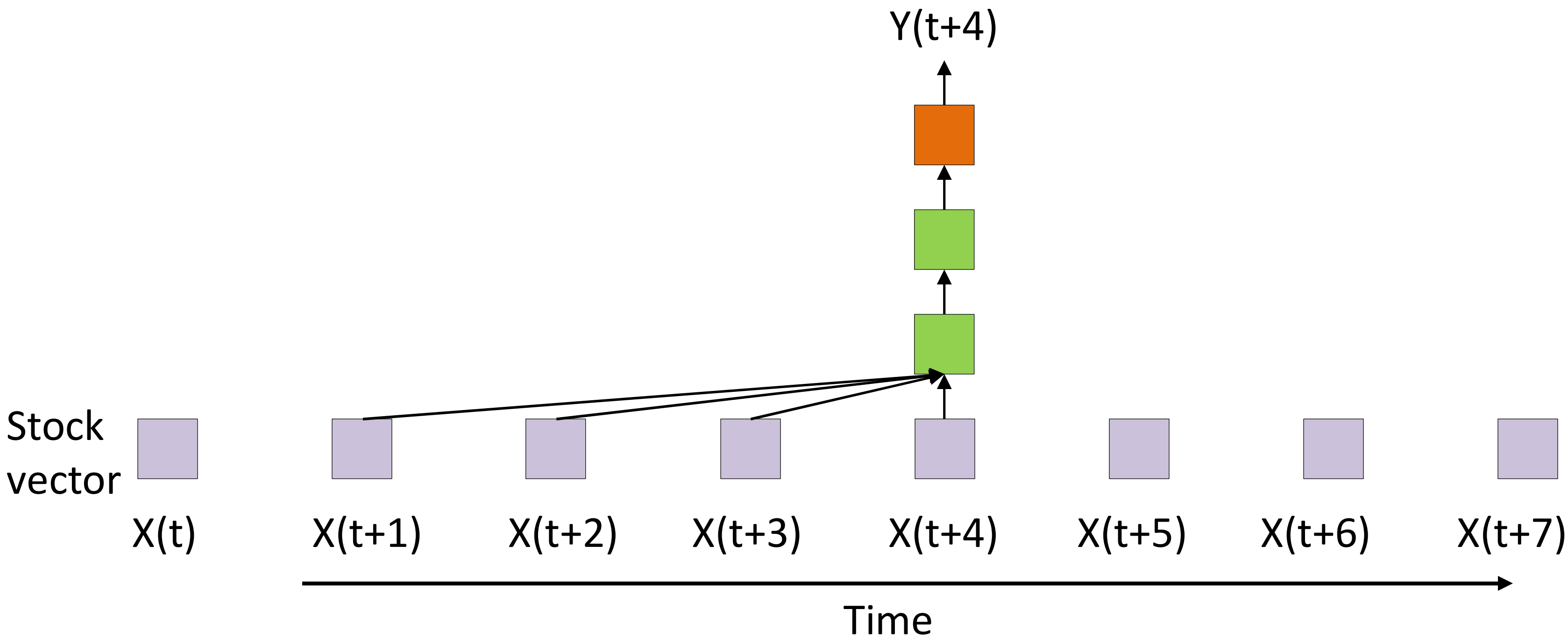
- Stock market
  - Must consider the series of stock values in the past several days to decide if it is wise to invest today
    - Ideally consider *all* of history

The stock predictor network



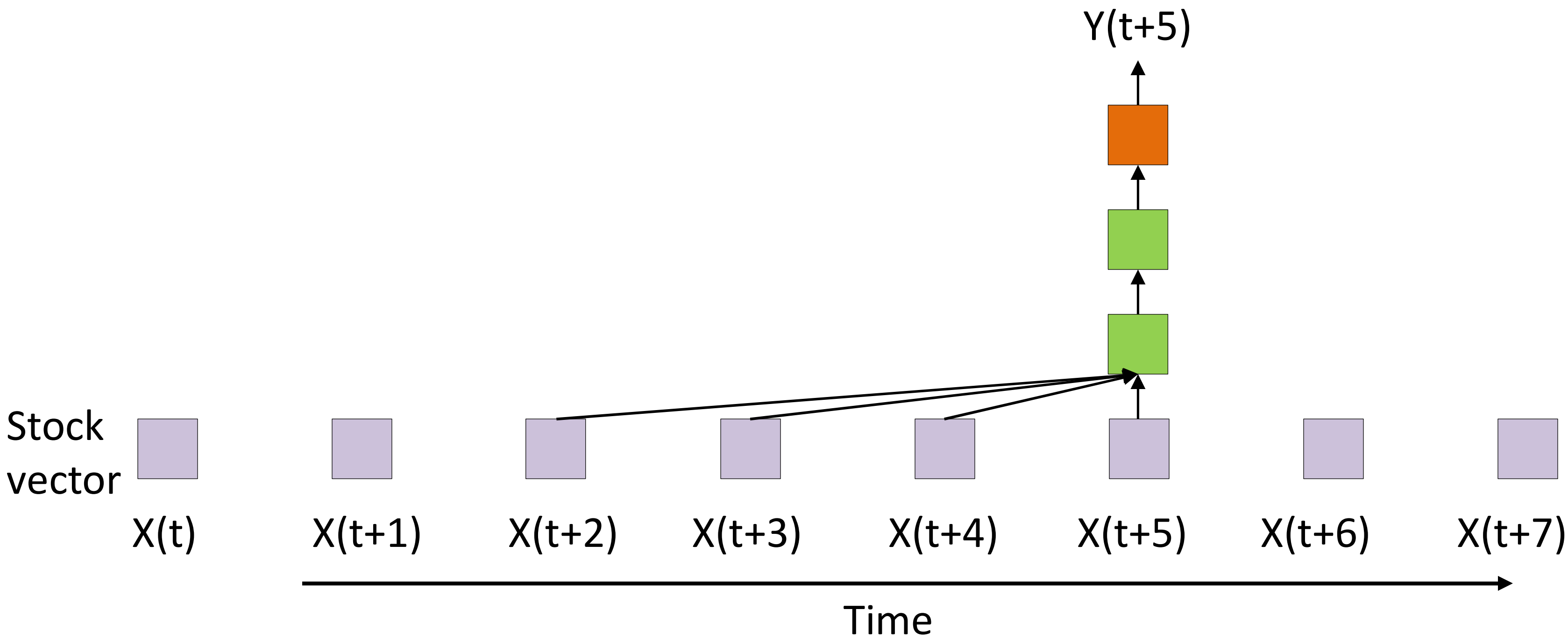
- The sliding predictor
  - Look at the last few days
  - This is just a convolutional neural net applied to series data
    - Also called a *Time-Delay neural network*

The stock predictor network



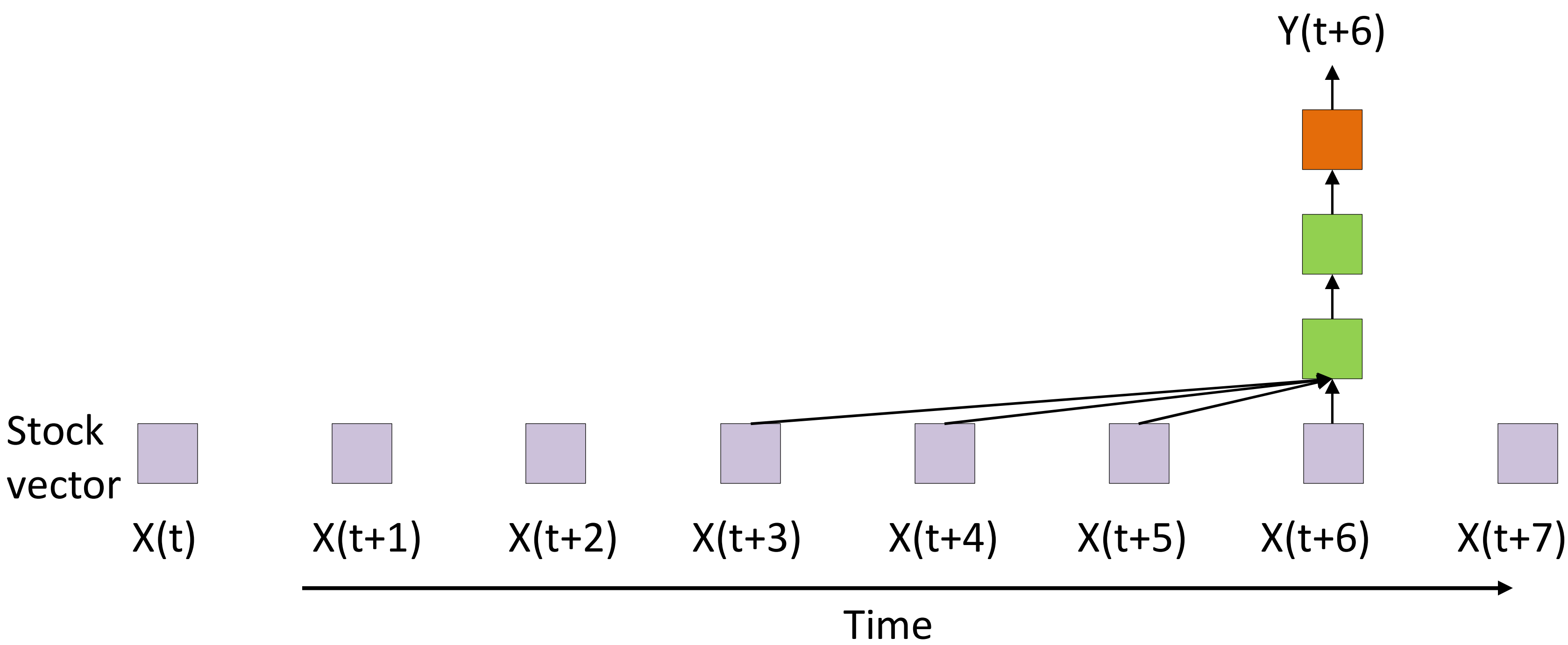
- The sliding predictor
  - Look at the last few days
  - This is just a convolutional neural net applied to series data
    - Also called a *Time-Delay neural network*

The stock predictor network



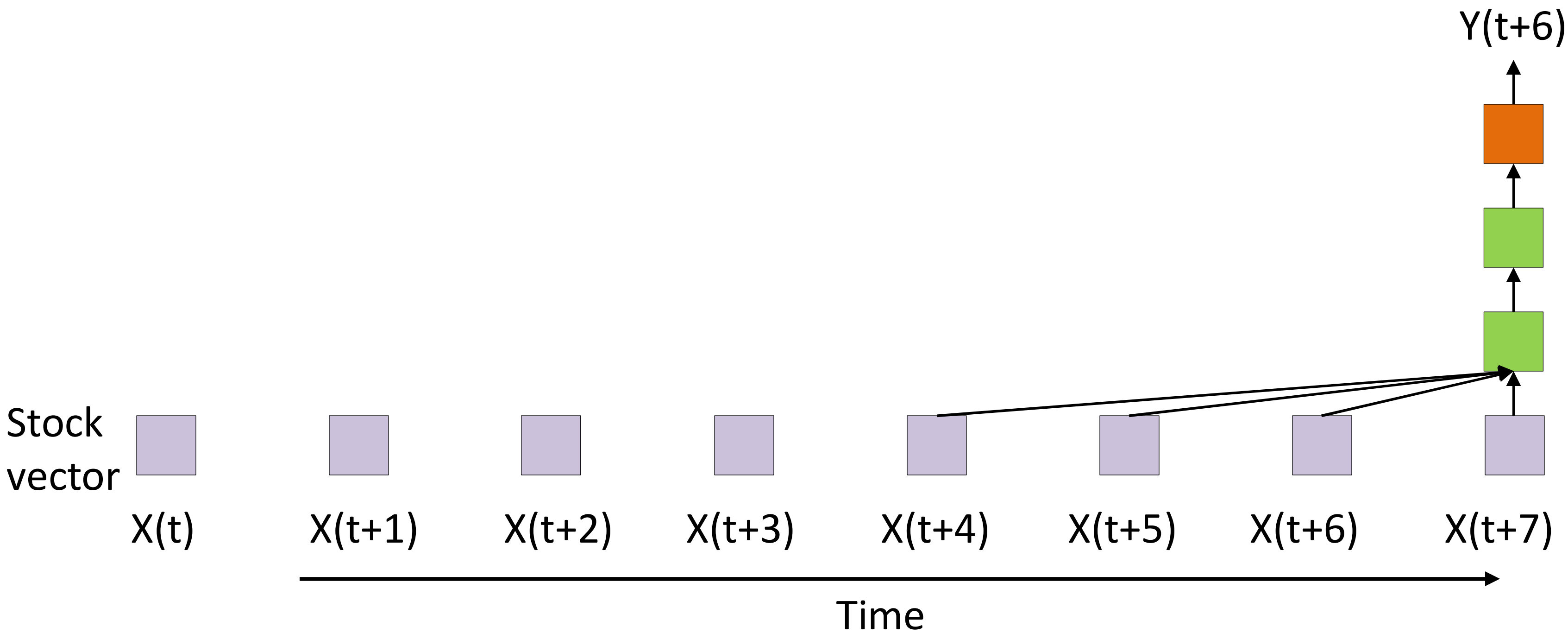
- The sliding predictor
  - Look at the last few days
  - This is just a convolutional neural net applied to series data
    - *Also called a* ***Time-Delay neural network***

The stock predictor network



- The sliding predictor
  - Look at the last few days
  - This is just a convolutional neural net applied to series data
    - Also called a *Time-Delay neural network*

The stock predictor network



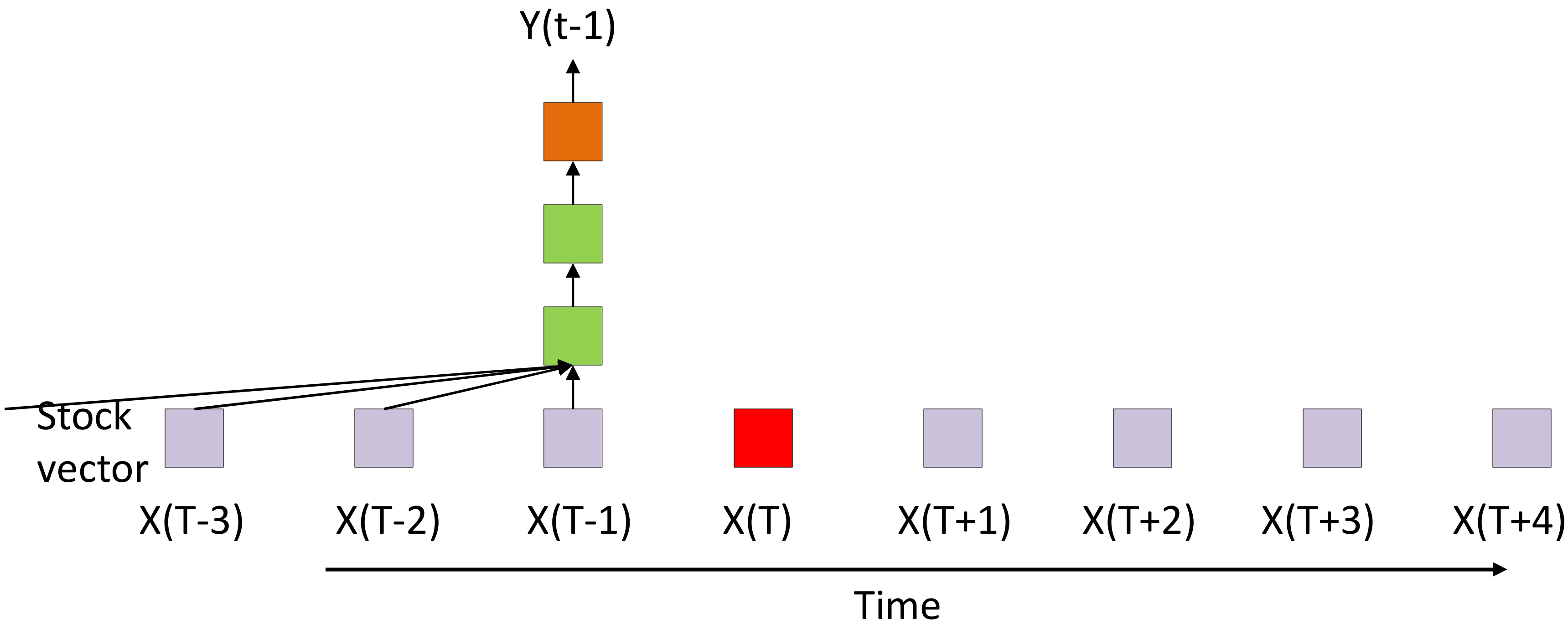
- The sliding predictor
  - Look at the last few days
  - This is just a convolutional neural net applied to series data
    - Also called a *Time-Delay neural network*

Finite-response model

- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-N})$$

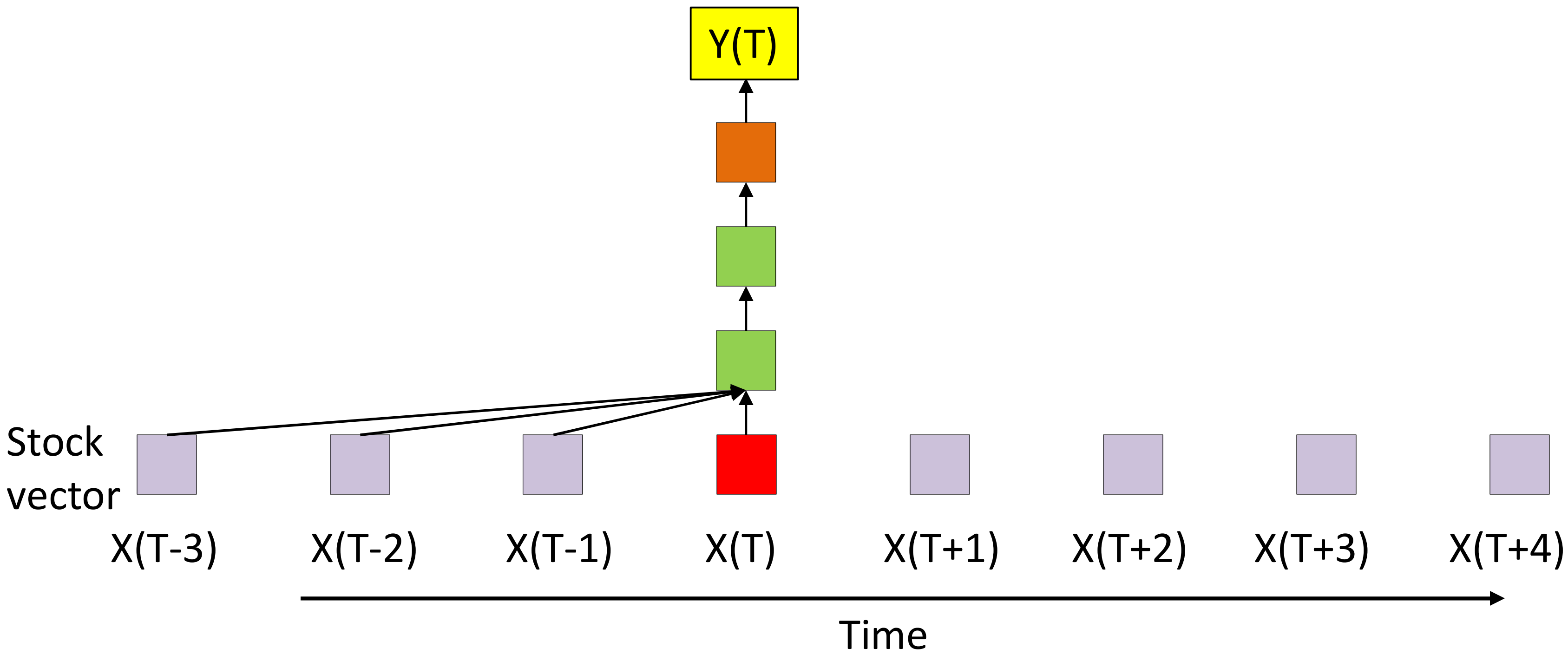
The stock predictor



- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \ldots, X_{t-N})$$

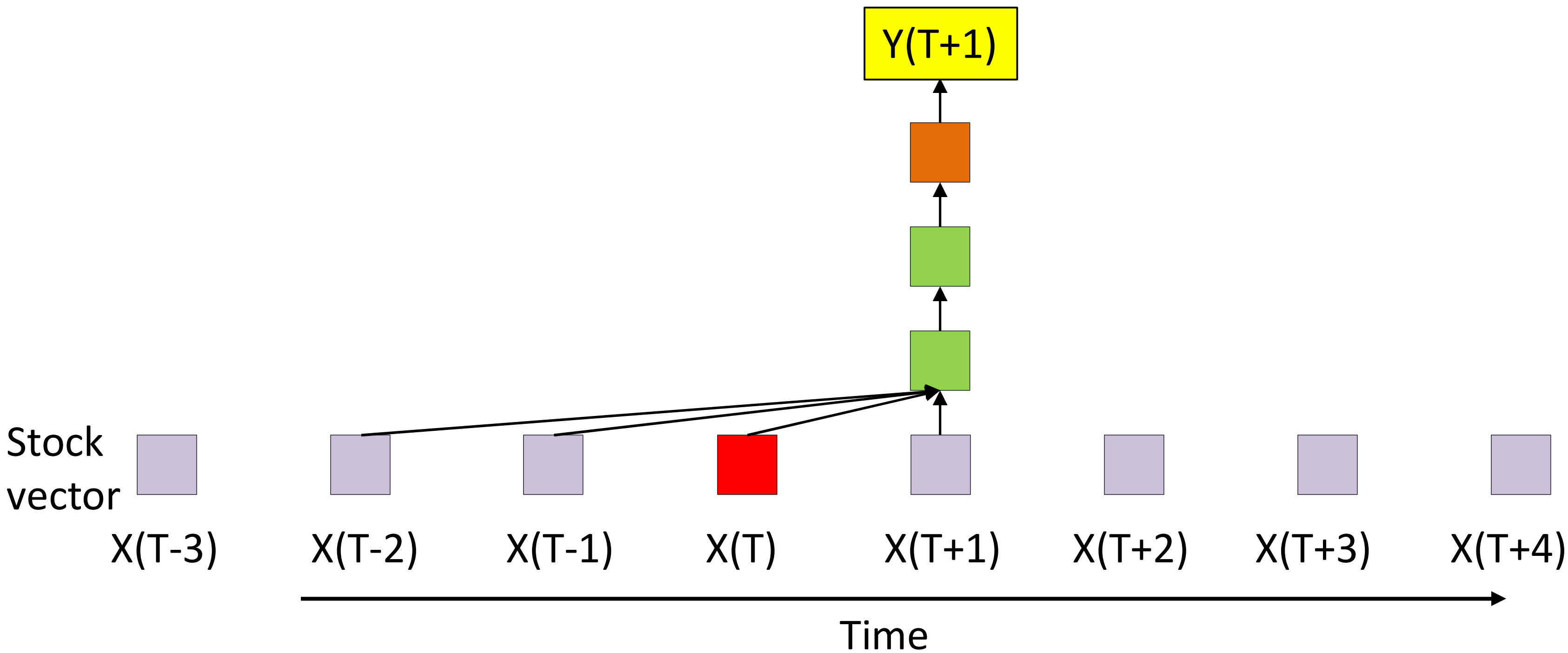
The stock predictor



- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \ldots, X_{t-N})$$

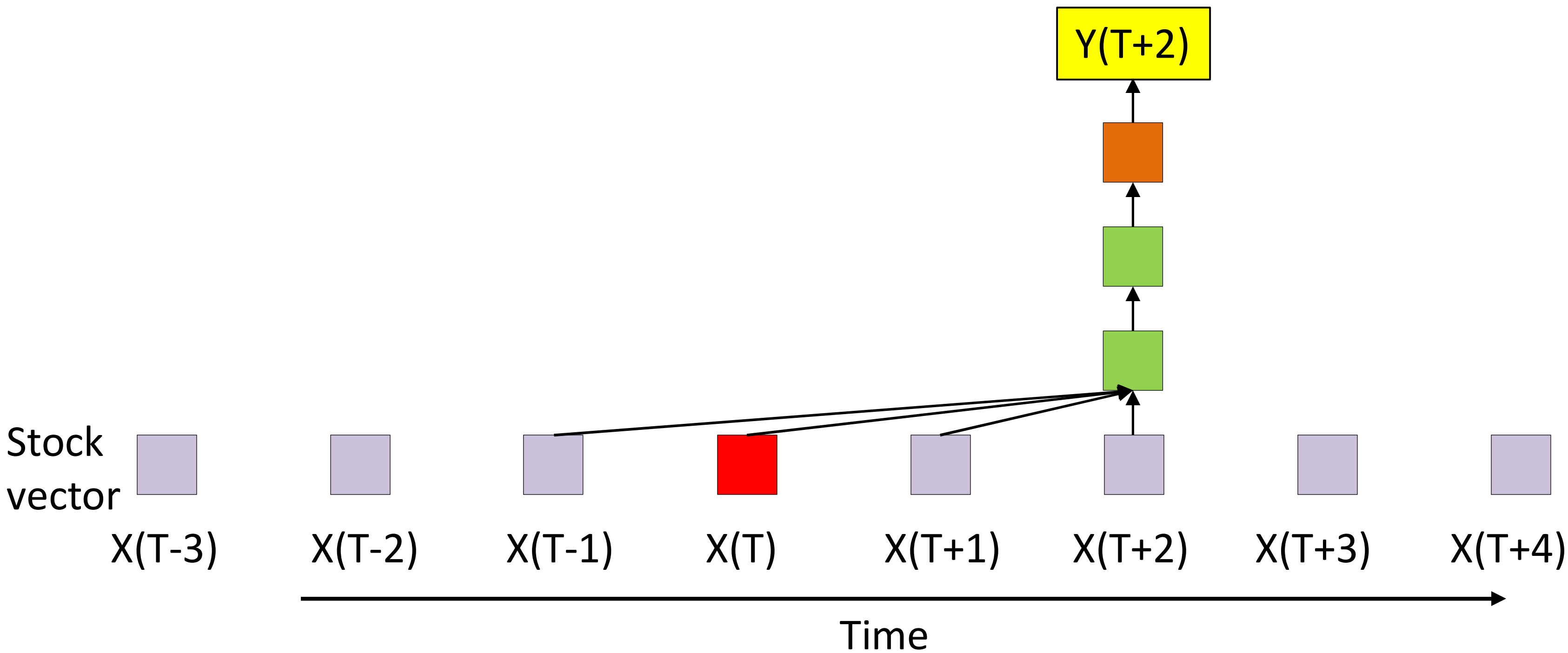
The stock predictor



- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-N})$$

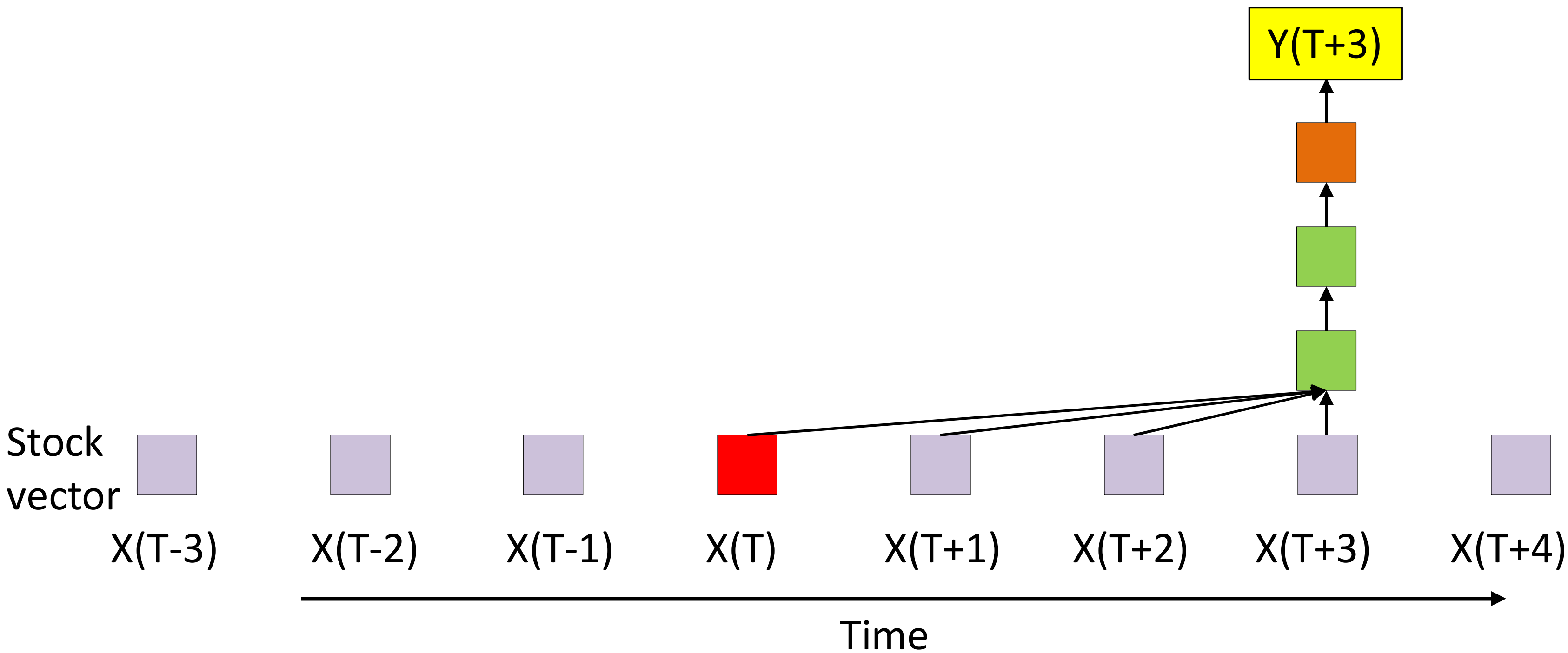
The stock predictor



- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \ldots, X_{t-N})$$

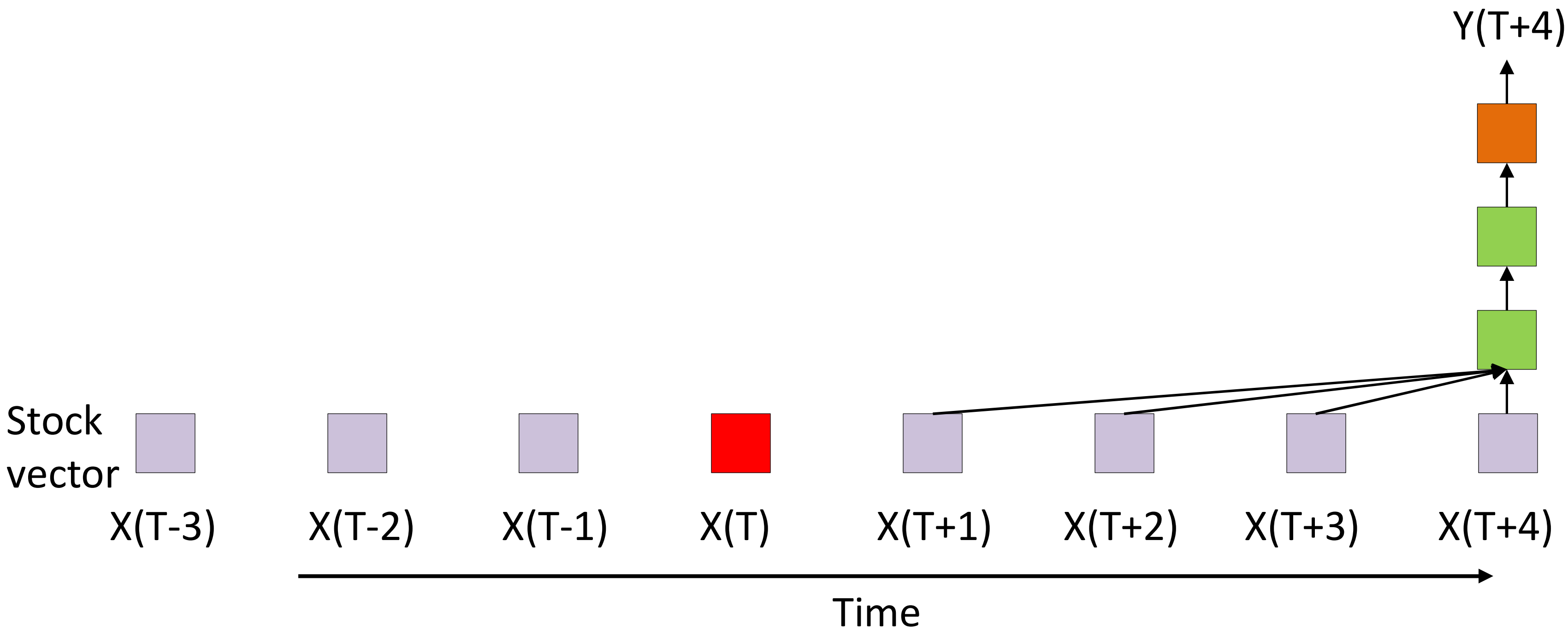
The stock predictor



- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-N})$$

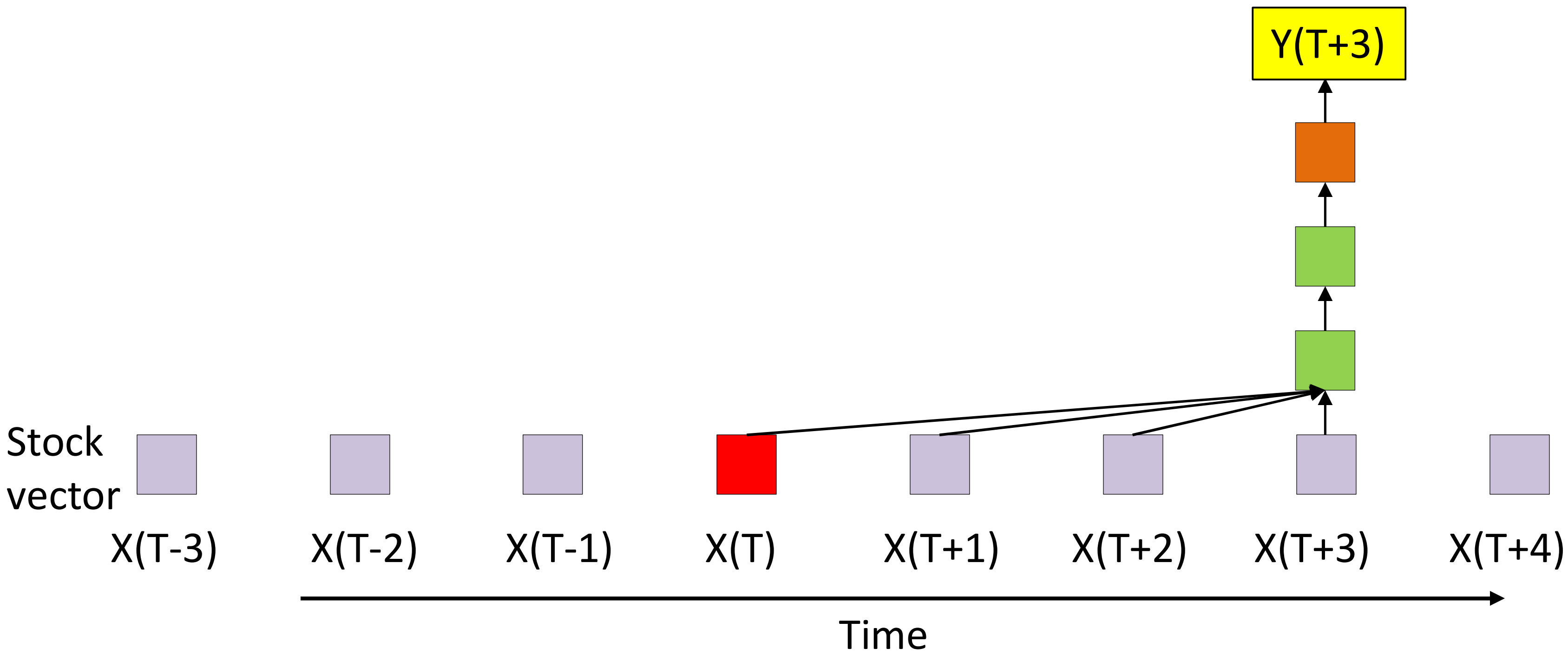
The stock predictor



- This is a *finite response* system
  - Something that happens *today* only affects the output of the system for N days into the future
    - N is the *width* of the system

$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-N})$$

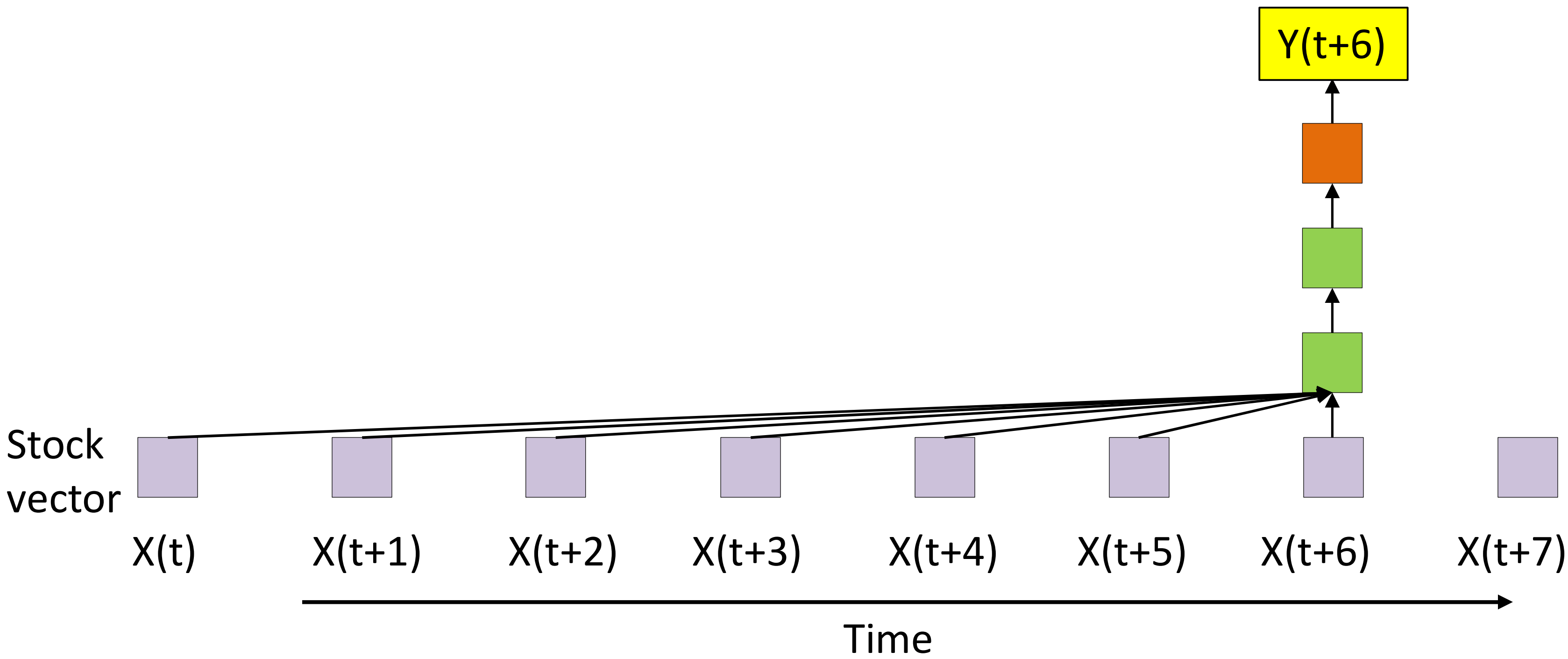
Finite-response model



- Something that happens *today* only affects the output of the system for N days into the future
  - **Predictions consider *N* days of history**

- To consider more of the past to make predictions, you must increase the "history" considered by the system

Finite-response

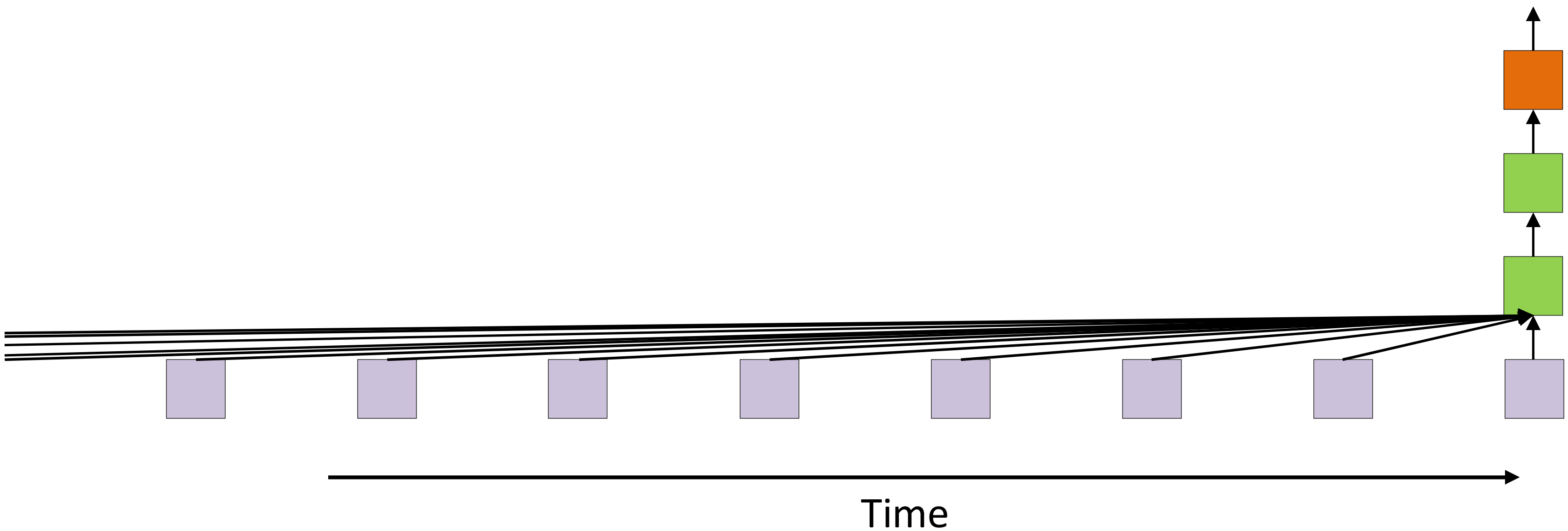


- Problem: Increasing the "history" makes the network more complex
  - No worries, we have the CPU and memory
    - Or do we?

Systems often have long-term dependencies

- Longer-term trends –
  - Weekly trends in the market
  - Monthly trends in the market
  - Annual trends
  - Though longer historic tends to affect us less than more recent events..

We want *infinite* memory



- Required: *Infinite* response systems
  - What happens today can continue to affect the output forever
    - Possibly with weaker and weaker influence

$$Y_t = f(X_t, X_{t-1}, \ldots, X_{t-\infty})$$

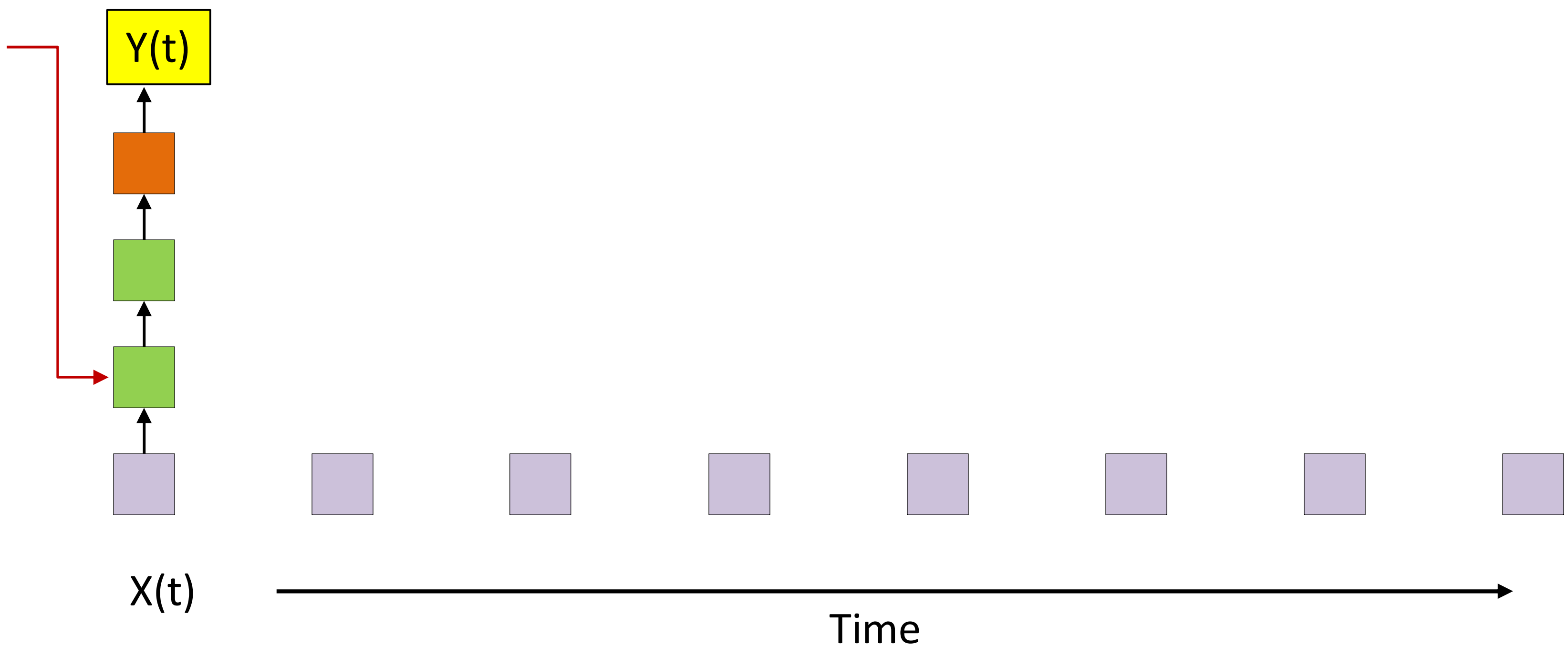
Examples of infinite response systems

$$Y_t = f(X_t, Y_{t-1})$$

- Required: Define initial state: Y_{-1} for $t = 0$
- An input at X_0 at $t = 0$ produces Y_0
- Y_0 produces Y_1 which produces Y_2 and so on until Y_∞ *even if* $X_1 \ldots X_\infty$ *are 0*
  - i.e. even if there are no further inputs!
- **A single input influences the output for the rest of time!**

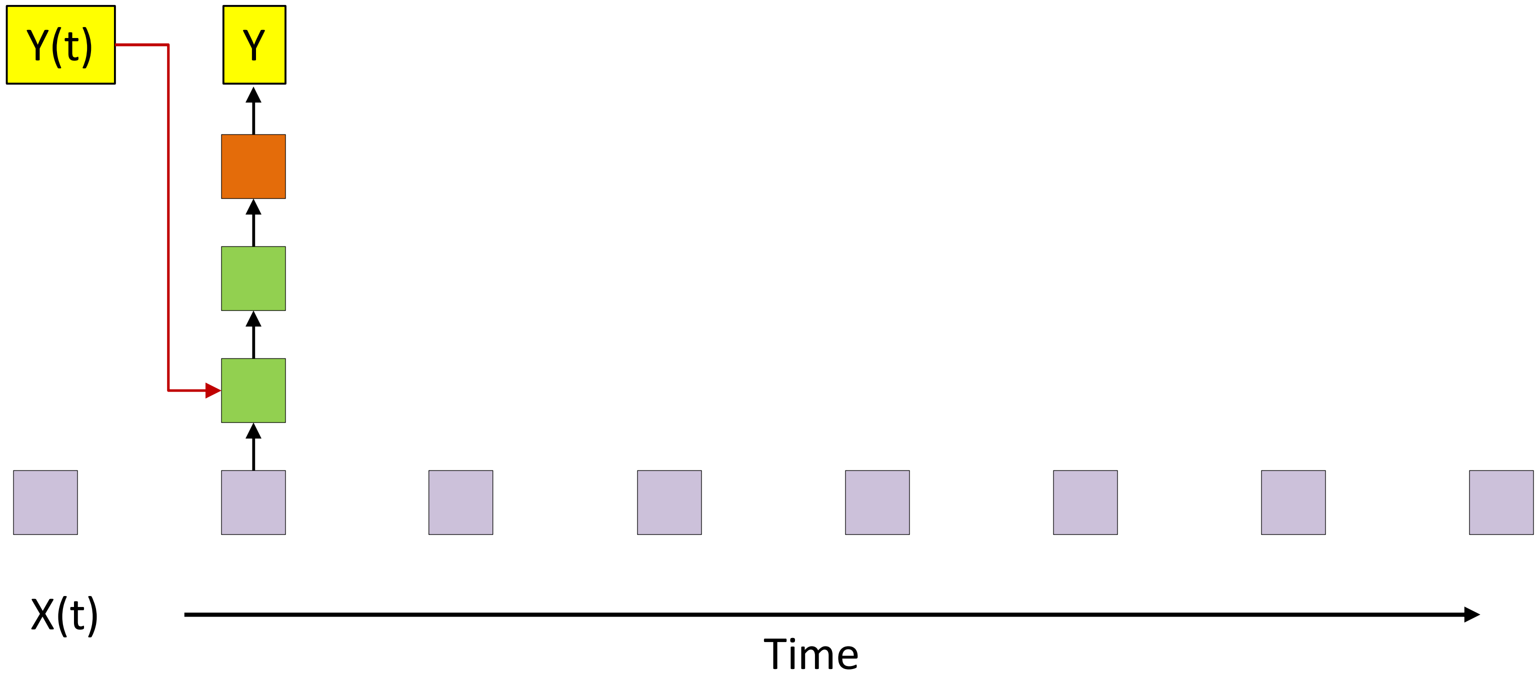
- This is an instance of a NARX network
  - "nonlinear autoregressive network with exogenous inputs"
  - $Y_t = f(X_{0:t}, Y_{0:t-1})$
- *Output* contains information about the entire past

A one-tap NARX network



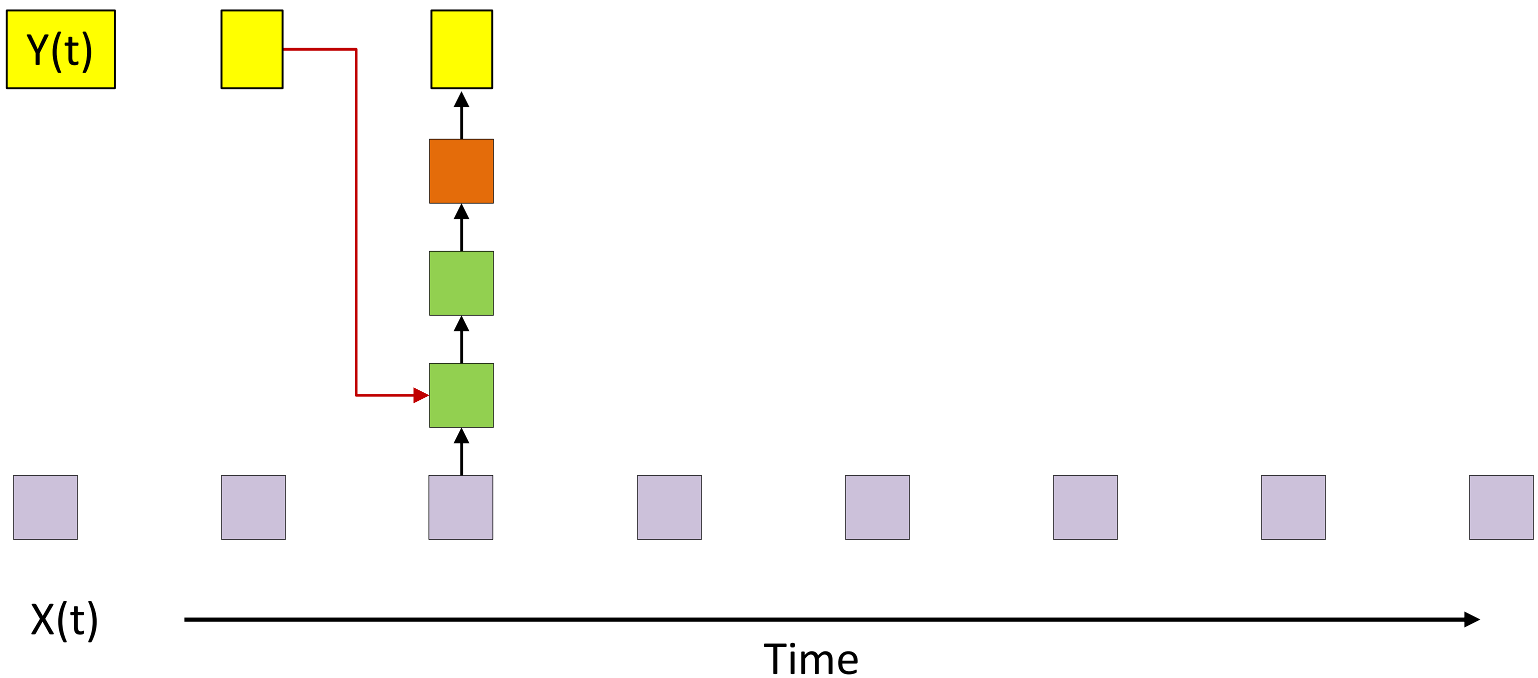
- A NARX net with recursion from the output

A one-tap NARX network



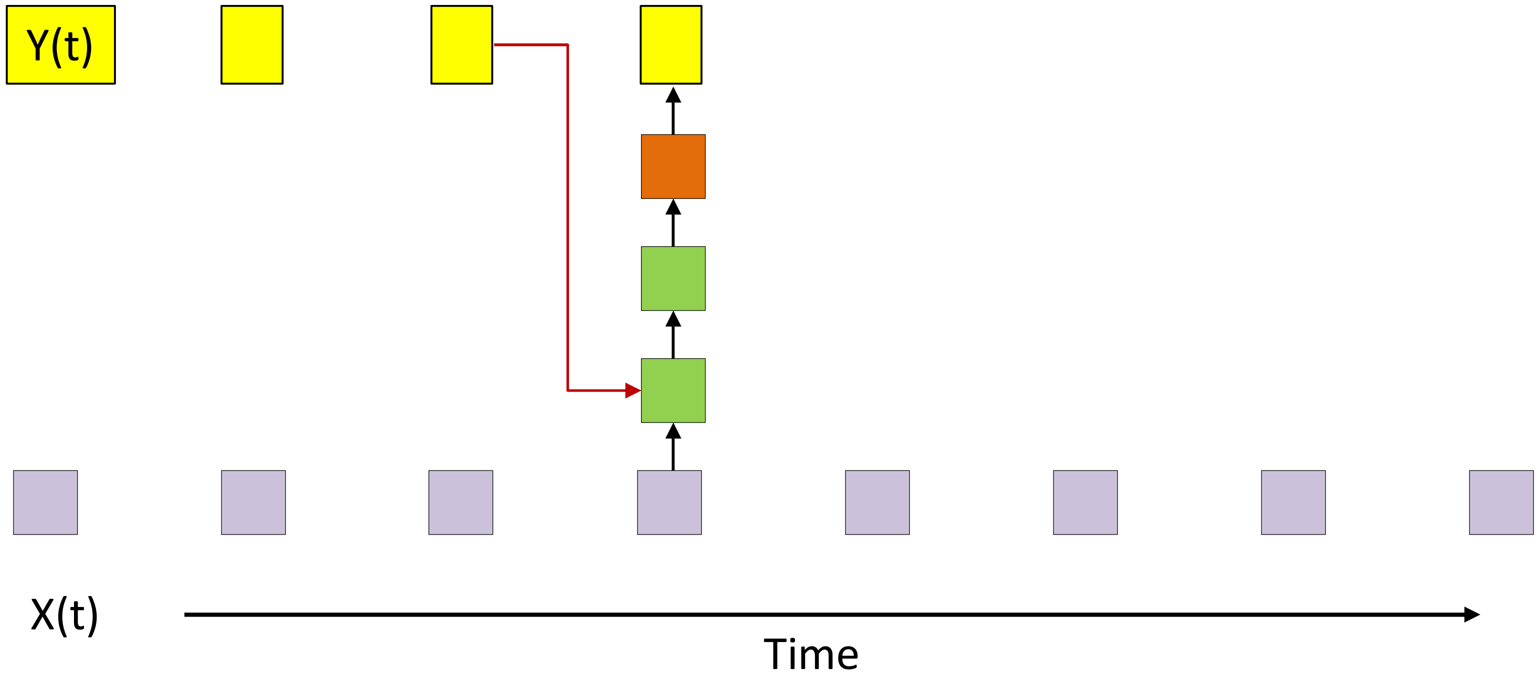
- A NARX net with recursion from the output

A one-tap NARX network



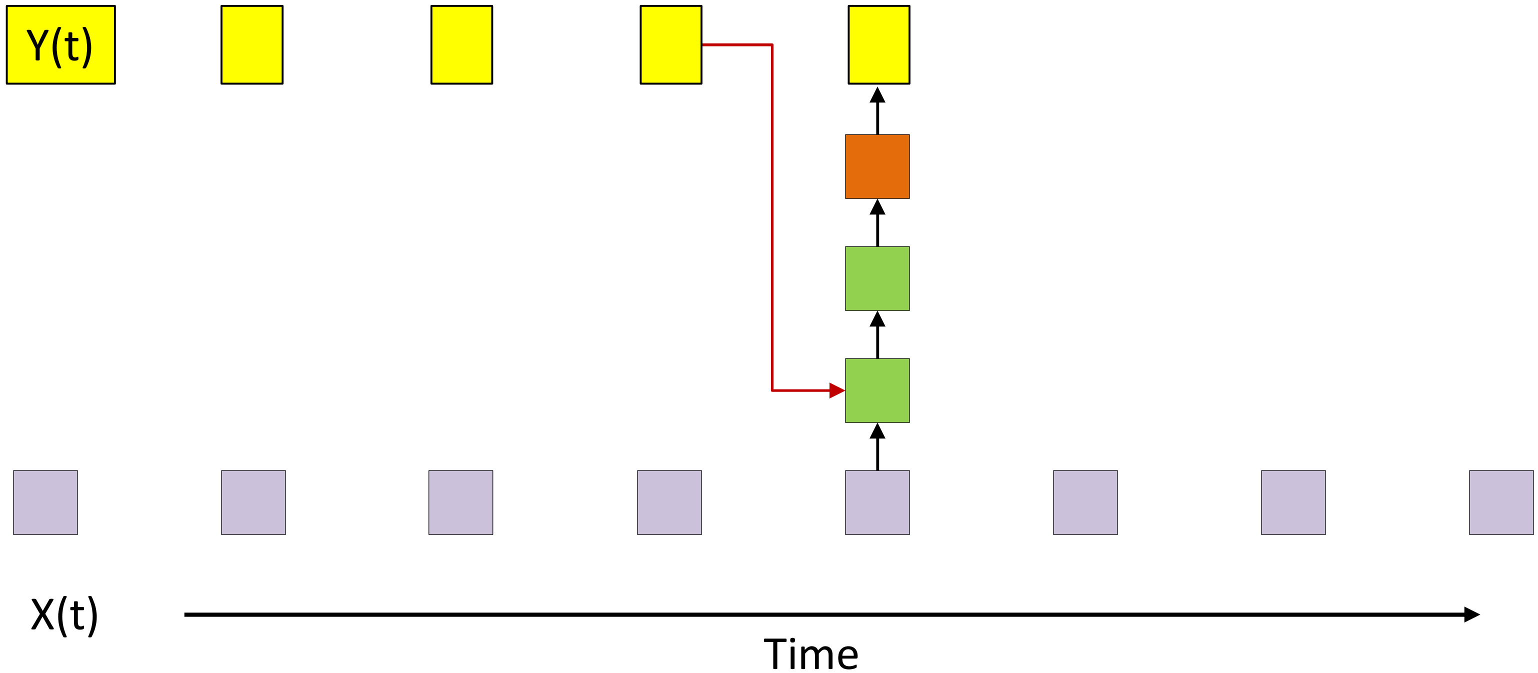
- A NARX net with recursion from the output

A one-tap NARX network



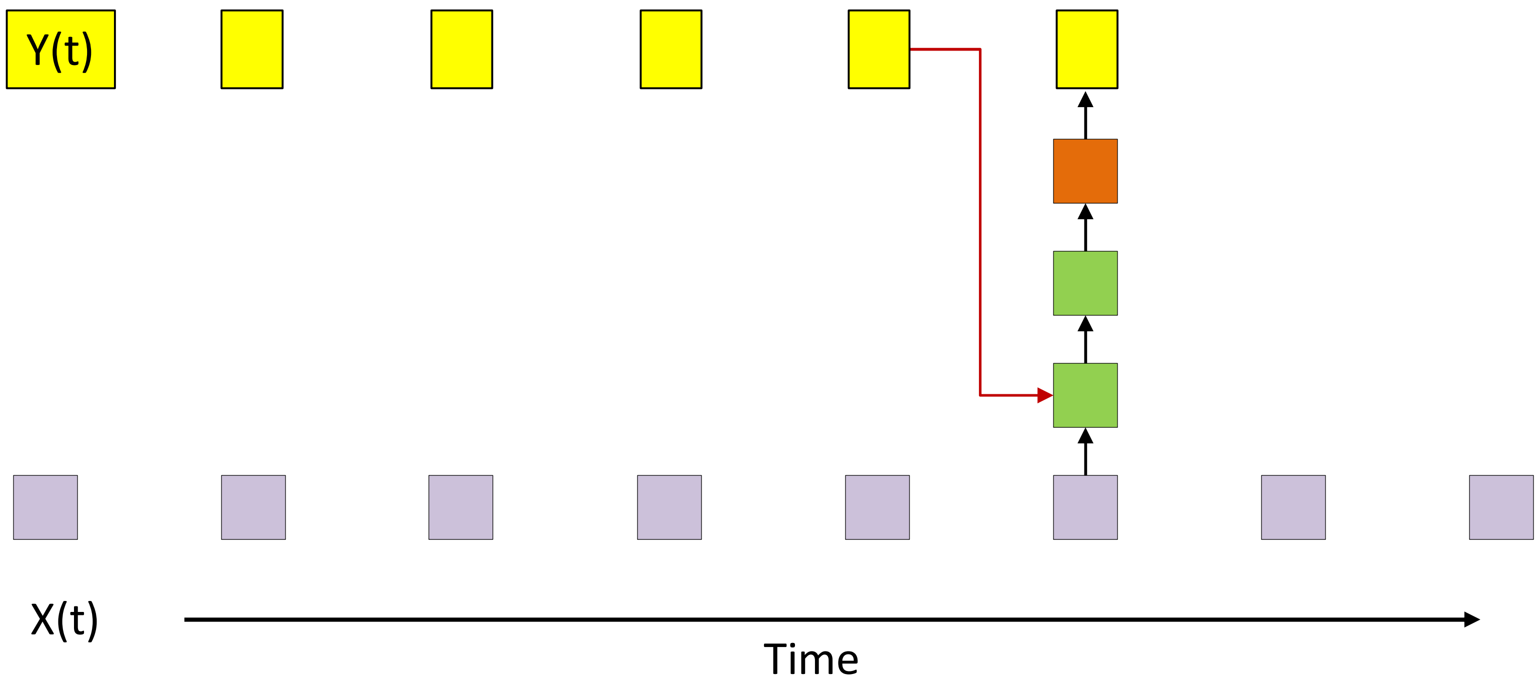
- A NARX net with recursion from the output

A one-tap NARX network



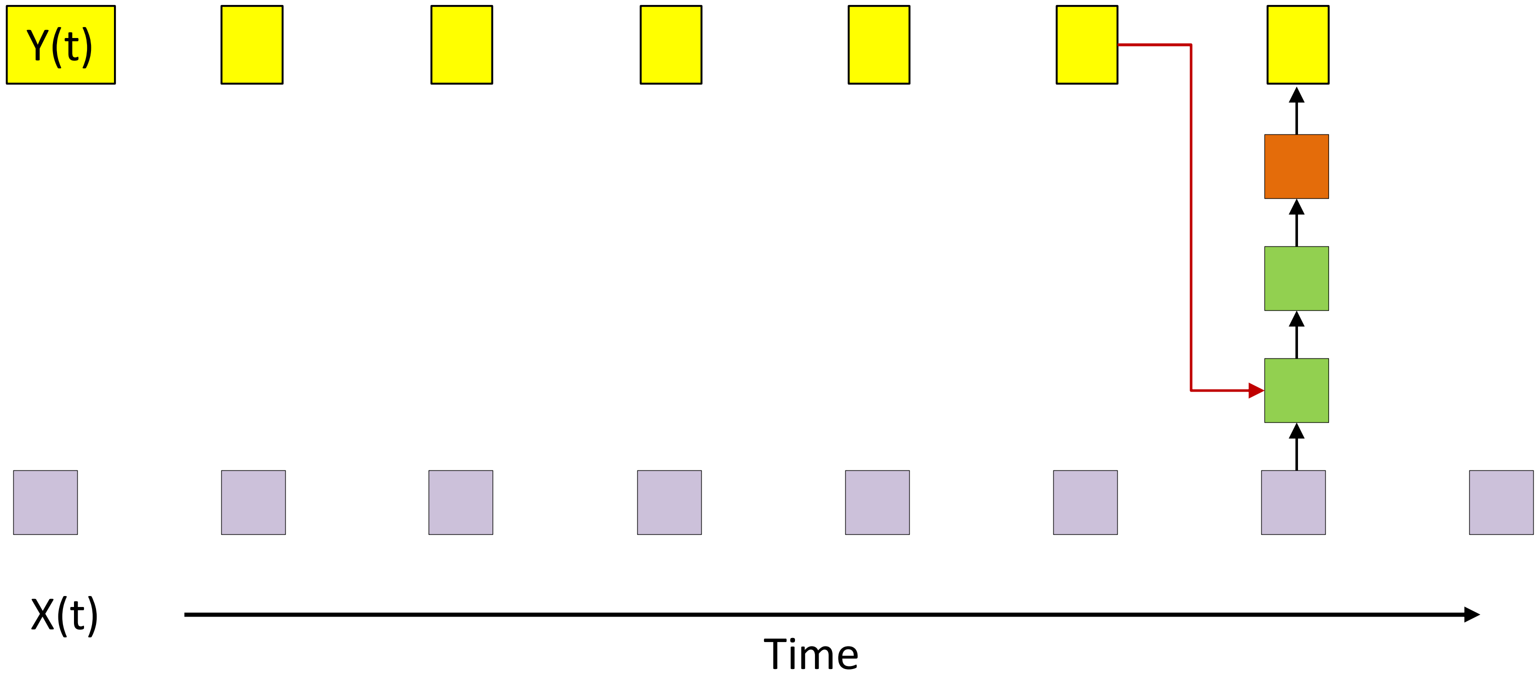
- A NARX net with recursion from the output

A one-tap NARX network



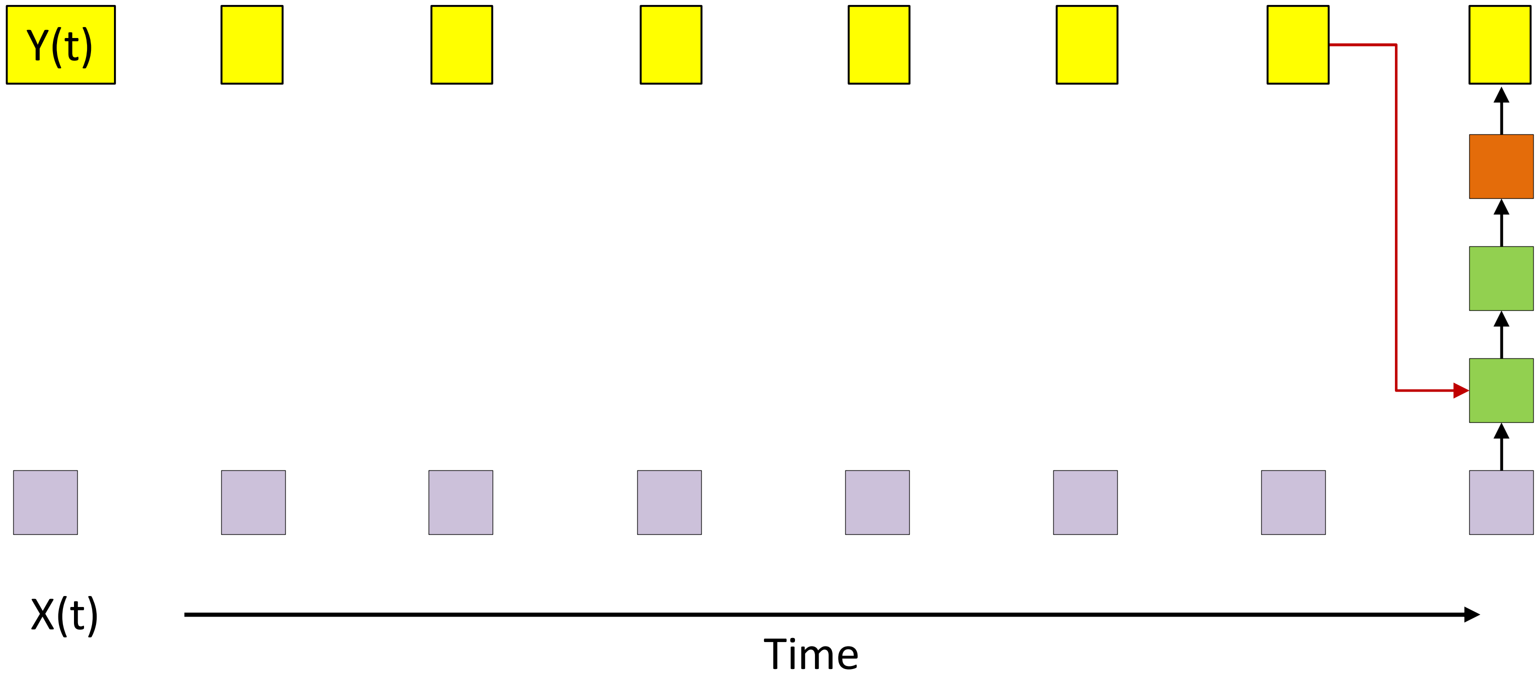
- A NARX net with recursion from the output

A one-tap NARX network



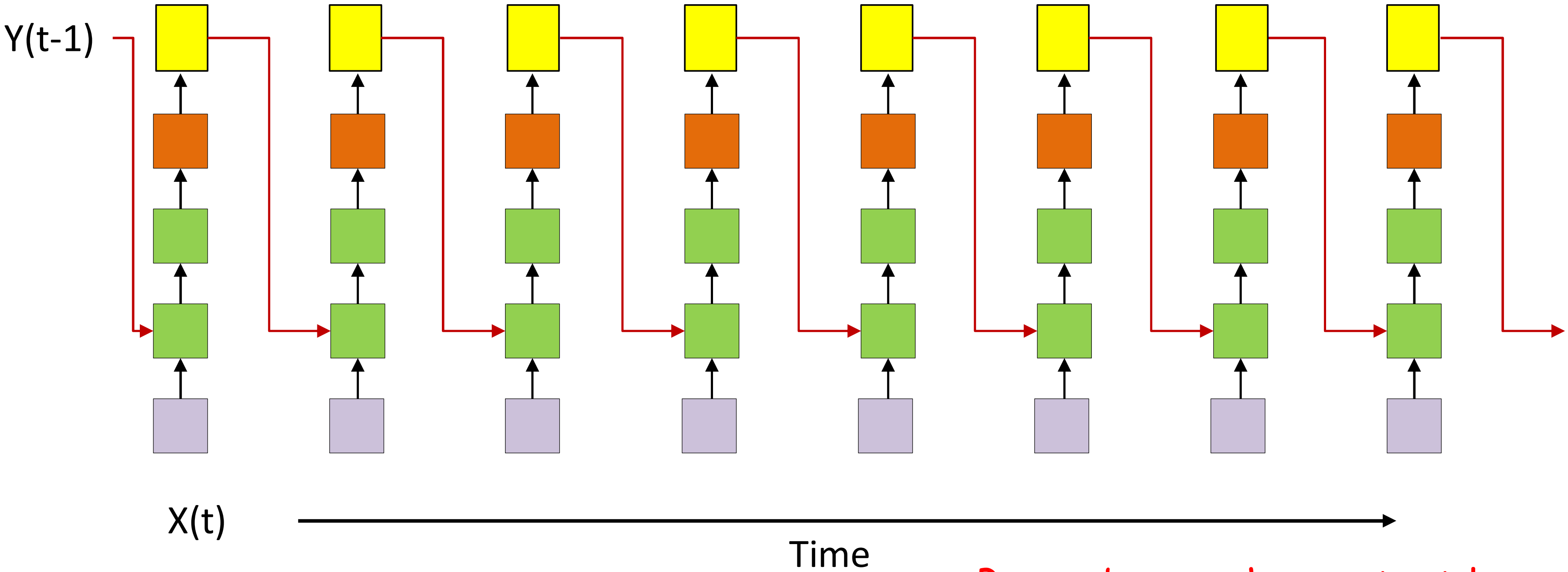
- A NARX net with recursion from the output

A one-tap NARX network



- A NARX net with recursion from the output

A more complete representation



Brown boxes show output layers
Yellow boxes are outputs

- A NARX net with recursion from the output
- Showing all computations
- All columns are identical
- *An input at t=0 affects outputs forever*

NARX Networks

- Very popular for time-series prediction
  - Weather
  - Stock markets
  - As alternate system models in tracking systems
- Any phenomena with distinct “innovations” that “drive” an output

- Note: here the “memory” of the past is in the output itself, and not in the network

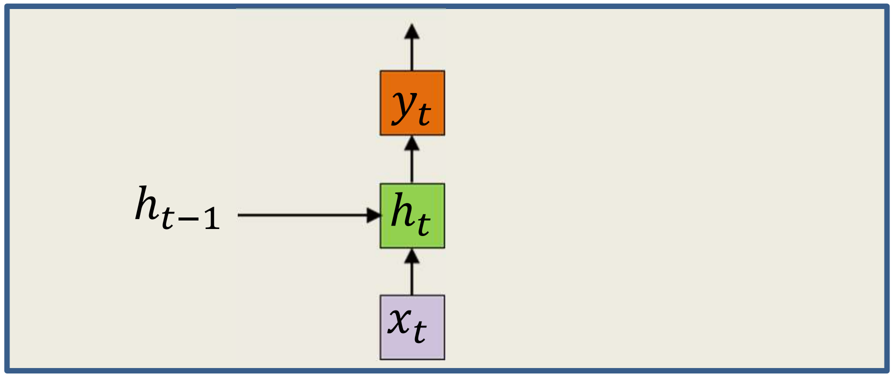
An alternate model for infinite response systems: the state-space model

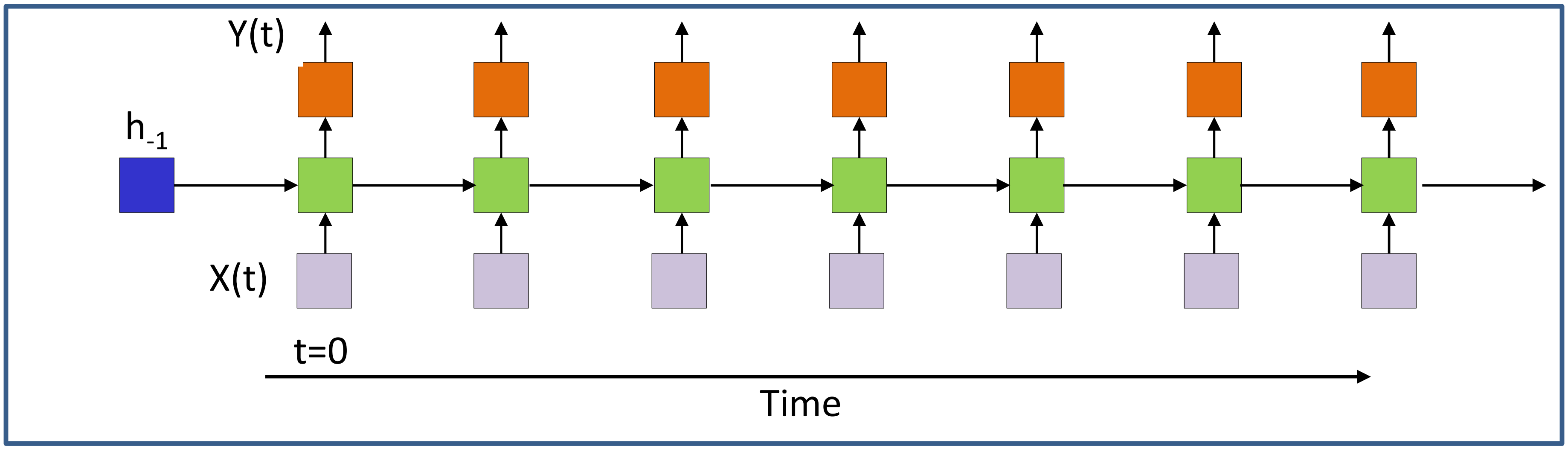
$$h_t = f(x_t, h_{t-1})$$
$$y_t = g(h_t)$$

- h_t is the *state* of the network
  - *State* summarizes information about the entire past
    - Model directly embeds the memory in the state
- Need to define initial state h_{-1}

- This is a *fully recurrent* neural network
  - Or simply a *recurrent neural network*

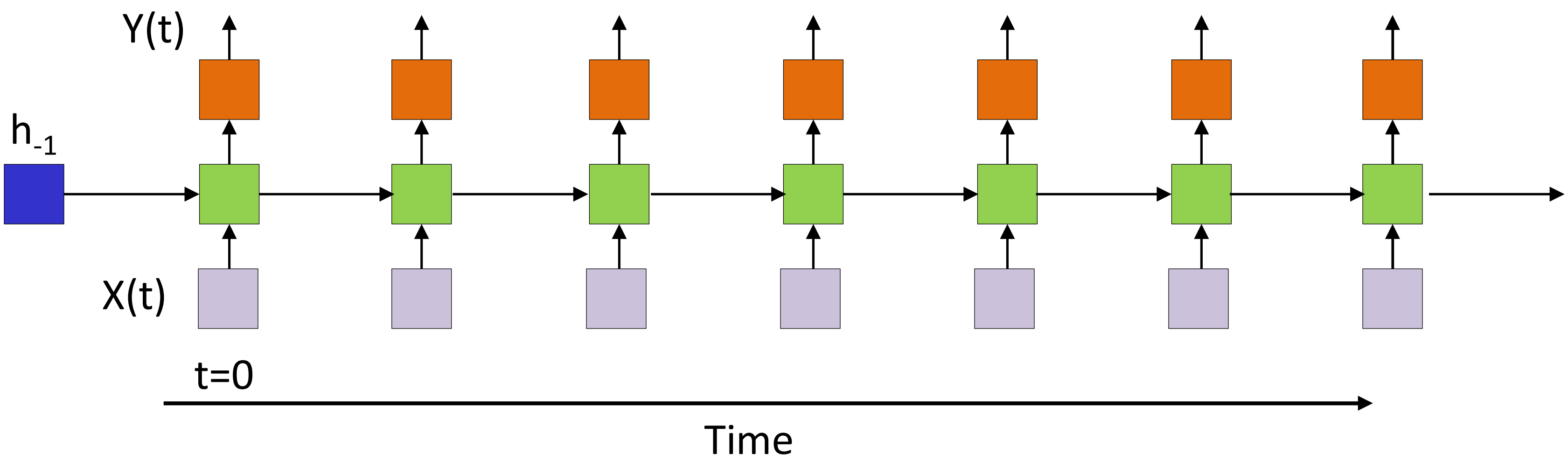
The simple state-space model





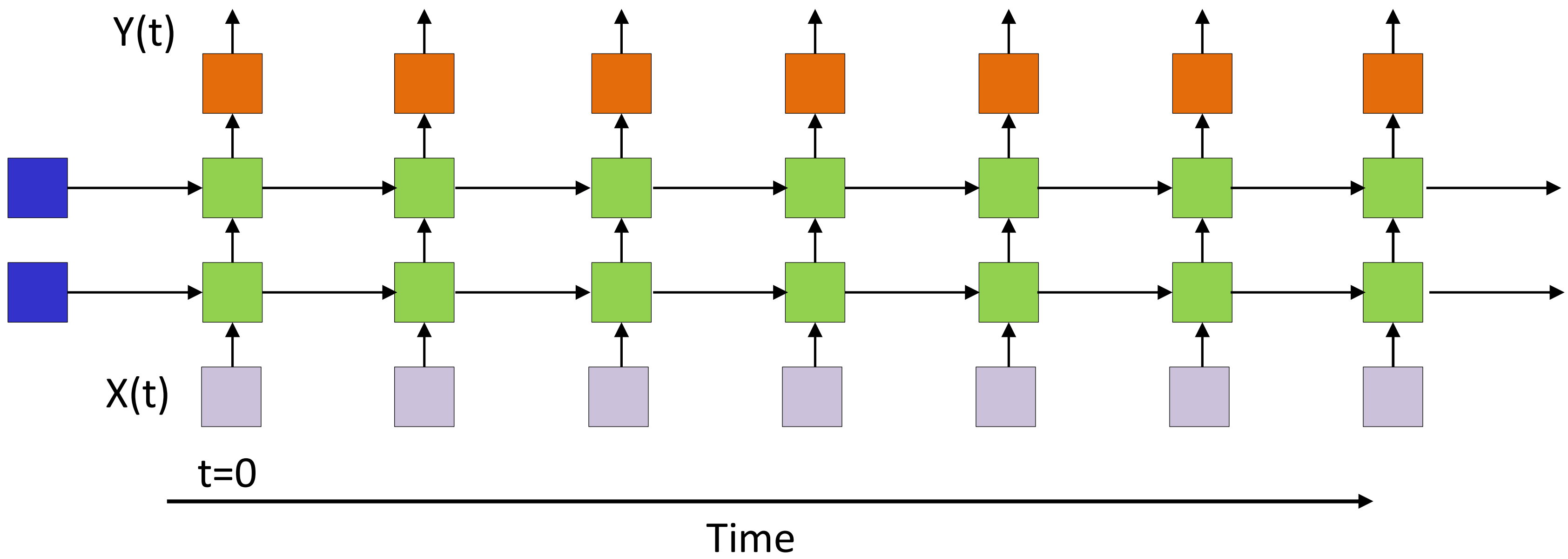
- The state (green) at any time is determined by the input at that time, and the state at the previous time
- *An input at t=0 affects outputs forever*
- Also known as a recurrent neural net

Single hidden layer RNN



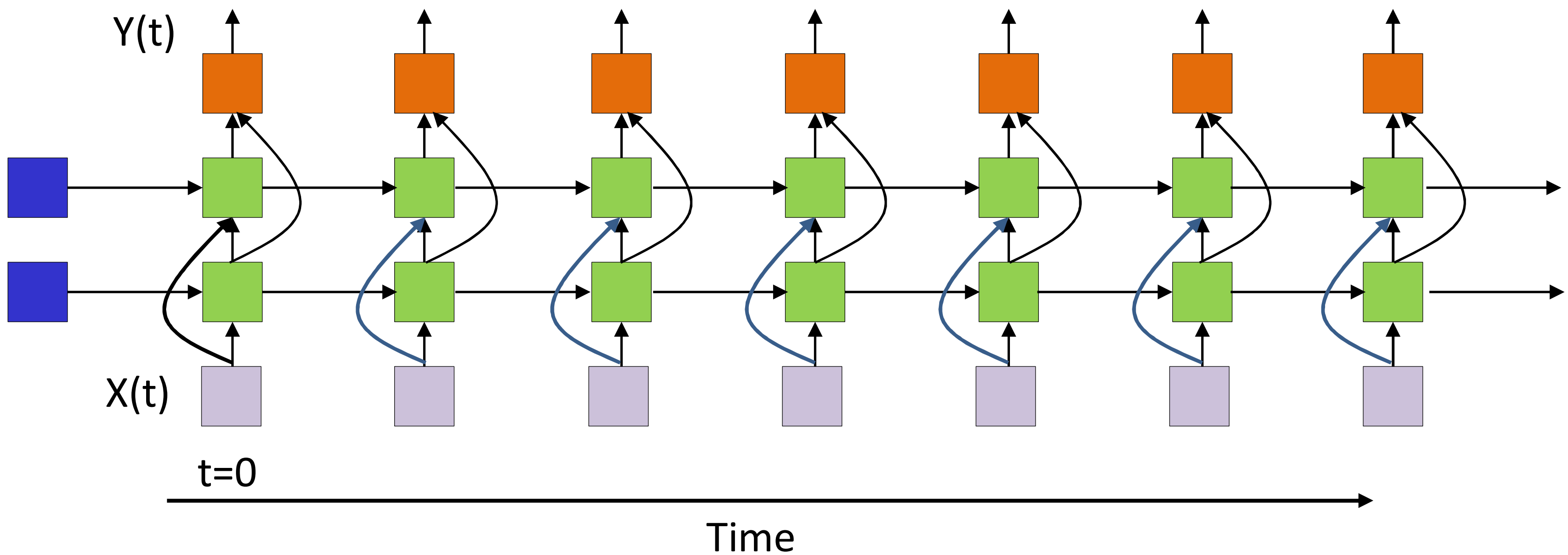
- Recurrent neural network
- All columns are identical
- *An input at t=0 affects outputs forever*

Multiple recurrent layer RNN



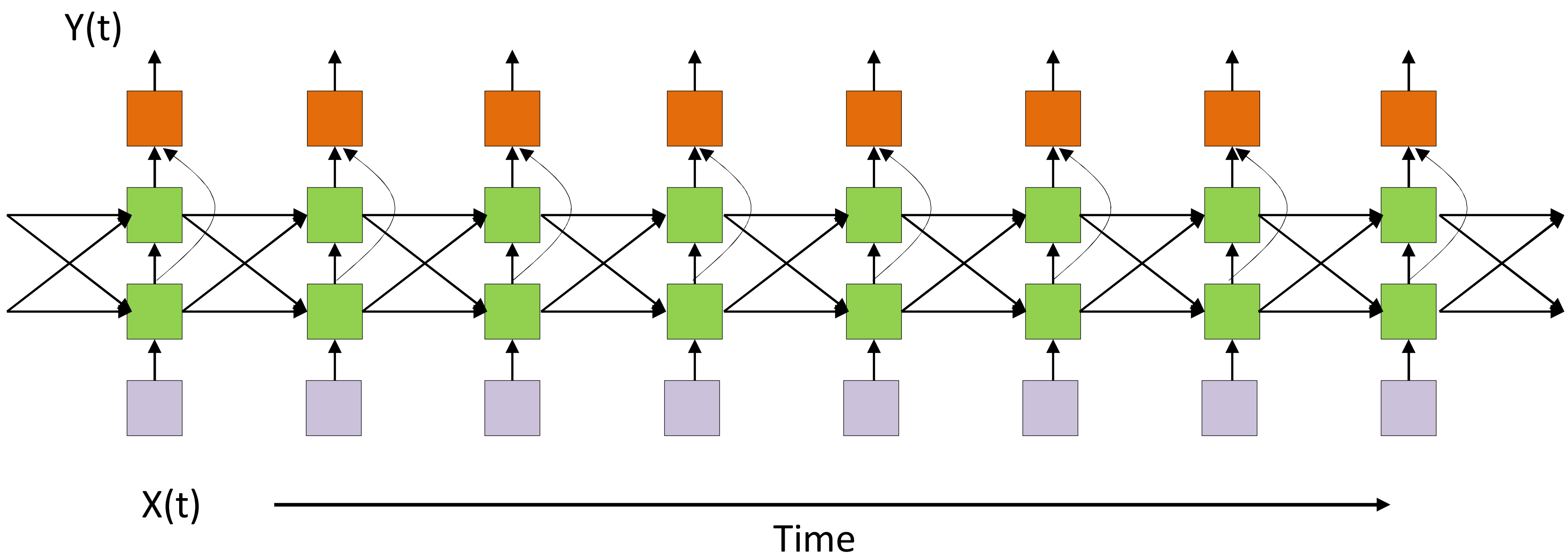
- Recurrent neural network
- All columns are identical
- *An input at t=0 affects outputs forever*

Multiple recurrent layer RNN



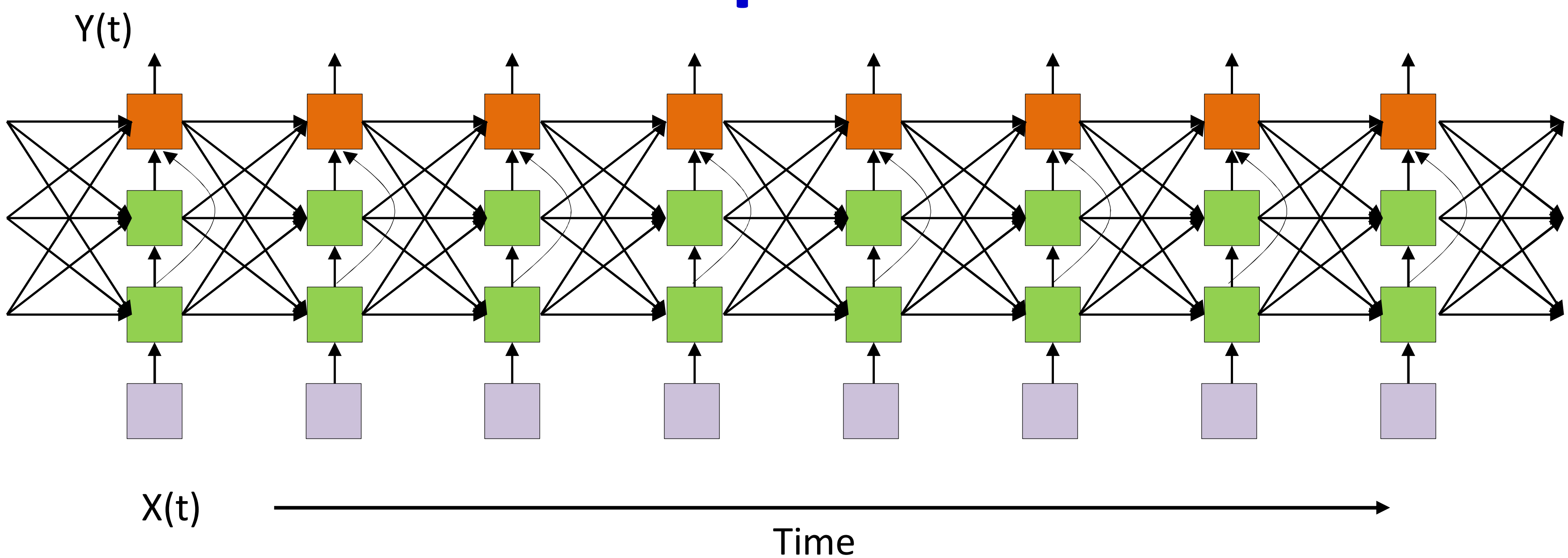
- We can also have skips..

A more complex state



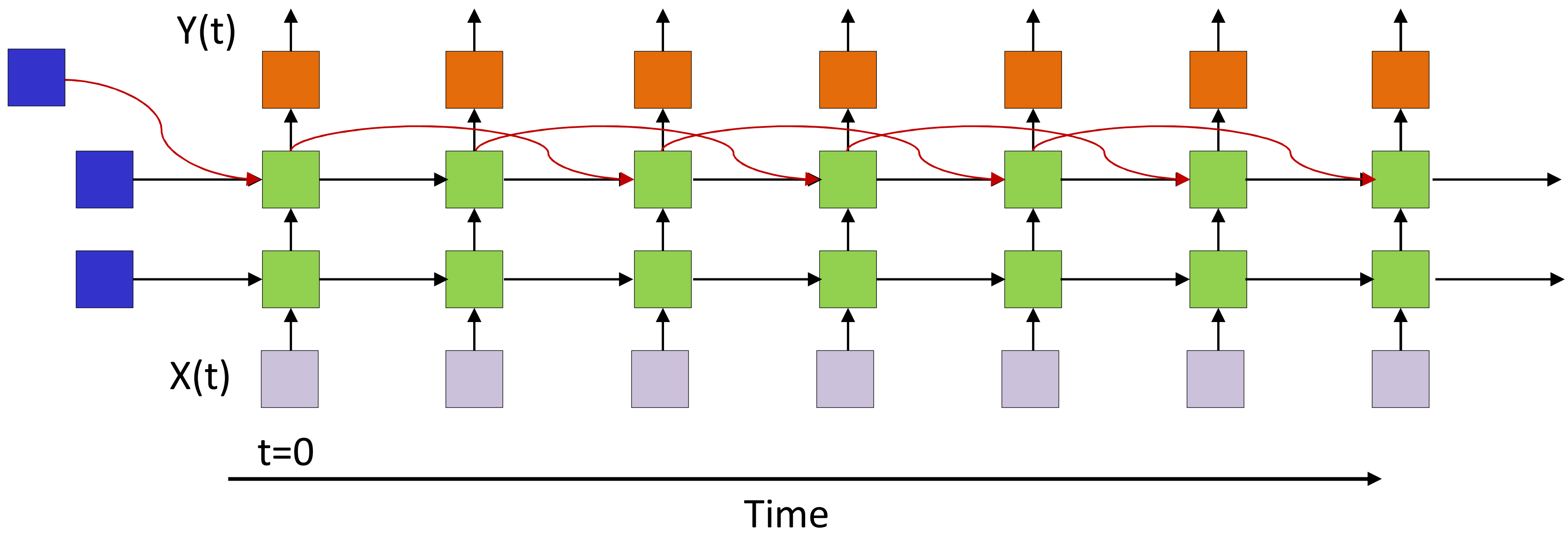
- All columns are identical
- *An input at t=0 affects outputs forever*

Or the network may be even more complicated



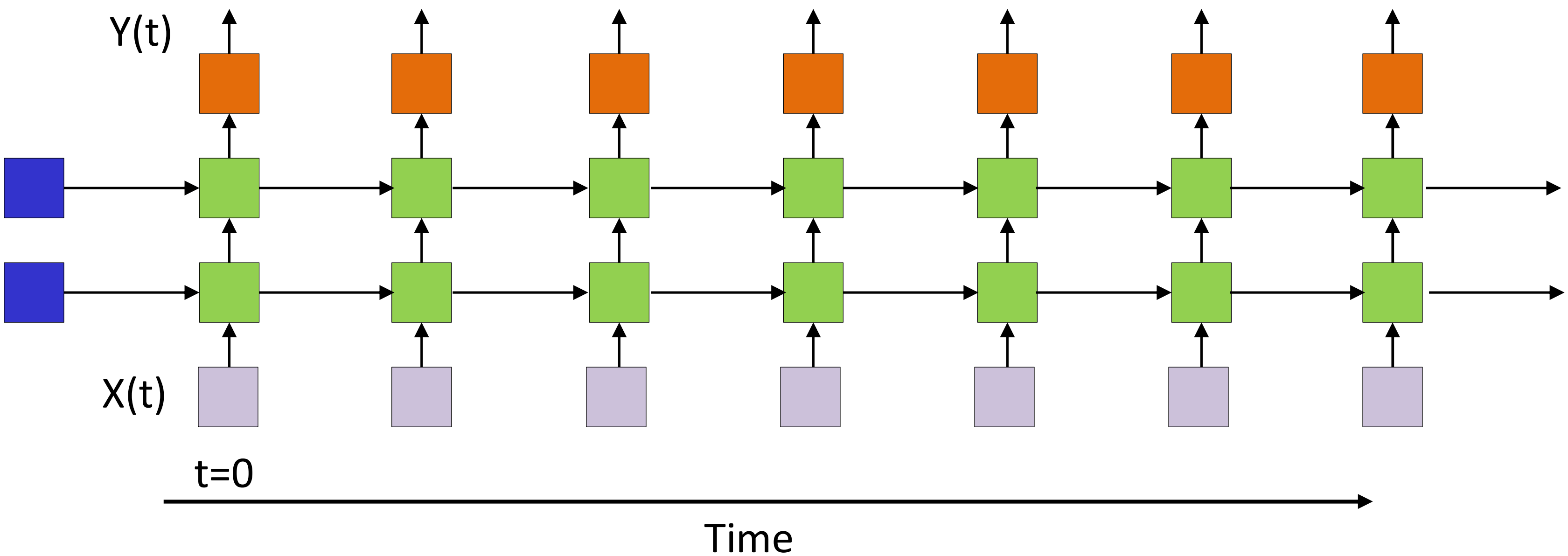
- Shades of NARX
- All columns are identical
- *An input at t=0 affects outputs forever*

Generalization with other recurrences



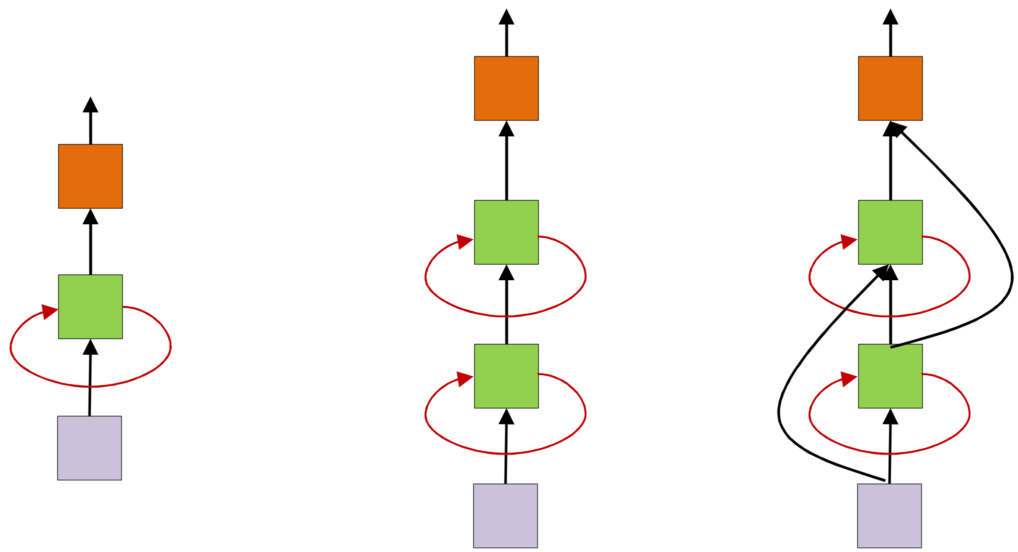
- All columns (including incoming edges) are identical

The simplest structures are most popular



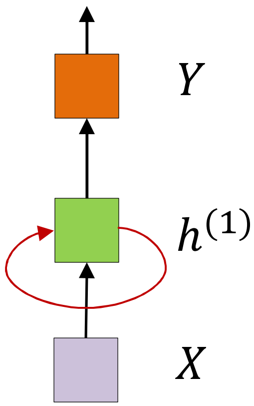
- Recurrent neural network
- All columns are identical
- *An input at t=0 affects outputs forever*

A Recurrent Neural Network



- Simplified models often drawn
- The loops imply recurrence

Equations



Current weights

Recurrent weights

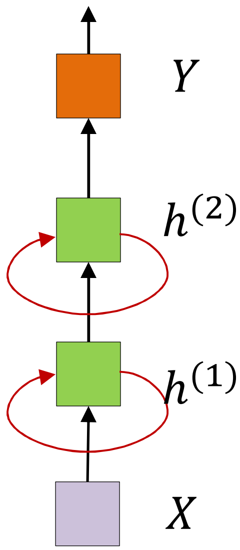
$$h_i^{(1)}(-1) = part\ of\ network\ parameters$$

$$h_i^{(1)}(t) = f_1\left(\sum_j w_{ji}^{(1)} X_j(t) + \sum_j w_{ji}^{(11)} h_i^{(1)}(t-1) + b_i^{(1)}\right)$$

$$Y(t) = f_2\left(\sum_j w_{jk}^{(2)} h_j^{(1)}(t) + b_k^{(2)}, k = 1..M\right)$$

- Note superscript in indexing, which indicates layer of network from which inputs are obtained
- Assuming vector function at output, e.g. softmax
- The *state* node activation, $f_1()$ is typically $\tanh()$
- Every neuron also has a *bias* input

Equations



$$h_i^{(1)}(-1) = part\ of\ network\ parameters$$

$$h_i^{(2)}(-1) = part\ of\ network\ parameters$$

$$h_i^{(1)}(t) = f_1\left(\sum_j w_{ji}^{(1)} X_j(t) + \sum_j w_{ji}^{(11)} h_i^{(1)}(t-1) + b_i^{(1)}\right)$$

$$h_i^{(2)}(t) = f_2\left(\sum_j w_{ji}^{(2)} h_j^{(1)}(t) + \sum_j w_{ji}^{(22)} h_i^{(2)}(t-1) + b_i^{(2)}\right)$$

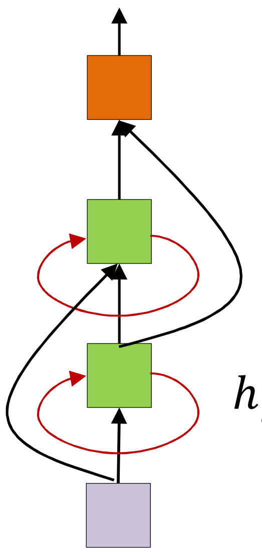
$$Y(t) = f_3\left(\sum_j w_{jk}^{(3)} h_j^{(2)}(t) + b_k^{(3)}, k = 1..M\right)$$

- Assuming vector function at output, e.g. softmax $f_3()$
- The *state* node activations, $f_k()$ are typically tanh()
- Every neuron also has a *bias* input

Equations

$$h_i^{(1)}(-1) = part\ of\ network\ parameters$$

$$h_i^{(2)}(-1) = part\ of\ network\ parameters$$

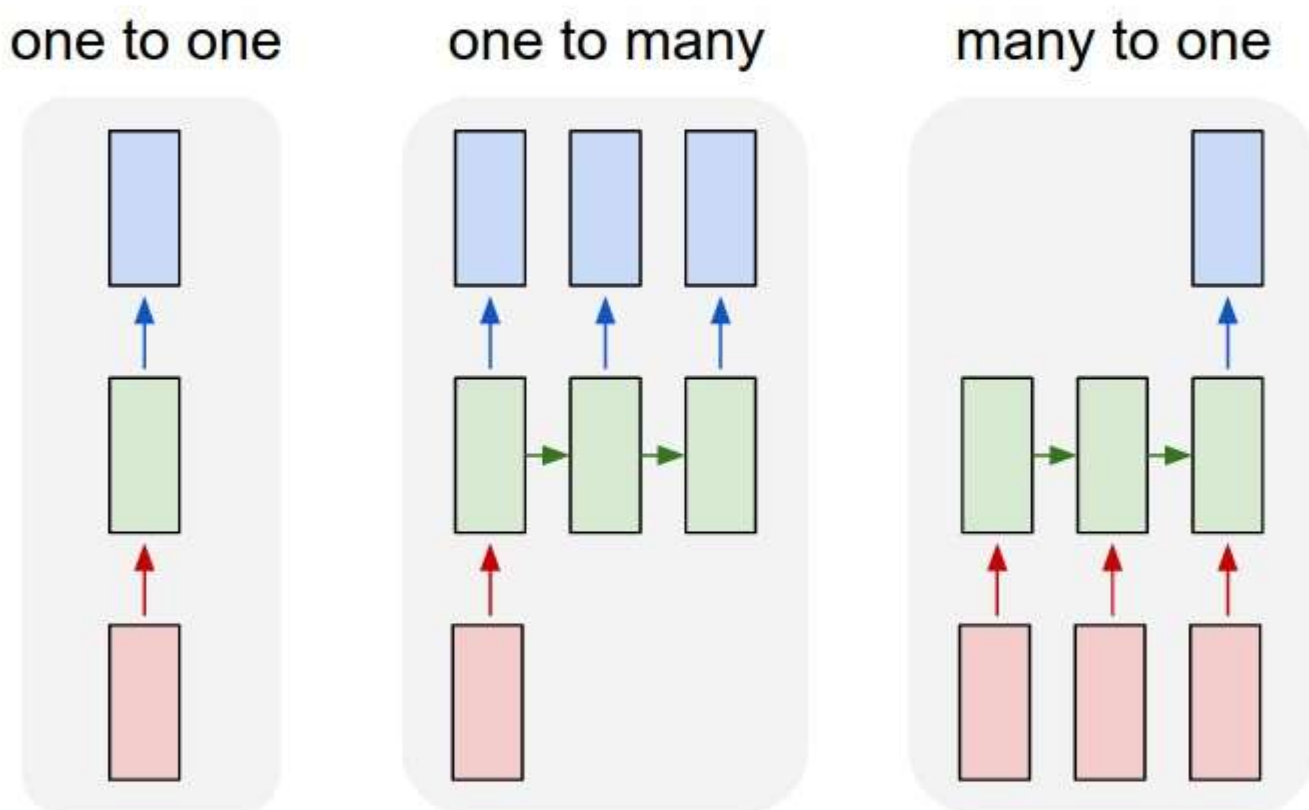


$$h_i^{(1)}(t) = f_1\left(\sum_j w_{ji}^{(0,1)} X_j(t) + \sum_i w_{ii}^{(1,1)} h_i^{(1)}(t-1) + b_i^{(1)}\right)$$

$$h_i^{(2)}(t) = f_2\left(\sum_j w_{ji}^{(1,2)} h_j^{(1)}(t) + \sum_j w_{ji}^{(0,2)} X_j(t) + \sum_i w_{ii}^{(2,2)} h_i^{(2)}(t-1) + b_i^{(2)}\right)$$

$$Y_i(t) = f_3\left(\sum_j w_{jk}^{(2)} h_j^{(2)}(t) + \sum_j w_{jk}^{(1,3)} h_j^{(1)}(t) + b_k^{(3)}, k = 1..M\right)$$

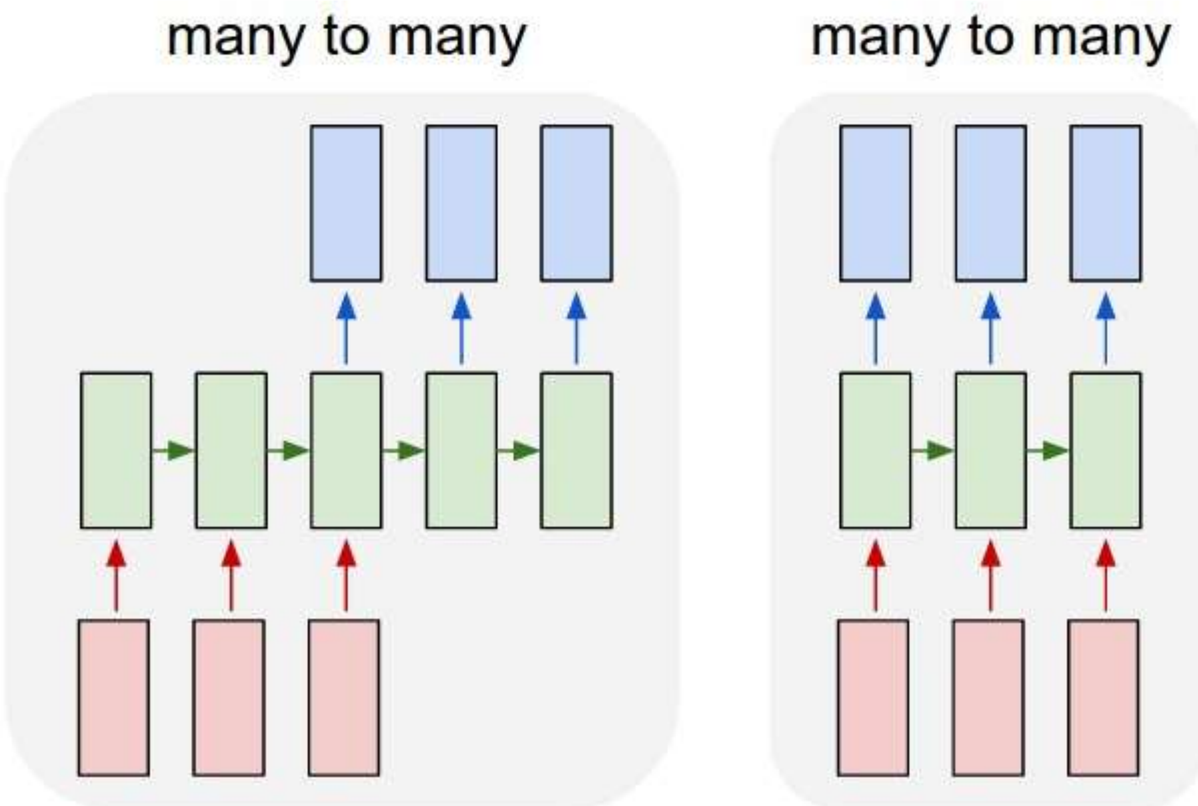
Variants on recurrent nets



Images from Karpathy

- 1: Conventional MLP
- 2: Sequence *generation*, e.g. image to caption
- 3: Sequence based *prediction or classification,* e.g. Speech recognition, text classification

Variants



Images from Karpathy

- 1: *Delayed* sequence to sequence, e.g. machine translation
- 2: Sequence to sequence, e.g. stock problem, label prediction
- Etc…