More on

Time and Neural Networks

Thus far

- Except for the Adaptive Filter, networks discussed so far have been “combinational”; input pattern presented at once.

- Now we wish to consider additional models where more general network inputs and learned behavior can include **functions of time.**

Models to be Considered Here

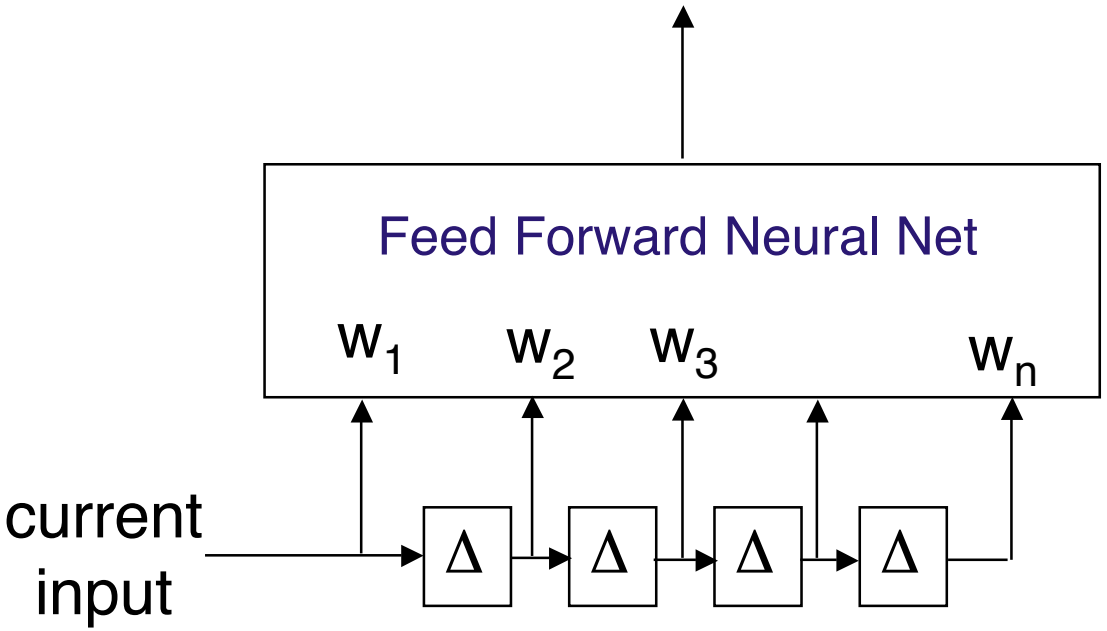
- Time-Lagged Feed-Forward Networks, Time-Delay Neural Networks (TLFF, TDNN)
- Elman nets, Jordan nets
- FIR-Multi-layer networks (FIRNET)
- Backpropagation through time (BPTT)
- Real-Time Recurrent Learning (RTRL)
- Temporal difference method (TD(λ))

Time-Lagged Feed-Forward Networks (TLFF)

- Extends the “Adaline” adaptive filter model.
- Use an arbitrary feed-forward net (MLP) in place of the Adaline.
- Train using ordinary backpropagation.

Time-Lagged Feed Forward

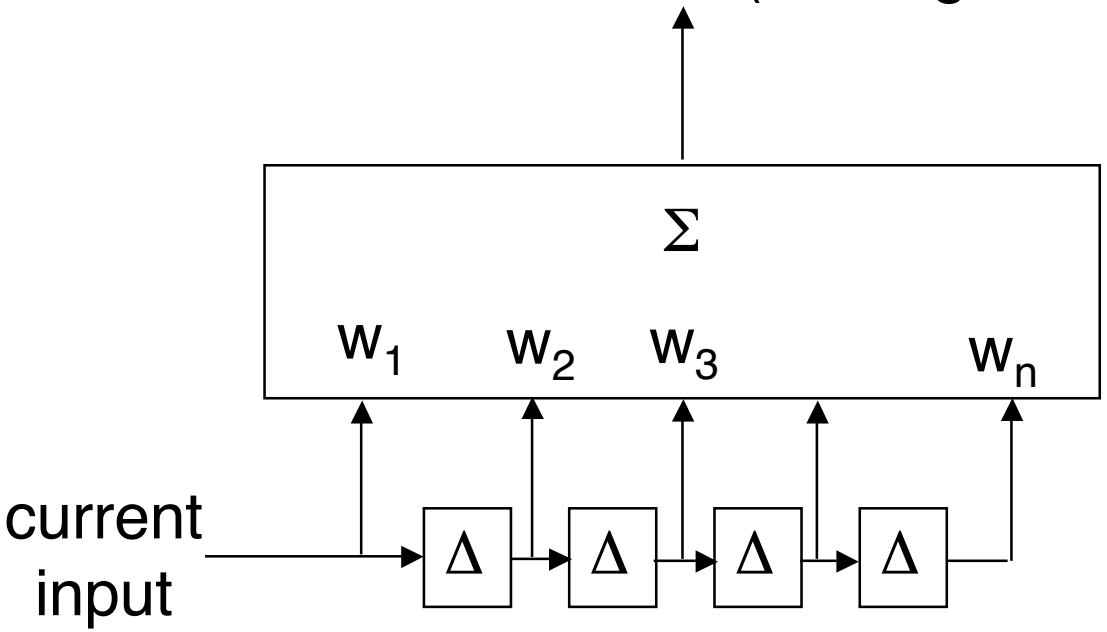
Use a fixed number of the most recent inputs in a sequence as inputs to a neural network. Train the network using the input samples with desired values at each time step.



Classical Nomenclature

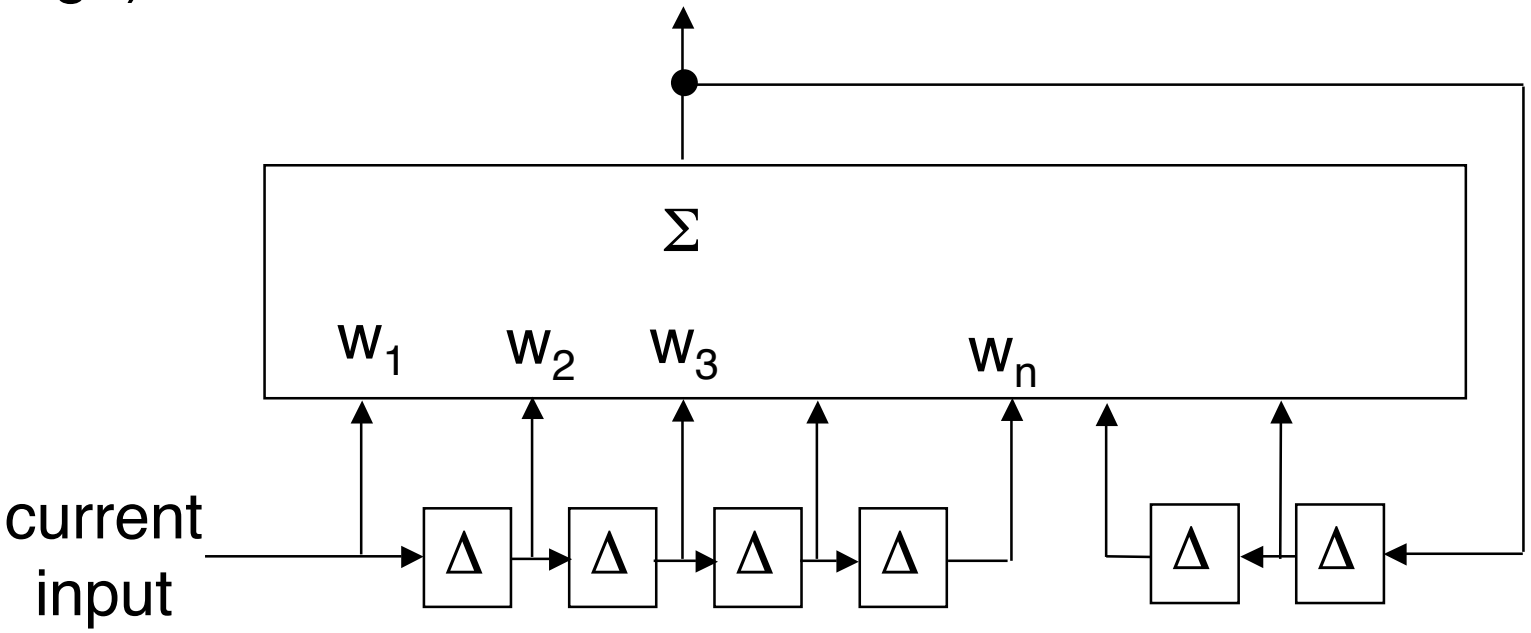
In engineering, a basic filter form such as the adaptive filter is called a **FIR** (**F**inite **I**mpluse **R**esponse) filter.

In statistics, it is called an **MA** (**M**oving **A**verage) filter.



Classical Nomenclature

If we add **feedback**, we have an **IIR** (**I**nfinite **I**mpluse **R**esponse) filter.
In statistics, it is called an AR**MA** (**A**uto**R**egressive **M**oving **A**verage) filter.

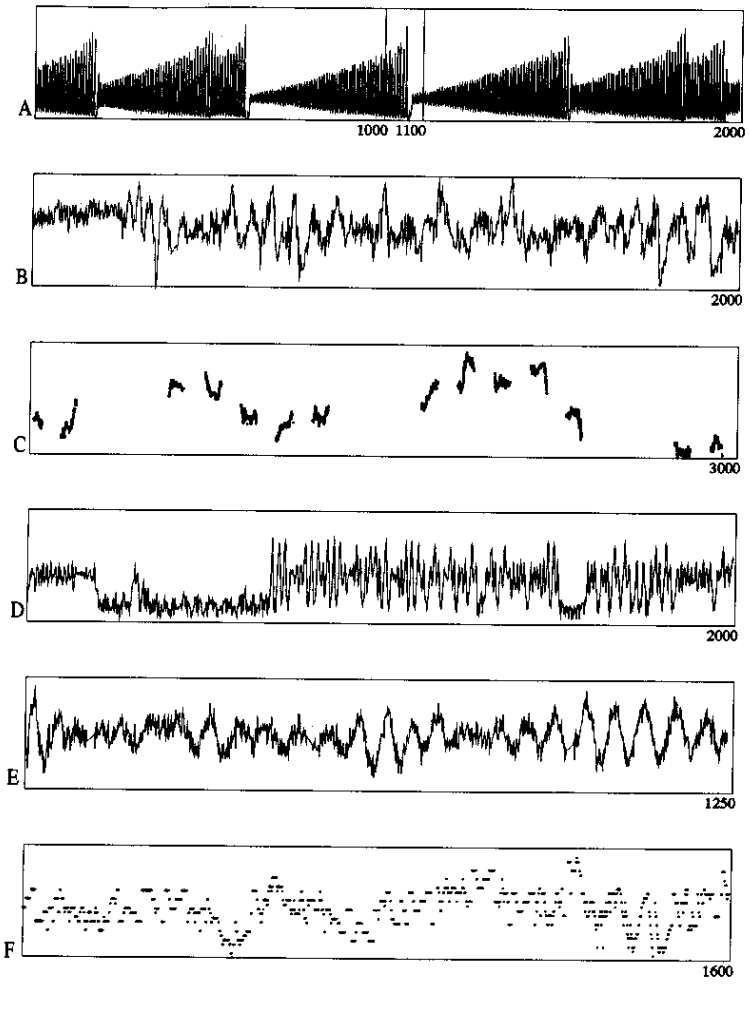


Classical Fitting of Time-Series

- A given ARMA's coefficients can be fit to generate an ***approximation*** (not necessarily very good) a given time series by using least-squares estimation on a set of simultaneous linear equations known as the "Yule-Walker equations".

- Reference: Weigend and Gershenfeld (eds.), Time series prediction, Addison-Wesley, 1994.

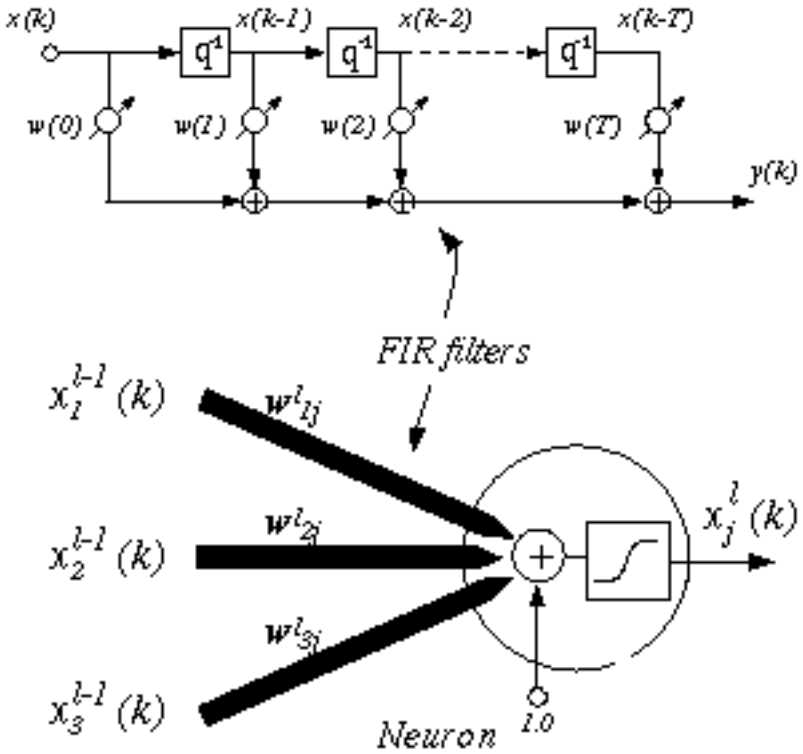
Sante Fe Institute Time-Series Prediction Competition
Example of 6 unknown time series

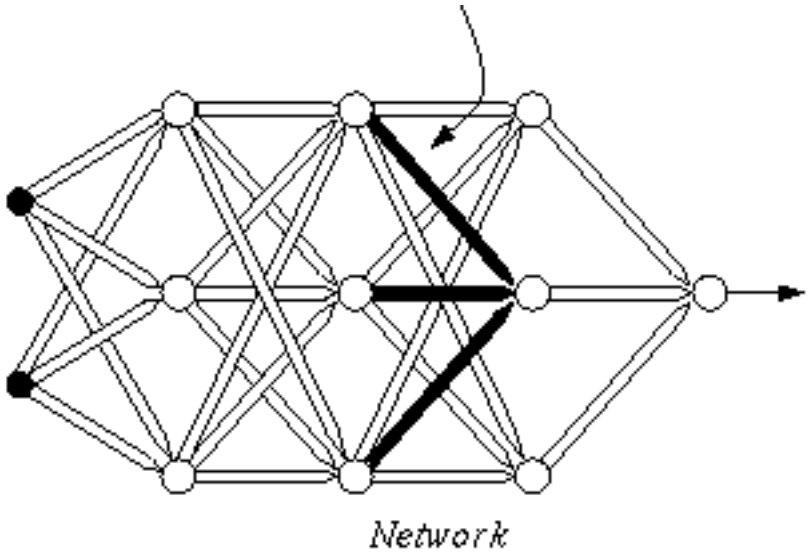


A Winning Approach: FIR Backpropagation

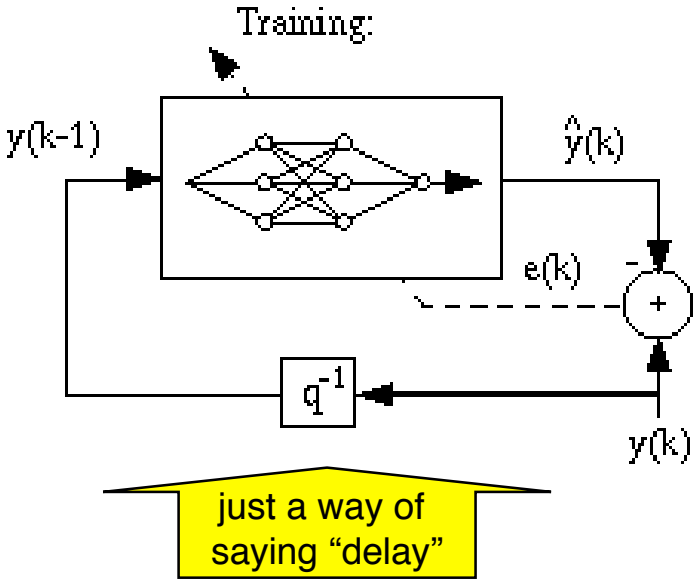
- Eric Wan (Stanford, OGI) came up with the idea of putting FIR filters **inside** a backprop network.
- In place of each single weight there is an entire FIR filter.
- Wan developed the training algorithm for such networks.

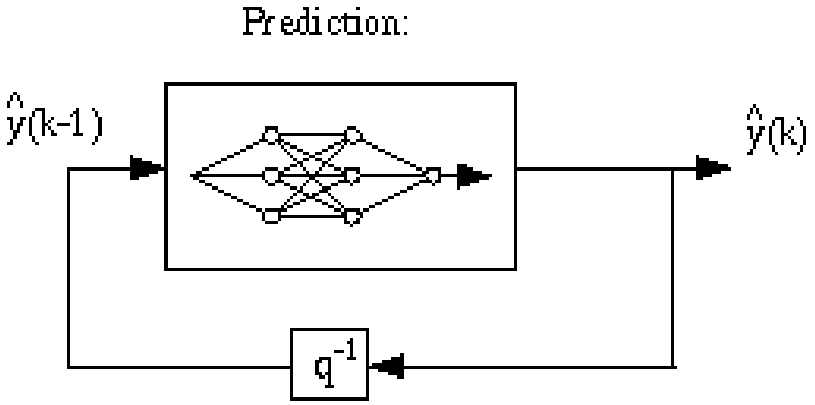
Wan's FIR Backprop Net



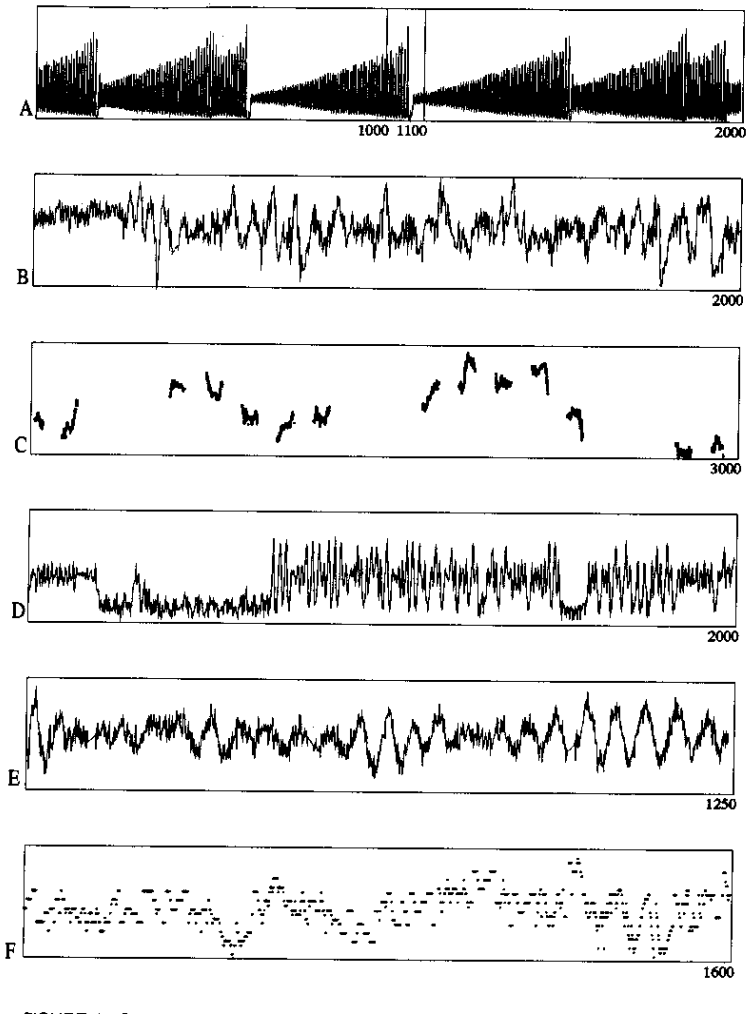


Recall: Training vs. Prediction





Sante Fe Institute Competition
6 unknown series



A A clean physics experiment. 1000 points of the **fluctuations in a far-infrared laser**, approximately described by three coupled non-linear DE's.

B Physiological data from a patient with sleep apnea. 34,000 points of heart rate, chest volume, blood oxygen concentration, and EEG state of a sleeping patient.

C High-frequency **currency exchange rate** data. Ten segments of 3,000 points each of the exchange rate between the Swiss franc and the U.S. dollar, 1-2 minutes apart.

D Numerically generated series. A driven particle in a 4-dimensional nonlinear multiple-well potential (9 degrees of freedom) with a small nonstationary drift in the depths.

E Astrophysical data from a **variable white dwarf** star. 27,704 points in 17 segments of the time variation of the intensity.

F J.S. Bach's final **(unfinished) fugue** from *The Art of the Fugue*.

Wan's Entry in Sante Fe Institute Competition

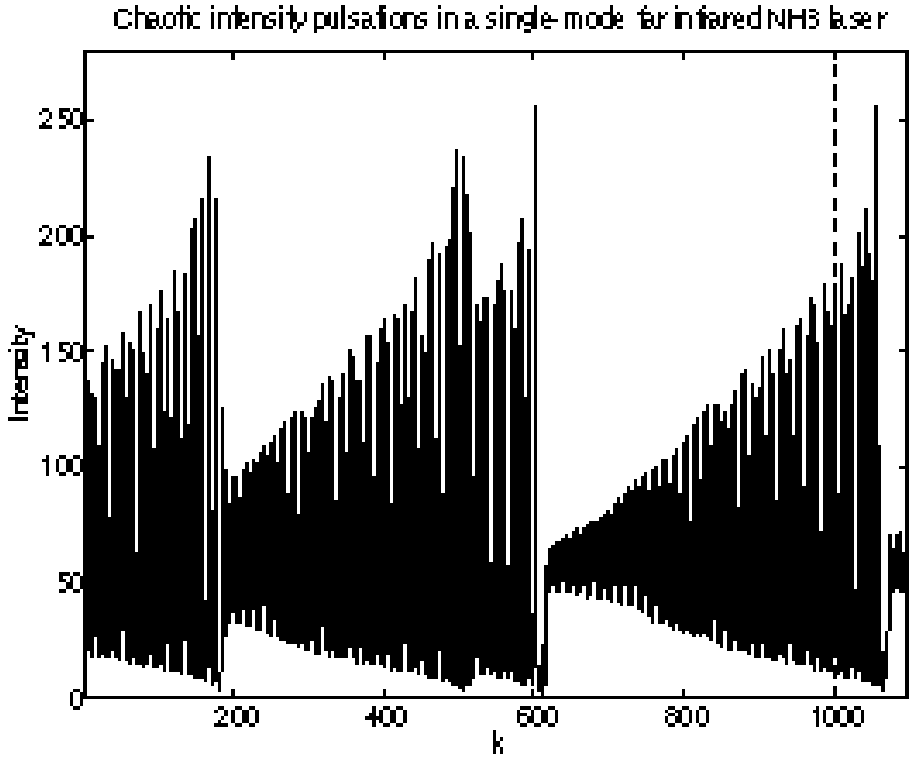
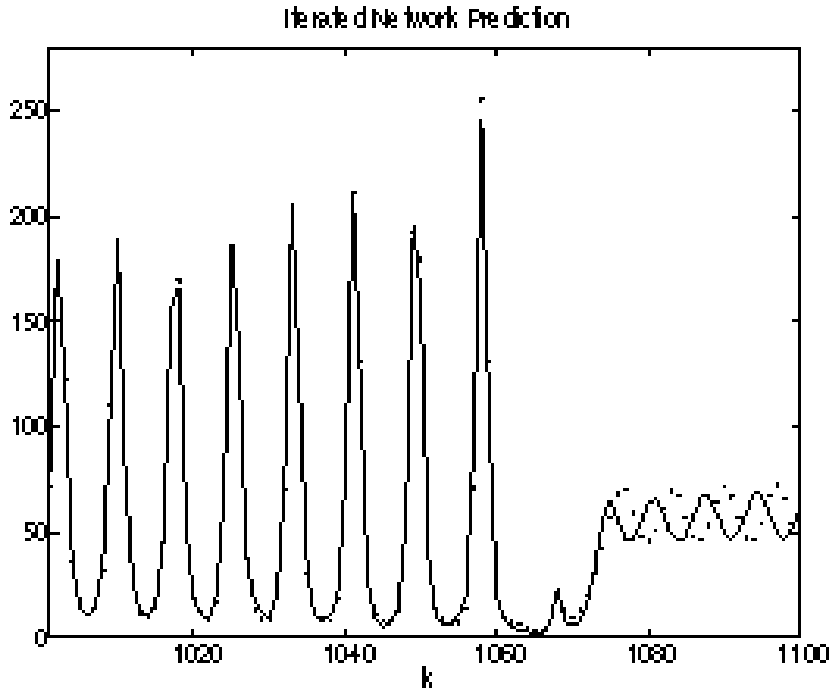
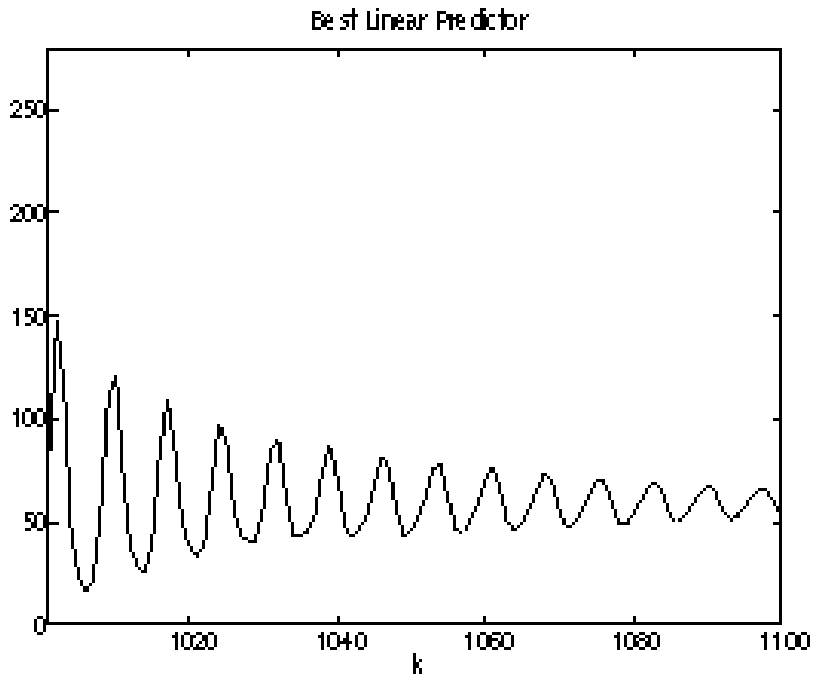


Fig. 3: 1000 points of laser data.

Wan's Prediction Expanded in Sante Fe Institute Competition





Solid line is actual

dashed line is predicted.

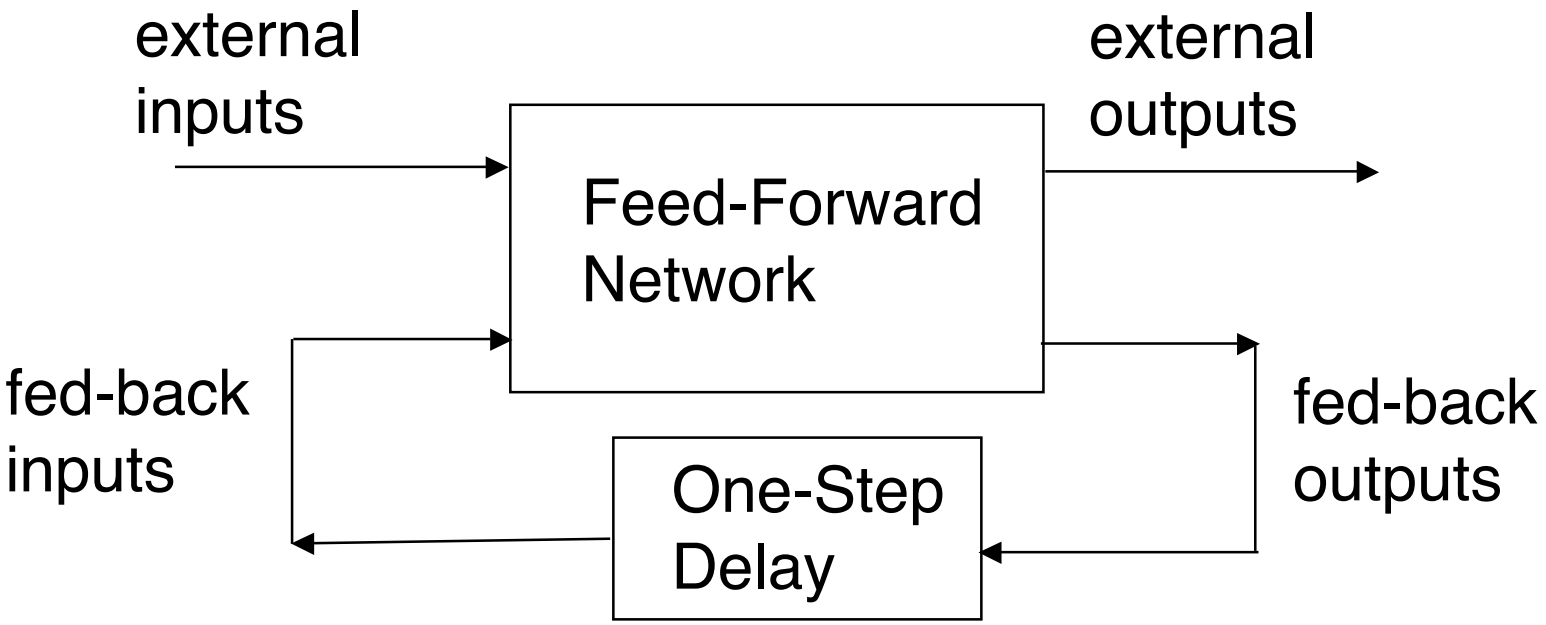
Recurrent Neural Nets

- These are networks with cycles.
- Elman networks
- Jordan networks
- RTRL (Real-time Recurrent Learning) networks

Applications of Recurrent Nets

- Natural Language Processing (learning a grammar from examples)
- Bankruptcy Prediction (credit worthiness)

A General Model for Recurrent Networks



A subset of the inputs and output of a feed-forward network are devoted to feedback connections.

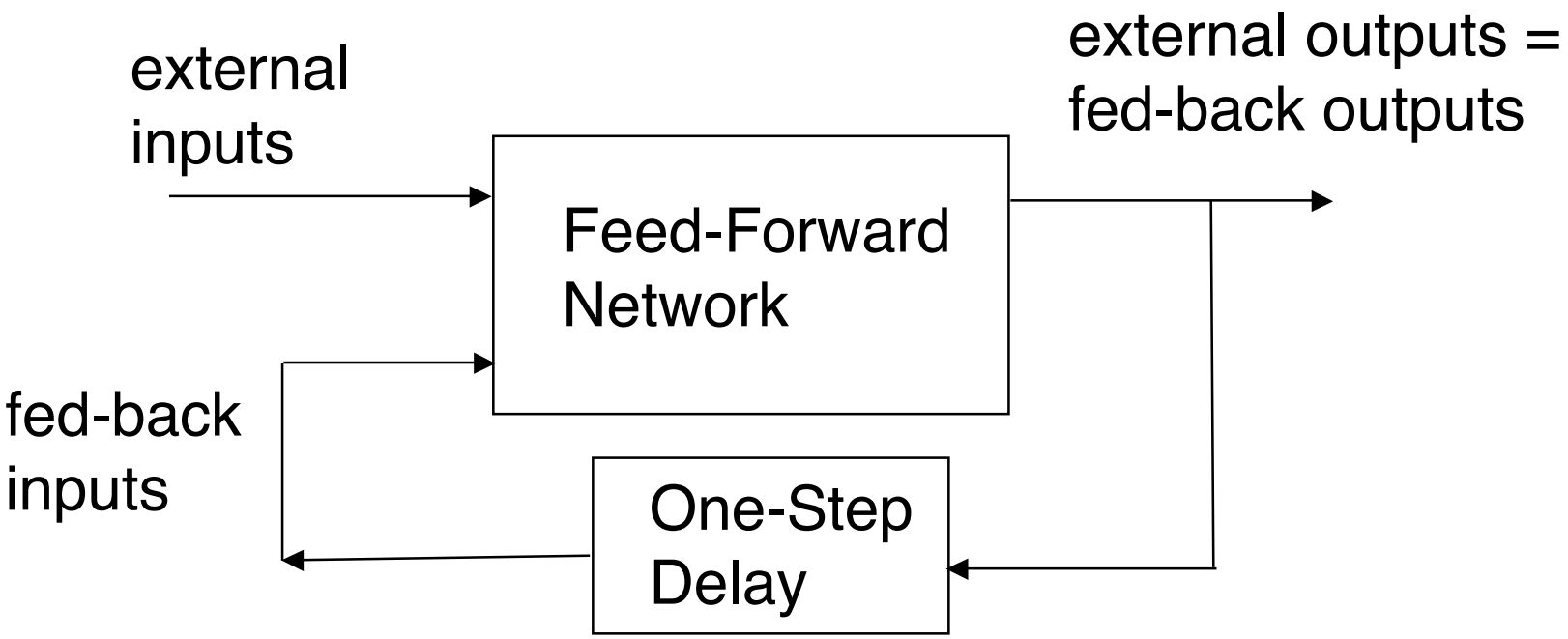
The Main Issue

- What are the **desired** values for the outputs that are fed back?
- (The desired values for the external outputs are assumed to be given.)
- Without the desired values for the feed-forward part it does not seem possible to use backpropagation.

Elman Approach (1990)

- In this model, the fed-back outputs and the external outputs are assumed to be the same.
- Since there are given desired values for the external outputs, these can be used to train the network.
- The Jordan model (1986) is similar to the Elman model, except that **delayed versions** of the output are feed back as inputs, resulting in exponential smoothing.

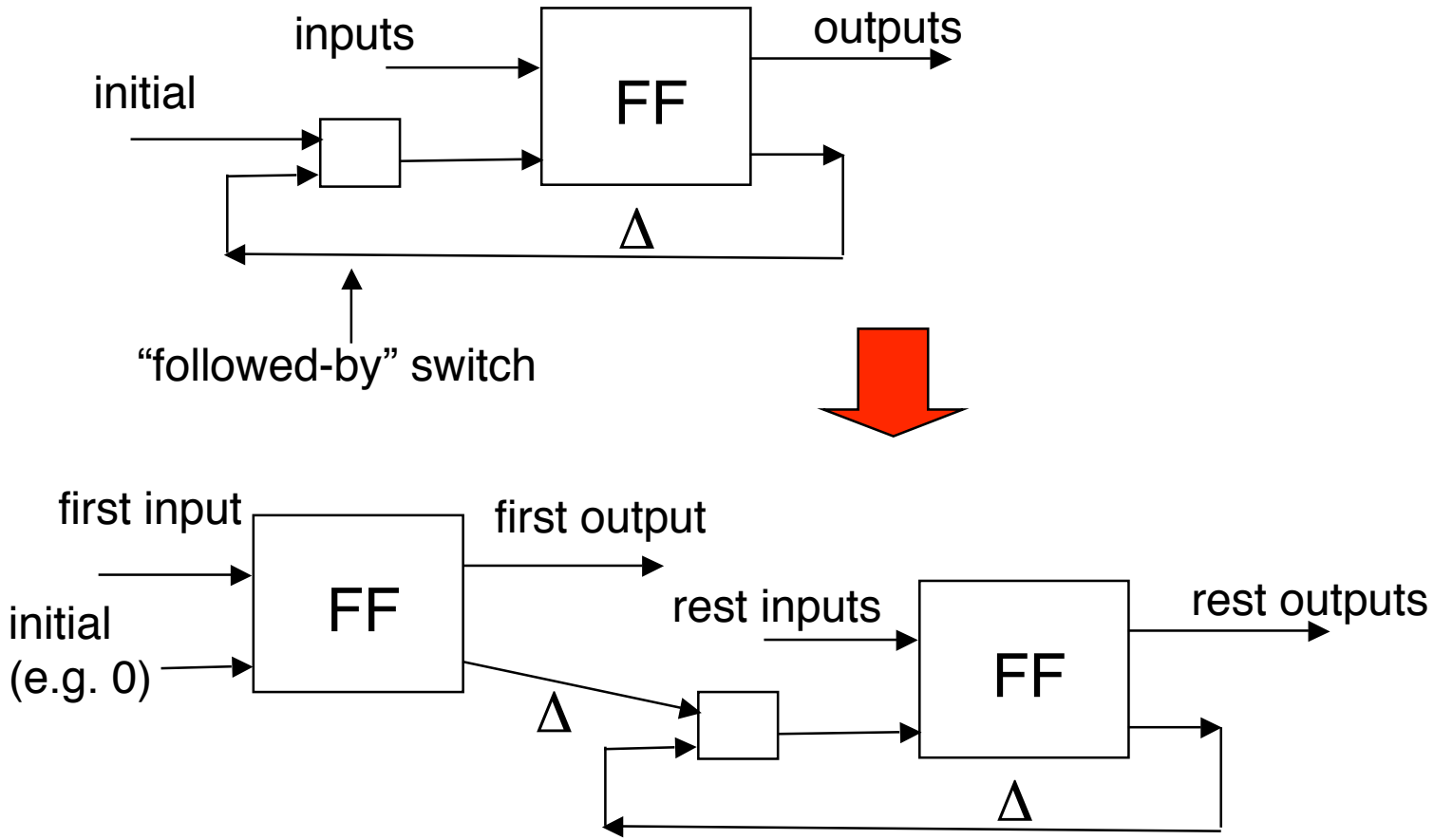
Elman Model


external
inputs
external outputs =
fed-back outputs
Feed-Forward
Network
fed-back
inputs
One-Step
Delay

Problem with the Elman Model

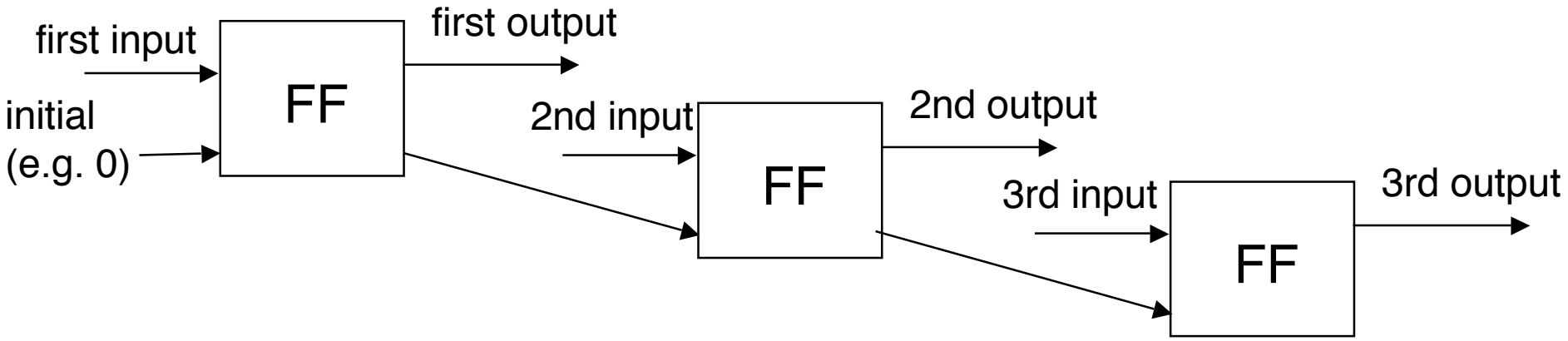
- The model is not fully general, since it effectively assumes that desired values are **given** for the fed-back outputs.

Unrolling Idea



repeat the construction
on the right subsystem

Unrolling Idea

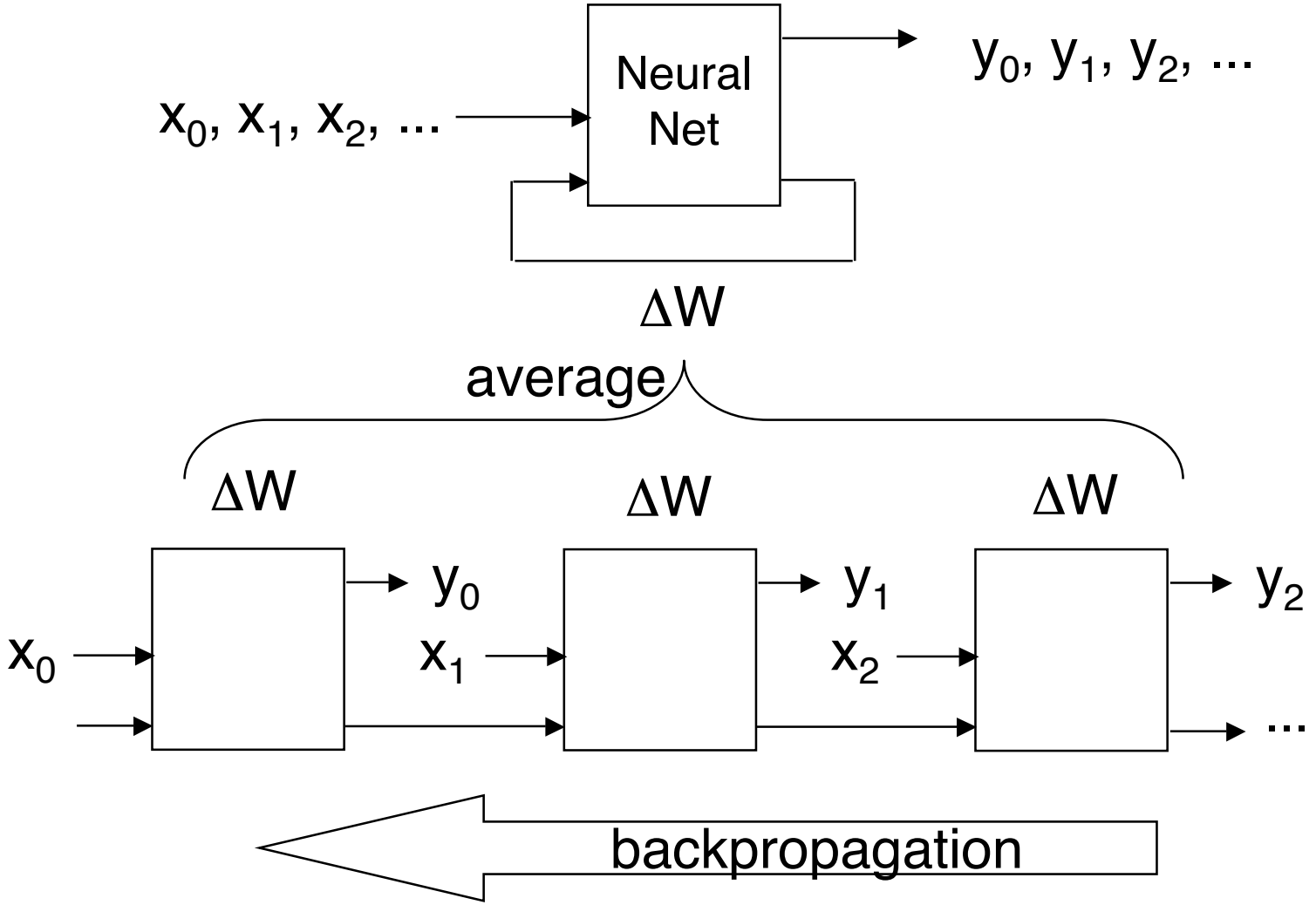


Backpropagation through time (BPTT)

- Training is **as if** the network were **unrolled** to accommodate the **entire sequence** of input samples.

- Only **one set of weights** is actually used in operation; the weight changes are **averaged** across stages to get the actual weight change

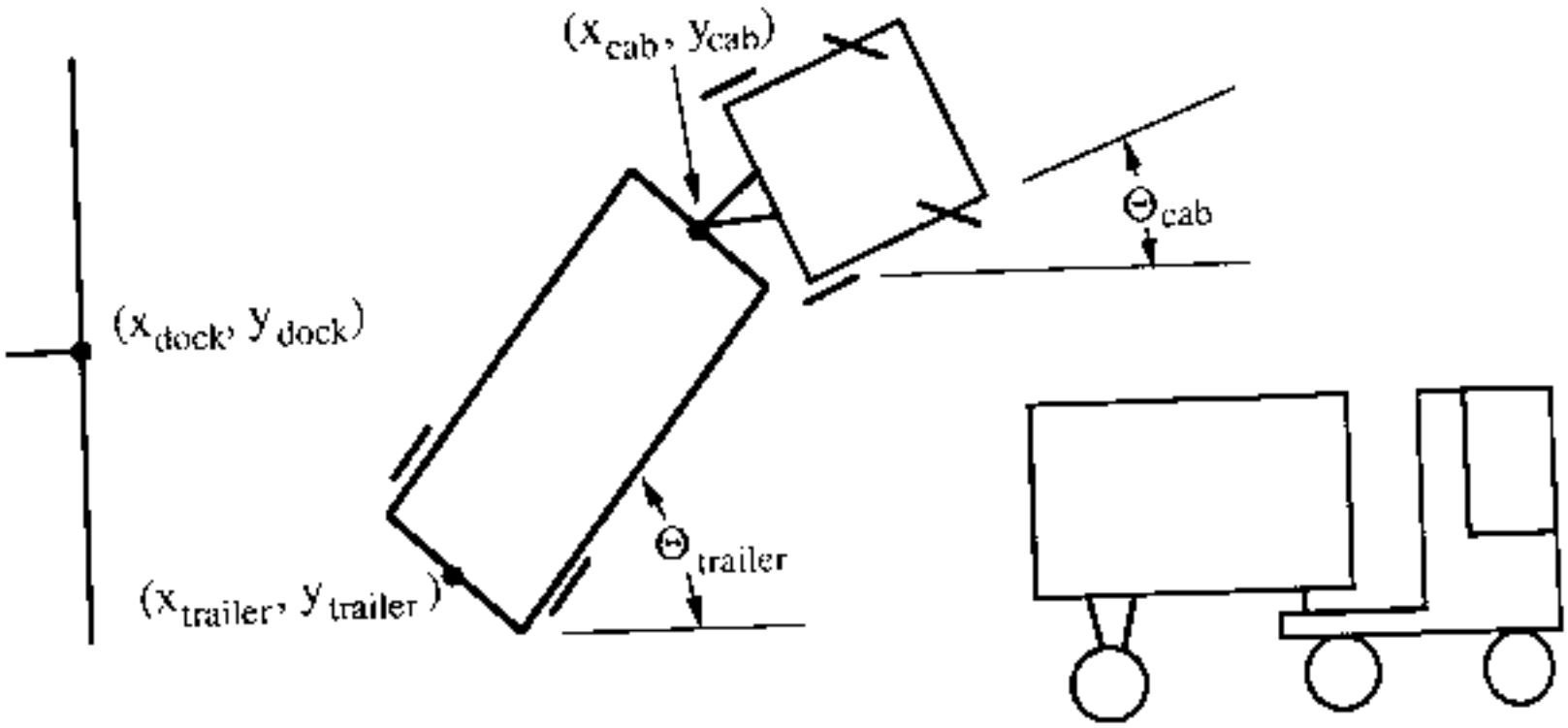
Backpropagation through time (BPTT)



BPTT Application

- The Truck Backer-Upper, D. Nguyen and B. Widrow
- reprinted in Miller, Sutton, and Werbos (eds.), Neural Networks for Control, MIT Press, 1990.
- Problem: Back up a truck so that $(x_{trailer}, y_{trailer}) = (x_{dock}, y_{dock})$, given initial values for $(x_{trailer}, y_{trailer}, x_{cab}, y_{cab}, \theta_{trailer}, \theta_{cab})$

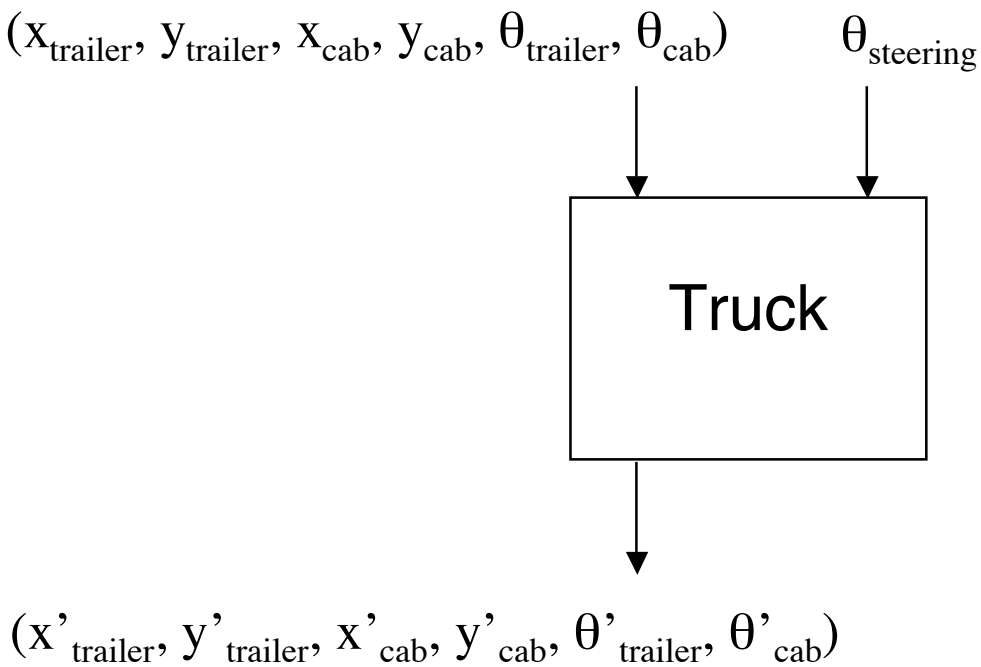
Truck-Backer Problem



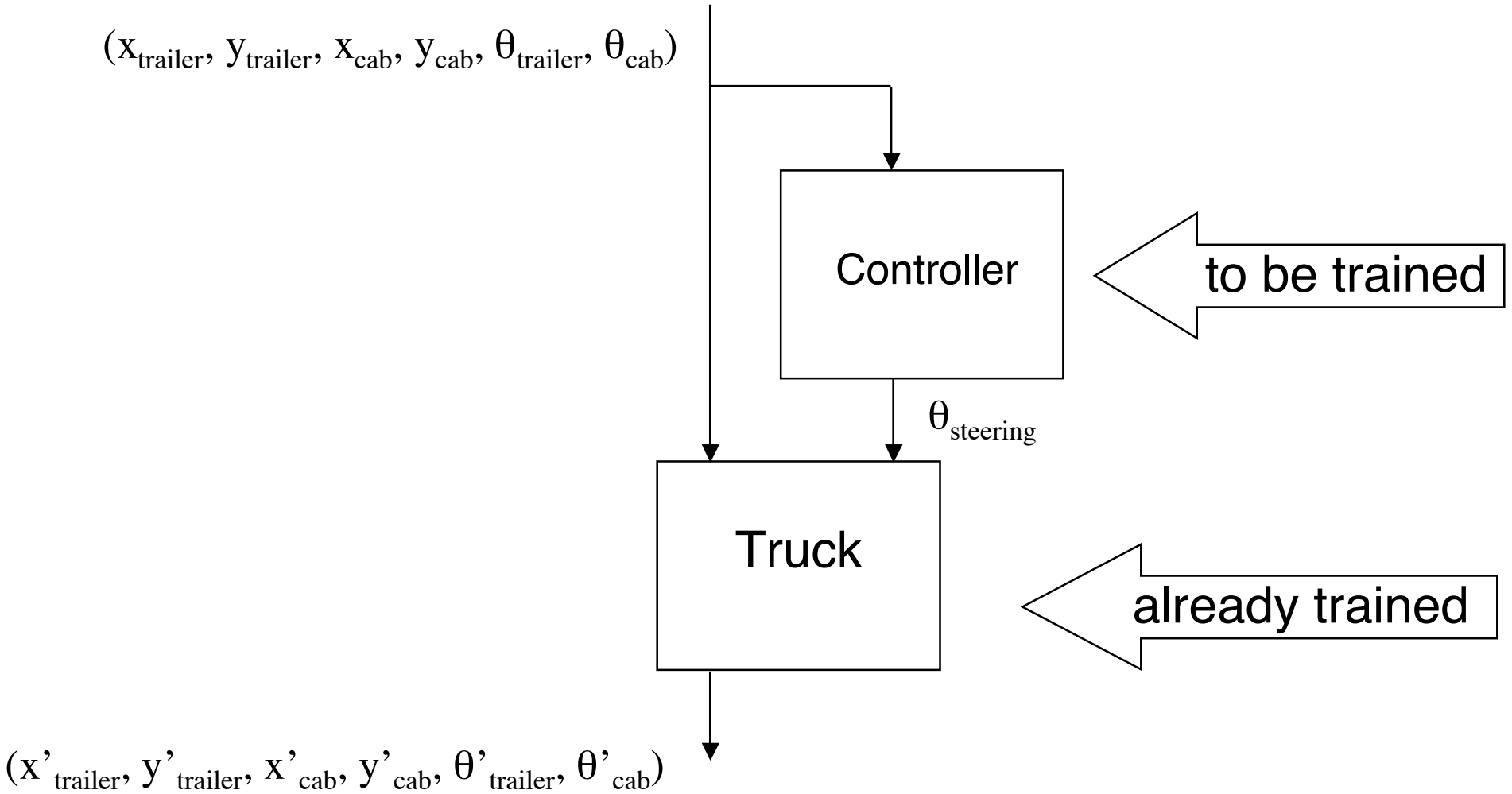
Training the Truck-Backer

- The truck moves in small time increments Δ
- A neural net is first trained to mimic the truck backing using **real truck dynamics**.
- Given the current state at a time t (which includes the steering angle), the network learns to determine the next state (at time t +Δ).
- This is done by starting the truck in random states, observing the error between what the network does and the dynamic model, and adjusting the weights.

The function being learned



Truck-Controller Combo

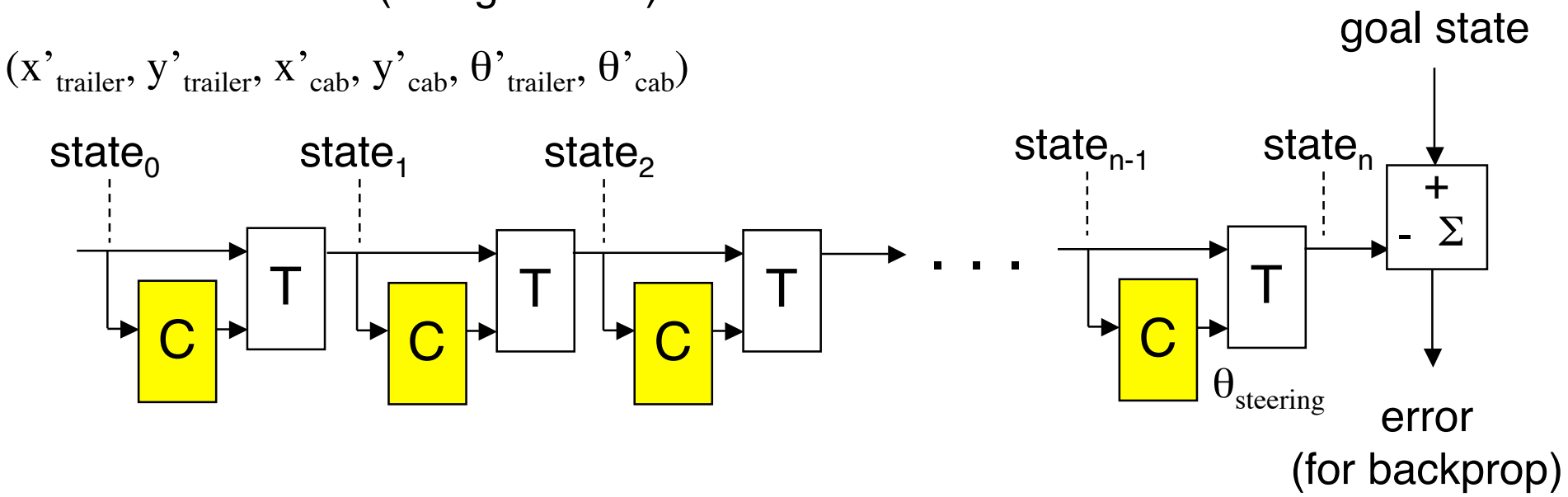


Training the Truck-Backer

- Starting from a random position, the controller backs up the truck one step at a time, until the goal is reached, or an obstacle (such as a side wall) is hit.
- An error value is produced by comparing the desired final state with the goal.
- The error value is backpropagated through the controller-truck combination to adjust the controller's weights, using BPTT.

BPTT for truck training

T = Truck (already trained, weights fixed),
C = Controller (being trained)



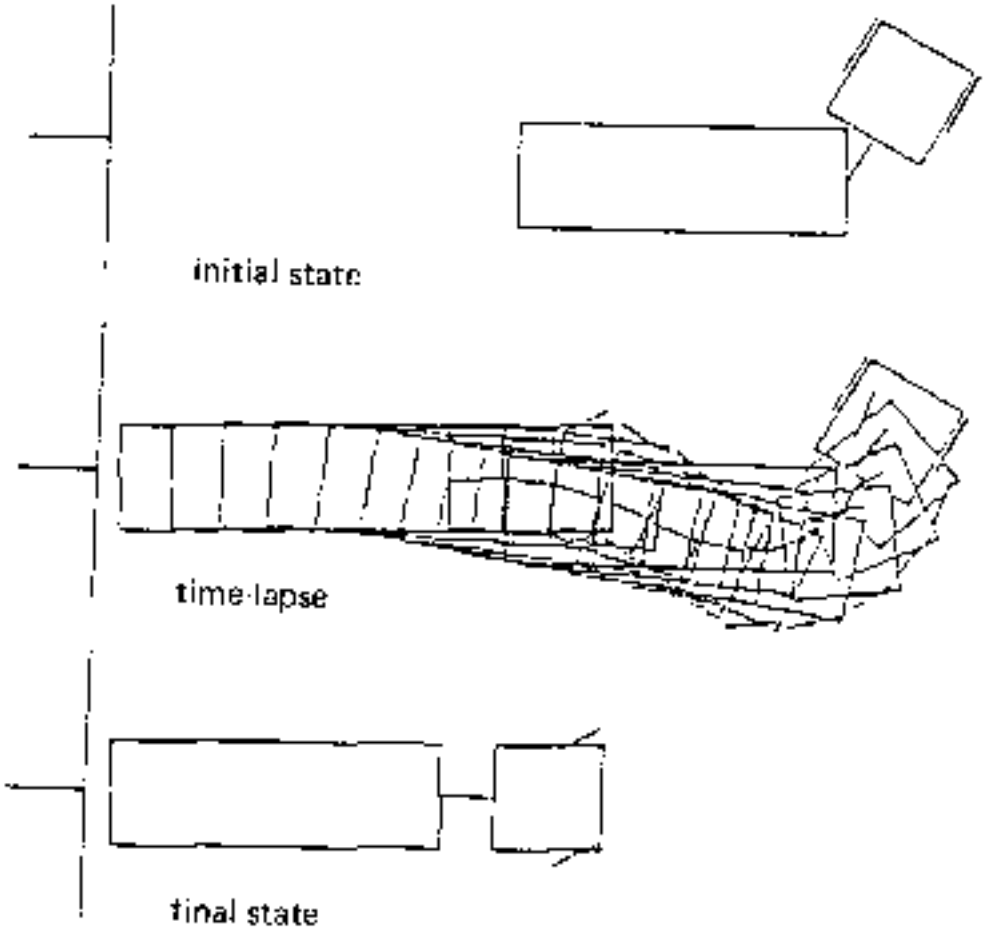
Network Statistics

- Truck Emulator:
  - 6-45-6 tansig-tansig network
- Controller
  - 6-25-1 tansig-tansig network

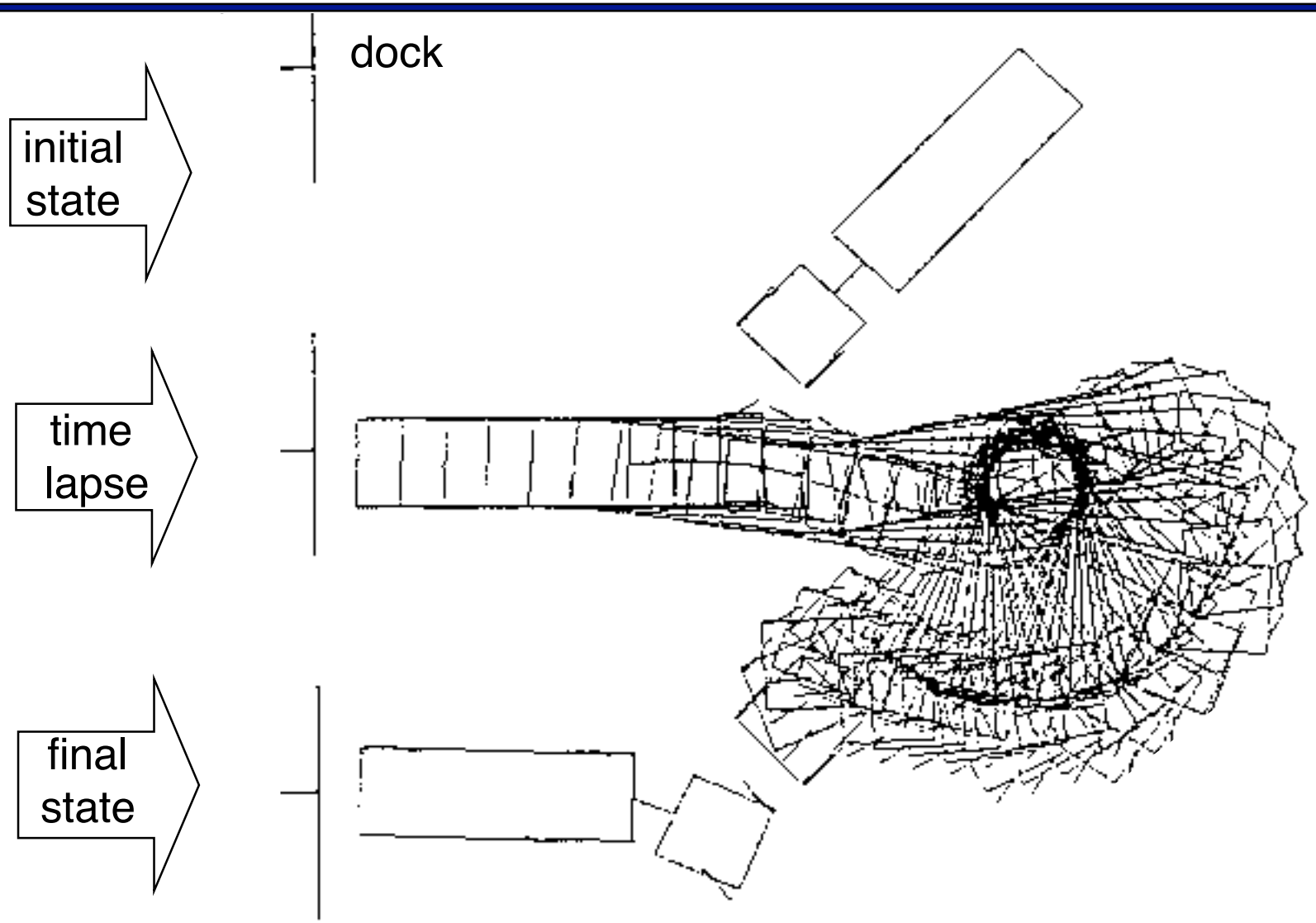
Training

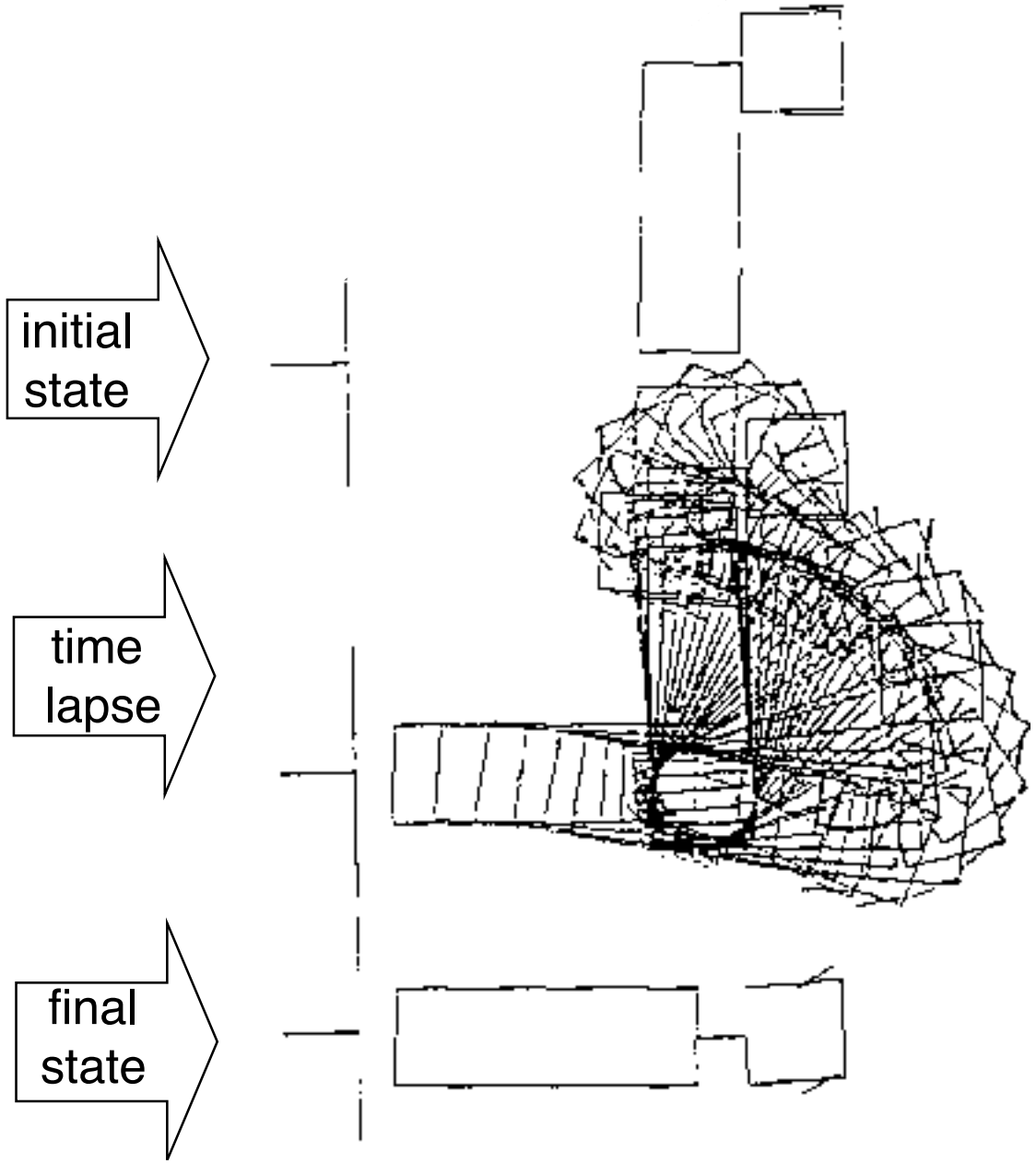
- 20,000 trials required to train
  - 16 lessons of 1000-2000 each
- Initially truck positioned very close to dock and in a nearly-correct position, so controller could **learn easy tasks first**.
- Final MSE was 3% of truck length, angle 7 degrees

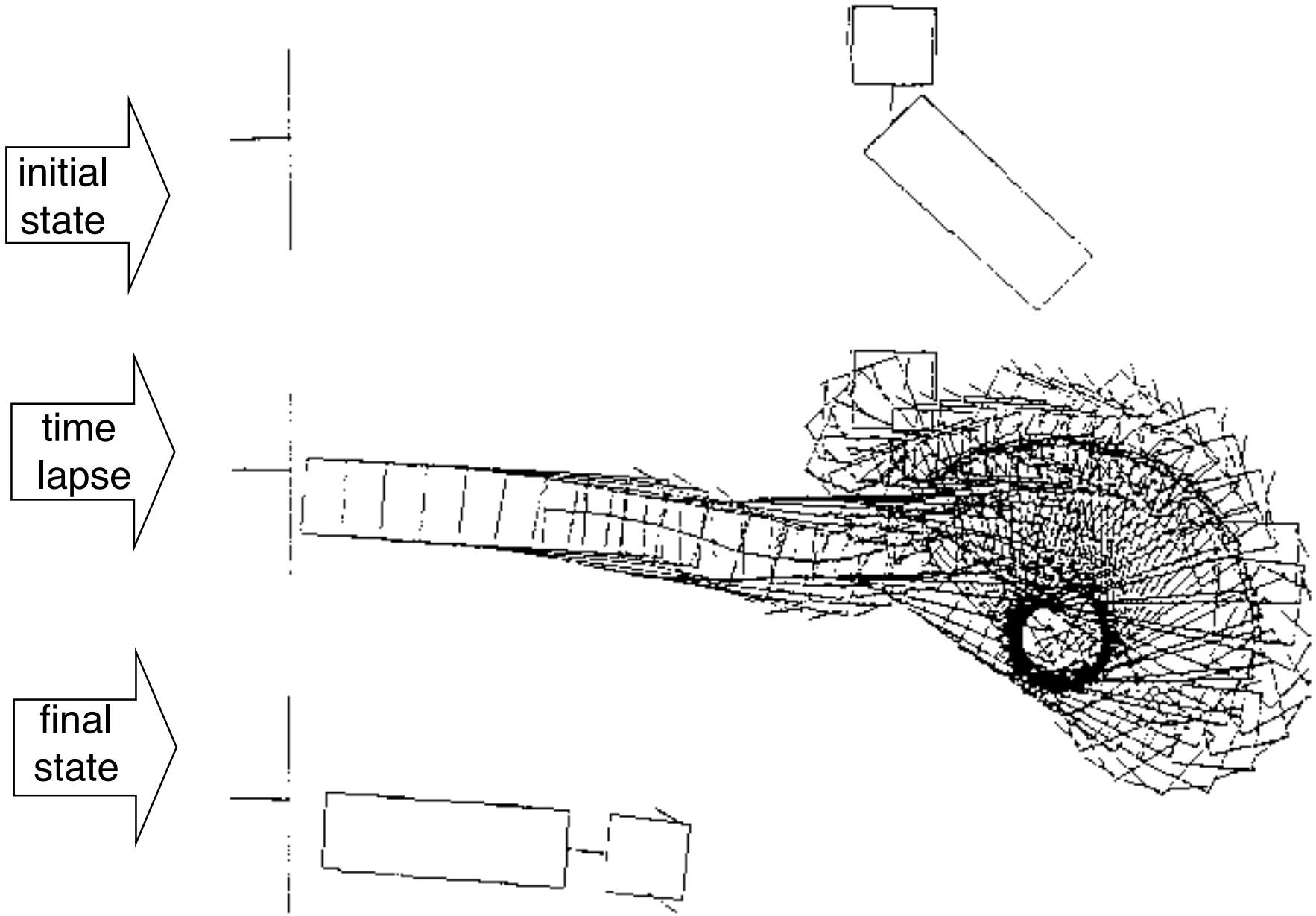
Simulations



Simulations




initial
state
time
lapse
final
state


initial state
time lapse
final state

Important Message

- The truck-backer example illustrates an important general idea, namely:

- By modeling a system as a neural network, we can include that model within larger contexts that are modeled similarly and treat the entire system in a uniform way (such as to adjusting parameters by backpropagation).

Real-Time Recurrent Learning
(Williams and Zipser, 1989)

- Trains a network without unrolling by deriving a recurrence for weight updates.

- The algorithm is “real-time” in the sense that the weights can be updated while the input sequence is being applied.

RTRL Weight Updating

- $\Delta w_{ij}(t) = \eta \sum_k e_k(t) p^k_{ij}(t)$

  k varying over the fed-back inputs

where

- $p^k_{ij}(t+1) = f'_k(net_k(t)) \left[\sum_r w_{kr} p^r_{ij}(t) + \delta_{ij} z_j(t)\right]$

  r varying over the fed-back inputs

- $p^k_{ij}(0) = 0$
- $z_j(t)$ is the value of the output of neuron j at time t
- δ_{ij} is the Kronecker delta (= 1 if i=j, 0 otherwise)