Recurrent Neural Networks

Fundamentals

Rahul Singh
rsingh@arrsingh.com

Feedforward Neural Networks

Feedforward Neural Networks use activation functions to transform the linear combination of inputs to make predictions

Feedforward Neural Networks

Feedforward Neural Networks use activation functions to transform the linear combination of inputs to make predictions

The neural networks we've seen in the past tutorials are also known as Feedforward neural networks because the data flows through the network from the input layer, to the hidden layers to the out put layer **without cycles or feedback loops**

Feedforward Neural Networks

Feedforward Neural Networks use activation functions to transform the linear combination of inputs to make predictions

$$Z_1 = f_1(XW_1 + \beta_1)$$
$$\hat{Y} = f_2(Z_1 W_2 + \beta_2)$$

$f_1(g)$ and $f_2(g)$ are the activation functions in Layers L_1 and L_2

Matrix Equations to compute the Predicted values $\hat{Y}$ given inputs X, Weights W_1 and W_2 and Biases β_1 and β_2

X is the matrix of n input features and m observations for each.
Each of the n features are independent in time and the order doesn't matter

Example: Features in a House price prediction model:

The order of the features doesn't matter

x_1: Number of Bedrooms
x_2: Number of Bathrooms
x_3: Square Footage
x_4: Lot Size
x_5: Year Built

Lets take another example: Network Traffic Prediction

Problem Statement: We have a network service (http) and we want to predict the traffic every hour (for capacity planning). We want to scale up the service (add capacity) if we predict that the traffic is going to increase, and we want to scale down the service (release capacity) if we predict the traffic is going to decrease.

Here are the observations of traffic (Requests Per Hour) for the past 8 hours:

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

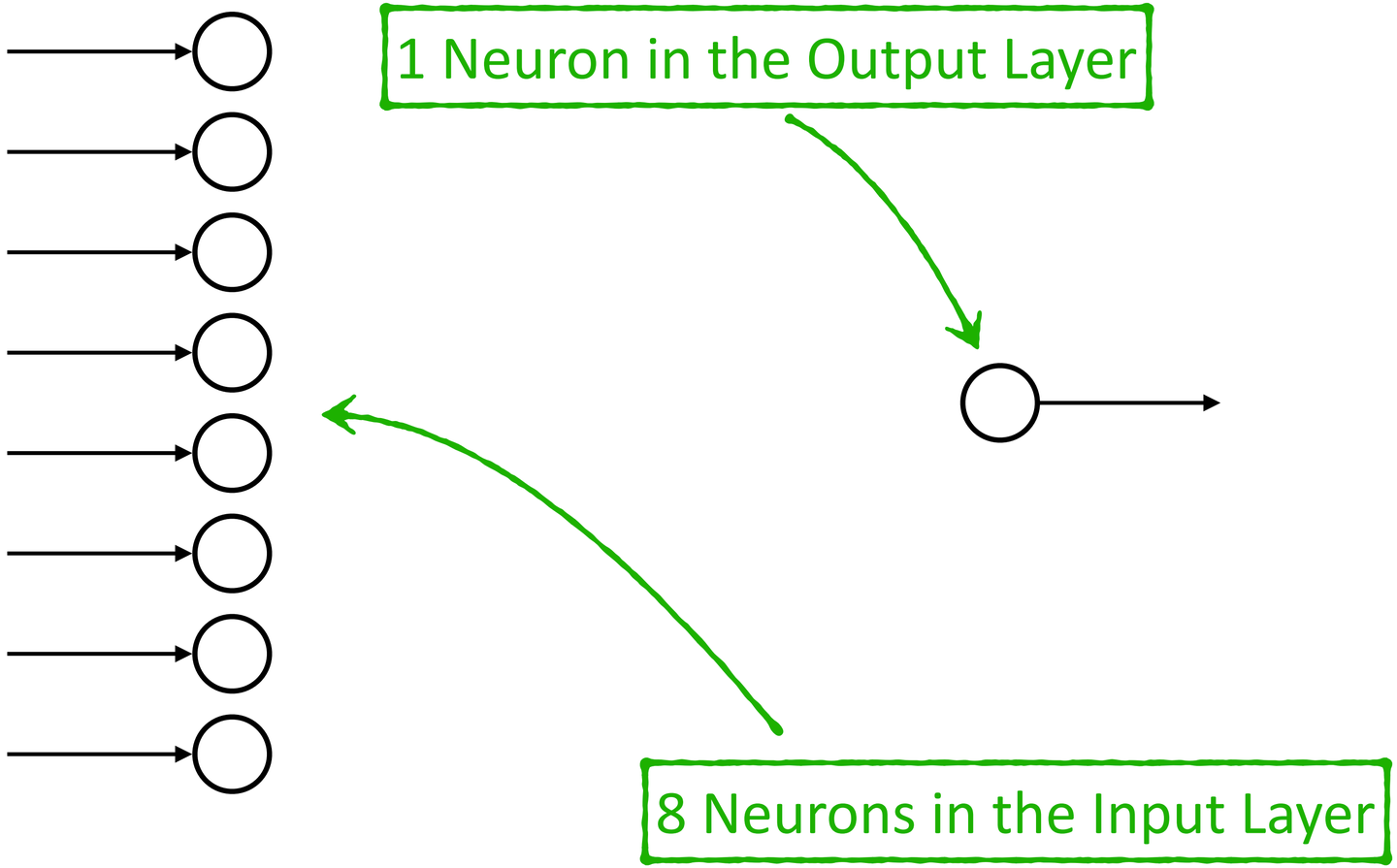
Question:
Can we predict the traffic at 6:00 AM?
Should we add capacity or reduce it?

Can we model this as a Feedforward neural network?

Neural Networks

Network Traffic Prediction

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

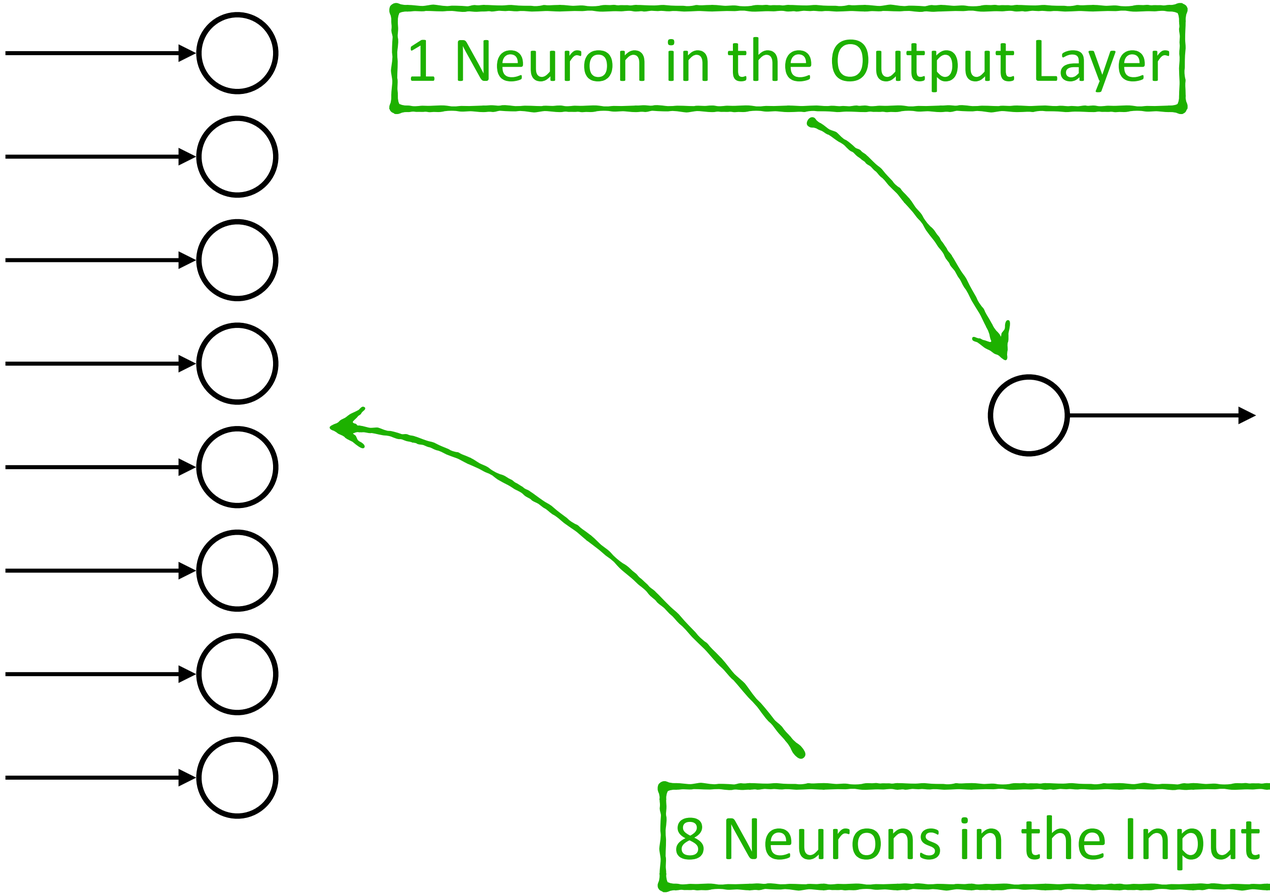


Can we model this as a Feedforward neural network?

Neural Networks

Network Traffic Prediction

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Linear combination of inputs

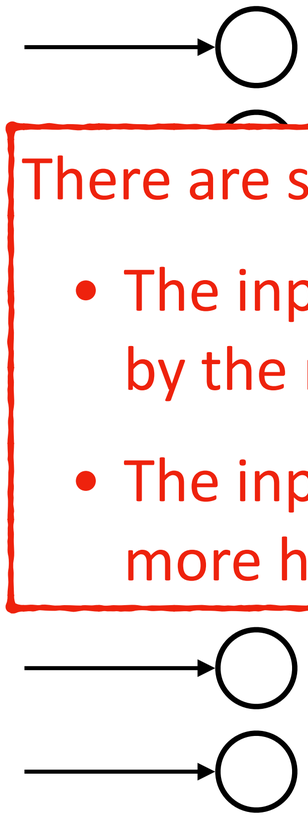
$$\hat{y} = \beta + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6 + w_7 x_7 + w_8 x_8$$

Can we model this as a Feedforward neural network?

Neural Networks

Network Traffic Prediction

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



1 Neuron in the Output Layer

There are several problems with this approach:

- The input data has a trend that is not captured by the model
- The inputs are fixed and cannot account for more historical data points

8 Neurons in the Input Layer

Linear combination of inputs

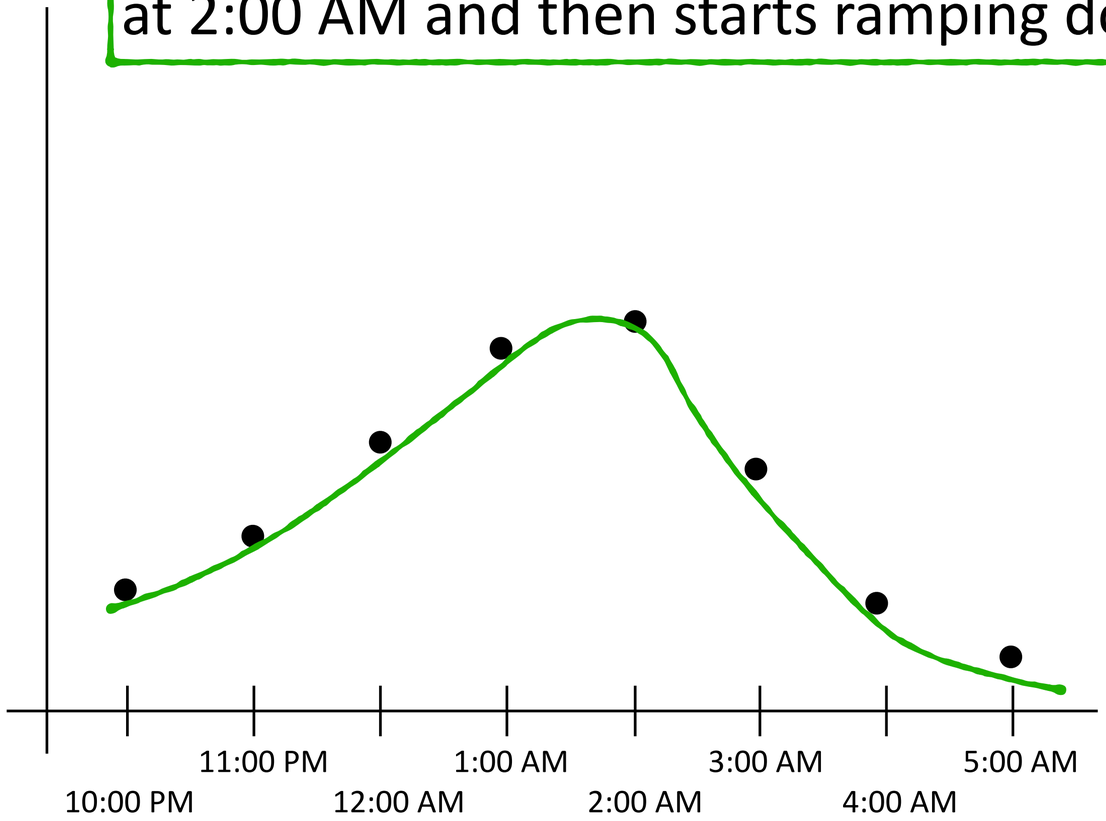
$$\hat{y} = \beta + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8$$

Can we model this as a Feedforward neural network?

Network Traffic Prediction

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

Traffic starts ramping up, at 10:00 PM, peaks at 2:00 AM and then starts ramping down

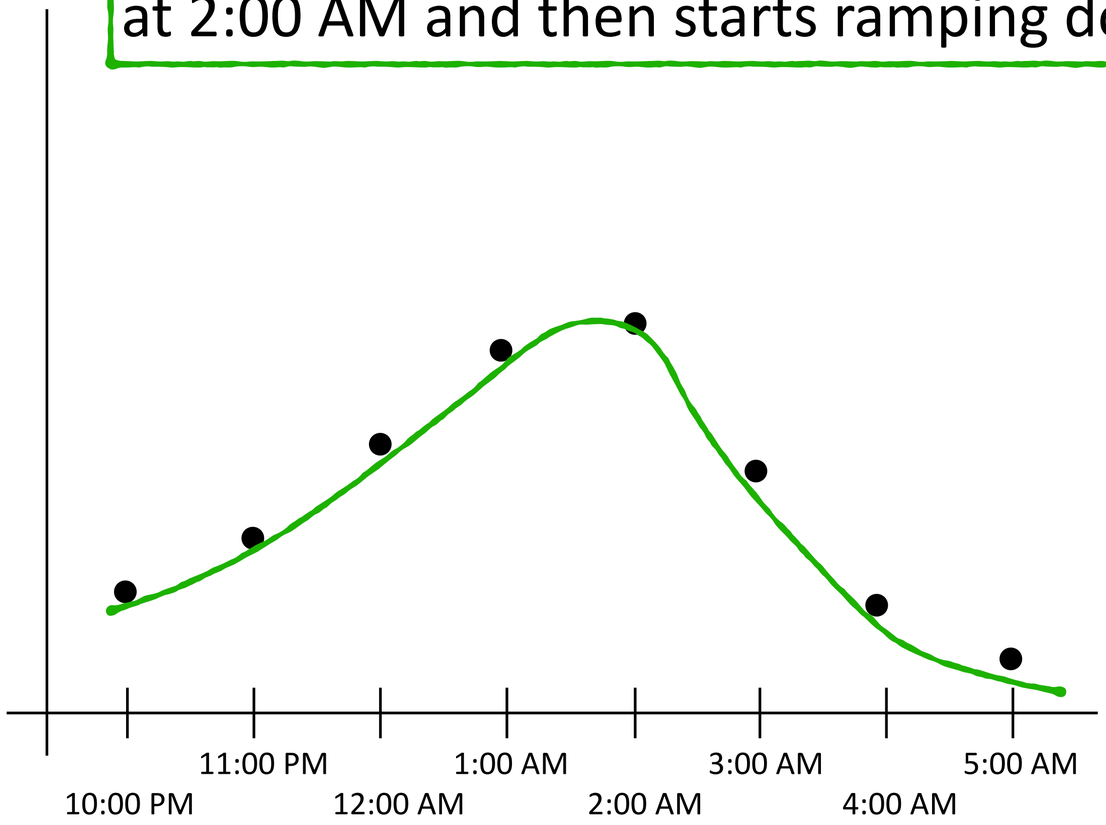


Can we model this as a Feedforward neural network?

Network Traffic Prediction

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

Traffic starts ramping up, at 10:00 PM, peaks at 2:00 AM and then starts ramping down



The Feedforward Neural Network can’t model this trend and simply calculates a prediction based on the weighted average of the inputs

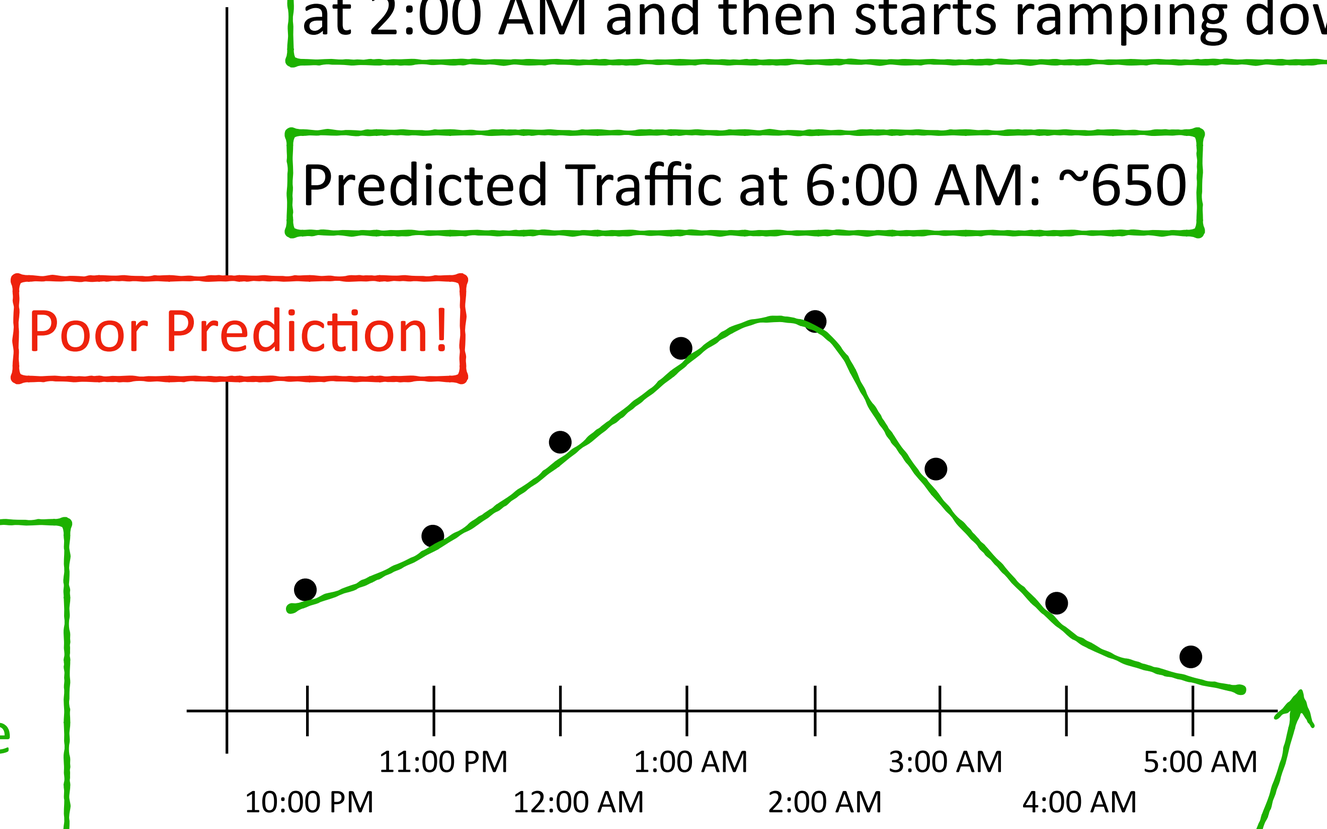
Network Traffic Prediction

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

Traffic starts ramping up, at 10:00 PM, peaks at 2:00 AM and then starts ramping down

Predicted Traffic at 6:00 AM: ~650

Poor Prediction!



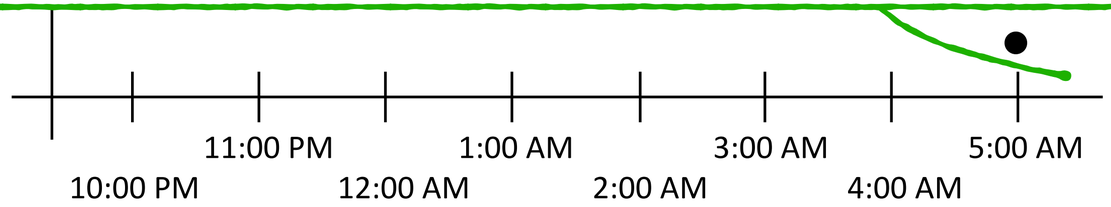
The Feedforward Neural Network can't model this trend and simply calculates a prediction based on the weighted average of the inputs

Traffic is actually in a decline. Should predict ~250

Network Traffic Prediction

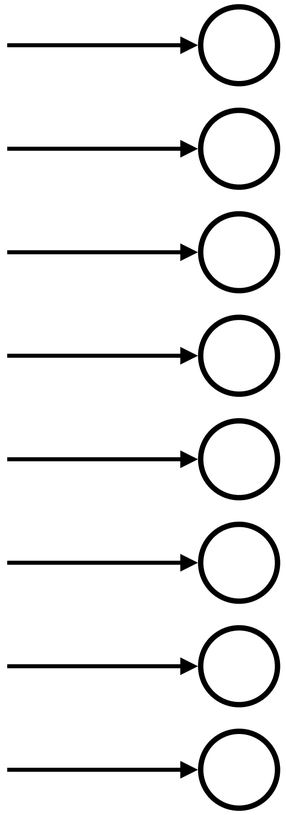
Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732

Feedforward Neural Networks cannot model inputs that have a temporal dependency and ordering



Recurrent Neural Networks

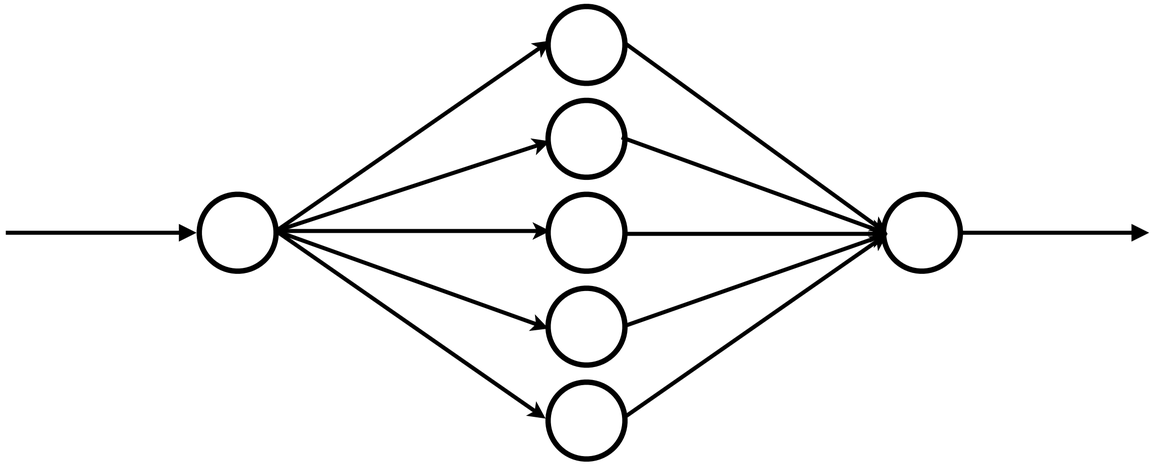
Recurrent Neural Networks can model inputs that have a temporal dependency and ordering



In a Feedforward Neural Network, all the inputs are fed to the network at the same time to produce a prediction

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

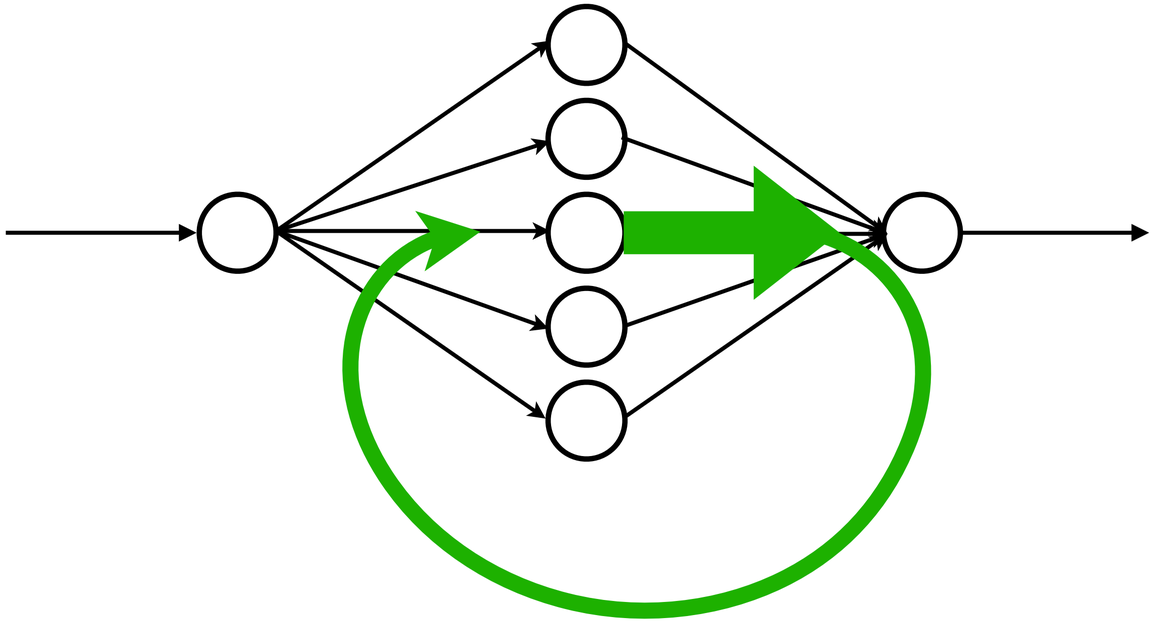


In an RNN, however since the inputs have a temporal dependency, the inputs are fed to the network and processed one at a time.

The structure of an RNN is similar to that of a FeedForward Network (input layer, hidden layers and an output layer) with one additional nuance…

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering



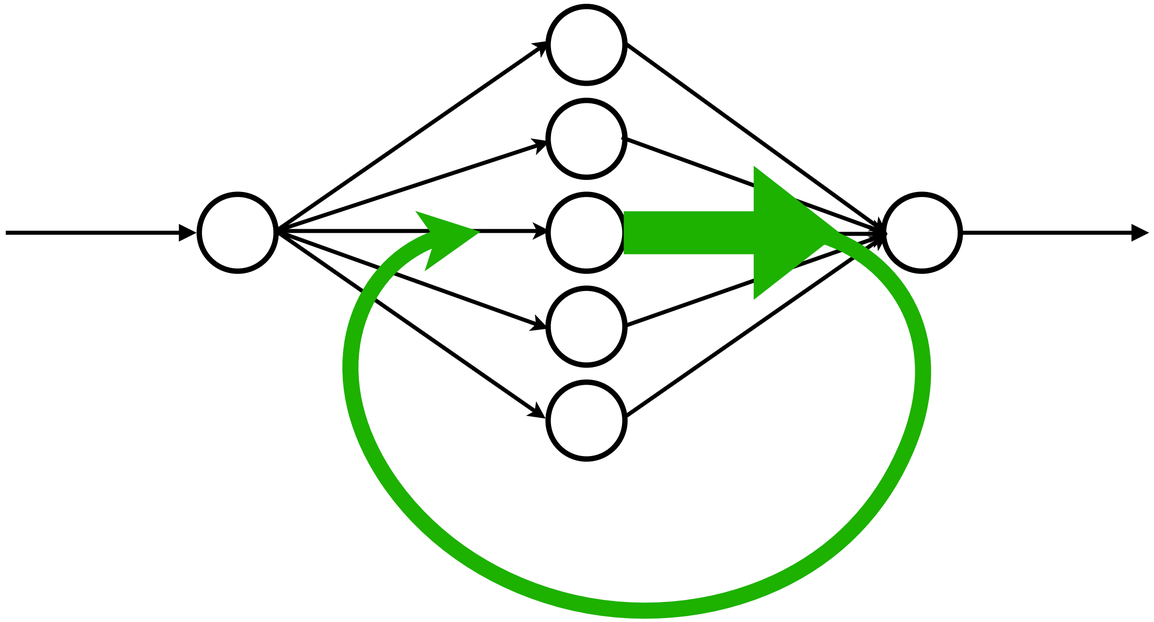
In an RNN, however since the inputs have a temporal dependency, the inputs are fed to the network and processed one at a time.

The structure of an RNN is similar to that of a FeedForward Network (input layer, hidden layers and an output layer) with one additional nuance…

… the output from the hidden layer at time t, is fed back as input to the hidden layer at time $t + 1$ along with the input at time $t + 1$

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering



In an RNN, however since the inputs have a temporal dependency, the inputs are fed to the network and processed one at a time.

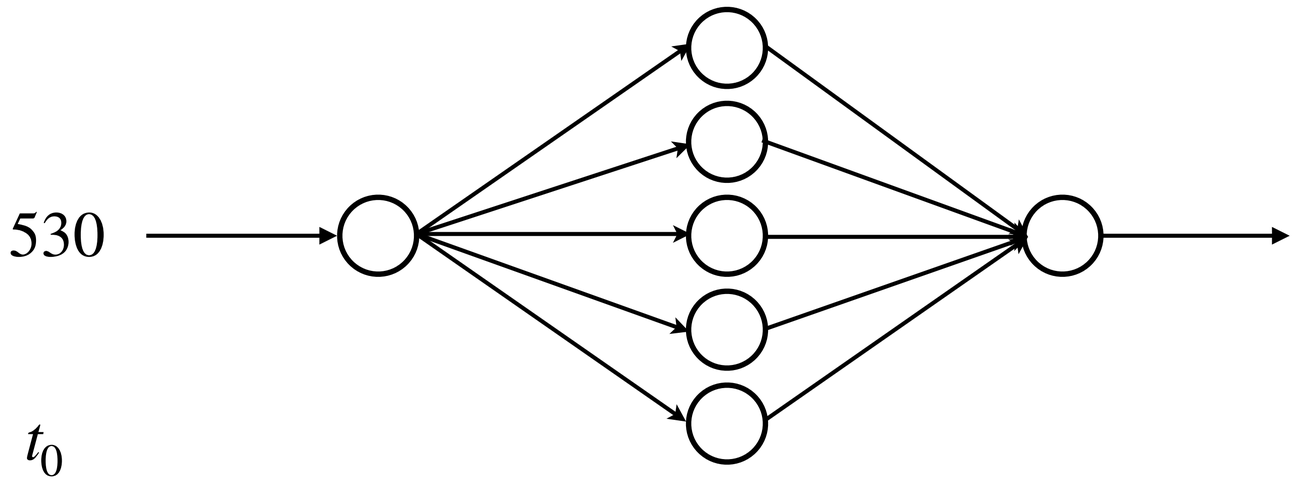
The structure of an RNN is similar to that of a FeedForward Network (input layer, hidden layers and an output layer) with one additional nuance…

… the output from the hidden layer at time t, is fed back as input to the hidden layer at time $t + 1$ along with the input at time $t + 1$

Let's walk through how this works…

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering



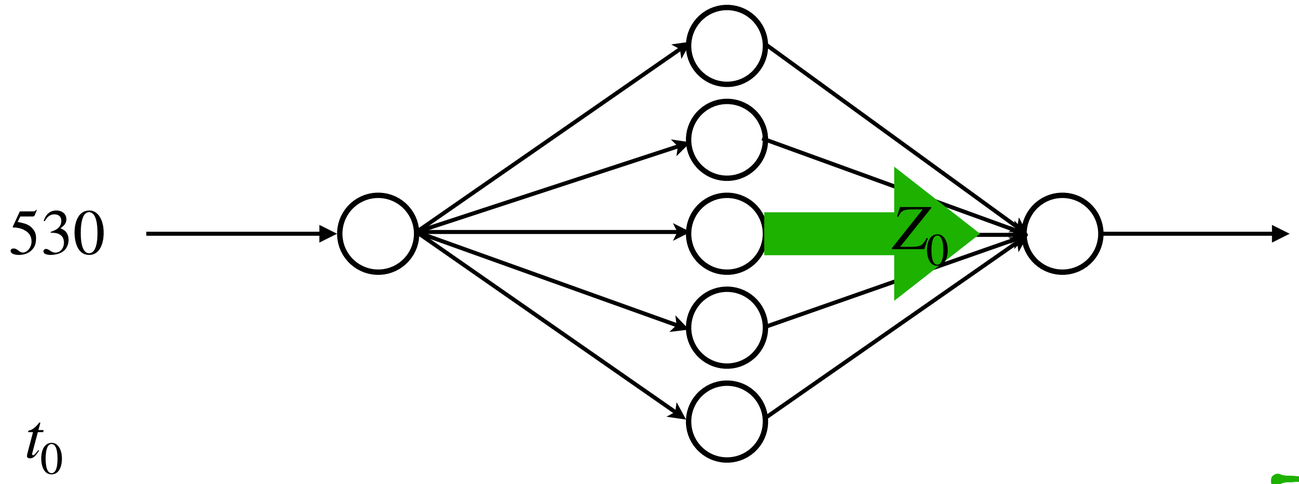
Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

At time t_0 (10:00 PM) the input is 530

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



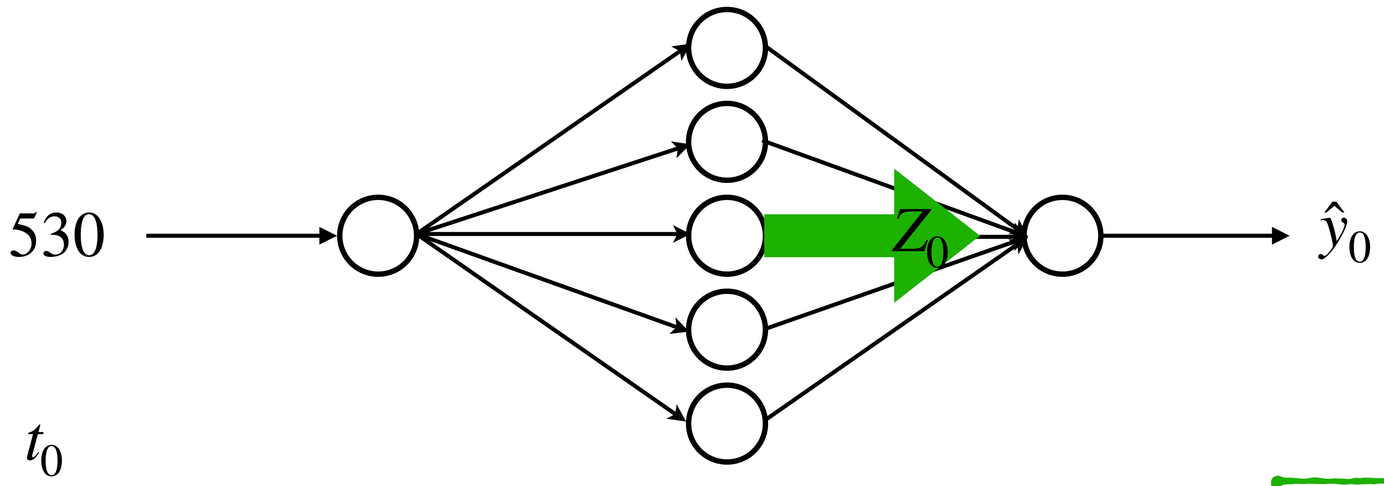
Produces Z_0 from Layer L_1

At time t_0 (10:00 PM) the input is 530

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



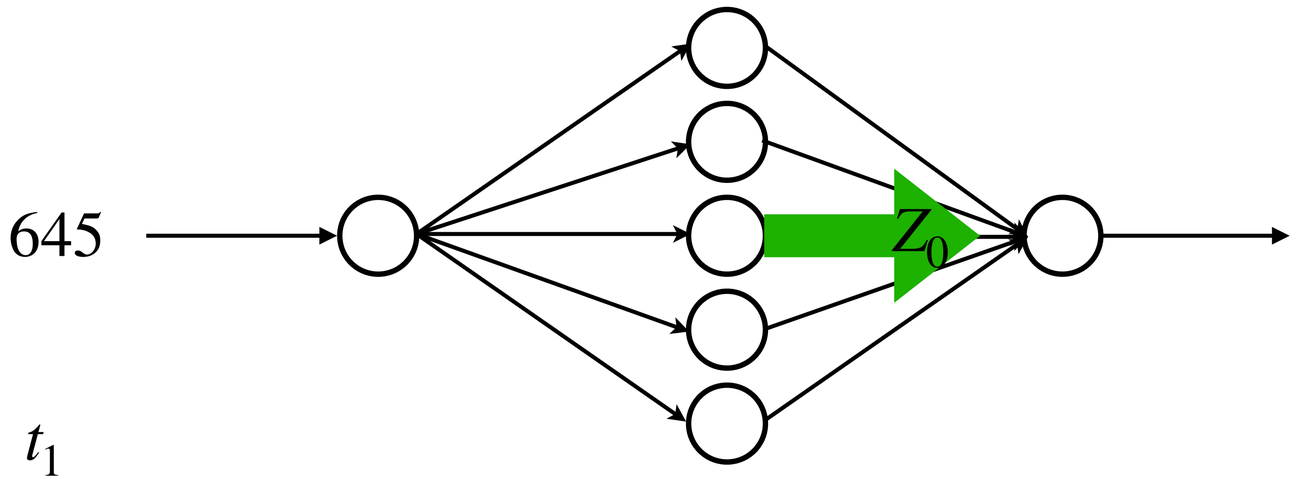
Produces output $\hat{y}_0$

Produces Z_0 from Layer L_1

At time t_0 (10:00 PM) the input is 530

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering



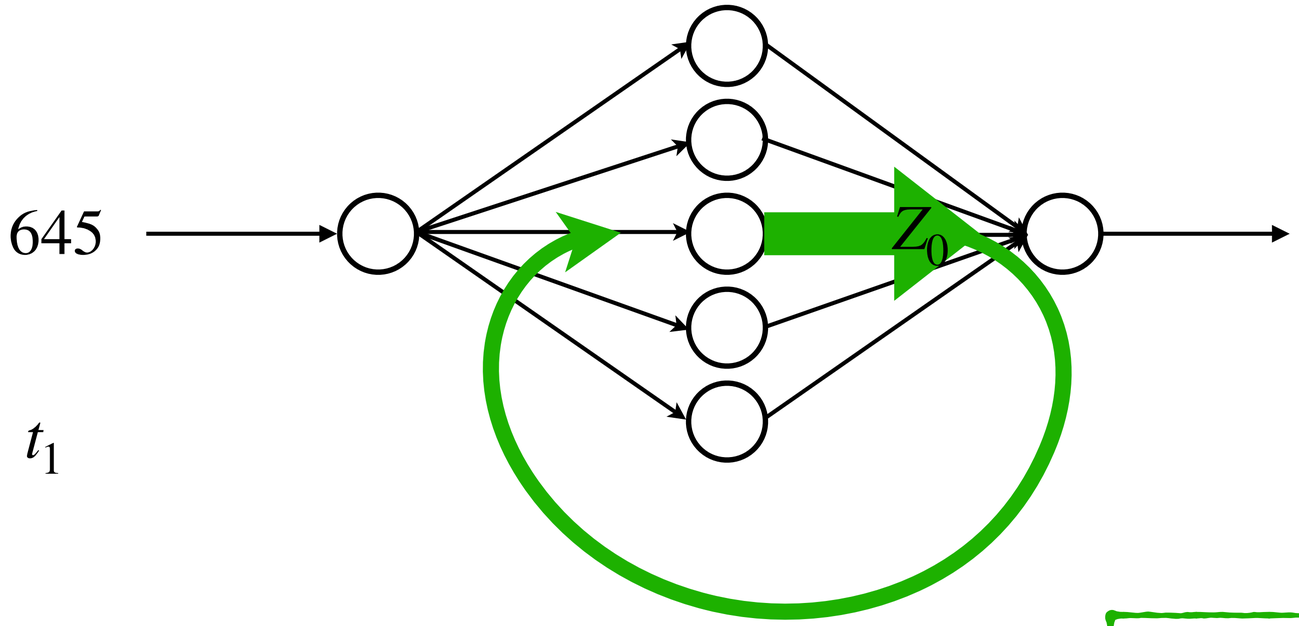
Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

At time t_1 (11:00 PM) the input is 645

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



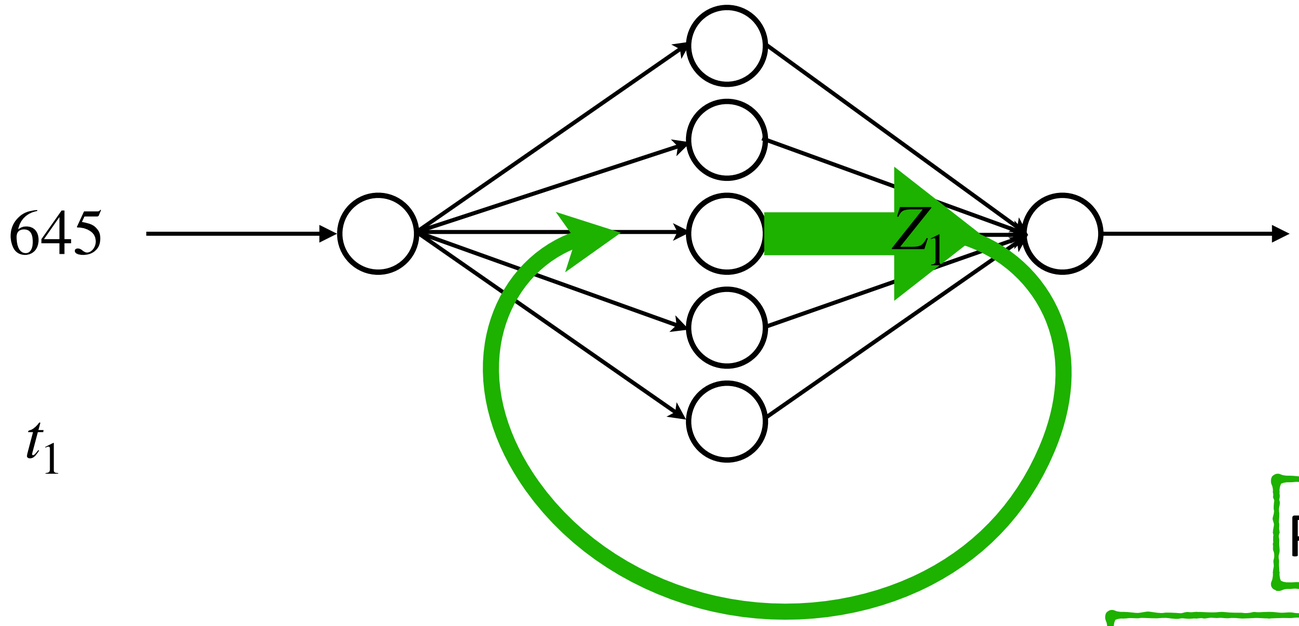
At time t_1 (11:00 PM) the input is 645

Z_0 (from t_0) is fed back to the hidden layer

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Produces Z_1 from Layer L_1

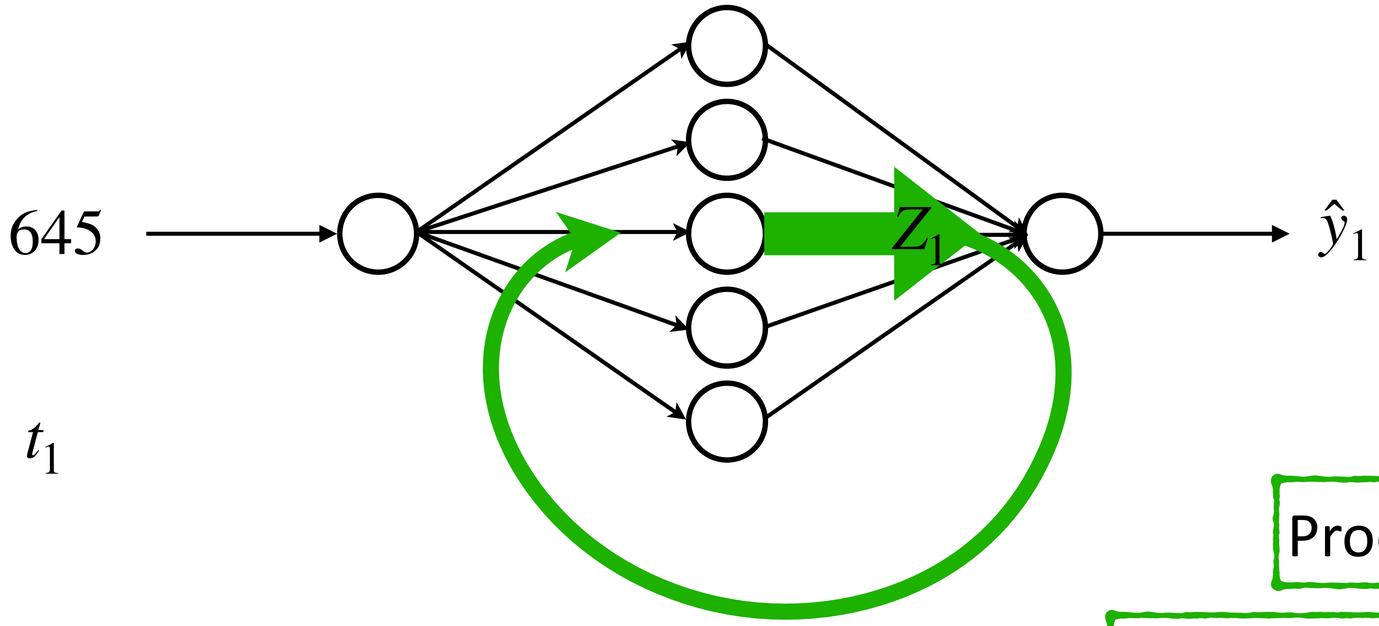
Z_0 (from t_0) is fed back to the hidden layer

At time t_1 (11:00 PM) the input is 645

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Produces output $\hat{y}_1$

Produces Z_1 from Layer L_1

Z_0 (from t_0) is fed back to the hidden layer

At time t_1 (11:00 PM) the input is 645

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

Let's unroll the loop so its easier to see what happens at each time step

Produces output $\hat{y}_1$

Produces Z_1 from Layer L_1

Z_0 (from t_0) is fed back to the hidden layer

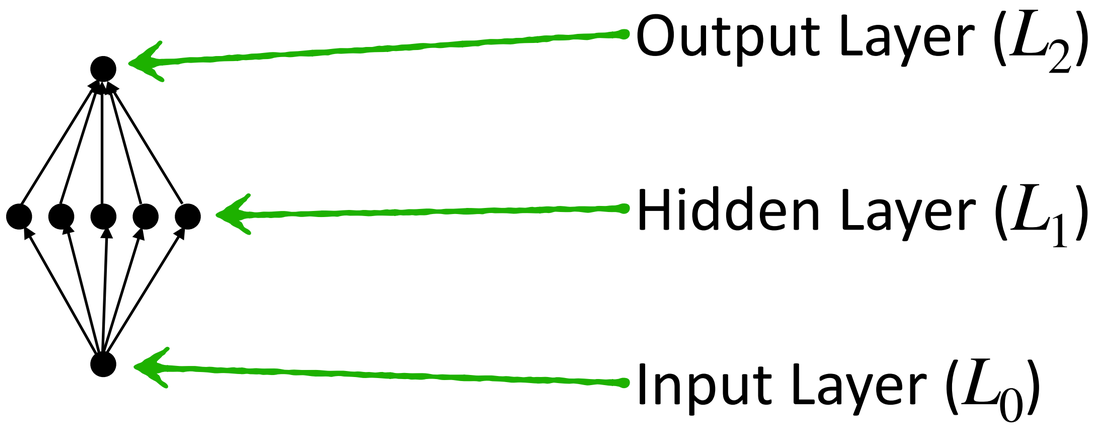
At time t_1 (11:00 PM) the input is 645

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…

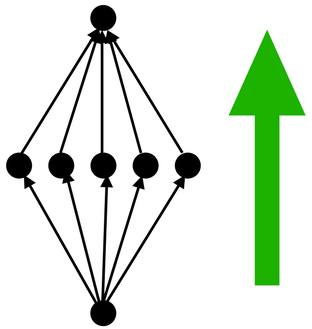


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…



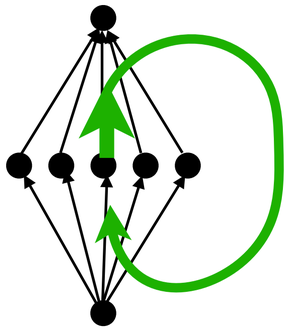
Inputs are fed from the input layer and move through the hidden layer to produce an output

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…



Output from the hidden layer at time t_0 is fed back as input to the same layer at time t_1

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…

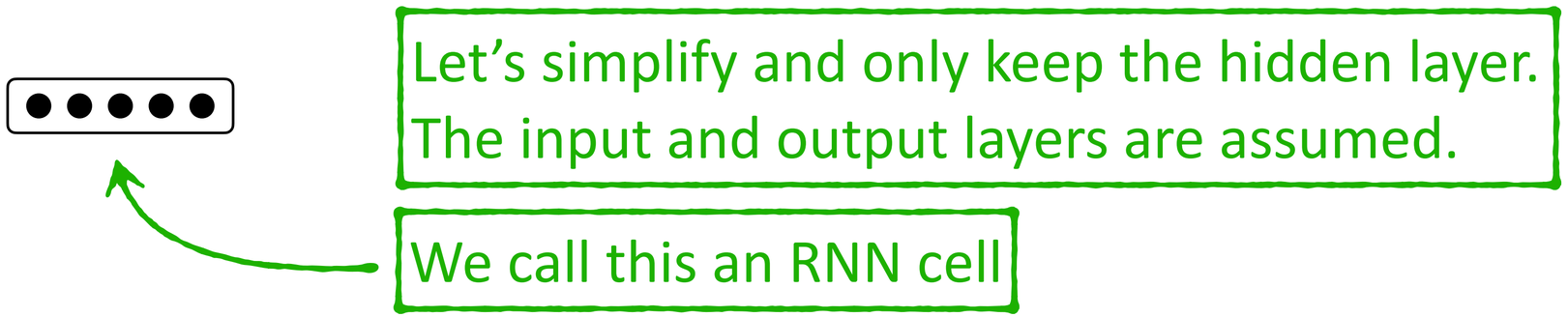
Let’s simplify and only keep the hidden layer. The input and output layers are assumed.

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…

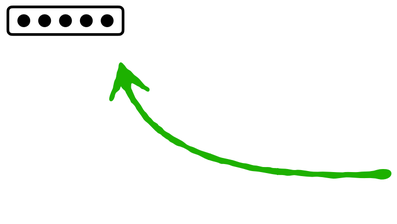


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…



Let’s simplify and only keep the hidden layer. The input and output layers are assumed.

We call this an RNN cell

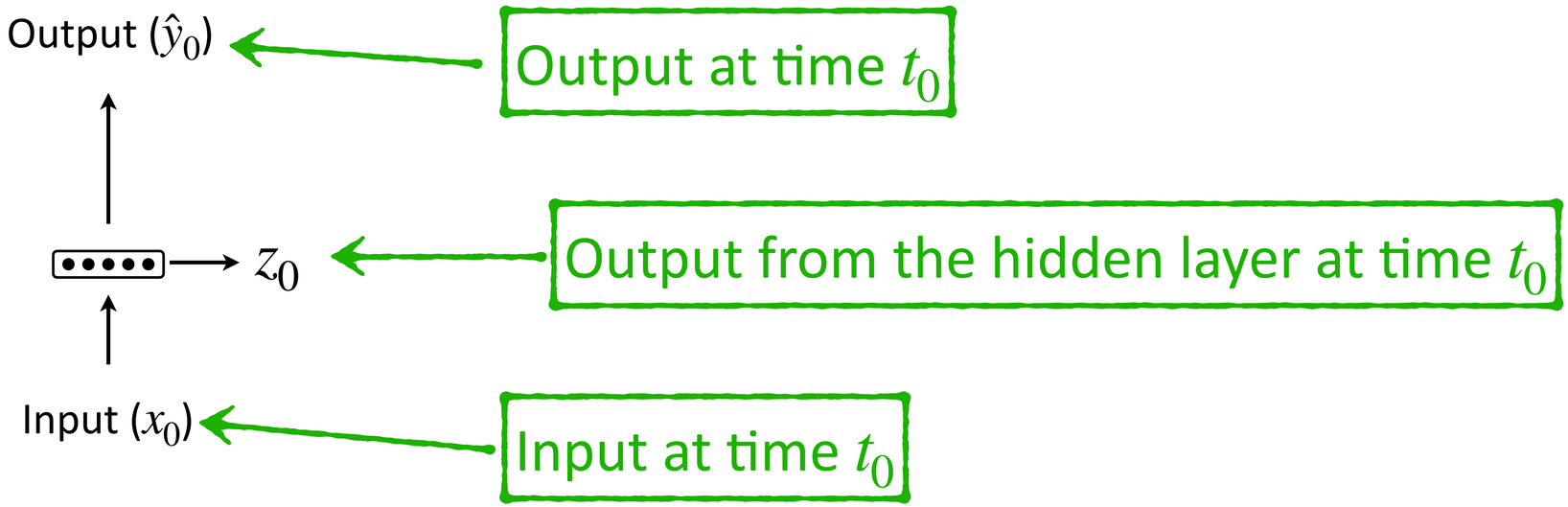
Making it a bit smaller so it fits on the slide…

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

First lets represent the RNN a bit differently…

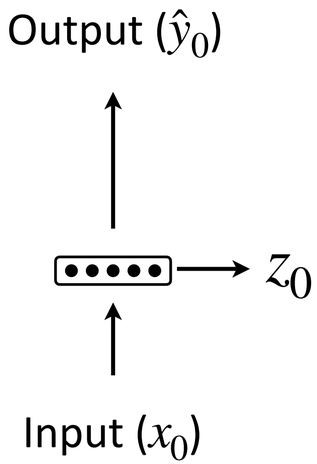


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

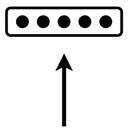
t_0

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Input (530)

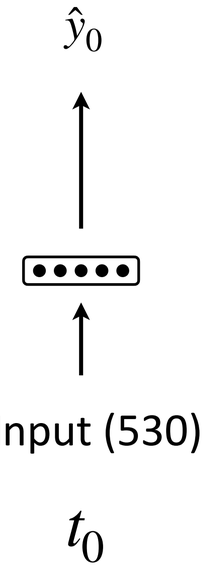
t_0

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

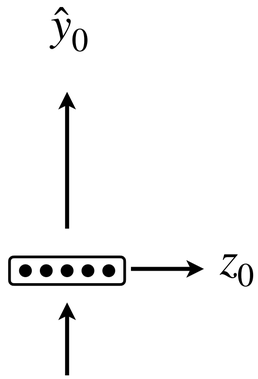


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Input (530)

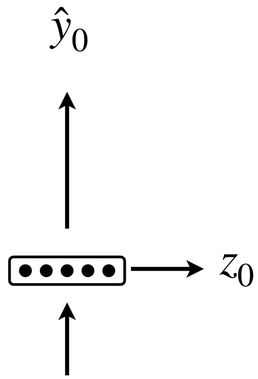
t_0

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Input (530)

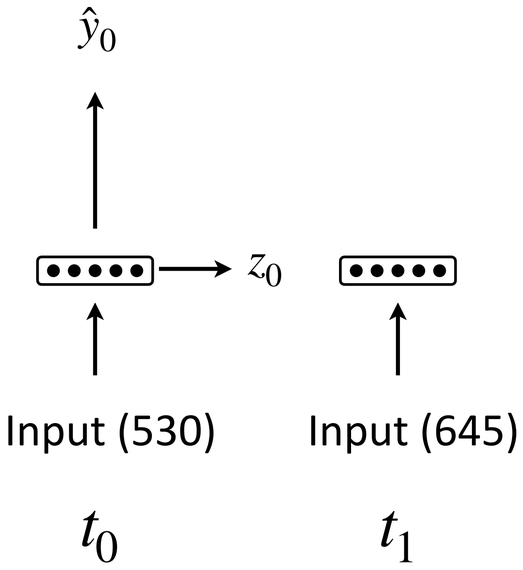
t_0 t_1

Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

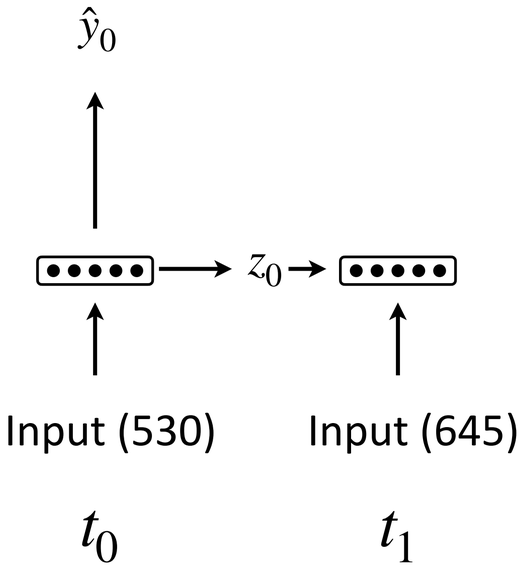


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

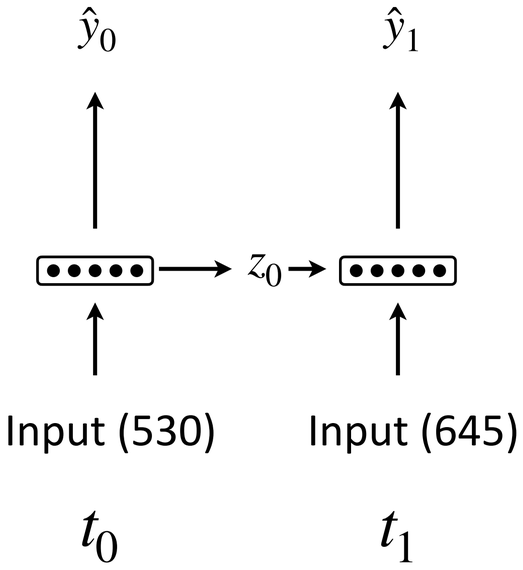


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

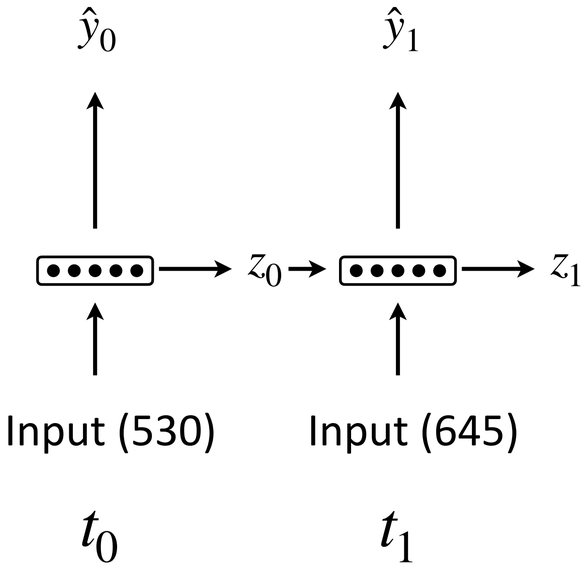


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

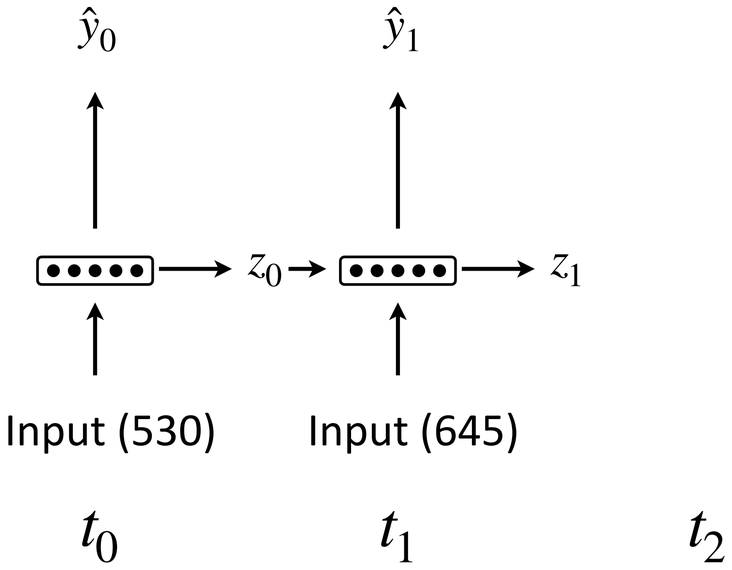


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

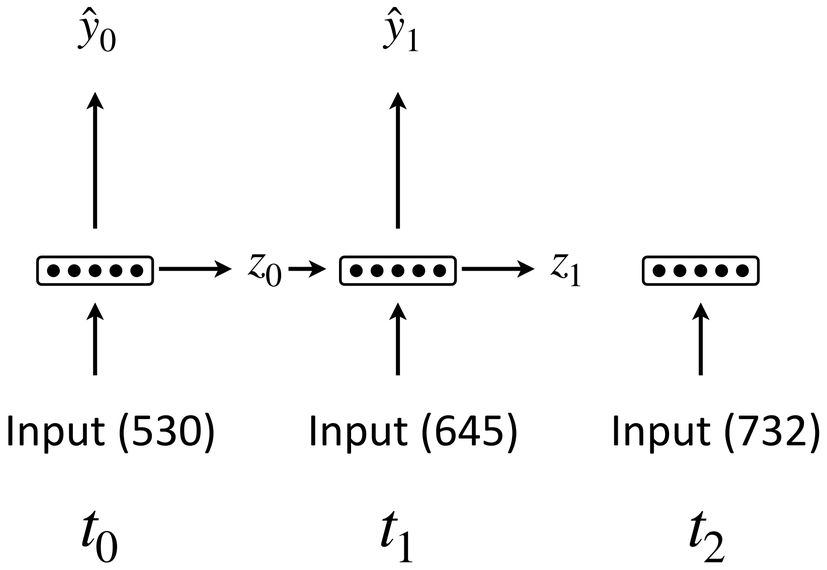


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

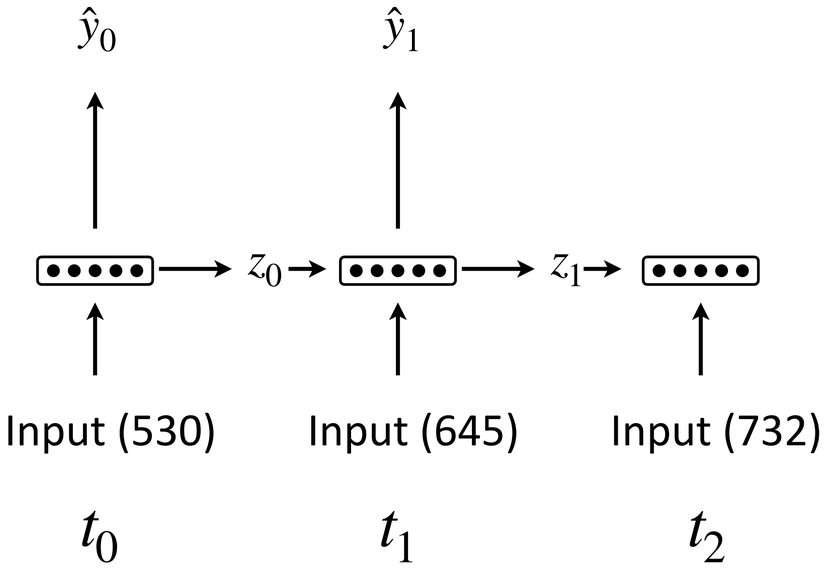


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

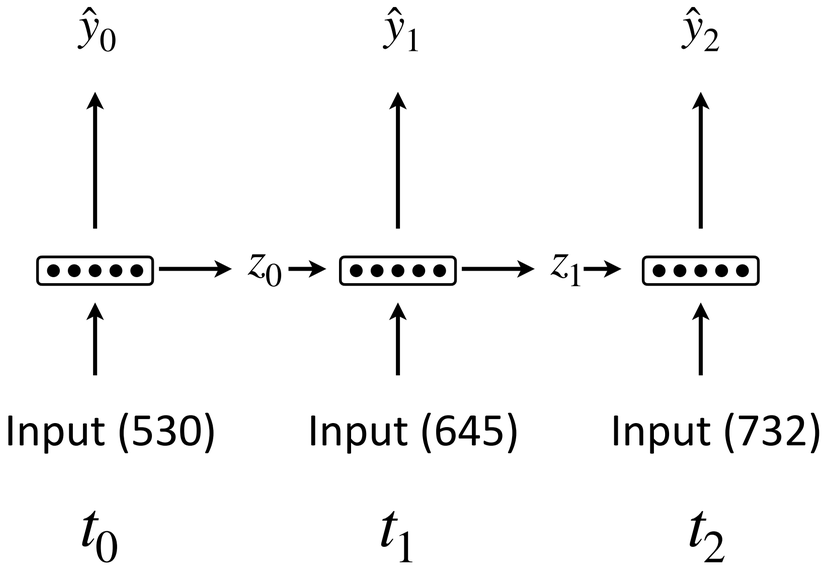


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

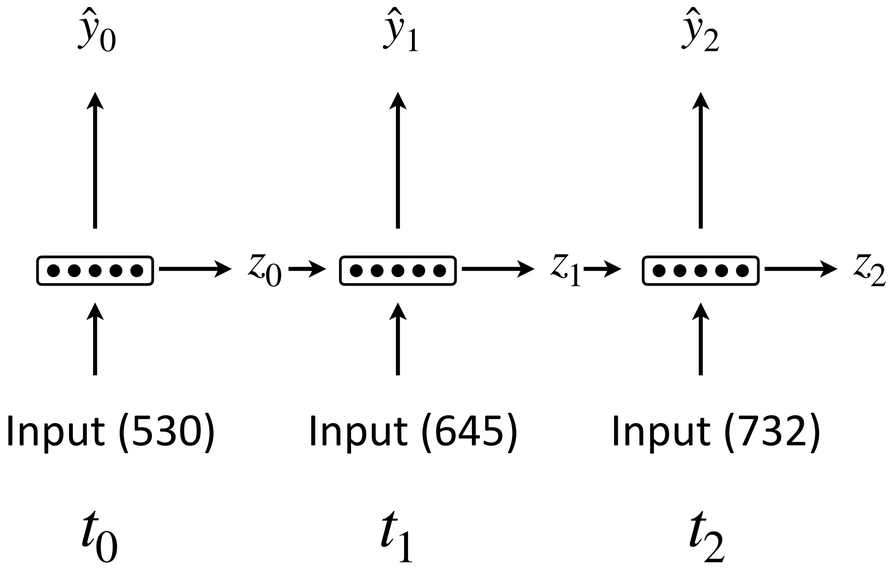


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

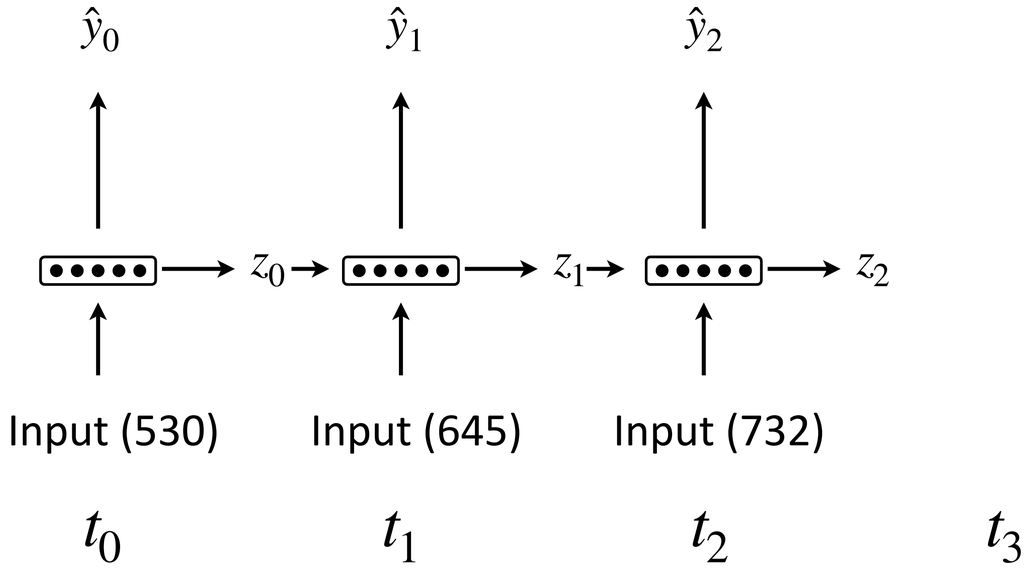


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

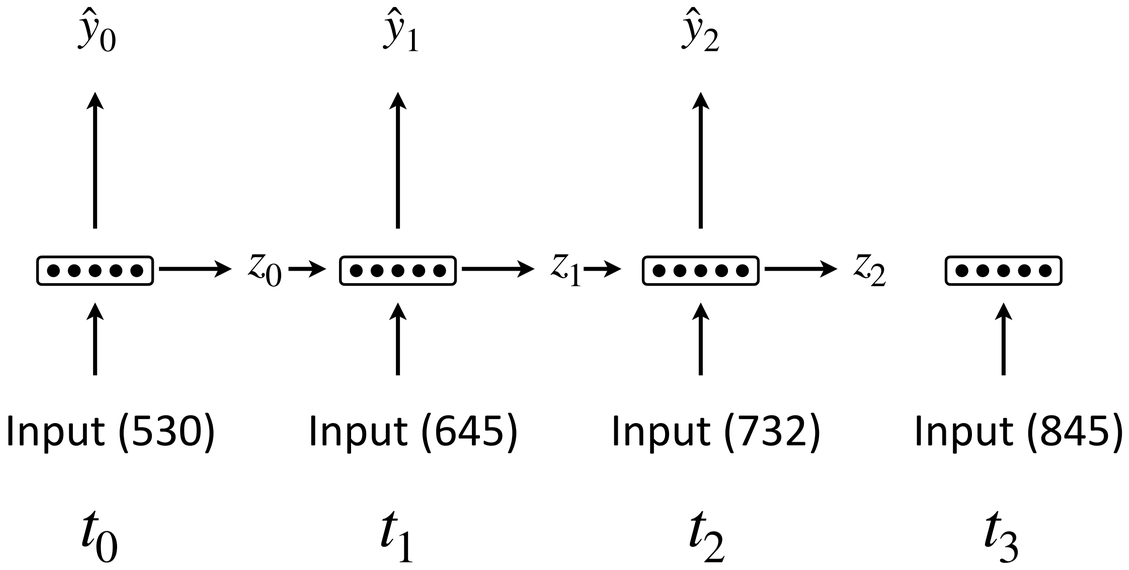


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

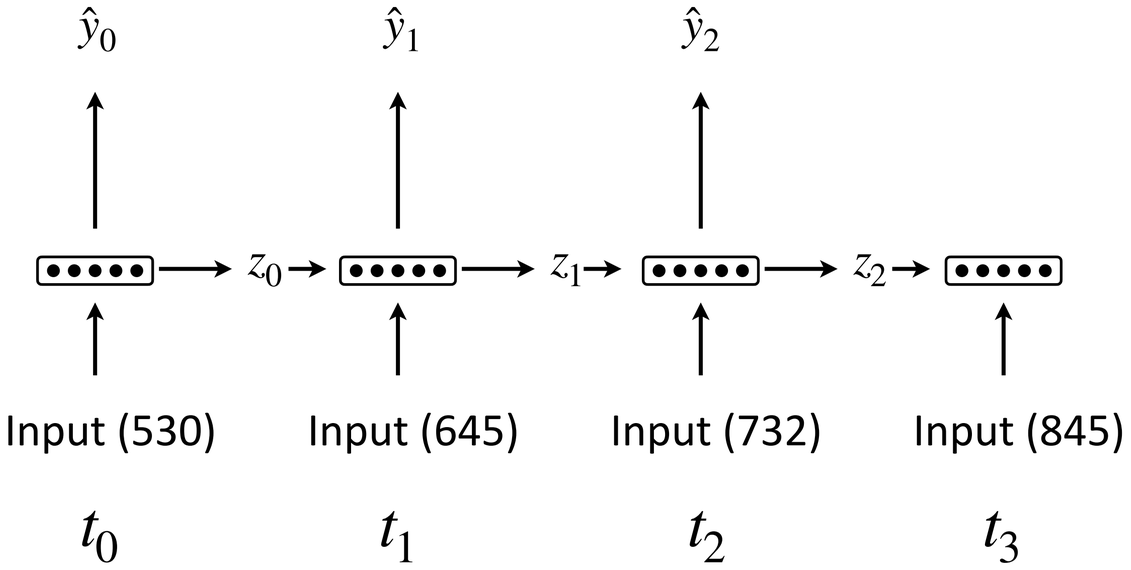


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

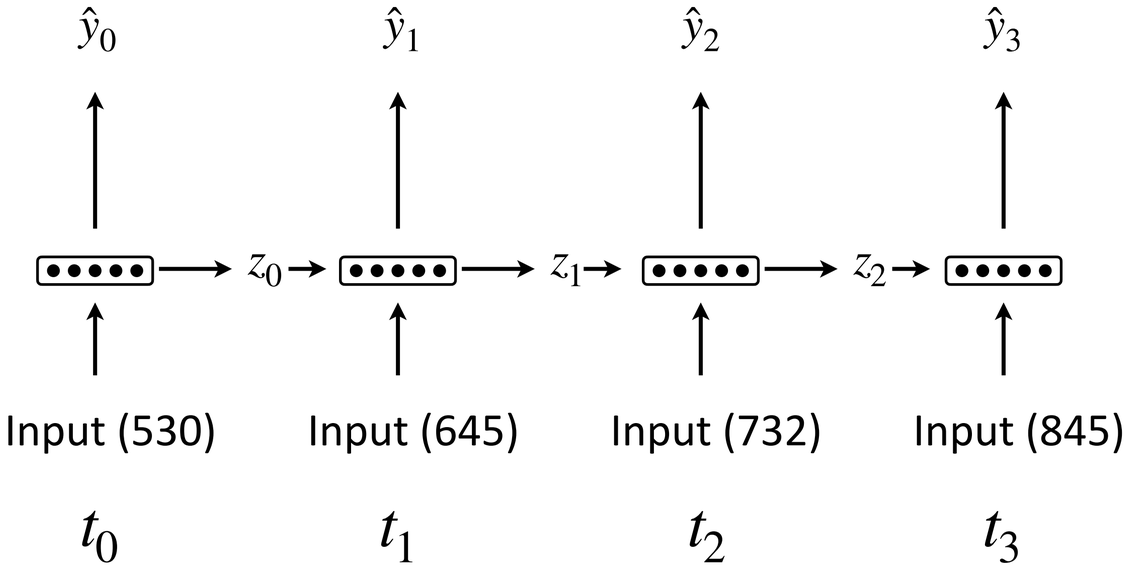


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

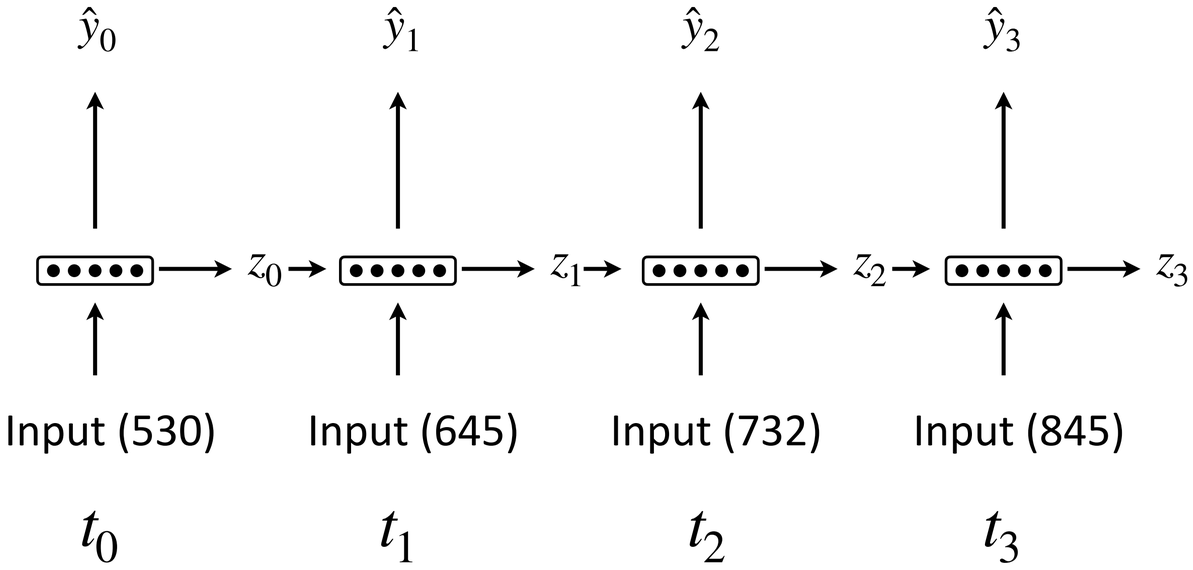


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

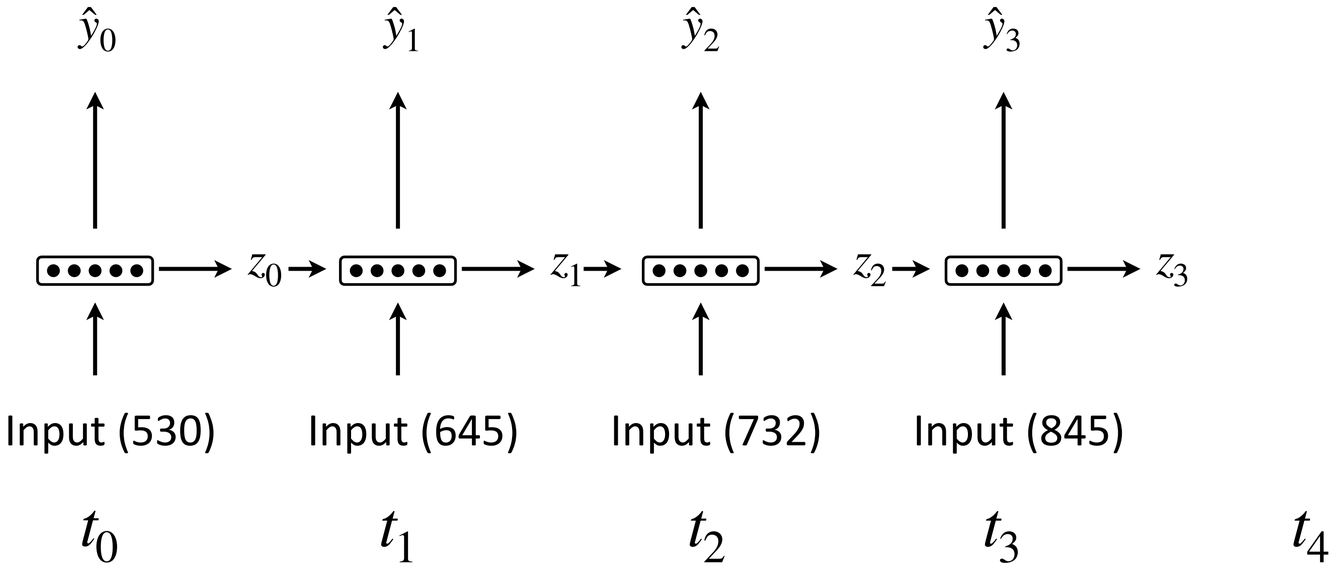


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

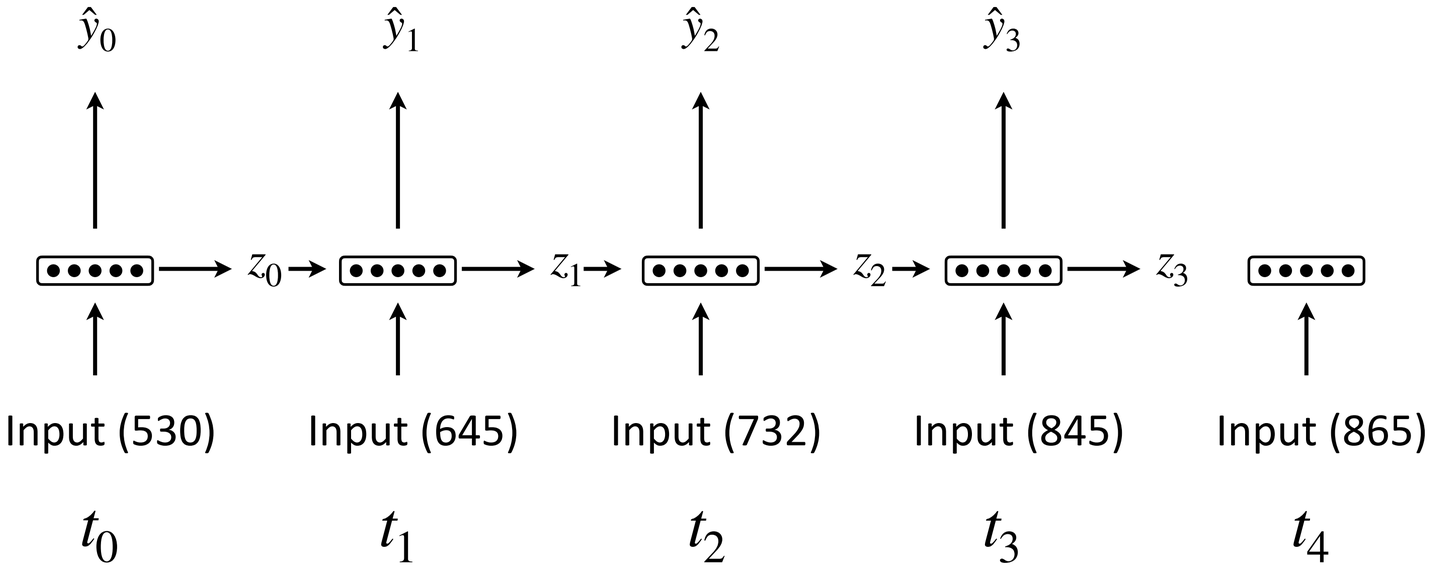


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

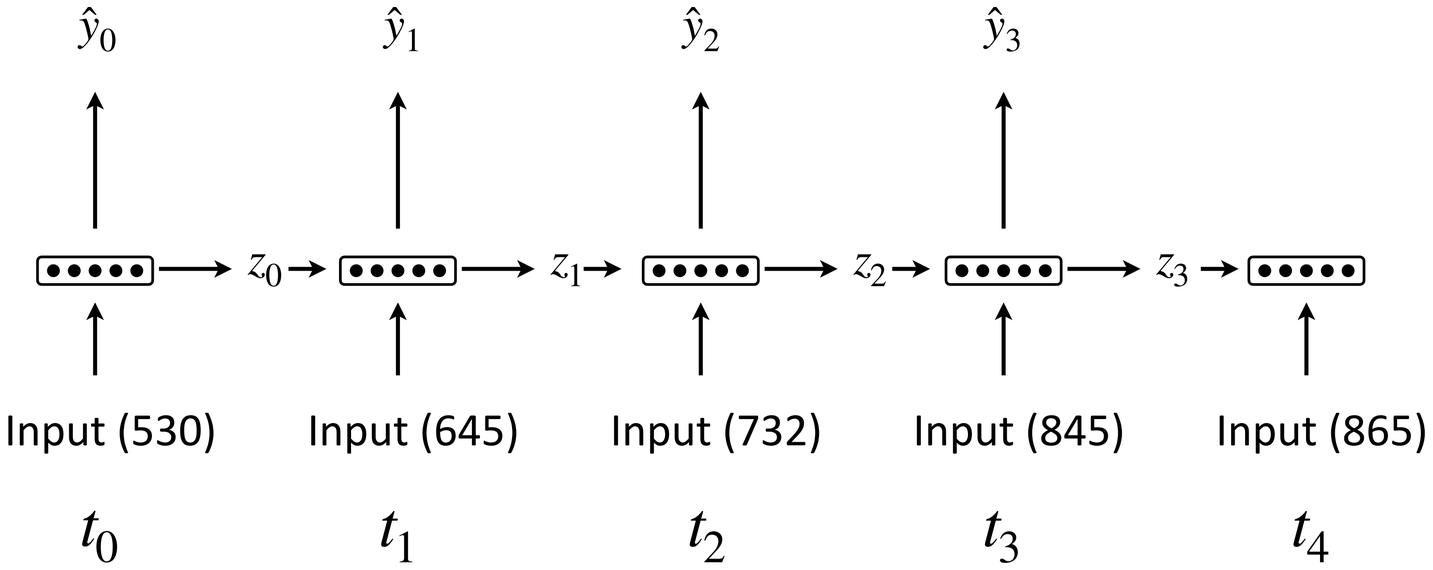


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

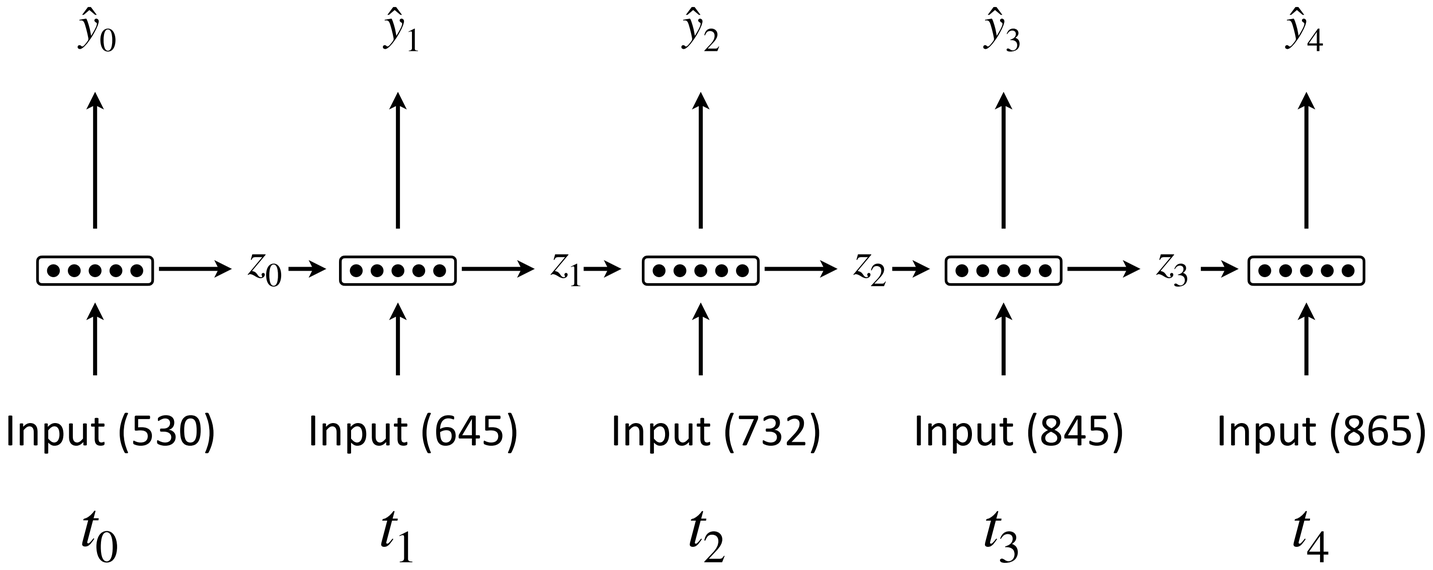


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

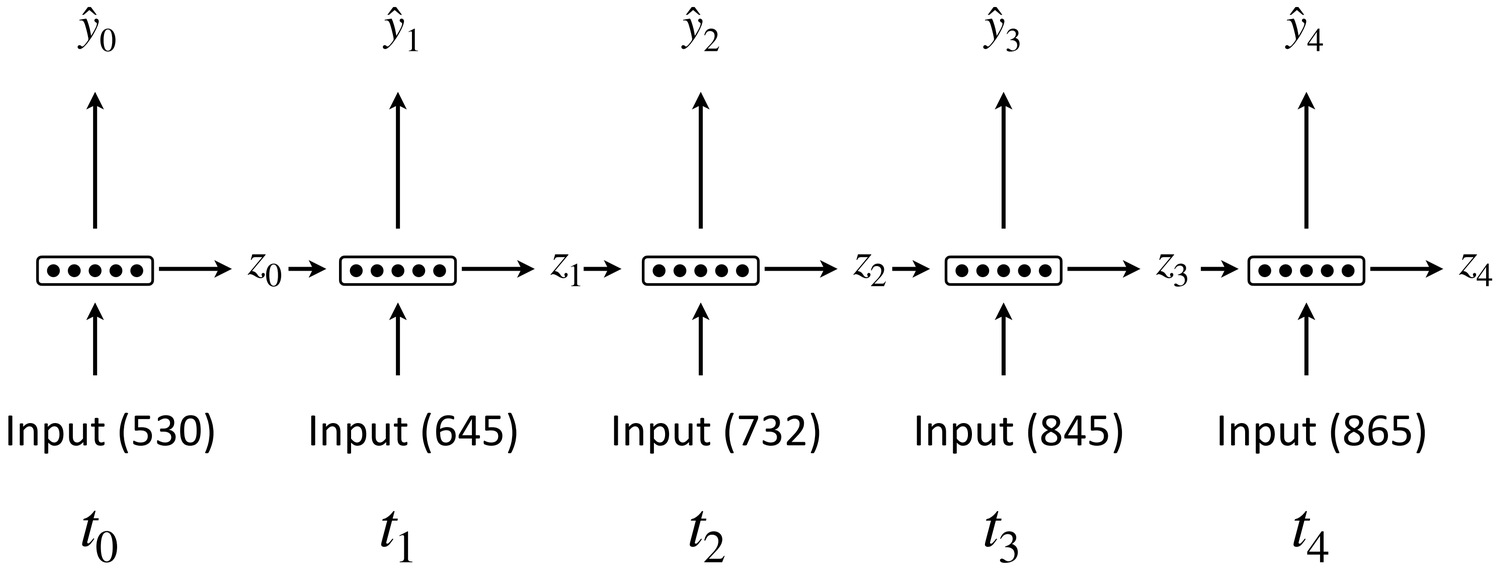


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

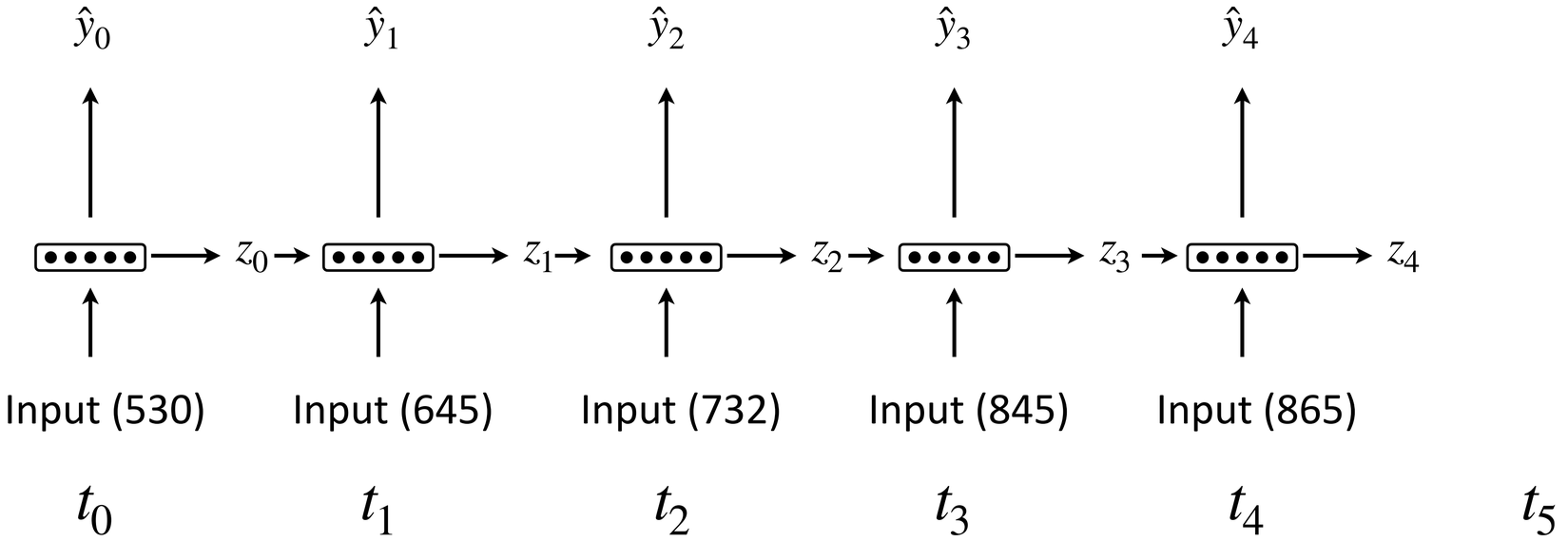


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

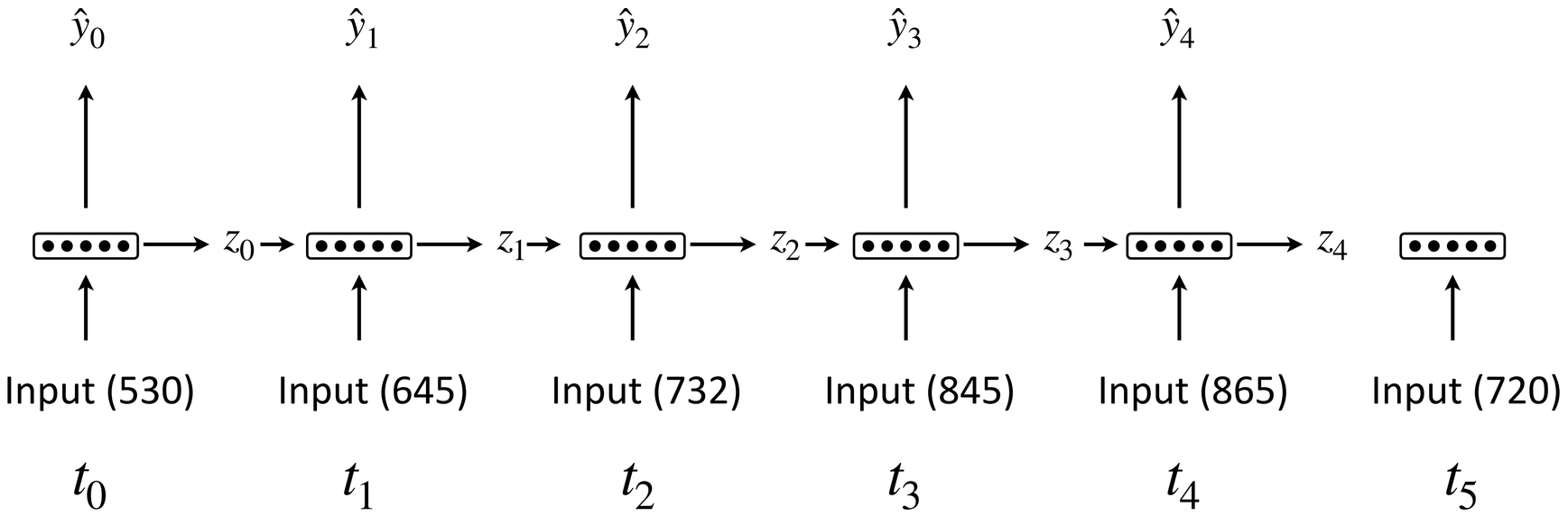


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let's unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

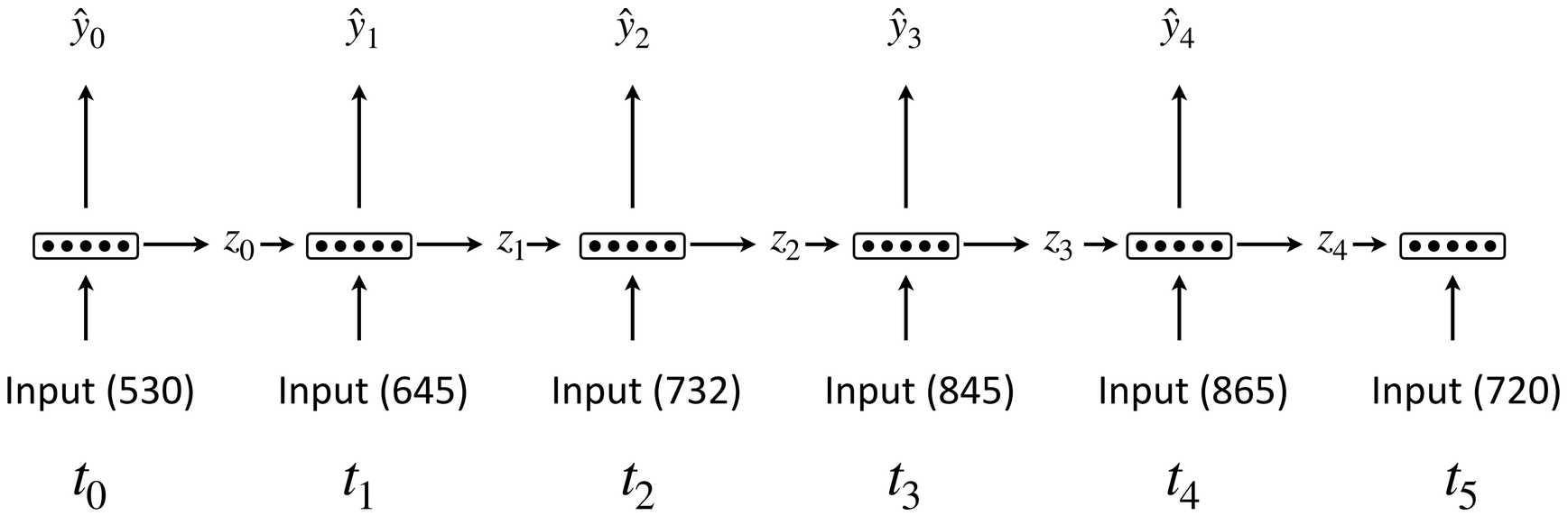


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

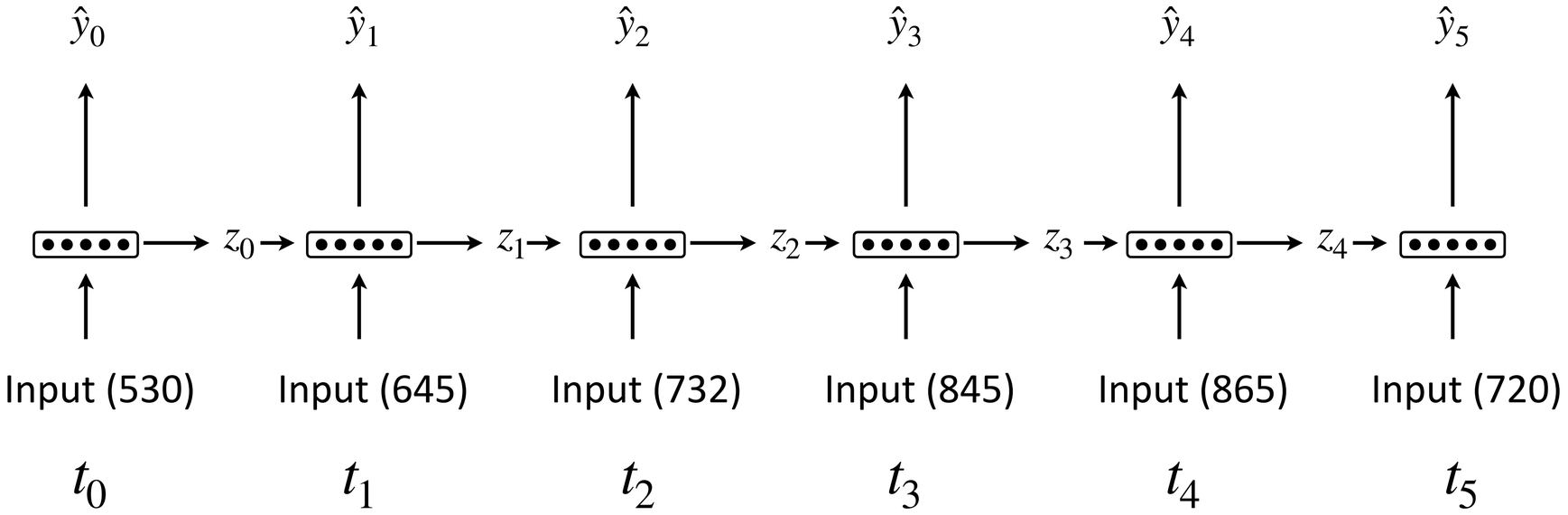


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

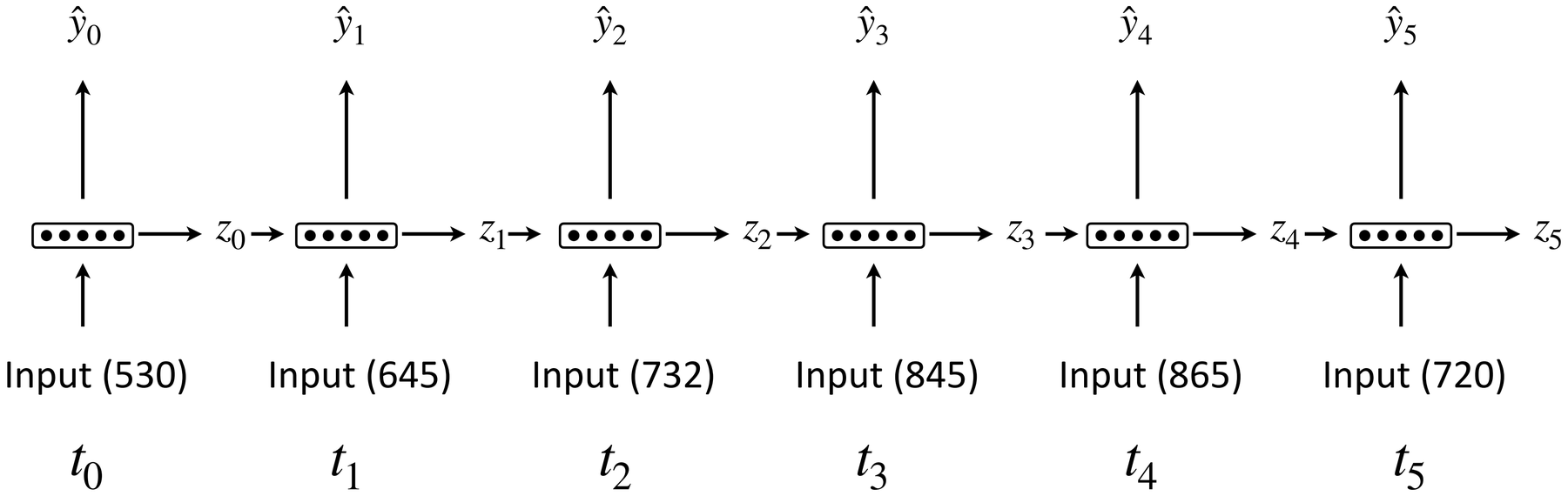


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

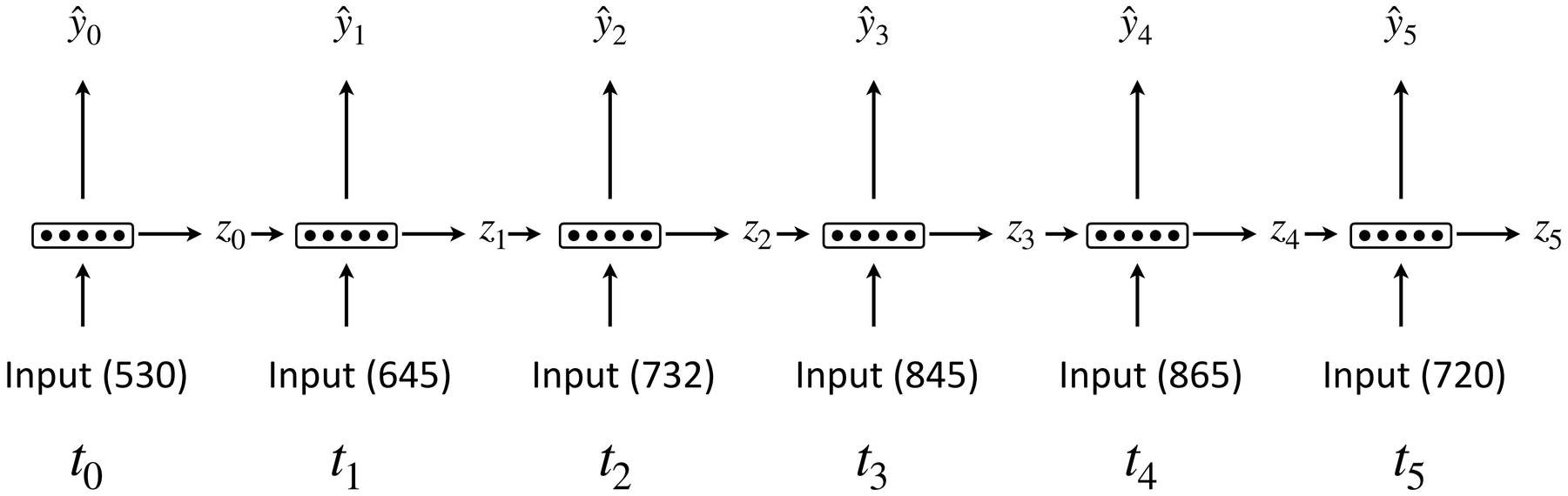


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

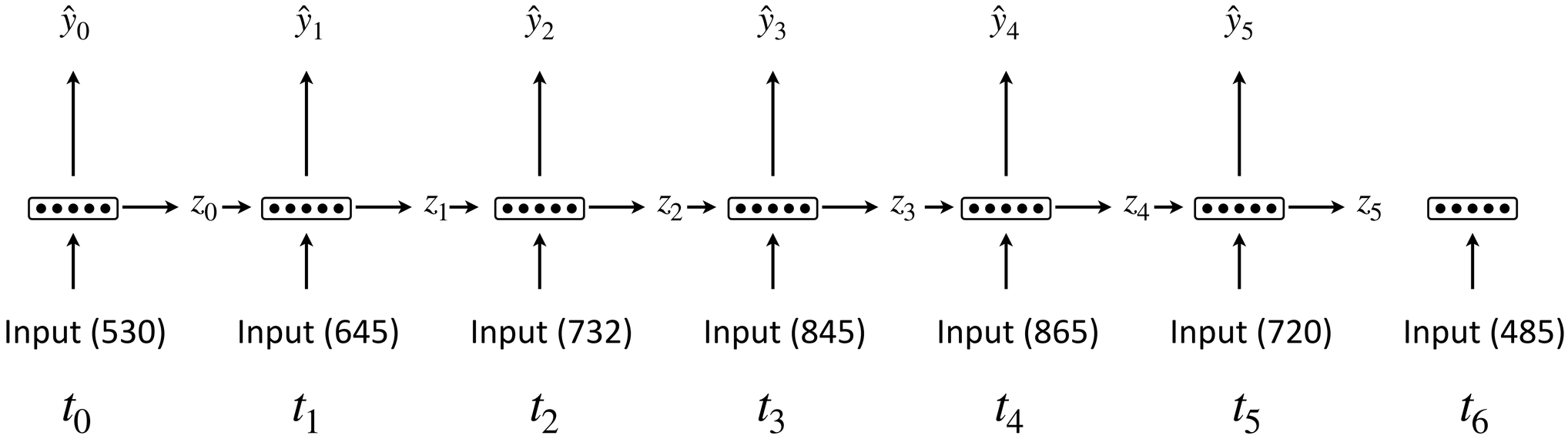


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

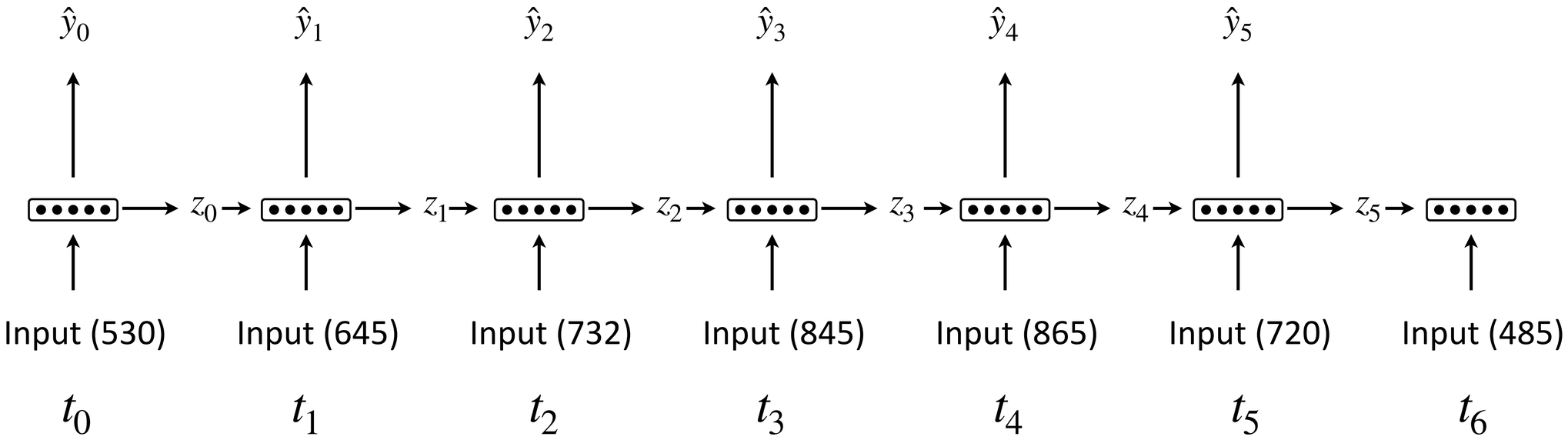


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

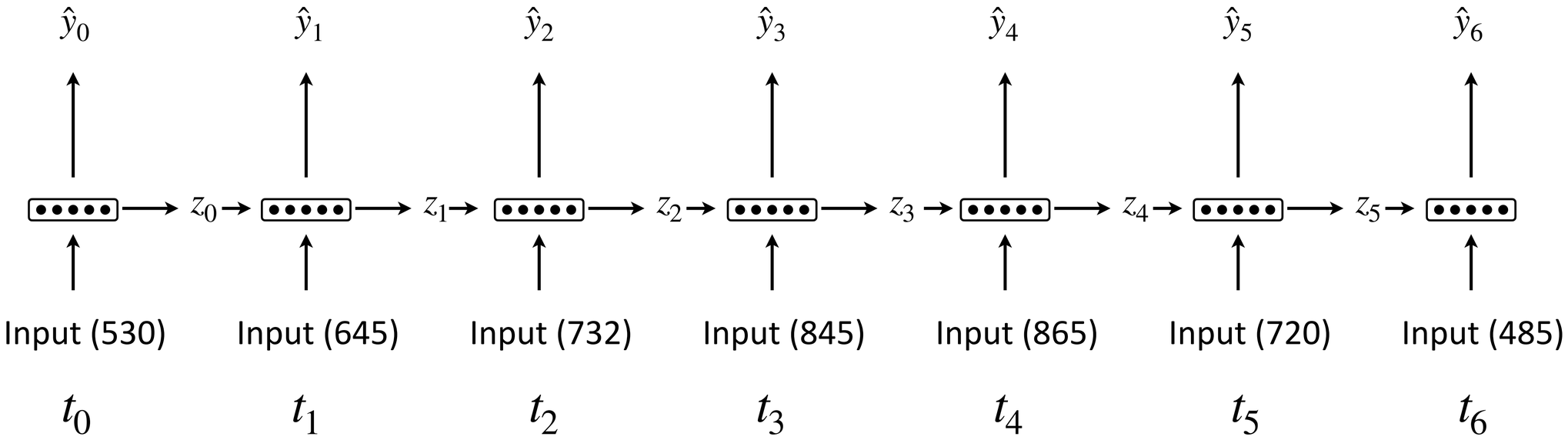


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

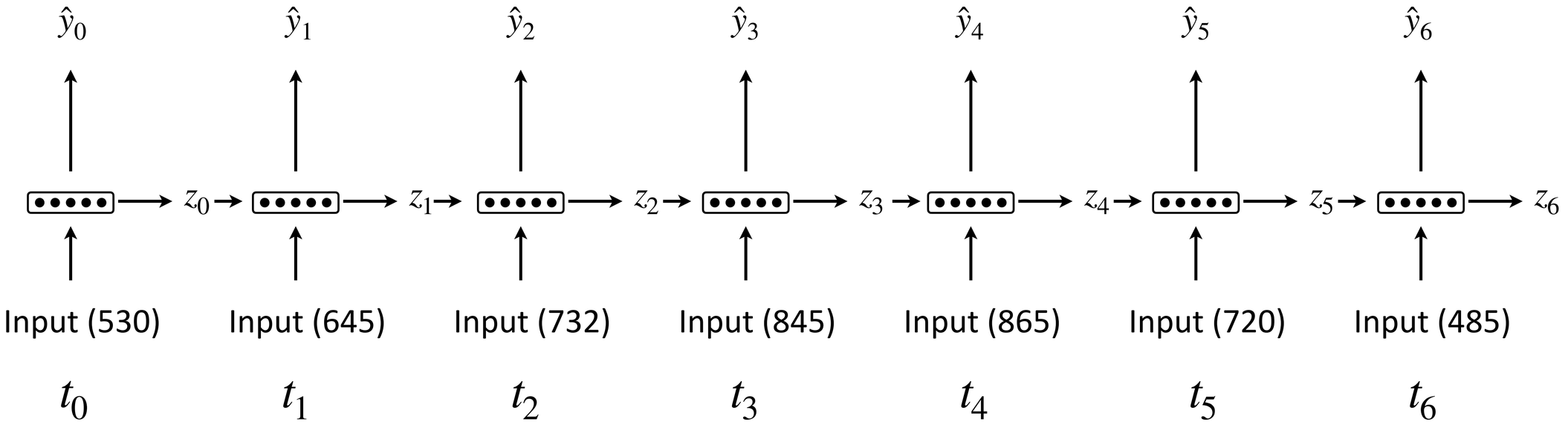


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

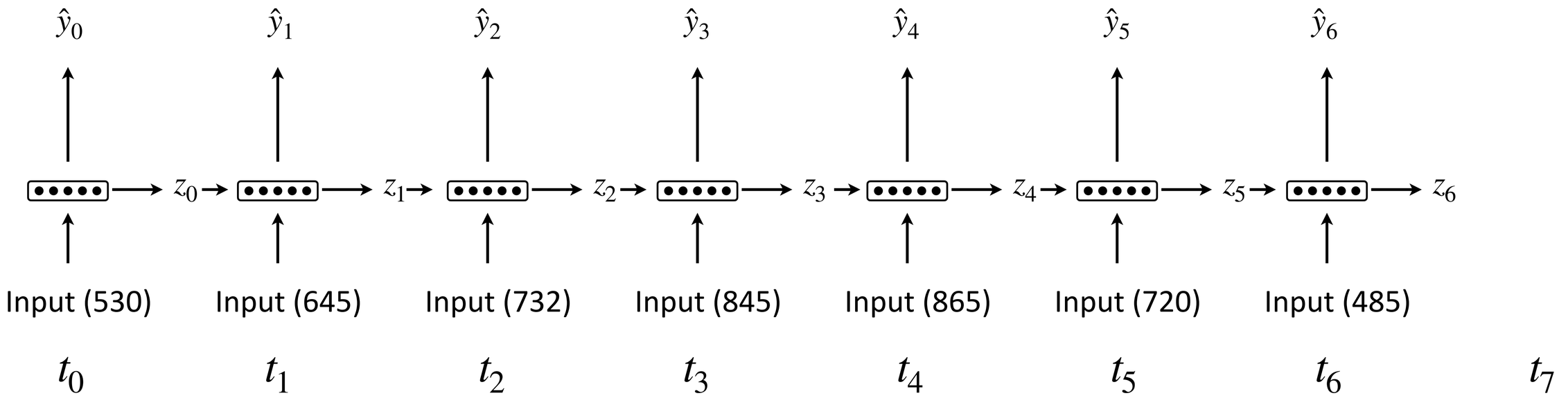


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

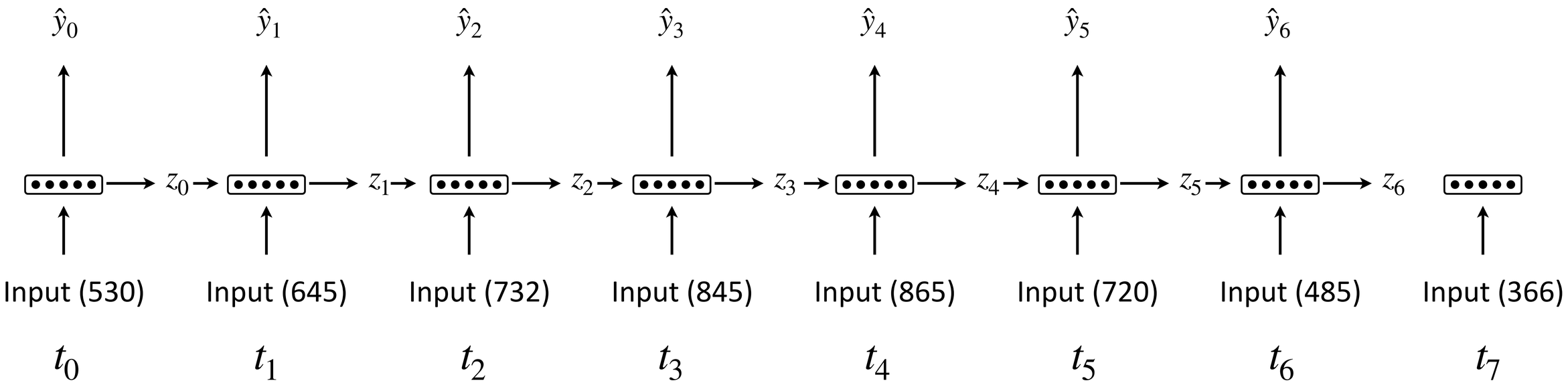


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

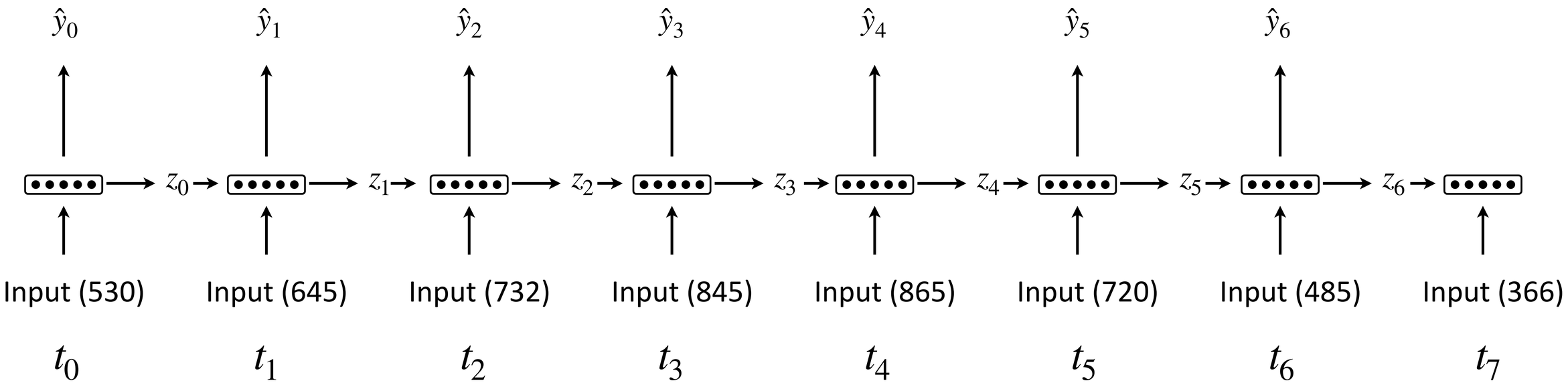


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

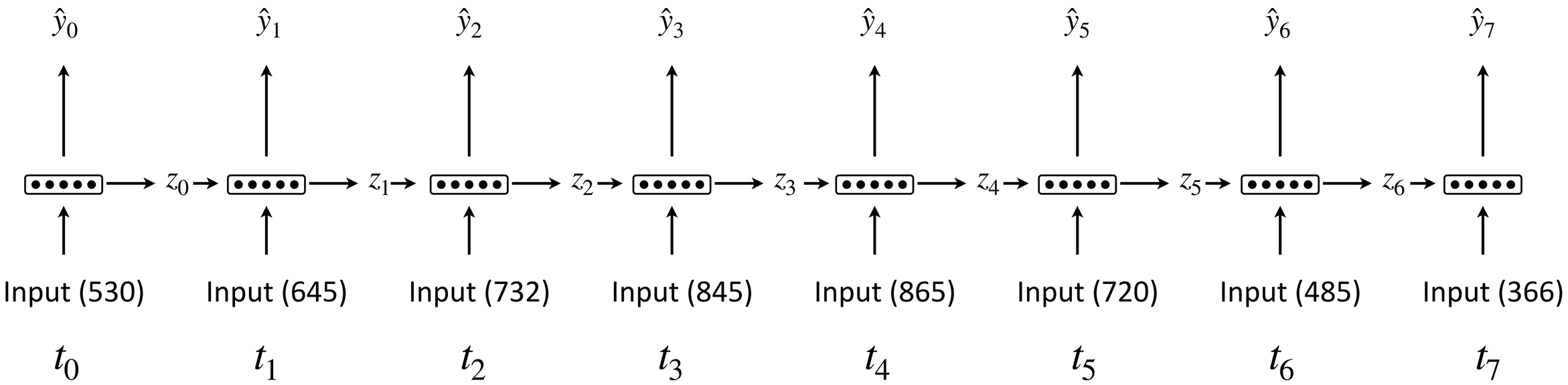


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

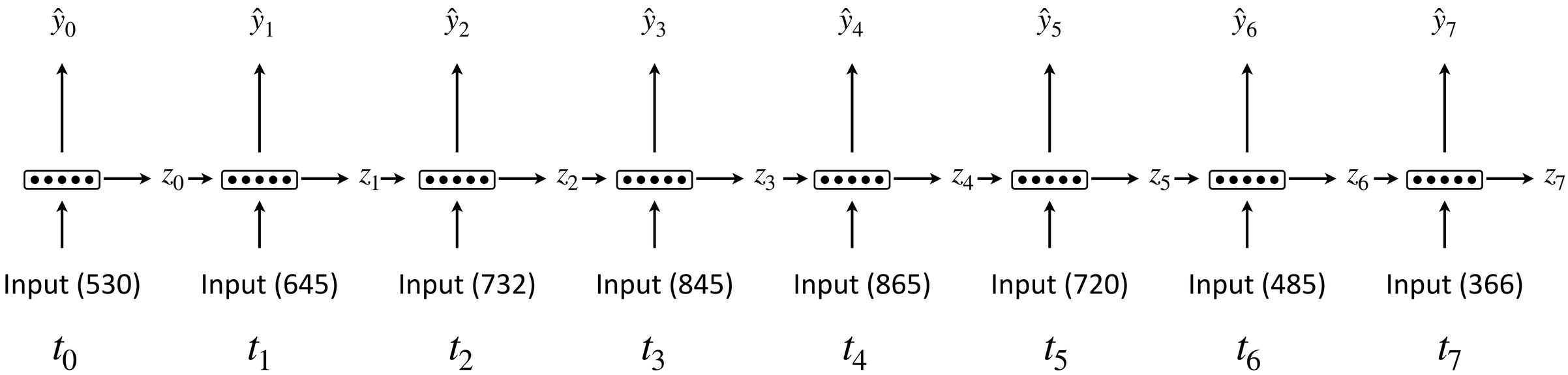


Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Let’s unroll the loop so its easier to see what happens at each time step

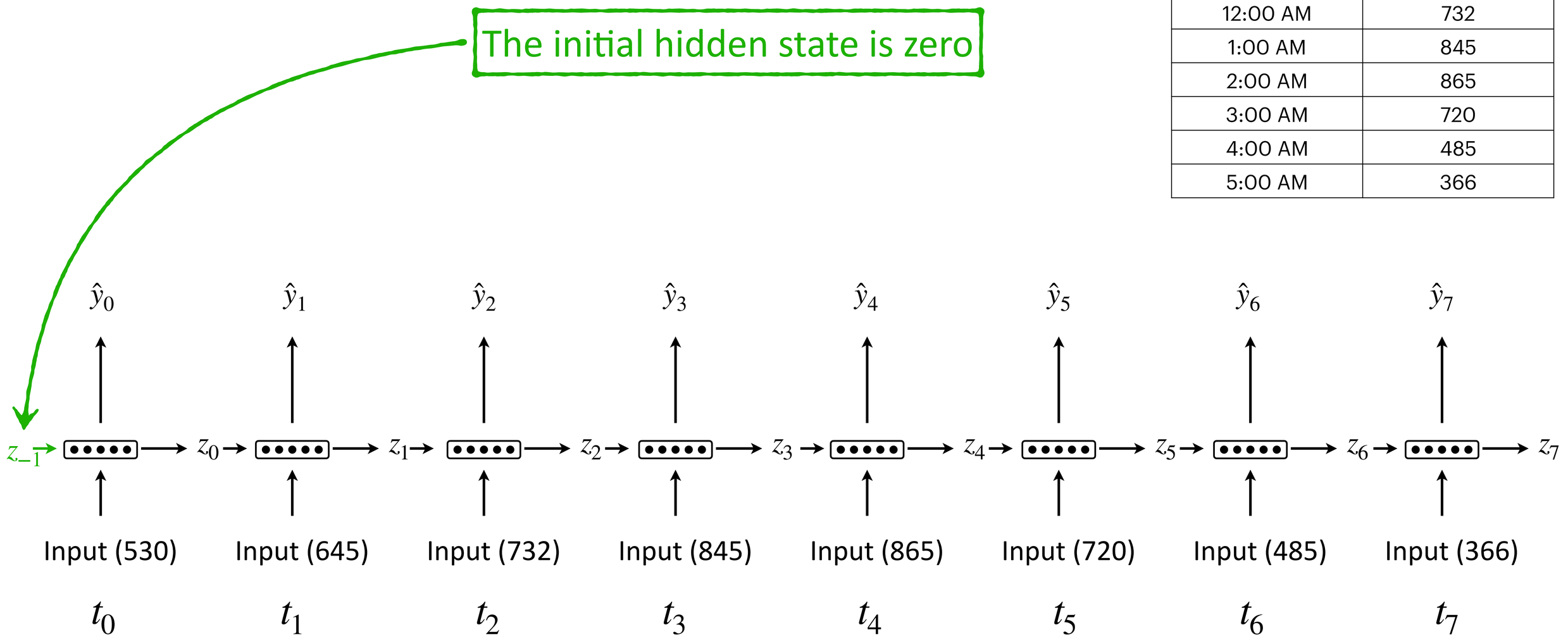
Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366



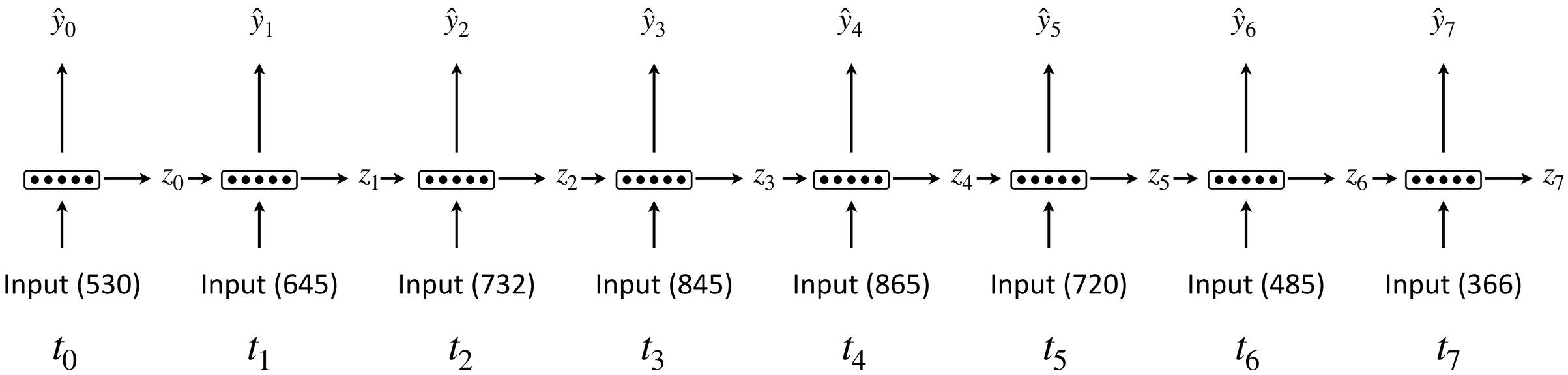
Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

The Output at each step is the predicted traffic at time $t + 1$

$\hat{y}_0$ Is the predicted traffic at t_1 (11:00 PM). Compare to actual value 645



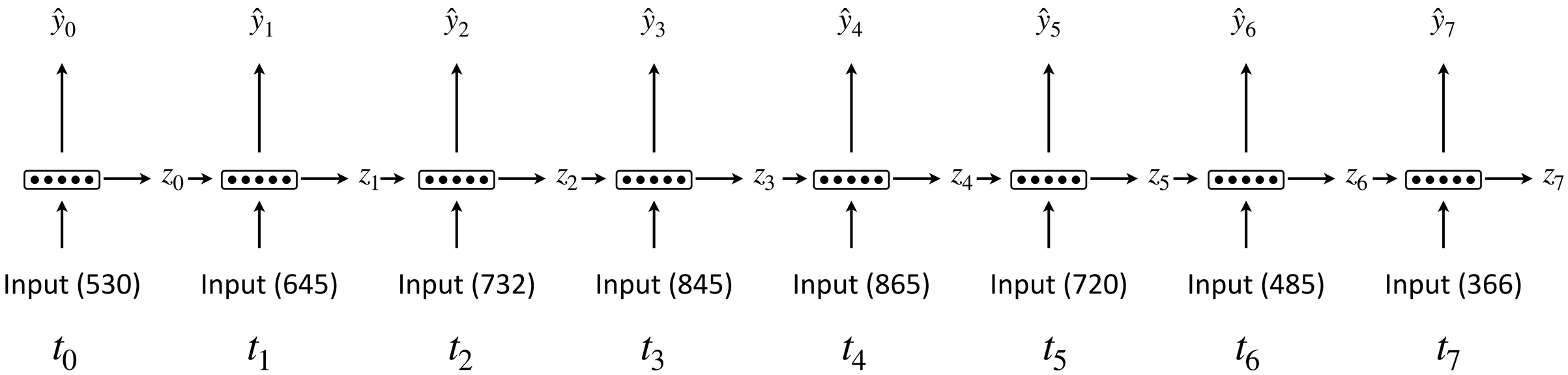
Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

The Output at each step is the predicted traffic at time $t + 1$

$\hat{y}_1$ Is the predicted traffic at t_2 (12:00 AM). Compare to actual value 732



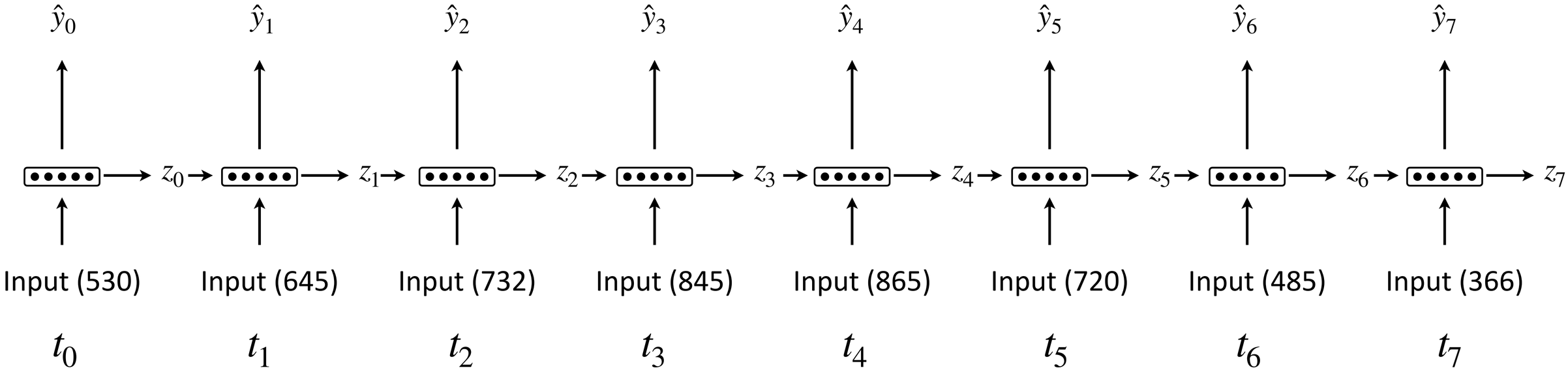
Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

The Output at each step is the predicted traffic at time $t + 1$

Original Question:
Can we predict the traffic at 6:00 AM?



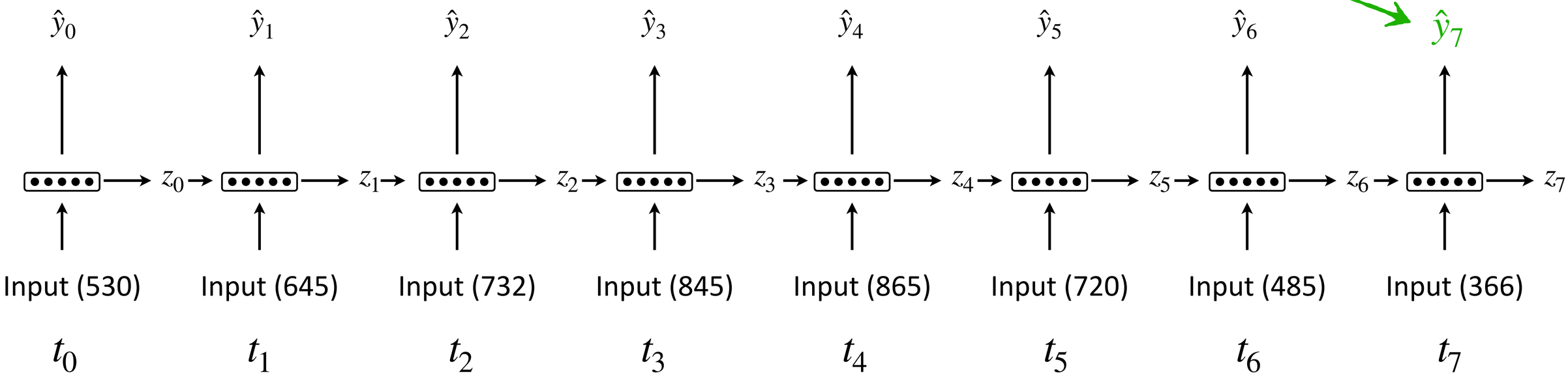
Recurrent Neural Networks

Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

The Output at each step is the predicted traffic at time $t + 1$

Original Question:
Can we predict the traffic at 6:00 AM?



Recurrent Neural Networks

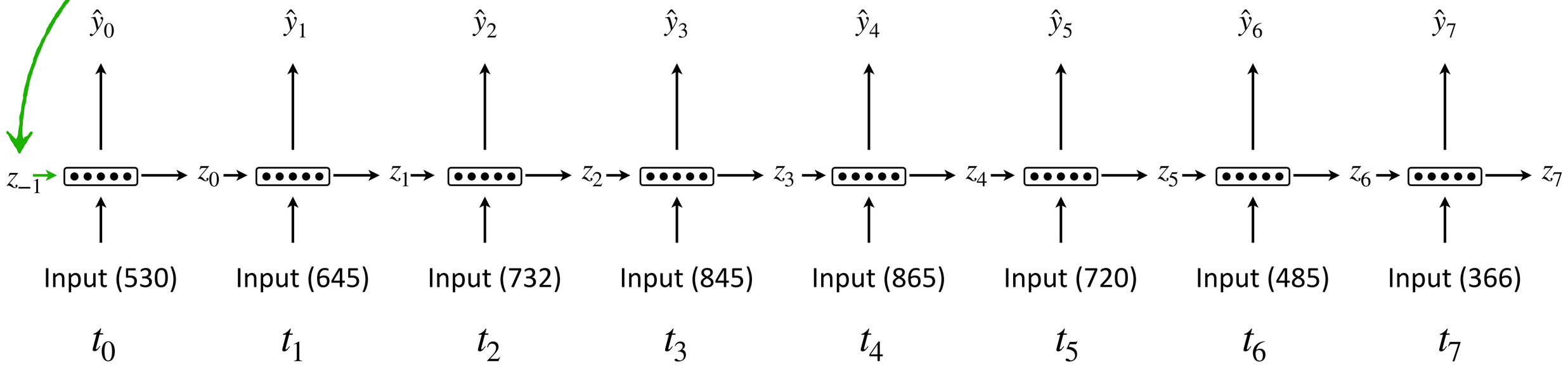
Recurrent Neural Networks can model inputs that have a temporal dependency and ordering

Hour	Traffic
10:00 PM	530
11:00 PM	645
12:00 AM	732
1:00 AM	845
2:00 AM	865
3:00 AM	720
4:00 AM	485
5:00 AM	366

The Output at each step is the predicted traffic at time $t + 1$

The initial hidden state is zero…

… and we discard the predictions $\hat{y}_0$ through $\hat{y}_6$ during inference



Recurrent Neural Networks

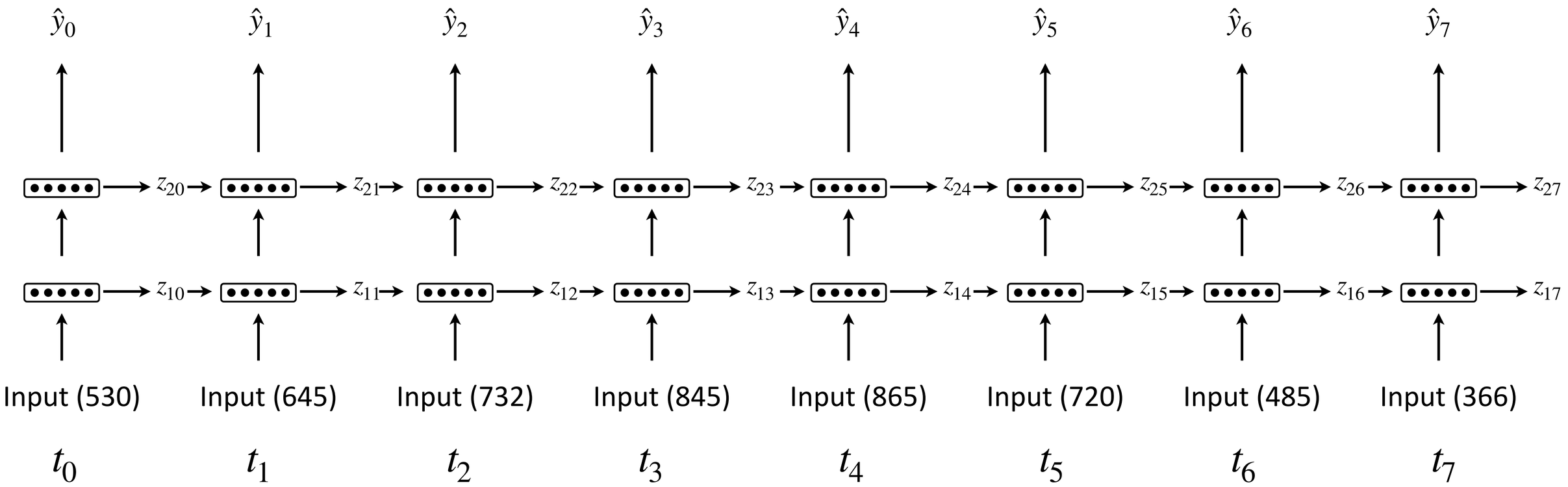
What if there are multiple hidden layers?

Multiple hidden layers, result in multiple RNN “cells"

Recurrent Neural Networks

Recurrent Neural Networks with multiple hidden layers, means multiple RNN “cells”.

The Output from a given layer at time t is passed as input to the same layer at time $t + 1$



Recurrent Neural Networks

How do we represent RNNs mathematically?

We compute the output of a hidden layer by:

1. Multiplying the current input with weights (W_i)
2. Multiplying the previous hidden state with weights (W_{hi})
3. Adding both together with a bias
4. Applying an activation function (typically $tanh$)

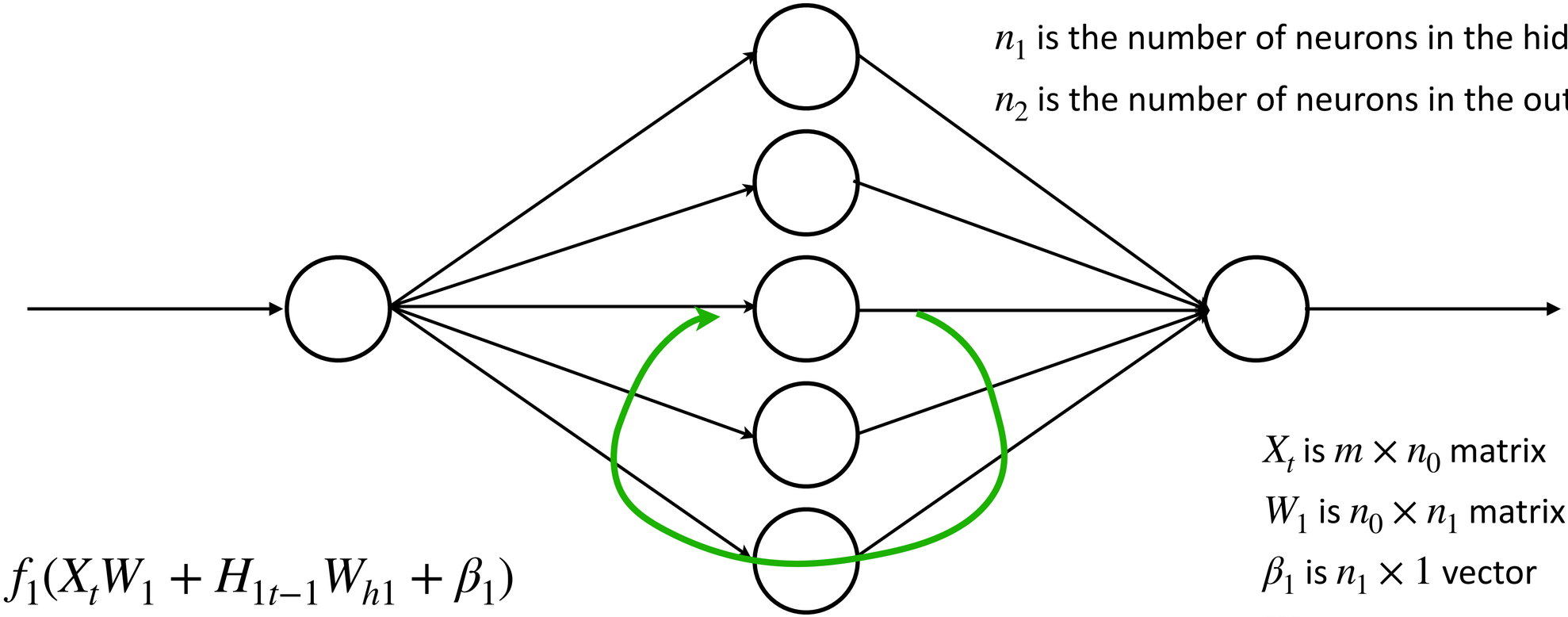
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

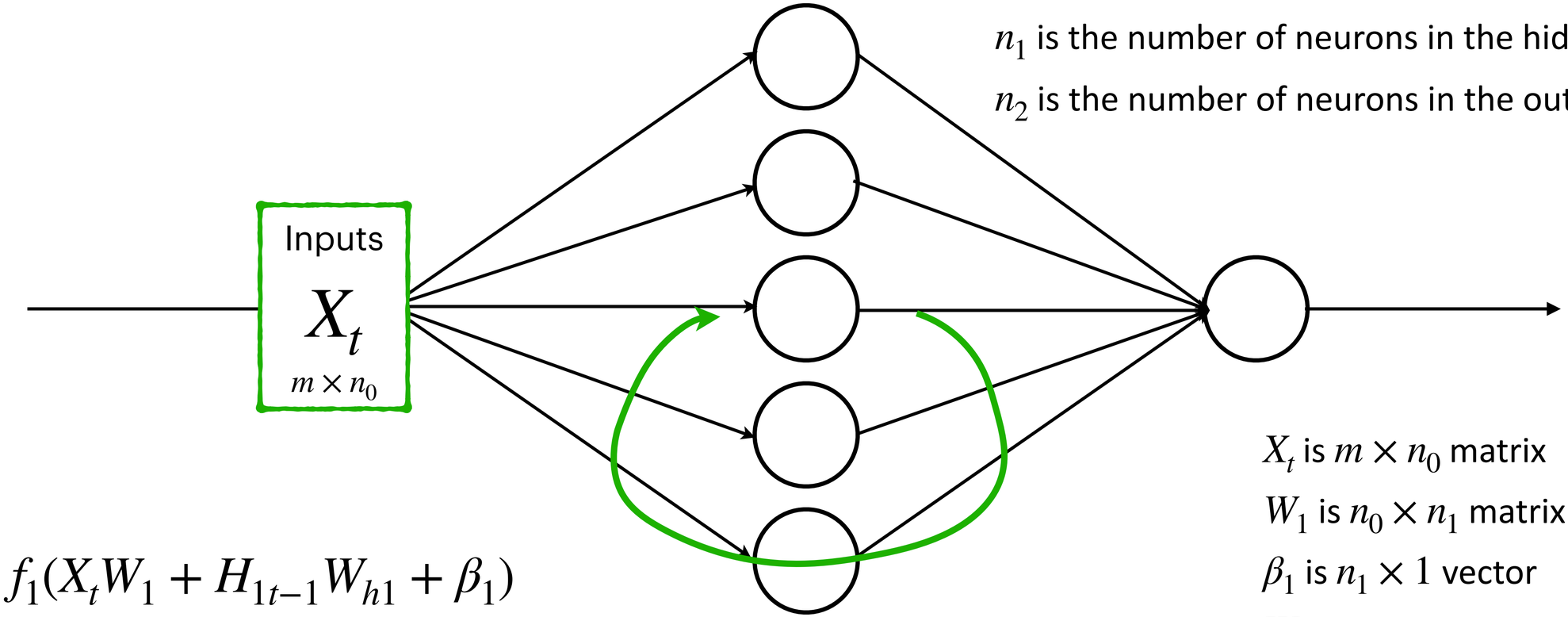
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

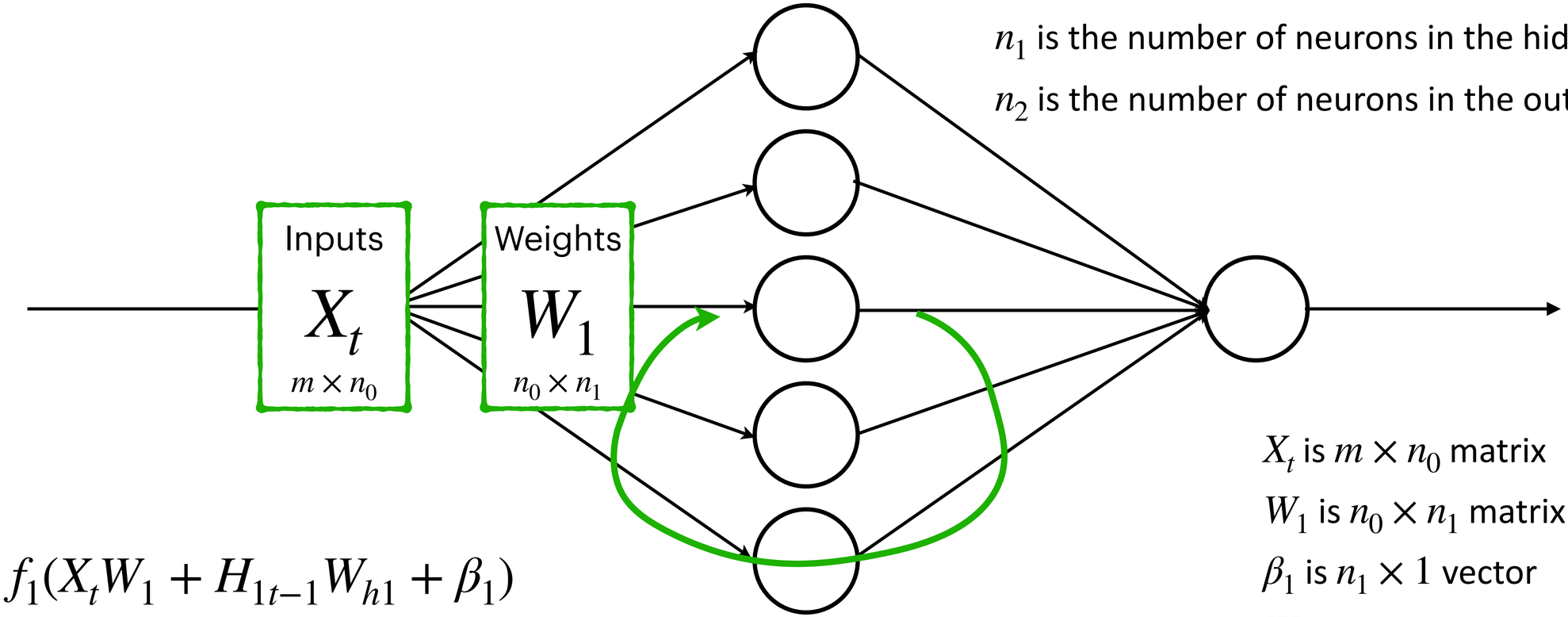
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

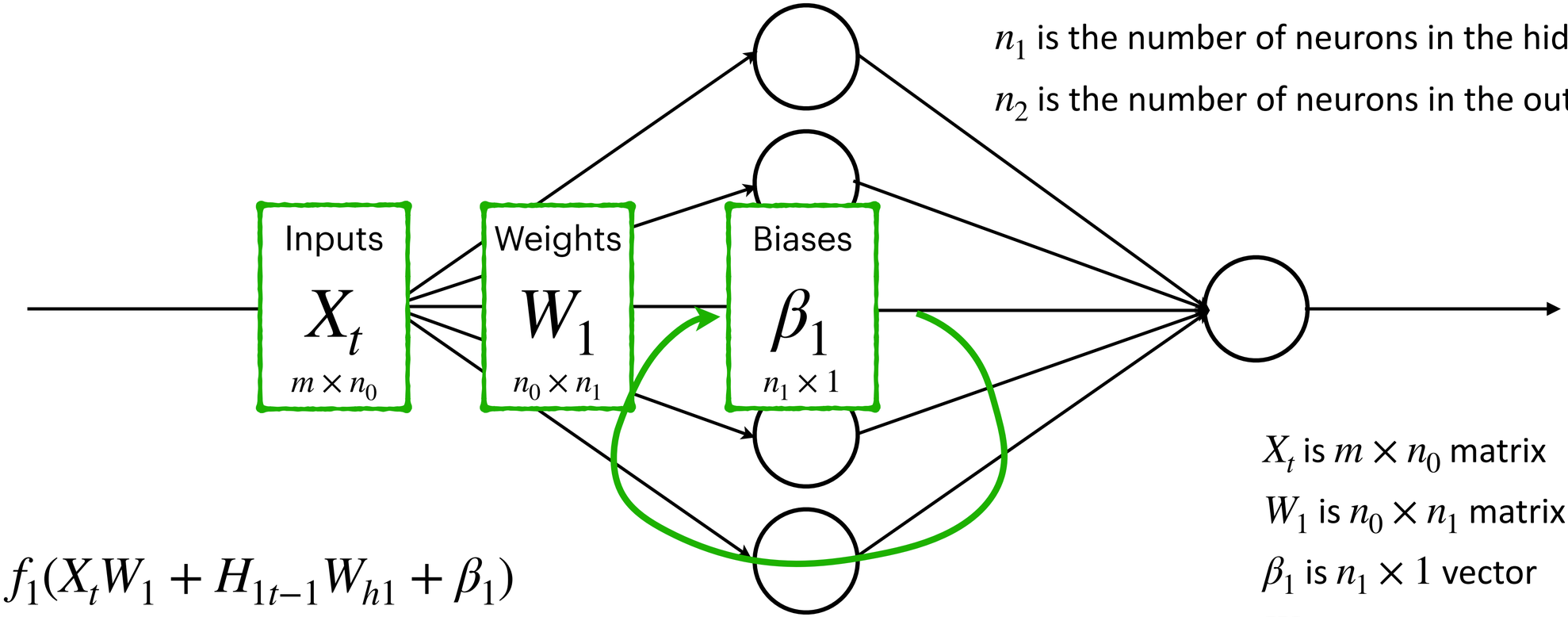
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

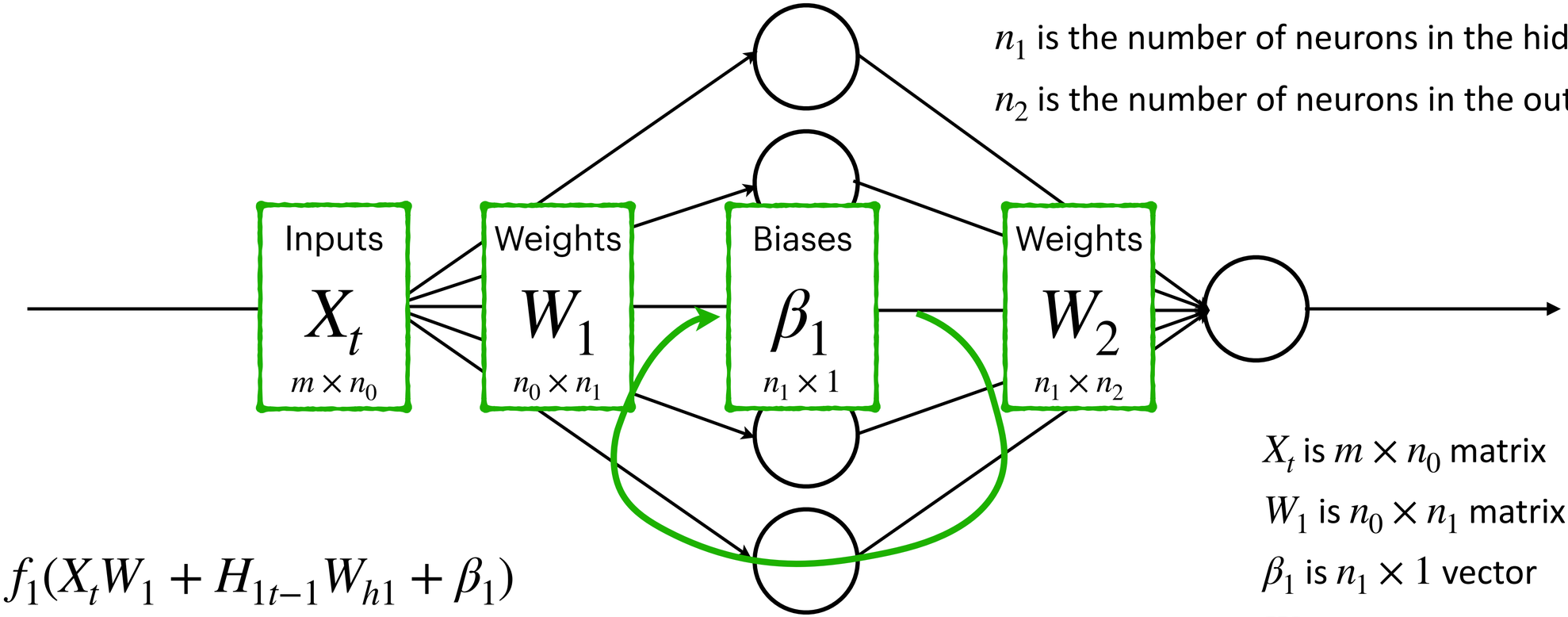
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

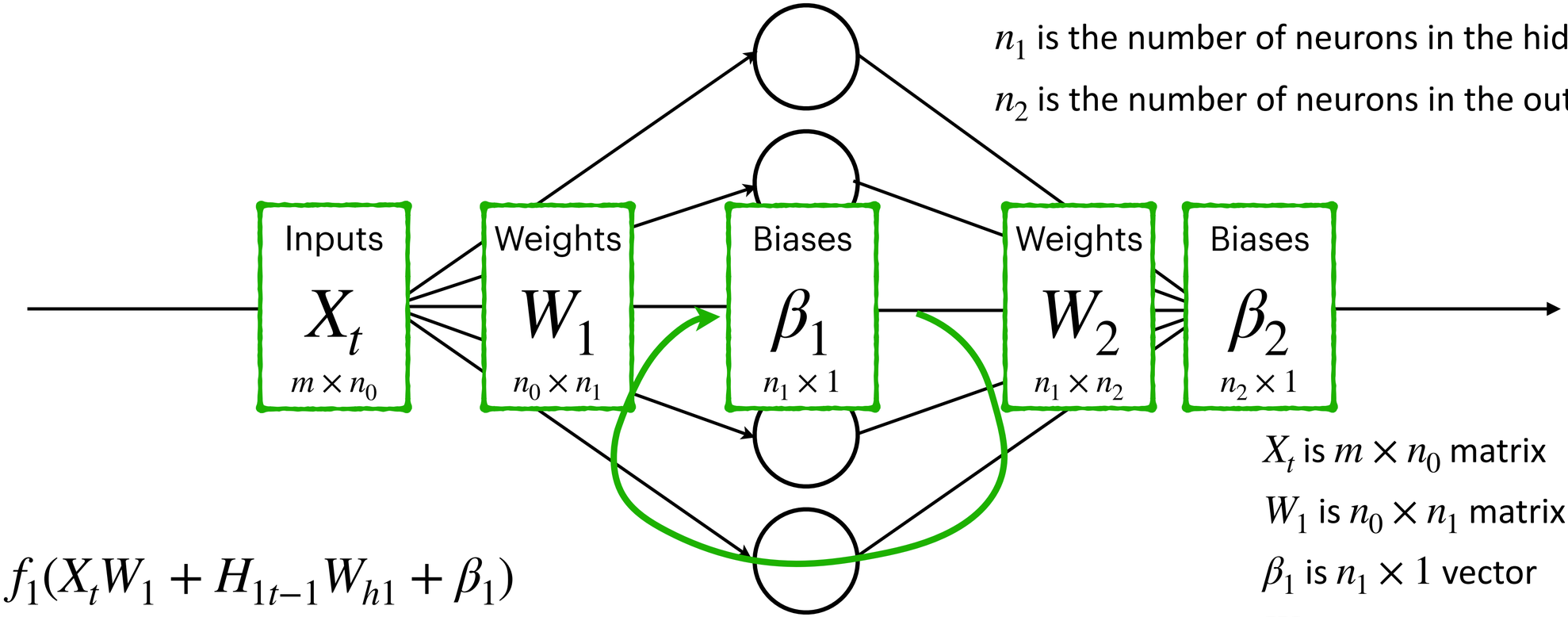
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

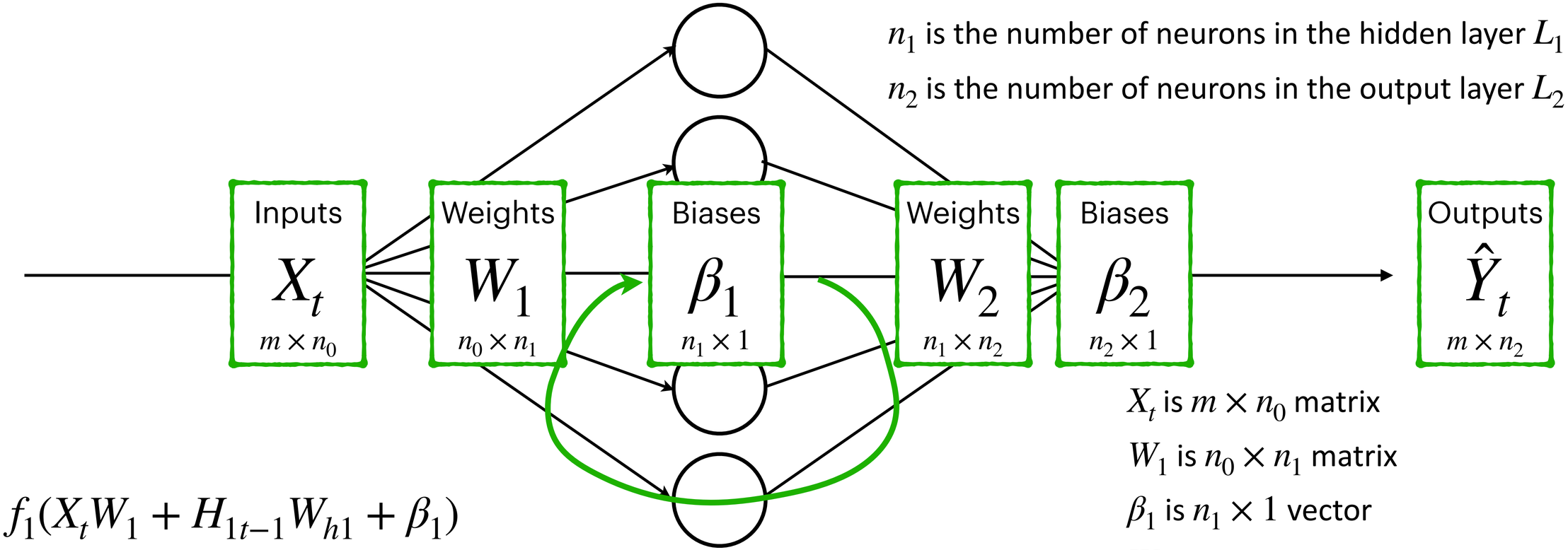
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

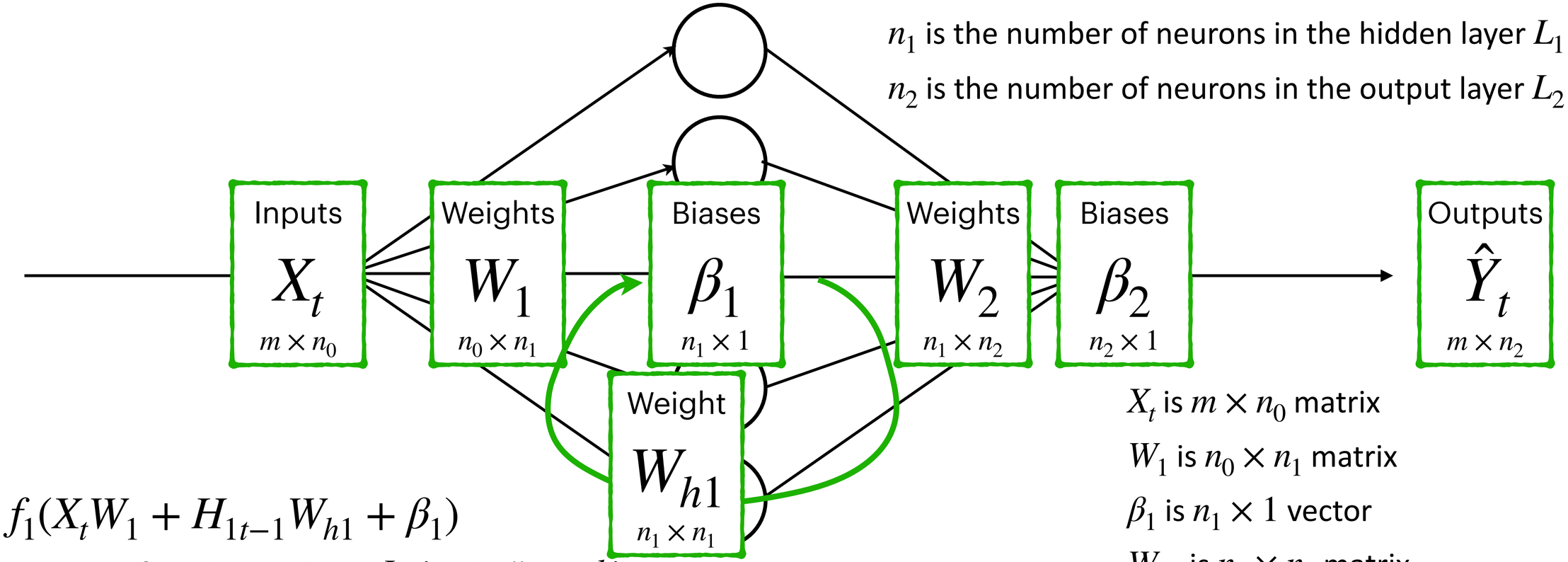
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

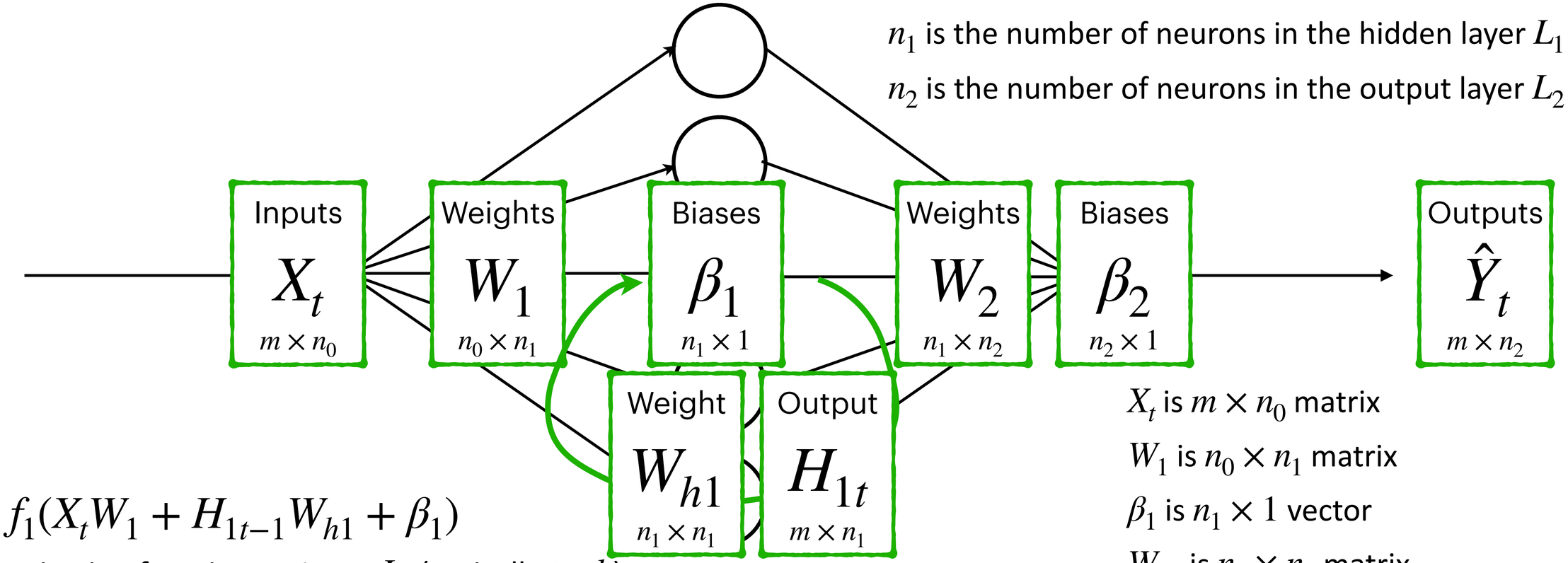
Recurrent Neural Networks

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2



$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

f_1 is an activation function on Layer L_1 (typically $tanh$)

$$\hat{Y}_t = f_2(H_t W_2 + \beta_2)$$

f_2 is an activation function on Layer L_2 (typically $softmax$ for classification, or $ReLU$ / $Identity$ for regression)

X_t is $m \times n_0$ matrix

W_1 is $n_0 \times n_1$ matrix

β_1 is $n_1 \times 1$ vector

W_{h1} is $n_1 \times n_1$ matrix

H_{1t} is $m \times n_1$ matrix

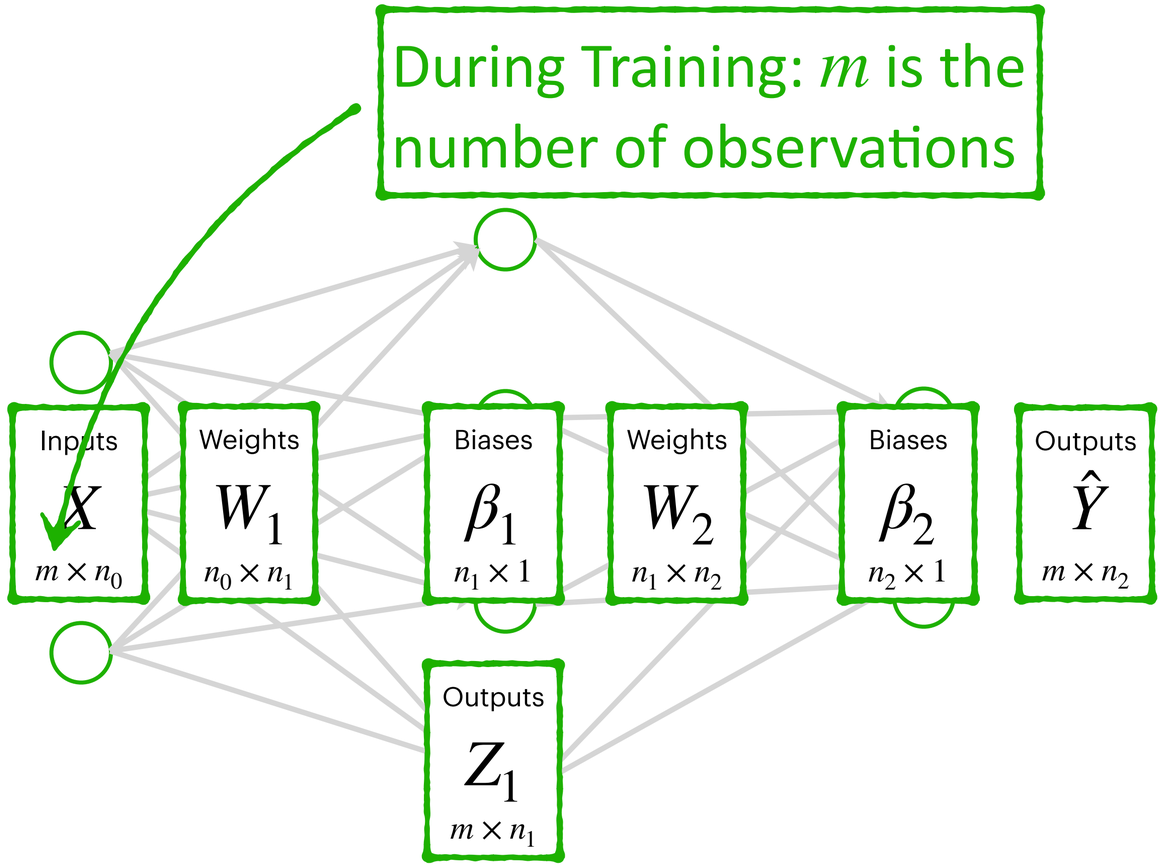
W_2 is $n_1 \times n_2$ matrix

β_2 is $n_2 \times 1$ vector

$\hat{Y}_t$ is $m \times n_2$ matrix

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

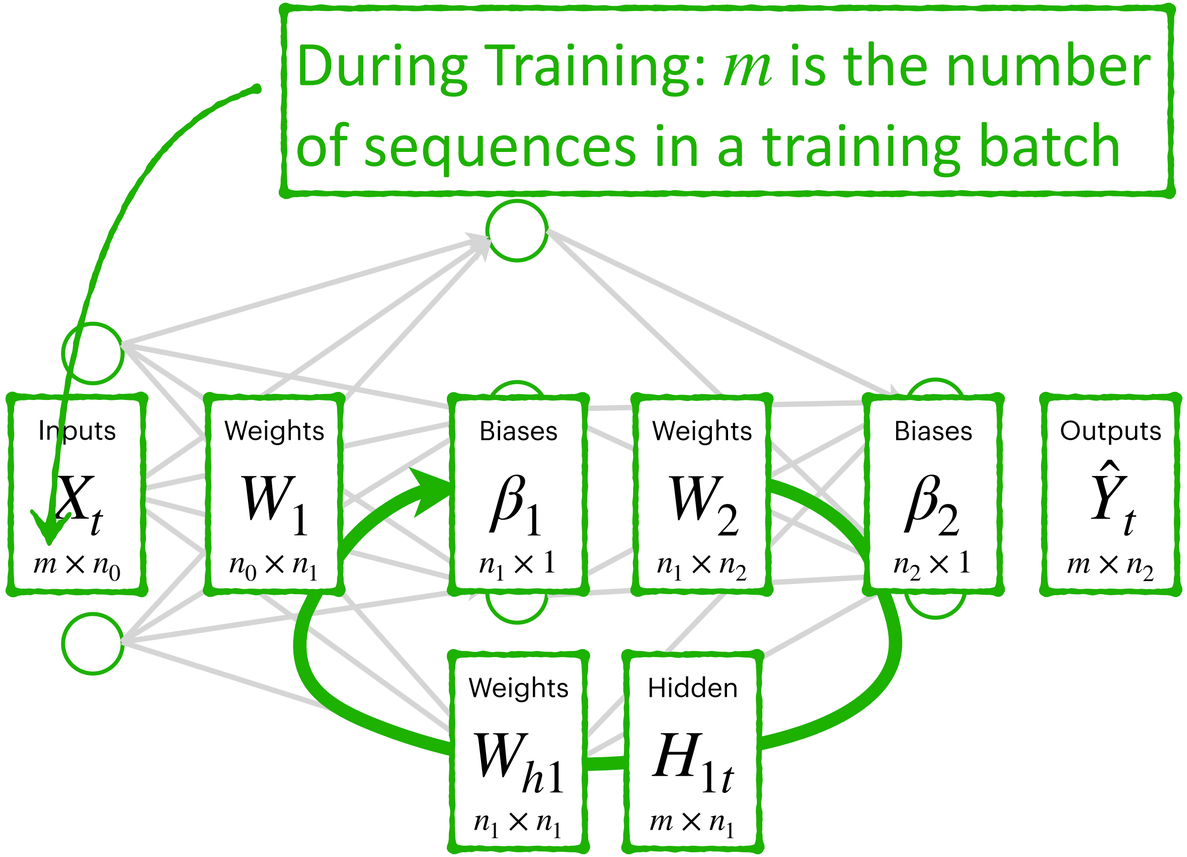


$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Feedforward Neural Network



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Training: m is the number of observations

Example: House Price Prediction

Inputs
X
$m \times n_0$

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

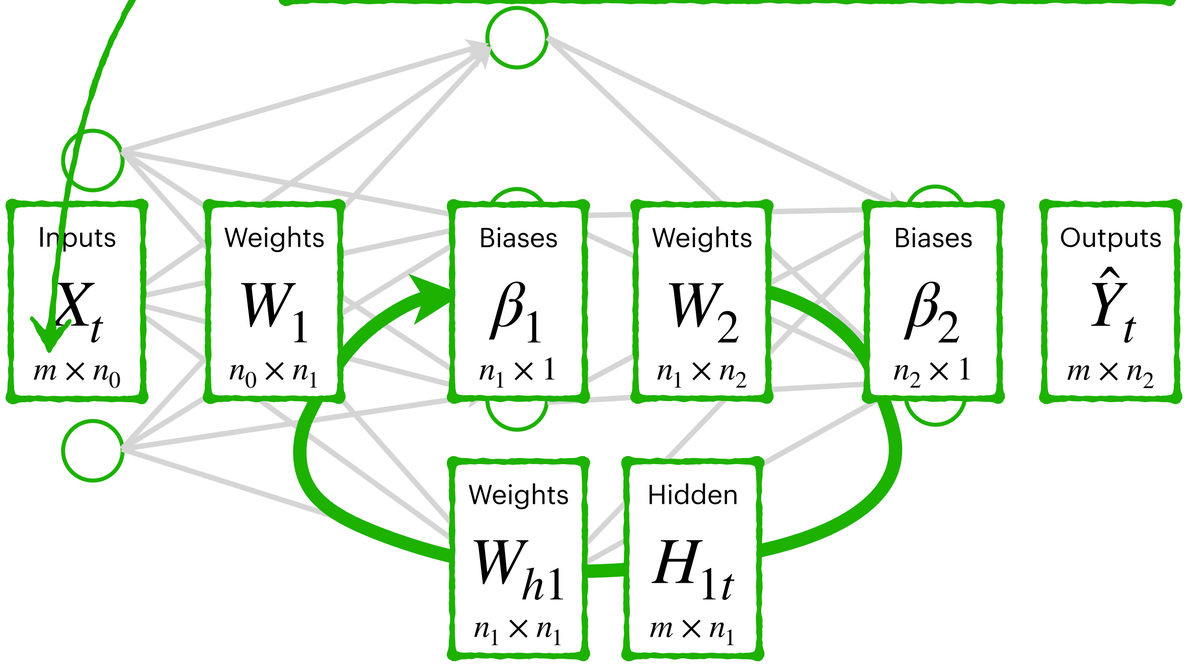
X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is training data for a single house for a total of m homes.

$n_0 = 3$
Input Layer

Feedforward Neural Network

During Training: m is the number of sequences in a training batch



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Training: m is the number of observations

Inputs

X

$m \times n_0$

$n_0 = 3$

Input Layer

Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is training data for a single house for a total of m homes.

Feedforward Neural Network

During Training: m is the number of sequences in a training batch

Inputs

X_t

$m \times n_0$

$n_0 =$

Input L

Example: Traffic Prediction

$$X_t = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X_t is a $m \times 3$ matrix of inputs (L_0)

$n_0 = 3$ features for each sequence: Traffic Volume, Latency & Error Rate

Each row in the matrix is one traffic sequence's data at time t with 3 features (e.g., volume, latency, error rate) for a total of m sequences in the training batch.

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Training: m is the number of observations

Inputs

X

$m \times n_0$

Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is training data for a single house for a total of m homes.

$n_0 = 3$

Input Layer

Feedforward Neural Network

The FNN is trained on data from m homes

During Training: m is the number of sequences in a training batch

Inputs

X_t

$m \times n_0$

Example: Traffic Prediction

$$X_t = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X_t is a $m \times 3$ matrix of inputs (L_0)

$n_0 = 3$ features for each sequence: Traffic Volume, Latency & Error Rate

Each row in the matrix is one traffic sequence's data at time t with 3 features (e.g., volume, latency, error rate) for a total of m sequences in the training batch.

$n_0 =$

Input L

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Training: m is the number of observations

Inputs

X

$m \times n_0$

Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is training data for a single house for a total of m homes.

$n_0 = 3$

Input Layer

Feedforward Neural Network

The FNN is trained on data from m homes

During Training: m is the number of sequences in a training batch

Inputs

X_t

$m \times n_0$

Example: Traffic Prediction

$$X_t = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X_t is a $m \times 3$ matrix of inputs (L_0)

$n_0 = 3$ features for each sequence: Traffic Volume, Latency & Error Rate

Each row in the matrix is one traffic sequence's data at time t with 3 features (e.g., volume, latency, error rate) for a total of m sequences in the training batch.

$n_0 =$

Input L

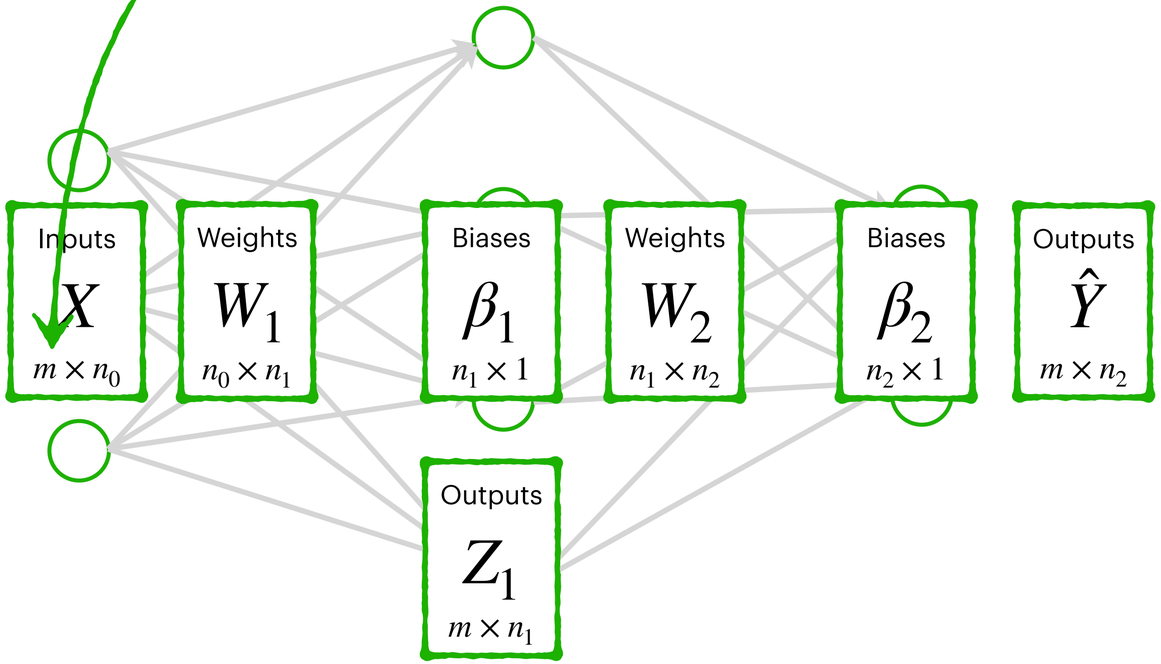
Recurrent Neural Network

The RNN is trained on traffic from m services in parallel, processing each sequence through T time steps

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Inference: m is the number of homes to predict the price for



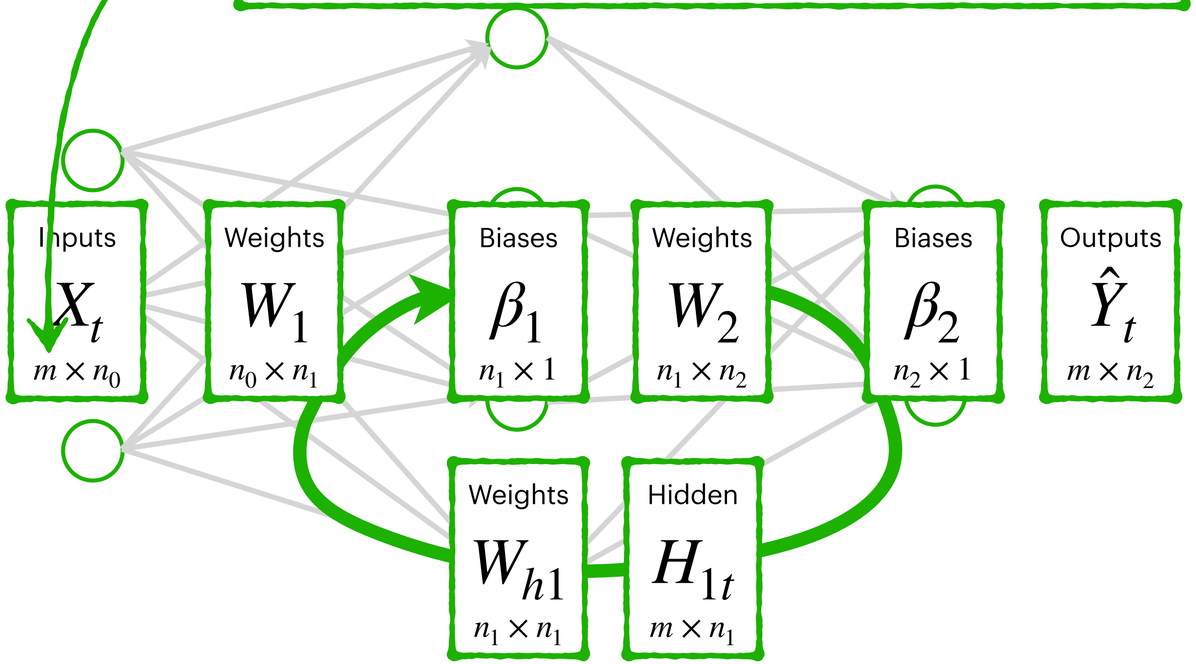
$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Feedforward Neural Network

During Inference: m is the number of sequences being processed in parallel, each predicting traffic at the next time step.



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

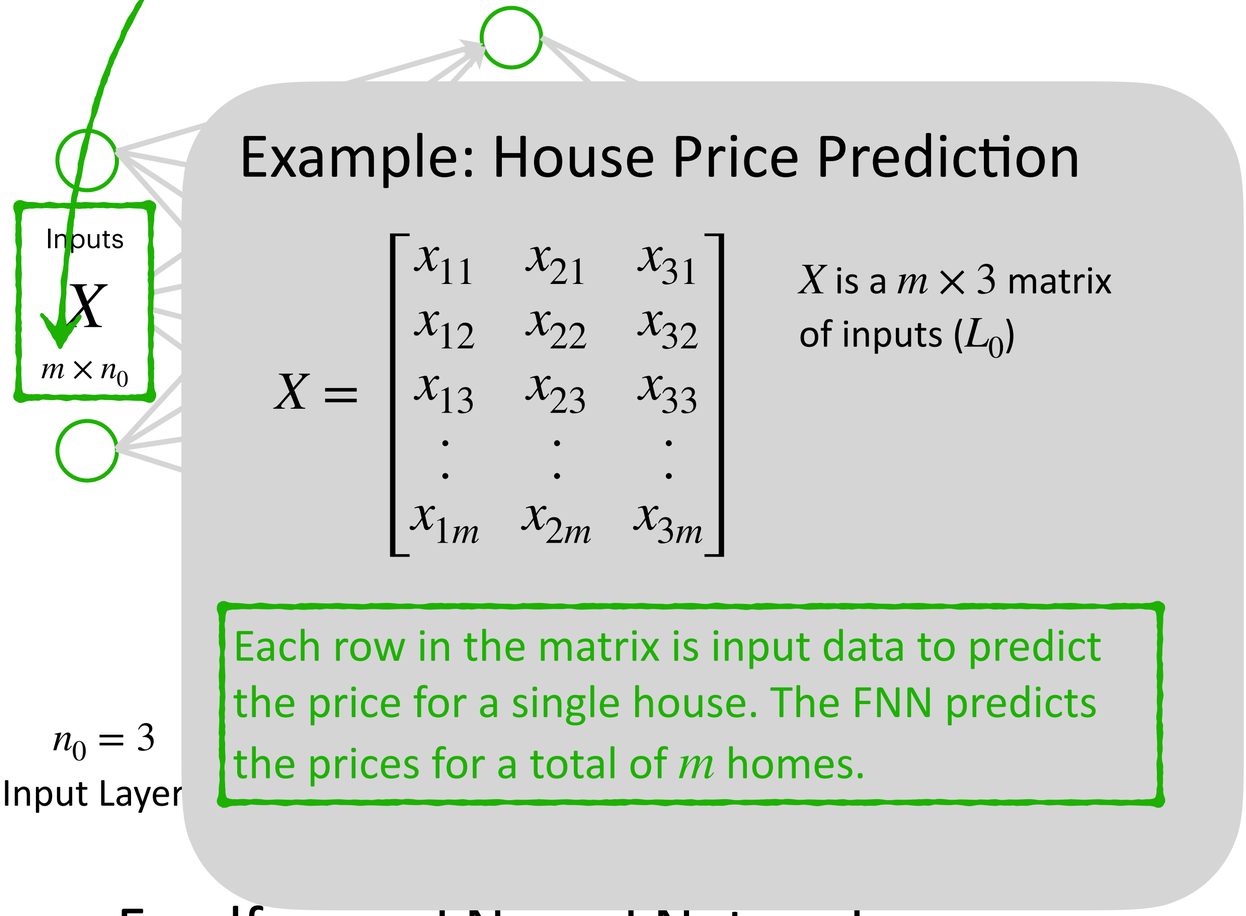
Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Inference: m is the number of homes to predict the price for

During Inference: m is the number of sequences being processed in parallel, each predicting traffic at the next time step.

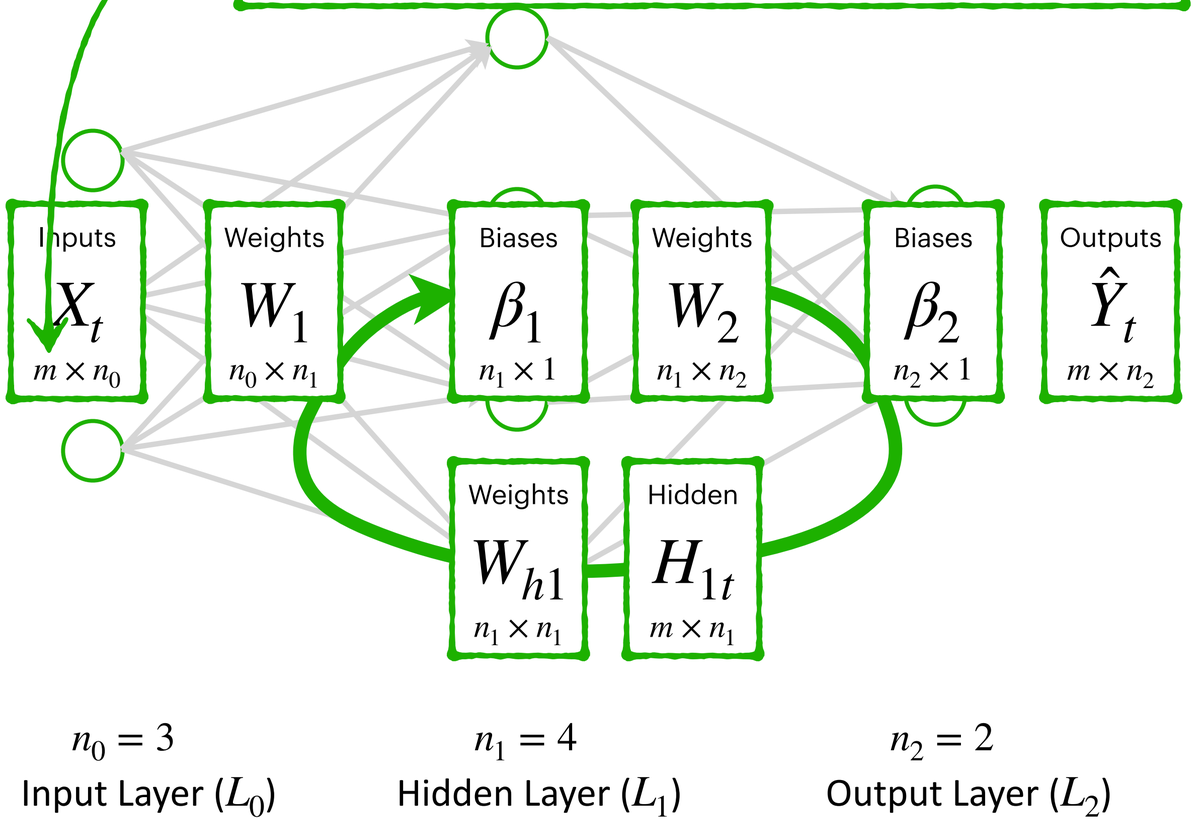


Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is input data to predict the price for a single house. The FNN predicts the prices for a total of m homes.



Feedforward Neural Network

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Inference: m is the number of homes to predict the price for

Inputs

X

$m \times n_0$

$n_0 = 3$

Input Layer

Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is input data to predict the price for a single house. The FNN predicts the prices for a total of m homes.

Feedforward Neural Network

During Inference: m is the number of sequences being processed in parallel, each predicting traffic at the next time step.

Inputs

X_t

$m \times n_0$

$n_0 =$

Input L

Example: Traffic Prediction

$$X_t = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X_t is a $m \times 3$ matrix of inputs (L_0)

$n_0 = 3$ features for each sequence: Traffic Volume, Latency & Error Rate

Each row in the matrix is input data (3 features) to predict the traffic for one service for the next time step. The RNN predicts the traffic for m services in parallel for the next time step.

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Inference: m is the number of homes to predict the price for

Inputs

X

$m \times n_0$

$n_0 = 3$

Input Layer

Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is input data to predict the price for a single house. The FNN predicts the prices for a total of m homes.

Feedforward Neural Network

The FNN predicts the price for m homes

During Inference: m is the number of sequences being processed in parallel, each predicting traffic at the next time step.

Inputs

X_t

$m \times n_0$

$n_0 =$

Input L

Example: Traffic Prediction

$$X_t = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X_t is a $m \times 3$ matrix of inputs (L_0)

$n_0 = 3$ features for each sequence: Traffic Volume, Latency & Error Rate

Each row in the matrix is input data (3 features) to predict the traffic for one service for the next time step. The RNN predicts the traffic for m services in parallel for the next time step.

Recurrent Neural Network

Notation: Feedforward Network vs Recurrent Neural Network

Recurrent Neural Networks

During Inference: m is the number of homes to predict the price for

Inputs

X

$m \times n_0$

Example: House Price Prediction

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X is a $m \times 3$ matrix of inputs (L_0)

Each row in the matrix is input data to predict the price for a single house. The FNN predicts the prices for a total of m homes.

$n_0 = 3$

Input Layer

Feedforward Neural Network

The FNN predicts the price for m homes

During Inference: m is the number of sequences being processed in parallel, each predicting traffic at the next time step.

Inputs

X_t

$m \times n_0$

Example: Traffic Prediction

$$X_t = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ \vdots & \vdots & \vdots \\ x_{1m} & x_{2m} & x_{3m} \end{bmatrix}$$

X_t is a $m \times 3$ matrix of inputs (L_0)

$n_0 = 3$ features for each sequence: Traffic Volume, Latency & Error Rate

Each row in the matrix is input data (3 features) to predict the traffic for one service for the next time step. The RNN predicts the traffic for m services in parallel for the next time step.

$n_0 =$

Input L

Recurrent Neural Network

The RNN predicts the traffic for m services in parallel for the next time step.

Recurrent Neural Networks

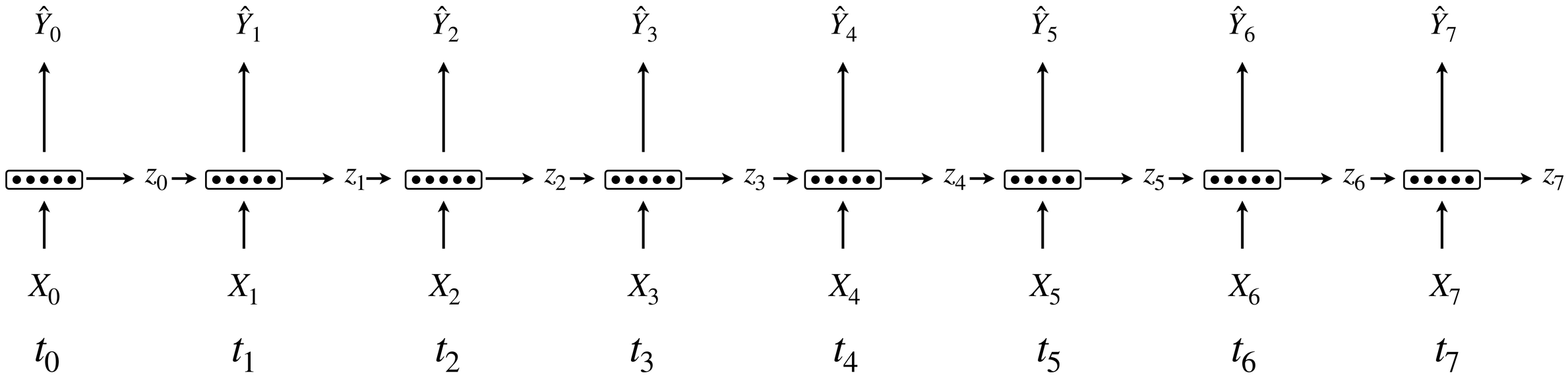
There are several different types of RNNs depending on the application…

Recurrent Neural Networks

Sequence to Sequence: For each input at a given time step (t) the RNN produces an output for that time step.

Typical Application: Stock Price Prediction, Traffic Prediction or Energy Demand Forecasting

Also known as many-to-many. The input sequence produces an output sequence



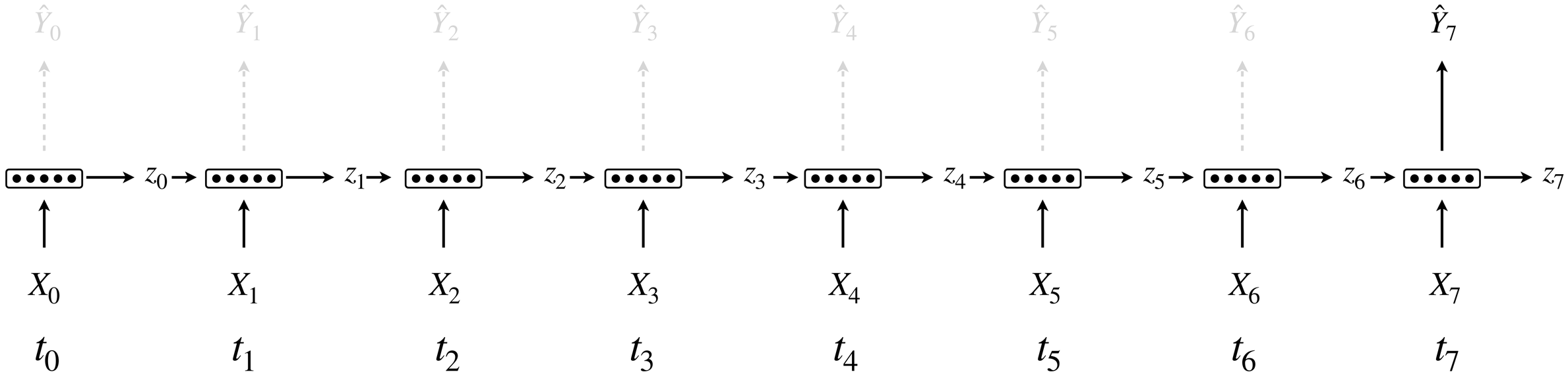
Recurrent Neural Networks

Sequence to Vector: The RNN processes the entire input sequence and produces a single output at the final time step.

Typical Application: Sentiment Analysis, Fraud Detection, Spam Filtering or Document Classification

Also known as many-to-one. The input sequence produces a single output at the last time step

Intermediate outputs are ignored



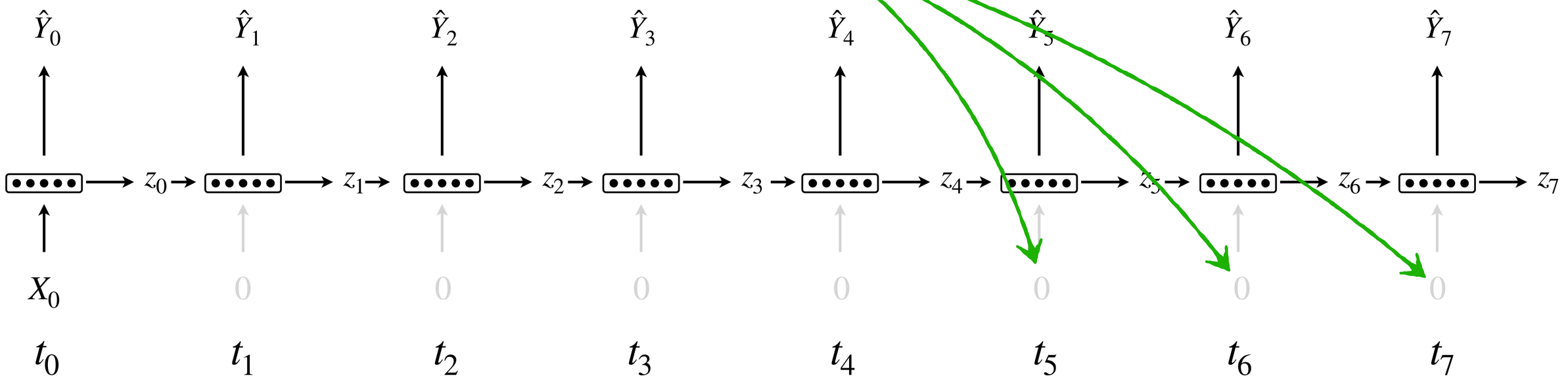
Recurrent Neural Networks

Vector to Sequence: The RNN processes a single initial input and produces a sequence of outputs at subsequent time steps.

Typical Application: Image Captioning, Music Generation, Video Description

Also known as one-to-many. The input at time t_0 produces a sequence

Approach 1: Inputs at the subsequent time steps are all 0



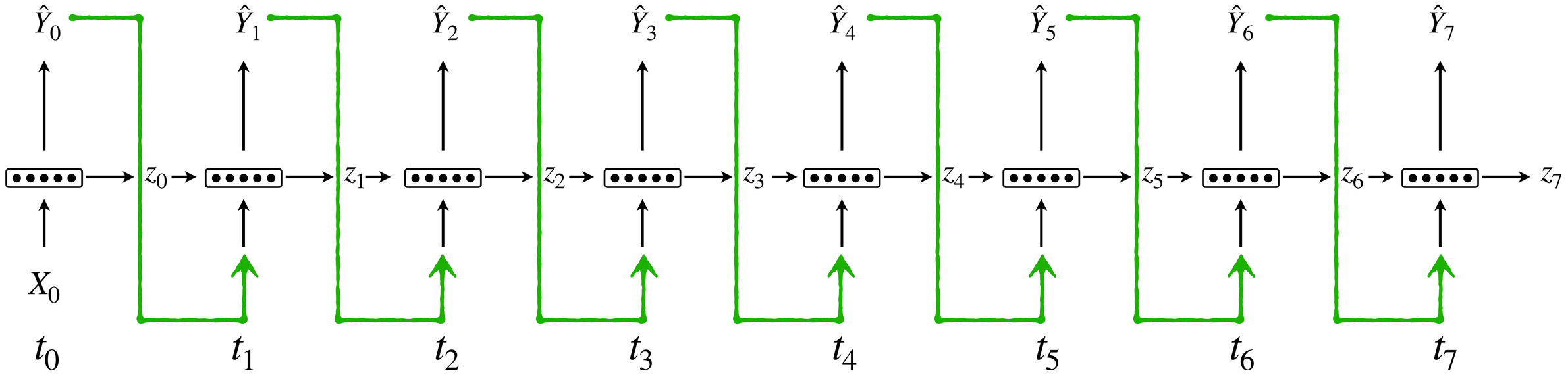
Recurrent Neural Networks

Vector to Sequence: The RNN processes a single initial input and produces a sequence of outputs at subsequent time steps.

Typical Application: Image Captioning, Music Generation, Video Description

Also known as one-to-many. The input at time t_0 produces a sequence

Approach 2: Output from a time step t is fed back as input at the subsequent time steps

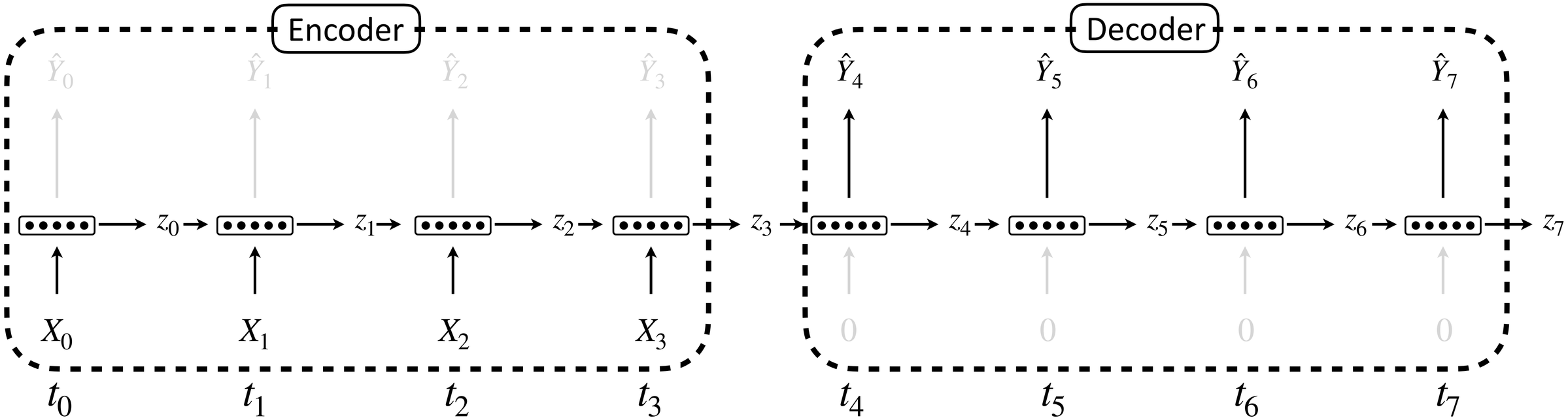


Recurrent Neural Networks

Encoder / Decoder: The RNN processes inputs over several time steps and then starts producing outputs over subsequent time steps.

Typical Application: Machine Translation, Text Summarization and Question to Answer

Also known as many-to-many. The encoder reads the input sequence, then the decoder generates the output sequence.
(input & output sequences can be of different lengths).



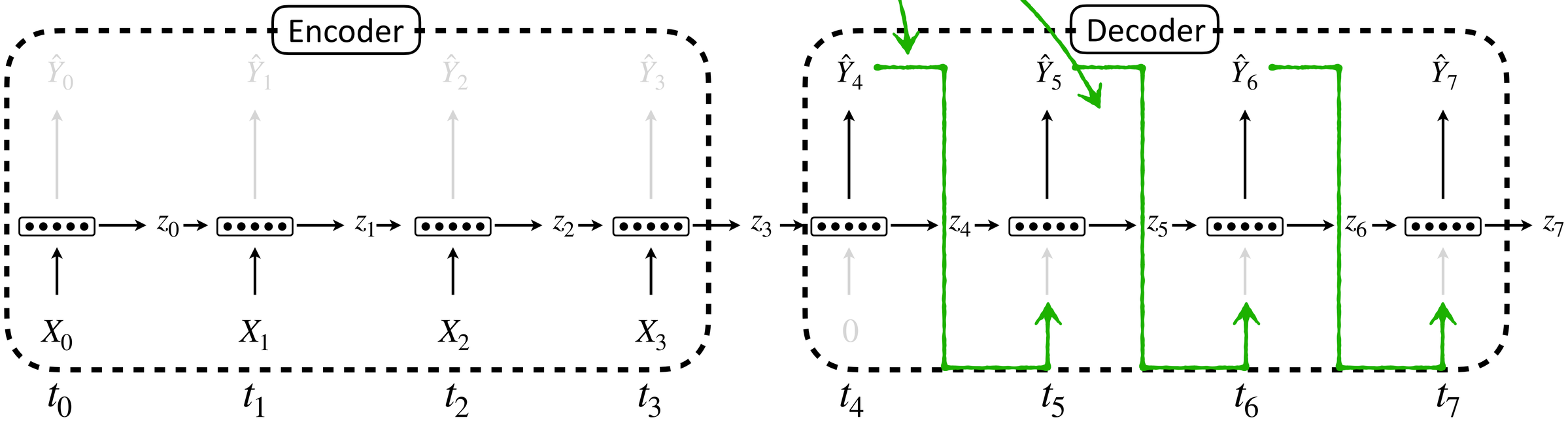
Recurrent Neural Networks

Encoder / Decoder: The RNN processes inputs over several time steps and then starts producing outputs over subsequent time steps.

Typical App
Summarizat

The output of each time step in the decoder can also be fed back as input to the next time step.

Also known as many-to-many. The encoder reads the input sequence, then the decoder generates the output sequence.
(input & output sequences can be of different lengths).



Recurrent Neural Networks

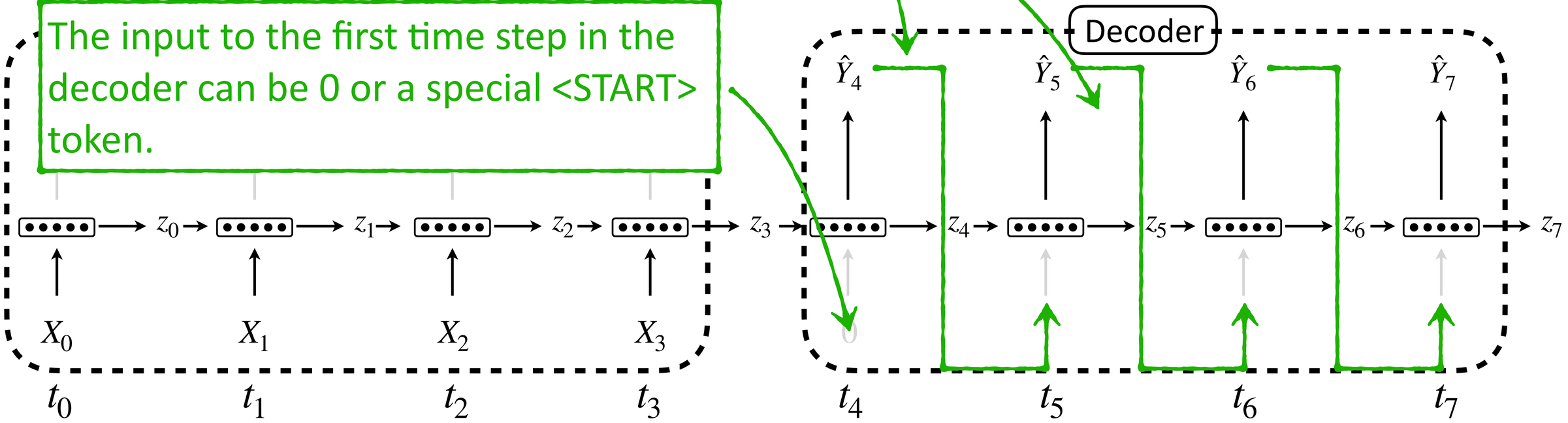
Encoder / Decoder: The RNN processes inputs over several time steps and then starts producing outputs over subsequent time steps.

Typical App
Summarizat

The output of each time step in the decoder can also be fed back as input to the next time step.

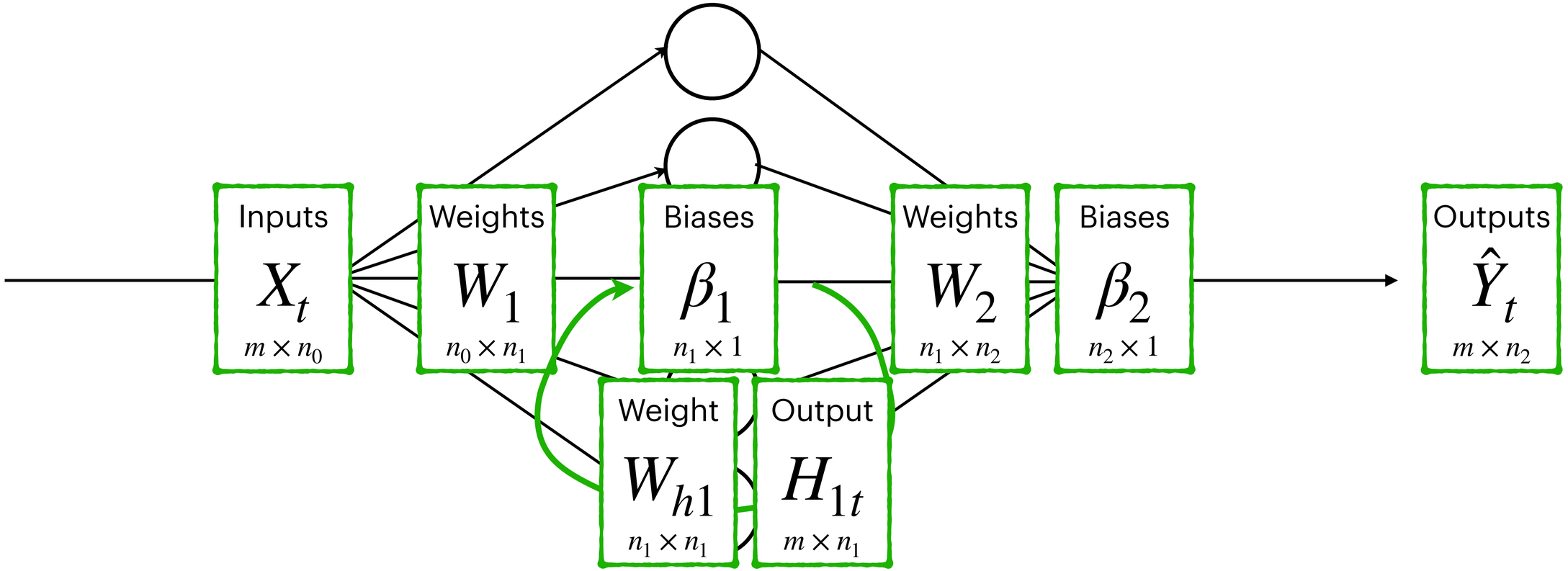
Also known as many-to-many. The encoder reads the input sequence, then the decoder generates the output sequence.
(input & output sequences can be of different lengths).

The input to the first time step in the decoder can be 0 or a special <START> token.



Recurrent Neural Networks

Training a Recurrent Neural Network



Training a Recurrent Neural Network

Problem Statement: Optimize the weights (W_1, W_2, W_{h1}) and Biases (β_1, β_2) to minimize the error between the predictions $(\hat{Y})$ and the observations (Y)

Question: How do we train a Recurrent Neural Network

(We'll use Gradient Descent with a technique known as Backpropagation Through Time (BPTT) to compute gradients across time steps)

Related Tutorials & Textbooks

Neural Networks

An introduction to Neural Networks starting from a foundation of linear regression, logistic classification and multi class classification models along with the matrix representation of a neural network generalized to *l* layers with *n* neurons

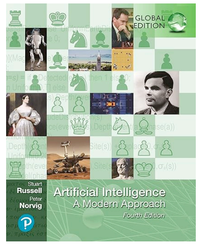
Forward and Back Propagation in Neural Networks

A deep dive into how Neural Networks are trained using Gradient Descent. Output predictions, are compared to observations to calculate loss and Backward propagation then computes gradients by working backward through the network

Gradient Descent for Multiple Regression

Gradient Descent algorithm for multiple regression and how it can be used to optimize k + 1 parameters for a Linear model in multiple dimensions.

Recommended Textbooks



Artificial Intelligence: A Modern Approach
by Peter Norvig, Stuart Russell

For a complete list of tutorials see:
https://arrsingh.com/ai-tutorials