Recurrent Neural Networks

Training & Back Propagation Through Time

Rahul Singh
rsingh@arrsingh.com

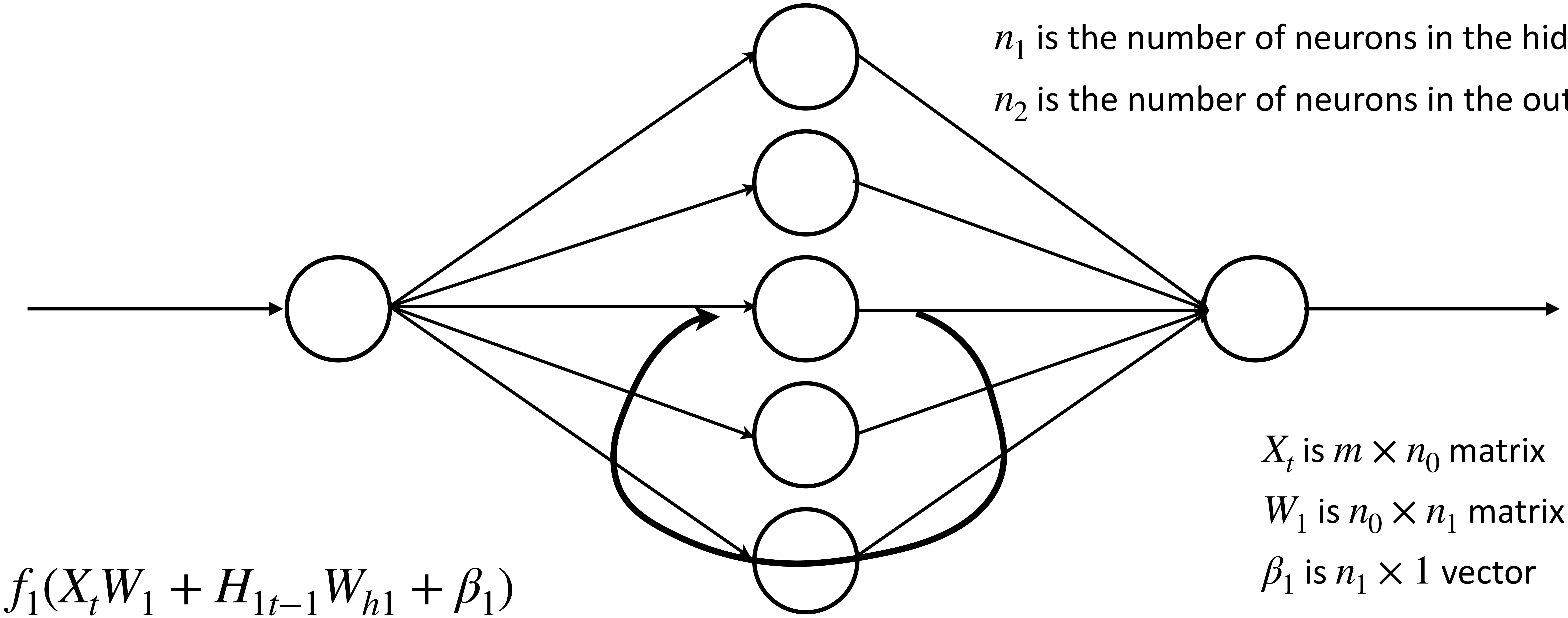
How do we represent RNNs mathematically?

Recurrent Neural Networks

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 $tanh$)

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

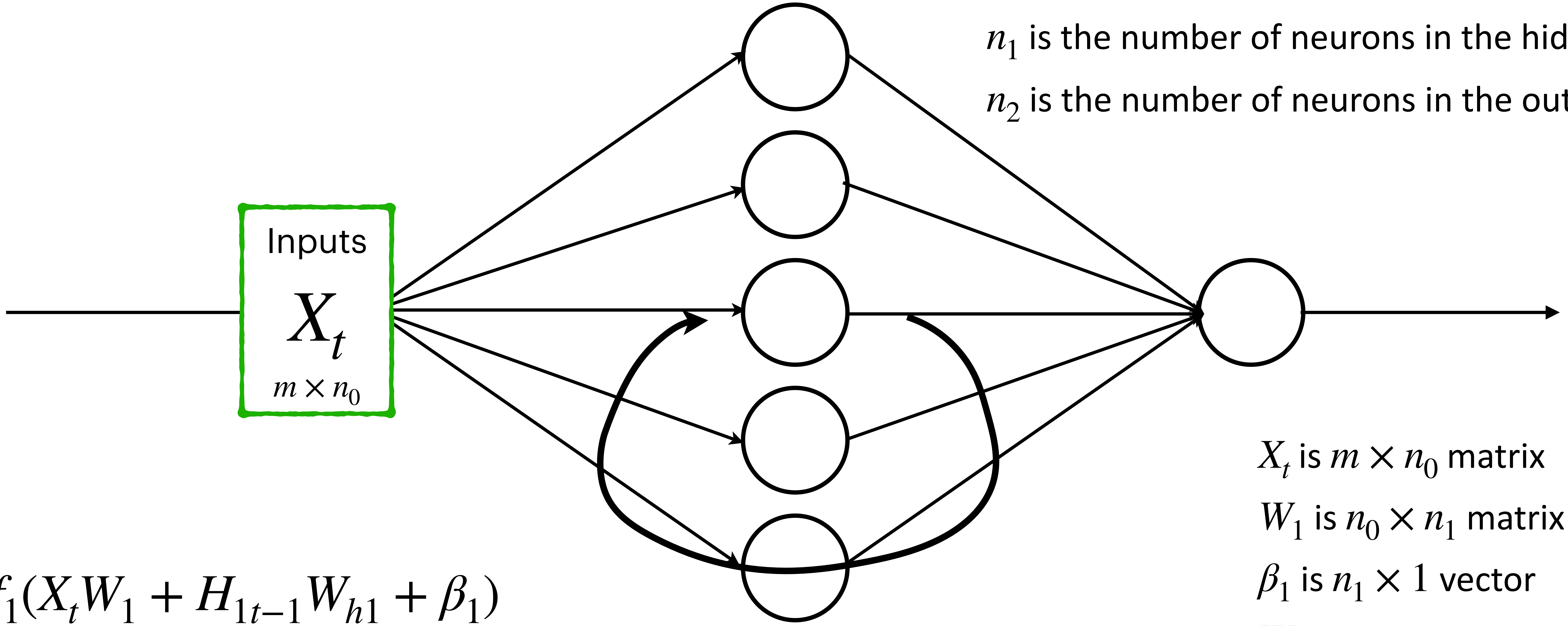
How do we represent RNNs mathematically?

Recurrent Neural Networks

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



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

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

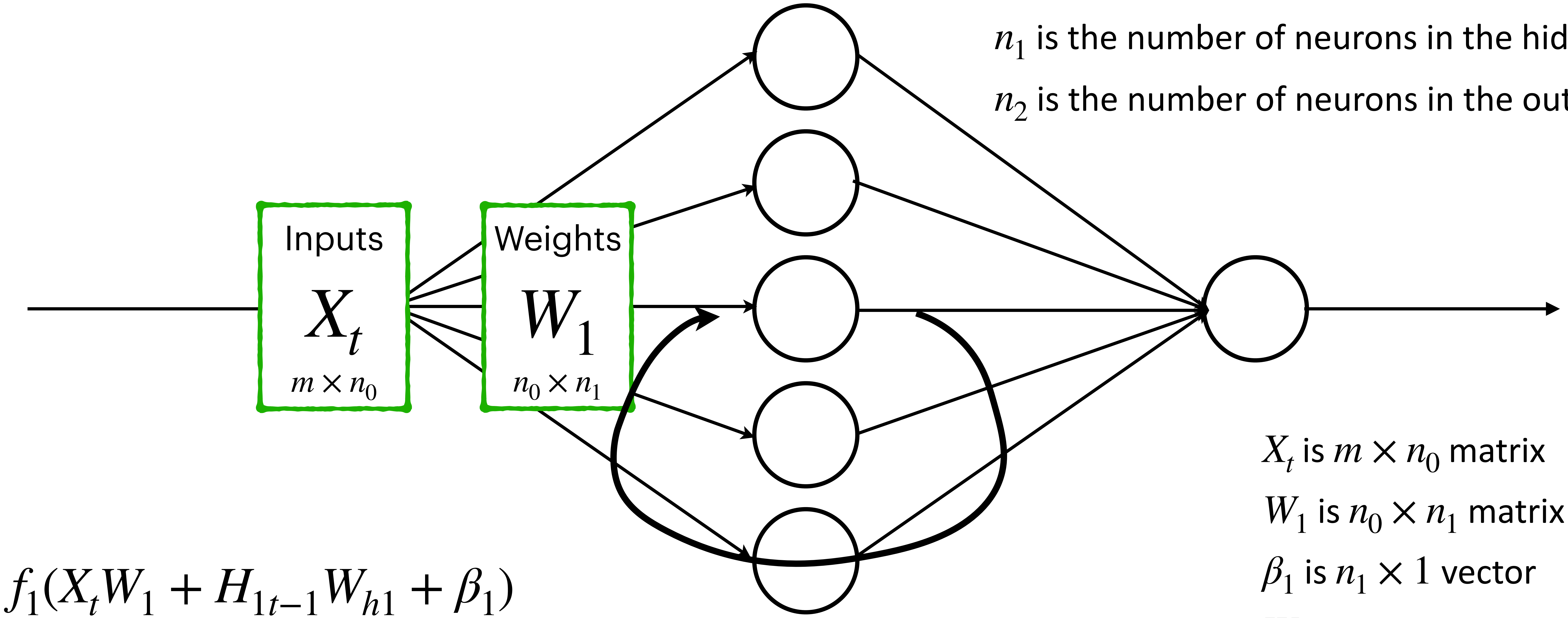
How do we represent RNNs mathematically?

Recurrent Neural Networks

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



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

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

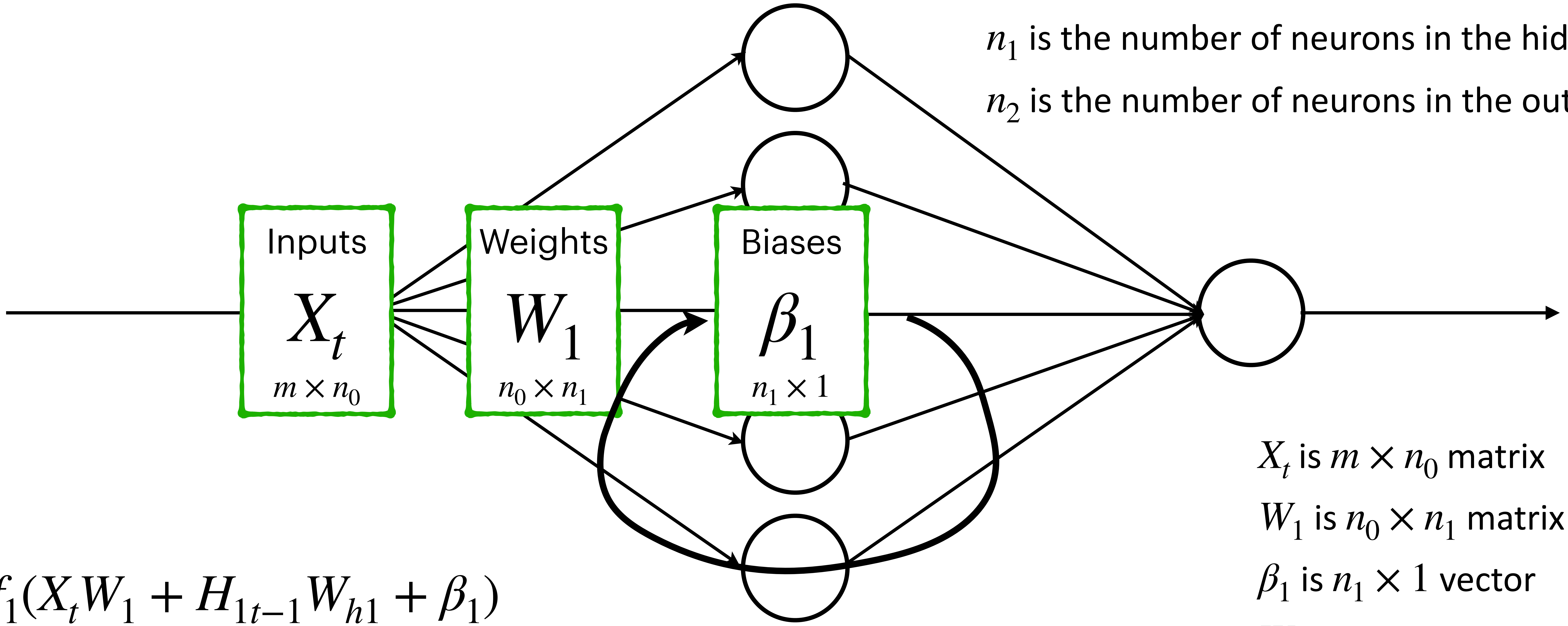
How do we represent RNNs mathematically?

Recurrent Neural Networks

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 $tanh$)

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

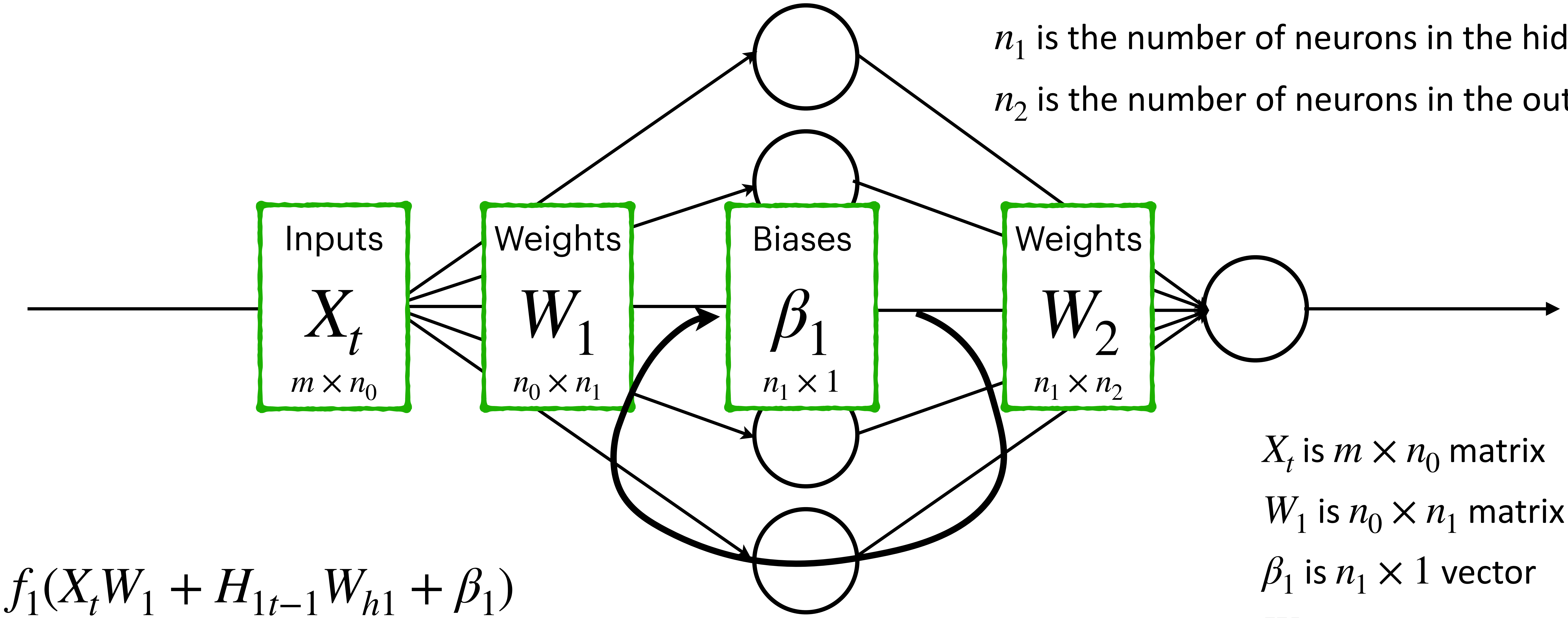
How do we represent RNNs mathematically?

Recurrent Neural Networks

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 $tanh$)

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

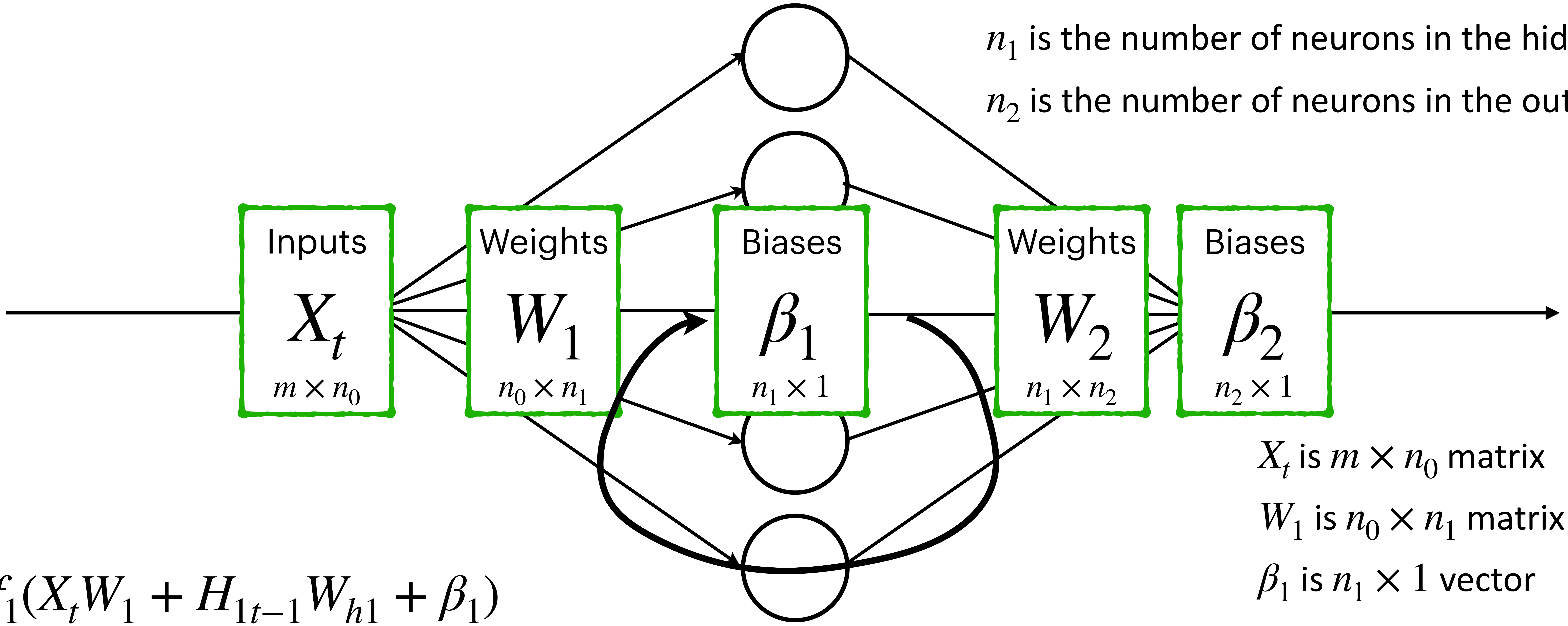
How do we represent RNNs mathematically?

Recurrent Neural Networks

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 $tanh$)

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

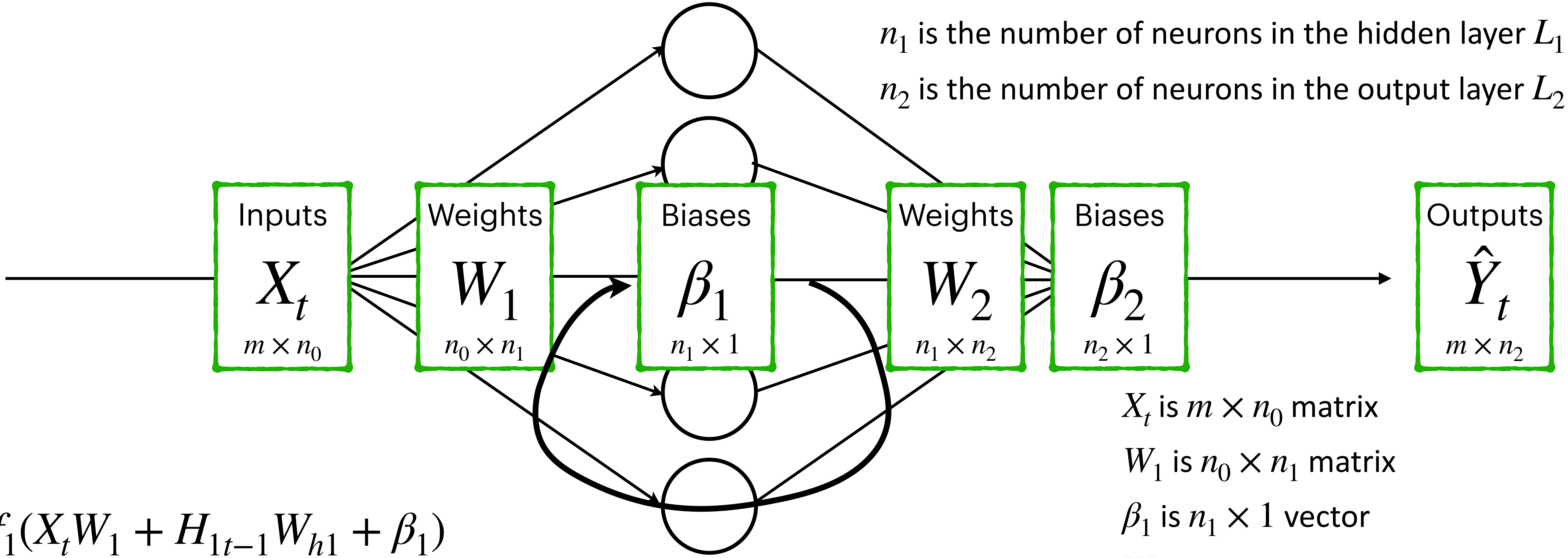
How do we represent RNNs mathematically?

Recurrent Neural Networks

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



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

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

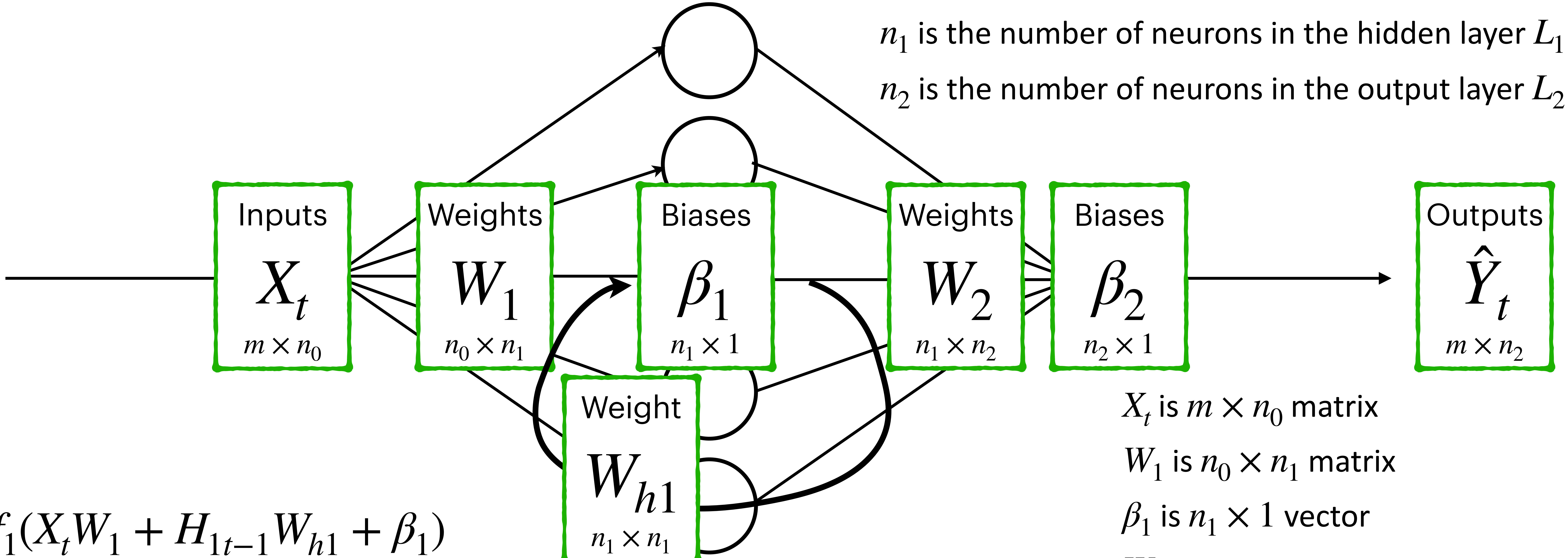
How do we represent RNNs mathematically?

Recurrent Neural Networks

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 $tanh$)

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

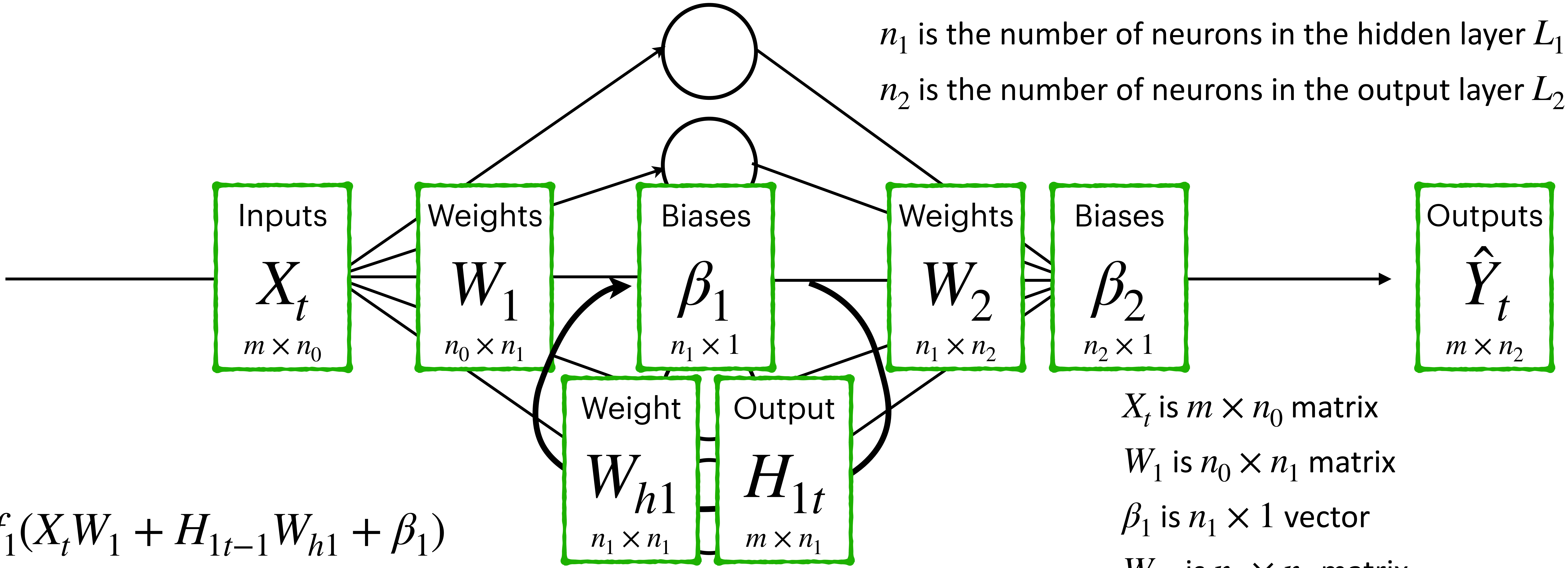
How do we represent RNNs mathematically?

Recurrent Neural Networks

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



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

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

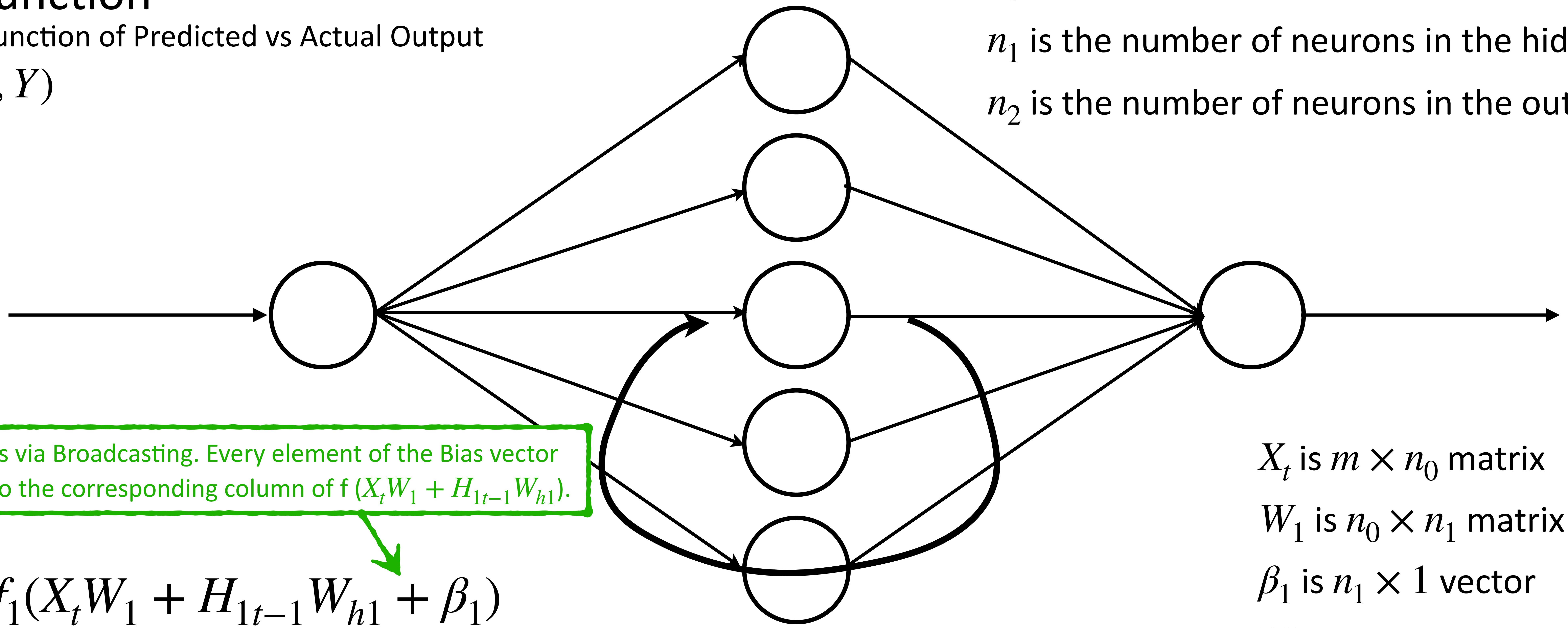
Loss is a function of Predicted vs Actual Output

$$L = f(\hat{Y}, Y)$$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f $(X_t W_1 + H_{1t-1} W_{h1})$.

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of $(H_{1t} W_2)$.

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

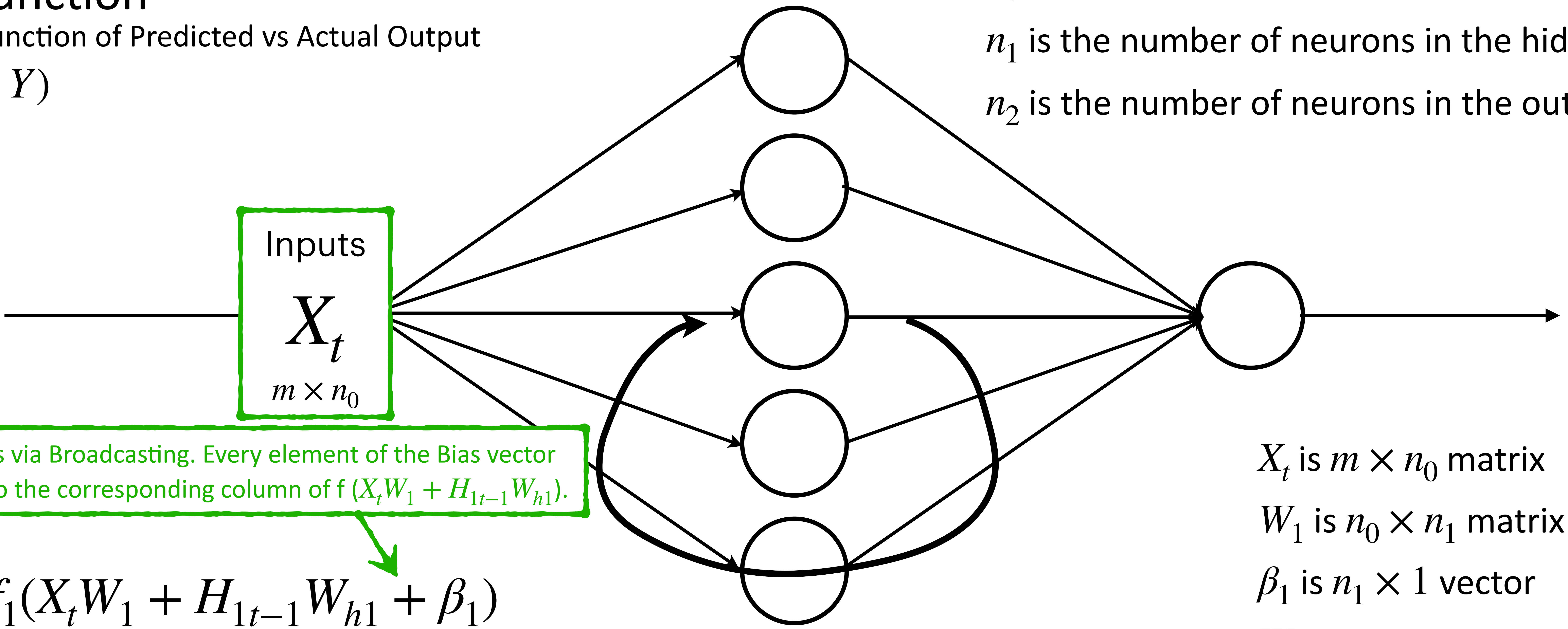
Loss is a function of Predicted vs Actual Output

$L = f(\hat{Y}, Y)$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f ($X_t W_1 + H_{1t-1} W_{h1}$).

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of ($H_{1t} W_2$).

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

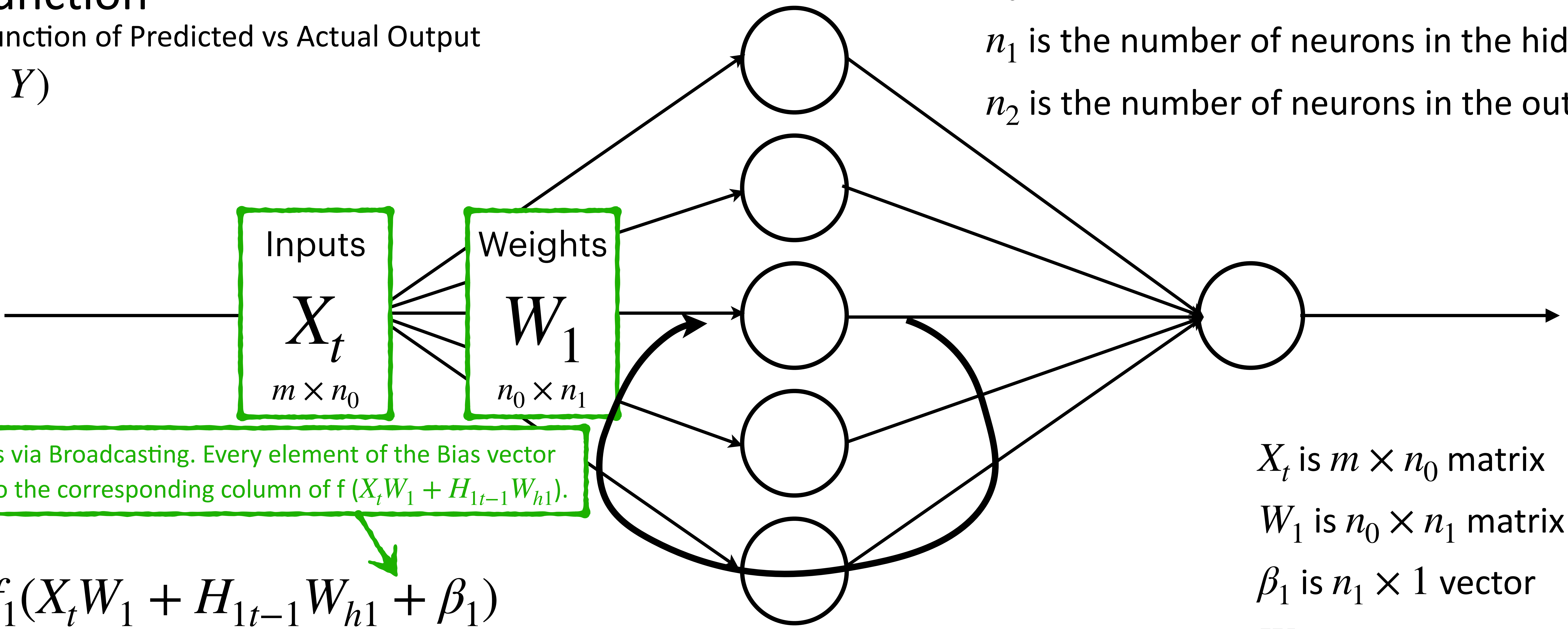
Loss is a function of Predicted vs Actual Output

$L = f(\hat{Y}, Y)$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f $(X_t W_1 + H_{1t-1} W_{h1})$.

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of $(H_{1t} W_2)$.

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

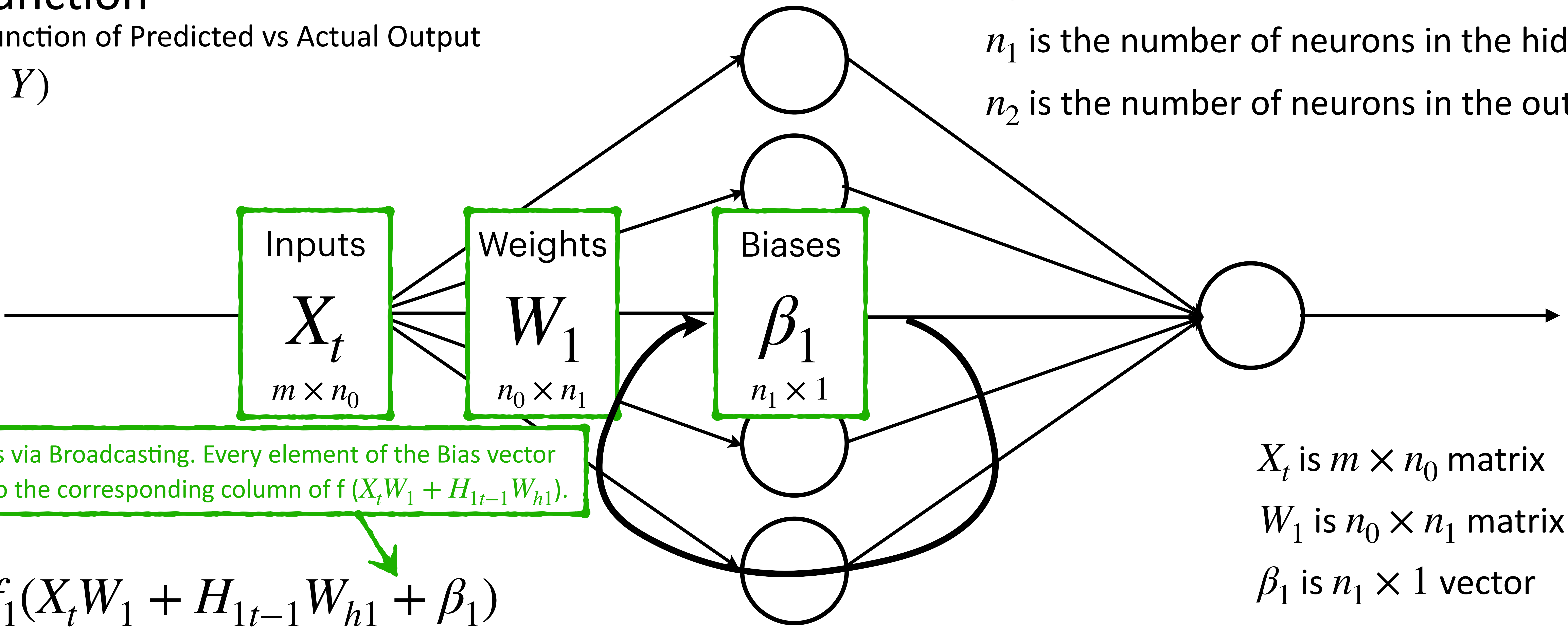
Loss is a function of Predicted vs Actual Output

$L = f(\hat{Y}, Y)$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f $(X_t W_1 + H_{1t-1} W_{h1})$.

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of $(H_{1t} W_2)$.

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

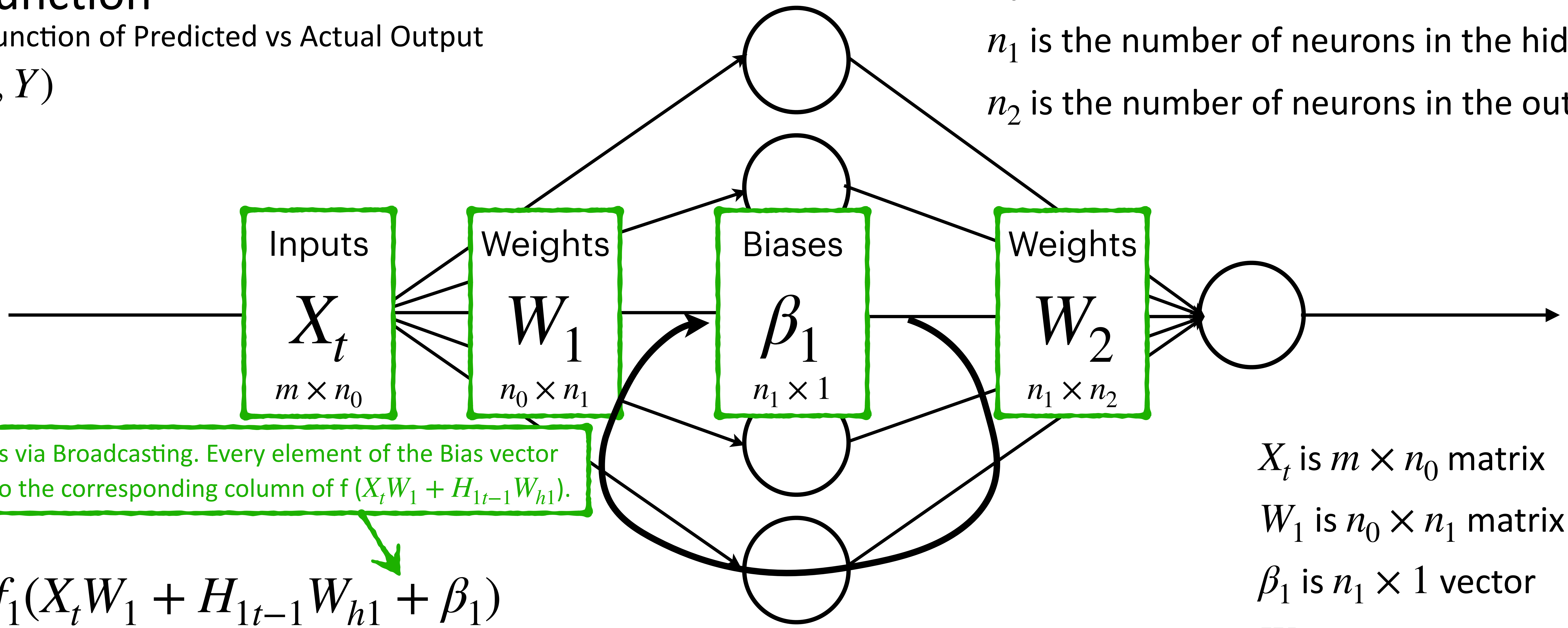
Loss is a function of Predicted vs Actual Output

$L = f(\hat{Y}, Y)$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f ($X_t W_1 + H_{1t-1} W_{h1}$).

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of ($H_{1t} W_2$).

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?

Loss Function

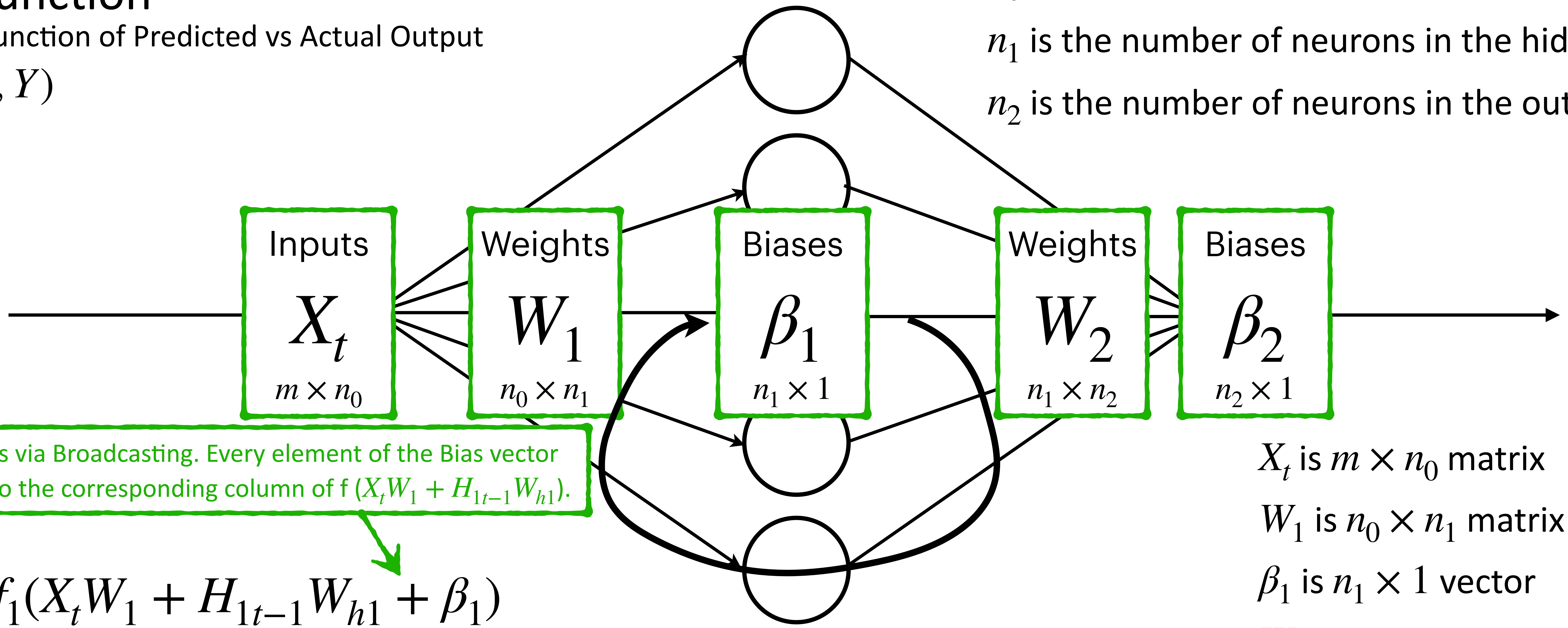
Loss is a function of Predicted vs Actual Output

$L = f(\hat{Y}, Y)$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f ($X_t W_1 + H_{1_{t-1}} W_{h1}$).

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of ($H_{1t} W_2$).

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

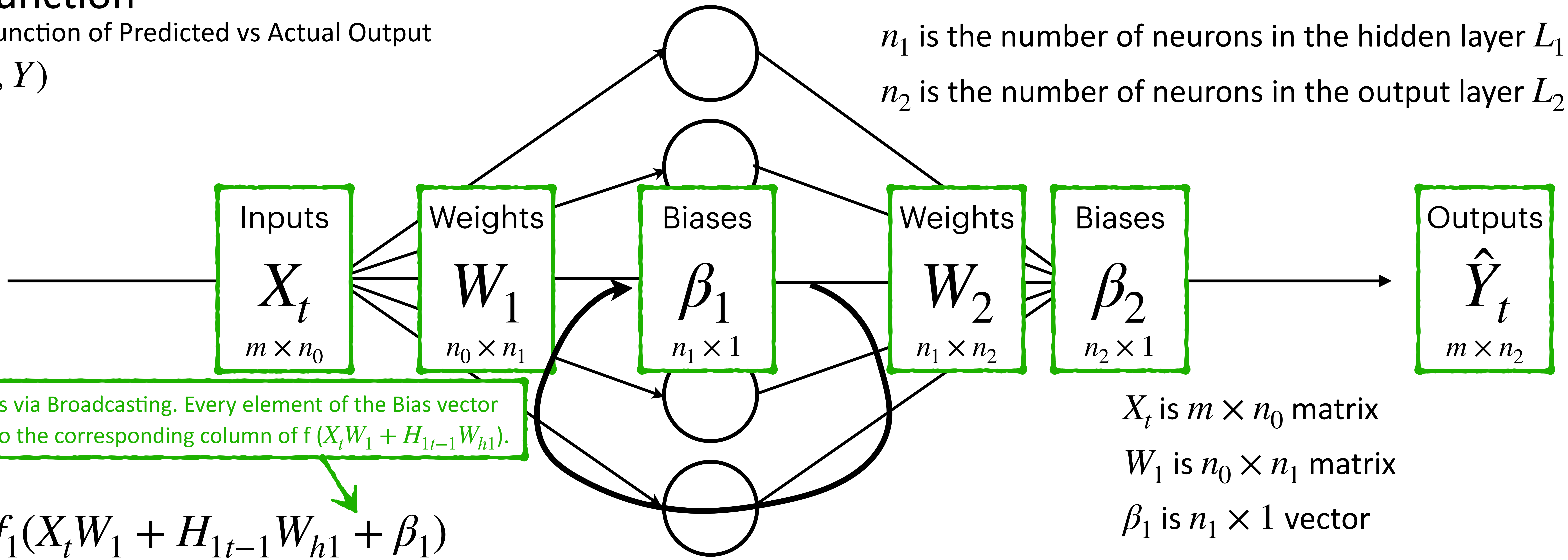
Loss is a function of Predicted vs Actual Output

$$L = f(\hat{Y}, Y)$$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f ($X_t W_1 + H_{1t-1} W_{h1}$).

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of ($H_{1t} W_2$).

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

How do we represent RNNs mathematically?

n_0 is the number of features in the input data

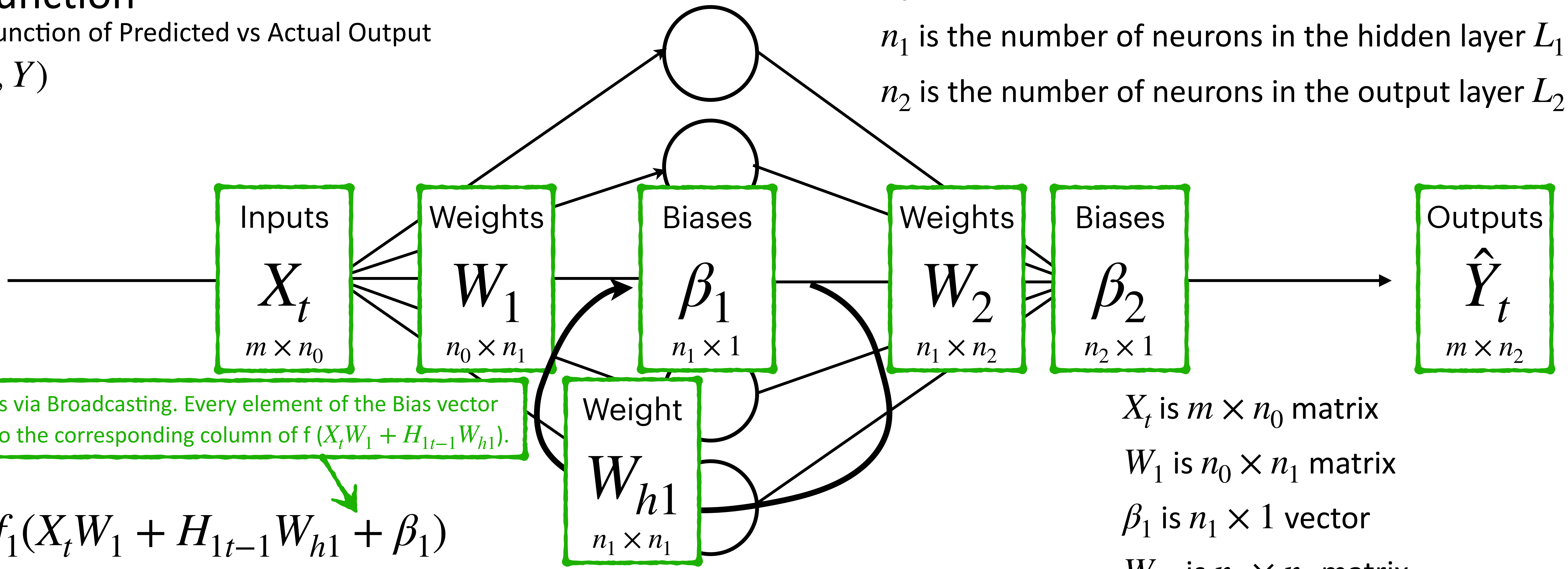
n_1 is the number of neurons in the hidden layer L_1

n_2 is the number of neurons in the output layer L_2

Loss Function

Loss is a function of Predicted vs Actual Output

$$L = f(\hat{Y}, Y)$$



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f ($X_t W_1 + H_{1t-1} W_{h1}$).

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of ($H_{1t} W_2$).

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

How do we represent RNNs mathematically?

Recurrent Neural Networks

Loss Function

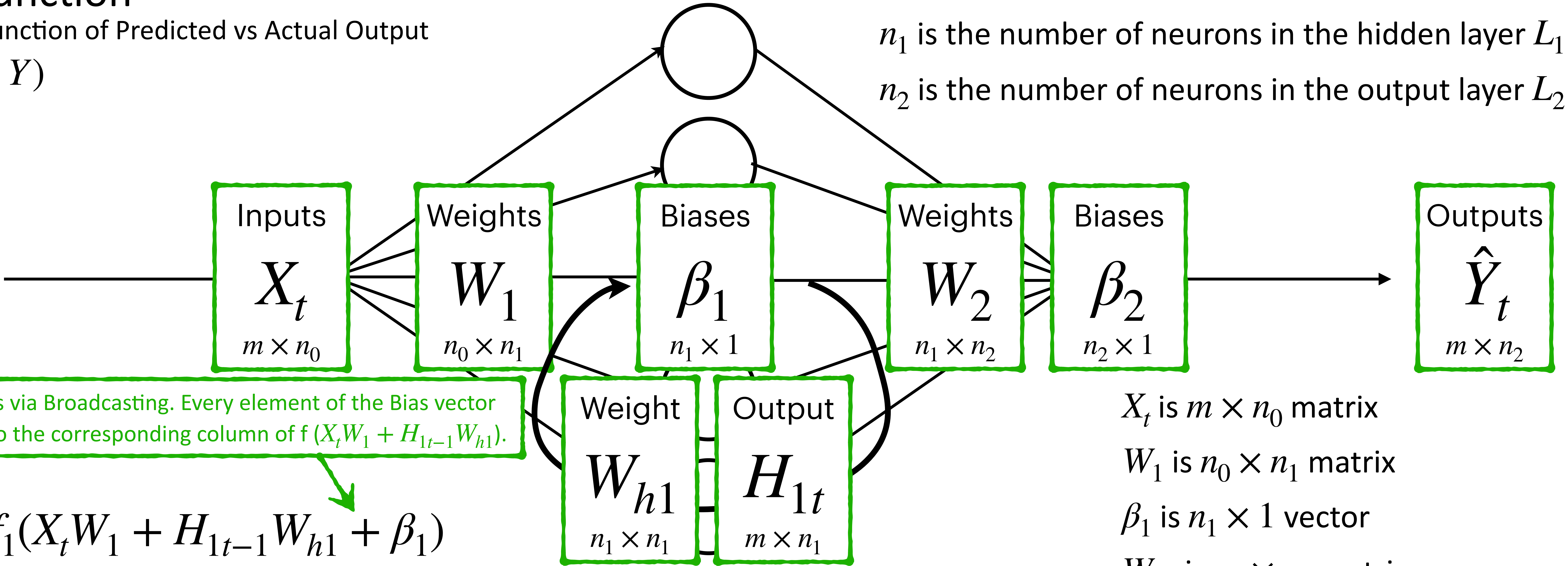
Loss is a function of Predicted vs Actual Output

$L = f(\hat{Y}, Y)$

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



Bias addition is via Broadcasting. Every element of the Bias vector (β_1) is added to the corresponding column of f ($X_t W_1 + H_{1t-1} W_{h1}$).

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

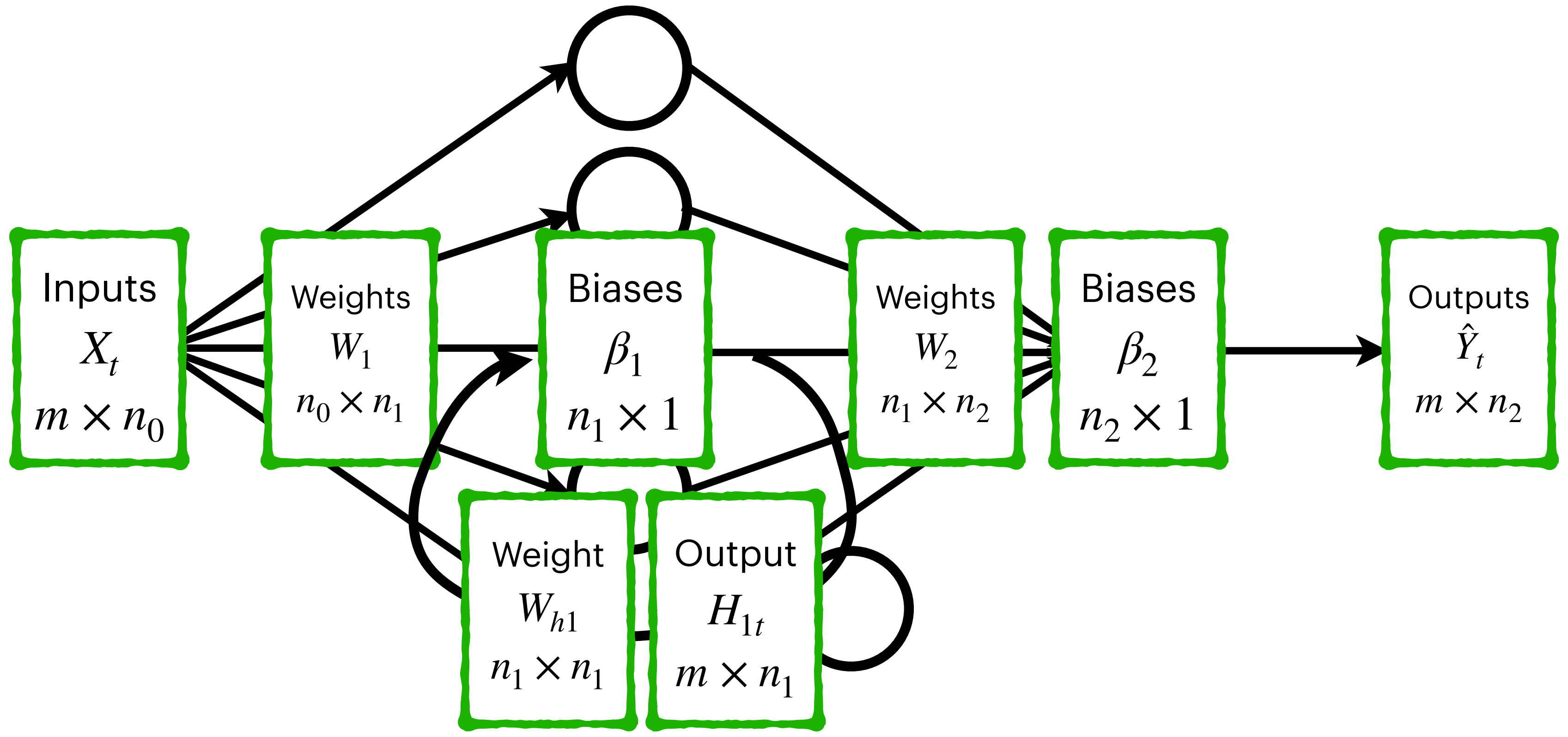
Bias addition is via Broadcasting. Every element of the Bias Vector (β_2) is added to the corresponding column of ($H_{1t} W_2$).

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

How do we represent RNNs mathematically?

Recurrent Neural Networks



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

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

In General the Total Loss is the sum of Losses over all time steps:

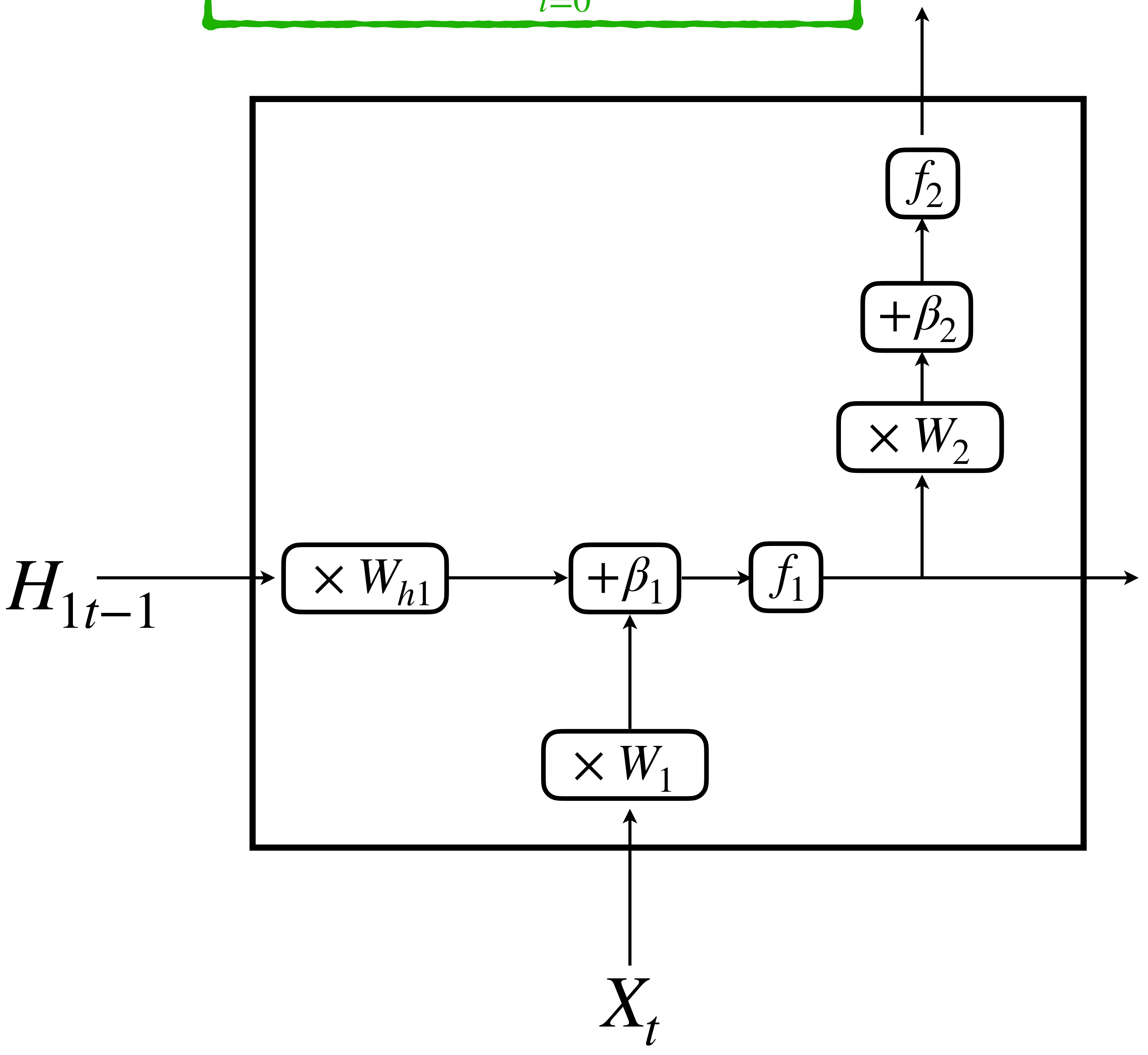
$$L = \sum_{t=0}^{T} L_t$$

Loss Function

Loss is a function of Predicted vs Actual Output

$$L_t = f(\hat{Y}_t, Y_t)$$

$$\hat{Y}_t = f_2(H_{1t}W_2 + \beta_2)$$



$$H_{1t} = f_1(X_tW_1 + H_{1t-1}W_{h1} + \beta_1)$$

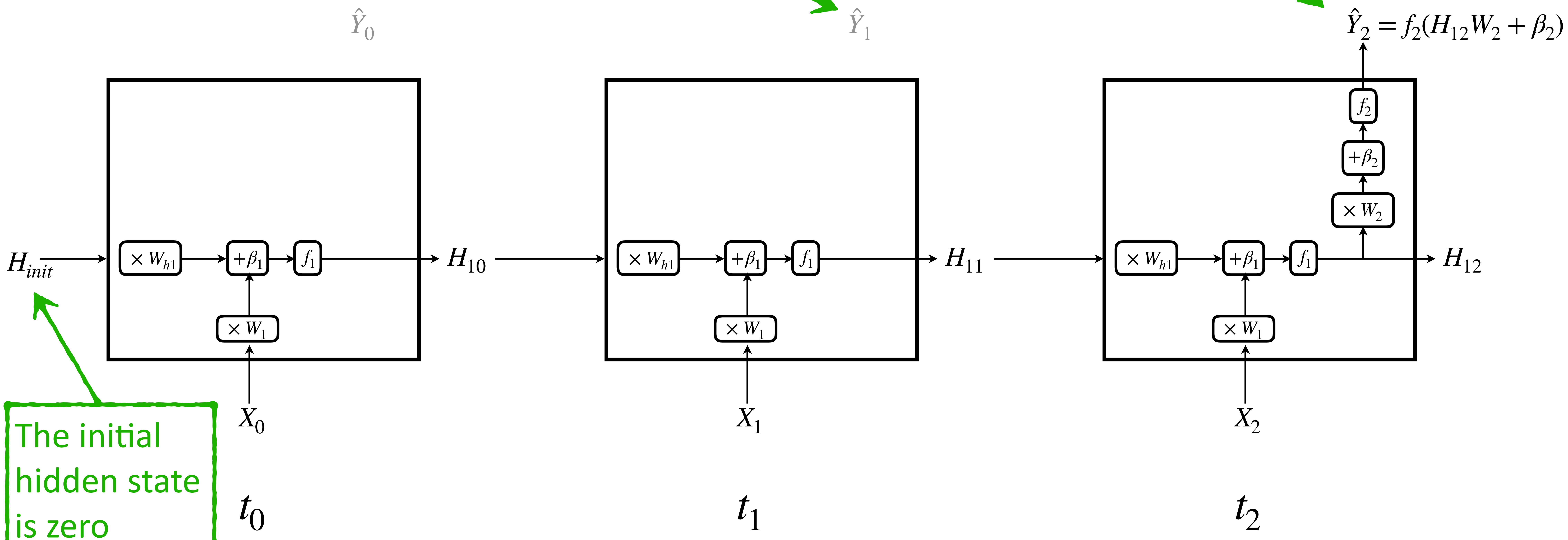
A Single RNN Layer

Lets look at training a **Sequence to Vector RNN** unrolled over 3 time steps

Sequence to Vector RNN over 3 Time Steps

Outputs at time steps t_0 and t_1 are not computed

The RNN only produces an output at the last time step



The initial hidden state is zero

Sequence to Vector RNN over 3 Time Steps

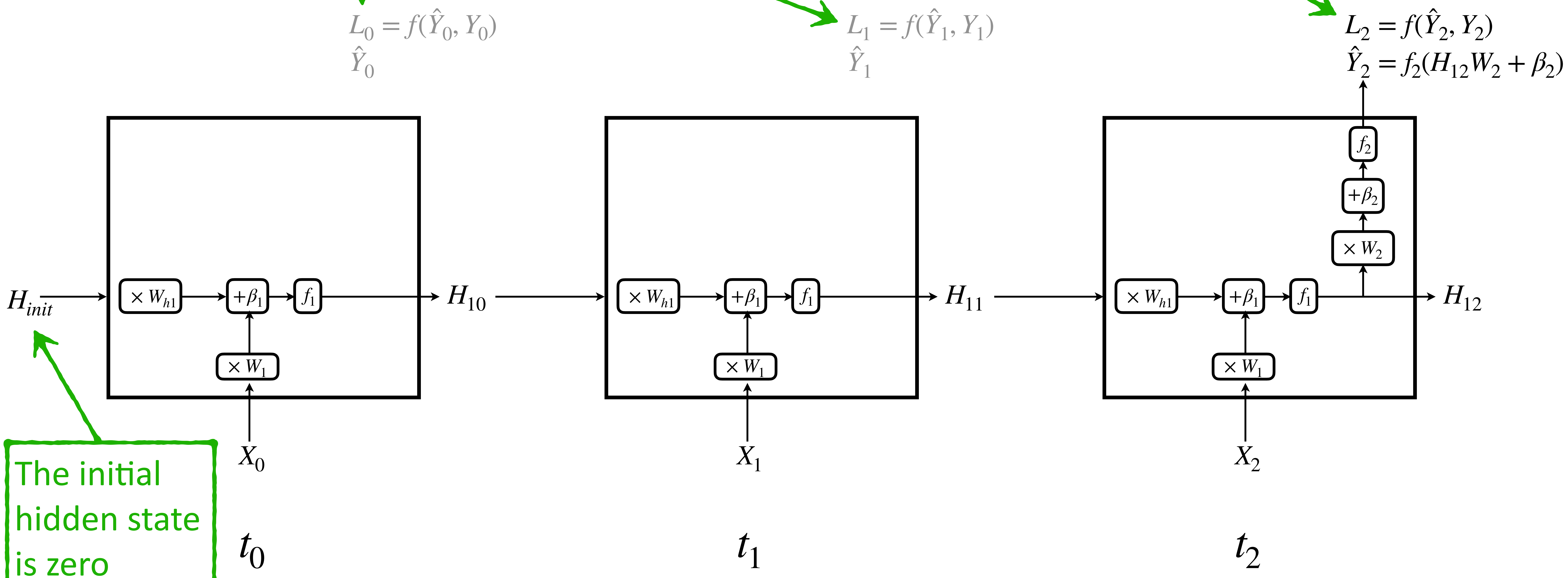
Loss at time steps t_0 and t_1 are not computed

The loss is only computed at the last time step

Total Loss

$$L = L_0 + L_1 + L_2$$

$$\Rightarrow L = L_2$$



The initial hidden state is zero

Let's walk through how we train this RNN unrolled over 3 time steps

Training via Gradient Descent involves a **Forward Pass**, Computing the **Cost Function**, **Backpropagation** and **Parameter Updates**

H_{init}

Backpropagation in an RNN must be done over multiple time steps. The algorithm is called **Backpropagation Through Time (BPTT)**

t_0 t_1 t_2

Sequence to Vector RNN over 3 Time Steps

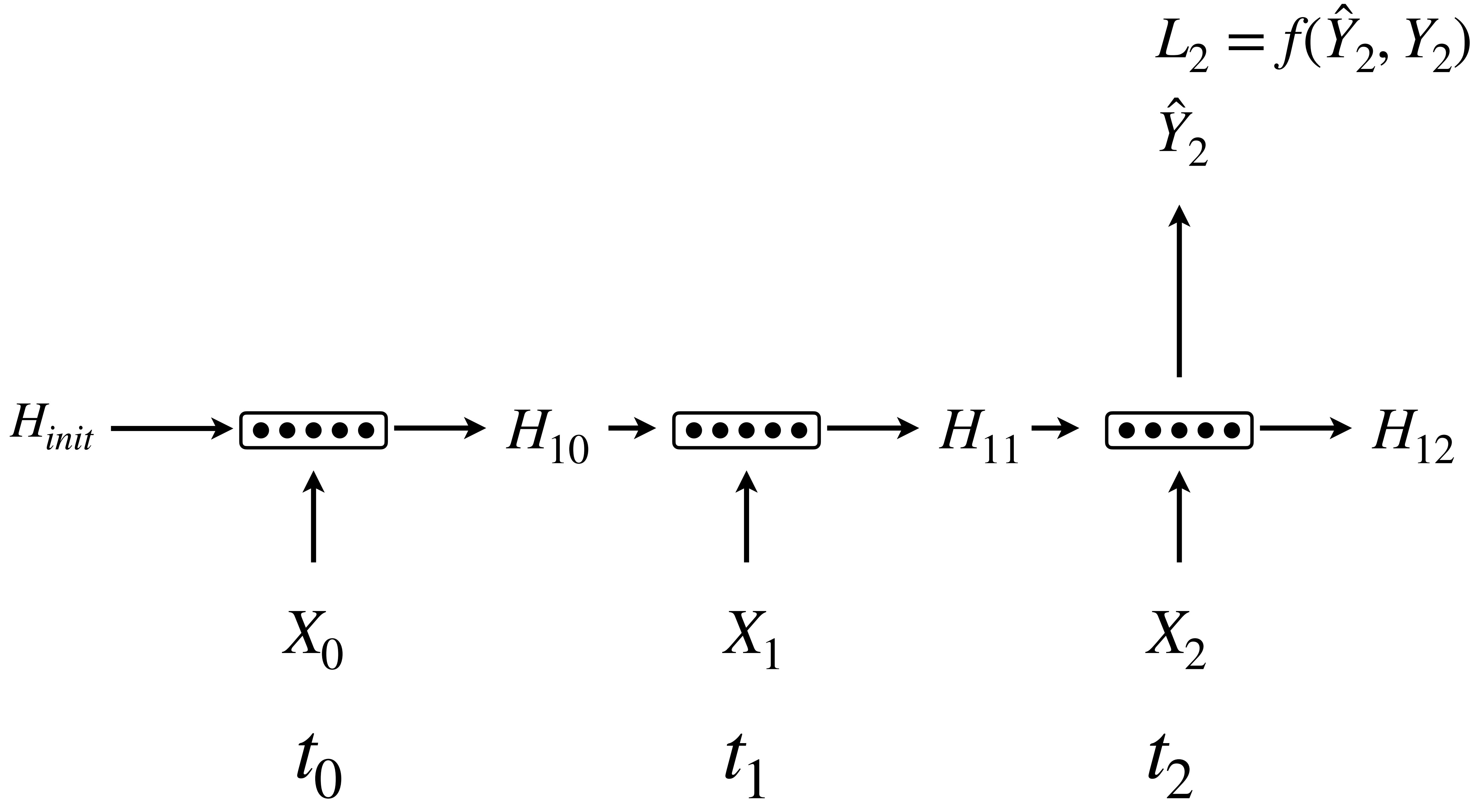
A Sequence to Vector RNN (many-to-one) only produces an output and Loss at the last time step.

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

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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



Forward Propagation

Only computes the output at the last time step

$$H_{10} = f_1(X_0 W_1 + H_{init} W_{h1} + \beta_1)$$
$$H_{11} = f_1(X_1 W_1 + H_{10} W_{h1} + \beta_1)$$
$$H_{12} = f_1(X_2 W_1 + H_{11} W_{h1} + \beta_1)$$
$$\hat{Y}_2 = f_2(H_{12} W_2 + \beta_2)$$

Loss Function

Loss function can be Categorical Cross Entropy or Binary Cross Entropy

$$L_2 = f(\hat{Y}_2, Y_2)$$
$$L = L_2$$

Total Loss is the loss at the last time step

Example Loss Functions

$$L = -\,[y\, log_e \hat{y} + (1-y)\, log_e(1-\hat{y})]$$

Binary Cross Entropy

$$L = -\sum_{j=1}^{K} y_j\, log_e \hat{y}_j$$

Categorical Cross Entropy

Sequence to Vector RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

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

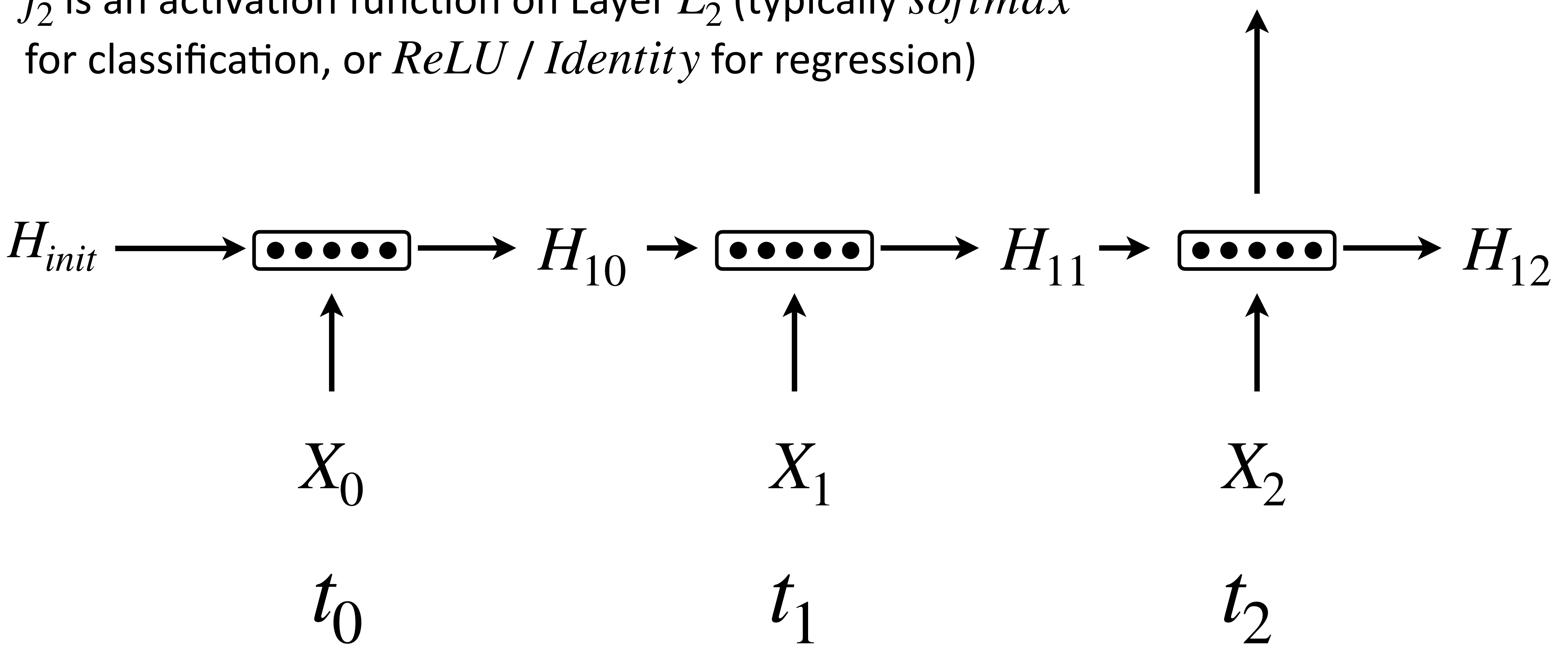
f_1 is an activation function on Layer L_1 (typically $tanh$)

$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$

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

$L_2 = f(\hat{Y}_2, Y_2)$

$\hat{Y}_2$



Backpropagation

Output layer gradients are only from the final time step

$$\Rightarrow \frac{\partial}{\partial \beta_2}L = \frac{\partial}{\partial \beta_2}L_2$$

$$\Rightarrow \frac{\partial}{\partial W_2}L = \frac{\partial}{\partial W_2}L_2$$

Sequence to Vector RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

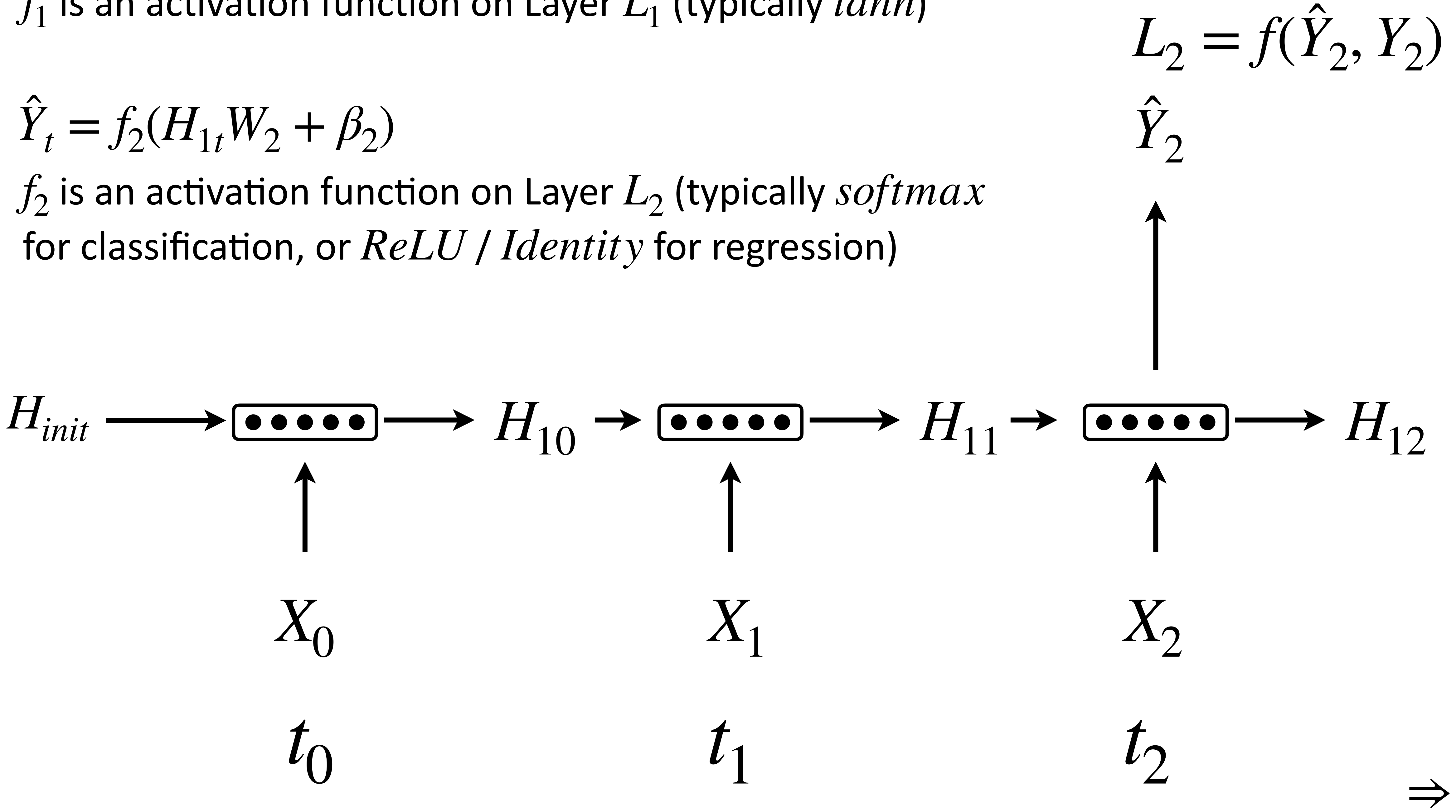
$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

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

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

$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$

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



Backpropagation Through Time (BPTT)

Hidden layer gradients are derivatives of Loss from the final time step

$$\Rightarrow \frac{\partial}{\partial \beta_1}L = \frac{\partial}{\partial \beta_1}L_2$$

Chain Rule. H_{12} depends on β_1

$$\Rightarrow \frac{\partial}{\partial \beta_1}L = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial \beta_1}H_{12} +$$

Chain Rule. H_{12} depends on H_{11}

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial \beta_1}H_{11} +$$

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial \beta_1}H_{10}$$

Chain Rule. H_{11} depends on H_{10}

BPTT sums the derivatives over all the time steps

$$\Rightarrow \frac{\partial}{\partial \beta_1}L = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \left[\frac{\partial}{\partial \beta_1}H_{12} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial \beta_1}H_{11} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial \beta_1}H_{10} \right]$$

Sequence to Vector RNN over 3 Time Steps

Recurrent Neural Networks

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

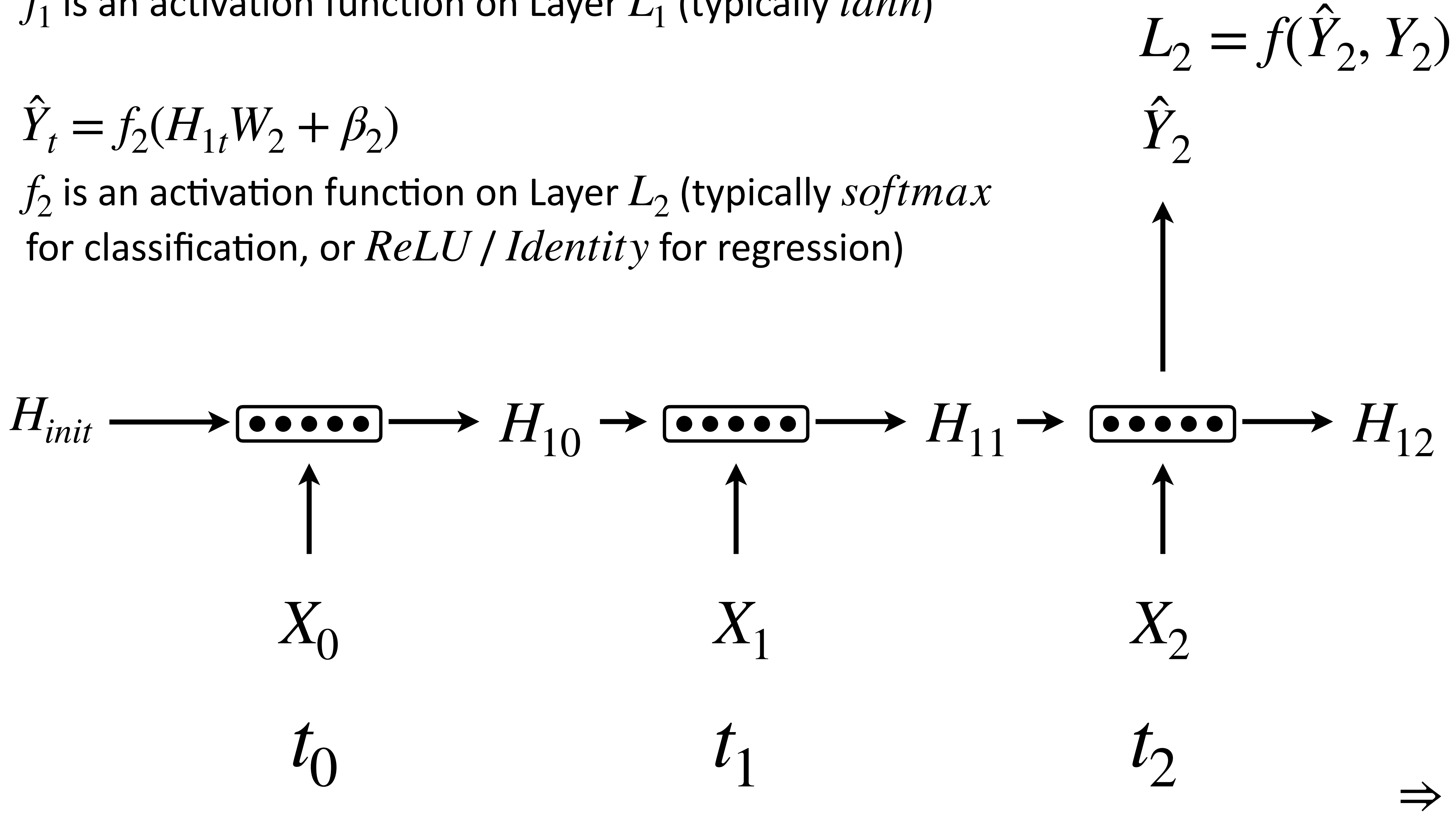
$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

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

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

$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$

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



Backpropagation Through Time (BPTT)

Hidden layer gradients are derivatives of Loss from the final time step

$$\Rightarrow \frac{\partial}{\partial W_1}L = \frac{\partial}{\partial W_1}L_2$$

$$\Rightarrow \frac{\partial}{\partial W_1}L = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial W_1}H_{12} +$$

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_1}H_{11} +$$

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_1}H_{10}$$

Chain Rule. H_{12} depends on W_1

Chain Rule. H_{12} depends on H_{11}

Chain Rule. H_{11} depends on H_{10}

BPTT sums the derivatives over all the time steps

$$\Rightarrow \frac{\partial}{\partial W_1}L = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \left[\frac{\partial}{\partial W_1}H_{12} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_1}H_{11} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_1}H_{10} \right]$$

Sequence to Vector RNN over 3 Time Steps

Recurrent Neural Networks

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

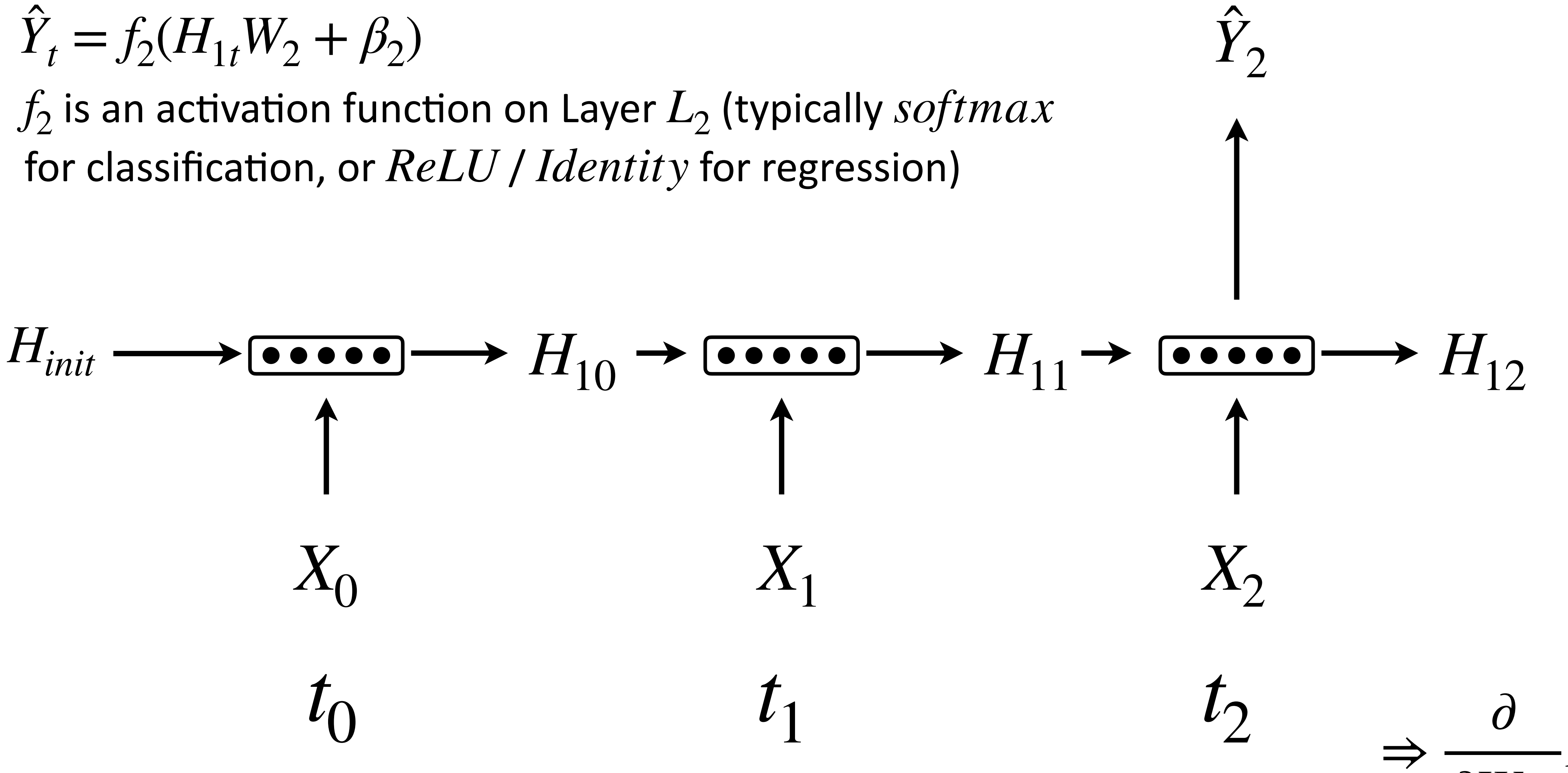
$$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$$

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

$$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$$

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

$$L_2 = f(\hat{Y}_2, Y_2)$$



Backpropagation Through Time (BPTT)

Hidden layer gradients are derivatives of Loss from the final time step

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L = \frac{\partial}{\partial W_{h1}}L_2$$

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial W_{h1}}H_{12} + \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_{h1}}H_{11} + \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_{h1}}H_{10}$$

Chain Rule. H_{12} depends on W_{h1}

Chain Rule. H_{12} depends on H_{11}

Chain Rule. H_{11} depends on H_{10}

BPTT sums the derivatives over all the time steps

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \left[\frac{\partial}{\partial W_{h1}}H_{12} + \frac{\partial}{\partial H_{11}}H_{12}\frac{\partial}{\partial W_{h1}}H_{11} + \frac{\partial}{\partial H_{11}}H_{12}\frac{\partial}{\partial H_{10}}H_{11}\frac{\partial}{\partial W_{h1}}H_{10}\right]$$

Sequence to Vector RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

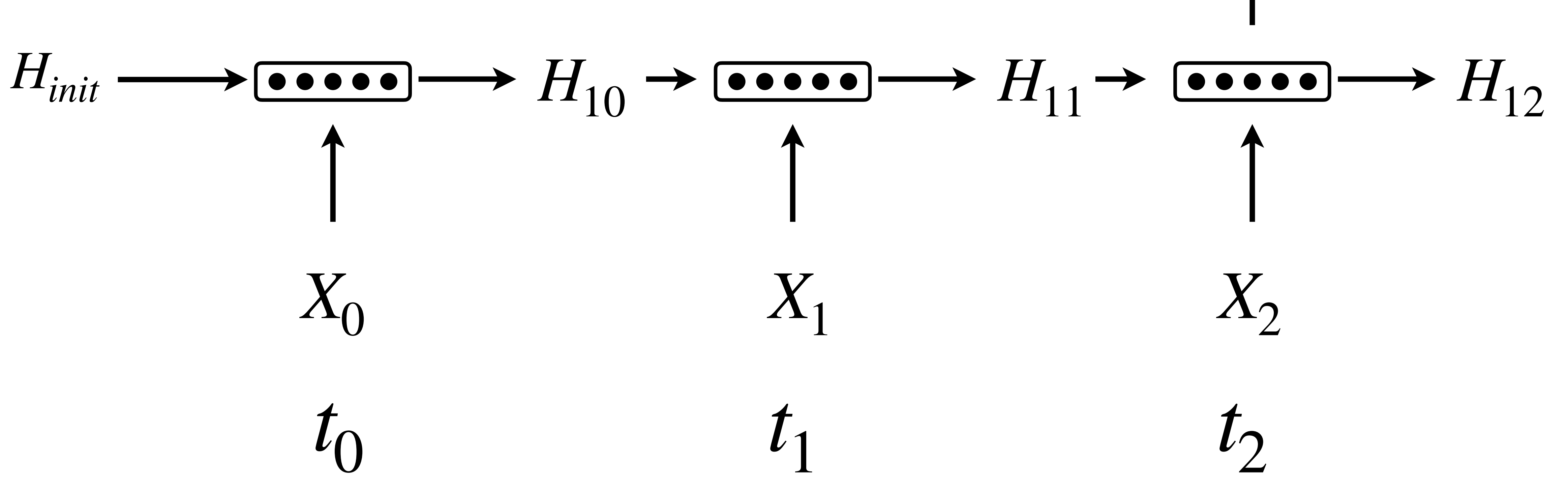
$$\frac{\partial}{\partial W_{h1}} L \quad \frac{\partial}{\partial W_1} L \quad \frac{\partial}{\partial W_2} L \quad \frac{\partial}{\partial \beta_1} L \quad \frac{\partial}{\partial \beta_2} L$$

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

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

$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$

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



Parameter Updates

$$\beta_2 = \beta_2 - \left(\frac{\partial}{\partial \beta_2} L\right) \times learning\_rate$$

$$W_2 = W_2 - \left(\frac{\partial}{\partial W_2} L\right) \times learning\_rate$$

$$\beta_1 = \beta_1 - \left(\frac{\partial}{\partial \beta_1} L\right) \times learning\_rate$$

$$W_1 = W_1 - \left(\frac{\partial}{\partial W_1} L\right) \times learning\_rate$$

$$W_{h1} = W_{h1} - \left(\frac{\partial}{\partial W_{h1}} L\right) \times learning\_rate$$

Sequence to Vector RNN over 3 Time Steps

Recurrent Neural Networks

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

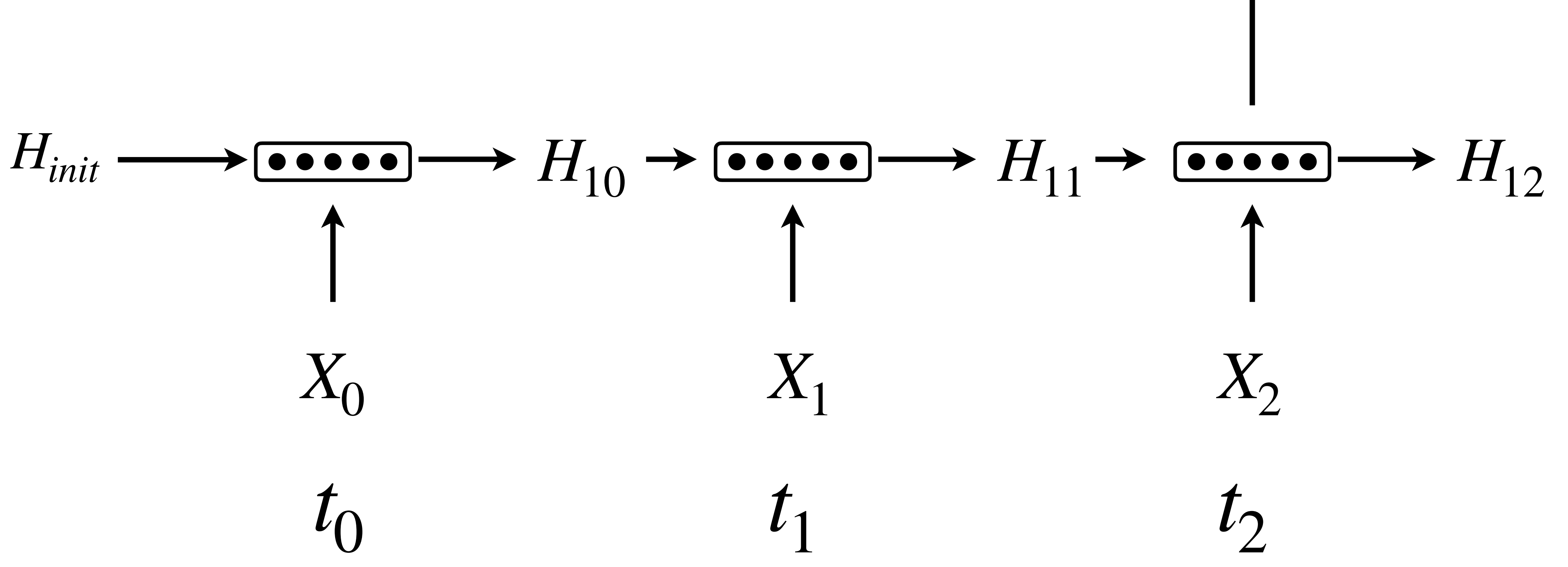
$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

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

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

$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$

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



Gradient Descent for Sequence to Vector RNN

Step 1: Start with initial values for $W_1, W_2, W_{h1}, \beta_1, \beta_2$

Step 2: Forward Propagation…

$$H_{10} = f_1(X_0 W_1 + H_{init} W_{h1} + \beta_1)$$

$$H_{11} = f_1(X_1 W_1 + H_{10} W_{h1} + \beta_1)$$

$$H_{12} = f_1(X_2 W_1 + H_{11} W_{h1} + \beta_1)$$

$$\hat{Y}_2 = f_2(H_{12} W_2 + \beta_2) \qquad L_2 = f(\hat{Y}_2, Y_2)$$

Step 3: Backpropagation Through Time

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

Step 4: Parameter Updates

$$\beta_2 = \beta_2 - \left(\frac{\partial}{\partial \beta_2}L\right) \times learning\_rate$$

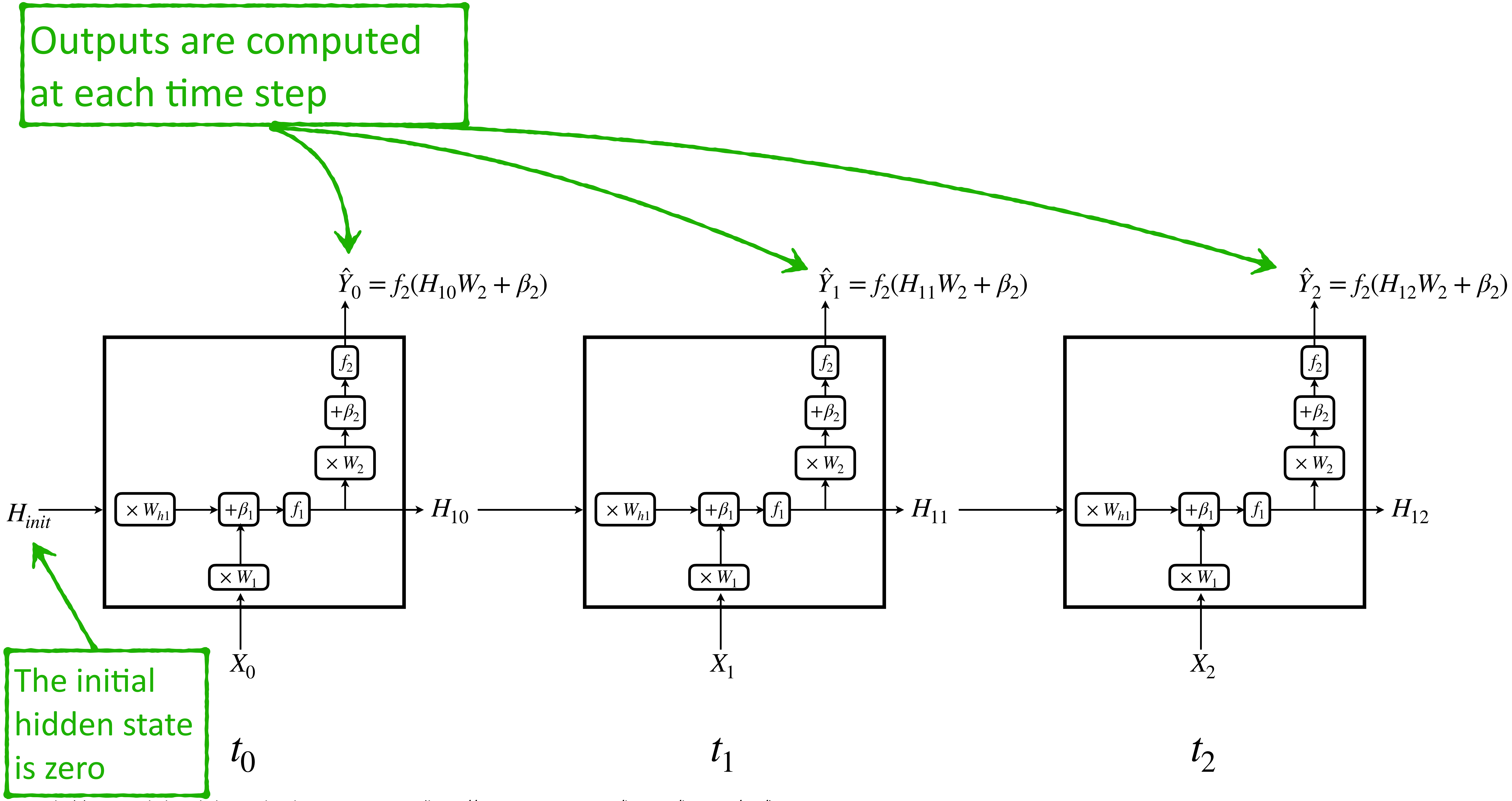
$$\beta_1 = \beta_1 - \left(\frac{\partial}{\partial \beta_1}L\right) \times learning\_rate \qquad W_2 = W_2 - \left(\frac{\partial}{\partial W_2}L\right) \times learning\_rate$$

$$W_1 = W_1 - \left(\frac{\partial}{\partial W_1}L\right) \times learning\_rate \qquad W_{h1} = W_{h1} - \left(\frac{\partial}{\partial W_{h1}}L\right) \times learning\_rate$$

Step 5: Go to step 2 and repeat

Lets look at training a **Sequence to Sequence RNN** unrolled over 3 time steps

Sequence to Sequence RNN over 3 Time Steps

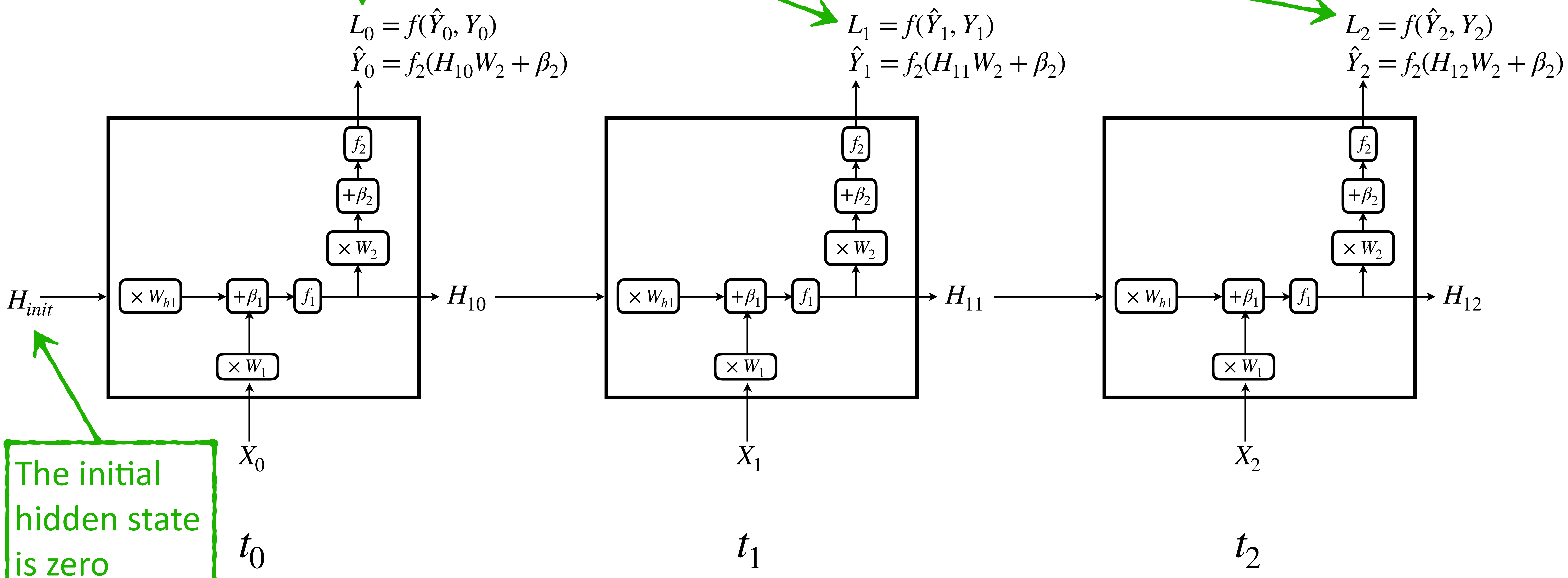


Sequence to Sequence RNN over 3 Time Steps

Loss is computed at each time step

Total Loss

$L = L_0 + L_1 + L_2$



The initial hidden state is zero

Let's walk through how we train this RNN unrolled over 3 time steps

Training via Gradient Descent involves a **Forward Pass**, Computing the **Cost Function**, **Backpropagation** and **Parameter Updates**

Backpropagation in an RNN must be done over multiple time steps. The algorithm is called **Backpropagation Through Time (BPTT)**

H_{init}

t_0 t_1 t_2

Recurrent Neural Networks

Sequence to Sequence RNN over 3 Time Steps

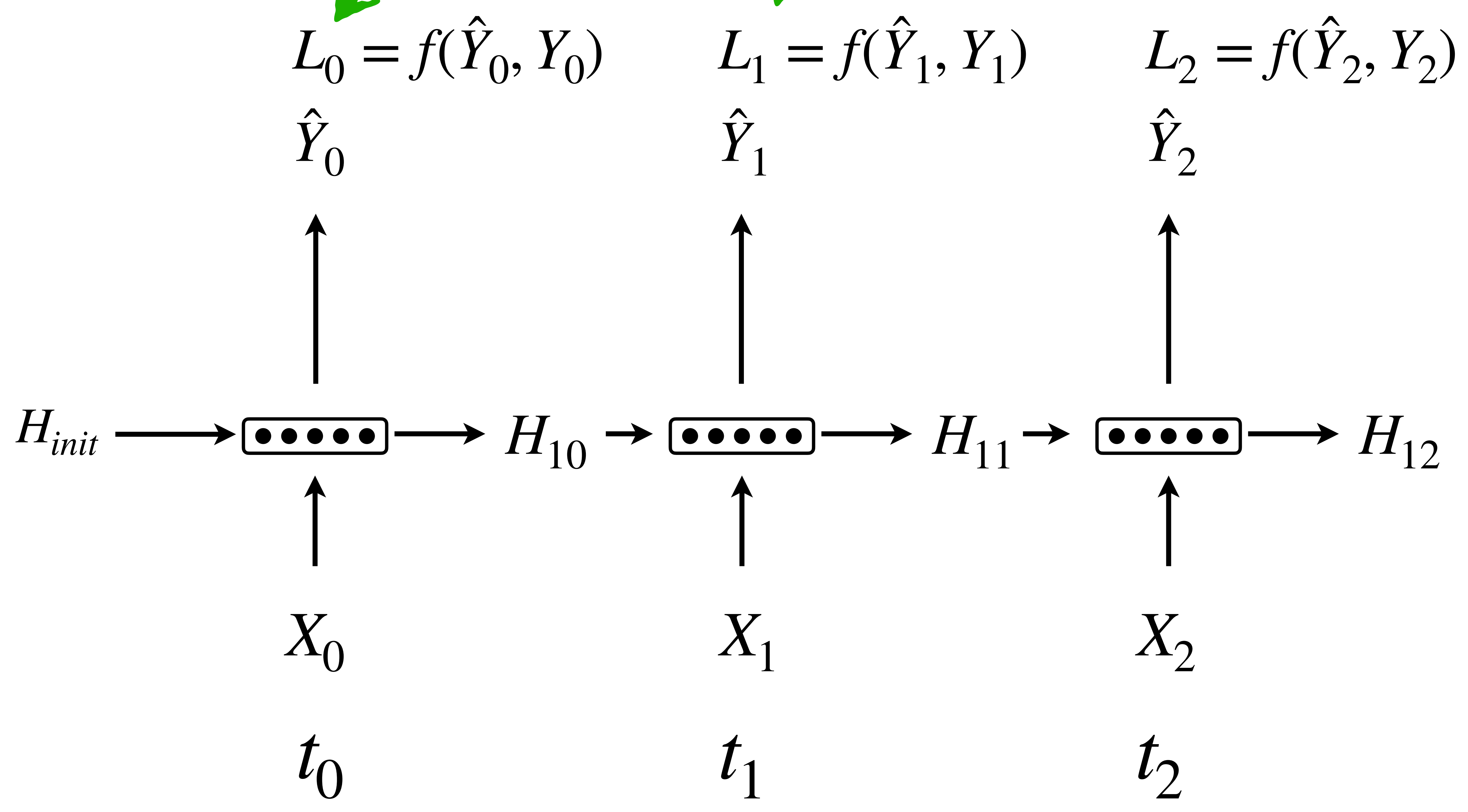
$H_{1t} = f_1(X_t W_1 + H_{1t-1} W_{h1} + \beta_1)$

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

$\hat{Y}_t = f_2(H_{1t} W_2 + \beta_2)$

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

A Sequence to Sequence RNN (many-to-many) produces an output and Loss at every time step.



Forward Propagation

Computes the output at every time step

$$H_{10} = f_1(X_0 W_1 + H_{init} W_{h1} + \beta_1) \qquad \hat{Y}_0 = f_2(H_{10} W_2 + \beta_2)$$

$$H_{11} = f_1(X_1 W_1 + H_{10} W_{h1} + \beta_1) \qquad \hat{Y}_1 = f_2(H_{11} W_2 + \beta_2)$$

$$H_{12} = f_1(X_2 W_1 + H_{11} W_{h1} + \beta_1) \qquad \hat{Y}_2 = f_2(H_{12} W_2 + \beta_2)$$

Loss Function

Loss function can be Categorical Cross Entropy or Binary Cross Entropy

$$L_0 = f(\hat{Y}_0, Y_0) \qquad L_1 = f(\hat{Y}_1, Y_1) \qquad L_2 = f(\hat{Y}_2, Y_2)$$

$$L = L_0 + L_1 + L_2$$

Total Loss is the sum of the losses at each time step

Example Loss Functions

$$L = -[y\, log_e \hat{y} + (1-y)\, log_e(1-\hat{y})]$$

Binary Cross Entropy

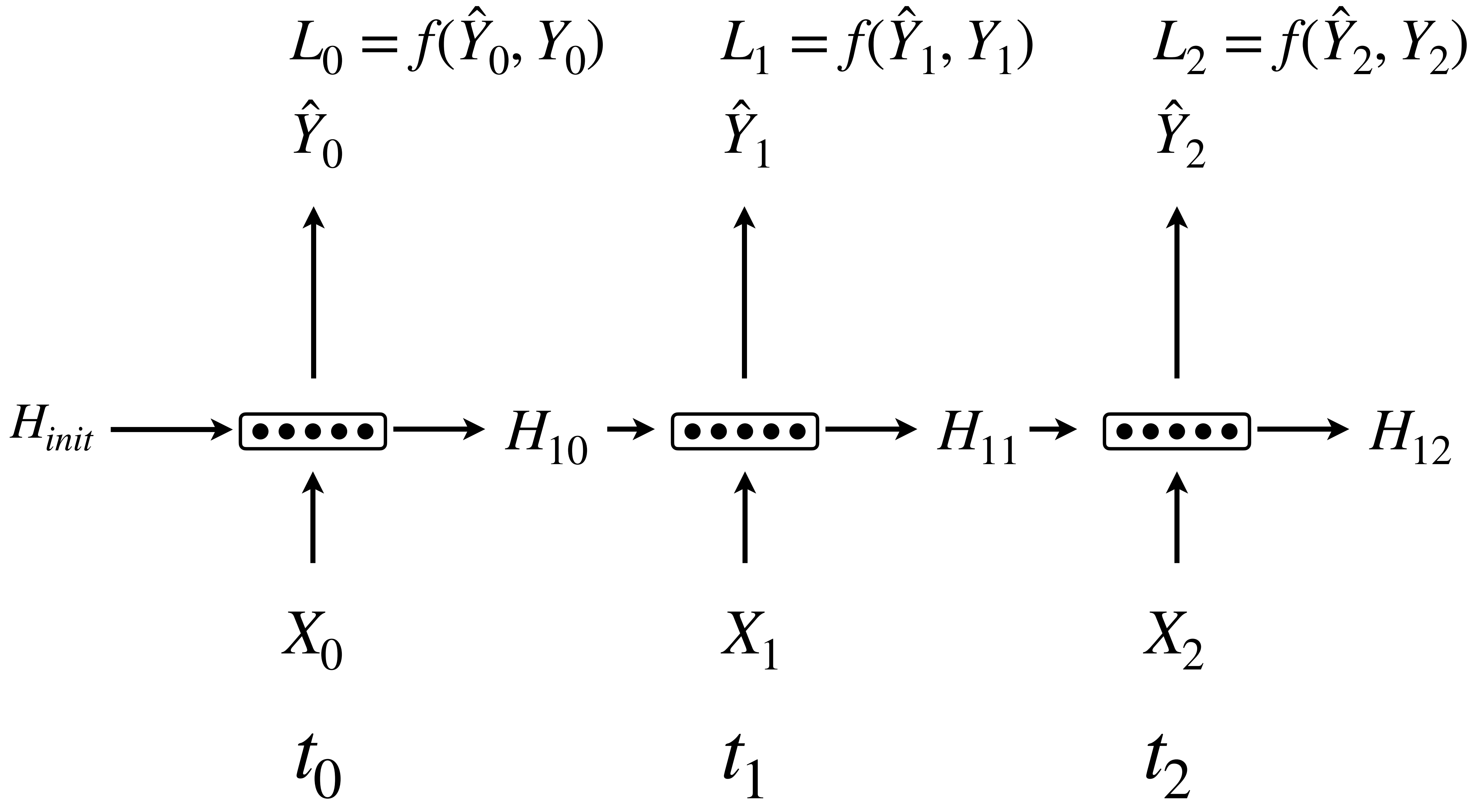
$$L = -\sum_{j=1}^{K} y_j\, log_e \hat{y}_j$$

Categorical Cross Entropy

Sequence to Sequence RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$



Backpropagation

Output layer gradients are summed from each time step

$$\Rightarrow \frac{\partial}{\partial \beta_2}L = \frac{\partial}{\partial \beta_2}L_0 + \frac{\partial}{\partial \beta_2}L_1 + \frac{\partial}{\partial \beta_2}L_2$$

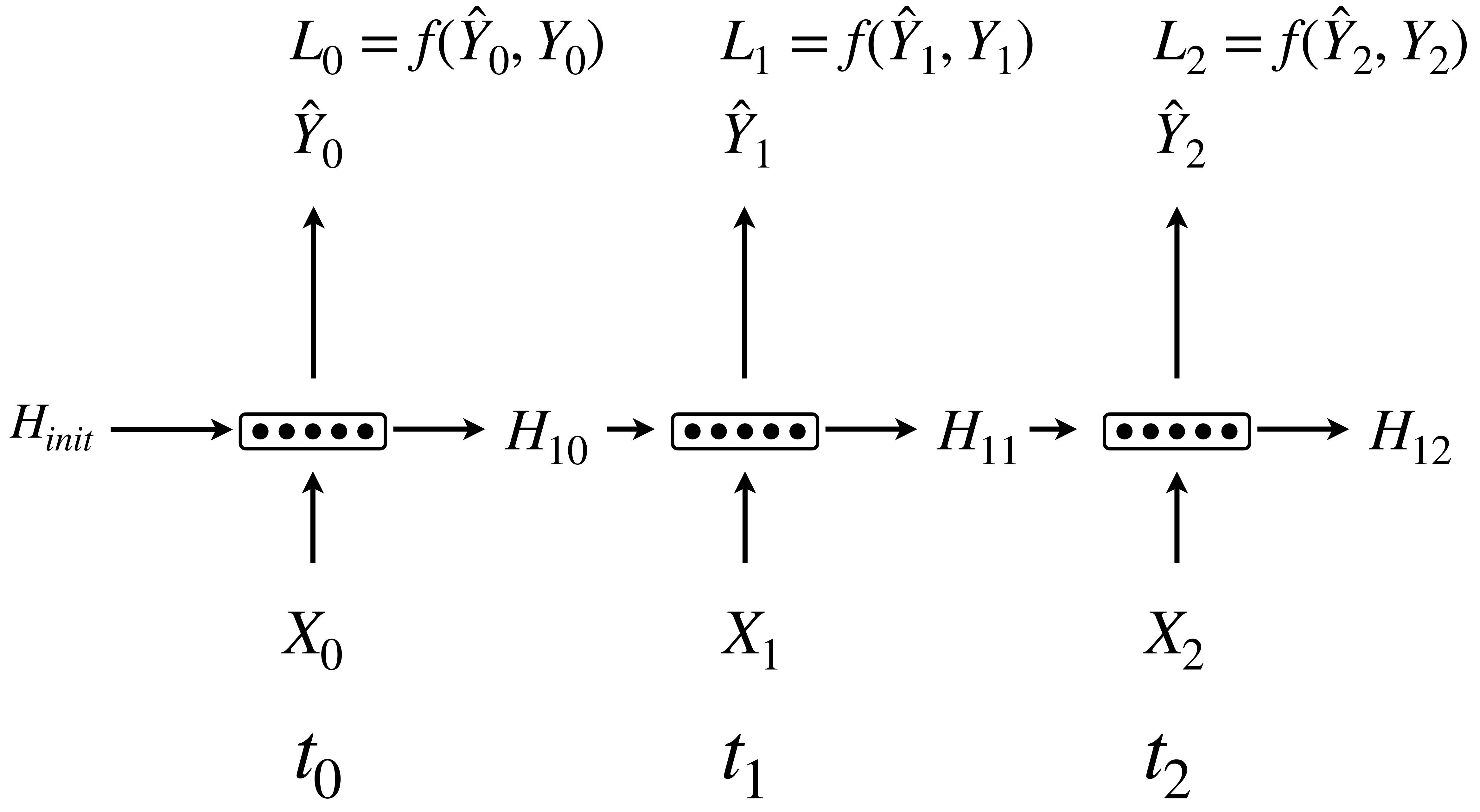
$$\Rightarrow \frac{\partial}{\partial W_2}L = \frac{\partial}{\partial W_2}L_0 + \frac{\partial}{\partial W_2}L_1 + \frac{\partial}{\partial W_2}L_2 +$$

Recurrent Neural Networks

Sequence to Sequence RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$



Backpropagation Through Time (BPTT)

Hidden layer gradients are the sum of the derivatives of Loss from each time step

$$\Rightarrow \frac{\partial}{\partial \beta_1}L = \frac{\partial}{\partial \beta_1}L_0 + \frac{\partial}{\partial \beta_1}L_1 + \frac{\partial}{\partial \beta_1}L_2$$

$$\Rightarrow \frac{\partial}{\partial \beta_1}L_0 = \frac{\partial}{\partial \hat{Y}_0}L_0 \frac{\partial}{\partial H_{10}}\hat{Y}_0 \frac{\partial}{\partial \beta_1}H_{10} +$$

Chain Rule. H_{10} depends on β_1

$$\Rightarrow \frac{\partial}{\partial \beta_1}L_1 = \frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \frac{\partial}{\partial \beta_1}H_{11} +$$

Chain Rule. H_{11} depends on β_1

$$\frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial \beta_1}H_{10} +$$

Chain Rule. H_{11} depends on H_{10}

$$\Rightarrow \frac{\partial}{\partial \beta_1}L_1 = \frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \left[\frac{\partial}{\partial \beta_1}H_{11} + \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial \beta_1}H_{10} + \right]$$

$$\Rightarrow \frac{\partial}{\partial \beta_1}L_2 = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial \beta_1}H_{12} +$$

Chain Rule. H_{12} depends on β_1

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial \beta_1}H_{11} +$$

Chain Rule. H_{12} depends on H_{11}

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial \beta_1}H_{10}$$

Chain Rule. H_{11} depends on H_{10}

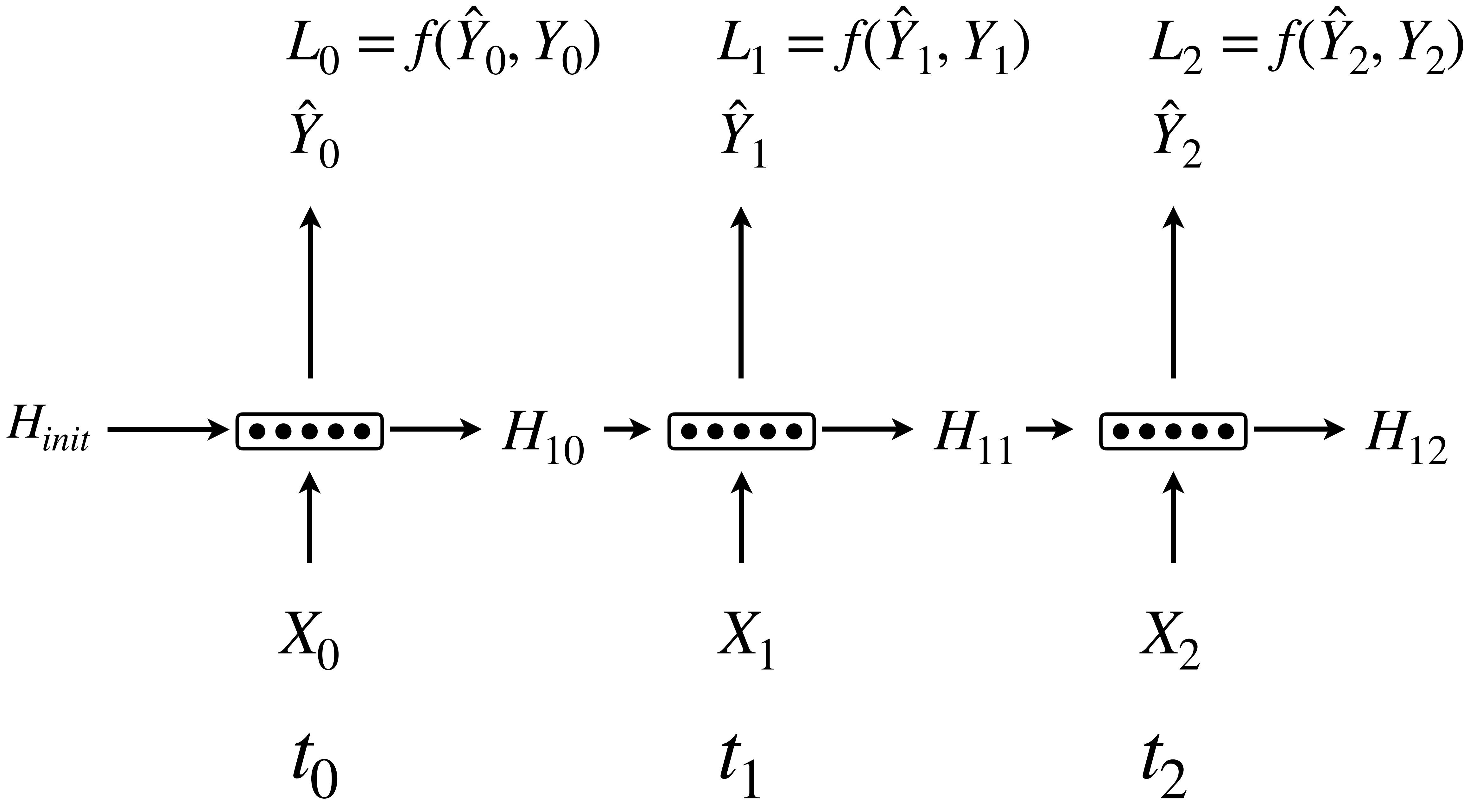
$$\Rightarrow \frac{\partial}{\partial \beta_1}L_2 = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \left[\frac{\partial}{\partial \beta_1}H_{12} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial \beta_1}H_{11} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial \beta_1}H_{10}\right]$$

Recurrent Neural Networks

Sequence to Sequence RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$



Backpropagation Through Time (BPTT)

Hidden layer gradients are the sum of the derivatives of Loss from each time step

$$\Rightarrow \frac{\partial}{\partial W_1}L = \frac{\partial}{\partial W_1}L_0 + \frac{\partial}{\partial W_1}L_1 + \frac{\partial}{\partial W_1}L_2$$

$$\Rightarrow \frac{\partial}{\partial W_1}L_0 = \frac{\partial}{\partial \hat{Y}_0}L_0 \frac{\partial}{\partial H_{10}}\hat{Y}_0 \frac{\partial}{\partial W_1}H_{10} +$$

Chain Rule. H_{10} depends on W_1

$$\Rightarrow \frac{\partial}{\partial W_1}L_1 = \frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \frac{\partial}{\partial W_1}H_{11} +$$

Chain Rule. H_{11} depends on W_1

$$\frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_1}H_{10} +$$

Chain Rule. H_{11} depends on H_{10}

$$\Rightarrow \frac{\partial}{\partial W_1}L_1 = \frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \left[\frac{\partial}{\partial W_1}H_{11} + \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_1}H_{10} + \right]$$

$$\Rightarrow \frac{\partial}{\partial W_1}L_2 = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial W_1}H_{12} +$$

Chain Rule. H_{12} depends on W_1

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_1}H_{11} +$$

Chain Rule. H_{12} depends on H_{11}

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_1}H_{10}$$

Chain Rule. H_{11} depends on H_{10}

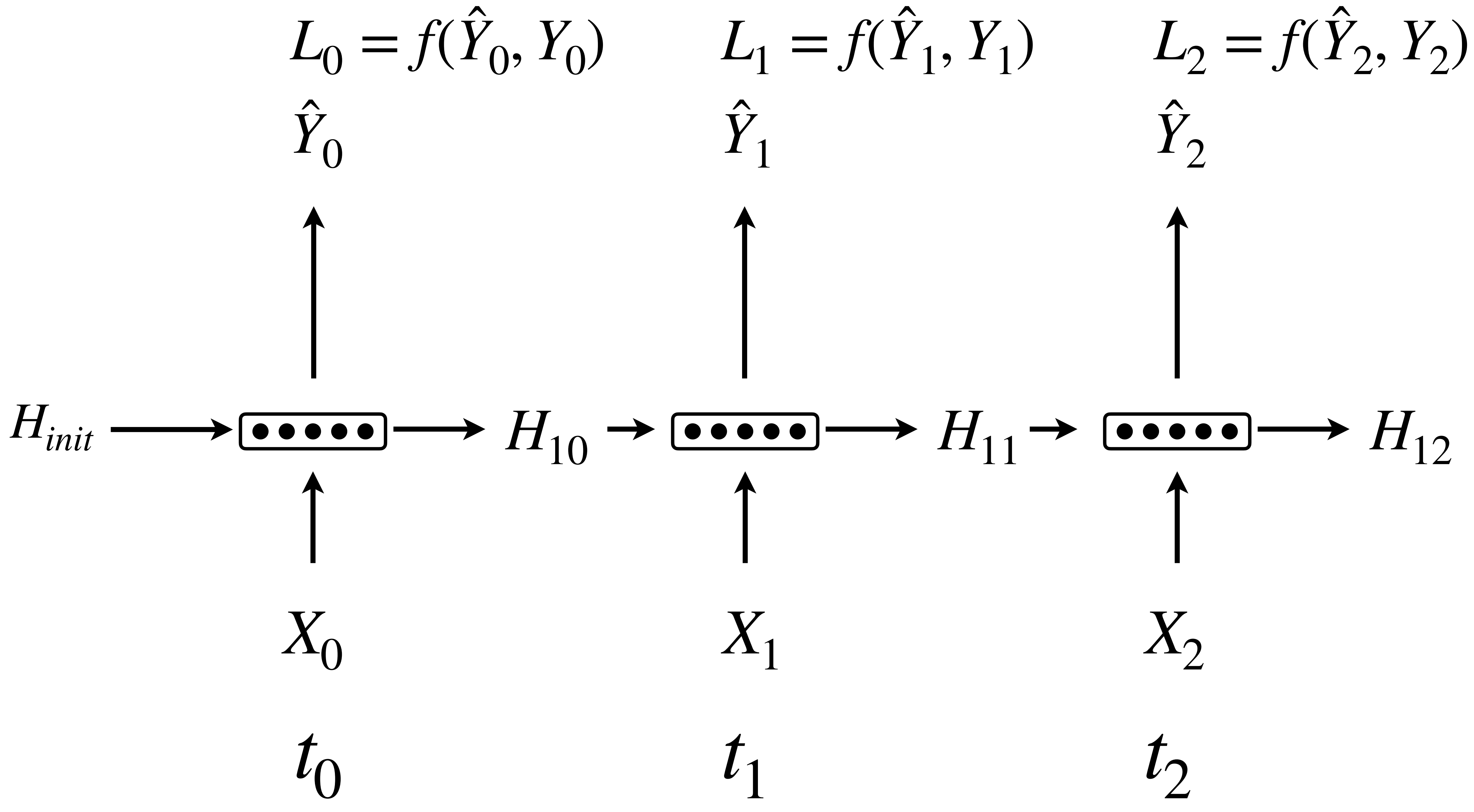
$$\Rightarrow \frac{\partial}{\partial W_1}L_2 = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \left[\frac{\partial}{\partial W_1}H_{12} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_1}H_{11} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_1}H_{10} \right]$$

Sequence to Sequence RNN over 3 Time Steps

Recurrent Neural Networks

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$



Backpropagation Through Time (BPTT)

Hidden layer gradients are the sum of the derivatives of Loss from each time step

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L = \frac{\partial}{\partial W_{h1}}L_0 + \frac{\partial}{\partial W_{h1}}L_1 + \frac{\partial}{\partial W_{h1}}L_2$$

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L_0 = \frac{\partial}{\partial \hat{Y}_0}L_0 \frac{\partial}{\partial H_{10}}\hat{Y}_0 \frac{\partial}{\partial W_{h1}}H_{10} +$$

Chain Rule. H_{10} depends on W_{h1}

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L_1 = \frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \frac{\partial}{\partial W_{h1}}H_{11} +$$

Chain Rule. H_{11} depends on W_{h1}

$$\frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_{h1}}H_{10} +$$

Chain Rule. H_{11} depends on H_{10}

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L_1 = \frac{\partial}{\partial \hat{Y}_1}L_1 \frac{\partial}{\partial H_{11}}\hat{Y}_1 \left[\frac{\partial}{\partial W_{h1}}H_{11} + \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_{h1}}H_{10} + \right]$$

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L_2 = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial W_{h1}}H_{12} +$$

Chain Rule. H_{12} depends on W_{h1}

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_{h1}}H_{11} +$$

Chain Rule. H_{12} depends on H_{11}

$$\frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_{h1}}H_{10}$$

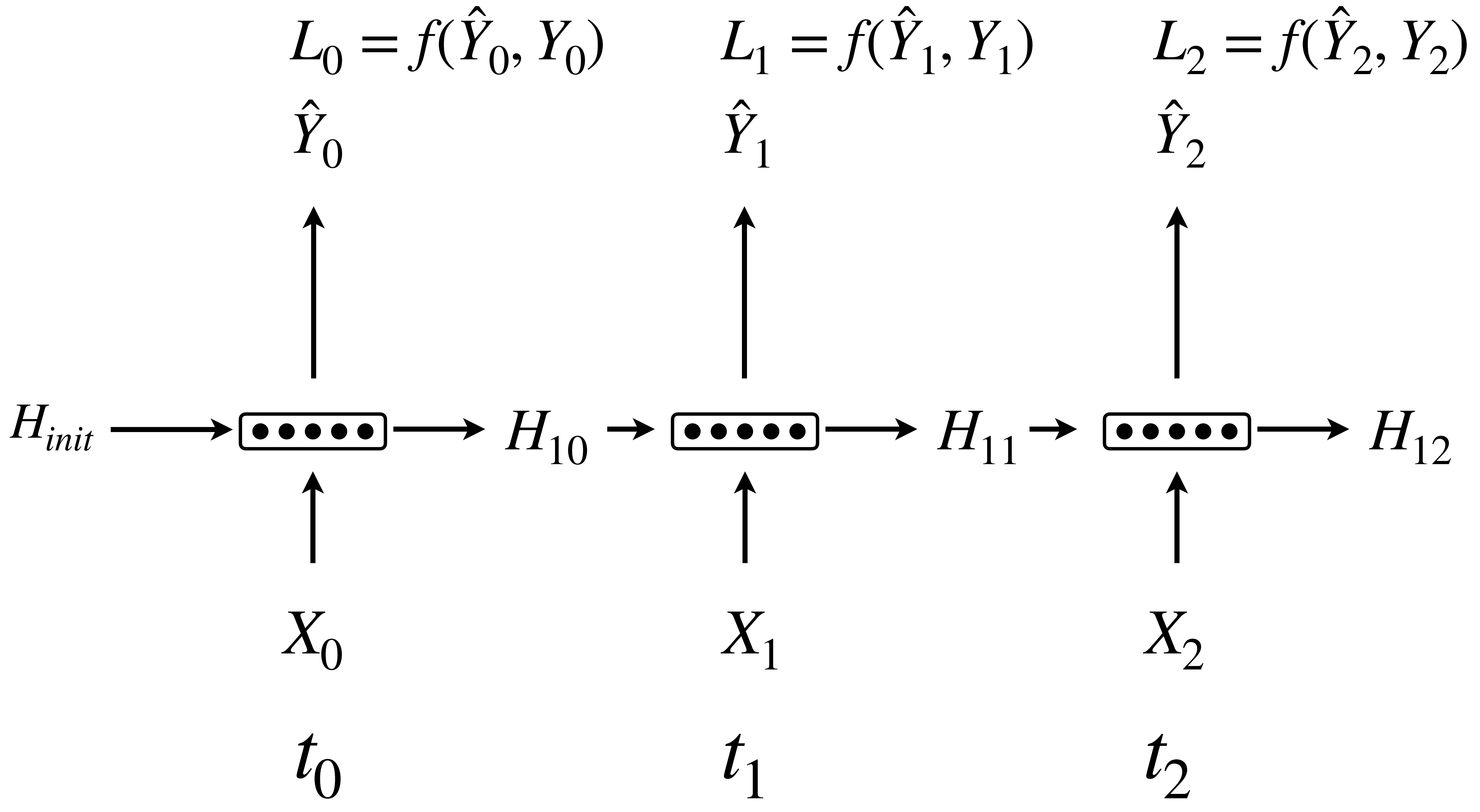
Chain Rule. H_{11} depends on H_{10}

$$\Rightarrow \frac{\partial}{\partial W_{h1}}L_2 = \frac{\partial}{\partial \hat{Y}_2}L_2 \frac{\partial}{\partial H_{12}}\hat{Y}_2 \left[\frac{\partial}{\partial W_{h1}}H_{12} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial W_{h1}}H_{11} + \frac{\partial}{\partial H_{11}}H_{12} \frac{\partial}{\partial H_{10}}H_{11} \frac{\partial}{\partial W_{h1}}H_{10} \right]$$

Sequence to Sequence RNN over 3 Time Steps

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$



Parameter Updates

$$\beta_2 = \beta_2 - \left(\frac{\partial}{\partial \beta_2}L\right) \times learning\_rate$$

$$W_2 = W_2 - \left(\frac{\partial}{\partial W_2}L\right) \times learning\_rate$$

$$\beta_1 = \beta_1 - \left(\frac{\partial}{\partial \beta_1}L\right) \times learning\_rate$$

$$W_1 = W_1 - \left(\frac{\partial}{\partial W_1}L\right) \times learning\_rate$$

$$W_{h1} = W_{h1} - \left(\frac{\partial}{\partial W_{h1}}L\right) \times learning\_rate$$

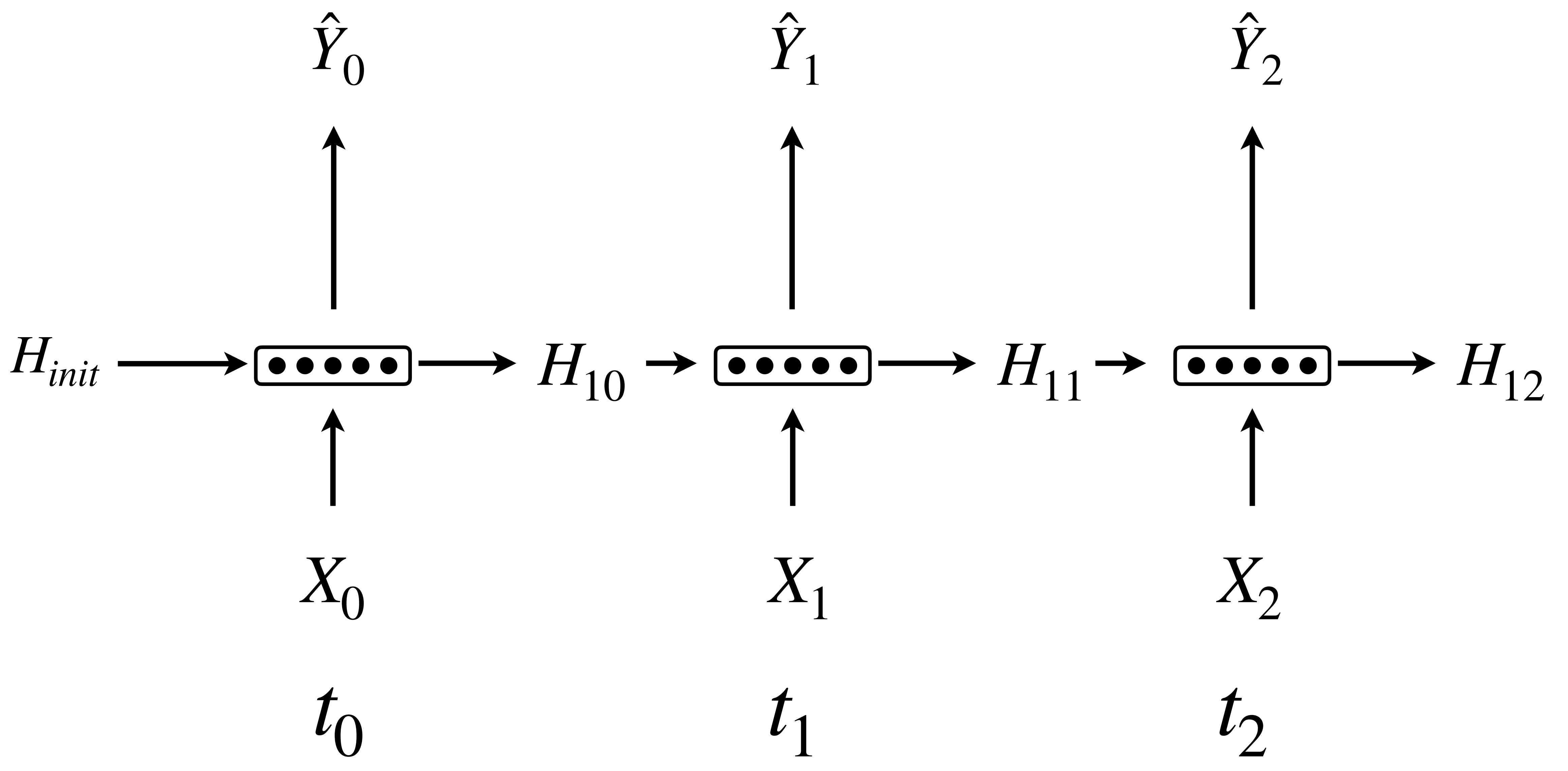
Sequence to Sequence RNN over 3 Time Steps

Recurrent Neural Networks

For backpropagation we have to calculate the partial derivative of the Loss function w.r.t the parameters $W_{h1}, W_1, W_2, \beta_1, \beta_2$

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

$$L_0 = f(\hat{Y}_0, Y_0) \qquad L_1 = f(\hat{Y}_1, Y_1) \qquad L_2 = f(\hat{Y}_2, Y_2)$$



Gradient Descent for Sequence to Sequence RNN

Step 1: Start with initial values for $W_1, W_2, W_{h1}, \beta_1, \beta_2$

Step 2: Forward Propagation…

$$H_{10} = f_1(X_0 W_1 + H_{init} W_{h1} + \beta_1) \qquad \hat{Y}_0 = f_2(H_{10} W_2 + \beta_2)$$

$$H_{11} = f_1(X_1 W_1 + H_{10} W_{h1} + \beta_1) \qquad \hat{Y}_1 = f_2(H_{11} W_2 + \beta_2)$$

$$H_{12} = f_1(X_2 W_1 + H_{11} W_{h1} + \beta_1) \qquad \hat{Y}_2 = f_2(H_{12} W_2 + \beta_2)$$

$$L = L_0 + L_1 + L_2$$

Step 3: Backpropagation Through Time

$$\frac{\partial}{\partial W_{h1}}L \quad \frac{\partial}{\partial W_1}L \quad \frac{\partial}{\partial W_2}L \quad \frac{\partial}{\partial \beta_1}L \quad \frac{\partial}{\partial \beta_2}L$$

Step 4: Parameter Updates

$$\beta_2 = \beta_2 - \left(\frac{\partial}{\partial \beta_2}L\right) \times learning\_rate$$

$$\beta_1 = \beta_1 - \left(\frac{\partial}{\partial \beta_1}L\right) \times learning\_rate \qquad W_2 = W_2 - \left(\frac{\partial}{\partial W_2}L\right) \times learning\_rate$$

$$W_1 = W_1 - \left(\frac{\partial}{\partial W_1}L\right) \times learning\_rate \qquad W_{h1} = W_{h1} - \left(\frac{\partial}{\partial W_{h1}}L\right) \times learning\_rate$$

Step 5: Go to step 2 and repeat

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

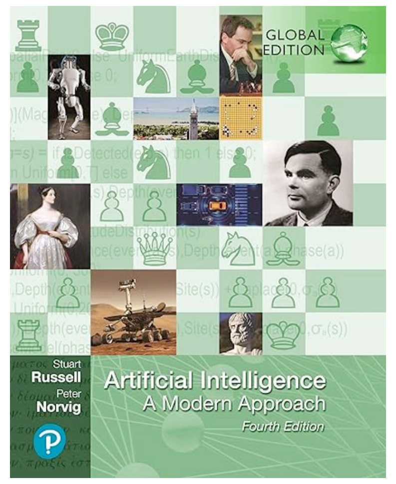
Forward and Back Propagation in Neural Networks

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