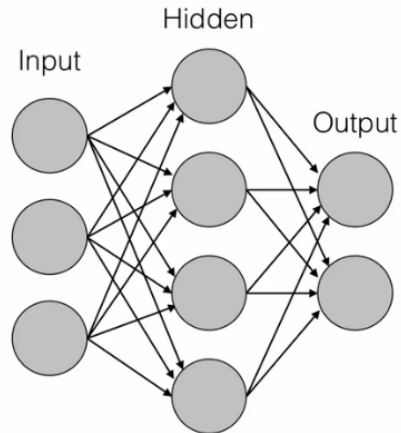


Recurrent Neural Networks

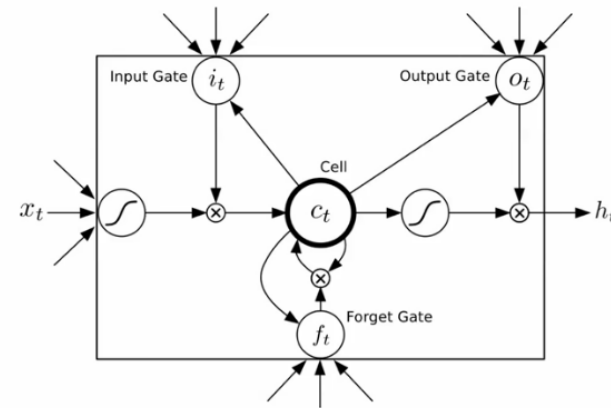
Lecture-6

Why RNN

- ✓ When to use: Patterns in your data change with time
- ✓ Training: GPU (instead of CPU); 1 day vs 8 months

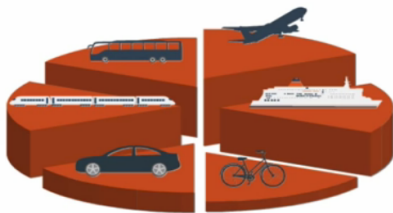


- Independence
- Fixed Length

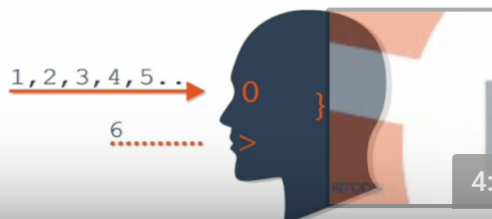


- Temporal dependencies
- Variable sequence length

Feedforward net = Classifier/Regressor



Recurrent Net = Forecaster



BRAINCHILD OF JURGEN SCHMIDHUBER & SEPP HOCHREITER



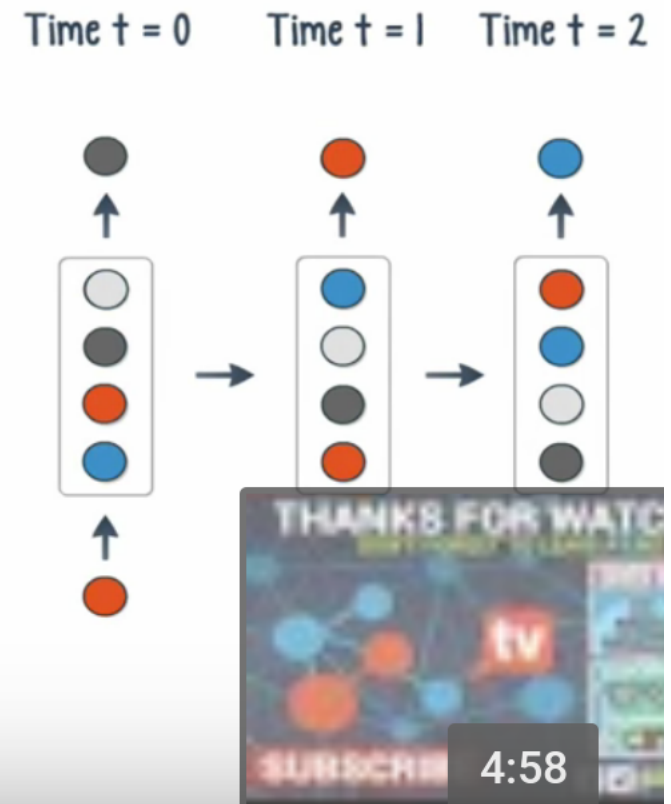
Speech Recognition



Driverless Cars

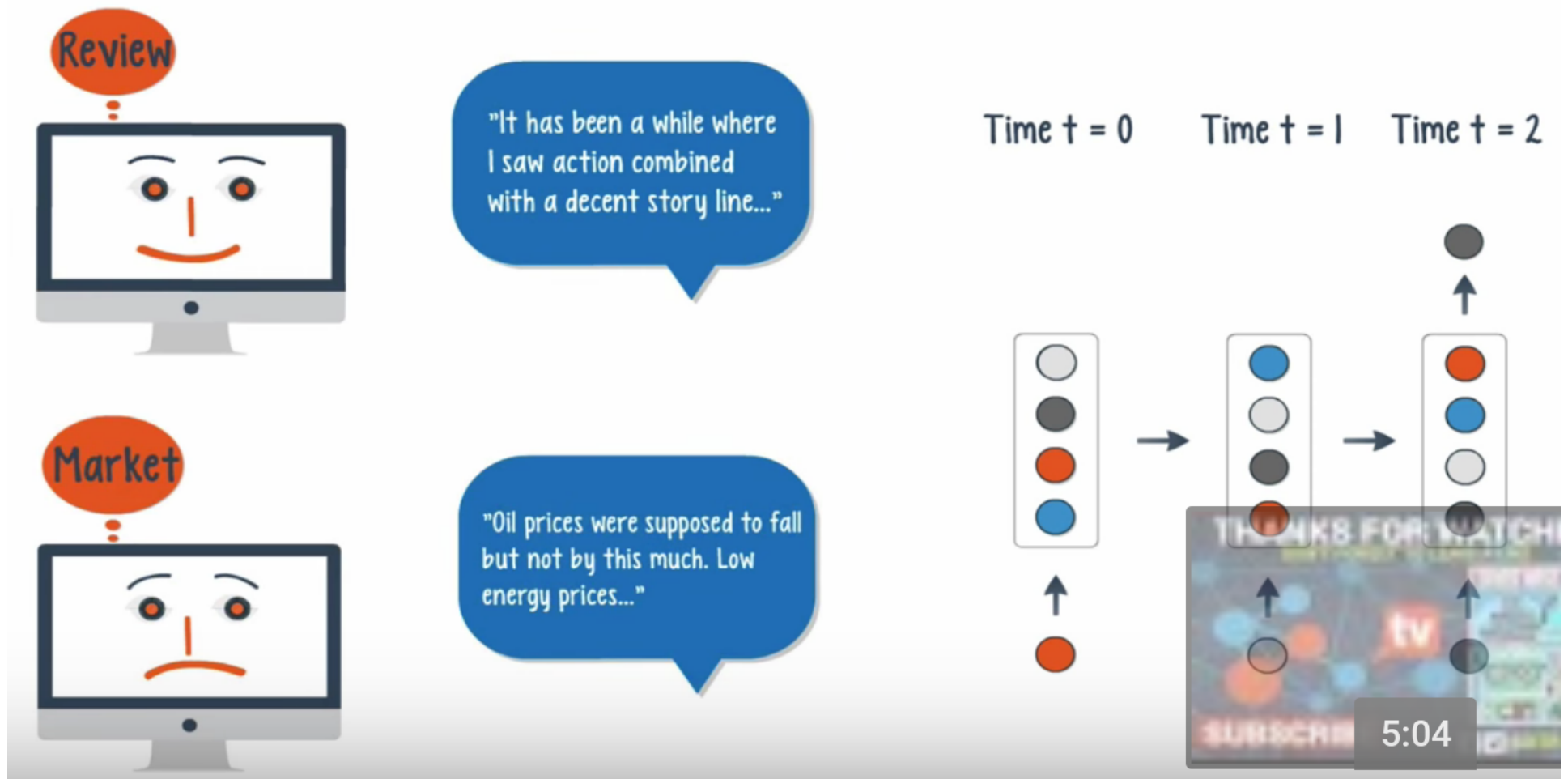
RNN Uses

- ✓ Input: Singular
- ✓ Output: Sequence
- ✓ Application: Image captioning



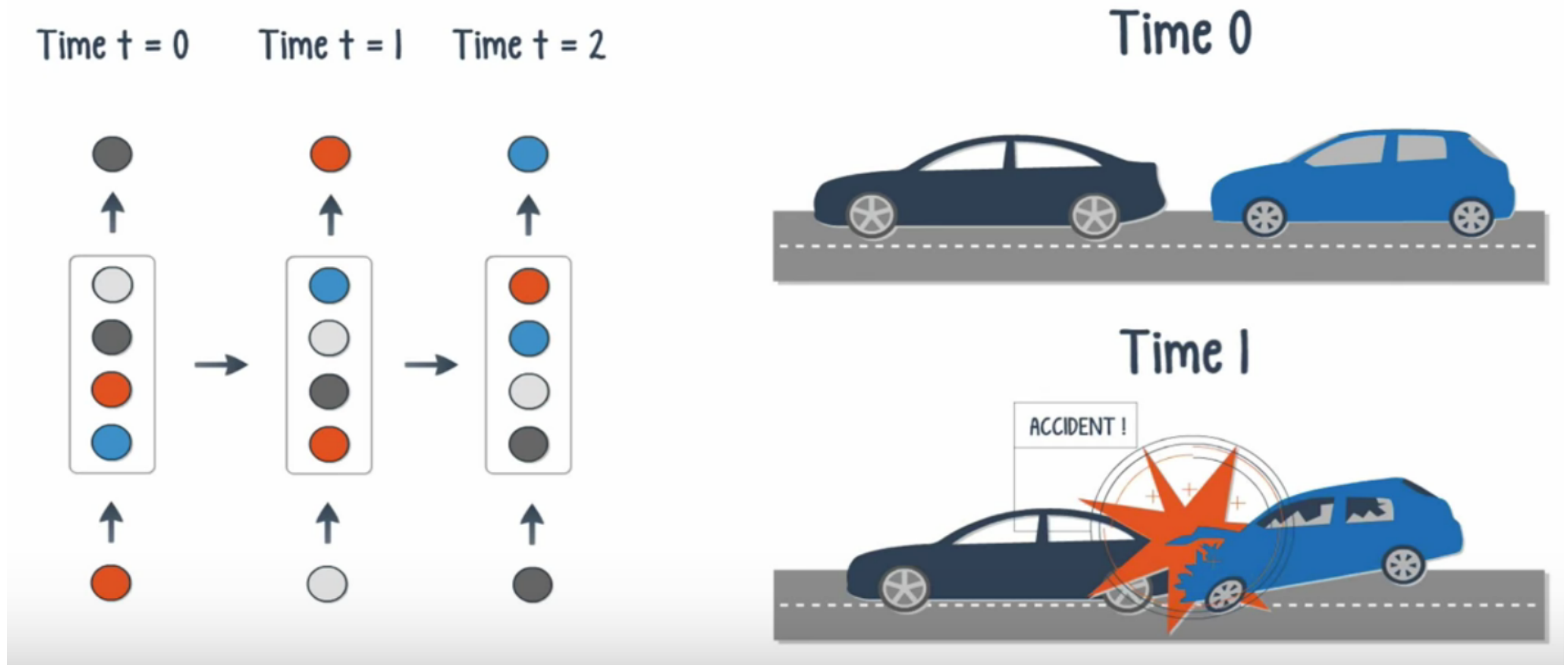
RNN Uses

- ✓ Input: Sequence
- ✓ Output: Singular
- ✓ Application: Document Classification



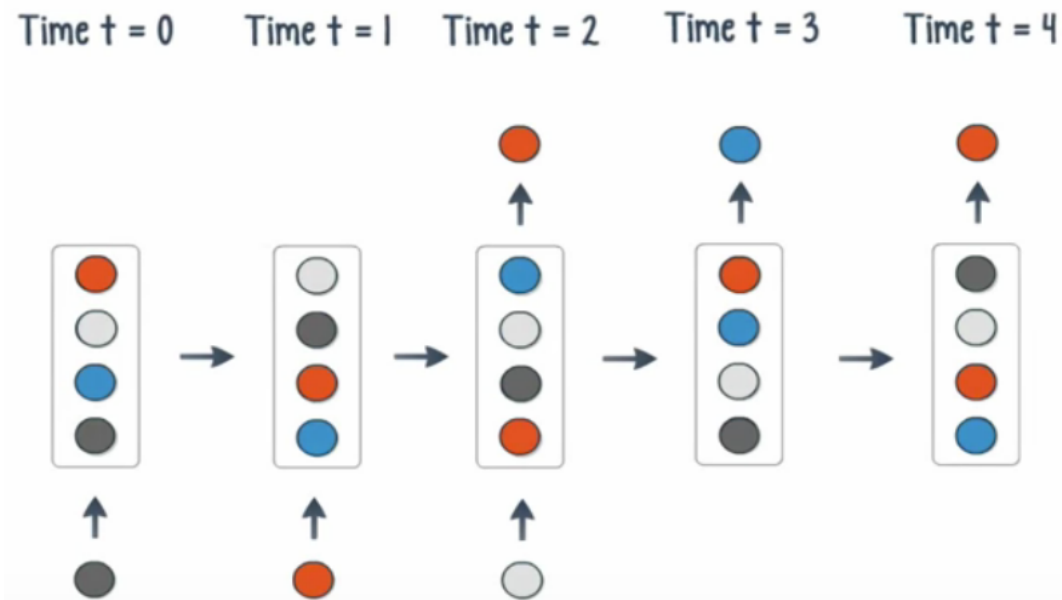
RNN Uses

- ✓ Input: Sequence
- ✓ Output: Sequence
- ✓ Application: Classify video frame by frame

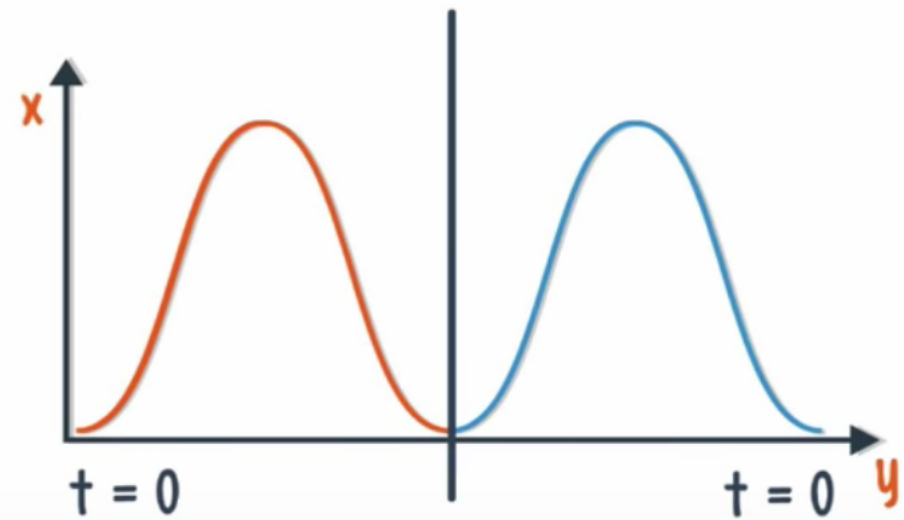


RNN Uses

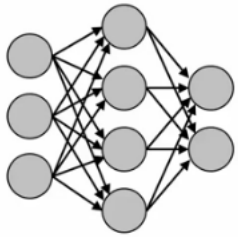
- ✓ Time delay introduced
- ✓ Application: Forecasting demand in supply chain management



Statistical forecasting

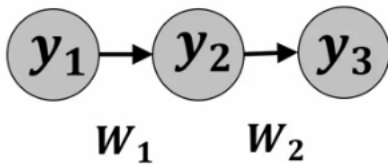


Review: Feed-forward Network



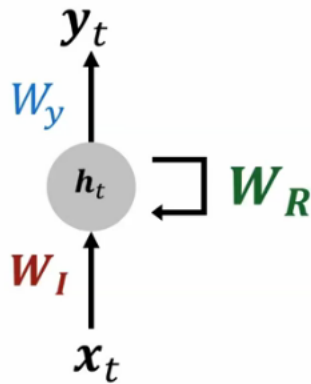
$$y_i = g \left(\sum_j W_{ij} x_j + b_i \right)$$

$$\frac{\partial C}{\partial W} = \frac{\partial C}{\partial g} \cdot \frac{\partial g}{\partial a} \cdot \frac{\partial a}{\partial W}$$



$$\mathbf{y}_k = g(W \mathbf{y}_{k-1} + \mathbf{b})$$

Recurrent Neuron



$$\mathbf{h}^{(t)} = g_h(W_I \mathbf{x}^{(t)} + W_R \mathbf{h}^{(t-1)} + \mathbf{b}_h)$$

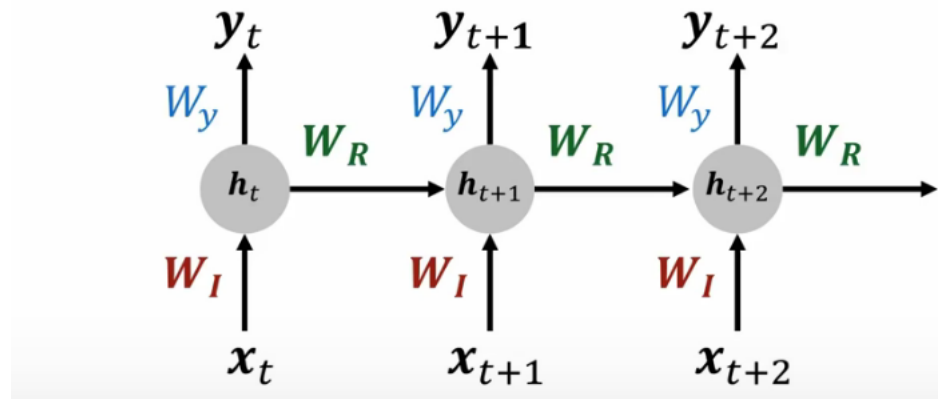
$$\mathbf{y}^{(t)} = g_y(W_y \mathbf{h}^{(t)} + \mathbf{b}_y)$$

How to train a RNN

- ✓ Unrolling a RNN into a feed-forward network

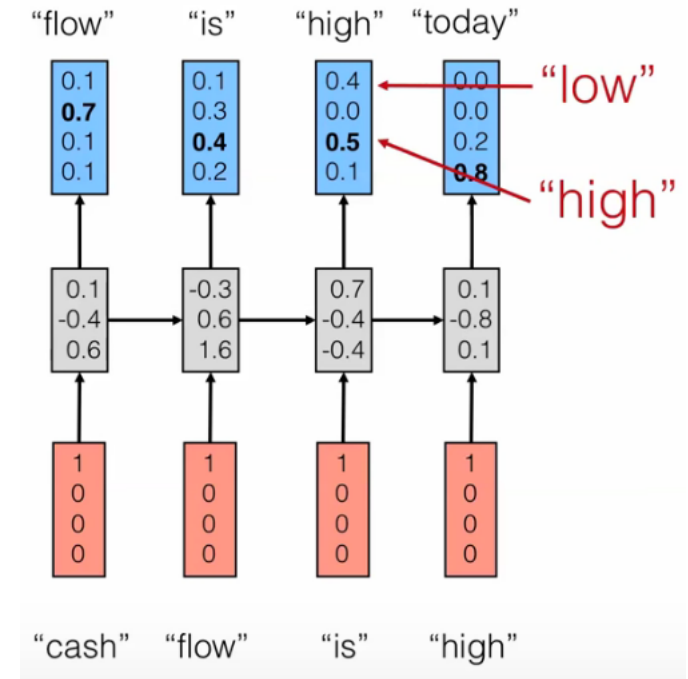
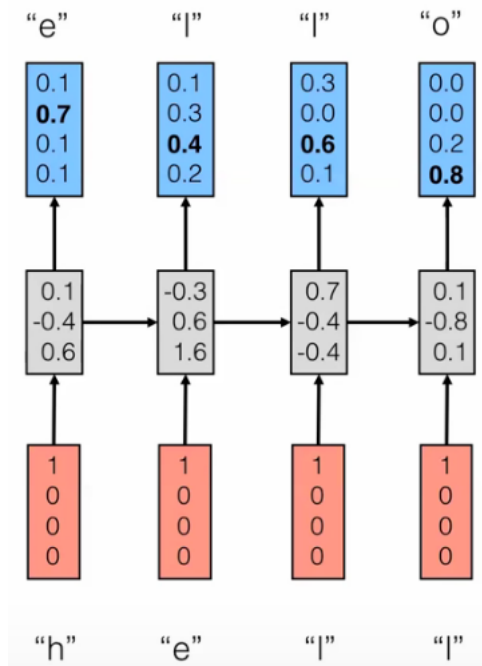
$$\mathbf{h}^{(t)} = g_h(W_I \mathbf{x}^{(t)} + W_R \mathbf{h}^{(t-1)} + \mathbf{b}_h)$$

$$\mathbf{y}^{(t)} = g_y(W_y \mathbf{h}^{(t)} + \mathbf{b}_y)$$



What is RNN good for?

- ✓ Alphabet of 4 letters
- ✓ Input one character then predict the following character

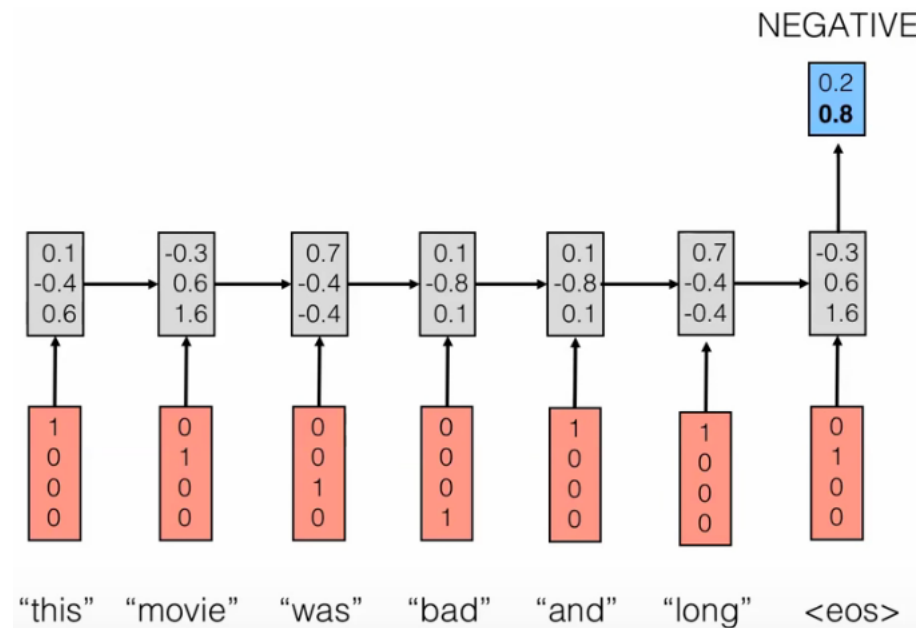


Learned a language model!

$$P(c_t | \{c_{t-1}, c_{t-2}, \dots, c_0\})$$

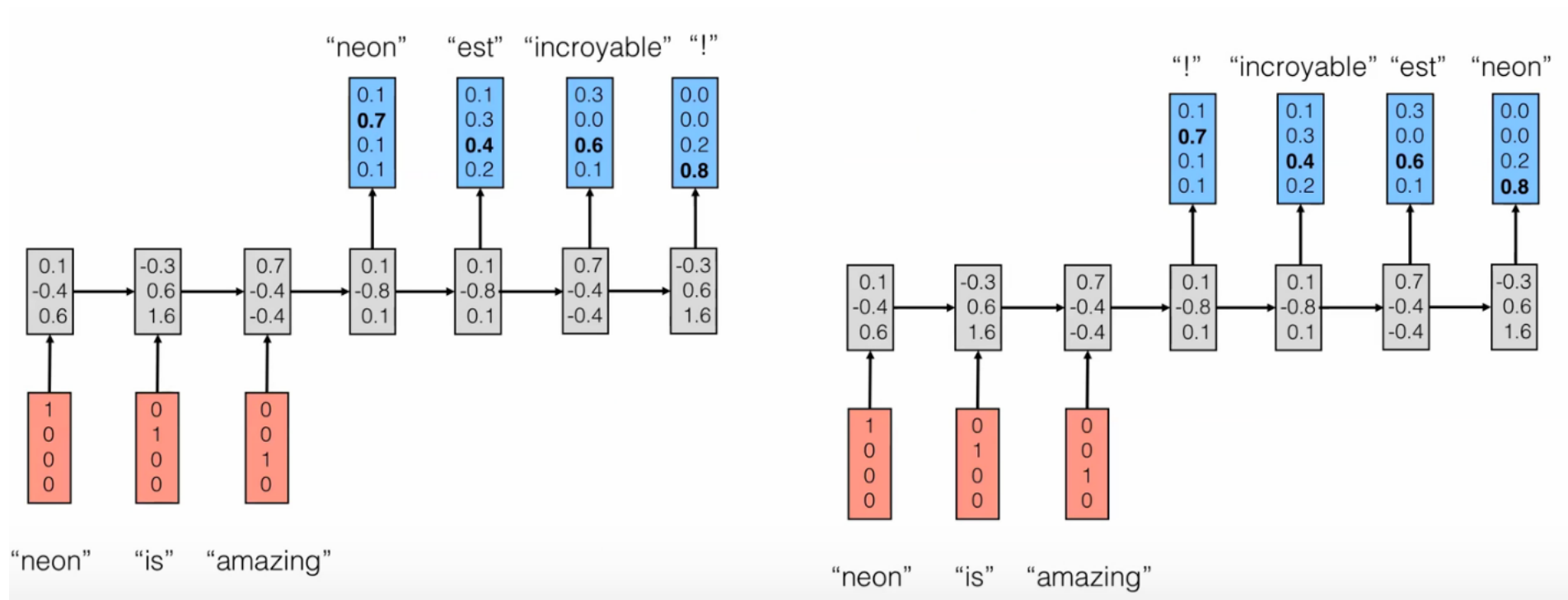
RNN Uses

- ✓ Sentiment analysis
- ✓ Input: a sentence
- ✓ Output: soft max (2 units)



RNN Uses

✓ Translation

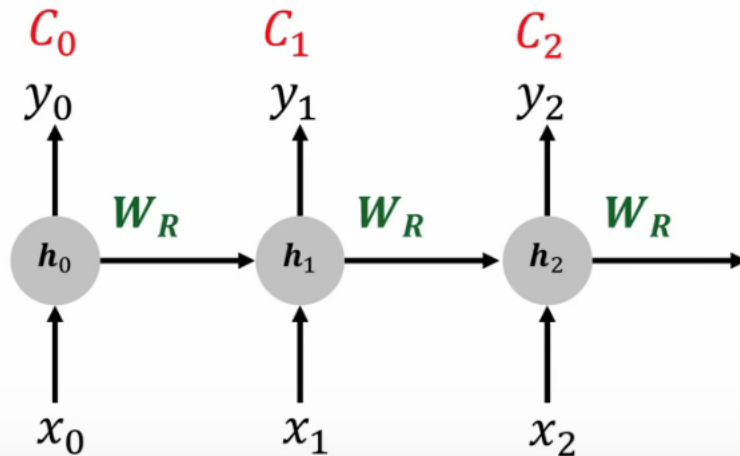


How to train a RNN

- ✓ Unrolling a RNN into a feed-forward network

$$\mathbf{h}^{(t)} = g_h(W_I \mathbf{x}^{(t)} + W_R \mathbf{h}^{(t-1)} + \mathbf{b}_h)$$

$$\mathbf{y}^{(t)} = g_y(W_y \mathbf{h}^{(t)} + \mathbf{b}_y)$$



Combine via:

$$\frac{\partial C}{\partial W_R} = \sum_t \frac{\partial C_t}{\partial W_R}$$

Example gradient:

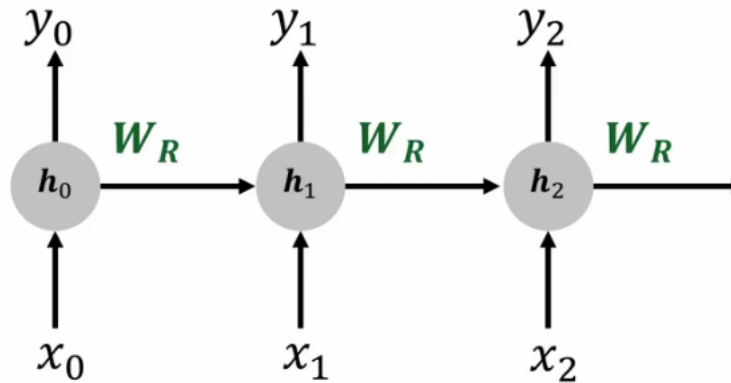
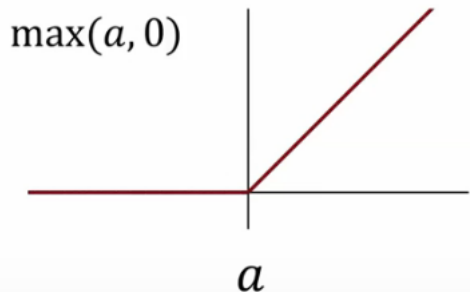
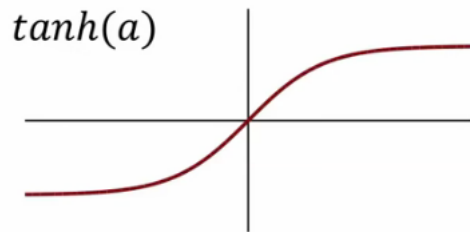
$$\frac{\partial C_2}{\partial W_R} = \frac{\partial C_2}{\partial y_2} \frac{\partial y_2}{\partial h_2} \frac{\partial h_2}{\partial g} \frac{\partial g}{\partial a} \frac{\partial a}{\partial W_R}$$

$$a = (W_I x_2 + W_R h_1 + b_h)$$

Depends on W_R too!

How to train a RNN

- ✓ Vanishing/Exploding gradients



$$\frac{\partial C_{100}}{\partial W_R} = \frac{\partial C_{100}}{\partial y_{100}} \dots W_R \frac{\partial g_{100}}{\partial a_{100}} \dots W_R \frac{\partial g_{99}}{\partial a_{99}} \dots$$

$$\frac{\partial C_T}{\partial W_R} \propto |W_R|^T \left| \frac{\partial g}{\partial a} \right|^T$$

1. Exploding gradients

- Truncated BPTT
- Clip gradients at threshold
- RMSprop to adjust learning rate

2. Vanishing gradients

- Harder to detect
- Weight initialization
- ReLu activation functions
- RMSprop
- LSTM, GRUs