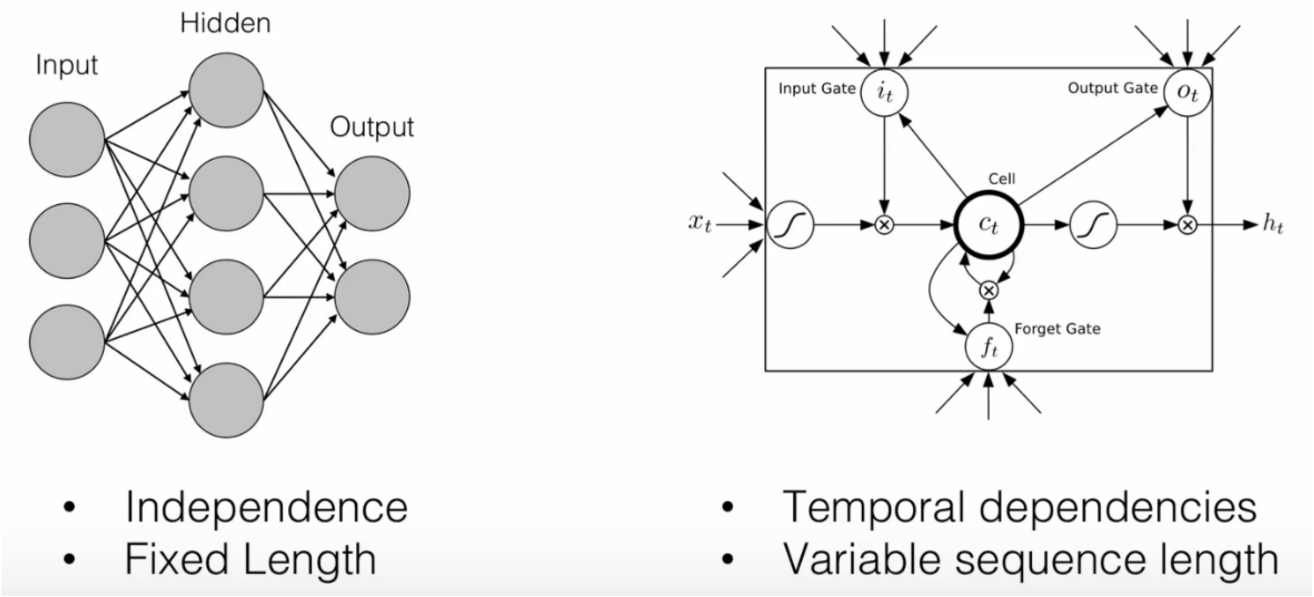
Recurrent Neural Networks

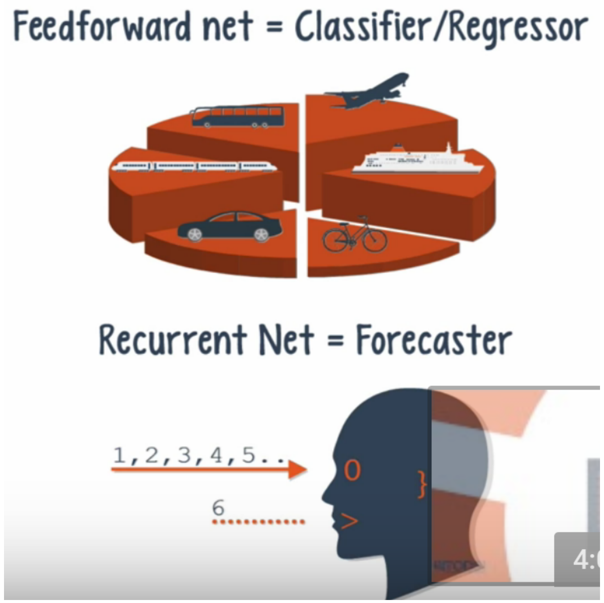
Lecture-6

Source: https://www.youtube.com/watch?v=Ukgii7Yd_cU

Why RNN

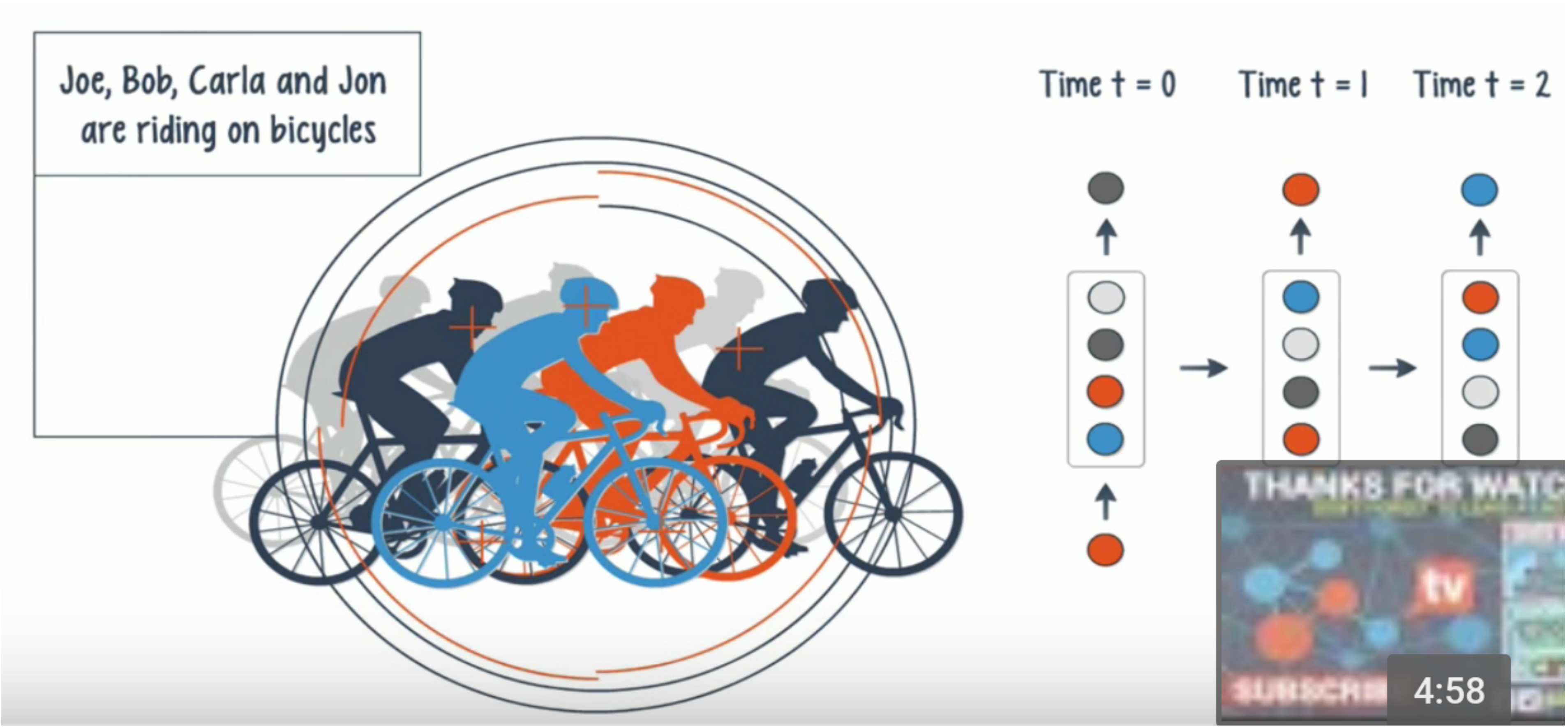
- When to use: Patterns in your data change with time
- Training: GPU (instead of CPU); 1 day vs 8 months





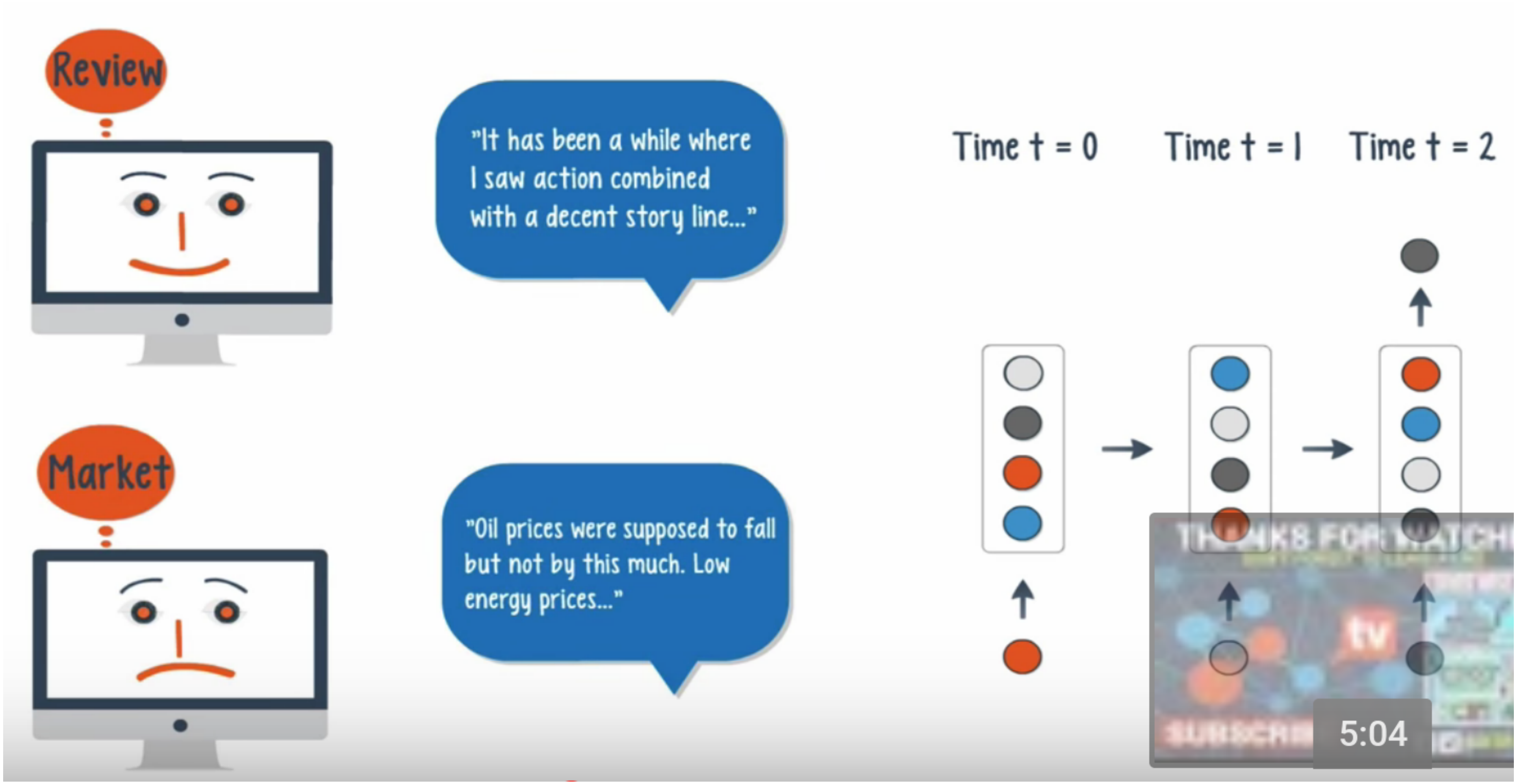
RNN Uses

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



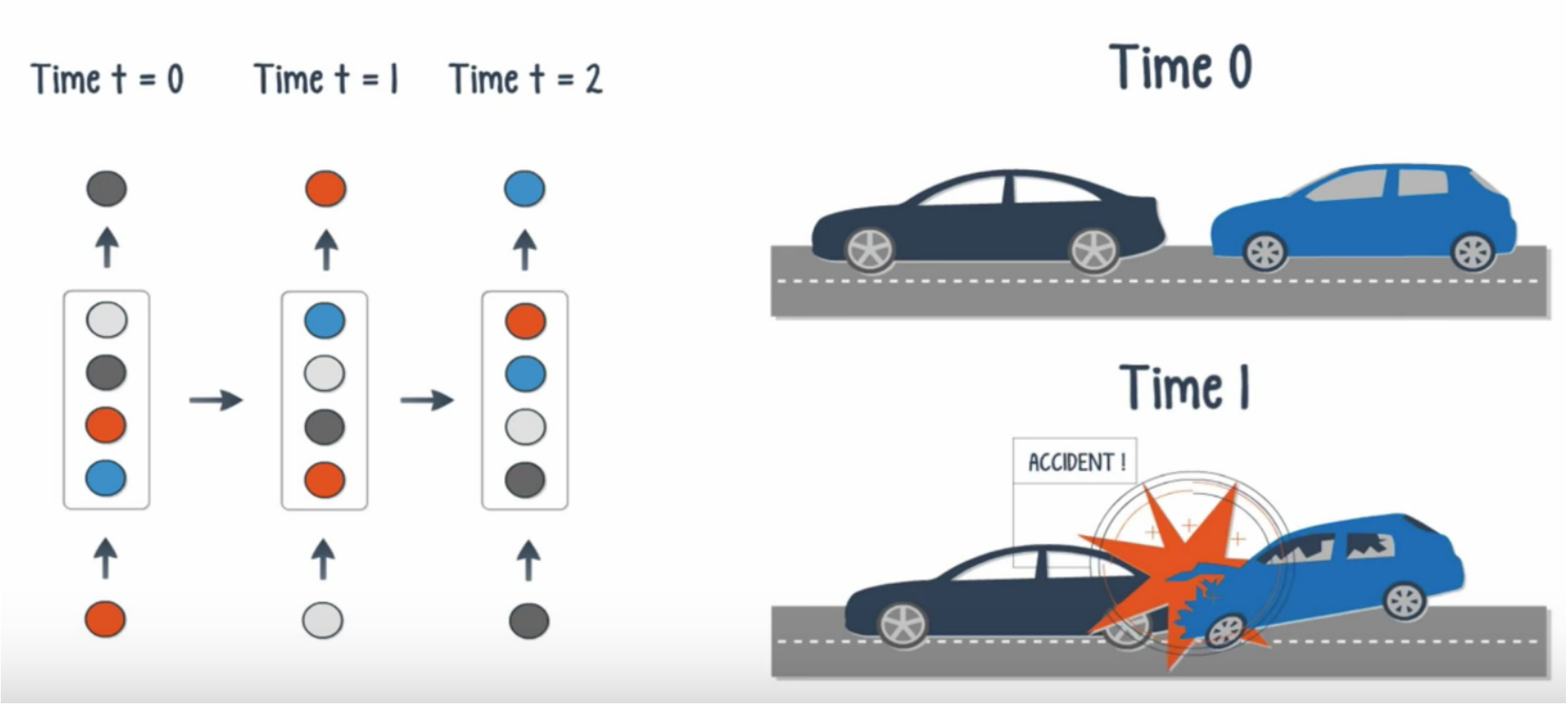
RNN Uses

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



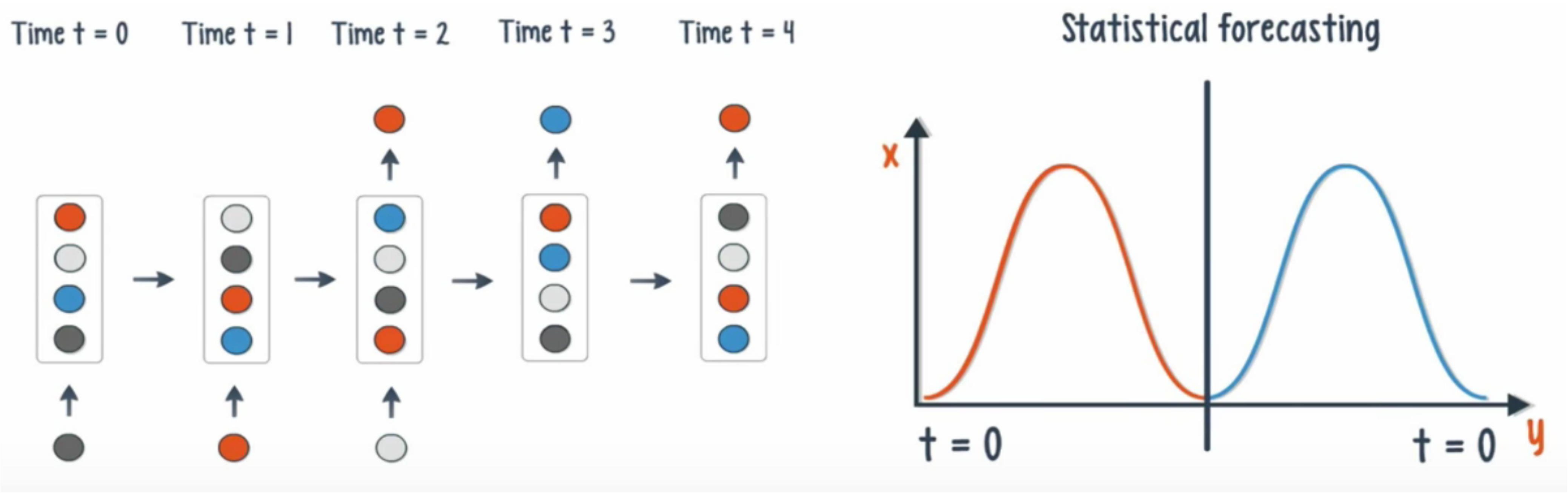
RNN Uses

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

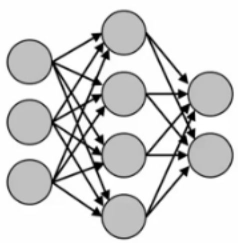


RNN Uses

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

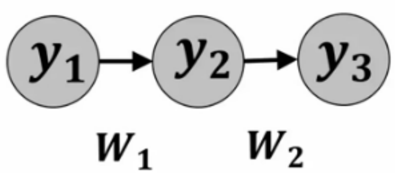


Review: Feed-forward Network



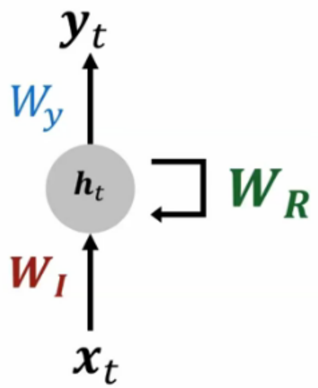
$$y_i = g\left(\sum_j W_{ij} x_j + b_i\right)$$

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



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

Recurrent Neuron



$$\boldsymbol{h}^{(t)} = g_h(W_{\mathrm{I}}\boldsymbol{x}^{(t)} + W_{\mathrm{R}}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)$$

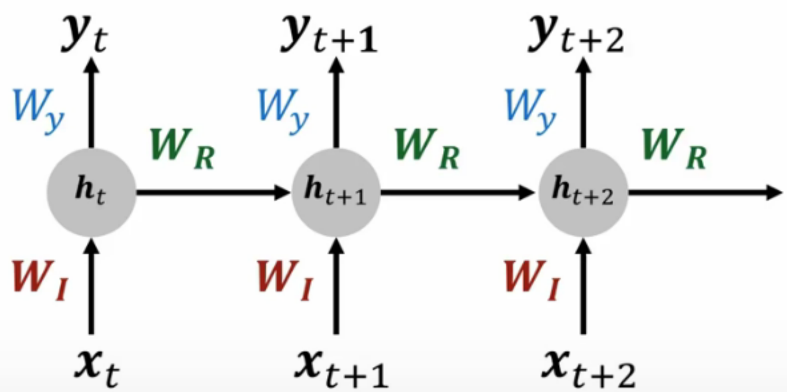
$$\boldsymbol{y}^{(t)} = g_y(W_y\boldsymbol{h}^{(t)} + \boldsymbol{b}_y)$$

How to train a RNN

- Unrolling a RNN into a feed-forward network

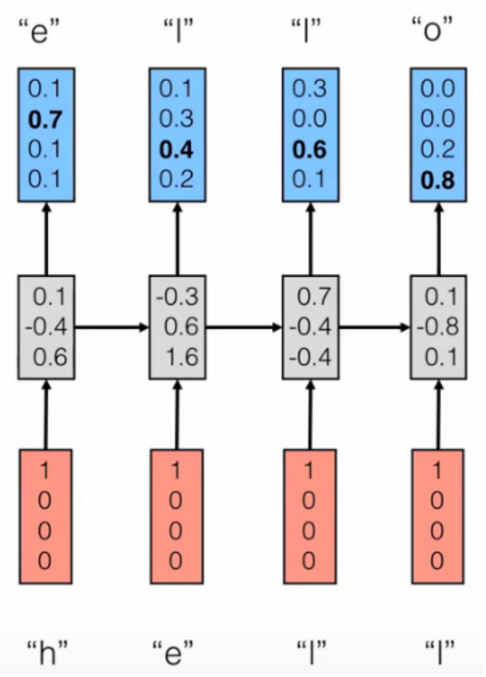
$$\boldsymbol{h}^{(t)} = g_h(W_\mathrm{I}\boldsymbol{x}^{(t)} + W_\mathrm{R}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)$$

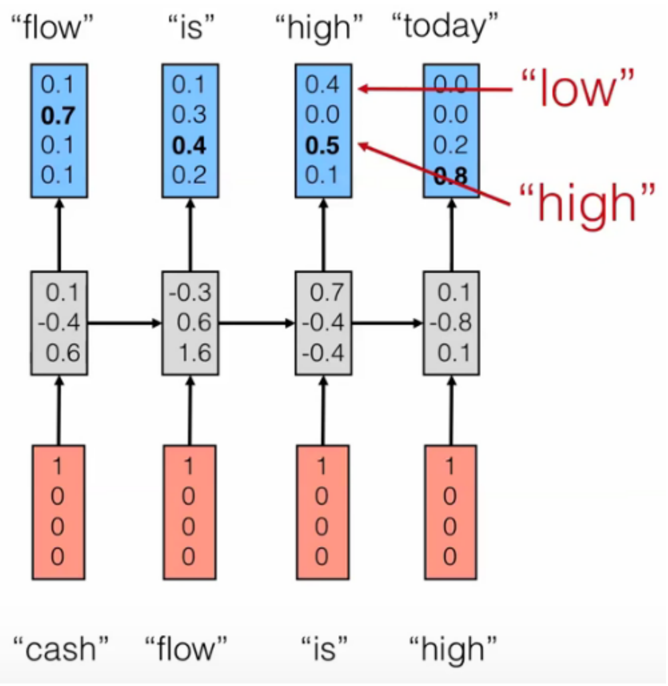
$$\boldsymbol{y}^{(t)} = g_y(W_y\boldsymbol{h}^{(t)} + \boldsymbol{b}_y)$$



What is RNN good for?

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



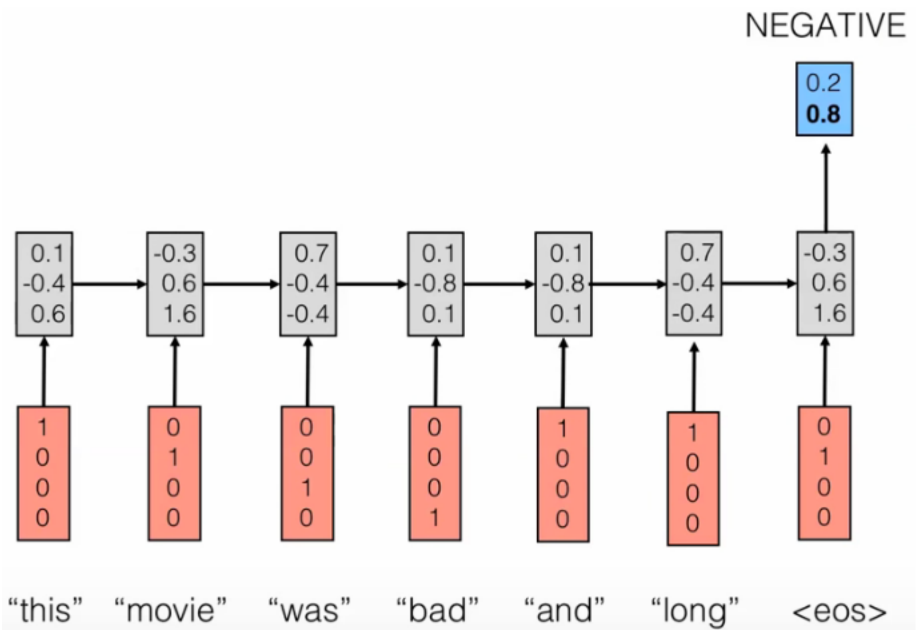


Learned a language model!

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

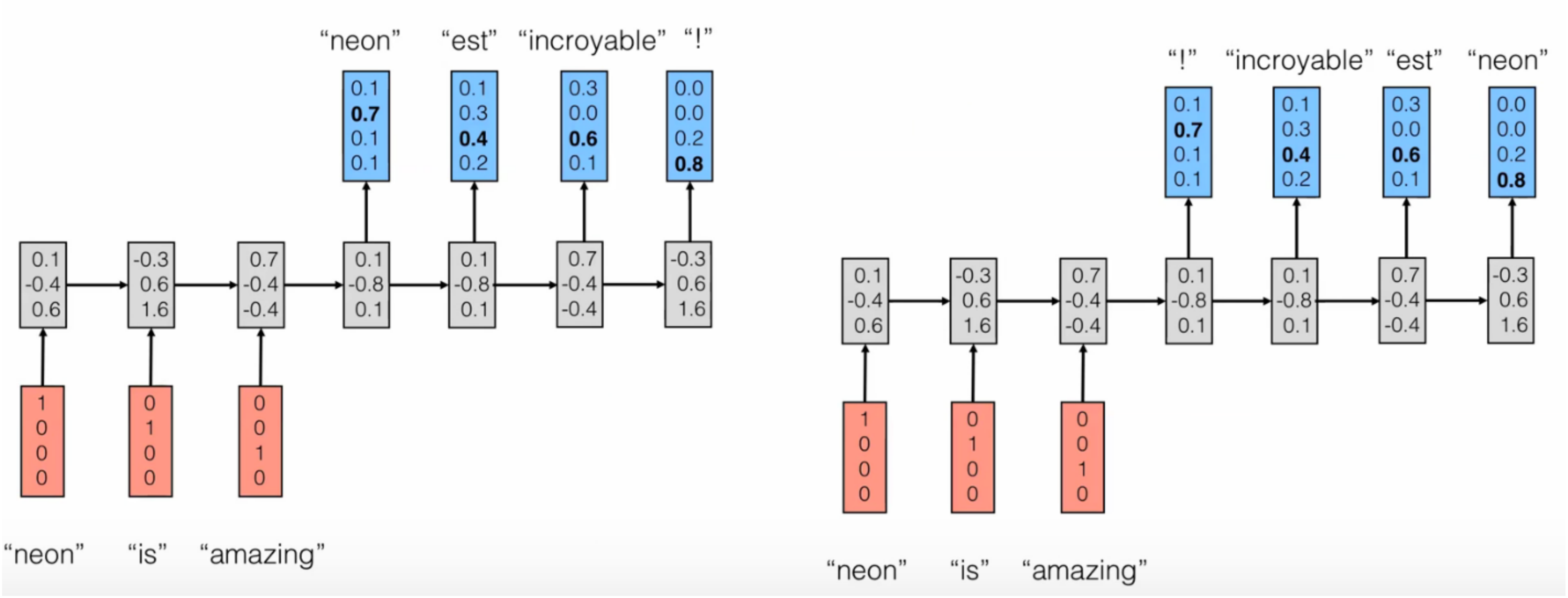
RNN Uses

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



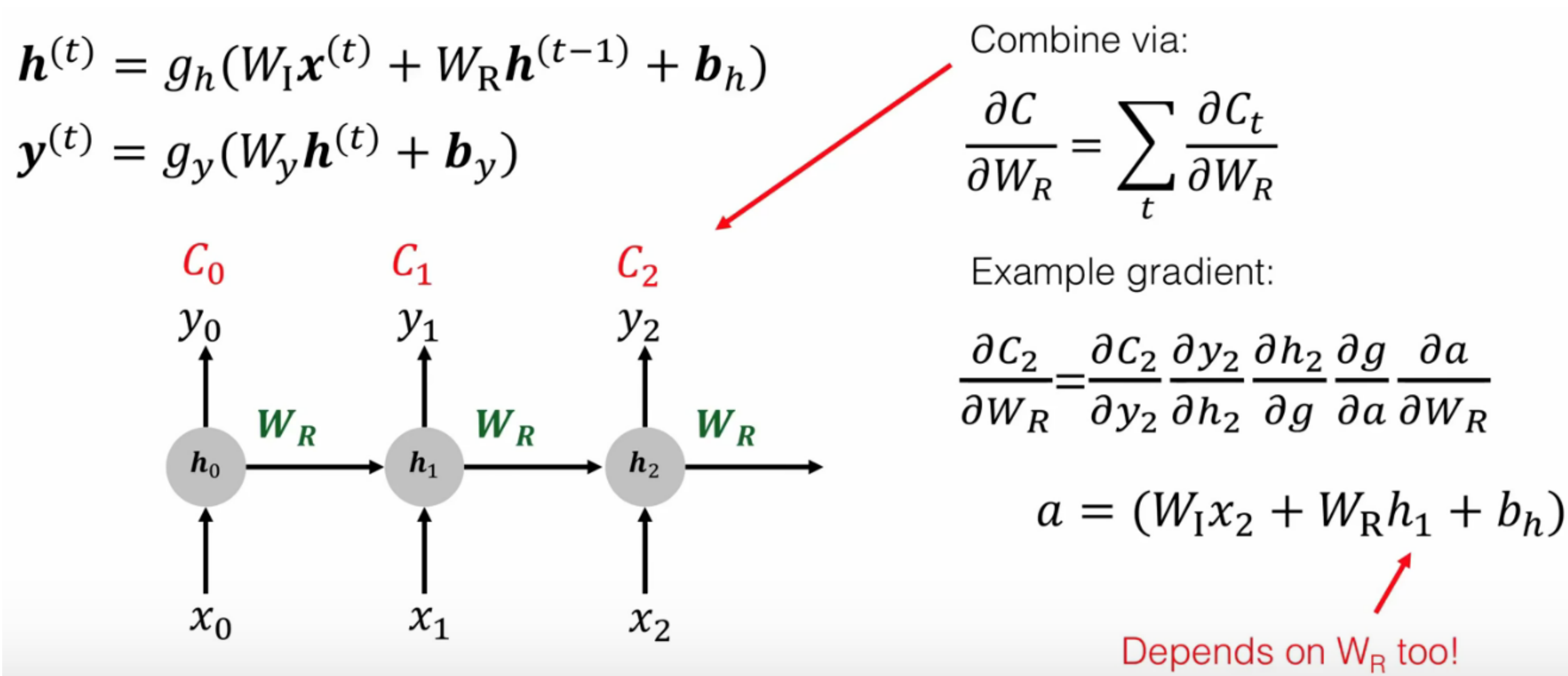
RNN Uses

- Translation



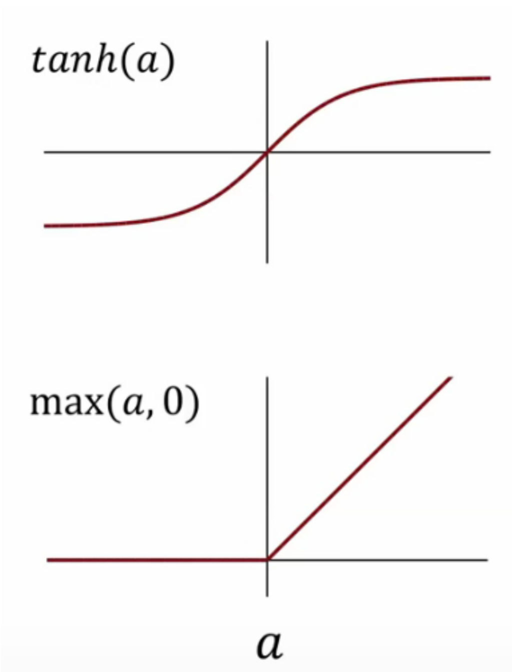
How to train a RNN

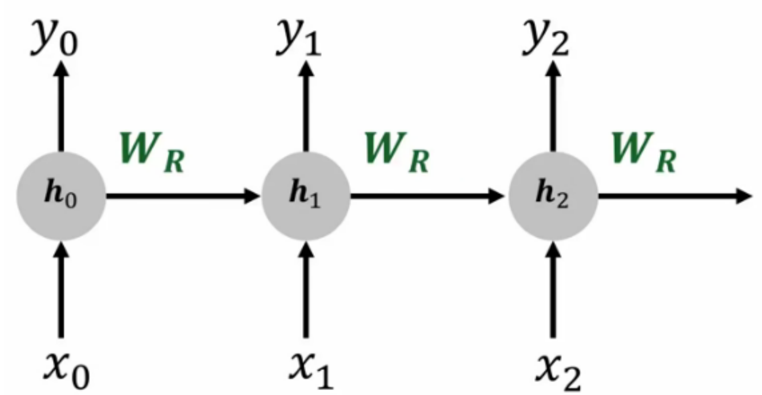
- Unrolling a RNN into a feed-forward network



How to train a RNN

- Vanishing/Exploding gradients





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

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

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
    - $\boxed{\text{LSTM, GRUs}}$