NATURAL LANGUAGE PROCESSING

WEEK-2

THIRUMURUGAN.R

NEURAL LANGUAGE MODELS:

Curse of dimensionality

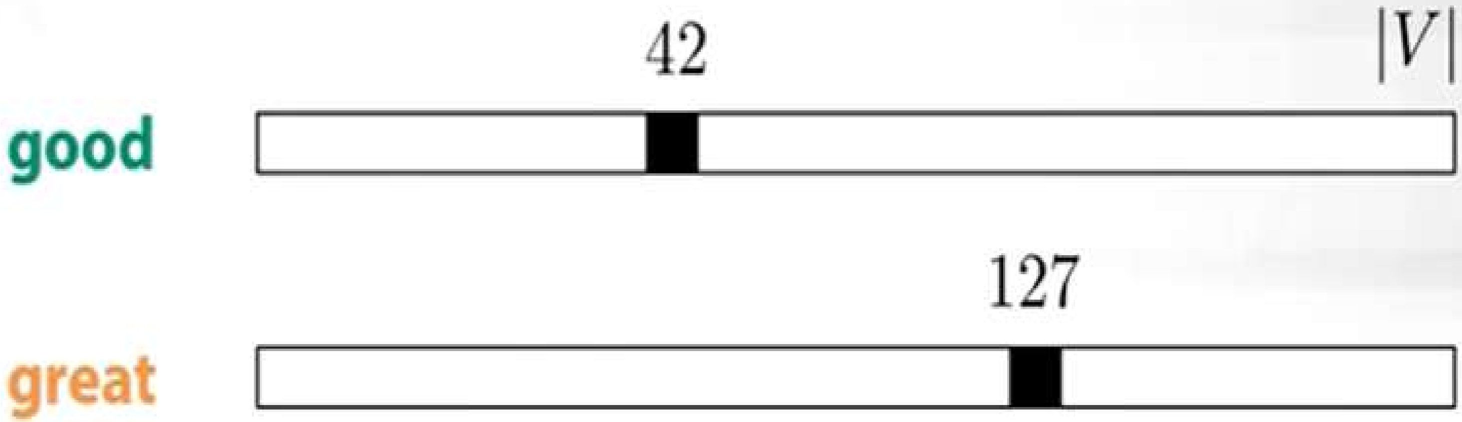
Imagine you have seen the following many times:

- **Have a good day**.

However, you have not seen the following:

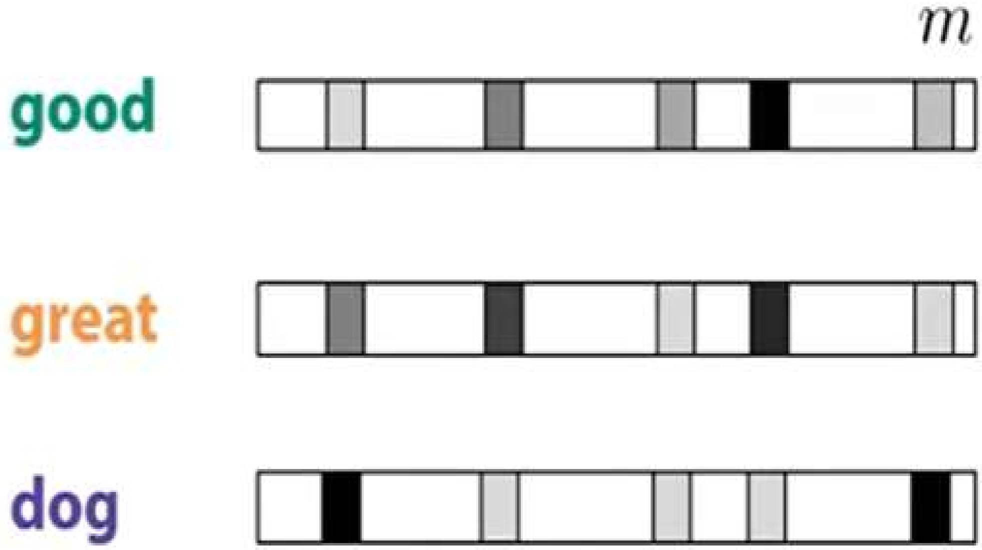
- **Have a great day**.

What happens than (even with smoothing)?



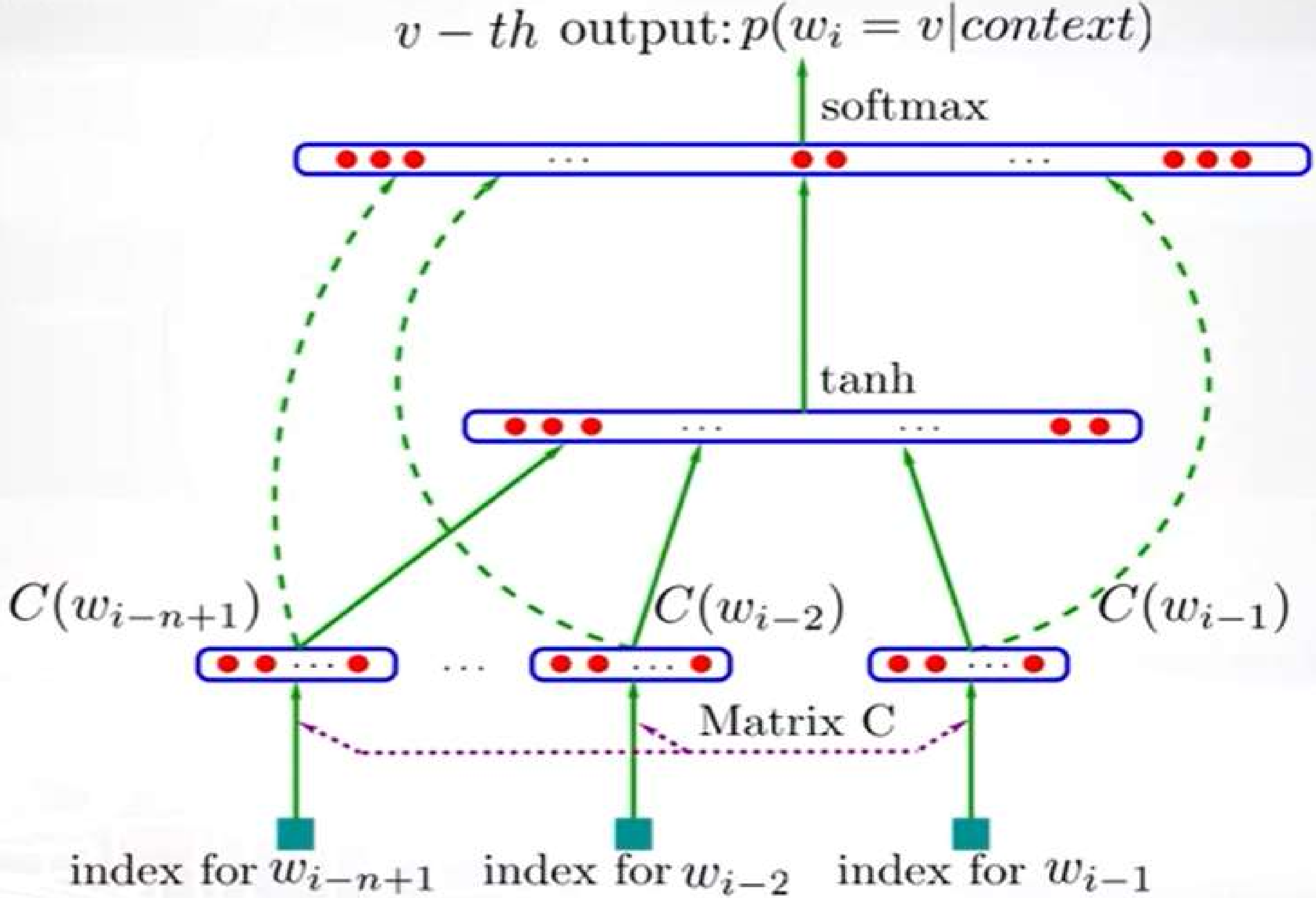
How to generalize better

- Learn distributed representations for words
- Express probabilities of sequences in terms of these distributed representations and learn parameters



$C^{|V| \times m}$ – matrix of distributed word representations.

Probabilistic Neural Language Model



Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, A Neural Probabilistic Language Model, JMLR, 2003

Probabilistic Neural Language Model

$$p(w_i|w_{i-n+1},\dots w_{i-1}) = \frac{\exp\left(y_{w_i}\right)}{\sum\limits_{w\in V}\exp\left(y_w\right)}$$

Softmax* over components of *y

$$y = b + Wx + U\tanh(d + Hx)$$

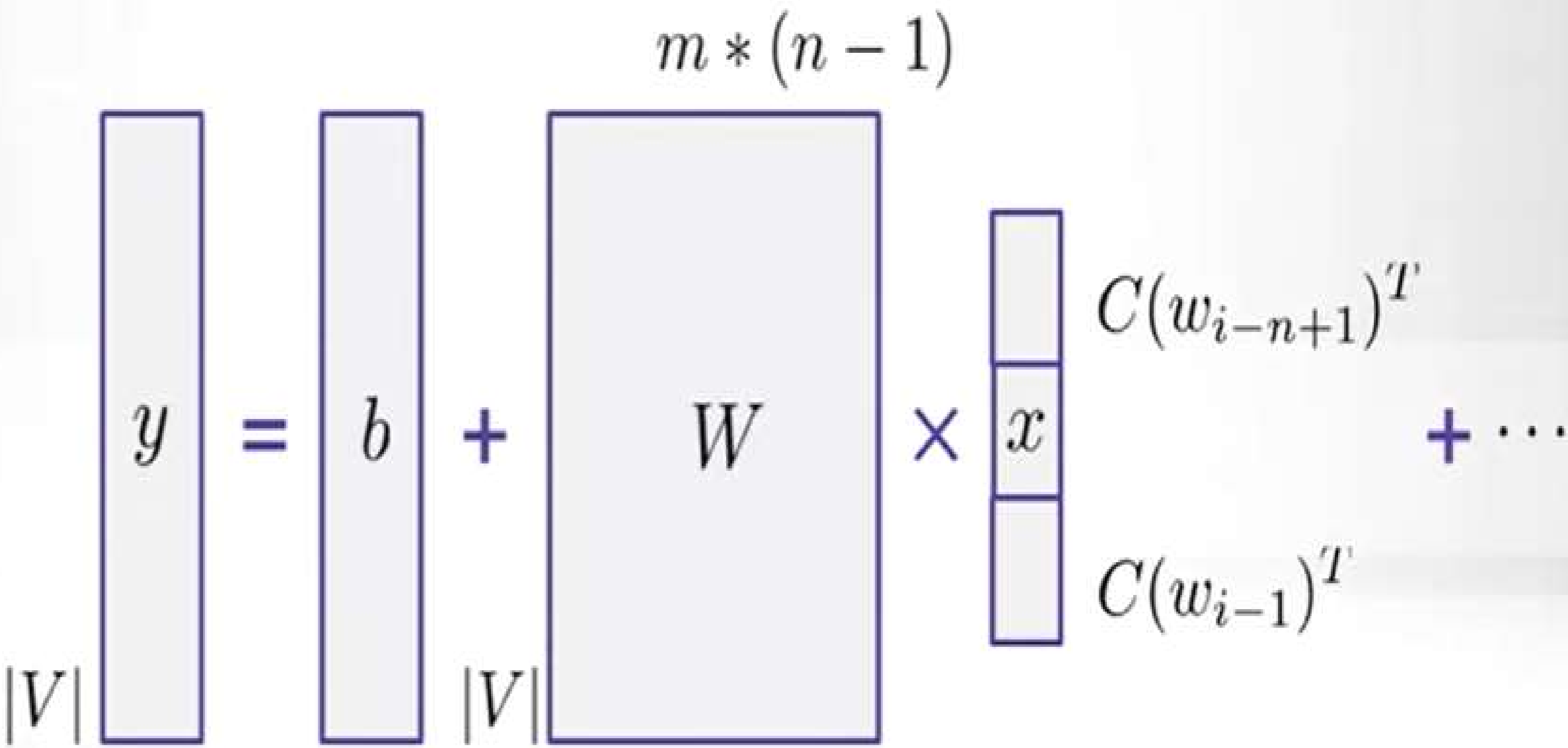
***Feed-forward NN* with tons of parameters**

$$x = [C(w_{i-n+1}),\dots C(w_{i-1})]^T$$

***Distributed representation* of context words**

It's over-complicated…

$$y = b + Wx + U\tanh(d + Hx)$$



Log-Bilinear Language Model

- Has much less parameters and non-linear activations
- Measures similarity between the word and the context:

$$p(w_i | w_{i-n+1}, \dots w_{i-1}) = \frac{\exp(\hat{r}^T r_{w_i} + b_{w_i})}{\sum_{w \in V} \exp(\hat{r}^T r_w + b_w)}$$

Representation of word:

$$r_{w_i} = C(w_i)^T$$

Representation of context:

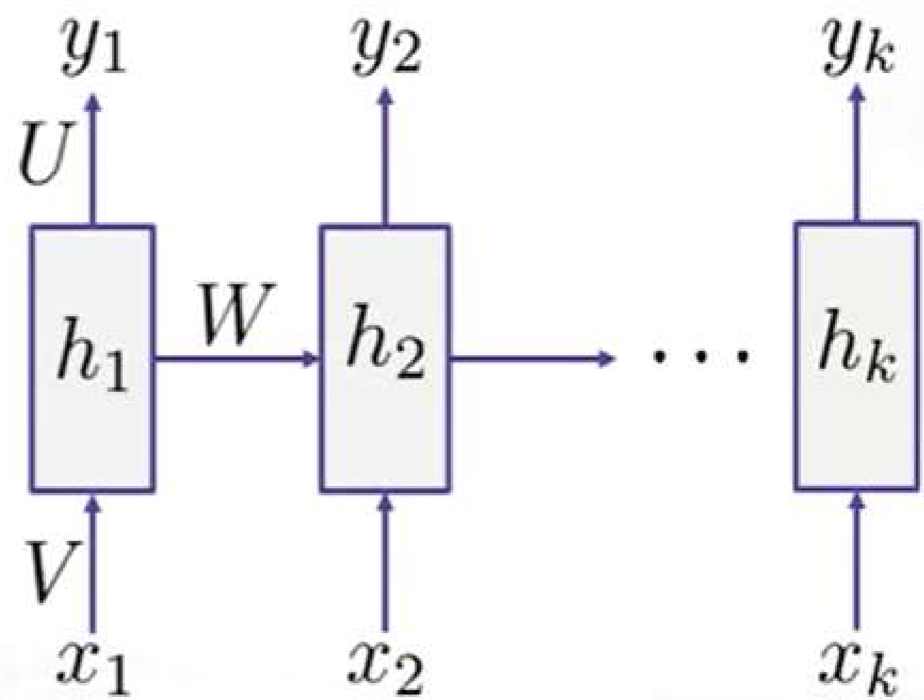
$$\hat{r} = \sum_{k=1}^{n-1} W_k C(w_{i-k})^T$$

Recurrent Neural Networks

- Extremely popular architecture for any sequential data:

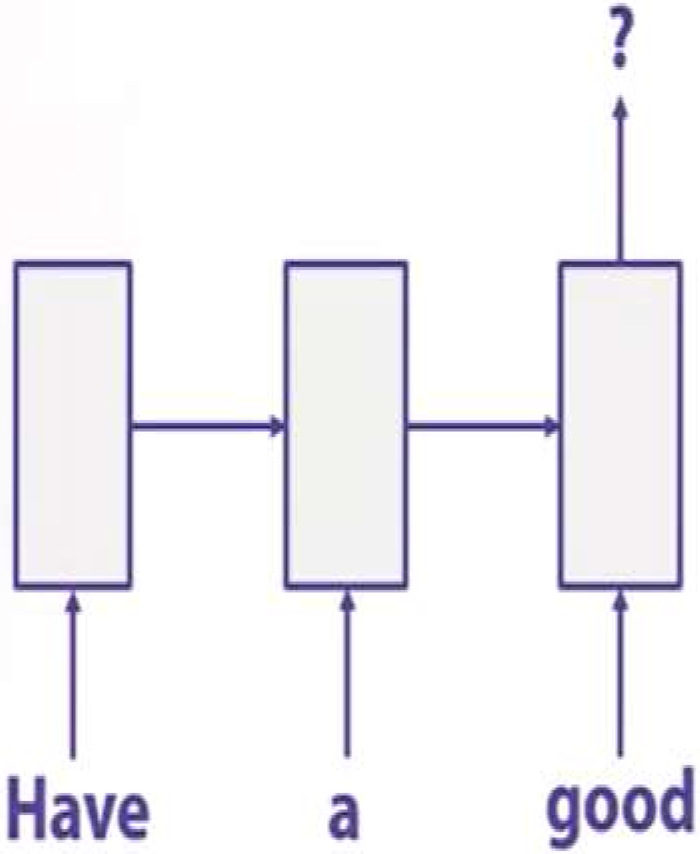
$$h_i = f(Wh_{i-1} + Vx_i + b)$$

$$y_i = Uh_i + \tilde{b}$$



RNN Language Model

- Predicts a next word based on a previous context



Architecture:

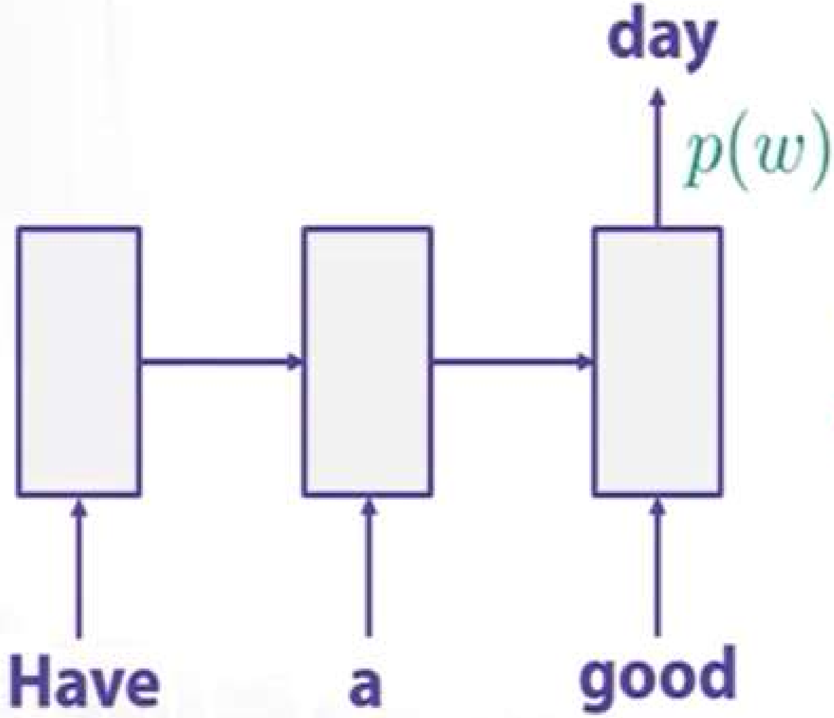
- Use the current state output
- Apply a linear layer on top
- Do *softmax* to get probabilities

How do we train it?

Cross-entropy loss (for one position):

$$-\log p(w_i) = -\sum_{w \in V} [w = w_i] \log p(w)$$

Only one non-zero

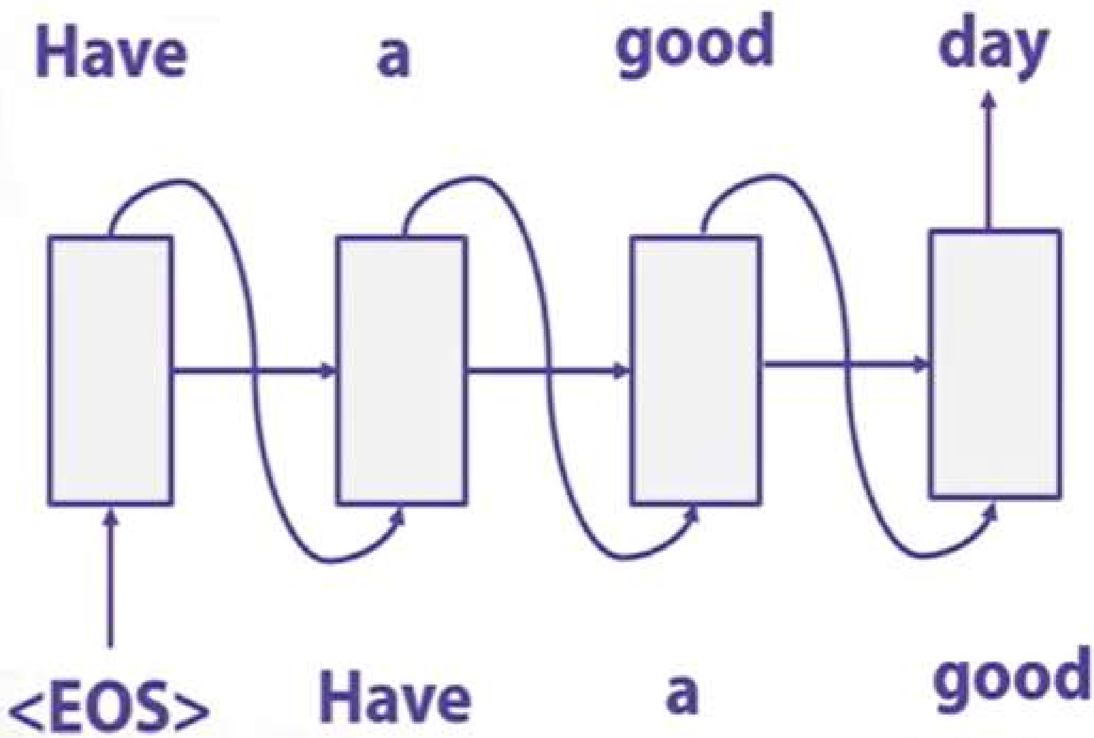


- **Target:** word w_i
- **Output:** probabilities $p(w)$

How do we use it to generate language?

Idea:

- Feed the previous output as the next input
- Take *argmax* at each step (greedily) or use *beam search*



RNN Language Model

- RNN-LM has lower *perplexity* and *word error rate* than 5-gram model with Knesser-Ney smoothing.
- The experiment is held on Wall Street Journal corpus:

Model	# words	PPL	WER
KN5 LM	200K	336	16.4
KN5 LM + RNN 90/2	200K	271	15.4
KN5 LM	1M	287	15.1
KN5 LM + RNN 90/2	1M	225	14.0
KN5 LM	6.4M	221	13.5
KN5 LM + RNN 250/5	6.4M	156	11.7

- Later experiments: char-level RNNs can be very effective!

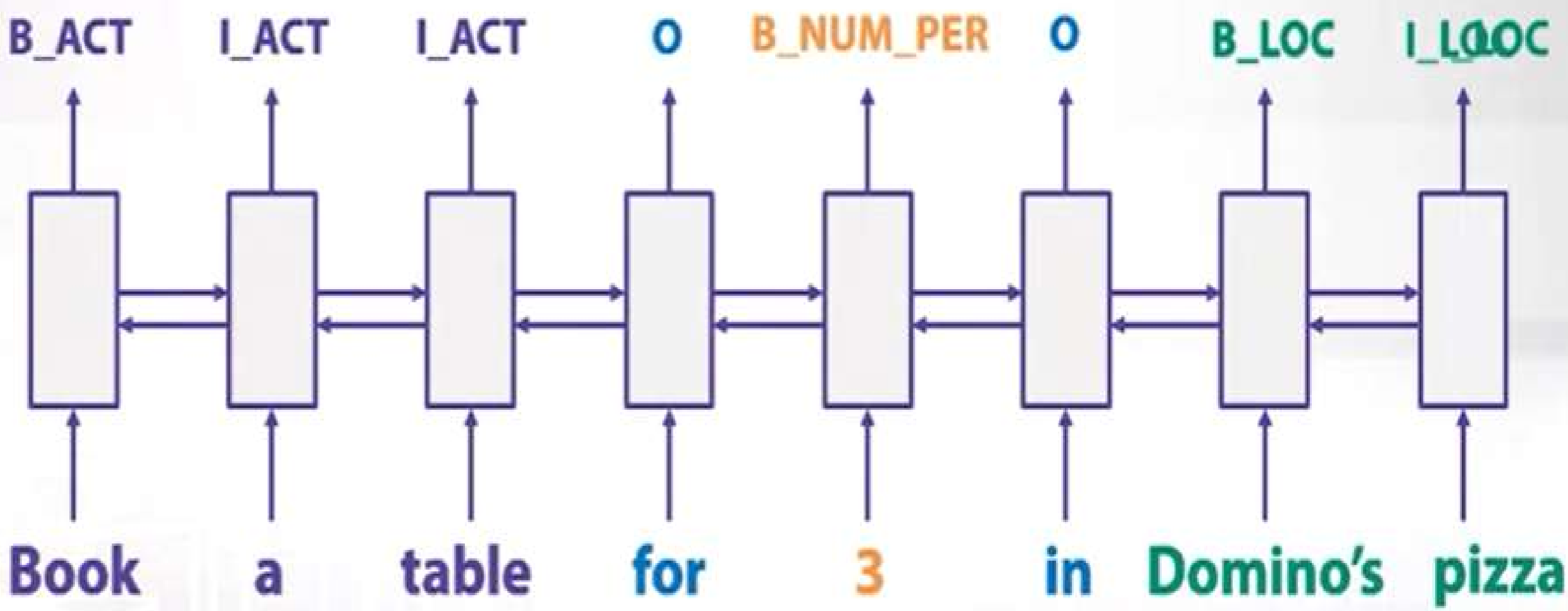
Mikolov, Karafiát, Burget, Cernocký, and Khudanpur. Recurrent neural network based language model. INTERSPEECH 2010.

Character-level RNN: Shakespeare example

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

Bi-directional LSTM

- Universal approach for sequence tagging
- You can stack several layers + add linear layers on top
- Trained by cross-entropy loss coming from each position



What are Recurrent Neural Networks (RNN) and Long Short Term Memory Networks (LSTM) ?

Applications

- Image Captioning
- Generating poems after being trained on Shakespeare poem’s
- Reading Handwriting from left to right
- Generating music

Problem with Feed Forward Neural Networks

- Not Designed for sequences / time series data, hence the results with time series / sequential data are bad.

- Does not model memory.

- Example of Sequential data :

  Sentences, Stock Prices, Video Stream etc.

How does RNN work ?

- Recursive Formula

$$S_t = F_w(S_{t-1}, X_t)$$

X_t - Input at time step t

S_t - State at time step t

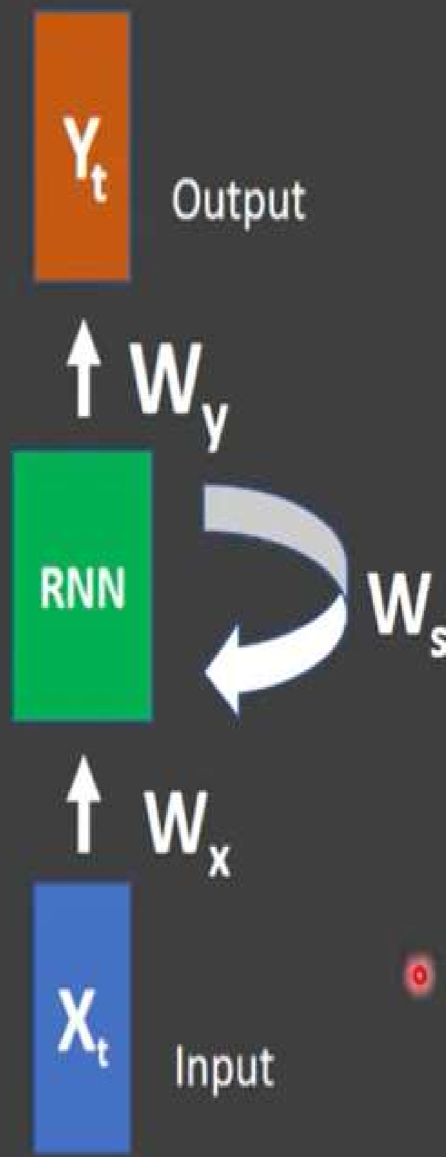
F_w - Recursive function

Simple RNN

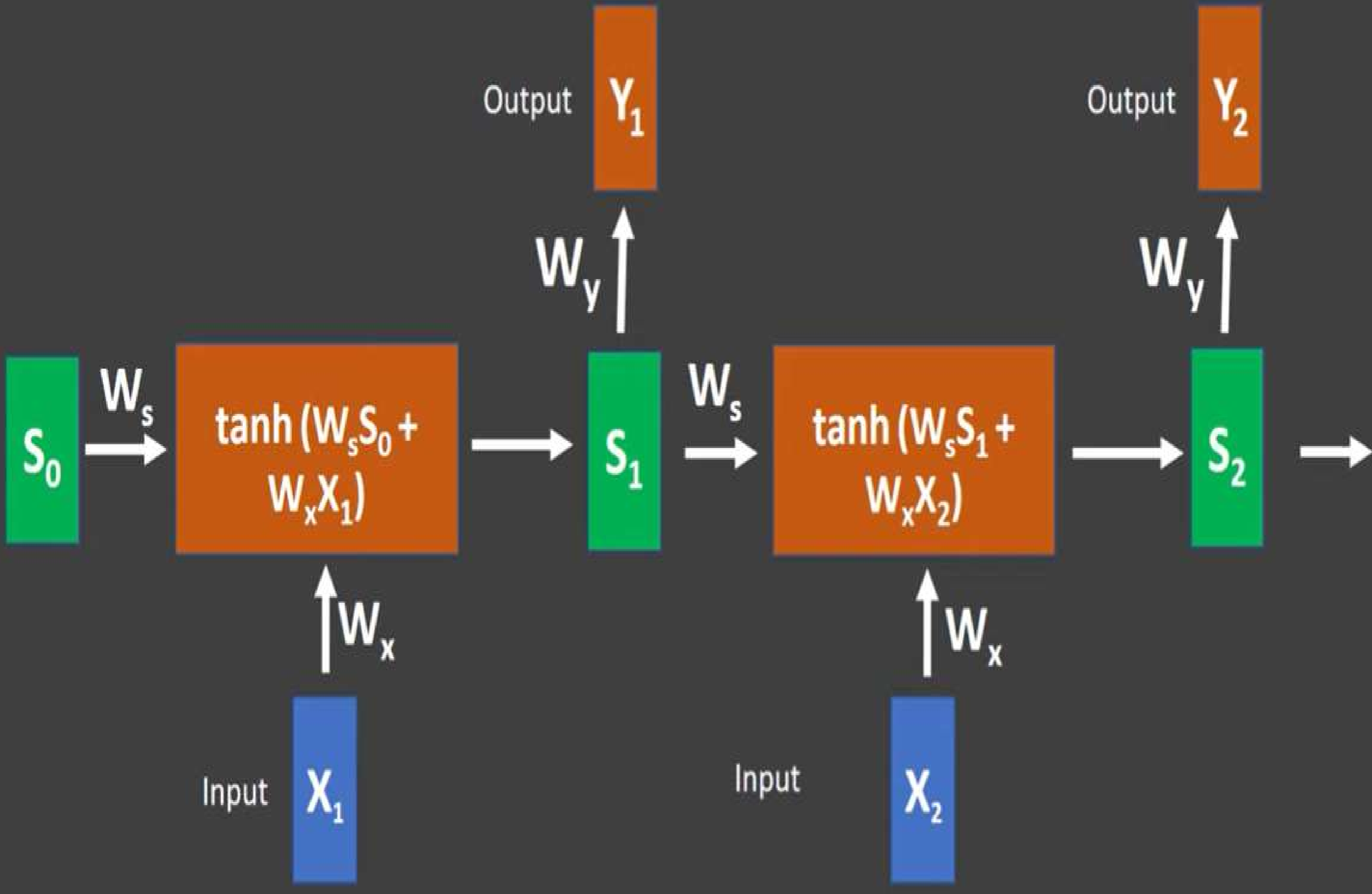
$$S_t = F_w(S_{t-1}, X_t)$$

$$S_t = \tanh(W_s S_{t-1} + W_x X_t)$$

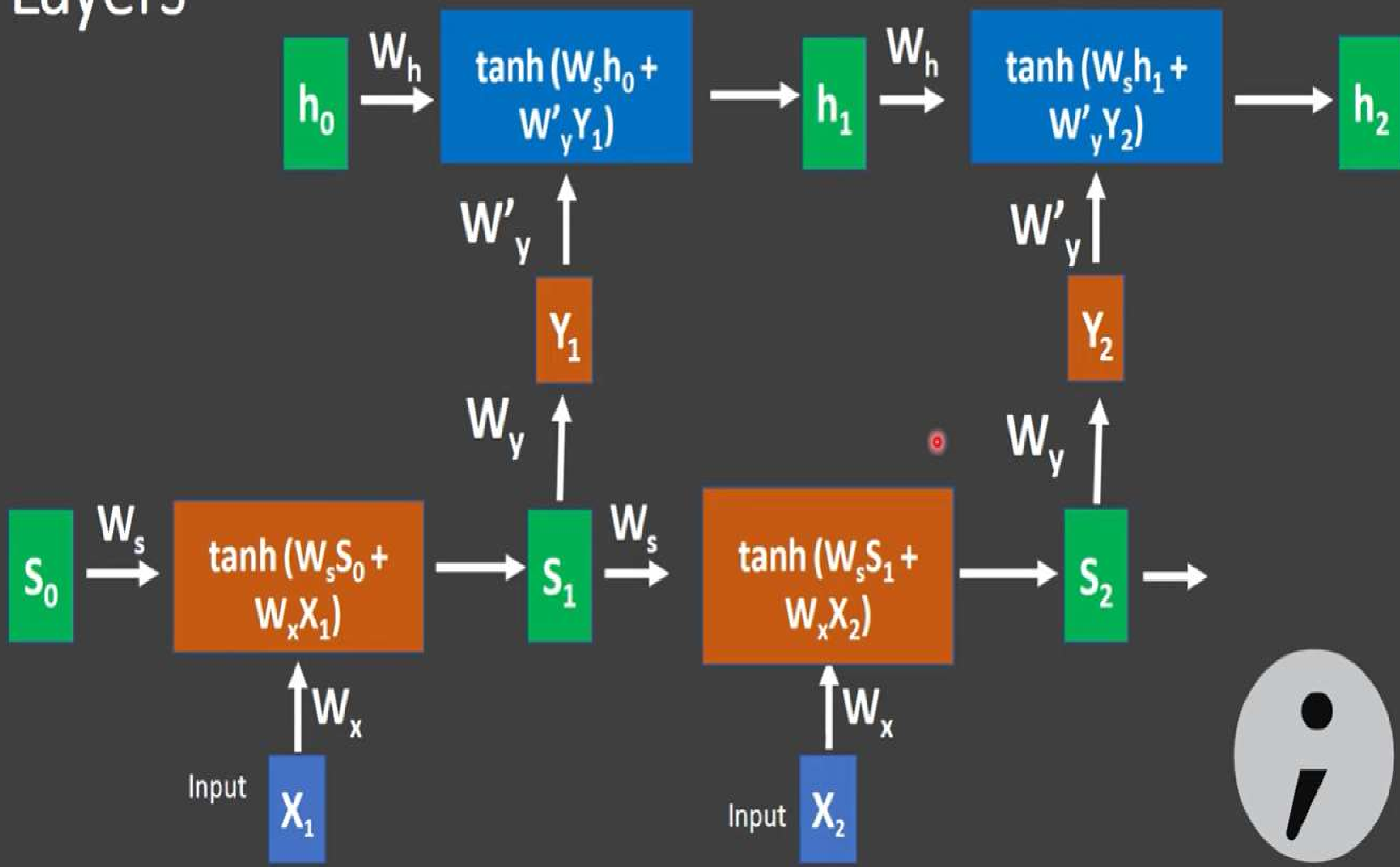
$$Y_t = W_y S_t$$



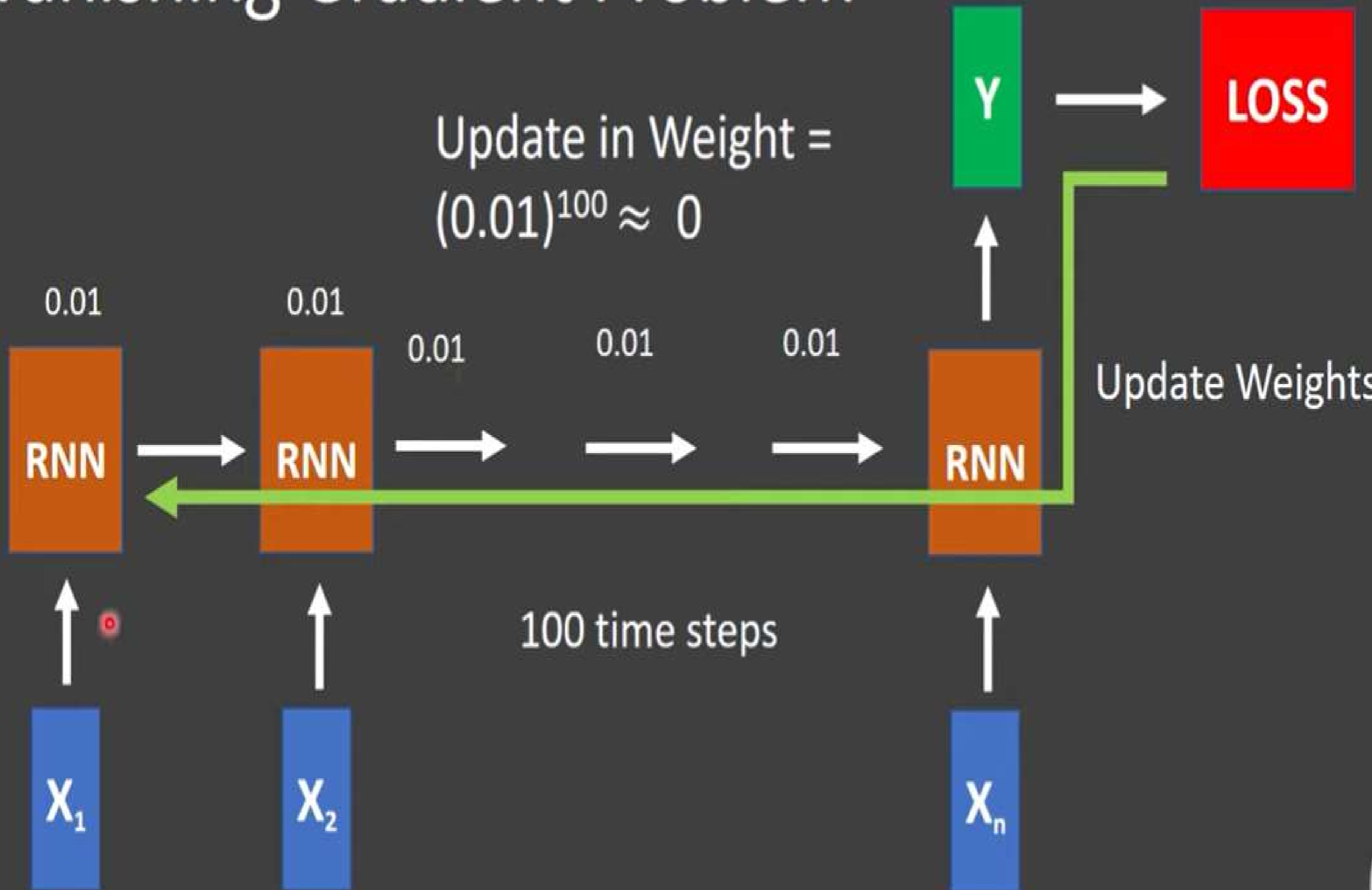
Simple RNN (Unrolled)



Multiple Hidden Layers



Vanishing Gradient Problem



Solution - LSTM

$f_t = \sigma(W_f S_{t-1} + W_f X_t)$ - Forget Gate

$i_t = \sigma(W_i S_{t-1} + W_i X_t)$ - Input Gate

$o_t = \sigma(W_o S_{t-1} + W_o X_t)$ - Output Gate

$\tilde{C}_t = tanh(W_c S_{t-1} + W_c X_t)$

$c_t = (I_t * \tilde{C}_t) + (f_t * c_{t-1})$ - Cell State

LSTM

$f_t = \sigma(W_f S_{t-1} + W_f X_t)$ - Forget Gate

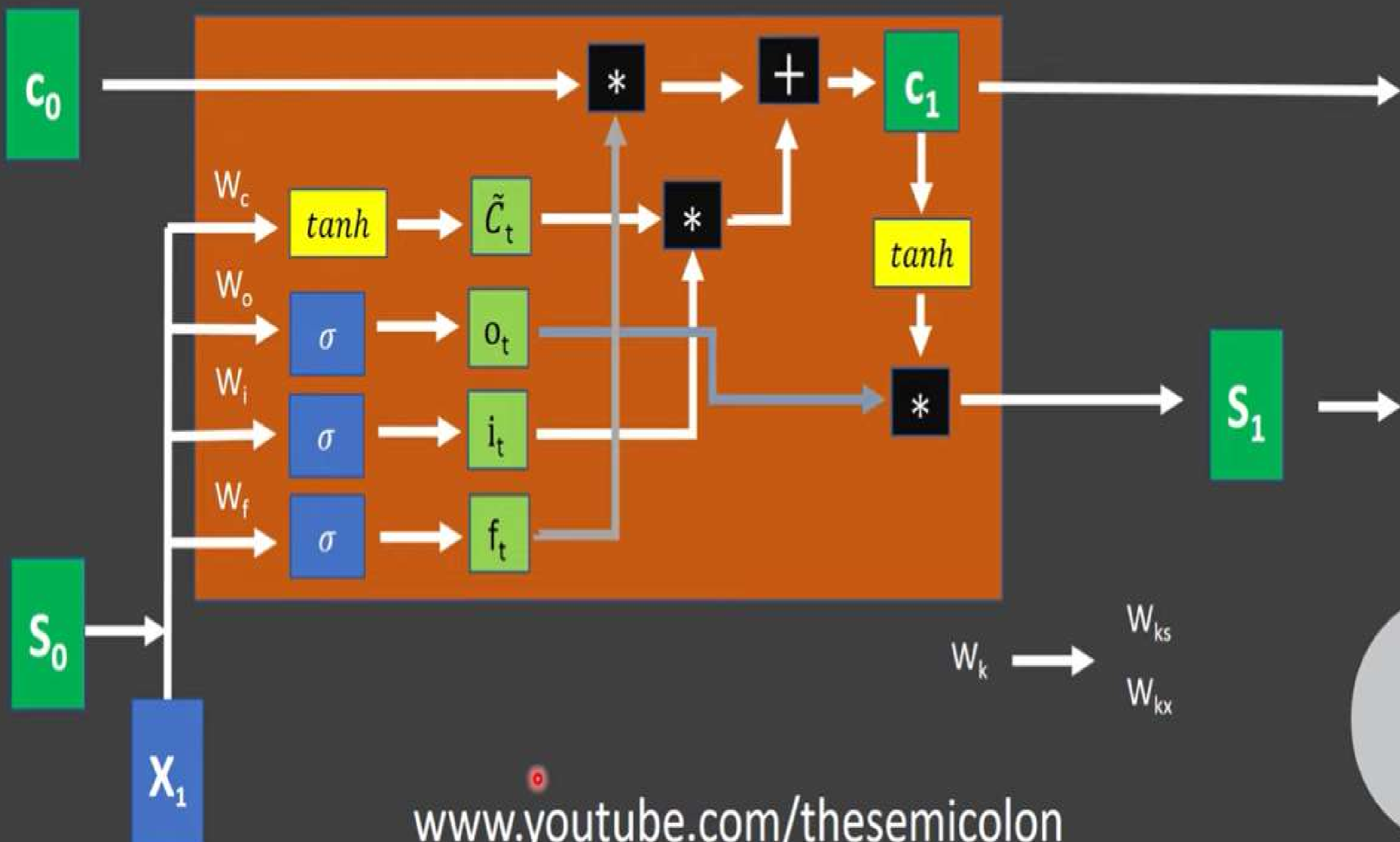
$i_t = \sigma(W_i S_{t-1} + W_i X_t)$ - Input Gate

$o_t = \sigma(W_o S_{t-1} + W_o X_t)$ - Output Gate

$\tilde{C}_t = tanh(W_c S_{t-1} + W_c X_t)$

$c_t = (I_t * \tilde{C}_t) + (f_t * c_{t-1})$ – Cell State

$h_t = o_t * tanh(c_t)$ – Output State



$W_k \rightarrow W_{ks}$, W_{kx}

www.youtube.com/thesemicolon