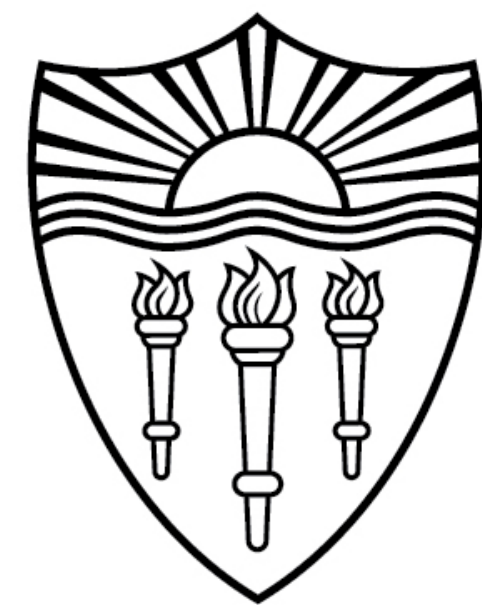CSCI 544: Applied Natural Language Processing

Recurrent Neural Networks
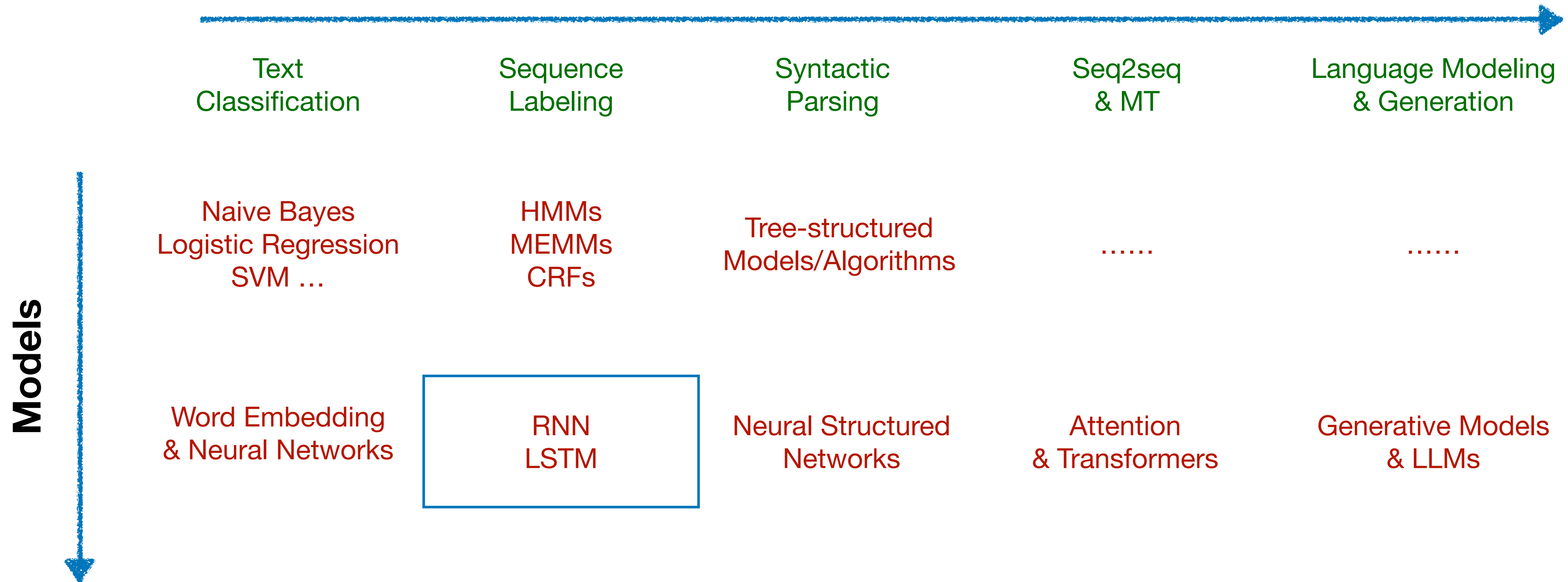
Xuezhe Ma (Max)

USC University of Southern California

# Course Organization

NLP Tasks

Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	Attention & Transformers	Generative Models & LLMs

Models

Recap: What is Sequence Labeling?

A type of structured prediction tasks

$$Y = <y_i, y_2, \ldots, y_n>$$

$$X = <x_i, x_2, \ldots, x_n>$$

NNP	VBZ	IN	NNP
USC	**is**	**in**	**California**
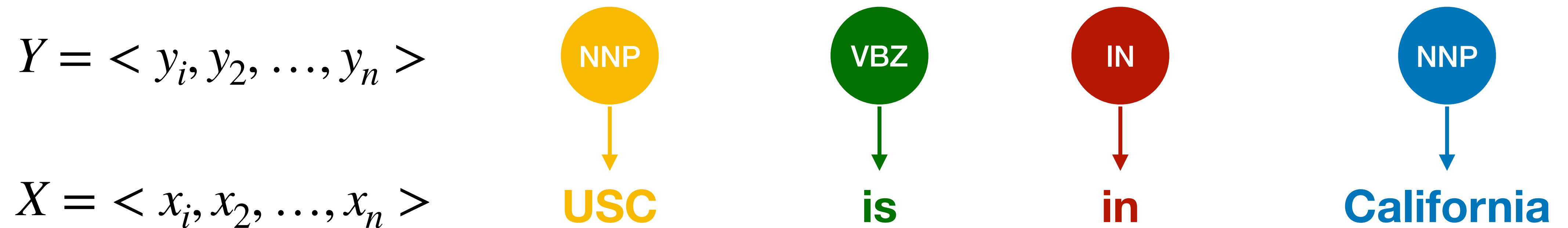
Assigning each token of X, e.g. x_i a corresponding label y_i

Recap: Classical Models for Sequence Labeling

- **Structured Models**
  - HMMs
  - MEMMs
  - CRFs
- **Decoding Algorithms**
  - Greedy decoding
  - Viterbi decoding

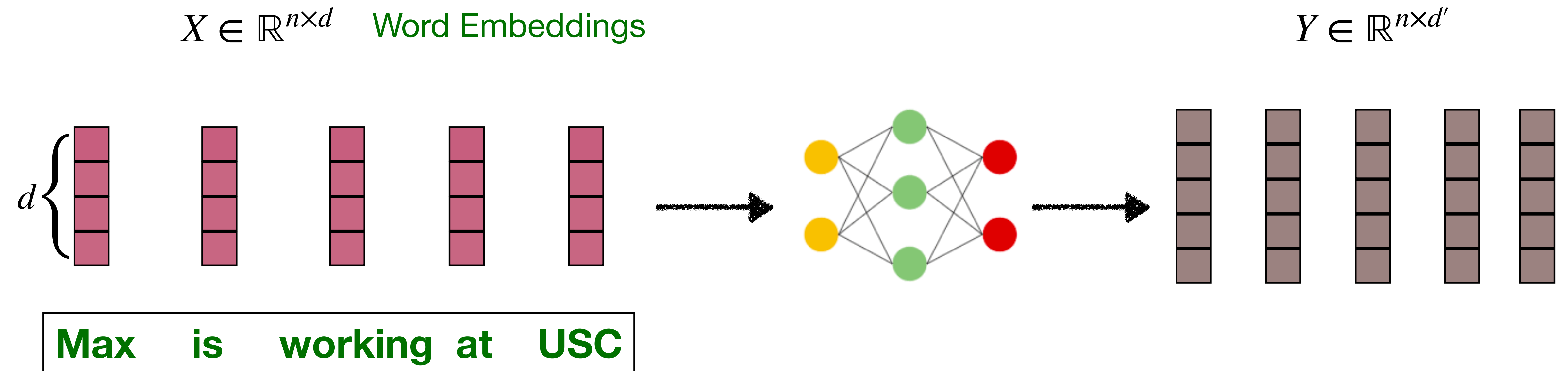Motivation: modeling dependencies between multiple labels/tags

An Essential Question

- Do we need structured models if the feature representations of the input sentence is perfect

VBD

The **cat** **sat** **on** a **mat**

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j \mid x_1, \ldots, x_n)$$

?

Our Goal: Sentence Representations

- **One hidden vector for each word**

$X \in \mathbb{R}^{n \times d}$ Word Embeddings $Y \in \mathbb{R}^{n \times d'}$

d {

Max   **is**   **working**   **at**   **USC**

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j \mid x_1, \ldots, x_n)$$

Our Goal: Sentence Representations

- **One hidden vector for each word**

$X \in \mathbb{R}^{n \times d}$ Word Embeddings

y_1 y_5

Classifier • • • Classifier

d

Max is working at USC
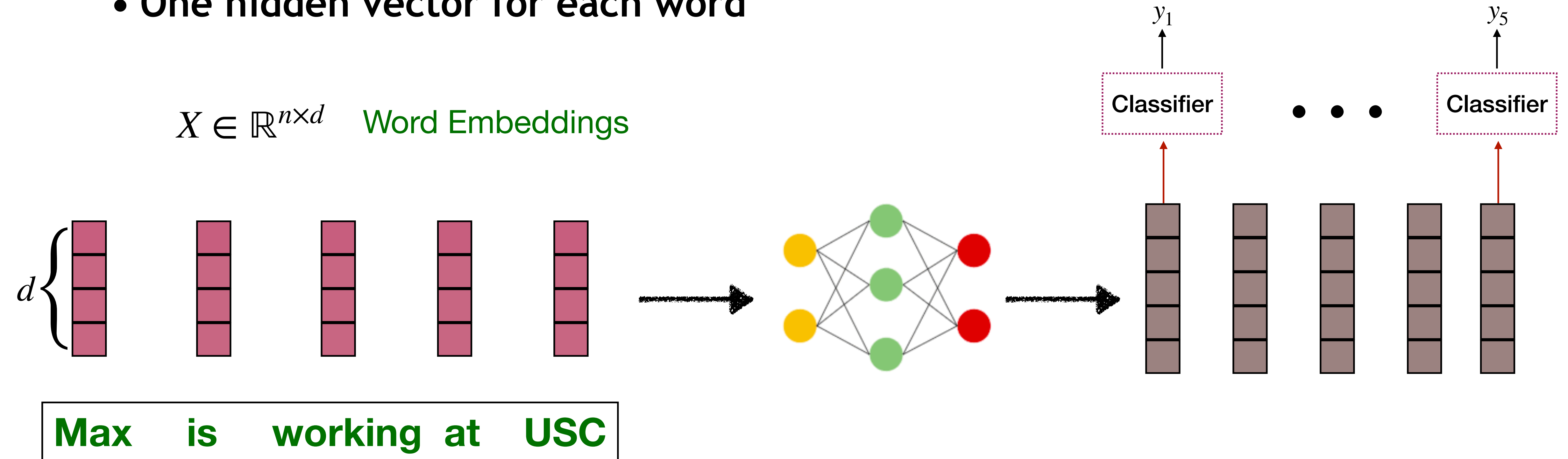
$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j \mid x_1, \ldots, x_n)$$

Convolutional Neural Networks (CNNs)

- **Basic Idea:** only model a segment of input with fixed window-size

$$f : \mathbb{R}^{k \times d} \to \mathbb{R}^{d'}$$

k is the window size

$k = 3$

$$Y = [y_1, y_2, \ldots, y_n] \in \mathbb{R}^{n \times d'}$$

$x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \qquad x_n \quad x_{n+1}$

$$X \in \mathbb{R}^{n \times d}$$

Pros:
- Simple architecture
- Parallel computation

Cons:
- Small context window
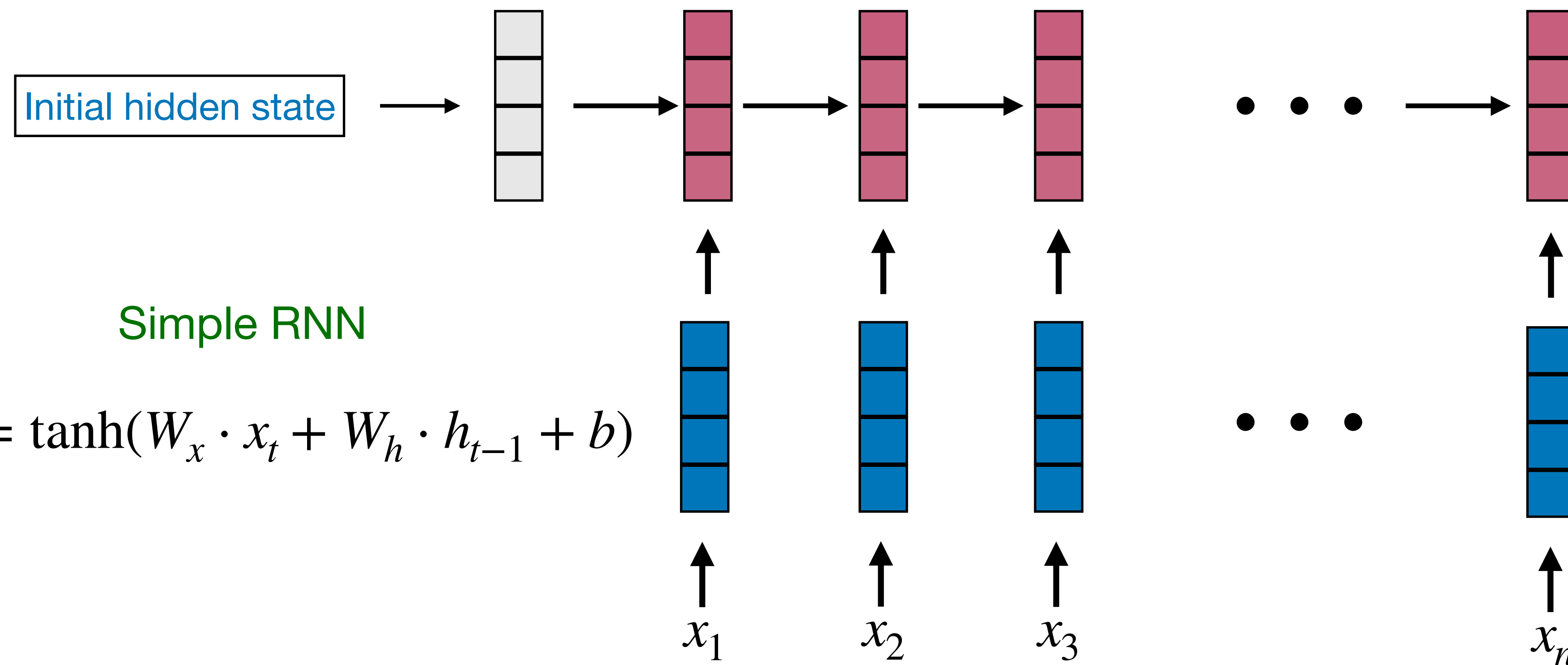- Not good at modeling long dependencies
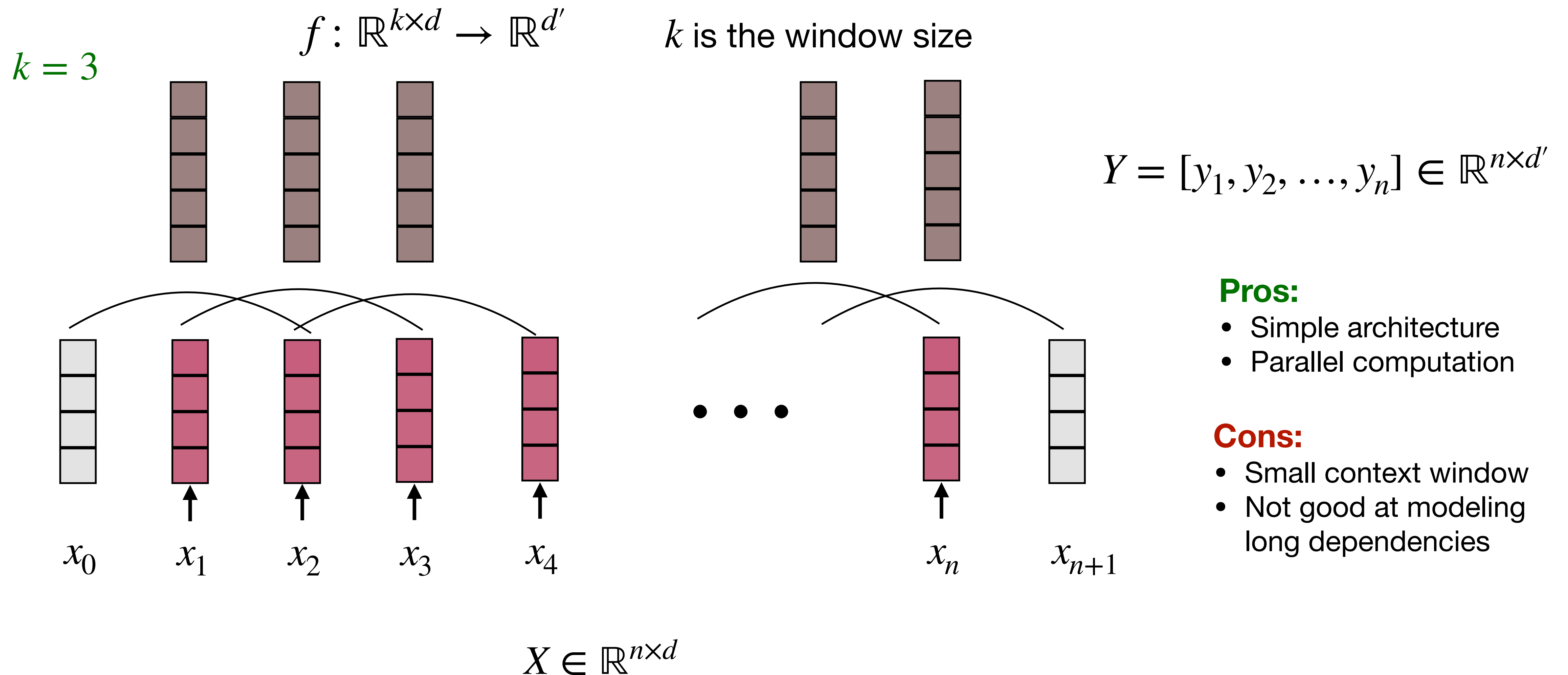
Recurrent Neural Networks

Neural Networks for Sentence Representations

- **Recurrent Neural Networks (RNNs)**
  - Re-using one feed forward network in a recurrent way

Initial hidden state

Simple RNN

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

x_1 x_2 x_3 x_n

Neural Networks for Sentence Representations
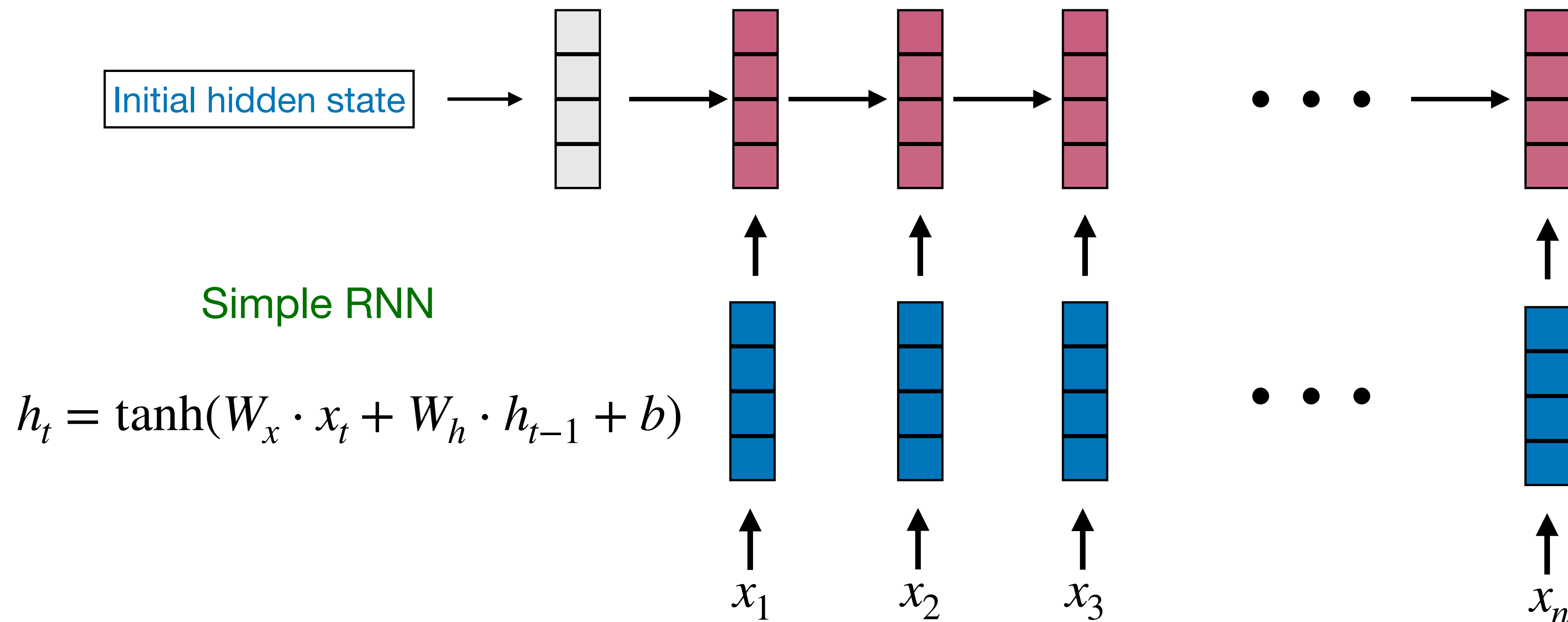
- **Recurrent Neural Networks (RNNs)**
  - Re-using one feed forward network in a recurrent way
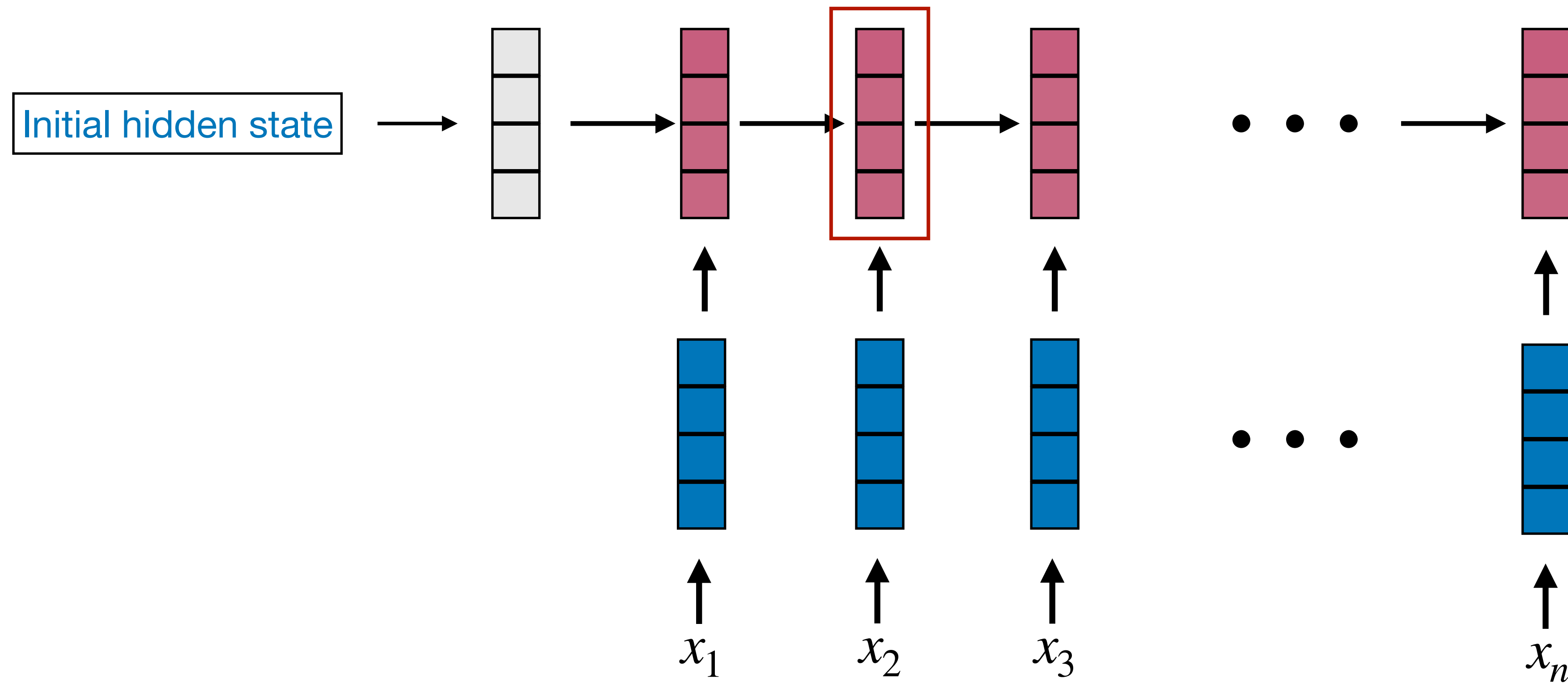
Inherent Markov Property
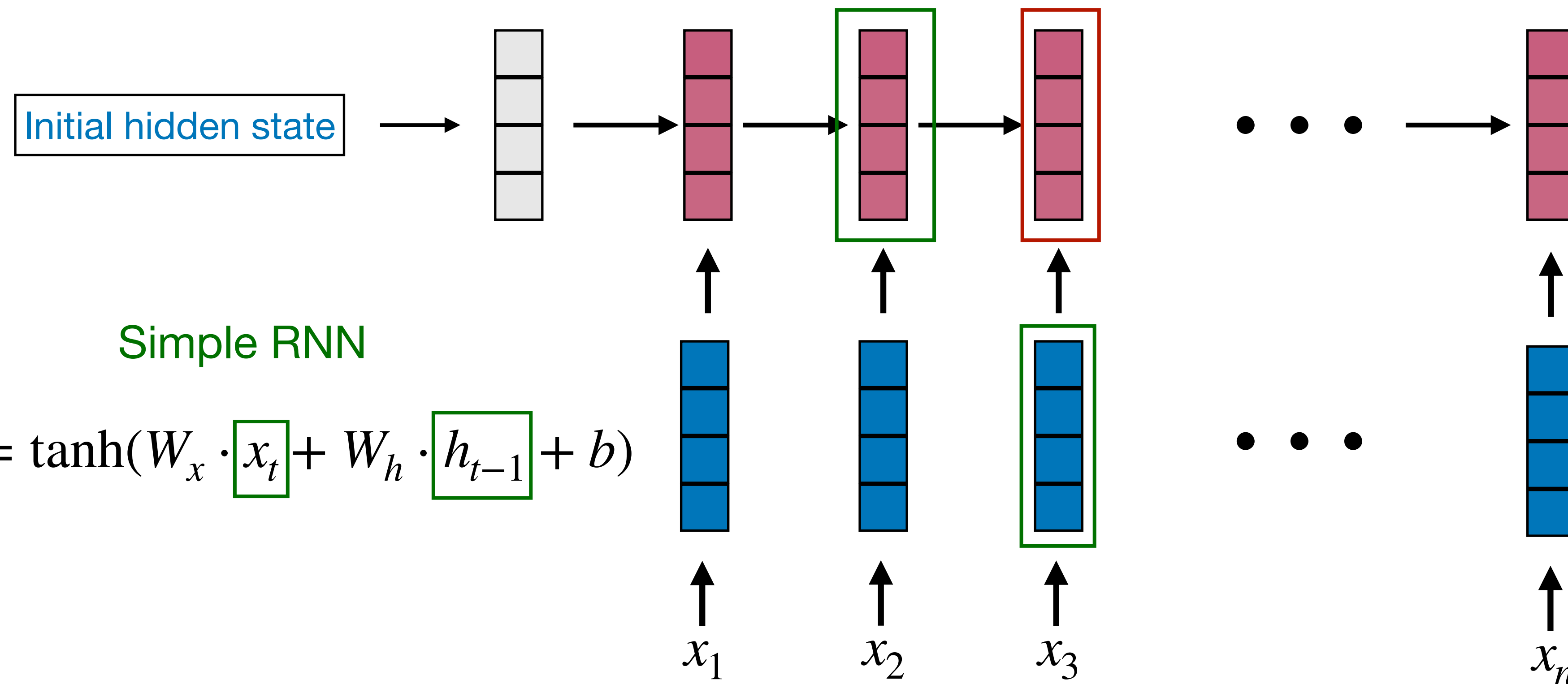
Simple RNN

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

Problems of Simple RNN

- **No future contexts**
- **Hard to train**
  - Gradient varnishing/exploding
- **Inefficient**
  - Sequential computation

Initial hidden state

Simple RNN

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

x_1 x_2 x_3 x_n
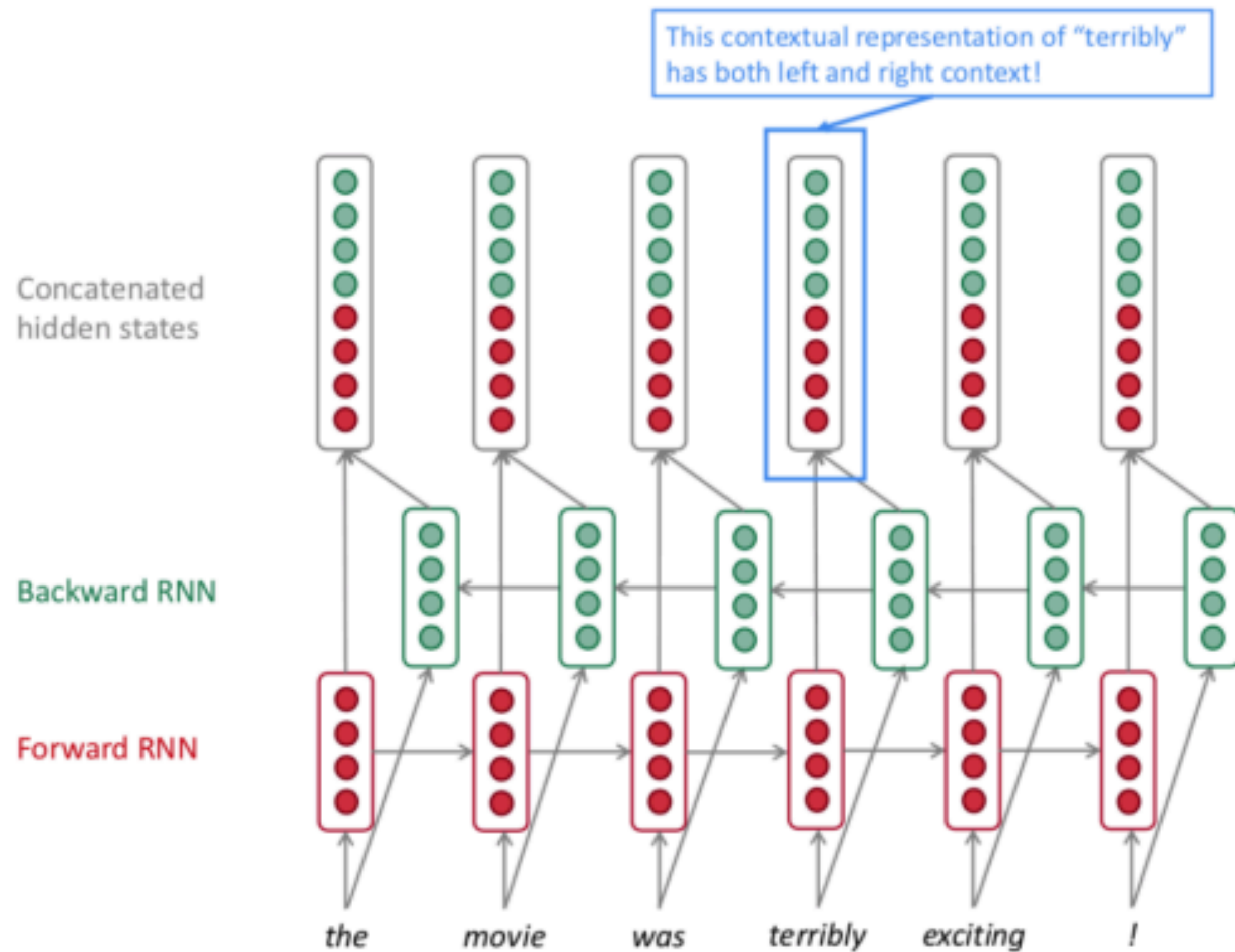
Bidirectional RNNs

- RNN **cannot** model **future** information

h_2 cannot access the information in $x_3, \ldots x_n$

Bidirectional RNNs

This contextual representation of "terribly" has both left and right context!

Concatenated hidden states

Backward RNN

Forward RNN

the   movie   was   terribly   exciting   !

the   movie   was   terribly   exciting   !
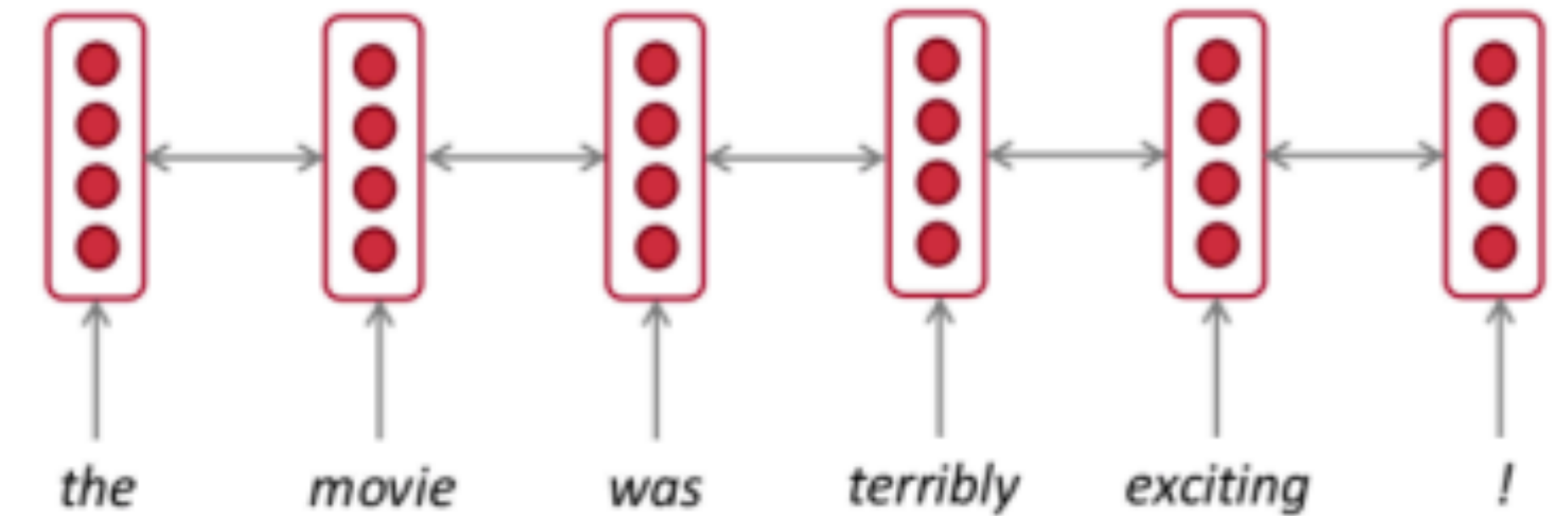
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \in \mathbb{R}^h$$

$$\overrightarrow{\mathbf{h}}_t = f_1(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t), t = 1, 2, \ldots n$$

$$\overleftarrow{\mathbf{h}}_t = f_2(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{x}_t), t = n, n-1, \ldots 1$$

$$\mathbf{h}_t = [\overleftarrow{\mathbf{h}}_t, \overrightarrow{\mathbf{h}}_t] \in \mathbb{R}^{2h}$$

Bidirectional RNNs

- **Bidirectional RNNs are only applicable if we have access to the entire input sequence**

- **If we do have entire input sequence, bidirectionality is powerful (and should be the default choice)**

- **A very common choice for sentence/document encoding: multi-layer bidirectional RNNs**

Problems of Simple RNN

- **No future contexts**
- **Hard to train**
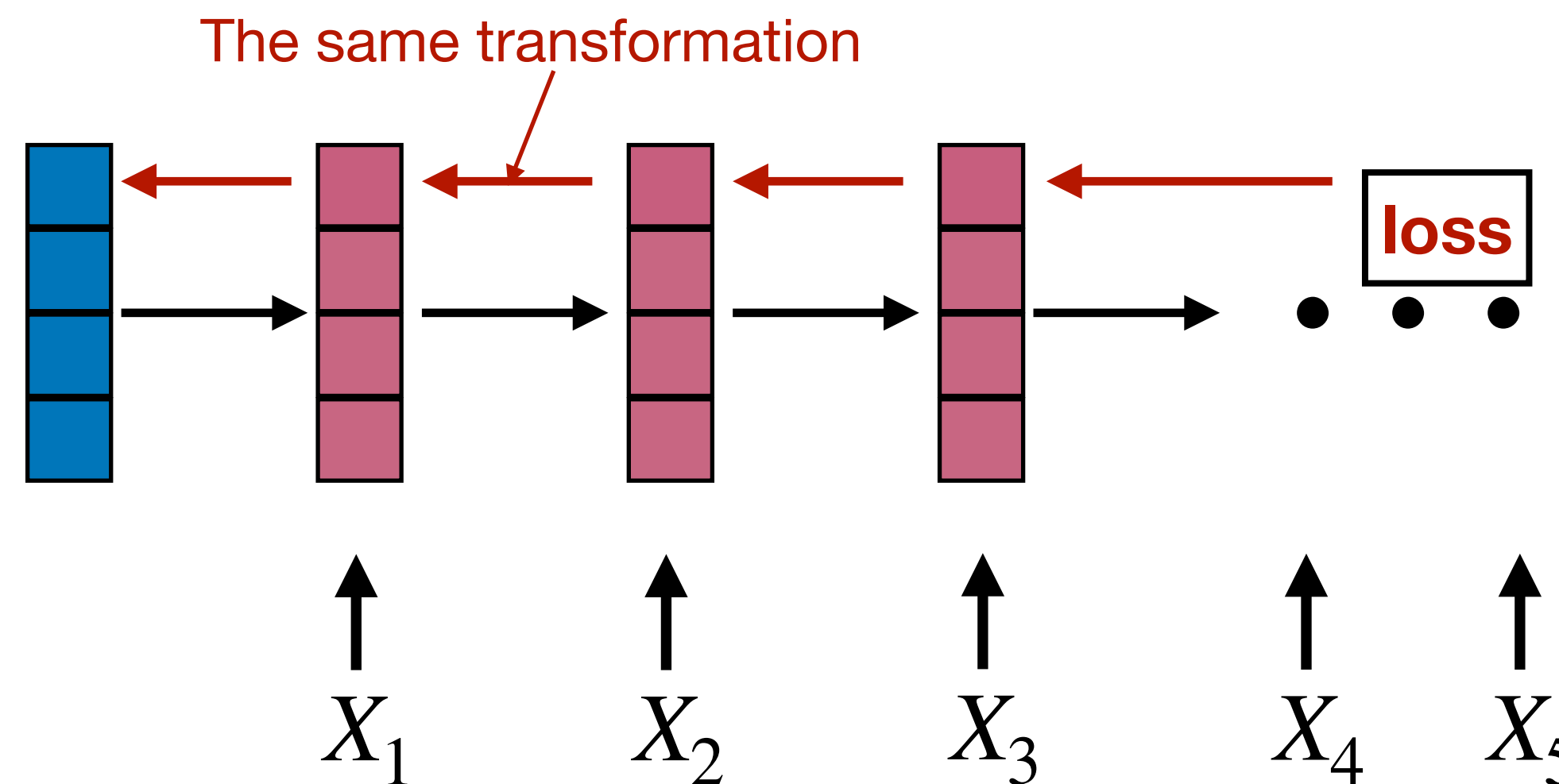 - Gradient varnishing/exploding
- **Inefficient**
  - Sequential computation

Simple RNN
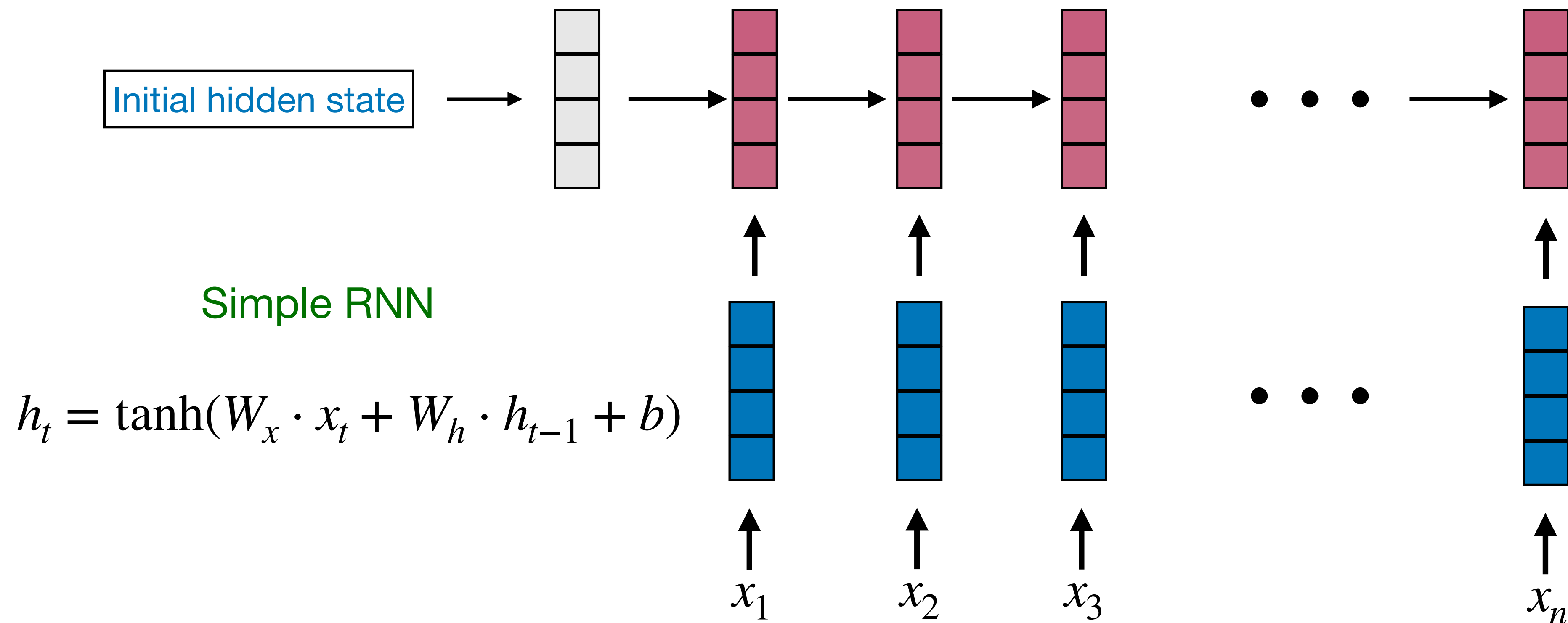
$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

Simple RNN is hard to train

- **Hard to capture long-distance information: vanishing/exploding gradients**

The same transformation

loss

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

$$\| \frac{\partial h_t}{\partial h_{t-1}} \| \sim \| W_h \|$$

$$\| \frac{\partial h_t}{\partial h_{t-m}} \| \sim \| W_h \|^m$$

$X_1 \qquad X_2 \qquad X_3 \qquad X_4 \quad X_5$

Why is this not a serious problem for multi-layer FFN/CNN?

Advanced RNN Variants

- **Long-short Term Memory (LSTMs)**
- **Gated Recurrent Units (GRUs)**

LSTMs

$$i_t = \sigma(\mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{U}^i \mathbf{x}_t + \mathbf{b}^i)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{U}^f \mathbf{x}_t + \mathbf{b}^f)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{U}^o \mathbf{x}_t + \mathbf{b}^o)$$

$$\mathbf{g}_t = \tanh(\mathbf{W}^g \mathbf{h}_{t-1} + \mathbf{U}^g \mathbf{x}_t + \mathbf{b}^g)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{g}_t \odot \mathbf{i}_t$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t$$

GRUs

$$\mathbf{r}_t = \sigma(\mathbf{W}^r \mathbf{h}_{t-1} + \mathbf{U}^r \mathbf{x}_t + \mathbf{b}^r)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}^z \mathbf{h}_{t-1} + \mathbf{U}^z \mathbf{x}_t + \mathbf{b}^z)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

LSTM is more an art than a science

LSTM: Long Short-Term Memory

$$\mathbf{i}_t = \sigma(\mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{U}^i \mathbf{x}_t + \mathbf{b}^i)$$

Input gate: $i_t \in (0,1)$

$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{U}^f \mathbf{x}_t + \mathbf{b}^f)$$

Forget gate: $f_t \in (0,1)$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{U}^o \mathbf{x}_t + \mathbf{b}^o)$$

Output gate: $o_t \in (0,1)$

$$\mathbf{g}_t = \tanh(\mathbf{W}^g \mathbf{h}_{t-1} + \mathbf{U}^g \mathbf{x}_t + \mathbf{b}^g)$$

Simple RNN

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{g}_t \odot \mathbf{i}_t$$

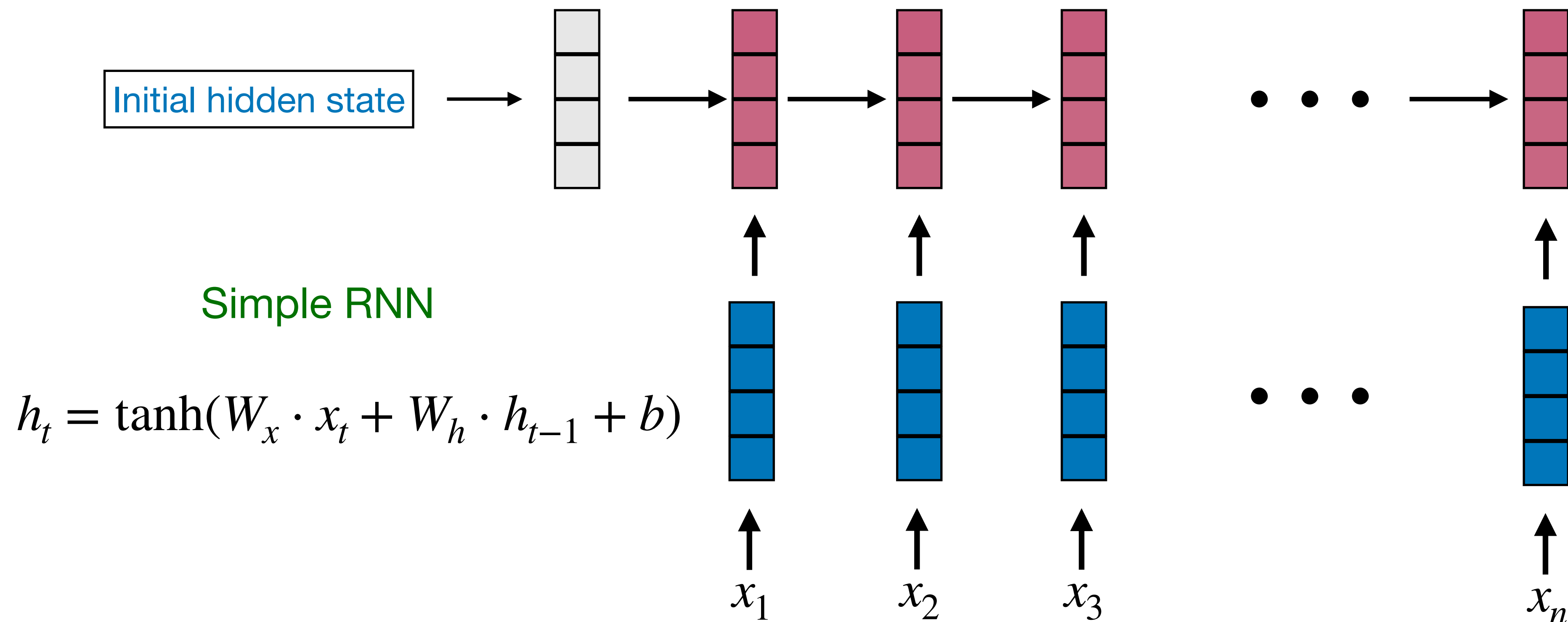**Cell state vector (internal memory)**

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t$$

Output

All the components are directly learned during training
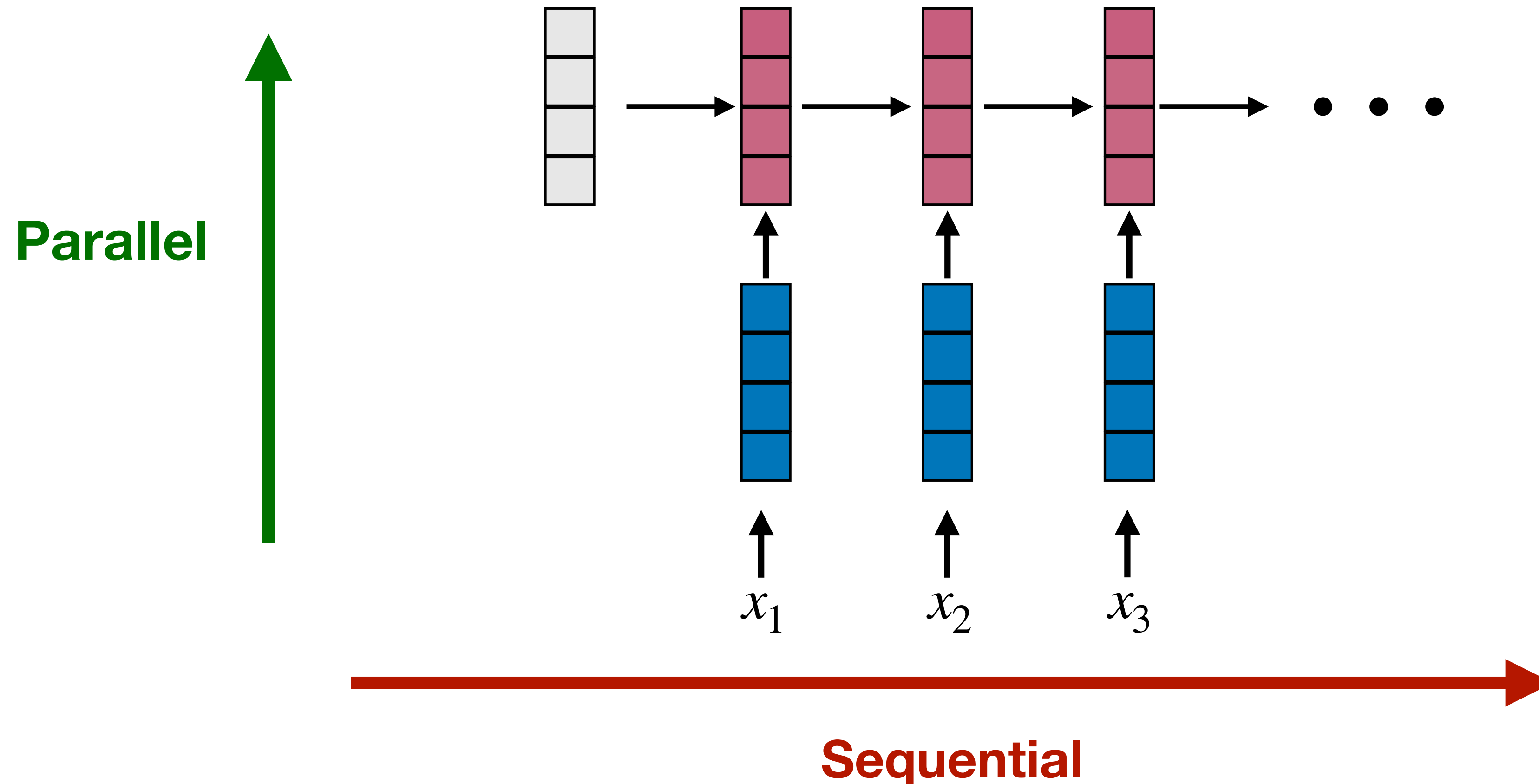
Problems of Simple RNN

- **No future contexts**
- **Hard to train**
  - Gradient varnishing/exploding
- **Inefficient**
  - Sequential computation

Initial hidden state

Simple RNN

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

x_1 x_2 x_3 x_n

Improving RNN efficiency

- **Simple Recurrent Unit**
  - Lei et al., 2018

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

Parallel

x_1 x_2 x_3

Sequential

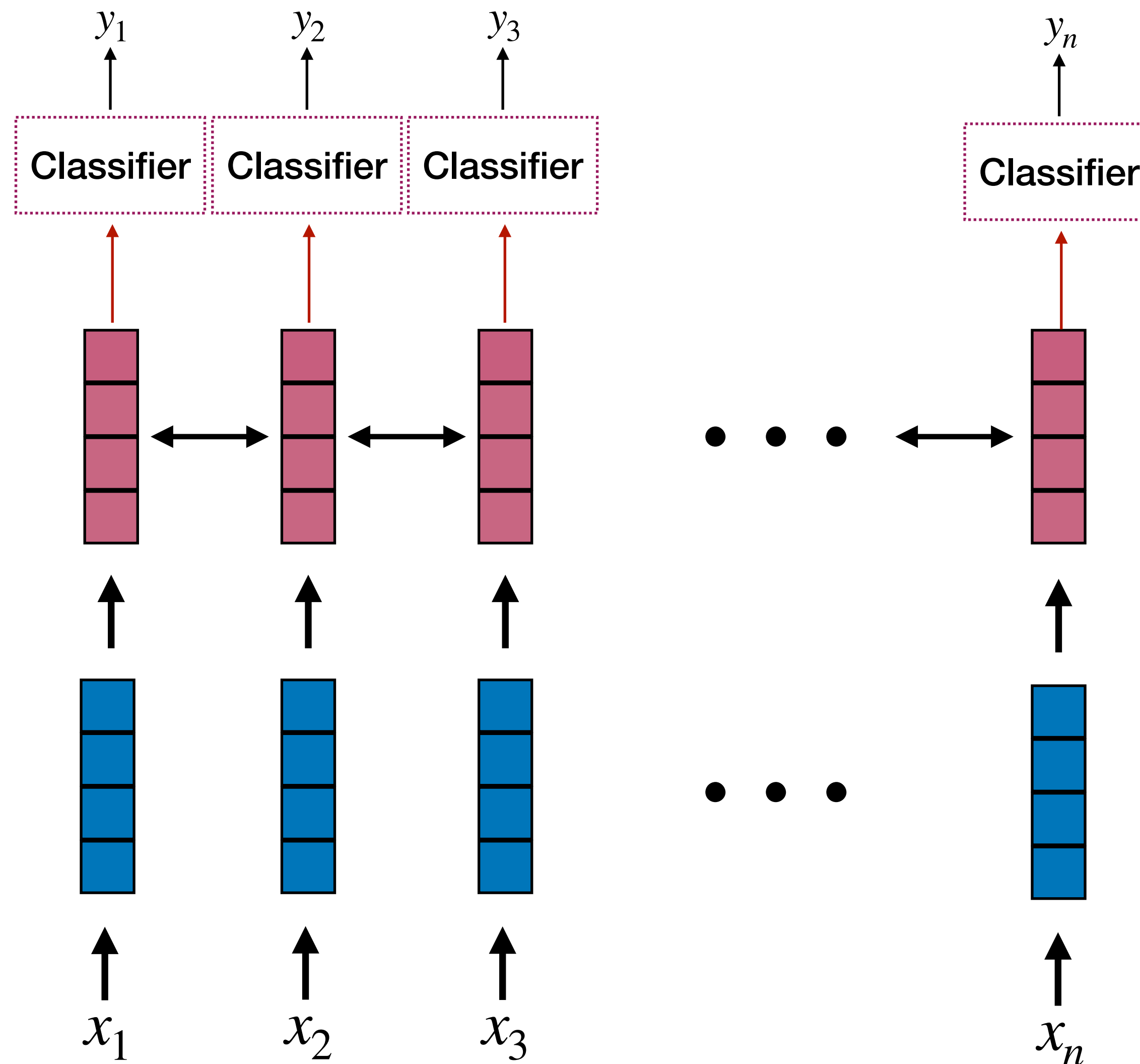# RNNs for Sequence Labeling

RNN for Sequence Labeling

- **A simple bidirectional LSTM model**
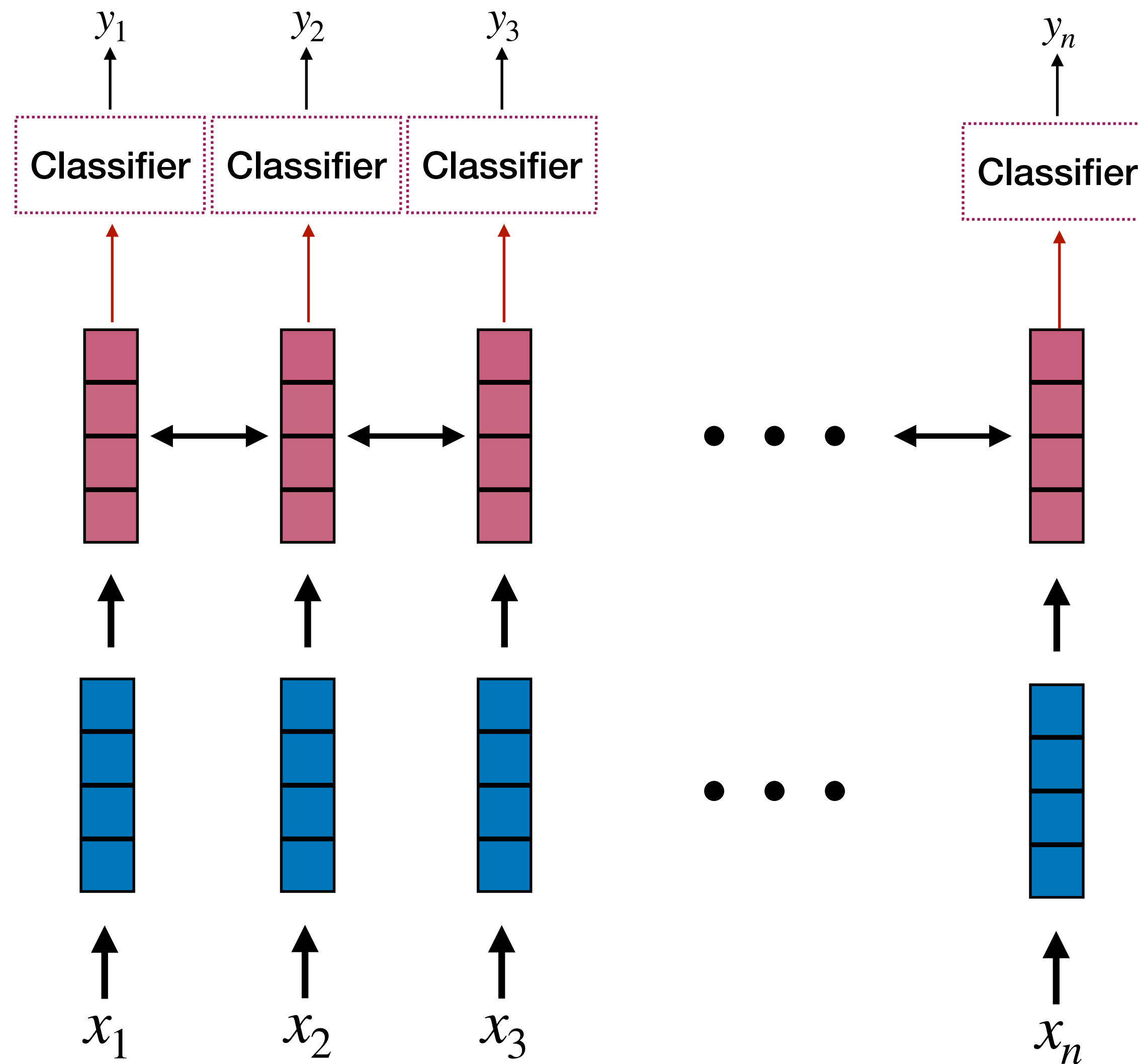
No structured models to consider dependencies between Y

Sequence Labeling

- **How to initialize parameters?**

y_1 y_2 y_3 ... y_n

Classifier Classifier Classifier ... Classifier

LSTM: $\mathbf{Unif}(-\sqrt{1/d}, \sqrt{1/d})$

Embedding: Word2Vec or GloVe

x_1 x_2 x_3 ... x_n

Sequence Labeling

- A simple bidirectional LSTM model

	POS Tagging	NER
HMM	96.4%	75.3%
CRF	97.0%	88.7%
CRF+external resources	97.3%	91.2%
BLSTM-Random	96.1%	80.7%
BLSTM-GloVe	96.9%	87.0%

BLSTM is not good enough!

Bidirectional LSTMs + CNNs

- **Bidirectional LSTM only encodes word-level information**
- **Spelling features are important**
  - Using CNN to model character-level information

Convolution Layers

LSTM Layers
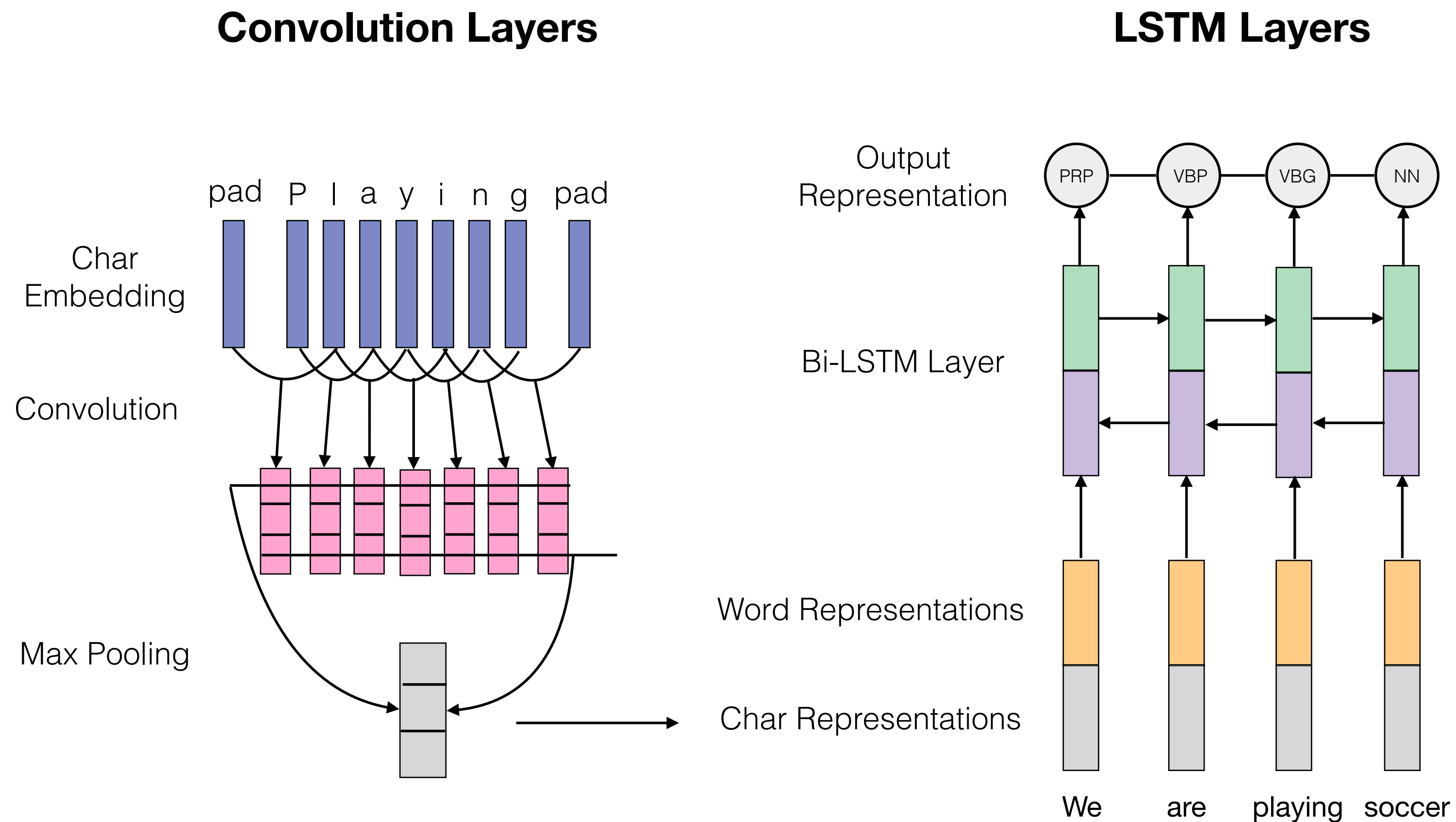
Sequence Labeling

- BLSTM-CNN

	POS Tagging	NER
CRF	97.0%	88.7%
CRF+external resources	97.3%	91.2%
BLSTM	96.9%	87.0%
BLSTM-CNN	97.3%	89.4%

BLSTM-CNN is better than CRF!

An Essential Question

- **Do we need structured models if the feature representations of the input sentence is perfect**

VBD

The **cat** **sat** **on** a **mat**

$$P(y_1, \ldots, y_n \,|\, x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j \,|\, x_1, \ldots, x_n)$$

?

BiLSTM + Word Embeddings is not strong enough…

Structured models are still useful?
Combining structured models with LSTM?

Recap: Log-Linear Models

- **MEMMs**

$$p(y_j \,|\, y_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, y_{j-1}, y_j))}{\sum_{y_j' \in \mathbb{Y}} \exp(v \cdot f(x_1, \ldots, x_m, i, y_{j-1}, y_j'))}$$
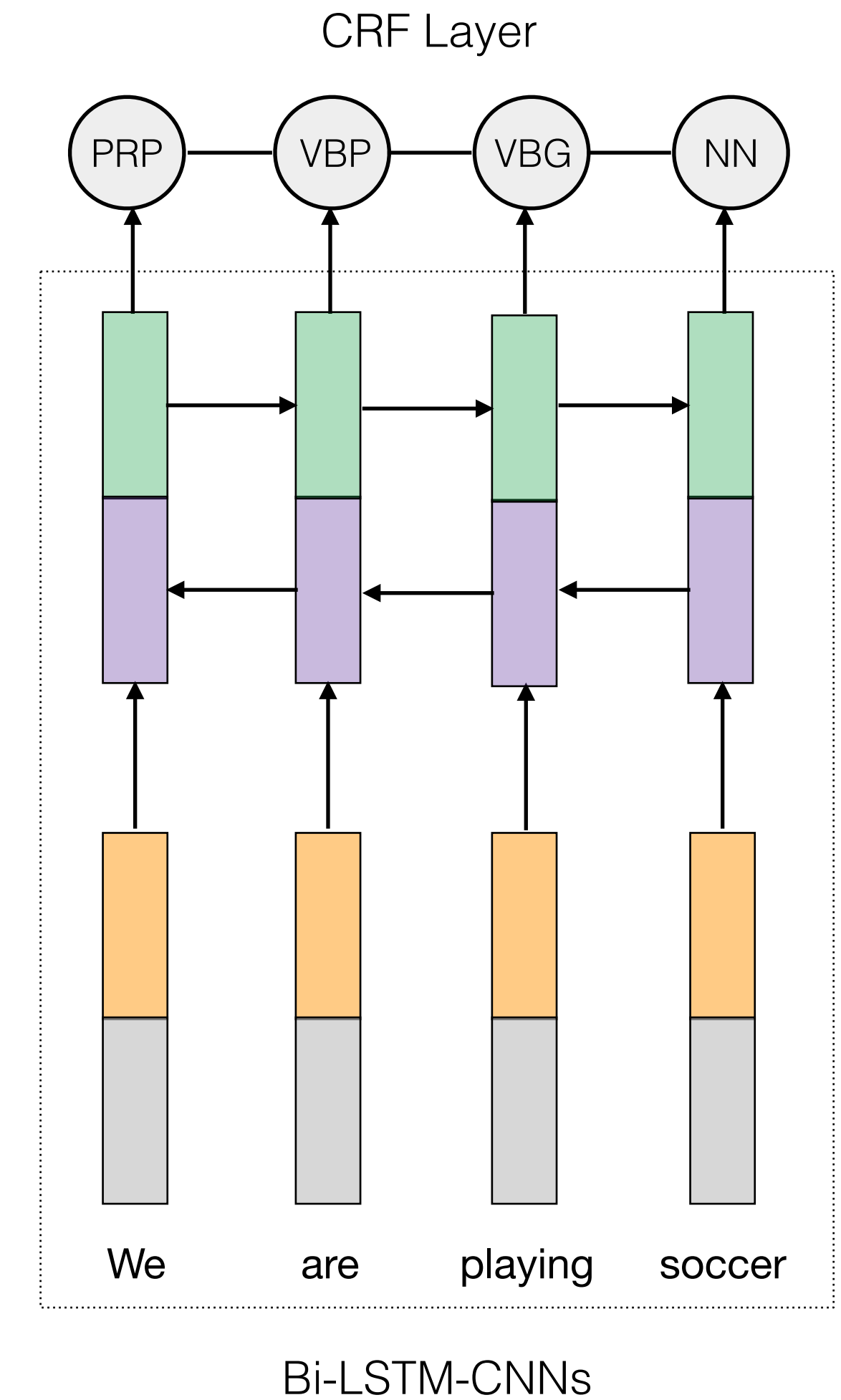
- **CRFs**

$$p(y_1, \ldots, y_m \,|\, x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, y_{j-1}, y_j))}{\sum_{y_1', \ldots, y_m' \in \mathbb{Y}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, y_{j-1}', y_j'))}$$

Sequence Labeling: BLSTM-CNNs-CRF

$$v \cdot f(x_1, \ldots, x_m, i, y_{j-1}, y_j) = \mathbf{W}^T_{y_{j-1}, y_j} \mathbf{h}_j$$

Parameters w.r.t y_{j-1}, y_j

LSTM hidden state at j

$$p(\boldsymbol{y} \mid \mathbf{x}; \mathbf{W}) = \frac{\prod_{j=1}^{n} \exp\left(\mathbf{W}_{y_{j-1}, y_j} \mathbf{h}_j\right)}{\sum_{y' \in \mathscr{Y}} \prod_{j=1}^{n} \exp\left(\mathbf{W}_{y'_{j-1}, y_j} \mathbf{h}_j\right)}$$

CRF Layer

PRP — VBP — VBG — NN

We   are   playing   soccer

Bi-LSTM-CNNs

Sequence Labeling

- **BLSTM-CNN-CRF**

	POS Tagging	NER
CRF	97.0%	88.7%
CRF+external resources	97.3%	91.2%
BLSTM	96.9%	87.0%
BLSTM-CNN	97.3%	89.4%
BLSTM-CNN-CRF	97.6%	91.2%

Structured models are still useful!

Reading Materials

- **Relavant Papers**
  - BLSTM-CNNs-CRF