THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound

Lecture 8: Language Models and RNNs

Language Models

Model the **probability of a sentence**; ideally be able to sample plausible sentences

Language Models

Model the **probability of a sentence**; ideally be able to sample plausible sentences

Why is this useful?

Language Models

Model the **probability of a sentence**; ideally be able to sample plausible sentences

Why is this useful?

$$\underset{wordsequence}{\arg\max}\, P(wordsequence \mid acoustics) =$$

$$\underset{wordsequence}{\arg\max}\, \frac{P(acoustics \mid wordsequence) \times P(wordsequence)}{P(acoustics)}$$

$$\underset{wordsequence}{\arg\max}\, P(acoustics \mid wordsequence) \times P(wordsequence)$$

* Slides from Louis-Philippe Morency

Language Models

Model the **probability of a sentence**; ideally be able to sample plausible sentences

Why is this useful?

$$\underset{wordsequence}{\arg\max}\, P(wordsequence \mid acoustics) =$$

$$\underset{wordsequence}{\arg\max}\, \frac{P(acoustics \mid wordsequence) \times \boxed{P(wordsequence)}}{P(acoustics)}$$

$$\underset{wordsequence}{\arg\max}\, P(acoustics \mid wordsequence) \times \boxed{P(wordsequence)}$$

* Slides from Louis-Philippe Morency

Simple **Language Models**: N-Grams

Given a word sequence: $w_{1:n} = [w_1, w_2, ..., w_n]$

We want to estimate $p(w_{1:n})$

* Slides from Louis-Philippe Morency

Simple **Language Models**: N-Grams

Given a word sequence: $w_{1:n} = [w_1, w_2, ..., w_n]$

We want to estimate $p(w_{1:n})$

Using **Chain Rule** of probabilities:

$$p(w_{1:n}) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \cdots p(w_n|w_{1:n-1})$$

Simple **Language Models**: N-Grams

Given a word sequence: $w_{1:n} = [w_1, w_2, ..., w_n]$

We want to estimate $p(w_{1:n})$

Using **Chain Rule** of probabilities:

$$p(w_{1:n}) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \cdots p(w_n|w_{1:n-1})$$

Bi-gram Approximation:

$$p(w_{1:n}) = \prod_{k=1}^{n} p(w_k|w_{k-1})$$

N-gram Approximation:

$$p(w_{1:n}) = \prod_{k=1}^{n} p(w_k|w_{k-N+1:k-1})$$

* Slides from Louis-Philippe Morency

Estimating **Probabilities**

N-gram conditional probabilities can be estimated based on raw concurrence counts in the observed sequences

Bi-gram:

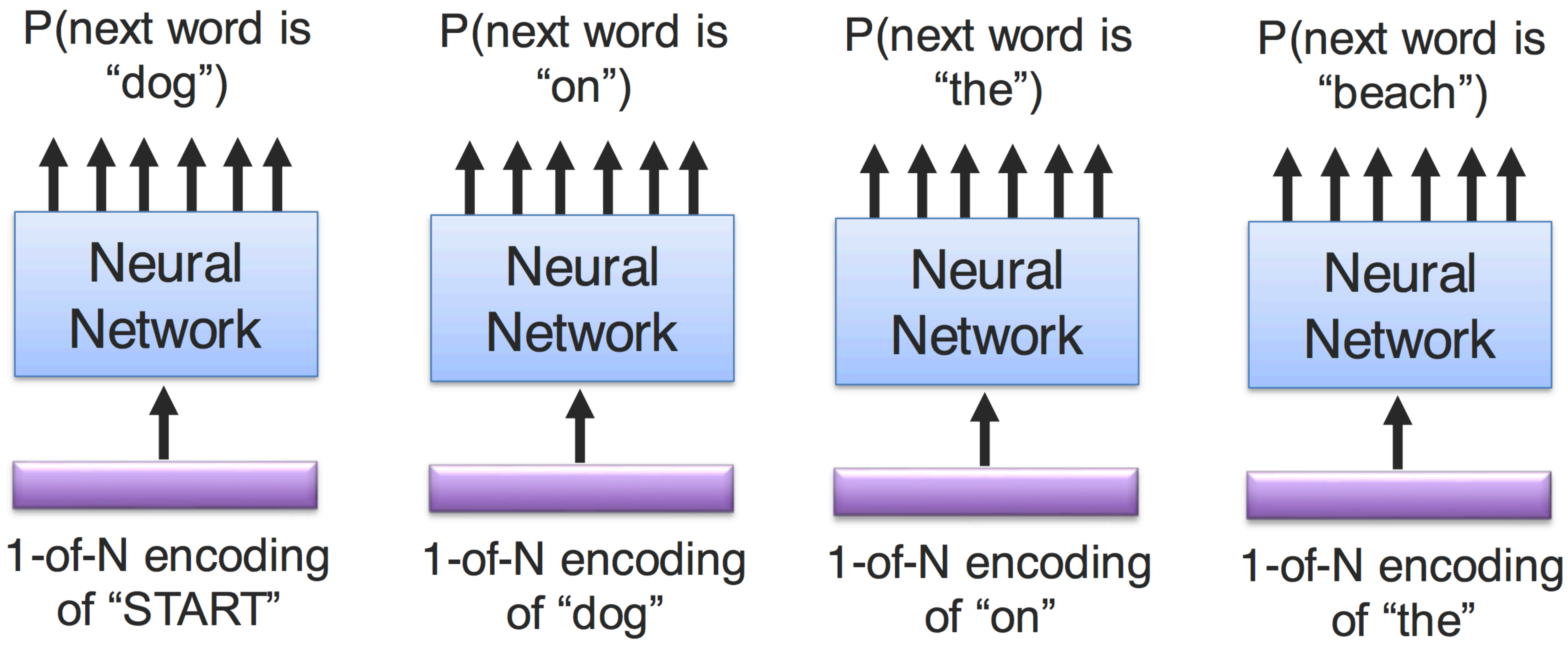
$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

N-gram:

$$p(w_n|w_{n-N-1:n-1}) = \frac{C(w_{n-N-1:n-1}w_n)}{C(w_{n-N-1:n-1})}$$

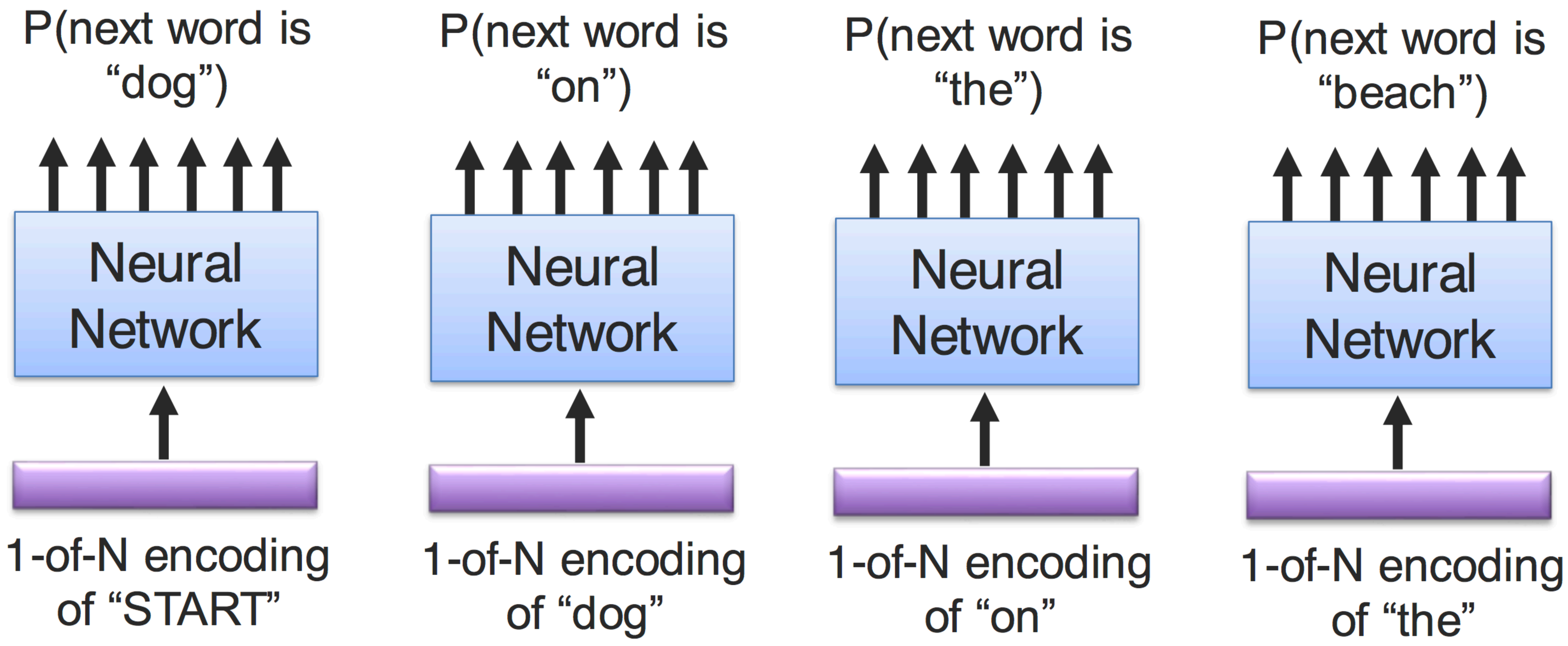
* Slides from Louis-Philippe Morency

Neural-based Unigram Language Mode



* Slides from Louis-Philippe Morency

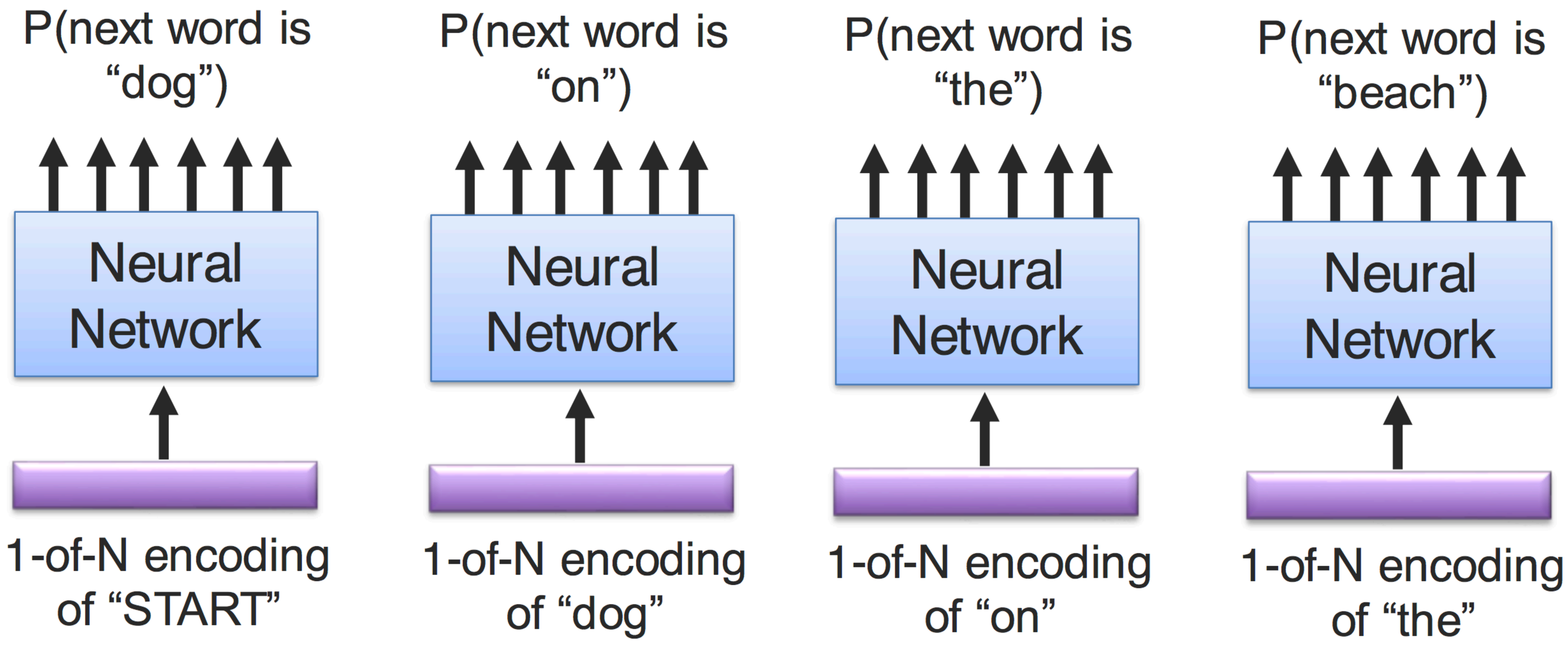
Neural-based Unigram Language Mode



Problem: Does not model sequential information (too local)

* Slides from Louis-Philippe Morency

Neural-based Unigram Language Mode

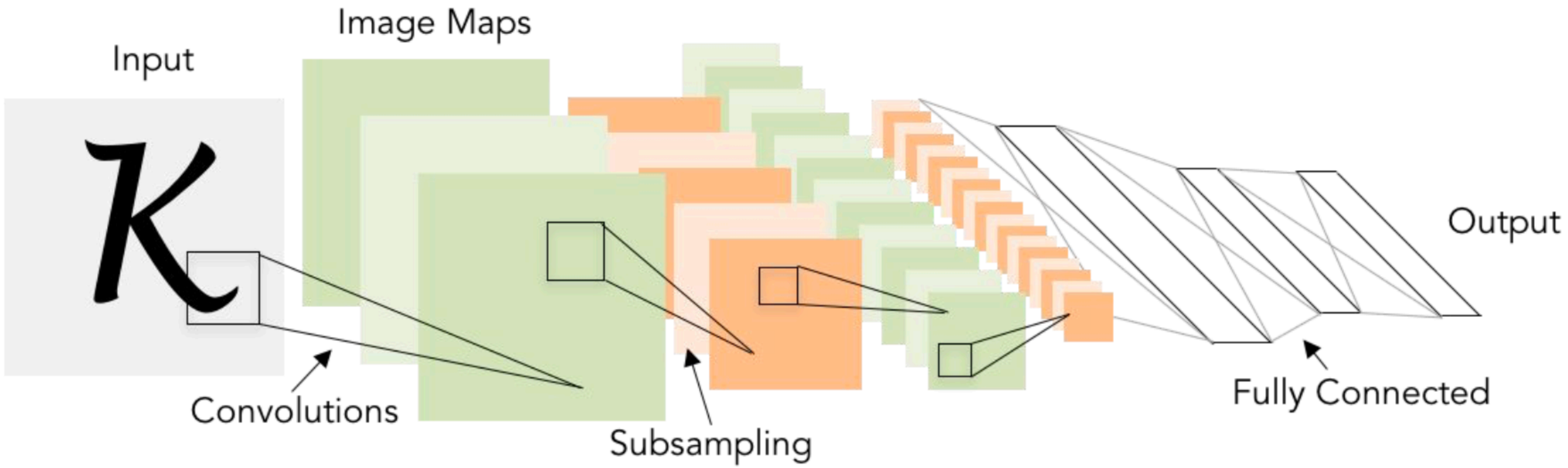


Problem: Does not model sequential information (too local)

We need sequence modeling!

* Slides from Louis-Philippe Morency

Sequence Modeling



Why Model **Sequences**?

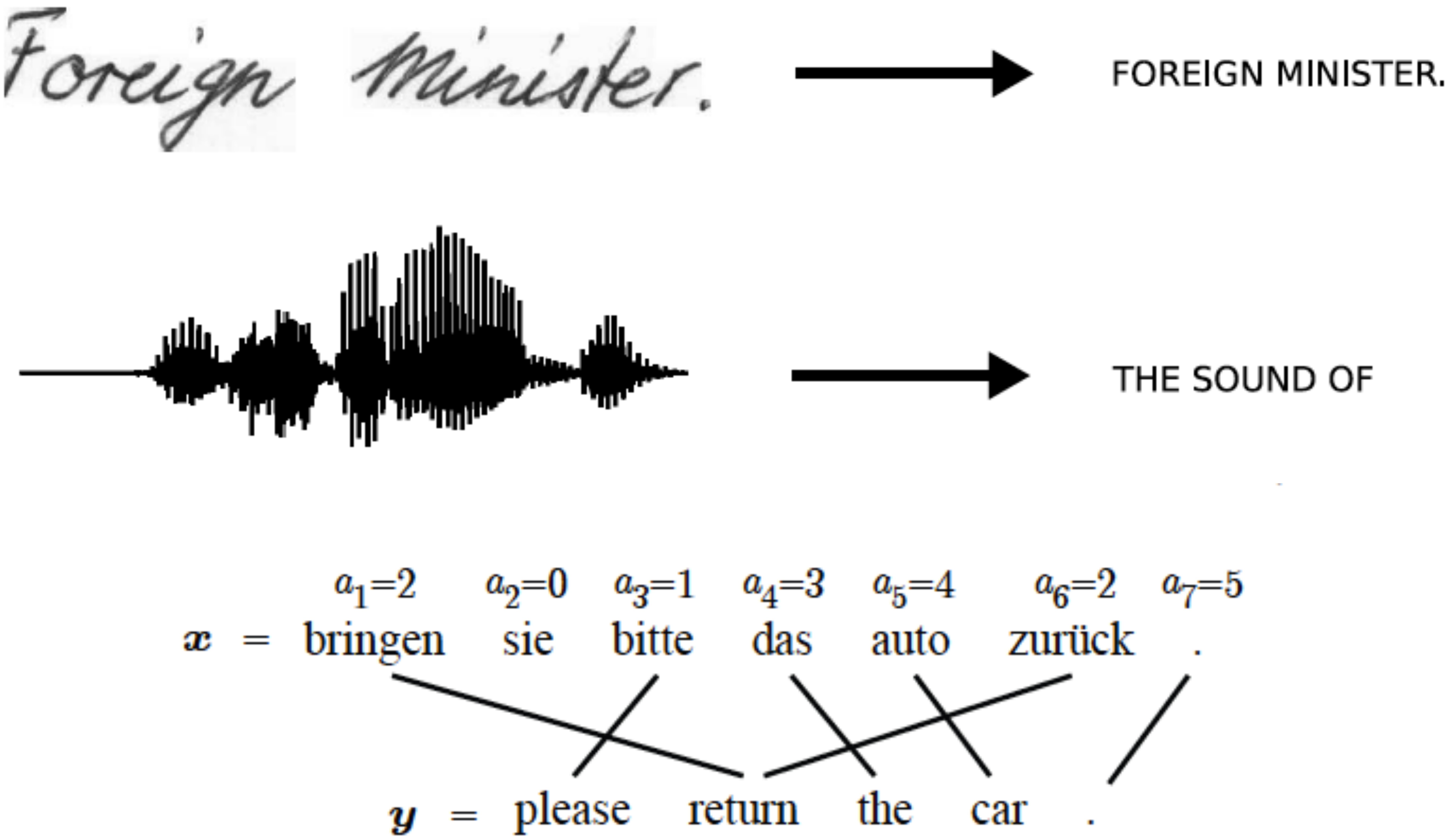
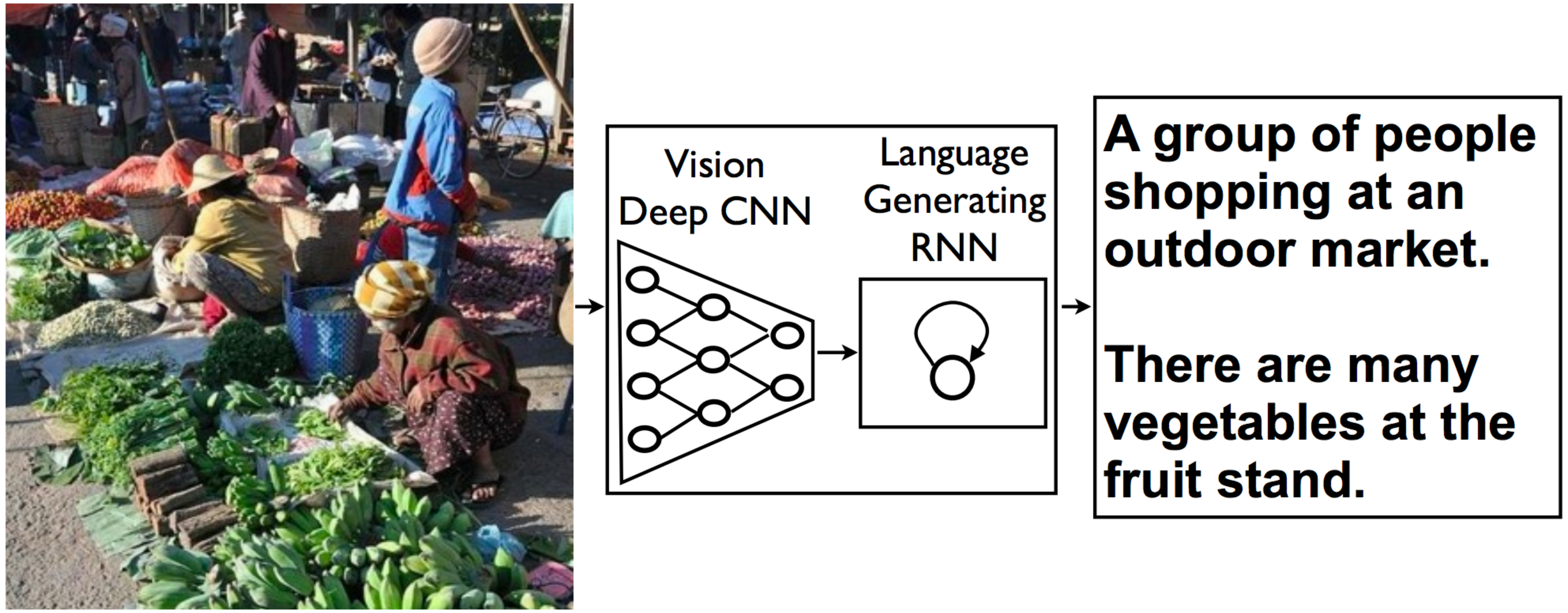


Image Credit: Alex Graves and Kevin Gimpel

Multi-modal tasks

[Vinyals *et al.*, 2015]



Sequences where you don't expect them …

Classify images by taking a series of "glimpses"

[Gregor et al., ICML 2015]

[Mnih et al., ICLR 2015]

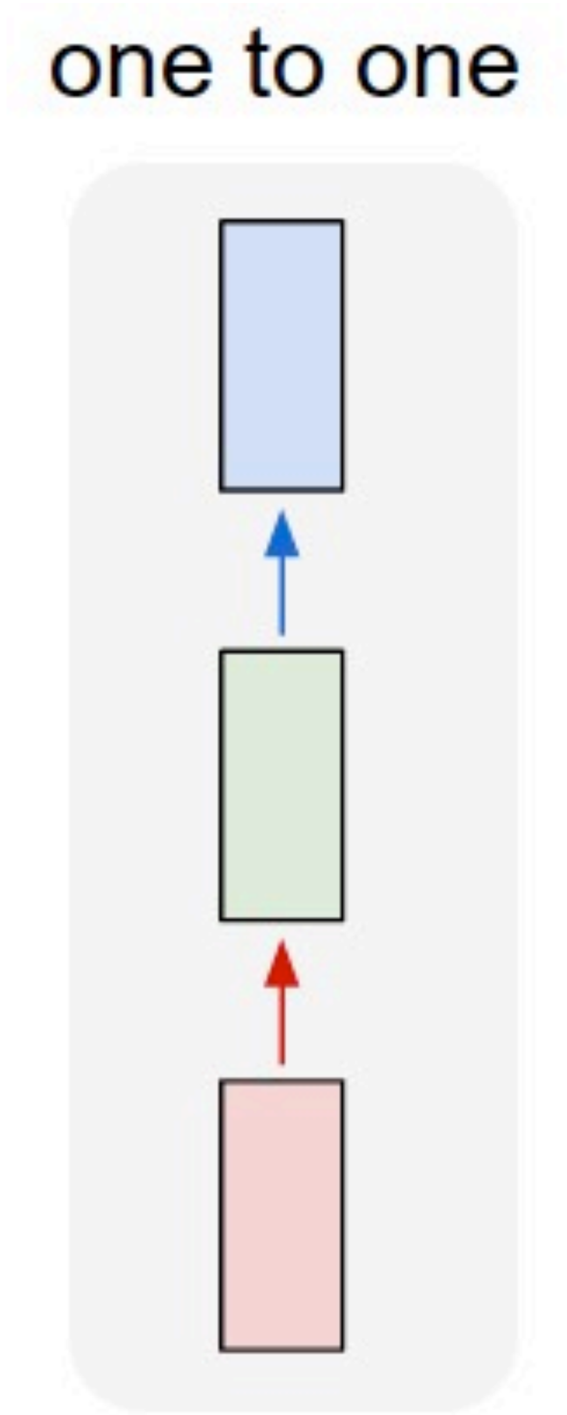
Sequences where you don't expect them …

Classify images by taking a series of "glimpses"

[Gregor et al., ICML 2015]

[Mnih et al., ICLR 2015]

Sequences in Inputs or Outputs?



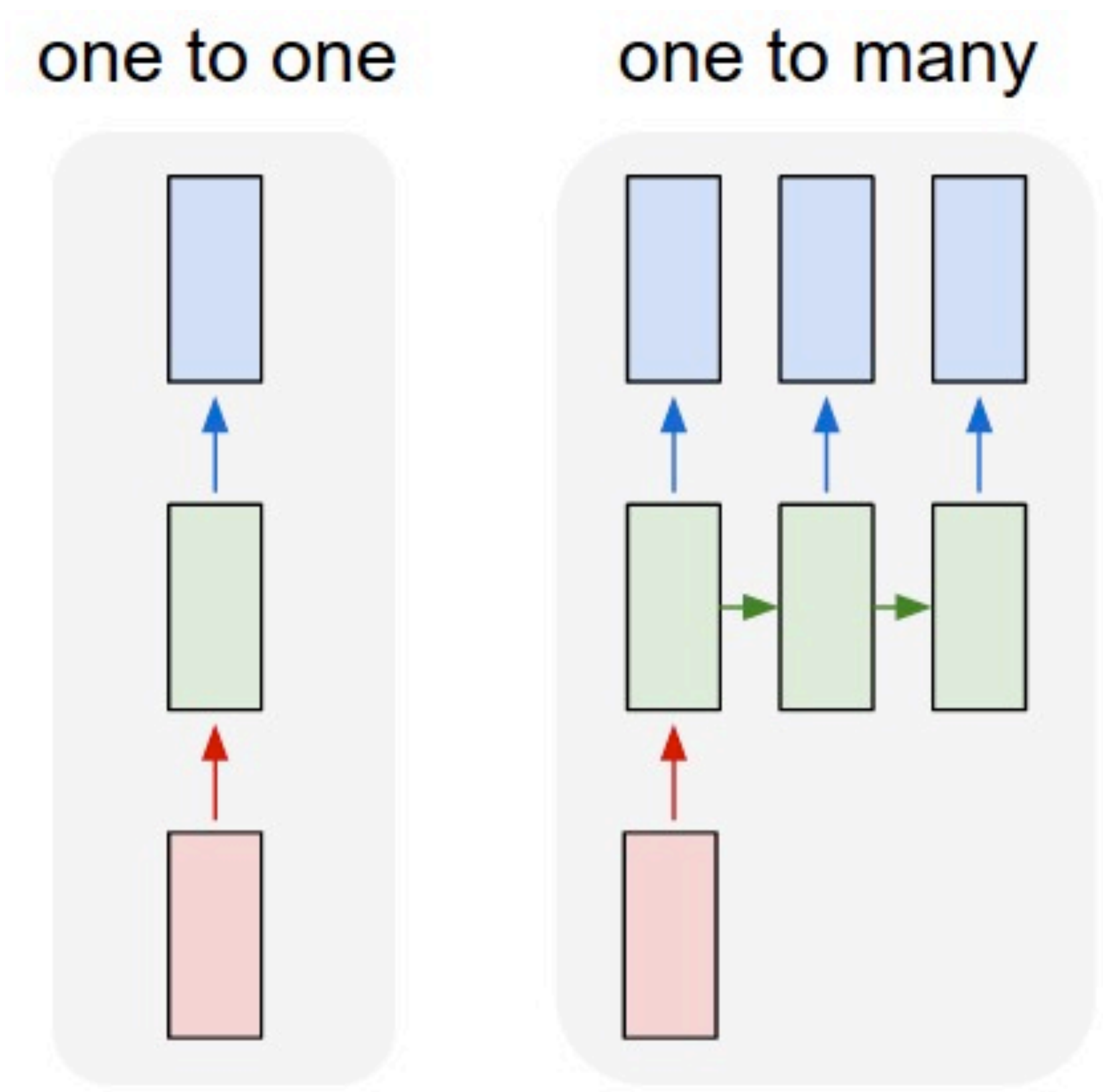
Input: No sequence

Output: No seq.

Example:
"standard"
classification /
regression problems

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Sequences in Inputs or Outputs?

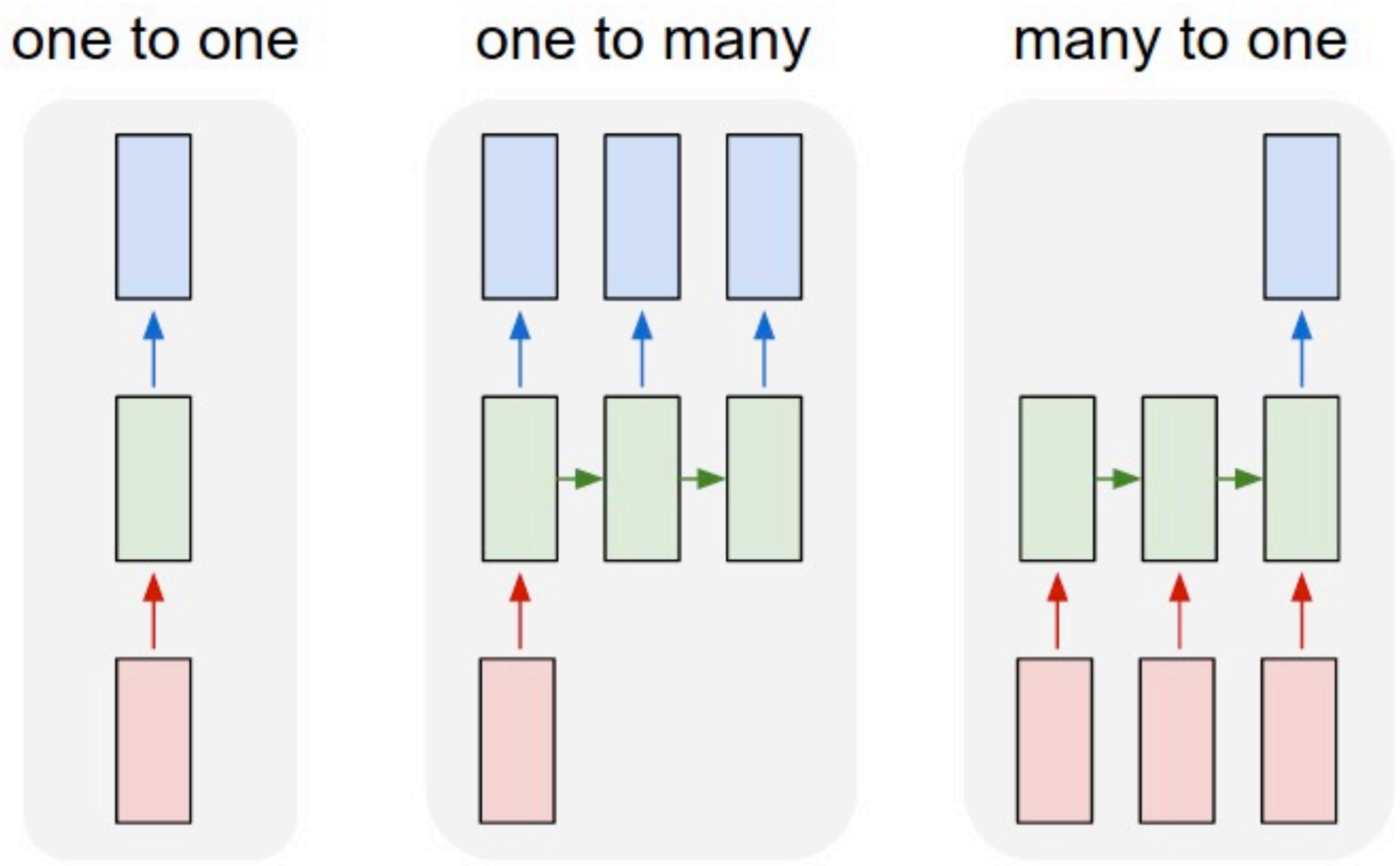


Input: No sequence
Output: No seq.
Example:
"standard" classification / regression problems

Input: No sequence
Output: Sequence
Example:
Im2Caption

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Sequences in Inputs or Outputs?



Input: No sequence
Output: No seq.
Example: "standard" classification / regression problems

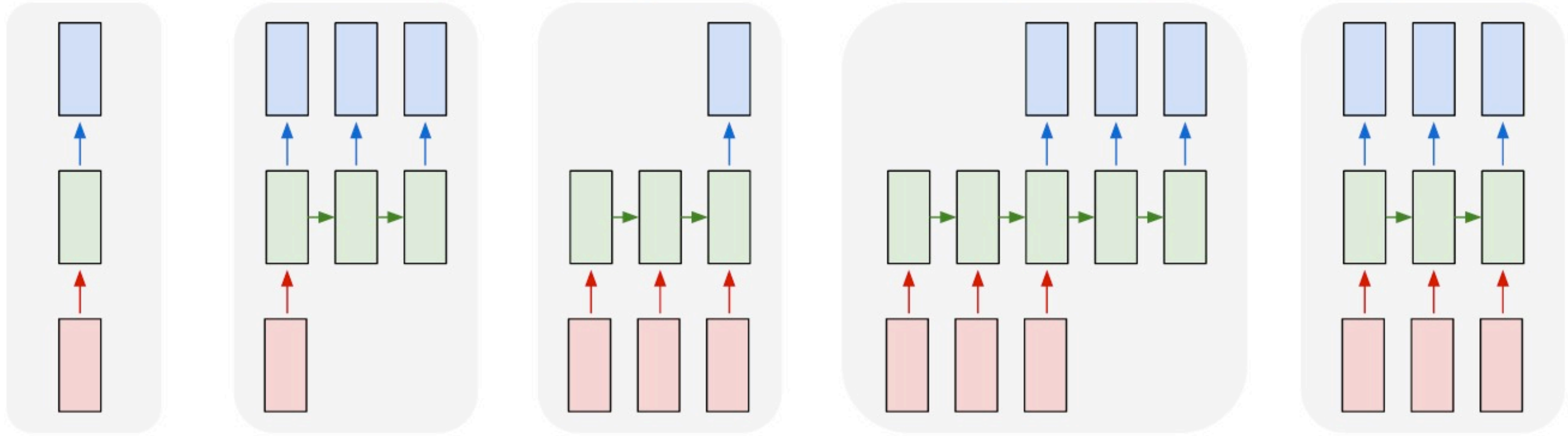
Input: No sequence
Output: Sequence
Example: Im2Caption

Input: Sequence
Output: No seq.
Example: sentence classification, multiple-choice question answering

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Sequences in Inputs or Outputs?

one to one | one to many | many to one | many to many | many to many



Input: No sequence
Output: No seq.
Example: "standard" classification / regression problems

Input: No sequence
Output: Sequence
Example: Im2Caption

Input: Sequence
Output: No seq.
Example: sentence classification, multiple-choice question answering

Input: Sequence
Output: Sequence
Example: machine translation, video captioning, open-ended question answering, video question answering

Key Conceptual Ideas

Parameter Sharing

— in computational graphs = adding gradients

“**Unrolling**”

— in computational graphs with parameter sharing

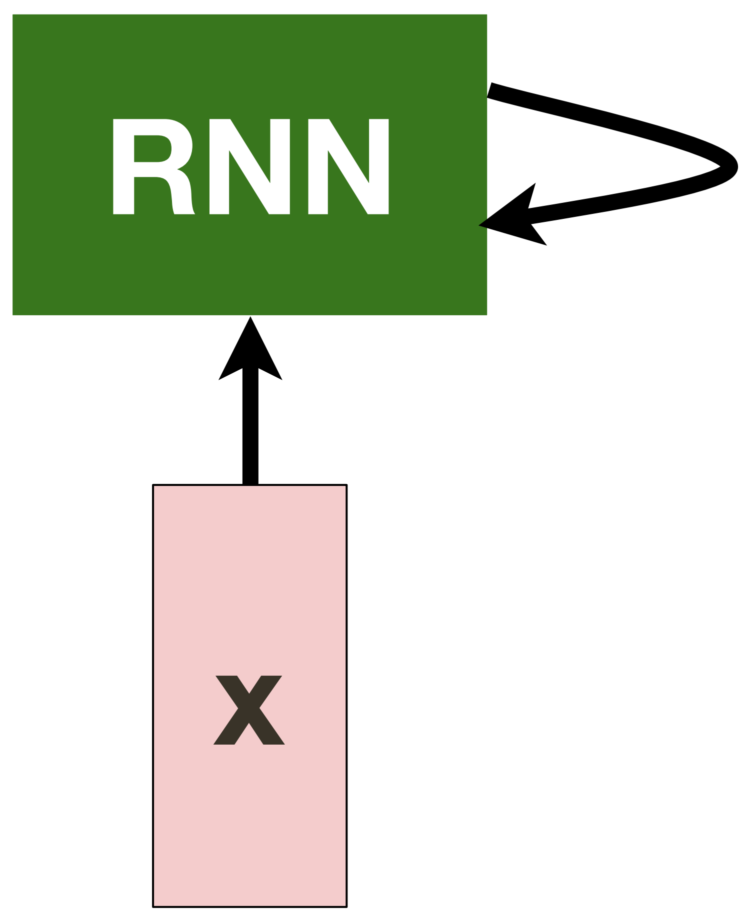
Parameter Sharing + “Unrolling”

— Allows modeling **arbitrary length sequences**!

— Keeps number of parameters in check

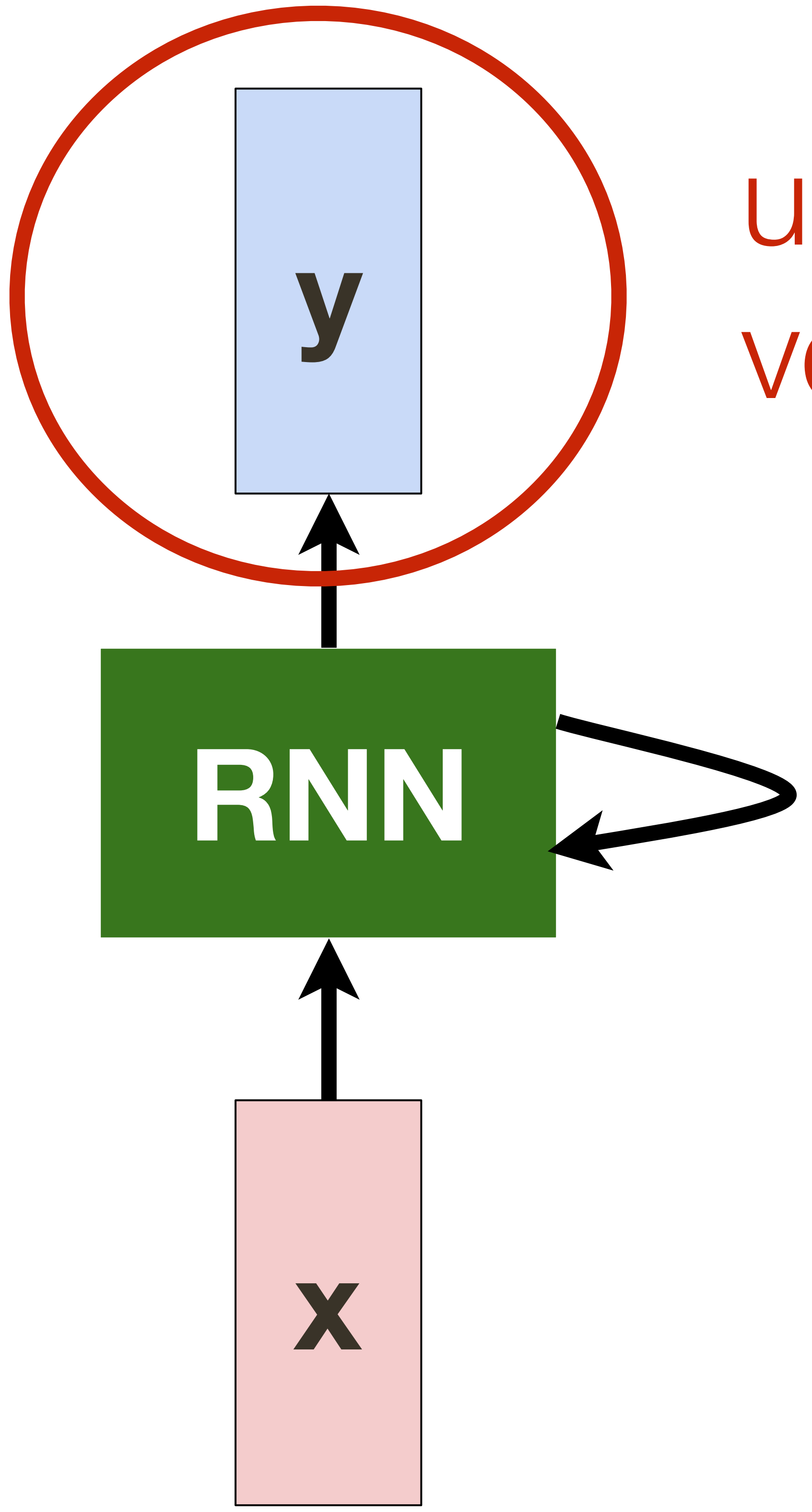
* slide from Dhruv Batra

Recurrent Neural Network



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Recurrent Neural Network



usually want to predict a vector at some time steps

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

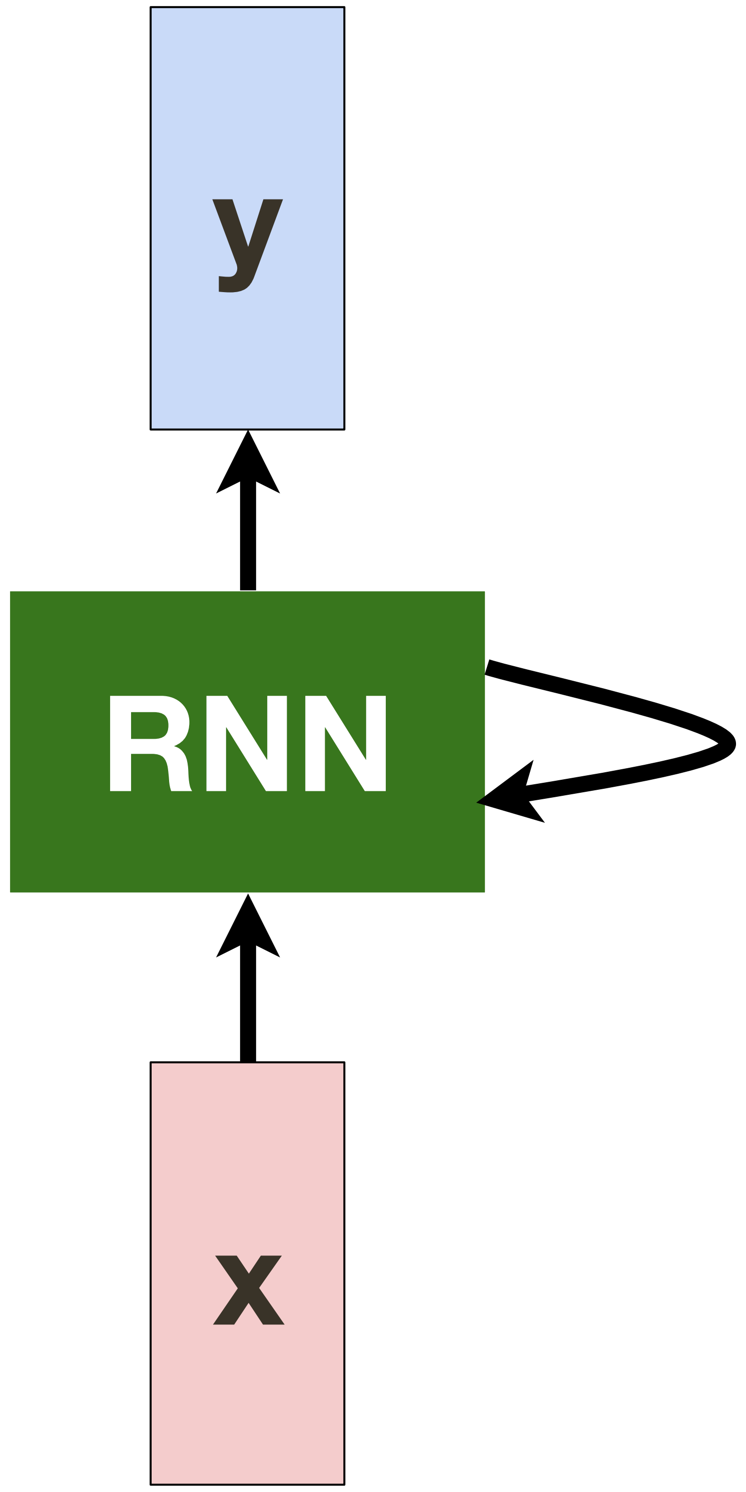
new **state**

old **state**

$$h_t = f_W(h_{t-1}, x_t)$$

input vector at some time step

some **function** with parameters W



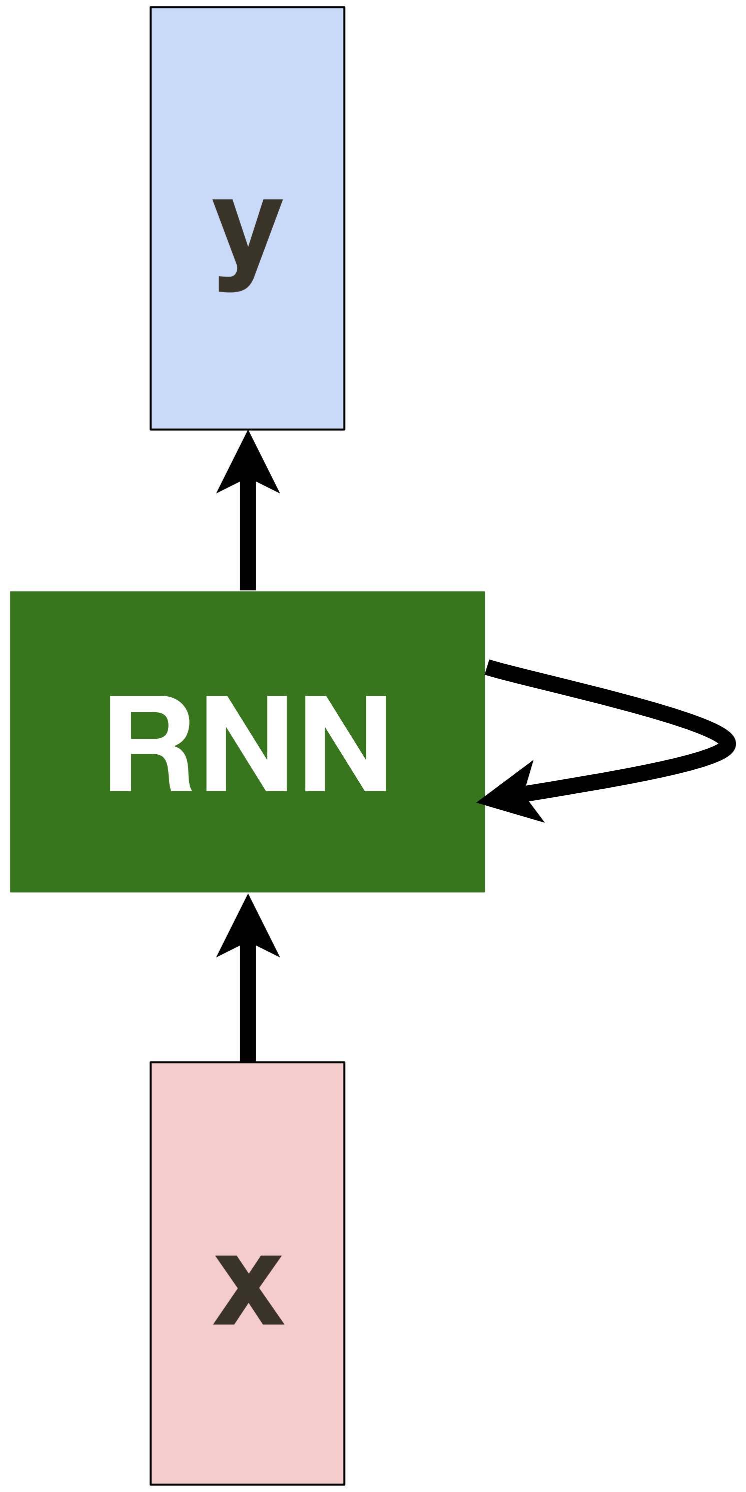
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

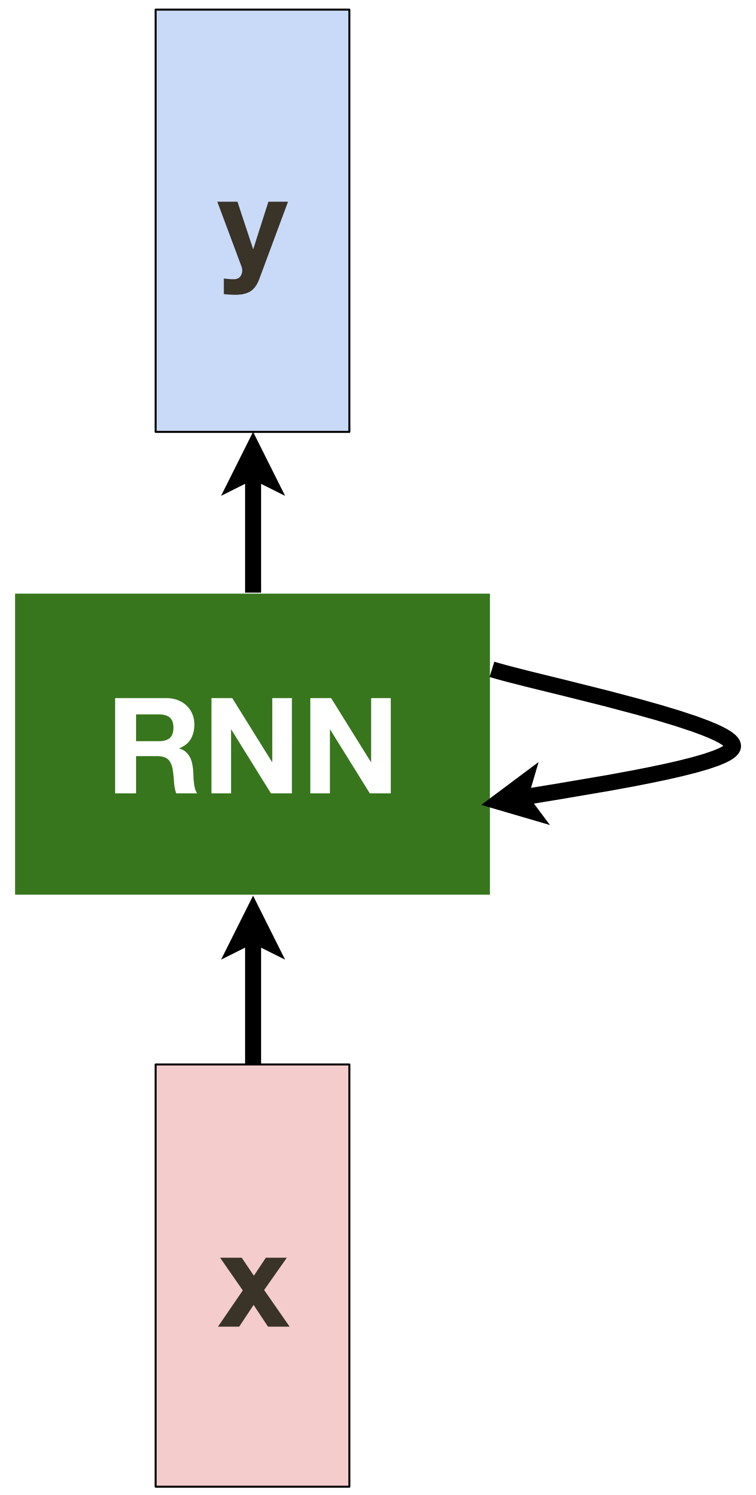
Note: the same function and the same set of parameters are used at every time step



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

(Vanilla) **Recurrent** Neural Network

$$h_t = f_W(h_{t-1}, x_t)$$



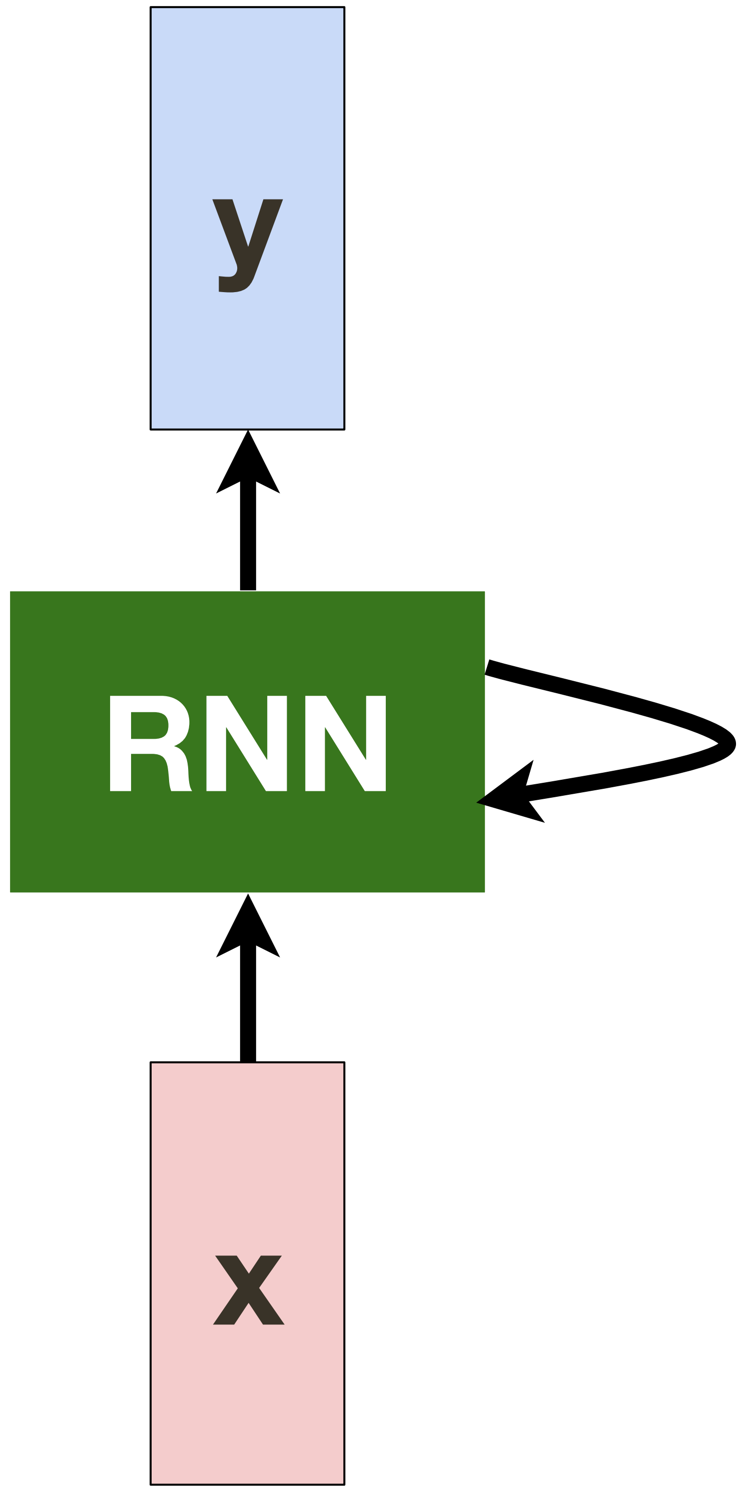
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

(Vanilla) **Recurrent** Neural Network

$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

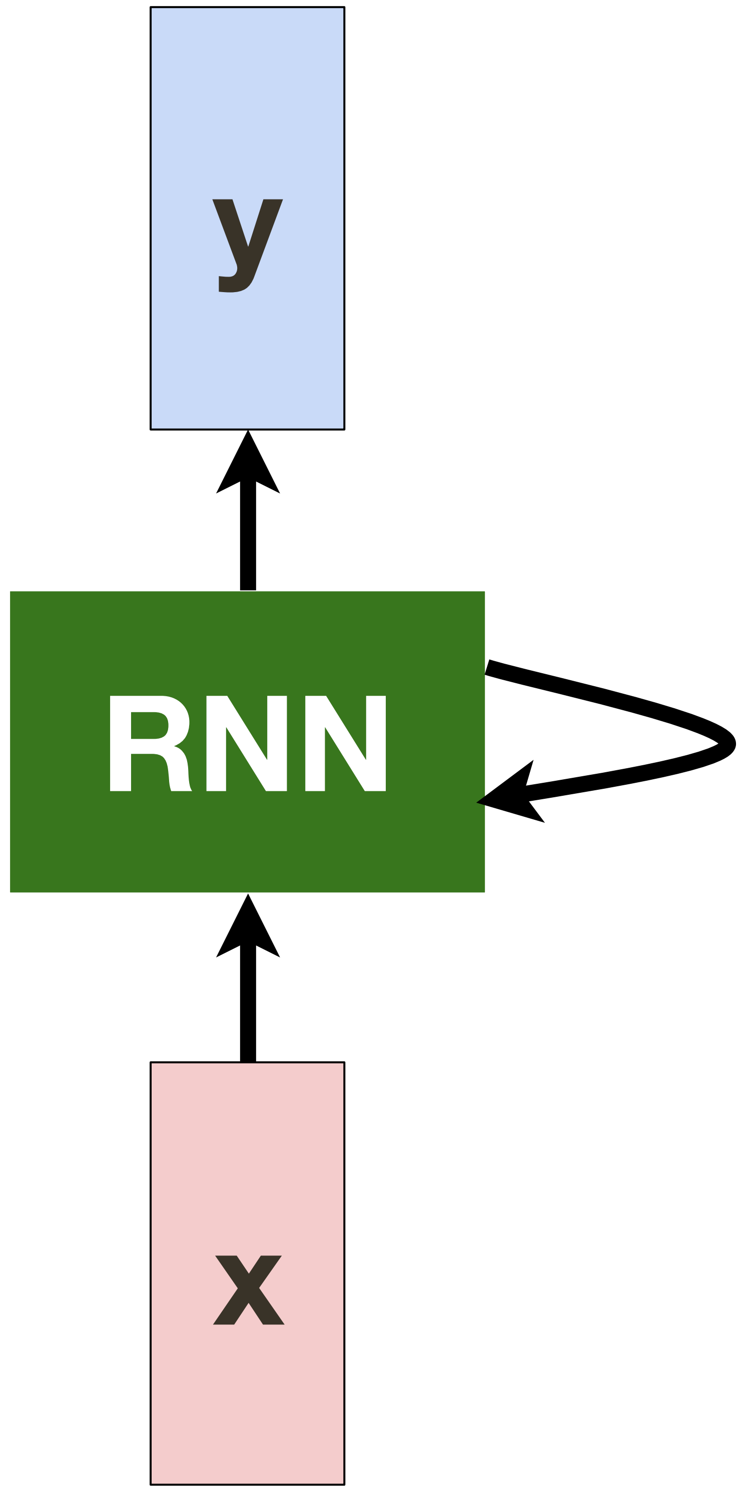


(Vanilla) **Recurrent** Neural Network

$$y_t = W_{hy} h_t + b_y$$

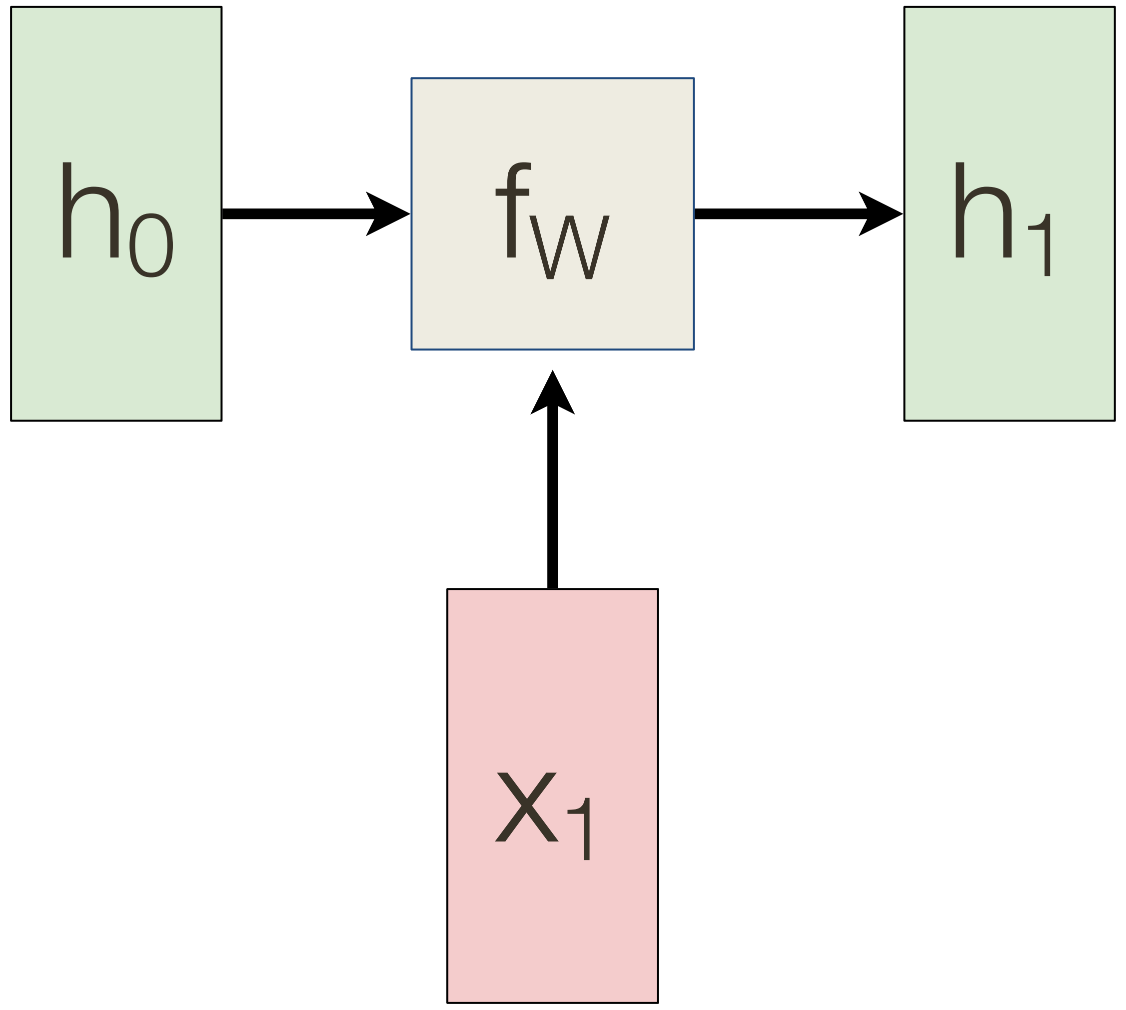
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$



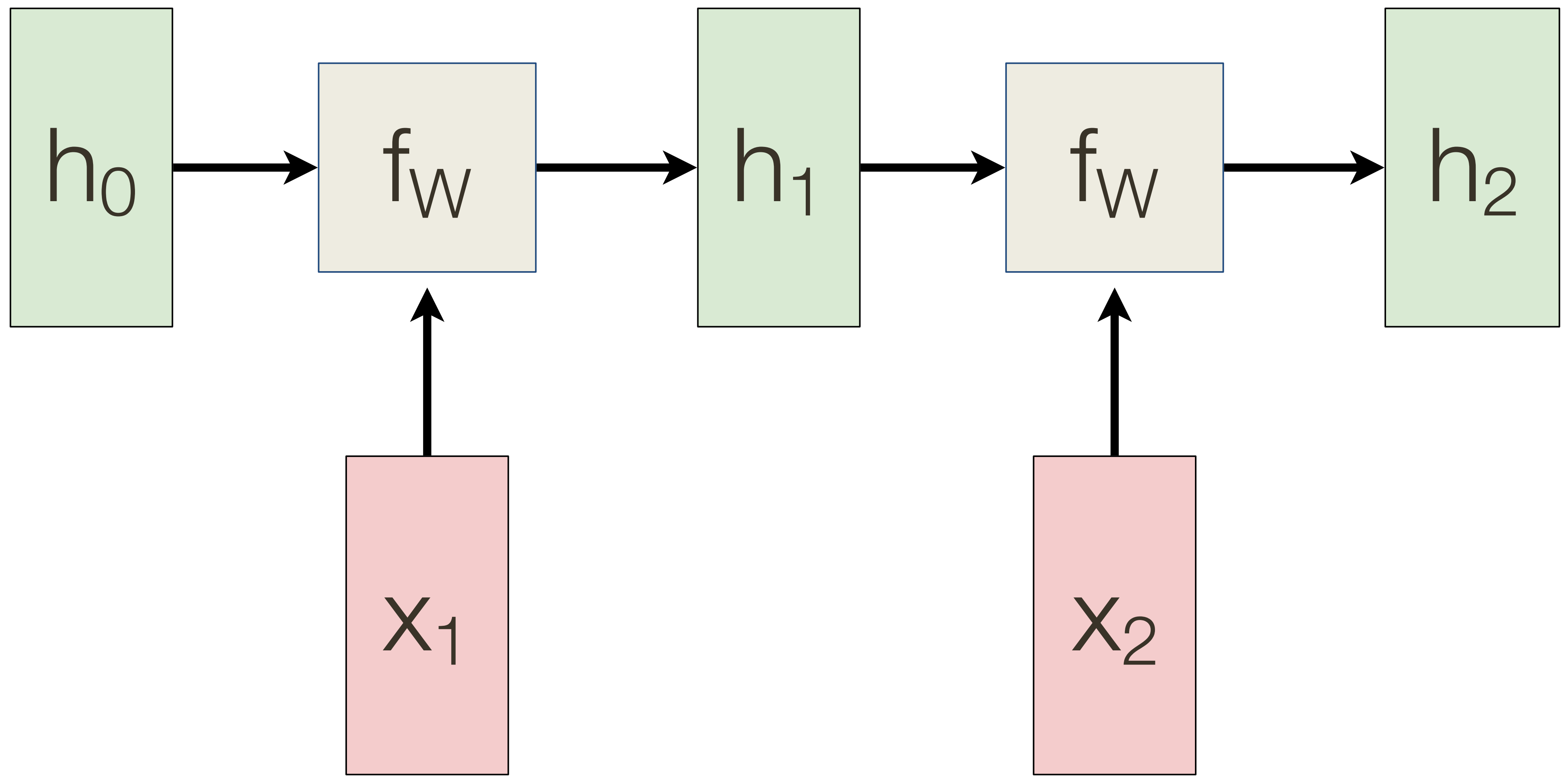
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**

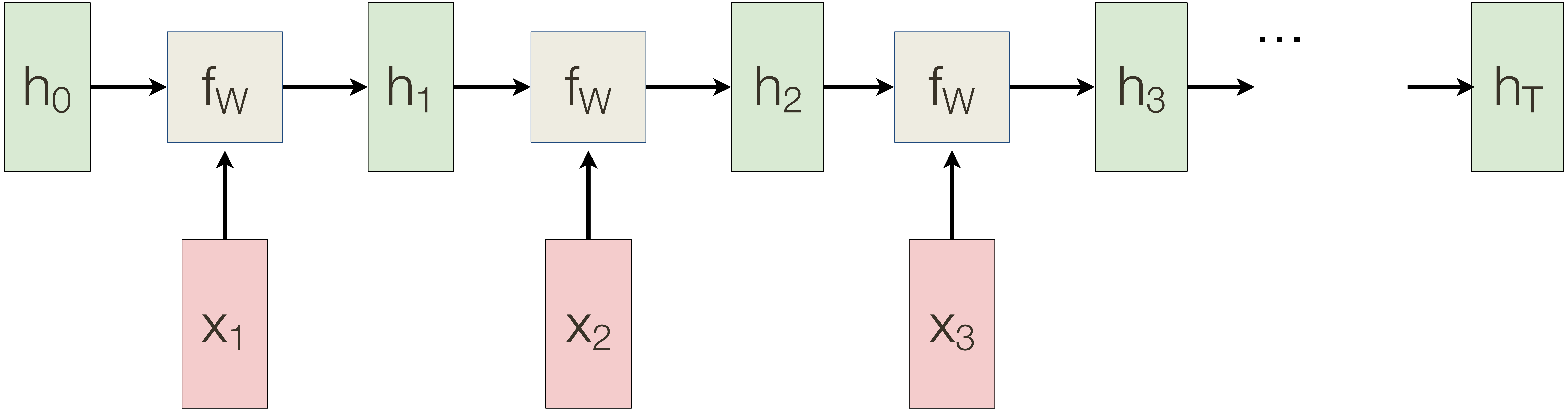


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**



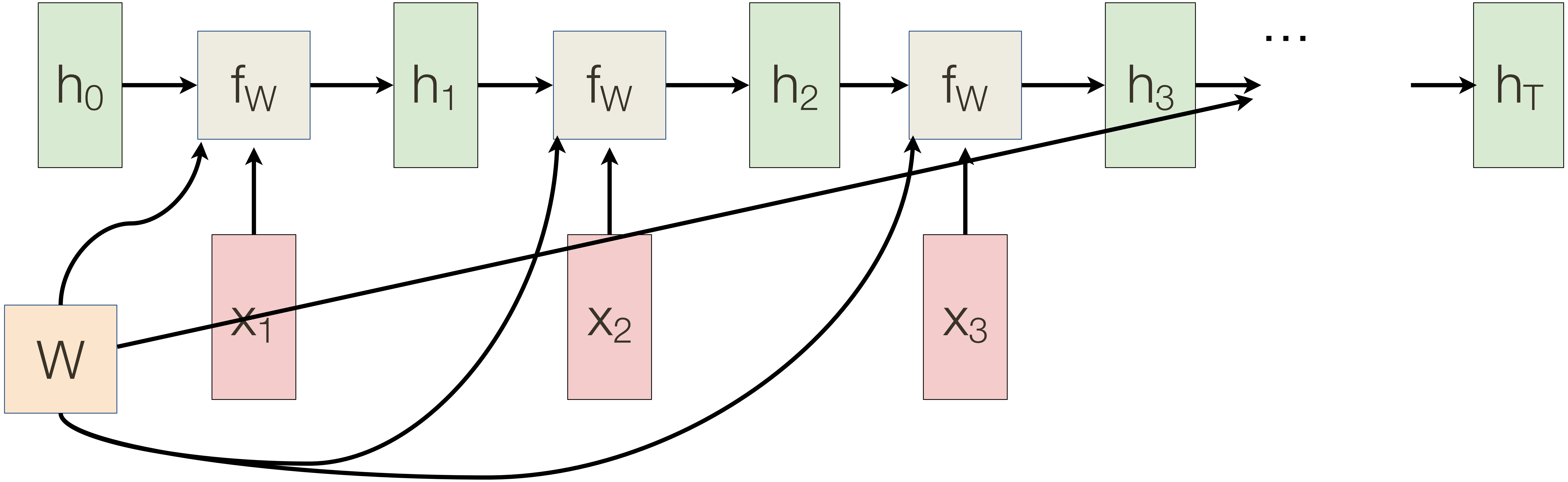
RNN **Computational Graph**



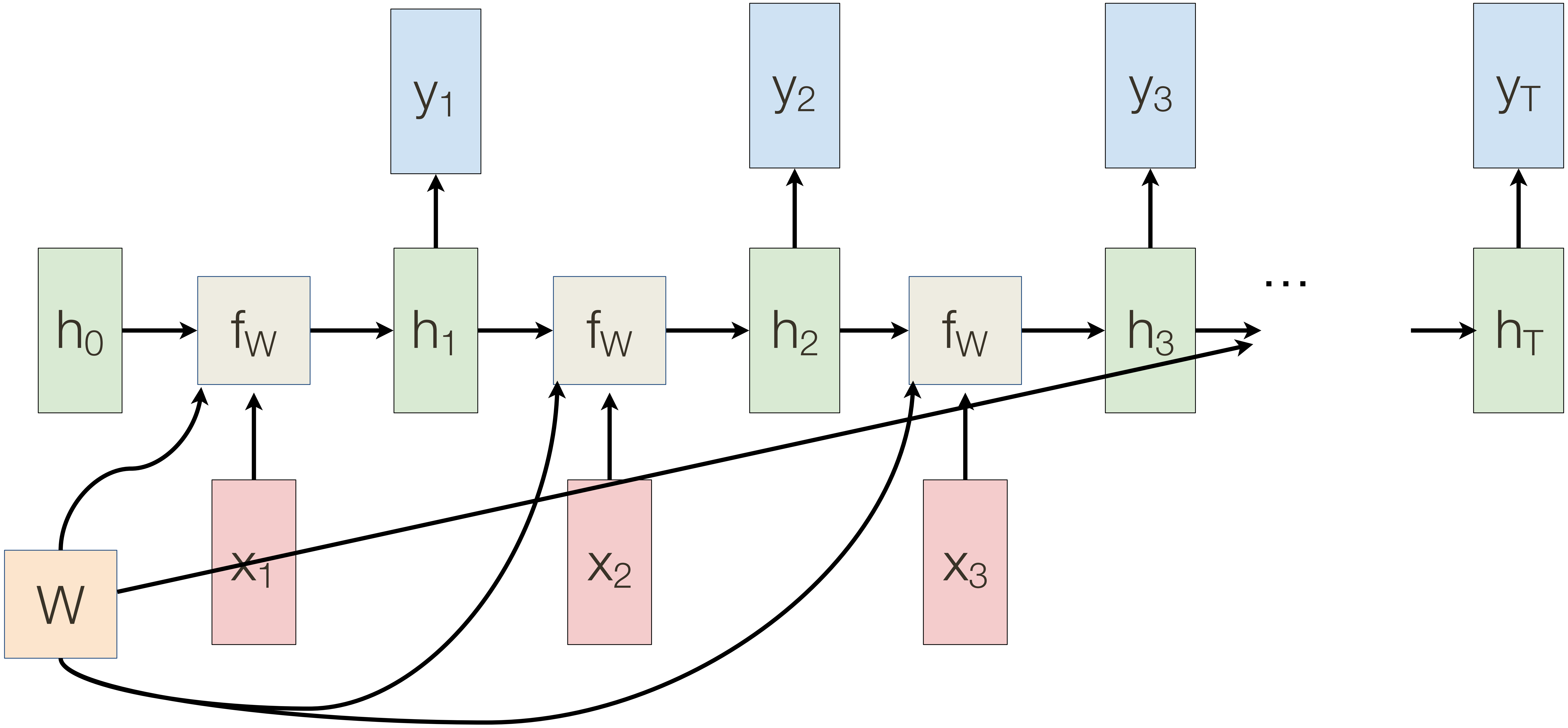
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**

Re-use the same weight matrix at every time-step

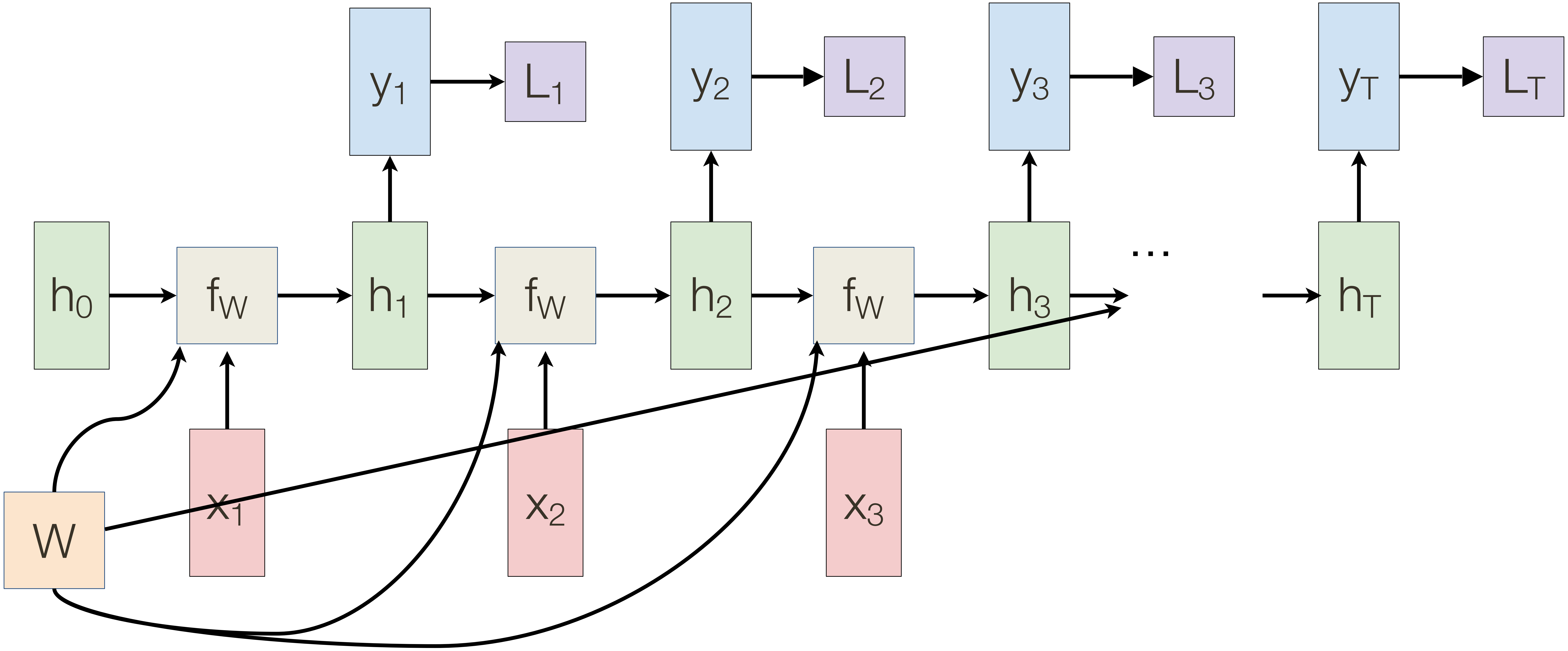


RNN **Computational Graph**: Many to Many



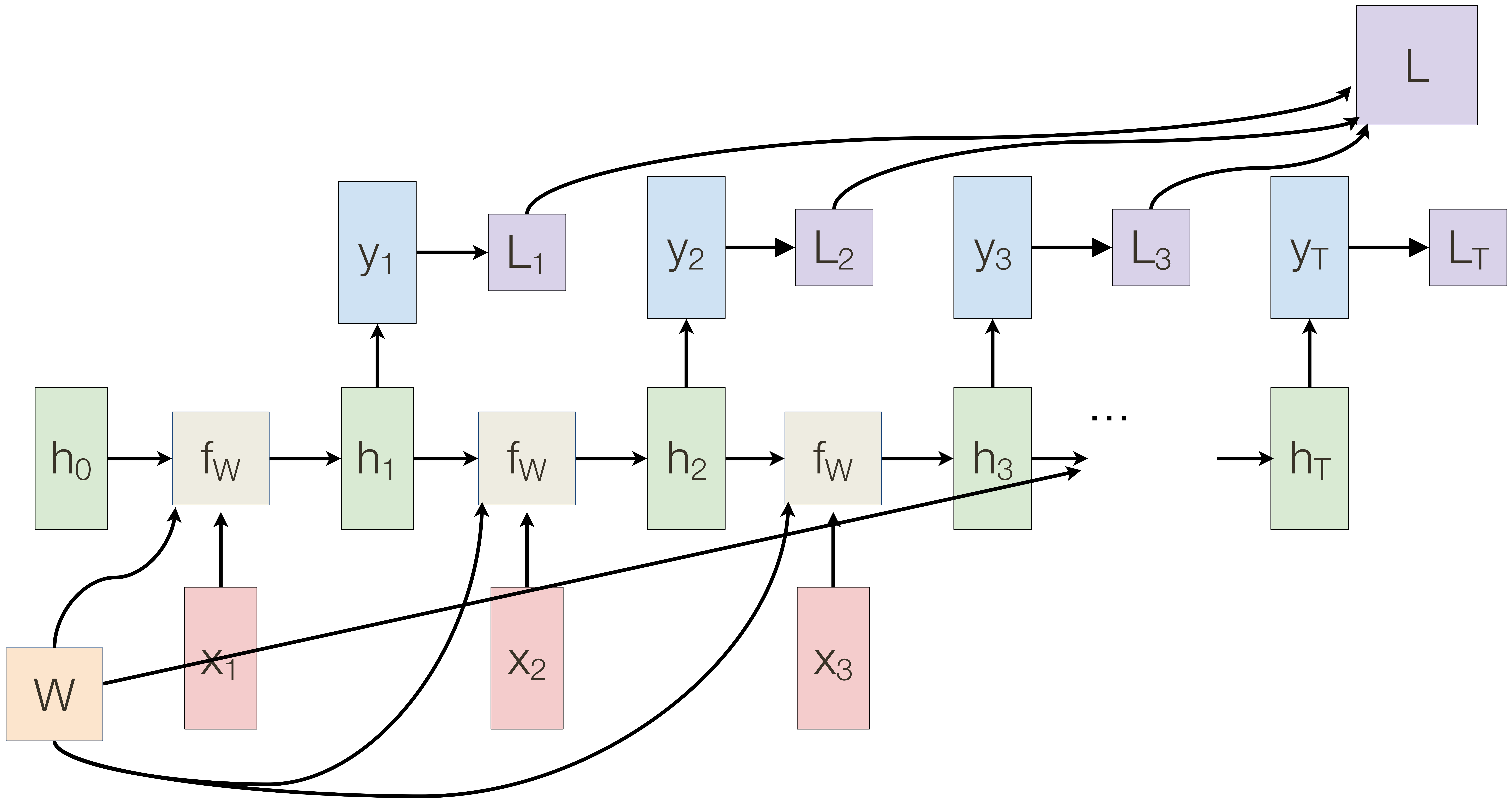
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**: Many to Many



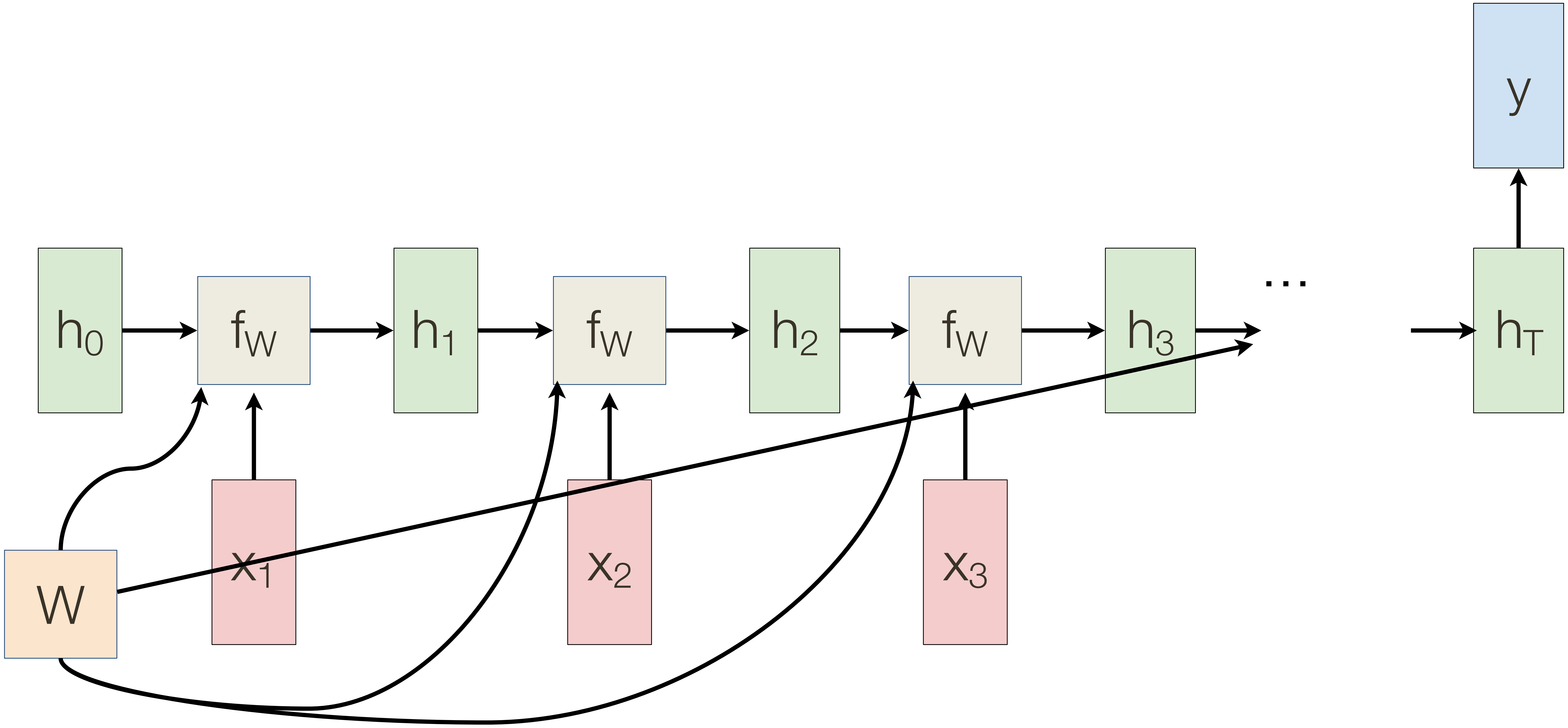
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**: Many to Many



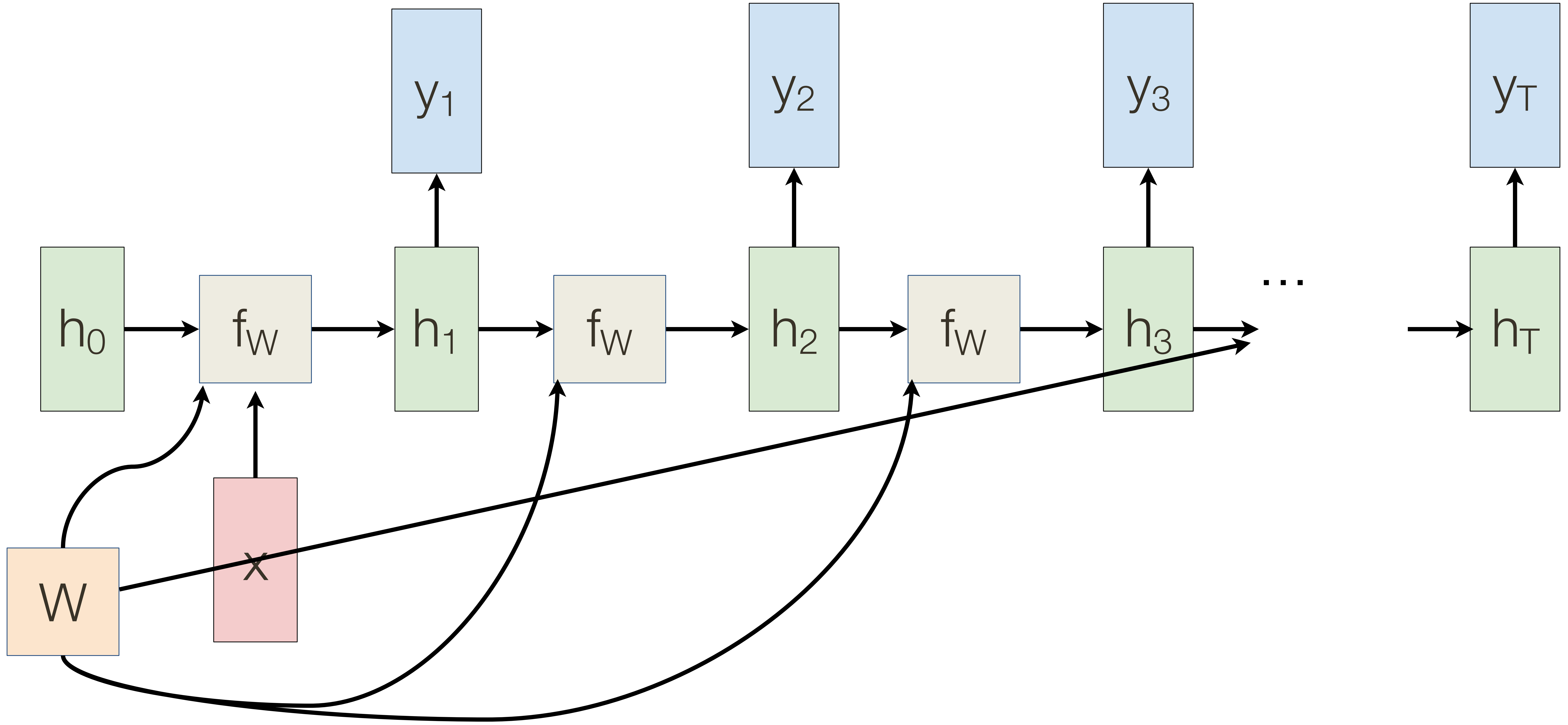
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**: Many to One



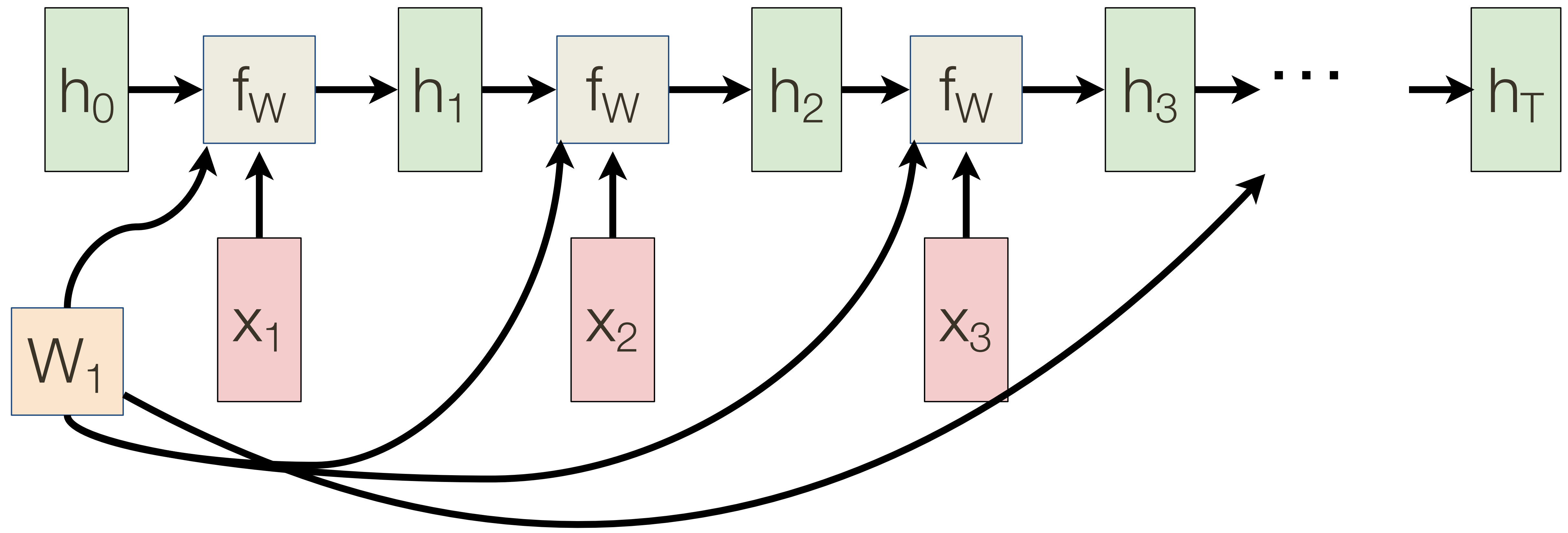
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

RNN **Computational Graph**: One to Many



Sequence to Sequence: Many to One + One to Many

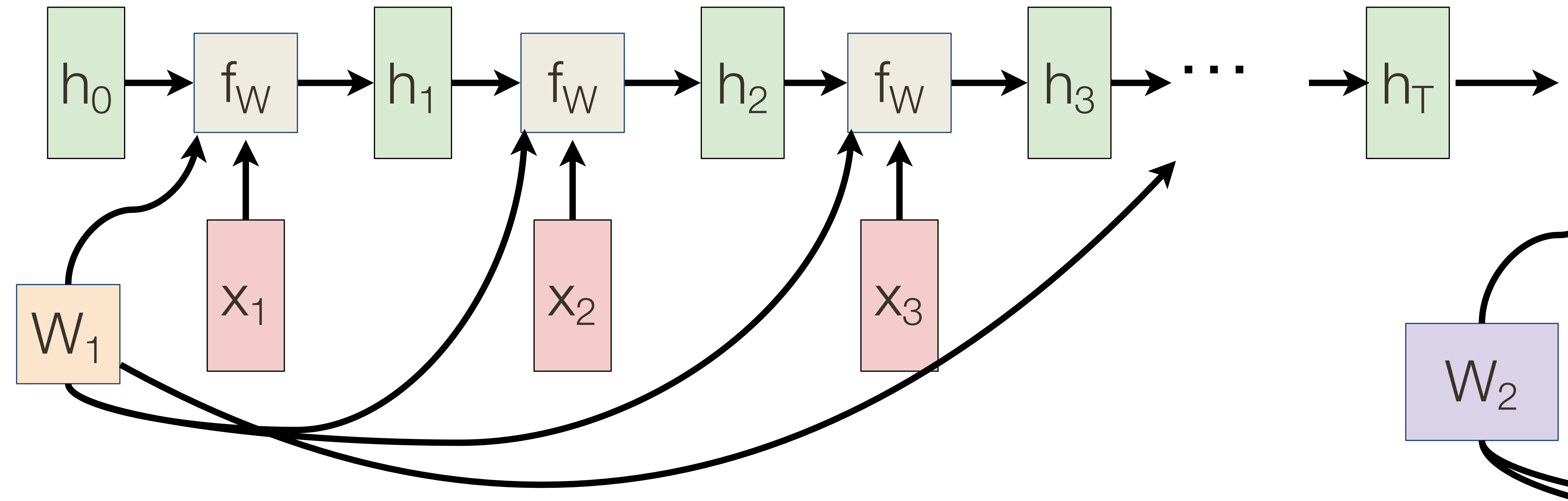
Many to one: Encode input sequence in a single vector



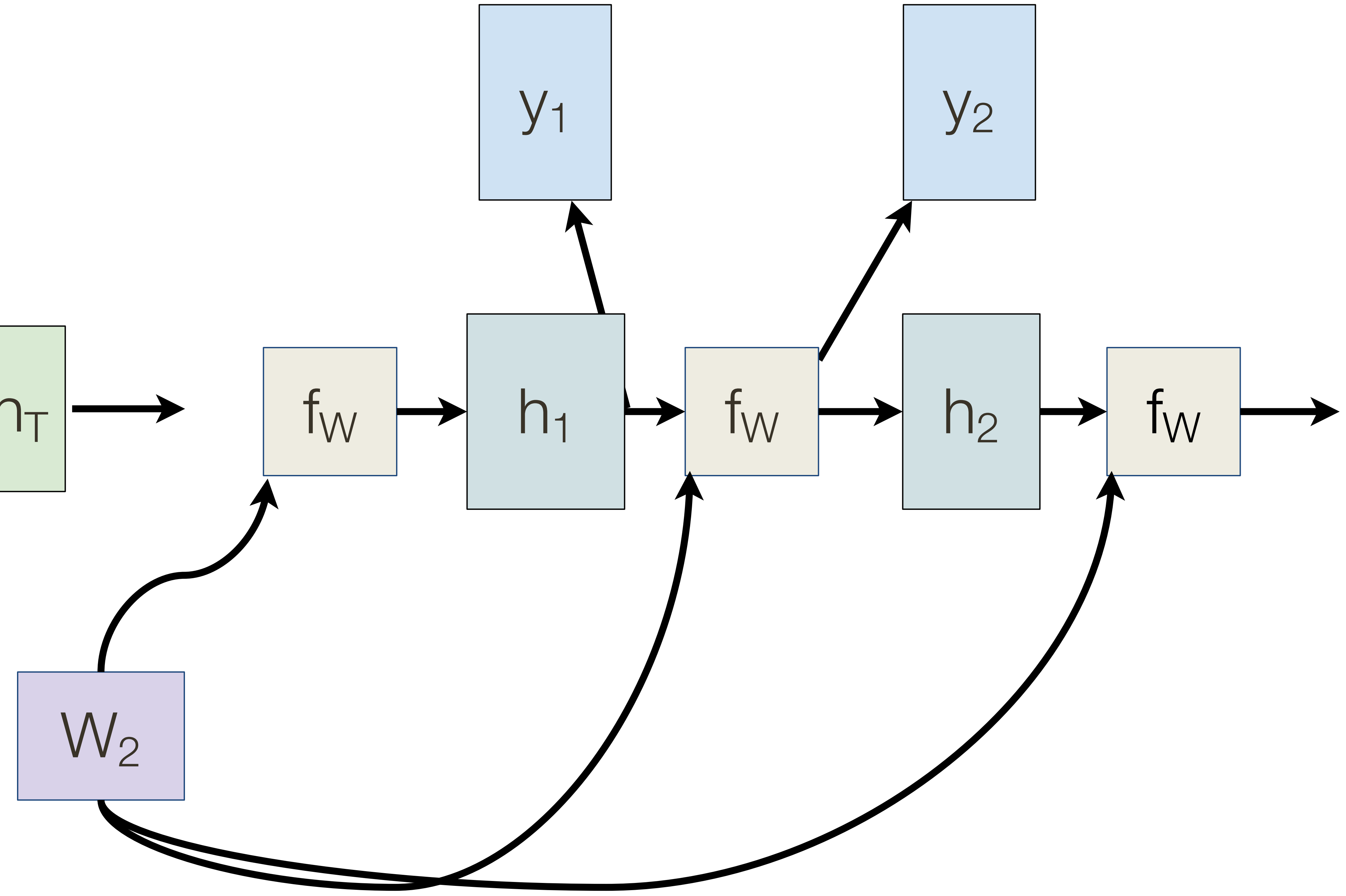
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Sequence to Sequence: Many to One + One to Many

Many to one: Encode input sequence in a single vector



One to many: Produce output sequence from single input vector



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**