

# Recurrent Neural Networks

모두의연구소  
박은수 Research Director

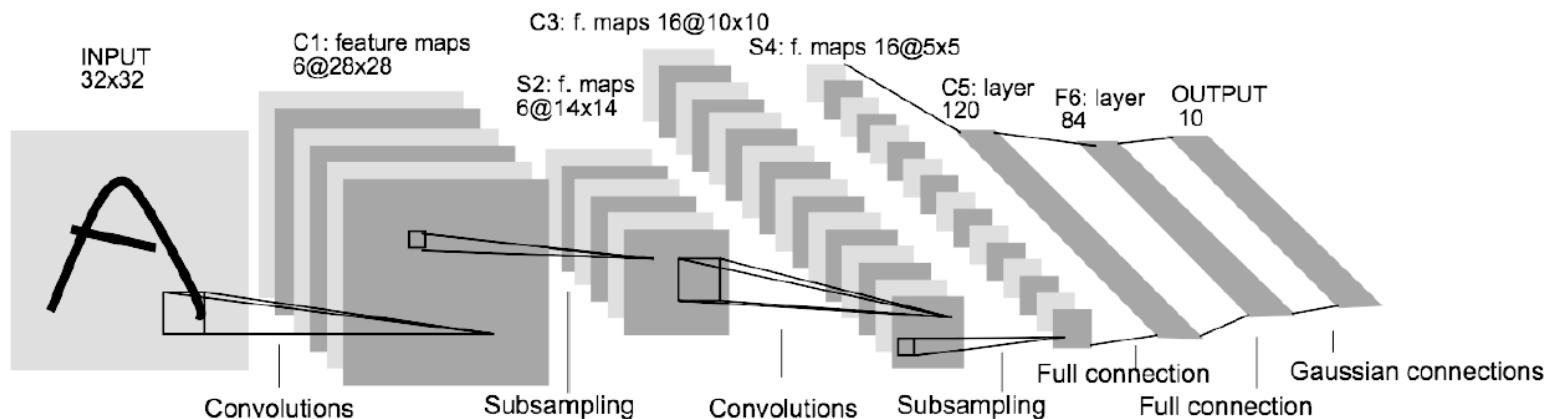
# 지난시간 돌아보기 ...



- **분류기의 구성**
  - Score function
  - Loss function
  - Optimization

# 지난시간 돌아보기 ...

## • Convolutional Neural Networks



# 지난시간 돌아보기 ...

## Momentum

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

## Gradient 이동누적 스러운 방법

- 업데이트 1)  $\mathbf{v}_1 \leftarrow \alpha * 0 - K_o : -K_o$
- 업데이트 2)  $\mathbf{v}_2 \leftarrow \alpha \mathbf{v}_1 - K_1 : -\alpha K_o - K_1$
- 업데이트 3)  $\mathbf{v}_3 \leftarrow \alpha \mathbf{v}_2 - K_2 : -\alpha^2 K_o - \alpha K_1 - K_2$
- 업데이트 4)  $\mathbf{v}_4 \leftarrow \alpha \mathbf{v}_3 - K_3 : -\alpha^3 K_o - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_1$
- 2)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_2$
- 3)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_3$
- 4)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_4$

## Adagrad

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1)  $\frac{1}{\sqrt{K_0^2}} K_o$

업데이트 2)  $\frac{1}{\sqrt{K_1^2 + K_0^2}} K_1$

업데이트 3)  $\frac{1}{\sqrt{K_3^2 + K_1^2 + K_0^2}} K_3$

업데이트 4)  $\frac{1}{\sqrt{K_4^2 + K_3^2 + K_1^2 + K_0^2}} K_4$

## Gradient Normalization 스러운 방법

## RMSprop

$$\mathbf{h} \leftarrow \alpha \mathbf{h} + (1 - \alpha) \left( \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}} \right)$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1)  $\mathbf{h}_1 = (1 - \alpha) \mathbf{K}_1^2$

업데이트 2)  $\mathbf{h}_2 = \alpha (1 - \alpha) \mathbf{K}_1^2 + (1 - \alpha) \mathbf{K}_2^2$

업데이트 3)  $\mathbf{h}_3 = \alpha^2 (1 - \alpha) \mathbf{K}_1^2 + \alpha (1 - \alpha) \mathbf{K}_2^2 + (1 - \alpha) \mathbf{K}_3^2$

업데이트 4)  $\mathbf{h}_4 = \alpha^3 (1 - \alpha) \mathbf{K}_1^2 + \alpha^2 (1 - \alpha) \mathbf{K}_2^2 + \alpha (1 - \alpha) \mathbf{K}_3^2 + (1 - \alpha) \mathbf{K}_4^2$

# 지난시간 돌아보기 ...

## Momentum

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

- 업데이트 1)  $\mathbf{v}_1 \leftarrow \alpha * 0 - K_0 : -K_0$
- 업데이트 2)  $\mathbf{v}_2 \leftarrow \alpha \mathbf{v}_1 - K_1 : -\alpha K_0 - K_1$
- 업데이트 3)  $\mathbf{v}_3 \leftarrow \alpha \mathbf{v}_2 - K_2 : -\alpha^2 K_0 - \alpha K_1 - K_2$
- 업데이트 4)  $\mathbf{v}_4 \leftarrow \alpha \mathbf{v}_3 - K_3 : -\alpha^3 K_0 - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_1$
- 2)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_2$
- 3)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_3$
- 4)  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}_4$

## Adagrad

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

- 업데이트 1)  $\frac{1}{\sqrt{K_0}} K_0$
- 업데이트 2)  $\frac{1}{\sqrt{K_1^2 + K_0^2}} K_1$
- 업데이트 3)  $\frac{1}{\sqrt{K_3^2 + K_1^2 + K_0^2}} K_3$
- 업데이트 4)  $\frac{1}{\sqrt{K_4^2 + K_3^2 + K_1^2 + K_0^2}} K_4$

두 방법의 같이쓰자  
Adam

## RMSprop

$$\mathbf{h} \leftarrow \alpha \mathbf{h} + (1 - \alpha) \left( \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}} \right)$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

- 업데이트 1)  $\mathbf{h}_1 = (1 - a) \mathbf{K}_1^2$
- 업데이트 2)  $\mathbf{h}_2 = a(1 - a) \mathbf{K}_1^2 + (1 - a) \mathbf{K}_2^2$
- 업데이트 3)  $\mathbf{h}_3 = a^2(1 - a) \mathbf{K}_1^2 + a(1 - a) \mathbf{K}_2^2 + (1 - a) \mathbf{K}_3^2$
- 업데이트 4)  $\mathbf{h}_4 = a^3(1 - a) \mathbf{K}_1^2 + a^2(1 - a) \mathbf{K}_2^2 + a(1 - a) \mathbf{K}_3^2 + (1 - a) \mathbf{K}_4^2$

# 지난시간 돌아보기 ...

## Batch Normalization

[Ioffe and Szegedy, 2015]

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots m\}$ ;  
 Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

**Note:** at test time BatchNorm layer functions differently:

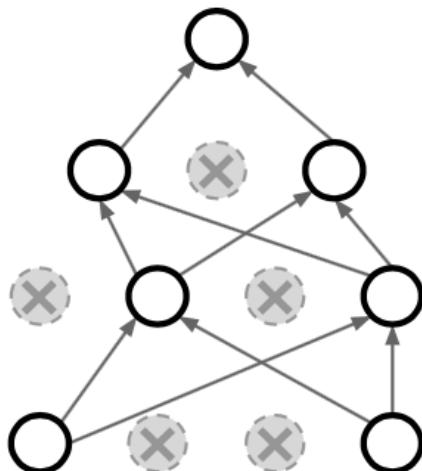
The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used.

(e.g. can be estimated during training with running averages)

# 지난시간 돌아보기 ...

## Regularization: Dropout

How can this possibly be a good idea?



Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

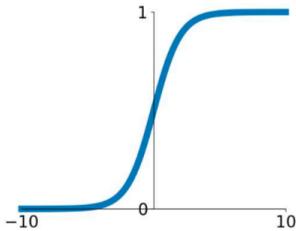
An FC layer with 4096 units has  $2^{4096} \sim 10^{1233}$  possible masks!

Only  $\sim 10^{82}$  atoms in the universe...

# Activation Functions

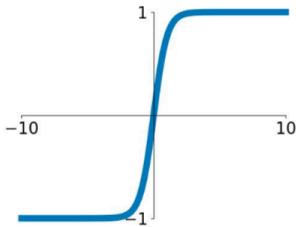
## Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



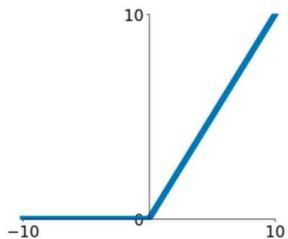
## tanh

$$\tanh(x)$$



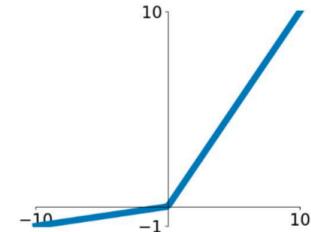
## ReLU

$$\max(0, x)$$



## Leaky ReLU

$$\max(0.1x, x)$$

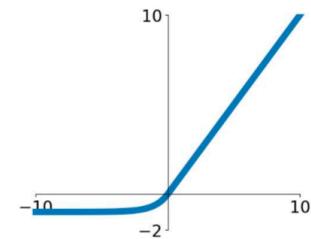


## Maxout

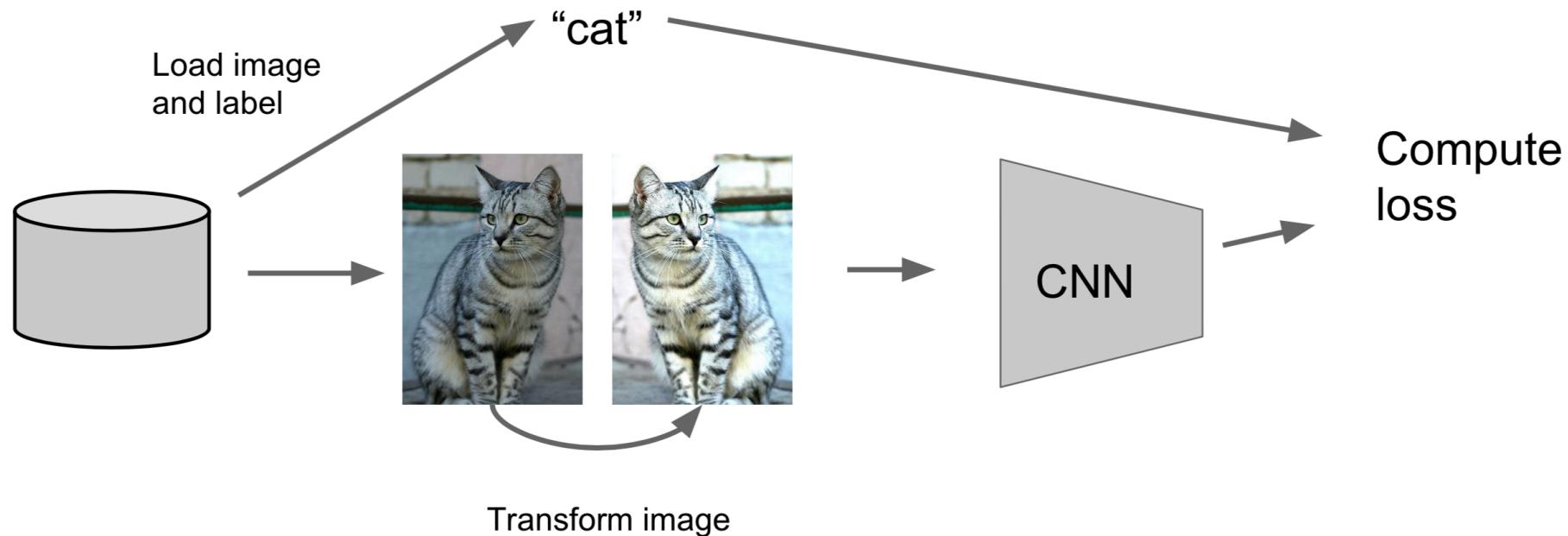
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

## ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



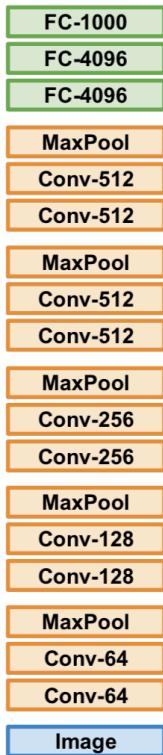
# Regularization: Data Augmentation



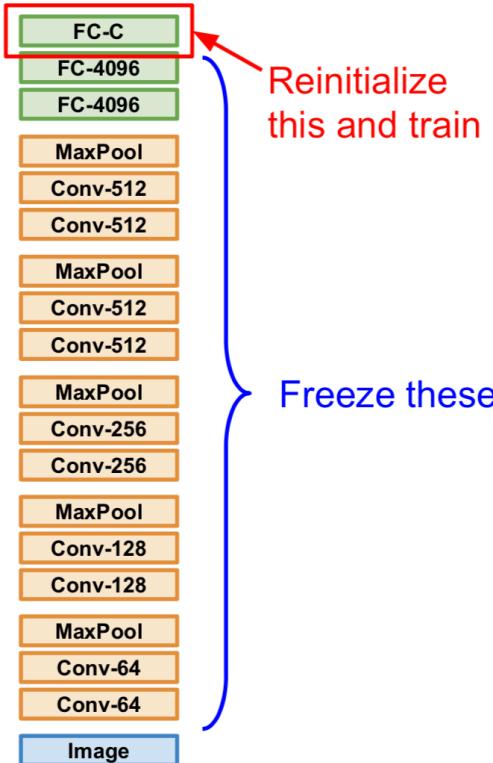
# Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014  
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

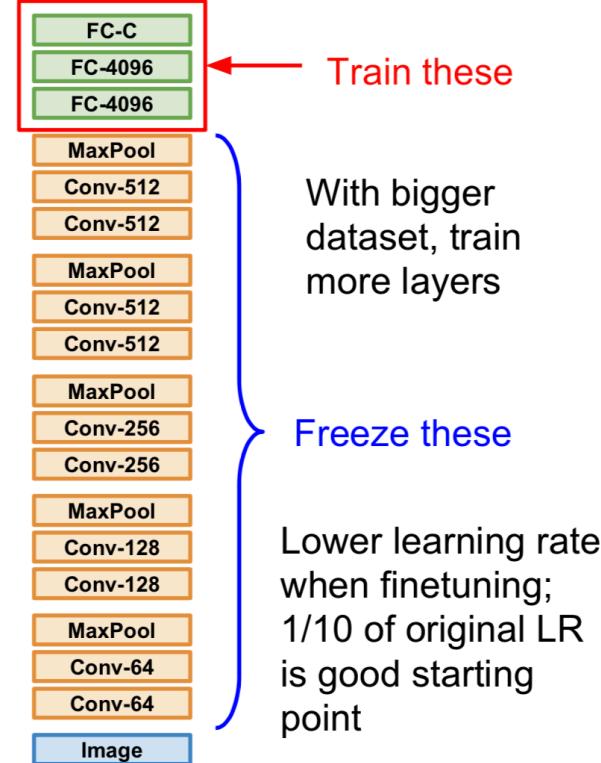
## 1. Train on Imagenet



## 2. Small Dataset (C classes)



## 3. Bigger dataset



가 나 다 라 마 ( ) ( )

가 나 다 라 마 사 아

A B C D E F ( ) ( ) ( ) ~~~

가 나 다 라 마 사 아

A B C D E F G H I ~~

가 나 다 라 마 사 아

A B C D E F G H I ~~

# 거꾸로~ 로꾸꺼~



하 파 타 카 차 자 ( ) ( ) ~~



# 거꾸로~ 로꾸꺼~



하 파 타 카 차 자 (아) (사) ~~



# 거꾸로~ 로꾸꺼~



하 파 타 카 차 자 (아) (사) ~~

Q P O N M L K J ( ) ( ) ( )~~



# 거꾸로~ 로꾸꺼~



하 파 타 카 차 자 (아) (사) ~~

Q P O N M L K J (I) (H) (G)~~



# 거꾸로~ 로꾸꺼~

## 로꾸꺼 가사 ...

(모두) 로꾸거 로꾸거 로꾸거 말해말  
로꾸거 로꾸거 로꾸거 말해말

(희철) 아 많다 많다 많다 많다  
다 이쁜 이쁜 이쁜이다  
여보게 저기 저게보여

(신동) 여보 안경 안보여

(강인) 통술집 술통 소주 만병만 주소  
다 이심전삼이다 뻑뻑뻑  
아 좋다좋아 수박이 박수

(희철) 다시 합창합시다

(모두) 로꾸거 로꾸거 로꾸거 말해말  
로꾸거 로꾸거 로꾸거 말해말

(이특) 니 가는데는 가니 일요일 수이수  
수리수리수 몰랑몰랑몰  
아 좋다좋아 수박이 박수  
다시 합창 합시다

(성민) 어제는 거꾸로 오늘도 거꾸로  
모두가 거꾸로 돌아가고 있어  
내일이 와야해 행복의 시계가  
째깍째깍 돌아가겠지

(모두) 째깍 째깍 째깍  
원투쓰리포파이브식스 고  
로꾸거 로꾸거 로꾸거 말해말  
로꾸거 로꾸거 로꾸거 말해말

(은혁) 하파타카차자아사바마라다나가

(신동) 십구팔칠육오사삼이일땡.

(은혁) 아래서 위로 뒤에서 앞으로

(신동) 모든걸 거꾸로 로꾸거

(신동&은혁) 할아버지 할머니 아저씨  
아줌마 남녀노소 짬짬짜  
얼씨구 절씨구 빠라빠라 뺄뺀  
모든걸 거꾸로 로꾸거

(이특) 나갔다오나 나오다갔나  
아들 딸이 다컸다 이 딸들아

(성민) 다 같은 별은 별은 같다

(은혁) 자꾸만 꿈만 꾸자

(신동) 장가간 가장 시집간 집시  
다 된 장국 청국장된다  
아 좋다좋아 수박이 박수  
다시 합창 합시다

(희철) 어제도 거꾸로 오늘도 거꾸로  
모두가 거꾸로 돌아가고 있어  
내일이 와야해 행복의 시계가  
째깍째깍 돌아가겠지

(모두) 째깍 째깍 째깍  
원투쓰리포파이브식스 고  
로꾸거 로꾸거 로꾸거 말해말  
로꾸거 로꾸거 로꾸거 말해말  
아 좋다 좋아 수박이 박수  
다시 합창 합시다  
로꾸거 로꾸거 로꾸거 로꾸거  
로꾸거 로꾸거 로꾸거 말해말



# 비슷한 사례



좋아하는 노래의 후렴구 바로 전을 떠 올려 보세요

# 비슷한 사례



좋아하는 노래의 후렴구 바로 전을 떠 올려 보세요

혹시 처음부터 시작해서 후렴구 바로 전을 찾지 않으셨나요?

# 왜 그럴까요?



하 파 타 카 차 자 (아) (사) ~~

Q P O N M L K J (I) (H) (G)~~

혹시 처음부터 시작해서 후렴구 바로 전을 찾지 않으셨나요?

# 왜 그럴까요?

하 파 타 카 차 자 (아) (사) ~~

Q P O N M L K J (I) (H) (G)~~

혹시 처음부터 시작해서 후렴구 바로 전을 찾지 않으셨나요?

인간은 기억할때 컴퓨터처럼 하드드라이브에 저장하지 않기 때문입니다



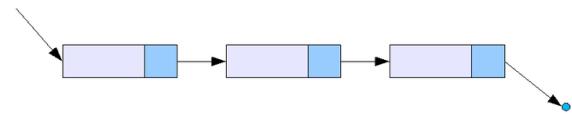
# 왜 그럴까요?

인간은 기억할때 컴퓨터처럼 하드드라이브에 저장하지 않기 때문입니다

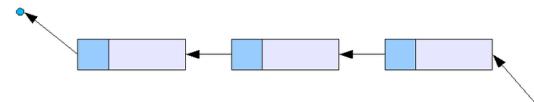


인간은 정보를 Sequence로 학습한다고 합니다

하 파 타 카 차 자 (아) (사) ~~



Q P O N M L K J (I) (H) (G)~~



# 왜 그럴까요?



그렇다고 우리가 A, B, C를 못 외우거나 노래 가사를 잊어 버리고 있는건 아니잖아요?

# 왜 그럴까요?



그렇다고 우리가 A, B, C를 못 외우거나 노래 가사를 잊어 버리고 있는건 아니잖아요?

넵, 단지 더 많은 시간이 걸리는 겁니다.

이런식으로 찾으려 시도한적이 없기 때문이지요  
뇌에서 이 정보가 위치하는 곳으로 이어지는 어떤 지도가 딱  
존재하지 않는 것 입니다

참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# 왜 그럴까요?

다니던 길을 찾아 가는  
것은 쉽게 상상할 수 있  
습니다



참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# 왜 그럴까요?

다니던 길이 아닌 직선  
의 길을 상상해서 가는  
것은 쉽지 않죠



참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

자주 사용하는 동작은 무의식적으로 발현됨을 기억해 보세요~

# 왜 그럴까요?



우리가 사고하는 방향 혹은 Sequence를 조건부 확률처럼  
뉴런이 학습되게 됩니다

또한 이전에 봐왔던 패턴 혹은 기억이 우리의 판단결과에 영향을 미치게 됩니다

참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# 왜 그럴까요?



우리가 사고하는 방향 혹은 Sequence를 조건부 확률처럼  
뉴런이 학습되게 됩니다

또한 이전에 봐왔던 패턴 혹은 기억이 우리의 판단결과에 영향을 미치게 됩니다

참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# Sequence 형태의 데이터를 뉴럴 네트워크가 학습할 수 있게 하는것?



?

참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# Sequence 형태의 데이터를 뉴럴 네트워크가 학습할 수 있게 하는것?



## Recurrent Neural Networks

참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.  
<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# 뉴럴 네트워크가 이전 정보를 반영하는 방법



일반적인 NN :      input -> hidden -> output

이전 정보를 반영하는 2가지 방법론

(input + prev\_hidden)->hidden->output

Previous hidden

VS.

(input + prev\_input)->hidden->output

Previous input

참고 : Anyone Can Learn To Code an LSTM-RNN in Python (한글번역), 유재준.

<http://jaejunyoo.blogspot.com/2017/06/anyone-can-learn-to-code-LSTM-RNN-Python.html>

# Previous Hidden vs. Previous Input

Previous hidden

**(input + empty\_hidden) -> hidden -> output**  
**(input + prev\_hidden) -> hidden -> output**

Previous input

**(input + empty\_input) -> hidden -> output**  
**(input + prev\_input) -> hidden -> output**

# Previous Hidden vs. Previous Input

Previous hidden

(**input** + **empty\_hidden**) -> **hidden** -> **output**  
(**input** + **prev\_hidden**) -> **hidden** -> **output**  
(**input** + **prev\_hidden**) -> **hidden** -> **output**

Previous input

(**input** + **empty\_input**) -> **hidden** -> **output**  
(**input** + **prev\_input**) -> **hidden** -> **output**  
(**input** + **prev\_input**) -> **hidden** -> **output**

# Previous Hidden vs. Previous Input

Previous hidden

(**input** + **empty\_hidden**) -> **hidden** -> **output**  
(**input** + **prev\_hidden**) -> **hidden** -> **output**  
(**input** + **prev\_hidden**) -> **hidden** -> **output**  
(**input** + **prev\_hidden**) -> **hidden** -> **output**

Previous input

(**input** + **empty\_input**) -> **hidden** -> **output**  
(**input** + **prev\_input**) -> **hidden** -> **output**  
(**input** + **prev\_input**) -> **hidden** -> **output**  
(**input** + **prev\_input**) -> **hidden** -> **output**

# Previous Hidden vs. Previous Input

4단계 과정비교

Previous hidden

전부다 기억할 수 있음

(**input** + **empty\_hidden**) -> **hidden** -> **output**

(**input** + **prev\_hidden**) -> **hidden** -> **output**

(**input** + **prev\_hidden**) -> **hidden** -> **output**

(**input** + **prev\_hidden**) -> **hidden** -> **output**

Previous input

바로 직전만을 기억함

(**input** + **empty\_input**) -> **hidden** -> **output**

(**input** + **prev\_input**) -> **hidden** -> **output**

(**input** + **prev\_input**) -> **hidden** -> **output**

(**input** + **prev\_input**) -> **hidden** -> **output**

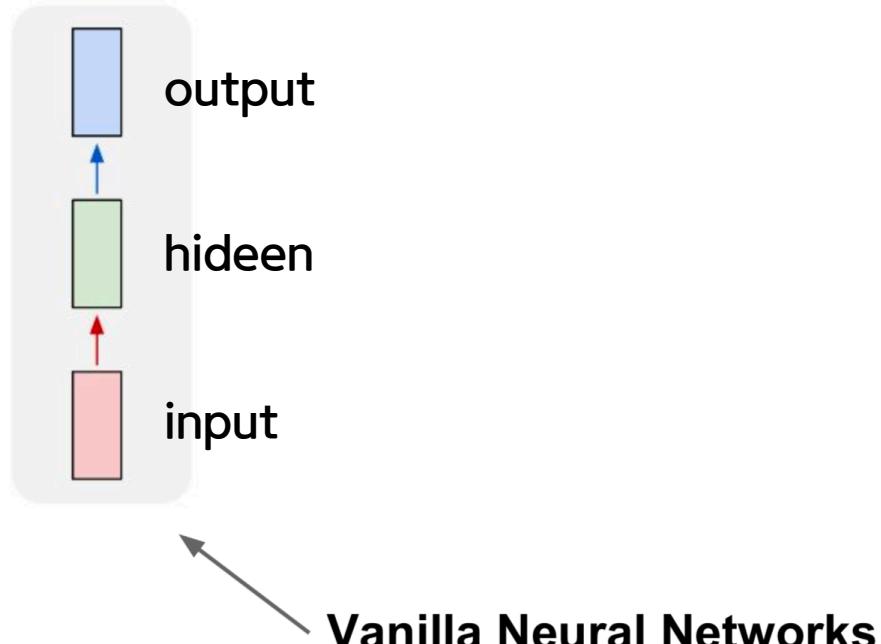
# Previous Hidden vs. Previous Input



Hidden과 input의 조합으로 네트워크를 구성해야 긴 sequence를 다룰 수 있겠군요

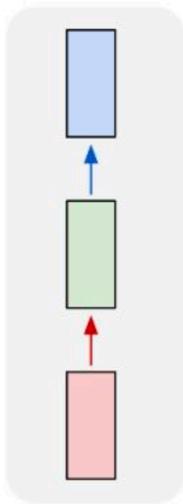
# “Vanilla” Neural Network

one to one

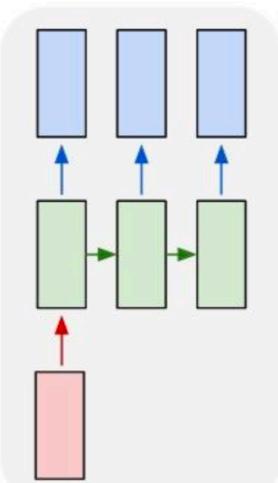


# Recurrent Neural Networks: Process Sequences

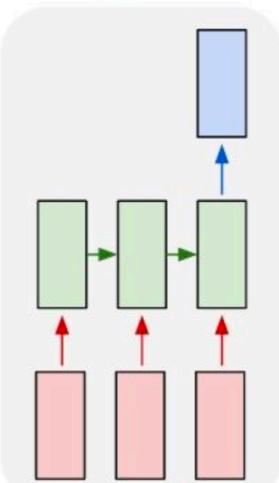
one to one



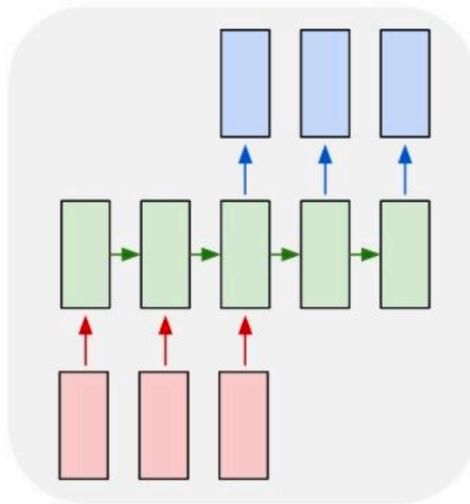
one to many



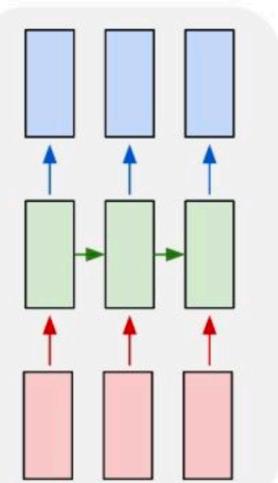
many to one



many to many



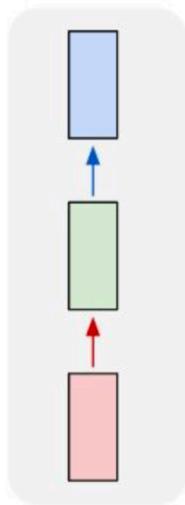
many to many



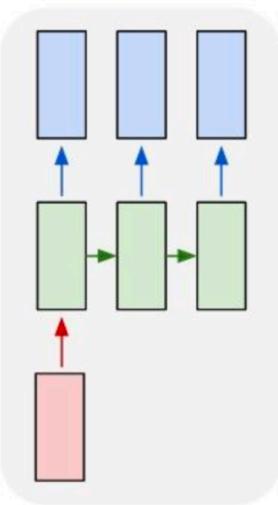
→ e.g. **Image Captioning**  
image -> sequence of words

# Recurrent Neural Networks: Process Sequences

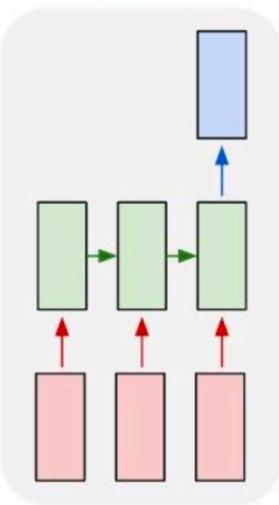
one to one



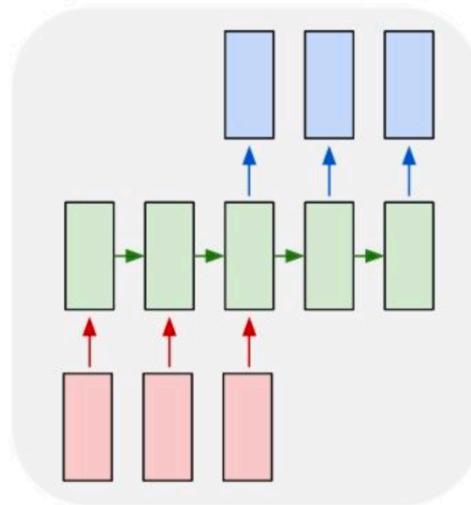
one to many



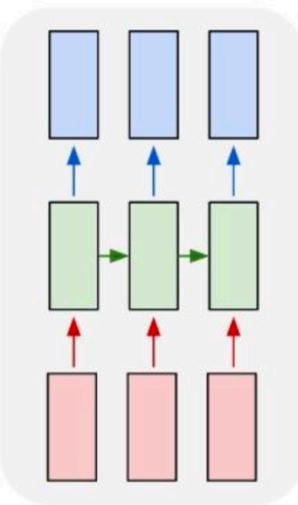
many to one



many to many



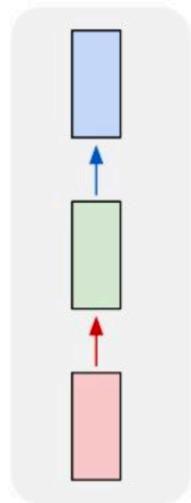
many to many



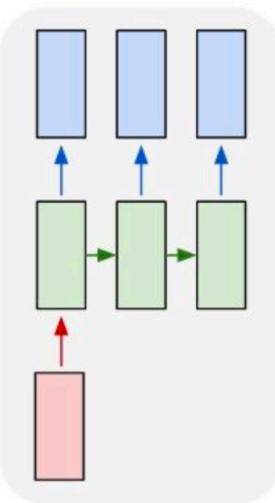
e.g. **Sentiment Classification**  
sequence of words -> sentiment

# Recurrent Neural Networks: Process Sequences

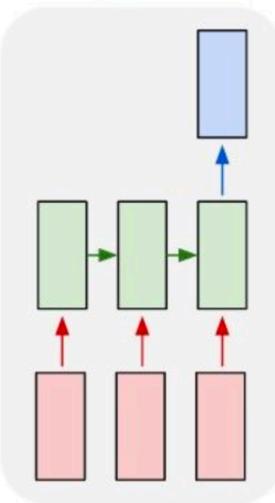
one to one



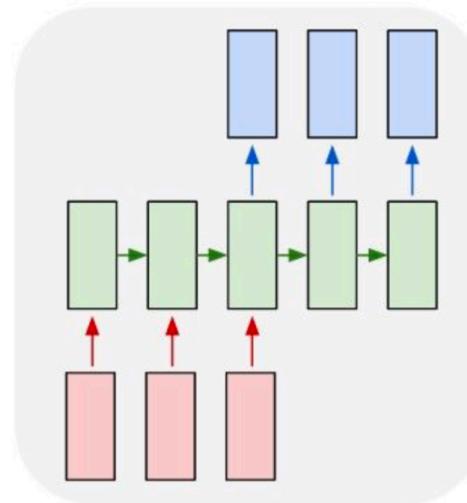
one to many



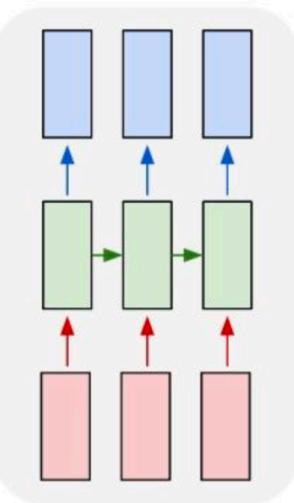
many to one



many to many



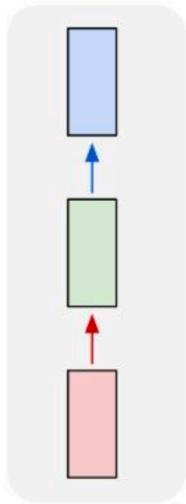
many to many



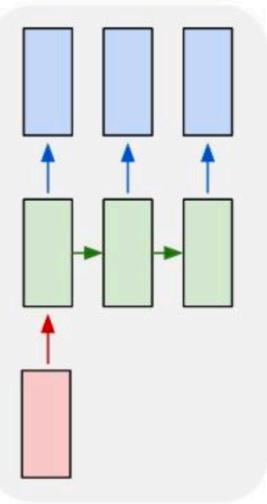
↑  
e.g. **Machine Translation**  
seq of words -> seq of words

# Recurrent Neural Networks: Process Sequences

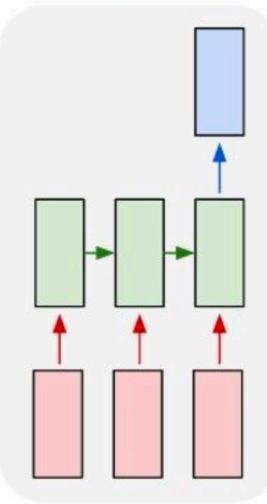
one to one



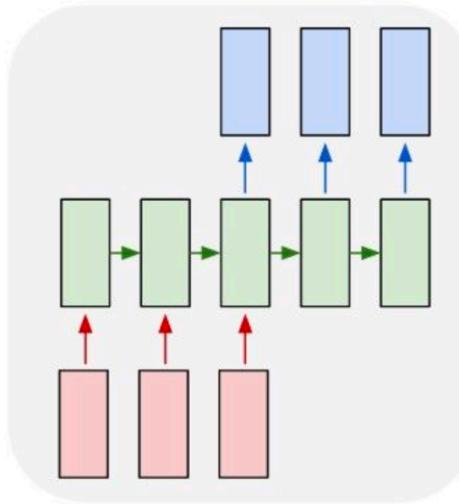
one to many



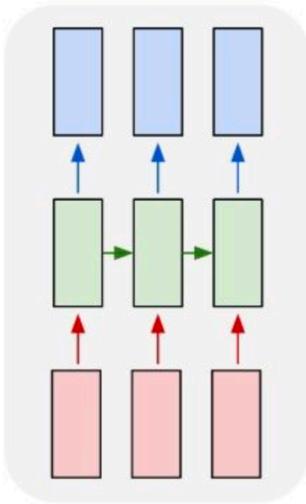
many to one



many to many

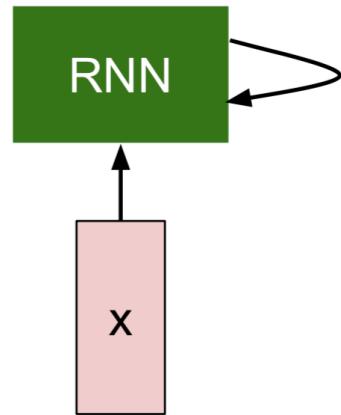


many to many

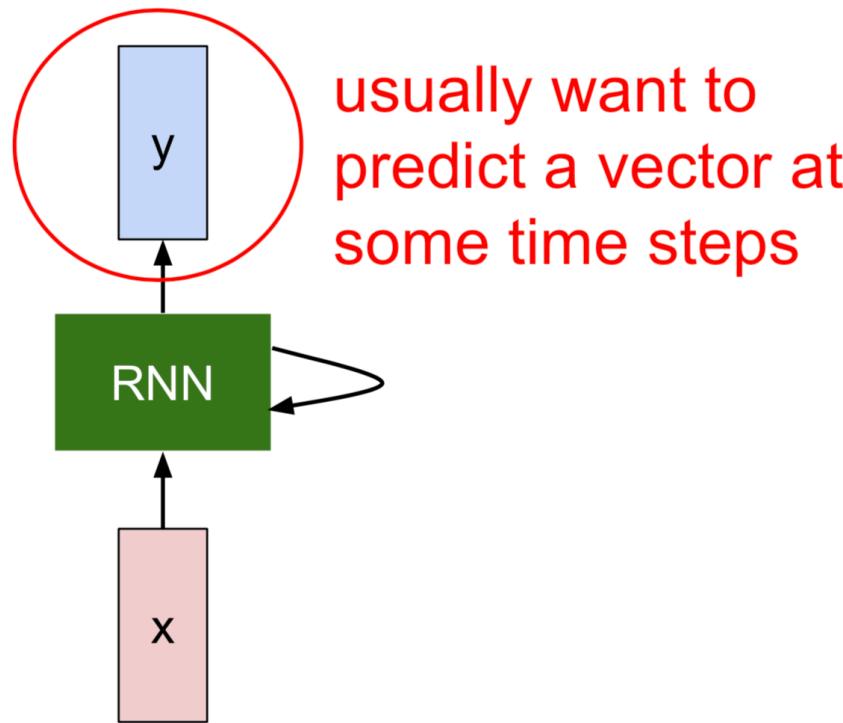


e.g. **Video classification on frame level**

# Recurrent Neural Network



# Recurrent Neural Network

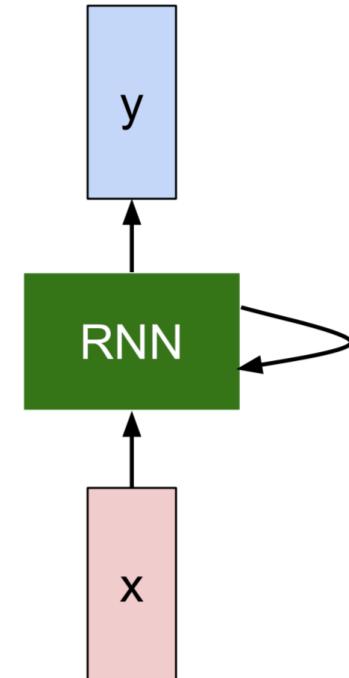


# Recurrent Neural Network

We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

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

new state      /      old state      input vector at  
                  \      some time step  
                  some function  
                  with parameters W

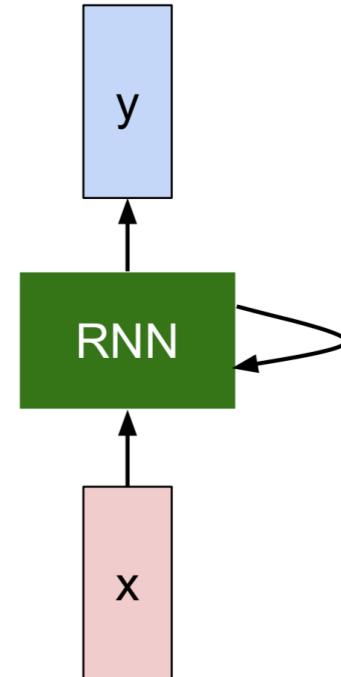


# Recurrent Neural Network

We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

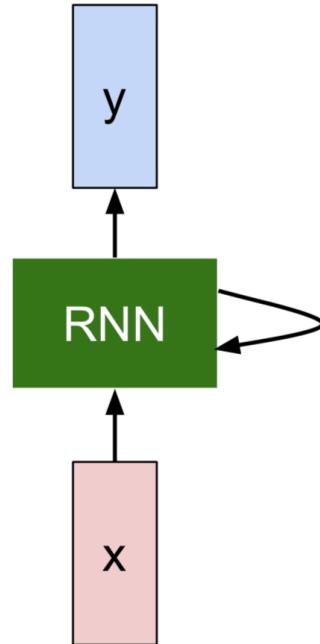
$$h_t = f_W(h_{t-1}, x_t)$$

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



# (Vanilla) Recurrent Neural Network

The state consists of a single “hidden” vector  $\mathbf{h}$ :



$$\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

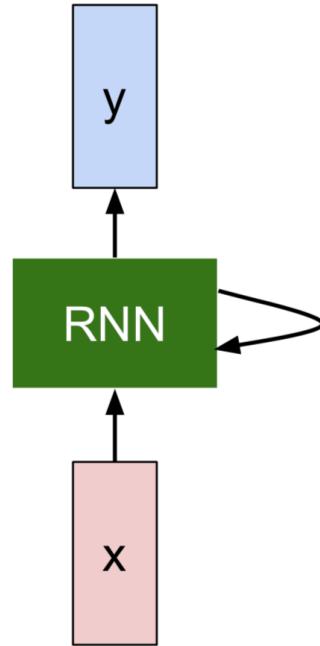


$$\mathbf{h}_t = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t)$$

$$y_t = W_{hy}\mathbf{h}_t$$

# (Vanilla) Recurrent Neural Network

The state consists of a single “hidden” vector  $\mathbf{h}$ :



$$\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



$$\mathbf{h}_t = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t)$$

$$y_t = W_{hy}\mathbf{h}_t$$

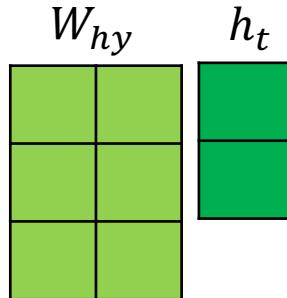
# 그림으로 대략 그려보면

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

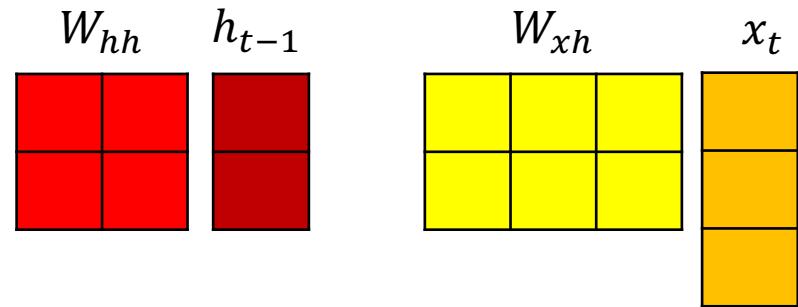


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

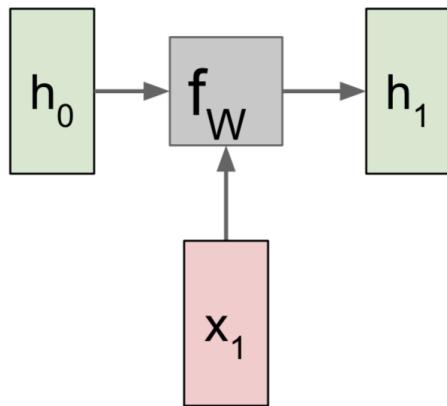
$$y_t = W_{hy}h_t$$



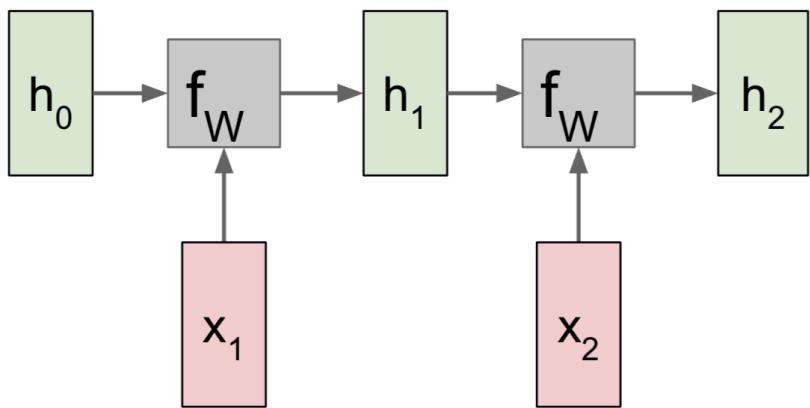
- 가정
- Input : (3,1)
  - Hidden : (2,1)
  - Output : (3,1)



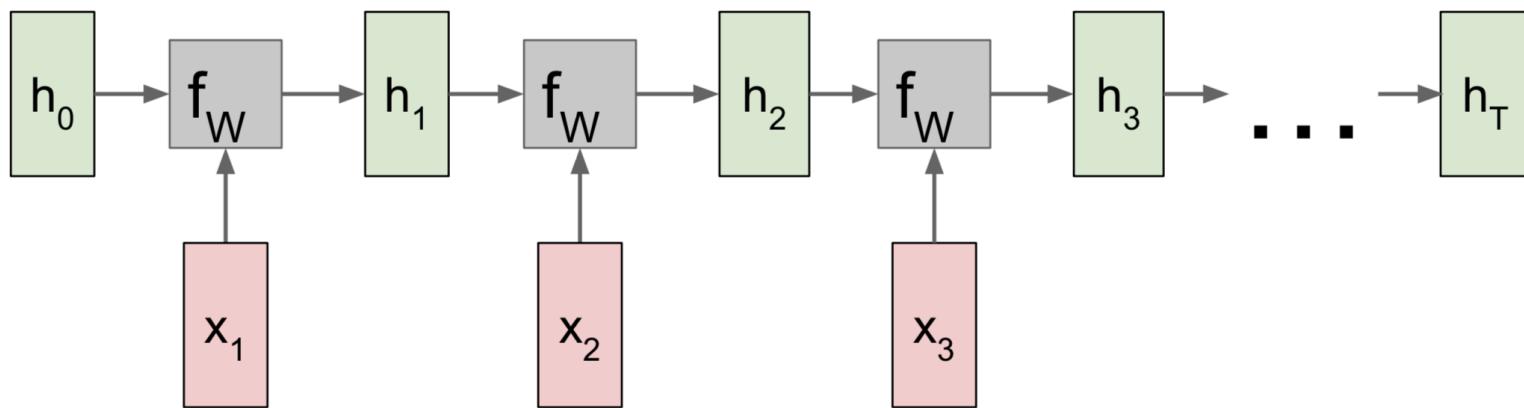
# RNN: Computational Graph



# RNN: Computational Graph

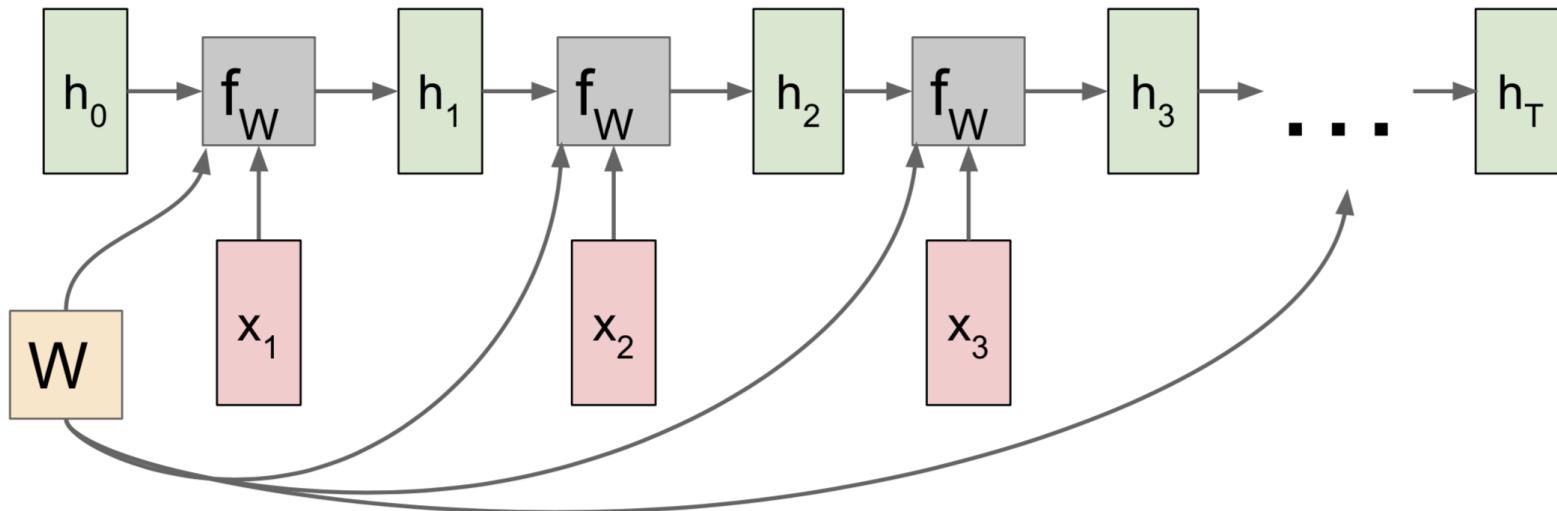


# RNN: Computational Graph

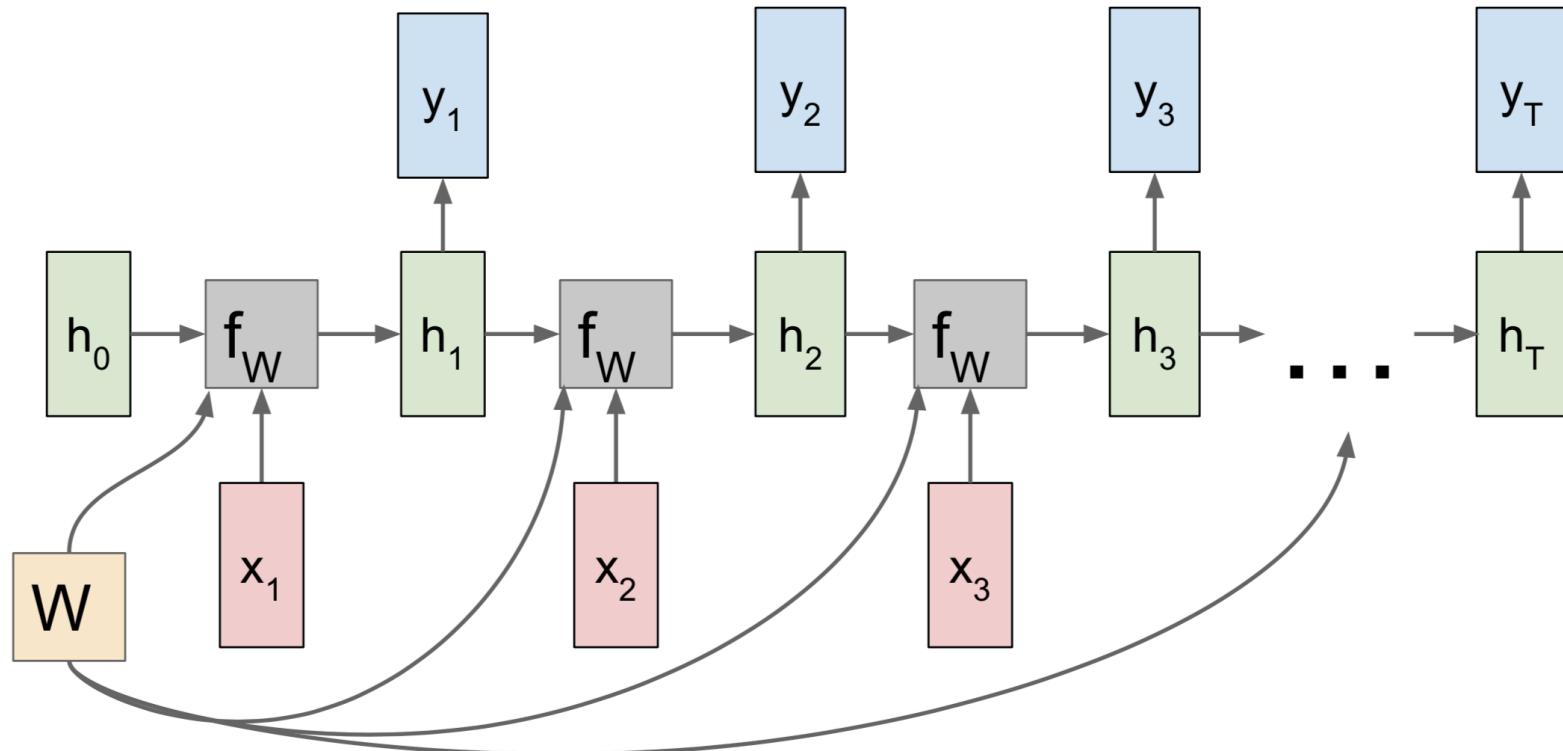


# RNN: Computational Graph

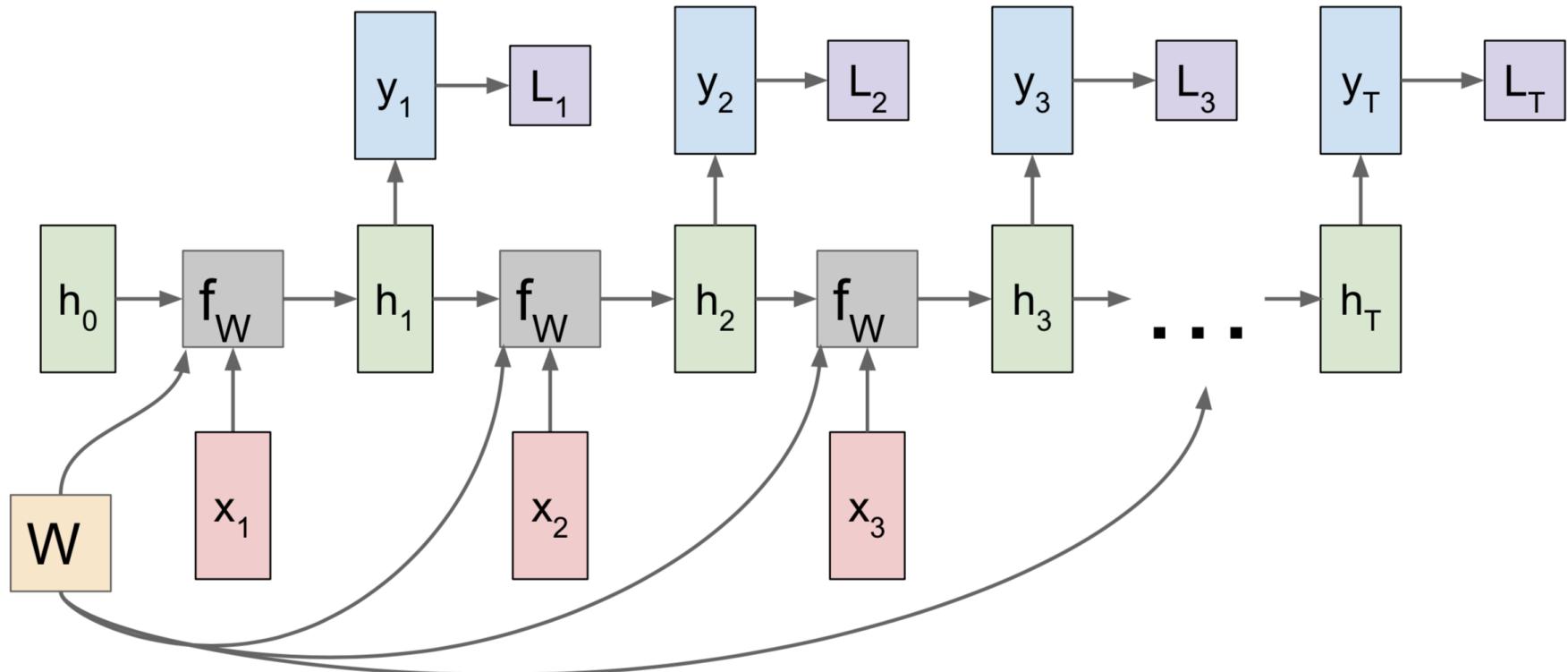
Re-use the same weight matrix at every time-step



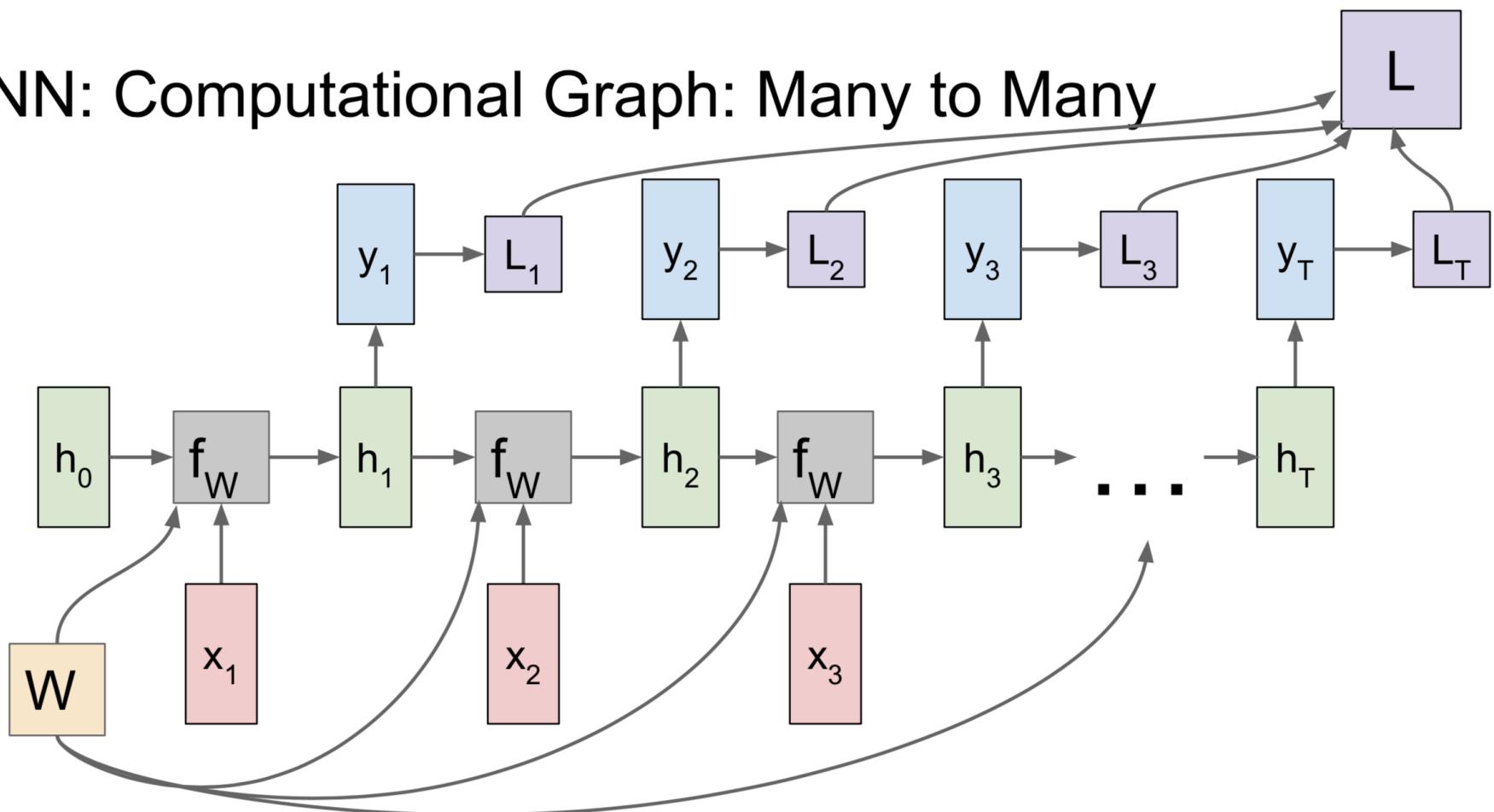
# RNN: Computational Graph: Many to Many



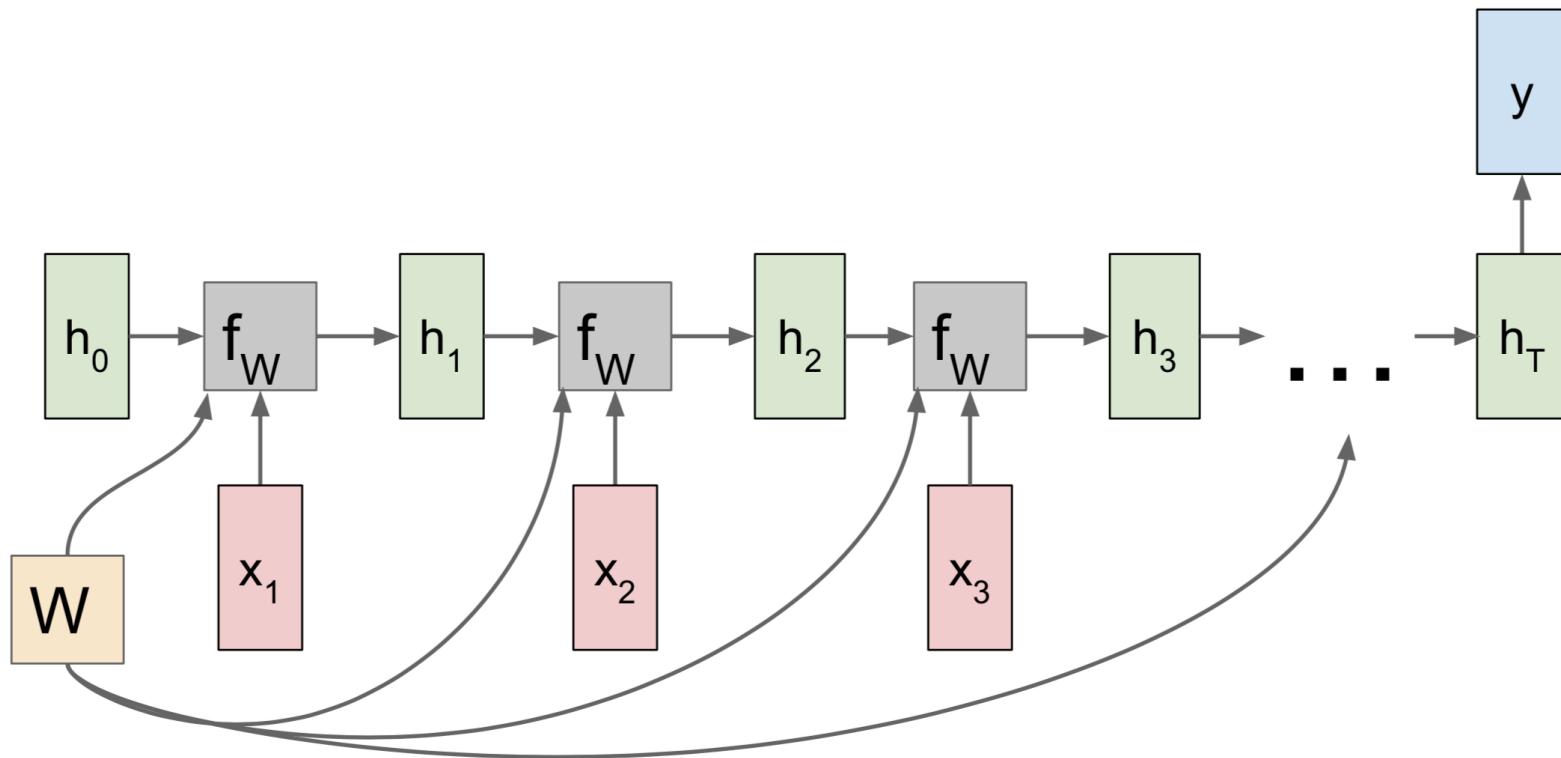
# RNN: Computational Graph: Many to Many



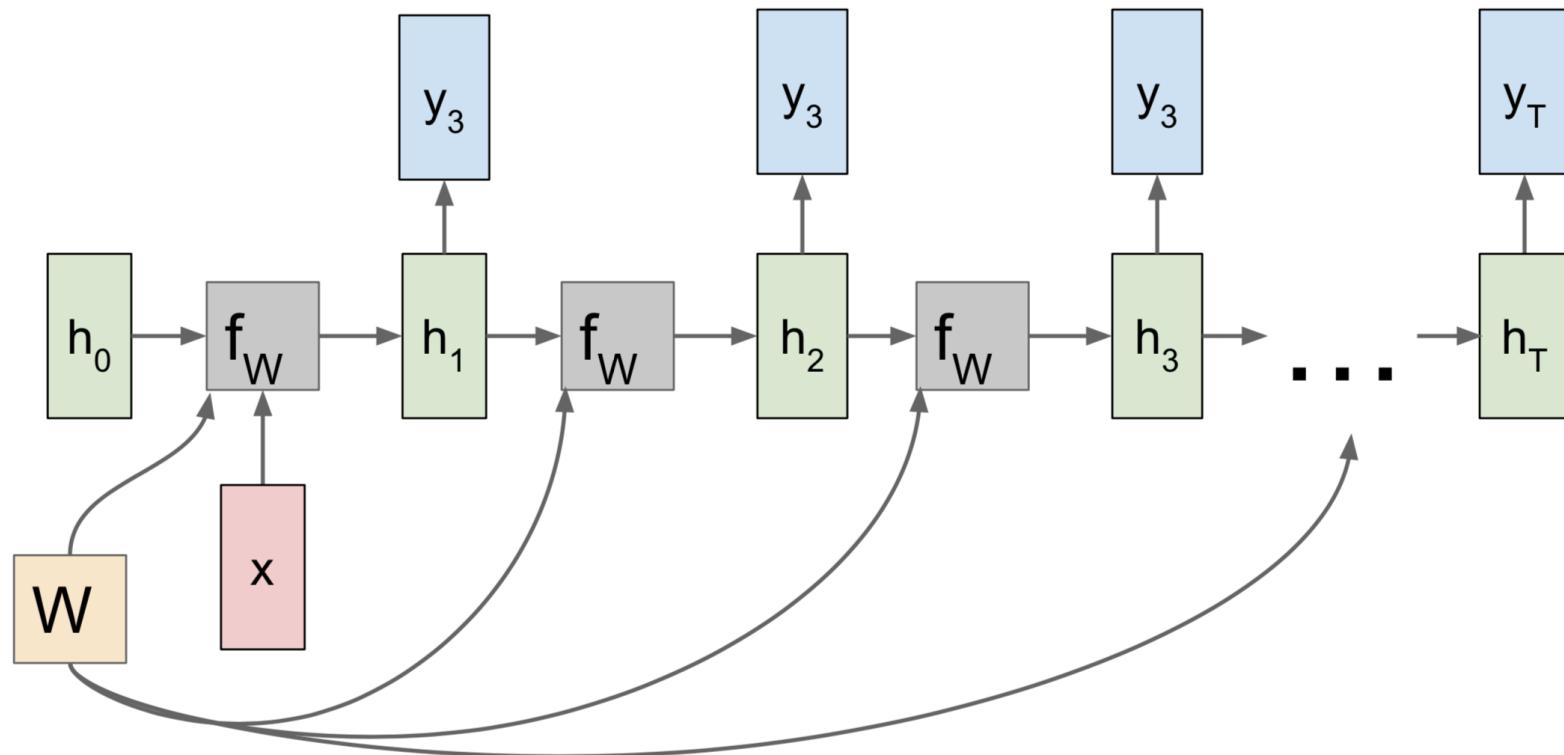
# RNN: Computational Graph: Many to Many



# RNN: Computational Graph: Many to One

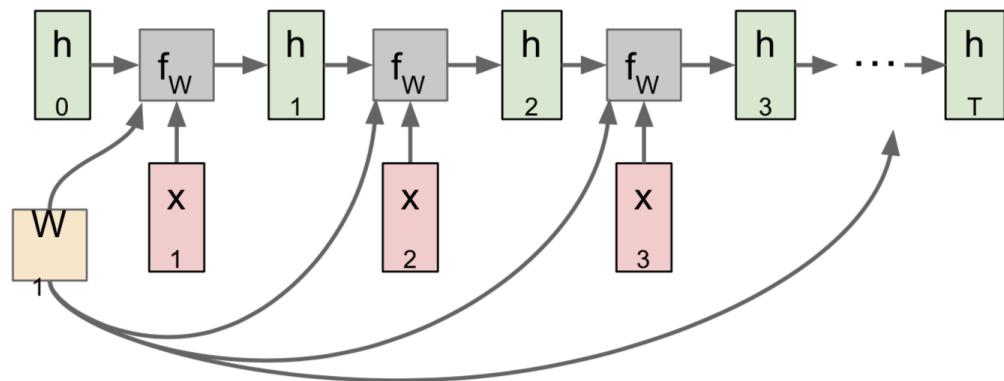


# RNN: Computational Graph: One to Many



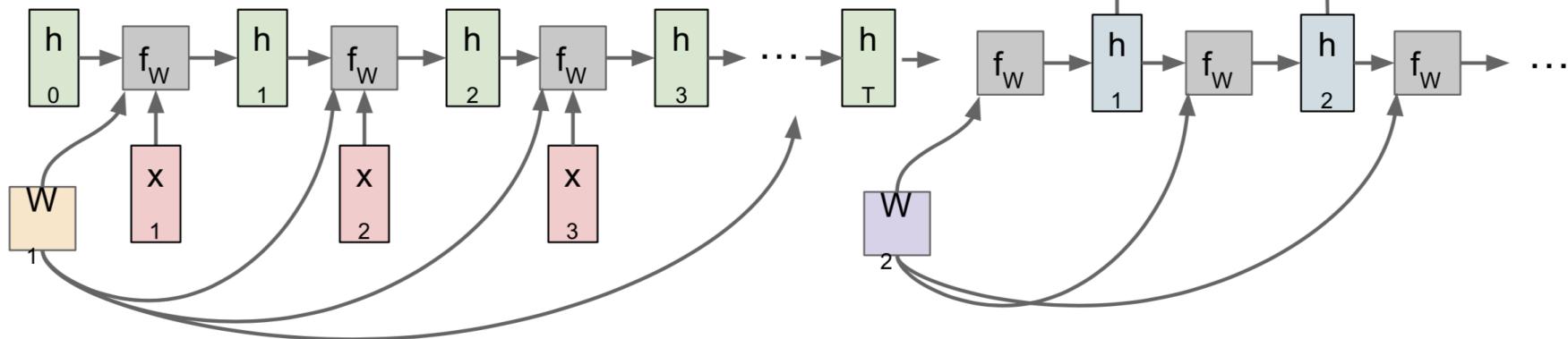
# Sequence to Sequence: Many-to-one + one-to-many

**Many to one:** Encode input sequence in a single vector



# Sequence to Sequence: Many-to-one + one-to-many

**Many to one:** Encode input sequence in a single vector

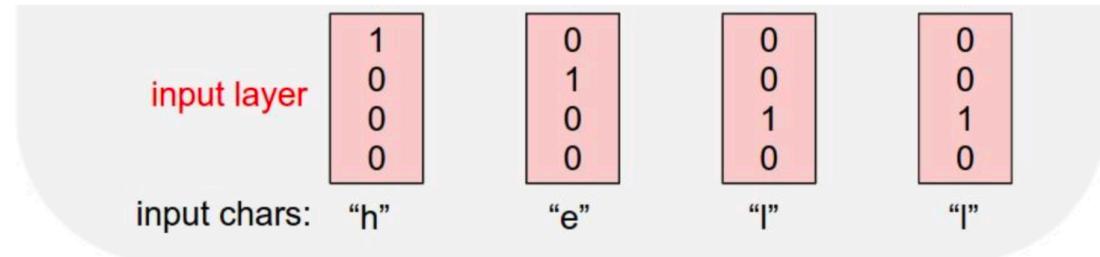


**One to many:** Produce output sequence from single input vector

# Example: Character-level Language Model

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
“hello”

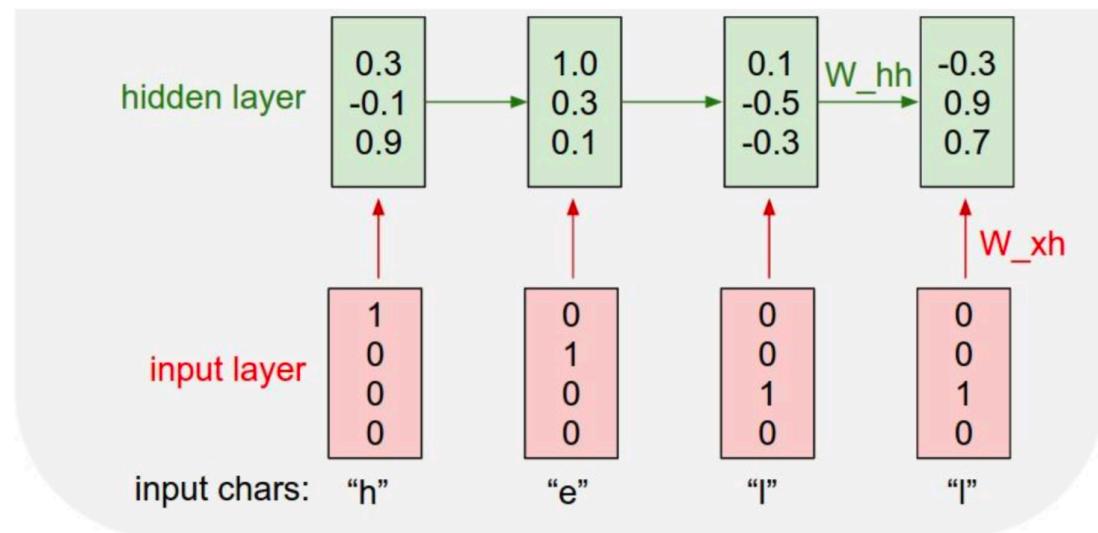


# Example: Character-level Language Model

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
“hello”

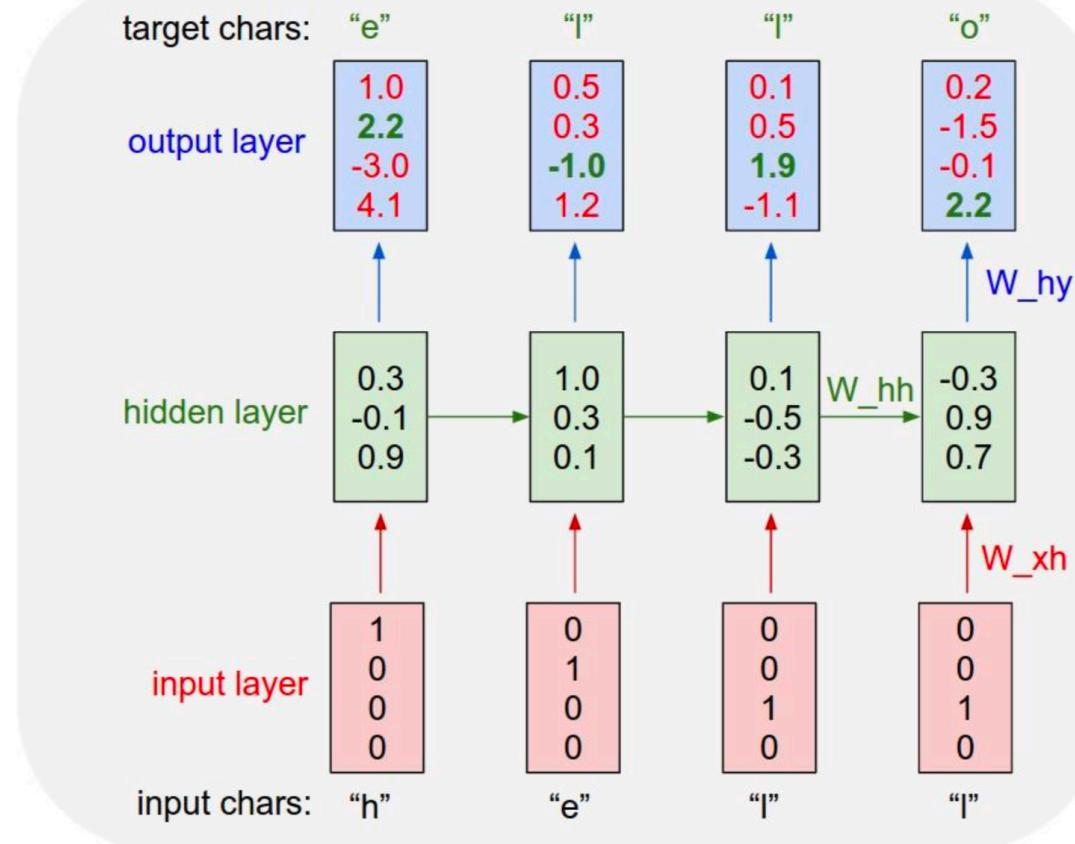
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



# Example: Character-level Language Model

Vocabulary:  
[h,e,l,o]

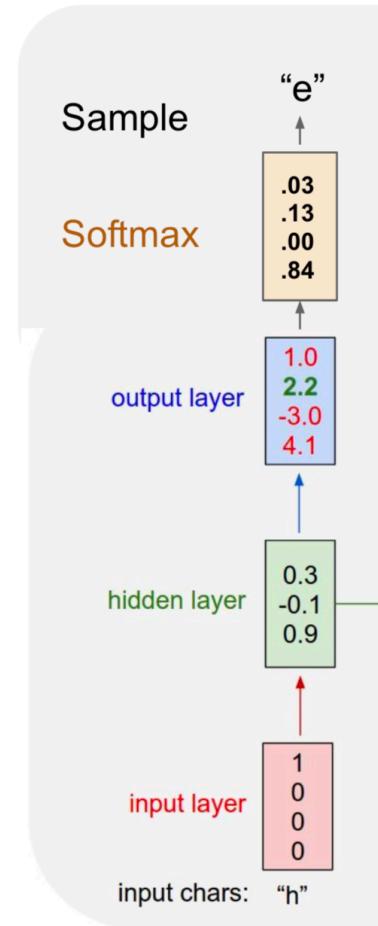
Example training  
sequence:  
“hello”



# Example: Character-level Language Model Sampling

Vocabulary:  
[h,e,l,o]

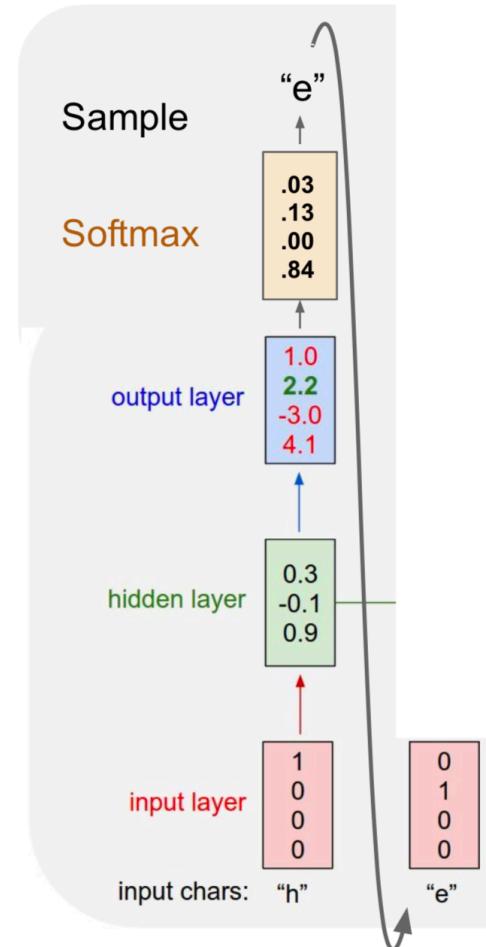
At test-time sample  
characters one at a time,  
feed back to model



# Example: Character-level Language Model Sampling

Vocabulary:  
[h,e,l,o]

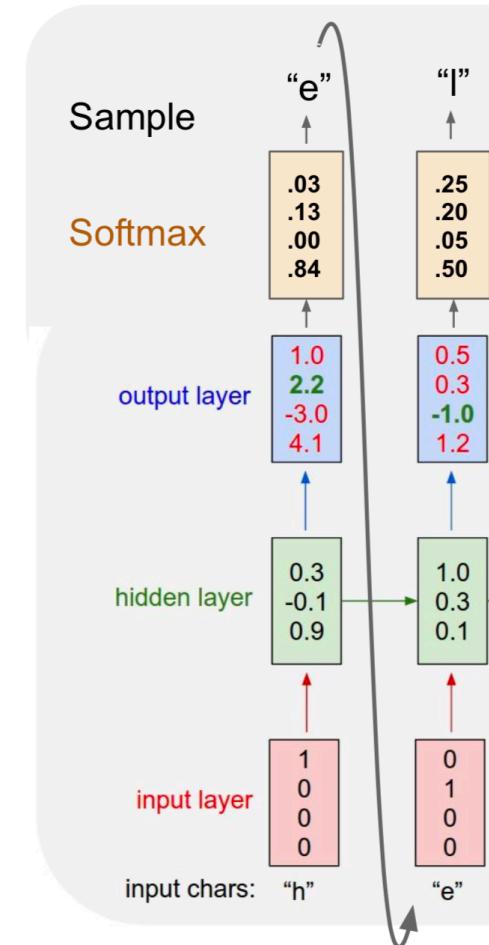
At test-time sample  
characters one at a time,  
feed back to model



# Example: Character-level Language Model Sampling

Vocabulary:  
[h,e,l,o]

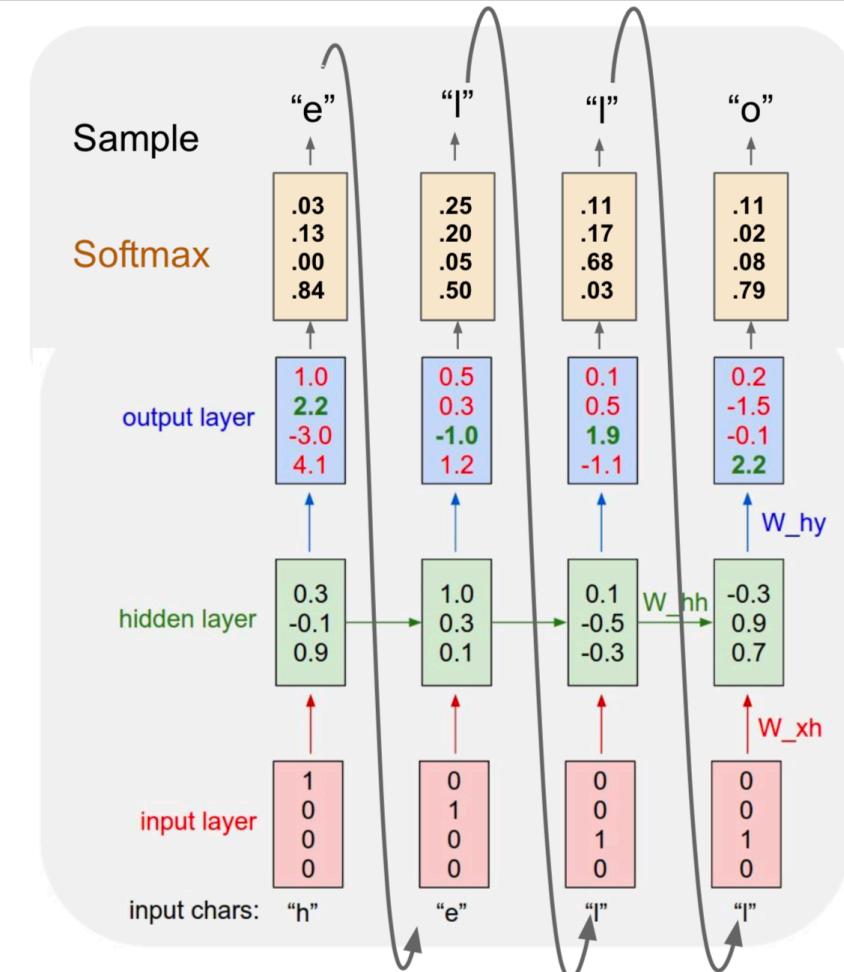
At test-time sample  
characters one at a time,  
feed back to model



# Example: Character-level Language Model Sampling

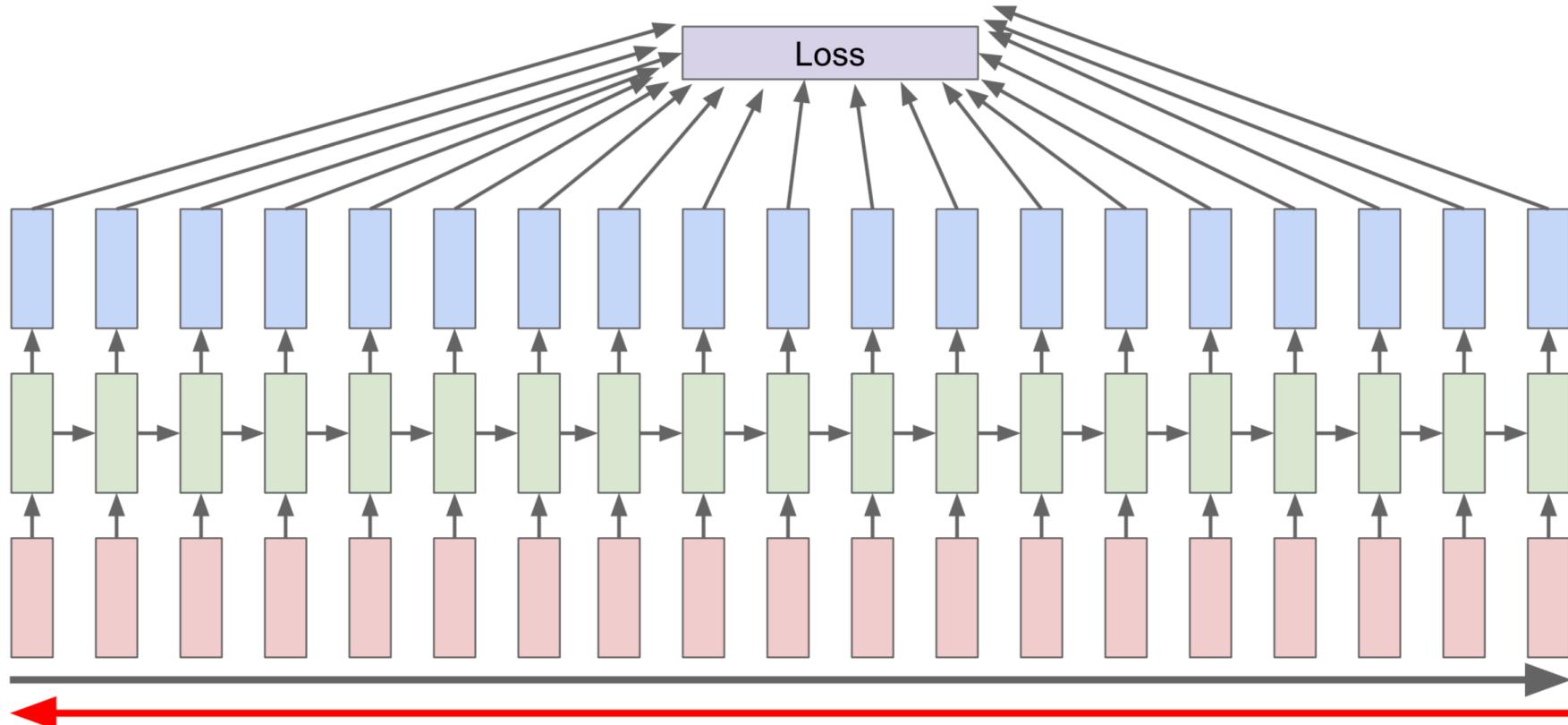
Vocabulary:  
[h,e,l,o]

At test-time sample  
characters one at a time,  
feed back to model

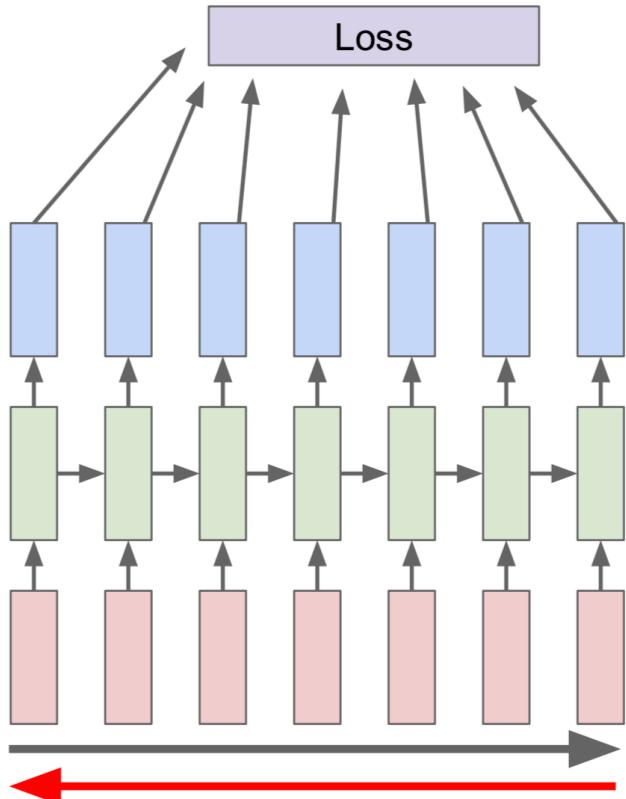


# Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

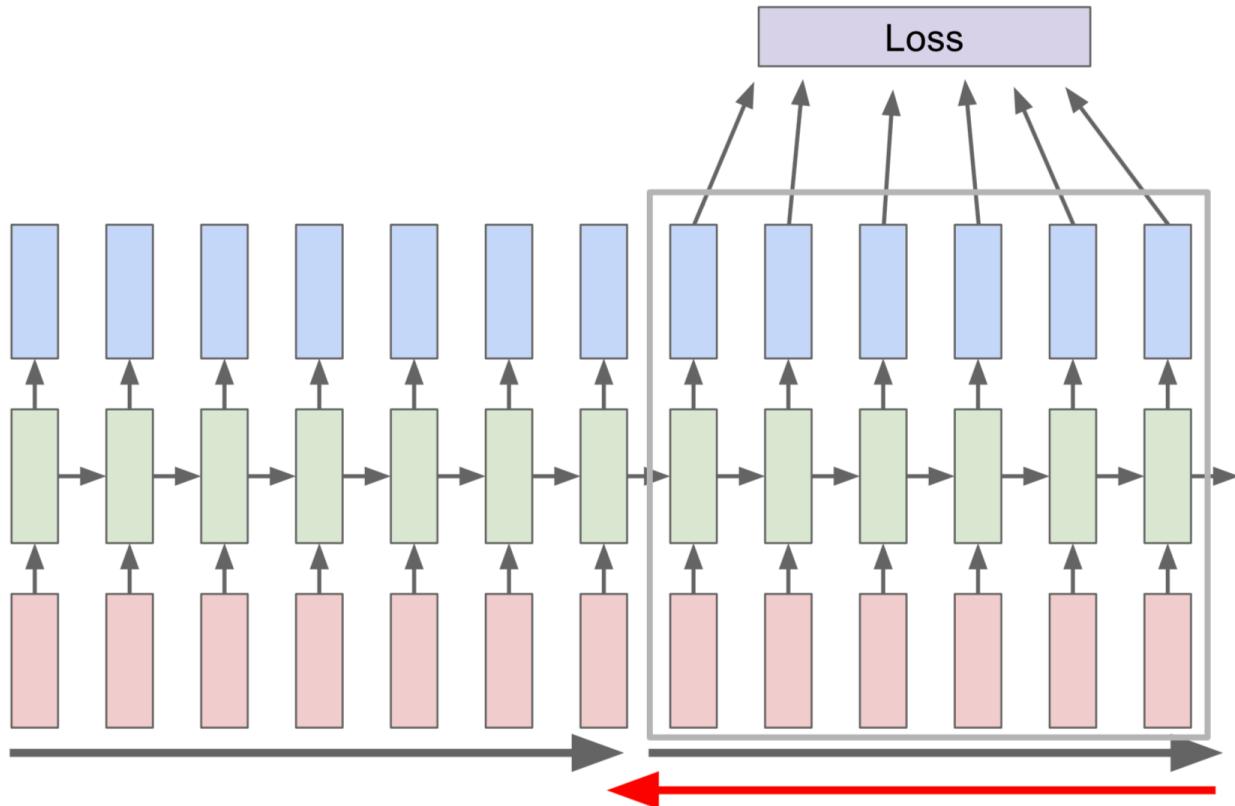


# Truncated Backpropagation through time



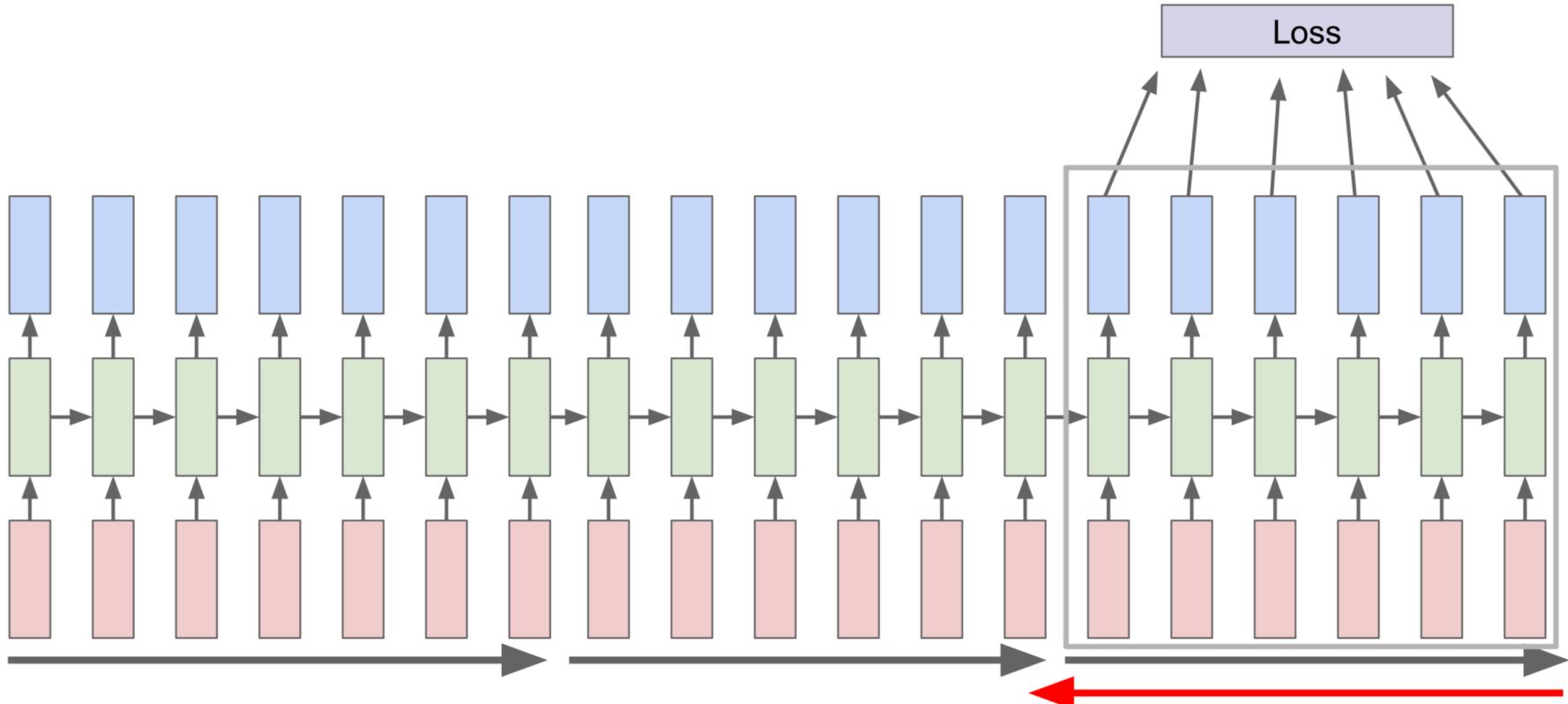
Run forward and backward  
through chunks of the  
sequence instead of whole  
sequence

# Truncated Backpropagation through time



Carry hidden states  
forward in time forever,  
but only backpropagate  
for some smaller  
number of steps

# Truncated Backpropagation through time



## min-char-rnn.py gist: 112 lines of Python

```

1 /**
2 Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3 BSD license
4 ***
5 import numpy as np
6
7 # data I/O
8 data = open('input.txt', 'r').read() # should be simple plain text file
9 chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print 'data has %d characters, %d unique.' % (data_size, vocab_size)
12 char_to_ix = { ch:i for i,ch in enumerate(chars) }
13 ix_to_char = { i:ch for i,ch in enumerate(chars) }
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 Wkh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
22 Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) * hidden_bias
25 by = np.zeros((vocab_size, 1)) * output_bias
26
27 def lossFun(inputs, targets, hprev):
28
29     inputs, targets = list_of_integers_to_matrixes(inputs, targets)
30     hprev = hprev_array_of_initial_hidden_state
31     return the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = ([], [], [], [])
34     hs[-1] = np.copy(hprev)
35     loss = 0
36     # forward pass
37     for t in xrange(len(inputs)):
38         xs[t] = np.zeros((vocab_size, 1)) # encode in 1-of-k representation
39         x[1][inputs[t]] = 1 # not one-hot because we use softmax later
40         h[1] = np.tanh(np.dot(wh, h[1-1]) + bh) # hidden state
41         y[1] = np.dot(why, h[1]) / np.sum(np.exp(y[1])) # unnormalized log probabilities for next chars
42         ps[1] = np.exp(y[1]) / np.sum(ps[1]) # softmax (cross-entropy loss)
43         loss += -np.log(ps[1][targets[1]]) # softmax (cross-entropy loss)
44         # backward pass: compute gradients going backwards
45         dwh, dbh, dwhy, dy = np.zeros_like(wh), np.zeros_like(bh), np.zeros_like(why)
46         dbx, dhy, dby = np.zeros_like(xs[0]), np.zeros_like(h[0]), np.zeros_like(by)
47         dhnext = np.zeros_like(h[0])
48         for t in reversed(xrange(len(inputs))):
49             dy = np.copy(ps[t])
50             dy[targets[t]] = 1 # backprop into y
51             dhy = np.dot(dy, h[1:T])
52             dby = dy
53             dh = np.dot(why.T, dy) + dhnext # backprop into h
54             dhrw = (1 - h[1:T]) * dh # backprop through tanh nonlinearity
55             dwh += np.dot(x[1:T].T, dhrw)
56             dbh += dhrw
57             dwhy += np.dot(dhrw, h[1:T])
58             dhnext = np.dot(whh.T, dhrw)
59             for param in [dwh, dbh, dwhy, dhy, dby]:
60                 np.clip(param, -5, 5, out=param) # clip to mitigate exploding gradients
61             return loss, dwh, dbh, dwhy, dhy, dby, h[1:len(inputs)-1]
62
63     def sample(h, seed_ix, n):
64         """
65         sample a sequence of integers from the model
66         h is memory state, seed_ix is seed letter for first time step
67         """
68         if n == 1:
69             x = np.zeros((vocab_size, 1))
70             x[seed_ix] = 1
71             ixes = []
72             for t in xrange(n):
73                 h = np.tanh(np.dot(whh, x) + np.dot(whh, h) + bh)
74                 y = np.dot(why, h) / np.sum(np.exp(y))
75                 p = np.exp(y) / np.sum(np.exp(y))
76                 ix = np.random.choice(range(vocab_size), p=p.ravel())
77                 x = np.zeros((vocab_size, 1))
78                 x[ix] = 1
79                 ixes.append(ix)
80             return ixes
81
82         n, p = 0, 0
83         meth, mabh, mwhy = np.zeros_like(whh), np.zeros_like(whh), np.zeros_like(why)
84         mbh, mbhy = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
85         smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
86         while True:
87             # sample inputs (we're unzipping from left to right in steps seq_length long)
88             if p+seq_length >= len(data) or n == 0:
89                 hprev = np.zeros((hidden_size, 1)) # reset RNN memory
90                 p = 0 # go from start of data
91                 inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
92                 targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
93
94             # sample from the model now and then
95             if n % 100 == 0:
96                 sample_ix = sample(hprev, inputs[0], 200)
97                 txt = ''.join(ix_to_char[i] for i in sample_ix)
98                 print '----\n' + txt + '\n----'
99
100            # forward seq_length characters through the net and fetch gradient
101            loss, dxwh, dmbh, dmwhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
102            smooth_loss = smooth_loss * .999 + loss * .001
103            if n % 100 == 0: print 'iter %d, loss %f' % (n, smooth_loss) # print progress
104
105            # perform parameter update with Adagrad
106            for param, dpParam, mem in zip([whh, whh, why, bh, bh, by],
107                                         [dwhh, dwh, dwhy, dbh, dbh, dby],
108                                         [meth, mabh, mwhy, mbh, mbhy]):
109                mem += dpParam * dpParam
110                param -= learning_rate * dpParam / np.sqrt(mem + 1e-8) # adagrad update
111
112            p += seq_length # move data pointer
113            n += 1 # iteration counter

```

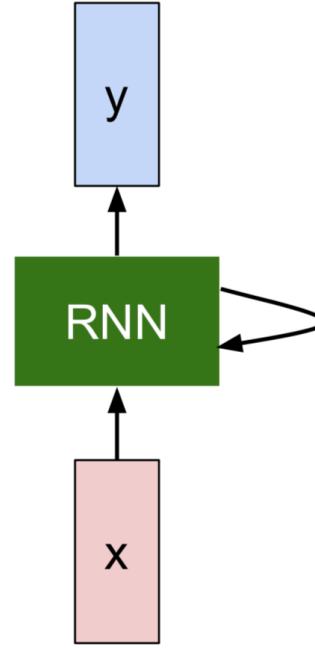
(<https://gist.github.com/karpathy/d4dee566867f8291f086>)

# THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,  
That thereby beauty's rose might never die,  
But as the riper should by time decease,  
His tender heir might bear his memory:  
But thou, contracted to thine own bright eyes,  
Feed'st thy light's flame with self-substantial fuel,  
Making a famine where abundance lies,  
Thyself thy foe, to thy sweet self too cruel:  
Thou that art now the world's fresh ornament,  
And only herald to the gaudy spring,  
Within thine own bud buriest thy content,  
And tender churl mak'st waste in niggarding:  
Pity the world, or else this glutton be,  
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,  
And dig deep trenches in thy beauty's field,  
Thy youth's proud livery so gazed on now,  
Will be a tatter'd weed of small worth held:  
Then being asked, where all thy beauty lies,  
Where all the treasure of thy lusty days;  
To say, within thine own deep sunken eyes,  
Were an all-eating shame, and thriftless praise.  
How much more praise deserved thy beauty's use,  
If thou couldst answer 'This fair child of mine  
Shall sum my count, and make my old excuse,'  
Proving his beauty by succession thine!  
This were to be new made when thou art old,  
And see thy blood warm when thou feel'st it cold.



at first:

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e  
plia tkldrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomesscerliund  
Keushey. Thom here  
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwyl fil on aseterlome  
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of  
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort  
how, and Gogition is so overelical and ofter.

↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the  
princess, Princess Mary was easier, fed in had oftened him.  
Pierre aking his soul came to the packs and drove up his father-in-law women.

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain 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.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

VIOLA:

Why, Salisbury must find his flesh and thought  
That which I am not aps, not a man and in fire,  
To show the reining of the raven and the wars  
To grace my hand reproach within, and not a fair are hand,  
That Caesar and my goodly father's world;  
When I was heaven of presence and our fleets,  
We spare with hours, but cut thy council I am great,  
Murdered and by thy master's ready there  
My power to give thee but so much as hell:  
Some service in the noble bondman here,  
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,  
Your sight and several breath, will wear the gods  
With his heads, and my hands are wonder'd at the deeds,  
So drop upon your lordship's head, and your opinion  
Shall be against your honour.

# The Stacks Project: open source algebraic geometry textbook

The screenshot shows the homepage of The Stacks Project. At the top, there is a navigation bar with links: home, about, tags explained, tag lookup, browse, search, bibliography, recent comments, blog, and add slogans. Below the navigation bar, there is a section titled "Browse chapters". A table lists chapters under the "Preliminaries" part. Each chapter row contains three columns: Chapter name, online link, TeX source link, and view pdf link. To the right of the table, there is a sidebar with sections for "Parts" and "Statistics". The "Parts" section lists numbered links to various topics: Preliminaries, Schemes, Topics in Scheme Theory, Algebraic Spaces, Topics in Geometry, Deformation Theory, Algebraic Stacks, and Miscellany. The "Statistics" section provides information about the project's size: 455910 lines of code, 14221 tags (56 inactive tags), and 2366 sections.

Part	Chapter	online	TeX source	view pdf
Preliminaries	1. Introduction	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	2. Conventions	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	3. Set Theory	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	4. Categories	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	5. Topology	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	6. Sheaves on Spaces	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	7. Sites and Sheaves	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	8. Stacks	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	9. Fields	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>
	10. Commutative Algebra	<a href="#">online</a>	<a href="#">tex</a>	<a href="#">pdf</a>

Latex source

<http://stacks.math.columbia.edu/>  
The stacks project is licensed under the [GNU Free Documentation License](#)

For  $\bigoplus_{n=1,\dots,m} \text{where } \mathcal{L}_{m,n} = 0$ , hence we can find a closed subset  $\mathcal{H}$  in  $\mathcal{H}$  and any sets  $\mathcal{F}$  on  $X$ ,  $U$  is a closed immersion of  $S$ , then  $U \rightarrow T$  is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by  $\coprod Z \times_U U \rightarrow V$ . Consider the maps  $M$  along the set of points  $\text{Sch}_{fppf}$  and  $U \rightarrow U$  is the fibre category of  $S$  in  $U$  in Section, ?? and the fact that any  $U$  affine, see Morphisms, Lemma ???. Hence we obtain a scheme  $S$  and any open subset  $W \subset U$  in  $\text{Sh}(G)$  such that  $\text{Spec}(R') \rightarrow S$  is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that  $f_i$  is of finite presentation over  $S$ . We claim that  $\mathcal{O}_{X,x}$  is a scheme where  $x, x', s'' \in S'$  such that  $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}_{X',x'}$  is separated. By Algebra, Lemma ?? we can define a map of complexes  $\text{GL}_{S'}(x'/S'')$  and we win.  $\square$

To prove study we see that  $\mathcal{F}|_U$  is a covering of  $\mathcal{X}'$ , and  $\mathcal{T}_i$  is an object of  $\mathcal{F}_{X/S}$  for  $i > 0$  and  $\mathcal{F}_p$  exists and let  $\mathcal{F}_i$  be a presheaf of  $\mathcal{O}_X$ -modules on  $\mathcal{C}$  as a  $\mathcal{F}$ -module. In particular  $\mathcal{F} = U/\mathcal{F}$  we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{\text{opp}}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \hookrightarrow (U, \text{Spec}(A))$$

is an open subset of  $X$ . Thus  $U$  is affine. This is a continuous map of  $X$  is the inverse, the groupoid scheme  $S$ .

*Proof.* See discussion of sheaves of sets.  $\square$

The result for prove any open covering follows from the less of Example ???. It may replace  $S$  by  $X_{\text{spaces},\text{étale}}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{\text{Zar}}$ , see Descent, Lemma ???. Namely, by Lemma ?? we see that  $R$  is geometrically regular over  $S$ .

**Lemma 0.1.** Assume (3) and (3) by the construction in the description.

Suppose  $X = \lim |X|$  (by the formal open covering  $X$  and a single map  $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$  over  $U$  compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that  $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$  is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If  $T$  is surjective we may assume that  $T$  is connected with residue fields of  $S$ . Moreover there exists a closed subspace  $Z \subset X$  of  $X$  where  $U$  in  $X'$  is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1)  $f$  is locally of finite type. Since  $S = \text{Spec}(R)$  and  $Y = \text{Spec}(R)$ .

*Proof.* This is form all sheaves of sheaves on  $X$ . But given a scheme  $U$  and a surjective étale morphism  $U \rightarrow X$ . Let  $U \cap U = \coprod_{i=1,\dots,n} U_i$  be the scheme  $X$  over  $S$  at the schemes  $X_i \rightarrow X$  and  $U = \lim_i X_i$ .  $\square$

The following lemma surjective restrocomposes of this implies that  $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x,\dots,x}$ .

**Lemma 0.2.** Let  $X$  be a locally Noetherian scheme over  $S$ ,  $E = \mathcal{F}_{X/S}$ . Set  $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$ . Since  $\mathcal{I}^n \subset \mathcal{I}^n$  are nonzero over  $i_0 \leq p$  is a subset of  $\mathcal{J}_{n,0} \circ \overline{A}_2$  works.

**Lemma 0.3.** In Situation ???. Hence we may assume  $q' = 0$ .

*Proof.* We will use the property we see that  $p$  is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where  $K$  is an  $F$ -algebra where  $\delta_{n+1}$  is a scheme over  $S$ .  $\square$

*Proof.* Omitted.  $\square$

**Lemma 0.1.** Let  $\mathcal{C}$  be a set of the construction.

Let  $\mathcal{C}$  be a gerber covering. Let  $\mathcal{F}$  be a quasi-coherent sheaves of  $\mathcal{O}$ -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

*Proof.* This is an algebraic space with the composition of sheaves  $\mathcal{F}$  on  $X_{\text{étale}}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where  $\mathcal{G}$  defines an isomorphism  $\mathcal{F} \rightarrow \mathcal{F}$  of  $\mathcal{O}$ -modules.  $\square$

**Lemma 0.2.** This is an integer  $\mathcal{Z}$  is injective.

*Proof.* See Spaces, Lemma ??.

**Lemma 0.3.** Let  $S$  be a scheme. Let  $X$  be a scheme and  $X$  is an affine open covering. Let  $\mathcal{U} \subset \mathcal{X}$  be a canonical and locally of finite type. Let  $X$  be a scheme. Let  $X$  be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let  $X$  be a scheme. Let  $X$  be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over  $S$  and  $Y$ .

*Proof.* Let  $X$  be a nonzero scheme of  $X$ . Let  $X$  be an algebraic space. Let  $\mathcal{F}$  be a quasi-coherent sheaf of  $\mathcal{O}_X$ -modules. The following are equivalent

- (1)  $\mathcal{F}$  is an algebraic space over  $S$ .
- (2) If  $X$  is an affine open covering.

Consider a common structure on  $X$  and  $X$  the functor  $\mathcal{O}_X(U)$  which is locally of finite type.  $\square$

This since  $\mathcal{F} \in \mathcal{F}$  and  $x \in \mathcal{G}$  the diagram

$$\begin{array}{ccccc}
 S & \xrightarrow{\quad} & & & \\
 \downarrow & & & & \\
 \xi & \xrightarrow{\quad} & \mathcal{O}_{X'} & \xleftarrow{\quad} & \\
 \text{gor}_s & & \uparrow & & \\
 & & =\alpha' \xrightarrow{\quad} & & \\
 & & \uparrow & & \\
 & & =\alpha' \xrightarrow{\quad} \alpha & & \\
 \text{Spec}(K_\psi) & & \text{Mor}_{\text{Sets}} & & X \\
 & & & & \downarrow d(\mathcal{O}_{X_{\text{f/k}}}, \mathcal{G})
 \end{array}$$

is a limit. Then  $\mathcal{G}$  is a finite type and assume  $S$  is a flat and  $\mathcal{F}$  and  $\mathcal{G}$  is a finite type  $f_*$ . This is of finite type diagrams, and

- the composition of  $\mathcal{G}$  is a regular sequence,
- $\mathcal{O}_{X'}$  is a sheaf of rings.

*Proof.* We have see that  $X = \text{Spec}(R)$  and  $\mathcal{F}$  is a finite type representable by algebraic space. The property  $\mathcal{F}$  is a finite morphism of algebraic stacks. Then the cohomology of  $X$  is an open neighbourhood of  $U$ .  $\square$

*Proof.* This is clear that  $\mathcal{G}$  is a finite presentation, see Lemmas ??.

A reduced above we conclude that  $U$  is an open covering of  $\mathcal{C}$ . The functor  $\mathcal{F}$  is a “field”

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_{\bar{x}} \xrightarrow{-1} (\mathcal{O}_{X_{\text{étale}}}) \rightarrow \mathcal{O}_{X_\ell}^{-1} \mathcal{O}_{X_\lambda} (\mathcal{O}_{X_\eta}^\pi)$$

is an isomorphism of covering of  $\mathcal{O}_{X_\ell}$ . If  $\mathcal{F}$  is the unique element of  $\mathcal{F}$  such that  $X$  is an isomorphism.

The property  $\mathcal{F}$  is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme  $\mathcal{O}_X$ -algebra with  $\mathcal{F}$  are opens of finite type over  $S$ .

If  $\mathcal{F}$  is a scheme theoretic image points.  $\square$

If  $\mathcal{F}$  is a finite direct sum  $\mathcal{O}_{X_\lambda}$  is a closed immersion, see Lemma ???. This is a sequence of  $\mathcal{F}$  is a similar morphism.

This repository Search

Explore Gist Blog Help

 karpathy + ·   

 torvalds / linux

 Watch · 3,711  Star 23,054  Fork 9,141

Linux kernel source tree

520,037 commits 1 branch 420 releases 5,039 contributors

branch: master · linux / +

Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux ...

torvalds authored 9 hours ago latest commit 4b1706927d

Category	Commit Message	Time Ago
Documentation	Merge git://git.kernel.org/pub/scm/linux/kernel/git/nab/target-pending	6 days ago
arch	Merge branch 'x86-urgent-for-linus' of git://git.kernel.org/pub/scm/l...	a day ago
block	block: discard bdi_unregister() in favour of bdi_destroy()	9 days ago
crypto	Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6	10 days ago
drivers	Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux	9 hours ago
firmware	firmware/ihex2fw.c: restore missing default in switch statement	2 months ago
fs	vfs: read file_handle only once in handle_to_path	4 days ago
include	Merge branch 'perf-urgent-for-linus' of git://git.kernel.org/pub/scm/...	a day ago
init	init: fix regression by supporting devices with major:minor:offset fo...	a month ago
linux		

HTTPS clone URL <https://github.com/torvalds/linux>

You can clone with **HTTPS**, **SSH**, or **Subversion**.

 Clone in Desktop  Download ZIP

Code

Pulse

Graphs

74 Pull requests

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

# Generated C code

```
/*
 * Copyright (c) 2006-2010, Intel Mobile Communications. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software Foundation,
 * Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/seteew.h>
#include <asm/pgproto.h>
```

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/seteew.h>
#include <asm/pgproto.h>

#define REG_PG      vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)      (func)

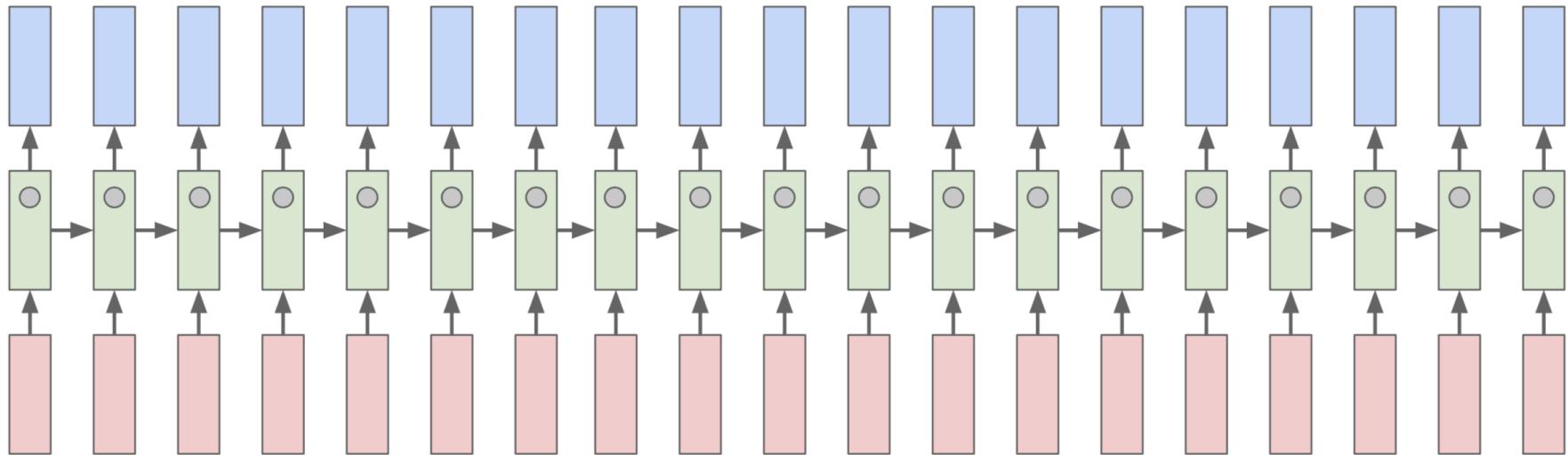
#define SWAP_ALLOCATE(nr)      (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %esp, %0, %3" : : "r" (0)); \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
    pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
                (unsigned long)-1->lr_full, low;
}

```

# Searching for interpretable cells



Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

# Searching for interpretable cells

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
    */
```

# Searching for interpretable cells

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016  
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

# Searching for interpretable cells

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact  
that it plainly and indubitably proved the fallacy of all the plans for  
cutting off the enemy's retreat and the soundness of the only possible  
line of action--the one Kutuzov and the general mass of the army  
demanded--namely, simply to follow the enemy up. The French crowd fled  
at a continually increasing speed and all its energy was directed to  
reaching its goal. It fled like a wounded animal and it was impossible  
to block its path. This was shown not so much by the arrangements it  
made for crossing as by what took place at the bridges. When the bridges  
broke down, unarmed soldiers, people from Moscow and women with children  
who were with the French transport, all--carried on by vis inertiae--  
pressed forward into boats and into the ice-covered water and did not,  
surrender.
```

## line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

# Searching for interpretable cells

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(*current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
            collect_signal(sig, pending, info);
        }
    }
    return sig;
}
```

if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016  
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

# Searching for interpretable cells

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
    struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void **) &df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM \\'%s\\' is invalid\n",
            df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

quote/comment cell

# Searching for interpretable cells

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

code depth cell

# Image Captioning

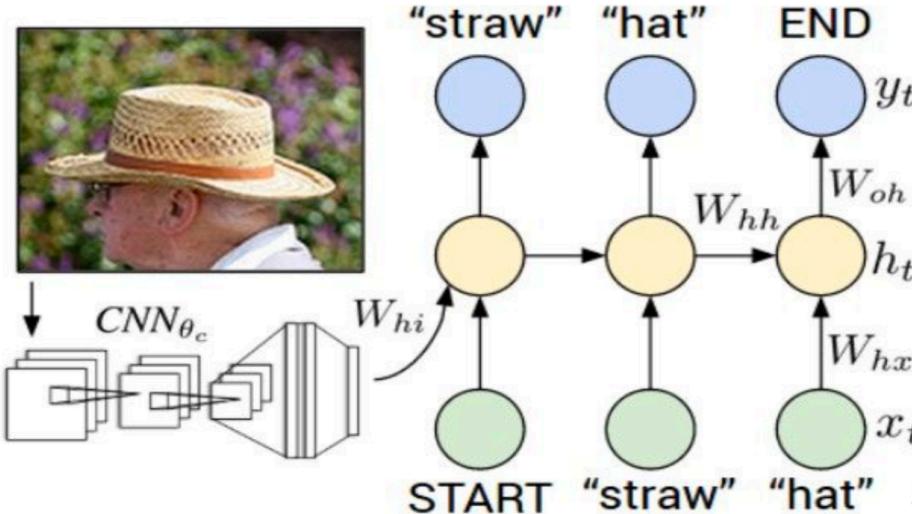


Figure from Karpathy et al, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015.  
Reproduced for educational purposes.

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.

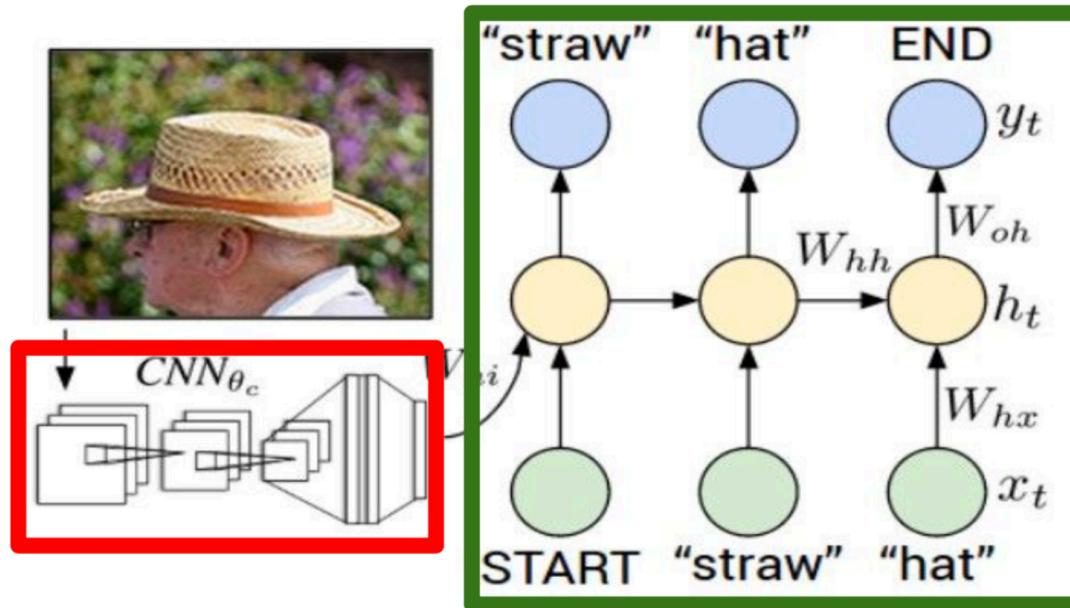
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei

Show and Tell: A Neural Image Caption Generator, Vinyals et al.

Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.

Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

# Recurrent Neural Network



## Convolutional Neural Network

test image



[This image is CC0 public domain](#)

image



test image



conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

image



test image



conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

X

image



test image



conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

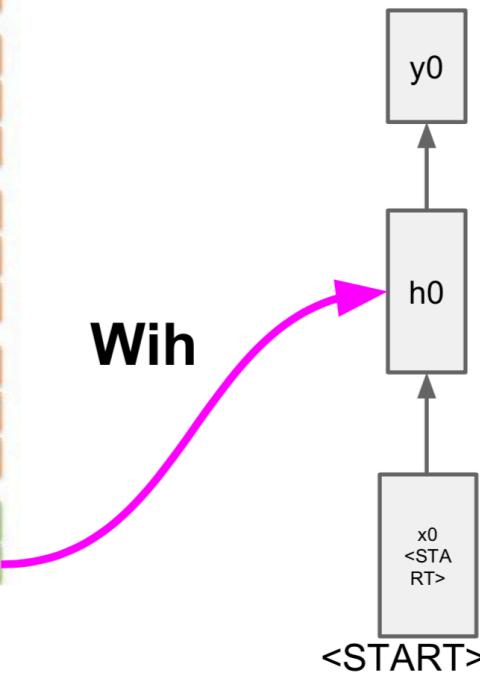
maxpool

FC-4096

FC-4096



<START>



**before:**

$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

**now:**

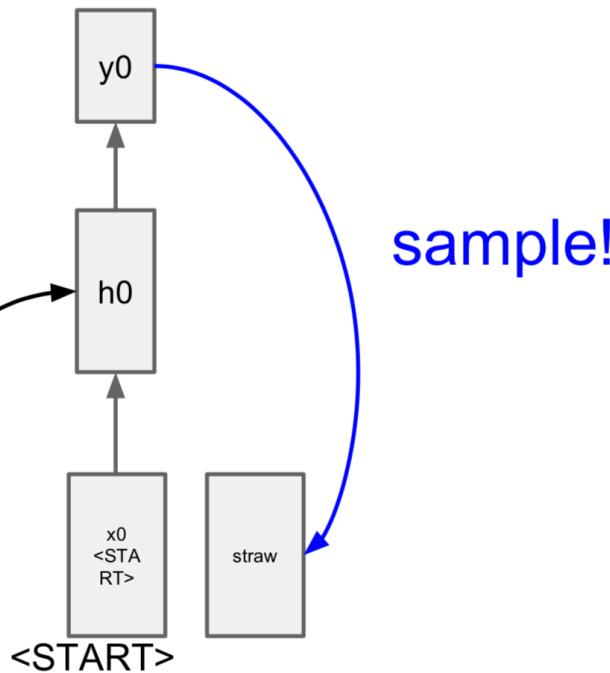
$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$$



test image



test image



image



conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

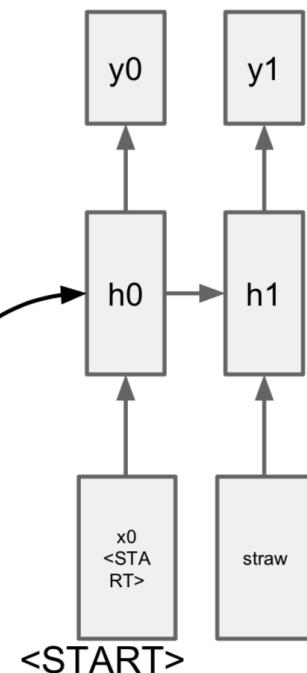
maxpool

FC-4096

FC-4096

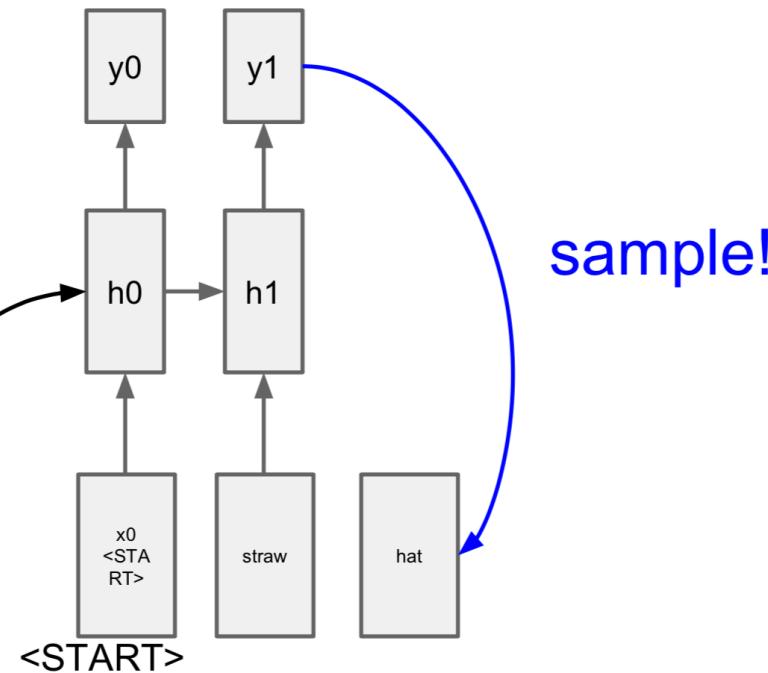


test image



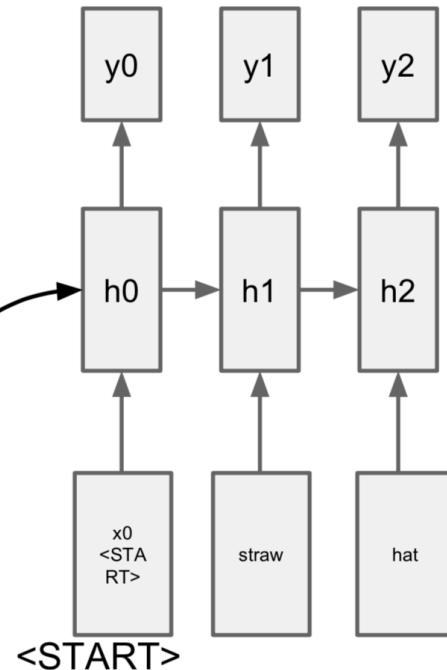


test image





test image



image



test image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

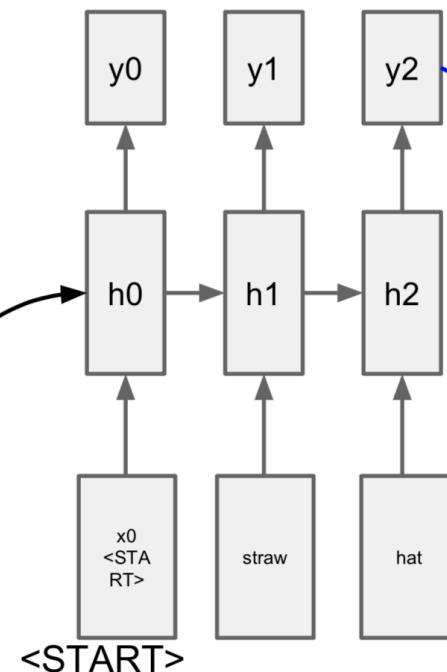
conv-512

conv-512

maxpool

FC-4096

FC-4096



sample  
<END> token  
=> finish.

# Image Captioning: Example Results



*A cat sitting on a suitcase on the floor*



*A cat is sitting on a tree branch*



*A dog is running in the grass with a frisbee*



*A white teddy bear sitting in the grass*



*Two people walking on the beach with surfboards*



*A tennis player in action on the court*



*Two giraffes standing in a grassy field*



*A man riding a dirt bike on a dirt track*

# Image Captioning: Failure Cases



*A woman is holding a cat in her hand*



*A person holding a computer mouse on a desk*



*A woman standing on a beach holding a surfboard*



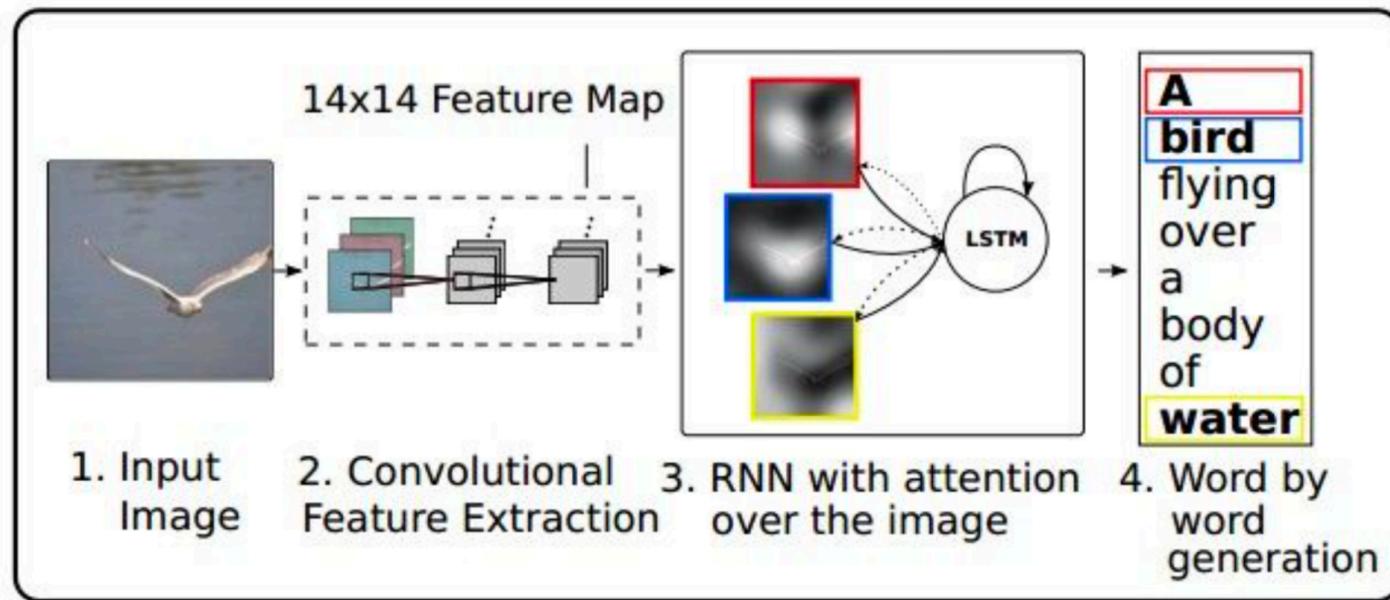
*A bird is perched on a tree branch*



*A man in a baseball uniform throwing a ball*

# Image Captioning with Attention

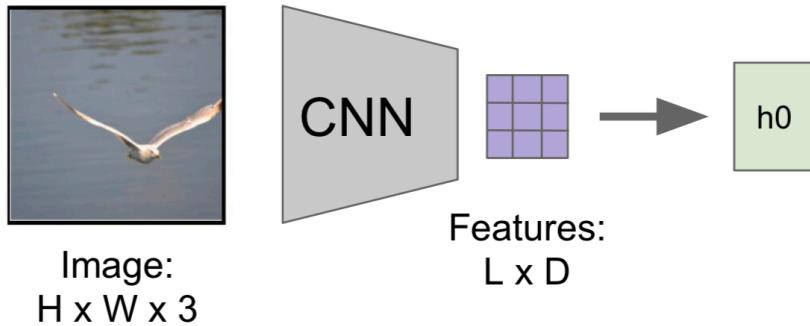
RNN focuses its attention at a different spatial location when generating each word



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

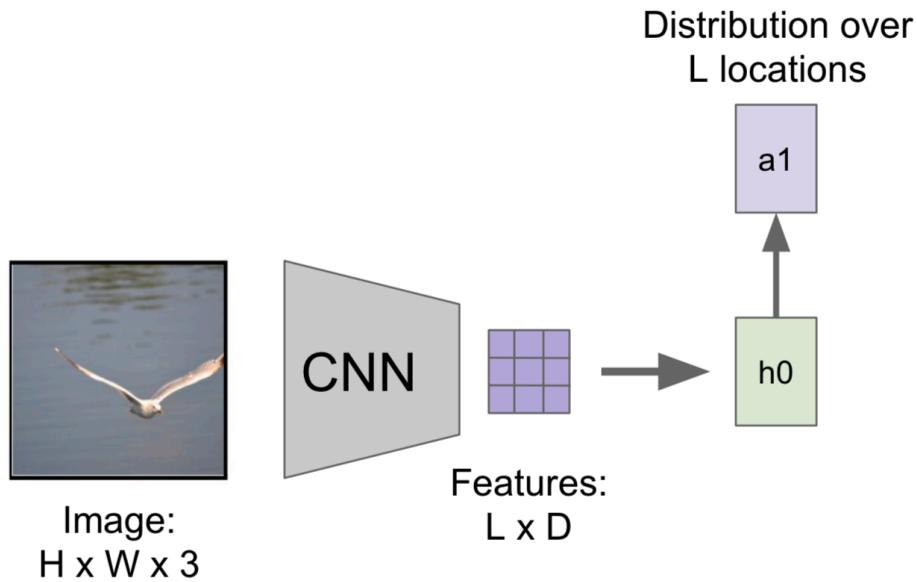
Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

# Image Captioning with Attention



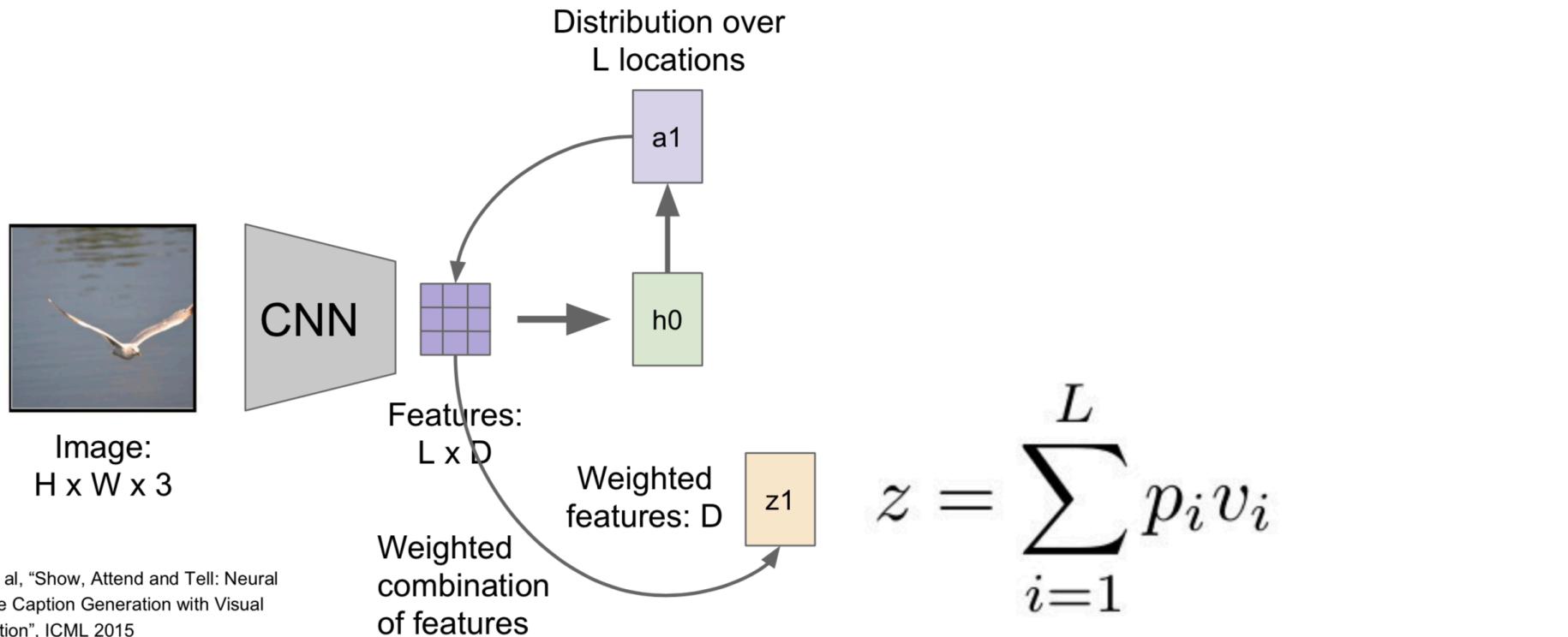
Xu et al, "Show, Attend and Tell: Neural  
Image Caption Generation with Visual  
Attention", ICML 2015

# Image Captioning with Attention



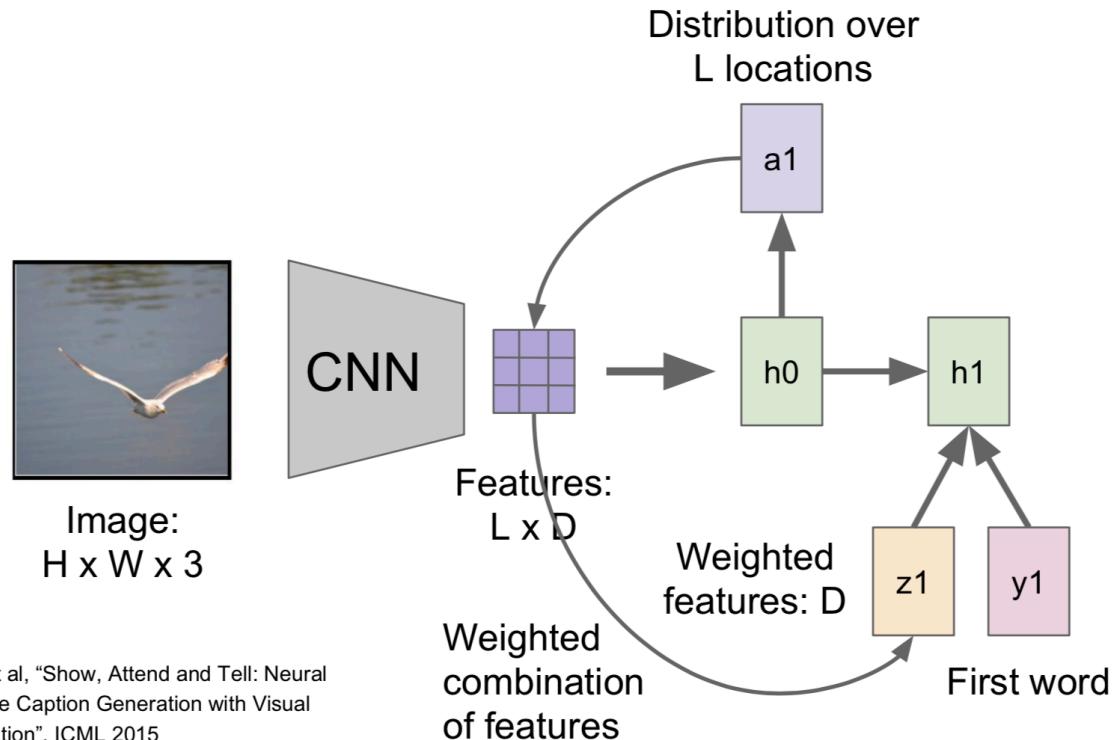
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Image Captioning with Attention

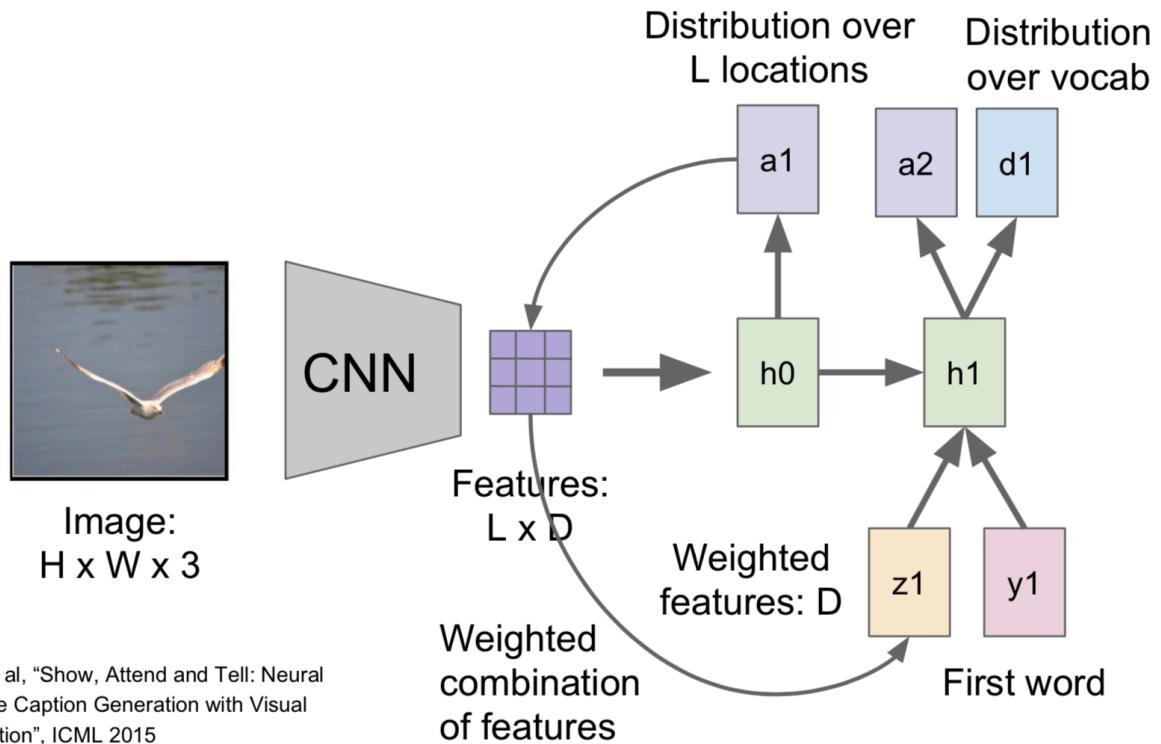


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Image Captioning with Attention



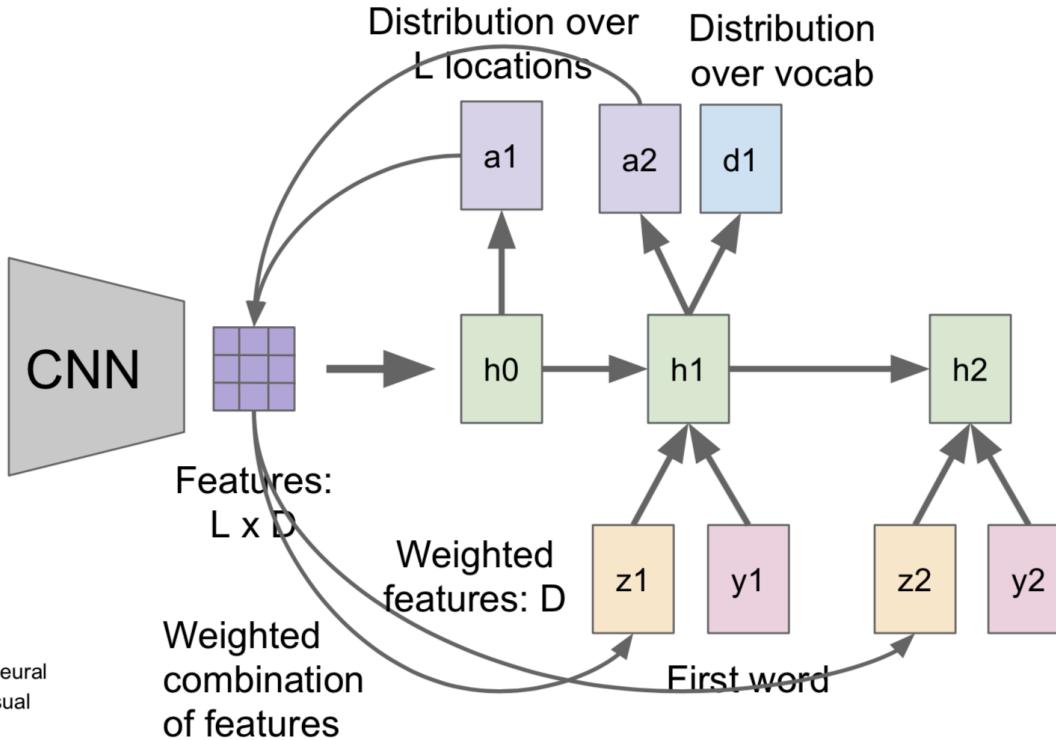
# Image Captioning with Attention



# Image Captioning with Attention

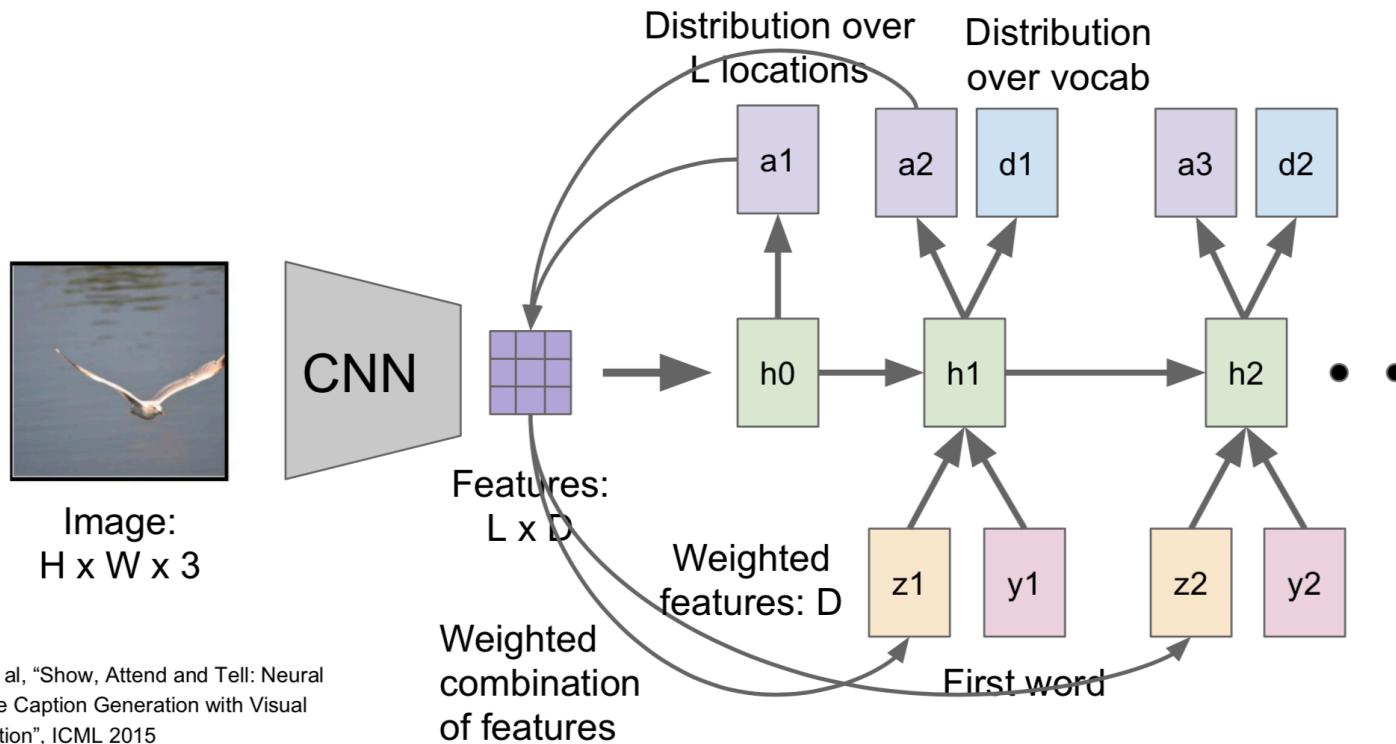


Image:  
 $H \times W \times 3$



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Image Captioning with Attention



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Image Captioning with Attention

Soft attention



Hard attention



A

bird

flying

over

a

body

of

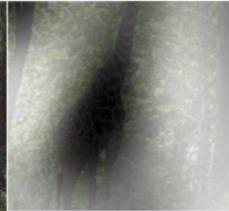
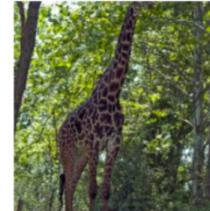
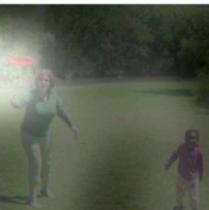
water

.

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

# Image Captioning with Attention



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

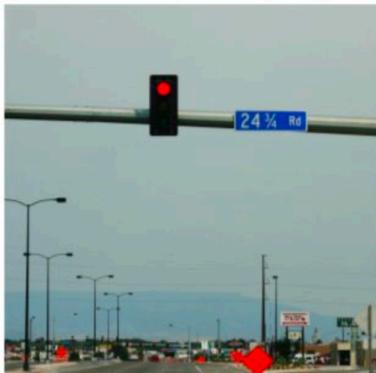
Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

# Visual Question Answering



Q: What endangered animal is featured on the truck?

- A: A bald eagle.
- A: A sparrow.
- A: A humming bird.
- A: A raven.



Q: Where will the driver go if turning right?

- A: Onto 24 ½ Rd.
- A: Onto 25 ¾ Rd.
- A: Onto 23 ¾ Rd.
- A: Onto Main Street.



Q: When was the picture taken?

- A: During a wedding.
- A: During a bar mitzvah.
- A: During a funeral.
- A: During a Sunday church service.



Q: Who is under the umbrella?

- A: Two women.
- A: A child.
- A: An old man.
- A: A husband and a wife.

Agrawal et al, "VQA: Visual Question Answering", ICCV 2015  
Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016  
Figure from Zhu et al, copyright IEEE 2016. Reproduced for educational purposes.

# Multilayer RNNs

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$h \in \mathbb{R}^n \quad W^l [n \times 2n]$$

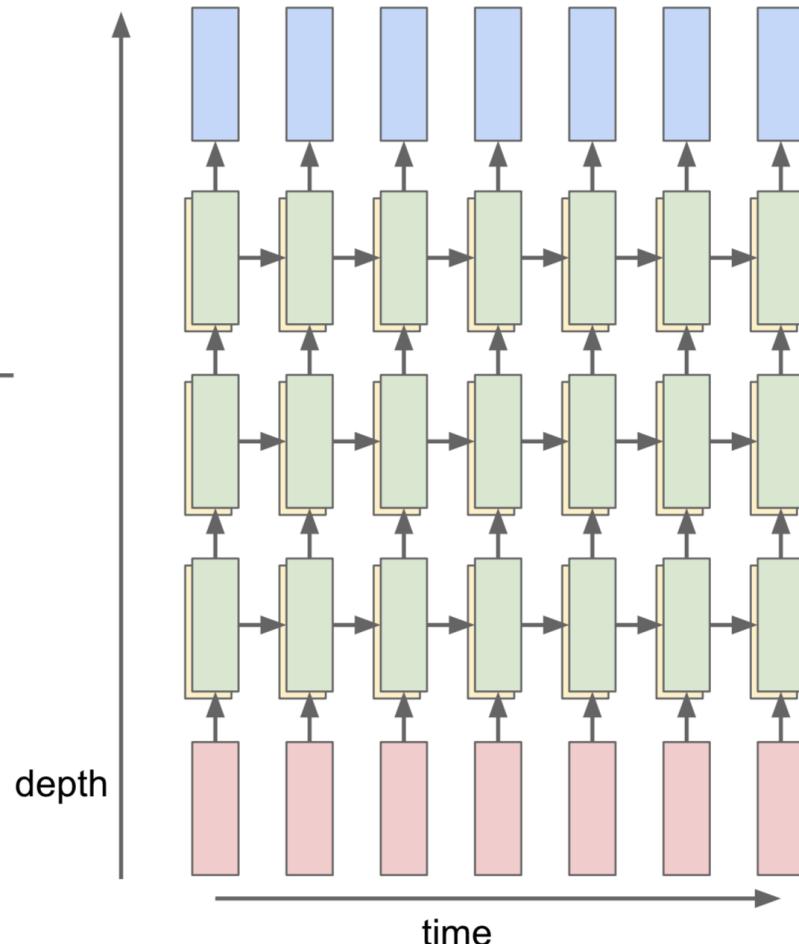
LSTM:

$$W^l [4n \times 2n]$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

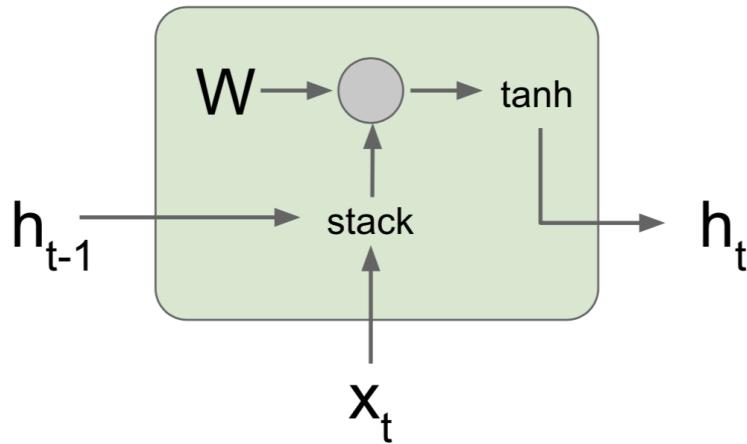
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



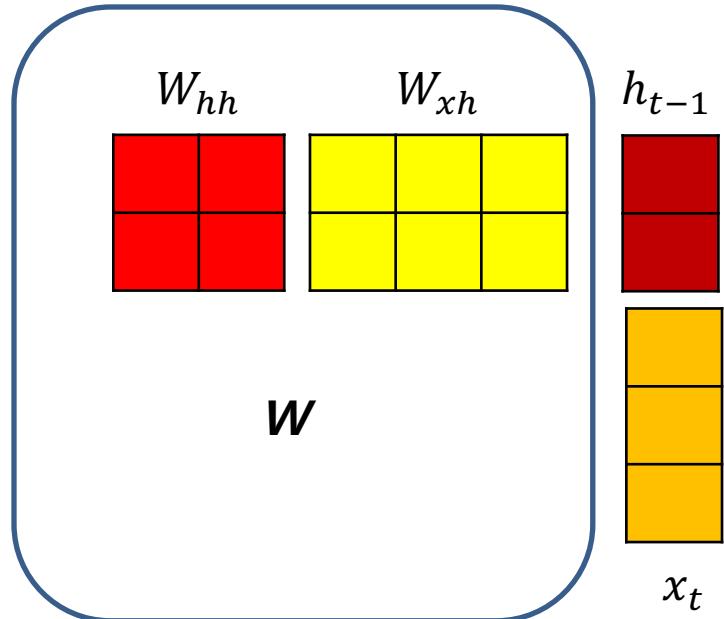
# Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh \left( \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\ &= \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \end{aligned}$$

# Vanilla RNN Gradient Flow



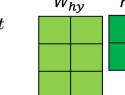
그림으로 대략 그려보면

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

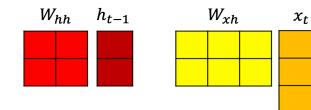


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



- 가정
- Input : (3, 1)
- Hidden : (2, 1)
- Output : (3, 1)

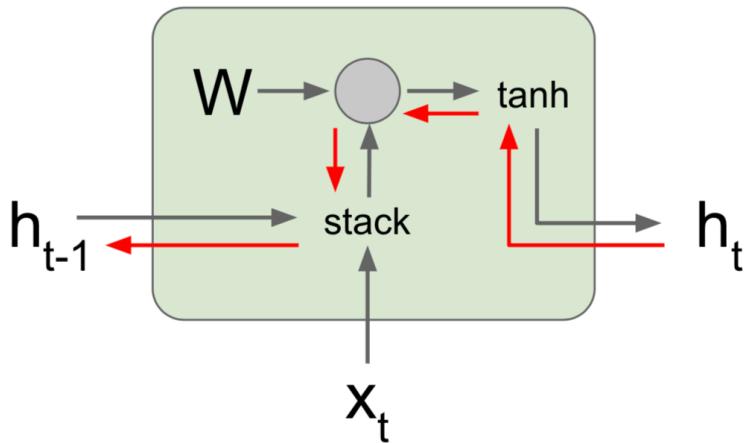


$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh \left( \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\ &= \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \end{aligned}$$

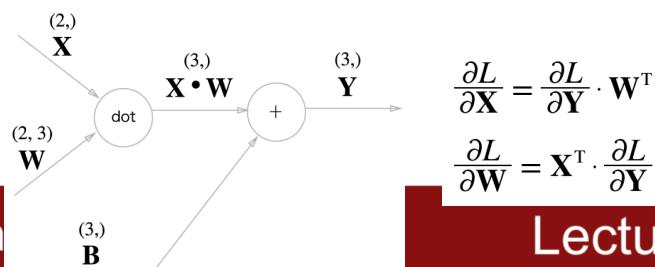
# Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Backpropagation from  $h_t$  to  $h_{t-1}$  multiplies by  $W$  (actually  $W_{hh}^T$ )



$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh \left( \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\ &= \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \end{aligned}$$

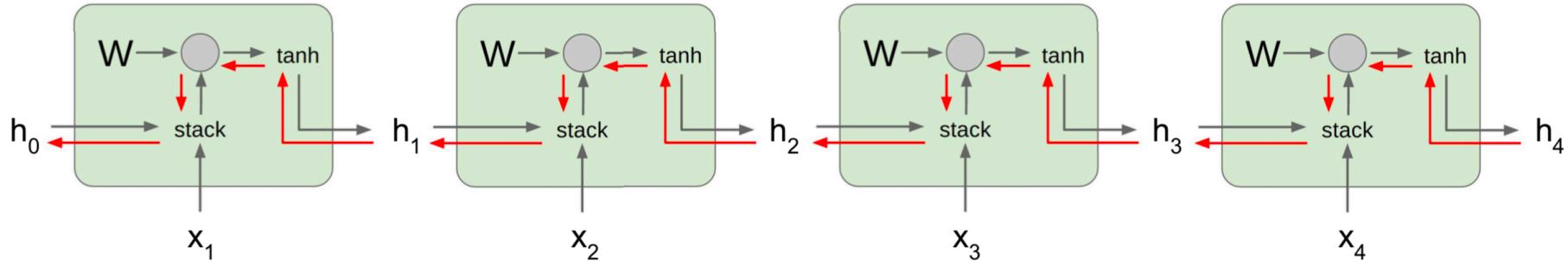


$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^T$$

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^T \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

# Vanilla RNN Gradient Flow

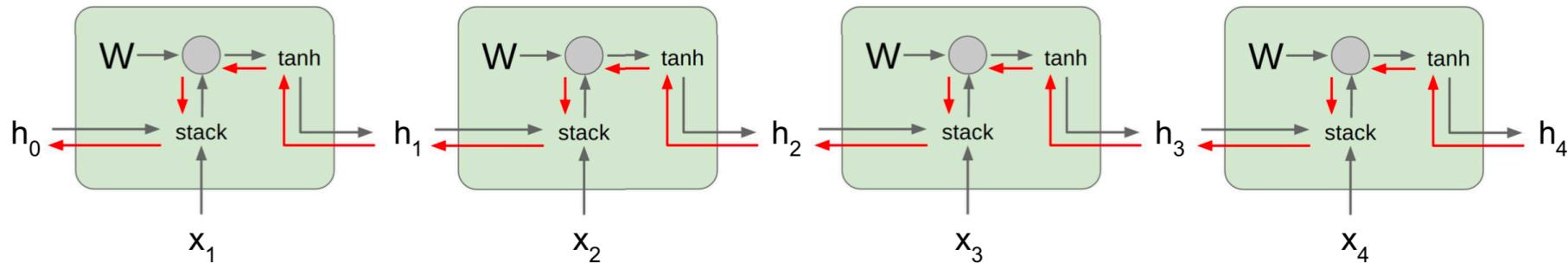
Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient  
of  $h_0$  involves many  
factors of  $W$   
(and repeated tanh)

# Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



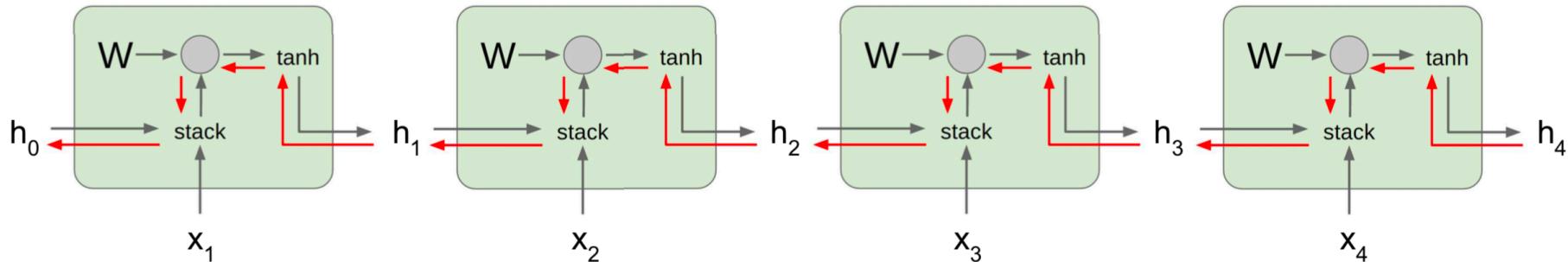
Computing gradient of  $h_0$  involves many factors of  $W$  (and repeated  $\tanh$ )

Largest singular value  $> 1$ :  
**Exploding gradients**

Largest singular value  $< 1$ :  
**Vanishing gradients**

# Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of  $h_0$  involves many factors of  $W$  (and repeated tanh)

Largest singular value  $> 1$ :  
**Exploding gradients**

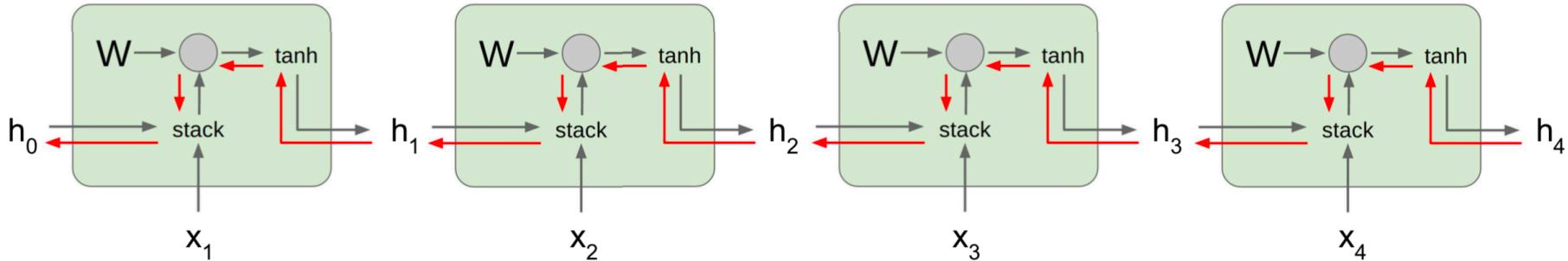
Largest singular value  $< 1$ :  
**Vanishing gradients**

**Gradient clipping:** Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```

# Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of  $h_0$  involves many factors of  $W$  (and repeated tanh)

Largest singular value  $> 1$ :  
**Exploding gradients**

Largest singular value  $< 1$ :  
**Vanishing gradients**

→ Change RNN architecture

# Long Short Term Memory (LSTM)

## Vanilla RNN

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

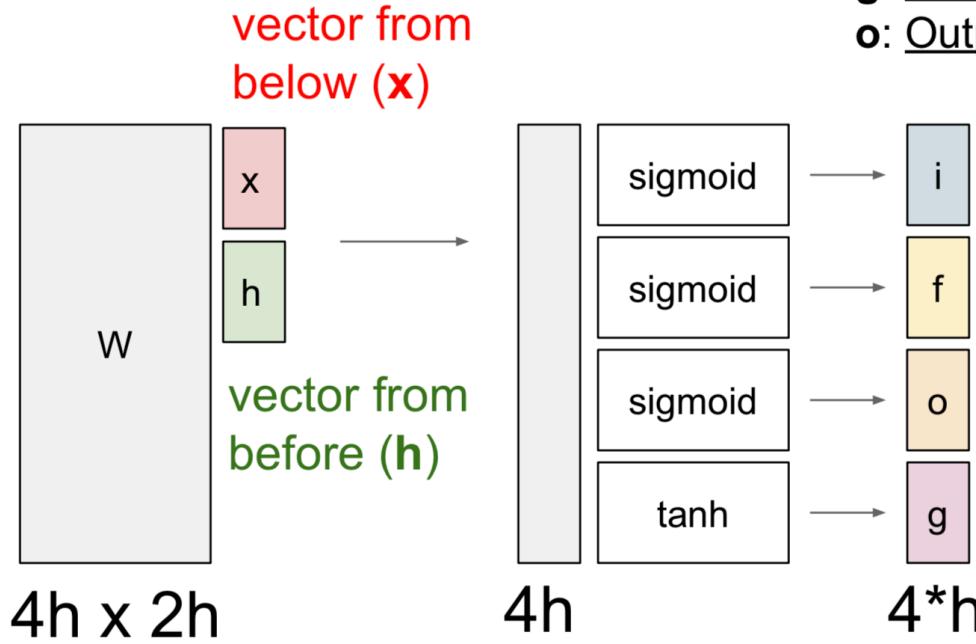
## LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation  
1997

# Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



- f: Forget gate, Whether to erase cell
- i: Input gate, whether to write to cell
- g: Gate gate (?), How much to write to cell
- o: Output gate, How much to reveal cell

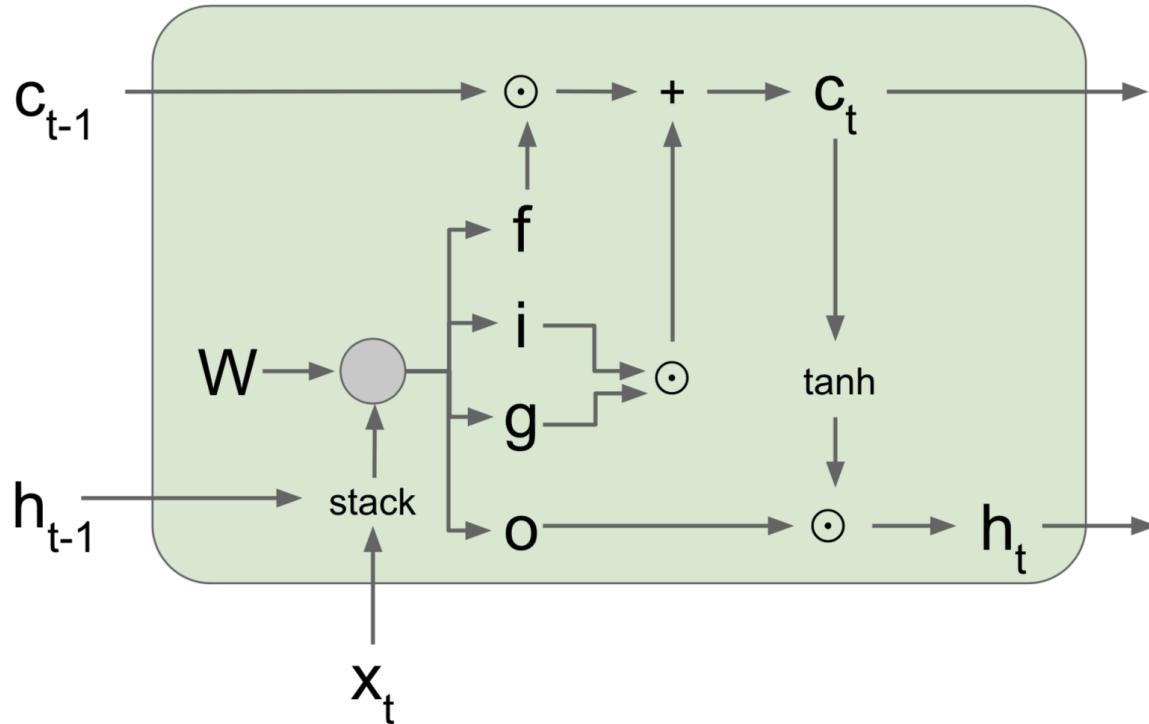
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



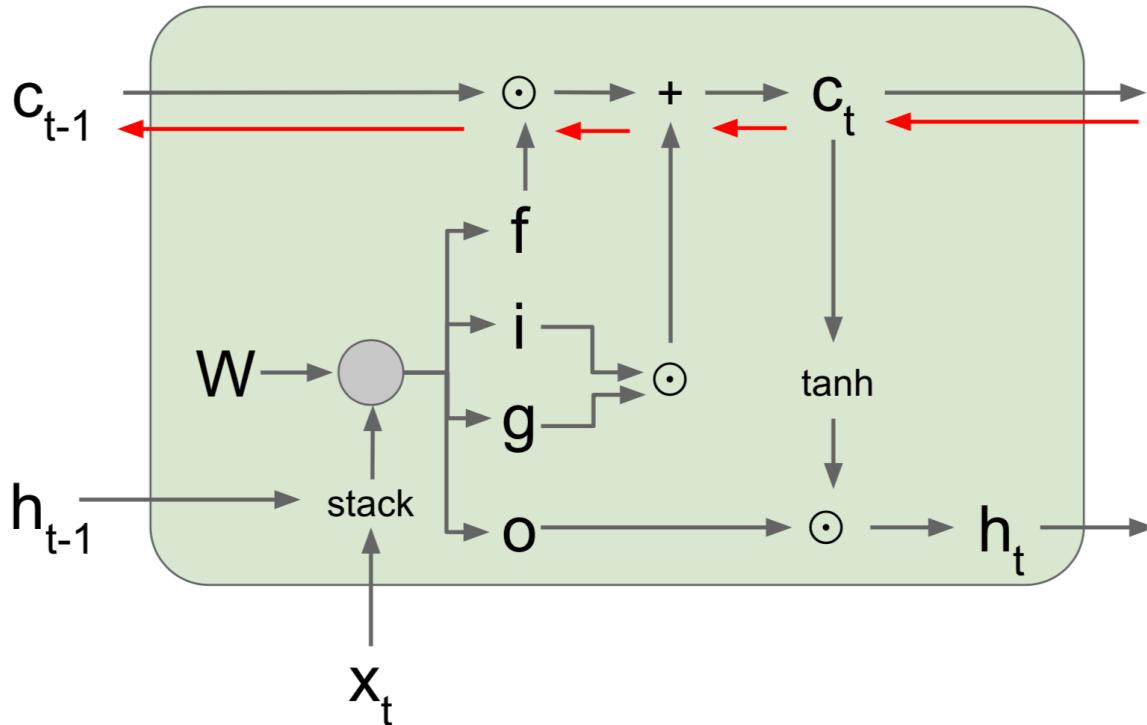
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



Backpropagation from  $c_t$  to  $c_{t-1}$  only elementwise multiplication by  $f$ , no matrix multiply by  $W$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

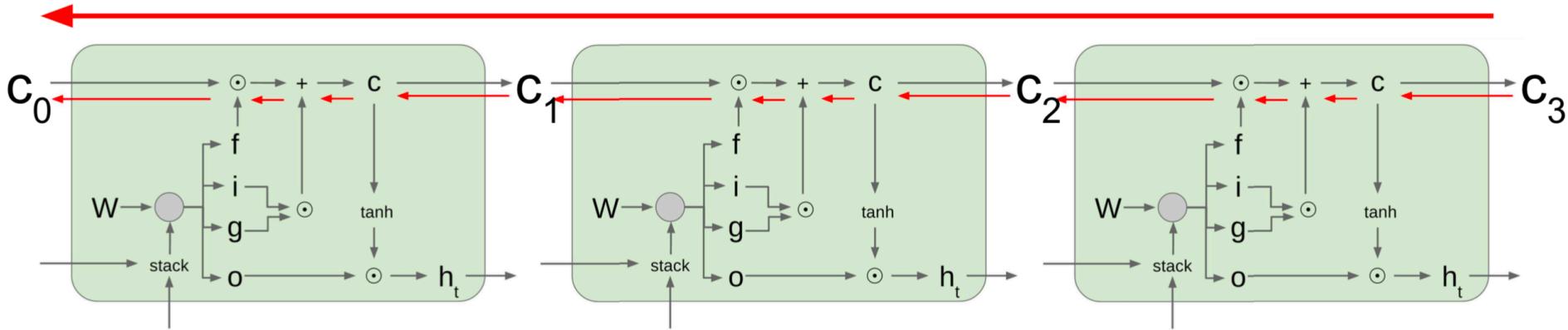
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

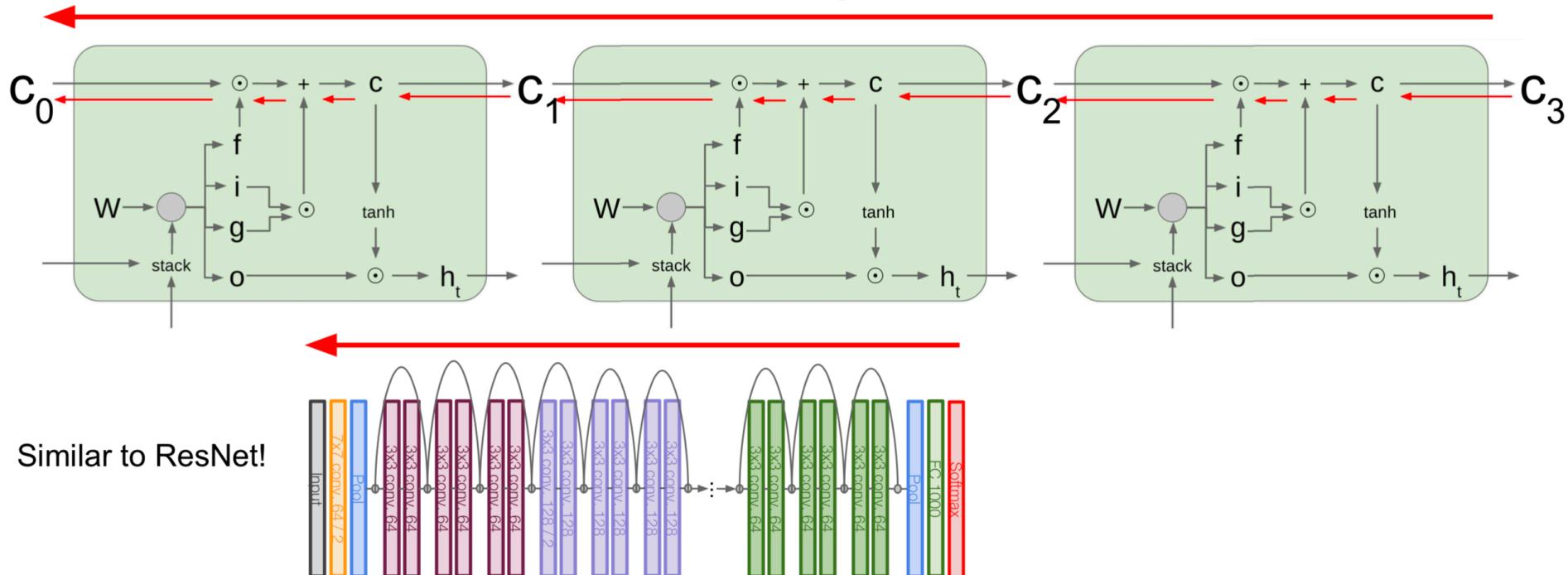
Uninterrupted gradient flow!



# Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

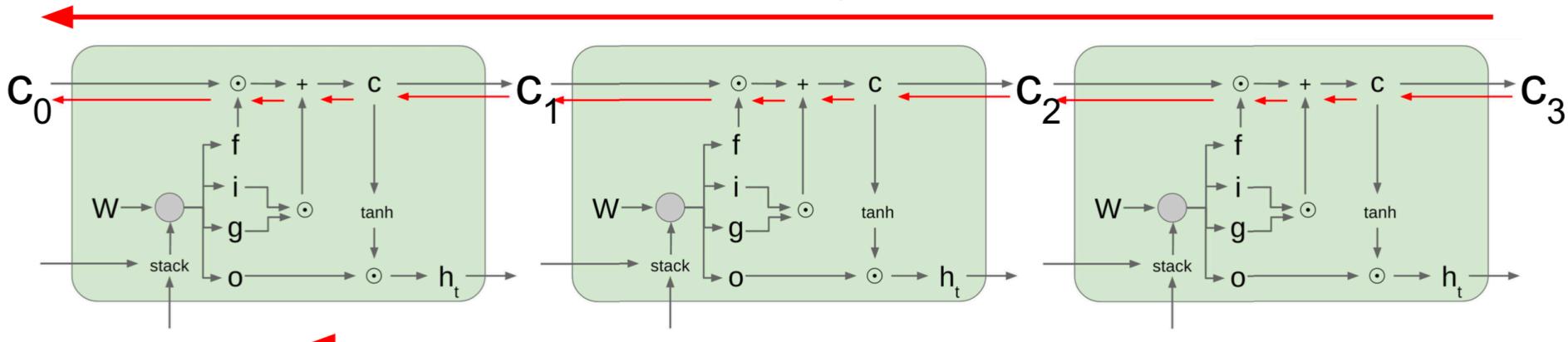
# Uninterrupted gradient flow!



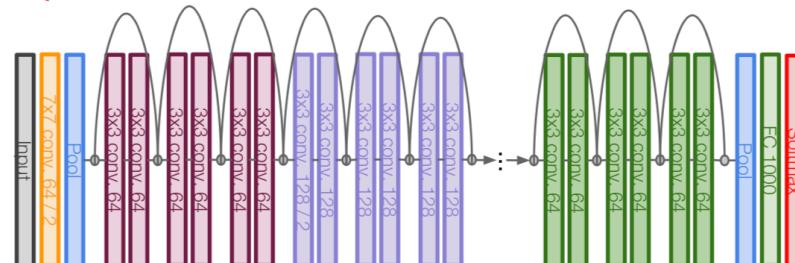
# Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

## Uninterrupted gradient flow!



Similar to ResNet!



In between:  
**Highway Networks**

$$g = T(x, W_T)$$

$$y = g \odot H(x, W_H) + (1 - g) \odot x$$

Srivastava et al, "Highway Networks",  
ICML DL Workshop 2015

# Other RNN Variants

**GRU** [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

[*LSTM: A Search Space Odyssey*, Greff et al., 2015]

[*An Empirical Exploration of Recurrent Network Architectures*, Jozefowicz et al., 2015]

MUT1:

$$z = \text{sigm}(W_{xz}x_t + b_z)$$

$$r = \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r)$$

$$\begin{aligned} h_{t+1} = & \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z \\ & + h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$z = \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z)$$

$$r = \text{sigm}(x_t + W_{hr}h_t + b_r)$$

$$\begin{aligned} h_{t+1} = & \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ & + h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$z = \text{sigm}(W_{xz}x_t + W_{hz}\tanh(h_t) + b_z)$$

$$r = \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r)$$

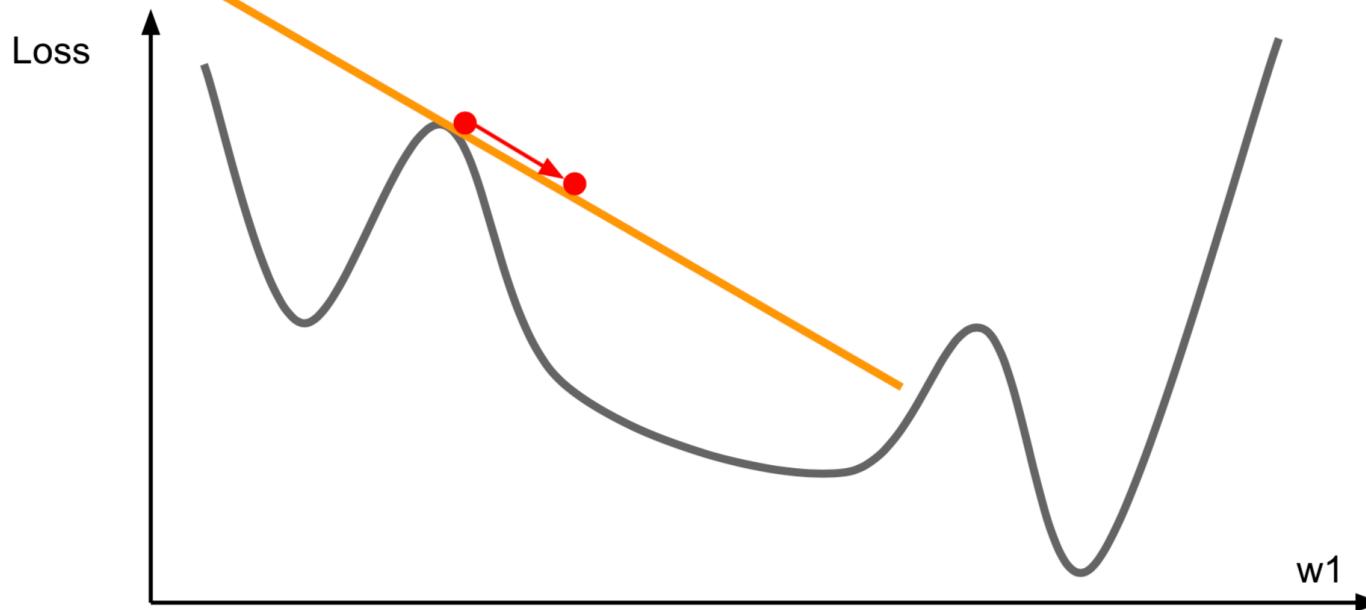
$$\begin{aligned} h_{t+1} = & \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ & + h_t \odot (1 - z) \end{aligned}$$

# Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don't work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed.

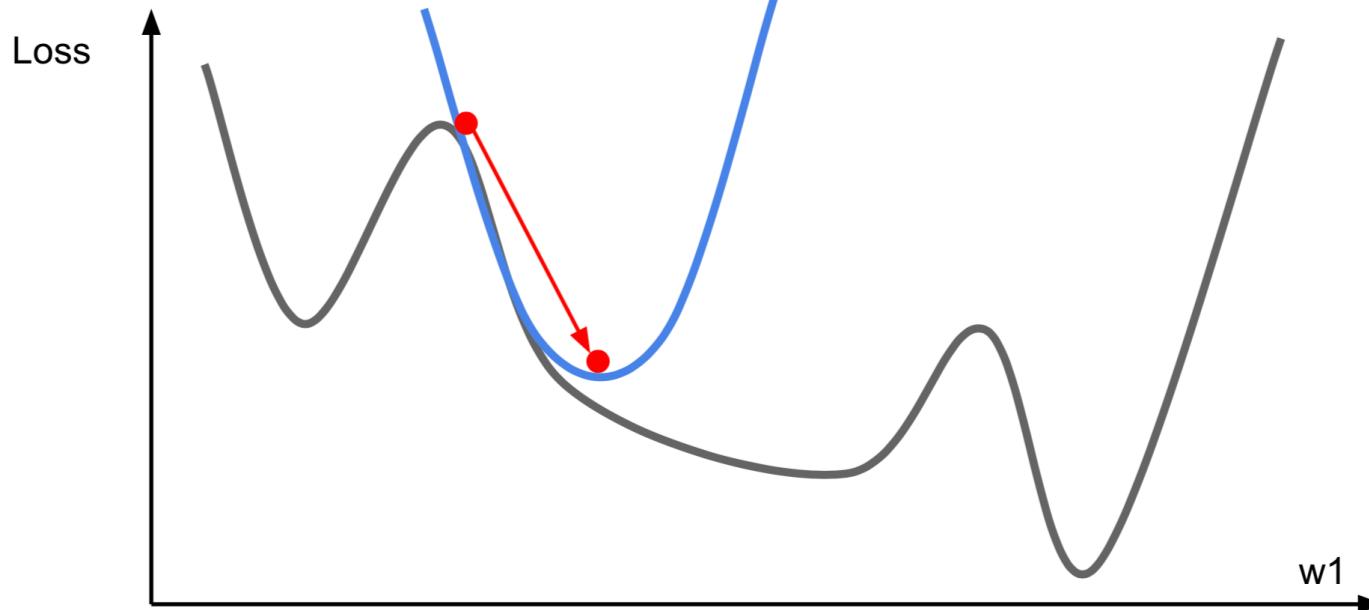
# First-Order Optimization

- (1) Use gradient form linear approximation
- (2) Step to minimize the approximation



# Second-Order Optimization

- (1) Use gradient and Hessian to form **quadratic approximation**
- (2) Step to the **minima** of the approximation



- 링크 : 2차 테일러급수 근사 후 최적점 찾기
  - <https://www.evernote.com/l/AJw-EgWHn01OYZVbOsbyhyiZZifhvATmI4>

# Second-Order Optimization

second-order Taylor expansion:

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Solving for the critical point we obtain the Newton parameter update:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$

Q: What is nice about this update?

# 헤시안

또한 다변수 함수의 이차 미분은 아래와 같은 헤시안(Hessian) 행렬로 정의됩니다.

$$H_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \quad --- (5)$$

# Second-Order Optim

second-order Taylor expansion:

$$\underline{f(x)} = \boxed{f(a)}$$
$$= f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2$$

$$J(\theta) \approx J(\theta_0) + (\theta - \theta_0)^\top \nabla_{\theta} J(\theta_0) + \frac{1}{2} (\theta - \theta_0)^\top H(\theta - \theta_0)$$

Solving for the critical point we obtain the Newton parameter update:

$$\theta^* = \theta_0 - H^{-1} \nabla_{\theta} J(\theta_0)$$

$$\gamma = a - \frac{f(a)}{f''(a)}$$

Q: What is nice about this update?

# Second-Order Optimization

second-order Taylor expansion:

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Solving for the critical point we obtain the Newton parameter update:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$

No hyperparameters!  
No learning rate!

Q: What is nice about this update?

# Second-Order Optimization

second-order Taylor expansion:

$$J(\theta) \approx J(\theta_0) + (\theta - \theta_0)^\top \nabla_{\theta} J(\theta_0) + \frac{1}{2} (\theta - \theta_0)^\top H(\theta - \theta_0)$$

Solving for the critical point we obtain the Newton parameter update:

$$\theta^* = \theta_0 - H^{-1} \nabla_{\theta} J(\theta_0)$$

Hessian has  $O(N^2)$  elements  
Inverting takes  $O(N^3)$   
 $N = (\text{Tens or Hundreds of}) \text{ Millions}$

Q2: Why is this bad for deep learning?

# Second-Order Optimization

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$

- Quasi-Newton methods (**BGFS** most popular):  
*instead of inverting the Hessian ( $O(n^3)$ ), approximate inverse Hessian with rank 1 updates over time ( $O(n^2)$  each).*
- **L-BFGS** (Limited memory BFGS):  
*Does not form/store the full inverse Hessian.*

# L-BFGS

- **Usually works very well in full batch, deterministic mode**  
i.e. if you have a single, deterministic  $f(x)$  then L-BFGS will probably work very nicely
- **Does not transfer very well to mini-batch setting.** Gives bad results. Adapting L-BFGS to large-scale, stochastic setting is an active area of research.



박 은 수 Research Director

E-mail : es.park@modulabs.co.kr