Deep Learning

Mayank Vatsa

ADAM

For each Parameter w^j

(*j subscript dropped for clarity*)

$$\nu_t = \beta_1 * \nu_{t-1} + (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} + (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*

g_t : *Gradient at time t along* ω^j

ν_t : *Exponential Average of gradients along* ω_j

s_t : *Exponential Average of squares of gradients along* ω_j

β_1, β_2 : *Hyperparameters*

ADAM

$$For\ each\ Parameter\ w^j$$

(j subscript dropped for clarity)

$$\nu_t = \beta_1 * \nu_{t-1} + (1 - \beta_1) * g_t$$

Exponential Weighted Averages for past gradients

$$s_t = \beta_2 * s_{t-1} + (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*

g_t : *Gradient at time t along* ω^j

ν_t : *Exponential Average of gradients along* ω_j

s_t : *Exponential Average of squares of gradients along* ω_j

β_1, β_2 : *Hyperparameters*

ADAM

$$For\ each\ Parameter\ w^j$$

(j subscript dropped for clarity)

$$\nu_t = \beta_1 * \nu_{t-1} + (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} + (1 - \beta_2) * g_t^2$$

Exponential Weighted Averages for past squared gradients
- more weight to recent gradients

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

$\eta : Initial\ Learning\ rate$

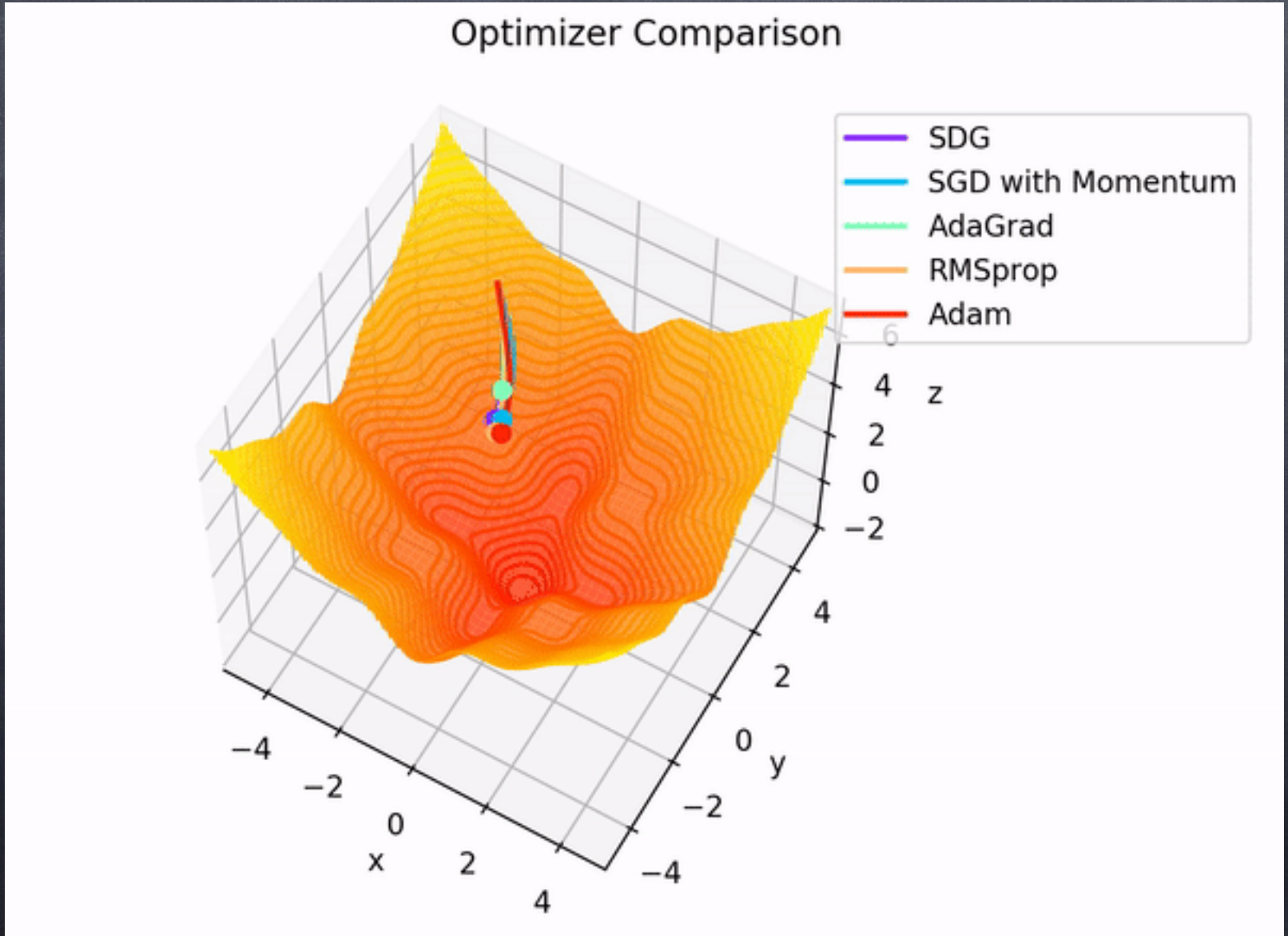
$g_t : Gradient\ at\ time\ t\ along\ \omega^j$

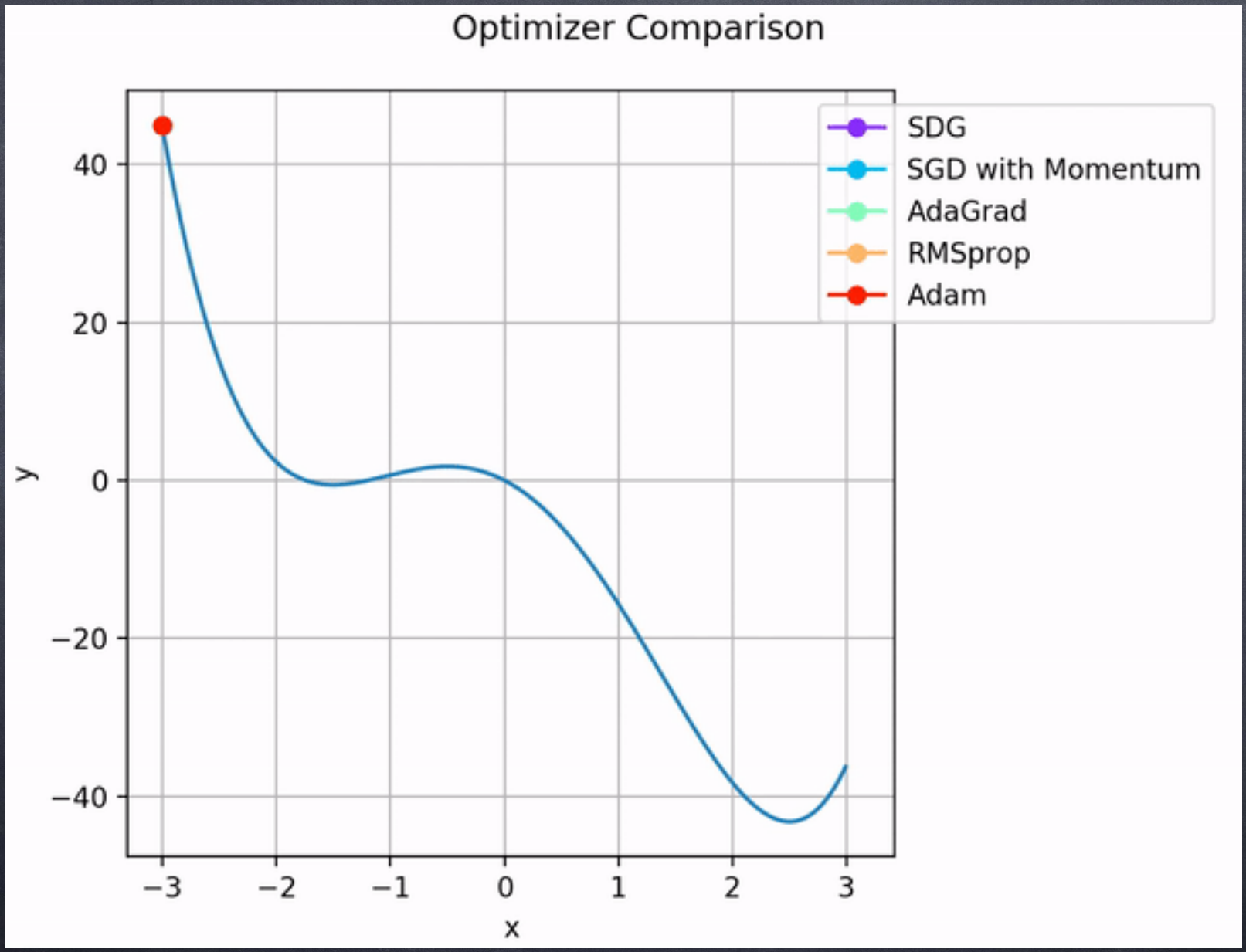
$\nu_t : Exponential\ Average\ of\ gradients\ along\ \omega_j$

$s_t : Exponential\ Average\ of\ squares\ of\ gradients\ along\ \omega_j$

$\beta_1, \beta_2 : Hyperparameters$

ADAM


Optimizer Comparison
SDG
SGD with Momentum
AdaGrad
RMSprop
Adam
z
6
4
2
0
−2
x
−4
−2
0
2
4
y
4
2
0
−2
−4

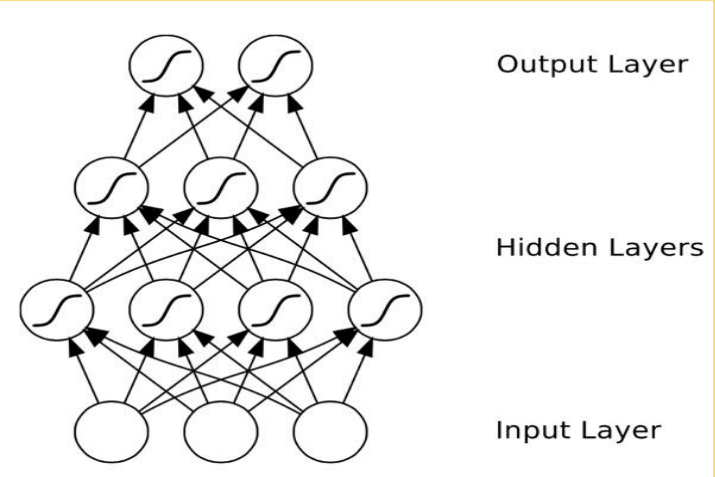

Optimizer Comparison
SDG
SGD with Momentum
AdaGrad
RMSprop
Adam
40
20
0
−20
−40
y
−3
−2
−1
0
1
2
3
x

Designing CNN for a given problem

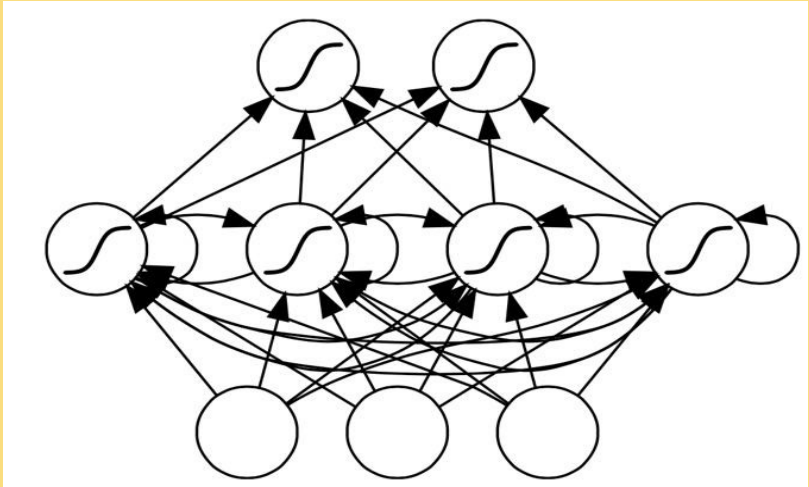
- Image/data size: for example 28X28
- Overall data size: n (50k)
- Design constrains
  - Number of layers (conv, FC, etc)
  - Loss constrains (problem specific)
- Regularization(s) and any other constrains

RNN: Another Class of Neural Network

RNN

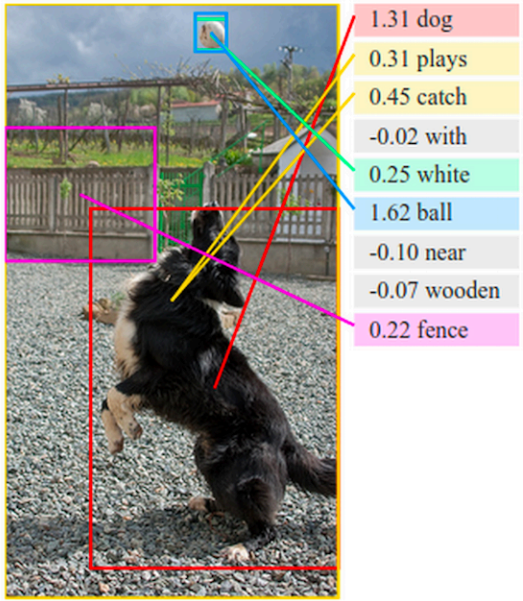


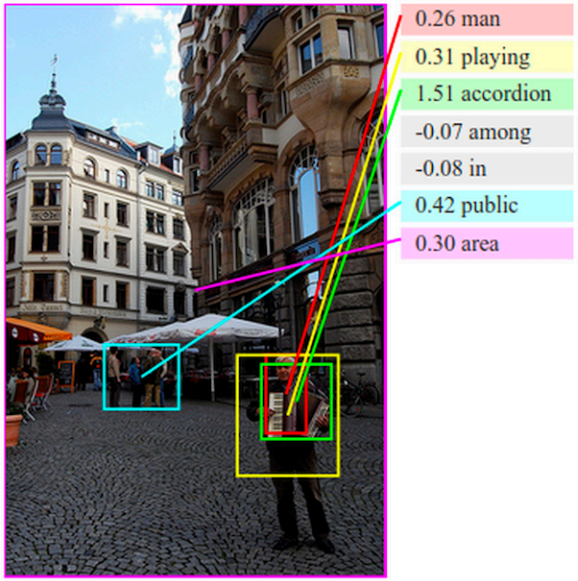
Multi-layer Perceptron

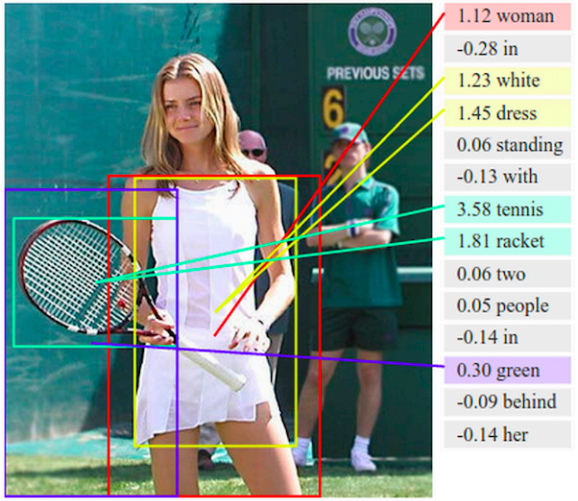


Recurrent Network

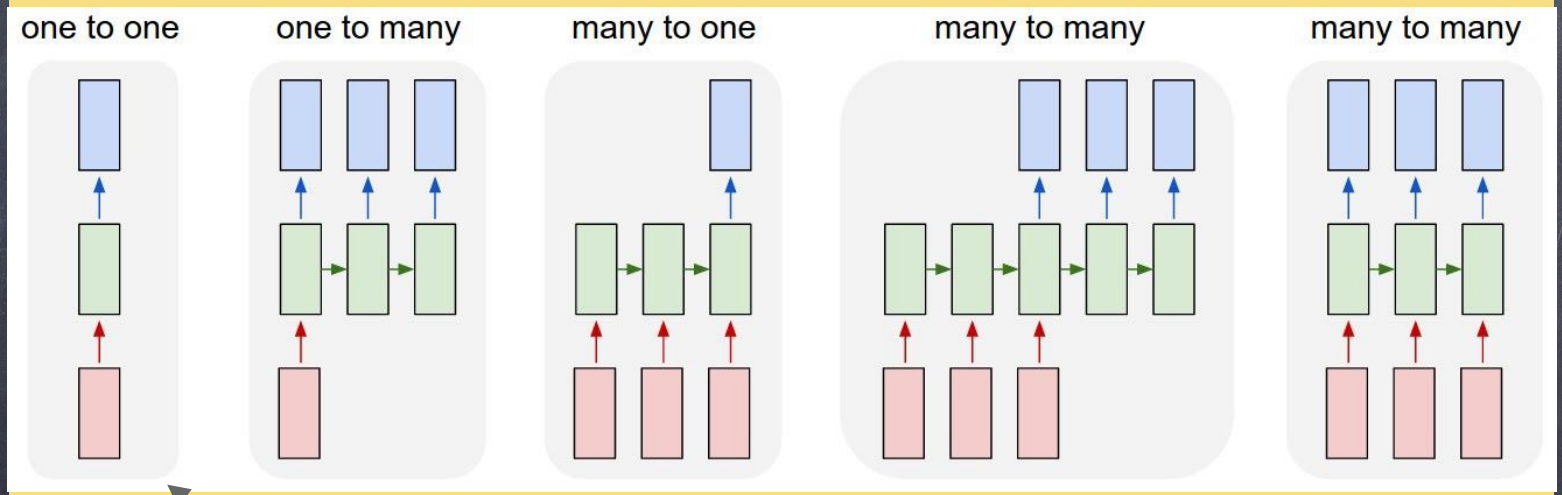
Sequence Modelling





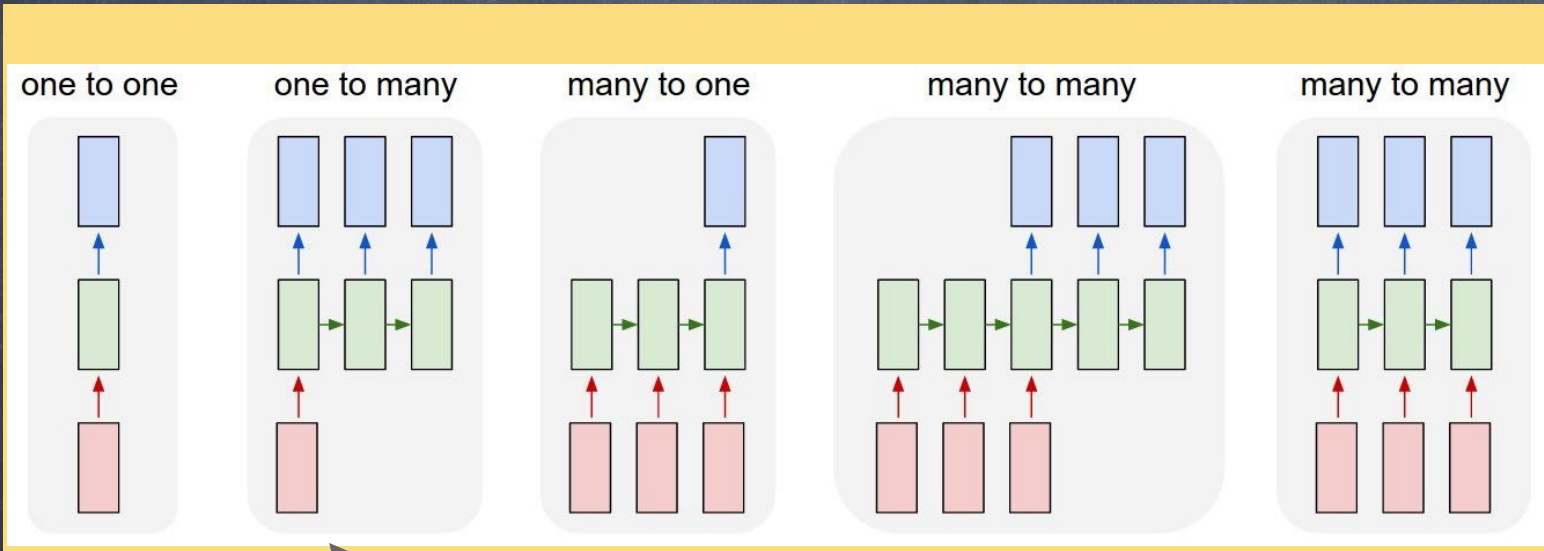


RNN



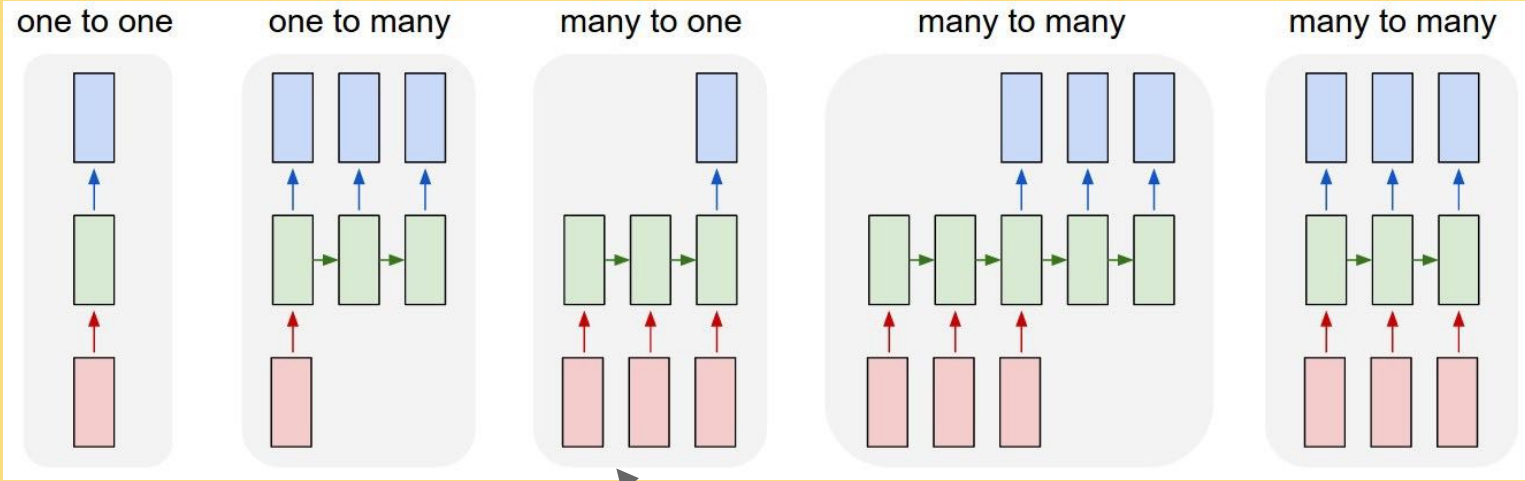
Vanilla Neural Networks

RNN



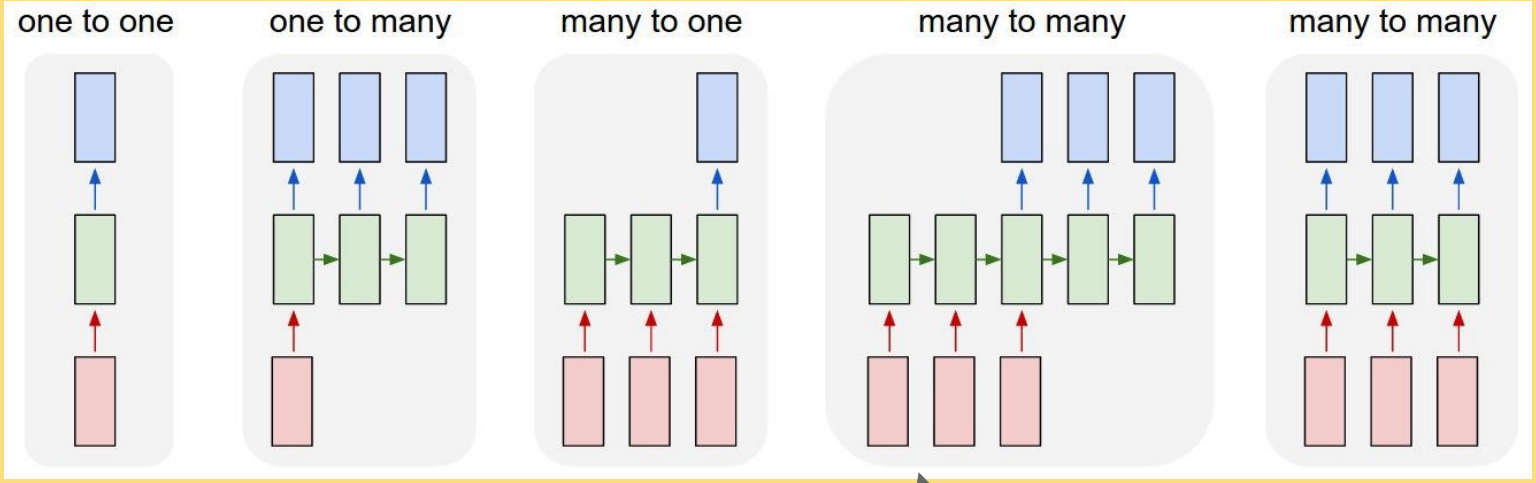
e.g. **Image Captioning**
image -> sequence of words

RNN



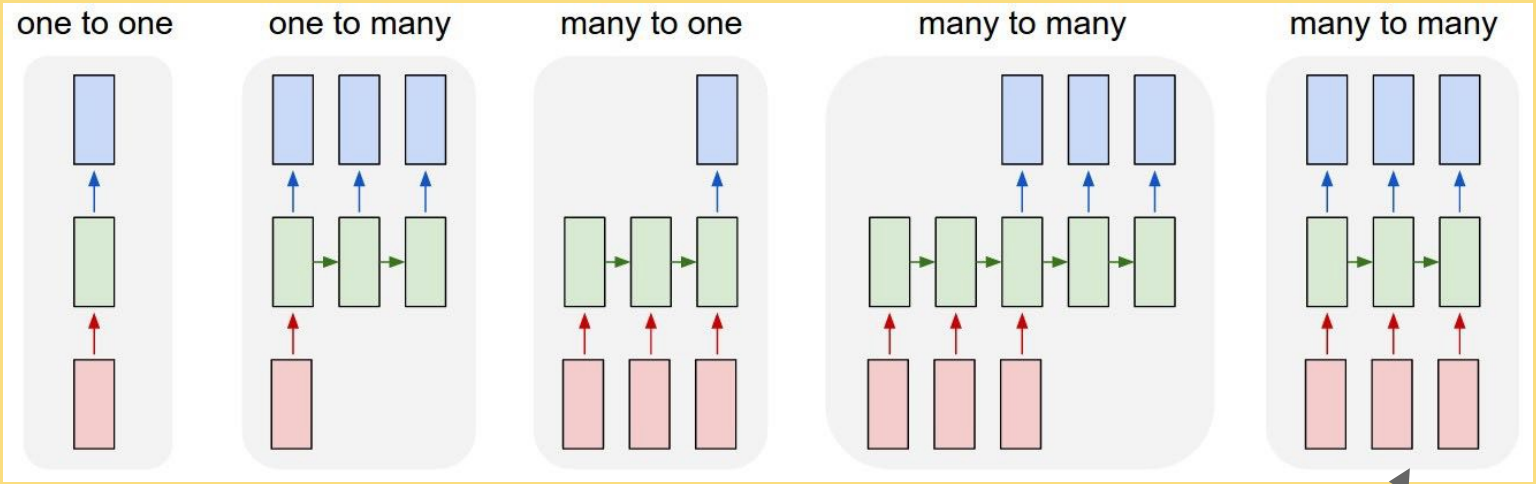
e.g. **Sentiment Classification**
sequence of words -> sentiment

RNN



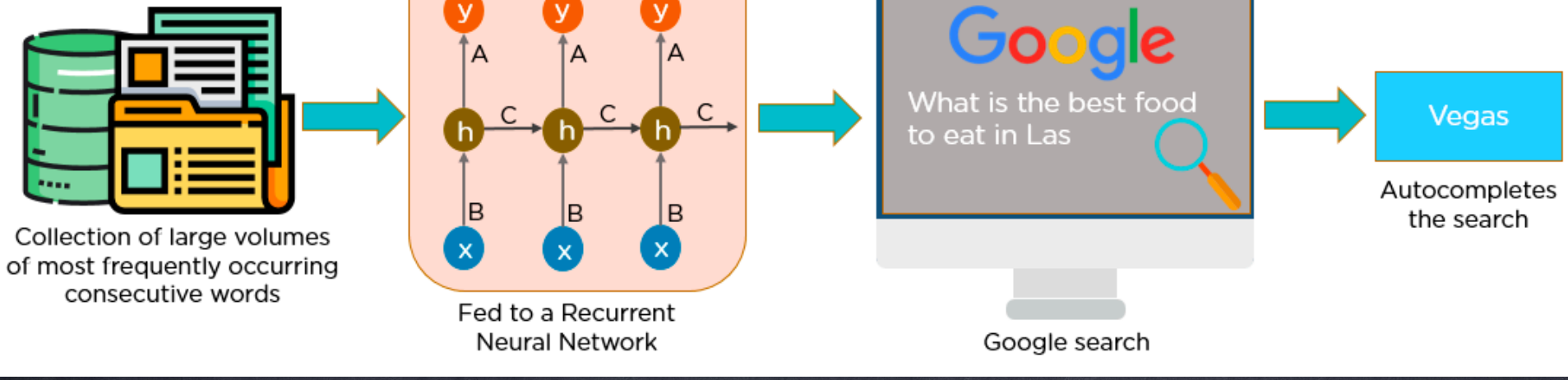
e.g. **Machine Translation**
seq of words -> seq of words

RNN

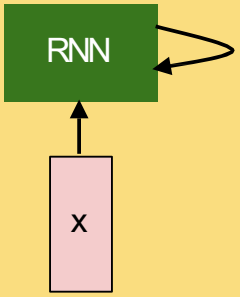


e.g. **Video classification on frame level**

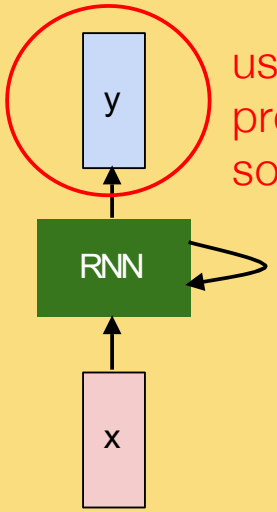
Example


Collection of large volumes of most frequently occurring consecutive words
y
A
h
C
B
x
Fed to a Recurrent Neural Network
Google
What is the best food to eat in Las
Google search
Vegas
Autocompletes the search

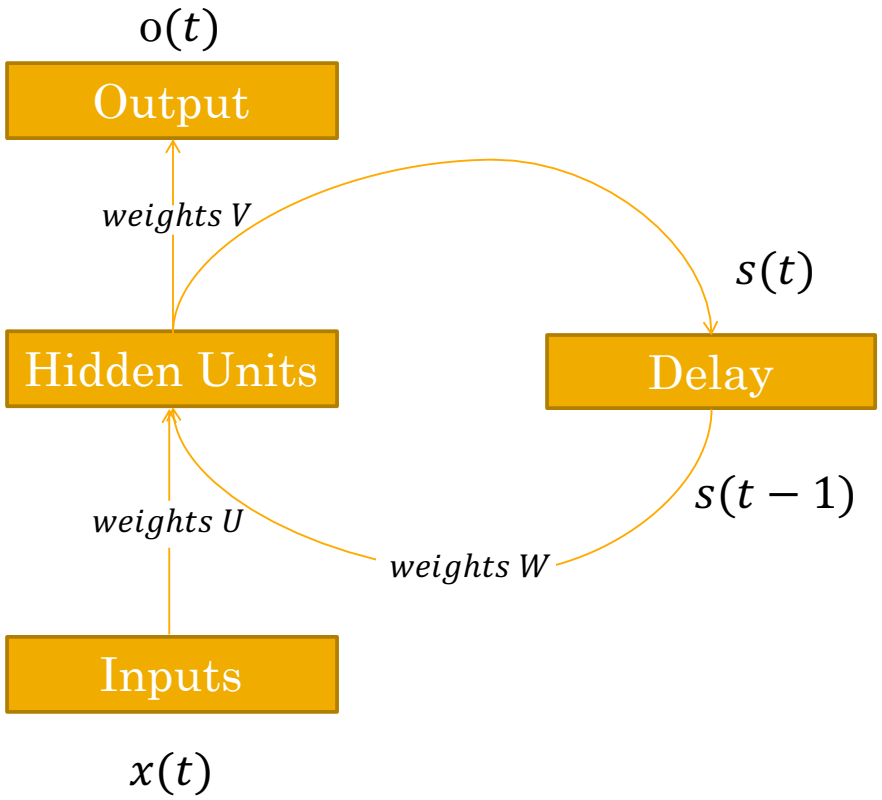
RNN: Basics



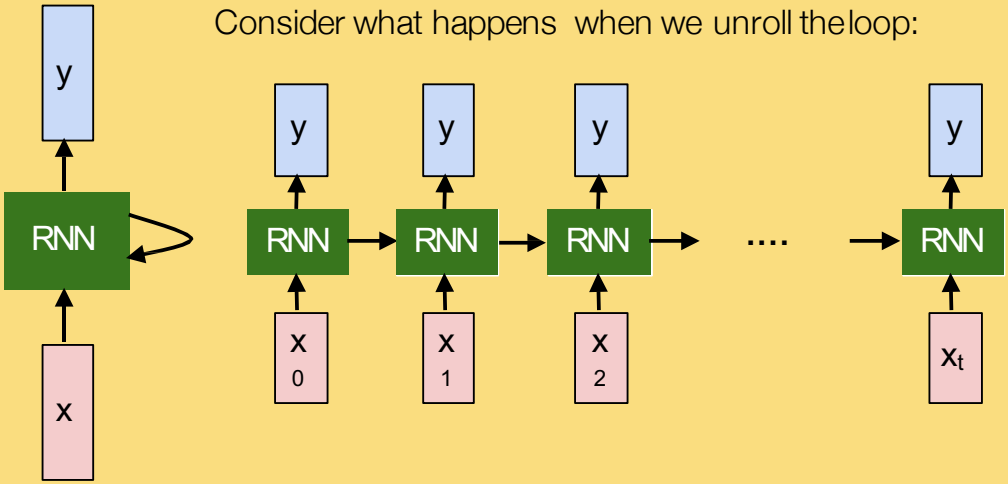
RNN: Basics



usually want to predict a vector at some time steps


o(t)
Output
weights V
s(t)
Hidden Units
Delay
s(t − 1)
weights U
weights W
Inputs
x(t)

RNN: Basics

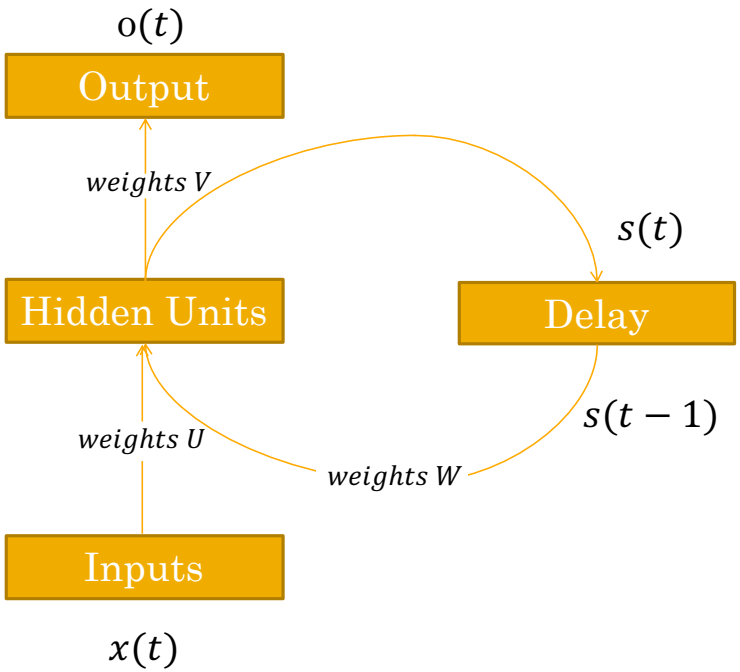

Consider what happens when we unroll the loop:
y
RNN
x
y
RNN
x
0
y
RNN
x
1
y
RNN
x
2
....
y
RNN
x
t

- The network input at time t:

$$a_h(t) = Ux(t) + Ws(t-1)$$

- The activation of the input at time t:

$$s(t) = f_h\left(a_h(t)\right)$$



- The network input to the output unit at time t:

$$a_o(t) = Vs(t)$$

- The output of the network at time t is:

$$o(t) = f_o\left(a_o(t)\right)$$

RNN: Basics

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

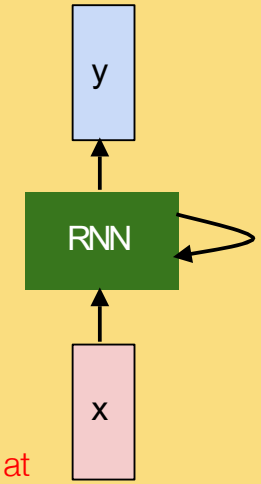
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

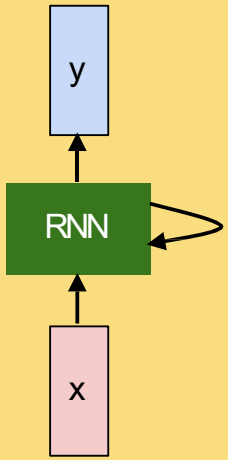
input vector at some time step



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

RNN: Basics

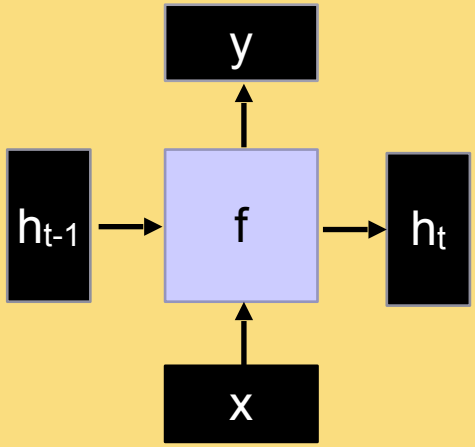
The state consists of a single *"hidden"* vector **h**:



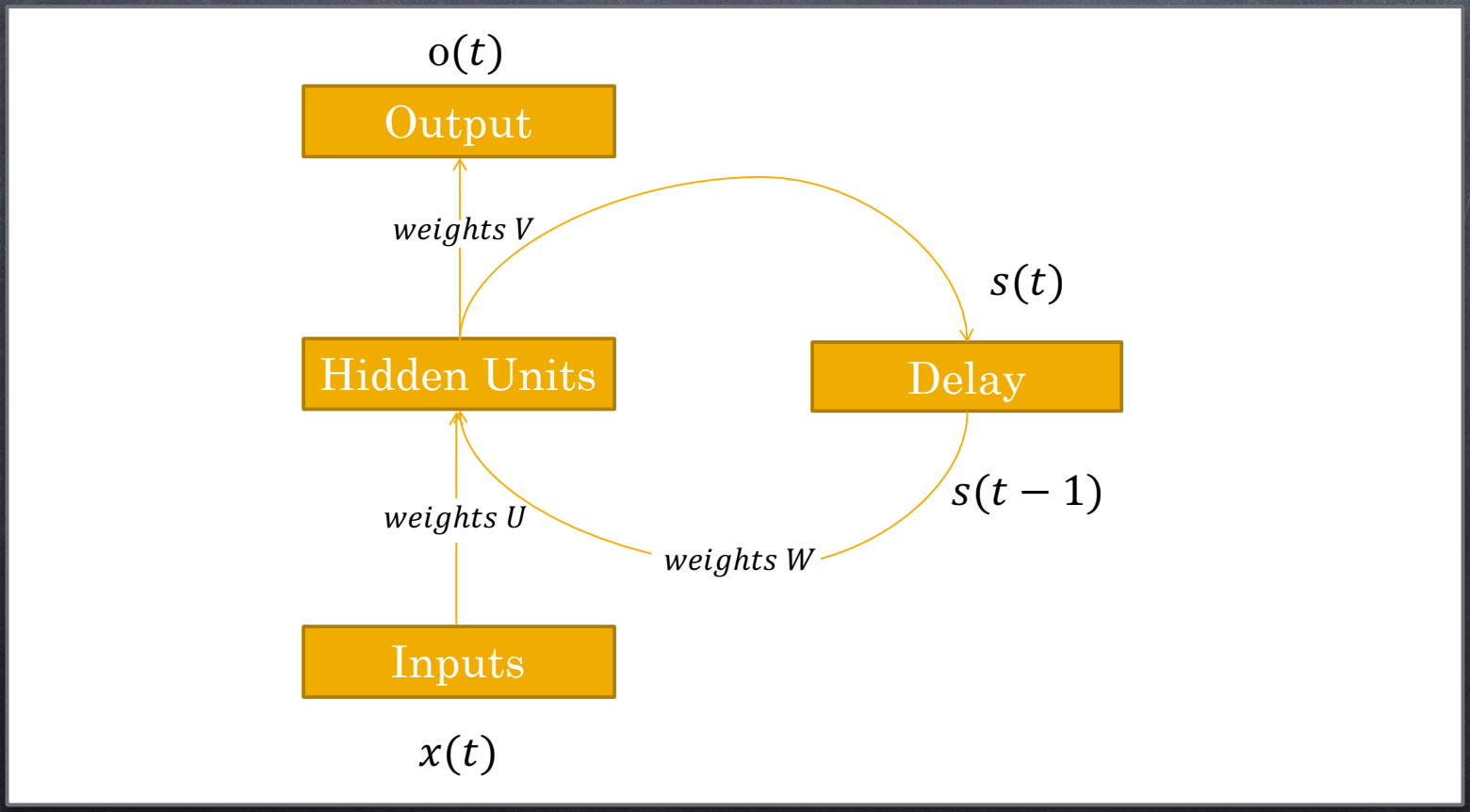
$$h_t = f_W(h_{t-1}, x_t)$$

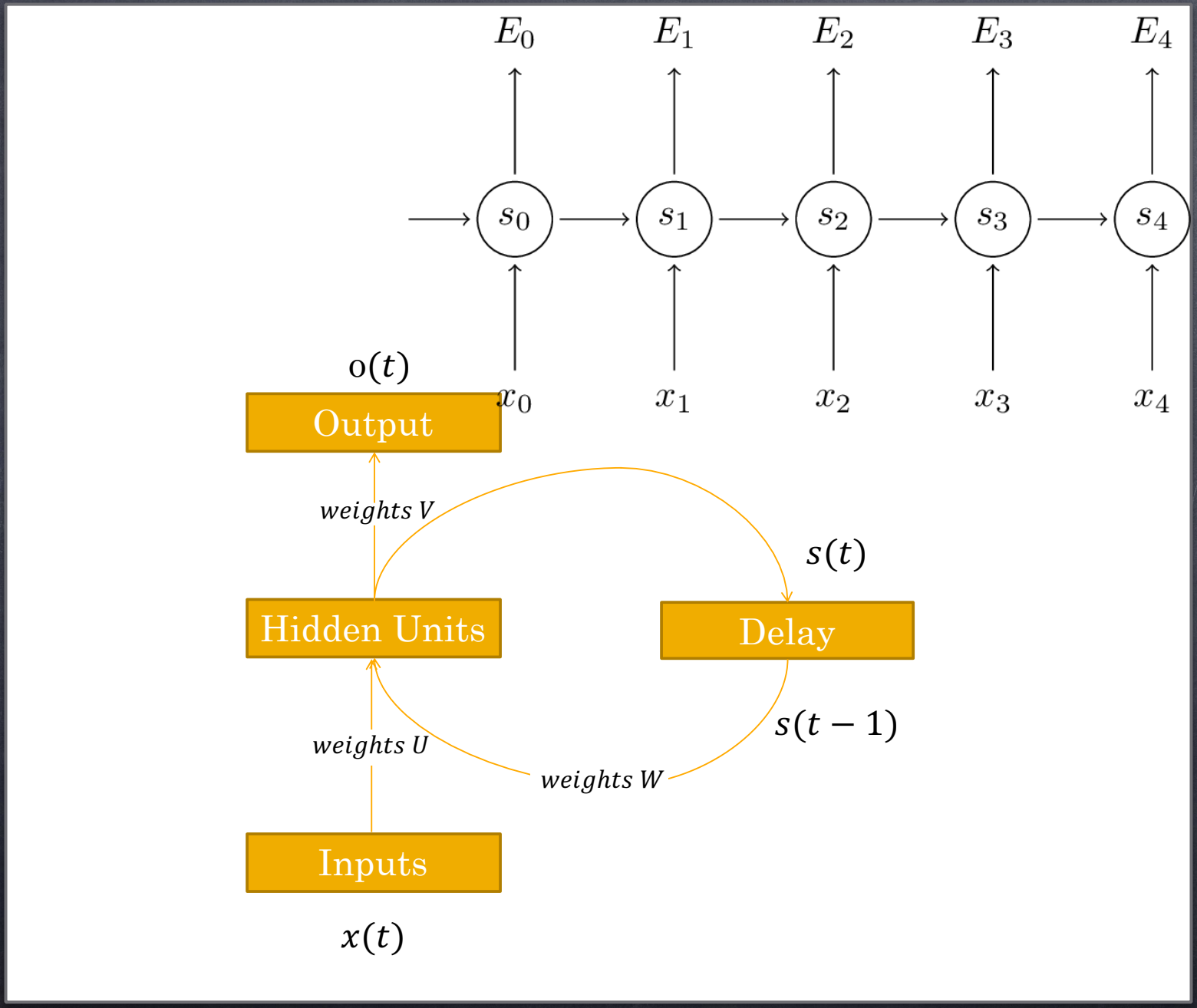
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

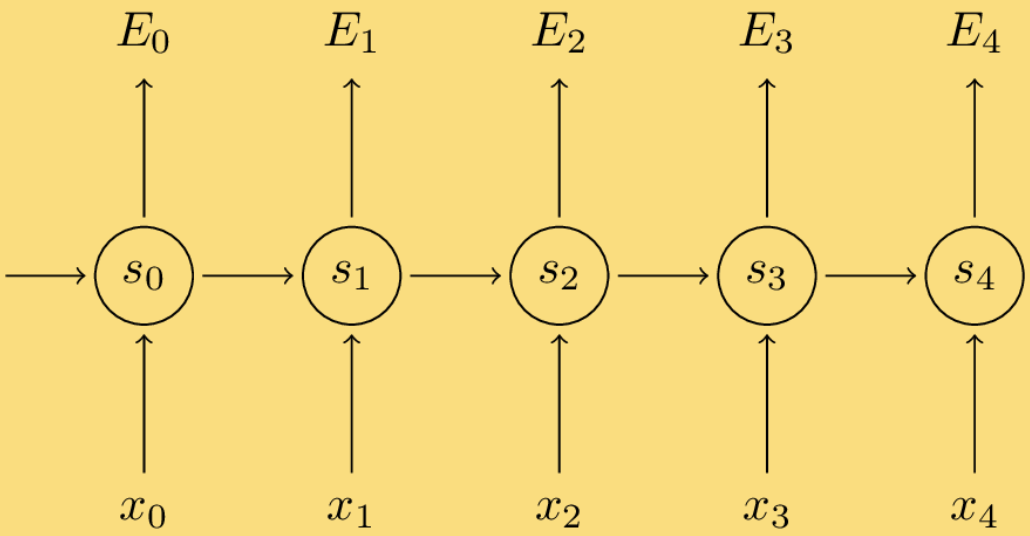
$$y_t = W_{hy}h_t$$

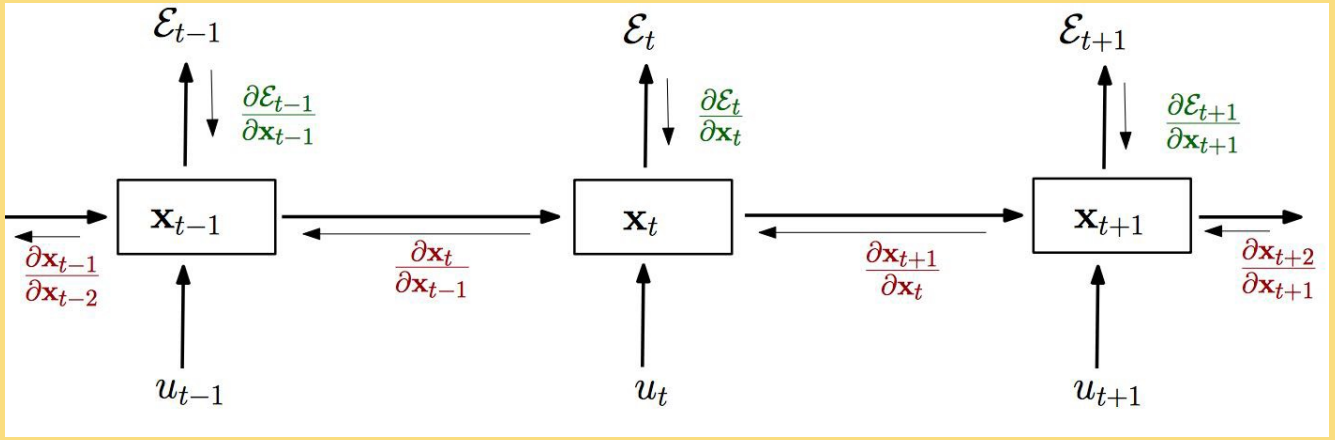


What are the sources of error backpropagation




E_0
E_1
E_2
E_3
E_4
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4
o(t)
Output
weights V
s(t)
Hidden Units
Delay
s(t − 1)
weights U
weights W
Inputs
x(t)

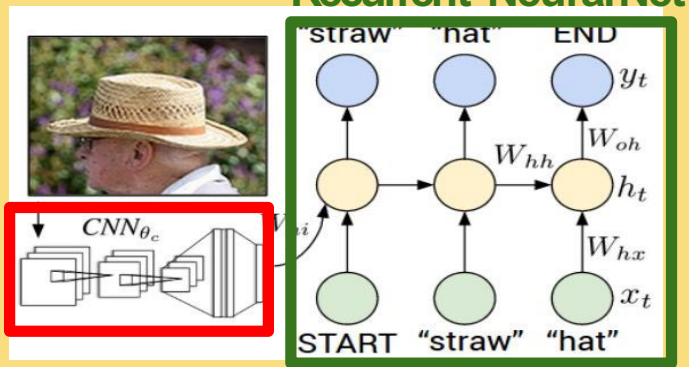




Backpropagation through time (BPTT)

RNN: Example

Image Captioning



test image

Slide credit: Andrej Karpathy

Slide credit: Andrej Karpathy


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
Slide credit: Andrej Karpathy

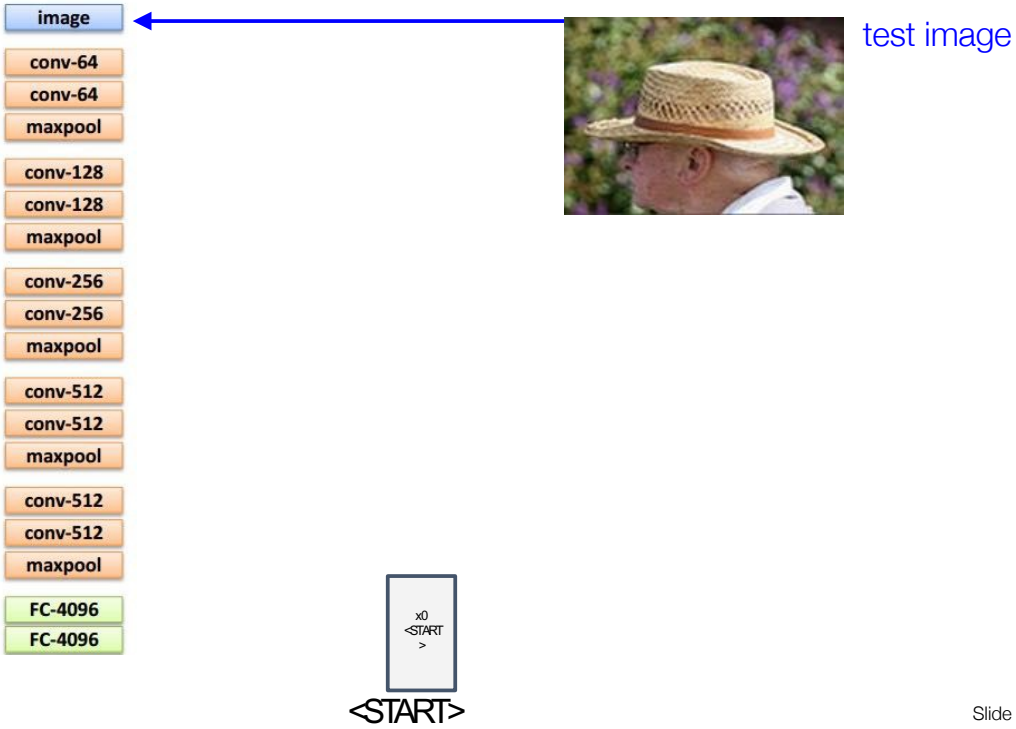
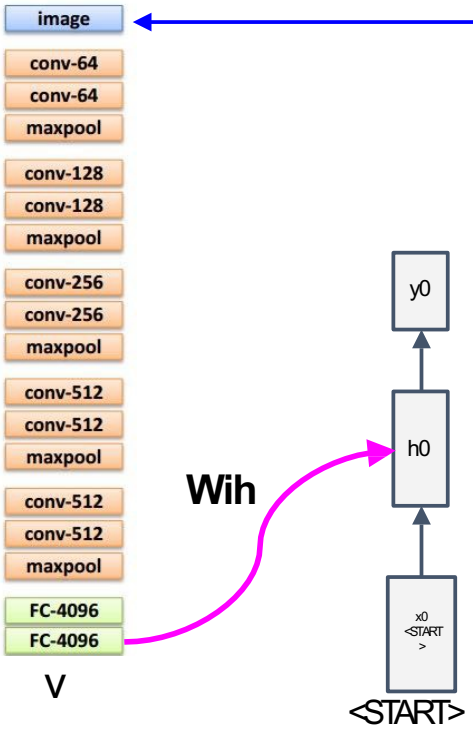


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
x0
<START
>
<START>
test image
Slide credit: Andrej Karpathy



test image

before:

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

now:

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

Slide credit: Andrej Karpathy

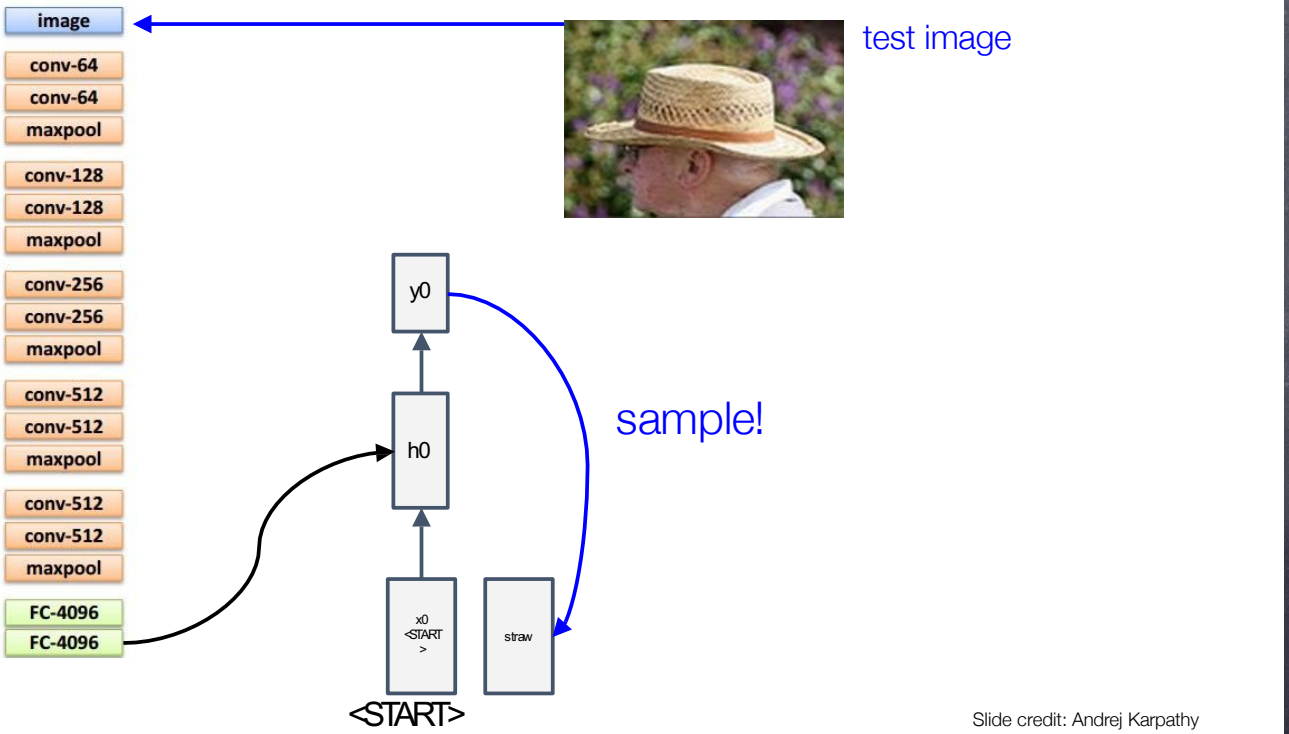


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
h0
x0
<START>
straw
<START>
sample!
Slide credit: Andrej Karpathy

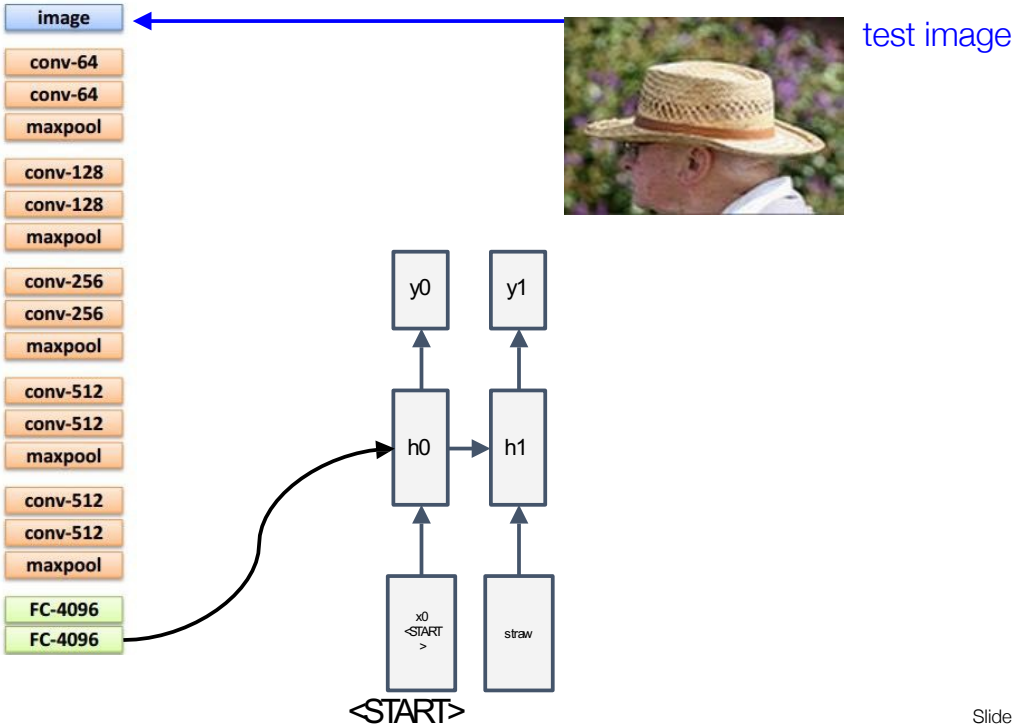


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<START
>
straw
<START>
Slide credit: Andrej Karpathy

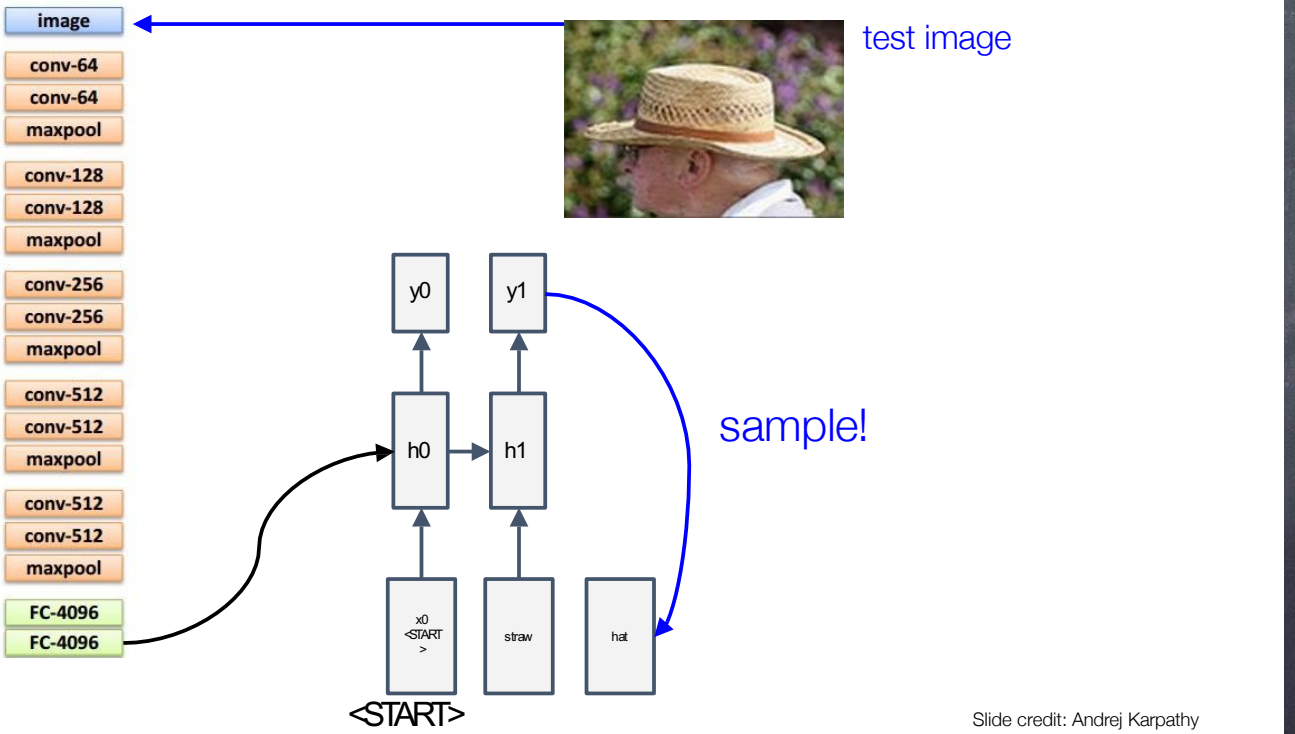


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<START
>
straw
hat
<START>
sample!
Slide credit: Andrej Karpathy

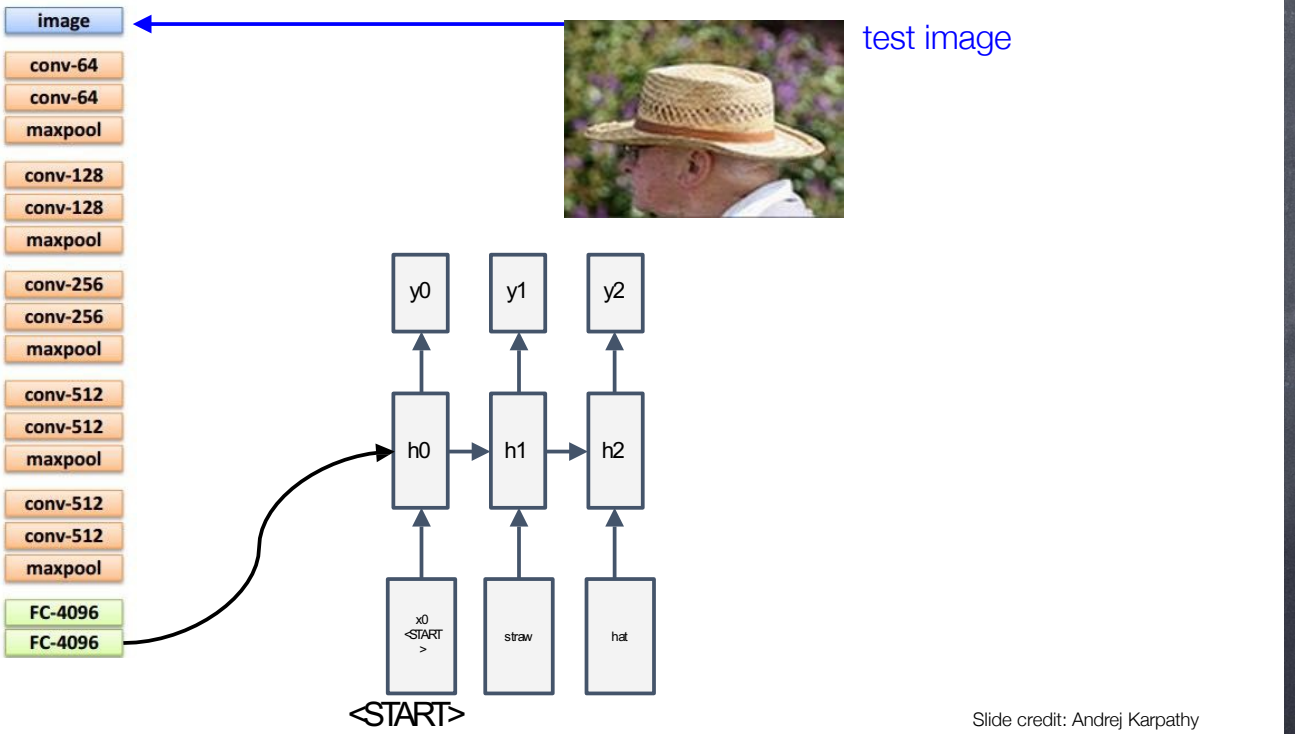


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0
<START>
straw
hat
<START>
Slide credit: Andrej Karpathy

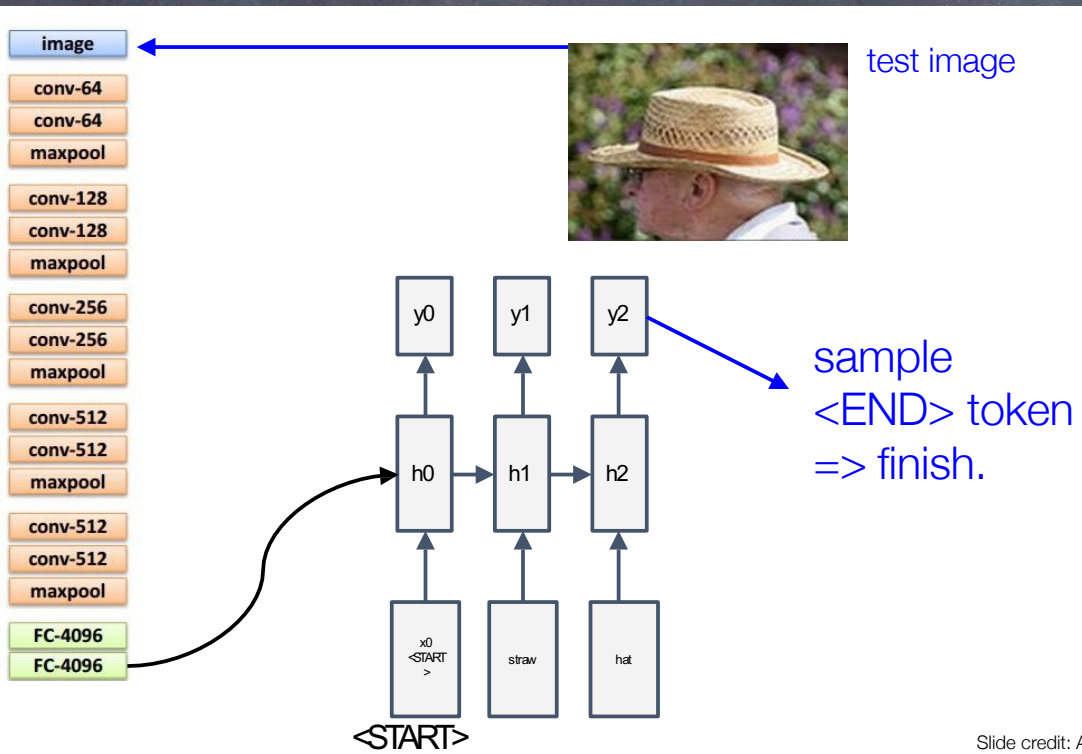


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0
<START>
straw
hat
<START>
sample
<END> token
=> finish.
Slide credit: Andrej Karpathy

Example

- Ground truth - What are the places to go in Las Vegas?
- Prediction - What are the places to go in Las Vegas

Pros and Cons

- Will predict: "the clouds are in the sky"
- May not predict: "I grew up in India ... I speak fluent Hindi."

Project(s)

- There will be two kinds of projects:
  - 3-0-0 part
  - 0-0-2/3 part (lab)

Project Team Size

- 3 students each project group
  - BSB students should create their groups - their projects will be tailored towards BSB problems

Project 1

- For 3-0-0 part
  - Select a problem of your choice (BSB - from your core department)
  - The project should be about total 60 hours of worth of work (20 hours per student)
  - Kaggle competition, implementation of existing papers, comparisons of multiple algorithms on two or more databases

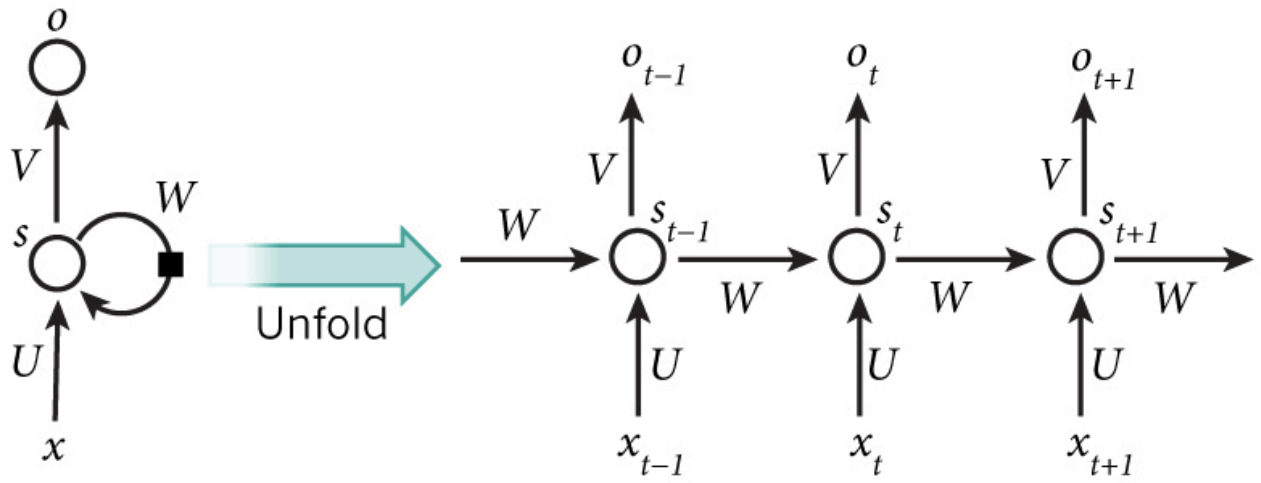
Project 2

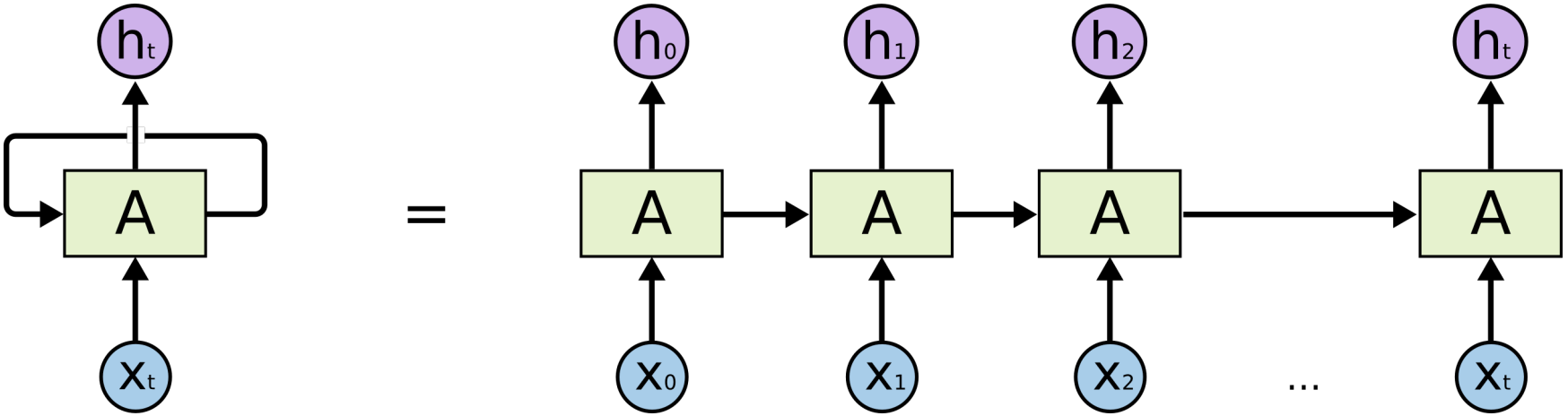
- x-0-2/3 part
  - The project should be real world application oriented implementation using learnings in the lab
  - Distinct from 3-0-0 project
  - Same team may do this project

Deadlines

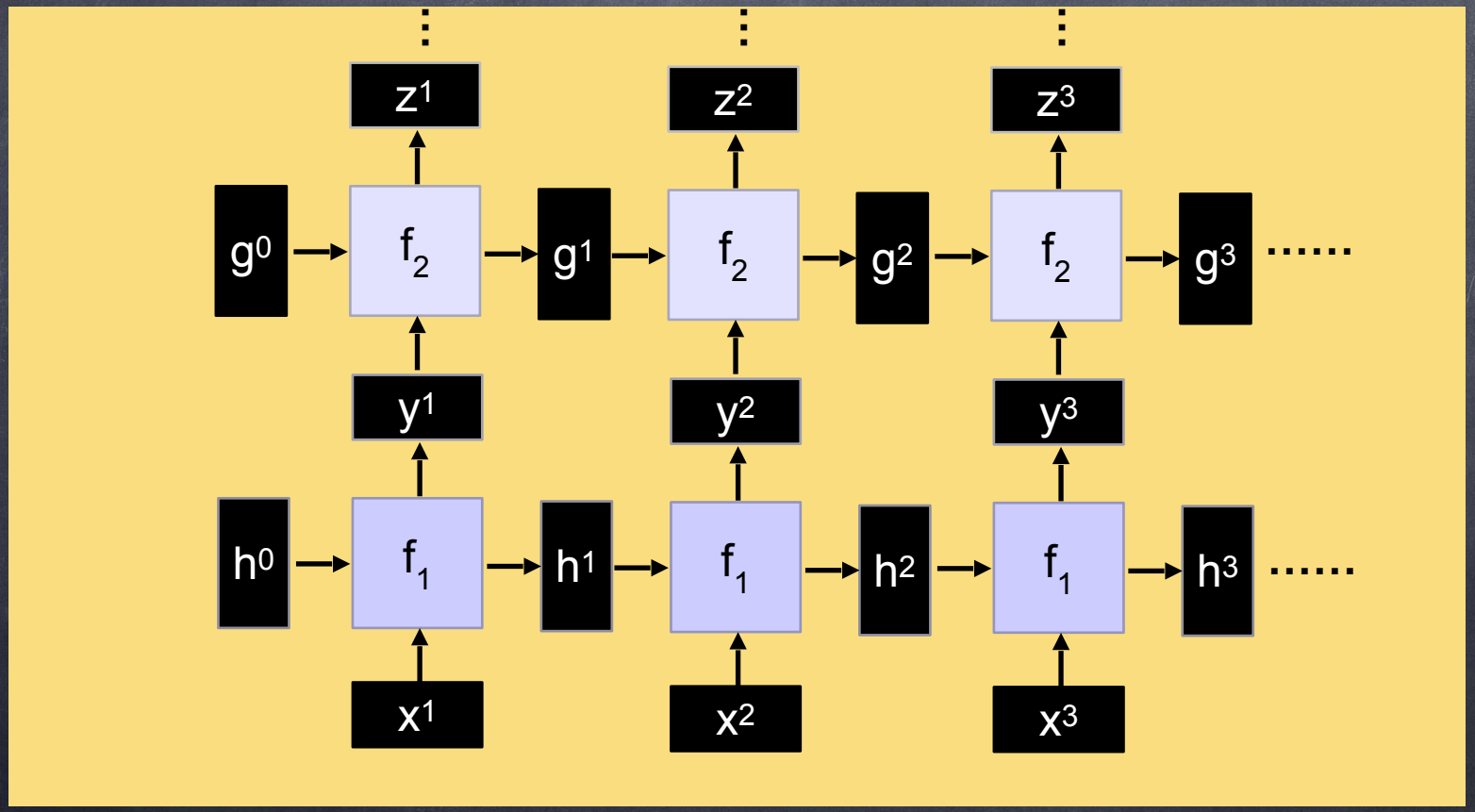
- Team formation: March 6
- Proposals from each team: March 15 (max 2 pages for each project)
  - Problem statement - 1 para
  - Lit review around problem statement - 1-2 para
  - Plan including databases and any other details
- Completing Project 2: April 15
- Completing Project 1: Just before exam - (April end/May Start)

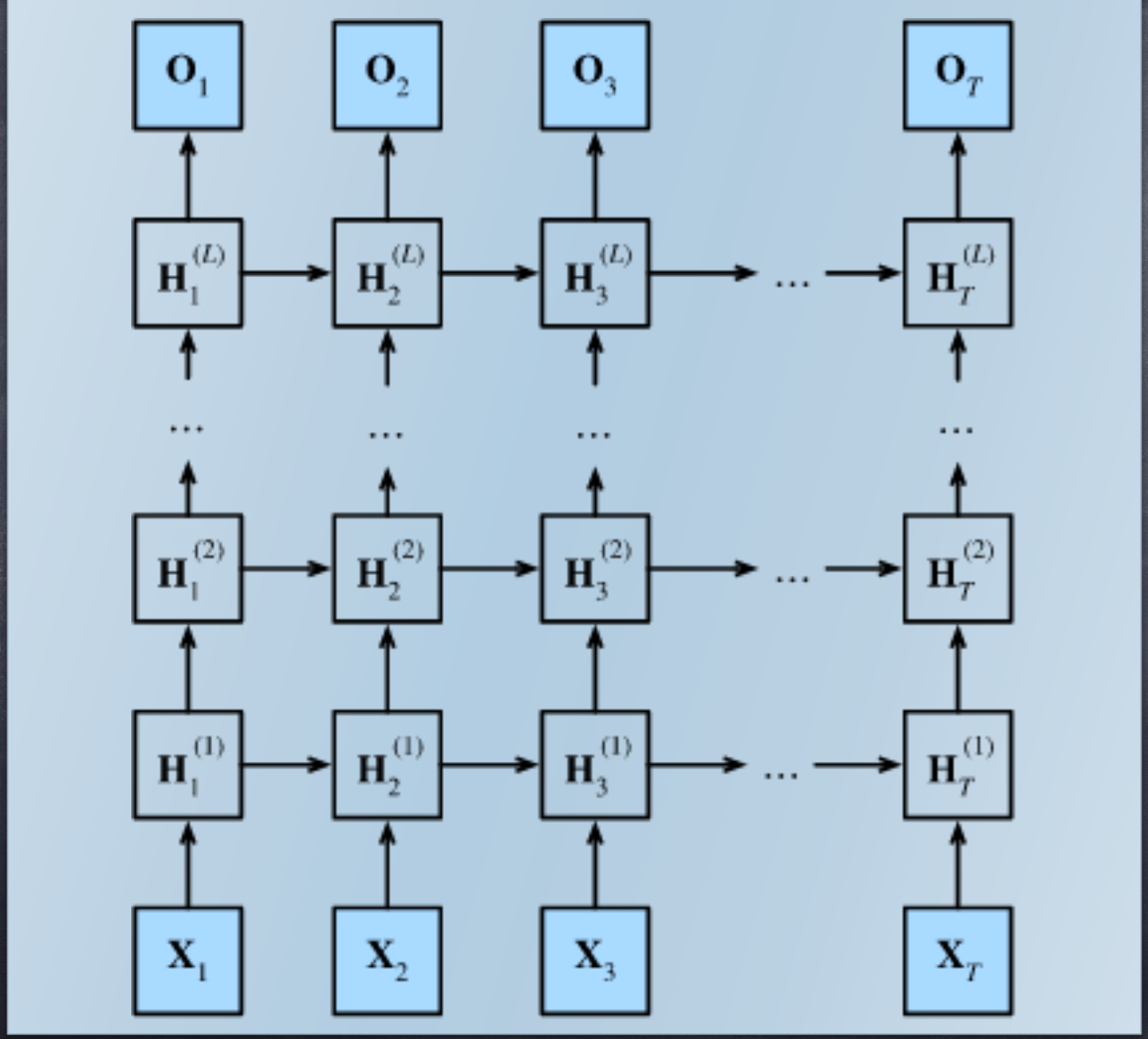
Recurrent Neural Network





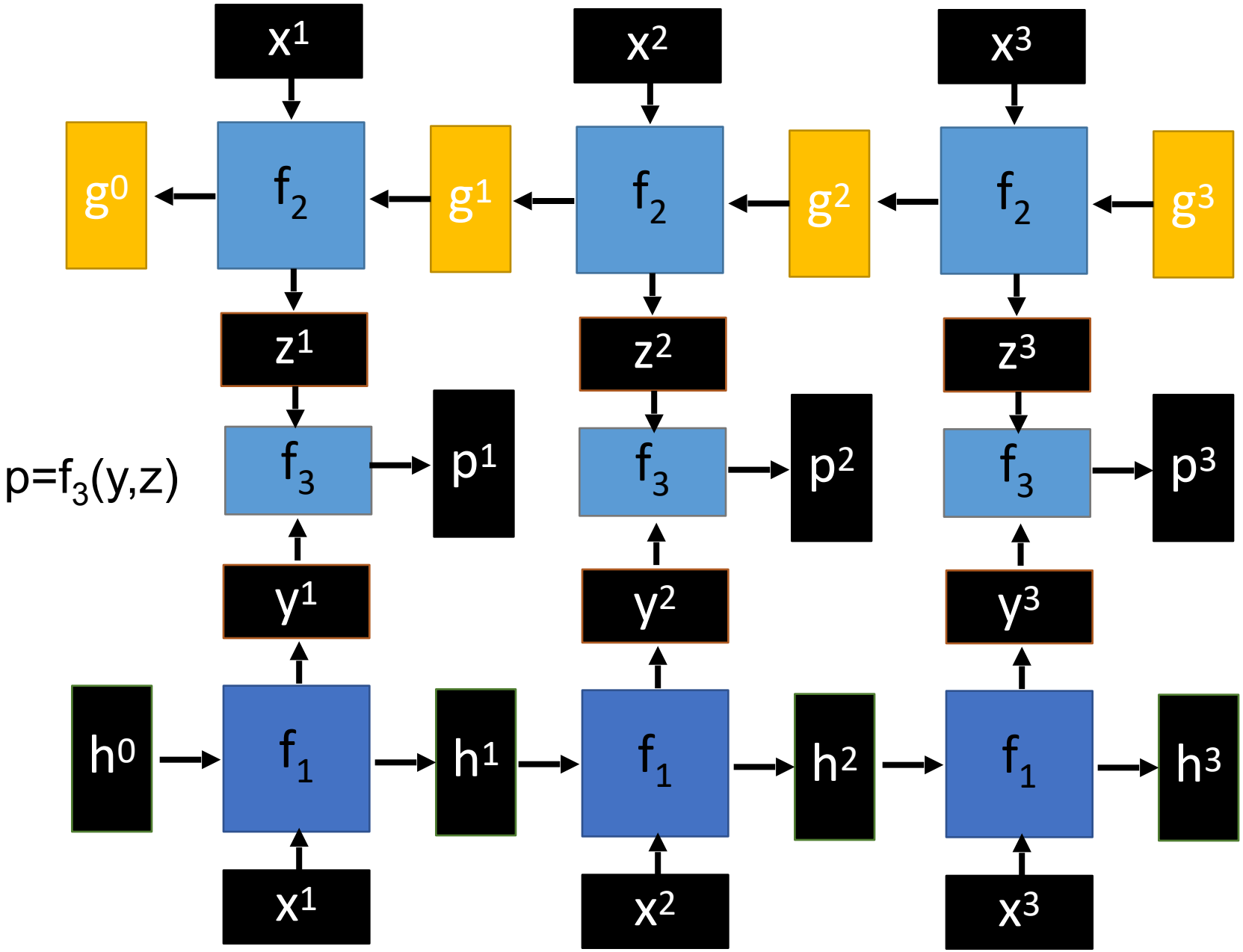
Deep RNN




$\mathbf{O}_1$
$\mathbf{O}_2$
$\mathbf{O}_3$
$\mathbf{O}_T$
$\mathbf{H}_1^{(L)}$
$\mathbf{H}_2^{(L)}$
$\mathbf{H}_3^{(L)}$
$\mathbf{H}_T^{(L)}$
$\mathbf{H}_1^{(2)}$
$\mathbf{H}_2^{(2)}$
$\mathbf{H}_3^{(2)}$
$\mathbf{H}_T^{(2)}$
$\mathbf{H}_1^{(1)}$
$\mathbf{H}_2^{(1)}$
$\mathbf{H}_3^{(1)}$
$\mathbf{H}_T^{(1)}$
$\mathbf{X}_1$
$\mathbf{X}_2$
$\mathbf{X}_3$
$\mathbf{X}_T$

Bidirectional RNN

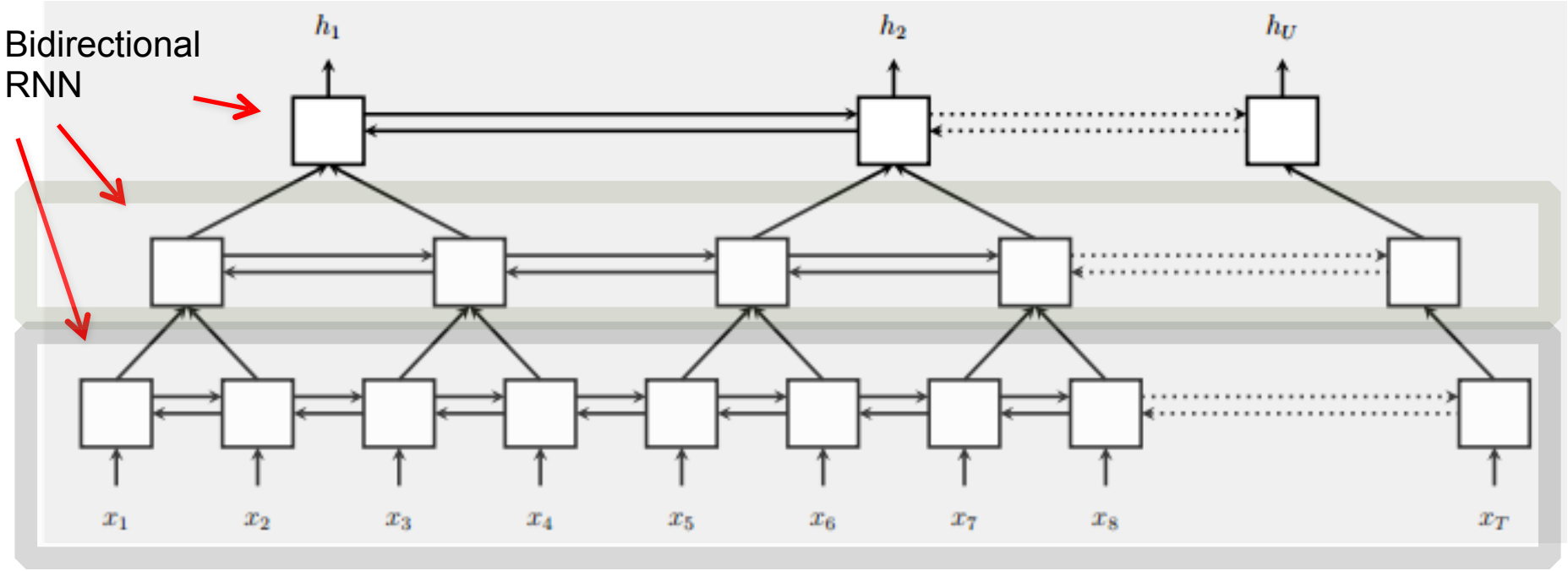
$y,h=f_1(x,h)$ $z,g = f_2(g,x)$



Pyramid RNN

Significantly speed up training

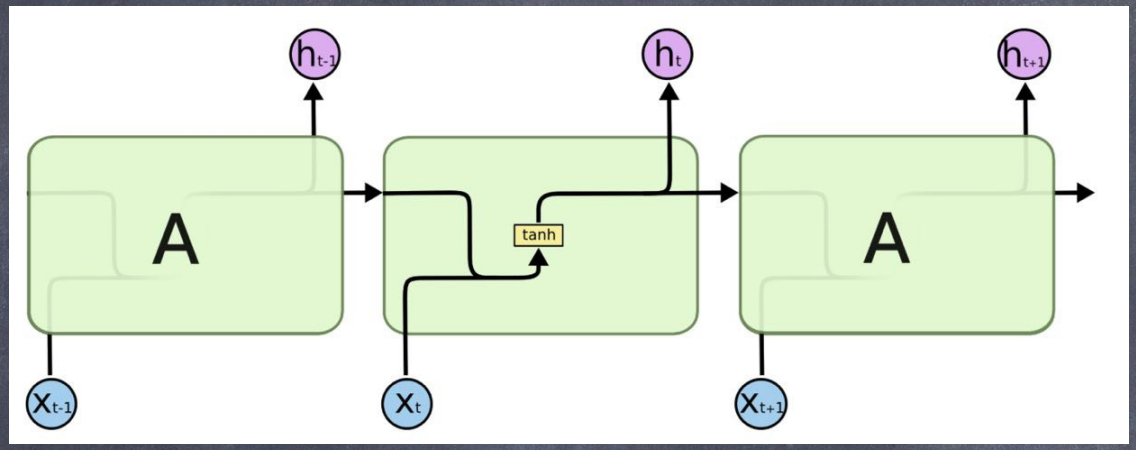
- Reducing the number of time steps

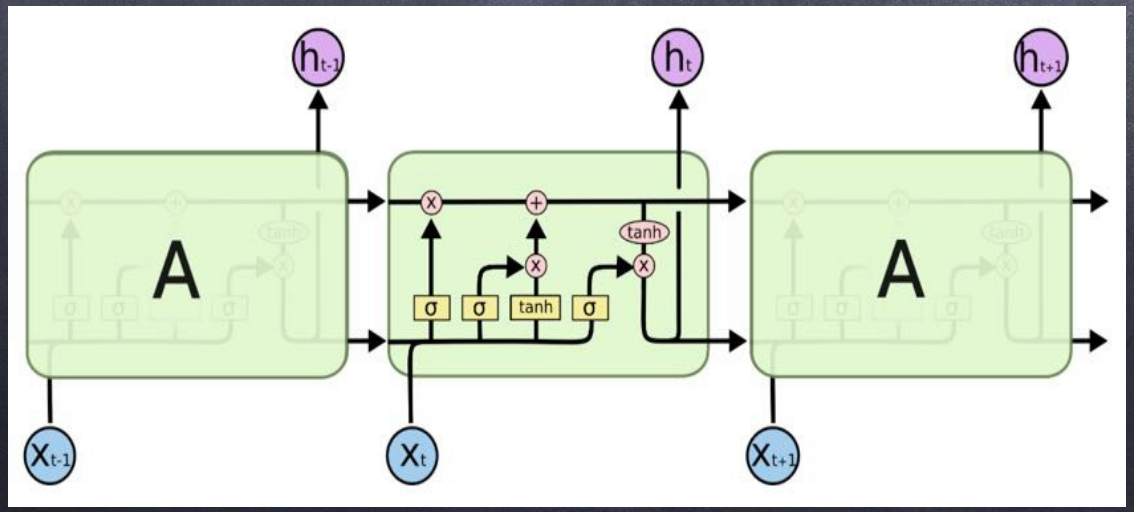


W. Chan, N. Jaitly, Q. Le and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," ICASSP, 2016

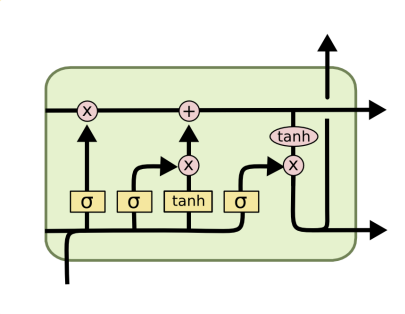
- long-term temporal dependencies

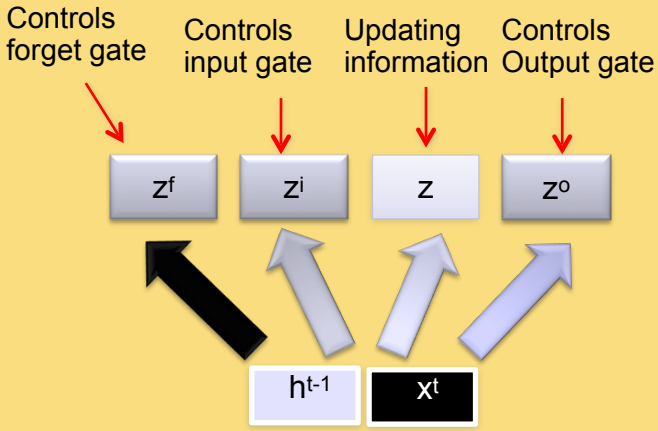
LSTM: Memory Extension of RNN





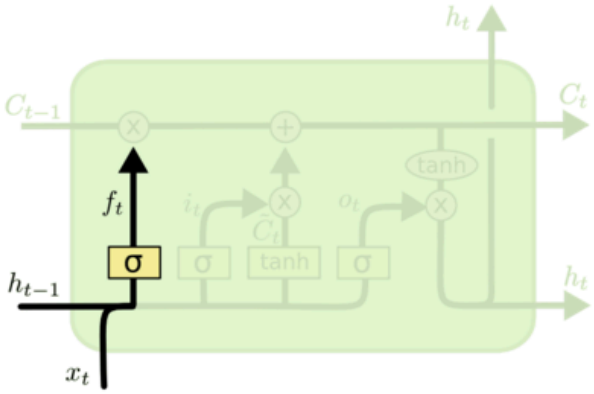
LSTM


tanh
σ
σ
tanh
σ


Controls forget gate
Controls input gate
Updating information
Controls Output gate
z^f
z^i
z
z^o
h^{t-1}
x^t

LSTM

Walkthrough

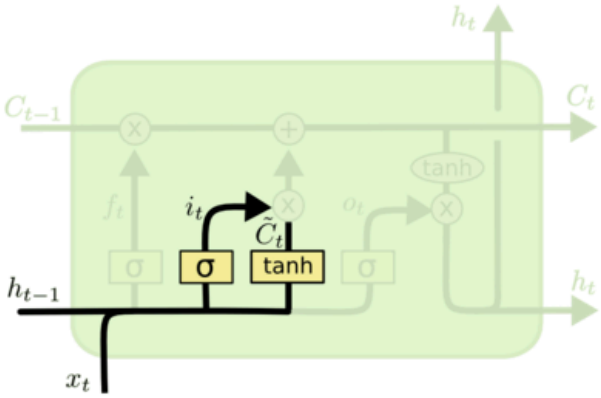


What part of memory to "forget" – zero means forget this bit

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM

Walkthrough



What bits to insert into the next states

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

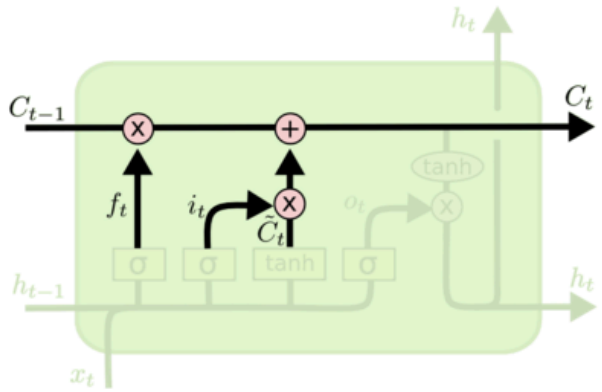
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

What content to store into the next state

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

Walkthrough



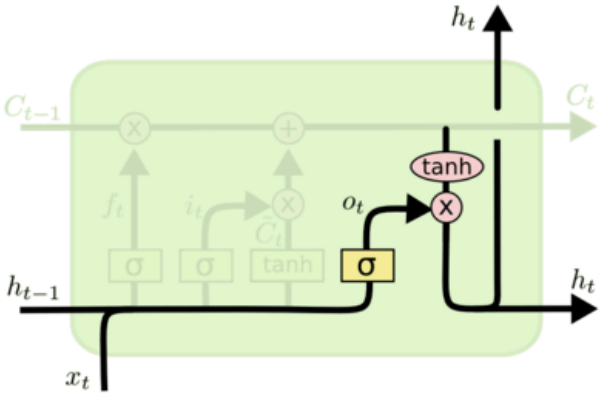
Next memory cell content – mixture of not-forgotten part of previous cell and insertion

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

Walkthrough



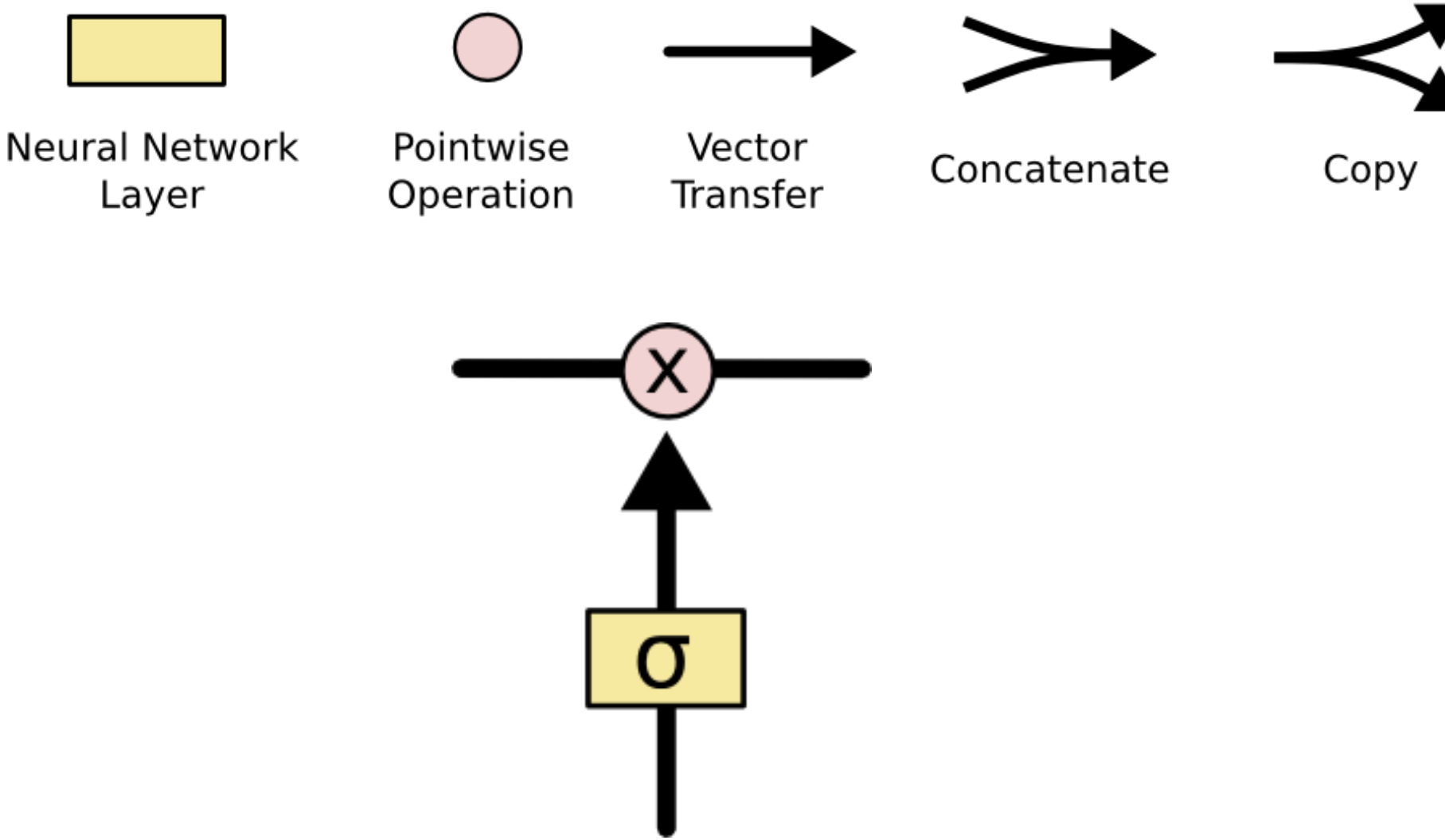
What part of cell to output

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

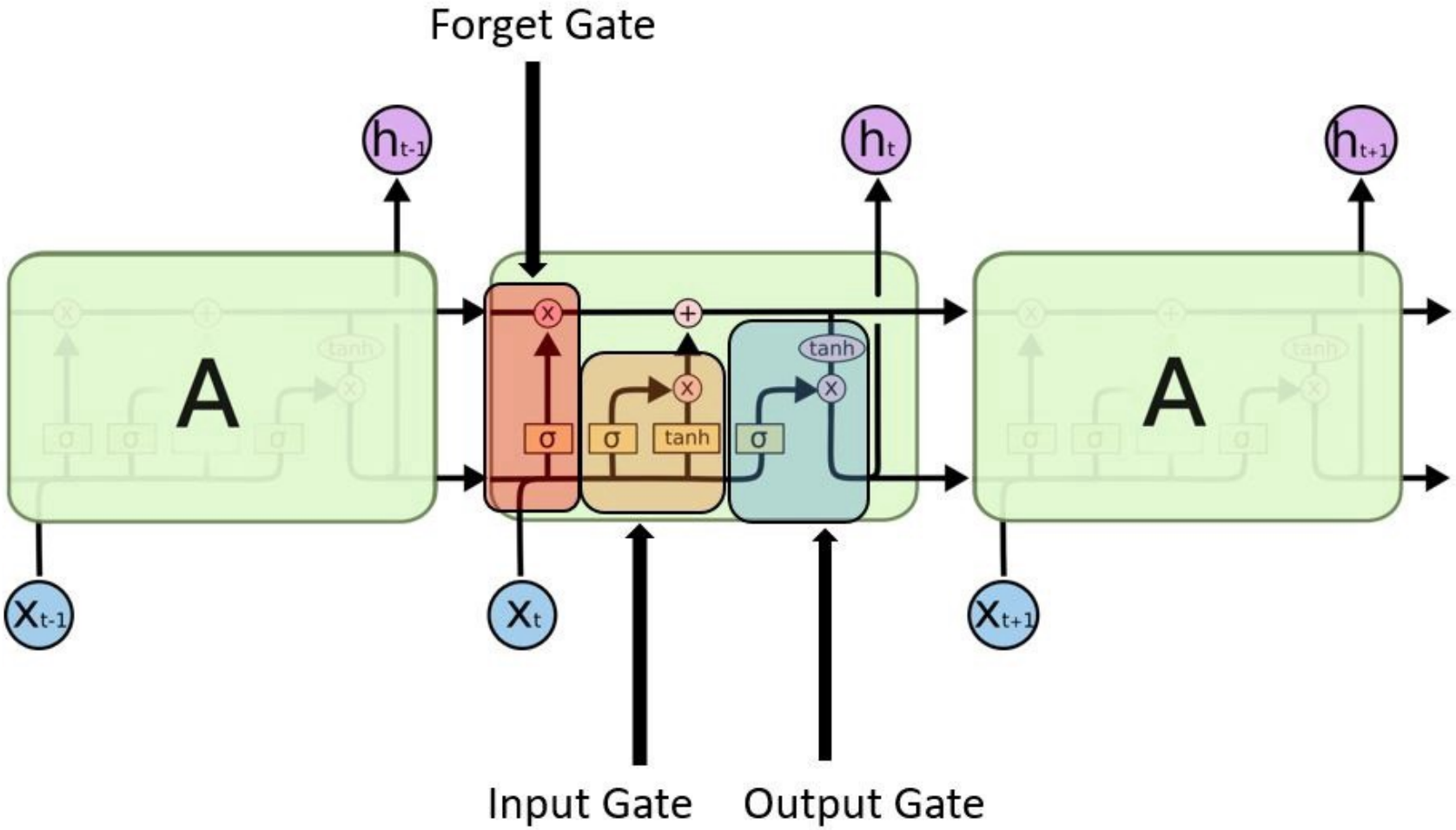
$$h_t = o_t * \tanh\left(C_t\right)$$

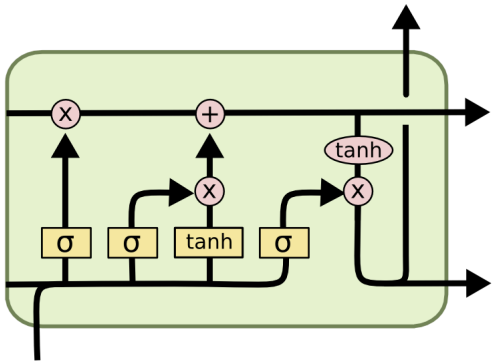
tanh maps bits to [-1,+1] range

http://colah.github.io/posts/2015-08-Understanding-LSTMs/



The sigmoid layer outputs numbers between 0-1 determine how much each component should be let through. Pink X gate is point-wise multiplication.


Forget Gate
h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
Input Gate
Output Gate



These 4 matrix computation should be done concurrently.

$z = tanh(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$

c^{t-1}

$z^i = \sigma(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$

Controls forget gate

Controls input gate

Updating information

Controls Output gate

$z^f = \sigma(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$

z^f z^i z z^o

h^{t-1} x^t

$z^o = \sigma(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$

Information flow of LSTM