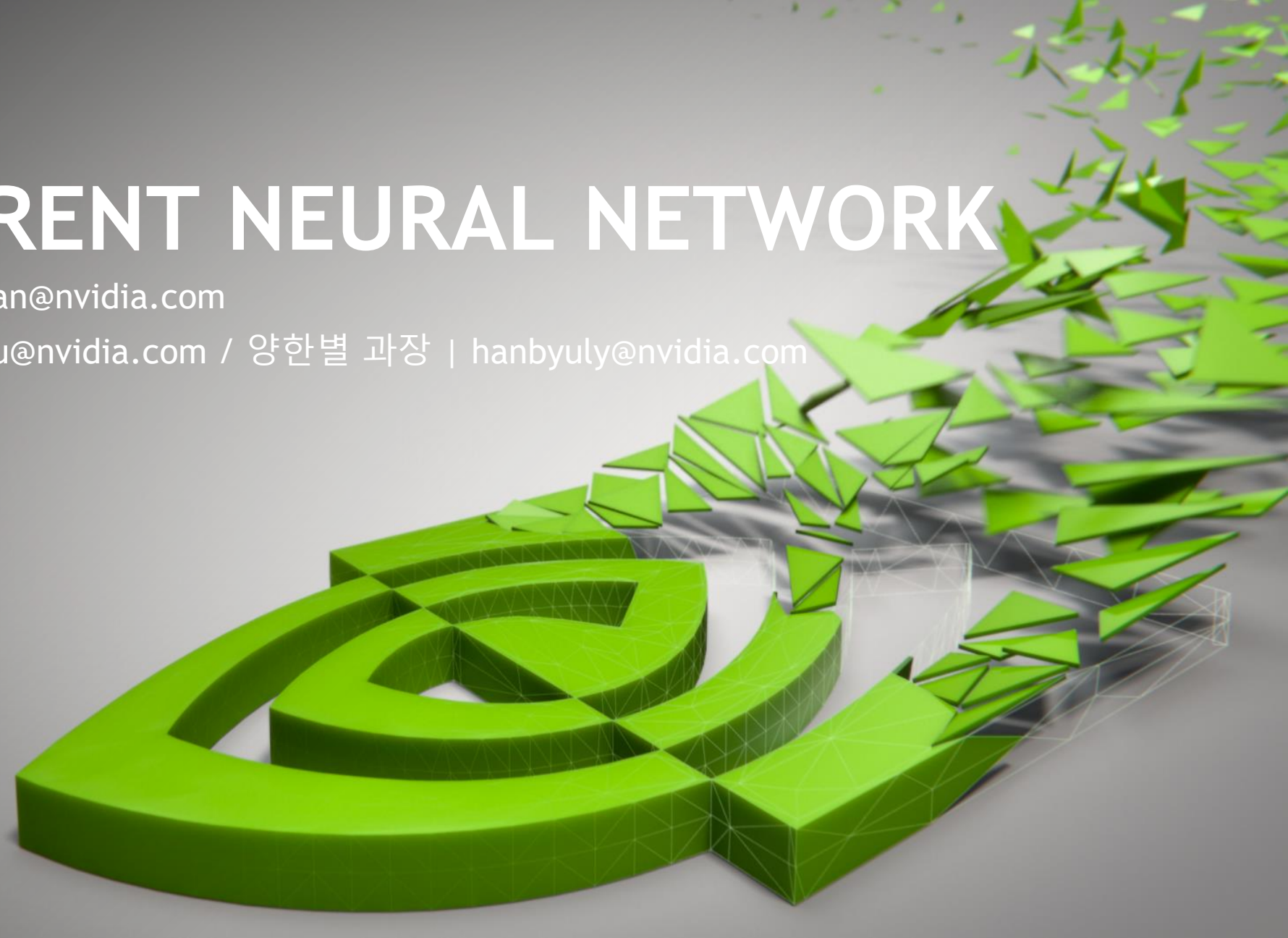
# RECURRENT NEURAL NETWORK

한재근 과장 | jahan@nvidia.com
유현곤 부장 | hryu@nvidia.com / 양한별 과장 | hanbyuly@nvidia.com

NVIDIA



# AGENDA

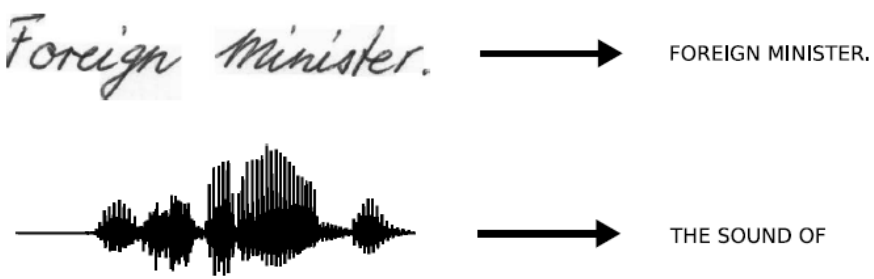
Introduction to Recurrent Neural Network

# Introduction to RNN

# Sequence Classification and Labeling

The sequence labeling:

- Handwriting Recognition
- Speech Recognition
- Language Translation

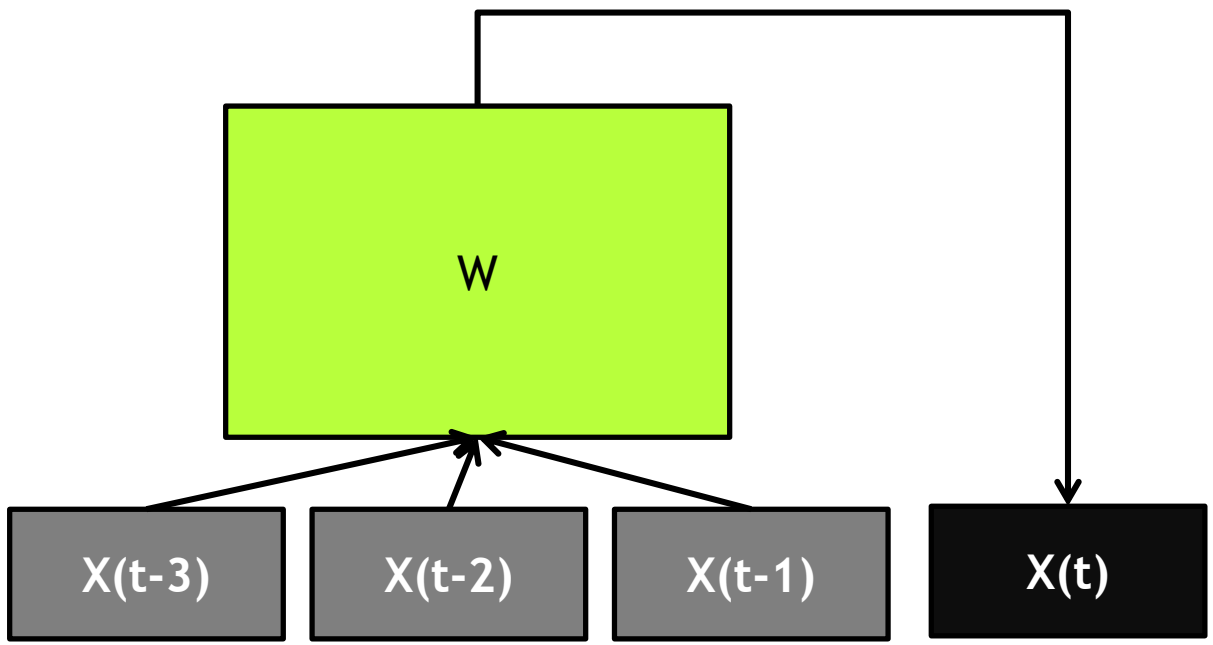



Special case of pattern classification: ind… to be independent. Both the inputs and the labels are strongly correlated.

# Example

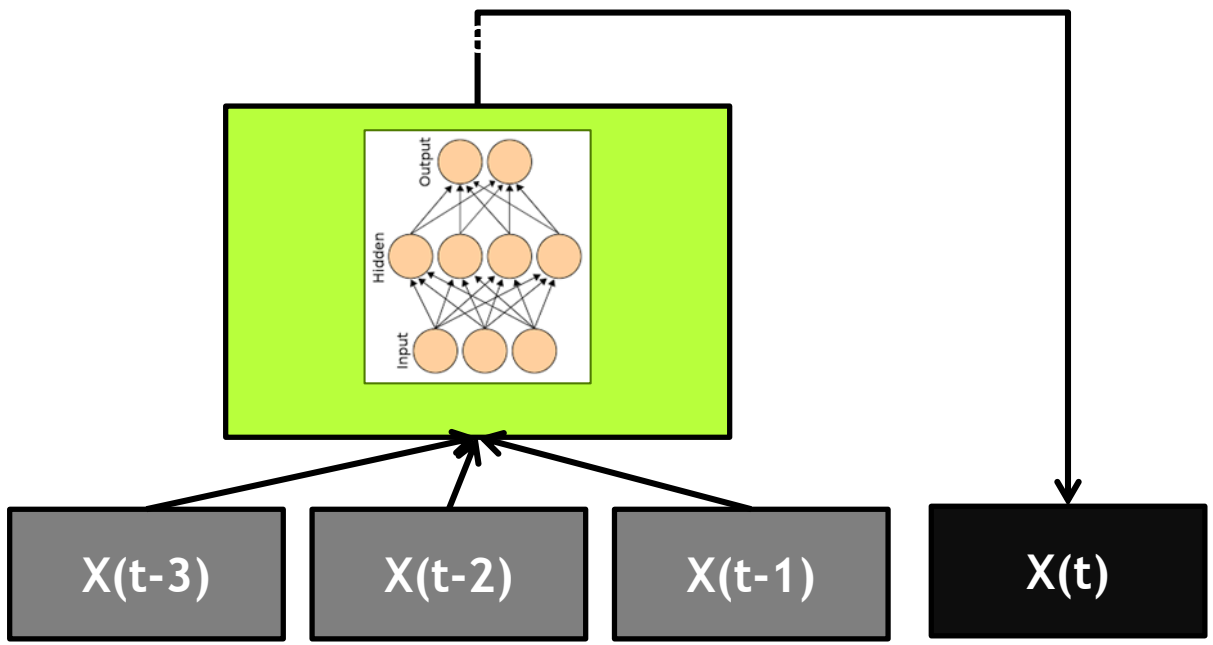
**Problem:** Find linear function which predicts the next term x(t+1) in the input sequence based on N previous elements { x(t-N).. X(t-1) }

Auto-regression is classical solution.



# Feed Forward Time-Delay NN

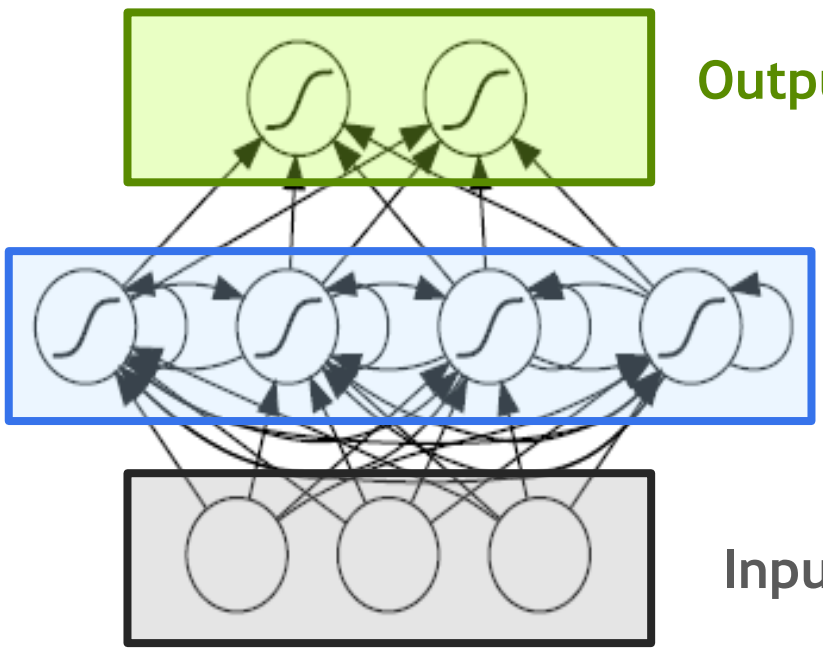
Feed-forward NN (e.g. Conv NN) is natural extension of Auto-regression:
we use network (non-linear function) with input composed from fixed number of sequence elements



# Recurrent NN

Time Delay-NN have fixed, limited temporal memory. How can we predict sequences with long or multi-scale time-dependencies?

Recurrent neural networks (RNNs) use their hidden states work as memory on previous inputs:

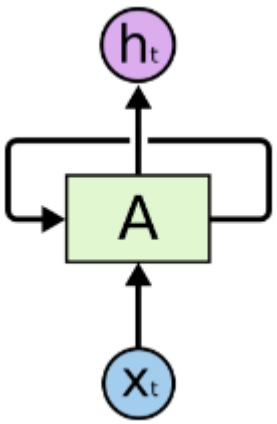


**Output** $y(t) = f\big(h(t-1)\big)$

**Hidden State** $h(t) = g\big(y(t), \boldsymbol{h(t-1)}\big)$

**Input** $x(t)$

# Neural Network with Memory




Previous input affects current output

→ We use this for future prediction

Current State — Previous State

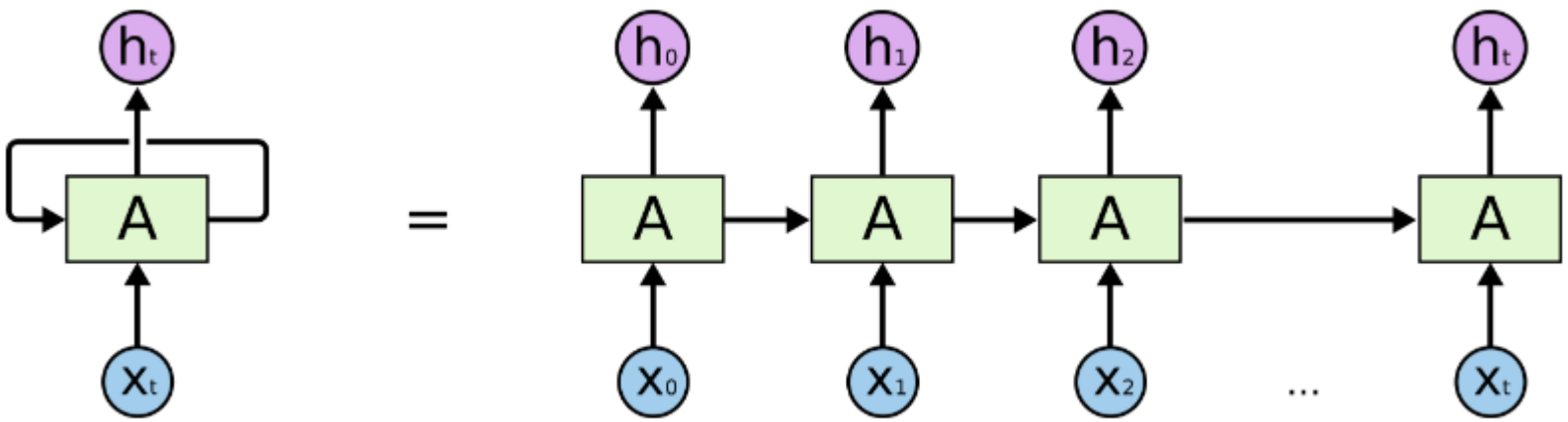
$$h_t = f_W(h_{t-1}, x_t)$$

Current Input — Current Input

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

$$y_t = W_{hy} h_t$$

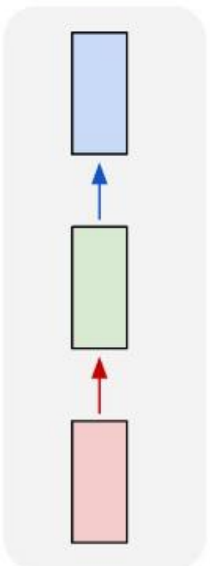
# An Unrolled Recurrent Neural Network



# Flexible RNNs

## Examples

one to one



Vanilla NN

one to many

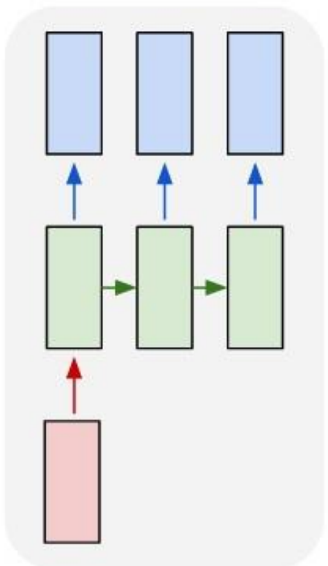
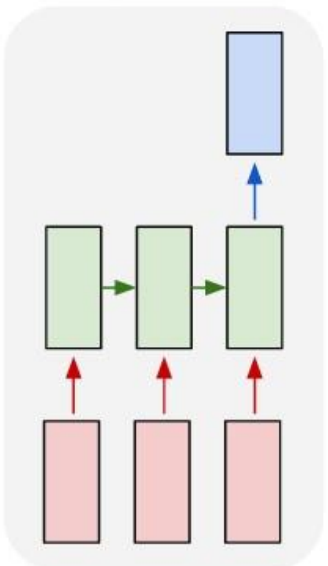


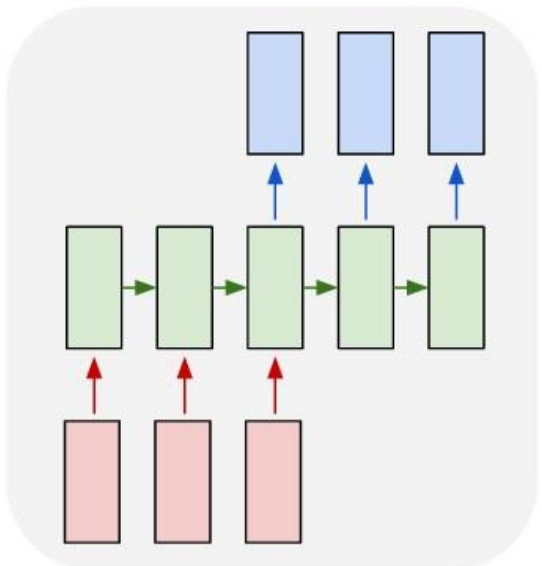
Image Captioning

many to one



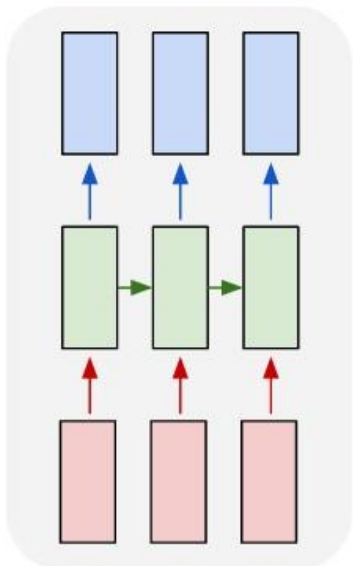
Sentiment Classification

many to many



Machine Translation

many to many

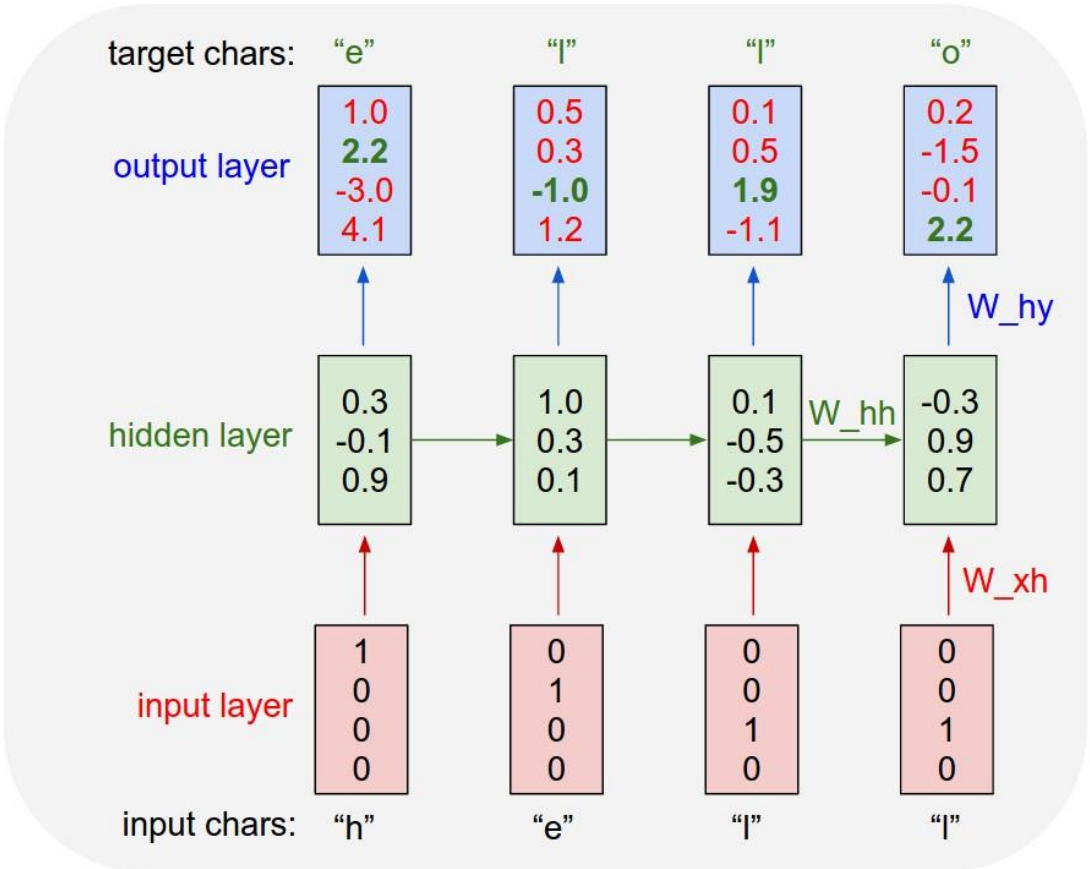


Video Classification

# Character Level Example

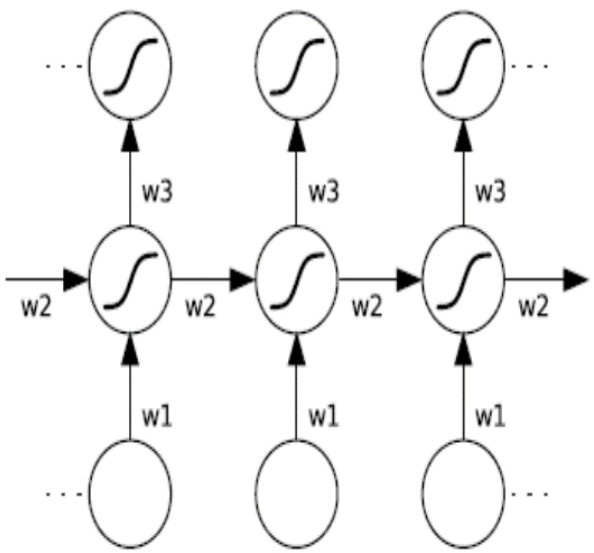
Vocabulary:
[h,e,l,o]

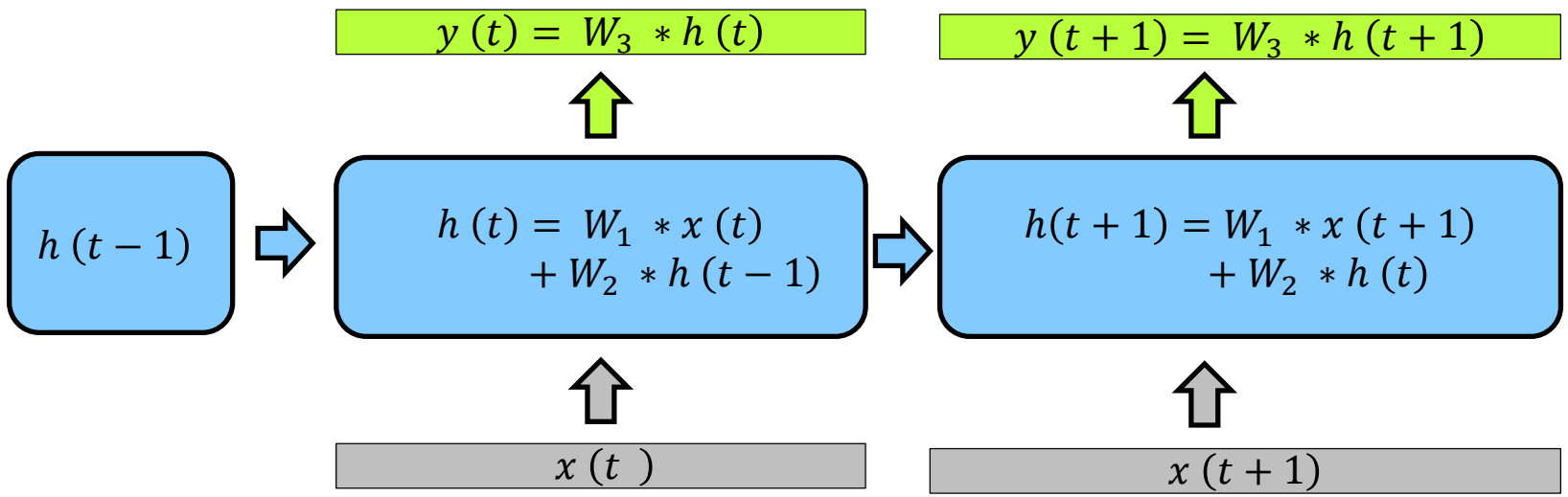
Example training sequence:
**“hello**



# Recurrent NN

Let’s unfold RNN in time:

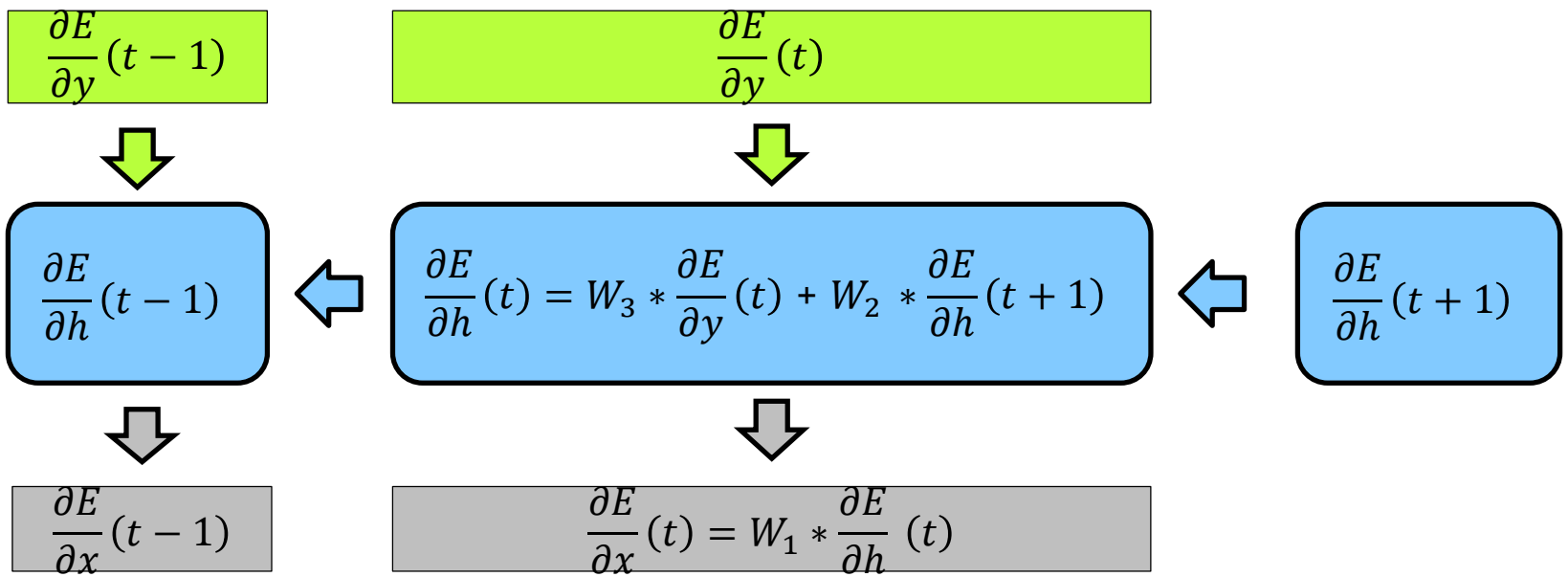


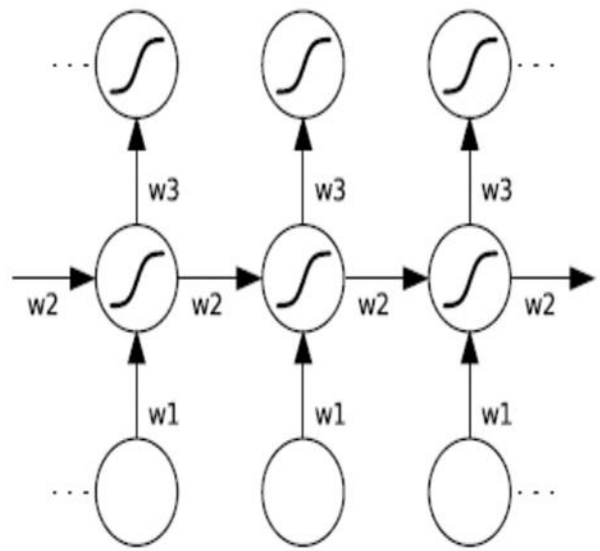




*Warning: we skip bias and non-linear function for simplicity!*

# Recurrent NN: training

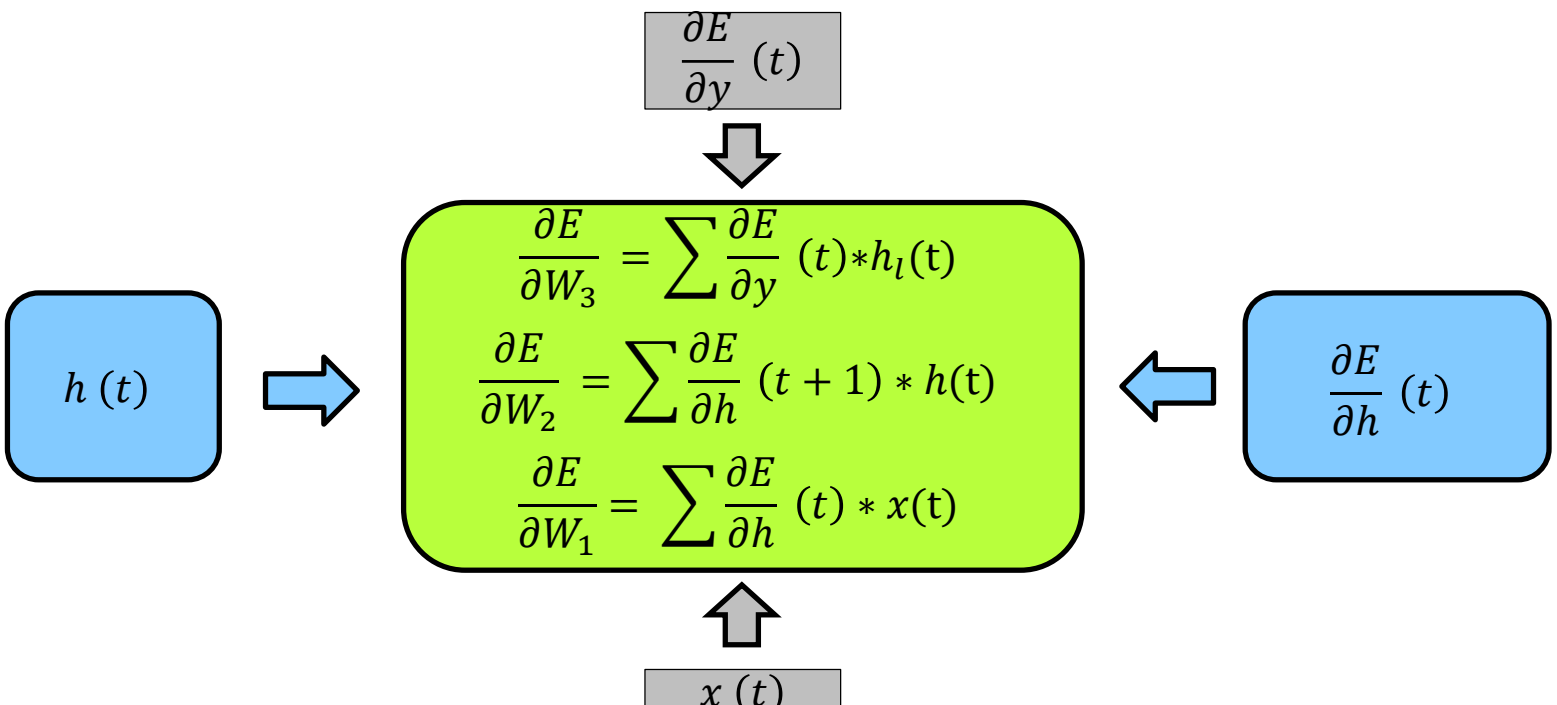
RNN training is done by using gradient back-propagation recursively back in time from T to 1. First let propagate $\frac{\partial E}{\partial y}(t)$

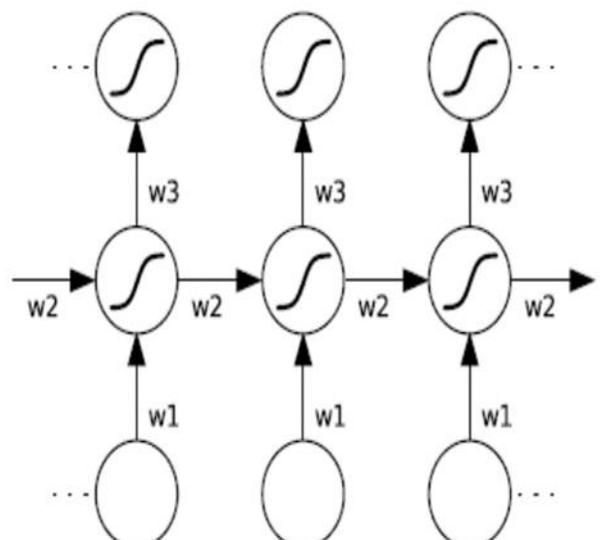


# Recurrent NN: training

After we have $\frac{\partial E}{\partial h}(t)$ we compute $\frac{\partial E}{\partial w}(t)$: we sum the “instant” derivatives with respect to the weights over all t=1..T

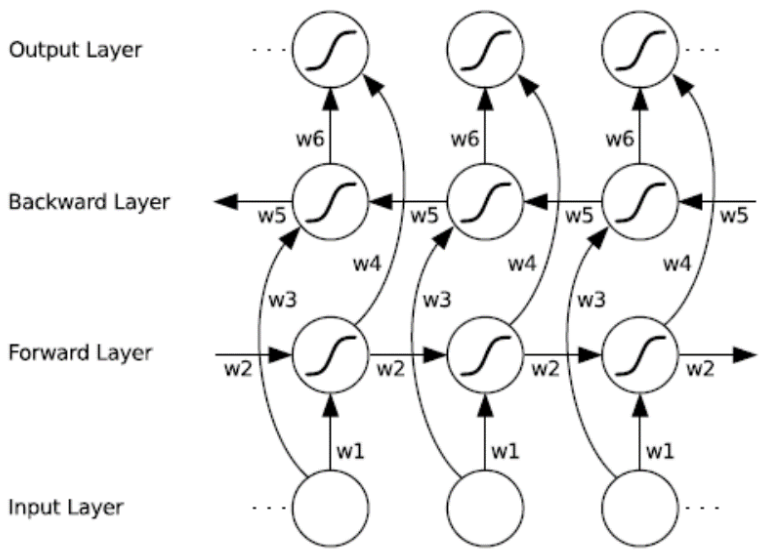


# Bi-directional RNN

For many problems (NLP, handwriting, speech recognition, protein structure prediction,…) it is beneficial to have access to future as well as to past context.

Bidirectional RNN: split recurrent layers into two separate recurrent layers both of which are connected to the same output layer. There is no information flows between the forward and backward hidden layers to make sure that the unfolded graph is acyclic



# Bi-directional RNN

Forward pass:

1. the input sequence is presented in opposite directions to the two hidden layers
2. the output layer is not updated until both hidden layers have processed the entire input sequence

**for** $t = 1$ to $T$ **do**
  Forward pass for the forward hidden layer, storing activations at each timestep
**for** $t = T$ to $1$ **do**
  Forward pass for the backward hidden layer, storing activations at each timestep
**for** all $t$, in any order **do**
  Forward pass for the output layer, using the stored activations from both hidden layers

# Bi-RNN: back-propagation

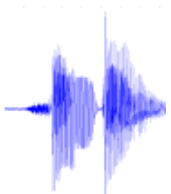
Back-propagation is similar to unidirectional RNN except that:

1. compute  all the gradients for output layer
2. send gradients back to the two hidden layers in opposite directions

for all $t$, in any order **do**
  Backward pass for the output layer, storing $\delta$ terms at each timestep
for $t = T$ to 1 **do**
  BPTT backward pass for the forward hidden layer, using the stored $\delta$ terms from the output layer
for $t = 1$ to $T$ **do**
  BPTT backward pass for the backward hidden layer, using the stored $\delta$ terms from the output layer

# Speech Recognition with RNN

# Traditional Speech Recognition Flow



Front End: MFCC

Acoustic-1: GMM

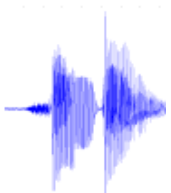
Acoustic-2: HMM / WFST

Word pronunciation: HMM / WFST

N-gram Language Model

1. The waveform spoken by a user is split into small consecutive slices or “frames” of 10 milliseconds of audio. Each frame is analyzed for its frequency content, and the resulting feature vector is passed to an acoustic model
2. Acoustic model GMM outputs a probability distribution over all the phonemes (sounds)
3. A Hidden Markov Model (HMM) impose temporal structure on this sequence of probability distributions.
4. This is then combined with other knowledge sources such as a Pronunciation Model that links sequences of sounds to valid words in the target language
5. Language Model that expresses how likely given word sequences.

# NN-based Speech Recognition Flow



Front End: MFCC

Acoustic Model

RNN

Word pronunciation

HMM / WFST

NN Language Model

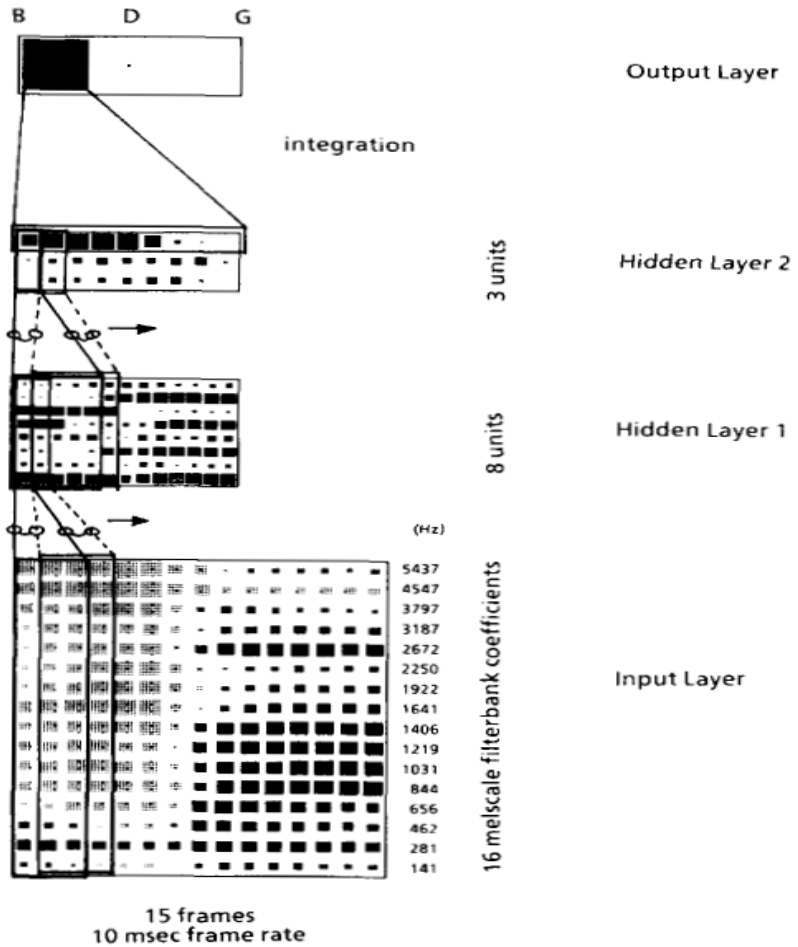
1. The waveform spoken by a user is split into small consecutive slices or “frames” of 10 milliseconds of audio. Each frame is analyzed for its frequency content, and the resulting feature vector is passed to an acoustic model
2. **New acoustic model based on Recurrent NN**
3. This is then combined with other knowledge sources such as a Pronunciation Model that links sequences of sounds to valid words in the target language
4. **NN-based Langiuage model**

# Time-Delay Neural Network (TD-NN)

TD-NN was proposed for phoneme recognition in [Weibel, Hinton, 1989].

3 layers NN:

- 8 units in 1st layer, 3 frames
- 3 units in 2nd layer; 5 frames
- higher layers to make decisions based on wider range in time

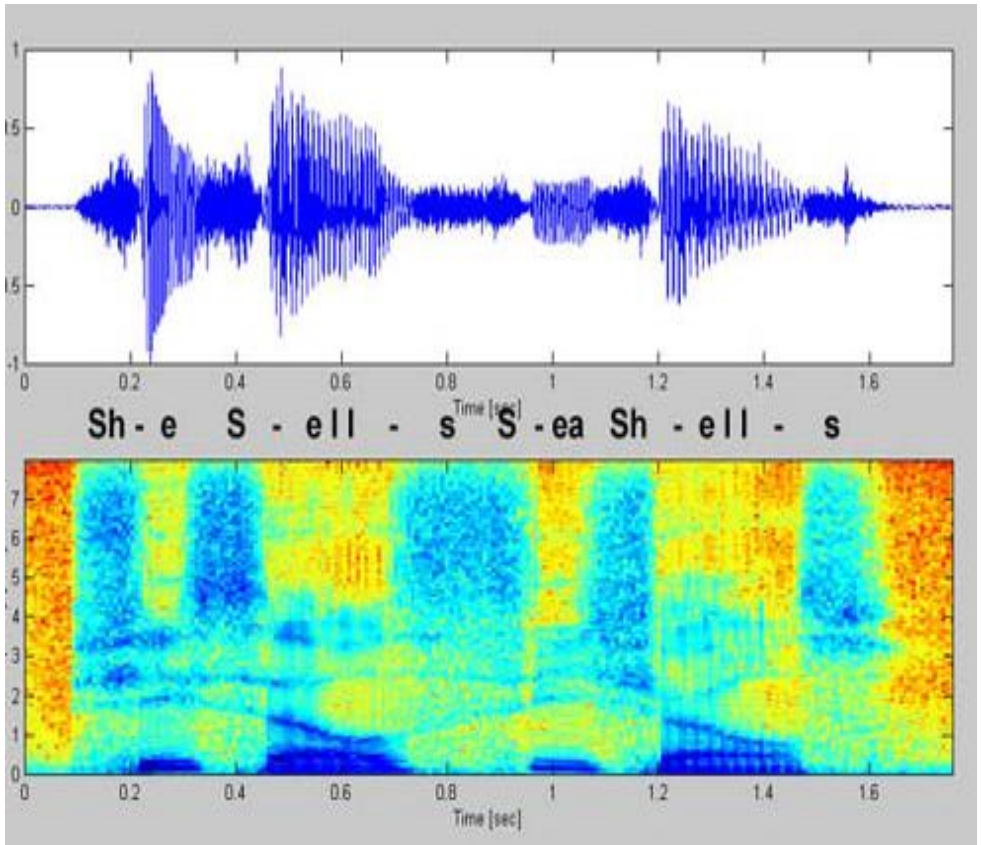



A.

# Acoustic Recognition with NN

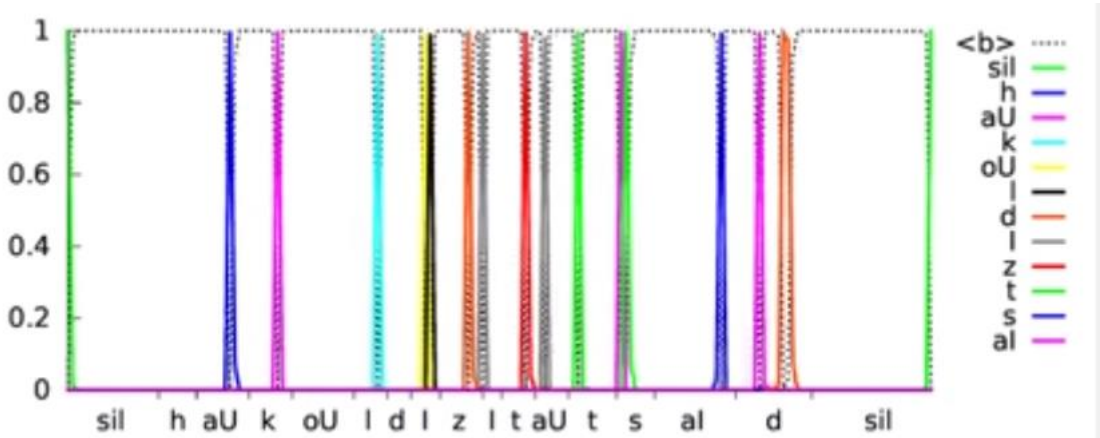
The basic approach:

1. Transform speech signal into sequence of spectrograms (2D representation of an acoustic signal based on *FFT* applied to window of speech).
2. Apply NN technique to “read” this 2D spectrogram



# Google Voice Search

Google just announced that they improved their voice search using acoustic models based on RNN and CTC (Connectionist Temporal Classification).




Example: "How cold is it outside".
The CTC model outputs spikes as it identifies various phonemes (shown in different colors) in the input speech signal. The x-axis shows the acoustic input timing for phonemes and y-axis shows the posterior probabilities as predicted by the neural network. The dotted line shows where the model chooses not to output a phoneme.

https://www.youtube.com/watch?v=5_9Soz3D41g&feature=youtu.be
http://googleresearch.blogspot.co.uk/2015/09/google-voice-search-faster-and-more.html

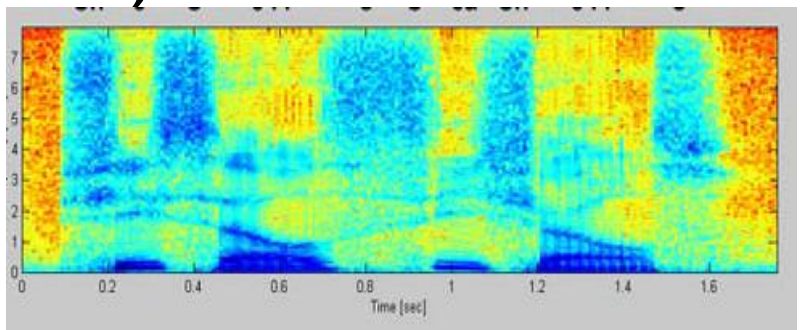
# DeepSpeech (Baidu)

RNN-based Automatic  Speech Recognition

1. Transcribe acoustic directly into letters
2. Decode transcription (fix “spelling”)

Transcription is based on RNN only.

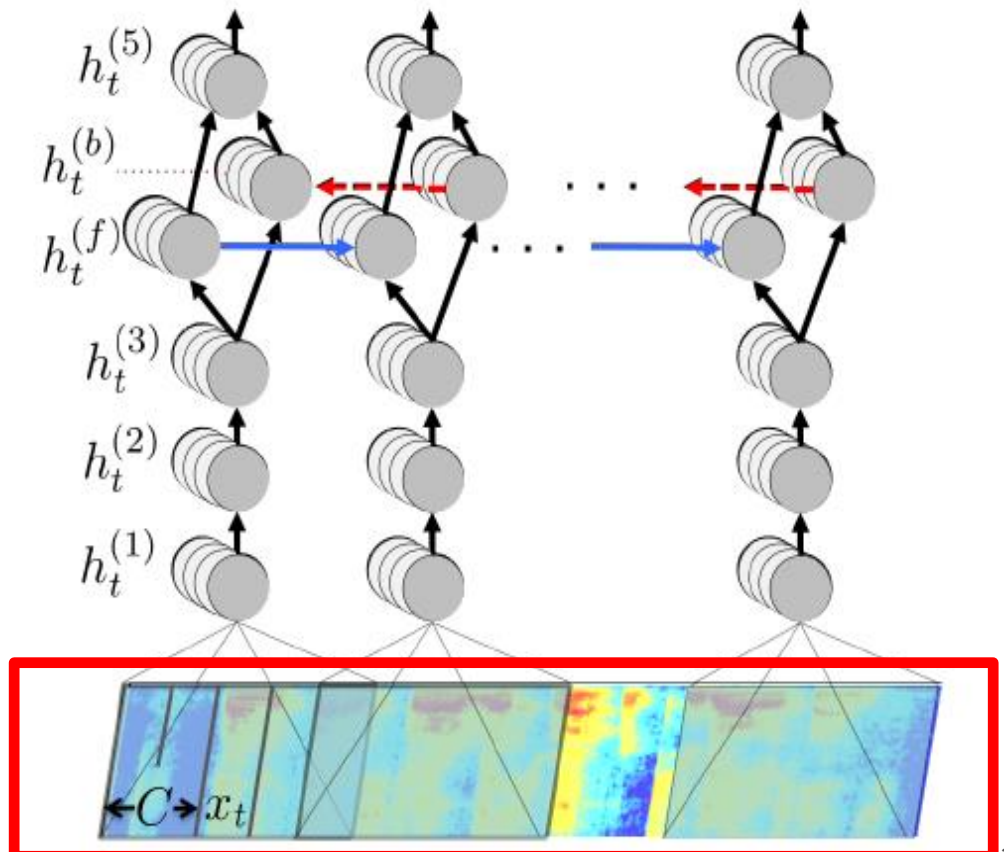
- No hand-designed models for background noise, reverberation, or speaker variation
- No phoneme dictionary
- No HMM / WFST




| RNN output |
| --- |
| what is the weather like in bostin right now<br>prime miniter nerenr modi<br>arther n tickets for the game |

| Decoded Transcription |
| --- |
| what is the weather like in boston right now<br>prime minister narendra modi<br>are there any tickets for the game |

# DeepSpeech



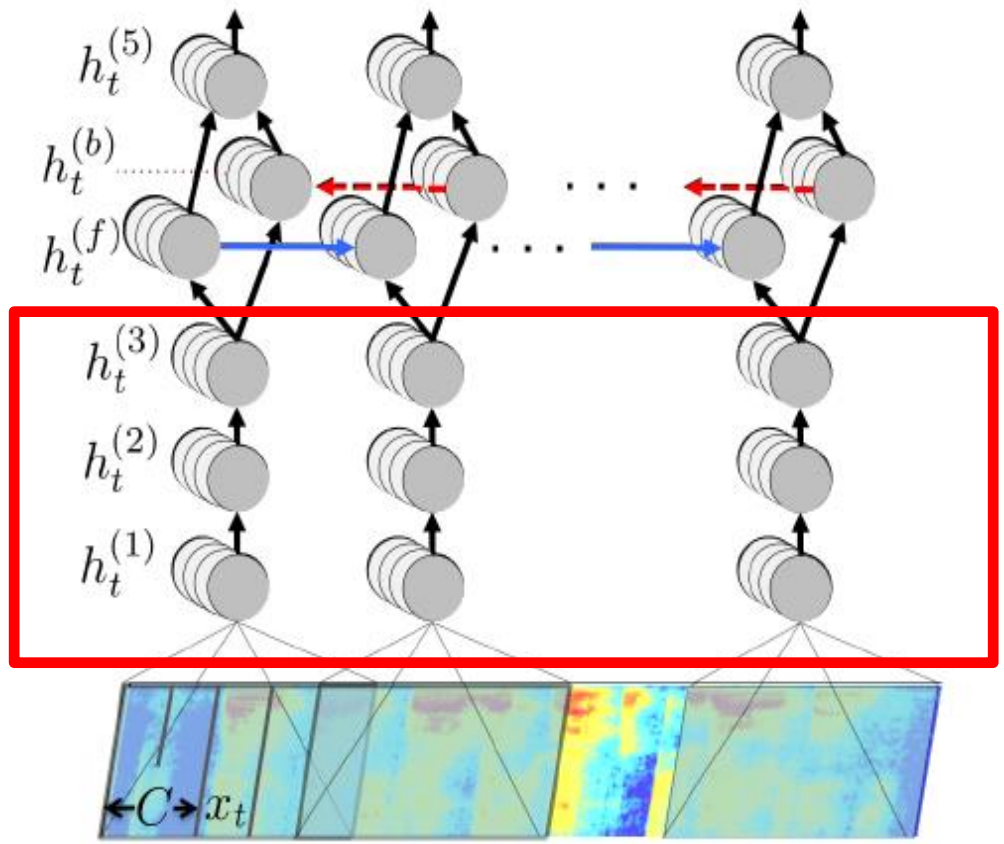

Input: the spectrogram frame x(t) along with a context of C frames on each side

# DeepSpeech

Layers $h^1$-$h^2$-$h^3$ : FC with clipped ReLU

$$h_t^l = \mathrm{g}\left(W^l * h_t^{l-1} + b^l\right)$$

$$g(x) = \min(\max(0, x), 20)$$

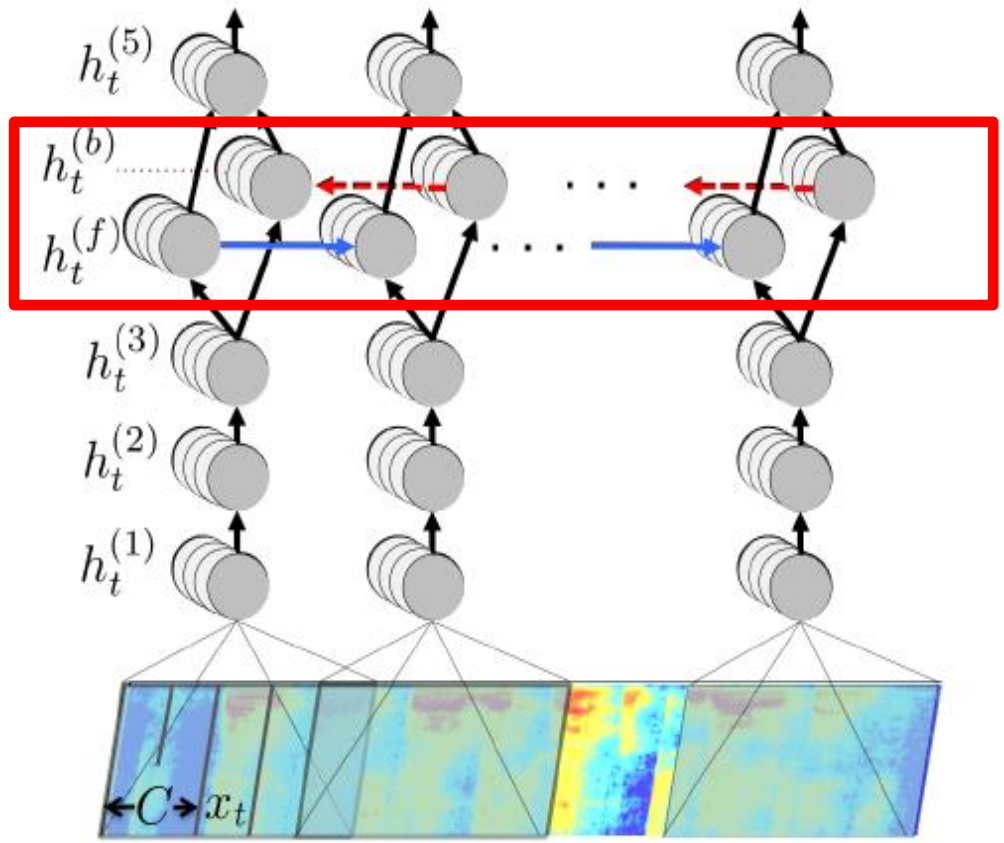


# DeepSpeech

Layer h4 is bi-directional recurrent layer, splitted into 2 sub-layers:

$$h_t^f = g\left(W^4 * h_t^3 + W_h^f * h_{t-1}^f + b^4\right)$$
$$h_t^b = g\left(W^4 * h_t^3 + W_h^b * h_{t+1}^b + b^4\right)$$

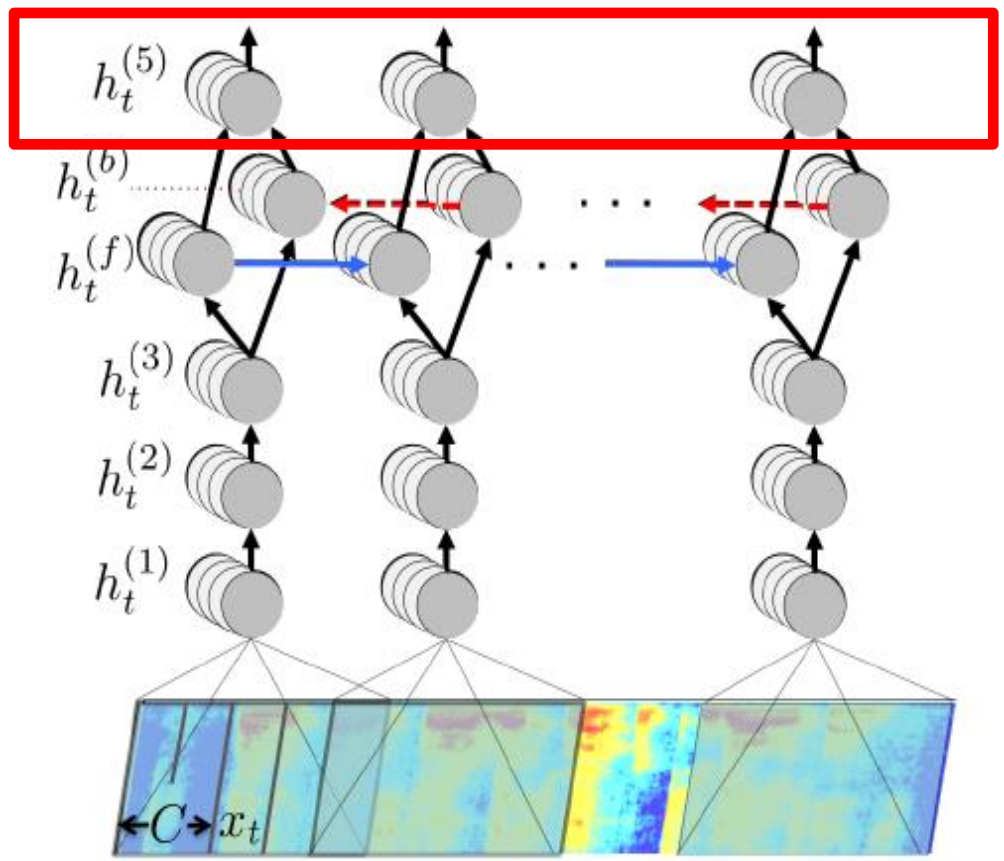
$h_t^f$ must be computed sequentially: 1 -> T
$h_t^b$ must be computed in reverse : T -> 1



# DeepSpeech

Last layer h5 is regular FC layer:

$$h_t^3 = g\left(W^5 * (h_t^f + h_t^b) + b^5\right)$$



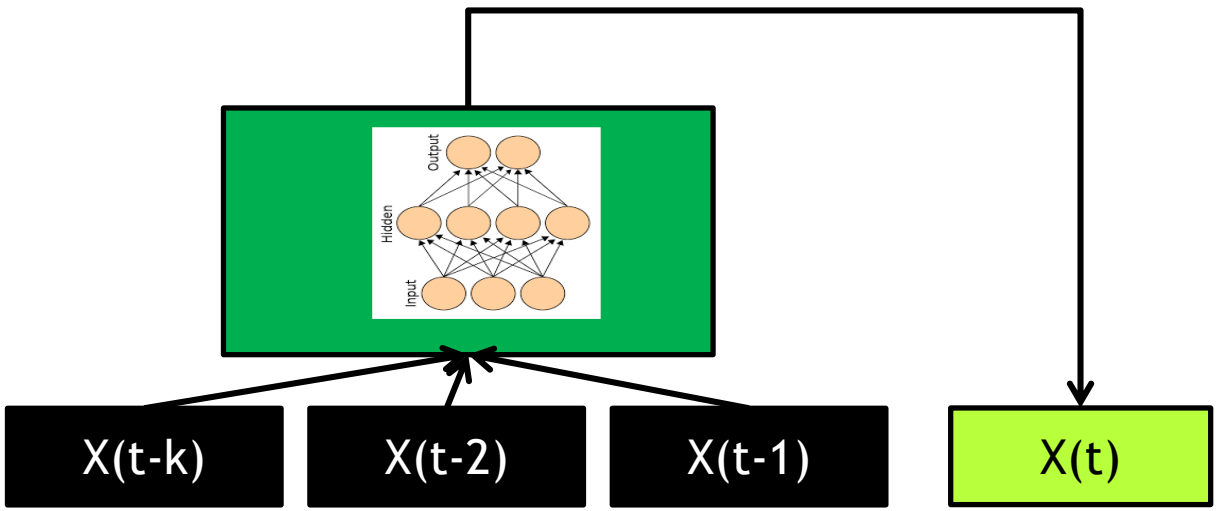
# Practice

Whale Sound Classification

# Natural Language Processing with RNN

# Feed Forward NN – based Language Model

Feed-forward NN (e.g. Conv. NN) can be used to predict next word from *k* previous words




Dictionary= 100,000 ➔ net has 100,000 outputs. Can we scale NN to 100,000 outputs? Is there a better way to deal with such a large number of outputs?

# RNN - based Language Model

RNN allows to extend fixed input window used by TD-NN

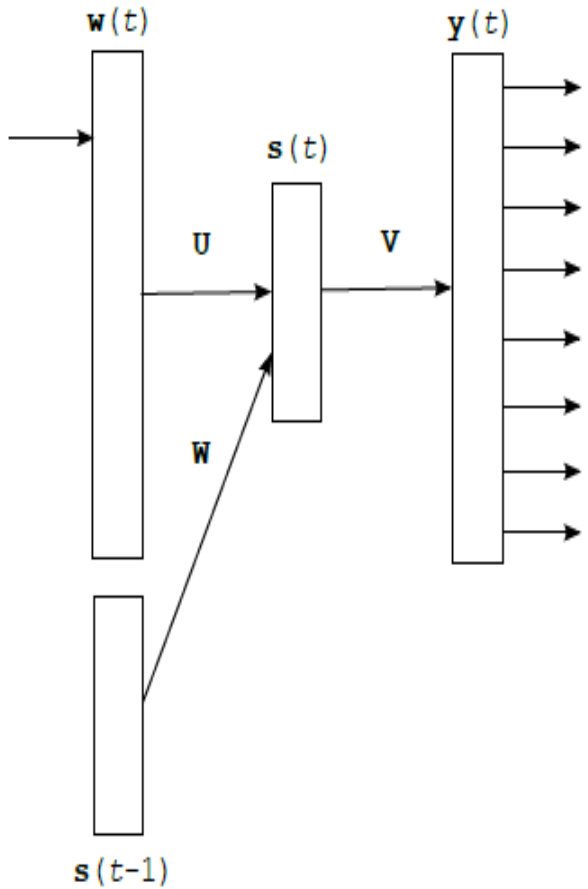
Input layer *w* and output layer *y* have the same dimensionality as the vocabulary (10K - 200K).

Hidden layer *s* is small (50 - 1000 neurons)

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right),$$

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

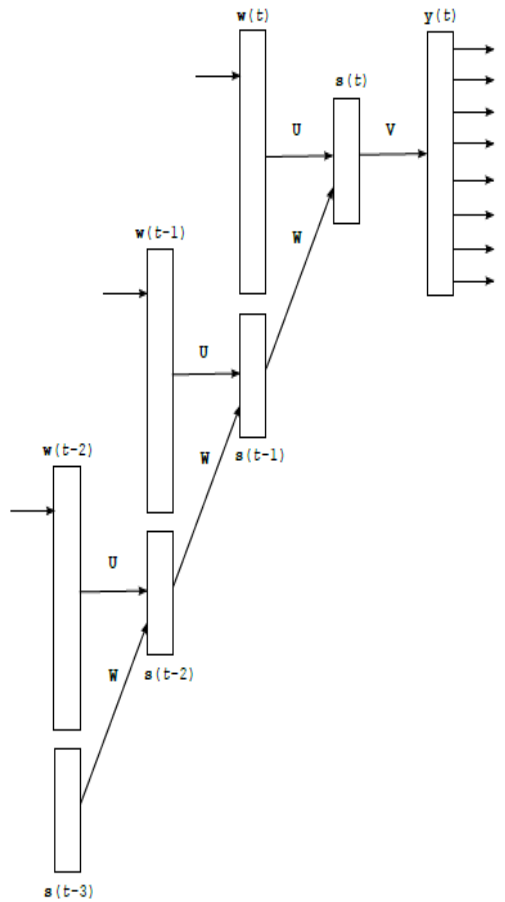



Network parameters:
- U - weights between input and hidden layer
- V - weights between hidden and output layer
- W – recurrent weights

# Training of RNN Language Model

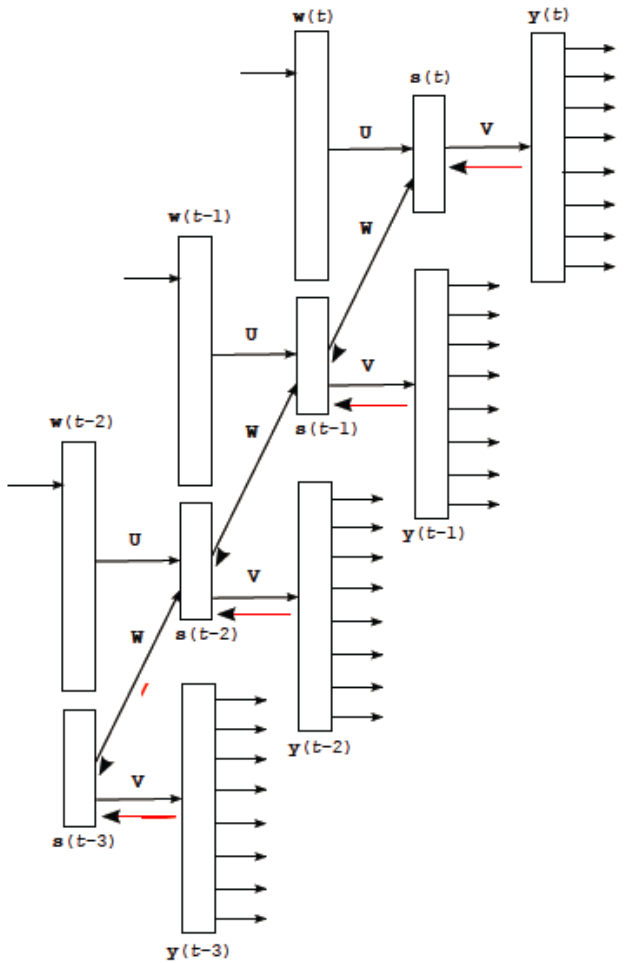
RNN is trained by unfolding in time and training as a deep feed-forward NN.

Gradients are propagated back through time



# Training of RNN Language Model

Example: Batch training on input sequence



# RNN vs n-gram language models

5-gram: IN TOKYO FOREIGN EXCHANGE TRADING YESTERDAY THE UNIT INCREASED AGAINST THE DOLLAR
RNNLM: IN TOKYO FOREIGN EXCHANGE TRADING YESTERDAY THE YEN INCREASED AGAINST THE DOLLAR

5-gram: SOME CURRENCY TRADERS SAID THE UPWARD REVALUATION OF THE GERMAN MARK WASN'T BIG ENOUGH AND THAT THE MARKET MAY CONTINUE TO RISE
RNNLM: SOME CURRENCY TRADERS SAID THE UPWARD REVALUATION OF THE GERMAN MARKET WASN'T BIG ENOUGH AND THAT THE MARKET MAY CONTINUE TO RISE

5-gram: MEANWHILE QUESTIONS REMAIN WITHIN THE E. M. S. WEATHERED YESTERDAY'S REALIGNMENT WAS ONLY A TEMPORARY SOLUTION
RNNLM: MEANWHILE QUESTIONS REMAIN WITHIN THE E. M. S. WHETHER YESTERDAY'S REALIGNMENT WAS ONLY A TEMPORARY SOLUTION

5-gram: MR. PARNES FOLEY ALSO FOR THE FIRST TIME THE WIND WITH SUEZ'S PLANS FOR GENERALE DE BELGIQUE'S WAR
RNNLM: MR. PARNES SO LATE ALSO FOR THE FIRST TIME ALIGNED WITH SUEZ'S PLANS FOR GENERALE DE BELGIQUE'S WAR

5-gram: HE SAID THE GROUP WAS MARKET IN ITS STRUCTURE AND NO ONE HAD LEADERSHIP
RNNLM: HE SAID THE GROUP WAS ARCANE IN ITS STRUCTURE AND NO ONE HAD LEADERSHIP

WSJ-20K, Kaldi

# Word Embedding

The classical NLP regards words as atomic symbols. Each word is represented as *one-hot* vector [0000010000]

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] **AND**
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Word embedding - a word is represented as a dense vector

*linguistics* =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

# Code a word through its context

You can get a lot of value by representing a word by means of its neighbors

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

One of the most successful ideas of modern statistical NLP

| | | |
|---|---|---|
| government debt problems turning into | banking | crises as has happened in |
| saying that Europe needs unified | banking | regulation to replace the hodgepodge |

↖ These words will represent *banking* ↗

# NN to learn word embedding

**The basic idea:**

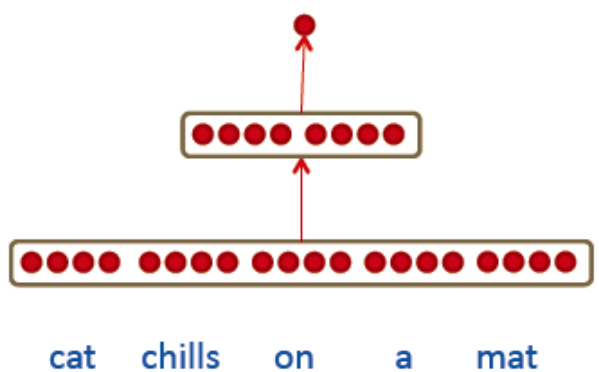
1. define score function that is 1 on true sequence and 0 on corrupted:
   score(*cat chills on a mat*) -> 1
   score(*cat chills Jeju a mat*) -> 0
2. Each word is associated with vector $R^n$ : [●●●●]
3. Build and train NN

$$s = U^T f(Wx + b) \qquad x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8\times 20}, U \in \mathbb{R}^{8\times 1}$$

$$\begin{aligned} s &= U^T a \\ a &= f(z) \\ z &= Wx + b \end{aligned}$$

$$x = [x_{cat} \quad x_{chills} \quad x_{on} \quad x_a \quad x_{mat}]$$

$$L \in \mathbb{R}^{n\times|V|}$$

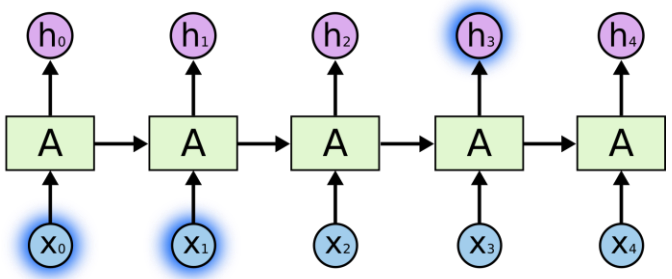


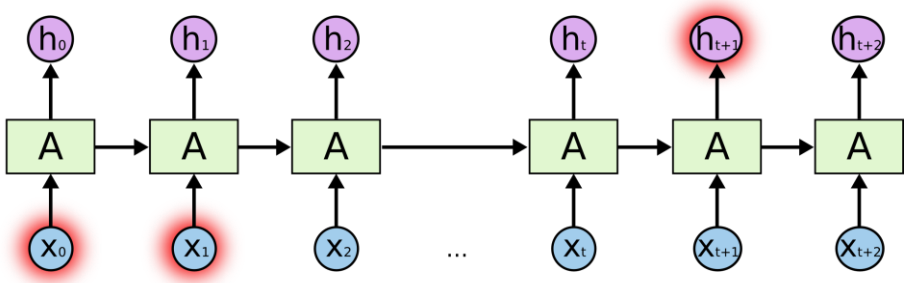
# Practice

RNN for Natural Language Processing

# Long-Short Temporal Memory RNN

# Problem of long term dependencies




RNN has *short memory*: the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially:

$$h(1) = W_1 * \boldsymbol{x(1)} + W_2 * h(0)$$

$$h(2) = W_1 * x(2) + W_2 * h(1)$$

$$= W_1 * x(2) + W_2 * (W_1 * x(1) + W_2 * h(0))$$

$$= W_1 * x(2) + \boldsymbol{W_2 * W_1 * x(1)} + W_2^2 * h(0)$$

$$h(k) = W_1 * x(k) + W_2 * W_1 * x(k-1) + \ldots$$

$$+ \boldsymbol{W_2^{k-1} * W_1 * x(1)} + W_2^k * h(0)$$

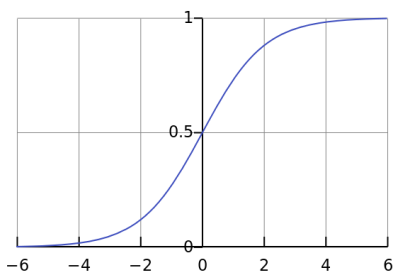
# Long Short-Term Memory (LSTM)

**Long** Short-Term Memory (LSTM) architecture was introduced to address RNN short memory. It has a n new basic element - *memory cell*

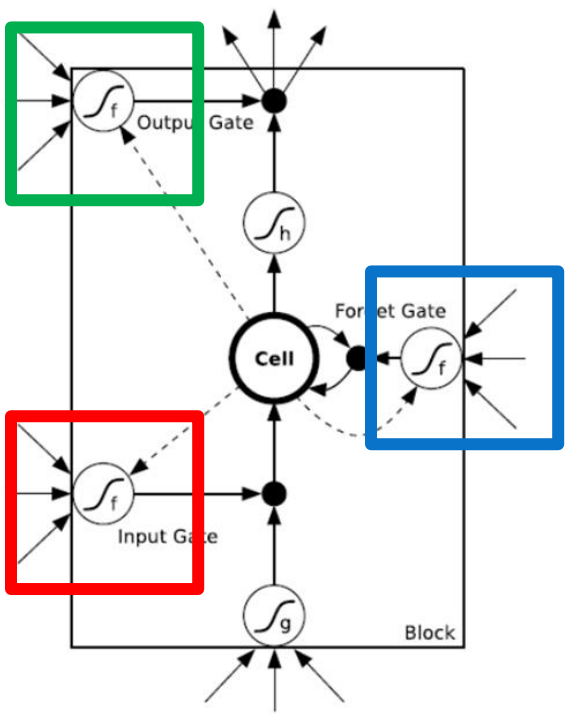
*All gates use* logistic sigmoid:

0 - gate closed,

1 - gate open




**Output gate** **"switch-off"** output of the cell

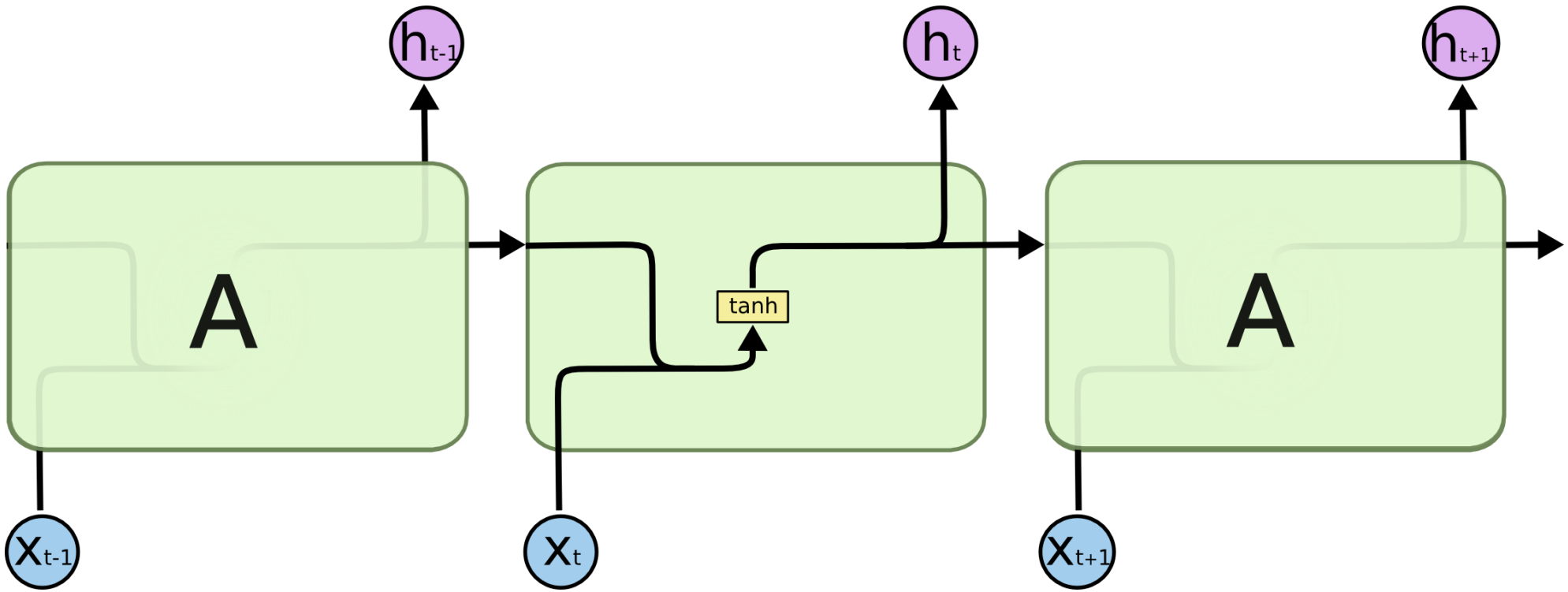



**Input gate** cut-off the input of the cell

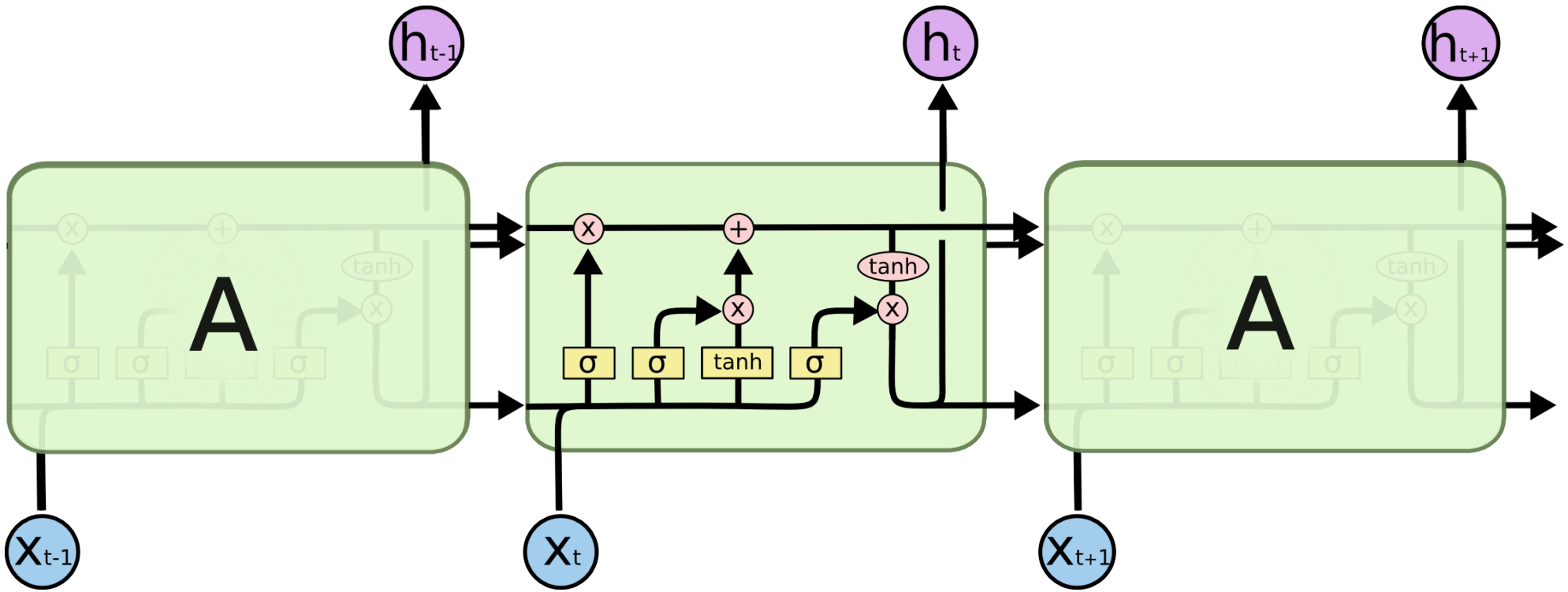
**Forget gate** reset the cell's state

*Hochreiter and Schmidhuber, 1997.*

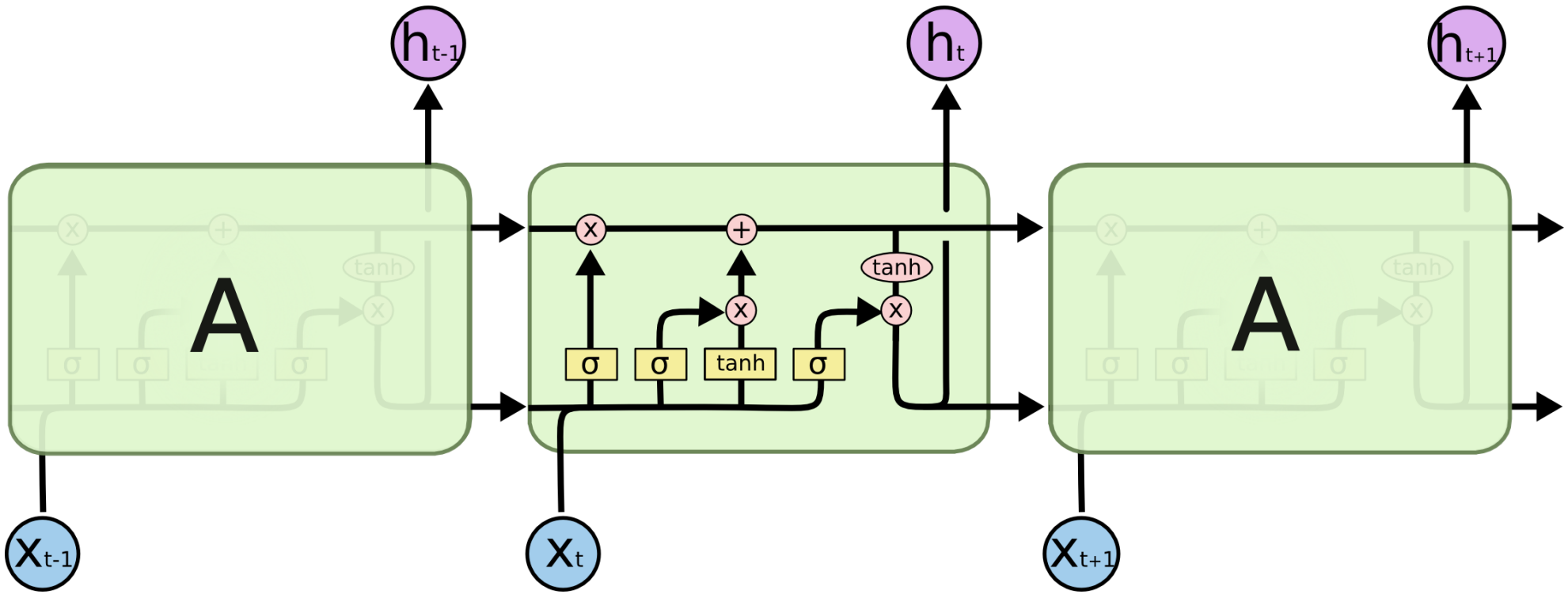
# RNN Sequence Representation



# LSTM representation

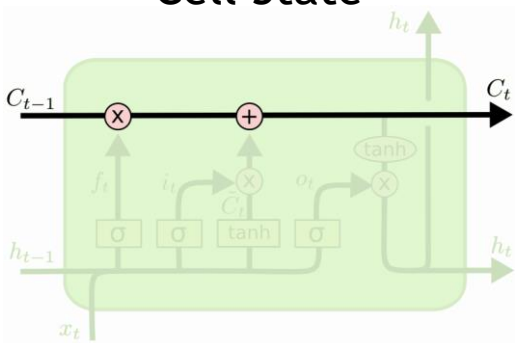


# LSTM representation

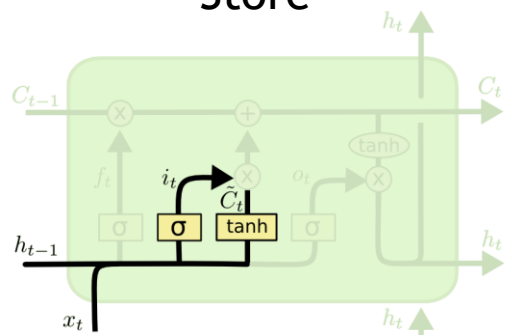


# LSTM Operation

## Cell State

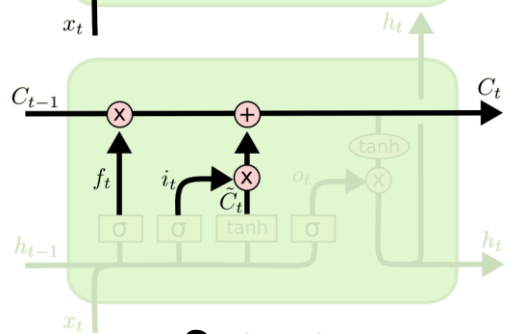



## Store



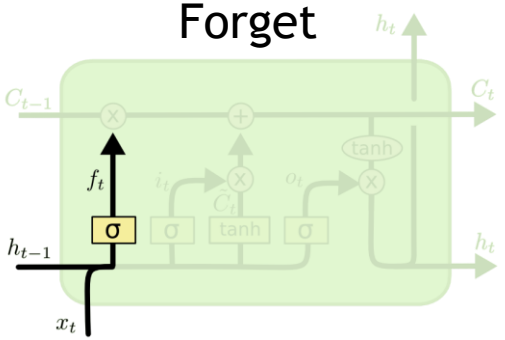

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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



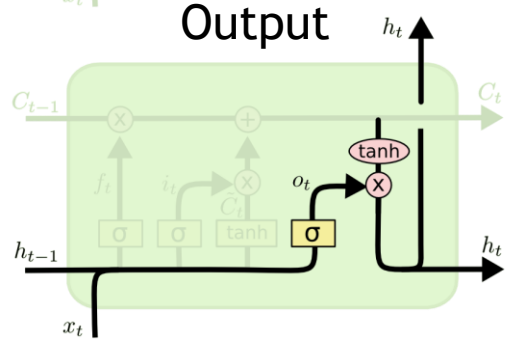

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

## Forget




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

## Output

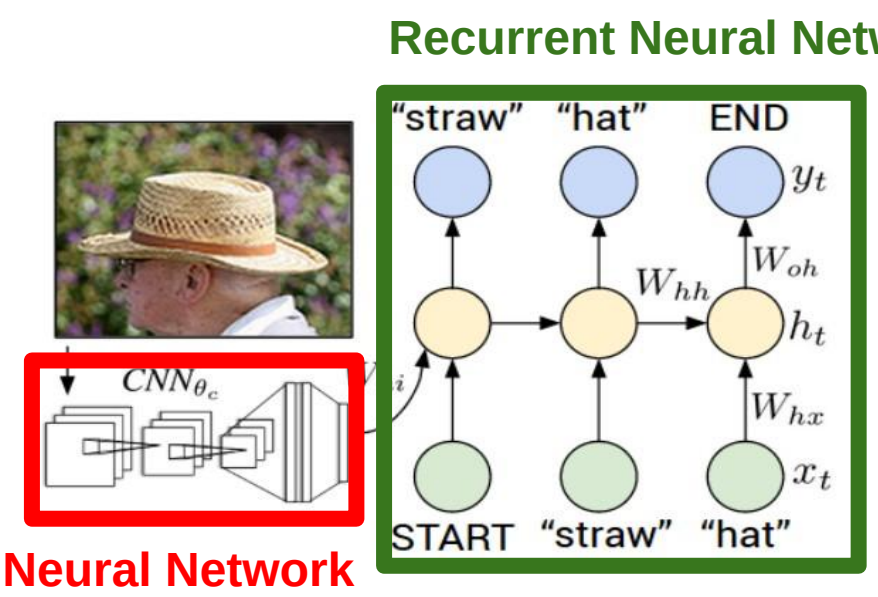



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

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

# Image Captioning

# Image Captioning




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

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

FC-1000

softmax

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

FC-1000

softmax

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

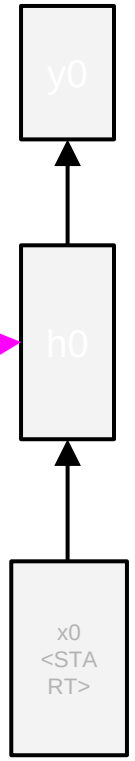
FC-4096

FC-4096

test image

x0
<STA
RT>


test image




**before:**

h = tanh(Wxh * x + Whh * h)

**now:**

h = tanh(Wxh * x + Whh * h + **Wih * v**)

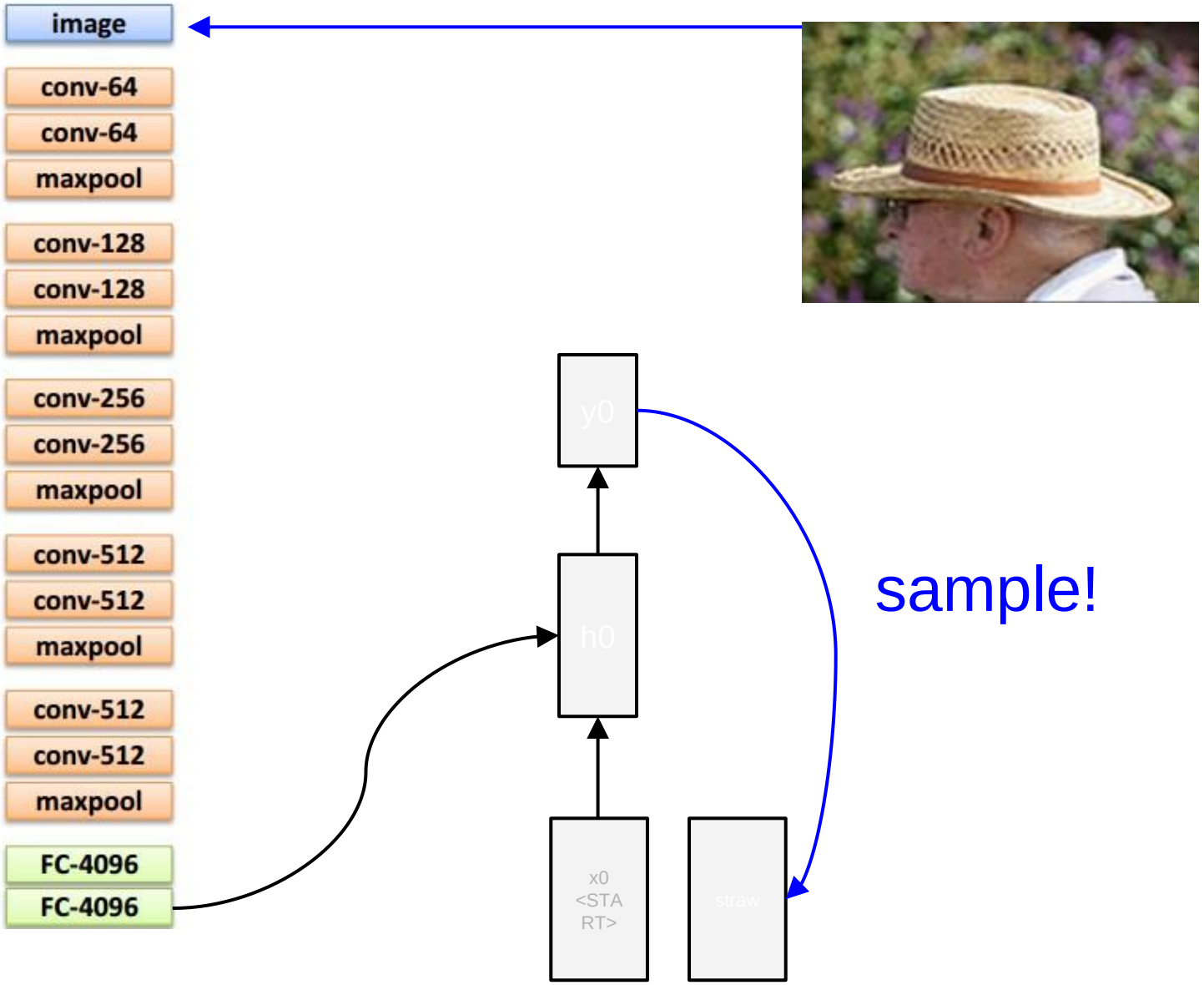


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
y0
h0
x0
<STA
RT>
straw
test image
sample!

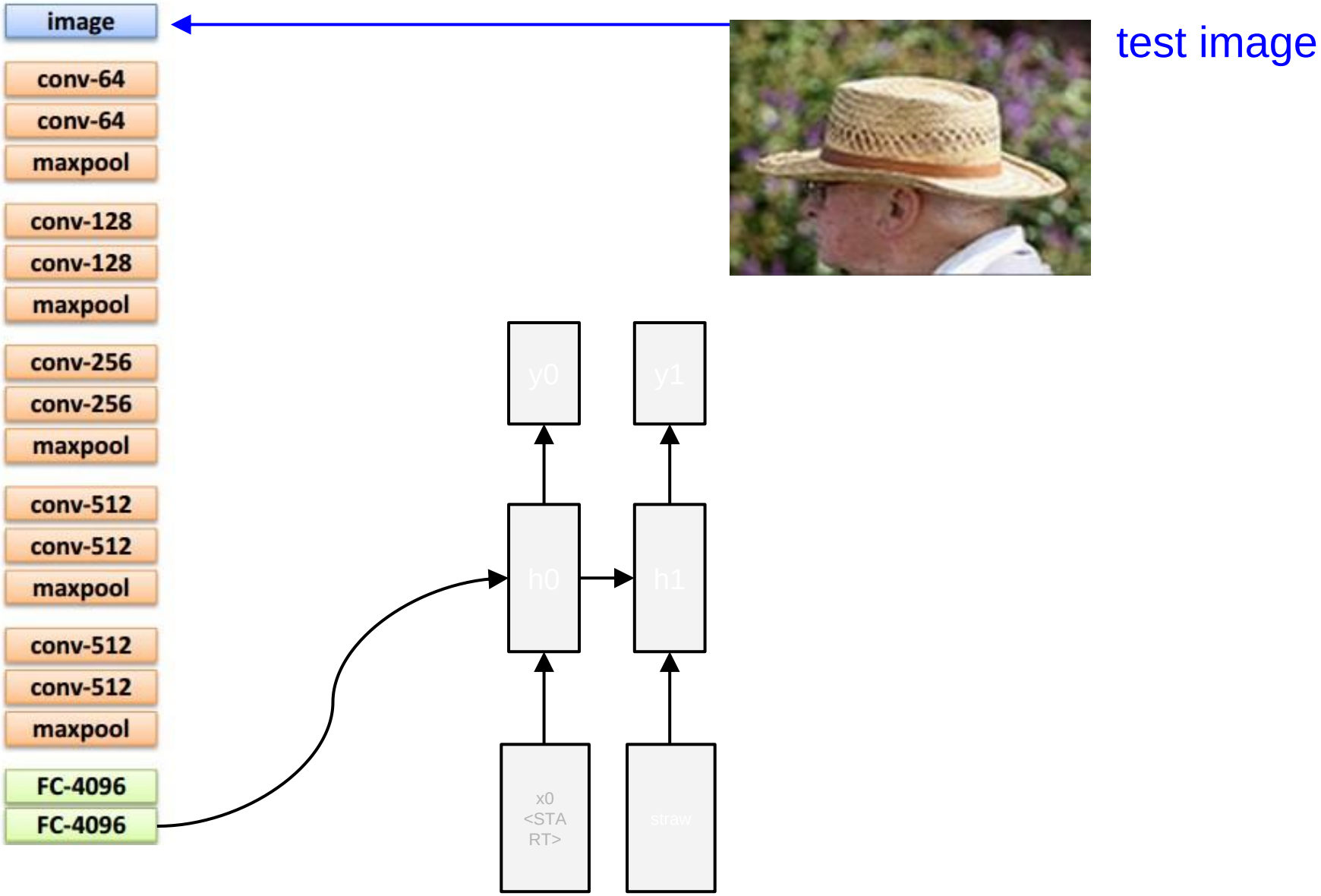


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<STA
RT>
straw

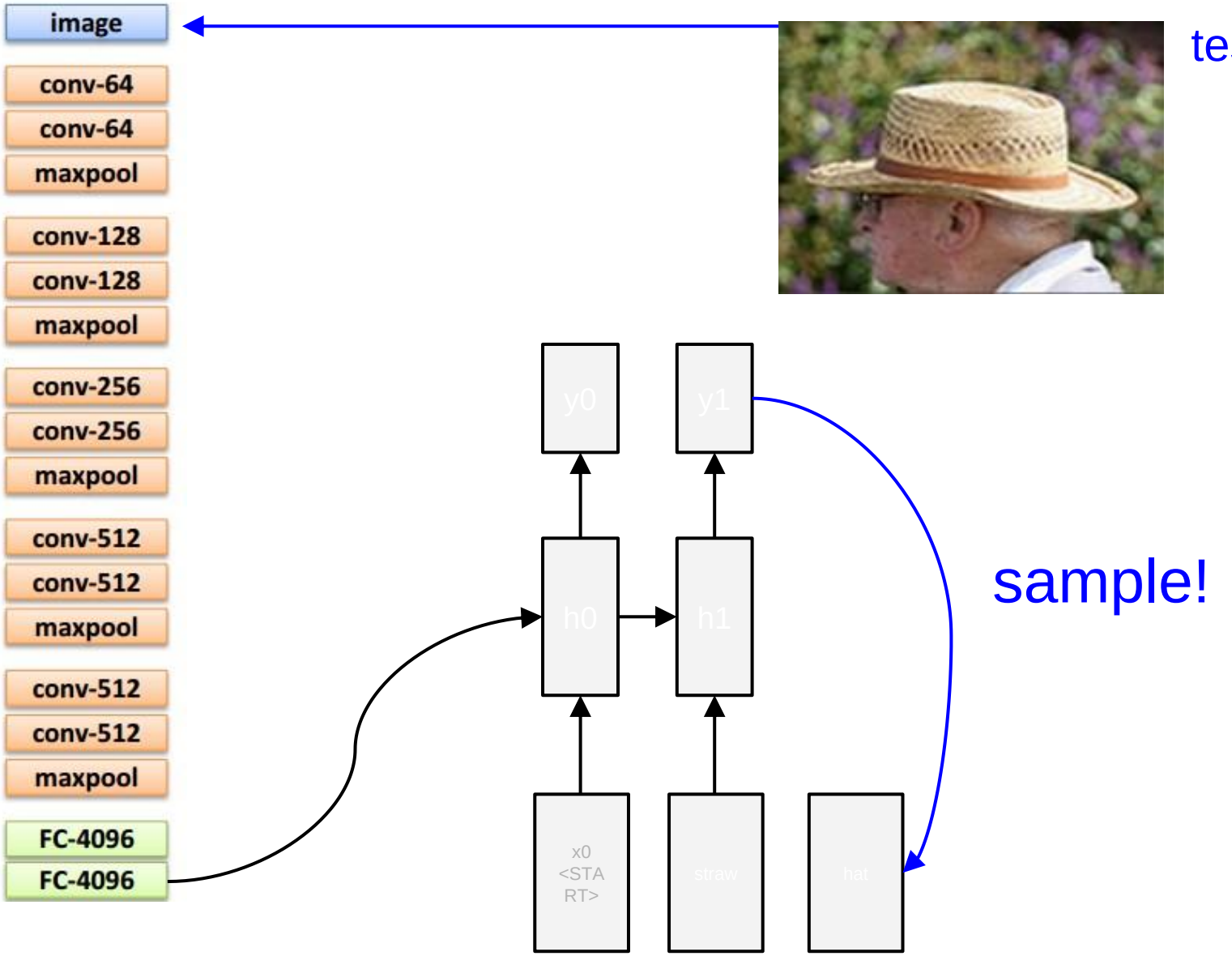


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<STA
RT>
straw
hat
sample!

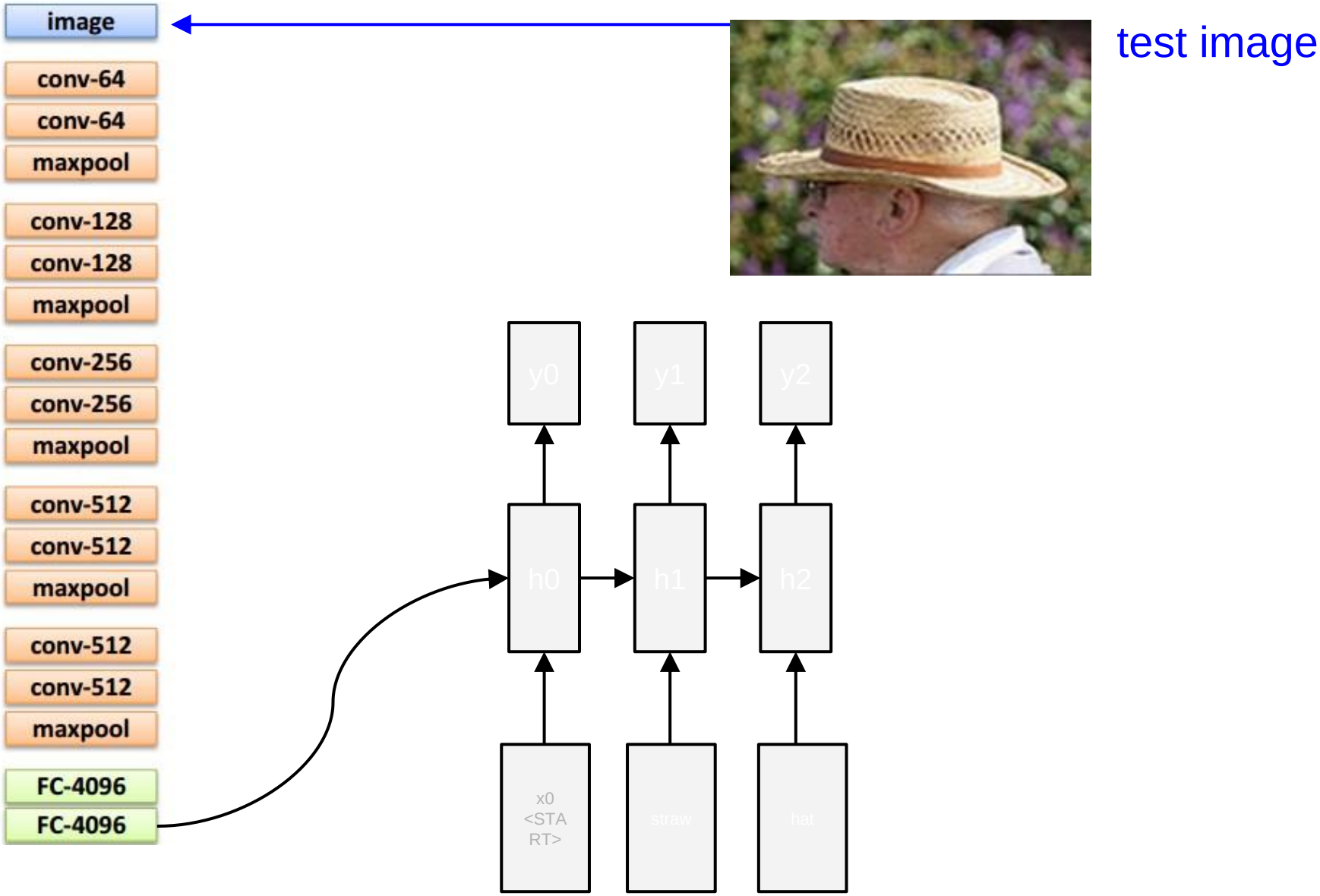


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0
<STA
RT>
straw
hat

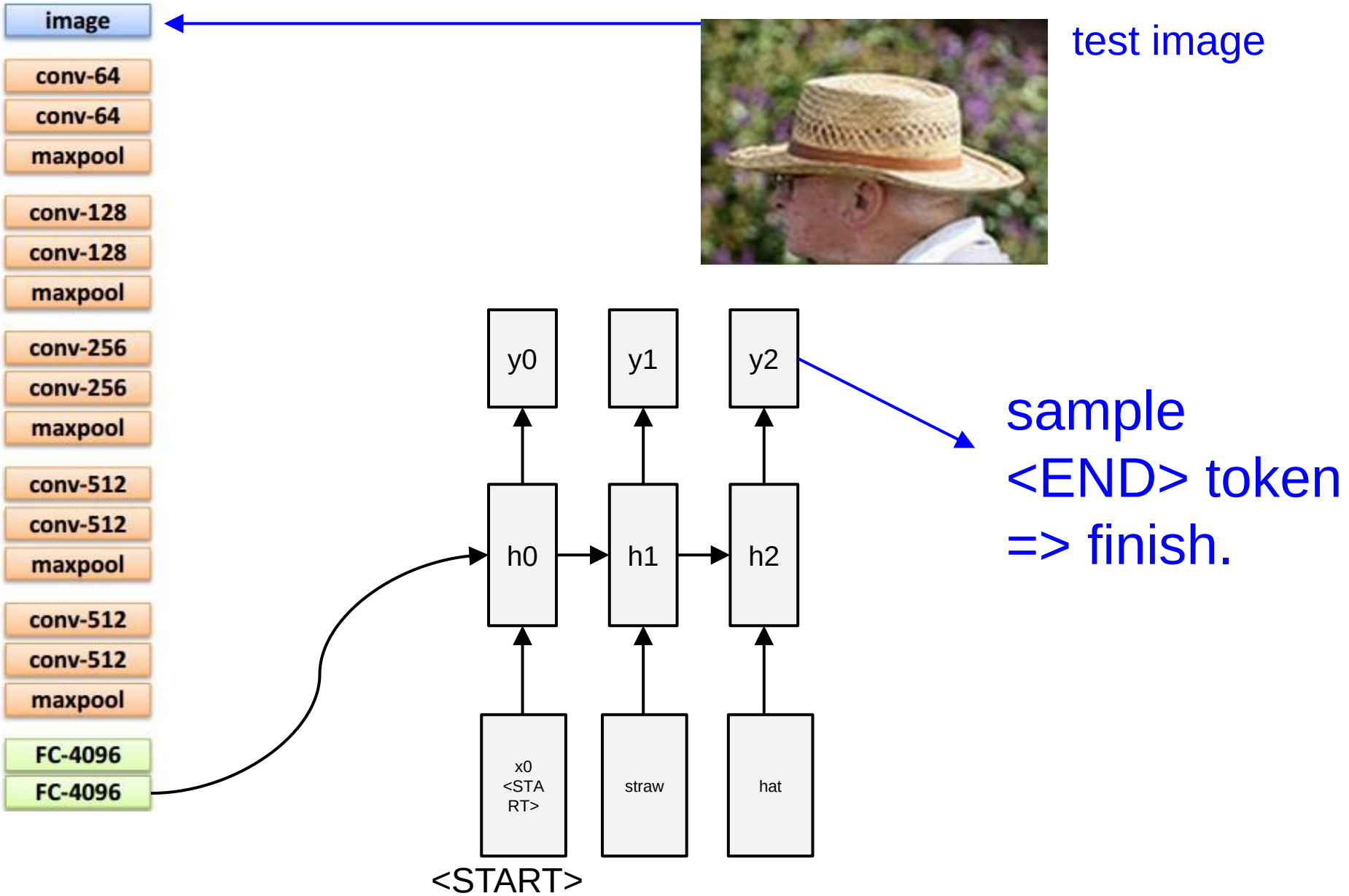
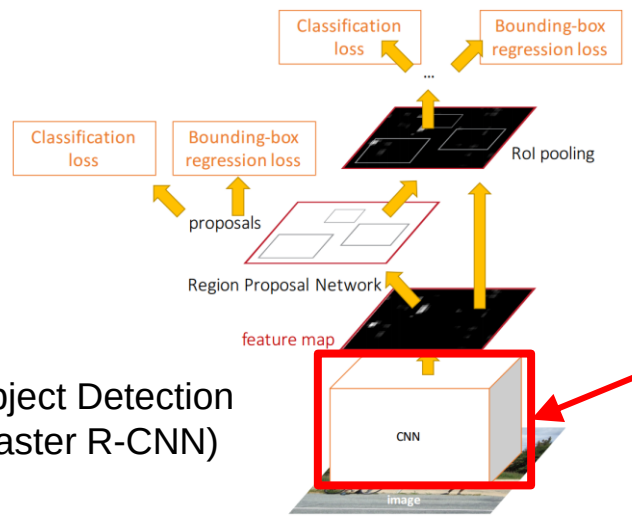


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>
sample
<END> token
=> finish.

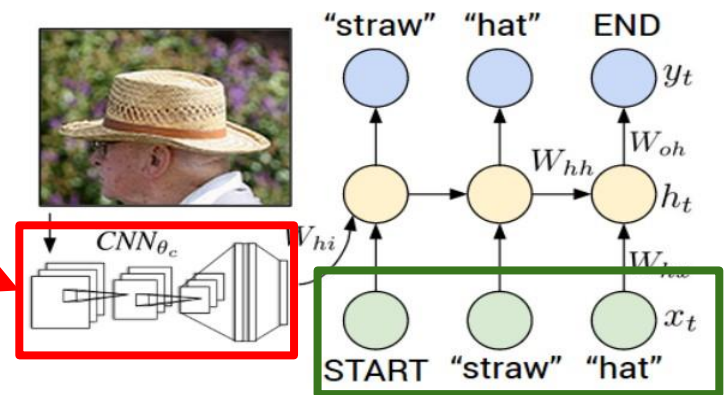
# Transfer learning with CNNs is pervasive…




Object Detection
(Faster R-CNN)

CNN pretrained
on ImageNet

Image Captioning: CNN + RNN




Word vectors pretrained
from word2vec

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

mscoco.org

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

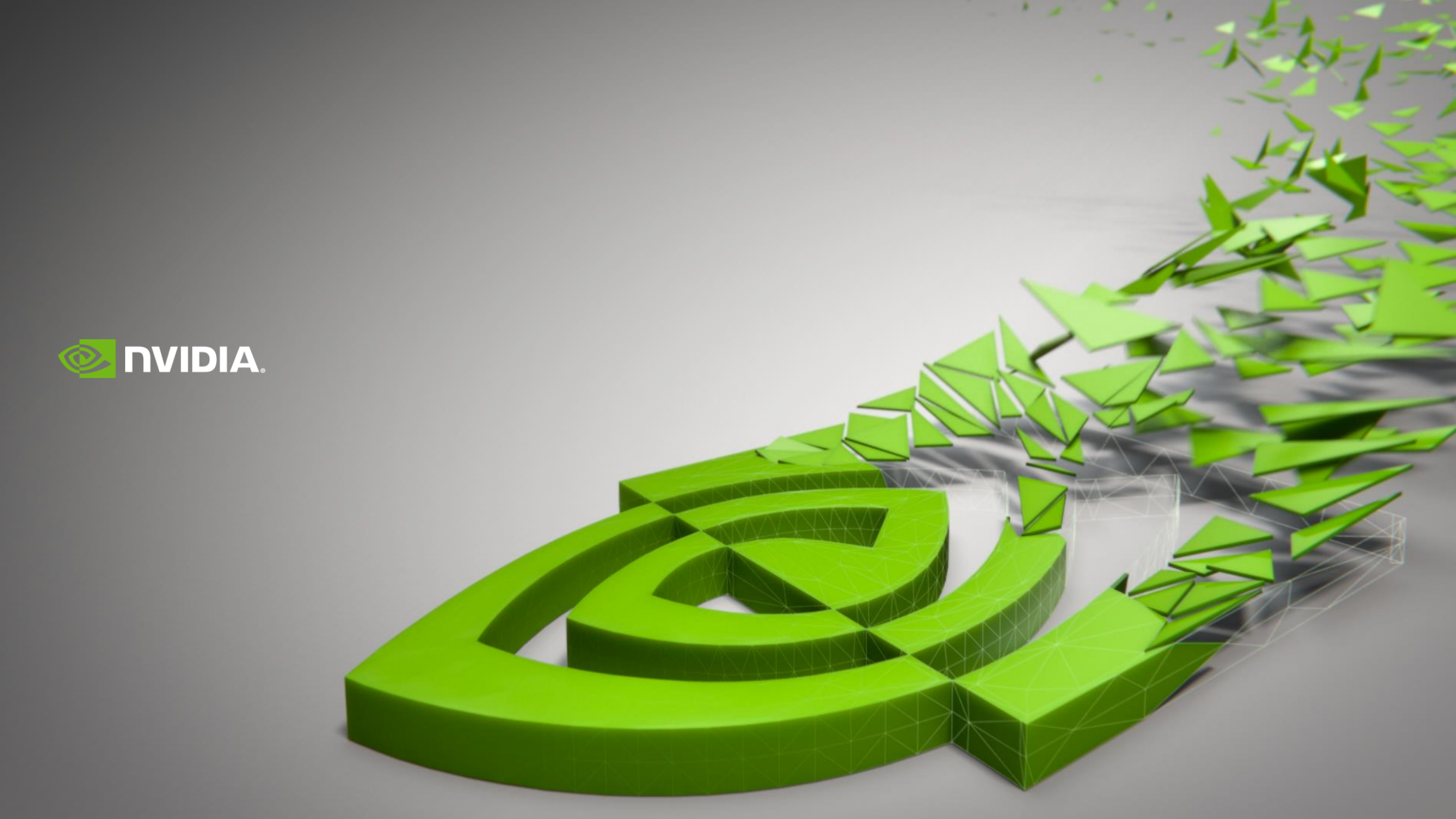
"boy is doing backflip on wakeboard."

"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."


"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."


NVIDIA