THE UNIVERSITY OF BRITISH COLUMBIA

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

Lecture 8: RNNs (cont.) + Applications

Course **Logistics**

- Assignment 1 grades are (finally) available
  - Part 1 solutions are available on Piazza
  - Part 2 solutions will be out soon
- Assignment 3 is out, due **Wednsday, February 7th**
- **Questions** on Assignment 3?
- Paper choices (google form) will is due **Wednsday** THIS week **(Jan 31st)**

- **Projects** pitches we will do during next class

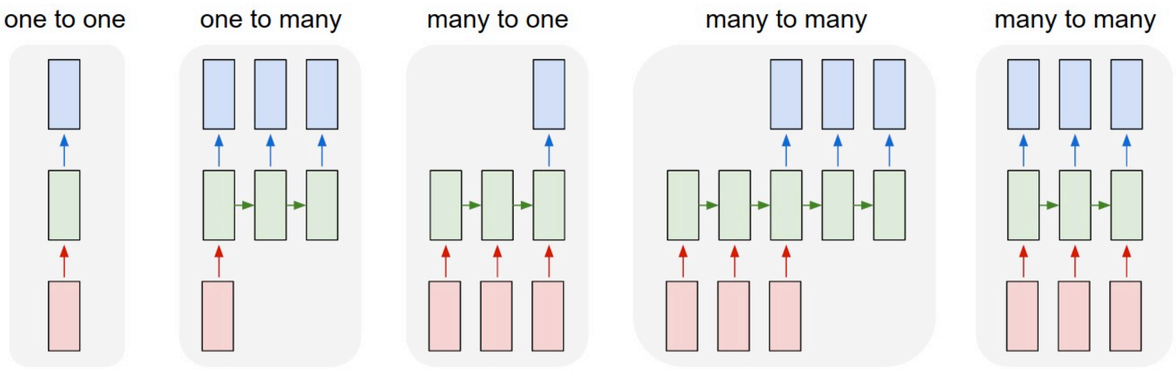
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



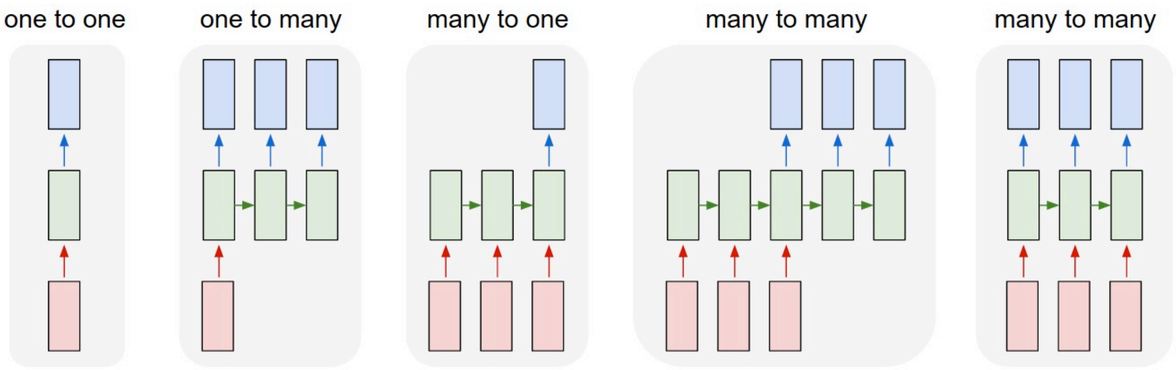
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



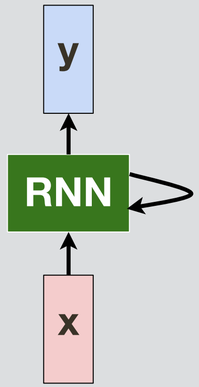
Vanilla **RNN**

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

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

$$\downarrow$$

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



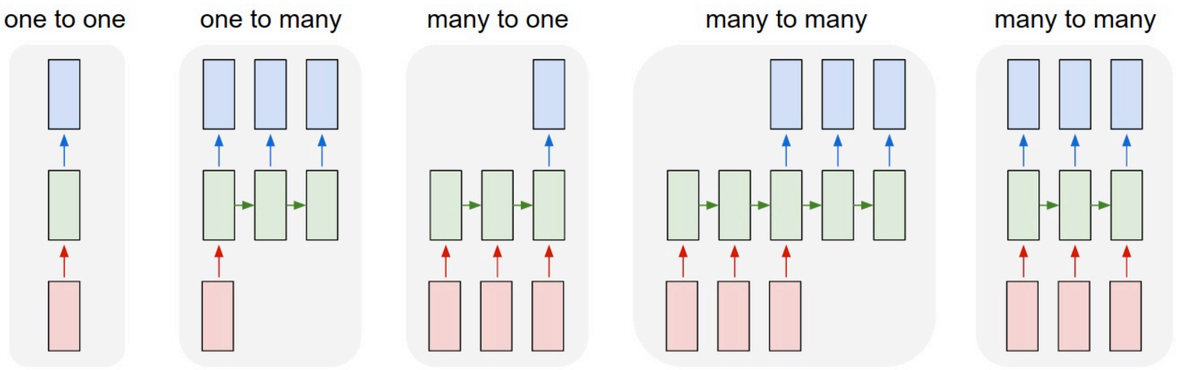
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



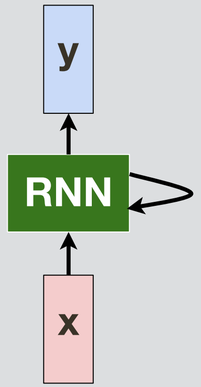
Vanilla **RNN**

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

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

$$\downarrow$$

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



Vanishing or **Exploding** Gradients

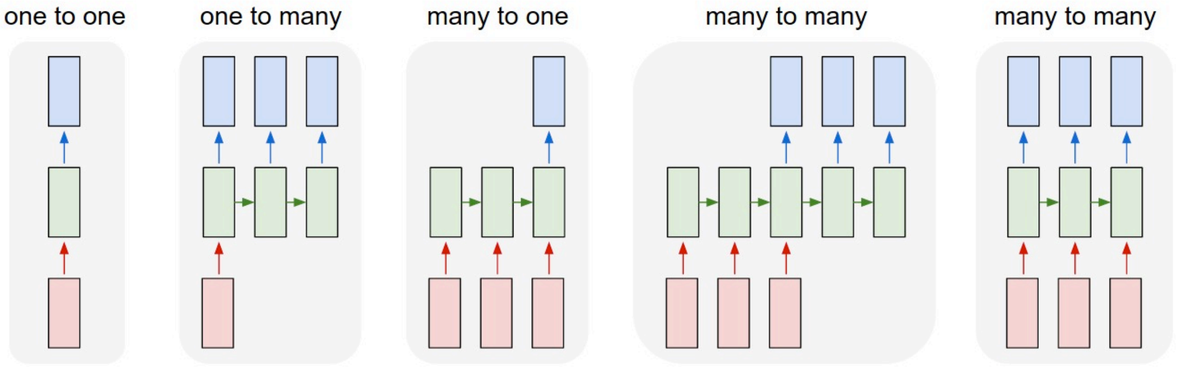
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



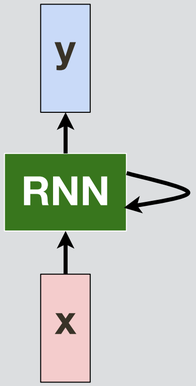
Vanilla **RNN**

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

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

$$\downarrow$$

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



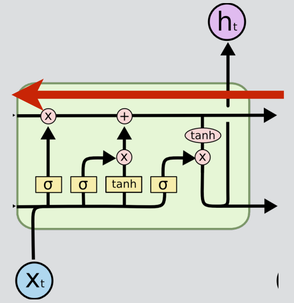
Vanishing or **Exploding** Gradients

Long-Short Term Memory (**LSTM**)

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

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

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



Uninterrupted gradient flow!

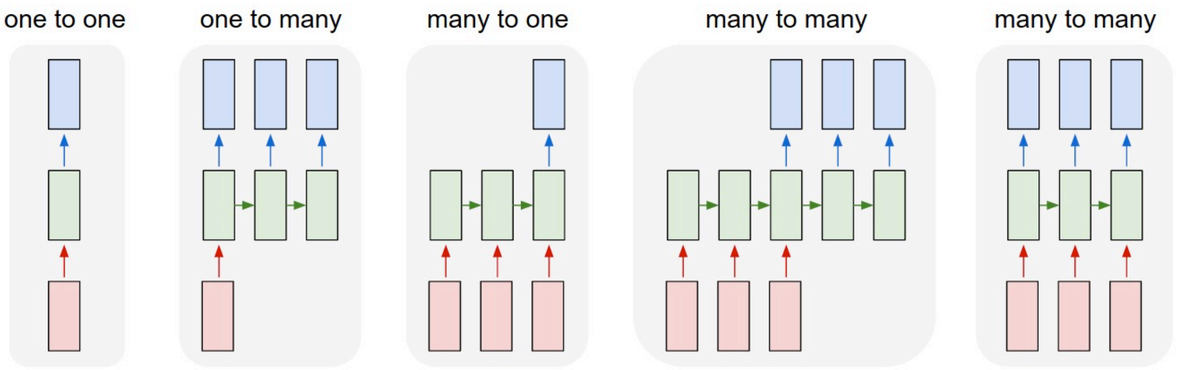
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



Loss functions: often cross-entropy (for classification); could be max-margin (like in SVM) or Squared Loss (regression)

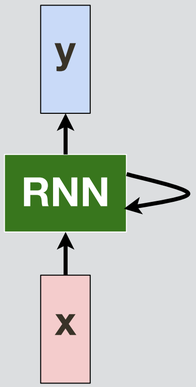
Vanilla **RNN**

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

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

$$\downarrow$$

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



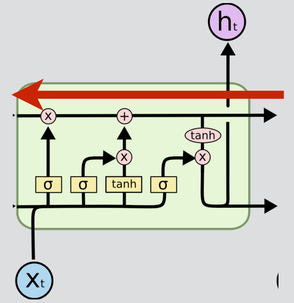
Vanishing or **Exploding** Gradients

Long-Short Term Memory (**LSTM**)

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

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

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



Uninterrupted gradient flow!

Long-Short Term Memory (**LSTM**)

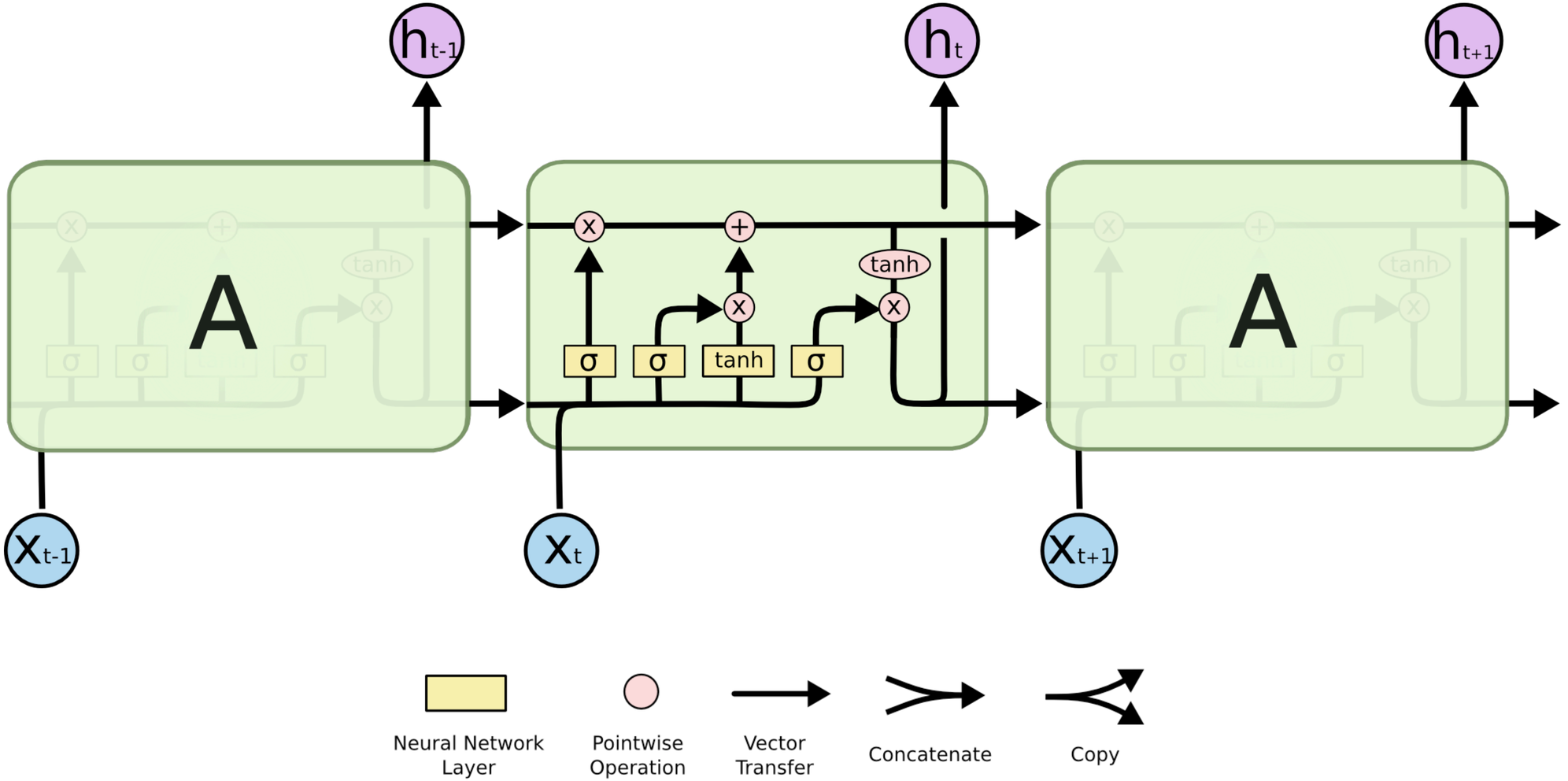


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

Long-Short Term Memory (**LSTM**)

Cell state / **memory**

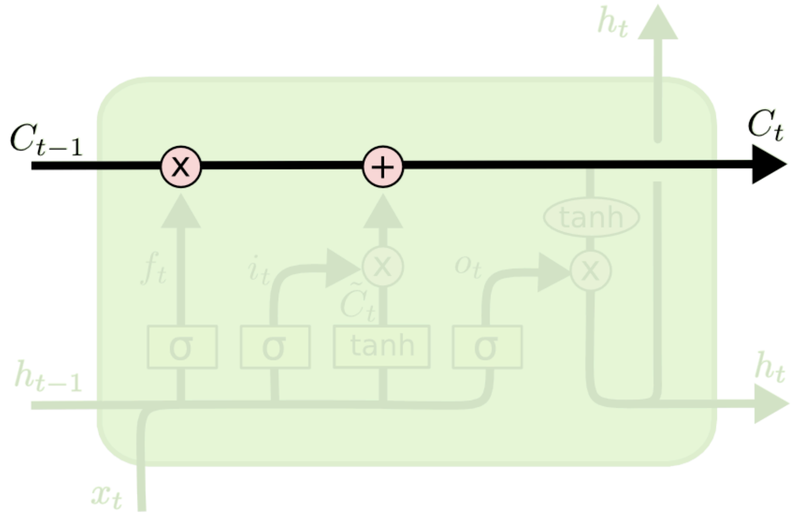
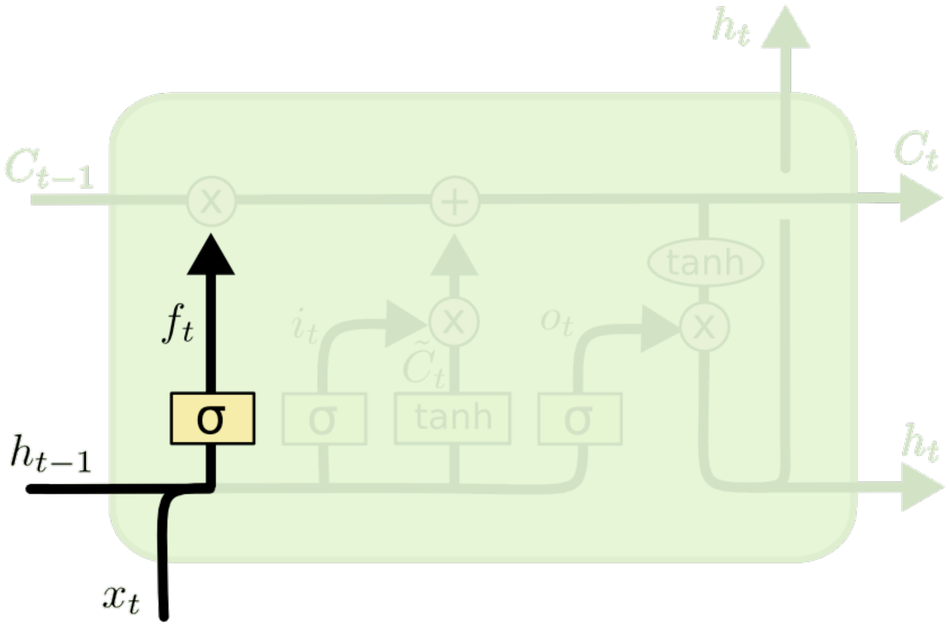


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Forget Gate

Should we continue to **remember** this “bit” of information or not?



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

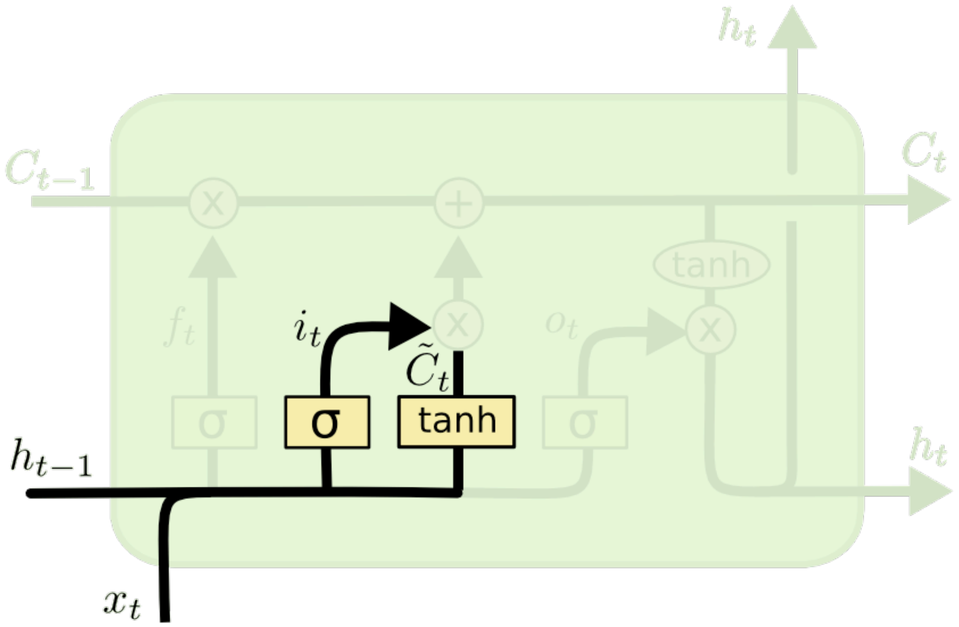
Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Input Gate

Should we **update** this “bit” of information or not?

If yes, then what should we **remember**?



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

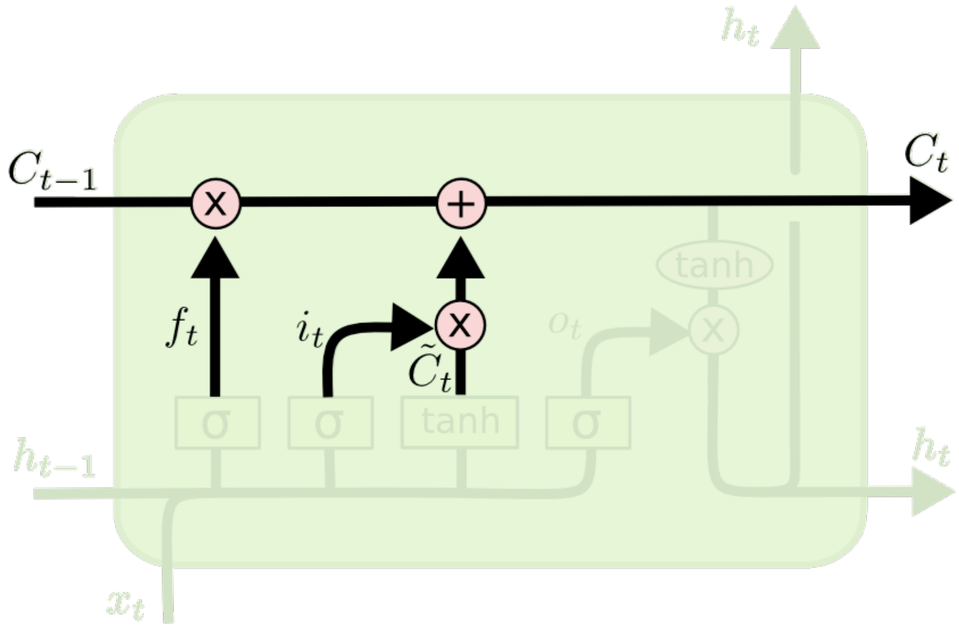
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Memory Update

Forget what needs to be forgotten + memorize what needs to be remembered



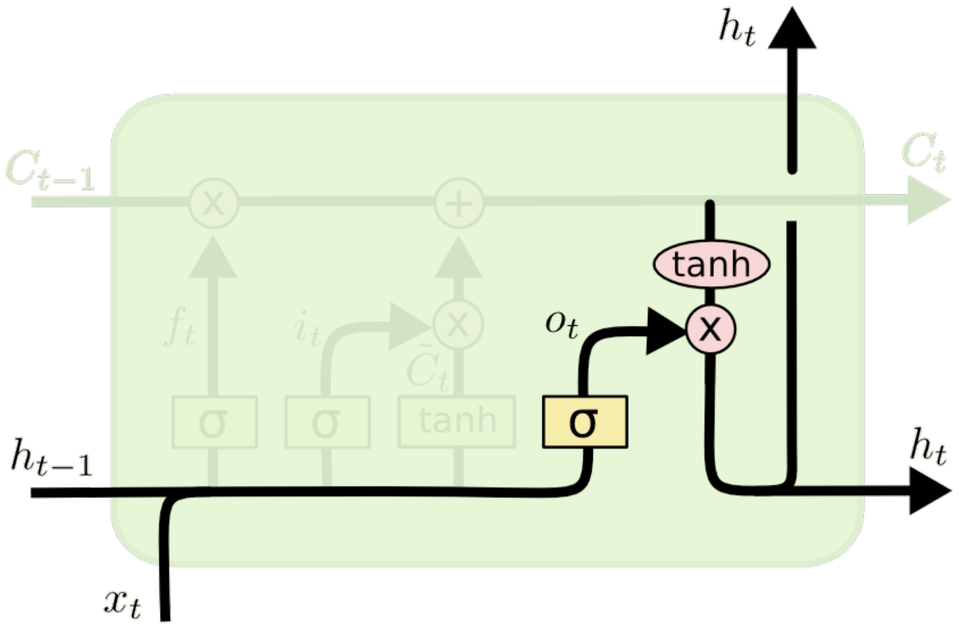
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Output Gate

Should we output this bit of information (e.g., to “deeper” LSTM layers)?



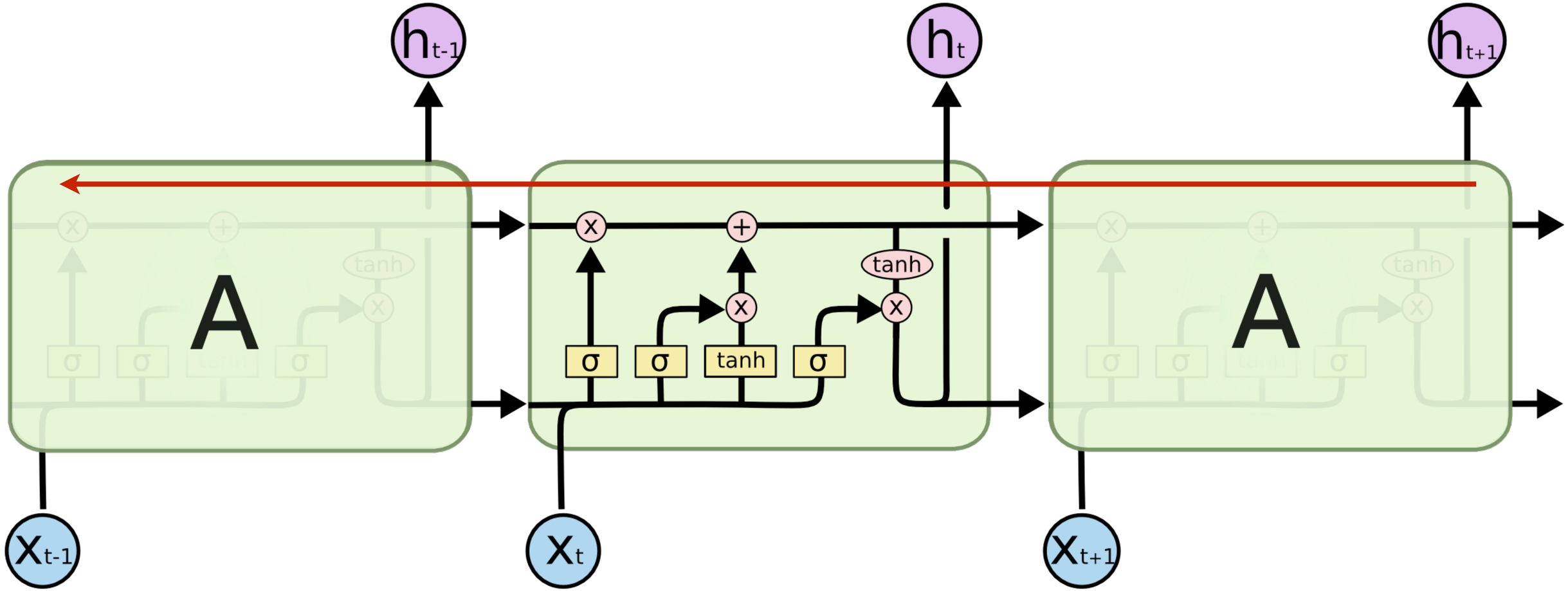
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Additive Updates

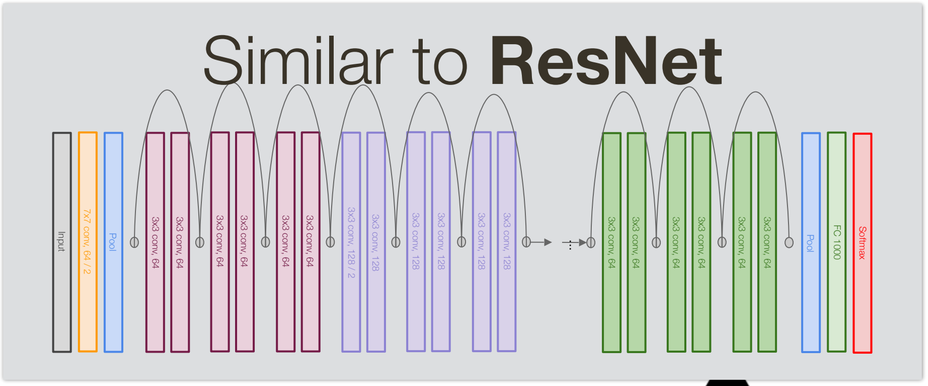


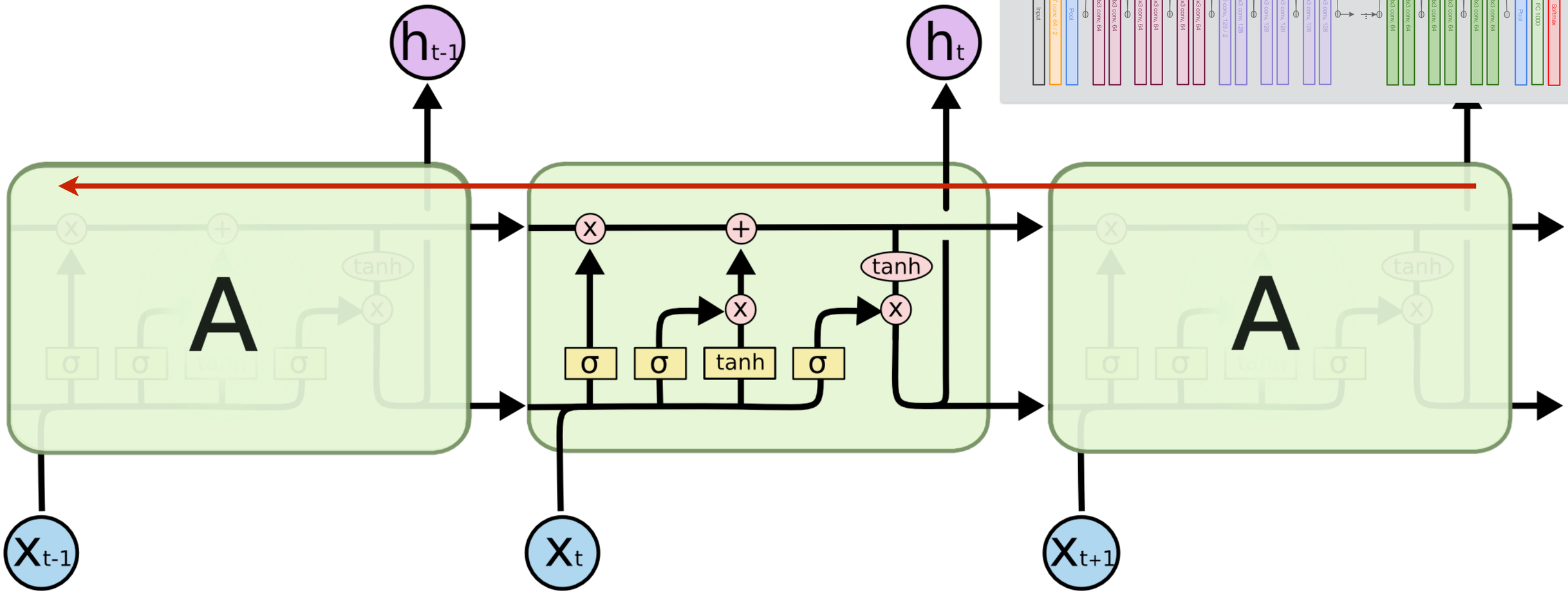
Uninterrupted gradient flow!

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Intuition: Additive Updates





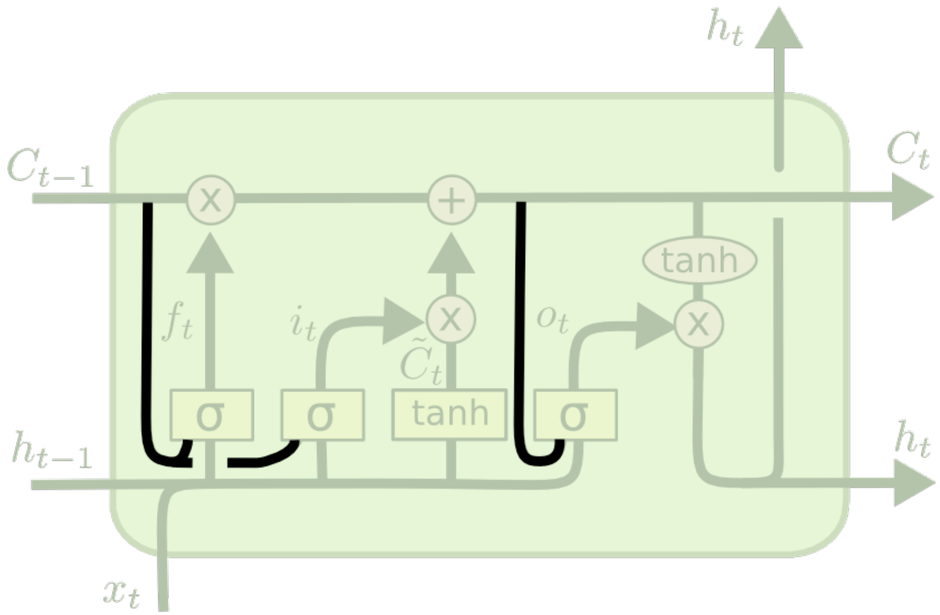
Uninterrupted gradient flow!

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Variants: with Peephole Connections

Lets gates see the cell state / memory



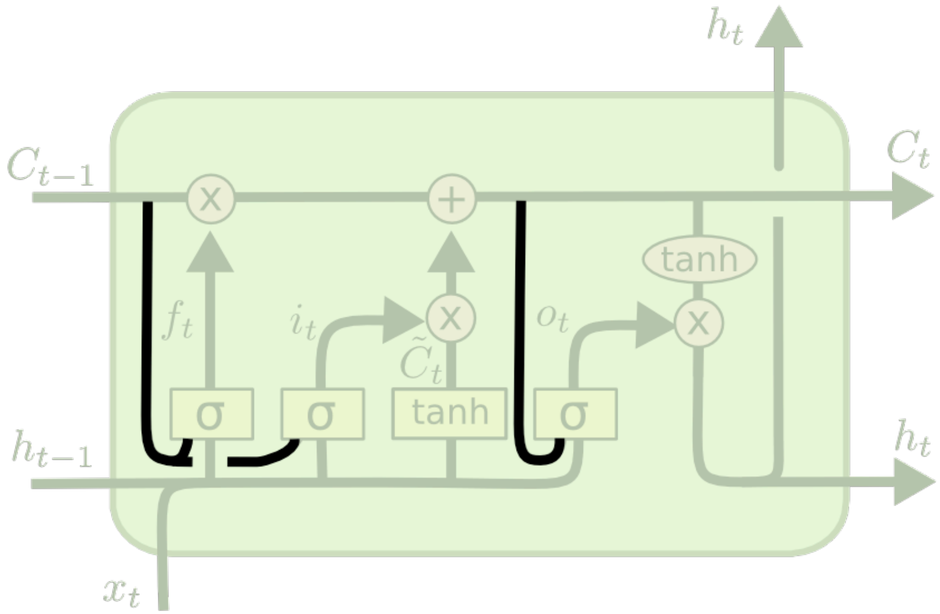
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Variants: with Peephole Connections

Lets gates see the cell state / memory



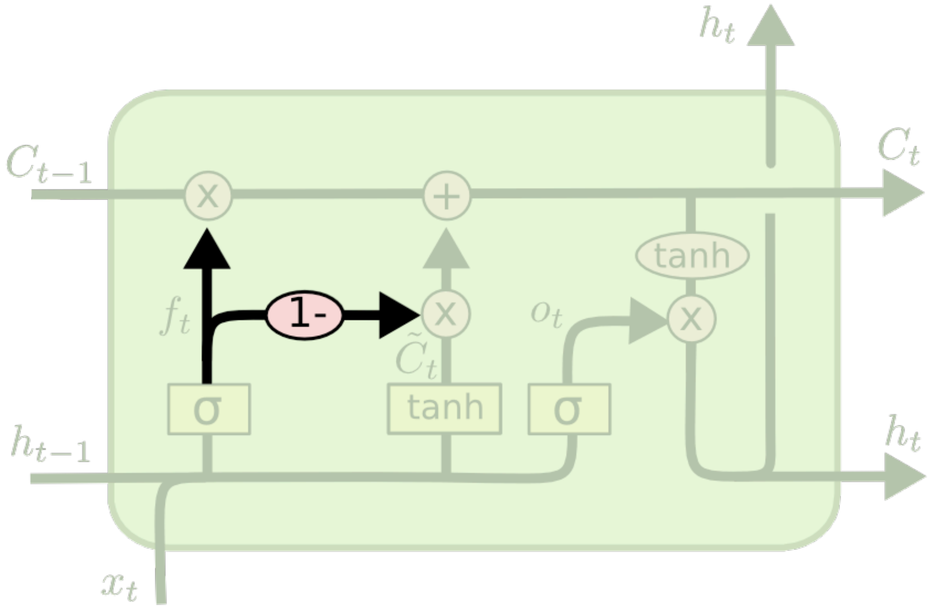
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM Variants: with Coupled Gates

Only memorize new information when you're forgetting old



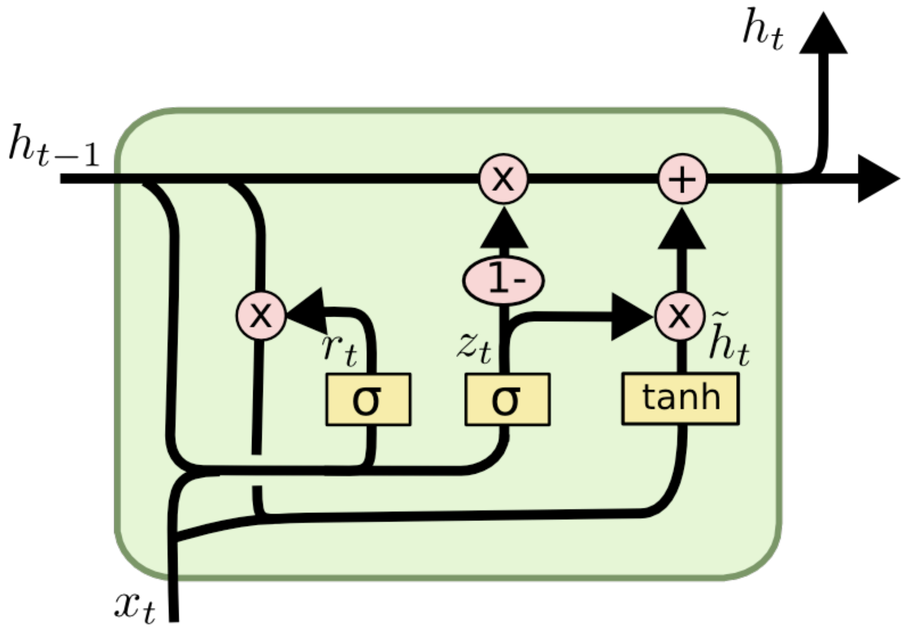
$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

Gated Recurrent Unit (**GRU**)

No explicit memory; memory = hidden output



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

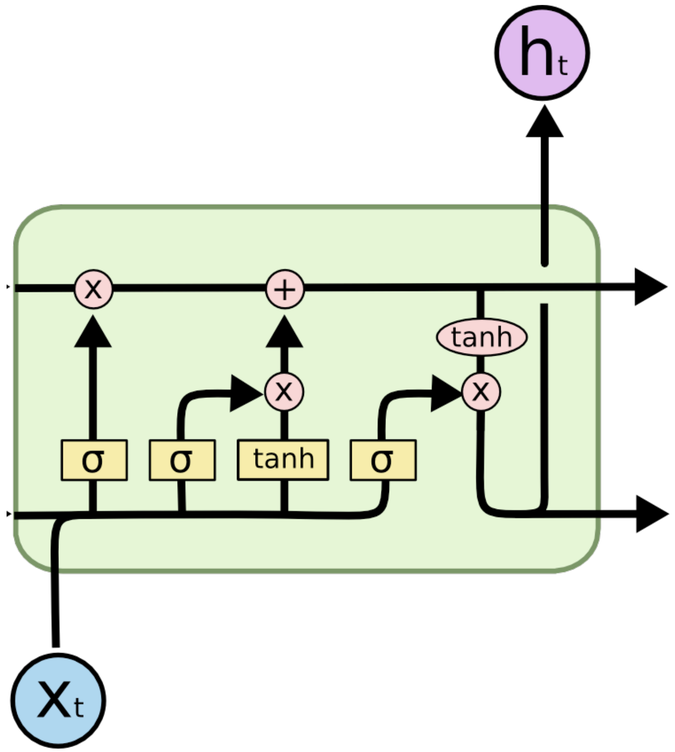
z = memorize new and forget old

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

* slide from Dhruv Batra

LSTM/RNN **Challenges**

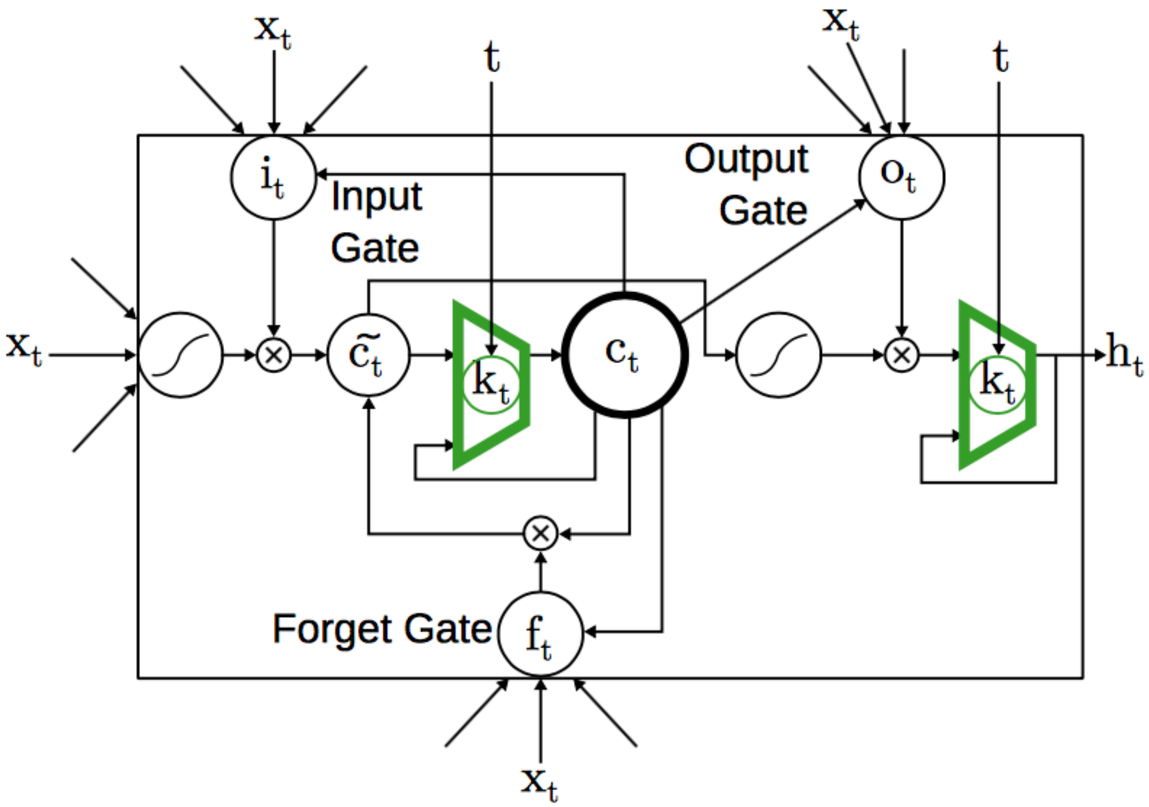
— LSTM can remember some history, but not too long
— LSTM assumes data is regularly sampled

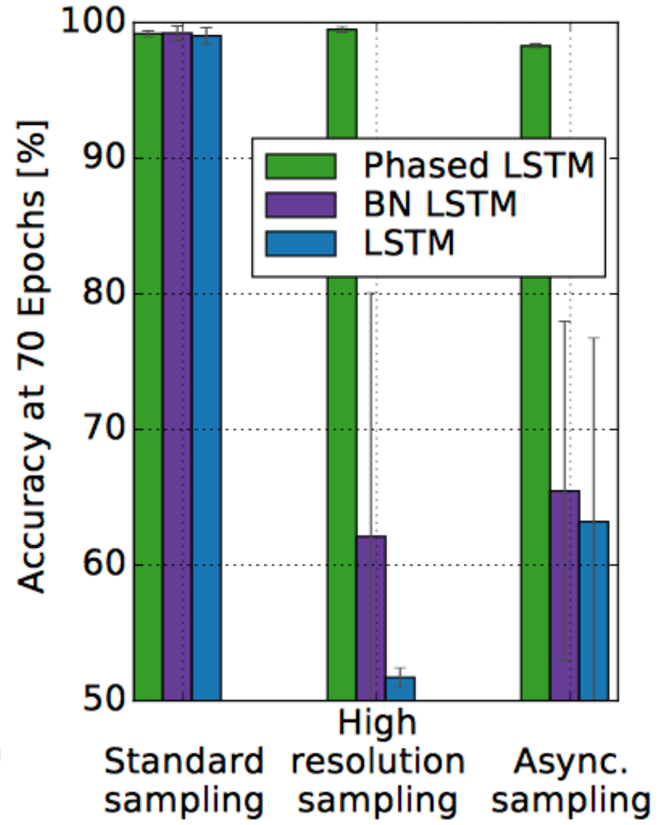


Phased LSTM

[Neil et al., 2016]

Gates are controlled by **phased** (periodic) **oscillations**



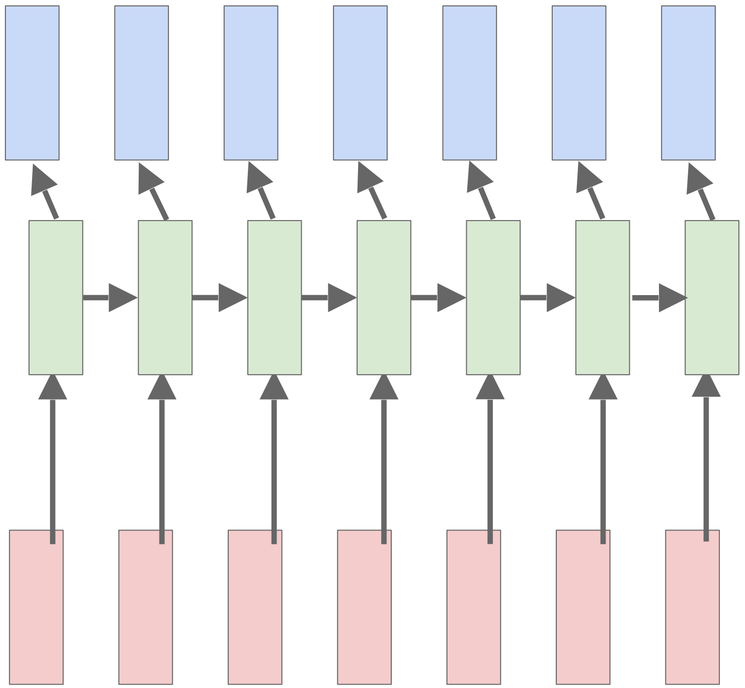


Bi-directional RNNs/LSTMs

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

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

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



Bi-directional RNNs/LSTMs

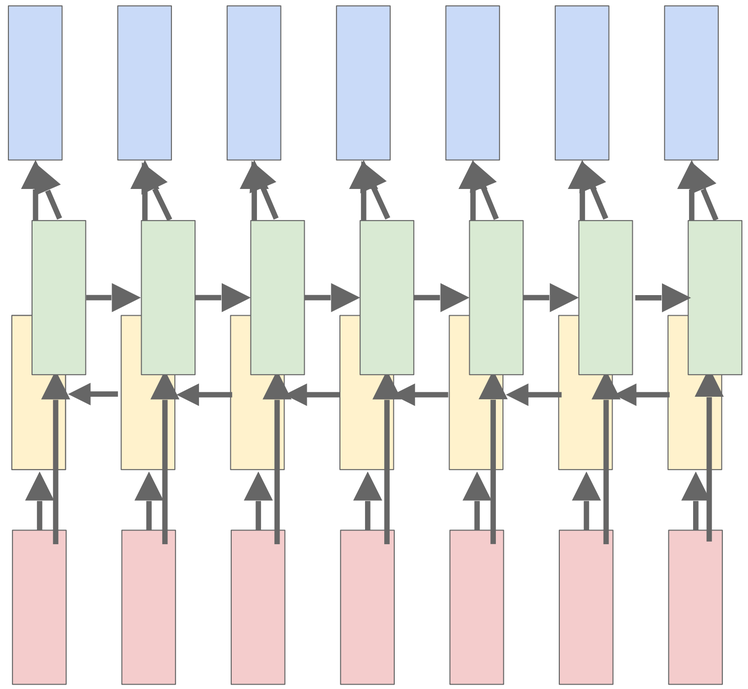
$$y_t = W_{hy}[\overrightarrow{h}_t, \overleftarrow{h}_t]^T + b_y$$

$$\overrightarrow{h}_t = f_{\overrightarrow{W}}(\overrightarrow{h}_{t-1}, x_t)$$

$$\overleftarrow{h}_t = f_{\overleftarrow{W}}(\overleftarrow{h}_{t+1}, x_t)$$

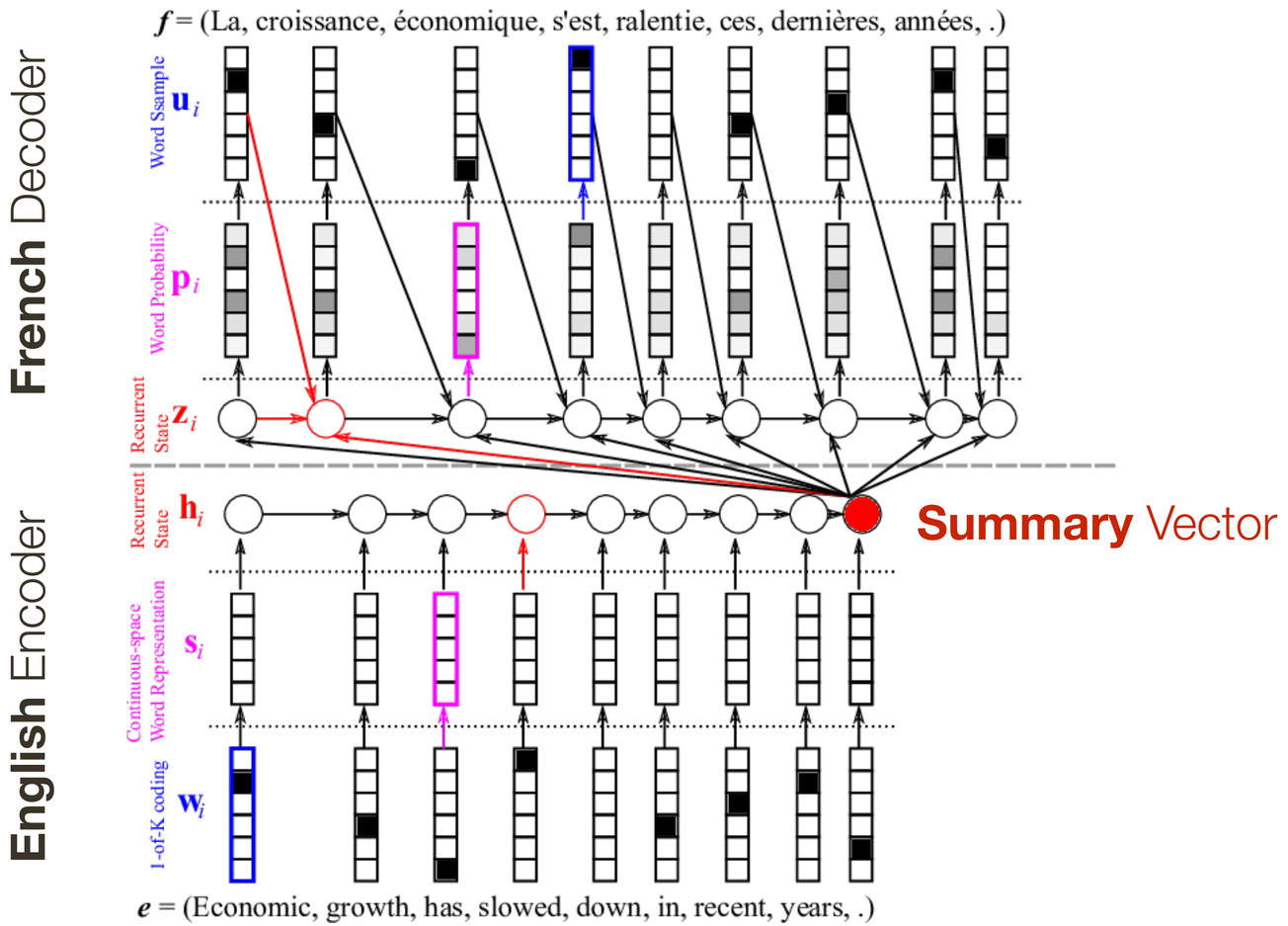
$$\overrightarrow{h}_t = \tanh(\overrightarrow{W}_{hh}\overrightarrow{h}_{t-1} + \overrightarrow{W}_{xh}x_t + \overrightarrow{b}_h)$$

$$\overleftarrow{h}_t = \tanh(\overleftarrow{W}_{hh}\overleftarrow{h}_{t+1} + \overleftarrow{W}_{xh}x_t + \overleftarrow{b}_h)$$



Attention Mechanisms and RNNs

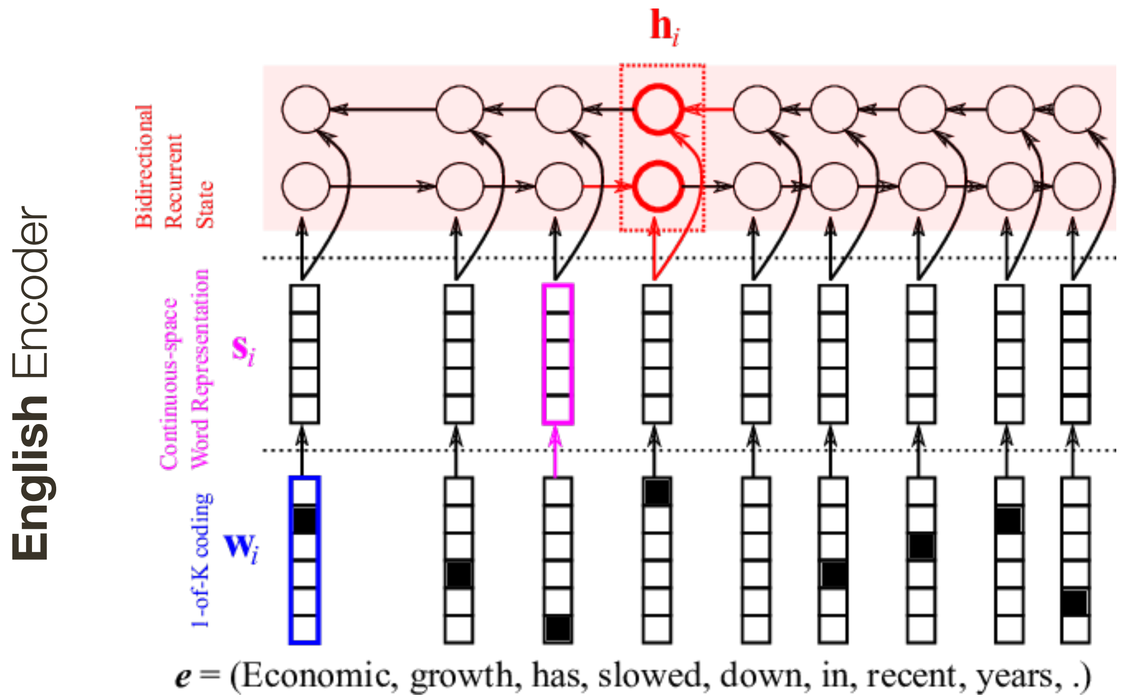
Consider a **translation task**: This is one of the first neural translation models



https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

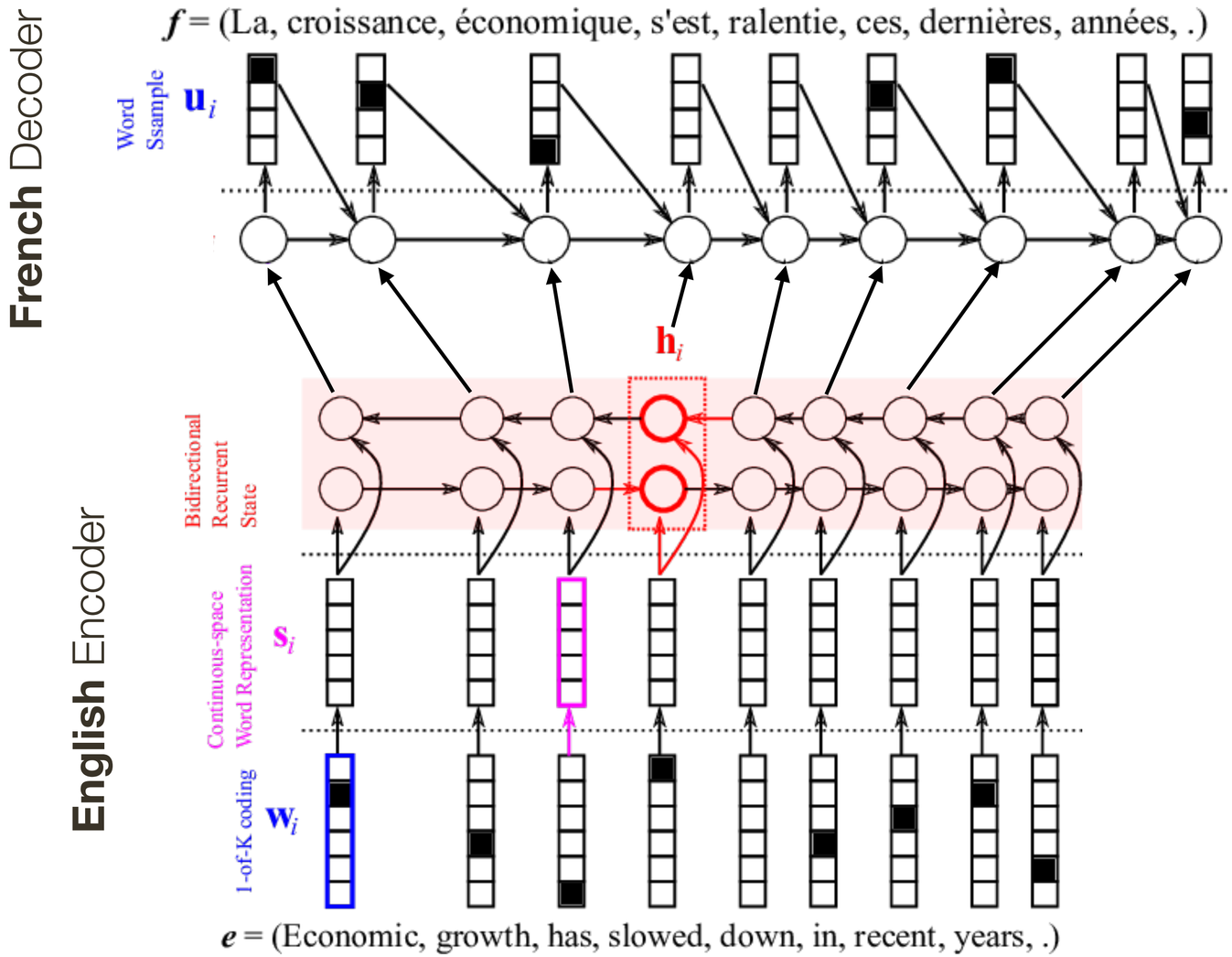
Consider a **translation task** with a bi-directional encoder of the source language



https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

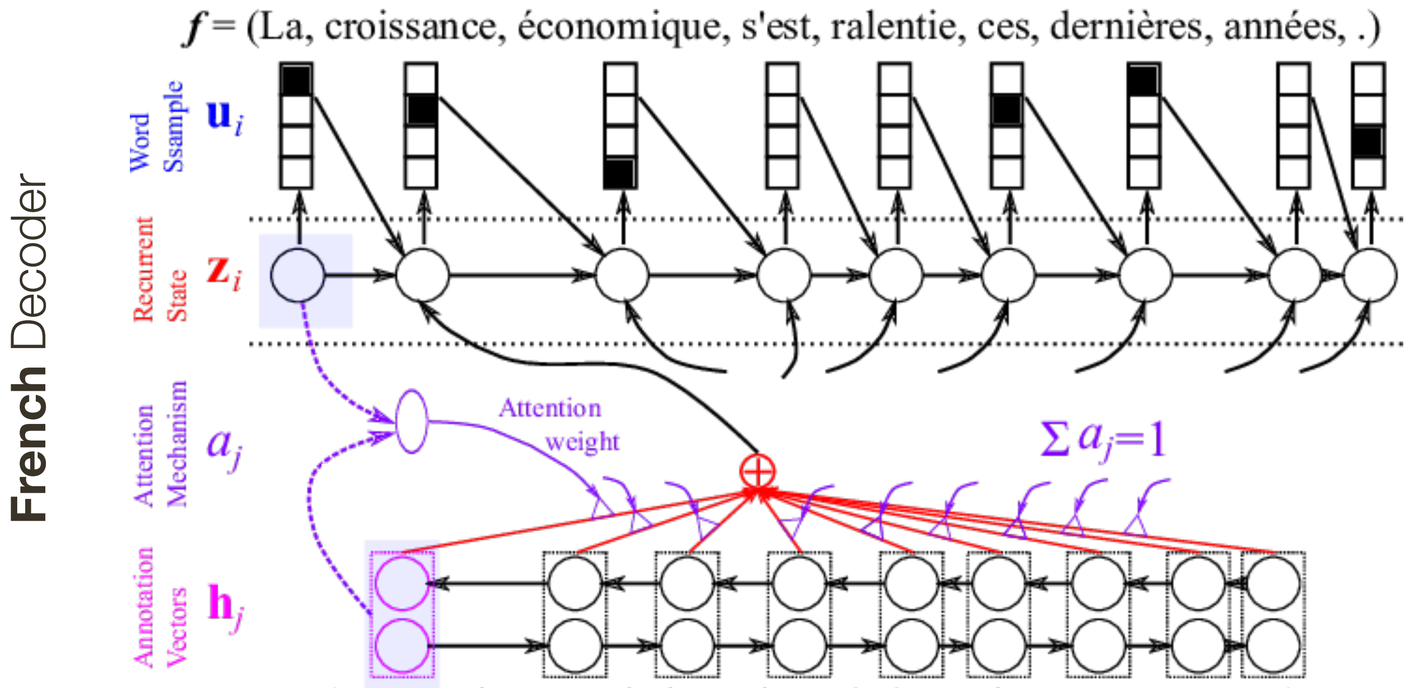
Consider a **translation task** with a bi-directional encoder of the source language



https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

Consider a **translation task** with a bi-directional encoder of the source language



[Cho et al., 2015]

Build a **small neural network** (one layer) with softmax output that takes

(1) everything decoded so far and (encoded by previous decoder state Zi)

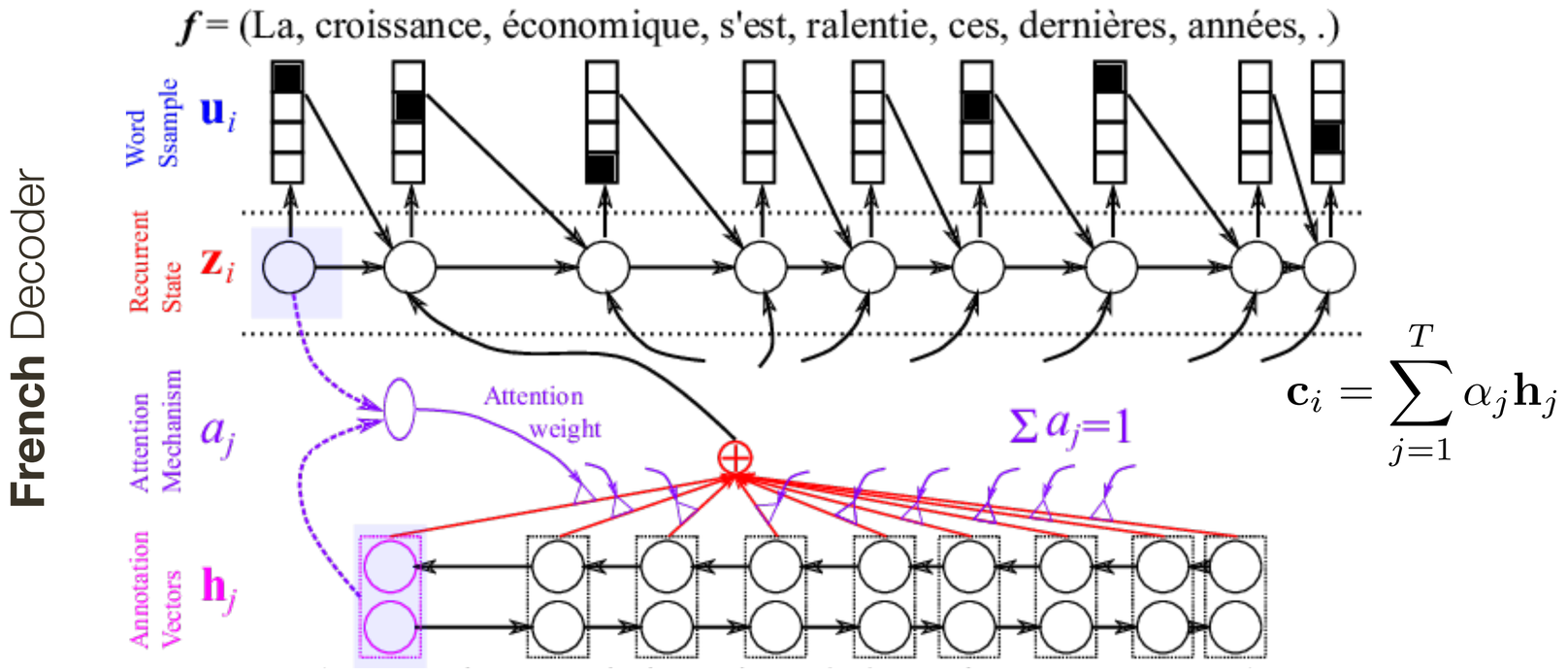
(2) encoding of the current word (encoded by the hidden state of encoder hj)

and predicts **relevance of every source word** towards next translation

https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

Consider a **translation task** with a bi-directional encoder of the source language



[Cho et al., 2015]

Build a **small neural network** (one layer) with softmax output that takes

(1) everything decoded so far and (encoded by previous decoder state Zi)

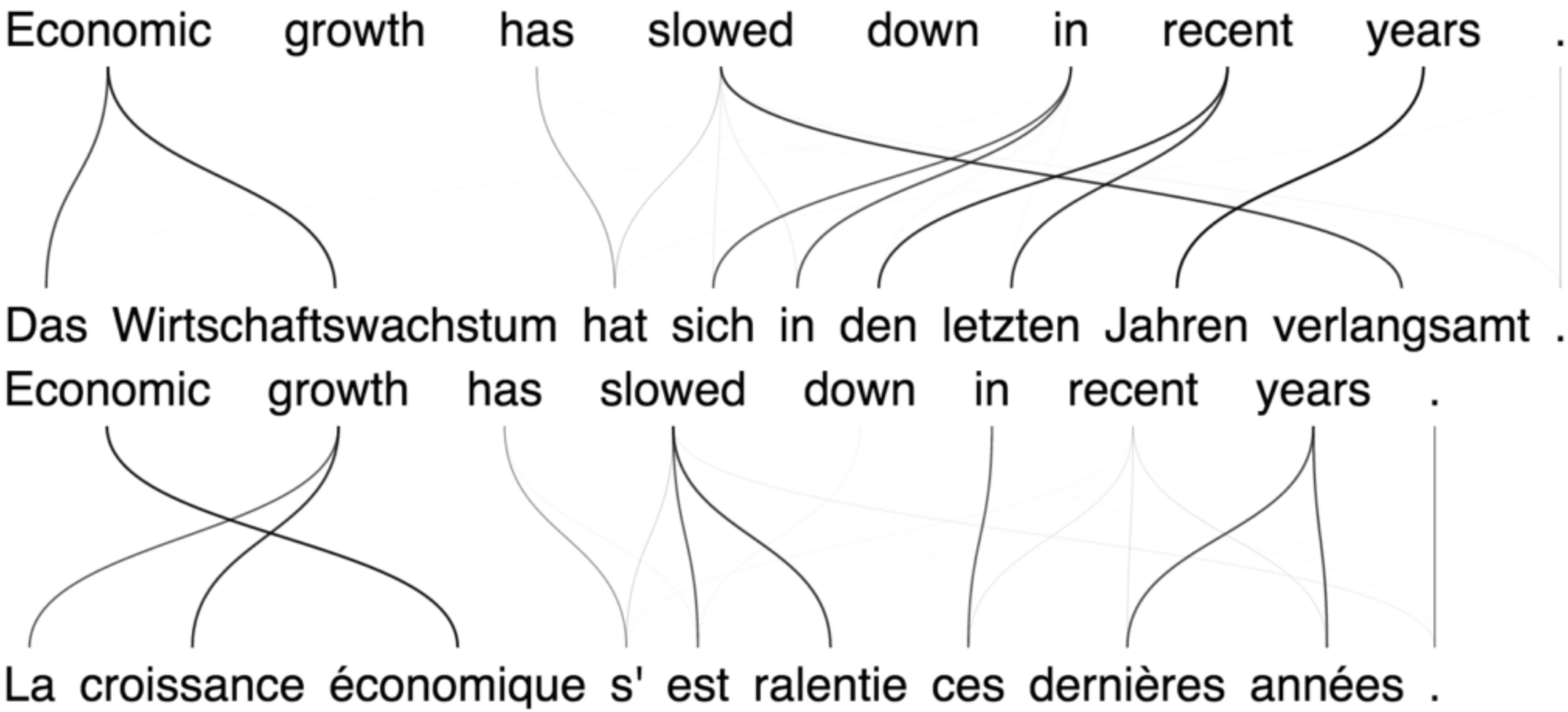
(2) encoding of the current word (encoded by the hidden state of encoder hj)

and predicts **relevance of every source word** towards next translation

https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Attention Mechanisms and RNNs

[Cho et al., 2015]



https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Regularization in RNNs

Standard dropout in recurrent layers does not work because it causes **loss of long term memory**!

* slide from Marco Pedersoli and Thomas Lucas

Regularization in RNNs

Standard dropout in recurrent layers does not work because it causes **loss of long term memory**!

— Dropout in input-to-hidden or hidden-to-output layers [Zaremba et al., 2014]

— Apply dropout at sequence level (same zeroed units for the entire sequence) [Gal, 2016]

— Dropout only at the cell update (for LSTM and GRU units) [Semeniuta et al., 2016]

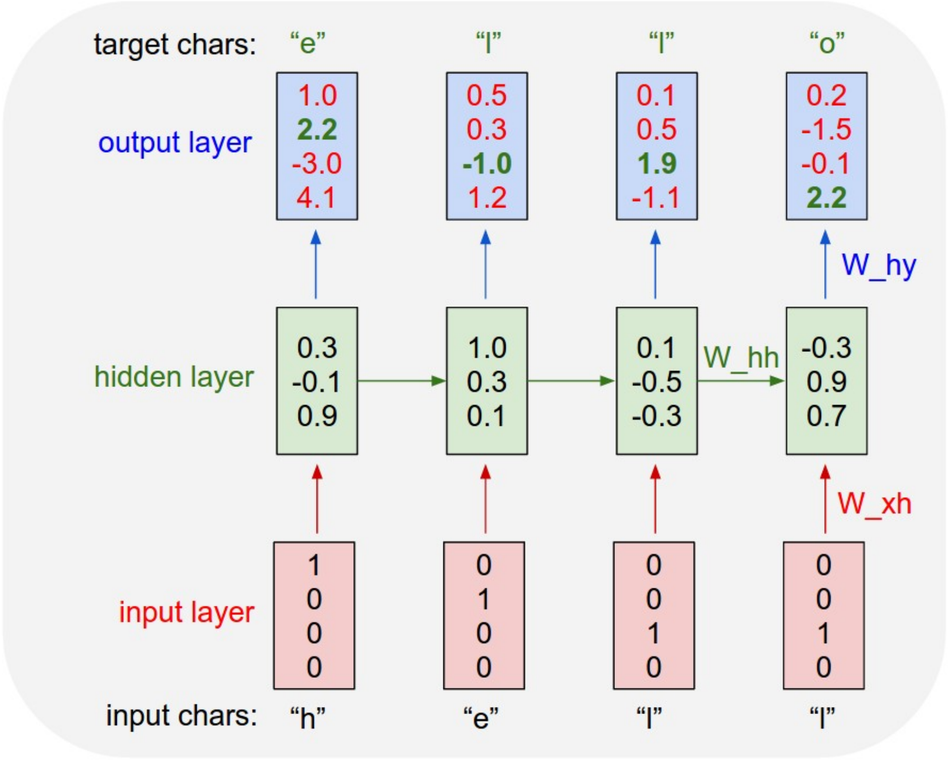
— Enforcing norm of the hidden state to be similar along time [Krueger & Memisevic, 2016]

— Zoneout some hidden units (copy their state to the next tilmestep) [Krueger et al., 2016]

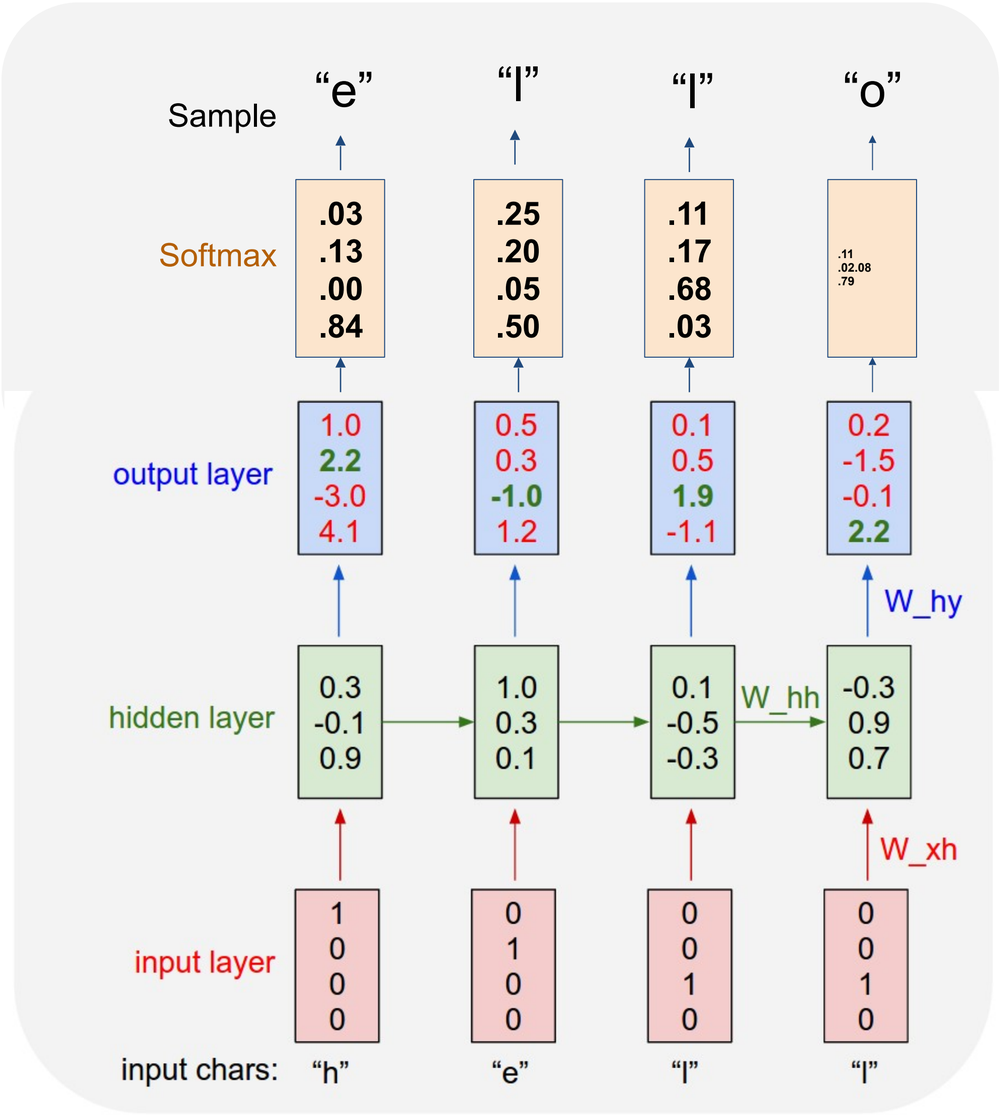
* slide from Marco Pedersoli and Thomas Lucas

Teacher Forcing

Training Objective: Predict the next word (cross entropy loss)

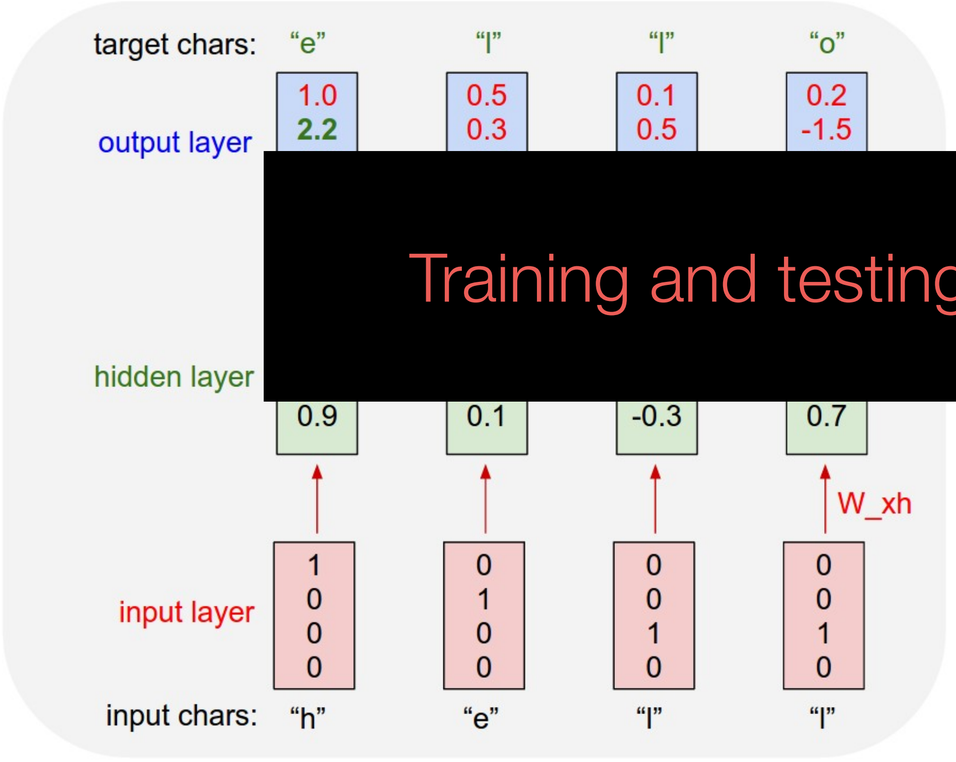


Testing: Sample the full sequence



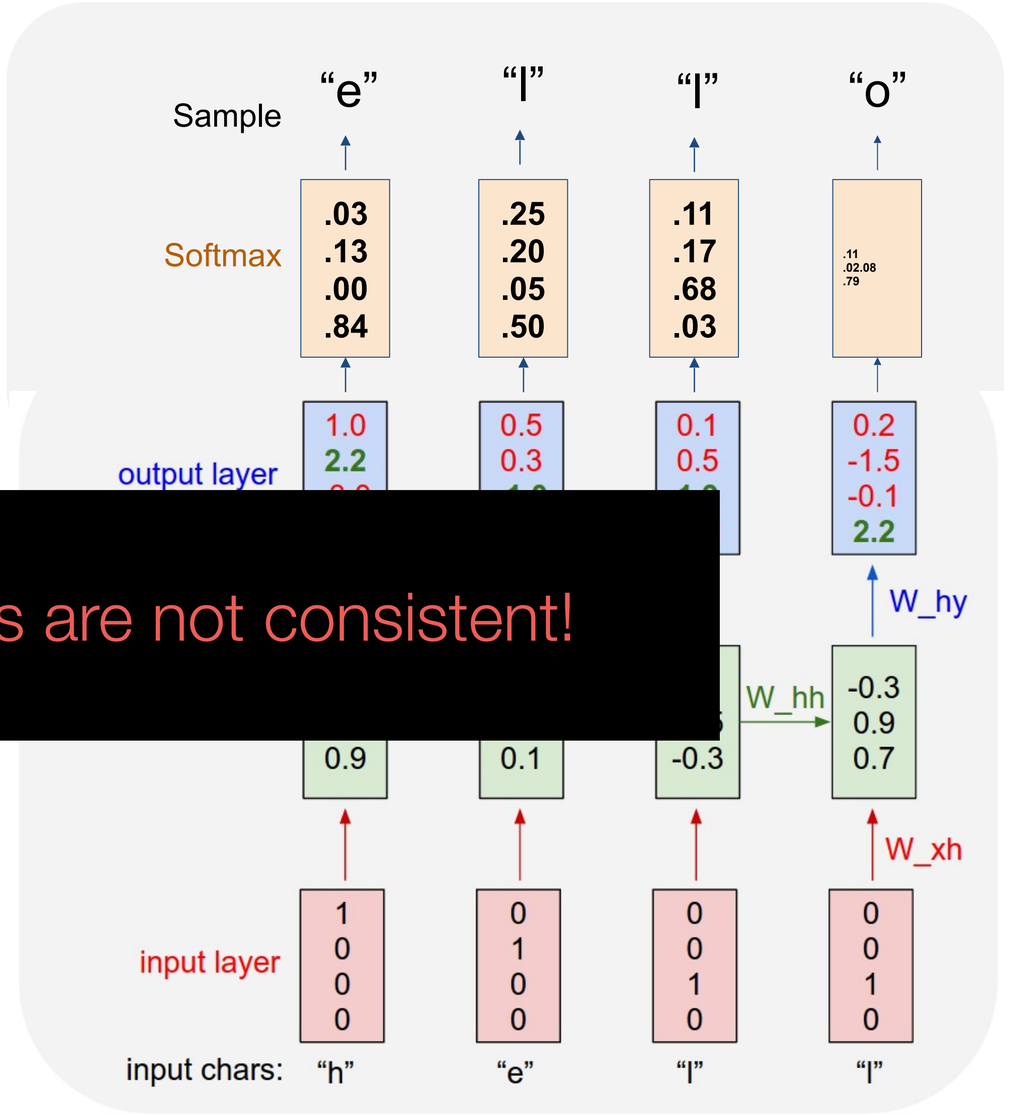
Teacher Forcing

Training Objective: Predict the next word (cross entropy loss)



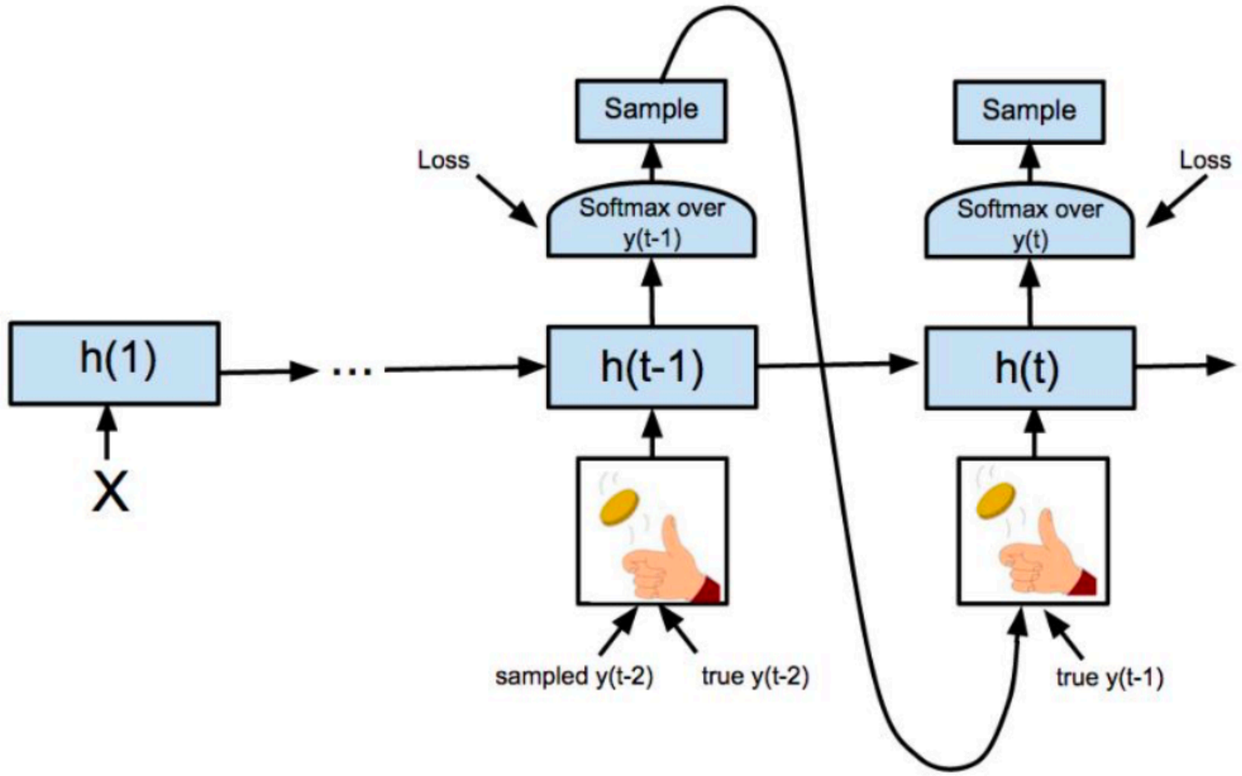
Training and testing objectives are not consistent!

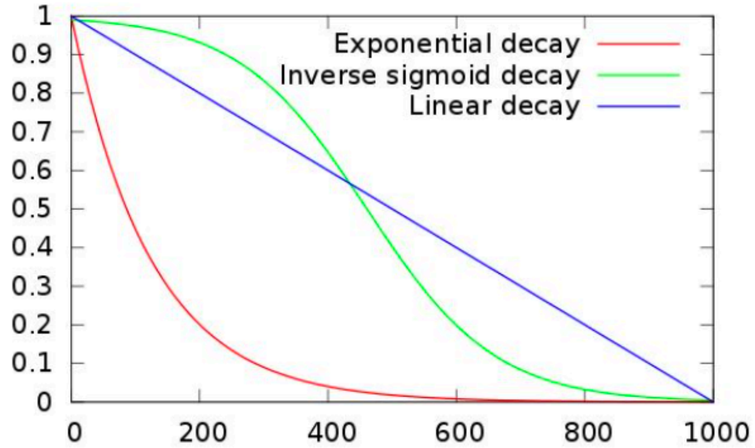
Testing: Sample the full sequence



Teacher Forcing

Slowly move from *Teacher Forcing* to *Sampling*





Probability of sampling from the ground truth

[Bengio et al., 2015]

* slide from Marco Pedersoli and Thomas Lucas

Teacher Forcing

Microsoft COCO developement set

Approach vs Metric	BLEU-4	METEOR	CIDER
Baseline	28.8	24.2	89.5
Baseline with Dropout	28.1	23.9	87.0
Always Sampling	11.2	15.7	49.7
Scheduled Sampling	**30.6**	**24.3**	**92.1**
Uniform Scheduled Sampling	29.2	24.2	90.9
Baseline ensemble of 10	30.7	25.1	95.7
Scheduled Sampling ensemble of 5	**32.3**	**25.4**	**98.7**

Baseline: Google NIC captioning model

Baseline **with Dropout**: Regularized RNN version

Always sampling: Use sampling from the beginning of training

Scheduled sampling: Sampling with inverse Sigmoid decay

Uniformed scheduled sampling: Scheduled sampling but uniformly

* slide from Marco Pedersoli and Thomas Lucas

Sequence Level Training

During training objective is different than at test time

— **Training:** generate next word given the previous

— **Test:** generate the entire sequence given an initial state

Optimize directly evaluation metric (e.g. BLUE score for sentence generation)

Set the problem as a Reinforcement Learning:

— RNN is an Agent

— Policy defined by the learned parameters

— Action is the selection of the next word based on the policy - Reward is the evaluation metric

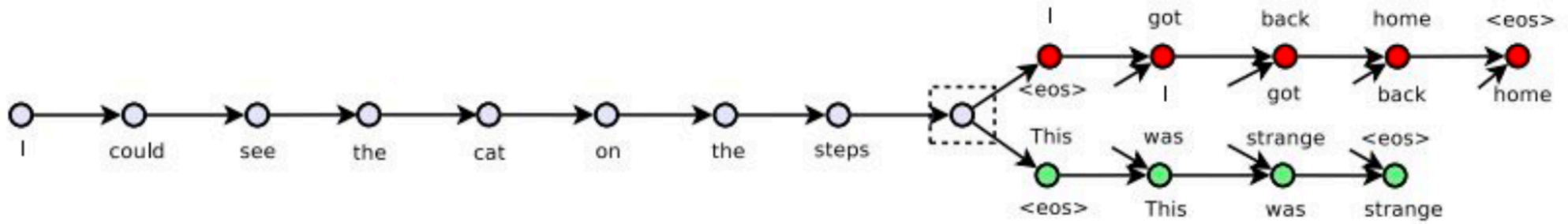
[Ranzato et al., 2016]

* slide from Marco Pedersoli and Thomas Lucas

Let us look at some actual practical uses of RNNs

Applications: Skip-thought Vectors

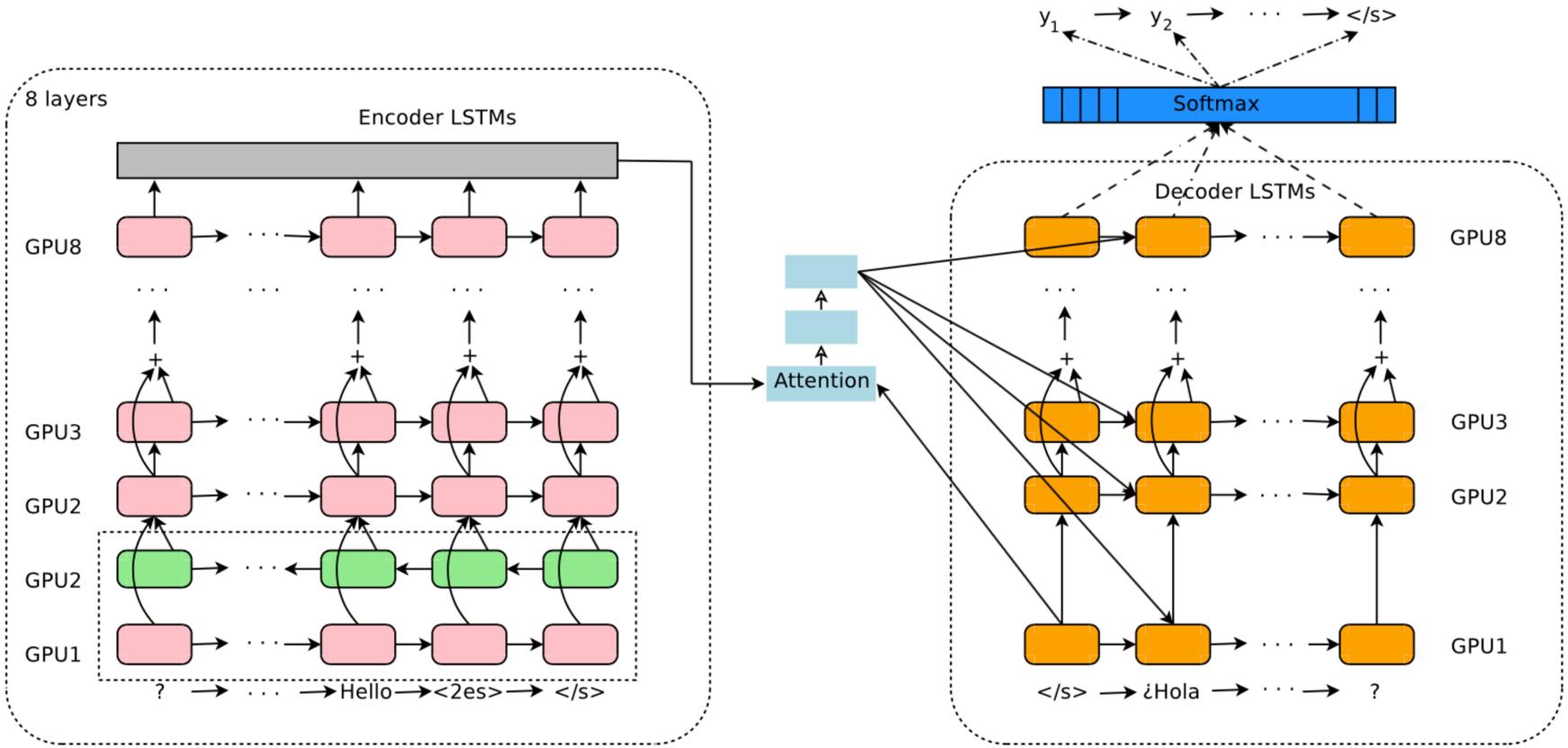
word2vec but for sentences, where each sentence is processed by an LSTM



[Kiros et al., 2015]

Applications: Google Language Translation

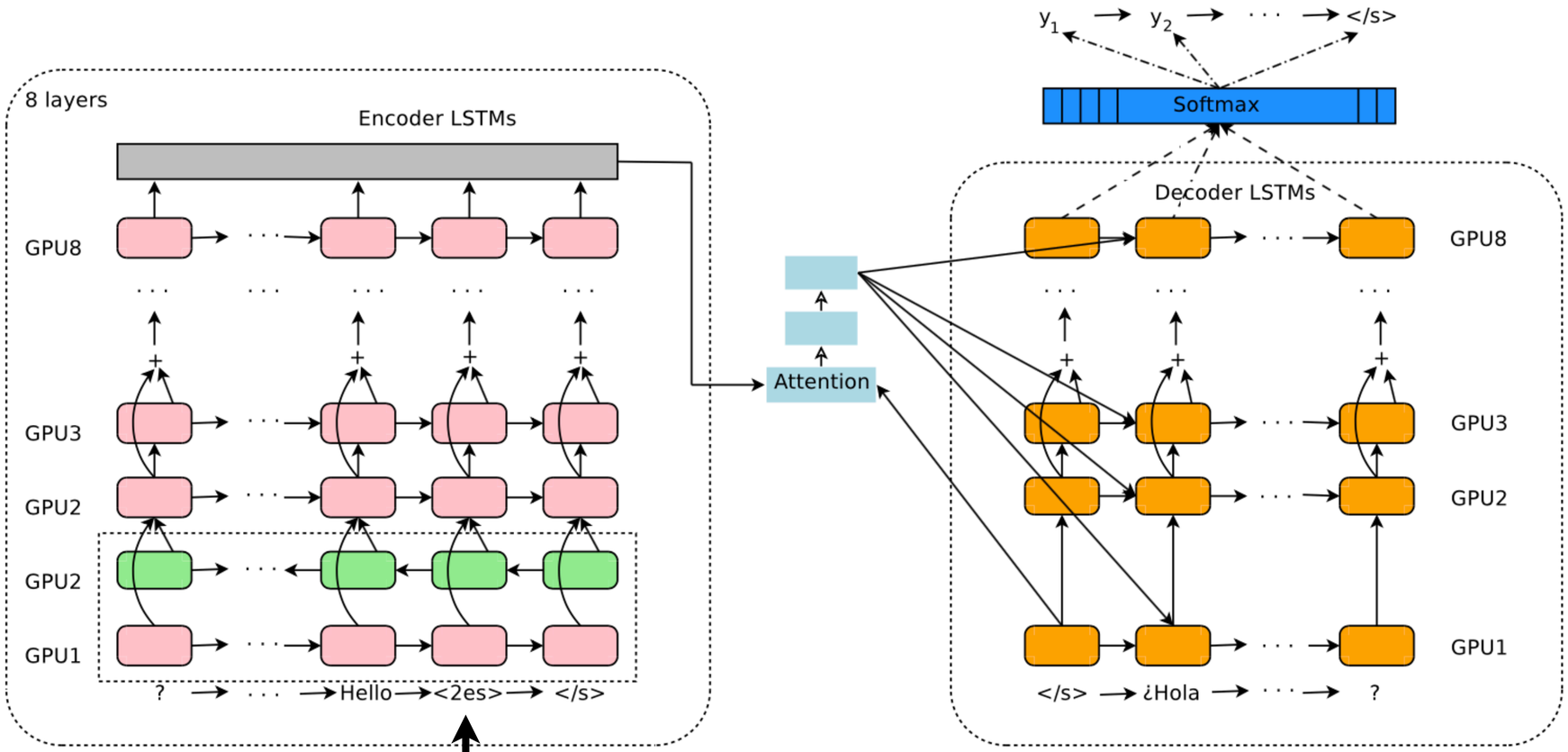
One model to translate from **any language** to any other language



[Johnson et al., 2017]

Applications: Google Language Translation

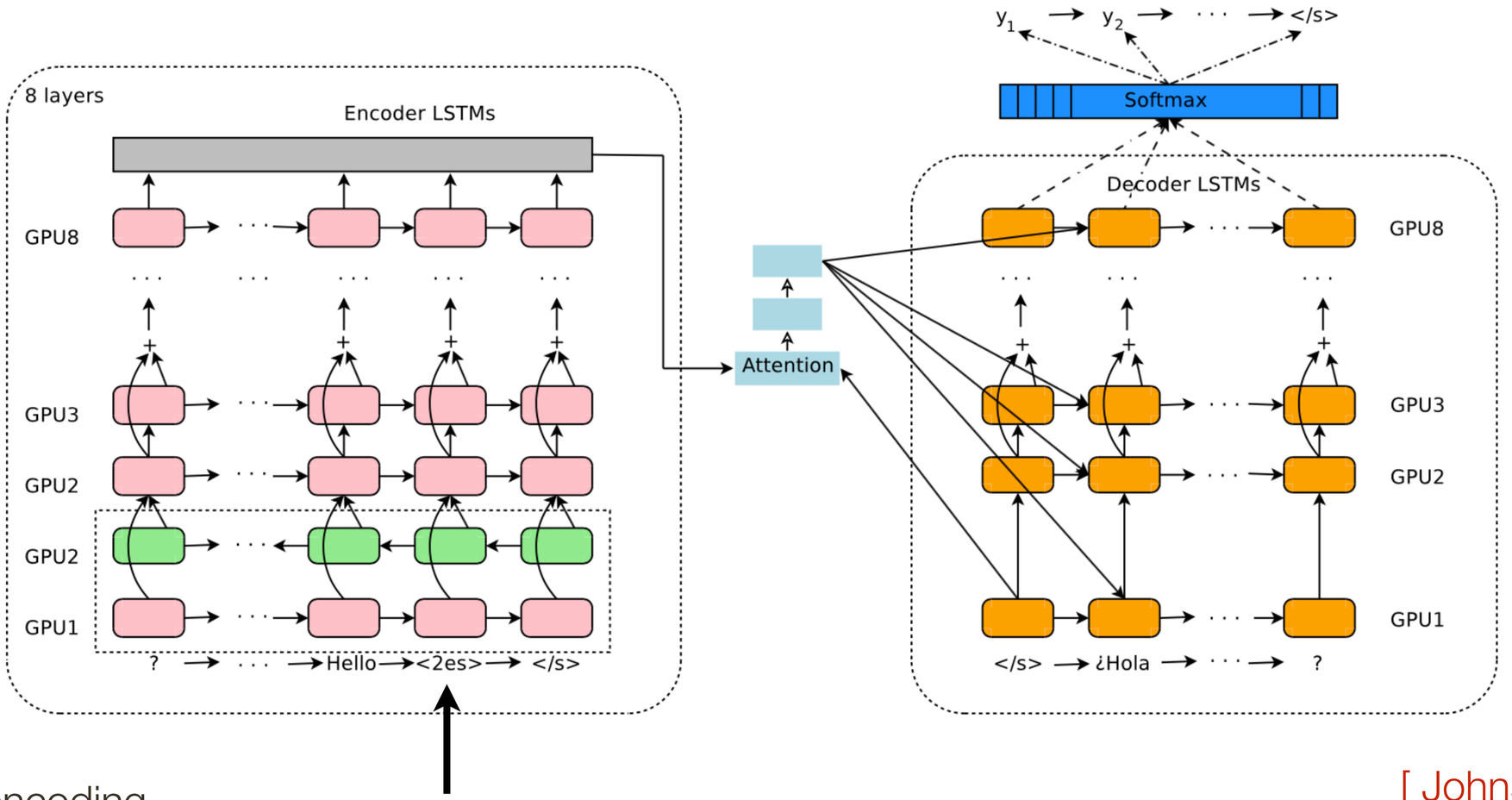
One model to translate from **any language** to any other language



[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language



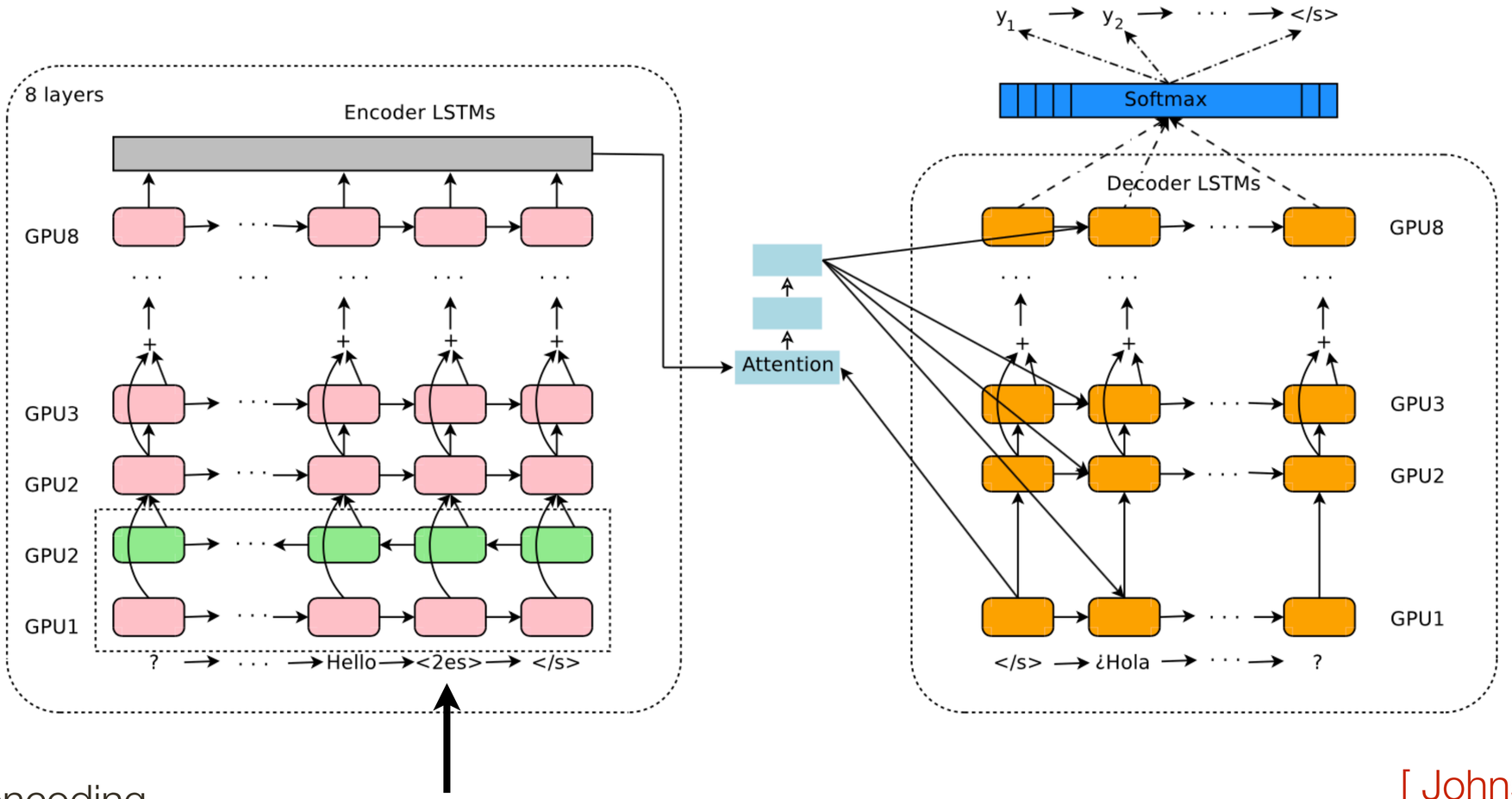
Flipped order encoding

Token designating **target** language

[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language



Flipped order encoding

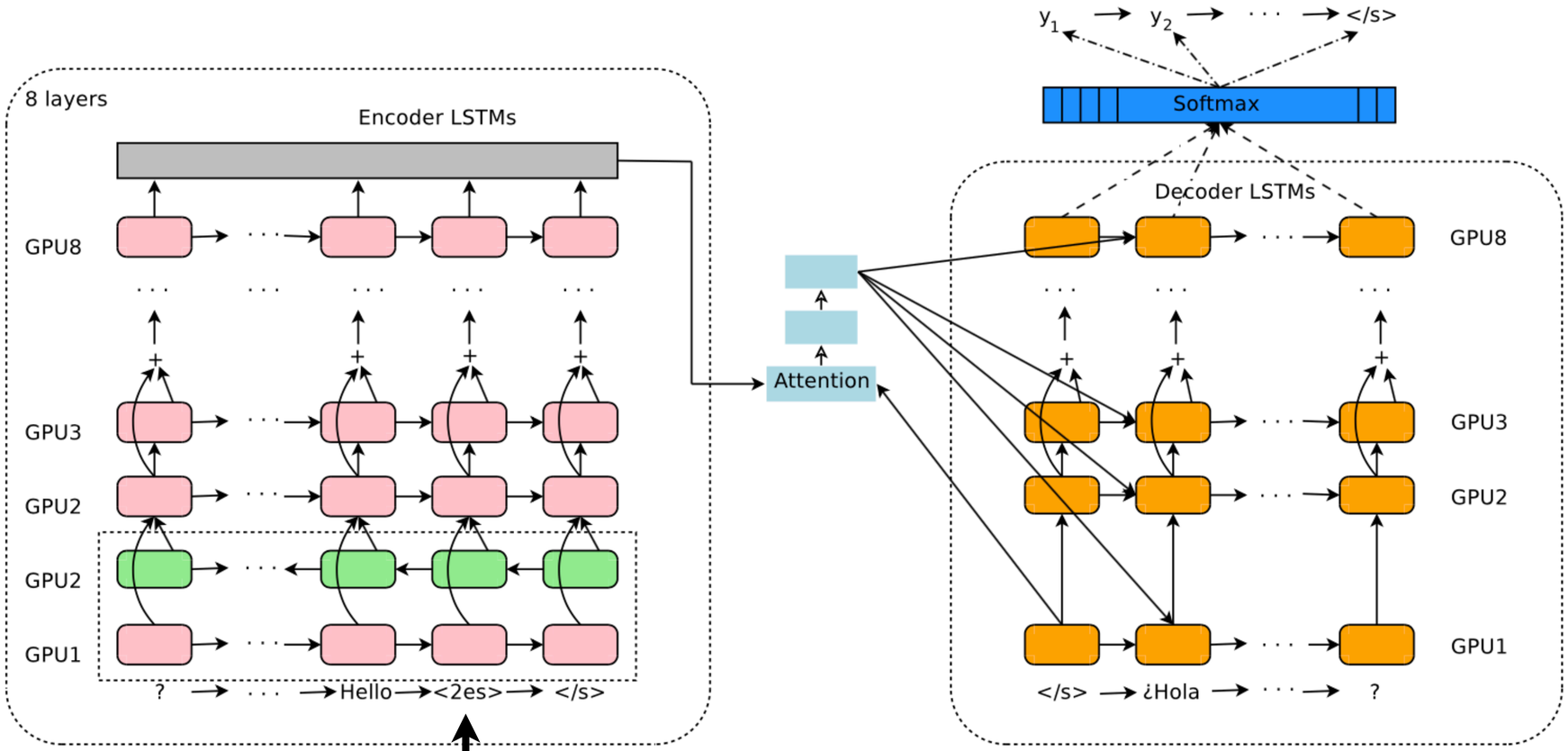
Why?

Token designating **target** language

[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language



Flipped order encoding

Token designating **target** language

[Johnson et al., 2017]

8! layer LSTM decoder and encoder

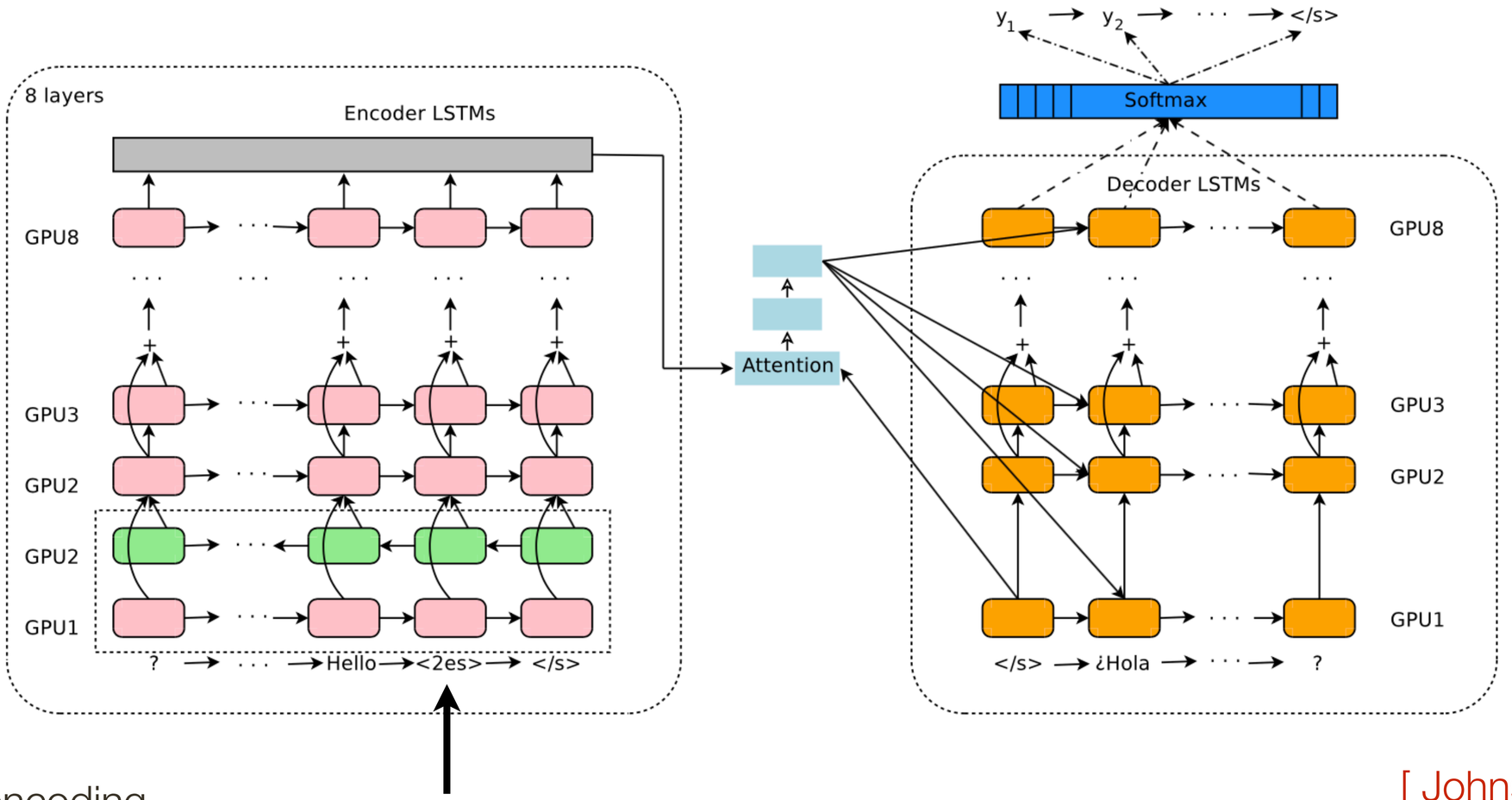
Applications: Google Language Translation

One model to translate from **any language** to any other language

Residual at other layers (ResNet style)

Bi-directional at lower layers

Flipped order encoding



Token designating **target** language

[Johnson et al., 2017]

8! layer LSTM decoder and encoder

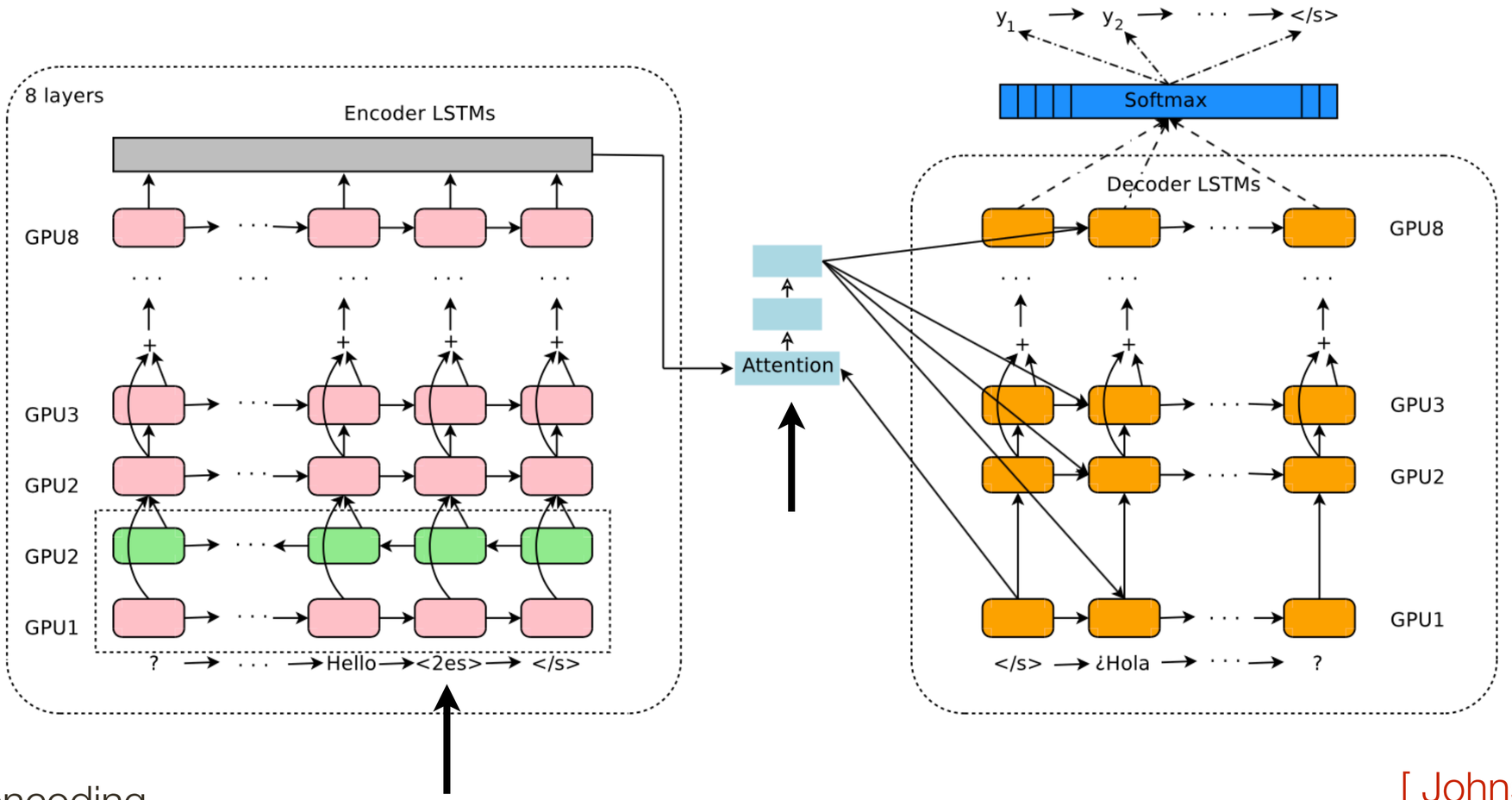
Applications: Google Language Translation

One model to translate from **any language** to any other language

Residual at other layers (ResNet style)

Bi-directional at lower layers

Flipped order encoding



Token designating **target** language

[Johnson et al., 2017]

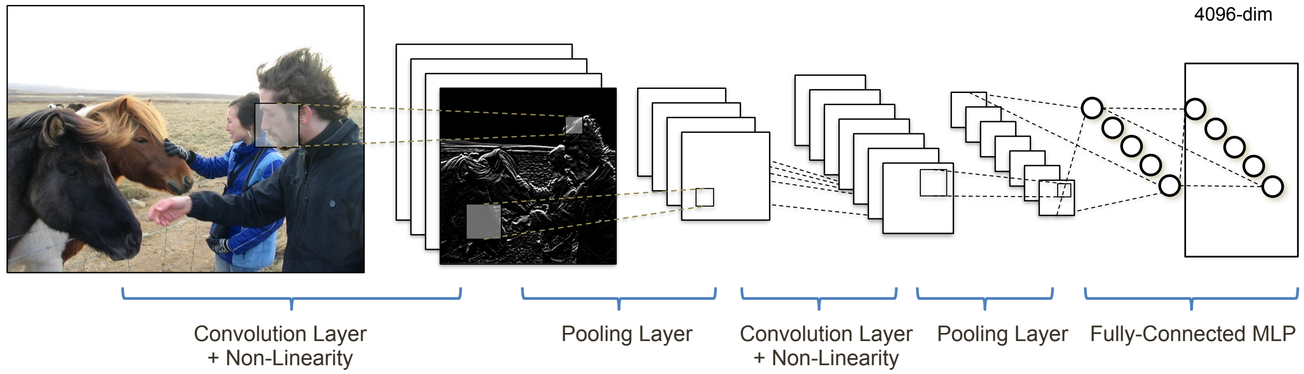
8! layer LSTM decoder and encoder

Applications: Neural Image Captioning

* slide from Dhruv Batra

Applications: Neural Image Captioning

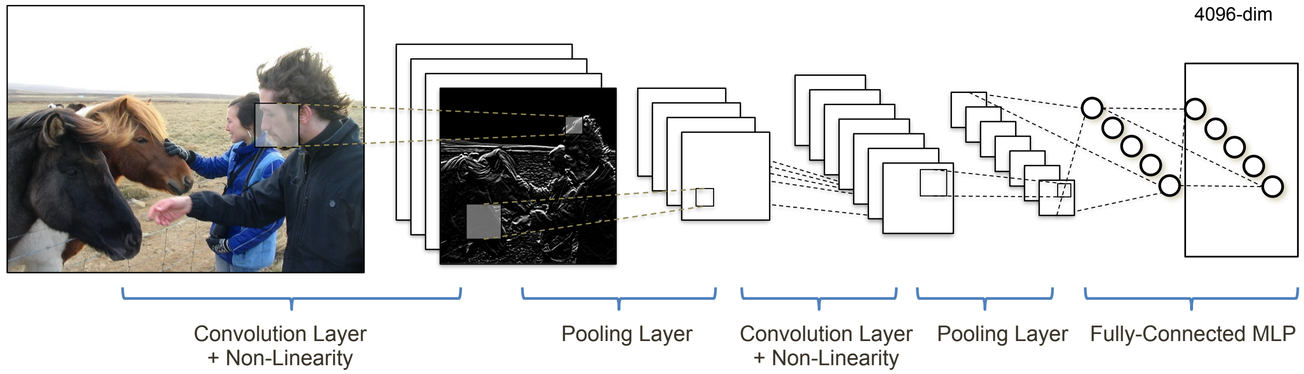
Image Embedding (VGGNet)



* slide from Dhruv Batra

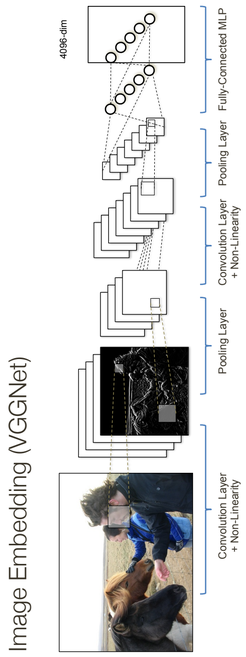
Applications: Neural Image Captioning

Image Embedding (VGGNet)



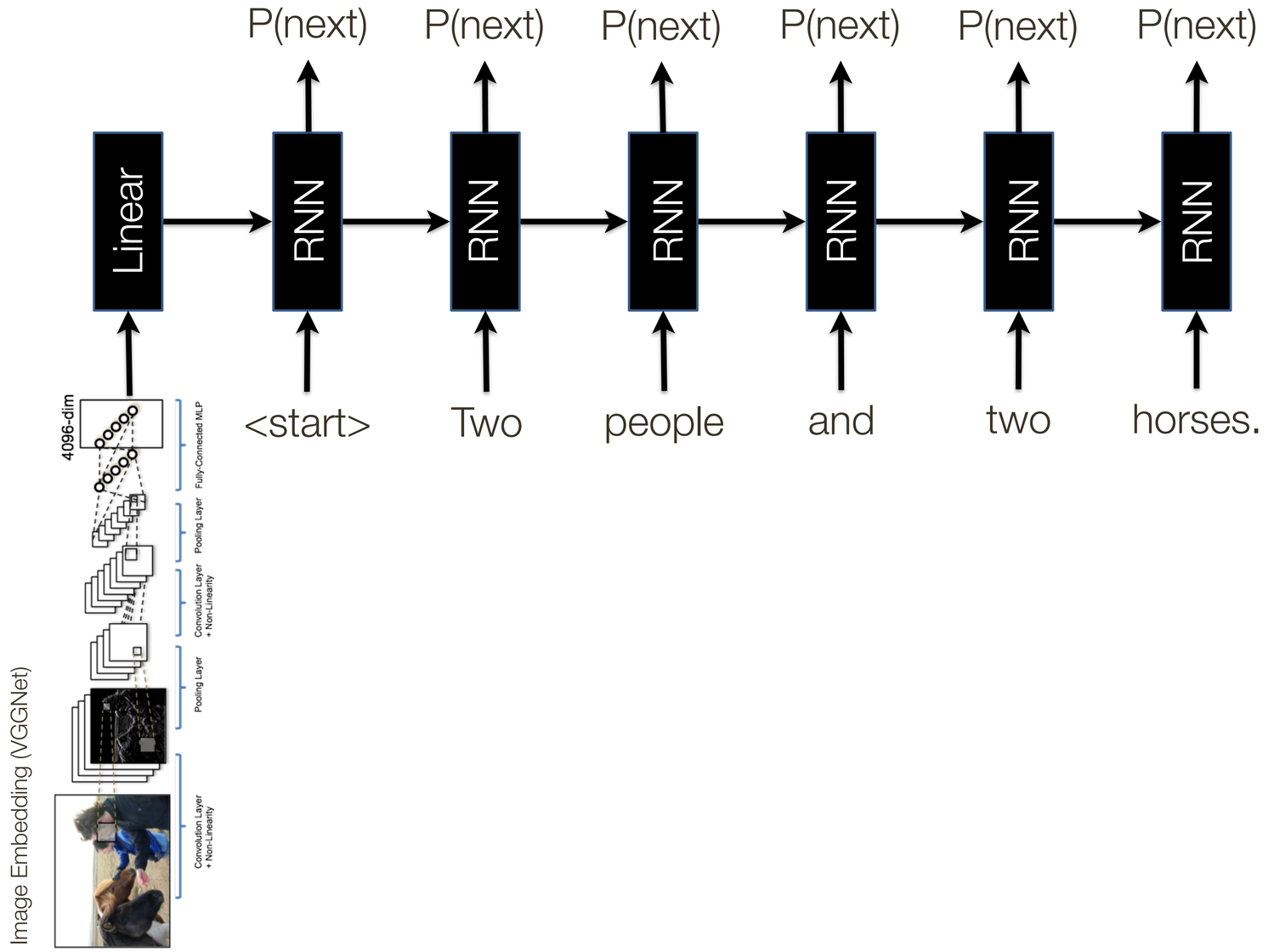
* slide from Dhruv Batra

Applications: Neural Image Captioning



* slide from Dhruv Batra

Applications: Neural Image Captioning



* slide from Dhruv Batra

Applications: Neural Image Captioning

Good 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

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

Applications: Neural Image Captioning

Failure cases

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A person holding a computer mouse on a desk

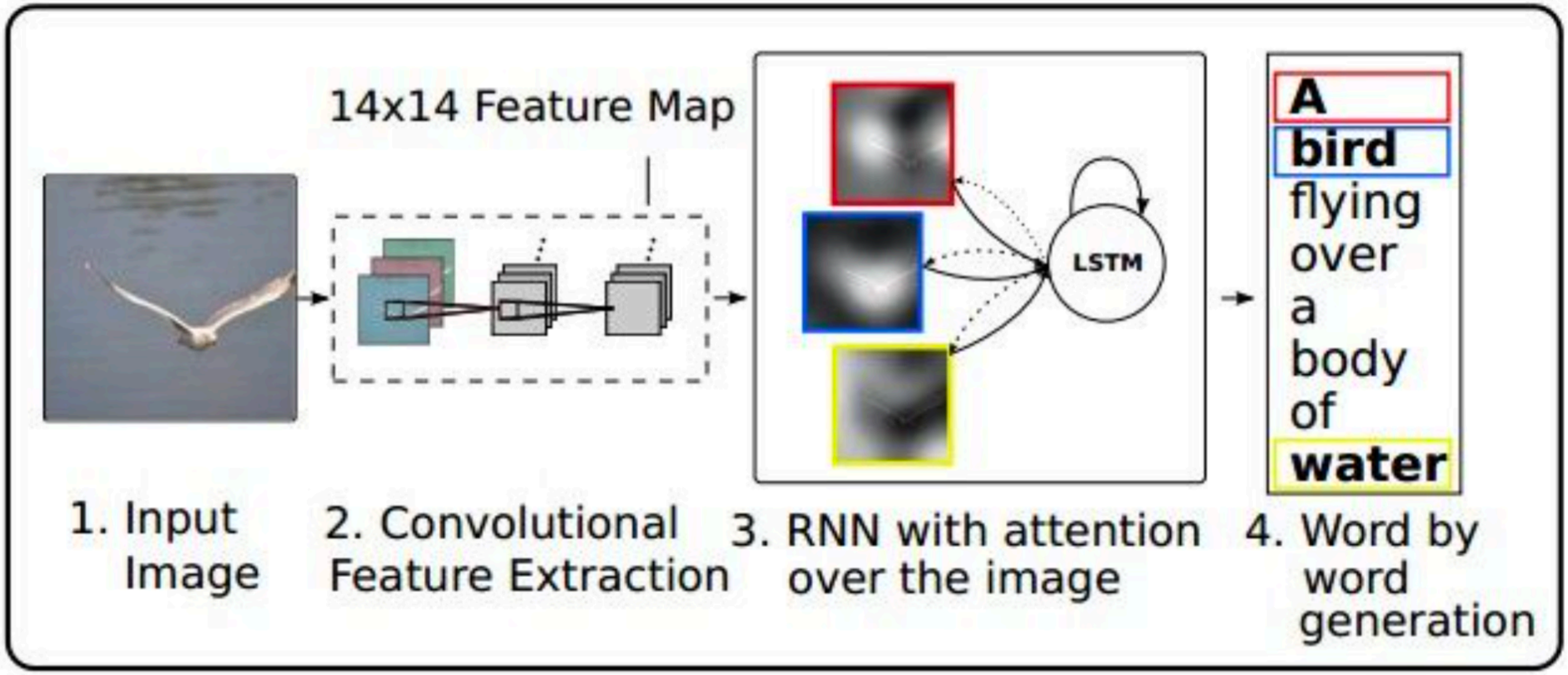
A man in a baseball uniform throwing a ball

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

Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

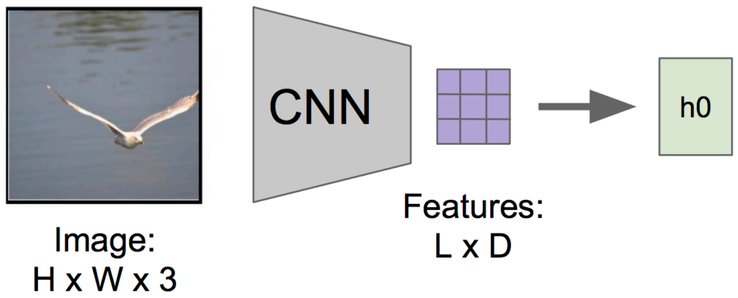
RNN focuses its attention at a different spatial location when generating each word



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

Applications: Image Captioning with Attention

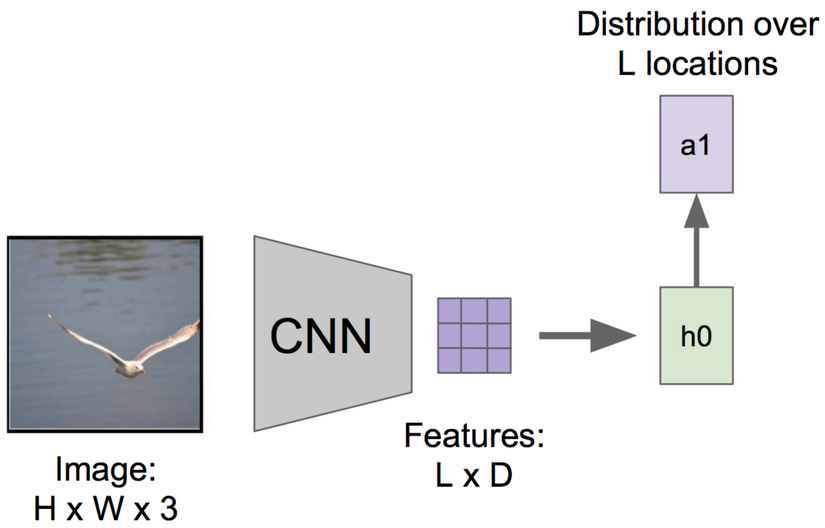
[Xu et al., ICML 2015]



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

Applications: Image Captioning with Attention

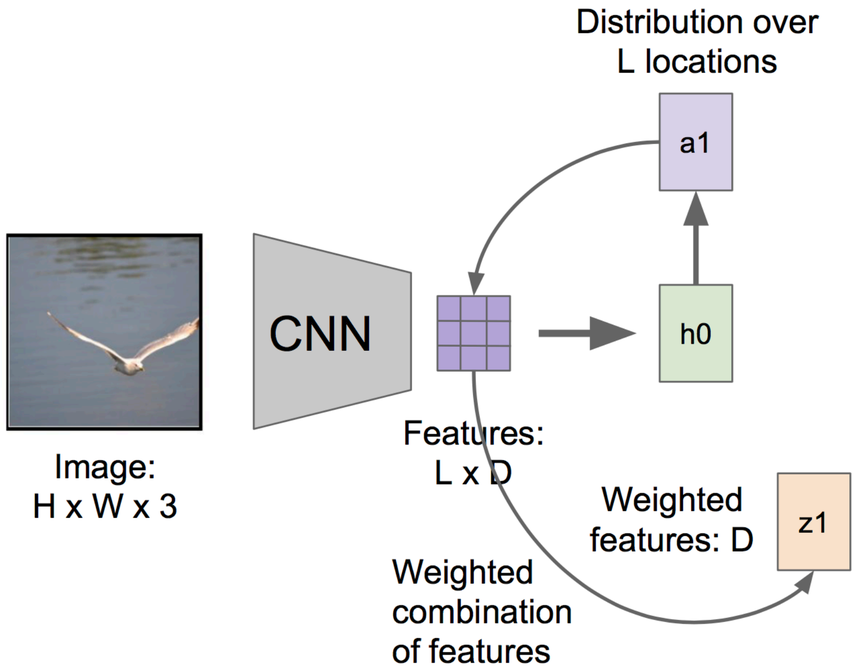
[Xu et al., ICML 2015]



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

Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

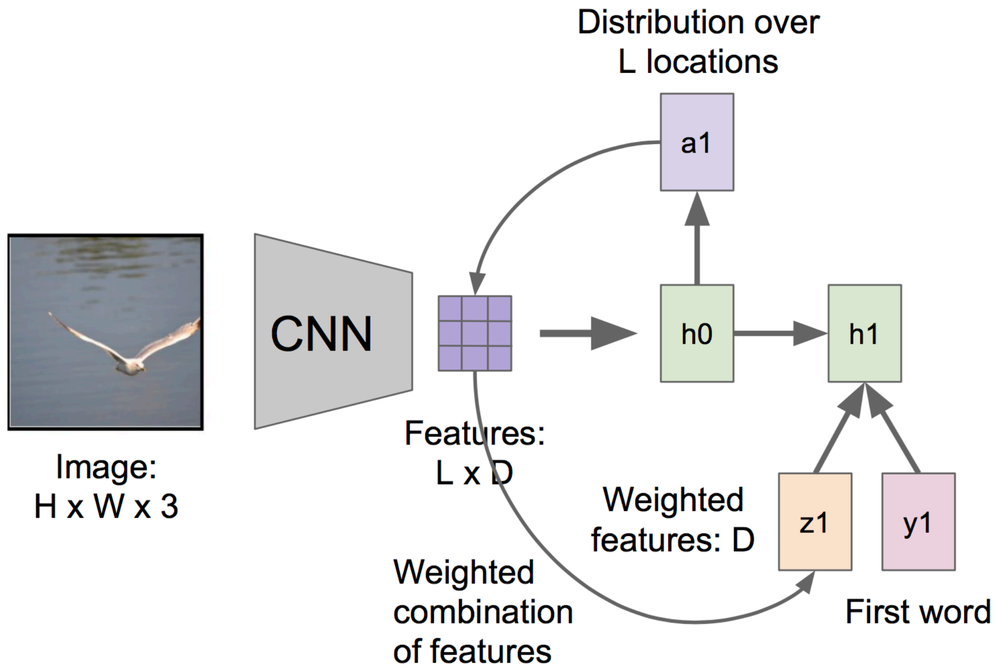


$$z = \sum_{i=1}^{L} p_i v_i$$

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

Applications: Image Captioning with Attention

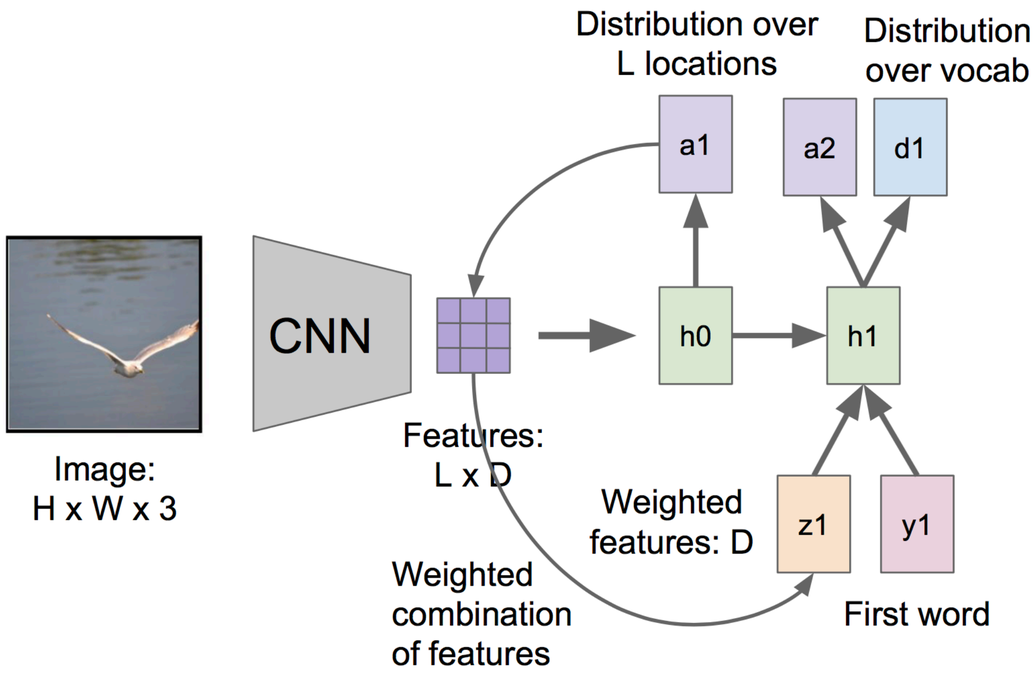
[Xu et al., ICML 2015]



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

Applications: Image Captioning with Attention

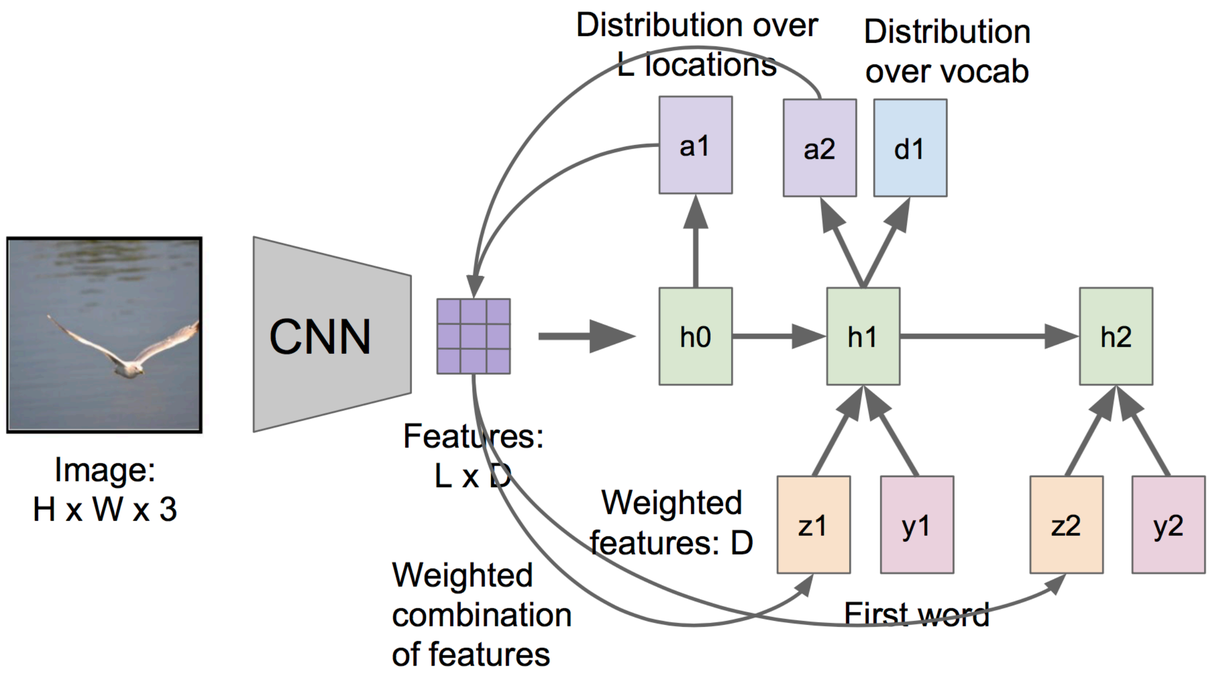
[Xu et al., ICML 2015]



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

Applications: Image Captioning with Attention

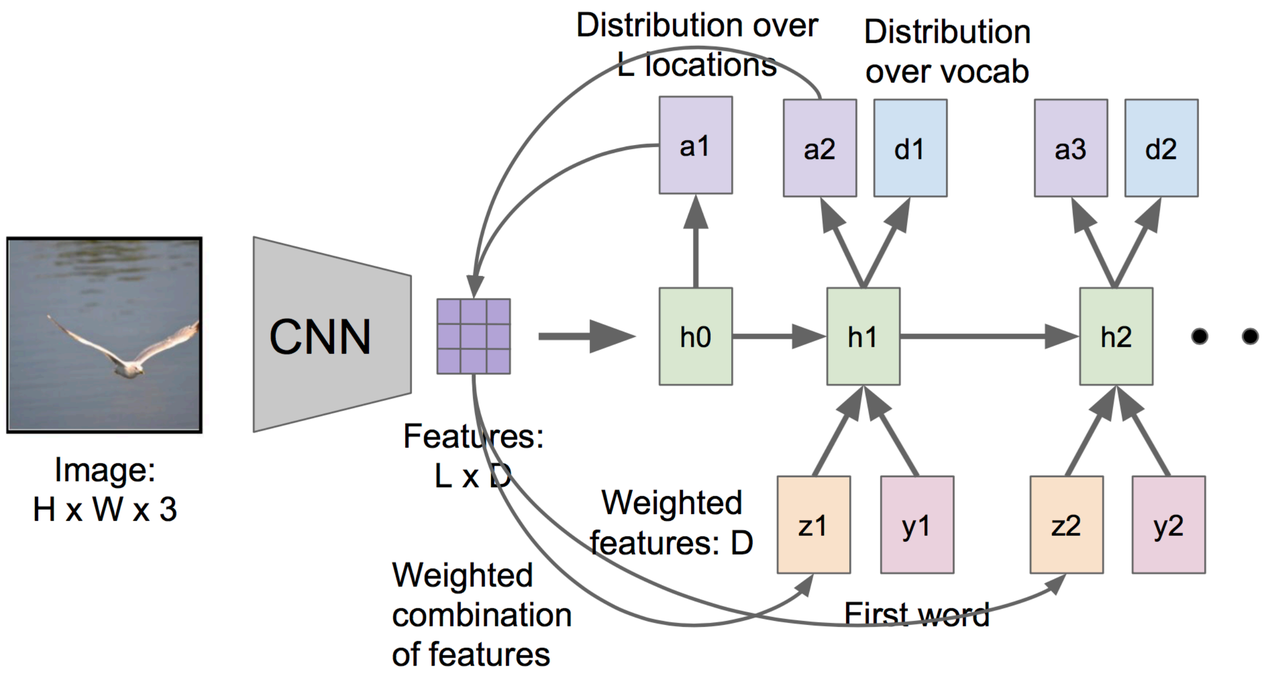
[Xu et al., ICML 2015]



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

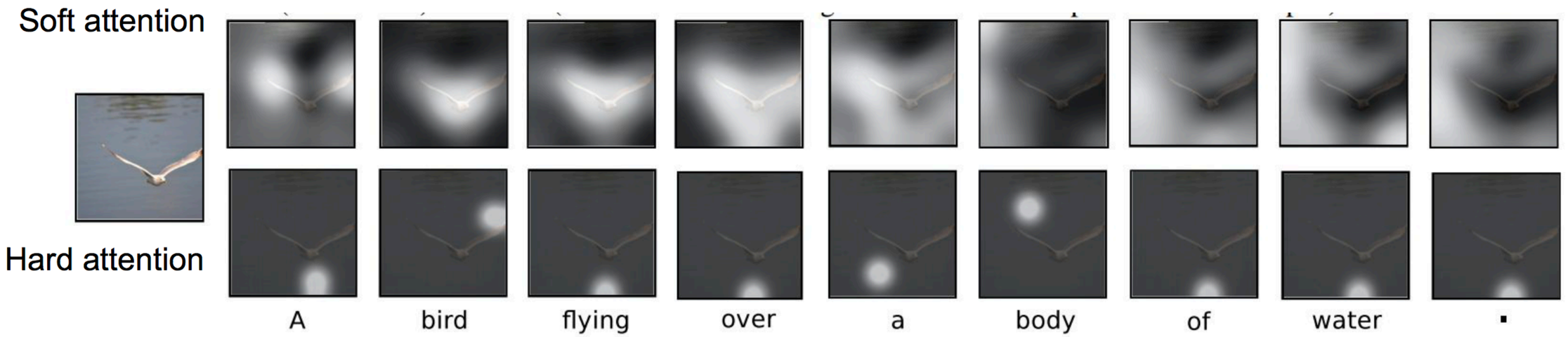
Applications: Image Captioning with Attention

[Xu et al., ICML 2015]



Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

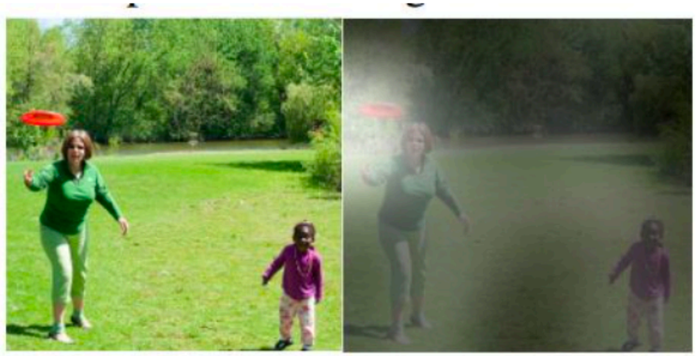


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

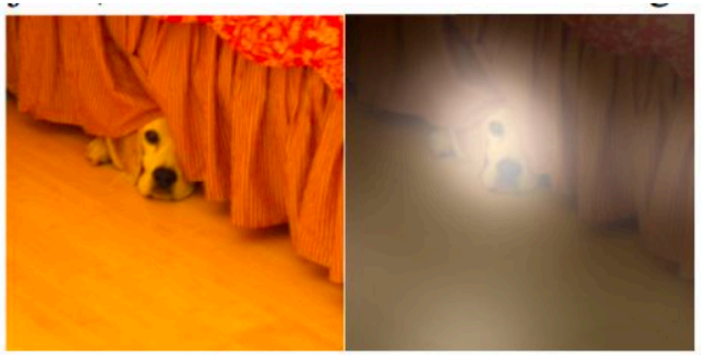
Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

Good results



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.

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

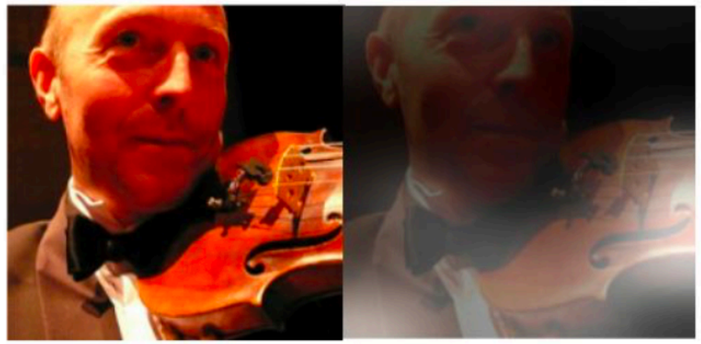
Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

Failure results

A large white bird standing in a forest.

A woman holding a clock in her hand.



A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

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

Applications: Typical Visual Question Answering (VQA)

Image

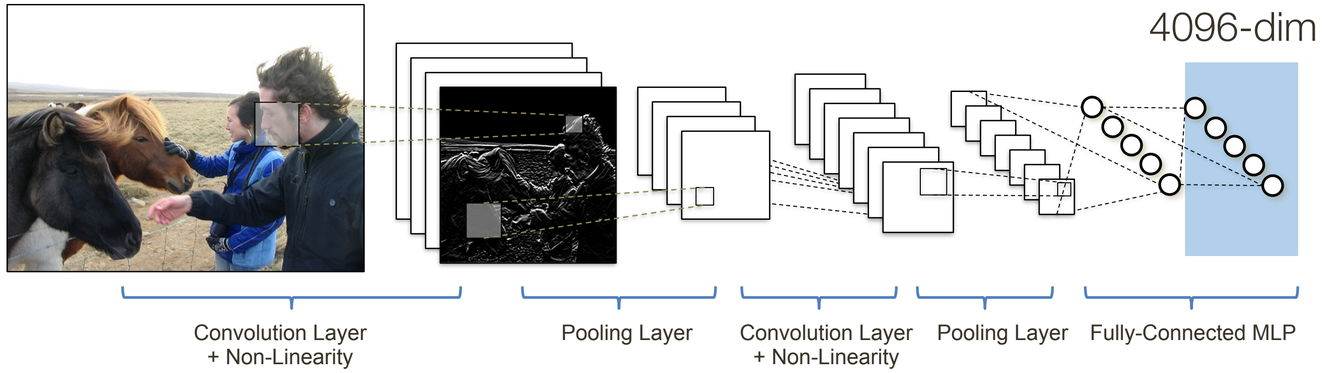
Question

“How many horses are in this image?”

* slide from Dhruv Batra

Applications: Typical Visual Question Answering (VQA)

Image Embedding (VGGNet)



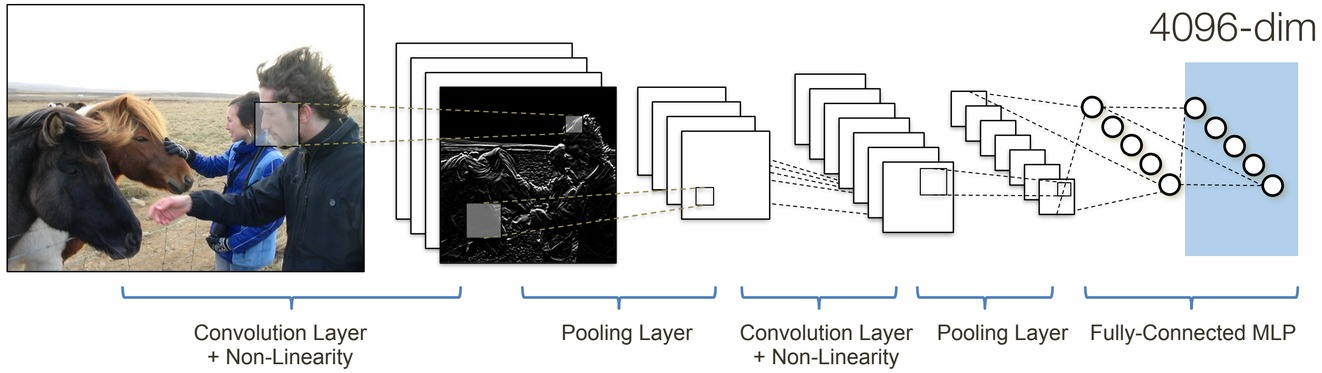
Question

“How many horses are in this image?”

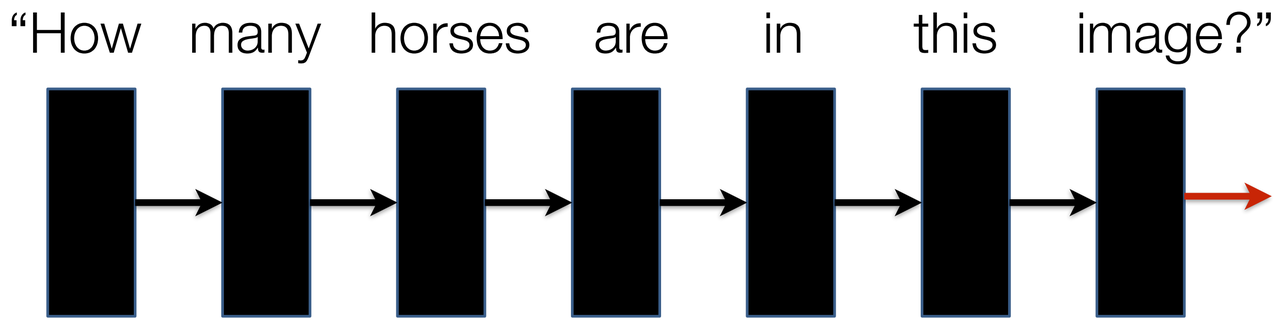
* slide from Dhruv Batra

Applications: Typical Visual Question Answering (VQA)

Image Embedding (VGGNet)



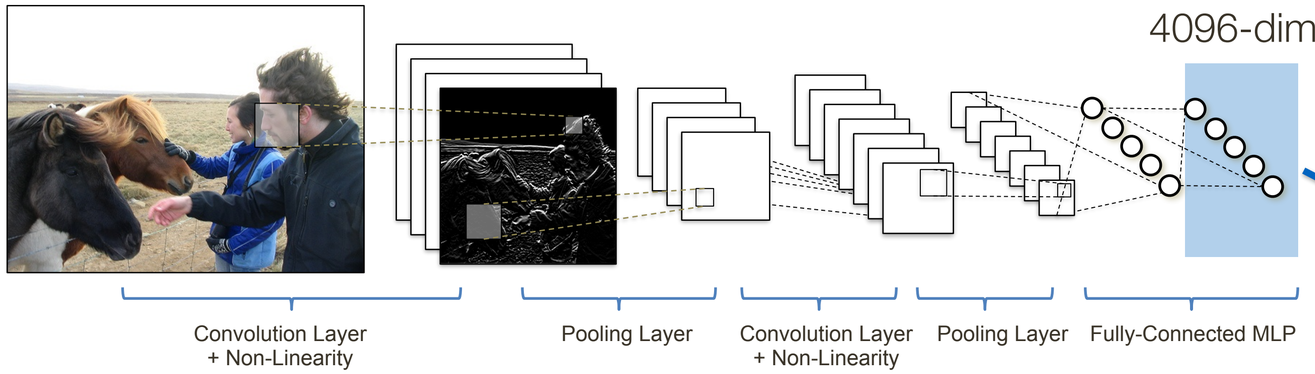
Question Embedding (LSTM)



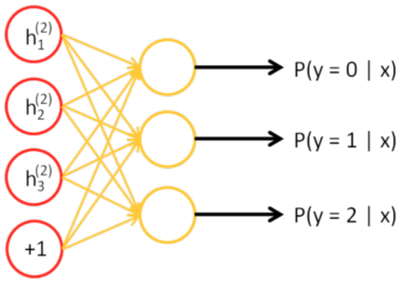
* slide from Dhruv Batra

Applications: Typical Visual Question Answering (VQA)

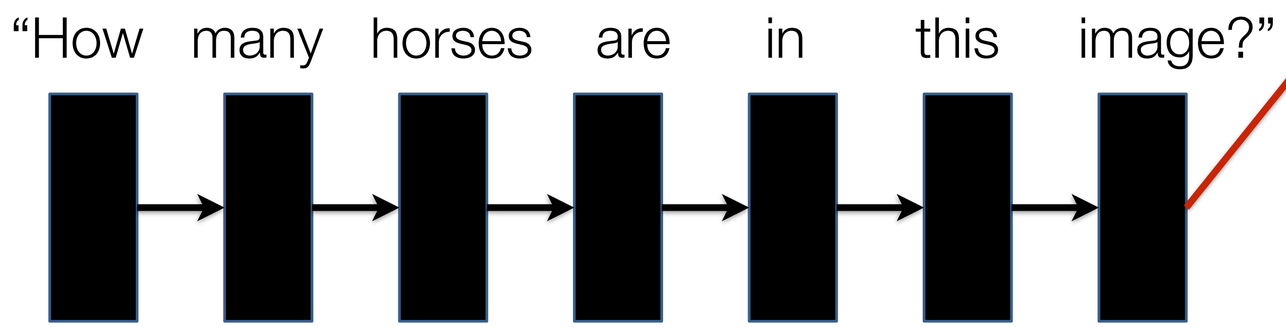
Image Embedding (VGGNet)



Neural Network
Softmax
over **top K answers**

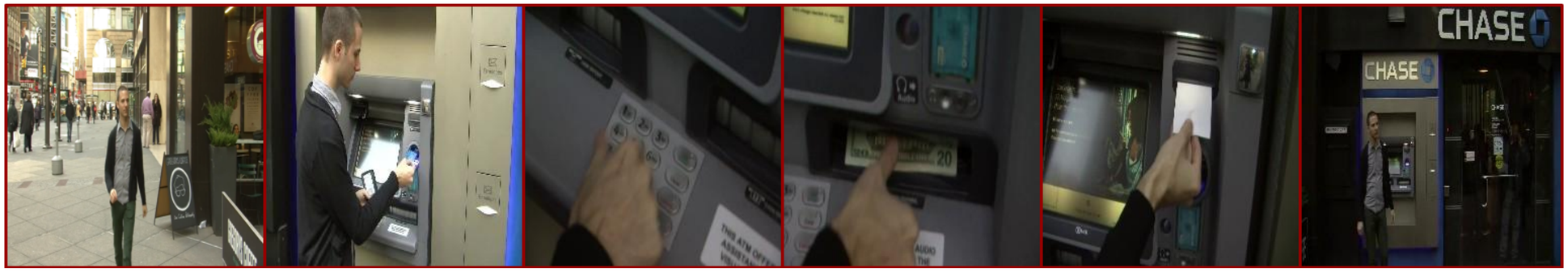


Question Embedding (LSTM)



* slide from Dhruv Batra

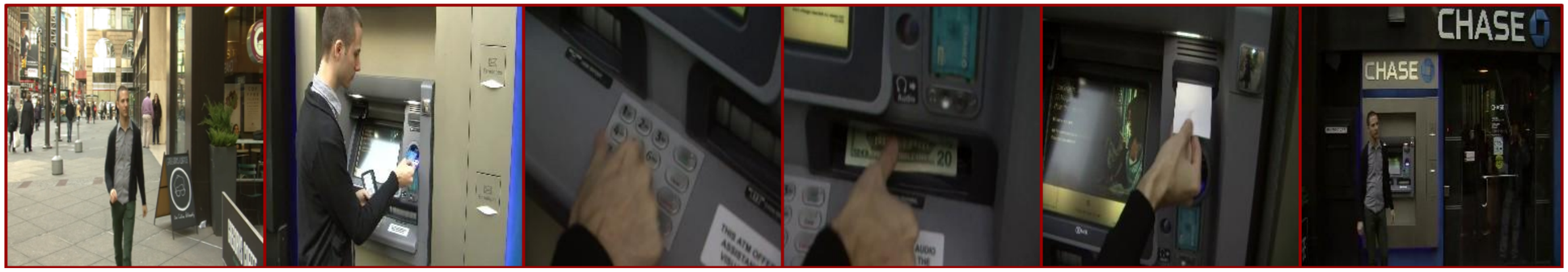
Applications: Activity Detection



[Ma et al., 2014]

Applications: Activity Detection

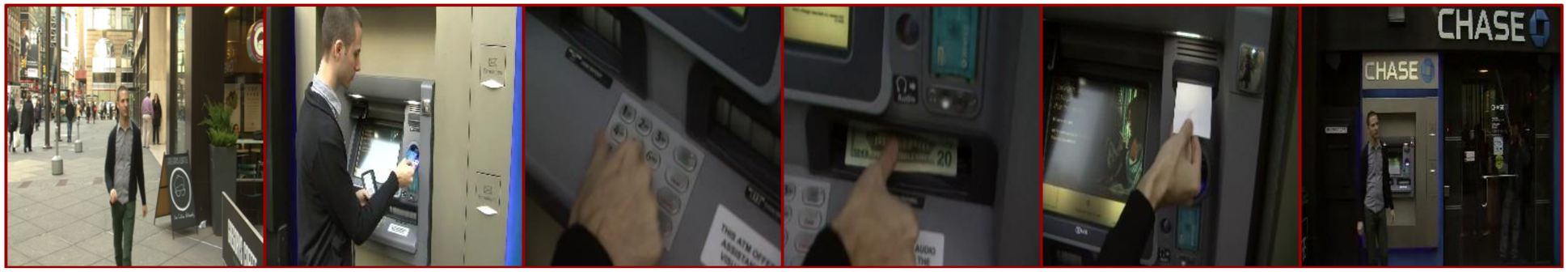
Activity: A collection of human/object movements with a particular semantic meaning



[Ma et al., 2014]

Applications: Activity Detection

Activity: A collection of human/object movements with a particular semantic meaning



Action Recognition: Finding if a video segment contains such a movement

[Ma et al., 2014]

Applications: Activity Detection

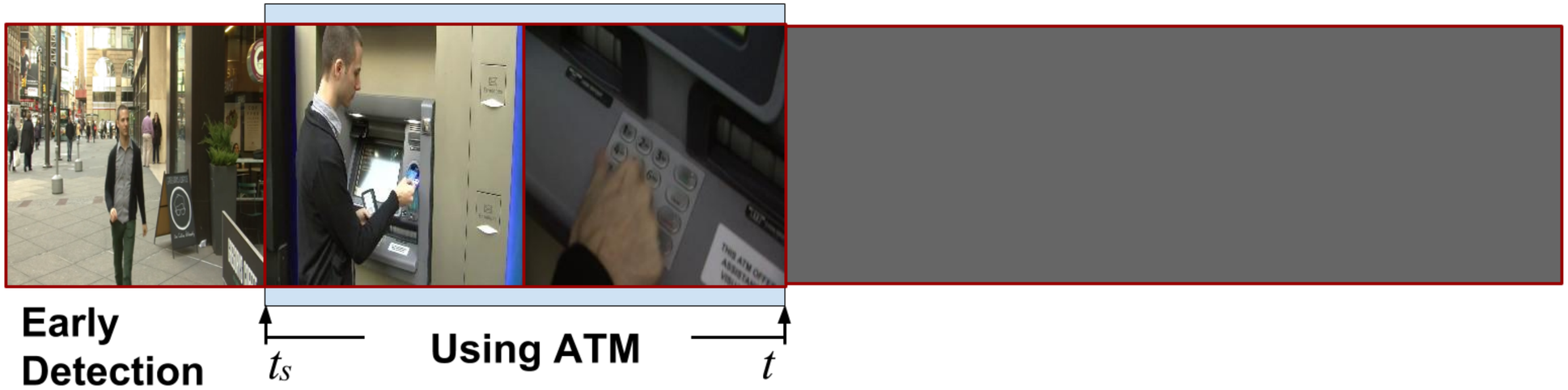
Activity: A collection of human/object movements with a particular semantic meaning

Action Recognition: Finding if a video segment contains such a movement

Action Detection: Finding a segment (beginning and start) and recognize the action in it

[Ma et al., 2014]

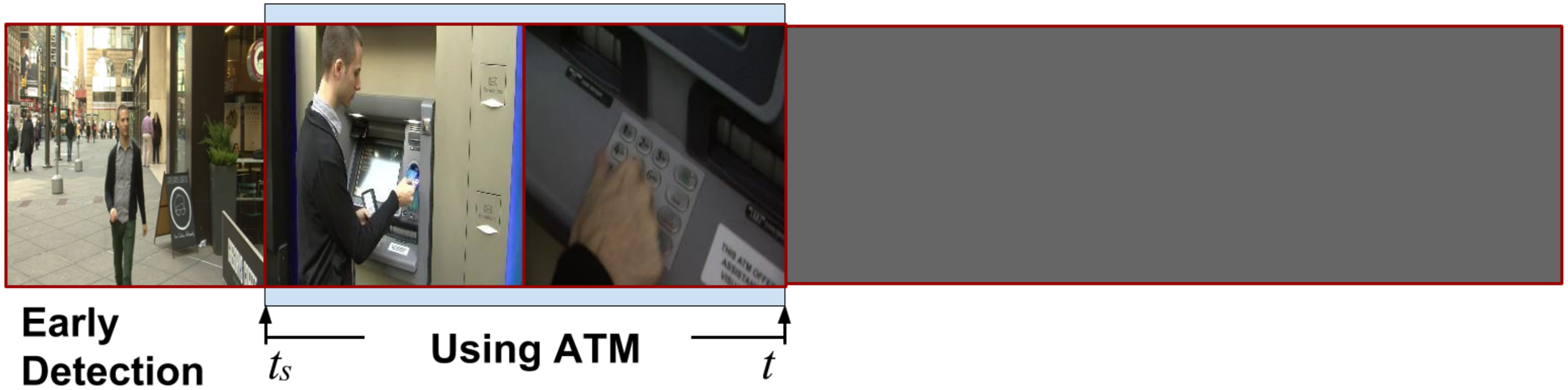
Applications: Activity Detection



[Ma et al., 2014]

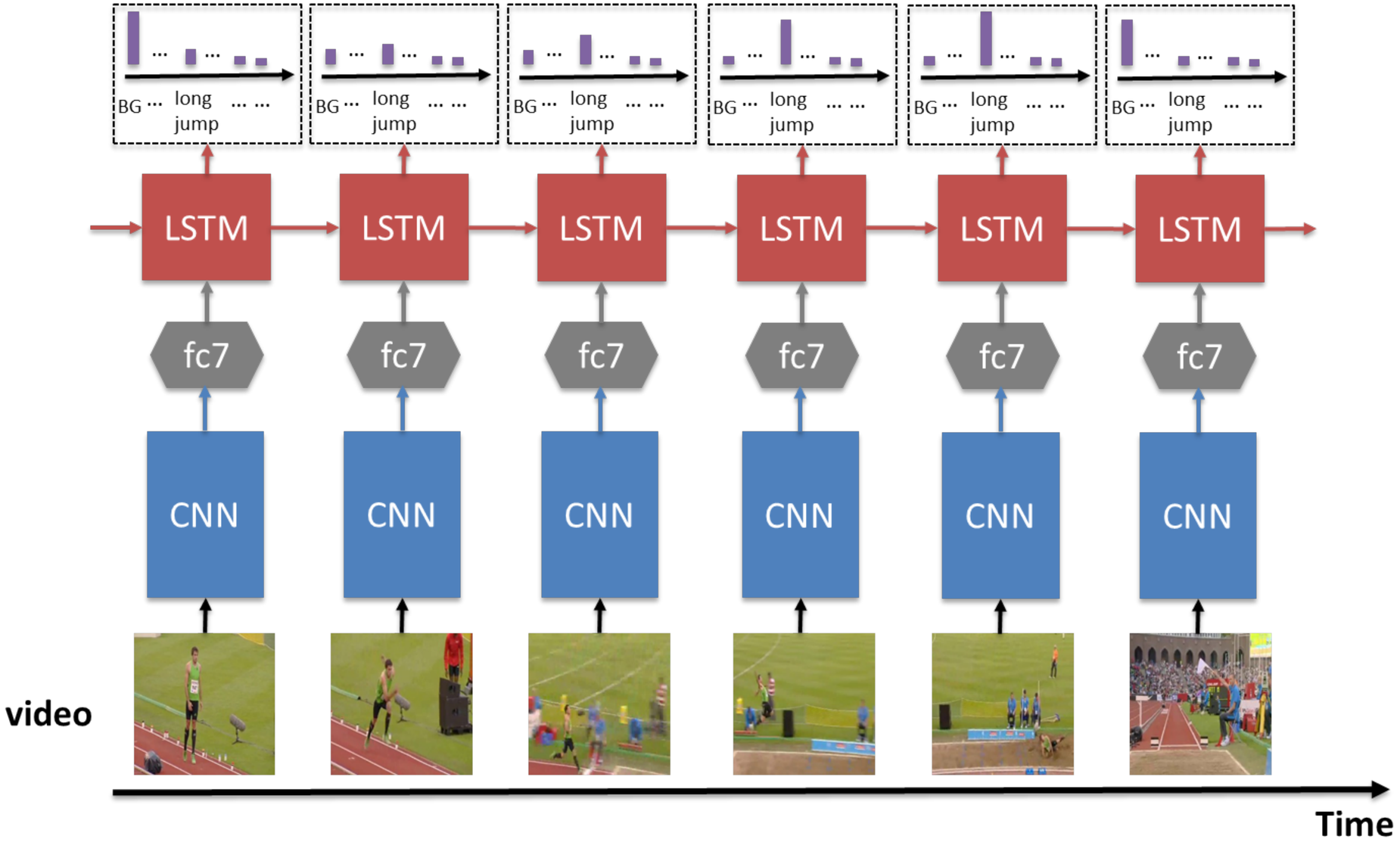
Applications: Activity Detection

Early Detection: Recognize when an action starts and try to predict which action is performed as quickly as possible.



[Ma et al., 2014]

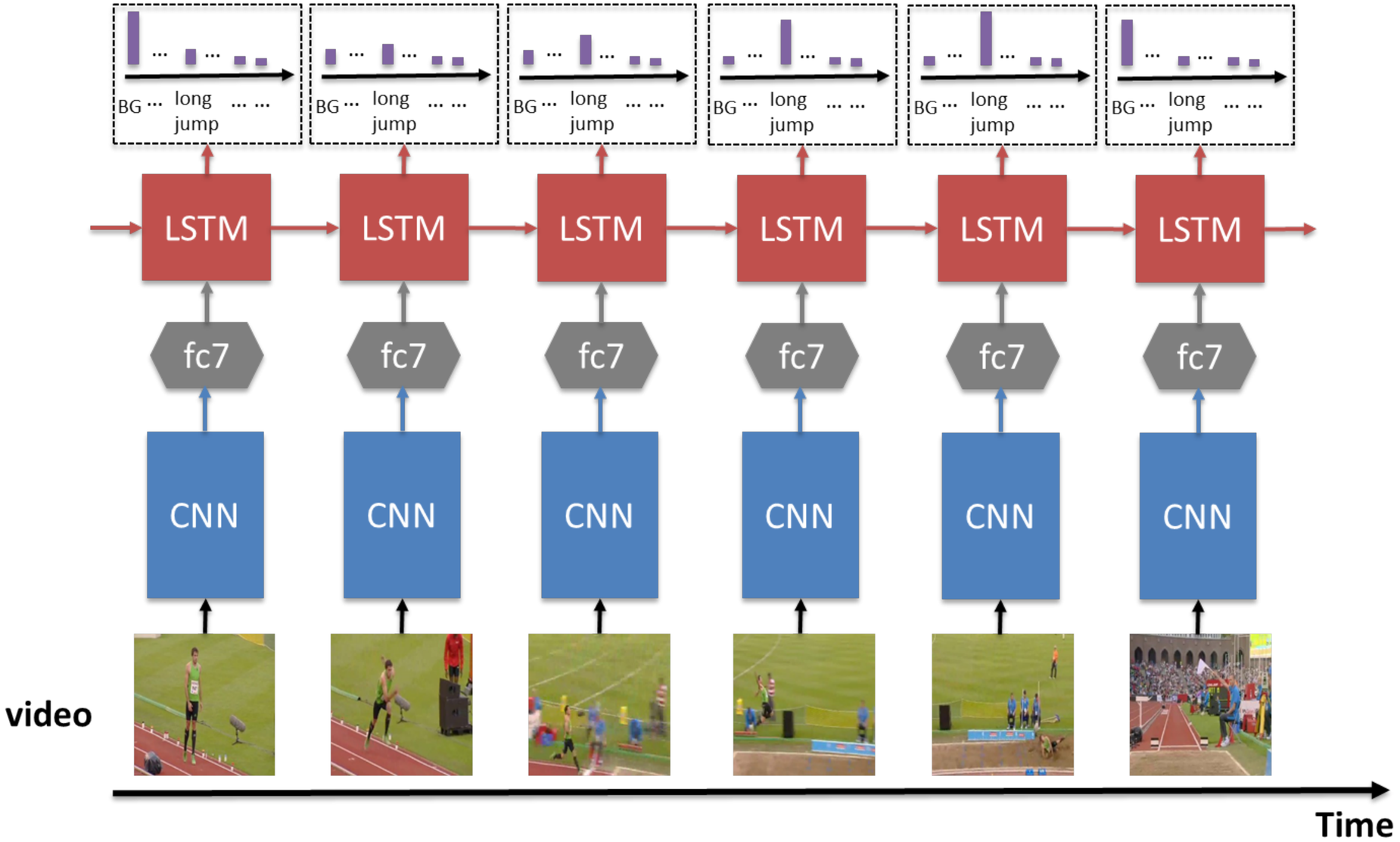
Applications: Activity Detection



[Ma et al., 2014]

Applications: Activity Detection

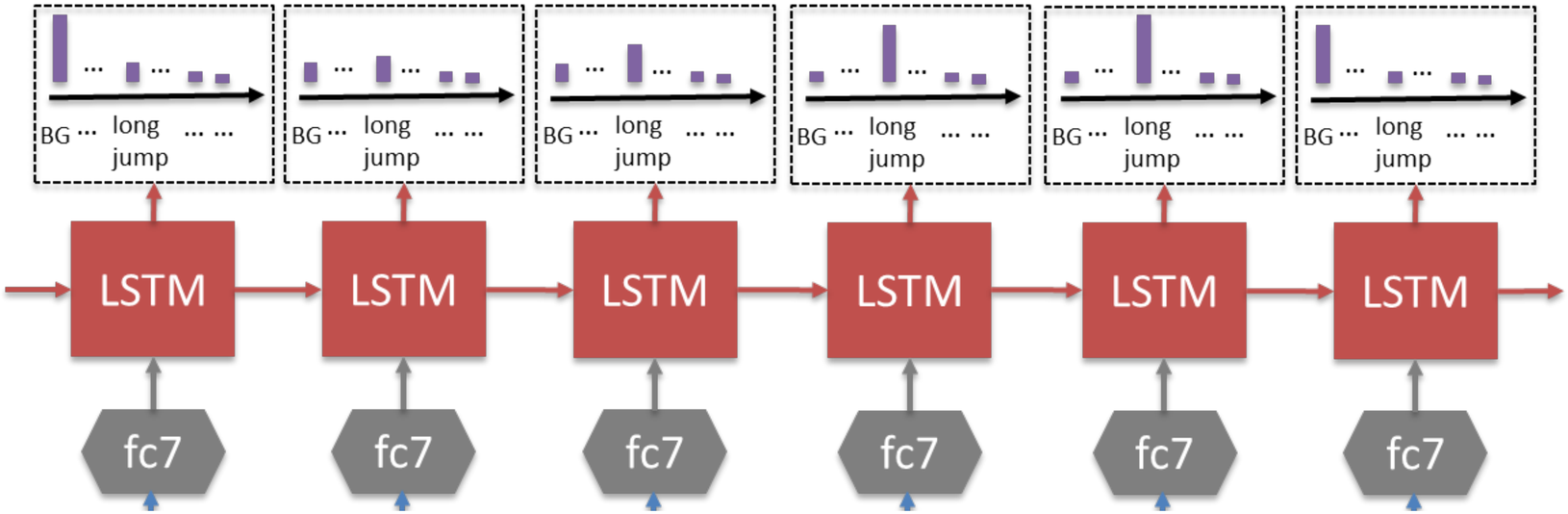
Penalty at every time step is the same

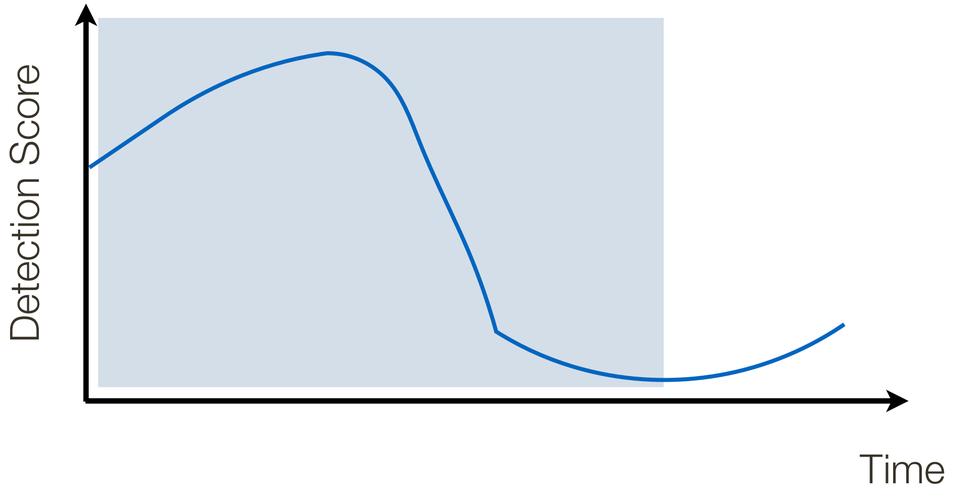


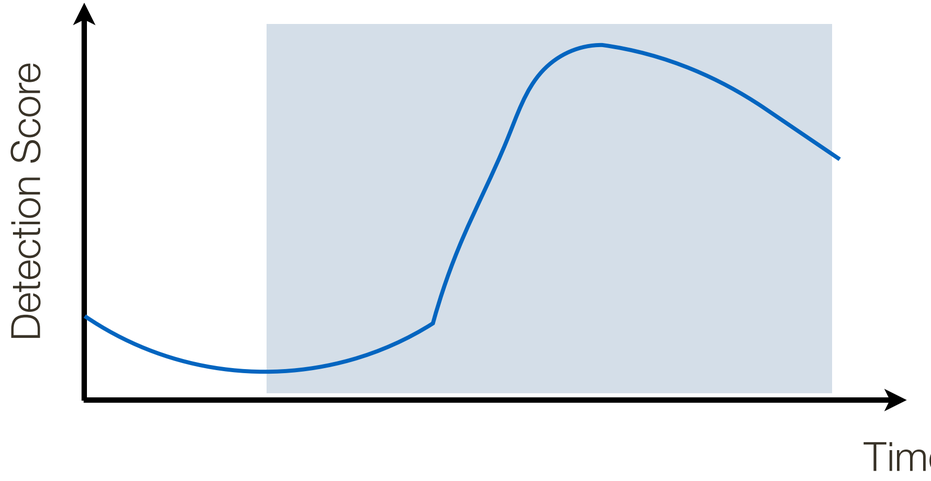
[Ma et al., 2014]

Applications: Activity Detection

Penalty at every time step is the same







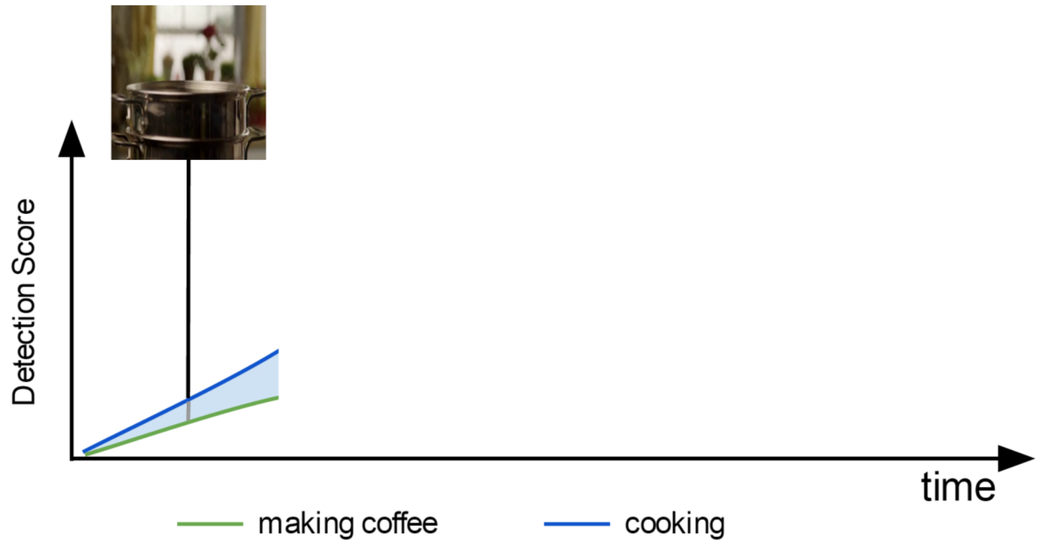
[Ma et al., 2014]

Applications: Activity Detection

As the detector sees more of an action, it should become more confident of

— Detecting the correct action class

— More confident that it is not the incorrect action class



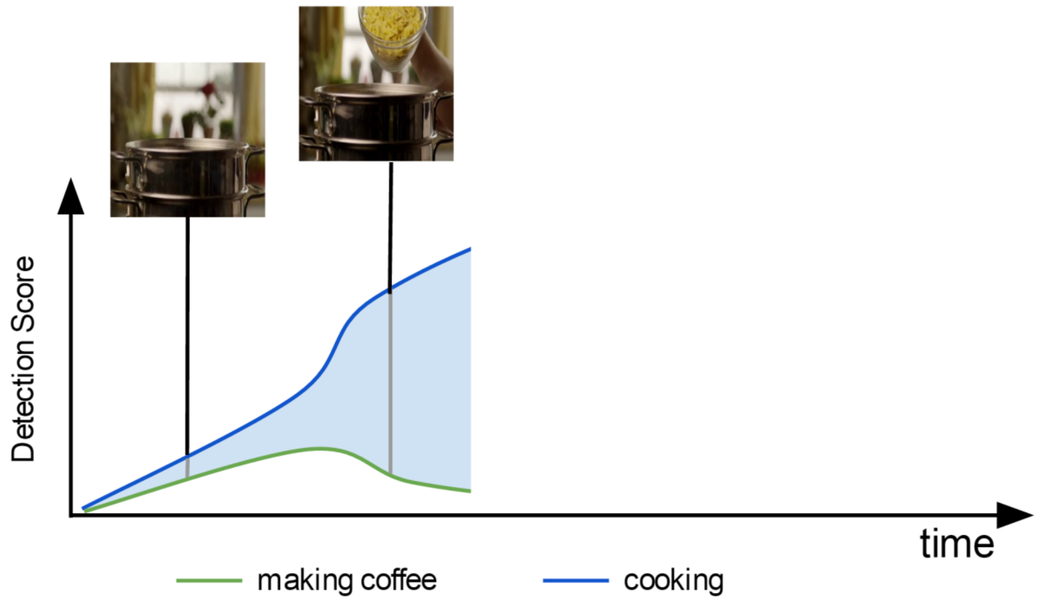
[Ma et al., 2014]

Applications: Activity Detection

As the detector sees more of an action, it should become more confident of

— Detecting the correct action class

— More confident that it is not the incorrect action class



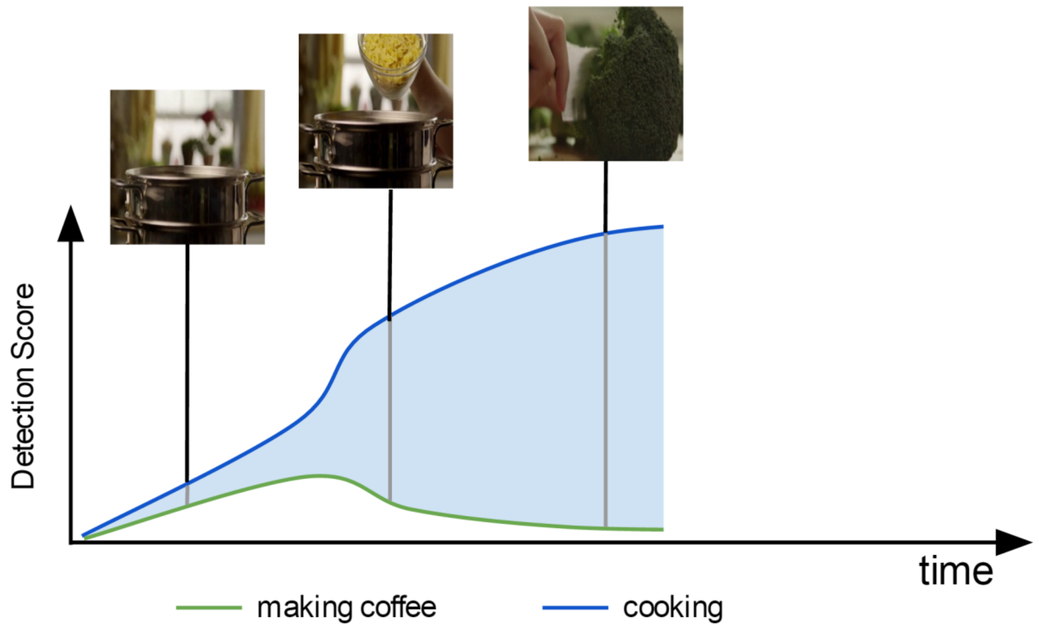
[Ma et al., 2014]

Applications: Activity Detection

As the detector sees more of an action, it should become more confident of

— Detecting the correct action class

— More confident that it is not the incorrect action class



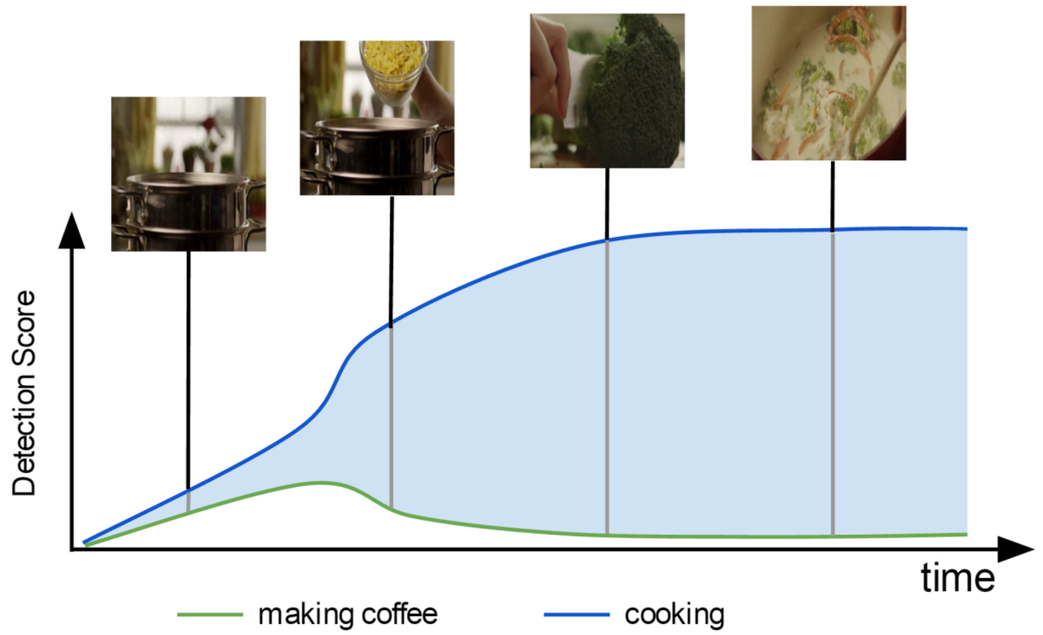
[Ma et al., 2014]

Applications: Activity Detection

As the detector sees more of an action, it should become more confident of

— Detecting the correct action class

— More confident that it is not the incorrect action class



[Ma et al., 2014]

New Class of Loss Functions

Classification loss at time t

Training loss at time t: $\mathcal{L}^t = \mathcal{L}_c^t + \lambda_r \mathcal{L}_r^t$

Ranking loss at time t

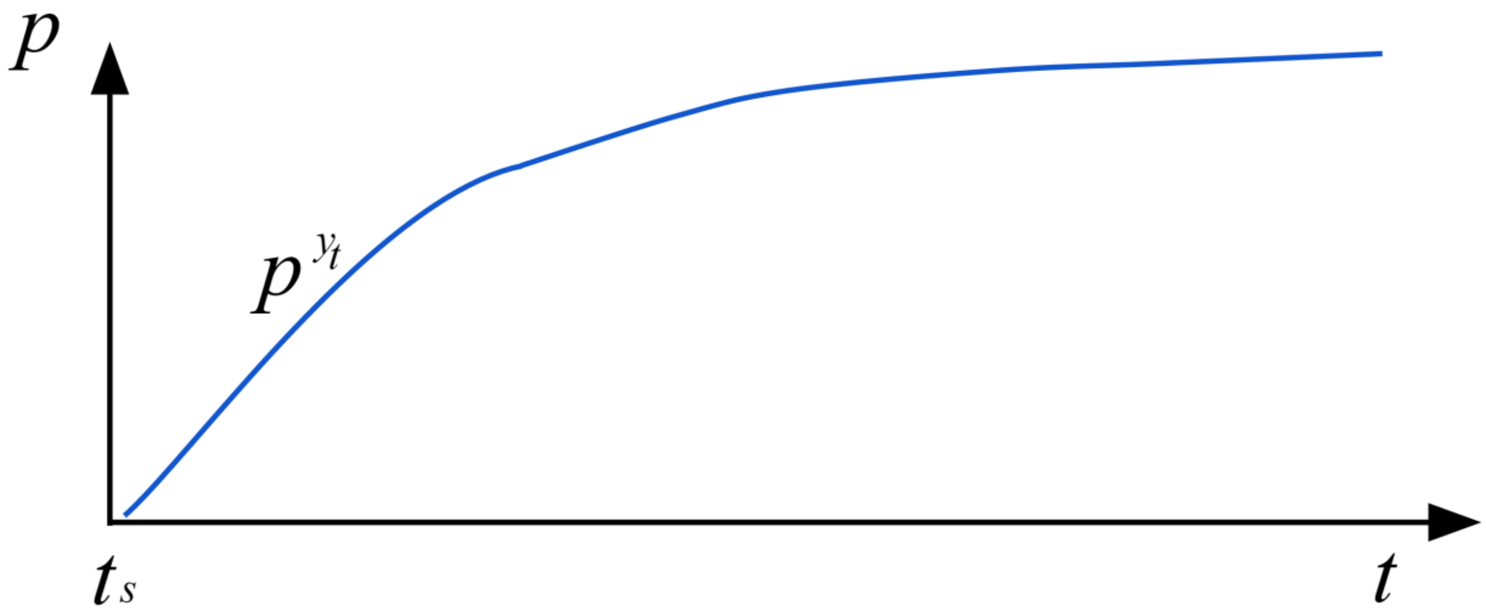
$\mathcal{L}_r^t$ is one of the following:

- $\mathcal{L}_s^t$ ranking loss on detection score
- $\mathcal{L}_m^t$ ranking loss on discriminative margin

[Ma et al., 2014]

Ranking Loss on Detection Score $\mathcal{L}_s^t$

Ideally what we want:

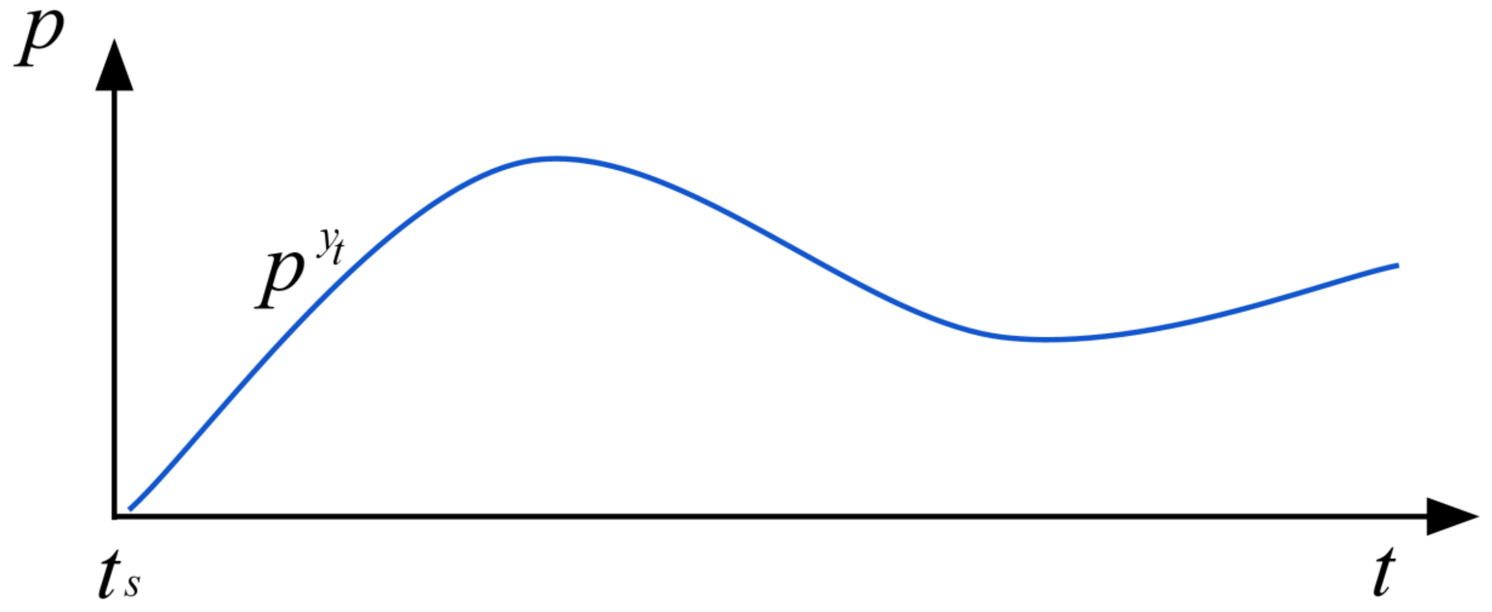


Prediction score of the ground truth action label

[Ma et al., 2014]

Ranking Loss on Detection Score $\mathcal{L}_s^t$

In **Practice:**

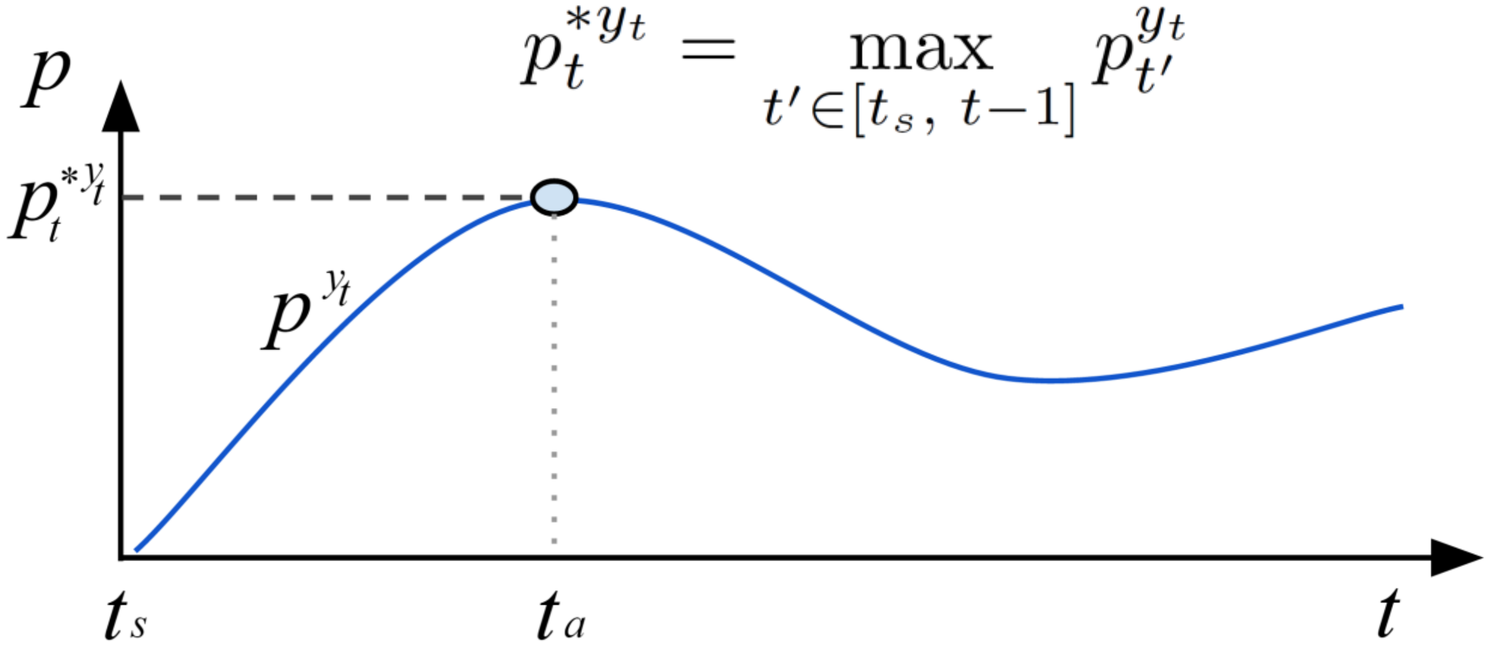


Prediction score of the ground truth action label

[Ma et al., 2014]

Ranking Loss on Detection Score $\mathcal{L}_s^t$

In **Practice:**

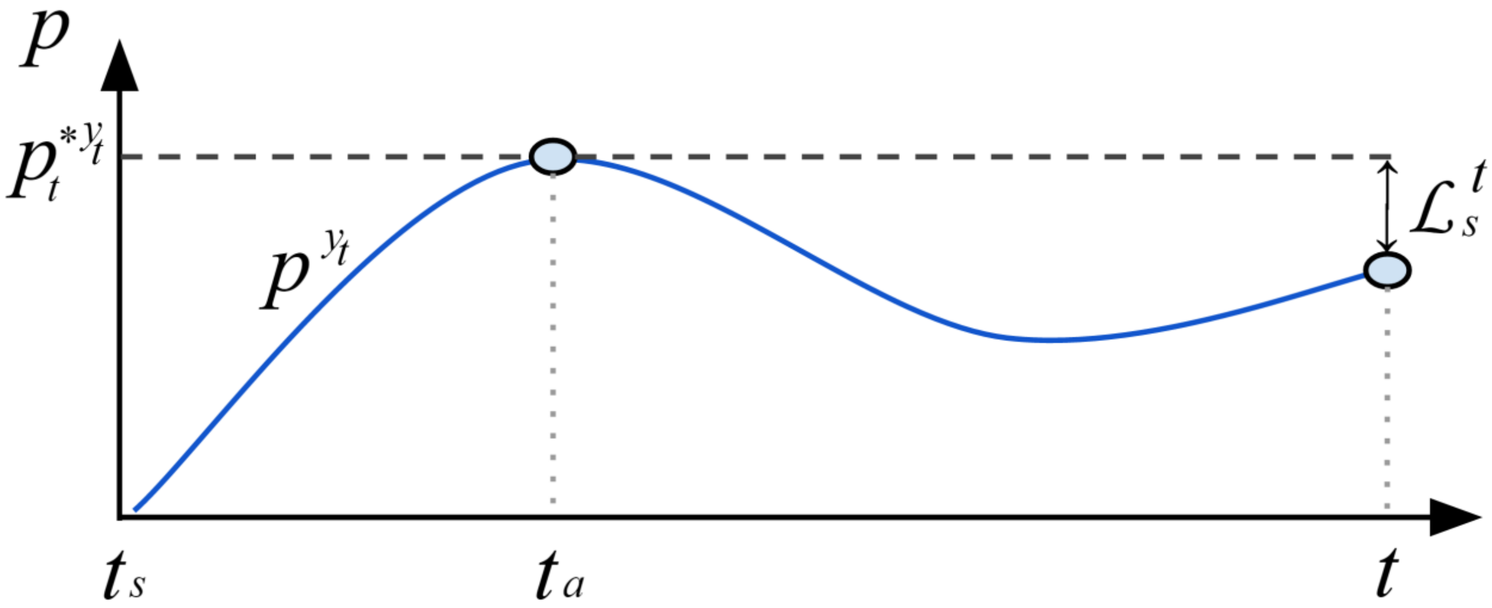


Prediction score of the ground truth action label

[Ma et al., 2014]

Ranking Loss on Detection Score $\mathcal{L}_s^t$

In **Practice:**



Prediction score of the ground truth action label

[Ma et al., 2014]

Applications: Activity Detection

Activity detection performance measured in mAP at different IOU thresholds

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
Heilbron *et al.*	12.5%	11.9%	11.1%	10.4%	9.7%	-	-	-
CNN	30.1%	26.9%	23.4%	21.2%	18.9%	17.5%	16.5%	15.8%
LSTM	48.1%	44.3%	40.6%	35.6%	31.3%	28.3%	26.0%	24.6%
LSTM-m	52.6%	48.9%	45.1%	40.1%	35.1%	31.8%	29.1%	27.2%
LSTM-s	**54.0%**	**50.1%**	**46.3%**	**41.2%**	**36.4%**	**33.0%**	**30.4%**	**28.7%**

LSTM-m LSTM trained using both classification loss and rank loss on *discriminative margin*.

LSTM-s LSTM trained using both classification loss and rank loss on *detection score*.

[Ma et al., 2014]

Applications: Early Activity Detection

Activity early detection performance measured in mAP at different IOU thresholds

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
CNN	27.0%	23.4%	20.4%	17.2%	14.6%	12.3%	11.0%	10.3%
LSTM	49.5%	44.7%	38.8%	33.9%	29.6%	25.6%	23.5%	22.4%
LSTM-m	52.6%	47.9%	41.5%	36.2%	31.4%	27.1%	24.8%	23.5%
LSTM-s	**55.1%**	**50.3%**	**44.0%**	**38.9%**	**34.1%**	**29.8%**	**27.4%**	**26.1%**

Note: first 3/10 of activity is seen by a detector

LSTM-m LSTM trained using both classification loss and rank loss on *discriminative margin*.

LSTM-s LSTM trained using both classification loss and rank loss on *detection score*.

[Ma et al., 2014]

Applications: Early Activity Detection

Activity early detection performance measured in mAP at different IOU thresholds

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
CNN	27.0%	23.4%	20.4%	17.2%	14.6%	12.3%	11.0%	10.3%
LSTM	49.5%	44.7%	38.8%	33.9%	29.6%	25.6%	23.5%	22.4%
LSTM-m	52.6%	47.9%	41.5%	36.2%	31.4%	27.1%	24.8%	23.5%
LSTM-s	**55.1%**	**50.3%**	**44.0%**	**38.9%**	**34.1%**	**29.8%**	**27.4%**	**26.1%**

Note: first 3/10 of activity is seen by a detector

LSTM-m LSTM trained using both classification loss and rank loss on *discriminative margin*.

LSTM-s LSTM trained using both classification loss and rank loss on *detection score*.

Take home: Early detection is only 1-3% worse than sewing the whole sequence

[Ma et al., 2014]

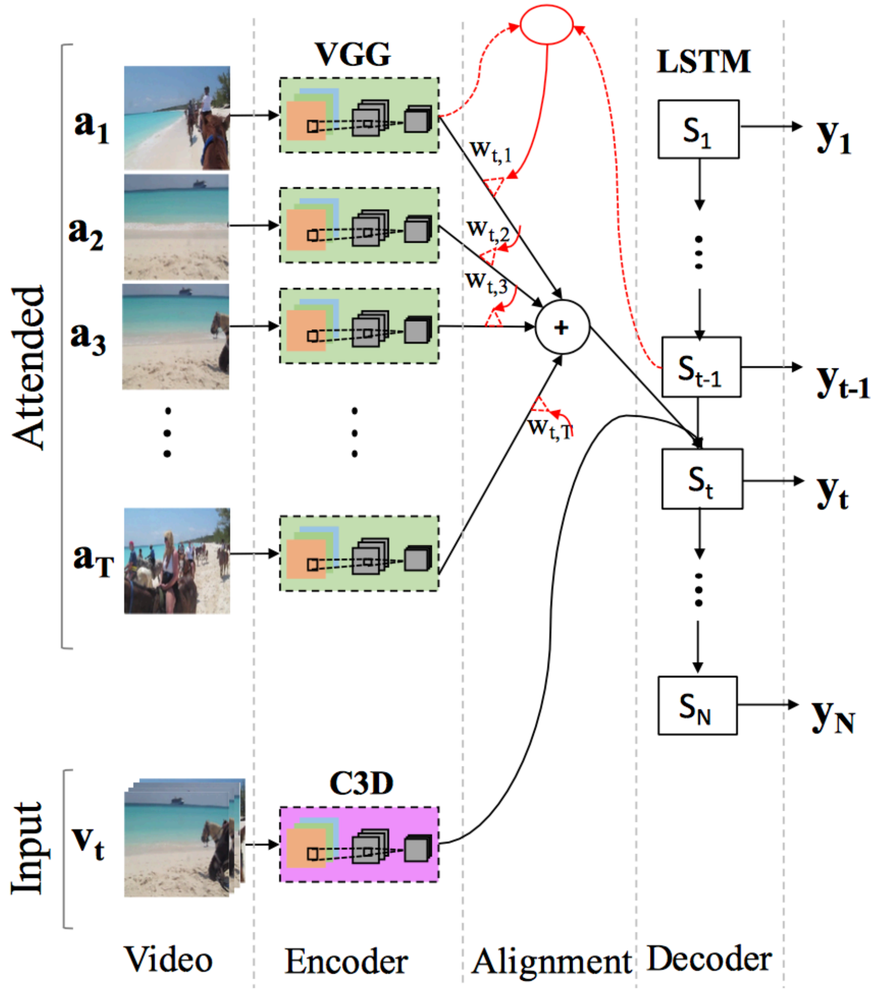
Applications: Activity Detection

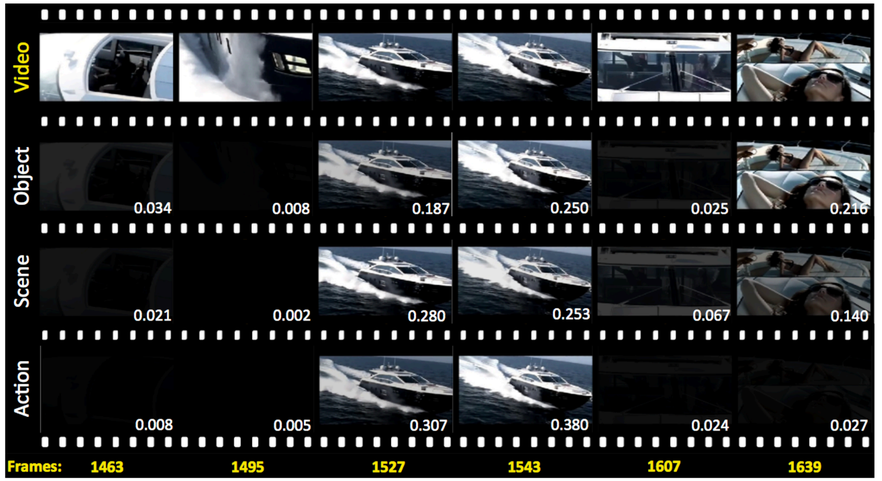
[Ma et al., 2014]

Applications: Activity Detection

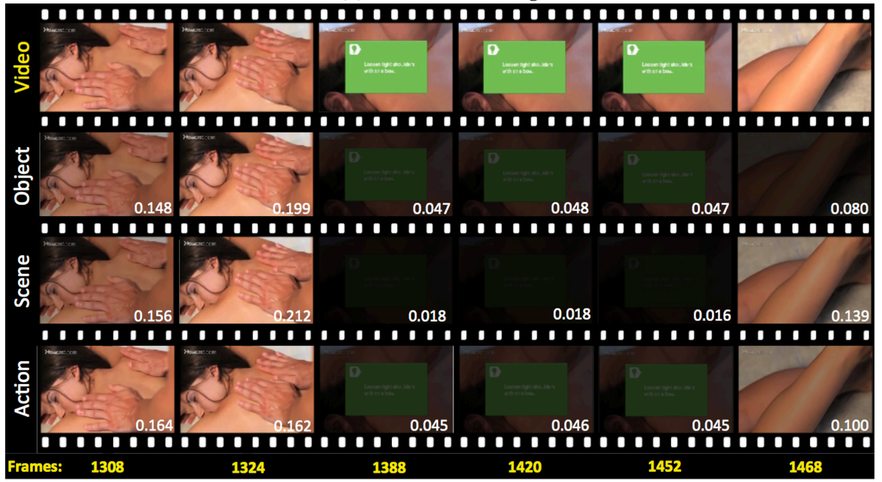
[Ma et al., 2014]

Attention Models for Action Highlighting





(a) Action: *Yatching*



(b) Action: *Having Massage*

[Torabi & Sigal, 2017]