Deep Learning for Computer Vision

Fall 2022

https://cool.ntu.edu.tw/courses/189345 (NTU COOL)

http://vllab.ee.ntu.edu.tw/dlcv.html (Public website)

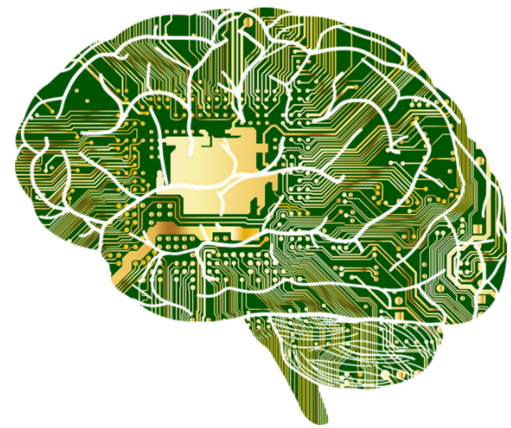
Yu-Chiang Frank Wang 王鈺強, Professor

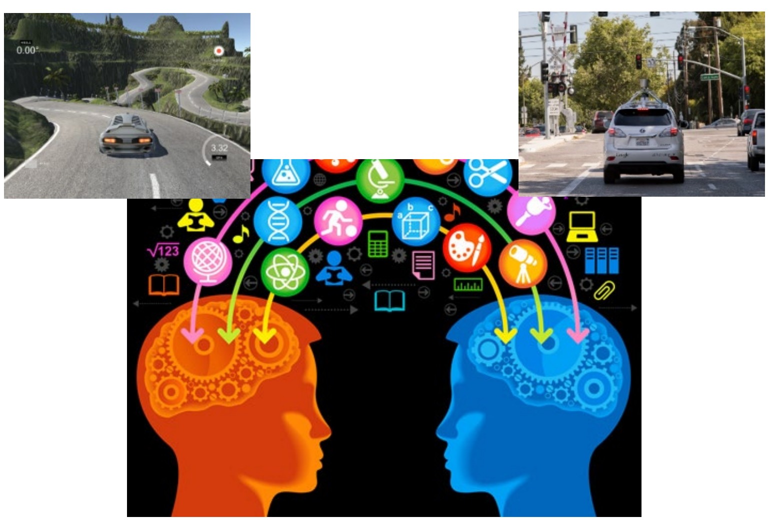
Dept. Electrical Engineering, National Taiwan University

2022/10/25

What to Be Covered Today…

- Transfer Learning
  - Visual Classification – Domain Adaptation
  - Visual Synthesis – Style Transfer
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Selected Models for Sequence-to-Sequence Learning
  - Attention in RNN





Transfer Learning for Image Synthesis

- Cross-Domain Image Translation
  - Pix2pix: Pairwise cross-domain training data
  - CycleGAN/DualGAN/DiscoGAN: Unpaired cross-domain training data
  - UNIT: Learning cross-domain image representation (with unpaired training data)
  - AdaIN: Single-image arbitrary style transfer in real-time
  - Beyond image translation

Pix2pix

- Image-to-image translation with conditional adversarial networks (CVPR'17)
  - Can be viewed as image style transfer

Sketch Photo

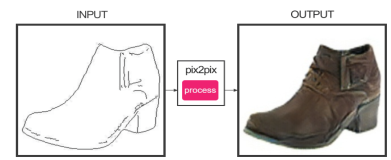
Isola et al. " Image-to-image translation with conditional adversarial networks." *CVPR* 2017.

Pix2pix

$\{\tilde{x}, x_{in}\} \rightarrow \boxed{D} \rightarrow T/F$

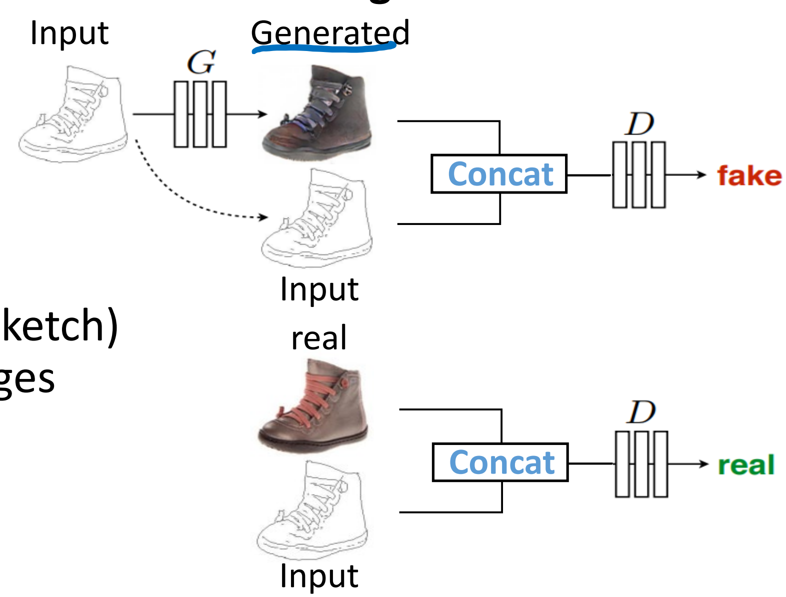
$\{x_{real}, x_{in}\}$

Testing Phase



- **Goal / Problem Setting**
  - Image translation across two distinct domains (e.g., sketch v.s. photo)
  - Pairwise training data
- Method: Conditional GAN
  - Example: Sketch to Photo
    - Generator
      Input: Sketch
      Output: Photo
    - Discriminator
      Input: Concatenation of Input(Sketch) & Synthesized/Real(Photo) images
      Output: Real or Fake

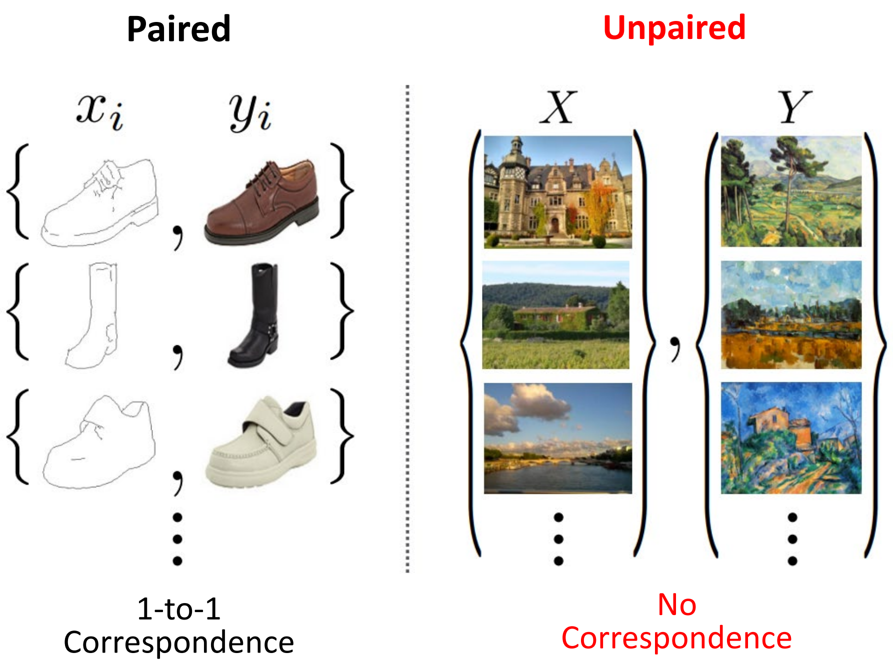
Training Phase



Isola et al. " Image-to-image translation with conditional adversarial networks." *CVPR* 2017.

CycleGAN/DiscoGAN/DualGAN

- CycleGAN
  - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks -to-image translation with conditional adversarial networks



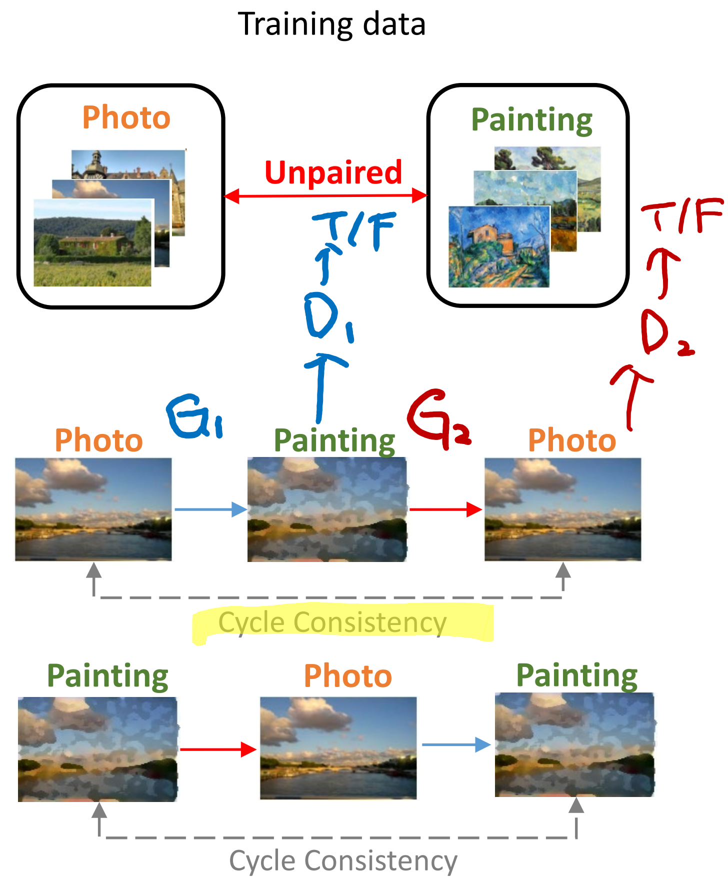
- Easier to collect training data
- More practical

Zhu et al. "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks." *CVPR* 2017.

CycleGAN

- **Goal / Problem Setting**
  - Image translation across two distinct domains
  - Unpaired training data

- **Idea**
  - Autoencoding-like image translation
  - Cycle consistency between two domains

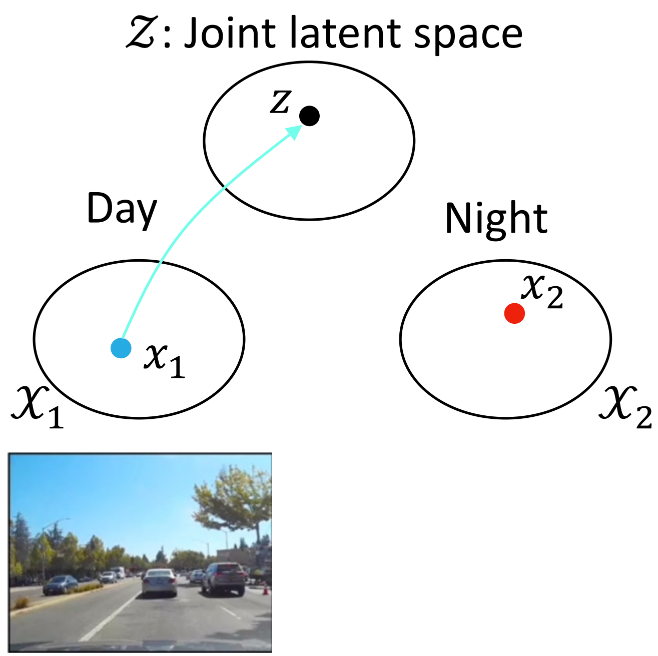


Zhu et al. "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks." *CVPR* 2017.

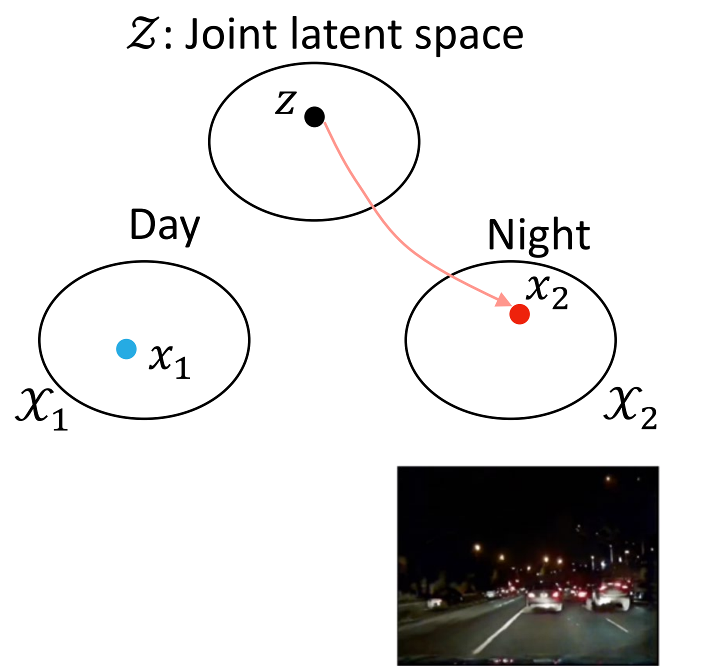
UNIT

- Unsupervised Image-to-Image Translation Networks (NIPS'17)
  - Image translation via learning cross-domain joint representation

Stage1: Encode to the joint space



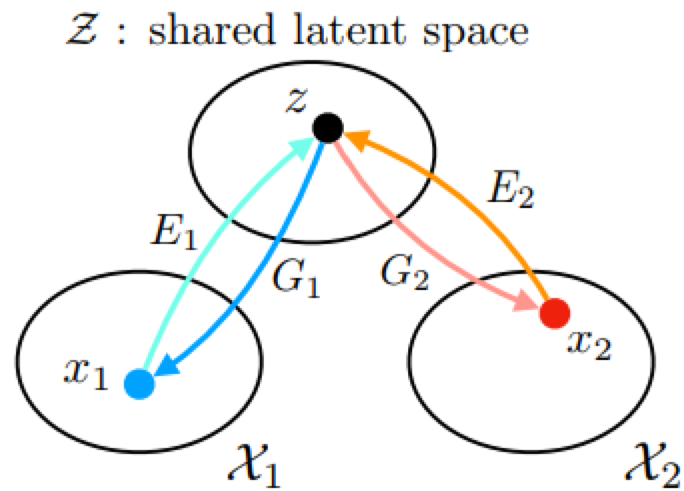
Stage2: Generate cross-domain images



Liu et al., "Unsupervised image-to-image translation networks.", NIPS 2017

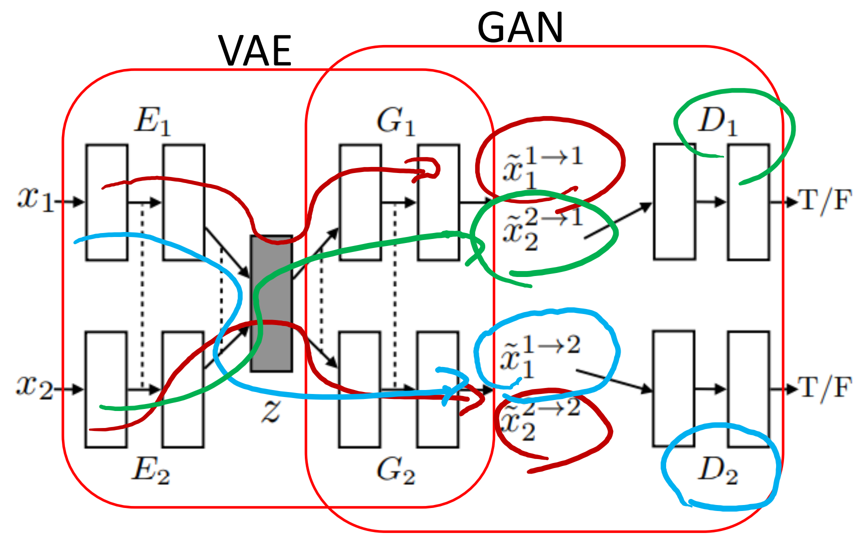
UNIT

- Goal/Problem Setting
  - Image translation
    across two distinct domains
  - Unpaired training image data
- Idea
  - Based on two parallel VAE-GAN models



source

tgt



Liu et al., "Unsupervised image-to-image translation networks.", NIPS 2017

Transfer Learning for Image Synthesis

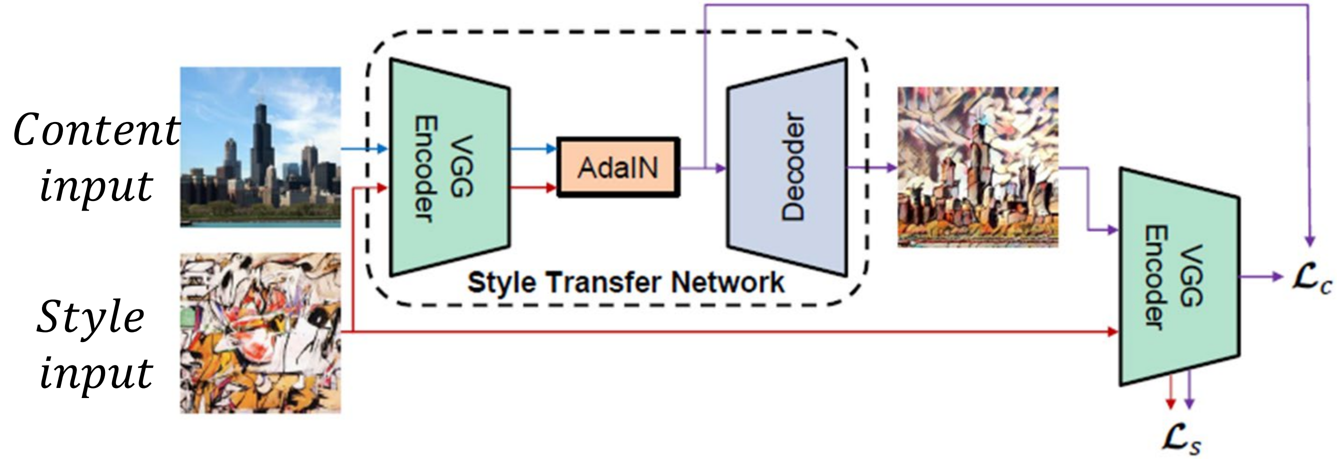
- Cross-Domain Image Translation
  - Pix2pix: Pairwise cross-domain training data
  - CycleGAN/DualGAN/DiscoGAN: Unpaired cross-domain training data
  - UNIT: Learning cross-domain image representation (with unpaired training data)
  - AdaIN: Single-image arbitrary style transfer in real-time
  - Beyond image translation

AdaIN

- We've talked about style transfer methods like Pix2Pix or CycleGAN.
- Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization (ICCV'17)
  - Single-image arbitrary style transfer in real-time

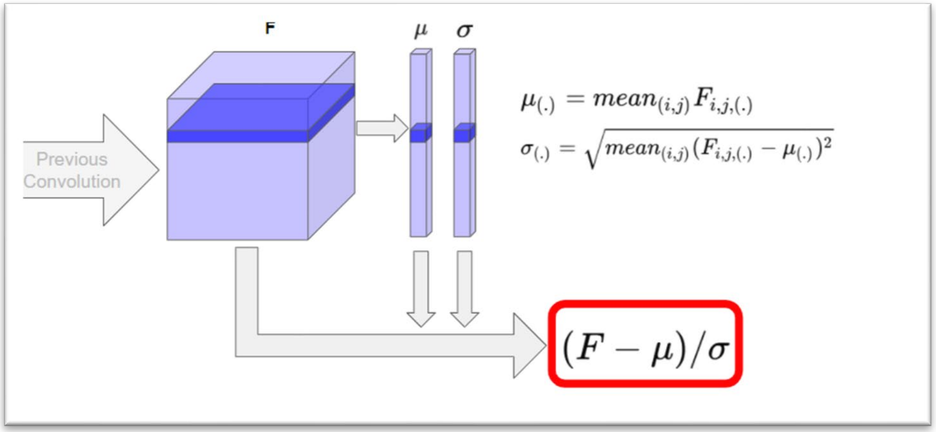
Huang *et al.* " Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization . " *ICCV* 2017

AdaIN

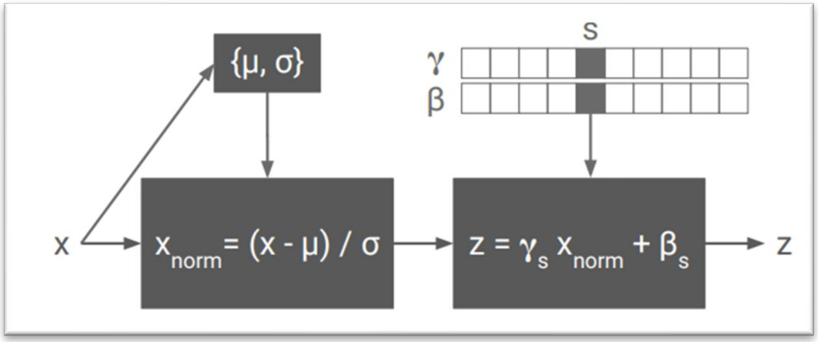


- Adaptive Instance Normalization

$$\mathrm{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$



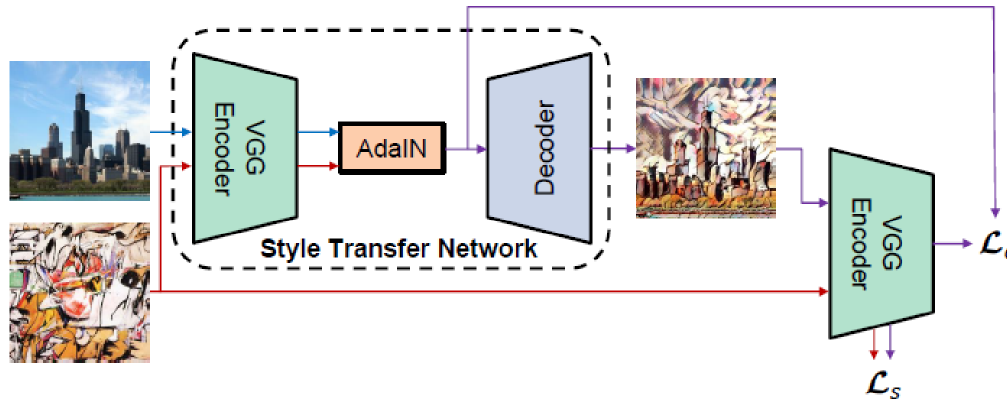
- x: content input, y: style input
- No learnable affine parameters
- Perform style transfer in the feature space



Huang *et al.* " Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization . " *ICCV* 2017 & http://shorturl.at/EW149

AdaIN (cont'd)

- f: Encoder, g: Decoder



Content loss (via content/perceptual consistency):

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2 .$$

Style loss (via Gram matrix loss):

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l . \quad \Rightarrow \quad E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

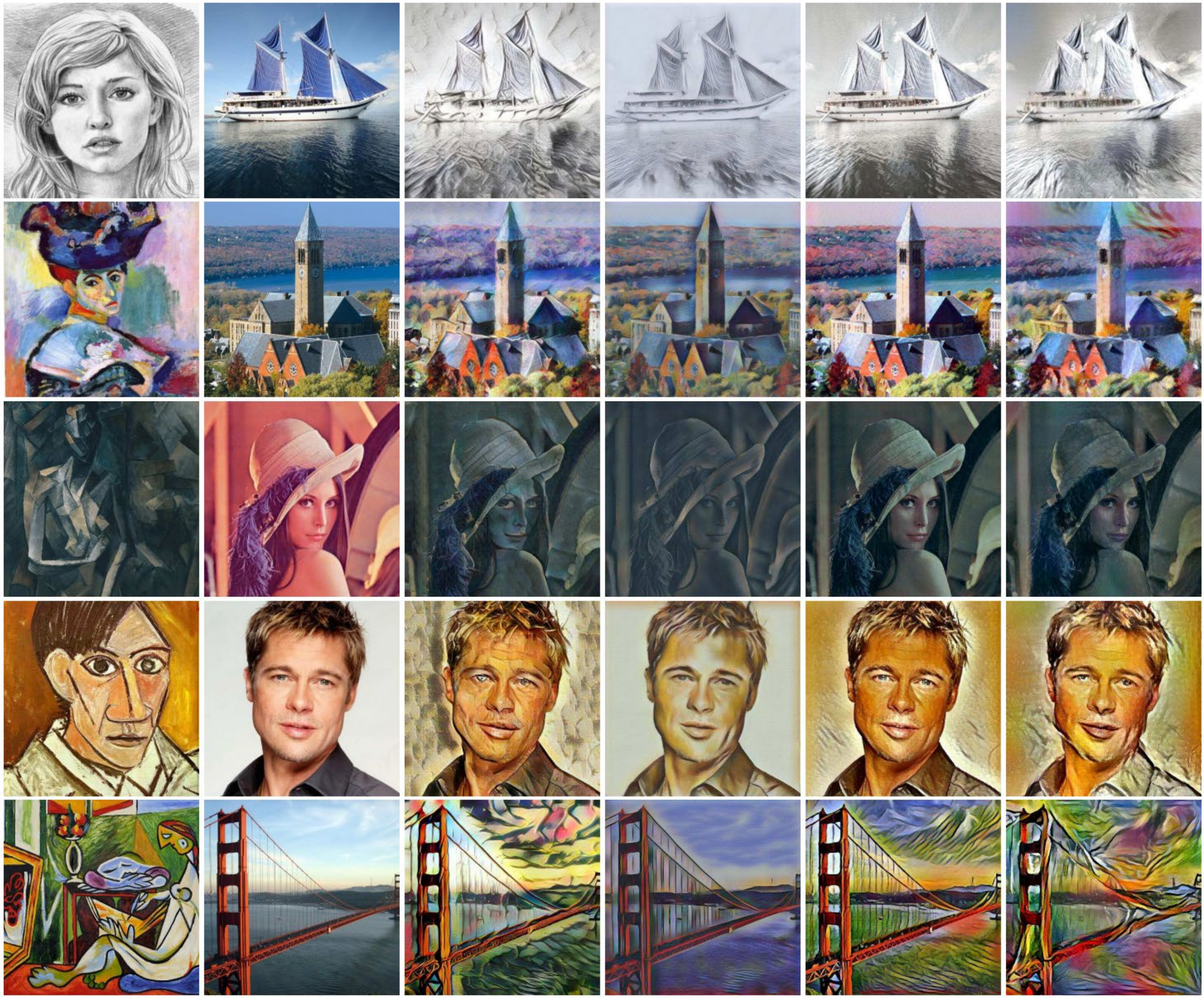
$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum^{L} w_l E_l$$

$$\Rightarrow \quad \mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Huang *et al.* " Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization . " *ICCV* 2017 & http://shorturl.at/egtQT

AdaIN

- Qualitative results



Style Content Ours Chen and Schmidt Ulyanov *et al.* Gatys *et al.*

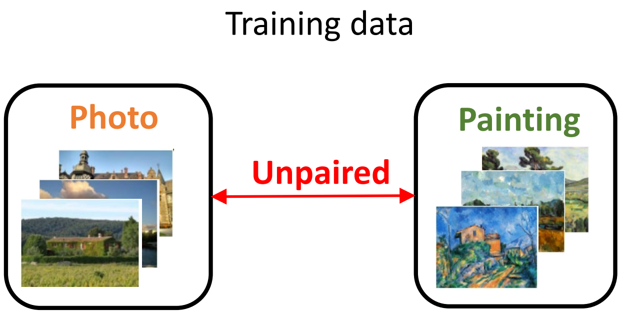
Huang *et al.* " Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization . " *ICCV* 2017

Transfer Learning for Image Synthesis

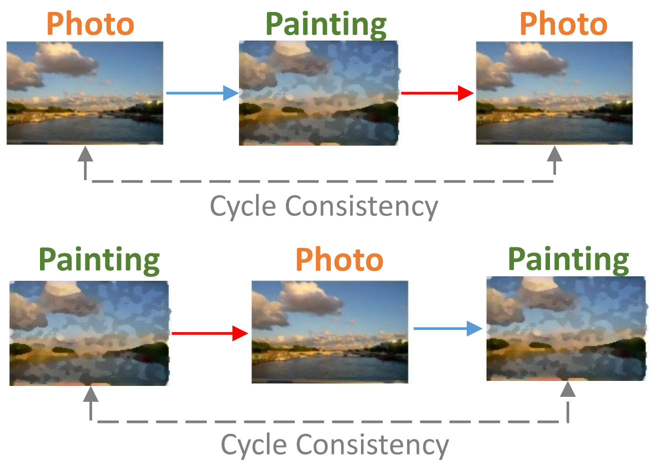
- Cross-Domain Image Translation
  - Pix2pix: Pairwise cross-domain training data
  - CycleGAN/DualGAN/DiscoGAN: Unpaired cross-domain training data
  - UNIT: Learning cross-domain image representation (with unpaired training data)
  - AdaIN
  - Beyond image translation

Revisit: CycleGAN

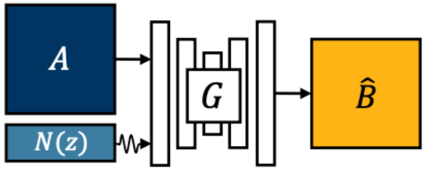
- **Goal / Problem Setting**
  - Image translation across two distinct domains
  - Unpaired training data



- **Idea**
  - Autoencoding-like image translation
  - Cycle consistency between two domains

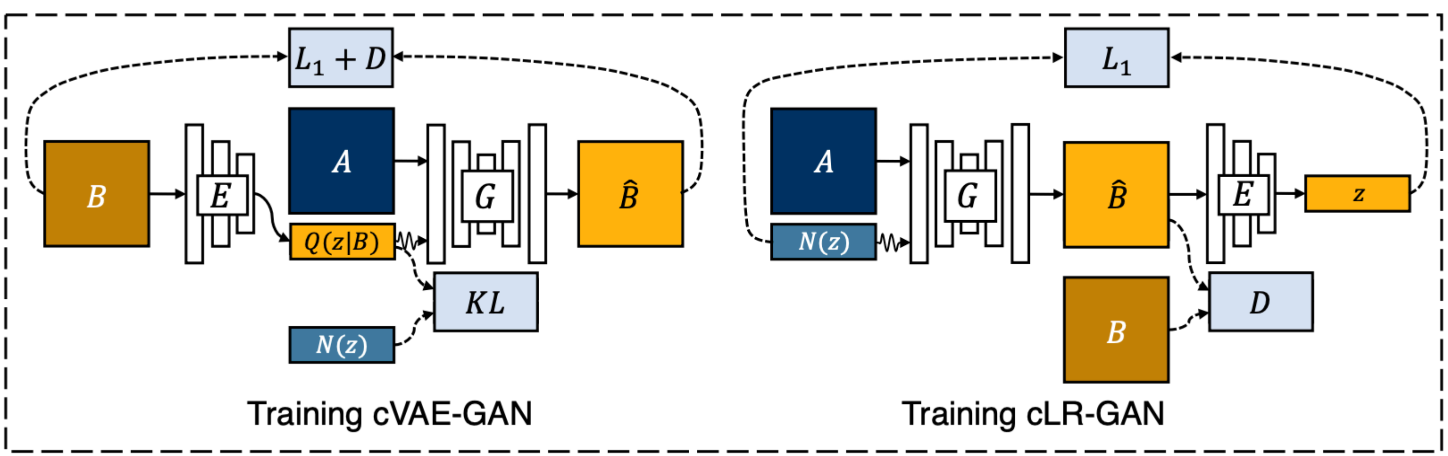


BicycleGAN



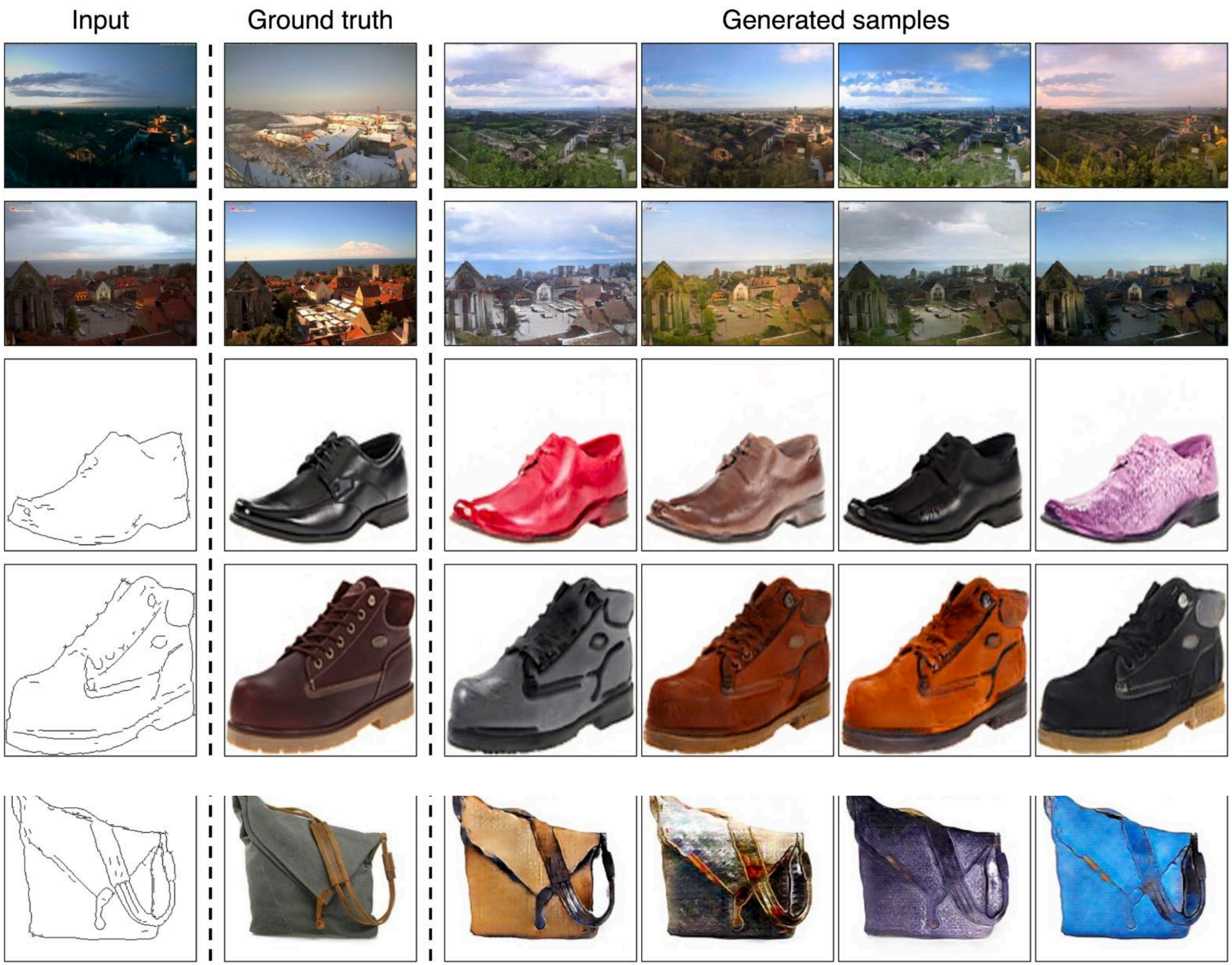
(a) Testing Usage for all models

- **Toward Multimodal Image-to-Image Translation** (NIPS’17)
- **Goal / Problem Setting**
  - Producing diverse images across two distinct domains.
  - Pairwise training data
- **Idea**
  - Combine conditional VAE-GAN and conditional Latent Regressor GAN.



Training BicycleGAN

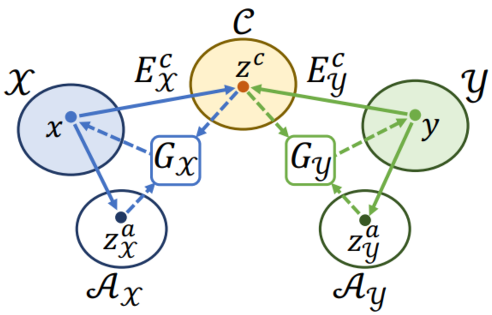
BicycleGAN - Experiment

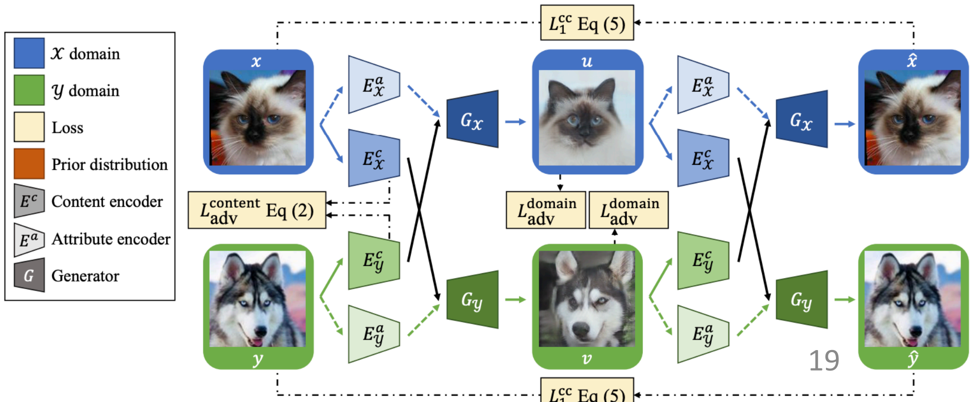


Zhu *et al*., Toward Multimodal Image-to-Image Translation, NIPS 2017

DRIT

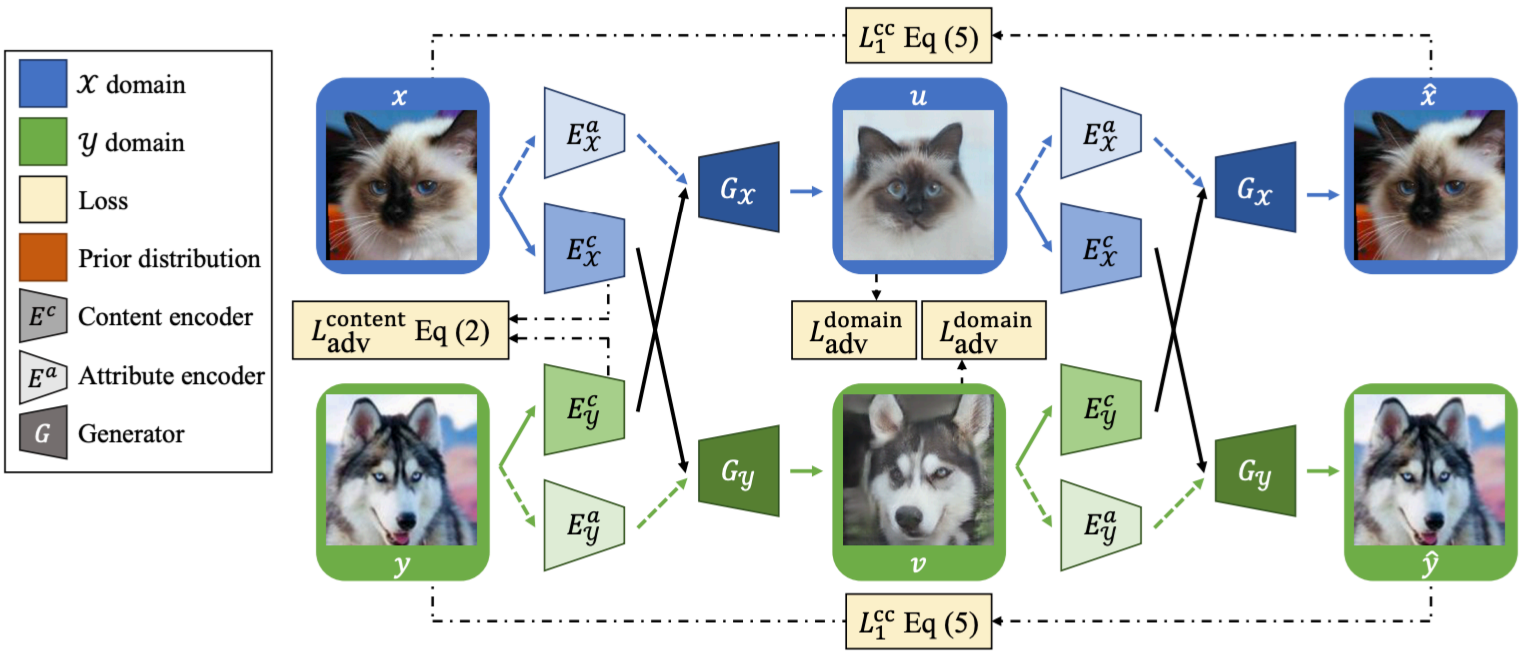
- **Diverse Image-to-Image Translation via Disentangled Representations** (ECCV'18 oral)
- **Goal / Problem Setting**
  - Producing diverse images across two distinct domains.
  - Unpaired training data
- **Idea**
  - Disentangle latent representation into domain-invariant and domain-specific features
  - Generate cross-domain images by swapping the latent feature from each domain.
  - Applied cross-cycle consistency





Method – Main Framework

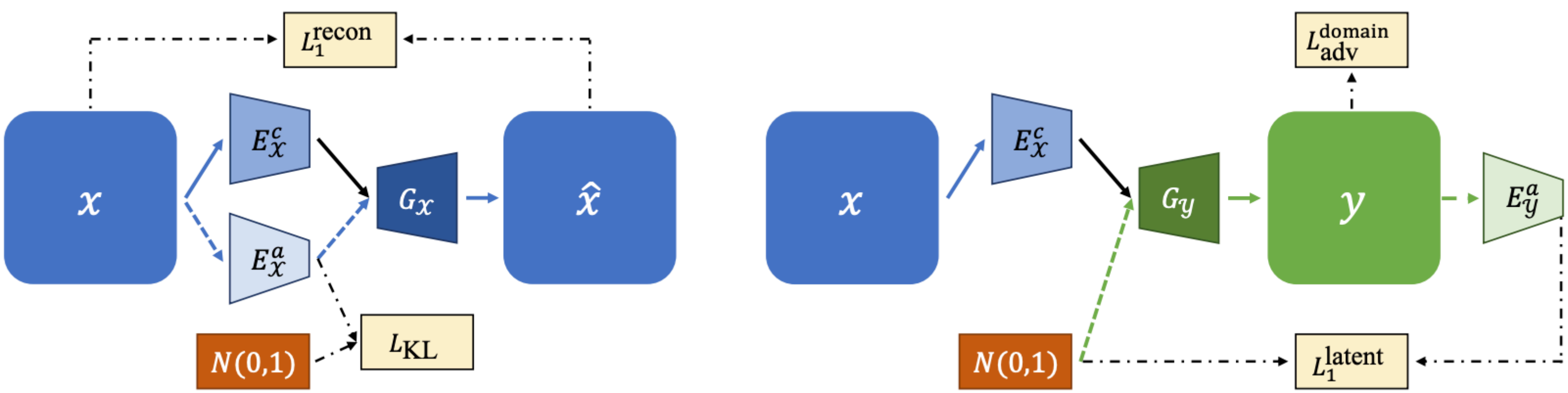
Attribute: species
Content: pose (style)



Lee *et al.*, Diverse Image-to-Image Translation via Disentangled Representations, ECCV 2018 (oral)

Method – For Attribute Features

- **KL loss**:
  perform stochastic sampling at test time.
- **Latent regression loss**:
  encourage invertible mapping btw image and latent representations



Lee *et al*., Diverse Image-to-Image Translation via Disentangled Representations, ECCV 2018 (oral)

Method – Inference phase

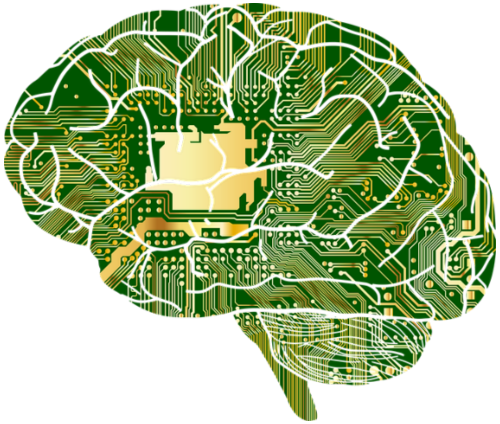
(b) Testing with random attributes (c) Testing with a given attribute

Lee *et al*., Diverse Image-to-Image Translation via Disentangled Representations, ECCV 2018 (oral)

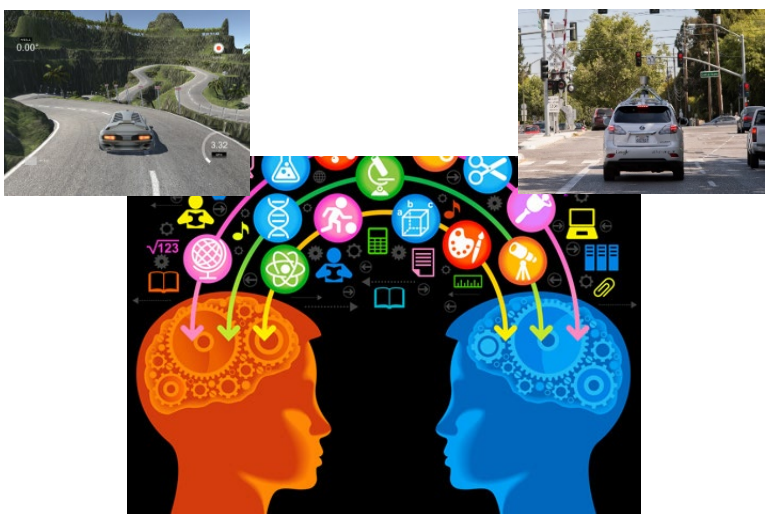
Example Results

Lee *et al*., Diverse Image-to-Image Translation via Disentangled Representations, ECCV 2018 (oral)

What to Be Covered Today…

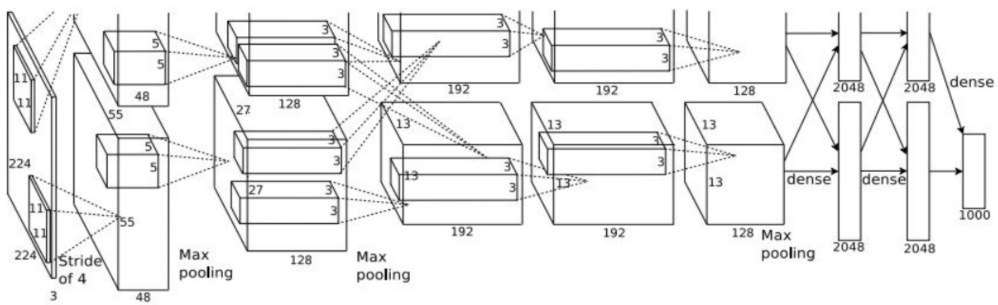


- Transfer Learning
    - Visual Classification – Domain Adaptation
    - Visual Synthesis – Style Transfer
- Recurrent Neural Networks
    - From RNN to LSTM & GRU
    - Selected Models for Sequence-to-Sequence Learning
    - Attention in RNN



What Are The Limitations of CNN?

- Deal with image data
    - Both input and output are images/vectors
- Simply feed-forward processing



Example of (Visual) Sequential Data

https://quickdraw.withgoogle.com/#

More Applications in Vision

Image Captioning

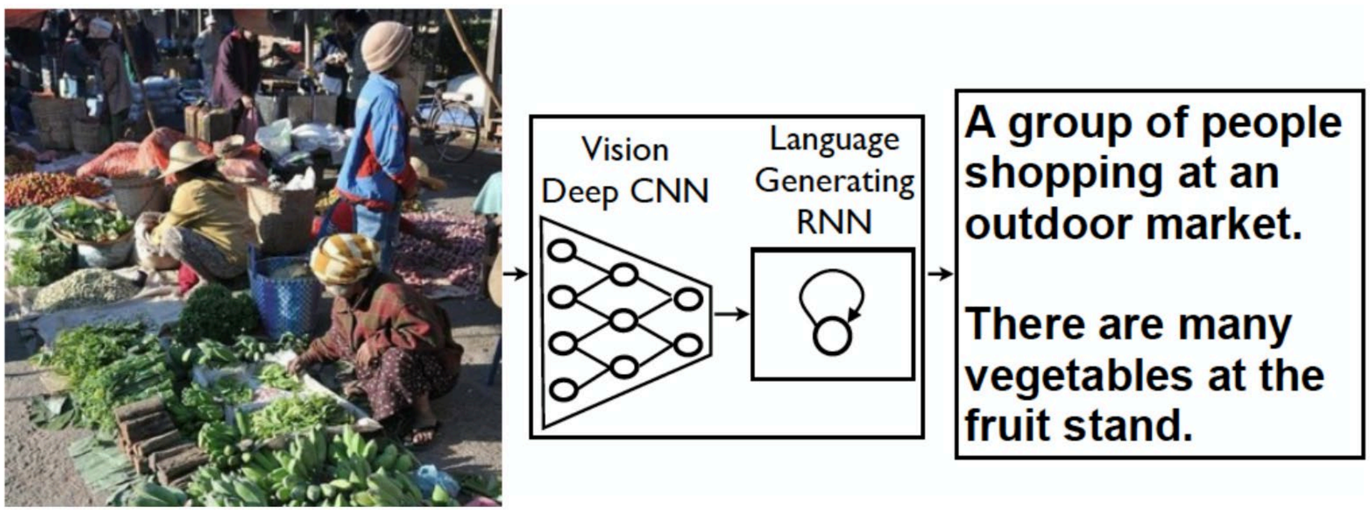


Figure from Vinyals et al, “Show and tell: A neural image caption generator”, CVPR 2015

More Applications in Vision

Visual Question Answering (VQA)

Input

Q: What endangered animal is featured on the truck?

output

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

Q: Where will the driver go if turning right?

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

Q: When was the picture taken?

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

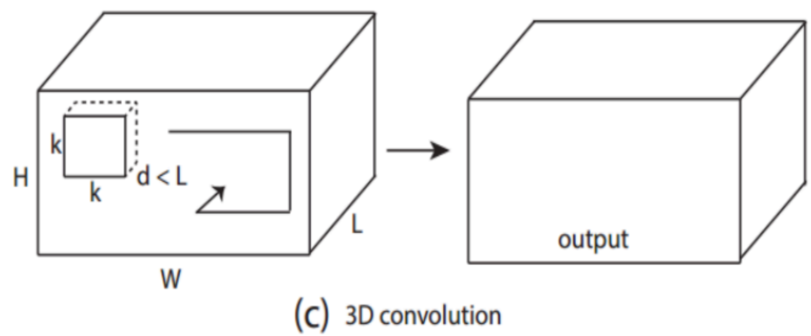
Q: Who is under the umbrella?

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

Figure from Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016

How to Model Sequential Data?

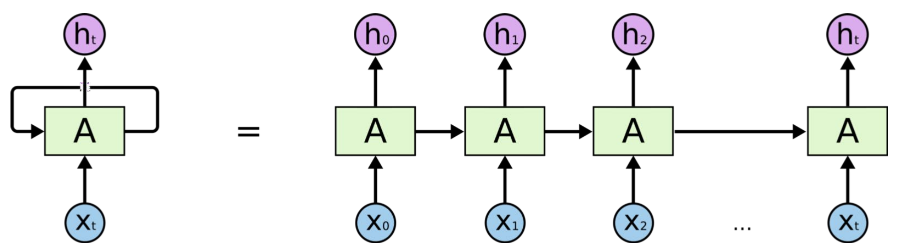
- Deep learning for sequential data
  - 3-dimensional convolution neural networks



3D convolution

How to Model Sequential Data?

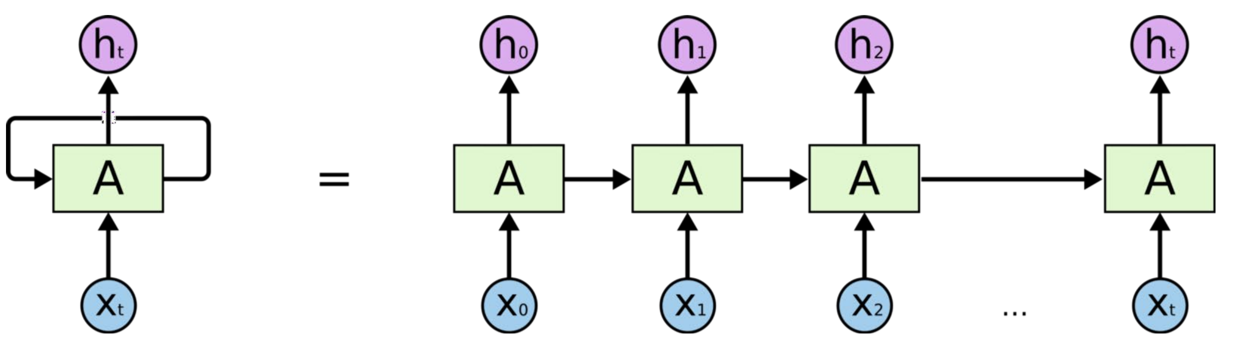
- Deep learning for sequential data
  - Recurrent neural networks (RNN)



RNN

Recurrent Neural Networks

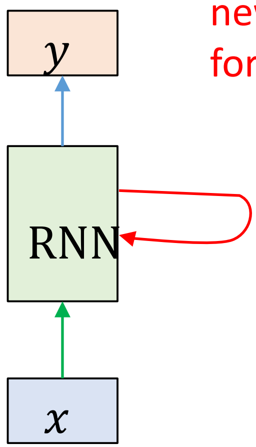
- Parameter sharing + unrolling
  - Keeps the number of parameters fixed
  - Allows sequential data with varying lengths
- Memory ability
  - Capture and preserve information which has been extracted



Recurrence Formula

- Same function and parameters used at every time step:

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



new state for time t

output vector at time t

function with parameters W

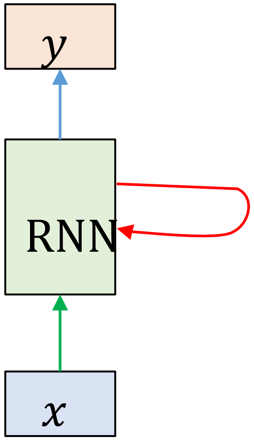
state at time t-1

input vector at time t

Recurrence Formula

- Same function and parameters used at every time step:

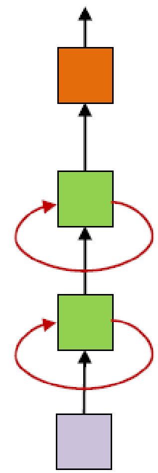
$$h_t, y_t = f_W(h_{t-1}, x_t)$$

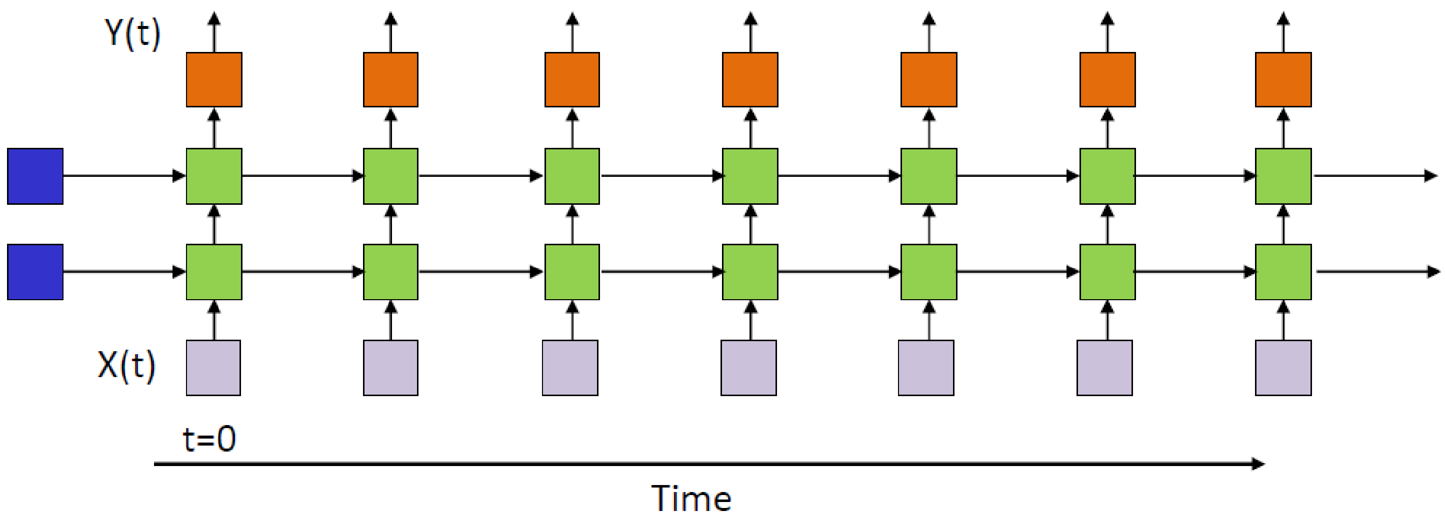


$$\downarrow$$

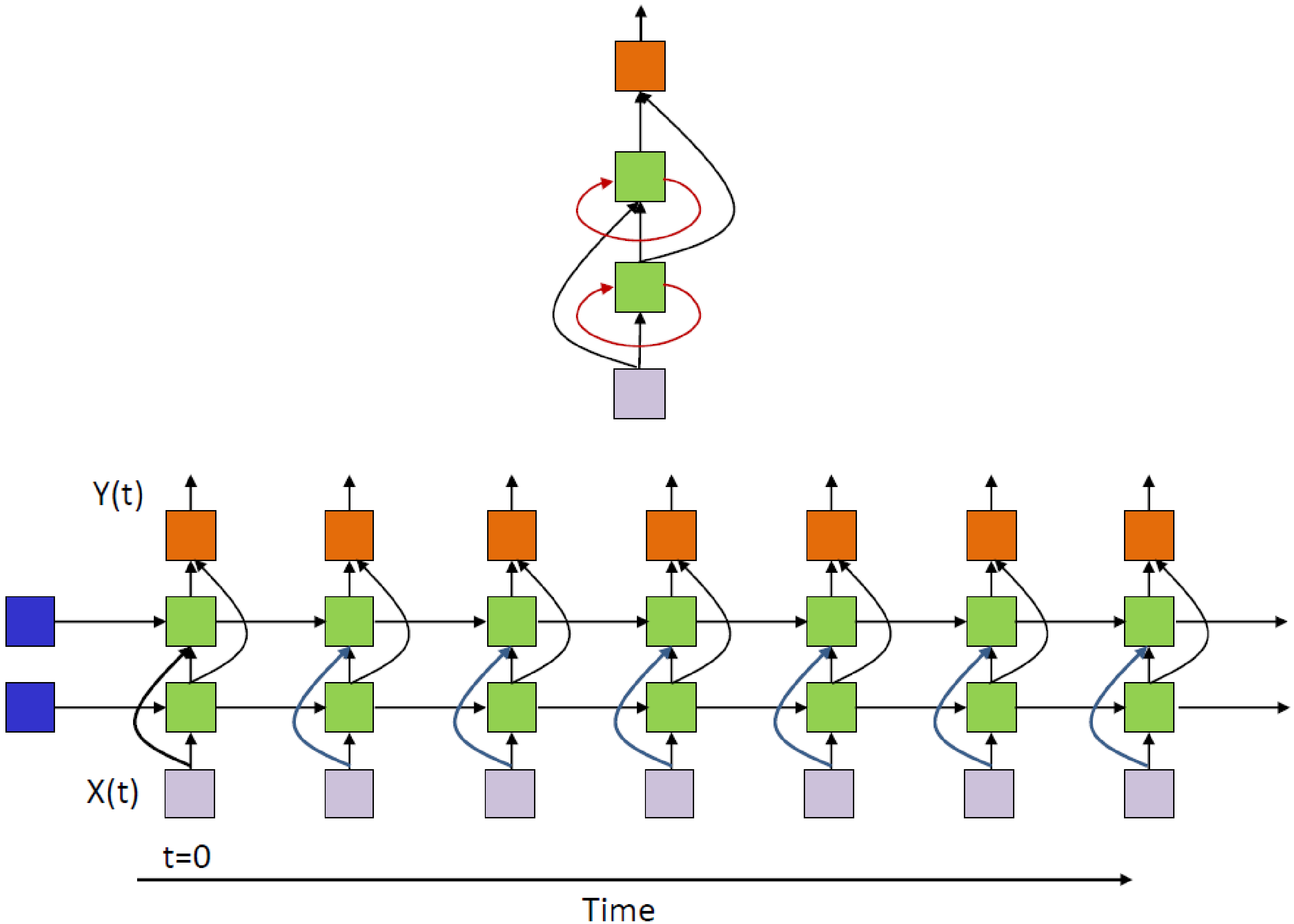
$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$

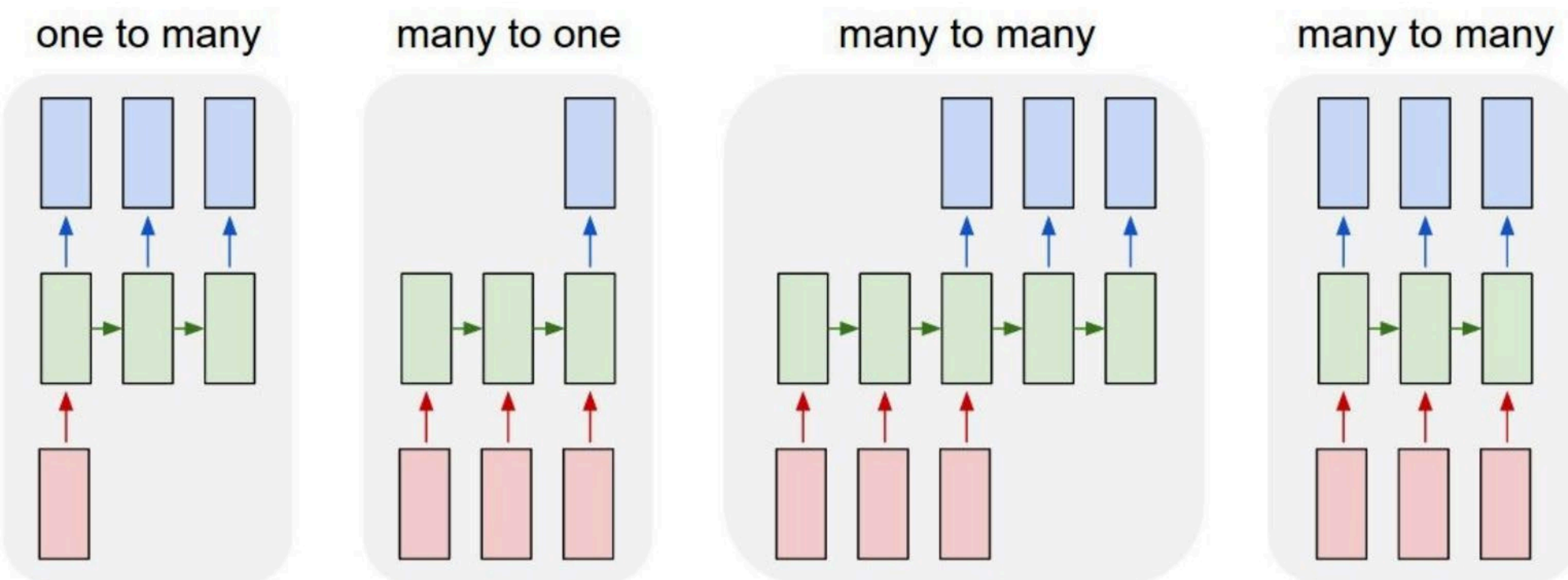
Multiple Recurrent Layers





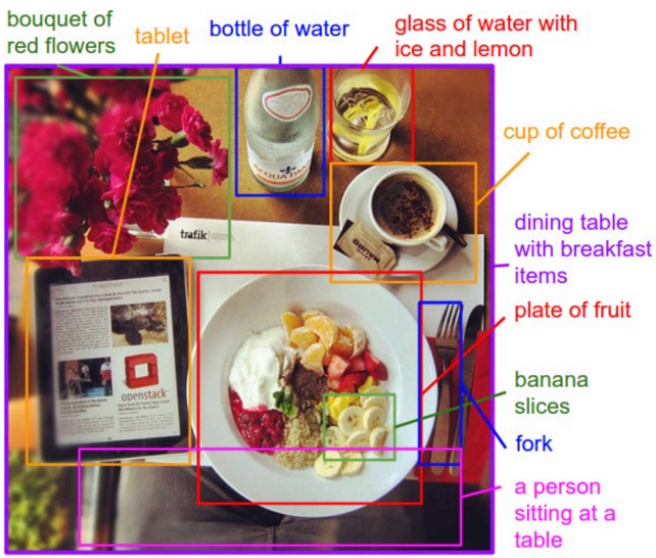
Multiple Recurrent Layers





e.g., image caption e.g., action recognition e.g., video prediction e.g., video indexing

Example: Image Captioning



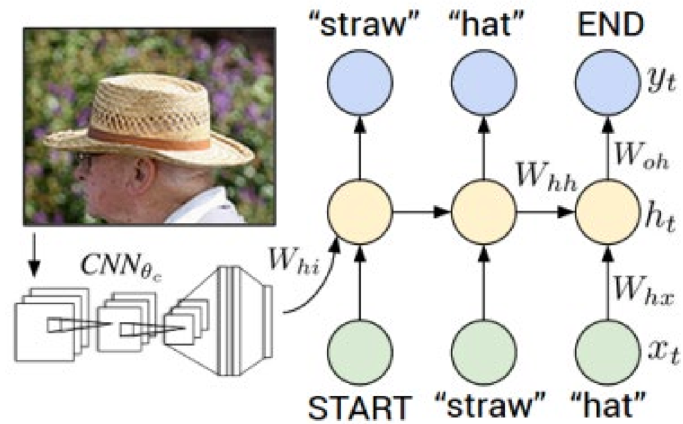


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

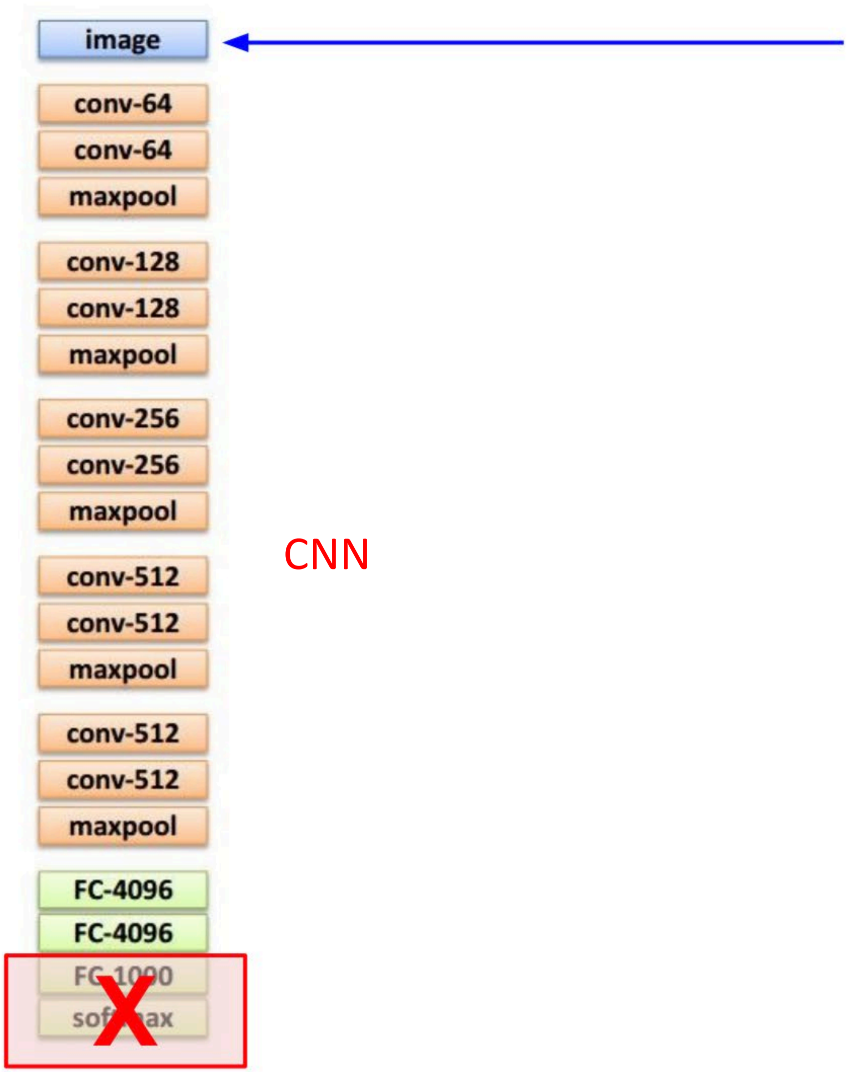
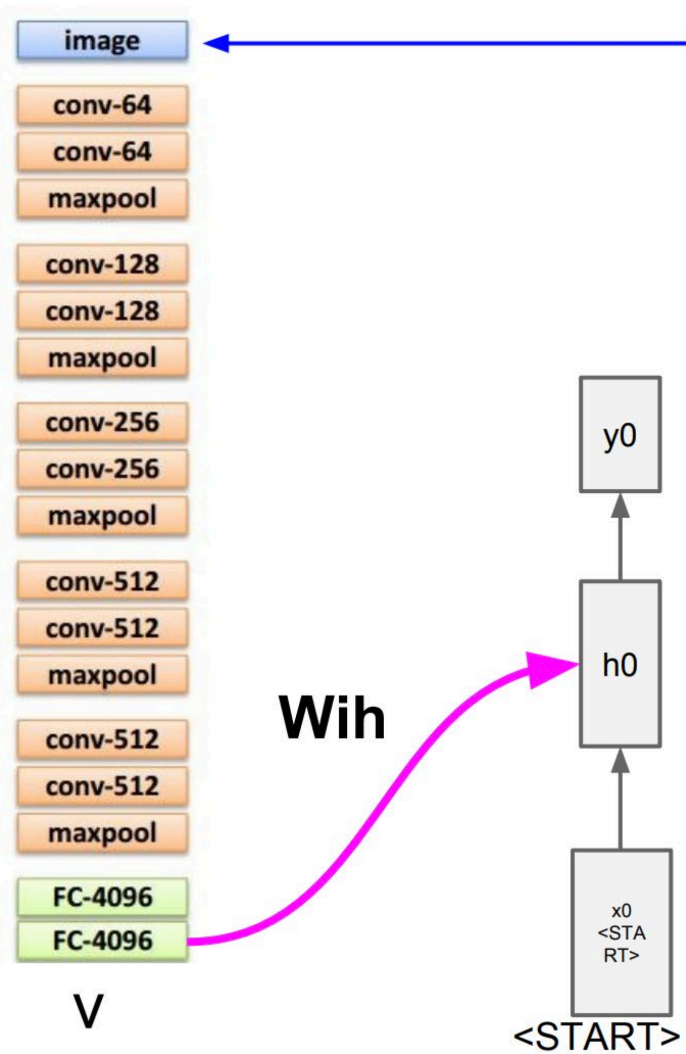
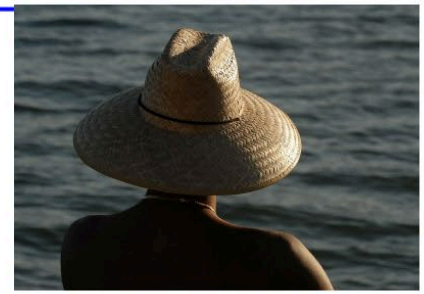


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
CNN

test image





test image

before:

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

now:

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

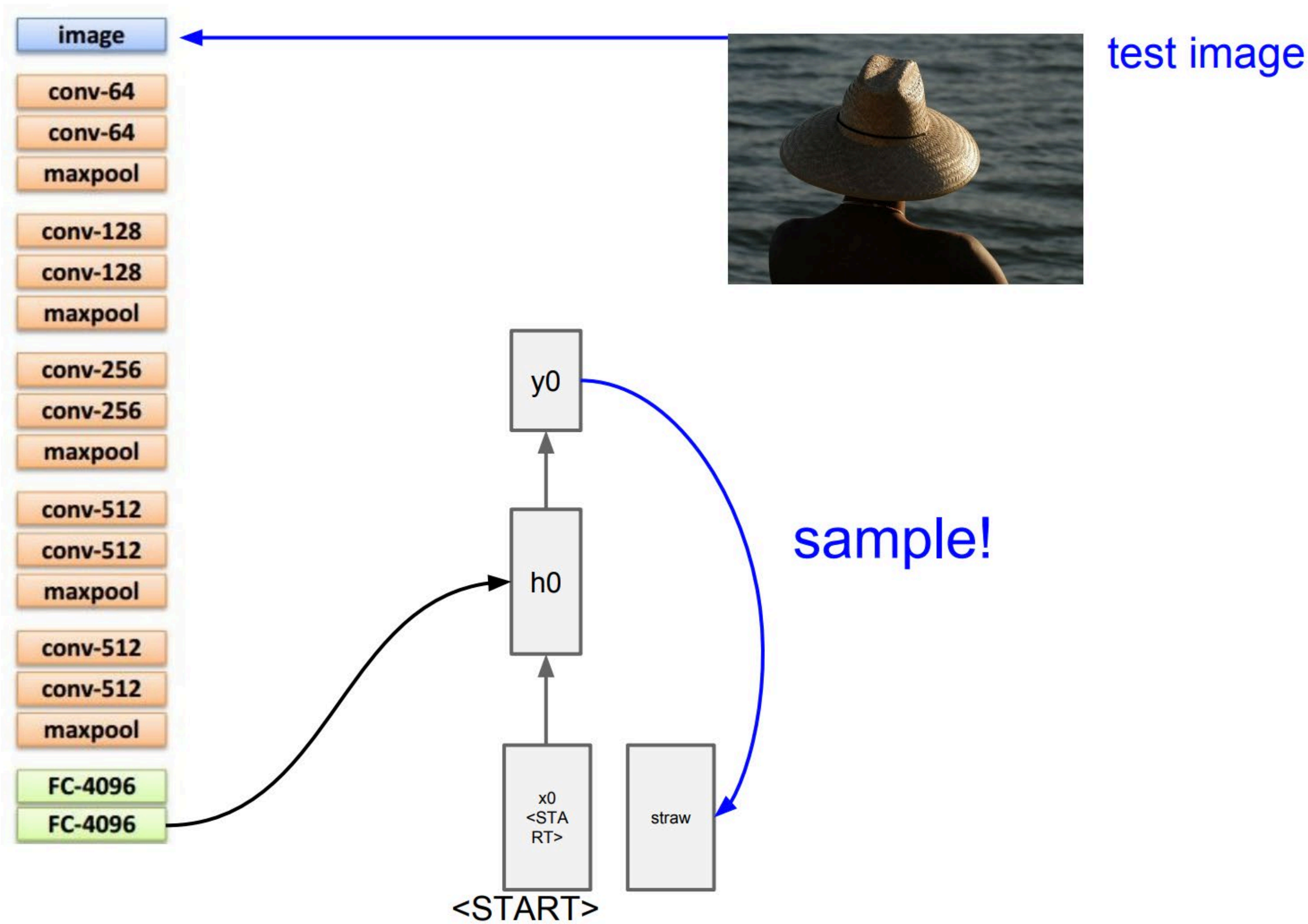


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
h0
sample!
x0 <STA RT>
straw
<START>

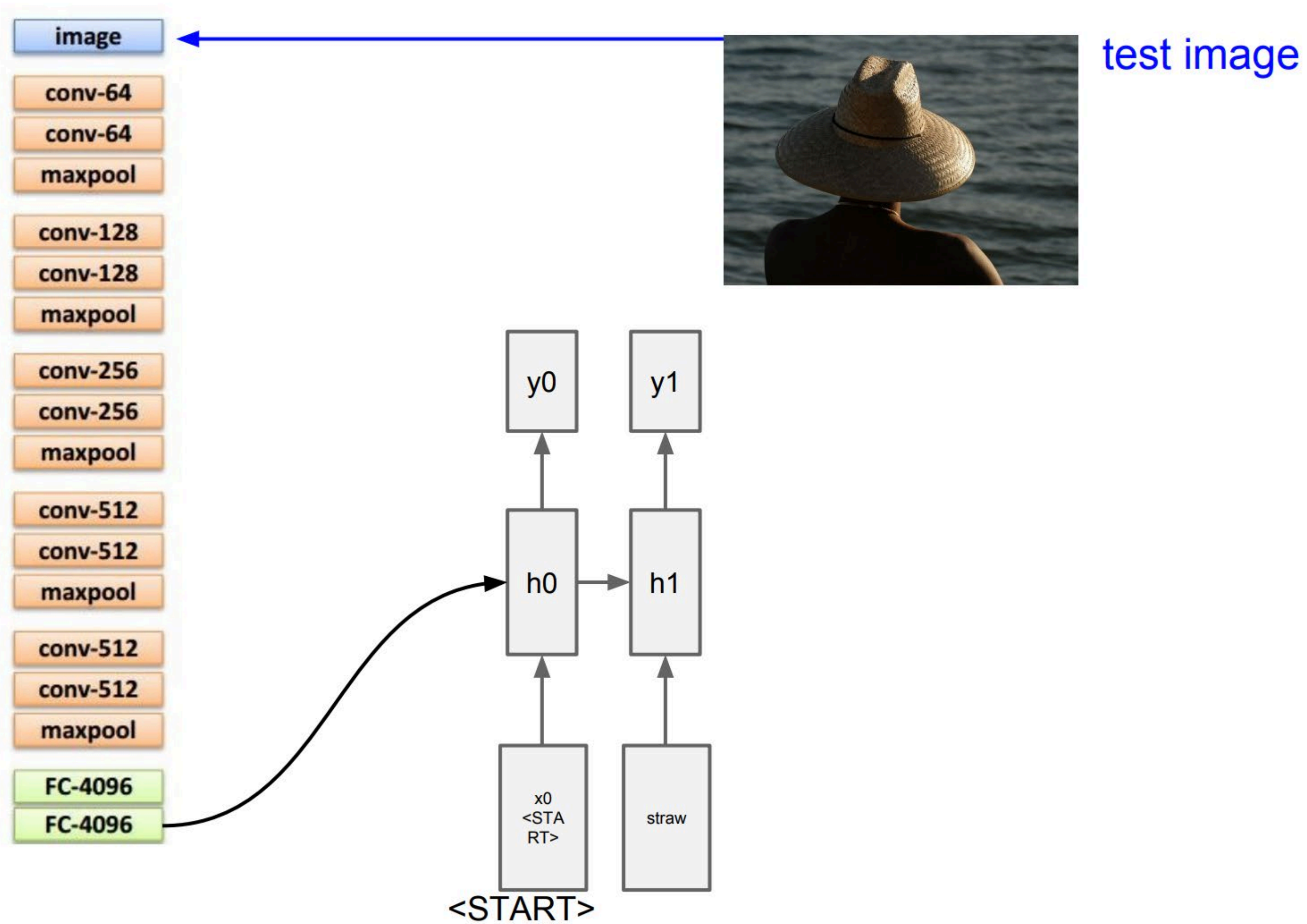


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<STA
RT>
straw
<START>

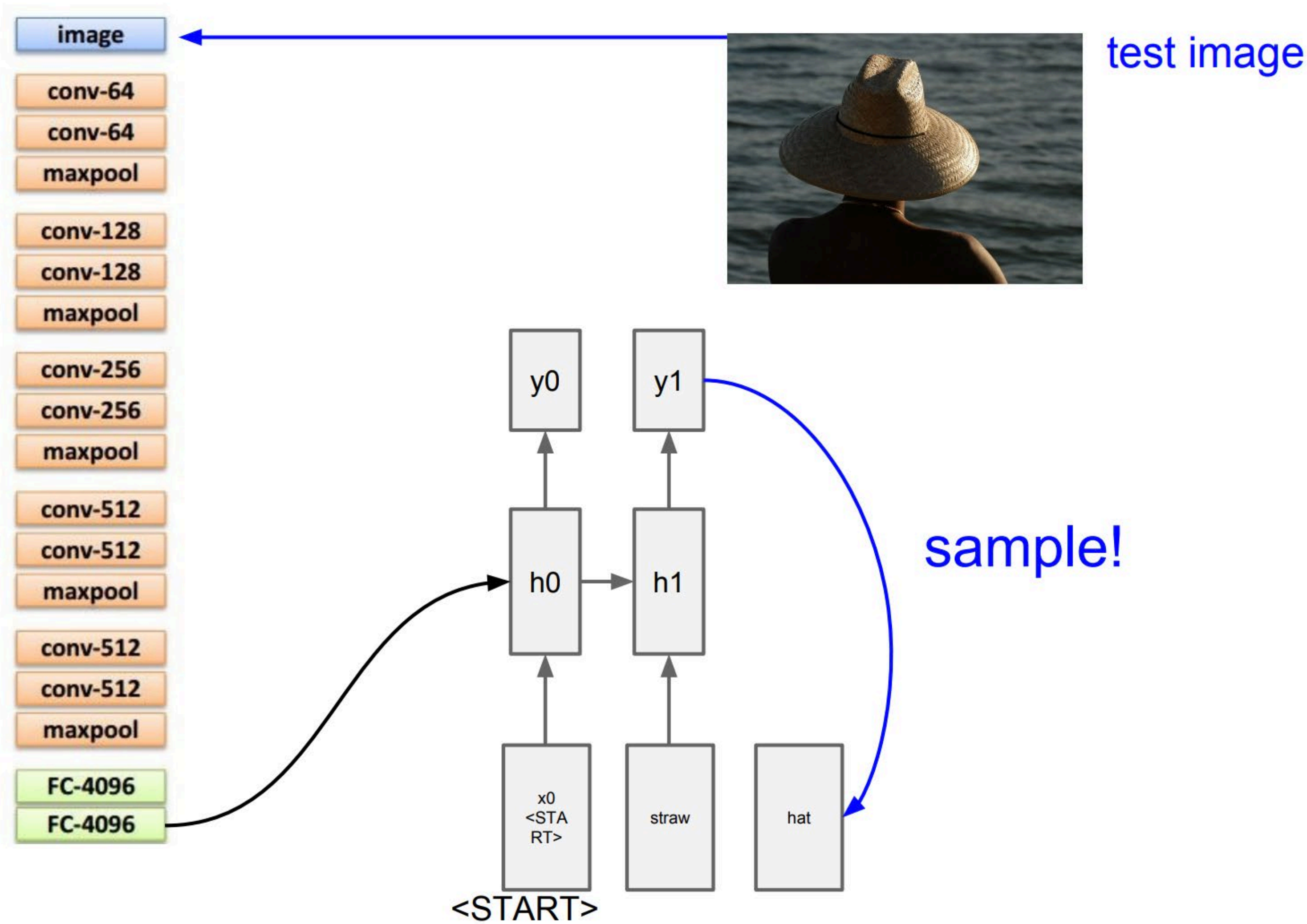


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
sample!
x0
<STA
RT>
straw
hat
<START>

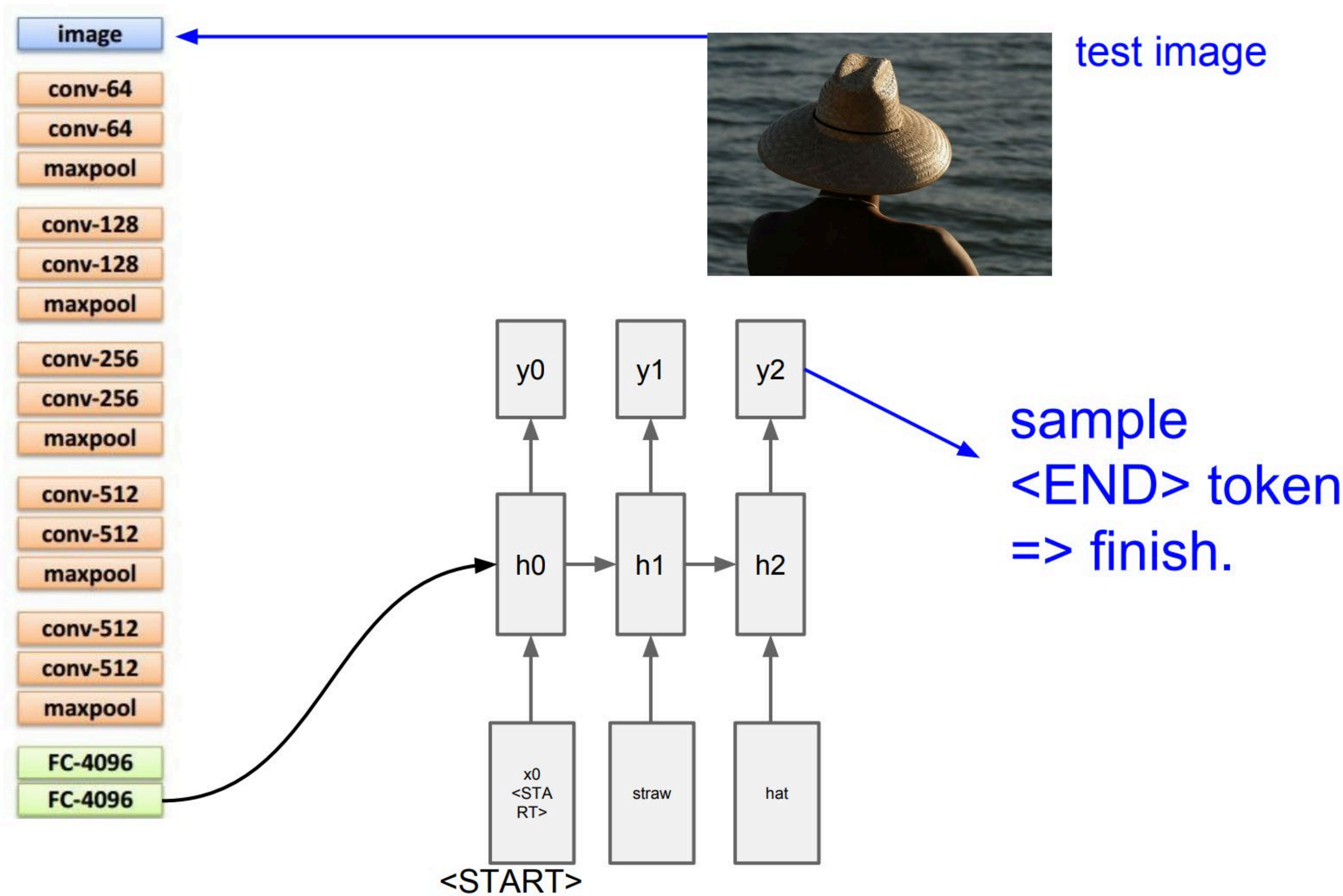
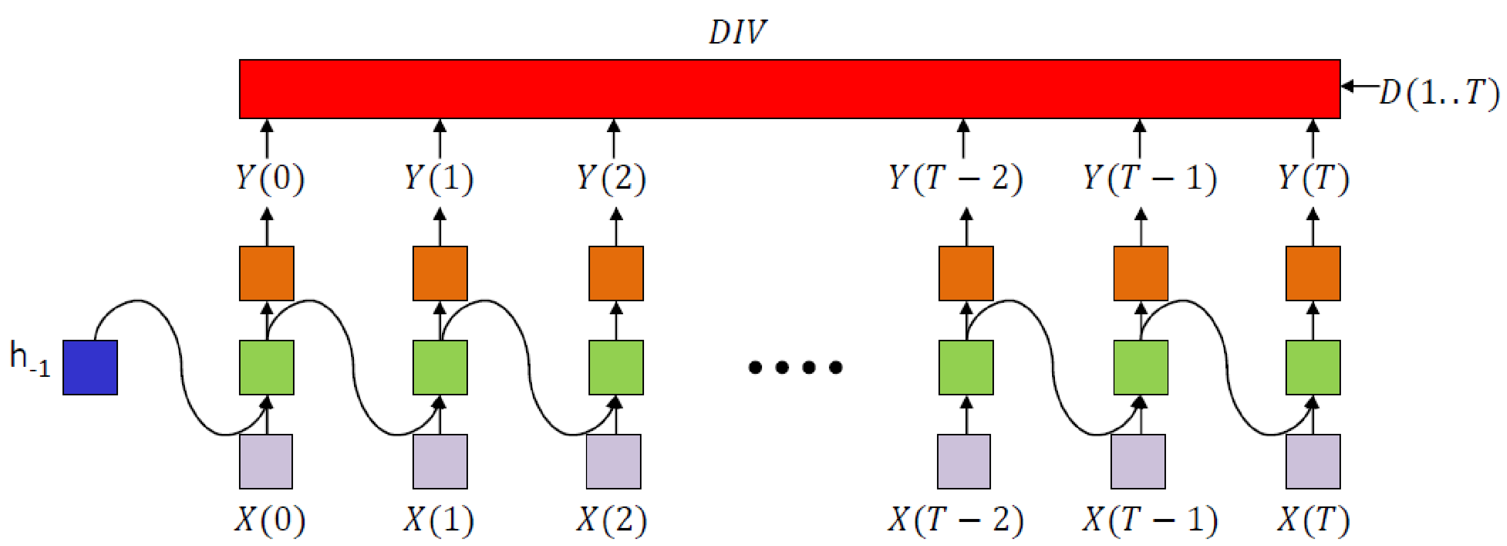


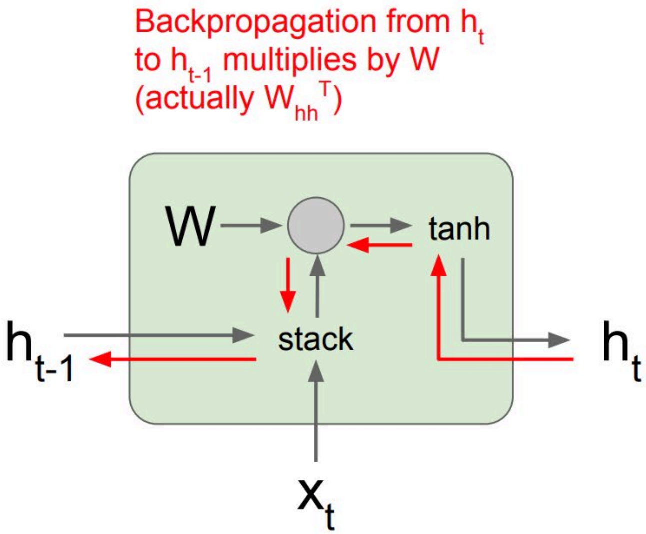
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
sample
<END> token
=> finish.
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>

Training RNNs: Back Propagation Through Time

- Let's focus on one training instance.
- The divergence to be computed is between the sequence of outputs by the network and the desired output sequence.
- Generally, this is not just the sum of the divergences at individual times.

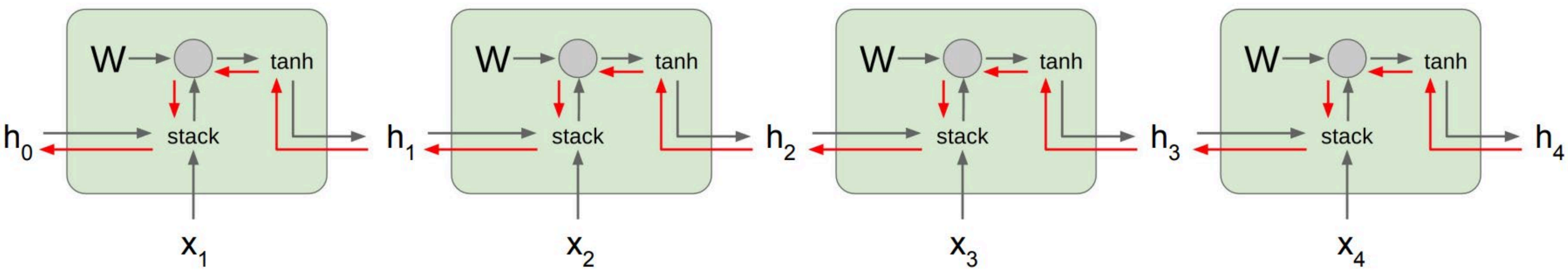


Back Propagation Through Time (BPTT)



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

Back Propagation Through Time (BPTT)

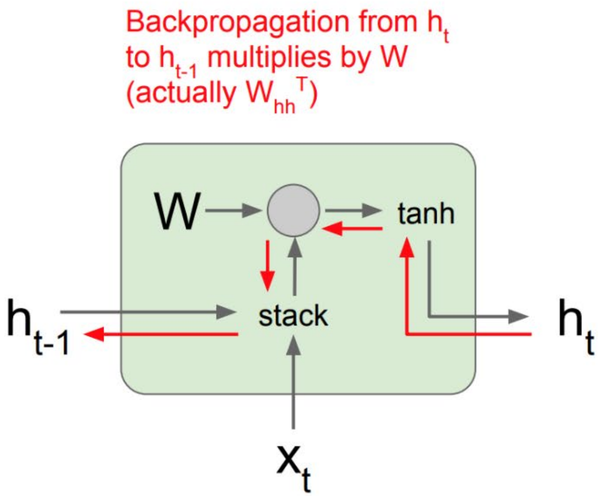


Computing gradient of h_0 involves many factors of W (and repeated tanh)

Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

Gradient Vanishing & Exploding

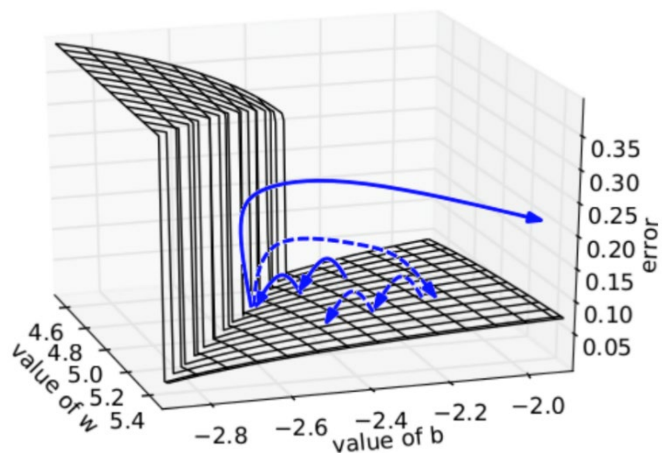
- Computing gradient involves many factors of W
  - Exploding gradients : Largest singular value > 1
  - Vanishing gradients : Largest singular value < 1



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

Solutions…

- Gradients clipping : rescale gradients if too large



standard gradient descent trajectories

gradient clipping to fix problem

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

- How about vanishing gradients?
    - Change RNN architecture!

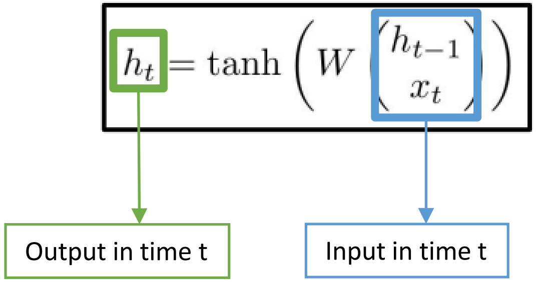
Variants of RNN

- Long Short-term Memory (LSTM) [*Hochreiter et al., 1997*]
  - Additional memory cell
  - Input/Forget/Output Gates
  - Handle gradient vanishing
  - Learn long-term dependencies

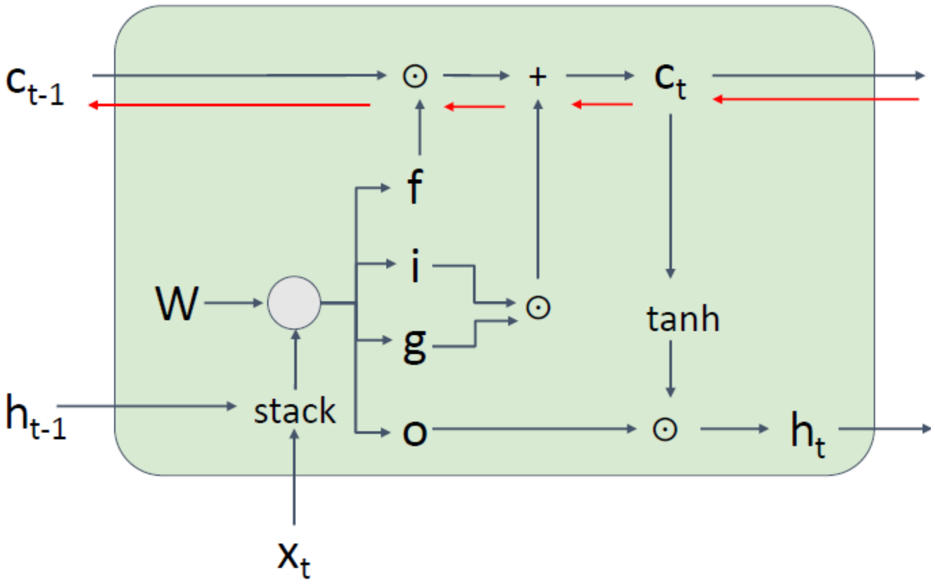
- Gated Recurrent Unit (GRU) [*Cho et al., EMNLP 2014*]
  - Similar to LSTM
  - No additional memory cell
  - Reset / Update Gates
  - Fewer parameters than LSTM
  - Comparable performance to LSTM [*Chung et al., NIPS Workshop 2014*]

Vanilla RNN vs. LSTM

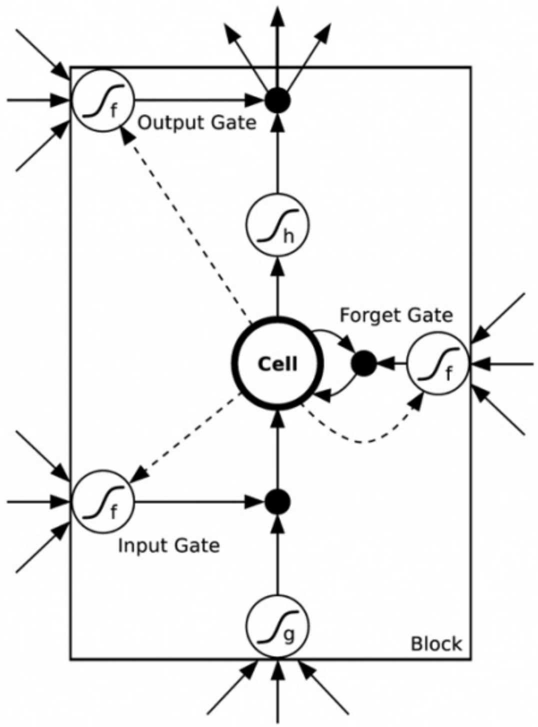
Vanilla RNN



LSTM



Long Short-Term Memory (LSTM)



Forget gate	$f = \sigma(W_f [h_{t-1}, x_t] + b_f)$
Input gate	$i = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
Input activation	$\tilde{h} = tanh(W_h [h_{t-1}, x_t] + b_h)$
Output gate	$o = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
Cell state	$c_t = f \odot c_{t-1} + i \odot \tilde{h}$
Hidden state	$h_t = o \odot tanh(c_t)$

Memory Cell

Output in time t

Input in time t

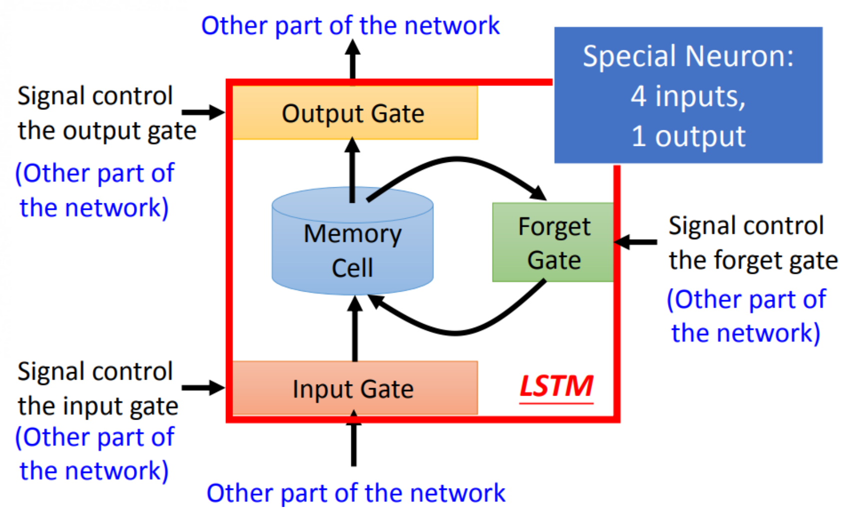
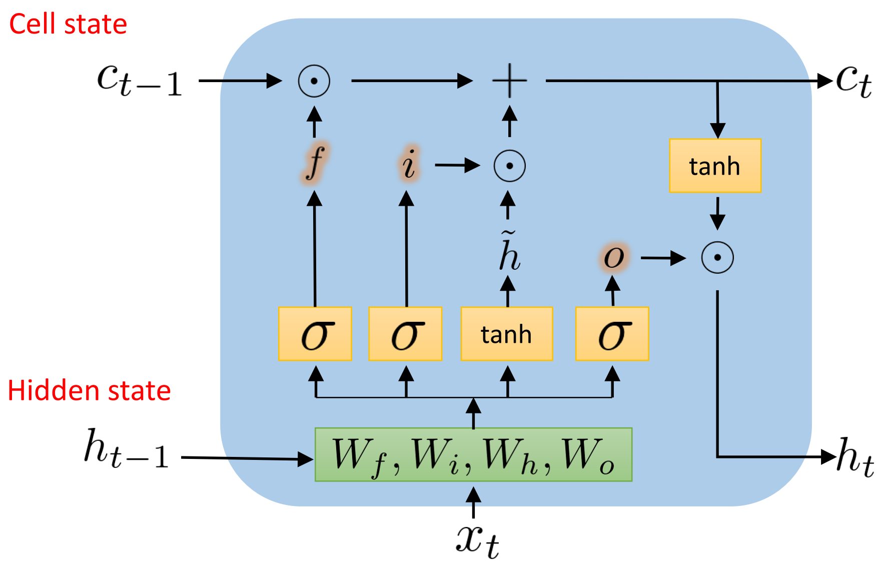


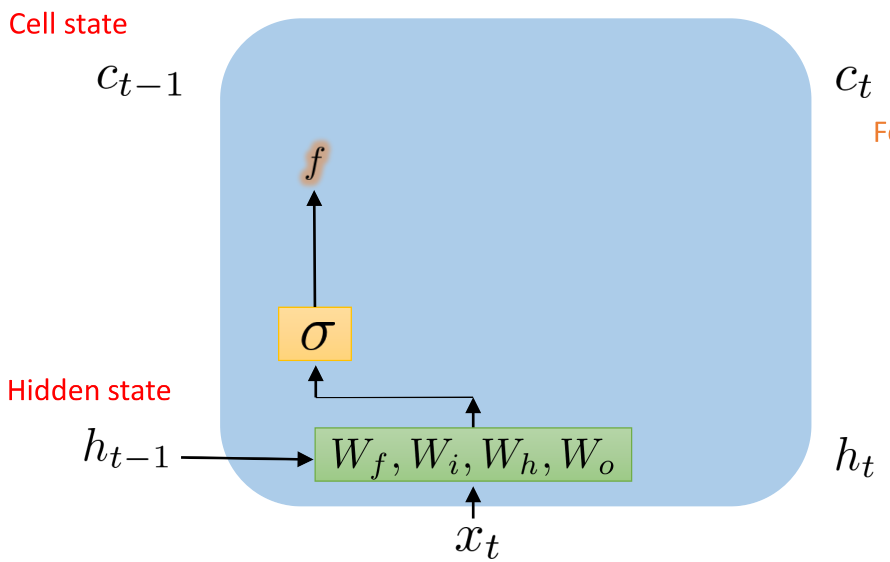
Image Credit: Hung-Yi Lee

Long Short-Term Memory (LSTM)



Forget gate	$f = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$
Input gate	$i = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$
Input activation	$\tilde{h} = tanh (W_h \cdot [h_{t-1}, x_t] + b_h)$
Output gate	$o = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o)$
Cell state	$c_t = f \odot c_{t-1} + i \odot \tilde{h}$
Hidden state	$h_t = o \odot tanh(c_t)$

Long Short-Term Memory (LSTM)

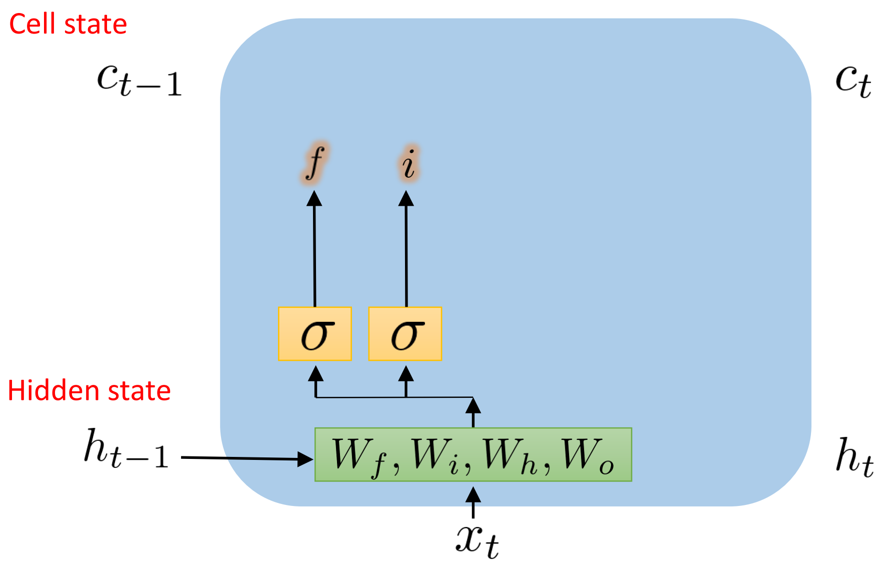


Calculate forget gate f:
whether to erase cell

Forget gate

$$
\begin{aligned}
f &= \sigma\ (W_f \cdot [h_{t-1}, x_t] + b_f) \\
i &= \sigma\ (W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{h} &= tanh\ (W_h \cdot [h_{t-1}, x_t] + b_h) \\
o &= \sigma\ (W_o \cdot [h_{t-1}, x_t] + b_o) \\
c_t &= f \odot c_{t-1} + i \odot \tilde{h} \\
h_t &= o \odot tanh(c_t)
\end{aligned}
$$

Long Short-Term Memory (LSTM)

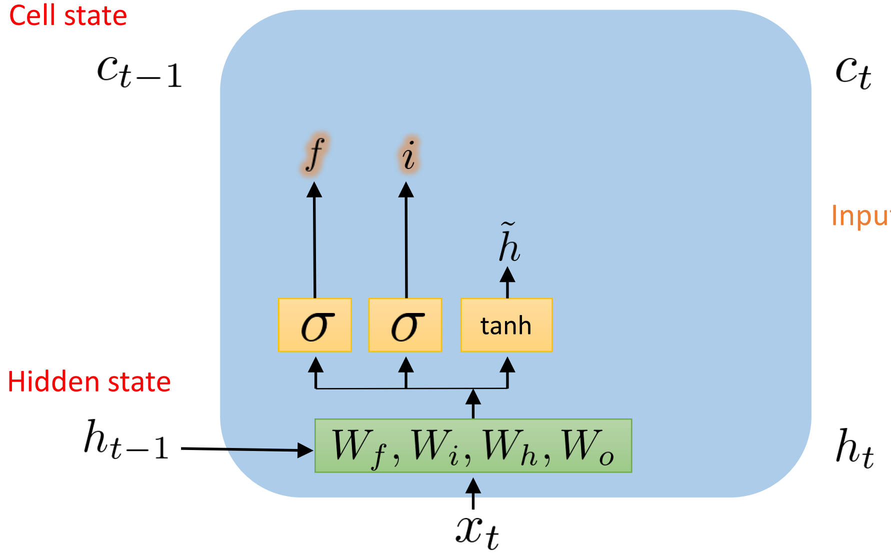


Calculate input gate i:
whether to write to cell

Input gate

$$
\begin{aligned}
f &= \sigma \quad (W_f \cdot [h_{t-1}, x_t] + b_f) \\
i &= \sigma \quad (W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{h} &= tanh \quad (W_h \cdot [h_{t-1}, x_t] + b_h) \\
o &= \sigma \quad (W_o \cdot [h_{t-1}, x_t] + b_o) \\
c_t &= f \odot c_{t-1} + i \odot \tilde{h} \\
h_t &= o \odot tanh(c_t)
\end{aligned}
$$

Long Short-Term Memory (LSTM)

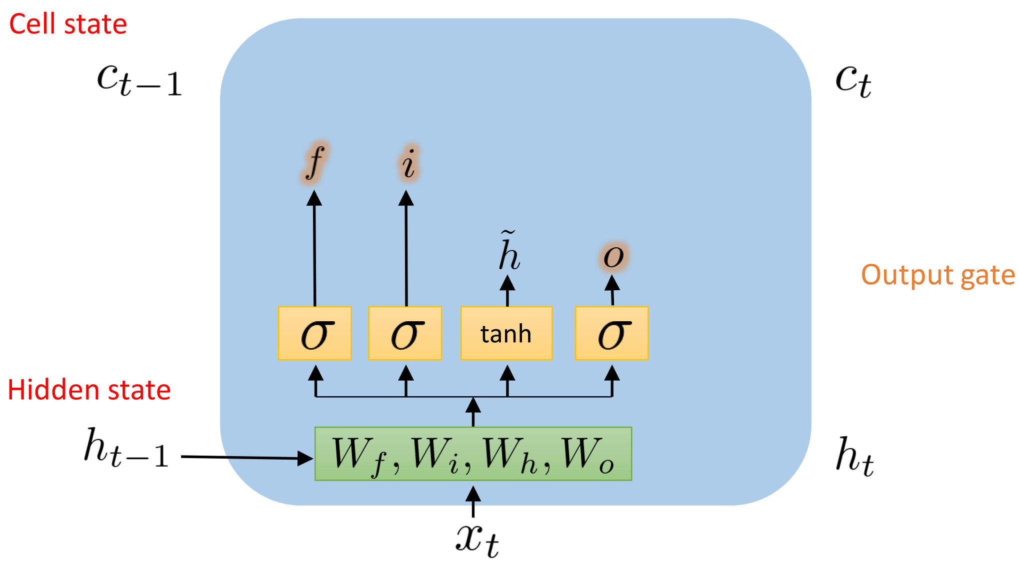


Calculate input activation:
how much to write to cell

Input activation

$$\begin{aligned} f &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{h} &= tanh(W_h \cdot [h_{t-1}, x_t] + b_h) \\ o &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ c_t &= f \odot c_{t-1} + i \odot \tilde{h} \\ h_t &= o \odot tanh(c_t) \end{aligned}$$

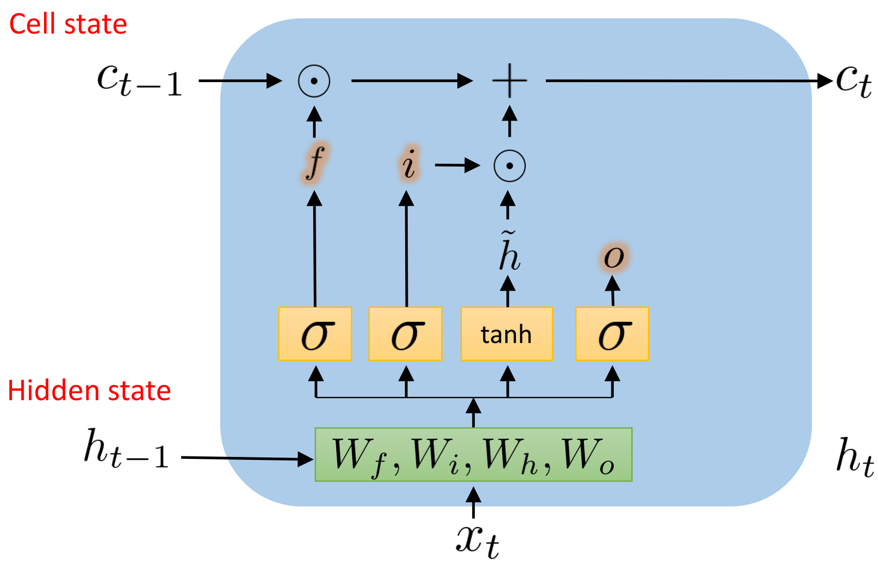
Long Short-Term Memory (LSTM)



Calculate output gate o:
how much to reveal cell

$$f = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{h} = tanh(W_h \cdot [h_{t-1}, x_t] + b_h)$$
$$o = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$c_t = f \odot c_{t-1} + i \odot \tilde{h}$$
$$h_t = o \odot tanh(c_t)$$

Long Short-Term Memory (LSTM)



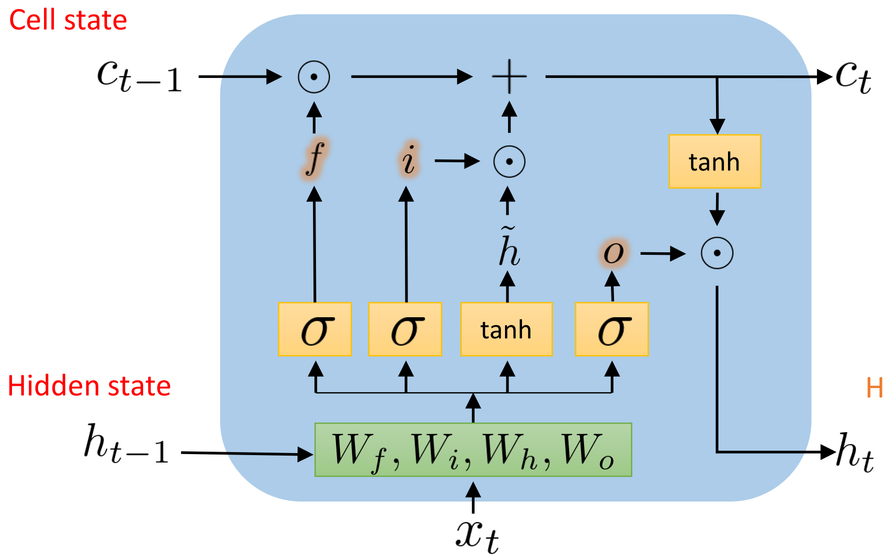
Update memory cell

$$f = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{h} = tanh (W_h \cdot [h_{t-1}, x_t] + b_h)$$
$$o = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell state

$$c_t = f \odot c_{t-1} + i \odot \tilde{h}$$
$$h_t = o \odot tanh(c_t)$$

Long Short-Term Memory (LSTM)

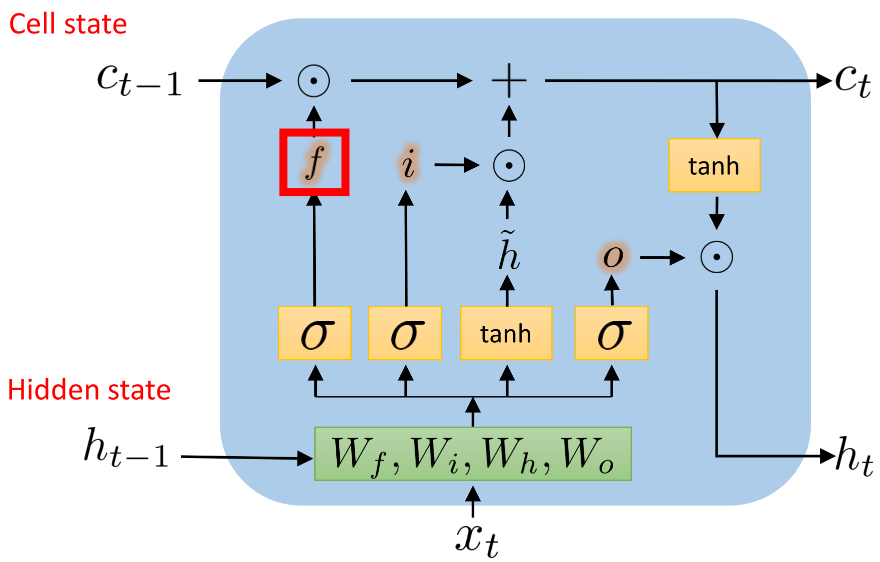


Calculate output h_t

$$
\begin{aligned}
f &= \sigma\ (W_f \cdot [h_{t-1}, x_t] + b_f) \\
i &= \sigma\ (W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{h} &= tanh\ (W_h \cdot [h_{t-1}, x_t] + b_h) \\
o &= \sigma\ (W_o \cdot [h_{t-1}, x_t] + b_o) \\
c_t &= f \odot c_{t-1} + i \odot \tilde{h} \\
h_t &= o \odot tanh(c_t)
\end{aligned}
$$

Hidden state

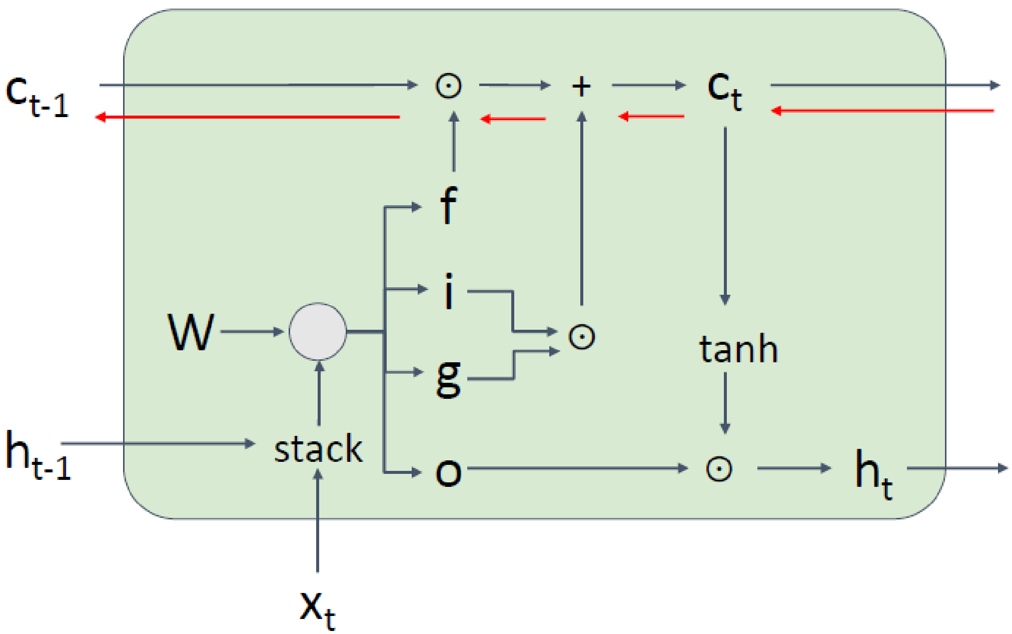
Long Short-Term Memory (LSTM)



Prevent gradient vanishing
if forget gate f is open (>0).

$$
\begin{aligned}
f &= \sigma \quad (W_f \cdot [h_{t-1}, x_t] + b_f) \\
i &= \sigma \quad (W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{h} &= tanh \quad (W_h \cdot [h_{t-1}, x_t] + b_h) \\
o &= \sigma \quad (W_o \cdot [h_{t-1}, x_t] + b_o) \\
c_t &= \boxed{f} \odot c_{t-1} + i \odot \tilde{h} \\
h_t &= o \odot tanh(c_t)
\end{aligned}
$$

LSTM: Gradient Flow



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

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h \right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

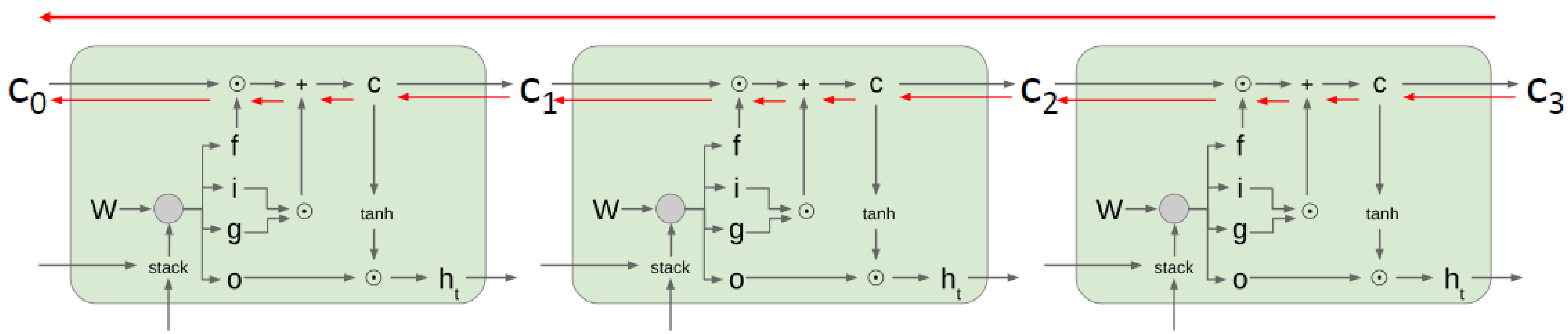
$$h_t = o_t \odot \tanh(c_t)$$

Remarks:

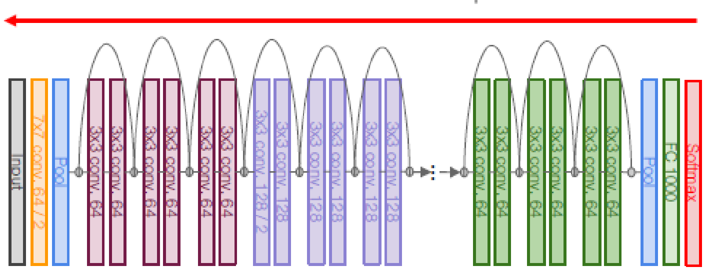
- Linear relationship between c_t and c_{t-1} instead of multiplication relationship of h_t and h_{t-1} in vanilla RNN

LSTM: Gradient Flow

Uninterrupted gradient flow!



Similar to ResNet!



Remarks:

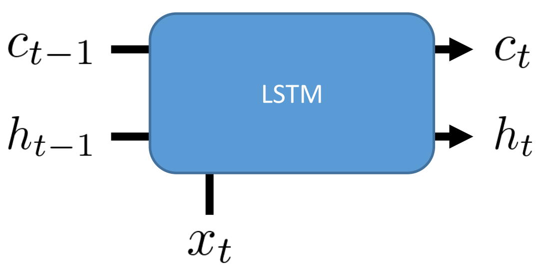
- Linear relationship between c_t and c_{t-1} instead of multiplication relationship of h_t and h_{t-1} in vanilla RNN
- Forget gate *f* provides shortcut connection across time steps

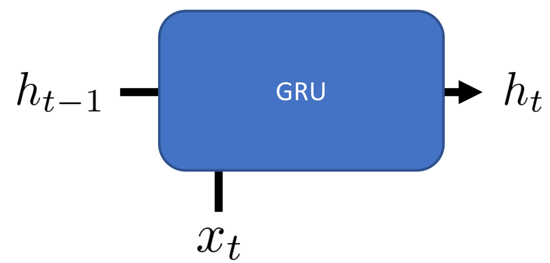
Variants of RNN

- Long Short-term Memory (LSTM) [*Hochreiter et al., 1997*]
  - Additional memory cell
  - Input / Forget / Output Gates
  - Handle gradient vanishing
  - Learn long-term dependencies

- Gated Recurrent Unit (GRU) [*Cho et al., EMNLP 2014*]
  - Similar to LSTM
  - No additional memory cell
  - Reset/Update Gates
  - Fewer parameters than LSTM
  - Comparable performance to LSTM [*Chung et al., NIPS Workshop 2014*]

LSTM vs. GRU





Forget gate $f = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

Input gate $i = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

Input activation $\tilde{h} = tanh(W_h \cdot [h_{t-1}, x_t] + b_h)$

Output gate $o = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

Cell state $c_t = f \odot c_{t-1} + i \odot \tilde{h}$

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

Reset gate $r = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$

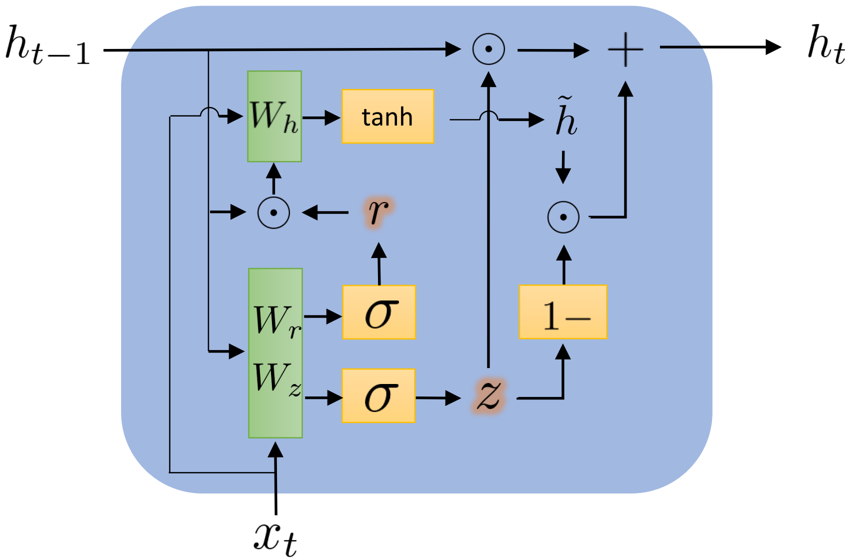
Update gate $z = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$

Input activation $\tilde{h} = tanh(W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)$

Hidden state $h_t = z \odot h_{t-1} + (1 - z) \odot \tilde{h}$

Gated Recurrent Unit (GRU)

Hidden state

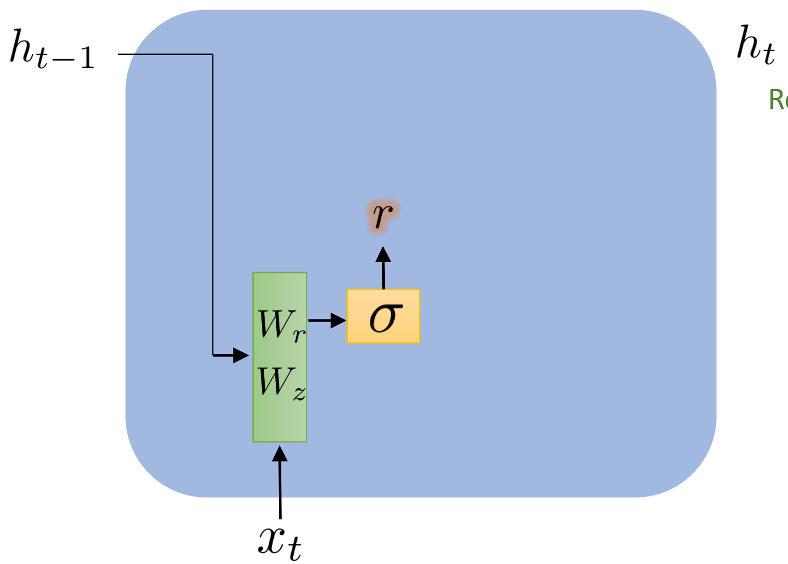


$$
\begin{aligned}
r &= \sigma \quad (W_r \cdot [h_{t-1}, x_t] + b_r) \\
z &= \sigma \quad (W_z \cdot [h_{t-1}, x_t] + b_z) \\
\tilde{h} &= tanh \quad (W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)
\end{aligned}
$$

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

Gated Recurrent Unit (GRU)

Hidden state



Calculate reset gate r

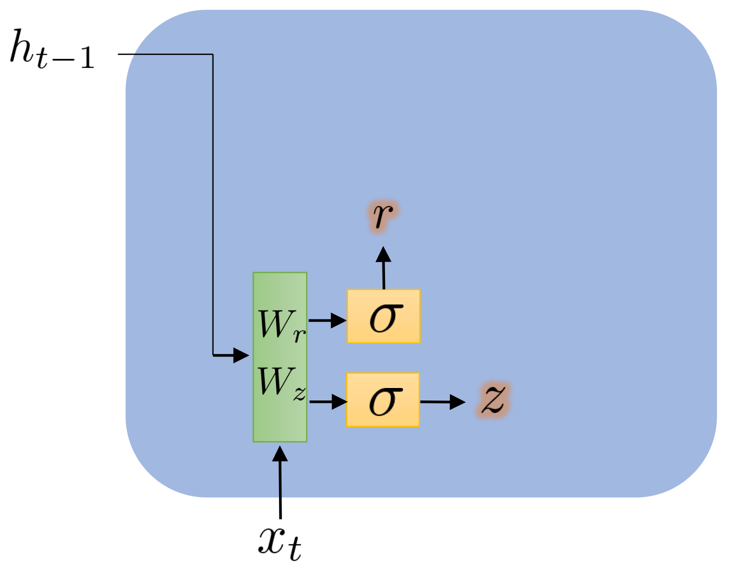
Reset gate

$$
\begin{aligned}
r &= \sigma \quad (W_r \cdot [h_{t-1}, x_t] + b_r) \\
z &= \sigma \quad (W_z \cdot [h_{t-1}, x_t] + b_z) \\
\tilde{h} &= tanh \quad (W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)
\end{aligned}
$$

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

Gated Recurrent Unit (GRU)

Hidden state



Calculate update gate z

Update gate

$$r = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

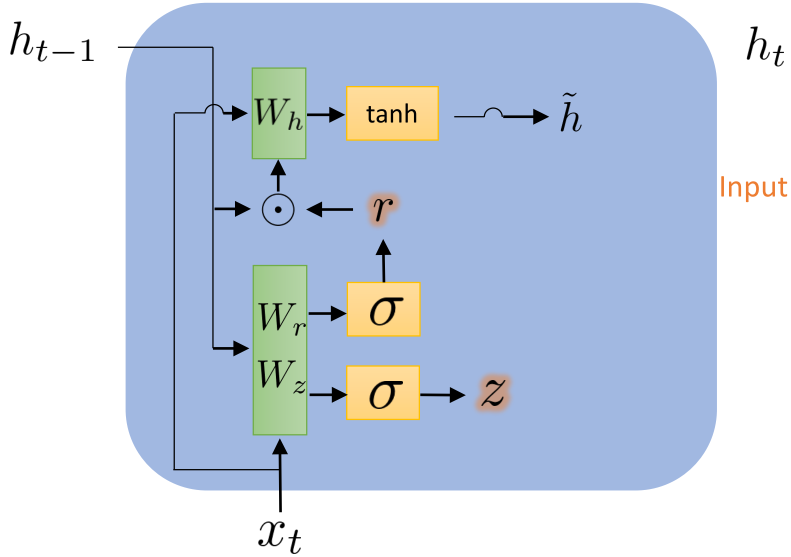
$$z = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$\tilde{h} = tanh(W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)$$

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

Gated Recurrent Unit (GRU)

Hidden state



Calculate input activation

$$r = \sigma (W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$z = \sigma (W_z \cdot [h_{t-1}, x_t] + b_z)$$

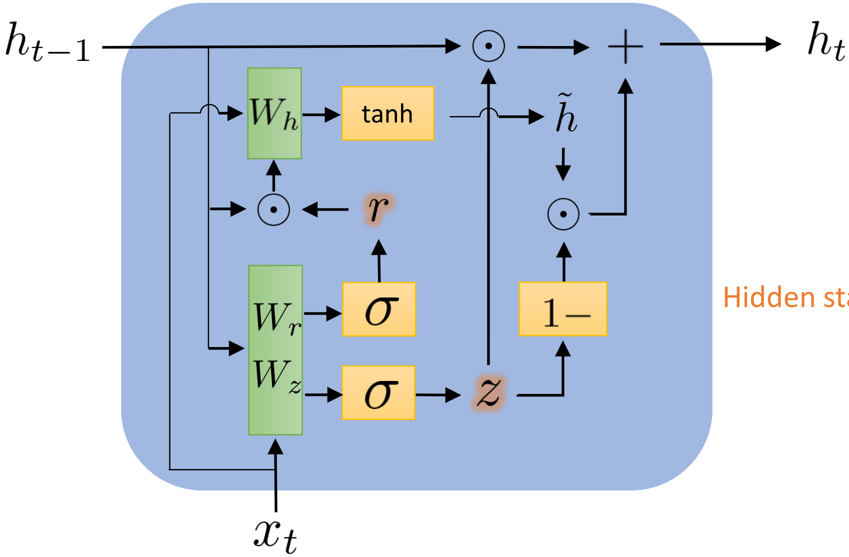
Input activation

$$\tilde{h} = tanh (W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)$$

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

Gated Recurrent Unit (GRU)

Hidden state



Calculate output

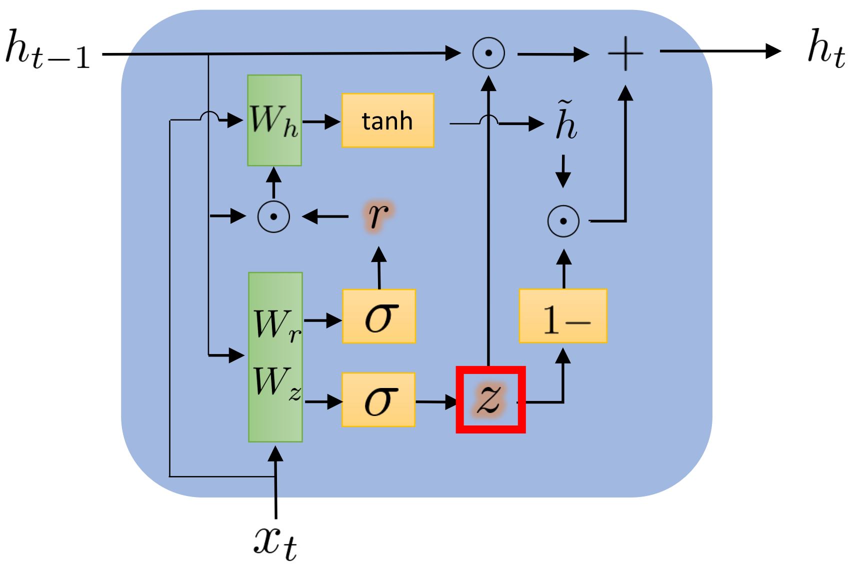
$$
\begin{aligned}
r &= \sigma \quad (W_r \cdot [h_{t-1}, x_t] + b_r) \\
z &= \sigma \quad (W_z \cdot [h_{t-1}, x_t] + b_z) \\
\tilde{h} &= tanh \quad (W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)
\end{aligned}
$$

Hidden state

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

Gated Recurrent Unit (GRU)

Hidden state



Prevent gradient vanishing if update gate z is open!

$$r = \sigma \quad (W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$z = \sigma \quad (W_z \cdot [h_{t-1}, x_t] + b_z)$$

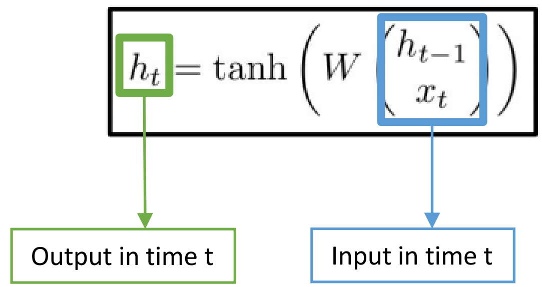
$$\tilde{h} = tanh \quad (W_h \cdot [(r \odot h_{t-1}), x_t] + b_h)$$

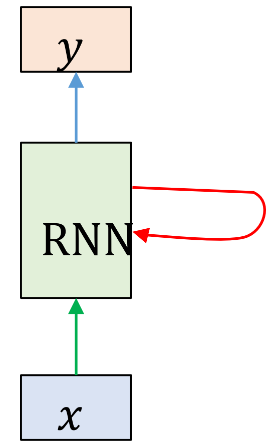
$$h_t = z \odot h_{t-1} + (1 - z) \odot \tilde{h}$$

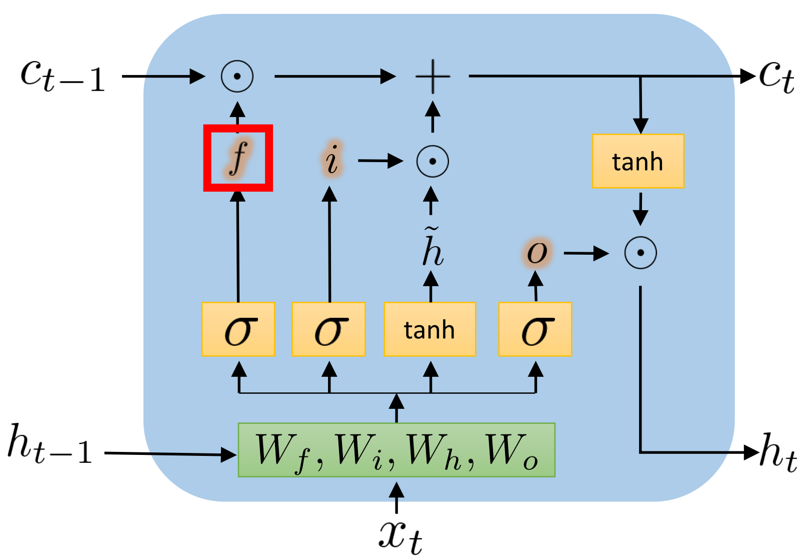
Remarks:

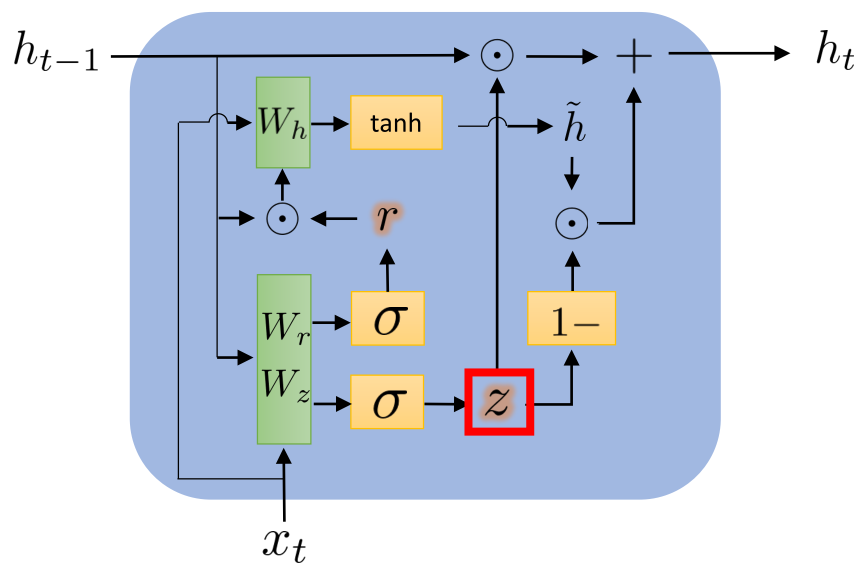
- Update gate *z* provides shortcut connection across time steps
- Linear relationship between h_t and h_{t-1} instead of multiplication relationship of h_t and h_{t-1} in vanilla RNN

Vanilla RNN, LSTM, & GRU







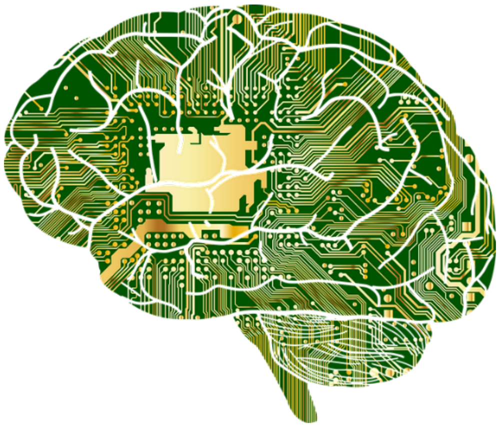


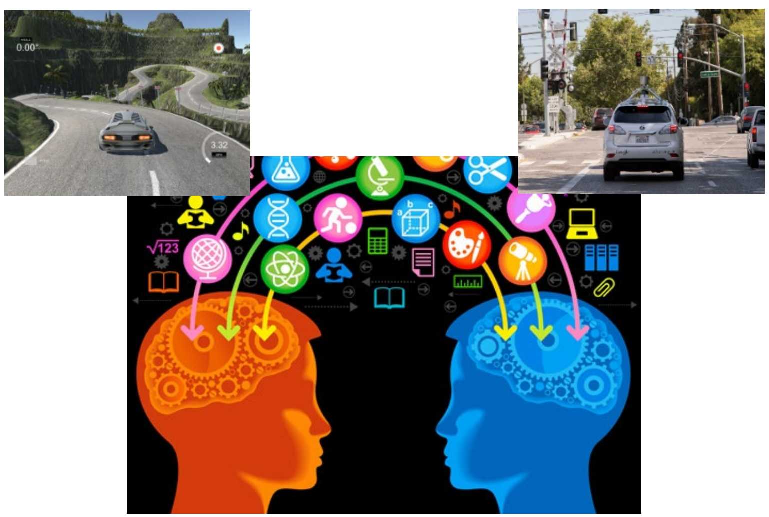
LSTM vs. GRU

	Vanilla RNN	LSTM	GRU
Cell state	X	O	O
Number of Gates	N/A	3	2
Parameters	Least	Most	Fewer
Gradient Vanishing / Exploding	☹	☺	☺

What to Be Covered Today…

- Transfer Learning
  - Visual Classification – Domain Adaptation
  - Visual Synthesis – Style Transfer
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Selected Models for Sequence-to-Sequence Learning
  - Attention in RNN





Sequence-to-Sequence Modeling

- Setting
  - An input sequence $\mathbf{X}_1, \ldots, \mathbf{X}_N$
  - An output sequence $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$
  - Generally $N \neq M$, i.e., no synchrony between **X** and **Y**
- Examples
  - Speech recognition: speech goes in, and a word sequence comes out
  - Machine translation: word sequence goes in, and another comes out
  - Video captioning: video frames goes in, word sequence comes out

深度學習好棒棒!

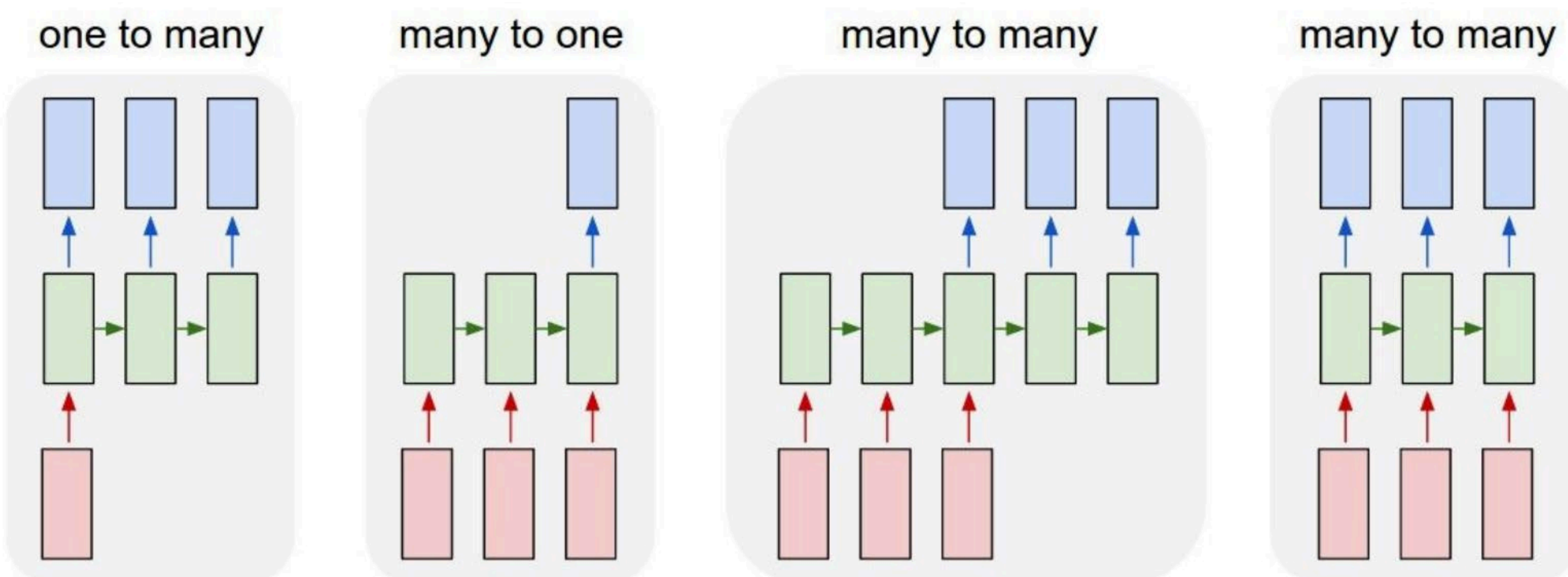
↑

↑

Deep learning rocks!

S-to-S Models with Alignment

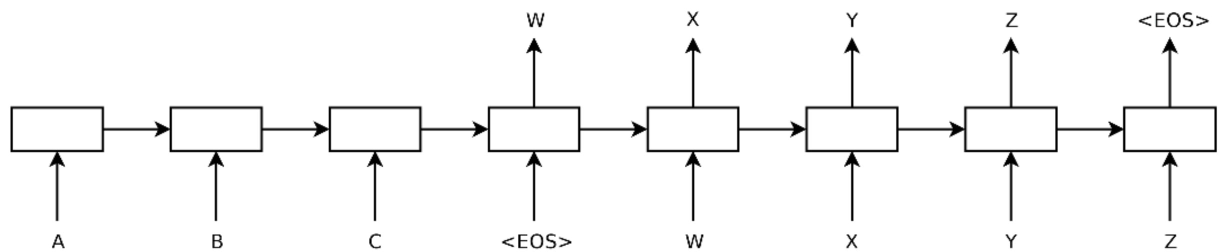
- The input and output sequences happen in the same order
  - The input/output sequences may be asynchronous.
  - E.g., speech recognition or video captioning, in which the input sequence corresponds to the phoneme/caption sequence out.
  - Recall that…

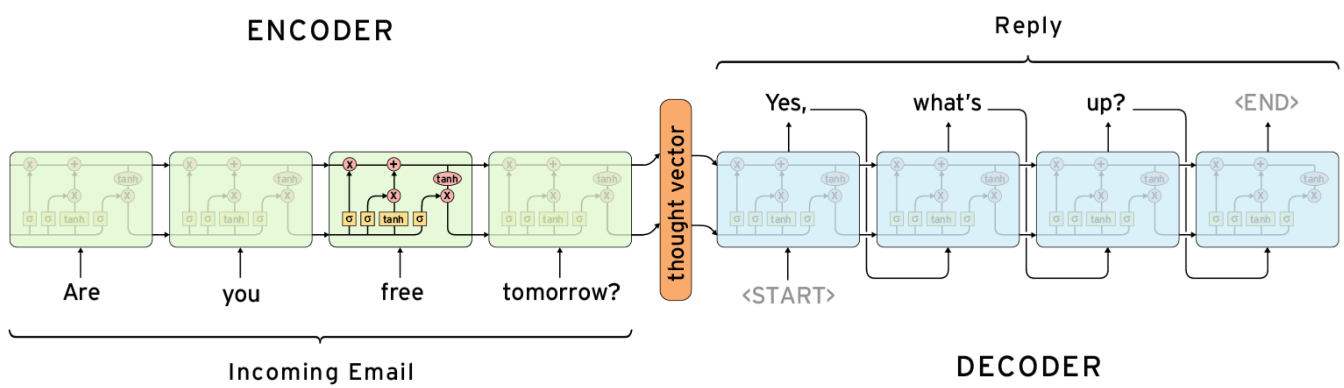


e.g., image caption e.g., action recognition e.g., video prediction e.g., video captioning

Sequence-to-Sequence Modeling (cont'd)

- Original model proposed in NIPS 2014
  - An encoder-decoder model

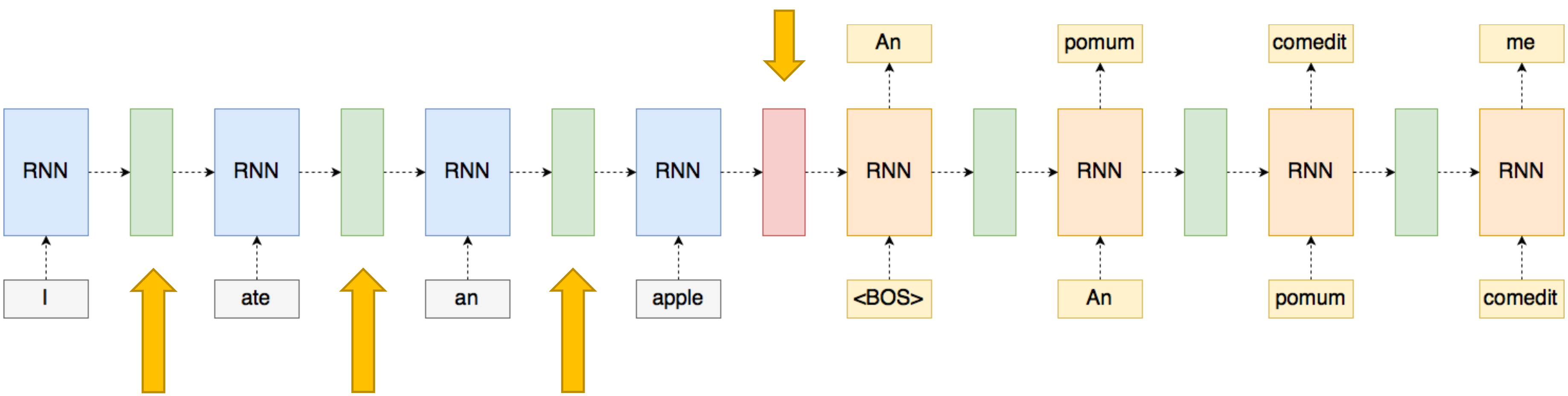




Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

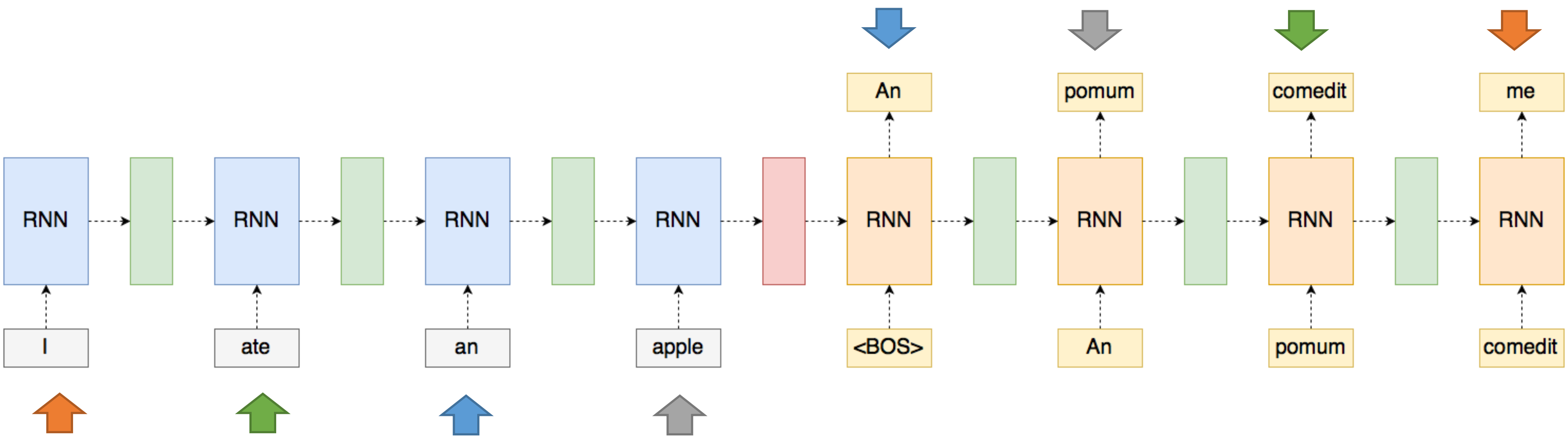
What's the Potential Problem?

- Each hidden state vector extracts/carries information across time steps (some might be diluted downstream).
- However, information of the entire input sequence is embedded into a single hidden state vector.



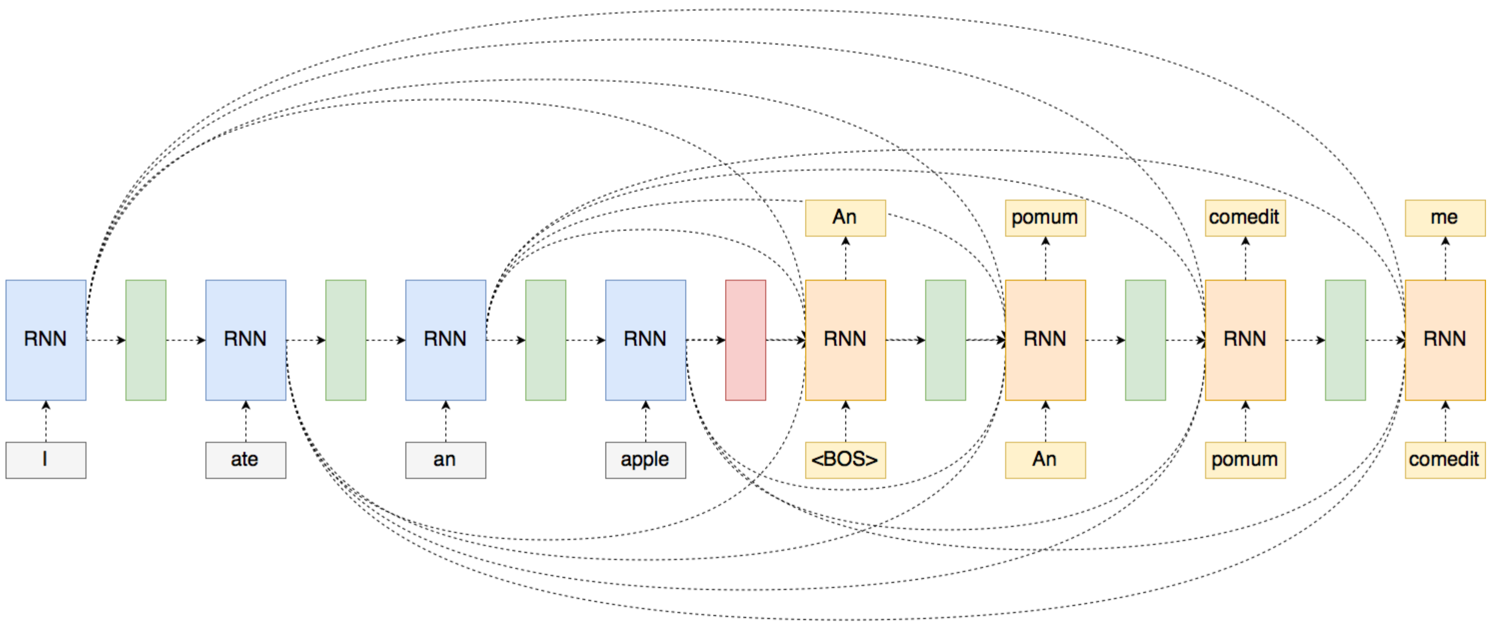
What's the Potential Problem? (cont'd)

- Outputs at different time steps have particular meanings.
- However, synchrony between **input** and **output seqs** is not required.



What’s the Potential Problem? (cont’d)

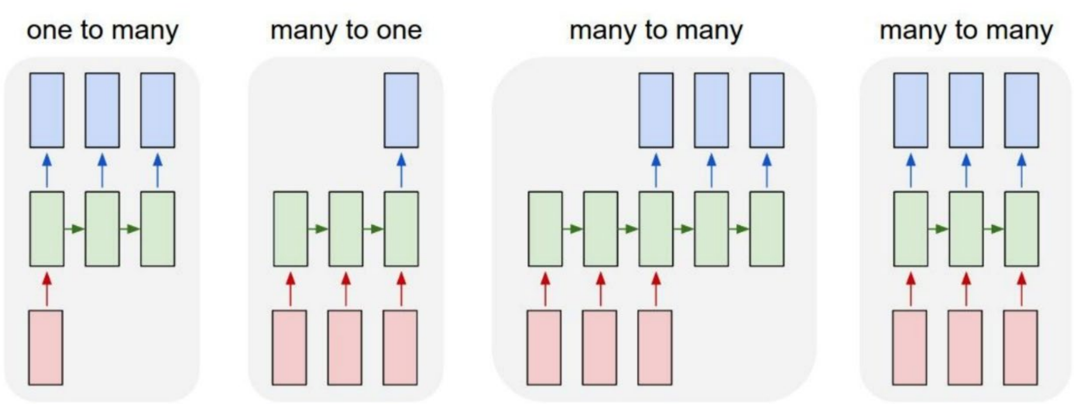
- Connecting every hidden state between encoder and decoder?



- Infeasible!
  - Both inputs and outputs are with varying sizes.
  - Overparameterized
  - Possible solution: attention (will cover next week)

Recent Advances of GAN-based Models for Video-Based Applications

- Video Generation
  - MoCoGAN: Decomposing Motion and Content for Video Generation (CVPR'18)
- Video Prediction
  - Deterministic
    - Unsupervised Learning of Video Representations using LSTMs (ICML'15)
    - Decomposing Motion and Content for Natural Video Sequence Prediction (ICLR'17)
    - Learning to generate long-term future via hierarchical prediction (ICML'17)
  - Stochastic (if time permits)
    - Stochastic Video Generation with a Learned Prior (Denton et al., ArXiv'18)



Video Generation

- Learning a latent space to describe image/video data
- Input: latent representation
- Output: sequence of images/frames (i.e., video)

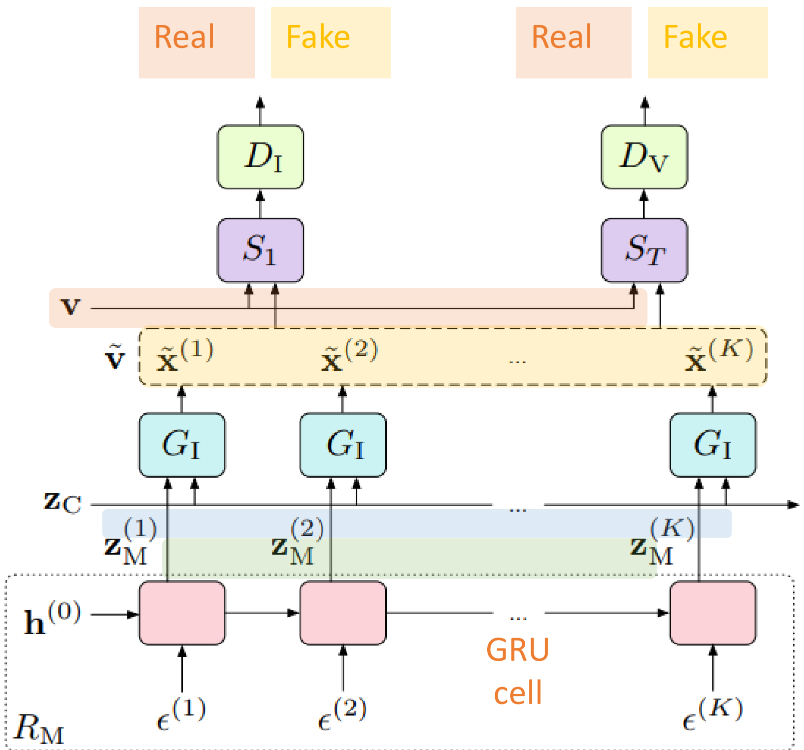
E.g., Latent space for content-motion decomposition

Latent Space

Image / Video Space

Video Generation

- MoCoGAN: Decomposing Motion and Content for Video Generation (Tulyakov et al, CVPR'18)



Generated Video: $\tilde{\mathbf{v}} = \left[G_\mathrm{I}\left(\begin{bmatrix} z_\mathrm{C} \\ R_\mathrm{M}(1) \end{bmatrix}\right), ..., G_\mathrm{I}\left(\begin{bmatrix} z_\mathrm{C} \\ R_\mathrm{M}(\kappa) \end{bmatrix}\right) \right]$

Real Video: $\mathbf{v}$

__Objective Function:__

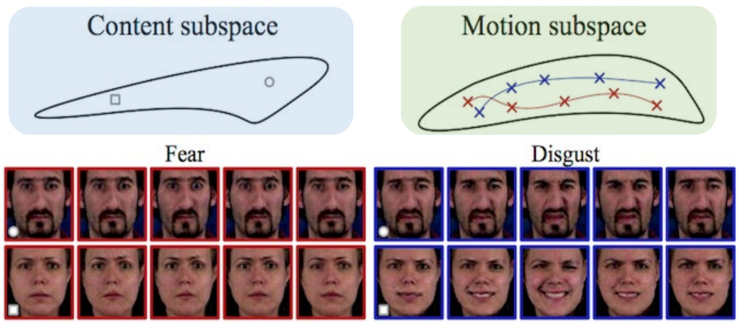
$$\max_{G_\mathrm{I}, R_\mathrm{M}} \min_{D_\mathrm{I}, D_\mathrm{V}} \mathcal{F}_\mathrm{V}(D_\mathrm{I}, D_\mathrm{V}, G_\mathrm{I}, R_\mathrm{M})$$

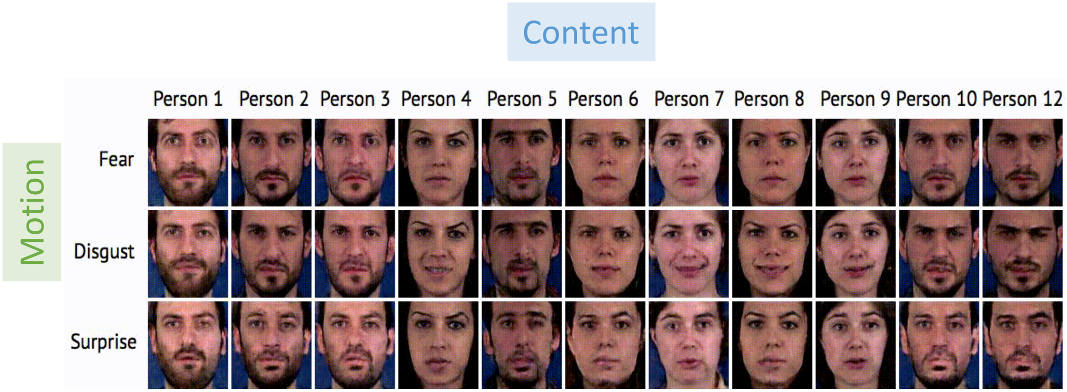
$$= \mathbb{E}_{\mathbf{v}}[-\log D_\mathrm{I}(S_1(\mathbf{v}))] + \mathbb{E}_{\tilde{\mathbf{v}}}[-\log(1 - D_\mathrm{I}(S_1(\tilde{\mathbf{v}})))] + \mathbb{E}_{\mathbf{v}}[-\log D_\mathrm{V}(S_\mathrm{T}(\mathbf{v}))] + \mathbb{E}_{\tilde{\mathbf{v}}}[-\log(1 - D_\mathrm{V}(S_\mathrm{T}(\tilde{\mathbf{v}})))]$$

Example Results

- MoCoGAN: Decomposing Motion and Content for Video Generation (Tulyakov et al, CVPR'18)

Results:



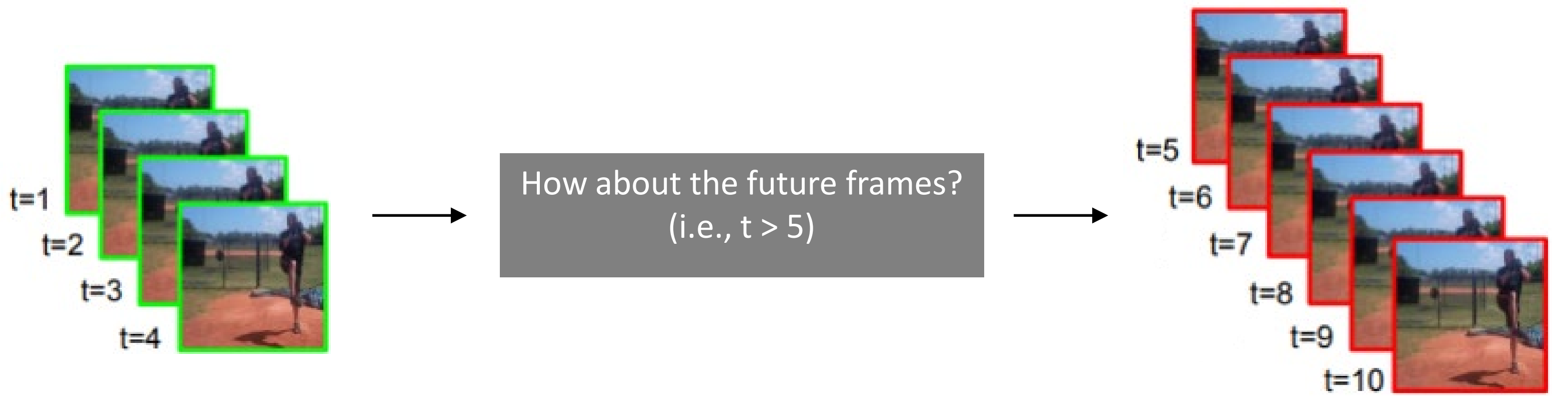


Recent Advances of Attention models in Video-Based Applications

- Video Generation
    - MoCoGAN: Decomposing Motion and Content for Video Generation (CVPR'18)
- Video Prediction
    - Deterministic
        - Unsupervised Learning of Video Representations using LSTMs (ICML'15)
        - Decomposing Motion and Content for Natural Video Sequence Prediction (ICLR'17)
        - Learning to generate long-term future via hierarchical prediction (ICML'17)
    - Stochastic
        - Stochastic Video Generation with a Learned Prior (Denton et al., ArXiv'18)

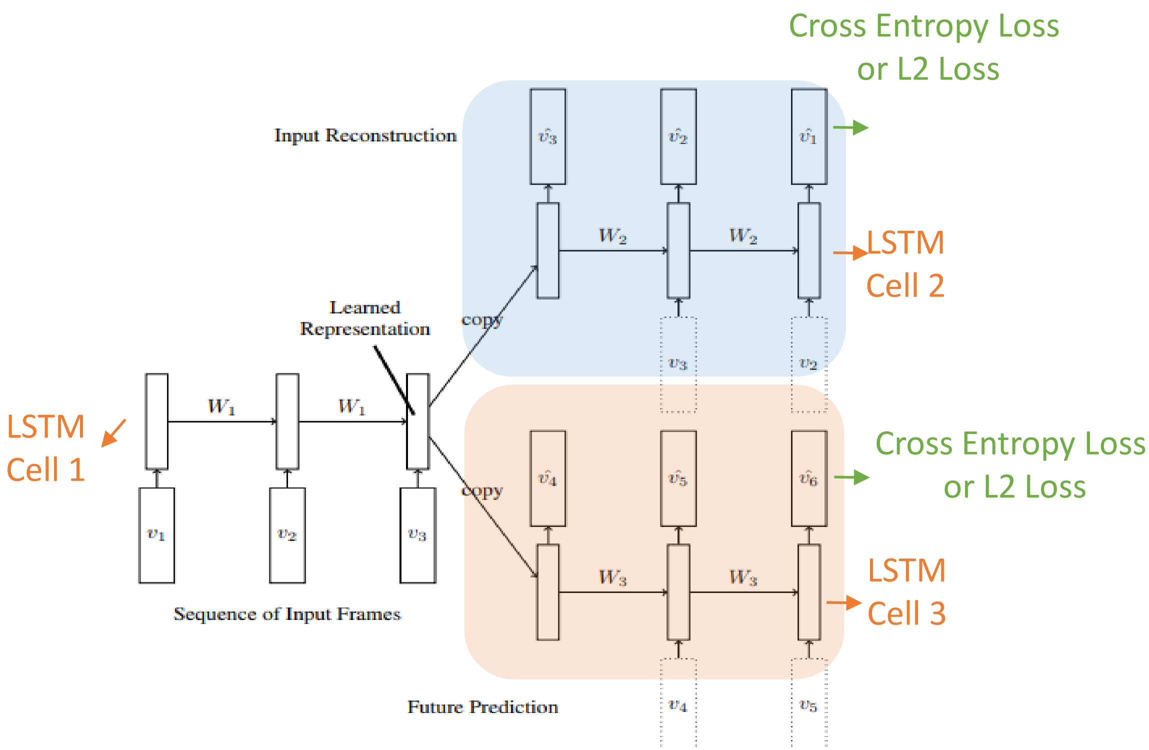
Video Prediction

- Input: A few known frames
- Output: Unknown future frames



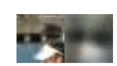
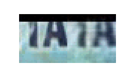
Video Prediction - Deterministic

- Unsupervised Learning of Video Representations using LSTMs
  (Srivastava et al., ICML'15)



- LSTM Encoder-Decoder model
- Two tasks: Reconstruction & Prediction
- Results (L: ref video, R: output video)
  - Bouncing (Moving) MNIST


  - Video patches of UCF-101

Video Prediction - Deterministic

- Decomposing Motion and Content for Natural Video Sequence Prediction
  (Villegas et al., ICLR'17)

History of image differences

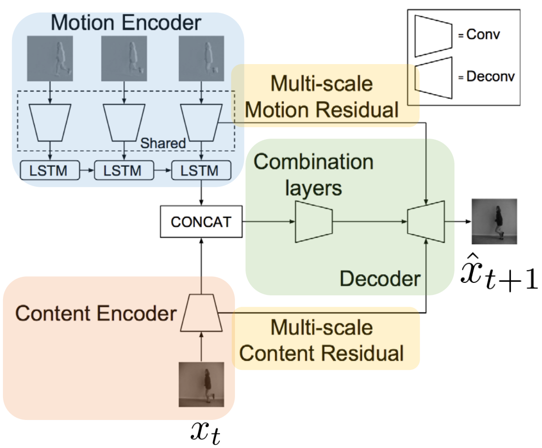


Image at time t

Hidden state (motion)

cell state

Motion Encoder

$$[\mathbf{d}_t, \mathbf{c}_t] = f^{\text{dyn}}\left(\mathbf{x}_t - \mathbf{x}_{t-1}, \mathbf{d}_{t-1}, \mathbf{c}_{t-1}\right)$$

Content Encoder

$$\mathbf{s}_t = f^{\text{cont}}\left(\mathbf{x}_t\right)$$

Multi-scale Motion & Content Residual

$$\mathbf{r}_t^l = f^{\text{res}}\left(\left[\mathbf{s}_t^l, \mathbf{d}_t^l\right]\right)^l$$

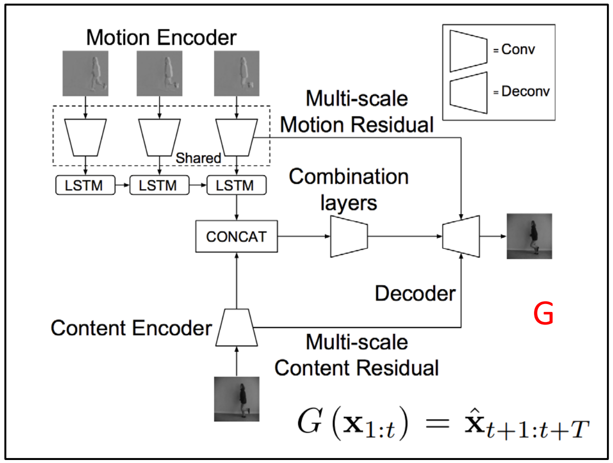
Combination Layers and Decoder

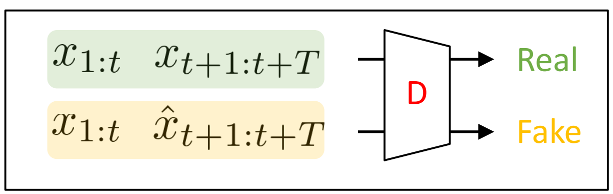
$$\mathbf{f}_t = g^{\text{comb}}\left([\mathbf{d}_t, \mathbf{s}_t]\right) \qquad \hat{\mathbf{x}}_{t+1} = g^{\text{dec}}\left(\mathbf{f}_t, \mathbf{r}_t\right)$$

$\longrightarrow$ Recurrently generate $\hat{x}_{t+2}, \hat{x}_{t+3}, \ldots, \hat{x}_{t+T}$

$\longrightarrow$ $G\left(\mathbf{x}_{1:t}\right) = \hat{\mathbf{x}}_{t+1:t+T}$

- Decomposing Motion and Content for Natural Video Sequence Prediction
  (Villegas et al., ICLR'17)





<u>Objective Function:</u> Adversarial Training -> alternate btw min L & Ldisc

Update Image Generation Network (G)

Pixel Value similarity

Gradient of Pixel Value similarity

$$\mathcal{L} = \alpha \mathcal{L}_{\text{img}} + \beta \mathcal{L}_{\text{GAN}}$$

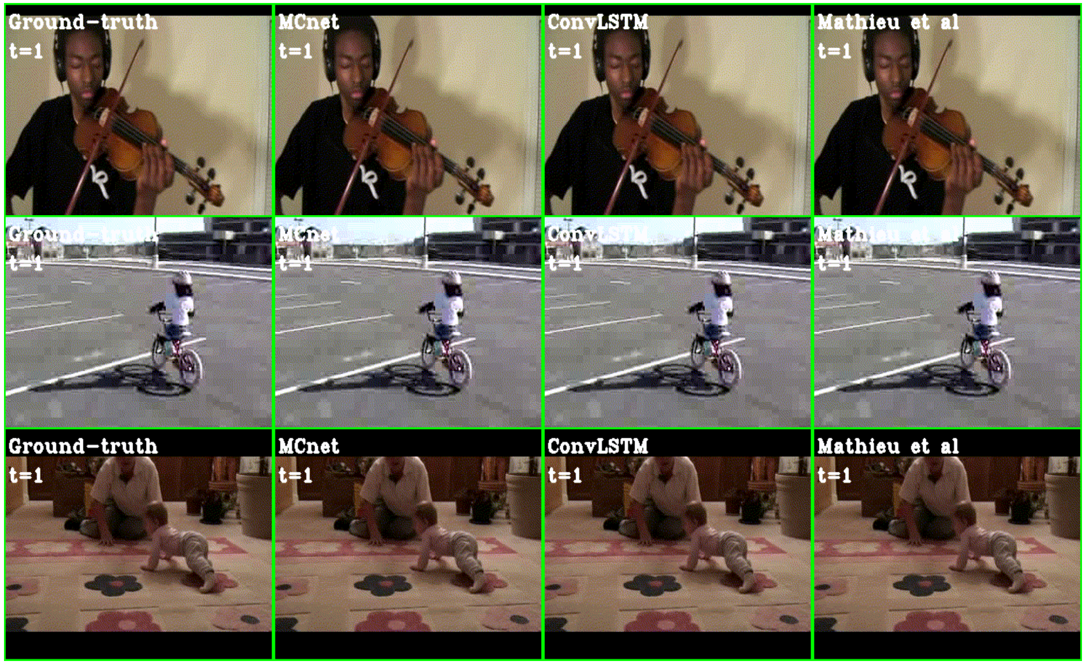
$$\mathcal{L}_{\text{img}} = \mathcal{L}_p\left(\mathbf{x}_{t+k}, \hat{\mathbf{x}}_{t+k}\right) + \mathcal{L}_{gdl}\left(\mathbf{x}_{t+k}, \hat{\mathbf{x}}_{t+k}\right)$$

$$\mathcal{L}_{\text{GAN}} = -\log D\left(\left[\mathbf{x}_{1:t}, G\left(\mathbf{x}_{1:t}\right)\right]\right)$$

Update Discriminator (D)

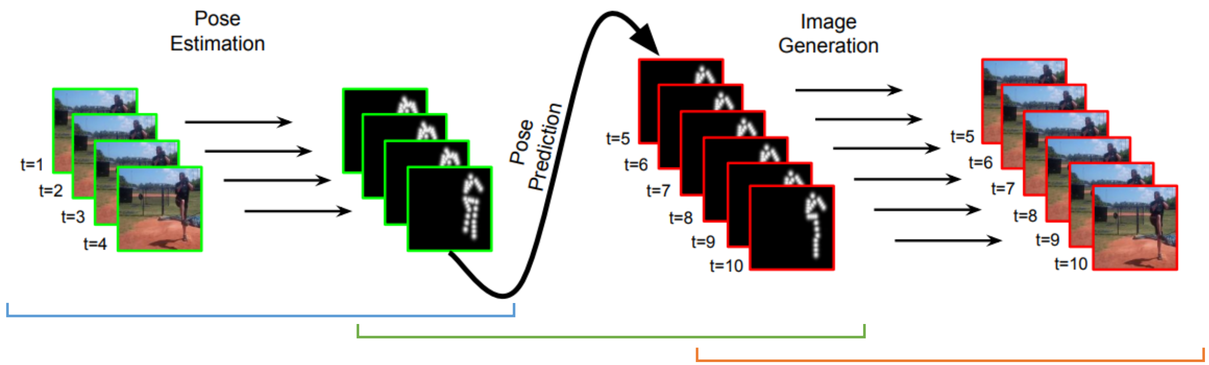
$$\mathcal{L}_{\text{disc}} = -\log D\left(\left[\mathbf{x}_{1:t}, \mathbf{x}_{t+1:t+T}\right]\right) - \log\left(1 - D\left(\left[\mathbf{x}_{1:t}, G\left(\mathbf{x}_{1:t}\right)\right]\right)\right)$$

- Example Results



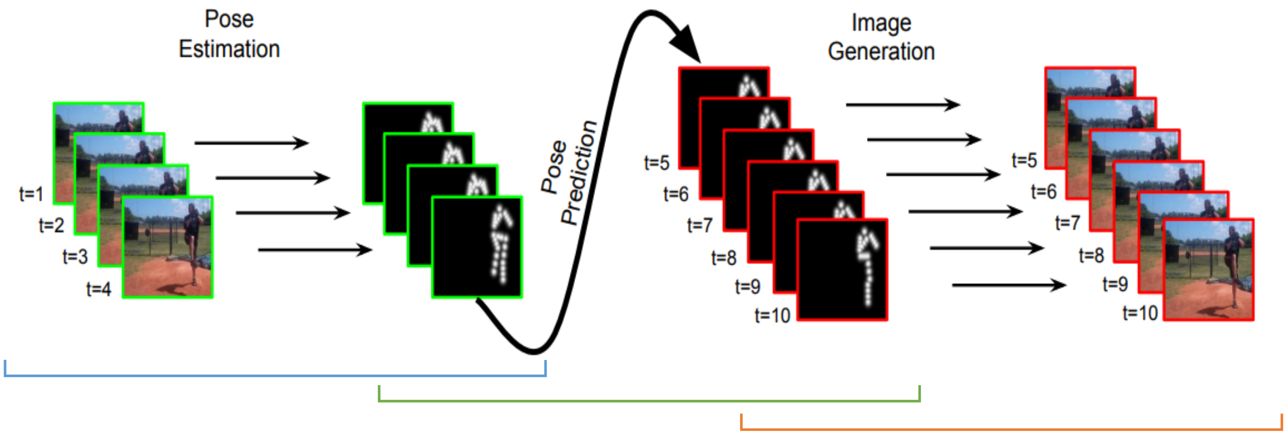
Video Prediction - Deterministic

- Learning to generate long-term future via hierarchical prediction
  (Villegas et al., ICML’17)



Stage 1:	*Pose Estimation*	Hourglass network (Newell et al., ECCV’16)
Stage 2:	*Future Pose Prediction*	Sequence-to-Sequence model on high-level structure
Stage 3:	*Image Generation*	Visual-Structure Analogy

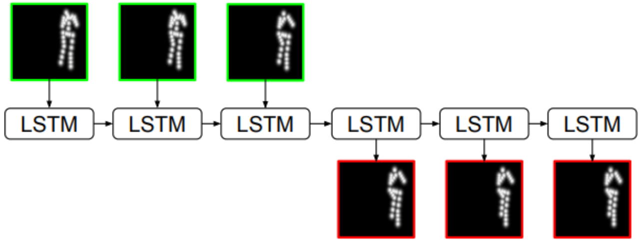
- Learning to generate long-term future via hierarchical prediction
  (Villegas et al., ICML'17)



<u>Step 2:</u>

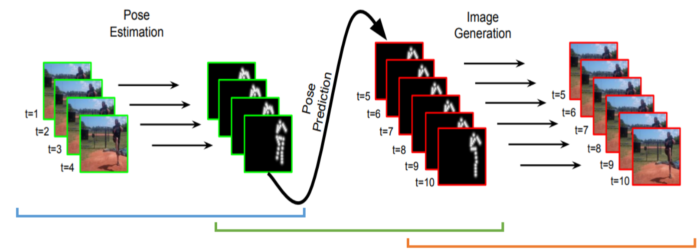
Future Pose Prediction

Sequence-to-Sequence model on high-level structure



<u>*Objective Function:*</u>

$$\mathcal{L}_{\text{pose}} = \frac{1}{TL}\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbb{1}_{\{m^l_{k+t}=1\}}\|\hat{\mathbf{p}}^l_{k+t} - \mathbf{p}^l_{k+t}\|_2^2$$



- Learning to generate long-term future via hierarchical prediction
  (Villegas et al., ICML'17)

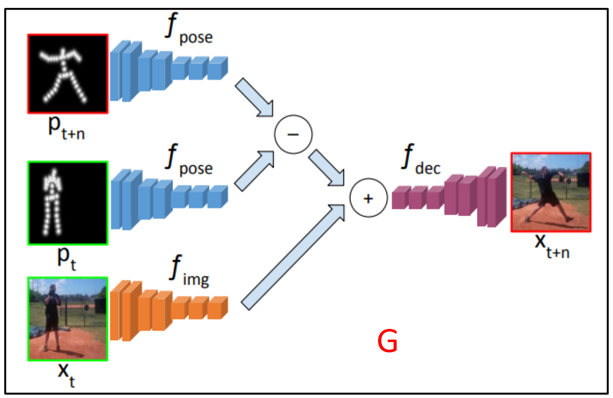
<u>Step 3:</u>

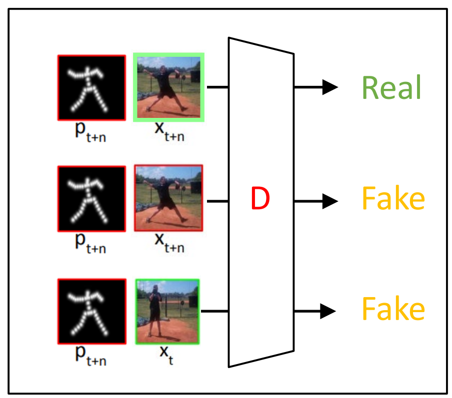
Image Generation

Visual-Structure Analogy

<u>*Objective Function:*</u>

Adversarial Training -> alternately minimize L & LDisc





Update Image Generation Network (G)

$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{Gen}}$$

$$\mathcal{L}_{\text{img}} = \|\mathbf{x}_{t+n} - \hat{\mathbf{x}}_{t+n}\|_2^2$$

$$\mathcal{L}_{\text{feat}} = \|C_1(\mathbf{x}_{t+n}) - C_1(\hat{\mathbf{x}}_{t+n})\|_2^2 + \|C_2(\mathbf{x}_{t+n}) - C_2(\hat{\mathbf{x}}_{t+n})\|_2^2$$

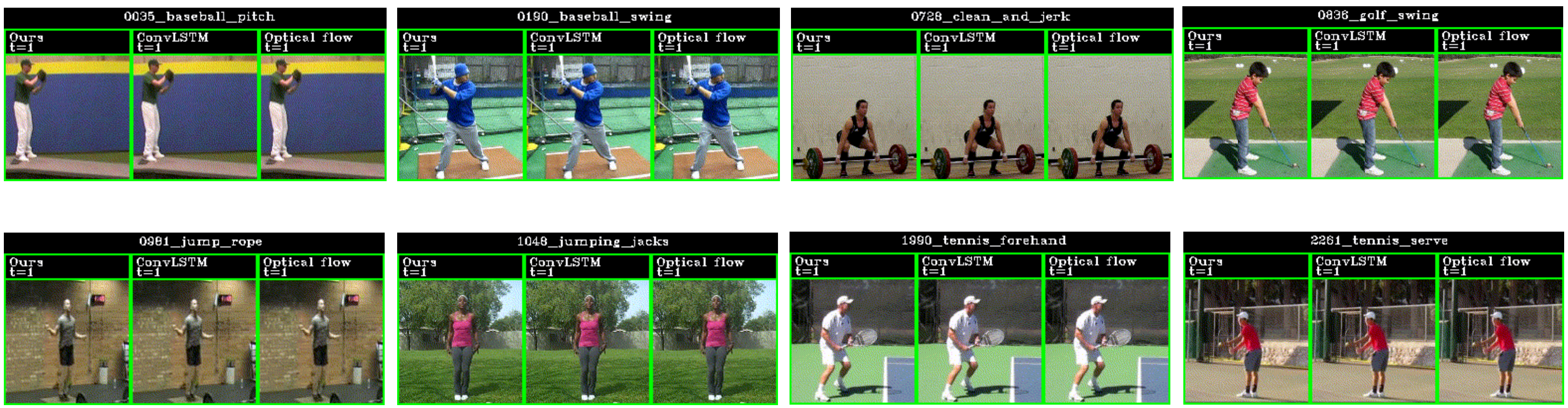
$$\mathcal{L}_{\text{Gen}} = -\log D([\mathbf{p}_{t+n}, \hat{\mathbf{x}}_{t+n}])$$

Update Discriminator (D)

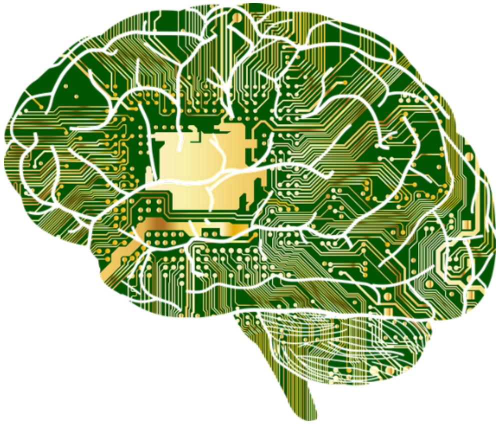
$$\mathcal{L}_{\text{Disc}} = -\log D([\mathbf{p}_{t+n}, \mathbf{x}_{t+n}]) - 0.5\log(1 - D([\mathbf{p}_{t+n}, \hat{\mathbf{x}}_{t+n}])) - 0.5\log(1 - D([\mathbf{p}_{t+n}, \mathbf{x}_t])),$$

- Example Results

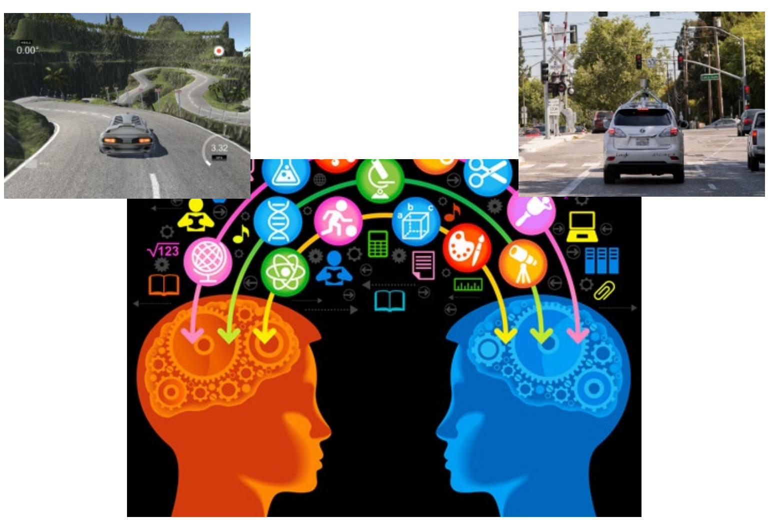
Results on Penn Action Dataset:

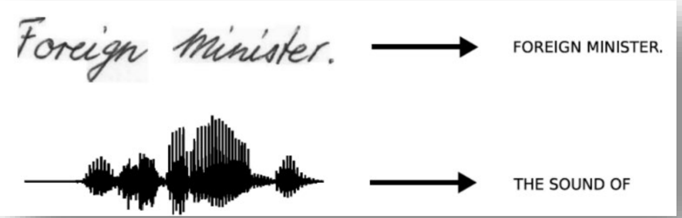


What to Be Covered Today…



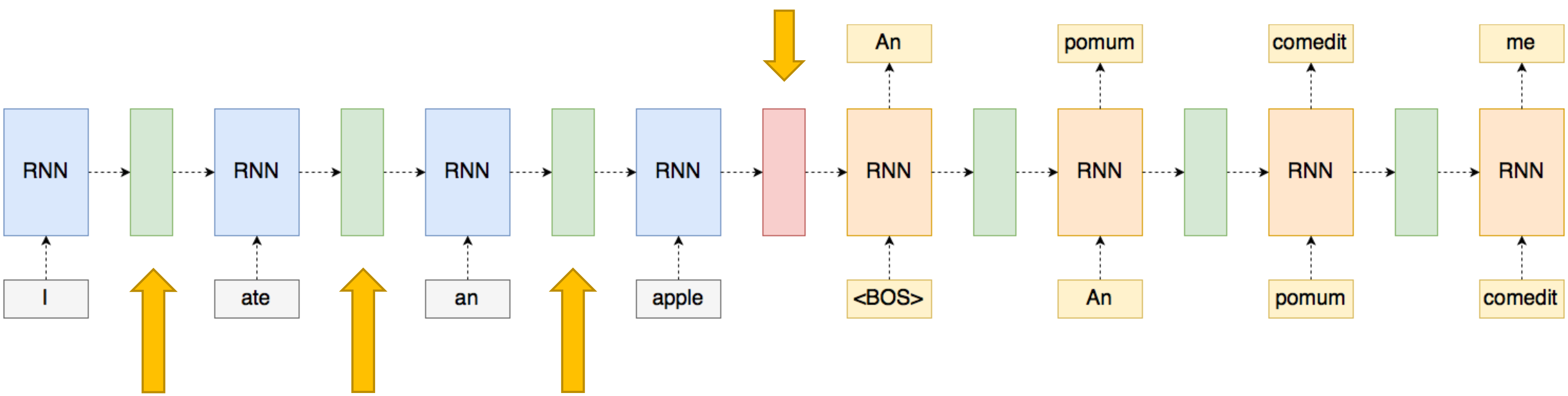
- Transfer Learning
  - Visual Classification – Domain Adaptation
  - Visual Synthesis – Style Transfer
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Selected Models for Sequence-to-Sequence Learning
  - Attention in RNN





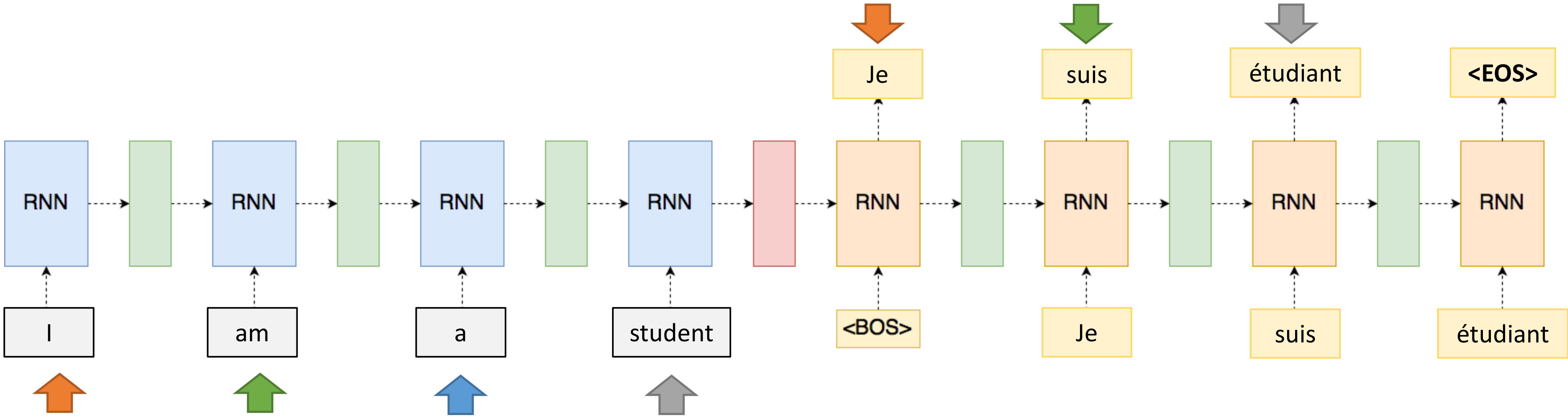
What's the Potential Problem of RNN?

- Each hidden state vector extracts/carries information across time steps (some might be diluted downstream).
- However, information of the entire input sequence is embedded into a single hidden state vector.



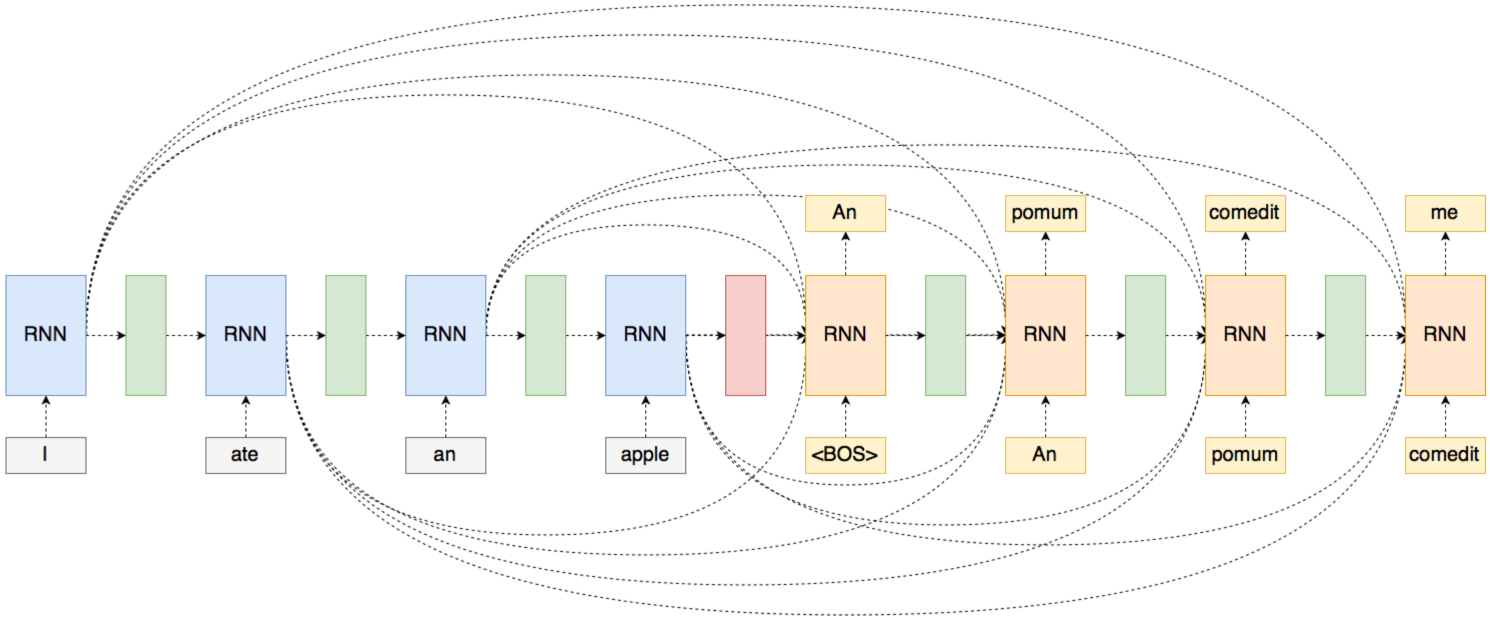
What’s the Potential Problem? (cont’d)

- Outputs at different time steps have particular meanings.
- However, synchrony between **input** and **output seqs** is not required**.**



RNN with Attention is Good, But..

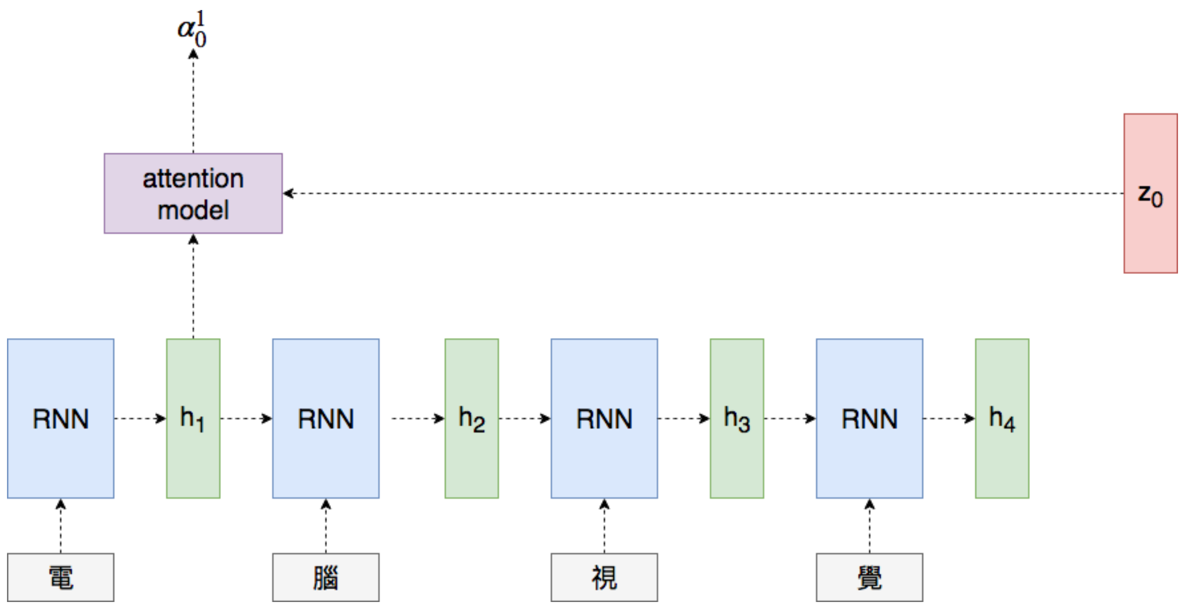
- Connecting every hidden state between encoder and decoder?



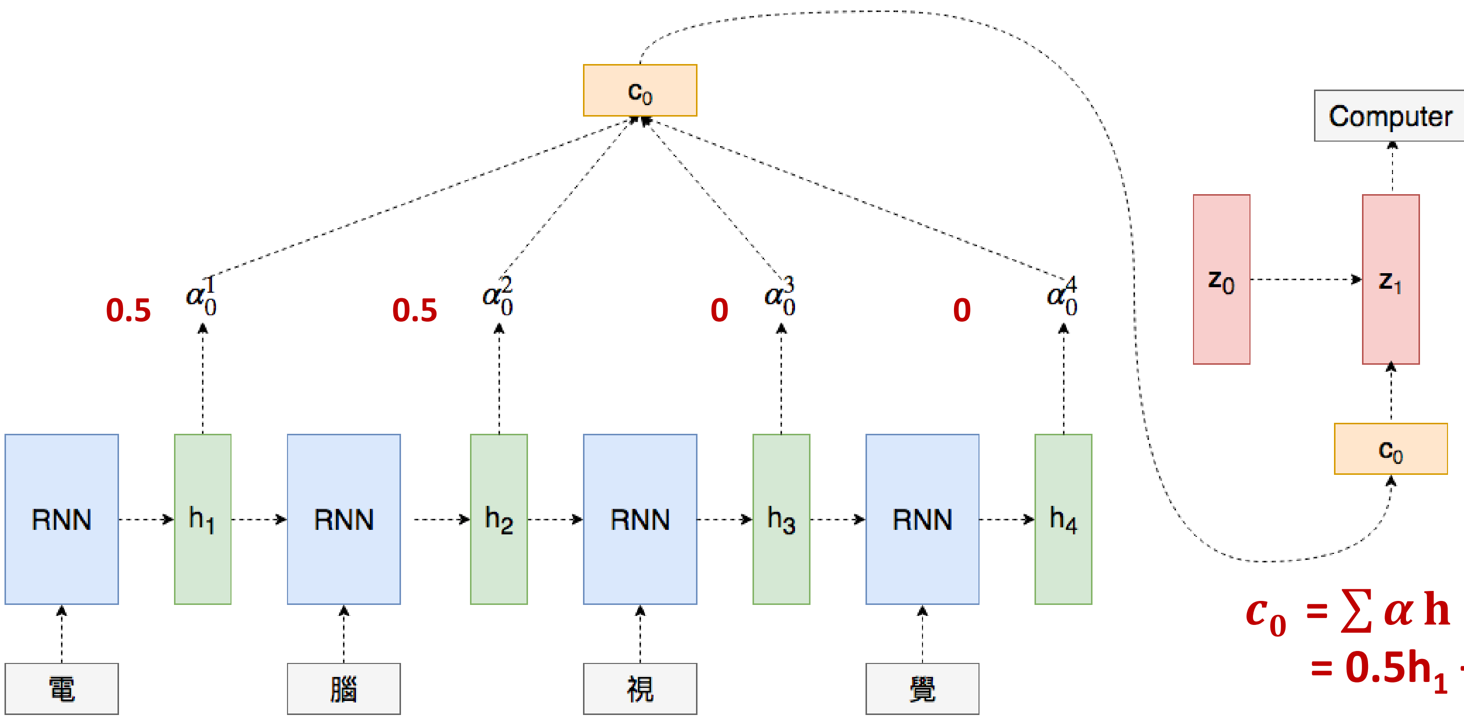
- Infeasible!
  - Both inputs and outputs are with varying sizes.
  - Overparameterized

Solution Ver. 1: Attention Model

- What should the attention model be?
  - A NN whose inputs are z and h while output is a scalar, indicating the similarity between z and h.
- Most attention models are jointly learned or trained with other parts of network (e.g., recognition, etc.)

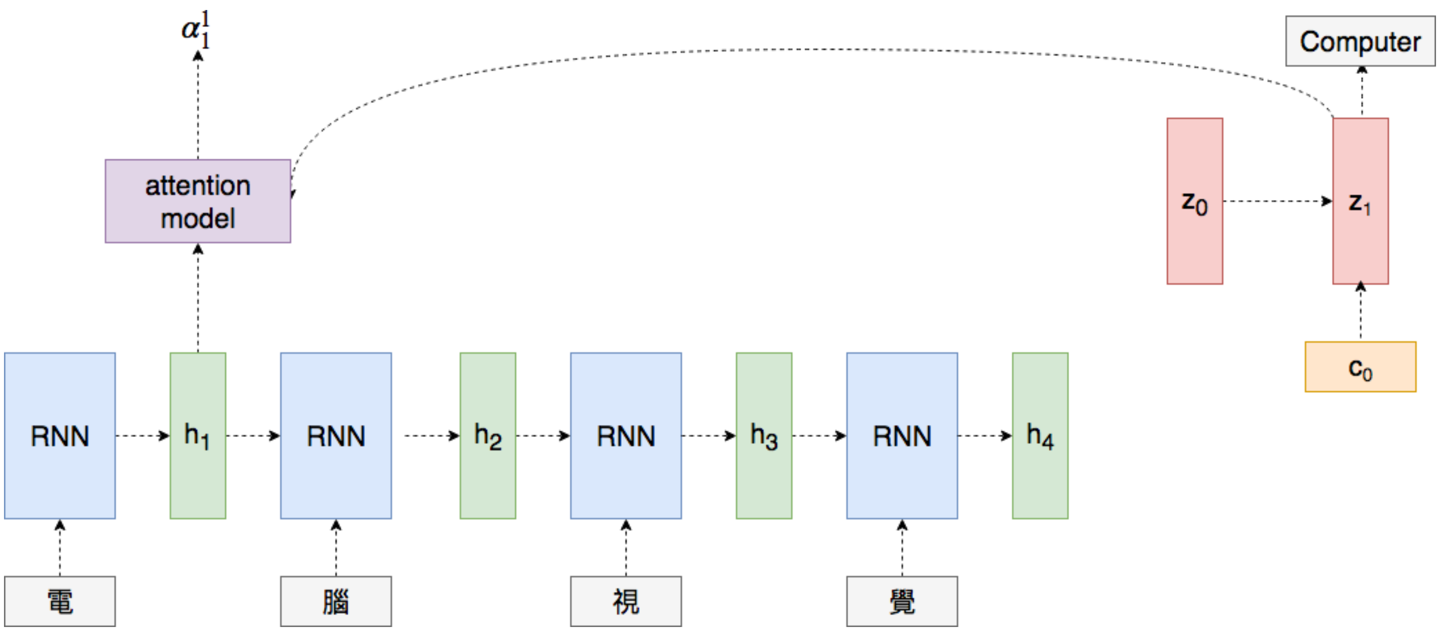


Solution: Attention Model

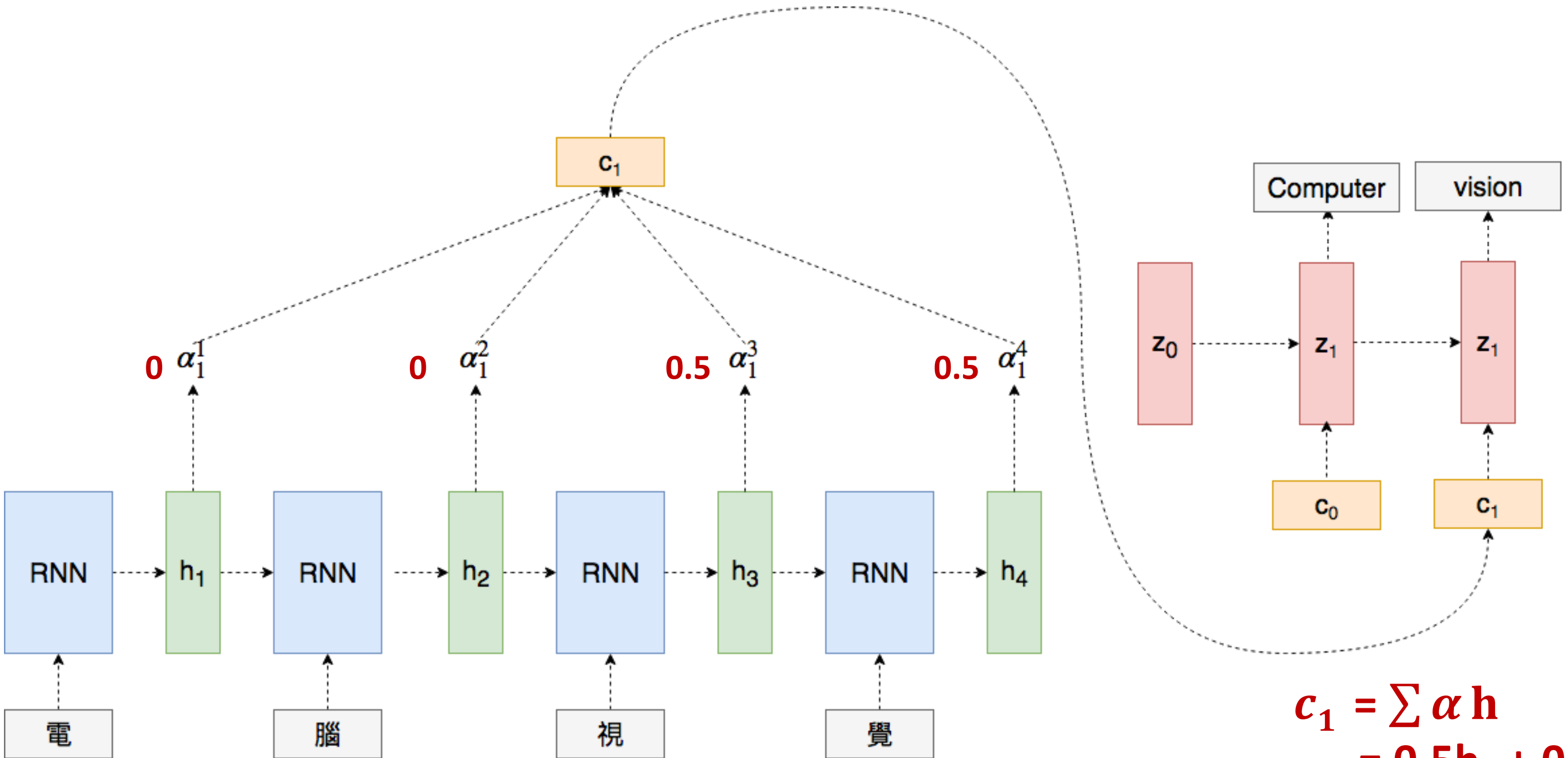


$$c_0 = \sum \alpha \, \mathrm{h}$$
$$= 0.5\mathrm{h}_1 + 0.5\mathrm{h}_2$$

Solution: Attention Model

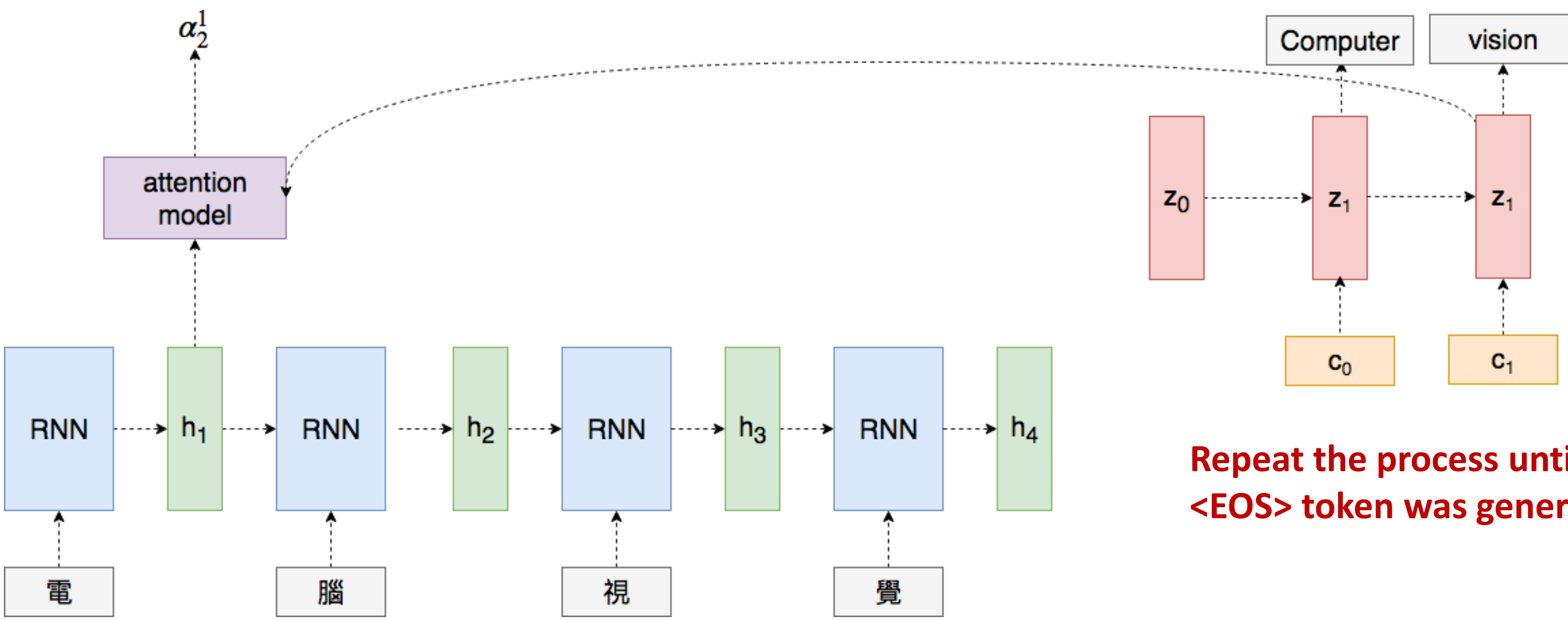


Solution: Attention Model



$c_1 = \sum \alpha \, \mathbf{h}$
$= 0.5\mathbf{h}_3 + 0.5\mathbf{h}_4$

Solution: Attention Model



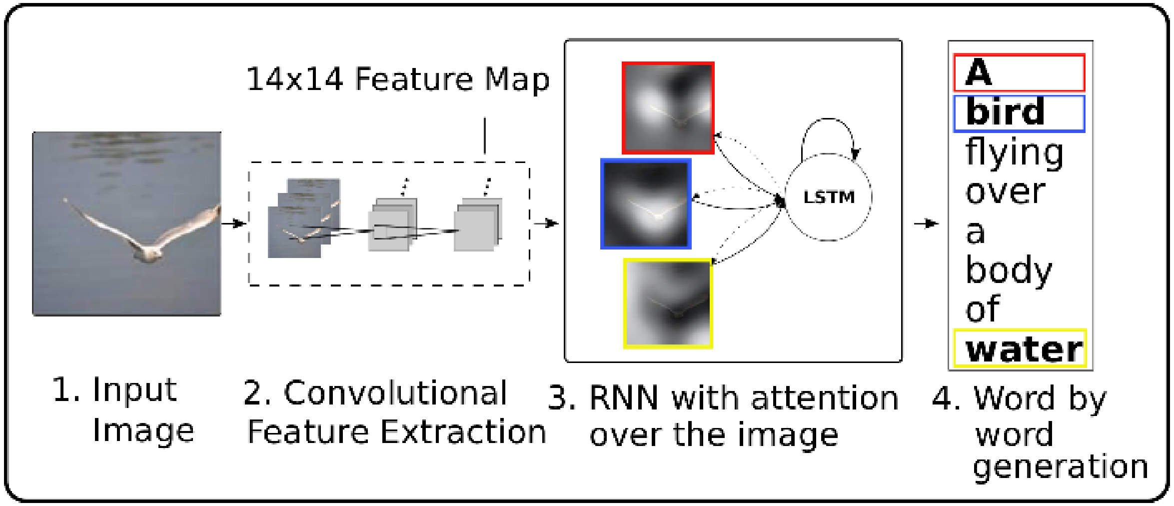
Repeat the process until <EOS> token was generated

Selected Attention Models for Image-Based Applications

- Image Captioning
  - Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML ’15
- Visual Question Answering
  - Zhu et al, “Visual7W: Grounded Question Answering in Images”, CVPR ’16
- Image Classification
  - Mnih et al, “Recurrent Models of Visual Attention”, NIPS ’14
- Image Generation
  - Gregor et al, “DRAW: A Recurrent Neural Network For Image Generation”, ICML ’15

Image Captioning with Attention

- RNN focuses visual attention at different spatial locations when generating corresponding words during captioning.



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

Image Captioning with Attention

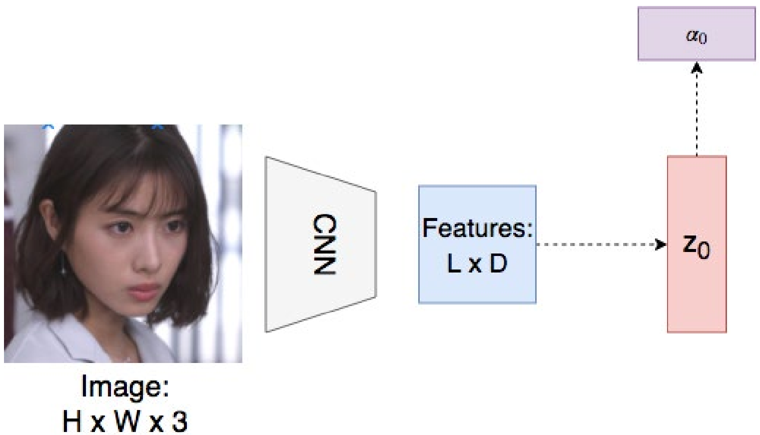


Image Captioning with Attention

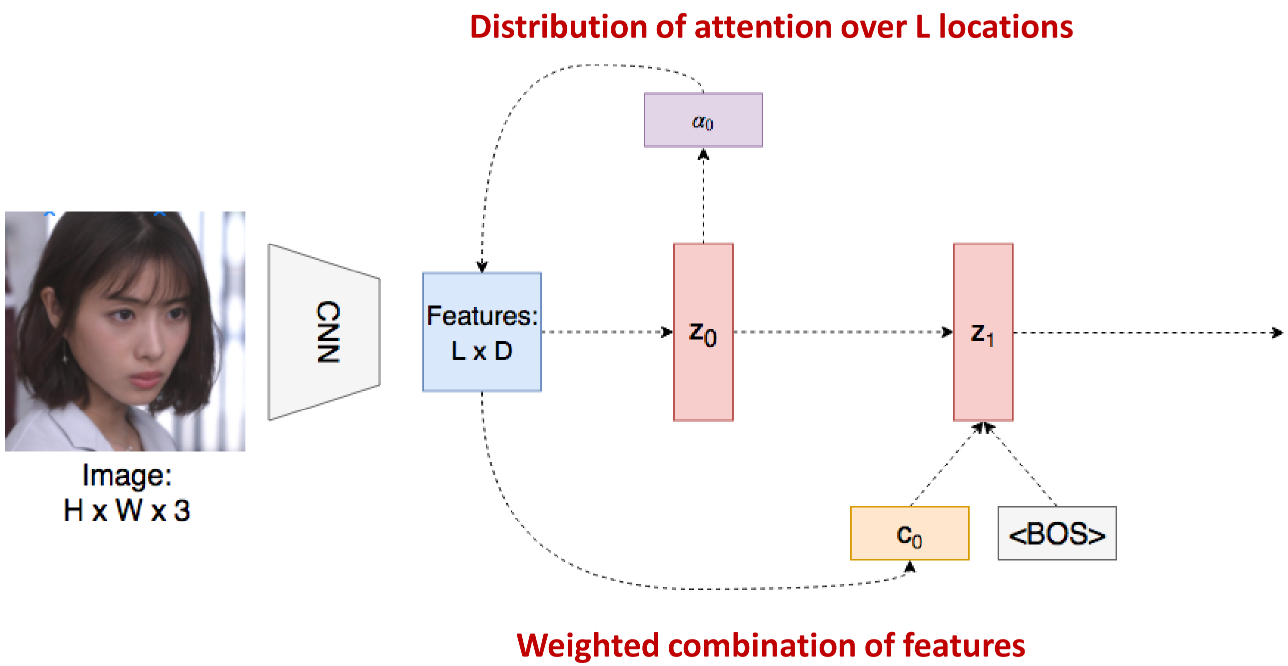


Image Captioning with Attention

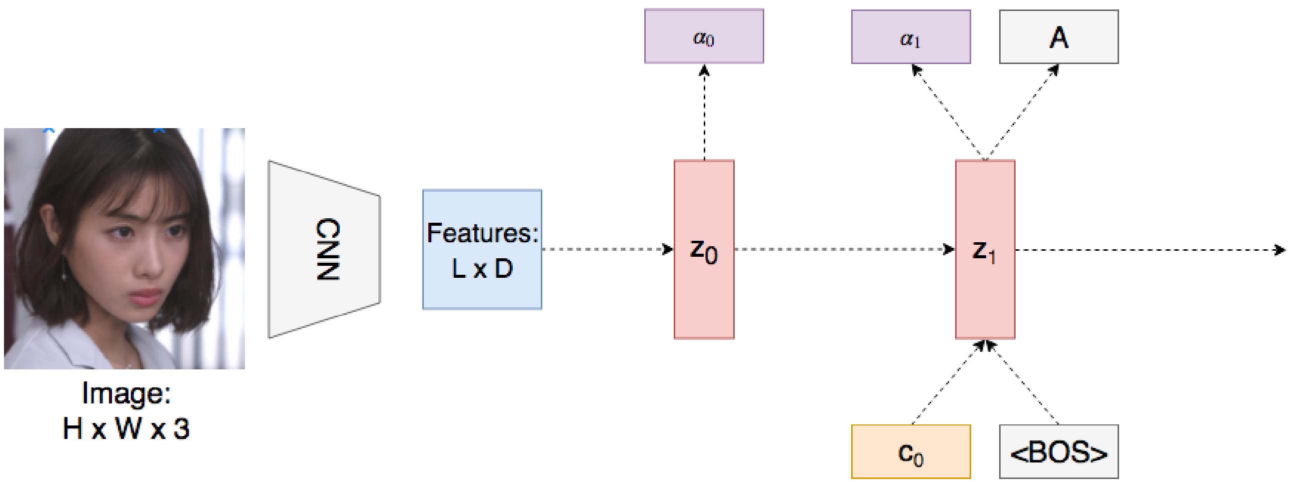


Image Captioning with Attention

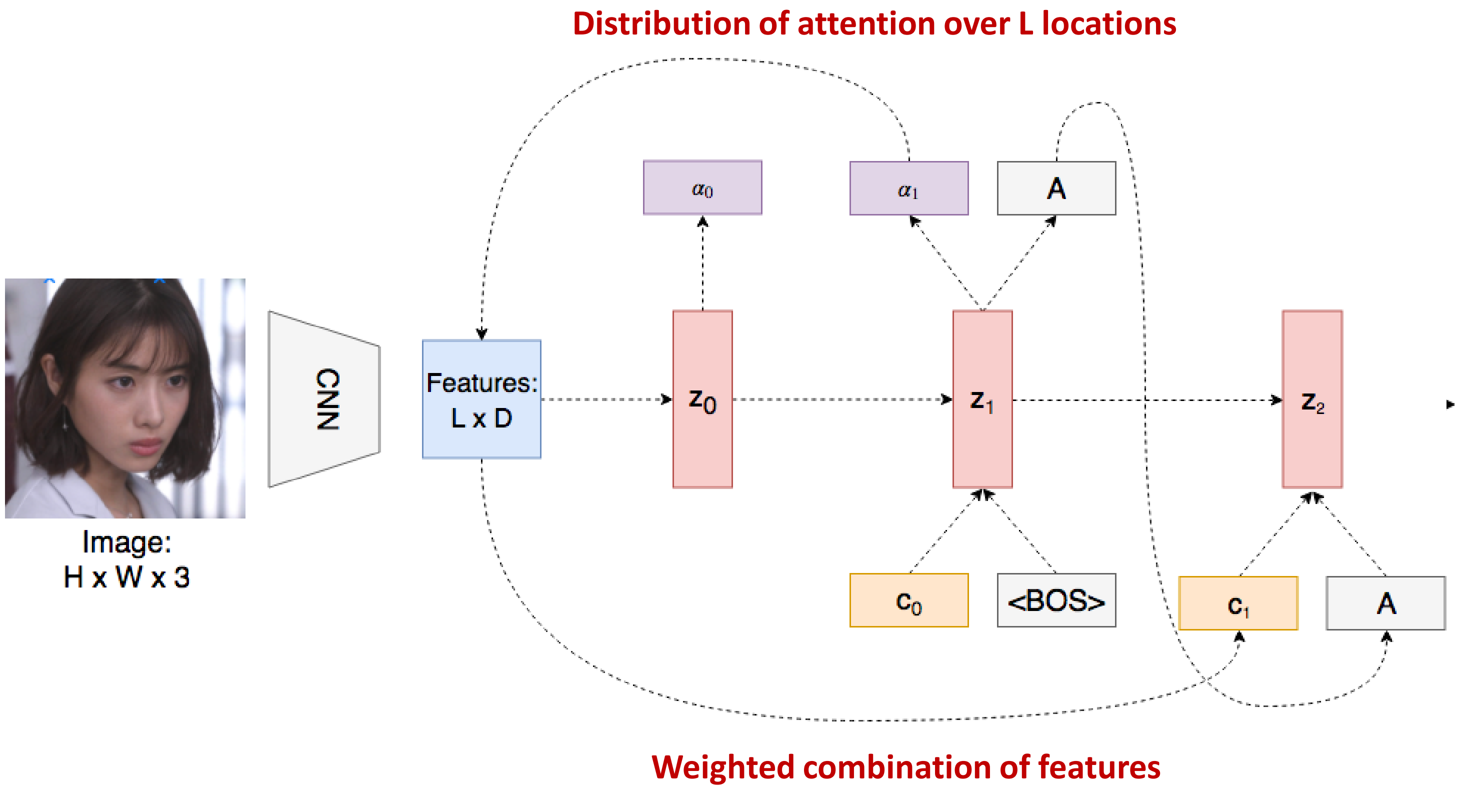
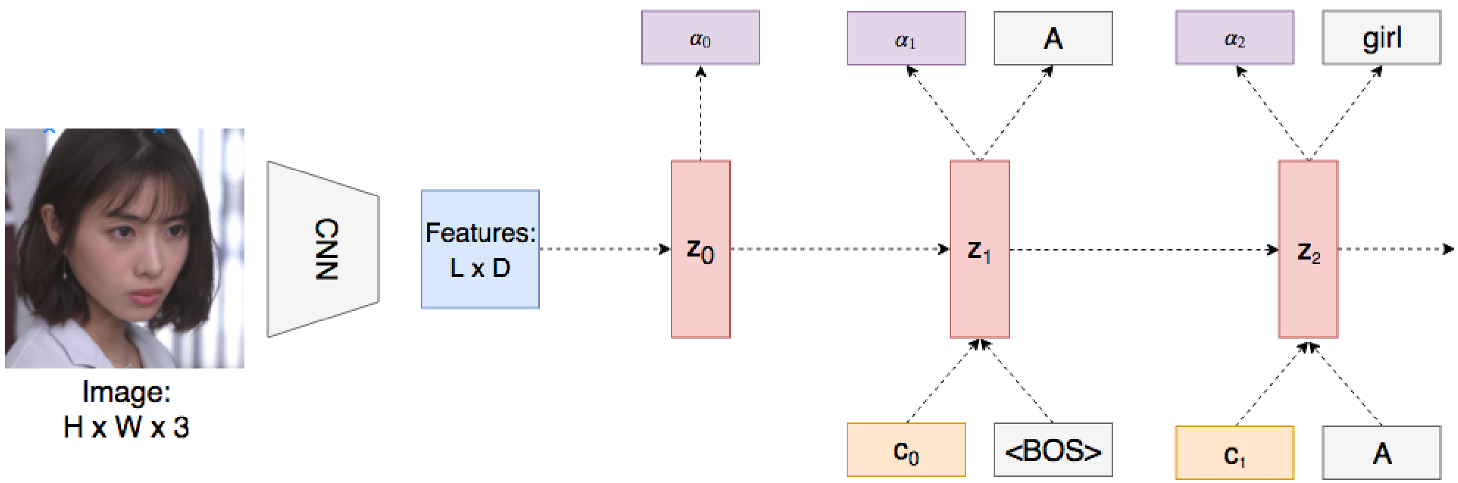
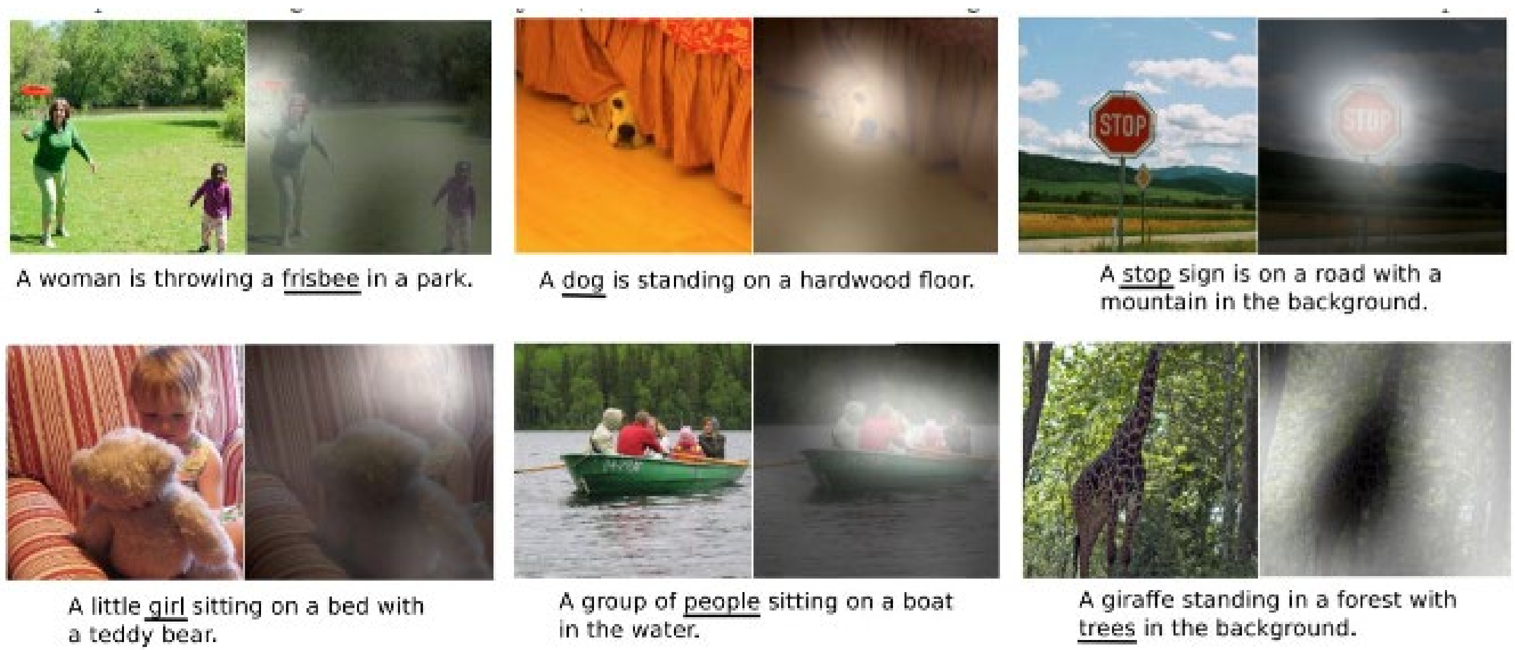


Image Captioning with Attention



Repeat the process until <EOS> token was generated

Image Captioning with Attention



Selected Attention Models for Image-Based Applications

- Image Captioning
  - Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML '15
- Visual Question Answering
  - Zhu et al, "Visual7W: Grounded Question Answering in Images", CVPR '16
- Image Classification
  - Mnih et al, "Recurrent Models of Visual Attention", NIPS '14
- Image Generation
  - Gregor et al, "DRAW: A Recurrent Neural Network For Image Generation", ICML '15

Visual Question Answering

- Examples of multiple-choice QA & pointing QA

Q: What endangered animal is featured on the truck?
A: A bald eagle.
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?
A: Onto 24 ¾ Rd.
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: When was the picture taken?
A: During a wedding.
A: During a bar mitzvah.
A: During a funeral.
A: During a Sunday church service.

Q: Who is under the umbrella?
A: Two women.
A: A child.
A: An old man.
A: A husband and a wife.

Q: Which pillow is farther from the window?

Q: Which step leads to the tub?

Q: Which is the small computer in the corner?

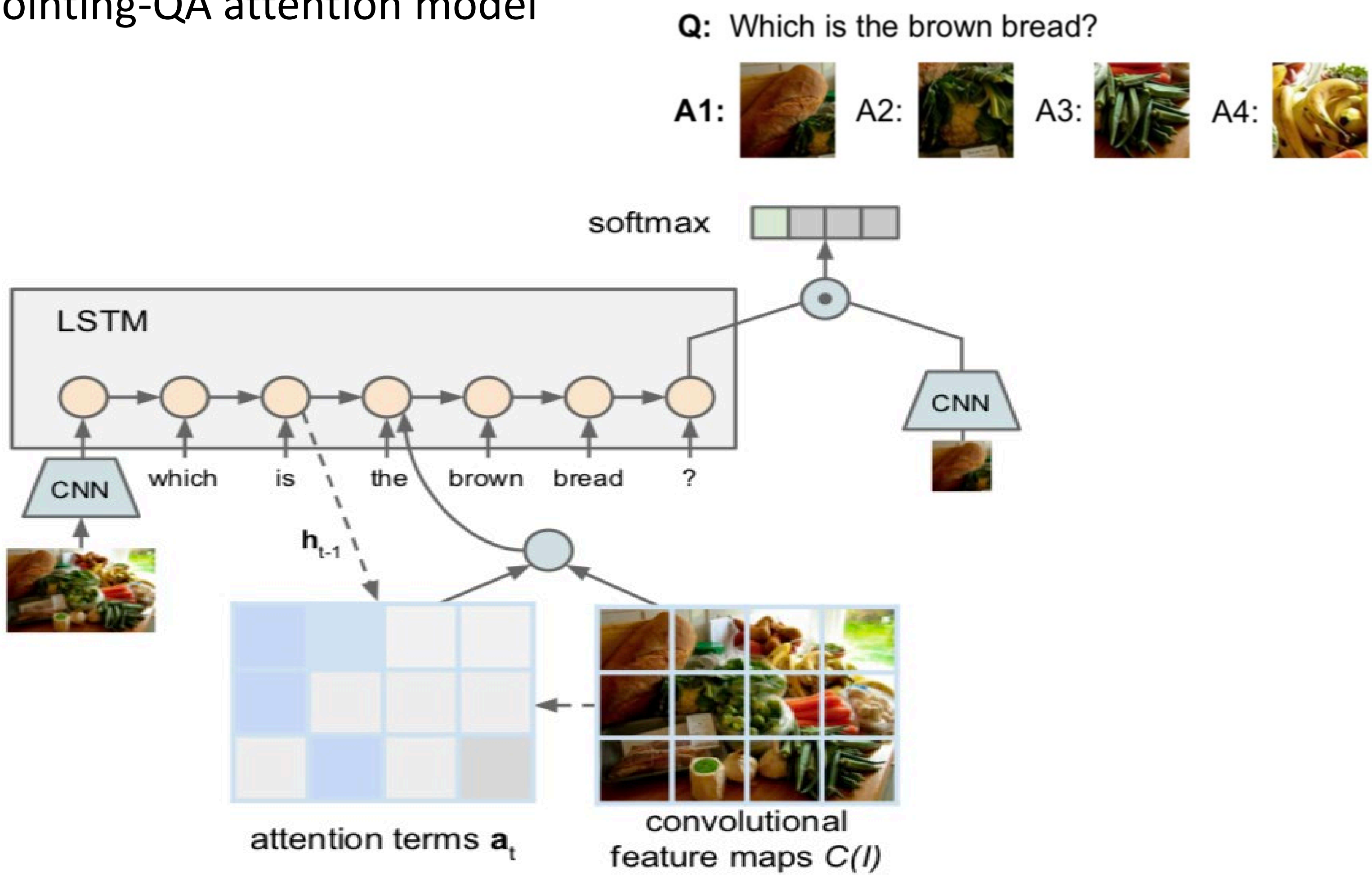
Q: Which item is used to cut items?

Q: Which doughnut has multicolored sprinkles?

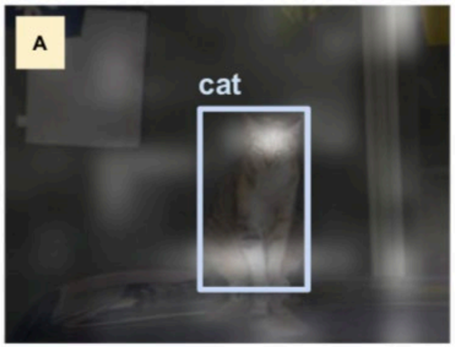
Q: Which man is wearing the red tie?

Visual Question Answering with Attention

- Pointing-QA attention model



Visual Question Answering with Attention



What kind of animal is in the photo?
A **cat**.

Why is the person holding a knife?
To cut the **cake** with.

Where are the carrots?
At the top.

How many people are there?
Three.

The peaks of the attention maps reside in the bounding boxes of the target objects.

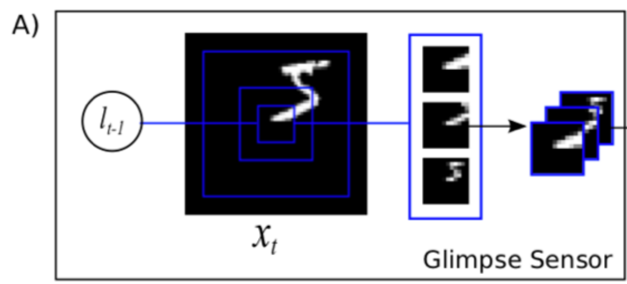
The bottom two examples show QA pairs with answers not explicitly containing objects. The attention heat maps are scattered around the image grids.

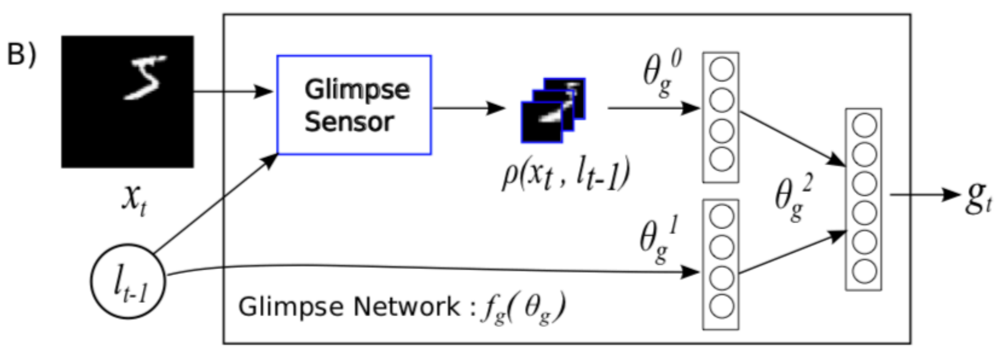
Selected Attention Models for Image-Based Applications

- Image Captioning
  - Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML ’15
- Visual Question Answering
  - Zhu et al, “Visual7W: Grounded Question Answering in Images”, CVPR ’16
- Image Classification
  - Mnih et al, “Recurrent Models of Visual Attention”, NIPS ’14
- Image Generation
  - Gregor et al, “DRAW: A Recurrent Neural Network For Image Generation”, ICML ’15

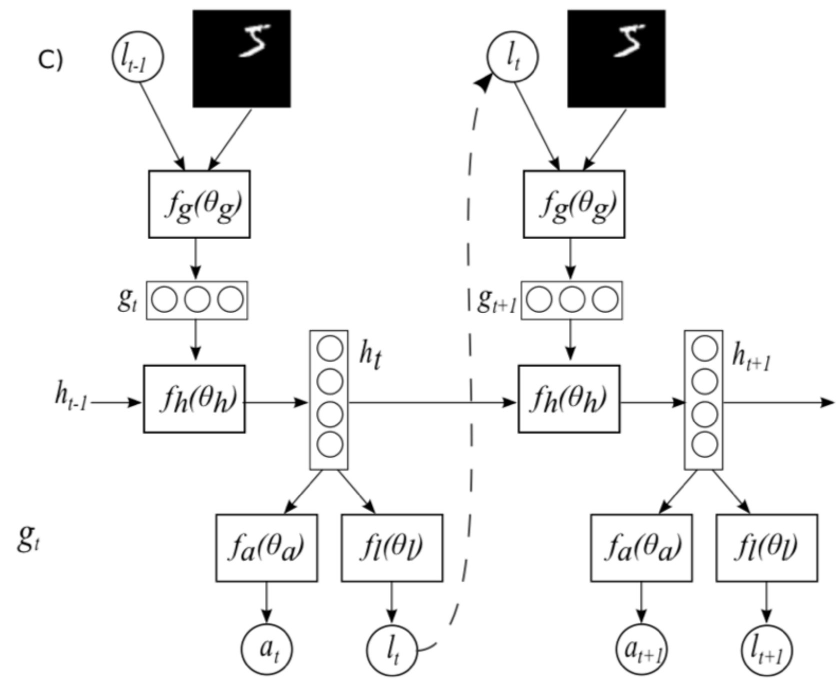
Glimpse Sensor & Glimpse Network

Glimpse sensor: extracts a retina-like representation centered at l_{t-1} that contains multiple resolution patches.





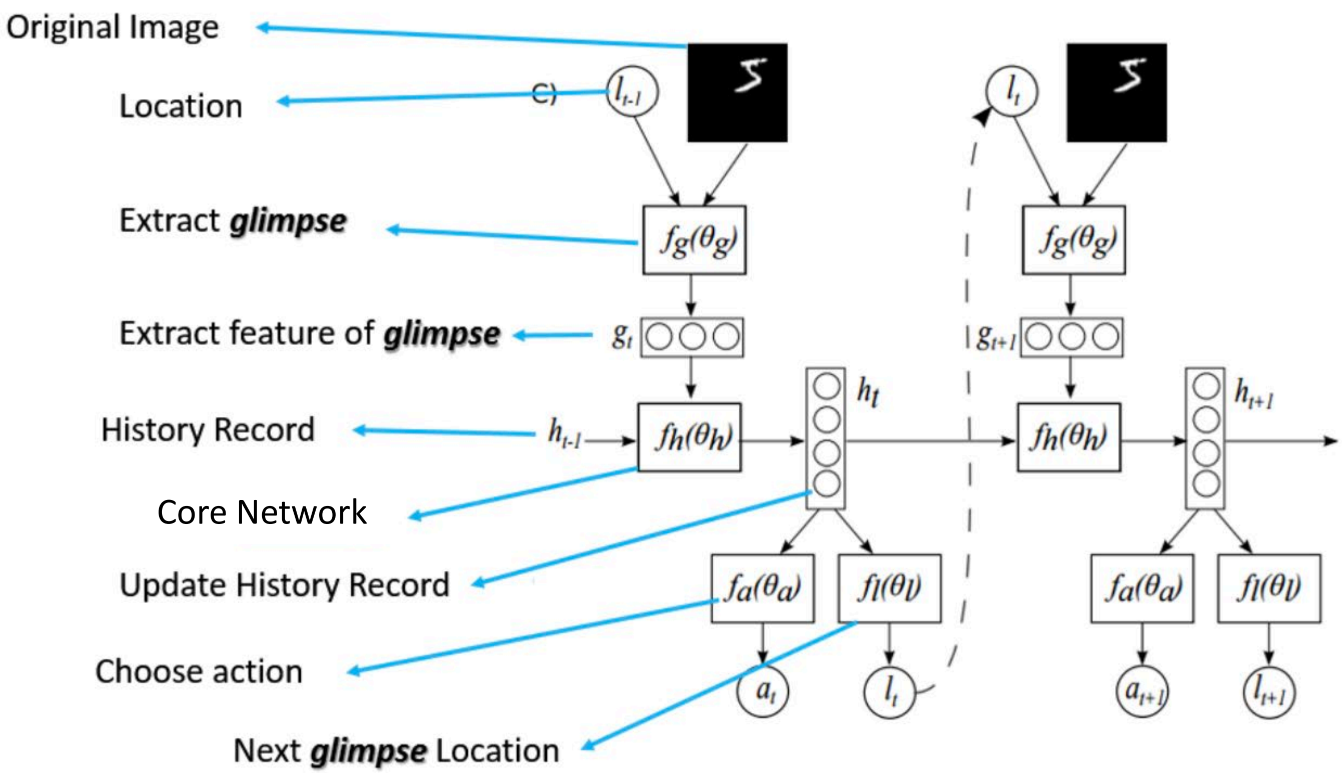
Glimpse network: given location l_{t-1} and image x_t, use the glimpse sensor to extract retina representation, which is mapped into a joint hidden space.



RNN-based model architecture: the core network takes the glimpse representation as input with the hidden state vector from the prevision step, and outputs the new hidden state resulting in **location** and **action** networks to predict the next location to attend and the associated action.

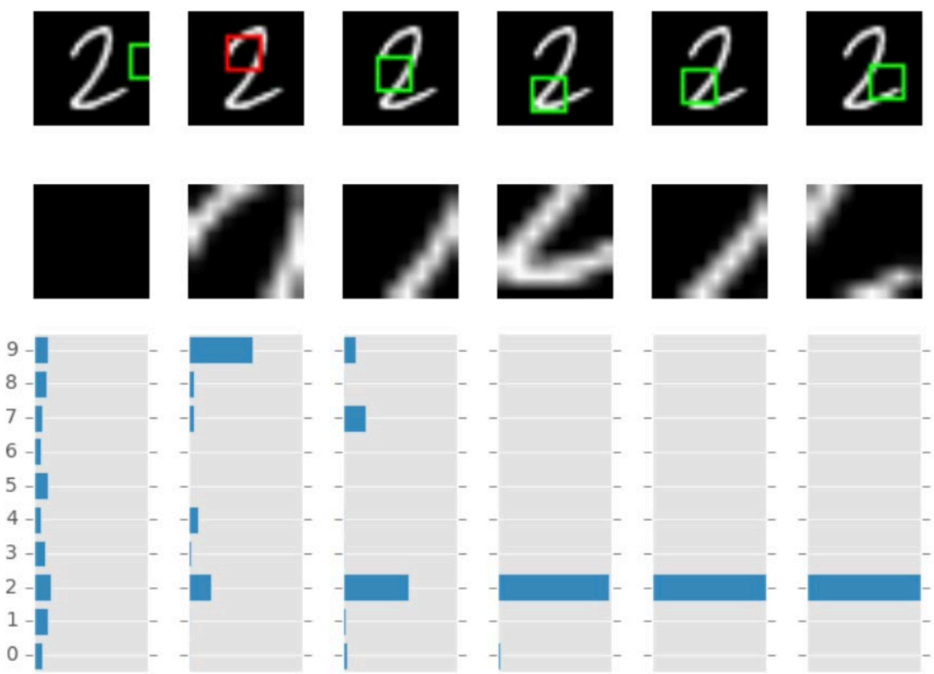
Mnih et al, “Recurrent Models of Visual Attention”, NIPS ‘14

Architecture: RNN with Attention Models

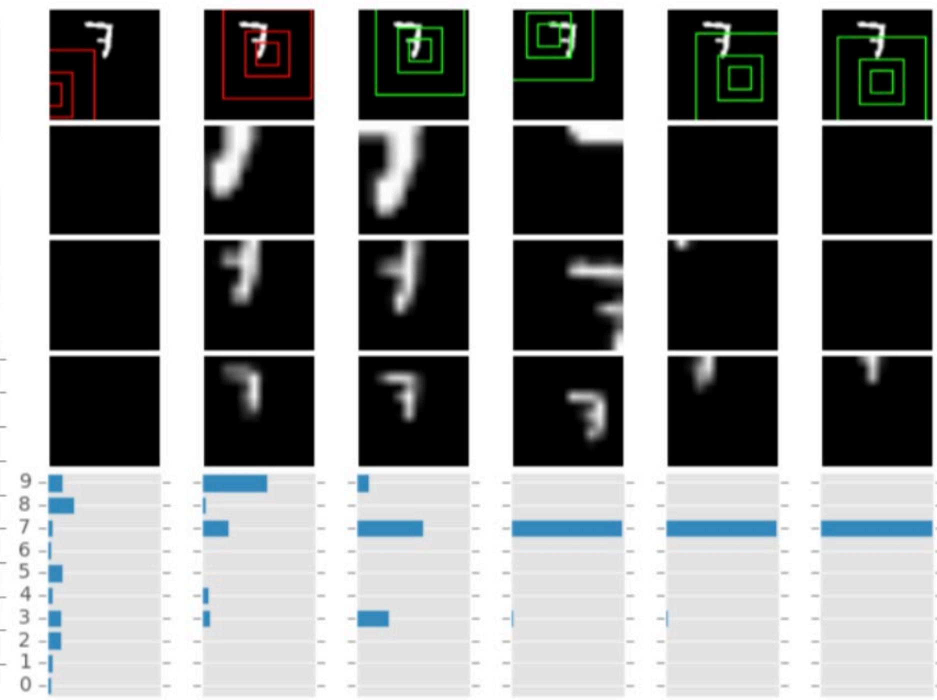


Example Results

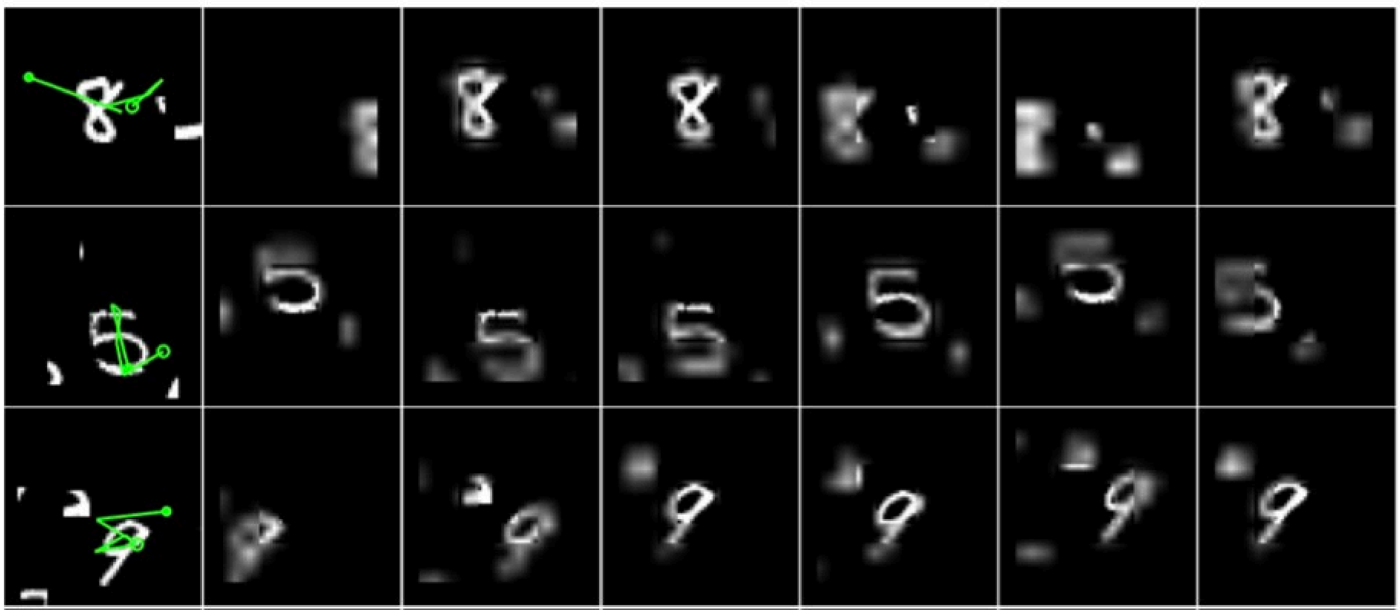
- Original MNIST



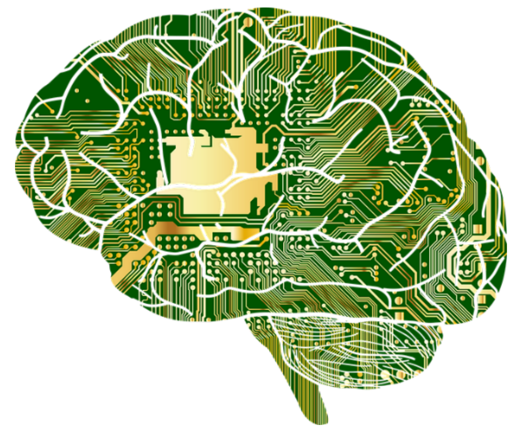
- Translated MNIST



Example: Actual Glimpse Path



What We’ve Covered Today…



- Transfer Learning
  - Visual Synthesis – Style Transfer
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Selected Models for Sequence-to-Sequence Learning
  - Attention in RNN
- Next week: Transformer