Deep Learning for Computer Vision

113-1/Fall 2024

https://cool.ntu.edu.tw/courses/41702 (NTU COOL)

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

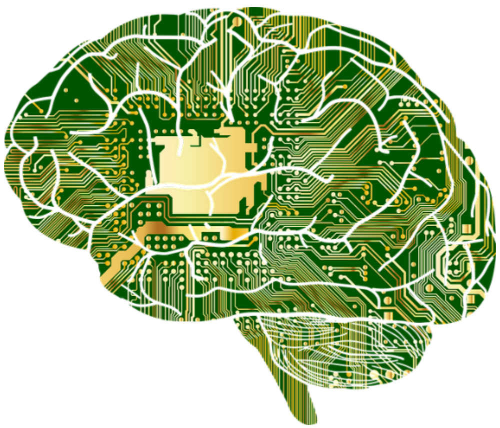
Yu-Chiang Frank Wang 王鈺強, Professor

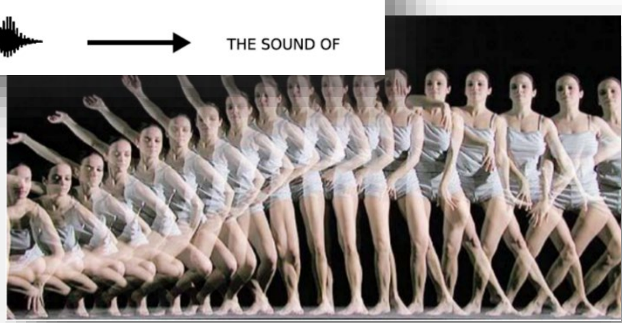
Dept. Electrical Engineering, National Taiwan University

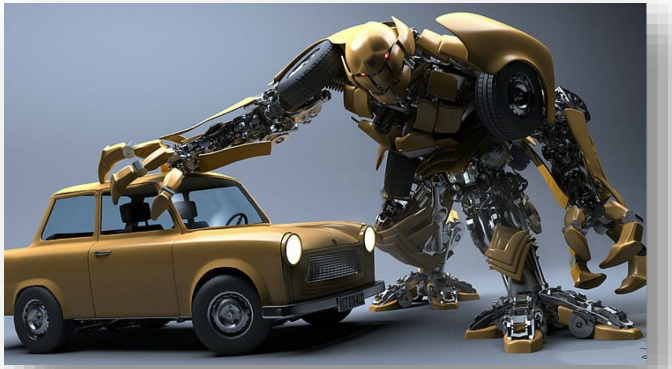
2024/10/15

What to Be Covered Today…

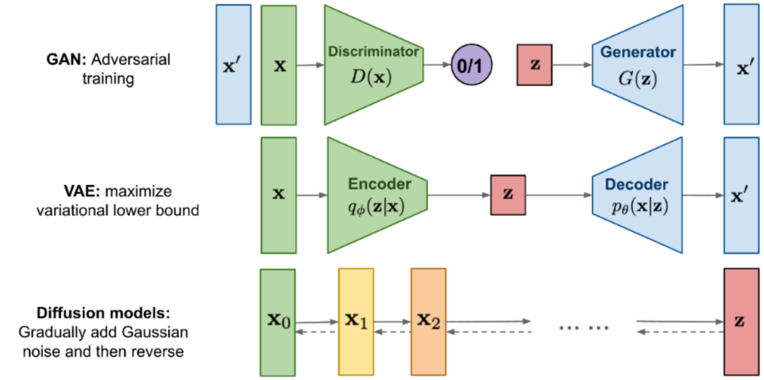
- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer (if time permits)



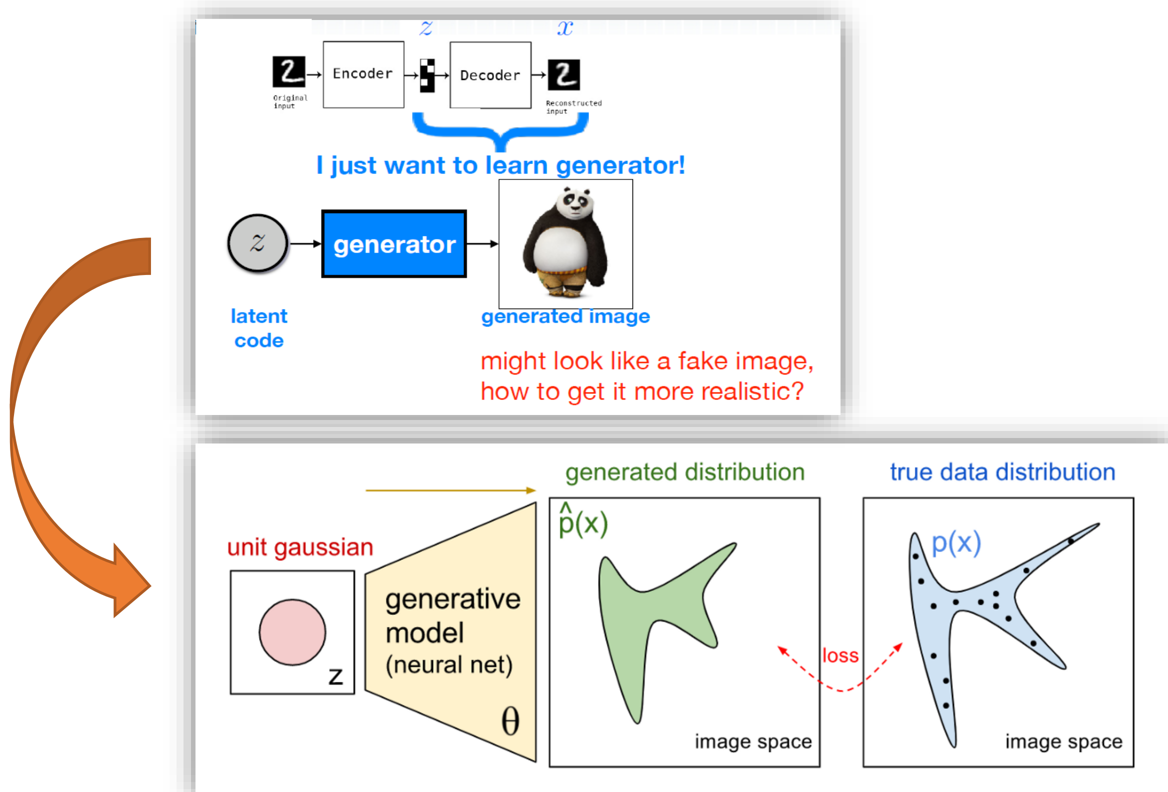




Recap: From VAE to GAN



- Remarks
  - We only need the decoder/generator in practice.
  - We prefer fast generation.
  - How do we know if the output images are sufficiently good?

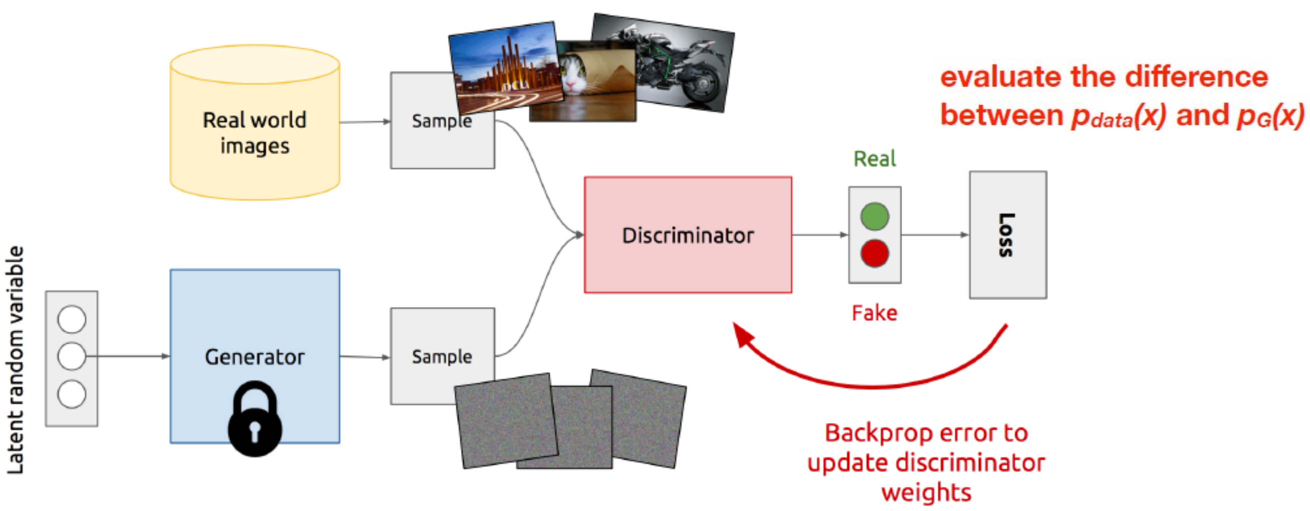


Slide credit: W. Chiu

GAN (cont'd)

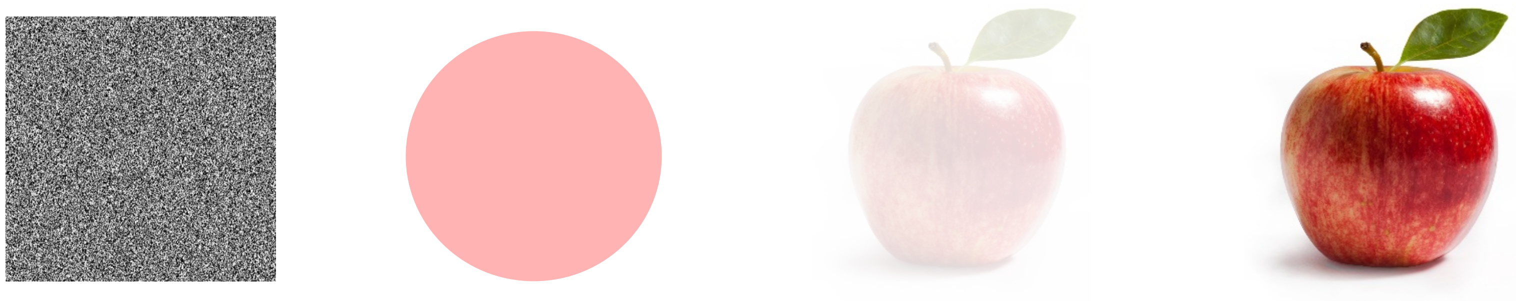
- Remarks
  - A function maps **normal distribution** $\boldsymbol{N(0, I)}$ to $\boldsymbol{P_{data}}$
  - How good we are in mapping P_g to P_{data}?
    - Train & ask the discriminator!
  - Conduct a two-player min-max game (see next slide for more details)

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



Recap: Problem #1 - Vanishing Gradients

- What Might Go Wrong?
  - GAN training is often unstable.
  - In other words, training might not converge properly.
  - The discriminator which we prefer is…



0 0.2 0.6 1

Energy-Based GAN

- Energy Function
  - Converting input data into scalar outputs, viewed as energy values
  - Desired configuration is expected to output low energy values & vice versa.
- Energy Function as Discriminator
  - Use of autoencoder; can be pre-trained!
  - Reconstruction loss outputs a range of values instead of binary logistic loss.
  - Empirically better convergence

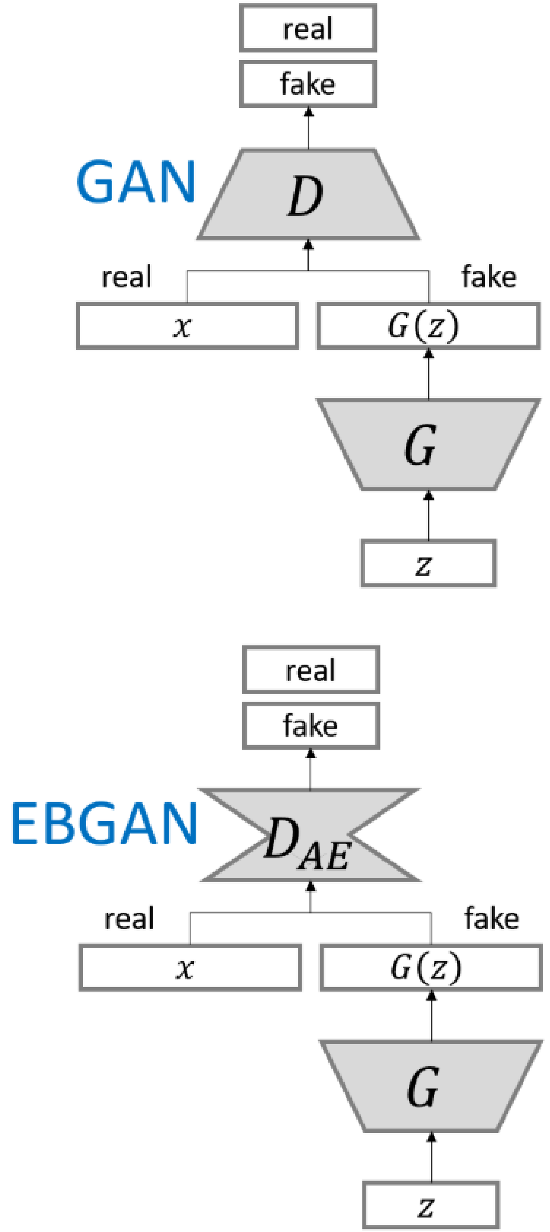
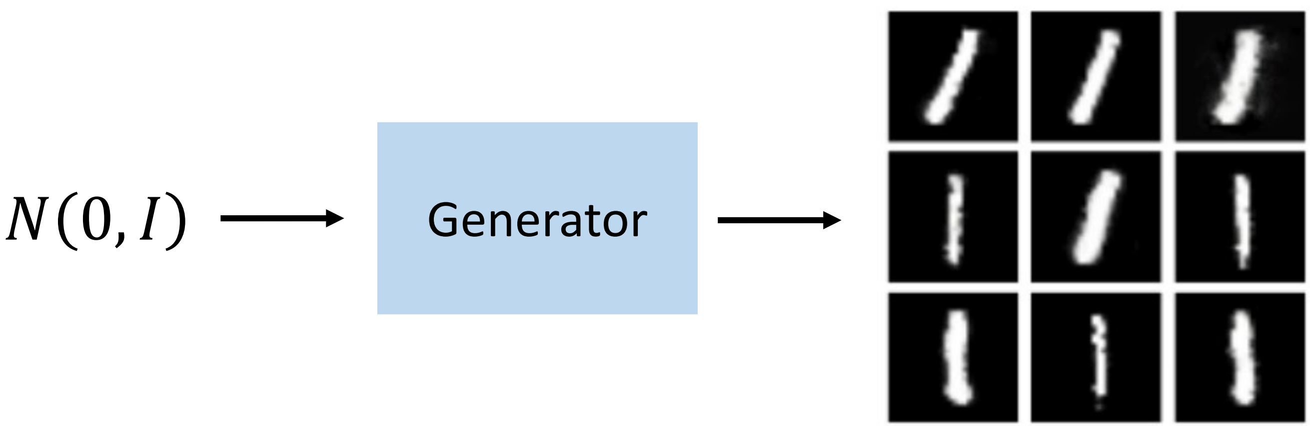


Photo credit: https://github.com/znxlwm/pytorch-generative-model-collections

Recap: Problem #2 - Mode Collapse

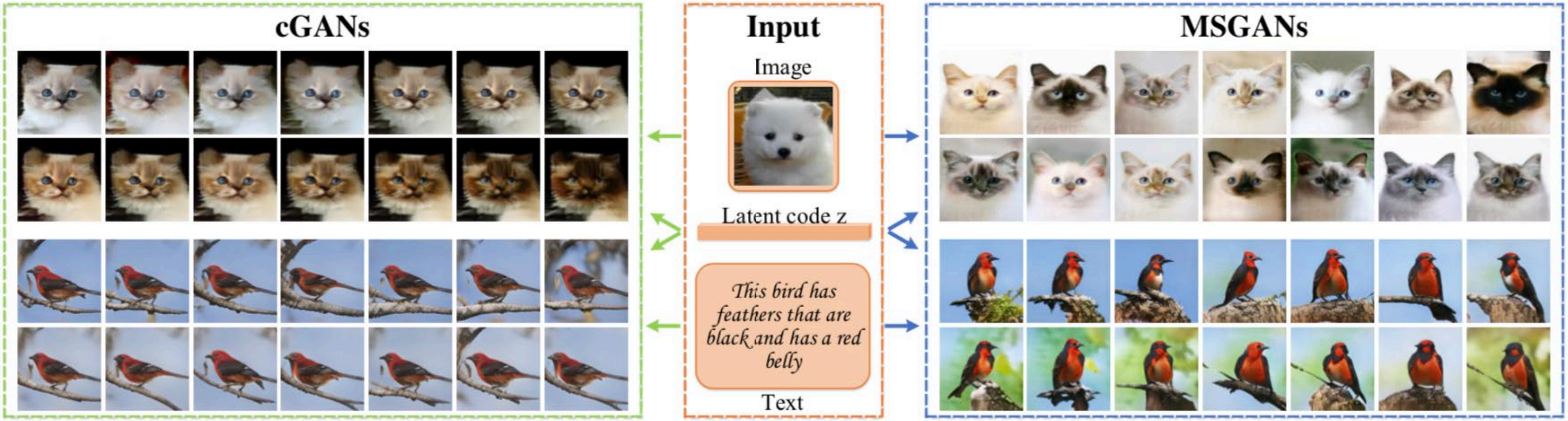
- Remarks
  - The generator only outputs a limited number of image variants regardless of the inputs.



Mode collapse!

MSGAN

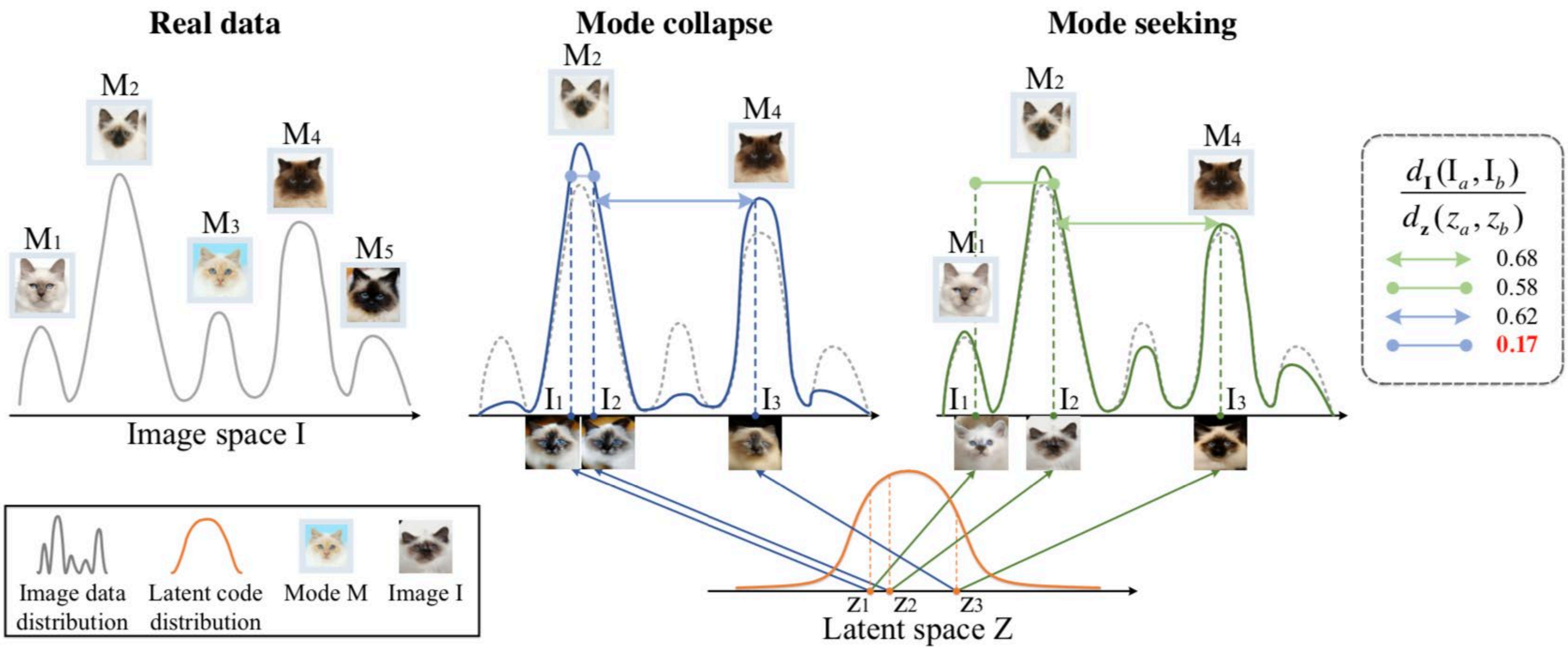
- To address the **mode collapse** issue by conditional GANs
- Mode Seeking Generative Adversarial Networks
  for Diverse Image Synthesis
- With the goal of producing **diverse** image outputs.



Mao *et al.* " Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis. " *CVPR* 2019

MSGAN (cont'd)

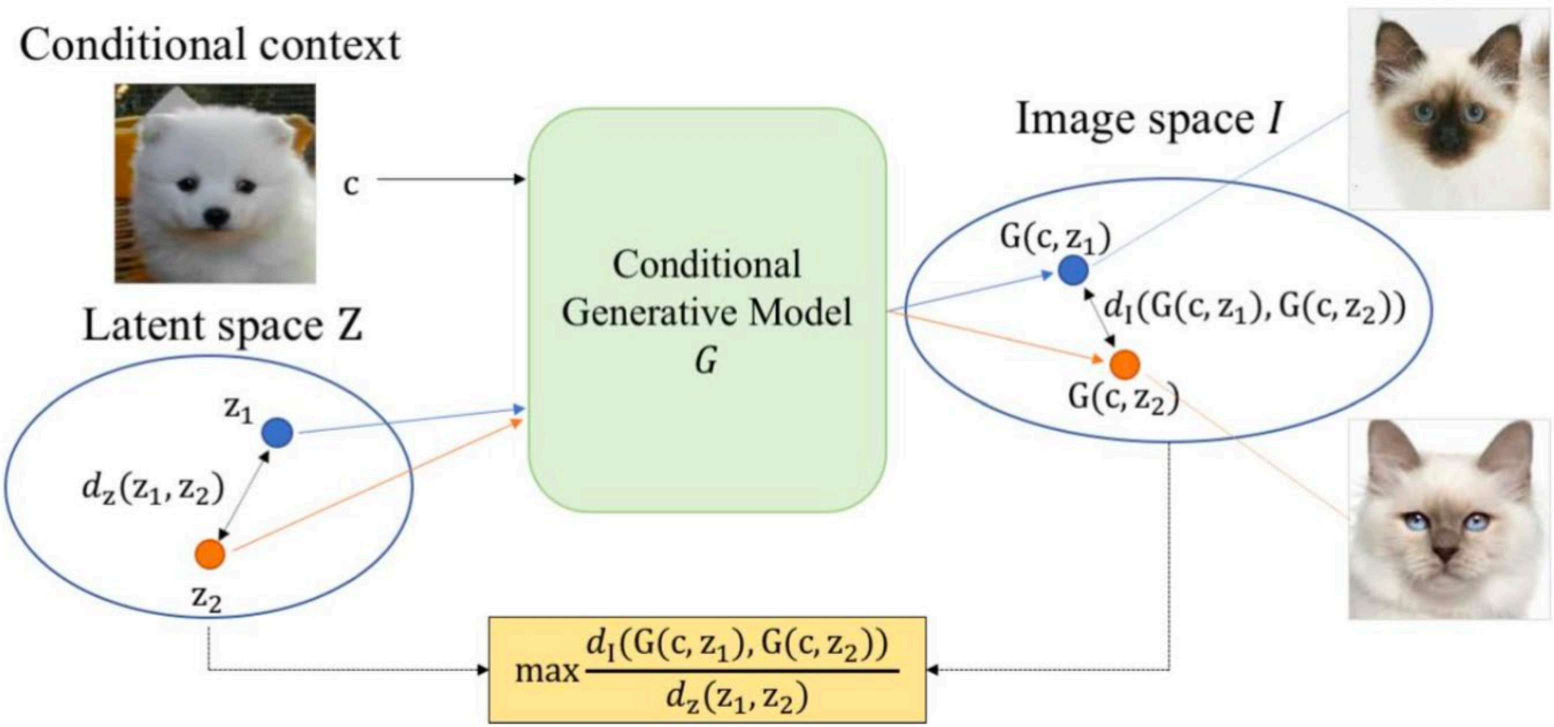
- Motivation (for unconditional GAN)



Mao *et al.* " Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis. " *CVPR* 2019

MSGAN (cont'd)

- Proposed Regularization (for conditional GAN)



Mao *et al.* " Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis. " *CVPR* 2019

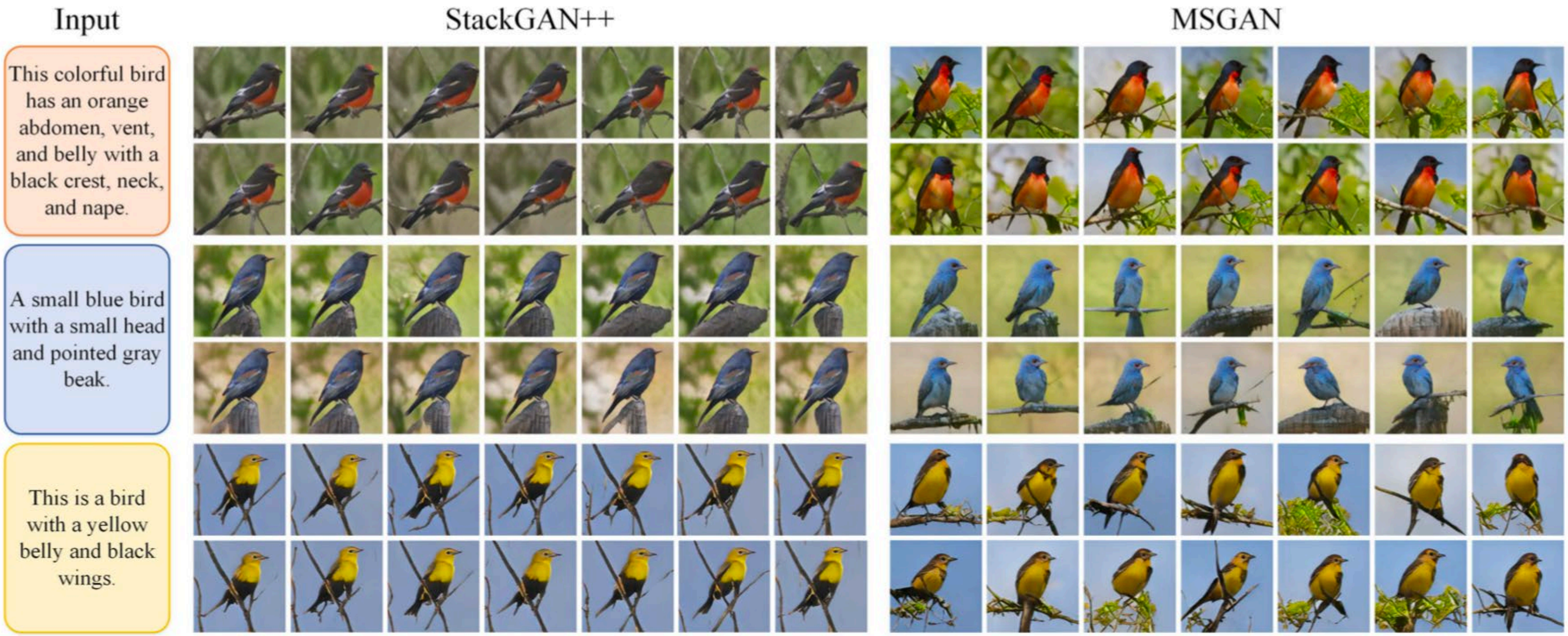
MSGAN (cont'd)

- Qualitative results
  - Conditioned on paired images

Mao *et al.* " Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis. " *CVPR* 2019

MSGAN (cont'd)

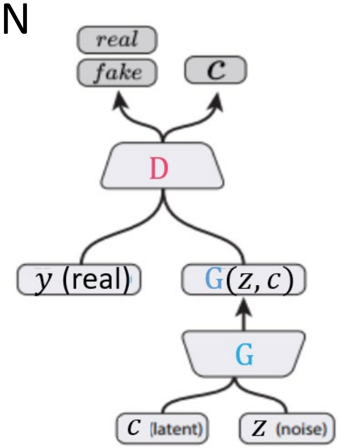
- Qualitative results
  - Conditioned on text (will talk about Vision & Language later this semester)



Mao *et al.* " Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis. " *CVPR* 2019

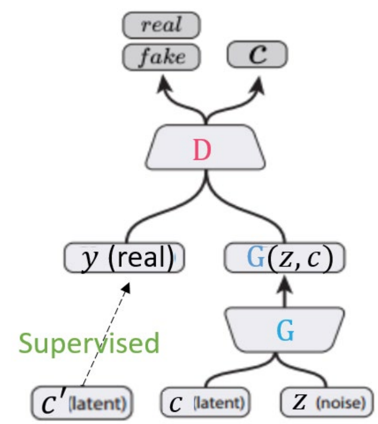
Recap: Conditional GAN

- Goal
  - Interpretable deep feature representation
  - Disentangle attribute of interest ***c*** from the derived latent representation ***z***
    - Unsupervised: InfoGAN
    - Supervised: AC-GAN



InfoGAN

Chen et al.
NIPS ’16



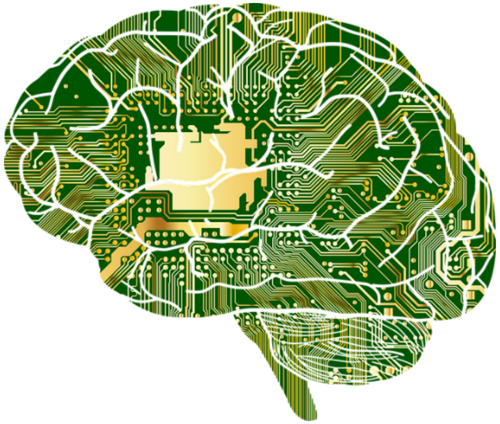
ACGAN

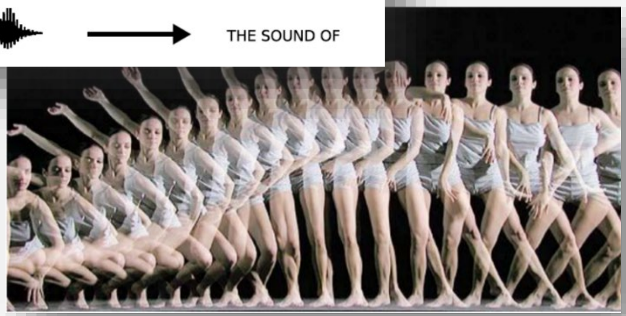
Odena et al.
ICML ’17

Chen et al., InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets., NIPS 2016.
Odena et al., Conditional image synthesis with auxiliary classifier GANs. ICML’17

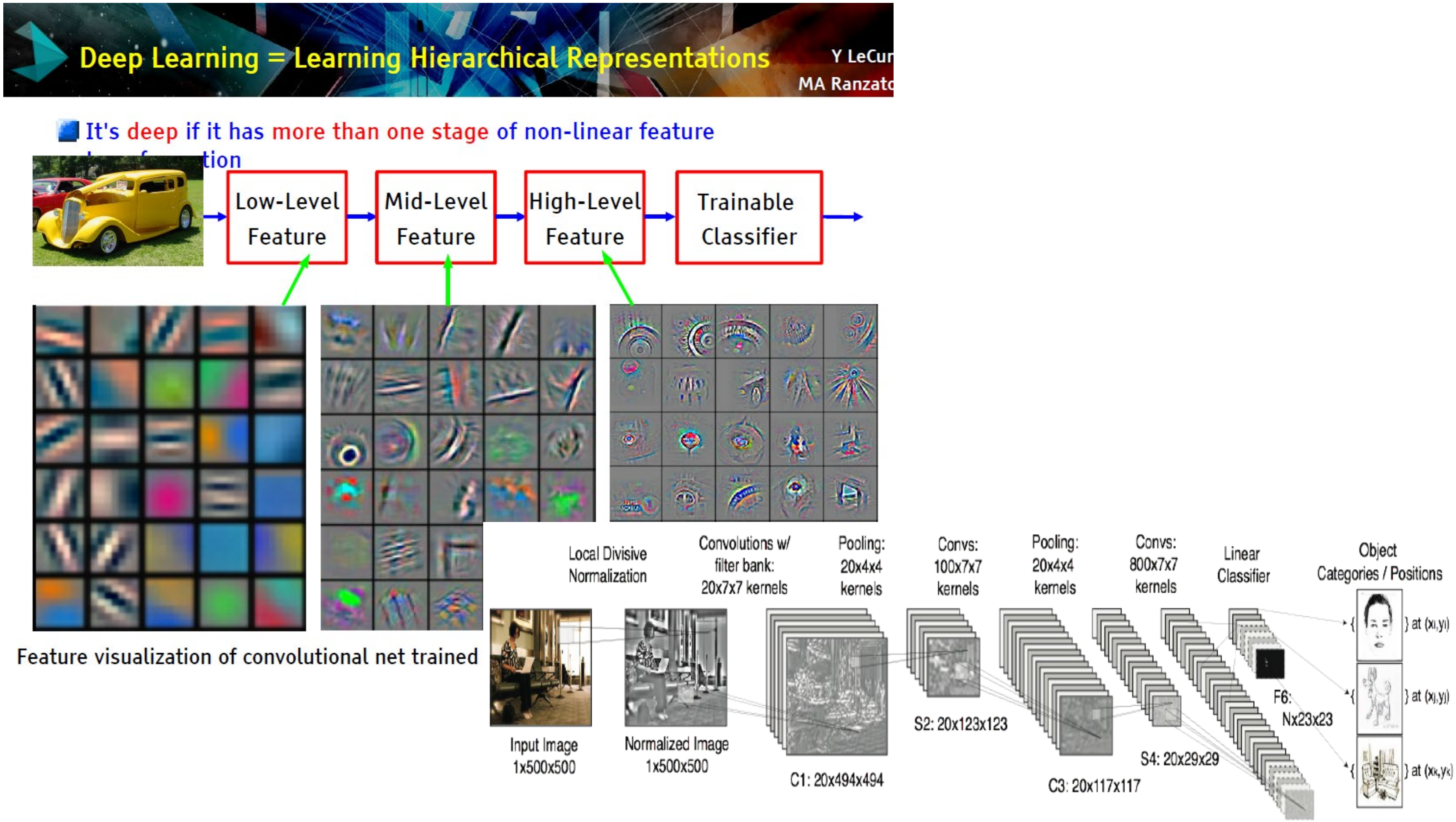
What to Be Covered Today…

- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer (if time permits)





Revisit of CNN for Visual Classification



LeCun & Ranzato, Deep Learning Tutorial, ICML 2013

(Traditional) Machine Learning vs. Transfer Learning

- Machine Learning
  - Collecting/annotating data is typically expensive.

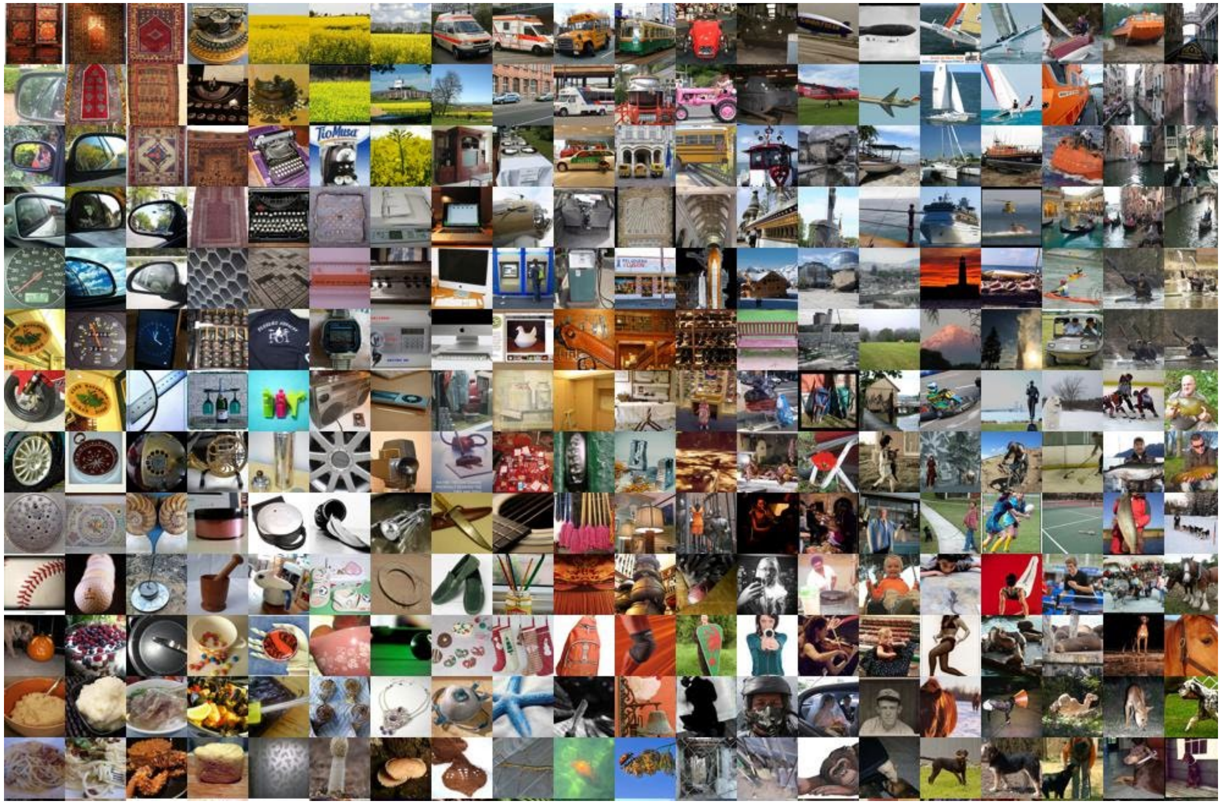
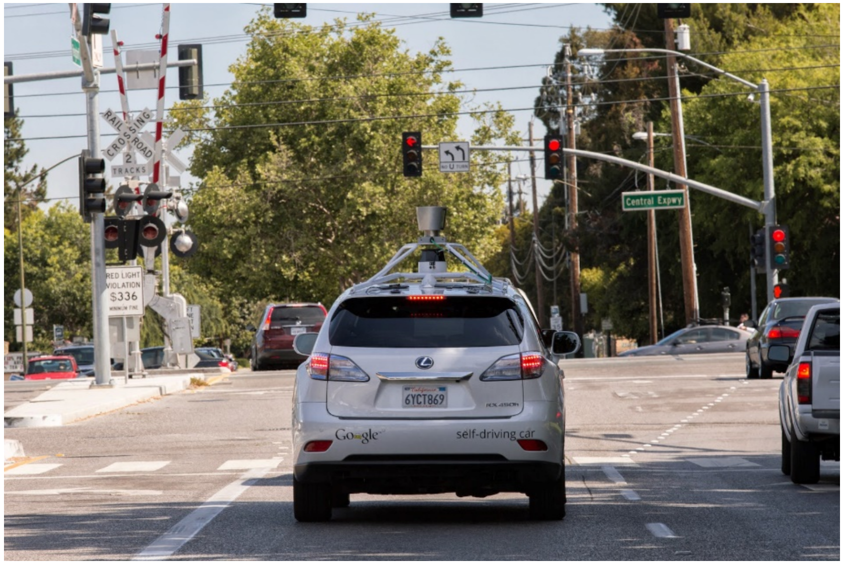


Image Credit: A. Karpathy

Transfer Learning: What, When, and Why? (cont'd)

- A More Practical Example



https://techcrunch.com/2017/02/08/udacity-open-sources-its-self-driving-car-simulator-for-anyone-to-use/
https://googleblog.blogspot.tw/2014/04/the-latest-chapter-for-self-driving-car.html

Domain Adaptation in Transfer Learning

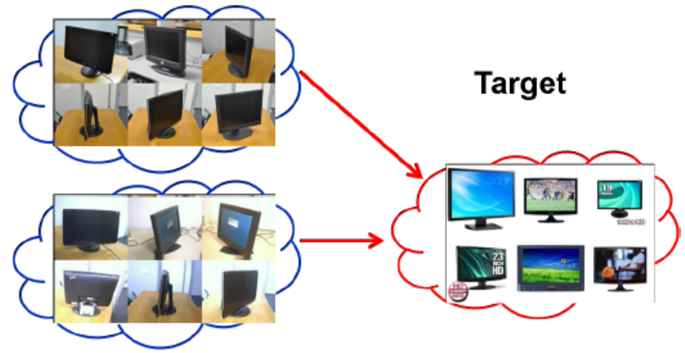
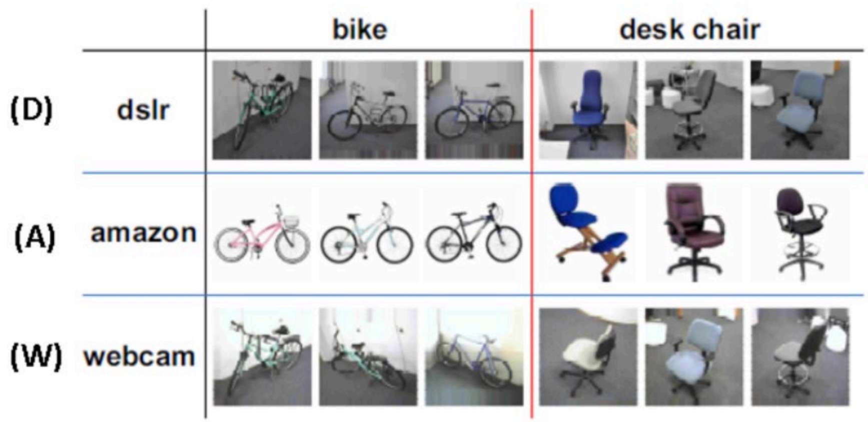


Image: Courtesy to S.J. Pan

- What's DA?
  - Leveraging info source to target domains, so that the same learning task across domains (or particularly in the target domain) can be addressed.
  - Typically all the source-domain data are labeled.
- Settings
  - Semi-supervised DA: few target-domain data are with labels.
  - Unsupervised DA: no label info available in the target-domain. (shall we address supervised DA?)
  - Imbalanced DA: fewer classes of interest in the target domain
  - Homogeneous vs. heterogeneous DA

Deep Feature is Sufficiently Promising.

- DeCAF
  - Leveraging an auxiliary large dataset to train CNN.
  - The resulting features exhibit sufficient representation ability.
  - Supporting results on Office+Caltech datasets, etc.

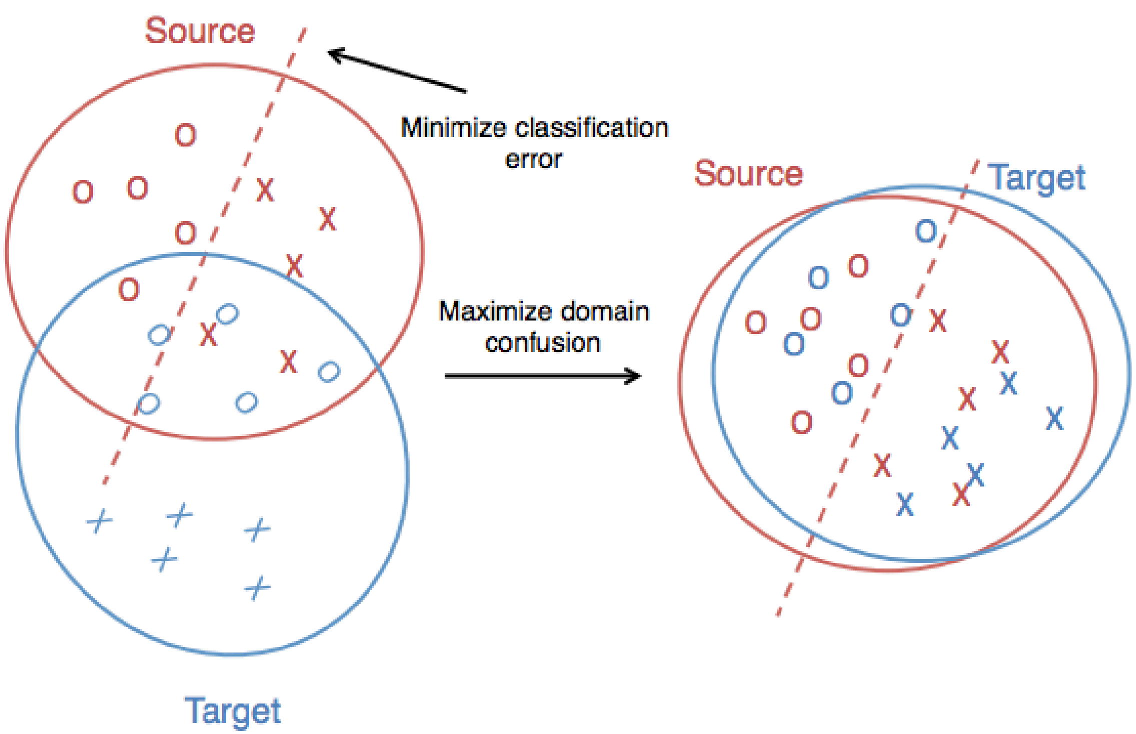


Feature	SURF											$Decaf_6$				
data	Raw	SA	SDA	GFK	TCA	JDA	TJM	SCA	JGSA primal	JGSA linear	JGSA RBF	JDA	OTGL	JGSA primal	JGSA linear	JGSA RBF
A→D	35.67	33.76	33.76	40.13	33.76	39.49	45.22	39.49	**47.13**	45.86	45.22	81.53	85.00	**88.54**	85.35	84.71
A→W	31.19	33.22	30.85	36.95	36.27	37.97	42.03	34.92	45.76	**49.49**	45.08	80.68	83.05	81.02	**84.75**	80.00
D→A	28.29	**39.87**	38.73	28.71	31.00	33.09	32.78	31.63	38.00	36.01	38.73	91.96	**92.31**	91.96	92.28	91.96
D→W	83.73	76.95	76.95	80.34	86.10	89.49	85.42	84.41	91.86	91.86	**93.22**	99.32	96.29	**99.66**	98.64	98.64
W→A	31.63	39.25	39.25	27.56	28.91	32.78	29.96	29.96	39.87	**41.02**	40.81	90.71	90.62	90.71	**91.44**	91.34
W→D	84.71	75.16	75.80	85.35	89.17	89.17	89.17	87.26	**90.45**	**90.45**	88.54	**100**	96.25	**100**	**100**	**100**

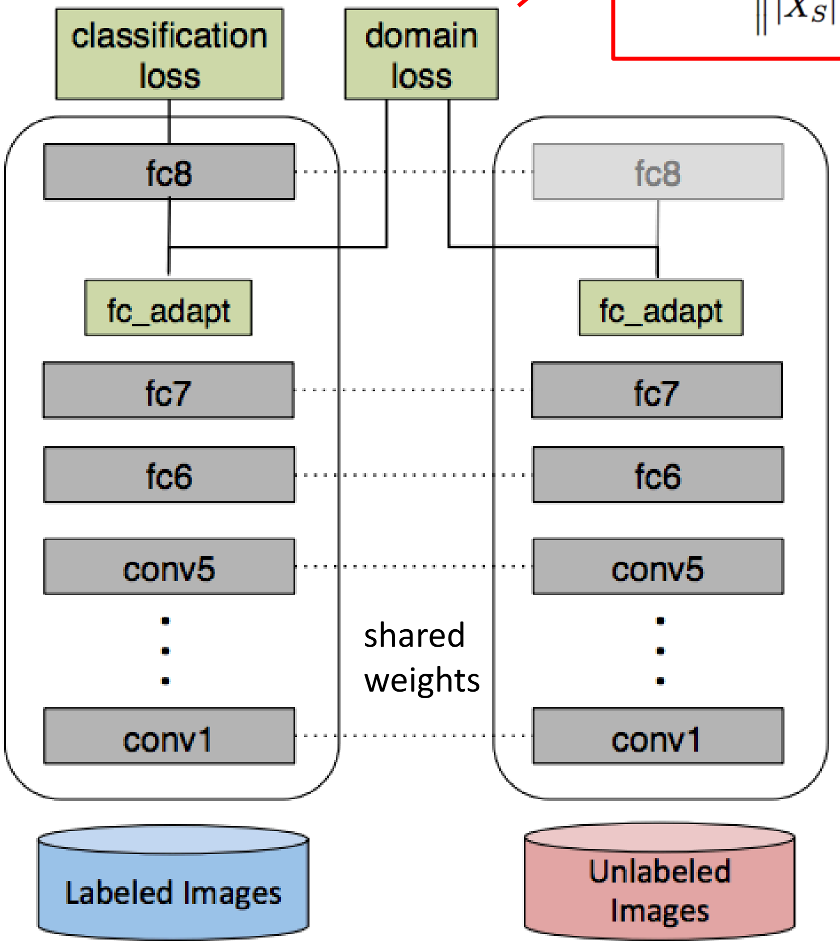
Donahue et al., DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, ICML 2014

Deep Domain Confusion (DDC)

- Deep Domain Confusion: Maximizing for Domain Invariance
  - Tzeng et al., arXiv: 1412.3474, 2014



Deep Domain Confusion (DDC)



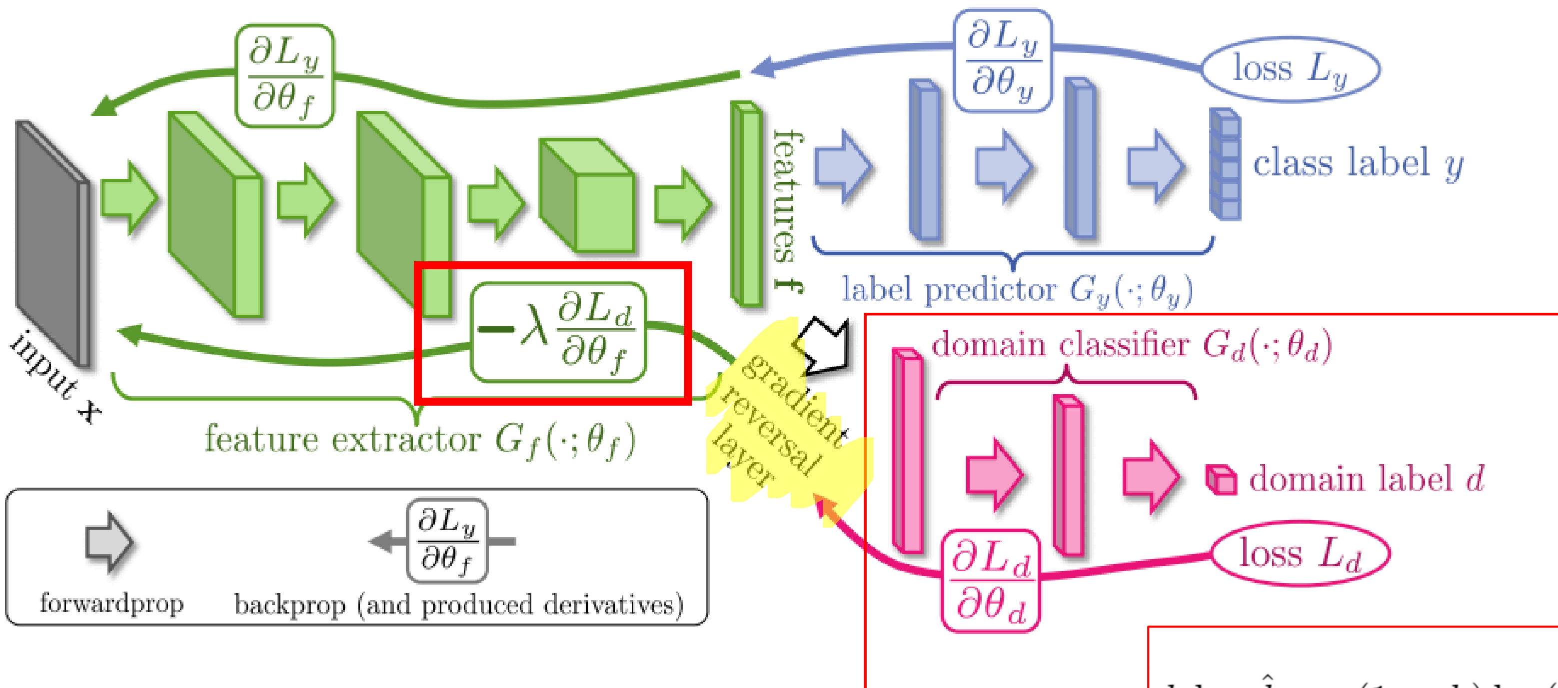
$$\mathrm{MMD}(X_S, X_T) = \left\| \frac{1}{|X_S|} \sum_{x_s \in X_S} \phi(x_s) - \frac{1}{|X_T|} \sum_{x_t \in X_T} \phi(x_t) \right\|$$

✓ Minimize classification loss:

$$\mathcal{L} = \mathcal{L}_C(X_L, y) + \lambda \, \mathrm{MMD}^2(X_S, X_T)$$

Domain Confusion by Domain-Adversarial Training

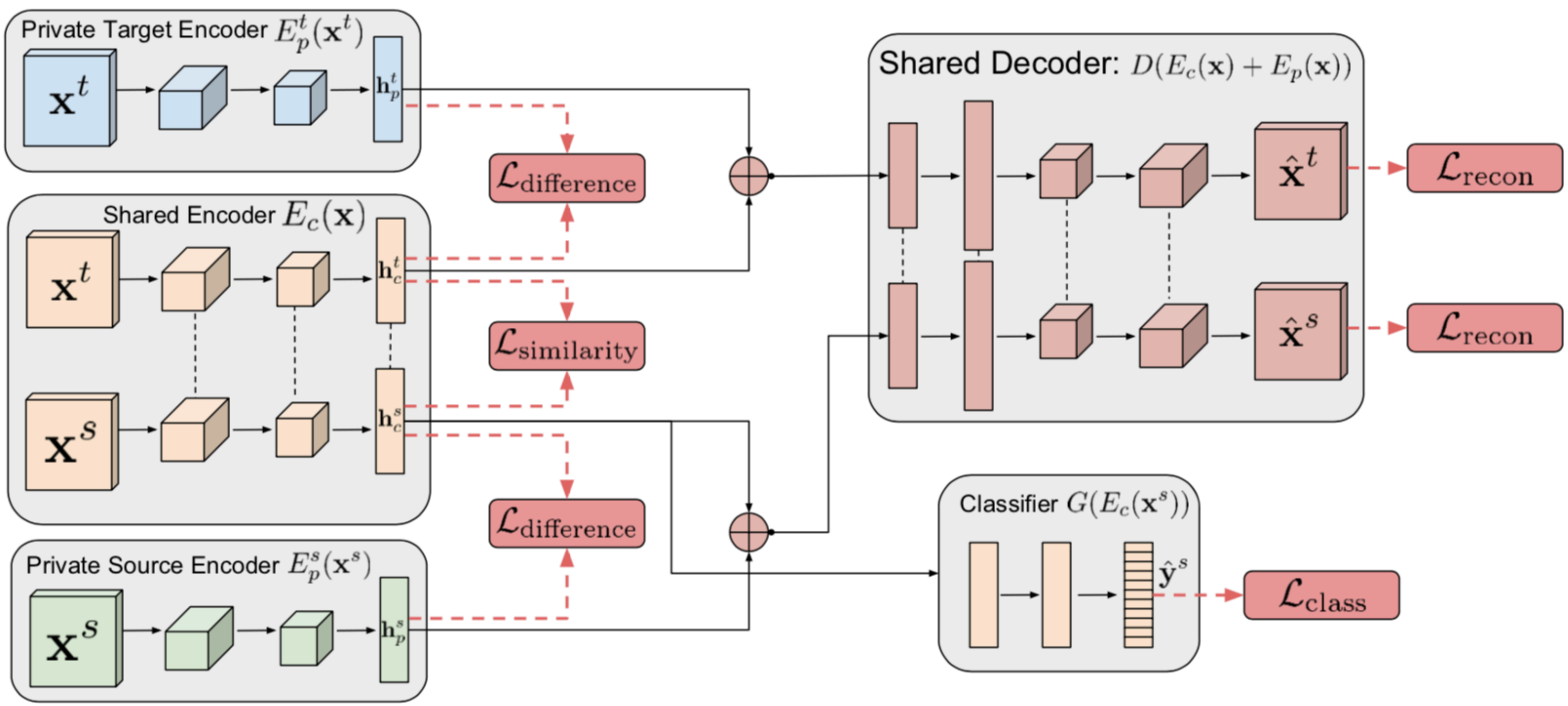
- Domain-Adversarial Training of Neural Networks (DANN)
  - Y. Ganin et al., ICML 2015
  - Maximize domain confusion = maximize domain classification loss
  - Minimize source-domain data classification loss
  - The derived feature f can be viewed as a disentangled & domain-invariant feature.



$$d_i \log \hat{d}_i \; + \; (1 - d_i) \log(1 - \hat{d}_i)$$

Beyond Domain Confusion

- **Domain Separation Network (DSN)**
  - Bousmalis et al., NIPS 2016
  - Separate encoders for domain-invariant and domain-specific features
  - Private/common features are *disentangled* from each other.



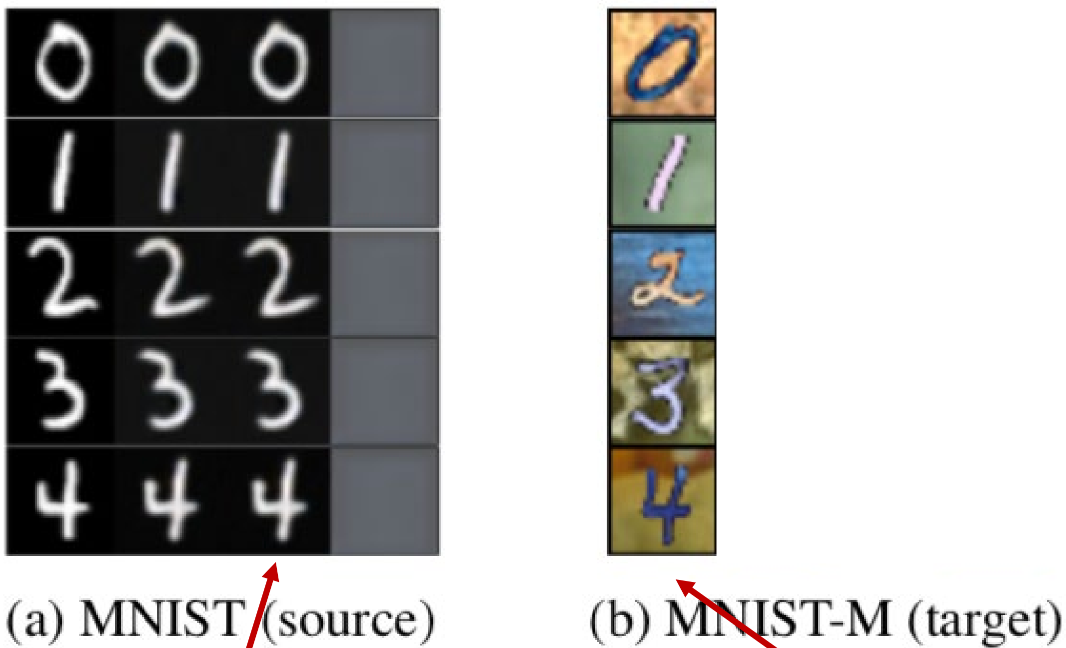
Beyond Domain Confusion

- Domain Separation Network, NIPS 2016
    - Example results

Source-domain image X_s

Reconstruct private + shared features $D(E_c(x_s)+E_p(x_s))$

Reconstruct private feature $D(E_p(x_s))$



Reconstruct shared feature only $D(E_c(x_s))$

Target-domain image X_T

Beyond Domain Confusion

- Domain Separation Network, NIPS 2016
  - Example results

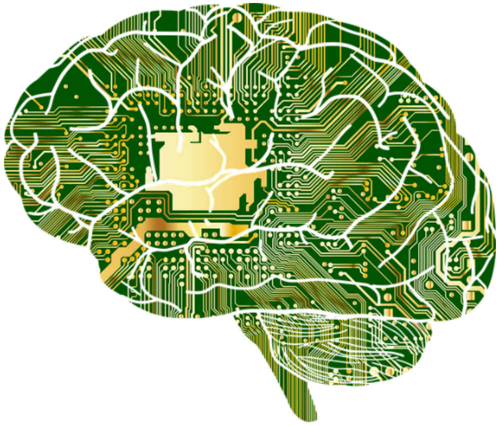
Source-domain image X_s

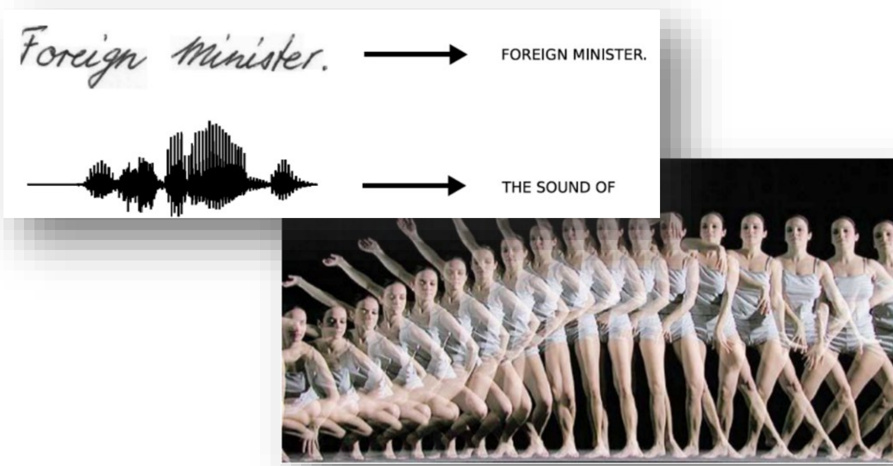
(a) MNIST (source) (b) MNIST-M (target)

Target-domain image X_T

What to Be Covered Today…

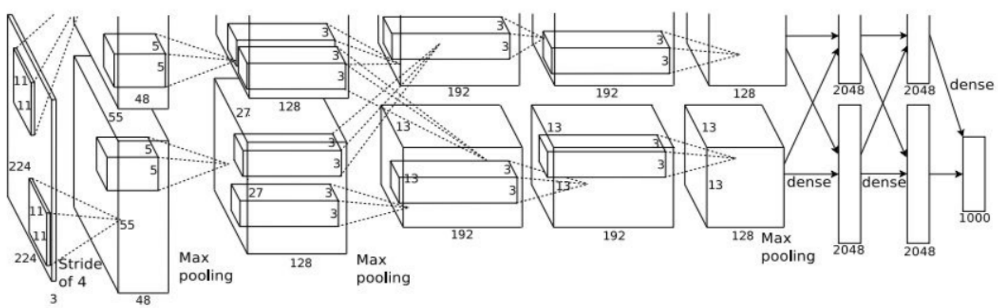
- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer (if time permits)





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

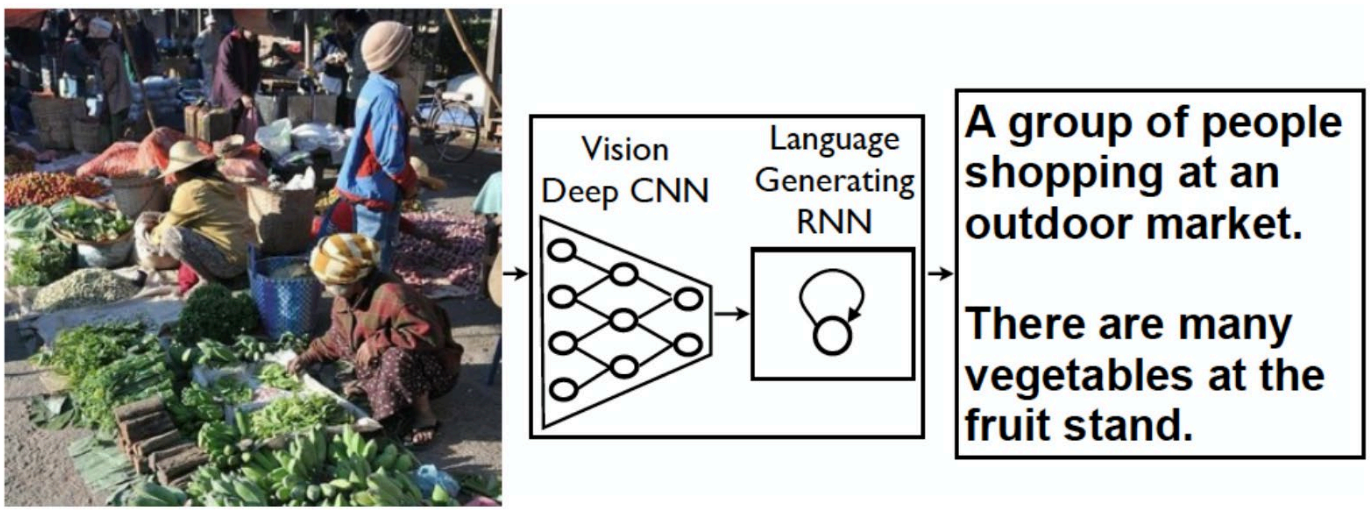
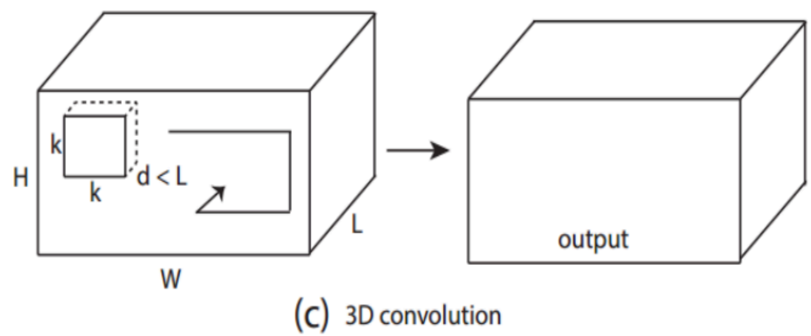


Figure from Vinyals et al, “Show and tell: A neural image caption generator”, CVPR 2015

How to Model Sequential Data?

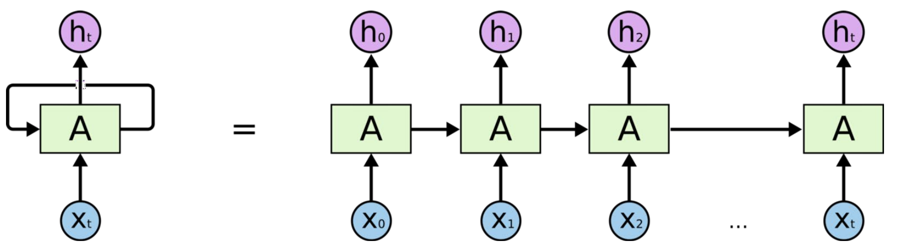
- Deep learning for sequential data
  - Possible solution: 3D convolution neural networks



3D convolution

How to Model Sequential Data?

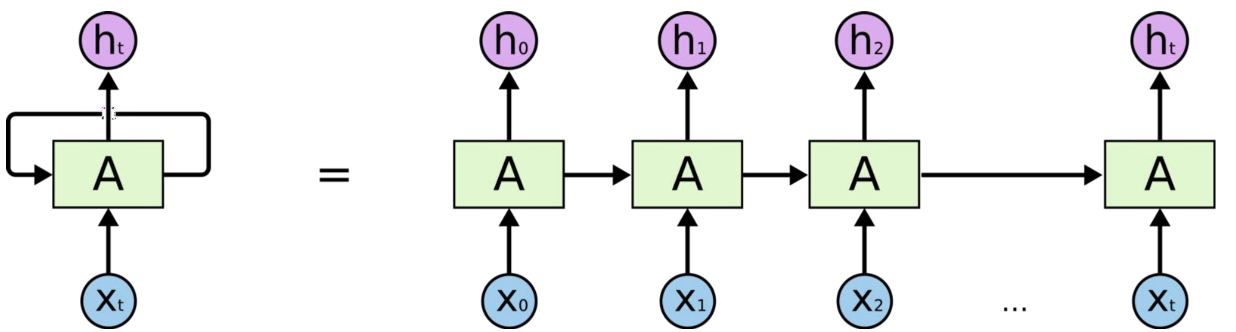
- Deep learning for sequential data
  - Possible solution: 3D convolution neural networks
  - Recurrent neural networks (RNN)



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/processed



Recurrence Formula

- Same function and parameters used at every time step:

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

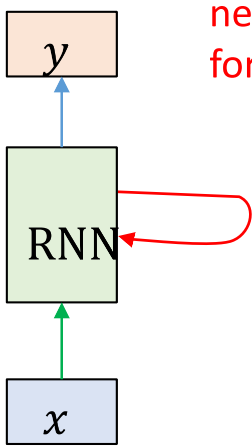
new state for time t

output vector at time t

function with parameters W

state at time t-1

input vector at time t

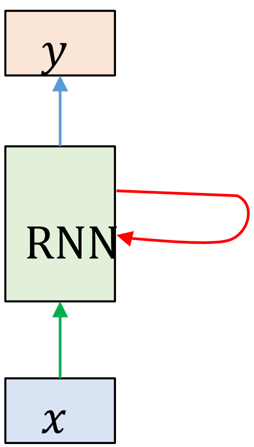


Recurrence Formula

- Same function and parameters used at every time step:

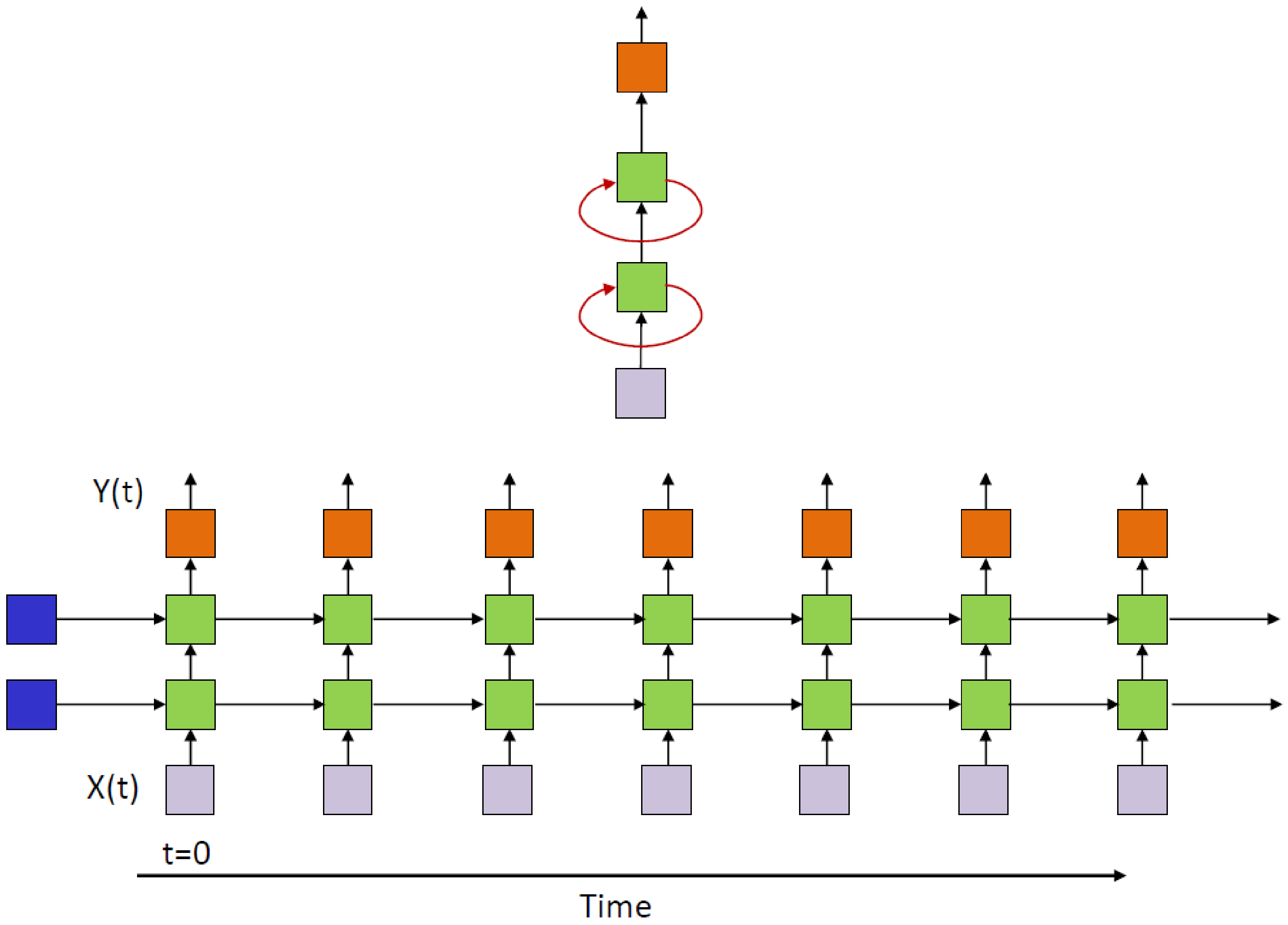
$$h_t, y_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

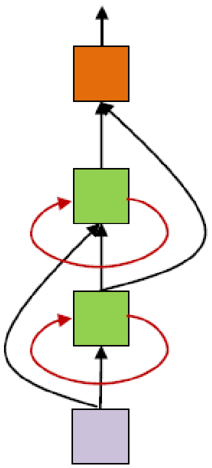


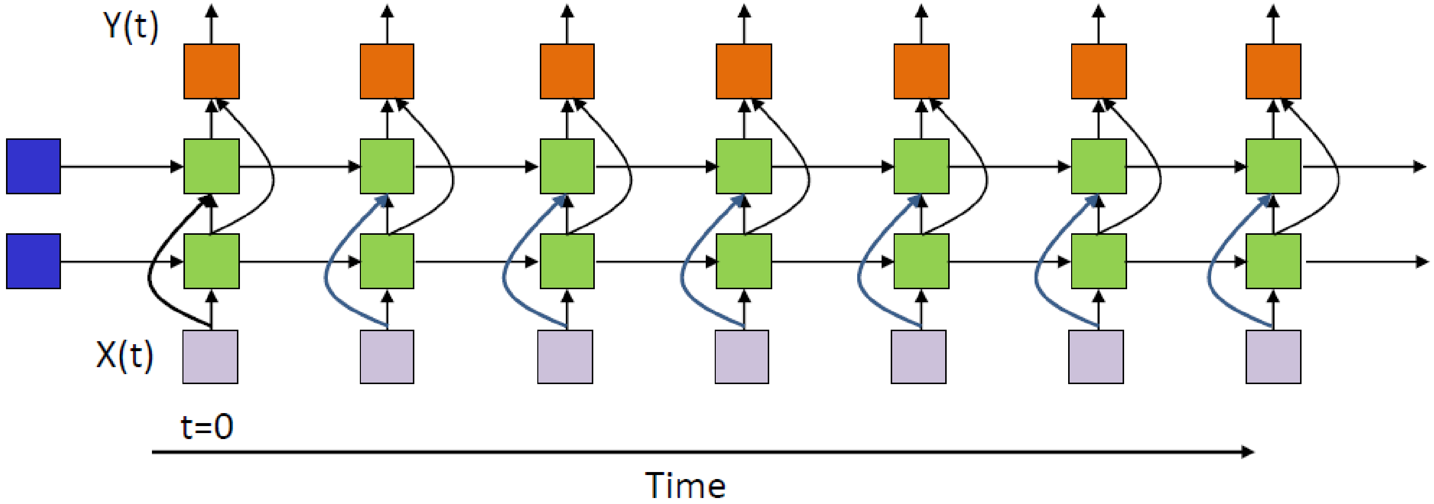
$$h_t = tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$
$$y_t = W_{hy} h_t$$

Multiple Recurrent Layers

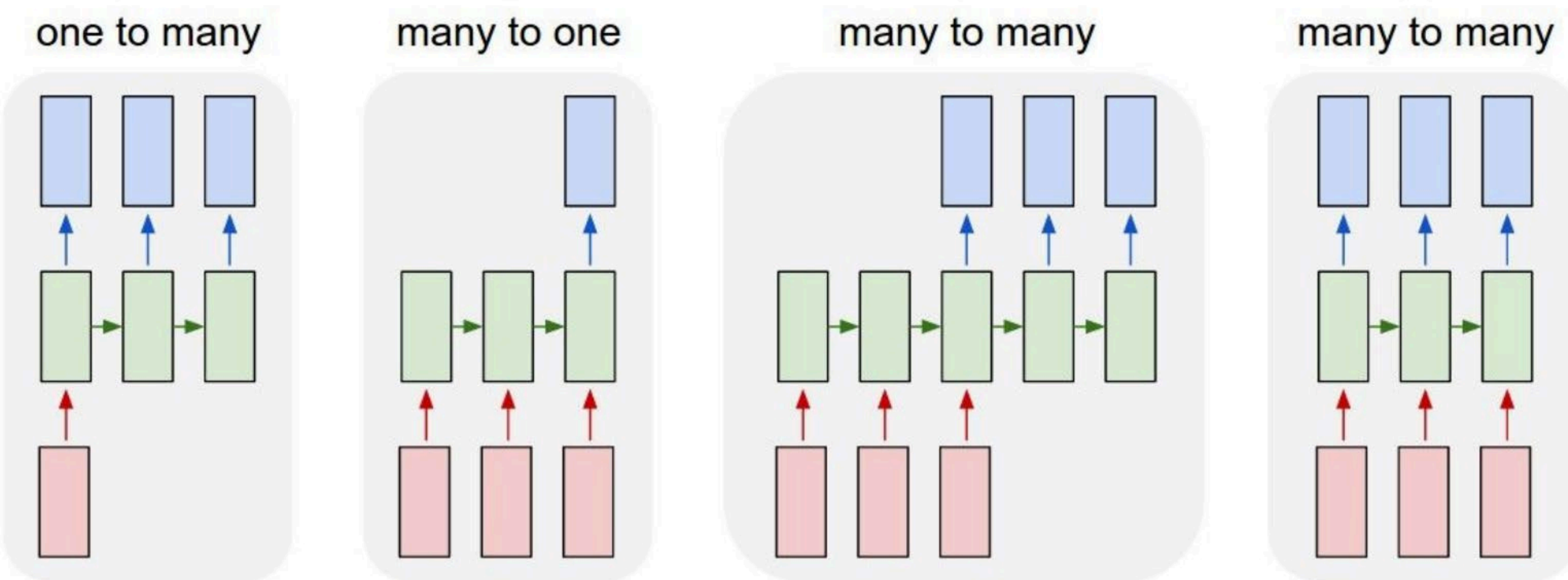


Multiple Recurrent Layers





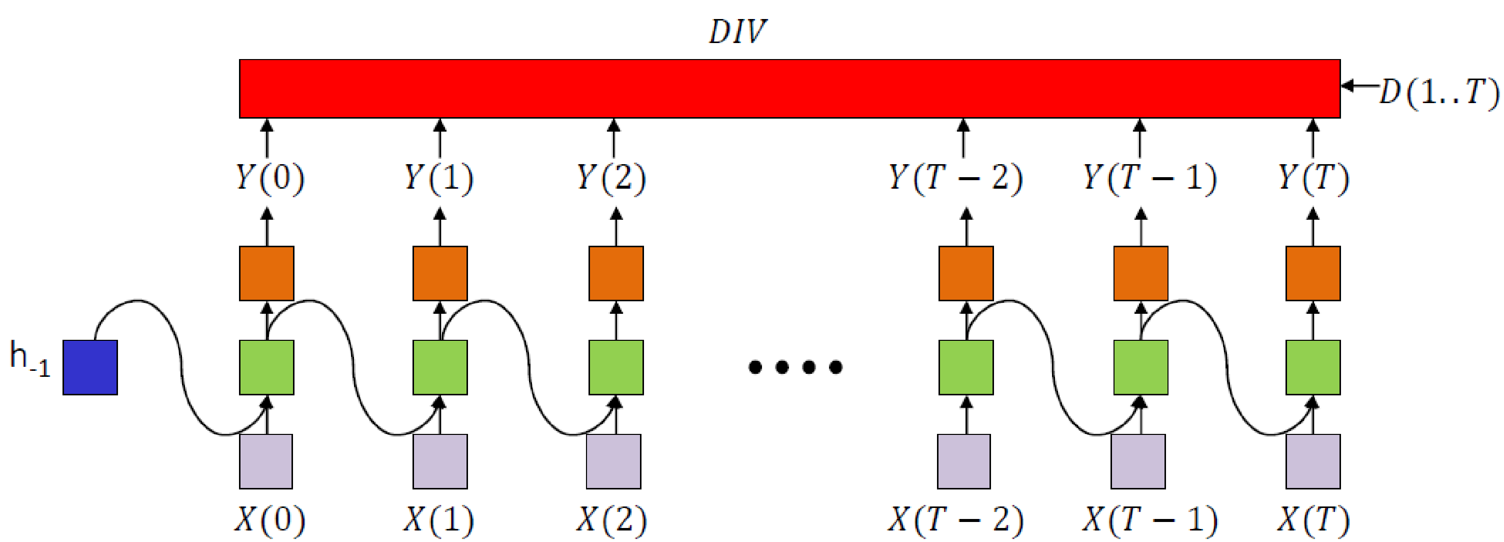
Sequence-to-Sequence Modeling



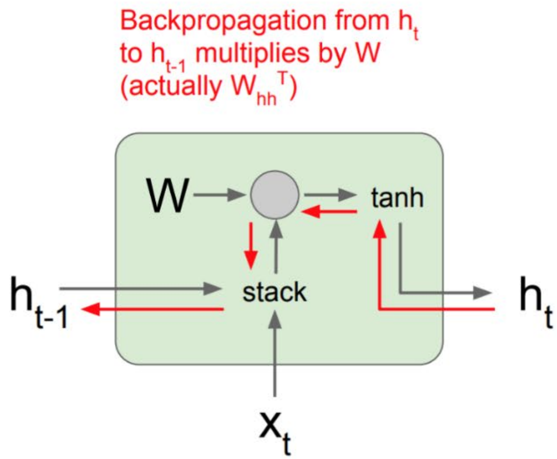
e.g., image caption e.g., action recognition e.g., video prediction e.g., video indexing

Training RNNs: *Back Propagation Through Time*

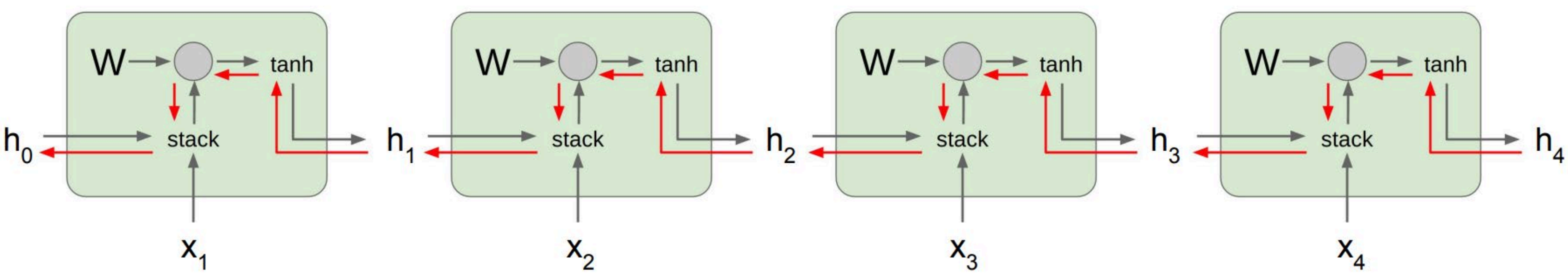
- Let's focus on one training instance.
- The divergence to be computed is between the sequence of outputs by the network and the desired output sequence.
- Generally, this is not just the sum of the divergences at individual times.



Back Propagation Through Time (BPTT)



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

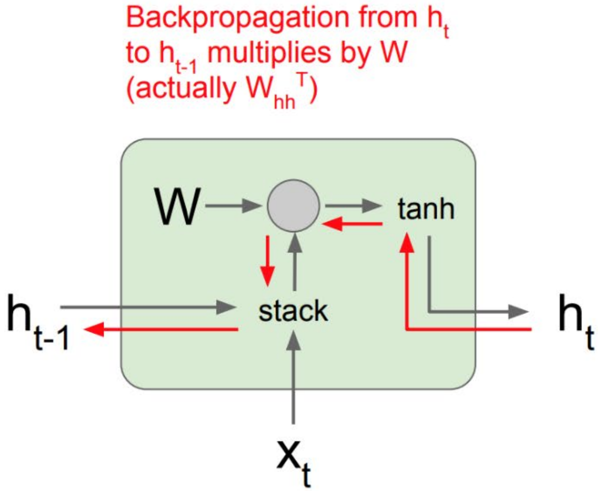


Computing gradient of h_0 involves many factors of W (and repeated tanh)

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

Gradient Vanishing & Exploding

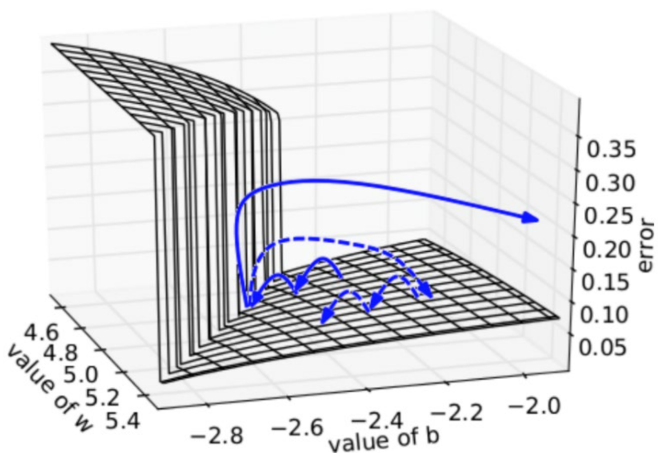
- Computing gradient involves many factors of W
  - Exploding gradients : Largest singular value > 1
  - Vanishing gradients : Largest singular value < 1

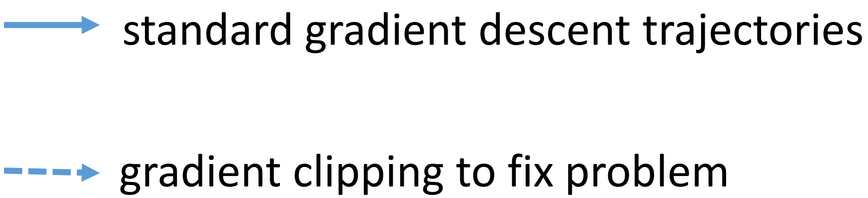


$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

Solutions...

- Gradients clipping : rescale gradients if too large






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

- How about vanishing gradients?
  - Change RNN architecture!

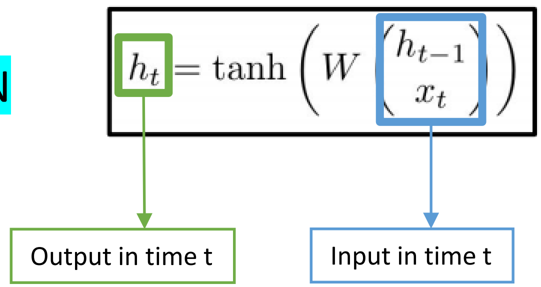
Variants of RNN

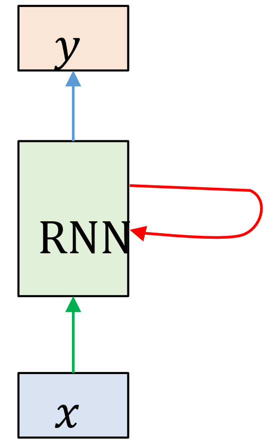
- Long Short-term Memory (LSTM) [*Hochreiter et al., 1997*]
  - Additional memory cell
  - Input/Forget/Output Gates
  - Handle gradient vanishing
  - Learn long-term dependencies

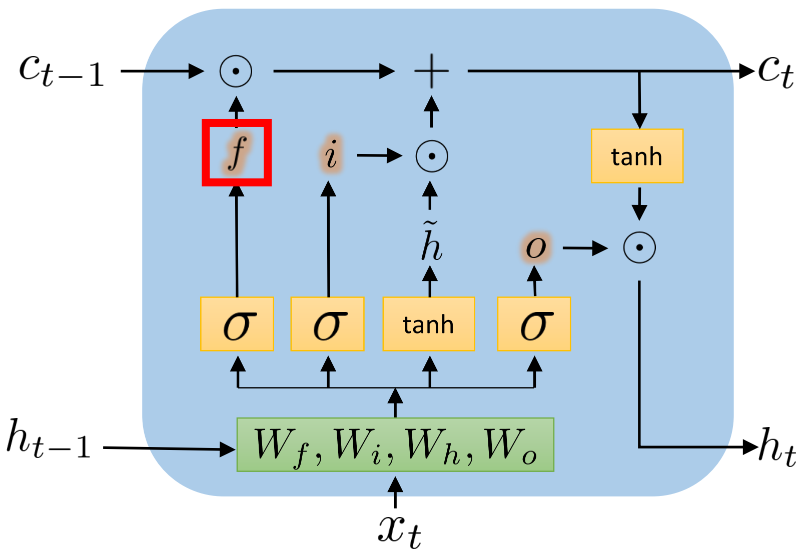
- Gated Recurrent Unit (GRU) [*Cho et al., EMNLP 2014*]
  - Similar to LSTM
    - handle gradient vanishing & learn long-term dependencies
  - No additional memory cell
  - Reset / Update Gates
  - Fewer parameters than LSTM
  - Comparable performance to LSTM [*Chung et al., NIPS Workshop 2014*]

Vanilla RNN, LSTM, & GRU

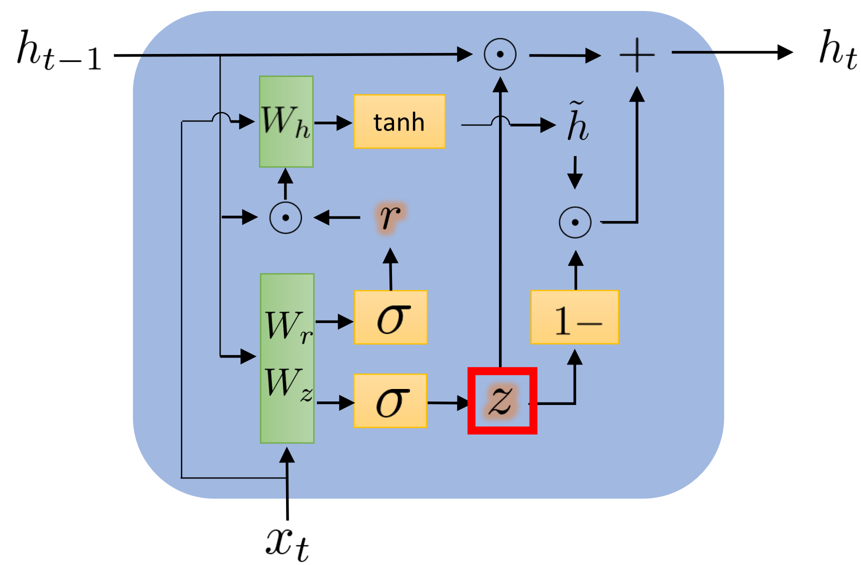
RNN







LSTM



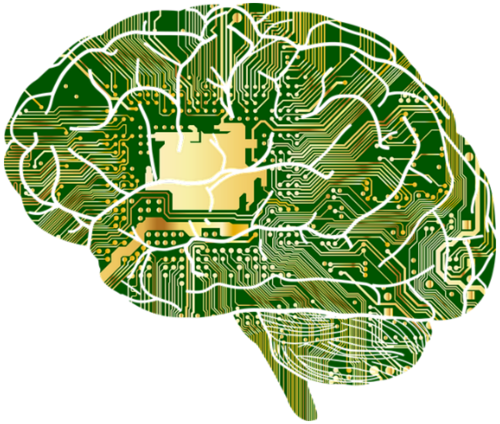
GRU

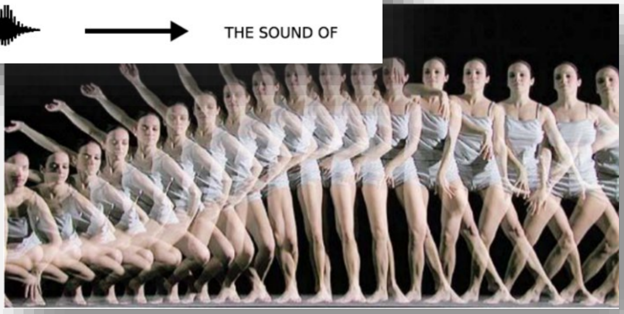
Vanilla RNN vs. LSTM vs. GRU

	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…

- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer (if time permits)





Sequence-to-Sequence Modeling

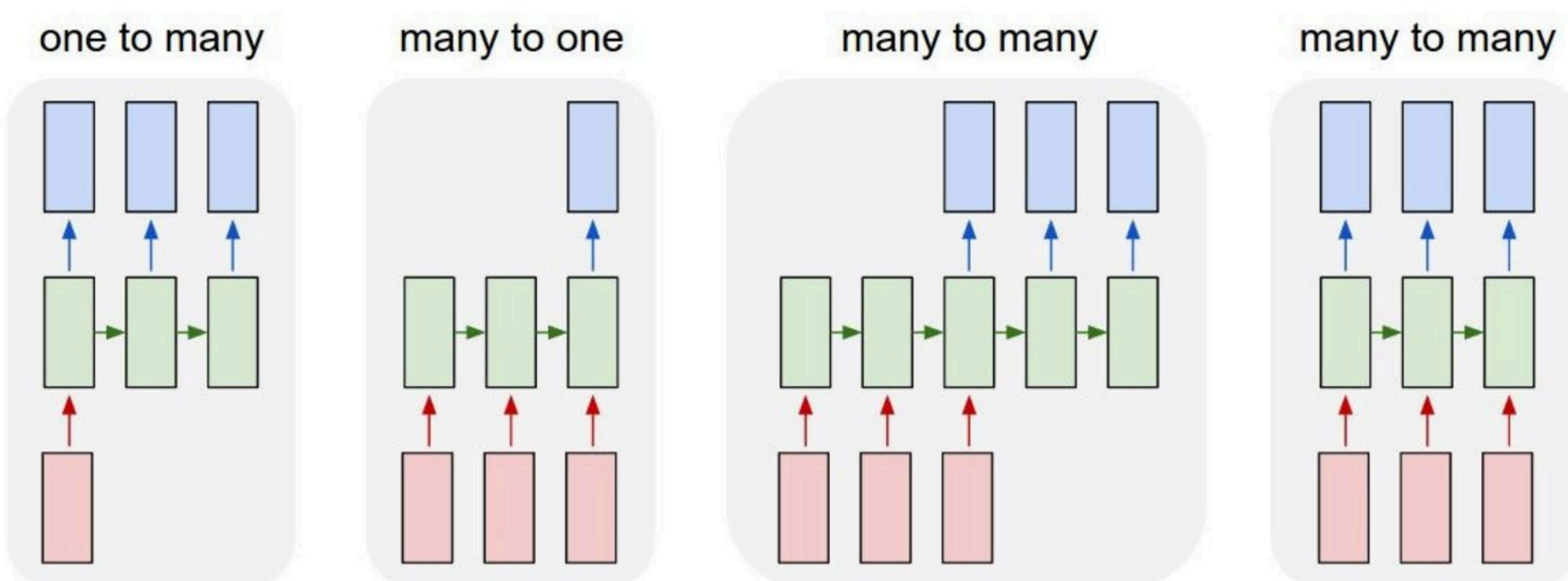
- Setting
  - An input sequence $\mathbf{X}_1, \ldots, \mathbf{X}_N$
  - An output sequence $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$
  - Generally $N \neq M$, i.e., no synchrony between **X** and **Y**
- Examples
  - *Speech recognition*: speech goes in, and a word sequence comes out
  - *Machine translation*: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

深度學習好棒棒!

Deep learning rocks!

S-to-S Models with Alignment

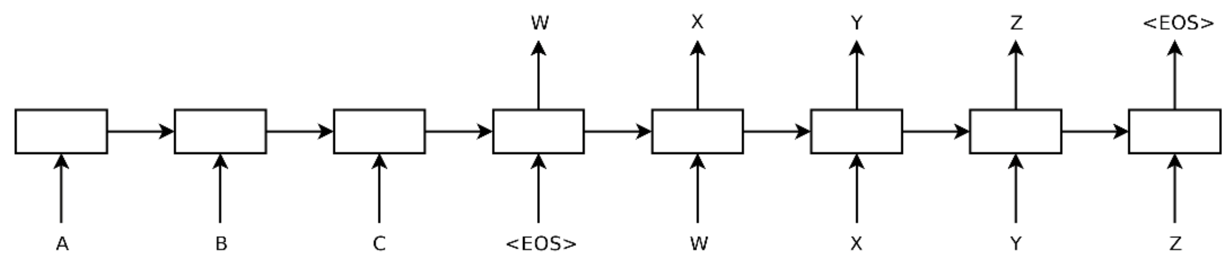
- The input and output sequences happen in the same order
  - The input/output sequences may be asynchronous.
  - E.g., speech recognition or video captioning, in which the input sequence corresponds to the phoneme/caption sequence out.
  - Recall that…

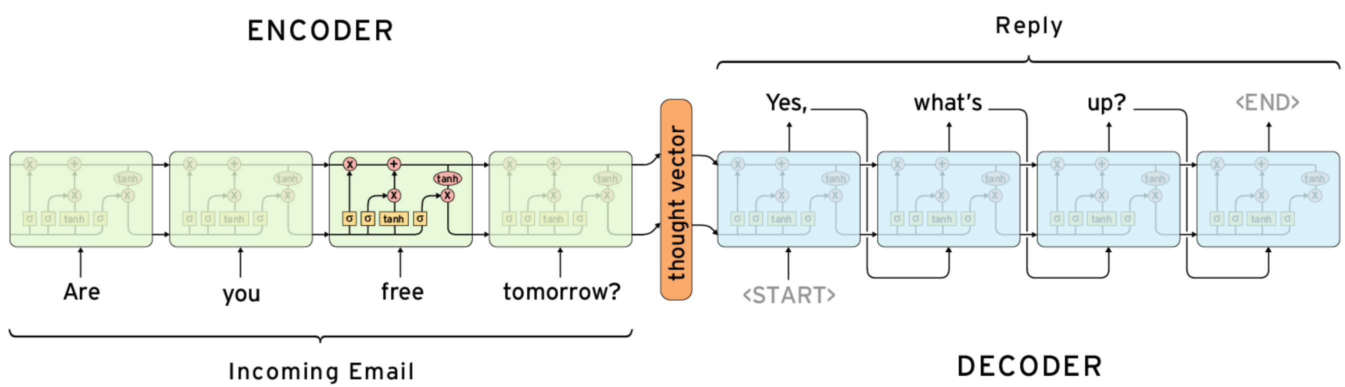


e.g., image caption | e.g., action recognition | e.g., video prediction | e.g., video captioning

Sequence-to-Sequence Modeling (cont'd)

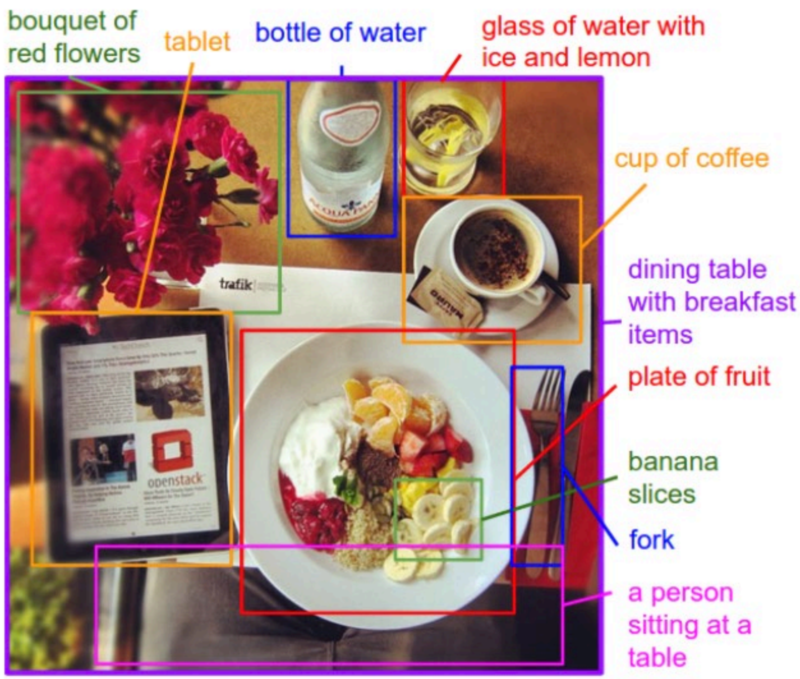
- Original model proposed in NIPS 2014
  - An encoder-decoder model





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

Example of Seq-to-Seq Modeling: Image Captioning



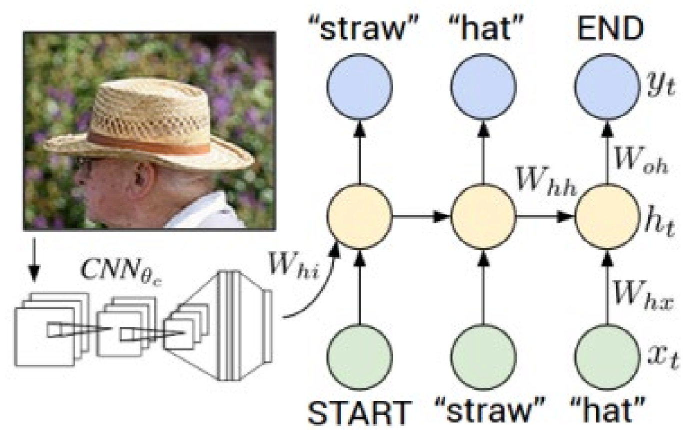


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015 image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

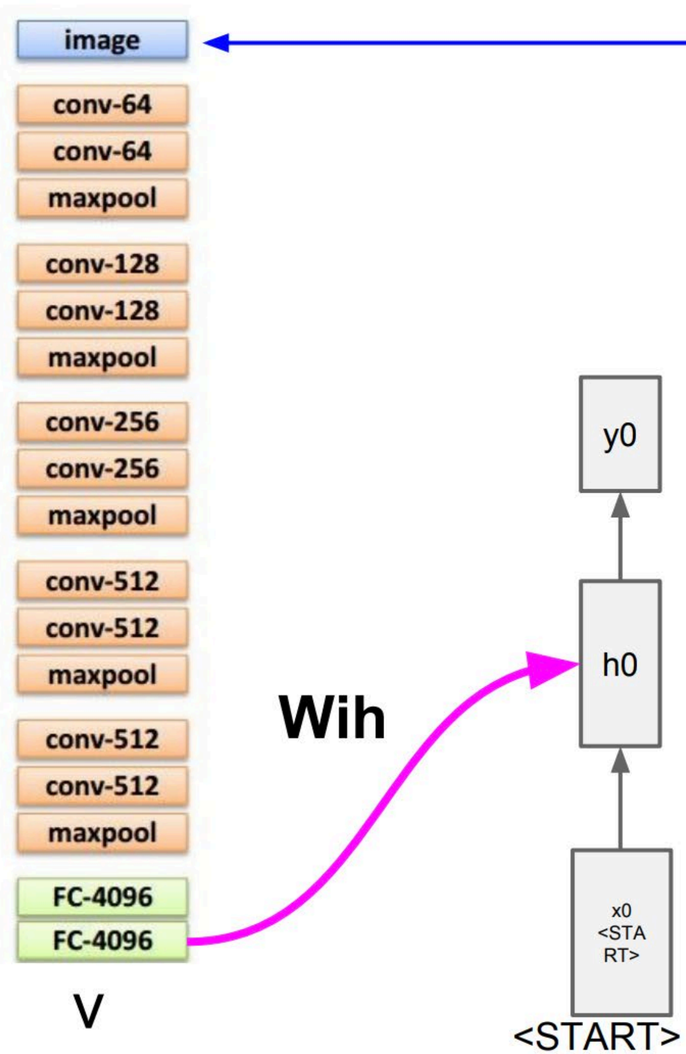
conv-512
conv-512
maxpool

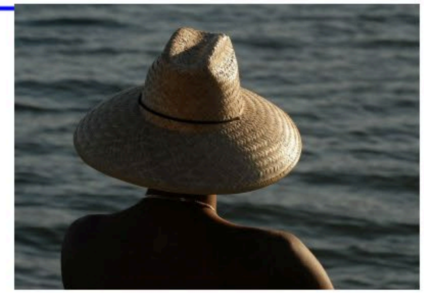
conv-512
conv-512
maxpool

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**)

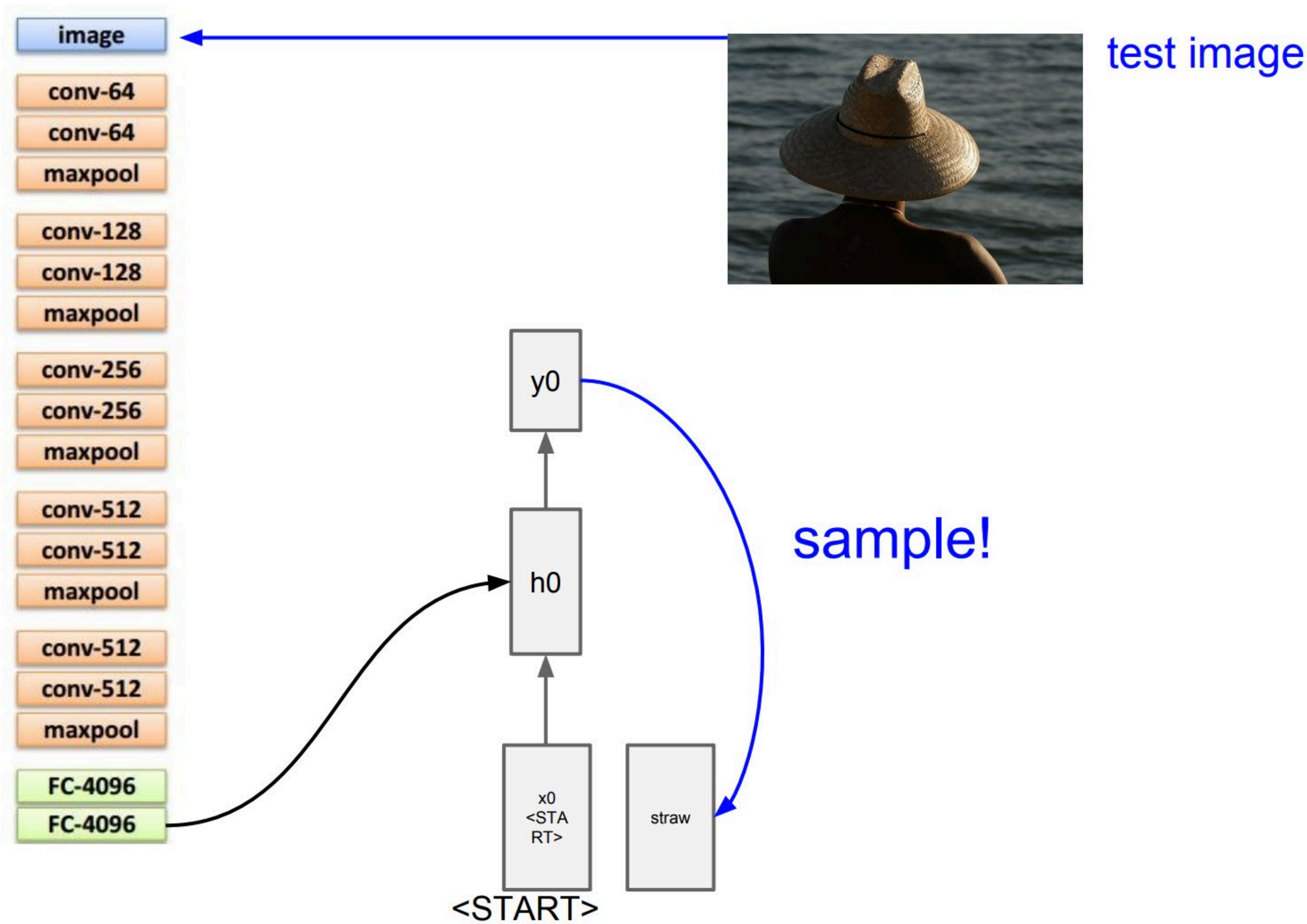
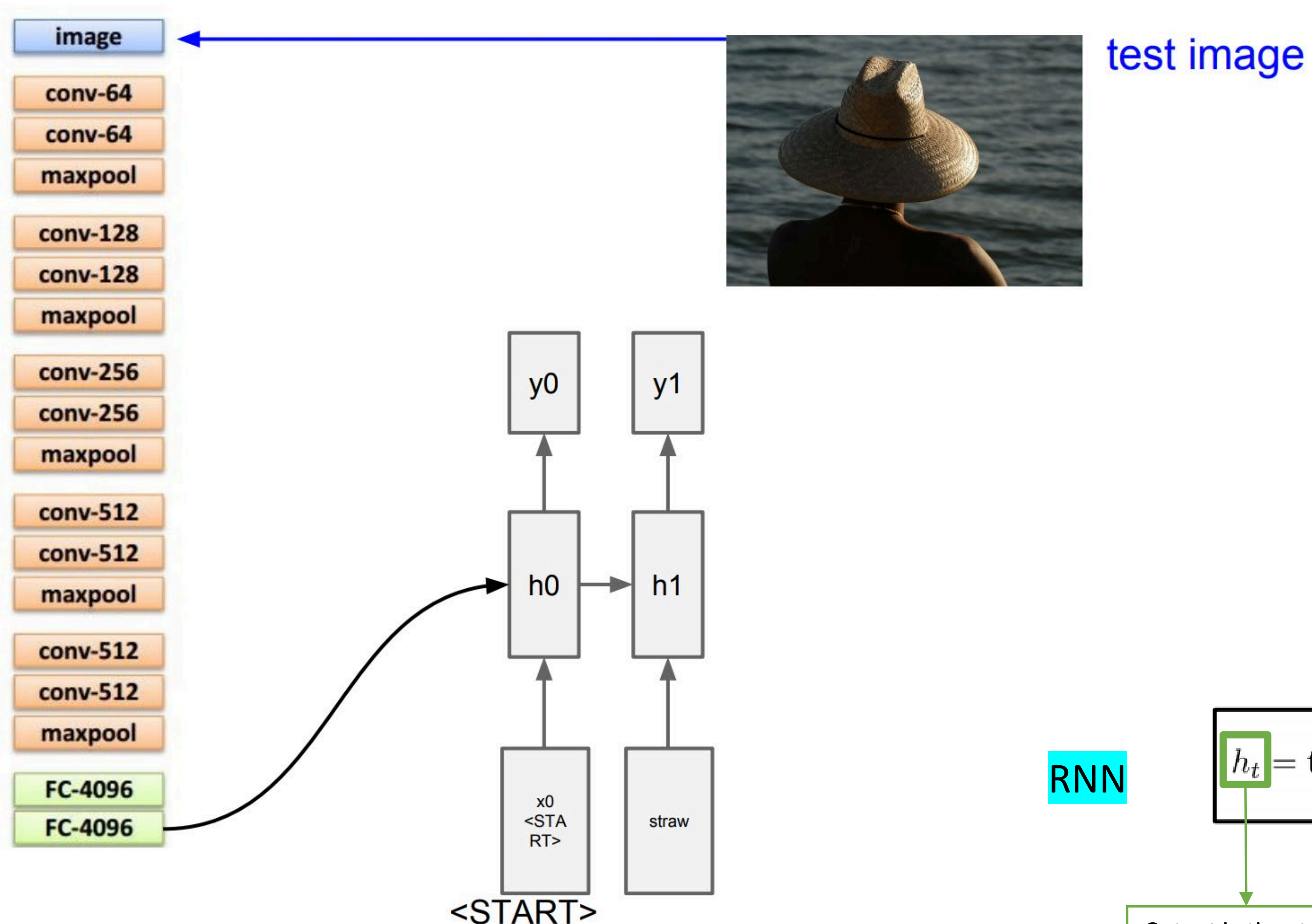


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
sample!
h0
x0 <STA RT>
straw
<START>



RNN

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

Output in time t

Input in time t

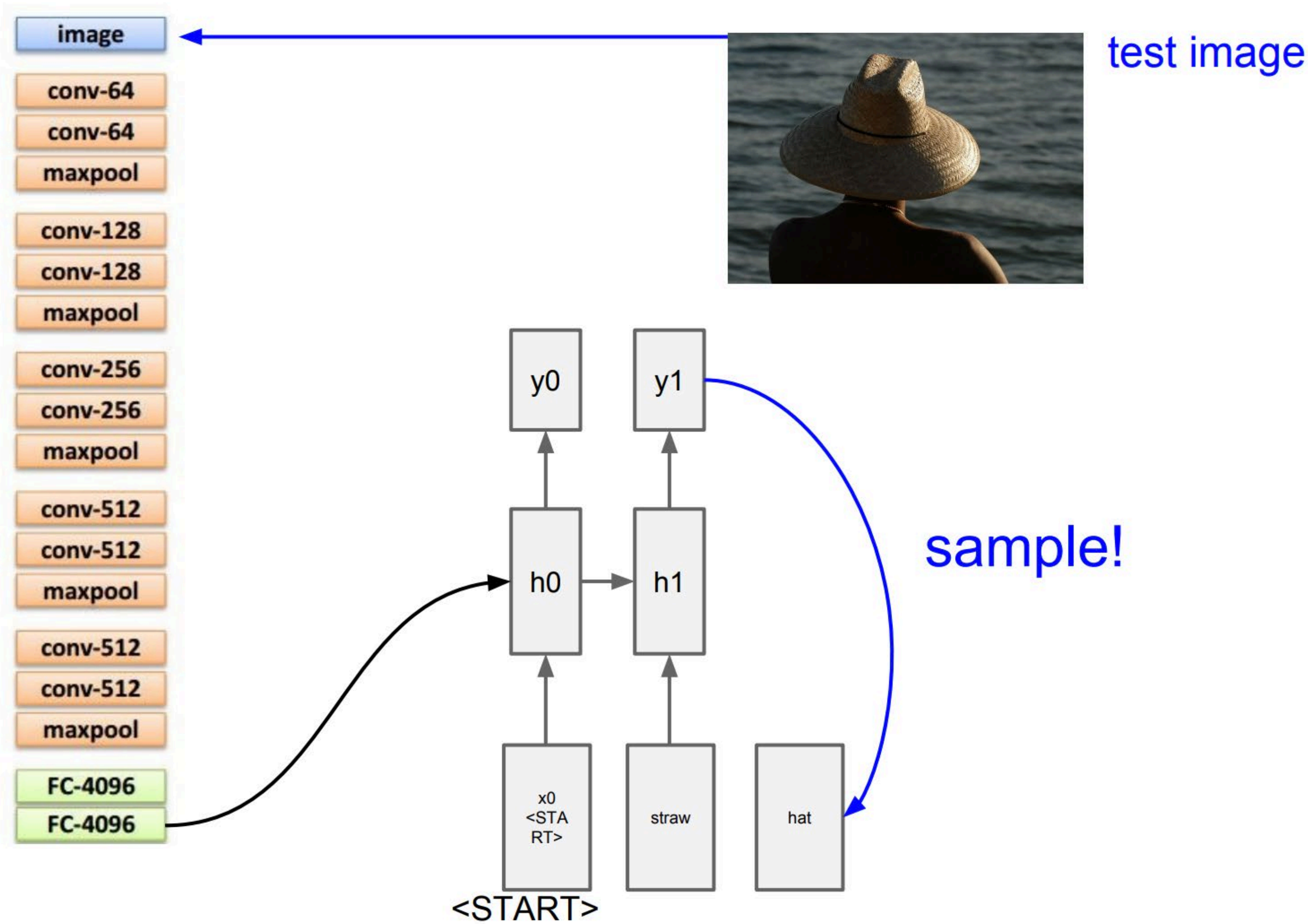


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
sample!
x0 <STA RT>
straw
hat
<START>

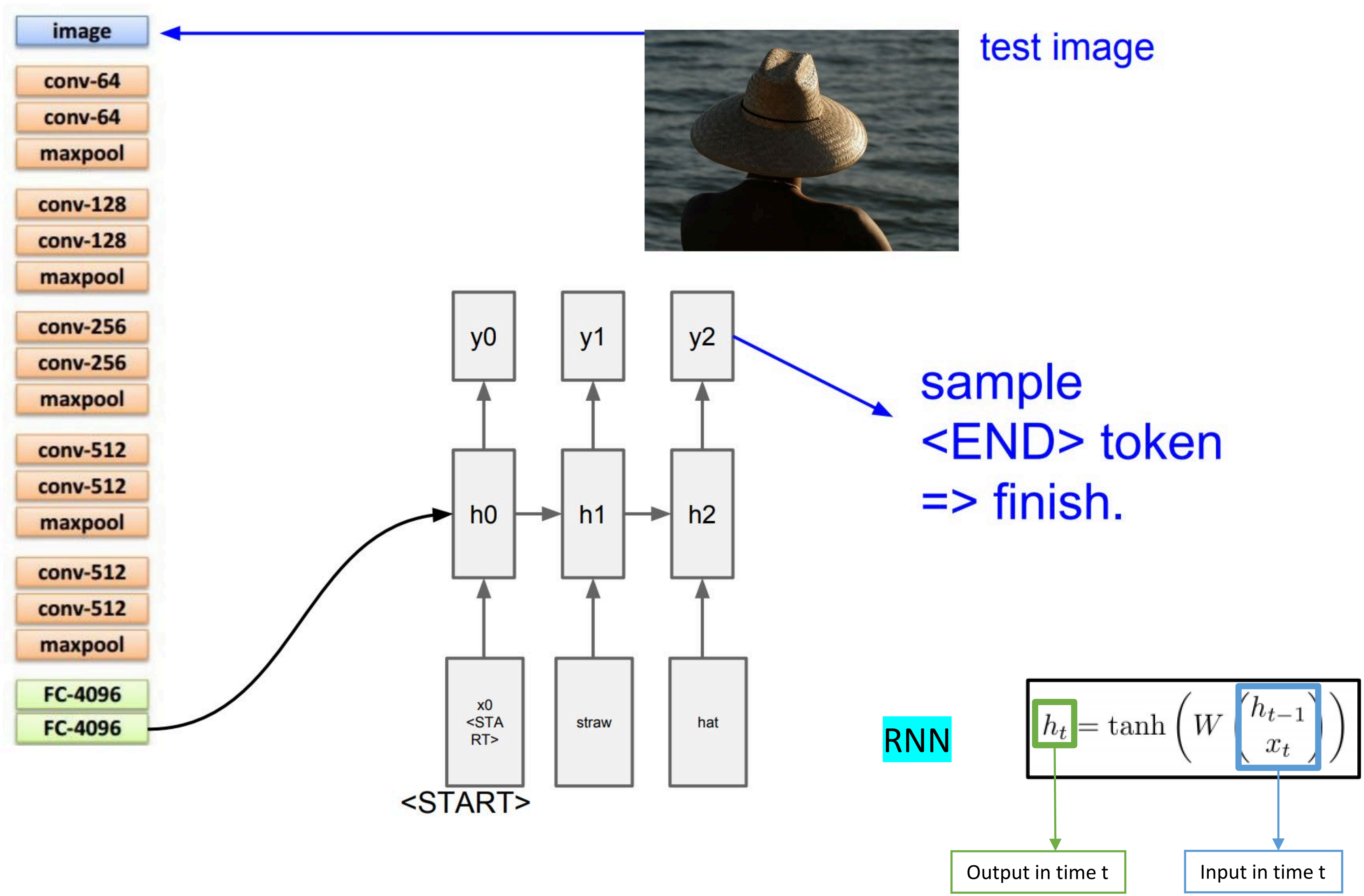
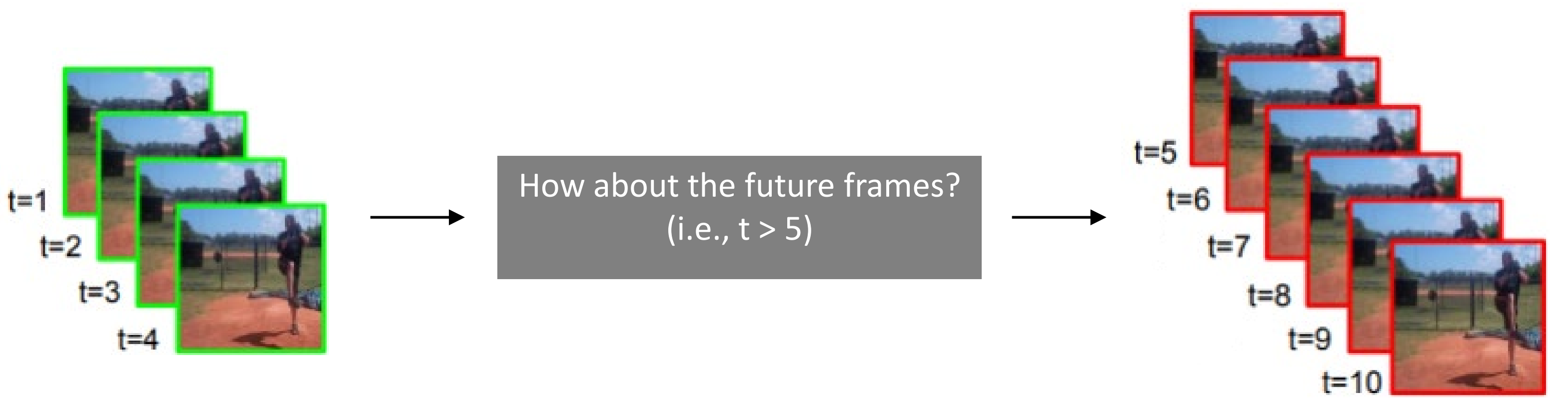


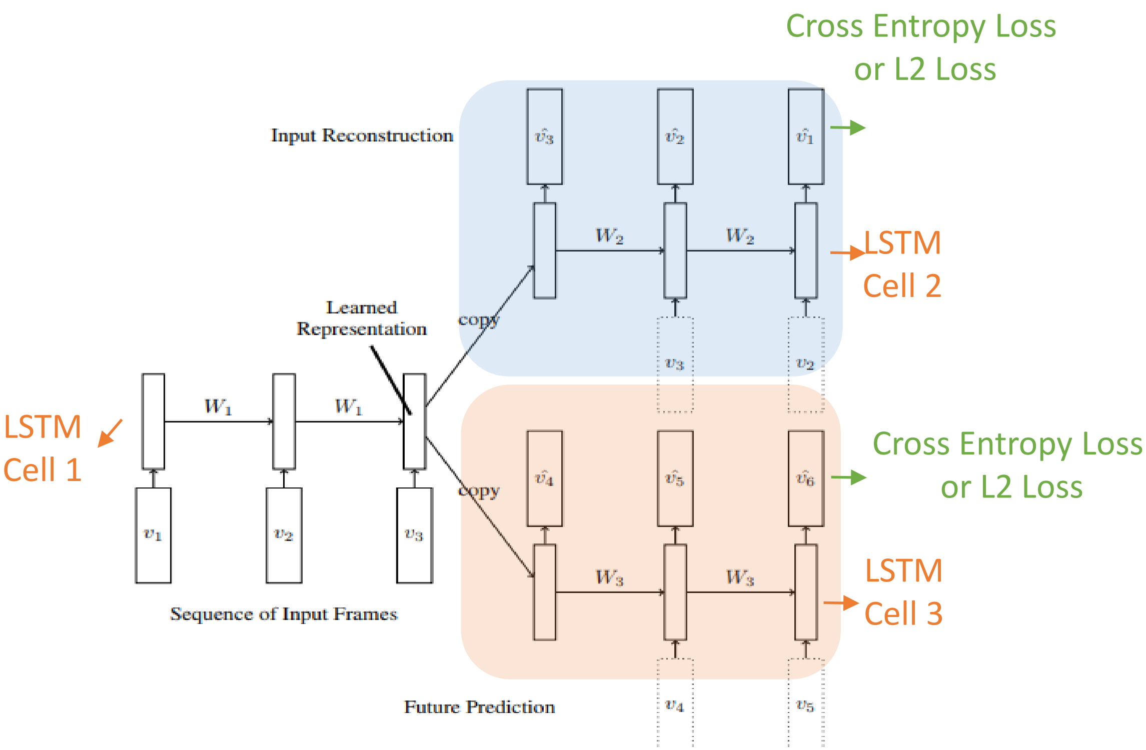
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0 <START>
straw
hat
<START>
sample <END> token => finish.
RNN
$h_t = \tanh \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$
Output in time t
Input in time t

Example of Seq-to-Seq Modeling: Video Prediction

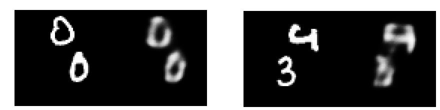
- Input: A few known frames
- Output: Unknown future frames



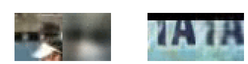
• Unsupervised Learning of Video Representations using LSTMs
(Srivastava et al., ICML'15)



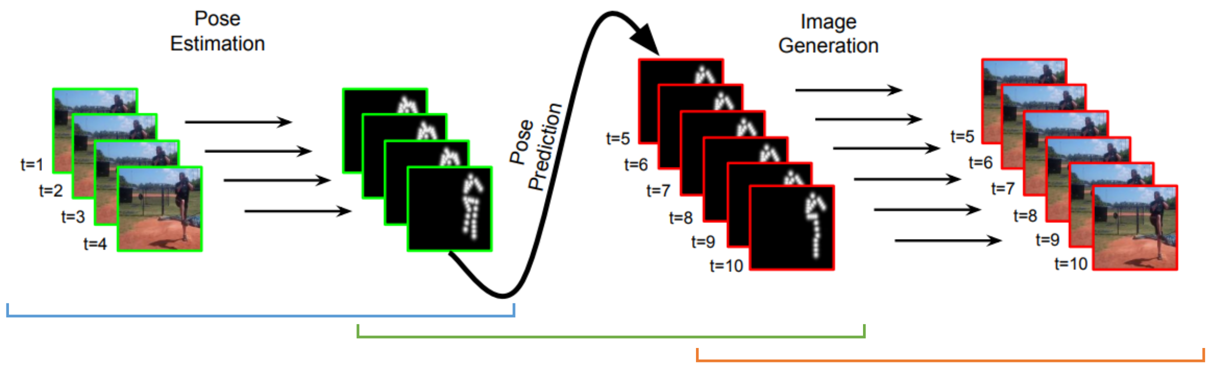
- LSTM Encoder-Decoder model
- Two tasks: Reconstruction & Prediction
- Results (L: ref video, R: output video)
  - Bouncing (Moving) MNIST



  - Video patches of UCF-101



- **Learning to Generate Long-Term Future via Hierarchical Prediction**
  (Villegas et al., ICML'17)



<u>Stage 1:</u>

Pose Estimation — Hourglass network (Newell et al., ECCV'16)

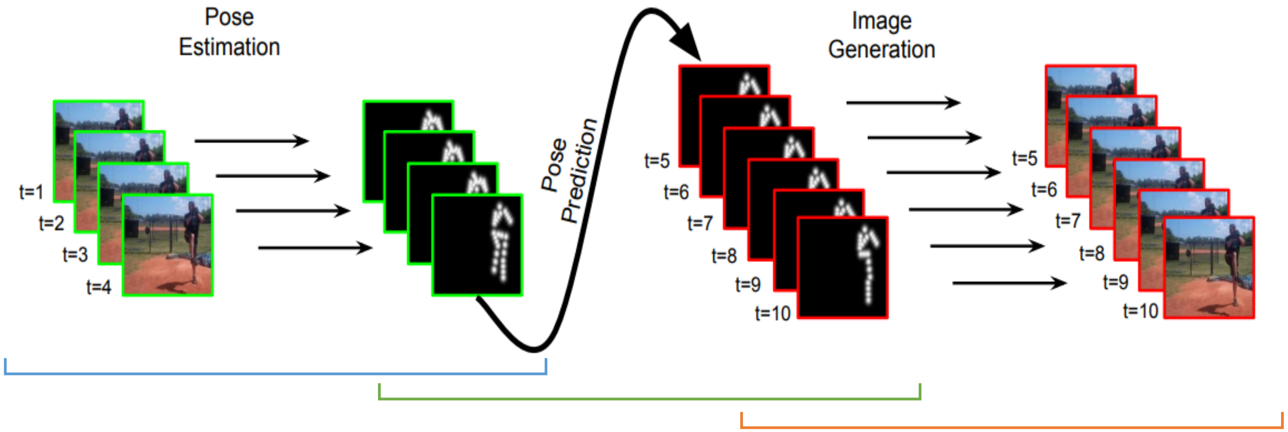
<u>Stage 2:</u>

Future Pose Prediction — Sequence-to-Sequence model on high-level structure

<u>Stage 3:</u>

Image Generation — Visual-Structure Analogy

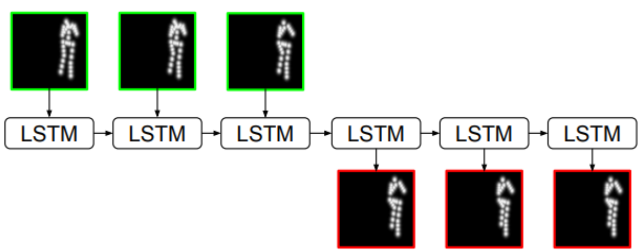
- Learning to generate long-term future via hierarchical prediction
  (Villegas et al., ICML'17)



<u>Step 2:</u>

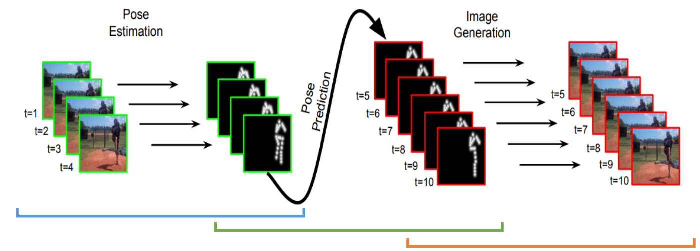
Future Pose Prediction

Sequence-to-Sequence model on high-level structure



<u>Objective Function:</u>

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



- 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

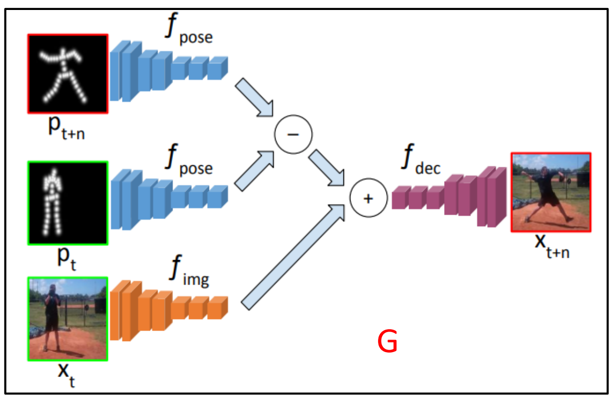
Update Image Generation Network (G)

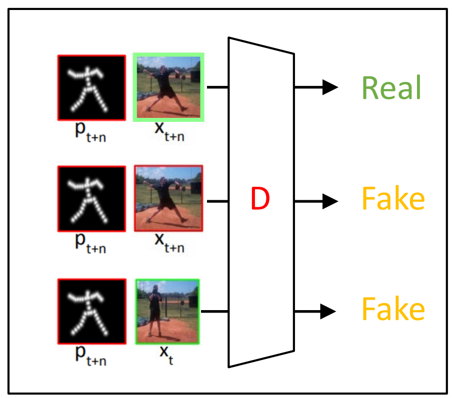
$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{Gen}}$$

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

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

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



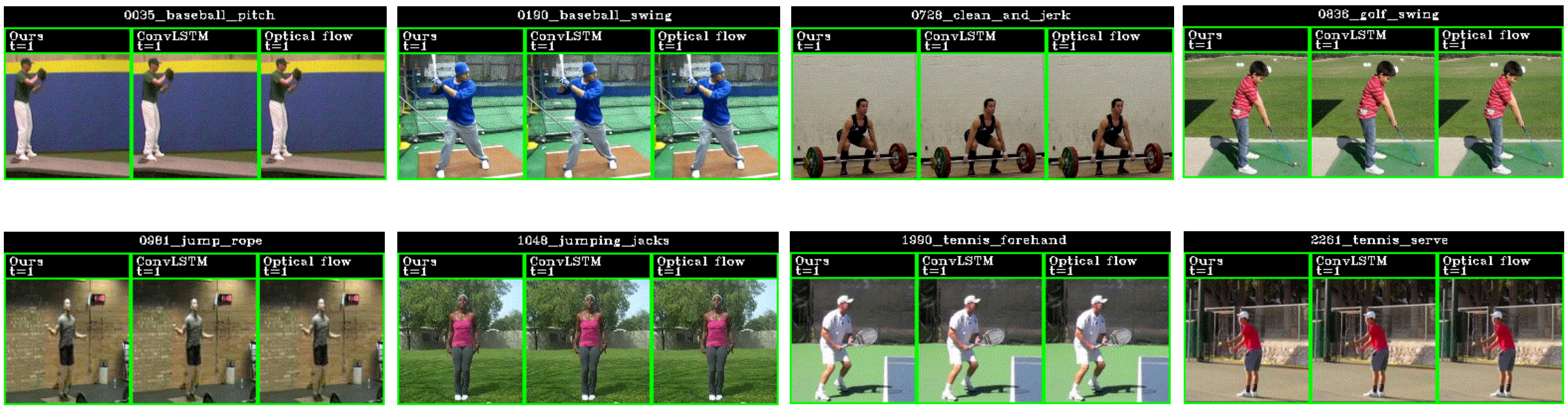


Update Discriminator (D)

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

- Example Results

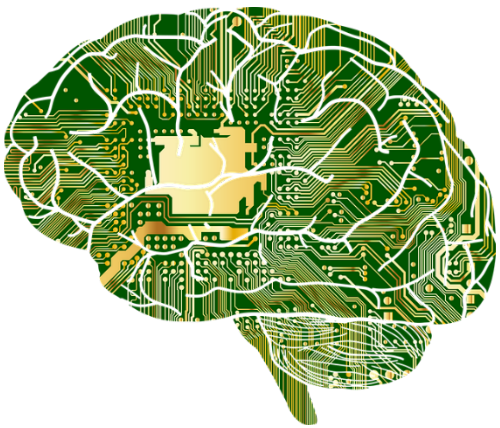
Results on Penn Action Dataset:

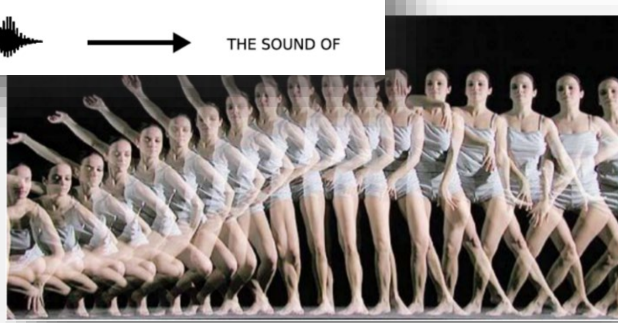


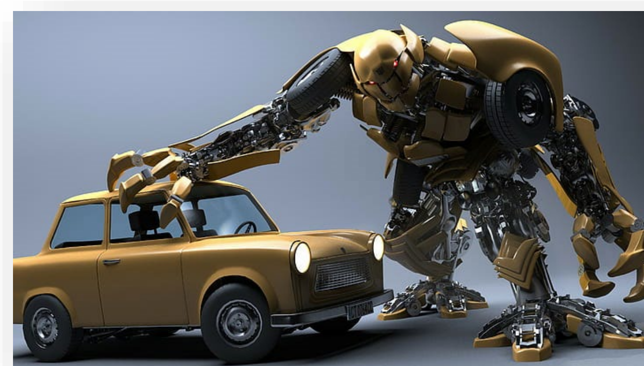
- Note that the above two video prediction works are deterministic, not stochastic!

What to Be Covered Today…

- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer (if time permits)

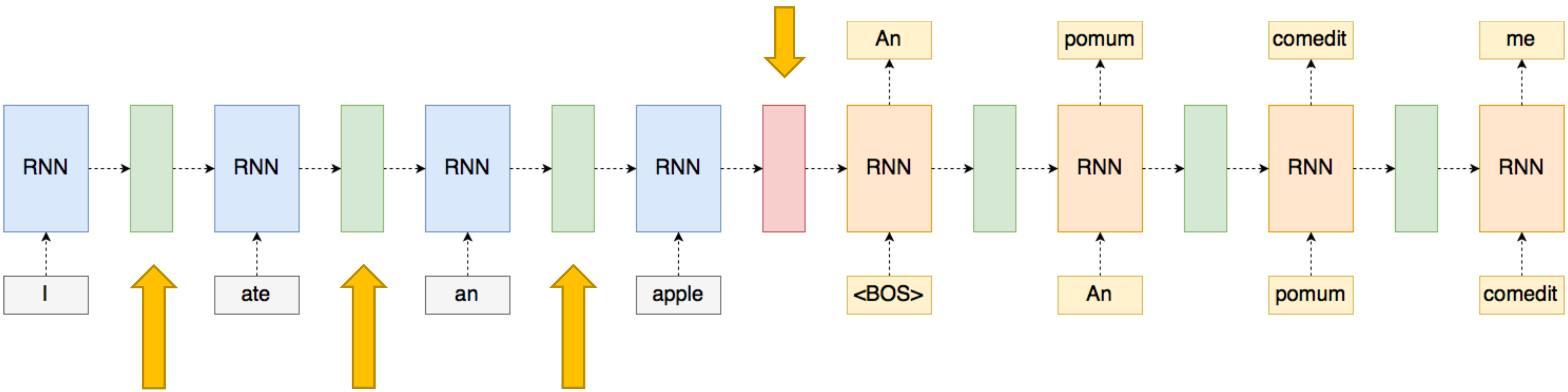






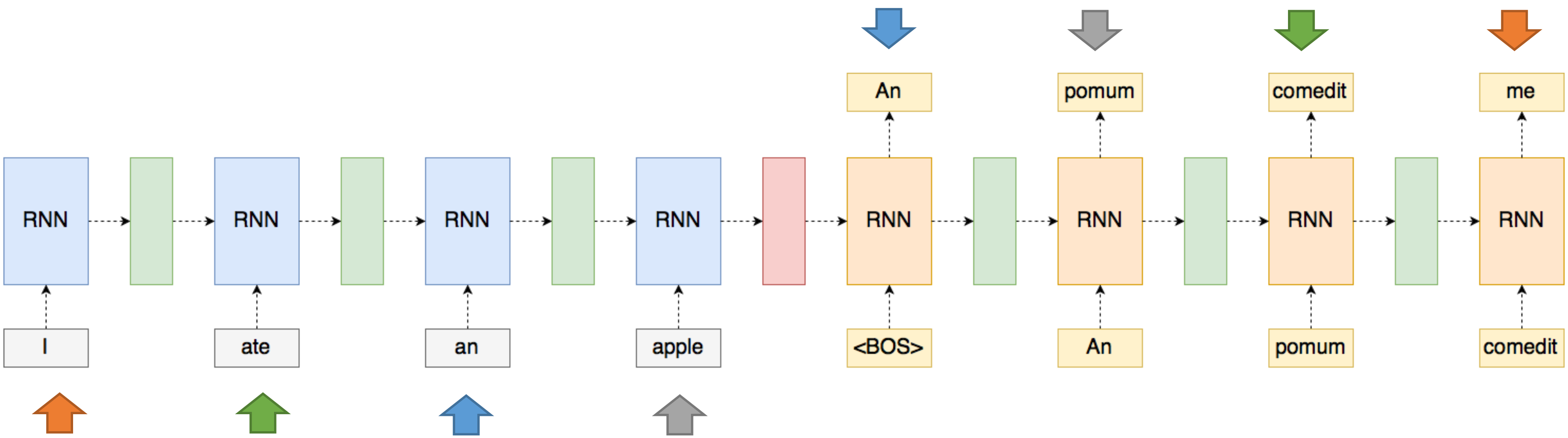
What’s the Potential Problem in RNN?

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



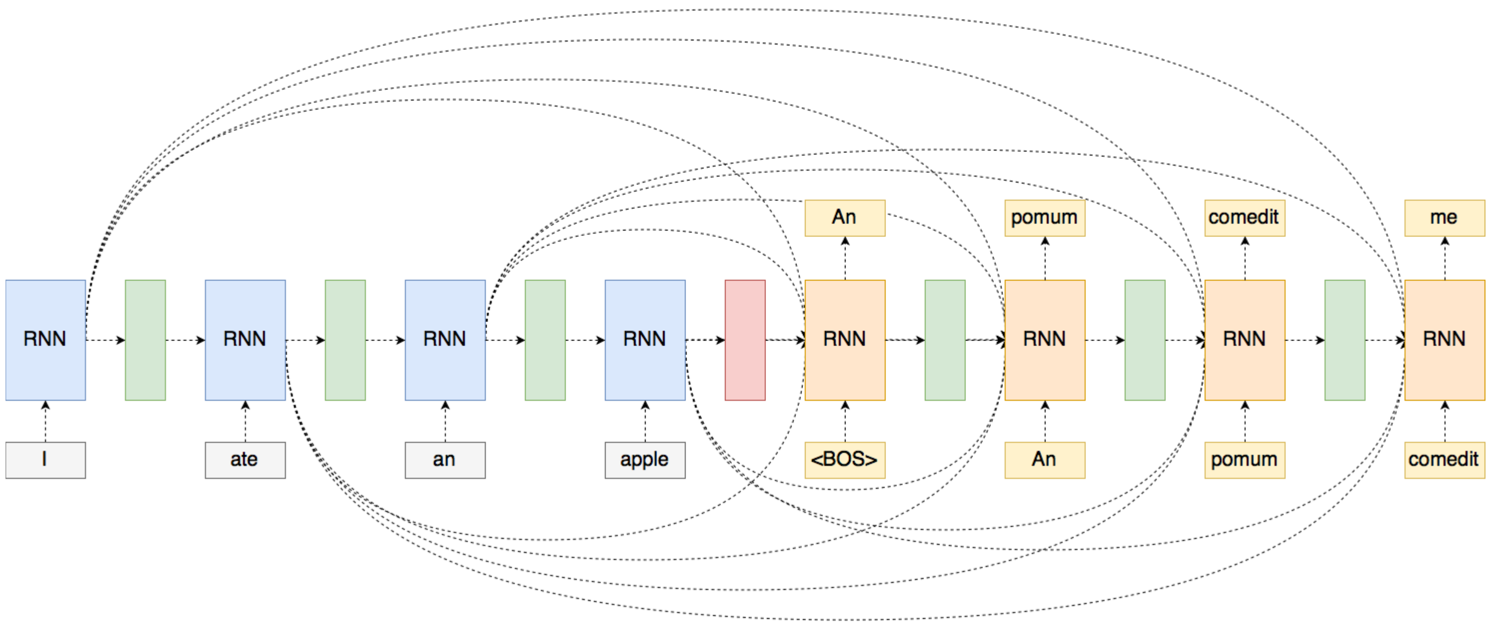
What's the Potential Problem? (cont'd)

- 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)

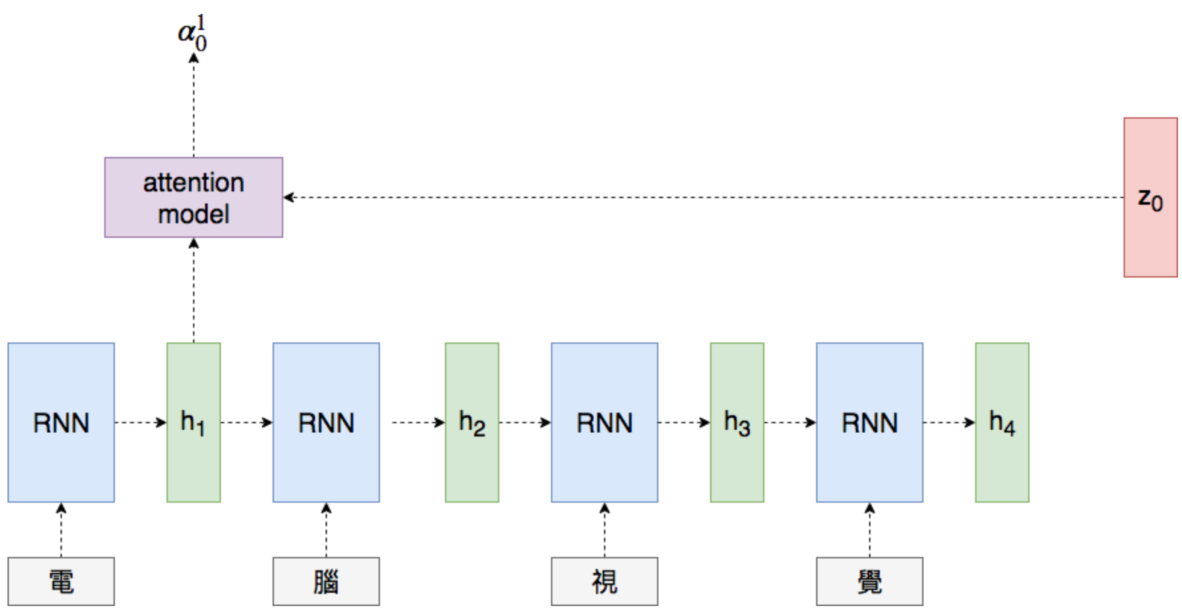
- Connecting every hidden state between encoder and decoder?



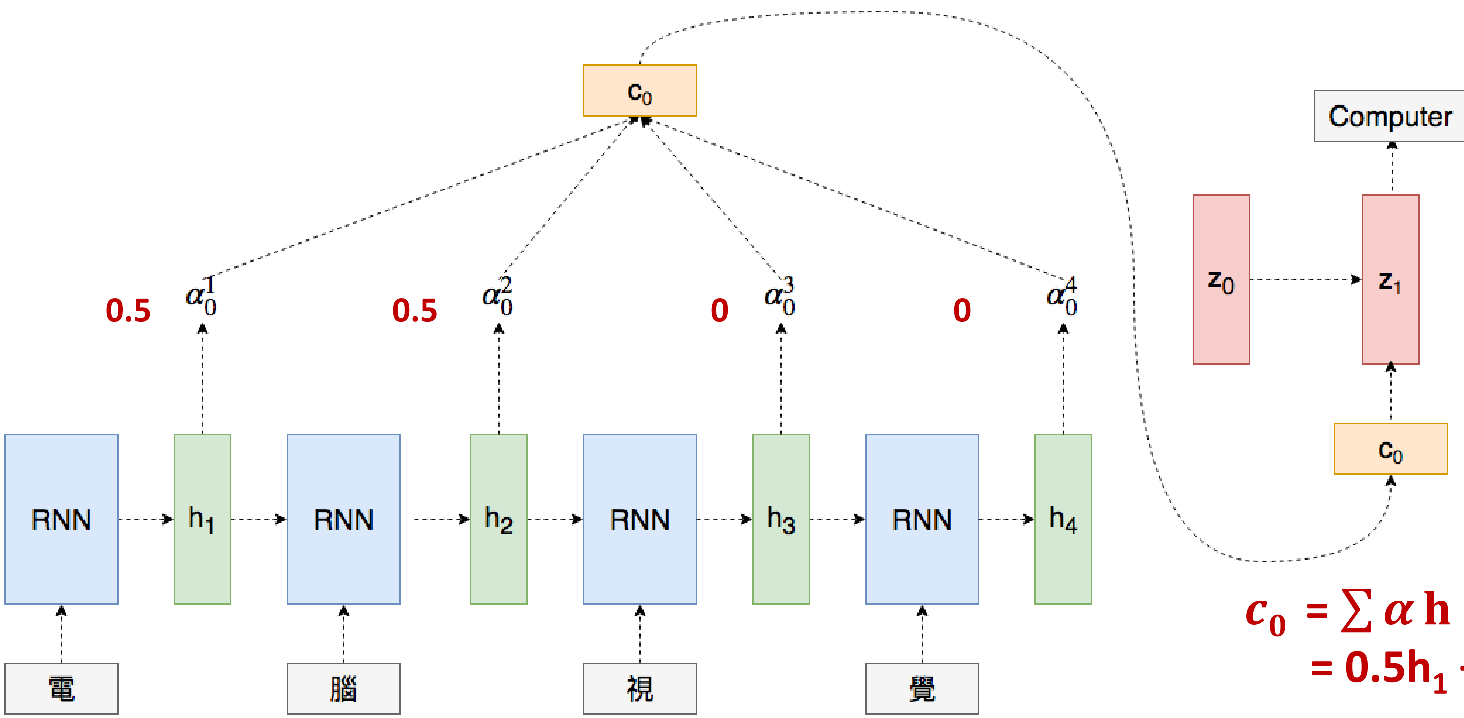
- Infeasible!
  - Both inputs and outputs are with varying sizes.
  - Overparameterized
  - Possible solution: **attention**

Solution #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 with other parts of network (e.g., recognition, etc.)

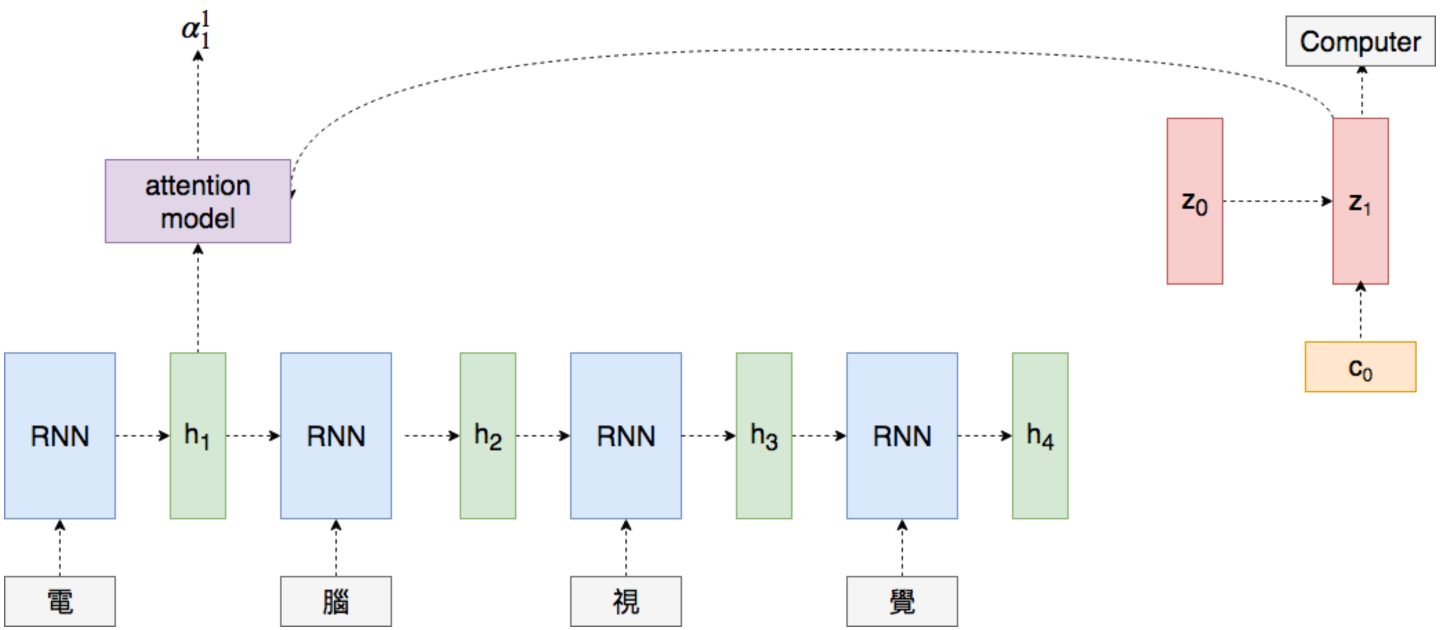


Solution: Attention Model

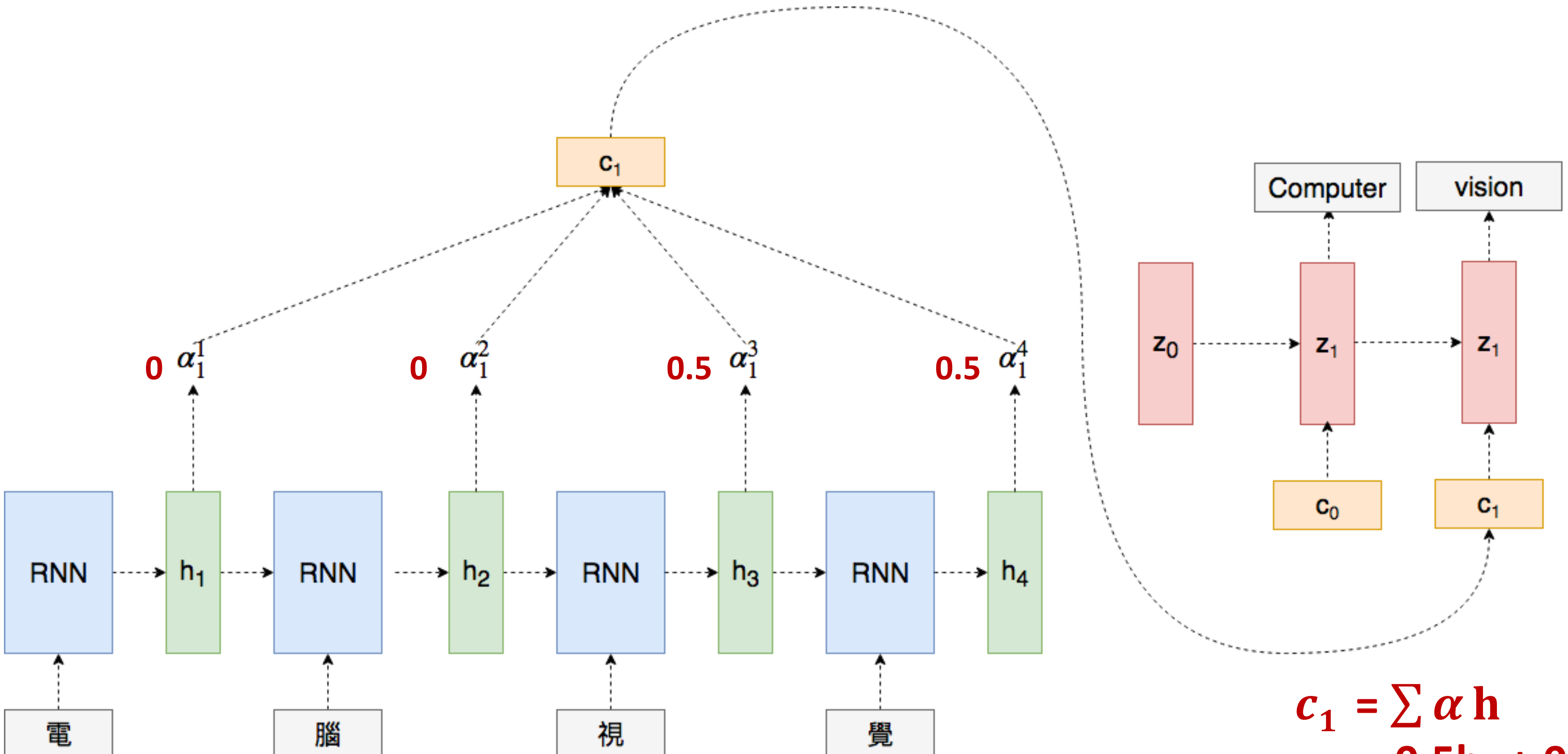


$$c_0 = \sum \alpha \mathrm{h}$$
$$= 0.5\mathrm{h}_1 + 0.5\mathrm{h}_2$$

Solution: Attention Model

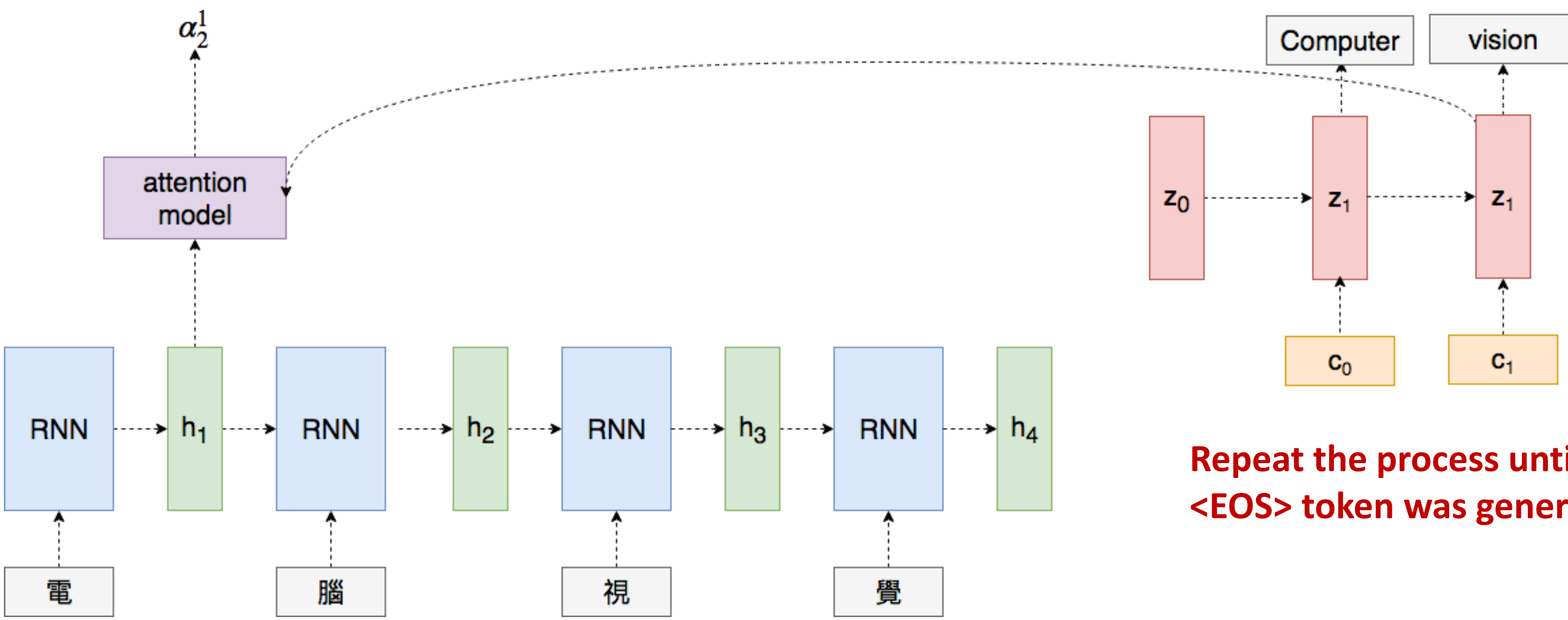


Solution: Attention Model



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

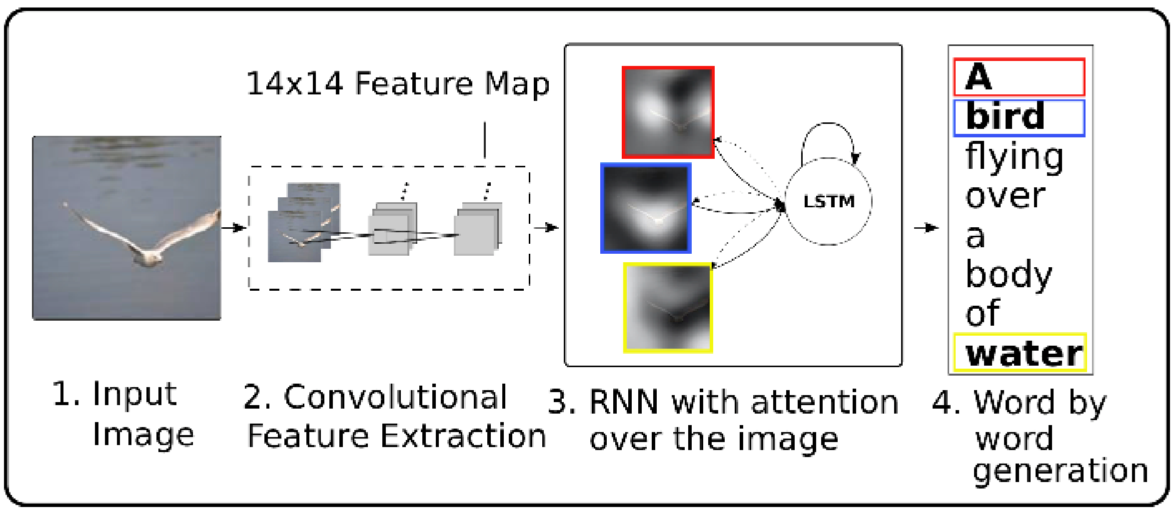
Solution: Attention Model



Repeat the process until <EOS> token was generated

Example: 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

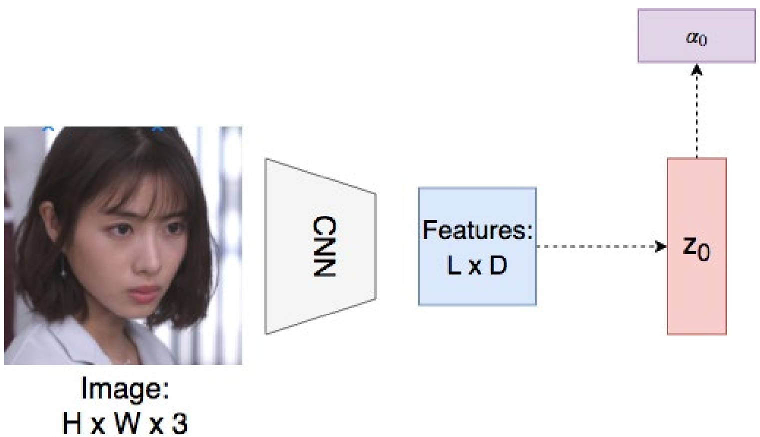


Image Captioning with Attention

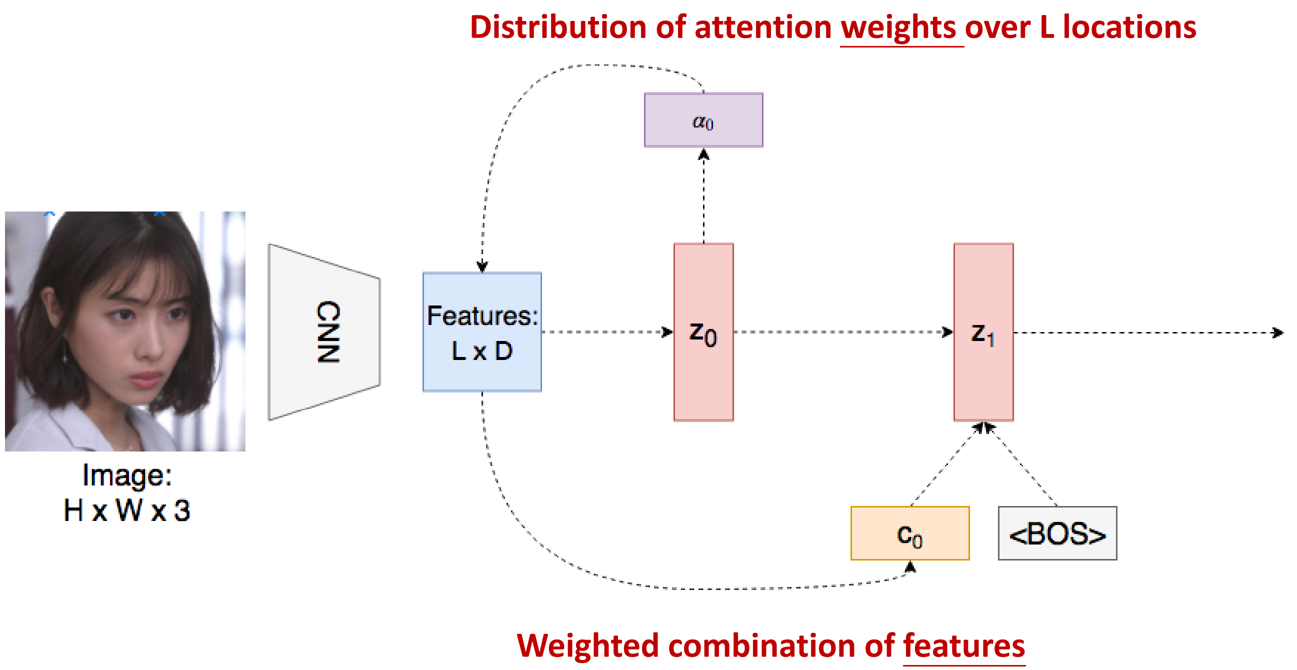


Image Captioning with Attention

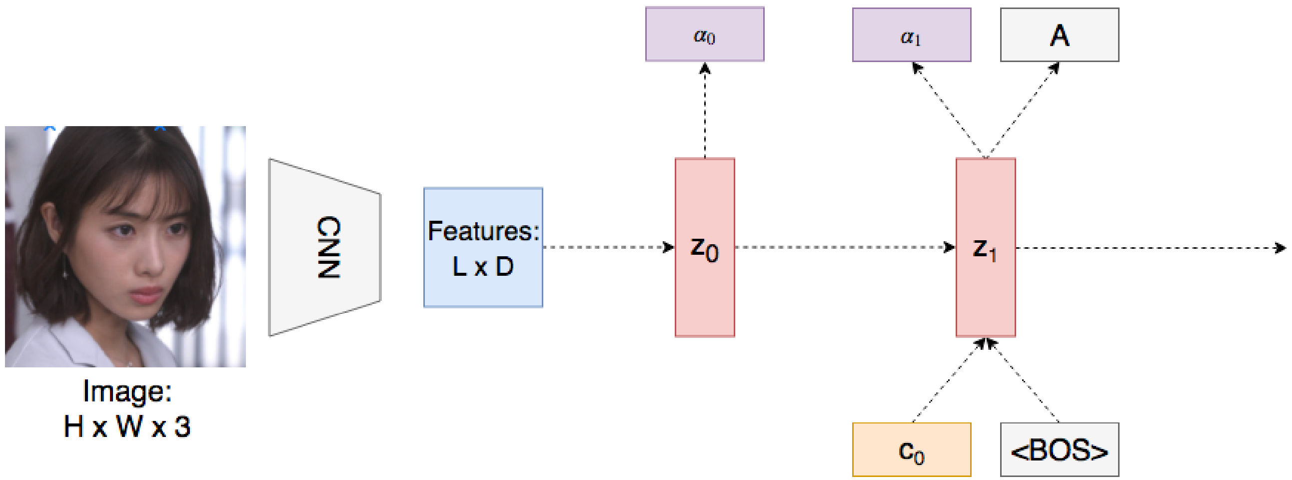


Image Captioning with Attention

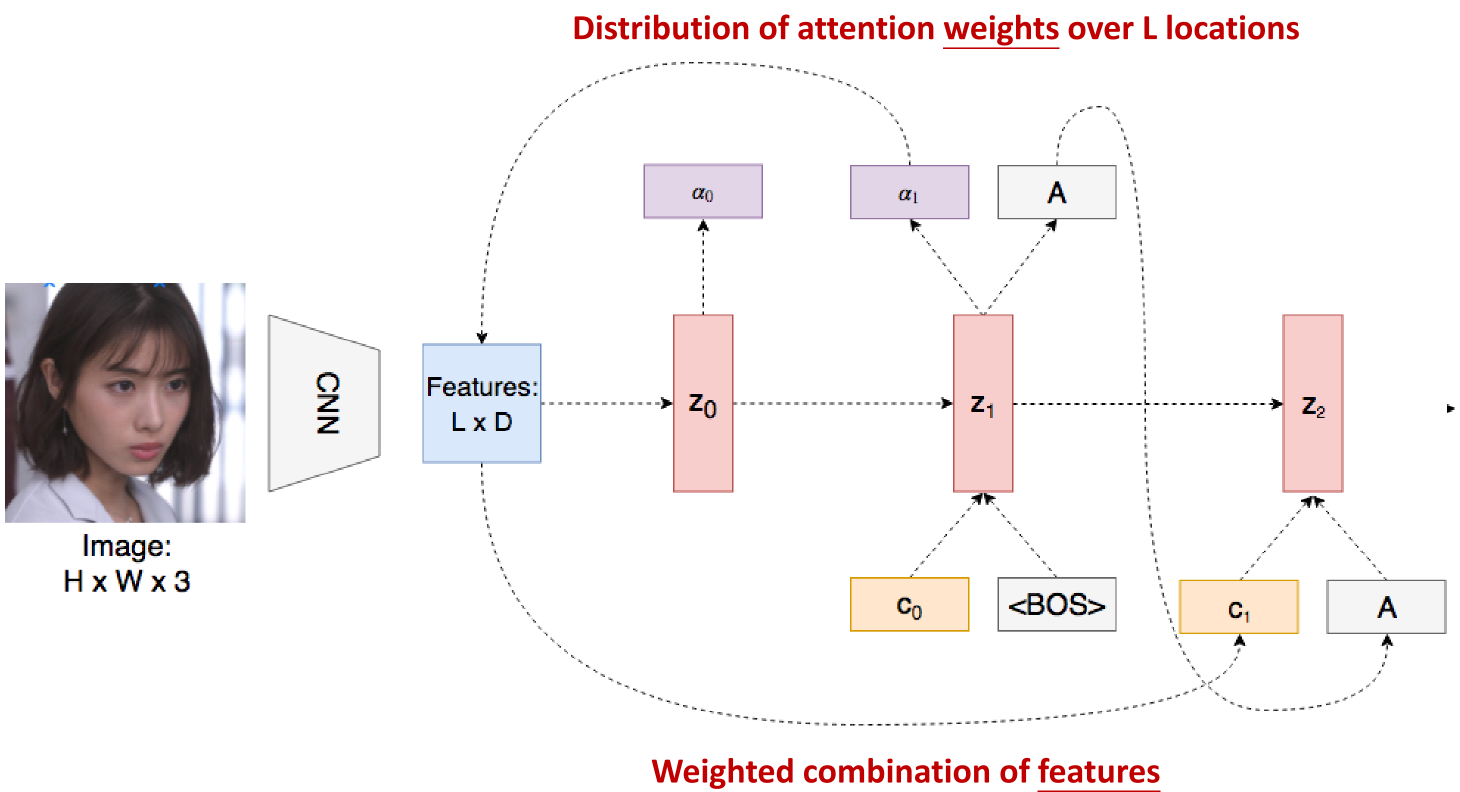
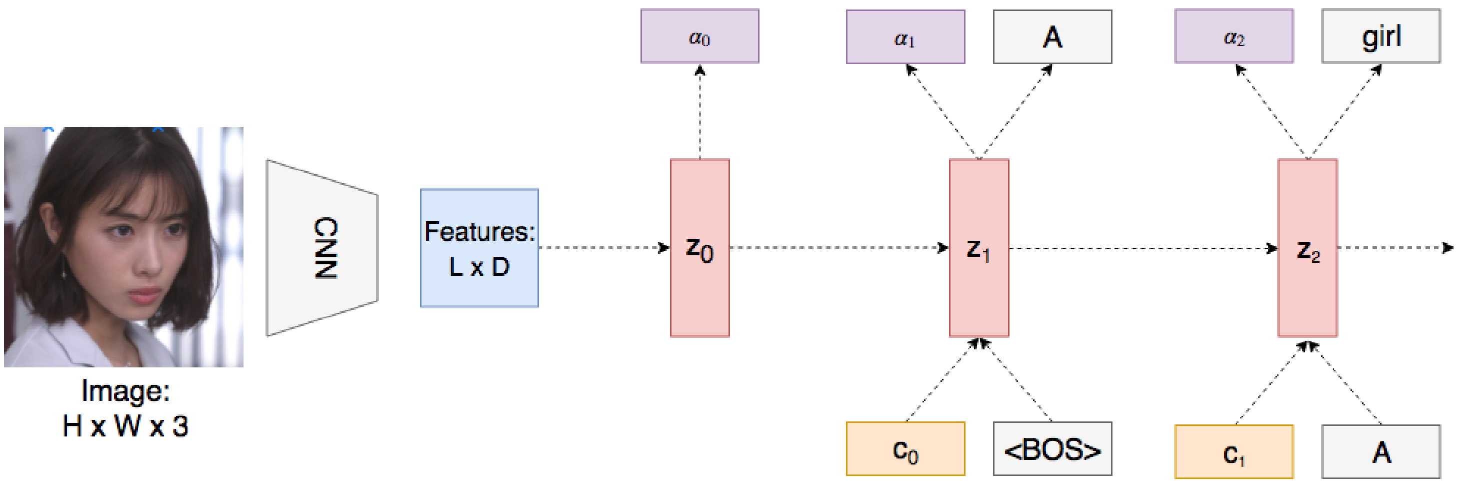
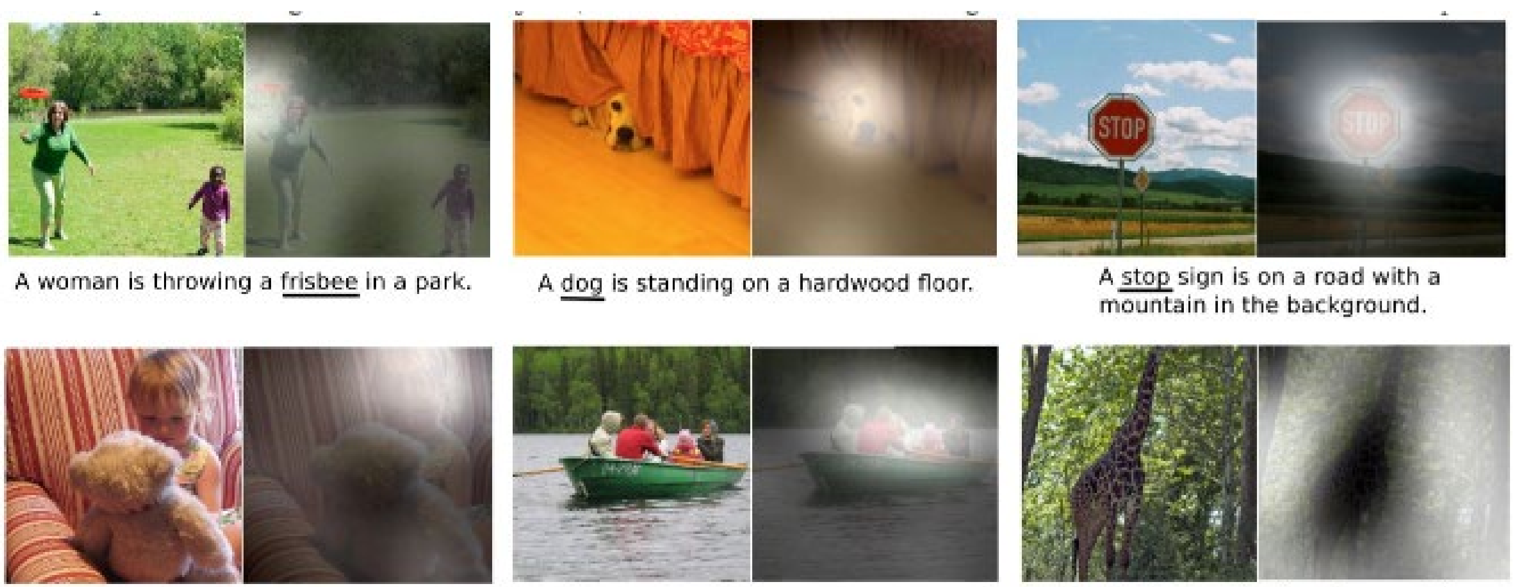


Image Captioning with Attention



Repeat the process until <EOS> token was generated

Attention helps image recognition…
What else?



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.

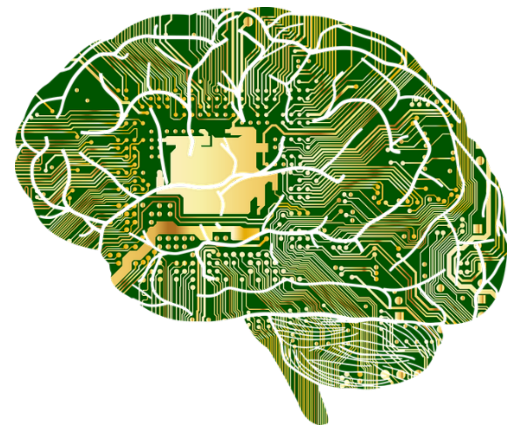
A little girl sitting on a bed with a teddy bear.

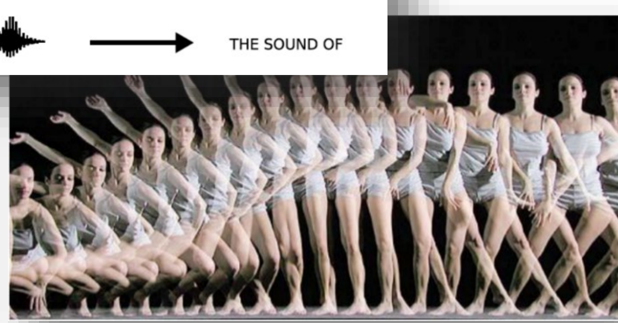
A group of people sitting on a boat in the water.

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

What to Be Covered Today…

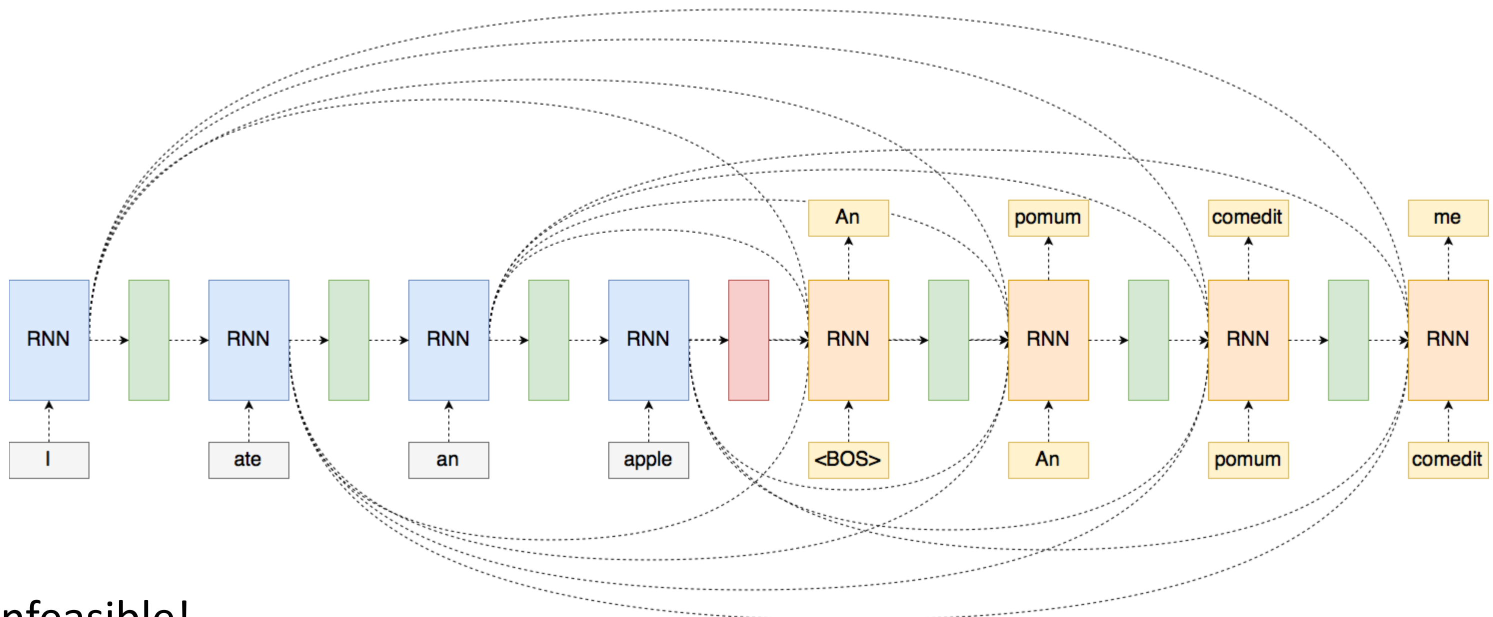
- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer





RNN with Attention is Good, But..

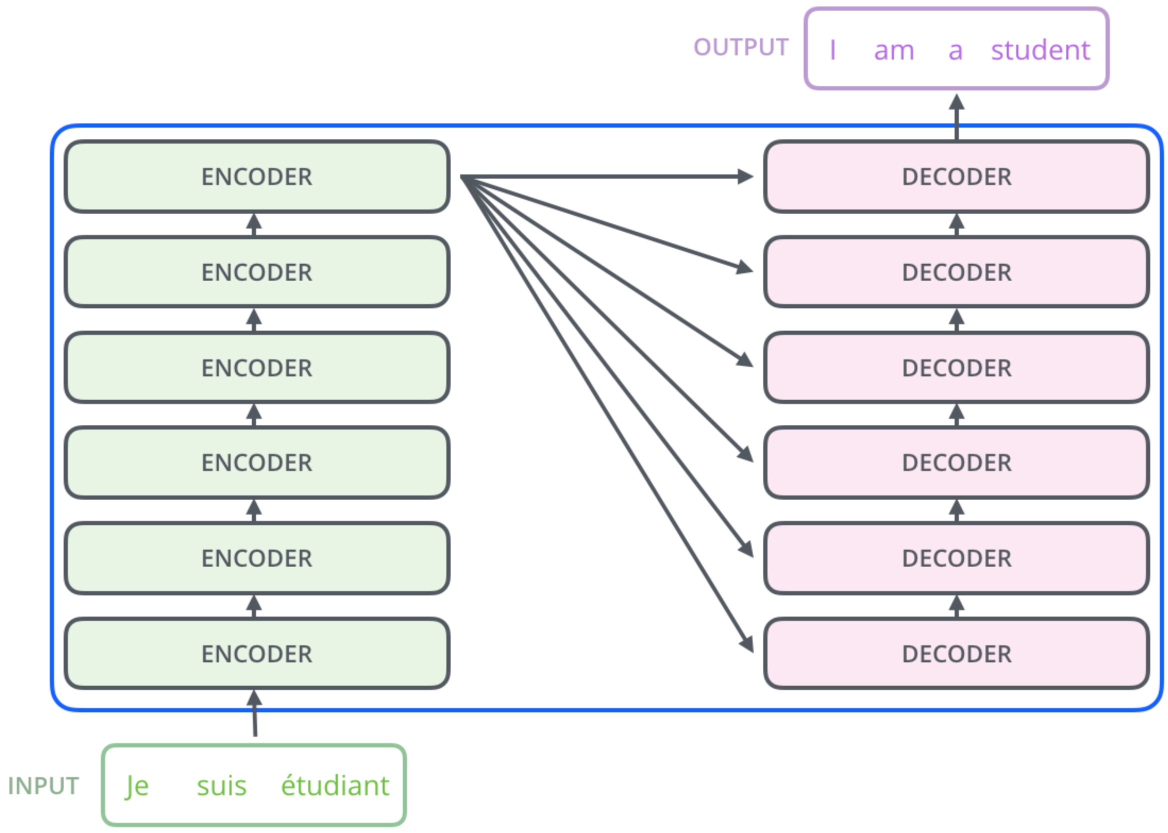
- Attention in a pre-defined sequential order
- Information loss due to long sequences…
- Connecting every hidden state between encoder and decoder?



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

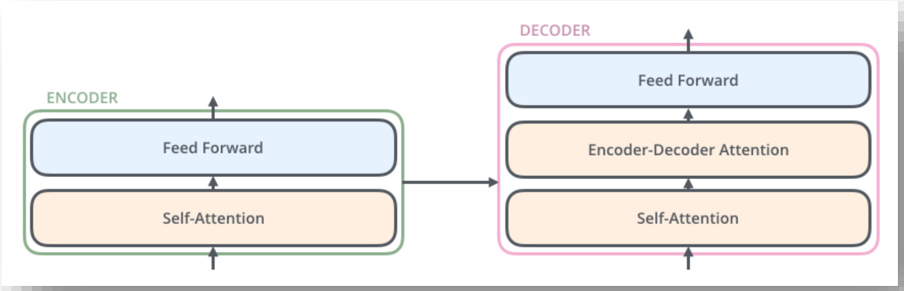
RNN with Attention is Good, But..

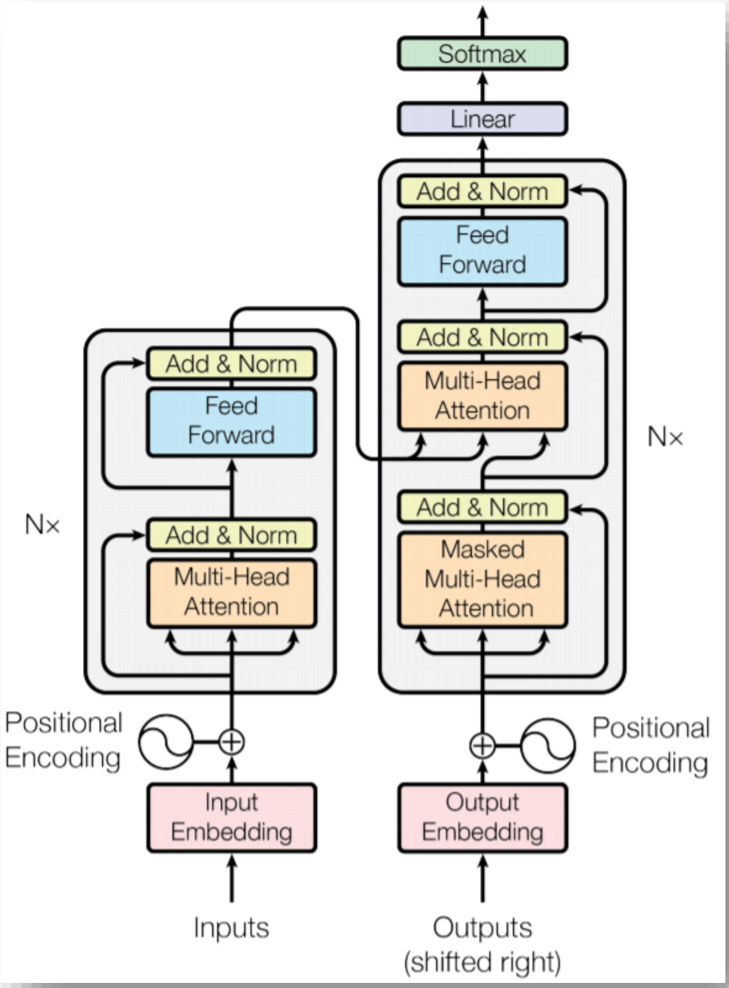
- Any better way to perform attention across features?



Solution #2: Transformer

- "Attention is all you need", NIPS/NeurIPS 2017
- Self-attention for text translation
- Say goodbye to CNN & RNN
- More details available at:
  https://www.youtube.com/watch?v=rcWMRA9E5RI
  http://jalammar.github.io/illustrated-transformer/





Tokenization

Many words map to one token, but some don't: indivisible.

Many	words	map	to	one	token	,	but	some	don	'	t:	indiv	isible	.
8607	4339	2472	311	832	4037	11	719	1063	1541	956	25	3687	23936	13

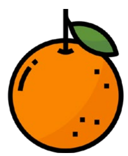
One-hot encoding

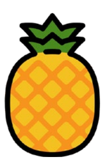
$$
\begin{array}{ccccccc}
 & \text{cat} & \text{dog} & \text{bear} & \text{cow} & & \text{indiv} \\
\text{\# tokens} & \begin{bmatrix}1\\0\\0\\0\\0\\\vdots\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\\0\\0\\\vdots\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\\0\\0\\\vdots\\0\end{bmatrix} & \begin{bmatrix}0\\0\\0\\1\\0\\\vdots\\0\end{bmatrix} & & \begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\0\end{bmatrix}
\end{array}
$$

Value 1 at 3687th entry

TOKEN EMBEDDING

One-hot encoding



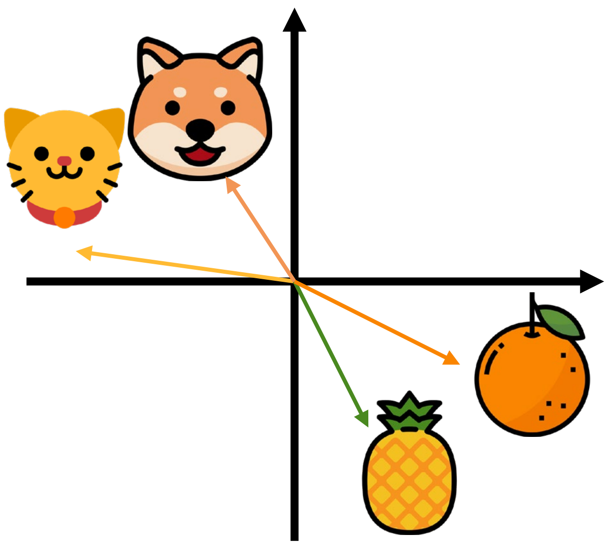


$$
\begin{array}{ccccc}
\text{cat} & \text{dog} & \text{bear} & \text{cow} & \text{indiv} \\
\begin{bmatrix}1\\0\\0\\0\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\1\\0\\0\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\1\\0\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\1\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\0\end{bmatrix}
\end{array}
$$

Value 1 at 3687^{th} entry

Slide credit: Prof. J.-B. Huang @ U. Maryland

TOKEN EMBEDDING

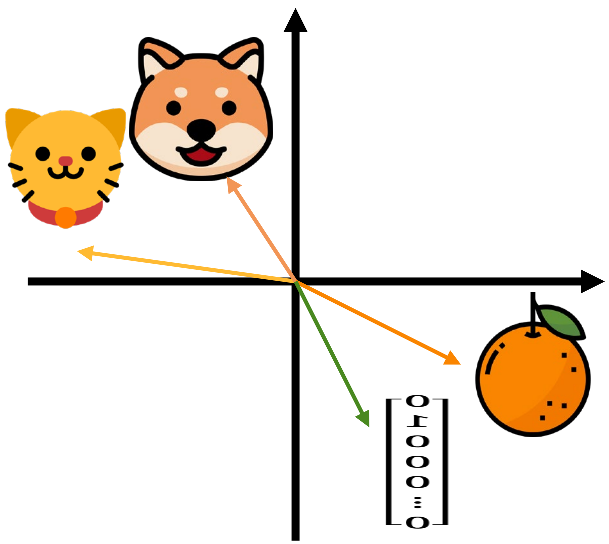


Embedding Space

$$
\begin{array}{ccccc}
\text{cat} & \text{dog} & \text{bear} & \text{cow} & \text{indiv} \\
\begin{bmatrix}1\\0\\0\\0\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\1\\0\\0\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\1\\0\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\1\\0\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\0\end{bmatrix}
\end{array}
$$

Value 1 at 3687^{th} entry

TOKEN EMBEDDING



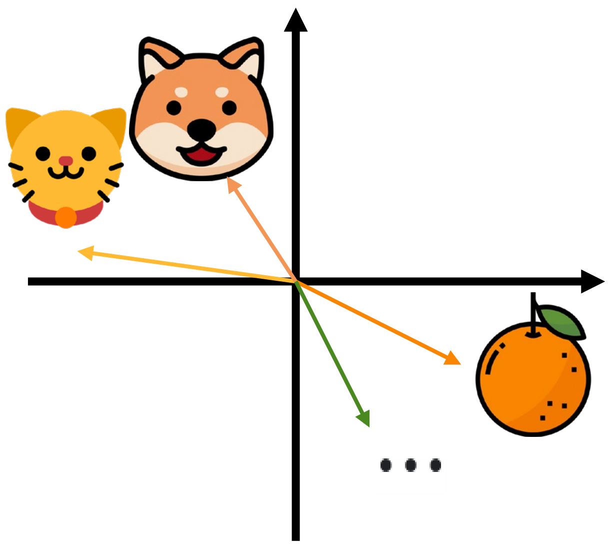
Embedding Space

$$d \begin{bmatrix} 0.5 \\ 2.7 \\ 1.2 \\ \vdots \\ 0.2 \end{bmatrix} = d \overset{\#\ \text{tokens}}{\begin{bmatrix} \quad W_E \quad \end{bmatrix}} \overset{\text{dog}}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}$$

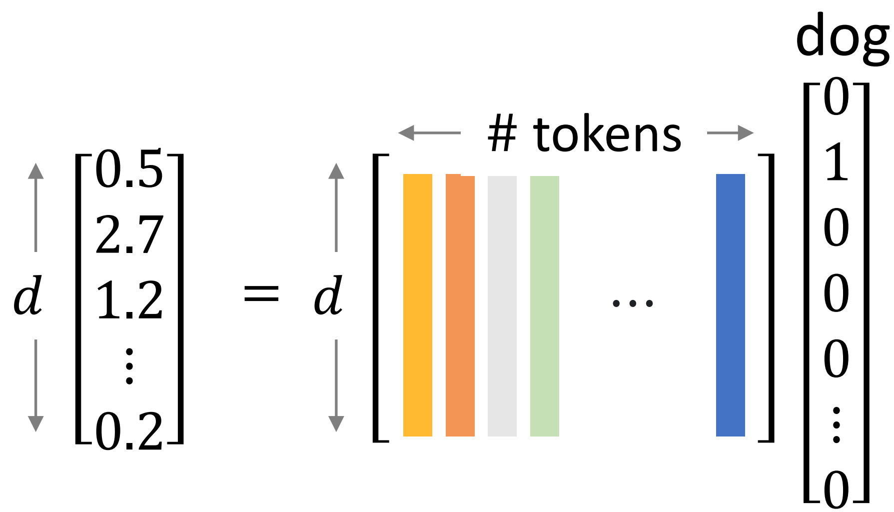
Embedded token

Embedding Matrix

TOKEN EMBEDDING



Embedding Space

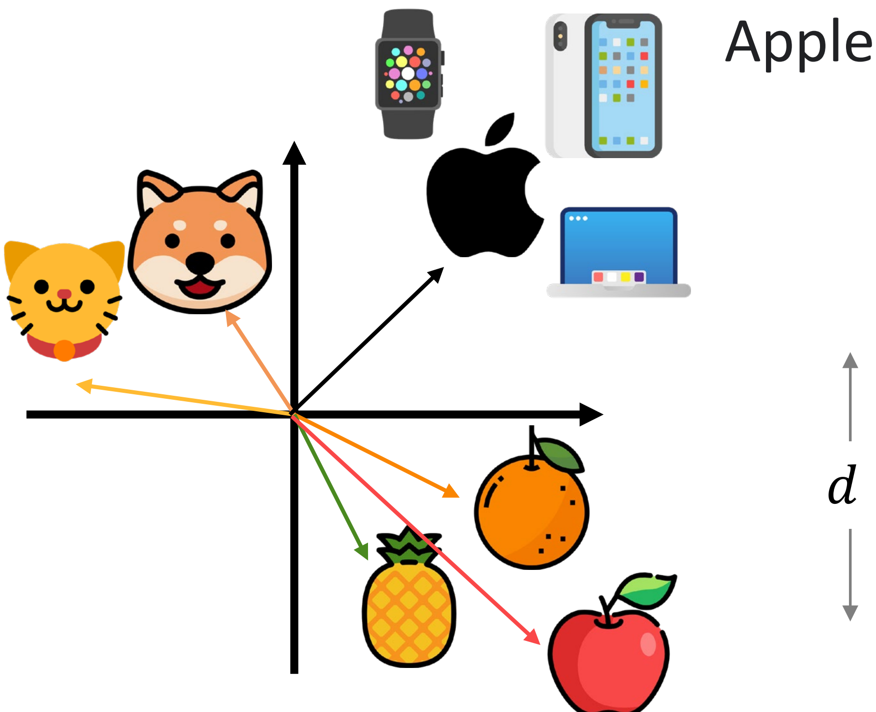


Embedded token

Embedding Matrix

Slide credit: Prof. J.-B. Huang @ U. Maryland

TOKEN EMBEDDING



Embedding Space

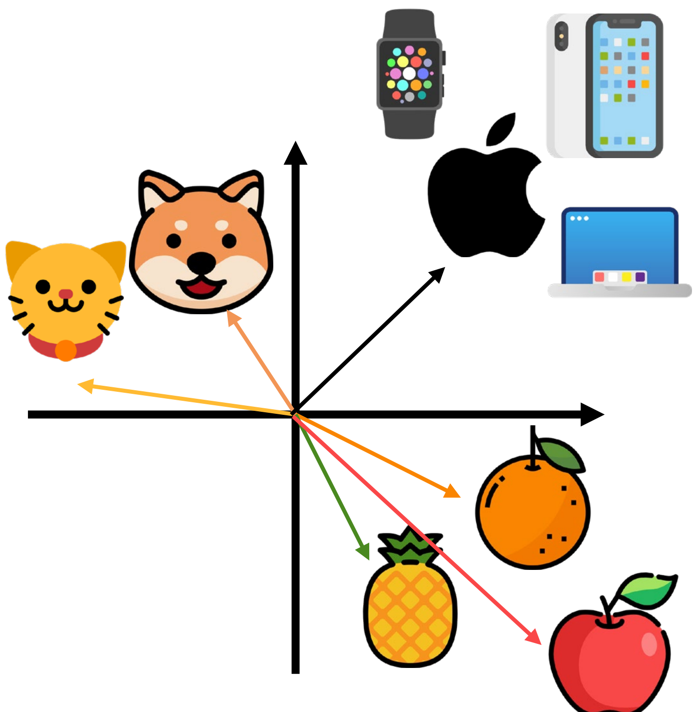
$$d \begin{bmatrix} 0.5 \\ 2.7 \\ 1.2 \\ \vdots \\ 0.2 \end{bmatrix} = d \overbrace{\begin{bmatrix} \; & \; & \; & \; & \cdots & \; \end{bmatrix}}^{\#\ \text{tokens}} \begin{array}{c} \text{dog} \\ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{array}$$

Embedded token

Embedding Matrix

Slide credit: Prof. J.-B. Huang @ U. Maryland

TOKEN EMBEDDING



Apple

Embedding Space

I bought an **apple** and an orange.

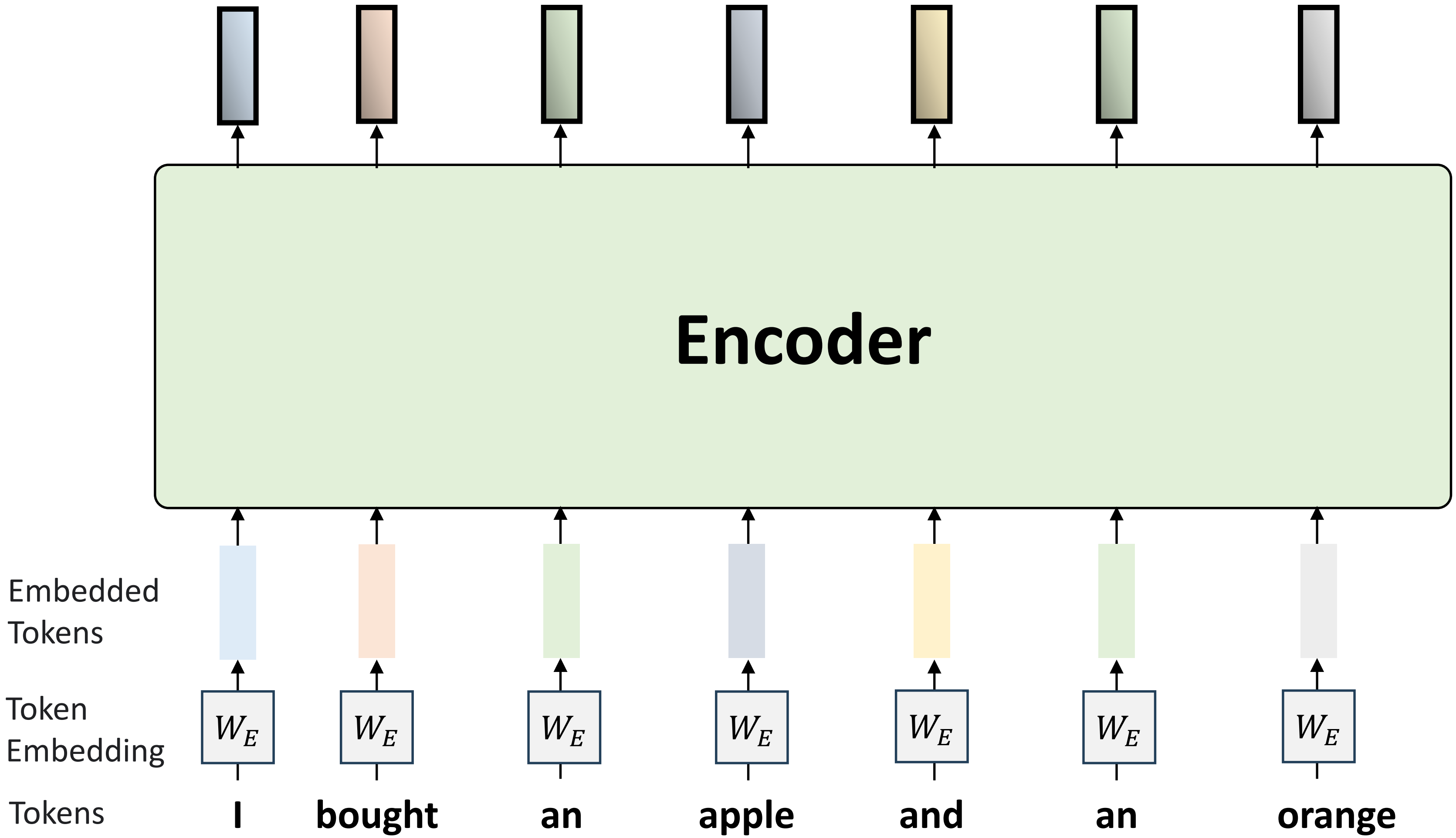
I bought an **apple** watch.

$$\begin{array}{c} \\ d \begin{bmatrix} 0.5 \\ 2.7 \\ 1.2 \\ \vdots \\ 0.2 \end{bmatrix} \end{array} = d \overbrace{\begin{bmatrix} \; \cdots \; \end{bmatrix}}^{\#\ \text{tokens}} \begin{array}{c} \text{dog} \\ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{array}$$

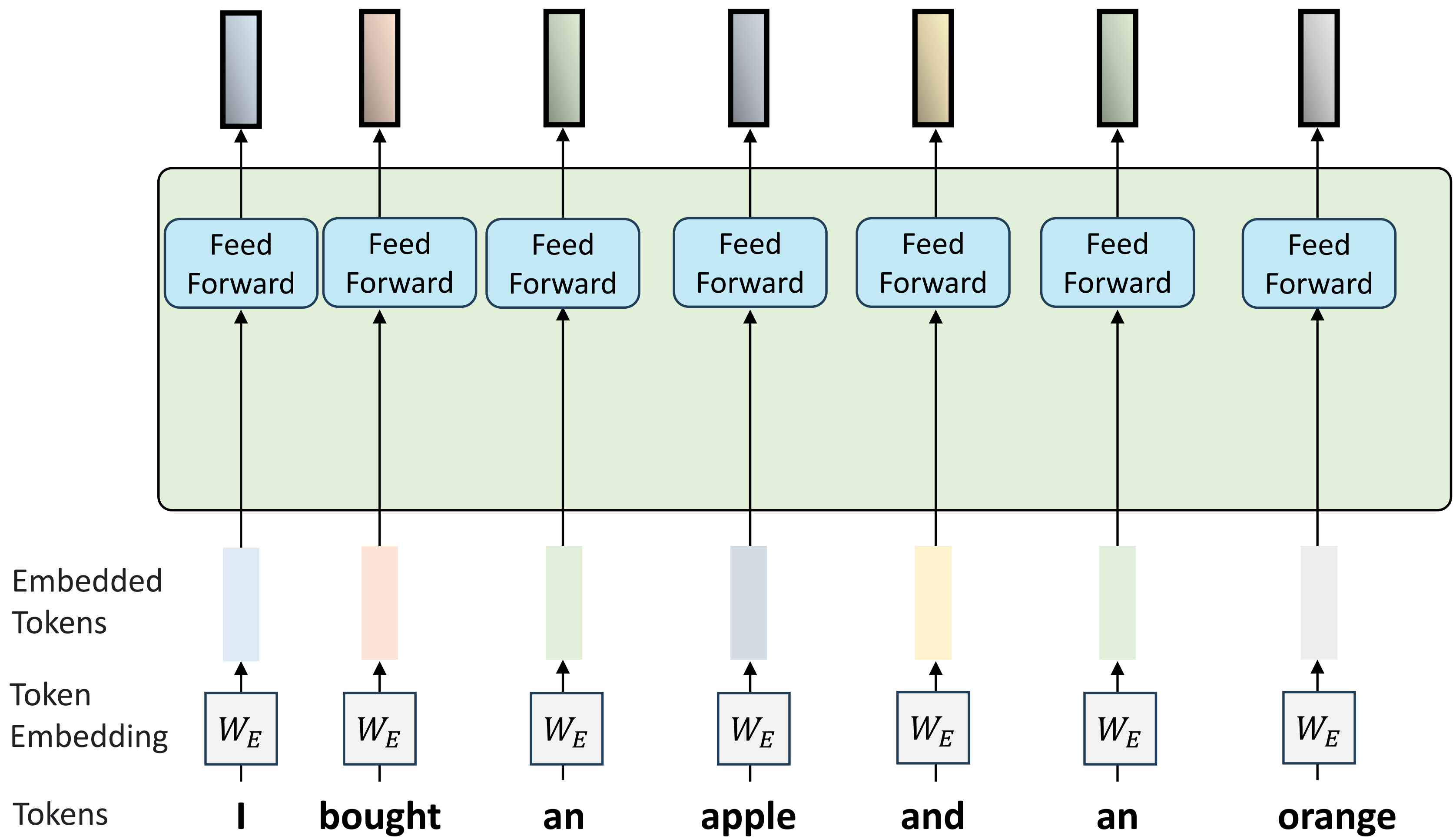
Embedded token

Embedding Matrix

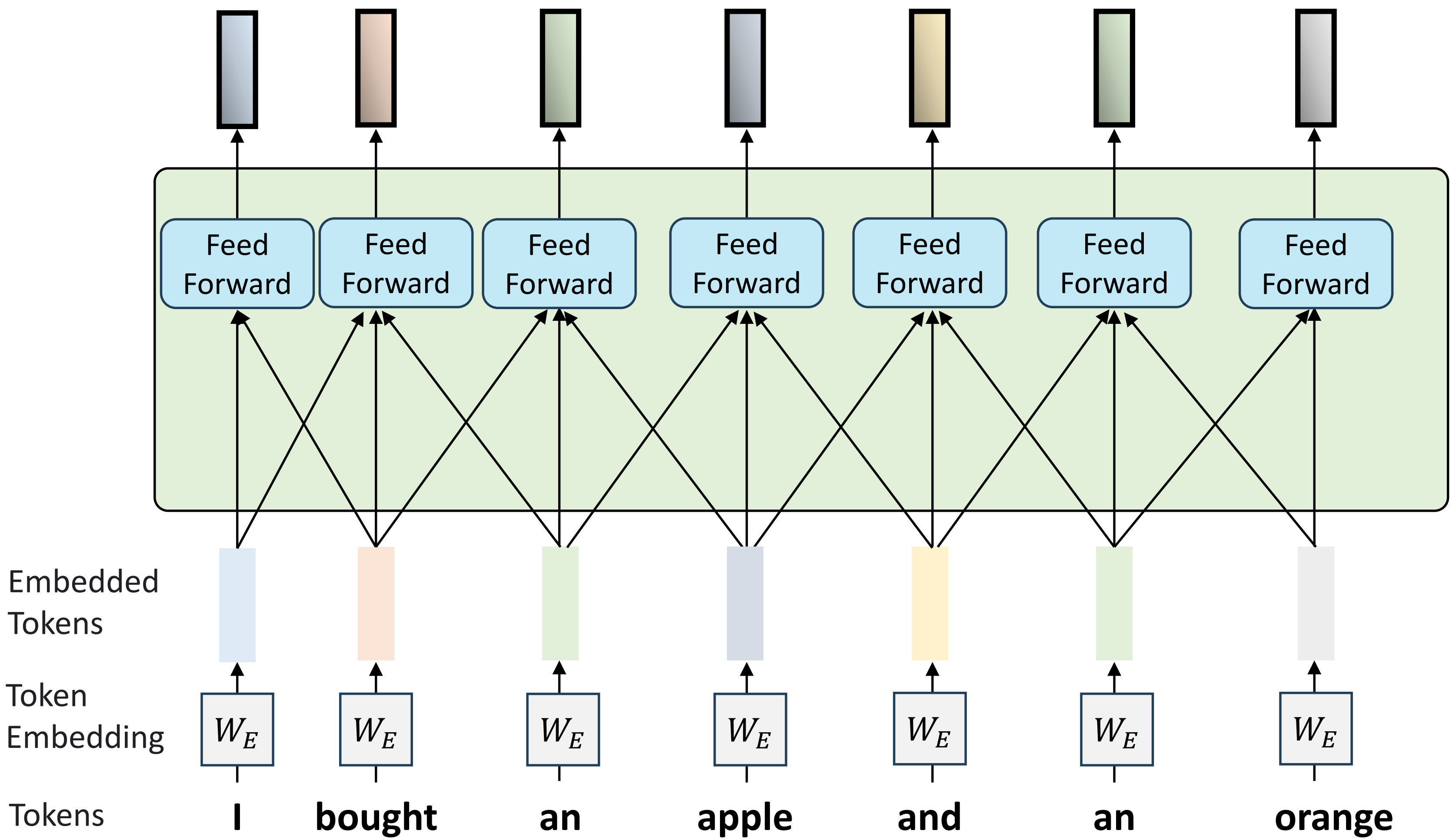
Slide credit: Prof. J.-B. Huang @ U. Maryland



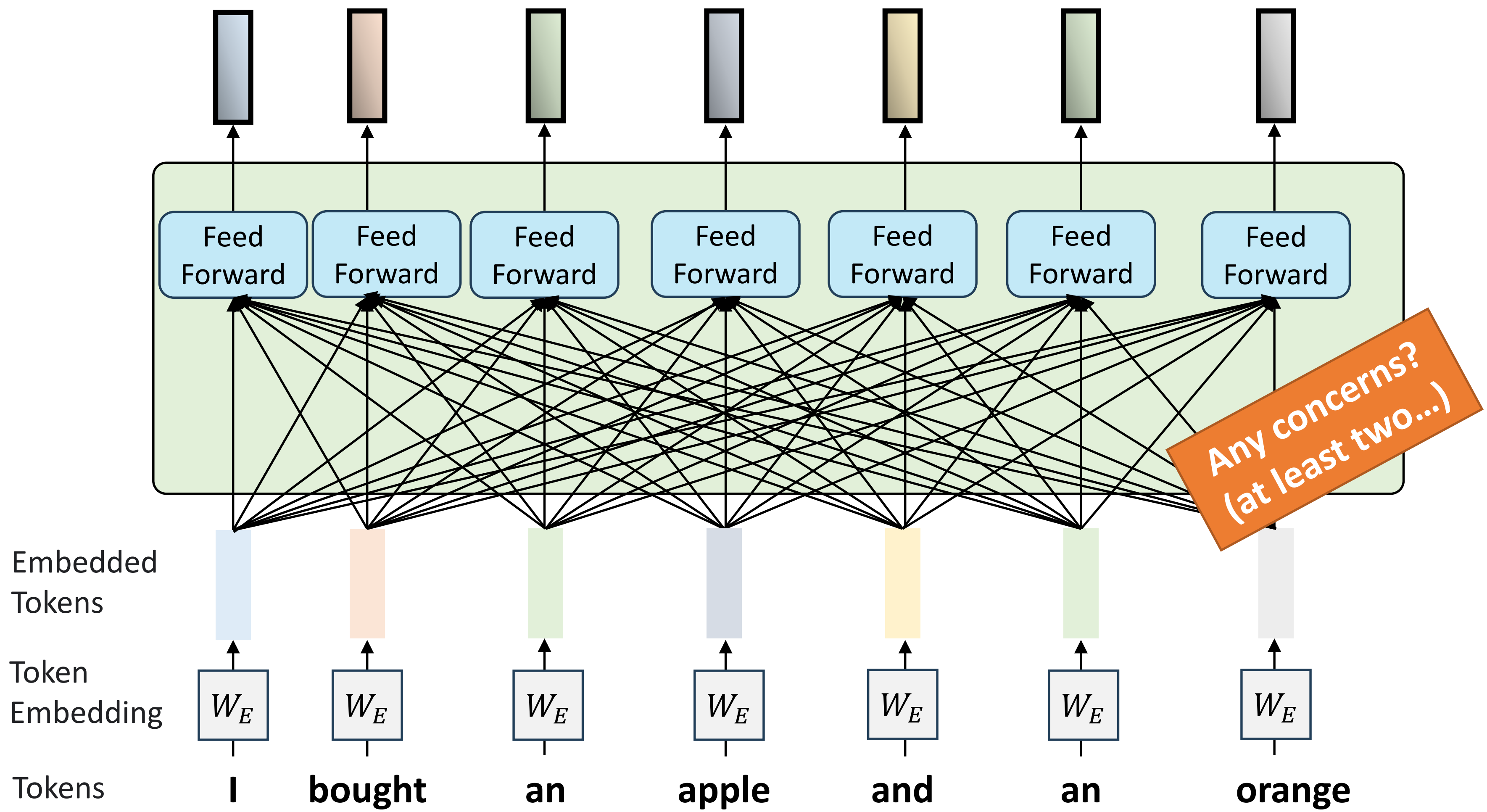
Slide credit: Prof. J.-B. Huang @ U. Maryland



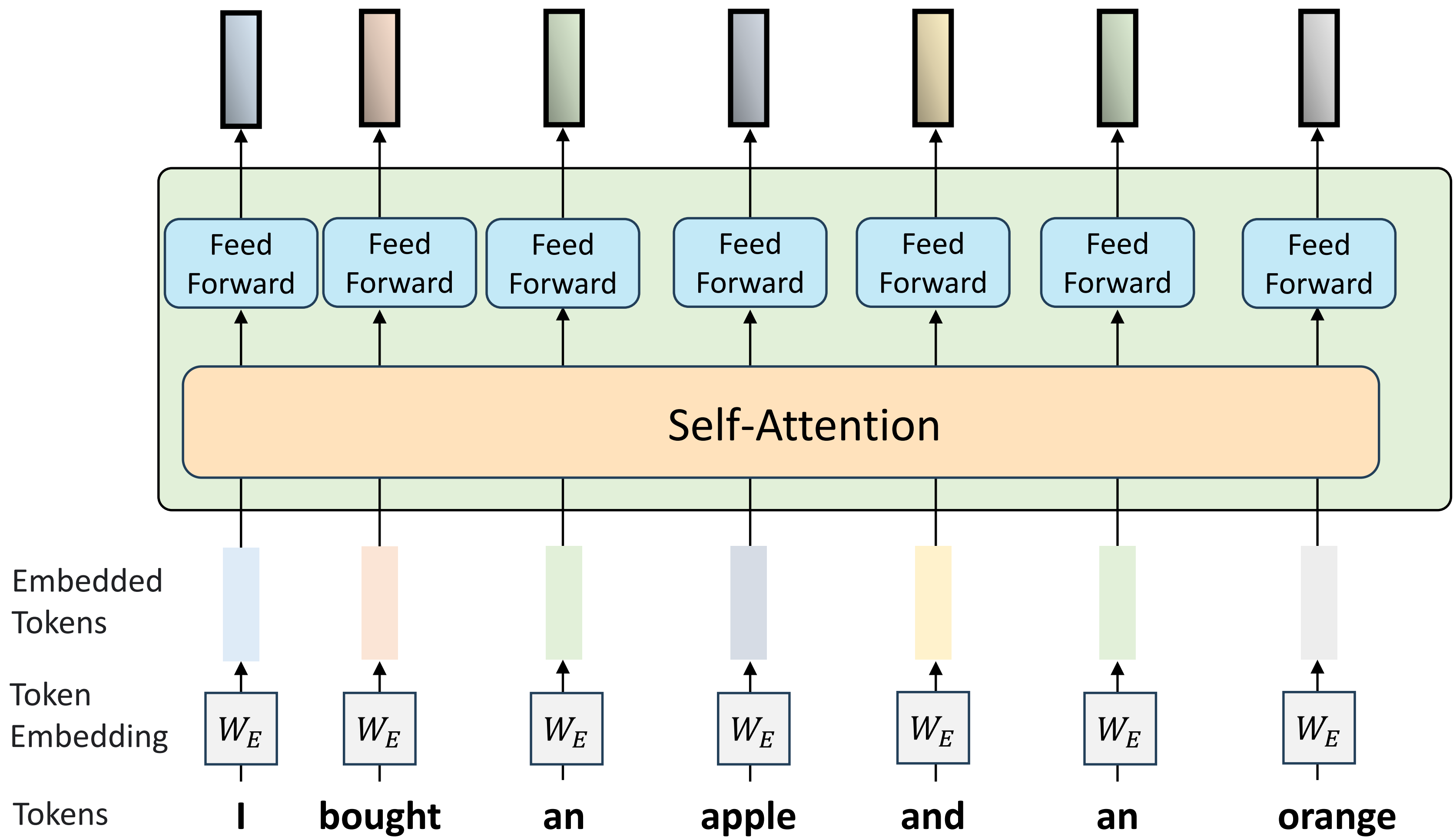
Slide credit: Prof. J.-B. Huang @ U. Maryland



Slide credit: Prof. J.-B. Huang @ U. Maryland

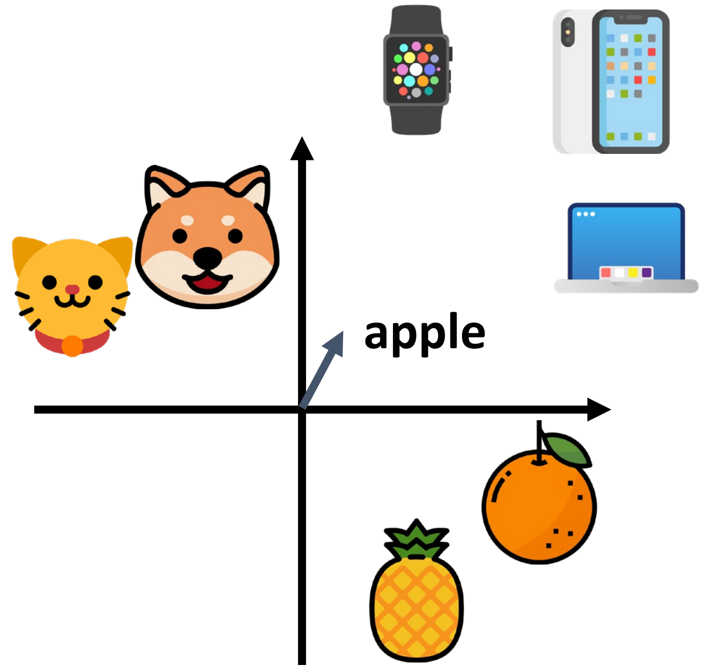


Slide credit: Prof. J.-B. Huang @ U. Maryland



Slide credit: Prof. J.-B. Huang @ U. Maryland

Self-Attention

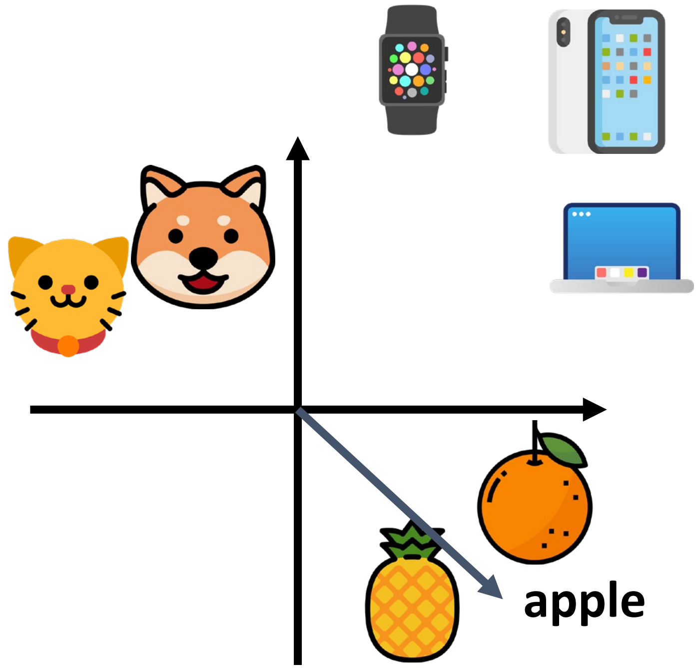


Embedding Space

Embedded Tokens							
Tokens	**I**	**bought**	**an**	**apple**	**and**	**an**	**orange**

Slide credit: Prof. J.-B. Huang @ U. Maryland

Self-Attention



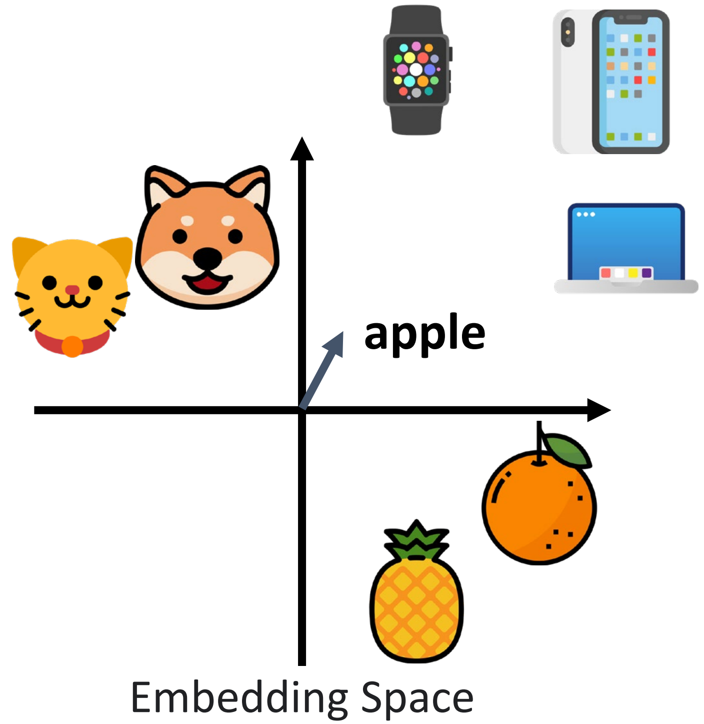
Embedding Space

Embedded Tokens

Tokens **I** **bought** **an** **apple** **and** **an** **orange**

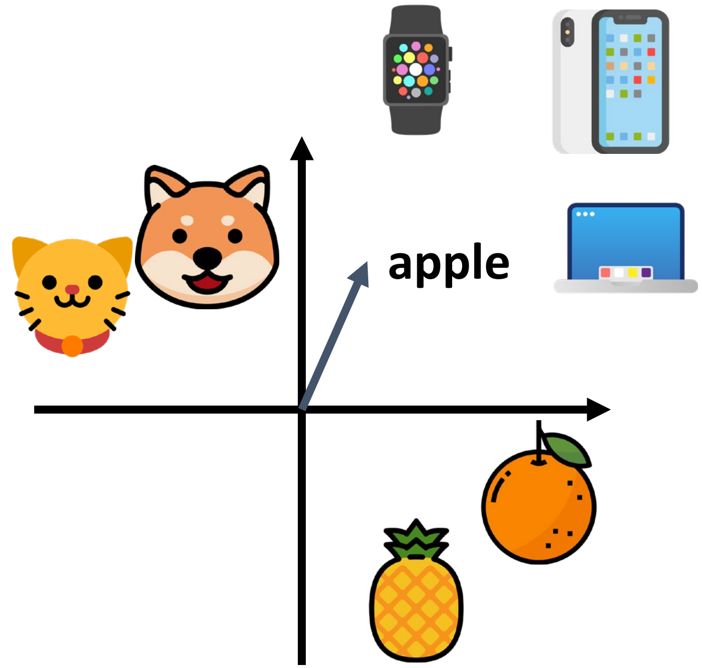
Slide credit: Prof. J.-B. Huang @ U. Maryland

Self-Attention



Slide credit: Prof. J.-B. Huang @ U. Maryland

Self-Attention



Embedding Space

Embedded Tokens

Tokens **I** **bought** **an** **apple** **watch**

Slide credit: Prof. J.-B. Huang @ U. Maryland

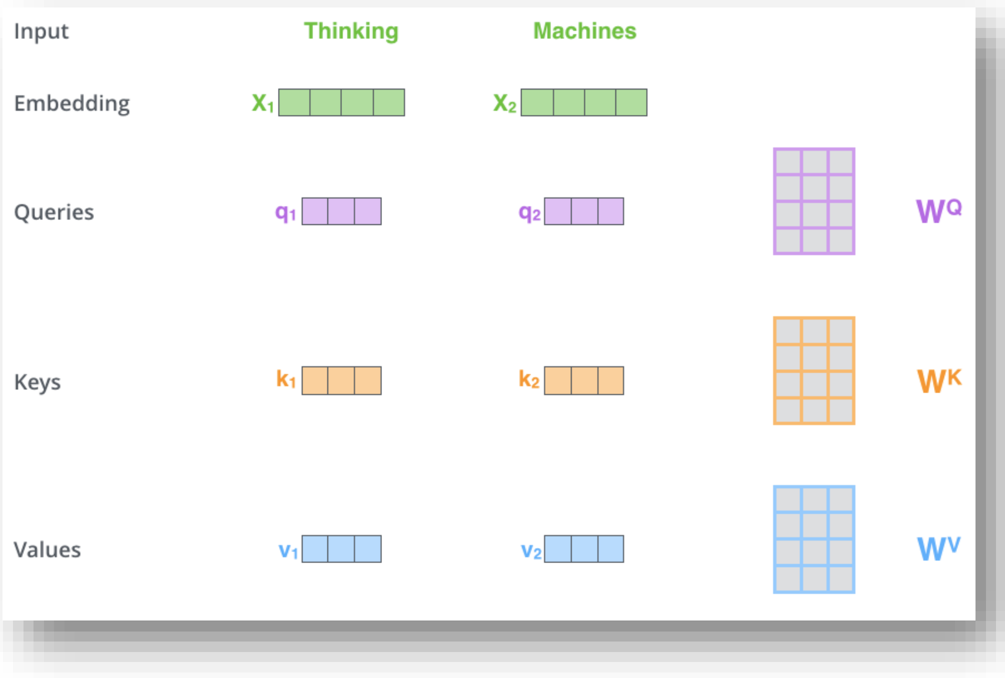
Self-Attention (1/5)

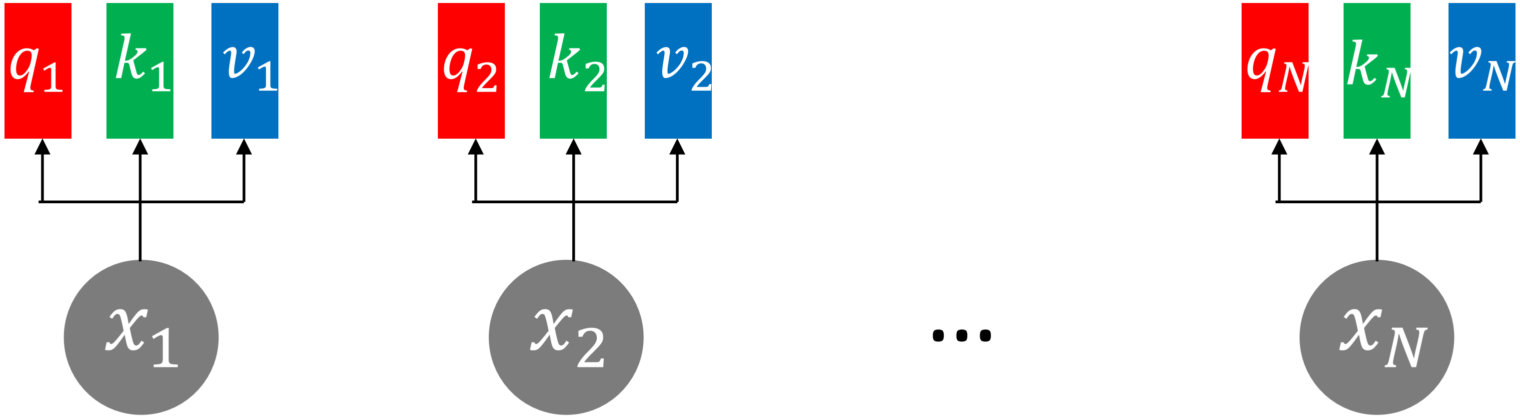
- Query $\boldsymbol{q}$, key $\boldsymbol{k}$, value $\boldsymbol{v}$ vectors are learned from each input $\boldsymbol{x}$

$$q_i = W^Q x_i$$

$$k_i = W^K x_i$$

$$v_i = W^V x_i$$

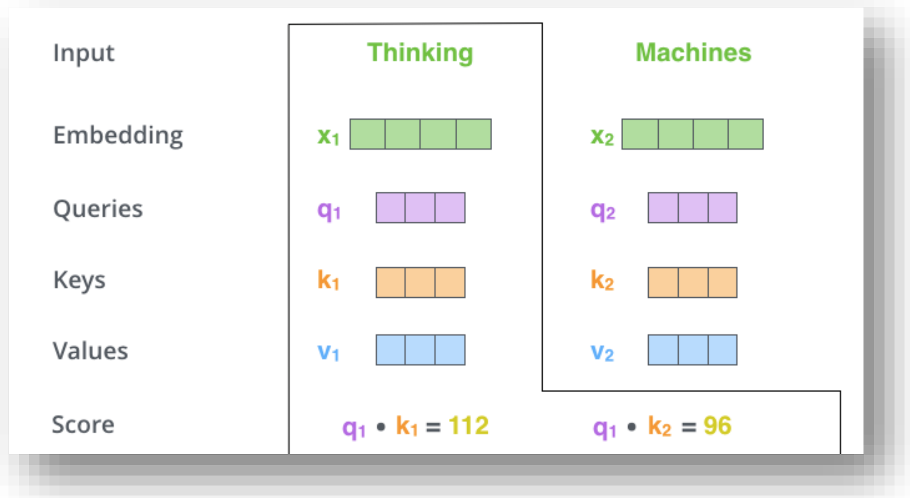


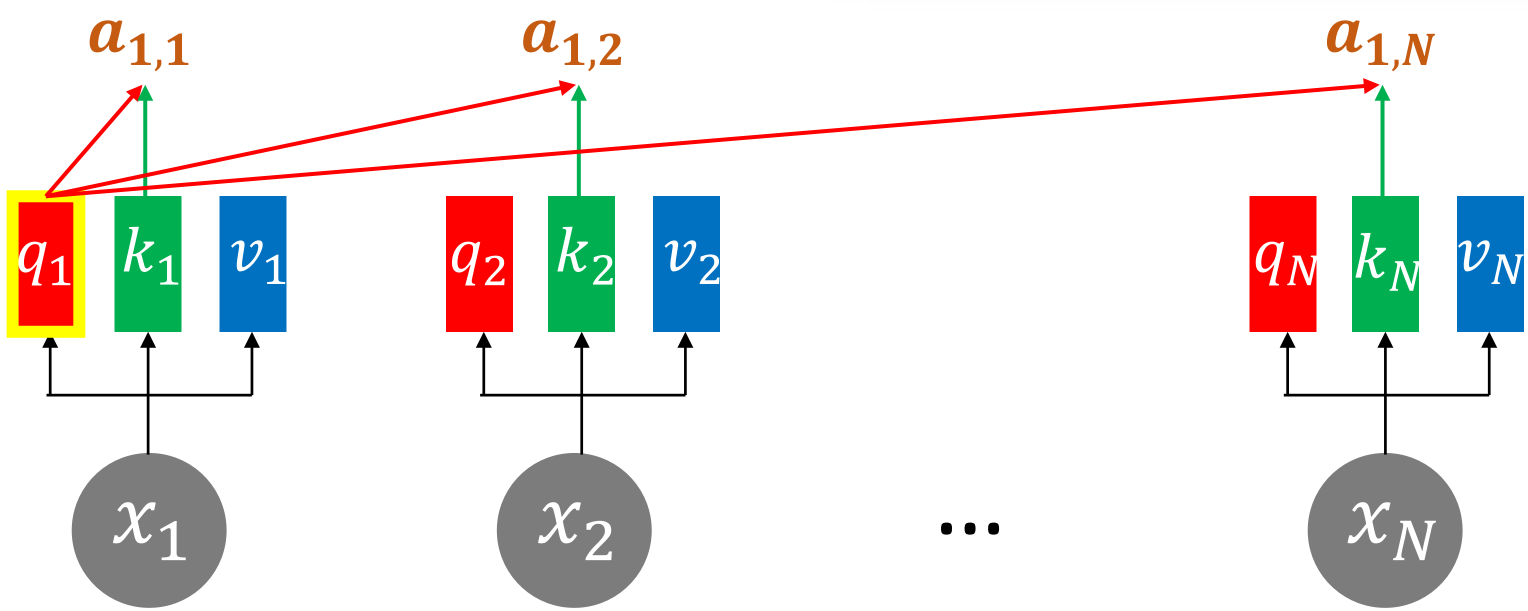


Self-Attention (2/5)

- Relation between each input is modeled by inner-product of query $\boldsymbol{q}$ and key $\boldsymbol{k}$.

$$a_{1,i} = \frac{q_1 \cdot k_i}{\sqrt{d}}, \text{ where } a \in R, q, k \in R^d$$

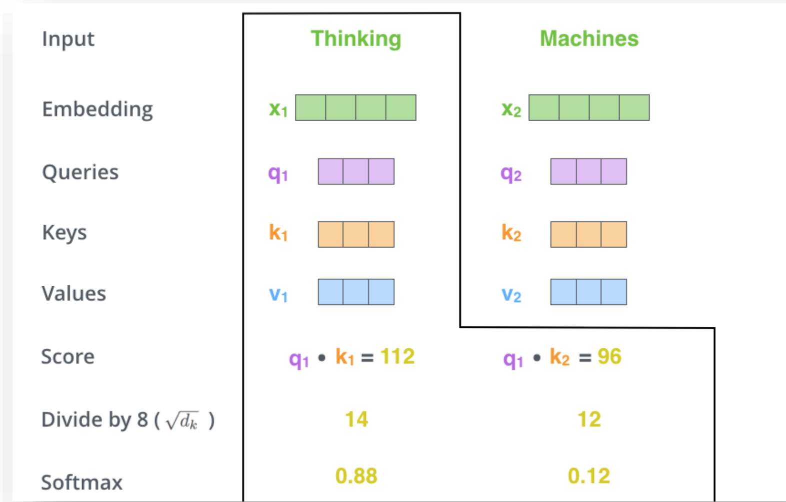


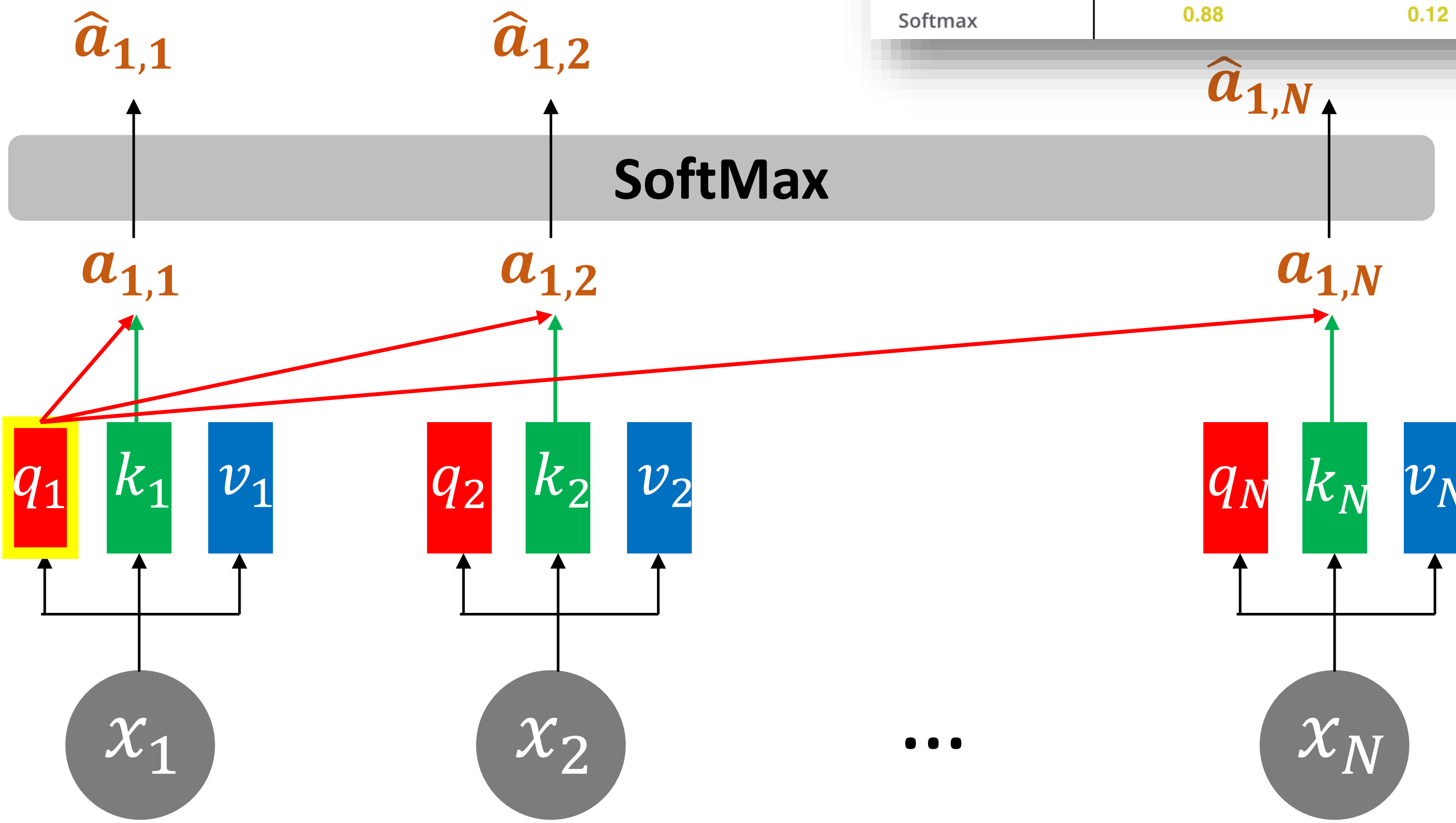


Self-Attention (3/5)

- SoftMax is applied:

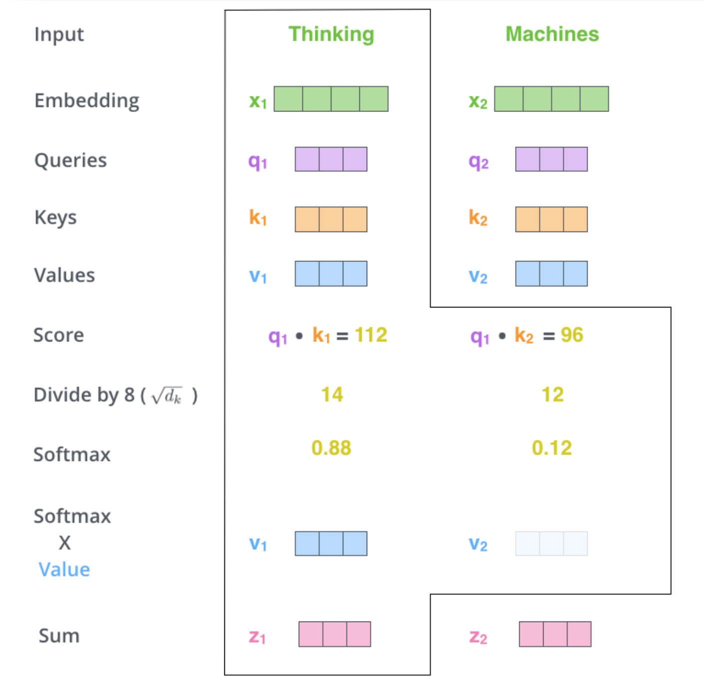
  $0 \le \hat{a}_i = e^{a_i} / \sum_j^{\mathrm{N}} e^{a_j} \le 1$, for I =1, …, N

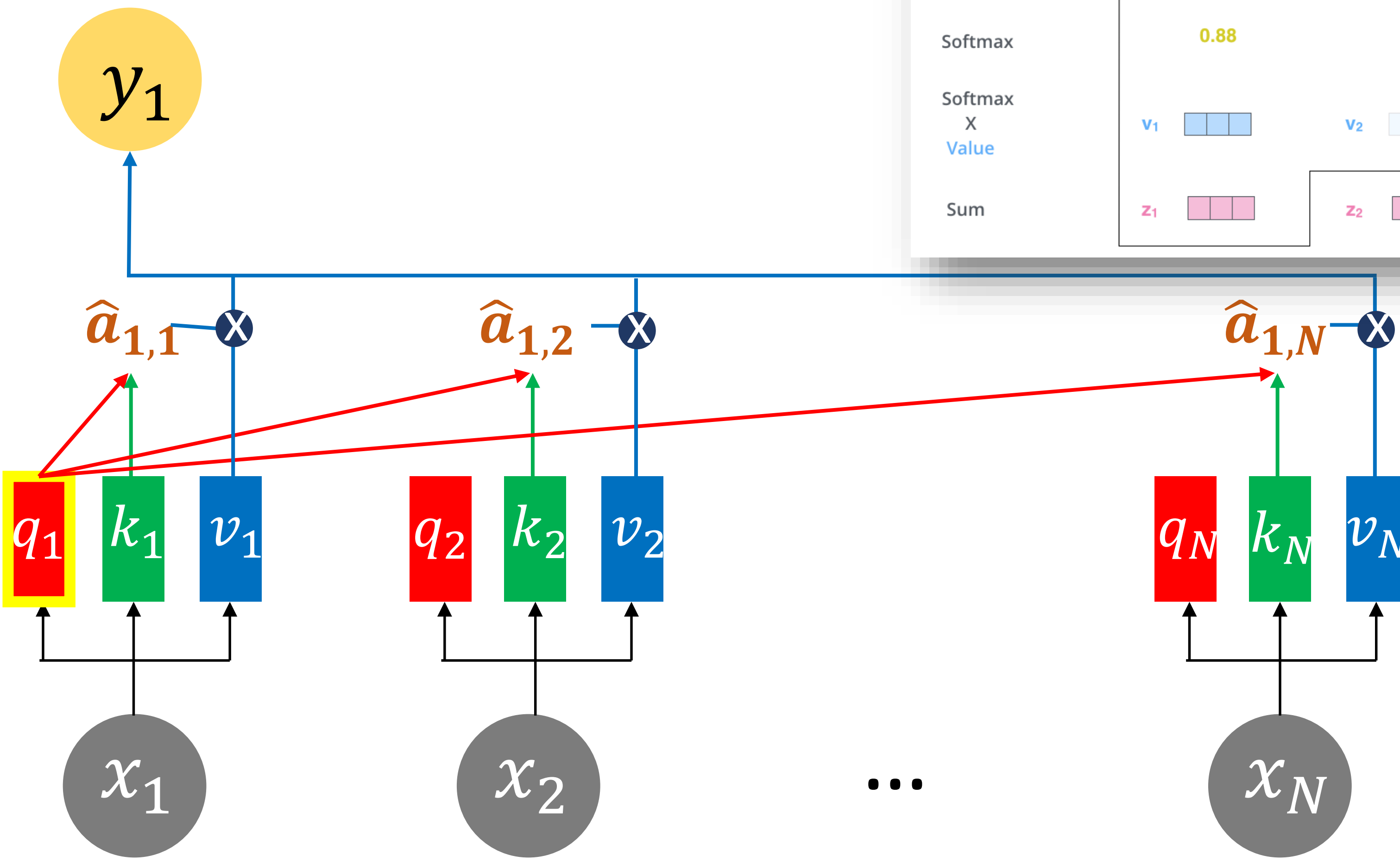




Self-Attention (4/5)

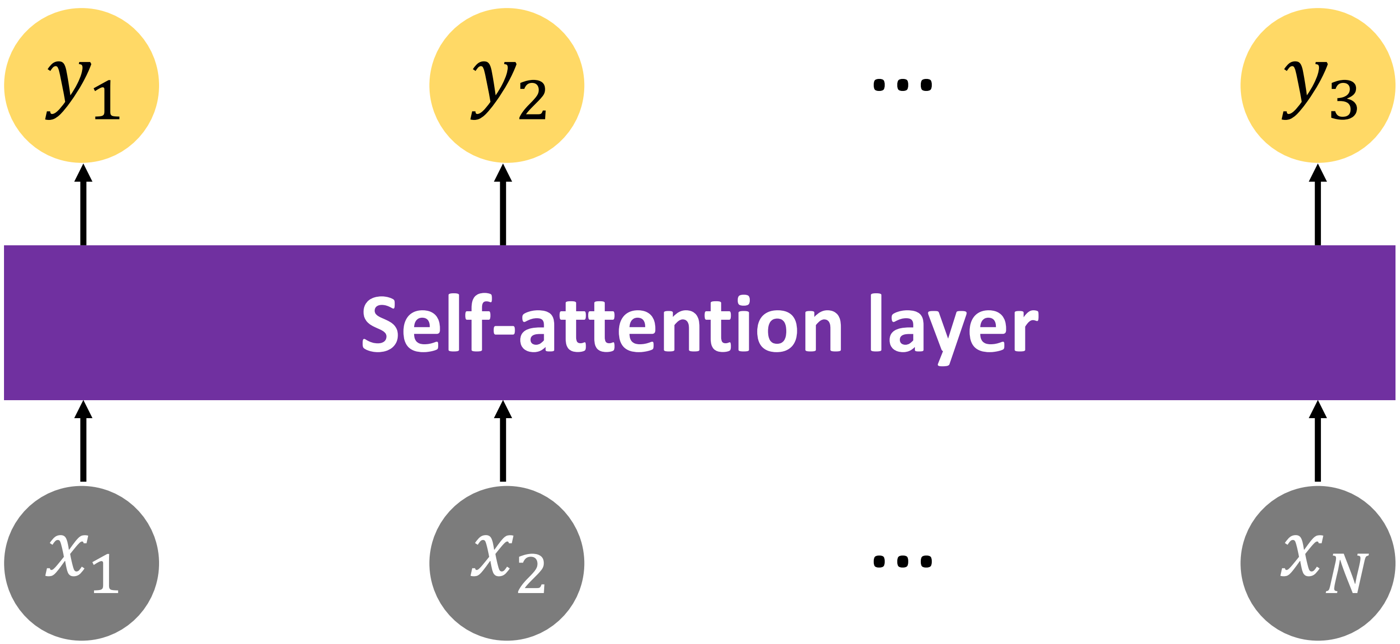
- Value vectors ***v*** are aggregated with attention weight $\hat{a}$, i.e., $y_1 = \sum_i^N \hat{a}_i \cdot v_i$



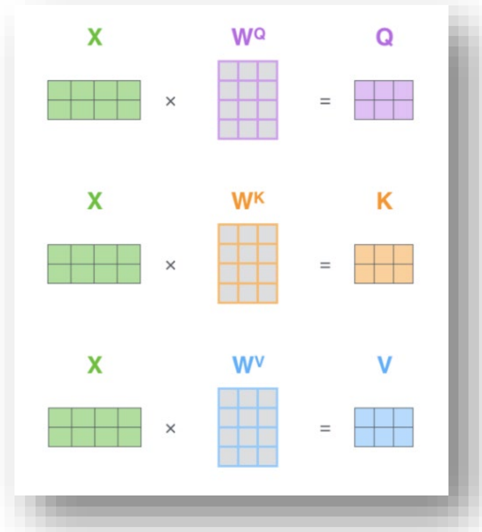


Self-Attention (5/5)

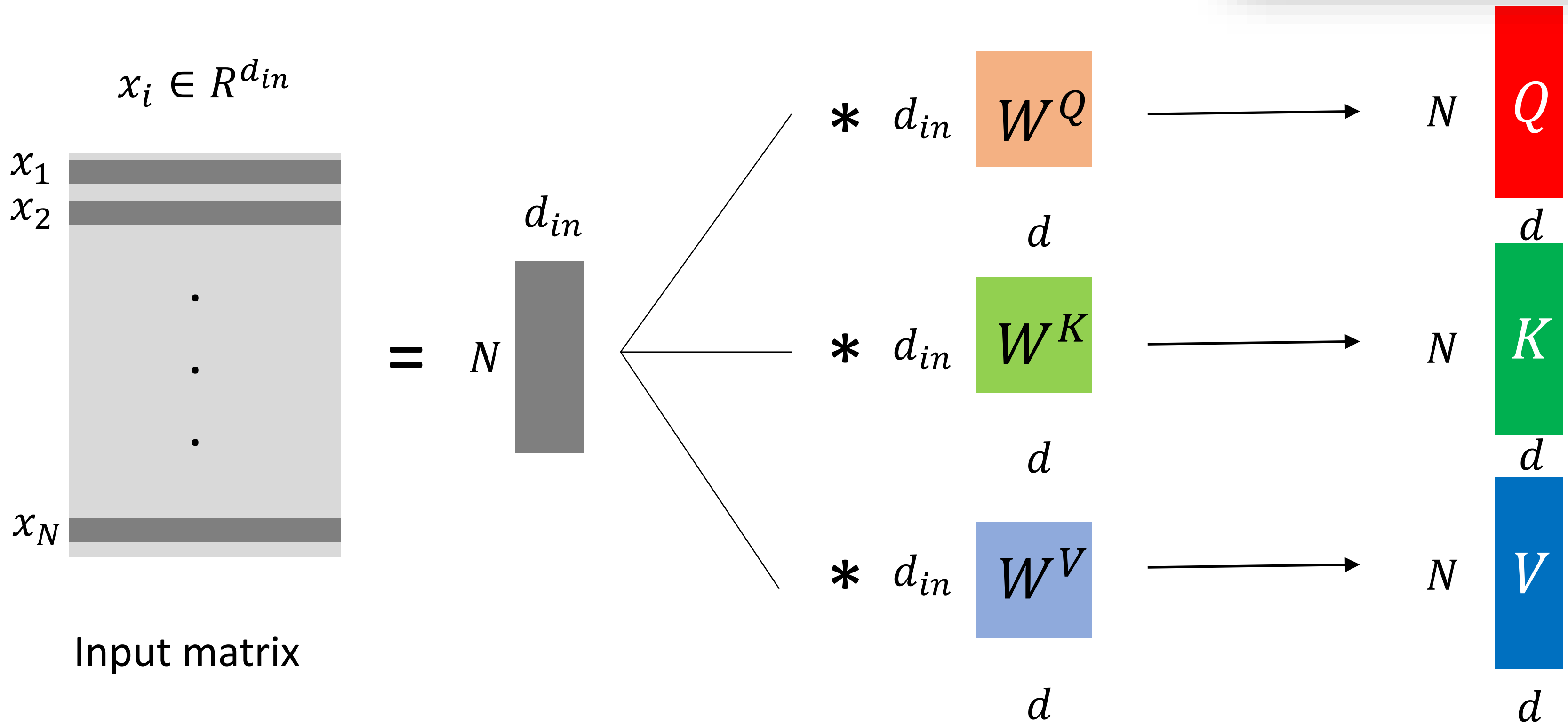
- All y_i can be computed **in parallel**
- Each y_i considers $x_1 \sim x_N$, modeling their **long-distance dependencies**.
- Global feature can be obtained by **average-pooling** over $y_1 \sim y_N$



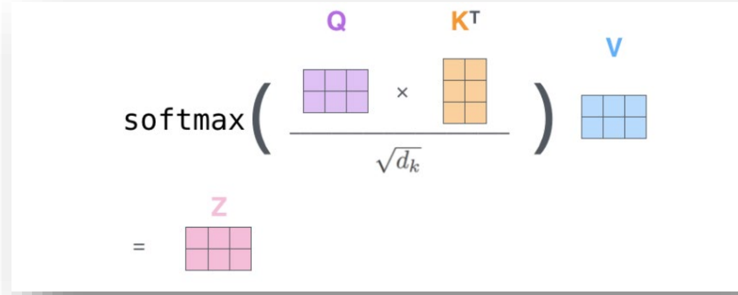
Self-Attention: Implementation



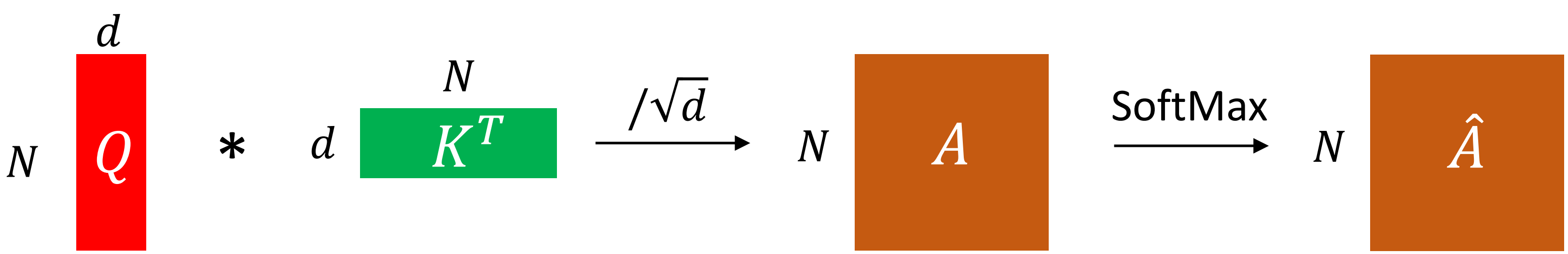
- Input sequence can be represented as a N x d_{in} matrix
- * denotes matrix multiplication

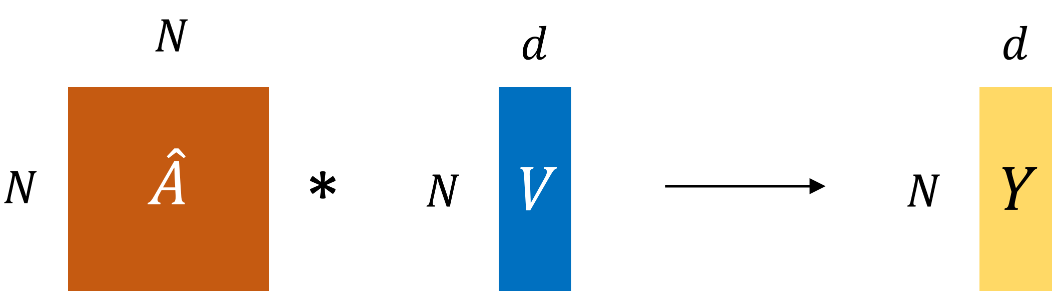


Self-Attention: Implementation



- Output matrix **Y**
- All operations are **matrix multiplication**, can be parallelized on GPU.

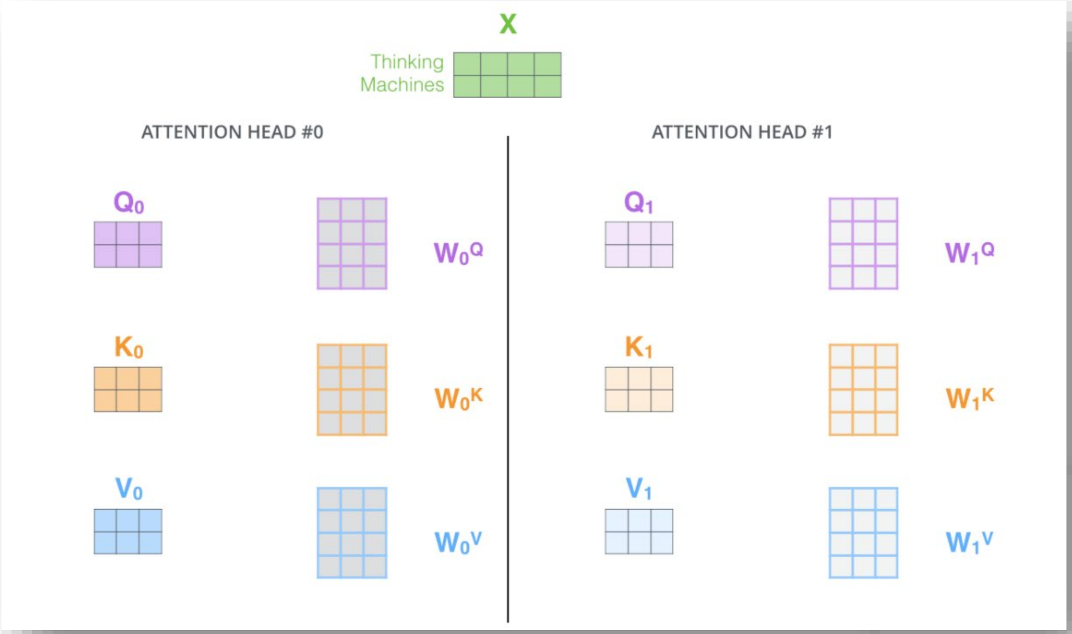


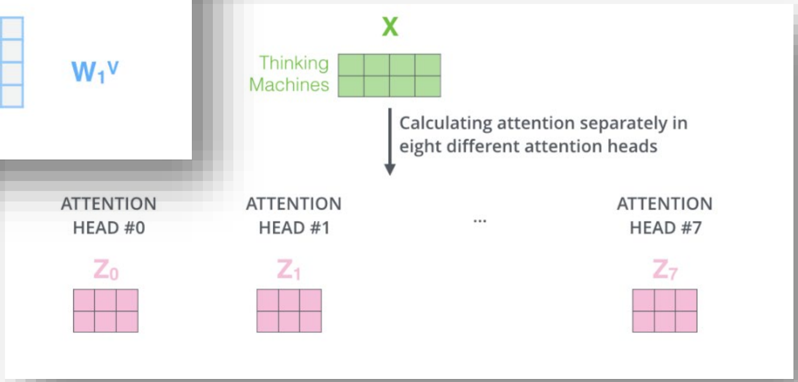


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

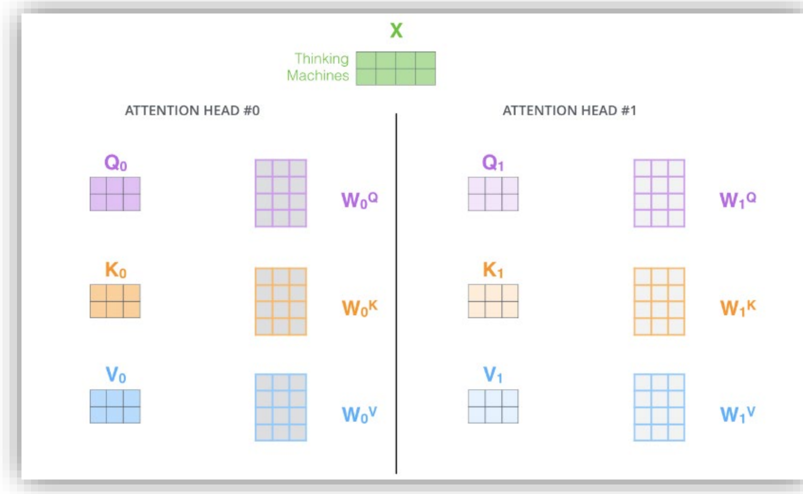
Multi-Head Self-Attention (1/4)

- Perform self-attention at different subspaces, implying performing attention over different input feature types (e.g., representations, modalities, positions, etc.)

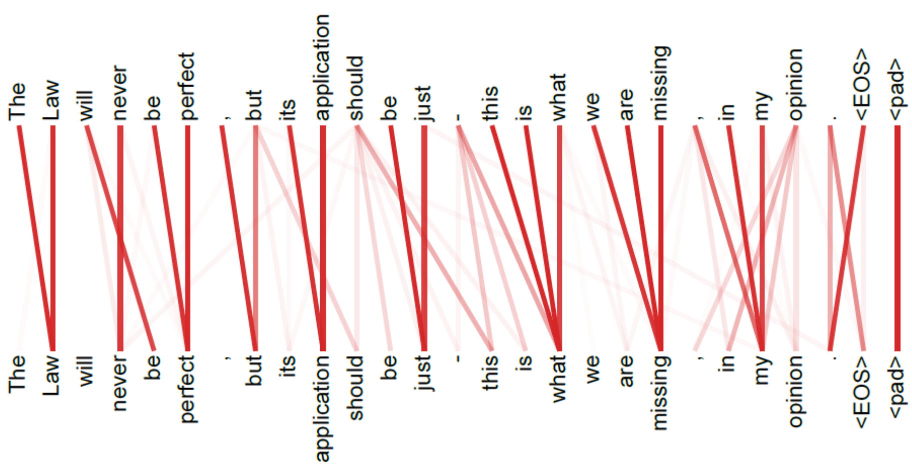




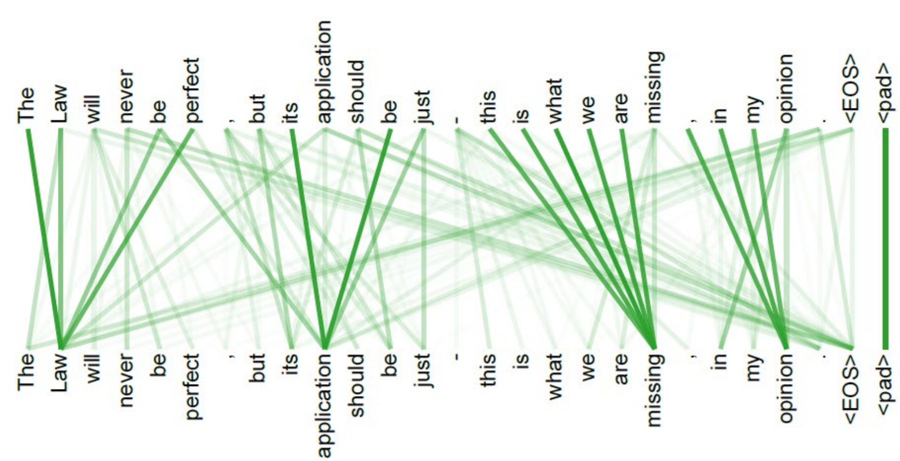
Multi-Head Self-Attention (2/4)



- Perform self-attention at different subspaces, implying performing attention over different input feature types
- See example below



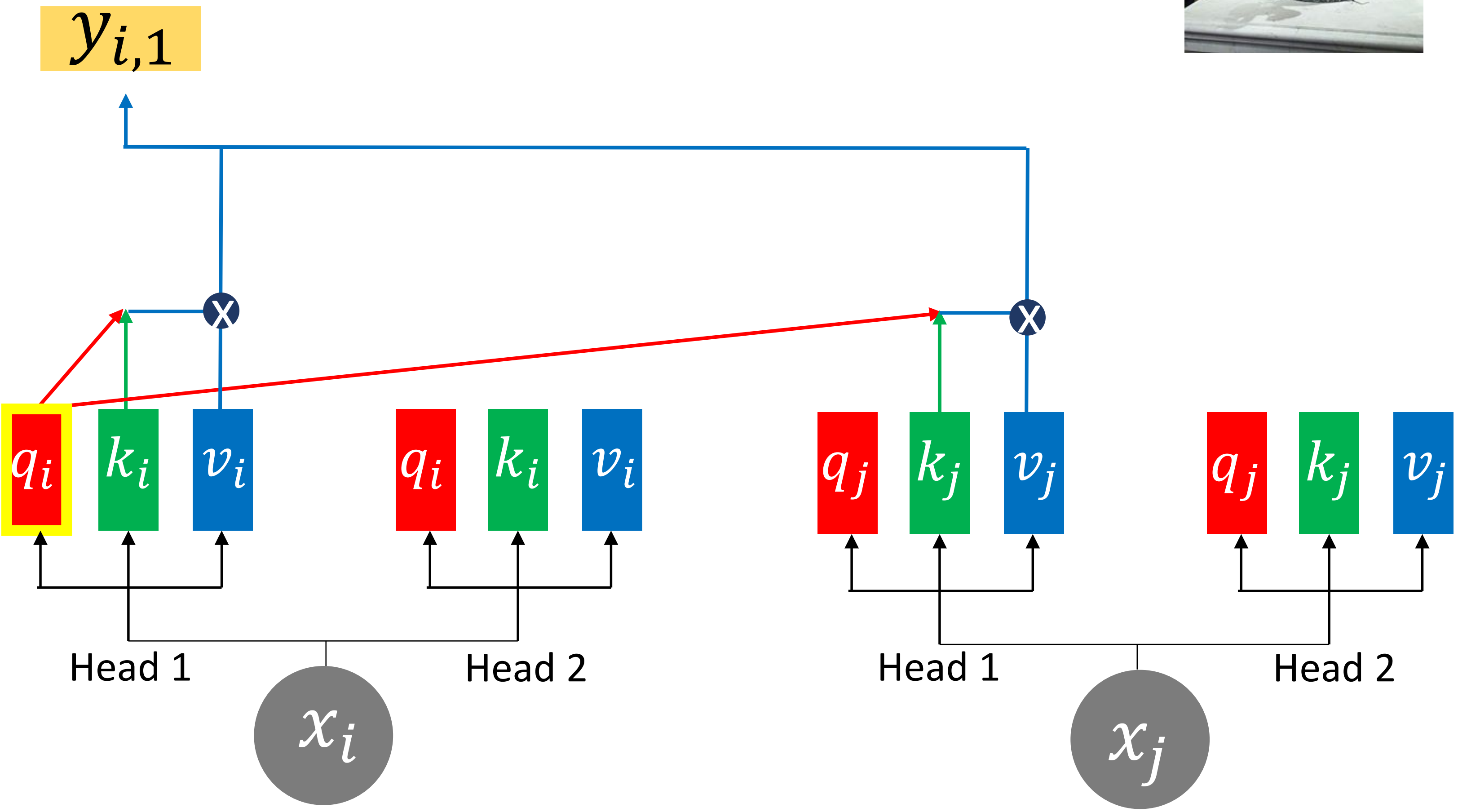
Attention weights of Head 1



Attention weights of Head 2

Multi-Head Self-Attention (3/4)

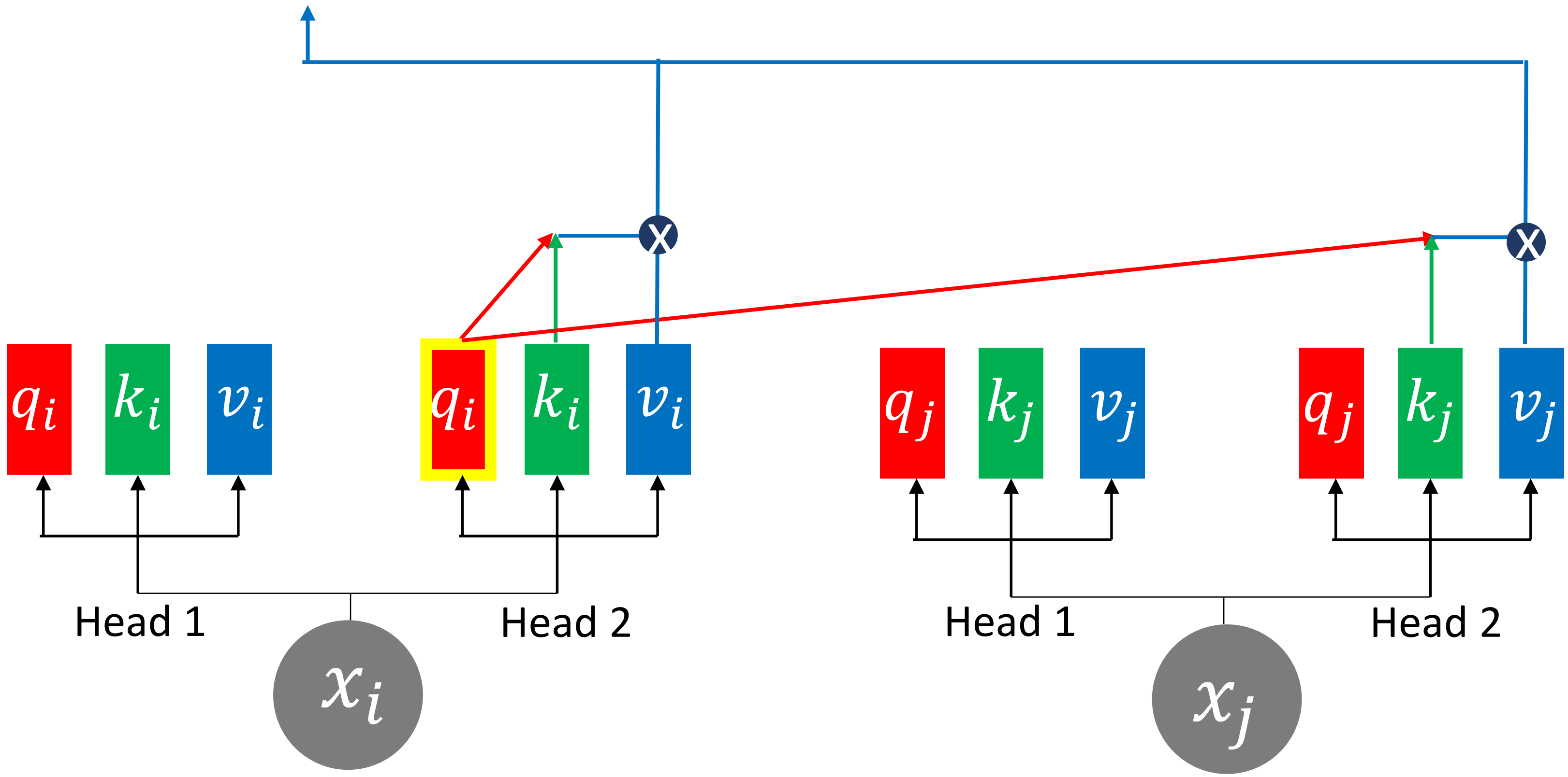
- A 2-head example, output of two heads are concatenated.

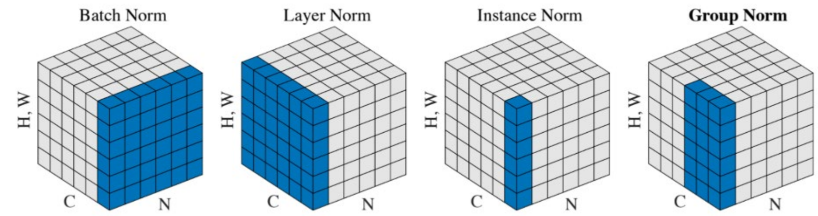


Multi-Head Self-Attention (4/4)

- A 2-head example, output of two heads are concatenated.

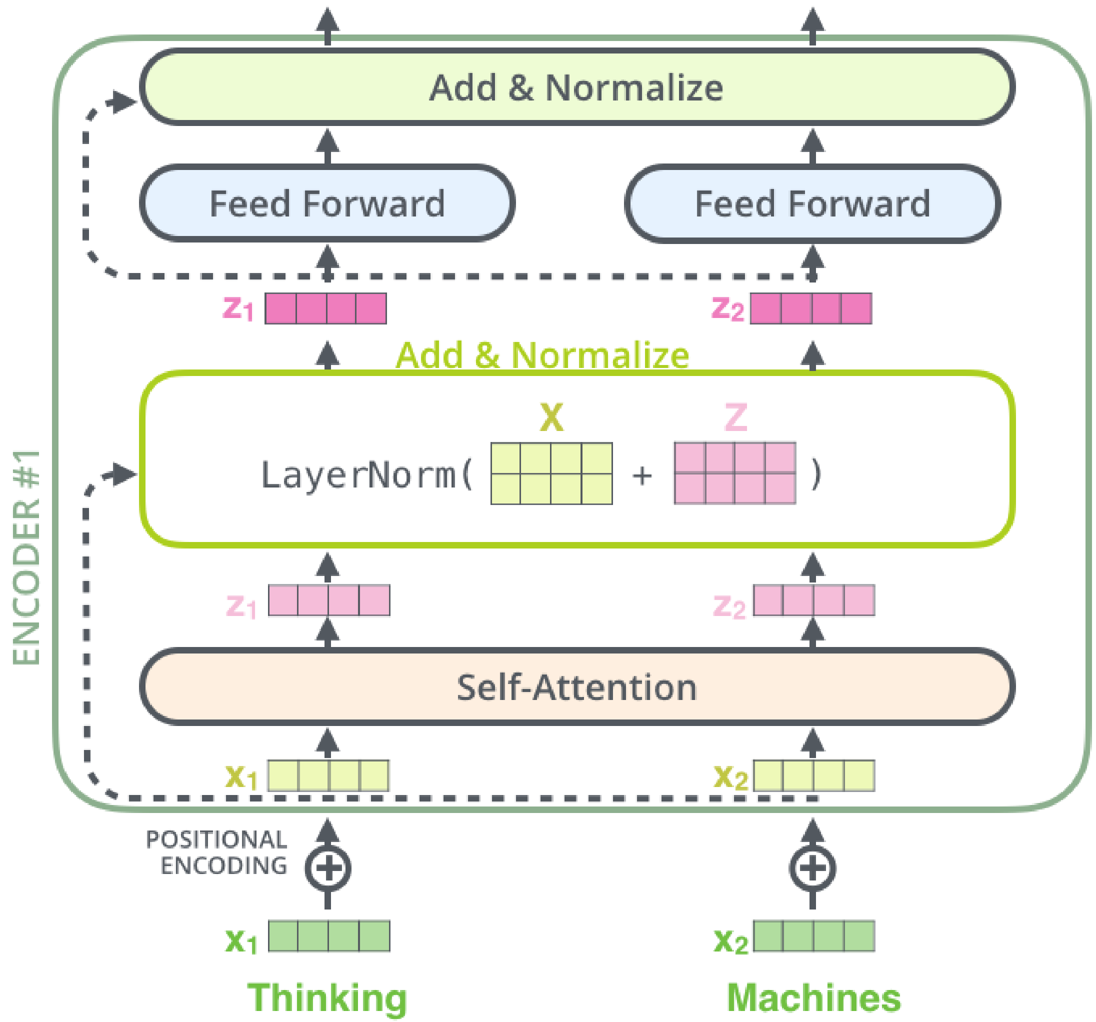
$$y_{i,1} \; y_{i,2} = y_i$$





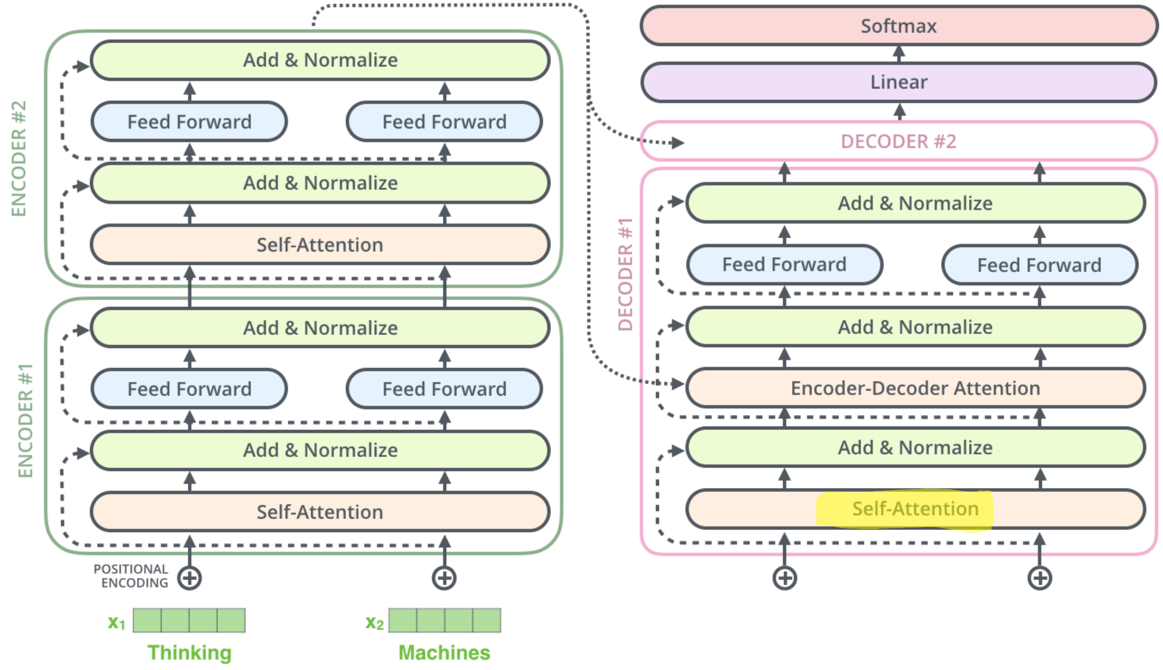
The Residuals

- A **residual connection** followed by layer normalization



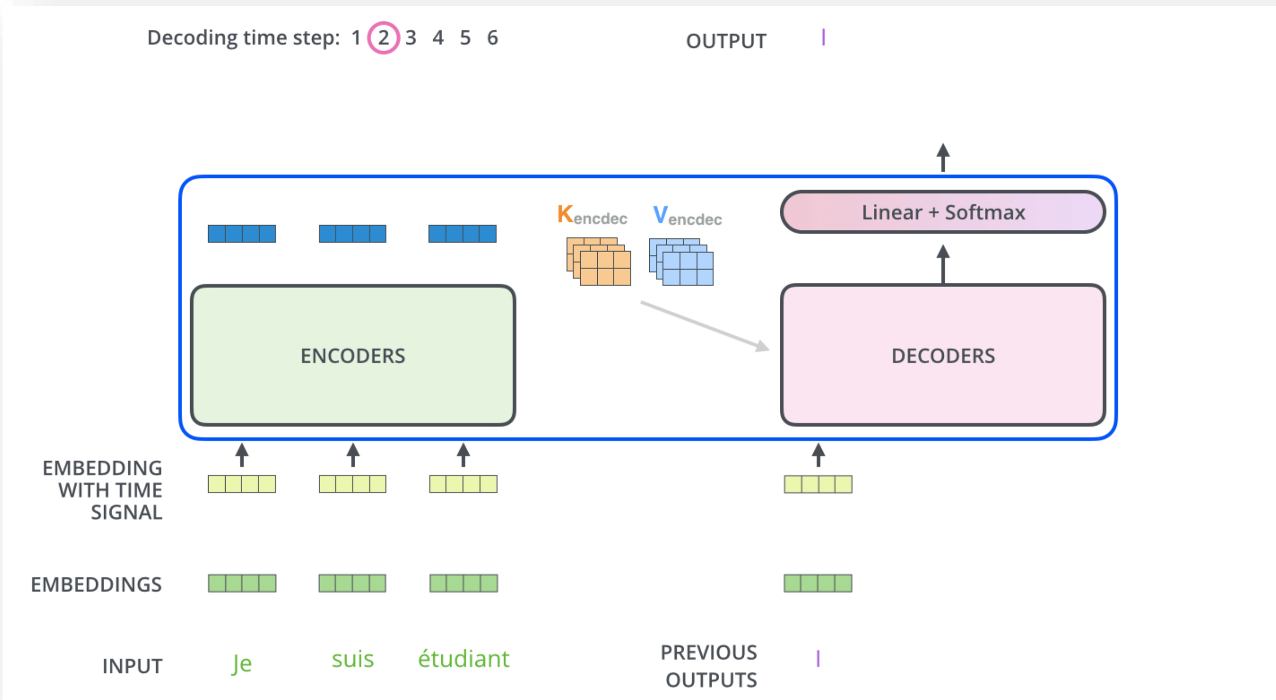
The Decoder in Transformer

- Encoder-decoder attention
  - Q from self-attn in decoder, K & V from encoder outputs
- Masked multi-head attention
  - Design similar to that of encoder, except for decoder #1 which takes additional inputs (of GT/predicted word embeddings).
  - Mask unpredicted tokens during softmax: why?



The Decoder in Transformer (cont'd)

- Encoder-decoder attention
  - Q from self-attn in decoder, K & V from encoder outputs
- Masked multi-head attention
  - Design similar to that of encoder, except for decoder #1 which takes additional inputs (of GT/predicted word embeddings).
  - Mask unpredicted tokens during softmax: why?



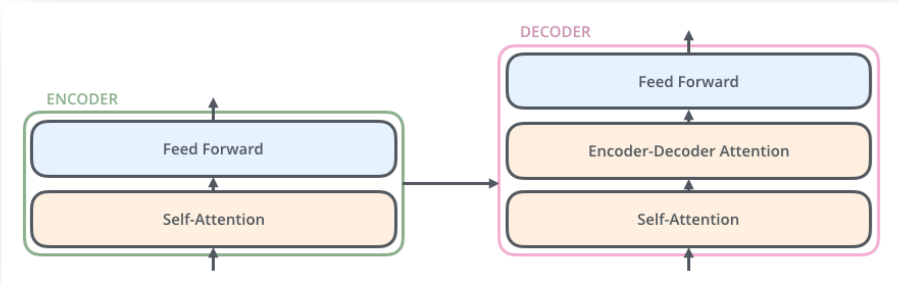
Overview of Decoding in Transformer

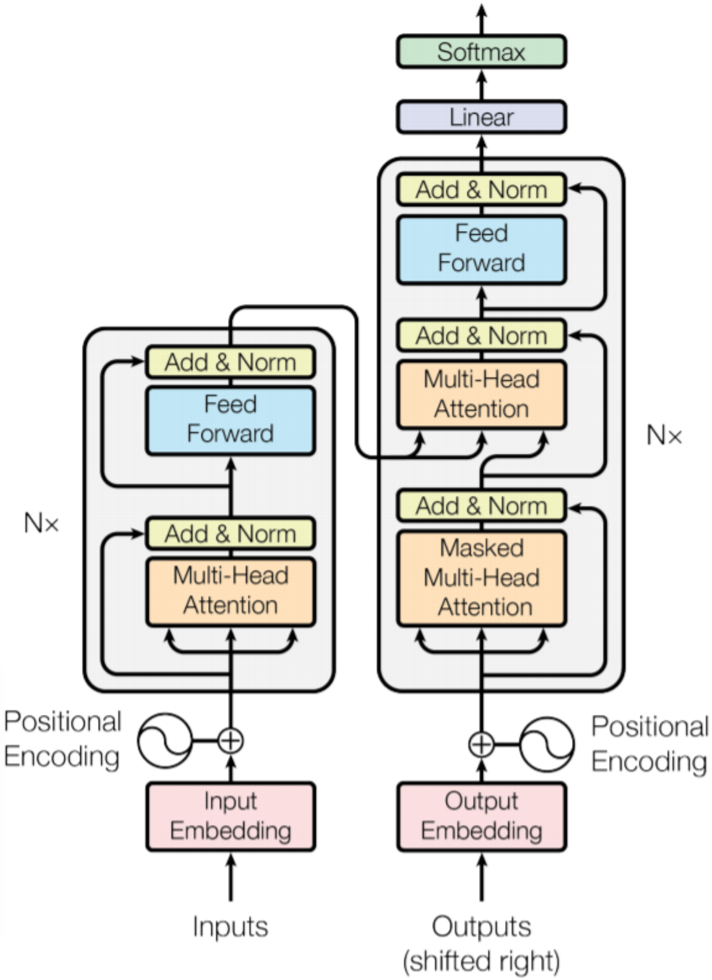
- Encoder/Decoder Cross-Attention + Decoder self-attention

https://medium.com/ml-note/autoencoder-%E4%B8%89-self-attention-transformer-c37f719d222

Recap: Transformer

- "Attention is all you need", NeurIPS 2017
- We didn't cover **positional encoding** (particularly for language translation)
- Potential problems of Transformer?





What We’ve Covered Today…

- Generative Model
  - Generative Adversarial Network
- Adversarial Learning for Transfer Learning
- Recurrent Neural Networks
  - From RNN to LSTM & GRU
  - Sequence-to-Sequence Learning
  - Attention in RNN
- Transformer