Attention Models and Memory Networks

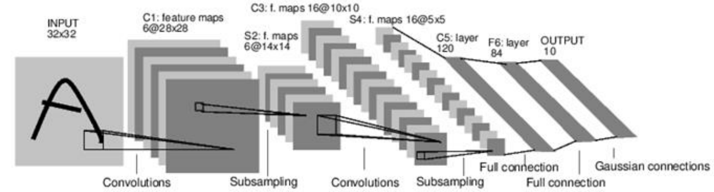
Injung Kim
Handong Global University

Agenda

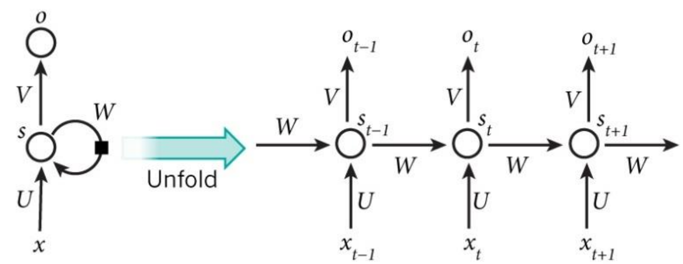
- Introduction
- Encoder–Attention–Decoder Models
- Memory Networks
- Transformer
- Visual Attention Models
- Q&A

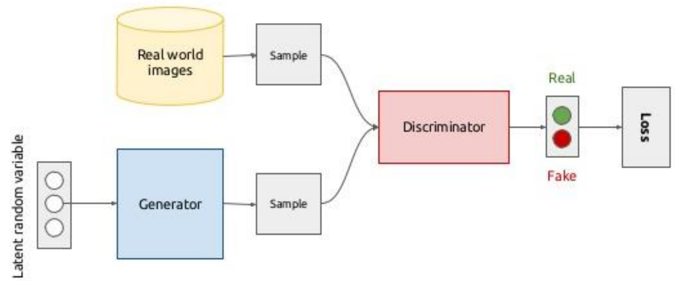
Deep Learning Architectures

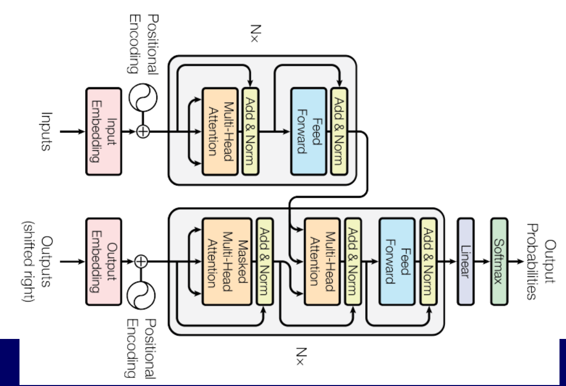
- Deep Neural Networks (DNN)
  - MLP, RBM, DBN, DBM, …
- Convolutional neural networks (CNN)
  - Image processing
  - Learns position invariant local features
- Recurrent neural networks (RNN)
  - Sequence processing (text, speech, etc.)
  - Recurrent connection (short-term memory)
- Deep generative models
  - GAN, VAE, autoreg. models, flow models, etc.
- Attention models
  - Encoder-attention-decoder, transformer, visual attention models
  - NLP, ASR/TTS, image processing
  - Closely related to memory network



A Full Convolutional Neural Network (LeNet)

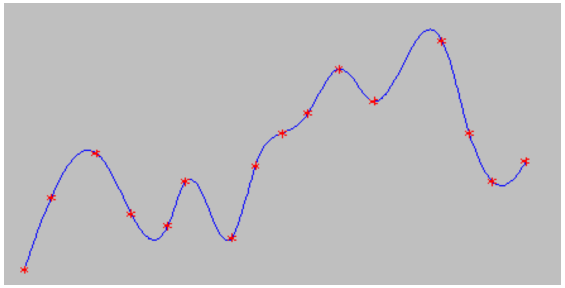


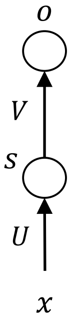




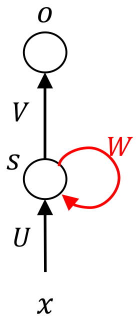
Recurrent Neural Networks

- Recurrent neural network is a neural network specialized for processing a sequence of values $x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}$.
  - Neural networks with recurrent connection
  - State of nodes affect the output and the next state
  - Model for dynamic process
  - Temporarily shared connections

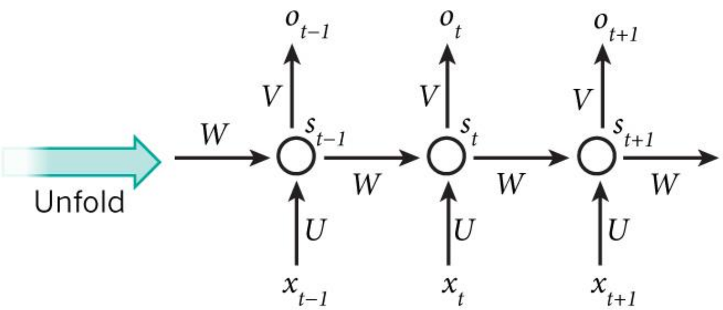




Feed-forward network

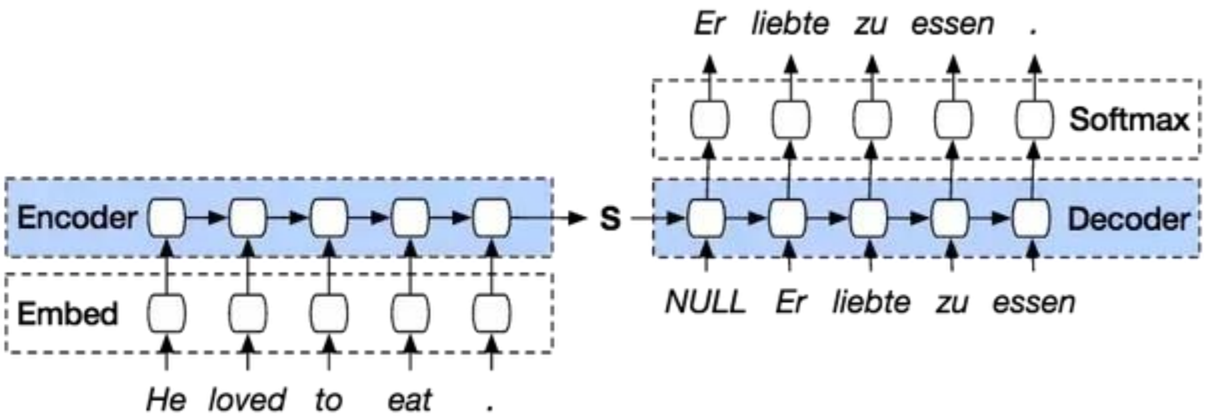


Recurrent network



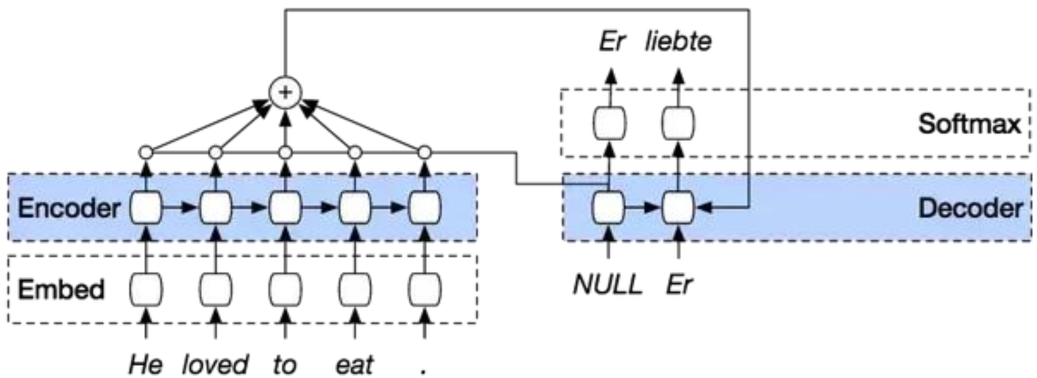
Encoder–Decoder Models

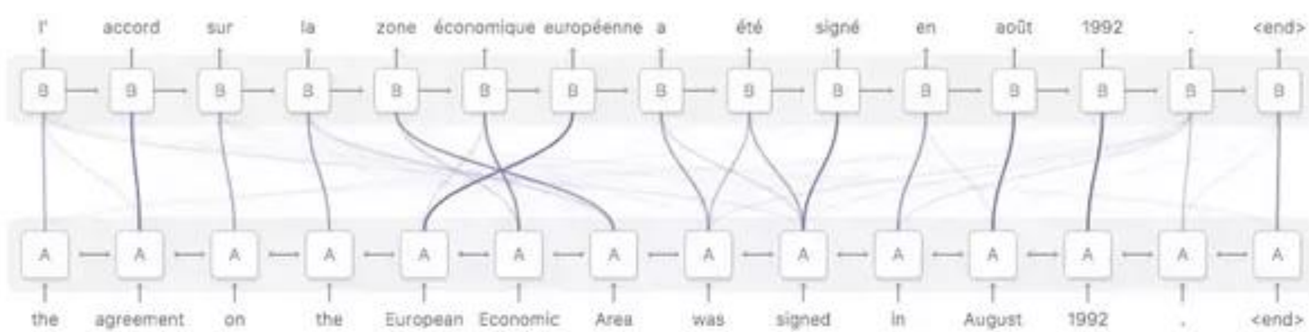
- Architecture for sequence-to-sequence mapping
  - Encoder: input sequence ➔ context vector
  - Decoder: context vector (+ previous output) ➔ new output
- The two modules are trained jointly to maximize the average of $\log P(y^1, y^2, \ldots, y^m | x^1, x^2, \ldots, x^n)$



Encoder–Attention–Decoder Models

- Decoder accesses context composed of weighted average of input states





Attention Models [Bahdanau16]

- Output

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

- Hidden state

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

- Context vector

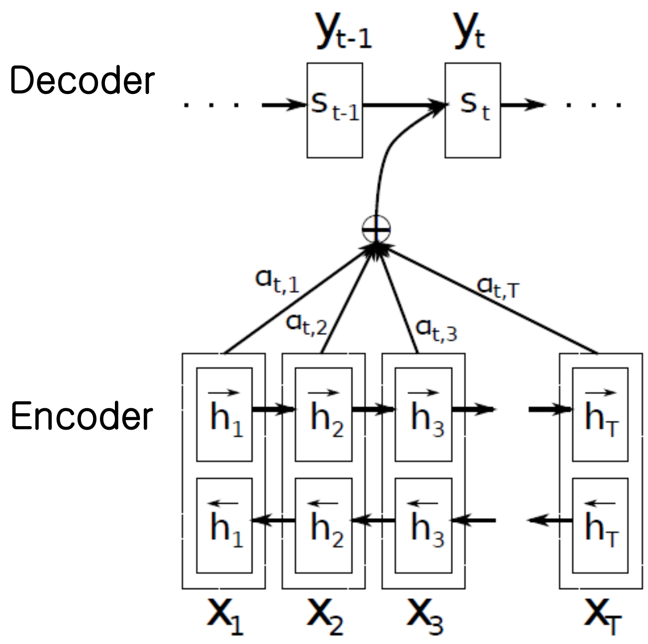
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

value

- Attention

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad e_{ij} = a(s_{i-1}, h_j)$$

query key



Agenda

- Introduction
- Encoder-Attention-Decoder Models
- <u>**Memory Networks**</u>
- Transformer
- Visual Attention Models
- Q&A

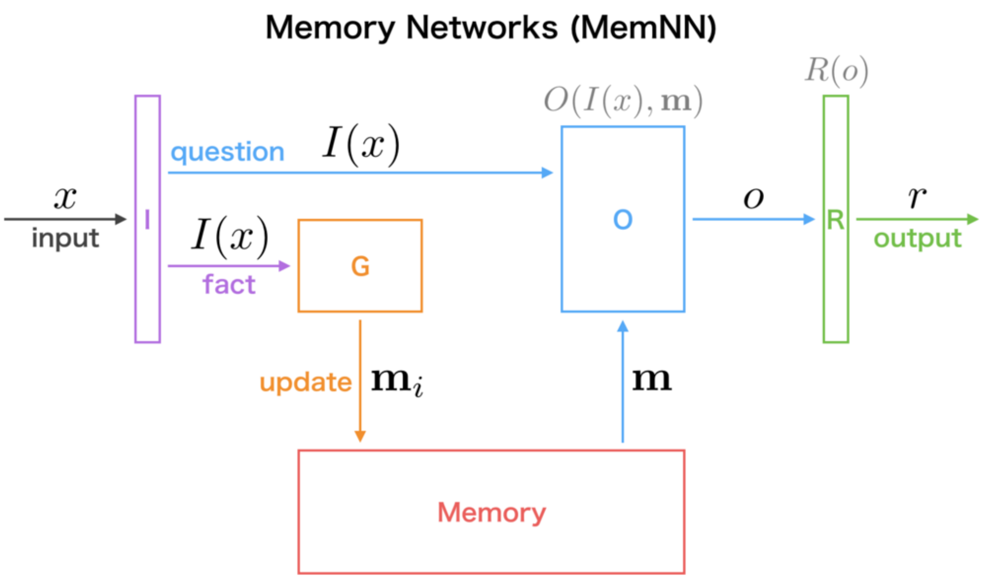
Memory Networks

- Motivation
  - Conventional encoder-decoder models only maps the input sequence to an output sequence
    - Question → Answer
  - Some tasks require explicit memory
    - Read a long story. Then, answer questions about that.
  - Conventional machine learning models lack an easy way to read and write to part of a (potentially very large) long-term memory component.

- MemNN
  - Neural network with explicit memory

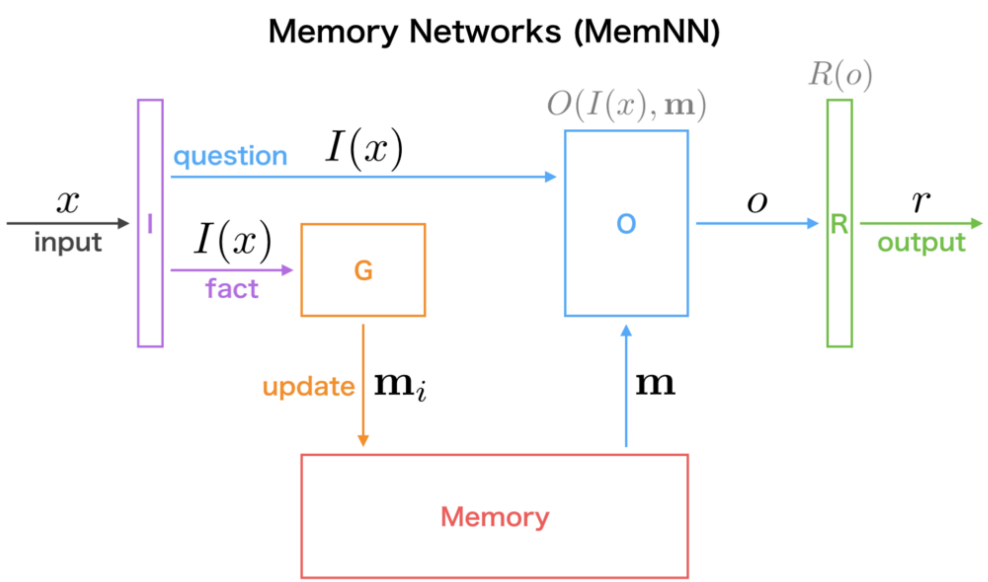
Memory Networks

- Weston, et al., "Memory Networks," Oct. 2014
  - Combine the successful learning strategies for inference with a memory component that can be read and written to.
  - Memory m: an array of objects indexed by m_i
  - Four components (I, G, O and R)



Memory Networks

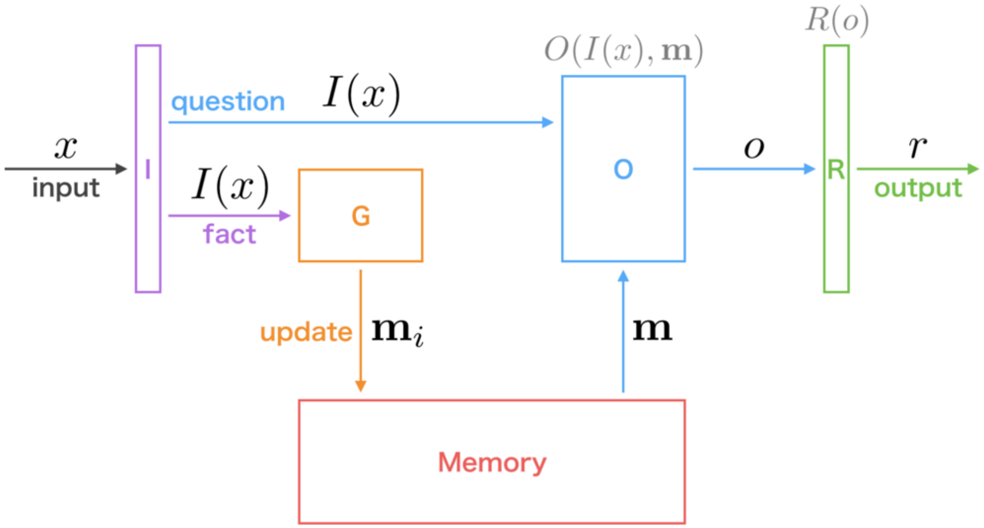
- Four (potentially learned) components
  - **I (input feature map)**: input ➔ feature
  - **G (generalization)**: updates old memories given input
  - **O (output feature map)**: (input + current memory state) ➔ (multiple) memory output
  - **R (response)**: memory output ➔ response



Memory Networks

- Operation given an input x,
  - The same for training/test time. But, model parameters are updated only in training time.
    1. Convert x to an internal feature representation $I(x)$.
    2. Update memories $\mathbf{m}_i$ given the new input: $\mathbf{m}_i = G(\mathbf{m}_i, I(x), \mathbf{m}), \ \forall i$.
    3. Compute output features o given the new input and the memory: $o = O(I(x), \mathbf{m})$.
    4. Finally, decode output features o to give the final response: $r = R(o)$.

Memory Networks (MemNN)



Memory Networks

- Example

o_1

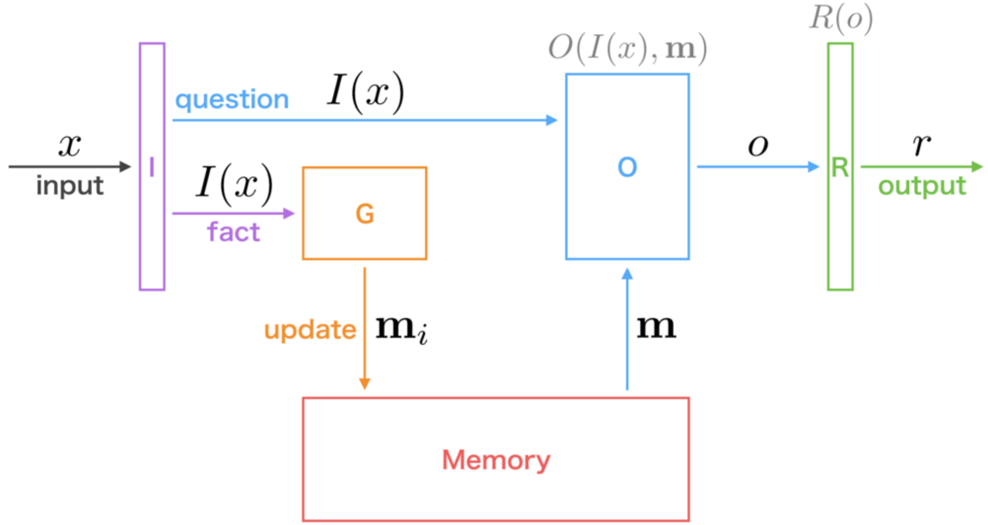
o_2

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.

x (question)

Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

Memory Networks (MemNN)



Hard Addressing vs. Soft Addressing

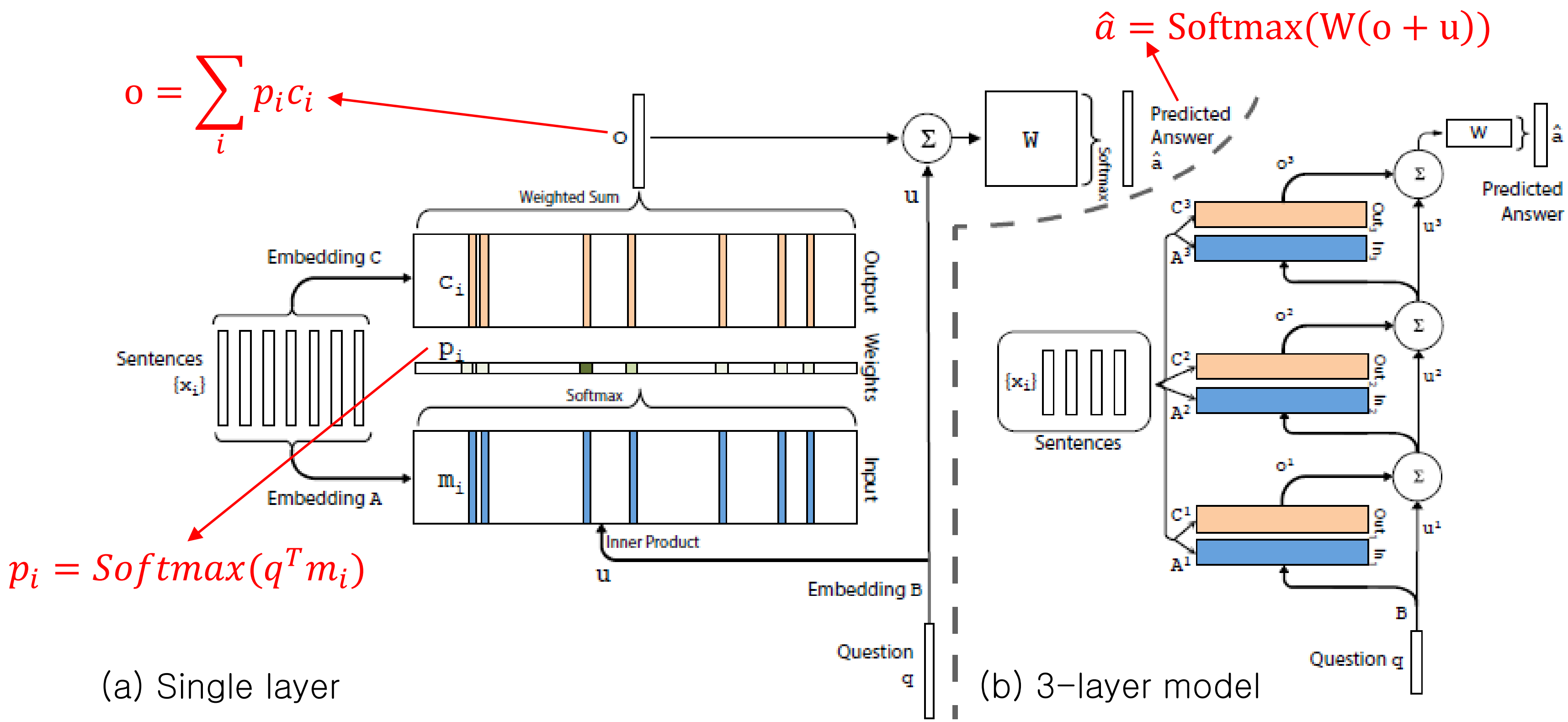
- Memory
  - $M = (M_1, M_2, \dots, M_m)$
    - M_i is memory content (n–dim vector)

- Hard addressing (a: address)
  - $r = M_a$
  - $r = \sum_i w_i M_i$
    - w is a one hot vector. ($w_i = 1$ if $i = a$, and $w_i = 0$, otherwise).

- Soft addressing (w: soft address vector)
  - $r = \sum_i w_i M_i$
    - w is an arbitrary vector (usually, $\sum_i w_i = 1$)

End-to-End Memory Networks [Sukhbaatar15]

- Inputs: a set of inputs $(x_1, x_2, \ldots, x_n)$ + query (q)
- Output: answer (a)
  - Produced by multiple *hops*.

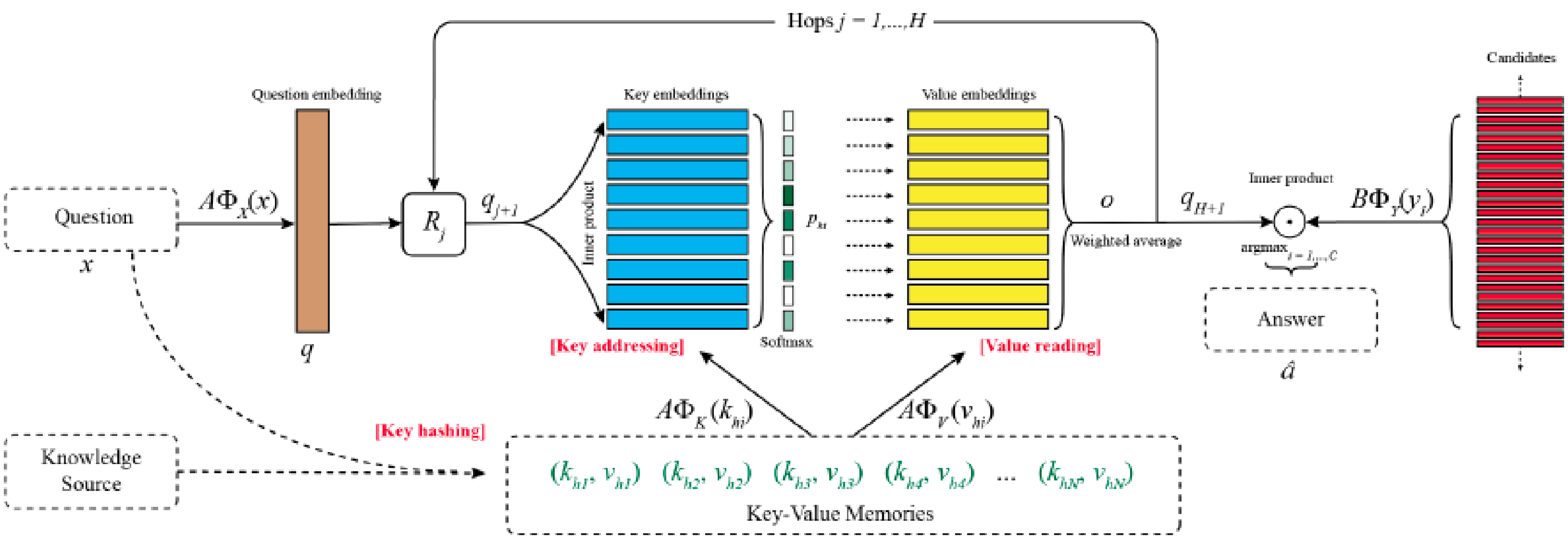


(a) Single layer

(b) 3-layer model

Key-Value Memory Network

- Miller, et al., "Key-Value Memory Networks for Directly Reading Documents", June, 2016.
  - KV-MemNN for question answering
  - Stores facts in key-value structured memory
    - Key: address relevant memories w.r.t. question
    - Value: the information the memory provides



Content-based vs. Location-based Addressing

- Key-value memory
  - $M = (M_1, M_2, \ldots, M_m), M_i = (K_i, V_i)$
- Accessing memory M with a weight vector w
  - $r = \sum_i w_i V_i = wV$
- Content-based addressing
  - Determine w by similarity between query (Q) and key (K_i)
    - $w_i^t = f(Q, K_i)$ (e.g., $Softmax(Q \cdot K_i)$)
    - $r = \sum_i w_i V_i$
- Location-based addressing
  - Choose new address near the previous address
    - $w_i^t = w_i^{t-1} + \Delta w_i^t$
    - $\Delta w_i^t = g(Q, K_i, w_i^{t-1})$

Attention Models [Bahdanau16]

- Output

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

- Hidden state

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

- Context vector

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

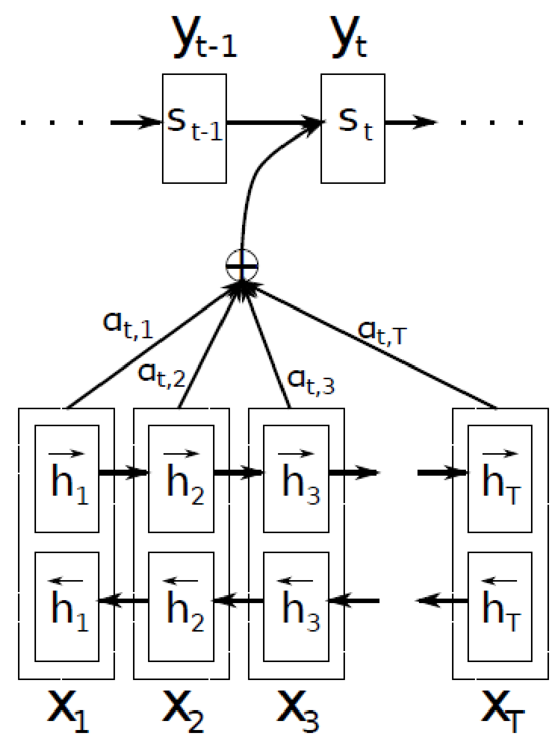
value

- Attention

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad e_{ij} = a(s_{i-1}, h_j)$$

query

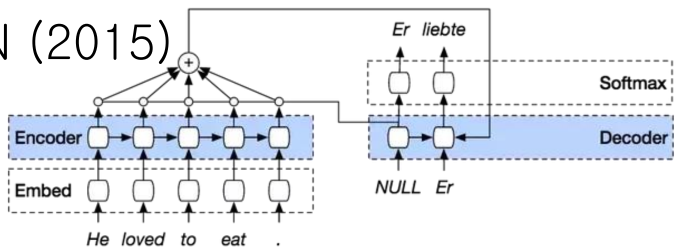
key



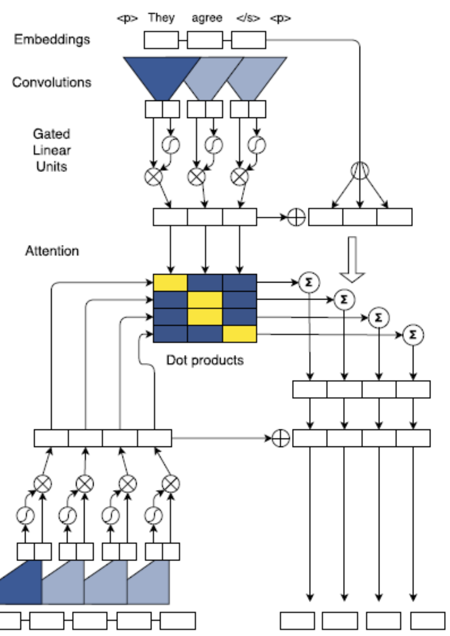
Natural Language Processing

- Encoder-Attention-Decoder models
  - RNN ➔ CNN ➔ Transformer ➔ BERT, GPT1/2/3

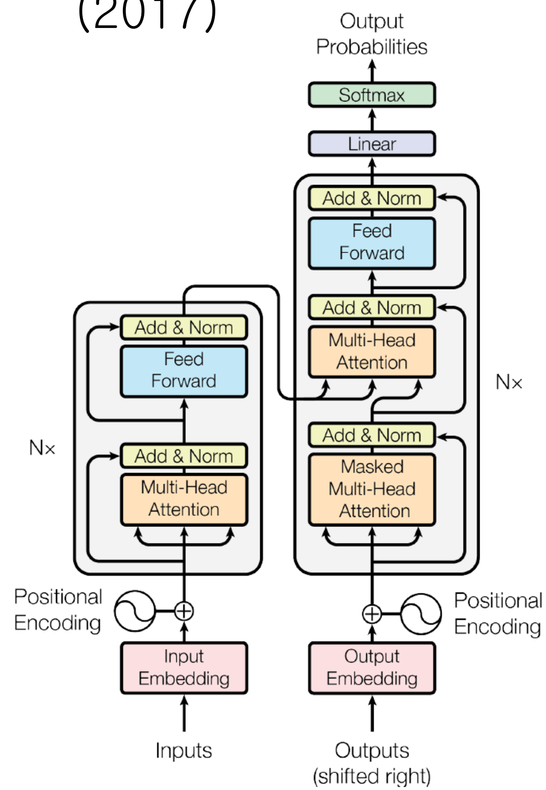
RNN (2015)



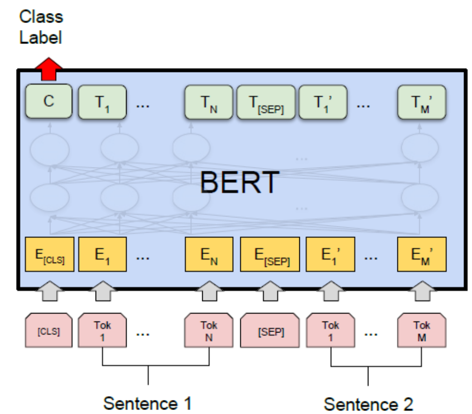
CNN (2017)



Transformer Net (2017)



BERT (2018)



Agenda

- Introduction
- Encoder-Attention-Decoder Models
- Memory Networks
- <u>Transformer</u>
- Visual Attention Models
- Q&A

Transformer

- A. Vaswan, "Attention Is All You Need" Jun. 2017
  - Attention mechanism instead of convolutional or recurrent units
  - Global dependencies between input and output
  - More parallelization
  - Reduced effective resolution
  - ➔ Counteract with multi-head attention

- References
  - Illustrated-transformer (http://jalammar.github.io/illustrated-transformer/)
  - The annotated transformer (http://nlp.seas.harvard.edu/2018/04/03/attention.html)

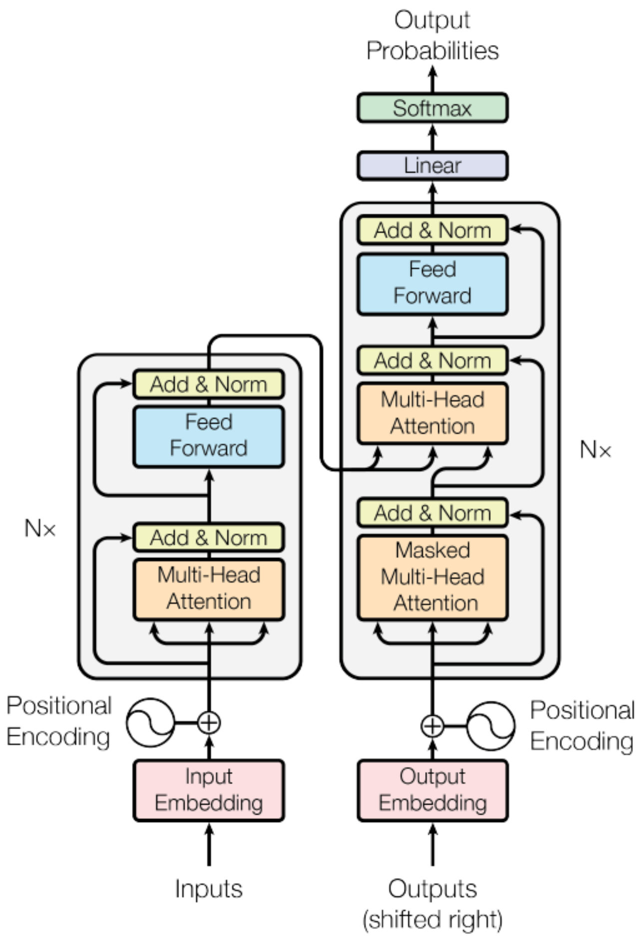
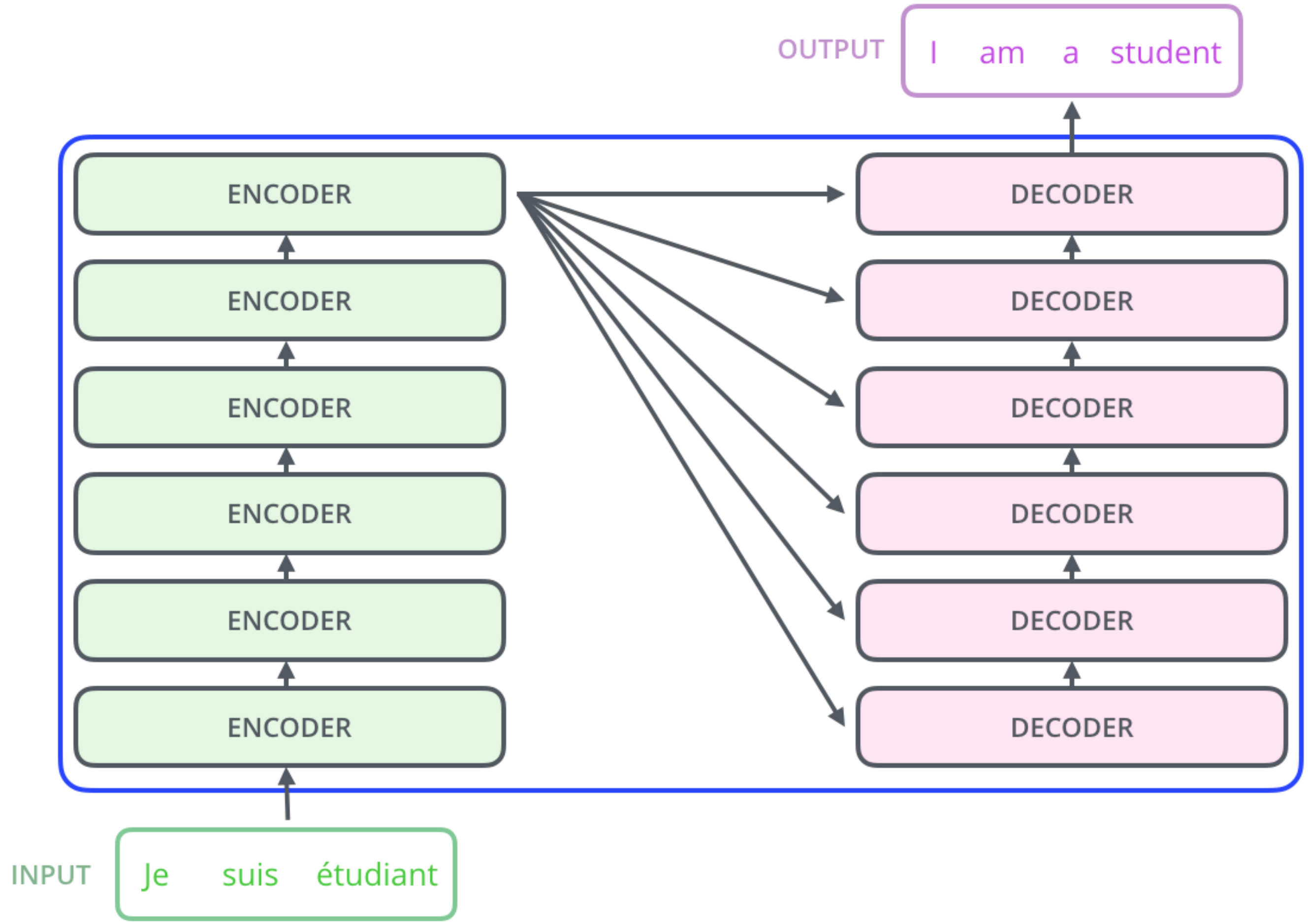


Figure 1: The Transformer - model architecture.

Transformer

- Illustration of Transformer operation



src: https://ai.googleblog.com

Transformer Encoder

- Encoder consists of N=6 identical layers
  - 1st sublayer: multi-head self-attention
  - 2nd sublayer: position-wise FC
  - Residual connection + layer norm

$$LayerNorm(x + sublayer(x))$$

  - All sublayers $d_{model} = 512$

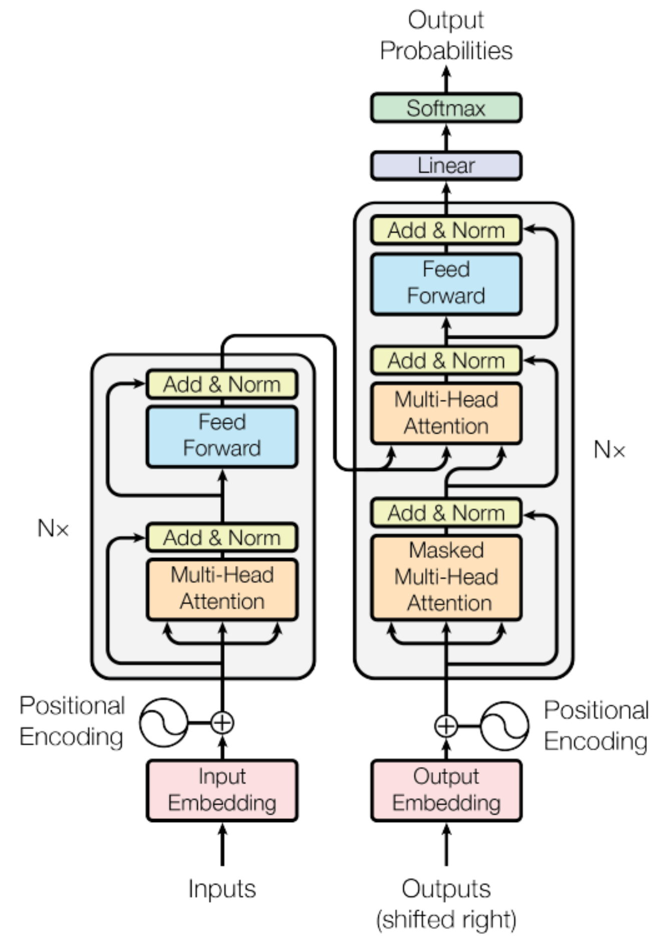


Figure 1: The Transformer - model architecture.

Transformer Decoder

- Decoder consists of N=6 identical layers
  - Similar to encoder layers
  - Additional sublayer: performs multi-head attention over the output of the encoder stack.
  - Masked self-attention only to preceding positions

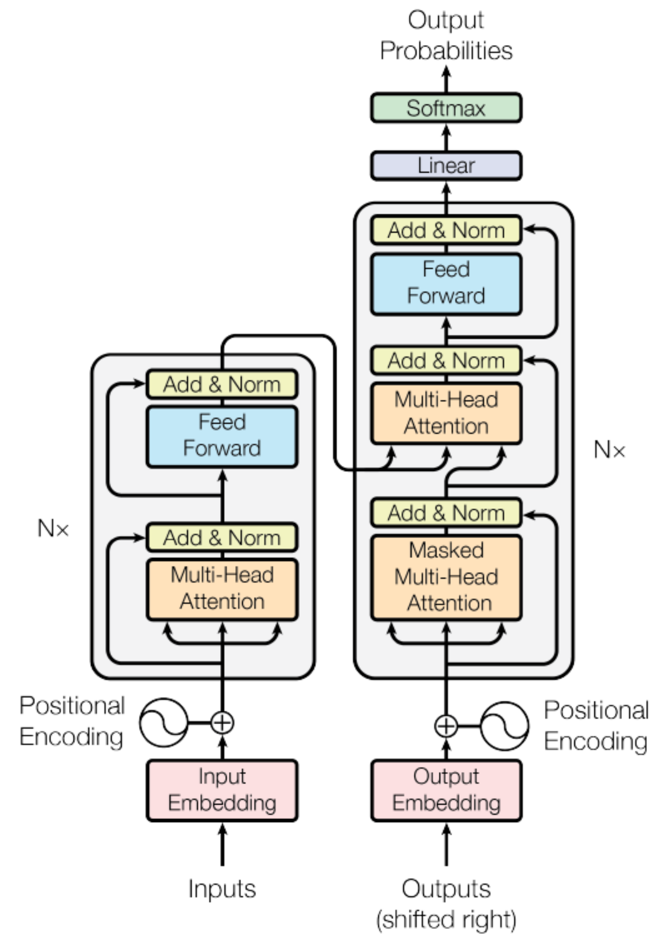


Figure 1: The Transformer - model architecture.

Attention

- Scaled dot-product attention
  - Input: a query, a set of key-value pairs
  - Output: weighted sum of values

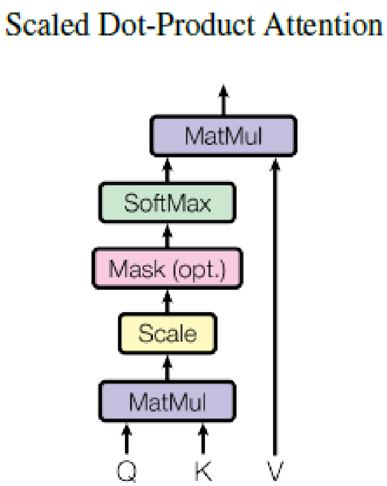
$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$$

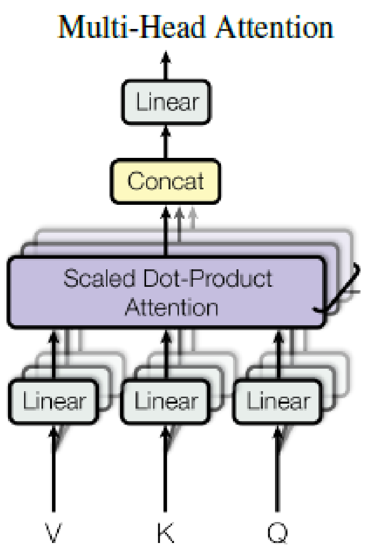
- Multi-header attention
  - Jointly attend to information from different representation subspaces at different positions

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

  - $h = 8, d_k = d_v = \frac{d_{model}}{h} = 64$



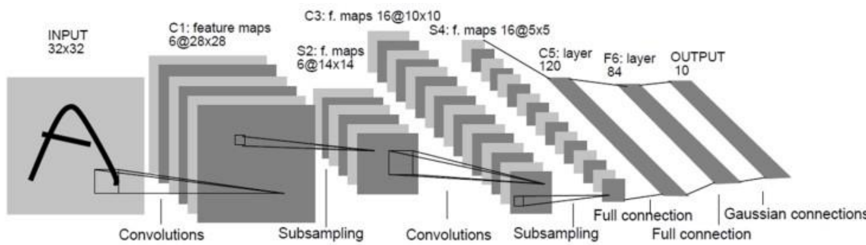


Agenda

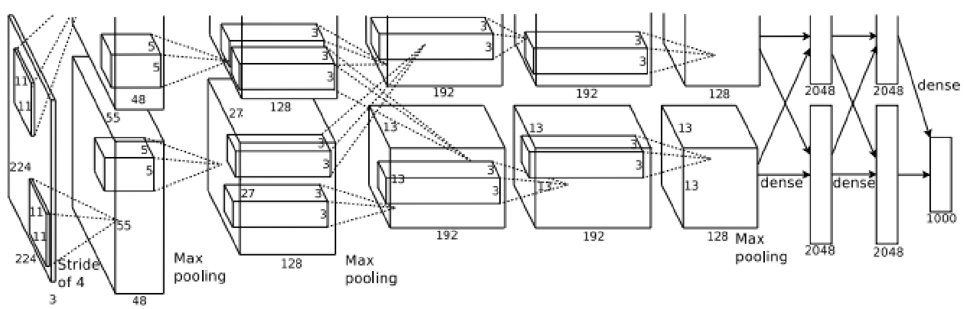
- Introduction
- Encoder-Attention-Decoder Models
- Memory Networks
- Transformer
- <u>Visual Attention Models</u>
- Q&A

Convolutional Neural Networks

- Convolutional Neural networks (CNN): a class of deep feed-forward network designed to mimic human/animal visual systems
  - LeNet 1998, MNIST, CPU



  - AlexNet 2012, ImageNet, GTX 580 x 2



ResNet

- He, et.al, "Deep Residual Learning for Image Recognition", Dec. 2015
  - Target function $H(x) = F(x) + x$
  - Residual Function $F(x) = H(x) - x$
  - Deep residual network contains 152 layers

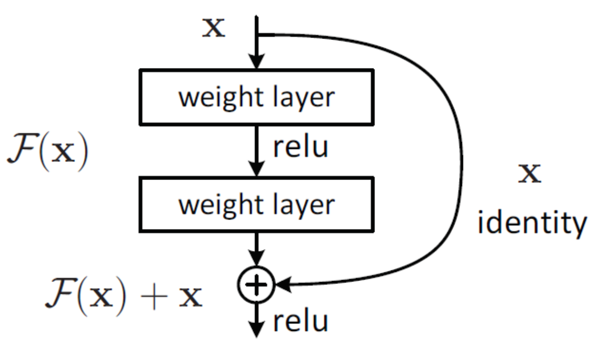
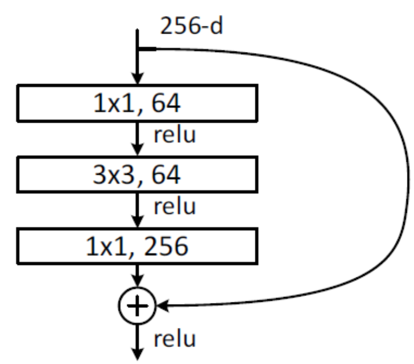


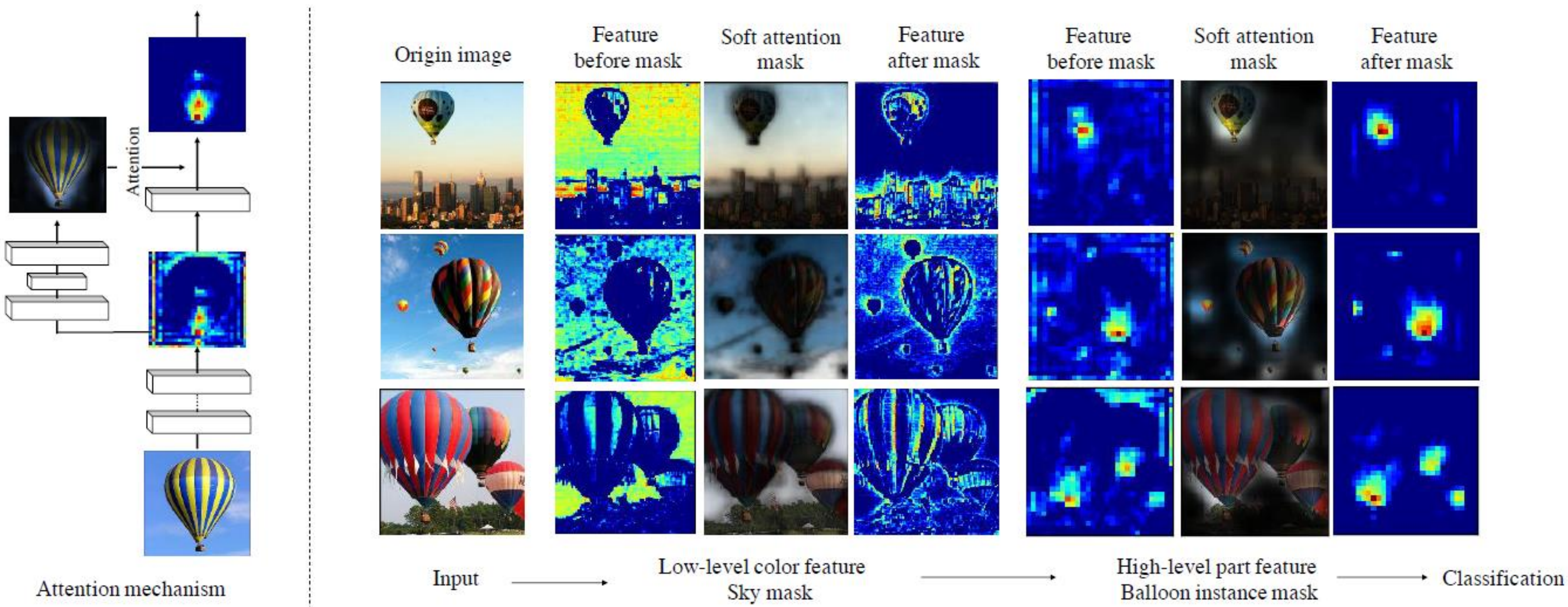
Figure 2. Residual learning: a building block.



Bottleneck building block

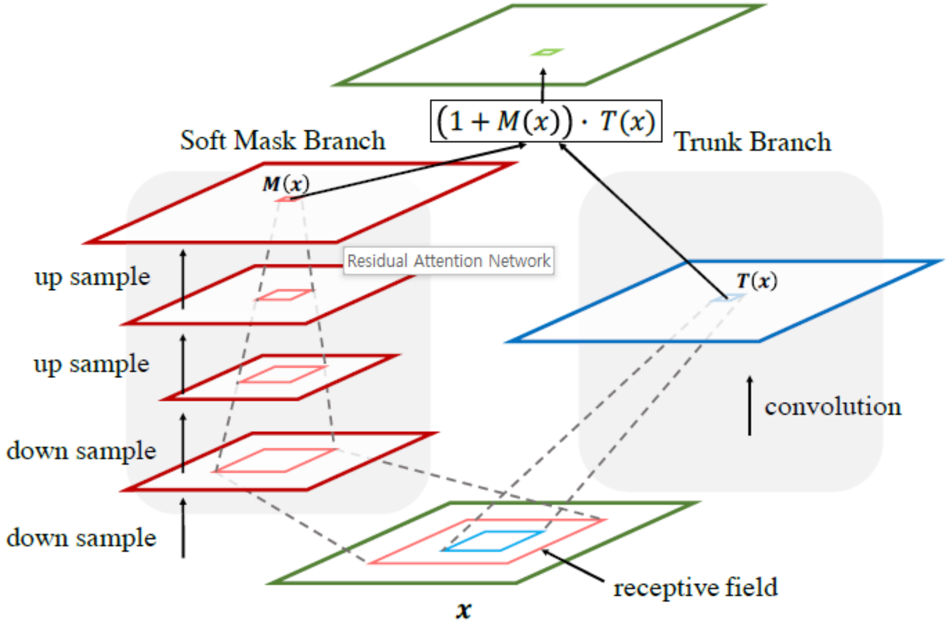
Residual Attention Network

- Wang, et al., "Residual Attention Network for Image Classification," Apr. 2017.
  - Residual Attention Network built by stacking Attention Modules
  - Attention residual learning
  - Bottom-up top-down feedforward network (hourglass model)



Residual Attention Network

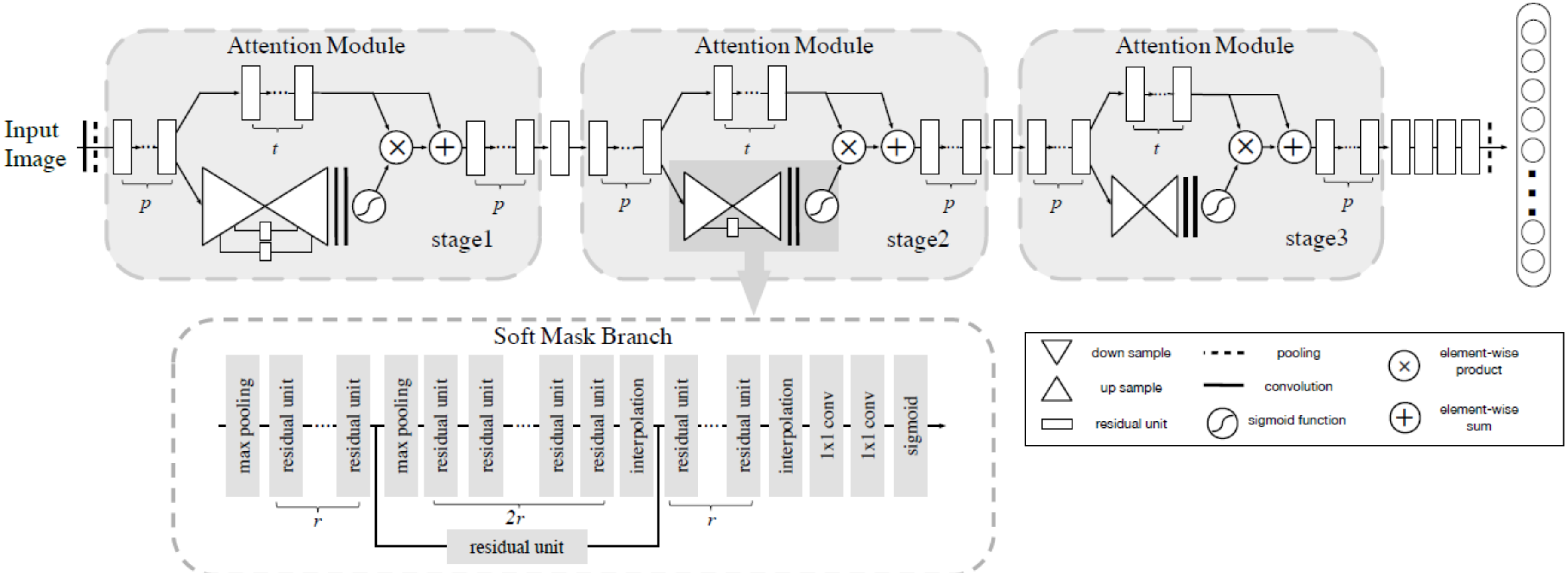
- Wang, et al., "Residual Attention Network for Image Classification," Apr. 2017.
  - Residual Attention Network built by stacking Attention Modules
  - Attention residual learning
  - Bottom-up top-down feedforward network (hourglass model)



- Resnet: $H(x) = x + F(x)$
- RAN: $T(x) + M(x)T(x)$
  $= (1 + M(x))T(x)$

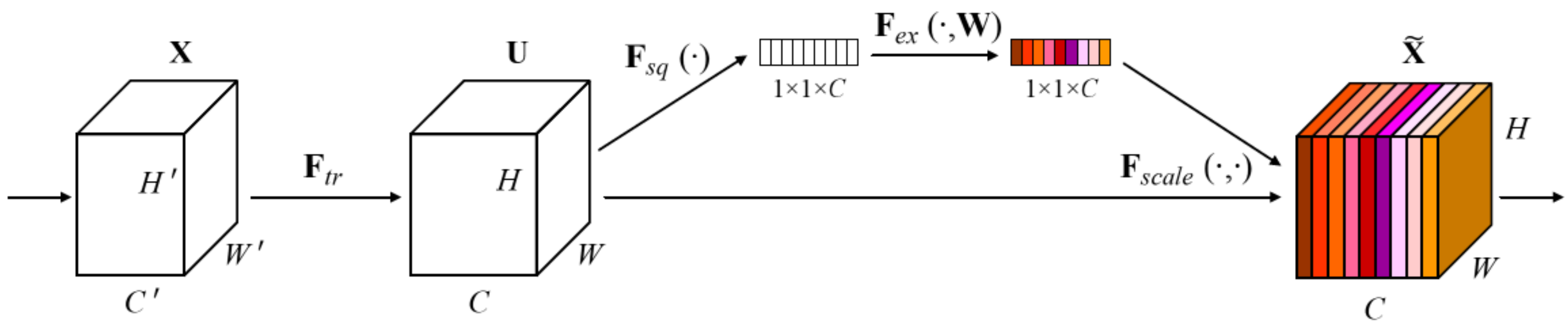
Residual Attention Network

- Wang, et al., "Residual Attention Network for Image Classification," Apr. 2017.
  - Residual Attention Network built by stacking Attention Modules
  - Attention residual learning
  - Bottom-up top-down feedforward network (hourglass model)



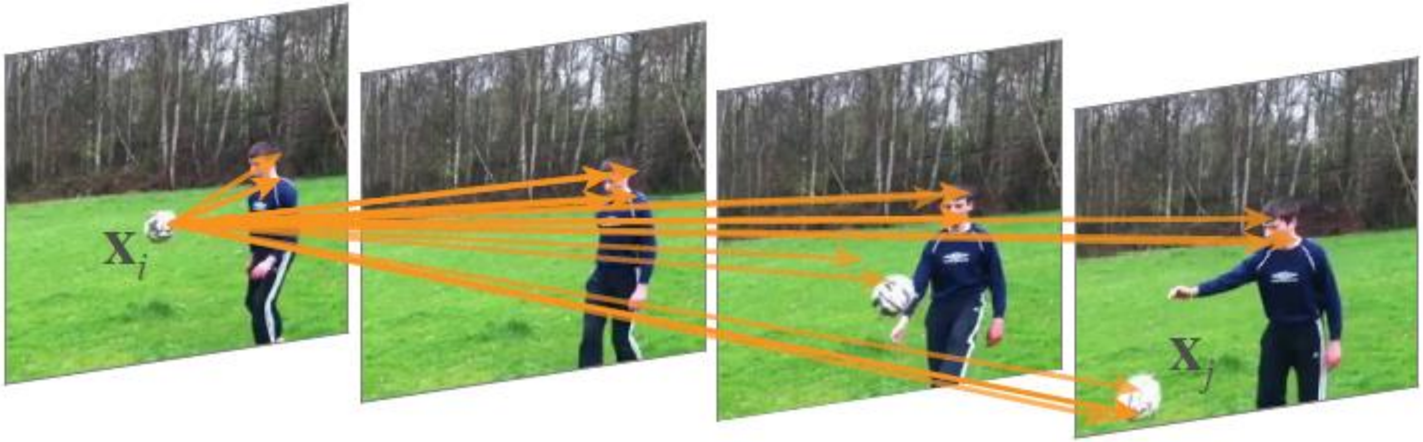
SENet

- Hu, et al., "Squeeze-and-Excitation Networks," Sep. 2017.
  - Recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels
  - "Attention along channels"



Non-Local Nets

- Wang, et al., "Non-local Neural Networks," Nov. 2017.
  - Learns relation among distant feature elements

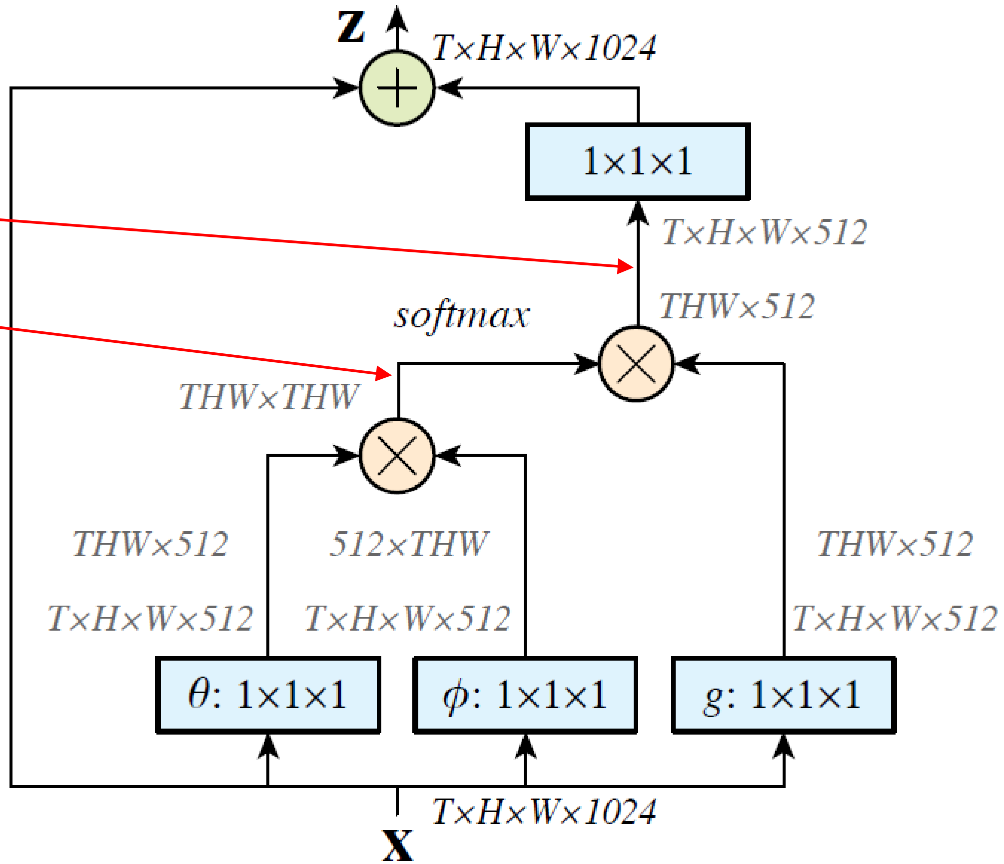


Non-Local Nets

- Wang, et al., "Non-local Neural Networks," Nov. 2017.

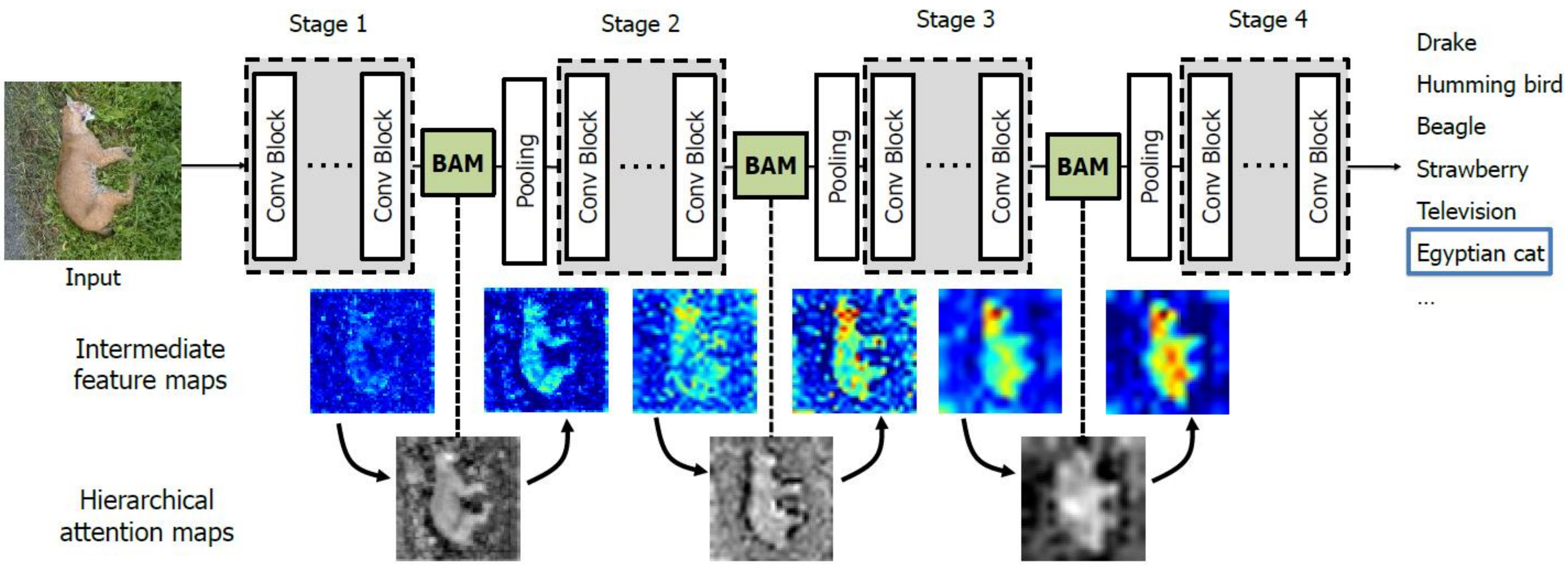
$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}.$$

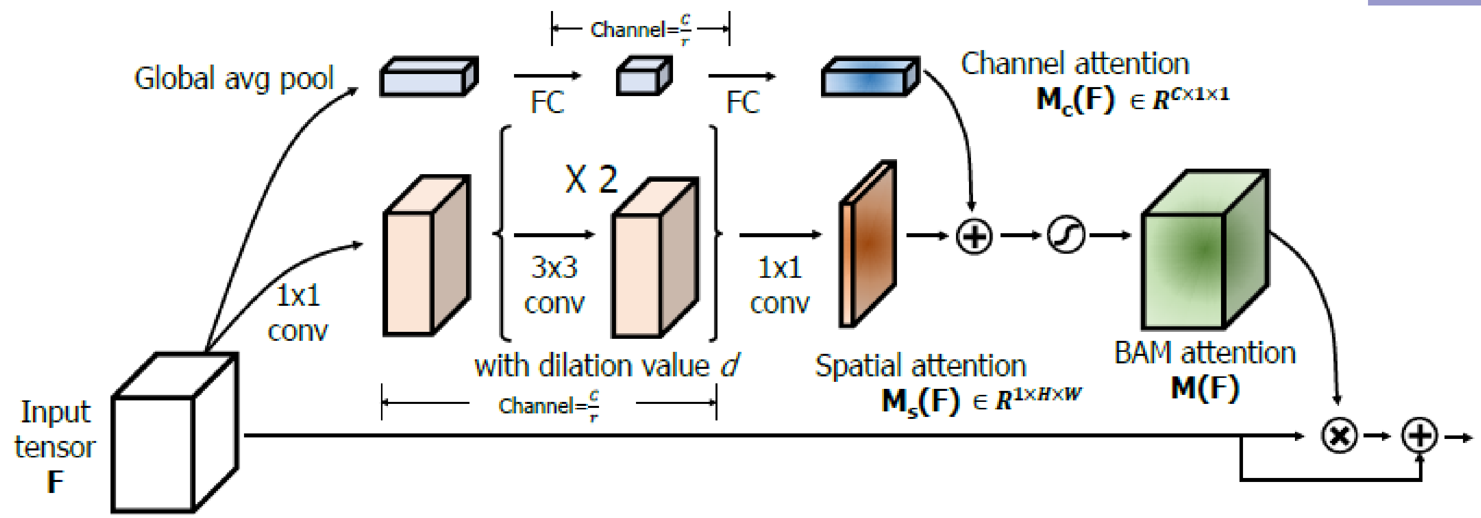


BAM

- Park, et al., “BAM: Bottleneck Attention Module,” Jul. 2018.
  - Infers an attention map along two channel and spatial
  - Place BAM at each bottleneck of models where the downsampling of feature maps occurs.



BAM



- Output feature map $F' = F + F \oplus M(F)$
- 3D attention map $M(F) = \sigma(M_c(F) + M_s(F))$
- Channel attention

$$\mathbf{M_c}(\mathbf{F}) = BN(MLP(AvgPool(\mathbf{F}))) = BN(\mathbf{W_1}(\mathbf{W_0}AvgPool(\mathbf{F}) + \mathbf{b_0}) + \mathbf{b_1})$$

- Spatial attention

$$\mathbf{M_s}(\mathbf{F}) = BN(f_3^{1\times1}(f_2^{3\times3}(f_1^{3\times3}(f_0^{1\times1}(\mathbf{F})))))$$

Q&A

Thank you
for your attention!