LLM Sys

Transformer

Lei Li

Language Technologies Institute | Carnegie Mellon University School of Computer Science

Recap

- Learning parameters of an NN needs gradient calculation
- Computation Graph
  - to perform computation: topological traversal along the DAG
- Auto Differentiation
  - building backward computation graph for gradient calculation
- Put together: Deep Learning Framework
  1. Define program i.e., symbolic computation graph w/ placeholders/variable/operation nodes
  2. Executes (optimized) computation graph on a set of available devices

Today's Topic

- Encoder-decoder Architecture

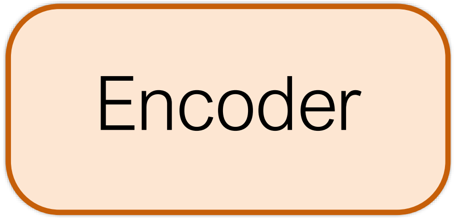
- Transformer Model
  - Embedding
  - Multihead Attention, Decoder Self-Attention
  - FFN
  - Layernorm

- Training Techniques and Performance of Transformer

- Code walkthrough

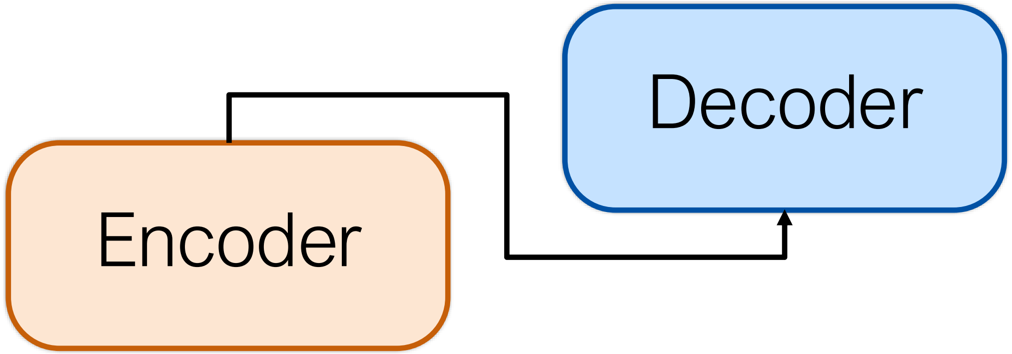
Type of Language Models

Encoder-only
Masked LM
Non-autoregressive



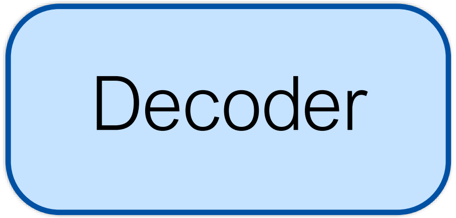
e.g. BERT
RoBERTa
ESM (for protein)

Encoder-decoder



e.g. T5

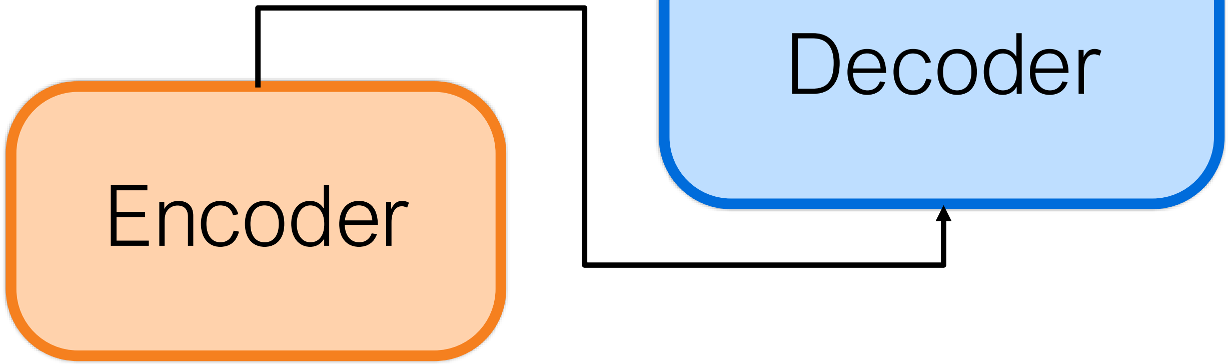
Decoder-only
Autoregressive



e.g. GPT
LLaMA
ProGen (for protein)

Encoder-Decoder Paradigm

target:
I like singing and dancing.



Source: 我喜欢唱歌和跳舞。

$$p_\theta(y|x) = \prod_i p(y_i|x, y_{1:i-1})$$

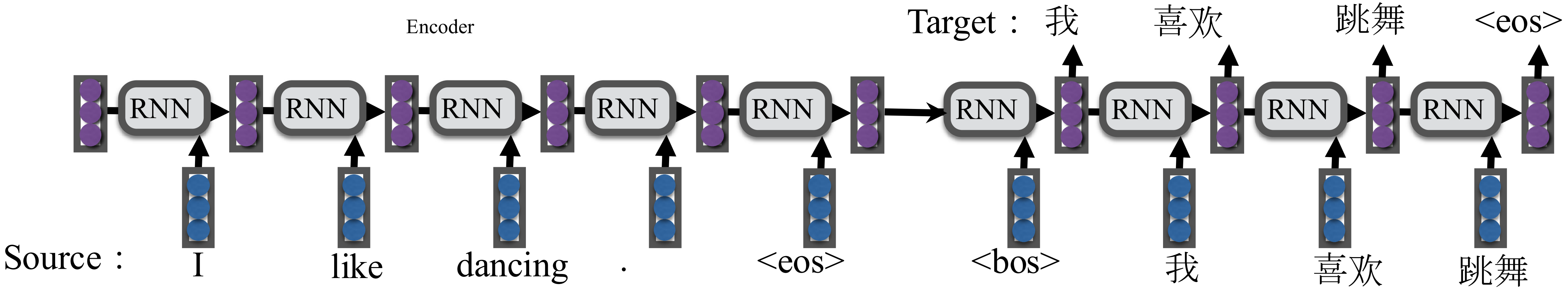
conditional prob. modeled by neural networks (Transformer)

Sequence to Sequence Learning

- Conditional text generation: directly learning a function mapping from source sequence to target sequence

$$p_\theta(y|x) = \prod_t p(y_t|x, y_{1:t-1}; \theta)$$

- Previous encoder/decoder: LSTM or GRU



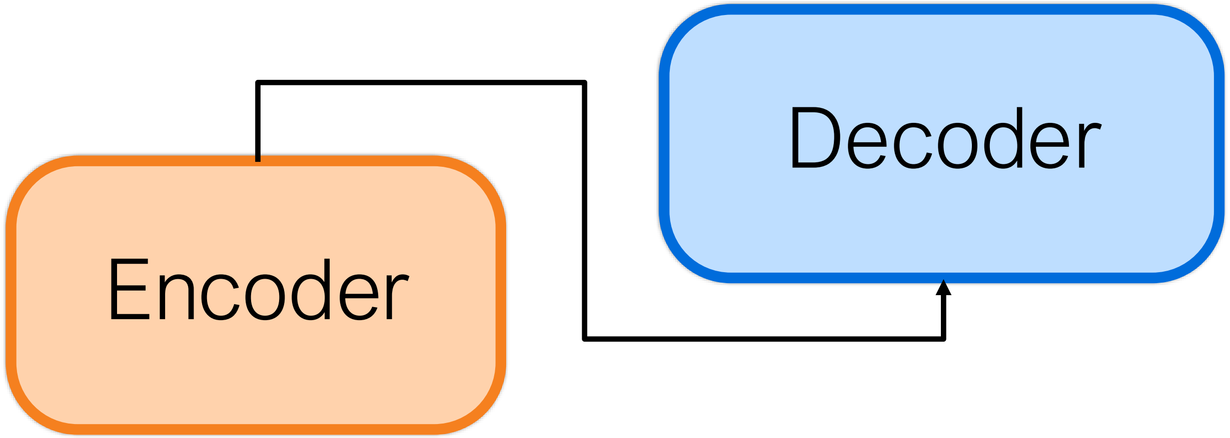
Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014

Motivation for a new Architecture

- Full context and parallel: use Attention in both encoder and decoder

- no recurrent ==> concurrent encoding

target:
I like singing and dancing.

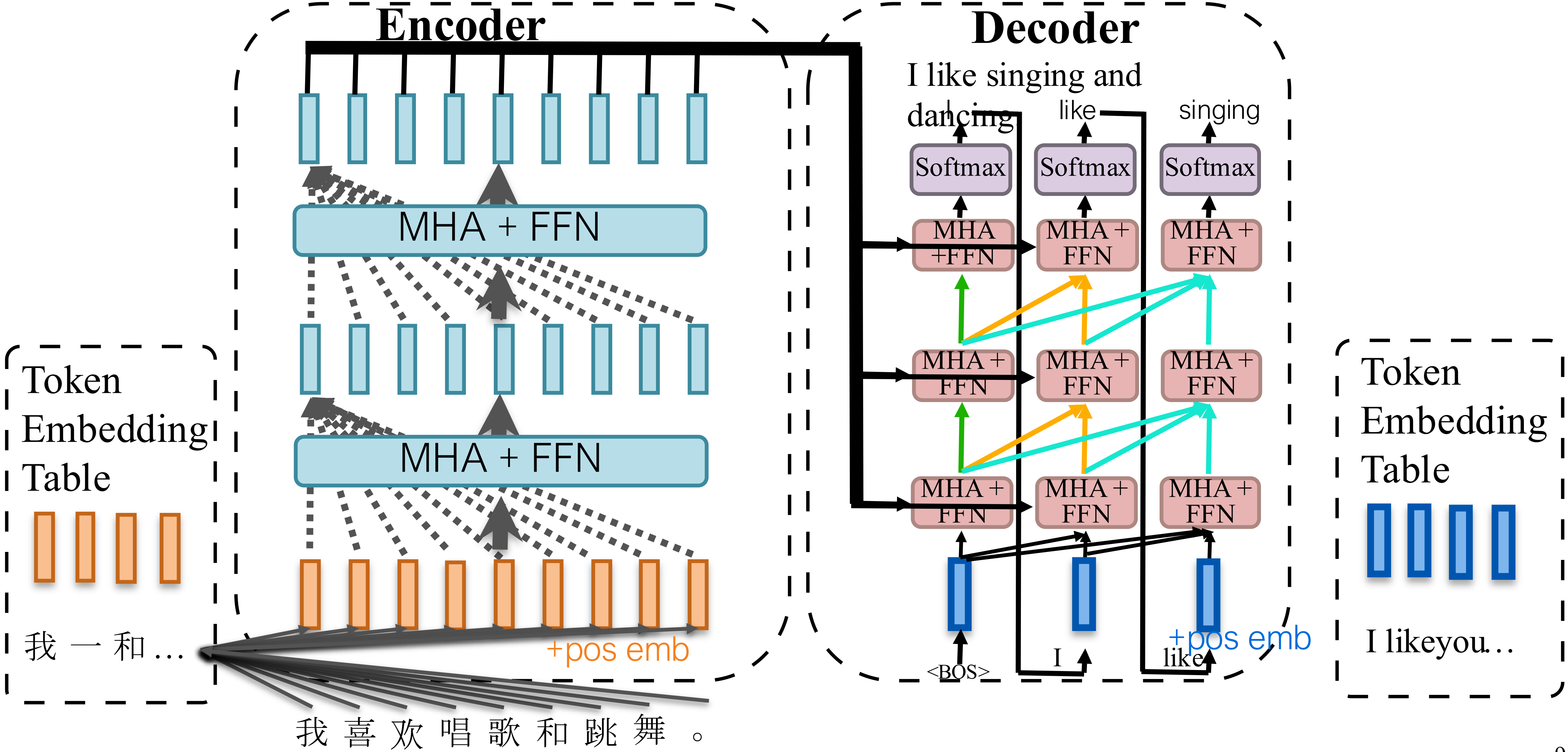


Source: 我喜欢唱歌和跳舞。

Today's Topic

- Encoder-decoder Architecture
- Transformer Model
  - Embedding
  - Multihead Attention, Decoder Self-Attention
  - FFN
  - Layernorm
- Training Techniques and Performance of Transformer
- Code walkthrough

Transformer



Vaswani et al. Attention is All You Need. 2017

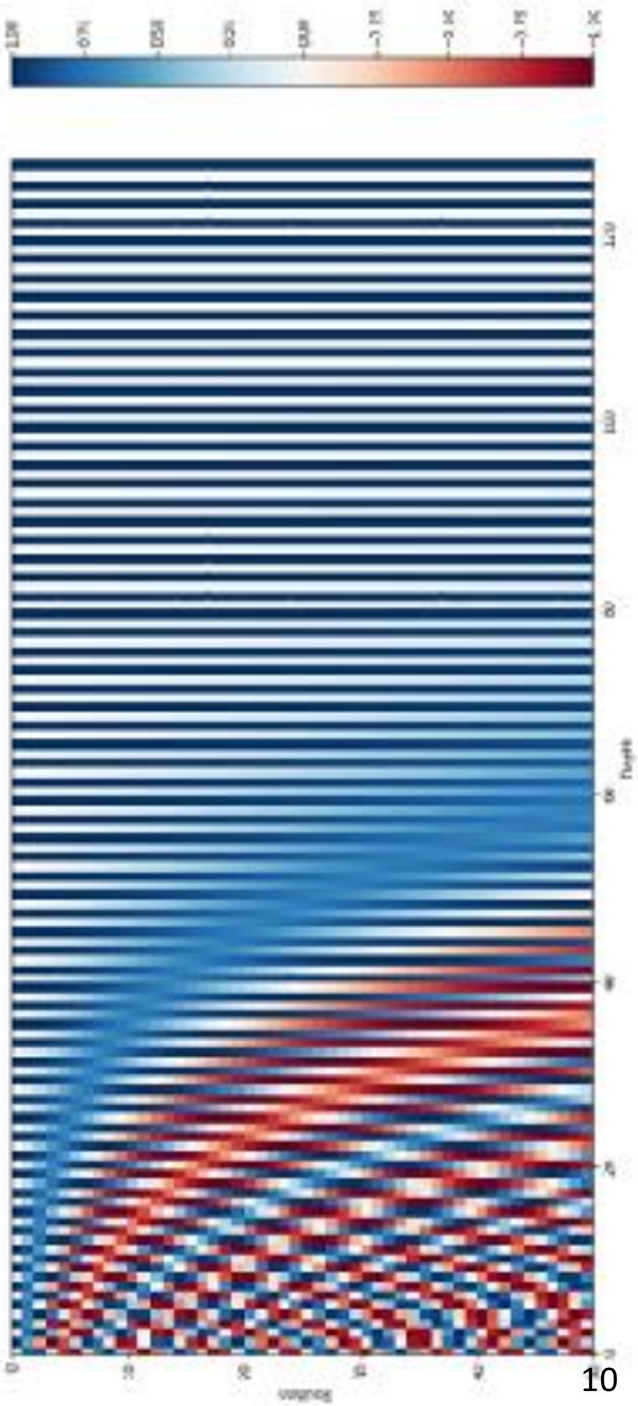
Embedding

- Token Embedding: (tokenization next lec.)
  - Shared (tied) input and output embedding from lookup table

- Positional Embedding:
  - to distinguish words in different position, Map position labels to embedding, dimension is same as Tok Emb, for t-th pos, i-th dim

$$PE_{\mathrm{t},2i} = \sin\left(\frac{t}{1000^{2i/d}}\right)$$

$$PE_{\mathrm{t},2i+1} = \cos\left(\frac{t}{1000^{2i/d}}\right)$$



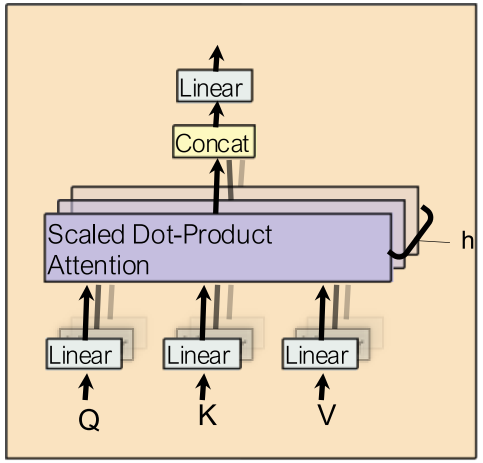
Multi-head Attention

- Instead of one vector for each token
- break into multiple heads
- each head perform attention

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V)$$

$$= \text{Concat}(\text{Head}_1, \text{Head}_2, \dots, \text{Head}_h)W^O$$



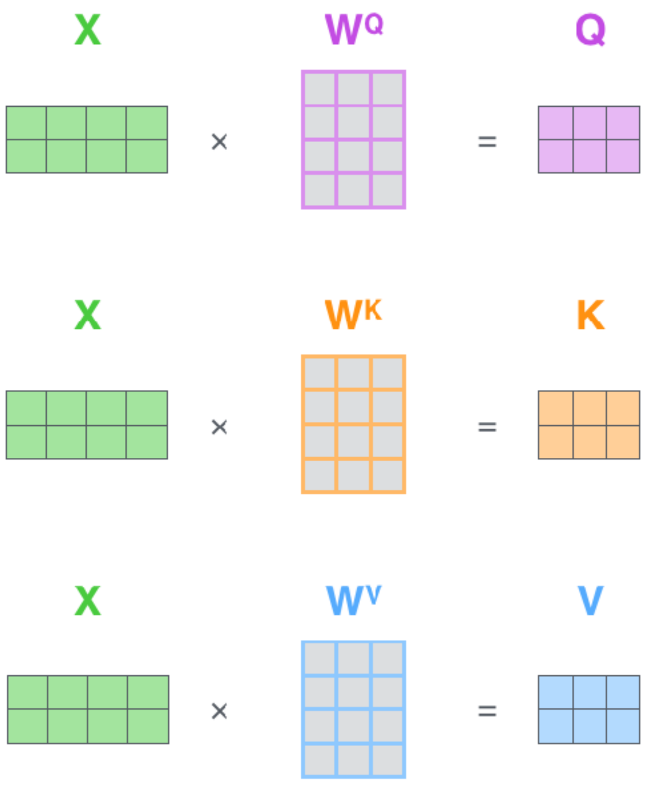
Multi-head Attention

X are input embeddings from previous layer (num of tok * dim)

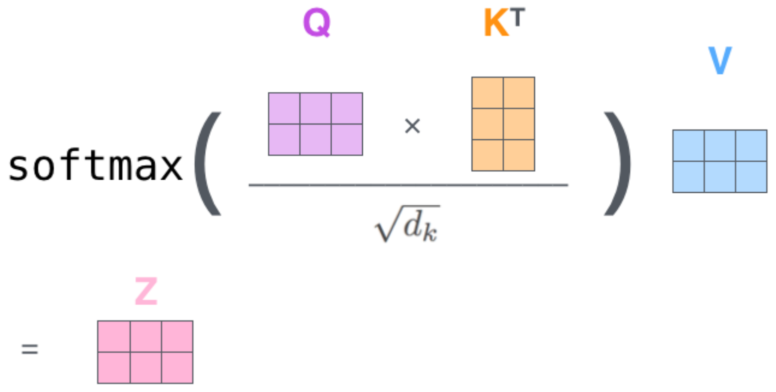
Query

Key

Value



sent len x sent len



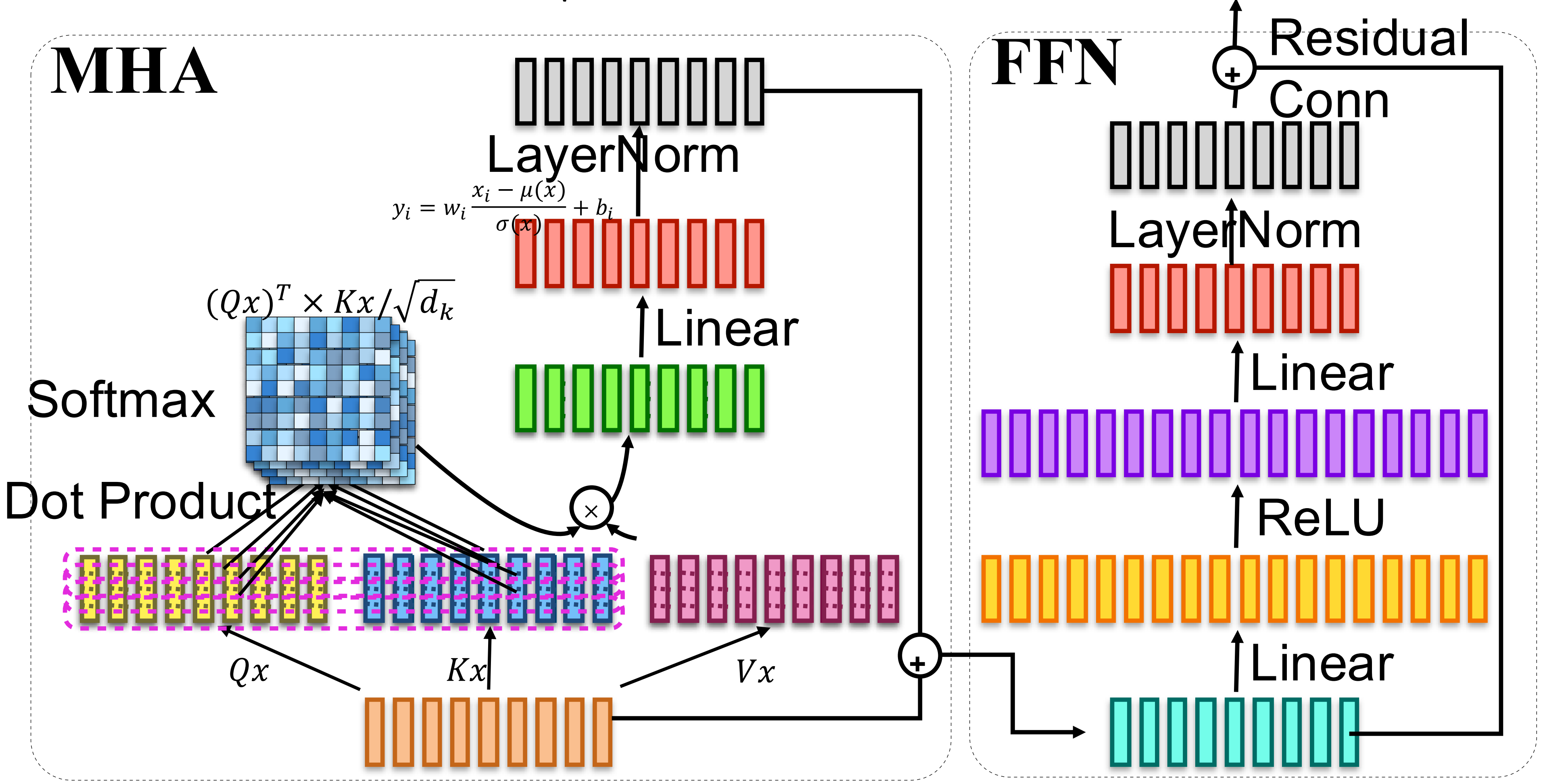
len x dim

Q: why divided by sqrt(d)?

Alammar, The Illustrated Transformer

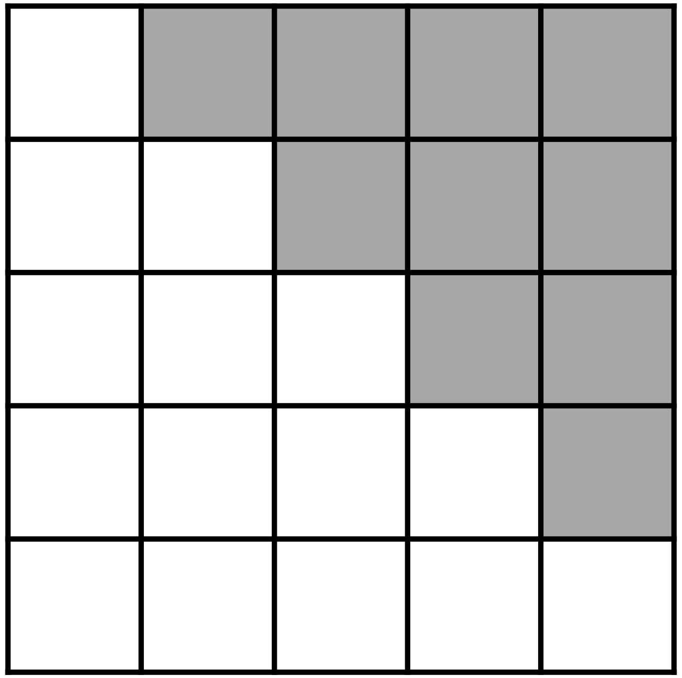
Multihead Attention and FFN

$$\text{Attention}(Q,K,V,x) = \text{Softmax}\left(\frac{(Qx)^T Kx}{\sqrt{d}}\right)\cdot (Vx)^T \qquad \text{FFN}(x) = max(0, x\cdot W_1 + b_1)\cdot W_2 + b_2$$

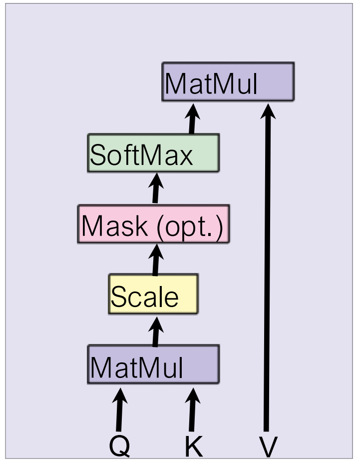


Decoder Self-Attention

- Maskout right side before softmax (-inf)

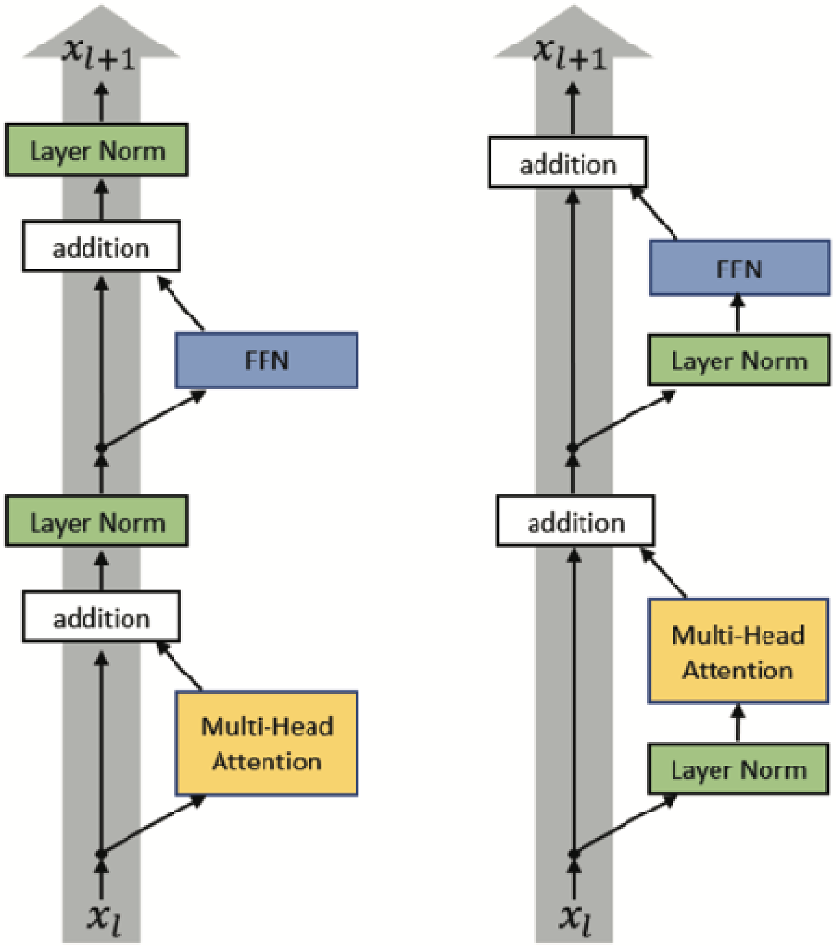


Scaled Dot-Product Attention



Residual Connection and Layer Normalization

- Residual Connection
- Make it zero mean and unit variance within layer
- Post-norm
- Pre-norm

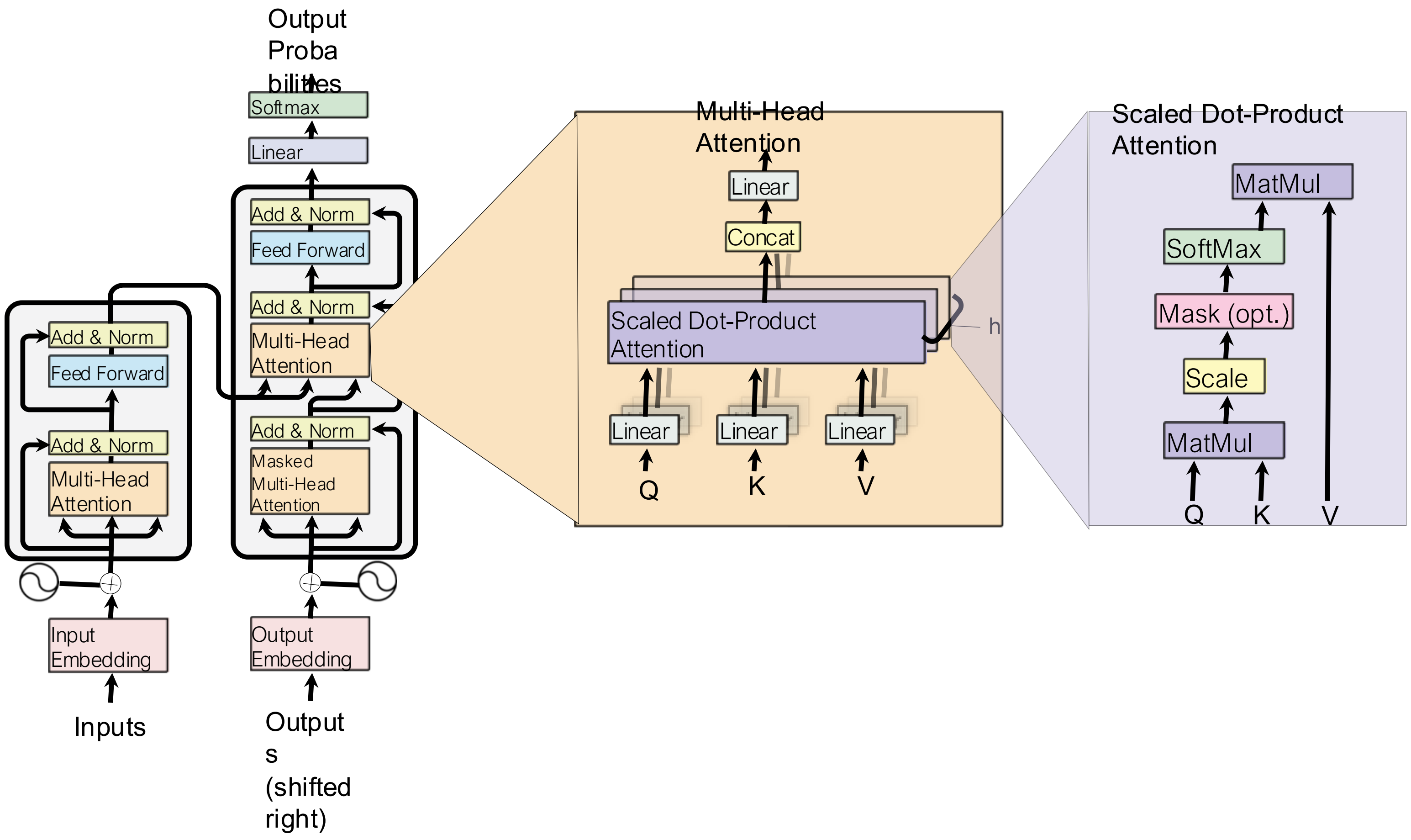


Today's Topic

- Encoder-decoder Architecture
- Transformer Model
  - Embedding
  - Multihead Attention, Decoder Self-Attention
  - FFN
  - Layernorm
- Training Techniques and Performance of Transformer
- Code walkthrough

Transformer in Original Paper

- C layers of encoder (=6)
- D layers of decoder (=6)
- Token Embedding: 512 (base), 1024 (large)
- FFN dim=2048



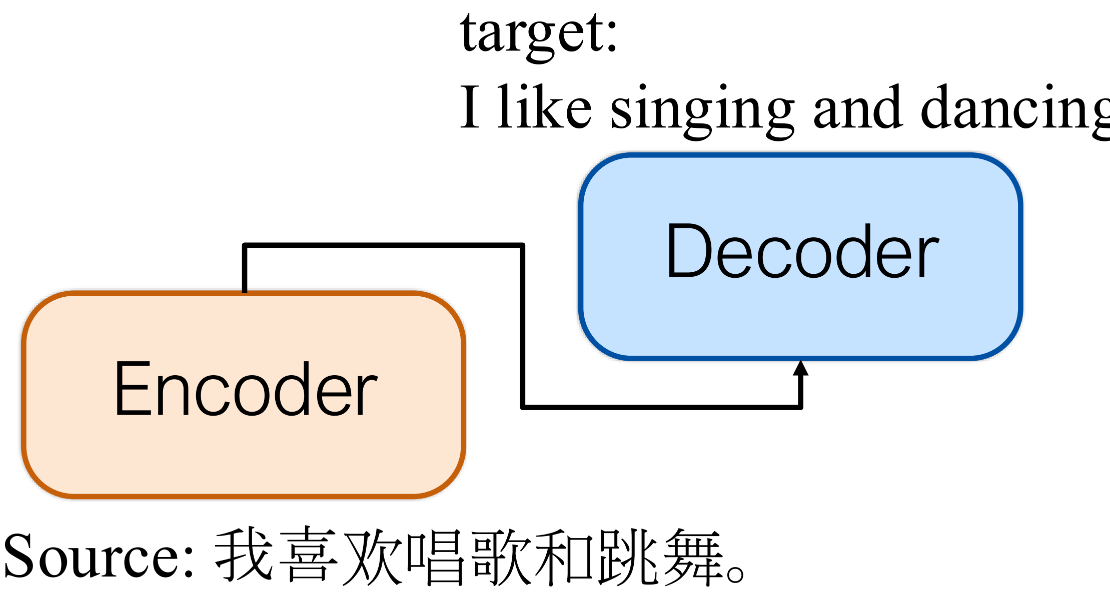
Training Transformer

$$P(Y|X) = \prod P(y_t|y_{<t}, x)$$

- Training loss: Cross-Entropy

$$l = -\sum_n \sum_t \log f_\theta(x_n, y_{n,1}, \ldots, y_{n,t-1})$$

- Teacher-forcing during training.
- pretend to know groundtruth for prefix



Training Transformer for MT

- Dropout
  - Applied to before residual
  - and to embedding, pos emb.
  - p=0.1 ~ 0.3
- Label smoothing
  - 0.1 probability assigned to non-truth
- Vocabulary:
  - En-De: 37K using BPE
  - En-Fr: 32k word-piece (similar to BPE)

Label Smoothing

- Assume $y \in R^n$ is the one-hot encoding of label

$$y_i = \begin{cases} 1 & \text{if belongs to class } i \\ 0 & \text{otherwise} \end{cases}$$

- Approximating 0/1 values with softmax is hard

- The smoothed version

$$y_i = \begin{cases} 1 - \epsilon & \text{if belongs to class } i \\ \epsilon/(n-1) & \text{otherwise} \end{cases}$$

  - Commonly use $\epsilon = 0.1$

Training

- Batch
  - group by approximate sentence length
  - still need shufflingHardware
  - one machine with 8 GPUs (in 2017 paper)
  - base model: 100k steps (12 hours)
  - large model: 300k steps (3.5 days)

- Adam Optimizer
  - increase learning rate during warmup, then decrease
  - $\eta = \frac{1}{\sqrt{d}} min(\frac{1}{\sqrt{t}}, \frac{t}{\sqrt{t_0^3}})$

ADAM

$$m_{t+1} = \beta_1 m_t - (1 - \beta_1)\nabla\ell(x_t)$$
$$v_{t+1} = \beta_2 v_t + (1 - \beta_2)(\nabla\ell(x_t))^2$$
$$\widehat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}$$
$$\hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}$$
$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}} + \epsilon}\widehat{m}_{t+1}$$

Model Average

- A single model obtained by averaging the last 5 checkpoints, which were written at 10-minute interval (base)

- decoding length: within source length + 50
    - more on decoding in next lecture

Summary

- Sequence-to-sequence encoder-decoder framework for conditional generation, including Machine Translation
- Key components in Transformer (why each?)
  - Positional Embedding (to distinguish tokens at different pos)
  - Multihead attention
  - Residual connection
  - layer norm
  - FFN

Code Go-through

https://nlp.seas.harvard.edu/annotated-transformer/