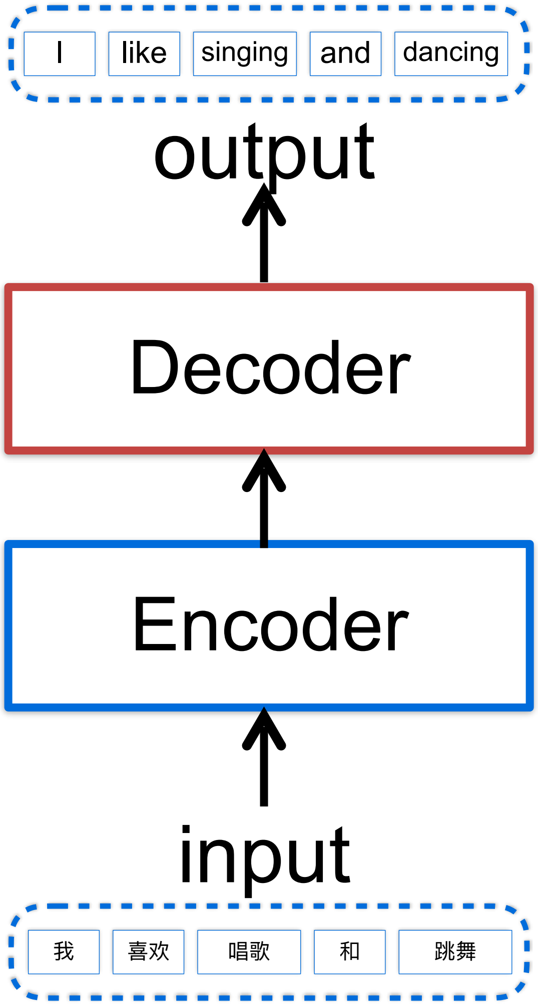
CS 190I
Deep Learning
Sequence-to-sequence Learning and Transformer

Lei Li (leili@cs)
UCSB

Outline

- Recurrent Neural Network (last lecture)
- Sequence-to-sequence learning (this lecture)
- Transformer network (this lecture)
- Pretrained Language Models (next)
  - BERT
  - GPT, ChatGPT

Encoder-Decoder Paradigm



A generic formulation for many tasks

Encoder-Decoder Paradigm

我喜欢唱歌和跳舞。 Machine Translation → I like singing and dancing.

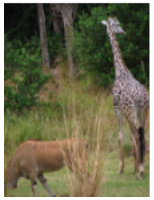


Image Captioning → A giraffe standing next to forest

Automatic Speech Recognition → “Alexa, turn off the lights”

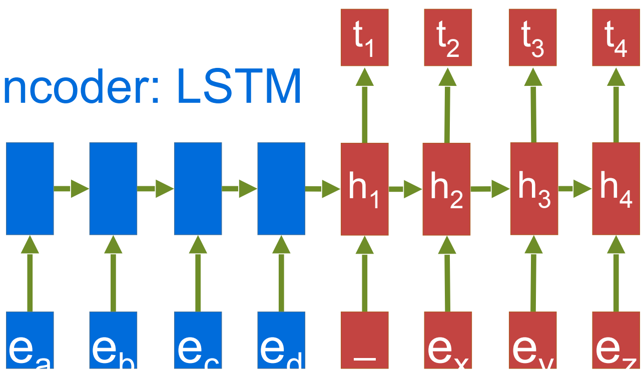
Graduate student reading papers on beach Text-to-Image Generation →

Sequence To Sequence (Seq2seq)

- Machine translation as directly learning a function mapping from source sequence to target sequence

target:
The weather is nice

Encoder: LSTM



Source: 天 气 很 好

Decoder: LSTM

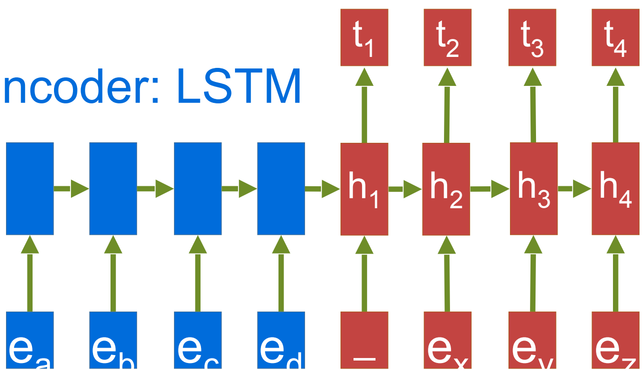
Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014

Sequence To Sequence (Seq2seq)

- Machine translation as directly learning a function mapping from source sequence to target sequence

target:
The weather is nice

Encoder: LSTM



Source: 天 气 很 好

Decoder: LSTM

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

Training loss: Cross-Entropy

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

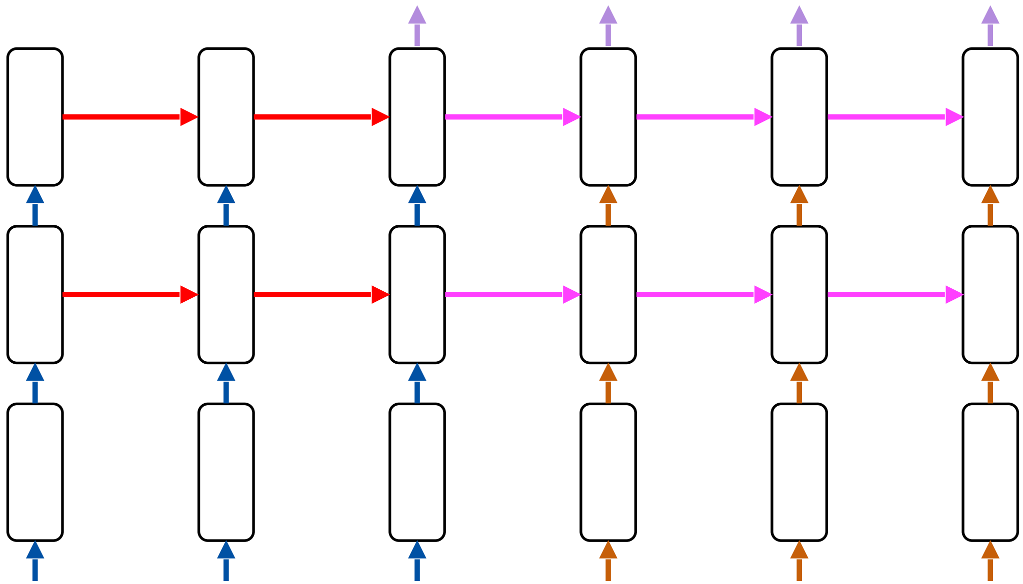
Teacher-forcing during training.

(pretend to know groundtruth for prefix)

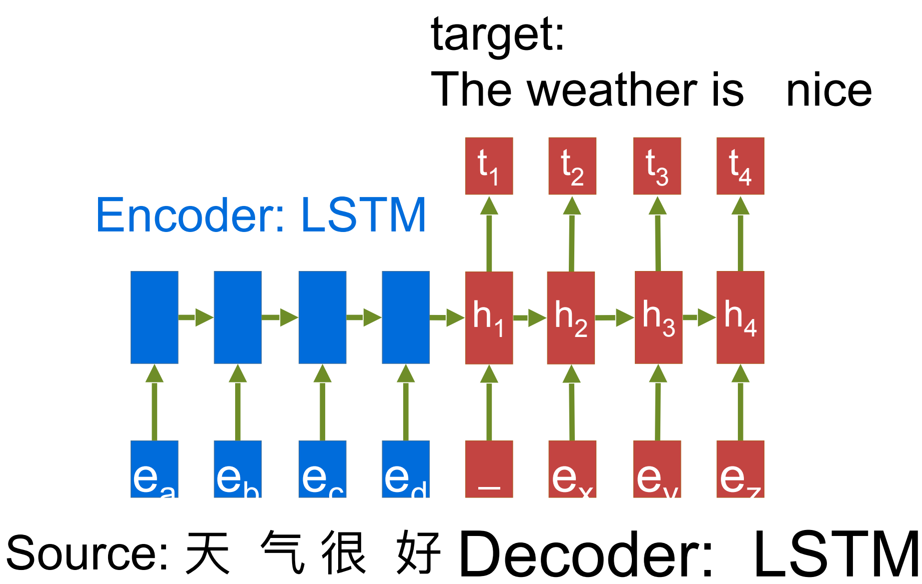
Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014

Stacked LSTM for seq-2-seq

- More layers of LSTM



Limitation of RNN/LSTM

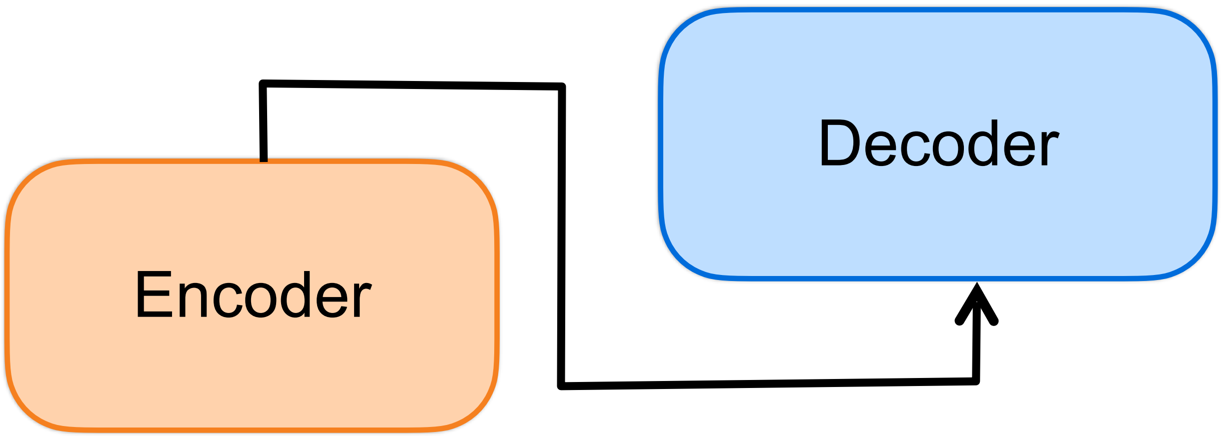


- No full context (only one-side)
  - Bidirectional LSTM encoder could alleviate
  - But still no long context
- Sequential computation in nature (encoder)
  - not possible to parallelize the computation
- Vanishing gradient

Motivation for New Network Architecture

- Full context and parallel: use Attention in both encoder and decoder
- no recurrent

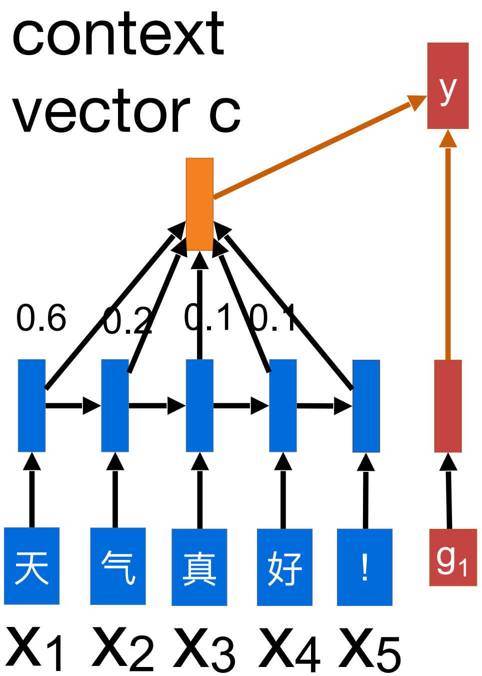
target:
I like singing and dancing.



Source: 我喜欢唱歌和跳舞。

Attention

Each output token depends on input tokens differently



A context vector c represents the related source context for current predicting word.

$$\alpha_{mj} = \text{Softmax}(D(g_m, h_{1...n})) = \frac{\exp(D(g_m, h_j))}{\sum_k \exp(D(g_m, h_k))}$$

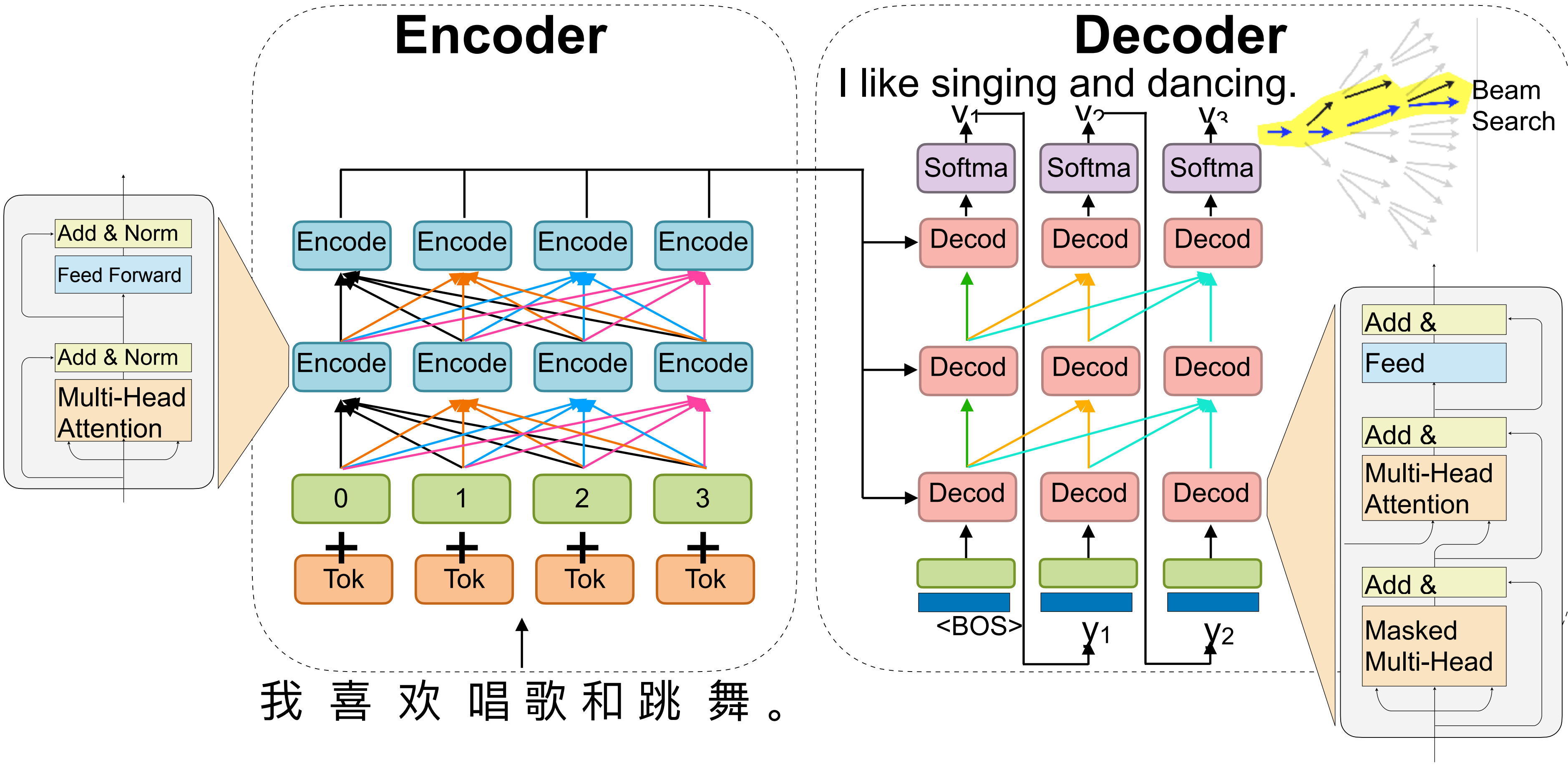
$$c_m = \sum_j \alpha_{mj} h_j$$

$$D(g_m, h_j) = g_m \cdot h_j$$

The probability of word y_i is computed as:

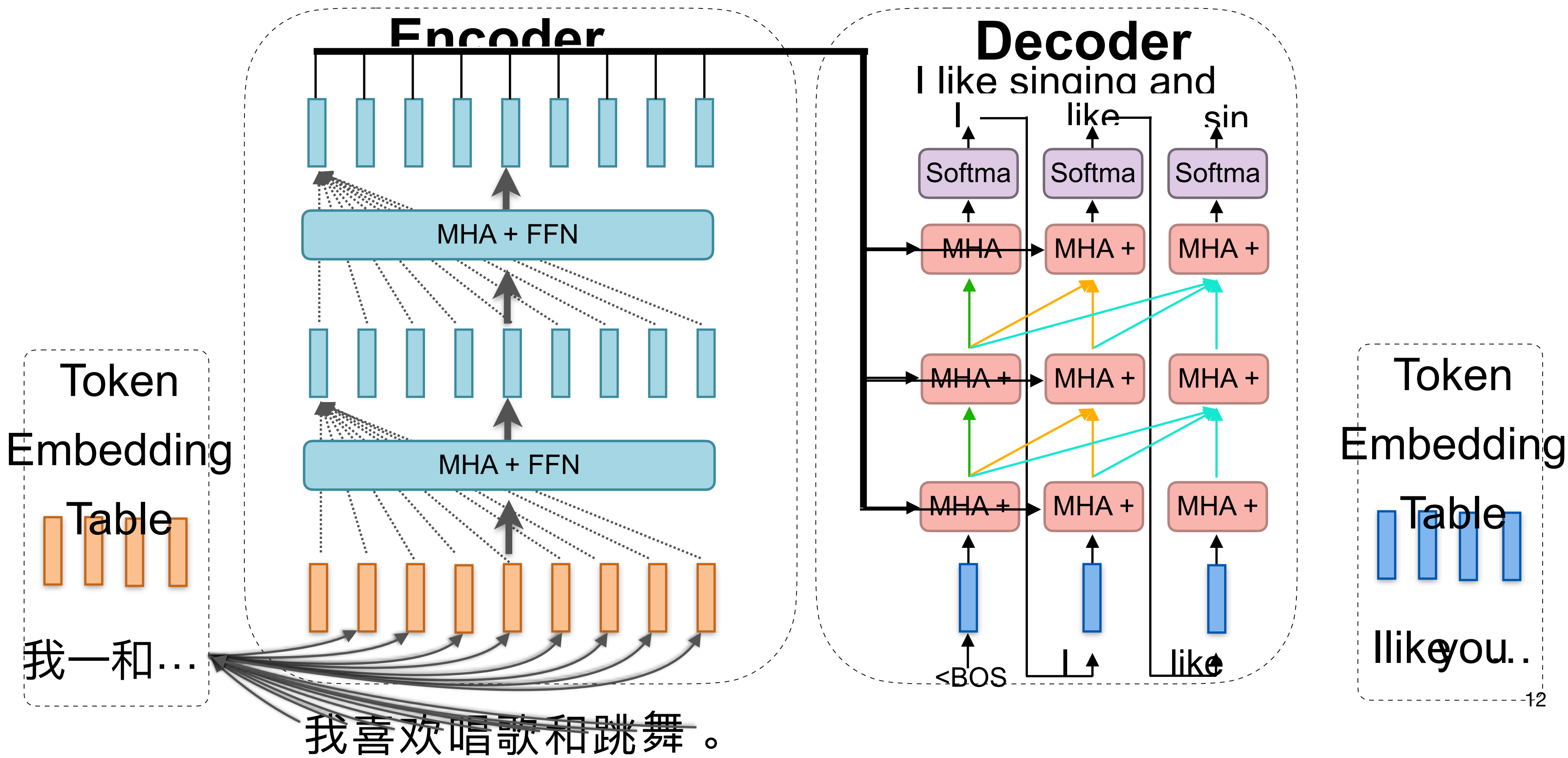
$$p(y_m) = \text{Softmax}(W \cdot \begin{bmatrix} g_m \\ c_m \end{bmatrix} + b)$$

Transformer



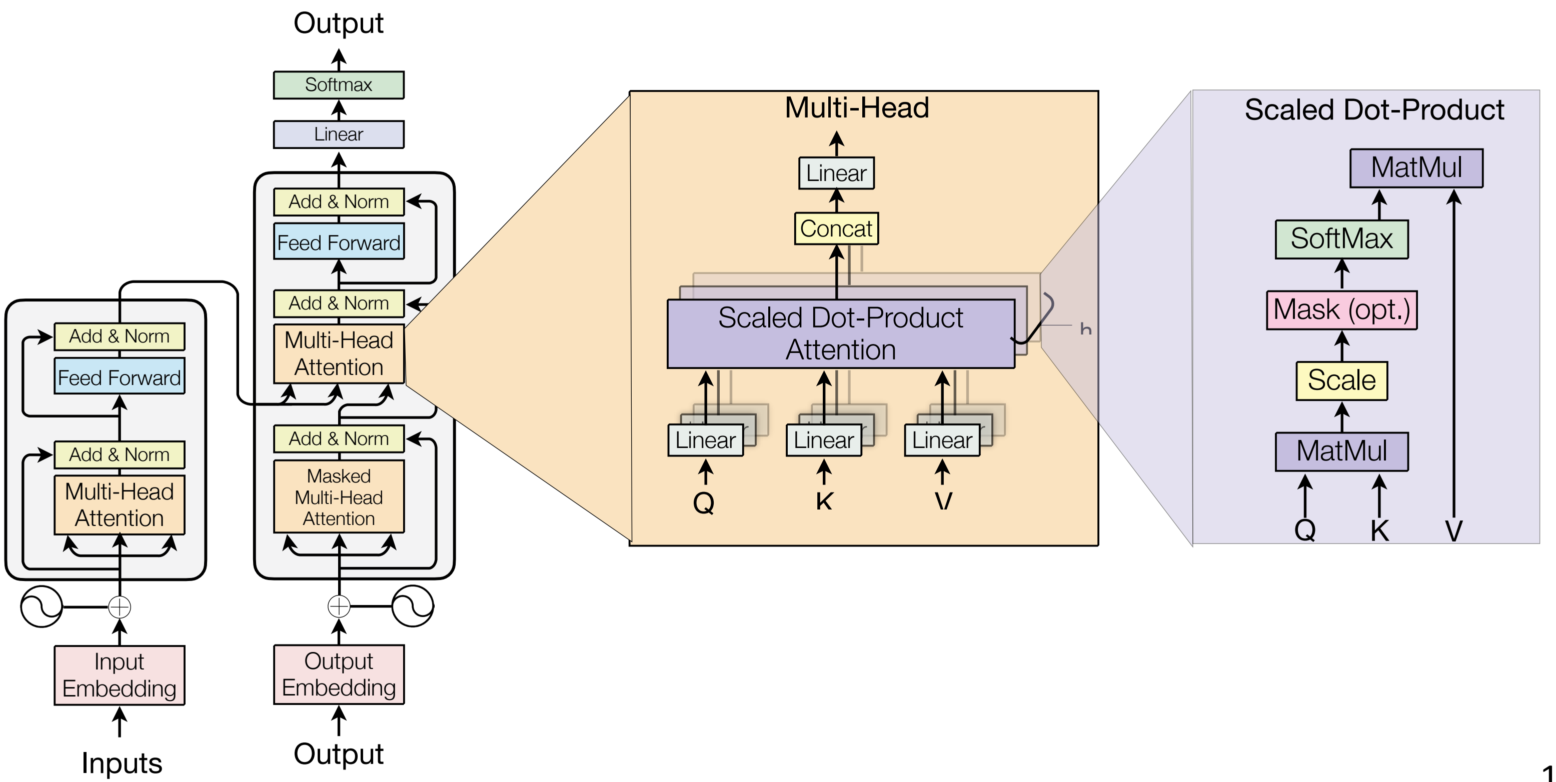
Vaswani et al. Attention is All You Need. 2017

How Does Transformer Translate?

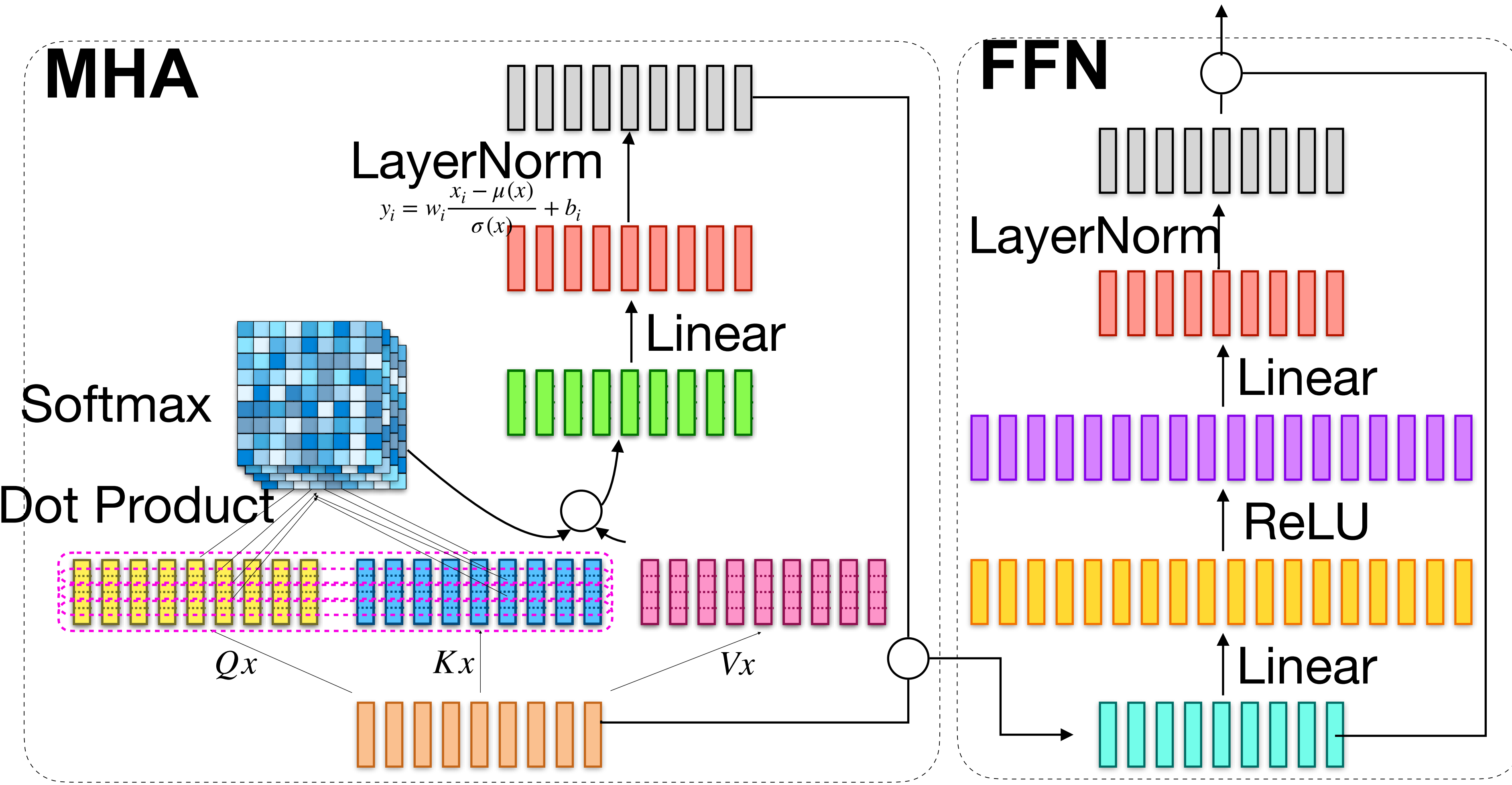


Transformer Multi-head Attention

- C layers of encoder (=6)
- D layers of decoder (=6)

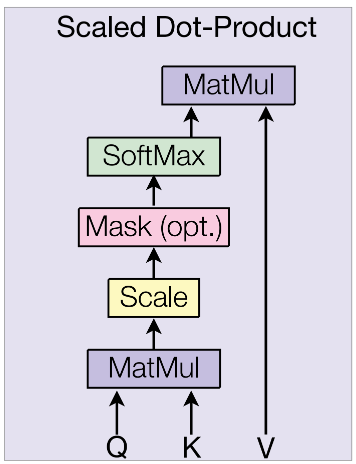


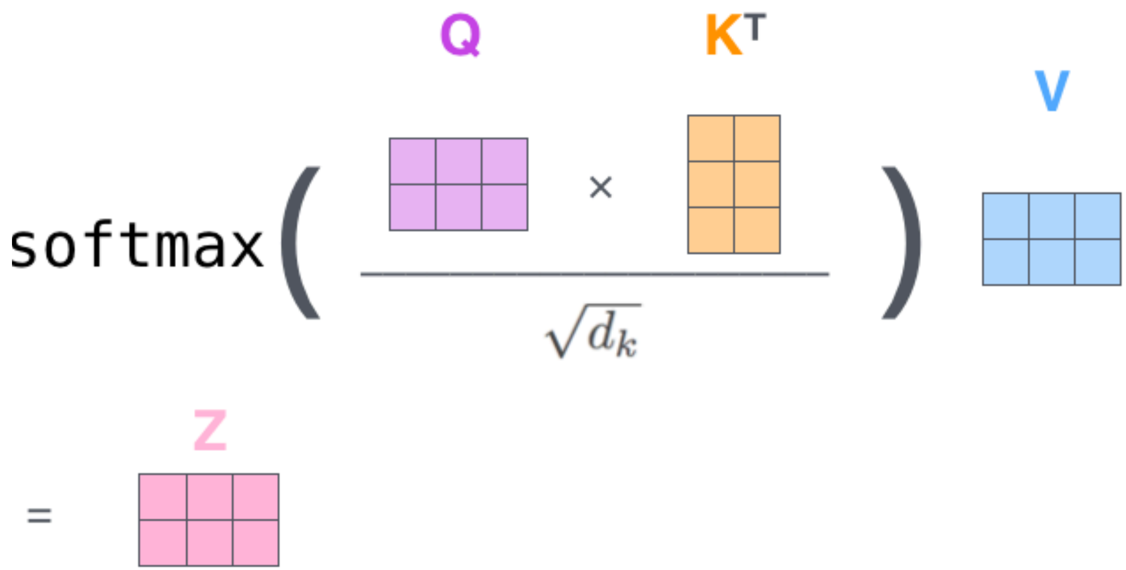
MultiHead Attention And Feed Forward Network



Scaled Dot-Product Attention

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



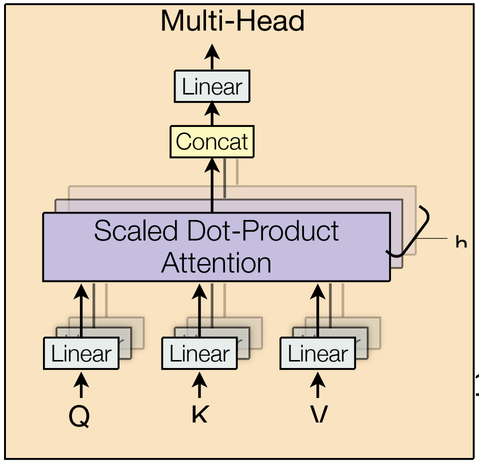


Multi-head Attention

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

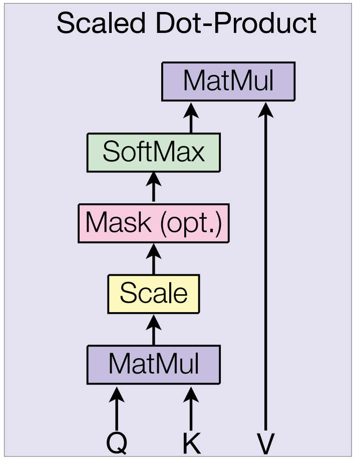
$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, \ldots, \text{Head}_h)W^o$$



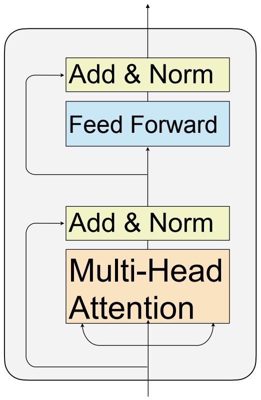
Self-Attention for Decoder

- Maskout right side before softmax (-inf)



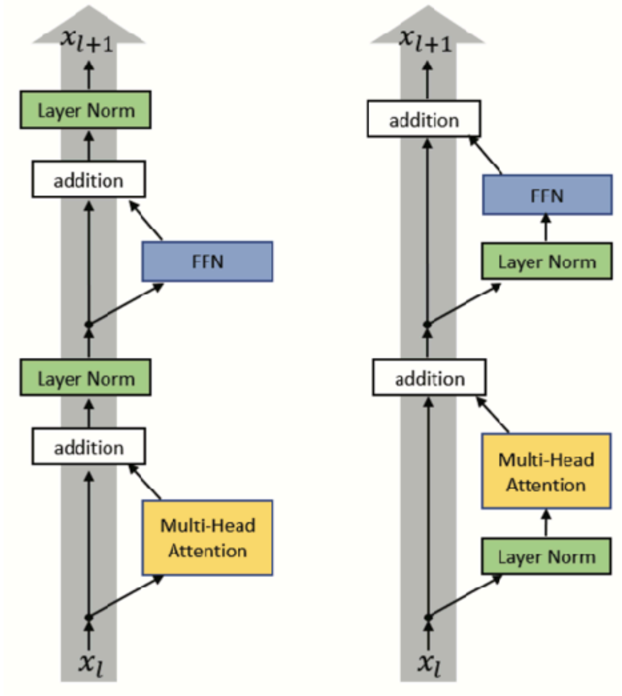
Feedforward Net

- $\mathrm{FFN}(x) = \max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2$
- internal dimension size = 2048 (in Vaswani 2017)



Residual Connection and Layer Normalization

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

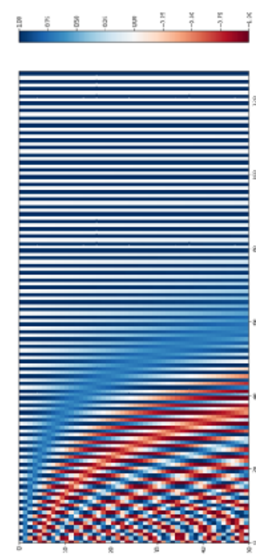


Embedding

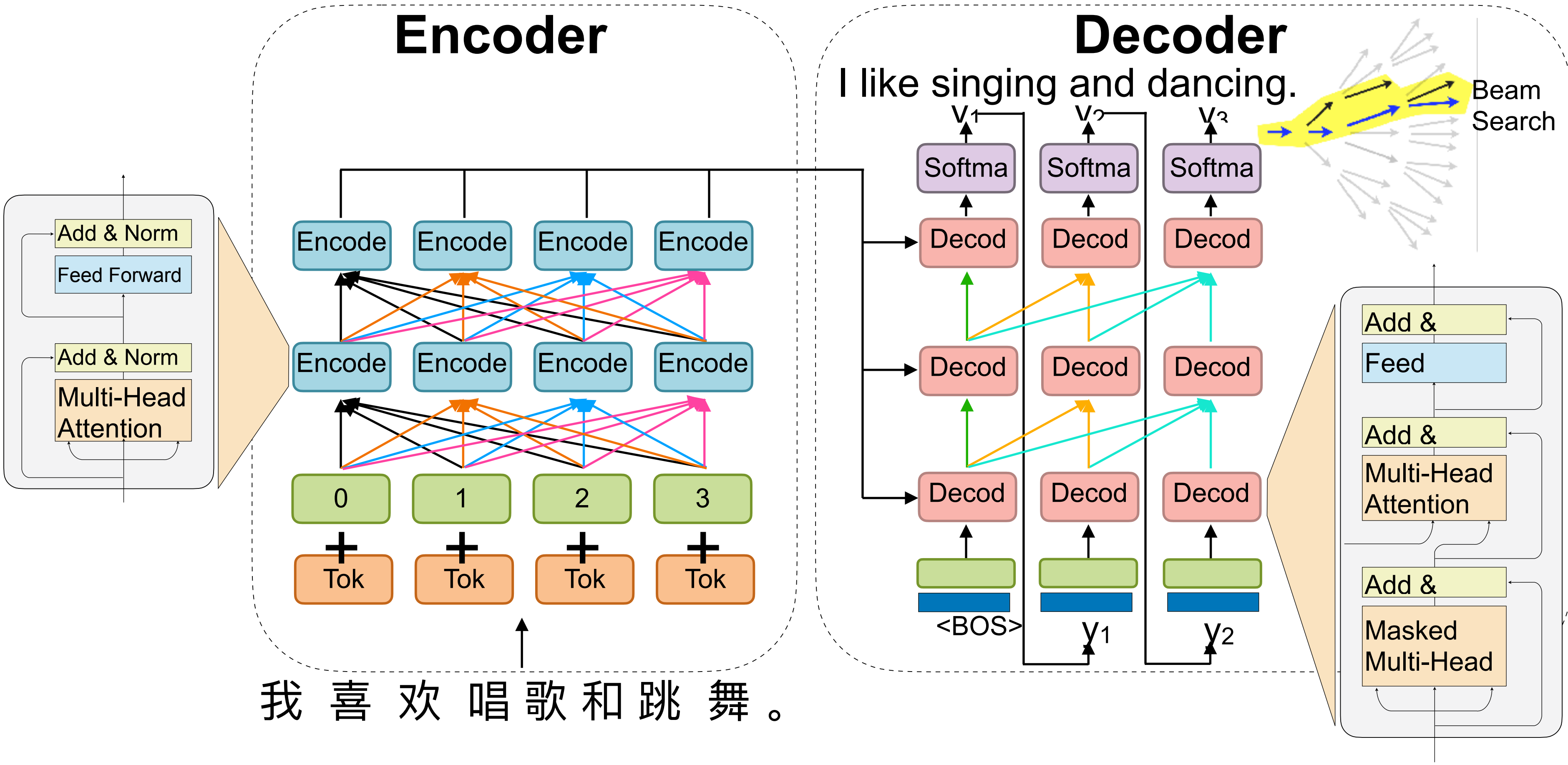
- Token Embedding: 512 (base), 1024 (large)
  - Shared (tied) input and output embedding
- Positional Embedding:
  - to distinguish words in different position, Map position labels to embedding, dimension is same as Tok Emb

$$PE_{pos,2i} = \sin(\frac{pos}{1000^{2i/d}})$$

$$PE_{pos,2i+1} = \cos(\frac{pos}{1000^{2i/d}})$$



Transformer



我 喜 欢 唱歌和跳 舞 。

Vaswani et al. Attention is All You Need. 2017

Training Loss

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

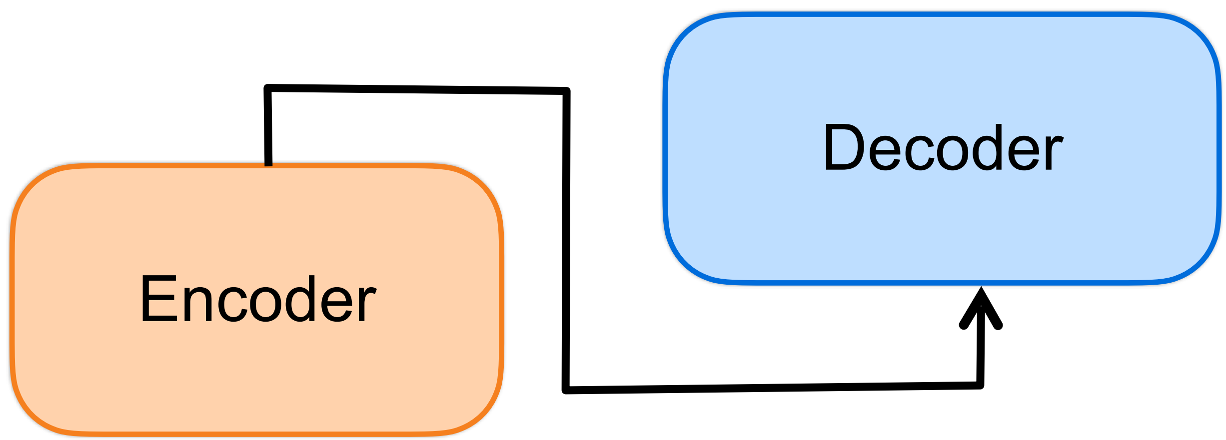
Training loss: Cross-Entropy

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

Teacher-forcing during training.

(pretend to know groundtruth for prefix)

target:
I like singing and dancing.



Source: 我喜欢唱歌和跳舞。

Training

- 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 $\mathbf{y} \in \mathbb{R}^n$ 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 shuffling
- Hardware
  - 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

$$m_{t+1} = \beta_1 m_t - (1-\beta_1)\nabla \ell(x_t)$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)(\nabla \ell(x_t))^2$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1^{t+1}}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2^{t+1}}$$

$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}} + \epsilon}\hat{m}_{t+1}$$

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

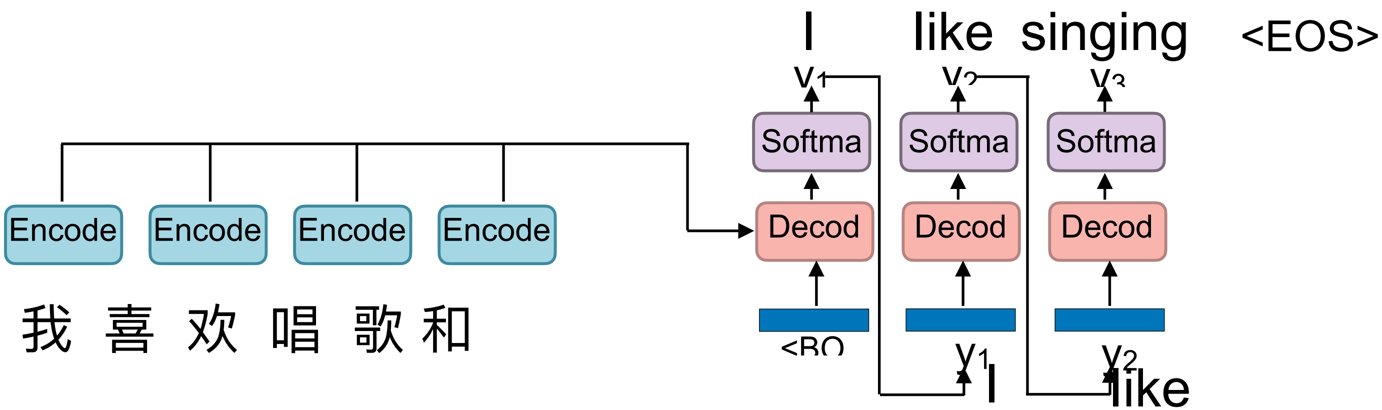
Quiz

- https://edstem.org/us/courses/31035/lessons/57196/slides/321725

Sequence Decoding

Autoregressive Generation

greedy decoding: output the token with max next token prob



But, this is not necessary the best

Inference

- Now already trained a model θ
- Decoding/Generation: Given an input sentence x, to generate the target sentence y that maximize the probability $P(y \mid x; \theta)$
- $\underset{y}{\operatorname{argmax}} P(y \mid x) = f_\theta(x, y)$
- Two types of error
  - the most probable translation is bad → fix the model
  - search does not find the most probably translation → fix the search
- Most probable translation is not necessary the highest BLEU one!

Decoding

- $\underset{y}{\operatorname{argmax}} P(y \mid x) = f_\theta(x, y)$
- naive solution: exhaustive search
  - too expensive
- Beam search
  - (approximate) dynamic programming

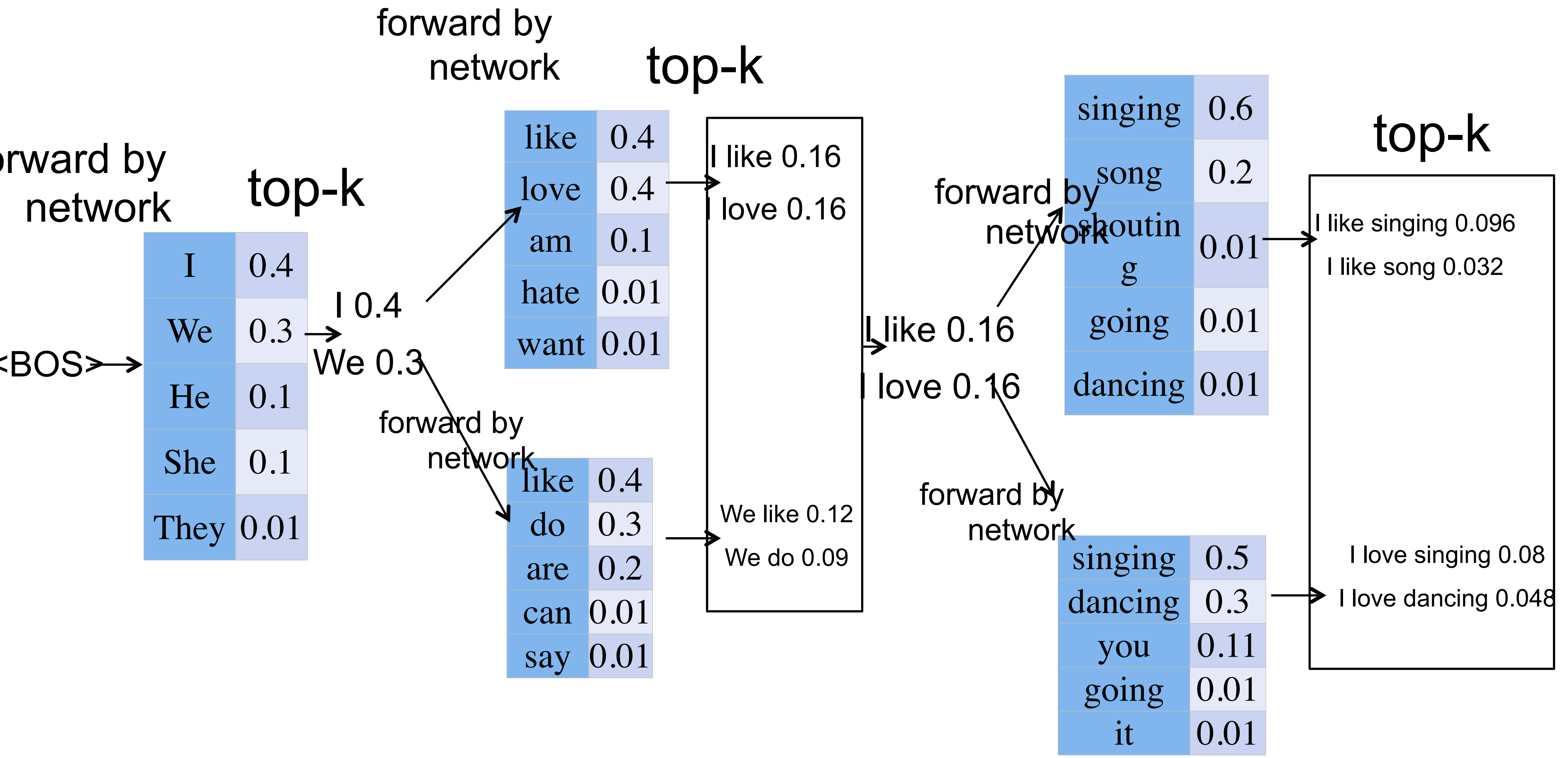
Beam Search

- start with empty S
- at each step, keep k best partial sequences
- expand them with one more forward generation
- collect new partial results and keep top-k

Beam Search (pseudocode)

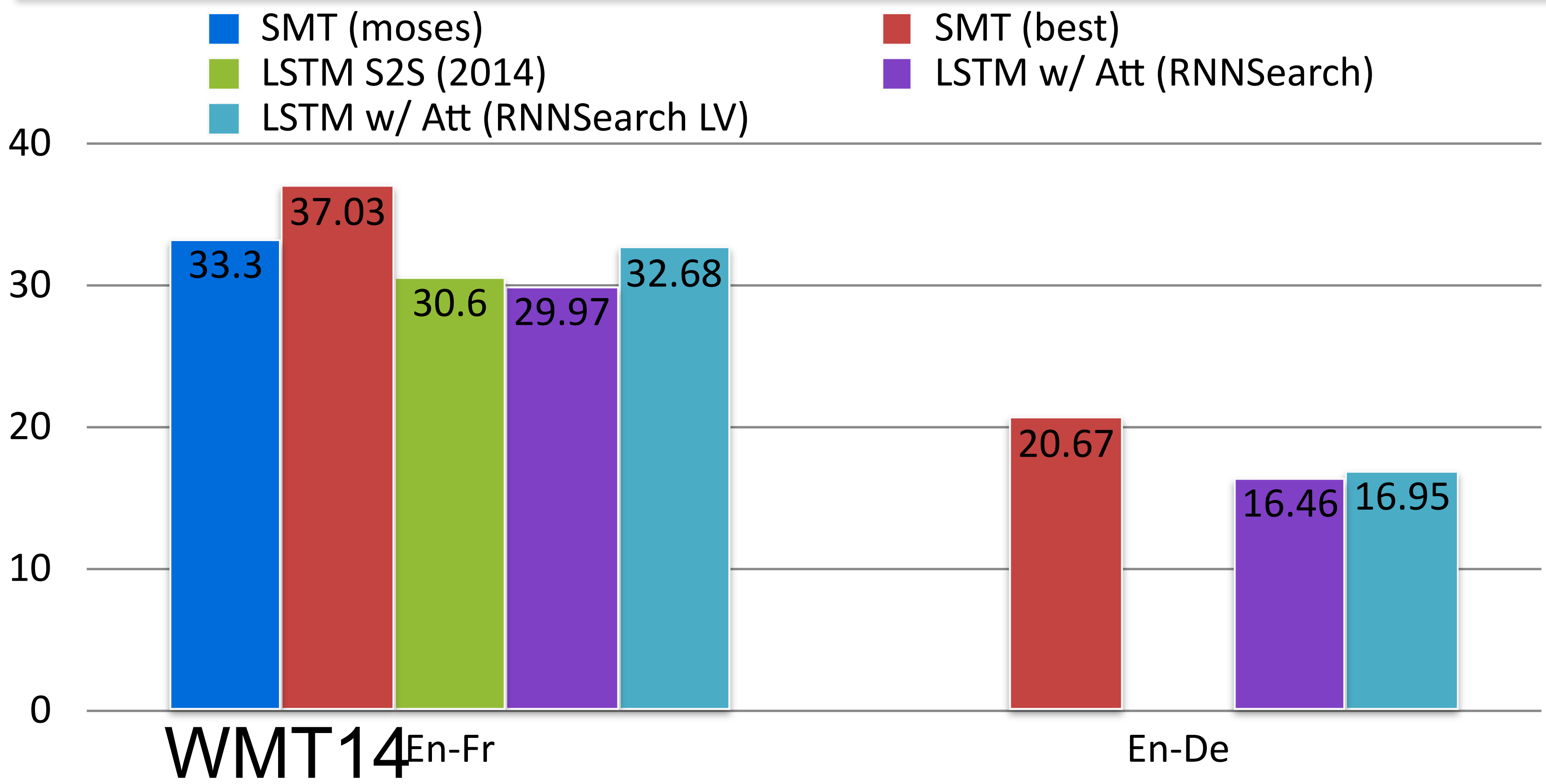
```
best_scores = []
add {[0], 0.0} to best_scores # 0 is for beginning of sentence token
for i in 1 to max_length:
  new_seqs = PriorityQueue()
  for (candidate, s) in best_scores:
    if candidate[-1] is EOS:
        prob = all -inf
        prob[EOS] = 0
      else:
      prob = using model to take candidate and compute next token
probabilities (logp)
    pick top k scores from prob, and their index
    for each score, index in the top-k of prob:
      new_candidate = candidate.append(index)
      new_score = s + score
      if not new_seqs.full():
```

Beam Search


<BOS>
forward by network
top-k
I 0.4
We 0.3
He 0.1
She 0.1
They 0.01
I 0.4
We 0.3
forward by network
top-k
like 0.4
love 0.4
am 0.1
hate 0.01
want 0.01
forward by network
like 0.4
do 0.3
are 0.2
can 0.01
say 0.01
I like 0.16
I love 0.16
We like 0.12
We do 0.09
I like 0.16
I love 0.16
forward by network
singing 0.6
song 0.2
shouting 0.01
going 0.01
dancing 0.01
forward by network
singing 0.5
dancing 0.3
you 0.11
going 0.01
it 0.01
top-k
I like singing 0.096
I like song 0.032
I love singing 0.08
I love dancing 0.048

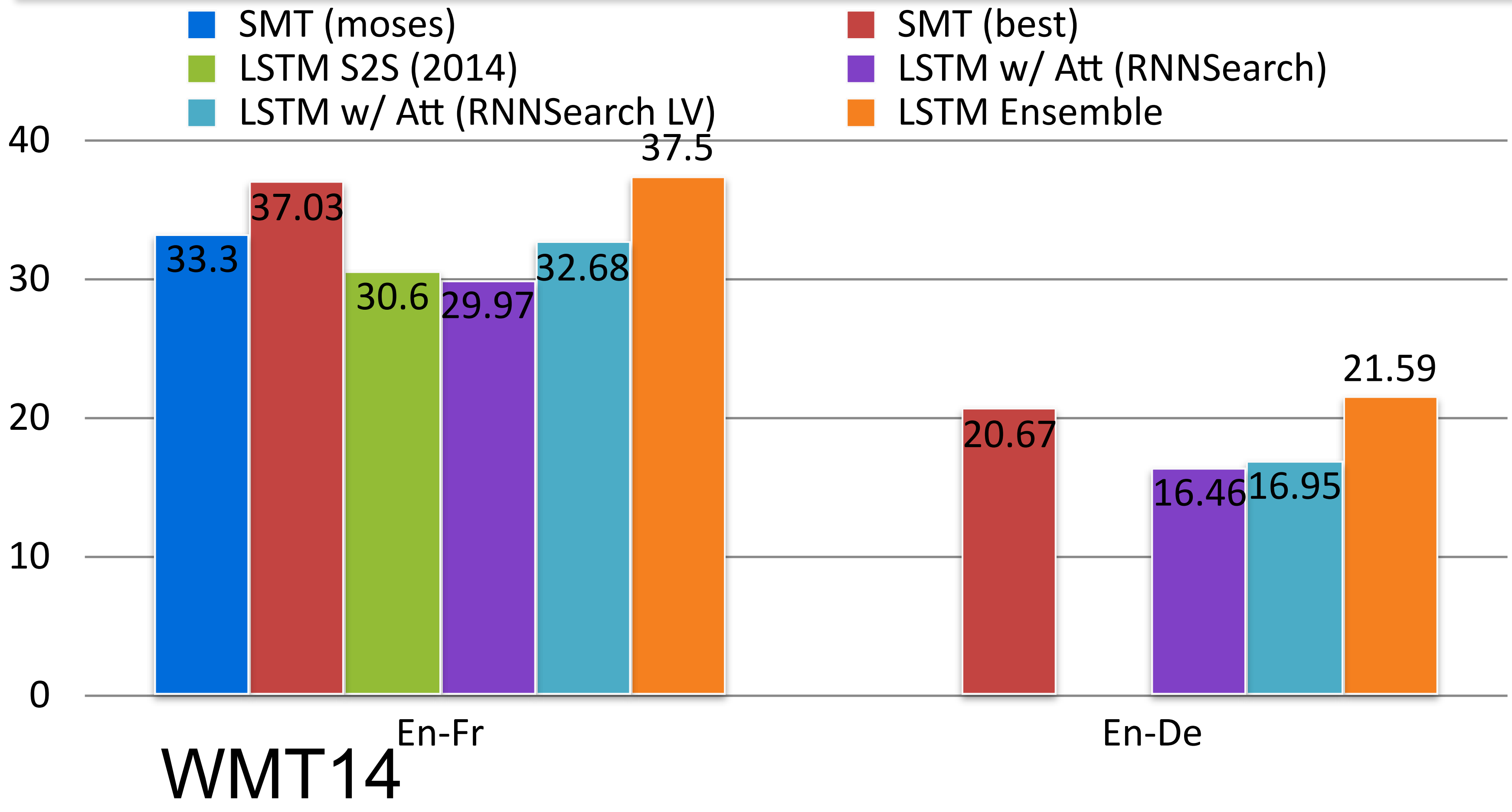
Machine Translation using Seq2seq and Transformer

LSTM Seq2Seq w/ Attention



Jean et al. On Using Very Large Target Vocabulary for Neural Machine Translation. 2015

Performance with Model Ensemble



Luong et al. Effective Approaches to Attention-based Neural Machine Translation. 2015

Results on WMT14

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Effectiveness of Choices

- num. head
- dim of key
- num layers
- hid dim
- ffn dim
- dropout
- pos emb

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Deep Transformer

- 30 ~ 60 encoder
- 12 decoder
- dynamic linear combination of layers (DLCL)
  - or. deeply supervised
  - combine output from all layers

Wang et al. Learning Deep Transformer Models for Machine Translation, 2019.

Model		Param.	Batch (×4096)	Updates (×100k)	†Times	BLEU	Δ
Vaswani et al. (2017) (Base)		65M	1	1	reference	27.3	-
Bapna et al. (2018)-deep (Base, 16L)		137M	-	-	-	28.0	-
Vaswani et al. (2017) (Big)		213M	1	3	3x	28.4	-
Chen et al. (2018a) (Big)		379M	16	†0.075	1.2x	28.5	-
He et al. (2018) (Big)		†210M	1	-	-	29.0	-
Shaw et al. (2018) (Big)		†210M	1	3	3x	29.2	-
Dou et al. (2018) (Big)		356M	1	-	-	29.2	-
Ott et al. (2018) (Big)		210M	14	0.25	3.5x	29.3	-
post-norm	Transformer (Base)	62M	1	1	1x	27.5	reference
	Transformer (Big)	211M	1	3	3x	28.8	+1.3
	Transformer-deep (Base, 20L)	106M	2	0.5	1x	failed	failed
	DLCL (Base)	62M	1	1	1x	27.6	+0.1
	DLCL-deep (Base, 25L)	121M	2	0.5	1x	29.2	+1.7
pre-norm	Transformer (Base)	62M	1	1	1x	27.1	reference
	Transformer (Big)	211M	1	3	3x	28.7	+1.6
	Transformer-deep (Base, 20L)	106M	2	0.5	1x	28.9	+1.8
	DLCL (Base)	62M	1	1	1x	27.3	+0.2
	DLCL-deep (Base, 30L)	137M	2	0.5	1x	**29.3**	**+2.2**

Wang et al. Learning Deep Transformer Models for Machine Translation, 2019.

Model	Param.	newstest17	newstest18	$\Delta_{avg.}$
Wang et al. (2018a) (post-norm, Base)	102.1M	25.9	-	-
pre-norm Transformer (Base)	102.1M	25.8	25.9	reference
pre-norm Transformer (Big)	292.4M	26.4	27.0	+0.9
pre-norm DLCL-deep (Base, 25L)	161.5M	26.7	27.1	+1.0
pre-norm DLCL-deep (Base, 30L)	177.2M	**26.9**	**27.4**	**+1.3**

Table 4: BLEU scores [%] on WMT'18 Chinese-English translation.

Wang et al. Learning Deep Transformer Models for Machine Translation, 2019.

Hot Topics in MT

- Parallel Decoding (e.g. NAT, GLAT, DAT,…)
- Low-resource MT
- Unsupervised MT
- Multilingual NMT, Zero-shot NMT
- Speech-to-text translation
  - (Offline) ST
  - Streaming ST

Summary

- Key components in Transformer
  - Positional Embedding (to distinguish tokens at different pos)
  - Multihead attention
  - Residual connection
  - layer norm
- Transformer is effective for machine translation, and many other tasks

Next Up

- Pretraining for NLP
  - BERT
  - GPT, ChatGPT