FAST AND ACCURATE READING COMPREHENSION BY COMBINING SELF-ATTENTION AND CONVOLUTION

Adams Wei Yu[1,*], **David Dohan**[2,‡], **Minh-Thang Luong**[2,†]
{weiyu}@cs.cmu.edu, {ddohan,thangluong}@google.com
[1]Carnegie Mellon University, [2]Google Brain

Rui Zhao, Kai Chen, Mohammad Norouzi, Quoc V. Le
Google Brain

ICLR 2018

——李芸

- Background
  - Machine reading comprehension and automated question answering have become an important topic in the NLP domain.
- Task
  - Read Comprehension Task
  - Given question and document
  - Find a span in the document as a answer to the question.
- Datasets:
  - SQuAD(Rajpurkar et al.,2016)
  - each example of SQuAD is a triple of (d, q, a)

Related work

Current end-to-end machine reading comprehension and question answering models are primarily based on recurrent neural networks (RNNs) with attention.

For example: BiDAF (Seo et al., 2016)

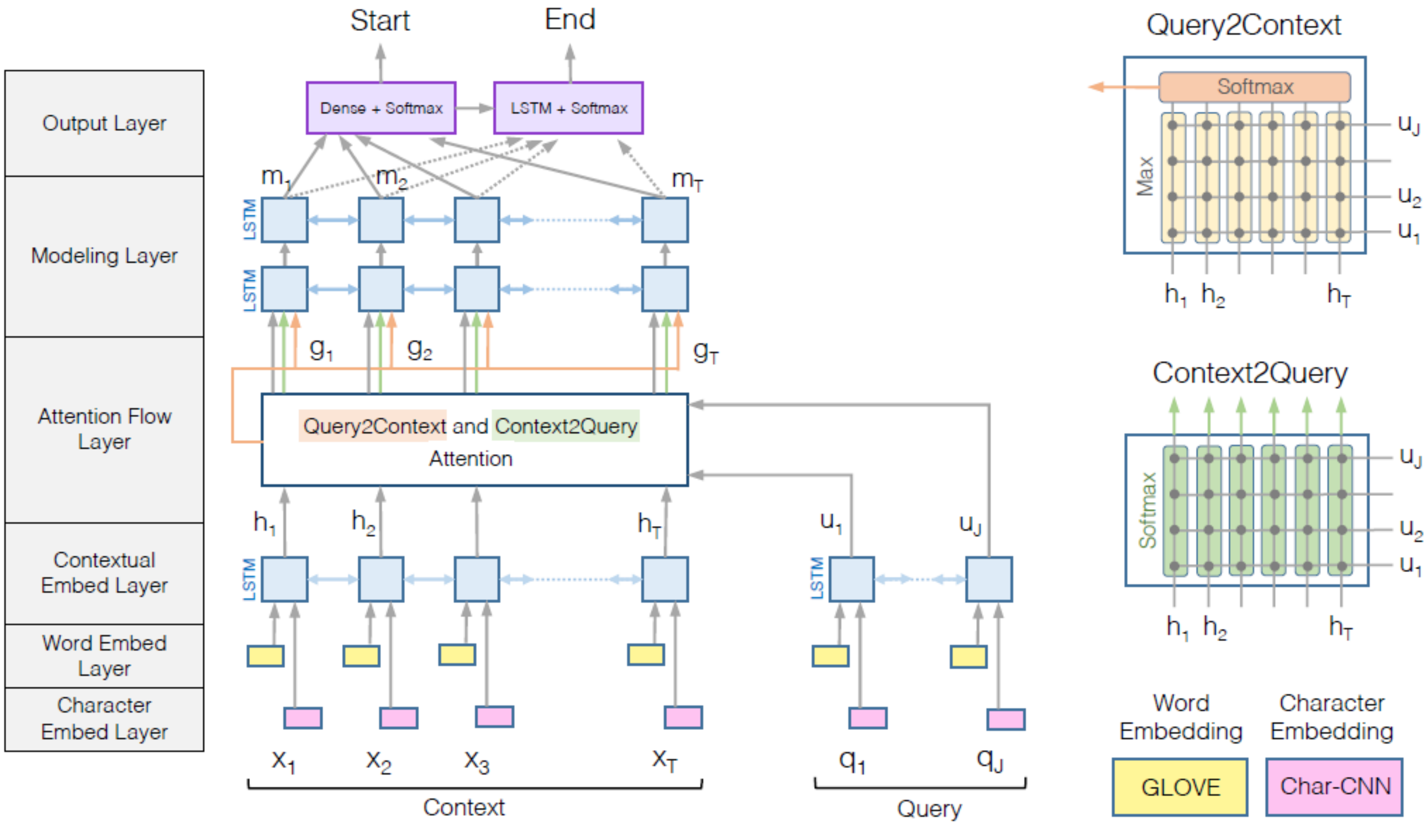


Figure 1: BiDirectional Attention Flow Model *(best viewed in color)*

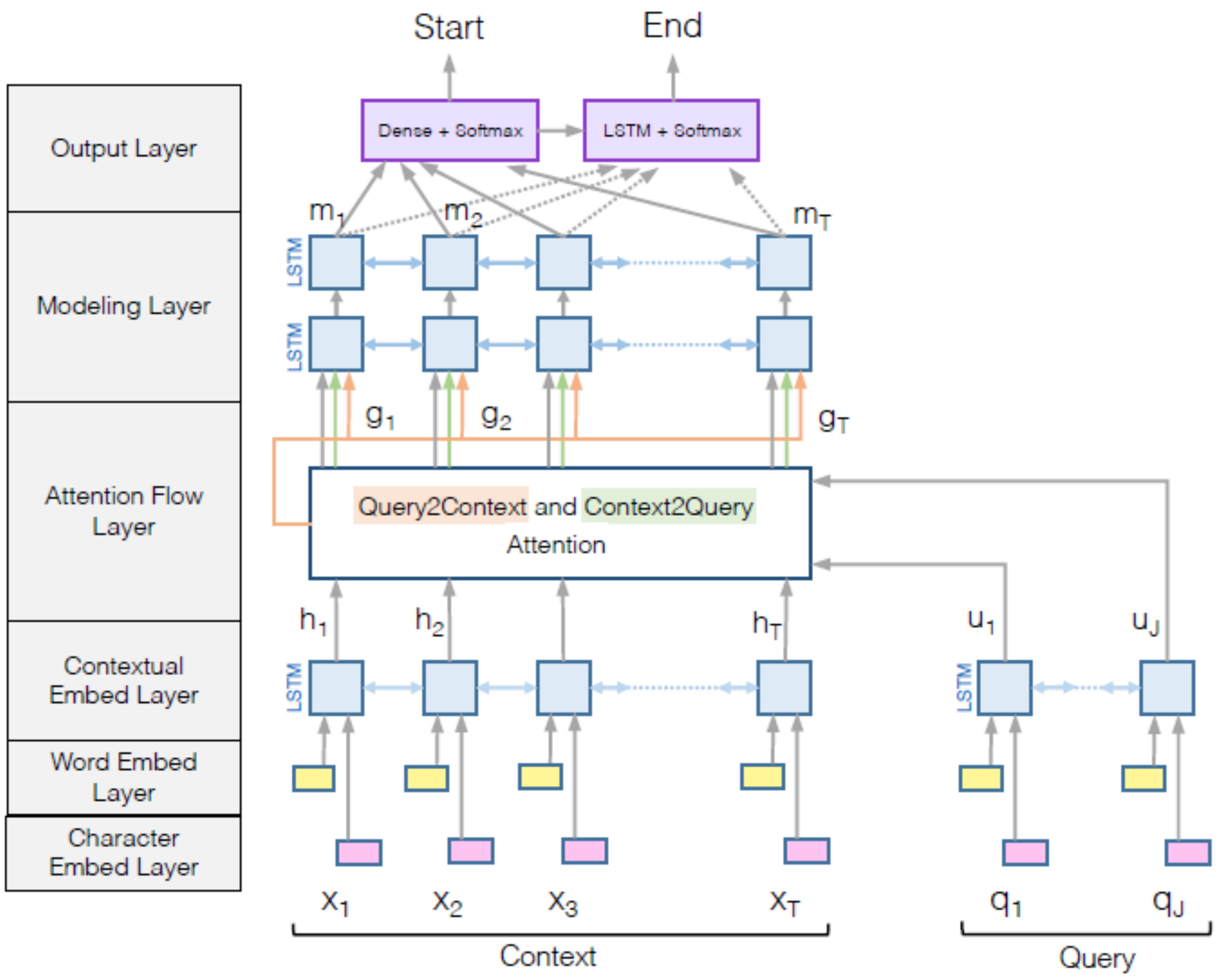
Despite their success, these models are often slow for both training and inference due to the sequential nature of RNNs.

In this paper

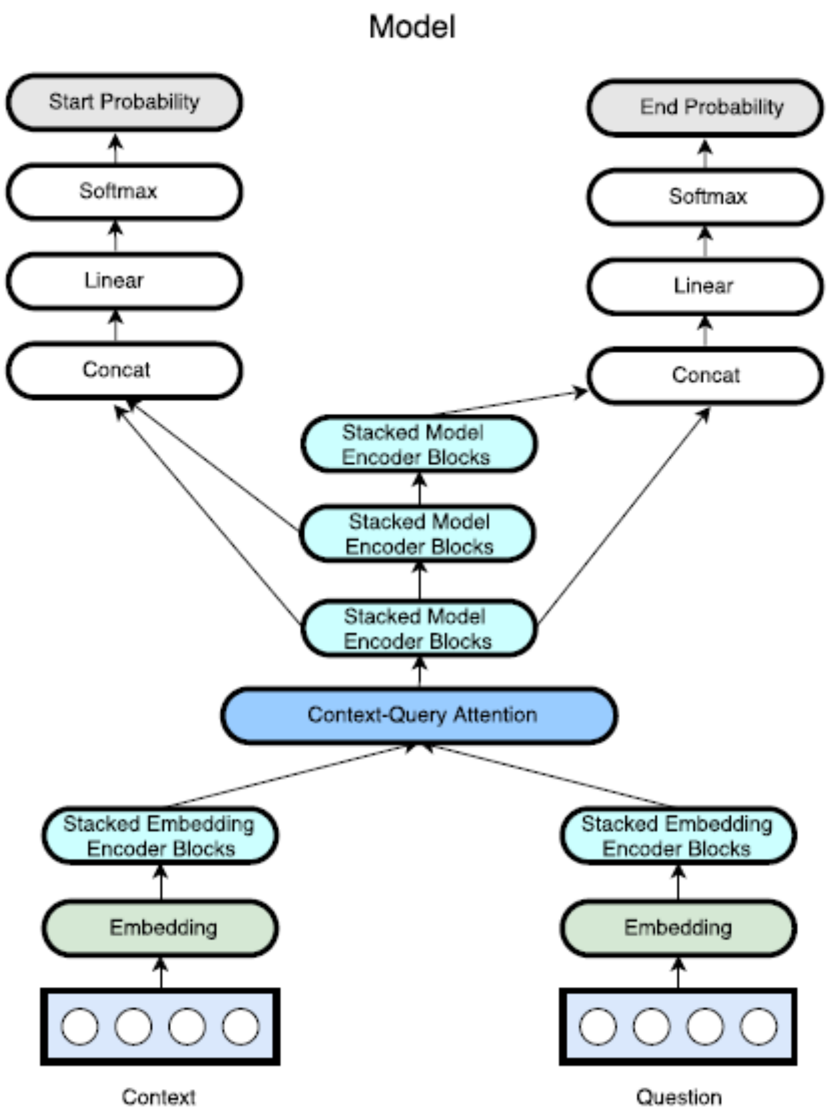
- Propose a new Q&A architecture that does not require recurrent networks.

- Combine self-attention and convolution

Model contrast

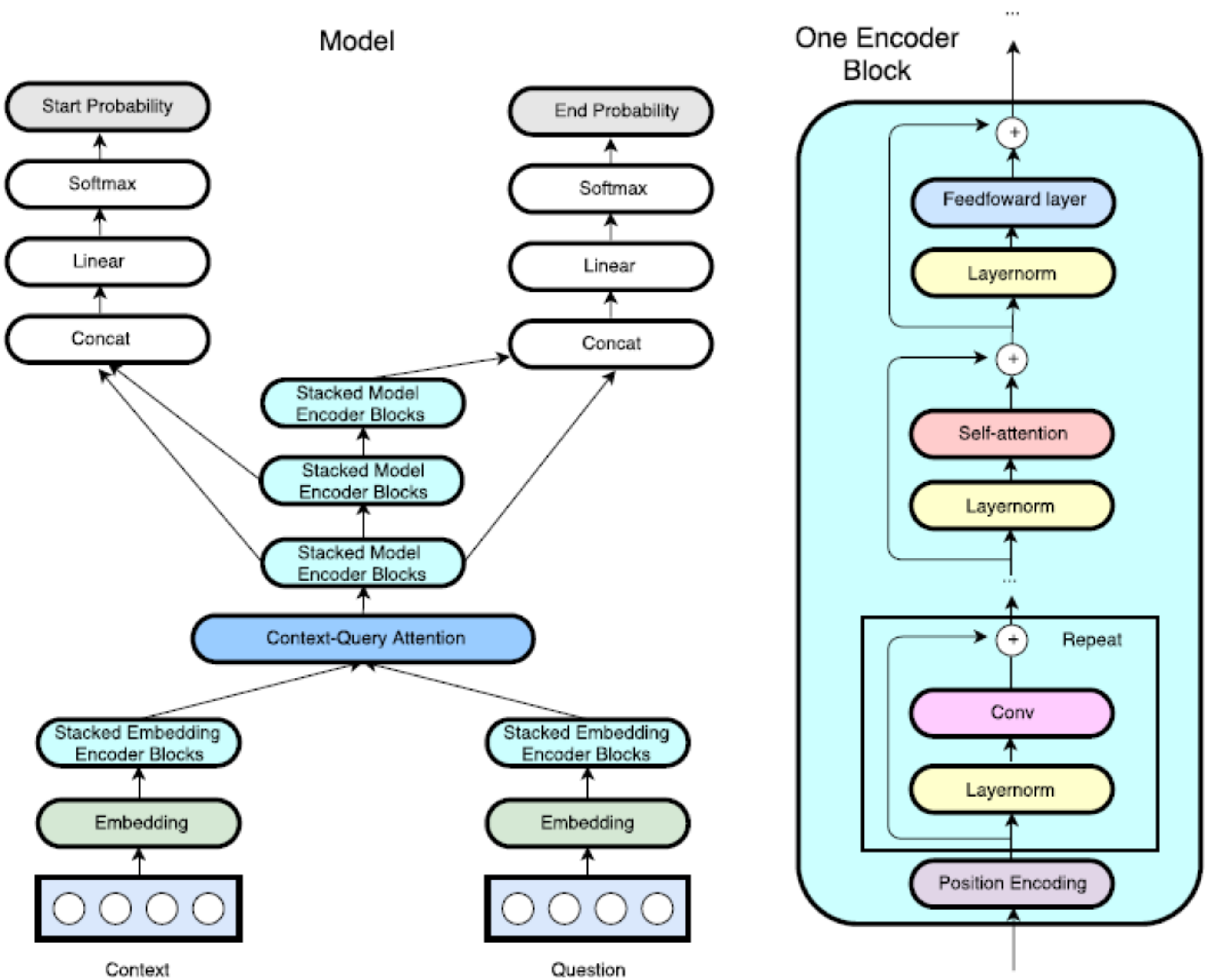


BiDAF



Our Model

Model


Model
Start Probability
Softmax
Linear
Concat
End Probability
Softmax
Linear
Concat
Stacked Model Encoder Blocks
Stacked Model Encoder Blocks
Stacked Model Encoder Blocks
Context-Query Attention
Stacked Embedding Encoder Blocks
Embedding
Context
Stacked Embedding Encoder Blocks
Embedding
Question
One Encoder Block
Feedfoward layer
Layernorm
Self-attention
Layernorm
Repeat
Conv
Layernorm
Position Encoding

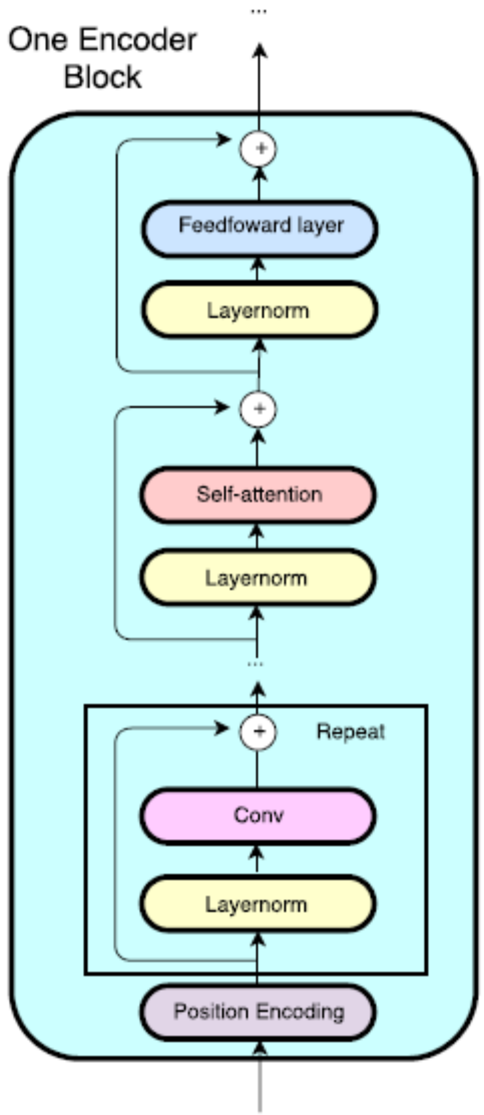
Input Embedding layer

- Word embedding
  - p1 = 300 dimensional pre-trained GloVe
- Character embedding
  - p2 = 200 dimensional trainable vector
  - concatenate all character of each word ->conv & max-pooling

- Output of this layer:

$$[x_w; x_c] \in \mathbf{R}^{p_1+p_2}$$

Embedding Encoder Layer



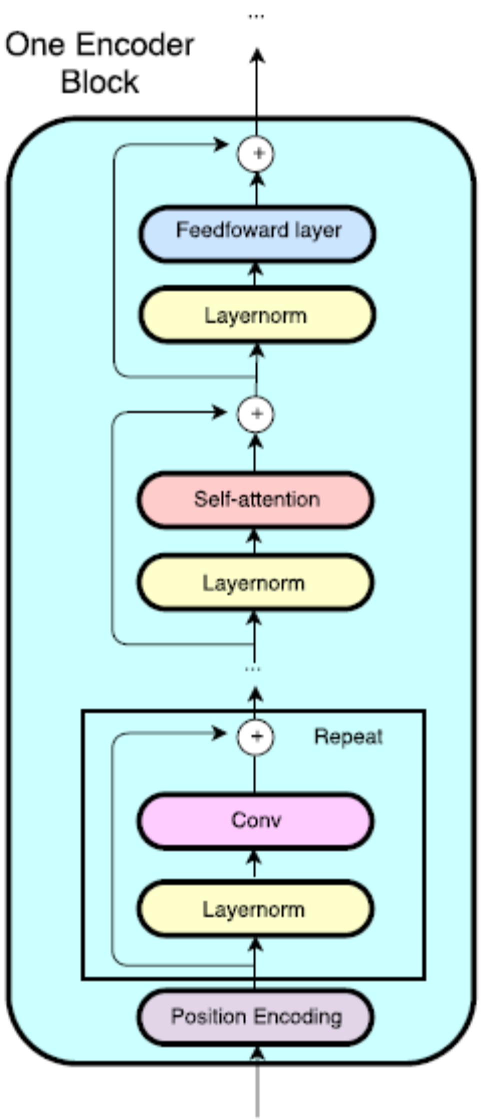
- Position Encoding:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Position embedding have the same dimension of d_{model};
In this paper, $d_{model} = p_1+p_2$

pos is the position
$2i$ & $2i+1$ is the dimension

Embedding Encoder Layer



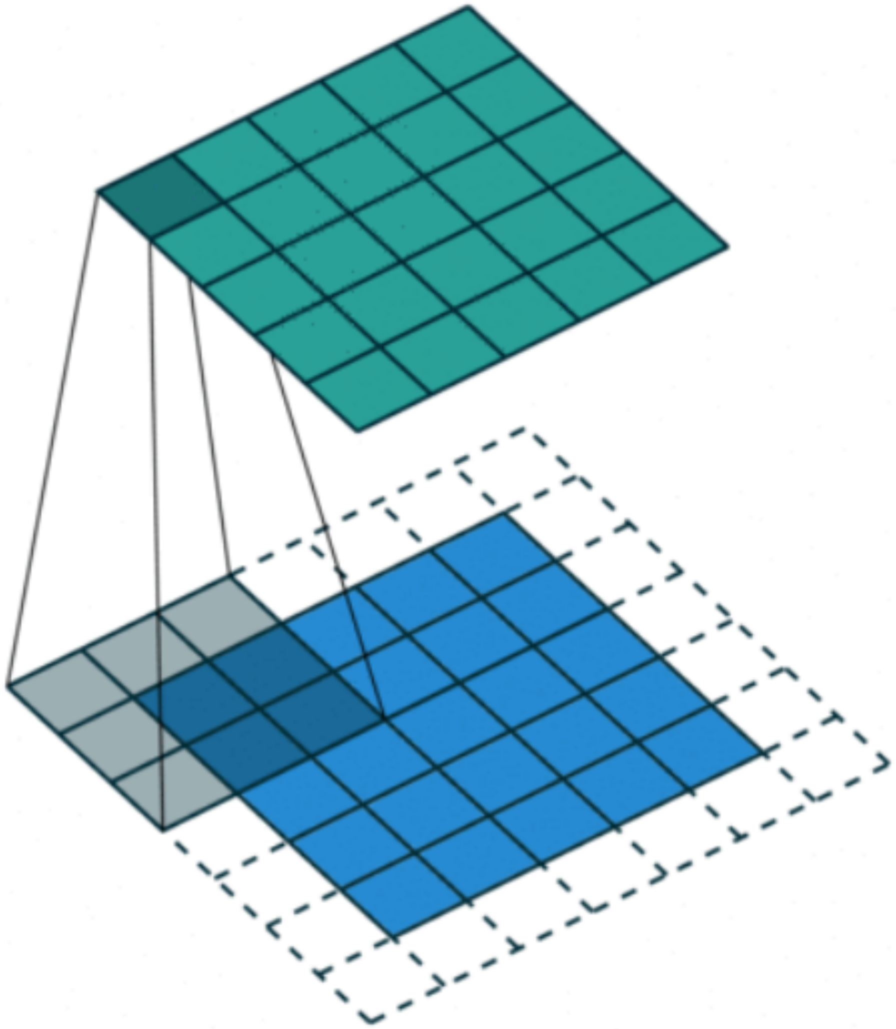
- [convolution-layer × # + self-attention-layer+ feed-forward-layer]
  - **convolution-layer**: depthwise separable convolutions
  - **self-attention-layer**: multi-head attention mechanism

Depthwise separable convolutions

- Memory efficient
- Have better generation

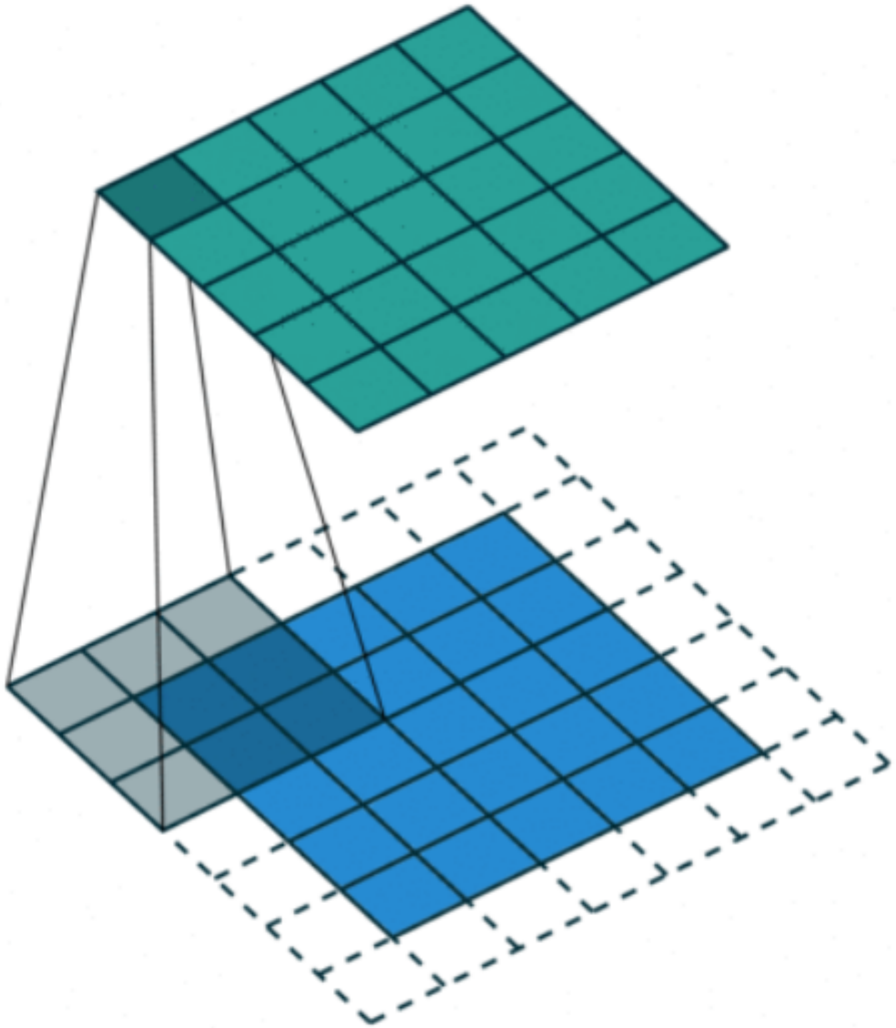
- kernel size = 7
- Filters = 128

Convolutions 参数



- 卷积核大小，Kernel size
- 步长，Stride
- 边界扩充，Padding
- 输入&输出通道数量，Input & Output Channels

Convolutions 处理过程



5×5 的图像

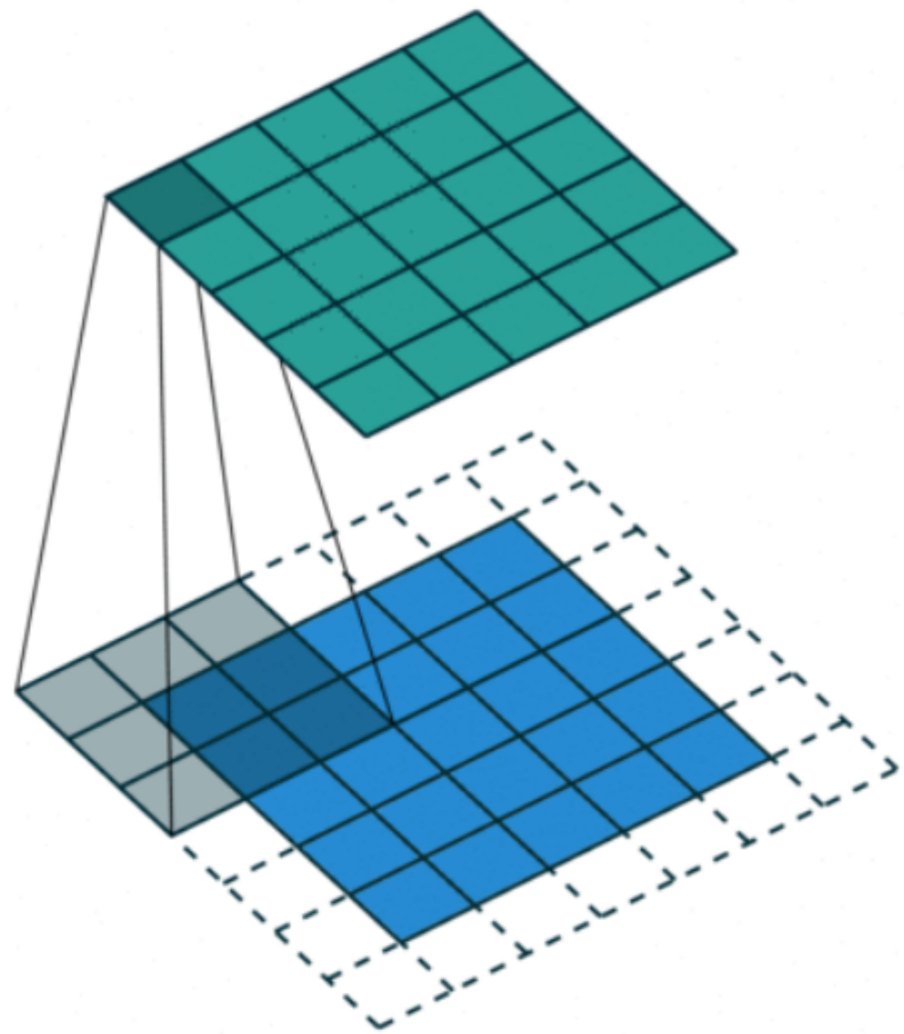
Input channels：16

设置：kernel Size = 3×3

output channels:32

1. 使用大小为3×3的filter对图像的每一个channel进行处理，得到16个feature map；
2. 对每个channel对应的feature map进行融合得到5×5×1；
3. 使用32个filter重复1，2步，最终得到输出5×5×32。

Depthwise separable convolutions处理过程



通道分离+深度卷积

depthwise separable convolution

= depthwise convolution + pointwise convolution

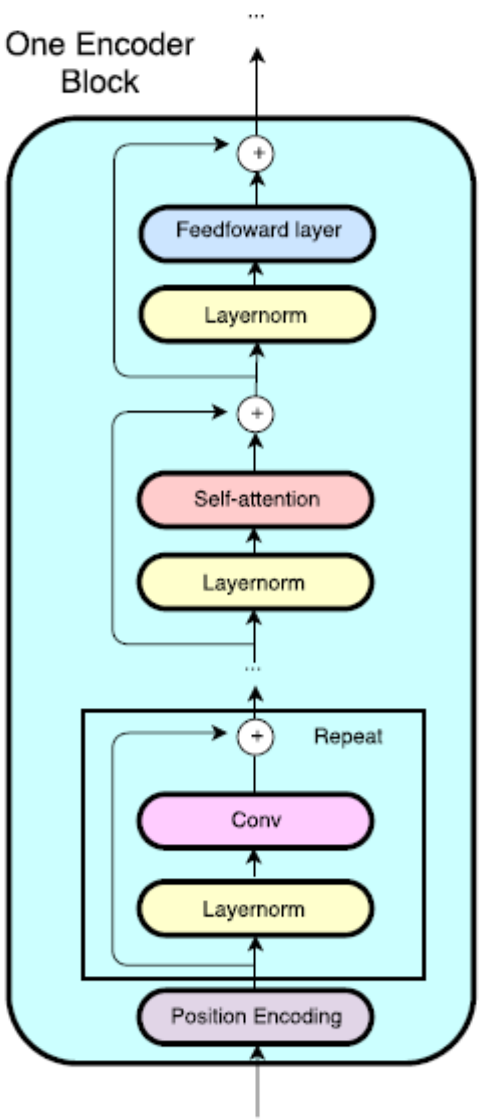
常用深度乘数（depth multiplier）设为1

1. 使用卷积核大小为3×3的filter对图像的每一个channel进行处理，得到16个feature map；
2. 使用32个卷积核大小为1×1的filter对遍历这16个feature map，进行相加融合。
3. 得到输出5×5×32。

Convolutions 与 Depthwise Separable Convolutions 中的参数数量计算

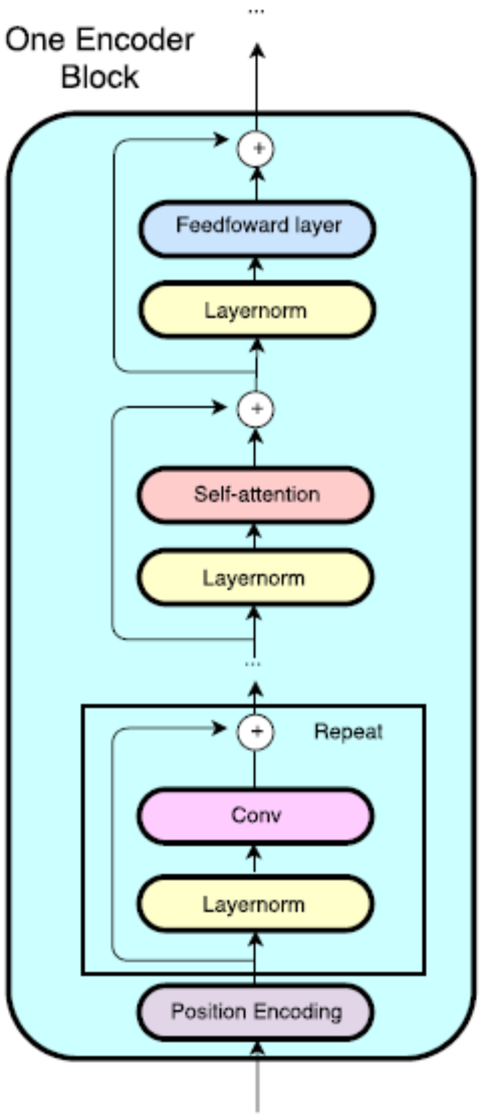
- 使用传统convolutions，需要的参数数量为：
  - $3 \times 3 \times 16 \times 32 = 4680$
- 使用Depthwise Separable Convolutions，需要的参数数量为：
  - $3 \times 3 \times 16 + 32 \times 1 \times 1 \times 16 = 656$
- Memory efficient！

Embedding Encoder Layer



- **convolution-layer**: depthwise separable convolutions
- Input channels: $p_1 + p_2$ =500
- kernel size = 7
- Filters = 128
- Repeat: 4

Embedding Encoder Layer

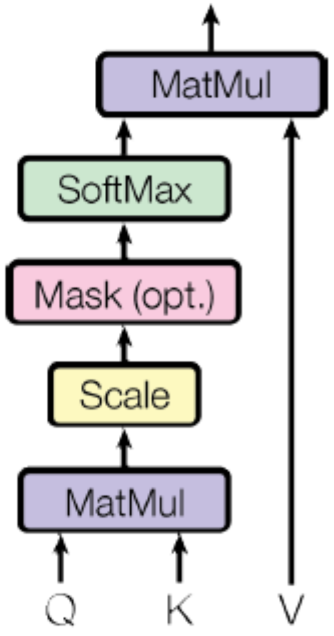


- **self-attention-layer**:

  multi-head attention mechanism

Attention

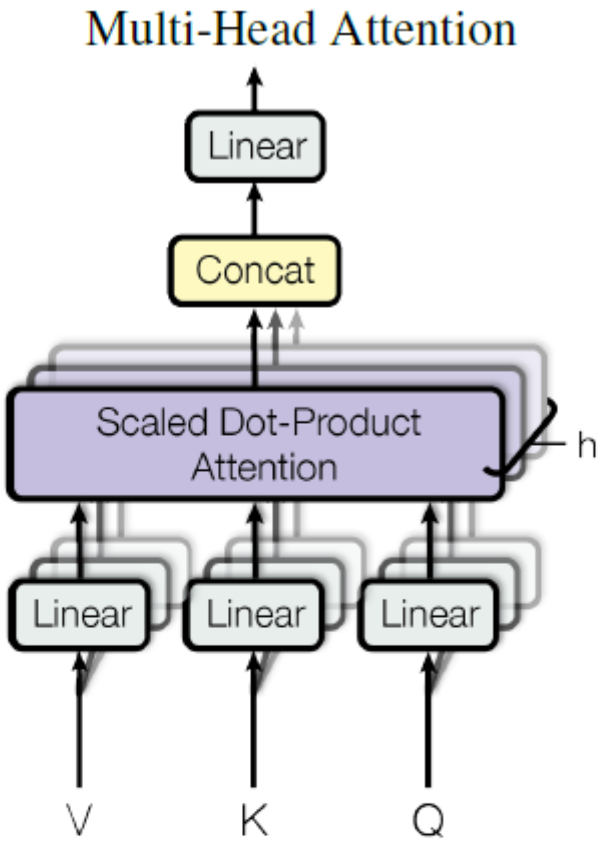
Scaled Dot-Product Attention



An attention function can be described as mapping a query and a set of key-value pairs to an output ;

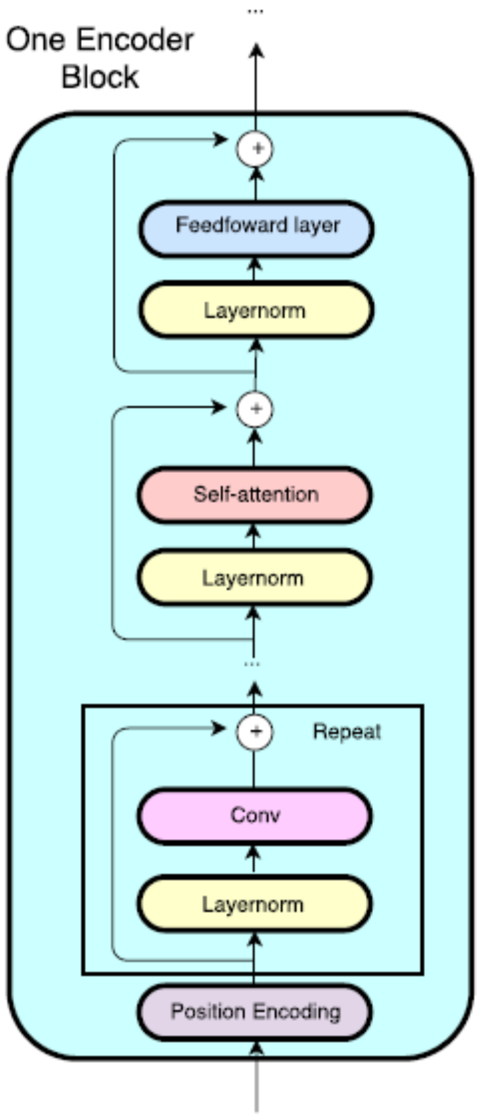
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head Attention



$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where head}_{\mathrm{i}} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Embedding Encoder Layer



Input :

$x \in R^{p_1+p_2}$, ($p_1 + p_2$ = 500)

Conv:

kernel size = 7

number of filters: 128

number of conv layers: 4

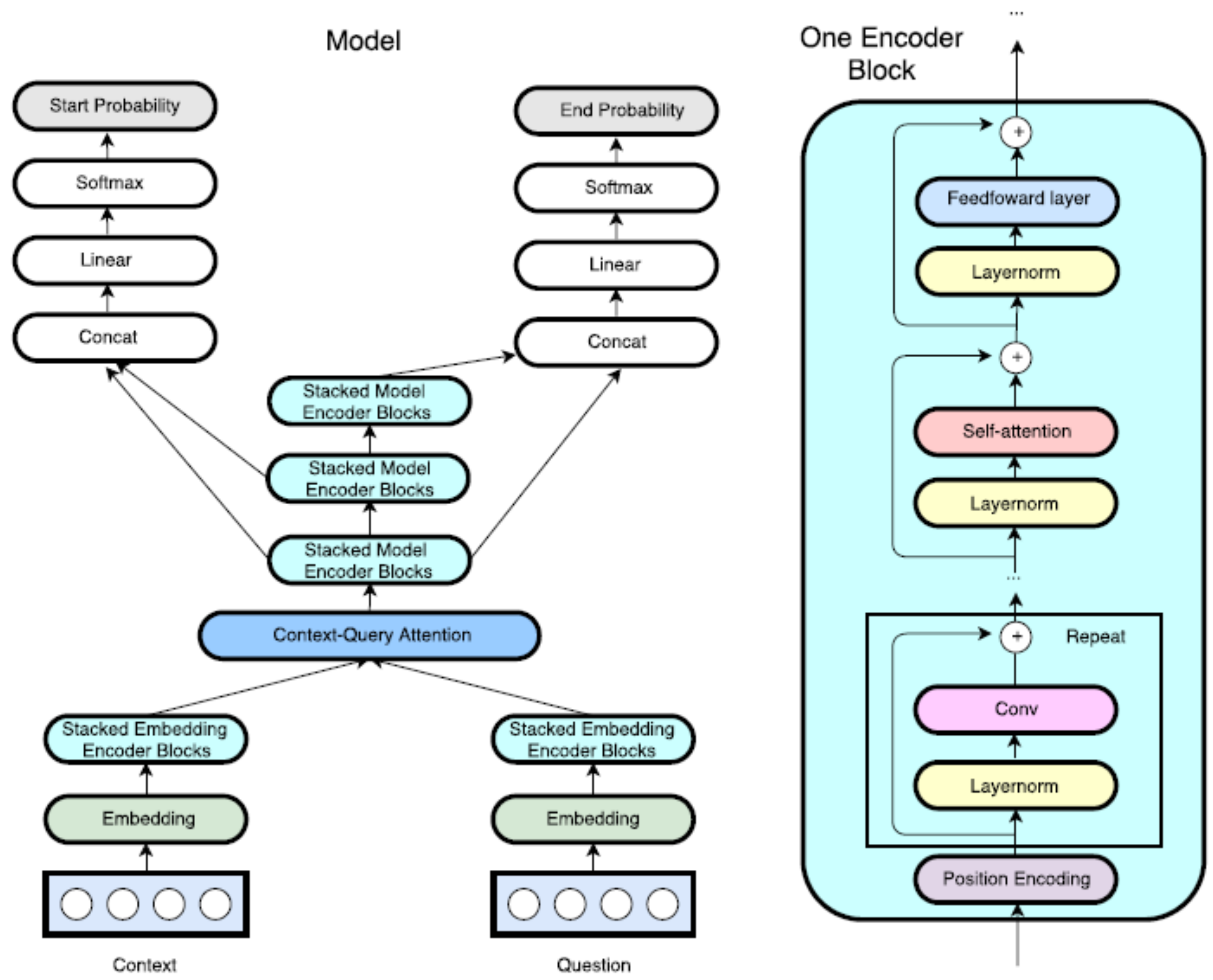
Self-attention:

number of heads : 8

Output:

$x \in R^{128}$

Model



Model
Start Probability
Softmax
Linear
Concat
End Probability
Softmax
Linear
Concat
Stacked Model
Encoder Blocks
Stacked Model
Encoder Blocks
Stacked Model
Encoder Blocks
Context-Query Attention
Stacked Embedding
Encoder Blocks
Stacked Embedding
Encoder Blocks
Embedding
Embedding
Context
Question
One Encoder
Block
Feedfoward layer
Layernorm
Self-attention
Layernorm
Repeat
Conv
Layernorm
Position Encoding

Context-Query Attention Layer

- $\mathbf{C}$: encoded context,

  length = n

- $\mathbf{Q}$: encoded query,

  length = m

- S : similarity matrix $S \in R^{n \times m}$

  similarity function :

$$f(q, c) = W_0[q, c, q \odot c]$$

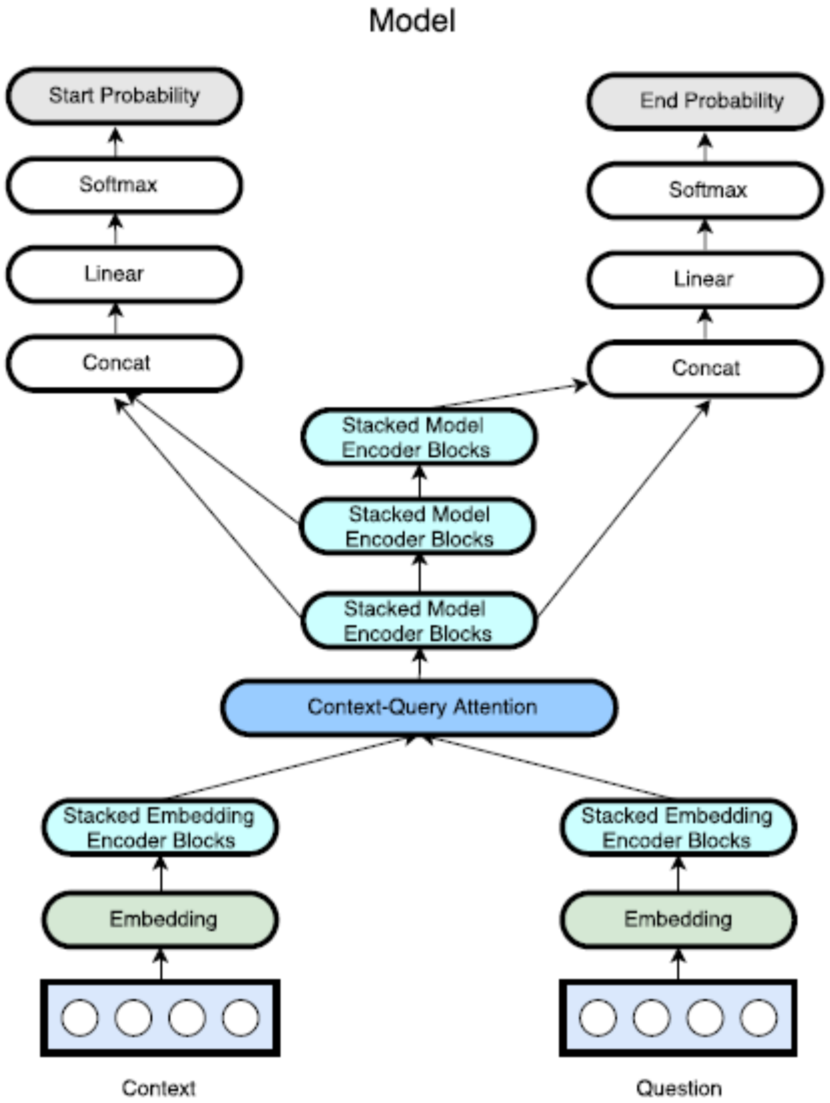
- Compute the row normalized matrix $\bar{S}$ of S
- query-to-context attention:

  A = $\bar{S} \cdot Q^T \in R^{n \times d}$

- Compute the column normalized matrix $\bar{\bar{S}}$ of S
- context-to-query attention:

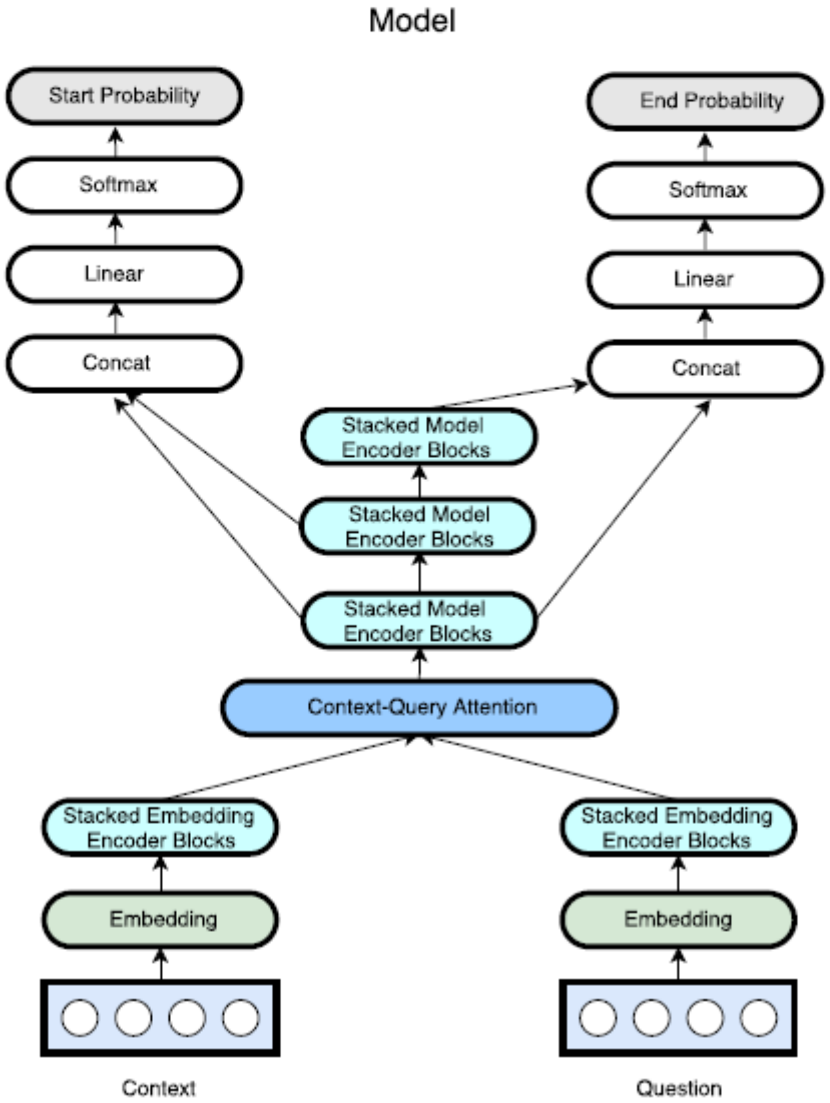
  B = $\bar{S} \cdot \bar{\bar{S}} \cdot C^T \in R^{n \times d}$

Model Encoder Layer



- Q2C attention: A ∈ $R^{n \times d}$
- C2Q attention: B ∈ $R^{n \times d}$
- Input of this layer at each position:
  - [c, a, c ⊙ a, c ⊙ b]

Output Layer

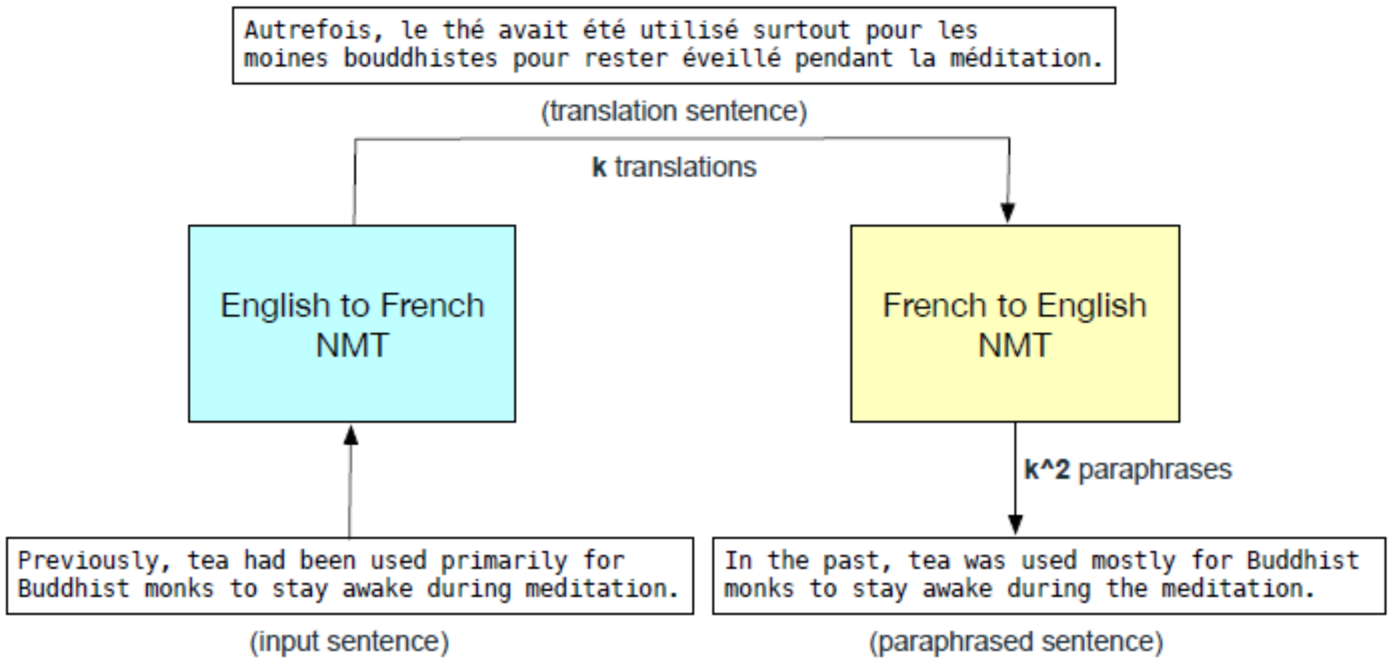


- Predict the probability of each position in the context being the start or end of an answer span.
- Start: $p^1 = \text{softmax}(W_1[M_0; M_1])$
- End: $p^2 = \text{softmax}(W_2[M_0; M_2])$
- Score of a span：
  - Product of its start position and end position probabilities.
- Object function:

$$L(\theta) = -\frac{1}{N}\sum_i^N \left[\log(p^1_{y^1_i}) + \log(p^2_{y^2_i})\right]$$

Data Augmentation By Backtranslation

- Train with more data.
- Idea：
  - Use two translation models: English to French; French to English



Handling SQuAD Documents and Answers

- each example of SQuAD is a triple of (d, q, a)
- Keep q unchanged, generate new triples (d', q, a')
- d' : simply replace each sentence in d with a randomly-selected paraphrase.
- a' ： compute score between each word in s' and the start/end words of a.

	Sentence that contains an answer	Answer
Original	All of the departments in the College of Science offer PhD programs, except for the Department of Pre-Professional Studies.	Department of Pre-Professional Studies
Paraphrase	All departments in the College of Science offer PHD programs with the exception of the Department of Preparatory Studies.	Department of Preparatory Studies

Experiment

- Accuracy:
  - F1: measure the portion of overlap tokens between the answer and groundtruth;
  - EM: exact match, 0/1

	Published[11]	LeaderBoard[12]
Single Model	EM / F1	EM / F1
LR Baseline (Rajpurkar et al., 2016)	40.4 / 51.0	40.4 / 51.0
Dynamic Chunk Reader (Yu et al., 2016)	62.5 / 71.0	62.5 / 71.0
Match-LSTM with Ans-Ptr (Wang & Jiang, 2016)	64.7 / 73.7	64.7 / 73.7
Multi-Perspective Matching (Wang et al., 2016)	65.5 / 75.1	70.4 / 78.8
Dynamic Coattention Networks (Xiong et al., 2016)	66.2 / 75.9	66.2 / 75.9
FastQA (Weissenborn et al., 2017)	68.4 / 77.1	68.4 / 77.1
BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
SEDT (Liu et al., 2017a)	68.1 / 77.5	68.5 / 78.0
RaSoR (Lee et al., 2016)	70.8 / 78.7	69.6 / 77.7
FastQAExt (Weissenborn et al., 2017)	70.8 / 78.9	70.8 / 78.9
ReasoNet (Shen et al., 2017b)	69.1 / 78.9	70.6 / 79.4
Document Reader (Chen et al., 2017)	70.0 / 79.0	70.7 / 79.4
Ruminating Reader (Gong & Bowman, 2017)	70.6 / 79.5	70.6 / 79.5
jNet (Zhang et al., 2017)	70.6 / 79.8	70.6 / 79.8
Conductor-net	N/A	72.6 / 81.4
Interactive AoA Reader (Cui et al., 2017)	N/A	73.6 / 81.9
Reg-RaSoR	N/A	75.8 / 83.3
DCN+	N/A	74.9 / 82.8
AIR-FusionNet	N/A	76.0 / 83.9
R-Net (Wang et al., 2017)	72.3 / 80.7	76.5 /84.3
BiDAF + Self Attention + ELMo	N/A	**77.9/ 85.3**
Reinforced Mnemonic Reader (Hu et al., 2017)	73.2 / 81.8	73.2 / 81.8
Dev set: Our Model	**73.6 / 82.7**	N/A
Dev set: Our Model + data augmentation $\times 2$	**74.5 / 83.2**	N/A
Dev set: Our Model + data augmentation $\times 3$	**75.1 / 83.8**	N/A
Test set: Our Model + data augmentation $\times 3$	**76.2 / 84.6**	76.2 / 84.6

Experiment

- Speed over RNNs
  - Replace each encoder block with a stack of bi-LSTM.
  - Speed = batches/second

	Ours	RNN-1-128	Speedup	RNN-2-128	Speedup	RNN-3-128	Speedup
Training	**3.2**	1.1	**2.9x**	0.34	**9.4x**	0.24	**13.3x**
Inference	**8.1**	2.2	**3.7x**	1.3	**6.2x**	0.92	**8.8x**

Experiment

- Speed over BiDAF model

	Train time to get 77.0 F1 on Dev set	Train speed	Inference speed
Our model	3 hours	102 samples/s	259 samples/s
BiDAF	15 hours	24 samples/s	37samples/s
Speedup	**5.0x**	**4.3x**	**7.0x**

BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
Dev set: Our Model	**73.6 / 82.7**	N/A
Dev set: Our Model + data augmentation ×2	**74.5 / 83.2**	N/A
Dev set: Our Model + data augmentation ×3	**75.1 / 83.8**	N/A
Test set: Our Model + data augmentation ×3	**76.2 / 84.6**	76.2 / 84.6

Abalation study and analysis

	EM / F1	Difference to Base Model EM / F1
Base Model	73.6 / 82.7	
- convolution in encoders	70.8 / 80.0	-2.8 / -2.7
- self-attention in encoders	72.2 / 81.4	-1.4 / -1.3
replace sep convolution with normal convolution	72.9 / 82.0	- 0.7 / -0.7
+ data augmentation ×2 (1:1:0)	74.5 / 83.2	+0.9 / +0.5
+ data augmentation ×3 (1:1:1)	74.8 / 83.4	+1.2 / +0.7
+ data augmentation ×3 (1:2:1)	74.3 / 83.1	+0.7 / +0.4
+ data augmentation ×3 (2:2:1)	74.9 / 83.6	+1.3 / +0.9
+ data augmentation ×3 (2:1:1)	75.0 / 83.6	+1.4 / +0.9
+ data augmentation ×3 (3:1:1)	**75.1 / 83.8**	**+1.5 / +1.1**
+ data augmentation ×3 (4:1:1)	75.0 / 83.6	+1.4 / +0.9
+ data augmentation ×3 (5:1:1)	74.9 / 83.5	+1.3 / +0.8

Table 5: An ablation study of data augmentation and other aspects of our model. The reported results are obtained on the *development set*. For rows containing entry "data augmentation", "$\times N$" means the data is enhanced to N times as large as the original size, while the ratio in the bracket indicates the sampling ratio among the original, English-French-English and English-German-English data during training.

Conclusion

- Core innovation:
  - Remove the recurrent neural networks
  - Parallel computation

- Depthwise separable convolution
- Multi-head Self-attention
- Data Augment technique