MASTER: Multi-Aspect Non-local Network for Scene Text Recognition

Ning Lu[a,1], Wenwen Yu[a,b,1,*], Xianbiao Qi[a], Yihao Chen[a], Ping Gong[b], Rong Xiao[a], Xiang Bai[c]

[a] *Visual Computing Group, Ping An Property and Casualty Insurance Company, Shenzhen, China*
[b] *School of Medical Imaging, Xuzhou Medical University, Xuzhou, China*
[c] *School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China*

Pattern Recognition

Wenwen Yu

04/22/2021

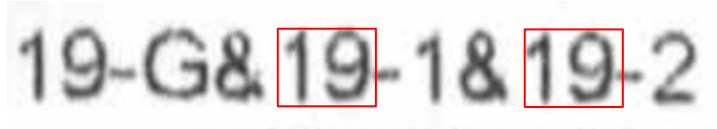
Introduction

- Motivation

  Drawbacks of RNN based encoder-decoder method:

  - Attention drift

  - Poor parallelization, step by step during training and testing



19-G&19-1&19-1&19-2

19-G&19-2

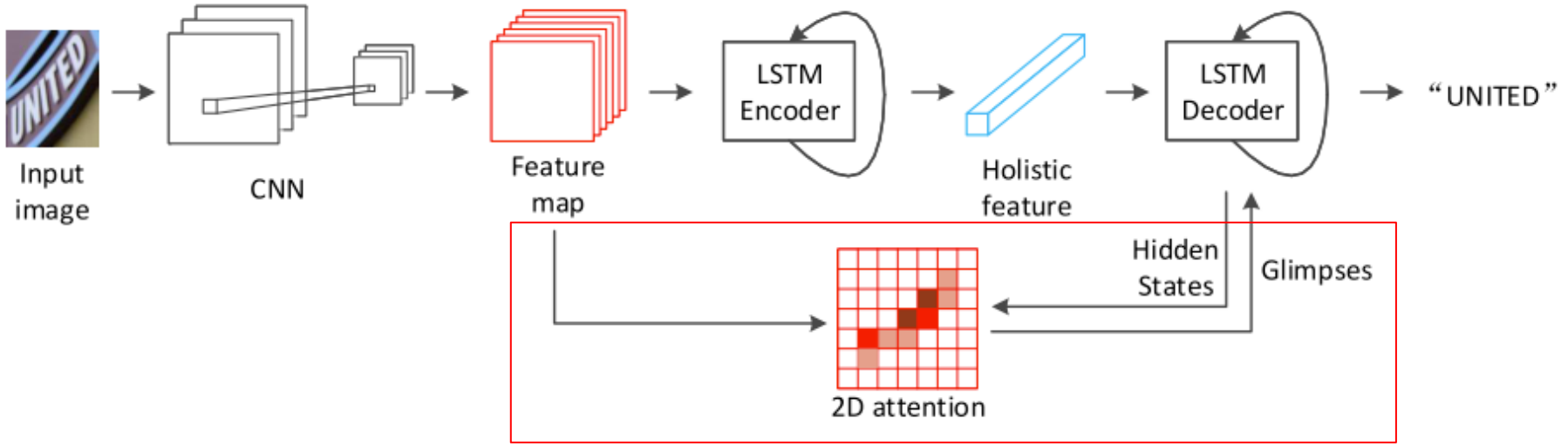
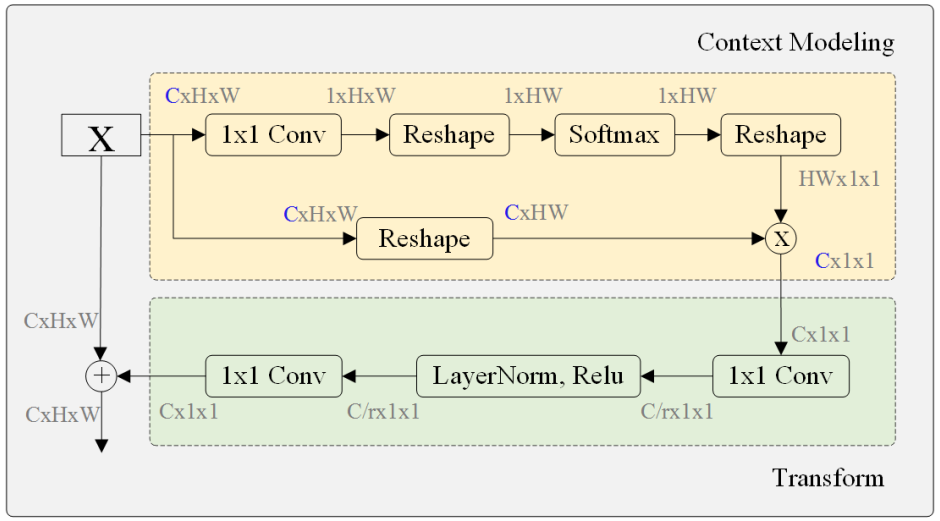
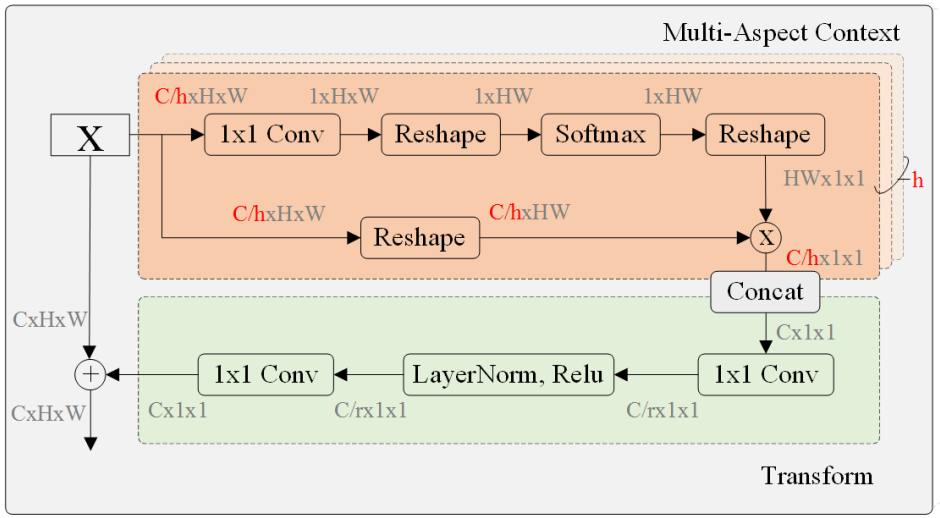


Image Source: Li et al. SAR. AAAI'2019

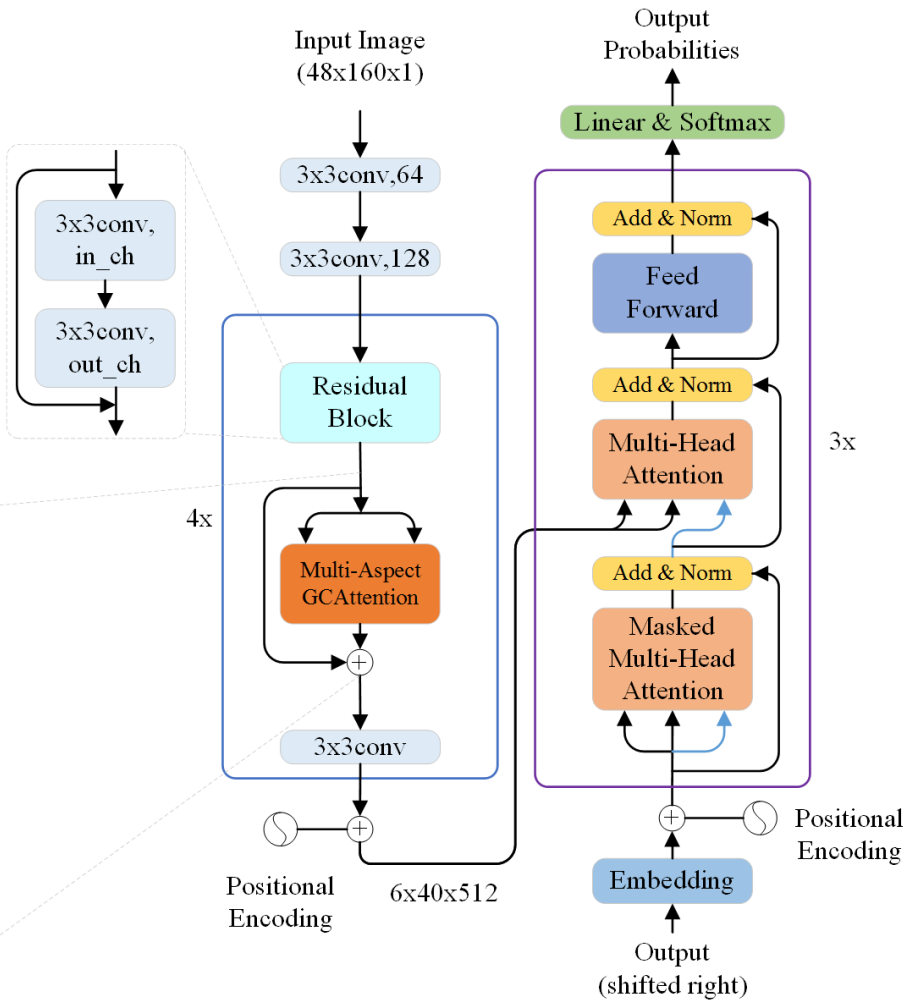
Method



(a) Global Context(GC) block



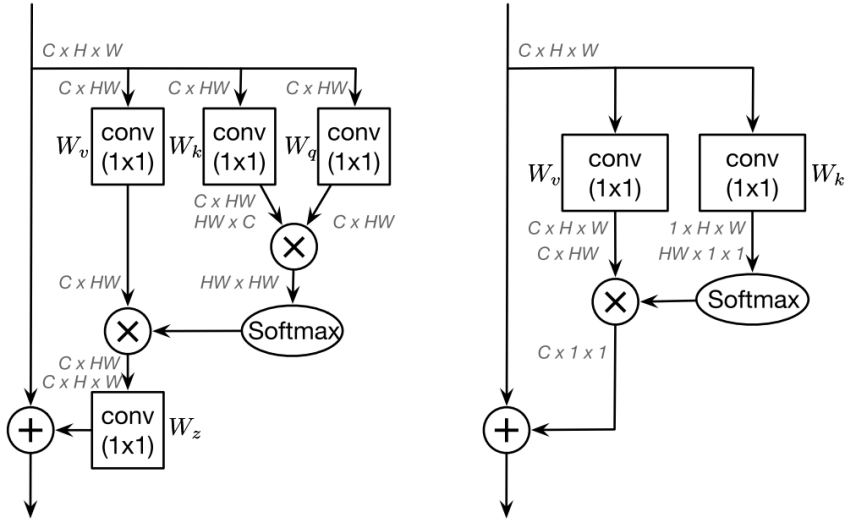
(b) Multi-Aspect GCAttention



(c) The architecture of MASTER model

Method

The non-local block can be regarded as a global context modeling block, which aggregates query-specific global context features



(a) NL block (b) Simplified NL block (Eqn 2)

GC Block[a]:

$$\mathbf{y}_i = \mathbf{x}_i + \mathbf{w}_{v2}\mathrm{ReLU}\left(\mathrm{LN}\left(\mathbf{w}_{v1}\sum_{\forall j}\frac{e^{\mathbf{w}_k\mathbf{x}_j}}{\sum_{\forall m}e^{\mathbf{w}_k\mathbf{x}_m}}\mathbf{x}_j\right)\right)$$

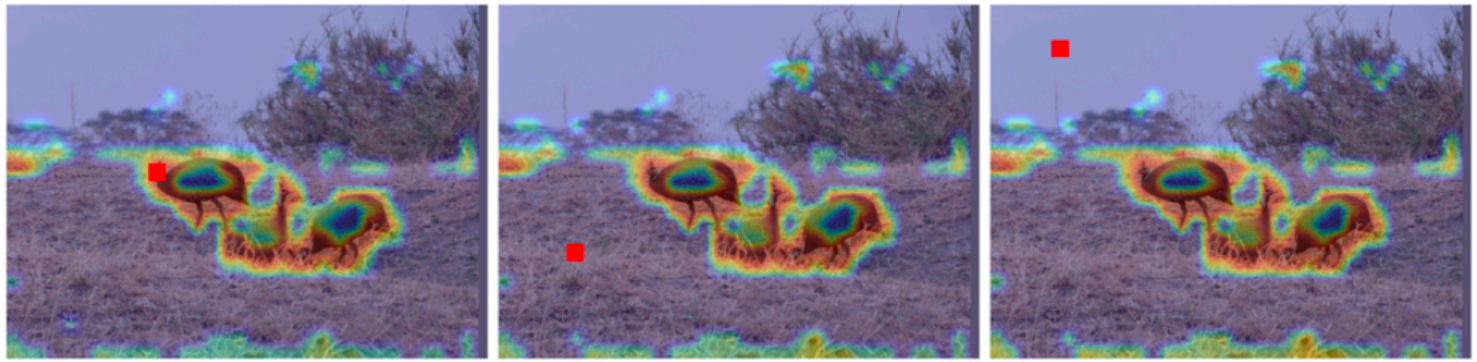
MAGC:

$$\begin{cases} \mathbf{y} = \mathbf{x} + \delta(MAGC(\mathbf{x})), \\ MAGC(\mathbf{x}) = Concat(gc_1, gc_2, \ldots, gc_h), \\ gc_i = \sum_{j=1}^{L}\alpha_j\mathbf{x}_j, \\ \boldsymbol{\alpha} = softmax\left(\frac{\mathbf{w}_k\mathbf{x}_1}{\sqrt{d_h}}, \frac{\mathbf{w}_k\mathbf{x}_2}{\sqrt{d_h}}, \ldots, \frac{\mathbf{w}_k\mathbf{x}_L}{\sqrt{d_h}}\right) \end{cases}$$

Self-attention:

$$\begin{cases} Atten(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{t'}]^T \in \mathcal{R}^{t'\times d}, \\ \mathbf{a}_i = Atten(\mathbf{q_i},\mathbf{K},\mathbf{V}), \\ Atten(\mathbf{q_i},\mathbf{K},\mathbf{V}) = \sum_{j=1}^{t}\alpha_j\mathbf{v}_j^T \in \mathcal{R}^d, \\ \boldsymbol{\alpha} = softmax\left(\frac{\langle\mathbf{q_i},\mathbf{k}_1^T\rangle}{\sqrt{d}}, \frac{\langle\mathbf{q_i},\mathbf{k}_2^T\rangle}{\sqrt{d}}, \ldots, \frac{\langle\mathbf{q_i},\mathbf{k}_t^T\rangle}{\sqrt{d}}\right) \end{cases}$$

Loss: Cross-entropy

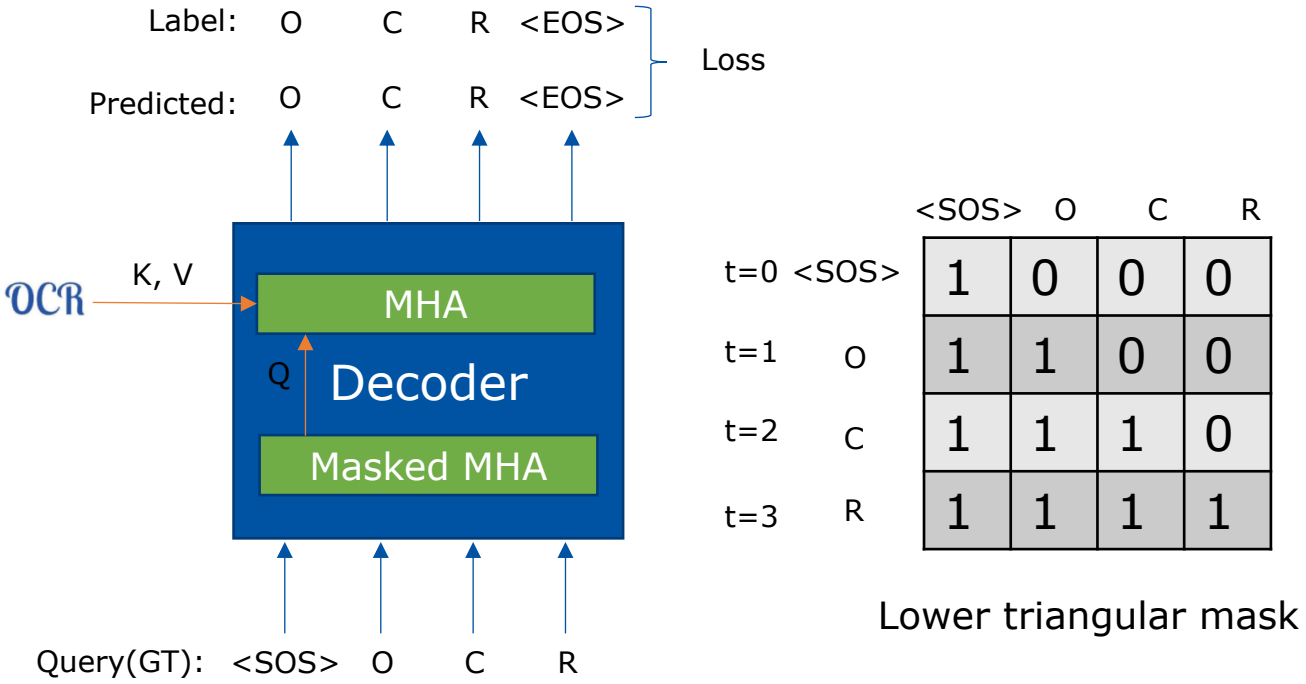


[a] Cao et al. GCNet: non-local networks meet squeeze-excitation networks and beyond. ICCV, 2019
[b] Wang et al. Non-local neural networks. CVPR, 2018

Method

Training stage: highly parallel vs RNN
- Inter-token interact parallelly by self-attention
- Batch mechanism

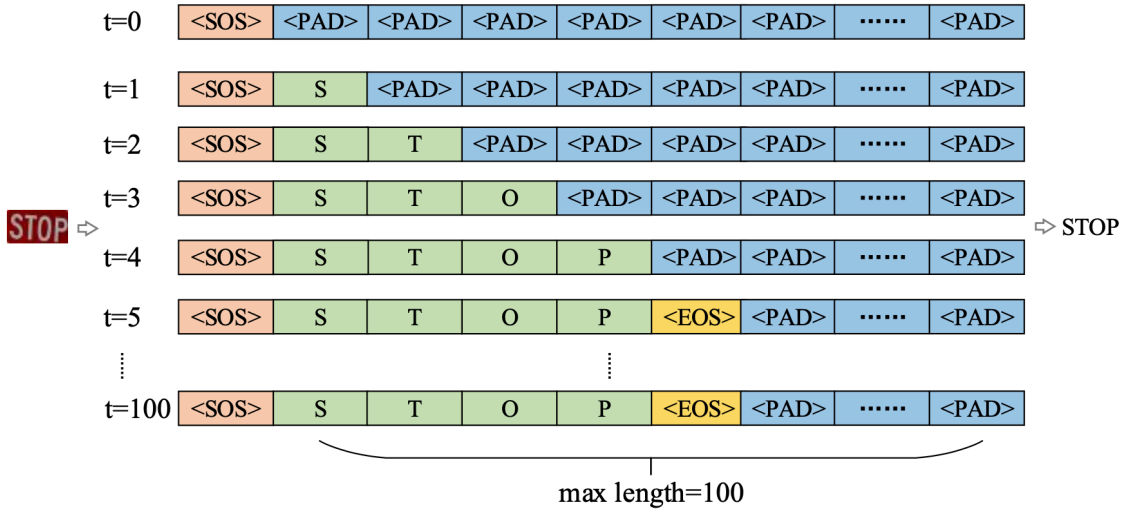
Inference stage: autoregressive progress, step by step decoding
- Memory-cache based inference



Training stage

	<SOS>	O	C	R
t=0	1	0	0	0
t=1	1	1	0	0
t=2	1	1	1	0
t=3	1	1	1	1

Lower triangular mask

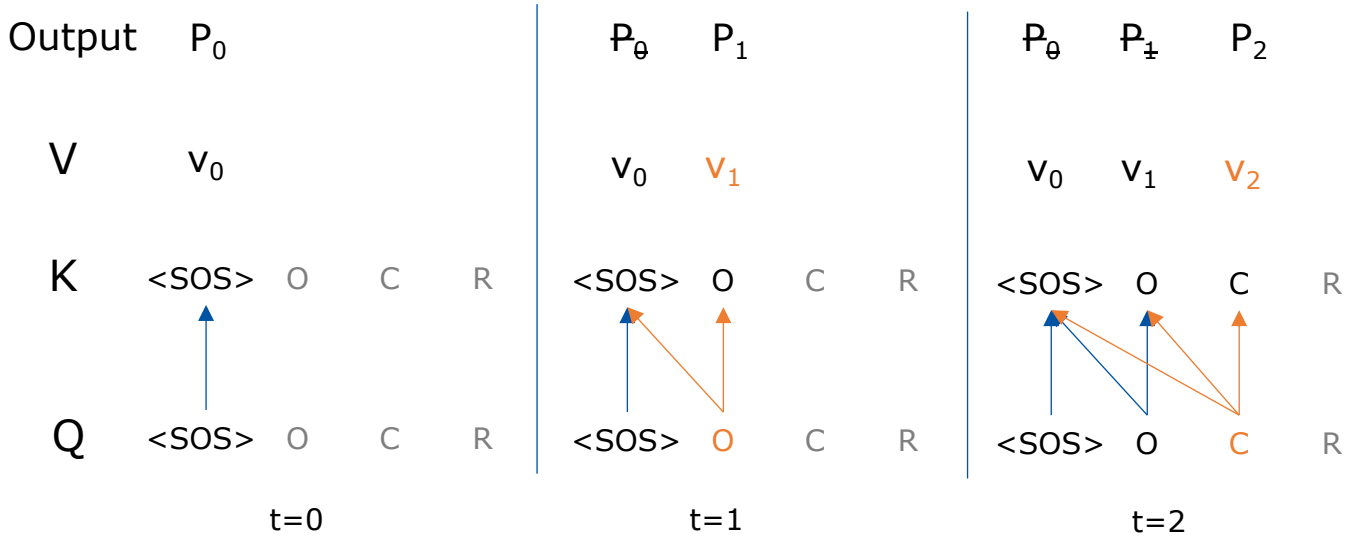


Inference stage

Method

Memory-cache based inference

- Cache intermediate results in Line 2, 11-12
- Remove unnecessary computation in Lines 13–14
- Q is always a 1D vector instead of a 2D matrix



Algorithm 1: Memory-cache based inference. B is the number of blocks. F is the addition of the CNN feature and the position embedding feature. T is the max decoder length. M and W are the parameters of the masked multi-head and multi-head attention. X_k^b, X_v^b, *keys_memory*, *values_memory* are the cached variables.

Input : CNN feature: F
Output : *outputs*

1 **for** b *in* $range(B)$ **do**
2 $X_k^b, X_v^b = W_k^b * F, W_v^b * F$;
3 $keys\_memory[b], values\_memory[b] = [\ ], [\ ]$;
4 **end**
5 $t \leftarrow 0$;
6 $outputs = [\ ]$;
7 $p_t \leftarrow$<SOS>;
8 **while** $p_t \neq$<EOS> *and* $t \leq T$ **do**
9 q = `Embedding` (p_t) + `PositionEmbedding` (t);
10 **for** b *in* $range(B)$ **do**
11 $keys\_memory[b].append(M_k^b * q)$;
12 $values\_memory[b].append(M_v^b * q)$;
13 $q \leftarrow$ `MaskedMHA`$(M_q^b * q, keys\_memory[b], values\_memory[b])$;
14 $q \leftarrow$ `MHA`$(W_q^b * q, X_k^b, X_v^b)$;
15 $q \leftarrow$ `FeedForward`(q);
16 **end**
17 $t \leftarrow t + 1$;
18 $p_t \leftarrow$ `Argmax(LinearSoftmax`(q)`)`;
19 $outputs.append(p_t)$
20 **end**

Results

Table 2
Performance of our model and other state-of-the-art methods on public datasets. All values are reported as a percentage (%). "None" means no lexicon. * indicates using both word-level and character-level annotations to train the model. ** denotes the performance of SAR trained only on the synthetic text datasets. In each column, the best performance result is shown in **bold** font, and the second-best result is shown with an underline. Our model achieves competitive performance on most of the public datasets, and the distance between us and the first place [50] is very small on IIIT5k and SVT datasets.

Method	IIIT5K	SVT	IC03	IC13	IC15	SVTP	CUTE
	None	None	None	None	None	None	None
Jaderberg et al. [34]	–	80.7	93.1	90.8	–	–	–
Shi et al. [33]	81.9	81.9	90.1	88.6	–	71.8	59.2
STAR-Net [51]	83.3	83.6	–	89.1	–	73.5	–
Wang and Hu [52]	80.8	81.5	–	–	–	–	–
CRNN [5]	81.2	82.7	91.9	89.6	–	–	–
Focusing Attention [31]*	87.4	85.9	94.2	93.3	70.6	–	–
SqueezedText [53]*	87.0	–	–	92.9	–	–	–
Char-Net [54]*	92.0	85.5	–	91.1	74.2	78.9	–
Edit Probability [6]*	88.3	87.5	94.6	94.4	73.9	–	–
ASTER [7]	93.4	89.5	94.5	91.8	76.1	78.5	79.5
NRTR [37]	86.5	88.3	<u>95.4</u>	<u>94.7</u>	–	–	–
SAR** [8]	91.5	84.5	–	91.0	69.2	76.4	83.3
ESIR [35]	93.3	90.2	–	91.3	76.9	79.6	83.3
MORAN [36]	91.2	88.3	95.0	92.4	68.8	76.1	77.4
Wang et al. [39]	93.3	88.1	–	91.3	74.0	80.2	85.1
Mask TextSpotter [50]*	**95.3**	**91.8**	95.2	**95.3**	<u>78.2</u>	<u>83.6</u>	**88.5**
MASTER (Ours)	<u>95.0</u>	<u>90.6</u>	**96.4**	**95.3**	**79.4**	**84.5**	<u>87.5</u>

Results

Table 3
Leaderboard of various methods on the online COCO-Text test server. In each column, **Bold** represent the best performance.

Method	Case sensitive		Case insensitive	
	Total Edit Distance	Correctly Recognised Words (%)	Total Edit Distance	Correctly Recognised Words (%)
SogouMM	3,496.3121	44.64	**1,037.2197**	**77.97**
SenseTime-CKD	4,054.8236	41.52	824.6449	77.22
HIK_OCR	3,661.5785	41.72	899.1009	76.11
Tencent-DPPR Team	4,022.1224	36.91	1,233.4609	70.83
CLOVA-AI [56]	3,594.4842	47.35	1,583.7724	69.27
SAR [8]	4,002.3563	41.27	1,528.7396	66.85
HKU-VisionLab [54]	3,921.9388	40.17	1,903.3725	59.29
MASTER (single model)	3,527.3165	45.96	1,528.7526	67.41
MASTER (Ours)	**3,272.0810**	**49.09**	1,203.4201	71.33

Results

Table 4
Under different parameter settings our model recognition accuracy: h, N denotes the numbers of Multi-Aspect Context in the encoder and identical layers in the decoder, respectively. Standard Setting uses $h = 8$ and $N = 3$. When h or N changes, all other parameters keep the same as the Standard Setting. All values are reported as a percentage (%).

Methods	IIIT5k	SVT	CUTE	IC03	IC13	IC15	SVTP
Standard setting: $h = 8$, $N = 3$	95.0	90.6	87.5	96.4	95.3	79.4	84.5
$h = 0$	94.6	90.1	86.2	95.9	95.0	78.4	82.3
$h = 1$	94.9	**91.5**	87.6	**96.9**	**95.7**	79.4	83.8
$h = 2$	94.93	90.7	**88.54**	96.6	95.4	79.5	84.0
$h = 4$	94.7	90.9	86.8	96.1	95.1	**79.6**	83.7
$h = 16$	**95.1**	91.3	85.4	96.0	95.3	79.4	84.1
$N = 1$	94.3	90.4	85.4	95.3	94.1	78.9	83.1
$N = 6$	91.3	87.4	76.7	94.3	91.6	72.9	75.7

Table 5
Speed comparison between MASTER (Ours) and SAR. MASTER is faster and more accurate than the SAR method. All timing information is on an NVIDIA Tesla V100 GPU.

Method	Input	Accuracy	Inference time (ms)	Training time (h)
SAR [8]	48 × 160	91.5	16.1	51
MASTER (original)	48 × 160	95.0	9.2	36
MASTER (improved)	48 × 160	95.0	4.3	36

Results

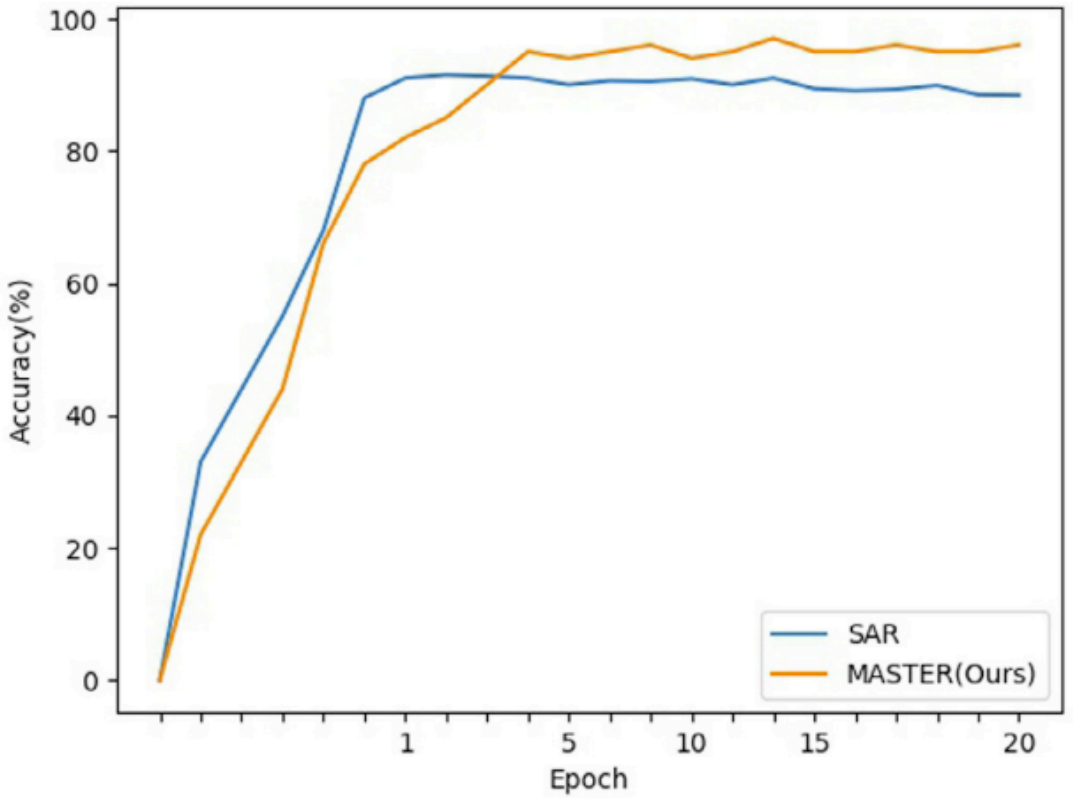


Fig. 4. The model stability comparison between MASTER (Ours) and SAR [8].

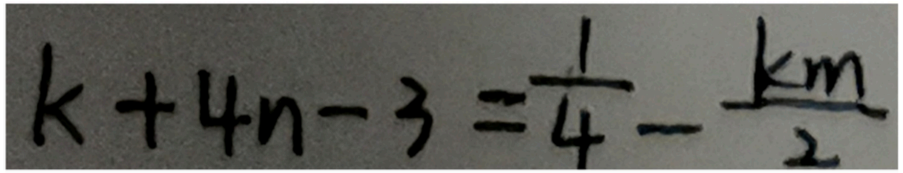
Input Images	Ours	By SAR [8]	GT
	ANDA	AMDA	ANDA
	GOOD	GCOD	GOOD
	wacom	waccom	wacom
	BONNIE	BONIE	BONNIE
	SERV	LEAD	SERV
	actaea	actara	actaea
	FOOTBALL	FOOTBAL	FOOTBALL
	Timms	Timmms	Timms

Fig. 3. Samples of recognition results of our MASTER and SAR. Green characters mean correct predictions and red characters mean wrong predictions.

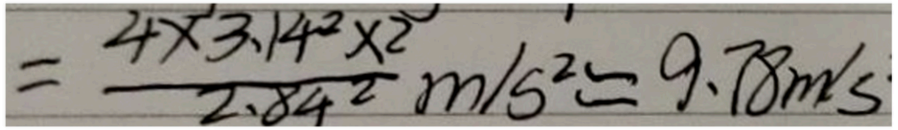
■Results

Handwritten Mathematical Expression Recognition

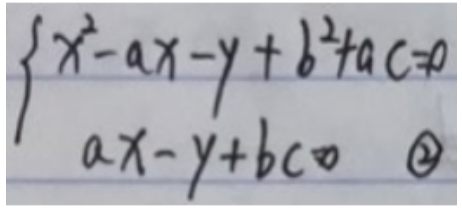
- 2nd place
  - model ensemble
  - multi-scale
- 5th place
  - single model
  - single scale

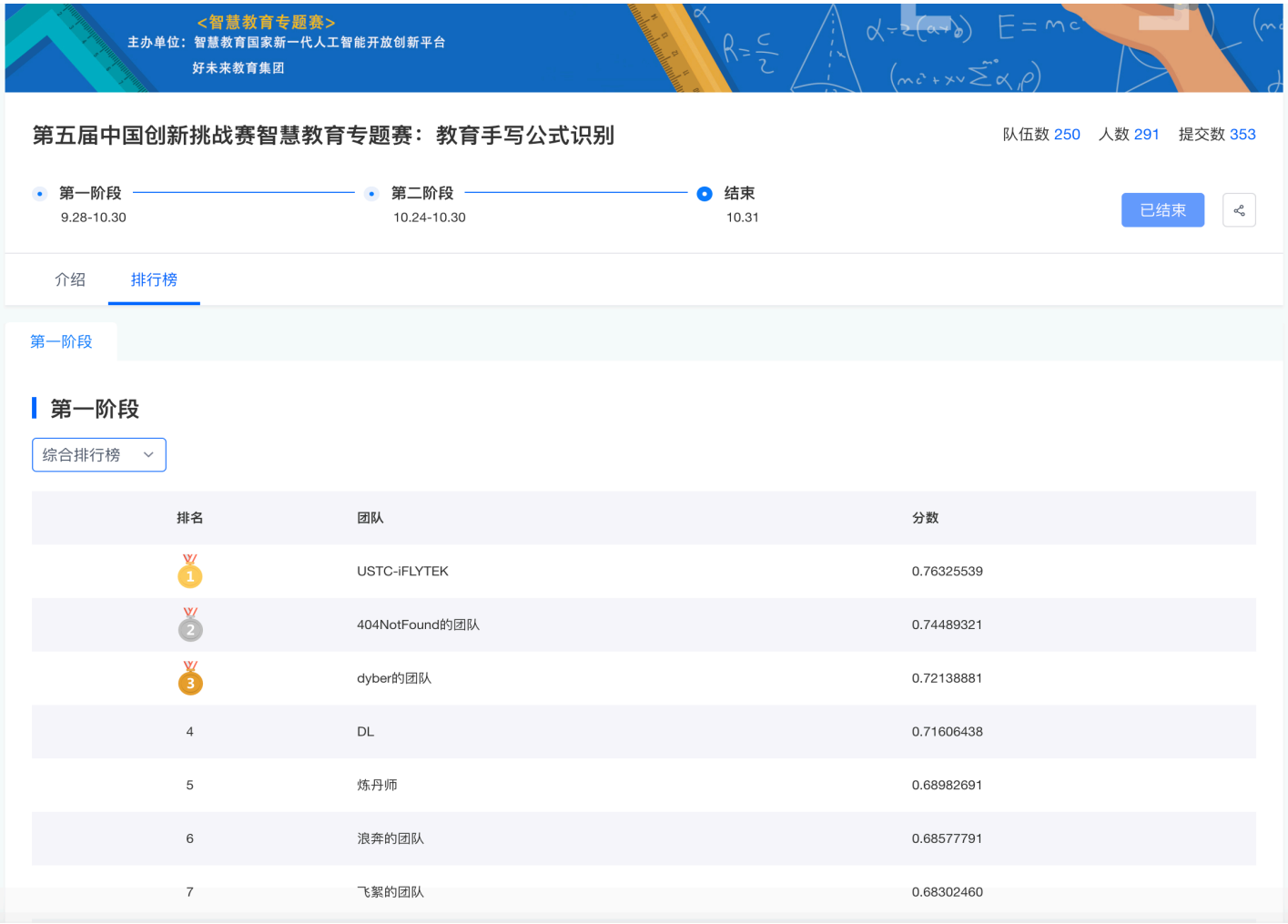


k+4n-3=\frac{1}{4}-\frac{km}{2}



=\frac{4\times3.14^{2}\times2}{2.84^{2}}m/s^{2}=9.78m/s





Learnt Lessons

- Self-attention easily cause overfiting and hard to adapt
  - Dropout
  - Label Smothing
  - **Task-specific data** and GPU are all you need
- Deal with more complex tasks
  - e.g., blurred background, multi-line, handwritten, vertical text, curved text, multilingual
- Extreme cases can’t be handled
  - e.g., long repeated char
- MASTER is the Master OCR
  - Theoretically, if it has enough data and GPU, it can handle all of seq2seq tasks

Q&A