Appendix: Scene Text Recognition with Permuted Autoregressive Sequence Models

Darwin Bautista and Rowel Atienza

Electrical and Electronics Engineering Institute
University of the Philippines, Diliman
{darwin.bautista,rowel}@eee.upd.edu.ph

A Issues with unidirectionality of AR models in STR

As discussed in the main text, the unidirectionality of AR models could result in spurious addition of suffixes and direction-dependent decoding. Shown in Table 1 is a sample output of a *left-to-right* (LTR) AR model trained on a 36-character lowercase charset. Since the input is fairly clear and horizontal, the model was very confident in the predictions for the first 10 characters. However, since it was trained on alphanumeric characters only, it did not know how to recognize the exclamation mark. The language context *swayed* the output of the model to add the *-ly* suffix in order to make sense of the unrecognized character. A *right-to-left* (RTL) AR model would not add the suffix due to the lack of context (since the right-most characters would have to be predicted first). This direction-dependent decoding is further illustrated in Table 2 where two AR models trained on opposing directions produce different outputs. In this case, the input contains ambiguity on the uppercase *N* character. If read from left to right, the context of the earlier characters can be used to infer that the ambiguous character is *N*. However, when read in the opposite direction, the context of *OPE* is not yet available, prompting the RTL model to recognize two *l*'s in place of a single *N* character.

Table 1. Example of a spurious suffix from a left-to-right AR model. *GT* refers to the ground truth label, while *Confidence* pertains to per-character prediction confidence

Input	GT	Prediction	Confidence
	terrifying	terrifyingly	[1.00, . . . , 1.00, 0.97, 0.72]

B Inefficiency of External Language Models in STR

As mentioned in the main text, ensemble methods such as ABINet [13] and SRN [38] utilize a standalone or external Language Model (LM). In Table 3, we show

Table 2. Example of direction-dependent decoding with two AR models

Input	GT	Direction	Prediction	Confidence
	open	LTR	open	[1.00, 1.00, 1.00, 0.66]
		RTL	opell	[1.00, 1.00, 0.52, 0.57, 0.94]

the cost measurements of `fvcore` on the full ABINet model for a single input, as well as the measurement breakdown for its component models. We can see that while the LM accounts for around 34.48% of the parameter count, it only uses 13.65% of the overall FLOPS and 15.78% of the overall activations (a measure shown to be correlated with model runtime [11,35]). When evaluated in spelling correction on the 36-character set, the LM achieves a top-5 word accuracy of only 41.9% [13]. With the ground truth label itself as input (Table 4), the same model gets a top-1 word accuracy of only 50.44% (36-char). This means that even if the Vision Model (VM) is perfect (always predicting the correct label), the LM will produce a wrong output 50% of the time. In summary, the external LM's dedicated compute cost, underutilization relative to its parameter and memory requirements, and dismal word accuracy show the inefficiency of this approach. For STR, an internal LM might be more appropriate since the primary input signal is the image, not the language context.

Table 3. Commonly used cost indicators as measured by `fvcore` for ABINet. *Full Model* pertains to the overall measurements

Module	# of Parameters (M)	FLOPS (G)	# of Activations (M)
Full Model	36.858 *(100.00%)*	7.289 *(100.00%)*	10.785 *(100.00%)*
- Vision	23.577 *(63.97%)*	6.249 *(85.73%)*	9.036 *(83.78%)*
- Language	12.707 *(34.48%)*	0.995 *(13.65%)*	1.702 *(15.78%)*
- Alignment	0.574 *(1.55%)*	0.045 *(0.62%)*	0.047 *(0.44%)*

Table 4. Performance of ABINet's LM when the ground truth label itself is used as the input. NED refers to the Normalized Edit Distance [28]

Dataset	# of samples	Word acc. (%)	1 - NED
IIIT5k	3,000	47.33	69.50
SVT	647	65.38	83.48
IC13	1,015	62.07	78.77
IC15	2,077	40.49	67.72
SVTP	645	65.27	83.08
CUTE80	288	46.88	68.65
Combined	**7,672**	**50.44**	**72.54**

C Multi-head Attention

The attention mechanism is central to the operation of Transformers [30]. In scaled dot-product attention, the similarity scores between two d_k-dimensional vectors $\mathbf{q}$ (query) and $\mathbf{k}$ (key), computed using their dot-product, are used to transform a d_v-dimensional vector $\mathbf{v}$ (value). Formally, scaled dot-product attention is defined as:

$$Attn(\mathbf{q}, \mathbf{k}, \mathbf{v}) = softmax\left(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d_k}}\right)\mathbf{v} \tag{1}$$

It accepts an optional *attention mask* that limits which *keys* the *queries* could attend to. In a Transformer with token dimensionality of d_{model}, $d_k = d_v = d_{model}$.

Multi-head Attention (MHA) is the extension of scaled dot-product attention to multiple representation subspaces or *heads*. To keep the computational cost of MHA practically constant regardless of the number of heads, the dimensionality of the vectors are reduced to $d_{head} = d_{model}/h$, where h is the number of heads. A *head* corresponds to an invocation of Equation (1) on projected versions of $\mathbf{q}$, $\mathbf{k}$, and $\mathbf{v}$ using parameter matrices $\mathbf{W}^q \in \mathbb{R}^{d_{model} \times d_{head}}$, $\mathbf{W}^k \in \mathbb{R}^{d_{model} \times d_{head}}$, and $\mathbf{W}^v \in \mathbb{R}^{d_{model} \times d_{head}}$, respectively, as shown in Equation (2). The final output is obtained in Equation (3) by concatenating the heads and multiplying by the output projection matrix $\mathbf{W}^o \in \mathbb{R}^{d_{model} \times d_{model}}$.

$$head_i = Attn(\mathbf{q}\mathbf{W}_i^q, \mathbf{k}\mathbf{W}_i^k, \mathbf{v}\mathbf{W}_i^v) \tag{2}$$

$$MHA(\mathbf{q}, \mathbf{k}, \mathbf{v}) = Concat(head_1, ..., head_h)\mathbf{W}^o \tag{3}$$

D Model Architecture

PARSeq uses an encoder which largely follows the original ViT [12], and a pre-$LayerNorm$ [3,33] decoder with more heads. The architectures are practically unchanged but are reproduced here for the convenience of the reader.

D.1 ViT Encoder

The encoder is composed of 12 layers. All layers share the same architecture shown in Figure 1. The output of the last encoder layer goes through a final $LayerNorm$.

D.2 Visio-lingual Decoder

The decoder (Figure 2) consists of only a single layer. The immediate outputs of all MHA and MLP layers go through *Dropout* ($p = 0.1$, not shown). *Image Features* are already $LayerNorm$'d by the encoder (hence no $LayerNorm$ prior to input).

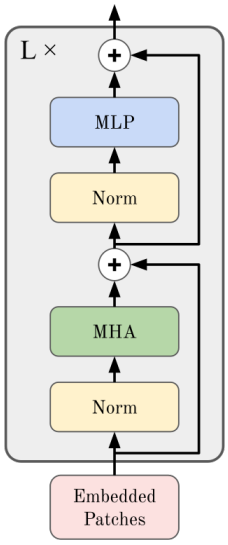


Fig. 1. Illustration of a ViT layer from Dosovitskiy *et al.* [12]. *Norm* pertains to $LayerNorm$

D.3 Architecture Configuration

The main results are obtained from the base model, PARSeq-S, which has a similar configuration to DeiT-S [29] but uses an image size of 128×32 and a patch size of 8×4 (a change also adapted in our reproduction of ViTSTR-S). Based on our experiments, scaling up the model only marginally improves word accuracy on the benchmark. We instead explore scaling down the model to make it more suitable for edge devices. PARSeq-Ti, which uses a configuration similar to DeiT-Ti [29], is more similar to CRNN [26] in terms of parameter count and FLOPS. The detailed configuration parameters are shown in Table 5.

Table 5. Configurations for the base (PARSeq-S) and smaller (PARSeq-Ti) model variants. d_{model} refers to the *dimensionality* of the model which dictates the dimensions of the vectors and feature maps. h refers to the number of *attention heads* used in MHA layers. d_{MLP} refers to the dimension of the intermediate features within the MLP layer. *depth* refers to the number of encoder or decoder layers used

		encoder			*decoder*		
Variants	d_{model}	h	d_{MLP}	depth	h	d_{MLP}	depth
PARSeq-Ti	192	3	768	12	6	768	1
PARSeq-S	384	6	1536	12	12	1536	1

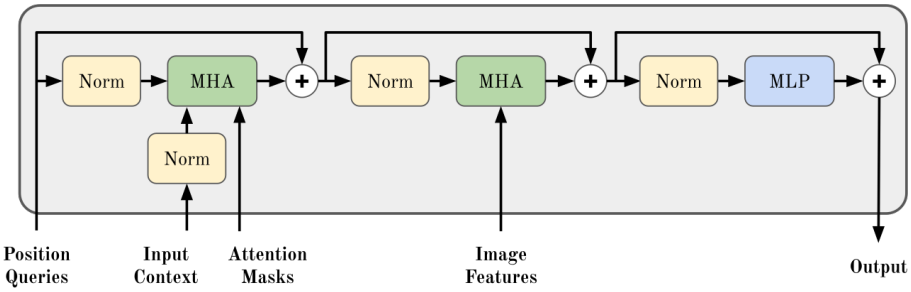


Fig. 2. Visio-lingual decoder architecture with *LayerNorm* layers shown

E Permutation Language Modeling

In this section, we provide additional details about the adaptation of PLM for use in PARSeq. We give a concrete illustration of masked multi-head attention first. Next, the intuition behind the usage of permutation pairs is discussed. Lastly, implementation details and considerations about the training procedure are discussed.

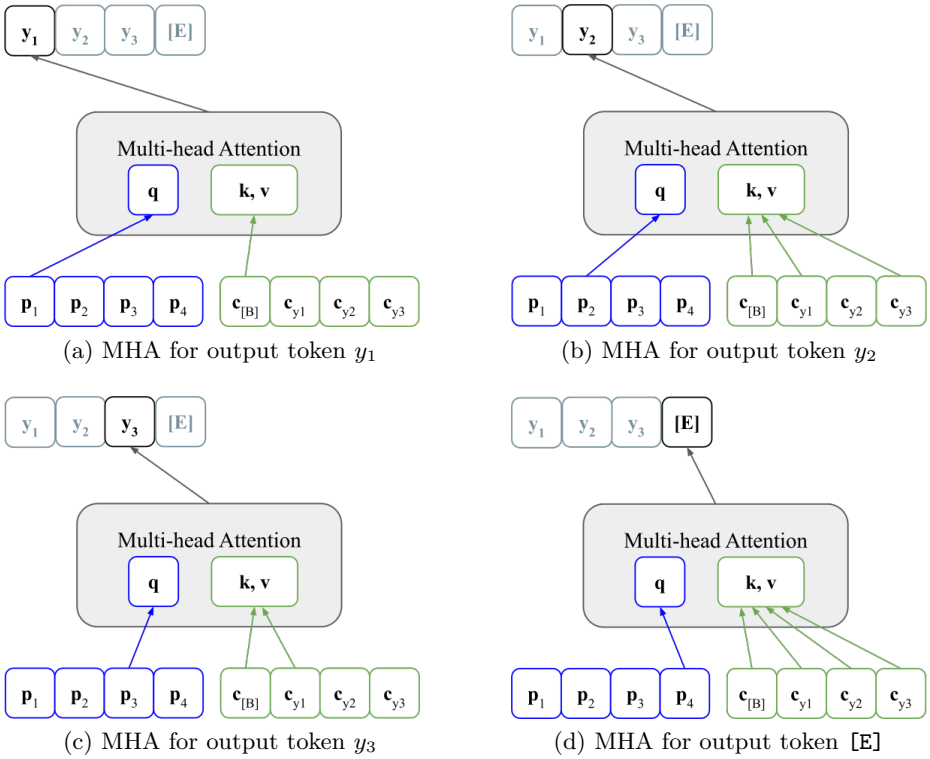
E.1 Illustration of attention masking

As discussed in the main text, Transformers process all tokens in parallel. In order to enforce the *AR* constraint which limits the conditional dependencies for each token, attention masking is used. Figure 3 shows a concrete example of masked multi-head attention for a sequence $\mathbf{y}$. The *position* tokens always serve as the *query* vectors, while the *context* tokens (context *embeddings* with position information) serve as the *key* and *value* vectors. Note that the sequence order is *fixed*, and that only the AR factorization order (specified by the attention mask) is permuted.

E.2 Permutation Sampling

As discussed in the main text, we sample permutations in a specific way. We use pairs of permutations, and the *left-to-right* permutation is always used. Thus, we only sample $K/2-1$ permutations every training step. To illustrate the intuition behind the usage of *flipped* permutation pairs, we give the following example. Given a three-element text label $\mathbf{y} = [y_1, y_2, y_3]$ and $K = 4$ permutations: $[1, 2, 3]$, $[3, 2, 1]$, $[1, 3, 2]$, and $[2, 3, 1]$. The first two permutations are the *left-to-right* and *right-to-left* orderings, respectively. Both are always used as long as $K > 1$. The corresponding factorizations of the joint probability per pair are as follows:

$$p(\mathbf{y})_{[1,2,3]} = p(y_1)p(y_2|y_1)p(y_3|y_1, y_2)$$
$$p(\mathbf{y})_{[3,2,1]} = p(y_3)p(y_2|y_3)p(y_1|y_2, y_3)$$



(a) MHA for output token y_1 (b) MHA for output token y_2

(c) MHA for output token y_3 (d) MHA for output token `[E]`

Fig. 3. Masked MHA for a three-element sequence $\mathbf{y} = [y_1, y_2, y_3]$ given the factorization order $[1, 3, 2]$. $\mathbf{c}$ are context embeddings with position information

$$p(\mathbf{y})_{[1,3,2]} = p(y_1)p(y_3|y_1)p(y_2|y_1, y_3)$$
$$p(\mathbf{y})_{[2,3,1]} = p(y_2)p(y_3|y_2)p(y_1|y_2, y_3)$$

For each permutation pair, if we group the probabilities per element, we get Table 6. Notice that the probabilities of each element for every permutation pair consists of disjoint sets of conditioning variables. For example, the probabilities of element y_1 for $[1, 2, 3]$ (*left-to-right* permutation) and $[3, 2, 1]$ (*right-to-left* permutation) are $p(y_1)$ and $p(y_1|y_2, y_3)$, respectively. The first term is the prior probability of y_1. It is not conditioned on any other element of the text label, unlike the second term which is conditioned on all other elements, y_2 and y_3. Similarly for y_2, the first term is conditioned only on y_1 while the second term is conditioned only on y_3. In our experiments, we find that using flipped permutation pairs results in more stable training dynamics where the loss is smoother and less erratic.

Table 6. Probability terms grouped by permutation pairs

Perm.	y_1	y_2	y_3
$[1, 2, 3]$	$p(y_1)$	$p(y_2\|y_1)$	$p(y_3\|y_1, y_2)$
$[3, 2, 1]$	$p(y_1\|y_2, y_3)$	$p(y_2\|y_3)$	$p(y_3)$
$[1, 3, 2]$	$p(y_1)$	$p(y_2\|y_1, y_3)$	$p(y_3\|y_1)$
$[2, 3, 1]$	$p(y_1\|y_2, y_3)$	$p(y_2)$	$p(y_3\|y_2)$

E.3 Special handling of end-of-sequence [E] token

Although the [E] token is part of the sequence, it is handled in a specific way in order to make training simpler. First, no character $c \in C$, where C is the training charset, is conditioned on [E]. Intuitively, it means that [E] marks the end of the sequence (hence its name) since no more characters are expected after it is produced by the model. More formally, it means that $p(c|\texttt{[E]}) = 0$. This is achieved by masking the positions of [E] in the input context. Second, we train [E] on only two permutations, *left-to-right* and *right-to-left*. The *left-to-right* lookahead mask provides the longest context to [E] (conditioned on all other characters in the sequence), while the *right-to-left* mask provides no context, which is necessary for NAR decoding. We could also train [E] on different subsets of the input context, but doing so needlessly complicates the training procedure without offering any advantages.

E.4 Considerations for batched training

Text labels of varying lengths can be included in a mini-batch. However, the sampled permutations for the mini-batch are always based on the longest sequence.

Hence, it is possible that after accounting for padding, multiple permutations would become equivalent. To see why this is the case, consider a mini-batch containing two samples: the first label has a single character, while the second label has four characters. The first label has a sequence length of one and total number of permutations also equal to one. On the other hand, the second label has a sequence length of four which corresponds to 24 total permutations. If we use $K = 6$ permutations, then it means that the permutations for the first label would be oversampled since there is only one valid permutation for $T = 1$. We find that this oversampling actually helps training. We experimented with a modified training procedure wherein sequences with $T < 4$ are grouped together (*i.e.* 1-, 2-, and 3-character sequences are grouped separately). This training procedure results in increased training time due to the mini-batch being split further into smaller batches, but it does not improve accuracy nor hasten convergence. Thus, we stick with the simpler batched training procedure.

F Dataset Matters

F.1 Open Images Datasets

TextOCR and OpenVINO are datasets both derived from Open Images—a large dataset with very diverse images often containing complex scenes with several objects (8.4 per image on average). Open Images is not specifically collected for STR. Thus, it contains text of varying resolutions, orientations, and quality, as shown in cropped word boxes in Figure 4. TextOCR and OpenVINO significantly overlap in terms of source scene images, as shown in Table 7. Samples of source scene images common to both are shown in Figure 5. Only the *validation* set of OpenVINO and the *test* set of TextOCR do not overlap any other image set. The labels of TextOCR's *test* set are kept private.

Table 7. Overlap between TextOCR and OpenVINO in terms of the number of common source scene images

		TextOCR		
		train	val	test
OpenVINO	train_1	1,612	225	0
	train_2	1,444	230	0
	train_5	1,302	184	0
	train_f	1,068	157	0
	validation	0	0	0

Fig. 4. Cropped word boxes from Open Images

Fig. 5. Examples of source scene images common to TextOCR and OpenVINO

F.2 Data preparation for LMDB storage

We use the archives released by Baek *et al.* [2] for RCTW17, Uber-Text, ArT, LSVT, MLT19, and ReCTS. Thus, we only preprocess data for the remaining datasets.

For COCO-Text, we use the v1.4 *test* annotations released as part of the ICDAR 2017 challenge. For *train* and *val*, we use the latest (v2.0) annotations. We preprocess TextOCR, OpenVINO, and COCO-Text with minimal filtering and modifications, in contrast to the usual practice of removing non-horizontal text and special characters. We only filter illegible and non-machine printed text. The only modification we perform is the removal of whitespace on either side of the label, or duplicate whitespace between non-whitespace characters.

For IC13 and IC15, we use the original data from the ICDAR competition website and perform no modifications to the data. We emulate the previous filtering methods [32,8] to create the subsets used for evaluation.

Long and Yao [21] have reannotated IIIT5k, CUTE, SVT, and SVTP because the original annotations are case-insensitive and lack punctuation marks. However, both the reannotations and the originals contain some errors. Hence, we review inconsistencies between the two versions and manually reconcile them to correct the errors.

Table 8 provides a detailed summary of how each dataset was used.

Table 8. Summary of dataset usage after on-the-fly filtering for the 94-character set. Numbers indicate how many samples were used from each dataset. ^{t} and ^{v} refer to splits that were repurposed as training and validation data, respectively. * indicates private ground truth labels. – indicates that the dataset does not have a particular split. IC13 and IC15 have two *versions* of their respective *test* splits commonly used in the literature

Dataset	*train*	*val*	*test*
MJSynth	7,224,586	802,731^{t}	891,924^{t}
SynthText	6,975,301	–	–
LSVT	41,439	–	–
MLT19	56,727	–	–
RCTW17	10,284	–	–
ReCTS	21,589	–	2,467^{t}
TextOCR	710,994	107,093^{t}	0*
OpenVINO	1,912,784	158,757^{t}	–
ArT	32,028	–	35,149
COCO	59,733	13,394^{t}	9,825
Uber	91,732	36,188^{t}	80,587
IIIT5k	2,000^{v}	–	3,000
SVT	257^{v}	–	647
IC13	848^{v}	–	857 / 1,015
IC15	4,468^{v}	–	1,811 / 2,077
SVTP	–	–	645
CUTE	–	–	288

G Training Details

In the main text, the 169,680 training iterations (batch size of 384) is equivalent to 20 epochs on the combined real training dataset (3,257,585 samples). The same exact number of training iterations is used when training on synthetic data (MJ+ST, 15.89M samples), resulting in just over 4 epochs of training. As shown in Table 9, this training schedule is more than twice as long as Baek *et al.* [2] but is still much shorter than ABINet's original training schedule of 8 epochs VM pretraining on MJ+ST, 80 epochs LM pretraining on WikiText-103, and 10 epochs full model training on MJ+ST. This explains why our reproduction of CRNN and TRBA obtain higher accuracy than the originals, and why ABINet gets a slightly lower (1.4%) accuracy compared to the original results.

Table 9. Training schedule comparison vs reproduced methods. Sorted from shortest to longest schedule based on the *sample count* (essentially *batch size* × *number of iterations*)

Method	**Batch size**	**# of iterations**	**Sample count** (M)
CRNN and TRBA [2]	128	200,000	25.6
ViTSTR [1]	192	300,000	57.6
Ours	384	169,680	65.2
ABINet (VM + full) [13]	384	745,074	286.1
ABINet (LM) [13]	4,096	1,688,720	6,917.0

G.1 Label preprocessing

Preprocessing and filtering are done as follows. Whitespace characters are removed from the labels. Unicode characters are normalized using the NFKD normalization form and then converted to ASCII. Next, labels longer than T characters are filtered. Case-sensitivity is inferred from the charset. If all letters in the charset are lowercase, the label is transformed to its lowercase version. If the charset consists of purely uppercase letters, the label is converted to its uppercase version. If the charset is mixed-case, no case conversion is done. Lastly, all characters not specified in the charset are removed from the labels.

G.2 Learning Rate Optimization

For fair comparison during evaluation, all training hyperparameters—except the learning rate—are kept constant across models. The learning rate is varied because different architectures and model sizes train differently [18]. Ray Tune [19] was used to automatically search for the optimum maximum learning rate given the fixed training schedule. Specifically, we used a combination of Median Stopping [14] and Bayesian Optimization [4] to efficiently narrow down

Table 10. Learning rates used for training. The *Base LR* is the raw value set in the configuration, while the *Effective LR* is the actual value used for training. During pretraining, *ABINet (LM)* is used for ABINet's language model

Model	Base LR	Effective LR
CRNN	5.10×10^{-4}	1.08×10^{-3}
ViTSTR-S	8.90×10^{-4}	1.89×10^{-3}
TRBC	1.00×10^{-4}	2.12×10^{-4}
TRBA	6.90×10^{-4}	1.46×10^{-3}
ABINet (LM)	3.00×10^{-4}	6.36×10^{-4}
ABINet	3.40×10^{-4}	7.21×10^{-4}
PARSeq	7.00×10^{-4}	1.48×10^{-3}

the configuration space. Finally, a grid search over the narrowed down learning rate range was performed with models trained to completion. The final learning rates used for training are shown in Table 10. The base learning rates are scaled using two multipliers: the DDP factor ($\sqrt{nGPU}$) and the batch size linear scaling rule ($bsize/256$) [15], where $nGPU$ refers to the number of GPUs used (*i.e.* $nGPU = 2$ for a dual-GPU setup) and $bsize$ refers to the batch size (*i.e.* $bsize = 384$).

H Accuracy of decoding schemes vs latency

Figure 6 shows how the word accuracy and latency evolve as functions of the number of refinement iterations. For AR decoding, refinement iterations after the first provide negligible increase in accuracy. For NAR decoding, the accuracy increase becomes insignificant after the second iteration. Hence, we use one and two refinement iterations for the AR and NAR decoding schemes, respectively.

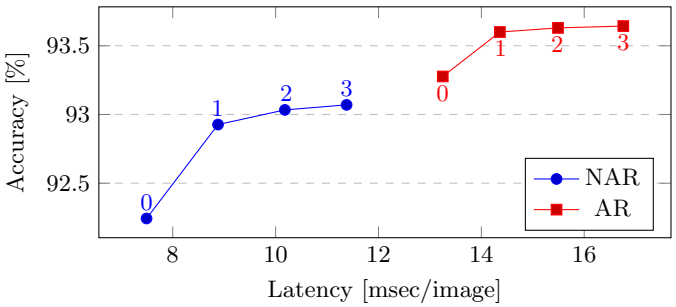


Fig. 6. PARSeq word accuracy and single-image latency for each decoding scheme. The number of refinement iterations used is indicated for each point

I Detailed Latency Measurements

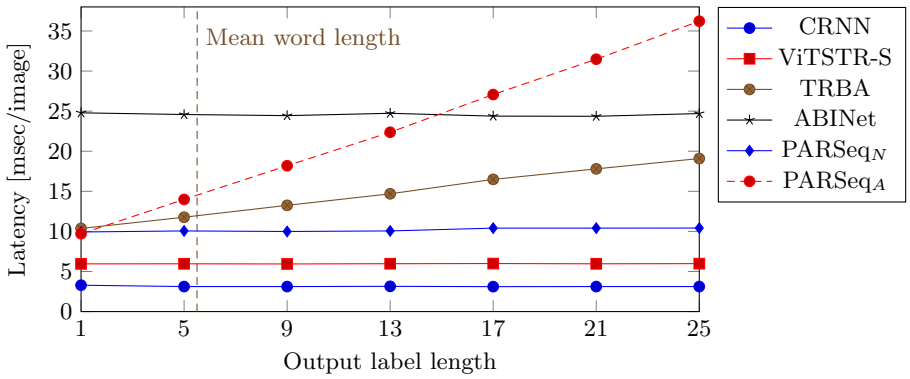


Fig. 7. Model latency vs output label length as measured by PyTorch's benchmark timer on an NVIDIA Tesla A100 GPU. Each point corresponds to a mean of five runs of `Timer.blocked_autorange()`. Lower is better. *Mean word length* is the average length of the labels from all test datasets

We measure model latency in an isolated manner. By doing so, we can reliably factor out the effects of data loading, storage, and CPU latency. We use the built-in benchmarking tool of PyTorch and measure latency for different label lengths, as shown in Figure 7. As expected, NAR methods including PARSeq$_N$ exhibit near-constant latency regardless of output label length. Meanwhile, the latency of AR methods increases linearly as a function of the output label length. The latency increase of PARSeq$_A$ is steeper than TRBA. However, since the average length of words in the test datasets is quite short at 5.4, the actual difference in mean latency between TRBA and PARSeq$_A$ is only about 2.3 msec.

J Experiments on arbitrarily-oriented text

In STR, the focus has mainly been on horizontal text, with a few explicitly tackling arbitrarily-oriented text [25,9,24,36,37]. In our experiments, we observe that existing attention-based models are capable of recognizing text in arbitrary orientation, as shown in Table 11. Only CRNN, a CTC-based model, exhibits dismal orientation robustness. We conjecture that the direct correspondence of visual feature positions to textual feature positions in CTC-based models causes this poor performance. On the other hand, attention-based models compute feature similarity scores on-the-fly, resulting in a more dynamic alignment between visual and textual features.

We hypothesize that regardless of architecture and training procedure, the attention mechanism makes STR models generally robust against orientation

Table 11. Orientation robustness benchmark. Word accuracy (94-char) on rotated versions of the six benchmark datasets. *%dec* refers to the percentage decrease of the *Mean* accuracy w.r.t. the 0° accuracy

Method	0°↑	90°↑	180°↑	270°↑	Mean↑	%dec↓
CRNN	85.8	11.8±0.1	6.4±0.5	10.7±0.2	9.6	88.8
ViTSTR-S	91.8	**87.9**±0.2	78.9±0.3	80.6±0.9	82.4	10.2
TRBA	92.5	84.6±0.1	83.5±0.2	78.6±0.3	82.3	11.0
ABINet	92.4	66.0±1.5	77.1±0.9	65.5±1.8	69.6	24.7
PARSeq_N	92.7	86.7±0.3	83.2±0.9	81.1±0.3	83.7	9.7
PARSeq_A	**93.3**	**88.0**±0.1	**86.6**±0.3	**84.1**±0.1	**86.2**	**7.6**

variations. To test our hypothesis, we created a pose-corrected version of TextOCR which contains text in canonical orientation (practically horizontal), as opposed to the original version which contains text in arbitrary orientation. One possible contributor to the orientation robustness of TRBA is its image rectification module [27]. To test if this is the case, we also train TRBC, the CTC-based version of TRBA. We train all models on either TextOCR variants exclusively and show the results in Table 12.

Table 12. Effect of training on horizontally-oriented (H) vs arbitrarily-oriented (A) variants of TextOCR. 0° pertains to model accuracy (94-char) on non-rotated benchmark datasets. *Rotated* refers to the mean accuracy on 90°, 180°, and 270° rotations of the benchmark datasets. In H vs A per row, bold indicates significantly higher accuracy

Method	0°		*Rotated*	
	H	A	H	A
CRNN	84.7±0.4	84.0±0.4	0.8±0.0	**8.7**±0.4
ViTSTR-S	87.8±0.7	88.1±0.3	2.5±0.2	**72.8**±1.3
TRBC	87.0±0.2	87.3±0.2	1.3±0.1	**17.3**±1.9
TRBA	89.7±0.0	**90.1**±0.2	2.7±0.1	**76.3**±0.4
ABINet	90.3±0.2	90.7±0.2	3.0±0.3	**63.9**±1.5
PARSeq_N	89.8±0.3	**90.4**±0.0	6.4±0.4	**77.9**±0.6
PARSeq_A	90.6±0.0	**91.3**±0.2	8.4±0.4	**80.7**±0.3

We observe that both CTC-based and attention-based models can be trained on arbitrarily-oriented text. The mean 0° accuracy decreased for CRNN but the decrease was not statistically-significant. For other models, the mean accuracy even increased with TRBA and PARSeq showing statistically-significant improvements. This suggests that the common practice of filtering non-horizontal text might be unnecessary. As far as arbitrarily-oriented text recognition is concerned, training on arbitrarily-oriented text expectedly improves the accuracy across all models. However, the improvement is minimal in CTC-based models compared to attention-based models. Moreover, TRBC exhibits slightly better orientation robustness compared to CRNN, but it still performs badly compared to TRBA. This suggests that the contribution of the image rectification module

to orientation robustness is minimal, and that the attention mechanism is the primary contributor to orientation robustness.

K Combined word accuracy

Six small datasets are typically used to benchmark STR methods, resulting in six different mean values for word accuracy. The combined word accuracy is typically reported too, but we did not include it in Table 6 of the main text because of space constraints and possible confusion due to inconsistencies in test sets used. Table 13 shows the combined word accuracy on the benchmark (7,672 samples) and on the smaller test subset (using IC13 857 and IC15 1,811) with a total of 7,248 samples.

Table 13. Word accuracy on the six benchmark datasets (36-character set). For *Train data*: Synthetic datasets (**S**) - MJ [17] and ST [16]; Benchmark datasets (**B**) - SVT, IIIT5k, IC13, and IC15; Real datasets (**R**) - COCO, RCTW17, Uber, ArT, LSVT, MLT19, ReCTS, TextOCR, and OpenVINO; "*" denotes usage of character-level labels. In our experiments, bold indicates the highest word accuracy per column. [1]Used with SCATTER [20]. [2]SynthText without special characters (5.5M samples). [3]LM Pretrained on WikiText-103 [22]

	Method	Train data	Test datasets and # of samples: Total 7,248	Test datasets and # of samples: Total (*benchmark*) 7,672
Published Results	PlugNet [23]	S	–	89.8
	SRN [38]	S	90.4	–
	RobustScanner [39]	S,B	–	89.2
	TextScanner [31]	S*	–	91.0
	AutoSTR [40]	S	–	–
	RCEED [10]	S,B	–	–
	PREN2D [36]	S	91.5	–
	VisionLAN [34]	S	91.2	–
	Bhunia *et al.* [6]	S	–	90.9
	CVAE-Feed.[1] [5]	S	–	–
	STN-CSTR [7]	S	–	–
	CRNN [2]	S	–	75.8
	ViTSTR-B [1]	S[2]	85.6	83.8
	TRBA [2]	S	–	85.7
	ABINet [13]	S[3]	92.7	–
Experiments	CRNN	S	84.5±0.2	83.2±0.2
	ViTSTR-S	S	90.0±0.1	88.6±0.0
	TRBA	S	92.0±0.2	90.6±0.1
	ABINet	S	91.3±0.2	89.8±0.1
	PARSeq$_N$ (Ours)	S	92.0±0.2	90.7±0.2
	PARSeq$_A$ (Ours)	S	93.2±0.2	91.9±0.2
	CRNN	R	89.6±0.1	88.5±0.0
	ViTSTR-S	R	94.7±0.1	94.3±0.1
	TRBA	R	95.7±0.1	95.2±0.1
	ABINet	R	95.9±0.2	95.2±0.1
	PARSeq$_N$ (Ours)	R	95.7±0.1	95.2±0.1
	PARSeq$_A$ (Ours)	R	96.4±0.0	96.0±0.0

L Qualitative Results

In the following tables, shown are qualitative results for all test datasets and for some images obtained from the internet. The input images are shown in their original orientations and in aspect ratios close to their original. For predictions which are roughly aligned to the ground truth, wrong characters are highlighted in red while missing characters are indicated by a red underscore _.

Table 14 shows the results for samples from regular datasets like IIIT5k, SVT, and IC13. Most of the models did not have a problem recognizing the fairly clear, horizontal, and high-resolution input images. The only exception is CRNN failing to recognize any character from the tilted *CITY* image sample of SVT. No model was able to correctly recognize *Verbandstoffe* due to the ambiguity caused by motion blur, making the character *o* look like an *e*.

Table 14. Qualitative results on samples from regular datasets IIIT5k, SVT, and IC13. *GT* refers to the ground truth label

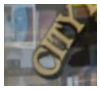
	Input	GT	Predictions				
			PARSeq$_A$	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
IIIT5k		Kellimar	Kellimar	Kellimar	Kellimar	Kellimar	Kellimar
		TIDE	TIDE	TIDE	TIDE	TIDE	TIDE
		Coca-Cola	Coca-Cola	Coca-Cola	Coca-Cola	Coca-Cola	Coca-Cola
		NESCAFE	NESCAFE	NESCAFE	NESCAFE	NESCAFE	NESCAFE
SVT		ICEBOX	ICEBOX	ICEBOX	ICEBOX	OCESOX	IREBOX
		CITY	CITY	CITY	CITY	CITY	—
		BREWERY	BREWERY	BREWERY	BREWERY	BREWERY	BREWERY
		THE	THE	THE	THE	THE	THE
IC13		Distributed	Distributed	Distributed	Distributed	Distributed	Distrm_uted
		Verbandstoffe	Verbandsteffe	Verbandsteffe	Verbandstelle	Verbandsteffe	Verbandsteffe
		GALORE	GALORE	GALORE	GALORE	CALORE	GALORE
	Input	**GT**	**PARSeq$_A$**	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
			Predictions				

Table 15 shows the qualitative results for samples from the IC15 dataset. Context-free methods, ViTSTR and CRNN, were not able to correctly predict *Kappa* possibly due to the ambiguity caused by distortion on the first *p* char-

acter. ABINet and CRNN both have difficulty in recognizing vertically-oriented (*CONCIERGE*) and rotated text (*UNSEEN*). No model correctly predicted *epiCentre* due to the case ambiguity of the character *C*. Only PARSeq and CRNN were able to correctly read the telephone number.

Table 15. Qualitative results from IC15 samples

		Predictions				
Input	**GT**	**PARSeq$_A$**	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	Kappa	Kappa	Kappa	Kappa	Kaopa	Kadpa
	UNSEEN	UNSEIN	UNITIN	UNSEEN	UNSEEN	MATA
	CONCIERGE	CONCIERGE	_ONNIIEOO	CONCIERGE	CONCIERGE	—
	epiCentre Tel:7778100	epicentre Tel:7778100	epicentre Tel:77778100	epicentre Teles7778100	epicentre Tel:17778100	eplcentre Tel:7778100

Table 16. Qualitative results from SVTP samples

		Predictions				
Input	**GT**	**PARSeq$_A$**	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	MINT	MINT	AINT	MINT	MINT	MINT
	REDWOOD	REDWA_D	maCyyro	Programmer	REDWOBD	Pe
	HOUSE	HOUCE	HOUSE	HOUSE	HOUCE	HOUSE
	Restaurant	Restaurant	Restaurant	Restaurant	Restaurant	Restaurant
	CARLTON	CARLTON	CARLTON	CARLTON	CARLTON	ANO

Table 16 shows the qualitative results for SVTP samples. All models except ABINet were able to recognize *MINT*. No model correctly recognized the vertically-oriented text, *REDWOOD*, with ViTSTR and PARSeq producing the two closest predictions. Surprisingly, both PARSeq and ViTSTR fail at the relatively easy *HOUSE*, where the character *S* is occluded. In PARSeq, the visual

features have a *stronger* effect on the final output than the textual features due to the *image–position* MHA being closer to the final decoder hidden state. Thus, a low-confidence visual feature might sway the output to the wrong character given enough magnitude relative to the textual features. All models correctly recognized *Restaurant* even though the image is relatively blurry. All models except CRNN correctly recognized the vertically-oriented text, *CARLTON*.

Table 17. Qualitative results from CUTE80 samples

		Predictions				
Input	**GT**	**PARSeq$_A$**	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	BALLYS	BALLYS	BALLY'S	BALLYS	BALLYS	BALLYS
	eBizu	eBizu	eBizu	eBizu	eBizu	eBizu
	CLUB	CLUB	CLUB	CLUB	CLUB	2U1
	SALMON	SALMON	SALMON	SALMON	SALMON	SA_NON

Table 17 shows the results for CUTE80, a dataset which primarily contains curved text. The samples are high-resolution and of good quality resulting in generally accurate recognition across models. The only exceptions are *BALLYS* for ABINet and the relatively vertical texts *CLUB*, and *SALMON* for CRNN.

Table 18. Qualitative results from ArT samples

		Predictions				
Input	**GT**	**PARSeq$_A$**	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	FONDENTE	FONDENIE	FONDENIE	FONDEN_S	FONDENTE	FOMEON
	cuisine	cuisine	cuisine	cuisine	cuisine	culsi_e
	COFFEE	COFFEE	COFFEE	COFFEE	COFFEE	COFFEE
	Franziskaner	Franziskaner	Franziskaner	Franziskaner	Franziskaner	_Ranzishanes
	TOMORROW'S	TOMORROV'S	TOMORRO_'SS	TOMORROW'S	TOMORROW'S	TO--------

Table 18 shows the results for samples from ArT, a dataset of arbitrarily-oriented and curved text. CRNN fails to recognize text which are vertically-oriented. Only ViTSTR is able to recognize *FONDENTE* correctly, with PARSeq and ABINet both predicting *I* in place of *T*. For the almost upside down *TOMORROW'S*, only TRBA and ViTSTR are able to recognize it. PARSeq possibly mistook *W* for a *V* due to aspect ratio distortion (vertical image being rescaled into a horizontal one).

Table 19 shows the results for COCO-Text samples. No model was able to recognize *ANTS*, possibly due to the presence of small stray characters around the main text. All models were able to recognize *XT-862K* in spite of the blurry image, and *People* in spite of the occluded *o* and *p* characters. *Chevron* is a particularly hard sample due to the last two characters being occluded by two different objects. Only PARseq was able to detect the last two characters and correctly recognize the *o* character, while all other models only recognized the first five characters. *GUNNESS* is another hard sample due to its low resolution and occluded character. Only PARSeq was able to infer the occluded character correctly.

Table 19. Qualitative results from COCO-Text samples

		Predictions				
Input	**GT**	**PARSeq$_A$**	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	ANTS	ANTSTS	KANTER	BANTSEN	AATSSE	_N__
	XT-862K	XT-862K	XT-862K	XT-862K	XT-862K	XT-862K
	People	People	People	People	People	People
	Chevron	Chevrol	Chevr__	Chevr__	Chevr__	Chevr__
	GUNNESS	GUNNESS	GSNNNSSS	_AWNESS	GONNESSS	GOWNESS

Table 20 shows the qualitative results for ReCTS, a dataset which contains fairly high-resolution text with unconventional font styles. Model performance across all samples is generally good since they are clear and horizontally-oriented. No model correctly predicted the string of digits, with PARSeq and TRBA producing the closest predictions with only one wrong character. Most models correctly predicted *AWON* except for PARSeq and CRNN which mistook the occluded *W* for an *N*.

Table 21 shows the results for Uber-Text, a dataset which contains many vertical or rotated text from outdoor signages. PARSeq is the only model to correctly recognize all samples.

Table 20. Qualitative results from ReCTS samples

Input	GT	Predictions				
		PARSeq$_A$	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	SEVEN	SEVEN	SEVEN	SEVEN	SEVEN	SEVEN
	TEA	TEA	TEA	TEA	TEA	TEA
	13031863597	13031867597	13071863599	13031863599	13071967597	19091967599
	Bubble	Bubble	Bubble	Bubble	Bubble	Bubble
	AWON	ANON	AWON	AWON	AWON	A'NON

Table 21. Qualitative results from Uber-Text samples

Input	GT	Predictions				
		PARSeq$_A$	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	MeridethLn	MeridethLn	Meritttеww.....	MeridethLn	MeridethLn	wata
	5811	5811	3911	5011	5811	40
	PARKING 2017	PARKING 2017	PARKING 2017	PARKING 2017	PARKING 2017	POME 2017

Table 22 shows results from additional samples obtained from the Internet. Overall, the samples are high-resolution, horizontally-oriented, and use unconventional fonts similar to ReCTS. PARSeq is the only model to correctly recognize all samples, particularly *Creative* which uses a cursive handwriting type of font.

Table 22. Qualitative results from samples obtained from the internet

Input	GT	Predictions				
		PARSeq$_A$	**ABINet**	**TRBA**	**ViTSTR-S**	**CRNN**
	COCKTAILS	COCKTAILS	COCKTAILS	COCKTAHIS	COCKTAILS	COCKTAILS
	Creative	Creative	Creat_ne	Crestire	Creat_ee	Cedrre
	TOGARASHI	TOGARASHI	TOGARASHI	TOGARASHI	TOGARASHI	TOGARASH!

References

1. Atienza, R.: Vision transformer for fast and efficient scene text recognition. In: International Conference on Document Analysis and Recognition (ICDAR) (2021)
2. Baek, J., Matsui, Y., Aizawa, K.: What if we only use real datasets for scene text recognition? toward scene text recognition with fewer labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3113–3122 (6 2021)
3. Baevski, A., Auli, M.: Adaptive input representations for neural language modeling. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=ByxZX20qFQ
4. Balandat, M., Karrer, B., Jiang, D.R., Daulton, S., Letham, B., Wilson, A.G., Bakshy, E.: BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In: Advances in Neural Information Processing Systems 33 (2020), http://arxiv.org/abs/1910.06403
5. Bhunia, A.K., Chowdhury, P.N., Sain, A., Song, Y.Z.: Towards the unseen: Iterative text recognition by distilling from errors. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14950–14959 (10 2021)
6. Bhunia, A.K., Sain, A., Kumar, A., Ghose, S., Chowdhury, P.N., Song, Y.Z.: Joint visual semantic reasoning: Multi-stage decoder for text recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14940–14949 (10 2021)
7. Cai, H., Sun, J., Xiong, Y.: Revisiting classification perspective on scene text recognition (2021), https://arxiv.org/abs/2102.10884
8. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: Proceedings of the IEEE international conference on computer vision. pp. 5076–5084 (2017)
9. Cheng, Z., Xu, Y., Bai, F., Niu, Y., Pu, S., Zhou, S.: Aon: Towards arbitrarily-oriented text recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5571–5579 (2018)
10. Cui, M., Wang, W., Zhang, J., Wang, L.: Representation and correlation enhanced encoder-decoder framework for scene text recognition. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) Document Analysis and Recognition – ICDAR 2021. pp. 156–170. Springer International Publishing, Cham (2021)
11. Dollár, P., Singh, M., Girshick, R.: Fast and accurate model scaling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 924–932 (2021)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
13. Fang, S., Xie, H., Wang, Y., Mao, Z., Zhang, Y.: Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7098–7107 (6 2021)
14. Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J.E., Sculley, D. (eds.): Google Vizier: A Service for Black-Box Optimization (2017), http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization

15. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
16. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
17. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. In: Workshop on Deep Learning, NIPS (2014)
18. Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., Gonzalez, J.: Train big, then compress: Rethinking model size for efficient training and inference of transformers. In: International Conference on Machine Learning. pp. 5958–5968. PMLR (2020)
19. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018)
20. Litman, R., Anschel, O., Tsiper, S., Litman, R., Mazor, S., Manmatha, R.: Scatter: Selective context attentional scene text recognizer. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
21. Long, S., Yao, C.: Unrealtext: Synthesizing realistic scene text images from the unreal world. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
22. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=Byj72udxe
23. Mou, Y., Tan, L., Yang, H., Chen, J., Liu, L., Yan, R., Huang, Y.: Plugnet: Degradation aware scene text recognition supervised by a pluggable super-resolution unit. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16. pp. 158–174. Springer (2020)
24. Munjal, R.S., Prabhu, A.D., Arora, N., Moharana, S., Ramena, G.: Stride: Scene text recognition in-device. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2021)
25. Phan, T.Q., Shivakumara, P., Tian, S., Tan, C.L.: Recognizing text with perspective distortion in natural scenes. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 569–576 (2013)
26. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence **39**(11), 2298–2304 (2016)
27. Shi, B., Wang, X., Lyu, P., Yao, C., Bai, X.: Robust scene text recognition with automatic rectification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4168–4176 (2016)
28. Sun, Y., Ni, Z., Chng, C.K., Liu, Y., Luo, C., Ng, C.C., Han, J., Ding, E., Liu, J., Karatzas, D., et al.: Icdar 2019 competition on large-scale street view text with partial labeling-rrc-lsvt. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1557–1562. IEEE (2019)
29. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357. PMLR (2021)

30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
31. Wan, Z., He, M., Chen, H., Bai, X., Yao, C.: Textscanner: Reading characters in order for robust scene text recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 12120–12127 (2020)
32. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: 2011 International Conference on Computer Vision. pp. 1457–1464. IEEE (2011)
33. Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D.F., Chao, L.S.: Learning deep transformer models for machine translation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 1810–1822 (2019)
34. Wang, Y., Xie, H., Fang, S., Wang, J., Zhu, S., Zhang, Y.: From two to one: A new scene text recognizer with visual language modeling network. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14194–14203 (10 2021)
35. Xiao, T., Dollar, P., Singh, M., Mintun, E., Darrell, T., Girshick, R.: Early convolutions help transformers see better. Advances in Neural Information Processing Systems **34** (2021)
36. Yan, R., Peng, L., Xiao, S., Yao, G.: Primitive representation learning for scene text recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 284–293 (6 2021)
37. Yan, R., Peng, L., Xiao, S., Yao, G., Min, J.: Mean: Multi-element attention network for scene text recognition. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 1–8. IEEE (2021)
38. Yu, D., Li, X., Zhang, C., Liu, T., Han, J., Liu, J., Ding, E.: Towards accurate scene text recognition with semantic reasoning networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12113–12122 (2020)
39. Yue, X., Kuang, Z., Lin, C., Sun, H., Zhang, W.: Robustscanner: Dynamically enhancing positional clues for robust text recognition. In: European Conference on Computer Vision. pp. 135–151. Springer (2020)
40. Zhang, H., Yao, Q., Yang, M., Xu, Y., Bai, X.: Autostr: Efficient backbone search for scene text recognition. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16. pp. 751–767. Springer (2020)