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

Abstract. Context-aware STR methods typically use internal autoregressive (AR) language models (LM). Inherent limitations of AR models motivated two-stage methods which employ an external LM. The conditional independence of the external LM on the input image may cause it to erroneously rectify correct predictions, leading to significant inefficiencies. Our method, PARSeq, learns an ensemble of internal AR LMs with shared weights using Permutation Language Modeling. It unifies context-free non-AR and context-aware AR inference, and iterative refinement using bidirectional context. Using synthetic training data, PARSeq achieves state-of-the-art (SOTA) results in STR benchmarks (91.9% accuracy) and more challenging datasets. It establishes new SOTA results (96.0% accuracy) when trained on real data. PARSeq is optimal on accuracy vs parameter count, FLOPS, and latency because of its simple, unified structure and parallel token processing. Due to its extensive use of attention, it is robust on arbitrarily-oriented text, which is common in real-world images. Code, pretrained weights, and data are available at: `https://github.com/baudm/parseq`.

Keywords: scene text recognition, permutation language modeling, autoregressive modeling, cross-modal attention, transformer

1 Introduction

Machines read text in natural scenes by first detecting text regions, then recognizing text in those regions. The task of recognizing text from the cropped regions is called Scene Text Recognition (STR). STR enables the reading of road signs, billboards, paper bills, product labels, logos, printed shirts, *etc*. It has practical applications in self-driving cars, augmented reality, retail, education, and devices for the visually impaired, among others. In contrast to Optical Character Recognition (OCR) in documents where the text attributes are more uniform, STR has to deal with varying font styles, orientations, text shapes, illumination, amount of occlusion, and inconsistent sensor conditions. Images captured in natural environments could also be noisy, blurry, or distorted. In essence, STR is an important but very challenging problem.

STR is mainly a vision task, but in cases where parts of the text are impossible to read, *e.g*. due to an occluder, the image features alone will not be enough

to make accurate inferences. In such cases, language semantics is typically used to aid the recognition process. Context-aware STR methods incorporate semantic priors from a word representation model [48] or dictionary [45], or learned from data [52,32,3,50,33,69,21,53,9] using sequence modeling [6,60].

Sequence modeling has the advantage of learning end-to-end trainable language models (LM). STR methods with *internal* LMs jointly process image features and language context. They are trained by enforcing an autoregressive (AR) constraint on the language context where *future* tokens are conditioned on *past* tokens but not the other way around, resulting in the model $P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|\mathbf{y}_{<t}, \mathbf{x})$ where $\mathbf{y}$ is the T-length text label of the image $\mathbf{x}$. AR models have two inherent limitations arising from this constraint. First, the model is able to learn the token dependencies in one direction only—usually the left-to-right (LTR) direction. This unidirectionality causes AR models to be biased towards a single reading direction, resulting in spurious addition of suffixes or direction-dependent predictions (illustrated in Appendix A). Second, during inference, the AR model can only be used to output tokens serially in the same direction used for training. This is called next-token or monotonic AR decoding.

(a) ABINet (b) Unified STR model (Ours)

Fig. 1. (a) State-of-the-art method ABINet [21] uses a combination of context-free vision and context-aware language models. The language model functions as a *spell checker* but is prone to erroneous rectification of correct initial predictions due to its conditional independence on the image features. (b) Our proposed method performs both initial decoding and iterative refinement by jointly processing image and context features, resulting in a single holistic output. This eschews the need for separate language and fusion models, resulting in a more efficient and robust STR method

To address these limitations, prior works have combined left-to-right and right-to-left (RTL) AR models [53,9], or opted for a two-stage approach using an ensemble of a context-free STR model with a standalone or external LM [69,21]. A combined LTR and RTL AR model still suffers from unidirectional context, but works around it by performing two separate decoding streams—one for each direction—then choosing the prediction with the higher likelihood. Naturally, this results in increased decoding time and complexity. Meanwhile, two-stage ensemble approaches like in Figure 1a obtain their initial predictions using parallel non-AR decoding. The initial context-less prediction is decoded directly from the image using the context-free model $P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|\mathbf{x})$. This enables the external LM, $P(\mathbf{y}) = \prod_{t=1}^{T} P(y_t|\mathbf{y}_{\neq t})$ in ABINet [21] for example, to use bidirectional context since all characters are available at once. The LM functions as a *spell checker* and rectifies the initial prediction, producing a

context-based output. The conditional independence of the LM from the input image may cause it to erroneously rectify correct predictions if they appear misspelled, or if a similar word with a higher likelihood exists. This is evident in the low word accuracy of the LM in SRN (27.6%) and in ABINet (41.9%) when used as a spell checker [21]. Hence, a separate fusion layer is used to combine the features from the initial prediction and the LM prediction to get the final output. A closer look at the LM of ABINet (Appendix B) reveals that it is inefficient for STR. It is underutilized relative to its parameter count, and it exhibits dismal word accuracy despite using a significant chunk of the overall compute requirements of the full ABINet model.

In sequence model literature, there has been recent interest in generalized models of sequence generation. Various neural sequence models, such as AR and refinement-based non-AR, were shown to be special cases in the generalized framework proposed by Mansimov *et al*. [40]. This result posits that the same generalization can be done in STR models, unifying context-free and context-aware STR. While the advantages of this unification are not apparent, we shall show later that such a generalized model enables the use of an internal LM while maintaining the refinement capabilities of an external LM.

Permutation Language Modeling (PLM) was originally proposed for large-scale language pretraining [68], but recent works [58,47] have adapted it for learning Transformer-based generalized sequence models capable of different decoding schemes. In this work, we adapt PLM for STR. PLM can be considered a generalization of AR modeling, and a PLM-trained model can be seen as an ensemble of AR models with shared architecture and weights [59]. With the use of attention masks for dynamically specifying token dependencies, such a model, illustrated in Figure 2, can learn and use conditional character probabilities given an arbitrary subset of the input context, enabling monotonic AR decoding, parallel non-AR decoding, and even iterative refinement.

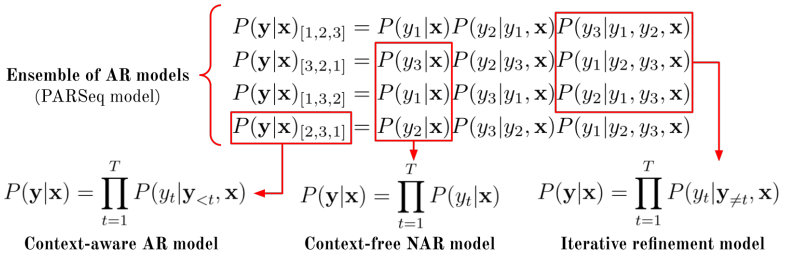

Fig. 2. Illustration of NAR and iterative refinement (cloze) models in relation to an ensemble of AR models for an image $\mathbf{x}$ with a three-element text label $\mathbf{y}$. Four different factorizations of $P(\mathbf{y}|\mathbf{x})$ (out of six possible) are shown, with each one determined by the factorization order shown in the subscript

In summary, state-of-the-art (SOTA) STR methods [69,21] opted for a two-stage ensemble approach in order to use bidirectional language context. The low

word accuracy of their external LMs, despite increased training and runtime requirements, highlights the need for a more efficient approach. To this end, we propose a **p**ermuted **a**uto**r**egressive **seq**uence (PARSeq) model for STR. Trained with PLM, PARSeq is a unified STR model with a simple structure, but is capable of both context-free and context-aware inference, as well as iterative refinement using bidirectional (*cloze*) context. PARSeq achieves SOTA results on the STR benchmarks for both synthetic and real training data (Table 6) across all character sets (Table 4), while being optimal in its use of parameters, FLOPS, and runtime (Figure 5). For a more comprehensive comparison, we also benchmark on larger and more difficult real datasets which contain occluded and arbitrarily-oriented text (Figure 4b). PARSeq likewise achieves SOTA results in these datasets (Table 5).

2 Related Work

The recent surveys of Long *et al*. [38] and Chen *et al*. [12] provide comprehensive discussions on different approaches in STR. In this section, we focus on the use of language semantics in STR.

Context-free STR methods directly predict the characters from image features. The output characters are conditionally-independent of each other. The most prominent approaches are CTC-based [22] methods [51,37,63,10], with a few using different approaches such as self-attention [20] for pooling features into character positions [2], or casting STR as a multi-instance classification problem [25,11]. Ensemble methods [69,21] use an attention mechanism [6,60] to produce the initial context-less predictions. Since context-free methods rely solely on the image features for prediction, they are less robust against corruptions like occluded or incomplete characters. This limitation motivated the use of language semantics for making the recognition model more robust.

Context-aware STR methods typically use semantics learned from data to aid in recognition. Most approaches [3,32,52,13,53] use RNNs with attention [6] or Transformers [50,33,9] to learn internal LMs using the standard AR training. These methods are limited to monotonic AR decoding. Ensemble methods [69,21] use bidirectional context via an external LM for prediction refinement. The conditional independence of the external LM on image features makes it prone to erroneous rectification, limiting usefulness while incurring significant overhead. VisionLAN [66] learns semantics by selectively masking image features of individual characters during training, akin to denoising autoencoders and Masked Language Modeling (MLM) [19]. In contrast to prior work, PARSeq learns an internal LM using PLM instead of the standard AR modeling. It supports flexible decoding by using a parameterization which decouples the target decoding position from the input context, similar to the *query stream* of two-stream attention [68]. Unlike ABINet [21] which uses the *cloze* context for both training and inference, PARSeq uses it for iterative refinement only. Moreover, as said earlier, the refinement model of ABINet is conditionally independent of the in-

put image, while PARSeq considers both input image and language context in the refinement process.

Generation from Sequence Models can be categorized into two contrasting schemes: autoregressive (one token at a time) and non-autoregressive (all tokens predicted at once). Mansimov *et al*. [40] proposed a generalized framework for sequence generation which unifies the said schemes. BANG [47] adapted two-stream attention [68] for use with MLM, in contrast to our use of PLM. PMLM [34] is trained using a generalization of MLM where the masking ratio is stochastic. A variant which uses a uniform prior was shown to be equivalent to a PLM-trained model. Closest to our work is Tian *et al*. [58] which adapts the two-stream attention parameterization [68] to decoders by interspersing the content and query streams from different layers. In contrast, our decoder does not use self-attention and does not intersperse the two streams. This allows our single layer decoder to use the query stream only, and avoid the overhead of the unused content stream.

3 Permuted Autoregressive Sequence Models

In this section, we first present the Transformer-based model architecture of PARSeq. Next, we discuss how to train it using Permutation Language Modeling. Lastly, we show how to use the trained model for inference by discussing the different decoding schemes and the iterative refinement procedure.

3.1 Model Architecture

Multi-head Attention (MHA) [60] is extensively used by PARSeq. We denote it as $MHA(\mathbf{q}, \mathbf{k}, \mathbf{v}, \mathbf{m})$, where $\mathbf{q}$, $\mathbf{k}$, and $\mathbf{v}$ refer to the required parameters *query*, *key*, and *value*, while $\mathbf{m}$ refers to the optional attention mask. We provide the background material on MHA in Appendix C.

PARSeq follows an encoder-decoder architecture, shown in Figure 3, commonly used in sequence modeling tasks. The encoder has 12 layers while the decoder is only a single layer. This *deep-shallow* configuration [28] is a deliberate design choice which minimizes the overall computational requirements of the model while having a negligible impact in performance. Details in Appendix D.

ViT Encoder. Vision Transformer (ViT) [20] is the direct extension of the Transformer to images. A ViT layer contains one MHA module used for *self-attention*, *i.e.* $\mathbf{q} = \mathbf{k} = \mathbf{v}$. The encoder is a 12-layer ViT without the classification head and the `[CLS]` token. An image $\mathbf{x} \in \mathbb{R}^{W \times H \times Ch}$, with width W, height H, and number of channels Ch, is *tokenized* by evenly dividing it into $p_w \times p_h$ patches, flattening each patch, then linearly projecting them into d_{model}-dimensional tokens using a patch embedding matrix $\mathbf{W}^p \in \mathbb{R}^{p_w p_h Ch \times d_{model}}$, resulting in $(WH)/(p_w p_h)$ tokens. Learned position embeddings of equal dimension are added to the tokens prior to being processed by the first ViT layer.

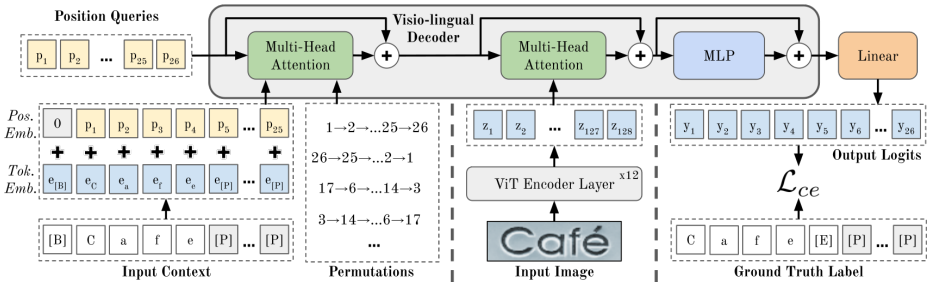

Fig. 3. PARSeq architecture and training overview. *LayerNorm* and *Dropout* layers are omitted due to space constraints. `[B]`, `[E]`, and `[P]` stand for *beginning-of-sequence (BOS)*, *end-of-sequence (EOS)*, and *padding* tokens, respectively. $T = 25$ results in 26 distinct *position* tokens. The position tokens both serve as query vectors and position embeddings for the input context. For `[B]`, no position embedding is added. Attention masks are generated from the given permutations and are used only for the *context-position* attention. $\mathcal{L}_{ce}$ pertains to the cross-entropy loss

In contrast to the standard ViT, all output tokens $\mathbf{z}$ are used as input to the decoder:

$$\mathbf{z} = Enc(\mathbf{x}) \in \mathbb{R}^{\frac{WH}{p_w p_h} \times d_{model}} \tag{1}$$

Visio-lingual Decoder. The decoder follows the same architecture as the pre-*LayerNorm* [5,65] Transformer decoder but uses twice the number of attention heads, *i.e.* $nhead = d_{model}/32$. It has three required inputs consisting of *position*, *context*, and *image* tokens, and an optional attention mask.

In the following equations, we omit *LayerNorm* and *Dropout* for brevity. The first *MHA* module is used for *context–position* attention:

$$\mathbf{h}_c = \mathbf{p} + MHA(\mathbf{p}, \mathbf{c}, \mathbf{c}, \mathbf{m}) \in \mathbb{R}^{(T+1) \times d_{model}} \tag{2}$$

where T is the context length, $\mathbf{p} \in \mathbb{R}^{(T+1) \times d_{model}}$ are the position tokens, $\mathbf{c} \in \mathbb{R}^{(T+1) \times d_{model}}$ are the context embeddings with positional information, and $\mathbf{m} \in \mathbb{R}^{(T+1) \times (T+1)}$ is the optional *attention mask*. Note that the use of special *delimiter* tokens (`[B]` or `[E]`) increases the total sequence length to $T + 1$.

The *position* tokens encode the target position to be predicted, each one having a direct correspondence to a specific position in the output. This parameterization is similar to the *query stream* of two-stream attention [68]. It decouples the context from the target position, allowing the model to learn from PLM. Without the position tokens, *i.e.* if the context tokens are used as *queries* themselves like in standard Transformers, the model will not learn anything meaningful from PLM and will simply function like a *standard* AR model.

The supplied mask varies depending on how the model is used. During training, masks are generated from random permutations (Section 3.2). At inference

(Section 3.3), it could be a standard left-to-right lookahead mask (AR decoding), a *cloze* mask (iterative refinement), or no mask at all (NAR decoding).

The second MHA is used for *image–position* attention:

$$\mathbf{h}_i = \mathbf{h}_c + MHA(\mathbf{h}_c, \mathbf{z}, \mathbf{z}) \in \mathbb{R}^{(T+1)\times d_{model}} \tag{3}$$

where no attention mask is used. The last decoder hidden state is the output of the MLP, $\mathbf{h}_{dec} = \mathbf{h}_i + MLP(\mathbf{h}_i) \in \mathbb{R}^{(T+1)\times d_{model}}$.

Finally, the output logits are $\mathbf{y} = Linear(\mathbf{h}_{dec}) \in \mathbb{R}^{(T+1)\times(C+1)}$ where C is the size of the character set (charset) used for training. The additional character pertains to the `[E]` token (which marks the end of the sequence). In summary, given an attention mask $\mathbf{m}$, the decoder is a function which takes the form:

$$\mathbf{y} = Dec(\mathbf{z}, \mathbf{p}, \mathbf{c}, \mathbf{m}) \in \mathbb{R}^{(T+1)\times(C+1)} \tag{4}$$

3.2 Permutation Language Modeling

Given an image $\mathbf{x}$, we want to maximize the likelihood of its text label $\mathbf{y} = [y_1, y_2, \ldots, y_T]$ under the set of model parameters θ. In standard AR modeling, the likelihood is factorized using the chain rule according to the canonical ordering, $[1, 2, \ldots, T]$, resulting in the model $\log p(\mathbf{y}|\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(y_t|\mathbf{y}_{<t}, \mathbf{x})$. However, Transformers process all tokens in parallel, allowing the output tokens to *access* or be conditionally-dependent on all the input tokens. In order to have a valid AR model, *past* tokens cannot have access to *future* tokens. The AR property is enforced in Transformers with the use of attention masks. For example, a standard AR model for a three-element sequence $\mathbf{y}$ will have the attention mask shown in Table 1a.

The key idea behind PLM is to train on all $T!$ factorizations of the likelihood:

$$\log p(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T} \log p_\theta(y_{z_t}|\mathbf{y}_{\mathbf{z}_{<t}}, \mathbf{x})\right] \tag{5}$$

where $\mathcal{Z}_T$ denotes the set of all possible permutations of the index sequence [1, 2, ..., T], and z_t and $\mathbf{z}_{<t}$ denote the t-th element and the first $t-1$ elements, respectively, of a permutation $\mathbf{z} \in \mathcal{Z}_T$. Each permutation $\mathbf{z}$ specifies an ordering which corresponds to a distinct factorization of the likelihood.

To implement PLM in Transformers, we do **not** need to actually permute the text label $\mathbf{y}$. Rather, we craft the attention mask to *enforce* the ordering specified by $\mathbf{z}$. As a concrete example, shown in Table 1 are attention masks for four different permutations of a three-element sequence. Notice that while the order of the input and output sequences remains constant, all four correspond to distinct AR models specified by the given permutation or factorization order. With this in mind, it can be seen that the standard AR training is just a special case of PLM where only one permutation, $[1, 2, \ldots, T]$, is used.

In practice, we cannot train on all $T!$ factorizations due to the exponential increase in computational requirements. As a compromise, we only use K of

the possible $T!$ permutations. Instead of sampling uniformly, we choose the K permutations in a specific way. We use $K/2$ permutation pairs. The first half consists of the *left-to-right* permutation, $[1, 2, \ldots, T]$, and $K/2 - 1$ randomly sampled permutations. The other half consists of *flipped* versions of the first. We found that this sampling procedure results in a more stable training.

With K permutations and the ground truth label $\hat{\mathbf{y}}$, the full training loss is the mean of the individual cross-entropy losses for each permutation-derived attention mask $\mathbf{m}_k$:

$$\mathcal{L} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{ce}(\mathbf{y}_k, \hat{\mathbf{y}}) \tag{6}$$

where $\mathbf{y}_k = Dec(\mathbf{z}, \mathbf{p}, \mathbf{c}, \mathbf{m}_k)$. *Padding* tokens are ignored in the loss computation. More PLM details are in Appendix E.

Table 1. Illustration of AR attention masks for each permutation. The table header (with the `[B]` token) pertains to the input context, while the header column (with the `[E]` token) corresponds to the output tokens. *1* means that the output token has conditional dependency on the corresponding input token. *0* means that no information flows from input to output

(a) $[1, 2, 3]$

	`[B]`	y_1	y_2	y_3
y_1	1	0	0	0
y_2	1	1	0	0
y_3	1	1	1	0
`[E]`	1	1	1	1

(b) $[3, 2, 1]$

	`[B]`	y_1	y_2	y_3
y_1	1	0	1	1
y_2	1	0	0	1
y_3	1	0	0	0
`[E]`	1	1	1	1

(c) $[1, 3, 2]$

	`[B]`	y_1	y_2	y_3
y_1	1	0	0	0
y_2	1	1	0	1
y_3	1	1	0	0
`[E]`	1	1	1	1

(d) $[2, 3, 1]$

	`[B]`	y_1	y_2	y_3
y_1	1	0	1	1
y_2	1	0	0	0
y_3	1	0	1	0
`[E]`	1	1	1	1

3.3 Decoding Schemes

PLM training coupled with the correct parameterization allows PARSeq to be used with various decoding schemes. In this work, we only use two contrasting schemes even though more are theoretically supported. Specifically, we elaborate the use of monotonic AR and NAR decoding, as well as iterative refinement.

Autoregressive (AR) decoding generates one new token per iteration. The *left-to-right* attention mask (Table 2a) is always used. For the first iteration, the context is set to `[B]`, and only the first position query token $\mathbf{p}_1$ is used. For any succeeding iteration i, position queries $[\mathbf{p}_1, \ldots, \mathbf{p}_i]$ are used, while the context is set to the previous output, $argmax(\mathbf{y})$ prepended with `[B]`.

Non-autoregressive (NAR) decoding generates all output tokens at the same time. All position queries $[\mathbf{p}_1, \ldots, \mathbf{p}_{T+1}]$ are used but no attention mask is used (Table 2b). The context is always `[B]`.

Iterative refinement can be performed regardless of the initial decoding method (AR or NAR). The previous output (truncated at `[E]`) serves as the context for the current iteration similar to AR decoding, but all position queries $[\mathbf{p}_1, \ldots, \mathbf{p}_{T+1}]$ are always used. The *cloze* attention mask (Table 2c) is used. It is created by starting with an all-one mask, then masking out the matching token positions.

Table 2. Illustration of information flow for the different decoding schemes. Conventions follow Table 1. In NAR decoding, no mask is used; this is equivalent to using an all-one mask. "..." pertains to elements y_3 to y_{T-1}

(a) *left-to-right* AR mask

	`[B]`	y_1	y_2	...	y_T
y_1	1	0	0	0	0
y_2	1	1	0	0	0
...	1	1	1	...	0
y_T	1	1	1	1	0
`[E]`	1	1	1	1	1

(b) NAR mask

	`[B]`
y_1	1
y_2	1
...	1
y_T	1
`[E]`	1

(c) *cloze* mask

	`[B]`	y_1	y_2	...	y_T
y_1	1	0	1	1	1
y_2	1	1	0	1	1
...	1	1	1	...	1
y_T	1	1	1	1	0
`[E]`	1	1	1	1	1

4 Results and Analysis

In this section, we first discuss the experimental setup including the datasets, preprocessing methods, training and evaluation protocols, and metrics used. Next, we present our results and compare PARSeq to SOTA methods in terms of the said metrics and commonly used computational cost indicators.

4.1 Datasets

STR models are traditionally trained on large-scale synthetic datasets because of the relative scarcity of labelled real data [3]. However, in recent years, the amount of labelled real data has become sufficient for training STR models. In fact, training on real data was shown to be more sample-efficient than on synthetic data [4]. Hence, in addition to the commonly used synthetic training datasets MJSynth (MJ) [25] and SynthText (ST) [23], we also use real data for training. Specifically, we use COCO-Text (COCO) [61], RCTW17 [54], Uber-Text (Uber) [73], ArT [14], LSVT [57], MLT19 [44], and ReCTS [72]. A comprehensive discussion about these datasets is available in Baek *et al.* [4]. In addition, we also use two recent large-scale real datasets based on Open Images [30]: TextOCR [55] and annotations from the OpenVINO toolkit [31]. More details in Appendix F.

Following prior works [3], we use IIIT 5k-word (IIIT5k) [42], CUTE80 (CUTE) [49], Street View Text (SVT) [64], SVT-Perspective (SVTP) [46], ICDAR 2013 (IC13) [27], and ICDAR 2015 (IC15) [26] as the datasets for evaluation. Baek

et al. [3] provides an in-depth discussion of these datasets. We use the case-sensitive annotations of Long and Yao [39] for IIIT5k, CUTE, SVT, and SVTP. Note that IC13 and IC15 have two *versions* of their respective *test* splits commonly used in the literature—857 and 1,015 for IC13; 1,811 and 2,077 for IC15. To avoid confusion, we refer to the *benchmark* as the union of IIIT5k, CUTE, SVT, SVTP, IC13 *(1,015)*, and IC15 *(2,077)*.

These six benchmark datasets only have a total of 7,672 test samples. This amount pales in comparison to benchmark datasets used in other vision tasks such as ImageNet [18] (*classification*, 50k samples) and COCO [35] (*detection*, 40k samples). Furthermore, the said datasets largely contain horizontal text only, as shown in Figure 4a, except for SVT, SVTP, and IC15 *2,077* which contain a number of rotated text. In the real world, the conditions are less ideal, and captured text will most likely be blurry, vertically-oriented, rotated, or even occluded. In order to have a more comprehensive comparison, we also use the test sets of more recent datasets, shown in Figure 4b, such as COCO-Text (9.8k samples; low-resolution, occluded text), ArT [14] (35.1k samples; curved and rotated text), and Uber-Text [73] (80.6k samples; vertical and rotated text).

(a) Samples from the *benchmark* datasets (b) Samples from Uber, COCO, ArT

Fig. 4. Sample test images from the datasets used

4.2 Training Protocol and Model Selection

All models are trained in a mixed-precision, dual-GPU setup using PyTorch DDP for 169,680 iterations with a batch size of 384. Learning rates vary per model (Appendix G.2). The Adam [29] optimizer is used together with the 1cycle [56] learning rate scheduler. At iteration 127,260 (75% of total), Stochastic Weight Averaging (SWA) [24] is used and the 1cycle scheduler is replaced by the SWA scheduler. Validation is performed every 1,000 training steps. Since SWA averages weights at the end of each epoch, the last checkpoint at the end of training is selected. For PARSeq, $K = 6$ permutations are used (Section 4.4). A patch size of 8×4 is used for PARSeq and ViTSTR. More details are in Appendix G.

Label preprocessing is done following prior work [53]. For training, we set a maximum label length of $T = 25$, and use a charset of size $C = 94$ which contains mixed-case alphanumeric characters and punctuation marks.

Image preprocessing is done like so: images are first augmented, resized, then finally normalized to the interval $[-1, 1]$. The set of augmentation operations consists primarily of RandAugment [16] operations, excluding `Sharpness`.

`Invert` is added due to its effectiveness in house number data [15]. `GaussianBlur` and `PoissonNoise` are also used due to their effectiveness in STR data augmentation [1]. A RandAugment policy with 3 layers and a magnitude of 5 is used. RGB images are used, which are resized unconditionally to 128×32 pixels.

4.3 Evaluation Protocol and Metrics

All experiments are performed on an NVIDIA Tesla A100 GPU system. Reported mean±SD values are obtained from four replicates per model. A t-test ($\alpha = 0.05$) is used to determine if model differences are statistically-significant. There can be multiple *best* results in a column if the differences are not statistically-significant. PARSeq results are obtained from the **same** model using two different decoding schemes: PARSeq$_A$ denotes AR decoding with one refinement iteration, while PARSeq$_N$ denotes NAR decoding with two refinement iterations (ablation study in Appendix H).

Word accuracy is the primary metric for STR benchmarks. A prediction is considered correct if and only if characters at all positions match.

Charset may vary at inference time. Subsets of the training charset can be used for evaluation. Specifically, the following charsets are used: 36-character (lowercase alphanumeric), 62-character (mixed-case alphanumeric), and 94-character (mixed-case alphanumeric with punctuation). In Python, these correspond to array slices `[:36]`, `[:62]`, and `[:94]` of `string.printable`, respectively.

4.4 Ablation on training permutations vs test accuracy

As discussed in Section 3.2, training on all possible permutations is not feasible in practice due to the exponential increase in computational requirements. We instead sample a number of permutations from the pool of all possible permutations. Table 3 shows the effect of the number of training permutations on the test accuracy for all decoding schemes. With $K = 1$, only the left-to-right ordering is used and the training simplifies to the standard AR modeling. In this setup, NAR decoding does not work at all, while AR decoding works well as expected. Meanwhile, the refinement or *cloze* accuracy is at a dismal 71.14% (this is very low considering that the ground truth itself is used as the initial prediction). All decoding schemes start to perform satisfactorily only at $K >= 6$. This result shows that PLM is indeed required to achieve a unified STR model. Intuitively, NAR decoding will not work when training on just the forward and/or reverse orderings ($K <= 2$) because the variety of training contexts is insufficient. NAR decoding relies on the priors for each character which could only be sufficiently trained if all characters in the charset naturally exist as the first character of a sequence. Ultimately, $K = 6$ provides the best balance between decoding accuracy and training time. The very high cloze accuracy (∼94%) of our internal LM highlights the advantage of jointly using image features and language context for prediction refinement. After all, the primary input signal in STR is the image, not the language context.

Table 3. 94-char word accuracy (real training data) on the benchmark vs number of permutations (K) used for training PARSeq. No refinement iterations were used for both AR and NAR decoding. *cloze acc.* pertains to the word accuracy of one refinement iteration. It was measured by using the ground truth label as the initial prediction

K	AR acc.	NAR acc.	cloze acc.	Training hours
1	93.04	0.01	71.14	5.86
2	93.48	22.69	94.55	7.30
6	93.34	92.22	94.81	8.48
12	92.91	91.71	94.59	10.10
24	92.67	91.72	94.36	13.53

4.5 Comparison to state-of-the-art (SOTA)

We compare PARSeq to popular and recent SOTA methods. In addition to the published results, we reproduce a select number of methods for a fair comparison [3]. In Table 6, most reproduced methods attain higher accuracy compared to the original results. The exception is ABINet (around 1.4% decline in combined accuracy) which originally used a much longer training schedule (with pretraining of 80 and 8 epochs for LM and VM, respectively) and additional data (WikiText-103). For both synthetic and real data, PARSeq$_A$ achieves the highest word accuracies, while PARSeq$_N$ consistently places second or third. When real data is used, all reproduced models attain much higher accuracy compared to the original reported results, while PARSeq$_A$ establishes new SOTA results.

In Table 4, we show the mean accuracy for each charset. When synthetic data is used for training, there is a steep decline in accuracy from the 36- to the 62- and 94-charsets. This suggests that diversity of cased characters is lacking in the synthetic datasets. Meanwhile, PARSeq$_A$ consistently achieves the highest accuracy on all charset sizes for both synthetic and real training data. Finally in Table 5, PARSeq is the most robust against occlusion and text orientation variability. Appendix J contains more experiments on arbitrarily-oriented text. Altogether, bigger charsets and more challenging large-scale test datasets provide a more stringent evaluation and reveal wider performance gaps between methods.

Figure 5 shows the cost-quality trade-offs in terms of accuracy and commonly used cost indicators like parameter count, FLOPS, and latency. PARSeq-S is the base model used for all results, while -Ti is its scaled down variant (details in Appendix D). Note that for PARSeq, the parameter count is fixed regardless of the decoding scheme. PARSeq-S achieves the highest mean word accuracy and exhibits very competitive cost–quality characteristics across the three indicators. Compared to ABINet and TRBA, PARSeq-S uses significantly less parameters and FLOPS. In terms of latency (details in Appendix I), PARSeq-S with AR decoding is slightly slower than TRBA, but is still significantly faster than ABINet. Meanwhile, PARSeq-Ti achieves a much higher word accuracy vs CRNN in spite of similar parameter count and FLOPS. PARSeq-S is Pareto-optimal, while -Ti is a compelling alternative for low-resource applications.

Table 4. Mean word accuracy on the benchmark vs evaluation charset size

Method	Train data	36-char	62-char	94-char
CRNN	S	83.2±0.2	56.5±0.3	54.8±0.2
ViTSTR-S	S	88.6±0.0	69.5±1.0	67.7±1.0
TRBA	S	90.6±0.1	71.9±0.9	69.9±0.8
ABINet	S	89.8±0.2	68.5±1.1	66.4±1.0
PARSeq_N	S	90.7±0.2	72.5±1.1	70.5±1.1
PARSeq_A	S	**91.9**±0.2	**75.5**±0.6	**73.0**±0.7
CRNN	R	88.5±0.1	87.2±0.1	85.8±0.1
ViTSTR-S	R	94.3±0.1	92.8±0.1	91.8±0.1
TRBA	R	95.2±0.2	93.7±0.1	92.5±0.1
ABINet	R	95.2±0.1	93.7±0.1	92.4±0.1
PARSeq_N	R	95.2±0.1	93.7±0.1	92.7±0.1
PARSeq_A	R	**96.0**±0.0	**94.6**±0.0	**93.3**±0.1

Table 5. 36-char word accuracy on large-scale and more challenging datasets

Method	Train data	Test datasets and # of samples			
		ArT 35,149	COCO 9,825	Uber 80,551	Total 125,525
CRNN	S	57.3±0.1	49.3±0.6	33.1±0.3	41.1±0.3
ViTSTR-S	S	66.1±0.1	56.4±0.5	37.6±0.3	47.0±0.2
TRBA	S	68.2±0.1	61.4±0.4	38.0±0.3	48.3±0.2
ABINet	S	65.4±0.4	57.1±0.8	34.9±0.3	45.2±0.3
PARSeq_N	S	69.1±0.2	60.2±0.8	39.9±0.5	49.7±0.3
PARSeq_A	S	**70.7**±0.1	**64.0**±0.9	**42.0**±0.5	**51.8**±0.4
CRNN	R	66.8±0.2	62.2±0.3	51.0±0.2	56.3±0.2
ViTSTR-S	R	81.1±0.1	74.1±0.4	78.2±0.1	78.7±0.1
TRBA	R	82.5±0.2	77.5±0.2	81.2±0.3	81.3±0.2
ABINet	R	81.2±0.1	76.4±0.1	71.5±0.7	74.6±0.4
PARSeq_N	R	83.0±0.2	77.0±0.2	82.4±0.3	82.1±0.2
PARSeq_A	R	**84.5**±0.1	**79.8**±0.1	**84.5**±0.1	**84.1**±0.0

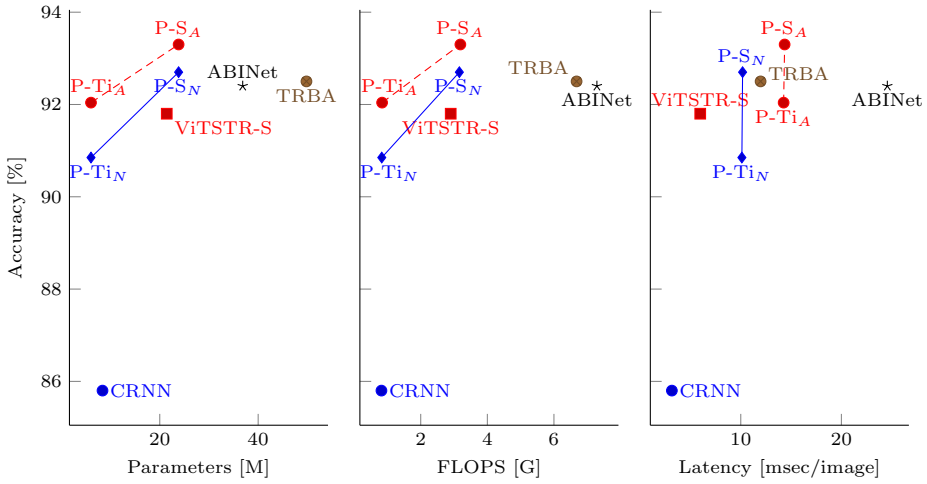

Fig. 5. 94-char mean word accuracy (real training data) vs computational cost. *P-S* and *P-Ti* are shorthands for PARSeq-S and PARSeq-Ti, respectively. For TRBA and PARSeq_A, FLOPS and latency correspond to mean values measured on the benchmark

Table 6. Word accuracy on the six benchmark datasets (36-char). For *Train data*: Synthetic datasets (**S**) - MJ [25] and ST [23]; 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 [36]. [2]SynthText without special characters (5.5M samples). [3]LM pretrained on WikiText-103 [41]. Combined accuracy values are available in Appendix K

	Method	Train data	Test datasets and # of samples							
			IIIT5k	SVT	IC13		IC15		SVTP	CUTE
			3,000	647	857	1,015	1,811	2,077	645	288
Published Results	PlugNet [43]	S	94.4	92.3	–	95.0	–	82.2	84.3	85.0
	SRN [69]	S	94.8	91.5	95.5	–	82.7	–	85.1	87.8
	RobustScanner [70]	S,B	95.4	89.3	–	94.1	–	79.2	82.9	92.4
	TextScanner [62]	S*	95.7	92.7	–	94.9	–	83.5	84.8	91.6
	AutoSTR [71]	S	94.7	90.9	–	94.2	81.8	–	81.7	–
	RCEED [17]	S,B	94.9	91.8	–	–	–	82.2	83.6	91.7
	PREN2D [67]	S	95.6	94.0	96.4	–	83.0	–	87.6	91.7
	VisionLAN [66]	S	95.8	91.7	95.7	–	83.7	–	86.0	88.5
	Bhunia *et al.* [8]	S	95.2	92.2	–	95.5	–	84.0	85.7	89.7
	CVAE-Feed.[1] [7]	S	95.2	–	–	95.7	–	84.6	88.9	89.7
	STN-CSTR [11]	S	94.2	92.3	96.3	94.1	86.1	82.0	86.2	–
	CRNN [4]	S	84.3	78.9	–	88.8	–	61.5	64.8	61.3
	ViTSTR-B [2]	S[2]	88.4	87.7	93.2	92.4	78.5	72.6	81.8	81.3
	TRBA [4]	S	92.1	88.9	–	93.1	–	74.7	79.5	78.2
	ABINet [21]	S[3]	96.2	93.5	97.4	–	86.0	–	89.3	89.2
Experiments	CRNN	S	91.2±0.2	85.7±0.7	92.1±0.7	90.9±0.5	74.4±1.0	70.8±0.9	73.5±0.6	78.7±0.7
	ViTSTR-S	S	94.0±0.2	91.7±0.4	95.1±0.7	94.2±0.7	82.7±0.1	78.7±0.1	83.9±0.6	88.2±0.6
	TRBA	S	96.3±0.2	**92.8**±0.9	96.3±0.3	95.0±0.4	84.3±0.1	80.6±0.2	86.9±1.3	91.3±1.6
	ABINet	S	95.3±0.2	**93.4**±0.2	**97.1**±0.4	95.0±0.3	83.1±0.3	79.1±0.2	87.1±0.6	89.7±2.3
	PARSeq_N (Ours)	S	95.7±0.2	92.6±0.3	96.3±0.4	95.5±0.6	85.1±0.1	81.4±0.1	87.9±0.9	91.4±1.5
	PARSeq_A (Ours)	S	**97.0**±0.2	**93.6**±0.4	**97.0**±0.3	**96.2**±0.4	**86.5**±0.2	**82.9**±0.2	**88.9**±0.9	**92.2**±1.2
	CRNN	R	94.6±0.2	90.7±0.4	94.1±0.4	94.5±0.3	82.0±0.2	78.5±0.2	80.6±0.3	89.1±0.4
	ViTSTR-S	R	98.1±0.2	95.8±0.4	97.6±0.3	97.7±0.3	88.4±0.4	87.1±0.3	91.4±0.2	96.1±0.4
	TRBA	R	98.6±0.1	97.0±0.2	97.6±0.3	97.6±0.2	89.8±0.4	88.7±0.4	93.7±0.3	**97.7**±0.2
	ABINet	R	98.6±0.2	**97.8**±0.3	**98.0**±0.4	97.8±0.2	90.2±0.2	88.5±0.2	93.9±0.8	**97.7**±0.7
	PARSeq_N (Ours)	R	98.3±0.1	**97.5**±0.4	**98.0**±0.1	**98.1**±0.1	89.6±0.2	88.4±0.4	94.6±1.0	**97.7**±0.9
	PARSeq_A (Ours)	R	**99.1**±0.1	**97.9**±0.2	**98.3**±0.2	**98.4**±0.2	**90.7**±0.3	**89.6**±0.3	**95.7**±0.9	**98.3**±0.6

5 Conclusion

We adapted PLM for STR in order to learn PARSeq, a unified STR model capable of context-free and -aware decoding, and iterative refinement. PARSeq achieves SOTA results in different charset sizes and real-world datasets by jointly conditioning on both image and text representations. By unifying different decoding schemes into a single model and taking advantage of the parallel computations in Transformers, PARSeq is optimal on accuracy vs parameter count, FLOPS, and latency. Due to its extensive use of *attention*, it also demonstrates robustness on vertical and rotated text common in many real-world images.

Acknowledgments. This work was funded in part by CHED-PCARI IIID-2016-005 (Project AIRSCAN). We are also grateful to the PCARI PRIME team, led by Roel Ocampo, who ensured the uptime of our GPU servers.

References

1. Atienza, R.: Data augmentation for scene text recognition. In: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). pp. 1561–1570 (2021). https://doi.org/10.1109/ICCVW54120.2021.00181
2. Atienza, R.: Vision transformer for fast and efficient scene text recognition. In: International Conference on Document Analysis and Recognition (ICDAR) (2021)
3. Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S.J., Lee, H.: What is wrong with scene text recognition model comparisons? dataset and model analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (10 2019)
4. 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)
5. Baevski, A., Auli, M.: Adaptive input representations for neural language modeling. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=ByxZX20qFQ
6. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
7. 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)
8. 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)
9. Bleeker, M., de Rijke, M.: Bidirectional scene text recognition with a single decoder. In: ECAI 2020, pp. 2664–2671. IOS Press (2020)
10. Borisyuk, F., Gordo, A., Sivakumar, V.: Rosetta: Large scale system for text detection and recognition in images. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 71–79 (2018)
11. Cai, H., Sun, J., Xiong, Y.: Revisiting classification perspective on scene text recognition (2021), https://arxiv.org/abs/2102.10884
12. Chen, X., Jin, L., Zhu, Y., Luo, C., Wang, T.: Text recognition in the wild: A survey. ACM Computing Surveys (CSUR) **54**(2), 1–35 (2021)
13. 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)
14. Chng, C.K., Liu, Y., Sun, Y., Ng, C.C., Luo, C., Ni, Z., Fang, C., Zhang, S., Han, J., Ding, E., et al.: Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1571–1576. IEEE (2019)
15. Cubuk, E.D., Zoph, B., Mané, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 113–123 (2019). https://doi.org/10.1109/CVPR.2019.00020

16. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 702–703 (2020)
17. 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)
18. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423
20. 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)
21. 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)
22. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. pp. 369–376 (2006)
23. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
24. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.: Averaging weights leads to wider optima and better generalization. In: Silva, R., Globerson, A., Globerson, A. (eds.) 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018. pp. 876–885. 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, Association For Uncertainty in Artificial Intelligence (AUAI) (2018)
25. 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)
26. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., et al.: Icdar 2015 competition on robust reading. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR). pp. 1156–1160. IEEE (2015)
27. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., i Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., De Las Heras, L.P.: Icdar 2013 robust reading competition. In: 2013 12th International Conference on Document Analysis and Recognition. pp. 1484–1493. IEEE (2013)

28. Kasai, J., Pappas, N., Peng, H., Cross, J., Smith, N.: Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=KpfasTaLUpq
29. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
30. Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Kamali, S., Malloci, M., Pont-Tuset, J., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., Murphy, K.: Openimages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://github.com/openimages (2017), https://storage.googleapis.com/openimages/web/index.html
31. Krylov, I., Nosov, S., Sovrasov, V.: Open images v5 text annotation and yet another mask text spotter. In: Balasubramanian, V.N., Tsang, I. (eds.) Proceedings of The 13th Asian Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 157, pp. 379–389. PMLR (17–19 Nov 2021), https://proceedings.mlr.press/v157/krylov21a.html
32. Lee, C.Y., Osindero, S.: Recursive recurrent nets with attention modeling for ocr in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (6 2016)
33. Lee, J., Park, S., Baek, J., Oh, S.J., Kim, S., Lee, H.: On recognizing texts of arbitrary shapes with 2d self-attention. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 546–547 (2020)
34. Liao, Y., Jiang, X., Liu, Q.: Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 263–274 (2020)
35. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
36. 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)
37. Liu, W., Chen, C., Wong, K.Y.K., Su, Z., Han, J.: Star-net: a spatial attention residue network for scene text recognition. In: BMVC. vol. 2, p. 7 (2016)
38. Long, S., He, X., Yao, C.: Scene text detection and recognition: The deep learning era. International Journal of Computer Vision **129**(1), 161–184 (2021)
39. 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)
40. Mansimov, E., Wang, A., Welleck, S., Cho, K.: A generalized framework of sequence generation with application to undirected sequence models. arXiv preprint arXiv:1905.12790 (2019)
41. 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
42. Mishra, A., Alahari, K., Jawahar, C.: Scene text recognition using higher order language priors. In: BMVC-British Machine Vision Conference. BMVA (2012)

43. 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)
44. Nayef, N., Patel, Y., Busta, M., Chowdhury, P.N., Karatzas, D., Khlif, W., Matas, J., Pal, U., Burie, J.C., Liu, C.l., et al.: Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition—rrc-mlt-2019. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1582–1587. IEEE (2019)
45. Nguyen, N., Nguyen, T., Tran, V., Tran, M.T., Ngo, T.D., Nguyen, T.H., Hoai, M.: Dictionary-guided scene text recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7383–7392 (6 2021)
46. 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)
47. Qi, W., Gong, Y., Jiao, J., Yan, Y., Chen, W., Liu, D., Tang, K., Li, H., Chen, J., Zhang, R., et al.: Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining. In: International Conference on Machine Learning. pp. 8630–8639. PMLR (2021)
48. Qiao, Z., Zhou, Y., Yang, D., Zhou, Y., Wang, W.: Seed: Semantics enhanced encoder-decoder framework for scene text recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (6 2020)
49. Risnumawan, A., Shivakumara, P., Chan, C.S., Tan, C.L.: A robust arbitrary text detection system for natural scene images. Expert Systems with Applications **41**(18), 8027–8048 (2014)
50. Sheng, F., Chen, Z., Xu, B.: Nrtr: A no-recurrence sequence-to-sequence model for scene text recognition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 781–786. IEEE (2019)
51. 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)
52. 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)
53. Shi, B., Yang, M., Wang, X., Lyu, P., Yao, C., Bai, X.: Aster: An attentional scene text recognizer with flexible rectification. IEEE transactions on pattern analysis and machine intelligence **41**(9), 2035–2048 (2018)
54. Shi, B., Yao, C., Liao, M., Yang, M., Xu, P., Cui, L., Belongie, S., Lu, S., Bai, X.: Icdar2017 competition on reading chinese text in the wild (rctw-17). In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 1429–1434. IEEE (2017)
55. Singh, A., Pang, G., Toh, M., Huang, J., Galuba, W., Hassner, T.: Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8802–8812 (2021)
56. Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications. vol. 11006, p. 1100612. International Society for Optics and Photonics (2019)

57. 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)
58. Tian, C., Wang, Y., Cheng, H., Lian, Y., Zhang, Z.: Train once, and decode as you like. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 280–293 (2020)
59. Uria, B., Murray, I., Larochelle, H.: A deep and tractable density estimator. In: International Conference on Machine Learning. pp. 467–475. PMLR (2014)
60. 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)
61. Veit, A., Matera, T., Neumann, L., Matas, J., Belongie, S.: Coco-text: Dataset and benchmark for text detection and recognition in natural images. In: arXiv preprint arXiv:1601.07140 (2016), `http://vision.cornell.edu/se3/wp-content/uploads/2016/01/1601.07140v1.pdf`
62. 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)
63. Wang, J., Hu, X.: Gated recurrent convolution neural network for ocr. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 334–343 (2017)
64. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: 2011 International Conference on Computer Vision. pp. 1457–1464. IEEE (2011)
65. 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)
66. 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)
67. 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)
68. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems **32** (2019)
69. 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)
70. 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)
71. 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)

72. Zhang, R., Zhou, Y., Jiang, Q., Song, Q., Li, N., Zhou, K., Wang, L., Wang, D., Liao, M., Yang, M., et al.: Icdar 2019 robust reading challenge on reading chinese text on signboard. In: 2019 international conference on document analysis and recognition (ICDAR). pp. 1577–1581. IEEE (2019)
73. Zhang, Y., Gueguen, L., Zharkov, I., Zhang, P., Seifert, K., Kadlec, B.: Uber-text: A large-scale dataset for optical character recognition from street-level imagery. In: SUNw: Scene Understanding Workshop - CVPR 2017. Hawaii, U.S.A. (2017), http://sunw.csail.mit.edu/abstract/uberText.pdf