Bidirectional Scene Text Recognition with a Single Decoder

Maurits Bleeker[1] and **Maarten de Rijke**[2]

Abstract. Scene Text Recognition (STR) is the problem of recognizing the correct word or character sequence in a cropped word image. To obtain more robust output sequences, the notion of bidirectional STR has been introduced. So far, bidirectional STRs have been implemented by using two separate decoders; one for left-to-right decoding and one for right-to-left. Having two separate decoders for almost the same task with the same output space is undesirable from a computational and optimization point of view. We introduce the Bidirectional Scene Text Transformer (Bi-STET), a novel bidirectional STR method with a single decoder for bidirectional text decoding. With its single decoder, Bi-STET outperforms methods that apply bidirectional decoding by using two separate decoders while also being more efficient than those methods, Furthermore, we achieve or beat state-of-the-art (SOTA) methods on all STR benchmarks with Bi-STET. Finally, we provide analyzes and insights into the performance of Bi-STET.

1 INTRODUCTION

Scene Text Recognition (STR) is the task of recognizing the correct word or character sequence in a cropped word image. Many different architectures have been proposed for STR. Since the rise of deep learning, most state-of-the-art STR methods adopt a Convolutional Neural Network (CNN) for feature extraction and an encoder-decoder architecture as the core component for sequence modeling. After feature extraction with a CNN, the extracted features of the input image are encoded into a new representation with an encoder. As a final step, conditioned on the encoded input image representation, the character sequence is decoded, which is depicted in the input image. Figure 1 summarizes a general pipeline. Baek et al. [1] identify sequence modeling as a core component in STR frameworks.

The encoder-decoder architecture for the sequence modeling stage of many state-of-the-art STR methods (see Section 2) can be characterized by (i) a bidirectional Recurrent Neural Network (RNN) for feature encoding, (ii) a directed RNN decoder for character decoding, and (iii) various types of attention mechanism [2, 23] to generate additional context vectors for the current decoding steps.

Two recent developments have accelerated progress in STR: (i) a move away from recurrent sequence modeling, and (ii) bidirectional decoding for STR.

Regarding the first, Sheng et al. [28] have changed the standard STR approach for sequence modeling by introducing a non-recurrent method based on a transformer encoder-decoder architecture [35]. By using a transformer-based encoder-decoder, the model architecture can be simplified and the time for model optimization can be reduced by an order of magnitude in comparison [28].

The second reason for recent progress in STR is bidirectional decoding. Bidirectional decoding is the idea of decoding an output sequence in two directions (i.e., from left-to-right and right-to-left) for more robust output predictions. This bidirectional decoding is implemented by using a different decoder for each decoding direction [31]. Decoding the text in two directions at the same time can be seen as two different sub-tasks for the model to perform.

It is important to reflect on different ways of modeling sub-tasks, especially using task conditioning. With task conditioning, the output of a method does not solely depend on the input data, but on a given (sub-)task as well. In other words, given the same input data, the output may be different based on the (sub-)task it is conditioned on. As explained by Radford et al. [26], task conditioning can be implemented in several ways.

One option is at the *algorithmic* level [7], where different models are learned for different tasks, and an overall algorithm selects the correct model for a particular task. Implementing task conditioning at the algorithmic level is not optimal, since in most cases, there is a lot of shared knowledge between different (sub-)tasks, which is not exploited when separate models are optimized for each task. Another way of implementing task conditioning is at the *architecture* level. For bidirectional STR, Shi et al. [31] have implemented the decoding direction as two sub-tasks at the architecture level, that is, by having two separate decoders: one for left-to-right and another one for right-to-left text decoding. Although both decoders share the same encoder, two separate decoders are optimized for two tasks that are (almost) identical and share the same output space.

Implementing bidirectional decoding at the architecture level is not uncommon, and has been done for other tasks besides STR [43, 44]. However, having two separate decoders for two tasks that are similar (i.e., left-to-right and right-to-left STR) is not desirable:

(1) From a computational point of view: The two network components do not share weights, which requires separate optimization for both parts.
(2) From a multi-task learning point of view: There is a lot of shared knowledge between left-to-right and right-to-left decoding and both tasks share the same output space, which is not utilized when optimizing the two decoders apart from each other.

Therefore, the question remains: *Can we have the benefits of bidirectional decoding (left-to-right and right-to-left decoding) for STR without implementing this at the architecture or algorithmic level?*

There is promising room for improvement on the implementation side of bidirectional decoding for STR by just using one decoder for both decoding directions. Instead of implementing this decoding direction at the algorithmic or at the architecture level, we propose a new way of implementing task conditioning, namely, at the *input*

[1] University of Amsterdam, The Netherlands, email: m.j.r.bleeker@uva.nl
[2] University of Amsterdam, The Netherlands, email: derijke@uva.nl

level. Implementing task conditioning at the input level means that extra feature information is added to the input of the model. This context information should be exploited by the model so as to condition on the right (sub-)task.

More specifically, the transformer architecture, as used by Sheng et al. [28] for the encoder-decoder part for STR, has no recurrent inductive bias. To solve sequential problems with transformers at the input level, additional position embeddings are added to provide the model with information about the order of the input sequence. Due to the "position unawareness" of the transformer, the model is also not limited to an inductive decoding direction (unlike RNNs). By adding an extra embedding to the input data, which tells the method to decode an input example from left-to-right or right-to-left, the model can exploit bidirectional decoding with one unified architecture for both directions. This means that the model has one decoder with one set of model parameters that can be optimized for both subtasks at the same time. This is in contrast with the method by Shi et al. [31], where two decoders are optimized, one for each decoding direction.

In this work, we show that we can simplify the bidirectional STR architecture by using a transformer based encoder-decoder which is able to perform bidirectional text recognition by using a single decoder. Our main technical contributions in this paper are the following:

- We introduce Bi-STET, **BI**directional **S**cene **TE**xt **T**ransformer. Bi-STET is a unified network architecture, optimized for two subtasks (left-to-right and right-to-left STR), using one forward pass. We achieve this through the implementation of bidirectional decoding at the input level as opposed to previous works which do this at the architecture level [31]. We condition the output sequence on a specific decoding direction by adding extra features at the input level, which results in a direction-agnostic decoder architecture. It is possible to exploit the transformer architecture for task conditioning at the input level, without requiring additional model components or algorithms to model this task conditioning.
- We show that Bi-STET achieves or outperforms state-of-the-art STR methods with a simpler and more efficient approach than other bidirectional STR methods. We achieve these similar results with fewer weight parameters and 50% less training iterations.
- We provide analyzes and insights on the performance of Bi-STET.[3] We analyze the generalisation of Bi-STET w.r.t. oriented and curved text, the learned attention mechanism of the encoder-decoder, the relation between sequence length and test-accuracy of Bi-STET and is Bi-STET makes more mistakes for rarer expressions than with dictionary words.

2 RELATED WORK

Traditional methods for STR [3, 32] apply a bottom-up approach. The input image is preprocessed for feature extraction and character segmentation is applied to obtain single characters from the input image for word inference. For an overview, see [45]. With the rise of deep learning, STR methods increasingly focus on end-to-end training from the input image to the desired output character sequence.

2.1 Deep-learning based text recognition

Jaderberg et al. [13] have proposed the first method for unconstrained STR with deep learning. The method predicts a sequence of characters with a fixed length by using a CNN classification model. A bag-of-N-grams is also predicted; representing an unordered set of character N-grams that occur in the word depicted in the input image. The predictions are combined in a Path Select Layer to predict the most likely character sequence. More recently, Jaderberg et al. [15] propose another method where the recognition task is formulated as a multi-class classification problem over a 90k-class lexicon.

Many other STR methods [4, 29, 30, 31, 39, 42] use a CNN for feature extraction in combination with an encoder-decoder model to map the sequence of image features to a character sequence. To solve the problem of rotation invariance of CNNs (i.e., for curved and oriented text), several methods have been proposed [29, 31, 39, 40, 42]. Shi et al. [29, 31] and Yang et al. [39] use a Spatial Transformer Network (STN) for input image rectification to handle perspective text and curved text. Zhan and Lu [42] have introduced an iterative Rectification Network where the input image is rectified multiple times by removing perspective distortion and text line curvature.

The alignment between the predicted character and the corresponding region in the input image is modelled with two different approaches. The first approach is to use CTC loss [8, 10, 21, 30, 34]. The other is to connect the RNN encoder to the decoder via an attention mechanism [4, 29, 31, 40, 42], which creates an additional context vector for the encoded input image conditioned on the already predicted output sequence.

Shi et al. [31] introduce the notion of bidirectional STR. Each output sequence is predicted in two directions with two separate decoders, which do not share parameters. The output sequence with the highest probability is selected as the final prediction in order to obtain more robust predictions. Sheng et al. [28] are the first to use a non-recurrent encoder-decoder approach based on a transformer architecture; they have also introduced a modality-transform block to map an image to a sequence feature representation.

In contrast to most work in STR, we do not use any specific component for image rectification. We also do not rely on RNNs for sequence modeling. Similar to Sheng et al. [28], we also use a transformer architecture which yields state-of-the-art results in text recognition without using extra image rectification components, for bidirectional sequence modelling STR. However, we achieve this by using a single decoder for the bidirectional decoding, resulting in significantly fewer parameters and training iterations.

2.2 Task conditioning

As indicated by Radford et al. [26], the distribution over the possible model outputs is naturally modelled as $p(y \mid x)$, where x is the input data, and y is a possible output prediction. With *task conditioning* (or modality conditioning), the output is not only conditioned on the input data, but also on a given task, dataset, or modality as well, i.e., $p(y \mid x, t)$, where t stands for the *task*.

One way of implementing task conditioning is at the *architecture* level. Kaiser et al. [16] introduce a single model for eight tasks; for each data-modality and/or subtask, different encoders and decoders are used with a shared latent space. Devlin et al. [6] adopt the transformer to train a task-agnostic language model. After training the language model, additional task-specific output layers are fined-tuned for each evaluation task. Finn et al. [7] introduce a meta-learning framework for multi-task learning where task conditioning is implemented at the *algorithmic* level: tasks are treated as training examples and are sampled from a distribution over tasks during training. During each training iteration, for each task, a separate model for each task is updated based on the loss for that specific task.

[3] For reproducibility and repeatability, the code and checkpoint files used to train and evaluate Bi-STET will be made available at `https://github.com/MauritsBleeker/Bi-STET`.

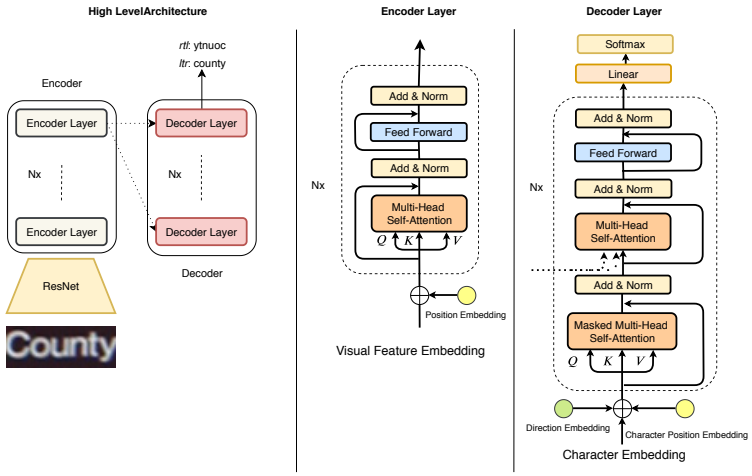

Figure 1: Overview of Bi-STET. A ResNet architecture is used for visual feature extraction. Next, a stack of n transformer encoder layers is used for encoding the visual image features. For decoding the output sequence, a stack of n decoder transformer layers is used.

Unlike previous work, we condition a subtask (i.e., the decoding direction) at the input level. As a result, we do not need different models or network components for each decoding direction. Having only one decoder is desirable from both a computational point of view (i.e., only one decoder to optimize) and from an optimization point of view (i.e., shared weights for all sub-tasks).

3 METHOD

To address the STR task, we take a fixed size image $\mathcal{I}$ as input and want to decode the sequence of output characters $y_1, \ldots, y_L$, where L is the length of the character sequence depicted in the input image. Briefly, we use a multi-layer stack of transformers for both the encoder and decoder. We use exactly the same implementation of the transformer as described in [35] for the encoder-decoder. Therefore, we refer to [35] for the exact details of the implementation. A full overview of Bi-STET is shown in Figure 1.

3.1 Visual feature extraction network

Like [4, 31, 39, 42], we use a ResNet [12] based architecture for the Visual Feature Extraction Network (VFEN). A ResNet architecture is a more suitable feature extractor than VGG [33] for STR, as shown in [31, 42]. We use a 45-layer residual network, with the same network configuration as [31]. We split the obtained feature representation $\mathcal{Q} \in \mathbb{R}^{W \times C \times H}$ column-wise, which results in a sequence of Visual Feature Embeddings (VFEs), $\mathbf{v}_1, \ldots, \mathbf{v}_W$, where $\mathbf{v}_i \in \mathbb{R}^{C \times H}$.

3.2 Feature encoding

The feature encoder is in charge of encoding the visual image embeddings. Each visual image embedding is encoded into a new representation in n steps, by using transformer encoder layers, while attending over the entire sequence of VFEs during each encoding step. We use scaled dot-product as the attention function:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}. \quad (1)$$

Scaled dot-product attention can be described as a weighted sum of the vectors in matrix $\mathbf{V}$, which is a horizontal concatenation of the flattened sequence of VFEs (also referred to as *values*). Each embedding $\mathbf{v}$ is weighted by the similarity between a key $\mathbf{k}$ and a query $\mathbf{q}$. In each transformer layer multiple heads of attention are used:

$$head_i = \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right) \quad (2)$$

and

$$MultiHeadSelfAttention = ConCat(head_1, \ldots, head_h)\mathbf{W}^O, \quad (3)$$

An advantage of using multiple attention heads is that it allows the model to learn to attend over different positions in the input image per attention head during each step of the encoding process. For self-attention during encoding, the matrices $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are consistent per layer (i.e., $\mathbf{Q} = \mathbf{K} = \mathbf{V}$) and obtained from the output of the previous layer. In the first layer, they can be obtained from the VFEN.

The weights of each transformer layer are not shared between encoder layers. We apply the positional encoding as introduced in [35].

3.3 Character decoding

The decoder consists of n transformer decoder layers. For both the encoder and the decoder we use a transformer architecture. The reason to choose a transformer over an RNN-based architecture is that an RNN already has an inductive bias in terms of decoding and encoding direction due to the recurrent nature of the architecture. The decoder takes the embeddings of the decoded output character sequence as input. Each decoder layer consists of three sublayers: two multi-head attention layers and one feed-forward neural network (the same implementation as in Section 3.2). The first multi-head attention layer attends over the decoded output characters (*decoder self-attention*). The second layer of multi-head attention (*decoder cross-attention*) attends over the encoded VFEs from the last encoder layer. The decoder cross-attention is able to look at the encoded input image at every step during decoding. This makes it possible to attend over different encoded image regions during decoding. Previous work on STR [4, 31, 42] only uses one attention distribution over the encoded states per decoding step. In contrast, per decoding layer n, we have h attention heads modeling complex alignments between encoder features and decoded output characters.

We add an extra direction embedding in order to add more context information by using additional embeddings. This direction embedding tells the model to decode the output sequence from left-to-right

or from right-to-left. By adding the direction embedding, we can use the same decoder network and still condition on the output sequence reading direction.

For every decoding step t, the output embedding $\mathbf{h}_n$ of the stack of transformer decoders is passed through a feed-forward layer with the output characters as the output space. A softmax is applied to obtain a distribution over all output characters. During training, this results in a $V \times L$ matrix, where V is the size of the output character space (or vocabulary) and L the length of the predicted character sequence.

3.4 Direction embedding

We define the decoding direction of the output sequence as two sub-tasks of STR. Each decoding direction is one sub-task of the method on which we condition the output sequence. To condition the output on a decoding direction, we randomly initialize two 512-d vectors at the start of training. During each training iteration, every input image in the batch is decoded twice; once from left-to-right and right-to-left. The ground truth description of the right-to-left decoded character sequence is just the reserved ground truth of the original description. During decoding, we add the direction embedding on top of the positional embedding and the token embedding. This is another way to provide additional context information to the model, similar to the position embeddings. Based on this information, the model should learn to decode the character sequence not only in the left-to-right direction but also in the other direction, otherwise the loss function for the right-to-left decoded images will not be minimized.

Similar to the character embeddings, the direction embeddings are trained end-to-end with the rest of the model.

4 EXPERIMENTAL SETUP

4.1 Datasets

Bi-STET is trained on two synthetically generated datasets. After training, the method is evaluated on seven real-word evaluation sets which are commonly used for scene text recognition.

4.1.1 Training datasets

- *Synth90K.* The *Synth90K dataset* [14] is a synthetically generated dataset for text recognition. It contains 7.2 million training images. The lexicon used contains 90,000 words. Each word has been used to render 100 different synthetic images.
- *SynthText.* The *SynthText dataset* [11] contains 800,000 synthetically generated images for text detection and recognition, with roughly 8 million annotated text instances placed in natural scenes. We crop all text instances from the original input images, by taking the smallest horizontally aligned bounding box around the annotated text instances in the image. We discard bounding boxes that are smaller than 32 pixels in height or 30 pixels in width. Bounding boxes larger than 800 pixels in width, 500 pixels in height or with a transcription label longer than 25 characters are removed too. We obtain 2.9 million cropped-word images from this dataset for training.

4.1.2 Evaluation datasets

Bi-STET is evaluated cross-dataset. The model is trained only on synthetically generated word images while we evaluate on real-world word images. Unless stated otherwise, we use the word-image crops and annotations as provided by the dataset to be consistent with other methods. This might result in over-cropped word images; in other cases adding a margin may lead other artifacts.

- *ICDAR03.* The *ICDAR03 dataset* [22] contains 258 images for training and 251 for testing. For the text recognition task only, 1,156 word instances can be cropped from the test set. This dataset was collected for the text detection and recognition task. Therefore, most text instances in the images are clearly horizontally visible and centred in the image [36]. Following [4, 31, 37, 38], we ignore all words that are shorter than three characters or contain non-alpha numeric characters during evaluation.
- *ICDAR13.* The *ICDAR13 dataset* [17] contains most images from the ICDAR03 dataset. In total this dataset contains 1,095 word images for evaluation. Similar to [31], we add a cropping-margin of 15% to prevent over-cropping.
- *ICDAR15.* The *ICDAR15 dataset* [18] contains 2,077 word images for evaluating. The word images are cropped from video frames collected with the Google Glass device. These frame crops contain substantial real-world interference factors such as: occlusions, motion blur, noise, and illumination factors, which are not present in the ICDAR03 and ICDAR13 datasets. Like Cheng et al. [4], we remove all examples where the ground truth transcription contains non-alpha numeric characters.
- *SVT.* The *Street View Text dataset* (SVT) [37, 38] contains images that have been taken from Google Street View. Due to this origin, some images have a low resolution and/or contain distortion factors such as noise or blur. This dataset contains 647 word images for evaluation. Per image, a 50-word lexicon is provided as well. Similar to [31], we add a cropping-margin of 5% to prevent over-cropping. Similar to [31], we add a cropping-margin of 5% to prevent over-cropping.
- *SVTP.* The *Street View Text Perspective dataset* (SVTP) [25] contains 645 word images cropped from Street View. Most images have perspective distortions due to the camera viewpoint angle.
- *IIIT-5K Word.* The *IIIT-5k Word dataset* (IIIT5K) [24] contains 3,000 images for evaluation. The word images are cropped from scene texts and born-digital images. For this dataset, per evaluation image, two lexicons of 50 and 1,000 words are provided for lexicon inference.
- *CUTE80.* The *Curved Text dataset* (CUTE80) [27] mainly contains curved and/or oriented text instances. The dataset was originally proposed for text detection, but later annotated for text recognition as well. In total, 288 high resolution word images can be cropped from the original dataset.

4.2 Implementation details

Our implementation consists of a feature extraction network followed by an encoder-decoder network. The code and checkpoint files used to train and evaluate Bi-STET are available at `https://github.com/MauritsBleeker/Bi-STET`.

4.2.1 Feature extraction network

All input images are resized to 32×256 without keeping the original aspect ratio. The maximum output sequence length during training is 24. All pixels are normalized with a per-channel calculated mean and standard deviation calculated on the ImageNet dataset [5].

4.2.2 Encoder-decoder network

For the encoder and decoder we use exactly the same configuration as the base model described in [35]. We use a stack of $n = 6$ trans-

Table 1: Accuracy of left-to-right vs. right-to-left vs. bidirectional word decoding. Measured without lexicon and compared with the method by Shi et al. [31].

Method	IIIT5k	SVT	IC03	IC13	IC15	SVTP	CUTE
Shi et al. [31], left-to-right	91.93	88.76	93.49	89.75	–	74.11	73.26
Shi et al. [31], right-to-left	91.43	89.96	92.79	89.95	–	73.95	74.31
Shi et al. [31], bidirectional	92.27	**89.5**	93.60	90.54	–	74.26	74.31
Bi-STET (this paper), left-to-right	94.2	88.3	95.1	92.5	75.0	78.8	81.8
Bi-STET (this paper), right-to-left	94.1	87.9	95.3	**93.4**	73.2	79.5	**83.6**
Bi-STET (this paper), bidirectional	**94.7**	89.0	**96.0**	**93.4**	**75.7**	**80.6**	82.5

former layers for both the encoder and decoder. Each layer has eight attentions heads ($h = 8$). The embedding dimensionality is set to $d = 512$. For the hidden state of the two layer feed-forward network in each transformer layer, we set $d_f = 2048$.

The output space of our model contains all the lower-case characters $\{a, \ldots, z\}$, digits $\{0, \ldots, 9\}$, 32 ASCII punctuation marks, similar to [4, 31, 42], and a start- and end-of-word symbol. The punctuation marks are included during training, but ignored during evaluation. All evaluation and training ground truth descriptions are lower-case, which makes the model case-insensitive.

4.3 Optimization

The entire method is trained from scratch. All the weights are initialized with Xavier initialization [9]. Similar to [4, 29, 30, 31, 40, 42], we use ADADELTA [41] as the optimizer for the model. ADADELTA has a self-adaptable learning rate, which we initialize to 1 Even though the learning rate of ADADELTA is self-adaptable, we apply a learning rate schedule where we reduce the initial learning rate by a factor of 0.1 after 150,000, 300,000 and 400,000 training iterations. Similar to [31], we find that a learning rate schedule is beneficial to the performance.

The model is trained for 500,000 training iterations in total, after which it converges. We use Kullback-Leibler divergence as the loss function. The batch size is set to 64. For each training batch we sample 32 images from the Synth90k dataset and 32 from the SynthText. Shi et al. [31] and Zhan and Lu [42] show that methods optimized with balanced batch (of size 64) on the SynthText and Synth90k datasets outperform methods solely trained on Synth90k. Per forward-backward pass, we decode the characters per example from left-to-right and from right-to-left.

During training, we do one forward pass for left-to-right decoding and one for right-to-left and accumulate the gradients. It is possible to train both decoding directions with one forward pass, but for computational reasons we have chosen gradient accumulation instead.

4.4 Metrics

We use the same evaluation metrics as in [4, 31, 42]. The text recognition task includes 68 characters in total. During evaluation the 32 ASCII punctuation marks are ignored. When a lexicon is provided, the word from the lexicon with the shortest edit distance is selected as the prediction. Only predicted sequences of characters that are completely correct are considered to be correctly predicted examples. We select the character with the highest probability per index in the sequence, until the end-of-word character is predicted. When decoding bidirectionally, the sequence with the highest product probability is selected as final output sequence.

5 RESULTS

First, we compare the bidirectional sequence predicting for STR with a single decoder vs. with two decoders. Next, we examine the performance of Bi-STET and other models on STR evaluation sets. Finally, we provide analyzes of the attention mechanism in Bi-STET and the capability of the method to handle curved and rotated text.

5.1 Bidirectional decoding

Similar to [31], we condition the output character sequence on a decoding direction. We validate our universal bidirectional decoding with three evaluation variants, similar to Shi et al. [31]. For the first variant, we decode the output sequence from left-to-right, by only using the left-to-right direction embedding. In the second variant, we only use the right-to-left directional embedding. In the third variant, we decode each evaluation example twice, once with each direction embedding. The two predicted outputs can have different sequence lengths. For each prediction (left-to-right and right-to-left) we take the sequence with the highest probability by taking the arg-max for each position and take the product of the probabilities as the probability of the entire sequence. We select the sequence with the highest output probability as the final prediction. In case that the right-to-left prediction has the highest probability, we reverse the sequence to match with the ground truth.

In Table 1 we show the results of the three aforementioned evaluation variants and the results obtained by Shi et al. [31]. For 6 out of 7 evaluation sets, we achieve state-of-the-art results for bidirectional STR. For 6 out of 7 of the evaluation sets, Bi-STETs bidirectional decoding leads to higher scoring sequence prediction than using a single decoding direction. Only for the CUTE80 set, the right-to-left decoding leads to a higher accuracy score than bidirectional. The gain in performance due to the bidirectional decoding is similar as in the method by Shi et al. [31].

We also show that, by using a transformer-based encoder-decoder, bidirectional STR can be substantially simplified in comparison to the method by [31]. In Table 3, we compare the number of model parameters and the number of training iterations with the method by Shi et al. [31]. We use similar training settings, in terms of batch size, optimization, data, etc. as [31]. Based on Table 3, it is clear that with a single bidirectional transformer decoder, the number of training iterations can be reduced by 50% compared to methods that use two separate RNN decoders – in combination with significantly fewer model parameters. Fewer training iterations and model parameters are excellent properties from an efficiency and computational point of view. We also outperform the RNN based method, which also uses an extra image rectification network, on most evaluation sets.

To summarize, the results in Table 1 and Table 3 show that similar or better results can be obtained with significant less training parameters and twice the efficiency by using a single transformer decoder.

Table 2: Accuracy compared to state-of-the-art. ST is short for the SynthText dataset, 90K for the Synth90K dataset; 50, 1k, full and 0 are the size of the used lexicons; 0 means that no lexicon is used.

Method	ConvNet, Data	IIIT5k			SVT		IC03			IC13	IC15	SVTP	CUTE
		50	1k	0	50	0	50	Full	0	0	0	0	0
Su and Lu [34]	–	–	–	–	83.0	–	92.0	82.0	–	–	–	–	–
Jaderberg et al. [15]	VGG, 90k	97.1	92.7	–	95.4	80.7	98.7	98.6	93.1	90.8	–	–	–
Jaderberg et al. [13]	VGG, 90k	95.5	89.6	–	93.2	71.7	97.8	97.0	89.6	81.8	–	–	–
Shi et al. [30]	VGG, 90k	97.8	95.0	81.2	97.5	82.7	98.7	98.0	91.9	89.6	–	–	–
Shi et al. [29]	VGG, 90k	96.2	93.8	81.9	95.5	81.9	98.3	96.2	90.1	88.6	–	71.8	59.2
Lee and Osindero [20]	VGG, 90k	96.8	94.4	78.4	96.3	80.7	97.9	97.0	88.7	90.0	–	–	–
Yang et al. [40]	VGG, Private	97.8	96.1	–	95.2	–	97.7	–	–	–	–	75.8	69.3
Cheng et al. [4]	ResNet, 90k+ST$^+$	99.3	97.5	87.4	97.1	85.9	**99.2**	97.3	94.2	93.3	70.6	–	–
Shi et al. [31]	ResNet, 90k+ST	**99.6**	98.8	93.4	97.4	89.5	98.8	98.0	94.5	91.8	76.1	78.5	79.5
Zhan and Lu [42]	ResNet, 90k + ST	**99.6**	98.8	93.3	97.4	**90.2**	-	-	-	91.3	76.9	79.6	83.3
Sheng et al. [28]	Modality-Transform, 90k	99.2	98.8	86.5	**98.0**	88.3	98.9	97.9	95.4	**94.7**	-	-	-
Yang et al. [39]	ResNet, 90k+ST	99.5	98.8	94.4	97.2	88.9	99.0	98.3	95.0	93.9	**78.7**	**80.8**	**87.5**
Bi-STET (this paper)	ResNet, 90k+ST	**99.6**	**98.9**	**94.7**	97.4	89.0	99.1	**98.7**	**96.0**	93.4	75.7	80.6	82.5

Table 3: Comparison of the number of trainable model parameters and training iterations.

Method	Model parameters ($\times 10^6$)	Training iterations ($\times 10^6$)	Batch size
Shi et al. [31]	88	1	64
Bi-STET (this paper)	**66**	**0.5**	64

5.2 Text recognition

In Table 2, we evaluate Bi-STET in terms of prediction accuracy on 7 public evaluation sets and compare it to other state-of-the-art (SOTA) STR methods. Bi-STET meets or outperforms SOTA methods on 6 out of 12 evaluation experiments. We achieve new SOTA results on the ICDAR03 and the IIIT5K datasets.

The strength of the transformer encoder-decoder w.r.t. to images with oriented and curved text is also shown by the results in Table 2. The five datasets where STET does not beat, but meets, the state-of-the-art are CUTE80, ICDAR13, ICDAR15, SVT-P and SVT. The fact that we do not achieve the state-of-the-art on the datasets SVT-P, ICDAR15 and CUTE80 can be explained by the fact that those datasets contain a considerable number of images that are rotated or have perspective distortions. We meet SOTA results on these datasets, we speculate that we don't exceed them because they have these distortions. It should also be noted we are able to meet these SOTA results without any specific network component for dealing with distortions, which other methods explicitly require [31, 39, 42]. For ICDAR13 and SVT-P we do not establish new SOTA performance figures, although we do meet the results of other methods with a small margin.

5.3 Analyzes

5.3.1 Attention heads analyzes

To get an understanding of the internal behaviour of Bi-STET, we extracted the attention distributions from Bi-STET during evaluation and visualize them in Figures 2a and 2b. Conditioned on different decoding directions, Bi-STET has learned to model an inverse alignment between the predicted output character and the region in the input images where the character is depicted – using only a single decoder. In Figure 2a, there is a clear attention alignment going from left to right over the image, while in Figure 2b, this alignment goes in the opposite direction. The model has jointly learned to model the character-image region alignment in both directions. Also, different attention heads do not specialize for left-to-right or right-to-left decoding, but learn how to change the attention direction when the output is conditioned on a different decoding direction. This shows the strength the method w.r.t. regularisation towards both sub-tasks. This is interesting from a multi-task learning point of view because this indicates that the same attention heads can learn different (sub)tasks.

5.3.2 Rotated and curved text

Bi-STET is solely trained as a general image-to-text encoder-decoder and does not contain a specific rectification component for handling rotated or curved text instances, unlike previous methods [31, 39, 42]. We are able to obtain results that meet those of state-of-the-art methods that are specifically optimized for curved and rotated text. Figure 3 provides a sample from the CUTE80 dataset with correctly and incorrectly predicted sequences. Looking at correctly predicted examples, we see that Bi-STET properly decodes words that are slightly curved or only curved in one direction. This is where the bidirectional decoding shows its strength. For example, the two middle images of the second row are correctly decoded when decoding from right-to-left decoder, but not when decoding in the other direction. From the first row of images we see that words that are curved in very strong arc shapes (heavy perspective distortions) are difficult for Bi-STET to decode. This also shows the strength of method w.r.t. regularisation towards curved and rotated text without using any specific rectification component.

5.3.3 Sequence length

Vaswani et al. [35] argue that transformer-based architectures are more suitable for capturing long-range dependencies for machine translation than RNNs, because of the global attention per encoding and decoding step. The self-attention results in the fact that the maximum distance in a sequence between two embeddings which are encoded or decoded is 1. Despite the fact that the maximum output sequence length in our evaluation experiment (max. length 17) is not as long as for other language tasks [19], we are interested in whether or not our transformer-based method is better in predicting longer output character sequence than an RNN-based method. In Figure 4, we show the relation between output sequence length and accuracy for the IIIT-5K evaluation set. We can see that the prediction accuracy given a sequence length is more or less constant until we reach

(a) Decoding attention while decoding from left-to-right.

(b) Decoding attention while decoding from right-to-left.

Figure 2: Visualization of the self-attention of layer 5 of the decoder while decoding the sequence (left-to-right). Each row in the matrix visualizes the attention distribution over the embeddings of the image on the X axis, while decoding the corresponding character in the input.

(a) ballys vs. bally (b) chathamkent vs. from (c) starbucks vs. and (d) coffee vs. offer

(e) ronaldo vs. ronaldo (f) meant vs. meant (g) football vs. football (h) united vs. united

Figure 3: Examples of curved text examples from the CUTE80 dataset that are correctly and incorrectly predicted by Bi-STET. In black the ground truth is given.

a character length of 11. After a sequence length of 11, the accuracy starts to degrade. It should be noted that there are very few samples with a sequence length of 11 or higher.

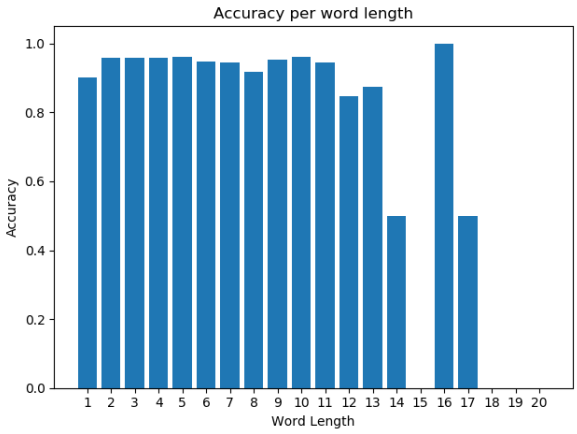

Figure 4: Text Recognition accuracies versus word length for Bi-STET. Tested on IIIT-5K.

By comparing Figure 4 to Figure 12 in [31], we see that Bi-STET performs similarly to Shi et al. [31]'s method for short character sequences and slightly better for longer character sequences which are longer than 11 characters. We conclude that Scene Text Transformer (STET) performs similar for short characters sequences and at least as good in decoding longer character sequences.

6 DISCUSSION AND CONCLUSION

We have introduced Bi-STET, a method for bidirectional STR with a single decoder. Bi-STET is capable of bidirectional decoding, without implementing the decoding direction conditioning at the architecture or algorithm level. The decoding conditioning is implemented at the input level, by adding an extra direction embedding to the input.

We show that Bi-STET achieves or outperforms state-of-the-art STR methods, with a considerably more efficient approach than other bidirectional STR methods (i.e., requiring 50 % less training iterations and significant less model parameters). By having fewer model parameters, the model can be executed on devices with less computational resources (for user applications). Besides that, less computational resources are required to obtain SOTA text recognition results. We also show that Bi-STET learns to exploit the same attention heads for both decoding directions, which means that there are no specialized attention heads in the model for each decoding direction. This is interesting from a multi-task learning point of view because different heads tend not to be focused on one decoding direction. Finally, we show that, due to the bidirectional decoding, Bi-STET is capable of handling slightly curved and orientated text and performs as well for longer text sequence as other bidirectional STR methods.

A future research direction is to combine Bi-STET with a Spatial Transformer Network [29, 31, 39] or a Rectification Network [42]. Bi-STET is able to handle oriented and perspective text in images; we believe that Bi-STET could benefit from an extra image processing component to be able to better handle oriented or perspective text. From a multi-task learning point of view, it would be interesting to explore task conditioning on the input level with more diverse tasks. In addition, an extension to tasks with more complex and diverse data modalities would also be a possible future research direction.

ACKNOWLEDGEMENTS

We thank Ana Lucic and Mostafa Dehghani for comments and suggestions. This research was supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Nationale Politie. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] J. Baek et al. What is wrong with scene text recognition model comparisons? dataset and model analysis. *arXiv preprint arXiv:1904.01906*, 2019.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] A. Bissacco et al. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, pages 785–792, 2013.

[4] Z. Cheng et al. Focusing attention: Towards accurate text recognition in natural images. In *ICCV*, pages 5076–5084, 2017.

[5] J. Deng et al. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009.

[6] J. Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135. JMLR.org, 2017.

[8] Y. Gao et al. Reading scene text with attention convolutional sequence modeling. *arXiv preprint arXiv:1709.04303*, 2017.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[10] A. Graves et al. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

[11] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016.

[12] K. He et al. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. *arXiv preprint arXiv:1412.5903*, 2014.

[14] M. Jaderberg et al. Deep features for text spotting. In *ECCV*, 2014.

[15] M. Jaderberg et al. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[16] L. Kaiser et al. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[17] D. Karatzas et al. ICDAR 2013 robust reading competition. In *ICDAR*, pages 1484–1493, 2013.

[18] D. Karatzas et al. ICDAR 2015 competition on robust reading. In *ICDAR*, pages 1156–1160, 2015.

[19] U. Khandelwal et al. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018.

[20] C.-Y. Lee and S. Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *CVPR*, pages 2231–2239, 2016.

[21] W. Liu et al. STAR-Net: a spatial attention residue network for scene text recognition. In *BMVC*, volume 2, page 7, 2016.

[22] S. M. Lucas et al. ICDAR 2003 robust reading competitions. In *ICDAR*, pages 682–687, 2003.

[23] M.-T. Luong et al. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[24] A. Mishra et al. Scene text recognition using higher order language priors. In *BMVC-British Machine Vision Conference*. BMVA, 2012.

[25] T. Quy Phan et al. Recognizing text with perspective distortion in natural scenes. In *ICCV*, pages 569–576, 2013.

[26] A. Radford et al. Language models are unsupervised multitask learners. `https://openai.com/blog/better-language-models`, 2019.

[27] A. Risnumawan et al. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41 (18):8027–8048, 2014.

[28] F. Sheng et al. NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. *arXiv preprint arXiv:1806.00926*, 2018.

[29] B. Shi et al. Robust scene text recognition with automatic rectification. In *CVPR*, pages 4168–4176, 2016.

[30] B. Shi et al. 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, 2017.

[31] B. Shi et al. Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[32] P. Shivakumara et al. A new gradient based character segmentation method for video text recognition. In *ICDAR*, pages 126–130, 2011.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] B. Su and S. Lu. Accurate scene text recognition based on recurrent neural network. In *ACCV*, pages 35–48. Springer, 2014.

[35] A. Vaswani et al. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[36] A. Veit et al. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

[37] K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, pages 591–604. Springer, 2010.

[38] K. Wang et al. End-to-end scene text recognition. In *ICCV*, pages 1457–1464. IEEE, 2011.

[39] M. Yang, Y. Guan, M. Liao, X. He, K. Bian, S. Bai, C. Yao, and X. Bai. Symmetry-constrained rectification network for scene text recognition. *arXiv preprint arXiv:1908.01957*, 2019.

[40] X. Yang, D. He, Z. Zhou, D. Kifer, and C. L. Giles. Learning to read irregular text with attention mechanisms. In *IJCAI*, pages 3280–3286, 2017.

[41] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[42] F. Zhan and S. Lu. Esir: End-to-end scene text recognition via iterative image rectification. *arXiv preprint arXiv:1812.05824*, 2018.

[43] X. Zhang et al. Asynchronous bidirectional decoding for neural machine translation. In *AAAI*, pages 5698–5705, 2018.

[44] L. Zhou et al. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105, 2019.

[45] Y. Zhu et al. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 10 (1):19–36, 2016.