Supplementary Material: Sequence-to-Sequence Contrastive Learning for Text Recognition

Aviad Aberdam* Technion aaberdam@cs.technion.ac.il

Ron Litman* AWS litmanr@amazon.com

Shahar Tsiper AWS tsiper@amazon.com

Oron Anschel AWS oronans@amazon.com

Ron Slossberg Technion ronslos@cs.technion.ac.il

Shai Mazor AWS smazor@amazon.com

R. Manmatha AWS manmatha@amazon.com

Pietro Perona Caltech and AWS peronapp@amazon.com

1. Text Recognition Scheme

In this section, we provide additional details on the text recognition architecture components considered in this work. In particular, we focus on three components: (i) the transformation performed by the thin-plate splines (TPS) [18, 9], (ii) the CTC based decoder [5, 17] and (iii) the attention based decoder [1, 3, 13].

1.1. Transformation

This stage transforms a cropped text image $\mathbf{X}$ into a normalized image $\mathbf{X}'$. This step is necessary when the input image contains text in a non-axis aligned layout, as often occurs in handwritten text and scene text images.

In this work, we follow [1], and utilize the Thin Plate Spline (TPS) transformation [18, 9] which is a variant of the spatial transformer network [9]. As depicted in Fig. 1, in this transformation, we first detect a pre-defined number of fiducial points at the top and bottom of the text region. Then, we apply a smooth spline interpolation between the obtained points to map the predicted textual region to a constant pre-defined size.

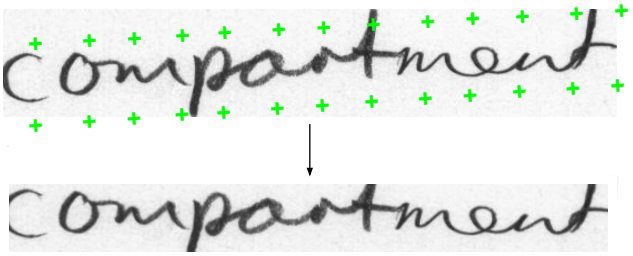

Figure 1: **TPS transformation.** This transformation first predicts fiducial points – marked as green points. Then, a smooth spline interpolation is employed, transforming these points to the border of a constant rectangle, yielding a normalized image with a fixed predefined size.

1.2. Connectionist Temporal Classification (CTC)

The CTC decoder [5] operates on a given sequential feature map $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_T]$, which in our framework can be $\mathbf{F} \in \{\mathbf{V}, \mathbf{H}, (\mathbf{H}, \mathbf{V})\}$. The inference phase consists of three stages. In the first stage, each frame $\mathbf{f}_t$ is transformed by a fully connected layer to yield $\mathbf{f}'_t$. Then, the CTC finds the sequence of characters with the highest probability:

$$\mathbf{c} = \arg\max_{\pi} \prod_{t=1}^{T} f'_{t,\pi_t}, \tag{1}$$

where $f'_{t,i}$ denotes the ith element in $\mathbf{f}'_t$. Next, the CTC removes repeated characters and blanks:

$$\mathbf{y} = \varphi(\mathbf{c}), \tag{2}$$

where $\varphi(\cdot)$ denotes the mapping function. For example, if $\mathbf{c}$ = "`aa-a-bbb-cc-ccc--`" then "$\mathbf{y}$ = `aabcc`".

For the CTC procedure during training we refer the reader to [5, 17].

1.3. Attention decoder

As for the CTC, the attention also operates on a given sequential feature map $\mathbf{F} \in \{\mathbf{V}, \mathbf{H}, (\mathbf{H}, \mathbf{V})\}$. The first step of decoding starts by computing the vector of attentional weights, $\alpha_{t'} \in \mathbb{R}^T$. For this goal, we first calculate $e_{t',t}$:

$$e_{t',t} = \mathbf{a}^T \tanh(\mathbf{W}\mathbf{s}_{t'-1} + \mathbf{V}\mathbf{f}_t + \mathbf{b}), \tag{3}$$

where $\mathbf{W}, \mathbf{V}, \mathbf{a}, \mathbf{b}$ are trainable parameters, and $\mathbf{s}_{t'-1}$ is the hidden state of the recurrent cell within the decoder at time

*Authors contribute equally and are listed in alphabetical order.

Figure 2: **Illustrations of augmentation procedures.** We show different augmentation pipelines that were considered in this work (a), which led to the final augmentation pipeline. We also show augmentations examples using the SimCLR [2] pipeline (b).

t'. Then, we compute $\alpha_{t'}$ by:

$$\alpha_{t',t} = \frac{\exp(e_{t',t})}{\sum_{j=1}^{T} e_{t',j}} \,. \tag{4}$$

As mention in the paper, the decoder linearly combines the columns of $\mathbf{F}$ into a vector $\mathbf{g}$ by utilizing the learned $\alpha_{t'}$:

$$\mathbf{g}_{t'} = \sum_{t=1}^{T} \alpha_{t',t} \mathbf{f}_t \,. \tag{5}$$

Next, the recurrent cell is fed with:

$$(\mathbf{x}_{t'}, \mathbf{s}_{t'}) = \mathrm{RNN}\left(\mathbf{s}_{t'-1}, [\mathbf{g}_{t'}, f(\mathbf{y}_{t'-1})]\right) \,, \tag{6}$$

where $f(\cdot)$ is a one-hot embedding, $[\cdot,\cdot]$ denotes the concatenation operator, and $\mathbf{y}_{t'}$ is obtained by:

$$\mathbf{y}_{t'} = \mathrm{softmax}(\mathbf{W}_0 \mathbf{x}_{t'} + \mathbf{b}_0) \,, \tag{7}$$

where $\mathbf{W}_0, \mathbf{b}_0$ are trainable parameters. The loss used for the attention decoder is the negative log-likelihood, as in [3].

2. Data Augmentation

In this section, we provide additional details for reproducing our augmentation procedure, implemented using the `imgaug` [10] augmentation package. As described in Section 5, this augmentation pipeline is used for the self-supervised training, where we stochastically augment each image twice. In Fig. 2, we present different augmentation procedures that we examined in our work, which eventually led us to the final pipeline. Our default procedure consists of a random subset of the following operations.

Linear contrast We modify the input image contrast by applying the pixel-wise transformation: $127 + \alpha(v - 127)$, where α is sampled uniformly from the interval $[0.5, 1.0]$ and $v \in [0, 255]$ is the pixel value.

Blur We blur the image using a Gaussian kernel with a randomly selected standard deviation of $\sigma \in (0.5, 1.0)$.

Sharpen The image is sharpened by blending it with a highly sharpened version of itself. The lightness parameter found in the `imgaug` framework, is sampled uniformly from the interval $[0.0, 0.5]$, and the alpha factor used for blending the image is sampled uniformly from the interval $[0.0, 0.5]$.

Crop We first extract a smaller-sized sub-image from the given full-sized input image. Then, we resize this crop to the original size. As mention in Section 6, the vertical cropping can be more aggressive than the horizontal cropping. Therefore, the percentage of the vertical cropping is sampled uniformly from the interval $[0\%, 40\%]$, while the horizontal cropping percentage is sampled from $[0\%, 2\%]$.

Perspective transform A four point perspective transformation is applied. These points are placed on the input image by using a random distance from the original image corners, where the random distance is drawn from a normal distribution with a standard deviation sampled uniformly from the interval $[0.01, 0.02]$.

Piecewise affine We apply an affine transformation that moves around each grid point by a random percentage drawn uniformly from the interval $[2\%, 3\%]$.

A pseudo-code for the augmentation pipeline, written with the `imgaug` [10] package, is as follows.

```
from imgaug import augmenters as iaa
iaa.Sequential([iaa.SomeOf((1, 5),
[
  iaa.LinearContrast((0.5, 1.0)),
  iaa.GaussianBlur((0.5, 1.5)),
  iaa.Crop(percent=((0, 0.4),
                    (0, 0),
                    (0, 0.4),
                    (0, 0.0)),
                   keep_size=True),
  iaa.Crop(percent=((0, 0.0),
                    (0, 0.02),
                    (0, 0),
                    (0, 0.02)),
                   keep_size=True),
  iaa.Sharpen(alpha=(0.0, 0.5),
              lightness=(0.0, 0.5)),
  iaa.PiecewiseAffine(scale=(0.02, 0.03),
                      mode='edge'),
  iaa.PerspectiveTransform(
                      scale=(0.01, 0.02)),
],
random_order=True)])
```

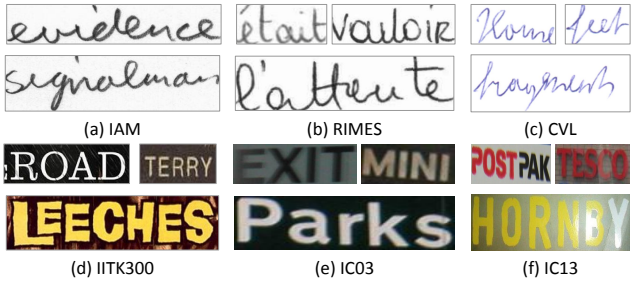

Figure 3: **Dataset samples.**

3. Datasets

In this work, we consider the following public datasets for handwriting and scene text, see examples in Fig. 3:

- **RIMES** [6] handwritten French text dataset, written by 1300 different writers, partitioned into writer independent training, validation and test. This collection contains 66,480 correctly segmented words.
- **IAM** [15] handwritten English text dataset, written by 657 different writers, partitioned into writer independent training, validation and test. This collection contains 74,805 correctly segmented words.
- **CVL** [12] handwritten English text dataset, written by 310 different writers, partitioned into writer independent training and test. 27 of the writers wrote 7 texts and the other 283 writers wrote 5 texts.
- **SynthText** (ST)[7] contains 8M cropped scene text images which were generated synthetically. This dataset was utilized for training the scene text recognizer.
- **IIIT5K-words** (IIIT5K) [16] contains 2000 training and 3000 testing cropped scene text images from the Internet.
- **ICDAR-2003** (IC03) [14] contains 867 cropped scene text images.
- **ICDAR-2013** (IC13) [11] contains 848 training and 1015 testing cropped scene text.

The last three datasets were used just for validation and test sets as described Section 4.

4. Implementation Details

Recognizer setting Unless otherwise specified, the text recognizer architecture consists of: a feature extraction stage of a 29-layer ResNet [8], as in [4, 1]; a sequence modeling stage using a two-layer Bidirectional-LSTM (BiLSTM) with 256 hidden units per layer; and if needed, an attention decoder of an LSTM cell with 256 memory blocks. Following common practice in text recognition [1], we preresize all images to 32×100 both for training and testing. For the English datasets (IAM, CVL, IIIT5K, IC03 and IC13), we use 95 symbol classes: 52 case-sensitive letters, 10 digits and 33 for special characters. For the French dataset (RIMES), we add to the above the French accent symbols. As for special symbols for CTC decoding, an additional "[blank]" token is added to the label set. For the attention decoder, two special symbols are added: "[S]", "[EOW]" which indicate the start of the sequence and the end of the word.

SimCLR re-implementation To compare our method to non-sequential contrastive learning methods, we re-implement the SimCLR algorithm, with the same augmentations and projection head as in [2]. This algorithm can be applied in our settings as we anyhow resize each input image to a fixed width following the common practice in text recognition [1, 13].

Pre-training procedure In general, for the self-supervised training, we use a batch size of 1024, and train for 200K iterations for handwritten datasets and 400K iterations for scene-text. That said, since frame-to-one mapping results in many more instances for the contrastive loss (Figure 5(c)), we needed to reduce the batch size to 256. To compensate for it, we increased the number of iterations to 300K for handwritten datasets and to 600K for scene-text. For optimization, we use the AdaDelta optimizer [19] with a decay rate of 0.95, a gradient clipping parameter with a magnitude of 5 and a weight decay parameter of 10^{-4}. The learning rate is initialized to 10, and is reduced by a factor of 10 after 60% and 80% of the training iterations. Finally, all experiments are trained and tested using the PyTorch framework on 4 cards of Tesla V100 GPU with 16GB memory.

Decoder-evaluation and fine-tuning procedures For these stages, we train the decoder using a batch size of 256 for 50K iterations, employing a similar learning rate scheduling as in the self-supervised phase. The augmentation procedure consists of light cropping, linear contrast and Gaussian blur. We select our best model using a validation dataset, where in handwritten text we use the public validation sets, and in scene text our validation data is the union of the training data of IC13 and IIIT, as done in [1].

5. Error Vs. Labeled Data Amount

In Fig. 4, we present the word error rate of SeqCLR and the *supervised baseline* as a function of the portion of labeled data used for the fine-tuning phase (see Section 5.2). For example, in the RIMES dataset when using an attention

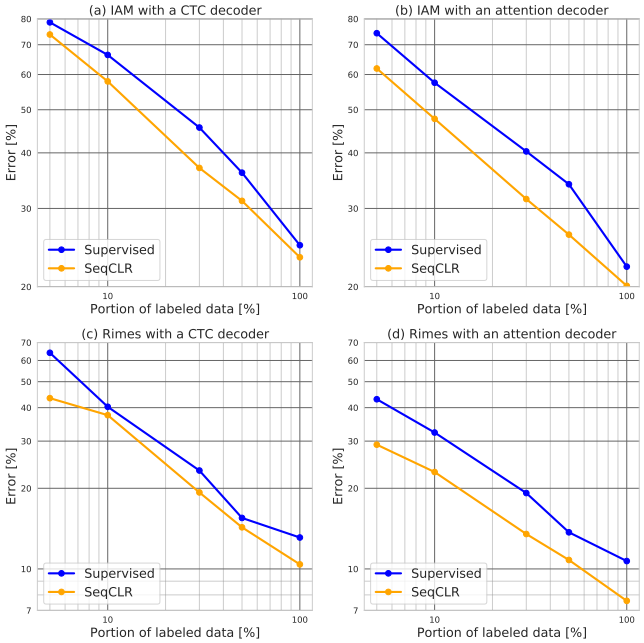

Figure 4: **Word error rate as a function of labeled data amount in a log-log scale.** Roughly speaking, SeqCLR unsupervised pre-training has the same effect as doubling the labeled data amount in the sense of reducing the error rate.

decoder, the SeqCLR algorithm utilizing 50% of the labeled data achieves the same error rate as training the *supervised baseline* on the entire labeled dataset. In general, as can be seen, SeqCLR is effective as almost doubling the size of the labeled dataset.

References

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4715–4723, 2019. 1, 3

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2, 3

[3] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5076–5084, 2017. 1, 2

[4] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of the IEEE international conference on computer vision*, pages 5076–5084, 2017. 3

[5] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. 1

[6] Emmanuèle Grosicki and Haikal El Abed. Icdar 2009 handwriting recognition competition. In *2009 10th International Conference on Document Analysis and Recognition*, pages 1398–1402. IEEE, 2009. 3

[7] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2315–2324, 2016. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 1

[10] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020. 2

[11] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE, 2013. 3

[12] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In *2013 12th international conference on document analysis and recognition*, pages 560–564. IEEE, 2013. 3

[13] Ron Litman, Oron Anschel, Shahar Tsiper, Roee Litman, Shai Mazor, and R Manmatha. Scatter: selective context attentional scene text recognizer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11962–11972, 2020. 1, 3

[14] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. Icdar 2003 robust reading competitions. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 682–687. Citeseer, 2003. 3

[15] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002. 3

[16] Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. 2012. 3

[17] Baoguang Shi, Xiang Bai, and Cong Yao. 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. 1

[18] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4168–4176, 2016. 1

[19] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 3