






SimAN: Exploring Self-Supervised Representation Learning of Scene Text via Similarity-Aware Normalization

Supplementary Material

Table 1. Visualization of the self-supervised learning scheme. The queries are denoted as red boxes. The proposed SimAN requires distinguishable representations to identify different patterns, thus enabling self-supervised representation learning of the encoder. Under the supervision of the \mathcal{L}_2 , the responses on the neighboring patches are becoming more and more accurate, suggesting the increasing quality of the representations.

Query															
Key (neighboring patch)															
$\mathcal{L}_2 \approx 0.3$															
Mask $\mathcal{L}_2 \approx 0.1$															
$\mathcal{L}_2 \approx 0.02$															

1. Visualization

To validate the effectiveness of the proposed SimAN, which estimates pattern similarity and then queries corresponding style keys for recovering the augmented patch, we visualize the attentional responses of style keys on the neighboring patch. As shown in Table 1, with the decrease of the \mathcal{L}_2 loss, the attentional mask presents a more and more accurate response based on a similar pattern. For instance, as shown in the first column, a “t” query (denoted as a red box) on the source patch obtains a response of “t” on the neighboring patch. This reveals the learning mechanism of the proposed SimAN, *i.e.*, distinguishable representations between different characters are required to identify patterns and align correct styles for image reconstruction.

2. Benchmark

We detail the public scene text benchmarks used for recognition evaluation as follows.

ICDAR 2003 [14] (**IC03**) contains 867 cropped images after discarding images that contain non-alphanumeric characters or less than three characters [19].

ICDAR 2013 [11] (**IC13**) inherits most of its samples from IC03. It contains 1015 cropped images.

ICDAR 2015 [10] (**IC15**) was collected by using Google Glasses. It includes more than 200 irregular text images.

Street View Text [19] (**SVT**) consists of 647 word images for testing. Some images are severely corrupted by noise and blur.

Street View Text Perspective [17] (**SVT-P**) is a perspective distorted version of SVT, containing 645 cropped images for testing.

IIIT5K-Words [15] (**IIIT5K**) contains 3000 and 2000 cropped word images for testing and training, respectively. Some texts are curved.

CUTE80 [18] (**CT80**) was specifically collected to evaluate the performance of curved text recognition. It contains 288 cropped natural images.

Total-Text [4] (**TText**) focuses on curved text recognition. It contains 2201 cropped word images.

3. Augmentation Strategy

Different from the previous study SeqCLR [1], we discard the spatial transformation augmentations because our approach recovers images based on consistent visual cues. Therefore, we limit the augmentation strategies to color changes, blurring, sharpening, and random noise. We use a CPU-efficient toolkit¹ to perform augmentation. The pseudo-code is shown as below for reference.

```

1 import albumentations as A
2 A.Sequential([
3     # Color Changes
4     A.InvertImg(),
5     A.OneOf([
6         A.ChannelDropout(),
7         A.ChannelShuffle(),
8         A.ToGray(),
9         A.RGBShift(),
10        A.Equalize(),
11        A.RandomBrightnessContrast(0.5, 0.5),
12        A.ColorJitter(0.5, 0.5, 0.5, 0.5),
13        A.HueSaturationValue(),
14        A.RandomToneCurve(),

```

¹<https://github.com/albumentations-team/albumentations>

```

15  ]),
16  A.OneOf([
17    # Sharpen Blending
18    A.Sharpen(alpha=(1.0, 1.0)),
19    # Blurring
20    A.OneOf([
21      A.ImageCompression(40, 80),
22      A.Blur(blur_limit=[3, 3]),
23      A.GaussianBlur(blur_limit=[3, 3]),
24      A.MedianBlur(blur_limit=[3, 3]),
25      A.MotionBlur(blur_limit=[3, 3]),
26    ]),
27    # Random Noise
28    A.OneOf([
29      A.Emboss((0.5, 1.0), (0.8, 1.0)),
30      A.GaussNoise(),
31      A.ISONoise((0.1, 0.5), (0.5, 1.0)),
32      A.MultiplicativeNoise(),
33    ]),
34  ]),
35 ]))

```

Table 2. Probe evaluation using an attentional probe with two BiLSTMs (256 hidden units). We report the word accuracy (Acc., %) and word-level accuracy up to one edit distance (E.D. 1, %). The two augmentation toolkits achieve comparable performance.

Augmentation Toolkit	IHITSK		IC03		IC13	
	Acc.	E.D. 1	Acc.	E.D. 1	Acc.	E.D. 1
SeqCLR’s	65.6	78.2	71.3	84.2	69.4	82.2
Ours	66.5	78.8	71.7	83.6	68.7	81.6

Note that we use a different toolkit *albugmentations* from that of SeqCLR [1]. We clarify the performance gain is achieved by our proposed approach, rather than the different toolkit. As shown in Table 2, the two augmentation toolkits achieve comparable performance.

4. Recognizer Initialization

In the section of *Probe Evaluation*, we simply initialize the recognizer backbone using the whole pre-trained backbone parameters. This is a common setting to perform a probe evaluation to validate the representation quality. The architecture of the recognizer backbone (ResNet-29) and the corresponding decoder for its self-supervised training are shown in Table 3 and Table 4, respectively.

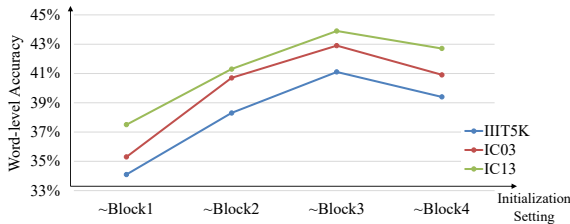


Figure 1. The recognizer achieves the best performance by using the pre-trained parameters up to a depth at “Block3”.

Table 3. Architecture of ResNet-29. We present the size of feature maps during representation learning and recognition training. The backbone (encoder) trained on image patches can generalize well to images of variant widths. We pad the output feature maps (whose height is one) along the vertical direction for the extraction of eight-neighborhood statistics.

Layers	Configurations	Size	
		Repr. Learn.	Reg.
Input	RGB image	32×32	32×100
Conv1	c: 32 k: 3×3	32×32	32×100
Conv2	c: 64 k: 3×3	32×32	32×100
Pool1	k: 2×2 s: 2×2	16×16	16×50
Block1	$\begin{bmatrix} c:128, k:3 \times 3 \\ c:128, k:3 \times 3 \end{bmatrix} \times 1$	16×16	16×50
Conv3	c: 128 k: 3×3	16×16	16×50
Pool2	k: 2×2 s: 2×2	8×8	8×25
Block2	$\begin{bmatrix} c:256, k:3 \times 3 \\ c:256, k:3 \times 3 \end{bmatrix} \times 2$	8×8	8×25
Conv4	c: 256 k: 3×3	8×8	8×25
Pool3	k: 2×2 s: 2×1 p: 0×1	4×9	4×26
Block3	$\begin{bmatrix} c:512, k:3 \times 3 \\ c:256, k:3 \times 3 \end{bmatrix} \times 5$	4×9	4×26
Conv5	c: 512 k: 3×3	4×9	4×26
Block4	$\begin{bmatrix} c:512, k:3 \times 3 \\ c:512, k:3 \times 3 \end{bmatrix} \times 3$	4×9	4×26
Conv6	c: 512 k: 2×2 s: 2×1 p: 0×1	2×10	2×27
Conv7	c: 512 k: 2×2 s: 1×1 p: 0×0	1×9	1×26

However, it is revealed by [3] that not all the pre-trained parameters can benefit the downstream task. Therefore, for the experiment of *Semi-Supervision Evaluation*, we explore different initialization settings, *i.e.*, how many layers (how deep) should be initialized by using pre-trained parameters. Specifically, we simply choose four blocks of the recognizer backbone as our four depth options. We fine-tune the recognizer using 10K labeled samples of SynthText [6]. As shown in Figure 1, the recognizer achieves the best performance with the initialization setting at depth “Block3”. We provide the decoder for the self-supervised learning of the first three blocks, as shown in Table 5.

5. Probe/Recognizer Objectives

After the self-supervised representation learning stage, we perform probe and semi-supervision evaluation. We set the batch size to 256 and train the recognizer for 50K iterations. The optimizer is AdaDelta [21] with the default setting. The learning rate is set to 1.0 and linearly decreased

Table 4. Architecture of the decoder for the self-supervised learning of ResNet-29 in the Section of *Probe Evaluation*.

Layers	Configurations	Size
Input	Feature Maps	1×9
DeConv	c: 256, k: 2×2 , s: 1×1 , p: 0×0 , ReLU	2×10
Conv	c: 256, k: 3×3 , s: 1×1 , p: 1×1 , BN, ReLU	2×10
DeConv	c: 192, k: 2×2 , s: 2×1 , p: 0×0 , ReLU	4×11
Conv	c: 192, k: 3×3 , s: 1×1 , p: 1×0 , BN, ReLU	4×9
DeConv	c: 160, k: 2×2 , s: 2×1 , p: 0×0 , ReLU	8×10
Conv	c: 160, k: 3×3 , s: 1×1 , p: 1×0 , BN, ReLU	8×8
Upsample	Ratio: $\times 2$, Mode: "nearest"	16×16
Conv	c: 128, k: 3×3 , s: 1×1 , p: 1×1 , ReLU	16×16
Conv	c: 128, k: 3×3 , s: 1×1 , p: 1×1 , BN, ReLU	16×16
Upsample	Ratio: $\times 2$, Mode: "nearest"	32×32
Conv	c: 64, k: 3×3 , s: 1×1 , p: 1×1 , ReLU	32×32
Conv	c: 64, k: 3×3 , s: 1×1 , p: 1×1 , BN, ReLU	32×32
Conv	c: 3, k: 3×3 , s: 1×1 , p: 1×1 , Tanh(\cdot)	32×32

Table 5. Architecture of the decoder for the self-supervised learning of first three blocks of ResNet-29 in the Section of *Semi-Supervision Evaluation*.

Layers	Configurations	Size
Input	Feature Maps	4×9
DeConv	c: 256, k: 2×2 , s: 1×1 , p: 0×0 , ReLU	5×10
Conv	c: 256, k: 3×3 , s: 1×1 , p: 1×1 , BN, ReLU	5×10
DeConv	c: 192, k: 2×2 , s: 2×1 , p: 0×0 , ReLU	11×11
Conv	c: 192, k: 3×3 , s: 1×1 , p: 0×0 , BN, ReLU	9×9
DeConv	c: 160, k: 2×2 , s: 1×1 , p: 0×0 , ReLU	10×10
Conv	c: 160, k: 3×3 , s: 1×1 , p: 0×0 , BN, ReLU	8×8
Upsample	Ratio: $\times 2$, Mode: "nearest"	16×16
Conv	c: 128, k: 3×3 , s: 1×1 , p: 1×1 , ReLU	16×16
Conv	c: 128, k: 3×3 , s: 1×1 , p: 1×1 , BN, ReLU	16×16
Upsample	Ratio: $\times 2$, Mode: "nearest"	32×32
Conv	c: 64, k: 3×3 , s: 1×1 , p: 1×1 , ReLU	32×32
Conv	c: 64, k: 3×3 , s: 1×1 , p: 1×1 , BN, ReLU	32×32
Conv	c: 3, k: 3×3 , s: 1×1 , p: 1×1 , Tanh(\cdot)	32×32

to 0.1. The input word images are resized to 64×200 . The experiments are conducted on the PyTorch framework [16] using two NVIDIA P100 GPUs (16GB memory per GPU).

The probe/recognizer outputs 95 categories, including 52 case-sensitive letters, 10 digits, 32 punctuation symbols, and an additional "Blank" token for CTC decoding [5] or an "End of Sequence" token for attention decoding [2].

1) The CTC decoder [5] transforms the feature sequence $F \in \mathbb{R}^{T \times C}$ to an output sequence $Y \in \mathbb{R}^{T \times 95}$ using a fully connected layer. For each time step, $y_t \in \mathbb{R}^{95}$ denotes the probability distribution over 95 categories. The objective is to minimize the negative log-likelihood of conditional prob-

ability of ground truth GT :

$$\mathcal{L}_{CTC} = -\log p(GT|Y), \quad (1)$$

where the conditional probability is defined as the sum of probabilities of all possible sequence $\pi_i \in \pi$ that can be mapped onto the GT (For instance, "CC--VVV---P--RR" can be mapped onto "CVPR"). It is formulated as

$$p(GT|Y) = \sum_{\pi} p(\pi|Y) = \sum_{\pi} \prod_{t=1}^T Y_t^{\pi_i}, \quad (2)$$

where $Y_t^{\pi_i}$ denotes the predicted probability at time step t with a sequence $\pi_i \in \pi$.

2) The attention decoder [2] is optimized by minimizing the negative log-likelihood of conditional probability of ground truth GT :

$$\mathcal{L}_{Att} = -\sum_{t=1}^T \log p(GT_t|y_t), \quad (3)$$

where y_t is the predicted probability over 95 categories at time step t , given by

$$y_t = \text{SoftMax}(W s_t + b). \quad (4)$$

The s_t is the hidden state at the t -th step, updated by

$$s_t = \text{GRU}(s_{t-1}, (y_{t-1}, g_t)), \quad (5)$$

where g_t represents the glimpse vectors

$$g_t = \alpha \cdot h. \quad (6)$$

The h denotes the feature sequence. The α is the attention mask, expressed as

$$\alpha = \text{SoftMax}(e), \quad (7)$$

$$e = w^T \text{Tanh}(W_s s_{t-1} + W_h h + b_e). \quad (8)$$

Here, W , w^T , W_s , W_h and b_e are trainable parameters.

6. Adversarial Loss

We adopt an adversarial objective to minimize the distribution shift between the generated and real data, which is a widely used setting for image generating tasks. To study the effectiveness of the adversarial training, we conduct an ablation experiment by disabling the adversarial loss \mathcal{L}_{adv} . As shown in Table 7, the \mathcal{L}_{adv} increases representation quality and makes the generated distribution closer to the real one. We believe the \mathcal{L}_{adv} is necessary for visual effects, because it achieves more lifelike images.

Table 6. Semi-supervised performance evaluation. We sample three orders of scales (10K, 100K, and 1M) of data from SynthText (6M). Our approach can learn high-quality representations from unlabeled data and improve the supervised baseline, especially when used with low-resource labeled data.

Labeled Data	Supervision	IIIT5K	SVT	IC03	IC13	SVT-P	CT80	IC15
10K	Sup.	35.0 \pm 6.7	7.9 \pm 3.4	37.6 \pm 6.3	38.6 \pm 6.5	6.8 \pm 2.8	8.5 \pm 3.2	10.4 \pm 3.5
	Semi-Sup.	41.1 \pm 1.3	16.2 \pm 1.4	42.9 \pm 2.1	43.9 \pm 1.2	14.2 \pm 1.2	15.5 \pm 1.7	17.5 \pm 1.2
100K	Sup.	72.6 \pm 0.3	55.2 \pm 1.2	79.4 \pm 1.1	75.3 \pm 0.8	45.4 \pm 0.9	46.7 \pm 1.0	47.6 \pm 1.0
	Semi-Sup.	73.6 \pm 0.5	55.3 \pm 1.0	79.9 \pm 1.0	75.6 \pm 0.5	45.6 \pm 0.8	46.8 \pm 1.5	47.9 \pm 0.4
1M	Sup.	84.1 \pm 0.5	73.1 \pm 0.2	88.2 \pm 0.6	86.4 \pm 1.0	60.5 \pm 0.8	59.5 \pm 1.5	58.9 \pm 0.8
	Semi-Sup.	84.1 \pm 0.6	73.1 \pm 1.0	89.2 \pm 1.1	86.5 \pm 0.9	62.1 \pm 1.1	63.7 \pm 2.8	59.7 \pm 0.6
6M	Sup.	86.6 \pm 0.5	79.6 \pm 0.6	91.5 \pm 0.7	89.0 \pm 0.3	68.3 \pm 0.7	71.9 \pm 1.8	66.2 \pm 0.6
	Semi-Sup.	87.5 \pm 0.3	80.6 \pm 0.5	91.8 \pm 0.7	89.9 \pm 0.6	68.3 \pm 1.1	71.4 \pm 1.7	66.2 \pm 0.4

Table 7. Ablation study of adversarial loss. We evaluate the representation quality using an attention probe with two BiLSTMs (256 hidden units), and the distribution shift using FID [7] score. We average the word accuracies (%) of IIIT5K, IC03 and IC13.

\mathcal{L}_{adv}	Acc. \uparrow	FID \downarrow
\times	68.8	24.0
\checkmark	69.0	23.2

7. Compare with AdaIN

It is known that the AdaIN [8, 12] can transfer style using global statistics (mean and standard deviation) of feature maps. We conduct a probe evaluation (following the setting of ResNet-FCN-Att.) to compare our SimAN with AdaIN. As shown in Table 8, the proposed SimAN outperforms AdaIN. This suggests the representation capability is improved by the similarity estimation, which minimizes the distance between similar patterns.

Table 8. Probe evaluation of AdaIN and SimAN.

Method	IIIT5K		IC03		IC13	
	Acc.	E.D. 1	Acc.	E.D. 1	Acc.	E.D. 1
AdaIN	9.7	21.9	9.1	18.7	11.1	24.6
SimAN	22.2	39.7	22.3	38.6	24.1	43.6

8. Semi-Supervision Evaluation

We provide experimental results of five runs on seven popular benchmarks in Table 6.

9. Network Architecture

We present the encoder and decoder used in the Section of *Generative Visual Task* in Table 9 and 10, respectively. These are popular architectures and are widely used [8, 9].

We present the discriminator in Table 11.

Table 9. Architecture of the encoder in the Section of *Generative Visual Task*.

Layers	Configurations			Size
Input	RGB image			$3 \times 64 \times 64$
Conv1	c: 3	k: 1		$3 \times 64 \times 64$
Conv2	c: 64	k: 3	Reflection Pad: 1, ReLU	$64 \times 64 \times 64$
Conv3	c: 64	k: 3	Reflection Pad: 1, ReLU	$64 \times 64 \times 64$
MaxPool	k: 2	s: 2		$64 \times 32 \times 32$
Conv4	c: 128	k: 3	Reflection Pad: 1, ReLU	$128 \times 32 \times 32$
Conv5	c: 128	k: 3	Reflection Pad: 1, ReLU	$128 \times 32 \times 32$
MaxPool	k: 2	s: 2		$128 \times 16 \times 16$
Conv6	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
Conv7	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
Conv8	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
Conv9	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
MaxPool	k: 2	s: 2		$256 \times 8 \times 8$
Conv10	c: 512	k: 3	Reflection Pad: 1, ReLU	$512 \times 8 \times 8$

10. Data Synthesis

Following the pipeline of SynthText [6], we simply render a text on a clean canvas. The fonts are publicly available². We follow the strict setting proposed by Long *et al.* [13] to include punctuation symbols, digits, upper-case and lower-case characters for evaluation. We also use the same recognizer trained on our 1M synthetic data.

We perform random blurring on the synthetic data to meet the low-quality practice of scene text images. The pseudo-code is shown as below for reference.

```

1 import albumentations as A
2 A.OneOf([
3     A.ImageCompression(40, 80),
4     A.Blur(blur_limit=[5, 11]),

```

²<https://fonts.google.com/>


```

5 A.GaussianBlur(blur_limit=(5, 11)),
6 A.MedianBlur(blur_limit=[5, 11]),
7 A.MotionBlur(blur_limit=[5, 11])
8 ])

```

Table 10. Architecture of the decoder in the Section of *Generative Visual Task*.

Layers	Configurations			Size
Input	Feature Map			$512 \times 8 \times 8$
Conv1	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 8 \times 8$
Upsample	Ratio: $\times 2$, Mode: "nearest"			$256 \times 16 \times 16$
Conv2	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
Conv3	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
Conv4	c: 256	k: 3	Reflection Pad: 1, ReLU	$256 \times 16 \times 16$
Conv5	c: 128	k: 3	Reflection Pad: 1, ReLU	$128 \times 16 \times 16$
Upsample	Ratio: $\times 2$, Mode: "nearest"			$128 \times 32 \times 32$
Conv6	c: 128	k: 3	Reflection Pad: 1, ReLU	$128 \times 32 \times 32$
Conv7	c: 64	k: 3	Reflection Pad: 1, ReLU	$64 \times 32 \times 32$
Upsample	Ratio: $\times 2$, Mode: "nearest"			$64 \times 64 \times 64$
Conv8	c: 64	k: 3	Reflection Pad: 1, ReLU	$64 \times 64 \times 64$
Conv9	c: 3	k: 3	Reflection Pad: 1, ReLU	$3 \times 64 \times 64$
Tanh	-			$3 \times 64 \times 64$

Table 11. Architecture of the discriminator.

Layers	Configurations					
Conv1	c: 64	k: 4	s: 2	p: 1	PReLU	
Conv2	c: 128	k: 4	s: 2	p: 1	PReLU	
Conv3	c: 256	k: 4	s: 2	p: 1	PReLU	
Conv4	c: 512	k: 4	s: 1	p: 1	PReLU	
Conv5	c: 1	k: 4	s: 1	p: 1		

11. Arbitrary-Length Text Editing

We follow the same setting as EditText [20] to generate a content image. We use a standard font style "Arial.ttf" to put the target text string on a clean canvas as content input. The target text is randomly selected from the corpus of SynthText [6]. Thus, the length of the source text string and the target one can be significantly different, which simulates practice challenges. We present the edited results in Figure 2.

12. Font Interpolation

We formulate the process of font interpolation as mathematical equations. First, we extract the content representations of the source image and target image as Q and K ,



Figure 2. Arbitrary-length text editing. All the images are resized to (64, 256).

respectively. Besides, we obtain their style representations $\mu^{\text{source}}, \sigma^{\text{source}}, \mu^{\text{target}}$ and σ^{target} .

12.1. Color Interpolation

First, we rearrange the target style representations according to the source content representation Q :

$$\begin{aligned}\mu^{\text{rearrange}} &= \mu^{\text{target}} \text{Softmax} \left(\frac{K^T Q}{\sqrt{d_k}} \right), \\ \sigma^{\text{rearrange}} &= \sigma^{\text{target}} \text{Softmax} \left(\frac{K^T Q}{\sqrt{d_k}} \right).\end{aligned}\quad (9)$$

Then we perform interpolation on the source and rearranged style representations:

$$\begin{aligned}\mu' &= (1 - \alpha)\mu^{\text{source}} + \alpha\mu^{\text{rearrange}}, \\ \sigma' &= (1 - \alpha)\sigma^{\text{source}} + \alpha\sigma^{\text{rearrange}}, \alpha \in [0, 1].\end{aligned}\quad (10)$$

Finally, we decode the feature maps to obtain an image:

$$Q'_{c,i,j} = Q_{c,i,j}\sigma'_{c,i,j} + \mu'_{c,i,j}, \quad (11)$$

$$I_{\text{rec}} = \text{Decoder}(Q'). \quad (12)$$

12.2. Glyph Interpolation

The glyph interpolation requires a same character/string on the source and target image. First, we normalize the target glyph image using the source style:

$$\begin{aligned}\mu^{\text{rearrange}} &= \mu^{\text{source}} \text{Softmax} \left(\frac{Q^T K}{\sqrt{d_k}} \right), \\ \sigma^{\text{rearrange}} &= \sigma^{\text{source}} \text{Softmax} \left(\frac{Q^T K}{\sqrt{d_k}} \right).\end{aligned}\quad (13)$$

The target glyph can be presented as:

$$K_{c,i,j}^{\text{rearrange}} = K_{c,i,j}\sigma_{c,i,j}^{\text{rearrange}} + \mu_{c,i,j}^{\text{rearrange}}. \quad (14)$$

This ensures the difference between the source and target representation is only the glyph.

Then we perform interpolation on the source and rearranged glyph representation:

$$Q'_{c,i,j} = (1 - \alpha)Q_{c,i,j} + \alpha K_{c,i,j}^{\text{rearrange}}, \alpha \in [0, 1]. \quad (15)$$

Finally, we obtain an image:

$$I_{\text{rec}} = \text{Decoder}(Q'). \quad (16)$$

References

- [1] Aviad Aberdam, Ron Litman, Shahar Tsiper, Oron Anshel, Ron Slossberg, Shai Mazor, R Manmatha, and Pietro Perona. Sequence-to-sequence contrastive learning for text recognition. In *CVPR*, pages 15302–15312, 2021.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
- [3] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pre-training from pixels. In *ICML*, pages 1691–1703, 2020.
- [4] Chee-Kheng Ch'ng, Chee Seng Chan, and Cheng-Lin Liu. Total-Text: toward orientation robustness in scene text detection. *Int. J. Doc. Anal. Recogn.*, 23(1):31–52, 2020.
- [5] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006.
- [6] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, pages 2315–2324, 2016.
- [7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pages 6626–6637, 2017.
- [8] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510, 2017.
- [9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016.
- [10] Dimosthenis Karatzas, Lluís Gomez-Bigorda, Angelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. ICDAR 2015 competition on robust reading. In *ICDAR*, pages 1156–1160, 2015.
- [11] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluís Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluís Pere De Las Heras. ICDAR 2013 robust reading competition. In *ICDAR*, pages 1484–1493, 2013.
- [12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.
- [13] Shangbang Long and Cong Yao. UnrealText: Synthesizing realistic scene text images from the unreal world. In *CVPR*, pages 5488–5497, 2020.
- [14] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. ICDAR 2003 robust reading competitions. In *ICDAR*, pages 682–687, 2003.
- [15] Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. In *BMVC*, pages 1–11, 2012.
- [16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017.
- [17] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. Recognizing text with perspective distortion in natural scenes. In *ICCV*, pages 569–576, 2013.
- [18] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41(18):8027–8048, 2014.

- [19] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *ICCV*, pages 1457–1464, 2011.
- [20] Liang Wu, Chengquan Zhang, Jiaming Liu, Junyu Han, Jingtuo Liu, Errui Ding, and Xiang Bai. Editing text in the wild. In *ACM Int. Conf. Multimedia*, pages 1500–1508, 2019.
- [21] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.