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

Canjie Luo\(^1\), Lianwen Jin\(^{1,2,*}\), Jingdong Chen\(^3\)

\(^1\)South China University of Technology, \(^2\)Peng Cheng Laboratory, \(^3\)Ant Group
{canjie.luo, lianwen.jin}@gmail.com, jingdongchen.cjd@antgroup.com

Abstract

Recently self-supervised representation learning has drawn considerable attention from the scene text recognition community. Different from previous studies using contrastive learning, we tackle the issue from an alternative perspective, i.e., by formulating the representation learning scheme in a generative manner. Typically, the neighboring image patches among one text line tend to have similar styles, including the strokes, textures, colors, etc. Motivated by this common sense, we augment one image patch and use its neighboring patch as guidance to recover itself. Specifically, we propose a Similarity-Aware Normalization (SimAN) module to identify the different patterns and align the corresponding styles from the guiding patch. In this way, the network gains representation capability for distinguishing complex patterns such as messy strokes and cluttered backgrounds. Experiments show that the proposed SimAN significantly improves the representation quality and achieves promising performance. Moreover, we surprisingly find that our self-supervised generative network has impressive potential for data synthesis, text image editing, and font interpolation, which suggests that the proposed SimAN has a wide range of practical applications.

1. Introduction

The computer vision community has witnessed the great success of supervised learning over the last decade. However, the supervised learning methods heavily rely on labor-intensive and expensive annotations. Otherwise, they might suffer from generalization problems. Recently self-supervised representation learning has become a promising alternative and is thus attracting growing interest \([24, 34]\). It has been shown that the self-supervised representations can benefit subsequent supervised tasks \([6–10, 18]\).

Despite the fast-paced improvements of representation learning on single object recognition/classification tasks, the field of scene text recognition is meeting extra challenges. For instance, multiple characters in one image cannot be regarded as one entity \([38, 63]\). Directly adopting current non-sequential contrastive learning schemes for sequence-like characters \([45]\) usually leads to performance deterioration \([1]\). This suggests the gap between the non-sequential and sequential schemes. Therefore, it is desirable to design a specific representation learning scheme for scene text recognition.

As a scene text image containing dense characters is significantly different from a natural image, SeqCLR \([1]\)
divided one text line into several instances using certain strategies and performed contrastive learning on these instances. The learning scheme is shown in Figure 1 (a). The SeqCLR designed for sequence-to-sequence visual recognition outperformed the representative non-sequential method SimCLR [7]. Although it brought a huge leap forward, the representation learning of scene text remains a challenging open research problem, where the nature of scene text has not been fully explored.

Thus, we review several properties of scene text that differ from those of general objects (e.g., face, car, and dog). For instance, one feature that highlights scene text is its constant stroke width [13]. Simultaneously, it is observed that color similarity typically occurs across one text line. These specialties provided cues for hand-crafted features, such as connected components [41], stroke width transform [13, 59], and maximally stable extremal region trees [21], which were popular before the dramatic success of deep neural networks.

In this paper, we explore the representation learning from a new perspective by considering the above unique properties of scene text. The learning scheme is shown in Figure 1 (b). Specifically, we randomly crop two neighboring image patches from one text line. One patch is augmented and the other one guides the recovery of the augmented one. As one text line usually exhibits consistent styles, including the strokes, textures, colors, etc., the original styles of the augmented patch can be found on the neighboring patch according to similar content patterns. Thus, we propose a Similarity-Aware Normalization (SimAN) module to align corresponding styles from the neighboring patch by estimating the similarity of the content representations between these two patches. This means that the representations are required to be sufficiently distinguishable so that different patterns can be identified and the corresponding styles can be correctly aligned. Only in this way, the network can produce a precise recovered image patch. Therefore, the proposed SimAN enables high-quality self-supervised representation learning in a generative way. Moreover, we find that our self-supervised network has competitive performance with state-of-the-art scene text synthesis methods [17, 23, 35, 61]. It is also promising to apply SimAN to other visual effect tasks, such as text image editing and font interpolation.

To summarize, our contributions are as follows:

- We propose a generative (opposite of contrastive [34]) representation learning scheme by utilizing the unique properties of scene text, which might inspire rethinking the learning of better representations for sequential data like text images. To the best of our knowledge, this is the first attempt for scene text recognition.
- We propose a SimAN module, which estimates the similarity of the representations between the augmented image patch and its neighboring patch to align corresponding styles. Only if the representations are sufficiently distinguishable, different patterns can be identified and be aligned with correct styles. Otherwise, the network might result in a wrong recovered image, e.g., in different colors.

- The proposed SimAN achieves promising representation performance. Moreover, the self-supervised network shows impressive capabilities to synthesize data, edit text images and interpolate fonts, suggesting the broad practical applications of the proposed approach.

2. Related Work

2.1. Data Hunger of Scene Text Recognition

Scene text recognition is a crucial research topic in the computer vision community, because the text in images provides considerable semantic information for us. One important open issue in this field is data hunger. Typically, mainstream scene text recognizers [14, 46, 55] require a large number of annotated data. However, data collection and annotation cost a lot of resources. For instance, annotating a text string is tougher than selecting one option as the ground truth for single object classification datasets, whereas tens of millions of training data are required to gain robustness. Although synthetic data are available, previous studies [26, 33, 37, 63] suggested that there is a gap between real and synthetic data. To mitigate this problem, Zhang et al. [63] and Kang et al. [26] proposed domain adaptation models to utilize unlabeled real data. Our study explores representation learning in a generative way, which is an alternative solution to make use of unlabeled real data.

2.2. Visual Representation Learning

In the big data era, tremendous amounts of unlabeled data are available. Making the best use of unlabeled data becomes a crucial topic. Self-supervised representation learning has drawn massive attention owing to its excellent capability of pre-trained feature extraction [24, 34]. For instance, an encoder trained after a pretext task can extract transferrable features to benefit downstream tasks. We summarize popular methods into two main categories according to their objectives as follows.

The contrastive learning scheme defines the pretext task as a classification task or a distance measuring task. For instance, the pretext task is to predict relative rotation [31] and position [57]. Recently the similarity measuring pretext task has become dominant, which aims to minimize the distance between the positive pairs while maximizing their distance to the negative ones using a discriminative head [5, 7, 8, 10, 18]. It is closely related to metric learning. Furthermore, the similarity measuring task using only positive pairs and discarding negative samples [9, 16] is also an
emerging topic.

For the field of scene text, Baek et al. [3] introduced existing self-supervised techniques [18, 31] to use unlabeled data but resulted in approximately the same performance. Aberdam et al. [1] proposed a contrastive representation learning scheme, termed SeqCLR, to satisfy the sequence-to-sequence structure of scene text recognition. This is the first step towards scene text representation.

The generative learning scheme has not been intensively studied in computer vision. One reason for this may be that the raw image signal is in a continuous and high-dimensional space, unlike the natural language sentences in a discrete space (e.g., words or phrases) [18]. Therefore, it is difficult to define an instance. Although it is possible to model the image pixel by pixel [51], this theoretically requires much more high-performance clusters [6]. Another solution is the denoising auto-encoder [50, 53], which learns features by reconstructing the (corrupted) input image.

Our approach falls into the second category of visual representation learning, i.e., the generative learning scheme. We propose a novel representation learning scheme by studying the unique properties of scene text and using an image reconstruction pretext task.

3. Methodology

In this section, we first introduce the design of the pretext task and the construction of the training samples. Then, we detail the proposed SimAN module. Finally, we present the objectives of the task and the complete learning scheme. The overall framework is shown in Figure 2.

3.1. Training Sample Construction

Constructing appropriate training samples is critical to the success of the pretext task. We enable the scene text representation learning by recovering an augmented image patch using its neighboring patch as guidance. This design considers the unique properties of scene text, i.e., the styles (e.g., stroke width, textures, and colors) within one text line tend to be consistent.

The pretext task requires decoupled style and content inputs. As shown in Figure 2, given an unlabeled text image \( I \in \mathbb{R}^{3 \times H \times W} \) (the width \( W \) is required to be larger than two times of height \( H \)), we randomly crop two neighboring image patches \( I_s, I_c \in \mathbb{R}^{3 \times H \times H} \) as style and content input, respectively. This ensures sufficient differences in content between the two patches. Even if the neighboring patches might contain a same characters, their positions are different. Then, we augment (blurring, random noise, color changes, etc.) the content patch \( I_c \) as \( I_{aug} \) to make its style different from the style patch \( I_s \). Finally, the pretext task takes \( I_{aug} \) as content input and \( I_s \) as the style guidance to recover an image \( I_{rec} \). The source content patch \( I_c \) serves as supervision.

Discussion As our pretext task is recovering an augmented patch under the guidance of its neighboring patch, the visual cues should be consistent in both patches. Some spatial augmentation strategies, such as elastic transformation, might break the consistency and lead to failed training. For instance, it might bring changes to the stroke width. The excessively distorted strokes are also diverse from the source font style. Therefore, we avoid all of the spatial transformation augmentation methods that are widely used for self-supervised representation learning. This is also a significant difference with previous study SeqCLR [1].
3.2. Similarity-Aware Normalization

Previous studies [22, 29] revealed that the statistics of feature maps, including mean and variance, can represent styles. Based on this finding, we perform instance normalization (IN) [22, 49] on the feature maps to remove the style and obtain content representations as key \((K, \text{from} \ I_s)\) and query \((Q, \text{from} \ I_{aug})\) as

\[
K = \text{IN}(\text{Encoder}(I_s)), Q = \text{IN}(\text{Encoder}(I_{aug})),
\]

where the \(K\) and \(Q\) are normalized feature maps with spatial scale \(\mathbb{R}^{C_F \times H_F \times W_F}\). The \(\text{IN}(\cdot)\) is compute as

\[
\text{IN}(x) = \frac{x - \mu(x)}{\sqrt{\sigma(x)^2 + \epsilon}},
\]

where \(\mu(\cdot)\) and \(\sigma(\cdot)\) respectively compute the mean and standard deviation, performing independently for each channel and each sample.

For the local style representations, we extract eight-neighborhood mean and standard deviation at position \((i, j)\) on the \(c\)-th channel of the feature maps as

\[
\mu_{c,i,j} = \frac{1}{9} \sum_{p,q \in \mathcal{N}_{i,j}} x_{c,p,q},
\]

\[
\sigma_{c,i,j} = \frac{1}{9} \sqrt{\sum_{p,q \in \mathcal{N}_{i,j}} (x_{c,p,q} - \mu_{c,i,j})^2},
\]

where \(\mathcal{N}_{i,j}\) is the position set comprising of the eight-neighborhood around the position \((i, j)\) and itself. Here \(\mu, \sigma \in \mathbb{R}^{C_F \times H_F \times W_F}\) serve as value \((\mathcal{V}, \text{from} \ I_s)\).

Then the statistics \(\mu\) and \(\sigma\) is adaptively rearranged according to the similarity between the patterns of the two inputs by (here \(K, Q, \mu\) and \(\sigma\) are reshaped to \(\mathbb{R}^{C_F \times H_F \times W_F}\))

\[
\mu' = \mu \text{ Softmax} \left( \frac{K^T Q}{\sqrt{d_k}} \right), \quad \sigma' = \sigma \text{ Softmax} \left( \frac{K^T Q}{\sqrt{d_k}} \right),
\]

where \(d_k\) is the dimension of the input \(K\). The \(\mu'\) and \(\sigma'\) are reshaped to \(\mathbb{R}^{C_F \times H_F \times W_F}\).

Finally, we perform a reverse process of \(\text{IN}(\cdot)\) to align rearranged styles to each position for image recovery as

\[
Q'_{c,i,j} = Q_{c,i,j} \sigma'_{c,i,j} + \mu'_{c,i,j},
\]

\[
I_{rec} = \text{Decoder}(Q').
\]

As the proposed SimAN integrates styles and contents to recover an image, it enables representation learning. If the encoder produces meaningless content or style representations, the decoder cannot correctly recover the source image. For instance, the unidentifiable content representations will confuse the style alignment and result in a messy image. The inaccurate style representations will lead to color distortions. In a word, the image reconstruction objective requires effective representations of both content and style.

3.3. Learning Scheme

As we formulate the pretext task as image reconstruction, the source patch \(I_s\) can serves as supervision. We minimize the distance between the recovered image \(I_{rec}\) and target image \(I_c\) as

\[
\mathcal{L}_2 = \|I_{rec} - I_c\|_2^2.
\]

Simultaneously, we adopt a widely used adversarial objective to minimize the distribution shift between the generated and real data:

\[
\min_{\text{Encoder, Decoder}} \mathcal{L}_{adv} = \mathbb{E}\left[(D(I_s) - 1)^2\right] + \mathbb{E}\left[(D(I_{rec}))^2\right],
\]

\[
\min_{\text{Encoder, Decoder}} \mathcal{L}_{adv} = \mathbb{E}\left[(D(I_{rec}) - 1)^2\right],
\]

where \(D\) denotes a discriminator.

The complete learning scheme is shown in Algorithm 1. The encoder/decoder and discriminator are alternately optimized to achieve adversarial training.

---

**Algorithm 1** Representation Learning Scheme

**Input:** Encoder, Decoder, Discriminator \(D\)

**Output:** Encoder, Decoder

1. for iteration \(t = 0, 1, 2, ..., T\) do
   1. Sample a mini-batch \(\{I_i\}_{i=1}^B\) from unlabeled data
   2. for each \(I_i\) do
      1. Randomly crop \(I_s\) and \(I_c\), augment \(I_c\) as \(I_{aug}\)
   3. Forward Encoder, SimAN and Decoder
   4. Compute loss for \(\{I_{rec,i}\}_{i=1}^B\)
   5. Update \(D\) using \(\min_D \mathcal{L}_{adv}\)
   6. Update Encoder and Decoder using
      \[
      \min_{\text{Encoder, Decoder}} \mathcal{L}_{adv} + \lambda \mathcal{L}_2
      \]
   7. end

end

9. (The \(\lambda\) is empirically set to 10.)

---

4. Experiments

In this section, we conduct extensive experiments to validate the effectiveness of the proposed approach. First, we compare the quality of the learned representations with that of the previous study SeqCLR [1]. Then, we study the performance of our approach by using a semi-supervised setting, where we pre-train the encoder using unlabeled data and fine-tune it using partially labeled data. Finally, we show the potential of our generative approach for other visual tasks. For instance, we attempt to synthesize diverse data to train a robust recognizer. Moreover, we compare our self-supervised model with mainstream supervised models on the text image editing task. We also demonstrate some promising visual effects on font interpolation.
4.1. Dataset

We evaluate our approach on several public benchmarks that are widely used in scene text recognition studies. These datasets include IC03 [36], IC13 [28], IC15 [27], SVT [54], SVT-P [43], IIIT5K [39], CUTE80 (CT80) [44] and Total-Text (TText) [12].

We construct a dataset for self-supervised representation learning. To obtain more realistic and diverse scene text images, we collect samples from public real training datasets, including IIIT5K [39], IC13 [28], IC15 [27], COCO-Text [52], RCTW [47], ArT [11], ReCTS [62], MTWI [19], LSVT [48] and MLT [40]. We discard low-resolution images with a height of less than 32 pixels or width of less than 64 pixels (the width should be greater than two times the height for constructing training samples). Because in practice, low-quality images confuse the image recovery task and lead to inefficient training. As a result, we discard their labels and obtain an unlabeled dataset composed of approximately 300k real samples, termed Real-300K\(^1\). Besides, we also use the popular synthetic dataset SynthText [17] for fair comparisons with the previous study SeqCLR [1].

4.2. Implementation Details

We provide more details, such as augmentations, architectures, probe objectives, and training settings, in the Supplementary Material.

**Encoder/Decoder** We adopt a popular recognizer backbone ResNet-29 [2] as our encoder. We symmetrically design a lightweight decoder.

**Recognizer** The complete architecture of the recognizer follows [1,2], including a rectification module, a ResNet-29 backbone, two stacked BiLSTMs and a CTC [15]/Attention [4] decoder, as shown in Figure 3.

**Optimization** In the self-supervised representation learning stage, we set the batch size to 256 and train the network for 400K iterations. It takes less than 3 days for convergence on two NVIDIA P100 GPUs (16GB memory per GPU). The optimizer is Adam [30] with the settings of \(\beta_1 = 0.5\) and \(\beta_2 = 0.999\). The learning rate is set to \(10^{-4}\) and linearly decreased to \(10^{-5}\). The images are resized to a height of 32 pixels, maintaining the aspect ratio. The training setting of recognizers follows previous study SeqCLR [1].

\(^{1}\)https://github.com/Canjie-Luo/Real-300K.

4.3. Probe Evaluation

We first study the representation quality using the common protocol, namely probe evaluation. Specifically, we perform self-supervised pre-training of the ResNet-29 backbone using SynthText [17]. Then we fix the parameters of the backbone and feed the frozen representations to a CTC/Attention probe. The probes are trained on the same labeled SynthText dataset. It is believed that the higher the representation quality, the better the probe can obtain cues for classification.

The quantized results, including word accuracy (Acc.) and word-level accuracy up to one edit distance (E.D.) [1], are reported in Table 1. Note that our generative scheme is significantly different from the contrastive scheme SeqCLR [1], which uses sufficient sequential modeling (RNN projection head and sequential mapping) in the self-supervised pre-training phase. Although the direct comparisons between the two approaches are somewhat unreasonable, we list SeqCLR’S results under a similar experimental setting for reference.

Here we analyze the results of our approach. Note that the sequential modeling (2*RNN) in the encoder reduces the quality of representations. This is because our approach models local patterns for recovery, but the sequential modeling introduces contexts to disturb this learning scheme. Therefore, we discard the sequential modeling in the encoder. This means our approach might lack the capacity of sequence modeling after self-supervised representation learning. However, it is possible to equip a lightweight RNN in the probe, which remarkably improves the representation quality. Overall, we obtain promising representations in a generative manner. This might bring a brand new learning perspective in the field of scene text recognition.

Moreover, we find that this experimental setting (pre-training the backbone and fine-tuning the probe using the very same synthetic dataset) might not meet the actual practice. In fact, we usually encounter one situation that we have vast amounts of unlabeled real-world data. It is worth making the best use of the real-world data. Therefore, we conduct an experiment under this new setting to further verify the effectiveness of our approach. We perform self-supervised learning of the backbone using the Real-300K dataset. As shown in Table 3, the recognition performance is significantly boosted. As the real-world dataset provides more realistic and diverse images, it benefits the robustness of the backbone. Another reason why using a real dataset...
As SynthText provides six million training samples, it is dant unlabeled data, it has important practical significance. Supervision manner. Since it can make the best use of abundant approaches is worth future explorations. In this regard, the complementarity of contrastive and generative learning scheme SimAN. 1) We summarize the benchmarks, which are also real-world datasets. There exist different properties of the two schemes. In our generative learning scheme decouples content and style representations, we can randomly integrate existing examples. As shown in Figure 4, we replace the input. Then the generative network can synthesize an image according to the consistent visual cues, we achieve to sample smaller subsets with three orders of scales (10K, 100K, and 1M from the original 6M data). After performing self-supervised pre-training of the backbone on SynthText, we use the pre-trained parameters to initialize the recognizer backbone. Finally, we fine-tune the entire recognizer using different subsets of SynthText.

As shown in Table 4, our approach using the semi-supervised setting outperforms the supervised baseline. For instance, under the 10K low-resource setting, our approach increases the accuracy by more than 5%, which suggests that the recognition robustness is highly correlated with representation quality. With the increase of the scale of labeled data, our approach can still contribute to recognition accuracy. We compare the semi-supervised results with the previous study termed SeqCLR [1] under the same setting. Note that our approach can still slightly improve recognition performance using the whole SynthText for fine-tuning, whereas the SeqCLR shows inconsistent performance. This indicates the generalization ability of our approach.

### 4.5. Generative Visual Tasks

We demonstrate the potential of our approach on generative visual effect tasks. For the generalization to several different tasks, we adopt a widely used VGG encoder and a corresponding decoder [22, 25] in our model. The training dataset is Real-300K. The image height is set to 64 pixels.

#### 4.5.1 Data Synthesis

As our generative learning scheme decouples content and style representations, we can randomly integrate existing styles and new contents to synthesize diverse training samples. As shown in Figure 4, we replace the \( I_\text{ref} \) with a style reference image and replace the \( I_{aug} \) with a new content input. Then the generative network can synthesize an image in a similar style retaining the required content. Note that the terms “style” and “content” are somewhat different from those of font style transfer tasks [56]. Here the...
Table 4. 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.

| Method       | Supervision | IIIT5K Labeled Training Data | IC03 Labeled Training Data | IC13 Labeled Training Data |
|--------------|-------------|------------------------------|---------------------------|---------------------------|
|              |             | 10K 100K 1M 6M               | 10K 100K 1M 6M            | 10K 100K 1M 6M            |
| SeqCLR [1]   | Sup.        | - - - 83.8                  | -                          | -                          |
|              | Semi-Sup.   | - - - 82.9 ↑ 0.9            | -                          | -                          |
|              |             | 92.2 ↑ 1.1                  | 88.1                       |
| Ours         | Sup.        | 35.0 72.6 84.1 86.6         | 37.6 79.4 88.2 91.5        |
|              | Semi-Sup.   | 41.1 ↑ 6.1 73.6 ↑ 1.0 94.1 87.5 ↑ 0.9 | 42.9 ↑ 5.3 79.9 ↑ 0.5 89.2 ↑ 1.0 91.8 ↑ 0.3 | 43.9 ↑ 5.3 75.6 ↑ 0.3 86.4 ↑ 0.1 89.9 ↑ 0.9 |

First, we visualize the distributions of the limited real labeled samples and our plentiful synthetic samples. As shown in Figure 5, the limited labeled real-world data cannot cover diverse styles. However, our synthetic data fills the empty style space, indicating the significantly enriched styles. Then, we conduct recognition experiments to show the quantitative results. Following the settings of UnrealText [35], we synthesize 1M samples to train the same recognizer and report the accuracy on several benchmarks. As shown in the second last row in Table 5, our samples outperform previous synthesis methods [17, 23, 35, 61] on four (out of six) benchmarks without bells and whistles. We find that our synthetic samples have a high resolution (height of 64 pixels), which usually cannot meet the low-quality practice of scene text. Therefore, we simply add blurring to the samples. The recognition performance is further boosted, suggesting that our synthesis pipeline is scalable.

4.5.2 Arbitrary-Length Text Editing

The goal of editing text in the wild is to change the word on the source image while retaining the realistic source look. As our approach can synthesize new words within source styles, we study the performance of our self-supervised approach and a popular supervised method EditText [58]. We generate 10K images using the corpus of SynthText [17] and the style of IC13 [28]. Then we evaluate the style distribution similarity using the FID score [20] and the readability using a mainstream recognizer\(^2\) [45]. As shown in Figure 6 and Table 6, the EditText cannot handle target text of various lengths. That means the editing is limited to ap-

---

\(^2\)https://github.com/youdao-ai/SRNet
\(^3\)https://github.com/meijieru/crnn.pytorch
proximately the same length words. Although its style distribution is closer to the source images, its generated images are unreadable. On the contrary, our approach can adaptively align correct styles to arbitrary-length text, indicating the flexibility of our self-supervised approach. In our practice, we find that our approach is sufficient for cross-language editing, as shown in Figure 7. It has a wide range of applications, such as menu translation and cross-border e-commerce.

![Figure 6. Visualization of text editing. The EditText [58] cannot deal with target strings of variant lengths, whereas our approach adaptively aligns correct styles and achieves more readable results.](image)

Table 6. Arbitrary-length Text editing evaluation. We report FID score and word-level recognition accuracy (%). Although the supervised EditText can imitate more font category and background texture, our self-supervised approach achieves better readability.

| Method     | Supervision | FID ↓ | Acc. ↑ |
|------------|-------------|-------|--------|
| EditText [58] | ✓           | 40.5  | 14.9   |
| Ours       | ×           | 67.9  | 57.6   |

![Figure 7. Cross language editing via our self-supervised approach.](image)

4.5.3 Font Interpolation

It is believed that font design is a professional technique belonging to a few experts [56]. We present an interesting application of our approach on font interpolation for automatically and efficiently generating font candidates. As we parameterize the style and content as representations, we can interpolate these representations to achieve transitional effects. For instance, we compute the style representations (local statistics) of two images and rearrange them according to the same content representations. We interpolate the two style representations to decode images so that we can obtain the gradually changing colors, sheens, and shadows, as shown in Figure 8. Simultaneously, we interpolate the content representations to achieve font glyph changes. This potential suggests our approach might facilitate font design.

5. Broader Impacts

The proposed self-supervised approach has a wide range of applications owing to its capability of decoupling styles and contents of scene text. For instance, it can swap text to achieve image (and video) manipulation, which can be used in many applications, such as menu translation and cross-border e-commerce. However, we point out the risks of text image editing. It can be employed to tamper sensitive data, such as personal information, license plate numbers, and financial statistics, to trick systems that rely on text recognition. It is necessary to reduce these negative impacts. One promising technological solution is to detect the edited/attacking image using a qualified discriminator. It is also essential to increase media literacy among vast swathes of the population.

6. Conclusion

We have presented a novel approach for self-supervised representation learning of scene text from a brand new perspective, i.e., in a generative manner. It takes advantage of the style consistency of neighboring patches among one text image to reconstruct one augmented patch under the guidance of its neighboring patch. Specifically, we propose a SimAN module to identify different patterns (e.g., background noise and foreground characters) based on the representation similarity between the two patches. The representations are required to be sufficiently distinguishable so that corresponding styles can be correctly aligned to reconstruct the augmented patch. Otherwise, it results in an inaccurate image. In this way, it enables self-supervised representation learning via the image reconstruction task.

Extensive experiments show that our generative approach achieves promising representation quality and outperforms the previous contrastive method. Furthermore, it presents the impressive potential for data synthesis, text image editing and font interpolation, demonstrating a wide range of practical applications. Our study might arouse the rethinking of self-supervised learning of scene text. In the future, we will study the complementarity of contrastive and generative learning schemes to further improve the representation quality.

Acknowledgment

This research was supported in part by NSFC (Grant No. 61936003) and GD-NSF (No. 2017A030312006).
References

[1] Aviad Aberdam, Ron Litman, Shahar Tsiper, Oron Anschel, Ron Slossberg, Shai Mazor, R Mannatha, and Pietro Perona. Sequence-to-sequence contrastive learning for text recognition. In CVPR, pages 15302–15312, 2021.

[2] 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 ICCV, pages 4715–4723, 2019.

[3] Jeonghun Baek, Yusuke Matsui, and Kiyouharu Aizawa. What if we only use real datasets for scene text recognition? toward scene text recognition with fewer labels. In CVPR, pages 3113–3122, 2021.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In NeurIPS, volume 33, pages 9912–9924, 2020.

[6] 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.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In ICML, pages 1597–1607, 2020.

[8] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. NeurIPS, 33:22243–22255, 2020.

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In ICLR, pages 15750–15758, 2021.

[10] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In ICCV, pages 9640–9649, 2021.

[11] Chee Kheng Chng, Yuliang Liu, Yuxin Wang, Zhendong Mao, and Yongdong Zhang. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In CVPR, pages 7098–7107, 2021.

[12] 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.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Pires, Zhaohan Guo, Mohammad Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In NeurIPS, volume 1, pages 1–1, 2020.

[14] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In CVPR, pages 2315–2324, 2016.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In CVPR, pages 9729–9738, 2020.

[16] Mengchao He, Yuliang Liu, Zhibo Yang, Sheng Zhang, Canjie Luo, Feiyu Gao, Qi Zheng, Yongpan Wang, Xin Zhang, and Lianwen Jin. ICDAR 2018 contest on robust reading for multi-type web images. In ICLR, pages 7–12, 2018.

[17] 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.

[18] Weilin Huang, Yu Qiao, and Xiaou Tang. Robust scene text detection with convolution neural network induced MSER trees. In ECCV, pages 497–511, 2014.

[19] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In ICCV, pages 1501–1510, 2017.

[20] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. Int. J. Comput. Vis., 116(1):1–20, 2016.

[21] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. IEEE Trans. Pattern Anal. Mach. Intell., 43(11):4037–4058, 2021.

[22] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, pages 694–711, 2016.

[23] Lei Kang, Marçal Rusinol, Alicia Fornés, Pau Riba, and Mauricio Villegas. Unsupervised writer adaptation for synthetistic-to-real handwritten word recognition. In WACV, pages 3502–3511, 2020.

[24] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Sunun 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 1571–1576, 2019.

[25] 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. In ICDAR, pages 1156–1160, 2015.

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In CVPR, pages 4401–4410, 2019.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[28] Nikos Komodakis and Spyros Gidaris. Unsupervised representation learning by predicting image rotations. In ICLR, 2018.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. NeurIPS, 25:1097–1105, 2012.
[33] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. In AAAI, volume 33, pages 8610–8617, 2019.

[34] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhayou Wang, Jing Zhang, and Jie Tang. Self-supervised Learning: Generative or Contrastive. IEEE Trans. Knowl. Data Eng., 1(1):1–1, 2021.

[35] Shangbang Long and Cong Yao. UnrealText: Synthesizing realistic scene text images from the unreal world. In CVPR, pages 5488–5497, 2020.

[36] 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.

[37] Canjie Luo, Qingxiang Lin, Yuliang Liu, Lianwen Jin, and Chunhua Shen. Separating content from style using adversarial learning for recognizing text in the wild. Int. J. Comput. Vis., 129(4):960–976, 2021.

[38] Canjie Luo, Yuanzhi Zhu, Lianwen Jin, and Yongpan Wang. Learn to augment: Joint data augmentation and network optimization for text recognition. In CVPR, pages 13746–13755, 2020.

[39] Anand Mishra, Kartteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. In BMVC, pages 1–11, 2012.

[40] Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowd, Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. In BMVC, pages 1–11, 2012.

[41] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In CVPR, pages 3538–3545, 2012.

[42] 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.

[43] Trung Quy Phan, Palaiahankote Shivakumara, Shangxuan Tian, and Chew Lim Tan. Recognizing text with perspective distortion in natural scenes. In ICCV, pages 569–576, 2013.

[44] Anhar Rismunawan, 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.

[45] 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 Trans. Pattern Anal. Mach. Intell., 39(11):2298–2304, 2017.

[46] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Astar: An attentional scene text recognizer with flexible rectification. IEEE Trans. Pattern Anal. Mach. Intell., 41(9):2035–2048, 2018.

[47] Baoguang Shi, Cong Yao, Minghui Liao, Mingkun Yang, Pei Xu, Linyan Cui, Serge Belongie, Shijian Lu, and Xiang Bai. ICDAR 2017 competition on reading chinese text in the wild (RCTW-17). In ICDAR, volume 1, pages 1429–1434, 2017.

[48] Yipeng Sun, Jiaming Liu, Wei Liu, Junyu Han, Errui Ding, and Jingtuo Liu. Chinese street view text: Large-scale chinese text reading with partially supervised learning. In ICCV, pages 9086–9095, 2019.

[49] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. CoRR, abs/1607.08022, 2016.

[50] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In NeurIPS, pages 6309–6318, 2017.

[51] Aaron Van Oord, NaL Calchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In ICML, pages 1747–1756, 2016.

[52] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. COCO-Text: Dataset and benchmark for text detection and recognition in natural images. CoRR, abs/1601.07140, 2016.

[53] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In ICML, pages 1096–1103, 2008.

[54] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In ICCV, pages 1457–1464, 2011.

[55] Tianwei Wang, Yuanzhi Zhu, Lianwen Lin, Canjie Luo, Xiaoyan Chen, Yaqiang Wu, Qianying Wang, and Mingxiang Cai. Decoupled attention network for text recognition. In AAAI, volume 34, pages 12216–12224, 2020.

[56] Yizhi Wang and Zhouruai Lian. DeepVecFont: Synthesizing high-quality vector fonts via dual-modality learning. ACM Trans. Graph., 1(1), 2021.

[57] Chen Wei, Lingxi Xie, Xutong Ren, Yingda Xia, Chi Su, Jiaying Liu, Qi Tian, and Alan L Yuille. Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning. In CVPR, pages 1910–1919, 2019.

[58] Qiang 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.

[59] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Editing text in the wild. In ACM Int. Conf. Multimedia, pages 1500–1508, 2019.

[60] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. CoRR, abs/1212.5701, 2012.
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 $L_2$, the responses on the neighboring patches are becoming more and more accurate, suggesting the increasing quality of the representations.

| Query | Key (neighboring patch) | Mask |
|-------|-------------------------|------|
| $L_2 \approx 0.3$ | - | - |
| $L_2 \approx 0.1$ | - | - |
| $L_2 \approx 0.02$ | - | - |

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 $L_2$ 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 [36] (IC03) contains 867 cropped images after discarding images that contain non-alphanumeric characters or less than three characters [54].

ICDAR 2013 [28] (IC13) inherits most of its samples from IC03. It contains 1015 cropped images.

ICDAR 2015 [27] (IC15) was collected by using Google Glasses. It includes more than 200 irregular text images.

Street View Text [54] (SVT) consists of 647 word images for testing. Some images are severely corrupted by noise and blur.

Street View Text Perspective [43] (SVT-P) is a perspective distorted version of SVT, containing 645 cropped images for testing.

IIIT5K-Words [39] (IIIT5K) contains 3000 and 2000 cropped word images for testing and training, respectively. Some texts are curved.

CUTE80 [44] (CT80) was specifically collected to evaluate the performance of curved text recognition. It contains 288 cropped natural images.

Total-Text [12] (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, sharpen blending, and random noise. We use a CPU-efficient toolkit to perform augmentation. The pseudo-code is shown as below for reference.

```python
import albumentations as A
A.Sequential([  # Color Changes
    A.InvertImg(),
    A.OneOf([  # Color Changes
        A.ChannelDropout(),
        A.ChannelShuffle(),
        A.ToGray(),
        A.RGBShift(),
        A.Equalize(),
        A.RandomBrightnessContrast(0.5, 0.5),
        A.ColorJitter(0.5, 0.5, 0.5, 0.5),
        A.HueSaturationValue(),
        A.RandomToneCurve(),
    ])
])
```

4https://github.com/albumentations-team/albumentations
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 | IIIT5K  | IC03  | IC13  |
|--------------|---------|-------|-------|
| 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 `albumentations` 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.

![Graph](image_url)  
**Figure 1.** The recognizer achieves the best performance by using the pre-trained parameters up to a depth at “Block3”.

However, it is revealed by [6] 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 [17]. 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 [60] with the default setting. The learning rate is set to 1.0 and linearly decreased...
The objective is to minimize the negative log-likelihood of case-sensitive letters, digits, punctuation symbols, etc. Experiments are conducted on the PyTorch framework [42] to study the effectiveness of the adversarial training, we conduct an ablation experiment by disabling the adversarial loss $L_{adv}$. As shown in Table 7, the $L_{adv}$ increases representation quality and makes the generated distribution closer to the real one. We believe the $L_{adv}$ is necessary for visual effects, because it achieves more lifelike images.

### 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 $L_{adv}$. As shown in Table 7, the $L_{adv}$ increases representation quality and makes the generated distribution closer to the real one. We believe the $L_{adv}$ 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 ± 6.7 | 7.9 ± 3.4 | 37.6 ± 6.3 | 38.6 ± 6.5 | 6.8 ± 2.8 | 8.5 ± 3.2 | 10.4 ± 3.5 |
|              | Semi-Sup.   | 41.1 ± 1.3 | 16.2 ± 1.4 | 42.9 ± 2.1 | 43.9 ± 1.2 | 14.2 ± 1.2 | 15.5 ± 1.7 | 17.5 ± 1.2 |
| 100K         | Sup.        | 72.6 ± 0.3 | 55.2 ± 1.2 | 79.4 ± 1.1 | 75.3 ± 0.8 | 45.4 ± 0.9 | 46.7 ± 1.0 | 47.6 ± 1.0 |
|              | Semi-Sup.   | 73.6 ± 0.5 | 55.3 ± 1.0 | 79.9 ± 1.0 | 75.6 ± 0.5 | 45.6 ± 0.8 | 46.8 ± 1.5 | 47.9 ± 0.4 |
| 1M           | Sup.        | 84.1 ± 0.5 | 73.1 ± 0.2 | 88.2 ± 0.6 | 86.4 ± 1.0 | 60.5 ± 0.8 | 59.5 ± 1.5 | 58.9 ± 0.8 |
|              | Semi-Sup.   | 84.1 ± 0.6 | 73.1 ± 1.0 | 89.2 ± 1.1 | 86.5 ± 0.9 | 62.1 ± 1.1 | 63.7 ± 2.8 | 59.7 ± 0.6 |
| 6M           | Sup.        | 86.6 ± 0.5 | 79.6 ± 0.6 | 91.5 ± 0.7 | 89.0 ± 0.3 | 68.3 ± 0.7 | 71.9 ± 1.8 | 66.2 ± 0.6 |
|              | Semi-Sup.   | 87.5 ± 0.3 | 80.6 ± 0.5 | 91.8 ± 0.7 | 89.9 ± 0.6 | 68.3 ± 1.1 | 71.4 ± 1.7 | 66.2 ± 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 [20] score. We average the word accuracies (%) of IIIT5K, IC03 and IC13.

| Method | IIIT5K Acc. | SVT Acc. | IC03 Acc. | IC13 Acc. |
|--------|-------------|----------|-----------|-----------|
| AdaIN  | 9.7 | 21.9  | 9.1 | 18.7 |
| SimAN  | 22.2 | 39.7  | 22.3 | 38.6 | 24.1 | 43.6 |

Table 8. Probe evaluation of AdaIN and SimAN.

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

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

7. Compare with AdaIN

It is known that the AdaIN [22, 29] 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.

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 [22, 25].

We present the discriminator in Table 11.

10. Data Synthesis

Following the pipeline of SynthText [17], we simply render a text on a clean canvas. The fonts are publicly available\(^5\). We follow the strict setting proposed by Long et al. [35] 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.

```python
import albumentations as A
A.OneOf([A.ImageCompression(40, 80), A.Blur(blur_limit=[5, 11])], p=0.1)
```

\(^5\)https://fonts.google.com/
Table 10. Architecture of the decoder in the Section of Generative Visual Task.

| Layers | Configurations | Size       |
|--------|----------------|------------|
| Input  | Feature Map    | 512 × 8 × 8|
| Conv1  | c: 256 k: 3 Reflection Pad: 1, ReLU | 256 × 8 × 8 |
| Upsample | Ratio: ×2, Mode: “nearest” | 256 × 16 × 16 |
| Conv2  | c: 256 k: 3 Reflection Pad: 1, ReLU | 256 × 16 × 16 |
| Conv3  | c: 256 k: 3 Reflection Pad: 1, ReLU | 256 × 16 × 16 |
| Conv4  | c: 256 k: 3 Reflection Pad: 1, ReLU | 256 × 16 × 16 |
| Conv5  | c: 128 k: 3 Reflection Pad: 1, ReLU | 128 × 16 × 16 |
| Upsample | Ratio: ×2, Mode: “nearest” | 128 × 32 × 32 |
| Conv6  | c: 128 k: 3 Reflection Pad: 1, ReLU | 128 × 32 × 32 |
| Conv7  | c: 64 k: 3 Reflection Pad: 1, ReLU | 64 × 32 × 32 |
| Upsample | Ratio: ×2, Mode: “nearest” | 64 × 64 × 64 |
| Conv8  | c: 64 k: 3 Reflection Pad: 1, ReLU | 64 × 64 × 64 |
| Conv9  | c: 3 k: 3 Reflection Pad: 1, ReLU | 3 × 64 × 64 |
| Tanh   | -              | 3 × 64 × 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: 2 p: 1 PReLU |
| Conv5  | c: 1 k: 4 s: 1 p: 1 PReLU |

11. Arbitrary-Length Text Editing

We follow the same setting as EditText [58] 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 [17]. 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 $K$, respectively.
respectively. Besides, we obtain their style representations $\mu_{\text{source}}$, $\sigma_{\text{source}}$, $\mu_{\text{target}}$ and $\sigma_{\text{target}}$.

### 12.1. Color Interpolation

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

$$
\mu_{\text{rearrange}} = \mu_{\text{target}} \text{Softmax} \left( \frac{K^TQ}{\sqrt{dk}} \right),
$$

$$
\sigma_{\text{rearrange}} = \sigma_{\text{target}} \text{Softmax} \left( \frac{K^TQ}{\sqrt{dk}} \right).
$$

(19)

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

$$
\mu' = (1 - \alpha)\mu_{\text{source}} + \alpha\mu_{\text{rearrange}},
$$

$$
\sigma' = (1 - \alpha)\sigma_{\text{source}} + \alpha\sigma_{\text{rearrange}}, \alpha \in [0, 1].
$$

(20)

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

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

(21)

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

(22)

### 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:

$$
\mu_{\text{rearrange}} = \mu_{\text{source}} \text{Softmax} \left( \frac{Q^TK}{\sqrt{dk}} \right),
$$

$$
\sigma_{\text{rearrange}} = \sigma_{\text{source}} \text{Softmax} \left( \frac{Q^TK}{\sqrt{dk}} \right).
$$

(23)

The target glyph can be presented as:

$$
K'_{c,i,j} = K_{c,i,j}\sigma_{c,i,j} + \mu_{c,i,j}.
$$

(24)

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:

$$
Q'_{c,i,j} = (1 - \alpha)Q_{c,i,j} + \alpha K'_{c,i,j}, \alpha \in [0, 1].
$$

(25)

Finally, we obtain an image:

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

(26)