# CMT-Co: Contrastive Learning with Character Movement Task for Handwritten Text Recognition

Xiaoyi Zhang[1][0000−0002−5345−2110], Jiapeng Wang[1][0000−0002−2060−3488], Lianwen Jin[1,2⋆][0000−0002−5456−0957], Yujin Ren[1][0000−0001−9647−6713], and Yang Xue[1][0000−0002−1947−4957]

[1] South China University of Technology, Guangzhou, China
[2] SCUT-Zhuhai Institute of Modern Industrial Innovation, Zhuhai, China
{xy_zhang08,scutjpwang}@foxmail.com,
{lianwen.jin, yujinren98}@gmail.com, yxue@scut.edu.cn

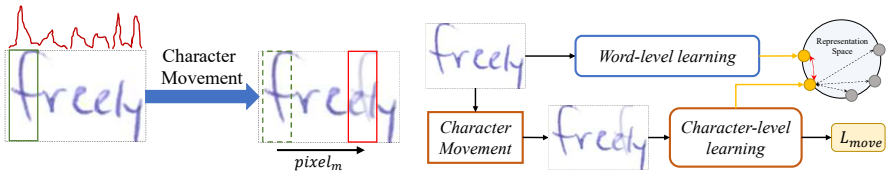
**Abstract.** Mainstream handwritten text recognition (HTR) approaches require large-scale labeled data for training to achieve satisfactory performance. Recently, contrastive learning has been introduced to perform self-supervised training on unlabeled data to improve representational capacity. It minimizes the distance between the positive pairs while maximizing their distance to the negative ones. Previous studies typically consider each frame or a fixed window of frames in a sequential feature map as a separate instance for contrastive learning. However, owing to the arbitrariness of handwriting and the diversity of word length, such modeling may contain the information of multiple consecutive characters or an over-segmented sub-character, which may confuse the model to perceive semantic clues information. To address this issue, in this paper, we design a character-level pretext task termed *C*haracter *M*ovement *T*ask, to assist word-level *co*ntrastive learning, namely *CMT-Co*. It moves the characters in a word to generate artifacts and guides the model to perceive the text content by using the moving direction and distance as supervision. In addition, we customize a data augmentation strategy specifically for handwritten text, which significantly contributes to the construction of training pairs for contrastive learning. Experiments have shown that the proposed CMT-Co achieves competitive or even superior performance compared to previous methods on public handwritten benchmarks.

**Keywords:** Self-supervised learning · Pretext task · Handwritten text recognition · Contrastive learning.

## 1 Introduction

Handwritten text recognition (HTR) is a vital field in computer vision [5, 16, 35, 32, 39, 38, 37, 29, 2, 36]. Most current methods for handwritten text recogni-

⋆ Corresponding author.




(a) Character Movement Task (CMT). (b) Overall Framework of CMT-Co.

**Fig. 1.** Illustration of CMT and our method CMT-Co. CMT-Co uses the CMT as an auxiliary task to assist contrastive learning, which contains word-level learning and character-level learning. The red arrow in the representation space represents minimizing the distance of positive pairs, while the black dashed arrows in the representation space represent maximizing the distance between negative samples.

tion [41, 25, 8, 26, 42] require full supervision, which not only takes much annotation time but also needs expensive costs. Moreover, with the advent of the era of big data and the development of network information technology, data acquisition is becoming easier, resulting in an exponential increase in the amount of unlabeled data. Therefore, it is necessary to explore how to effectively utilize them without manual annotation.

Self-supervised learning provides a solution to this problem and has been widely studied [11, 3, 15, 7, 6, 46, 33, 20, 31, 21]. It aims to learn representations from the data itself, without manual annotation. Features learned through the self-supervised process are then fine-tuned in specific downstream tasks to speed up convergence or achieve better performance, while greatly reducing the amount of data annotation.

Early self-supervised methods often designed pretext tasks [11, 46, 33, 9, 31, 44, 22, 18]. They focused on discovering tasks in which labels can be derived from prior knowledge or manual modification of data. For example, Gidaris *et al.* [11] rotated the original image by specific angles, and then let the network deal with the rotation prediction task. It is worth noting that almost no pretext task has been specifically designed for handwritten text recognition.

In recent years, self-supervised methods based on contrastive learning have received considerable attention [24, 11, 6, 3, 15, 7, 17, 30, 4]. Contrastive learning aims to minimize the distance between the positive pairs while maximizing their distance to the negative ones to achieve feature learning. A few related studies have introduced this concept into the field of handwritten text recognition [1, 23]. Aberdam *et al.* [1] proposed SeqCLR, which creates positive and negative samples by sliding a window over sequential feature maps and then performs contrastive learning on them. This method must guarantee sequence alignment between different augmented views of the same image, and consequently, the data augmentation method of SeqCLR is limited. Liu *et al.* [23] proposed PerSec, which learns both low- and high-level features through contrastive learning from shallow and deep feature maps. It aims to enable each element of sequential features to distinguish itself from the context. These methods for handwritten text recognition typically consider each frame or a fixed window of frames in a

sequential feature map as a separate instance for contrastive learning. However, owing to the arbitrariness of handwriting and the diversity of word length, such modeling may contain the information of multiple consecutive characters or an over-segmented sub-character, which may confuse the model to perceive semantic clues information.

To address this issue, in this paper, we design a character-level pretext task called the Character Movement Task (CMT), which is suitable for handwritten text. The handwritten text has unique prior knowledge, such as its vertical and horizontal projection distributions, which were applied to the traditional document text line and word segmentation methods [14, 34]. For handwritten word images, we first use vertical projection distribution, denoted by the red line in Figure 1(a), to estimate the approximate position of each character. Then, some characters are selected to move, and the artifact is generated. Note that the artifact does not change the meaning of the word. Finally, the network is required to predict the moving direction and distance. To solve the Character Movement Task, the network needs to recognize the moving characters and achieve the purpose of character-level feature learning.

To utilize both word-level and character-level semantic information, we adopt CMT as an auxiliary task to assist the contrastive learning of the entire word image, termed *CMT-Co*. In addition, to enhance the effectiveness of contrastive learning, we customize a data augmentation strategy for handwritten text, named Text-Aug. It includes four aspects: affine transformation, stroke jitter, stroke overlap, and stroke thickness. Text-Aug can sufficiently provide variety for contrastive learning frameworks.

Experimental results have shown that our method achieves competitive or even superior performance on public handwritten benchmarks compared to the previous self-supervised method for text recognition.

The main contributions of the paper are as follows:

(1) We propose a character-level pretext task for handwritten text, called the Character Movement Task (CMT), which is combined with the word-level contrastive learning process. To the best of our knowledge, this is the first pretext task in the field of handwritten text recognition.
(2) We propose a data augmentation strategy named Text-Aug, including affine transformation, stroke jitter, stroke overlap, and stroke thickness, to further unlock the power of our framework.
(3) The overall frameworks CMT-Co achieve competitive or even superior performance compared to the previous self-supervised method in terms of representation quality and downstream fine-tuning of handwritten text.

## 2 Related works

### 2.1 Pretext Task

Early self-supervised methods often designed self-supervised tasks, also called pretext tasks [11, 46, 33, 9, 31]. Self-supervised tasks have been studied extensively [44, 10]. They focus on discovering tasks in which labels can be derived

from prior knowledge or manual modification of data. For example, in image classification, Gidaris *et al.* [11] rotated the original image by specific angles and then let the network deal with the rotation prediction task. Doersch *et al.* [9] extracted random pairs of patches from each image and drove the network to predict the position of the second patch relative to that of the first.

However, to the best of our knowledge, currently in the field of handwritten text recognition, there is no method for designing a pretext task based on unique prior knowledge of the handwritten text.

### 2.2 Contrastive Learning

In addition to designing specific pretext tasks, recent self-supervised methods based on contrastive learning have shown significant potential [15, 6, 3, 12, 1, 23]. SimCLR [3] generates negative samples through a large batch size. Furthermore, because of rich data augmentation, it learns sufficient representations after contrastive learning and shows pleasing performance in downstream tasks. MoCo [15] designed a symmetric structure, and then one party generated negative samples through momentum update, which not only did not need to store all the data in advance but also did not require a large batch size to generate negative samples. BYOL [12] designed an asymmetric structure, and a pleasing performance could be achieved without negative samples.

In the field of handwritten text recognition, SeqCLR [1] improves upon SimCLR [3], which uses the instance-mapping function on the sequential feature map to generate positive and negative samples. Multiple instances are generated from a single image through a sliding window. The same position in the same image with different data augmentation is a positive pair. Different positions and other images are negative samples. However, this method must ensure sequence alignment; otherwise, the same position of the same image with different data augmentation may not contain the same characters, which greatly limits the data augmentation method of SeqCLR. PerSec [23] learned both low- and high-level features through contrastive learning from shallow and deep feature maps within one image. It aims to enable each element of the sequential features to distinguish itself from the context. In general, the instance of the aforementioned contrastive learning for handwritten text recognition may contain the information of multiple consecutive characters or an over-segmented sub-character, thus confusing the model in perceiving semantic information.

Therefore, this paper will use prior knowledge of handwritten word text to design a character-level Character Movement Task, and then assist the word-level learning of the entire word image in contrastive learning to achieve better results on handwritten text.

## 3 Method

### 3.1 Text-Aug

It is known from recent self-supervised studies on contrastive learning [3, 15, 6, 12, 40] that data augmentation plays an important role in feature represen-

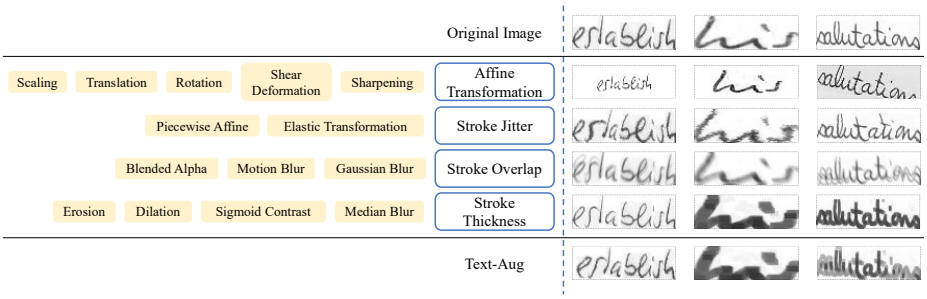



**Fig. 2.** Composition of Text-Aug and examples of different parts.

tation learning. Thus, we designed a data augmentation strategy suitable for handwritten text.

It is well known that text is composed of strokes. For handwritten text, the same word or character may have different sizes due to different writers' styles. A stroke of the same character may also have different degrees of bending or jitting distortions. Moreover, when people write, they may not erase the wrongly written characters and choose to overwrite the correct characters directly on them, which will result in many overlapping strokes. Due to different writing equipment and writing strength, strokes of the same character may be inconsistent in thickness, and handwriting may be separated or stuck together.

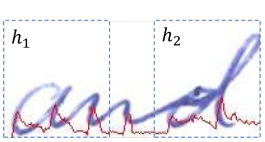
Therefore, for the above situation, we design *Text-Aug* for handwritten text. It includes four types of text augmentation: affine transformation, stroke jitter, stroke overlap, and stroke thickness. Figure 2 gives some examples of these four types of text augmentation in Text-Aug. The affine transformations include scaling, translation, rotation, shear deformation, and sharpening. This data augmentation provided text images of different scales, positions, and brightness. The stroke jitter includes piecewise affine and elastic transformation. They can simulate the bending and jittering of strokes in a text. Data augmentation for stroke overlap simulates the overlapping and blurring of strokes in text images. It includes blended alpha, motion blur, and Gaussian blur. The stroke thickness contains erosion, dilation, sigmoid contrast, and median blur. These augmentations can change the thickness of the strokes and may also produce modifications in character sticking and separation.

It can be seen that Text-Aug greatly improves the diversity of the text, which can enable the network to distinguish more variable texts. Pseudo-code and more examples of Text-Aug are shown in the Appendix.

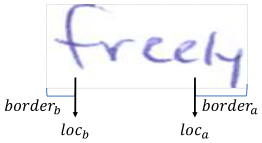
### 3.2 Character Movement Task

It can intuitively know some prior knowledge from the handwritten word image, such as the vertical projection distribution, which can give the approximate location of the characters in the image. The Character Movement Task (CMT)


(a) Illustration of Character Localization.




(b) Character Selection ($|U| = 1$).




(c) Determination of $border_b$ and $border_a$.

**Fig. 3.** Process illustration of Character Movement Task (CMT).

first roughly locates the characters through the vertical projection distribution, then moves the characters, and finally, drives the network to predict the movement direction and the moving distance of the characters. To solve the Character Movement Task, the network needs to recognize the moving characters and achieve the purpose of character-level feature learning. The process of the CMT is shown in Algorithm 1 and explained in detail below. We assume that each word image $I$ has characters.

**Character Localization** Because this task requires moving characters, we first need to locate the position of each character in the word image. For the handwritten grayscale word image, it is resized to $H \times W$ and then adaptively binarized and normalized to $[0, 1]$. Note that the value of the area where the characters are located is one. Finally, row summation is performed to obtain the vertical projection distribution $Sta$ of the text image. As shown in the first image of Figure 3(a), the red line represents the vertical projection distribution, indicating the projected cumulative value of the character pixel at the corresponding column position. According to $Sta$, we can locate the approximate position of the characters in the word image.

However, writers usually have continuous strokes between different characters in the word. Therefore, to approximately eliminating the interference of stroke adhesion, we set the number less than $t$ in $Sta$ to zero, where $t$ takes the second smallest value in $Sta$. As shown in the second image of Figure 3(a), the blue line represents the value of $t$, and the position below the blue line is set to zero, resulting in the third image of Figure 3(a). We define a continuous region with non-zero projected values as the character block region $u$. Taking the third image in Figure 3(a) as an example, there are three continuous regions with non-zero projection values, implying that there are three character block regions, $U = \{u_1, u_2, u_3\}$. It can be inferred from the process of character block region generation that each character block region contains characters.

**Character Selection** After locating the positions of the characters in the word image, we then need to select the characters to move. We define the center position of the selected moving characters as $loc_b$ and the moving target position as $loc_a$. The characters will move from $loc_b$ to $loc_a$. The position $loc_b$ and $loc_a$ are randomly selected from the character block region set $U$.

If $|U| = 1$, it means that there is only one character block region, $U = u_1$. If the movement is too small (such as the movement of one or two pixels), there is no difference to the naked eye, so it is unreasonable to force the network to predict the moving distance. Thus, the characters will move a certain distance here. As shown in Figure 3(b), we first denote the front 40% of $u_1$ as $h_1$ and the back 40% of $u_1$ as $h_2$. Then, we randomly select a position from each of $h_1$ and $h_2$. Finally, these two positions are randomly used as $loc_a$ and $loc_b$. If $|U| \geq 2$, it means there are two or more character block regions. In this case, two character block regions $u_b$ and $u_a$ are randomly selected from $U$ as the character block region before the movement and the character block region after the movement. Then, we randomly select a location from $u_b$ as $loc_b$ and randomly select a location from $u_a$ as $loc_a$.

After determining the center position of the selected moving characters, we need to specify the width of the characters. Firstly, we randomly sample a value from $[\frac{0.15}{2}W, \frac{0.25}{2}W]$ and regard it as half of the initial moving character area width, denoted as $w_{ini}$, where $W$ is the image width. Then the minimum distance between $loc_b$ and the image border is $border_b$, and the minimum distance between $loc_a$ and the image border is $border_a$, as shown in Figure 3(c). Half of the final moving character area width $w_{move}$ is the minimum value among $w_{ini}$, $border_b$, and $border_a$.

The selected character area is denoted as

$$img_b = I[0 : H, loc_b - w_{move} : loc_b + w_{move}] \tag{1}$$

The original image of the area to which the characters are moved is

$$img_a = I[0 : H, loc_a - w_{move} : loc_a + w_{move}] \tag{2}$$

Finally, the selected character area is superimposed on the image at a scale of $1 - \lambda$, which means that

$$img_a = \lambda img_a + (1 - \lambda) img_b \tag{3}$$

**Algorithm 1** Process of the Character Movement Task

**Input:** Grayscale image $I$

**Output:** Image $MI$ after character movement, Label $y$

1. Obtain the vertical projection distribution $Sta$ of image $I$.
2. Obtain the character block region set $U$.
3. Randomly select the position $loc_b$ and $loc_a$ from the character block region set $U$. The characters will move from $loc_b$ to $loc_a$.
4. Determine the width $w_{move}$ of the character movement, and then superimposes the selected character area on the target position to get image $MI$.
5. Determine the label $y$ of the character movement.

The rest of $I$ remains unchanged. We set image $I$ after character movement as $MI$, which does not change the meaning of the initial word image $I$.

**Loss Function** Character Movement Task is defined as a classification task. The label $y$ is given by the formula (4).

$$y = pixel_m + W \tag{4}$$

where $W$ denotes the image width and $pixel_m = loc_a - loc_b$. When $pixel_m = 0$, there is no movement in the image, while $pixel_m < 0$, means that the character moves to the left; and $pixel_m > 0$, means that the character moves to the right.

Because the image has been resized before the character movement, the value range of $pixel_m$ is $[-W, W]$. Thus, the number of categories for classification is $2W + 1$. The loss in the Character Movement Task is given by equation (5).

$$\mathcal{L}_{move} = -\sum_{i=1}^{N} y_i \log p_i \tag{5}$$

where

$$p_i = \frac{e^{F(MI_i)}}{\sum_{j=1}^{2W+1} e^{F(MI_j)}} \tag{6}$$

where $F$ represents the encoder and the multilayer perceptron and $N$ represents the mini-batch size.

Figure 4 shows the $pixel_m$ distribution generated by the Character Movement Task on the IAM training set. It can be seen that the range of movement to the left and right is roughly equal and the distribution is roughly balanced. Figure 5 shows some examples of the Character Movement Task.

### 3.3 Overall Framework

Our CMT-Co method uses CMT as a character-level auxiliary task to assist in the learning of the entire word-level image in contrastive learning. MoCo v2 [6] is

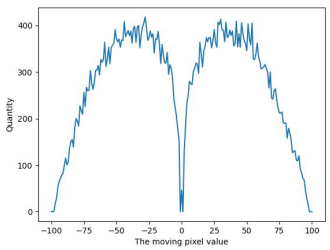



**Fig. 4.** Distribution of the $pixel_m$ generated by the CMT.

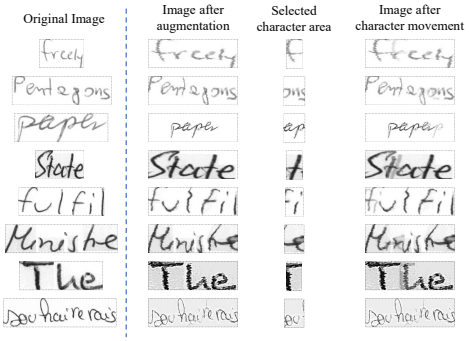



**Fig. 5.** Examples of character movement.

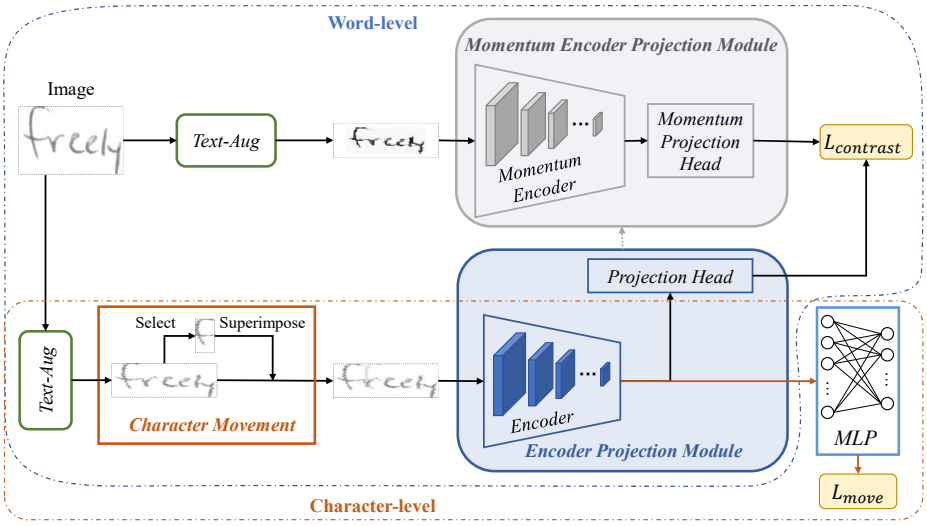



**Fig. 6.** Overall framework of our method CMT-Co. CMT-Co uses the CMT as a character-level auxiliary task to assist the learning of the entire word-level image in contrastive learning. Noted that the labels used to calculate the loss $L_{move}$ are generated by the Character Movement module.

selected as the basic architecture. The overall architecture of CMT-Co is shown in Figure 6. In CMT-Co, the high-level semantic information of the word can be learned through the contrastive learning of the whole-word image, and the low-level single-character information can be learned through the CMT.

We use Text-Aug to augment the text images. However, due to the overlapping phenomenon of the Character Movement Task, we remove the data augmentation method that produces overlapping characters to prevent ambiguity and interference with the Character Movement Task learning. Similar to MoCo [15], we predefine a queue with a large length and initialize the queue randomly. In the training process of the network, the current mini-batch is enqueued and the oldest mini-batch is dequeued. For each feature vector generated by the encoder projection module, the positive sample is the feature vector generated by the momentum encoder projection module after different data augmentation of the same image and the negative samples are the feature vectors in the queue. The feature vector in the queue is extracted by the momentum encoder projection module; therefore, it should be relatively consistent with the features extracted by the encoder projection module. Inspired by MoCo v2 [6], the momentum encoder projection module adopts the same structure as the encoder projection module, but it does not share parameters. Formally, we denote the parameters of the momentum encoder projection module as $\theta_v$ and those of the encoder projection module as $\theta_q$, $\theta_v$ is initialized by $\theta_q$ and updated by

$$m\theta_v + (1-m)\theta_q \rightarrow \theta_v \tag{7}$$

where $m \in [0, 1)$ is a momentum coefficient.

The encoder and the momentum encoder are CNN-based networks. The projection and momentum projection heads are multilayer perceptrons with a hidden layer, and they are used to map the visual representation to the contrastive space. The multilayer perceptron (MLP) for the Character Movement Task consists of two fully connected layers, which mainly convert the visual feature map into a vector for classification.

The total loss of CMT-Co can be formulated as (8).

$$\mathcal{L} = \mathcal{L}_{contrast} + \alpha \mathcal{L}_{move} \tag{8}$$

where

$$\mathcal{L}_{contrast} = -\log \frac{\exp\left(MI_q \cdot k_+ / \tau\right)}{\sum_{i=1}^{C} \exp\left(MI_q \cdot k_i / \tau\right)} \tag{9}$$

Here, $C$ is the queue size, $\tau$ is a temperature hyper-parameter [43], $MI_q$ is the feature vector output by the encoder projection module, $k_+$ is the feature vector output by the momentum encoder projection module and $k_i$ is the feature vector in the queue. $\alpha$ is a hyper-parameter that is adopted for weighting loss items.

## 4 Experiments

### 4.1 Implementation Details

**Datasets and Metrics** We conduct our experiments on public handwritten benchmarks, which are IAM [28], RIMES [13], and CVL [19]. We use word-level accuracy as the evaluation metric for all experiments. We follow the dataset setting of SeqCLR [1] for pre-training and downstream fine-tuning, which only uses the training set when training. All images are resized to $32 \times 100$ as input size.

**Pre-training and Downstream Fine-tuning Settings** We conduct all experiments on one A40 GPU with a mini-batch size of 256. For *pre-training*, we adopt the same settings as MoCo v2 [6]. We use SGD as the optimizer with an initial learning rate of 0.03. For the CMT, the superposition scale $\lambda$ in formula (3) is 0.7. The hyper-parameter $\alpha$ in formula (8) is 0.2. The momentum coefficient $m$ in formula (7) is 0.999. The momentum encoder and the encoder is

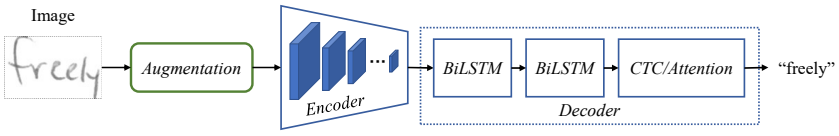



**Fig. 7.** Framework for downstream task training. We follow the framework of SeqCLR [1].

**Table 1.** Representation quality. We freeze the encoder and only train the decoder. **Bold** represents the best result, and underlined represents the second-best result. "Ft Aug." represents the augmentation during downstream fine-tuning. "Seq. Aug." represents the augmentation in SeqCLR [1]. "MoCo v2$^{\dagger}$" indicates that our data augmentation method Text-Aug is used in pre-training.

| Method | Ft Aug. | Decoder | IAM | RIMES | CVL |
|---|---|---|---|---|---|
| Baseline | Seq. Aug. [1] | CTC | 29.9 | 22.7 | 24.3 |
| SimCLR [3] | | | 4.0 | 10.0 | 1.8 |
| SeqCLR [1] | | | 39.7 | 63.8 | 66.7 |
| MoCo v2$^{\dagger}$ [6] | | | 51.7 | **66.9** | 71.1 |
| CMT-Co(Ours) | | | **53.1** | 66.0 | **72.1** |
| Baseline | Seq. Aug. [1] | Attention | 33.9 | 28.7 | 35.0 |
| SimCLR [3] | | | 16.0 | 22.0 | 26.7 |
| SeqCLR [1] | | | 51.9 | **79.5** | 74.5 |
| MoCo v2$^{\dagger}$ [6] | | | 56.3 | 75.1 | 74.1 |
| CMT-Co(Ours) | | | **58.2** | 74.7 | **74.7** |

ResNet29 [8], which is the same as the backbone of SeqCLR [1]. The MLP for the Character Movement Task is two fully connected layers. The projection head and the momentum projection head are multilayer perceptrons with a hidden layer. We train the model for 68K iterations for each dataset, which takes about 20 hours.

For *downstream fine-tuning*, we follow the settings of SeqCLR [1] for a fair comparison. Specifically, we use the "encoder-decoder" paradigm in text recognition, in which there are two types of decoders: CTC-based decoder and attention-based decoder, as shown in Figure 7. We use Adadelta [45] optimizer with an initial learning rate of 2. To further demonstrate the effectiveness of Text-Aug, we also present results using our data augmentation approach for downstream fine-tuning.

### 4.2 Experimental Results

In this section, we quantitatively verify the effectiveness of the proposed method. We compare our method with previous self-supervised methods based on contrastive learning ( i.e. SeqCLR [1], PerSec [23]) on the handwritten dataset. We first conduct experiments on the representation quality to validate the representational ability of the encoder parameters learned from the pre-training stage. We then conduct experiments on a certain proportion of labeled training data to further confirm the effect of our method on fine-tuning with few data.

**Representation Quality** To examine the quality of encoder representations learned by pre-training, we follow SeqCLR [1] and PerSec [23], which freeze the parameters of the encoder and only train the decoder. The decoder is all randomly initialized. For Baseline, we initialize the encoder randomly, but for other methods, we use pre-trained parameters to initialize the encoder.

**Table 2.** Downstream Task Fine-tuning. We train the entire model including the encoder and the decoder. **Bold** represents the best result, and underlined represents the second-best result. "Ft Aug." represents the augmentation during downstream fine-tuning. "Seq. Aug." represents the augmentation in SeqCLR [1]. "MoCo v2†" indicates that our data augmentation method Text-Aug is used in pre-training.

| Method | Ft Aug. | Decoder | IAM | | | RIMES | | | CVL | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 5% | 10% | 100% | 5% | 10% | 100% | 5% | 10% | 100% |
| PerSec-CNN [23] | Luo *et al.* [27] | CTC | - | - | 77.9 | - | - | - | - | - | 78.1 |
| SimCLR [3] | Seq. Aug. [1] | CTC | 15.4 | 21.8 | 65.0 | 36.5 | 52.9 | 84.5 | 52.1 | 62.0 | 74.1 |
| SeqCLR [1] | Seq. Aug. [1] | CTC | 31.2 | 44.9 | 76.7 | 61.8 | 71.9 | 90.1 | 66.0 | 71.0 | 77.0 |
| Baseline | Seq. Aug. [1] | CTC | 35.8 | 44.2 | 78.6 | 56.7 | 67.7 | 87.0 | 63.0 | 70.6 | 77.7 |
| MoCo v2† [6] | Seq. Aug. [1] | CTC | 42.2 | 51.7 | 79.6 | 63.2 | 72.6 | 89.4 | 67.3 | 72.7 | 77.8 |
| CMT-Co (Ours) | Seq. Aug. [1] | CTC | 41.7 | 52.0 | 79.9 | 63.8 | 72.7 | 89.3 | 68.1 | 72.8 | 78.0 |
| CMT-Co (Ours) | Text-Aug | CTC | **47.7** | **56.0** | **80.1** | **67.1** | **75.4** | **90.1** | **71.9** | **74.5** | **78.2** |
| PerSec-CNN [23] | Luo *et al.* [27] | Attention | - | - | 80.8 | - | - | - | - | - | **80.2** |
| SimCLR [3] | Seq. Aug. [1] | Attention | 22.7 | 32.2 | 70.7 | 49.9 | 60.9 | 87.8 | 59.0 | 65.6 | 75.7 |
| SeqCLR [1] | Seq. Aug. [1] | Attention | 40.3 | 52.3 | 79.9 | **70.9** | 77.0 | **92.5** | 73.1 | 74.8 | 77.8 |
| Baseline | Seq. Aug. [1] | Attention | 40.5 | 49.8 | 79.5 | 62.1 | 72.7 | 90.0 | 68.4 | 73.8 | 77.3 |
| MoCo v2† [6] | Seq. Aug. [1] | Attention | 44.2 | 53.3 | 80.7 | 66.8 | 75.7 | 90.8 | 70.2 | 74.9 | 78.6 |
| CMT-Co (Ours) | Seq. Aug. [1] | Attention | 45.5 | 53.4 | 81.3 | 66.9 | 76.0 | 90.4 | 71.5 | 75.1 | 78.5 |
| CMT-Co (Ours) | Text-Aug | Attention | **50.4** | **55.8** | **81.9** | 68.8 | **77.6** | 91.2 | **73.6** | **76.2** | 78.7 |

Table 1 shows the representation quality comparison with our method, SimCLR [3], and SeqCLR [1] on IAM, RIMES, and CVL datasets. Combined with our data augmentation Text-Aug in pre-training, "MoCo v2†" can achieve even better performance than SeqCLR [1], which is specifically designed for text recognition in most datasets. "MoCo v2†" is improved by 12% on the IAM dataset on the CTC-based decoder method compared to SeqCLR [1]. Then, assisted by CMT, CMT-Co achieves better performance on IAM and CVL datasets compared to "MoCo v2†". This proves that our Character Movement Task designed with prior knowledge of the handwritten text is indeed beneficial for self-supervised representation learning. Finally, our method CMT-Co outperforms SeqCLR [1] on most datasets, with the most significant improvement on the IAM dataset with a gain of 13.4%.

**Downstream Task Fine-tuning** To further demonstrate the pre-trained model representation ability, we explore the performance of the model on a small amount of labeled data. In Table 2, we present the results for 5%, 10%, and 100% of the labeled training data, respectively. In fine-tuning, we train the entire model, including the encoder and decoder. The encoder and decoder parameters of the baseline are initialized randomly. The encoders of other methods load the pre-trained parameters, and the decoder parameters are initialized randomly.

Table 2 shows the downstream task fine-tuning comparison with our method, SimCLR [3], SeqCLR [1], and PerSec [23]. It can be seen that in most datasets, combined with our data-augmented Text-Aug in pre-training, "MoCo v2†" can achieve comparable performance to SeqCLR [1] and PerSec [23], which are specially designed for text recognition. Assisted by CMT, CMT-Co has further

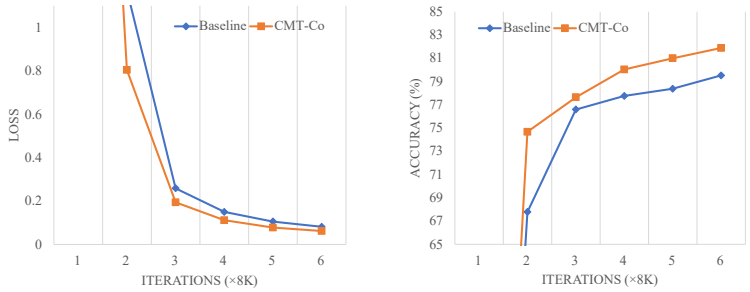



**Fig. 8.** Training loss and word-level accuracy between baseline and CMT-Co during downstream fine-tuning with an attention-based decoder on the IAM dataset.

improvements on most scales of datasets. Eventually, when augmenting data using Text-Aug during fine-tuning, our method improves again, achieving best or second-best results on all scales of datasets. With the dual power of Text-Aug and CMT, our method achieves promising performance on the IAM dataset.

Figure 8 shows training loss and word-level accuracy between baseline and CMT-Co during downstream fine-tuning with an attention-based decoder on the IAM dataset. It can be seen that the encoder parameters pre-trained by CMT-Co indeed speed up the convergence and achieve better results.

### 4.3 Ablation Study

In this section, we first explore the effect of important parts in CMT-Co, including CMT and data augmentation Text-Aug. Then, we perform ablation of the important parameters $\lambda$ and loss function in CMT, where $\lambda$ determines the superposition scale. More ablations to verify the effectiveness of the proposed method are shown in the Appendix.

We conduct ablation experiments on the IAM dataset. We use attention for the decoder and the data augmentation in SeqCLR [1] during fine-tuning. Table 3 explores the ablation of two parts of CMT-Co. It can be seen that Text-Aug performs better than "MoCo v2-Aug." and "Seq. Aug.", verifying the effectiveness of our Text-Aug strategy. Assisted by the pretext task CMT, the performance

**Table 3.** The ablation of data augmentation and CMT. "Pre-training Aug." represents the augmentation during pre-training. "MoCo v2-Aug." represents the augmentation in MoCo v2 [6]. "Seq. Aug." represents the augmentation in SeqCLR [1].

| Method | Pre-training Aug. | Pretext Task | Accuracy |
|---|---|---|---|
| MoCo v2 [6] | MoCo v2-Aug. [6] | ✗ | 80.2 |
| MoCo v2* [6] | Seq. Aug. [1] | ✗ | 79.8 |
| MoCo v2† [6] | Text-Aug | ✗ | 80.7 |
| CMT-Co (Ours) | Text-Aug | CMT | **81.3** |

**Table 4.** The ablation of $\lambda$.

| $\lambda$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| Accuracy | 80.7 | 80.0 | 80.5 | **81.3** | 80.9 |

**Table 5.** The ablation of different loss functions for the CMT.

| Loss | CE | MSE |
|---|---|---|
| Accuracy | **81.3** | 72.5 |

of the CMT-Co method is further improved, showing the effectiveness of CMT. The influence of $\lambda$ that determines the superposition scale is shown in Table 4. The performance of CMT-Co reaches its best when $\lambda$ is 0.7.

Although in this paper, we define the Character Movement Task as a classification task, it is also reasonable for CMT to be defined as a regression task. Therefore, we also explored its effect as a regression task. The ablation of different loss functions of the Character Movement Task is studied in Table 5. "CE" refers to the cross-entropy loss, which means we regard CMT as a classification task, and the equation is shown in (5). "MSE" refers to the mean-squared loss, which means we regard CMT as a regression task, and the equation is shown in (10).

$$\mathcal{L}_{move} = \|y_{pred} - y\|^2 \tag{10}$$

where $y_{pred}$ is the prediction of the network and $y$ is generated by CMT.

It can be seen that when CMT is used as a classification task, the performance is better. We believe that CMT is different from reconstruction and generation tasks, which reduce the penalty when the predicted distance gap becomes smaller. To better learn the representation of the character, the position of the character should be accurately predicted.

## 5 Conclusion

In this paper, we propose a Character Movement Task (CMT) to learn character-level features based on prior knowledge of the handwritten text. Our method CMT-Co uses the CMT as a character-level auxiliary task to assist the learning of the entire word-level image in contrastive learning. Furthermore, to better improve the performance of contrastive learning, we also propose a data augmentation strategy named Text-Aug which is suitable for handwritten text. Experiments show that our method can achieve comparable performance or even better performance than existing self-supervised methods for text recognition.

In the future, it is worth exploring more effective text-related pretext tasks and multi-task self-supervised learning methods. In addition, we hope that this work can inspire more research on self-supervised learning for text recognition, which has not been well investigated in the literature.

**Acknowledgment** This research is supported in part by NSFC (Grant No.: 61936003), GD-NSF (no.2017A030312006, No.2021A1515011870), Zhuhai Industry Core and Key Technology Research Project (no. ZH22044702200058PJL), and the Science and Technology Foundation of Guangzhou Huangpu Development District (Grant 2020GH17)

## References


1. Aberdam, A., Litman, R., Tsiper, S., Anschel, O., Slossberg, R., Mazor, S., Manmatha, R., Perona, P.: Sequence-to-sequence contrastive learning for text recognition. In: CVPR. pp. 15302–15312 (2021)
2. Bhunia, A.K., Ghose, S., Kumar, A., Chowdhury, P.N., Sain, A., Song, Y.Z.: MetaHTR: Towards writer-adaptive handwritten text recognition. In: CVPR. pp. 15830–15839 (2021)
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML. pp. 1597–1607 (2020)
4. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.E.: Big self-supervised models are strong semi-supervised learners. In: NeurIPS. pp. 22243–22255 (2020)
5. Chen, X., Jin, L., Zhu, Y., Luo, C., Wang, T.: Text recognition in the wild: A survey. ACM Computing Surveys **54**(2), 1–35 (2021)
6. Chen, X., Fan, H., Girshick, R.B., He, K.: Improved baselines with momentum contrastive learning. CoRR **abs/2003.04297** (2020)
7. Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR. pp. 15750–15758 (2021)
8. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: ICCV. pp. 5076–5084 (2017)
9. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV. pp. 1422–1430 (2015)
10. Gan, Y., Han, R., Yin, L., Feng, W., Wang, S.: Self-supervised multi-view multi-human association and tracking. In: ACM Int. Conf. Multimedia. pp. 282–290 (2021)
11. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)
12. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. In: NeurIPS. vol. 33, pp. 21271–21284 (2020)
13. Grosicki, E., El Abed, H.: ICDAR 2009 handwriting recognition competition. In: ICDAR. pp. 1398–1402 (2009)
14. Ha, J., Haralick, R.M., Phillips, I.T.: Document page decomposition by the bounding-box project. In: ICDAR. pp. 1119–1122 (1995)
15. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR. pp. 9729–9738 (2020)
16. Ingle, R.R., Fujii, Y., Deselaers, T., Baccash, J., Popat, A.C.: A scalable handwritten text recognition system. In: ICDAR. pp. 17–24 (2019)
17. Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F.: A survey on contrastive self-supervised learning. Technologies **9**(1), 2 (2020)
18. Kinakh, V., Taran, O., Voloshynovskiy, S.: ScatSimCLR: Self-supervised contrastive learning with pretext task regularization for small-scale datasets. In: ICCV Workshops. pp. 1098–1106 (2021)
19. Kleber, F., Fiel, S., Diem, M., Sablatnig, R.: CVL-DataBase: An off-line database for writer retrieval, writer identification and word spotting. In: ICDAR. pp. 560–564 (2013)
20. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: CVPR. pp. 1920–1929 (2019)

21. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1920–1929 (2019)
22. Li, W., Wang, G., Fidon, L., Ourselin, S., Cardoso, M.J., Vercauteren, T.: On the compactness, efficiency, and representation of 3d convolutional networks: Brain parcellation as a pretext task. In: IPMI. pp. 348–360 (2017)
23. Liu, H., Wang, B., Bao, Z., Xue, M., Kang, S., Jiang, D., Liu, Y., Ren, B.: Perceiving stroke-semantic context: Hierarchical contrastive learning for robust scene text recognition. In: AAAI. pp. 1702–1710 (2022)
24. Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., Tang, J.: Self-supervised learning: Generative or contrastive. IEEE Transactions on Knowledge and Data Engineering pp. 1–1 (2021)
25. Liu, Y., Chen, H., Shen, C., He, T., Jin, L., Wang, L.: ABCNet: Real-time scene text spotting with adaptive bezier-curve network. In: CVPR. pp. 9806–9815 (2020)
26. Luo, C., Jin, L., Sun, Z.: MORAN: A multi-object rectified attention network for scene text recognition. Pattern Recognition **90**, 109–118 (2019)
27. Luo, C., Zhu, Y., Jin, L., Wang, Y.: Learn to augment: Joint data augmentation and network optimization for text recognition. In: CVPR. pp. 13746–13755 (2020)
28. Marti, U.V., Bunke, H.: The IAM-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition **5**(1), 39–46 (2002)
29. Michael, J., Labahn, R., Grüning, T., Zöllner, J.: Evaluating sequence-to-sequence models for handwritten text recognition. In: ICDAR. pp. 1286–1293 (2019)
30. Misra, I., Maaten, L.v.d.: Self-supervised learning of pretext-invariant representations. In: CVPR. pp. 6706–6716 (2020)
31. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV. pp. 69–84 (2016)
32. Parvez, M.T., Mahmoud, S.A.: Offline arabic handwritten text recognition: A survey. ACM Computing Surveys **45**(2), 1–35 (2013)
33. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR. pp. 2536–2544 (2016)
34. Ptak, R., Żygadło, B., Unold, O.: Projection-based text line segmentation with a variable threshold. International Journal of Applied Mathematics and Computer Science **27**(1), 195–206 (2017)
35. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: ICDAR. pp. 67–72 (2017)
36. Sánchez, J.A., Bosch, V., Romero, V., Depuydt, K., de Does, J.: Handwritten text recognition for historical documents in the transcriptorium project. In: DATeCH. pp. 111—117. ACM (2014)
37. Sánchez, J.A., Romero, V., Toselli, A.H., Vidal, E.: ICFHR2014 competition on handwritten text recognition on transcriptorium datasets (HTRtS). In: ICFHR. pp. 785–790 (2014)
38. Sánchez, J.A., Romero, V., Toselli, A.H., Vidal, E.: ICFHR2016 competition on handwritten text recognition on the read dataset. In: ICFHR. pp. 630–635 (2016)
39. Sánchez, J.A., Romero, V., Toselli, A.H., Villegas, M., Vidal, E.: ICDAR2017 competition on handwritten text recognition on the read dataset. In: ICDAR. pp. 1383–1388 (2017)
40. Thoker, F.M., Doughty, H., Snoek, C.G.: Skeleton-contrastive 3d action representation learning. In: ACM Int. Conf. Multimedia. pp. 1655–1663 (2021)
41. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: ICCV. pp. 1457–1464 (2011)

42. Wang, T., Zhu, Y., Jin, L., Peng, D., Li, Z., He, M., Wang, Y., Luo, C.: Implicit Feature Alignment: Learn to convert text recognizer to text spotter. In: CVPR. pp. 5973–5982 (2021)
43. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: CVPR. pp. 3733–3742 (2018)
44. Yan, J., Wang, J., Li, Q., Wang, C., Pu, S.: Self-supervised regional and temporal auxiliary tasks for facial action unit recognition. In: ACM Int. Conf. Multimedia. pp. 1038–1046 (2021)
45. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. CoRR **abs/1212.5701** (2012)
46. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV. pp. 649–666 (2016)