Sequence-to-Sequence Contrastive Learning for Text Recognition

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

Ron Litman*
AWS
litmanr@amazon.com

Shahar Tsiper
AWS
tsiper@amazon.com

Oron Anschel
AWS
oronans@amazon.com

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

Shai Mazor
AWS
smazor@amazon.com

R. Manmatha
AWS
manmatha@amazon.com

Pietro Perona
Caltech and AWS
peronapp@amazon.com

Abstract

We propose a framework for sequence-to-sequence contrastive learning (SeqCLR) of visual representations, which we apply to text recognition. To account for the sequence-to-sequence structure, each feature map is divided into different instances over which the contrastive loss is computed. This operation enables us to contrast in a sub-word level, where from each image we extract several positive pairs and multiple negative examples. To yield effective visual representations for text recognition, we further suggest novel augmentation heuristics, different encoder architectures and custom projection heads. Experiments on handwritten text and on scene text show that when a text decoder is trained on the learned representations, our method outperforms non-sequential contrastive methods. In addition, when the amount of supervision is reduced, SeqCLR significantly improves performance compared with supervised training, and when fine-tuned with 100% of the labels, our method achieves state-of-the-art results on standard handwritten text recognition benchmarks.

1. Introduction

Contrastive learning techniques for self-supervised representation learning have recently demonstrated significant improvements on several semi-supervised computer vision applications, including image classification, object detection, and segmentation [9, 27, 18, 8, 26, 52, 10, 60, 49]. As illustrated in Fig. 1(a) contrastive learning is performed by maximizing agreement between representations of differently augmented views of the same image and distinguishing them from representations of other dataset images.

Despite obvious advantages, unsupervised and semi-supervised schemes have hardly been explored for text recognition ([25, 68, 34]). For example, currently, most handwritten text recognition approaches still rely on fully supervised learning, requiring large amounts of annotated data. The reason for this is simple: current contrastive schemes for visual representation learning are tailored to-

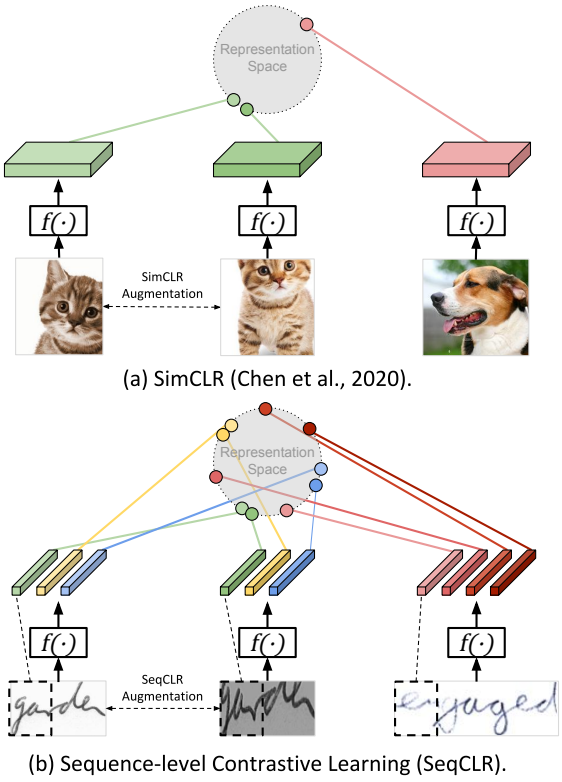

Figure 1: **Sequence contrastive learning.** (a) Current contrastive methods compare representations computed from whole images. (b) We propose a sequence-to-sequence approach, by viewing the feature map as a sequence of separate representations. This is useful in text recognition, where words are composed of sequences of characters.

*Authors contribute equally and are listed in alphabetical order.

wards tasks such as object recognition or classification, where images are atomic input elements. For example, in image classification, positive examples are created by augmenting each image while all other images in the dataset are assumed to be negative (Fig. 1(a)). On the other hand, for sequential prediction as used in text recognition, a word is viewed as a sequence of characters, and thus the image of a word is best modeled as a sequence of adjacent image slices (frames), each one of which may represent a different class as depicted in Fig. 1(b). Thus, the standard 'whole image' contrastive learning approach is inadequate for this task.

We propose an approach that extends existing contrastive learning methods to sequential prediction tasks such as text recognition. The key idea is to apply contrastive learning to the individual elements of the sequence, while maintaining information about their order. To do so, we introduce an instance-mapping function that yields an instance from every few consecutive frames in a sequence feature map. The instance is the atomic element that will be used in contrastive learning. A given image, depending on its width, may produce an arbitrary number of instances. This enlarges the number of negative examples in every batch without requiring a memory bank [27] or architecture modifications [2]. Individual instances are part of a sequence, thus we design an augmentation procedure that ensures a sequence-level alignment, which is crucial for yielding effective representations (Fig. 2).

We validate our method experimentally, comparing its performance with non-sequential contrastive approaches on several handwritten and scene text datasets. To evaluate the quality of the learned visual representation, we lay out a decoder evaluation protocol that extends the widely-used linear evaluation criteria [67, 37] for encoder-decoder based networks. Utilizing this evaluation, we demonstrate significant improvements over current contrastive learning approaches. Furthermore, we find that our method outperforms supervised training methods with limited amounts of labeled training data, and it achieves state-of-the-art results on standard handwritten datasets, reducing the word error rate by 9.5% on IAM and by 20.8% on RIMES.

To summarize, the key contributions of our work are:

- A contrastive learning approach for visual sequence-to-sequence recognition.
- Viewing each feature map as a sequence of individual instances, leading to contrastive learning in a sub-word level, such that each image yields several positive pairs and multiple negative examples.
- Defining sequence preserving augmentation procedures, and custom projection heads.
- Extensive experimental validation showing state-of-the-art performance on handwritten text.

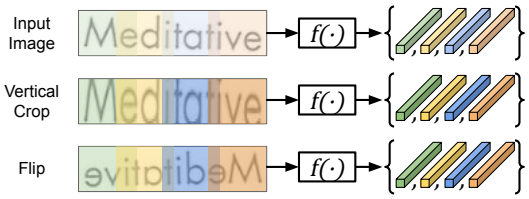

Figure 2: **Sequence preserving augmentations.** We propose an augmentation procedure which meets the sequential structure of the feature map. For example, as opposed to vertical cropping, horizontal flipping results in a sequence-level misalignment which leads to poor positive pairing.

2. Related Work

Visual representation learning Unsupervised representation learning has recently achieved success, not only in natural language processing [48, 44, 16, 50, 51] and speech recognition [4, 32, 63], but also in computer vision. The first methods suggested learning visual representations by training the network on an artificially designed pretext task, such as denoising auto-encoders [61], patch ordering [17], colorizing an image [67], and others [46, 20].

In this paper, we focus on the contrastive learning approach, which has recently shown promising results on several tasks [30, 29, 2, 9, 18, 27, 10, 8, 52]. In this method, we maximize agreement between representations of differently augmented views of the same data and contrast between representations coming from different images [2]. This process may be viewed as a classification task where each image is assumed to be its own class.

Several papers explored this approach, introducing several advances over the base contrastive scheme. The authors in [9] proposed an augmentation pipeline and an additional projection head which maps the representations into space where the contrastive loss is applied. In [27] a momentum-based contrastive scheme was suggested, and [10] included a teacher-student distillation phase. Additional papers [26, 52, 60, 49] introduced contrastive learning schemes for action classification of sequential inputs and non-sequential outputs. Motivated by these papers, we expand the contrastive learning framework to visual sequence-to-sequence predictions as in text recognition.

Un- and semi-supervised learning for text recognition Despite clear advantages, currently, most text recognition methods do not utilize unlabeled real-world text images. Specifically, handwritten recognition usually relies on fully-supervised training [64, 59], while scene text models are trained mostly on synthetic data [3, 38]. That said, [68] and [34] have recently suggested domain adaptation techniques to utilize an unlabeled dataset along with labeled data. Using adversarial training, these methods align the

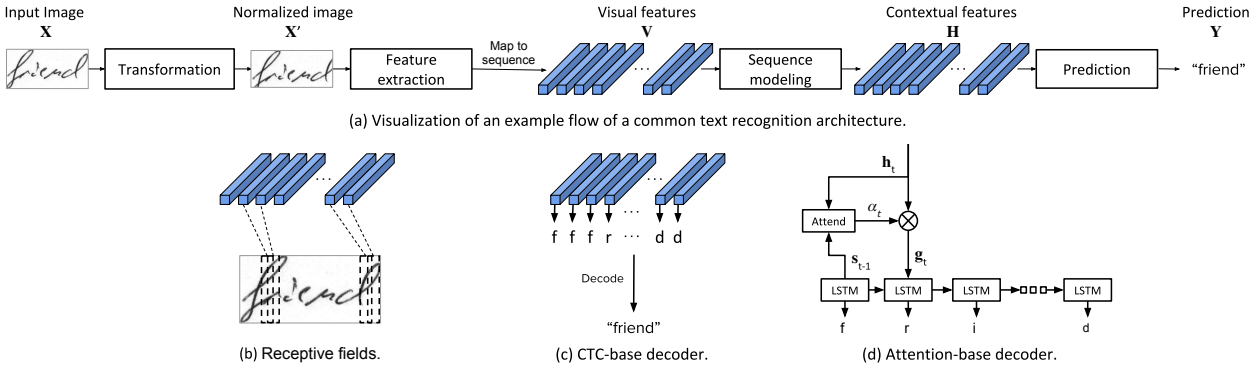

Figure 3: **Typical text recognition architecture**. Visualization of an example flow of a text recognition architecture (a), as described in Section 3. After the input text image is normalized, we extract a sequence of visual features from it, where each frame is associated with a different receptive field of the input image (b). Then, these visual representations go through a sequence modeling (LSTM scheme), and finally are decoded using a CTC (c) or an attention (d) decoder.

feature map distributions of both datasets.

For images of printed text, [25] recently proposed a completely unsupervised scheme in which a discriminator enforces the predictions to align with a distribution of a given text corpus. Nevertheless, this method requires restricting the recognizer architecture to use only local predictions.

To the best of our knowledge, this work is the first to propose self-supervised representation learning for text recognition. Our method further leads to state-of-the-art results on handwritten text.

3. Text Recognition Background

Several architectures have been proposed over the years for recognition of scene text [40, 12, 53] and handwritten text [58, 43]. Throughout this work, we focus on a general text recognition framework, which was proposed by [3]. This framework describes the building blocks of many text recognizers, including [54, 57, 55, 56, 39, 62, 13, 14, 7, 68, 38, 19, 64]. As shown in Fig. 3, this architecture consists of the following four stages (see more details in Appendix A):

1. **Transformation:** A normalization of the input text image using a Thin Plate Spline (TPS) transformation [56, 39], which is a variant of the spatial transformer network [31]. This stage is optional yet important for images of text in diverse shapes.

2. **Feature extraction:** A convolutional neural network (CNN) that extracts features from the normalized image, followed by a map-to-sequence operation that reshapes the features into a sequence of frames, denoted by $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_T]$. As illustrated in Fig. 3(b), the resulting frames correspond to different receptive fields in the image. Note that the sequence length depends on the width of the input image.

3. **Sequence modeling:** An optional Bidirectional LSTM (BiLSTM) scheme which aims to capture the contextual information within the visual feature sequence. This network yields the contextual features $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T]$, which in turn, are concatenated to the feature map $\mathbf{V}$, as suggested in [38].

4. **Prediction:** A text decoder using (i) a connectionist temporal classification (CTC) decoder [21] that decodes separately each frame, and then deletes repeated characters and blanks (Fig. 3(c)); or (ii) an attention decoder [14, 56], which linearly combines the frames to feed them into a one-layer LSTM (Fig. 3(d)).

4. Sequence-to-Sequence Contrastive Learning

Inspired by self-supervised methods for visual representation learning [9, 18, 27, 10, 8], we propose a contrastive learning framework for sequence-to-sequence visual recognition. To do so, we introduce a novel instance-mapping stage that yields a separate instance from every few consecutive frames in the sequential feature map. These instances then serve as the atomic elements in the contrastive loss. In addition, we design an augmentation procedure that maintains the sequential structure (Fig. 2) which, as demonstrated in Section 5, is crucial for yielding effective representations.

As depicted in Fig. 4, we suggest a framework consisting of the following five building-blocks:

1. A *stochastic data augmentation* module that is designed to ensure a sequence-level alignment. This operation transforms any given image $\mathbf{X}_i$ in a batch of N images, into two augmented images $\mathbf{X}_i^a, \mathbf{X}_i^b \in \mathbb{R}^{C \times H \times W_i}$, where C denotes the number of input channels, H the image height, and W_i the width of each image which may vary.

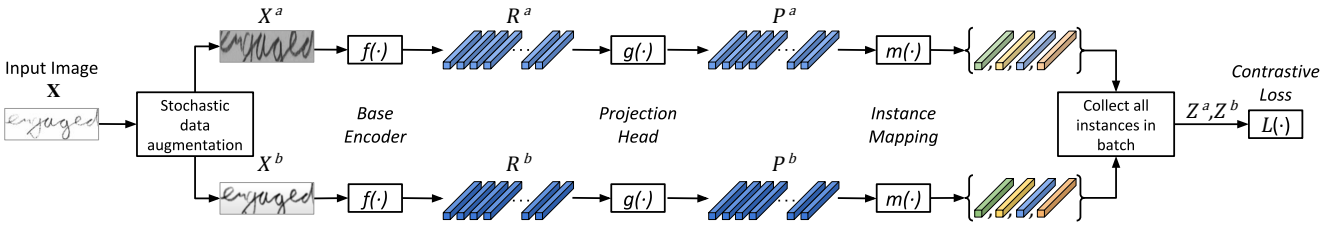

Figure 4: **SeqCLR block diagram.** Each image in a batch is augmented twice, and then fed separately into a base encoder and projection head, to create pairs of representation maps. Next, to account for the sequential structure of these representations, we apply an instance-mapping function (see also Fig. 5) that transforms them into several instances and thus allows us to apply contrastive learning at a sub-word level.

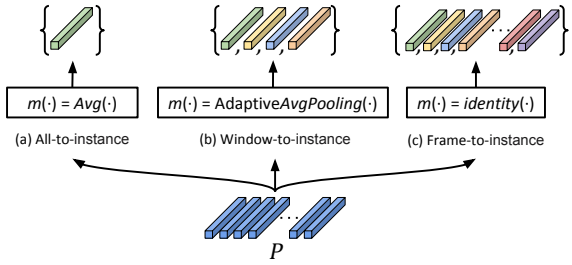

Figure 5: **Instance-mapping function.** This function yields separate instances for the contrastive loss out of each (possibly projected) feature map. The all-to-instance mapping (a) averages all the frames and thus improves robustness to sequence-level misalignment. On the other hand, the frame-to-instance alternative (c) maps each frame to a separate instance, which enlarges the number of negative examples. The window-to-instance mapping (b) represents a trade-off between these options.

2. A *base encoder* $f(\cdot)$ consisting of several blocks of the recognizer scheme (Fig. 3(a)). For each pair of augmented images, this component extracts a pair of sequential representations, $\mathbf{R}_i^a, \mathbf{R}_i^b \in \mathbb{R}^{F \times T_i}$, where F is the feature dimension, and T_i is the number of frames (columns) which is dependent on the image width (Fig. 3(b)).

3. An optional *projection head* $g(\cdot)$, that transforms the representations using a small auxiliary network, as in [9]. We suggest new projection head types that can handle varying sequence sizes, and denote this stage output by $\mathbf{P}_i^a, \mathbf{P}_i^b \in \mathbb{R}^{F' \times T_i}$, where F' is the feature dimension after the projection.

4. A novel *instance-mapping function* $m(\cdot)$ is utilized before the contrastive loss to yield T_i' instances out of T_i projected frames, as illustrated in Fig. 5. These instances are then used as the atomic elements in the contrastive loss. Next, we collect all the instances in the batch into two aligned sets $\mathcal{Z}^a, \mathcal{Z}^b$, each of size $\sum_{i=1}^{N} T_i'$, such that corresponding indices refer to corresponding frames of the same input image.

5. A *contrastive loss* function as in [9, 27, 18], that aims to pull closer together representations of corresponding indices of $\mathcal{Z}^a, \mathcal{Z}^b$, i.e. positive pairs, and to push all the others, i.e. negative examples, farther apart:

$$\mathcal{L}(\mathcal{Z}^a, \mathcal{Z}^b) = \sum_{r \in |\mathcal{Z}^a|} \ell_{\text{NCE}}\left(\mathbf{z}_r^a, \mathbf{z}_r^b;\, \mathcal{Z}^a \cup \mathcal{Z}^b\right) + \sum_{r \in |\mathcal{Z}^b|} \ell_{\text{NCE}}\left(\mathbf{z}_r^b, \mathbf{z}_r^a;\, \mathcal{Z}^a \cup \mathcal{Z}^b\right), \quad (1)$$

where $\ell_{\text{NCE}}(\cdot)$ is the noise contrastive estimation (NCE) loss function [47]:

$$\ell_{\text{NCE}}(\mathbf{u}^a, \mathbf{u}^b; \mathcal{U}) = -\log \frac{\exp\left(\text{sim}(\mathbf{u}^a, \mathbf{u}^b)/\tau\right)}{\sum_{\mathbf{u} \in \mathcal{U} \setminus \mathbf{u}^a} \exp\left(\text{sim}(\mathbf{u}^a, \mathbf{u})/\tau\right)}.$$

As in [9], for the similarity operator we use the cosine distance, $\text{sim}(\mathbf{v}, \mathbf{u}) = \mathbf{v}^T\mathbf{u} / \|\mathbf{v}\| \, \|\mathbf{u}\|$.

We now detail each of these components and describe their configurations.

Data augmentation As pointed out in previous papers [10, 11, 2], the augmentation pipeline plays a key part in the final quality of the learned visual representations. Current stochastic augmentation schemes [9] are mostly based on aggressive cropping, flipping, color distortions, and blurring. These schemes cannot properly serve the task of text recognition, as they often render the text in the image unreadable. For example, we should refrain from aggressive horizontal cropping as this might cut out complete characters.

In addition, these augmentation compositions were tailored for tasks as object recognition or classification, where images are atomic input elements in the contrastive loss. However, since in our framework individual instances are part of a sequence, we design an augmentation procedure that ensures sequence-level alignment. Therefore, we avoid transformations such as flipping, aggressive rotations, and substantial horizontal translations. Figure 6 depicts different augmentation types considered in this work, including vertical cropping, blurring, random noise, and different perspective transformations.

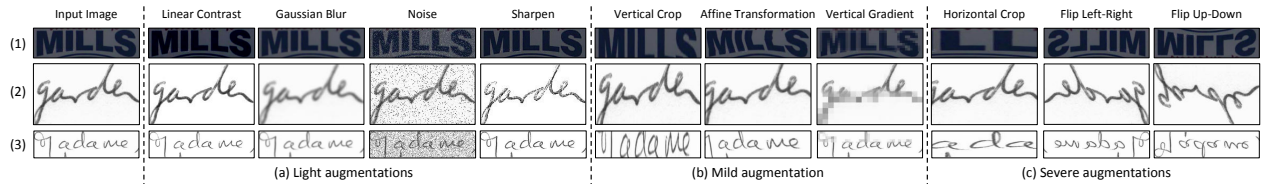

Figure 6: **Considered augmentations.** Examples of different augmentation types considered in this work, illustrated on three datasets: (1)-IIT5k, (2)-IAM and, (3)-RIMES. As discussed in Section 4, while flipping and aggressive horizontal cropping are fundamental augmentations in learning visual representations for classification, they should be avoided for text recognition training as they cause sequence-level misalignment which leads to poor contrastive learning.

Base encoder The encoder extracts sequential representations from the augmented images $\mathbf{X}_i^a, \mathbf{X}_i^b$. While in most contrastive learning schemes the identity of the representation layer is pretty clear – usually the visual backbone output [9, 18], in text recognition there are different options. In particular, we consider two candidates as the sequential representations $\mathbf{R}_i \in \mathbb{R}^{F \times T_i}$, where each option defines $f(\cdot)$ as the text recognizer scheme (Fig. 3) up to this stage:

1. The visual features, $\mathbf{R}_i = \mathbf{V}_i$.
2. The contextual feature map, $\mathbf{R}_i = \mathbf{H}_i$, which better captures contextual information within the sequence.

Projection head The representations are optionally transformed by a projection head, $\mathbf{P}_i = g(\mathbf{R}_i)$, which is a small auxiliary neural network that is discarded entirely after the pre-training stage. As indicated in [9, 10], this mapping improves the quality of the learned representations.

Currently, a commonly used projection head is the multilayer perceptron (MLP) [9, 10]; however, it can only accommodate fixed-size inputs and thus cannot serve text images. Therefore, we propose two new projection heads in light of the instance-mapping functions defined below: an MLP projection head that operates on each frame independently as the frame-to-instance mapping (Fig. 5(c)), and a BiLSTM projection head for improving contextual information in the others mappings (Fig. 5(a,b)).

Instance-mapping Previous work [9, 18, 27, 2] considered images as atomic input elements in the contrastive loss. Therefore, each projected map was vectorized to a single instance, $\mathbf{z}_i = \text{flatten}(\mathbf{P}_i)$. However, the inputs and the feature maps in text recognition are of varying sizes and thus cannot be handled by the flatten operator. More importantly, in text recognition the feature maps have a sequential structure and thus do not represent a single class. Therefore, we propose to view every few consecutive frames in the feature map as an atomic input element for the contrastive loss.

We propose two approaches for creating individual instances out of sequential feature maps of varying sizes. In the first approach, we transform every *fixed number of frames* into separate instances, for example, by averaging each W consecutive frames. In the second approach, we *fix the number of instances* created out of each image, for example, by using adaptive average pooling.

In particular, as depicted in Fig. 5, we consider three instance-mapping functions as specifications of these approaches, which extract T_i' instances out of T_i given frames:

1. **All-to-instance**: All the frames in a sequential feature map are averaged to a single instance, $m(\mathbf{P}) = \text{Avg}(\mathbf{P})$, resulting in sets $\mathcal{Z}^a, \mathcal{Z}^b$ of N instances each.
2. **Window-to-instance**: Create an instance out of every few consecutive frames. We choose to fix the number of instances and use adaptive average pooling to obtain T' instances. Thus, this operation results in sets $\mathcal{Z}^a, \mathcal{Z}^b$ of size $N \cdot T'$ each.
3. **Frame-to-instance**: Each frame is considered as a separate instance, $T_i' = T_i$, resulting in sets of size $\sum_{i=1}^{N} T_i$, which depend on the input sizes.

Averaging over frames compensates for sequence-level misalignment, which is especially needed for dealing with text written in arbitrary shapes as in scene text images (see Section 5.1 below). On the other hand, this operation reduces the number of negative examples in each batch, which, as demonstrated in [9, 27], can deteriorate the quality of the learned representation. In this vein, the window-to-instance mapping represents the trade-off between misalignment robustness and sample efficiency. Note, however, that there are other components in our framework that can also handle this misalignment, such as the BiLSTM projection head and the sequence modeling in the base encoder.

5. Experiments

In this section, we experimentally examine our method, comparing its performance with the non-sequential *SimCLR* method [9] on several handwritten and scene text datasets. For this goal, we first consider a decoder evaluation protocol, which is an analog to the linear evaluation procedure ([67, 37]) for encoder-decoder based networks. Then, we test our models in semi-supervised settings in which we

Method	Decoder	Handwritten Dataset						Scene-Text Dataset					
		IAM		RIMES		CVL		IIT5K		IC03		IC13	
		Acc	ED1	Acc	ED1	Acc	ED1	Acc	ED1	Acc	ED1	Acc	ED1
SimCLR [9]	CTC	4.0	16.0	10.0	20.3	1.8	11.1	0.3	3.1	0.0	1.0	0.3	5.0
SimCLR Contextual		6.0	17.2	13.4	25.8	7.1	17.8	1.1	4.0	2.0	2.9	1.5	6.3
SeqCLR All-to-instance		34.4	60.9	59.0	80.0	55.2	73.4	20.4	42.8	24.2	49.7	24.4	51.3
SeqCLR Frame-to-instance		29.4	53.1	57.5	77.5	64.3	76.0	3.0	11.4	4.6	12.3	4.9	17.5
SeqCLR Window-to-instance		**39.7**	**63.3**	**63.8**	**81.8**	**66.7**	**77.0**	**35.7**	**62.0**	**43.6**	**71.2**	**43.5**	**67.9**
SimCLR [9]	Attention	16.0	21.2	22.0	28.3	26.7	30.6	2.4	3.6	3.7	4.3	3.1	4.9
SimCLR Contextual		17.8	23.1	34.3	40.6	34.3	38.0	3.6	5.0	4.4	5.4	3.9	6.7
SeqCLR All-to-instance		51.6	**65.0**	77.9	85.8	73.1	75.3	37.3	51.8	48.0	62.2	45.8	60.4
SeqCLR Frame-to-instance		46.6	56.6	76.6	84.5	73.5	75.9	15.3	23.7	20.8	28.7	21.4	30.6
SeqCLR Window-to-instance		**51.9**	63.6	**79.5**	**86.7**	**74.5**	**77.1**	**49.2**	**68.6**	**63.9**	**79.6**	**59.3**	**77.1**

Table 1: **Representation quality.** Accuracy (Acc) and single edit distance (ED1) of the decoder evaluation – an analog of the linear evaluation for encoder-decoder networks, in which we train a decoder with labeled data on top of a frozen encoder that was pre-trained on unlabeled images. We compare our SeqCLR method of different instance-mapping functions (Fig. 5) with the non-sequential method SimCLR [9]. Averaging frames in the feature map, as in all-to-instance and window-to-instance mappings, is especially important in scene-text recognition. Table 2 below presents semi-supervised performance, while Table 3 shows state-of-the-art results in handwritten datasets.

fine-tune a pre-trained model with limited amounts of labeled training data. Finally, we find that when fine-tuned on the entire labeled data, our method achieves state-of-the-art results on standard handwritten datasets.

Datasets We conduct our experiments on several public datasets of handwritten and scene text recognition. For handwriting we consider the English datasets IAM [42] and CVL [36], and the French dataset RIMES [23]. For scene text, we train on the synthetic dataset SyntText [24], and test on three real world datasets: IIT5K [45], IC03 [41] and IC13 [35]. We present samples from each dataset and include more details on the datasets in Appendix C.

Metrics To evaluate performance, we adopt the metrics of word-level accuracy (Acc) and word-level accuracy up to a single edit distance (ED1). For the state-of-the-art comparison in handwriting in Table 3, we employ the Character Error Rate (CER) and the Word Error Rate (WER) [59, 68].

Contrastive learning configurations While in Section 6 we study the effect of modifying each component in our framework, in this section we limit ourselves to the best configuration found for each instance-mapping function (Fig. 5): all-to-instance, frame-to-instance and window-to-instance with $T' = 5$. In all of these schemes, the augmentation pipeline consists of linear contrasting, blurring, sharpening, horizontal cropping and light affine transformations, as further detailed in Appendix B, including examples and pseudo-code. The base encoder contains a sequential modeling, i.e. $\mathbf{R} = \mathbf{H}$. Since in such a base encoder the projection head might be redundant (see Section 6), we maximize over having and discarding a projection head. To compare our method to non-sequential contrastive learning, we re-implement the *SimCLR* scheme [9] where the visual features are the representation layer ($\mathbf{R} = \mathbf{V}$). For a fair comparison, we consider also *SimCLR Contextual* where the representation layer is the contextual features ($\mathbf{R} = \mathbf{H}$).

Additional implementation details are described in Appendix D, including the recognizer settings and the procedures for pre-training, decoder evaluation and fine-tuning.

5.1. Decoder evaluation

We start our experimental study by evaluating the quality of the learned visual representation. To this end, we establish a decoder evaluation protocol that extends the widely-used linear evaluation protocol [67, 37] to encoder-decoder based networks. In this protocol, we first train a base-encoder $f(\cdot)$ on the unlabeled data, using some self-supervised method. Then, we freeze the encoder weights and train on top of it a CTC or an attention decoder (Fig. 3(c,d)) with all the labeled data. Since we keep the encoder untouched, this test can be seen as a proxy to the representation learning efficiency.

Table 1 shows the results of our proposed SeqCLR method, compared with vanilla *SimCLR* [9] and *SimCLR Contextual*, over public datasets of handwritten and scene text benchmarks, with either a CTC or an attention decoder (Fig. 3(c,d)). As discussed above, current contrastive methods for visual representations are designed for tasks such as classification and object detection, where images are atomic input elements. However, in text recognition, a word is viewed as a sequence of characters, and therefore, the standard 'whole image' concept leads to poor performance. Specifically, the augmentation procedure considered in [9, 27] usually breaks the sequential structure of the input text image. In addition, in these prior papers, the feature map is treated as a single representation, whereas in text recognition, it is eventually decoded as a sequence of representations.

The comparison between the different instance-mapping

Method	Decoder	Handwritten Dataset									Scene-Text Dataset		
		IAM			RIMES			CVL			IIT5K	IC03	IC13
		Label fraction									Label fraction		
		5%	10%	100%	5%	10%	100%	5%	10%	100%	100%	100%	100%
Supervised Baseline	CTC	21.4	33.6	75.2	35.9	59.7	86.9	48.7	63.6	75.6	76.1	87.9	84.3
SimCLR [9]		15.4	21.8	65.0	36.5	52.9	84.5	52.1	62.0	74.1	69.1	83.4	79.4
SimCLR Contextual		20.4	27.8	63.7	48.6	55.6	84.4	51.8	62.3	74.1	64.5	81.7	78.1
SeqCLR All-to-instance		27.5	44.8	**76.7**	50.4	66.4	89.1	60.1	69.4	76.9	74.7	88.2	83.2
SeqCLR Frame-to-instance		**31.2**	**44.9**	75.1	**61.8**	**71.9**	**90.1**	**66.0**	**71.0**	**77.0**	69.8	84.2	81.8
SeqCLR Window-to-instance		26.2	42.1	**76.7**	56.6	62.5	89.6	61.2	69.7	76.9	**80.9**	**89.8**	**86.3**
Supervised Baseline	Attention	25.7	42.5	77.8	57.0	67.7	89.3	64.0	72.1	77.2	**83.8**	91.1	**88.1**
SimCLR [9]		22.7	32.2	70.7	49.9	60.9	87.8	59.0	65.6	75.7	77.8	88.8	84.9
SimCLR Contextual		24.6	32.9	70.2	51.9	63.0	87.3	59.7	66.2	75.2	72.2	87.0	82.3
SeqCLR All-to-instance		**40.3**	51.6	79.8	69.7	76.9	**92.5**	69.5	73.2	77.6	80.9	90.0	87.0
SeqCLR Frame-to-instance		37.2	48.5	78.2	68.8	75.9	92.3	69.7	73.4	77.5	76.3	90.2	85.8
SeqCLR Window-to-instance		38.1	**52.3**	**79.9**	**70.9**	**77.0**	92.4	**73.1**	**74.8**	**77.8**	82.9	**92.2**	87.9

Table 2: **Semi-supervised results.** Accuracy of fine-tuning a pre-trained model with 5%, 10% and 100% of the labeled data. For scene-text datasets we test only for 100% as the data is anyhow synthetic. As presented in Table 3, our method achieves state-of-the-art results on handwritten datasets.

Dataset	Method	WER	CER	Average
IAM	Bluche et al. [5]	24.7	**7.3**	16.00
	Bluche et al. [6]	24.6	7.9	16.25
	Sueiras et al. [59]	23.8	8.8	16.30
	ScrabbleGAN [19]	23.6	-	-
	SSDAN* [68]	22.2	8.5	15.35
	SeqCLR	**20.1**	9.5	**14.80**
RIMES	Alonso et al. [1]	11.9	4.0	7.95
	ScrabbleGAN [19]	11.3	-	-
	Chowdhury et al. [15]	9.6	3.4	6.55
	SeqCLR	**7.6**	**2.6**	**5.5**

Table 3: **SOTA error rates.** Word and character error rates (WER and CER) of our method compared to current state-of-the-art word-level methods on IAM and RIMES datasets. '*' indicates using the unlabeled test set for training.

functions demonstrates that the best results are achieved by the window-to-instance mapping (Fig. 5(b)). As can be seen, the frame-to-instance mapping, which does not average consecutive frames, performs poorly on scene text images. These images are prone to sequence-level misalignment by even mild augmentations, as they contain text that already comes in diverse shapes. On the other hand, the all-to-instance mapping, which averages all the frames, significantly reduces the number of negative examples in each batch, which in turn, also affects performance. The window-to-instance mapping succeeds in balancing these concerns and therefore leads to better performance.

5.2. Fine-tuning

We further evaluate our method by considering semi-supervised settings. We use the same encoders as before, which were pre-trained on the unlabeled data, but now let the whole network be fine-tuned using 5% or 10% of the labeled dataset. Contrary to prior work [66, 9], which consider class-balanced datasets, we simply use the same randomly selected data for all the experiments. We also test for fine-tuning on the entire labeled data, as suggested in [9]. Note that this is the only evaluation we examine for scene text recognition, as the training dataset is anyhow synthetic in this case.

As opposed to the decoder evaluation, here, the goal is to achieve the best results and not just qualify the learned representations. Therefore, contrary to the decoder evaluation, in the fine-tuning phase, one can attach additional layers besides the decoder on top of the encoder. That said, we only attach a text decoder (CTC or attention), as the base encoder in the following experiments already contains a sequence modeling.

Table 2 compares our method with *SimCLR* [9], *SimCLR Contextual* and *supervised baseline* training. As can be seen, in the case of text recognition, pre-training using non-sequential contrastive learning schemes often leads to deterioration in performance compared to the *supervised baseline*. SeqCLR, on the other hand, achieves better performance for every semi-supervised scenario and on every handwritten dataset. In particular, the window-to-instance mapping performs the best for the attention decoder, while the frame-to-instance alternative is superior when using the CTC decoder. This is an interesting result that might indicate that frame-to-instance better fits the CTC decoder as they both operate on individual frames of the feature map.

In the case of fine-tuning on 100% of the labeled data, although our method does not use any additional data, it still succeeds in significantly improving the results of the fully *supervised baseline* training on handwritten datasets. In

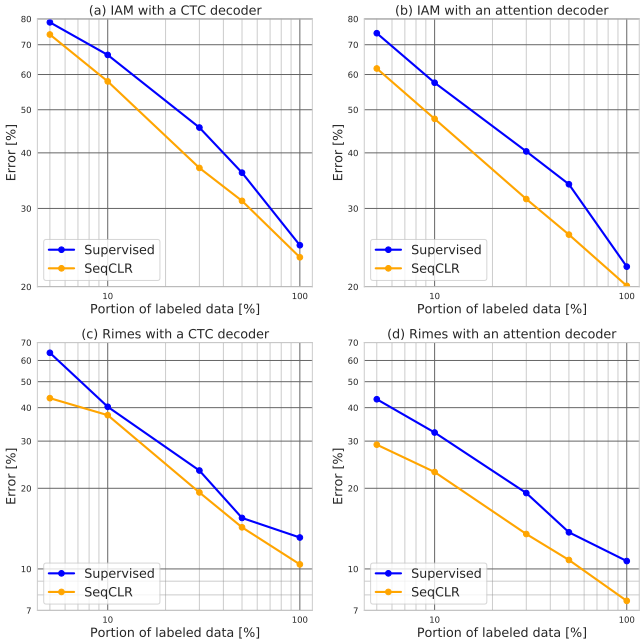

Figure 7: **Word error rate as a function of labeled data amount in a log-log scale.** Roughly speaking, SeqCLR unsupervised pre-training has the same effect as doubling the labeled data amount in the sense of reducing the error rate.

particular, our method gains an improvement of +1.9% on average for the CTC decoder and +1.7% on average for the attention decoder. In scene text datasets, SeqCLR achieves an improvement of 2.9% on average for the CTC decoder; however, it performs the same on average as the *supervised baseline* for the attention decoder. The mixed performance in scene text might be a result of utilizing only synthetic data in the representation learning phase.

Notably, as presented in Table 3, SeqCLR using window-to-instance mapping outperforms the current word-level state-of-the-art performance on both IAM and RIMES datasets, even when compared to methods which used the test-set in their unsupervised training. Note that for a fair comparison, we only include results that considered the same test-set and that did not attach a language model.

In Fig. 7, we present the word error rate of SeqCLR and the *supervised baseline* as a function of the portion of the labeled data for IAM and RIMES datasets using CTC and attention decoders. As can be seen, in the RIMES dataset using an attention decoder, SeqCLR utilizing 50% of the labeled data achieves the same error rate as the current state-of-the-art algorithm trained on the entire labeled data.

6. Ablation Study

After studying the effect of the instance-mapping functions in Section 5.1, in this section, our goal is two-fold. First, to match the components in our framework, and second, to demonstrate the importance of maintaining sequence-level alignment by applying different augmentation procedures.

For evaluating the representation learning in this section, we adopt the decoder evaluation protocol (Section 5.1) on the RIMES dataset, considering representation layers of visual features ($\mathbf{R} = \mathbf{V}$) and contextual features ($\mathbf{R} = \mathbf{H}$).

Projection head	Mapping $m(\cdot)$	Visual feat.		Contextual feat.	
		CTC	Attn	CTC	Attn
None	All-to-instance	2.0	55.8	55.4	**77.9**
MLP per frame		29.9	70.3	50.4	72.5
BiLSTM		**35.5**	**70.9**	**59.0**	74.8
None	Frame-to-instance	27.4	56.9	49.9	75.8
MLP per frame		**39.9**	69.8	**57.5**	**76.6**
BiLSTM		37.4	**69.9**	43.5	64.3
None	Window-to-instance	27.9	67.6	59.9	**79.5**
MLP per frame		29.9	70.3	50.4	72.5
BiLSTM		**35.8**	**74.0**	**63.8**	75.9

Table 4: **Matching projection heads to mappings**. Representation qualities (decoder evaluation accuracy) of combining different projection heads with instance-mapping functions (Fig. 5). While BiLSTM head fits the all-to-instance and window-to-instance mappings, the MLP per frame performs better with the frame-to-instance mapping.

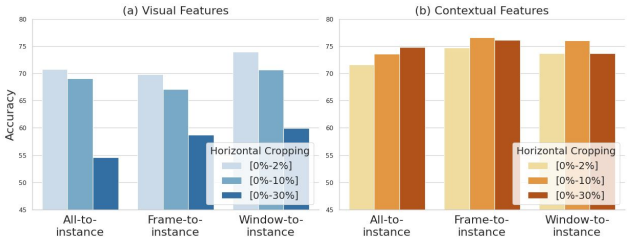

Figure 8: **Horizontal cropping.** The effect of horizontal cropping on the representation learning. This augmentation affects learning when the representation layer is chosen to be the visual features (a); however, a light version of it can help in encoders that contain sequence modeling (b).

Matching the components As can be seen in Table 4, including a sequence modeling in the base encoder ($\mathbf{R} = \mathbf{H}$) leads to significant improvements in the quality of the learned representation for both CTC and attention decoders. In general, incorporating a projection head also improves representation effectiveness; however, when utilizing a sequence modeling and an attention decoder, each containing BiLSTM layers, then the BiLSTM projection head appears as a redundant component. The key message from these experiments is that the SeqCLR components should be selected dependently.

Sequence-level alignment In text recognition, individual instances are part of a sequence, and thus, the augmentation procedure needs to maintain a sequence-level alignment (Fig. 2). On the other hand, as suggested in [9, 11], strong data augmentations contribute to contrastive learning. The following experiments aim to study this trade-off and to identify components in our framework that improve the robustness to sequence-level misalignment.

Figure 8 demonstrates the effect of horizontal cropping on the representation learning. As observed, having a base encoder with sequence modeling is crucial for handling sequence-level misalignment caused by even mild horizontal cropping. Note that such cropping might cut out complete characters from the input text images. This indicates that during the representation learning, the sequence modeling successfully captures contextual information within the visual features, which compensates for missing visual information and sequence-level misalignment.

7. Discussion and conclusions

We presented SeqCLR, a contrastive learning algorithm for self-supervised learning of sequence-to-sequence visual recognition that divides each feature map into a sequence of individual elements for the contrastive loss. In order to take full-advantage of self-supervision, we proposed a number of sequence-specific augmentation techniques that differ from whole-image equivalents.

The main take-home lesson is that paying attention to the task's structure, i.e. treating an image as a sequence of frames, pays off. Our experiments show that SeqCLR largely outperforms current non-sequential contrastive learning methods in recognizing handwritten and scene text images when the amount of supervised training is limited. Furthermore, our method achieves state-of-the-art performance on handwriting – compared with the best methods in the literature SeqCLR reduces the word error rate by 9.5% and 20.8% on IAM and RIMES, the standard benchmark datasets. SeqCLR is the result of careful experimental evaluation of different design options, including different augmentation compositions, encoder architectures, projection heads, instance-mapping functions, and decoder types.

The success of SeqCLR will hopefully encourage other researchers to explore semi-supervised and self-supervised schemes for text recognition, as well as contrastive learning algorithms for different sequence-to-sequence predictions.

References

[1] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial generation of handwritten text images conditioned on sequences. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 481–486. IEEE, 2019. 7

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15535–15545, 2019. 2, 4, 5

[3] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4715–4723, 2019. 2, 3, 12, 14

[4] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*, 2020. 2

[5] Théodore Bluche. *Deep neural networks for large vocabulary handwritten text recognition*. PhD thesis, Paris 11, 2015. 7

[6] Théodore Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *Advances in Neural Information Processing Systems*, pages 838–846, 2016. 7

[7] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79. ACM, 2018. 3

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 1, 2, 3

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1, 2, 3, 4, 5, 6, 7, 9, 13, 14

[10] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 1, 2, 3, 4, 5

[11] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 4, 9

[12] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *arXiv preprint arXiv:2005.03492*, 2020. 3

[13] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5076–5084, 2017. 3, 12

[14] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of*

the IEEE international conference on computer vision, pages 5076–5084, 2017. 3, 14

[15] Arindam Chowdhury and Lovekesh Vig. An efficient end-to-end neural model for handwritten text recognition. *arXiv preprint arXiv:1807.07965*, 2018. 7

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[17] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 2

[18] William Falcon and Kyunghyun Cho. A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*, 2020. 1, 2, 3, 4, 5

[19] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. Scrabblegan: Semi-supervised varying length handwritten text generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4324–4333, 2020. 3, 7

[20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 2

[21] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006. 3

[22] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. 12

[23] Emmanuèle Grosicki and Haikal El Abed. Icdar 2009 handwriting recognition competition. In *2009 10th International Conference on Document Analysis and Recognition*, pages 1398–1402. IEEE, 2009. 6, 13

[24] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2315–2324, 2016. 6, 13

[25] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Learning to read by spelling: Towards unsupervised text recognition. In *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, pages 1–10, 2018. 1, 3

[26] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *Neurips*, 2020. 1, 2

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 2, 3, 4, 5, 6

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 14

[29] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 2

[30] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018. 2

[31] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 3, 12

[32] Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, Wei Zou, and Xiangang Li. Improving transformer-based speech recognition using unsupervised pre-training. *arXiv preprint arXiv:1910.09932*, 2019. 2

[33] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020. 12, 13

[34] Lei Kang, Marçal Rusiñol, Alicia Fornés, Pau Riba, and Mauricio Villegas. Unsupervised adaptation for synthetic-to-real handwritten word recognition. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3491–3500. IEEE, 2020. 1, 2

[35] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE, 2013. 6, 13

[36] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In *2013 12th international conference on document analysis and recognition*, pages 560–564. IEEE, 2013. 6, 13

[37] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1920–1929, 2019. 2, 5, 6

[38] Ron Litman, Oron Anschel, Shahar Tsiper, Roee Litman, Shai Mazor, and R Manmatha. Scatter: selective context attentional scene text recognizer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11962–11972, 2020. 2, 3, 12, 14

[39] Wei Liu, Chaofeng Chen, Kwan-Yee K Wong, Zhizhong Su, and Junyu Han. Star-net: A spatial attention residue network for scene text recognition. In *BMVC*, volume 2, page 7, 2016. 3

[40] Shangbang Long, Xin He, and Cong Ya. Scene text detection and recognition: The deep learning era. *arXiv preprint arXiv:1811.04256*, 2018. 3

[41] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. Icdar 2003 robust reading competitions. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 682–687. Citeseer, 2003. 6, 13

[42] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002. 6, 13

[43] Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. Handwritten optical character recognition (ocr): a comprehensive systematic literature review (slr). *IEEE Access*, 8:142642–142668, 2020. 3

[44] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 2

[45] Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. 2012. 6, 13

[46] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 2

[47] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4

[48] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 2

[49] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *arXiv preprint arXiv:2008.03800*, 2020. 1, 2

[50] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2

[51] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2

[52] Haocong Rao, Shihao Xu, Xiping Hu, Jun Cheng, and Bin Hu. Augmented skeleton based contrastive action learning with momentum lstm for unsupervised action recognition. *arXiv preprint arXiv:2008.00188*, 2020. 1, 2

[53] Payel Sengupta and Ayatullah Faruk Mollah. Journey of scene text components recognition: Progress and open issues. *Multimedia Tools and Applications*, pages 1–26, 2020. 3

[54] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016. 3, 12

[55] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016. 3

[56] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4168–4176, 2016. 3, 12

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[58] Manoj Sonkusare and Narendra Sahu. A survey on handwritten character recognition (hcr) techniques for english alphabets. *Advances in Vision Computing: An International Journal (AVC)*, 3(1), 2016. 3

[59] Jorge Sueiras, Victoria Ruiz, Angel Sanchez, and Jose F Velez. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing*, 289:119–128, 2018. 2, 6, 7

[60] Michael Tschannen, Josip Djolonga, Marvin Ritter, Aravindh Mahendran, Neil Houlsby, Sylvain Gelly, and Mario Lucic. Self-supervised learning of video-induced visual invariances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13806–13815, 2020. 1, 2

[61] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. 2

[62] Jianfeng Wang and Xiaolin Hu. Gated recurrent convolution neural network for ocr. In *Advances in Neural Information Processing Systems*, pages 335–344, 2017. 3

[63] Weiran Wang, Qingming Tang, and Karen Livescu. Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6889–6893. IEEE, 2020. 2

[64] Mohamed Yousef and Tom E Bishop. Origaminet: Weakly-supervised, segmentation-free, one-step, full page text recognition by learning to unfold. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14710–14719, 2020. 2, 3

[65] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 14

[66] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485, 2019. 7

[67] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 2, 5, 6

[68] Yaping Zhang, Shuai Nie, Wenju Liu, Xing Xu, Dongxiang Zhang, and Heng Tao Shen. Sequence-to-sequence domain adaptation network for robust text image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2740–2749, 2019. 1, 2, 3, 6, 7

A. Text Recognition Scheme

In this section, we provide additional details on the text recognition architecture components considered in this work. In particular, we focus on three components: (i) the transformation performed by the thin-plate splines (TPS) [56, 31], (ii) the CTC based decoder [22, 54] and (iii) the attention based decoder [3, 13, 38].

A.1. Transformation

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

In this work, we follow [3], and utilize the Thin Plate Spline (TPS) transformation [56, 31] which is a variant of the spatial transformer network [31]. As depicted in Fig. 9, in this transformation, we first detect a pre-defined number of fiducial points at the top and bottom of the text region. Then, we apply a smooth spline interpolation between the obtained points to map the predicted textual region to a constant pre-defined size.

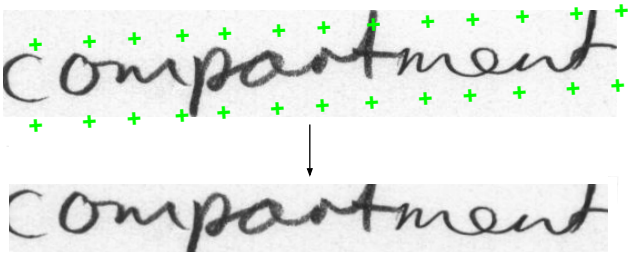

Figure 9: **TPS transformation.** This transformation first predicts fiducial points – marked as green points. Then, a smooth spline interpolation is employed, transforming these points to the border of a constant rectangle, yielding a normalized image with a fixed predefined size.

A.2. Connectionist Temporal Classification (CTC)

The CTC decoder [22] operates on a given sequential feature map $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_T]$, which in our framework can be $\mathbf{F} \in \{\mathbf{V}, \mathbf{H}, (\mathbf{H}, \mathbf{V})\}$. The inference phase consists of three stages. In the first stage, each frame $\mathbf{f}_t$ is transformed by a fully connected layer to yield $\mathbf{f}'_t$. Then, the CTC finds the sequence of characters with the highest probability:

$$\mathbf{c} = \arg\max_{\pi} \prod_{t=1}^{T} f'_{t,\pi_t}, \tag{2}$$

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

$$\mathbf{y} = \varphi(\mathbf{c}), \tag{3}$$

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

For the CTC procedure during training we refer the reader to [22, 54].

A.3. Attention decoder

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

$$e_{t',t} = \mathbf{a}^T \tanh(\mathbf{W}\mathbf{s}_{t'-1} + \mathbf{V}\mathbf{f}_t + \mathbf{b}), \tag{4}$$

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

$$\alpha_{t',t} = \frac{\exp(e_{t',t})}{\sum_{j=1}^{T} e_{t',j}}. \tag{5}$$

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

$$\mathbf{g}_{t'} = \sum_{t=1}^{T} \alpha_{t',t} \mathbf{f}_t. \tag{6}$$

Next, the recurrent cell is fed with:

$$(\mathbf{x}_{t'}, \mathbf{s}_{t'}) = \mathrm{RNN}(\mathbf{s}_{t'-1}, [\mathbf{g}_{t'}, f(\mathbf{y}_{t'-1})]), \tag{7}$$

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

$$\mathbf{y}_{t'} = \mathrm{softmax}(\mathbf{W}_0 \mathbf{x}_{t'} + \mathbf{b}_0), \tag{8}$$

where $\mathbf{W}_0, \mathbf{b}_0$ are trainable parameters. The loss used for the attention decoder is the negative log-likelihood, as in [13].

B. Data Augmentation

In this section, we provide additional details for reproducing our augmentation procedure, implemented using the `imgaug` [33] augmentation package. As described in Section 5, this augmentation pipeline is used for the self-supervised training, where we stochastically augment each image twice. In Fig. 10, we present different augmentation procedures that we examined in our work, which eventually led us to the final pipeline. Our default procedure consists of a random subset of the following operations.

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

Figure 10: **Illustrations of augmentation procedures.** We show different augmentation pipelines that were considered in this work (a), which led to the final augmentation pipeline. We also show augmentations examples using the SimCLR [9] pipeline (b).

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

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

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

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

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

A pseudo-code for the augmentation pipeline, written with the `imgaug` [33] package, is as follows.

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

C. Datasets

In this work, we consider the following public datasets for handwriting and scene text, see examples in Fig. 11:

- **RIMES** [23] handwritten French text dataset, written by 1300 different writers, partitioned into writer independent training, validation and test. This collection contains 66,480 correctly segmented words.
- **IAM** [42] handwritten English text dataset, written by 657 different writers, partitioned into writer independent training, validation and test. This collection contains 74,805 correctly segmented words.
- **CVL** [36] handwritten English text dataset, written by 310 different writers, partitioned into writer independent training and test. 27 of the writers wrote 7 texts and the other 283 writers wrote 5 texts.
- **SynthText** (ST)[24] contains 8M cropped scene text images which were generated synthetically. This dataset was utilized for training the scene text recognizer.
- **IIIT5K-words** (IIIT5K) [45] contains 2000 training and 3000 testing cropped scene text images from the Internet.
- **ICDAR-2003** (IC03) [41] contains 867 cropped scene text images.
- **ICDAR-2013** (IC13) [35] contains 848 training and 1015 testing cropped scene text.

The last three datasets were used just for validation and test sets as described Appendix D.

D. Implementation Details

Recognizer setting Unless otherwise specified, the text recognizer architecture consists of: a feature extraction

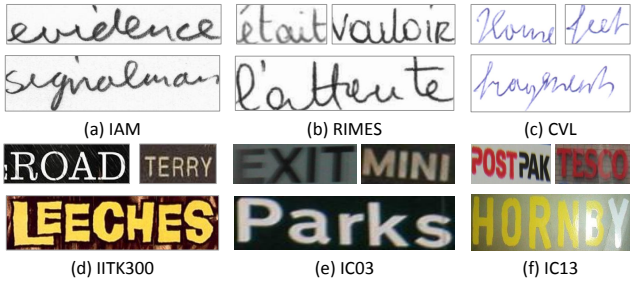

Figure 11: **Dataset samples.**

stage of a 29-layer ResNet [28], as in [14, 3]; a sequence modeling stage using a two-layer Bidirectional-LSTM (BiLSTM) with 256 hidden units per layer; and if needed, an attention decoder of an LSTM cell with 256 memory blocks. Following common practice in text recognition [3], we pre-resize all images to 32×100 both for training and testing. For the English datasets (IAM, CVL, IIIT5K, IC03 and IC13), we use 95 symbol classes: 52 case-sensitive letters, 10 digits and 33 for special characters. For the French dataset (RIMES), we add to the above the French accent symbols. As for special symbols for CTC decoding, an additional "[blank]" token is added to the label set. For the attention decoder, two special symbols are added: "[S]", "[EOW]" which indicate the start of the sequence and the end of the word.

SimCLR re-implementation To compare our method to non-sequential contrastive learning methods, we re-implement the SimCLR algorithm, with the same augmentations and projection head as in [9]. This algorithm can be applied in our settings as we anyhow resize each input image to a fixed width following the common practice in text recognition [3, 38].

Pre-training procedure In general, for the self-supervised training, we use a batch size of 1024, and train for 200K iterations for handwritten datasets and 400K iterations for scene-text. That said, since frame-to-one mapping results in many more instances for the contrastive loss (Figure 5(c)), we needed to reduce the batch size to 256. To compensate for it, we increased the number of iterations to 300K for handwritten datasets and to 600K for scene-text. For optimization, we use the AdaDelta optimizer [65] with a decay rate of 0.95, a gradient clipping parameter with a magnitude of 5 and a weight decay parameter of 10^{-4}. The learning rate is initialized to 10, and is reduced by a factor of 10 after 60% and 80% of the training iterations. Finally, all experiments are trained and tested using the PyTorch framework on 4 cards of Tesla V100 GPU with 16GB memory.

Decoder-evaluation and fine-tuning procedures For these stages, we train the decoder using a batch size of 256 for 50K iterations, employing a similar learning rate scheduling as in the self-supervised phase. The augmentation procedure consists of light cropping, linear contrast and Gaussian blur. We select our best model using a validation dataset, where in handwritten text we use the public validation sets, and in scene text our validation data is the union of the training data of IC13 and IIIT, as done in [3].