Expert Systems With Applications 165 (2021) 114196

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A comprehensive comparison of end-to-end approaches for handwritten digit string recognition

Andre G. Hochuli [a,*], Alceu S. Britto Jr [a], David A. Saji [b], José M. Saavedra [b], Robert Sabourin [c], Luiz S. Oliveira [d]

[a] Pontifical Catholic University of Parana (PUCPR), Curitiba, Brazil, R. Imaculada Conceição, 1155, Curitiba, PR 80215-901, Brazil
[b] Computer Vision Research Group, ORAND S.A, Estado 360, of 702, Santiago, Chile
[c] École de Technologie Supérieure (ÉTS), 1100 Notre Dame West, Montreal, Quebec, Canada
[d] Federal University of Parana (UFPR), Curitiba, Brazil, Rua Cel. Francisco H. dos Santos, 100, PR 81531-990, Brazil

ARTICLE INFO

Keywords:
Handwritten digit string recognition
Handwritten digit segmentation
Convolutional neural networks
Deep learning

ABSTRACT

Over the last decades, most approaches proposed for handwritten digit string recognition (HDSR) have resorted to digit segmentation, which is dominated by heuristics, thereby imposing substantial constraints on the final performance. Few of them have been based on segmentation-free strategies where each pixel column has a potential cut location. Recently, segmentation-free strategies has added another perspective to the problem, leading to promising results. However, these strategies still show some limitations when dealing with a large number of touching digits. To bridge the resulting gap, in this paper, we hypothesize that a string of digits can be approached as a sequence of objects. We thus evaluate different end-to-end approaches to solve the HDSR problem, particularly in two verticals: those based on object-detection (e.g., Yolo and RetinaNet) and those based on sequence-to-sequence representation (CRNN).

The main contribution of this work lies in its provision of a comprehensive comparison with a critical analysis of the above mentioned strategies on five benchmarks commonly used to assess HDSR, including the challenging Touching Pair dataset, NIST SD19, and two real-world datasets (CAR and CVL) proposed for the ICFHR 2014 competition on HDSR. Our results show that the Yolo model compares favorably against segmentation-free models with the advantage of having a shorter pipeline that minimizes the presence of heuristics-based models. It achieved a 97%, 96%, and 84% recognition rate on the NIST-SD19, CAR, and CVL datasets, respectively.

1. Introduction

Research in handwritten digit string recognition (HDSR) has picked up over the past few decades. Most works covering the subject share a common strategy, which involves segmenting a string into isolated digits and then applying a classifier capable of recognizing 10 classes (0…9). However, a straightforward solution becomes unfeasible in the presence of noise, broken digits, and in the worst case, touching digits. The impacts of the first two cases are reduced when some heuristic-based pre-processing modules are applied. The challenge, however remains over touching digits.

To handle the presence of touching digits, algorithms based on contour and profile information over segment the numerical string, generating components that may represent a digit or part of it. After each resulting component is classified, a fusion method determines the best combination among many hypotheses. The rationale behind over-segmentation is to maximize the chances of producing the correct segmentation, even at a high post-processing computational cost. This strategy is illustrated in Fig. 1. Readers interested in different global and local approaches may refer to Casey and Lecolinet (1996) and Ribas, Oliveira, Britto, and Sabourin (2013). These two works survey the state-of-the-art up to 2012, while the approaches proposed by Gattal and Chibani (2015) and Gattal, Chibani, and Hadjadji (2017) were the last attempts using the segmentation-based approach.

The alternative approaches resort to segmentation-free based methods (Choi & Oh, 1999; Procter, Illingworth, & Elms, 1998; Britto-

* Corresponding author.
E-mail addresses: aghochuli@ppgia.pucpr.br (A.G. Hochuli), alceu@ppgia.pucpr.br (A.S. Britto Jr), david.saji@ing.uchile.cl (D.A. Saji), jose.saavedra@orand.cl (J.M. Saavedra), robert.sabourin@etsmtl.ca (R. Sabourin), luiz.oliveira@ufpr.br (L.S. Oliveira).

https://doi.org/10.1016/j.eswa.2020.114196
Received 17 December 2019; Received in revised form 25 September 2020; Accepted 29 October 2020
Available online 9 November 2020
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

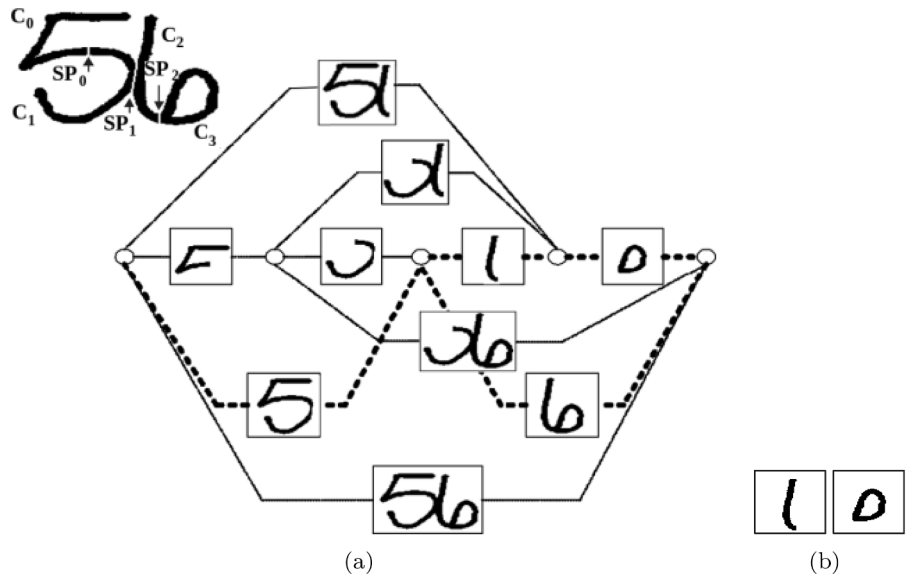


Fig. 1. (a) Segmentation paths for the string "56" and (b) Images that can be easily confused with digits "0" and "1" (extracted from Vellasques et al., 2008).

Jr, Sabourin, Bortolozzi, & Suen, 2003; Ciresan, Meier, & Schmidhuber, 2012; Hochuli, Oliveira, Souza Britto, & Sabourin, 2018) in which the string is recognized without the need for its a priori segmentation into isolated digits. This approach only recently started gaining attention among the research community, prodded by advances in machine learning thanks to deep learning techniques. While over-segmentation based methods demand certain specific strategies to generate segmentation cuts, a robust isolated digit recognizer, as well as a strategy for searching the best path among the generated segmentation hypothesis, the segmentation-free demands a significant amount of training data. Both strategies are caracterized by common complexes pipelines surrounded by handcrafted features, heuristic modules, and fusion rules to assembly task-specific classifiers. The need for an end-to-end approach is therefore evident.

Contrary to the handwritten digit string recognition, the object recognition field is evolving very rapidly. Each year, new algorithms surface and outperform the previous ones. Consequently, there presently are a plethora of ready-to-use pre-trained deep learning end-to-end models available (Redmon, Divvala, Girshick, & Farhadi, 2016; Redmon & Farhadi, 2017; Girshick, Donahue, Darrell, & Malik, 2014; Girshick, 2015; Ren, He, Girshick, & Sun, 2015; Lin, Goyal, Girshick, He, & Dollár, 2017). In the same vein, sequence-to-sequence based models (Voigtlaender, Doetsch, & Ney, 2016; Shi, Bai, & Yao, 2017; Dutta, Krishnan, Mathew, & Jawahar, 2018) have produced end-to-end solutions for temporal series, handwritten text, and text scene recognition. Besides high performance, these approaches contribute significantly by providing a reduced number of handcrafted features and heuristics methods, producing a straightforward pipeline as compared to related state-of-the-art works.

In discussing end-to-end approaches, one aspect that is very often highlighted in the literature is the importance of context. Several recent computer vision approaches have demonstrated that the use of context

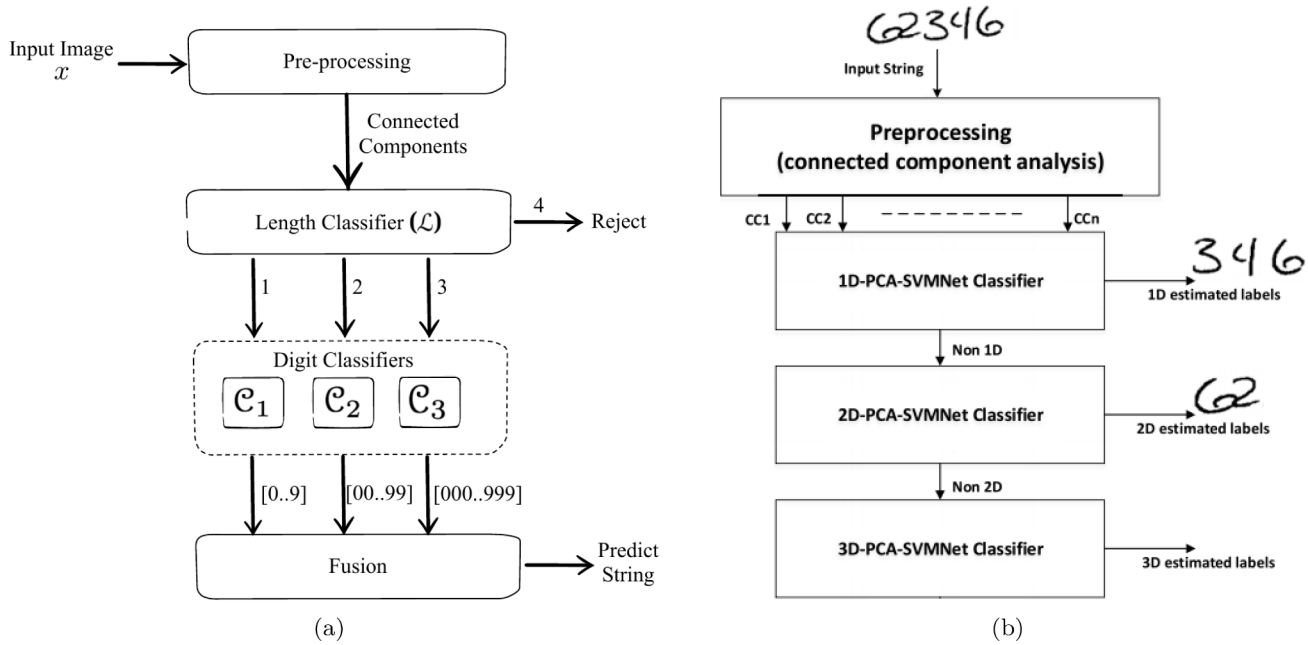


Fig. 2. Dynamic Selection approaches proposed by (a) Hochuli et al. (2018) and (b) Aly and Mohamed (2019).

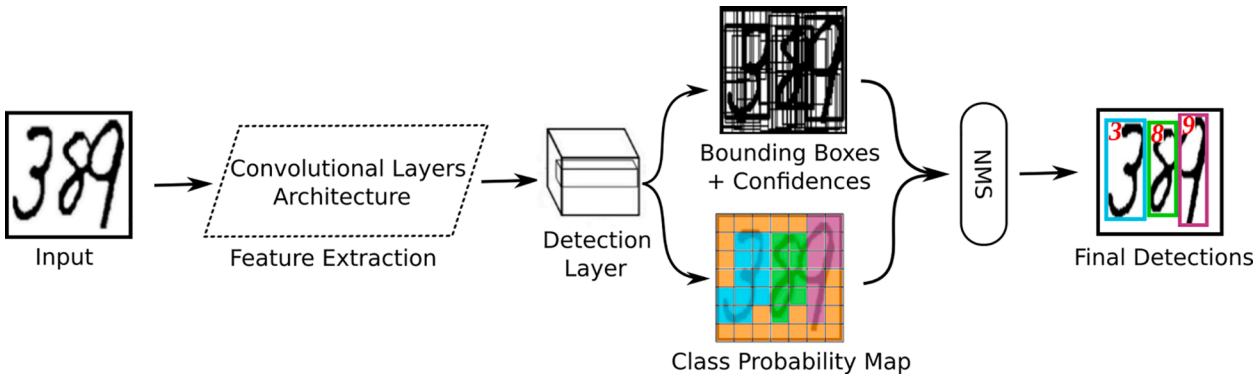


Fig. 3. The Yolo framework divides the image into a grid and for each cell predicts bounding boxes and classes.

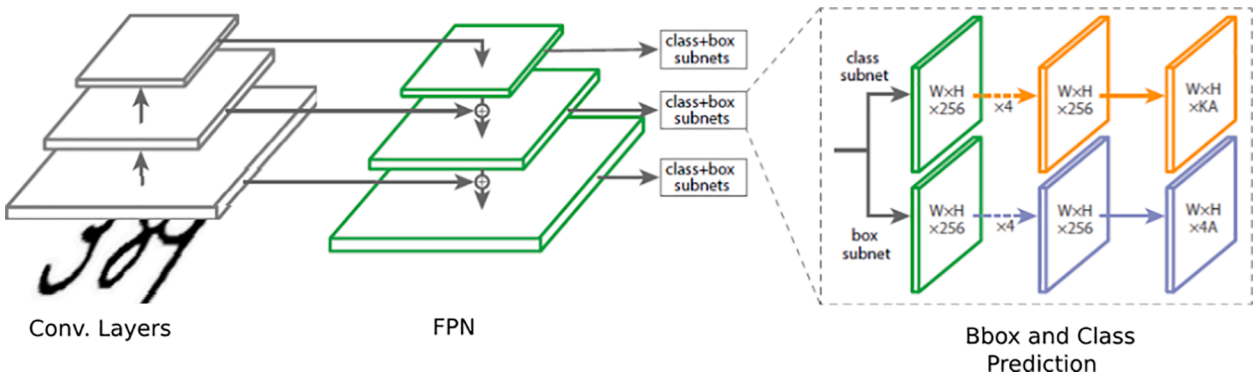


Fig. 4. RetinaNet Framework: A Feature Pyramid Network (FPN) on top of convolutional layers produces rich and multi-scale features from one single input. Moreover, the proposed loss function (*focal*) improved the class imbalance issue among background and foreground samples.

improves recognition performance (Divvala, Hoiem, Hays, Efros, & Hebert, 2009). In the case of digit string recognition, contextual information is more limited, but it nonetheless plays a vital role, as demonstrated in Oliveira, Sabourin, Bortolozzi, and Suen (2002).

In this paper, we argue that a string of digits is a sequence of objects. Therefore, we restricted our scope to the following neural network-based approaches: (a) Yolo (Redmon et al., 2016; Redmon & Farhadi, 2017), (b) RetinaNet (Lin et al., 2017) which is a state-of-the-art architectures for object detection/recognition, and (c) CRNN (Shi et al., 2017), a sequence-to-sequence model composed of a convolutional network combined with a long-short term memory (LSTM) (Schuster & Paliwal, 1997). To complete our analysis, we also consider two approaches based on dynamic selection (Hochuli, Oliveira, Britto, & Sabourin, 2018 and Aly & Mohamed, 2019). To deploy end-to-end approaches for this problem, we generate a large dataset of strings mimicking real datasets, which provides contextual information for training. Even though Zhan, Wang, and Lu, 2017 applied CRNN for courtesy amount recognition on bank checks, we provide an in-depth analysis of this model based on different challenging benchmarks, and compare it with other end-to-end approaches, such as those that are object detection-based.

The main contributions of this work lies in its provision of a comprehensive comparison, along with a critical analysis of the end-to-end object recognition strategies, sequence-to-sequence approaches used for handwritten words, and the recently published specific segmentation-free HDSR methods. Our extensive experimental protocol include experiments on the following benchmarks: (i) Touching Pair (TP) dataset (Ribas et al., 2013), which contains 79,464 touching digits and has been used a benchmark for both heuristic-based and segmentation-free algorithms; (ii) 570,000 images of strings composed of 2-, 3-, and 4-touching digits; (iii) NIST SD19, which is composed of 11,585 real-world numerical strings, ranging from 2 to 6 digits, and (iv) ICFHR 2014 competition (Diem et al., 2014), which contains real courtesy amount of bank checks and a significant variability of handwritten styles.

Our experimental analysis shows the limits of the proposed strategies for the HDSR. End-to-end approaches, especially in the Yolo model, compare favorably against the segmentation-free methods in Hochuli et al. (2018) and Hochuli et al. (2018) with the clear advantage of having a shorter pipeline that minimizes the presence of heuristic-based modules, such as those pre-processing. On the other hand, bottlenecks associated with the laborious task of annotation of ground-truths when synthetic data are not applicable and the lack of lexicon for digit strings is a matter of discussion.

This paper is organized as follows: Section 2 examines related works. The problem statement is presented in Section 3. A detailed review of architectures is given in Section 4. In Section 5, we tackle the approaches using the aforementioned datasets. Finally, Section 6 concludes this work.

2. Related works

To avoid the burden of over-segmentation, some authors have devoted efforts towards segmentation-free approaches. To the best of our knowledge, the first attempt in this direction was in the Space Displacement Neural Network (SDNN) introduced by Matan, Burges, LeCun, and Denker (1992). This strategy produces a series of output vectors used by a post-processor to extract the best possible label sequence from the vector sequence. As stated by LeCun, Bottou, Bengio, and Haffner (1998), SDNN is an attractive technique but has not managed to yield better results than heuristic over-segmentation methods.

The Hidden Markov Model (HMM), initially developed in the field of speech recognition, has been used to build segmentation-free methods

for handwriting recognition. Elms, Procter, and Illingworth (1998) first applied HMM to word recognition and then adapted their work to classify handwritten digit strings of unknown length (Procter et al., 1998). Britto-Jr et al. (2003) revisited these two studies and proposed a two-stage segmentation-free method using features extracted from lines and columns that are processed by a set of HMMs. This framework achieved an average recognition rate of 91.0% in NIST-SD19.

Choi and Oh (1999) designed a framework based on 100 neural networks to avoid the segmentation of touching pairs. Their approach achieves 95.3% of the recognition rate of touching pairs extracted from NIST-SD19 (Grother, 2016). A decade later, Ciresan (2008) took advantage of Convolutional Neural Networks by training two CNNs, one for isolated digits and one for touching pairs. The authors combined these two networks to recognize 3-digit strings of the NIST database achieving a 93.4% recognition rate. At that time, strings with three digits connected were not considered.

Another decade later, advances in the field of machine learning, especially with the popularization and better understanding of deep learning techniques (Bengio, Courville, & Vincent, 2013; Gu et al., 2017), lead to advances in different areas of handwriting recognition, such as digit recognition (Das, Saha, & Nasipuri, 2016; Sabour, Frosst, & Hinton, 2017), character recognition (Xiao et al., 2017; Laroca et al., 2018; Laroca et al., 2019), word recognition (Roy, Bhunia, Das, Dey, & Pal, 2016; Tamen, Drias, & Boughaci, 2017; Wua, Yin, & Liu, 2017), script identification (Ziyong, Zhaoyang, Shuanping, & Jun, 2017), and signature verification (Hafemann, Sabourin, & Oliveira, 2017). Leveraging this evolution, Hochuli et al. (2018) introduced a segmentation-free approach capable of recognizing digit strings of any size. In their work, the authors combined four CNNs into a Dynamic Selection (DS) scheme (Britto, Sabourin, & Oliveira, 2014; Cruz, Sabourin, & Cavalcanti, 2018). The first CNN works as a high-level classifier that determines the size of components, while the other three operate at a low-level by classifying 1-digit, 2-digit, and 3-digit components, respectively. This approach achieved the state-of-the-art for NIST-SD19 and Touching Pairs (Ribas et al., 2013) datasets, surpassing segmentation-based and segmentation-free methods.

Despite this good performance, this approach has certain limitations. First, it is based on a hierarchical framework composed of heuristic-based pre-processing and four classifiers, which leads to various error sources. Second, the strategy recognizes strings of any size but limited to 3-digit touching. To mitigate some of these problems, Hochuli et al. (2018) reduced the number of classifiers by introducing a single classifier ($\mathscr{C}_{1110}$) capable of classifying 1110 classes (0…9, 00…99, and 000…999). Although these approaches achieve high recognition rates, they are still carried by complex pipelines, and are surrounded by heuristic processes, pre-processing modules, and fusion strategies.

Recently, sequence-to-sequence architectures have been successfully applied to the tasks of handwritten text recognition and scene text recognition (Voigtlaender et al., 2016; Shi et al., 2017; Dutta et al., 2018). Those solutions combine a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) to produce a sequence of probabilities interpreted by a transcription layer. This pipeline produces an end-to-end trainable model which achieves state-of-art performance of handwritten text recognition. However, it relies on a specific lexicon to mitigate confusions.

In object recognition, the main goal is to detect and recognize a set of predefined classes of objects in a given input image. Until the last decade, a classical approach used to be based on a sliding window and its variants (Lampert, Blaschko, & Hofmann, 2008; Felzenszwalb, McAllester, & Ramanan, 2008; Felzenszwalb, Girshick, McAllester, & Ramanan, 2010). This approach uses a classifier trained with handcrafted features at several spatial locations of the image. A limitation is the high number of windows needed to search over multiple scales and aspect ratios. Moreover, in this exhaustive search strategy, the computational cost increases very rapidity.

A breakthrough occurred due to the arising of large-scale datasets (Russakovsky et al., 2015; Lin et al., 2014), the popularization of GPUs and the popularization of deep networks in the ILSVRC 2012 (Russakovsky et al., 2015). At that time, this field had recovered the attention of the research community, and several deep learning-based methods were proposed to improve the state-of-art (Han, Zhang, Cheng, Liu, & Xu, 2018).

One of the first successful approaches in this regard consisted of the Region-based Convolutional Network (R-CNN) proposed by Girshick et al. (2014). This architecture begins by extracting region proposals from the image space using the selective search algorithm (Uijlings, van de Sande, Gevers, & Smeulders, 2013). Then, each region is warped to a fixed size, and a CNN extracts features. Finally, an SVM classifier determines a class, and a bounding-box regressor refines the locations. The main drawback of this strategy is that it requires the extraction of features of each warped region proposal, which is computationally expensive.

To overcome this obstacle, SPPnet (He, Zhang, Ren, & Sun, 2015) and Fast-RCNN (Girshick, 2015) have been proposed. These models predict region proposals direct over feature maps. A spatial pooling layer is introduced to produce fixed-length representations (wrapping at feature level). Although these strategies speed up the entire process,

Table 1
Architectures of Darknet (left) and ResNet-50 (right). In the ResNet-50, a downsampling with a stride of 2 is performed after each convolutional block.

Darknet (Yolo)			
Layer	Type	Filters	Size/Stride
#1	Conv.	32	3 × 3/ 1
#2	Maxpool		2 × 2/ 2
#3	Conv.	64	3 × 3/ 1
#4	Maxpool		2 × 2/ 2
#5	Conv.	128	3 × 3/ 1
#6	Conv.	64	1 × 1/ 1
#7	Conv.	128	3 × 3/1
#8	Maxpool		2 × 2/ 2
#9	Conv.	256	3 × 3/ 1
#10	Conv.	128	1 × 1/ 1
#11	Conv.	256	3 × 3/ 1
#12	Maxpool		2 × 2/ 2
#13	Conv.	512	3 × 3/ 1
#14	Conv.	256	1 × 1/ 1
#15	Conv.	512	3 × 3/ 1
#16	Conv.	256	1 × 1/ 1
#17	Conv.	512	3 × 3/ 1
#18	Maxpool		2 × 2/ 2
#19	Conv.	1024	3 × 3/ 1
#20	Conv.	512	1 × 1/ 1
#21	Conv.	1024	3 × 3/ 1
#22	Conv.	512	1 × 1/ 1
#23	Conv.	1024	3 × 3/ 1
#24	Conv.	1000	1 × 1
#25	Avgpool		Global
#26	Softmax		

ResNet-50 (RetinaNet)		
Layer	Type	Filters
#1	Conv.	7×7, 64, stride 2
#2	Max-Pool	3×3, stride 2
#3..11	Conv.	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
#12..23	Conv.	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 3$
#24..41	Conv.	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 3$
#42..50	Conv.	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
#51		Avgpool
#52		1000-d FC
#53		Softmax

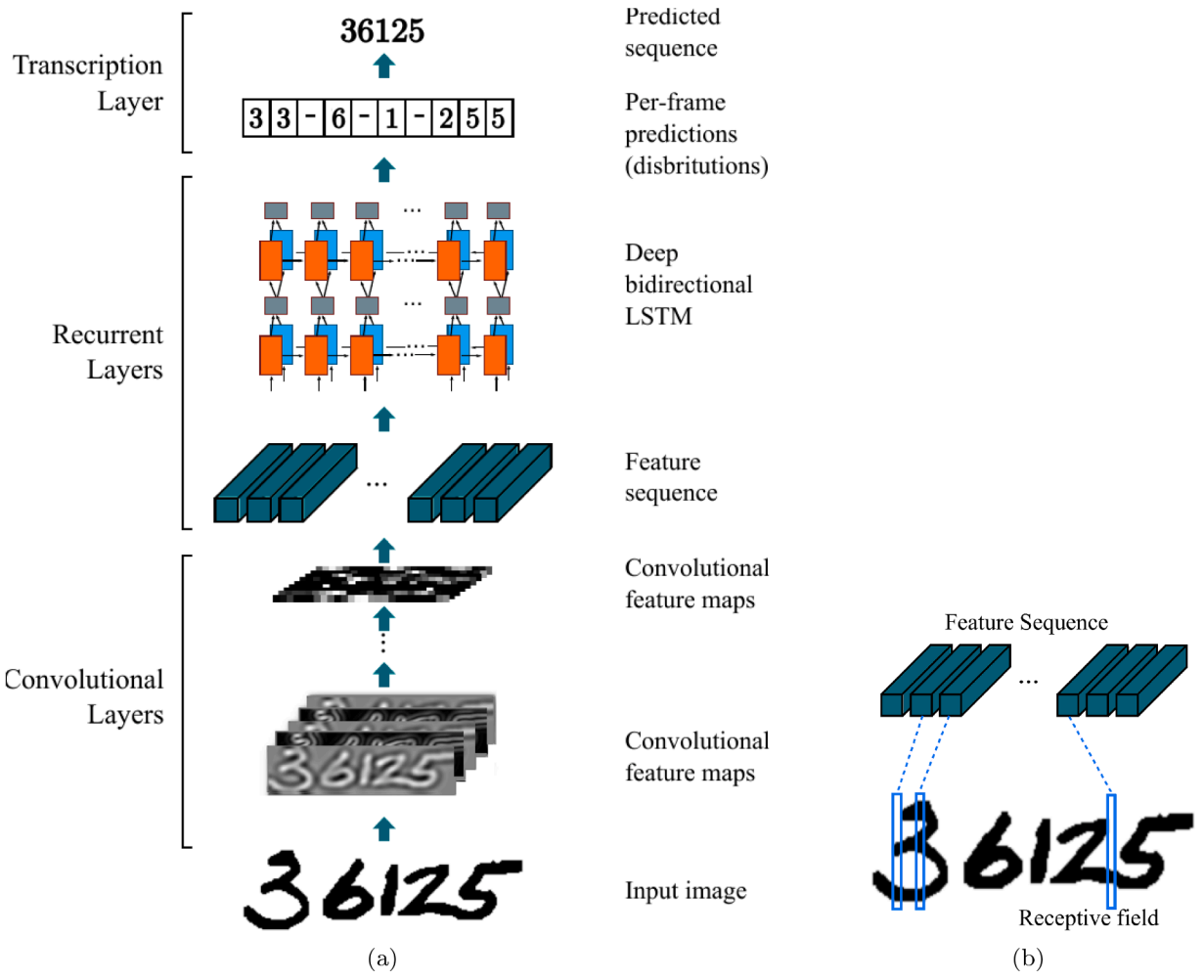


Fig. 5. CRNN architecture proposed by Shi et al. (2017): (a) the pipeline from convolutional layers to transcription layer and (b) the receptive field for each feature vector.

Table 2
CRNN Architecture proposed by Shi et al. (2017).

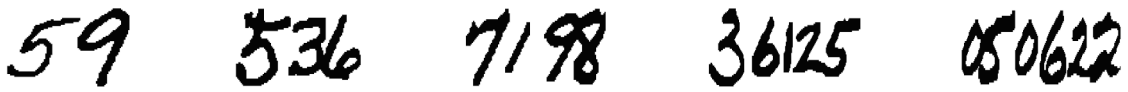
Layer	Type	Filters	Size/Stride
#1	Convolutional	64	3 × 3/ 1
#2	Maxpool		2 × 2/ 2
#3	Convolutional	128	3 × 3/ 1
#4	Maxpool		2 × 2/ 2
#5	Convolutional	256	3 × 3/ 1
#6	Convolutional	256	3 × 3/ 1
#7	Maxpool		1 × 2/ 2
#8	Convolutional	512	3 × 3/ 1
#9	BatchNormalization		
#10	Convolutional	512	3 × 3/ 1
#11	BatchNormalization		
#12	Maxpool		1 × 2/ 2
#13	Convolutional	512	2 × 2/ 1
#14	Map-to-Sequence		
#15	Bidirecional-LSTM	256 (hidden units)	
#16	Bidirecional-LSTM	256 (hidden units)	
#17	Transcription		

they still rely on a handcrafted region proposal method. To overcome this limitation, He, Gkioxari, Dollár, and Girshick (2017) introduced a region proposal network (RPN), which implicit produces candidate locations. With this approach, the features produced by the last convolutional layer are used on both (a) region proposal and (b) region classification tasks.

Despite their advatages, the above approaches must still handle a two-stage pipeline whenever a region proposal strategy is needed, regardless of whether or not this need is implicit. A more ingenious alternative was proposed by Redmon et al. (2016) with the Yolo architecture, in which the authors proposed a regression-based approach that encapsulates all stages into a single network. With a single forward pass, the network provides bounding box locations and class probabilities. An essential aspect of Yolo is that it can encode the context and appearance from the neighborhood of objects, which is an important feature for implicit digit segmentation. A year later, the RetinaNet (Lin et al., 2017) was proposed and add a Feature Pyramid Network (FPN) to produce multi-scale features. Its novelty lay in its introduction of an improved loss function known as *focal loss* to deal with class imbalance among background and foreground samples, which stifles the learning process as most image locations contain no objects. Although the RetinaNet achieves the state-of-art in object detection benchmarks, Yolo provides a good tradeoff between speed and accuracy.

3. Problem statement

As stated earlier, traditional approaches address the problem by grouping foreground pixels into connected components, and then classifying them. The main problem with in scenario is that when a group of pixels is extracted from an image, only a local view of the problem is obtained, with a lot of contextual information eliminated. Without this valuable information, the algorithms suffer from the presence of noise



(a) (b) (c) (d) (e)

Fig. 6. Synthetic data representing numerical strings ranging from 2 to 6 digits.

Table 3
Distribution of the synthetic dataset.

Length/Classes	Samples	Authors	Purpose
2-Digit String	42,614	1000–1599	Training
	14,202	1600–1799	Validation
	14,838	1800–1999	Testing
3-Digit String	76,890	1000–1599	Training
	25,570	1600–1799	Validation
	27,025	1800–1999	Testing
4-Digit String	82,625	1000–1599	Training
	27,487	1600–1799	Validation
	29,166	1800–1999	Testing
5-Digit String	82,944	1000–1599	Training
	27,663	1600–1799	Validation
	29,371	1800–1999	Testing
6-Digit String	82,926	1000–1599	Training
	27,609	1600–1799	Validation
	29,396	1800–1999	Testing

and touching digits.

An end-to-end approach addresses this problem holistically. Deep learning models can learn the interaction between digits in the context of an image, which contains noise, touching, overlapping, and broken digits. Therefore, end-to-end approaches usually have short pipelines: the object detector $\mathscr{D}$ receives as input an image I containing n digits (objects) and produces as output the location (bounding boxes) and the digit classes $[0,\ldots,9]$ associated with an estimation of the posterior probability. Considering that the input image I may contain n connected components, the most probable interpretation of the written amount M is given by Eq. (1). It is worth noting that the CRNN approach does not provide bounding box locations because it does not implement bounding box regressors. However, the digit's location may be estimated by the receptive fields of the feature sequence (Fig. 5b).

$$P(M|I) = \prod_{i=1}^{n} P(\omega_j|x_i) \tag{1}$$

where $\omega_i = \{0\ldots9\}$ and x_i stands for the digits candidates.

4. End-to-end strategies for HDSR

In this section, we present all the approaches evaluated in our work. Section 4.1 describes the dynamic selection approaches proposed by Hochuli et al. (2018) and Aly and Mohamed (2019), which represented a breakthrough in the HDSR field as they introduced a set of classifiers to produce a segmentation-free solution for the HDSR field. Section 4.2 describes the object detection approaches (Yolo and RetinaNet), while Section 4.3 describes the sequence-to-sequence framework (CRNN). The training protocol used for all models is presented in Section 4.4.

To ensure a fair evaluation, we used the source code provided by the authors whenever they were available. The repositories for the approaches reported in Hochuli et al. (2018), Redmon et al. (2016) and Lin et al. (2017) are available in,[1,2] and,[3] respectively. In the case of the CRNN, the original code[4] was outdated, and therefore, we used a more recent version.[5] Aly and Mohamed (2019) did not share their source code, and as a result, in this paper, we replicate the results reported by the authors.

Table 4
Average recognition time of end-to-end approaches.

Method	#Models (#Classes)	Recognition (sec)[1]	
		1-Digit	3-Digit
CRNN	1 (10)	0.001	0.001
Yolo	1 (10)	0.010	0.011
Hochuli et al., 2018	4 (1114)	0.060	0.062
RetinaNet	1 (10)	0.160	0.161

[1] NVIDIA Titan Xp GPU.

4.1. Dynamic selection approaches

The dynamic selection framework proposed by Hochuli et al. (2018) is depicted in Fig. 2a. Here, a digit string x is first classified by the Length classifier ($\mathscr{L}$), which will assign a probability of having 1, 2, 3, or 4 touching digits. The digit classification module comprises three classifiers ($\mathscr{C}_1$, $\mathscr{C}_2$, $\mathscr{C}_3$) designed to discriminate 10 [0…9], 100 [00…99], and 1000 [000…999] classes. The classifiers that will be used for a given image depend on the output of the Length Classifier. In accordance with a fusion rule, more than one digit classifier may be invoked to mitigate any possible confusion.

The fusion rule used in this case considers the Top-2 outputs of $\mathscr{L}$. Let $\mathscr{L}^i(x) = p^i(x)$ be the probability of the input pattern, and let x be composed of i, $(i = 1, 2, 3, 4)$ digits. Let $\mathscr{C}_1(x) = \max_{0\leqslant i\leqslant 9} p^i(x)$, $\mathscr{C}_2(x) = \max_{0\leqslant i\leqslant 99} p^i(x)$, and $\mathscr{C}_3(x) = \max_{0\leqslant i\leqslant 999} p^i(x)$ be the probability produced by 10-class, 100-class, and 1000-class classifiers, respectively, for the input pattern x. Let Top1($\mathscr{C}$) and Top2($\mathscr{C}$) be the functions that return the classes with first and second highest scores of a given classifier $\mathscr{C}$, respectively. Then, x is assigned to the class $\omega \in [0\ldots1110]$, according to Eq. (2),

$$P(\omega|x)\begin{cases} \text{if } \mathscr{L}(x) < T, & \max(\mathscr{C}_{Top1(\mathscr{L})}(x), \mathscr{C}_{Top2(\mathscr{L})}(x)) \\ \text{otherwise}, & \mathscr{C}_{Top1(\mathscr{L})}(x) \end{cases} \tag{2}$$

where T is a threshold defined empirically on the validation set.

The authors justify dealing with 1, 2, and 3 touching digits because most of the touching occurs between two digits and sometimes between three digits (Wang, Govindaraju, & Srihari, 2000). Strings composed of more than three touching digits are rare in real problems, and where one occurs, it is rejected by $\mathscr{L}$.

An alternative approach, depicted in Fig. 2b, was proposed by Aly and Mohamed (2019). In this case, the length classifier and the fusion

[1] https://github.com/andrehochuli/digitstringrecognition.
[2] https://pjreddie.com/darknet/yolov2/.
[3] https://github.com/facebookresearch/Detectron.
[4] https://github.com/bgshih/crnn.
[5] https://github.com/yalecyu/crnn.caffe.

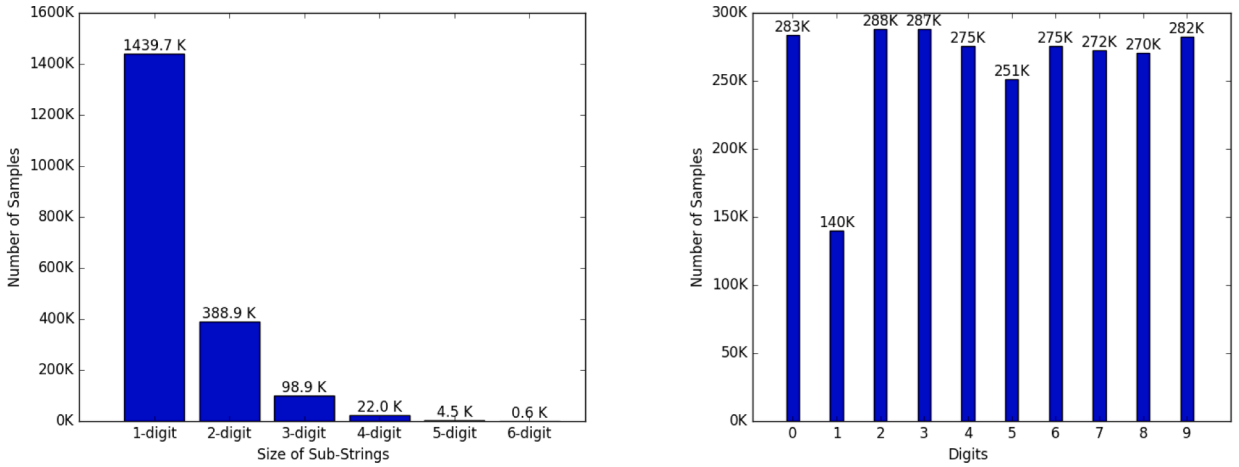


Fig. 7. Distribution of the dataset (a) Distribution regarding isolated and touching digits, and (b) Distribution of the 10 classes of digits in the database.

Table 5
Performance of the segmentation algorithms (reported in Ribas et al., 2013; Hochuli et al., 2018; Gattal and Chibani, 2015), in terms of correct segmentation, on the TP Database.

Strategy	Method	Performance	Connection Type (%)				Segmentation
		%	I	II	III	V	Cuts
Seg-Based	Shi and Govindaraju (1997)	59.30	68.31	59.72	60.35	25.44	1
	Congedo et al. (1995)	63.07	62.88	67.51	59.40	40.45	1
	Lacerda and Mello (2013)	65.79	71.75	71.21	63.64	56.57	1
	Elnagar and Alhajj (2003)	67.34	63.88	71.51	56.40	58.73	1
	Pal et al. (2003)	71.21	73.96	74.69	80.09	41.52	1
	Oliveira et al. (2000)	88.03	90.40	90.78	89.01	64.88	1
	Fujisawa et al. (1992)	89.85	95.45	91.27	83.57	63.72	3.66
	Fenrich and Krishnamoorthy (1990)	92.37	97.54	93.79	99.45	65.57	4.07
	Gattal and Chibani (2015)	93.24	96.67	93.75	99.68	77.58	24.11
	Chen and Wang (2000)	93.80	97.87	94.23	97.55	76.76	45.40
Seg-Free	CRNN	68.58	68.52	64.19	84.83	56.81	0
	RetinaNet	88.48	89.95	88.51	97.15	78.32	0
	Aly and Mohamed (2019)	95.05	95.65	96.20	97.15	91.21	0
	Yolo	96.53	96.98	97.64	98.97	92.55	0
	Hochuli et al. (2018)	97.12	97.02	97.89	98.97	93.03	0

rule were eliminated by a cascade architecture of PCA-SVMNet classifiers, which is a combination of PCA-Convolutional layers used to extract features and a linear multi-class SVM to predict classes. An extra class was introduced on each classifier as rejection, i.e., for the isolated digit classifier (10[0…9]), the class '11' contains samples of touching digits ([00…999]). The number of classes of each SVM classifier increases according to the level on the cascade.

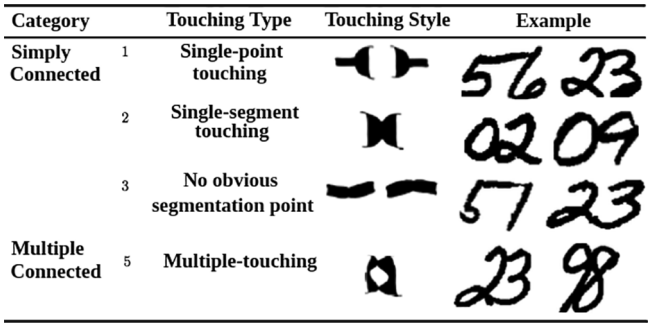


Fig. 8. Types of connected numeral string (extracted from Ribas et al., 2013).

4.2. Object detection approaches

Yolo (Redmon et al., 2016) is a general-purpose object detection framework that can be trained in an end-to-end fashion. Using a single network and looking at the entire image, it can predict bounding boxes and classes with a single forward pass instead of applying the model at every location as in the case with traditional sliding window or region purpose-based methods (Girshick, 2015; Ren et al., 2015). The framework is illustrated in Fig. 3.

First, the convolutional layers (see Section 4.2.1) extract features from the entire image, and then the detection layer divides the image into a grid. Next, each grid cell predicts the coordinates of bounding boxes, and the confidence of each box encloses an object. Handpicked anchor boxes are preliminary defined to help the network learn how to predict the right bounding boxes. Moreover, it provides class probabilities for the cells belonging to a given object. Finally, to mitigate confusion among overlapped boxes, the Non-Maximum Suppression (NMS) algorithm is used.

The input resolution of the Darknet reported in Redmon et al. (2016) is 416 × 416. However, given that strings of digits are usually wider than higher, we used an initial input size of 128 × 256 (height × width) to train the model. It is worth mentioning, though, that this architecture does not set the input image size. Rather, it changes the network after every few iterations. After, every ten batches, the network randomly

Fig. 9. Missed detections of Yolo for TDP dataset: (a) '51' as '57', (b) '21' as '24', (c) '12' as '62' and (d) '76' as '7'.

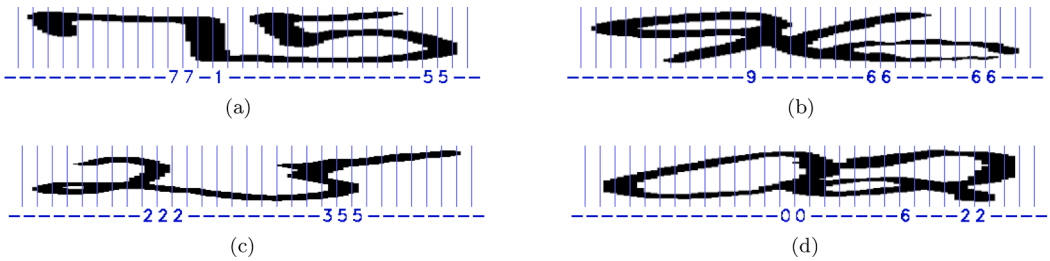


Fig. 10. Missed predictions of CRNN for TP dataset: (a) '75' as '715' (TYPE-I), (b) '96' as '966' (TYPE-II), (c) '25' as '235' (TYPE-III) and (d) '02' as '062' (TYPE-V).

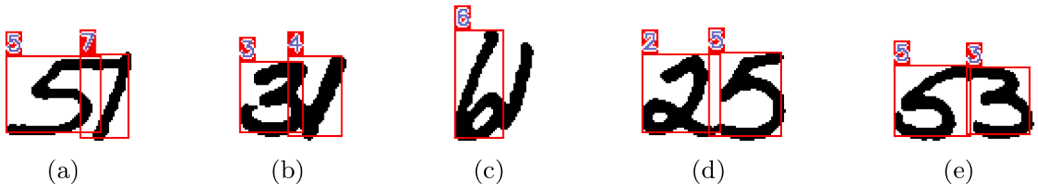


Fig. 11. Detections of RetinaNet for TP dataset: (a) '51' as '57', (b) '31' as '34', (c) '61' as '6', representing missed prediction, and (d) '25' as '25' and (e) '53' as '53' representing correct predictions.

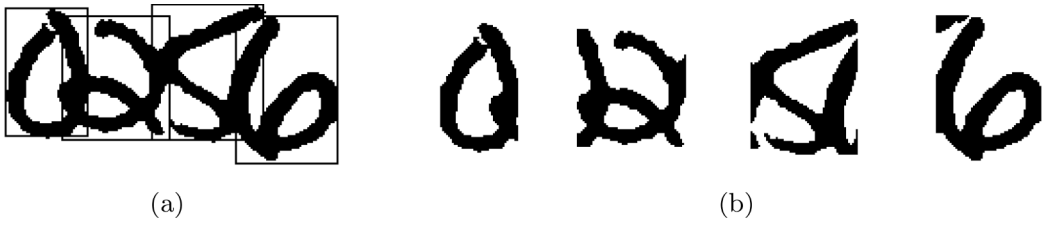


Fig. 12. (a) Ground truth for a 4-digit string (0256) and (b) Shape of digits impacted by its neighbors.

chooses a new image dimension size, and the training is resumed. This forces the network to learn to accurately predict across a variety of input dimensions. In Section 5.5, we show through experiments that during recognition, the input size can be easily defined as a function of the testing input image. Because Yolo looks at the whole input, it implicitly encodes contextual information about objects and their neighborhood.

The RetinaNet architecture (Lin et al., 2017) is depicted in Fig. 4. A Feature Pyramid Network (FPN) on the top of convolutional layers produces rich and multi-scale features based on a single input resolution. Compared with Yolo, both frameworks have a similar workflow despite these slight changes: the convolutional layers produce features to bounding box regressors and class predictors, which, with the aid of anchors boxes, determine locations and classes for objects in the input image. 4.2.1 provides detailed information about convolutional layers as well as a definition of anchors.

What distingues RetinaNet from other approaches is its proposed loss function, also know as the *focal loss*. The authors evidence that a significant issue encountered in most object detection approaches is the class imbalance that exists among foreground and background samples. Since most image locations do not contain an object of interest, the ratio between foreground and background locations is about 1:100 or even 1:1000. Therefore, the background samples dominate the loss gradient, and consequently, the result is a biased model. The solution proposed is to define a loss function that penalizes "easy" classified samples.

Let the cross-entropy loss (CE) for classification be:

$$\mathrm{CE}(p,y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases} \tag{3}$$

where $y \in \{\pm 1\}$ denotes the ground-truth class and $p \in [0,1]$ is the estimated probability for the class with label $y = 1$. For the sake of simplicity, let p_t be:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \tag{4}$$

Table 6
Accuracy of the segmentation-free approaches on the synthetic data. (The best performances are highlighted in bold).

Method	Isolated digit	2-digit	3-digit	4-digit
Hochuli et al. (2018)	**99.56**	**99.00**	94.88	–
CRNN	21.97	65.33	84.29	90.61
RetinaNet	86.63	87.32	81.58	77.52
Yolo	99.42	98.68	**96.89**	**95.50**

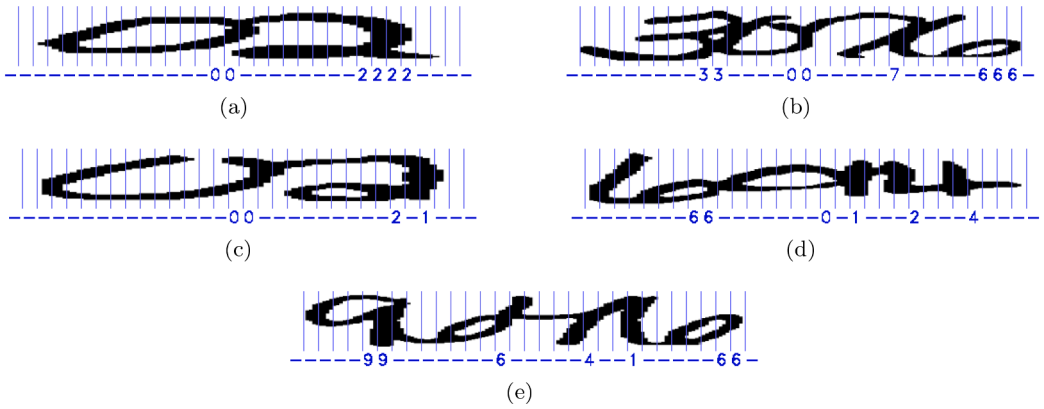


Fig. 13. Predictions of sequence-to-sequence approach: (a) '02' as '02' and (b) '3076' as '3076' representing correct predictions, (c) '02' as '021', (d) '6014' as '60124' and (e) '9646' as '96416' representing missed predictions.

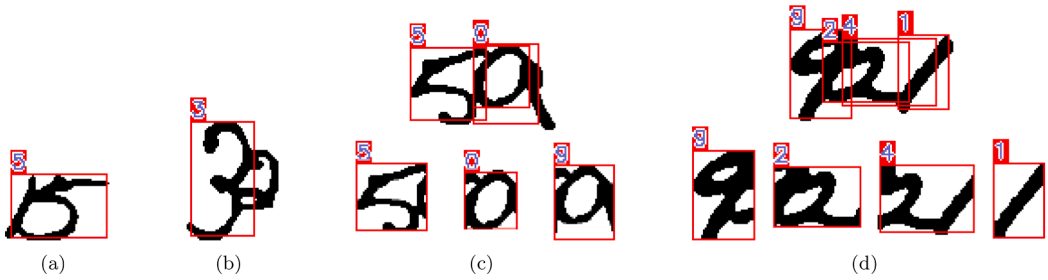


Fig. 14. Missed predictions of RetinaNet: (a) '15' as '5', (b) '32' as '3', (c) '59' as '509' and (d) '921' as '9241'.

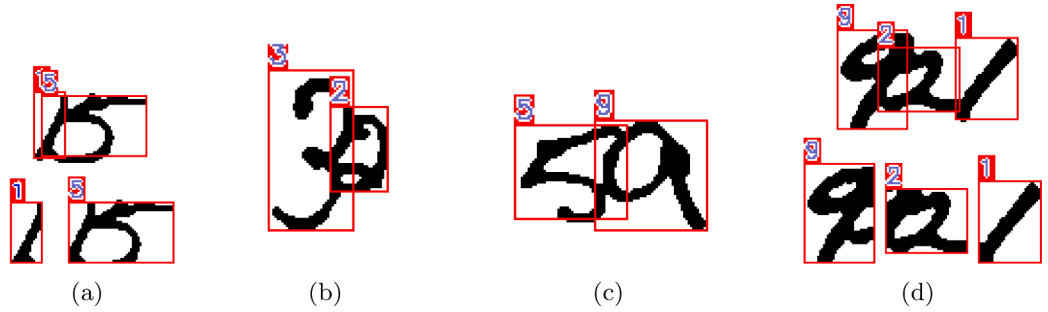


Fig. 15. Correct predictions of Yolo approach: (a) '15' as '15', (b) '32' as '32', (c) '59' as '59' and (d) '921' as '921'.

Finally, $\mathrm{CE}(p,y) = \mathrm{CE}(p_t) = -\log(p_t)$.

Once a weighting factor ($-\alpha_t \log(p_t)$) should balance the priority of background and foreground, it does not give attention to easy or hard samples. Therefore, the author proposes to add a modulating factor $(1-p_t)^\gamma$ to the cross-entropy loss, with tunable focusing parameter $\gamma \geqslant 0$:

$$\mathrm{FL}(p_t) = -\alpha_t (1-p_t)^\gamma \log(p_t). \tag{5}$$

When an example is misclassified and p_t is small, the modulating factor is close to 1, and the loss is unaffected. As $p_t \to 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted. The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, FL is equivalent to CE, and as γ is increased, the effect of the modulating factor is likewise increased.

4.2.1. Network architectures

The network architectures used by both Yolo and RetinaNet are presented in Table 1. Yolo was first introduced with an architecture called Darknet (Redmon & Farhadi, 2017) to perform the classification of 1000 object categories. It is composed of 19 convolutional layers and 5 max-pooling layers. To perform detection, they suppressed the last convolutional layer and added three 3×3 convolutional layer with 1024 filters.

The concept of residual networks (ResNet) was introduced by He, Zhang, Ren, and Sun (2016) to deal with the vanish gradient issue in deep networks. It provided a breakthrough as it allowed to skipping connections between convolution blocks. Using this concept, the authors

Table 7
Recognition rates for 2- to 6-digit strings of NIST SD19 dataset.

Method	Recognition Rate (%)	Error (%)	
		Classification	Detection
Yolo	97.1	2.4	0.5
Aly and Mohamed (2019)	96.1	N/A	N/A
Hochuli et al. (2018)	95.2	3.9	0.9
CRNN	80.3	11.8	7.9
RetinaNet	75.3	1.5	23.2

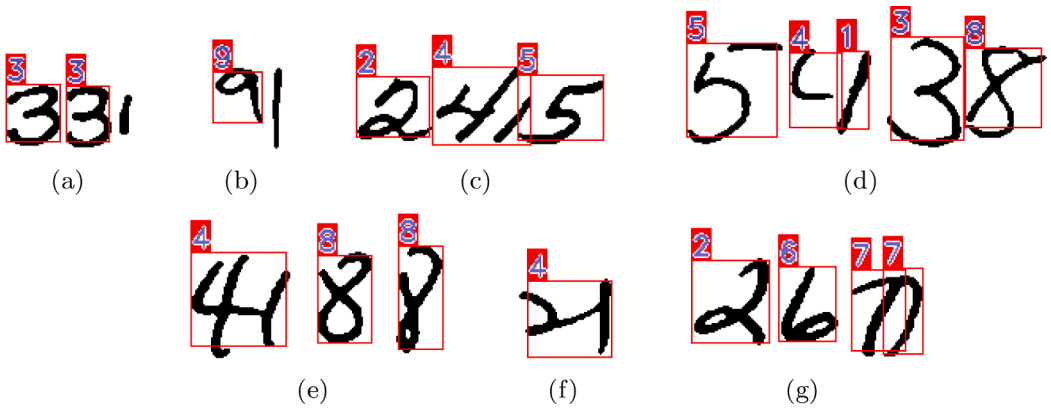


Fig. 16. Detection problems: (a) 331 recognized as 33, (b) 91 recognized as 9, (c) 2415 recognized as 245, (d) 5438 recognized as 54138, (e) 4188 recognized as 488, (f) 21 recognized as 4, and (g) 260 recognized as 2670.

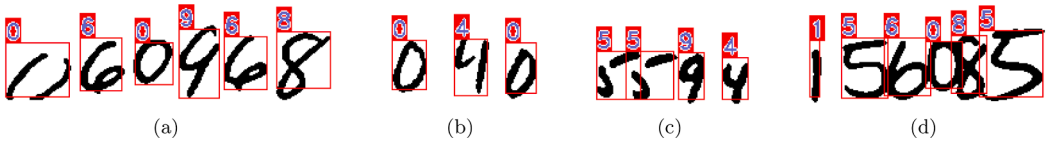


Fig. 17. Correct detection: (a) 060968, (b) 040, (c) 5594, and (d) 156085.

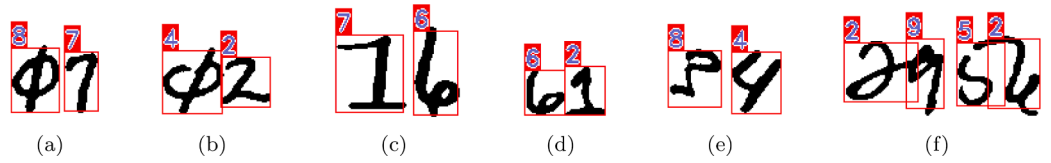


Fig. 18. Misclassification (a) 07 recognized as 87, (b) 02 recognized as 42, (c) 16 recognized as 76, (d) 61 recognized as 62, (e) 34 recognized as 84, and (f) 2956 recognized as 2952.

proposed several networks between 34 and 152 layers, and which achieved outstanding performance on the benchmark datasets. *The ResNet-50 provides* a good tradeoff between speed and accuracy and it is the backbone for the RetinaNet framework. Its architecture is detailed in Table 1. Moreover, an FPN with levels ranging from P_3 to P_7, produces rich and multi-scale features from a single input resolution.

The default dimensions of anchor boxes were defined by authors using samples of the Imagenet Dataset, composed of 1000 classes of real-life objects. Although the dataset includes a wide range of classes, to make anchors feasible for digits, we performed a *k*-means clustering over 10,000 ground-truth bounding boxes from the training samples. This resulted in three anchors with the following aspect ratios: 0.5, 0.6 and 1.0.

4.3. Sequence-to-sequence approach

A Convolutional Recurrent Neural Networks (CRNN) (Voigtlaender et al., 2016; Shi et al., 2017; Dutta et al., 2018) is a sequence-to-sequence model that can be trained from end-to-end. The pipeline for a such network in Fig. 5a. First, convolutional layers extract features from an input image, and then a sequence of feature vectors is extracted from feature maps.

Since each region of the feature map is associated with a receptive field in the input image, each vector in the sequence is a descriptor of this image field, as illustrated in Fig. 5b. Next, this sequence fed the recurrent layers, which are composed of a bidirectional Long-Short Term Memory (LSTM) (Schuster & Paliwal, 1997) network, producing a per-frame prediction from left to right of the image. Finally, the transcription layer determines the correct sequence of classes to the input image by removing the repeated adjacent labels and the blanks, represented by the character '-'. This solution is well suited when the past and future context of a sequence contribute to the recognition of the whole input. With the aid of contextual information, such as a lexicon, this approach achieves high text recognition performance. The application of this solution to handwritten digits is a matter of discussion once we have fewer classes than words (0..9), but there is no lexicon to mitigate possible confusion.

4.3.1. Network architecture

Shi et al. (2017) proposed the CRNN architecture to recognize English words. To produce feature maps with a larger width, they adopted 1×2 size max-pooling on layers #7 and #12 instead of squared ones (see Table 2). The input resolution is defined as 32×128 (*height* $\times$ *width*). We kept the network architecture unchanged where we want to

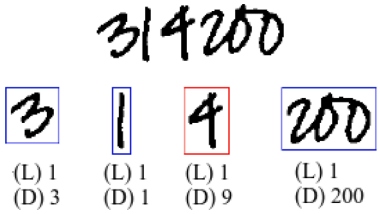


Fig. 19. Missed prediction of Hochuli et al. (2018): 314200 as 319200. The classifier (L) correctly predicted the length of components, however, the 1-digit classifier (D) confused the number '4' as '9'.

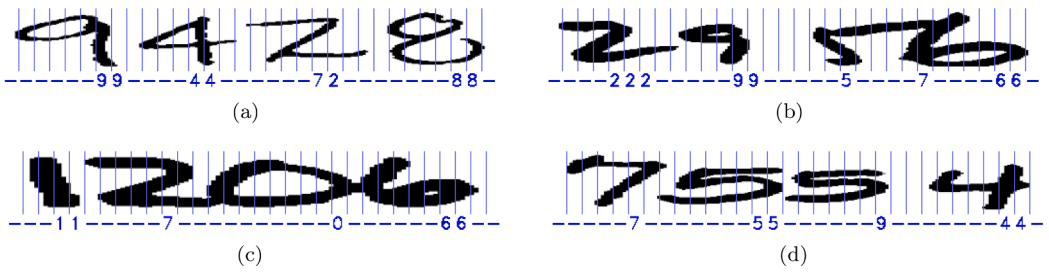


Fig. 20. Missed predictions of CRNN for NIST dataset: (a) '9428' as '94728' and (b) '2956' as '29576' representing over-segmentation errors (length), and (c) '1206' as '1706' and (d) '7554' as '7594' representing misclassification.

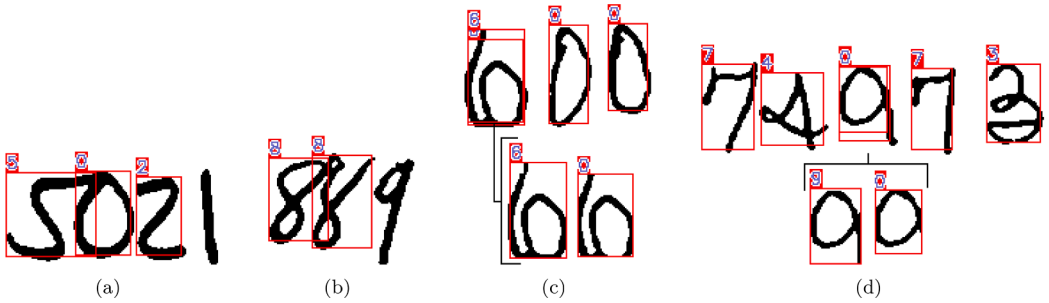


Fig. 21. Missed detections of RetinaNet for NIST dataset: (a) '5021' as '502' , (b) '889' as '88', (c) '600' as '6000' and (d) '74973' as '749073'.

evaluate handwritten digit recognition performance.

4.4. Training

Since deep networks require a considerable amount of data to learn a representation, we created a synthetic dataset composed of numerical strings ranging from 2- to 6-digits, and containing isolated and touching components. The rationale for this strategy was to create a dataset with contextual information about the neighborhood of isolated and touching digits. The strings are built by concatenating isolated digits of NIST SD19 (Grother, 2016) through the algorithm described in Ribas et al. (2013). Fig. 6 shows some samples.

To avoid building a biased dataset, we used information on authors available on the NIST SD19, which ensure that digits from different authors were used exclusively for training, validation, and testing. Table 3 shows the purpose (training, validation, and testing), as well as the amount of data created.[6]

Another aspect we took into consideration when creating this dataset was the distribution of isolated and touching digits in the strings. When analyzing real datasets, one may observe something similar to an exponential distribution dominated by isolated digits. Fig. 7a shows such a distribution while 7b depicts the distribution of the 10 classes of digits in the database. The digit "1" is less represented since it is the class with less occurrence in touching strings (Ribas et al., 2013).

The models detailed in Sections 4.1–4.3 were trained from scratch using the synthetic data described in Table 3. Except by input size, training is performed with the Stochastic Gradient Descent (SGD) using back-propagation with mini-batches of 64 instances, a momentum factor of 0.9, and a weight decay of 5×10^{-4}. Initially, the learning rate is set to 10^{-3}, to allow the weights to quickly fit the long ravines in the weight space, after which it is reduced over time (until 5×10^{-4}) to make the weights fit the sharp curvatures.

In the present work, regularization was implemented through early-stopping, which prevents overfitting from interrupting the training procedure once the performance of the network on a validation set deteriorates. During training, the network's performance on the training set will continue to improve, but its performance on the validation set will only improve up to a certain point, where the network starts to overfit the training data. At that point, the learning algorithm is terminated. The models were trained using an NVidia GeForce Titan X GPU.[7]

4.4.1. Time consuming

Table 4 presents the average time consumed by each approach in terms of recognition. Since training is not often used, the impact of the time consumed for this task is not considered in this evaluation.

In light of this, we can observe that the number of objects (digits) that composing a string does not contribute to a significant increase in the recognition time for all approaches. The reason for this is that the network forward has a similar cost irrespective of the number of objects in the input. It is worth mentioning that the time analysis for Aly and Mohamed (2019) is not reported once the code is not released.

5. Experiments

We designed a set of experiments on five different benchmarks to allow a better comparison of the different approaches. Firstly, we used the challenging Touching Pairs (TP) dataset (Section 5.1), which contains different touching pairs styles. Then, we focus on the Synthetic Touching Strings dataset (Section 5.2) to evaluate the limits of each approach in a hard task, i.e., one using strings with up to four touching digits. The third dataset (Section 5.3) is a well-known NIST-SD19 composed of 11,585 strings ranging from 2 to 6 digits. The fourth benchmark was built for the ICFHR 2014 HDSR challenge (Section 5.4), which contains two different datasets. Finally, we present an experiment with very long strings to emphasize the power of the object-detection approach.

[6] All the synthetic data is available upon request for research purposes at https://web.inf.ufpr.br/vri/databases-software/touching-digits/.

[7] All trained classifiers are available for research purposes at https://web.inf.ufpr.br/vri/databases-software/touching-digits/.

Table 8
Comparison of the recognition rates on NIST SD19.

Length	Samples	RetinaNet	CRNN	Britto-Jr et al. (2003)	Oliveira et al. (2002)	Oliveira and Sabourin (2004)	Sadri et al. (2007)	*Sadri et al. (2007)	Gattal et al. (2017)	Hochuli et al. (2018)	Aly and Mohamed (2019)	Yolo	Samples	Liu et al. (2004)	Ciresan (2008)
2	2370	85.3	70.3	94.8	96.8	97.6	95.5	98.9	99.0	97.6	98.8	98.6			
3	2385	81.5	84.4	91.6	95.3	96.2	91.4	97.2	97.3	96.2	96.4	97.6	1476	96.8	93.4
4	2345	75.7	86.8	91.3	93.3	94.2	91.0	96.1	96.5	94.6	95.0	97.1			
5	2316	68.5	83.8	88.3	92.4	94.0	88.0	95.8	95.9	94.1	95.4	96.5			
6	2169	65.7	76.3	89.0	93.1	93.8	88.6	96.1	96.6	93.3	95.0	95.8	1471	96.7	
Average		75.3	80.3	91.0	94.2	95.2	90.9	96.8	97.1	95.2	96.1	97.1		96.7	93.4

5.1. TP dataset

The TP dataset contains 79,464 samples of touching digits and it was proposed in Ribas et al. (2013) as a benchmark for segmentation algorithms. The authors were interested in evaluating when the segmentation cuts may produce a correct segmentation no matter how many cuts were produced. The solution in these situations is straightforward for approaches that produce only one cut: if the resulting components (after classification) match the ground-truth, the segmentation is deemed correct. However, for approaches that produce multiple cuts, the segmentation is only deemed correct, if there are at least two correct digits among hypotheses.

For this experiment, we assume a correct segmentation when the model provides the correct number of digits/objects and classes. Otherwise, there is an error. Two sources of errors are possible: a wrong estimation of the string length or its misclassification. Table 5 compares the results of the end-to-end approaches with both segmentation-based and segmentation-free algorithms. It should be mentioned that all the works presented in Table 5 use the same testing set proposed in Ribas et al. (2013). The training sets for both the segmentation-based and the segmentation-free algorithms used isolated digits extracted from NIST SD19. However, they differ in that all segmentation-based approaches use isolated digits to train single-digit classifiers while the segmentation-free ones use the strings of digits described in Table 3. Table 5 also illustrates the performance according to the connection types depicted in Fig. 8.

5.1.1. Discussion

Algorithms based on a single segmentation hypothesis (segmentation cuts = 1) usually fail in more complex touching cases (e.g., type V) since just one segmentation cut is often not enough to correctly split the digits. On the other hand, algorithms based on multiple cuts, such as Chen and Wang (2000) and Gattal and Chibani (2015), find the correct segmentation but at a high computational cost, which makes them impractical for real applications.

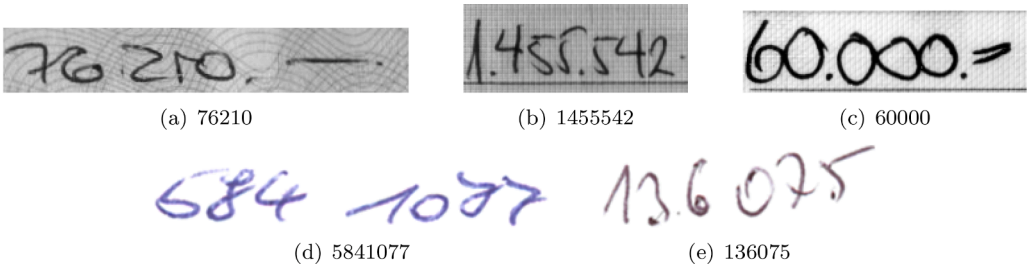
Yolo compares to Hochuli et al. (2018) in terms of classification for most types of connections depicted in Fig. 8, except on Type V. In this case, the task-specific classifier trained on touching pairs performs better since it can cope with highly slanted images better. This is related to the limitations of Yolo, as reported by Redmon et al. (2016). Yolo imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that the model can predict. In our case, we observed this phenomenon in Fig. 9d.

CRNN and RetinaNet, on the other hand, performed quite poorly with performances even worse than those of several segmentation-based algorithms. One of the bottlenecks of the CRNN is that the local perspective of the problem given by each receptive field, or by a subsequence, may represent a digit fragment. In this case, a fragment of a digit taken out of context can be easily misclassified with high probability when its shape is somewhat similar to that of a digit. This issue is quite similar to the over-segmentation strategy implemented by segmentation-based approaches. Considering that there is no lexicon or post-processing method, the transcription layer may collapse by missed predictions. The worst performance is seen in complex cases, i.e., type V, where the neighborhood of digits is severely affected because it has more overlapping than other types. In analyzing the errors, we observe that most of these complex cases could be solved using contextual information, which, unfortunately, is not available in most applications of HDSR. These cases are depicted in Fig. 10.

RetinaNet also fails to efficiently encode the neighborhood of digits, which explains the model collapse on hard overlapped digits (Type V). It should however, be noted that it performs well in easy cases, such as Type III. Moreover, pairs featuring the digit “1” produce more missed detections if their aspect ratio are significantly different from those of the other classes. Fig. 11 illustrates some of these problems.

Table 9
Distribution of Orand-Car and CVL datasets.

	Car-A			Car-B			CVL		
Length	Train	Val	Test	Train	Val	Test	Train	Val	Test
2	17	5	36	0	0	0	0	0	0
3	176	28	387	0	0	0	0	0	0
4	633	71	1425	60	3	5	0	0	0
5	819	84	1475	1080	120	69	113	12	789
6	127	18	363	1432	167	1241	683	75	4144
7	27	2	87	127	10	1452	340	39	1765
8	1	1	11	1	0	157	0	0	0
9	0	0	0	0	0	2	0	0	0
10	0	0	0	0	0	0	0	0	0
Total	1800	209	3784	2700	300	2926	1136	126	6698



(a) 76210 (b) 1455542 (c) 60000

(d) 5841077 (e) 136075

Fig. 22. Sample data of (a) Car-A, (b) and (c) are samples of Car-B and (d) and (e) are samples of CVL dataset.

5.2. Touching strings dataset

This goal of this experiment is to illustrate the limits of the evaluated approaches when dealing with a challenging task, i.e., tasks involving strings with up to four touching digits (e.g., Fig. 12a). As pointed out earlier, this is not very often observed in real databases, but it is useful for assessing the limits of the proposed strategies discussed in this work. An important point here is that, as we can observe in Fig. 12b, the shape of the digits may be severely affected the neighbors, which is quite different from those observed in the isolated digit datasets especially those in the middle of the string. This is why learning from strings rather than from isolated digits is important, particularly for approaches that use contextual information into the learning process.

In this experiment, 570,000 images of isolated digits, 2-, 3-, and 4-touching digits described in Hochuli et al. (2018) were used. The accuracy of all the strategies employed and the average recognition time are reported in Table 6. As stated in Section 4.4.1, a more in-depth analysis of Aly and Mohamed (2019)'s approach is not reported as the code was not released.

5.2.1. Discussion

As can be observed, the best overall results were achieved by Yolo followed by the approach proposed in Hochuli et al. (2018). Yolo's main advantage is that it has no constraints regarding the number of touching digits in the string.

Regarding the CRNN, the design of its architecture imposes few constraints over its performance on digit strings. Since its input size is fixed, a shorter string has its aspect ratio stretched, which has more probability of suffering from over-segmentation. Fig. 13d illustrates a missed prediction of digit '2' as a fragment of digit '4'. In such a case, taking the representation contained in the receptive fields, out of context, could reasonably leads to a digit '2' being composed. An extrapolation is possible to the missed predictions of digit '1' in Fig. 13c and e. Furthermore, in Fig. 13, we can observe the impact of the aspect ratio and aforementioned over-segmentation.

RetinaNet suffers when encoding the neighborhood of digit. In Fig. 14a and b the aspect ratio of digits '1' and '2' are quite different from that of the neighborhood, which then results in a misclassification. In Fig. 14d, a segment touching misleads the network in the detection of a digit '4'. Moreover, the multi-scale strategy can magnify a fragment that can be confused with a digit. In such a case, the number '9' was recognized as '0', which is quite similar to over-segmentation. Fig. 14c illustrates the problem. The Yolo approach (Fig. 15) successfully overcomes these issues.

5.3. NIST SD19 strings

Experiments using real-world strings are based on 11,585 numeral strings extracted from the hsf_7 series and distributed into five classes: 2_digit (2,370), 3_digit (2,385) 4_digit (2,345), 5_digit (2,316), and 6_digit (2,169) strings, respectively. The strings were cropped from original samples leaving a border of 5 pixels. These data exhibit different problems, such as touching and fragmentation, and were also used as test sets in Hochuli et al. (2018), Oliveira et al. (2002), Britto-Jr et al. (2003), Liu, Sako, and Fujisawa (2004), Oliveira and Sabourin (2004), Sadri, Suen, and Bui (2007) and Gattal et al. (2017). It is important to mention that hsf_7 was never used for training.

5.3.1. Discussion

To better compare the approaches, we divided the errors into two classes: misdetection and misclassification. Table 7 summarizes the results for this experiment for the approaches.

The Yolo error analysis shows that most detection problems are related to the digit "1". The problem occurs when (i) the height of the image is too small (Fig. 16a), (ii) is too high (Fig. 16b) or (iii) the slant of the image is big (Fig. 16c). In these cases, the digit "1" is not detected. Another source of error is the digit "4" (very often related to the digit "1"). In these cases, the model sometimes detects two objects ("4" and "1") in the digit "4" (Fig. 16d) and sometimes just the digit "4" is detected, missing the digit "1" (Fig. 16e). Finally, we observed a few samples behaving similarly to under-segmentation (Fig. 16f) and over-segmentation (Fig. 16g).

It is worth mentioning that the average misdetection rate was below 1%, and most of the cases featuring broken digits (Figs. 17a–c) and

Table 10
Comparison of the recognition rates on Orand and CVL datasets (ICFHR 2014 Competition).

Methods	CAR-A	CAR-B	CVL
Tebessa I★	37.05	26.62	59.30
Tebessa II★	39.72	27.72	61.23
Hochuli et al. (2018)	50.10	40.20	66.10
Singapore★	52.30	59.30	50.40
RetinaNet	72.51	69.17	61.06
Pernanbuco★	78.30	75.43	58.60
Beijing★	80.73	70.13	85.29
CRNN*	88.01	89.79	26.01
Saabni (2016)★,†	85.80		–
Zhan et al. (2017)	89.75	91.14	2707
Xu et al. (2018)	91.89	93.79	63.03
Yolo	96.20	96.80	84.20

★ Algorithms reported in Diem et al. (2014).
† Unified CAR-A and CAR-B datasets.
* Reported by Zhan et al. (2017).

densely connected strings (Fig. 17d), where other approaches show their limitations, were successfully recognized by the Yolo.

Table 7 shows an average error rate of 2.4%, in which most misclassifications is related to handwriting variability. Fig. 18 shows some common mistakes involving classes "0" and "1". In these cases, the handwriting styles are poorly represented in the training set (see Fig. 19).

In the method based on dynamic selection (Hochuli et al., 2018), misclassification is the primary source of error, with 1.0% due to length classifier and 2.9% to digit classifiers. Since most of the connected components in the NIST SD19 strings are composed of isolated digits, the 1-digit classifier is responsible for most of the connected components classification.

The detection errors of the CRNN reported in Table 7 occur both in isolated digits (Fig. 20a) and in the touching digits (Fig. 20b). As mentioned in Section 4.3, the aspect ratios of shorter and longer strings are deformed by a fixed input size, which explains the highest error rate for 2- and 6-digit strings. Performance was severely impacted by misclassification into all string sizes. Since the handwriting was highly variable, CRNN did not generalize the representation. This issue is depicted in Fig. 20c and d, where the digits '2' and '5' were missed.

Finally, Table 7 shows that the bottleneck of RetinaNet is detection, as it either misdetects or overdetects digits. The former is related to the shape of digit, while the latter is caused by a multi-scale technique which allows a fragment of a digit to be magnified to a scale that represents a digit, with high accuracy. This issue is similar to over-segmentation. The aforementioned issues are depicted in Fig. 21.

Table 8 compares the recognition rates of several systems reported in the literature on NIST-SD19. For completeness, we replicate the results compiled by Hochuli et al. (2018). The works by Britto-Jr et al. (2003), Oliveira et al. (2002) and Oliveira and Sabourin (2004) use different segmentation (implicit and explicit) and classification strategies, such as Hidden Markov Models, Multi-layer Perceptrons and Support Vector Machines. Except for Liu et al. (2004) and Ciresan (2008), all the works use the same strings for testing. Regarding the training data, all of them used isolated digits from NIST SD19. However, the number of digits and how they are used may vary according to the strategy used in each system. In the case of the Yolo, RetinaNet, CRNN, and Hochuli et al. (2018) approaches, the classifiers were trained with the synthetical strings reported in Table 3, which were built by combining the same isolated digits.

The work presented by Sadri et al. (2007) is reported in two columns. The authors proposed a system based on over-segmentation, in which they used a genetic algorithm to optimize their segmentation algorithm. As pointed out in Hochuli et al. (2018), the second set of experiments (marked with an * in Table 8) is somehow biased since the heuristics were defined using a subset of the testing set. Gattal et al. (2017) also reported good performance, but evaluating their results is ccomplicated by the fact that several thresholds used for segmentation appear to be adjusted on the testing set.

Finally, a straightforward comparison is possible with the segmentation-free methods proposed in Hochuli et al. (2018) and recently improved by Aly and Mohamed (2019), which implemented a different fusion strategy, even while, keeping the pre-processing steps and specific-task classifiers. As discussed in Section 4, the end-to-end approaches cuts off all the heuristics used for pre-processing, the need to train several deep learning models, and the parameter used in the fusion strategy. Additionally, Yolo improves the average recognition rate.

Table 11
Distribution of CVL dataset in terms of string labels variability.

Dataset	Samples	# of Different String Labels
Train	1136	10
Test	6698	26

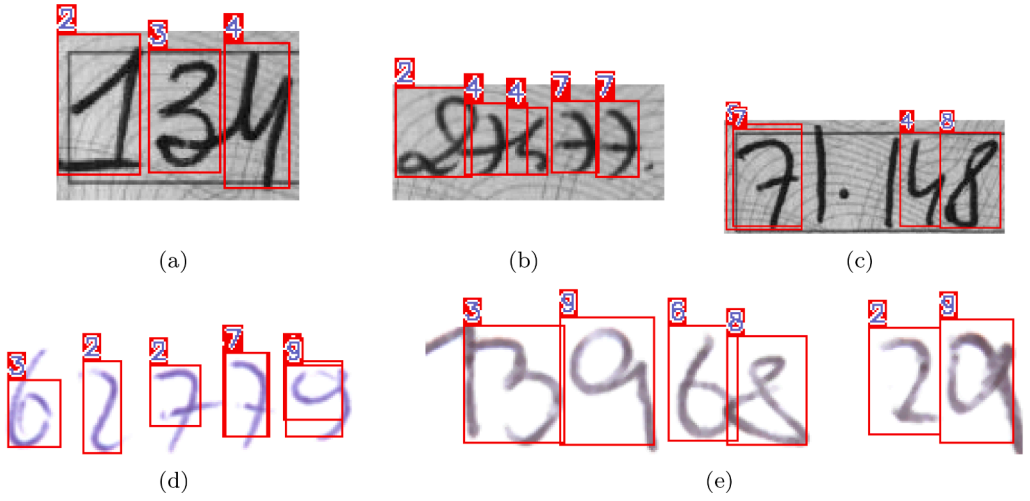


Fig. 23. Missed predictions of RetinaNet for ORAND/CVL dataset: (a) '134' as '234' , (b) '27477' as '24477', (c) '71148' as '9748', (d) '1800000' as '800000', (e) '62779' as '32279' and (f) '1396829' as '396829'.

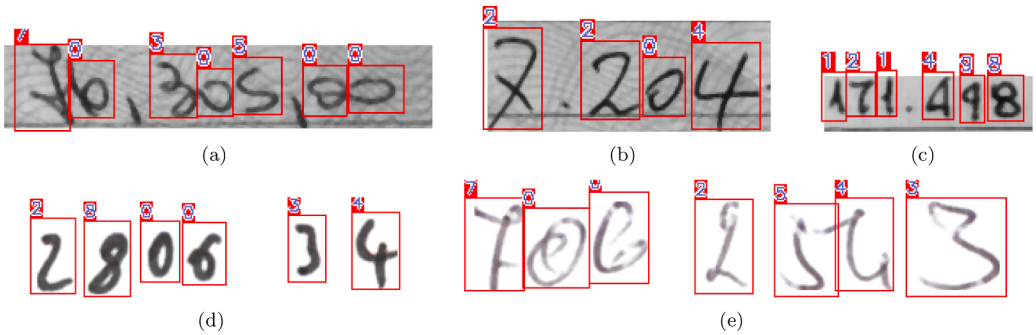


Fig. 24. Missed predictions of Yolo for ORAND/CVL dataset: (a) '7630500' as '7030500' , (b) '7204' as '2204', (c) '171448' as '121498', (d) '280634' as '280034 ' and (e) '7062543' as '7002543'.

Table 12
Image input size that maximizes the recognition rate for each string length.

String Length	Average String Width (S_w)	Input Image Size (Π_w) ($128 \times w$)	Recognition Rate (%)
2	75	128	98.6
4	150	256	97.6
6	228	384	97.6
8	306	512	96.4
10	381	640	94.8
12	448	768	94.2
14	524	896	91.0
16	596	1024	90.6
18	666	1152	88.8
20	750	1280	89.6

5.4. ICFHR datasets

The experiment in this case performed on two real-world datasets built for the ICFHR 2014 challenge on HDSR (Diem et al., 2014).

The ORAND-CAR-2014 consists of digit strings of the courtesy amount recognition (CAR) field extracted from real bank checks with a resolution of 200 dpi. Besides the traditional challenges present in handwriting such as noise, broken digits, and touching, this dataset presents samples with background and currency symbols such as '#', '$', dots, commas, and dashes. The CVL Database was collected mostly amongst students of the Vienna University of Technology, and contains about 300 writers, female and male alike. The images are delivered with RGB information and at a resolution of 300 dpi. It includes varying sizes and writing styles. This database poses new challenges to the community since it is harder than previously published datasets, especially in terms of variance in writing style. Table 9 shows the amount of data used for training and testing in both datasets. Some samples are depicted in Fig. 22.

Whenever the handwriting styles of these datasets are different from those of NIST SD19, models already trained using synthetic data provide unreliable results, since the encoded information is quite different. We thus trained all models using the data described in Table 9, since it is the protocol suggested in the ICFHR 2014 competition. We kept the training parameters unchanged, following that described in Section 4.4. To provide sufficient information to the object-detection approach, we annotated the digits bounding-boxes (ground-truths) of each training sample.[8] This laborious task was necessary since most of the samples have a complex background, noise, and symbols, which are difficult to reproduce synthetically.

[8] The annotated dataset is available upon request for research purposes at https://web.inf.ufpr.br/vri/databases-software/touching-digits/.

5.4.1. Discussion

Table 10 presents the performances of end-to-end approaches on the testing set. The performances of all methods are reported on the same testing datasets (Table 9), which were proposed in the ICFHR 2014 challenge. Zhan et al. (2017) previously implemented the CRNN approach to these datasets; and we therefore, we just replicated the results. The worst results were found on the CVL dataset (26.01%). Besides the unbalanced distribution between the training and testing sets, a short variety of string labels in the training set (only 10) do not provide an efficient representation of digit iterations into a sequence. For example, the sequence pair "98", which is not available in the training set, is found in two different strings of the testing set ("120398", "662498"). Table 11 shows the poor variation of labels. Since these end-to-end models must learn the variability introduced by the neighborhood, this lack of samples strongly penalizes such models. Unlike in the other benchmarks, in which the dynamic selection approach (Hochuli et al., 2018) performed quite well, it struggled in these experiments, mostly because of its heuristic-based pre-processing module. Since ORAND-CAR provides a hard background and currency symbols, the pre-processing module collapsed when detecting connected components. It performed slightly better on the CVL dataset, which has no significant challenges in background suppression. However, the poor distribution of the training set penalized the performance of the specific-task classifiers.

The Yolo and RetinaNet object-based models achieve a performance close to those reported in Section 5.3, which denote that the network could encode a hard background. A remarkable performance was achieved by the Yolo, point to the robustness of the model in encoding context, noise, and background. The ORAND/CVL dataset also faced challenges in the form of overlapping digits, handwriting variability, and different aspect ratios that severely impact the models performances. These issues are illustrated in Figs. 23 and 24.

Finally, the main drawback of object-based approaches is the laborious task of data annotation when synthetic samples are not applicable.

5.5. Very large strings

The results show that approaching the HDSR as an object detection/recognition problem is absolutely feasible. Additionally, it produced (with Yolo) the most consistent performance for all the benchmarks used in this study. In this final experiment, our goal is to assess the Yolo on very large strings.

As mentioned previously, the images were resized to 128×256 ($height \times width$) for training. However, since Yolo changes the input size after every few iterations during training, this network can recognize testing images of different sizes. The question is how to properly resize the testing input image to maximize the network's performance. This is relevant since the image width may vary considerably according to the number of digits in the string. A 20-digit string is significantly longer than a 2-digit string, for example. Resizing both of them to 128×256 is

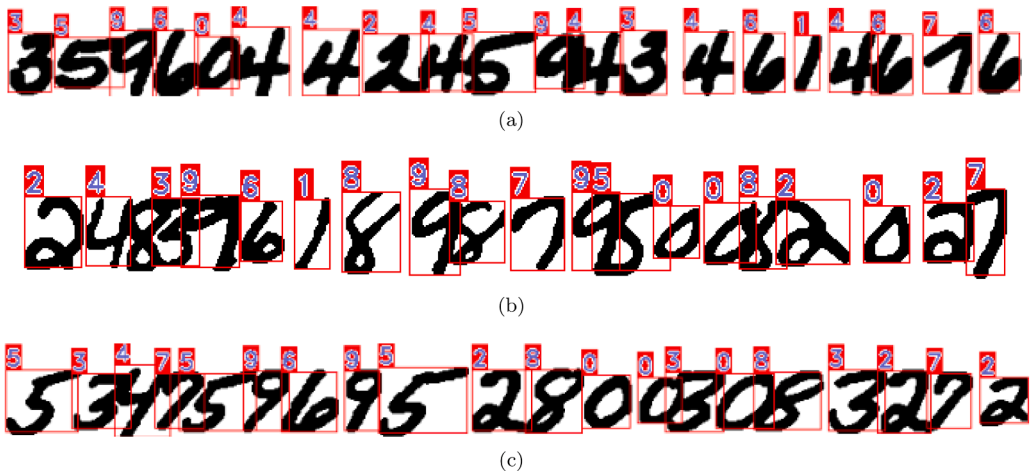


Fig. 25. 20-digit strings correctly recognized by the Yolo-based approach.

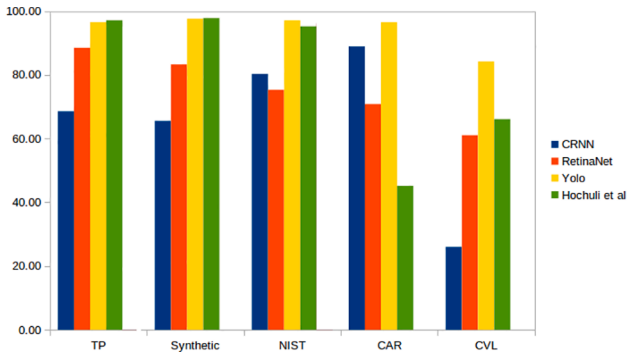


Fig. 26. Average recognition of approaches per dataset.

not the right choice.

To address this, we experimented on 5,000 strings ranging from 2 to 20 digits, which were synthetically created by concatenating isolated digits from NIST SD19. For each string length, we tested the input image width in the following range: [128, 256, 384, 512, 640, 768, 896, 1024, 1152,1280]. The image height was always 128. Table 12 summarizes the image input size that maximizes the recognition rate for each string length.

5.5.1. Discussion

From Table 12, we can notice that there is a quasi-linear relation between the average string width of the testing images[9] and the best input size for the Yolo. In light of this, we propose a rule (Eq. (6)) to compute the input size width of the Yolo based on the width of the testing image. Such a rule is used for all experiments reported in this paper.

$$II_w = \begin{cases} 128 & \text{for } S_w \leqslant 75 \\ S_w \times 1.70 & \text{otherwise} \end{cases} \quad (6)$$

Fig. 25 shows some examples of 20-digit strings recognized by the system using the rule above. These corroborate the efficiency of the adopted resizing strategy and show that the approach can perform well even for very long strings composed of broken, overlapping, and different configurations of touching digits.

[9] The number of pixels may vary depending on the image resolution. In this work, all the images were acquired in 300dpi.

5.6. Summary of the experiments

Fig. 26 summarizes the performance of the assessed methods on the different datasets used in this study. As we can see, Yolo achieved outstanding performance in all scenarios. However, its bottleneck is the ground-truth annotation when synthetic samples are not feasible.

Even though RetinaNet also implements an object detection approach, it suffers from the built-in multi-scale strategy (FPN), once a magnified fragment of digit misleads the model. A similar issue occurs with CRNN in which the various different receptive fields fragment the input. These issues are close to the over-segmentation problem faced by segmentation-based algorithms.

Finally, the segmentation-free approach of Hochuli et al. (2018) perform well in scenarios where there is no hard background, but, suffer from handling a complex pipeline composed of heuristic process and multiple classifiers. We did not add the Aly and Mohamed (2019) method in this comparison because we had no access to its source code.

6. Conclusion

This paper described end-to-end solutions for HDSR in which the string of digits is assumed to be composed of objects that can be automatically detected and recognized. To this end, several strategies were evaluated.

A robust experimental protocol based on numeral string datasets was defined to validate the proposed methods containing several types of noise, touching digits, fragmentation, complexes backgrounds, and long strings. The experimental results show that the object-detection approach is a feasible end-to-end solution that compares favorably to the state-of-the-art in HDSR in terms of recognition rates. Also, it considerably reduces the complexity of the string recognition task and avoiding heuristic-based methods, special pre-processing, segmentation, and classifiers devoted to specific-length strings, meaning, no constraints related to the string length exist. However, the difficulty posed by need for data annotation when synthetic samples are not applicable is the main drawback of this approach.

Conversely, the sequence-to-sequence strategy provides a short pipeline. No significant efforts related to the annotation of ground-truth is needed, as in the case with the object-detection based approach. However, the strategy depends on contextual information, such as a lexicon, to achieve good results. Thus, its design for handwritten digits needed to be reviewed.

CRediT authorship contribution statement

Andre G. Hochuli: Conceptualization, Methodology, Software,

Validation, Investigation, Writing - original draft. **Alceu S. Britto Jr:** Conceptualization, Supervision, Writing - review & editing, Funding acquisition, Resources. **David A. Saji:** Software, Investigation. **José M. Saavedra:** Supervision, Writing - review & editing, Funding acquisition. **Robert Sabourin:** Supervision, Writing - review & editing. **Luiz S. Oliveira:** Conceptualization, Supervision, Writing - review & editing, Funding acquisition, Resources.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by The National Council for Scientific and Technological Development (CNPq) grants 303252/2018-9 and 306684/2018-2, CAPES (PhD scholarship - Finance Code 001), Fondecyt Chile (project number 11150945), STIC-Amsud 19-STIC-04 and Araucária Foundation. In addition, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan XP GPU used for this research.

References

Aly, S., & Mohamed, A. (2019). Unknown-length handwritten numeral string recognition using cascade of pca-svmnet classifiers. *IEEE Access, 7*, 52024–52034. https://doi.org/10.1109/ACCESS.2019.2911851

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*, 1798–1828.

Britto-Jr, A., Sabourin, R., Bortolozzi, F., & Suen, C. Y. (2003). The recognition of handwritten numeral strings using a two-stage HMM-based method. *International Journal on Document Analysis and Recognition, 5*, 102–117.

Britto, A. S., Sabourin, R., & Oliveira, L. S. (2014). Dynamic selection of classifiers—a comprehensive review. *Pattern Recognition, 47*, 3665–3680.

Casey, R., & Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. *IEEE Transactions on PAMI, 18*, 690–706.

Chen, Y. K., & Wang, J. F. (2000). Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*, 1304–1317.

Choi, S., & Oh, I. (1999). A segmentation-free recognition of two touching numerals using neural networks. In *Proc. of 5th international conference on document analysis and recognition, Bangalore, India* (pp. 253–256).

Ciresan, D. (2008). Avoiding segmentation in multi-digit numeral string recognition by combining single and two-digit classifiers trained without negative examples. In *10th International symposium on symbolic and numeric algorithms for scientific computing* (pp. 225–230).

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3642–3649). https://doi.org/10.1109/CVPR.2012.6248110

Congedo, G., Dimauro, G., Impedovo, S., & Pirlo, G. (1995). Segmentation of numeric strings. In *3rd International conference on document analysis and recognition* (pp. 1038–1041).

Cruz, R. M., Sabourin, R., & Cavalcanti, G. D. (2018). Dynamic classifier selection: Recent advances and perspectives. *Information Fusion, 41*, 195–216.

Das, R. S. N., Saha, K. A., & Nasipuri, M. (2016). A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition. *Pattern Recognition, 58*, 172–189.

Diem, M., Fiel, S., Kleber, F., Sablatnig, R., Saavedra, J. M., Contreras, D., Barrios, J. M., & Oliveira, L. S. (2014). Icfhr 2014 competition on handwritten digit string recognition in challenging datasets (hdsrc 2014). In *2014 14th International conference on frontiers in handwriting recognition* (pp. 779–784). IEEE.

Divvala, S. K., Hoiem, D., Hays, J. H., Efros, A. A., & Hebert, M. (2009). An empirical study of context in object detection. In *IEEE conference on computer vision and pattern recognition, 2009. CVPR 2009* (pp. 1271–1278). IEEE.

Dutta, K., Krishnan, P., Mathew, M., & Jawahar, C. V. (2018). Improving cnn-rnn hybrid networks for handwriting recognition. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)* (pp. 80–85). https://doi.org/10.1109/ICFHR-2018.2018.00023

Elms, A. J., Procter, S., & Illingworth, J. (1998). The advantage of using an hmm-based approach for faxed word recognition. *International Journal of Document Analysis and Recognition*, 18–36.

Elnagar, A., & Alhajj, R. (2003). Segmentation of connected handwritten numeral strings. *Pattern Recognition, 36*, 625–634.

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *IEEE conference on computer vision and pattern recognition, 2008. CVPR 2008* (pp. 1–8). IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*, 1627–1645.

Fenrich, R., & Krishnamoorthy, S. (1990). Segmenting diverse quality handwritten digit strings in near real-time. In *5th USPS advanced technology conference* (pp. 523–537).

Fujisawa, H., Nakano, Y., & Kurino, K. (1992). Segmentation methods for character recognition: From segmentation to document structure analysis. *Proceedings of IEEE, 80*, 1079–1092.

Gattal, A., & Chibani, Y. (2015). SVM-based segmentation-verification of handwritten connected digits using the oriented sliding window. *International Journal of Computational Intelligence and Applications, 14*, 1–17.

Gattal, A., Chibani, Y., & Hadjadji, B. (2017). Segmentation and recognition system for unknown-length handwritten digit strings. *Pattern Analysis and Applications, 20*, 307–323.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the 2015 IEEE international conference on computer vision (ICCV), ICCV '15* (pp. 1440–1448). Washington, DC, USA: IEEE Computer Society.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

Grother, P. J. (2016). NIST Special Database 19 – Handprinted forms and characters database. NIST.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Shen, T. (2017). Recent advances in convolutional neural networks. *Pattern Recognition*.

Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2017). Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition*, 163–176.

Han, J., Zhang, D., Cheng, G., Liu, N., & Xu, D. (2018). Advanced deep-learning techniques for salient and category-specific object detection: A survey. *IEEE Signal Processing Magazine, 35*, 84–100. https://doi.org/10.1109/MSP.2017.2749125

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *2017 IEEE international conference on computer vision (ICCV)* (pp. 2980–2988). IEEE.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37*, 1904–1916.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hochuli, A. G., Oliveira, L .S., Souza Britto, A. d., & Sabourin, R. (2018). Segmentation-free approaches for handwritten numeral string recognition. In *2018 International joint conference on neural networks (IJCNN)* (pp. 1–8).

Hochuli, A. G., Oliveira, L. S., Britto, A. S., & Sabourin, R. (2018). Handwritten digit segmentation: Is it still necessary? *Pattern Recognition, 78*, 1–11.

Lacerda, E., & Mello, C. A. B. (2013). Segmentation of connected handwritten digits using self-organizing maps. *Expert Systems with Applications, 40*, 5867–5877.

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE conference on computer vision and pattern recognition, 2008. CVPR 2008* (pp. 1–8). IEEE.

Laroca, R., Barroso, V., Diniz, M. A., Gonçalves, G. R., Schwartz, W. R., & Menotti, D. (2019). Convolutional neural networks for automatic meter reading. *Journal of Electronic Imaging, 28*, 1–14. https://doi.org/10.1117/1.JEI.28.1.013023

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., & Menotti, D. (2018). A robust real-time automatic license plate recognition based on the YOLO detector. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 1–10). https://doi.org/10.1109/IJCNN.2018.8489629

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE, 86*, 2278–2324.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Cham: Springer International Publishing.

Liu, C.-L., Sako, H., & Fujisawa, H. (2004). Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *IEEE Transactios on Pattern Analysis and Machine Intelligence, 26*, 1395–1407.

Matan, O., Burges, J. C., LeCun, Y., & Denker, J. S. (1992). Multi-digit recognition using a space displacement neural network. In J. E. Moody, S. J. Hanson, & R. L. Lippmann (Eds.), *Advances in neural information processing systems* (Vol. 4, pp. 488–495). Morgan Kaufmann.

Oliveira, L. S., Lethelier, E., Bortolozzi, F., & Sabourin, R. (2000). A new approach to segment handwritten digits. In *Proc. of 7th international workshop on frontiers of handwriting recognition, Amsterdam, Netherlands* (pp. 577–582).

Oliveira, L. S., & Sabourin, R. (2004). Support vector machines for handwritten numerical string recognition. In *9th International workshop on frontiers in handwriting recognition* (pp. 39–44).

Oliveira, L. S., Sabourin, R., Bortolozzi, F., & Suen, C. Y. (2002). Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Transactions on Pattern Analysis on Machine Intelligence, 24*, 1438–1454.

Pal, U., Belaid, A., & Choisy, C. (2003). Touching numeral segmentation using water reservoir concept. *Pattern Recognition Letters, 24*, 261–272.

Procter, S., Illingworth, J., & Elms, A. J. (1998). The recognition of handwritten digit strings of unknown length using hidden Markov models. In *Proc. of 14th international conference pattern recognition (ICPR)* (pp. 1515–1517).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Redmon, J., & Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 6517–6525). IEEE.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th international conference on neural information processing systems – Vol. 1 NIPS'15* (pp. 91–99). Cambridge, MA, USA: MIT Press.

Ribas, F. C., Oliveira, L. S., Britto, A. S., & Sabourin, R. (2013). Handwritten digit segmentation: A comparative study. *International Journal on Document Analysis and Recognition, 16*, 567–578.

Roy, P., Bhunia, A., Das, A., Dey, P., & Pal, U. (2016). HMM-based Indic handwritten word recognition using zone segmentation. *Pattern Recognition, 60*, 1057–1075.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV), 115*, 211–252. https://doi.org/10.1007/s11263-015-0816-y

Saabni, R. (2016). Recognizing handwritten single digits and digit strings using deep architecture of neural networks. In *2016 Third international conference on artificial intelligence and pattern recognition (AIPR)* (pp. 1–6). https://doi.org/10.1109/ICAIPR.2016.7585206

Sabour, S., Frosst, N., & Hinton, G. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems*, 30 (NIPS 2017).

Sadri, J., Suen, C. Y., & Bui, T. D. (2007). A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. *Pattern Recognition, 40*, 898–919.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing, 45*, 2673–2681. https://doi.org/10.1109/78.650093

Shi, B., Bai, X., & Yao, C. (2017). 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*, 2298–2304. https://doi.org/10.1109/TPAMI.2016.2646371

Shi, Z., & Govindaraju, V. (1997). Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition, 30*, 1501–1504.

Tamen, Z., Drias, H., & Boughaci, D. (2017). An efficient multiple classifier system for arabic handwritten words recognition. *Pattern Recognition Letters, 93*.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision, 104*, 154–171. https://doi.org/10.1007/s11263-013-0620-5. URL: https://doi.org/10.1007/s11263-013-0620-5.

Vellasques, E., Oliveira, L. S., Britto, A. S., Koerich, A., & Sabourin, R. (2008). Filtering segmentation cuts for digit string recognition. *Pattern Recognition, 41*, 3044–3053.

Voigtlaender, P., Doetsch, P., & Ney, H. (2016). Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th international conference on frontiers in handwriting recognition (ICFHR)* (pp. 228–233). https://doi.org/10.1109/ICFHR.2016.0052

Wang, X., Govindaraju, V., & Srihari, S. N. (2000). Holistic recognition of handwritten character pairs. *Pattern Recognition, 33*, 1967–1973.

Wua, Y., Yin, F., & Liu, C. L. (2017). Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition, 65*, 251–264.

Xiao, X., Jin, L., Yang, Y., Yang, W., Sun, J., & Chang, T. (2017). Building fast and compact convolutional neural networks for offline handwritten Chinese character recognition. *Pattern Recognition*, 72–81.

Xu, X., Zhou, J., & Zhang, H. (2018). Screen-rendered text images recognition using a deep residual network based segmentation-free method. In *2018 24th international conference on pattern recognition (ICPR)* (pp. 2741–2746). https://doi.org/10.1109/ICPR.2018.8545678

Zhan, H., Wang, Q., & Lu, Y. (2017). Handwritten digit string recognition by combination of residual network and rnn-ctc. In D. Liu, S. Xie, Y. Li, D. Zhao, & E.-S. M. El-Alfy (Eds.), *Neural information processing* (pp. 583–591). Cham: Springer International Publishing.

Ziyong, F., Zhaoyang, Y., Shuanping, J. L. H., & Jun, S. (2017). Robust shared feature learning for script and handwritten/machine-printed identification. *Pattern Recognition Letters, 100*.