Segmentation-Free Approaches for Handwritten Numeral String Recognition

Andre G. Hochuli, Luiz S. Oliveira
Federal University of Parana
Department of Informatics (DInf)
Curitiba, PR - Brazil
Email: {aghochuli, lesoliveira}@inf.ufpr.br

Alceu de Souza Britto Jr.
Pontifical Catholic University of Parana
PPGIA
Curitiba, PR - Brazil
Email: alceu@ppgia.pucpr.br

Robert Sabourin
Ecole de Technologie Superieure
Montreal, QC - Canada
Email: robert.sabourin@etsmtl.ca

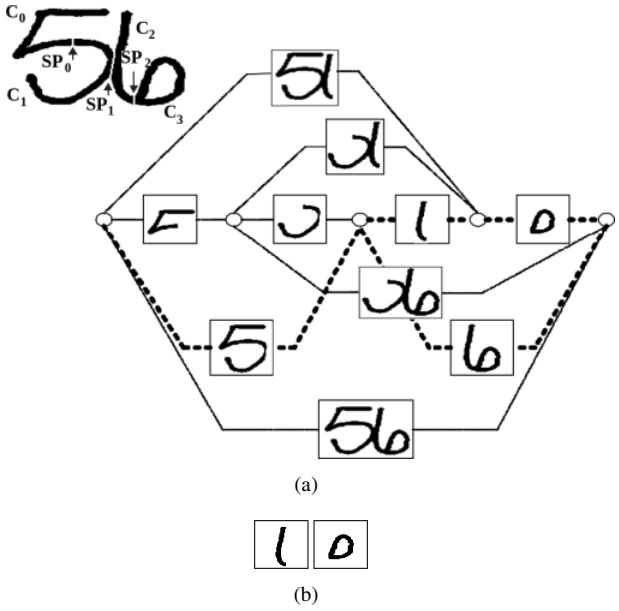
***Abstract*—This paper presents segmentation-free strategies for the recognition of handwritten numeral strings of unknown length. A synthetic dataset of touching numeral strings of sizes 2-, 3- and 4-digits was created to train end-to-end solutions based on Convolutional Neural Networks. A robust experimental protocol is used to show that the proposed segmentation-free methods may reach the state-of-the-art performance without suffering the heavy burden of over-segmentation based methods. In addition, they confirmed the importance of introducing contextual information in the design of end-to-end solutions, such as the proposed length classifier when recognizing numeral strings.**

I. Introduction

The challenge of recognizing numeral strings of unknown length which are not neatly written is still an open problem in the field of document analysis and recognition. The difficulties contributing to the unsatisfactory performance of many methods available in the literature are usually related to the presence of broken, overlapping and touching digits in the string. In such a case, a straightforward solution to segment the string into components representing single digits usually becomes unfeasible.

One may find in the literature a variety of segmentation algorithms that explore different background and foreground features to provide potential segmentation cuts for a given unknown length numeral string. An interesting comparison of different approaches is presented in [1], in which it is possible to observe that an alternative to reduce the heuristics necessary to provide the correct string segmentation cuts is the over-segmentation based algorithms. They segment the string, as many as necessary, into components that may represent digits or part of them. After obtaining the recognition result of each component, or their combination, the algorithms in this approach compute the optimal integrated result. The over-segmentation process for touching numeral '56' is depicted on Figure 1. In fact, the rational behind this approach is to maximize the chances of generating the correct segmentation cuts, however paying the high price of increasing significantly the computational cost of the segmentation/recognition process.

The alternative methods resort to segmentation-free based methods in which the string is recognized without the need of its a priori segmentation into isolated digits. Such an approach has recovered the attention of the research community in the last years with the recent advances in machine learning



(a)

(b)

Figure 1. (a) Segmentation paths for the string "56" and (b) Images that can be easily confused with digits "1" and "0" (extracted from [2]).

motivated by the deep learning techniques. While the over-segmentation based methods demand some specific strategy to generate segmentation cuts, a robust isolated digit recognizer and a strategy for searching the best path among the generated segmentation hypothesis, the segmentation-free demands a significant amount of training data.

One of the first attempts to apply Convolutional Neural Networks (CNN's) to recognize large input fields with unsegmented characters was done by Matan et al. [3]. For a given vector sequence, their SDNN (Spatial Displacement Neural Network) provides a series of output vectors that are post-processed in order to find out the best possible label sequence. The authors observed 66% of correct classification on 3000 images of ZIP Codes. Even being an important contribution, the SDNN did not provide better results than the segmentation methods, as observed by LeCun et al. [4].

Another segmentation-free strategy was presented by Choi and Oh [5]. The authors trained a modular neural network composed of 100 separate subnetworks. They reported a 95.3% recognition rate on 1374 digit pairs extracted from the NIST

dataset. In a similar strategy, Ciresan [6] trained a 100-class CNN using 200,000 images, and reported a 94.65% recognition rate. In addition, he described experiments on 3-digit strings using two CNNs, one for isolated digits and the other for touching pairs. Spite of the fact that this work does not consider three overlapping digits, a 93.4% recognition performance was reported on 1,476 3-digit strings from the NIST dataset.

The aforementioned segmentation-free strategies rests on the assumption that most touching occurs between two adjacent digits. In the same direction, a quite recent approach was proposed by Hochuli et al. [7], which consists of a segmentation-free method based on dynamic selection of classifiers [8][9]. The first one, named $\mathcal{L}$, is applied to estimate the number of components in the string, while other three are responsible to discriminate 10 $[0 \ldots 9]$, 100 $[00 \ldots 99]$, and 1000 $[000 \ldots 999]$ classes. Their approach achieved state-of-art levels, surpassing segmentation based approaches for touching components.

Although Hochuli et al. [7] brings up a new perspective to the problem, towards an end-to-end solution two important questions are still open: (a) Could a single classifier, capable to discriminate those 1110 classes, surpass the proposed dynamic selection strategy based on four classifiers? and, in this case, (b) May the string length classifier remain useful in a single based classifier method?

To answer those questions we have implemented end-to-end solutions composed of a 1110 digit classifier combined with the string length classifier. In order to assess those approaches, we used a robust experimental protocol on a Touching Pairs (TP) dataset of 79,464 touching digits, as well as on Synthetic Dataset composed of 570,461 samples of isolated digits and touching strings of 2- and 3-digit. We observed that the evaluated segmentation-free approaches can achieves state-of-the-art performance without suffering the heavy burden of segmentation. In addition, the experiments confirm the importance of the Length classifier when recognizing strings of digits of unknown length. The information related to the number of digits in the string allow us to introduce some context to the problem, solving different confusions between isolated digits and touching components.

The remainder of this paper is organized as follows: Section II presents the process used to create a synthetic dataset necessary to train our models on touching numeral strings with 2-, 3-, and 4-digits. Section III describes the proposed segmentation free strategies, while Section IV presents the experiments performed to validate the proposed strategies for numeral string segmentation. Finally, the last section presents our conclusions and perspectives of future work.

II. Synthetic Data

In order to efficiently learn representation from data, we had to rely on a considerable amount of samples. We thus created a synthetic dataset composed of touching numerical strings of sizes 2, 3, and 4. The strings are built by concatenating isolated digits of NIST SD19 [10] through the algorithm described by Ribas et al. in [1]. Figure 2 shows some samples. The SD19 database, which is an update of SD3 and SD7, is provided by the American National Institute of Standards and Technology (NIST). This database contains the full page binary images of 3699 Handwriting Sample Forms (HSFs) and 814,255 segmented hand-printed digits and alphabetic characters from the forms.

To avoid building a biased dataset, we used the information on the authors available on the NIST SD19, such that digits from different authors were used exclusively for training, validation, and testing. Table I shows the purpose (training, validation, and testing), as well as the amount of data created[1]. Isolated digits were extracted from NIST SD19. No data augmentation was necessary since more than 240,000 isolated digits are available in this dataset.

Table I
Distribution of the data used for training and testing the classifiers. Samples are uniformly distributed among the classes.

Length/Classes	Samples	Authors	Purpose
1 (Isolated digits)	197,784	0000-2099	Training
10 classes	23,384	3850-4099	Validation
	23,621	3600-3849	Testing
2-Digit String	161,563	1000-1599	Training
100 classes	53,907	1600-1799	Validation
	55,091	1800-1999	Testing
3-Digit String	1,448,680	1000-1599	Training
1000 classes	484,346	1600-1799	Validation
	491,749	1800-1999	Testing
4-Digit String	100,000	1000-1599	Training
*	20,000	1600-1799	Validation
	20,000	1800-1999	Testing

*Data used to train the Length classifier.

(a) (b) (c)

Figure 2. Synthetic data representing touching numerical strings composed of (a) 2-digit, (b) 3-digit and (c) 4-digit.

III. Segmentation-free Strategies

The framework proposed in [7] is depicted in Figure 3. An image x is first classified by the Length classifier ($\mathcal{L}$) which will assign to it a probability of having 1, 2, 3 or 4 touching digits. The digit classification module comprises three classifiers ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$) designed to discriminate 10 $[0 \ldots 9]$, 100 $[00 \ldots 99]$, and 1000 $[000 \ldots 999]$ classes. The classifiers that will be used for a given image depends on the output of the Length Classifier. According to a fusion rule, more than one digit classifier may be invoked to mitigate any possible confusions.

The fusion rule used in this case considers the Top-2 outputs of $\mathcal{L}$. Let $\mathcal{L}^i(x) = p^i(x)$ be the probability of the input pattern x be composed of $i, (i = 1, 2, 3, 4)$ digits. Let

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

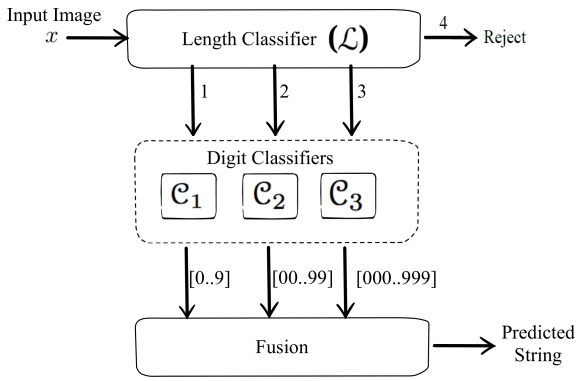


Figure 3. Segmentation-free framework proposed in [7].

$\mathcal{C}_1(x) = \max_{0 \le i \le 9} p^i(x)$, $\mathcal{C}_2(x) = \max_{0 \le i \le 99} p^i(x)$, and $\mathcal{C}_3(x) = \max_{0 \le i \le 999} p^i(x)$ be the probability produced by 10-class, 100-class, and 1000-class classifiers, respectively, for the input pattern x. Let Top1($\mathcal{C}$) and Top2($\mathcal{C}$) be the functions that return the classes with first and second highest scores of a given classifier $\mathcal{C}$, respectively. Then, x is assigned to the class $\omega \in [0...1110]$, according to Equation 1,

$$P(\omega|x) \begin{cases} \text{if } \mathcal{L}_(x) < T, & \max(\mathcal{C}_{Top1(\mathcal{L})}(x), \mathcal{C}_{Top2(\mathcal{L})}(x)) \\ \text{otherwise,} & \mathcal{C}_{Top1(\mathcal{L})}(x) \end{cases} \quad (1)$$

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

The justification for dealing with 1, 2, 3 touching digits is based on the fact that most of touching occurs between two digits and sometimes between three digits [11]. Strings composed of more than three touching digits are very rare in real problems and in the case of occurring $\mathcal{L}$ will reject them.

This dynamic selection strategy has been proved quite efficient surpassing the results reported by all segmentation-based techniques reported in the literature. However, one may argue that an end-to-end solution with just one classifier ($\mathcal{C}_{1110}$) capable of recognizing those 1110 classes (10 isolated, 100 pairs, and 1000 triples), such as the one depicted in Figure 4, is more elegant and easier to implement. In this case the classifier should encode not only the class of the object but also the length of the string.

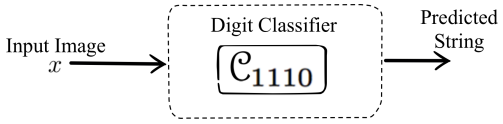


Figure 4. An end-to-end solution for touching digits.

In fact the end-to-end solution is easier to implement, since it is based on a single classifier. However, as we will discuss in Section IV, this solution makes some confusions that could be easily solved having the information about the length of the string. With that in mind, we assess a third strategy (Figure 5), in which we combine the output of the $\mathcal{C}_{1110}$ classifier with the Length classifier ($\mathcal{L}$). This approach uses the same fusion rule described earlier in this section. The difference is that the probability is produced by a single classifier instead of three.

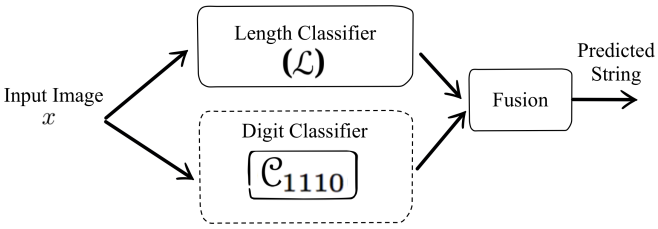


Figure 5. Single digit classifier combined with Length classifier ($\mathcal{L}$).

Figure 10a exemplifies the fusion process. In this case, $\mathcal{C}_{1110}$ misclassifies the input '60' by assigning it to class '610'. However, using the Top-1 output of $\mathcal{L}$, the correct class may be selected. In the case illustrated in Figure 10b, because $\mathcal{L}$ produces a score smaller than T, the two outputs (Top-1 and Top-2) of $\mathcal{L}$ was used to solve the confusion.

A. Classifiers

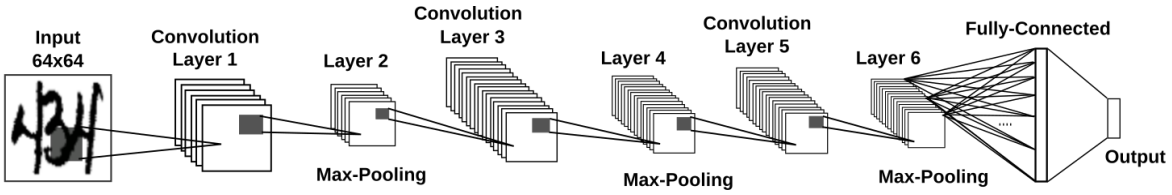
All the classifiers used in this work are CNNs that are constructed using multiple layers considering the following operations: convolutions, max-pooling, and dot products (fully-connected layers), where convolutional layers and fully connected layers have learnable parameters that are optimized during training. With the exception of the last layer in the network, after each learnable layer we apply ReLU non-linearity. The last layer uses the softmax non-linearity.

Training is performed with the Stochastic Gradient Descent (SGD) using back-propagation with mini-batches of 256 instances, a momentum factor of 0.9 and a weight decay of 5×10^{-4}. The learning rate is set to 10^{-2} in the beginning to allow the weights to quickly fit the long ravines in the weight space, after which it is reduced over the time (until 5×10^{-4}) to make the weights fit the sharp curvatures. The network makes use of the well known cross-entropy loss function.

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 performance of the network 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. To implement the CNN models we have used the Caffe framework [12] on a NVidia GeForce GTX Titan Black GPU and NVidia GeForce GTX Titan Xp GPU[2].

1) Length Classifier: The length classifier ($\mathcal{L}$) was designed to predict the length of x. We have tested several different architectures for this classifier but the one that yielded the best results was based on the well-known LeNet 5 [4]. The final architecture contained three convolutional layers followed by max pooling layers. This architecture, which was defined empirically on the validation set, is depicted in Figure 6.

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



Models	Input	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Fully Connected	Output
$\mathcal{L}$	64x64 Binary	Convolution 5x5@1@6	Max Pooling 2x2@2@6	Convolution 5x5@1@32	Max Pooling 2x2@2@32	Convolution 3x3@1@20	Max Pooling 2x2@2@20	72	4

Figure 6. CNN architecture for $\mathcal{L}$. Layer parameters are represented as Kernel Size @ Stride @ Feature Maps.

The classifier was trained using the protocol described in Section III-A using 400,000, 79,157 and 79,742 samples (uniformly distributed) for training, validation, and testing, respectively. Using the Caffe framework and the hardware mentioned in Section III-A, it took about 90 minutes to train this model over 30,000 iterations. Classifying a single input image takes about 0.4 milliseconds (ms). In our experiments, the best results were achieved when the input image was resized to 64×64 pixels. The recognition rate on the testing set was 98.4% and 99.9% for Top-1 and Top-2, respectively. Table II shows the confusion matrix.

Table II
CONFUSION MATRIX (%) FOR THE $\mathcal{L}$ ON THE TESTING SET.

Length	(1)	(2)	(3)	(4)
(1)	99.9	0.01		
(2)	0.02	99.2	0.07	
(3)		0.9	96.9	2.3
(4)			2.3	97.7

Analyzing the confusions resulting from $\mathcal{L}$ we conclude that the number and location of the vertical strokes seem to bear important information needed to determine the size of the string. For example, single digits that are classified as 2-digit string are often slashed zeros, zeros with missing parts, and the digit "6" similar to those presented in Figure 7a and b. Digits that are almost overlapping such as the "3" and "9" in Figure 7c and strings with several vertical strokes close together such as in the "44" in Figure 7d are also sources of confusion.

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

Figure 7. Some images misclassified by $\mathcal{L}$: (a) single digit classified as 2-digit string, (b) 2-digit classified as 3-digit string, (c) 3-digit classified as 2-digit string, and (d) 3-digit classified as 4-digit string.

2) Digit Classifiers: The classifiers $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_{1110}$ presented in the previous section are based on the architecture depicted in Figure 8. The four CNNs, which also are based on the LeNet 5 [4], share the same structure but with different numbers of filters, kernel sizes, and strides. Figure 8 summarizes the parameters used in all four classifiers, which were defined empirically on the validation set.

Table III shows the amount of data used for training, validation, and testing for all four classifiers. It also shows training (30,000 iterations) and classification time using the Caffe framework and the hardware mentioned in Section III-A.

All four classifiers were trained using the protocol described in Section III-A and yielded the accuracies reported in Table IV.

IV. EXPERIMENTS

In order to validate the segmentation-free strategies we have used the 79,464 images of touching digits available in the Touching Pairs (TP) database [1]. This dataset allows us to better compare with the literature. We also perform experiment on the dataset described in Section II, which contains single digits, 2-, and 3-digit connected.

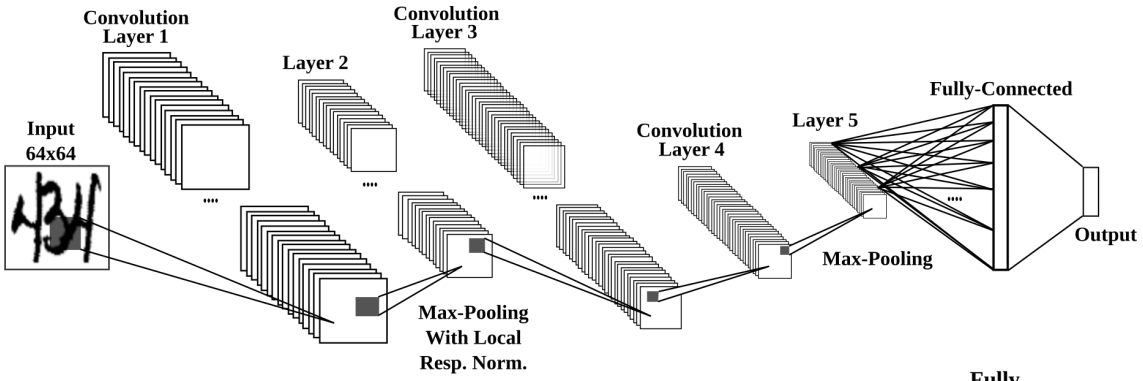
It is important to mention that, because these datasets contains only a single connected component per image, the pre-processing module was suppressed in those approaches. Also, the threshold value T from Equation 1 was set to 0.95 according to the authors [7].

A. TP dataset

When evaluating the segmentation algorithms, the authors in [1] were interested in knowing whether or not the segmentation cuts produced by the algorithms were the good ones, independently of their quantity. For the algorithms based on the segmentation-recognition approach, this task is straightforward, since there is only one hypothesis to be assessed. For those algorithms based on over-segmentation, all the cuts must be assessed. In the latter case, the strategy used is as follows: if there are two digits among the hypotheses (using a classification engine) that match to the ground truth, the segmentation is considered successful. It is clear that this strategy considers the best case scenario since all misclassifications due to over- and under-segmentation are not considered.

Table V summarizes the results reported in [1] and [13] where the authors compare several segmentation algorithms in terms of correct segmentation on the TP database. Besides the overall performance, this table also shows the performance depending on the connection types depicted in Figure 9.

Table V also allows us to draw some conclusions. Algorithms based on a single segmentation hypothesis (segmentation cuts = 1) usually fail in more complex touching cases



Models	Input	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Fully Connected	Output
$\mathcal{C}_1$	64x64 Binary	Convolution 7x7@3@64	Max Pooling and Local Resp. Norm. 3x3@2@64	Convolution 3x3@1@128	Convolution 3x3@1@32	Max Pooling 3x3@2@32	128	10
$\mathcal{C}_2$	64x64 Binary	Convolution 7x7@3@72	Max Pooling and Local Resp. Norm. 3x3@2@72	Convolution 3x3@1@192	Convolution 3x3@1@64	Max Pooling 3x3@2@64	1024	100
$\mathcal{C}_3$	64x64 Binary	Convolution 7x7@1@24	Max Pooling and Local Resp. Norm. 2x2@2@24	Convolution 5x5@1@42	Convolution 5x5@1@32	Max Pooling 2x2@2@32	1200	1000
$\mathcal{C}_{1110}$	64x64 Binary	Convolution 7x7@1@24	Max Pooling and Local Resp. Norm. 2x2@2@24	Convolution 5x5@1@42	Convolution 5x5@1@32	Max Pooling 2x2@2@32	1200	1110

Figure 8. CNN architecture for digit classifiers. Layer parameters are represented as Kernel Size @ Stride @ Number of Feature Maps.

Table III
DATA USED TO TRAIN THE DIGIT CLASSIFIERS.

Classifier	Number of Classes	Amount of data (×1000) for Train	Validation	Testing	Source	Training Time (min)	Classification Time (ms)
$\mathcal{C}_1$	10	197	23	23	NIST SD 19	70[1]	0.57[1]
$\mathcal{C}_2$	100	161	53	55	Synthetic data	90[1]	0.60[1]
$\mathcal{C}_3$	1000	1448	484	491	Synthetic data	200[1]	0.63[1]
$\mathcal{C}_{1110}$	1110	1808	561	570	Synthetic data	183[2]	1.10[1]

(1) NVIDIA Titan Black GPU and (2) NVIDIA Titan Xp GPU

Table IV
RECOGNITION RATE OF THE DIGIT CLASSIFIERS ON TESTING SET.

Classifier	Top 1	Top 2
$\mathcal{C}_1$	99.6	99.9
$\mathcal{C}_2$	99.7	100.0
$\mathcal{C}_3$	97.7	98.9
$\mathcal{C}_{1110}$	95.9	98.6

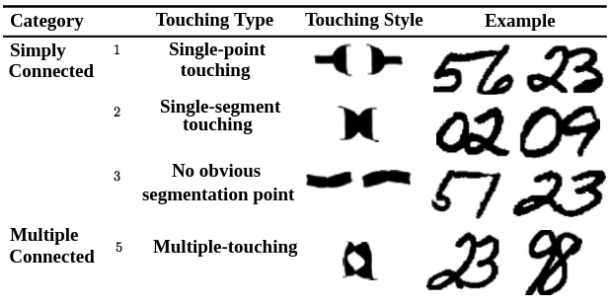


Figure 9. Types of connected numeral string (extracted from [1]).

(e.g., type V) since a single segmentation cut is very often not enough to correctly split the digits. Algorithms based on multiple cuts, on the other hand, achieve better performance in terms of finding the correct segmentation cut, but with the computational cost of having to evaluate several hypotheses.

In this context, the segmentation-free approaches compare favorably to traditional segmentation algorithms. In the End-to-End approach the expensive process of finding the segmentation cuts, filtering out unlikely hypotheses, and classifying the remaining ones is replaced by one classifier call ($\mathcal{C}_{1110}$). As we can see in Table V, this simple approach achieves 94.37% of correct classification, which compares to the best methods reported in the literature, Chen and Wang [22] and Gattal et al.[13]. However, these two methods generate a large

Table V
PERFORMANCE OF THE SEGMENTATION ALGORITHMS (REPORTED IN [1] AND [13]), IN TERMS OF CORRECT SEGMENTATION, ON THE TP DATABASE.

Method	Performance	Connection Type (%)				Segmentation
	%	I	II	III	V	Cuts
Shi and Govindaraju [14]	59.30	68.31	59.72	60.35	25.44	1
Congedo et al. [15]	63.07	62.88	67.51	59.40	40.45	1
Lacerda and Mello [16]	65.79	71.75	71.21	63.64	56.57	1
Elnagar and Alhajajj [17]	67.34	63.88	71.51	56.40	58.73	1
Pal et al. [18]	71.21	73.96	74.69	80.09	41.52	1
Oliveira et al. [19]	88.03	90.40	90.78	89.01	64.88	1
Fusijawa et al. [20]	89.85	95.45	91.27	83.57	63.72	3.66
Fenrich and Krishnamoorthy [21]	92.37	97.54	93.79	99.45	65.57	4.07
Gattal and Chibani [13]	93.24	96.67	93.75	99.68	77.58	24.11
Chen and Wang [22]	93.80	97.87	94.23	97.55	76.76	45.40
Proposed End-to-end	94.37	94.33	95.13	96.13	91.90	0
Proposed End-to-end+$\mathcal{L}$	96.05	95.95	96.87	98.03	93.35	0
Hochuli et al. [7]	97.12	97.02	97.89	98.97	93.03	0

number of hypotheses, which makes them unfeasible for real applications due to the high computational cost.

Figure 10 shows some confusions made by the $\mathcal{C}_{1110}$ classifier. Most of the errors are related to touching pairs confused with single digits or 3-digit strings. In light of this, the End-to-End approach could benefit somehow from the information provided by the Length classifier ($\mathcal{L}$), which is the strategy depicted in Figure 5.

Using the End-to-End+($\mathcal{L}$), some of these confusions are solved increasing the recognition rate in about two percentage points (96.05%). The total error (3.95%) is caused in parts by $\mathcal{L}$ (1.76%) and $\mathcal{C}_{1110}$ (2.79%). In terms of computational cost, there is a little penalty since we have to add another classifier call and the fusion rule. However, compared to the traditional segmentation algorithms the cost is still negligible.

Finally, the dynamic selection strategy presented in Figure 3 solves some of the confusions caused by the $\mathcal{C}_{1110}$. Instead of using a general-purpose classifier for 1-, 2-, and 3-digit strings, it divides the classification task into three parts, creating this way task-specific classifiers. In this experiment, though, only one of them is used along with $\mathcal{L}$. Assume that the size of the string is unknown, $\mathcal{C}_2$ is only used to classify the images that were assigned as 2-digit string by $\mathcal{L}$. This strategy reaches the highest performance (97.12%). Comparing to the End-to-End+($\mathcal{L}$), the classification error was reduced from 2.79% to 1.10%. Figure 11 shows some images that were misclassified by $\mathcal{C}_2$. As reported in Table V, the poorest performance (93%) is achieved on type V (multiple touching), which shows the highest variability. However, when compared to others, such a performance is outstanding.

B. Synthetic Data

In the previous experiments only touching pairs were considered so that we could compare the segmentation-free approaches with the literature. Besides, segmenting touching pairs is the main bottleneck of any string recognition system. However, in several cases, a digit string is composed mainly by isolated digits and sometimes it may contain three or more digits connected. In this section we assess the segmentation-free approaches on the synthetic data described in Section II, which contains over 570,000 images of isolated digits, touching strings of 2- and 3-digits.

One may argue that recognition of isolated digits is a problem already solved since the literature shows accuracy close to 100% [23], [24]. It is worth remembering, however, that the lack of context in digit string recognition makes the problem more challenging since an image may contain an isolated digit or several digits connected. To deal with this problem, heuristic-based segmentation algorithms rely on over-segmentation to maximize the chances of finding the correct segmentation point, even when segmentation is not necessary. This strategy has a downside, i.e., isolated digits that do not need segmentation may be segmented and the over-segmented pieces recognized with high probabilities. This is exemplified in Figure 12 where the digit "9" was over-segmented into two parts, which were recognized as "0"and "1" with high probability.

Table VI shows the results of the three segmentation-free approaches discussed in this work.

Table VI
PERFORMANCE OF THE SEGMENTATION-FREE APPROACHES ON THE SYNTHETIC DATA.

Method	Single digit	2-digit	3-digit
End-to-end	97.68	94.09	96.05
End-to-end+$\mathcal{L}$	98.73	96.82	95.50
Hochuli et al. (2018)	99.56	99.00	94.88

The results achieved by both End-to-end+$\mathcal{L}$ and Hochuli et. al. [7] corroborate to the importance of the Length classifier when recognizing strings of digits of unknown length. Several confusions between isolated digits and touching digits are solved by using the information about the number of digits in the string.

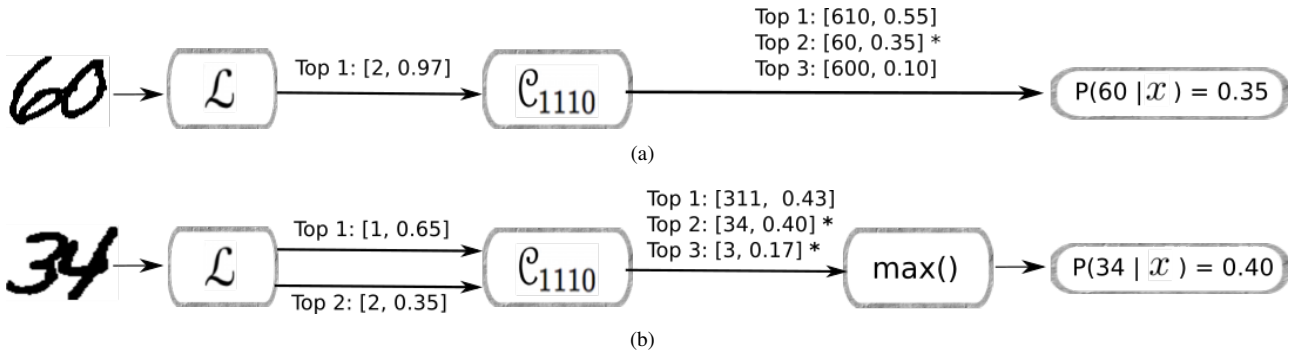


Figure 10. Confusions made by $\mathcal{C}_{1110}$ classifier - [Prediction, Probability]: Missed predictions for (a) '60' and (b) '34' were solved using information provided by the Length classifier ($\mathcal{L}$) and the fusion strategy from Equation 1.

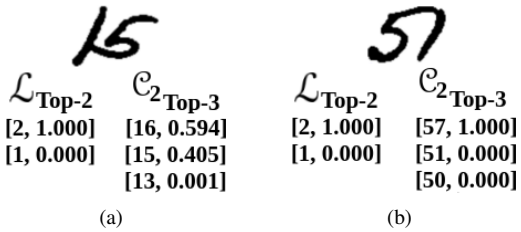


Figure 11. Confusions made by $\mathcal{C}_2$ classifier - [Prediction, Probability]: (a)'15' predicted as '16', (b) '51' predicted as '57'.

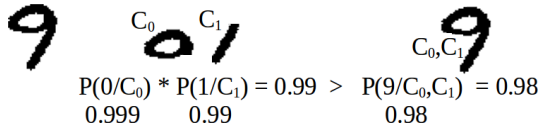


Figure 12. Misclassification caused by over-segmentation (extracted from [25]).

In the case of 3-digit strings, which are not very often in real datasets, most of the confusions occur intra-class, e.g., "426" confused with "406" depicted in Figure 13. Since strings with three touching digits contain more information to encode the size of the string the Length classifier does not contribute to improve the recognition rate. On the other hand, segmentation-based approaches will suffer with a higher number of hypotheses to be assessed.

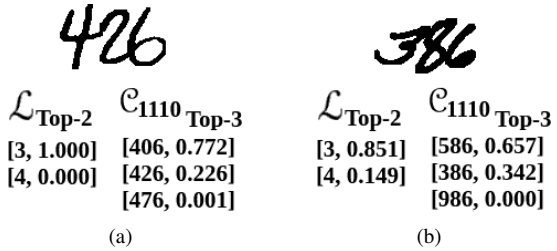


Figure 13. Confusion made by $\mathcal{C}_{1110}$ - [Predicion,Probability]: (a) '426' predicted as '406' and (b) '386' predicted as '586'.

V. Conclusion

Since segmentation of touching digits remains a challenge for handwritten numeral recognition, in this paper we have presented segmentation-free approaches corroborating with the recent work [7] that achieved state-of-art performance through a dynamic selection strategy based on four deep learning models. Towards an end-to-end solution, we have implemented a touching digit classifier that is capable to discriminate 1110 classes (10 for isolated, 100 for pairs and 1000 for triples). Using a strong experimental protocol, the proposed approaches surpass segmentation-based methods bringing up a new perspective to the problem. Further analysis confirmed that introducing context information related to the length of string predicted by a trained classifier is an useful strategy to solving some confusions made by digit classifiers. For future works, we are developing an approach that combine length and digits classifiers into an end-to-end solution.

Acknowledgements

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 303513/2014-4. In addition, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] F. C. Ribas, L. S. Oliveira, A. S. Britto, and R. Sabourin, "Handwritten digit segmentation: A comparative study," *International Journal on Document Analysis and Recognition*, vol. 16, no. 2, pp. 567–578, 2013.

[2] E. Vellasques, L. S. Oliveira, A. S. Britto, A. Koerich, and R. Sabourin, "Filtering segmentation cuts for digit string recognition," *Pattern Recognition*, vol. 41, no. 10, pp. 3044–3053, 2008.

[3] O. Matan, J. C. Burges, Y. LeCun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," in *Advances in Neural Information Processing Systems*, J. E. Moody, S. J. Hanson, and R. L. Lippmann, Eds. Morgan Kaufmann, 1992, vol. 4, pp. 488–495.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Procs of IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] S. Choi and I. Oh, "A segmentation-free recognition of two touching numerals using neural networks," in *Proc. of 5^{th} International Conference on Document Analysis and Recognition*, Bangalore, India, 1999, pp. 253–256.

[6] D. Ciresan, "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*, 2008, pp. 225–230.

[7] A. G. Hochuli, L. S. Oliveira, A. S. Britto, and R. Sabourin, "Handwritten digit segmentation: Is it still necessary?" *Pattern Recognition*, vol. 78, pp. 1 – 11, 2018.

[8] A. S. Britto, R. Sabourin, and L. S. Oliveira, "Dynamic selection of classifiers—a comprehensive review," *Pattern Recognition*, vol. 47, no. 11, pp. 3665 – 3680, 2014.

[9] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195 – 216, 2018.

[10] P. J. Grother, *NIST Special Database 19 - Handprinted forms and characters database*, NIST, 2016.

[11] X. Wang, V. Govindaraju, and S. N. Srihari, "Holistic recognition of handwritten character pairs," *Pattern Recognition*, vol. 33, no. 12, pp. 1967–1973, 2000.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[13] A. Gattal and Y. Chibani, "SVM-based segmentation-verification of handwritten connected digits using the oriented sliding window," *International Journal of Computational Intelligence and Applications*, vol. 14, no. 1, pp. 1–17, 2015.

[14] Z. Shi and V. Govindaraju, "Segmentation and recognition of connected handwritten numeral strings," *Pattern Recognition*, vol. 30, no. 9, pp. 1501–1504, 1997.

[15] G. Congedo, G. Dimauro, S. Impedovo, and G. Pirlo, "Segmentation of numeric strings," in *3rd International Conference on Document Analysis and Recognition*, 1995, pp. 1038–1041.

[16] E. Lacerda and C. A. B. Mello, "Segmentation of connected handwritten digits using self-organizing maps," *Expert Systems with Applications*, vol. 40, pp. 5867–5877, 2013.

[17] A. Elnagar and R. Alhajj, "Segmentation of connected handwritten numeral strings," *Pattern Recognition*, vol. 36, no. 3, pp. 625–634, 2003.

[18] U. Pal, A. Belaid, and C. Choisy, "Touching numeral segmentation using water reservoir concept," *Pattern Recognition Letters*, vol. 24, pp. 261–272, 2003.

[19] L. S. Oliveira, E. Lethelier, F. Bortolozzi, and R. Sabourin, "A new approach to segment handwritten digits," in *Proc. of 7^{th} International Workshop on Frontiers of Handwriting Recognition*, Amsterdam, Netherlands, 2000, pp. 577–582.

[20] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: from segmentation to document structure analysis," *Proc. of IEEE*, vol. 80, pp. 1079–1092, 1992.

[21] R. Fenrich and S. Krishnamoorthy, "Segmenting diverse quality handwritten digit strings in near real-time," in *5^{th} USPS Advanced Technology Conference*, 1990, pp. 523–537.

[22] Y. K. Chen and J. F. Wang, "Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1304–1317, 2000.

[23] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3642–3649.

[24] S. Sabour, N. Frosst, and G. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

[25] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Automatic recognition of handwritten numerical strings: A recognition and verification strategy," *IEEE Trans. on Pattern Analysis on Machine Intelligence*, vol. 24, no. 11, pp. 1438–1454, 2002.