A Deep Learning Approach to Handwritten Number Recognition

Victoria Ruiz, Maria T. Gonzalez de Lena, Jorge Sueiras, Angel Sanchez, and Jose F. Velez(✉)

Universidad Rey Juan Carlos, c/Tulipan sn, Mostoles, Madrid, Spain
{victoria.ruiz.parrado,mariateresa.gonzalezdelena,jorge.sueiras, angel.sanchez,jose.velez}@urjc.es

Abstract. Nowadays, Deep Learning is one of the most popular techniques which is used in several fields like handwriting text recognition. This paper presents our propose for a handwritten digit sequences recognition system. Our system, based in two stage model, is composed by Convolutional Neural Networks and Recurrent Neural Networks. Moreover, it is trained using on-demand scheme to recognize numbers from digits of the MNIST dataset. We will see that, with these training samples is not necessary segment or normalize the input images. Average recognition results were on 88,6% of accuracy in numbers of variable-length, between 1 and 10 digits. This accuracy is independent on the number length. Moreover, in most of the wrongly predicted numbers there was only one digit error.

Keywords: Handwritten character recognition · Synthetic number database · Advanced convolutional neural networks · Recurrent Neural Networks · Deep learning

1 Introduction

Handwriting style is a personal behavior characteristic which can differentiate between individuals and consequently can be used as a biometric modality. Handwriting text recognition (HTR) is still an open research problem today due the multiple difficulties to be analyzed in an automatic form [18,23]. For example, the intrapersonal and interpersonal variabilities in handwriting make the task quite difficult. That means not everyone writes with the same style and the same person could not write in the same way at each time and situation. Also, it is very common to capture some noise in the image that could produce classification errors while performing this task in an automatic method.

The handwriting recognition problem uses two approaches based on data capture process: on-line and off-line recognition [18]. The data available in the on-line recognition are time series of coordinates, representing the movement of the pen. In the off-line case the data are only an image of the text. Because the data available are more detailed and noised free, the on-line recognition obtains generally better results [19]. For the off-line case, there are also two main approaches: holistic and segmentation-based ones [15,22]. While in the holistic (or global) approach the classification is carried out using directly the whole word image, in the segmentation-based case, the isolated characters are extracted before performing their classification.

Typically, a HTR system has five main steps: pre-processing, segmentation, feature extraction, training, recognition and post-processing [17]. In the segmentation-based case and previous to character extraction, it is a frequent practice to normalize the text images to reduce their variability in order to achieve better recognition results. Although there are several techniques which handle isolated characters, and some new techniques which recognize large vocabulary with the help of some dictionaries or grammars [12], there are still few works that consider the problem of recognizing long character sequences. In this paper, we present a new method for such handwriting text recognition problem, which is focused on the off-line approach and does not need from any normalization or segmentation stage. Our approach was tested using variable-length digit sequence images.

Despite the fact that there are many improvements in the HTR problem, especially in the on-line case, there are still several open problems to be solved [12,17]. The possible high variability in the text words and the image noise presence must also be taken into account. In particular, length variability in words may difficult the character segmentation task, and the presence of noise can cause the system wrongly isolates some characters.

One additional important problem we have to deal with, is the variable length of words (or digits in numbers). There exist some systems which produce very good recognition results on isolated characters (or digits), but their performance drop down when dealing with large words or numbers [12]. For example, different methods are applied to classify isolated characters [7,17] from the MNIST database [14]. One of these latest methods, proposed by Ciresan et al. [5], achieved a 99.77% of accuracy. These methods used Hidden Markov Models (HMM), Neural Netwoks (NN), Hybrid NN/HMM and Convolutional Networks (CNN) as classifiers [2,16]. More recently, some other proposals [9] aimed to recognize complete text words using Recurrent Neural Networks (RNN) [3,7,9] such as the Long-Short Term Memory (LSTM) [10,11], complemented with some types of language models.

Normalizing the images is one of the most common way to reduce the variability in handwritten text. As it is shown by Arica et al. [1], skew normalization and baseline extraction, slant normalization, size normalization and contour smoothing are some of the most common techniques for reducing text variability and simplify the practical problem. In our approach, for the sake of producing a more realistic solution, we avoid this image normalization through increasing the training database through data augmentation. Moreover, we generate the inputs ‘on demand’ from a handwritten numbers database, as it will be explained in the next Section.

For the above referred normalization-based methods, after that step a segmentation algorithm is applied in order to get isolated characters. One of the most used ones consists on splitting the words into connected components which are classified as characters [15]. As our method does need neither from normalization nor segmentation, the text and noise variability conditions will be higher and this makes it more difficult the recognition task as happens in many practical situations.

In this paper, we describe a new deep learning approach for handwritten number recognition. Our method has been successfully tested on large numbers (i.e. long chains of digits). The proposed system learns by an 'on demand' basis which generate an infinite number of samples, instead of using a standard fixed-length database.

The paper is organized as follows. Section 2 describes the building of our experimental number database produced using the MNIST dataset. Section 3 outlines the system architecture for the considered problem. Experimental results are presented and explained in Sect. 4. Finally, the last Section summarizes the conclusions of this work.

2 Handwritten Digit Database

The samples used to train and test our system are built on the well-known MNIST database [6]. This database consists of by 70.000 digit images, which of 60.000 are used for training and the remaining 10.000 are for tests. All of these samples are black and white images, centered and size-normalized (with 28×28 pixels). We use these images to create the new sample, as follows:

1. By combining randomly the MNIST digits to create variable-length numbers, a of digits which will be the data of our sample datasets.
2. Move and transform the generated sequenced in order to add variability.

Figure 1 shows the process of sample generation.

2.1 Generation of Sequence Numbers from MNIST Database

To develop our handwriting number recognition system, we have created a generator of numbers from the MNIST database images. In this generator it is possible to choose the numbers length with a maximum of 10 digits, the number of classes which are in the sequence, and the transformation applied to the images. So that:

1. First, the number of digits of the sequence is randomly chosen and an empty image of size 28×280 is created.
2. Second, the allowed digit classes are also randomly chosen.
3. Third, the following processes are performed as many times as number of digits the new sequence has:

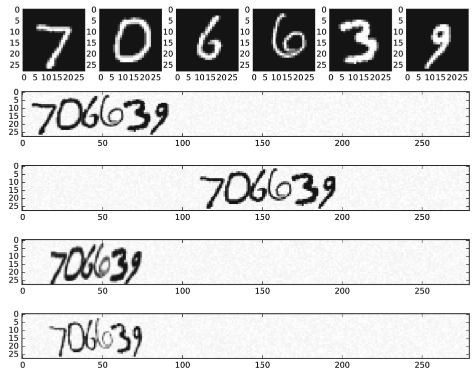


Fig. 1. Example of a generated number sequence. The first row shows the original MNIST digits, the second row show a generated digit sequence and the rest of rows shows some transformations which can be applied to the image sequence

   (a) Random choice of the digit between the allowed classes
   (b) The new digit is added to the 28×280 image. This digit is placed as close as possible to the remaining digits of the sequence

4. As the digits are joined without spaces, the width of the created number is less than 280, so it is possible to move the image around the free pixels. So that, if it is chosen, the digits can be randomly moved.

2.2 Transformations Applied to Images

To add variability and produce a potentially infinite sample, some transformations are applied to the created image. The generator can choose whether the following transformations are applied:

1. Affine transformations to the right or to the left in order to rotate the images.
2. Increase the scale of the digits by width or height.
3. Morphological erosions.
4. Morphological dilations.
5. Translate the positions of digits.

After these transformations a random background (with pixels between 240–255 gray values) is added to avoid the noise which is produced when a transformation is applied. Note that it is possible to apply more than one transformation into the same generated sequence. On the other hand, in the translation transformation, the sequence will be displaced to the right randomly or by a fixed scalar. Moreover, as we explain in the next Section, our system has two different models: a convolution neural network which is used to feature extraction, and a recurrent neural network which predicts the digit ordering of the sequence. In the convolution model the random translate transformation has been applied, while in the recurrent model the images were moved 8 pixels.

3 Classification

The proposed number recognition system (see Fig. 2) includes two main steps: (1) Convolutional Neural Network model (CNN) to obtain image digit features and (2) Long-Short Term Memory (LSTM) Network to recognize the digit sequence. Convolutions for feature extraction have been used formerly in works like Bluche [2] who applied them to predict the states of a Hidden Markov Model. In our system, we get the digits features by predicting the existing digits of the sequence in a CNN model based on the VGG architecture [21]. For that, the target of that model is an array of size 10 which outputs 1 if the number is in the image or 0, otherwise. Note that the CNN model only detect the presence of digits classes in the number sequence, but not their positions.

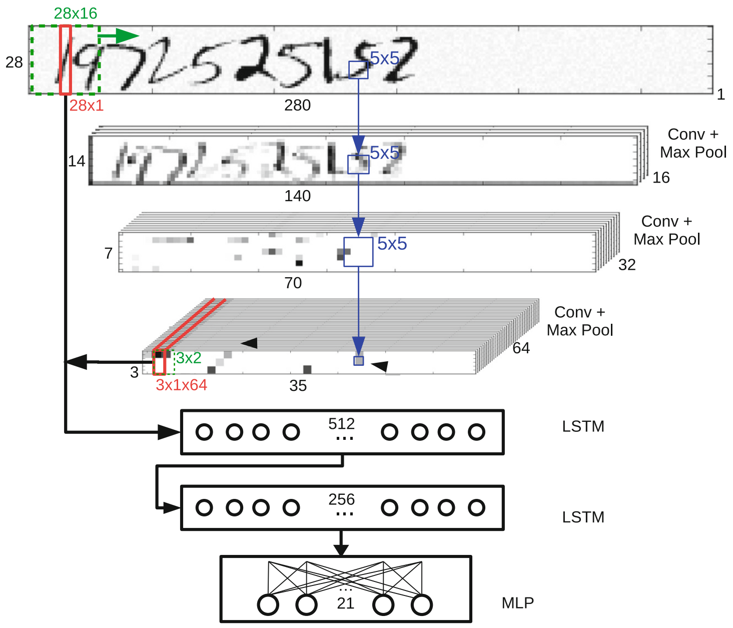


Fig. 2. General model architecture. Blue boxes correspond to the convolution masks. Green boxes correspond to the sliding window. Red boxes correspond to the tensors which are the inputs of the LSTM model. (Color figure online)

As the positions of the digits are important to get the number, a LSTM model placed at the output of the CNN model is applied to perform local predictions about positions of existing digits in the number. Moreover, to distinguish two or more consecutive identical digits, we transform each class in two classes. Instead of having class 0 we have class 'beginning of 0' (b0) and class 'end of 0' (e0), and so on with the rest of the classes. With this type of target, when we present the sequence 22 to the LSTM model, we get the output 'b2e2b2e2'. We use also and extra class which correspond to the ‘blank’ class, which is similar to Graves et al. uses [8]. Figure 2 shows the general model architecture of our system.

The CNN model is a simpler version of the VGG architecture and aims to predict just if a digit is present or not in the number. This model was trained in an incremental process where the number of classes were increased slowly. The training process of the CNN model had 10 phases:

1. In the first phase, the inputs had a variable length between 1 and 10. Moreover, each input only contains random patterns from one class. This first phase ended when the training error was lower than 1%.
2. In the second phase, the process was the same, but random patterns were chosen between 1 or 2 different classes.
3. We repeat this incremental process until the last phase, where all the classes could be present in the numbers.

Table 1 shows examples of the CNN model inputs. As it is shown in Fig. 3, the final CNN model is composed by three groups of two convolution layers followed by a sub sampling (in particular max-pooling) layer, and two dense layers. In the first dense layer we used a dropout [20] of 60%. All the convolutions layers have a stride of 3×3 (which is equivalent to have a stride of 5×5 due to there are two consecutive convolution layers) and zero-padding. Also, all the neurons have ReLU [13] as activation function.

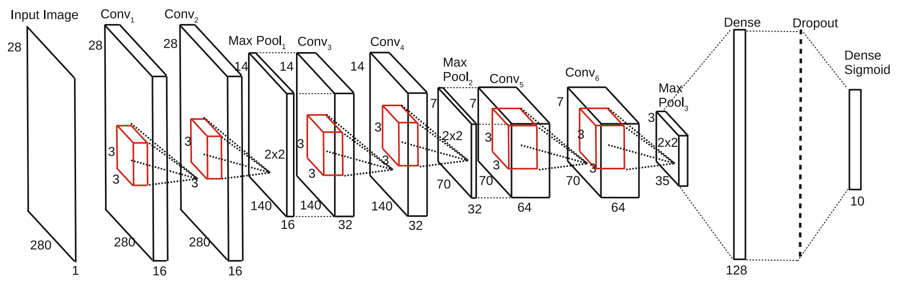


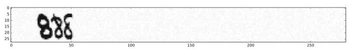
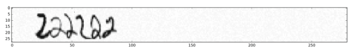
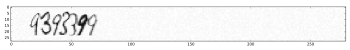
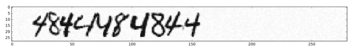
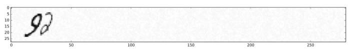
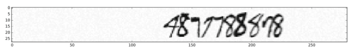
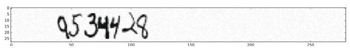
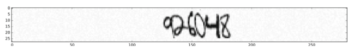
Fig. 3. CNN model architecture.

By the other hand, the inputs of the LSTM model come from a sliding window over the original image merged with the last convolutional layer of the CNN model. We chose the last convolutional layer, instead of the last dense layer, because we are interested in the local information obtained from the convolutions. To build the LSTM inputs, we scored each image and reshape the obtained $64 \times 3 \times 35$ tensors in a 192×35 vector. In the next step, we merge the convolutions with the original image, joining each column of the image (28 values) with its corresponding columns of the convolution layers (192), obtaining 220 values. Finally, each input sample for the LSTM training process is composed of a sliding window of 16 columns, and so of 220×16 values.

The LSTM model that we built has two LSTM layers and a dense layer as it is shown at the bottom of Fig. 2. For each input tensor, the output is a new tensor of 1×21. As the original image has a width of 280, we append the 280 outputs to get the output sequence of 'beginnings' and 'ends'. The final step is to convert the 'beginning-end' label sequence in the final digit sequence. Note that, the final softmax output give us different possibilities for each class, and we need only one. So, we select the combination that maximize the sum of the selected classes, taking into account two rules:

1. If there is a 'beginning' of a number it is necessary to have an 'end' of the same number.
2. The 'blank' can be applied only after an 'end'.

Table 1. Example of phases. In every phase the generated numbers have between 1 and 10 digits. In Phase 1 there is only 1 different class, in Phase 2 two classes, and so on until phase 10 where all the classes are can be present in the same sequence.

Phase	Example 1	Example 2
Phase 1		
Phase 2		
Phase 3		
...	...	...
Phase 10		

4 Experimental Results

Experiments have been carried out in two stages. In the first one, the CNN model was trained in several phases as it was explained in previous Section. We use a learning rate of 0.01 and the Stochastic Gradient Descent (SGD) [4] algorithm as optimizer and we need 60 epochs to train the whole model. To achieve the error classification we build a test number sequence dataset with 1000 images, which was created with the same algorithm as in the training case, except that in the test case no transformation was applied on images. Figure 4(a) shows an example of an original test image, and the scores that we got with the CNN+LSTM model. In the example is how increase the 'beginning' and the 'end' of each number.

The CNN model produced 99% Character Accuracy Rate (CAR) and a 90% of Sequence Accuracy Rate (SAR), which means that in 90% of the cases all the digits were correctly found. By the other hand, in the global CNN+LSTM model, there is a 98,4% of accuracy in CAR and an 88,6% of accuracy in SAR. To measure how the CNN model improves the final results, we trained a LSTM model with the same architecture, but only with the original image as input data. The results were a few worse than the presented system as we obtained 86,2% of accuracy in SAR. These results are shown by Table 2.

Figure 4(b) shows the comparison between a perfect model, which would predict successfully all the numbers, and our system. This graphic is built sorting the tested numbers by their confidence and counting how many numbers are successfully predicted. It is seen that if we would predict just the 20% of the tested numbers with more confidence all the digits would be successfully classified. If we would predict the numbers with less confidence numbers, the accuracy would start to decrease until the 88,6% (i.e. the accuracy that we achieved when owe predicted all the numbers of the test sample).

Finally, in Fig. 5(a) shows the distribution of the edit distance on the test numbers. It shows that 88,6% of the tested numbers have 0 as edit distance

Table 2. Summary of recognition results of our system for isolated digits (CAR column) and variable-length numbers (SAR column)

	CAR	SAR
CNN+ LSTM model	98,4%	88,6%
Only LSTM	98,0%	86,2%

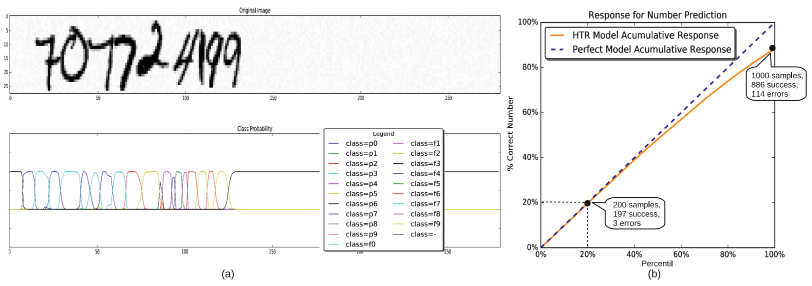


Fig. 4. (a) Example of an test image and their scores with the CNN+LSTM model. (b) Comparison between the accumulative response in a perfect model and our system.

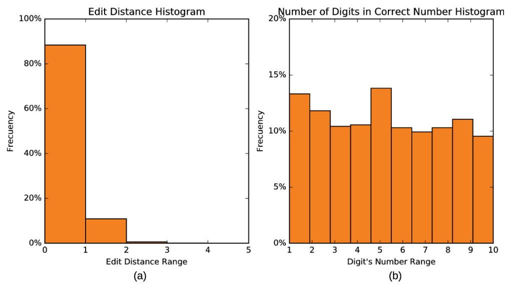


Fig. 5. Recognition results: (a) frequency of tested numbers by their edit distance and (b) frequency of well classified test number by their number length

(that means that they were correctly predicted), an 11% have edit distance of 1 (and just 1 digit were wrongly predicted), and a 0.4% have han edit distance of 2 or more. Figure 5(b) shows that the distribution of the successfully recognized numbers is uniform over the number length. So that, it was not relevant the number of digits in the sequences for our classifications results.

5 Conclusion

This paper describes a new automatic system for recognizing numbers of variable length. This system is based on deep learning neural models and was trained using an 'on demand' scheme to produce numbers using digit images of the MNIST dataset. We avoid the normalization step of many handwriting recognition systems. Moreover, our solution does not need to segment the digits and classifies the whole sequence as a number. Our model architecture was divided in two steps. In the first step a Convolutional Neural Network (CNN) model was trained to extract the digit features. This model was trained in several phases where the number of digits and classes were increased as well as the architecture of the model. In the second step, the convolutions were used with the original image by a LSTM model in order to predict the beginning and the end of each digit. Finally, the obtained digit sequence was transformed in the recognized number. Using this system, an 88,2% of the test numbers were correctly predicted. Moreover, we noticed that the lengths of the numbers do not influence in the classification and, when a number was misclassified, only one digit of it was wrongly predicted.

In the future, we would like to improve these models for longer numbers as well as to train them for arbitrary types of sequence of characters.

Acknowledgements. This work was funded by the Spanish Ministry of Economy and Competitiveness under grant number TIN2014-57458-R.

References

1. Arica, N., Yarman-Vural, F.T.: An overview of character recognition focused on off-line handwriting. Syst. Man Cybern. **31**(2), 216–233 (2001)
2. Bluche, T., Ney, H., Kermorvant, C.: Feature extraction with convolutional neural networks for handwritten word recognition. In: 12th International Conference on Document Analysis and Recognition, pp. 285–289 (2013)
3. Bluche, T., Ney, H., Kermorvant, C.: A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In: Besacier, L., Dediu, A.-H., Martín-Vide, C. (eds.) SLSP 2014. LNCS, vol. 8791, pp. 199–210. Springer, Cham (2014). doi:10.1007/978-3-319-11397-5_15
4. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. Adv. Neural Inf. Process. Syst. **20**, 161–168 (2008)
5. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. Conf. Comput. Vis. Pattern Recognit. **2012**, 3642–3649 (2012)

6. Deng, L.: The MNIST database of handwritten digit images for machine learning research. In: IEEE Signal Processing Magazine, pp. 141–142 (2012)
7. Graves, A., Eck, D., Beringer, N., Schmidhuber, J.: Isolated digit recognition with LSTM recurrent networks. In: First International Workshop on Biologically Inspired Approaches to Advance Information Technology (2003)
8. Graves, A., Fernandez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequece data with recurrent neural networks. In: International Conference on Machine Learning, pp. 369–376 (2006)
9. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. Adv. Neural Inf. Process. Syst. **21**, 545–552 (2009)
10. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for improved unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 855–868 (2014)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Koerich, A.L., Sabourin, R., Suen, C.Y.: Large vocavulary off-line handwriting recognition: a survey. Pattern Anal. Appl. **6**(2), 97–121 (2003)
13. Krizhesvky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: 26th Advances in Neural Information Processing Systems (NIPS), pp. 1097–1105 (2012)
14. LeCun, Y., Cortes, C., Burges, C.J.C.: The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist (1998)
15. Lu, Y., Sridhar, M.: Character segmentation in handwritten words-an overview. Pattern Recognit. **29**(1), 77–96 (1995)
16. Matan, O., Burges, C.J.C., LeCun, Y., Denker, J.S.: Multi-digit recognition using a space displacement neural network. Neural Inf. Process. Syst. **4**, 488–495 (1992)
17. Patel, M., Thakkar, S.P.: Handwritten character recognition in english: a survey. Int. J. Adv. Res. Comput. Commun. Eng. **4**(2), 345–350 (2015)
18. Plamondon, R., Srihari, S.N.: On-line and off-line handwriting recognition: a comprehensive survey. Pattern Anal. Mach. Intel. IEEE Trans. **22**(1), 63–84 (2000)
19. Seiler, R., Schenkel, M., Eggimann, F.: Off-line cursive handwriting recognition compared with on-line recognition. In: Proceedings of the International Conference on Pattern Recognition IV-7472, pp. 505–509 (1996)
20. Srivastava, N., Hinton, G., Krizhesvsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
21. Symonian, K., Zisserman, A.: Very deep convolutional networks for large scale image recognition. In: 3rd International Conference on Learning Representation, pp. 1–14 (2015)
22. Vinayakumar, R., Paul, V.: A survey on recognition and analysis of handwrittten document. Int. J. Comput. Sci. Eng. Technol. **6**, 19–23 (2016)
23. Vinciarelli, A.: A survey on off-line cursive script recognition. Pattern Recognit. **35**(7), 1433–1446 (2002)