Noisy Character Recognition using Deep Convolutional Neural Networks

Sirlene Peixoto[1], Gabriel Gonçalves[2], Andrea Bianchi[1], Alceu De S. Brito[3], William Robson Schwartz[2] and David Menotti[4]

[1]Federal University of Ouro Preto, Belo Horizonte (MG), Brazil
[2]Federal University of Minas Gerais, Ouro Preto (MG), Brazil
[3]Pontifical Catholic University of Paraná, Curitiba (MG), Brazil
[4]Federal University of Paraná, Curitiba (MG), Brazil

Abstract. Due to degradation and low quality in noisy images, such as natural scene images and CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) based on text, the character recognition problem continues to be extremely challenging. In this work, we study two convolutional neural network approaches (filter learning and architecture optimization) to improve the feature representations of these images through deep learning. We perform experiments in the widely used Street View House Numbers (SVHN) dataset and a new dataset of CAPTCHAs created by us. The approach to learn filter weights through back-propagation algorithm using data augmentation technique and the strategy of adding few locally-connected layers to the Convolutional Neural Network (CNN) has obtained promising results on the CAPTCHA dataset (97.36% of accuracy for characters and 85.4% for CAPTCHAs) and results very close to the state-of-the-art regarding the SVHN dataset (97.45% of accuracy for digits).

Keywords: Deep Learning, Convolutional Network, Noisy Character Recognition, CAPTCHAs, Street House View Numbers.

1 Introduction

Automatically character recognition in noisy images such as natural scene images and CAPTCHAs based on text is still a difficult task and a recurrently researched subject. Such images usually have complex background, uneven illumination, distorted characters, multiple sources and varying sizes, which lead to misrepresentation of the data. In general, the performance of machine learning methods is highly related to the recognition system ability to extract features to maximize and minimize intra- and inter-class similarities, respectively.

Over the past few years numerous studies show the effectiveness of deep learning techniques in various fields of challenging applications. Deep learning techniques aim to gradually map the representation of raw pixel into more abstract representations. Besides success in recognizing handwritten text [11], deep learning techniques have also been applied in face recognition [9], speech recognition [13], scene recognition [12] problems, among others.

Aiming at improving the feature representation and therefore the classification task, many studies have applied deep learning for character recognition in noisy images [7, 10, 2]. In [7], the authors compare and show the supremacy of two deep learning techniques over two hand-designed descriptors using optimized classifiers in SVHN database. In [10], the authors included a multi-stage characteristic approach and different pooling methods (Lp pooling) in the architecture of a traditional convolutional neural network, achieving an accuracy rate of 95.1% in the SVHN database. More recently, Goodfellow et al. [2] proposed an approach that integrates the three traditional steps of a recognition system (location, segmentation and recognition) by using a deep convolutional neural network, achieving an accuracy of 97.84% in the SVHN dataset.

Despite the importance of the aforementioned contributions, the recognition of noisy characters is still an open problem. In this paper, following the same direction than the literature, we apply deep learning to address such a challenging problem. However, the contribution of this paper is fourfold: i) evaluation of two different approaches to learn/generate the filter weights during the convolution operation of the CNN, named Learning Filters and Architecture Optimization; ii) evaluation of locally-connected layer and data augmentation for training the proposed CNN; iii) comparison of the proposed deep learning recognition method with other approaches based on hand-designed descriptors; and iv) a new dataset composed of noisy character-based CAPTCHA images, which is currently available for research purposes upon request.

According to the experiments based on the well-known SVHN dataset and the proposed CAPTCHA image dataset, we observe that the CNN presents higher accuracy when compared to the hand-crafted approaches evaluated. These results also indicate that the application of the two investigated techniques reduces the error rate in about 2% in the SVHN dataset. We achieved an error rate in this dataset near to the one obtained in [2] using fewer convolutional layers (three against nine). In the CAPTCHA dataset, very promising results were achieved, i.e. 97.36% of accuracy for characters and 85.4% for CAPTCHAs. Moreover, the approach that learns filter weights through back-propagation algorithm using data augmentation technique and the strategy of adding few locally-connected layers to the CNN has obtained promising results for the CAPTCHA dataset.

2 Theoretical Background

A CNN is composed of layers that usually perform a series of four sequential linear and nonlinear operations [8], such as:

1. Convolution: calculates the intensity of a given pixel according to the weighted sum of its neighbors pixels.
2. Activation: maps the output of convolution operation aiming to limit the “neuron” output.
3. Spatial Pooling: reduces the resolution of the map features. It reduces the sensitivity of the output map displacements and other forms of distortion, selecting invariant features that improve the generalization performance.

4. Normalization: usually creates competition between maps. Aims further strengthening the robustness of the CNN output feature vector.

Through these operations, CNN maps the input image to a higher-level representation that can be viewed as a multi-band image. In other words, the CNN gradually transforms the raw pixel in representations highlighting abstract concepts. This multi-band representation is later applied to a classification task.

In this work, we discuss two kinds of CNNs available in the literature that apply different approaches to learn/generate the filter weights of the convolution operation. While one uses a supervised learning algorithm [3], the other optimizes the architecture employing random filters with zero mean and unit norm [8, 4]. Both approaches are briefly detailed in the next sections.

2.1 Learning Filters (LF)

In the LF approach, the convolution layer indeed performs two required operations: convolution and activation, and two additional operations: normalization and pooling. The order of these last two operations is interchangeable. The learning of the filters of the convolution layers as well as the weights of the other layers are estimated in a supervised fashion through back-propagation with the stochastic gradient descent rule to update the filters and their weights.

While the feature representation of this approach is performed by the convolutional layers, the classifier consists of one (or more) fully connected layer with n output values (in which n is the number of classes), and a multinomial logistic regression layer. Multinomial logistic regression is a classification method that generalizes logistic regression to multi-class problems by softmax function. In particular, we use the CUDA-convnet library implemented in C++/CUDA by Alex Krizhevsky [3][1] to evaluate the LF approach.

2.2 Architecture Optimization (AO)

Each convolutional layer of a CNN is defined by n hyperparameters such the number and size of the filters related to the convolution operation, among others [8]. Considering c convolutional layers and the possible values of each of the $n \times c$ hyperparameters (the *search space*), there are a large number of possible architectures to be evaluated. An effective way to explore such *search space* is by employing the AO approach as described in [5], in which some few thousands of randomly generated architectures are evaluated and the best found convolutional topology is returned, this process is illustrated in Fig. 1.

To make the AO practical, the filter weights used to evaluate the candidate architectures are randomly generated from a uniform distribution $U(0, 1)$ and normalized to zero mean and unit variance [5]. The prediction is performed by linear support vector machines (SVMs).

[1] https://code.google.com/p/cuda-convnet/

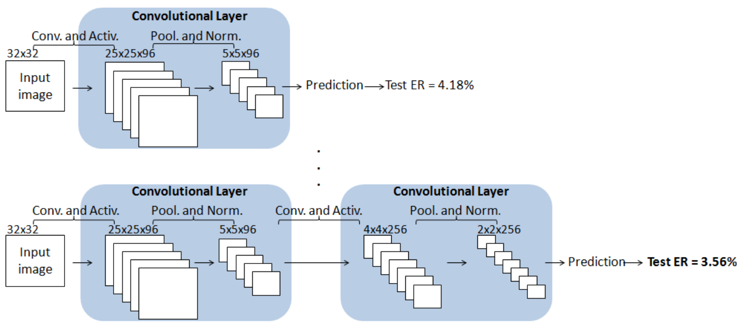

Fig. 1. AO: randomly generates thousands of architectures, evaluates the candidate architectures and returns the best convolutional architecture found.

Table 1. Hyperparameter values of the best architectures.

Operation	convolution		pooling		normalization
CNN	filters	size	size	stride	size
AO-SVHN	{64, 128}	{5, 3}	{3, 3}	{2, 1}	{3, -}
LF-SVHN	{64, 128, 192}	{3, 3, 3}	{3, 3, 3}	{2, 2, 2}	{9, 9, 9}
AO-CAPTCHA	{192, 192}	{3, 3}	{3, 5}	{2, 1}	{-, -}
LF-CAPTCHA	{32, 64}	{5, 5}	{3, 3}	{2, 2}	{9, 9}

3 Experimental Results

In this section, we present the experiments undertaken to evaluated the proposed approaches for noisy character recognition. After describing the used datasets, we present and analyze the results obtained.

3.1 Datasets

The Street View House Numbers Dataset. This image dataset consists of 73,257 digits for training, 26,032 digits for testing, and 531,131 additional digits for training. Here, we combine the training and additional sets for training. The digit images are organized into 10 classes and available in two formats, as follows: 1) original images with character level bounding boxes; and 2) images centered around a single character (cropped digits). We use the latter format as illustrated on the left of Fig. 2[2].

CAPTCHA Dataset. This dataset consists of 12,000 images of CAPTCHAs composed of 6 characters. The characters are segmented by fixed position in the images and they belong to 36 classes (26 upper case letters and 10 digits). It is divided in training and testing sets of 8k/48k and 2k/12k CAPTCHAs/characters, respectively. Two samples of this database are shown on the right in Figure 2. This dataset is currently available for research purposes upon request.

[2] http://ufldl.stanford.edu/housenumbers/

Table 2. AO results: evaluating the training set size to learn the final linear SVM.

CN	# training per class	# training total	# testing images	test error rate (%)
AO-SVHN	100	1,000	26,000	24.28
AO-SVHN	560	5,600	26,000	17.13
AO-SVHN	2,000	20,000	26,000	14.03
AO-CAPTCHA	100	3,600	12,000	17.85
AO-CAPTCHA	560	20,160	12,000	5.45

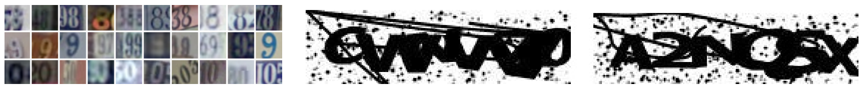

Fig. 2. Samples of the datasets: one from SVHN and two from CAPTCHAs

3.2 Experiments

We now evaluate the effectiveness of the LF and AO approaches described in Section 2 for character recognition. We execute experiments applying these approaches in the SVHN and CAPTCHA datasets. More specifically, we present results for the use of locally-connected layers in CNN and the use of data augmentation in the learning process. We also compare the effectiveness of our deep learning-based approach with others based on hand-designed descriptors.

Architecture Optimization. Due to the memory space required for both feature representation and the extra space required by the linear SVM for learning in the AO approach, it was necessary to restrict the size of training set for both datasets, SVHN and CAPTCHA, to 100 samples per class, such that the best convolutional architectures, *i.e.*, AO-SVHN and AO-CAPTCHA, respectively, could be obtained in dozens of processing hours. The hyperparameter values of the AO-SVHN and AO-CAPTCHA CNNs are detailed in Table 1, in which the value x^i in tuple $\{x^1, x^2, ..., x^n\}$ corresponds to the hyperparameter value of convolutional layer i and the absence of any operation in the i-th convolution layer is representing by '-'. Note that the input image size in both datasets is always 32×32 pixels.

Our first experiment evaluates the impact in the accuracy when varying the training set size to learn the linear SVM, once the architecture optimization has been performed using few samples per class due to memory constraints[3]. These results are presented in Table 2. It is observed that the error rate decreases in the SVHN and CAPTCHA datasets as the number of samples per class increases.

Learning Filters. Based on standard publicly convolutional architectures available in the CUDA-convnet library and on our knowledge domain, we define and evaluate various architectures based on LF approach. To train our CNNs, we fol-

[3] Our system is a Intel i7 Core, 32Gb RAM with NVIDIA GTX Geforce Titan Black GPU and the largest training set possible to be evaluated was around 20k samples.

lowed the methodology of CUDA-convnet library[4]. Those performing better in the SVHN (LF-SVHN) and CAPTCHA (LF-CAPTCHA) datasets are detailed in Table 1. The obtained test error rates are 4.29% and 3.12% for the SVHN and CAPTCHA datasets, respectively.

Locally-connected Layer. Locally-connected layer is similar to a convolutional layer but the former does not share weights as the latter does. That is, a different set of filter weights is applied to each portion of the input map/image, which can help to better discriminate that particular region of the character image, since different regions of a character have different local statistics. As the CUDA-convnet library supports this type of operation/layer, we investigated the influence on the effectiveness by inserting one, two, and three locally-connected layers in a CN, atop the last convolution layer. Table 3 reports the test error rate (ER) obtained, in which L corresponds to the number of locally-connected layers added. The smallest error rates were found for $L = 2$ and $L = 1$ in the SVHN and CAPTCHA datasets, respectively. Hereinafter, we call these architectures LF-LL-SVHN and LF-L-CAPTCHA.

Table 3. Test Error Rates (ER) related to the number of locally-connected layers added (L).

CNN	L	Test ER. (%)
LF-SVHN	1	3.95
LF-SVHN	2	3.84
LF-SVHN	3	3.95
LF-CAPTCHA	1	2.73
LF-CAPTCHA	2	2.91
LF-CAPTCHA	3	2.98

Table 4. Data Augmentation. Test Error Rate (ER) for different values of crop borders.

CNN	*crop_border*	Test ER (%)
LF-LL-SVHN	1	2.75
LF-LL-SVHN	2	**2.55**
LF-LL-SVHN	4	2.75
LF-L-CAPTCHA	1	**2.64**
LF-L-CAPTCHA	2	2.91
LF-L-CAPTCHA	4	4.01

Table 5. Comparison of the proposed CNNs with classifiers based on hand-designed descriptors

Dataset	Test Error Rates (%)			
	Oblique Random Forest	Linear SVM	RBF SVM	Proposed CNN
SVHN	21.9	43.5	39.7	2.55
CAPTCHA	13.8	37.6	24.2	2.64

Data Augmentation. The data augmentation technique strikes overfitting by artificially expanding the dataset. Our data augmentation consists of generating horizontally translated images from the original images. Let I be an image of $n \times n$ pixels. From I, squared sub-regions (and their horizontal reflections) of $n - 2 \times \textit{crop\_border}$ order are generated, in which $2 \times \textit{crop\_border}$ specifies the width of the cropped region in pixels. We investigated the influence of data augmentation on the effectiveness of the LF-LLL-SVHN and LF-L-CAPTCHA

[4] https://code.google.com/p/cuda-convnet/wiki/Methodology

by varying the *crop_border* parameter. According to Table 4, different values of *crop_border* obtain better results for the two datasets.

Comparison with other Approaches based on Hand-designed Descriptors. To compare the deep learning approaches with other approaches based on hand-designed descriptors, we perform experiments in both datasets using the hand-designed descriptor *Histograms-of-Oriented-Gradients* (HOG) [1] with 8 bins and 4 different cell sizes: 16×16, 8×8, 4×8 and 8×4. All cell have stride set up to 50%. To the prediction, we use three classifiers. The first is the *Oblique Random Forest* (oRF) method with SVM on each node [6]. The oRF is an approach that utilizes another classifier to split the data on each nodes of its trees instead of perform using just one feature as custom Random Forest does. The other two classifiers are SVM with a Linear kernel and SVM with a RBF kernels. The results are reported in Table 5. The smallest error rates are obtained using the oRF classifier, but its effectiveness are far away from the ones obtained by the CNNs using learning filters.

Another important topic to be described is the processing time to perform these classifications. In the case of the Oblique Random Forest with a SVM on each node, which proved to be the best classifier of all three baselines, it takes more than twenty hours to execute the learning phase in both datasets, reaching up to fourteen hours in the case of the SVHN dataset. All SVMs utilized on each node are Linear and the parameter C is 0.01, which was determined empirically. The linear SVM has a learning-time of 1 hour to SVHN dataset and about 30 minutes to the other dataset. The SVM with RBF kernel executes the learning in three hours in SVHN dataset, ninety minutes in the Captcha dataset. On the other hand, the CNN LF-LL-SVHN and CNN LF-L-CAPTCHA with data augmentation takes in average nine hours and one hour, respectively, to perform the learning, which is a much smaller time compared to the ORF baseline.

3.3 Discussion

The results show that deep learning approaches are far superior than the ones using hand-designed descriptors in the two datasets of characters evaluated. The LF approach has obtained the smallest error rates in both datasets. We believe that the use of complete and large sets of training samples contributed greatly to this result. For the SVHN dataset, the more complex, the LF-SVHN did not converged during the training (underfitting) when we used training sets of size similar to the ones used in AO (100, 560 and 2000 samples per class). When training the LF-CAPTCHA with the first two sets of training (100 and 560 samples per class), we obtained the respective error rates of 29.50% and 4.09% and the AO-CAPTCHA obtained 17.85% and 5.45%, respectively (see Table 2). Note that AO approach was able to use 3.5% and 42% of the entire training sets of the SVHN and CAPTCHA databases, respectively.

4 Conclusions

In this paper we employed deep learning approaches and hand-designed descriptors for character recognition problem in noisy images. In our experiments, we use the CAPTCHA dataset, proposed by us, and the widely used SVHN dataset. Our experiments demonstrated that the most promising results have been obtained by LF approach. Furthermore, the use of data augmentation and the addition of locally-connected layers to CN reduce the error rates.

For future work, we intend to independently evaluate the influence of the classifiers on the results obtained by deep learning representations, since AO employs linear SVM and the classifier used by the LF approach is based on the fully connected layers and softmax regression.

Acknowledgments

The authors thank UFPR, PUCPR, UFOP, UFMG, FAPEMIG, CAPES and CNPq (Grants #428333/2016-8 & # 307010/2014-7) for supporting this work.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Comp. Vision and Pat. Recognition (CVPR), IEEE Conf. vol. 1, pp. 886–893 (2005)
2. Goodfellow, I. J. and others: Multi-digit number recognition from street view imagery using deep CNNs. In: Int. Conf. Learning Representation (2014)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: (NIPS). pp. 1097–1105 (2012)
4. Menotti, D., Chiachia, G., Falcao, A., Oliveira Neto, V.: Vehicle license plate recognition with random convolutional networks. In: Graphics, Patterns and Images (SIBGRAPI), 2014 27th SIBGRAPI Conference on. pp. 298–303 (2014)
5. Menotti, D., et al.: Deep representations for iris, face, and fingerprint spoofing detection. Information Forensics and Security, IEEE Trans. on 10(4), 864–879 (2015)
6. Menze, B.H., et al.: On oblique random forests. In: Machine Learning and Knowledge Discovery in Databases, pp. 453–469. Springer (2011)
7. Netzer, Y., et al.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsup. FL. pp. 1–9 (2011)
8. Pinto, N., et al.: A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS 5(11), e1000579 (2009)
9. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. arXiv preprint arXiv:1503.03832 (2015)
10. Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: (ICPR). pp. 3288–3291 (2012)
11. Vajda, S., Rangoni, Y., Cecotti, H.: Semi-automatic ground truth generation using unsupervised clustering and limited manual labeling: Application to handwritten character recognition. Pattern Recognition Letters 58, 23–28 (2015)
12. Yuan, Y., Mou, L., Lu, X.: Scene recognition by manifold regularized deep learning architecture. IEEE Trans. on Neural Netw. Learn. Syst. pp. 1–12 (2015)
13. Zhang, X., Trmal, J., Povey, D., Khudanpur, S.: Improving deep neural network acoustic models using generalized maxout networks. In: IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP). pp. 215–219 (2014)