Real-Time Handwritten Letters Recognition on an Embedded Computer Using ConvNets

Dennis Núñez Fernández
Universidad Nacional de Ingeniería
Lima, Peru
dnunezf@uni.pe

Sepidehsadat Hosseini
Seoul National University
Seoul, South Korea
sepid@ispl.snu.ac.kr

***Abstract*—This paper describes the design and implementation of a convolutional neural network for 26 handwritten letters recognition on a regular embedded computer. Recognition task is carried out using a customized convolutional neural network, designed to work with low computational resources. Furthermore, training was conducted on the recently published dataset EMNIST. The experimental results show that the proposed neural network achieves an outstanding accuracy rate compared to similar architectures, also, inference shows a fast response time on a Raspberry Pi 3 board.**

***Index Terms*—ConvNets, Convolutional Neural Networks, Embedded Computer, Human-Machine Interface.**

I. Introduction

Real-world tasks such as control of touchless screens, interaction with robots, digitization of texts and more, directly or indirectly depend on efficient handwritten letter recognition systems. For this reason, recognition of these characters has been spreadly studied during the last decades. But only few works implemented handwritten letters recognition by using the new EMNIST dataset. Works related to this topic usually focus on handwritten digits [1] [2] and not on handwritten letters since these are composed by more characters.

In this work, we propose a real-time handwritten recognition system to work with low computational resources and low power consumption. In order to achieve these requirements, we implemented a customized ConvNet with few layers and few parameters. This architecture is very important since power consumption, time response and recognition rate depend mainly on it. In the following sections we will explain in detail this architecture as well as how the whole system works.

II. Related Work

Early works on handwritten character recognition are based on hand-crafted extracted features [3] and SVM classifiers. For instance, [4] uses Fourier descriptors and achieves an accuracy of 86.66% on the Chars74K dataset. However, in recent years, Deep Learning techniques has been shown excellent results on characters recognition. As evidence of this, [5] accomplishes an accuracy of 96.87% on the CASIA-OLHWDB1.1 dataset for Chinese character recognition using convolutional neural networks (ConvNets) based on domain-specific knowledge. Even, more recent works [6] [7] propose application of Gabor filters in pre-processing step (before the ConvNet) to improve the recognition rate of the whole system.

III. Proposed Method

A. Overview

The proposed system for handwritten letters recognition works in real-time with RGB images captured from a camera. Additionally, this system was designed to be executed on embedded computers with low computational resources, without GPU support and using a low power consumption. Computational and timing requirements of this system highly depend on the ConvNet, therefore, it was designed with a customized architecture.

The selected embedded platform for this recognition system was the Raspberry Pi 3 since its versatility and its low cost make of this a perfect option. The system works on this embedded computer as follows: First, the camera captures RGB images of 400x300 pixels at 30 frames per second. Then, the image is transformed to a binary image, thresholding the strokes of the characters. Later, the region that contains the handwritten letter is extracted and adjusted in order to be used as input of the ConvNet. Finally, the result of the ConvNet is shown in the screen. Fig. 1 depicts the system.

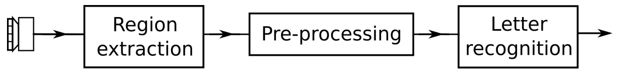


Fig. 1. Block diagram of the proposed recognition system.

B. Region Extraction and Pre-Processing

Separate the handwritten shapes from the background is the first and an important step for the proposed system. In this regard, we apply several image processing techniques that works in the following way: At beginning, handwritten lines are extracted based on color thresholding, so the whole image is converted to a binary image. At this point, the square region that contains the handwritten letter is extracted. Then, morphological opening operations (first erode and then dilate) are applied to the handwritten lines in order to remove small objects from the foreground. Next, morphological closing operations (first dilate and then erode) are applied to remove small holes from the foreground. After these operations, the moments of the transformed images are calculated. Finally, based on these moments, the smallest areas are discarded since these are mainly produced by noise.

C. Handwritten Letters Recognition

The design and implementation of the recognition system has two steps: training and inference. Both steps will be explained in detail in the next paragraphs.

At the training step, the model learns from a dataset based on the back-propagation algorithm [8], wich is a method used to calculate a gradient and needed in the estimation of the weights of the network. In this regard, the deep learning framework used to deploy and train our model was Caffe [9]. This is a framework made with expression, speed, and modularity; these features make of Caffe a good option for our project. As well, since training step demands high computational resources, this is usually performed on computers with powerful GPUs. In this project, training step is conducted on a computer with the next computational resources: Intel Core 7 Octa-Core CPU @3.8 GHz, 12 GB RAM, and GeForce GTX 1050 Ti GPU. After training step, we obtained the .caffemodel file with the trained weights of our model.

Afer the training step, the inference step is carried out. This means that the trained parameters are used to perform classification in real-time. Thereby, we use the .caffemodel file into our real-time recognition system in order to identify different handwritten letters. Because our project focuses on obtain a fast time response using low computational resources, this system was fully implemented in C++ and using OpenCV libraries. Furthermore, the hardware device used to conduct the inference phase was the Raspberry Pi 3 platform, wich has the following computational resources: ARM Cortex A53 Quad-Core CPU @1.2 GHz and 1 GB RAM.

D. Handwritten Letters Dataset

Several datasets can be found in specialized websites, however the EMNIST Dataset (Extended Modified National Institute of Standards and Technology Dataset) [10] is the most remarkable. So, this dataset was used for training and testing our customized deep neural model. The EMNIST Letters Dataset is part of the EMNIST Dataset [10], and is composed by handwritten characters derived from the NIST Special Database [11] and converted to a 28x28 pixel image format. Moreover, this handwritten letters dataset has a total of 145,600 uppercase and lowercase letters divided into 26 balanced classes, see Fig. 2. We used 86% of these images for training and 14% for testing, it means that 124,800 images were used for training and 20,800 images for testing.

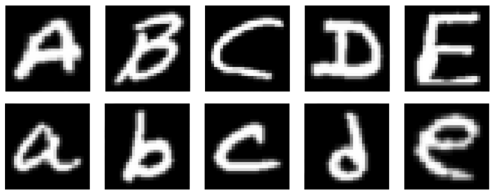


Fig. 2. Samples of letters used for training and testing.

E. Convolutional Neural Network

As explained in the last sections, we focused on a small ConvNet with the aim of work on embedded computers using low computational resources and low power consumption. Since each handwritten letter is composed of strongly different strokes, recognition doesn't need large images and complex ConvNets to extract useful features. In this way, we only use binary images of 28x28 pixels and a small deep neural network with few layers and parameters.

The proposed ConvNet is formed by two convolutional layers with kernels of 5x5 size each one, a non linearity (ReLU) activation function and a max-pooling layer after every convolutional layer, and two full-connected (FC) layers of 500 neurons lenght followed by a final 26-way softmax. Furthermore, this ConvNet is composed by only 60K learnable parameters. This number of parameters is significantly less than the AlexNet network (60M learnable parameters and 650,000 neurons) [12] and the GoogleNet network (6.8M learnable parameters) [13]. The architecture used for handwritten letter recognition is presented in Fig. 3.

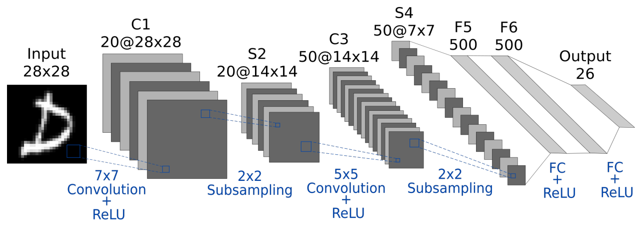


Fig. 3. Architecture of the proposed ConvNet.

The most important components of a ConvNet are the convolutional layers, which detect dominant features presented in input images. In our neural network, the convolutional kernels of the first layer show characteristic horizontal/vertical/diagonal grayscale features of the handwritten letters, see Fig. 4. As well, max-pooling is applied after each convolutional and full connected layers to add translation invariance, which help with position independence. Since cascade linear convolutions adresses to a linear system, a ReLU activation function is added after each convolutional and full connected layer to add non linearity to the network. This neural network considers also dropout regularization to selectively ignore some neurons during training step, avoiding in this way overfitting of the model. Finally, the outputs of the last convolutional layer are flattened and connected to the output layer through two fully-connected layers of 500 neurons length each one.

F. Training Step

We trained our proposed ConvNet using 124,800 images (4,800 images per class), and testing on 20,800 images (800 images per class). As mentioned in previous sections, training was performed on a Nvidia GeForce GTX 1050 Ti GPU. The training itself is carried out using the Caffe framework through a series of forward-propagation and error back-propagation

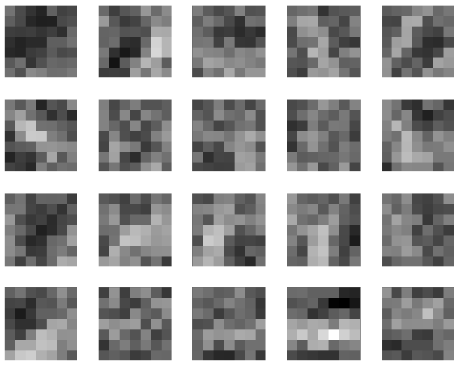


Fig. 4. Learned convolutional kernels at the first convolutional layer.

mechanisms in order to adjust the weights of the ConvNet. In this way, training uses the learning rate to control how much we adjust the weights of our ConvNet with respect to loss gradient. Therefore, training is conducted based on several pre-defined parameters, see Table I, that define the variation of this learning rate as follows: The training process starts with a learning rate equals to the base learning rate (base_lr) and changes over time using 'Inverse Decay' learning policy. It means that the learning rate descreases as a function of the iteration number (iter): $base\_lr * (1 + gamma * iter)^{-power}$. Since training consists of adjust the weights of the ConvNet, the momentum indicates how much of the previous weight will be retained in the new calculation.

TABLE I
TRAINING PARAMETERS

Parameter	Value
Learning policy	Inverse Decay
Base learning rate	0.01
Gamma	0.0001
Power	0.75
Momentum	0.9
Batch size	128
Max. iterations	5,000
Epoches	5.1

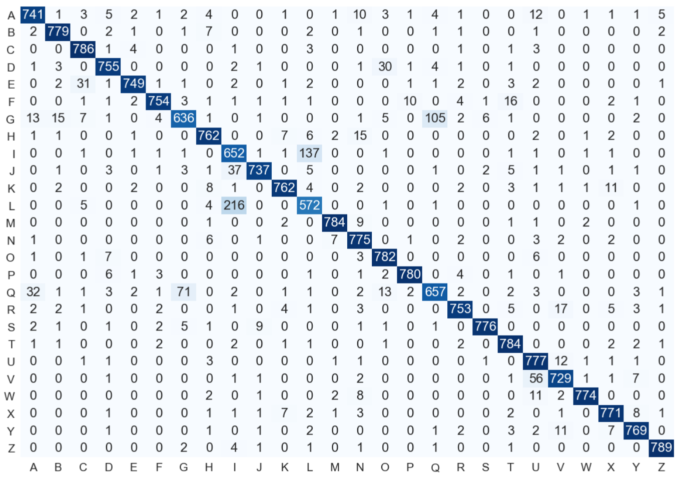
IV. EXPERIMENTAL RESULTS

A. Experimental Results of the Model

After training our model on the EMNIST Letter Dataset, we evaluated its performance on the testing images, which were extracted from EMNIST Letter Dataset, using several consecutive iterations and analyzing the confusion matrix.

First, since each training process of the ConvNet initializes with random weights, the accuracy of every training iteration has a different value. Thereby, in order to have accurate performance metrics of the designed model, we evaluated the model on 40 continuous iterations. After that, our architecture for handwritten recognition shows an outstanding maximum accuracy of 94.2%, an average accuracy rate of 93.4% and a standard deviation of 0.27.

Another useful performance metric is the confusion matrix. This presents a visualization of the misclassified clases and helps to add more training images in order to improve the model. The confusion matrix of our model is showed in Fig. 5 and discloses which letters are misclassified. These errors of classification are mainly because by shape similarities, for instance, letter 'L' sometimes is confused with 'I', 'Q' with 'G', and 'D' with 'O'. In fact, the letters mentioned above are difficult to discern, even for a human.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	741	1	3	5	2	1	2	4	0	0	1	0	1	10	3	1	4	1	0	0	12	0	1	1	1	5
B	2	779	0	2	1	0	1	7	0	0	0	2	0	1	0	0	1	1	0	0	1	0	0	0	0	2
C	0	0	786	1	4	0	0	0	1	0	0	3	0	0	0	0	0	1	0	1	3	0	0	0	0	0
D	1	3	0	755	0	0	0	0	2	1	0	0	0	1	30	1	4	1	0	1	0	0	0	0	0	0
E	0	2	31	1	749	1	1	0	2	0	1	2	0	0	0	1	1	2	0	3	2	0	0	0	0	1
F	0	0	1	1	2	754	3	1	1	1	1	1	0	0	0	10	0	4	1	16	0	0	0	2	1	0
G	13	15	7	1	0	4	636	1	0	1	0	0	0	1	5	0	105	2	6	1	0	0	0	0	2	0
H	1	1	0	0	1	0	0	762	0	0	7	6	2	15	0	0	0	0	0	0	2	0	1	2	0	0
I	0	0	1	0	1	1	1	0	652	1	1	137	0	0	1	0	0	0	0	1	1	0	1	1	0	0
J	0	1	0	3	0	1	3	1	37	737	0	5	0	0	0	0	1	0	2	5	1	1	0	1	1	0
K	0	2	0	0	2	0	0	8	1	0	762	4	0	2	0	0	0	2	0	3	1	1	1	11	0	0
L	0	0	5	0	0	0	0	4	216	0	0	572	0	0	1	0	1	0	0	0	0	0	0	0	1	0
M	0	0	0	0	0	0	0	1	0	0	2	0	784	9	0	0	0	0	0	1	1	0	2	0	0	0
N	1	0	0	0	0	0	0	6	0	1	0	0	7	775	0	1	0	2	0	0	3	2	0	2	0	0
O	1	0	1	7	0	0	0	0	0	0	0	0	0	3	782	0	0	0	0	0	6	0	0	0	0	0
P	0	0	0	6	1	3	0	0	0	0	0	1	0	1	2	780	0	4	0	1	0	1	0	0	0	0
Q	32	1	1	3	2	1	71	0	2	0	1	1	0	2	13	2	657	2	0	2	3	0	0	0	3	1
R	2	2	1	0	0	2	0	0	1	0	4	1	0	3	0	0	0	753	0	5	0	17	0	5	3	1
S	2	1	0	1	0	2	5	1	0	9	0	0	0	1	1	0	1	0	776	0	0	0	0	0	0	0
T	1	1	0	0	0	2	0	0	2	0	1	1	0	0	1	0	0	2	0	784	0	0	0	2	2	1
U	0	0	1	1	0	0	0	3	0	0	0	0	1	1	0	0	0	0	1	0	777	12	1	1	1	0
V	0	0	0	1	0	0	0	0	1	1	0	0	0	2	0	0	0	0	0	1	56	729	1	1	7	0
W	0	0	0	0	0	0	0	2	0	1	0	0	2	8	0	0	0	0	0	0	11	2	774	0	0	0
X	0	0	0	1	0	0	0	1	1	1	7	2	1	3	0	0	0	0	0	2	0	1	0	771	8	1
Y	0	0	0	1	0	0	0	1	0	1	0	2	0	0	0	0	1	2	0	3	2	11	0	7	769	0
Z	0	0	0	0	0	0	2	0	4	1	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	789

Fig. 5. Confusion matrix of the designed model.

The Table II shows the results and a comparison with some previously published results. For the EMNIST dataset, it's difficult to find similar works due its recently publication. However, works carried on a similar dataset, so called NIST dataset, were used as comparative references. In this table we can see a high accuracy rate for the proposed neural network despite the fact that the EMNIST dataset uses uppercase/lowercase letters and the NIST dataset only uses lowercase letters.

TABLE II
AVERAGE ACCURACY OF THE EXPERIMENTS ± STANDARD DEVIATION AND RESULTS FROM THE LITERATURE (%)

	Proposed CNN	**Published results and paper**		
Dataset	EMNIST-26	EMNIST-26	NIST-26	NIST-26
Method	CNN	OPIUMS	CNN	HMM
Accuracy (%)	93.4 ± 0.27	78.0 [10]	92.3 [14]	84.0 [15]

B. Experimental Results of Inference

Once the learned model is obtained using the Caffe framework, we get the the architecture and learned weights on a single .caffemodel file. Then, this file (the information of our trained model) is downloaded into the real-time recognition system, which was implemented on the Raspberry Pi 3 using C++ language. Finally, the overall handwritten letters recognition system is evaluated in real-time under different conditions and perturbations.

The implementation of the proposed recognition system on a common or standard personal computer has no issues due its relatively high computational resources. However, when a recognition system is implemented on an embedded computer like the Raspberry Pi 3 we have two major barriers working against us: restricted RAM memory (only 1 GB) and limited processor speed (four ARM Cortex-A53 @1.2 GHz).

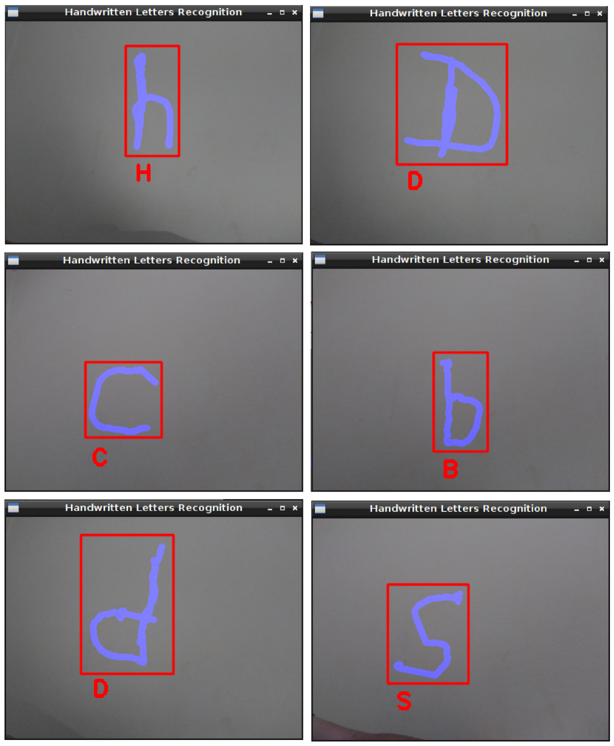


Fig. 6. Results of handwritten letters recognition on a Raspberry Pi 3.

In spite of the processing and memory limitations mentioned above, our real-time recognition system shows promising results during evaluation process. Fig. 6 depicts its performance in real enviroments. As you can observe, the system correctly recognizes different uppercase and lowercase handwritten letters although some shape distortions, low light conditions, and different sizes. In addition to this, we obtain an fast response time of about 21.9 ms (average from 40 iterations) to classify a single handwritten letter.

The Table III shows some details of CNNs tested on the Raspberry Pi 3 platform. As you can see, the proposed CNN achieves the fastest time response by using the lowest power consumption because its simple but efficient design.

TABLE III
RESPONSE TIME AND POWER CONSUMPTION FOR EVALUATION OF DIFFERENT CNN MODELS ON A RASPBERRY PI 3 USING CAFFE

Model	Proposed CNN	VGG_F [16]	NiN [17]	AlexNet [12]	GoogLeNet [13]
Layers	9	13	16	11	27
Power (W.)	0.620	0.760	0.840	0.750	0.790
Time (s.)	0.022	0.857	0.553	1.803	1.175

V. Conclusions

This paper introduces a CNN architecture for handwritten letters recognition on images obtained with a camera. So, this system recognizes 26 letters on an embedded computer with limited computational resources and using low power consumption. The results show an accuracy rate of 93.4% and a response time of 21.9 ms. This demonstrates that simple CNNs are capable to solve relative complex classification problems. In addition, this work is a reference for future projects that make use of the recently published EMNIST dataset or deal with similar recognition tasks.

References

[1] E. Bouvett, O. Casha, I. Grech, M. Cutajar, E. Gatt and J. Micallef, ”An FPGA embedded system architecture for handwritten symbol recognition,” 2012 16th IEEE Mediterranean Electrotechnical Conference, Yasmine Hammamet, 2012, pp. 653-656.

[2] L. B. Saldanha and Ch. Bobda. An embedded system for handwritten digit recognition. J. Syst. Archit. 61, 10 (Nov. 2015), 693-699.

[3] N. Sharma, T. Patnaik, B. Kumar, “Recognition for Handwritten English Letters: A Review”, International Journal of Engineering and Innovative Technology (IJEIT), India, 2013.

[4] A. Gupta, M. Srivastava and C. Mahanta, ”Offline handwritten character recognition using neural network,” 2011 IEEE Int. Conf. on Computer Applications and Industrial Electronics (ICCAIE), Penang, 2011.

[5] W. Yang, L. Jin, Z. Xie, and Z. Feng, ”Improved deep convolutional neural network for online handwritten Chinese character recognition using domain-specific knowledge,” 2015 13th Int. Conf. on Document Analysis and Recognition (ICDAR 2015), Tunis, 2015, pp. 551-555.

[6] S. Hosseini, S.H. Lee, and N.I. Cho, “Feeding Hand-Crafted Features for Enhancing the Performance of Convolutional Neural Networks”, arXiv:1801.07848v1, 2018.

[7] Núñez Fernández D., Kwolek B. Hand Posture Recognition Using Convolutional Neural Network. In: Mendoza M., Velastn S. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2017. Lecture Notes in Computer Science, vol 10657. Springer, Cham.

[8] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Learning representations by back-propagating errors. In Neurocomputing: foundations of research, James A. Anderson and Edward Rosenfeld (Eds.). MIT Press, Cambridge, MA, USA 696-699.

[9] Y. Jia, *et al.*. Caffe: Convolutional Architecture for Fast Feature Embedding. In Proceedings of the 22nd ACM international conference on Multimedia (MM ’14). ACM, New York, NY, USA, 675-678.

[10] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. 2921-2926.

[11] P. J. Grother, K. K. Hanaoka, “NIST Special Database 19 Handprinted Forms and Characters Database”, National Institute of Standards and Technology, USA, 2016.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS’12), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 1. Curran Associates Inc., USA, 1097-1105.

[13] C. Szegedy et al., ”Going deeper with convolutions,” 2015 IEEE Conf. on Computer Vision and Pattern Recogn. (CVPR), Boston, MA, 2015.

[14] D. C. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber, ”Convolutional Neural Network Committees for Handwritten Character Classification,” 2011 International Conference on Document Analysis and Recognition, Beijing, 2011, pp. 1135-1139.

[15] P. R. Cavalin, A. de Souza Britto Jr., F. Bortolozzi, R. Sabourin, and L. E. S. de Oliveira, An implicit segmentation-based method for recognition of handwritten strings of characters. in SAC06, 2006, pp. 836840.

[16] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In Proceedings of the British Machine Vision Conference 2014, pages 6.16.12. British Machine Vision Association, 2014.

[17] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. In International Conference on Learning Representations (ICLR) 2014.