Multiple-Classifier Fusion Using Spatial Features for Partially Occluded Handwritten Digit Recognition

Hieu M. Le, An T. Duong, and Son T. Tran

University of Science, Ho Chi Minh City, Vietnam
{0812141,0912019}@student.hcmus.edu.vn, ttson@fit.hcmus.edu.vn

Abstract. The subject of handwritten digit recognition is a great concern and has many applications in various fields. Although highly restricted forms of digit recognition are widely utilized, reading incomplete and occluded digit image is still a challenge for both academia and industries. In this paper, we attack the problems of recognizing occluded handwritten digits by finding the influence of small patches in digit images to the recognition results. We apply one-hidden-layer neural networks to train and validate each patch independently and enhance the performance of each classifier by the results of its correlated patches. This method allows us to restrict the effect of false information solely into areas that small patches lay on and then correct recognition results by their neighbors. The result of the proposed method shows a noticeable improvement in the stable ability of recognition model with different kinds of simulated distortions.

Keywords: Digit Recognition, Neural Network, Denoising, Particially Occluded Handwritten Digit, Spatial Feature.

1 Introduction

Reading digits in natural images is a hard computer vision issue which is central to a variety of applications such as reading house numbers from outdoor photos or reading zip codes on letters. With a large number of applications, the problem of recognizing characters has been widely addressed by both academia and industries [1].

The most famous benchmark for digit recognition is MNIST [2]. Since the first time it has been introduced, it also had been tested with a series of state-of-the-art algorithms of linear and non-linear classifiers such as adaboost [4], k-nearest-neighbor [5], SVM [6], neural network [8] [9] [11] and convolutional nets [12] [13]. As a result, the lowest error rate recorded with MNIST is only 0.35 by Ciresan [14].

The achievement of almost completely solving MNIST and the more difficult challenges in practical situations lead to the development of SVHN dataset [3] which is more complex, diverse and full of challenge. Creating SVHN dataset

M. Kamel and A. Campilho (Eds.): ICIAR 2013, LNCS 7950, pp. 124–132, 2013.

is an effort of human to reflect the whole complex natural scenes with non-contrasting backgrounds, low resolution, de-focused and motion-blurred images and large illumination differences.

In this work, we do not try to break the record but focus on solving the problem of recognizing occluded digit images. What we want to achieve is a classifier model that can well-perform with distortion - regardless of kind and intensity but only occur on a part of input. For example, the house numbers that are partly covered by leaves or the post codes that are blurred a corner.

This paper is organized as follows. We first survey some related works in digit recognition that inform our approach in section II. We describe the learning architecture that we propose in section III. Section IV will present our experimental results and Section V is our conclusion.

2 Related Work

Using neural network to solve digit recognition problems has been proved to be effective via many experiments. Indeed, artificial neural networks or Multilayer Perceptrons were among the first classifiers tested on MNIST. However, only very limited neural network models were used at that time. One typical example is neural network of LeCun [15] with only 2 or 3 layers and under 800 hidden units. This model achieved an error rate of 1.6 percent in 1998. A more recent neural network with a single hidden layer of 800 units of Simard [9] achieved an error rate of 0.7 percent. Ranzato [10] with supervised learning and pre-training is also a remarkable method which has achieved 0.39% error rate.

In a recent decade, we have observed a series of record breaking in the accuracy rate of neural network. The 5-layer neural network presented by Salakhutdinov [11] with unsupervised pre-training achieved 1% error rate without domain-specific knowledge. After that, taking advantage of powerful modern computers, Ciresan [14] built 6-layer neural network with the number of hidden units varying from 500 to 2500 officially achieved the highest accuracy rate with MNIST dataset with only 35 false samples which are also really hard to recognize with human.

One particularly concerned issue is handling noise input. Dealing with this, a common strategy of supervised learning algorithms is diversifying the training dataset by adding plenty of different noises such as elastic and affine distortion [15]. Meanwhile, Using PCA [17] and other feature extracting methods such as K-means by Adam [7] are also effective unsupervised learning approaches to reduce computational complexity and noisy effects.

Among other unsupervised learning, denoising autoencoder [16] is a typical algorithm trying to represent the input data into a more stable and general form that still retain substantial information of the input. In fact, the idea of training a multi-layer perceptron for a denoising task is not new. LeCun and Gallinari [18] firstly introduced method to learn an (auto-)associative memory in 1987. The networks were trained and tested on binary input patterns, corrupted by flipping a fraction of input bits chosen at random. Both the model and training procedure in this precursory work are very similar to the denoising autoencoder.

Local shape descriptors such as contour-based and region-based shape representations are also significant algorithms, especially for solving problem of recognizing partially occluded images. By decomposing a whole shape into primitives, the digits are then represented by combining these primitives in a suitable coding way. Specifically, some prominent methods of this approach are chain code, polygon decomposition, syntactic analysis (contour-based), convex hull, and medial axis (region-based). However, they still have some main drawbacks: no formal definition for primitives of a certain shape, complexity of partial matching, no preservation of global characteristics and sensitivity to noise.

Comparing to method listed above, our method using neural network approaches in a slightly different manner. Instead of trying to improve the accuracy of classifier, we use multi-classifiers on sub-patches divided from image and build a correlative model between them to get rid of the misclassification.

3 Model

Our system proceeds in 3 steps: 1) Selecting patches and classifying; 2) Adjusting results and 3) Fusion.

3.1 Selecting Patches and Classifying

Many algorithms processed input images as single objects. Consequently, every single local noise in the input image would affect the accuracy of recognition result. Instead of analyzing the whole image, we first divide input images into smaller patches and consider each of them as a complete input image to recognize. Being treated locally, noise in input images will only change the accuracy of the patches that the noise lay on.

Moreover, our target is creating patches which satisfy three conditions: 1) their size are large enough to contain certain structures of digits; 2) their size are small enough to avoid local noises; 3) their position in image are defined to have some correlative information through overlapped regions of patches, where the ratio of overlapped regions between patches helps to define the influence of recognition results from patches to input image.

With each patch received after division, we use it as the input data to classify by supervised learning algorithms trained independently on each patch. The result of this phase is a set of classified output values of patches.

3.2 Adjusting Results

We then mutually adjust the set of output units by using spatial information of patches. Assuming that each pixel of the images has same amount of information supporting for classifiers, we build a correlated model for patches based on the shared area between them. Apparently, the more area shared by two certain patches, the more similar their classified results should be. By exchanging the output values between neighbors corresponding to the area of regions they share, we adjust the results to harvest a less-affected-by-noise set of results.

3.3 Supervised Learning for Patches Fusion

By dividing initial image into several patches, we also divide information using for classification into smaller groups. Intuitively, for each case of output label, each group has different level of useful information. Simulating this relation, we consider all adjusted results as a new input data and feed it to an one-hidden-layer neural network to get the fusion output. This neural network trained supervisedly on the training set has the role as an election that collect the voting values from patches. As a result, our model yeild the correct answer only if the number of unaffected patches is larger than affected patches. However, recognizing digit images lost more than 50% of their shape is an imposible task even with human's ability.

3.4 Algorithm

Our algorithm can be summarized as follows :

1. Step 1 : The partial supervised learning
   - Select K patches ($m \times m$ pixel) from initial image ($n \times n$ pixel, $n > m$) to yield K input vectors $x_i \in \mathbb{R}^{m^2 \times 1}, i = 1..K$
   - Feed x_i into K independently trained supervised learning models to get K output vectors $y_i \in \mathbb{R}^{10 \times 1}, i = 1..K$ (corresponding to the probability of 10 digits case).
2. Step 2 : Adjusting output vectors
   - Adjust K vectors $y_i \in \mathbb{R}^{10 \times 1}, i = 1..K$ according to the shared areas between them (1) to create K adjusted vector $y'_i \in \mathbb{R}^{10 \times 1}, i = 1..K$.

$$y'_k = \sum_{i}^{m} \left[\frac{S(q_k^i)}{S(P_i)} . r_k^i \right] \tag{1}$$

$$r_k^t = mean(y_{i, q_k^t \subset P_i}) \tag{2}$$

   Where :
   - $P_{k,k=1..K}$: K patches divided from initial image, containing l subpatches $q_k^i, i = 1..l, l < \sum_{i=1}^{K-1} C_{K-1}^i$ which are the regions shared with defined set of patches.
   - r_k^t : evaluated output values for subpatches are equal to average output values of patches sharing subpatches (2).
   - $S(region)$: area of region.
3. Step 3: Fusion supervised learning
   - Combine K adjusted vectors $y'_i \in \mathbb{R}^{10 \times 1}, i = 1..K$ to get vector $Y \in \mathbb{R}^{T \times 1}, T = K \times 10$
   - Feed Y into trained fusion supervised learning model to get 10 final output values and predict the label.

4 Experiments and Results

Our model was tested on MNIST dataset which consists of two datasets, one for training (60000 images) and one for testing (10000 images). One usual approach is using 50000 images for training and 10000 for cross validation. However, we use all of 60000 images for training and then generate new testing datasets by filling noises as *white* or *random* pixel to certain regions in the original testing images. Specifically, we filled value *'1'* or *random* on two kinds of regions : squares k-by-k pixels with (k running from 5 to 12) and horizontal *white* rectangle with its width size equal to the width of image and its height size changing from 4 to 12. (Fig.1)

Fig. 1. Examples of 2 testing datasets: squares k-by-k pixels and horizontal rectangles

In this paper, we use one-hidden neural network with 50 hidden units for partial classifying and fusion (with different input layers). For comparision, we also test two testing datasets mentioned above with state-of-the-art neural network with 2-layer and 300 hidden units per each layer of LeCun [15] which partly shares the same architecture with the model we used.

With the 28-by-28 pixel images of MNIST, we have evaluated various methods to divide the initial space to smaller patches. From the space of 28×28 pixels, we create 9 patches (with 14×14 pixels each) as demonstrate in Fig.2; each of them includes 25% of the whole image's area and has at least 25% shared area with its neighbor patches. From each patch, we yield an input vector $x \in \mathbb{R}^{196}$ and then harvest a set of 9 output vectors $y \in \mathbb{R}^{10}$ which contain 10 output values corresponding to the probability of 10 digit numbers (0..9) by using neural network mentioned above.

In Fig.2, we can see that patch 1, for instance, have 25% of its area shared by 3 other patches (2, 4 and 5), 25% of its area shared with patch 2, 25% with patches 4, and 25% of its area that only belong to its own. Consequently, we have 4 subpatches divided from patch 1 that can be easily calculated their evaluated output values. Likewise, we adjust all 9 received result vectors y by formulas (1) and (2) to get the combined adjusted results vector $Y \in \mathbb{R}^{90}$. We then feed Y into the input layer of three layers neural network (90-50-10) to get 10 output units and predict the label.

Testing with the original testing dataset, our classifier achieved the accuracy rate of 98.1% without the adjusting phase and 97.9% accuracy rate with the

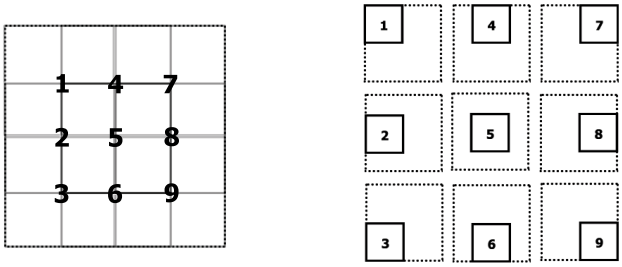

Fig. 2. 9 patches with sizes of 14×14 pixels

whole system. Now, we will present the experimental results of our classifier on MNIST dataset modified.

4.1 Adding a Square k-by-k Pixels

In this experiment, we filled a region of k-by-k pixels in each image in MNIST dataset with value *'1'* or *random*. Then we analyzed the effect of this region to the performance of classifiers. For each value of k, we added noise in every available position of k-by-k region in the original 28×28 pixel area of images in dataset. With $k = 7$, for instance, we have 22×22 available positions to define the region 7×7 pixel. As such, for the whole 10000 images of dataset, we create a new testing dataset containing $22 \times 22 \times 10000$ images.

Fig. 3 summarizes in comparison to 2-layer neural network accuracy rate for each case of k from 5 to 12 when we add value *'1'* or *random* to the selected regions. We see that our method outperforms the traditonal neural network with at least 10% accuracy rate better with any kinds of added noises. On the other hand, the perfomance of normal neural network is worse with *white* noise than *random* noise.

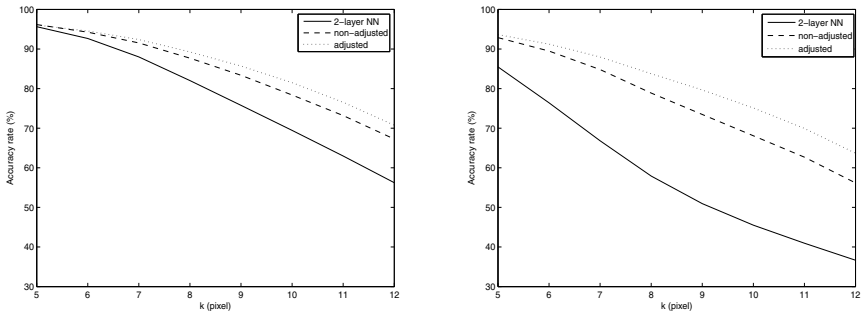

Fig. 3. Accuracy rate of classifiers with k varying from 5 to 12 on two modified dataset (left : *random* noise, right : *white* noise)

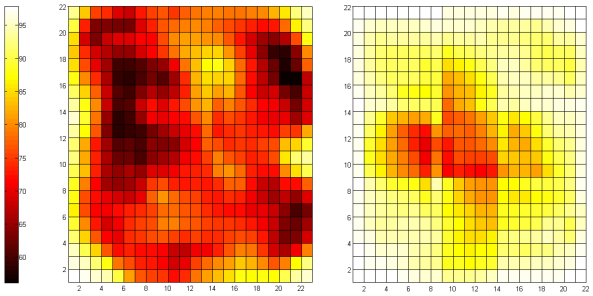

Fig. 4. The accuracy rates of LeCun's neural network (left) and our model (right) in case of adding 7×7 noise regions into 22×22 available positions

Another considerable point is the position of noise regions. Considering human's ability, adding noise right in the center of the images will apparently create a huge difference with adding in the corner. Fig.4 demonstrates the influence of the positions of noise regions on the performance. As can be seen, only noise adding in the center can decrease the recognition ability of our method while traditional neural network's result can be easily reduced with any positions.

4.2 Adding a Horizontal Rectangle

Easily encounted in many real situation, we made experiments with dataset adding horizontal rectangles of false pixels. We fill *white* pixel to rectangles whose height ranges from 5 to 12 pixel and move these rectangles downwards from the top to the bottom of the images. The width of the rectangle is alway equal to that of images. The average accuracy rate is summarized in Fig.5. One among the clearest examples illustrating our result is when filling noise to rectangles that lay from 22th to 28th pixel of images, our method significantly achieved

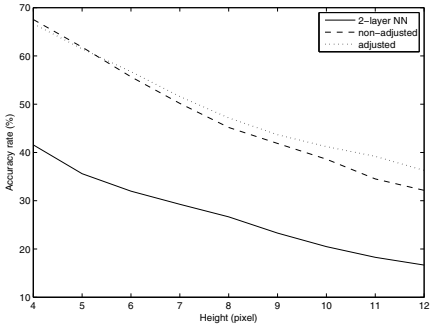

Fig. 5. Average accuracy rates of models testing on dataset filled noise to rectangles whose height from 4 to 12, moving downwards from top to bottom

94.41% accuracy rate and completely outperformed normal neural network with only 37.48% accuracy rate, this testing case can be seen in the right image of Fig.1.

5 Conclusion

In this paper, we propose a hierarchical system of classifiers using spatial features to solve the problem of occluded handwritten digit recognition. The multi-classifiers work independently on sub-patches of input images in training process to avoid the effect of noise. For each handwritten digit number, our system finds the useful information for recognition in each patch. To improve recognition accuracy, we also consider recognition results of neighboring patches and combine them to that of checking patch. Then we utilize the three-layer neural network to fusion all useful information from multi patches for recognition. The results of our experiments show that our proposed method has a high potential to work with this challenge.

Acknowledgment. This work was financially supported by the National Foundation for Science and Technology Development (NAFOSTED), Vietnam.

References

1. Plamondon, R., Srihari, S.N.: On-line and off-line handwriting recognition: A comprehen-sive survey. IEEE Trans. Pattern Anal. Mach. Intell 22(1), 63–84 (2000)
2. MNIST, `http://yann.lecun.com/exdb/mnist/`
3. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading Digits in Natural Images with Unsupervised Feature Learning. In: NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning (2011)
4. Kégl, B., Busa-Fekete, R.: Boosting products of base classifiers. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009), pp. 497–504. ACM, New York (2009)
5. Belongie, S., Malik, J., Puzicha, J.: Shape Matching and Object Recognition Using Shape Contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(24), 509–521 (2002)
6. DeCoste, D., Schoelkopf, B.: Training invariant support vector machines. Machine Learning Journal (MLJ) 46(1-3) (2002)
7. Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D.J., Ng, A.Y.: Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning. In: ICDAR 2011 (2011)
8. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)
9. Simard, P., Steinkraus, D., Platt, J.C.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: ICDAR 2003, pp. 958–962 (2003)
10. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient Learning of Sparse Representations with an Energy-Based Model. In: Platt, J., et al. (eds.) Advances in Neural Information Processing Systems (NIPS 2006), vol. 19. MIT Press (2006)

11. Salakhutdinov, R., Hinton, G.E.: Learning a nonlinear embedding by preserving class neighbourhood structure. AI and Statistics (2007)
12. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column Deep Neural Networks for Image Classification. In: CVPR 2012, pp. 3642–3649 (2012)
13. Lauer, F., Suen, C.Y., Bloch, G.: A trainable feature extractor for handwritten digit recognition. Pattern Recognition 40(6), 1816–1824 (2007)
14. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition, CoRR abs/1003.0358 (2010)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
16. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. J. Mach. Learn. Res. 11, 3371–3408 (2010)
17. Jolliffe, I.T.: Principal Component Analysis, p. 487. Springer (1986)
18. LeCun, Y.: Proceedings of Cognitiva. Paris 85, 599–604 (1985)