Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 138 – 151

doi:10.3906/elk-2001-12

Engraved digit detection using HOG–real AdaBoost and deep neural network

Tuan Linh DANG[1,*], **Thang CAO**[2], **Yukinobu HOSHINO**[3]

[1]Department of Data Communications and Networks, School of Information and Communications Technology, Hanoi University of Science and Technology, Hanoi, Vietnam

[2]Machine Imagination Technology Corporation (MITECH), Tokyo, Japan

[3]School of Systems Engineering, Kochi University of Technology, Kochi, Japan

Received: 03.01.2020 • **Accepted/Published Online:** 11.09.2020 • **Final Version:** 27.01.2021

Abstract: This paper proposes a framework for recognizing sequences of digits engraved on steel plates. These digits are normally blurred, dirty, not clear, tilted, and sometimes overlapped by other digits. Several digits in a string with uneven spacing and different sizes are detected at the same time. The framework consists of two main components called histogram of oriented gradient–real AdaBoost module and deep neural network module. The first component is used to detect digit windows, and the second component is employed to recognize digits inside the detected windows. Experimental results demonstrated that the proposed framework could be a potential solution to recognize the engraved digits.

Key words: Histogram oriented gradient, real AdaBoost, deep neural network, engraved digit recognition

1. Introduction

From the factory warehouse, steel may be automatically classified to be delivered to different customer companies. The classification is based on the engraved digits on the steel. However, during storage, the steel may be degraded, rusted, and the engraved digits may be faded or dirty. In addition, the engraved characters are normally blurred, smudgy, dirty, not clear, tilted, and sometimes overlapped due to engraving mechanisms.

To our knowledge, there is no system to identify rusted or dirty engraved digits on steel with fast processing speed. Previous studies have investigated the digit recognition. However, normally digits recognized in previous research have been clear, easy to read [1–4], or blurred but not dirty, font variation digits [5]. In addition, several papers only recognized each digit separately [1, 6–8]. Some studies have also investigated only the engraved digit in other material [9] or the clear digit engraved on steel [10].

Therefore, this manuscript proposes a framework that may quickly identify rusted or dirty characters on steel. The proposed framework can be applied in actual operation. Moreover, several digits in one piece of steel are recognized at the same time. The appearance order of all digits on the steel needs to be recognized correctly. These digits could be detected correctly even if they are not clear or blurred. The order of the digits in a string can also be detected. For example, if the digits in a string are *15792468*, the recognized results are also *15792468*, even though the detected digits have uneven spacing and different sizes. Our architecture uses two main modules called the histogram of oriented gradient (HOG)–real AdaBoost module and the deep neural network (NN) module. In addition, the results concerning the operating speed of the proposed architecture

*Correspondence: linhdt@soict.hust.edu.vn

were also examined.

In the first component, the HOG is used for the feature extraction phase. The extracted features are fed to the real AdaBoost algorithm to detect windows that consist of the digits. If the string has eight digits, eight rectangular windows will be extracted from the image. Each window represents one digit. After that, the detected windows are sent to the second component, the deep neural network, to be processed. The outputs of the NN will determine which classes the digits inside windows belong to. There are eleven classes corresponding to ten digits from 0 to 9 and "no digit". There are three main mechanisms in the two proposed modules, which are HOG, AdaBoost, and NN, respectively.

This paper is presented as follows. Section 2 presents the proposed architecture. Section 3 shows the experimental results. Section 4 concludes our paper.

2. Engraved digit recognition

2.1. Methodology

Our proposed architecture has two main components called the HOG–real AdaBoost component and the deep neural network component. The HOG algorithm serves as the feature extraction algorithm. After feature extraction, we use the real AdaBoost algorithm to detect windows containing the digits. The goal of this phase is to locate the positions of the digits. Then, the windows detected by the HOG–real AdaBoost component are fed into the DNN to recognize the digits which are from 0 to 9.

The HOG–real AdaBoost component is employed before the DNN component to reduce the input data of the DNN. The input data are exactly a digit window instead of the whole image. On the other hand, an AdaBoost-only approach may yield a low recognition rate and a high false-positive rate [11, 12]. Therefore, the DNN component after the HOG–real AdaBoost component could increase the recognition rate of the real AdaBoost algorithm.

2.2. HOG

In the first module, concerning the feature extraction mechanisms, the HOG is a well-known feature extraction method used in image processing because this method is not affected by geometric and photometric transformations [13]. Regarding the classification techniques, the idea to create a strong classifier using several weak classifiers called the AdaBoost algorithm has also emerged as an exciting topic of research and study [14, 15].

In the feature extraction step, a region of interest (ROI) of an image can be represented by one feature vector that contains gradient information. The ROI is divided into smaller regions called cells. In each cell, the HOG is calculated. In this calculation, the gradient information in each cell is assigned to corresponding bins. This assignation is the data discretization mechanism. Several adjacent cells are grouped as one block. The block is considered a sliding-window detector over the ROI. The histogram of the block which is the concatenation of the histograms of all cells in this block will be normalized. The feature vector is the histogram of all blocks [13].

The operations of the HOG algorithm are as follows [13].

- Step 1: Define the parameters.
  - Define the cell size C_s.
  - Define the block size B_s. The number of pixels in one block equals $C_s \times B_s$.
  - Define the overlapping rate between the blocks.

- Define the ROI size.

- Step 2: Calculate the gradient of each pixel.
  - Calculate the horizontal gradients x_{gra} with vector [–1,0,1].
  - Calculate the vertical gradients y_{gra} with the transpose of above vector.
  - Calculate gradient magnitude using Eq. (1).

$$magnitude = \sqrt{x_{gra} \times y_{gra}} \tag{1}$$

  - Calculate the gradient orientation using Eq. (2).

$$orientation = \arctan(\frac{y_{gra}}{x_{gra}}) \tag{2}$$

  - Assign the gradient information to different bins. The value of each bin is the weighted gradient magnitude.

- Step 3: Create the histogram of each cell using gradient information obtained from the previous step.

- Step 4: Create the histogram of each block. The histogram is represented by a vector. The N_D dimensions of the vector can be shown in Eq. (3).

$$N_D = N_C \times N_B \tag{3}$$

  where N_C is the number of cells inside one block, and N_B is the number of bins.

  The vector is normalized using L_2-norm as Eq. (4).

$$\vec{v} = \frac{\vec{v}}{\sqrt{||\vec{v_2}||^2 + \alpha}} \tag{4}$$

  where $\vec{v}$ is nonnormalized feature vector of the block, α is a constant number, $||\vec{v}_2||$ is the two-norm of vector $\vec{v}$.

- Step 5: The final feature vector of ROI is the concatenation of the vector $\vec{v}$ of all blocks. The dimensions of the HOG feature vector equals the number of blocks $\times$ the number of cells $\times$ number of bins.

The HOG calculation is shown in Figure 1.

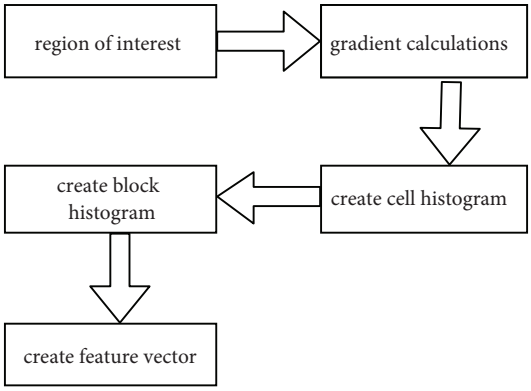

Figure 1. The operation of the HOG feature extraction.

2.3. Real AdaBoost

An ROI window, represented by the HOG feature vector will be classified as a window containing a whole digit or not by the real AdaBoost algorithm. Each image of 8 digits was processed eight times by HOG–real AdaBoost to detect each engraved digit. The idea of the original AdaBoost algorithm is to create a robust classifier based on the combination of weak classifiers. The real AdaBoost algorithm was also proposed to improve the AdaBoost algorithm since the outputs of the weak classifiers in the original AdaBoost algorithm have only Boolean values for results. In real AdaBoost, the outputs of the weak classifiers are real numbers which denote the probabilities of each class. Therefore, the classification border between classes could obtain better accuracy, leading to better recognition rate [16–19].

The operations of the Real AdaBoost are presented as follows [16–19].

- Step 1: In the initial phase, the m_weight of each feature in the feature vector is generated as Eq. (5). This generation is the distribution of each feature.

$$m\_weight(i) = \frac{1}{N} \tag{5}$$

  where N is the number of learning sample, i is the i^{th} feature.

- Step 2: The feature vector is partitioned into S sections. At each iteration, the best element of the vector that has a minimum error is selected. In other words, the optimal classifier at this iteration is determined. The best element is calculated using the real AdaBoost algorithm as follows.

  – For each feature, the probability density distributions W_+^j and W_-^j are calculated as can be seen in Eqs. (6) and (7).

$$w_{t+}^j = \sum_{i:x(i)\epsilon S_j \Lambda y(i)=1} m\_weight_t(i) \tag{6}$$

$$w_{t-}^j = \sum_{i:x(i)\epsilon S_j \Lambda y(i)=-1} m\_weight_t(i) \tag{7}$$

where j is the partition index ($1 \leq j \leq S$), ($x(i)$, $y(i)$) is one training sample ($1 \leq i \leq N$), $y(i)$ = 1 if ($x(i)$, $y(i)$) is the positive training sample, $y(i)$ = -1 if ($x(i)$, $y(i)$) is the negative sample, t is the t^{th} iteration.

– From calculated W_+^j and W_-^j, the normalized vector for each feature is calculated as shown in Eq. (8).

$$Z = 2\sum_j \sqrt{w_{t+}^j w_{t-}^j} \tag{8}$$

– The value of Z indicates the separation degree between the distribution of the negative samples and the distribution of the positive samples. The feature f which holds the lowest Z has the highest separation. This f feature becomes the best element of feature vector at this iteration, and this f feature along with its W_+^j and W_-^j will be stored. The output of the classifier using feature f at time t can be seen in Eq. (9).

$$h_t(f) = \frac{1}{2}\ln(\frac{W_+^j + \alpha}{W_-^j + \alpha}) \tag{9}$$

where α is a random number.

- Step 3: Update the weight m_weight at iteration $(t+1)$ as Eq. (10).

$$m\_weight_{t+1}(i) = m\_weight_t \times e^{-y(i)h_t(x(i))} \tag{10}$$

m_weight_t is the distribution at iteration t of $x(i)$.

- Step 4: Repeat from Step 2 until the number of iterations is reached.

2.4. Deep neural network

The second module consists of the NN which was introduced to represent a human brain [20]. Nowadays, the NN is widely used in different fields. The application of the NN can be seen in medicine [21], power systems [22], finance [23], and monitoring [24]. However, in a conventional fully connected NN, each node in one layer connects to all nodes in the previous layer and the next layer [20]. This type of NN requires many parameters concerning the weights and biases when it has a significant number of nodes. A new type of NN called deep NN (DNN) was invented to solve the problem of the parameters [25, 26].

The AdaBoost algorithm normally obtains a low recognition rate [11, 12], and does not meet the requirements of recognizing engraved digits. We improve the recognition rate by using DNN after locating the digit position by the HOG–real AdaBoost.

Detected ROI windows from the HOG–real AdaBoost are fed into a deep neural network (DNN) to recognize which digit is in the ROIs. Our approach uses a convolutional neural network (CNN) as the DNN.

Although fully connected NNs have been widely used, this type of NN requires many parameters concerning the weights and biases. CNN is used to overcome such parameter issues. Compared to a fully connected NN, a CNN has three main differences. The first one is the use of a local receptive field. In a fully connected

NN, all nodes in the first hidden layer connect to every input. On the other hand, each node in the first hidden layer of a CNN only connects to a small part of the inputs. The connected part is called the local receptive field of one hidden node. The local receptive field will be slid to the right to use the next hidden node. Similar to a fully connected NN, the connections inside a CNN have weights, and each hidden node has a bias. Using the local receptive field can reduce the NN parameters dramatically.

The second difference is the appearance of shared weights. In this situation, the hidden nodes have the same weights and biases. Thus, all nodes in the hidden layers detect the same feature of an input image. The feature is the input pattern which activates the hidden nodes. Each set of shared weights and biases detects one feature. To process an image, a CNN may consist of several sets to detect or recognize different features.

The third difference is the presence of a pooling layer which reduces the number of weights and biases. In other words, the pooling layer decreases the size of the outputs from the previous layer before sending the reduced data to the next layer. For example, the pooling layer uses a 2×2 filter and executes the max-pooling. In this situation, the input of the pooling layer will be slid by a 2×2 window. The output of each sliding window which is the maximum value of the 2×2 region will be sent to the nodes in the next layer.

The details of these differences can be seen in previous papers [25, 26].

2.5. Operations of the proposed architecture

2.5.1. Operation of the HOG–real AdaBoost component

The operations of both the training and testing phases in the HOG–real AdaBoost component are illustrated in Figure 2.

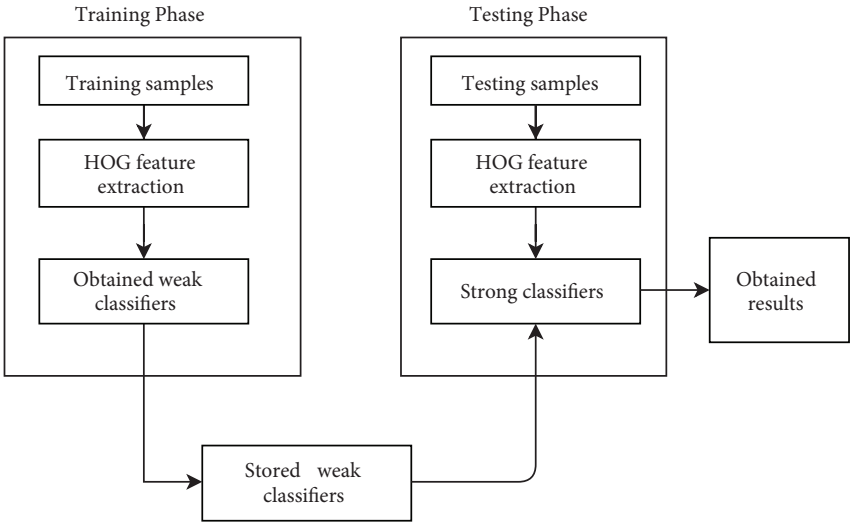

Figure 2. The operation of the HOG–real AdaBoost algorithm.

In the training phase, the HOG feature in each training image is calculated, and the HOG feature vector is created. The operations of the HOG algorithm are presented in Section 2.2. The real AdaBoost algorithm processes the obtained HOG feature vector. The real AdaBoost is described in Section 2.3. Finishing the training phase, the final W_+^j and W_-^j are stored. The calculations of W_+^j and W_-^j are shown in Eqs. (6) and (7).

In the testing phase, the HOG features of the testing images are also generated similar to the training phase. The learned feature list W_+^j and W_-^j from the stored list will be loaded to calculate h_t according to Eq. (9). A strong classifier is created by summing up the h_t for every input $x(i)$ based on Eq. (11). If the obtained result $H(x(i))$ exceeds a defined threshold, the sample is considered a positive sample.

$$H(x(i)) = \sum_t h_t(x(i)) \tag{11}$$

where t is t^{th} iteration.

To locate a one-digit window in an ROI, the testing phase uses the meanshift algorithm. The idea behind the meanshift algorithm is to shift one data point to the center of the closest cluster. The direction to the closest cluster is the direction that has most of the nearby points [27, 28]. The meanshift algorithm used in our program is as follows.

- Put center points of all detected positive windows into the inputs of the meanshift algorithm.
- Execute the meanshift algorithm. Finishing the algorithm, the new center of points are found. The old center points will be removed. Let the number of the obtained center points be *tcount*.
- For each point of *tcount* points inside a defined radius, extract one rectangle.

  The rectangle which has the highest number of matched points is the new detected window.

2.5.2. Operation of the DNN component

The windows detected by the HOG–real AdaBoost are sent to a CNN. We use the AlexNet model that contains eight layers (five convolutional and three fully connected layers) [29]. The details of our AlexNet model can be seen in Figure 3. Each rectangular in this figure represents one layer. The "conv" means the convolutional layer and the "FC" means the fully connected layer. "11×11", "5×5", and "3×3" are the convolution sizes, and stride is the movement of the filter. Another number is the output number of this layer. For example, the first convolutional layer has a filter size of "11×11", 96 outputs, and a stride of 4. The eighth layer (final layer) is the fully connected layer that has 11 outputs. Training data are labeled positive images of ten numbers (from number 0 to number 9) and negative images that do not contain any number. The images are gathered and resized to the same size as the detected windows. The CNN has eleven outputs corresponding to eleven classes (ten positive classes and one negative class). The operations of the proposed system can be illustrated in Figure 4.

3. Experiments

3.1. Engraved digit image data

The experiments in our paper were conducted on the image data provided by a Japanese manufacturer. An image has eight steel engraved digits. The dataset consists of 1493 testing images and 2630 training images.

The confidentiality of the data must be protected by a privacy agreement. Therefore, several images that are similar to our obtained data are presented in Figures 5–7.

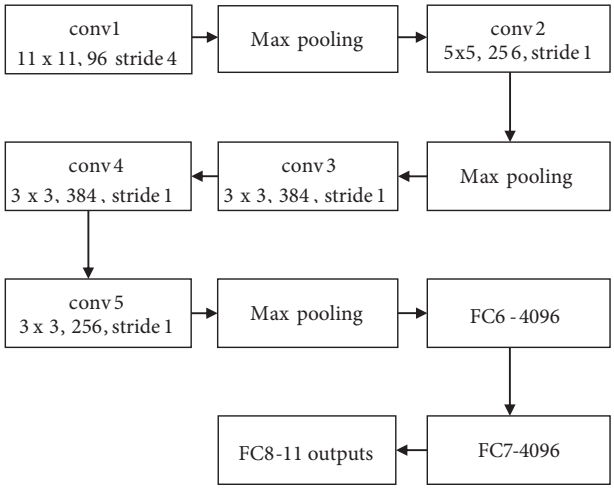

Figure 3. The AlexNet model.

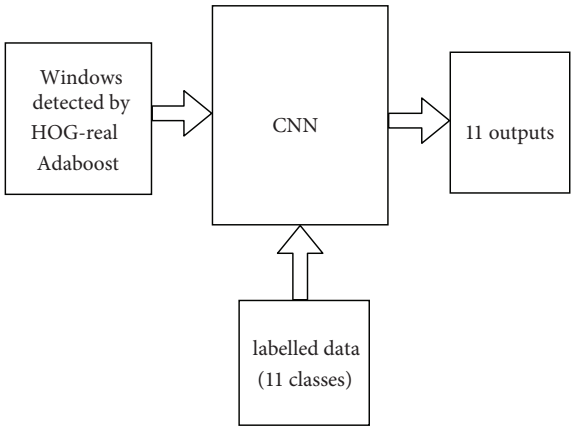

Figure 4. The operations of the CNN in the proposed architecture.

3.2. Parameters

For the HOG algorithm, the cell size C_s is 8×8 pixels. The block size has 2×2 cells. We quantized the gradient orientation (direction) into nine bins. In the normalization step of the HOG, $\alpha = 1$ in Eq. (4). In the real AdaBoost algorithm, $\alpha = 0.0000000001$ in Eq. (9). The original window size of the digits was 40×50 pixels.

The configurations of the AlexNet model were used in our CNN [29] except the number of outputs. Therefore, Our NN had five convolutional layers and three fully connected layers. The size of the convolutions in each layer, the output in each layer, and the movement of the filter (NN stride) can be seen in Figure 3. In our CNN, 11 outputs correspond to 10 positive classes (from number 0 to number 9) and one negative class.

The HOG–real AdaBoost was experimented in a computer powered by Intel Xeon X5650 2.67GHz CPU and GCC 4.8.5 20150623 compiler. the CNN was trained with NVIDIA GeForce GTX 1060 graphics processing unit (GPU).

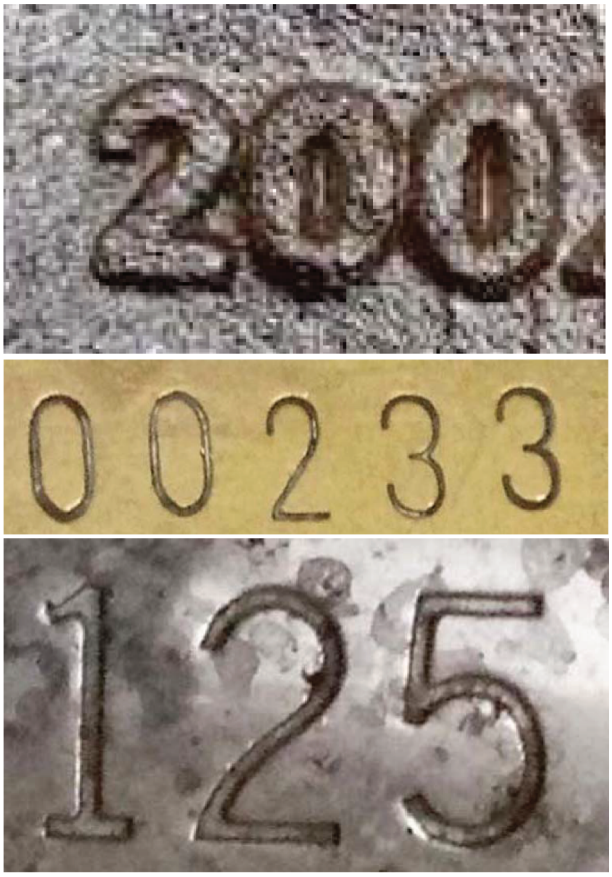

Figure 5. Examples of engraved digits.

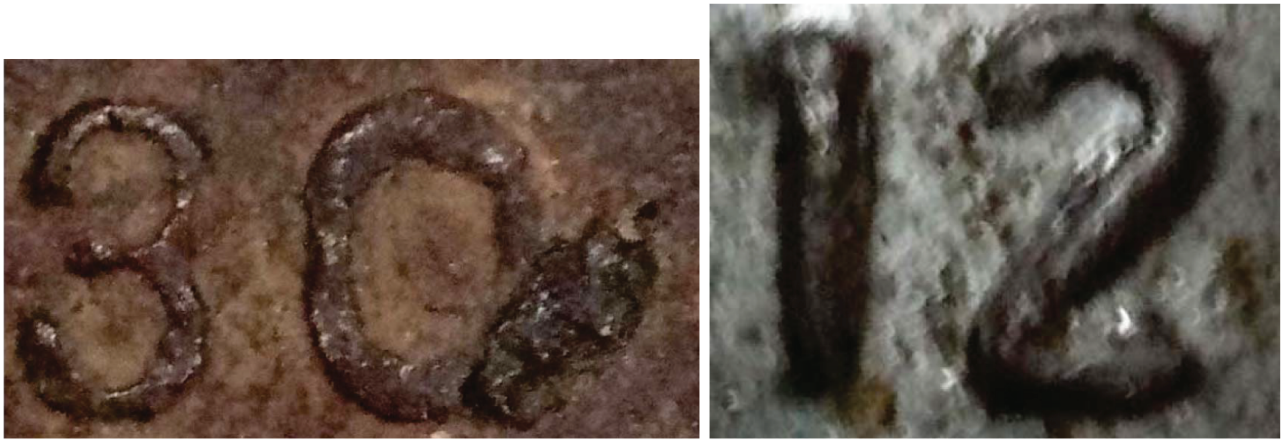

Figure 6. Examples of engraved digits which are dirty.

3.3. Results

In our experiments, the digits in one image were only considered a correct recognition if all the digits were recognized correctly. If any digit was wrong, the image became incorrectly recognized.

Figure 7. Example of engraved digit which is blurred.

The recognition rate of the engraved digits is shown in Table 1. Our architecture could recognize 92.97% of the investigated digit strings.

Table 1. The recognition rate of the engraved digits.

Correct	Wrong	Recognition rate
1388	105	92.97%

After recognizing all the digits in one image, we also examined the recognition of separated digits. The digits were divided into three different categories called "not difficult to see", "very blurred", and "very dirty", respectively. The first set was the digits which were not hard to see. The second set was the digits which were highly blurred. The third set contained the very dirty digits. Examples of "not difficult to see", "very dirty", and "very blurred" images are presented in Figures 5–7, respectively.

Table 2 presents the experimental results for recognition of the "not difficult to see" digits. The false-positive rate was low in our experiment. Several digits such as "2", "3", "4", and "5" comprised 0% of the false-positives.

Table 3 shows the experimental results for recognition of the "very blurred" digits. The false-positive rate was higher than "not difficult to see" but still lower than 10%. The lowest was digit "2" with 0.92% while the highest was digit "8" with 7.90%.

The experimental results of the "very dirty" digits are given in Table 4. This table shows the low false-positive rate of the "very dirty" digits. Even digit "3" had 12.50%, other digits had lower than 10% of the false-positives. Especially, the digit "2" had 0% of the false-positives.

Along with the recognition rate, the operating speed of the proposed architecture was also observed. Table 5 presents the timing testing of one digit in seconds. The experimental results showed that the proposed framework could detect the engraved digit in a very short period of time. In addition, the CNN had many matrix multiplications which are suitable for using the GPU. Thus, classification by the CNN was very fast.

Table 2. Recognition of the "not difficult to see" digits.

	False-positive		Recognized (not difficult to see)									
Digits	Total	Percentage (%)	0	1	2	3	4	5	6	7	8	9
0	3	0.17	1770	0	0	0	0	2	1	0	0	0
1	4	0.11	0	3568	0	0	2	1	1	0	0	0
2	0	0.00	0	0	524	0	0	0	0	0	0	0
3	0	0.00	0	0	0	160	0	0	0	0	0	0
4	0	0.00	0	0	0	0	221	0	0	0	0	0
5	0	0.00	0	0	0	0	0	146	0	0	0	0
6	3	0.28	0	0	0	0	0	3	1076	0	0	0
7	1	0.06	0	0	0	0	0	1	0	1695	0	0
8	2	0.58	0	0	1	0	0	1	0	0	344	0
9	2	1.03	0	0	0	0	0	2	0	0	0	193

Table 3. Recognition of the "very blurred" digits.

	False-positive		Recognized ("very blurred" digit)									
Digits	Total	Percentage (%)	0	1	2	3	4	5	6	7	8	9
0	15	1.49	991	2	1	0	0	0	12	0	0	0
1	10	0.92	1	1083	3	0	3	1	2	0	0	0
2	1	0.54	0	0	185	0	0	1	0	0	0	0
3	4	7.84	0	1	1	47	1	0	0	1	0	0
4	7	2.78	0	6	1	0	245	0	0	0	0	0
5	5	5.26	1	0	0	0	0	90	4	0	0	0
6	9	1.22	6	0	0	0	0	3	731	0	0	0
7	3	2.03	0	2	0	0	0	0	1	145	0	0
8	6	7.90	1	0	0	0	0	3	2	0	70	0
9	2	7.69	2	0	0	0	0	0	0	0	0	24

The training process of two modules (HOG–real AdaBoost and CNN) was also investigated in our experiments. Figure 8 presents the true-positive rate of the HOG–real AdaBoost module. After 200 iterations, the true-positive rate rose to 0.9982. These experimental results demonstrated the accuracy of the training phase of the HOG–real AdaBoost module.

Figure 9 shows the true-positive rate of the CNN training phase. At iteration 2060, the true-positive rate was 0.9962. These results also confirmed the accuracy of the CNN training module. We train the CNN for 2060 iterations because after that the accuracy is stable.

4. Conclusions

This paper presents an experimental architecture for recognizing sequences of engraved digits on steel plates. Unlike conventional digit datasets, the digits on steel are generally difficult to recognize because these digits could be blurred, smudgy, dirty, not clear, tilted, and overlapped by other digits. Hence, the recognition task of engraved digits is a more challenging task than the recognition of normal digits which are not engraved.

Table 4. Recognition of the "very dirty" digits.

	False-positive		Recognized (very dirty)									
Characters	Total	Percentage (%)	0	1	2	3	4	5	6	7	8	9
0	6	1.52	388	0	2	0	0	0	4	0	0	0
1	5	1.33	2	370	1	1	1	0	0	0	0	0
2	0	0.00	0	0	53	0	0	0	0	0	0	0
3	2	12.50	0	0	1	14	0	0	0	1	0	0
4	4	5.80	2	2	0	0	65	0	0	0	0	0
5	5	7.81	0	2	0	0	1	59	2	0	0	0
6	8	2.85	6	2	0	0	0	1	273	0	0	0
7	2	1.83	0	1	0	0	0	0	1	107	0	0
8	2	6.06	0	0	0	0	0	0	2	0	31	0
9	3	8.33	1	0	0	0	1	0	1	0	0	33

Table 5. The operating speed of the proposed architecture in seconds.

	Locating digits by HOG–real AdaBoost	Recognizing candidate digit windows by CNN	Total
Average time (s)	0.0410	0.0035	0.0456
Minimum time (s)	0.0349	0.0034	0.0393
Maximum time (s)	0.0575	0.0039	0.0871

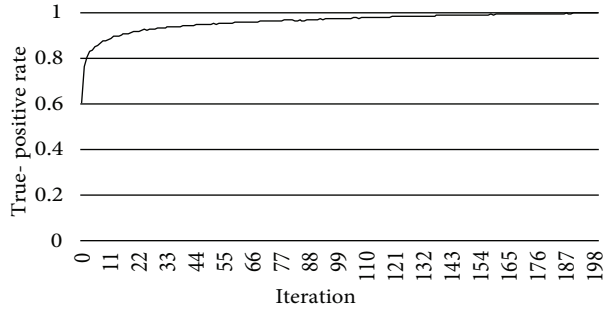

Figure 8. True-positive rate of the HOG–real AdaBoost module.

Our architecture detects not only one digit but also a string of digits. Several digits with uneven spacing and different sizes are detected at the same time. The appearance order of all digits of the string was detected.

The experimental results showed that our proposed architecture could be a solution to recognize the engraved digits on steel. It can also be used to recognize other sequences of objects on images. A promising avenue for future research could be the optimization of the proposed architecture to increase the recognition rate.

Figure 9. True-positive rate of the CNN module.

References

[1] Lecun Y, Boser B, Denker JS, Henderson D, Howard RE et al. Handwritten digit recognition with a back-propagation network. In: 2nd International Conference on Neural Information Processing Systems; Denver, CO, USA; 1989. pp. 396-404.

[2] Matan O, Burges CJC, Lecun Y, Denker JS. Multi-digit recognition using a space displacement neural network. In: 4th International Conference on Neural Information Processing Systems; Denver, CO, USA; 1991. pp. 488-495.

[3] Parisi R, Claudio EDD, Lucarelli G, Orlandi G. Car plate recognition by neural networks and image processing. In: Proceedings of the 1998 IEEE International Symposium on Circuits and Systems; Monterey, CA, USA; 1998. pp. 195-198.

[4] Anagnostopoulos CNE, Anagnostopoulos IE, Loumos V, Kayafas E. A license plate-recognition algorithm for Intelligent Transportation System Applications. IEEE Transactions on Intelligent Transportation Systems 2006; 7(3): 377-392. doi: 10.1109/TITS.2006.880641

[5] Hsieh P, Liang Y, Liao HM. Recognition of Blurred License Plate Images. In: 2010 IEEE International Workshop on Information Forensics and Security; Seattle, WA, USA; 2010. pp. 1-6

[6] Lecun Y, Jackel L, Bottou L, Cortes C, Denker J et al. Learning algorithms for classification: A comparison on handwritten digit recognition. In: Oh JH, Kwon C, Cho S (editors). Neural networks: The Statistical Mechanics Perspective. Singapore: World Scientific, 1995, pp. 261-276.

[7] Kussul E, Baidyk T. Improved method of handwritten digit recognition tested on MNIST database. Image and Vision Computing 2004; 22 (12): 971-981. doi: 10.1016/j.imavis.2004.03.008

[8] Simard PY, Steinkraus D, Platt J. Best practices for convolutional neural networks applied to visual document analysis. In: 7th International Conference on Document Analysis and Recognition; Edinburgh, UK; 2003. pp. 958-963.

[9] Sanchez DA, Bulon SG, Moreno L, Birlutiu A, Kadar M. Automatic Character Recognition in Porcelain Ware. Acta Technica Napocensis 2018; 59(3): 8-12.

[10] Patil AV, Dhanvijay MM. Engraved character recognition using computer vision to recognize engine and chassis numbers: Computer vision technique to identify engraved numbers. In: 2015 International Conference on Information Processing; Pune, India; 2015. pp. 151-154

[11] Zakaria Z, Suandi SA. Face detection using combination of Neural Network and AdaBoost. In: IEEE Region 10 Conference; Bali, Indonesia; 2011. pp. 335-338.

[12] Yang S, Chen LF, Yan T, Zhao YH, Fan YJ. An ensemble classification algorithm for convolutional neural network based on AdaBoost. In: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science; Wuhan, China; 2017. pp. 401-406.

[13] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition; San Diego, CA, USA; 2005. pp. 886-893.

[14] Rätsch G, Onoda T, Müller KR. Soft margins for AdaBoost. Machine Learning 2001; 42(3): 287-320. doi: 10.1023/A:1007618119488

[15] Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 1997; 55(1): 119-139. doi: 10.1006/jcss.1997.1504

[16] Huang C, Wu B, Ai H, Lao S. Omni-directional face detection based on real AdaBoost. In: IEEE International Conference on Image Processing; Singapore, Singapore; 2004. pp. 593-596.

[17] Yan C, Wang Y, Zhang Z. Face recognition based on real AdaBoost and Kalman Forecast. In: International Conference on Artificial Intelligence and Computational Intelligence; Taiyuan, China; 2011. pp. 489-496.

[18] Aoki D, Watada J. Human tracking method based on improved HOG+Real AdaBoost. In: 10th Asian Control Conference; Kota Kinabalu, Malaysia; 2015. pp. 1-6.

[19] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The Annals of Statistics 2000; 28(2): 337-407. doi: 10.1214/aos/1016218223

[20] Haykin S. Neural Networks and Learning Machines 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[21] Ravdin PM, Clark GM. A practical application of neural network analysis for predicting outcome of individual breast cancer patients. Breast Cancer Research and Treatment 1992; 22(3): 285-293. doi: 10.1007/BF01840841.

[22] Cardoso G, Rolim JG, Zurn HH. Application of neural-network modules to electricpower system fault section estimation. IEEE Transactions on Power Delivery 2004; 19(3): 1034-1041. doi: 10.1109/TPWRD.2004.829911.

[23] Celik AE, Karatepe Y. Evaluating and forecasting banking crises through neural network models: An application for Turkish banking sector. Expert Systems with Applications 2007; 33(4): 809-815. doi: 10.1016/j.eswa.2006.07.005.

[24] Grzonka D, Kolodziej J, Tao J, Khan SU. Artificial neural network support to monitoring of the evolutionary driven security aware scheduling in computational distributed environments. Future Generation Computer Systems 2015; 51: 72-86. doi: 10.1016/j.future.2014.10.031.

[25] Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks 2015; 61: 85-117. doi: 10.1016/j.neunet.2014.09.003.

[26] Bengio Y. Learning deep architectures for AI. Foundations and trends in Machine Learning 2009; 2(1): 1-127. doi: 10.1561/2200000006.

[27] Cheng Y. Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 1995; 17(8):790-799. doi: 10.1109/34.400568.

[28] Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 2002; 24(5): 603-619. doi: 10.1109/34.1000236.

[29] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Neural Information Processing Systems; Stateline, NV, USA; 2012. pp. 1-9.