5th INTERNATIONAL CONFERENCE
TRANSPORT SYSTEMS TELEMATICS TST'05

ZESZYTY NAUKOWE
POLITECHNIKI ŚLĄSKIEJ 2005
TRANSPORT z. 59, nr kol. 1691

traffic control,
number plate recognition,
Artificial Neural Networks

Raimond LAPTIK [1]
Dalius NAVAKAUSKAS [2]

APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN NUMBER PLATE RECOGNITION SYSTEM

We investigate the recognition of car number plates by Artificial Neural Networks aiming to make the system cheaper by the use of web cameras at the same time not sacrificing the performance and recognition rate. Main attention is devoted to the investigation of Kohonen Feature Map and Multilayer Perceptron, their structure and training parameters. The description of the system under construction, the composition of the recognition software module, simulation experiments, achieved recognition results and suggestions for the future development are given.

ZASTOSOWANIE SZTUCZNYCH SIECI NEURONOWYCH W SYSTEMIE ROZPOZNAWANIA TABLIC REJESTRACYJNYCH

Badamy możliwości rozpoznawania tablic rejestracyjnych samochodów za pomocą sztucznych sieci neuronowych, którego celem byłoby potanienie systemu poprzez użycie kamer internetowych, jednocześnie nie poświęcając jego sprawności i wydajności rozpoznawania. Główna uwaga skupiona jest na badaniu Mapy Cech Kohonena, oraz Postrzeganiu Wielowarstwowym, ich strukturze i parametrach szkoleniowych. Zaprezentowano opis budowanego systemu, skład modułu oprogramowania rozpoznającego, eksperymenty symulacyjne, uzyskane wyniki rozpoznawania i sugestie dla dalszego rozwoju.

1. INTRODUCTION

Rapidly growing traffic requires intelligent control. Systems able to control traffic lights, track movement of vehicles under investigation, for example, stolen ones or violators, are required. The essential part of such a system is the recognition engine that is able to recognize car number plates (CNP) from captured images. Currently there are many CNP recognition systems: "VisiCar" [3], "CARMEN" [7], "Iris" [1], "NC6000" [8]. Algorithms of these systems are similar to those used in commercial optical character recognition software,

[1] Faculty of Electronics, Vilnius Gediminas Technical University, Naugarduko 41-422, Vilnius, Lithuania
+370 5 2744756, raimond.laptik@el.vtu.lt
[2] Faculty of Electronics, Vilnius Gediminas Technical University, Naugarduko 41-422, Vilnius, Lithuania
+370 5 2744756, dalius.navakauskas@el.vtu.lt

like "Fine Reader", "CuneIForm", "Text Bridge", or optical character devices, like "Siemens Pocket Reader" and "C pen". The outstanding feature of CNP recognition systems is their capability to work in a real time or at least to have a much shorter processing time.

In the following, the recognition of car number plates by Artificial Neural Networks [4] ensuring quick recall and straightforward parallel implementation is studied. The main aim is to make the system cheaper by the use of web cameras at the same time not sacrificing the performance and recognition rate. We start with the description of the system being created, present and explain the composition of the recognition software module. Afterwards, devote main attention to the investigation of two types of Neural Networks, namely, Kohonen Feature Map [6] and Multilayer Perceptron [10], their structure and training parameters. Basing on the results of experiments when Neural Networks recognize numerals taken from CNP, we pick the best Neural Network. Finally, we present achieved CNP recognition results and give suggestions for the future development.

2. CONSTRUCTION OF THE SYSTEM

In order to make minimal system, hardware implementation is quite simple (see Fig.1). System consists of image input device – web camera with resolution of 640×480 pixels, and personal computer that can be of desktop or laptop type. The later one suits well when someone needs mobility from CNP recognition system.

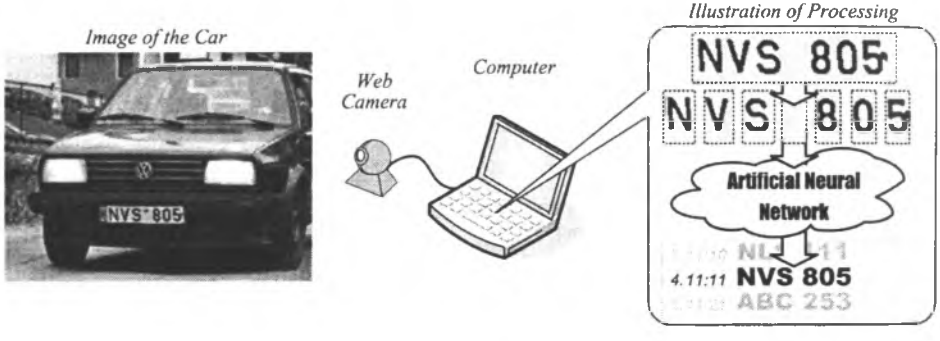

Fig.1. An illustration of the car number plate recognition system

The essence of CNP recognition system is special software [9] used for image processing and recognition. Fig.2 presents the flowchart of recognition module. Let us briefly explain its main processing stages.

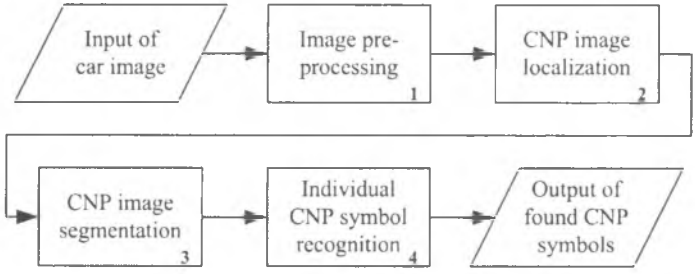

Fig.2. The flowchart of the recognition module

After the input of image, it is pre-processed [2]. Doing so, image is converted to grayscale [5] using simple expression $p = (r + g + b)/3$, here r, g, and b are red, green and blue color components correspondingly, p – gray level. Then employing adaptive threshold function image is converted to black and white scale by: $t = (p > T)$. Threshold value T is calculated by analyzing neighboring pixels in 10×10 pixels wide window. Experimentally threshold calculation expression is selected as $T = (p_{min} + p_{max})/2$, here p_{min} and p_{max} are maximum and minimum gray level values in each window.

Next processing step is the localization of car number plate. We assume that CNP image consists of a rectangle and several symbols – all of know size. All contours in the whole image and their coordinates are estimated by the use of flood-fill algorithm [11] that also enables to calculate the sizes of individual contours. If estimated contour size is similar to CNP size, then contour's inside area is further investigated. The conclusion that CNP is located is done only if objects inside of the contour are of symbol size. During successive processing only that part of image is analyzed.

In third step segmentation of discovered CNP image is performed, when boundaries of individual symbols are calculated. At this step again information about standardized size of CNP symbol is used.

In fourth step symbols are passed to Artificial Neural Network for recognition. This is the most important step of all the procedure as from it depends the correctness of CNP symbols outputted from the recognition module. That is why main effort in the following will be devoted to determination of the best ANN structure and training parameters.

3. EMPLOYMENT OF ARTIFICIAL NEURAL NETWORKS

In the CPN recognition system being created Artificial Neural Network is used for symbol recognition. ANN should classify normalized size symbol image – assign it to respective symbol class.

Before the presentation to ANN image of the symbol is normalized to 10×10 pixels size, with the following re-arrangement of corresponding matrix **A** to a 100 elements vector **a**. Normalization in size is done according to:

$$a_{ij} = \begin{cases} 0, & q_{ij} < 0.5; \\ 1, & q_{ij} \geq 0.5, \end{cases} \tag{1}$$

with variable q_{ij} expressing averaged blackness of normalized pixel that calculates by:

$$q_{ij} = \frac{1}{k_{\mathrm{H}}^2} \sum_{I=I_0+i-k_{\mathrm{H}}/2}^{I_0+i+k_{\mathrm{H}}/2} \sum_{J=J_0+j-k_{\mathrm{H}}/2}^{J_0+j+k_{\mathrm{H}}/2} t_{IJ} \,. \tag{2}$$

Here a_{ij} and t_{IJ} are the binary value of the pixel taken from normalized and original symbol images correspondingly; i and j are indexes of normalized size symbol image; I and J indexes pixels in the original image; I_0 and J_0 are global coordinates of the beginning of symbol image. Necessary value of the coefficient k_{H} is calculated according to the change of symbol size by $k_{\mathrm{H}} = H_{\mathrm{t}} / H_{\mathrm{n}}$, here H_{n} – required symbol height (in our case – 10), H_{t} – real symbol height.

3.1. KOHONEN FEATURE MAP

Kohonen Feature Map (KFM) was chosen because it has good generalization (network is able to classify correctly patterns that were not used during training). To be studied KFM is made of one-dimensional input layer with 100 inputs and two-dimensional output layer with in general $N_{\mathrm{X}} \times N_{\mathrm{Y}}$ outputs (see Section 4.1 for experimental investigation of its best size).

KFM operates as follows. Symbol vector **a** is passed to the network. Neuron which has the closest to **a** weight vector $\mathbf{w}_{\mathrm{L}}$ becomes a winner (its output $s_{\mathrm{L}} = 1$). The closest neuron employing Euclidian metric is determined by:

$$\|\mathbf{a} - \mathbf{w}_{\mathrm{L}}\| = \min_i \|\mathbf{a} - \mathbf{w}_i\|, \quad \text{when} \tag{3}$$

$$\|\mathbf{a} - \mathbf{w}_i\| = \sqrt{\sum_j (a_j - w_{ij})^2} \,. \tag{4}$$

KFM is trained by the rule:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{\mathrm{L}i}(\mathbf{a}(t) - \mathbf{w}_i(t)). \tag{5}$$

Here $\mathbf{w}_i(t)$ is i-th output layer neuron at t time instance; $\mathbf{a}(t)$ – at the same time passed symbol vector; h_{Li} – neighborhood function:

$$h_{Li} = \alpha(t)\exp\left(\frac{\|\mathbf{r}_i - \mathbf{r}_L\|^2}{2\sigma^2(t)}\right). \tag{6}$$

Here $\alpha(t)$ is the learning rate factor, which decreases monotonically with the learning step; $\mathbf{r}_i \in \Re^2$ and $\mathbf{r}_L \in \Re^2$ correspondingly are locations of current neuron and winner in the two-dimensional output layer grid; $\sigma(t)$ – width of the neighbourhood function, which is also decreasing monotonically with learning steps. Each neuron corresponds to unique class. Neuron is assigned to class if neighbourhood function value is not less than 0.01.

3.2. MULTILAYER PERCEPTRON

Multilayer Perceptron (MLP) was selected due to its ability to approximate any non-linear relationship. We leave experimental selection of MLP size to Section 4.2.

MLP is trained using error backpropagation rule:

$$\tilde{w}_{jl}^{k-1} = (1-\mu)w_{jl}^{k-1} - \mu\lambda s_j^{k-1}\delta_l^k, \tag{7}$$

here w_{jl}^{k-1} – weight connecting j neuron in k–1 layer with l neuron in k layer; $\tilde{w}_{jl}^{k-1}$ – a new value of the same weight; μ – a momentum term; λ – learning rate constant; s_j^{k-1} – output signal of j neuron in k–1 layer; δ_l^k – l neuron error in k layer, calculated by:

$$\delta_l^k = \begin{cases} f'(z_l^k)(s_l^k - \bar{s}_l), & k = K; \\ f'(z_l^k)\sum_j w_{lj}^k \delta_j^{k+1}, & 1 \le k < K. \end{cases} \tag{8}$$

Here $\bar{s}_l$ – expected l neuron output; z_l^k – sum of all inputs of l neuron in k layer; $f'(\cdot)$ – derivative of neuron activation function (sigmoid or hyperbolic tangent).

Sum of all inputs of l neuron in k layer is calculated by:

$$z_l^k = \sum_{i=1}^{M^{k-1}} w_{il}^{k-1} s_i^{k-1}. \tag{9}$$

Here the output of i neuron in k–1 layer is calculated by $s_i^{k-1} = f(z_i^{k-1})$.

4. INVESTIGATION OF SELECTED ARTIFICIAL NEURAL NETWORKS

In order to evaluate selected types of ANNs and determine the one that is mostly suited for CNP recognition, recognition of CPN numerals is performed. ANNs are trained using a collection of training sets while for the check of ANN recognition accuracy single validation set is used. Network generalization (ability to classify samples, not used during training) is explored.

4.1. RECOGNITION WITH KOHONEN FEATURE MAP

For Kohonen Feature Map Gauss neighbourhood function with $\sigma(t) \in [10^{-1}, 3]$, $\alpha(t) \in [10^{-4}, 1]$ and maximum number of iterations 3000 is selected. Varying the size of KFM, input normalization limits are determined. As can be seen in Table 1, best recognition results are received when input values are centered around "0" within [–1, 1] limits. Moreover, the best output layer size is 20×20 neurons.

Table 1

Results on the determination of normalization limits for inputs

Output layer, $N_X \times N_Y$	*Normalization limits for inputs*		*Recognition accuracy on the validation set, %*
	max	min	
10×10	1	–1	70
	0	–1	80
	0	1	60
20×10	1	–1	70
	0	–1	60
	0	1	70
20×20	**1**	**–1**	**80**
	0	–1	50
	0	1	70
30×20	1	–1	70
30×30	1	–1	50

Adjustment of $\alpha_{max}(t) = 0.5$ and $\sigma_{max}(t) = 3$ give not better then 90 % recognition accuracy on the validation set (see Table 2). Recognition of complete validation set becomes possible only after increase to 5 different training sets.

Table 2

Results on the estimation of the training parameters for Kohonen Feature Map

$\alpha(t)$		*Recognition accuracy on the validation set, %*	$\sigma(t)$		*Recognition accuracy on the validation set, %*
max	min		max	min	
1.0	10^{-4}	80	5	10^{-1}	80
0.7		80	4		80
0.5		90	3		90
0.3		70	2		60
0.1		50	1		50

4.2. RECOGNITION WITH MULTILAYER PERCEPTRON

For Multilayer Perceptron momentum term μ value was chosen to be 0.3. However, after few experiments (see the upper part of Table 3) recognition accuracy still was low. So weight decay [10] was introduced and applied. Decrease of weights was performed each 20th iteration using:

$$\tilde{w}_{ji}^{k-1} = (1-\gamma)\tilde{w}_{ji}^{k-1}, \tag{10}$$

here γ – weight decay coefficient is equal to 0.1 (estimated experimentally). Recognition results on the validation set are presented in Table 3 (lower part of table). It can be seen that with weight decay recognition results are much better. It appears that the best MLP size is 15–20–10, while additional experiments showed that quickest training of this MLP become possible when learning rate constant λ value is 0.05.

Table 3

Results on the size estimation of Multilayer Perceptron

γ	*Size of MLP,* N_1–N_2–N_3	*Number of training sets*	*Number of training iterations*	*Recognition accuracy on the validation set, %*
0	30–30–10	1	230	60
	100–30–10	5	146	80
0.1	15–30–10	5	554	100
	15–25–10	5	1620	100
	15–20–10	**5**	**554**	**100**
	10–15–10	5	596	90

Summarizing results from both ANN investigations we draw conclusion that Multilayer Perceptron is more suitable for CNP recognition systems. Main factors influencing this decision were: the fast training and recall phases as also as small size of the network.

5. REZULTS ON CAR NUMBER PLATE RECOGNITION

For CPN recognition multilayer perceptron network was selected. MLP output layer was increased to 37 neurons: 10 neurons were dedicated to numerals, 26 neurons were dedicated to letters and 1 neuron was selected to represent non-symbol class. 5 % amount of random noise (estimated experimentally) was added to training data in order to higher generalization. Training set was made of 943 patterns in total. Learning rate $\lambda = 0.02$ decreased by 0.0001 in each iteration (estimated experimentally), momentum term $\mu = 0.3$, weight decay was used each 25th iteration (estimated experimentally) with decay ratio $\gamma = 0.1$.

As can be seen from Table 4, the best recognition accuracy on the validation set was reached when network size is 100-60-37 neurons.

The best MLP was applied in the recognition system and tested on recognition of about 720 symbols. System reached recognition accuracy of 98.2 %.

Table 4

Results on the size estimation of Multilayer Perceptron

Size of MLP, N_1–N_2–N_3	*Number of iterations*	*Average square error on the training se)*	*Validation set recognition accuracy, %*
100–20–37	7887	0.059	91.89
100–30–37	5853	0.081	94.59
100–40–37	1422	0.046	94.59
100–50–37	5904	0.032	94.59
100–60–37	**3915**	**0.023**	**94.59**
100–70–37	4846	0.049	94.59

6. CONCLUSIONS

1. Construction of car number plate recognition system and employment of selected Artificial Neural Networks for the recognition are presented.
2. Experimentally estimated structures and training parameters of Kohonen Feature Map and Multilayer Perceptron confirm their suitability to recognize car number plates. Nevertheless, the best Multilayer Perceptron showed outstandingly fast training and recall phases as well as small size of the network.
3. Car number plate recognition system with best Multilayer Perceptron was tested on the 120 car images (~720 symbols) achieving 82 % overall recognition accuracy with: 96 % the place of car number plate discovery correctness; 98.2 % symbol recognition accuracy.

BIBLIOGRAPHY

[1] Automatic Number Plate Reader – IRIS // Roke Manor Research Limited, 1998.– http://www.roke.co.uk
[2] BROUGHTON A. S. Transform Methods in Imaging Processing // Lecture 2.– Rose-Hulman Institute of Technology, USA.
[3] DRAGHICI S. VisiCar System // Wayne State University, 1997.
[4] GUPTA M. M., JIN L., HOMMA N. Static and Dynamic Neural Networks.– John Willey & Sons Inc., 2003.
[5] KLETTE R., ZAMPERONI P. Handbook of Image Processing Operators.– John Willey & Sons Ltd., 1996.
[6] KOHONEN T. Self-Organizing Maps.– Springer-Verlag, 2001.
[7] Number Plate Recognition Software Engine – CARMEN.– Adaptive Recognition Hungary.– http://www.anpr.net
[8] Number Plate Recognition System Module NC6000 // NeuriCam S. p. A.– http://www.neuricam.com
[9] RAO V. B. C++ Neural Networks and Fuzzy Logic.– MTBooks and IDG Books Worldwide Inc., 1995.
[10] RUMELHART D. E., HINTON G. E., WILLIAMS R. J. Learning internal representations by error propagation // In Parallel Distributed Processing: Explorations in the Microstructures of Cognition.– MA: MIT Press, 1986.– Vol. 1.– P. 318–362.
[11] SHAW J. R. QuickFill: An efficient flood fill algorithm // The Code Project, 2004.– http://www.codeproject.com

Reviewer: Ph. D. Stanisław Krawiec