Using a Convolutional Neural Network for Machine Written Character Recognition

Ladislav Karrach [1], Elena Pivarčiová [1]

[1]*Department of Manufacturing and Automation Technology, Faculty of Technology, Technical University in Zvolen, Masarykova 24, 960 01 Zvolen, Slovakia*

***Abstract* – Convolutional neural networks are special types of artificial neural networks that can solve various tasks in computer vision, such as image classification, object detection, and general recognition. The paper presents the basic building blocks of convolutional neural networks and their architecture, and compares their recognition accuracy with other character recognition techniques using the example of character recognition from vehicle registration plates. The purpose of the experiments was to determine the optimal configuration of the convolutional neural network and the influence of the size and design method of the training set on the recognition rate. The study shows that although convolutional neural networks have recently gained attention, traditional recognition methods are still relevant, and the choice of the right classifier and its configuration depends on the type of recognition task.**

***Keywords* – Convolutional neural network, optical character recognition, feature vector, feature extraction, classification.**

1. Introduction

Traditional Optical Character Recognition (OCR) systems usually consist of these steps [2], [7]:

- **Pre-processing** prepares the input image to be optimal for character recognition (includes operations like binarization, noise reduction, smoothing, contrast enhancement, illumination transformations, and skew correction).
- **Segmentation** identifies individual lines of text (text line detection), words (word extraction), and characters (character segmentation).
- **Feature extraction** each segmented character must be unambiguously and concisely described by a feature vector – a point in n-dimensional space. This point must be far enough from points that describe other characters, and vice versa, close enough to points that describe modifications of the same character.
- **Classification** the input image represented by the feature vector must be assigned to a class that represents one particular character (0–9, A–Z).
- **Post-processing** improves the quality of recognized text, for example by using of a dictionary for correcting minor errors.

In our prior works, we focused on feature extraction and classification steps, and we compared various feature extractors (statistical, structural, image moments) and classifiers (Nearest Neighbor, Multilayer Perceptron, Probabilistic Neural Network, Radial Basis Function Network, Naive Bayes Classifier, Template Matching) for recognition of characters taken from vehicle registration plates [6]. We found that the choice of an appropriate feature vector had a major impact on the recognition rate, and the choice of a classifier is of secondary importance. In other words, how to construct a feature vector (n-dimensional vector of real values) from an input image is a critical question.

A suitable feature vector must accurately describe the object (character) of interest and distinguish it from objects (characters) of other classes. Feature extractors are often designed by experts (humans) and are specific to one particular task (character recognition, face recognition, traffic surveillance, 2D code recognition, etc.).

DOI: 10.18421/TEM123-03
https://doi.org/10.18421/TEM123-03

Corresponding author: Elena Pivarčiová,
Department of Manufacturing and Automation Technology, Faculty of Technology, Technical University in Zvolen, Masarykova 24, 960 01 Zvolen, Slovakia
Email: pivarciova@tuzvo.sk

Received: 30 March 2023.
Revised: 11 June 2023.
Accepted: 19 July 2023.
Published: 28 August 2023.

Convolutional neural networks (CNNs) combine a feature extraction step with a classification step. I.e. they are able to learn (in the training phase) how to efficiently extract features from the input image and then classify it.

2. Convolutional Neural Networks (CNN or ConvNet)

A typical convolutional neural network (CNN) is made up of these building blocks (layers) (Figure 1) [16], [17].

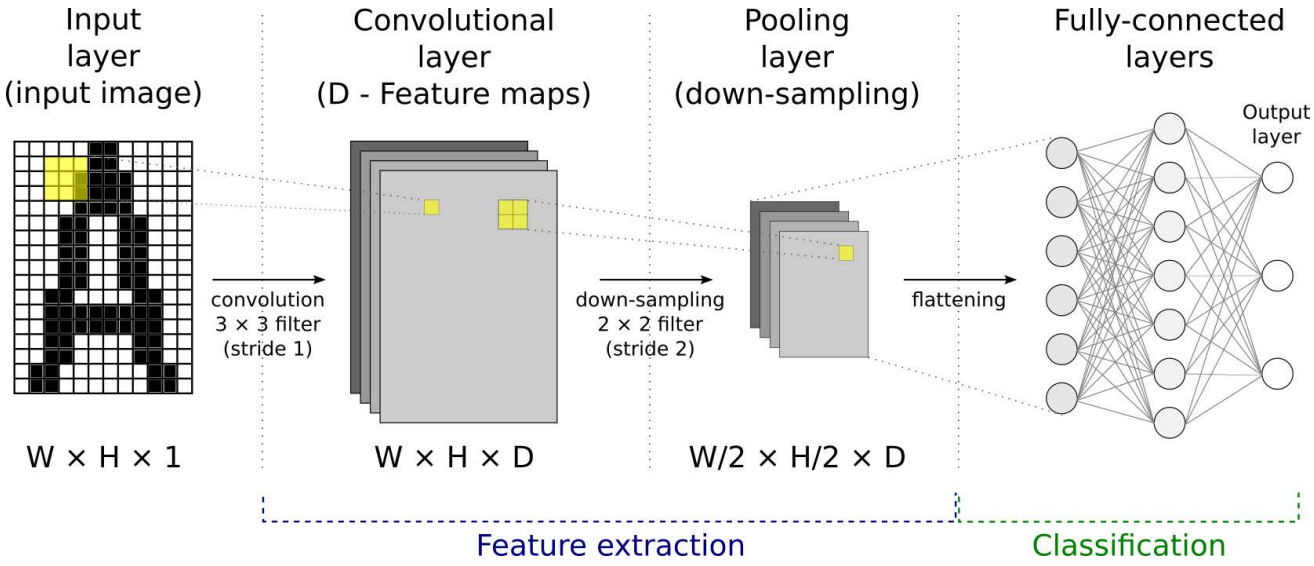

Figure 1. The basic building blocks of a convolutional neural network

- **Input layer**: CNN explicitly assumes that the input is an image (image – spatially organized data). The size of the input image should be multiple times divisible by 2, i.e. common sizes are 32, 64, 96, 224 (Standford.edu).
- **Convolutional layer (C)**: Neurons in a convolutional layer are connected only to a small region (receptive field) of the layer before it, instead of all neurons in fully connected layers. Each neuron in the convolutional layer computes the dot product between its weights (learnable filter, kernel, and mask) and this small region to which it is connected (Figure 2). Each convolutional layer works with multiple filters and produces multiple feature maps. The number of filters (as well as the number of feature maps produced in the convolutional layer) determines the "depth" of the layer. The task of different filters is to extract different features (the first layers capture low-level features such as corners, edges, end-points, gradient orientation, and colour); by increasing the number of convolutional layers high-level features begin to be captured). Filters (convolutional kernels) are trained using the backpropagation algorithm (i.e., filters are not created by hand, but their weights are initialized randomly and subsequently modified during training).

It is common to use 3×3 filters, and for larger input images, 5×5 or even 7×7 filters. A filter must always have the same number of channels as the input (often referred to as "depth"; convolution operation uses a multi-channel kernel sliding over a multi-channel feature map to produce a single output feature map). Parameter "stride" determines the step by which the filter is shifted over the input image (feature map; both horizontally and vertically).

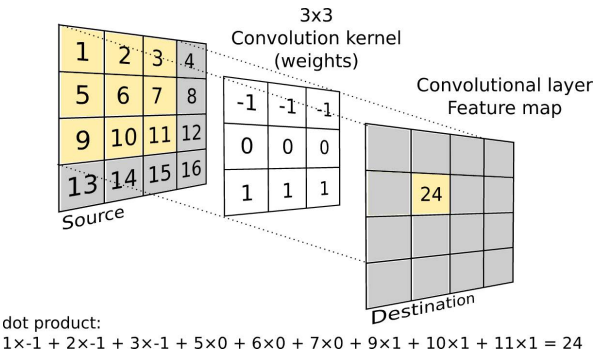

Figure 2. Convolution operation

- **Activation function**: The output of each convolutional layer is passed through an activation function. Commonly used is the Rectified Linear Unit (ReLU) nonlinear activation function: $y = \max(0, x)$.

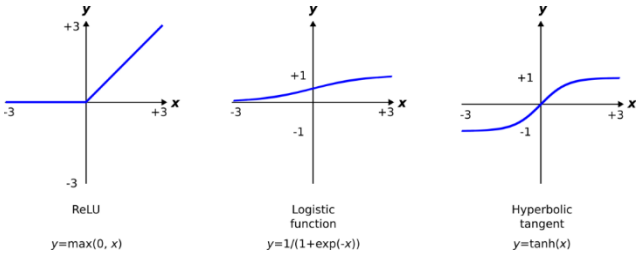

Figure 3. Common activation functions

- **Pooling layer (S)**: performs down-sampling and thus reduces the dimensionality (computational complexity) of the network. A 2×2 patch (filter, kernel, mask) shifted with stride 2 over the input feature map is commonly used (Figure 4). Max-Pooling calculates the maximum of the input values (highlights the most present feature in the patch), while Avg-Pooling calculates the average of the input values. One purpose of the pooling operation is to make the model independent of slight differences in extracted feature positions (shift and distortion invariance), and the other is to reduce the data amount for the next processing layers, thus making the model faster. After several convolutional and pooling layers, the feature map sizes are reduced and more complex features are extracted. The output of the last convolutional or pooling layer is flattened to an input for the fully connected layer.

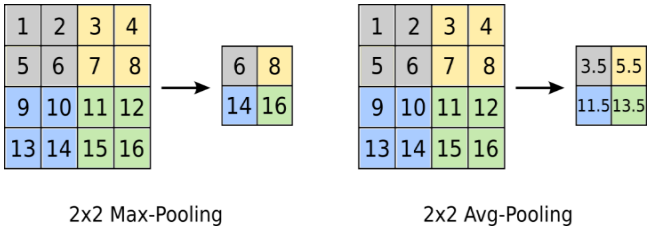

Figure 4. Pooling operation

- **Fully-connected layers (F)**: every neuron in a fully-connected (dense) layer is connected to every neuron in the previous layer. Same as a traditional Multi-Layer Perceptron (MLP), which merely is a neural network with at least three layers [4]. The number of neurons in the last (output) layer corresponds to the number of classes into which the input images are to be classified.

3. Related Work

CNNs have attracted the attention of researchers in various computer vision tasks.

3.1. Convolutional Neural Network Architectures

The way the individual building blocks (layers) of the CNN are arranged and interconnected, together with their parameters (such as depth, filter size, stride size, padding size) determines the architecture of the CNN. Some of the well-known CNN architectures are:

- **LeNet-5** [12]: used for handwritten character recognition task and to compare the performance of several learning techniques on a benchmark data set for handwritten digit recognition (The first LeNet network designed for the recognition of handwritten zip code numbers was published back in 1989 by Y. LeCun [11]). LeNet-5 consists of seven layers (C1, S2, C3, S4, F5-7) in addition to input layer (32×32 input images, 1 channel). The first layer – convolutional (C1) – produces 6 feature maps, each of size 28×28. The second layer – subsampling/average-pooling (S2) – produces 6 feature maps, each of size 14×14. The third layer – convolutional (C3) – produces 16 feature maps, each of size 10×10 (One C3 feature map is connected to several S2 feature maps). The fourth layer – subsampling/average-pooling (S4) – produces 16 feature maps, each of size 5×5. The fifth layer – convolutional (C5) – produces 120 feature maps, each of size 1×1. Each convolutional layer (Cx) works with a 5×5 filter and a sigmoid activation function, and each subsampling layer (Sx) works with a 2×2 patch and performs average pooling. The sixth layer – fully-connected layer has 84 neurons, all connected to each of the 120 neurons in the prior layer. The seventh layer – fully-connected has 10 neurons (representing the digits "0"–"9"), all connected to each of the 84 neurons in the prior layer.
- **AlexNet** [9]: AlexNet consists of eleven layers (C1, S2, C3, S4, C5-7, S8, F9-11). The first five are convolutional layers, some of which are followed by max-pooling layers, and the last three are fully-connected layers. It used the non-saturating ReLU activation function, which showed improvement in training performance over TanH and Sigmoid. The first convolutional layer (C1) filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels. The second convolutional layer (C3) filters output of the S2 layer with 256 kernels of size 5×5×96. The third convolutional layer (C5) has 384 kernels of size 3×3×256 connected to the normalized outputs of the S4 layer.

- **ZFNet** [18]: Similar like AlexNet, it is made up of 11 layers (C1, S2, C3, S4, C5-7, S8, F9-11). The first convolutional layer (C1) filters the 224×224×3 input image with 96 kernels of size 7×7×3 with a stride of 2 pixels. The second convolutional layer (C3) filters output of the S2 layer with 256 kernels of size 5×5×96. The third and fourth convolutional layer (C5-6) has 384 kernels of size 3×3×256 and 3×3×384 connected to the normalized outputs of the S4 layer. The fifth convolutional layer (C7) has 256 kernels of size 3×3×384. Max-pooling layers use 3×3 patch with a stride 2 pixels.
- **VGGNet** [15]: VGGNet increased the depth to 16–19 layers and used small 3×3 filters. All the hidden layers in the VGG network use ReLU activation function. Pooling layer is not placed after each convolutional layer (C1-2, S3, C4-5, S6, C7-9, S10, C11-13, S14, C15-17, S18, F19-21).

Table 1. Properties of the well-known CNNs

	Input Image	Number of all layers	Number of convolutional layers, Filter sizes	Number of pooling layers, Patch size	Number of fully-connected layers	Number of neurons in output layer
LeNet-5	32×32×1	7	2, (5×5)	2, (2×2, Avg-Pool)	3	10
AlexNet	227×227×3	11	5, (11×11, 5×5, 3×3)	3, (3×3, Max-Pool)	3	1000
ZFNet	224×224×3	11	5, (7×7, 5×5, 3×3)	3, (3×3, Max-Pool)	3	1000
VGGNet16	224×224×3	21	13, (3×3)	5, (2×2, Max-Pool)	3	1000
Darknet-19	224×224×3	25	19, (3×3)	5+1 (Avg-Pool), (2×2, Max-Pool)	0 (Soft-Max)	1000

Common CNN architectures are:
- INPUT → CONV → ReLU → FC
- INPUT → [CONV → ReLU → POOL]*2 → FC → ReLU → FC
- INPUT → [[CONV → ReLU]*2 → POOL]*3 → [FC → ReLU]*2 → FC

(two CONV layers stacked before every POOL layer)

where CONV is Convolutional layer, ReLU is nonlinear activation function, POOL is Pooling layer and FC is Fully-Connected layer.

Just like regular neural networks, CNNs need to be trained before they can be used for image classification. Training consists of forward and backward propagation. During forward propagation, the output of the network is calculated using the existing weights and biases (initialized with small random numbers). In backward propagation, the error recorded at the output is propagated back through the network, and the weights and biases are updated to minimize the error (gradient descent back propagation algorithm).

3.2. Convolutional Neural Networks Used in Handwritten Character Recognition

Bora [1] used CNN for feature extraction and the Error Correcting Output Code (ECOC) for classification. They tested 4 CNN architectures (LeNet1, LeNet2, AlexNet, ZfNet) with the ECOC classifier. AlexNet-ECOC achieved the highest (97.1 %) accuracy on the NIST handwritten character image dataset.

Saqib [14] proposed CNN with four convolutional layers, three max-pooling layers, and two fully-connected layers (C1-2, S3, C4, S5, C6, S7, F8-9). Input image is 28×28, C1 employs 3×3 kernel and the ReLU activation function. Next three convolutional layers C2, C4, C6 are accompanied also by ReLU activation function and by 2×2 Max-Pooling layers S3, S5, S7. Two optimizers, “ADAM” and “RMSprop”, were evaluated. The CNN model with the “ADAM” optimizer achieved 99.56% accuracy on the Kaggle dataset (with English letters A–Z).

Khandokar [8]showed that the average recognition accuracy increases with a higher number of training images.

The accuracy reaches 92.9% with the 1000 training images on the NIST dataset. Zhang [19] explored the problem of recognizing handwritten Chinese characters. They found that for CNNs with small filter sizes, the deeper the network, the higher the accuracy.

Chowdhury [3] presented CNN for Bangla handwritten character recognition and proved that the CNN method is more efficient compared to other machine learning approaches.

Table 2. Properties of CNNs used for handwritten character recognition

	Input Image	Architecture	Number of convolutional layers; Filter sizes	Number of feature maps in individual convolutional layers	Dataset, training / testing images	Recognition accuracy
Bora [1]	128×128×3	AlexNet - ECOC	5; 11×11, 5×5, 3×3	96, 256, 384, 384, 256	NIST (upper case A–Z), 38558 / 25740	97.1 %
Saqib [14]	28×28×1	C1-2, S3, C4, S5, C6, S7, F8-9	4; 3×3	32, 64, 128, 256	Kaggle letters (A–Z), 297000 / 74490	99.56 %
Khandokar [8]	28×28×1	C1, S2, C3, S4, F5-6	2; 5×5	n1, n2	NIST, 1000 / 200	92.9 %
Zhang [19]	64×64×1	various	3–8; 3×3	64, 128, 256, 512	CASIA HWDB1.1 Chinese characters, 240 / 60 per class	99.5 – 99.6 %
Chowdhury [3]	50×50×1	C1, S2, C3, S4, F5-7	2; 5×5	32, 32	BanglaLekha-Isolated, 67500 / 7500	91.8 %

4. Material and Methods

Different CNN architectures and their impact on the resulting recognition accuracy were tested. Table 3 shows individual CNN configurations (K–Kernel size, S–Stride, P–Padding). The input of the CNN was always a binary image of size 12×17 and the output was a classification into one of 36 classes (numbers 0–9 and letters A–Z). The first convolutional layer had either 4 or 8 feature maps and after each convolutional layer the ReLU activation function was applied.

Table 3. Tested architectures of CNNs used for machine written character recognition

<table>
<tr><th rowspan="2">Layers</th><th colspan="4">Network</th></tr>
<tr><th>CNN_1.1_4/8</th><th>CNN_1.2_4/8</th><th>CNN_2.1_4/8</th><th>CNN_2.2_4/8</th></tr>
<tr><td>Input</td><td colspan="4">12 × 17 × 1</td></tr>
<tr><td rowspan="2">Conv1. / ReLU</td><td rowspan="2">12 × 17 × 4/8
K = 3×3, S = 1, P = 1</td><td>12 × 17 × 4/8
K = 3×3, S = 1, P = 1</td><td colspan="2" rowspan="2">12 × 17 × 4/8
K = 3×3, S = 1, P = 1</td></tr>
<tr><td>12 × 17 × 4/8
K = 3×3, S = 1, P = 1</td></tr>
<tr><td>Max-Pool.1</td><td colspan="2">6 × 9 × 4/8,
K = 2×2, S = 2</td><td>6 × 9 × 4/8,
K = 2×2, S = 2</td><td>6 × 9 × 4/8,
K = 2×2, S = 2</td></tr>
<tr><td>Conv2. / ReLU</td><td colspan="2">–</td><td>6 × 9 × 4/8
K = 3×3, S = 1, P = 1</td><td>6 × 9 × 8/16
K = 3×3, S = 1, P = 1</td></tr>
<tr><td>Max-Pool.2</td><td colspan="2">–</td><td>3 × 5 × 4/8,
K = 2×2, S = 2</td><td>3 × 5 × 8/16,
K = 2×2, S = 2</td></tr>
<tr><td>Flattening</td><td colspan="2">216 / 432</td><td>60 / 120</td><td>120 / 240</td></tr>
<tr><td>FC1</td><td colspan="2">180 / 324</td><td>76 / 116</td><td>116 / 196</td></tr>
<tr><td>FC2</td><td colspan="4">36</td></tr>
</table>

Figure 5 shows one of the tested convolutional neural network architectures with a first convolutional layer with 4 feature maps, followed by a first max-pooling layer, followed by a second convolutional layer with 8 feature maps, followed by a second max-pooling layer, followed by a fully-connected layer.

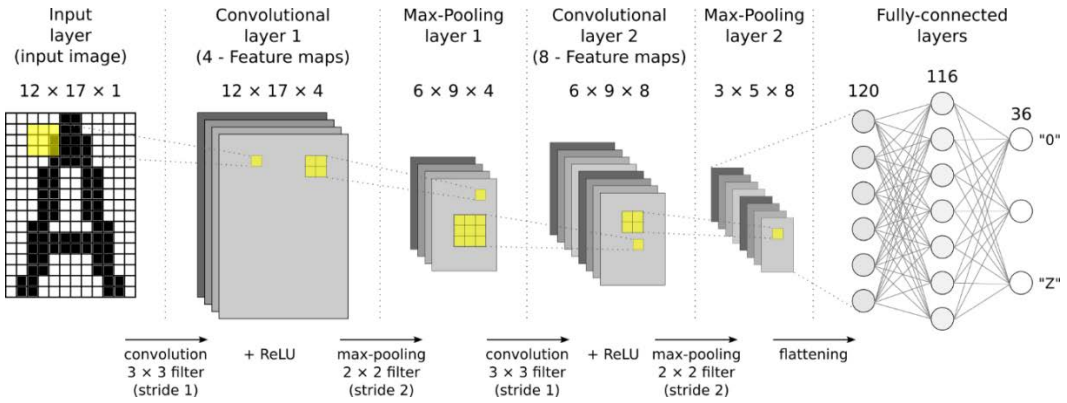

Figure 5. Visualization of CNN_2.2_4 architecture

5. Results

The testing set consisted of 56 real-scene images of vehicle registration plates (VRP), where each VRP contained 7 characters. The VRP was localized in an image, and individual characters forming the VRP were segmented [5]. Each segmented character was resized to the size of the training samples (12×17).

The numbers presented in the following tables are successfully recognized characters out of total 392 expected.

For training the convolutional neural network, the same training samples as in our previous experiments were used [6]. Therefore, we first recapitulate some of these results in Table 4. At the beginning, the training set contained just one training sample for one class (i.e. 36 samples). We then gradually added more samples to the training set (for misrecognized characters) and repeated the training and testing.

In Table 5 we present the recognition rates achieved by different CNN architectures (Table 3). The same training samples (training set 1) as in our previous experiments were used to train the CNNs.

As can be seen, competitive results were achieved only by a simple CNN with one convolutional (C1) and one pooling layer (S2) for a small number of training samples (36, 46). This indicates a relatively good ability to generalize from a small number of training samples. However, as the number of training samples increased (56, 62), the recognition rate decreased! This may point to the fact that the CNN could not effectively extract features that distinguish similar characters (8-B, D-O, O-0) and they are confused. Table 6 lists the most frequently occurring misclassifications.

Table 4. Recognition rates achieved by some of our previous experiments

Feature vector	**Classifier**	**Number of training samples**			
		36	**46**	**56**	**62**
		vs. recognition rate			
Bitmap	Nearest Neighbour	350	380	386	**392**
	Multilayer Perceptron	348	372	382	384
Structural features	Nearest Neighbour	384	385	385	382
Template matching, Euclidean distance (tested image resized to fit training sample bounding box)		379	386	388	**392**

Table 5. Recognition rates achieved by various CNN architectures (training set 1)

CNN architecture	**Number of training samples**			
	36	**46**	**56**	**62**
	vs. recognition rate			
CNN_1.1_4 (C1/4, S2, F3-4)	364	386	381	382
CNN_1.1_8 (C1/8, S2, F3-4)	364	382	382	381
CNN_1.2_4 (C1-2/4, S3, F4-5)	362	371	375	376
CNN_1.2_8 (C1-2/8, S3, F4-5)	360	377	379	376
CNN_2.1_4 (C1/4, S2, C3/4, S4, F5-6)	326	343	342	344
CNN_2.1_8 (C1/8, S2, C3/8, S4, F5-6)	335	349	351	**338**
CNN_2.2_4 (C1/4, S2, C3/8, S4, F5-6)	328	354	341	341
CNN_2.2_8 (C1/8, S2, C3/16, S4, F5-6)	342	374	377	369

(Unsatisfactory results compared to Nearest Neighbor classifier results from Table 4 are marked in red)

Table 6. The most frequently occurring misclassifications (training set 1)

CNN architecture	Number of training samples			
	36	46	56	62
CNN_1.1_4	**8B**(6), **DO**(6), 0D(3)	?B(1), 6G(1), 0O(1)	?B(2), DO(2), 0O(2)	?B(2), 2Z(2), 7T(2)
CNN_1.1_8	**DO**(6), 8B(4), 0D(3)	0G(1), ?B(1), 68(1)	?A(3), DO(2), 6G(1)	?B(4), ?A(3), ?D(1)
CNN_2.1_4	**8B**(8), **4A**(5), 39(4)	8B(4), 5?(3), 68(3)	**?A**(7), **56**(6), **8B**(6)	**8B**(13), **4A**(7), **56**(5)
CNN_2.1_8	**8B**(17), **DO**(6), 0Q(3)	DO(2), 0O(2), 8B(2)	DO(2), 0O(2), 8B(2)	4A(4), 8B(3), 5G(3)

("?" stands for a case when the activation of the winning neuron in the output layer is less than 0.1)

In the next test, we started with the same test set of 36 samples as in training set 1, but we added that character sample that CNN misclassified (Table 7).

As with training set 1, training set 2 also performed better for the simpler CNN architecture with one convolutional (C1) and one pooling layer (S2). As the number of training samples increases, CNN's recognition rate improves, but it did not achieve 100 % recognition rate.

The differences in recognition rates (shown in Table 5 and Table 7) when using different training sets also indicate that there are different representative samples for different classifiers (i.e., the samples needed for optimal MLP training are not necessarily the optimal ones for CNN training).

Table 7. Recognition rates achieved by various CNN architectures (training set 2)

CNN architecture	Number of training samples			
	36	46	56	62
	vs. recognition rate			
CNN_1.1_4 (C1/4, S2, F3-4)	364	384	388	391
CNN_1.1_8 (C1/8, S2, F3-4)	364	380	390	391
CNN_1.2_4 (C1-2/4, S3, F4-5)	362	380	382	389
CNN_1.2_8 (C1-2/8, S3, F4-5)	360	378	387	388
CNN_2.1_4 (C1/4, S2, C3/4, S4, F5-6)	326	355	358	363
CNN_2.1_8 (C1/8, S2, C3/8, S4, F5-6)	335	350	367	378
CNN_2.2_4 (C1/4, S2, C3/8, S4, F5-6)	328	365	385	381
CNN_2.2_8 (C1/8, S2, C3/16, S4, F5-6)	342	372	387	387

The overall recognition accuracy of the CNN could also be affected by the small number of training samples (36–62) from which the CNN learned to extract suitable features.

CNN was implemented in Object Pascal using Intel® oneAPI Deep Neural Network Library (oneDNN). There are currently several deep learning frameworks available for different programming languages (such as TensorFlow, Keras, PyTorch, The Microsoft Cognitive Toolkit, Caffe, Deep Learning Toolbox for MatLab).

6. Conclusion

Nowadays, neural networks are widely used for a variety of tasks related to information processing, image processing, the recognition of complex signals, facial and character recognition, and tasks of an optimization and prediction nature (such as stock market prediction, weather forecasting). They can also be utilized in areas such as diagnostics of mechatronic system nodes [13] or in industrial robotics [10] . Nevertheless, neural networks are still finding new areas of application.

We presented a convolutional neural network as a tool for image classification. Convolutional neural networks integrate the capability of feature extraction and classification, and during training, they learn how to efficiently extract features from input images. We tested different CNN architectures and their impacts on the recognition accuracy of machine written characters taken from vehicle registration plates. Our tests showed that for small input images (12×17) the best results were achieved by the simplest convolutional neural network architecture with one convolutional and one pooling layer.

In the further research, it will be necessary to expand both the training and testing sets with changed types of characters (both letters and numbers) that are used on the new vehicle registration plates.

Acknowledgements

This contribution was prepared within the framework of the project KEGA 006STU-4/2021: "Progressive form of interdisciplinary education and support for the development of the study of vocational subjects in the university environment".

References:

[1]. Bora, M. B., Daimary, D., Amitab, K. & Kandar, D. (2020). Handwritten Character Recognition from Images using CNN-ECOC. *Procedia Computer Science*, *167*, 2403–2409. Doi: 10.1016/j.procs.2020.03.293.

[2]. Hamad, K. & Kaya, M. (2016). A Detailed Analysis of Optical Character Recognition Technology. *International Journal of Applied Mathematics, Electronics and Computers, 4*, 244–249. Doi: 10.18100/ijamec.270374.

[3]. Chowdhury, R. R., Hossain, M. S., Islam, R., Andersson, K. & Hossain, S. (2019). Bangla Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation. *Joint 8th International Conference on Informatics, Electronics & Vision* and 2019 *3rd International Conference on Imaging, Vision & Pattern Recognition*, 318–323. Doi: 10.1109/ICIEV.2019.8858545.

[4]. Kafunah, J. (2016). *Backpropagation In Convolutional Neural Networks*. Retrieved from: https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/ [accessed: 15 February 2023].

[5]. Karrach, L. & Pivarčiová, E. (2019). Location of vehicle registration plates in real scene images. *Acta Facultatis Technicae*, 63–70.

[6]. Karrach, L. & Pivarčiová, E. (2020). Comparative study of feature extraction and classification methods for recognition of characters taken from vehicle registration plates. *The Imaging Science Journal*, *68*, 56–68. Doi: 10.1080/13682199.2020.1719748.

[7]. Khan, N. H. & Adnan, A. (2018). Urdu Optical Character Recognition Systems: Present Contributions and Future Directions. IEEE *Access* 6, 46019–46046. Doi: 10.1109/ACCESS.2018.2865532.

[8]. Khandokar, I. et al. (2021). Handwritten character recognition using convolutional neural network. *Journal of Physics: Conference Series*, *1918*. Doi: 10.1088/1742-6596/1918/4/042152.

[9]. Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM , 60*(6), 84–90. Doi: 10.1145/3065386.

[10]. Kuric, I., Tlach, V., Sagova, Z., Cisar, M. & Gritsuk, I. (2019). Measurement of industrial robot pose repeatability. *Innovative technologies in engineering production*, *244*. Doi: 10.1051/matecconf/201824401015.

[11]. Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1*(4), 541–551. Doi: 10.1162/neco.1989.1.4.541.

[12]. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. Doi: 10.1109/5.726791.

[13]. Peterka, J., Nikitin, Y.R. & Bozek, P. (2020). Diagnostics of automated technological devices. *MM Science Journal*, 4027–4034. Doi: 10.17973/MMSJ.2020_10_2020051.

[14]. Saqib, N., Haque, K.F., Yanambaka, V.P. & Abdelgawad, A. (2022). Convolutional-Neural-Network-Based Handwritten Character Recognition: An Approach with Massive Multisource Data. *Algorithms*, *15*(4), 129. Doi: 10.3390/a15040129.

[15]. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. Doi: 10.48550/arXiv.1409.1556.

[16]. Solai, P. (2018). *Convolutions and Backpropagations.* Pavisj.medium. Retrieved from: https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c [accessed: 20 May 2023]

[17]. *CS231n: Convolutional Neural Networks for Visual Recognition*. (n.d.). Cs231n. Retrieved from: https://cs231n.github.io/convolutional-networks/ [accessed: 18 March 2023]

[18]. Zeiler, M. D. & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. *arXiv preprint arxiv.1311.2901*. Doi: 10.48550/arXiv.1311.2901

[19]. Zhang, Y. (2015). Deep convolutional network for handwritten Chinese character recognition. *Computer Science Department, Stanford University.* Retrieved from: http://cs231n.stanford.edu/reports/2015/pdfs/zyh_project.pdf [accessed: 20 March 2023]