Marcin KOŁODZIEJ[1], Ernest SZYPUŁA[2], Andrzej MAJKOWSKI[3], Remigiusz Rak[4]

Warsaw University of Technology
ORCID: 1. 0000-0003-2856-7298 3. 0000-0002-6557-836X; 4. 0000-0003-0674-453X

doi:10.15199/48.2022.06.06

Using deep learning to recognize the sign alphabet

***Abstract**. This article describes a vision system that uses deep learning to recognize 24 static signs of the American Sign Alphabet in real time. As part of the project, images of signs from four publicly available databases were used as a training set. A DenseNet was implemented for image recognition. For testing, images were acquired with the use of a web camera. The accuracy of sign recognition in images is more than 80%. The real-time version of the system was implemented.*

***Streszczenie.** Artykuł zawiera opis systemu wizyjnego wykorzystującego uczenie głębokie do rozpoznawania, w czasie rzeczywistym 24 statycznych znaków Amerykańskiego Alfabetu Migowego. W ramach realizacji projektu, w charakterze zbioru uczącego, wykorzystano obrazy znaków pochodzące z czterech ogólnodostępnych baz danych. Zastosowano sieć DenseNet do rozpoznawania obrazów. Do testów stworzono własne obrazy z wykorzystaniem kamery internetowej. Skuteczność rozpoznawania znaków migowych z wykorzystaniem obrazów przekroczyła 80%. Zaimplementowano wersję systemu pracującą w czasie rzeczywistym- (**Wykorzystanie uczenia głębokiego do rozpoznawania alfabetu migowego**).*

Keywords: deep learning, convolutional neural network, CNN, sign alphabet.
Słowa kluczowe: uczenie głębokie, sieci konwolucyjne, CNN, język migowy.

Introduction

According to the World Health Organization, there are approximately 466 million people in the world who suffer from severe hearing problems [1]. Most of them use the sign language as their primary communication mechanism. There are more than 300 kinds of sign language in the world. The dominant language among the deaf community in many countries is the American Sign Language. It is a natural language that contains 26 characters known as the American Finger Alphabet that represents each letter. All characters except "j" and "z" are static, displayed with one hand. Each sign can be described by five parameters: shape, orientation, position of the hand, movement, and expression.

The aim of the research on automatic sign language recognition is to develop a convenient and generally accessible system that will facilitate communication between deaf people and hearing people. Research follows two paths, one involves the use of computer vision, and the other involves the use of a set of sensors that respond to hand movements. In recent years, convolutional neural networks have found a very large application in vision systems for image recognition and analysis. In work [2] the development and practical implementation of an American Sign Language alphabet fingerspelling translator based on a convolutional neural network were presented. The *GoogleNet* architecture trained on the ILSVRC2012 and the Surrey University and Massey University ASL datasets was utilized to apply transfer learning to this task. A robust model was developed that consistently classifies the letters ‘a-e’ correctly with first-time users and correctly classifies the letters ‘a-k’ in most cases. A validation accuracy of nearly 98% with five letters and 74% with ten letters was achieved. In line with recent advances in the field of deep learning, the authors of [3] also present a method for using deep convolutional networks to classify letter and digit images of the American Sign Language. The best accuracy achieved was 82,5%. The authors of [4] proposed a new model to recognize the American Sign Language alphabet from images. The input colour images (RGB) for the training were resized and preprocessed before training the deep neural network. The *SqueezeNet* was trained to run on mobile devices with an accuracy of 83.29%.

In more advanced studies, to obtain high precision, special cameras and depth sensors, such as Microsoft Kinect are used. In the work [5] an interactive user interface is presented for hand shape recognition for American Sign Language finger spelling. The Microsoft Kinect device is used to collect images, and the *OpenNI+NITE* framework for hand detection and tracking. Hand-shapes corresponding to the ASL alphabet are recognized on registered images and classified using random forests. The authors compare classification results using normal and depth images and show a combination of both leading to the best results. This system works in real-time, allows the signer to select between ambiguous detections, and is integrated with an English dictionary for efficient writing. The precision achieved was about 75%. In the work [5] a relatively large number of classes is considered compared to the previous literature. CNN is trained for the classification of 31 characters using a subset of depth data collected from multiple subjects. While using different learning configurations, such as hyperparameter selection with and without validation, they achieved 99.99% accuracy for observed signers and 83.58% to 85.49% accuracy for quite new signers. The result shows that the accuracy improves when more data from different subjects are included during the training. The prediction of a single image took 3 ms.

The aim of the article is to compare the operation of two CNN structures for the purpose of recognizing signs of the sign alphabet. Two types of network were compared: conventional CNN and *DenseNet*. To train the network, a training set consisting of four publicly available databases was created. As a result, a program was developed that recognizes 24 characters of the sign alphabet in real time. The program uses an ordinary web camera and allows for effective recognition of the image representing the sign.

Materials

As part of this project, images from four open databases were used to build a training set containing 24 signs representing the static letters of the American Sign Alphabet (ASL):

- ASL Gesture Dataset - 2296 photos with a resolution of 3968×2976 pixels, available on the GITHUB platform [6].
- ASL Alphabet Static - 1778 photos with a resolution of 720×480 pixels, available on the KAGGLE platform [7].
- ASL sign language pictures - 8442 photos with a resolution of 1920×1920 pixels, available on the KAGGLE platform [8].
- ASL Alphabet - 87,000 photos with a resolution of 200×200 pixels, available on the KAGGLE platform [9].

The photos were scaled to a resolution of 100×100 pixels. In total, the training set contains 7917 photos (about 300 photos per character). Figure 1 shows some examples of photos from the training set.

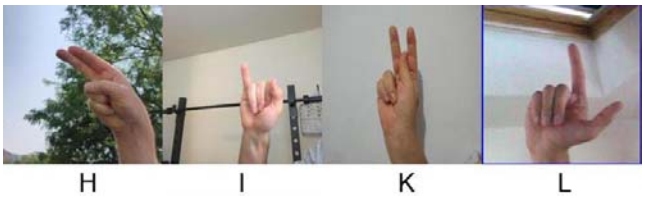

Fig. 1. Examples of signs from the training set

In order to verify the learning process, two additional sets were created by authors with the use of a web camera: validation and testing. Each collection contains 720 images. Figure 2 shows sample photos of the validation and test sets.

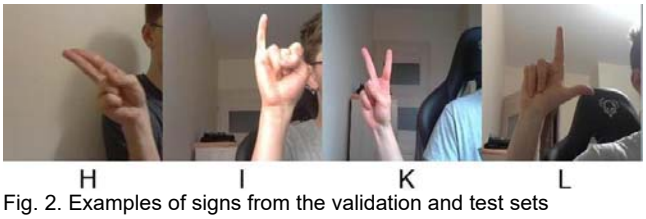

Fig. 2. Examples of signs from the validation and test sets

Methods

The Python language, which contains many deep learning libraries, was chosen to create the software. The *TensorFlow* library and its *Keras* high-level *API* were used to create and train the models. Furthermore, the *OpenCV* library was utilized to preprocess the images. In addition, the following packages were employed: *ScikitLearn* - a machine learning module containing functions to assess the quality of learning, *Matplotlib* - for data visualizations and *Seaborn* - *Matplotlib* overlays - for presentations.

Convolutional networks are commonly used to recognize and classify images. In their commonly known form, they consist of a convolutional layer, a pooling layer, and a fully connected layer. The most common input to the network is a tensor that represents an image with three colour channels. The convolutional layer is a filter known from computer vision, the parameters of which are calculated in the learning process [10]. Deeper convolutional layers can reveal more complex features. The operation of the pooling layer is to replace the data from the filter area most often by the maximum value or, less frequently, by the average value and reduce the size of the input image. The pooling layer is placed after the convolutional layer. After several such sets of layers, a flatten layer is applied to create a vector from the image. This is followed by a fully connected layer that acts as a classifier.

Table 1. Used CNN architecture

Layer	Details
Convolution	3×3Conv2D, stride1
Pooling	2×2MaxPool, stride2
Convolution	3×3Conv2D, stride1
Pooling	2×2MaxPool, stride2
Convolution	3×3Conv2D, stride1
Pooling	2×2MaxPool, stride2
Convolution	3×3Conv2D, stride1
Pooling	2×2MaxPool, stride2
Classification Layer	GlobalAveragePool2D
	24Dfully − connected, softmax

During the research, we trained and tested a CNN, the structure of which is presented in Table 1. A ReLU was applied after each of the network layers. With an increasing number of layers, the gradient vanishing problem appears [11]. This problem can be partly solved by simplifying the interconnection patterns between layers in networks such as *HighwayNetworks* [12], *ResidualNetworks* [13], and *FractalNetworks* [14]. On the basis of the above networks, the structure of *DenseNet* was proposed.

The most popular *DenseNet* architecture consists of a *DenseBlocks* and a transition layers. *DenseBlock* consists of several convolutional layers. Each convolutional layer gets a feature map from all previous layers, which means that the number of channels may be smaller, and thus the network has fewer parameters to learn. A transition layer is placed between the *DenseBlocks* and is used to reduce the size of the feature map. It consists of a convolution layer of size one, which reduces the number of channels, and a *maxpooling layer*, which reduces the size of the image. Finally, the *GlobalAvaragePool2D* layer is used together with a *softmax* classifier. The biggest advantages of implementing *DenseNet* are: strong gradient flow, greater variety of features in each layer, significantly fewer parameters to learn, and keeping low-complexity features on deeper layers. The architecture of *DenseNet121* used by us is described in Table 2.

Table 2. DenseNet121 Architecture

Layer	Details
Convolution	7×7Conv2D, stride2
Pooling	3×3M axPool, stride2
Dense Block	(1×1Conv2D, 3×3Conv2D) x 6
Transition Layer	1×1Conv2D
	2×2AveragePool, stride2
Dense Block	(1×1Conv2D, 3×3Conv2D) × 12
Transition Layer	1×1Conv2D,
	2×2AveragePool stride2
Dense Block	(1×1Conv2D, 3×3Conv2D) × 24
Transition Layer	1×1Conv2D
	2×2AveragePool, stride2
Dense Block	(1×1Conv2D, 3×3Conv2D) ×16
Transition Layer	1×1Conv2D
	2×2AveragePool, stride2
Clasyfication Layer	GlobalAveragePool2D
	24Dfully − connected, softmax

Neural networks are trained to perform well on data that was not placed in the training set. To increase network generalization, the following techniques can be used: early stop, dropout, and augmentation [15]. Using the validation set, we are able to decide when a given network achieves the best results, so we can stop learning the network when the target function increases drastically (early stop). The dropout means that during the training process, some randomly selected layer outputs are reset to zero in each iteration. The dropout factor determines the number of layer outputs to be reset and is predetermined in advance [16]. Augmentation of the image set means the use of various transformations such as rotation, shift, zoom, clipping, and colour operations to increase the training set. Each image is transformed randomly. Two methods of input augmentation were used to train neural networks in order to increase efficiency: *RandomRotation, which is* responsible for rotating the image at a random angle, and *RandomZoom, which is* responsible for zooming in or out of the image.

Results and discussion

In our experiment, a standard convolutional neural network (Table 1) was found to be unable to achieve more than 60% accuracy. This result is not satisfactory.

Therefore, it was decided to use a more complex structure of *DenseNet121*. The loss function is shown in Fig. 3. It is the categorical cross entropy [17].

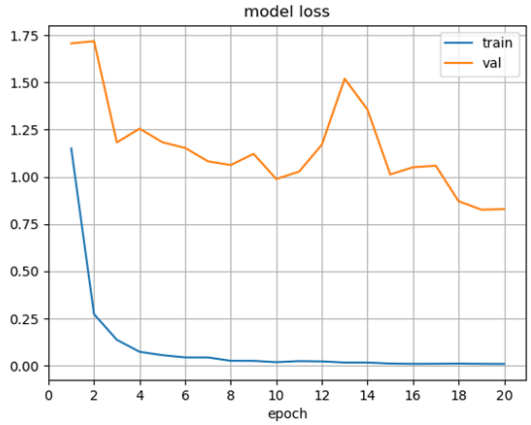

Fig. 3. Loss function for the train and validation set

For *DenseNet121* we achieved an accuracy of around 80% (Fig. 4), which was 20% better than the best result for the convolutional network used previously.

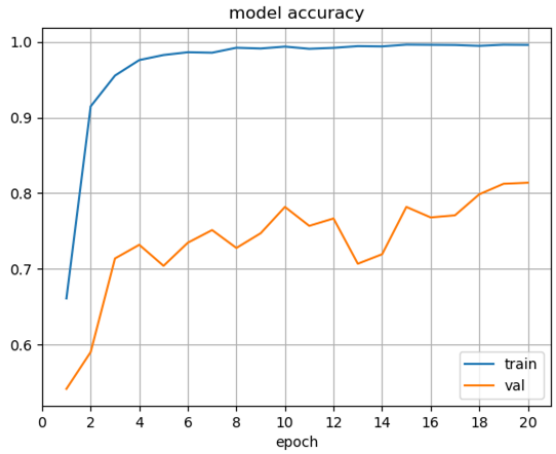

Fig. 4. Accuracy for the train and validation set

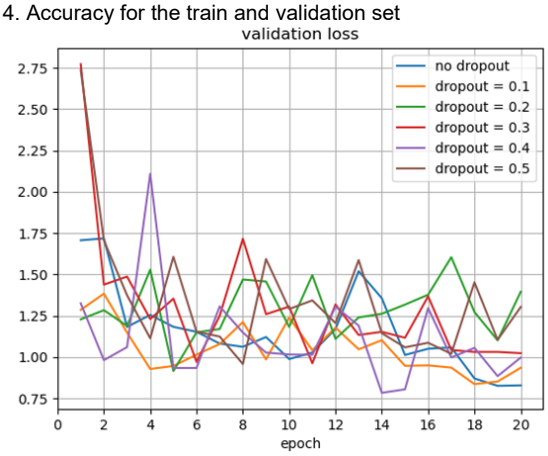

Fig. 5. Loss function for the validation set

In order to reduce the variance of the model, a dropout was used. To determine the value of the dropout factor, the network was trained for dropout values ranging from 0.1 to 0.5. The loss function (Fig. 5) reaches the minimum value for the dropout value of 0.4. The network achieves the best accuracy of 83% in the 14th epoch (Fig. 6).

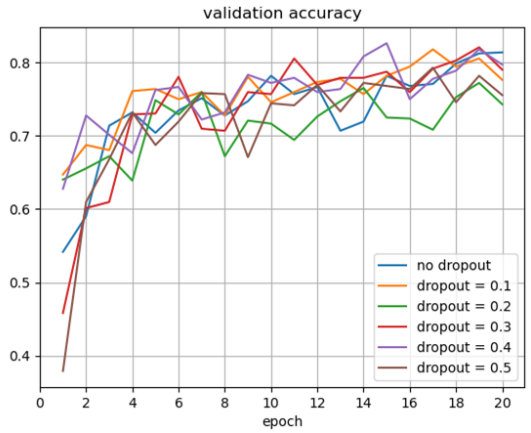

Fig. 6. Accuracy for the validation set

Table 2 presents a comparison of the results achieved with the results published in the literature. The data presented show that the quality of the designed model does not differ significantly from the quality of other models using an image without a depth map. Differences in accuracy may result from differences in training and testing data sets, as there is no single uniform data set that is used in all research on the problem of sign language recognition. Models using the depth map are much more accurate, but require specialized cameras.

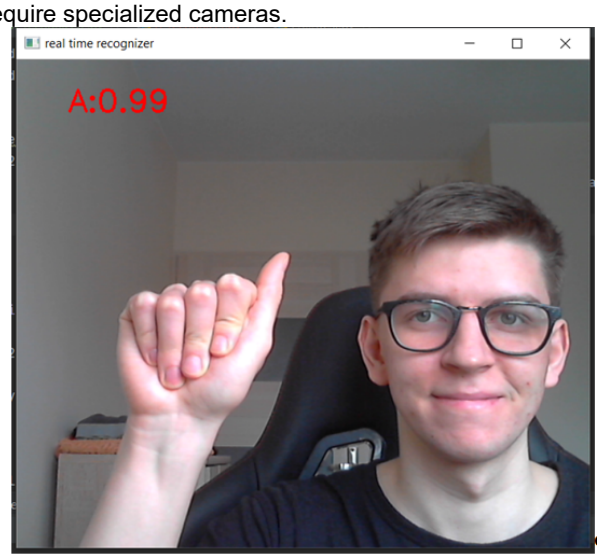

Fig. 7. Program operation recognizing the character "A"

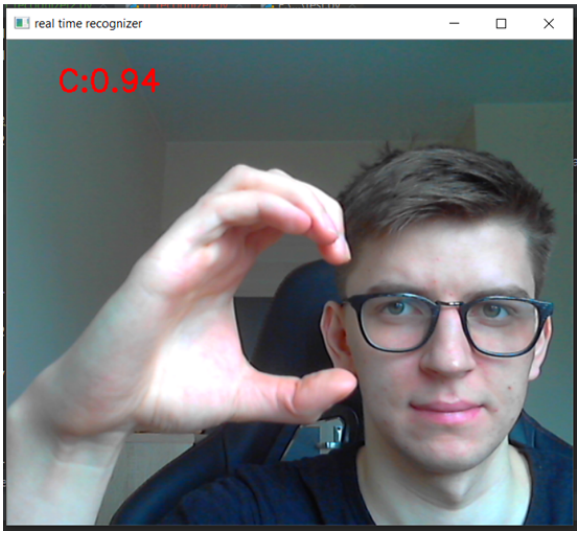

Fig. 8. Program operation recognizing the character "C"

Table 3. Accuracy comparison with other studies

Work	Picture type	Accuracy [%]
[18]	RGB	72
[4]	RGBD	75
Author	RGB	80.69
[2]	RGB	82.5
[3]	RGB	90.3
[5]	depth map	99.9

The confusion matrix (Table 4) shows that *DenseNet121* works very well for the test data. Referring to the full set of photos, it can be easily noticed that mistakes occur for such letters as: [C, F, O], [E, S], [M, N], [L, T], [K, V]. These mistakes result from the high similarity of the sign images for these letters. Slight differences in hand position, camera angle, or lighting can cause misinterpretation.

A program was developed that recognizes the signs of the sign alphabet in real time. The application uses a standard webcam and DenseNet121 trained to recognize the presented sign. The software, on a laptop with an NVIDIA GTX1050 graphics card, recognizes signs at a speed of 20 frames per second. As there were no examples classifying the lack of a hand in a photo in the training dataset, the model was protected against photos without a hand by implementing the hand detector in the image from the mediapipe library. Figures 7 and 8 show how the program works. For the recorded image, the program detects the presented sign, writes out the character, and the probability of proper recension of the sign.

Table 4: Confusion matrix for the test set

		PREDICTED																						
		A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
ACTUAL	A	30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	B	-	30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	C	-	-	30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	D	-	-	1	28	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	E	-	-	-	-	22	2	-	-	-	-	-	1	1	-	-	-	-	3	-	-	-	1	-	-
	F	-	-	8	-	-	17	-	-	-	2	-	-	-	-	2	-	-	-	-	-	-	-	-	1
	G	-	-	-	-	-	-	29	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
	H	-	-	-	-	-	-	-	30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	I	-	-	-	-	-	-	-	-	29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
	K	-	-	-	-	-	-	-	-	-	25	-	-	-	-	-	-	-	-	-	-	5	-	-	-
	L	-	-	-	-	-	-	-	-	-	-	30	-	-	-	-	-	-	-	-	-	-	-	-	-
	M	1	-	-	3	-	-	-	-	-	-	-	17	9	-	-	-	-	-	-	-	-	-	-	-
	N	1	-	2	-	-	1	-	-	-	-	-	2	21	2	-	1	-	-	-	-	-	-	-	-
	O	-	-	11	-	-	-	-	-	-	-	-	-	-	19	-	-	-	-	-	-	-	-	-	-
	P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	27	3	-	-	-	-	-	-	-	-
	Q	-	-	-	-	-	-	-	-	-	-	-	-	-	3		27	-	-	-	-	-	-	-	-
	R	-	-	-	3	-	-	-	-	1	1	-	-	-	-	-	-	23	-	-	2	-	-	-	-
	S	2	-	-	-	14	-	-	-	-	-	-	-	-	-	-	-	-	7	7	-	-	-	-	-
	T	3	-	-	-	-	1	-	-	-	-	9	-	1	-	1	-	-	-	11	-	-	-	3	1
	U	-	-	-	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	27	1	-	-	-
	V	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	25	-	-	-
	W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	30	-	-
	X	-	-	-	8	-	-	-	-	2	-	-	-	-	-	-	1	-	-	-	-	-	-	18	1
	Y	-	-	-	-	-	-	-	-	6	-	1	-	-	-	-	-	-	-	-	-	-	-	-	23

Conclusions

The problem of recognizing the Sign Language alphabet has not yet been perfectly solved. Research is still underway to create more accurate and accessible methods. A big problem with the use of computer vision is the lack of a single data set to test and learn neural networks. In the course of our research, we used publicly available image collections that are not perfect, as most of the photos were taken in the process of one session and with the same background.

We have noticed that the accuracy of the sign recognition depends on the percentage of the area covered by the hand - the smaller this area, the lower the accuracy of the classification. Lighting is also of great importance for our solution, as it can deteriorate or improve the results of recognizing a sign. We obtained the accuracy of recognizing 24 signs of the sign alphabet of 80.7%. The solution was tested in real conditions. The results give hope for supporting sign language recognition by computer systems.

***Authors**: dr hab. inż. Marcin Kołodziej, inż. Ernest Szypuła, dr hab. inż. Andrzej Majkowski, prof. dr hab. Inż. Remigiusz Rak, Warsaw University of Technology, pl. Politechniki 1, 00-661 Warszawa, marcin.kolodziej@.pw.edu.pl, andrzej.majkowski@pw.edu.pl remigiusz.rak@pw.edu.pl.*

REFERENCES

[1] „Deafness and hearing loss". https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss (access: December 3, 2021).

[2] V. Bheda i D. Radpour, „Using Deep Convolutional Networks for Gesture Recognition in American Sign Language", *ArXiv171006836 Cs*, lis. 2017, access: December 3, 2021. [Online]. Dostępne na: http://arxiv.org/abs/1710.06836

[3] R. Daroya, D. Peralta, i P. Naval, „Alphabet Sign Language Image Classification Using Deep Learning", *TENCON 2018 - 2018 IEEE Reg. 10 Conf.*, 2018, doi: 10.1109/TENCON.2018.8650241.

[4] N. Pugeault i R. Bowden, „Spelling it Out: Real-Time ASL Fingerspelling Recognition", lis. 2011. https://eprints.gla.ac.uk/219319/ (access: December 3, 2021).

[5] B. Kang, S. Tripathi, i T. Q. Nguyen, „Real-time sign language fingerspelling recognition using convolutional neural networks from depth map", *2015 3rd IAPR Asian Conf. Pattern Recognit. ACPR*, 2015, doi: 10.1109/ACPR.2015.7486481.

[6] T. Northall-Little, *ASL-Gesture-Dataset*. 2018. access: December 3, 2021. [Online]. Dostępne na: https://github.com/tomnlittle/ASL-Gesture-Dataset

[7] „American Sign Language Alphabet (Static)". https://kaggle.com/jordiviader/american-sign-language-alphabet-static (access: December 3, 2021).

[8] „ASL Sign Language Alphabet Pictures [Minus J, Z]". https://kaggle.com/signnteam/asl-sign-language-pictures-minus-j-z (access: December 3, 2021).

[9] „ASL Alphabet”. https://kaggle.com/grassknoted/asl-alphabet (access: December 3, 2021).
[10] W. S. Ahmed i A. amir A. Karim, „The Impact of Filter Size and Number of Filters on Classification Accuracy in CNN”, w *2020 International Conference on Computer Science and Software Engineering (CSASE)*, kwi. 2020, s. 88–93. doi: 10.1109/CSASE48920.2020.9142089.
[11] S. Squartini, A. Hussain, i F. Piazza, „Attempting to reduce the vanishing gradient effect through a novel recurrent multiscale architecture”, w *Proceedings of the International Joint Conference on Neural Networks, 2003.*, lip. 2003, t. 4, s. 2819–2824 t.4. doi: 10.1109/IJCNN.2003.1224018.
[12] W. Luo i F. Yu, „Recurrent Highway Networks With Grouped Auxiliary Memory”, *IEEE Access*, t. 7, s. 182037–182049, 2019, doi: 10.1109/ACCESS.2019.2959655.
[13] K. He, X. Zhang, S. Ren, i J. Sun, „Deep Residual Learning for Image Recognition”, w *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, cze. 2016, s. 770–778. doi: 10.1109/CVPR.2016.90.
[14] J. Gu, Z. Qu, X. Wang, J. Dan, i J. Sun, „Residual Fractal Network for Single Image Super Resolution by Widening and Deepening”, w *2020 25th International Conference on Pattern Recognition (ICPR)*, sty. 2021, s. 1596–1603. doi: 10.1109/ICPR48806.2021.9412184.
[15] A. Mikołajczyk i M. Grochowski, „Style transfer-based image synthesis as an efficient regularization technique in deep learning”, w *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, sie. 2019, s. 42–47. doi: 10.1109/MMAR.2019.8864616.
[16] J. Shen i M. O. Shafiq, „Deep Learning Convolutional Neural Networks with Dropout - A Parallel Approach”, w *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, grudz. 2018, s. 572–577. doi: 10.1109/ICMLA.2018.00092.
[17] Y. Kim, S. Kim, T. Kim, i C. Kim, „CNN-Based Semantic Segmentation Using Level Set Loss”, w *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, sty. 2019, s. 1752–1760. doi: 10.1109/WACV.2019.00191.
[18] B. Garcia, „Real-time American Sign Language Recognition with Convolutional Neural Networks”. https://www.semanticscholar.org/paper/Real-time-American-Sign-Language-Recognition-with-Garcia/07a152c14004082a393caa31a6052578570a8b95 (access: December 3, 2021).