The Fusion of two NN Architectures for the Improvement of Image Classification

Tatiana Jaworska
Systems Research Institute
Polish Academy of Sciences
01-447, 6 Newelska Street, Warsaw,
Poland
Tatiana.Jaworska@ibspan.waw.pl
https://orcid.org/0000-0001-5399-8474

***Abstract*— Content-Based Image Retrieval (CBIR) systems require image classification. This paper presents an application of neural networks (NN) to image segment classification for our CBIR system. A new NN architecture which connects a convolutional NN (CNN) with a shallow NN is proposed in order to improve the classification results. This improvement is necessary due to the shortage of image samples to train a CNN separately. We obtained 5-11% improvement thanks to the fusion of CNN and shallow NN.**

Keywords*— *CNN, shallow NN, pattern recognition, CBIR, image processing

I. Introduction

Content-Based Image Retrieval (CBIR) systems consist of different types of search engines, some of which search for image similarity based only on low-level features. Whereas, others can use classified segments or whole images. So far, in our CBIR system several classical classifiers, such as minimal distance, Naïve Bayes, decision trees, fuzzy rule-based have been implemented [1]. However, the rapid development of convolutional neural networks (CNNs) as a tool for image detection and classification encourages us to implement them also in our CBIR system. We have examined a new approach to the classification of existing images in our CBIR using neural networks and the tentative results we included in the research report [2].

The deep learning concept [3] has a long history, beginning from artificial neural networks described by Fukushima in 1980 [4], through LeCun's suggestion in 1989 that the problem of handwriting recognition can be solved by a back propagation network [5] but eventually Krizhevsky et al. [6] in 2012 introduced a deep convolutional neural network (CNN) to image classification. They won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for which they classified images into a thousand different categories, and in order to do this they had trained their CNN based on about 1.2 mln images. Unfortunately, the fewer images in the training set, the worse classification accuracy.

The main problem with applying CNN to the classification of particular image sets is the insufficient number of images and the fact that the classes are unbalanced. In our system there are also many classes and subclasses with various numbers of examples. Hence, we have analysed some pre-trained CNNs and our own architecture of CNN but the results obtained have been unsatisfactory, mainly because of the small number of samples to train networks.

In such a situation the transfer learning method is suggested. It works under two conditions, first the learning task and the target task have the same distribution of probability. Second the amount of the target-domain data must be smaller than in the source domain. It turns out that in our case the first condition is weakly fulfilled so for this reason, we decided to incorporate the numeric features collected in the DB in order to boost the classification results. We have prepared a shallow NN, dedicated to the numeric features analysis for pattern recognition, and combined it with our previously prepared CNNs. This paper describes the CNN architecture and shallow NN, and additionally touches the transfer learning problem.

Our contribution consists in combining CNN and shallow NN architectures thanks to which the total accuracy increases. Table I presents the total accuracy augmentation in comparison to separate accuracies.

II. Classification Problem

A. Classification in CBIR

The house images have been collected in our CBIR system. Each new image is segmented, creating a set of objects. In turn, each object, segmented according to the algorithm presented in detail in [7], is described by a vector of 45 low-level features, such as shape descriptors, colours, area, iteration moments, etc. For classification segments based on the feature vectors, we have implemented two-level classification systems. There are minimal distance, Naïve Bayes, decision trees classifiers at the base level and in the case of an inconsistency in the attribution of classes, the fuzzy rule-based classifier selects the winner only from among the three classes found at the first level.

B. CNN Classification

In contrast to a classification based on a feature vector, a CNN extracts many image features by using a convolution input image with different so-called masks, filters or kernels as a result of which an activation map is produced. In the second layer a Rectified Linear Unit (ReLU) is applied which represents a simplified neuron model with different transfer functions which in turn decide what value is transfered farther to the next layer. And the third layer is MaxPooling which down-samples a feature map and reduces the details in the input layer (reduces the resolution) thanks to which the space layout of the object becomes more important. A deep NN

consists of many such triples of layers – the more of them, the deeper the network is (see **Błąd! Nie można odnaleźć źródła odwołania.**).

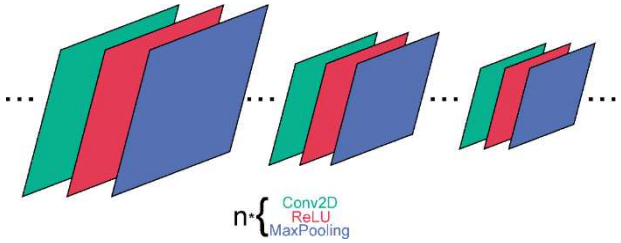

Fig. 1 A CNN deep structure.

We have had 7070 classified segments of 92 classes of architectural elements at our disposal which, additionally, are unbalanced in terms of image number for particular classes. The most numerous class is 'window pane', consisting of 800 samples. The next four classes were less numerous, but sufficient to train the network.

III. *Neural Network Architecture*

In the backpropagation technique, first a calculation of the dot products of input signals and their corresponding weights is propagated forward and then an activation function to these sums of products is applied, in order to transform the input signal into an output one and to introduce non-linearities to the complex model which will enable the model to learn almost any arbitrary functional mappings.

In the next step, error terms are propagated backwards in the network and weight values are updated using gradient descent and the gradient of error function with respect to the weights. The bias values are recalculated and the weights are updated in the opposite direction of the loss function gradient with respect to the model bias.

A transfer function for each neuron is very simple, but the difficulty arises from both the size of the deep neural network and the fact that the model training requires finding the optimal value of a non-convex objective function [8]. The internal parameters of a CNN model play an important role in efficient and effective training of a model and producing accurate results. This is why we have tested various optimization algorithms to update and calculate optimum values of such model's parameters which influence our model's learning process. First, order optimization algorithms minimize (or maximize) a loss function using its gradient values with respect to the parameters. The most widely used first order optimization algorithm is gradient descent. The first order derivative tells us whether the function is decreasing or increasing at a particular point. The first order derivative basically gives us a line tangent at the minimum point on the error surface. Second-order methods we are applied for training of our Shallow NN.

Having assumed that Matlab implementation had been prepared by experts, we examined three standard optimization solvers for our CNN: the Stochastic Gradient Descent with Momentum (SGDM), Root Mean Square Propagation (RMSProp) [8] and ADAptive Moment estimation (ADAM) [9] – all implemented in Matlab.

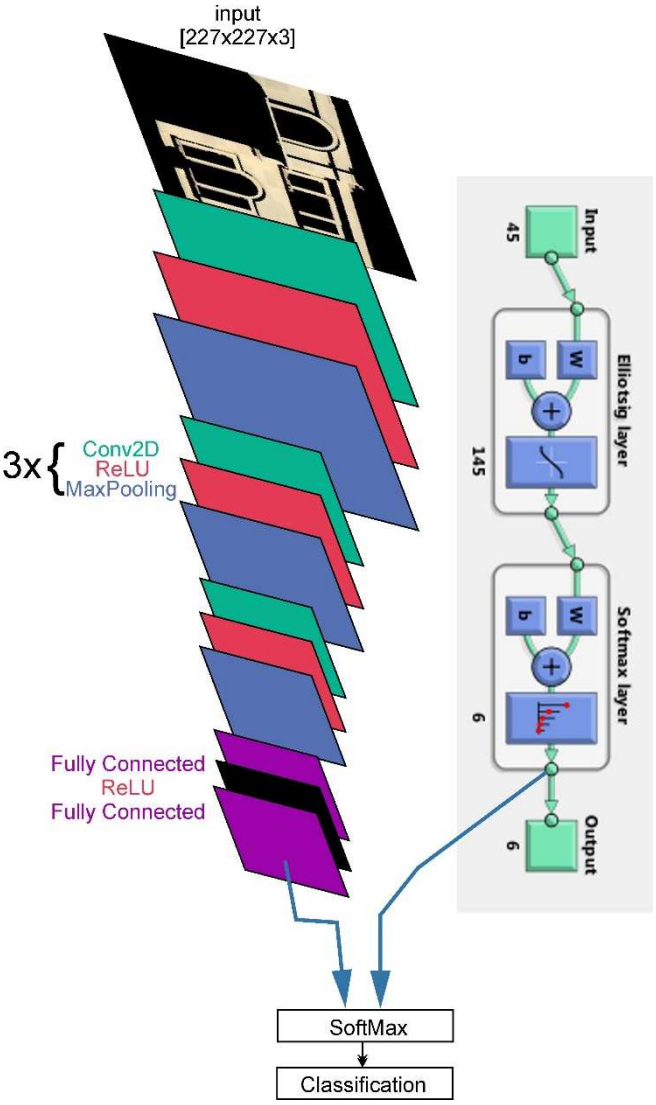

Fig. 2. Our fusion structure of CNN and shallow NN

A. *Fusion of two NN architecture*

The whole system is implemented in MatLab ver. 2018b. Fig. presents an example of a simple CNN consisting of 15 layers applied to image classification [2] and shallow NN applied to the classification of the same images, based on their feature vectors [10]. The product of probabilities coming from both networks gives the winner class. The size of the input vector for the shallow NN is 45, the first hidden layer consists of 145 neurons, each with the elliot symmetric sigmoid transfer function. The second softmax layer generates probabilities for 6 classes.

B. *Training Process*

The CNN and the Shallow NN were trained separately but synchronously in the sense that when an image is put into the CNN first layer, the feature vector for this same image is put into the SNN first layer. Then, from both outputs, we can combine the probabilities for a class of image and a class of image feature vector in the common softmax layer as a diagonal of a cross product. At the fusion output we get the boost class, thanks to which we have obtained an increase in accuracy, which is shown in the result section.

In order to train our CNN we assumed a fixed size of image input 227×227×3 with the zero-centre normalization

for our network. Segment sizes in our DB had an area from 7 to 1 982 449 pixels, depending on the original image resolution. Hence, we had to rescale the smallest segments and divide the biggest ones into two or four blocks. This procedure offers the best segment matching for the input window size because the small elements are not enlarged and the big ones, for instance roofs, are not shrunk more than 10%. Moreover, the division of big elements enlarges the number of samples, which is very advantageous in case of an insufficiency of training elements. Having done this, we trained each of the networks for the 5 classes, of which each exceeds 500 samples.

Our CNN has been trained on a mini-batch size = 128, for all experiments [11]. We tested it with the filter size equal to [3×3], [7×7] and [9×9], at 40 epochs with 55 iterations per epoch, which gives 2200 maximum iterations (see Fig. 3). Details of our approach are described in [2].

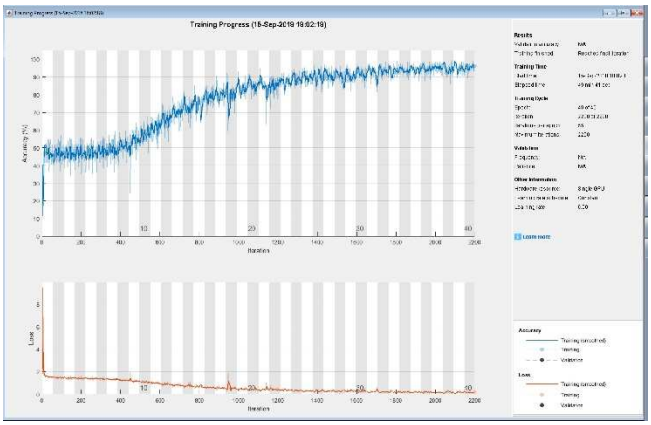

Fig. 3 Example of a training progress for our network with the RMSProp optimization algorithm and filter size [3 3]. (Top) The curve of accuracy (blue) during the training process. (Bottom) The curve of the loss function (red).

In order to train our SNN, first of all, we had normalised the tagged set of 7070 samples in the range [-1 1]. Later, we divided this set into the training part which contains 70% of samples, the validation set which contains 15%, and the testing set which also contains 15% of samples. During experiments we have determined that 47 training epochs is enough.

For our shallow NN we have tested three learning functions with respect to a mean squared error (MSE):

- Levenberg-Marquardt backpropagation that updates weight and bias values according to the Hagan optimization algorithm [12]. The algorithm was designed to approach the second-order training speed without having to compute the Hessian matrix. It is implemented in the Matlab Deep Learning Toolbox as the ‘trainlm’ function.

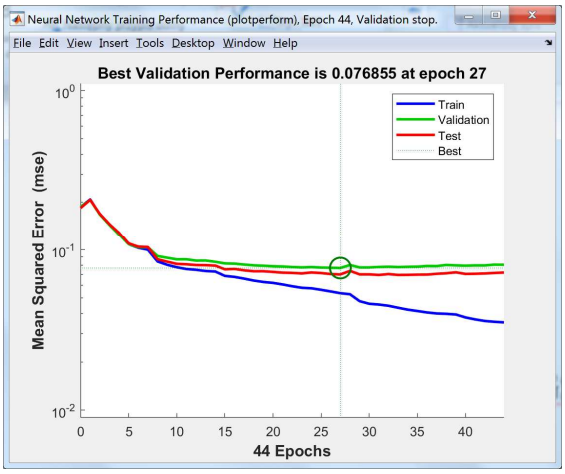

Fig. 4. Mean squared error for the process of training. The best validation performance is marked by the green circle.

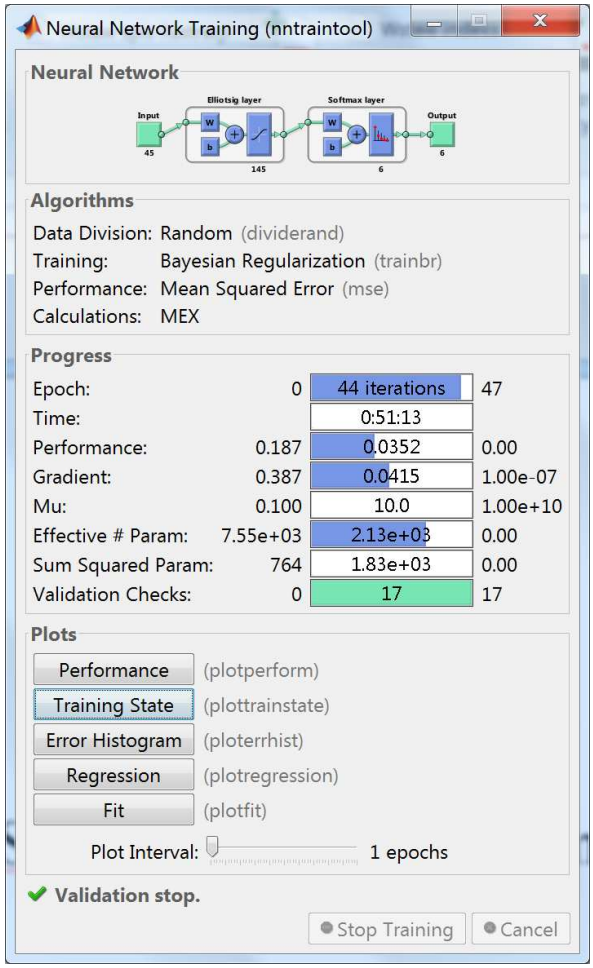

Fig. 5 The neutral network training toolbox.

- Scaled conjugate gradient backpropagation is used to calculate derivatives of performance related to the weight and bias variables *w* and *b*, respectively (see Fig.) [13]. It is implemented in the Matlab Deep Learning Toolbox as the ‘trainscg’ function.
- Bayesian regularization backpropagation works similarly to the Levenberg-Marquardt optimization. It minimizes a combination of squared errors and

weights, and then determines the correct combination so as to produce a network that generalizes effectively. It is implemented in the Matlab Deep Learning Toolbox as the 'trainbr' [14] function.

We apply the neutral network training toolbox in order to easily visualize training parameters (see Fig. 5).

C. *Transfer Learning*

The transfer learning technique assumes that we only fine-tuned the pre-trained network in such a way that we transfer the layers to the new classification task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer. This technique works under the condition that the domain of the source training set and the domain of the target training set have the same distribution of probability We have tested a transfer learning method on the AlexNet which has been trained on over a million images and can classify images into 1000 object categories One example is shown in Table I for the AlexNet [6]. We have fine-tuned this network, using the stochastic gradient descent with momentum with MiniBatchSize = 64 for 10 epochs with 712 iterations each. But the results are similar to our simple CNN, probably because the domains of both training sets differed too much.

In this situation we have cancelled further tests simply because, in this technique, the whole NN from which the transfer is done is in the memory. Since there is a shortage of memory, we decided that it would be better to develop small own CNN.

IV. Results

Our experiments were carried out on NVIDIA GeForce GTX 1050 Ti, with 768 CUDA cores and the graphics memory of 4 GB.

The best results of many experiments have confirmed the selection of the 'trainbr' as the network training function which realizes the Bayesian regularization backpropagation [14].

We tested our network on three sets of segments. Each set contains segments from one image: 'aksamit.jpg' has been divided into 84 segments, 'amadeusz_1.jpg' into 140, and 'Abeeku.jpg' into 55, respectively. Table I sums up the results the best of which shows an increase of accuracy up to 11%.

TABLE I. Examples of the Results Obtained for 5 Classes

Sample Set (image name)	Accuracy		
	CNN accuracy	***Shallow NN accuracy***	***Total Accuracy***
aksamit.jpg Fig. 6, Fig. 7	CNN_RMS =0.5543	0.4457	0.5978
amadeusz_1.jpg Fig. 8, Fig. 9 **Fig. 10, Fig. 11** Fig. 12, Fig. 13	CNN_RMS = 0.5930 **CNN_ADAM = 0.5581** alex_trans_sgdm_MBS_10_16 = 0.5407	0.5349 **0.5349** 0.5349	0.6453 **0.6628** 0.5930
Abeeku.jpg Fig. 14, Fig. 15 Fig. 16, Fig. 17	CNN_RMS =0.4253 CNN_SGDM = 0.4598	0.3793 0.3793	0.4713 0.5057

Below we present the details of the fusion classification only for each image. Examples of classes: 1 - window pane, 3 - frame, 5 - roof, 7 - wall, 9 - roof edge.

Image name = aksamit.jpg

```
CNN RMS_1_3_5_7_9.mat
CNN accuracy = 0.5543
Fusion classification:

class = 7  recall = 0.25
class = 9  precision = 0.22222
class = 9  recall = 0.16667
class = 0  precision = 0.71875
class = 0  recall = 0.7541
class = 1  precision = 0.2
class = 1  recall = 1
class = 5  precision = 0.5
class = 5  recall = 0.5
class = 3  precision = 0.33333
class = 3  recall = 0.25
total precision = 0.39572
total recall = 0.48679
total specificity  = 0.5
Total accuracy = 0.5978
```

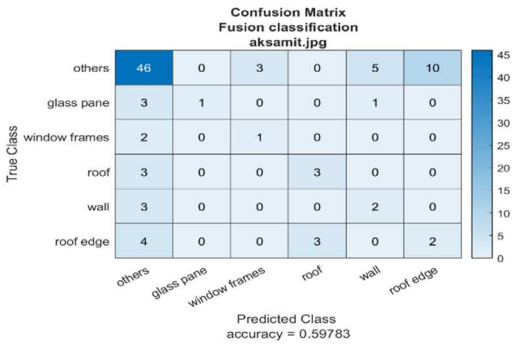

Fig. 6. Confusion matrix of fusion accuracy with the CNN for the RMSProp optimization solver for image name = 'aksamit.jpg'

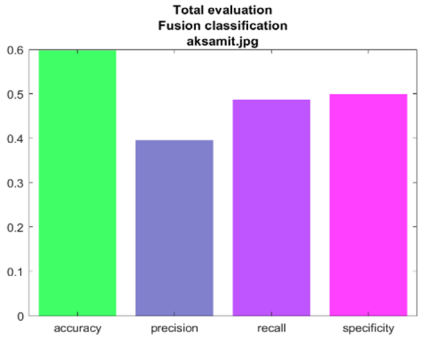

Fig. 7. Total evaluation of fusion classification with the CNN for the RMSProp optimization solver for image name = 'aksamit.jpg'

amadeusz_1.jpg

```
CNN RMS_1_3_5_7_9.mat
CNN accuracy = 0.5930
Fusion classification:

class = 7  precision = 0.6
class = 7  recall = 0.14286
class = 9  precision = 0.75
class = 9  recall = 0.25
class = 0  precision = 0.64615
class = 0  recall = 0.90323
class = 1  precision = 0.44444
class = 1  recall = 0.5
class = 5  precision = 0.75
class = 5  recall = 0.6
class = 3  precision = 0.90909
class = 3  recall = 0.4
total precision = 0.68328
total recall = 0.46601
total specificity  = 0.48837
Total accuracy = 0.6453
```

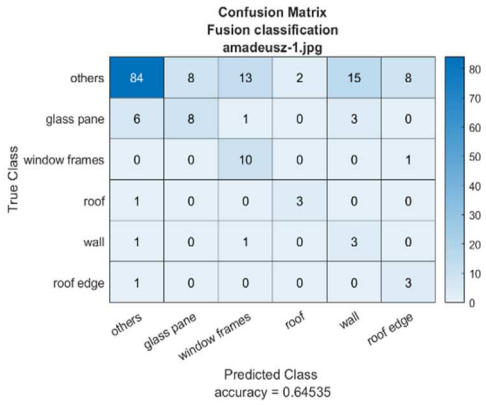

Fig. 8. Confusion matrix of fusion accuracy with the CNN for the RMSProp optimization solver for image name = ‘amadeusz_1.jpg’.

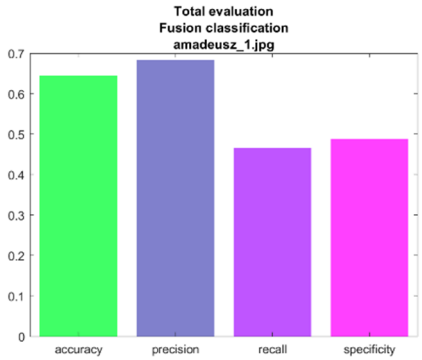

Fig. 9. Total evaluation of fusion classification with the CNN for the RMSProp optimization solver for image name = ‘amadeusz_1.jpg’.

```
CNN ADAM_1_3_5_7_9.mat
CNN accuracy = 0.5581
Fusion classification:

class = 7  precision = 0.57143
class = 7  recall = 0.19048
class = 9  precision = 0.85714
class = 9  recall = 0.5
class = 0  precision = 0.66116
class = 0  recall = 0.86022
class = 1  precision = 0.61111
class = 1  recall = 0.6875
class = 5  precision = 1
class = 5  recall = 0.6
class = 3  precision = 0.625
class = 3  recall = 0.4
total precision = 0.72097
total recall = 0.5397
total specificity  = 0.46512
Total accuracy = 0.6628
```

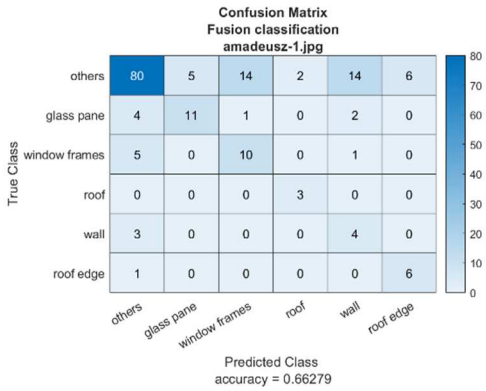

Fig. 10. Confusion matrix of fusion accuracy with the CNN for the ADAM optimization solver for image name = ‘amadeusz_1.jpg’.

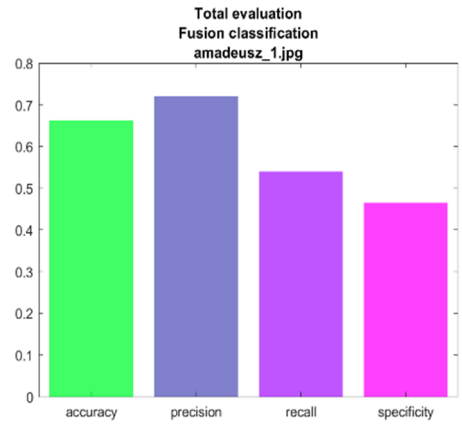

Fig. 11. Total evaluation of fusion classification with the CNN for the ADAM optimization solver for image name = ‘amadeusz_1.jpg’

Below we present the results for a transfer learning example:

```
alex_trans_sgdm_MBS_10_16.mat
CNN accuracy = 0.5407
Fusion classification:

class = 7  precision = 0.33333
class = 7  recall = 0.14286
class = 9  precision = 0.5
class = 9  recall = 0.58333
class = 0  precision = 0.63303
class = 0  recall = 0.74194
class = 1  precision = 0.375
class = 1  recall = 0.375
class = 5  precision = 1
class = 5  recall = 0.6
class = 3  precision = 0.66667
class = 3  recall = 0.56
total precision = 0.58467
total recall = 0.50052
total specificity  = 0.40116
Total accuracy = 0.5930
```

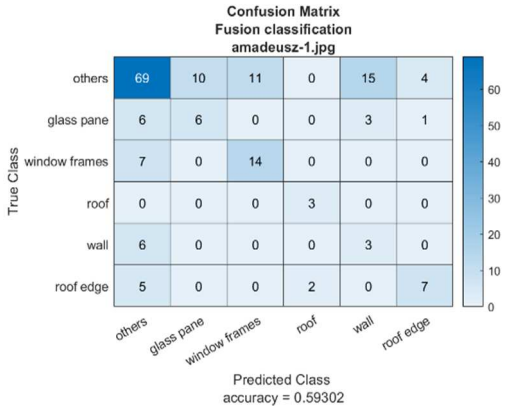

Fig. 12. Confusion matrix of fusion accuracy calculated with transfer learning with the AlexNet CNN for the SGDM optimization solver for image name = ‘amadeusz_1.jpg’.

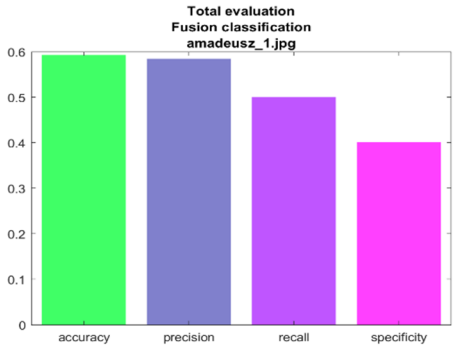

Fig. 13. Total evaluation of fusion classification calculated with transfer learning with the AlexNet CNN for the SGDM optimization solver for image name = ‘amadeusz_1.jpg’.

'Abeeku-house-plan-Small.jpg'

```
CNN RMS_1_3_5_7_9.mat
CNN accuracy = 0.4253
Fusion classification:

class = 7  precision = 0.6
class = 7  recall = 0.46154
class = 9  precision = 0
class = 9  recall = 0
class = 0  precision = 0.41509
class = 0  recall = 0.88
class = 1  precision = 0.25
class = 1  recall = 0.66667
class = 5  precision = 1
class = 5  recall = 0.38462
class = 3  precision = 1
class = 3  recall = 0.4
total precision = 0.54418
total recall = 0.46547
total specificity  = 0.25287
Total accuracy = 0.4713
```

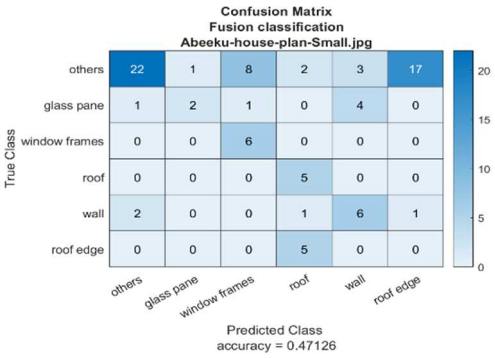

Fig. 14 Confusion matrix of fusion accuracy with the CNN for the RMSProp optimization solver for image name = 'Abeeku-house-plan-Small.jpg'.

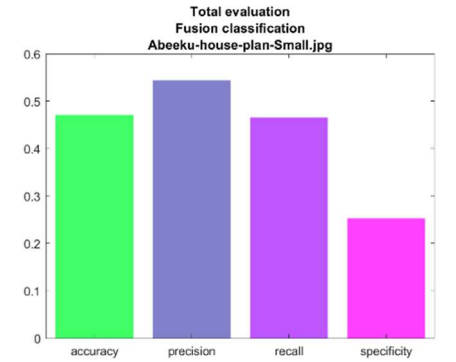

Fig. 15. Total evaluation of fusion accuracy with the CNN for the RMSProp optimization solver for image name = 'Abeeku-house-plan-Small.jpg'.

```
CNN SGDM_1_3_5_7_9.mat
CNN accuracy = 0.4598
Fusion classification:

class = 7  precision = 0.875
class = 7  recall = 0.53846
class = 9  precision = 0.4
class = 9  recall = 0.22222
class = 0  precision = 0.44681
class = 0  recall = 0.84
class = 1  precision = 0.1
class = 1  recall = 0.33333
class = 5  precision = 1
class = 5  recall = 0.38462
class = 3  precision = 0.85714
class = 3  recall = 0.4
total precision = 0.61316
total recall = 0.45311
total specificity  = 0.24138
Total accuracy = 0.5057
```

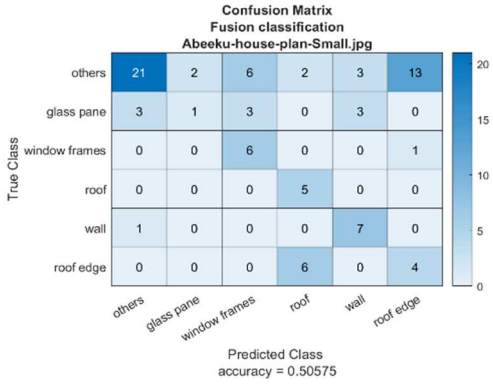

Fig. 16. Confusion matrix of fusion accuracy with the CNN for the SGDM optimization solver for image name = 'Abeeku-house-plan-Small.jpg'.

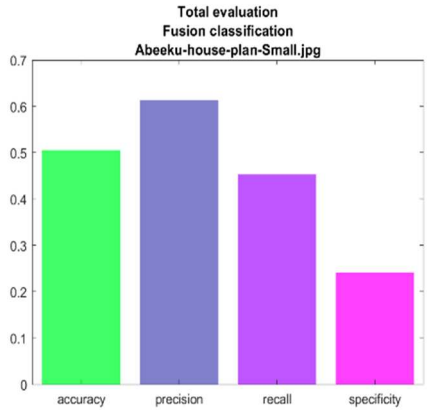

Fig. 17. Total evaluation of fusion accuracy with the CNN for the SGDM optimization solver for image name = 'Abeeku-house-plan-Small.jpg'.

V. Discussion and Further Works

Our solution can be widely applied in all the systems that consist of images and alphanumeric data, for instance, in patient examination DBs, especially when medical image-based classification is combined with the results of other tests (blood, biochemical, etc.).

New architecture of CNN, as well as Shallow NN might improve accuracy, so we will work further in this direction. We are going to try to test our solution on medical images and data.

References

[1] T. Jaworska, “Classification problem in CBIR,” *Journal of Theoretical and Applied Computer Science*, vol. 7, no. 2, pp. 3-15, 2013.

[2] T. Jaworska, “CNN as a Tool for Image Segment Classification,” Warsaw, Dec., 2018.

[3] Q. Abbas, M. E. A. Ibrahim and M. . A. Jaffar, “A comprehensive review of recent advances on deep vision systems,” *Artificial Intelligence Review*, vol. 52, pp. 39-76, May on-line 2018.

[4] K. Fukushima, “Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193-202, April 1980.

[5] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard i L. D. Jackel, „Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation*, tom 1, nr 4, pp. 541-551, Winter 1989.

[6] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - NIPS'12*, Lake Tahoe, Nevada, USA, 03 - 06 Dec. 2012.

[7] T. Jaworska, “A Search-Engine Concept Based on Multi-Feature Vectors and Spatial Relationship,” in *Flexible Query Answering Systems*, vol. 7022, H. Christiansen, G. De Tré, A. Yazici, S. Zadrożny and H. L. Larsen, Eds., Ghent, Springer, 2011, pp. 137-148.

[8] Y. Dauphin, H. de Vries, J. Chung and Y. Bengio, “RMSProp and equilibrated adaptive learning rates for non-convex optimization,” *Neural Information Processing Systems*, 15 Feb. 2015.

[9] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.,” COURSERA:Neural Networks for Machine Learning, 2012.

[10] N. N. K. Win, T. Zin and H. M. Tun , “Comparison Of Power Quality Disturbances Classification Based On Neural Network,” *International Journal of Scientific & Technology Research*, vol. 4, no. 7, pp. 97-103, July 2015.

[11] D. Masters and C. Luschi, “Revisiting Small Batch Training for Deep Neural Networks,” ArXiv e-prints, Bristol, UK, 2018.

[12] M. Hagan, H. B. Demuth , M. H. Beale and . O. De Jesús , Neural Network Design, vol. 2nd edition, Boston, Massachusetts: PWS Publishing Co., 1996.

[13] M. F. MEILLER, “A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning,” *Neural Networks*, vol. 6, no. 4, pp. 525-533, 1993.

[14] F. D. Foresee and M. T. Hagan, “Gauss-Newton approximation to Bayesian learning,” in *Proceedings of the International Joint Conference on Neural Networks*, July, 1997.