BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 67, No. 2, 2019
DOI: 10.24425/bpas.2019.128485

Selected technical issues of deep neural networks for image classification purposes

M. GROCHOWSKI*, A. KWASIGROCH, and A. MIKOŁAJCZYK

Gdańsk University of Technology, Faculty of Electrical and Control Engineering, 11/12 Narutowicza St., 80-223 Gdańsk, Poland

Abstract. In recent years, deep learning and especially deep neural networks (DNN) have obtained amazing performance on a variety of problems, in particular in classification or pattern recognition. Among many kinds of DNNs, the convolutional neural networks (CNN) are most commonly used. However, due to their complexity, there are many problems related but not limited to optimizing network parameters, avoiding overfitting and ensuring good generalization abilities. Therefore, a number of methods have been proposed by the researchers to deal with these problems. In this paper, we present the results of applying different, recently developed methods to improve deep neural network training and operating. We decided to focus on the most popular CNN structures, namely on VGG based neural networks: VGG16, VGG11 and proposed by us VGG8. The tests were conducted on a real and very important problem of skin cancer detection. A publicly available dataset of skin lesions was used as a benchmark. We analyzed the influence of applying: dropout, batch normalization, model ensembling, and transfer learning. Moreover, the influence of the type of activation function was checked. In order to increase the objectivity of the results, each of the tested models was trained 6 times and their results were averaged. In addition, in order to mitigate the impact of the selection of learning, test and validation sets, k-fold validation was applied.

Key words: deep neural network, deep learning, image classification, batch normalization, transfer learning, dropout.

1. Introduction

Image analysis is a branch of engineering that has been developed by the research community for decades. The first generation of classification algorithms was based on two modules: a fixed feature extraction module and a classification module. The feature extraction module consists of a set of image analysis algorithms that usually work sequentially to extract significant features. The images are processed by a set of algorithms that includes simple ones, such as shape and edge detection, image filtering, morphological operations, and brightness or contrast adjustment, as well as more complex algorithms, such as HAAR feature extractors [1], Hough transforms [2], histogram of oriented gradients (HOG) [3], or SIFT descriptors [4]. The feature extraction module is designed by hand and does not require any training. The reason for employing such algorithms in image classification is that the images are highly dimensional data which classification algorithms could not process effectively. Therefore, there is a need for reducing the dimension of information provided to the classification algorithms. Moreover, the extracted features provide more useful information than raw image pixels. The extracted features feed the algorithm in the classification module, which typically consists of one certain classification algorithm. In early image analysis systems, the classification modules were built using such methods as fuzzy logic, rule-based system, or even simple if-else set of rules. The next generation of classification modules utilized trainable supervised learning algorithms, such as support vector machine (SVM) [5] or neural network (NN). Although the supervised algorithms enable to achieve better results, they need data to train. Consequently, the data acquisition process is needed.

The above-described approaches involve engineering skills and knowledge in the domain of the solved task. For example, to create a skin lesion classification system, we should have knowledge not only on image processing but also on medical knowledge on traits of benign and malignant lesions [6]. Despite difficulties in feature extraction module preparation, these algorithms are still used today and have a wide range of applications [7, 8]. This family of methods involves less data for system training and is not computationally expensive. Thus, they can work on mobile devices, for instance, the face detection algorithm in a digital camera.

In recent years, more attention was focused on automatic feature extraction, partially because of difficulties in the preparation of feature extraction algorithms and, on the other hand, thanks to the development of algorithms and the increase in computing power, in particular, that of graphics cards. In this area, the leading role is played by deep neural networks (DNN), especially by the convolutional neural network (CNN).

Deep learning is a wide family of machine learning algorithms that evolved from classical fully connected neural networks. Although a lot of works on deep learning have been done since the early eighties [9–11], for decades this family of techniques did not attract broad attention, mainly because of significant data requirements and high computational burden of algorithms. In fact, the potential of deep learning algorithms

*e-mail: michal.grochowski@pg.edu.pl

Manuscript submitted 2018-05-22, revised 2018-10-11, initially accepted for publication 2018-10-24, published in April 2019.

could be only realized in recent years, due to computationally effective hardware [12] and the wide availability of large datasets. Moreover, the appearance of libraries has facilitated the effective employment of deep learning models without direct low-level GPU programming [13–16]. The current advent of deep learning research began in 2012 when the deep learning based system has won the ImageNet large scale visual recognition challenge (ILSVRC) [17]. Deep neural networks significantly outperformed the systems based on classical image classification methods that involved the preparation of features and the use of relatively simply classifiers, such as SVM or shallow networks [18]. Although the Imagenet challenge is an image processing problem, the great success of the deep neural network in that challenge triggered the research not only in the area of image processing but at many more fields, such as natural language processing (e.g. language translation [19], language recognizing [20], speech-to-text [21] and text-to-speech conversions [22]), synthetic image generation [23], deep reinforcement learning [24], and many more.

Convolutional neural networks belonging to the deep learning family consist of many layers of neurons with different activation functions, structured in a special way (see Fig. 1). The word "deep" refers to a large number of layers composing the network. Inside the CNN, two modules can be distinguished, namely: feature extractor and classifier. Both the feature extractor and the classifier are trainable modules.

Training of the feature extraction module is carried out in such a way as to enable it to operate even on raw images, by appropriate adjustment of extractor weights, being the convolutional filters.

Present deep learning algorithms can reach human-level or almost human-level performance in many tasks. Deep neural networks have become the preferred approach in many computer vision applications. Despite this great success, deep learning applications still suffer from many drawbacks.

Deep neural networks need a lot of data to be trained properly. Moreover, network training is computationally expensive. Furthermore, the networks are described by many hyperparameters that should be tuned properly to achieve high performance. Thus, the utilization of deep learning involves many trials with different values of hyperparameters. These drawbacks are the motivation for further research in the field of deep learning.

A number of methods were proposed to enhance the classification abilities of deep learning algorithms. In this paper, the most popular methods were evaluated on the VGG based neural networks [25]. During the research, the influence on the classification performance of different neural structures, activation functions and methods such as batch normalization [26], dropout [27] and transfer learning [28] was examined. The dropout method increases the generalization abilities of the neural network by preventing the co-adaptation of neurons (the situation when a group of neurons acts in the same way). The batch normalization method comes from the idea of normalizing the input of a classifier. It is well known that normalization enhances the optimization process. Having this in mind, the batch normalization method normalizes not only the first layer input but also the subsequent layers. This technique allows us to use higher learning rates, moreover, it has been empirically shown that batch normalization prevents against neural network overfitting. Another important issue which was tested was the type of activation function used.

The influence of the most often used activation functions: the rectified linear Unit (ReLU) function and its modification, i.e. the leaky rectified linear unit (LReLU) function [29], on neural network performance were checked. The other examined issue was the scale in which transfer learning and model ensembling methods affect the results of classifiers. The results of the research are presented in a form of tables and figures, which are thoroughly discussed and summarized.

The aforementioned algorithms are usually tested on benchmark datasets that contain many well-balanced instances. In this paper, the architectures and methods are tested on a practical problem – skin lesion classification. This problem has properties that many researchers working on similar tasks have to deal with, for instance: low quantity of photos, unbalanced number of images in different classes, different size and quality of images, unclearly specified differences between classes, etc [30].

The remainder of this paper is organized as follows: Section 2 describes the methods used in the research, then Sec-

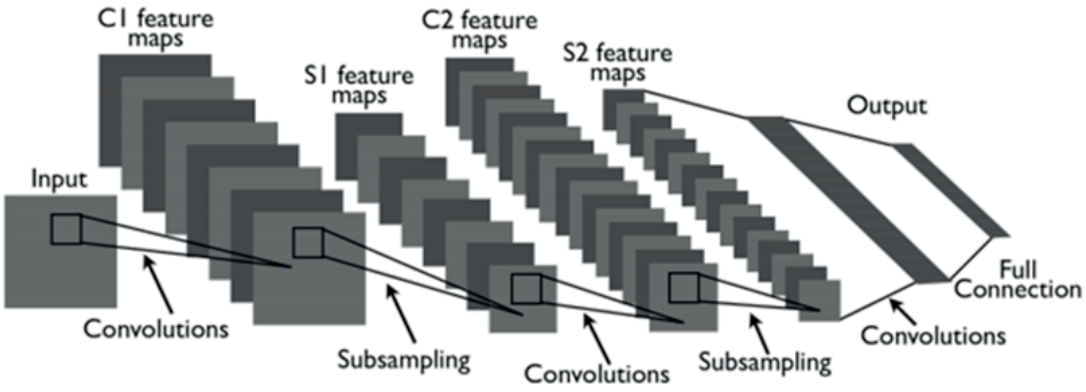


Fig. 1. Basic convolutional neural network [31]

tion 3 outlines the details of the problem of skin lesion analysis and dataset used. The utilized architectures and details of the training are described in Section 4. The achieved results are discussed in Section 5 and concluded in Section 6.

2. Methods

The most popular representative of the deep learning family is the convolutional neural network (Fig. 1). The basic convolutional neural network is built from convolutional layers that perform convolution operations on input data. The convolutional layers can be interpreted as a set of neurons, and convolutional operations as the image filtering process. However, there are two differences between the classical neural layers and convolutional layers. Firstly, each neuron inside the convolutional layer is connected to only a small fraction of adjacent neurons in the previous layer. This way the single neuron analyzes a small section of the image and detects particular features. Secondly, the neurons in the same group (filter) share their weights, as a result of which neurons in different positions detect the same feature. These two traits of convolutional layers are especially relevant for image processing applications. Due to these attributes, the image detector becomes invariant to object translation in the image. Moreover, these attributes of convolutional layers drastically reduce the number of neural network weights, thus reducing the training time. Such an approach enables parallel processing when using multiprocessor devices such as graphics cards.

Each convolutional layer is followed by a nonlinear transformation (activation function). The most popular one is the rectified linear unit (ReLU) described by the formula: $y = \max(0, x)$, where y stands for output and x for input. An additional layer, bearing the name of subsampling or pooling layer, is placed between some convolutional layers. This layer reduces the sizes of the images processed by successive network layers. The most common operations taking place in subsampling layers are MaxPooling and AveragePooling. Those operations aggregate the group of adjacent pixels by replacing them with pixels with maximal value (MaxPooling) or pixels being the mean of the analyzed group of pixels (AveragePooling). A typical convolutional neural network consists of many convolutional layers and pooling layers between them. The top of the deep neural network consists of a classical fully connected neural network that is fed by the output of the last convolutional layer. This top network is terminated by the sigmoid neuron or softmax layer, depending on the number of classes in the task.

As previously mentioned, the convolutional neural networks have an ability to learn how to extract relevant features. The feature extraction is performed by the convolutional layers, whose parameters are adjusted during the training. The extracted features are provided to the fully connected neural network classifier that classifies the input images. The feature extractors have an ability to process the raw data, therefore preliminary image processing is not required. Practically, only simple operations are needed, such as resizing (to fit the image size to the input size), or normalization (to improve the neural network training). The feature extraction is performed by a stack of convolutional layers, which performs the convolution operation on the output of the previous layer. Each layer transforms the representation of layer outputs to the higher abstract level. For instance, the first convolutional layer detects simple features, such as colors, edges, corners, whereas the last layers can detect more complex features, such as parts of the object.

Combining convolutional layers with nonlinear layers and pooling operations gives a complex and efficient classifier. Currently used architectures utilize even hundreds of layers, at the same time making use of additional methods enhancing the learning process and accuracy. Nevertheless, the network presented in Fig. 1 is a backbone of many modern architectures.

The rise of deep learning research caused the development of dedicated methods for deep neural networks, the most popular of which are: dropout, batch normalization, and model ensembling. Each method improves neural network training and performance in a different way, mainly by changing the structure and/or method of training. Many papers report the useful impact of these methods on the improvement of generalization abilities of deep neural networks, verified on popular benchmark tasks.

2.1. Dropout. Dropout is a technique that improves the performance of neural networks in a wide variety of application domains, including object classification, natural language processing, analysis of scientific data [27].

A method to improve generalization abilities of a deep neural network consists in combining the outputs of many trained networks (model ensembling). However, this method is computationally expensive. Dropout overcomes this problem by providing a way of approximately combining many neural networks into one model. The method involves temporarily removing some of the neurons during the training. The group of neurons to be removed is chosen randomly. The probability of retaining a neuron is a tunable hyperparameter called the dropout rate. The neurons are drawn and removed each time when a batch of samples is provided to the network. In practice, the neuron is removed by multiplying its output by zero. The dropout operation can be interpreted as a sampling of many thinned networks from the neural network. The total number of possible sampled thinned networks from the original network equals 2^n, where n is the number of neurons in the original network. Training of such a network can be seen as training of a collection of thinned networks with extensive weight sharing. After the training, the parameters of the neural network are multiplied by the value of the dropout rate.

The neural network that uses the dropout mechanism can be trained using the backpropagation and stochastic gradient descent algorithms without major modifications. In this case, partial derivatives of connections of the dropped neuron are set to zero. The classical methods, such as momentum or L2 regularization, work well with the dropout mechanism and improve network training. The dropout method improves the quality of detected features by preventing their co-adaptation. This is enforced by the attributes of the dropout method – each neuron

has to learn to cooperate with a randomly chosen sample of other units. Therefore, the trained neurons become more independent and create useful features on their own, without relying on other neurons.

Although dropout provides decent generalization results, the training of the neural network with dropout mechanism lasts 2–3 times longer than without it. The reason for longer training is that adjusting parameters is a very noisy process, due to different random architectures that are drawn each time when the batch is provided to the network [27].

The authors of [27] reported that the neural network with the dropout mechanism provided superior results, compared to the network without the dropout mechanism. The mechanism was tested on diverse classification tasks, including speech recognition, natural language processing, and image analysis. The experiments showed the effectiveness of the dropout method. It allowed decreasing the test error in the MNIST classification task from 1.60% (neural network without dropout) to 1.35% (with dropout). Both networks were equipped with logistic activation functions. The use of the ReLU activation function and the constraint on the maximum norm of weight vector has led to further improvement in accuracy, i.e. further error decrease to 0.95%. The dropout method was tested on CIFAR-10 and CIFAR-100 benchmarks, showing great test error decrease. For CIFAR-10, the dropout method decreased the test error from 14.98% (previous state of the art) to 14.32% by adding dropout to fully connected layers. Applying the dropout method to all layers, including convolutional layers, decreased the test error to 12.61%. Huge improvement in CIFAR-100 classification task was also reported, where applying dropout to the neural network decreased the test error by 6.28 percentage points. Unfortunately, the dropout mechanism does not work well on all types of architectures. For instance, applying dropout to ResNet identity blocks resulted in the failure of training [32]. On the other hand, the positive influence of dropout on ResNet was reported in [33] when inserted between the convolutional layers.

An ability to prevent overfitting, along with the effectiveness and simplicity of the method, were proved in a wide variety of application domains. The dropout method was used in numerous states of art classification models in the Imagenet classification task [18, 25].

2.2. Batch normalization. The batch normalization method solves the problem of changing the distribution of inputs of hidden layers. This problem is of especially high importance in very deep networks with many layers. Changing the distribution in one layer causes further changes in subsequent layers. This phenomenon slows down the training of such a network and makes it more difficult. In order to reduce the effect of this phenomenon, the networks should be trained with low learning rates. Moreover, proper weight initialization is required to obtain satisfactory results. To address this issue, the authors of [26] proposed the batch normalization method. This method is based on the well-known fact that whitening of classifier inputs improves the optimization process during the training. It will also be profitable to normalize the inputs of all hidden layers in the network. This method could accelerate the training process of the neural network. However, simple normalization of inputs to every layer does not work in practice, due to the canceling effect of backpropagation and normalization terms. Hence, the normalization should be made visible to the algorithm, which means that the gradient of loss function needs to backpropagate through normalization terms. Moreover, the method introduces two extra learnable parameters per neuron, which scale and shift the normalized neuron input values. This mechanism allows recovering the input values from before the normalization if it is beneficial for the training process. In the case of convolutional layers, the pair of parameters is assigned per one feature map.

The employment of batch normalization caused an increase in classification accuracy on the Imagenet benchmark dataset. For instance, adding batch normalization layers to the Inception network caused an increase in classification accuracy of the test set from 72.2% to 74.8% [26]. Moreover, the network required 10 times less training steps than the network without batch normalization. The method also works well on Google LeNet architecture, where the employment of batch normalization decreased the error from 29.0% to 26.8% in the Imagenet challenge [34]. The batch normalization method was successfully applied to many modern network structures, including ResNet [35], DenseNet [36], and Wide ResNet [33].

Although it would seem reasonable to combine dropout and batch normalization techniques, this combination usually leads to conflict in the network structure. The conflict is visible in the deep architecture test stage. Batch normalization operates on an average and a variance of the train set statistics, while dropout introduces a shift in the variance of layer inputs during the test stage by multiplying the weights by the dropout rate value. Hence, batch normalization fails to make use of its potential because of changed distribution of layer inputs.

2.3. Activation functions. Rectified linear unit is the most often used activation function in the deep learning area. Glorot [37] found that the rectified function performs much better than traditional activation functions. The neural networks equipped with ReLU functions achieved better classification accuracies than the networks equipped with a hyperbolic tangent function (tanh). Moreover, the authors of [18] stated that the usage of ReLU accelerates the training of the neural network by four times, compared with the neural network employing the tanh function. This is caused by the simplicity of ReLU and special attributes of this function. The function is not saturating, moreover, it has a piece-wise constant gradient. That fact allows preventing the well-known problem of vanishing gradient. The employment of ReLU function accelerates the convergence of the optimization process when training a network with many layers and neurons. ReLU enables faster training, compared to its saturating counterparts (tanh or sigmoid function). Additionally, the ReLU function is less sensitive to improper weight initialization, whilst in the case of traditional saturating nonlinearities, it is possible to initialize the weights in such a way as to have most of the functions saturated.

However, the ReLU function has still a chance for improper initialization, what might lead to the so-called dead neuron problem. The neuron is “dead” when its weight makes it stay

not activated by any training example, it is not possible to further learn the weights because of zero gradient. Such a neuron will never fire and will never have an ability to adjust their parameters.

To overcome this problem, the ReLU function was slightly modified, hence the Leaky Rectified Linear Unit was firstly introduced in [29]. The authors showed that employing the Leaky ReLU instead of ReLU improved the performance of acoustic models. The Leaky ReLU is a similar function to ReLU, but instead of taking zero value for arguments below zero, it replaces this part of the function with a linear function with a small slope, which is chosen before the training. Therefore, using leaky ReLU requires tuning an extra hyperparameter. The small slope in the leaky ReLU function enables the gradient to flow when the input is below zero, thus preventing the dead neuron problem.

Unlike the standard ReLU function, the leaky ReLU activation function has a non-zero gradient over its entire domain. This attribute allows the gradient to flow slowly when the neuron is not active. Moreover, the features of the function prevent the appearance of the dead neuron problem occurring in the ReLU function. In [38], the authors have empirically shown that the Leaky ReLU function achieved better test accuracy on benchmark datasets than its standard counterpart (ReLU). For CIFAR-10, the use of leaky ReLU reduced the test error from 12.45% to 11.20%, while for CIFAR-100 from 42.9% to 40.42%. Moreover, the authors also showed that the leaky ReLU performs better on the real-world problem of classifying plankton. In [39], the use of leaky ReLU improved the performance (measured by the F-score) from 0.78 points (ReLU) to 0.85 points on the real-world problem – lung cancer classification. The slope of the function for negative arguments in leaky ReLU was reported to have a strong influence on the performance. For instance, changing the slope from 0.01 to 0.3 caused an increase in the performance (F-score) from 0.81 to 0.85 points. This finding suggests that studying activation functions is still a crucial research field.

2.4. Transfer learning. Transfer learning is a technique that highly improves neural network performance [28]. The method is especially useful for small datasets. Training the deep neural network on a small dataset from a scratch could be difficult, as too many parameters are to be found in relation to the available training data. Transfer learning involves training the network on a big dataset, Imagenet for example, that contains 1M images. Then the weights of the network trained in the above way might serve as a good starting point in training it on another classification task. It is possible because many visual objects share similar low-level features like edges, shapes or colors. Therefore, convolutional layers could be trained once on a given task and then trained again on the target dataset (so-called fine-tuning). An intuitive explanation of the transfer learning effect comes from the real world. For instance, a person who can play the guitar will learn to play the piano more easily than the person who does not play any instrument, because both activities involve knowledge and specific skills such as note reading, knowledge of music theory, etc. It was reported in the literature [18, 40–43] that utilization of transfer learning enables to achieve much better results than the networks trained from scratch. Moreover, transfer learning shortens the neural network training time. However, there is little possibility of modification of the pretrained network, for example, it is impossible to change the activation function because different activation functions work with different sets of weights.

Transfer learning was successfully used especially in vision applications. For instance, the authors of [18] reported that pretraining on a huge dataset with further target task training caused the reduction of validation error from 18.2% to 16.6%. In such applications as mammographic tumor classification [40], pulmonary module detection [41], chest pathology detection [42], and image segmentation [43], employing the transfer learning method enhanced the classification accuracy.

2.5. Model ensembling. The model ensembling method involves training many classifiers, then combining classifier outputs to produce better classification result than a single classifier. The performance of models working together is better than that of a single model, because different neural models make mistakes for different testing inputs, despite similar levels of training error. Therefore, utilizing a number of different neural models may decrease the likelihood of a mistake. There is a variety of approaches to model ensembling, e.g. outputs of the models may be simply averaged or may stand as the inputs for another final classifier, such as logistic regression, neural network, or SVM. In practice, the application of the model ensembling method is computationally expensive, because it requires training of many classifiers. Therefore, this is an active field of research on how to make use of advantages of model ensembling and simultaneously keep the training time low [44, 45].

The model ensembling method is widely used in modern deep learning applications. Almost all high-performance Imagenet architectures employ the method to improve the performance of classification models. The first Imagenet challenge winner group used an ensemble of 6 networks that reduced the validation error from 18.2% to 16.4% [18]. An ensemble of two VGG networks decreased the top-5 test error from 7.0% to 6.8% [25]. An ensemble of six Google LeNet networks caused even bigger improvement in the top-5 test error – from 7.9% to 6.7% [46].

2.6. k-fold validation. In many real-world applications, huge datasets are not available. Hence, it is hard to split the data into a sufficiently relevant training set and a reliable testing one. It is obvious that small number of images in the test set can lead to statistical uncertainty around the averaged test error [47]. For example, if the test set contains only 50 images, one accurately classified image yields the accuracy increase by 2 percentage points. As a result, in a small dataset, statistical measures are heavily dependent on chosen instances in the training and test sets. To overcome this problem, the k-fold cross validation method is employed, which significantly substantiates the results, especially in the case of small dataset applications. The method allows using even all examples from a given dataset to estimate the performance of the classifier more accurately. The method involves splitting the dataset into k non-overlapping

subsets. Then, the classifier is trained k times, each time another subset becoming the test set, with the rest of the data used as the training set. The performance is computed by taking an average of accuracies (and another statistical measures) of k trials. The problem is that the method requires classifier training on each subset, which increases the time of training by k times [48].

3. The problem of skin lesion analysis

In order to practically test the influence of the described parameters and methods on the efficiency and effectiveness of a deep neural network-based classifier, a decision was made to tackle a very important real live problem of skin cancer diagnosis.

This task involves making a distinction between benign and malignant instances of skin lesions. Screening the skin lesion is a relevant task, due to the high death rate in people being affected by the disease. Early detection is a crucial aspect of curation because it increases the chance of full patient recovery [49]. Classification of lesions is a very challenging task because of similarity between malignant and benign melanomas, but also due to a large diversity of images of various quality, the existence of hair, different markers, and other obstacles hindering proper feature extraction.

The classical method of skin cancer detection involves examination of a lesion by the skilled specialist. The examination can be conducted by an unaided eye or using a dermoscope that allows high-quality lesion observation in fixed lighting conditions. The decision of whether the lesion is benign or malignant is made based on dermoscopic methods, such as the ABCD method, 7 point checklist, or Menzies method [50–52]. The application of these methods by the physician involves inspection of specific features. For example, to examine the lesion based on ABCD criterions, the physician should check the following traits: asymmetry (A), border (B), color (C) and differential structures (D) of the lesion, and, as a result, assign points which make the basis for final evaluation of the lesion.

The research on automated, computer-aided skin lesion classification is an active field. The classification of the skin lesion is a non-trivial task, due to difficulties caused by the characteristic of the task. The main problem is that the rules how to classify skin lesion are not precisely defined. That means that the border between benign and malignant lesions can be fuzzy and can lead to a different diagnosis given by different physicians. Such an inconsistency in diagnosis makes the problem of automatic classification much more difficult. Moreover, due to the privacy of patients and the characteristic of medical data, the datasets are more difficult to prepare. However, the growth in availability of the well-structured and labeled skin lesion datasets has been observed in recent years [53–56].

Classical methods of automatic lesion classification involve tedious and careful preparation of hand-crafted features which are then provided to the simple classifier, mostly SVM or shallow neural network, as was described in Section 1. The other type of methods, especially in medical applications, involves extraction of features related to the method for manual detection of the skin lesion. In such a solution, the algorithm

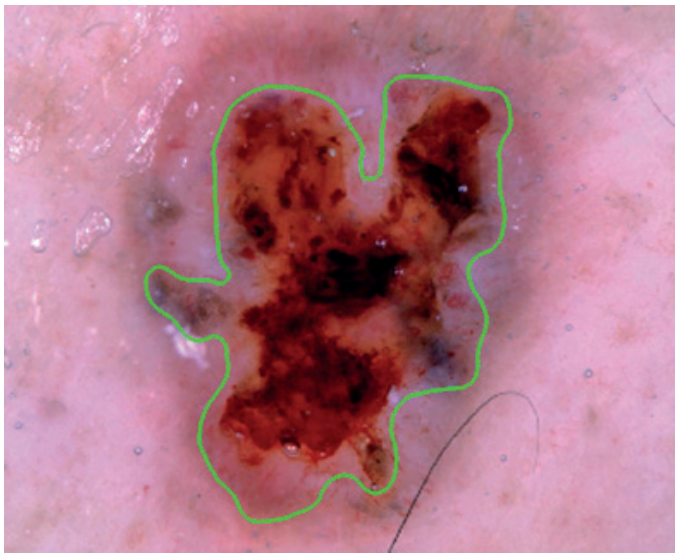


Fig. 2. An image with highlighted border feature [57]

detects the features described in the method used by the medical specialist. For instance, the authors of [57] based their system on the ABCD method. Fig. 2 shows an example of border feature detection according to the ABCD rule [57]. The methods have the higher trust of the medical society because the algorithm automates the work that is done manually by them.

Unlike the classical methods, the deep learning-based methods do not require preliminary preparation of features. The work is done automatically during the neural network training. Decent results of skin lesion classification with deep neural networks have been reported [46].

The dataset used in this research was provided by the International Society for Digital Imaging of the Skin [50]. The dataset is publicly available and contains about 13 000 high-quality, labeled dermatoscopic images of skin lesion and additional masks that indicate the position of the lesion in the image. The images sizes varies from 900×900 px up to even 3000×4000 px. The dataset is highly unbalanced, it consists of about 12 500 benign instances and only 1100 malignant instances. This disproportion between the classes makes proper training of classification systems more difficult. Selected examples of lesions from the database are shown in Fig. 3. In

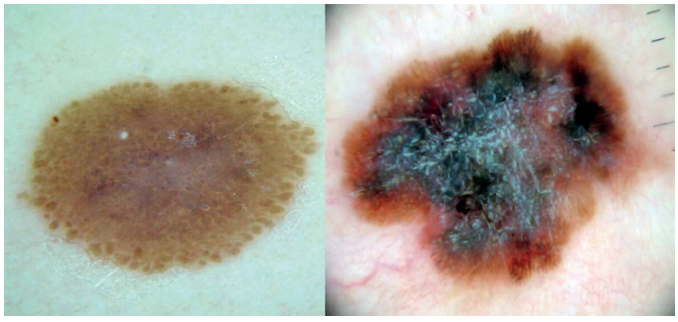


Fig. 3. The examples of the lesion from the dataset. Benign lesion (left), malignant lesion (right)

many cases, making a distinction between benign and malignant lesions is difficult for both humans and algorithms.

4. Architectures used and training

This section presents the details of the experiments performed to investigate the influence of the aforementioned methods on the classification accuracy. A decision was made to base the tested architectures on a popular VGG network family, which has had a great impact on the research in the field of deep learning. The VGG networks served as a backbone for further modifications and improvements of deep architectures. In the present research, use was made of popular VGG11 and VGG16 networks, a very detailed description of which can be found in [25]. In order to check the influence of the network size on performance and on generalization abilities, we proposed the VGG8 architecture with fewer parameters than the VGG11 and VGG16 networks (see Table 4 for the details). The structure of these networks is presented in Table 1 and Table 2. A set of modifications was introduced with different combinations of methods described in Section II. In the remainder of this section, the details about the training process and data preparation are described.

Table 1
Basic architectures

VGG8	VGG11	VGG16
conv-64	conv-64	conv-64 conv-64
maxpool		
conv-128	conv-128	conv-128 conv-128
maxpool		
conv-256	conv-256 conv-256	conv-256 conv-256 conv-256
maxpool		
conv-512	conv-512 conv-512	conv-512 conv-512 conv-512
maxpool		
conv-512	conv-512 conv-512	conv-512 conv-512 conv-512
maxpool		
FC-1024 FC-1024		
Sigmoid neuron		

4.1. Architectures. Different architectures based on the VGG family were employed and tested. The elements of the structures of the VGG neural networks analyzed during the research are presented in Table 1.

The size of the input of each network is $224 \times 224 \times 3$, which corresponds to the image size of 224×224 pixels, and the last dimension is the color channel (RGB).

The values alongside the convolutional layers refer to a number of filters included in these layers, for example, conv-64 is the convolutional layer consisting of 64 convolutional filters. Each convolutional filter has a $3 \times 3 \times n$ kernel, where n stands for a number of filters in the previous layer. The assumed filter stride (the number of pixels with which the filter slides over the image) equals 1. In order to preserve the same output and input size of the layer, zero padding around the feature maps was employed. The FC layer refers to a fully-connected layer (classical neural network) and the values indicate the numbers of neurons, for example, FC-1024 is a fully connected layer with 1024 neurons. There is also a sigmoid neuron on the top of the network.

A number of experiments have been conducted with different configurations of parameters and methods described in Chapter II. Details of the analyzed networks, along with the accompanying methods, are presented in Table 2. The dropout method was applied after two fully-connected layers (networks: C, F, G) with the dropout rate set to 0.5. The batch normalization mechanism was applied before each activation function (networks D, E). The influence of activation function was also tested. In those tests, the Leaky ReLU function was applied with the slope equal to 0.2. The influence of transfer learning was tested on the Network G, which was previously pre-trained on the Imagenet dataset consisting of 1M images divided into 1000 classes (1000 images per class). During network G training, all weights were adjusted. The publicly available model of pre-trained VGG16 network from Keras library was utilized [58].

4.2. Optimization algorithm. In order to train the networks, the mini-batch gradient descent with Nesterov momentum algorithm was utilized to minimize the loss function. The

Table 2
Tested architectures

	Network A	Network B	Network C	Network D	Network E	Network F	Network G
Basic network	VGG8	VGG8	VGG8	VGG8	VGG11	VGG16	VGG16
Activation function	Leaky ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
Transfer learning							**X**
Dropout			**X**		**X**	**X**	**X**
Batch normalization	**X**			**X**			

momentum coefficient was set to 0.9, which is a frequently chosen value in many applications. The binary cross-entropy was chosen as the loss function for the two-class classification problem. The initial learning rate was set equal to 0.01 for networks B, C, D, and E, while the training of networks A, F, and G was started with the learning rate set to 0.001, as the optimization process was unable to converge at higher learning rates. During the training, the learning rate was divided by 3 each time when the validation loss did not decrease during 4 subsequent epochs. The early stopping method was applied to prevent overfitting: the training was to stop when no improvement was recorded in the value of the validation loss function during 8 subsequent epochs.

4.3. Data preparation. The size of the input images was chosen as (224×224×3), which is a standard size used in many image recognition applications. This size is a compromise between the image quality on the one hand, (higher-size images can contain more details) and the computational requirements on the other hand. The size of 224×224×3 pixels is also used in the pretrained network. The original size of the images in the dataset is much higher, compared to the size of the neural network input. It is noteworthy that the lesions in the images cover only a small fraction of image size. Therefore, the lesions were extracted from the image using masks provided by the organization and then resized to the size of 224×224×3. The masks consisting of less than 10 white pixels were qualified as incorrect and the corresponding images were kept unchanged. The lesions that covered less than 20 percent of the image were cut out from the image. First, using the mask, the minimum radius of the circle covering the lesion was found. Next, the bounding square of this circle was found in such a way that the edges of the square were parallel to image edges. In order to have some skin left around the lesion, the size of the square was extended by an extra 44%. Finally, the square with the lesion was cut out from the image of the lesion. The images with lesions that covered more than 20 percent were kept unchanged. Then, the images in the dataset were cropped and resized to fit the neural network input size. Finally, the dataset was normalized by subtracting the mean and setting the standard deviation set to one and zero, respectively.

For all experiments, a train-validation-test scheme was applied. In order, to obtain more substantial results, k-fold cross-validation with 5 folds was used. After these modifications, each fold contained 24 709 training images, which were divided into 12 333 benign lesions and 12 376 malignant lesions. The malignant lesions were generated by copying 13 times 884 malignant instances available in the dataset (upsampling). Although several the same images were in the malignant class, they would become different due to data augmentation applied online during the training. The validation set and the test set consisted of 200 images, equally divided into two classes. In each fold, the validation set and the test sets were unique, which means that the image used in one fold did not appear in other folds of the test or validation set.

The dataset was augmented by numerous modifications, such as rotation, width and height shift, horizontal and vertical flip, and zooming. The data augmentation was performed online, before providing the images to the neural network input.

4.4. Model ensembling. The influence of model ensembling on classification results of all tested networks was examined as well. Each network was trained 6 times to combine their outputs in model ensembling. Next, two types of model ensembling were applied. The first type involves averaging outputs of 6 trained networks, in the remainder of the paper this method is referred to as 'average ensembling'. The second type involves training a sigmoid neuron which takes outputs of 6 networks as an input. This method is referred to as 'logistic classifier ensembling'. The classifier was trained on outputs of the deep networks fed with 4000 images randomly chosen from the train set. The linear regression classifier contained 7 parameters (6 per each input and one for bias term). The classifier training was conducted using the stochastic gradient descent optimizer.

4.5. Classification threshold tuning. In order to increase the classification accuracy, classification threshold tuning was performed following a simple pipeline. 100 threshold values ranging from zero to one were iterated, and the classification accuracy was checked on the validation set. Next, the threshold with best classification accuracy on the validation set was selected. To tune the threshold, the validation set from all 5 folds was used, which resulted in the same threshold for all folds.

4.6. Software and hardware. The networks were trained using the Python Keras [58] library running on the top of the Theano [14] library. The Keras library allowed easy and fast prototyping of neural networks.

During the calculations, the Nvidia CUDA library was utilized that allowed parallel computing on GPU. The networks were trained on GPU, while the data augmentation operations were performed by CPU.

All tests were performed on a computing unit equipped with: GeForce GTX 980 Ti GPU with 6 GB memory, Intel Core i7–4930K processor, and 16 GB RAM memory.

5. Experimental results

A series of experiments were conducted with all previously described architectures. All results are summarized in Table 3. The ROC curves of evaluated systems are presented in Figs. 4–7. The influence of certain methods on averaged results of a single network (without model ensembling) is discussed.

5.1. Evaluation metrics. To evaluate the performance of the tested networks, the metrics of accuracy (ACC), specificity (SPC) (also known as True Negative Rate – TNR), and sensitivity (SST) (also known as True Positive Rate – TPR) were used. The accuracy was calculated as the ratio of properly classified instances to all instances in the dataset. The sensitivity, in that case, means the ratio of properly classified malignant in-

Table 3
Tested architectures

Network	Single network				Model ensembling by average				Model ensembling by neural network			
	ACC	AUC	SST	SPC	ACC	AUC	SST	SPC	ACC	AUC	SST	SPC
A	72.12	0.808	0.754	0.688	74.10	0.822	0.774	0.708	73.60	0.823	0.738	0.734
B	70.38	0.786	0.734	0.673	72.50	0.802	0.842	0.608	73.00	0.804	0.854	0.606
C	72.87	0.804	0.775	0.683	76.50	0.821	0.786	0.744	75.50	0.821	0.822	0.688
D	73.77	0.828	0.784	0.692	76.80	0.851	0.796	0.740	76.90	0.852	0.802	0.736
E	68.83	0.765	0.752	0.625	71.20	0.779	0.816	0.608	71.30	0.781	0.766	0.660
F	67.78	0.749	0.687	0.668	65.40	0.759	0.586	0.722	66.20	0.758	0.628	0.696
G	76.67	0.858	0.801	0.732	79.40	0.882	0.868	0.720	79.60	0.883	0.878	0.714

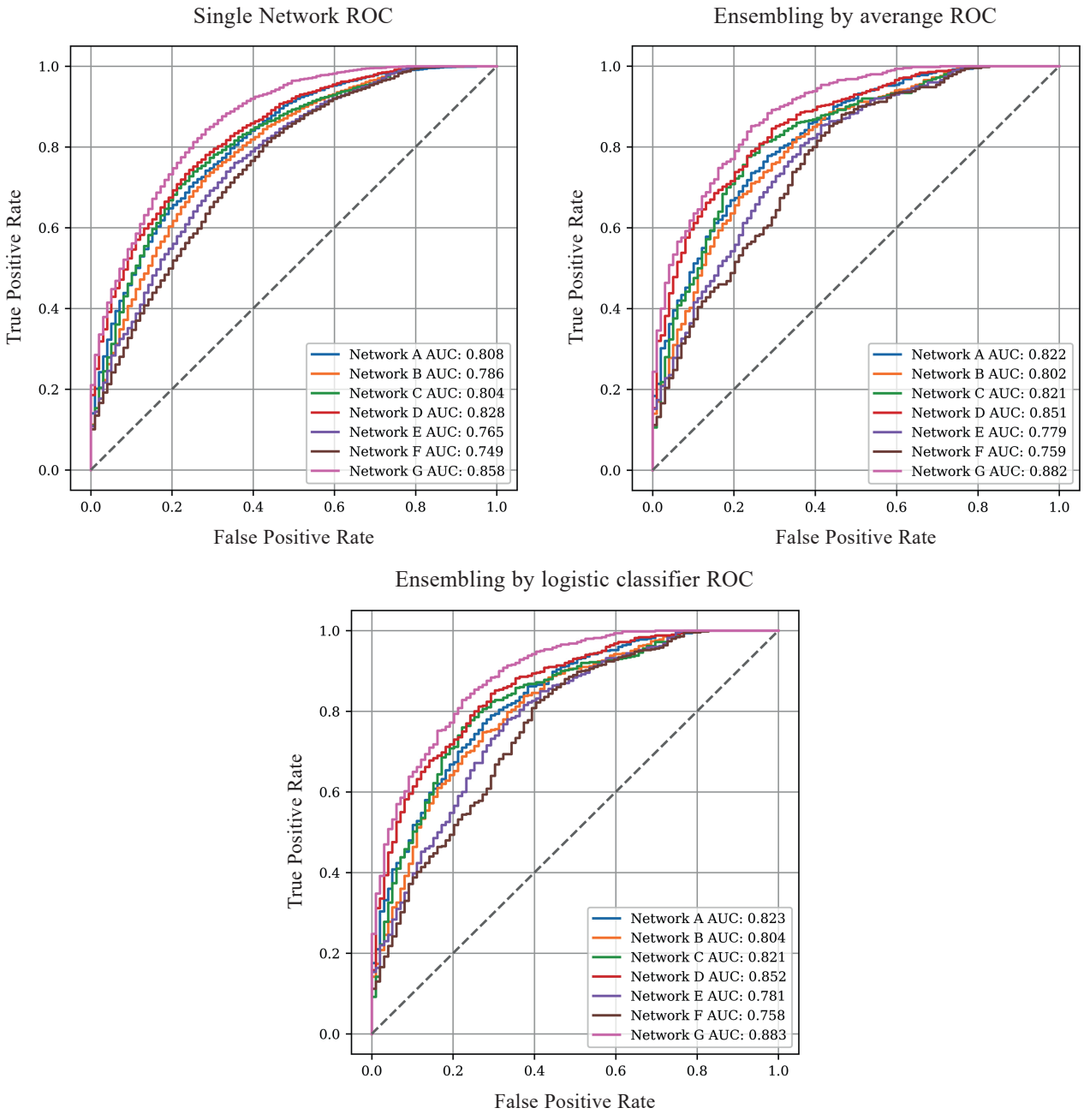
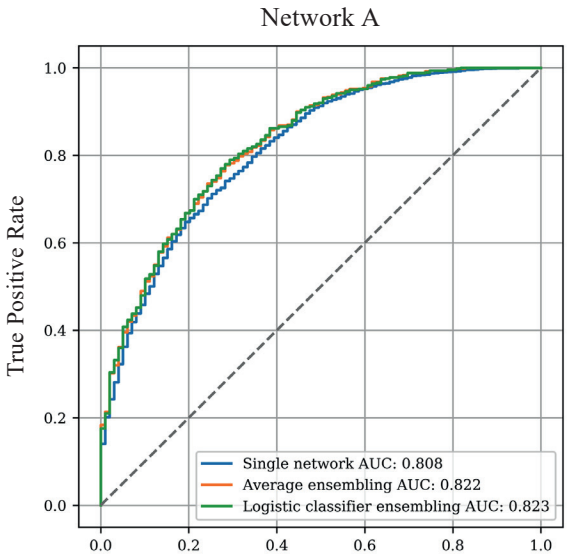
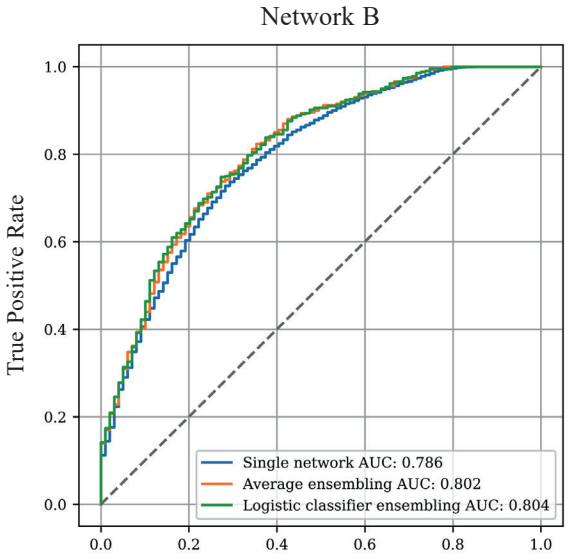


Fig. 4. Comparison of different architectures





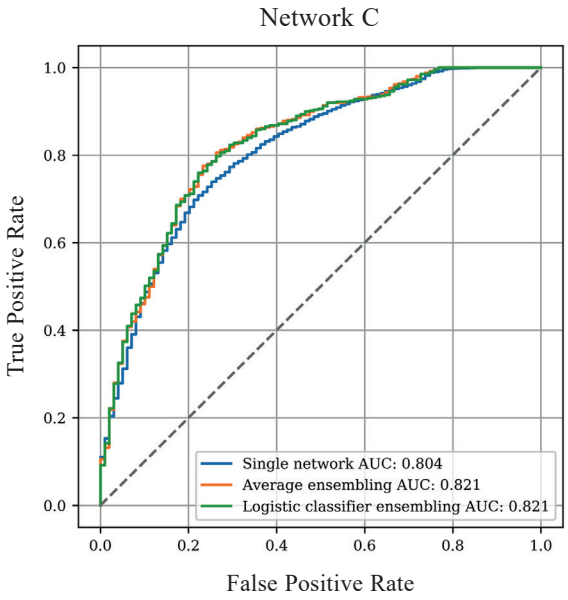
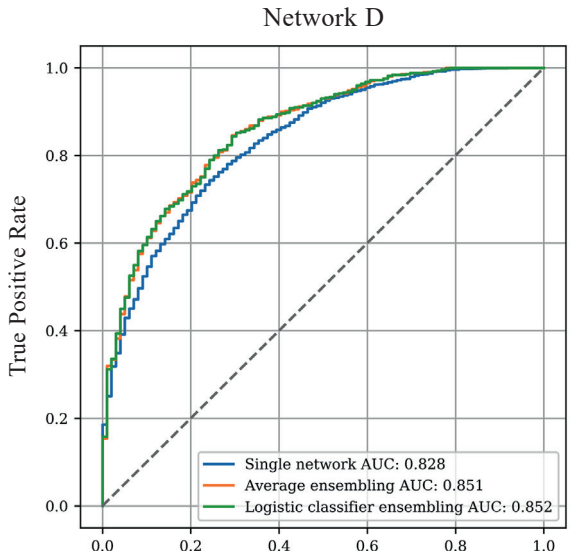
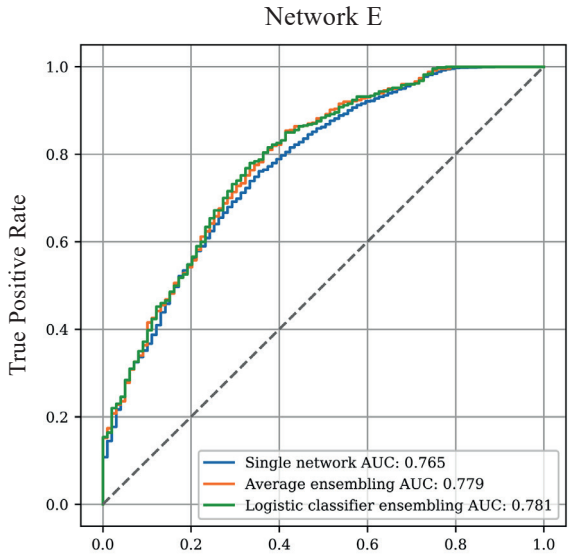


Fig. 5. Influence of model ensembling on different architectures – networks A–C





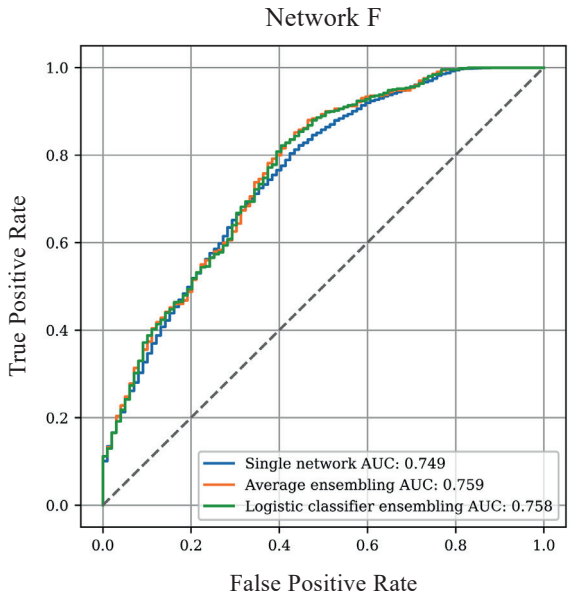


Fig. 6. Influence of model ensembling on different architectures – networks D-F

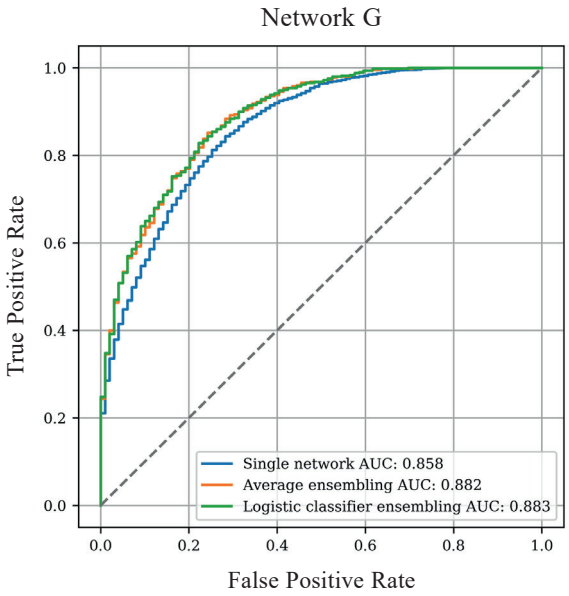


Fig. 7. Influence of model ensembling on different architectures – network G

stances to their total number (1), while the specificity is the ratio of properly classified benign instances to their total number (2):

$$\text{Sensitivity (SST)} = \text{TP/P} = \text{TP}/(\text{TP} + \text{FN}) \quad (1)$$

$$\text{Specificity (SPC)} = \text{TN/N} = \text{TN}/(\text{TN} + \text{FP}) \quad (2)$$

where: P is the number of positive samples (malignant), N is the number of negative samples (benign), TP-True Positive: is the number of malignant instances correctly identified as malignant; FP-False Positive: is the number of benign instances incorrectly identified as malignant; TN-True Negative: is the number of benign instances correctly identified as benign; and finally FN-False Negative: is the number of malignant instances incorrectly identified as benign. Moreover, the ROC (Receiver Operator Characteristic) curves were prepared and the area under that curves were calculated, which is known as the AUC acronym. The ROC curve is created by changing the classification threshold from 0 to 1 and measuring the True Positive Rate (Sensitivity) and the False Positive Rate (1-specificity).

5.2. Transfer learning influence. The influence of transfer learning on classification accuracies was investigated. For this purpose, two the same architectures (VGG16) but differently initialized were used. The first network (F) was initialized with randomly generated weights, while the second network (G) – with the weights transferred from the network trained on the Imagenet set. The results are presented in Table 3 and in Fig. 4. Transfer learning yielded the accuracy improvement of about 10.24 percentage points, compared to the network trained from scratch. Moreover, network G achieved the best result among all tested networks, while network F, with the same architecture, achieved the worst result among all tested architectures. It shows that if we are forced to train the network from the scratch it is better to use the network with relatively small amounts of layers.

5.3. Model ensembling influence. Table 3 compares the performance of all networks. It is noteworthy that model ensembling improves the performance of all evaluated architectures. The influence of model ensembling is depicted in the form of ROC curves in Figs. 5–7. The single network ACC stands for the averaged accuracy of six individual neural networks. Model ensembling by averaging stands for combining outputs of six individual networks by averaging their last sigmoid neuron outputs. Model ensembling by logistic classifier stands for the accuracy obtained by sigmoid neuron fed by outputs of six individual networks within the tested architecture. Both model ensembling methods improved the performance of the evaluated systems. The ensembling by logistic classifier in most cases is slightly better then one obtained for ensembling by averaging.

5.4. Comparing different activation functions. Two the same architectures but with different activation functions were trained: with the Leaky ReLU function (network A) and with the ReLU function (network D). Both networks were equipped with the batch normalization mechanism. Although superior results of Leaky ReLU have been reported in the literature, in our experiments this function performed worse (see Table 3 and Fig. 4). It shows that the used methods are not universal and cannot be successfully applied to an arbitrary problem. The performance of different architectures depends on attributes of the solved problem. Therefore, there is still room for further research in this field.

5.5. Depth influence. We examined how the number of layers influencing the classification accuracy. We tested three networks: C, E, and F containing 8, 11 and 16 layers, respectively. The neural network with the smallest number of layers performed best (see Table 3 and Fig. 4). Conducted experiments showed that an increase in the number of convolutional layers does not always lead to better performance. In the case of having a relatively small dataset, it is much better to start searching an architecture with a low number of layers, and eventually increase their size if needed.

5.6. Dropout vs batch normalization. Two methods were compared that have regularization effect: dropout and batch normalization. The neural network equipped with batch normalization significantly outperformed the network with dropout (see Table 3 and Fig. 4). The performance improvement was even higher in the ensembled networks. The results show that neural networks with batch normalization are more appropriate for model ensembling than those equipped with dropout. The reason for this could be that the dropout method is, in fact, an approximation of ensembling of many networks. Therefore, the influence of standard model ensembling on networks trained with dropout is smaller. Moreover, the architecture (not included in Table 3) with both dropout and batch normalization was evaluated. The network converged during the training, but

in the test stage, it failed to classify images properly, which is also confirmed by the results obtained by other researchers.

5.7. Network size, time of learning, number of epochs to converge. The details of the training are outlined in Table 4. The time of training and the number of epochs to converge were calculated as an average of the training process conducted on 5 folds. We can notice huge improvement caused by the implementation of transfer learning. It reduced by about twice the required number of epochs as well as the time to converge. A slight difference is observed between the results for networks equipped with ReLU and Leaky ReLU activation functions. The networks were evaluated in different ways: without any mechanism, with dropout, and with batch normalization.

Table 4
Models details

	Network size (MB)	Number of parameters	Number of epochs to converge	Time to converge (h)
Network A (VGG8)	240	30659587	31.6	3.6
Network B (VGG8)	240	30652545	17.8	1.5
Network C (VGG8)	240	30652545	24.6	2.1
Network D (VGG8)	240	30659587	29.8	3.5
Network E (VGG11)	281	35971843	20.4	3.75
Network F (VGG16)	320	41456449	30.0	5.2
Network G (VGG16)	320	41456449	13.4	2.4

The network without any mechanism converged fastest, but it achieved worst results among the three evaluated networks. Although it was reported in [26] that batch normalization accelerates the training, in the present research the training time of this network was the longest.

6. Discussion

In the previous section, we have presented comprehensive results of research on the influence of the methods described in the article on training and performance of the convolutional neural networks, based on the skin moles classification benchmark.

Transfer learning significantly influences the performance of the classifier. In the case of analyzed testing data sets, applying of the transfer learning before the training, increased the classification accuracy of about 10% compared with the networks trained from scratch. Moreover, it shortened the training time twice.

The model ensembling of the networks in case of each kind of network has improved the results. However, there is only a small difference between averaging the outputs of the network and applying an extra neural network at the top of the networks.

Unlike many other studies on the use of different activation functions, our studies have not confirmed the superiority of leaky ReLU function over the ReLU. Thus, it seems that this is not a universal dependence and one can expect different results for the different applications.

Regarding the dropout and batch normalization mechanisms, the batch normalized networks achieved superior performance, compared to the network with the dropout mechanism. As reported in other researches combination of batch normalization and dropout mechanism leads to the deterioration of classification performance.

The number of layers has a major impact on the classification performance. In described in the paper case, the network with the smallest number of layers performed best. The reason for that is a relatively small number of training samples comparing with parameters of the neural network to be found during the optimization. Therefore, in case of having a small dataset, it is advised to start from the architecture with a low number of layers and increase the numbers of layers if needed. Another way of having the larger set of data is applying the data augmentation methods. The examples of such approach can be found e.g [59–61].

In order to increase the classification accuracy, classification threshold tuning was performed. Such method let for increasing the classification comparing with a fixed threshold.

7. Conclusions

In this paper, the influence of recent advances in the area of deep learning such as transfer learning, dropout, batch normalization, threshold tuning, model ensembling, is examined for three different convolutional structures. The analyzed architectures were tested on the very important real-world problem of skin cancer classification, in which the algorithm has to make a distinction between benign and malignant lesion.

Seven representative neural networks were evaluated in different settings and methods applied. In order, to obtain more confident results, k-fold cross-validation with 5 folds was used. From the same reason, all the experiments were conducted 6 times and the results were averaged.

Certain differences in the conclusions from the conducted research, as compared to the results obtained by other researchers, may result from the specificity of the considered case study. This means that some of these conclusions are not universal.

The authors believe that the presented and described methods, together with the results demonstrating their practical influence on the results of classification achieved by networks, will help other researchers in their work.

Acknowledgments. This research was funded by Polish Ministry of Science and Higher Education in the years 2017–2021, under the Diamond Grant No. DI2016020746. The authors wish to express their thanks for the support.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 1, pp. I-511-I–518 vol. 1.

[2] P. Mukhopadhyay and B. B. Chaudhuri, "A survey of Hough Transform," *Pattern Recognition*, vol. 48, no. 3, pp. 993–1010, Mar. 2015.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 886–893 vol. 1.

[4] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] M. Grochowski, A. Mikołajczyk, and A. Kwasigroch, "Diagnosis of malignant melanoma by neural network ensemble-based system utilising hand-crafted skin lesion features," *Metrology and Measurement Systems*, vol. 26, no. 1, 2019.

[7] T. Markiewicz, M. Dziekiewicz, S. Osowski, M. Maruszynski, W. Kozlowski, *et al.*, "Thresholding techniques for segmentation of atherosclerotic plaque and lumen areas in vascular arteries," *Bull. Pol. Ac.: Tech.*, vol. 63, no. 1, pp. 269–280, 2015.

[8] A. Czajka, W. Kasprzak, and A. Wilkowski, "Verification of iris image authenticity using fragile watermarking," *Bull. Pol. Ac.: Tech.*, vol. 64, no. 4, pp. 807–819, 2016.

[9] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-13, no. 5, pp. 826–834, 1983.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[12] N.P. Jouppi, A. Borchers, R. Boyle, P.-L. Cantin, C. Chao, *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 1–12, 2017.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," in *OSDI*, 2016, vol. 16, pp. 265–283.

[14] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, *et al.*, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, vol. 472, p. 473, 2016.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, *et al.*, "Automatic differentiation in PyTorch," 2017.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[18] A. Krizhevsky, I. Sutskever, and H. Geoffrey E., "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pp. 1–9, 2012.

[19] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, *et al.*, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation," 2016.

[20] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, *et al.*, "Automatic language identification using deep neural networks," 2014, pp. 5337–5341.

[21] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, *et al.*, "Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin." pp. 173–182, 2016.

[22] S.Ö. Arık, G. Diamos, A. Gibiansky, J. Miller, K. Peng, *et al.*, "Deep Voice 2: Multi-Speaker Neural Text-to-Speech."

[23] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, *et al.*, "Generative Adversarial Networks," 2014.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, *et al.*, "Playing Atari with Deep Reinforcement Learning."

[25] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations (ICRL)*, pp. 1–14, 2015.

[26] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[28] S.J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[29] A.L. Maas, A.Y. Hannun, and A.Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," *Proceedings of the 30 th International Conference on Machine Learning*, vol. 28, p. 6, 2013.

[30] A. Kwasigroch, A. Mikołajczyk, and M. Grochowski, "Deep neural networks approach to skin lesions classification #x2014 – A comparative analysis," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2017, pp. 1069–1074.

[31] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 253–256.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks."

[33] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision."

[35] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimedia Tools and Applications*, pp. 1–17, 2017.

[36] G. Huang, Z. Liu, L. van der Maaten, and K.Q. Weinberger, "Densely Connected Convolutional Networks," Aug. 2016.

[37] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315–323, 2011.

[38] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[39] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, "Lung pattern classification for interstitial lung diseases using a deep convolutional neural network," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1207–1216, 2016.

[40] B.Q. Huynh, H. Li, and M.L. Giger, "Digital mammographic tumor classification using transfer learning from deep convolutional neural networks," *Journal of Medical Imaging*, vol. 3, no. 3, p. 034501, 2016.

[41] B. van Ginneken, A.A. Setio, C. Jacobs, and F. Ciompi, "Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans," in *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, 2015, pp. 286–289.

[42] Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen, *et al.*, "Chest pathology detection using deep learning with non-medical training," in *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, 2015, pp. 294–297.

[43] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[44] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, *et al.*, "Snapshot ensembles: Train 1, get M for free," *arXiv preprint arXiv: 1704.00109*, 2017.

[45] T. Garipov, P. Izmailov, D. Podoprikhin, D.P. Vetrov, and A.G. Wilson, "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs," *arXiv preprint arXiv:1802.10026*, 2018.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, *et al.*, "Going deeper with convolutions," 2015.

[47] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[48] C.M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2011.

[49] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[50] F. Nachbar, W. Stolz, T. Merkle, A.B. Cognetta, T. Vogt, *et al.*, "The ABCD rule of dermatoscopy: high prospective value in the diagnosis of doubtful melanocytic skin lesions," *Journal of the American Academy of Dermatology*, vol. 30, no. 4, pp. 551–559, 1994.

[51] R.H. Johr, "Dermoscopy: alternative melanocytic algorithms – the ABCD rule of dermatoscopy, menzies scoring method, and 7-point checklist," *Clinics in dermatology*, vol. 20, no. 3, pp. 240–247, 2002.

[52] J.S. Henning, S.W. Dusza, S.Q. Wang, A.A. Marghoob, H.S. Rabinovitz, *et al.*, "The CASH (color, architecture, symmetry, and homogeneity) algorithm for dermoscopy," *Journal of the American Academy of Dermatology*, vol. 56, no. 1, pp. 45–52, 2007.

[53] "ISIC Archive." [Online]. Available: https://isic-archive.com/. [Accessed: 10-Jan-2018].

[54] "Edinburgh Innovations: Dermofit Image Library," *Edinburgh Innovations, online licensing portal*. [Online]. Available: https://licensing.eri.ed.ac.uk/i/software/dermofit-image-library.html. [Accessed: 26-Apr-2018].

[55] G. Argenziano, H.P. Soyer, V. De Giorgi, D. Piccolo, P. Carli, *et al.*, "Dermoscopy: a tutorial," *EDRA, Medical Publishing & New Media*, vol. 16, 2002.

[56] T. Mendonça, P.M. Ferreira, J.S. Marques, A.R. Marcal, and J. Rozeira, "PH 2-A dermoscopic image database for research and benchmarking," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, 2013, pp. 5437–5440.

[57] A. Mikołajczyk, A. Kwasigroch, and M. Grochowski, "Intelligent system supporting diagnosis of malignant melanoma," in *Polish Control Conference*, 2017, pp. 828–837.

[58] F. Chollet and others, "Keras," 2015.

[59] A. Galdran, A. Alvarez-Gila, M.I. Meyer, C.L. Saratxaga, T. Araújo, *et al.*, "Data-Driven Color Augmentation Techniques for Deep Skin Image Analysis," *arXiv preprint arXiv:1703.03702*, 2017.

[60] M. Grochowski, M. Wąsowicz, A. Mikołajczyk, M. Ficek, M. Kulka, *et al.*, "Machine Learning System For Automated Blood Smear Analysis," *Metrology and Measurement Systems*, vol. 26, no. 1, 2019.

[61] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, 2018, pp. 117–122.