Date of publication MAR-30, 2022, date of current version DEC-14, 2021.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.21.1.2512

Deep Neural Network with Adaptive Parametric Rectified Linear Units and its Fast Learning

YEVGENIY BODYANSKIY[1], ANASTASIIA DEINEKO[2], VIKTORIA SKORIK[1], FILIP BRODETSKYI[3]

[1]Control Systems Research Laboratory, National University of Radio Electronics, Nauky av. 14, Kharkiv, 61166, Kharkiv, Ukraine,
(e-mail: yevgeniy.bodyanskiy@nure.ua, viktoriia.skorik@nure.ua)
[2]Dept. of Artificial Intelligence, National University of Radio Electronics, Nauky av. 14, Kharkiv, 61166, Kharkiv, Ukraine,
(e-mail: anastasiia.deineko@nure.ua)
[3] Dept. of Informatics, National University of Radio Electronics, Nauky av. 14, Kharkiv, 61166, Kharkiv, Ukraine,
(e-mail: filip.brodetskyi@nure.ua)

Corresponding author: Yevgeniy Bodyanskiy (e-mail: yevgeniy.bodyanskiy@nure.ua).

ABSTRACT The adaptive parametric rectified linear unit (AdPReLU) as an activation function of the deep neural network is proposed in the article. The main benefit of the proposed system is adjusted activation function whose parameters are tuning parallel with synaptic weights in online mode. The algorithm of the simultaneous learning of all neurons parameters with AdPReLU and the modified backpropagation procedure based on this algorithm is introduced. The approach under consideration permits to reduce volume of the training data set and increase tuning speed of the DNN with AdPReLU. The proposed approach could be applied in the deep convolutional neural networks (CNN) in conditions of the small value of training data sets and additional requirements for system performance. The main feature of DNN under consideration is possibility to tune not only synaptic weights but the parameters of activation function too. The effectiveness of this approach is proved by experimental modeling.

KEYWORDS deep neural network; convolutional neural network; adaptive parametric rectified unit; activation function.

I. INTRODUCTION

FOR today the artificial neural networks (ANN) are commonly used for solving different tasks arising in Data Science, Data Mining, Big Data and others, first of all, due to universal approximation properties and ability to learn (tuning its synaptic weights and maybe its architecture) during input data of arbitrary nature processing. Here, the most widely used are multilayer perceptrons whose universal approximation possibilities were proved in the frames of G. Cybenko's and K. Honrik's theorems [1, 2]. The nodes of these neural networks are elementary F. Rosenblatt's perceptrons with so-called squashing activation functions that include well known $\sigma -$functions (sigmoidal functions), tanh, Softsign, Satlin, arctan [3] and others.

Based on the multilayer perceptrons, so-called deep neural networks (DNN) [4-9] were designed, that proved their effectiveness in the tasks of image processing of different nature, natural language processing, time series analysis, including audio signals, etc. Despite their undoubted advantages, these networks are not devoid of some problems that arise, first of all, in the process of their training. In the case of large volume data sets processing deep neural networks suffer from the so-called vanishing-exploding gradient effect that is connected with stopped learning process in DNN. The vanishing-exploding gradient effect first of all is connected with the shape of the squashing functions that had led to the abandonment of their using.

In such situations the most widespread activation functions are ones from so-called rectified unit family [10] which includes rectified linear units (ReLU), parametric rectified linear units (PReLU), exponential linear units (ELU) and other [5-13]. This is, as usual, piecewise functions with fixed parameters that traditionally are selected from empirical reasons. Their main advantage is that their derivatives are constants, which simplifies the learning process and permits to avoid the effect of vanishing-exploding gradient. Their main disadvantage is that they do not satisfy the requirements of the basic approximation theorems [1, 2] and to achieve the required accuracy of the piecewise linear approximation, DNN must contain in architecture a large number of tuning parameters – synaptic weights. This, in turn, increases the training time and the required size of training data set.

In this regard the idea of learning-adaptation of piecewise activation function was proposed. So, in [14] the "maxout" activation function was introduced, in [15] – adaptive piecewise linear units (APL), in [16] – S-shaped rectified linear unit (SReLU), in [17] – adaptive blending units (ABU). Parameters of these functions are tuned using the gradient procedures (stochastic and regularized versions) with constant learning rate independently of synaptic weights tuning by error backpropagation. This approach can improve the approximation properties of DNN, however, it does not lead to speed increasing of the learning process. It is possible to increase the speed of the learning process by simultaneous adjusting the synaptic weights and parameters of activation functions within a united tuning procedure optimized by speed, taking into account the mutual influence of the weights and functions parameters on each other.

In this regard, in this paper we propose, for DNN learning the adaptive parametric linear activation function, whose parameters are tuned simultaneously with synaptic weights, and learning algorithms for both an individual neuron and the network as a whole, optimized in the sense of speed and reaching an extremum for the adopted learning criterion (goal function).

II. LEARNING OF THE NEURON WITH ADAPTIVE PARAMETRIC RECTIFIED LINEAR ACTIVATION FUNCTION

As nodes of deep neural networks elementary F. Rosenblatt's perceptrons that realize nonlinear mapping in the form:

$$\hat{y}_j(k) = \psi_j\left(\theta_{j0} + \sum_{i=1}^{n} w_{ji}\, x_i(k)\right) = \\ = \psi_j\left(\sum_{i=0}^{n} w_{ji} x_i(k)\right) = \psi_i\left({w_j}^T x(k)\right) = \\ = \psi_j\left(u_j(k)\right)$$

are used, where $\hat{y}_j(k)$ – output signal of the j-th neuron at discrete time k=1, 2,…, N,…, $\psi_j(\cdot)$ – nonlinear activation function of this neuron, θ_{j0} – bias (threshold), n – number of neurons inputs, w_{ji} – tuned synaptic weight, $x_i(k)$ – input signal on the i-th neuron input at k-th instant of time, $\theta_{j0} = w_{j0}$, $x(k) = \left(1, x_1(k), \dots, x_n(k)\right)^T$ – $(n+1)\times 1$ – vector of the input signals, $w_j = \left(w_{j0}, w_{j1}, \dots, w_{jn}\right)^T$ - $(n+1)\times 1$ – vector of the tuned synaptic weights, $u_j(k)$ – internal activation signal.

The chosen of the nonlinear activation function $\psi_j(\cdot)$ is usually performed based on empirical considerations, but the most popular is $\sigma-$function, considered by G. Cybenko in [1]

$$\hat{y}_j(k) = \psi_j\left(u_j(k)\right) = \\ = \left(1 + exp\left(-\gamma_j u_j(k)\right)\right)^{-1} \tag{1}$$

with derivation:

$$\psi_j'\left(u_j(k)\right) = \gamma_j \hat{y}_j(k)\left(1 - \hat{y}_j(k)\right), \tag{2}$$

where γ_j – so called gain parameter, that describes the shape of activation function and hyperbolic tangent function:

$$\hat{y}_j(k) = \psi_j\left(u_j(k)\right) = tanh\gamma_j u_j(k) \tag{3}$$

with derivation:

$$\psi_j'\left(u_j(k)\right) = \gamma_j\left(1 - \hat{y}_j^2(k)\right). \tag{4}$$

Note that, if in the (1), (2) $\hat{y}_j(k)$ tends to 0 or 1, and in the (3), (4) – to -1 or +1, the effect of the vanishing gradient is arisen. The activation function of the rectified unit family overcomes this effect and can be written as follows:

$$\psi_j\left(u_j(k)\right) = \begin{cases} u_j(k)\ if\ u_j(k) > 0, \\ a_j u_j(k)\ otherwise \end{cases} \tag{5}$$

with derivation:

$$\psi_j'\left(u_j(k)\right) = \begin{cases} 1\ if\ u_j(k) > 0, \\ a_j\ otherwise \end{cases},$$

where parameter a_j is commonly chosen arbitrary. Note also, that in the most popular ReLU $a_j = 0$.

Natural generalization of the (5) is activation function as follows:

$$\psi_j\left(u_j(k)\right) = \begin{cases} a_j^R u_j(k)\ if\ u_j(k) > 0, \\ a_j^L u_j(k)\ otherwise \end{cases} \tag{6}$$

with derivation:

$$\psi'\left(u_j(k)\right) = \begin{cases} a_j^R \; if \; u_j(k) > 0, \\ a_j^L \; otherwise. \end{cases}.$$

In this case, the question naturally arises: how to choose parameters a_j^R, a_j^L for each neuron or how to organize their tuning-learning process. It is clear that in this situation for each neuron should be tuned n+3 parameters instead of traditional n+1 ones. As it is known, the standard F. Rosenblatt's perceptron is adjusted by δ –rule in the form [3]:

$$w_{ji}(k) = w_{ji}(k-1) + \eta(k)\delta_j(k)x_i(k), \quad i = 0,1,2,\dots,n \tag{7}$$

or in the vector form:

$$w_j(k) = w_j(k-1) + \eta(k)\delta_j(k)x(k), \tag{8}$$

where $\eta(k)$ – learning rate parameter, usually chosen empirically and it is not changing in learning process, $\delta_j(k) = \psi_j'\left(u_j(k)\right)e_j(k) -$ δ –error, $e_j(k) = y_j(k) - \psi_j\left(u_j(k)\right) -$ learning error, $y_j(k) -$ reference signal.

Because the learning algorithms (7), (8) contain the derivation of the activation function, the choice of its parameters significantly affects the rate of convergence of this procedure.

For the convergence process improving for activation function (3) in [18] along with the synaptic weights vector $w_j(k)$ it was proposed to tune gain parameter $\gamma_j(k)$ with the procedure:

$$\begin{cases} \gamma_j(k) = \gamma_j(k-1) + \\ +\eta_\gamma(k)e_j(k)\left(1 - \hat{y}_j^2(k)\right)u_j(k), \\ w_j(k) = w_j(k-1) + \eta_j(k)e_j(k) \times \\ \times\left(1 - \hat{y}_j^2(k)\right)x(k) \end{cases}$$

that is not protected from the vanishing gradient effect.

The neuron tuning process with activation function (6) in the each discrete moment of time k was proposed to carry out learning in the form of the two-step procedure [19]. Tuning of the parameters a_j^R, a_j^L (in the next transformations for simplifying the record indexes R and L are temporarily dropped) and based on the adjusted parameters $a_j(k)$ – the synaptic weights vectors are updated.

The tuning process of the parameters $a_j(k)$ is realized by gradient procedure of quadratic learning criterion minimization in the form:

$$\begin{gathered} a_j(k) = a_j(k-1) + \eta_a(k) \times \\ \times\left(y_j(k) - a_j(k-1)u_j(k)\right)u_j(k) = \\ = a_j(k-1) + \eta_a(k) \times \\ \times\left(y_j(k) - a_j(k-1)w_j^T(k-1)x(k)\right) \times \\ \times w_j^T(k-1)x(k). \end{gathered} \tag{9}$$

Thus, the neuron output signal is linearly depends on the adjusted parameters a_j, the procedure (9) can be optimized by the speed where in the optimal value of the learning rate is given by expression:

$$\eta_a(k) = u_j^{-2}(k),$$

i.e., returning to indexes R and L finally it could be rewritten:

$$\begin{cases} a_j^R(k) = a_j^R(k-1) + \\ +\left(y_j(k) - a_j^R(k-1)u_j(k)\right) \times \\ \times u_j^{-1}(k) \; if \; u_j(k) > 0, \\ a_j^L(k) = a_j^L(k-1) + \\ +\left(y_j(k) - a_j^L(k-1)u_j(k)\right) \times \\ u_j^{-1}(k) \; otherwise. \end{cases} \tag{10}$$

Next aposterior learning error after training a_j is introduced:

$$\tilde{e}_j(k) = y_j(k) - a_j(k)w_j^T(k-1)x(k).$$

The gradient procedure of the synaptic weights tuning can be written as follows:

$$\begin{gathered} w_j(k) = w_j(k-1) + \eta(k) = \\ = \left(y_j(k) - a_j(k)w_j^T(k-1)x(k)\right) = \\ = a_j(k)x(k) = w_j(k-1) + \\ +\eta(k)\left(y_j(k) - w_j^T(k-1)\tilde{x}(k)\right)\tilde{x}(k), \end{gathered} \tag{11}$$

where

$$\tilde{x}(k) = a_j(k)x(k).$$

Procedure (11) also can be optimized by speed, and optimum value of $\eta(k)$ is determined by expression:

$$\eta(k) = \|\tilde{x}(k)\|^{-2}$$

and (11) at the same time takes the form:

$$\begin{gathered} w_j(k) = w_j(k-1) + \left(y_j(k) - \right. \\ \left. -w_j^T(k-1)\tilde{x}(k)\right)\|\tilde{x}(k)\|^{-2}\tilde{x}(k) = \\ = w_j(k-1) + \tilde{e}_j(k)\tilde{x}^{+T}(k), \end{gathered} \tag{12}$$

where $(\cdot)^+ -$ symbol of matrix pseudoinversion.

It is easy to see, that (12) is optimal by speed one-step adaptive learning algorithm proposed by Kaczmarz-Widrow-Hoff [20-23]. Because procedures (10), (12) are affected by exploding gradient, its regularized version could be taken into consideration:

$$\begin{cases} a_j(k) = a_j(k-1) + \left(\alpha + u_j^2(k)\right)^{-1} \\ \left(y_j(k) - a_j(k-1)u_j(k)\right)u(k), \\ w_j(k) = w_j(k)(k-1) + (\alpha + \|\tilde{x}(k)\|^2)^{-1} \\ (y_j(k) - w_j^T(k-1)\tilde{x}(k))\tilde{x}(k) \end{cases}, \quad (13)$$

where $\alpha > 0$ – momentum term that is in fact additive form of Kaczmarz's algorithm.

Thus, algorithm (10-13) provides maximal speed of convergence and does not suffer from the vanishing-exploiding gradient. In situation when processing signals are disturbed by noise of arbitrary nature, additional filtering properties could be given to the learning algorithm (12). In this situation the synaptic weights learning procedure takes the form [23]:

$$w_j(k) = w_j(k-1) + r^{-1}(k)\tilde{e}_j(k)\tilde{x}(k) = \\ = w_j(k-1) + \\ +(\beta r(k-1) + \|\tilde{x}(k)\|^2)^{-1}\tilde{e}_j(k)\tilde{x}(k),$$

where $0 \le \beta \le 1 -$ forgetting factor that is the same as (12) when $\beta = 0$ and, with the algorithm of stochastic approximation of Goodwin-Ramadge-Caines [24] at $\beta = 1$. Then, finally, the learning procedure of a single neuron – F. Rosenblatt's perseptron with adaptive rectified linear activation function can be written in the form:

$$\begin{cases} a_j^R(k) = a_j^R(k-1) + \left(r_a^R(k)\right)^{-1} \times \\ \times \left(y_j(k) - a_j^R(k-1)u_j(k)\right)u_j(k), \\ r_a^R(k) = \beta r_a^R(k-1) + u_j^2(k), \\ w_j(k) = w_j(k-1) + \left(r^R(k)\right)^{-1} \times \\ \times \left(y_j(k) - a_j^R(k)w_j^T(k-1)x(k)\right)a_j^R(k)x(k), \\ r^R(k) = \beta r^R(k-1) + \left(a_j^R(k)\right)^2 \|x(k)\|^2 \end{cases} \quad (14)$$

if $w_j^T(k-1)x(k) > 0$, and

$$\begin{cases} a_j^L(k) = a_j^L(k-1) + \left(r_a^L(k)\right)^{-1} \times \\ \times \left(y_j(k) - a_j^L(k-1)u_j(k)\right)u_j(k), \\ r_a^L(k) = \beta r_a^L(k-1) + u_j^2(k), \\ w_j(k) = w_j(k-1) + \left(r^L(k)\right)^{-1} \times \\ \times \left(y_j(k) - a_j^L(k)w_j^T(k-1)x(k)\right)a_j^L(k)x(k), \\ r^L(k) = \beta r^L(k-1) + \left(a_j^L(k)\right)^2 \|x(k)\|^2 \end{cases} \quad (15)$$

otherwise.

Let us note that all described procedures are in fact the gradient optimization algorithms, providing maximum speed of the learning process and possessing by filtering properties.

III. LEARNING OF THE MULTILAYER NEURAL NETWORK BASED ON ADAPTIVE PARAMETRIC RECTIFIED LINEAR UNITS

Let us consider the learning process of the multilayer neural network that contains n inputs, m outputs and Q layers. In this network first hidden layer contains n_1 neurons, q-th layer – n_q neurons and output layer – m neurons respectively. Indexes R and L are omitted again and learning rates parameters determined as in the (14), (15). The output signals of the q-th hidden layer (q=1,2,…, Q) are designated $o_j^{[q]}(k)$, $j = 1,2,\dots,n_q$, and its input signals $-o_j^{[q-1]}(k)$ respectively.

For the neural network learning procedure standard error backpropagation is used, and on the each learning step for every network neuron firstly parameters $a_j^{[q]}$ are specified, next based on them – synaptic weights w_j^a for all neurons in all network layers are specified. Then for Q-th output network layer, on the input of which signals $o_i^{[Q-1]}(k)$, i=1,2,…, n_{q-1} are fed, could be used learning procedure (14), (15) written in the form:

$$\begin{cases} a_j^{[Q]}(k) = a_j^{[Q]}(k) + \eta_a^{[Q]}(k)e_j(k) \times \\ \times u_j^{[Q]}(k), \; j = 1,2,\dots,n_Q = m, \\ \delta_j^{[Q]}(k) = \left(\psi_j^{[Q]}\left(u_j^{[Q]}(k)\right)\right)' \times \\ \times e_j(k) = a_j^{[Q]}(k)e_j(k), \\ w_{ji}^{[Q]}(k) = w_{ji}^{[Q]}(k-1) + \eta^{[Q]}(k)\delta_j^{[Q]}(k)o_i^{[Q-1]}(k), \\ i = 0,1,2,\dots,n_{Q-1} \end{cases}$$

or introducing notations:

$$\begin{cases} \Delta a_j^{[Q]}(k) = a_j^{[Q]}(k) - a_j^{[Q]}(k-1), \\ \Delta w_{ji}^{[Q]}(k) = w_{ji}^{[Q]}(k) - w_{ji}^{[Q]}(k-1), \end{cases}$$

$$\begin{cases} \Delta a_j^{[Q]}(k) = \eta_a^{[Q]}(k)e_j(k)u_j^{[Q]}(k), \\ \delta_j^{[Q]}(k) = \left(\psi_j^{[Q]}\left(u_j^{[Q]}(k)\right)\right)' \times \\ \times e_j(k) = a_j^{[Q]}(k)e_j(k), \\ \Delta w_{ji}^{[Q]}(k) = \eta^{[Q]}(k)\delta_j^{[Q]}(k)o_i^{[Q-1]}(k). \end{cases}$$

The (Q-1)-th hidden layer is tuned according to the relations:

$$\begin{cases} \Delta a_j^{[Q-1]}(k) = \eta_a^{[Q-1]}(k) \times \\ \times \left(\sum_{i=0}^{n_Q} \delta_i^{[Q]}(k) w_{ij}^{[Q]}(k) \right) u_j^{[Q]}(k), \\ \delta_j^{[Q]}(k) = a_j^{[Q-1]}(k) \sum_{i=0}^{n_Q} \delta_i^{[Q]}(k) w_{ij}^{[Q]}(k), \\ \Delta w_{ji}^{[Q]}(k) = \eta^{[Q-1]}(k) \delta_j^{[Q-1]}(k) o_i^{[Q-2]}(k), \end{cases}$$

the q-th hidden layer:

$$\begin{cases} \Delta a_j^{[q]}(k) = \eta_a^{[q]}(k) \times \\ \times \left(\sum_{i=0}^{n_{q+1}} \delta_i^{[q+1]}(k) w_{ij}^{[q+1]}(k) \right) u_j^{[q]}(k), \\ \delta_j^{[q]}(k) = a_j^{[q]}(k) \sum_{i=0}^{n_{q+1}} \delta_i^{[q+1]}(k) w_{ij}^{[q+1]}(k), \\ \Delta w_{ji}^{[q]}(k) = \eta^{[q]}(k) \delta_j^{[q]}(k) o_i^{[q-1]}(k) \end{cases}$$

and, finally, the first network layer:

$$\begin{cases} \Delta a_j^{[1]}(k) = \eta_a^{[1]}(k) \times \\ \times \left(\sum_{i=0}^{n_2} \delta_i^{[2]}(k) w_{ij}^{[2]}(k) \right) u_j^{[1]}(k), \\ \delta_j^{[1]}(k) = a_j^{[1]}(k) \sum_{i=0}^{n_2} \delta_i^{[2]}(k) w_{ij}^{[2]}(k), \\ \Delta w_{ji}^{[1]}(k) = \eta^{[1]}(k) \delta_j^{[1]}(k) x_i(k), \\ x_0(k) = 1. \end{cases}$$

Proposed relations differ from the standard error backpropagation procedure, that in the learning process not only synaptic weights are tuned, but also the activation functions parameters. Moreover, calculated by a special way values of the learning rate parameters obtained for the tuning process high speed permit to decrease total learning time of the deep neural network.

IV. EXPERIMENTAL MODELING

For the evaluation of the proposed deep neural network with adaptive parametric rectified linear units data set “Carvana” was taken from Kaggle platform. This data set was proposed by Carvana company for segmentation task in 2017. Carvana data set includes 318 images of cars visualization in the 16-th different views (angles). Every image has 1918x1280 pixels resolution. All data set consists of the 5088 marked images and every segmentation mask includes two classes: background and foreground. All experiments were realized in the TensorFlow 2.4.0.

Examples of the images and their masks are demonstrated Fig. 1. As a prototype of the deep neural network U-Net network was used. Before network training data preprocessing was made all input images had been resized (256, 256, 3), random regularization for image tint that should be in the interval [0; 0,5], horizontal images rotation on the central axes with probability equal to 0,5 and rescale (1/255) also were made.

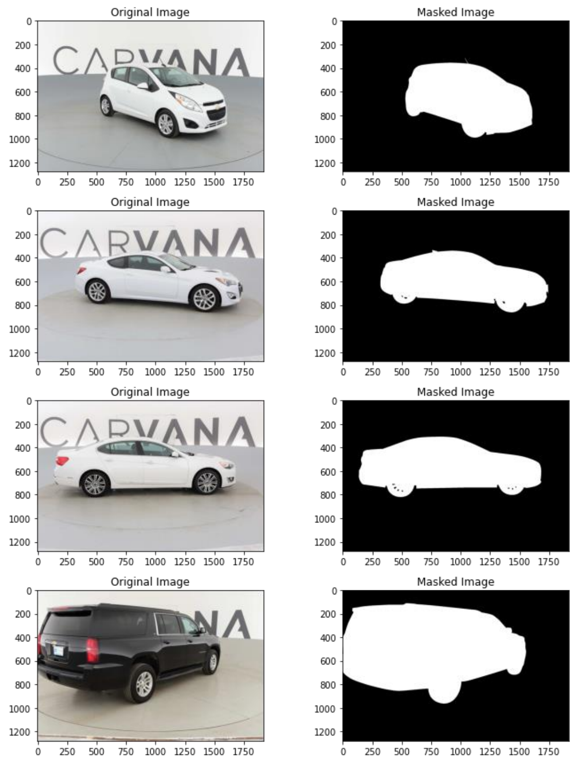

Figure 1. Examples of the images and their masks

Adjusted parameters of U-Net base modal with standard ReLU and Adaptive ReLU are shown in Table 1. The second part of Table 1 shows only adjusted parameters for the Adaptive ReLU, first part is similar for standard ReLU and Adaptive ReLU.

Table 1. Adjusted parameters of the standard ReLU and Adaptive ReLU

Parameters of the U-Net base model with standard ReLU	
The loss function	Dice loss + Binary Cross-Entropy
Numbers of epoch	10
Starter earning rate parameter (lr)	0.001
Parameters for the Adaptive ReLU	
lr_forgetting factor (β)	0, 0.3, 0.6, 0.9 (different for each experiment)
The initial a_j^R	0.5
The initial a_j^L	0.05

The internal activation signal u_j^2 was normalized by l2-normalization with respect to all elements of the tensor, otherwise values of parameters r_a^R, r_a^L became very big, that led to exploding of the weights coefficients. Then, the mean value of the internal activation signal u_j^2 was used to calculate the new learning rate.

The graphic of the train loss function is demonstrated in Fig. 2 and in Fig. 3 the validation loss function graphic is shown.

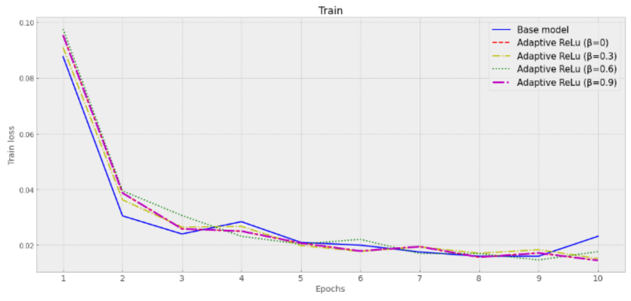

Figure 2. The train loss function graphic

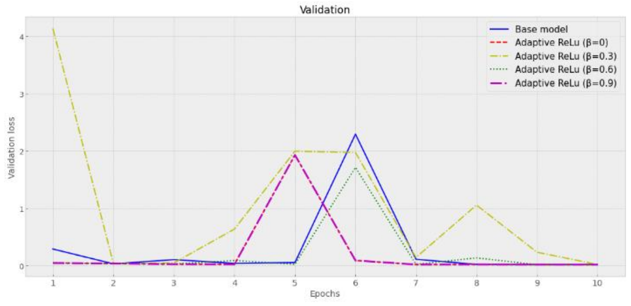

Figure 3. The validation loss function graphic

Visualization of the train dice coefficient is demonstrated in Fig. 4 and in Fig. 5 the validation dice coefficient is shown.

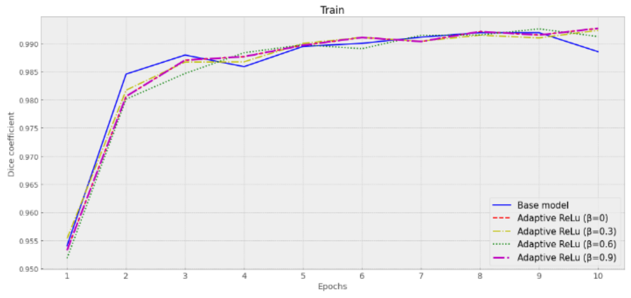

Figure 4. The train dice coefficient

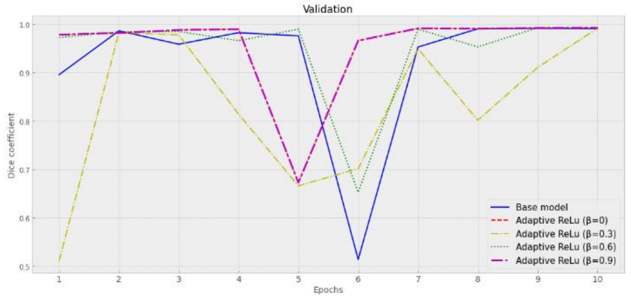

Figure 5. The validation dice coefficient

As can be seen in the graphs in Fig. 3 and Fig. 5 the metric on the validation dataset pulls, that indicates a small number of learning epochs. In Table 2 numerical results of train and validation dice coefficients are presented for base U-net model and for U-Net with adaptive ReLU activation function. In Fig. 6 angles changing of the adaptive ReLU are presented.

Table 2. Numerical results of train and validation dice coefficients

Model	Train dice coefficient	Validation dice coefficient
Base Model	0.9886	0.9909
Adaptive ReLu (β=0)	0.9927	0.9928
Adaptive ReLu (β=0.3)	0.9923	0.9918
Adaptive ReLu (β=0.6)	0.9912	0.9917
Adaptive ReLu (β=0.9)	0.9931	0.9914

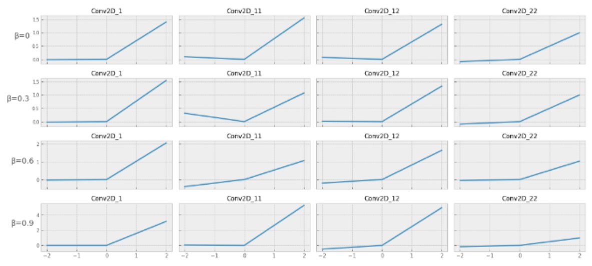

Figure 6. Changing the angles of Adaptive ReLU

Experimental results show that with the same accuracy of solving the problem under consideration, parameters adaptation of the Adaptive ReLU activation function can reduce the learning rate by approximately 10 %, while the smaller is value of the forgetting factor \beta, the faster is the neural network tuning, i.e., the learning algorithm approaches the speed-optimal Kaczmarz-Widrow-Hoff procedure.

Segmentation results using the model based on the U-net deep neural network with standard ReLU activation function are presented in Fig. 7. And finally, in Fig. 8, Fig. 9, Fig. 10 and Fig. 11 the U-Net neural network with Adaptive ReLU activation function are shown. Fig. 8 demonstrates using adaptive ReLU with forgetting factor 0.9, Fig. 9 demonstrates using adaptive ReLU with forgetting factor 0.6, Fig. 10 demonstrates using adaptive ReLU with forgetting factor 0.3 and Fig. 11 demonstrates using adaptive ReLU with forgetting factor 0.

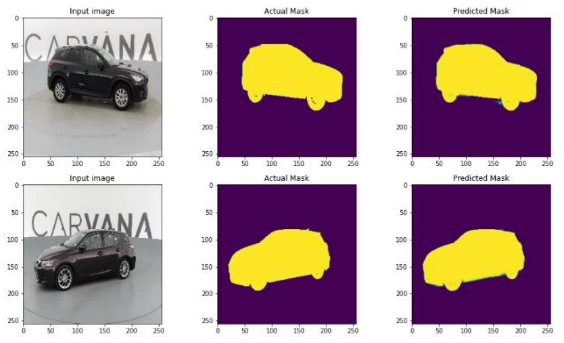

Figure 7. Segmentation results by base model

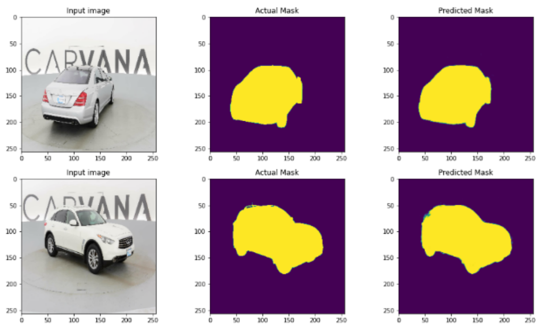

Figure 8. U-Net neural network with Adaptive ReLU activation function with forgetting factor 0.9

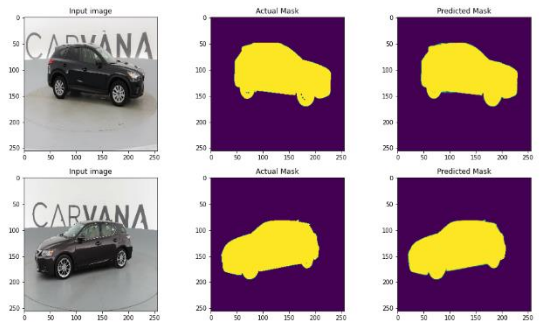

Figure 9. U-Net neural network with Adaptive ReLU activation function with forgetting factor 0.6

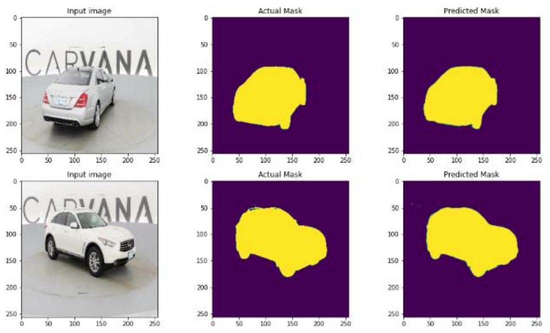

Figure 10. U-Net neural network with Adaptive ReLU activation function with forgetting factor 0.3

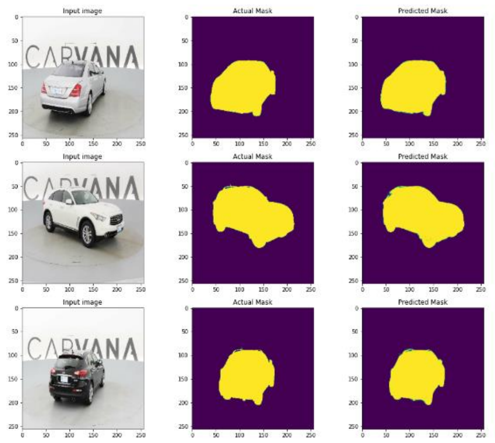

Figure 11. U-Net neural network with Adaptive ReLU activation function with forgetting factor 0

The experimental modeling has proved theoretical researches and shows that MLP with adaptive parametric rectified linear units could be used in the CNN with traditional fully connected layers.

V. CONCLUSION

The task of deep neural network training with adaptive parametric rectified linear activation function, whose parameters are adjusted simultaneously with synaptic weights is considered in the paper. Adaptive optimal learning algorithms for all network parameters with additional filtering properties are introduced. Based on these algorithms, the learning procedure based on the error backpropagation that permits to reduce total learning time of the neural network in general is proposed.

References

[1] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Control Signals Systems*, vol. 2, pp. 303-314, 1989, https://doi.org/10.1007/BF02551274.

[2] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, issue 2, pp. 251-257, 1994, https://doi.org/10.1016/0893-6080(91)90009-T.

[3] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, Wiley, Chichester, 1993, 536 p. https://doi.org/10.1002/acs.4480080309.

[4] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015, https://doi.org/10.1038/nature14539.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85-117, 2015, https://doi.org/10.1016/j.neunet.2014.09.003.

[6] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, The MIT Press, 2016.

[7] D. Graupe, *Deep Learning Neural Networks: Design and Case Studies*, New Jersey: World Scientific, 2016. https://doi.org/10.1142/10190.

[8] A. L. Caterini, D. E. Chang, *Deep Neural Networks in a Mathematical Framework*, Cham: Springer, 2018, https://doi.org/10.1007/978-3-319-75304-1.

[9] C. C. Aggarwal, *Neural Networks and Deep Learning*, Cham: Springer, 2018, https://doi.org/10.1007/978-3-319-94463-0.

[10] B. Xu, N. Wang, T. Chen, M. Li, *Empirical Evaluation of Rectified Activations in Convolutional Network*, arXiv preprint arXiv:1505.00853, 2015.

[11] D.-A. Clevert, T. Unterthiner and S. Hochreiter, *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*, arXiv preprint arXiv:1511.07289, 2015.

[12] K. He, X. Zhang, S. Ren and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 1026-1034, https://doi.org/10.1109/ICCV.2015.123.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778, https://doi.org/10.1109/CVPR.2016.90.

[14] I. Goodfellow, D. Warde-Farley, M. Mirza, A.Courville and Y. Bengio, *Maxout Networks*, arXiv preprint arXiv:1302.4389, 2013.

[15] F. Agostinelli, M. Hoffman, P. Sadowski and P. Baldi, *Learning Activation Functions to Improve Deep Neural Networks*, arXiv preprint arXiv:1412.6830, 2015.

[16] X. Jin, Ch. Xu, J. Feng, Yu. Wei, J. Xiong and Sh. Yan, *Deep Learning with S-shaped Rectified Linear Activation Units*, arXiv preprint arXiv:1512.07030, 2015.

[17] L.R. Sütfeld, F. Brieger, H. Finger, S. Füllhase and G. Pipa, *Adaptive Blending Units: Trainable Activation Functions for Deep Neural Networks*, arXiv preprint arXiv:1806.10064, 2018.
[18] J. K. Kruschke, J. R. Movellan, "Benefits of gain: Speeded learning and minimal hidden layers in back-propagation networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 1, pp. 273-280, 1991, https://doi.org/10.1109/21.101159.
[19] Y. Bodyanskiy, A. Deineko, I. Pliss and V. Slepanska, "Formal neuron based on adaptive parametric rectified linear activation function and its learning," *Proceedings of the 1st International Workshop on Digital Content and Smart Multimedia "DCSMart 2019",* Lviv, Ukraine, December 23-25, 2019, CEUR Workshop Proceedings volume 2533, pp. 14-22.
[20] S. Kaczmarz, "Angenäherte Auslösung von Systemen linearer Gleichungen," *Bull. Internat. Acad. Polon.Sci., Lettres A*, pp. 355-357, 1937. (in German)
[21] S. Kaczmarz, "Approximate solution of systems of linear equations," *International Journal of Control*, vol. 57, issue 6, pp. 1269-1271, 1993, https://doi.org/10.1080/00207179308934446.
[22] B. Widrow, M. Hoff, "Adaptive Switching Circuits," *IRE WESCON Convention Record*, New York, IRE, Part 4, 1959, pp. 96-104. https://doi.org/10.21236/AD0241531.
[23] P. Otto, Y. Bodyanskiy and V. Kolodyazhniy, "A new learning algorithm for a forecasting neuro-fuzzy network," *Integrated Computer-Aided Engineering*, vol. 10, no. 4, pp. 399-409, 2003. https://doi.org/10.3233/ICA-2003-10409.
[24] G. C. Goodwin, P. J. Ramadge and P. E. Caines, "A globally convergent adaptive predictor," *Automatica*, vol. 17, issue 1, pp. 135-140, 1981. https://doi.org/10.1016/0005-1098(81)90089-3.

YEVGENIY BODYANSKIY, Profes–sor at the Department of Artificial Intelligence, Scientific Head at the Control Systems Research Laboratory (CSRL), Kharkiv University of Radio Electronic, Member of the specialized scientific council, Member of STC, IEEE Senior Member, Doctor of Technical Sciences, Professor. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Deep Learning, Evolving Systems.

ANASTASIIA DEINEKO, Associated Professor of AI Department, Candidate of Technical Sciences, Senior Scientist Researcher at the CSRL, Kharkiv University of Radio Electronic. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Deep Learning, Evolving Systems.

VIKTORIA SKORIK, Student, Artificial Intelligence Department, Kharkiv University of Radio Electronics. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Deep Learning, Evolving Systems.

FILIP BRODETSKYI, Ph.D. Student, Assistant of Informatics Depart–ment, Kharkiv University of Radio Electronics. Scientific interests: Computer Vision, Signal Processing, Pattern Recognition, Texture Analysis, Systems and Applications.