Janusz Kolbusz, Paweł Różycki
Wyższa Szkoła Informatyki i Zarządzania w Rzeszowie

Tomasz Bartczak
Akademia Finansów i Biznesu Vistula – Warszawa

The Study of the Influence of Architecture on Effectiveness of Deep Neural Networks Training

Summary

Paper presents impact of the neural network architecture on the training effectiveness and training time. Selected network architectures and training algorithm are described. Presented experimental results of research confirming the significant influence of architecture on the success of network training.

Key words: deep learning, ANN architectures, Bridged MLP, NBN.

JEL codes: L86

Background

One of the fast growing fields of science is artificial intelligence. It allows you to build systems that significantly exceed human intelligence possibilities. AI-based systems allow to solve complex problems without the need for mathematical modeling, which can solve many practical problems that cannot be described with mathematical equations and when they are unable to understand in details by human perception.

Training neural networks, many researchers build neural networks in the MLP (Multi-Layer Perceptron) architecture and use a simple EBP (*Error Back Propagation*) algorithm (Rumelhart, Hinton, Williams 1986, pp. 533-536; Wilamowski, Yu 2010a, pp. 1793-1803). Current studies show, however, that the EBP algorithm is slow, inefficient, and not capable for teaching a network other than in the MLP architecture. Historically, a great achievement in learning neural networks was the development of an LM algorithm (*Levenberg-Marquardt*). This algorithm is able to teach the network in a significantly smaller number of iterations (up to 100 times smaller). However, it has a major disadvantage – its usage to solve more complex problems is greatly limited because the size of the Jacobian matrix is proportional to the number of patterns. Also our research confirms that the number of patterns is important factor for effectiveness of most neural network training algorithms.

One of the most popular neural network learning methods is the MLP architecture trained with EBP algorithm. Such networks need to determine the optimal number of hidden neu-

rons before start of training process. Many scientific papers devoted to the use of the EBP learning algorithm or its modifications to the MLP network (Bengio 2009; Yu et al. 2014, pp. 1793- 1803; Hang et al. 2013; Lalis et al. 2013; Kaytez et al. 2015, pp. 431-438) mainly concern the improvement of efficiency or the reduction of the learning time of such networks. One of the way to train such networks is to train with a small number of patterns. Once a learning error has been established, the number of patterns is gradually increased in the next stages of learning (Boughrara et al. 2016, pp. 709-731).

However, due to the MLP architecture has limited capabilities, new deep neural network architectures such as BMLP (*Bridged MLP*) (Rumelhart, Hinton, Williams 1986, pp. 533-536; Fahlman, Lebiere 1990, pp. 524-532; Paczek, Adhikari 2014, pp. 227-240) or DNN (*Dual Neutral Network*) have been developed that make it possible to solve up to 100 times more complex problems with the same number of neurons (Lang, Witbrock 1988). Literature provides us with many new or improved learning algorithms for neural networks, but most of them are only for shallow architectures, such as ErrCor (*Error Correction*) algorithm (Yu et al. 2012, pp. 609-619) or fully connected MLP architecture. However, training capability of such specific architectures with connections across network layers like BMLP, is limited due to lack of algorithms that support such networks. The only known by authors algorithm that support new architectures with arbitral connected neural units including both MLP and BMLP is the NBN algorithm published w *IEEE Transactions on Neural Networks and Learning Systems*, so in our study, this algorithm has been used.

Neural network architectures

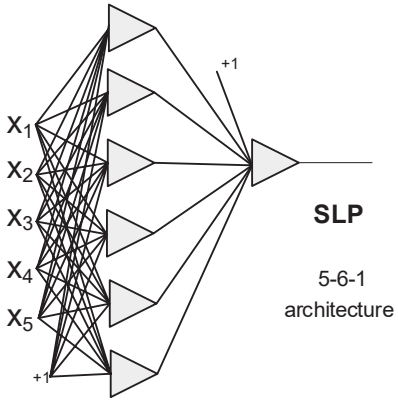
The paper demonstrates that the manner in which neural units are connected in the network has a great impact on neural network training. One of the most challenging benchmarks for testing the capability of neural networks are Parity-N problems. These benchmarks have the advantage because their complexity increases with the increase of N, so it is relatively easy to evaluate and to compare the capabilities of different neural network architectures. For example, one may use the same number of differently connected neurons and find which of the most complex Parity-N problems can be solved. For instance, 10 neurons in the most commonly used three-layer MLP architecture (with one hidden layer), the biggest problem that such a network can solve is a Parity-9 problem. The same 10 neurons connected in the BMLP architecture, is able to solve the Parity-1023 problem. It is therefore clear that the changing the commonly used MLP for more sophisticated BMLP architecture with the same number of neurons, increases the network capacity even a hundredfold (Wilamowski, Hunter, Malinowski 2003; Hunter et al. 2012, pp. 228-240). The problem, however, is that most popular training algorithms such as EBP or LM cannot cope with these new highly effective architectures. Therefore, not only architecture, but also the teaching algorithm is important.

Let us briefly consider three types of architectures: SLP (*Single-Layer Perceptron*), MLP (*Multi-Layer Perceptron*), BMLP (*Bridged MLP*), as shown in Figure 1

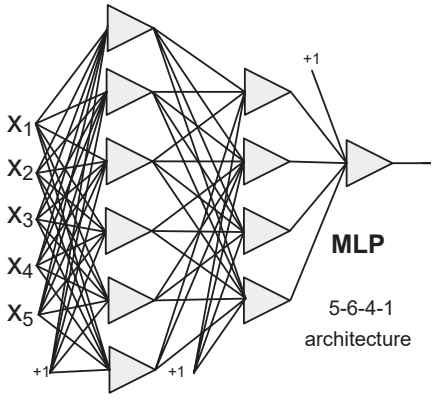
Figure 1

The architectures of the neural networks

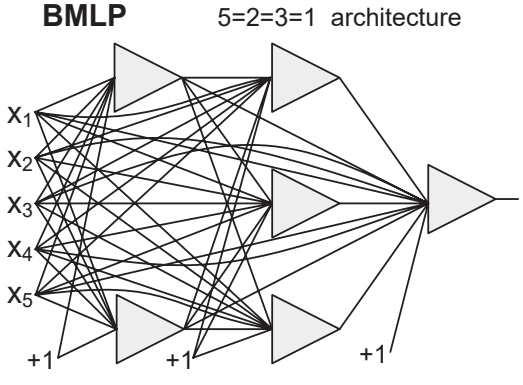
a) SLP (Single Layer Perceptron)



b) MLP (Multi-Layer Perceptron)



c) BMLP (Bridged MLP)



Source: own preparation.

These architectures have different capabilities of solving Parity-N problems (Wilamowski, Yu 2017) with hard threshold activation functions:

- SLP (MLP with a single hidden layer) architecture with 7 neurons (Fig. 1(a)) can solve a Parity-6 problem.
- MLP architecture with 11 neurons (Fig. 1(b)) can solve a Parity-23 problem. If these 11 neurons are connected as SLP then only a Parity-10 problem can be solved
- BMLP architecture with 6 neurons (Fig. 1(c)) can solve a Parity-23 problem.

Experimental results

To demonstrate that the architecture of a trained neural network has a significant impact on the achieved results, a two well-known benchmarks have been used: Rastrigins function and Schwefel function, given by equations (1) and (2) respectively

$$f_R(x) = 10d + \sum_{i=1}^{d} \left(x_i^2 - 10cos(2\pi x_i) \right) \tag{1}$$

$$f_S(x) = 418.9829d + \sum_{i=1}^{d} \left(x_i sin\left(\sqrt{|x_i|} \right) \right) \tag{2}$$

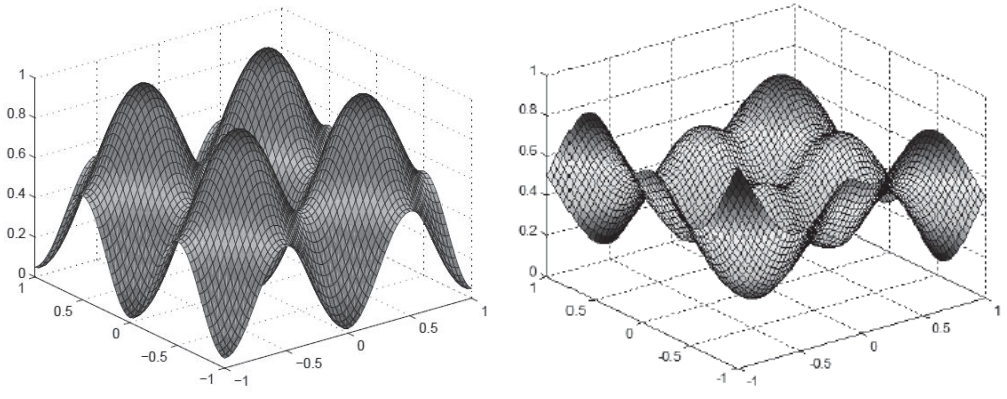
where *d* is number of dimensions.

The normalized two-dimensional graphs of these functions are shown in Figure 2.

Figure 2

Two-dimensional functions used in experiments

a) Rastrigins function **b) Schwefel function**



Source: as in Figure 1.

Experiments have been done for two- and three-dimensional set of data. All problems have been tried to resolve with neural networks with different architectures from two-layer MLP shallow network to multilayer BMLP deep network (up to 8 layers and up to 8 neurons in each layer) and second-order NBN algorithm. In all experiments NBN 2.08 software (Wilamowski, Yu) have used.

Table 1

Success rates for Rastrigins 2D function approximation with MLP and BMLP architectures and NBN algorithm

Number of hidden layers	Number of neurons per hidden layer	Number of hidden neurons	Rastrigins function 2D – number of training-100, Maximum Error – 0.01				
			Succ. Rate	Number of weights	Average number of iterations	Average traning time [s]	Network architecture
2	4	8	0,54	50	2607,61	12,08	MLP
2	4	8	0,94	51	1145,90	4,56	BMLP
2	8	16	0,93	122	1843,61	35,71	MLP
2	8	16	1	131	783,81	13,38	BMLP
3	3	9	0,38	59	3361,35	18,62	MLP
3	3	9	1	66	1183,90	5,34	BMLP
3	7	21	1	171	2946,00	116,64	MLP
3	7	21	1	234	623,40	35,52	BMLP
4	2	8	0,03	58	2579,50	13,11	MLP
4	2	8	0,98	59	1197,70	4,86	BMLP
4	7	28	0,94	243	3187,42	267,66	MLP
4	7	28	1	409	453,00	113,91	BMLP
5	2	10	0,05	77	4676,00	30,80	MLP
5	2	10	1	83	768,45	7,11	BMLP
5	6	30	0,95	253	4104,60	408,43	MLP
5	6	30	1	483	469,00	196,31	BMLP
6	2	12	0	98	-	-	MLP
6	2	12	1	111	824,88	11,57	BMLP
6	7	42	1	393	3244,00	922,93	MLP
6	7	42	1	906	875,00	4748,73	BMLP
7	2	14	0	121	-	-	MLP
7	2	14	1	143	604,47	17,52	BMLP
7	8	56	1	577	3357,03	3775,35	MLP
7	8	56	1	1571	966,33	24607,46	BMLP
8	1	8	0	107	-	-	MLP
8	1	8	0,98	63	997,20	5,98	BMLP
8	7	56	0,74	551	3177,66	1882,45	MLP
8	7	56	1	1599	1915,33	31386,99	BMLP

Source: own preparation.

Table 2

Success rates for Rastrigins 3D function approximation with MLP and BMLP architectures and NBN algorithm

Number of hidden layers	Number of neurons per hidden layer	Number of hidden neurons	Rastrigins function 3D – number of training-100, Maximum Error – 0.01				
			Succ. Rate	Number of weights	Average number of iterations	Average traning time [s]	Network architecture
2	6	12	0,19	90	3112,85	29,35	MLP
2	6	12	0,97	100	1464,60	17,03	BMLP
2	8	16	0,97	132	1828,02	36,69	MLP
2	8	16	1	148	968,48	21,21	BMLP
3	5	15	0,88	115	2185,22	58,76	MLP
3	5	15	1	154	846,08	24,13	BMLP
3	7	21	1	181	1513,90	42,33	MLP
3	7	21	1	256	722,96	48,22	BMLP
4	3	12	0,57	122	3349,48	56,18	MLP
4	3	12	0,98	118	1186,42	21,00	BMLP
4	8	32	0,98	310	2261,71	229,11	MLP
4	8	32	1	548	587,21	222,75	BMLP
5	2	10	0,07	117	4674,00	92,84	MLP
5	2	10	0,9	94	1690,73	19,98	BMLP
5	7	35	0,78	329	2738,05	390,77	MLP
5	7	35	1	669	554,64	350,64	BMLP
6	2	12	0,11	146	3709,50	106,10	MLP
6	2	12	0,98	124	1029,62	22,84	BMLP
6	8	48	0,98	496	3906,40	1377,49	MLP
6	8	48	1	1204	467,50	4008,12	BMLP
7	2	14	0,09	177	3818,89	157,59	MLP
7	2	14	0.89	158	2759,40	51,87	BMLP
7	8	56	0,65	592	3112,40	2104,32	MLP
7	8	56	1	1628	1875.00	39601,20	BMLP
8	2	16	0,08	210	3639,83	225,35	MLP
8	2	16	1	196	743,51	38,96	BMLP
8	8	64	0,73	688	1835,50	2193,64	MLP
8	8	64	1	2116	432,05	17592,00	BMLP

Source: as in Table 1.

Table 3

Success rates for Schwefel 2D function approximation with MLP and BMLP architectures and NBN algorithm

Number of hidden layers	Number of neurons per hidden layer	Number of hidden neurons	Schwefel function 2D – number of training-100, Maximum Error – 0.01				
			Succ. Rate	Number of weights	Average number of iterations	Average traning time [s]	Network architecture
2	3	6	0,45	37	3795,11	13,76	MLP
2	3	6	0,82	36	3048,78	8,41	BMLP
2	8	16	0,83	122	3563,36	59,02	MLP
2	8	16	0,99	131	2016,60	33,99	BMLP
3	3	9	0,37	59	3773,78	28,41	MLP
3	3	9	0,92	66	2570,33	15,46	BMLP
3	8	24	0,91	209	3113,19	210,76	MLP
3	8	24	1	291	1833,86	153,94	BMLP
4	2	8	0,31	58	3650,80	35,68	MLP
4	2	8	0,83	59	2638,05	13,04	BMLP
4	8	32	0,79	298	3353,00	454,07	MLP
4	8	32	1	512	1998,00	1112,62	BMLP
5	1	5	0,20	77	3720,08	112,58	MLP
5	1	5	0,64	33	3300,31	7,49	BMLP
5	8	40	0,62	389	3819,33	1200,24	MLP
5	8	40	1	803	1433,00	2698,05	BMLP
6	1	6	0	69	-	-	MLP
6	1	6	0,79	42	2971,66	9,94	BMLP
6	8	48	0,45	482	3979,08	1200,38	MLP
6	8	48	1	1155	2041,50	22093,25	BMLP
7	1	7	0,12	121	3935,60	82,26	MLP
7	1	7	0,85	52	2863,03	15,56	BMLP
7	7	49	0,59	471	1756,00	825,01	MLP
7	7	49	1	1228	1635,50	24480,94	BMLP
8	1	8	0	107	-	-	MLP
8	1	8	0,86	63	2604,00	37,12	BMLP
8	8	64	0,13	674	3521,33	3378,52	MLP
8	8	64	0,94	2051	2841,66	46174,52	BMLP

Source: as in Table 1.

Table 4

Success rates for Schwefel 3D function approximation with MLP and BMLP architectures and NBN algorithm

Number of hidden layers	Number of neurons per hidden layer	Number of hidden neurons	Schwefel function 3D – number of training-100, Maximum Error – 0.01				
			Succ. Rate	Number of weights	Average number of iterations	Average traning time [s]	Network architecture
2	5	10	0.05	72	4258,50	28,00	MLP
2	5	10	0.44	79	3245.60	22,10	BMLP
2	8	16	0.45	132	3136,00	76,32	MLP
2	8	16	0.94	148	2698.71	50,19	BMLP
3	4	12	0.04	88	3387,00	31,82	MLP
3	4	12	0.76	112	2901.13	33,77	BMLP
3	8	24	0.56	220	3803,00	255,48	MLP
3	8	24	1	316	2394.00	250,57	BMLP
4	3	12	0.17	122	2959,00	62,15	MLP
4	3	12	0.80	118	3342,67	39,42	BMLP
4	8	32	0.63	310	3016,42	468,92	MLP
4	8	32	1	548	2926,00	1067,55	BMLP
5	3	15	0.02	117	4166,40	81,08	MLP
5	3	15	0.97	169	2759,66	65,66	BMLP
5	8	40	0.94	402	3090,05	855,49	MLP
5	8	40	1	844	1495,00	1938,30	BMLP
6	2	12	0.06	146	4949,00	174,47	MLP
6	2	12	0.84	124	2945,23	43,41	BMLP
6	8	48	0.85	496	3091,47	1411,32	MLP
6	8	48	1	1204	1435,50	7406,20	BMLP
7	2	14	0.03	177	4565,00	228,33	MLP
7	2	14	0.93	158	2357,40	61,87	BMLP
7	8	56	0.78	592	3257,80	2569,20	MLP
7	8	56	1	1628	1691.00	43601,20	BMLP
8	2	16	0.02	210	3192,08	539,01	MLP
8	2	16	0.58	196	1741,70	71,53	BMLP
8	8	64	0.56	688	2841,66	6974,62	MLP
8	8	64	1	2116	4044,00	123873,73	BMLP

Source: as in Table 1.

In experiments the following values have been measured: success rate (SR) that as a percentage of successfully trained (error below 0.01) cases for 100 trials, number of iterations needed to train with success and training time in these cases.

Selected experiment results have been shown in tables 1-4. The tables show the results of experiments to show the difference between used architectures. They contain also the number of neurons and the number of weights in each case.

For each number of layers have been selected cases there the SR for MLP architecture and BMLP architecture are similar (grayed). As can be observed neural network with MLP architecture requires at least two times more neurons to achieve similar results as neural network in BMLP architecture. Moreover, to train with success MLP ANN much more iterations are needed and it takes much more time.

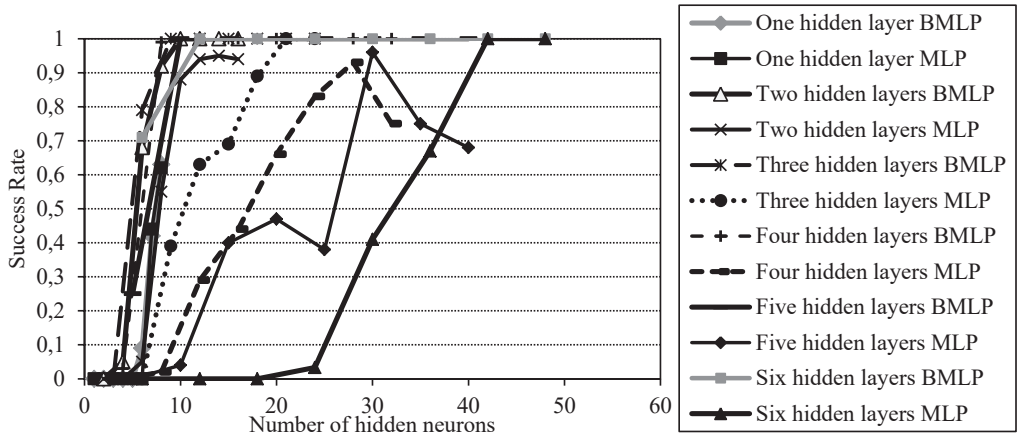
For instance, as shown in Table 1, four-layer MLP network require 28 neurons (7 neurons in each layer) to resolve Rastrigins 2D problem with almost the same result as 8 neurons in four-layer or two-layer BMLP configuration with almost 3 times more iterations and over 60 times longer training process. Similar relations can be observed in other presented results.

Charts in Figure 3 and Figure 4 present the results achieved for Rastrigins and Schwefel functions respectively for different MLP and BMLP architectures with respect on number of neurons in hidden layers.

As shown neural network with BMLP architecture is able to resolve problems with much less neural units. Practically networks with 15 neurons in such architecture are successfully trainable regardless on number of layers. In the contrary, network with MLP architecture cannot be trained

Figure 3

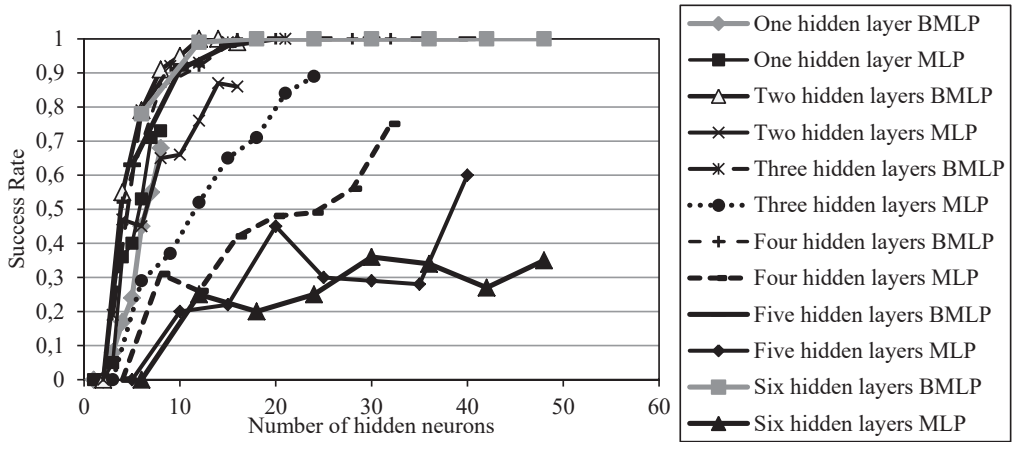
Rastrigins function 2D – number of training-100, Maximum Error– 0.01



Source: as in Figure 1.

Figure 4

Shwefel function 2D – number of training-100, Maximum Error– 0.01



Source: as in Figure 1.

with success especially in deeper configurations where effect of so called "vanishing gradient" is observable, and which is reduced by cross layer connections implemented in BMLP architecture.

Conclusions

The paper demonstrates experimentally that the capabilities of the neural network depend on the architecture used and that they grow with the depth of the network. The results of the research show that in order to achieve a similar success in training MLP network architecture, we need to use much more neurons training the network 2, 3 times longer than more sophisticated BMLP network.

The results also confirm that the widely used MLP architecture is not suitable for deep learning, because if the number of hidden layers exceeds 3, the success training of such network is rapidly decreasing. Studies have shown that training an MLP architecture with 6 or more hidden layers is practically impossible due to so called vanishing gradient effect. Introducing connections across a layers, using the BMLP architecture this problem can be practically eliminated.

Bibliography

Bengio Y. (2009), *Learning deep architectures for AI*, "Foundations and Trends in Machine Learning", No. 2(1).

Boughrara H. et al. (2016), *Facial expression recognition based on a mlp neural network using constructive training algorithm*, "Multimedia Tools and Applications", No. 75.2.

Fahlman S.E., Lebiere C. (1990), *The cascade-correlation learning architecture*, (in:) Touretzky D.S. (Ed.) *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, San Mateo.

Hang S. et al. (2013), *Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription*, *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Hunter D., Yu H., Pukish M.S., Kolbusz J., and Wilamowski B.M. (2012), *Selection of Proper Neural Network Sizes and Architectures – A Comparative Study*, "IEEE Trans. on Industrial Informatics", Vol. 8.

Kaytez F. et al. (2015), *Forecasting electricity consumption: a comparison of regression analysis, neural networks and least squares support vector machines*, "International Journal of Electrical Power & Energy Systems", No. 67.

Lalis J.T. et al. (2013), *Ubiquitous Stopping Criterion for Backpropagation Learning in Multilayer Perceptron Neural Networks*.

Lang K.L., Witbrock M.J. (1988), *Learning to Tell Two Spirals Apart Proceedings of the 1988 Connectionists Models Summer School*, Morgan Kaufman.

Paczek S., Adhikari B. (2014), *Analysis of MultiLayer Neural Networks with Direct and Cross Forward Connection*, "Fundamenta Informaticae", Vol. 133, No. 2-3.

Rumelhart D.E., Hinton G.E., Williams R.J. (1986), *Learning representations by back-propagating errors*, "Nature", Vol. 323.

Wilamowski B.M., Hunter D., Malinowski A. (2003), *Solving parity-N problems with feedforward neural networks*, Proc. 2003 IEEE IJCNN.

Wilamowski B.M., Yu H. (2010a), *Neural Network Learning Without Backpropagation*, "IEEE Trans. on Neural Networks", Vol. 21, No.11.

Wilamowski B.M., Yu H. (2010b), *Improved Computation for Levenberg Marquardt Training*, "IEEE Trans. on Neural Networks", Vol. 21, No. 6.

Wilamowski B.M., Yu H. (2017), *NNT – Neural Networks Trainer*, http://www.eng.auburn.edu/ wilambm/nnt/ [access: 15.09.2017]

Xie T., Yu H., Hewlett J., Rozycki P., Wilamowski B.M. (2012), *Fast and Efficient Second-Order Method for Training Radial Basis Function Networks*, "IEEE Trans. on Neural Networks and Learning Systems", Vol. 23, No. 4.

Yu H., Reiner P., Xie T., Bartczak T., Wilamowski B.M. (2014), *An Incremental Design of Radial Basis Function Networks*, "IEEE Transactions on Neural Networks and Learning Systems", Vol. 25, No. 10.

Yu H., Xie T., Hewlett J., Rozycki P., Wilamowski B.M. (2012), *Fast and Efficient Second Order Method for Training Radial Basis Function Networks*, "IEEE Transactions on Neural Networks", Vol. 24, Iss. 4.

This work was supported in part by the National Science Centre, Krakow, Poland, under Grant No. 2015/17/B/ST6/01880

Badania wpływu architektury na skuteczność uczenia głębokich sieci neuronowych

Streszczenie

W artykule przedstawiono wpływ architektury sieci neuronowej na skuteczność i czas uczenia sieci. Opisano wybrane architektury sieci, algorytm uczenia oraz zaprezentowano wyniki badań potwierdzających znaczący wpływ architektury na sukces uczenia sieci.

Słowa kluczowe: uczenie głębokie, architektury sztucznych sieci neuronowych, Bridged MLP, NBN.

Kody JEL: L86

Artykuł zaakceptowany do druku w lutym 2018 roku

Afiliacja:
dr inż. Janusz Kolbusz
dr inż. Paweł Różycki
Wyższa Szkoła Informatyki i Zarządzania w Rzeszowie
Wydział Informatyki Stosowanej
Katedra Elektroniki i Telekomunikacji
ul. Sucharskiego 2
35-225 Rzeszów
e-mail: jkolbusz@wsiz.rzeszów.pl
e-mail: prozycki@wsiz.rzeszów.pl

dr inż. Tomasz Bartczak
Akademia Finansów i Biznesu Vistula
Wydział Inżynierski
ul. Stokłosy 3
02-787 Warszawa
e-mail: t.bartczak@vistula.edu.pl