2019 IEEE Symposium Series on Computational Intelligence (SSCI)
December 6-9 2019, Xiamen, China

Feature Significance in Wide Neural Networks

Janusz A. Starzyk, Rafał Niemiec
University of Information Technology and Management
in Rzeszow, Department of Applied Information Systems,
35-225 Rzeszow, Poland

Adrian Horzyk
AGH University of Science and Technology in Krakow,
Department of Biocybernetics and Biomedical
Engineering, 30-059 Krakow, Poland

***Abstract*—Wide neural networks were recently proposed as a less costly alternative to deep neural networks. In this paper, we analyze the properties of wide neural networks regarding feature selection and their significance. We compared the random selection of weights in the hidden layer to the selection based on radial basis functions. Wide neural networks were also compared with fully connected cascade networks. Feature significance was introduced as a measure to compare various feature selection techniques. Another performance measure introduced in this paper – incremental feature significance - determines the level of improvement that results from selecting only some features, which were added to the existing features, rather than replacing one set of features with another. In both cases, we can also estimate the number of features saved by replacing the original features with the selected ones for which recognition levels improve. This approach can be applied to wide networks that use different feature selection methods than those that are analyzed in this paper; like a k-nearest neighbor, an autoencoder etc.**

***Keywords*—Broad neural networks; feature significance; incremental feature significance.**

I. Introduction

In 2018, Chen and Liu [1] presented wide neural networks (also known as broad neural networks) as an alternative to deep neural networks (DNN) that can be easier and faster trained. These networks had a structure of a single layer feedforward neural network with a large number of neurons in the hidden layer. They were characterized with good performance and much higher speed of training than deep neural networks. Wide neural networks are flat, so there is no problem with exploding or vanishing gradient descent (an advantage in comparison to DNN).

Moreover, in flat neural networks, real features can be created as a group of combined weights and neurons influencing the results in the output layer. The question we try to answer in this paper is about the quality of features represented in wide neural networks. Can we say that one set of features is better than another one, and how to measure the feature quality?

Properties of a single-layer feedforward neural network (nonlinear perceptron) as a general function approximator were investigated since 1980ies [2], [3]. A nonlinear perceptron uses a single hidden layer with a nonlinear transformation of the combination of the input signals in each hidden neuron to approximate an unknown function f(x). Each output of nonlinear perceptron combines outputs from the hidden layer neurons to produce an approximation function

$$f(x) = \sum_{k=1}^{n} a_k \cdot g(w_k x + b_k) \quad (1)$$

where n is the number of hidden neurons, $w_k x$ is the inner product of a weight vector $w_k \in R^d$, and the input signal vector $x \in R^d$, and a_k, b_k are scalars that can be obtained during the network training. Nonlinear transformation function $g(.)$ is a prespecified logistic function (e.g. sigmoidal or radial).

Barron [4] has proved that the approximation error (a distance between a function to be approximated and the best approximation in a given class of approximating functions) for the nonlinear perceptron tends to zero with the integrated squared error rate of the order $O(1/n)$ where n is the number of neurons in the hidden layer. Leshno et al. [5] generalized this result proving that a nonlinear perceptron and other similar structures can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial.

Practical results were developed by Igelnik and Pao [6] who introduced a Random Vector Functional Link Neural Network (RVFLNN) that was shown to be a universal approximator for continuous functions with the approximation error converging to zero at the rate of $O(1/\sqrt{n})$. RVFLNN used the neural network with one hidden layer that implemented the approximating function $f_n(x)$. Like in the earlier works, this function was shown to approximate any function $f \in C(I^d)$ which is continuous on the standard hypercube $I^d = [0;1]^d \subset R^d$. From a practical point of view, it is important that the weights w_k and coefficients b_k were randomly selected from a given distribution and parameters a_k were learned using simple quadratic optimization.

Recently Chen and Liu introduced broad learning systems (BLS) as an alternative structure to deep neural networks [1]. The BLS was based on the idea of RVFLNN presented by Igelnik and Pao in [6]. Chen and Liu pointed out that deep neural networks require a very time-consuming training process due to a large number of weights in filters, layers, and gradient-based weight adjustment with many training epochs. The BLS is a flat network in which the input signals are amended by "enhancement nodes" to obtain a wide neural network structure. The incremental learning was used to add feature nodes as well as enhancement nodes. Incremental learning uses pseudoinverse and allows for incremental increases of neurons as well as training data without a need to repeat training for all data. They verified their approach on Modified National Institute of Standards and Technology data (MNIST) for classification of handwritten characters and NYU object recognition data

(NORB) for 3D object recognition. Weights of the enhancement nodes can either be randomly selected or trained. Chen and Liu used a sparse autoencoder approach to fine-tune initially random weights in order to obtain better features. They compared their test results with these of many classification methods including autoencoders [7],[8],[9], multilayer perceptrons [10],[11], extreme learning machines [12],[13], fuzzy restricted Boltzmann machines [14], and deep neural networks [15][16]. The BLS network obtained the accuracy of the results comparable to these reference methods but had the training time for the MNIST database from 3.6 to 7.3 times shorter than in extreme learning and fuzzy restricted Boltzmann machines and from 276 to 1543 times shorter than for autoencoders, multilayer perceptrons, and deep neural networks. Similar results were obtained for the NORB data, indicating that the method is much faster than the reference while maintaining their accuracy of classification.

The importance of studying wide neural networks was recently emphasized in [17], where authors pointed out the need for a comprehensive understanding of the tradeoff between depth and width in neural networks. They provided a proof that a deep fully-connected ReLU NN with the width less or equal to (n+4) can approximate any Lebesgue integrable function in n-dimensional R space with arbitrary precision. They pointed out the importance of proving lower and upper bounds to understand the tradeoff between width and depth of neural networks. Earlier works [18], [19] pointed out the existence of deep convolutional networks that cannot be realized by a wide shallow network if its width is less than the exponential bound of the depth of the convolutional network. No such equivalent result was established for wide networks.

There are also other concepts of neural network structures (e.g. Fully Connected Cascade Networks (FCCN) or Cascade Correlation [20], [21]) that are neither wide nor deep. These networks freeze the already learned network structure before adding a new neuron that is connected not only to all inputs but also to all neurons of all previous layers. In this concept, each hidden layer consists of only a single neuron, so it is similar to the concept of BLS. We refer to such networks comparing their performance to the performance of wide networks.

In this work, we examine in detail the convergence of RVFLNN approximation on the MNIST database [22]. In particular, we show that the convergence is slower than the claimed rate of $O(1/\sqrt{n})$. It is essential to know this convergence rate since we can introduce and test different than random features for the better network organization and performance. On this ground, we introduced and tested the concept of feature significance. The introduced feature significance can be a tool that guides the development of wide neural networks. It is also possible that this kind of estimate of the future quality can help in designing better neural networks in general (including deep neural networks).

II. Wide Neural Network Organization

The simplest form of the wide network organization based on a single layer feedforward neural network is shown in Fig. 1. In this figure, Z_1 is a $k \times m$ matrix of the input signals W_1 and W_2 that are weight matrices, $Y_1 = Z_1 * W_1$ is an input matrix to the hidden layer, Z_2 is determined through the sigmoidal transformation function of neurons in the hidden layer, so e.g. $Z_2 = tanh(Y_1)$ or using another logistic function $Z_2 = \psi(Y_1)$, and the neural network output is $Y_2 = Z_2 * W_2$.

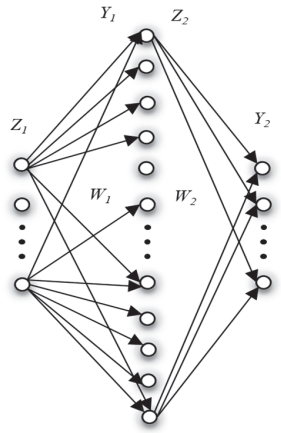

Fig. 1 Single layer feedforward neural network

In RVFLNN networks, W_1 is randomly generated, and W_2 is learned by using pseudoinverse

$$W_2 = pinv(Z_2) \cdot Y_D \tag{2}$$

where Y_D is a desired value of the output signal Y_2, so Y_2 obtained from the output of the neural network is a $k \times o$ matrix. If the hidden layer has n neurons then W_1 is an $m \times n$ matrix, and W_2 is an $n \times o$ matrix. Thus, according to the theory of RVFLNN networks, any continuous output function (desired values) can be approximated, and the approximation error is converging to zero at the rate of $O(1/\sqrt{n})$. If using these networks for classification problems, the desired output is a class indicator, and if each output neuron represents a single class, we can code this fact by assigning the value 1 if the input sample is from a given class and 0 otherwise. In such a case, the output is not a continuous function, and we would like to test what is the convergence rate in this case.

Consider four matrices $Z_1^{k\times m}$, $W_1^{m\times n}$, $W_2^{n\times o}$, and $Y_2^{k\times o}$ representing signals and weights from Fig. 1. We say that the input matrix Z_1 consists of k examples of m input features each. Thus, the network has m input features, n neural features and o output features. Our study is devoted to neural features. Thus, when we use the term "feature" we mean a neural feature.

A. Test of the Wide Neural Network with Random Weights

In order to determine the error convergence rate, we tested many RVFLNN neural networks on the MNIST data [22] varying the number of neurons in the hidden layer. The MNIST dataset consists of a number of handwritten digits collected from Census Bureau employees and 500 high-school students. Every digit is stored as a gray-scaled image with the size of 28×28 pixels. The digits have been normalized and centered in the image plane. The whole data set was divided into a training set containing 60 000 images and a test set of 10 000 images.

In our test, each input image was transformed into a vector of 784 pixel values scaled from -1 to 1, and desired outputs were

written as 10 element vectors representing digits from 0 to 9. The k-th element of the output vector had value 1 if the training data represented digit k-1 and was -1 otherwise. Thus the input Z_1 was a 60000×784 training matrix, and the desired output Y_D was a 60000×10 matrix with elements equal to 1 and -1. The $784 \times n$ weight matrix W_1 had randomly generated weights of uniform distribution from -1 to 1. $Y_1 = Z_1 \cdot W_1$ was computed from the input data as a $60000 \times n$ matrix. The logistic function used was $\psi(Y_1) = tanh(Y_1)$, therefore $Z_2 = \tanh(Y_1)$. Also, the $n \times 10$ weight matrix W_2 was calculated from $W_2 = pinv(Z_2) \cdot Y_D$.

Table I shows the mean values of the testing error rates and their standard deviations for the neural network sizes changing from 2 to 6000 neurons in the hidden layer. The results were obtained by averaging 20 runs with randomly generated weight matrices W_1.

TABLE I. TESTING ERROR OF WIDE NEURAL NETWORKS AS A FUNCTION OF THE NUMBER OF FEATURES (HIDDEN NODES).

Number of features n	4	8	16	32	64	128	256
Testing error rate in %	78.02	69.90	59.26	49.93	36.86	23.67	16.19
Standard deviation	3.44	3.28	3.85	2.73	2.02	0.87	0.44
Number of features n	500	1000	2000	3000	4000	5000	6000
Testing error rate in %	12.54	8.55	5.71	4.35	3.47	2.88	2.54
Standard deviation	0.23	0.15	0.14	0.11	0.09	0.08	0.05

When the random weights were generated in the range -0.1 to 0.1, the results of testing with random weights were better. These results show that weight normalization is important for better performance.

TABLE II. TESTING ERROR AS A FUNCTION OF THE NUMBER OF FEATURES FOR RANDOM WEIGHTS FROM -0.1 TO 0.1 INTERVAL.

Number of features n	4	8	16	32	64	128	256
Testing error rate in %	72.23	61.87	48.38	35.09	23.73	16.59	11.95
Standard deviation	2.31	2.75	1.74	1.69	0.56	0.40	0.36
Number of features n	500	1000	2000	3000	4000	5000	6000
Testing error rate in %	8.53	5.69	3.66	2.70	2.14	1.74	1.38
Standard deviation	0.20	0.07	0.05	0.04	0.04	0.03	0.02

Results were only slightly different when the random weights were generated in the interval from -0.01 to 0.01.

What is unexpected in these results is that even a single layer of the feedforward neural network with only two hidden nodes gives better than chance recognition (90% error). It contains only 20 trained weights. In addition, a linear perceptron with all the weights trained by pseudoinverse gives 14.22% error, so the network with 250 hidden neurons which gives better accuracy (error 12.1%) is easier to train since it computes pseudoinverse of the 60000×250 matrix rather than the 60000×784 matrix required for the linear perceptron.

All the tests were performed on a personal computer Intel-i5-7400, 3.0 GHz CPU, with 8 GB memory using MATLAB.

We can approximate the classification error obtained in Table II by the following function:

$$f(n) = \frac{2.46}{\sqrt{n+7}} \tag{3}$$

The constants 2.46 and 7 were obtained experimentally to match the error level for 2 hidden nodes and get a good fit to results obtained in the wide neural networks with the increasing number of hidden nodes in all random features experiments. This function is compared to the results obtained in the experiments with the random selection of weights, as shown in Fig. 2.

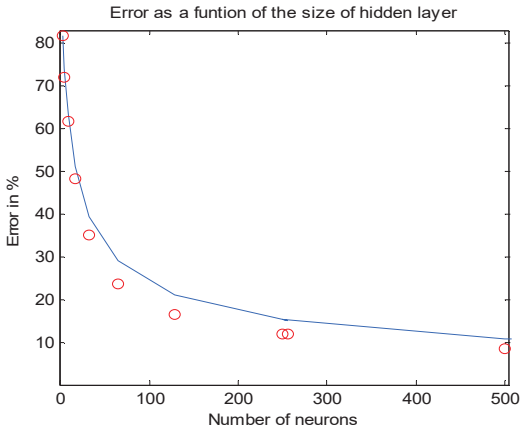

Fig. 2 Testing error as a function of the number of hidden nodes with randomly selected weights.

B. *Test of the Wide Neural Network with Radial Basis Functions*

Next, we tested MNIST data recognition using a new set of features based on radial basis functions (RBF) to determine the interconnection weights from the input data to hidden neurons. To obtain RBF for individual hidden neurons, we first generated their input weights as equal to the coordinates of the randomly selected training data. Each hidden neuron had a logistic function described by:

$$f(x) = e^{\frac{-d(x,w)^2}{2 \cdot \sigma^2}} \tag{4}$$

where x is the input data vector, w is the vector of weights of the hidden neuron, $d(x, w)$ is the distance between the input data and a hidden neuron, where weights are obtained from

$$d(x, w) = max_{+}((\|x - w\| - \bar{d} + 2 \cdot \sigma), 0) \tag{5}$$

and $\bar{d}$ is the mean value of norms of differences between weights of hidden neurons (or coordinates of selected training data vectors) for all pairs of hidden neurons

$$\bar{d} = \sum_{i,k=1,i\neq k}^{n} \frac{\|w_i - w_k\|}{n\cdot(n-1)} \quad (6)$$

and σ is the standard deviation of such norms.

Notice that the values $\bar{d}$ and σ are calculated only once after the random selection of training data points. During testing, each hidden neuron calculates its own logistic function value based on the distance of the input data x to its weights vector w.

To obtain statistics needed for the RBF function, we calculated the mean of norms of the differences between weights and their standard deviations for a various number of hidden neurons. The results were surprisingly stable across these numbers, as we can see in Table III.

TABLE III. DIFFERENCE BETWEEN WEIGHTS OF HIDDEN NEURONS USED IN THE SELECTION OF FEATURES BASED ON THE RADIAL BASIS FUNCTION.

Number of features n	4	8	16	32	64	128	256	512	1024
Mean norm of difference	20	19.3	20.3	20.8	20.5	20.4	20.1	20.3	20.2

We used the average mean for all dimensions equal to $\bar{d} = 20.2$ and the standard deviation $\sigma = 3.74$ to design radial basis functions for hidden neurons (equations (4) and (5)). When the radial basis functions were used in structuring wide neural networks, the performance increased (classification errors were reduced), as shown in Table IV.

TABLE IV. CLASSIFICATION ERRORS FOR WIDE NETWORKS WITH RADIAL BASIS FUNCTIONS.

Number of features n	4	8	16	32	64	128	256
Testing error rate in %	70.03	54.43	41.52	26.39	17.62	11.98	7.52
Standard deviation	3.82	2.72	2.69	1.53	0.61	0.40	0.20
Number of features n	500	1000	2000	3000	4000	5000	6000
Testing error rate in %	5.44	3.85	2.64	2.03	1.69	1.36	1.15
Standard deviation	0.10	0.05	0.04	0.04	0.03	0.02	0.02

These results are partially illustrated in Fig. 3 by green * points. As we can see, results are more accurate than the results for a random selection of features. Thus, this is a better selection of features for wide neural networks.

B. *Test of the FCC Deep Neural Network with Random Weights*

Wide neural networks are characterized by fast learning and an easy to design architecture. Another approach which promised similar advantages is a fully connected cascade. We conducted a speed-accuracy test on a Fully Connected Cascade (FCC) to discover if the computation time investment into more layers pays off. We did not expect an overall accuracy increase of the models, compared to the wide networks, due to the Universal Approximation Theorem [3].

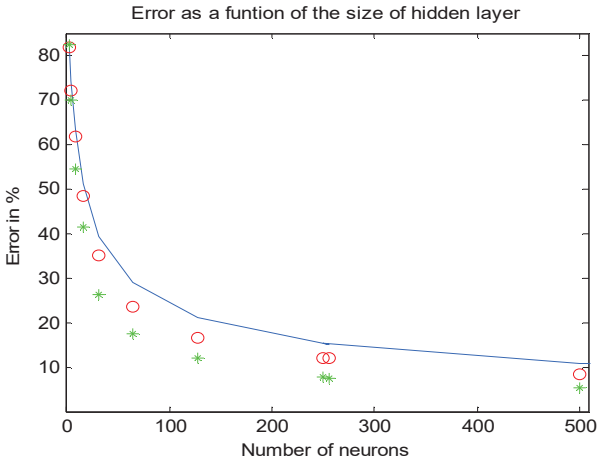

Fig. 3 Comparison of error functions for the random and RBF selections

We checked how the test accuracy converges with the increase in the number of FCC layers. We used an FCC network structure shown in Fig. 4.

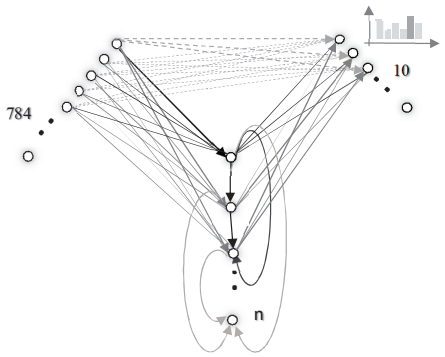

Fig. 4. FCC trained on MNIST

In Fig. 4, each hidden neuron is connected to all input nodes, all hidden neurons of the higher number, and all outputs. The number of hidden neurons is gradually increasing, effectively changing the number of layers (network size) and network complexity.

The networks of various sizes were trained with the Levenberg-Marquardt (LM) nonlinear optimizer, aimed to minimize the mean squared output error (MSE). In order to make the test results of FCC network in agreement with the MSE error, we scaled the output vector 10 times. This improved the recognition stability of the test results, and classification results are shown in Table V.

TABLE V. CLASSIFICATION ERRORS AND TRAINING TIMES OF THE FCC NETWORK

Number of layers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Number of weights in thousands	7.9	8.6	9.4	10.1	10.8	11.6	12.3	13.0	13.8	14.5
Test error rate in %	8.0	8.5	8.6	8.4	8.2	7.9	8.0	16.9	8.9	17.8
MSE in %	23	23	23	23	23	23	23	24	23	24
Training time [h]	28	28	30	24	28	59	50	51	72	92
Epochs	18	28	30	24	28	27	19	23	21	33

LM is the second-order training method. In practice, the second-order derivatives matrix inversion should be computed every time the weight will change. As we can see, the training of this network is very slow. The fastest training time was 24h. Thus, we conclude that the LM method is not suitable for practical problems of this size. Moreover, the FCC was not designed to be deep in the sense of today standards.

A Modified Cascade (MC) network (without connections marked by the dotted line in Fig. 4) is notably more compact, thus faster. Table VI presents the results of this modification. An error converges to ~ 9% in deeper networks with a standard deviation of 1%. Comparing to results presented in Table V, training time (here in minutes) is significantly smaller.

TABLE VI. CLASSIFICATION ERRORS AND TRAINING TIMES OF THE MC NETWORK

Number of layers	3	4	5	6	7	8	9	10	11	12
Number of weights in thousands	2.2	2.9	3.7	4.4	5.1	5.9	6.6	7.7	8.1	8.8
Error rate in %	37	21	16	12	12	10	9	11	11	8
MSE in %	23	23	23	23	23	23	23	24	23	24
Training time [min]	3	5	14	28	36	45	57	65	93	96

Overall, we did not find the FCC network to be competitive with wide networks of a similar level of complexity.

First, FCC network has a delay proportional to the number of cascaded nodes. Second, the number of weights used is much larger to reach the same test accuracy, which increases the training time.

III. FEATURE SIGNIFICANCE

As we could observe using RBF features, the recognition error was systematically lower than in the case when hidden neurons had assigned random weights. The question is, how significant is this improvement?

To be able to answer this question in quantitative terms, we introduce here a feature significance measure. The random selection of weights is used as a reference. Any feature selection method that gives better results than the random selection in lowering the recognition error should have the significance higher than 0, and those that give worse result should have the significance lower than 0. One way of measuring the feature significance will be to use the ratio of errors for the two methods compared (for a specific number of the hidden nodes).

Let us define the feature significance using

$$S_f = \frac{e_1}{e_2} - 1 \tag{7}$$

where e_1 is the classification error level obtained by the reference network, e_2 is the error level obtained by the second method with the same number of hidden neurons.

For instance, the significance of the random weights selection vs. theoretical limit $\frac{1}{\sqrt{n}}$ [6] is shown in Fig. 5. As we can see, the significance increases for larger networks, which tells us that the classification error with a random selection of weights in the hidden layer decreases faster than $\frac{1}{\sqrt{n}}$.

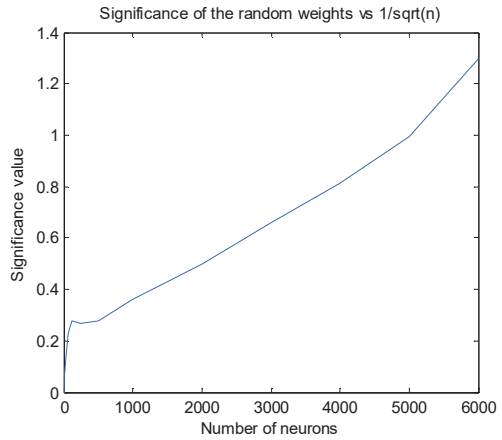

Fig. 5 Significance of randomly selected weights vs. theoretically established limits of error convergence.

Using significance, we can also compare one method to another. For instance, if e_1 represents the errors obtained in the random selection of weights in the hidden layer and e_2 weights of neurons in the RBF approach, then the significance plot is as illustrated in Fig. 6.

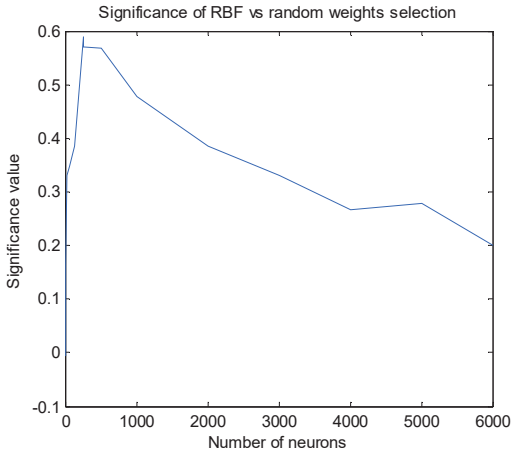

Fig. 6 Significance of RBF functions vs. randomly selected weights.

The result confirms that all network sizes using the RBF are better than the networks that use randomly selected weights.

This happens despite the error reduction is defined as a function of the number of features in both these methods.

Defining the significance of feature selection in a wide neural network using (7) is a quick and easy check if one method is better than another. The better method will have a systematically positive value of the significance measure and the bigger the significance measure, the better is the feature selection method.

Another approach to defining the feature selection significance is to relate it to the number of nodes in the hidden layer of the worse method needed to obtain a similar accuracy of classification results as obtained by the better method. We can do it using theoretical limits of the approximation error as a function of the number of hidden nodes expressed as $O(1/\sqrt{n})$. Since $e \sim O(1/\sqrt{n})$, we can relate the number of nodes to the observed error level as $n \sim O(1/e)$ and define the feature selection significance as

$$S_f = \sqrt{\left(\frac{e_1}{e_2}\right)^2 - 1} \sim \sqrt{\frac{n_2}{n_1} - 1} \tag{8}$$

With this definition, at a larger number of hidden neurons in wide neural networks, the lesser method should increase the number of neurons in the hidden layer $\frac{n_2 - n_1}{n_1} = (S_f)^2$ times to match the better method.

Using this definition of feature significance, comparing the errors in the randomly generated weights of hidden neurons to those generated by RBF hidden neurons, we obtain the results shown in Table VII.

TABLE VII. % INCREASE IN THE NUMBER OF HIDDEN NEURONS IN RANDOM SELECTION IN COMPARISON TO THE RBF SELECTION.

Number of features n	4	8	16	32	64	128	256
Required increase in	6.4	29.2	35.8	76.8	81.4	91.8	153
Number of features n	500	1000	2000	3000	4000	5000	6000
Required increase in	146	118	92.2	76.9	60.3	63.7	44.0

We can see that in the random selection of weights, the number of neurons needed to obtain an error that matches the one of RBF should be increased between 6-150%, which indicates large savings in the neural hardware and simulation time if RBF features are used to build wide neural networks.

IV. INCREMENTAL FEATURE SELECTION SIGNIFICANCE

The question we asked next is this. Suppose that we already have a number of features n in a wide network and would like to add a specific number n_f of new features that we believe are good features for a given problem. Thus, we wish to have an incremental significance measure that evaluates how many randomly generated features can be saved if the new features are used. The measure should tell us how much better is to use these new features than those randomly generated. The savings are specifically addressed to a subset of all features instead of replacing all features as we had in the previous case.

We can extend the definition of feature significance (8) to include the incremental increase in the number of features used in wide neural networks as follows

$$S_{f1} = \sqrt{\frac{\frac{e_1^2}{e_2^2} - 1}{\frac{e_3^2}{e_1^2} - 1}} \tag{9}$$

This definition is based on the rate of the change of errors if we use a selected group of new features instead of the same number of original features. In (9), e_1 is the error level produced by the network using only the original set of n features, e_2 is the error level produced by the network in which n_f features were replaced by the new set of features, and e_3 is the level of estimated error for the original features with n_f features removed. For simplicity of discussion, let us assume that the original features were randomly generated and the error level can be estimated from

$$e_1 = O\left(\frac{C}{\sqrt{n}}\right) \tag{10}$$

We can estimate the number of original (random) features needed to get the testing error e_2 from

$$e_2 = O\left(\frac{C}{\sqrt{n_2}}\right) \tag{11}$$

and error e_3 can be estimated from

$$e_3 = O\left(\frac{C}{\sqrt{n - n_f}}\right) \tag{12}$$

Using these estimates, we can compute incremental feature significance S_f using

$$S_{f1} = \sqrt{\frac{\frac{e_1^2}{e_2^2} - 1}{\frac{e_3^2}{e_1^2} - 1}} = \sqrt{\frac{\frac{e_1^2}{e_2^2} - 1}{\frac{n}{n - n_f} - 1}} = \sqrt{\frac{\left(\frac{e_1^2}{e_2^2} - 1\right)(n - n_f)}{n - (n - n_f)}} = \sqrt{\left(\frac{e_1^2}{e_2^2} - 1\right)\left(\frac{n}{n_f} - 1\right)} \tag{13}$$

A. Example

Let us illustrate the incremental feature selection significance with a simple example shown in Fig. 5. Let us assume that the error rate was $e_1 = 0.26$ for n=15 and when two random features were replaced by selected features (i.e. $n_f = 2$) we got the error rate $e_2 = 0.16$. Using (3), we can calculate the incremental significance of these features as

$$S_{f1} = \sqrt{\left(\frac{e_1^2}{e_2^2} - 1\right)\left(\frac{n}{n_f} - 1\right)} = 2.01 \tag{14}$$

In Fig. 7, the continuous line represents the theoretical error level estimated by $O(1/\sqrt{n})$, on the vertical line, we have errors e_1 and e_2, the top horizontal line intersects the curve $O(1/\sqrt{n})$ yielding the estimated error e_3, and the bottom horizontal line intersects $O(1/\sqrt{n})$ giving an estimate of the number of random features n_2 that would yield the same level of the error level e_2.

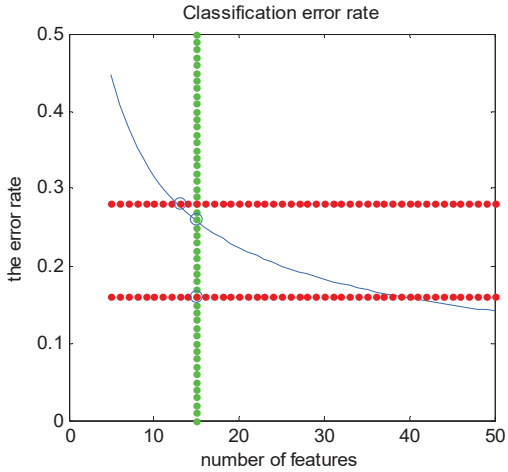

Fig. 7 Theoretical example to illustrate the effect of incremental addition of selected features over the set of random features.

Using (10) and (11), we can estimate this number as

$$n_2 = \frac{e_1^2}{e_2^2} \cdot n = 39.61 \quad (15)$$

Thus, it would take more than twice the current number of random features (around 15) to reach the same error level (around 0.16). This theoretical result can be confirmed by inspecting Fig. 7.

B. Incremental Significance of Selected Features

To further explore the effect of incremental addition of features different than the existing ones, we designed a set of selected features (which we believed will be useful for handwritten digit recognition) by observing that in the handwritten digits, important information is included in the endpoints of each digit. Both the number of endpoints and their location are important.

We designed a simple procedure that uses skeletonization to find the location of the endpoints in all data images. An example is shown in Fig. 8 for the handwritten digit 3. The location of the endpoints and their number is recorded and used as selected features.

Fig. 8. Original digit image, its skeleton, and the location of its endpoints

All endpoints are represented by their x and y coordinates on the scale from -1 to 1. This location information corresponds to the activation level of a hidden neuron (one hidden neuron per x or y coordinate of the endpoint). We limited the number of endpoints for each digit to be at most 5. Some digits had no endpoints at all, but some extra endpoints were created due to sloppy writing or as a result of the skeletonization algorithm, as illustrated in Fig. 9, where digit 6, that typically has one endpoint, here, had two of them.

Fig. 9 Spurious endpoint in digit 6.

The problem with the representation of the endpoints through their coordinates is that when the endpoint is near the center of the image then the value of its (x, y) coordinates is close to (0, 0), and effectively such endpoints do not influence the neural network response (since the corresponding hidden neurons are not activated). A better approach is to have each location counts. We can accomplish this by assigning two real values to x and y using the nonlinear transformation of each x and y coordinates. First, we normalize x and y within the [0 1] interval and then calculate a vector function

$$f(x) = \begin{bmatrix} cos(x \cdot \pi) \\ sin(x \cdot \pi) \cdot 2 - 1 \end{bmatrix} \quad (16)$$

Using this transformation, each coordinate value from [0, 1] interval will be transformed into a pair of values between -1 and 1. This transformation will activate at least one of the hidden neurons describing where the end-points are. With this improved representation, each endpoint requires 4 hidden neurons. Thus for 5 endpoints, we add 20 hidden neurons.

The question is how significant are these extra features represented by the coordinates of the endpoints. To answer such a question, we tested wide neural networks with various numbers of hidden neurons. This resulted in two types of errors, error e_1 which was obtained when all features of the hidden neurons were randomly generated, and error e_2 obtained in the networks in which 20 random features were replaced by 20 features representing the endpoints. Next, we calculated the incremental significance measure S_{f1} using (13) and the number of saved neurons in the hidden layer if all features are based on the random selection from

$$\Delta n = \frac{e_1^2}{e_2^2} \cdot n - n \quad (17)$$

The results are shown in Table VIII.

TABLE VIII. THE INCREMENTAL SIGNIFICANCE OF THE ENDPOINT FEATURES AND THE NUMBER OF SAVED FEATURES COMPARED TO RANDOM FEATURE SELECTION.

Number of features	32	64	125	250	500	1000	2000	3000
incremental significance S_{f1}	0.8	1.7	2.9	3.97	5.2	7.2	8.7	10.9
number of futures saved	39	82	202	342	568	1058	1543	2374

As we can see, both the significance of the selected endpoint features as well as the number of random features needed to compensate for the lack of these features are increasing with the network size.

V. Conclusion

This paper discussed the significance of the feature selection for wide neural networks. Obtained experimentally, the testing accuracy confirmed claims that the testing error decreases with the increasing number of hidden neurons, although there are significant differences in the performance of different feature selection methods for the same number of hidden neurons. We compared recognition accuracy on the MNIST database using two approaches: 1) randomly selected weights from the input layer to hidden neurons, and 2) weights based on radial basis functions. Both methods gave wide neural network structures without expensive iterative learning that characterize deep neural networks. It requires only a pseudoinverse to train the output layer that classifies the input data. However, the selection of weights using radial basis functions required a smaller number of hidden neurons to obtain a similar level of recognition accuracy. We also compared wide networks to connected cascades and demonstrated that these techniques, although their simplicity in construction, are not competitive with wide networks.

To measure the quality of the applied feature selection method, we introduced two significance measures. The first measure compared two methods of feature selection applied to all hidden neurons. The second method allowed for incremental testing of selected features while leaving the remaining features without change. This second approach is particularly useful to test a smaller number of specifically selected features that may require special preprocessing of data to establish the features. In addition, we can translate the feature significance into the number of random features that would have to be added to the wide network in order to obtain a similar performance as the network with added (possible more elaborate) features.

In future work, we want to explore tradeoffs between the number of hidden neurons in wide neural networks vs. width and depth of deep neural networks to better understand tradeoffs between the two parameters of modern neural network structures.

Acknowledgment

This work was supported by the grant from the National Science Centre, Poland DEC-2016/21/B/ST7/02220 and AGH 16.16.120.773.

References

[1] C.L.P. Chen, and Z. Liu, "Broad Learning System: An Effective and Efficient Incremental Learning System Without the Need for Deep Architecture", IEEE Transactions on Neural Networks and Learning Systems, vol. 29 , Issue: 1 , Jan. 2018, pp. 10-24.

[2] K-I Funahashi, "On the approximate realization of continuous mappings by neural networks", Neural networks, vol. 2, pp.183-192, 1989.

[3] K. Hornik, M. Stichcombe, H. White, "Multilayer Feedforward Newtworks are Universal Approximators", Neural networks, vol. 2, pp. 359-366, 1989.

[4] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," IEEE Trans. Inform. Theory, vol. 39, pp. 930-945, 1993.

[5] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," Neural Networks, vol. 6, no. 6, pp. 861–867, 1993.

[6] B. Igelnik and Y-H. Pao, Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net", IEEE Trans. on Neural Networks, vol. 6 , Issue: 6 , Nov. 1995, pp. 1320-1329.

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Comput., vol. 18, pp. 0899–7667, May 2006.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006.

[9] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in Proc. 25th Int. Conf. Machine Learning (ICML), New York, NY, USA, 2008, pp. 1096–1103.

[10] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics). Secaucus, NJ, USA: Springer-Verlag, 2006.

[11] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS), vol. 1. 2009, p.3.

[12] E. Cambria et al., "Extreme learning machines [trends controversies]," IEEE Intelligent Syst., vol. 28, no. 6, pp. 30–59, Nov. 2013.

[13] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," IEEE Trans. Neural Networks Learn. Syst., vol. 27, no. 4, pp. 809–821, Apr. 2016.

[14] S. Feng and C. L. P. Chen, "A fuzzy restricted Boltzmann machine: Novel learning algorithms based on crisp possibilistic mean value of fuzzy numbers," IEEE Trans. Fuzzy Systems, vol. 26, issue 1, Feb. 2018, pp. 117-130.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25 (NIPS 2012), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. New York, NY, USA, 2012, pp. 1097–1105.

[16] J. Chen, K. Li, K. Bilal, X. Zhou, K. Li, P.S. Yu, "A Bi-layered Parallel Training Architecture for Large-Scale Convolutional Neural Networks, IEEE Trans. on Parallel and Distributed Systems, vol. 30, issue 5, May 2019, pp 965-976.

[17] Z. Lu, et. all. "The Expressive Power of Neural Networks: A View from the Width", 31st Conf. on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, arXiv:1709.02540.

[18] N. Cohen, O. Sharir, and A. Shashua. "On the expressive power of deep learning: A tensor analysis", Conf. on Learning Theory, pp. 698–728, 2016.

[19] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks", Conf. on Learning Theory, pp. 907–940, 2016.

[20] I.V. Tetko, A.E.P. Villa ,"An Enhancement of Generalization Ability in Cascade Correlation Algorithm by Avoidance of Overfitting Overtraining Problem", Neural Processing Letters August 1997, vol. 6, Issue 1–2, pp 43–50.

[21] C.L. Giles et all., "Constructive learning of recurrent neural networks: limitations of recurrent cascade correlation and a simple solution", IEEE Trans. on Neural Networks, vol.6 , Issue: 4 , Jul 1995.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.