Comparison of Neural Network and KNN classifiers, for recognizing hand-written digits

Iwo Różycki[a], Adam Wolszleger[a]

[a]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice

Abstract

This paper presents a comparison between two different types of classifiers, neural Network and KNN (k nearest neighbors) . The object is to decide which one better handles recognising numbers, from zero to ten, that have been handwritten. In this project we designed a program that analyses data from a database, and feeds this data to the classifiers, which in turn analyse the input from the user and make a prediction based on the information gained from the training data. The aim is to see whether there are any significant differences when it comes to the accuracy and speed of the learning processes. The comparison will be made based on the accuracy of the predictions with different configurations and parameters. Both these classifiers work in a different way and can be used efficiently for different purposes and this paper will present data that will help decide whether either of these is better suited in this particular area.

Keywords

Hand-written digits, Handwritten signature verification, Handwritten character recognition, Neural networks

1. Introduction

Artificial neural networks are computing systems, which are based on the anatomy of the human brain. The brain is made up of billion of neurons, which are interconnected by quadrillions of connections, called synapses, which allow the information to flow between the neurons. Currently, because of the huge network of connections, the human brain exceeds any supercomputer in terms of processing power. Its structure can be used to create artificial networks, which use artificially created neurons and synapses, which are able to process information and make predictions [1, 2, 3] and guesses based on the knowledge gained [4, 5]. This process is known as Deep Learning and it is widely used today in voice recognition, driver less cars and can be found in devices we use every day, such as smart phones or TVs [6]. In the process of deep learning, a computer model learns classification, directly from data, in the form of images, sound or text. A well constructed model can even exceed human ability.

KNN, or k nearest neighbors is a much simple algorithm, that classifies new data based on a similarity measure, usually a distance function, to the already stored available cases [7]. The assumptions made by these algorithm ,is that similar things exist close to each other and it is most useful when that assumption is true.

The idea behind it, is that it can calculate straight-line distance, also known as Euclidean distance, between the input data and all that data that is available and which represents all the possible classes, that the input can be classified as. It therefore relies on this data set to be complete and would be unable to recognize objects beyond what is already known to it.

There are many aproahes to classify text, by ranking evaluation [8], convolutional features extraction [9] or devoted benchmark [10]. This project aims to compare these classifiers in the area of recognizing handwritten digits. This is an important area, as recognition of handwriting is used in a variety of modern devices. We will test different configurations of parameters for both neural networks and KNN and hopefully see significant differences in performances in terms of accuracy and speed.

1.1. Related works

These two classifiers have been compared in different articles and areas of research. In one of them, they have been used to monitor the conditions of machines, by analyzing the vibration of journal-bearings. In this research another technique, which is more commonly used in this area, was also used for comparison. At the end, the research has concluded that neural network performed better in this particular task. Another research, that will be mentioned is one where these models where compared at classifying magnetic resonance images (MRI). This is a very important area of research, as MRI is not always easy to interpret by a

SYSTEM 2020: Symposium for Young Scientists in Technology, Engineering and Mathematics, Online, May 20 2020

iworozy581@student.polsl.pl (I. Różycki);
adamwol165@student.polsl.pl (A. Wolszleger)

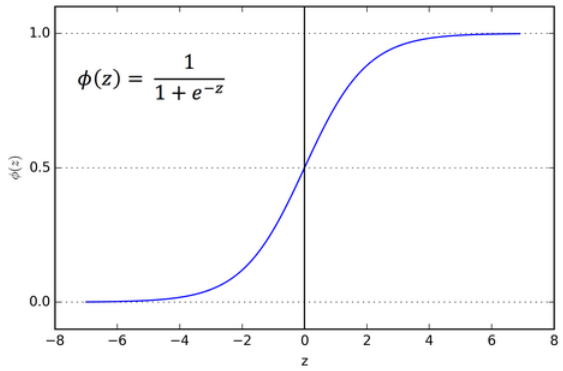

Figure 1: Sigmoid function

human, and using automated analysis would increase the probability of correct diagnosis. A special technique was used on the images, to extract most important features, which were then fed into a feed forward back propagation neural network and a KNN classifier. In this particular research KNN came out on top in terms of accuracy [11, 12, 13, 14, 15, 16]. As seen by these examples of research, both neural network and KNN can perform well in different conditions and only by close comparison can we find out which one performs better in a given area.

2. Mathematical model

For this project we needed various functions to modify the data as it passed between the neurons. The function used to sum the inputs from the previous layer of neurons looked as follows:

$$Input = X_1 \cdot W_1 + X_2 \cdot W_2 + X_3 \cdot W_3 + ... + X_n \cdot W_n \quad (1)$$

The 'X' represents the output from a given neuron and 'W' is the weight attached to the synapse. After the input has been weighted and summed, it is modified by the activation function. These functions act to modify the input, so that the output from the model can be interpreted and used as needed. The function, that has been used in this project is the sigmoid function and it shown on the following graph. The graph shows the function that is used to model the data:

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (2)$$

The derivative, which is used later for back propagation,is as follows:

$$f'(x) = \frac{1}{1 + \exp(-x)} \cdot (1 - \frac{1}{1 + \exp(-x)}) \quad (3)$$

As seen on the graph the function only exists between 0 and 1 and it is used in models, that predict probability as their output. This makes it ideal for this project, as the aim is to produce a probability with which the input resembles one of the digits. For the back propagation algorithm we need to calculate the error of the neurons to correctly modify the weights of the synapses. For the last layer of neurons the following equation was used:

$$\Delta W = LR \cdot (expected - output) \cdot f'(x) \cdot s_j \quad (4)$$

Here $\Delta W'$ represents the change in weight, 'LR' is the learning rate, which is chosen for the network, $f'(x)$ is the derivative of the activation function and s'_j is the input from the j-th neuron from the previous layer. We have defined a gradient of error for each neuron to help us with the above equation. For the last layer it is calculated as follows:

$$grad = f'(x) \cdot (expected - output) \quad (5)$$

For the other layers:

$$grad = f'(x) \cdot \sum_{i=1}^{n} w_i \cdot d_i \quad (6)$$

Here the sum of the weights is multiplied by the error gradient from i-th neuron, that is connected to the current neuron. Based on the equations above we can simplify the $\}\Delta W'$ to the following:

$$\Delta W = LR \cdot d \cdot s \quad (7)$$

For the KNN we have used the Euklidean distance as the measure of similarity. For this we have used the following equation:

$$D(X, Y) = \sqrt{(X_1 - Y_1)^2 + ... + (X_n - Y_n)^2} \quad (8)$$

Here 'X' represents the known data and 'Y' represents the input. Based on the distances calculated the algorithm, then calculates which class of objects appears the most in the 'k' closest neighbors. The value of 'k' was decided after experimenting on different values.

3. Description of the proposed system

The deep learning process of the neural network is based on the idea of data being fed into the neurons and modified, as it passes further along the different

layers. In order to achieve this we have to create a model, that uses various functions to modify the data appropriately. We start with the building block of the neural network, which is a neuron. The neuron receives information via synapses from the neurons that are connected to it.

Each synapse has a weight attached to it, which is initially generated randomly, when the synapse is first created. The data, that comes into a neuron is weighted using the previously described equation. This means that the input depends on both the output of the previous neuron, as well as the weight of the synapse that connects them. When all the inputs have been summed, the data is then modified by the activation function of the neuron. The function used in this project in the sigmoid function, for the reasons described previously. The neurons within the network are organized through layers. The two layers,the first and the last, are present in all neural networks. The middle layers, known as hidden layers, are optional, but can be added to improve the accuracy of the learning process of the network. The optimal number of the hidden layers and the configuration of neurons within them was decided on through testing of the network. For the input data we have used bitmaps, sized 28x28 bits, that represent one the ten digits. For that we needed enough neurons in the first layer for each individual bit of the picture. The number of neurons in the last layer represents the total number of classes that the input can be classified as, which in this project was ten. The experiments we conducted to choose the number of hidden layers will be described later in the article. To modify the weights of the synapses we have used the back propagation algorithm, described in the previous section.

The KNN classifier was much simpler to set up. In this algorithm the objects are classified based on their closeness to data, that the algorithm was trained on. When an input is presented, the Euclidean distance between it and all the available options is calculated. The distances are then sorted and the smallest are at the top of the list. The value of 'k' determines the number of distances that are considered during the voting process, where the algorithm checks which class appears most frequently within these chosen distances. This class is then given as the prediction for the input. This means that only the value of 'k' has to be determined before setting up the classifier. This has been chosen based on tests described later.

The data used for this project has been taken from the MINST database. It has been sorted into a test and a training set. For this project we have trained both classifiers using the training set and then used the test

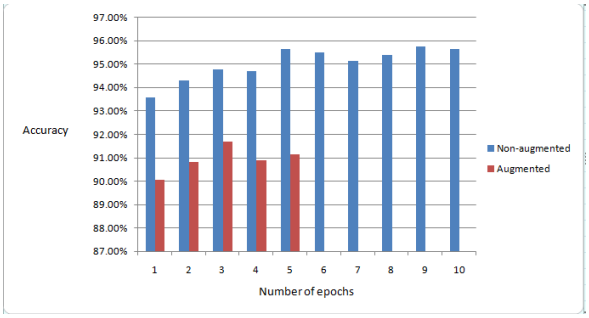

Figure 2: Epoch number test

set to determine their accuracy. This was calculated based on how many times the model chose the correct digit, based on the information gained from the training data. We have used this accuracy to compare the two models in different scenarios.

4. Experiments

For our experiments, we have decided to create another set of training data, by augmenting the one we already had. This was done by randomly modifying the pictures with one of four functions, which were rotation by 90 degrees left or right, rotation by 90 degrees horizontally(upside down) or adding 'noise', which was just changing some of the pixels to black. This was done to see, whether there would be a significant difference for the classifiers when dealing with random nature of the modifications of the data. The final data sets had 60000 images for non-augmented data and 120000 for the augmented data(this one had both the augmented and non-augmented).

For the first test, we deiced to test how the number of epochs would affect the accuracy of the neural network, when using augmented and non-augmented data. For this we have kept the learning rate at 0.3 and have added a singular hidden layer of 32 neurons. This setup was chosen to decrease the learning time and to allow us to complete further tests. We started the experiment at 1 epoch and increased it gradually until 10 epochs. For the augmented data we have only increased this number to 5, because of the increased size of data. The results are presented on the diagram below. From the diagram we can see that the accuracy varied between 93.5% and almost 96% for the non-augmented data.

The highest accuracy was reached at 9 epochs, and the biggest difference can be observed as we increased the number from 4 to 5. Above 5 epochs the accuracy

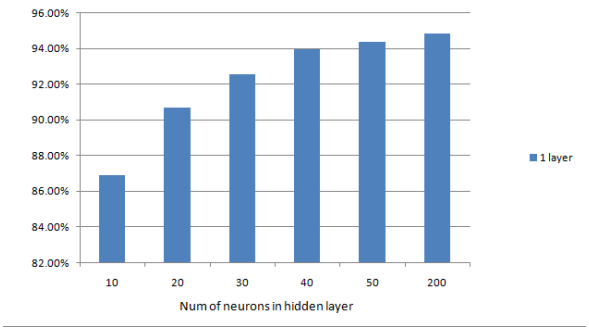

Figure 3: Hidden layers test 1

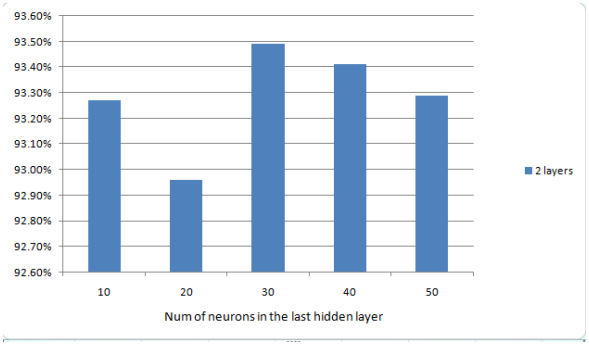

Figure 4: Hidden layer test 2

doesn't fall below 95% and it remains rather constant. The average accuracy for the non-augmented data is 95.05% with a standard deviation of 0.0071%. When it comes to the augmented data the accuracy is visibly decreased. Here is varied between 90% and 92%, so there was slightly lower variation, but this might be due to lower number of epochs tested. The biggest difference between the two sets can be seen at 5 epochs, where it reaches 4.52%. The best accuracy can be observed at 3 epochs at 91.68%, and it seems to decrease past this number. The average accuracy for augmented data was 90.92% with a standard deviation of 0.0059%. The average difference between the two sets at each number of epochs was calculated to be 3.69%. Based on these values, we can conclude that the neural network was better at recognizing data without augmentation, but even with random modifications it was able to correctly guess the digit over 90% of the time. Across both sets the average accuracy was calculated to be 93.67% with a standard deviation of 0.0212%. Another test was designed to check, how the number of hidden layers influenced the accuracy. In this test we kept the learning rate at 0.3 and set the number of epochs to one. We started with 10 neurons in the hidden layer. The results can be seen below. Here was can see, that as we increased the number of neurons the accuracy increased. Here it ranged between 86.5% and went up to almost 95% for 200 neurons. We did a test with 200 neurons to check whether a bigger increase in number led to larger increase in accuracy. The downside to increasing this number was higher learning times, although the increase was small.

The biggest increase can be seen between 10 and 20 neurons, where it reached 3.80%. Between 50 and 200 neurons the increase was only 0.45%. From this we can conclude, that the number of neurons in the hidden layer can be kept quite small and this won't adversely affect the accuracy and will lead to faster learning speeds. Even with a number as small as 10 we can see, that the accuracy is still 86.86%.

In the next test we have added another hidden layer. We have kept the same learning rate and number of epochs as in previous test and set added 50 neurons to the first hidden layer. The second one started with 10 neurons and we have increased this number gradually. The results can be seen below. Here we can see the results are less conclusive. The best accuracy of 93.49% was achieved at 30 neurons. This value is smaller, than what was achieved with one hidden layer with 40 neurons. The worst accuracy of 92.96% was achieved at 20 neurons and this value is higher, than the worst accuracy with one hidden layer. Adding another set of hidden layers leads to increased learning times, which has to be taken under consideration. From this it can be concluded, that adding another layer of neurons will not necessarily lead to higher accuracies, but might be a safer option. In our final setup we have decided to keep a single hidden layer, mostly due to decreased learning time.

The next test was designed to check how changing the value of 'k' would influence the accuracy of the KNN classifier. This is the only parameter, that we could modify, therefore it is the only test done for the KNN. Here we started with k=10 and increased the number. KNN took a very long time, when we used the whole of test set for accuracy calculation, so we decided to only use a 100 elements from the test set. The distances were calculated for the whole of the training set. Results of the test can be seen below. As can be seen on the graph, we have decided to increase the values drastically as gradual increase led to unsubstantial differences. From the graph we can see, that the accuracy stays the same for low values of 'k', namely below 30 and then decreases as this number goes up. This decrease is more rapid as we reach values of 1000 and more. KNN handled non-augmented data better, as

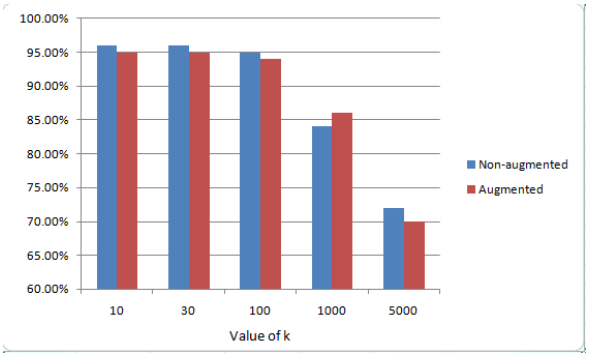

Figure 5: K value test

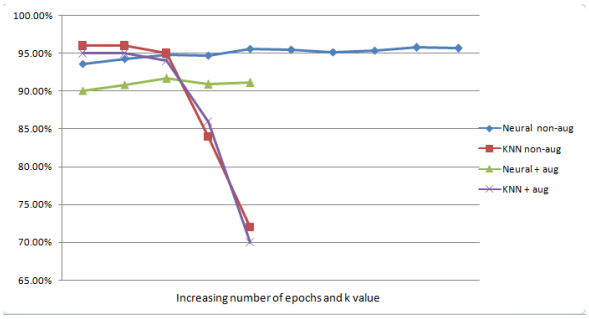

Figure 6: Combined results of previous tests

was the case with the neural network, but here the differences varied between 1-2%, and at k=1000 the accuracy for augmented data was higher at 86% compared to 84%. The highest observed accuracy was higher for KNN at 96.00% compared to 95.65% for the neural network. From this experiment we can conclude that lower values of 'k' lead to higher accuracy of the classification process. This makes sense as higher values of 'k' means higher distance values are taken into consideration during the voting process. For comparison we collected the data from the experiments described previously and put the together on a single diagram visible below. As can be seen on the graph, the accuracy for KNN, at low values of 'k', is higher when compared with the neural network. Increasing the number of epochs lead to higher accuracies, which cannot be said for increased values of k. The biggest differences can be observed with augmented data. KNN achieved higher accuracy when compared with neural network, with the highest difference being 5%. At 3 epochs this difference is lower. Based on this data we can say that KNN should, in theory be better at correctly predicting the answer. We have designed one last experiment to test this. We have drawn each digit 20 times and tested each classifier with and without the use of augmentation. In the table below we have recorded the most prevalent answer for each case. From the table we can see, that both classifiers were able to recognize digits 0,2,3,5 and 8 quite well, but digits 7 and 9 posed a difficulty. Neural network was able to correctly guess the digit 80% with both data sets. KNN classifier was able to guess the digit 70% of the time with both sets. The differences were not big enough be safely conclude which classifier was better. Preferentially this test would be carried out a 100 or more times, to give more meaningful results.

Digit drawn	Knn non-augmented k=30	Knn augmented k=30	Sieć augmented 3 epochs	Sieć non-augmented 9 epochs
0	0	0	0	0
1	1	1	1	7
2	2	2	2	2
3	3	3	3	3
4	1	1	4	4
5	5	5	5	5
6	6	6	6	8
7	1	1	3	7
8	8	8	8	8
9	3	3	2	9

Figure 7: Table of results of the drawing test

5. Conclusions

After carrying out the project and analyzing the data, it cannot be safely concluded, that one classifier would be better than the other one in the area of recognizing handwritten digits. Although the tests, that were concluded did show some differences in the accuracy, especially when the data was augmented, not enough tests were performed to show, that the differences between the classifiers were significant. What can be concluded from these tests is, that there is indeed a difference in how these perform. KNN seemed to better handle data, that was augmented, probably due to the fact that all the training data was available to it for comparison with the test data. Neural network on the other hand increased in accuracy as we increased the number of epochs, as this allowed the weights on the synapses to be changed, according to how many errors in judgment the network made. With the back propagation algorithm the network gets better each time we train it on the training data, where the KNN algorithm only needs to be given the training data once for it to work. Both classifiers show quite high accuracy, when dealing with our data, but this would preferably be increased to very close to a 100%, with correct configuration. More tests would have to be done, mostly to find the best possible setups for both these models

and to produce more statistically significant data. Finally in the testing phase we noticed that there is a slight improvement in accuracy if the images are prefiltered [17].

References

[1] M. Woźniak, D. Połap, Intelligent home systems for ubiquitous user support by using neural networks and rule-based approach, IEEE Transactions on Industrial Informatics 16 (2019) 2651–2658.

[2] C. Napoli, F. Bonanno, G. Capizzi, An hybrid neuro-wavelet approach for long-term prediction of solar wind, Proceedings of the International Astronomical Union 6 (2010) 153–155.

[3] C. Napoli, F. Bonanno, G. Capizzi, Exploiting solar wind time series correlation with magnetospheric response by using an hybrid neuro-wavelet approach, Proceedings of the International Astronomical Union 6 (2010) 156–158.

[4] G. Capizzi, G. L. Sciuto, C. Napoli, M. Woźniak, G. Susi, A spiking neural network-based long-term prediction system for biogas production, Neural Networks 129 (2020) 271–279.

[5] G. Cardarilli, L. Di Nunzio, R. Fazzolari, A. Nannarelli, M. Petricca, M. Re, Design space exploration based methodology for residue number system digital filters implementation, IEEE Transactions on Emerging Topics in Computing (2020).

[6] M. Woźniak, D. Połap, Soft trees with neural components as image-processing technique for archeological excavations, Personal and Ubiquitous Computing 24 (2020) 363–375.

[7] M. Husnain, S. Mumtaz, M. Coustaty, M. Luqman, J.-M. Ogier, S. Malik, Urdu handwritten text recognition: A survey, IET Image Processing (2020).

[8] N. D. Cilia, C. De Stefano, F. Fontanella, A. S. di Freca, A ranking-based feature selection approach for handwritten character recognition, Pattern Recognition Letters 121 (2019) 77–86.

[9] H.-h. Zhao, H. Liu, Multiple classifiers fusion and cnn feature extraction for handwritten digits recognition, Granular Computing (2019) 1–8.

[10] A. Islam, F. Rahman, A. S. A. Rabby, Sankhya: An unbiased benchmark for bangla handwritten digits recognition, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 4676–4683.

[11] A. Venckauskas, A. Karpavicius, R. Damaševičius, R. Marcinkevičius, J. Kapočiūte-Dzikiené, C. Napoli, Open class authorship attribution of lithuanian internet comments using one-class classifier, in: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2017, pp. 373–382.

[12] A. Mishra, K. Kumar, P. Kumar, P. Mittal, A novel approach for handwritten character recognition using k-nn classifier, in: Soft Computing: Theories and Applications, Springer, 2020, pp. 887–894.

[13] C. Napoli, E. Tramontana, G. L. Sciuto, M. Wozniak, R. Damaevicius, G. Borowik, Authorship semantical identification using holomorphic chebyshev projectors, in: 2015 Asia-Pacific Conference on Computer Aided System Engineering, IEEE, 2015, pp. 232–237.

[14] C. Napoli, G. Pappalardo, E. Tramontana, An agent-driven semantical identifier using radial basis neural networks and reinforcement learning, arXiv preprint arXiv:1409.8484 (2014).

[15] T. Jadhav, Handwritten signature verification using local binary pattern features and knn, International Research Journal of Engineering and Technology (IRJET) 6 (2019) 579–586.

[16] M. Wróbel, J. T. Starczewski, C. Napoli, Handwriting recognition with extraction of letter fragments, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2017, pp. 183–192.

[17] G. Capizzi, S. Coco, G. Sciuto, C. Napoli, A new iterative fir filter design approach using a gaussian approximation, IEEE Signal Processing Letters 25 (2018) 1615–1619.