Annales Mathematicae et Informaticae
Accepted manuscript
DOI: https://doi.org/10.33039/ami.2020.10.001
URL: https://ami.uni-eszterhazy.hu

Fuzzification of training data class membership binary values for neural network algorithms

Tibor Tajti

Eszterházy Károly University
tajti.tibor@uni-eszterhazy.hu

Submitted: August 16, 2020
Accepted: October 21, 2020
Published online: October 21, 2020

Abstract

We propose an algorithm improvement for classifying machine learning algorithms with the fuzzification of training data binary class membership values. This method can possibly be used to correct the training data output values during the training. The proposed modification can be used for algorithms running individual learners and also as an ensemble method for multiple learners for better performance. For this purpose, we define the single and the ensemble variants of the algorithm. Our experiment was done using convolutional neural network (CNN) classifiers for the base of our proposed method, however, these techniques might be used for other machine learning classifiers as well, which produce fuzzy output values. This fuzzification starts with using the original binary class membership values given in the dataset. During training these values are modified with the current knowledge of the machine learning algorithm.

Keywords: Machine learning, neural networks, fuzzification

MSC: 92B20, 03B70, 03B52

1. Introduction

The increasing performance of computers enables the wide use of artificial intelligence and machine learning technologies. These technologies come into our daily

lives, with image recognition, automatic translation, AI assistants, chatbots, autonomous cars, etc. One of the most widely used machine learning algorithms is the Artificial Neural Network and its Deep and Convolutional variants [8, 17]. Neural network algorithms are supervised machine learning algorithms, their major applications include classification, regression, pattern recognition, function approximation, intelligent control, learning from data. The neural network is basically a set of interconnected artificial neurons and the appropriate algorithms working on them [8].

A variation of the multi-layer perceptron model is the convolutional neural network. LeNet was one of the very first convolutional neural networks creating an area of deep learning. Yann LeCun's pioneering work has been named LeNet-5, after many successful iterations [11]. Convolutional networks have shown to be very effective e.g. in image classification [4, 5], natural language processing [6] and time series forecasting [3]. CNNs have a convolution operator, hence the name convolutional network. This convolution operator does feature extraction, e.g. when learning to classify a 2D image, smaller (e.g. 3×3 or 5×5 pixels) parts of the image will be processed as a sliding window over the whole image, so the network learns such smaller-scale features of the images. Committee machines and ensemble methods have been shown to improve the accuracy of neural networks and other machine learning algorithms.

One of the most widely used public datasets is the Modified National Institute of Standards and Technology database (MNIST) [12], which contains 60,000 handwritten numbers in the training set and 10,000 handwritten numbers in the test set. Different classifiers, like K-Nearest Neighbors, SVMs, Neural Nets, Convolutional Neural Nets, proved on this database had shown fail rate down to about 0.2% (20 failures from 10000 test samples) [12]. We have used this dataset for our research.

State-of-the-art architecture as of the time writing this paper is the squeeze-and-excitation network[1] [9].

Modification of training data is often useful for regularization. This can be done by e.g. making distortion, adding noise, using data augmentation [21] or adversarial training [19]. Changing the class membership values of the training data can be considered as one such method.

Usual classification is done providing binary class membership values in the training data, although even for the input patterns for which the classification could be considered uncertain. Fuzzy logic has advantages compared to binary logic having values between false and true as well [1, 2, 10, 15, 22]. Fuzzy logic can be used in machine learning as well, e.g. combining with neural network [7], even with ensemble methods [17]. Using fuzzy class membership values can have performance improvement and this method can also be considered to provide a kind of confidence, which can be an additional advantage in cases where the confidence of the outputs is also required [13].

[1] MNIST classifier with average 0.17% error, 25 February 2020, `https://github.com/Matuzas77/MNIST-0.17/blob/master/MNIST_final_solution.ipynb`

2. Improvements for neural network classifiers

We propose the fuzzification of training data output class membership values. This can be used with standalone learners and with multiple (ensemble) learners as well for better result.

One common problem is that training data usually has binary output values, even when the train samples may belong to more than one class at a certain fuzzy level. [7, 23] These data come usually labeled so that each sample has one or more labels, each of which means the crisp True membership in the class behind that label, and crisp False membership value for the other classes in the same category. There can be cases where these crisp class membership values can be considered misleading, so the correction of these values can lead to reducing the confounding effect of them.

The proposed fuzzification technique might be applied to other classifying algorithms as well, in case they are able to give fuzzy membership values in their output. Research on other algorithms in order to apply the fuzzification technique on them can be a future research, in the current research we conducted our measurements with convolutional neural network algorithms.

We define simple methods which can be used to modify the target output values given for train patterns during the training process to get fuzzy output values from the crisp (binary) values of the training data set. This class membership fuzzification is done so that the knowledge gained during the learning process will be used to correct the inaccurate or incorrect output class membership values of the train patterns. In the following, we will show and describe the proposed algorithm variants. The performance of these algorithm variations will be analysed and shown in Section 3.2.

Three versions of the algorithm will be presented below. The first of them (Algorithm1) is for single learners, the second version (Algorithm2) is for multiple learners the result of which can be used with committee machine voting functions, the third variant (Algorithm3) is a simple modification to handle the parameters of the fuzzification for multiple learners.

```
function FuzzyTraining(model, train_X, train_Y, a, b, c):
    epoch = 0
    fuzzy_Y = train_Y
    while epoch < MAX_EPOCHS and CheckEarlyStopCondition() == False:
      model.fit(train_X, fuzzy_Y, epochs=1)
      out = model.predict(train_X)
      if epoch > START\_FUZZY:
        fuzzy_Y =  a*fuzzy_Y + b*out + c*train_Y
    epoch = epoch + 1
```

Algorithm 1: Fuzzy Training

Algorithm1 must be called with the training data inputs and outputs, and the parameters for the fuzzification for the training of a learning model. The parameter a is the coefficient for the momentum which means the importance of the actual (current) class membership values, which are in the fuzzy_Y vector. This affects

the change from original values towards the desired values. The parameter b is the coefficient for the current knowledge (the out vector), that means the courage to change. The parameter c is the coefficient used for the train_Y vector, which means the importance of the original target output data. The sum of the parameters a, b and c must be 1.0 . Of course the condition when to start the correction must be also considered. In the algorithm presented above, it is a simple condition to have a number of epochs before the first correction. This of course can be changed to an adaptive condition to achieve better performance, however, for our measurement it is more important to know the number of correction operations. When the learning starts, the initial values in the fuzzy_Y vector are the same as given in the train_Y vector.

As it can be seen in Algorithm1, the defined algorithm can work with individual learner algorithm.

We give an extended variant of the algorithm as well to enable to use the combined knowledge of multiple (ensemble) learners. The usage of multiple learners of similar level usually gives better result compared to the individual learners, well-known methods are the committee machines and the ensemble methods [14, 16, 20]. In this version of the algorithm, all the learners will modify the same fuzzy_Y corrected output values, so their combined opinion will have an effect on the subsequent training epochs.

```
function FuzzyTrainingEnsemble(models, train_X, train_Y, a, b, c):
  epoch = 0
  fuzzy_Y = train_Y
  while epoch < MAX_EPOCHS and CheckEarlyStopCondition() == False:
    for model in models:
      model.fit(train_X, fuzzy_Y, epochs=1)
      out = model.predict(train_X)
      if epoch > START_FUZZY:
        fuzzy_Y = a*fuzzy_Y + b*out + c*train_Y
    epoch = epoch + 1
```

Algorithm 2: Fuzzy Training Ensemble

In the case of this new variant of the proposed algorithm (Algorithm2) the correction of the training data outputs will be better, because the combined knowledge of the learners has a better performance compared to the individual results. The correction will also be faster, because after every learning epoch of each individual learner a correction of the training data outputs will be done. In case of multiple learners, parameter a affects the change from original values towards the desired values and it affects the averaging effect on the outputs of multiple learners too. In a future development, it might be useful to change the algorithm with an additional parameter to separately control these two effects.

In this case, the number of times the correction statement will be run is the number of (epochs - START_FUZZY) multiplied by the number of the learners. This can be taken into account when setting the parameters for the training data output value fuzzification. We provide a modification to Algorithm2 with a simple normalization with respect to the number of learners.

Let M be the number of learners, a, b and c the weights for the train output class membership value fuzzification as described for the algorithm. We can calculate the normalized a', b' and c' weights as follows:

- $a' = \sqrt[M]{a}$
- $b' = \frac{(1-a')b}{b+c}$
- $c' = 1 - (a' - b')$

The parameters a', b', c' now correspond to the parameters a, b, c so that the speed of convergence with M learners giving the same output will be the same as the speed of convergence would be using the parameters a, b and c with one learner. Now we apply the above formulae to get the new version of this algorithm.

```
function FuzzyTrainingEnsemble2(models, train_X, train_Y, a, b, c):
  epoch = 0
  fuzzy_Y = train_Y
  a = power(a, 1/len(models))
  b = (1-a)*b/(b+c)
  c = 1-(a+b)
  while epoch < MAX_EPOCHS and CheckEarlyStopCondition() == False:
    for model in models:
      model.fit(train_X, fuzzy_Y, epochs=1)
      out = model.predict(train_X)
      if epoch > START_FUZZY:
        fuzzy_Y = a*fuzzy_Y + b*out + c*train_Y
    epoch = epoch + 1
```

Algorithm 3: Fuzzy Training Ensemble 2

In Algorithm3 the normalization of the parameters has to be done only once, before the training loop. Certainly it might be possible to adaptively change the parameters during the training process, the research of this can be conducted in the future. Note that we have overwritten the original values of the parameters a, b and c. If this is not the desired behavior then these values can be preserved. Since with given number of learners and given (not changing) a, b and c parameters the difference between the Algorithm2 and Algorithm3 variants lies only on changing the parameters, we have not conducted any separate measurements on Algorithm3.

3. Performance evaluation of fuzzification of training data binary class membership values

3.1. Performance evaluation framework

The experiments ran on personal computers equipped with NVIDIA and AMD GPUs using Tensorflow from Python programs. Our simple framework was based on file interface which enables to run the machine learning on multiple machines, and then later collected and processed the output files generated by the learners.

For the research, two convolutional neural network learning algorithms with different strength have been chosen as the basis of the modifications.

The problem set given to the learning algorithms was the well known MNIST database of handwritten digits [12].

The results may vary given the stochastic nature of the algorithms, so thousands of experiments with different parameters were performed, and average results were analysed. For the analyses we first measured the performance of the individual learners on the test dataset with different parameters for fuzzification. Since multiple learners have proven to be more successful when we combine their results through voting, we might expect better results by setting the fuzzified class membership together as well. We will show the results of this research in Section 3.2. In these experiments we have measured the standalone test results of the learners, as well as the results of the fuzzy average voting of multiple learners, as a committee machine. When we talk about committee machine voting we can choose from many voting functions, e.g. fuzzy averaging, plurality (or majority) voting, etc. In our research we have used the well-known fuzzy voting [18].

For the analyses we used the Python Numpy and Pandas frameworks. The algorithms run with different epoch counts to see the behavior of our proposed algorithm variations not only with the statistically best settings. In the following sub-section we will show the performance of the proposed fuzzification of training data binary class membership values. For the evaluation we run about one million learning sessions with convolutional neural network algorithms modified according to our proposed methods. The first algorithm variant was built from the algorithm introduced in[2]. The second algorithm variant was based on the algorithm which uses the Squeeze-and-Excitation Network method. We have added to both algorithms the proposed fuzzification of binary class membership values of training data.

3.2. Performance evaluation of training data class membership value fuzzification

We have executed several experiments with two algorithms of different strengths. The algorithm variations were executed with different parameters, e.g. the number of epochs to run, the number of instances in the ensembles and the parameters for the fuzzification of binary class membership values of training data, including parameters which keep the original class membership values. We note that we have executed many learning sessions without fuzzification as well in order to have more reliable results for comparison.

3.2.1. Fuzzification experiment 1

The first experiment ran using the modified variant of. Thousands of learners learned with different epoch counts and different parameters for fuzzification, in-

[2] https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist

cluding parameters which keep the original class membership values($a = 0$, $b = 0$, $c = 1$). We will first show the average results in function of the $c/(b + c)$ ratio.

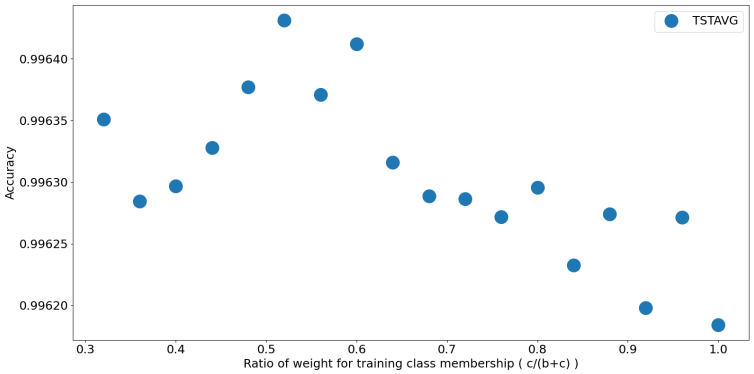

Figure 1: The average accuracy results of our algorithm on test data using different parameters for the fuzzification of the training data class membership values.

Figure 1 shows the average accuracy of the individual learners with different parameters used for the fuzzification. The ratio $c/(b + c)$ of the parameters of Algorithm1 has the meaning of how important the original binary class membership values provided in the training data are. If the ratio is 1.0 , then no fuzzification will happen. As it can be seen, the accuracy achieved was better when the algorithm was used with fuzzification. We note that when the ratio goes below 50% then the performance gets again lower. In that case the fuzzification can change the class membership values to have a big difference from the original values. We will also show the performance using the fuzzy average voting function when using multiple learners.

Figure 2 shows the accuracy of the V1 fuzzy average voting on the same experiment. As we can see, the results using the fuzzy average voting are similar, the fuzzification helps to achieve better performance, i.e. higher accuracy on the training dataset. As the ratio of $c/(b + c)$ increases, i.e., the possibility of fuzzification decreases, so the accuracy achieved tends to decrease as well. We note, that although the shown results are mean values of several measurements, the random behavior of the algorithms can result in fluctuation in performance, some values can be the effect of that. We also show a 3D diagram to better understand the results for different parameters. Since the sum of the parameters a, b and c must be 1.0 we can choose two of these parameters for the X and Y axes of the diagram, and the Z axis can show the average accuracy values. We have chosen the a and b parameters for that, the c parameter for every measurement is $1 - (a + b)$.

Figure 3 shows the average individual accuracy on test data for the a and b parameters. The value with $a = 0$, $b = 0$ coordinates shows the average result without fuzzification. We can see that with values of parameter b around 0.4

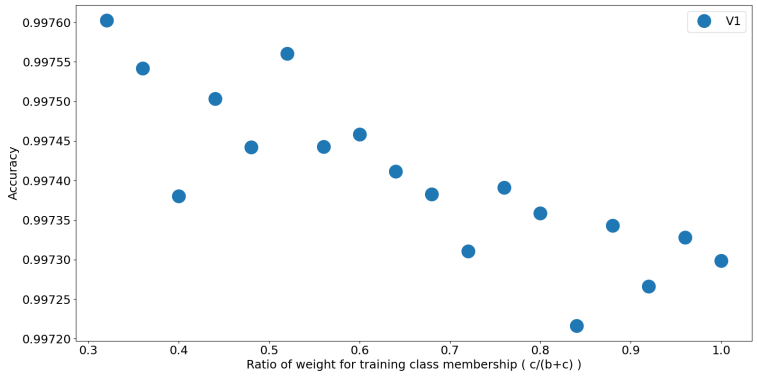

Figure 2: The performance results of our algorithms V1 fuzzy average voting function by 6-20 voters on test data using different parameters for the fuzzification of the training data class membership values.

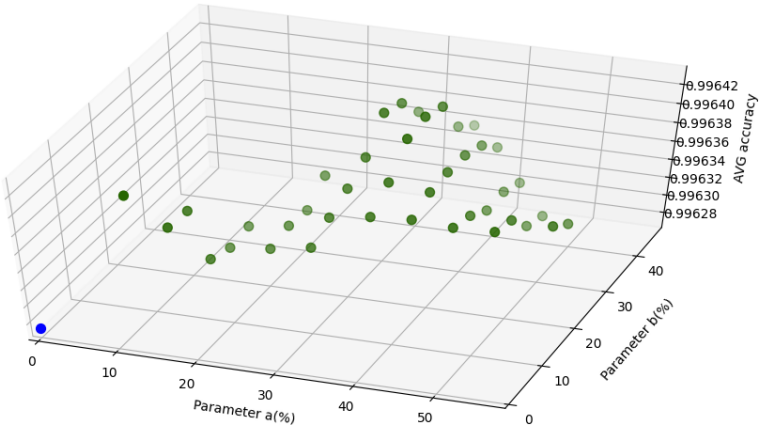

Figure 3: The average performance results of our algorithms on test data using different parameters for the fuzzification of the training data class membership values.

(40%) we had better accuracy, especially when the value of parameter a was close to 0.3 (30%).

3.2.2. Fuzzification experiment 2

The next experiment ran using a modified algorithm of , using the parameters for the fuzzification. The number of epochs we had run the algorithm was from 15 to 20.

Figure 4 shows the average accuracy of the individual learners with different

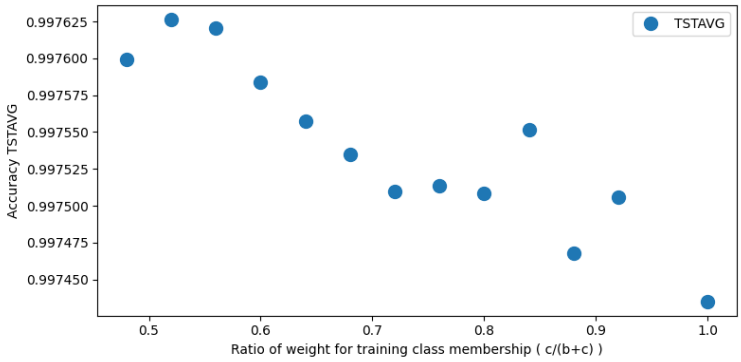

Figure 4: The individual accuracy results of the algorithm on test data using different parameters for the fuzzification of the training data class membership values.

parameters used for the fuzzification. As described for Figure 1 the ratio $c/(b+c)$ tells the importance of the original class membership values of the training data, fuzzification can be done only if the ratio is below 1.0 . As we can see, the accuracy can be better with modest fuzzification. We also note that if the ratio of $c/(b+c)$ decreases to below 0.5 then the accuracy seems to decrease as well. This can be the effect of too much freedom of the algorithm to change the class membership values. For this experiment, too, we have measured the performance using the well-known fuzzy average voting function (V1) when using multiple (6–20) learners.

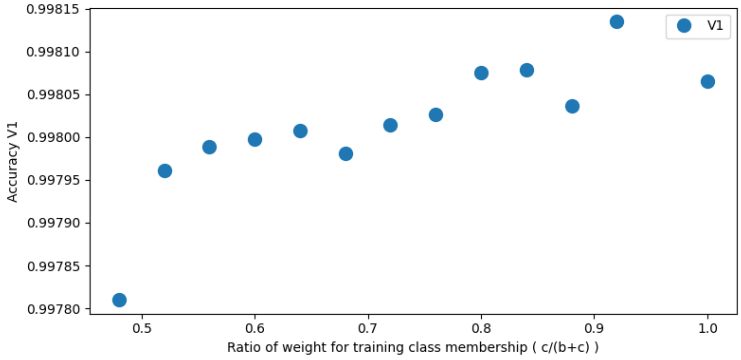

Figure 5: The individual accuracy results of algorithm on test data using different parameters for the fuzzification of the training data class membership values.

Figure 5 shows the results of the V1 fuzzy mean vote in this experiment. The results are different in this case. The accuracy averages using fuzzified training

data class membership values were lower for most parameters than the accuracy using only the original training data. However, there is a promising range what we can look from another perspective as well. Below we show the average accuracy of the learners for different a, b and c parameters on a 3D figure using the fuzzy average (V1) voting function with 6–20 voters. We show the results in function of a and b parameters, while parameter c is dependent on them.

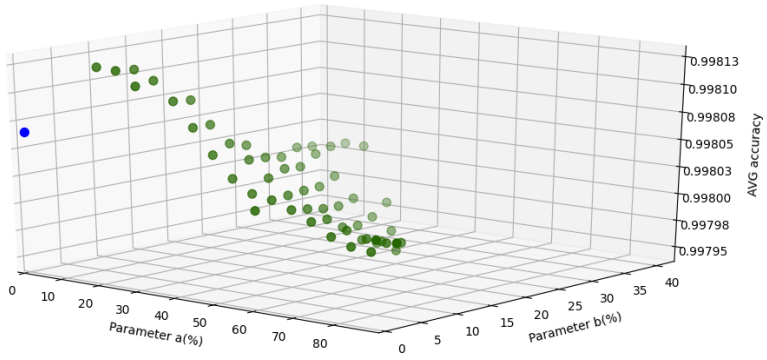

Figure 6: The results of our algorithms' average accuracy on test data using different parameters for the fuzzification of the training data class membership values.

Figure 6 shows the results of thousands of learning sessions which were executed with different a, b and c parameters. The point with $a = 0$, $b = 0$ coordinates shows the average result when class membership values of training data were not corrected. We can see that the results were higher with lower a and b parameter values. For such parameters the c parameter is higher, so only minor corrections on the training data class membership values can be made.

We note that this is a strongly different behavior compared to the performance of fuzzification with the first (weaker) algorithm variant. This is probably because the Squeeze-and-Excitation Network has much higher accuracy on this dataset, and this might mean that it can handle misclassified train samples better, so fuzzification may result only in minor improvement.

For a range of parameter values where parameter a and b are not zero but both have low values the accuracy was better using the proposed fuzzification. That means that fuzzification in a lower rate had an improvement even for this strong algorithm.

4. Conclusion

From the results of our fuzzification experiments we can conclude that the fuzzification of the training data binary class membership values can improve the accuracy of the prediction of class membership values.

The results show that the parameters of our proposed fuzzification algorithm highly affect the accuracy of the predictions of the learners. Their effect was different depending on the basic algorithm to which we added it. The performance improvement of individual test accuracy was significant for both algorithms we used for the evaluation. When we compared the accuracy of the fuzzy average voting function for different parameters of the fuzzifying algorithm we had also significant improvement for the weaker algorithm with wider range of the parameters of the fuzzification algorithm, but in case of the stronger algorithm only a minor improvement was observed for a narrow range of these parameters. Further measurements will be performed to analyze this behavior with the same dataset and with other datasets as well.

References

[1] R. Basbous, B. Nagy, T. Tajti: *Short Circuit Evaluations in Gödel Type Logic*, Proc. of FANCCO 2015: 5th International Conference on Fuzzy and Neuro Computing, Advances in Intelligent Systems and Computing 415 (2015), pp. 119–138, doi: https://doi.org/10.1007/978-3-319-27212-2_10.

[2] R. Basbous, T. Tajti, B. Nagy: *Fast Evaluations in Product Logic: Various Pruning Techniques*, in: FUZZ-IEEE 2016 - the 2016 IEEE International Conference on Fuzzy Systems, Vancouver, Canada: IEEE, 2016, pp. 140–147, doi: 10.1109/FUZZ-IEEE.2016.7737680.

[3] A. Borovykh, S. Bohte, C. W. Oosterlee: *Conditional time series forecasting with convolutional neural networks*, arXiv preprint arXiv:1703.04691 (2017).

[4] D. Ciresan, U. Meier, J. Schmidhuber: *Multi-column deep neural networks for image classification*, in: 2012 IEEE conference on computer vision and pattern recognition, IEEE, 2012, pp. 3642–3649.

[5] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber: *Flexible, high performance convolutional neural networks for image classification*, in: Twenty-second international joint conference on artificial intelligence, 2011.

[6] A. Conneau, H. Schwenk, L. Barrault, Y. Lecun: *Very deep convolutional networks for text classification*, arXiv preprint arXiv:1606.01781 (2016).

[7] R. Fullér: *Fuzzy systems*, in: Introduction to Neuro-Fuzzy Systems, Springer, 2000, pp. 1–131.

[8] S. Haykin: *Neural Networks: A Comprehensive Foundation*, 2nd, USA: Prentice Hall PTR, 1998, isbn: 0132733501.

[9] J. Hu, L. Shen, G. Sun: *Squeeze-and-excitation networks*, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[10] G. Kovásznai, C. Biró, B. Erdélyi: *Puli–A Problem-Specific OMT solver*, in: Proc. 16th International Workshop on Satisfiability Modulo Theories (SMT 2018), 371, 2018.

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner: *Gradient-based learning applied to document recognition*, Proceedings of the IEEE 86.11 (1998), pp. 2278–2324.

[12] Y. LeCun, C. Cortes, C. J. Burges: *The MNIST database of handwritten digits, 1998*, URL http://yann. lecun. com/exdb/mnist 10.34 (1998), p. 14.

[13] L. Li, Q. Hu, X. Wu, D. Yu: *Exploration of classification confidence in ensemble learning*, Pattern recognition 47.9 (2014), pp. 3120–3131.

[14] U. Naftaly, N. Intrator, D. Horn: *Optimal ensemble averaging of neural networks*, Network: Computation in Neural Systems 8.3 (1997), pp. 283–296,
doi: 10.1088/0954-898X_8_3_004, eprint: https://doi.org/10.1088/0954-898X_8_3_004,
url: https://doi.org/10.1088/0954-898X_8_3_004.

[15] B. Nagy, R. Basbous, T. Tajti: *Lazy evaluations in Łukasiewicz type fuzzy logic*, Fuzzy Sets and Systems 376 (2019), Theme: Computer Science, pp. 127–151, issn: 0165-0114,
doi: https://doi.org/10.1016/j.fss.2018.11.014,
url: http://www.sciencedirect.com/science/article/pii/S0165011418309357.

[16] D. Opitz, R. Maclin: *Popular ensemble methods: An empirical study*, Journal of artificial intelligence research 11 (1999), pp. 169–198.

[17] S. Russell, P. Norvig: *Artificial intelligence: a modern approach* (2002).

[18] C. Sammut, G. I. Webb: *Encyclopedia of machine learning*, Springer Science & Business Media, 2011.

[19] F. Tramèr, A. Kurakin, N. Papernot, et al.: *Ensemble adversarial training: Attacks and defenses*, arXiv preprint arXiv:1705.07204 (2017).

[20] S. Wan, H. Yang: *Comparison among Methods of Ensemble Learning*, 2013 International Symposium on Biometrics and Security Technologies (2013), pp. 286–290.

[21] S. C. Wong, A. Gatt, V. Stamatescu, M. D. McDonnell: *Understanding data augmentation for classification: when to warp?*, in: 2016 international conference on digital image computing: techniques and applications (DICTA), IEEE, 2016, pp. 1–6.

[22] L. A. Zadeh, G. J. Klir, B. Yuan: *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*, WORLD SCIENTIFIC, 1996,
doi: 10.1142/2895, eprint: https://www.worldscientific.com/doi/pdf/10.1142/2895,
url: https://www.worldscientific.com/doi/abs/10.1142/2895.

[23] L. A. Zadeh: *Fuzzy logic—a personal perspective*, Fuzzy Sets and Systems 281 (2015), Special Issue Celebrating the 50th Anniversary of Fuzzy Sets, pp. 4–20, issn: 0165-0114,
doi: https://doi.org/10.1016/j.fss.2015.05.009,
url: http://www.sciencedirect.com/science/article/pii/S0165011415002377.