Improving the Performance of Neural Networks with an Ensemble of Activation Functions

Arijit Nandi[1], Nanda Dulal Jana[2], Swagatam Das[3]
[1,2]*Department of Computer Science and Engineering, National Institute of Technology Durgapur-713209, India*
[3]*ECS Unit, Indian Statistical Institute, Kolkata- 700108, India*
[1]an.17p10354@mtech.nitdgp.ac.in, [2]nandadulal@cse.nitdgp.ac.in, [3]swagatam.das@isical.ac.in

***Abstract*—Activation functions in the neural networks play an important role by introducing non-linear properties to the neural networks. Thus it is considered as one of the essential ingredients among other building blocks of a neural network. But the selection of the appropriate activation function for the enhancement of model accuracy is strenuous in a sense; the performance of the NN-model is influenced by a proper selection of activation function for a dataset. Proper activation function selection is still a trial and error method for which the model accuracy improves for classification. As a solution to this problem, we have proposed an activation function ensembling by majority voting that has significantly improved the model accuracy in a classification context. The proposed model is tested on four benchmark datasets such as MNIST, Fashion MNIST, Semeion, and ARDIS IV datasets. The result shows that the performance of the proposed model is appreciably better than other traditional methods such as Convolutional Neural Network (CNN), Support Vector Machine (SVM), Recurrent Neural Network (RNN).**

***Index Terms*—Neural Network, Activation function, Multi classifier ensemble, Ensemble Learning, Activation function ensemble.**

I. Introduction

Classification is one of the most frequently encountered decision-making tasks of human activity; the problem occurs when an object needs to be assigned into a predefined group or class based on several observed attributes related to that object [1]. According to the *No free lunch theorem for optimization* [2]–[4], no single algorithm can produce the most appropriate learning model for all problems in any domain, which enforces to introduce the ensemble methods. One technique that universally increases accuracy is by ensembling multiple predictive models [5]. In [6], the author explained the advantages of using multiple classifier models rather than a single model. The general idea behind combining multiple pattern classifiers is the use of a methodology to produce a final decision given on the output of the learners [7]. In 1959, the first learning model of a system with multiple experts was developed [8]. Based on the multiple expert models, different studies were made such as a committee, classifier fusion, combination, aggregation, a mixture of experts, etc. to solve pattern recognition problems [9]–[11]. Recently, researchers and practitioners are employing ensemble learning concepts in the task of a multi-classifier pattern recognition system. In the world of statistics and machine learning, ensemble learning techniques attempt to make the performance of the predictive models better by improving their accuracy.

The ensemble of classifiers is a set of learning models where decisions are combined to enhance the performance of the pattern recognition system. The relevant and appropriate patterns, as well as inadequate patterns for the ensembling method, has been studied experimentally and theoretically in [12]. Most of the studies demonstrated that the classifier ensembling technique for classification problems are often more appropriate than the individual based learner. Sometimes, it can be found that a weak classifier is capable of outperforming on a classification problem that was proposed in [13]. Neural network-based classifiers that are unstable for a problem were stabilized by using a multi-classifier system [14]. Moreover, the problem of handling noisy data can be processed better by the ensemble of classifiers, which increases the robustness of the decisions [15]. The ensemble-based mechanism developed in [16] reveals that many classifiers have the potential to improve the accuracy and speed based on ensemble techniques.

Artificial Neural Network (ANN) is a valuable tool and widely used for classification problems as well as in function approximation for optimization problems. From the ANN literature, it can be observed that the activation function is an essential component of the neural network architecture and paying a significant impact on the performance of classification accuracy. It has also seen that for one particular data-set, one activation is giving better results, but others are giving a poor performance. This leads to the employment of different activation functions in neural architecture for different classification problems. The various activation functions started from sigmoid to ReLU have shown their potential advantages in learning while solving a problem. However, there is a lack of deficiency in choosing an appropriate activation function for a particular classification problem.

The recent advancements of the study of ANN's mainly focused on the research areas such as network architecture [17], [18], optimization method (AdaDelta) [19] and batch normalization [20], activation functions and objective functions for loss surface [21]. In the case of neural architecture, Highway Network [22], Residual Network [23], Memory Network [24], etc. have been developed for enhancing the performance of classification accuracy on different complex classification problems. The ground breaking work is done by proposing an activation function named Rectified Linear Unit (ReLu) [25], [26], which revolutionizes the way of ANN

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

working principle. Before ReLu, the most widely used popular activation functions are sigmoid and hyperbolic tangent (tanh). Unfortunately, these are incapable in case of training deep neural architecture due to gradient vanishing problem. On the other hand, the ReLu activation function has a problem when the value becomes negative, and its gradient becomes zero imply no further training (weight updating) is possible for Deep Neural Network (DNN). The drawbacks of ReLu activation is overcome by a variant of ReLu known as LeakyRelu. Therefore, researchers are focused on the development of new activation functions for DNN and incorporating ensemble learning mechanisms on activation functions to select appropriate one rather single one for solving complex classification problems.

In [27], researchers have utilized softplus units for DNNs in acoustic modeling for context-independent phoneme recognition tasks. They have revisited the Restricted Boltzman Machine (RBM) in which the pre-training and dropout strategies are applied to improve the performance of softplus units. Their experiment shows that DNNs with softplus units have performed significantly better performance, and convergence speed is improved as compared to standard sigmoid units and ReLUs. Most of the cases, while training DNN, the trainable parameters are weights and biases. But the proper selection of activation functions is set across the DNN by trial and error methods. In [28], the authors applied a technique called trainable activation function for DNN. They approximated conventional nonlinear activation functions for a Taylor Series, and the coefficients were retrained simultaneously with other parameters. In their experiment, the nonlinear activation functions are sigmoid, tanh, ReLu, and softplus. Convolution Neural Network (CNN) is a special kind of DNN comprises of many filters for automatic feature extraction from the image and classifying those images. In [29], researchers have proposed a new activation function called hyperbolic linear units (HLUs) for deep CNN. They have experimented on three popular CNN architectures, LeNet, Inception network, and ResNet on various benchmark datasets such as MNIST, CIFAR-10, and CIFAR-100. According to the result, their proposed activation function speeds up the learning process in deep CNN with better performance in image classification tasks. In [29], researchers have proposed a new computationally efficient activation function called SQNL for Multi-Layer Perceptron to make faster convergence. In [30], researchers have proposed an activation function called Hexpo. According to them, this activation function is vanishing proof. Hexpo activation function can scale the gradient and to overcome the vanishing gradient problem. Researchers have tested their proposed approach MNIST hand recognition dataset and CIFAR-10 tiny image recognition data-set. Their proposed activation function outperforms the rectified linear unit family (rectified linear unit and exponential linear unit) by the classification accuracy and speed of training. Also, for CIFAR-10, Hexpo outperforms ReLu but performs similarly as ELU. There is a plethora of work for developing the best activation functions in deep neural network research. Even though the development of DNN is advance but choosing a proper activation function is still a bottleneck in the DNN research community.

In [5], the activation ensemble idea is combined in each activation function at every neuron in the network. In the proposed activation ensemble, extracted features are capture by various activation functions and ensembling techniques to achieve the best feature combination, which is available in the classification. The proposed approach was experimented on four benchmark data sets such as MNIST, CIFAR-100, ISOLET, and STL-10 and provided results with and without ensemble methods. The author claimed that their proposed approach is significantly better than original models in terms of better classification accuracy. From the literature of the activation functions, it can be summarized that most of the works are related to the improvement of the classification accuracy. More precisely, the mechanism proposed in [5], can be achieved more easy way. The paper is focused on the ensembling of activation functions which is entirely different from [5] for solving image classification data sets.

In this work, activation functions ensemble at decision-level fusion (called AFE) is proposed. The idea is to ensemble various activation functions at decision level by majority voting based ensembler. In the majority voting based ensembler, the output from each of activation is ensembled to generate the final output class for classification, making sure an increment in classification accuracy and prediction rate. The effectiveness of our proposed approach is examined with four well-known benchmark data-sets such as MNIST, Fashion MNIST, Semeion, and ARDIS IV. The obtained results are compared with five different activation functions and favoring significant performance improvement of the proposed methods. Moreover, the proposed method is compared with some state-of-the-art models such as Convolutional Neural Network (CNN), Support Vector Machine (SVM), Recurrent Neural Network (RNN) on the same data sets.

The rest of the paper is organized as follows: section II discusses the preliminaries. Section III presents the proposed model for the activation function ensemble. In section IV, experimental results are described, including the benchmark data-sets, experimental settings of different activation functions and comparison with the state-of-the-art classification models is presented. Finally, conclusions are summarized and future work is highlighted in section V.

II. Preliminaries

A. Feed-forward Neural Network (FFNN)

FFNN is a type of artificial neural network consisting of many connected layers of neurons. The first layer is the input layer (IL) because it takes the input features of observation, the last layer is the output layer (OL) because it brings the output class associated with the input feature. The layers in between IL and OL are called hidden layers (HL). Each layer consisting of a different number of artificial neurons. Each neuron consists of activation function which introduces nonlinearity in data. In the hidden and output layers, a bias term is associated with each neuron for adjusting the threshold value

of the activation function. The simple FFNN is depicted in Figure 1.

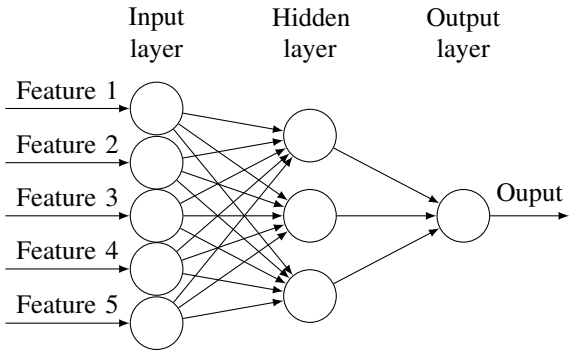

Fig. 1. Feed Forward Neural Network

Suppose, an FFNN consists of a single hidden layer of size Q and output layer size R, the input vector x_p with I dimension. Then the mathematical calculation for the output will be as follows:

$$y_{q,p} = af_{y_q}\left(\sum_{i=1}^{I+1} w_{q,i}x_p, p\right), \forall q \in \{1, 2, ..., Q\} \quad (1)$$

$$O_{r,p} = af_{O_r}\left(\sum_{q=1}^{Q+1} w_{r,q}y_q, p\right), \forall r \in \{1, 2, ..., R\} \quad (2)$$

where y_q is the q^{th} hidden unit, w_{qi} is weight from input value x_p to hidden layer unit y_q. O_r is the r^{th} output unit and w_{rq} is the weight from hidden unit to output unit. af_{y_q} and af_{O_r} are the activation functions for hidden unit and output unit respectively.

B. Activation Functions

As discussed in earlier section that each layer consists of artificial neurons, which are the processing and central unit of a neural network. Each neuron comprises of two functionalities, first is the net input or summation of weighted input's ($\sum$) and another is activation function or transfer function ($af(.)$). The activation functions are used to introduce non-linearity in the input signal so that the mapping from input signal to the output will be perfect. The most popular activation functions used in the study are discussed as follows:

- **Sigmoid:** This activation function is widely used with FFNNs [1]. The mathematical representation is given below.

$$af(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

The function is differentiable, monotonically increasing and produces output in (0, 1).

- **Hyperbolic Tangent (Tanh):** The mathematical representation of this function is as follows:

$$af(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (4)$$

The property of hyperbolic tangent is similar to sigmoid function. This function has larger output range in (-1, 1).

- **Rectified Linear Unit (ReLu):** The ReLu is simple activation function and is given by [31]:

$$af(x) = max(0, x) \quad (5)$$

It has lower bound and used to represent data in range in $(0, \infty)$. Relu is successfully applied in the areas of Deep Neural Neworks (DNNs).

- **Exponential Linear Unit (ELU):** It is an another activation function. It is designed to rectify the problems of ReLu activation function. It has better tendency to converge cost to zero and produce better result [32]. The activation function is defined as follows:

$$af(x) = \begin{cases} x & x > 0 \\ \alpha.(e^{x}–1) & x <= 0 \end{cases} \quad (6)$$

- **Leaky Rectified Linear Unit (LeakyRelu):** LeakyRelu is a variantion of ReLU activation function. LeakyReLU allows a small, non-zero, constant gradient α (Normally, $\alpha = 0.01$). This activation function is defined as follows:

$$af(x) = \begin{cases} x & x > 0 \\ \alpha x & x <= 0 \end{cases} \quad (7)$$

C. Stochastic Gradient Descent

This section discusses the training algorithm (Supervised Learning) for FFNN. In case of the supervised learning, each sample S represent as a pair (x, y) comprises of input x and a scalar output y. A loss function or cost function $L(\hat{y}, y)$ is consider which measures the difference (cost) between predicted output($\hat{y}$) and the actual output y. A family F of functions $f_w(x)$, which is parameterized by weight vector w. It is needed that the function $f \in F$ minimizes the loss $Q(S, w) = L(f_w(x), y)$ averaged on samples.

$$E_n(f) = \frac{1}{n}\sum_{i=i}^{n} L\left(f(x_i), y_i\right)$$

$E_n(f)$ is the empirical risk, which needs to be minimized. It measures the performance of a model on training set.

The empirical risk $E_n(f)$ is often minimized by the gradient descent(GD) method. In GD, each iteration updates the weights w on the basis of the gradient of $E_n(f)$. The weight vector can be updated as follows:

$$w_{t+1} = w_t - \eta\frac{1}{n}\sum_{i=1}^{n} \nabla_w Q(S_i, w_i)$$

where η is the learning rate or step size. It is to be noted that large η increases the chances of overshooting the global minima, on the other side small η converges too slow. So, in practice η is usually selected with experiments.

The gradient descent algorithm may be infeasible when the training data-set is huge. Then the stochastic version of the gradient descent algorithm called stochastic gradient algorithm (SGD) is used. In SGD, instead of calculating the gradient of

$E_n(f)$ exactly, each iteration estimates this gradient on the basis of single randomly picked sample S_i.

$$w_{t+1} = w_t - \eta_t \nabla_w Q(s_i, w_i)$$

The stochastic process $\{w_t, t = 1, ...\}$ depends upon on randomly picked examples in each iteration.

D. Ensemble Learning

The term ensemble resembles a group that is working for an overall result. In general, the idea behind ensemble learning is to combine multiple learners (base or weak learners) in an appropriate or meaningful way to enhance the accuracy of a decision-making process (e.g., Classification) [33]–[35]. The first underlying theoretical foundation of the combination methods can be found in the political science field, known as Condorcet's Jury Theorem in 1785 ([36]). In [37], has given three reasons (Statistical, computational ,and representative) for supporting ensemble learning methods. Fig. 2 depicts the general ensemble process.

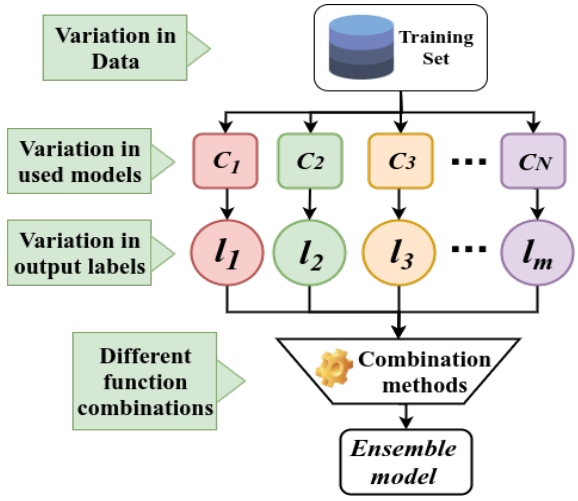

Fig. 2. Traditional ensemble process.

1) Majority Voting/Hard voting: It is a meta-classifier for combining similar or computationally different classifiers for classification via majority or plurality voting. This is simplest case of majority voting. Let consider, a classification problem with class label $l = \{l_1, l_2, ..., l_m\}$ and N number of classifier set $M = \{C_1, C_2, ..., C_N\}$. For each instance i let consider,

$$\chi l_k(C_j(i)) = \begin{cases} 1 \;\; if \;\; C_j(i) = l_k \;\; with \;\; l_k \in l \\ 0 \;\; if \; C_j(i) = l_q \;\; with \;\; l_k, l_q \in l \;\; and \; q \neq k \end{cases} \tag{8}$$

then, combining N classifiers decision with a majority vote, i,e. the ensemble predictions can be defined as:

$$P_{ens} = \arg\max \sum_{j=1}^{N} \chi l_k(C_j(i)) \tag{9}$$

In this case all N classifiers' votes are having same priority for the ensemble prediction.

III. Proposed Model

In this section, the proposed AFE approach is discussed. In the AFE method, instead of different classification models, one FFNN structure is trained with five different activation functions as compared to the traditional ensemble methods. Where each activation function is fixed throughout the whole network (i,e input layer, hidden layer, and output layer are having the same activation function) while training. After training, the base model is tested, and the results are stored unless all activation functions from the activation pool are over. Then from each of every predicted class by each activation function is passed through the majority voting ensemble approach, which picks up the major class labels from the predicted labels to declare the final class for the particular observation. The proposed approach is presented in Fig. 3.

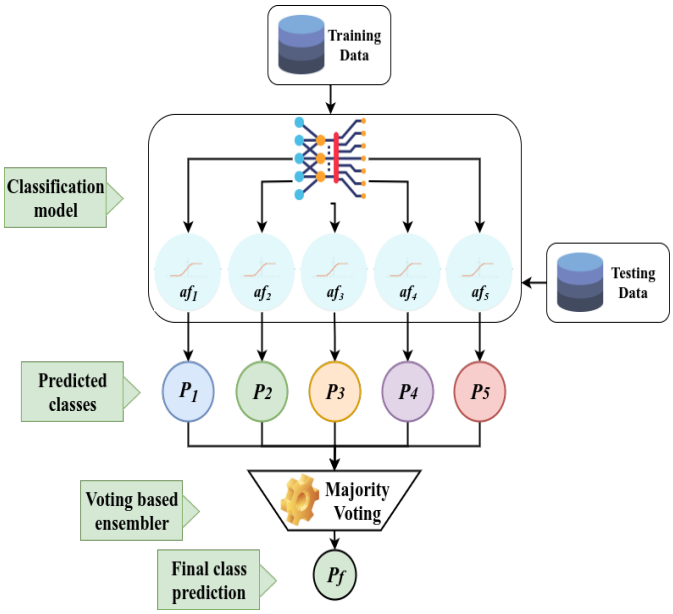

Fig. 3. Activation function ensemble model

For the experimental purpose of the proposed method, five activation functions are considered for ensembling. In Fig. 3, af_1, af_2, ..., af_5 represents five activation functions where af_1 is Sigmoid, af_2 is Relu, af_3 is hyperbolic tangent, af_4 is ELU and finally af_5 is LeakyRelu. An SGD optimizer is utilized for training the FFNN (Fig. 1) with different activation functions. When a particular activation function is selected, it is the same for all layers in FFNN while training. It goes on until the activation function set (af) is empty. And when an activation function is selected and used for training, that activation function is removed from the activation function set (af). P_1, P_2, .., P_5 are the predicted outputs of af_1, af_2, af_3, af_4, and af_5 activation functions applied to the same FFNN, respectively. The final predicted class labels for the testing data (P_f) is generated after applying voting based (Majority voting) ensemble technique in all the predicted outputs. The pseudo-code of the proposed AFE method is presented in Algorithm 1.

Algorithm 1: AFE pseudo code

Result: P_f: final class prediction
Initialization;
1. Activation function set: $af = \{af_1, af_2, ..., af_5\}$;
2. ANN model define ;
3. K-fold cross divition of dataset: $folds\{\text{dataset}\}$;
4. Prediction result matrix: $P[|af|][|folds|]$;
Part 1: Model fitting ;
for $i \leftarrow 1$ **to** $|af|$ **do**
  for $j \leftarrow 1$ **to** $|folds|$ **do**
    1. Train ANN model af_i ;
    2. Test ANN model;
    3. Store class predictions in $P_{i,j}$;
Part 2: Ensembling (Majority voting);
1. Final class prediction(P_f)= *majorityvote*(P) ;
2. Test ensemble model.;

IV. Experimental Results

A. Dataset Description:

In this experiment, four different datasets are used. The details of data sets are mentioned Table I:

TABLE I
Details of six datasets used for this experiment

Dataset	Instances	Attributes	Classes
Semion Handwritten digit [38]	1593	256	10
MNIST [39]	70000	785	10
Fashion MNIST [40]	70000	785	10
ARDIS dataset IV [41]	7600	785	10

For the sake of simplicity, the experimented data-sets are aliased as: Semion Handwritten digit as **D1**, MNIST as **D2**, Fashion MNIST as **D3** and ARDIS dataset IV as **D4**.

B. Experimental setup

This experiment is done in Python 3.6 and the packages used to develop FFNN are Tensorflow 1.15, Keras [42] and sklearn 22.1. The machine configuration is Ubuntu 18.04 64 bit OS, processor core-i7-7700HQ with RAM 8Gb 2400MHz and 4Gb-Nvidia GTX-1050 graphics.

C. Neural Network structure

For unbiased comparisons of the proposed method, the 5-Fold cross-validation method is considered throughout the experiment. The training batch size is set as 32 and the epoch is set to 100 for every data-set. The network size is having input layer (number of neurons is equal to feature size), four hidden layers with size (hidden layer 1 is having 300 neurons, hidden layer 2 is having 100 neurons, hidden layer 3 is having 200 neurons) and output layer (number of neurons is equal to the class size).

D. Performance metric

The proposed classifier is evaluated using standard metrics such as Precision (*Pre*), Recall (*Rec*), average Accuracy (*Acc*) and F-Measure (*FM*) for multiclass classification measures [43]. There is micro and macro averaging for those performance metrics in multiclass classification. For our case, we have considered the macro-averaging because it indicates how good a classifier performs in each class equally. These metrics with macro-averaging for multiclass classification are shown in Eq. 10, Eq. 11, Eq. 12 and Eq. 13 respectively [43].

$$Pre_{macro} = \frac{\sum_{i=1}^{|l|} \frac{TP_i}{TP_i+FP_i}}{|l|} \tag{10}$$

$$Rec_{macro} = \frac{\sum_{i=1}^{|l|} \frac{TP_i}{TP_i+FN_i}}{|l|} \tag{11}$$

$$Acc = \frac{\sum_{i=1}^{|l|} \frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}}{|l|} \tag{12}$$

$$FM_{macro} = 2 * \frac{Pre_{macro} * Rec_{macro}}{Pre_{macro} + Rec_{macro}} \tag{13}$$

- $|l|$ is the number of classes.
- **True positives (TP_i):** actual class is positive and predicted class is positive.
- **True negatives (TN_i):** actual class is negative and predicted class is negative.
- **False positives (FP_i):** actual class is negative and predicted is negative.
- **False negatives (FN_i):** actual class is positive and predicted is negative.

Where FM_{macro} is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. In this experiment, FM_{macro} and Acc metrics are prioritised to establish the effectiveness of the proposed approach.

E. Results and Discussions

Table II and Table III represents the average F-measure and classification accuracy comparison among five activation functions with the proposed AFE technique for improving classification accuracy.

TABLE II
FM_{macro}-measure(%) comparison table

Dataset	Activation Functions					
	Sigmoid	ReLu	Tanh	ELU	Leaky Relu	AFE
D1	4.60	92.40	92.55	92.28	92.43	**98.53**
D2	93.18	96.71	96.22	97.23	96.76	**99.46**
D3	85.84	88.74	88.32	88.87	88.71	**98.05**
D4	18.70	92.99	91.01	92.56	92.64	**98.64**

TABLE III
ACCURACY(%) COMPARISON TABLE

Dataset	Activation Functions					
	Sigmoid	**ReLu**	**Tanh**	**ELU**	**Leaky Relu**	**AFE**
D1	12.68	92.40	92.58	92.27	92.46	**98.55**
D2	93.18	96.74	96.25	97.25	96.79	**99.47**
D3	85.91	88.75	88.32	88.89	88.69	**98.06**
D4	27.91	92.99	91.02	92.57	92.64	**98.64**

Table II and Table III reported the results of the mean F-measure and accuracy for the data-sets over 100 epochs with 5-fold cross validations. From the F-measure table, it can be depicted that the proposed approach is capable to outperform all the activation functions individually in terms of F-measure. Also in accuracy comparison AFE has achieved significantly better accuracy than other activation functions. The possible reason for this significant accuracy as well as FM_{macro} improvement is because of taking majority voting of class label decisions from those different base classifiers (activation functions) to make the final class label instead of relying on a single one.

While experimenting, we have set the epoch is 100, and the batch size is 32 for all activation functions. In D2 and D3, the 70000 data samples are adequate for stable training of the models. The model with the sigmoid activation function, therefore, has shown better performance. But in D1 and D4, the data samples are only 1593 and 7600, respectively. Thus, due to the availability of comparatively less data samples with the same epoch and batch size, the model is prone to fail over capturing the intricate pattern in those two datasets [44]. This leads the neural network model with the sigmoid activation function to have an under-fitting phenomenon for D1 and D4 dataset only, resulting poor performance in terms of Acc and FM_{macro} measures.

F. Comparison with state-of-the-art literature

The propose method is compared with recent models present in the state-of-the-art literature [5] and [41] to demonstrate the potential capability for solving the classification problem.

1) **Comparison with [5]:** For MNIST dataset, the researchers have used CNN and FFN. From their article, the FFN structure is 784 (input layer neurons)−400 (first hidden layer)−400 (second hidden layer)−400 (third hidden layer)−10 (output layer neurons) with an accuracy of 98.37% (activation ensemble scheme) and the epoch size is 82. AdaDelta is used for training the neural network with a learning rate of 1.0. They have not mentioned any cross-validation technique. In the AFE experiment, network size is 784 (input layer neurons)−300 (first hidden layer)−100 (second hidden layer)−200 (third hidden layer)−10 (output layer neurons) with an average accuracy of 99.34% (5-fold cross-validation) and 50 epochs are used for training. The optimizer is SGD with a learning rate of 0.1. The comparison is shown Table IV.

TABLE IV
ACCURACY(%) COMPARISON WITH RECENT MODELS PRESENT IN LITERATURE [5].

Method	**MNIST dataset (D2) recognition accuracy (%)**
CNN	99.18
FFN	93.78
AFE	**99.34**

From the comparison, it is noted that the proposed approach AFE is significantly better in terms of classification accuracy and outperforms CNN and FFN.

2) **Comparison with [41]:** Another comparison with recent literature is presented in Table V.

TABLE V
ACCURACY(%) COMPARISON WITH RECENT MODELS PRESENT IN [41]

Method	**MNIST dataset (D2) recognition accuracy (%)**	**ARDIS dataset (D4) recognition accuracy (%)**
CNN	99.18	98.60
SVM	93.78	92.40
HOG–SVM	97.82	95.50
kNN	97.31	89.60
Random forest	94.82	87.00
RNN	96.95	91.12
AFE	**99.47**	**98.64**

From the comparison, it is also clear that the proposed approach AFE has significantly achieved better accuracy improvement than several traditional methods and outperforms them all in the MNIST dataset. AFE has achieved similar accuracy in comparison to CNN in the ARDIS dataset.

V. CONCLUSION

One activation function can not be able to deliver more in-depth insight into the data to make the correct classification. But applying many diverse activation functions for a particular can understand the data better, and their opinions about the classification put together can deliver the correct class label. Which is beneficial for correct class label prediction instead of relying on one activation function. In this work, the activation function ensemble at decision-level for final class prediction is proposed to increase classification accuracy. The experiment results show that the proposed approach AFE can increase the classification and outperforms each activation function along with outperforming state-of-the-art classification models.

Future research will involve investigating the effects of each activation function's contribution and importance to make final class label prediction in classification. Also, the impact on classification using activation function ensembles with the consideration that the activation functions will differ in hidden and output layers. Lastly, the diverse behavior of ensemble methods will be investigated.

REFERENCES

[1] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, Nov 2000.

[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[3] D. H. Wolpert and D. H. Wolpert, "What the no free lunch theorems really mean; how to improve search algorithms," 2012.

[4] Y. Ho and D. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of Optimization Theory and Applications*, vol. 115, no. 3, pp. 549–570, Dec 2002.

[5] M. Harmon and D. Klabjan, "Activation Ensembles for Deep Neural Networks," *arXiv e-prints*, p. arXiv:1702.07790, Feb 2017.

[6] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated, 2012.

[7] *Pattern Classification Using Ensemble Methods*, pp. 1–15. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789814271073

[8] O. G. Selfridge, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Pandemonium: A Paradigm for Learning, pp. 115–122.

[9] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, Jan. 2000.

[10] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[11] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.

[12] G. Brown and L. Kuncheva, ""good" and "bad" diversity in majority vote ensembles," 04 2010, pp. 124–133.

[13] L. I. Kuncheva, J. C. Bezdek, and R. P. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, vol. 34, no. 2, pp. 299 – 314, 2001.

[14] F. A. Breve, M. P. Ponti-Junior, and N. D. A. Mascarenhas, "Multilayer perceptron classifier combination for identification of materials on noisy soil science multispectral images," in *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007)*, Oct 2007, pp. 239–244.

[15] M. P. Ponti and N. D. A. Mascarenhas, "Material analysis on noisy multispectral images using classifier combination," in *6th IEEE Southwest Symposium on Image Analysis and Interpretation, 2004.*, March 2004, pp. 1–5.

[16] M. P. Ponti and J. P. Papa, "Improving accuracy and speed of optimum-path forest classifier using combination of disjoint training subsets," in *Multiple Classifier Systems*, C. Sansone, J. Kittler, and F. Roli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 237–248.

[17] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2017–2025.

[18] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, "Noisy activation functions," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 3059–3068.

[19] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[21] Ç. Gülçehre, M. Moczulski, F. Visin, and Y. Bengio, "Mollifying networks," *CoRR*, vol. abs/1608.04980, 2016. [Online]. Available: http://arxiv.org/abs/1608.04980

[22] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," *CoRR*, vol. abs/1507.06228, 2015. [Online]. Available: http://arxiv.org/abs/1507.06228

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[24] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1410.3916

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814.

[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[27] Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li, "Improving deep neural networks using softplus units," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–4.

[28] Hoon Chung, Sung Joo Lee, and Jeon Gue Park, "Deep neural network using trainable activation functions," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 348–352.

[29] A. Wuraola and N. Patel, "Sqnl: A new computationally efficient activation function," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–7.

[30] S. Kong and M. Takatsuka, "Hexpo: A vanishing-proof activation function," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 2562–2567.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 807–814.

[32] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015.

[33] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms: Second Edition*, 01 2014, vol. 47.

[34] R. Polikar, *Ensemble Learning*. Boston, MA: Springer US, 2012, pp. 1–34.

[35] G. Valentini and F. Masulli, "Ensembles of learning machines," in *Proceedings of the 13th Italian Workshop on Neural Nets-Revised Papers*, ser. WIRN VIETRI 2002. London, UK, UK: Springer-Verlag, 2002, pp. 3–22.

[36] C. Zucco, "Multiple learners combination: Cascading," in *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds. Oxford: Academic Press, 2019, pp. 539 – 541.

[37] T. G. Dietterich, "Ensemble methods in machine learning," in *MULTIPLE CLASSIFIER SYSTEMS, LBCS-1857*. Springer, 2000, pp. 1–15.

[38] I. h. Tactile Srl, Brescia, "Semeion handwritten digit data set," https://archive.ics.uci.edu/ml/datasets/semeion+handwritten+digit, 1994.

[39] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[40] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[41] H. Kusetogullari, A. Yavariabdi, A. Cheddad, H. Grahn, and J. Hall, "Ardis: a swedish historical handwritten digit dataset," *Neural Computing and Applications*, pp. 1–14, 2019.

[42] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[43] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing Management*, vol. 45, no. 4, pp. 427 – 437, 2009.

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.