JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 5 No. 2 (2024) 264-273
https://publisher.uthm.edu.my/ojs/index.php/jscdm

ABNet - Leveraging the AdaBoost to Boost Neural Network Performance

Ifra Altaf[1], Manzoor Ahmad Chachoo[1]*

[1] *Department of Computer Sciences*
University of Kashmir, Srinagar, 190006, INDIA

*Corresponding Author: c_manzoor@yahoo.com
DOI: https://doi.org/10.30880/jscdm.2024.05.02.019

Article Info

Received: 2 June 2024
Accepted: 27 October 2024
Available online: 18 December 2024

Keywords

AdaBoost, ensemble algorithm, neural network, backpropagation, boosting algorithm

Abstract

Within the healthcare segment, Artificial Intelligence applications have made diagnosis much better through early problem detection with much ease, accurate diagnosis, and tailored treatment plans. Apart from effective medical action, these advantages come with improved patient outcomes. The integration of many benefits in different models through the ensemble methods improved these advantages thus offering even higher accuracy and dependability.In this paper we propose an ensemble technique known as ABNet algorithm that combines the resilience of the AdaBoost algorithm with the agility of neural networks so as to increase classification accuracy. The algorithm implements the traditional architecture of neural networks, utilizing backpropagation for training through multiple layers of differentiable modules. Within this method the weights allocated to individual neural networks are influenced by their accuracy, with more accurate networks acquiring higher weights. This adjustability enables AdaBoost to highlight previously misclassified instances, effectively directing the neural networks to learn from their errors. Our startegy, substantiated on three datasets, show sturdiness and generalization, centerd on emphasizing the model's adaptability within various diagnostic disease datasets.

1. Introduction

Compared to using a single model, ensemble methods combine many models to improve the prediction accuracy and flexibility. By using diverse models, ensembles can alleviate bias and variance, resulting in more reliable outcomes. Permitting the ensemble to address the individual model weaknesses, the well-known approaches like voting or averaging take account of combining model outputs. Ensemble methods have been used extensively in health care, finance, agriculture [1] [2] etc. owing to their capability to increase performance and convey more precise and steadfast predictions.

One of the popular ensemble-learning techniques in machine learning is boosting that improves the model accuracy by combining multiple weak learners changing into a strong learner. Sequentially, the models are trained. Each new model focuses on adjusting errors created by the previous ones. AdaBoost is regarded as one of the preferred and adaptive boosting algorithms, which starts by allocating equal weights to all training samples. It trains a weak learner then evaluates its errors. After that it increases the weights of misclassified samples to put emphasis on them in following iterations. This process is repeated, and each model's role is determined by its accuracy. AdaBoost is robust against overfitting and improves prediction accuracy.

Influenced by the human brain, neural networks [3], [4] are machine learning models that are composed of interconnected layers of nodes known as neurons. They comprise of an input layer, hidden layers, and an output layer. Each neuron in the network processes data by means of weighted connections and then applies the

activation function to introduce the non-linearity that enables complex pattern learning. The training in neural networks involves the forward propagation [5] and backward propagation. The former is a step where the data passes through the network to generate the predictions whereas in latter weights are adjusted based on the errors so as to minimize a loss function [6]. Mostly used for image recognition, natural language processing, and speech recognition, neural networks achieve high accuracy due to their aptitude to learn intricate patterns from large datasets, but they need large data and computational power. Attributable to their complex internal workings, they are often thought of as black boxes.

Neural networks do not display peak performance when applied to tabular data. Consequently, tree-based models are habitually preferred in these scenarios [7]. Tree-based models, by nature, produce piece-wise functions, providing them with better interpretability in contrast with neural networks [8]. They also aid as effective tools for understanding how various features affect the dependent variable. Many researchers have explored the fusion of neural networks with the tree models to construct ensembles. The combination of boosting with neural networks improves model performance considerably as detailed in literature. The results have highlighted an enhanced accuracy, predominantly for tabular datasets [9], emphasizing the efficacy of this method in disease diagnosis and prediction datasets [10]. Using the iterative correction of errors and emphasizing upon the indeterminate instances, this approach increases accuracy, decreases overfitting, and improves robustness, particularly in noisy or imbalanced datasets. Boosting helps the neural networks to converge more proficiently [11], resulting in enhanced generalization and likewise helps in feature selection by focusing upon the most pertinent features [12], which makes the model extra adaptive and responsive. On the other hand, to achieve the optimal results, boosted neural networks require careful hyperparameter and it can also be computationally intensive. When it is about accuracy and sturdiness, this combination is greatly effective. In this domain, our research further attempts to contribute to the exploration of this promising research area. We propose ABNet (Adaptive Boosted Neural Network) algorithm that combines the resilience of neural networks in conjunction with the adaptive learning capabilities of the AdaBoost algorithm to improve classification accuracy. The novelty lies in the dynamic adaptation of weights during each iteration focusing on the importance of indeterminate instances. The paper is divided into seven sections. Section 1 gives the introduction, Section 2 gives the review of literature, and Section 3 gives the methods used in this research work followed by Section 4 that gives the results of the research work. Section 5 provides the discussion; Section 6 gives the conclusion and Section 7 follows with the limitations and future work of the research study.

2. Related Work

Deep ensembles, both together with and separately from neural networks, show favorable performance in classification tasks, mostly when tested on disease datasets. This twofold approach displays their adaptability and efficacy in contributing to precise and robust disease classification. Over the years, advanced machine learning models have seen numerous innovative approaches for the classification of disease datasets. In [13] (2010), the authors introduced a Weighted Random Forest that assigned weight built on tree classification to fine-tune the tree influence on accuracy. A weight-adjusted voting algorithm was proposed by [14] (2011), which emphasized upon tree efficiency through weight consignment to decision trees. [15] (2011) highlighted the importance of ensemble methods like Random Forest and XGBoost in tabular data solutions owing to their consistency and sturdiness, often starring in winning solutions. [16] (2014) introduced Neural Decision Forests by combining the decision tree comprehensibility with the features of neural network thereby providing an exclusive method to innovative feature learning. [4] (2016) presented convolutional layers for tabular data, augmenting prediction accuracy by capturing complicated patterns. gcForest (Deep Forest) was developed by [11] (2017) which is a multi-layer ensemble of decision tree ensembles that surpasses with restricted scale data and overtakes many approaches through simple and direct training. Deep Neural Decision Tree was proposed by [17] (2018) that aligned the weights with specific decision trees to improve model complexity and flexibility. Transfer learning with Decision Forests (DF) was explored by [2] (2018). The authors used quadratic optimization to decide weights for a weighted average, substituting standard averaging. [8] (2021) pioneered with ConvXGB, employing convolution layers for encoding and feature map inference in XGBoost trees. The combination of CNN with XGBoost for breast cancer data classification proposed by [18] (2021), improved the accuracy by capitalizing on CNN's pattern recognition and boosting competences. TabNet developed by [19] (2021) is a deep learning model for tabular data that utilizes sequential attention for selection of features. The authors were able to demonstrate that their proposed model outperforms other models. XBNet developed by [20] (2022) is a distinguishing approach where trees at each layer are integrated with gradient descent-based weight adjustments to improve model performance.

Inspired by the success of integration of neural networks and tree-based models for developing improved prediction methods, we consider a boosted neural network ensemble approach for classification of tabular disease datasets. The assimilation of these approaches is inspired by the neural networks' capability to model difficult data patterns and tree-based models' efficient handling of feature interactions.

J. of Soft Computing and Data Mining Vol. 5 No. 2 (2024) p. 264-273

3. Methods

3.1 ABNet Architecture

The ABNet algorithm combines the robustness of neural networks with the flexible learning abilities of the AdaBoost algorithm to improve classification accuracy. Primarily, the algorithm allocates equal weights to entire data points and trains a neural network on this equally weighted dataset. In succeeding boosting iterations, the algorithm concentrates on refining the classification of formerly misclassified samples by adjusting their weights.

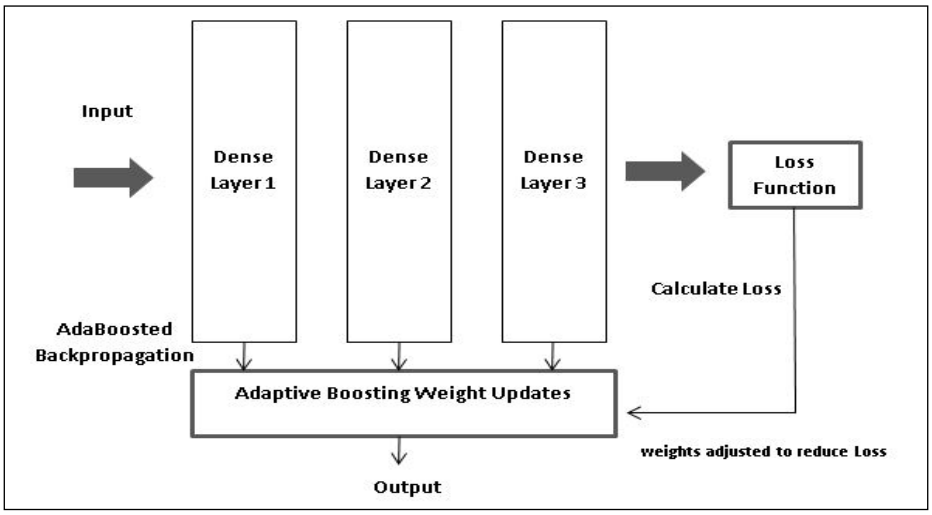


Fig. 1 *Proposed ABNet architecture*

It attains this by training extra neural networks and assigning higher weights to misclassified instances while decreasing weights for correctly classified ones. Fig. 1 provides the architecture of ABNet.

The significant innovation lies in the dynamic adaptation or revision of weights throughout each iteration, highlighting the importance of indeterminate instances that are difficult to classify. The weights designated to individual neural networks are dictated by their accuracy, with more accurate networks getting higher weights. This adaptableness permits the algorithm to learn based on its mistakes and uninterruptedly refine its predictive capabilities. The final or ultimate ensemble is a blend of these neural networks, where every single network contributes based on its singular performance. This method proves valuable in scenarios where conventional neural networks may contend with certain data patterns. By including AdaBoost's adaptive weighting system, the ABNet targets to produce a robust ensemble model that surpasses in handling complex datasets, eventually improving overall classification accuracy.

3.1.1 Gradient Descent Update Rule

Adaptive Boosted Neural Networks (ABNet), the fundamental gradient descent update rule remains unchanged like the standard gradient descent algorithm. The common update rule for a parameter θj in gradient descent is given by:

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{1}$$

$$\theta = \theta - \alpha . \nabla J(\theta) \tag{2}$$

where θ_j is the j^{th} parameter (weight and bias) that is being updated, θ represents the model parameters, α is the learning rate J(θ) is the loss function and $\nabla J(\theta) = \frac{\partial J(\theta)}{\partial \theta_j}$ is the derivative of the loss function with regard to j^{th} parameter.

The update equation for Stochastic Gradient Descent (SGD) within the framework of neural networks is characteristically expressed as follows:

$$\theta_{t+1} = \theta_t - \alpha \nabla J\left(\theta_t; x^{(i)}, y^{(i)}\right) \tag{3}$$

where θ_t represents the parameters at time step t, $J(\theta_t;\ x^{(i)}, y^{(i)})$ is the loss function that calculates the difference between the predicted output and the actual output for a particular training example $(x^{(i)}, y^{(i)})$, $\nabla J(\theta_t;\ x^{(i)}, y^{(i)})$ is the gradient of the loss with regard to the parameters θ_t. The update is accomplished for each mini-batch or individual training example, where *i* is used to denote the current training example.

The gradient descent equation remains unchanged, the exceptional characteristic lies in how these updates relate with AdaBoost's adaptive weighting mechanism during the training iterations.

3.1.2 Weight Update in ABNet

a) Backpropagation Update

A very important step in the neural network training is the backpropagation [8], where the error acquired from forward propagation is transmitted backward through the network layers so as to adjust the weights and improve or optimize the model. The weights in standard backpropagation are updated by means of the gradient of the loss regarding the weights. Let L be the loss function, then

$$w_{bp\_new} = w_{bp\_old} - \eta \frac{\partial L}{\partial w_{bp\_old}} \tag{4}$$

where w_{bp_new} is the updated weight after the backpropagation, w_{bp_old} is the current weight, η is the learning rate and $\frac{\partial L}{\partial w_{bp\_old}}$ is the derivative of loss with respect to the weight.

The weight update equation for SGD is given by:

$$w_{sgd\_new} = w_{sgd\_old} - \eta \nabla L(w_{sgd\_old}) \tag{5}$$

where w_{sgd_new} is the updated weight after the SGD update, w_{sgd_old} is the current weight, η is the learning rate, $\nabla L(w_{sgd\_old})$ is the gradient of the loss with respect to the weight.

b) AdaBoost Weight Update

The weight assigned to a data point is adjusted during each iteration of AdaBoost. This adjustment involves a decrease if the point is correctly classified by the model produced in that iteration and an increase else. The weights are updated as reported by the misclassification error as

$$w_{ab_{new}} = w_{ab_{old}} \times \exp(\alpha_t \,.\, y_{true} . y_{pred}) \tag{6}$$

where $w_{ab_{new}}$ is the updated weight after AdaBoost update, $w_{ab_{old}}$ is the current weight, α_t is the weight llotted to the neural network by AdaBoost, y_{true} is the true label of the sample, and y_{pred} is the predicted output of the neural network.

3.1.3 Proposed Combined Update

The blend of the updating incorporates the impact of both backpropagation and adaptive boosting, in improving or updating the weights of the neural network. On combining, the final weight is given by:

$$W_{final} = w_{bp\_new} \times w_{ab\_new} \tag{7}$$

$$w_{final} = w_{sgd\_new} \times w_{ab\_new} \tag{8}$$

Equation 7 and Equation 8 is used for updating weights intended for gradient descent and stochastic gradient descent respectively. This step is where the boosting mechanism affects the weights of misclassified and correctly classified samples. This combination permits each sample's final weight to be determined by both the backpropagation method and the AdaBoost method, displaying the adaptive learning methodology of ABNet. Normalization of weights is applied to make sure that the steadiness and convergence of the algorithm is maintained. Moreover, the hyperparameters that need to be tuned during training are the learning rate η in the backpropagation update and the AdaBoost weight α_t.

The insertion of 0.5 in the formula is a scaling feature to guarantee that the weight is suitably adjusted. This scaling aids in attaining a balanced effect of the neural networks, avoiding any single network from ruling the ensemble. The algorithm for ABNet is given in Fig. 2.

1. Initialization
Equally initialize the weights for the data points
2. The Boosting Iterations
For each boosting iteration (t):
- On the weighted data train, a new neural network
- Find out the error of the new neural network.
- Based on its error, compute neural network's weight in the final ensemble.
- Use AdaBoost to update data point weights
- Increase the weights of the misclassified instances.
- Reduce the weights of the accurately classified instances.
- Normalize the weights to ensure that they total up to 1.

3. Repeat
Repeat step 2 until the performance criterion of neural network is met (Early Stopping).
4. Combine the Neural Networks
- From each boosting iteration, merge the neural networks to create the final adaptive boosted neural network
- Determine each neural network's contribution weighted by its AdaBoost-calculated performance.

Fig. 2 *Algorithm for ABNet architecture*

3.2 Experimental Setup

3.2.1 Dataset

We have acquired three commonly used datasets, Indian Liver Patient Dataset (ILPD) [21], Pima Indians Diabetes Database (PIMA) [22], and Cleveland Heart Disease (CHD) [23] dataset from Kaggle, a platform that is known for its huge collection of datasets, used for data science and machine learning. A short description of ILPD, PIMA and CHD datasets are presented in the Table 1.

Table 1 *Dataset description*

Dataset	No. of Records	No. of features	Missing Value [%]	No. of Diseased Records	No. of Normal Records
ILPD	583	9	0.1%	416	167
PIMA	768	8	0%	500	268
CHD	303	13	0%	165	138

3.2.2 Implementation Details

We implemented our proposed model in a Jupyter Notebook using the Scikit-learn and Tensorflow/Keras library. We utilized the MICE (Multivariate Imputation by Chained Equations) method to handle missing values by importing critical libraries, loading the datasets and then we implemented the MICE technique using scikit-learn's *IterativeImputer* class. Also, the model is implemented by means of a base neural network model, and the AdaBoost algorithm is used for boosting. The optimal settings that we used are 50 estimators in the adaptive boosting with decision stump as the base model, SAMME.R (Stage-wise Additive Modeling using Multi-class Exponential loss function) as the boosting method. The learning rate for neural network is taken as 0.001 and the *tanh* activation with a threshold of 0 has been used for a smooth gradient.

The algorithm can be stopped either by setting the maximum number of boosting iterations in AdaBoost or else setting the maximum number of training epochs for the neural networks. In our case we have used the neural network epochs to stop the algorithm using early stopping that is used by implementing the *EarlyStopping* callback from *TensorFlow/Keras*.

The training testing ratio is stratified 4-fold at 70:30, where 70% of the data is reserved for training the model, and 30% is used for evaluating its performance. This is to ensure that the training as well as testing sets reflect the distribution of the original one to get the proper performance measure in imbalanced datasets.The stratified

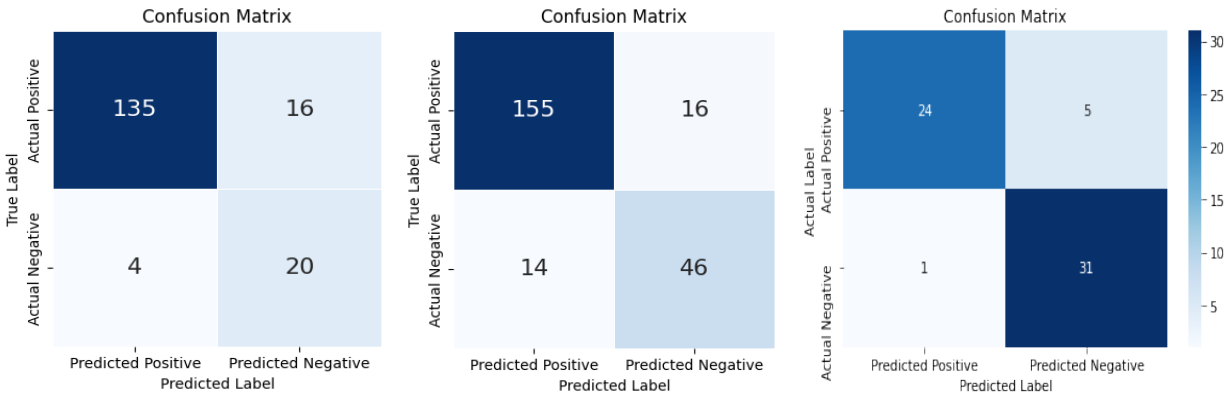
5-fold cross-validation has been implemented using scikit-learn's *StartifiedKFold* class that splits the data into 5 folds while keeping the distribution intact.

4. Result

The ultimate classification is obtained by aggregating the predictions of distinct neural networks, weighted by their respective final ensemble weight (α_i) values. This process safeguards that neural networks that have a better performance add more to the final ensemble. The output of the ABNet is the combination of these weighted prediction results. We have used the weighted average of the distinct predictions. Table 2 and Table 3 give the accuracy metrics and Matthews Correlation Coefficient (MCC) [43] for the three datasets obtained by applying conventional gradient descent and stochastic gradient descent in the framework of backpropagation in ABNet. In our case, the MCC values are greater than 0.5 in all three datasets taken, which specifies a very sturdy positive relationship between the prediction of the model and the actual values. Therefore, the model's predictive correctness is classifying the instances correctly. The coefficient also reveals that there is no random guessing, which means the algorithm captures the fundamental data patterns very efficiently. Fig. 3 shows the confusion matrix obtained for ABNet via traditional gradient descent in backpropagation.

Table 2 *Accuracy comparison for ABNet*

	ABNet with GD			ABNet with SGD		
	ILPD	**PIMA**	**CHD**	**ILPD**	**PIMA**	**CHD**
True Positive	135	155	24	137	158	25
False Positive	16	16	01	14	13	03
True Negative	20	46	31	17	49	29
False Negative	04	14	05	07	11	04
Precision	89.54%	90.64%	96.00%	90.73%	92.40%	89.29%
Recall	97.12%	91.72%	82.76%	95.14%	93.49%	86.21%
F1 Score	93.10%	91.18%	88.82%	92.88%	92.94%	87.68%
Accuracy	88.57%	87.01%	90.16%	88.00%	89.61%	88.52%



(a) **(b)** **(c)**

Fig 3 *Confusion Matrix for (a) ILPD; (b) PIDD; and (c) CHD Using ABNet with GD*

Table 3 *Matthews correlation coefficient values*

ABNet with GD			ABNet with SGD		
ILPD	**PIMA**	**CHD**	**ILPD**	**PIMA**	**CHD**
0.62	0.67	0.80	0.55	0.73	0.77

Table 4 shows the distribution of the records across each fold for stratified 5 folds used in cross-validated training.

Table 4 *Stratified 5-fold cross validation*

	Folds	Total Samples	Diseased Samples	Normal Samples
ILPD	Fold 1	117	83	34
	Fold 2	117	83	34
	Fold 3	117	83	34
	Fold 4	116	84	32
	Fold 5	116	83	33
PIDD	Fold 1	154	54	100
	Fold 2	154	54	100
	Fold 3	154	54	100
	Fold 4	153	53	100
	Fold 5	153	53	100
CHD	Fold 1	61	33	28
	Fold 2	61	33	28
	Fold 3	61	33	28
	Fold 4	60	33	27
	Fold 5	60	33	27

The *tanh* activation with a threshold of 0 leads to faster convergence while training because it is zero-centered. The learning rate for neural networks is 0.001, and the stratified 5-fold train-test split is used with 70% of the data for training so as to avoid overfitting. This ensured the training as well as testing sets agree with the distribution of the original one. This helps to get the appropriate performance measure in imbalanced datasets. Table 5 gives the best parameters found for multilayer perceptron after the hyperparameter tuning.

Table 5 *Optimal hyperparameter values*

	Hyperparameter	Value
Neural Network	Number of Hidden Layers	100
	Neurons in Hidden Layers	50
	Activation Function	tanh
	Learning Rate	constant
	Solver/Optimizer	GD, SGD
	Alpha	0.0001
	Epochs	Early Stopping
AdaBoost	Number of Estimators	50
	Base Model	Decision Stump
	Boosting Method	SAMME.R
	Loss Function	Exponential

5. Discussion

Dynamic weighting mechanism is the main advantage of incorporating AdaBoost into the training process of neural networks. It helps adjust the implications of different data points during each iteration. This adaptableness allows the algorithm to learn from its errors and uninterruptedly improve its predictive proficiencies. This adaptation of weights impacts ABNet's ability to handle complex datasets compared to conventional neural networks by continuously improving the focus on the difficult samples that contain the maximum of the informative patterns for decision-making. It adopts the adaptive learning approach, which makes it different from the conventional neural networks, which treat every sample equally. The integration of AdaBoost with the standard gradient descent update rule influences the learning process in ABNet by updating the decision boundary and trying to reduce the errors in each boosting iteration. This combination can prove beneficial when the data to be worked upon is noisy or imbalanced. The dynamic weight adaptation refines the model gradually and thus results in a robust and generalized model.

Moreover, we tried using a simpler neural network and early stopping to reduce the unnecessary computations. For faster convergence, training is adjusted using the zero centered activation function. We started

with the predefined setting i.e., the default values for the hyperparameters and then upgraded based on the results. Our algorithm effectually differentiates between the two classes of classification. The MCC value maintained by the algorithm divulges the making of precise predictions rather than the incorrect ones. The choice of using the early stopping and tanh activation function contributes to the model's convergence. The former stabilizes the learning process and prevents overfitting while the latter tries to introduce smooth gradient flow to avoid saturation. The challenges, however, faced during tuning these parameters include selecting the number of epochs before stopping and thresholding.

Table 6 *Comparative analysis with other proposed algorithms*

Year	Authors	Dataset	Classifier	Accuracy [%]
2020	Kuzhippallil et al. [24]	ILPD	XGBoost and Light GBM	86.00
2021	Sravani et al. [25]	ILPD	SVM	78.00
2022	Hassim et al. [26]	ILPD	MLP-BP	70.61
2021	Barik et al. [27]	PIMA	Hybrid XGBoost	74.10
2022	Harleen et al. [28]	PIMA	SVM Linear Model	89.00
2022	Tushar [20]	PIMA	XBNet	78.78
2016	Purushottam et al. [29]	CHD	Association Rules	86.70
2018	Viijayashree et al. [30]	CHD	Particle Swarm Optimization	88.22
2022	El-Hasnony et al. [31]	CHD	Multi label active learning	57.40

AdaBoost is famous for assigning a higher weight to misclassified samples in conventional machine learning; in this manner, it allows subsequent models to emphasize these challenging cases. When used for neural networks, AdaBoost allies with the standard backpropagation process, impelling the learning and adapting the models. AdaBoost improves the ensemble's flexibility and adaptability by repeatedly adapting weights and emphasizing the significance of unclassifiable samples. This flexibility becomes particularly valued when standard neural networks encounter difficulties with explicit data patterns. The collaboration of AdaBoost's adaptive weighting and the neural networks' proficiency to acquire sophisticated representations aims to create a robust ensemble capable of capturing difficult relationships inside the data, eventually enhancing general predictive accuracy.

Furthermore, the lower accuracy seen in SGD with normalization as opposed to normal Gradient Descent (GD) with normalization arises from the stochastic nature of SGD, which requires a very cautious learning rate, fine-tuning, and mini-batch changeability. The challenges can be addressed by first employing dynamic learning rates to quicken convergence. Batch normalization and data augmentation can be used to address SGD's sensitivity to hyperparameters. Early stopping and L1 and L2 regularization can reduce overfitting and enhance the model's reliability. In addition, challenges in normalization and hyperparameter sensitivity, accompanied by concerns of model complexity, influence the noted inconsistencies in accuracy. Cautious experimentation and hyperparameter tuning are vital to tackle these issues and increase the model performance using SGD and normalization. Table 6 compares our proposed model with other models trained on the same datasets.

6. Conclusion

The ABNet algorithm, a key focus of our research, associates the neural networks with the adaptive learning competencies of the AdaBoost algorithm. This exclusive approach proves to improve classification accuracy, as well as dynamically helping to adapt weights throughout the iterations. The approach focuses on challenging records and incessantly refines its predictive capabilities. The general result is a robust ensemble algorithm that is capable of handling complex datasets. Our approach's generalization ability and robustness are confirmed through performance assessment on the three databases. Together, these procedures offer a comprehensive approach to taking advantage of data in decision-making processes. Basically, this research work makes available valued insights and approaches for researchers as well as decision-makers in the difficult demand of healthcare analytics. Due to its sequential nature, adaptive boosting is computationally demanding, as the training involves a considerable time investment. In the future, we will try to fine-tune the parameters and use weighted loss functions to represent the class imbalance by assigning different weights to classes. We shall focus on using the hyperparameter optimization libraries, for instance, to automate and rationalize hyperparameter tuning. Furthermore, it is worth noting that our existing model exclusively relies on tabular data. As part of our future considerations, we plan to evaluate and adapt the model's performance to unstructured data. This expansion of the model's capabilities to handle diverse data types aligns with our goal to create a more versatile and adaptable solution.

Acknowledgement

The authors would like to appreciate the support of the Department of Computer Sciences, University of Kashmir.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm the contribution to the paper as follows: ***study conception and design:*** *Ifra Altaf;* ***data collection:*** *Ifra Altaf;* ***analysis and interpretation of results:*** *Ifra Altaf;* ***draft manuscript preparation:*** *Ifra Altaf;* ***supervision:*** *Manzoor Ahmad Chachoo. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Altaf, I., Butt, M. A., & Zaman, M. (2022). Disease detection and prediction using the liver function test data: A review of machine learning algorithms. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2021, Volume 2*. Springer Singapore.

[2] Utkin, L. V., & Ryabinin, M. A. (2018). A deep forest for transductive transfer learning by using a consensus measure. In *Artificial Intelligence and Natural Language: 6th Conference, AINL 2017, St. Petersburg, Russia, September 20–23, 2017, Revised Selected Papers 6*. Springer International Publishing.

[3] Abdi, H., Valentin, D., & Edelman, B. (1999). *Neural networks* (No. 124). Sage.

[4] Geng, Y., Tao, J., Zheng, C., Wu, M., Lu, Y., & Sun, X. (2016). Learning convolutional neural network to maximize pos@ top performance measure. *arXiv preprint arXiv:1609.08417*.

[5] Altaf, I., Butt, M. A., & Zaman, M. (2021). A pragmatic comparison of supervised machine learning classifiers for disease diagnosis. In *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE.

[6] Fayaz, S. A., Zaman, M., Kaul, S., & Butt, M. A. (2022). How M5 model trees (M5-MT) on continuous data are used in rainfall prediction: An experimental evaluation. *Revue d'Intelligence Artificielle, 36*(3), 409.

[7] Alexandropoulos, S.-A. N., Malliaros, F. D., Panagopoulos, A., Nikolopoulos, V., & Giannakis, G. B. (2019). A deep dense neural network for bankruptcy prediction. In *Engineering Applications of Neural Networks: 20th International Conference, EANN 2019, Hersonissos, Crete, Greece, May 24-26, 2019, Proceedings 20*. Springer International Publishing.

[8] Thongsuwan, S., Sanongboon, W., Ruangchaijatupon, N., Angkititrakul, A., & Boonpook, W. (2021). ConvXGB: A new deep learning model for classification problems based on CNN and XGBoost. *Nuclear Engineering and Technology, 53*(2), 522-531.

[9] Banday, I. R., Zaman, M., Quadri, S. M. K., Fayaz, S. A., & Butt, M. A. (2022). Big data in academia: A proposed framework for improving students' performance. *Revue d'Intelligence Artificielle, 36*(4).

[10] Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A., & Malkov, Y. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems, 34*, 18932-18943.

[11] Zhou, Z.-H., & Feng, J. (2019). Deep forest. *National Science Review, 6*(1), 74-86.

[12] Mayr, A., Fenske, N., Hofner, B., Kneib, T., & Schmid, M. (2014). The evolution of boosting algorithms. *Methods of Information in Medicine, 53*(6), 419-427.

[13] Li, H. B., Gao, J., Zheng, X., Zhang, J., & Wang, H. (2010). Trees weighting random forest method for classifying high-dimensional noisy data. In *2010 IEEE 7th International Conference on E-Business Engineering*. IEEE.

[14] Kim, H., Kim, J., Kim, K., Lee, C., & Park, H. (2011). A weight-adjusted voting algorithm for ensembles of classifiers. *Journal of the Korean Statistical Society, 40*(4), 437-449.

[15] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research, 12*, 2825-2830.

[16] Rota Bulo, S., & Kontschieder, P. (2014). Neural decision forests for semantic image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[17] Yang, Y., Garcia Morillo, I., & Hospedales, T. M. (2018). Deep neural decision trees. *arXiv preprint arXiv:1806.06988*.

[18] Sugiharti, E., Kuswanto, W., Syakur, M. A., Handoyo, E., & Prasetya, A. P. (2021). Convolutional neural network-XGBoost for accuracy enhancement of breast cancer detection. *Journal of Physics: Conference Series, 1918*(4). IOP Publishing.

[19] Arik, S. Ö., & Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, 35*(8).

[20] Sarkar, T. (2022). XBNet: An extremely boosted neural network. *Intelligent Systems with Applications, 15*, 200097.

[21] Indian Liver Patient Dataset. Retrieved from https://www.kaggle.com/datasets/rahulrajpandey31/ilpd-indian-liver-patient-dataset-data-set.
[22] Pima Indians Diabetes Database. Retrieved from https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.
[23] Heart Disease Cleveland Dataset. Retrieved from https://www.kaggle.com/datasets/ritwikb3/heart-disease-cleveland.
[24] Kuzhippallil, M. A., Joseph, C., & Kannan, A. (2020). Comparative analysis of machine learning techniques for Indian liver disease patients. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE.
[25] Sravani, K., Mukesh, A., Sharma, P., Reddy, A., & Reddy, V. (2021). Prediction of liver malady using advanced classification algorithms. In *Machine Learning Technologies and Applications: Proceedings of ICACECS 2020*. Springer Singapore.
[26] Hassim, Y. M. M., Mazwin, N., Mohd, Z. M., Mansoor, I., & Ariffin, F. (2022). Firefly algorithm for functional link neural network learning. In *Recent Trends in Mechatronics Towards Industry 4.0: Selected Articles from iM3F 2020, Malaysia*. Springer Singapore.
[27] Barik, S., Pandey, A., Kumar, M., Ghosh, S., & Srivastava, A. (2021). Analysis of prediction accuracy of diabetes using classifier and hybrid machine learning techniques. In *Intelligent and Cloud Computing: Proceedings of ICICC 2019, Volume 2*. Springer Singapore.
[28] Kaur, H., & Kumari, V. (2022). Predictive modeling and analytics for diabetes using a machine learning approach. *Applied Computing and Informatics, 18*(1/2), 90-100.
[29] Saxena, K., & Sharma, R. (2016). Efficient heart disease prediction system. *Procedia Computer Science, 85*, 962-969.
[30] Vijayashree, J., & Sultana, H. P. (2018). A machine learning framework for feature selection in heart disease classification using improved particle swarm optimization with support vector machine classifier. *Programming and Computer Software, 44*, 388-397.
[31] El-Hasnony, I. M., Mohamed, H. A., Abdalla, S., Ali, H. M., & Abdel-Kader, R. F. (2022). Multi-label active learning-based machine learning model for heart disease prediction. *Sensors, 22*(3), 1184.