2021 International Conference on Intelligent Technologies (CONIT)
Karnataka, India. June 25-27, 2021

Analyzing and Evaluating Boosting-Based CNN Algorithms for Image Classification

Masrafi Rahman
Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh
masrafi.rahman.bd@gmail.com

Raiyan Prodhan
Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh
raiyanrashidprodhan@gmail.com

Yeasir Shishir
Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh
subroshishir0@gmail.com

Shamim Ripon
Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh
dshr@ewubd.edu

***Abstract*— CNN is the most widely used method for image classification. In many occasions, its accuracy is improved by combining it with several boosting techniques. Determining such an efficient boosting model can be very time conserving and benevolent for many image classification applications. This study aims at analyzing and evaluating the performance of CNN added with various boosting techniques, namely XGBoost, Gradient Boost and AdaBoost. At first, CNN mode is trained, and image features are extracted by using convolutional layers, then for training the boosting algorithms intermediate data is extracted from three different layers of the CNN model. After comparing the performance of each individual techniques, it is revealed that boosted CNNs overperforms the others. While comparing the performances of the three CNN-Boosting algorithms, using data extracted from the flatten layer and the first dense layer, CNN-Gradient Boost performs better with an average of approximately 0.1921% more test accuracy and 0.2455% more F1 score for the flatten layer and 0.0505% more test accuracy and 0.0479% more F1 score for the first dense layer. For the third dense layer, CNN-XGBoost achieves better result with 0.0505% more test accuracy and 0.0479% more F1 score than the others. This finding is beneficial to improve performance of any application requiring image classification.**

Keywords*— *CNN, XGBoost, Adaboost, Gradient boost, image classification, Boosted-CNN.

I. INTRODUCTION

Classification of various categories of images is an easy task for the human eyes, but in terms of computers, it is considerably a complex task[1]. For that very reason, image classification is one of the most important phenomena in the arena of computer vision. Image classification is used in many applications like object detection, vehicle automation, Optimizing Medical Imagery etc. [1]. There are many existing algorithms that is used to do image classification. As it is a heavy task, computational performance has to be taken into consideration [2].

Convolutional Neural Networks (CNN) is one of the most efficient methods for image analysis having a higher accuracy than the other algorithms[3][4]. It is a specialized neural network for processing data that has an input shape like a 2D matrix like images. Many researchers have used it for solving various problems. Fabio Alexandre Spanhol et al. [14] used CNN for breast cancer histological image classification. Where they used BreaKHis dataset and showed that they can use CNN's architecture on Alexnet [14]. Le Kang et al. worked on document image classification using CNN. On this research they were able to do genre classification of each document that they used for experiment. There were ten classes of documents in the dataset [15].

A CNN model is a combination of a certain amount of convolution layers and a fully connected dense layer. The convolution is for extracting features of the input images and the fully connected feedforward neural network is for classifying the category of the given image [4][5].With a large dataset, CNN gives more accuracy than the others, while a small dataset may result otherwise [6].

"Boosting" is a general method for improving the performance of any learning algorithm and tree boosting is a widely used method [7] in the machine learning community. There are 3 types of boosting algorithm AdaBoost [8], Gradient Boost [9], Extreme Gradient Boost [10]. Gradient boost is a popular boosting algorithm which relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error [9]. XGBoost is the better version of gradient boost. It is an end-to-end tree boosting system that generate weak learners parallelly [17], whereas, Adaboost creates the weak learners sequentially [8].

Boosting algorithms are ensemble methods based on the idea of combining many weak learners into a strong learner for making highly accurate prediction or classifications [11]. CNN, on the other hand, has many parameters and layers and it takes a lot of time to tune its parameters for optimal result. Combining CNN with Boosting algorithms can accomplish better performance in terms of accuracy and classification [12]. It can give high accuracy even with imbalanced dataset [11]. CNN and boosting algorithms both are very powerful classifiers. In terms of image classification, combining both can be very promising [13].

The aim of this study is to evaluate the performance of image classification when CNN is combined with boosting algorithms. The objectives are – 1) Evaluate the performance of CNN, XGBoost, AdaBoost and Gradient Boost for classifying images. 2) Evaluate classification accuracy after combining individual boosting algorithms with CNN. 3) Compare performance of Boosted-CNN models with individual algorithms. 4) Identify the best model out of all the combinations.

II. Related works

Combining boosting algorithms with CNN to increase accuracy of image classification is a common practice in the machine learning field [13][18][19]. Laura Leal-Taix´ et al. [19] in their research applied a combination of Gradient Boost and CNN algorithm on the MOT15 dataset [20] and achieved 41% increase in classification performance compared to using only CNN. Elad Walach et al. [18] applied a combination of CNN and Gradient Boost algorithm on the Bacterial Cells Microscopy Images dataset [21] and the UCSD dataset [22]. For the Bacterial Cells Microscopy Images dataset their fine-tuned boosted CNN model generated 4.24 less mean absolute error than the ensemble of 7 CNNs for test dataset. For the UCSD dataset their fine-tuned boosted CNN model generated 0.43 less mean absolute error than the ensemble of 3 CNNs for test dataset.

Manqing Dong et al. [23] applied a Convolutional Autoencoder with Neural decision forest (CAN) and Gradient boost Convolutional Autoencoder with Neural decision forest (GrCAN) on the Epileptic Seizure Recognition Dataset [24] and Fashion-MNIST dataset [20]. For the Epileptic Seizure Recognition Dataset their GrCAN model generated 0.008% more test accuracy, 0.0065% more Precision, 0.0062% more recall and 0.0058% more F1-score than CAN. For the Fashion-MNIST dataset their GrCAN model generated 0.0051% more test accuracy, 0.0129% more Precision, 0.012% more recall and 0.0126% more F1-score than CAN.

Xudie Ren et al. [13] applied a combination of CNN and XGBoost algorithm on the MNIST [25] and CIFAR-10 [26] databases to show the increase in classification accuracy compared to the individual algorithms, and they achieved an increased accuracy of 0.42% for the MNIST dataset and 4.49% for CIFAR-10 dataset compared to using only CNN as a classifier. Xianghai Xu et al. [27] applied a combination of CNN and XGBoost algorithm on the Georgia Tech (GT) face database [28] and Olivetti Research Laboratory (ORL) face database [29]. For the Georgia Tech (GT) face database, their CNN-XGBoost model achieved 18% more recognition accuracy than CNN and 0.67% more accuracy than XGBoost. For the Olivetti Research Laboratory (ORL) face database their CNN-XGBoost model achieved 7.5% more accuracy than CNN and no accuracy increase than the XGBoost model. Liuwu Li et al. [30] applied a combination of CNN and XGBoost algorithm on a Social Media Prediction (SMP) dataset, and they achieved 2.7293 mean squared error (MSE), 1.2475 mean absolute error (MAE) which are all less than the MSE and MAE obtained from individual CNN and XGBoost.

Aboozar Taherkhani et al. [11] applied a combination of AdaBoost and CNN algorithm on five datasets: A synthetic dataset [31], CIFAR-10 [26], Fashion-MNIST [21], EMNIST (an Extended version of MNIST) [32], a Human Activity Recognition (HAR) dataset [33][34], and they were able to achieve 16.98% higher accuracy than the conventional AdaBoost with decision tree, and 2.03% higher accuracy than the conventional CNN for 10 estimators.

Xiaona Song et al. [35] applied a combination of CNN and AdaBoost algorithm on MIT Car dataset and Automobile Driver Recorder (ADR) dataset, and they were able to achieve 0.83% lower error rate for the CNN-AdaBoost model compared to the CNN model. Xuezhi Xiang et al. [36] applied a combination of CNN and AdaBoost algorithm on a test dataset containing a total of 144 hours of surveillance video, which was taken at two different gas stations, and they achieved 10.3% more precision rate for strong illumination, 13.3% more for general illumination, and 0.3% less recall rate for strong illumination, 0.2% less for general illumination during the daytime compared to only AdaBoost.

III. Dataset

The dataset used in this paper contains images of cats and dogs. The dataset consists total of 24946 images. It is a very balance dataset meaning exactly 50% of the dataset has the images of cats and other 50% has the images of dogs. All the images are of JPG type and labeled as n.jpg format, where n is the chronological integer. The images are real world pictures of different sizes that were taken from different types of devices. Some images are hard to categorize as a cat or dog as they contain cats and dogs or humans in the same picture, shown in Fig. 1. Some images were considered as garbage values and were avoided during model training. Which was essential for eradicating unwanted errors and low accuracy.

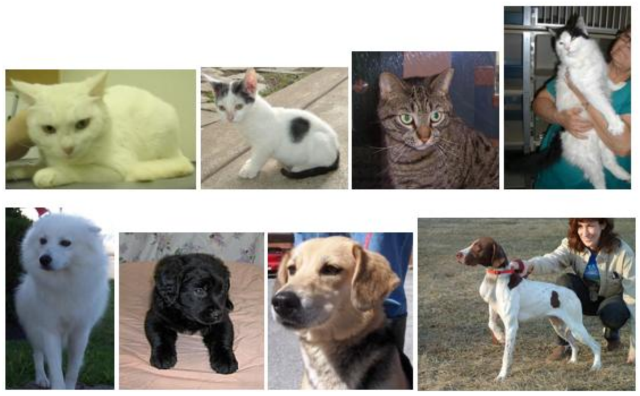
Fig. 1**.** Few samples from the dataset

Each image is set to $h \times w$ size, where h is the height and w is the width. The raw pictures are RGB pictures and has been converted to grayscale pictures for less computational expense. All the images pixel values which are $120\times120\times1$ matrix, is stored in a `NumPy` array. From that array data is separated for training and testing. 20,000 pictures are used for training and 4,946 for testing the models.

IV. Methodology

The proposed model consists of two phases: the convolutional layers extract the image features, and boosting algorithms classify the images. The Fig. 2 shows the complete structure of the model. For training the classifiers, data has been extracted from the flatten layer, the first dense layer and the third dense layer. In order to extract the intermediate data, the main CNN model needs to be trained first, as without the model being trained the weights and biases will be assigned with random values. Training the classifiers with these values will result in very low accuracy.

A. *Feature Extraction*

At the first step each features of the images are extracted before going any further. As shown in Fig. 2, there are 6 convolutional layers in total. The benefit of feature extraction in convolutional layer is that we can shorten the image values without losing valuable information. Each layer has the kernel size of 3×3 matrix which is the filter size. In addition, the stride is 1, which means that the convolutional filter will be moved by one cell from the left to right and from up to down. For the convolutional layer from 1 to 6 the number of filters is 32, 64, 128, 256, 256, 256 respectively. Each has the padding value as ‘`same`’. The picture that is being filtered is a 120×120 matrix m. When the padding size is ‘`same`’ the matrix m is padded with values of 0 on each side of the matrix making the matrix size 5×5. As each cell of the matrix contains a pixel value of the input image, the activation function “`relu`” (Rectifier linear unit) is used to control which cell or neuron is to be activated or used for the calculation during the filtration before sending the values to the next layer. After each convolutional layer batch normalization has been used to standardize the inputs that allows to use much higher learning rates and be less careful about initialization. After that, pooling layers take place. `MaxPool` has been used in here. The pooling filter size is a 2×2 matrix with the stride of 1. This filters the matrix generated by the conv layer and forwards the max values of the previous matrix. Next there is a dropout layer with the dropout value of 0.25 for avoiding overfitting problems in the model. After the filtration of 6 layers, a 1×1 matrix of 256 filters for each image is received. Now before sending the extracted image values to the dense layer, the outcome needs to be flattened. The flatten layer converts the outcome matrix into a single array value. The pseudocode in Table 1**,** the steps 1 to 22 show the inner working procedure for feature extraction of the images.

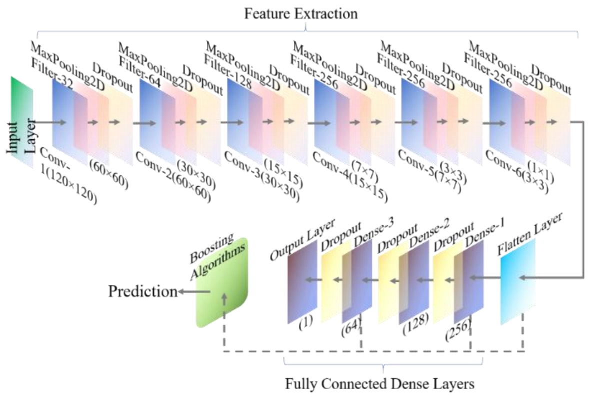

Fig. 2. The Structure of CNN-Boosting algorithm model

1) *Fully Connected Dense Layer*

The fully connected dense layer is a feedforward neural network of the normal CNN model that does the classification part. Fig. 2, shows that there are 4 layers in total in this model having 256, 128, 64, 1 nodes or neurons respectively. The last layer has 1 node because we have a binary dataset, and the outcome will be 1 or 0 denoting the image of a cat or dog. Each layer is presided by a dropout layer with a value of 0.5 for handling overfitting issues.

2) *Intermediate Data Extraction*

Fig. 2 shows that data has been extracted from 3 different places. Extracting data from the flatten layer gives the raw extracted feature values of the input images meaning no calculative relation with the dense layers. Data is also extracted from the first and third dense layer. Inside the CNN model, the first dense layer provides the non-randomly initialized weights at each node that has been assigned after passing data from the flatten layer. The third dense layer outputs the information that is used for classification which means it is the most compiled information inside this whole CNN model. When data is extracted from the first dense layer, there has been calculations made regarding weights & biases and assigned in its nodes. Therefore, when the layer is accessed, it provides the output produced by 256 nodes in the first dense layer. Extracting the values of the third dense layer gives more in-depth information of the nodes weights and the extracted value is less in amount. In Table 1, step 22, 23 and 27 shows the procedure of data extraction from these 3 layers.

B. *Classification with Boosting Algorithms*

After getting the extracted information, it is used for training the boosting algorithms. Each boosting algorithm is combined with CNN separately, but the procedure is same for the most part as shown in Fig. 3. The pseudocode in Table 1, shows that steps 1 to 33 is the procedure for feature extraction of the images, flattening the feature data and accessing it from 3 different layers. This work is same for each of the boosting algorithm.

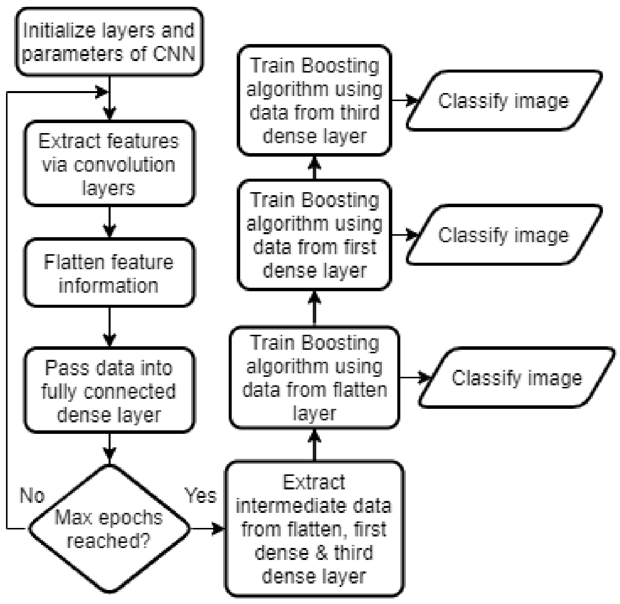
Initialize layers and parameters of CNN
Extract features via convolution layers
Flatten feature information
Pass data into fully connected dense layer
Max epochs reached?
No
Yes
Extract intermediate data from flatten, first dense & third dense layer
Train Boosting algorithm using data from flatten layer
Train Boosting algorithm using data from first dense layer
Train Boosting algorithm using data from third dense layer
Classify image
Classify image
Classify image

Fig. 3. Overall workflow procedures of Boosted-CNN

XGBoost algorithm is a parallelized algorithm that creates N number of different models parallelly and combines the outcomes to get the best outcome. The intermediate value has been passed to the XGBoost as a 2D array input. All the classifiers have 500 estimators meaning 500 different model is being created. For XGBoost the estimators are being created in parallel. The learning rate is 0.015 with the max depth of each tree of 7.

TABLE I. PSEUDOCODE FOR CNN-XGBOOST

```
BEGIN
Input: Data D & a learning method CNN with XGBoost
Output: Classification of cats & dogs images
Method:
   1.  Initialize classification labels with data D
   2.  Initialize numberOfLayers, numberOfFilters, filterSize,
       neuronPerLayer, numberOfWeights, activationFunction,
       numberOfEpochs, batchSize
   3.  Derive Model, model
   4.    For l=0 to numberOfLayers:
   5.        Get feature map via image filtration
   6.        Update values with batch normalization
   7.        MaxPool to reduce features
   8.        dropout (0.25)
   9.    End for
   10. accessDataFromFlattenLayer ← Flatten (outputData)
   11. accessDataFromFirstDenseLayer ← Calculate first
       dense layer's weights, biases based on input
   12. dropout (0.5)
   13. Calculate second dense layer's weights, biases based
       on input
   14. dropout (0.5)
   15. accessDataFromThirdDenseLayer ← Calculate third
       dense layer's weights, biases based on input
   16. dropout (0.5)
   17. outcome ← output layer
   18. trainDataFlattenLayer ←
       accessDataFromFlattenLayer.output().predict(D)
   19. trainDataFirstDenseLayer ←
       accessDataFromFirstLayer.output().predict(D)
   20. trainDataThirdDenseLayer ←
       accessDataFromThirdLayer.output().predict(D)
   21. trainData_layers ← [trainDataFlattenLayer,
       trainDataFirstDenseLayer, trainDataThirdDenseLayer]
   22. Initialize numberOfEstimators, objective, numOfClass
   23. Derive Model, xgbModel
   24. Classify testValue, using xgbModel
   25. For each extracted trainData_layers:
   26.     Fit training data
   27.     Initialize xgbModel with constant values
   28.     For i=0 to numberOfEstimators:
   29.         Minimize loss function
   30.         Update current model
   31.     End for
   32.     prediction = xgbModel.predict(testValue)
   33.     return prediction
   34. End for
END
```

For the AdaBoost classifier, the learning rate is 0.5. Table 2 shows the pseudocode of it. The minimum error is calculated based on the equation (1) and the value of α is received from equation (2). By this medium, the minimum error is calculated, and the model keeps on updating until the maximum number of estimators is reached.

$$err_m = \frac{\sum_{i=1}^{N} \omega_i I\big(\omega_i \neq C_m(x_i)\big)}{\sum_{i=1}^{N} \omega_i} \tag{1}$$

$$a_m = \log[1 - err_m/err_m] \tag{2}$$

Gradient Boost classifier has been trained with a 0.5 learning rate and max tree depth of 2. Table 3 shows that for each weak learner pseudo-reseduals are calculated which is based on equation (3). The multiplier is generated from equation (4).

$$\check{y}_{\pi(i)m} = -\left[\frac{\partial\psi\big(y_{\pi(i)},F(x_{\pi(i)})\big)}{\partial F(x_{\pi(i)})}\right]_{F(x)=F_{m-1}(x)} ,i = \check{N} \tag{3}$$

$$\gamma t_m = \underset{\gamma}{argmin} \sum_{x_{\pi(i)} \epsilon R_{im}} \psi\big(y_{\pi(i)}, F_{m-1}(x_{\pi(i)}) + \gamma\big) \tag{4}$$

The model gets updated until the max number of estimators are reached and the strong model is generated.

TABLE II. PSEUDOCODE FOR CNN-ADABOOST

```
BEGIN
   1.  Initialize numberOfEstimators, learning_rate
   2.  Derive Model, adaModel
   3.  Classify testValue, using adaModel
   4.  For each extracted trainData_layers:
   5.      Fit training data
   6.      For i=0 to numberOfEstimators:
   7.          train weak learners
   8.          Calculate minimum error
   9.          Choose α ← weights of the samples
   10.         Update weights
   11.     End for
   12.     Produce the strong classifier
   13.     prediction = adaModel.predict(testValue)
   14.     return prediction
   15. End for
END
```

TABLE III. PSEUDOCODE FOR CNN-ADABOOST

```
BEGIN
   1.  Initialize numberOfEstimators, learning_rate,
       max_features, max_depth
   2.  Derive Model, GBModel
   3.  Classify testValue, using GBModel
   4.  For each extracted trainData_layers:
   5.      Initialize GBModel with constant values
   6.      Fit training data
   7.      For i=0 to numberOfEstimators:
   8.          Compute pseudo-reseduals
   9.          Fit a base learner
   10.         Compute the multiplier $\gamma t$
   11.         Update the model
   12.     End for
   13.     prediction = GBModel.predict(testValue)
   14.     return prediction
   15. End for
END
```

For experimental purposes, each individual classifier is trained with the training dataset to observe their results. There are 4 individual algorithms that has been trained.

TABLE IV. COMPARISON OF BOOSTED-CNN MODELS IN EACH LAYER

Metrics	**CNN-XGBoost**			**CNN-Gradient Boost**			**CNN-AdaBoost**		
	Flatten Layer (%)	***First Dense Layer (%)***	***Third Dense Layer (%)***	***Flatten Layer (%)***	***First Dense Layer (%)***	***Third Dense Layer (%)***	***Flatten Layer (%)***	***First Dense Layer (%)***	***Third Dense Layer (%)***
Test Accuracy	92.195	92.155	92.317	92.357	92.357	92.276	92.135	92.114	92.256
F1 Score	92.179	92.142	92.323	92.342	92.379	92.286	92.127	92.124	92.261
Precision	93.582	93.506	93.456	93.747	93.319	93.379	93.431	93.216	93.412
Recall	90.818	90.818	91.217	90.978	91.457	91.217	90.858	91.057	91.137

V. RESULT ANALYSIS

Performance of each of the algorithms used in the study is measured on the basis of 4 metrics- test accuracy, F1 score, precision and recall.

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall= TPTP+FP \quad (6)$$

$$F1\ score = \frac{2\times precision\times recall}{precision+recall} \quad (7)$$

$$Test\ accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (8)$$

Here, TP, TN, FP, and FN are true positive, true negative, false positive, and false negative respectively. Fig. 4 shows the test accuracy, F1 score, precision and recall for CNN, XGBoost, AdaBoost and Gradient Boost. The figure represents the performance of these algorithms after training them with the source dataset. Here it can be seen that CNN has resulted the best performance among all the algorithms.

Table 4 shows the performance of the Boosted-CNN models. CNN-Gradient Boost outperforms both CNN-XGBoost and CNN-AdaBoost when the training data is extracted from the flatten layer and the first dense layer. It achieves an average of approximately 0.1921% more test accuracy and 0.2455% more F1 score for the flatten layer and 0.0505% more test accuracy and 0.0479% more F1 score for the first dense layer. For the third dense layer, CNN-XGBoost achieved better results with 0.0505% more test accuracy and 0.0479% more F1 score.

By colliding the results in Table 4 and Fig. 4**,** it can be seen that all the Boosted-CNN models of each of the layer, produces higher performance than the individual classifiers. Table 4 and Fig. 4 also shows that, by taking the average of the three layers, CNN-XGBoost produces approximately 31% more test accuracy and 33.5% more F1 score than XGBoost. CNN-Gradient Boost achieves 29% more test accuracy and 30% more F1 score than Gradient Boost. CNN-AdaBoost obtained 31% more test accuracy and 32.5% more F1 score than AdaBoost. By comparing the accuracy based on data extracted from each layer, Table 4 shows that, data extracted from the third dense layer helps to get the best performance for CNN-XGBoost. CNN-Gradient Boost gets the same accuracy when data is retrieved from the flatten and the first dense layer. In case of CNN-AdaBoost, it gets highest accuracy from the third dense layer.

TABLE V. AVERAGE PERFORMANCE OF BOOSTED-CNN MODELS IN COMPARISON TO CNN MODEL

Metrics	**CNN**	**CNN-XGBoost**	**CNN-Gradient Boost**	**CNN-AdaBoost**
Test Accuracy	92.114%	92.222%	92.33%	92.168%
F1 Score	92.24%	92.215%	92.336%	92.171%
Precision	91.947%	93.515%	93.842%	93.353%
Recall	92.534%	90.951%	91.217%	91.017%

Table 5 shows the comparison between Boosted-CNN models and normal CNN model. Here, average of the three layers has been considered for comparing the performance. In the table, it can be seen that all the Boosted-CNN models on average provide a higher test accuracy than CNN. CNN-XGBoost has 0.108% more test accuracy than CNN. Here, CNN-Gradient Boost has the highest increased test accuracy in comparison to CNN which is 0.142%. Finally, CNN-AdaBoost has 0.054% better test accuracy than CNN.

VI. CONCLUSION

The aim of this research has been set to evaluate the performance of different CNN-Boosting algorithm combinations and compare their performances for image classification. To reach that goal, the performance of CNN-Gradient Boost, CNN-XGBoost and CNN-AdaBoost models has been evaluated by assessing test accuracy, precision, recall and F1 score. These are trained by using data extracted from 3 different layers of the CNN model. It has resulted in CNN-Gradient Boost obtaining better performance with the data extracted from the flatten layer and the first dense layer. CNN-XGBoost has performed superior by using the data from the third dense layer. Additionally, all the Boosted-CNN models has resulted in higher accuracy than basic CNN model.

Although the study has achieved its main goal of evaluating the performance of various CNN-Boosting algorithm combinations and finding the best model amongst them for image classification, it can be extended further by evaluating more combinations of other classification and boosting algorithms. Combining other popular classification algorithms like, SVM, LSTM, KNN etc. with CNN can result in better performance. Also increasing the layers in the fully connected dense layer, will give more options to do experiments on intermediate data extraction.

REFERENCES

[1] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," Int. J. Remote Sens., vol. 28, no. 5, pp. 823–870, Mar. 2007, doi: 10.1080/01431160600746456.

[2] R. M. Haralick, I. Dinstein, and K. Shanmugam, "Textural Features for Image Classification," IEEE Trans. Syst. Man Cybern., vol. SMC-3, no. 6, pp. 610–621, 1973, doi: 10.1109/TSMC.1973.4309314.

[3] K. S. Sudeep and K. K. Pal, "Preprocessing for image classification by convolutional neural networks," in 2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2016 - Proceedings, Jan. 2017, pp. 1778–1781, doi: 10.1109/RTEICT.2016.7808140.

[4] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017, Mar. 2018, vol. 2018-Janua, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.

[5] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Nov. 2015, Accessed: Jan. 26, 2021. [Online]. Available: http://arxiv.org/abs/1511.08458.

[6] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," Pattern Recognit. Lett., vol. 141, pp. 61–67, Jan. 2021, doi: 10.1016/j.patrec.2020.07.042.

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 2016, vol. 13-17-Augu, pp. 785–794, doi: 10.1145/2939672.2939785.

[8] R. E. Schapire, "Explaining adaboost," in Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, Springer Berlin Heidelberg, 2013, pp. 37–52.

[9] J. H. Friedman, "Stochastic gradient boosting," Comput. Stat. Data Anal., vol. 38, no. 4, pp. 367–378, Feb. 2002, doi: 10.1016/S0167-9473(01)00065-2.

[10] T. Chen, T. He, and M. Benesty, "Extreme Gradient Boosting," arXiv, pp. 1–4, 2016, [Online]. Available: https://github.com/dmlc/xgboost.

[11] A. Taherkhani, G. Cosma, and T. M. McGinnity, "AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning," Neurocomputing, vol. 404, pp. 351–366, Sep. 2020, doi: 10.1016/j.neucom.2020.03.064.

[12] S. J. Lee, T. Chen, L. Yu, and C. H. Lai, "Image Classification Based on the Boost Convolutional Neural Network," IEEE Access, vol. 6, pp. 12755–12768, Jan. 2018, doi: 10.1109/ACCESS.2018.2796722.

[13] X. Ren, H. Guo, S. Li, and S. Wang, "A Novel Image Classi fi cation Method with CNN-XGBoost Model," pp. 378–390, 2017, doi: 10.1007/978-3-319-64185-0.

[14] F. A. Spanhol, L. S. Oliveira, C. Petitjean, and L. Heutte, "Breast cancer histopathological image classification using Convolutional Neural Networks," in Proceedings of the International Joint Conference on Neural Networks, Oct. 2016, vol. 2016-October, pp. 2560–2567, doi: 10.1109/IJCNN.2016.7727519.

[15] L. Kang, J. Kumar, P. Ye, Y. Li, and D. Doermann, "Convolutional neural networks for document image classification," in Proceedings - International Conference on Pattern Recognition, Dec. 2014, pp. 3168–3172, doi: 10.1109/ICPR.2014.546.

[16] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm DRAFT-PLEASE DO NOT DISTRIBUTE," 1996. Accessed: Mar. 16, 2021. [Online]. Available: http://www.research.att.com/orgs/ssr/people/fyoav,schapireg/.

[17] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 2016, vol. 13-17-August-2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[18] E. Walach and L. Wolf, "Learning to count with CNN boosting," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9906 LNCS, pp. 660–676, 2016, doi: 10.1007/978-3-319-46475-6_41.

[19] L. Leal-Taixé and K. Schindler ETH Zurich Zurich, "Learning by tracking: Siamese CNN for robust target association," 2016.

[20] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mniST: A novel image dataset for benchmarking machine learning algorithms," arXiv, pp. 1–6, 2017.

[21] V. Lempitsky and A. Zisserman, "Learning to count objects in images," Adv. Neural Inf. Process. Syst. 23 24th Annu. Conf. Neural Inf. Process. Syst. 2010, NIPS 2010, pp. 1–9, 2010.

[22] A. B. Chan, Z. S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," 2008, doi: 10.1109/CVPR.2008.4587569.

[23] M. Dong, L. Yao, X. Wang, B. Benatallah, and S. Zhang, "GrCAN: Gradient boost convolutional autoencoder with neural decision forest," arXiv, vol. 14, no. 8, pp. 1–12, 2018.

[24] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top., vol. 64, no. 6, p. 8, 2001, doi: 10.1103/PhysRevE.64.061907.

[25] L. Deng, "The MNIST database of handwritten digit images for machine learning research," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 141–142, 2012, doi: 10.1109/MSP.2012.2211477.

[26] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," Unpubl. Manuscr., pp. 1–9, 2010, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Convolutional+Deep+Belief+Networks+on+CIFAR-10#0.

[27] X. Xu, X. Wang, Z. Sun, and S. Wang, "Face recognition technology based on CNN, XGBoost, model fusion and its application for safety management in power system," IOP Conf. Ser. Earth Environ. Sci., vol. 645, p. 012054, 2021, doi: 10.1088/1755-1315/645/1/012054.

[28] R. Zeng et al., "Color image classification via quaternion principal component analysis network," Neurocomputing, vol. 216, no. 2, pp. 416–428, 2016, doi: 10.1016/j.neucom.2016.08.006.

[29] M. Yao and C. Zhu, "SVM and Adaboost-based classifiers with fast PCA for face reocognition," 2016 IEEE Int. Conf. Consum. Electron. ICCE-China 2016, pp. 0–4, 2017, doi: 10.1109/ICCE-China.2016.7849742.

[30] L. Li, R. Situ, J. Gao, Z. Yang, and W. Liu, "A hybrid model combining convolutional neural network with XGBoost for predicting social media popularity," MM 2017 - Proc. 2017 ACM Multimed. Conf., pp. 1912–1917, 2017, doi: 10.1145/3123266.3127902.

[31] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class AdaBoost," Stat. Interface, vol. 2, no. 3, pp. 349–360, 2009, doi: 10.4310/sii.2009.v2.n3.a8.

[32] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," Proc. Int. Jt. Conf. Neural Networks, vol. 2017-May, pp. 2921–2926, 2017, doi: 10.1109/IJCNN.2017.7966217.

[33] "Sensor HAR recognition App - File Exchange - MATLAB Central." https://uk.mathworks.com/matlabcentral/fileexchange/54138-sensor-har-recognition-app (accessed Feb. 06, 2021).

[34] "Human Activity Recognition Simulink Model for Smartphone Deployment - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/supportpkg/android/examples/human-activity-recognition-simulink-model-for-smartphone-deployment.html (accessed Feb. 06, 2021).

[35] X. Song, T. Rui, Z. Zha, X. Wang, and H. Fang, "The AdaBoost algorithm for vehicle detection based on CNN features," ACM Int. Conf. Proceeding Ser., vol. 2015-Augus, pp. 180–184, 2015, doi: 10.1145/2808492.2808497.

[36] X. Xiang, N. Lv, M. Zhai, and A. El Saddik, "Real-Time Parking Occupancy Detection for Gas Stations Based on Haar-AdaBoosting and CNN," IEEE Sens. J., vol. 17, no. 19, pp. 6360–6367, 2017, doi: 10.1109/JSEN.2017.2741722.