Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)
IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1

Handwritten Digit Recognition Using Ensemble Learning

Kuppa Venkata Padmanabha Nandan, Manoj Panda and S Veni
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India
nandankuppa07@gmail.com, mk_panda@cb.amrita.edu, s_veni@cb.amrita.edu

***Abstract*—In pattern recognition, the recognition of handwritten digits has always been a very challenging and tedious task. In this work, a simple novel approach is proposed to recognize the handwritten digits. The primary goal of this work is recognition of the handwritten digits by using ensemble learning. Ensemble learning improves convergence by decreasing the complexity of the model. The distribution of data in the random split and class-wise split for the base learners has been studied. Detailed analysis of how the load is distributed among the base learners and how it impacts the model accuracy and training time is also discussed. The overall trends of the ensemble model have also been analyzed.**

Keywords—Ensemble Learning, Deep Learning, Convolutional Neural Network (CNN), Digit recognition, Pattern Recognition

I. INTRODUCTION

In recent years, there is a huge and rapid increase in the advancement of technology. The human race is moving towards automation and digitalization in every sphere of life.

In pattern recognition, handwritten digit image recognition plays a vital role. The digit recognition has a wide range of applications like user identification, verification and handwritten digits on the bank's cheques, etc., In the past, various machine learning classifiers like linear regression classifier, Support Vector Machines (SVM) and Naïve Bayes had been used to classify handwritten pattern recognition [1] [2]. But the machine learning methods are only limited to small datasets. CNN is one of the powerful algorithms in many applications like image detection and classification. In recent years, by using CNN, the accuracy of the models has been significantly increased. CNN gives the highest accuracy when compared with other machine learning classifiers like SVM, Decision Trees (DT), and Random Forest Classifier (RFC). Due to its highest accuracy in performance metrics, CNN is being used in image classification, video analysis [3]. The distributed ensemble in CNN is implemented using Keras.

In this work, a novel approach is proposed to recognize the handwritten digits by using a technique called *ensemble learning*. The performance metrics achieved with ensemble learning are analyzed. Ensemble learning is one of the powerful methods to address the challenge of high computational load, which is normally encountered for training computationally large models. As the trained data increases, the performance of the model improves. However, when the training dataset increases in size, the computational load, and the training time of the model also significantly increases. Ensemble learning can address this problem.

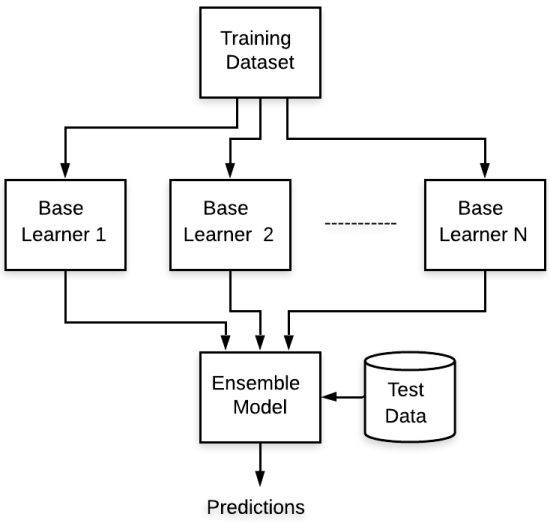

Fig. 1. Ensemble model

In Ensemble Learning, a training dataset is divided among the combination of N base learners (see Fig. 1) instead of a single model, where each base learner is a trained model. The training dataset gets split into N-base learners and each base learner is trained with a subset or a part of the original dataset (random split or class-wise split). The computational burden is reduced by increasing the base learners and the training time also reduces because the base learners can be simultaneously trained in parallel.

The main contributions of this paper is to propose a novel approach for recognition for handwritten digits using the Ensemble Learning (EL) techniques, which reduces the computation load on the model and also reduces the time for training of the model.

The remainder of the paper is organized as follows. Section II discusses the related works on the handwritten digit recognition and ensemble learning. The proposed method of the research work is discussed in Section III. The Results and Discussions are summarized in the Section. Section V contains the conclusion and future work of the paper.

II. RELATED WORKS

In recent years, advanced algorithms such as CNN and Generative Adversarial Network (GAN) [4] are being used for the training and learning to recognize the handwritten digits. But the real challenge for the researchers is to get better accuracy with low error. Hyeran Byun et al. [5] attempted the work using the SVM algorithm using the MNIST dataset and achieved 97.3% accuracy on the Test data. Dan Claudiu Ciresan et al. [6] has proposed the work with a Simple Neural network and backpropagation using the MNIST dataset and achieved a 0.7% error rate with an accuracy of 99.1%, but this process required a higher processor, high cost and it is time-consuming. Kumar et.al. [7] have illustrated the digit-recognition of handwritten digits or characters using the SVM classifier and the Multi-Layer Perceptron (MLP) Neural Network. They have used different kernels like linear, polynomial, and quadratic based SVM classifiers. Among all the three kernels, the linear kernel has given the best accuracy of 94.8%. Retno Larasati et. al. [8] illustrated that the ensemble neural networks and ensemble decision tree can classify the **MNIST (Modified National Institute of Standards and Technology database)** and **USPS (US Postal Service) dataset**, and achieved accuracy up to 84%. I. Jeena Jacob [9] has achieved an accuracy of 89.8% by using the Capsule Neural Networks (Caps-Net) for classifications of texts and digits. T. Siva Ajay [10] has also achieved a higher rate of accuracy by using CNN for digit-recognition of the handwritten digits. CNN and LeNet as its backbone architecture and obtained an accuracy of more than 98%.

The ensemble learning-based parallel and distributed algorithms for classification are the advanced algorithms to reduce the computational load [11]. A typical CNN model has parameters that are generally numbered in millions that need to be trained. The complex tasks require high memory, more training time and they are computationally very complex. Hence by using the distributed algorithms, the computation load, and the training time of the model can be reduced. The training on distributed models can be achieved either by model parallelism or by data parallelism. Although the accuracy on MNIST by CNN achieved is 99.1%, the process involved to achieve this accuracy includes high computation load and also the time required for training the model is very high.

III. PROPOSED METHOD

A. Dataset Preparation

The standard dataset for handwritten digits' classification is MNIST dataset. It is a derivative of the NIST dataset. The MNIST dataset has 70000 gray-scale images. The train set has 60000 gray-scale images and the test set has 10000 gray-scale images. Out of the 60000 gray-scale images in the training dataset are again divided into train set and validation set. The train set has 54000 gray-scale images whereas the validation set contains 6000 gray-scale images (10% of the training set). The dimensions of each image in the dataset is 28 pixels x 28 pixels.

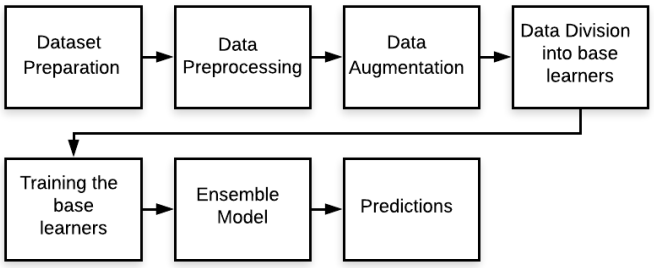

Fig. 2. Block Diagram of the Proposed Methodology.

B. Data Preprocessing

Before sending the dataset into the training process, the dataset prepared has to be preprocessed which includes reshaping of the image from RGB (3-channels) to gray-scale (1-channel), image equalization, and normalization. But the MNIST dataset already consists of gray-scale images, so only the preprocessing of reshaping and normalization are applied. to the entire training dataset. The preprocessing of reshaping is applied to all the images in the dataset, so the resolution of the images will be converted from 28 pixels x 28 pixels (28 x 28) into 28 pixels x 28 pixels x 1 color channel (28 x 28x 1). By applying the normalization to the pixel values of the grayscale images of the training set and validation set, the images will be rescaled from 0-255 range (0-black and 255-white) to 0-1.

The MNIST has 10 classes and each class is represented with a unique integer. By using the *to_categorical* function, the class element of each sample will be one hot encoded which transforms the integer into a 10 element binary vector with a 1 for the index of the class value, and 0 values for all other classes.

C. Data Augmentation

In recent years, the models in deep learning have been largely based on the diversity and quantity of the data. The data augmentation is a strategy which significantly increases the data for the models at the time of training, without actually collecting new data. The *ImageDataGenerator* class is the data augmentation library that is applied to the model from the Keras deep learning library. In this work, the model is provided with performing various data augmentation transformations like horizontal shifting (random width shifting in the range of 0.1 fractions of the total width of the image), vertical shifting (random height shifting in the range of 0.1 fractions of the total height of the image), scaling (randomly zooming in or zooming out in the range of 20% of the image), shearing (randomly shearing in the range of 0.1 fractions of the image), rotation (random rotate images in the range of 10 degrees) on the training dataset which is preprocessed. The data augmentation overcomes the model from overfitting which occurs due to the random noise.

D. Division of Dataset into Base Learners

After Data preprocessing and Data Augmentation, the training dataset is divided among the different models or base

learners in such a way that the dataset on each base learner almost has an equal number of samples (images) with their respective labels. The training dataset is divided into 2,3,4-base learner models.

1) Random Based Ensemble Model: In a randomly based division, the original training dataset will be equally divided among the various base learners irrespective of their classes.

a) 2 Base Learner random ensemble model: The original training dataset is divided among the two base learners. Each base learner will get 27000 images with labels for training.

b) 3 Base Learner random ensemble model: The original training dataset is divided among the three base learners. Each base learner will get 18000 images with labels for training.

c) 4 Base Learner random ensemble model: The original training dataset is divided among the four base learners. Each base learner will get 13500 images with labels for training.

2) Class-Based Ensemble Model: In a class-based division, the original training dataset is divided between the various base learners concerning their classes.

a) 2 Base Learner class-based ensemble model: The original training dataset is divided among the two base learners. The first base learner will be trained with the images having the labels that belong to the digit class 0,1,2,3,4 and the second base learner will be trained with the images having the labels that belong to the digit class 5,6,7,8,9.

b) 3 Base Learner class-based ensemble model: The original training dataset is divided among the three base learners. The first base learner will be trained with the images having the labels that belong to the digit class 0,1,2,3. The second base learner will be trained with the images having the labels that belong to the digit class 4,5,6 and the third base learner will be trained with the images having the labels that belong to the digit class 7,8,9.

c) 4 Base Learner class-based ensemble model: The original training dataset is divided among the four base learners. The first base learner will be trained with the images having the labels that belong to the digit class 0,1,2. The second base learner will be trained with the images having the labels that belong to the digit class 3,4, and the third base learner will be trained with the images having the labels belonging to the digit class 5,6,7. The fourth base learner will be trained with the images having the labels that belong to the digit class 8,9.

E. Training the Base Learners with CNN Model

A simple CNN model in Fig.3 is used to train the base learners. The CNN model architecture has an Input layer, three convolutional layers (feature extraction layers), a flatten layer, and dense layers (FC layers).

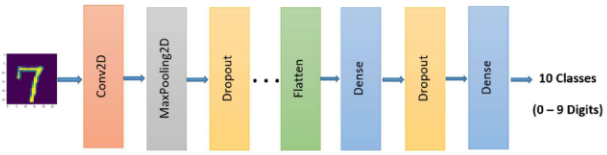

Fig. 3. Convolution Neural Network.

'ReLU' (Rectified Linear Unit) activation function is used at the end of each convolution layer as it is faster than alternative methods [12] and gives better performance than the Sigmoid and Tanh functions [13] [14]. The equation for ReLU is represented in (1),

$$f(x) = max(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \quad (1)$$

where, x = input

The Max-Pooling layer is added after the convolution layer. It is added to decrease the number of parameters of the output information coming from the convolution layer. The flatten layer converts the 2-Dimension featured map into a 1-Dimension feature vector. This layer comes after the pooling layer.

At the output layer, a Softmax activation function is used which is to represent the categorical split over labels of the class that it belongs to, and thereby getting the probabilities of each input belonging to that label. The Softmax activation function is generally used for multi-class tasks in which it returns the probabilities of each class with the corresponding highest probability target class. In almost all the output layers of the neural network architecture, a Softmax function is generally applied because they are popular for multivariate classification tasks. The equation for the Softmax function [15] is represented in (2),

$$f(x_i) = \frac{exp\ (x_i)}{\sum_j exp\ (x_j)} \quad (2)$$

The CNN model is trained using Adam as the optimizer and the default learning rate (lr) = 0.001 [16]. The default values of the Adam optimizer are $\alpha = 0.001$, $\beta 1 = 0.9$, $\beta 2 = 0.999$ and $\varepsilon = 10{-}8$ are used. The Dropout layer is added after the dense layer in the CNN, which reduces the complexity of the model and also prevents the model from overfitting. To prevent the model from under-fitting or overfitting, the hyper-parameters are tuned accordingly.

F. Ensemble Model

The trained base learner models are then combined into an aggregated ensemble model by Bagging. The Bagging is a special case of a model averaging approach. The model ensemble of trained base learner models is used to predict the handwritten digits from the test set. The prediction accuracies

of different models for the handwritten images from the test set are considered. The base learner which gives the higher prediction accuracy will be allowed to make the final call.

G. Predictions

The accuracy (f1-score) is evaluated from the predictions of the ensemble model with the original test set data. The highest prediction accuracy model from the ensemble model will take the decision.

IV. RESULTS AND DISCUSSIONS

The MNIST training dataset of 54000 gray-scale images is divided based on class label wise split and randomly split into two base learner models, three base learner model, and four base learner model.

For random based division, the overall original training dataset is divided equally among the various base learners irrespective of their classes.

For class-based division, the overall original training dataset is divided among the various base learners concerning their classes.

The base learners are trained using the simple CNN model in Keras. The CNN model is trained using Adam as the optimizer. The default learning rate of Adam optimizer is lr = 0.001. To prevent the model from overfitting, a dropout layer of value 0.3 is added after the pooling layer. To achieve good performance metrics, the CNN model is trained for 15 epochs. The trained base learner models are then combined into an aggregated ensemble model.

The training of the CNN model was done using the Google Colab or simply called **COLAB** which is a widely used computing platform that has eased the learning and development of machine learning applications. Google Colab provides a s**ingle 12GB NVIDIA Tesla K80 GPU** that can be used up to 12 hours continuously and it is free. It does not allow one model to train using multiple GPUs and hence the gain in training time is achieved by dividing the model into multiple base learner models rather than dividing the single model into multiple GPUs.

The training time and the performance metrics (accuracy) of the 2,3,4- base learner ensemble models are evaluated and observed.

The accuracy (f1-score) observed for the two base learners based ensemble system is 99% (random based split of training data) and 95% (class based split). For three base learners based ensemble system, the accuracy (f1-score) observed is 99% (random based split of training data) and 92% (class based split). For four base learners based ensemble system, the accuracy (f1-score) observed is 98% (random based split of training data) and 88% (class based split).

The accuracy (f1-score) and the training time of the ensemble model are plotted in Figs. 4-5 and are tabulated in TABLE I & II.

TABLE I. PERFORMANCE SUMMARY OF RANDOM BASED SPLIT ENSEMBLE MODEL.

Base Learners (in number)	Single Base Learner	Two Base Learners	Three Base Learners	Four Base Learners
F1-score	99%	99%	99%	98%
Time saved while training	0%	50%	65%	80%

TABLE II. PERFORMANCE SUMMARY OF CLASS-BASED SPLIT ENSEMBLE MODEL.

Base Learners (in number)	Single Base Learner	Two Base Learners	Three Base Learners	Four Base Learners
F1-score	99%	95%	92%	88%
Time saved while training	0%	50%	67%	78%

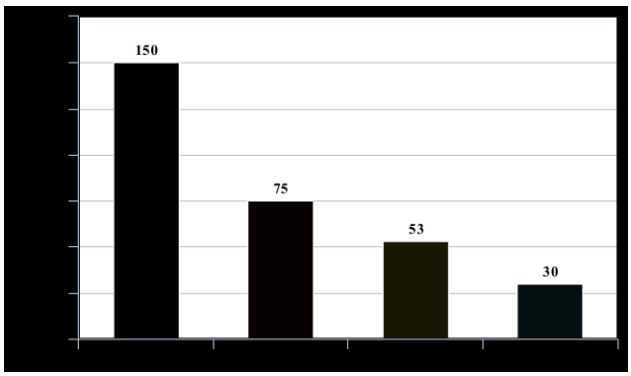

Fig. 4. Training Time (Random Split).

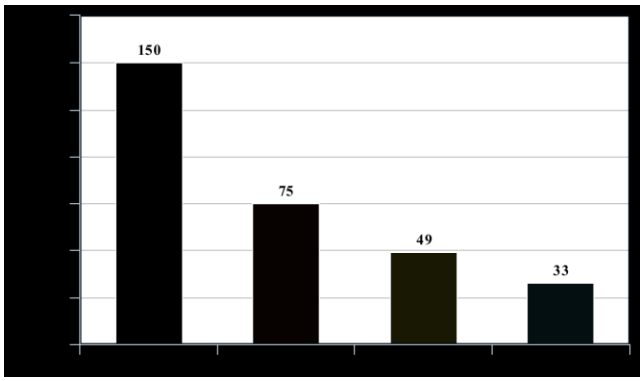

Fig. 5. Training Time (Class Wise Split).

When an ensemble model system is compared to that of a single learner model system, a significant time gain of 50% of training time is observed in both random split and class-wise split in two base learner model ensemble system.

In the three base learner ensemble system, a significant time gain of 65% of training time for randomly based split and 67% of training time gain for class wise split is observed when compared to the single learner model system.

In the four base learner ensemble system, a significant time gain of 80% of training time for randomly based split and 78% of training time for class wise split is observed when compared to the single learner model system.

Although the split or division of data is done among the increasing number of base learners, still the accuracy of the ensemble model is preserved in a random based split because each base learner is getting enough training data which contains all the class labels to train with a significant amount of time gain.

The accuracy is decreasing as the number of base learners increases in a class based split is because each base learner is not getting enough training data of all class labels to train. Hence for achieving a good performance of the model, it is important to have a balance between the reduction of load and also accuracy.

The confusion matrices of each ensemble model (two- base learner model, three base learner model, and four base learner model) concerning the test dataset are evaluated with the confusion matrix.

The confusion matrix for the Random division based ensemble models is plotted in Figs. 6-8.

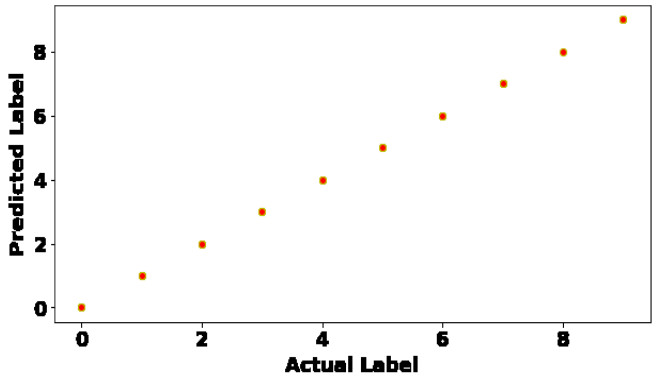

Fig. 6. Confusion Matrix with Two Base Learners (Random Split).

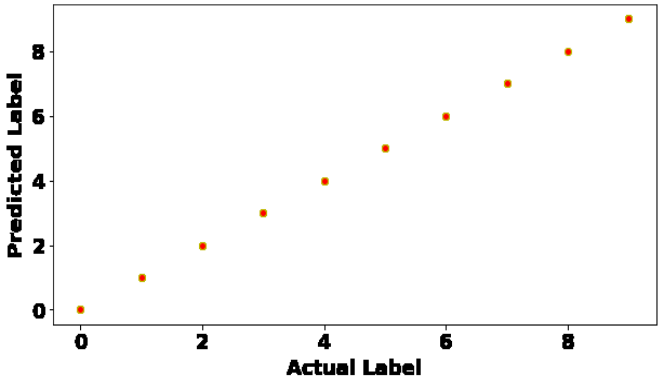

Fig. 7. Confusion Matrix with Three Base Learners (Random Split).

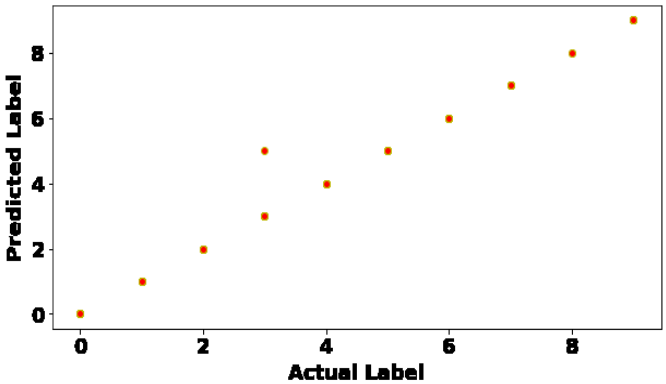

Fig. 8. Confusion Matrix with Four Base Learners (Random Split).

The confusion matrix for the Class-based ensemble models is plotted in Figs. 9-11.

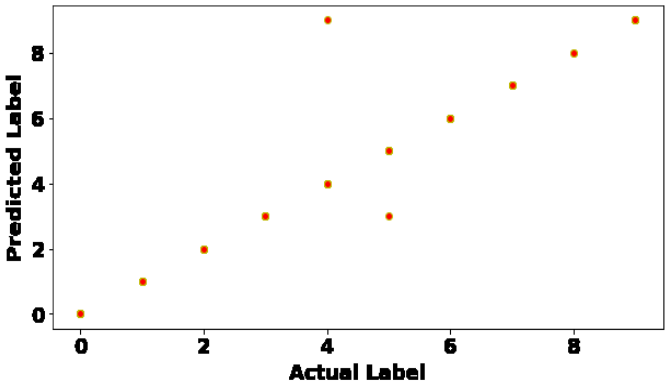

Fig. 9. Confusion Matrix with Two Base Learners (Class Wise Split).

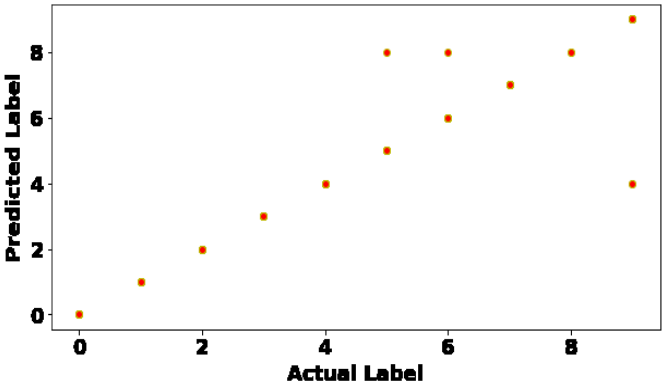

Fig. 10. Confusion Matrix with Three Base Learners (Class Wise Split).

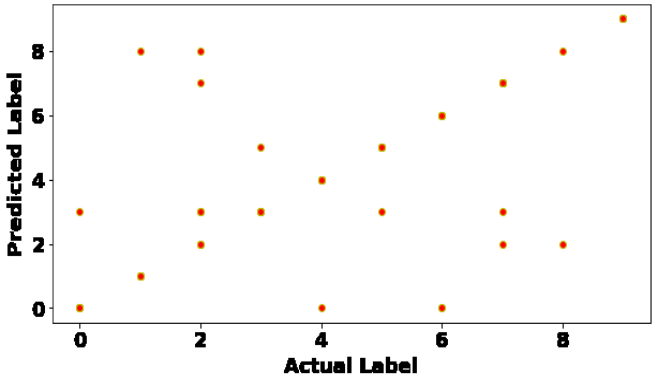

Fig. 11. Confusion Matrix with Four Base Learners (Class Wise Split).

V. CONCLUSION AND FUTURE WORK

By using ensemble learning, it has been observed that the training time and computational load on the system have been drastically reduced. As the data is distributed between the many numbers of base learner models in the ensemble model, the load complexity and training time has been gradually decreased. As the number of base learners is added to the distributed ensemble model, the training time gets reduced for each base learner as the load is shared between the base learners.

The training time is considered as an important factor while training a deep learning model. By using the ensemble models, significant time gains are achieved.

In the future, it is planned to implement a model ensemble with different variants of ensemble techniques such as Boosting, an active mixture of experts to further improve the performance metrics (accuracy) by giving less training data per base learner model. The training of the models will be explored with different deep learning networks and with different backbone architectures.

The paper explores ensemble learning for automotive applications such as driver emotion recognition and also on large object detection based datasets like Audi, Cityscapes, KITTI, etc. for speed bump detection and depth estimation.

REFERENCES

[1] Y. LeCun, C. Cortes, L. Bottou, and L. Jackel, "Comparison of Learning Algorithms for Handwritten Digit Recognition," Int. Conf. Artif. Neural Networks, pp. 53–60, 1995.

[2] M. Patil and S. Veni, "Driver Emotion Recognition for Enhancement of Human Machine Interface in Vehicles," 2019 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2019, pp. 0420-0424, doi: 10.1109/ICCSP.2019.8698045.

[3] F. Siddique, S. Sakib, and M. A. B. Siddique, "Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers," 2019 5th Int. Conf. Adv. Electr. Eng. ICAEE 2019, pp. 541–546, 2019, doi: 10.1109/ICAEE48663.2019.8975496.

[4] Patil, S.O., Variyar., V.V., & Soman, K.P. (2020). Speed Bump Segmentation an Application of Conditional Generative Adversarial Network for Self-Driving Vehicles. 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), 935-939.

[5] H. Byun and S. W. Lee, "A survey on pattern recognition applications of support vector machines," Int. J. Pattern Recognit. Artif. Intell., vol. 17, no. 3, pp. 459–486, 2003, doi: 10.1142/S0218001403002460.

[6] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," Neural Comput., vol. 22, no. 12, pp. 3207–3220, 2010, doi: 10.1162/NECO_a_00052.

[7] P. Kumar, N. Sharma, and A. Rana, "Handwritten character recognition using different kernel-based SVM classifier and MLP neural network (a comparison)," International Journal of Computer Applications, Vol. 53, No. 11, pp. 25-31, September 2012.

[8] Retno Larasati, Hak Keung Lam, "Handwritten digits recognition using ensemble neural networks and ensemble decision tree", Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS), 2017 International Conference on, 8-10 Nov. 2017.

[9] Jacob, I. Jeena. "Performance Evaluation of Caps-Net Based Multitask Learning Architecture for Text Classification." Journal of Artificial Intelligence 2, no. 01 (2020): 1-10.

[10] A. Thepaut and Y. Autret, "Handwritten digit recognition using combined neural networks," Proc. Int. Jt. Conf. Neural Networks, vol. 2, pp. 1365–1368, 1993, doi: 10.1109/ijcnn.1993.716797.

[11] Khalifa, S., Martin, P., & Young, R. (2019). Label-Aware Distributed Ensemble Learning: A Simplified Distributed Classifier Training Model for Big Data. Big Data Res., 15, 1-11.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[13] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, and G. E. Hinton, "On rectified linear units for speech processing," in the International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013, pp. 3517–3521, IEEE. https://doi.org/10.1109/ICASSP.2013.6638312.

[14] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013.

[15] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." MIT Press, 2016.

[16] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.