International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.2 (Mar-2020)
http://dx.doi.org/10.12785/ijcds/090215

Assessing the Efficacy of Machine Learning Techniques for Handwritten Digit Recognition

Tausifa Jan Saleem[1] and Mohammad Ahsan Chishti[1]

[1] *Department of Computer Science and Engineering, National Institute of Technology Srinagar, India.*

Received 16 Feb. 2019, Revised 30 Jan. 2020, Accepted 23 Feb. 2020, Published 01 Mar. 2020

Abstract: The efficiency of handwritten digit recognition models greatly relies on the classification technique used and the optimization technique involved. Motivated to explore the efficacy of machine learning for handwritten digit recognition, this study assesses the performance of three machine learning techniques, logistic regression, multilayer perceptron, and convolutional neural network for recognition of handwritten digits. The experimental results reveal that convolutional neural network outperforms logistic regression and multilayer perceptron in terms of accuracy. This study also evaluates the performance of three optimizers, namely stochastic gradient descent, adadelta, and adam for handwritten digit recognition. The experiments conducted in the study demonstrate that adam performs better than stochastic gradient descent and adadelta. It is concluded that convolutional neural network with adam is the best choice for handwritten digit recognition in terms of accuracy. However, the convolutional neural network is quite expensive in terms of training time and execution time. To this purpose, this paper proposes a methodology for the design of a light-weight convolutional neural network model.

Keywords: Logistic Regression, Multilayer Perceptron, Convolutional Neural Network, Stochastic Gradient Descent, Adadelta, Adam.

1. INTRODUCTION

The current resurgence in the arena of machine learning is a direct consequence of the efficiency of the machine learning techniques in solving intricate classification and regression tasks on huge datasets [1]. The prime objective of machine learning is to acquire knowledge from experience and to construct a model that can execute prediction tasks for effectual decision-making.

Machine learning techniques are classified into three categories: Supervised learning, unsupervised learning, and reinforcement learning. Supervised learning approaches model dependencies and associations between the target prediction outcome and the input attributes so that outputs for upcoming data instances are forecasted depending on the associations it learned from the dataset [2]. Unsupervised learning applies techniques on the input data instances to mine useful information, detect patterns and group the data instances so that valuable insights are obtained. Reinforcement learning algorithms learn incessantly from the experience of the environment in an iterative manner until they inspect the full range of feasible states.

Automated recognition of handwritten digits has got diverse applications like processing of bank cheques, processing of mails in post offices, etc. [3]. Machine learning techniques have shown remarkable outcomes in diverse pattern recognition tasks, including automated handwritten digit recognition. In spite of being a well-researched domain, handwritten digit recognition is yet a hot area of research [4].

The prime aim of this paper is to evaluate the performance of three supervised machine learning techniques, namely, logistic regression, multilayer perceptron, and convolutional neural network for handwritten digit recognition. Moreover, the performance analysis of three optimizers, stochastic gradient descent, adadelta, and adam is also experimented. The rationale for this performance evaluation stems from the fact that assessing the efficacy of machine learning techniques and the optimizers on MNIST dataset is crucial for deciding the suitability of machine learning techniques and optimizers for handwritten digit recognition.

The rest of the paper is structured as follows: Section 2 presents the related work. Section 3 describes the machine learning techniques used. Section 4 provides a description of optimizers. Section 5 describes the

E-mail: tausifasaleem19@gmail.com, ahsan@nitsri.net

experimental setup. Section 6 contains the experimental results and analysis. Section 7 presents the proposed model for convolutional neural network compression. Section 8 presents the concluding remarks.

2. RELATED WORK

During the recent years, various state-of-the-art models have been developed for handwritten digit recognition. This section reviews the literature related to handwritten digit recognition using machine learning techniques.

Akmaljon et al. [5] provide the performance comparison of four machine learning techniques; Logistic Regression, Convolutional Neural Network, Resnet and Capsule Network for handwritten digit recognition on MNIST in a real-time environment. The study concludes that Capsule Network achieves the highest accuracy (98.1%) and is the most appropriate model among the aforementioned models for real-time handwritten digit recognition. Eva et al. [6] proposed a novel method based on Support Vector Machine (SVM) and Bat algorithm for recognition of handwritten digits. The proposed algorithm achieved an accuracy of 95.6%. In [7], Ravi et al. carried out the proficiency analysis of K-Nearest Neighbor (KNN) for handwritten digit recognition. And it is concluded that KNN classifier generates results with an accuracy of 96.94%. Shruti et al. [8] made use of Spiking Neural Networks (SNN) for recognition of handwritten digits. The work demonstrates that the network attains an accuracy of 98.17%. Savita et al. [9] employed an Artificial Neural Network (ANN) for recognition of handwritten numerals. The experiments conducted in the research work demonstrate that the employed technique achieves an accuracy of 97.5%. Yang et al. [10] proposed an architecture based on deep neural network for classification of handwritten digits. The experiments show that the proposed architecture attains the error rate of 0.47% ± 0.05%. Chayapom et al. [11] presented a comparative performance analysis of three classifications techniques; ANN, KNN, and SVM for the recognition of handwritten digits. The study demonstrates that the SVM outperforms the other two techniques and attains an accuracy of 96.93%. Renata et al. [12] proposed a hybrid model based on SVM and KNN for recognition of handwritten digits. The experiments conducted demonstrate that the proposed model achieves an accuracy of 97.97%. Ahmed et al. [13] employed a convolutional neural network for recognition of handwritten Arabic digits. The study demonstrates that the employed technique shows considerable improvement (an error rate of 12 %) over other machine learning techniques. Sourav et al. [14] proposed a framework based on ANN for recognition of handwritten digits. The work demonstrates that the proposed framework attains an accuracy of 99 %. Y. LeCun et al. [15] carried out a comparison of several classifiers on MNIST dataset. Experiments conducted in the study demonstrate that Boosted LeNet 4 gives the best results in comparison to other classifiers. In [16], authors proposed a model based on the SNN for handwritten digit recognition. The proposed model achieved an accuracy of 95%. Fabien et al. [17] proposed a hybrid model based on LeNet5 and SVM for handwritten digit recognition. The proposed model outperforms the individual LeNet5 and SVM models. Cheng-Lin et al. [18] carried out the comparative analysis of the combinations of several classification and feature extraction techniques on MNIST dataset. Experiments conducted in the study demonstrate that support vector classifier with radial basis function outperforms other combinations. Oliveira et al. [19] proposed a technique based on genetic algorithm for feature selection on MNIST dataset. The proposed algorithm reduces the number of features in the dataset effectively without affecting the classification accuracy. In another study, Oliveira et al. [20] carried out a comparison of two genetic algorithm approaches on MNIST dataset, simple genetic algorithm, and iterative genetic algorithm. Experiments conducted in the study demonstrate that simple genetic algorithm is more suitable for handwritten digit recognition. Dejan et al. [21] proposed a hybrid classification model based on ANN and SVM for handwritten digit recognition. Experiments demonstrate that the proposed model is on par with the state-of-the-art models in terms of classification accuracy and time. Bellili et al. [22] proposed a model based on multilayer perceptron and SVM for handwritten digit recognition. The proposed model achieved an accuracy of 98.01%.

3. MACHINE LEARNING TECHNIQUES

This section provides a brief description of the machine learning techniques used in our study. Three machine learning models based on logistic regression, multilayer perceptron, and convolutional neural network have been developed. Following provides a brief description of these techniques:

A. Logistic Regression:

Logistic Regression (LR) is a machine learning algorithm that can be utilized for multivariate classification [23]. While as linear regression is utilized for predicting the future instances of a dependent variable, logistic regression is typically used for classification tasks. Logistic regression makes use of a sigmoid function ($f(z) = \frac{1}{1+e^{-z}}$) to generate the probability values, which are then mapped to multiple classes [24].

B. Multi-Layer Perceptron

Neuron is the elementary computational unit in ANN. It accepts one or more inputs and performs their weighted sum, which is then passed as an input to a non-linear function called as activation function. Activation function may be a threshold function, piecewise linear function, logistic function, gaussian function, etc. ANNs can be contemplated as a directed weighted graph with neurons as nodes and weights as directed edges. Feed-forward

ANN with at least one hidden layer is called as a Multi-Layer Perceptron (MLP) [25].

C. Convolutional Neural Network

A Convolutional Neural Network (CNN) receives a two-dimensional input in the form of an image or a voice signal and digs out hierarchical characteristics by means of a sequence of hidden layers [26, 27]. The hidden layers in case of a CNN comprise of convolutional layers, pooling layers, and a fully connected layer [28]. The convolutional layer comprises of filters that have the same shape as that of the input data. However, the dimensions of filter are smaller than the dimensions of input. The output of the convolutional layers are the feature maps that are obtained by the inner product of input and the filter. These feature maps are passed through pooling layers in order to shrink the dimensions of the representation. This is done to reduce computation time and avoid over-fitting. Another main part of CNN is the Rectified Linear Units (ReLU) that constitutes of neurons with softmax activation function in the form of $f(z) = \max(0, z)$.

4. OPTIMIZERS

Gradient descent is the most widely used technique for optimizing machine learning models. It is a method for minimizing a function known as objective function represented by $J(w)$, where w denotes the model parameters. It proceeds by updating the values of model parameters in the direction opposite to the gradient of $J(w)$, until the point of minima is attained. The length of steps taken to reach the point of minima is determined by a parameter called as learning rate, η. Training machine learning models with gradient descent is a slow process and consumes a lot of time. To overcome this limitation, various optimization algorithms have been designed. Following provides a description of the commonly used optimizers for gradient descent.

A. Stochastic Gradient Descent (SGD)

SGD computes the model parameter update for every input-output pair in the dataset. The parameter update equation is given as [29];

$$w = w - \eta . \nabla_w J(w; y^{(i)}; z^{(i)}) \quad (1)$$

Where $y^{(i)}$ represents the i^{th} input data instance, and $z^{(i)}$ represents the i^{th} output label. The main advantage of SGD over the other optimizers is that it is fast as it doesn't recompute the gradient for similar examples.

B. Adadelta

Adadelta doesn't need the manual fine-tuning of η. The parameter update rule in adadelta consists of the following steps [30]:

Step1: Calculate gradient ($G(t)$) at time t,

$$G_{t,i} = \nabla_{\mathrm{w}} \mathrm{J}(\mathrm{w_{t,i}}) \quad (2)$$

$$w_{t+1,i} = w_{t,i} - \eta . G_{t,i} \quad (3)$$

Step 2: Calculate the running average of the gradient at time t, ($E[G^2]_t$),

$$E[G^2]_t = m\, E[G^2]_{t-1} + (1-m)\, G^2{}_t \quad (4)$$

m is the decay constant with value of 0.9 approximately.

Step 3: Calculate the parameter update at time t, (Δw_t),

$$\Delta w_t = -\frac{\eta}{\sqrt{E[G^2]_t + \epsilon}}\, G_t \quad (5)$$

ϵ is a small number that is added to prevent division by zero.

$$RMS(G_t) = \sqrt{E[G^2]_t + \epsilon} \quad (6)$$

$RMS(G_t)$ is the root mean square of gradient G_t

$$\Delta w_t = -\frac{\eta}{RMS[G]_t}\, G_t \quad (7)$$

From equation 4, we have,

$$E[\Delta w^2]_t = mE[\Delta w^2]_{t-1} + (1-m)\, \Delta w^2{}_t$$

$$RMS[\Delta w]_t = \sqrt{E[\Delta w^2]_t + \epsilon} \quad (8)$$

From equations 4, 7, and 8, we get,

$$\Delta w_t = -\frac{RMS[\Delta w]_{t-1}}{RMS[G]_t}\, G_t \quad (9)$$

Step 4: Apply the parameter update,

$$w_{t+1} = w_t + \Delta w_t \quad (10)$$

C. Adam

Adam (Adaptive moment estimation) is an adaptive learning technique that calculates the learning rate for every model parameter. The parameter update rule in Adam consists of the following steps [31]:

Step 1: Compute the average of past gradients, M_t,

$$M_t = \propto_1 M_{t-1} + (1 - \propto_1) G_t \quad (11)$$

Step 2: Compute the average of past squared gradients, V_t

$$V_t = \propto_2 V_{t-1} + (1 - \propto_2)\, G^2{}_t \quad (12)$$

$\propto_1, \propto_2$ are the decay rates.

Step 3: Calculate the estimate of M_t (equation 11), $\widehat{M_t}$

$$\widehat{M_t} = \frac{M_t}{1-\alpha_1{}^t} \quad (13)$$

Step 4: Calculate the estimate of V_t (equation 12), $\widehat{V_t}$

$$\widehat{V_t} = \frac{V_t}{1-\alpha_2{}^t} \quad (14)$$

Step 5: The weight update equation is given as,

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\widehat{V_t}+\epsilon}}\widehat{M_t} \quad (15)$$

5. EXPERIMENTAL SETUP

The experiments conducted in this study were performed on an Intel (R) Core i3-6006U CPU@ 2.00GHz system with 4GB RAM. The machine learning techniques were implemented on python 3.5 using keras. Three models based on LR, MLP, and CNN were built for classification of handwritten digits on MNIST dataset.

MNIST (Modified National Institute of Standards and Technology) dataset was created by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The dataset consists of scanned images of handwritten digits (0-9). Figure 1 shows the example of handwritten digits in MNIST dataset.

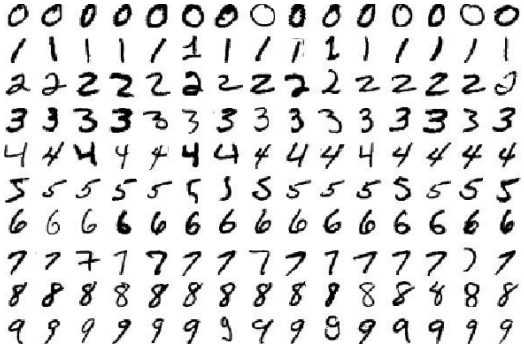

Figure 1. Handwritten digit examples in MNIST dataset

Each digit is a 28*28 image. MNIST comprises of 60,000 samples for training and 10,000 samples for testing. Table 1 provides the distribution of various digits (classes) in MNIST dataset. The class distribution is pictorially represented by figure 2.

TABLE 1. CLASS DISTRIBUTION IN MNIST DATASET

Digit	Number of instances
0	5923
1	6742
2	5958
3	6131
4	5842
5	5421
6	5918
7	6265
8	5851
9	5949

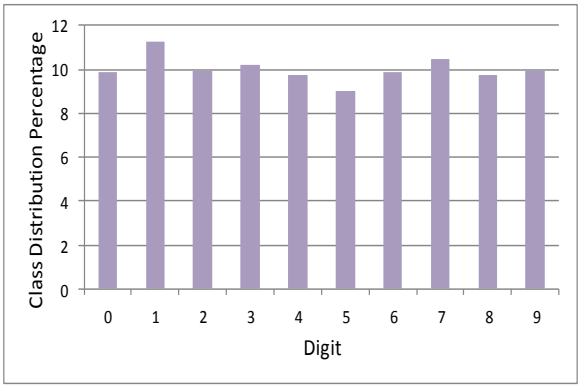

Figure 2. Class distribution in MNIST dataset

Table 2 lists the various parameter values that are common to all three models.

TABLE 2. PARAMETER VALUES

Parameter	Value
Number of training samples	60,000
Number of testing samples	10,000
Input dimension	784
Number of classes	10
Batch size	128
Number of epochs	15
Activation Function	ReLu, Softmax
Loss Function	categorical_crossentropy
Optimizers	SGD, adadelta, adam

The number of training and testing samples in MNIST dataset are 60,000 and 10,000, respectively. Models are trained in batches of size 128 and 15 epochs are performed on the dataset. This is because after 15 epochs, the accuracy almost remains steady. Softmax is used as an activation function in case of LR. The MLP model consists of an input layer with 784 neurons, one hidden layer with 784 neurons and an output layer with 10 neurons. A dropout with rate of 0.25 is applied on hidden layer neurons in order to improve the generalization capability of the model and to reduce the chances of model overfitting. CNN model consists of two convolutional layers, a max-pooling layer, and an output

layer. The number of kernels in the convolutional layers is 32 and 64 respectively with a kernel size of 3*3. ReLu is used as an activation function in the convolutional layers. A dropout of 0.25 rate is applied after max pooling layer in order to avoid overfitting. A Softmax activation function is used in the output layer.

6. Results and Analysis

The performance of the models is evaluated based on three different optimizers, namely SGD, adadelta, and adam. Table 3 presents the validation accuracies of all the three models in case of all the optimizers.

Figure 3 depicts the effect of the increase in the number of epochs on validation accuracy of the logistic regression model in case of all the three optimizers. It is apparent from the figure that the validation accuracy increases monotonically with an increase in the number of epochs in case of all the optimizers. However, the accuracy of adam and adadelta surpass the accuracy of SGD. Also, the accuracy of Adam is slightly greater than that of adadelta. Initially, there is a considerable difference between the validation accuracies of adam and adadelta. At epoch 1, the validation accuracy of adadelta is 86.87%, and that of adam is 90.28%, which means that the differences in the accuracies at epoch 1 is equal to 3.41%. And after 15 epochs, the validation accuracy of adadelta is 92.31%, and that of adam is 92.71%, and hence the difference in accuracies is equal to 0.4%. This means that at higher number of epochs adadelta and adam perform almost equally in the logistic regression model. However, in case of adam and SGD, even after 15 epochs there is a considerable difference between accuracies which is equal to 2.19%. Hence, it is concluded that in case of logistic regression adam and adadelta perform almost equally, and the performance of these two algorithms surpass the performance of SGD.

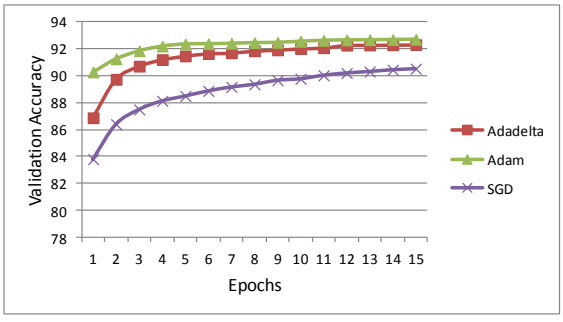

Figure 3. Validation Accuracy versus epochs in case of Logistic Regression

Figure 4 illustrates the effect on validation accuracy of multilayer perceptron model with an increase in the number of epochs in case of all the three optimizers. It is evident from the figure that validation accuracy increases monotonically with an increase in the number of epochs in case of all the optimizers. However, in this case as well, the accuracy of adam and adadelta is greater than the accuracy of SGD. And, the accuracy of adam is slightly greater than adadelta. At epoch 1, the difference between accuracies of adam and adadelta is equal to 1.82%, and after 15 epochs the difference between the accuracies of the aforementioned optimizers diminishes to 0.12%. In case of adam and SGD, the difference between accuracies at epoch 1 is equal to 9.35%, and after 15 epochs the difference diminishes to 4.68%. This means that in case of multilayer perceptron the difference between the accuracies of adam and adadelta is even lesser than that of logistic regression. However, there is a considerable difference between the accuracies of adam and SGD.

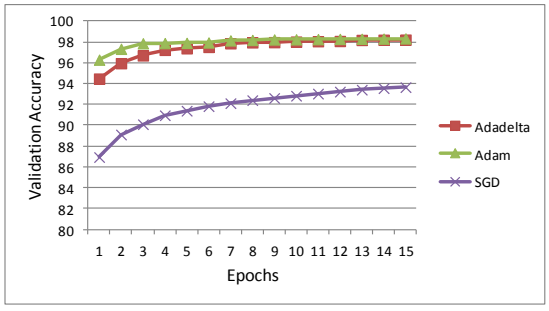

Figure 4. Validation Accuracy versus epochs in case of MultiLayer Perceptron

Figure 5 describes the effect on validation accuracy of the convolutional neural network model with an increase in the number of epochs in case of all the three optimizers. It is apparent from the figure that validation accuracy increases monotonically with an increase in the number of epochs in case of all the optimizers. In this case as well, the same order follows for the accuracy: Adam > Adadelta > SGD. However, there is a little difference in the accuracies of adam and adadelta. At epoch 1, the difference in the accuracies of adam and adadelta is equal to 0.68%, and after 15 epochs the difference diminishes to 0.18%. In case of adam and SGD, the difference in the accuracies at epoch 1 is equal to 7.81%, and after 15 epochs the difference diminishes to 1.96%. Hence, it can be concluded that in case of CNN as well, there is a little difference in the accuracies of adam and adadelta while as a considerable difference exists between the accuracies of adam and SGD.

TABLE 3. VALIDATION ACCURACIES

	Logistic Regression			Multilayer Perceptron			Convolutional Neural Network		
No. of Epochs	SGD	Adadelta	Adam	SGD	Adadelta	Adam	SGD	Adadelta	Adam
1	83.79	86.87	90.28	86.94	94.47	96.29	90.46	97.59	98.27
2	86.39	89.73	91.25	89.1	95.95	97.33	92.85	98.17	98.84
3	87.49	90.71	91.85	90.09	96.7	97.86	93.91	98.53	98.86
4	88.13	91.2	92.2	90.95	97.23	97.88	94.8	98.74	98.94
5	88.49	91.47	92.36	91.39	97.38	97.9	95.32	98.76	98.99
6	88.87	91.65	92.38	91.82	97.51	97.94	95.6	98.83	99.01
7	89.16	91.7	92.41	92.11	97.86	98.14	95.91	98.85	99.03
8	89.36	91.85	92.46	92.4	97.95	98.18	96.35	98.87	99.05
9	89.66	91.92	92.48	92.61	97.97	98.23	96.36	98.9	99.14
10	89.77	92.02	92.56	92.83	98.02	98.24	96.62	98.95	99.15
11	90.03	92.08	92.63	93.03	98.06	98.26	96.65	98.87	99.17
12	90.2	92.27	92.66	93.22	98.08	98.28	96.77	98.98	99.2
13	90.31	92.28	92.67	93.41	98.18	98.29	97.03	99.02	99.22
14	90.44	92.3	92.69	93.55	98.2	98.31	97.17	99.04	99.23
15	90.52	92.31	92.71	93.65	98.21	98.33	97.29	99.07	99.25

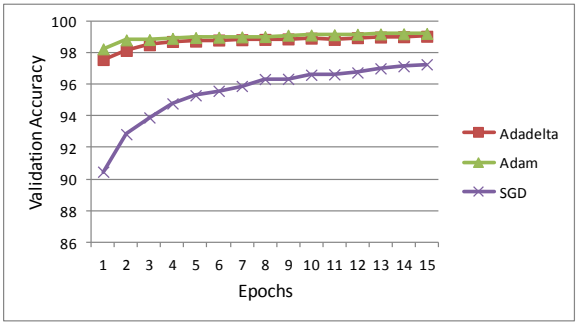

Figure 5. Validation Accuracy versus epochs in case of Convolutional Neural Network

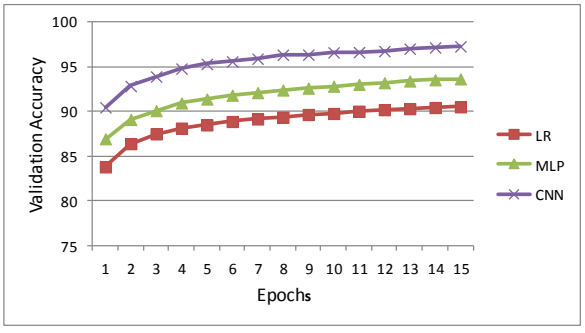

Figure 6. Validation Accuracy versus epochs in case of SGD

The comparative analysis of the models is also carried out in case of all the three optimizers.

Figure 6 illustrates the relationship between validation accuracy and the number of epochs for all the three models in case of SGD optimizer. It is evident from the graph that the validation accuracy of all the three models increases monotonically with the rise in the number of epochs. However, the accuracy of the models is in the order: CNN > MLP > LR. The difference in the accuracies of CNN and MLP after 15 epochs is equal to 3.64%. In case of CNN, and LR the difference in the accuracies after 15 epochs is equal to 6.77%, which is much higher. Therefore, in case of SGD optimizer CNN performs better.

Figure 7 illustrates the relationship between validation accuracy and the number of epochs for all the three models in case of adadelta optimizer. It is obvious from the graph that validation accuracy of all the three models increases monotonically with the rise in the number of epochs. Moreover, the accuracy of the models follows the same order as that of SGD: CNN > MLP > LR. The difference in the accuracies of CNN and MLP after 15 epochs is equal to 0.86%, which is lesser than the difference in accuracies in case of SGD optimizer. In case of CNN and LR, the difference in the accuracies after 15 epochs is equal to 6.76%, which is almost of the same magnitude as that of SGD.

Hence, in case of adadelta optimizer as well, CNN performs better, and there is a lesser difference in the accuracies of CNN and MLP than that of SGD.

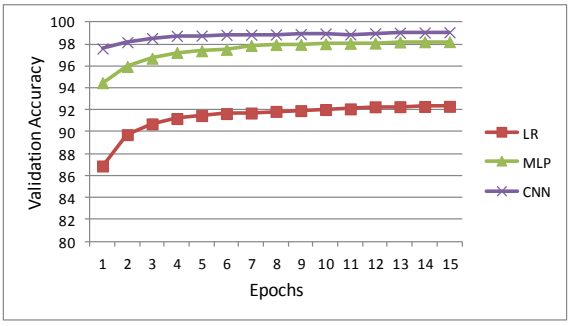

Figure 7. Validation Accuracy versus epochs in case of Adadelta

Figure 8 illustrates the relationship between validation accuracy and the number of epochs for all the three models in case of adam optimizer. It is evident from the graph that the validation accuracy of all the three models increases monotonically with the rise in the number of epochs. The accuracy of the models in this case also follows the same trend: CNN > MLP > LR. The difference in the accuracies of CNN and MLP after 15 epochs is equal to 0.92% which is slightly higher than the difference in the accuracies in case of adadelta but lesser than that of SGD optimizer. In case of CNN and LR, the difference in the accuracies after 15 epochs is equal to 6.54%, which is slightly lesser than that of SGD and adadelta.

Hence, the same trend follows in case of adam optimizer as well, and there is a comparatively lesser difference in the accuracies of CNN and LR than that of SGD and adadelta.

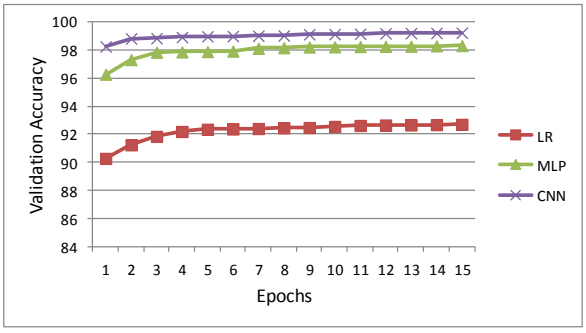

Figure 8. Validation Accuracy versus epochs in case of Adam

Figure 9 presents the test accuracy of all the three models in case of all the three optimizers. It is evident from the figure that the accuracy of CNN with Adam optimizer (99.07 %) surpasses all the other models. Hence, CNN is the best choice for handwritten digit classification in terms of accuracy.

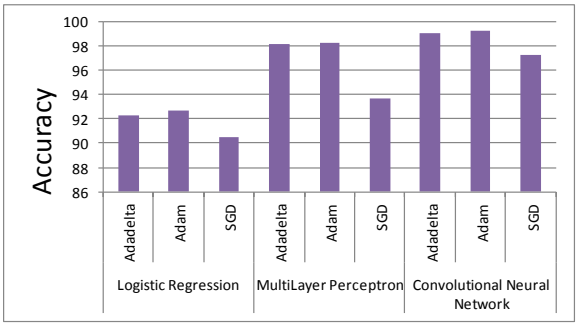

Figure 9. Comparison of accuracies

Figure 10 illustrates the comparison of the training times per epoch of all the three models. The training times/epoch follows the order: CNN > MLP > LR, which means CNN is costly in terms of training time than the other two models. The training time per epoch in case of CNN is equal to 315 seconds which is far greater than the training times of MLP (44 seconds) and LR (22 seconds).

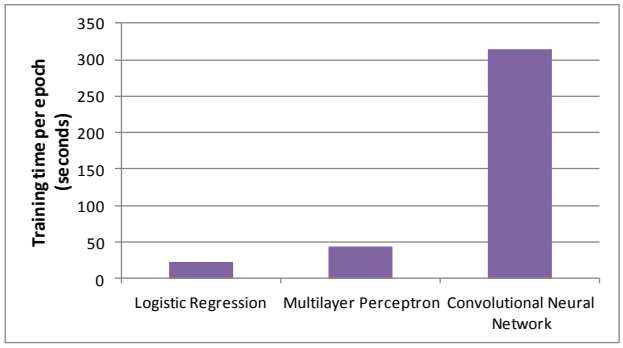

Figure 10. Training time per epoch

Figure 11 illustrates the comparison of the execution times of all the three models. The execution time also follows the same order as that of training time: CNN > MLP > LR, which means CNN is costly in terms of execution time as well as compared to the other two models. The execution time of the CNN model is equal to 8 seconds which is greater than the execution times of MLP (3.5 seconds) and LR (2 seconds).

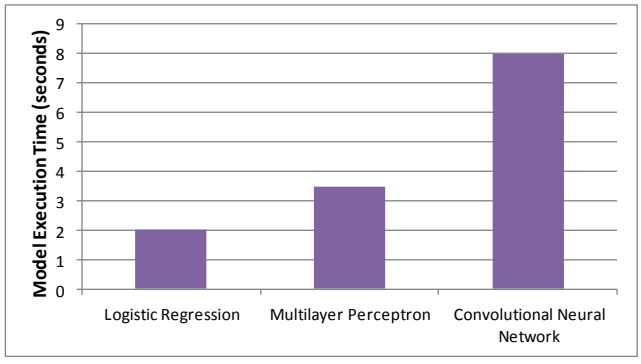

Figure 11. Model Execution Time

The following provides a summary of the experimental findings:

- The validation accuracy increases with an increase in the number of epochs in case of all the scenarios. This is because of the reason that the model parameters are optimized after every epoch and with the result, the value of the loss function decreases considerably.
- The accuracy of the models follows the order: CNN > MLP > LR. This is due to the reason that CNN has got tremendous feature extraction potential in comparison to other models.
- The accuracy of the optimizers follows the order: Adam > Adadelta > SGD. This is because of the adaptive learning nature of adam.
- Training time and execution time of the models follow the order: CNN > MLP > LR. This is due to the large number of computations and the parameters involved in CNN in comparison to MLP and LR.

From the above experimental findings, it is concluded that although CNN performs better than the other two models in terms of accuracy, the computational complexity of CNN is much higher than that of MLP and LR. Therefore, there exists a tradeoff between accuracy and model complexity of the applied techniques for handwritten digit recognition. Moreover, the computational complexity of the CNN model restricts its deployment in resource-constrained systems. The solution to this problem lies in compressing the CNN model without affecting the model accuracy. The following section presents a methodology for the design of a lightweight CNN model.

7. PROPOSED MODEL FOR CNN COMPRESSION

The proposed model consists of the following steps (Figure 12):

A. Prune the unimportant and redundant weight connections: The connections in the original CNN are analyzed, and the weight connections with values less than a particular threshold are removed. Also, the network is analyzed for parameter redundancies, and the redundant connections in the network are pruned. This results in a network with a lesser number of parameters.

B. Weight Sharing: This step reduces the number of effective weights in the network by sharing the weights between multiple connections. Hence, a lesser number of bits is required for representing the weights.

Suppose a is the number of effective weights and b is the total number of connections, the compression rate C will be given as,

$$C = \frac{bl}{b \log_2 a + al}$$

Where l is the number of bits used to represent each connection.

Huffman Encoding: In this step, the frequently occurring weights are represented with a lesser number of bits so as to reduce the number of computations.

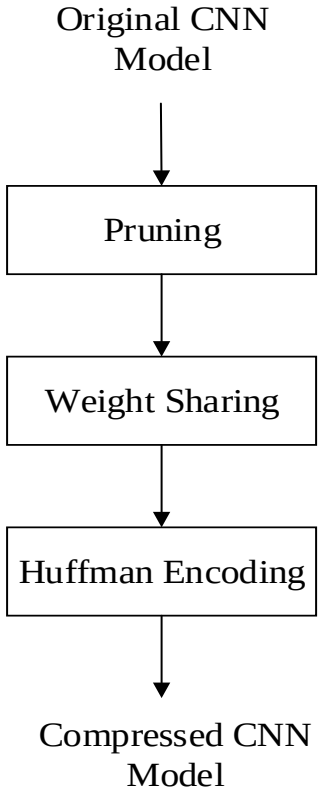

Figure 12. Proposed Model

The proposed methodology discussed above reduces the number of parameters in the CNN model. Moreover, the number of bits used to represent the parameters is also reduced. This limits the number of computations, and with the result, computational complexity of the model is alleviated and that too without affecting accuracy.

8. CONCLUSION

This paper assesses the efficacy of three machine learning techniques, namely logistic regression, multilayer perceptron, and convolutional neural network for classification of handwritten digits on MNIST dataset. The performance of the aforementioned techniques is evaluated for three different optimizers, namely adadelta, adam, and SGD. It is evident from the experiments that the accuracy of CNN surpasses the other two machine learning models in case of all the aforementioned optimizers. This study also carries out the performance analysis of the optimizers in case of all the three models. Experimental results demonstrate that adam outperforms

the other two optimizers for all the three machine learning models. It is thereby concluded that CNN with adam optimizer is the best choice for handwritten digit classification in terms of accuracy. However, the computational complexity of CNN is quite higher than the other two models. To this purpose, this paper proposes a methodology for the design of a lightweight CNN model. The future work would be to implement the proposed model and to analyze the effect of model compression on the accuracy and execution time of the model.

REFERENCES

[1] Mark D.McDonnell, Migel D. Tissera, Tony Vladusich, Andrévan Schaik, and Jonathan Tapson, "Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the Extreme Learning Machine Algorithm," PLOS ONE, August, 2015.

[2] Tausifa Jan Saleem, and Mohammad Ahsan Chishti, "Data Analytics in Inernet of Things: A Survey," Scalable Computing: Practice and Experience, vol. 20, no. 4, pp. 607-629.

[3] Areej Alsaafin, and Ashraf Elnagar, "A Minimal Subset of Features Using Feature Selection for Handwritten Digit Recognition," Journal of Intelligent Learning Systems and Applications, vol. 9, pp. 55-68, 2017.

[4] Angelo cardoso, and Andreas Wichert, "Handwritten digit recognition using biologically inspired features," Neurocomputing, vol. 99, pp. 575-580, 2013.

[5] Akmaljon Palvanov, and Young Im Cho, "Comparisons of Deep Learning Algorithms for MNIST in Real-Time Environment," International Journal of Fuzzy Logic and Intelligent Systems, vol. 18, No. 2, pp. 126-134, June 2018.

[6] Eva Tuba, Milan Tuba, and Dana Simian, "Handwritten Digit Recognition by Support Vector Machine Optimized by Bat Algorithm," 24th Conference on Computer Graphics, Visualization and Computer Vision, 2016.

[7] U Ravi Babu, Dr. Y Venkateswarlu, Aneel Kumar Chintha, "Handwritten Digit Recognition Using K-Nearest Neighbour Classifier," World Congress on Computing and Communication Technologies, IEEE, 2014.

[8] Shruti R.Kulkarni, and BipinRajendran, "Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization," Neural Networks, vol. 103, pp. 118-127, April 2018.

[9] Savita Ahlawata, and Rahul Rishib, "Off-line Handwritten Numeral Recognition using Hybrid Feature Set – A Comparative Analysis," Procedia Computer Science, vol. 122, pp. 1092–109, 2017.

[10] Yang Li, Hang Li, Yulong Xu, Jiabao Wang, and Yafei Zhang "Very Deep Neural Network for Handwritten Digit Recognition," IDEAL 2016, Springer, pp. 174–182, 2016

[11] Chayaporn Kaensar "A Comparative Study on Handwriting Digit Recognition Classifier Using Neural Network, Support Vector Machine and K-Nearest Neighbor," IC2IT2013, Springer, pp. 155–163, 2013.

[12] Renata F.P. Neves, Cleber Zanchettin, and Alberto N.G. Lopes Filho, "An Efficient Way of Combining SVMs for Handwritten Digit Recognition," ICANN 2012, Springer, pp. 229–237, 2012.

[13] Ahmed El-Sawy, Hazem EL-Bakry, and Mohamed Loey, "CNN for Handwritten Arabic Digits Recognition Based on LeNet-5," Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016, Advances in Intelligent Systems and Computing, 2016.

[14] Sourav Saha, Sudipta Saha, Suhrid Krishna Chatterjee and Priya Ranjan Sinha Mahapatra, "A Machine Learning Framework for Recognizing Handwritten Digits Using Convexity-Based Feature Vector Encoding," Proceedings of International Ethical Hacking Conference 2018, Advances in Intelligent Systems and Computing, 2018.

[15] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison Of Learning Algorithms For Handwritten Digit Recognition," International Conference on Artificial Neural Networks, Paris, pp. 53-60, 1995.

[16] Peter U. Diehl, and Matthew Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity, " Frontiers in computational Neuroscience, vol. 9, 2015.

[17] Fabien Lauera, ChingY. Suen, Gérard Blocha, "Atrainablefeatureextractorforhandwrittendigitrecognition," Pattern Recognition, vol. 40, pp. 1816–1824, 2007.

[18] Cheng-Lin Liu, Kazuki Nakashima,Hiroshi Sako, Hiromichi Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art technique," Pattern Recognition, vol. 36, pp. 2271–2285, 2003.

[19] L. S. Oliveira, R. Sabourin, F. Bortolozzi and C. Y. Suen, "Feature Selection Using Multi-Objective Genetic Algorithms for Handwritten Digit Recognition," Object recognition supported by user recognition for service robots, IEEE, 2002.

[20] L. S. Oliveira, R. Sabourin, F. Bortolozzi and C. Y. Suen, "Feature Subset Selection Using Genetic Algorithms for Handwritten Digit Recognition," IEEE, 2001.

[21] Dejan Gorgevik, Dusan Cakmakov, "An Efficient Three-Stage Classifier for Handwritten Digit Recognition," Proceedings of the 17th International Conference on Pattern Recognition, IEEE, UK, 2004.

[22] A. Bellili, M. Gilloux, P. Gallinari, "An MLP-SVM combination architecture for offline handwritten digit recognition," IJDAR, vol. 5, pp. 244–252, 2003.

[23] Tausifa Jan Saleem, and Mohammad Ahsan Chishti, "Exploring the Applications of Machine Learning in Healthcare," International Journal of Sensors, Wireless Communications and Control. DOI: http://dx.doi.org/10.2174/2210327910666191220103417.

[24] Jason Brownlee, "Logistic Regression for Machine learning," Available at: https://machinelearningmastery.com/logistic-regression-for-machine-learning/.

[25] Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil, "Multilayer Perceptron: Architecture Optimization and Training," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 4, pp. 26-30, 2016.

[26] Keiron O'Shea, and Ryan Nash, "An Introduction to Convolutional Neural Networks," Available at: https://arxiv.org/abs/1511.08458.

[27] Tausifa Jan Saleem, and Mohammad Ahsan Chishti, "Deep Learning for Internet of Things Data Analytics," Procedia Computer Science, vol. 163, pp. 381-390.

[28] L. Jayasinghe, N. Wijerathne, C. Yuen, M. Zhang, "Feature Learning and Analysis for Cleanliness Classification in Restrooms", IEEE Access, vol. 7, pp. 14871-14882, Jan 2019.

[29] Rasmus Hall´en, "A Study of Gradient-Based Algorithms," Available at: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8904399&fileOId=8904400.

[30] Matthew D. Zeiler, "Adadelta: An Adaptive Learning Rate Method," Available at: https://arxiv.org/pdf/1212.5701.pdf.

[31] DiederikP.Kingma, Jimmy Lei Ba, "Adam: A Method For Stochastic Optimization," Available at: https://arxiv.org/abs/1412.6980.

Tausifa Jan Saleem is pursuing her PhD at the Department of Computer Science & Engineering, National Institute of Technology Srinagar, India. She has done bachelors in Information Technology from National Institute of Technology Srinagar and masters in Computer Engineering from University of Jammu, India. Her research focuses on Internet of Things, Data analytics, and Machine Learning.

Mohammad Ahsan Chishti has done his Bachelor of Engineering and M.S. in Computer and Information Engineering from International Islamic University Malaysia with specialization in Computer Networking. He has received his Ph.D. from National Institute of Technology Srinagar, India. Presently he is working as Assistant Professor in the Department of Computer Science & Engineering, National Institute of Technology Srinagar, India. He has more than 50 research publications to his credit and 12 patents with two granted International Patents. He is Senior Member-Institute of IEEE, Member IEI, Life Member CSI, and Member IETE. He is a certified White belt in Six Sigma by Six Sigma Advantage Inc. of USA (SSAI).