International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I1P105

Volume 72 Issue 1, 48-55, January 2024

Original Article

Impact of Machine Learning and Deep Learning Models on Handwritten Digits and Letters Recognition (HDaLR)

Ratna Nitin Patil[1], Yogita Deepak Sinkar[2], Shitalkumar Adhar Rawandale[3] , Varsha D. Jadhav[4]

[1,4]*Department of AI & DS, Vishwakarma Institute of Information Technology, Pune, Maharashtra, India.*
[2]*Department of Computer Engineering, SVPM's College of Engineering, Baramati, Pune, Maharashtra, India.*
[3]*Department of Mechanical Engineering, Pimpri Chinchwad College of Engineering and Technology, Maharashtra, India.*

[1]*Corresponding Author : ratna.nitin.patil@gmail.com*

Received: 20 September 2023 Revised: 05 December 2023 Accepted: 12 December 2023 Published: 07 January 2024

Abstract *- In numerous practical applications, such as data form entry, postal code sorting, and bank check account processing, handwritten digit recognition is one of the crucial and difficult tasks. Because each person writes in a distinct way with varying sizes, widths, and slopes, it can be challenging to recognise digits. Various artificial neural network-based models have been used in the past for pattern matching. While conducting the experiment, significant differences in the use of fonts by various authors were observed using the MNIST (Modified National Institute of Standards and Technology database) dataset as a benchmark. In this study, we evaluated machine learning algorithms on the MNIST dataset, including Naive Bayes, K-Nearest Neighbor, Support Vector Machine, Decision Tree, Random Forest, Artificial Neural Network, Convolution Neural Network, and Long Short-Term Memory. The purpose of this research is to evaluate and contrast the effectiveness of deep learning and machine learning models over handwritten letters and digits datasets. It was noted that CNN has outperformed, and the accuracy obtained is 99.9% over the MNIST dataset and 88% over the EMNIST dataset. Every identification approach faces the crucial challenge of extracting key features, and deep learning has been used to solve this problem with results that have been evaluated.*

Keywords *- Handwritten digits, MNIST, SVM, Deep learning, CNN, LSTM.*

1. Introduction

Since 1980, image processing has actively pursued the research of handwritten digit recognition; nevertheless, it only gained widespread attention in 1998 after a comparison and investigation of numerous handwritten digit identification systems. This is in response to the MNIST handwritten numerical dataset's publishing. The variety in people's writing styles is the main obstacle to accurately identifying handwritten numerals [1]. Machine learning and deep learning have made significant progress since 2011 in a variety of research fields, which include handwritten digit recognition, disease detection, traffic prediction, face expression identification, image categorization sentiment analysis, etc. Deep learning and machine learning methods can be used to recognize handwritten digits [2].The efficacy of the digit recognizer must be improved through feature extraction. In the field of computer vision, Handwritten Digit Recognition (HDR) is a classic problem that has plenty of applications, including processing bank check amounts, identifying zip codes for mail sorting, online handwriting recognition using tablets, and numeric entries in forms filled out by hand. The digits written by hand are not always of the same size, thickness/width, slope, or position relative to the margins [3] [4].

The aim of this work is to compare the performance of deep learning and machine learning techniques and to present the most accurate Handwritten Digit and Letter Recognition (HDaLR) system that can automatically extract the key features.The MNIST dataset of images was used in the current study to test the implementation of pattern recognition for handwritten digits (0-9).

The identification of 9 & 8, 8 & 3, 5 & 6, 1 & 7, etc., was the main problem with Optical Digital Recognition (ODR). Another serious issue is that various people have different ways of writing the same digit. For example, 9 is written by different individuals as '9', '9', '٩' etc. It has been observed that a person's distinctiveness and variety of handwriting affect the way their digits seem and shape. Humans are able to see, notice, and visualize their surroundings by using their eyes and minds [5].

The foundation of computer vision is giving computers the ability to view and analyse images, similarly to how humans do. The purpose of this study is to assess and compare the performance of deep learning and machine learning models.

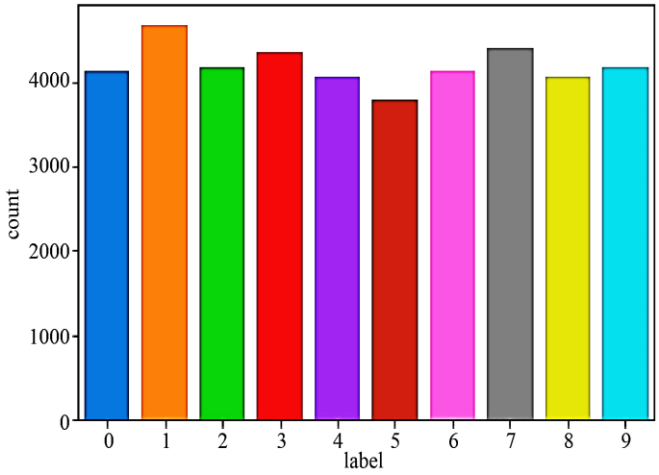

Fig. 1 Distribution of the MNIST data set class labels

The MNIST dataset served as the testing ground for a variety of machine learning algorithms in this study, including Naive Bayes, K-Nearest Neighbor (k-NN), Support Vector Machine (SVM), Decision Tree, Random Forest, Artificial Neural Network (ANN), Convolution Neural Network (CNN), and Long Short-Term Memory (LSTM) [6] [7].

2. Related Work

In this section, a brief overview of earlier research on handwritten digit recognition is discussed.

2.1. Machine Learning Models

Previous works are centered on creating new classification models based on a specific numerical dataset. The traditional machine learning methods used for handwritten digit and letter recognition are examined in this section. Among the techniques addressed are SVM, k-NN, Random Forests (RF), Decision Trees, and Naive Bayes Classifiers, which are not based on neural networks [8].

2.2. Deep Learning Models

The subject of pattern recognition, particularly the recognition of handwritten characters, has been transformed by deep learning models. The deep learning architectures covered in this section include CNN, RNN and LSTM. Convolutional neural networks are used to represent deep learning models in the area of handwritten digit recognition. Bhattacharya and Chaudhuri [9] presented a multi-level recognition of mixed numbers in order to recognize handwritten numbers in Bangla, English and Devanagari. In recent years, many authors have used the deep learning model for handwritten recognition in different languages [10] [11].

3. Methods

3.1. Dataset

A thriving research community exists for digits recognition. Due to autonomous feature extraction, deep learning approaches are currently being used by computer vision researchers. The classification of handwritten numbers uses the MNIST dataset, which is accessible on KAGGLE [12]. The counts for the labels (Labels) in the MNIST dataset are shown in Fig 1.

Extended MNIST (EMNIST), obtained from NIST Special Database 19, is then transformed into a 28 x 28-pixel image format dataset whose structure perfectly matches that of the MNIST dataset of handwritten character digits [13].

3.2. Naïve Bayes Classifier

The Naive Bayes algorithm is a method of probabilistic machine learning used for classification tasks. The Bayes theorem serves as the basis for the algorithm. The key concept is that the likelihood of an unseen sample falling into each class is determined using the method in Equation 1. The class with the highest likelihood is then used to forecast the label instance based on the estimated probability. Next, depending on the determined likelihood, the label instance is predicted using the class with the highest probability.The predictions in this machine learning algorithm are presumed to be independent. The performance of the predictive model is compromised since many of the traits of real-life instances do not conflict.

The dataset utilized is the MNIST dataset, which has 70000 grayscale images with digits ranging from 0 to 9. 10,000 images are used for testing, and 60,000 images are included in the training set. P(c | image) must be calculated in order to apply Naive Bayes to find the digit as referred to by Equation 1.

$$P(C \mid image) = \frac{P(image \mid C) * P(C)}{P(image)} \quad (1)$$

It ignores P(image), which is constant for all digit images and has no impact on the outcome. Then, calculate the probabilities for each class, ranging from 0 to 9, in order to identify the class of an image. The class for which the image was predicted with the highest probability is the predicted class, as given by Equation 2.

$$Predicted\ label = \arg\max_{C_i \epsilon\, M} P(\text{image} \mid C_i) * P(C_i) \quad (2)$$

Where M represents the total number of classes and C is the category or class.

Class prior probability P(C) is computed by Equation 3,

$$P(C) = \frac{count(images\ of\ the\ class)}{count(All\ the\ images)} \quad (3)$$

$Predictedlabel$=argmaxCi∈MProbabilityX1,X2,...Xn$Ci)Probability(Ci)$

P(image | Ci) is calculated using each image's 28 by 28 pixels, which have a colour range of 0 to 255. Additionally, each image has 255*784 possible outcomes if we map the probability distribution, which is quite time-consuming computationally.

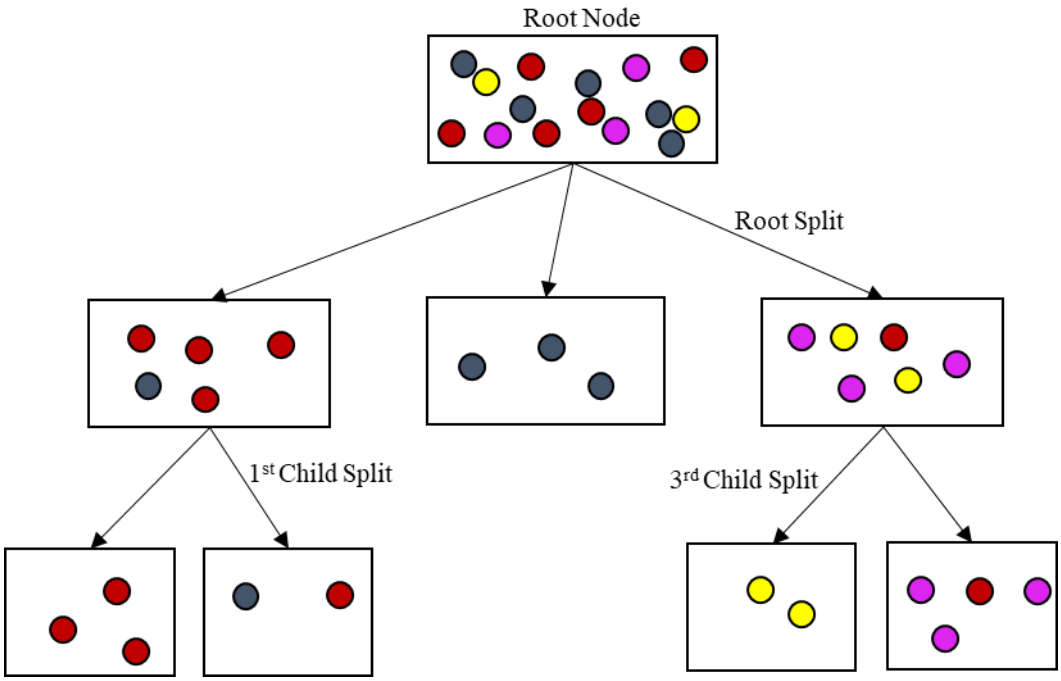

Fig. 2 Decision tree model

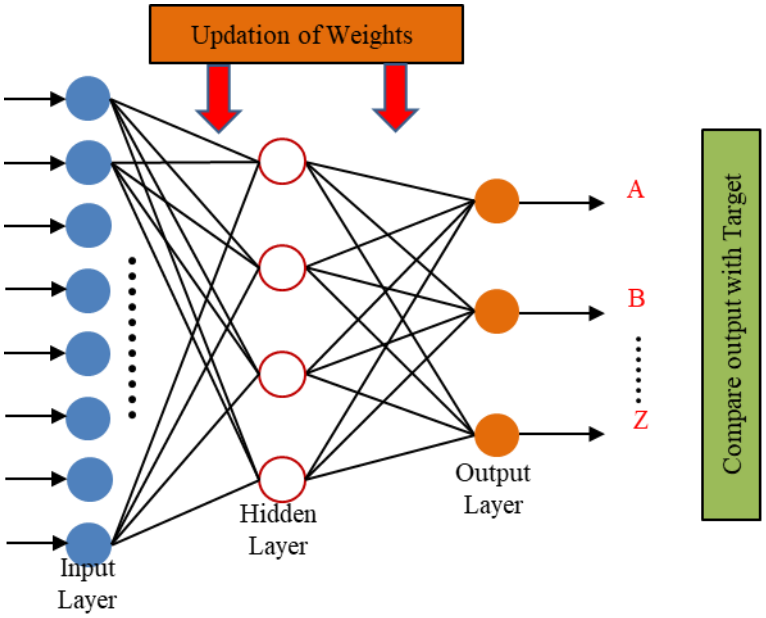

Fig. 3 Structure of ANN

In order to make the computations simpler, it was decided to use a gaussian distribution. The naive presumption is that each pixel in an image exists separately from every other image. In addition, each covariance is represented by using simply the variance of 784 pixels, which decreases the complexity of the problem from a matrix of size 784 * 784 to 784 alone. Equation 4 computes P(image | Ci).

$$P(image\ |c) = \frac{dy}{dx}\prod_{i=1}^{784}\frac{1}{\sqrt{2\pi\sigma_i^2}}\ \exp\ (-\frac{x_i-\mu_i}{2*\sigma_i^2}) \quad (4)$$

Where, μ_i is the mean value of each i[th] image pixel of a class, and σ_i^2 is the variance of each i[th] image pixel of a class.

3.3. Decision Tree

The links between features and possible outcomes are modelled using decision trees, which have a flowchart-like appearance. Figure 2 illustrates a decision tree with the root node at the top and decision nodes as internal nodes. It learns to divide data into branches based on the value of the attribute

and displays potential decision outcomes. The best attribute is chosen for splitting the data based on Gain Ratio, Information Gain, and Gini Index. Make that attribute a decision node and break the dataset up into smaller subsets. As shown in Figure 2, a decision tree is created by repeatedly repeating this process until all records belong to the same attribute value, there are no more instances or attributes, or there are no more attributes. It is simple to understand and possible to identify handwritten numerals using this tree-like structure.

3.4. Random Forest

The assembling technique-based supervised machine learning method known as random forest is particularly well-liked and frequently applied to classification issues. A decision tree is constructed using several subsets of data, and the categorization result is then decided by a majority vote. This technique produces an effective classifier from vast datasets and is resistant to outliers.

3.5. KNN

KNN determines the distance between the query and all of the dataset's accessible tuples. K closest tuples are chosen based on the computed Euclidean distance, and the label is predicted based on the labels that are used the most frequently choose the K closest tuples.

3.6. SVM

SVM is used for classification tasks with the aim of identifying a maximal margin hyperplane (decision border) in n-dimensional space (n-number of characteristics) that clearly categorises data points. For classification in SVM, the decision boundary is used. The Lagrangian formula, which provides a stable and predictable performance on untested examples of data, is a solid mathematical foundation for the margin maximisation principle. SVM has additionally proven to operate at the cutting edge in a variety of real-world applications. Kernel functions (linear, nonlinear, polynomial, gaussian, rbf, sigmoid) transform data points from the original nonlinear input space to a high-dimensional feature space where they are completely or almost linearly separable. The scalar product between two locations in a common feature dimension is what these kernel functions essentially return.

3.7. Artificial Neural Network

ANN creates a model to help computers learn similarly to how human brains do. The three layers of connected neurons that make up an ANN are the input layer, output layer, and hidden layer, as mentioned in Figure 3. The input layer directs data into the system for processing, the hidden layer uses a set of weighted inputs to compute an outcome through an activation function, and the output layer forecasts the program's outcome. Input nodes are weighted based on the significance of the attributes. Artificial neurons spread incoming data throughout the network according to the activation function used. Artificial neural networks come in a variety of forms.

3.8. CNN

The classification and recognition of objects can be done using CNN [14]. With the use of CNN, it is also able to identify faces, people, cancers, road signs, and objects [15]. CNN was employed for Arabic handwritten digit images (MADBase database) with the use of LeNet-5 [16]. Convolutional layers are the main constituents of CNN. After every convolution layer, there are learnable filters (kernels). Each filter computes the dot product to build a 2D feature map of that filter during the forward pass by being convolved over the width and height of the input images. The filter map is then given to the pooling layer. Convolution and pooling layers are alternated in the construction of the model before the fully connected layer (FC). The FC layer classifies images based on information collected from the preceding layers (convolution and pooling), as shown in Figure 4.

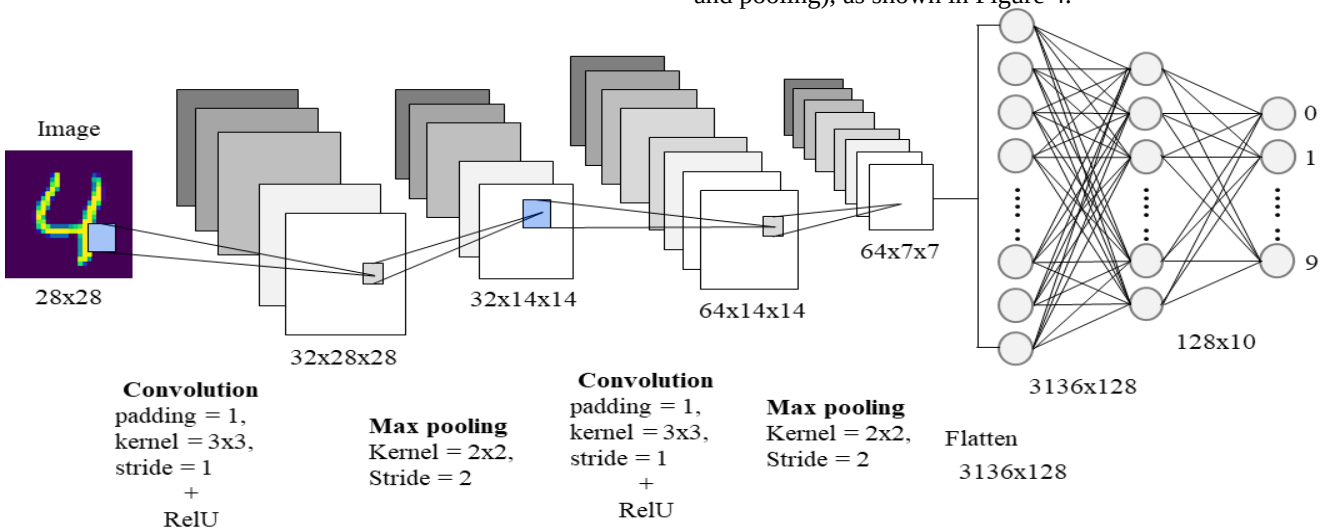

Fig. 4 CNN Structure [17]

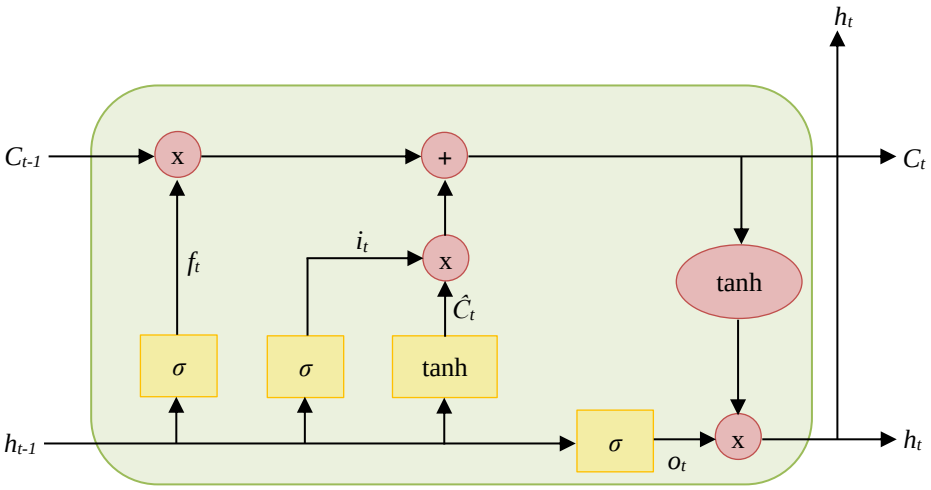

Fig. 5 LSTM Architecture [18]

3.9. LSTM

Due to its recurrent nature, which has numerous applications across many areas, Recurrent Neural Networks (RNN) provide a significant memory advantage. However, Vanilla recurrent neural networks cannot be used in real-world applications because of the vanishing gradient and inflating gradient issues. Any neural network's priority is to learn, but vanishing and exploding gradients prevent this from happening. Backpropagation gradients build up and explode or lessen and disappear, preventing weight adjustments and learning. The shortcomings of RNNs are resolved by LSTM networks [18]. A backpropagation-trained LSTM network has three gates: forget, input, and output. Figure 5 depicts the LSTM's structure.

4. Results and Discussion

Modern machine learning methods like Decision Tree, Random Forest, and KNN algorithm were used in this study to classify the MNIST dataset. Testing accuracy for decision trees was 85.2%; for random forests, it was 94.1, and for KNN, it was 96.9%.

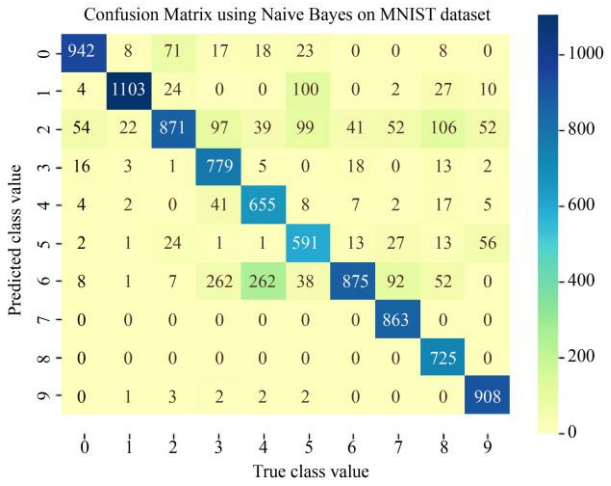

Fig. 6 Confusion matrix over MNIST using Naive Bayes

Table 1. Individual accuracy of each digit (class) obtained by Naive Bayes model

Label	Accuracy %
0	91.45
1	96.66
2	87.01
3	77.98
4	66.70
5	68.64
6	91.72
7	83.14
8	75.44
9	87.90

On MNIST, the digits were classified using the Naive Bayes technique. In order to examine the model, overall accuracy was computed, and the overall accuracy obtained was 80.6%. The accuracy obtained by individual classes is shown in Table 1. Confusion Matrix of the Naïve Bayes classifier is plotted as portrayed in Fig. 6. It is noticed that, with the exception of classes 4 and 5, the other classes produced better outcomes. Naive Bayes model's overall accuracy is 80.6 percent.

The SVM model was developed for digits classification 0 – 9 on the MNIST dataset. Both linear and nonlinear SVM were built, and the grid search technique was used for tuning the hyperparameters. The SVM model was then adjusted to produce the ideal values of C and gamma for the RBF kernel. There was a 5-fold cross-validation. A testing accuracy of 91.3 percent was provided by linear SVM.

The accuracy of the nonlinear RBF kernel testing with default parameter values was 93.4 percent. With the help of the grid search technique, it was discovered that gamma =.001 and C = 15 achieved the maximum testing accuracy of 94.4 percent while preventing the overfitting scenario and the confusion matrix is depicted in Fig. 7.

```
accuracy 0.9438888888888889
[  [1163    0    4    1    1    2    8    6    3    0]
   [0    1389    4    2    4    0    1    9    4    0]
   [1     4    1184   14    5    1    9    30   7    5]
   [0     3    15   1263   0    14   2    23   8    3]
   [1     2    20    3   1149   0    10   10   2    21]
   [2     8    3    30    4   1064  15    9    11   3]
   [8     1    3     0    3    13  1167   23   1    0]
   [4     9    10    8    12   0    0    1255  2    30]
   [5     18   17   23    8    20   5    13  1098  10]
   [5     3    2    27   21    1    1    51   3   1161]  ]
```

Fig. 7 Confusion Matrix using SVM (Nonlinear RBF) on MNIST

In this study, we further developed the CNN model and attempted to identify and categorize various handwritten numbers. For this, the MNIST digit recognizer dataset was used. The required libraries were imported, and a dataframe object of pandas was created after reading the dataset. The dataset was split, and 80% was used for training, while 20% was used for testing. The images are grayscale; therefore, the pixel values fall between 0 and 255.

In contrast to 0 to 255, CNN converges more quickly on values between 0 and 1. Consequently, normalization of the input data is carried out (Divide by 255). Then, a (28,28,1) matrix is created by reshaping the array of pixel values. The labels were encoded using just one hot encoding. To validate the model's accuracy and generalizability, 10% of the data from the training dataset were employed.

This will allow for cross-validation. The validation loss during model training will show underfitting and overfitting. The choice of the optimizer is an important part of Neural network training. The literature revealed that RMSProp converges more quickly and is more efficient; hence, RMSProp was used. It was seen during the experiment that a learning rate of 0.01 produced good effects.

To generate more images Before fitting the model, operations including flip, rotation, zoom, and crop were utilised. By extracting more images from the training dataset, data was enhanced. The model was trained using Google Colab.

The model was examined for a number of factors, including loss, accuracy, and confusion matrix. With a batch size of 112 tests and 30 epochs, accuracy was 99.98%. Fig. 7 shows the plotted confusion matrix for the MNIST classification.

Data augmentation was carried out to lessen the variance. Dropout regularisation and L2 regularisation were used to reduce variance further. More layers were added to the developed model to address bias-related problems.

The CNN model on MNIST is implemented as given below:

```
from tensorflow.keras import layers
from tensorflow.keras import models
modelmc = models.Sequential()
modelmc.add(layers.Conv2D(64,(3,3),activation='relu',
input_shape=(28,28,1)))
modelmc.add(layers.MaxPooling2D((2,2)))
modelmc.add(layers.Conv2D(128,(3,3), activation='relu'))
modelmc.add(layers.MaxPooling2D((2,2)))
modelmc.add(layers.Conv2D(128,(3,3), activation='relu'))
modelmc.add(layers.Flatten())
modelmc.add(layers.Dense(128, activation='relu'))
modelmc.add(layers.Dense(47, activation='softmax'))
modelmc.summary()
optimizer = tf.keras.optimizers.RMSprop(lr=0.001, rho=0.9,
epsilon=1e-08, decay=0.0)
```

Total trainable parameters are 3,75,727

epochs	- 30
loss	- 0.0016
training accuracy	- 99.98%
validation_loss	- 0.0412
validation_accuracy	- 99.05%

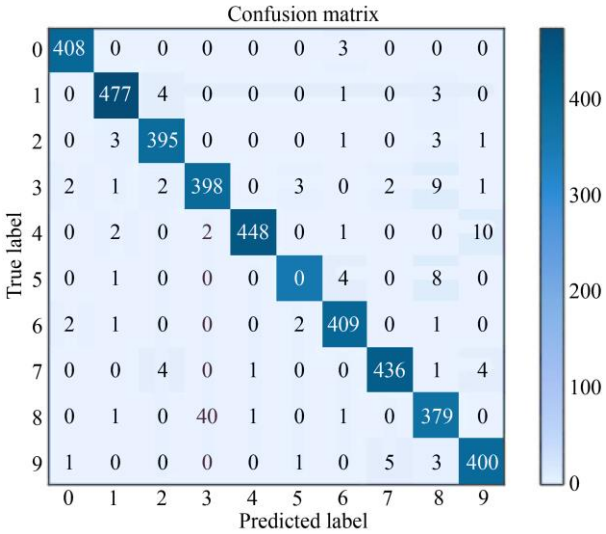

Fig. 8 Confusion matrix using CNN on MNIST

CNN model was very efficient in MNIST dataset classification, so it was decided to extend the model further on the EMNIST dataset for letters and digits classification. The model implemented on EMNIST is as follows:

```
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras import optimizers
modele = models.Sequential()
modele.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28,28,1), padding="same"))
modele.add(layers.MaxPooling2D((2, 2)))
modele.add(layers.Conv2D(64, (3, 3), activation='relu', padding="same"))
modele.add(layers.MaxPooling2D((2, 2)))
modele.add(layers.Conv2D(128, (3, 3), activation='relu', padding="same"))
modele.add(layers.MaxPooling2D((2, 2)))
modele.add(layers.Conv2D(128, (3, 3), activation='relu', padding="same"))
modele.add(layers.MaxPooling2D((2, 2)))
modele.add(layers.Flatten())
modele.add(layers.Dropout(0.5))
modele.add(layers.Dense(512, activation='relu'))
modele.add(layers.Dense(47, activation='softmax'))
modele.compile(loss='categorical_crossentropy', optimizer = optimizers.Adam(learning_rate=1e-4), metrics= ['accuracy'])
modele.fit(train_images,train_labels, epochs=20, batch_size = 20, validation_data = (test_images, test_labels))
```

Training and validation accuracy obtained by the above CNN model was 88% and 88.1%, respectively, on EMNIST dataset classification. It was ensured that the model was not overfitting, as depicted in Figure 9. The training loss and validation loss vs epochs of the developed CNN model are depicted in Figure 10.

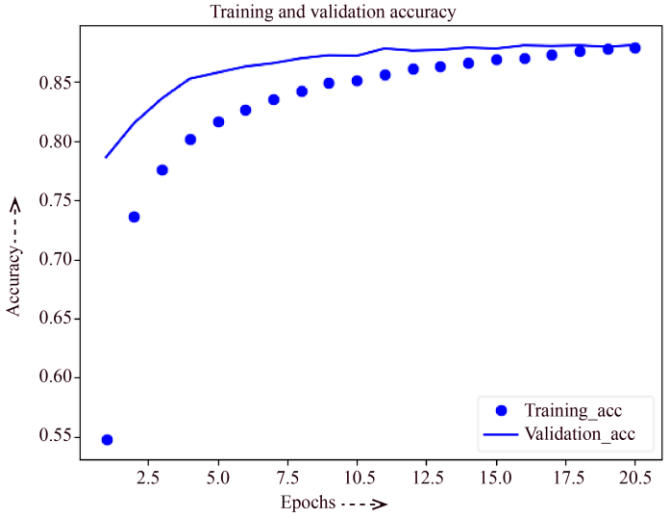

Fig. 9 Accuracy vs Epochs on EMNIST dataset by CNN Model

In this work, the LSTM model was also built for the classification of digits on the MNIST dataset.

```
modelL= Sequential()
modelL.add(LSTM(128, input_shape=(x_train.shape[1:]), activation='relu', return_sequences=True))
modelL.add(Dropout(0.2))
modelL.add(LSTM(128, activation='relu'))
modelL.add(Dropout(0.2))
modelL.add(Dense(32, activation='relu'))
modelL.add(Dropout(0.2))
modelL.add(Dense(10, activation='softmax'))
opt = tf.keras.optimizers.Adam(lr=1e-3, decay=1e-5)
modelL.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
modelL.fit(x_train, y_train, epochs=3, validation_data=(x_test, y_test))
```

Training accuracy using the above LSTM model is 94%, and validation accuracy was 97.6% on MNIST. The LSTM model was not extended for EMNIST letter classifications as the accuracy obtained by the CNN model has outperformed the accuracy obtained by LSTM on MNIST. A performance comparison of all the mentioned algorithms on the MNIST dataset is depicted in Table 2.

Table 2. Performance comparison of algorithms

S. No.	Algorithm	Dataset	Accuracy (%)
1	Decision Tree	MNIST	85.2
2	Random Forest	MNIST	94.1
3	KNN	MNIST	96.9
4	Naïve Bayes	MNIST	80.6
5	SVM (Linear)	MNIST	91.3
5	SVM (Nonlinear- RBF)	MNIST	94.4
7	CNN	MNIST	99.9
8	CNN	EMNIST	88
9	LSTM	MNIST	94

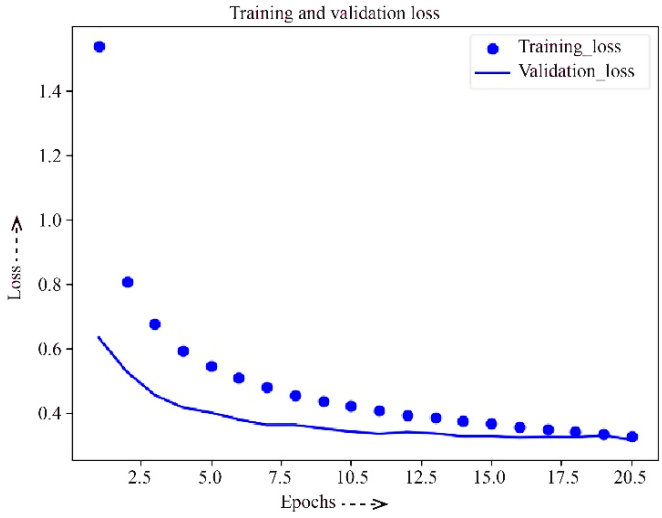

Fig. 10 Loss vs Epochs on EMNIST dataset by CNN Model

5. Conclusion

Presently, deep neural nets are widely used for image analysis and computer vision projects. Raw pixel data of the image is the input to deep neural networks, which automatically extracts the features after the training phase. Well-known machine learning algorithms like KNN, Naïve Bayes, SVM- linear and SVM – nonlinear RBF, Decision tree, and Random forest were employed over the MNIST dataset, and the accuracy obtained was compared. The finding has shown that the accuracy was enhanced using noise reduction, resizing, and cropping in the preprocessing stage.

We have developed the HDaLR system, which can recognize handwritten digits and was extended for the recognition of handwritten characters. During the experiment it has been observed that CNN has outperformed traditional machine learning algorithms.

The work can be extended by implementing letter recognition on real-time handwritten datasets. The models in this work will be validated on the Hindi and Marathi datasets in the future study.

References

[1] Farah Essam et al., "MLHandwritten Recognition: Handwritten Digit Recognition Using Machine Learning Algorithms," *Journal of Computing and Communication*, vol. 2, no. 1, pp. 9-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Alejandro Baldominos, Yago Saez, and Pedro Isasi, "A Survey of Handwritten Character Recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, pp. 1-16, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Sardar Hasen Ali, and Maiwan Bahjat Abdulrazzaq, "A Comprehensive Overview of Handwritten Recognition Techniques: A Survey," *Journal of Computer Science*, vol. 19, no. 5, pp. 569-587, 2023. [CrossRef] [Publisher Link]

[4] Kartik Dutta et al., "Improving CNN-RNN Hybrid Networks for Handwriting Recognition," *16th International Conference on Frontiers in Handwriting Recognition*, pp. 80-85, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[5] Oleksandr Voloshchenko, and Małgorzata Plechawska-Wójcik, "Comparison of Classical Machine Learning Algorithms in the Task of Handwritten Digits Classification," *Journal of Computer Sciences Institute*, pp. 279-286, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] A.C. Faul, *A Concise Introduction to Machine Learning*, CRC Press, 2019. [Google Scholar] [Publisher Link]

[7] Owais Mujtaba Khanday, and Samad Dadvandipour, "Analysis of Machine Learning Algorithms for Character Recognition: A Case Study on Handwritten Digit Recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 1, pp. 574-581, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Birjit Gope et al., "Handwritten Digits Identification Using Mnist Database via Machine Learning Models," *IOP Conference Series: Materials Science and Engineering*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Ujjwal Bhattacharya, and Bidyut Baran Chaudhuri, "Handwritten Numeral Databases of Indian Scripts and Multistage Recognition of Mixed Numerals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 444-457, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[10] Md Zahangir Alom et al., "Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1-13, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Md Shopon, Nabeel Mohammed, and Md Anowarul Abedin, "Bangla Handwritten Digit Recognition using Autoencoder and Deep Convolutional Neural Network," *International Workshop on Computational Intelligence(IWCI)*, Dhaka, Bangladesh, pp. 64-68, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yann LeCun, Corinna Cortes, and J.B. Christopher, MNIST Dataset. [Online]. Available: https://www.kaggle.com/datasets/hojjatk/mnist-dataset

[13] Gregory Cohen, Saeed Afshar, Onathan Tapson, and S.V. Andre, EMNIST. [Online]. Available: https://www.kaggle.com/datasets/crawford/emnist

[14] Rohit Rastogi et al., "Knowledge Extraction in Digit Recognition using MNIST Dataset: Dataset: Evolution in Handwriting Analysis," *International Journal of Knowledge Management*, vol. 17, no. 4, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Savita Ahlawat et al., "Improved Handwritten Digit Recognition using Convolutional Neural Networks(CNN)," *Sensors*, vol. 20, no. 12, pp. 1-18, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] 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*, pp. 566-575, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[17] Krut Patel, MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN), Towards Data Science, 2023. [Online]. Available: https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

[18] C. Olah, Understanding LSTM Networks, 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/