Digit Recognition of Handwritten Character using CNN

Chimata Triveni
Computer Science & Engineering
Narasaraopeta Engineering College
Narasaraopet, India
trivenchimata6@gmail.com

Chunchu Meghana
Computer Science & Engineering
Narsaraopeta Engineering College
Narasaraopet, India
chunchumeghana@gmail.com

Lakshmi Adimulam
Computer Science & Engineering
Narasaraopeta Engineering College
Narasaraopet, India
adimulamlakshmi910@gmail.com

Amulya Pallikonda
Computer Science & Engineering
Narasaraopeta Engineering College
Narasaraopet, India
amulyapallikonda47@gmail.com

***Abstract*—Predicting the transcribed digits and numerals is the main goal of handwritten character recognition using CNN. Because there are so many different styles, it might be challenging to identify the digits while using manual data. This model accurately predicts the images, which aids in the solution to the problem. For model training, the MNIST dataset is utilized. To improve the performance of the models, preprocessing methods like normalization are used. With great precision, the Convolutional Neural Network architecture is intended to efficiently extract patterns and spatial hierarchies from the input images. TensorFlow, Pandas, Keras, and Numpy modules are needed to do this. The images are categorized into transcribed digits based on the model's evaluation and training. This paper lays a foundation to predict the numbers.**

Keywords—* *Deep Learning, Handwritten digit Recognition, MNIST, CNN

I. INTRODUCTION

Neural networks [1] are used in deep learning [2], a branch of machine learning that addresses the drawbacks of conventional machine learning models by using neural networks. Those drawbacks include failing to classify the image and failing to recognize the audio. Deep learning algorithms automatically extract features from raw data through the use of a hierarchical framework. Owing to these characteristics, deep learning finds applicability across multiple fields. Deep learning models can recognize patterns with greater complexity.

Deep learning is based on Neural networks. Input, hidden, and output layers are present in every neural network, as Figure 1 illustrates. The input layer takes the raw data, and it can be images, text, numbers, or any other applicable format. The hidden layer, or therein can be multiple based on the requirement. These multiple layers are stacked together and serve as a foundation for neural networks. In this layer, complex computations are performed on the data that is received from the input layer. This data will pass through interconnected nodes. By processing the data through these multiple hidden layers, the network can extract intricate features and patterns. The layer that is output is the final layer. It generates the final output after receiving the information from the last hidden layer that has been processed.

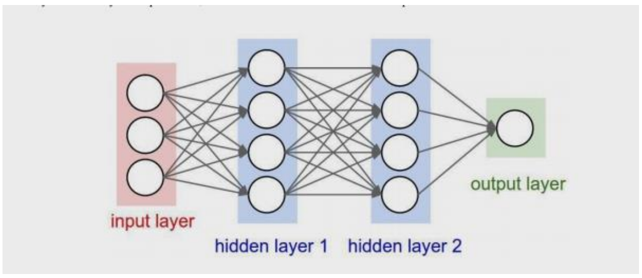


Fig. 1. Sequential Neural Network

Deep learning includes supervised [3], unsupervised [4], and reinforcement learning [5], just like machine learning. Supervised learning works when there is only labeled data. It includes artificial neural networks (ANN) [6], convolutional neural networks (CNN) [7], and recurrent neural networks (RNN) [8]. These Networks are used for specific purposes Artificial Neural Networks are used for structured and unstructured data, including tabular data, text data, and images. Convolutional Neural Networks are used for images and videos. Recurrent Neural Networks are used for audio data. Unsupervised learning includes Self Organizing Map, Boltzman Machine, and AutoEncoder.

The dataset now we use is MNIST [9] which consists of images. Now to deal with these images we can use Artificial Neural Network and Convolutional Neural Network. But ANN can't deal with hierarchies deeply so we use Convolutional Neural Network [7] instead.

There are different layers in the Convolutional Neural Networks. Convolutional, pooling, and fully connected layers are among them. Applying filters is the initial layer. The reduction of spatial dimensions is the second layer. The values are finally flattened.

The model's extreme complexity allows for a wide range of application areas. This paper implements the

Convolutional Neural Networks for image recognition. The model is built using the MNIST [9] dataset and evaluated.

II. LITERATURE REVIEW

Recent advancements in image identification and recognition using machine learning techniques, particularly deep learning algorithms, have significantly improved accuracy across various datasets. These algorithms have shown remarkable performance on tasks such as object detection, image classification, and facial recognition, revolutionizing image processing capabilities in fields like computer vision and artificial intelligence.

In a study by Abdullah et al. [10] three distinct classifiers—SVM, KNN, and CNN were compared for their performance on the MNIST dataset. The Multilayer Perceptron (MLP) faced challenges on this platform, particularly in the difference of the two numbers 9 and 6 accurately due to being stuck in local optima. However, using the CNN model on the Keras platform significantly enhanced performance, even surpassing other classifiers in accuracy.

Similarly, Gawlikowski et al. [11] explored the performance of CNN, DBF, and DNN classifiers on the MNIST dataset. Their comparative analysis revealed that the DNN model achieved the highest accuracy of 98.08%, outperforming the others. The different execution times and error rates of each classifier, however, emphasize how important it is to choose the best model depending on the demands of the particular application and performance indicators.

The method of Principal Component Analysis (PCA) was employed by Yehya Abouelnaga et al. [12] to decrease overfitting in a combined method that Involved KNN and CNN. When compared to using each classifier separately, this special combination generated a huge 0.7% gain in accuracy, which is a considerable improvement.

Researchers are actively improving CNN accuracy by minimizing errors, with recent studies showing that deeper networks perform better due to reduced overfitting to study by Pashine, Samay et al. [13] Compared to NORB and CIFAR10 on MNIST, their designs exhibit fewer errors in handwriting recognition. One study used a 3-NN arrangement on MNIST and was able to achieve a low error rate of 1.19%. Innovations like the coherent recurrent convolutional network (CRCN) are being explored to extract words from images. Methods like Ncfm (No combination of feature maps) are enhancing CNN performance on MNIST datasets, reaching 99.81% accuracy. CNN is also being applied to improve dark image enhancement and traffic sign verification models in Germany, achieving an accuracy rate of 98%.

Numerous deep learning as well as machine learning strategies have been investigated by researchers, including SVM, RFC, K-NN, MLP, and CNN, for handwritten digit recognition. This study compares by Ritik Dixit et al. [14] CNN's effectiveness with SVM and KNN, proposing CNN as a superior deep learning method using Keras for MNIST digit recognition. The suggested CNN, utilizing the RMSprop optimizer, achieves high training (99.06%) and testing (98.80%) accuracy through convolutional layer augmentation, pooling, dropout, and model hyperparameter tuning.

Image processing is one of the numerous areas in which CNN is a major player. It significantly affects a wide range of fields. Even in nanotechnologies, like the semiconductor manufacturing process used by K. B. Lee et al. [15], CNN is employed for fault diagnosis and classification. Researchers are interested in the recognition of handwritten numerals. There are a lot of papers and articles written about this topic these days. According to studies, in contrast to well-known machine learning methods such as SVM, KNN, and RFC. The maximum accuracy is produced by Deep Learning algorithms like multilayer CNN that combine Keras with Theano and TensorFlow.

Because of its great accuracy, Convolutional Neural Networks (CNNs) are frequently used in the classification of images, analysis of videos, and other applications. Several researchers are attempting to recognize sentiment in a text. CNN is used in sentiment analysis and natural language processing by K. G. Pasi et al. [16] by varying a number of parameters. A large-scale neural network may require more parameters, which could make performance goals hard to meet. Many academics are attempting to decrease inaccuracy while increasing accuracy on CNN.

Error rates are being reduced as much as possible by researchers. Error rates are being observed using CIFAR and MNIST datasets. CNN is used to produce photos with smooth blur. Using the MNIST dataset, a new model was proposed for this purpose. This method has a 98% accuracy rate and a 0.1% to 8.5% loss range. CNN is proposed as a traffic sign recognition model in Germany by J. Jin et al. [17] It suggested a 99.65% accurate quicker performance A loss function was developed that is compatible with lightweight 1D and 2D CNN. In this case, the accuracy was 93% and 91%, respectively.

III. PROPOSED SYSTEM

Handwritten digit recognition (HDR) [18] is the procedure of hand written integers. HDR has its operations in numerous fields like bank check processing and digitization. This task can be fulfilled in numerous ways. However, the proposed system focuses on completing using deep-learning neural networks.

- **Data Analysis**
- **Data visualization**
- **Preprocessing techniques**
- **Model creation and evaluation**
- **Accuracy**

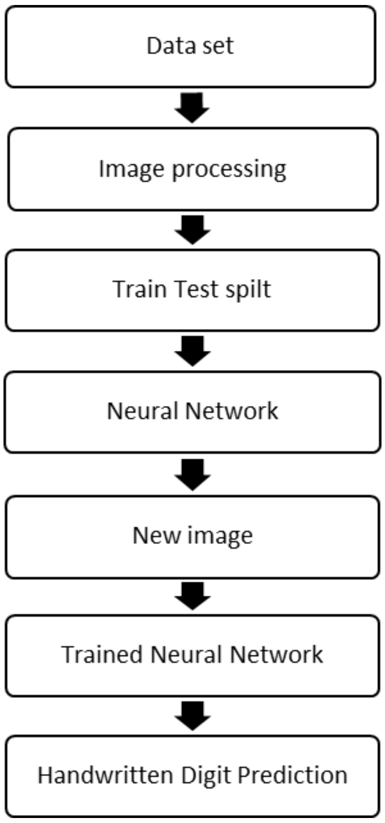


Fig. 2. Workflow for prediction of images

A. Data Analysis

We use the MNIST [9] The Modified National Institute of Standards and Technology database is where the MNIST dataset is imported. To use this first we need to use keras [19]. dataset import mnist. Then the dataset is loaded using the load data function. So that without manual intervention the training data and testing data are separated.

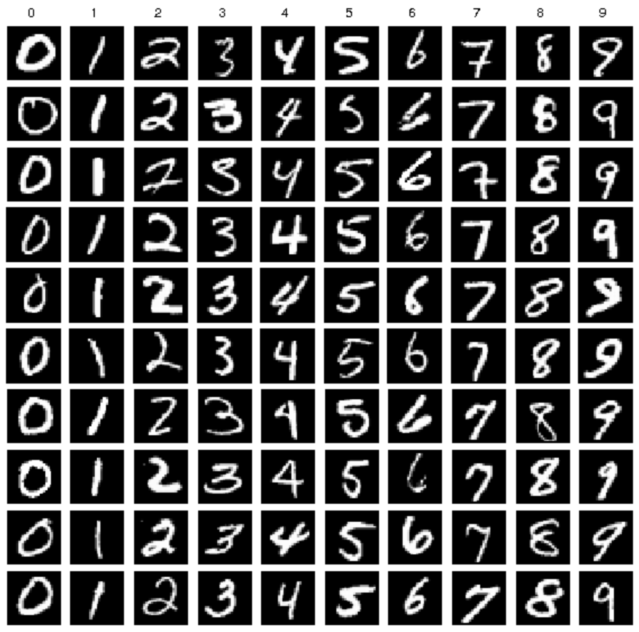
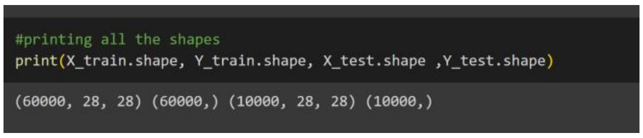


Fig. 3. A sample of the MNIST dataset

The data set collection contains 70,000 images, each with a resolution of 28 by 28 pixels. This dataset is divided into training and testing sets, with 60,000 images and 10,000 images. Since every image has 28 by 28 pixels, they combine to create an array that can be converted into a vector with dimensions of 28 by 28 by 784. Every element in the vector is a binary value that represents the pixel's intensity. These photos are used as input for the digit recognition process, and the digits 0 through 9 are represented by 10 output class labels.



```
#printing all the shapes
print(X_train.shape, Y_train.shape, X_test.shape ,Y_test.shape)

(60000, 28, 28) (60000,) (10000, 28, 28) (10000,)
```

Fig. 4. Shapes of training and testing data

MNIST [9] is a popular benchmark for deep learning tasks that assesses how well algorithms perform in the area of image categorization, especially when it comes to hand written digit recognition. Convolutional Neural Networks [7] (CNNs) are among the several deep learning models that it is ideal for testing. which are excellent at picture recognition tasks, because of their standardized format and manageable size.

B. Data visualization

In image datasets like the MNIST [9] dataset, missing or null values are not typically encountered, as each image is represented by a fixed-size array of pixel values. Therefore, the concept of missing or null values does not apply in the context of image datasets. However, other preprocessing steps such as data visualization are still crucial for understanding the dataset and detecting outliers. Data visualization involves representing the dataset graphically to gain insights into its distribution and characteristics.

In the case of the MNIST [9] dataset, which contains images of handwritten digits ranging from 0 to 9, data visualization techniques can be employed to analyze the distribution of samples across different classes. By visualizing the number of images belonging to each class using techniques like bar graphs, one can gain a better understanding of the dataset's composition. For example, upon plotting the distribution of samples, it may be observed that certain classes have a higher number of images compared to others. Understanding these class imbalances can help to adjust the modeling approach accordingly and mitigate any potential biases or complications during training. Ultimately, the insights gained from data visualization facilitate informed decision-making throughout the model development process, leading to more effective and reliable results.

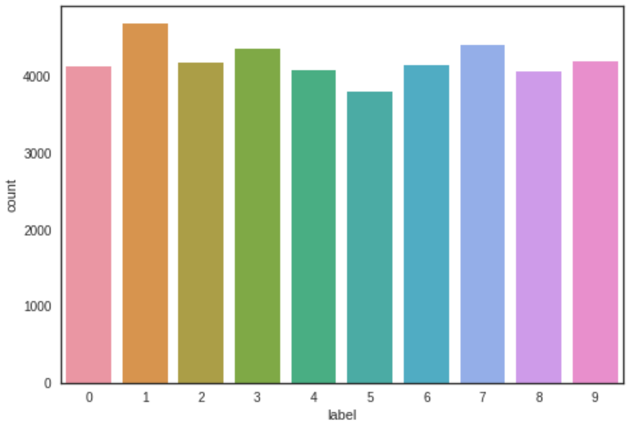


Fig. 5. Visualizing the number of samples

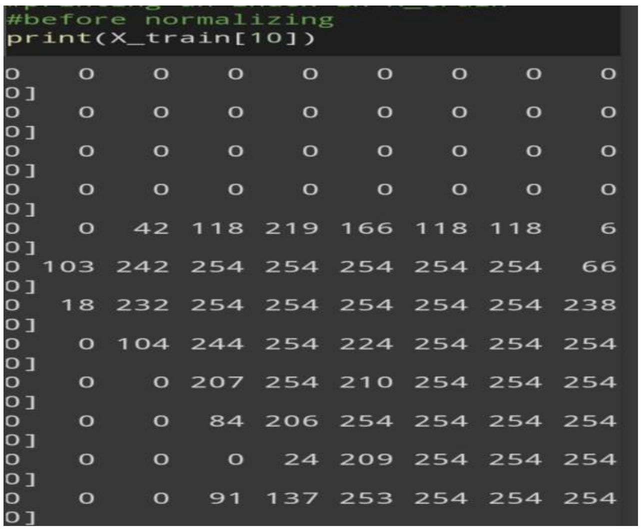
C. Preprocessing techniques

Firstly, the dataset should be collected. The mnist dataset is gathered. Secondly, the preprocessing should be done. The data is not acceptable to feed into the algorithm without preprocessing. For this the techniques used are

- Normalization
- Grayscale Conversion
- One hot encoding

- Normalization:

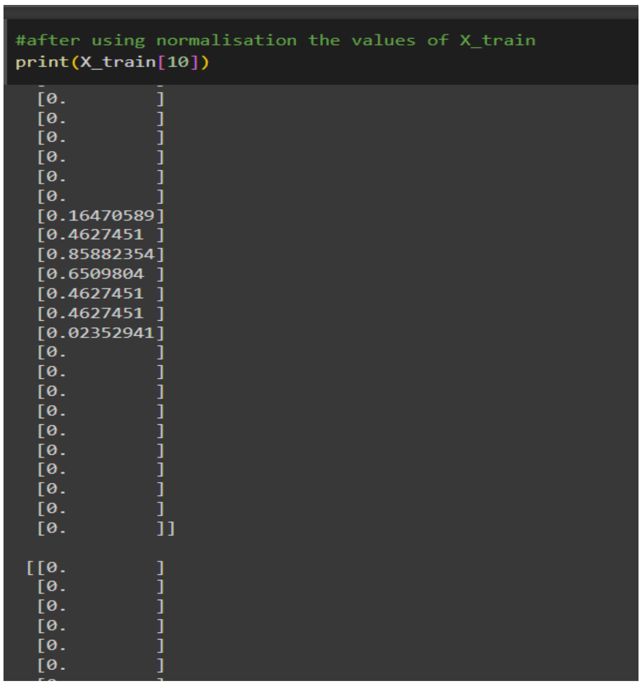
The data values in the training and testing sets range from 0 to 255. The Min-Max normalization is performed by dividing the pixel values by 255 if the photo is loaded with the pixel values represented by a range between 0 and 255. A popular method for scaling features to a range between 0 and 1 is min-max normalization. We make sure that every pixel value is inside this normalized range by dividing the pixel values by 255. Additionally, by ensuring that the model is less sensitive to the scale of the input characteristics, min-max normalization produces predictions that are more dependable and strong.



```
#before normalizing
print(X_train[10])
```

```
0     0     0     0     0     0     0     0     0
0]
0     0     0     0     0     0     0     0     0
0]
0     0     0     0     0     0     0     0     0
0]
0     0     0     0     0     0     0     0     0
0]
0     0    42   118   219   166   118   118     6
0]
0   103   242   254   254   254   254   254    66
0]
0    18   232   254   254   254   254   254   238
0]
0     0   104   244   254   224   254   254   254
0]
0     0     0   207   254   210   254   254   254
0]
0     0     0    84   206   254   254   254   254
0]
0     0     0     0    24   209   254   254   254
0]
0     0     0    91   137   253   254   254   254
0]
```

Fig. 6. Before normalizing



```
#after using normalisation the values of X_train
print(X_train[10])
```

```
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.16470589]
  [0.4627451 ]
  [0.85882354]
  [0.6509804 ]
  [0.4627451 ]
  [0.4627451 ]
  [0.02352941]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]]

 [[0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
  [0.        ]
```

Fig. 7. After normalizing

- Grayscale Conversion:

Another preprocessing technique we use is the conversion of RGB values to grayscale. By default, the dataset contains the images in RGB format. To save the computational power we convert them to grayscale i.e., the images will be in black and white colors only.

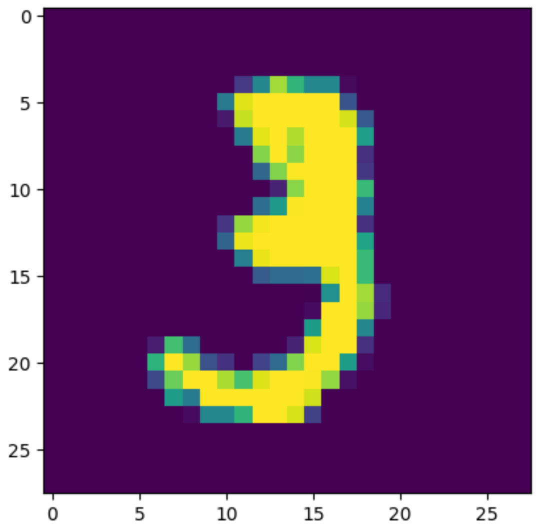


Fig. 8. A sample image of mnist dataset before conversion

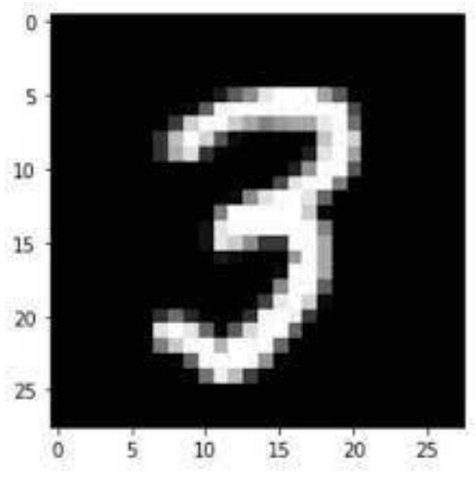
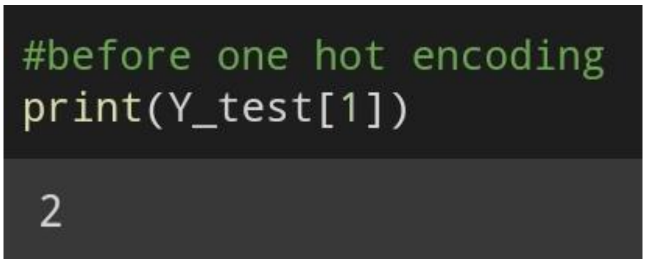


Fig. 9. The image after converting to grayscale

- One hot encoding:

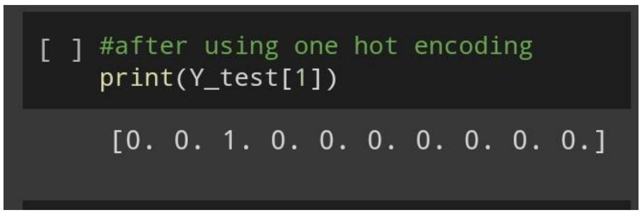
One hot encoding [20] refers to transforming the categorical input to numerical data. We use one-hot encoding to ensure correct predictions and not to prone them to Ordinality. Here the ordinality means that the model doesn't choose 0 or 9 to be the predictions in most cases because 0<1<2<3<4<5<6<7<8<9. This refers to ordinality. So to avoid these the data must undergo encoding.



```
#before one hot encoding
print(Y_test[1])
```

```
2
```

Fig. 10. Before one hot encoding



```
[ ] #after using one hot encoding
    print(Y_test[1])
```

```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```

Fig. 11. After one hot encoding

From Fig (9, 10) we can see the changes in the testing data. Before applying the value is 2. After applying one hot encoding the value is encoded to [0,0,1,0,0,0,0,0,0 0]

By using these preprocessing techniques there will be elimination of bias, overfitting [21], and underfitting [22] issues. After preprocessing, the train test split is listed in Fig 3. But while loading the data itself the train test data is split. Subsequently, the preprocessed data is input into the neural network following these procedures. Preprocessed data is fed into the neural network. Later the new image is fed into the trained neural network. After these steps, the Handwritten digit recognition [18] is done.

D. Model creation and evaluation

One supervised deep learning approach is Convolutional Neural Networks [7]. The human brain's cortex of vision served as an inspiration for these networks. These are intended for data that resemble grids, such as pictures.

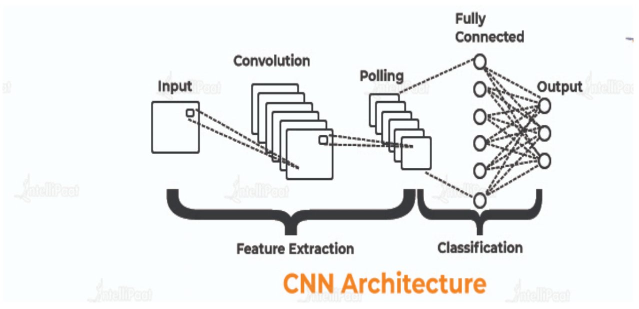


Fig. 12. CNN architecture

A CNN's design normally includes of a number of layers, such as fully connected, pooling, and convolutional layers. From the figure (12), we can see that the Convolutional layers are responsible for feature extraction. The filters, 3×3pixel size, input size, and activation functions are specified. Now pooling is done. The spatial dimensions of the feature maps acquired from the preceding convolutional layer are decreased through pooling. helping to extract the most important features while reducing computational complexity. Now as we use Deep CNN another layer of Convolutional layer is added with more filter size. Now another pooling layer is added. This process will be continued up to the desired level for hierarchical learning.

The Flatten layer is added after these two layers have been used. The output of the preceding convolutional layers is flattened into a one-dimensional vector by the flatten layer. Convolutional layers must be joined to fully connected layers through this. The completely linked layer has now been introduced. The addition of a completely linked layer with many neurons and ReLU activation

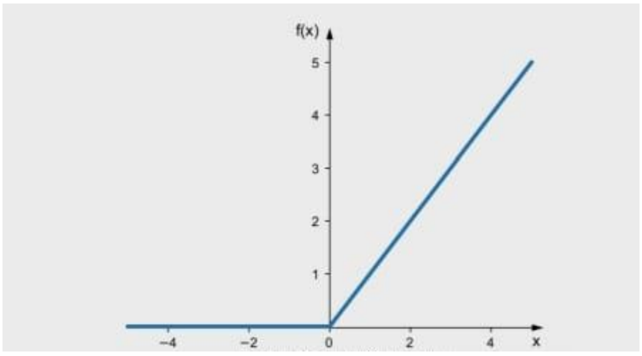


Fig. 13. ReLu activation graph

This layer is responsible for learning high-level features from the flattened input data. Finally, a fully connected output layer with some neurons is added. The activation function used here is softmax, which converts the raw output values into probabilities, indicating the likelihood of each class.

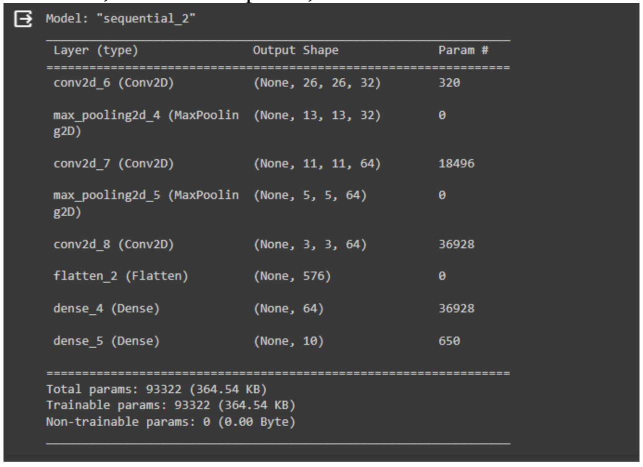
$$R(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2}$$

After the model is built the model should be compiled and evaluated. For compilation, we need an optimizer, loss function, and accuracy.

Adam is the optimizer in use. Adam is a popular optimization technique for neural network training. It can be applied to a variety of optimization situations since it adjusts the learning rate during the training process. When 'adam' is used as the optimizer, the model will utilize the Adam optimizer to update the neural network's weights while it is being trained. The model will minimize the categorical cross-entropy loss during training if 'categorical_crossentropy' is specified as the loss function.

The evaluation metric(s) to be applied during training and validation are specified by the accuracy parameter. Here, "accuracy" is defined as the parameter that determines the percentage of accurately predicted samples among all samples. During training, the model will compute and display the accuracy metric to monitor the performance of the model. After compiling ee should evaluate the model by using model.fit() method. In the training phase, the Hyperparameter tuning [23] is done. The hyperparameters include batch size, activation functions, number of epochs, and number of neurons.



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_4 (MaxPoolin  (None, 13, 13, 32)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_5 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 conv2d_8 (Conv2D)           (None, 3, 3, 64)          36928

 flatten_2 (Flatten)         (None, 576)               0

 dense_4 (Dense)             (None, 64)                36928

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 93322 (364.54 KB)
Trainable params: 93322 (364.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 14. The CNN model after building.

E. Accuracy

One essential metric for assessing the model's performance is accuracy. It is challenging to evaluate the model's training success in the absence of accuracy. By calculating the percentage of accurately classified occurrences among all assessed examples, it provides a simple means of evaluating performance.

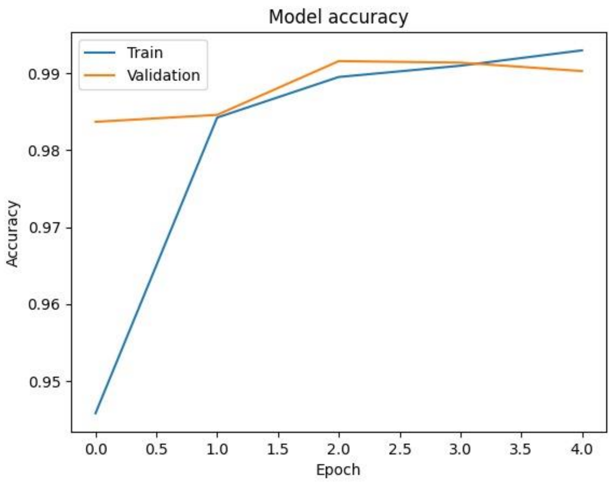


Fig. 15. Model accuracy over epochs

The machine learning model's performance over the training epochs is depicted in the accuracy graph. The number of epochs, or iterations across the whole dataset during training, is represented by the x-axis. The model's accuracy on the training and validation datasets is shown on the y-axis.

.

In Fig (15), The blue line represents the training accuracy, showing how well the model performs on the training dataset as training progresses. The red line indicates the validation accuracy, indicating the model's performance on a separate validation dataset, which helps evaluate its ability to generalize to unseen data.

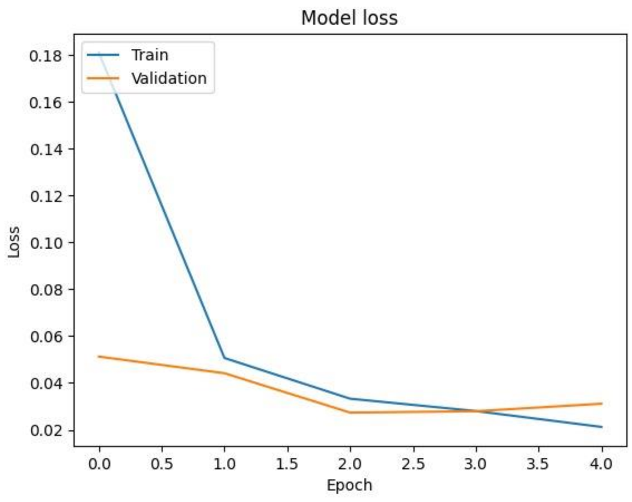


Fig. 16. Model loss over epochs

The loss graph illustrates the performance of the machine learning model in terms of loss over the course of training epochs. Similar to the accuracy graph, the x-axis represents the number of epochs, while the y-axis represents the loss, which is a measure of how well the model's predictions match the actual labels.

In Fig (16), the blue line represents the training loss, showing how the loss decreases as the model learns from the training data during each epoch. The red line represents the validation loss, indicating the loss on a separate validation dataset.

MNIST Dataset	Model	Accuracy
Existing System [24]	Simple CNN	93%
	Deep CNN	91%
Proposed System	CNN	98%

Table 1: Accuracy of the Existing and Proposed System

The table describe that the existing system is built on using simple cnn and deep cnn and their accuracies are 93% and 91% respectively. The proposed system is built on CNN where the accuracy is 98%.

In this paper, accuracy is determined using the evaluation method, and it can also be assessed through the confusion matrix [25]. These methods provide valuable perceptions of the model's predictive capabilities and as a whole performance.

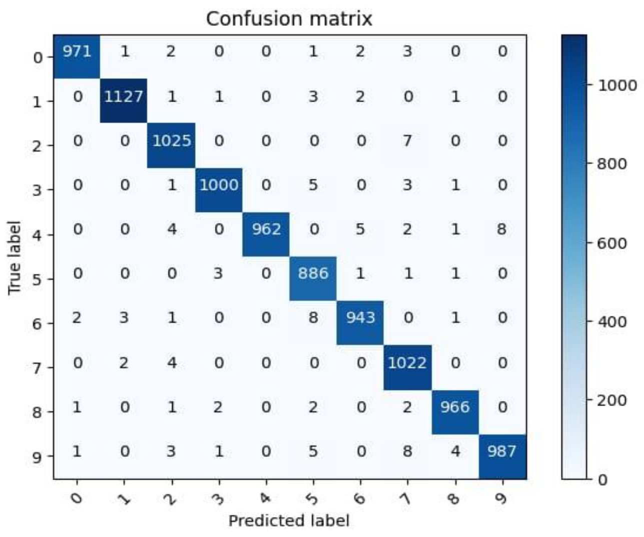


Fig. 17. Confusion matrix

The confusion matrix gives the total number of true positive, true negative, false positive, and false negative prediction generated across several classes, offering a thorough analysis of the model's performance.

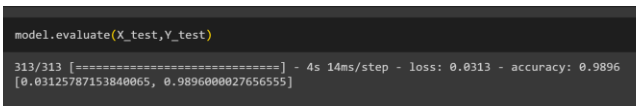
The resulting confusion matrix shows:

- the actual classes or true labels as rows.
- The projected labels or model predictions are shown by the columns.
- The number of times a genuine label matches a predicted label is indicated in each cell of the matrix.

Analyzing the confusion matrix enables us to evaluate the correctness of the model, identify any patterns of misclassification, and understand which classes are being confused with each other.

IV. CONCLUSION AND FUTURE SCOPE

In summary, this study investigated the use of convolutional neural networks for number and digit prediction. The outcomes show that the predictions were correct. The accuracy on the mnist dataset is 98.9%.



```
model.evaluate(X_test,Y_test)

313/313 [==============================] - 4s 14ms/step - loss: 0.0313 - accuracy: 0.9896
[0.03125787153840065, 0.9896000027656555]
```

Fig. 18. Accuracy of the model

Those who need to know about the unclear digits in photos might utilize this model. This guarantees that picture recognition is one of the highlights of deep learning. To guarantee even greater accuracy, this work can be extended in the future to include work on hybrid algorithms.

V. REFERENCES

[1] Rene Y. Choi, Aaron S. Coyner, Jayashree Kalpathy-Cramer, Michael F. Chiang, and J. Peter Campbell "Introduction to Machine Learning, Neural Networks, and Deep Learning, " Translational Vision Science & Technology, 2020 Feb.

[2] Amitha Mathew, P. Amudha, and S. Sivakumari, "Deep Learning Techniques: An Overview, " In Advanced Machine Learning Technologies and Applications, pp. 599-608, January 2021.

[3] Nafea, Ahmed Adil, Saeed Amer Alameri, Russel R Majeed, Meaad Ali Khalaf, and Mohammed M AL-Ani, "A Short Review on Supervised Machine Learning and Deep Learning Techniques in Computer Vision, " Babylonian Journal of Machine Learning 2024, pp. 48-55, 2024.

[4] Wani, M Arif, Farooq Ahmad Bhat, Saduf Afzal, Asif Iqbal Khan, M Arif Wani, Farooq Ahmad Bhat, Saduf Afzal, and Asif Iqbal Khan, "Unsupervised deep learning architectures, " Advances in Deep Learning, pp. 77-94, 2020.

[5] Yutaka Matsuo, Yann LeCun, Maneesh Sahani, Doina Precup, David Silver, Masashi Sugiyama, Eiji

Uchibe, and Jun Morimoto, "Deep learning, reinforcement learning, and world models," Neural Networks,2022, 152: 267-275.

[6] Roza Dastres, and Mohsen Soori. "Artificial neural network systems," International Journal of Imaging and Robotics (IJIR) 21 (2), pp.cx13-25, 2021.

[7] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," J Big Data. 2021, 8(1): 53. 2021 Mar 31. doi: 10.1186/s40537-021-004448

[8] Patel Dhruv, and Subham Naskar, "Image classification using convolutional neural network (CNN) and recurrent neural network (RNN): A review." Machine learning and information processing: proceedings of ICMLIP 2019, pp. 367-381, 2020.

[9] https://www.tensorflow.org/datasets/catalog/mnist

[10] Abdullah, Dakhaz Mustafa, Abdulazeez, and Adnan Mohsin, "Machine learning applications based on SVM classification: a review," Qubahan Academic Journal 1 (2), 81-90 (2021).

[11] Gawlikowski, Jakob, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu, "A survey of uncertainty in deep neural networks," Artificial Intelligence Review, vol. 56, no. Suppl 1, pp. 1513-1589, 2023.

[12] Yehya Abouelnaga, Ola S. Ali, Hager Rady, and Mohamed Moustafa, " CIFAR-10: KNN-based ensemble of classifiers", 66 IEEE, March 2017.

[13] Samay, Pashine, Ritik Dixit, and Rishika Kushwaha, "Handwritten digit recognition using machine and deep learning algorithms," arXiv preprint arXiv:2106.12614, 2021.

[14] Ritik Dixit, Samay, Pashine, and Rishika Kushwaha, "Handwritten digit recognition using machine and deep learning algorithms," arXiv preprint arXiv:2106.12614, 2021.

[15] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," IEEE Transactions on Semiconductor Manufacturing, vol. 30, no. 2, pp. 135-142, 2017.

[16] K. G. Pasi and S. R. Naik, "Effect of parameter variations on accuracy of Convolutional Neural Network," in 2016 International Conference on Computing, Analytics and Security Trends (CAST), 2016, pp. 398-403: IEEE.

[17] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 1991-2000, 2014.

[18] M. Wu and Z. Zhang, "Handwritten digit classification using the mnist data set," Course project CSE802: Pattern Classification & Analysis, 2010.

[19] https://keras.io/

[20] Hancock, John T., and Taghi M. Khoshgoftaar. "Survey on categorical data for neural networks," Journal of Big Data 7 (1), 1-41, 2020.

[21] Leslie Rice, Eric Wong, and Zico Kolter, "Overfitting in adversarially robust deep learning." International conference on machine learning, 8093-8104, 2020.

[22] Qipei Li, Ming Yan, and Jie Xu. "Optimizing convolutional neural network performance by mitigating underfitting and overfitting," 2021 IEEE/ACIS 19th International Conference on Computer and Information Science (ICIS), vol. -, pp. 126-131, 2021.

[23] Antunes, André, Bruno Ferreira, Nuno Marques, and Nelson Carriço. "Hyperparameter optimization of a convolutional neural network model for pipe burst location in water distribution networks," Journal of Imaging 9 (3), 68, 2023.

[24] Sakhtimohan.M,ElizabethRani.G,NavaneethaKrishnan.M.Abhinaya.M,Karthigadevi.K and P.V.Siddhartha, "Digit Recognition of MNIST Handwritten Using Convolutional Neural Networks (CNN)," 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), Coimbatore, India, 2023, pp. 328-332, doi: 10.1109/ICISCoIS56541.2023.10100602.

[25] Liang, and Jingsai, "Confusion matrix: Machine learning, " POGIL Activity Clearinghouse 3 (4), 2022.