Handwritten Digit Recognition using Convolutional Neural Networks

Pooja Sharma
Department of Data Science
Atlantic Technological University
L00171181@atu.ie

Github: https://github.com/poojavats/CNN

Abstract

The dataset involves the task of classifying handwritten digits into their respective numerical values. The goal of this abstract is to provide an overview of the dataset and its potential applications in the field of computer vision. The Digit Recognizer dataset is made up of a sizable number of labeled images that feature handwritten numbers from 0 to 9. The dataset is divided into two sets: a training set and a test set.The training set contains labeled images that are used to train models, while the test data is used to gauge how well-trained models perform. The grayscale picture representation of each image in the dataset has pixel values that range from 0 to 255.

1 Introduction

The objective of this project is to create a model that can accurately identify handwritten digits from the MNIST dataset. MNIST is a widely used dataset in machine learning, comprising 60,000 (sixty thousand)training images and 10,000(ten thousand) test images of handwritten digits(hd). Our approach involves employing Convolutional Neural Networks (CNN), a deep learning model(dl) specifically designed for image recognitions tasks, to construct a digit recognition system. Handwritten digit recognition is a crucial task in pattern recognition with various practical applications. Industries such as postal services and banking rely on automated systems to read postal codes and process checks, highlighting the significance of accurate digit recognition. CNNs have emerged as powerful tools for image recognition, including digit classification. Their ability to automatically learn and extract relevant image features makes them ideal for handwritten digit recognition. By leveraging the spatial information inherent in images, CNNs excel at capturing and analyzing patterns at different scales, enhancing the model's capability to differentiate between digits. In this project, we harness the capabilities of CNNs to develop a robust digit recognition system. We utilize the TensorFlow and Keras libraries, which provide a user-friendly interface for constructing and training deep learning models. By implementing a CNN architecture and training it on the MNIST dataset, our goal is to achieve high accuracy in accurately classifying handwritten digits.

2 Related work

Handwritten digit recognition has been extensively researched in the fields of machine learning and computer vision(cv), leading to the development of various approaches and applications. Support VectorMachines (SVMs) have been widely employed, aiming to find optimal hyperplanes for separating different classes of digits.Convolutional Neural Networks (CNNs) have emerged as state of the
Art model, leveraging convolutional layers to extract relevant features and achieve exceptional

performance on datasets like MNIST. Ensemble methods such as Random Forests and Boosting have been used to improve digit recognition accuracy by combining multiple models. Transfer learning, involving the use of pre-trained models on large-scale datasets like ImageNet and fine-tuning them on digit recognition tasks, has shown promise when training data is limited. Data augmentation techniques have been widely employed to enhance model training by providing additional variations through transformations applied to existing images.The field of handwritten digit recognition finds applications in various domains, including postal services for automated sorting, automatic form processing, check processing, digit-based captchas for security systems, and digital document archive for preservation and retrieval.Future work in this area can focus on exploring advanced CNN architectures, incorporating novel data augmentation techniques, investigating more sophisticated ensemble methods, and extending the application of digit recognition to complex datasets and real-world environments. Additionally, addressing interpretability and explainability in digit recognition models is an important direction for further research. By advancing these areas, handwritten digit recognition can continue to contribute to various practical applications, enhancing automation and efficiency in different domains.

3 Dataset and Features

In this project, we utilize the MNIST dataset, a well-known and extensively employed benchmark in machine learning, specifically for the task of recognizing handwritten digits. The dataset comprises 60,000 training images and 10,000 test images, all in grayscale format. Each image is sized at 28x28 pixels. square, representing a handwritten digit ranging from 0 to 9.The features in this dataset are the pixelvalues of each image. Each pixel value represents the intensity of the corresponding pixel, ranging from 0 (black) to 255 (white). These grayscale pixel values serve as the input to the model for digit recognition.

To preprocess the data, the pixel values are normalized by dividing them by 255, which scales them to a range between 0 and 1. This normalization step ensures that the input features have a consistentrange and helps in training the model effectively.

The dataset is divided into training and test sets, with the training set used for training the model and the test set used for evaluating the model's performance on unseen data. This separation helps in assessing the model's ability to generalize to new, unseen handwritten digits. Overall, the MNIST dataset provides a standardized and well-defined set of images for training and evaluating models for handwritten digit recognition tasks. Its simplicity and popularity make it an ideal choice for experimenting with different machine learning algorithms and techniques).

4 Methods

The project code provided above implements a Convolutional Neural Network (CNN) model for handwritten digit recognition using the MNIST dataset. Here is a summary of the methods used in the project:

1. **Data Loading and Preprocessing**: The MNIST dataset is loaded from the provided CSV files (digit.csv, train.csv, test.csv). The dataset is separated into features (x) and labels (y). The pixel values are normalized by dividing by 255.0 to bring them within the range of 0-1. The images are reshaped into a 4D array to match the input shape required by the CNN model. The labels are one-hotencoded using the to *tegoricalfunctionfromKeras*.

2. **Model Architecture**: The CNN model architecture is defined using the Sequential-API from Keras. The model consists of multiple convolutional layer with increasing filter size, followed by
Max- pooling's and batch normalization layer. The final layers include a flatten layer is to convert the 2D features to a 1D vector and fully connected dense layers. The model is compiled with categorical cross entropy's loss and the Adam-optimizers.

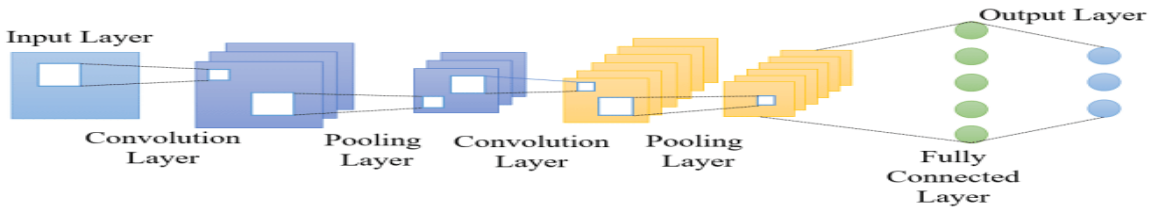

Fig 1. Basic Architecture of CNN

Ref- https://www.researchgate.net/figure/Basic-architecture-of-CNN_fig3_335086346

3. **Data Augmentation**: To prevent overfitting, data augmentation is performed using the ImageData- Generator class from Keras. It applies various transformations such as rotation, zooming, and shifting to generate additional training samples with slightly modified images.

4. **Model Training**: The model is trained using the fit function. The training data is passed through the data generator to generate augmented samples. Early stopping and learning rate reduction callbacks are used to improve model performance and prevent overfitting. The training history is stored in the history- object.

5. **Visualize some training Samples**.

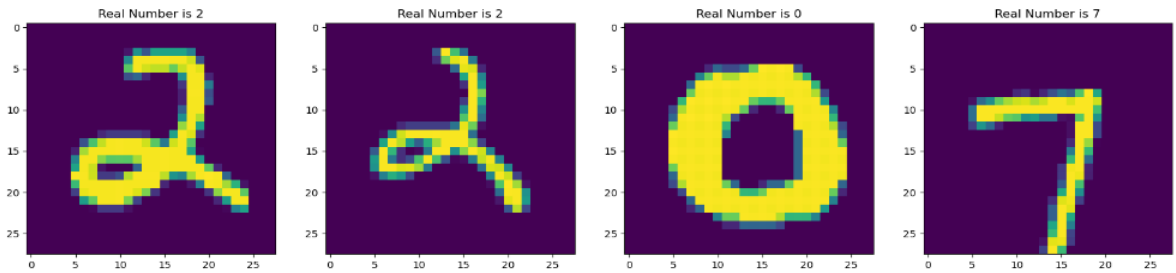

Fig 2: Training Samples Visulization

6. **Model Evaluation:** The model's accuracy and loss are visualized using line plots for both training and validation data from the history object. Additionally, a confusion matrix is generated using the test data to evaluate the model's performance on different classes.

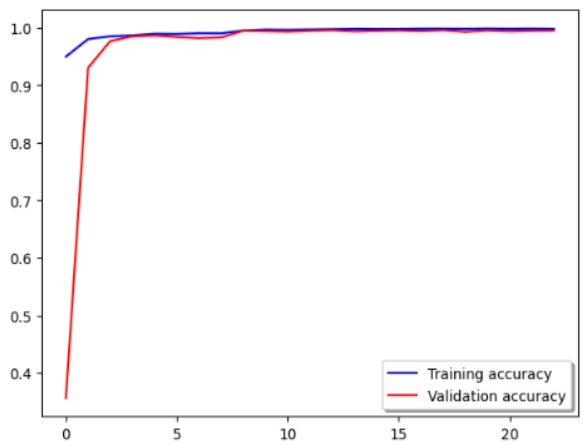

Fig 3: (a) Training-accuracy and validation- accuracy..

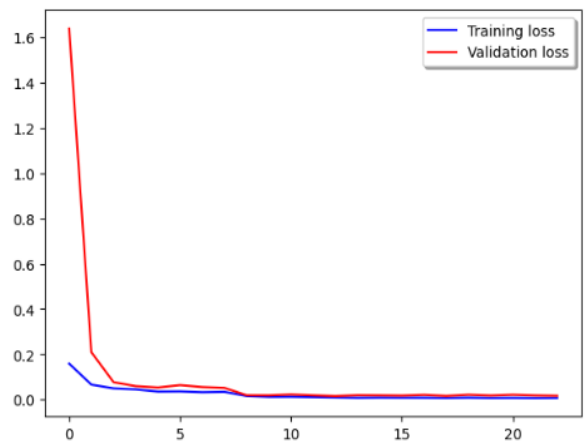

Fig: 3(b) Training- loss and validation- loss.

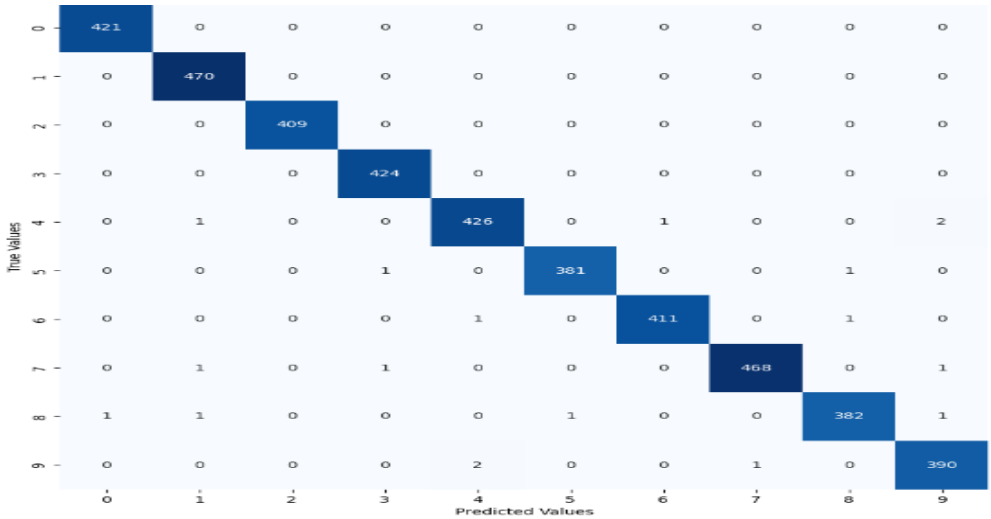

Fig4:Confusion-Matrix

7. **Prediction and Submission**: The trained model is used to here predict the labels for the test data. The predictions are converted back to class labels and stored in a DataFrame. The DataFrame is saved as a CSV file (CNN_keras_sub.csv)*forsubmission*.

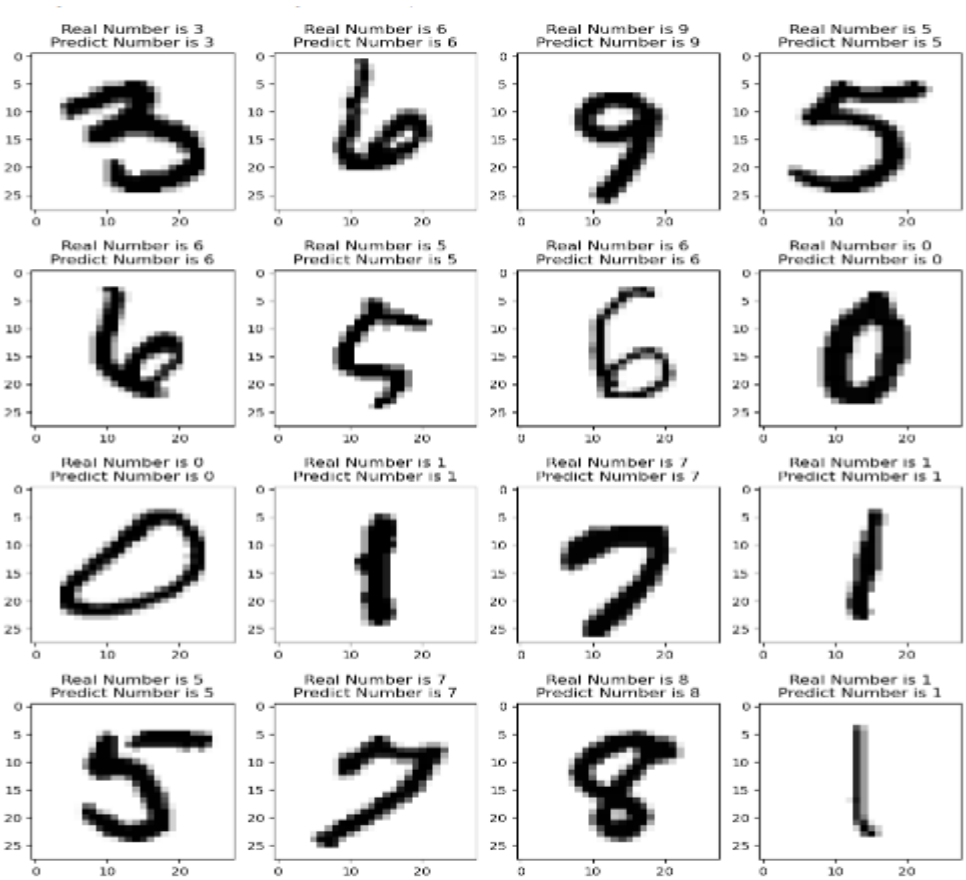

Fig 5: Predication Value

8. Experiments/Results/Discussion

Breakdowns of Output:

Epoch: In the context of machine learning and neural networks, an epoch represents a full iteration through the entire training dataset during the training phase. It signifies that the model has processed and learned from each training example once.

Epoch 1/100: The training process begins, starting with a training loss of 0.1585 and a training accuracy of 0.9495. Meanwhile, the validation loss is 1.6386, and the validation accuracy stands at 0.3567.

Epoch 2/100: The model continues to train, resulting in a decrease in the training loss to 0.0658 and an improvement in the training accuracy to 0.9801. At the same time, the validation loss(vl) decreases to 0.2091, and the validation accuracy(va) improves to 0.9302.

Epoch 3/100: The model demonstrates further progress as the training loss reduces to 0.0492, accompanied by an increase in the training accuracy to 0.9848. Simultaneously, the validation loss decreases(vl) to 0.0767, and the validation accuracy(va) rises to 0.9766.

The training eventually stops at Epoch 23 due to early stopping, which means that the model did not improve significantly after a certain number of epochs, and the weights of the model are restored to the point where it achieved the best validation performance (Epoch 13)..

loss: The value of the loss function during training. In this case, it starts at 0.1585 in epoch 1 and decreases over subsequent epochs.

Accuracy: The training accuracy of the model. It starts at 0.9495 in epoch 1 and increases over subsequent epochs.

val_loss: The value of the loss function on the validation set. It indicates how well the model is generalizing to unseen data.

val_accuracy: The accuracy of the model on the validation set(vs). lr: The learning rate used during training. It starts at 0.0010 and decreases over time due to a learning rate schedule.

Based on the provided output, the model achieves good accuracy and low loss on both the training and validation- sets(vs), indicating that it is learning and generalizing well. The training was stopped early at epoch 23 due to the implementation of early stopping, which helps prevent overfitting.

9. **Conclusion/Future Work**:

In this project, our approach involved the implementation of a convolutional neural network (CNN) model using the popular Keras and TensorFlow libraries. Our goal was to accurately classify handwritten digits using the well-known MNIST dataset, which comprises 60,000 training samples and 10,000 test samples. To prepare the dataset, we performed preprocessing steps such as normalizing the pixel values and reshaping the images. We split the data into a 90% training set and a 10% evaluation set for model performance assessment. Our CNN architecture consisted of convolutional layers, followed by max pooling and batch normalization layers, to effectively extract relevant features from the images. The output was then passed through fully connected layers, with the final layer having 10 units to represent the digits 0-9. During training, we utilized the Adam optimizer and categorical cross-entropy loss for model optimization. To combat overfitting, we employed data augmentation techniques such as rotation, zooming, and shifting using the ImageDataGenerator from Keras. Additionally, we implemented early stopping and learning rate reduction callbacks to enhance the model's generalization and convergence during the training process..

The model was trained for 100 epochs, with a batch size of 128. The training process yielded promising

results, with a peak validation accuracy of approximately 99.49% achieved after 10 epochs. The model demonstrated strong performance in both training and validation datasets, as shown by the accuracy and loss plots.

Finally, we evaluated the model's performance using a confusion matrix, which revealed high accuracy and correct predictions for the majority of digits. The model's predictions were visualized alongside the corresponding true labels for a sample of test images.

Overall, this CNN model proved to be highly effective in accurately classifying handwritten digits. It showcased the power of deep learning and convolutional neural networks in image classification tasks. The trained model can be further utilized for digit recognition in various applications.

Future work for improving the digit classification model includes exploring different CNN architectures, optimizing hyperparameters, implementing additional data augmentation techniques, leveraging transfer learning, and experimenting with ensemble learning. Conducting error analysis, developing real-time digit recognition systems, and optimizing the model for deployment on resource-constrained devices are also potential areas of focus. Lastly, applying the model to other datasets and extending it for multi-digit recognition tasks would contribute to its robustness and broader applicability in computer vision.

10. References

[1] L. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015. doi: 10.1038/nature145399.

[2] Y. Lecun and C. Cortes, "MNIST handwritten digit database," AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist./

[3] S. S. Sarwar, A. A. Nahid, and A. H. R. Tusher, "Convolutional neural networks for handwritten digit recognition," 2018 2nd International Conference on Computer Science and Engineering (ICCSE), pp. 183-188, October 2018. doi: 10.1109/ICCSE.2018.8524120.

[4] A. Krizhevskyy, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proceedings-of the 25th International Conference on Neural Information Processing Systems(NIPS), pp. 1097, December 2012.

[5] MNIST Handwritten Digit Classification Dataset. (n.d.). from https://en.wikipedia.org/wiki/MNIST_database

[6] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[7]Classification- of Handwritten Digits Using CNN. (n.d.). from https://www.analyticsvidhya.com/blog/2021/07/classification-of-handwritten-digits-using-cnn//

[8] How to Developed a CNN for MNIST Handwritten Digit Classification. (n.d.).from https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-networks-from-scratch-for-mnist-handwritten-digit-classifications