PRESENTED BY

MATHIVATHANI T
au821721104032
BE-CSE
3RD YEAR
mathivathanit23112003@gmail.com
SIR ISSAC NEWTON COLLEGE OF ENGINEERING AND TECHNOLOGY
NAGAPATTINAM – 611 102.

HANDWRITTEN DIGIT RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

AGENDA

- Introduction
- Problem Statement
- Approach
- Convolution Neural Network
- Loading the Dataset
- Processing the data
- Algorithm and deployment
- Result
- Conclusion
- References

INTRODUCTION:

The method of digitizing photographs of handwritten human numbers is called handwritten digit recognition. Because handwritten numerals can be formed with a range of flavors and are not always precise, the machine finds this work challenging. To tackle this problem, we developed HDR, which recognizes the digit existing in an image by using its image. In order to recognize handwritten numbers, we created a Convolutional Neural Network (CNN) model in this project using the Tensorflow technology. Convolutional neural networks, often known as ConvNets, are Deep Learning algorithms that can recognize different objects in an input image by giving them learnable weights and biases.

PROBLEM STATEMENT:

Handwritten digit recognition is a fundamental task in the field of image processing and machine learning. With the increasing digitization of information, there is a growing need for accurate and efficient systems that can automatically recognize handwritten digits. The problem statement for this project revolves around developing a Convolutional Neural Network (CNN) model capable of accurately classifying handwritten digits from images.

APPROACH:

- We have used Sequential Keras model which has two pairs of Convolution2D and MaxPooling2D layers. The MaxPooling layer acts as a sort of downsampling using max values in a region instead of averaging. After that, we will use Flatten layer to convert multidimensional parameters to vectors.

- The last layer has a Dense layer with 10 Softmax outputs. The output represents the network guess. The 0-th output represents a probability that the input digit is 0, the 1-st output represents a probability that the input digit is 1 and so on…

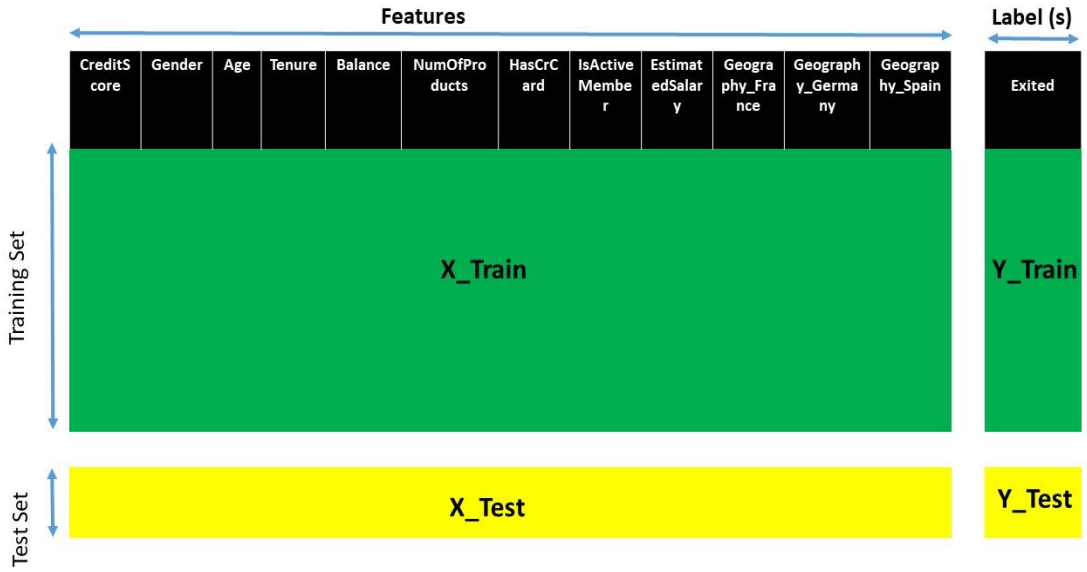
CONVOLUTIONAL NEURAL NETWORK:

- For those of you new to this concept, CNN is a deep learning technique to classify the input automatically (well, after you provide the right data).
- Over the years, CNN has found a good grip over classifying images for computer visions and now it is being used in healthcare domains too.
- This indicates that CNN is a reliable deep learning algorithm for an automated end-to-end prediction.

LOADING THE DATASET:

(X_train, y_train) , (X_test, y_test) = keras.datasets.mnist.load_data()

This dataset returns four values and in the same order as mentioned above. Also, x_train, y_train, x_test, and y_test are representations for training and test datasets. To get how a dataset is divided into training and test, check out the picture below which I used during a session.



PROCESSING THE DATA:

- Data has to be processed, cleaned, rectified in order to improve its quality. CNN will learn best from a dataset that does not contain any null values, has all numeric data, and is scaled.

- So, here we will perform some steps to ensure that our dataset is perfectly suitable for a CNN model to learn from. From here onwards till we create CNN model, we will work only on the training dataset.

- Split the dataset into training and validation sets. The training set is used to train the model, while the validation set is used to tune hyperparameters and evaluate the model's performance during training.

ALGORITHM AND DEPLOYMENT:

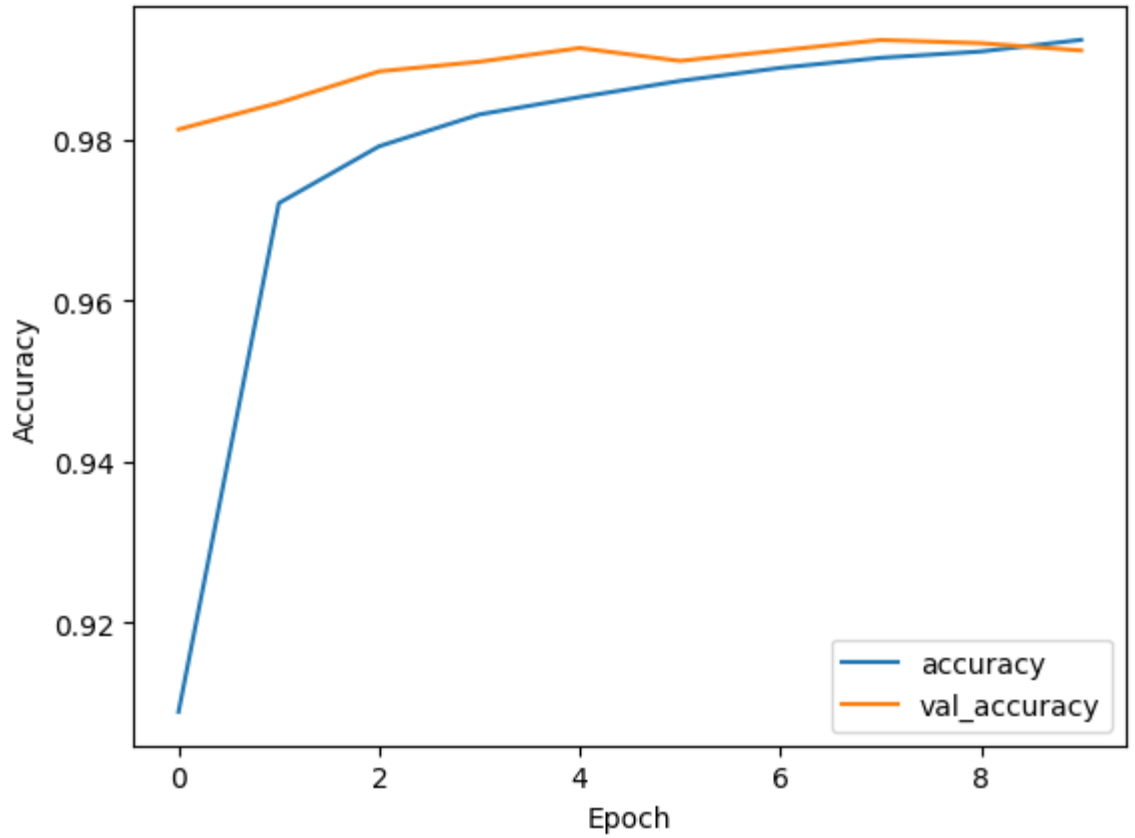
- Model Architecture: Design a CNN architecture suitable for the task of digit recognition. A typical architecture might consist of convolutional layers followed by max-pooling layers, and then fully connected layers. You can experiment with different architectures to see what works best for your problem.

- Training: Train your CNN on the training set of handwritten digits. Use techniques such as stochastic gradient descent (SGD) with backpropagation to optimize the model parameters. Monitor the performance of the model on a validation set and adjust hyperparameters accordingly to prevent overfitting.

- Evaluation: Evaluate the trained model on a separate test set to assess its performance. Calculate metrics such as accuracy, precision, recall, and F1-score to quantify the model's performance.

- Deployment: Once you have a trained model that performs satisfactorily, you can deploy it for practical use. This typically involves integrating the model into a software application or service where it can be used to recognize handwritten digits in real-time.

RESULT:



CONCLUSION:

Overall, the results obtained demonstrate the potential of CNNs in the field of handwritten digit recognition, with high accuracy rates achieved across various experiments. However, there is still room for improvement, particularly in handling more complex datasets and extending the application to recognize other types of symbols or characters beyond digits. Future research could focus on exploring more advanced CNN architectures, incorporating techniques such as transfer learning and data augmentation to further improve performance. In conclusion, while this project has made significant strides in the domain of handwritten digit recognition using CNNs, there are still exciting opportunities for further advancements and applications in the field.

REFERENCES:

1. LeCun, Y., Cortes, C., & Burges, C. (1998). The MNIST database of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist/

2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

3. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

4. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.