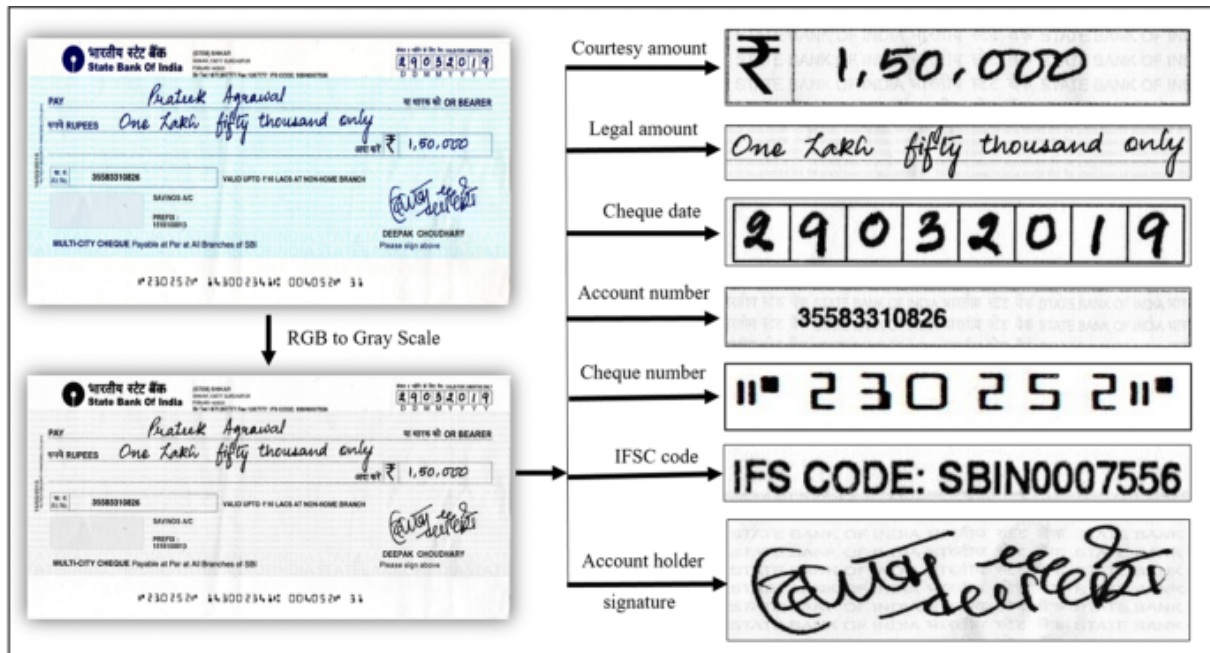


# Handwritten Digit Classification (MNIST)

Dinesh Umasankar

May 1, 2024

## Handwritten Digit Classification (MNIST)



**Figure 1:** OCR Application on an Indian Check (Agrawal et al., 2020)

## Introduction

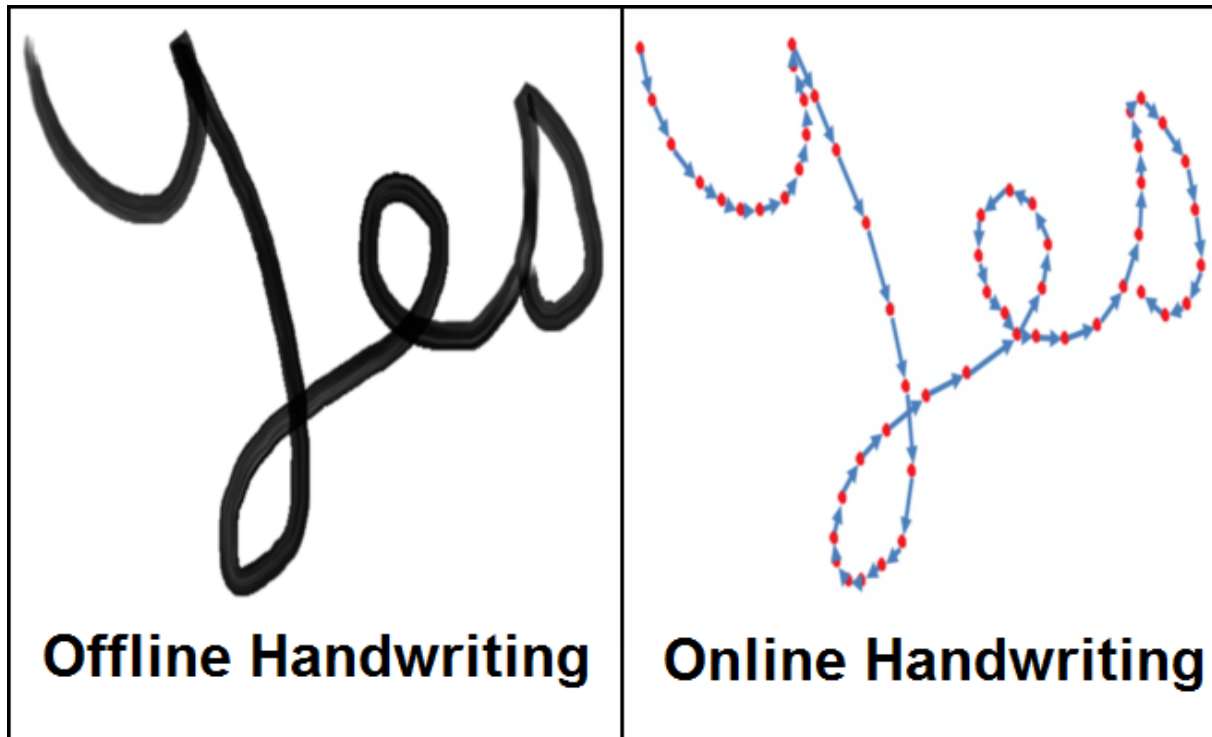
In our digital age, handwritten documents still hold a significant place in our lives. Our daily lives are filled with handwritten letters, legal documents, shopping lists and mathematical equations.

Through the marvels of optical character recognition (OCR), handwritten content has become seamless to convert into various electronic formats. With a simple picture of from our notebook, many of our phones are able to convert our notes into a text file. As our world becomes increasingly connected, the demand for rapid and accurate analysis of physical documents have increased. For Instance, millions of transactions via check are automatically processed via OCR to ensure their accuracy.

Within this project, we explore the seminal paper: “Gradient-Based Learning Applied to Document Recognition”, in order to apply various classification methods to recognize handwritten digits (Lecun et al., 1998).

## Background Context

“Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling” (Lecun et al., 1998). Our project will focus mainly on the recognition part of the process. Within the realm of recognition, specifically handwriting recognition (HWR), there are two methods: online and offline.



**Figure 2:** Showcasing offline vs online methodologies of HWR (Matcha, 2022)

1. **Online methods** - “Online methods involve a digital pen/stylus and have access to the stroke information, pen location while text is being written as shown in the figure. Since they tend to have a lot of information in regard to the flow of text being written they can be classified at a pretty high accuracy and the demarcation between different characters in the text becomes much more clear” (Matcha, 2022).
2. **Offline methods** - Offline methods involve recognizing text once it’s written down and thus won’t have information to the strokes/directions involved during writing, however, it will also have the addition of some background noise such as paper and shadows (Matcha, 2022).

This project will take an exploratory angle at the traditional classification problem presented within the seminal paper, where we will build an identification/recognition module to read a bank check. Due to time constraints, we will focus only on building the model to classify the handwritten digits within the amount specified field of a bank check, and leave the rest of the implementation to our readers as our approach can be extended to the other fields and contexts as well.

## Problem Statement

**Note:** This problem statement is slightly different from the original project submission due to my own personal interests changing, but the subject area and solution application remain the same. The focus has shifted from online HWR method from a pen computer into an offline HWR method.

**As a user of a bank, I would like to send a picture of my check in order to process a transaction.**

Within this scenario, our hypothetical bank already has the multiple module system running, except they require a handwritten digit recognition model in order to extract the amount that needs to be transacted between both parties.

As the user will be sending a picture, the project will primarily focus on handwriting recognition via the offline method.

## Objective

- Build out an identification / recognition model for the amount (in numeric form) field from a check to classify handwritten digits in the context of a bank check reader application.
- We are assuming this bank check reader application already has built the field extraction and segmentation models.

## Motivation

There are numerous physical forms and papers, and building out a digital classification system could further digital transformation of existing businesses and industries. The digitization of data presents a diverse range of opportunities for comprehensive analysis for any organization, including the ability to feed the data into a Retrieval-Augmented Generation (RAG) model to extract crucial insights. Moreover, it would serve as the initial step towards constructing a comprehensive OCR system that can be utilized regardless of the context.

## Significance

The significance of this project is to showcase the ability to convert physical documents into electronic format(s), which would increase one's capability to search and retrieve, and possibly generate new content, especially in our modern day of generative artificial intelligence. The principle(s) of this project can be applied into any application-specific or industry-specific instance to provide key insights or accelerate existing processes.

## Related Works

There are numerous previous works who have delved into classifying handwritten digits, specifically from the MNIST Dataset. From the seminal paper in 1998, Yann Lecun's team had created LeNet-5, a convolutional neural network (CNN), which had achieved an accuracy of over 98%. In addition, their team had also compared various other classification methods, notably, they used a support vector machine (SVM) that used the polynomial kernel that also achieved accuracies of over 98% (Lecun et al., 1998). As of now, 35 years after the seminal paper, the highest accuracy recorded is 99.87% achieved through an ensemble of Homogeneous Vector Capsules (HVCs) that also optimizes on the number of parameters and required amount of training epochs (Byerly et al., 2020). With recent advancements in activation functions and weight initialization, alongside different optimizations for training a neural network such as the Adam Optimizer, there have been plenty of improvements to make the classification not only just more accurate, but more efficient and faster to train (Karpathy, 2022). Furthermore, many implementations have started to create "synthetic data" by shifting, tilting, rotating the digits in a slight manner to create variations similar to human handwriting via affine and elastic distortions in order to feed their neural network models with more training data, which did result much better accuracies (Simard et al., 2003). Throughout that time, most of the further experimentation to improve accuracies have still relied on the fundamental principle(s) of utilizing deep learning to classify digits, and still use the base essence of CNNs (either in ensemble format or combined into different architectures) to achieve higher accuracies.

In terms of the scope of this project, (and what I personally understand), I will tackle this problem with my own rudimentary understanding of CNNs, and attempt to optimize my process from external research and resources.

## Data

### Data Source

“The database used to train our models comes from modifications made by the original seminal paper’s authors. The original database was constructed by combining the National Institute of Standards and Technology’s (NIST) Special Database 3 (SD-3) and Special Database 1 (SD-1), which contained binary images of handwritten digits. SD-3 was originally collected from Census Bureau employees, while SD-1 was collected among high-school students” (Lecun et al., 1998). The original black and white (bilevel) images were size-normalized and centered in a fixed size of 28x28 pixels by the original authors, and they had also created many versions that involved anti-aliasing (image interpolation) and deslanting techniques applied on the images as well (Lecun et al., 1998).



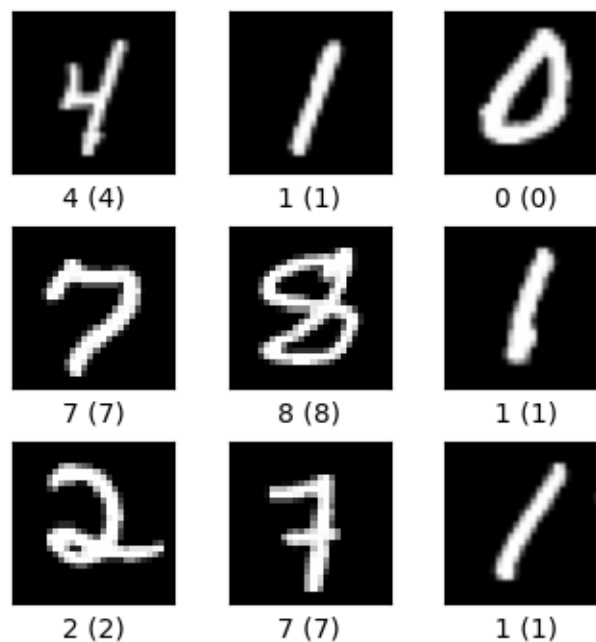
**Figure 3:** Size-normalized examples from the MNIST database (Lecun et al., 1998).

### Data Format

These images of scanned handwriting samples are from 250 people (each for the training and test set) and are grayscale and 28 by 28 pixels in size, where each pixel range(s) from 0-255 on the black to white color channel (Nielsen, 2015). The pixels will be represented as a 1-D vector of 784 entries that range from 0-255 representing pixel intensity.

There are 60,000 train samples (bilevel images) of handwritten digits from 250 people, and 10,000 test samples (bilevel) images from 250 different people as well, which will respectively be used to train up classification models to recognize handwritten digits (LeCun et al., 2010).

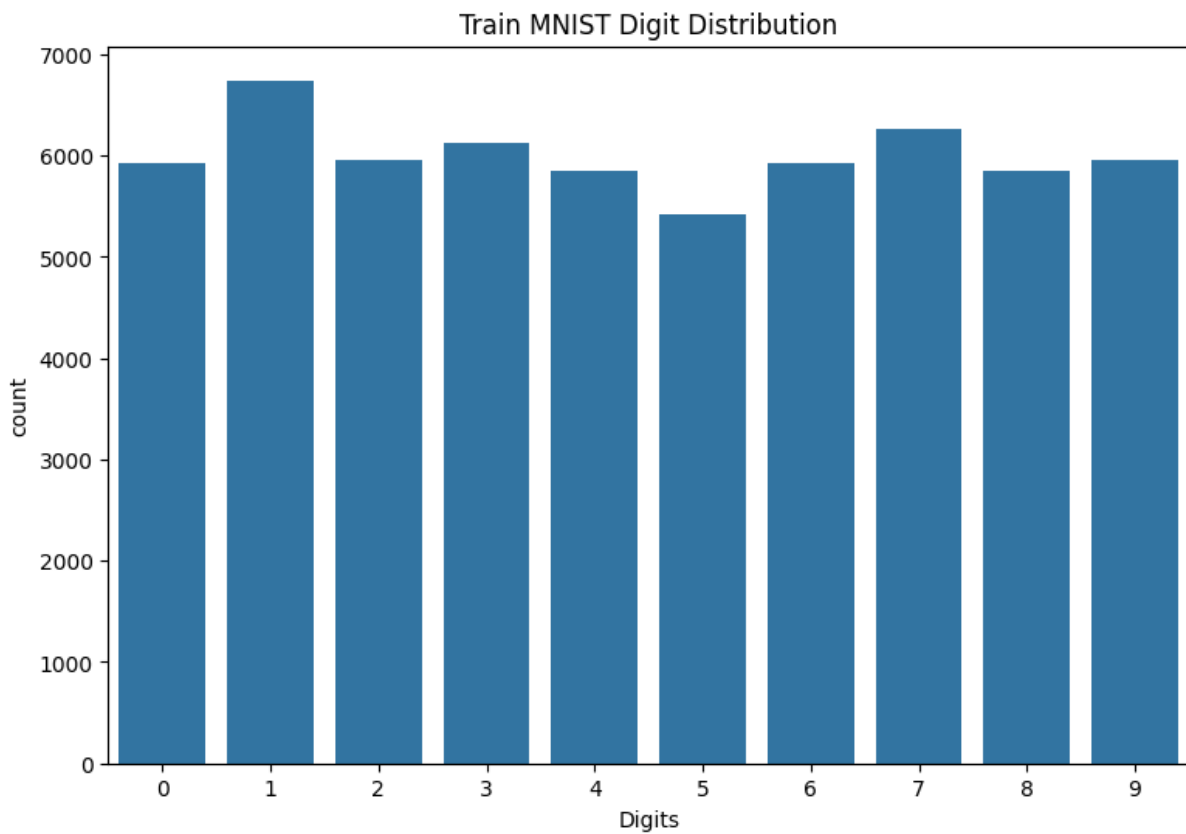
### Data Example



**Figure 4:** Graphically visualizing (all pixels) first 9 handwritten digits from the training dataset

## Data Visualization

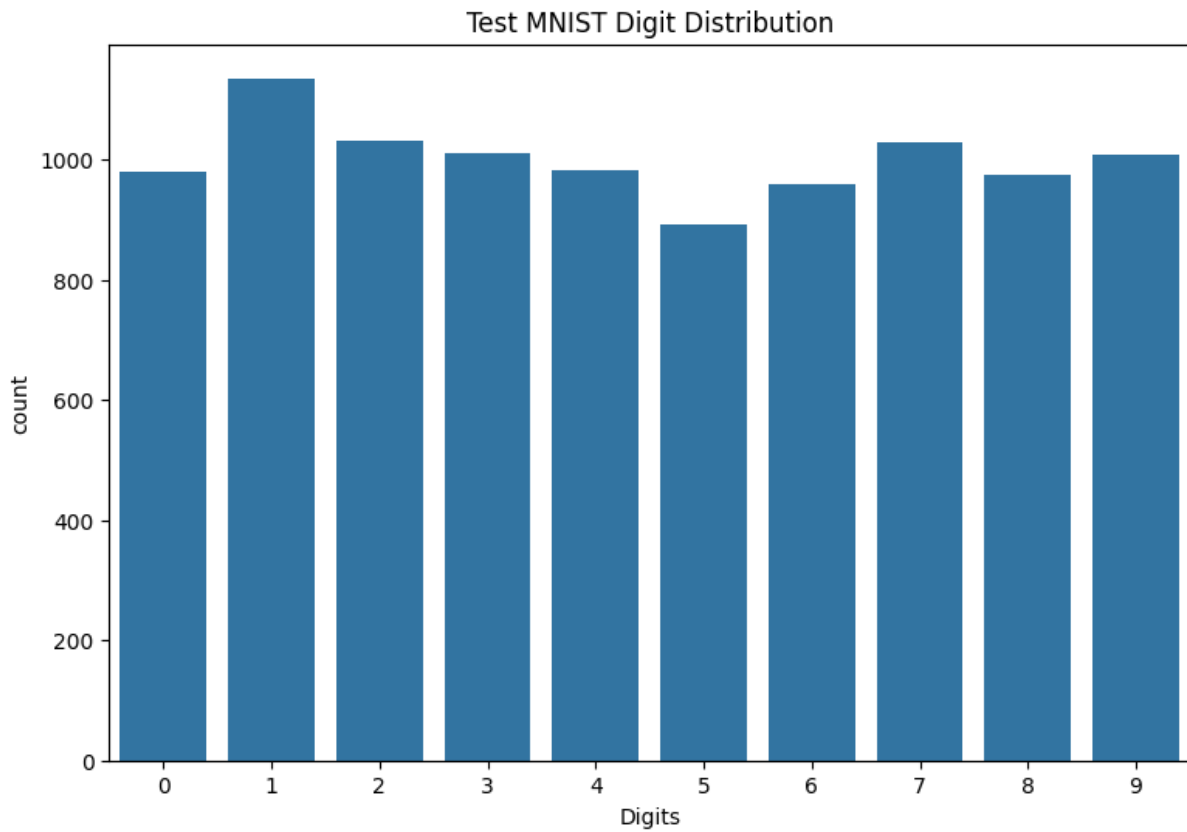
As we are dealing with a classification problem, I wanted to identify if there were any class imbalances before training a model, so I visualized the frequency of digits within the training dataset, and it appears to be evenly spread.



**Figure 5:** Training MNIST Database Digit Distribution



Along the same manner, I wanted to ensure there is no class imbalance in the testing dataset as well, and the frequencies of each digit appear to be distributed evenly as well.



**Figure 6:** Test MNIST Database Digit Distribution

## Proposed Methodologies

In order to first propose methodology towards reading digits from a bank check, we have to discuss what type of problem we are solving. As we are classifying handwritten digits from an image, our project is a classification problem, where we have multiple labels (digits 0-9) to classify.

To solve our multinomial classification problem, our project starts off with naive approaches which utilize arbitrary parameters, which are later tuned if the initial performance is desirable.

## Classification Approaches

The following approaches are initially tested with arbitrary parameters that were chosen based on prior experience, and then tuned if these initial models perform desirably.

- Logistic Regression Classifier
- Naive Bayes Classifier
- Decision Trees Classifier
- K-Nearest Neighbors Classifier
- Support Vector Machines Classifier
- Artificial Neural Networks (ANN)

## Preprocessing

As we know, the 28x28 grayscale images have pixels that range from [0, 255]. When building a neural network, typically a neuron's activation function performs better when their input is constrained between [0, 1]. As such, we will scale our pixels within our image to the range [0, 1]. We divide the original pixel value by 255, as that is the maximum intensity of a pixel within our black and white handwritten digit images.

$$pixel\_normalized = \frac{original\_pixel}{255.0} \quad (1)$$

## Frameworks

Our project uses the following list of tools to create & train up models:

- Python (3.12.12)
- Scikit-learn
- Pandas
- NumPy
- Matplotlib
- Seaborn
- Jupyter Notebook
- TensorFlow

To find more specifics on how to reproduce our results or details regarding dependency versions, please see the appendix section. Most of our initial development was done in Google Colaboratory through Jupyter Notebooks and then was transferred to our local development environments.

## Procedures

## Experiments

### Data Division (Training / Testing)

Our dataset contains a total of 70,000 images, and we decided to split into 60,000 images dedicated towards training and the remaining 10,000 images towards testing, as the original paper had split the dataset in the same way ([Lecun et al., 1998](#)).

### Parameter Tuning

### Evaluation Metrics

- Accuracy
- F1-Score
- Recall
- Precision

## Results

### Analysis

### Conclusion

### Limitations

### Unresolved Issues

GPU-Based SVM by cuML notes that it doesn't work best on sparse datasets, but to verify if this statement is true, and comparing it with other GPU based SVM implementations (thunderSVM).

### Future Direction

We could extend into tackling this problem with Convolutional Neural Networks, and applying better hyperparameter tuning on the traditional ANN (MLP) models.

## Appendix

Source code can be found at [GitHub](#), and is recommended to run in Google Colaboratory within a GPU-based environment to reproduce results.

## References

- Agrawal, P., Chaudhary, D., Madaan, V., Zabrovskiy, A., Prodan, R., Kimovski, D., & Timmerer, C. (2020). Automated bank cheque verification using image processing and deep learning methods. *Multimedia Tools and Applications*, 80(4), 5319–5350. <https://doi.org/10.1007/s11042-020-09818-1>
- Byerly, A., Kalganova, T., & Dear, I. (2020). *No routing needed between capsules*. <https://doi.org/10.48550/ARXIV.2001.09136>
- Karpathy, A. (2022, March). *Deep neural nets: 33 years ago and 33 years from now*. <https://karpathy.github.io/2022/03/14/lecun1989/>
- Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>
- LeCun, Y., Cortes, C., & Burges, C. (2010). MNIST handwritten digit database. *ATT Labs [Online]*, 2. <http://yann.lecun.com/exdb/mnist>
- Matcha, A. C. N. (2022, June). *How to easily do handwriting recognition using machine learning*. Handwriting Recognition with ML (An In-Depth Guide); Nanonets. <https://nanonets.com/blog/handwritten-character-recognition/>
- Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press. <http://neuralnetworksanddeeplearning.com>
- Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings*. <https://doi.org/10.1109/icdar.2003.1227801>