# MNSIT Digit Recognition Using Manual Feature Extraction

**Claudio Perinuzzi**
Department of Computer Science
City University of New York, Queens College
Flushing, United States

May 15, 2025

**Abstract**

The classification of handwritten digits poses significant challenges due to the inherent nature of varied writing styles. Many classification architectures have been explored, with Convolutional Neural Networks (CNNs) being the go to deep learning architecture for classification of many images today. These deep learning architectures automate feature extraction and are really good at it. However, these models have a higher level of complexity due to the nature of having many hidden layers[5]. As a result, they are not always necessary for simple classification tasks. Instead of using a CNN for feature extraction, specific image processing techniques can be used, leading to an increase in time efficiency and lower complexity. This paper explores two specific image processing techniques for feature extraction and the use of a machine learning model to classify handwritten digits. Specifically, a simple kernel average convolution discussed here[6] and the Sobel edge detection algorithm[7] are used to extract features to train a Support Vector Machine (SVM). A pipeline utilizing these algorithms is created to process data that have already been normalized from the MNIST dataset to classify handwritten digits [1]. We investigate this simple architecture and evaluate its overall and per-class accuracy in classifying the ten digits (0-9). This architecture achieves an overall accuracy score of 95.32% in the test set, revealing that CNNs are not always necessary for image classification tasks.

## 1 Introduction

The recognition of handwritten digits, while seemingly simple, represents a challenge in both image processing and classification. Its importance comes from a wide range of practical applications, including automated license plate reading for traffic enforcement, mail sorting, and bank check processing [2]. The MNIST dataset has become a widely adopted benchmark for evaluating the performance of numerous algorithms for this task. Image processing techniques can be used to extract features from the MNIST dataset. As such, the goal of this work is to explore two image processing techniques for manual feature extraction and evaluating their impact on the classification accuracy of an SVM model.

Convolutional Neural Networks (CNNs) have become the gold standard for image classification in recent years. They have repeatedly demonstrated their ability to extract useful and relevant features for image classification tasks. However, they are computationally expensive and may not be necessary for already small, normalized and well-centered images, like that is found in the MNIST dataset. Similarly, introducing a prepossessing step on new digits can be employed instead of using raw pixel values as input. Simpler models may offer a better alternative in efficiency with respect to time and complexity. The purpose of this paper is to investigate image processing techniques to extract relevant features derived from pre-processed 16x16 images by looking at the neighborhood of pixel information as well as edges, rather than relying on more complicated feature extractions of a CNN. These handpicked relevant features will eventually be fed into a Support Vector Machine (SVM) machine learning model for classification.

## 2 Literature review

The handwritten digit classification problem has been studied extensively with the introduction of the MNIST dataset. This data set has become a standard benchmark for evaluating different algorithms and machine learning models. [1]

Generally, image recognition can be performed by utilizing simple image processing techniques that can extract different and meaningful features of an image [3]. Classification models like SVM rely on relevant features for accurate predictions. They are therefore, bounded by the best features to be trained on. However, it is not always known which features are relevant and there has been some discussion on finding methods for identification of relevant features [4].

Previously, feature extraction was performed by humans before deep learning was introduced. Before deep learning, Artificial Neural Networks (ANNs) were used and considered to be a very good machine learning model in the early 2000s. These models relied on manually extracted features from humans and were bounded by a human's ability to pick the best features for the model to be trained on. The issue here lies in a human's ability to pick the best features. Many of the hand picked features may be irrelevant, which can affect the performance of an ANN.[5]

With the advancement of deep learning, manual feature extraction has been abstracted away from humans. While this is beneficial, deep learning models are not shallow with respect to how many layers they have. The number of layers for a deep learning architecture can be much higher than an Artifical Neural Network. As such, image classification requires many layers for a deep learning model. For example, convolutional neural networks perform feature extraction on a very large number of convolutional layers. Therefore, the complexity of using a CNN is usually large and affects the performance when a large number of layers is used. [5]

This paper aims at evaluating and picking the best features for an SVM model instead of using a Convolutional Neural Network.

## 3 Methodology

### 3.1 Algorithm 1: Combining Average and Edge Features

The goal is to combine two simple image processing techniques over the input images to pull out features for the SVM machine learning model. To do this, the 1D image data is first rearranged into a 16x16 2D array which is useful for indexing into certain neighborhood regions of the image. Then methods for extracting average pooling features and edge detection features are used. These features are combined into one feature vector that will be used for the SVM model. The following algorithm describes the combination of features that will be learned by the SVM machine learning model.

**Algorithm 1** Combined Feature Extraction (Average + Edge)

```
1. Initialize an empty list to store features
2. For each image:
3.    Reshape the 1D image vector into a 2D $16 \times 16$ array
4.    Extract average pooling features using kernel size = 2 (Algorithm 2)
5.    Extract edge detection features using kernel size = 3 (Algorithm 3)
6.    Concatenate the average and edge features into a single 1D vector
7.    Append the combined feature vector to the features list
8. Convert the list of feature vectors into a NumPy array
9. Return the feature array
```

### 3.2 Algorithm 2: Average Pooling

To extract features from average pooling, a single convolution operation is performed across the 16x16 grid using a 2x2 kernel. This convolution calculates the average value within each 2x2 region of the image and stores these averages in a smaller 8x8 feature map. Finally, this 8x8 feature map is flattened into a 1D vector, which is then normalized and will be added to a set of features for the SVM machine learning model to learn.

**Algorithm 2** Average Pooling

```
1. **Calculate** output size of feature map
2. **Initialize** an empty feature map (size 8x8)
3. **For each** row:
4.   **For each** col:
5.     **Calculate** the starting and ending row indices
6.     **Calculate** the starting and ending column indices
7.     **Extract** the kernel region of the 16x16 array using the indices
8.     **Calculate** the average of the kernel region
9.     **Store** the average in the feature map[row][col]
10. **Flatten** the feature map into a 1D vector
11. **Normalize** the flattened feature vector
12. **Return** the normalized feature vector
```

## 3.3 Algorithm 3: Edge Features

To extract edge features, we first define the horizontal and vertical edge detections kernels and pad the original 16x16 image with a 1-pixel border of zeros which ensures that the kernels detect edges along the border as well. The Sobel operator masks are used on 3x3 regions and the sum is taken for the horizontal and vertical gradient directions. This is then used to calculate the gradient magnitude using Pythagorean Theorem which is used to represent the edge strength of the current pixel. After the gradients are calculated, we apply average pooling of the gradient magnitude, flatten it and normalize the vector. This is then returned to Algorithm 1 which will be added to a set of features for the SVM machine learning model to learn.

**Algorithm 3** Edge Feature Extraction using Sobel Operator and Average Pooling

```
1. Pad the 16x16 image with a 1-pixel border of zeros to create an 18x18 array
2. Initialize an empty 16x16 array for gradient magnitudes
3. Define horizontal ($G_x$), vertical ($G_y$) and 45 degree ($G45$) Sobel kernels
4. Enhance contrast of the image
5. For each pixel $(i, j)$ in the 16x16 image:
6.   Extract the $3 \times 3$ region around pixel $(i, j)$ from the padded image
7.   Compute $g_x = \sum(G_x \cdot \text{region})$
8.   Compute $g_y = \sum(G_y \cdot \text{region})$
9.   Compute $g_{45} = \sum(G_{45} \cdot \text{region})$
10.   Compute gradient magnitude: $M(i, j) = \sqrt{g_x^2 + g_y^2 + g_{45}^2}$
11. Initialize an empty output array of size $\frac{16}{\text{kernel\_size}} \times \frac{16}{\text{kernel\_size}}$
12. For each block $(r, c)$ in output feature map:
13.   Extract the corresponding $k \times k$ region from $M$ (where $k$ = kernel size)
14.   Compute the mean of the region
15.   Store the mean in output feature map at $(r, c)$
16. Flatten the output feature map into a 1D vector
17. Normalize the vector to range [0, 1]
18. Return the normalized feature vector
```

## 4 Implementation & Experimentation

### 4.1 Description of Dataset

The data set used to evaluate the proposed algorithm is the MNIST data set which contains a total of 9,298 images of handwritten digits (0-9). Approximately 22% of the data were set aside for the test set (2007 test samples) leaving 7291 training samples left. The data here are normalized handwritten digits that were scanned from US postal service envelopes. The digits in this dataset have been deslanted and their sized normalized which has resulted in 16x16 grayscale images [1].

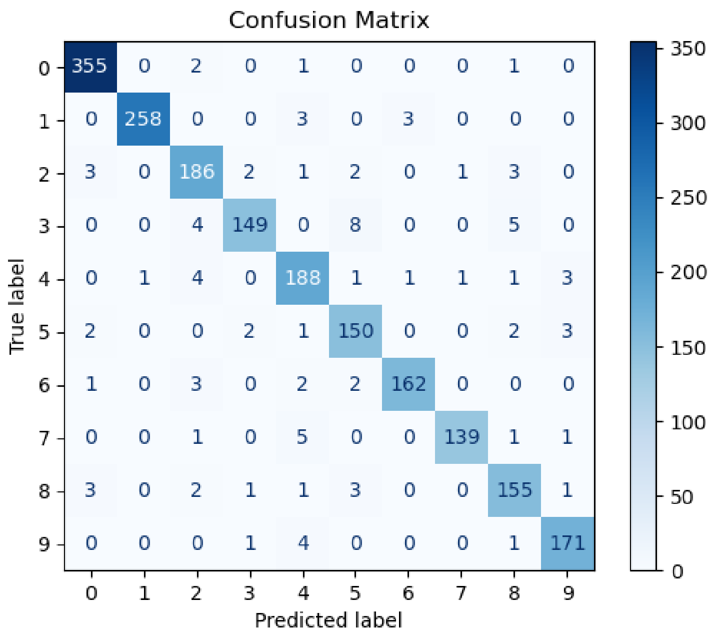
### 4.2 Validation

The SVM model was trained from the manual features on the train set which includes 78% of the data or 7291 training samples. After training, the SVM model was evaluated against the test set which includes 22% of the data or 2007 samples. The performance is discussed below.

## 4.3 Performance

Employing the aforementioned algorithm results in the following per-class accuracy and overall accuracy followed by the confusion matrix:

- Overall: 0.95
- Class 0: 0.99
- Class 1: 0.98
- Class 2: 0.94
- Class 3: 0.90
- Class 4: 0.94
- Class 5: 0.94
- Class 6: 0.95
- Class 7: 0.95
- Class 8: 0.93
- Class 9: 0.97



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 355 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 258 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | 0 |
| 2 | 3 | 0 | 186 | 2 | 1 | 2 | 0 | 1 | 3 | 0 |
| 3 | 0 | 0 | 4 | 149 | 0 | 8 | 0 | 0 | 5 | 0 |
| 4 | 0 | 1 | 4 | 0 | 188 | 1 | 1 | 1 | 1 | 3 |
| 5 | 2 | 0 | 0 | 2 | 1 | 150 | 0 | 0 | 2 | 3 |
| 6 | 1 | 0 | 3 | 0 | 2 | 2 | 162 | 0 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 | 5 | 0 | 0 | 139 | 1 | 1 |
| 8 | 3 | 0 | 2 | 1 | 1 | 3 | 0 | 0 | 155 | 1 |
| 9 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 1 | 171 |

Figure 1: Confusion matrix between the true and predicted labels. The model achieves a 95.32% accuracy in the test set. The highest error count in the figure is 8 errors between the digits 5 and 3 suggesting that the model may mistake these digits with each other more so than others. It also suggests that the digits 5 and 3 are similar in shape/features.

### 4.4 Results

The architecture described here was able to achieve an overall accuracy score of 95.32% on the test set.

Removing features reduces the accuracy of the model. For example:

- Using only average pooling features (Algorithm 2) resulted in an accuracy of 0.9527.
- Using only edge detection features (Algorithm 3) resulted in an accuracy of 0.9098.
- Using the raw normalized input without any feature extraction resulted in an accuracy of 0.9472.

Overall, it has been shown that convolutional neural networks are not always necessary for image classification tasks, especially when they are simple, like classifying digits from the MNIST dataset. Additionally, including more features that represent images in different ways helps improve the models accuracy. Including both the average pooling features and edge detection features has boosted the overall accuracy score from 0.9472 (raw normalized input) to 0.9532. Although the increase in accuracy score was small, we have shown that using different features from the input data yields a better classification accuracy.

## 5 Future Work

The architecture proposed here has only been evaluated in terms of accuracy and per-class accuracy using a Support Vector Machine model. The method in this paper still needs to be evaluated against a CNN in terms of complexity and efficiency. Additionally, a limitation of the model is handling inference of new digits that are off center. Although the model was not evaluated on digits that were off centered, it is most probable that the model will have a hard time classifying such digits because the MNIST dataset is cleaned in such way to have mostly centered digits. Data augmentation could be performed to move some digits off the border, but this was out of the scope of the main purpose of this paper. As such, this is one interesting future direction of this work, by training the model of some semi-out of border digits. Finally, for inference to work on new data, a preprocessing pipeline that matches MNISTS preprocessing steps also needs to be implemented. This is also another future direction that can be explored.

Overall, the main purpose of this paper was to focus on using simple image processing techniques to extract features, and to evaluate those features for classification. If time permitted, additional image processing techniques would have been explored to evaluate the model accuracy. However, this is one of the drawbacks to manually picking features, in that it is not always known which features will work the best and it takes time to meaningfully pick the best features.

## References

[1] LeCun, Y., Cortes, C., and Burges, C, “The MNIST database of handwritten digits”, 1998, Retrieved from `http://yann.lecun.com/exdb/mnist/`

[2] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[3] Rani, Neetu. (2017). Image Processing Techniques: A Review. Journal on Today’s Ideas - Tomorrow’s Technologies. 5. 40-49. 10.15415/jotitt.2017.51003.

[4] Sanz, H., Valim, C., Vegas, E. et al. SVM-RFE: selection and visualization of the most relevant features through non-linear kernels. BMC Bioinformatics 19, 432 (2018). https://doi.org/10.1186/s12859-018-2451-4

[5] D. J. Hemanth, "Automated feature extraction in deep learning models: A boon or a bane?," 2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Semarang, Indonesia, 2021, pp. 3-3, doi: 10.23919/EECSI53397.2021.9624287.

[6] https://stackoverflow.com/questions/72229909/handwritten-digit-recognition-without-deep-learning-techniques

[7] https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm