Assignment 2 - Documentation

Handwritten Number Recognition

**Intelligent Systems**

Faculty of Automation and Computer Science

2022-2023



Submitted by:

**Sarolta Jakab-Gyik**

**Noémi Veres**

group – 30433

Contents

[Abstract 3](#_Toc136689569)

[Introduction 4](#_Toc136689570)

[Related Work 5](#_Toc136689571)

[The Single-Digit Model 6](#_Toc136689572)

[The Multiple-Digit Model 9](#_Toc136689573)

[Tests 11](#_Toc136689574)

[Conclusion 12](#_Toc136689575)

[References 13](#_Toc136689576)

Abstract

Handwritten number recognition plays a crucial role in various applications, ranging from automated form processing to digit-based authentication systems. One of the key achievements in the field of handwritten number recognition is predicting single-digit numbers with high accuracy. The success in achieving high accuracy for single-digit predictions has leads to extending this progress to the recognition of multi-digit numbers. This study focuses on developing an effective approach for recognizing handwritten numbers with up to three digits.

The proposed method uses deep learning techniques, specifically convolutional neural networks (CNNs), to extract meaningful features from handwritten digit images. The CNN model is trained on a large dataset (10000 examples) of labelled handwritten digits, enabling it to learn certain patterns and generalize well to unseen examples.

The trained CNN model achieves good performance in recognizing handwritten numbers up to three digits. For a balanced dataset where each number appears with equal frequency, the baseline accuracy for random guessing would be 1/1000 (0.001). In this case, achieving an accuracy of 0.62 with our model is significantly better than random guessing, but the model can be further improved to obtain better results.

Introduction

In recent years, deep learning techniques, particularly convolutional neural networks, have revolutionized the field of image recognition. CNNs have demonstrated exceptional performance in various image-related tasks, including image classification.

Compared to traditional machine learning approaches, CNNs capture spatial relationships within images. They employ specialized convolutional layers that apply filters to extract local patterns and features, followed by pooling layers that aggregate information and reduce spatial dimensionality. This hierarchical feature extraction process allows CNNs to learn hierarchical representations of the input images, effectively capturing both low-level details and high-level semantic information.

By training CNN models on large datasets of labelled handwritten number images, these models can learn to automatically extract relevant features and recognize the underlying patterns that distinguish one number from another. Additionally, the ability of CNNs to capture spatial dependencies allows them to handle variations in writing styles, sizes, and orientations commonly found in handwritten numbers.

This study proposes a handwritten number recognition model that utilizes CNNs to recognize numbers with up to three digits. Our model aims to achieve accurate predictions with a good performance.

Related Work

Over the years, many approaches have been proposed to handle the challenges associated with accurately recognizing handwritten numbers.

Traditional approaches to handwritten number recognition often relied on features such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), and Local Binary Patterns (LBP), which were commonly used to capture distinctive characteristics of handwritten digits. Classifiers such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Decision Trees were then employed to classify the extracted features. While these traditional methods achieved moderate success, they did not handle the complexity and diversity of handwritten digits well.

Convolutional neural networks revolutionized the field of handwritten number recognition. CNNs have shown excellent performance in various image recognition tasks.

LeNet-5, introduced by LeCun (1998), was one of the pioneering CNN architectures applied to handwritten digit recognition. It used a simple yet effective convolutional architecture and achieved impressive results on the MNIST dataset. Building on this success, further studies focused on improving network architectures and training techniques to enhance recognition performance.

Later, deeper architectures such as AlexNet, introduced by Krizhevsky (2012) and VGGNet, introduced by Simonyan and Zisserman (2014) appeared, which demonstrated improved accuracy by taking advantage of deeper network structures. These architectures employed additional layers and parameters, enabling them to learn more complex representations of handwritten digits.

Moreover, researchers explored various data augmentation techniques to enhance the robustness of CNN models. Augmentation methods, such as rotation, scaling, translation, and elastic deformations, were employed to simulate real-world variations in handwriting styles and improve generalization capabilities.

The Single-Digit Model

In our project, we focused on handwritten number recognition. Before moving on to multiple-digit numbers, we made a model for single-digit recognition using the MNIST dataset (for some examples see *Figure 1*).

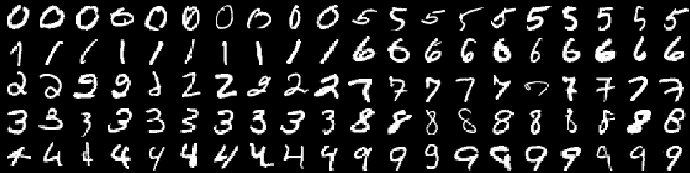


Figure 1 Examples of digits from the MNIST dataset

We started by loading the MNIST dataset using the tf.keras.datasets.mnist.load\_data() function. This function loads the dataset and splits it into training and test sets. The training set contained 60000 images, and the test set contained 10000 images. The images are represented as 28x28 matrices, and the pixel values are of type uint8. The labels corresponding to the images are integers ranging from 0 to 9.

The input images were reshaped into arrays of dimension 1, being aware that this method results in the loss of some spatial information.

The model architecture consisted of an input layer, a rescaling layer to normalize the pixel values, a dense layer with 512 units and ReLU activation function, and an output layer with 10 units and softmax activation function. The model was compiled with the RMSprop optimizer, a learning rate of 0.0001, sparse categorical cross-entropy loss, and accuracy as the evaluation metric.

We trained the model on the training dataset for 10 epochs with a batch size of 32. The training process showed a steady improvement in both loss and accuracy over the epochs. After the final epoch, the model achieved an accuracy of 0.9807 on the training data.

To evaluate the model's performance, we tested it on the test dataset and obtained a test accuracy of 0.9756. This high accuracy demonstrates the effectiveness of the model in recognizing handwritten digits.

Furthermore, we conducted an experiment where we randomly selected an example from the test set and visualized the original image along with the predicted label and the true label (see *Figure 1*). The model demonstrated its ability to make accurate predictions on unseen data.

A picture containing screenshot, diagram, square, pixel

Description automatically generated

Figure 2 Random example

In addition to individual predictions, we also generated a confusion matrix (see *Figure 2*) to gain a comprehensive understanding of the model's performance. The confusion matrix provided insights into the true labels and predicted labels for all the images in the test set. We can see, that in almost all the cases the predicted value corresponds to the true label.

Overall, the results indicate that the model achieved a remarkable accuracy of 0.97. This lays a strong foundation for further exploration and expansion of the model to handle multi-digit number recognition tasks.

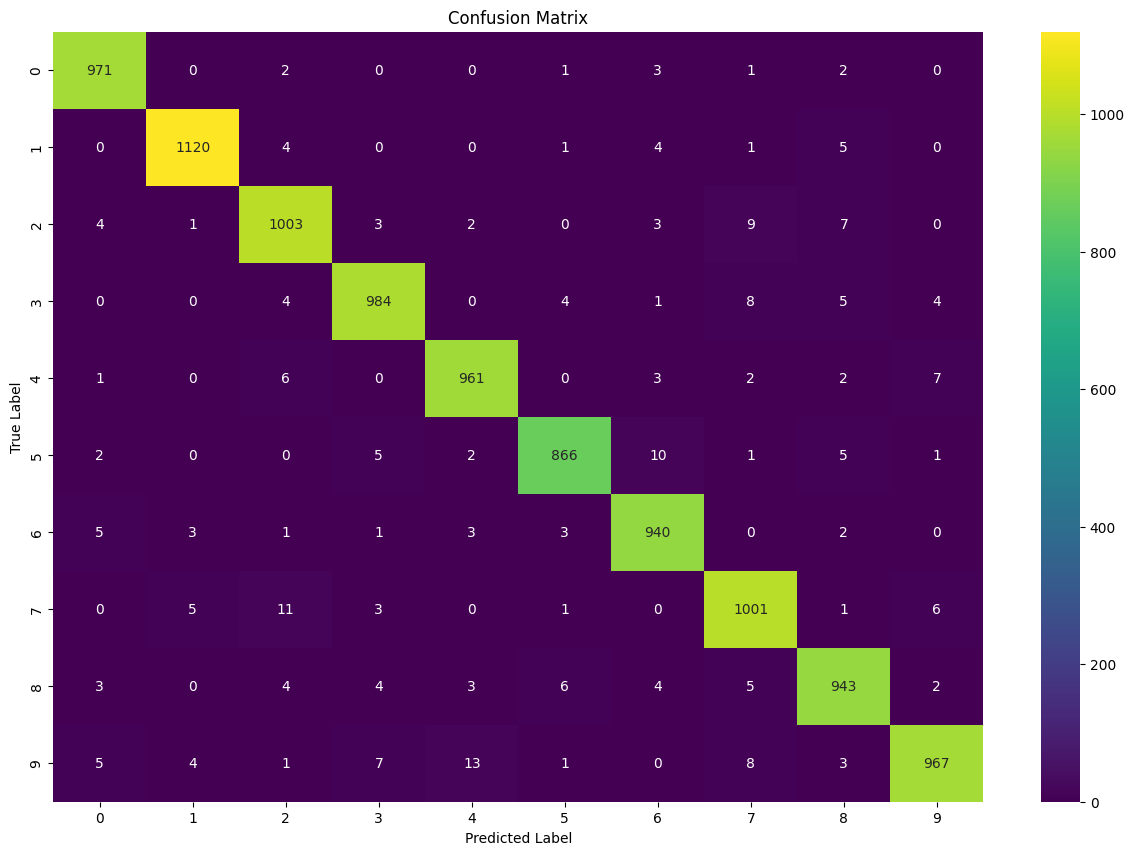


Figure 2 Confusion Matrix for our Single-Digit Model

The Multiple-Digit Model

We extended our approach to handling images containing multiple digits in the multiple-digit model. The images were read from the "Dataset" folder and sorted by filename. To extract the number from the filename, a custom function was used. The images were converted to numpy arrays and then added to the data list.

Next, the labels were read from a CSV file and converted into a numpy array. The labels and images were correctly paired, as confirmed by visualizing an example image and its corresponding label (see *Figure 2*).

A screenshot of a computer program

Description automatically generated with low confidence

Figure 3 Image with its corresponding label

To ensure uniform dimensions for all images, padding was added. The maximum width and height of the images were determined, and white pixels were inserted into the matrices representing the images to match the maximum dimensions. After applying padding, random images were checked to verify that the padding was correctly inserted.

The labels were converted from strings to integers for further processing. The data was split into training and testing sets using the train\_test\_split function, with a test size of 0.2.

All the data and labels were converted into numpy arrays. An example image from the training set and its corresponding label were visualized to ensure proper alignment.

The multiple-digit model was successfully prepared, and the labels and images were correctly ordered for training.

We built a Convolutional Neural Network (CNN) model for our multiple-digit recognition task. The model architecture consisted of several layers:

* Input layer: Sized according to the maximum width and height of the images, with a depth of 1 for grayscale images.
* Rescaling layer: Normalizes the input by dividing all grayscale values by 255 to ensure they range between 0 and 1.
* Convolutional layers: with 64 and 128 filters/neurons respectively, using a 3x3 kernel size and ReLU activation function.
* Max pooling layers: Reduce the input size by half in height and width while preserving the depth.
* Flattening layer: Converts the multidimensional matrices into a one-dimensional vector.
* Dense layers: Fully connected layers with 2048 and 1000 neurons respectively, using ReLU activation.
* Output layer: The final dense layer with 1000 neurons, representing the numbers from 0 to 999, using the softmax activation function.

We compiled the model using the Adam optimizer and sparse categorical cross-entropy as the loss function. The model summary provided an overview of the layer configuration and the number of parameters in each layer.

Next, we trained the model using the fit function with a batch size of 128 and 10 epochs. The training process showed the training and validation loss, as well as accuracy for each epoch.

Tests

After training, we evaluated the model on the test set, obtaining a test accuracy of 0.6145. While this accuracy may be lower than desired, it can be further improved by fine-tuning the model's architecture and parameters. However, due to limitations in computational resources, we decided to conclude our experiments at this point.

We also realized that our dataset is not balanced, most of the numbers represent only one digit. This may lead to an uneven distribution of training examples for each number, which can impact the model's performance. To address this issue, we can consider implementing data augmentation techniques or resampling methods to balance the dataset. Data augmentation involves applying random transformations to the existing images, such as rotation, translation, scaling, or adding noise. This can help create additional variations of the images and increase the diversity of the training set.

Overall, we iteratively adjusted the model architecture and parameters, achieving improvements from an initial accuracy of 0.3 to a final accuracy of 0.62.

Conclusion

This project aimed to develop a model for multiple-digit recognition using a convolutional neural network. The process involved several steps, including data pre-processing, model architecture design, training, and evaluation. While the achieved accuracy of the model may not be ideal, there were valuable insights and lessons learned throughout the project.

The initial dataset consisted of grayscale images of digits, and various techniques were applied to prepare the data for training. These included sorting the images, padding them to a consistent size, and extracting labels from a CSV file. Exploratory data analysis was performed to visualize the images and ensure the labels were correctly paired.

The model architecture was designed with convolutional layers, max-pooling layers, a flattening layer, dense layers, and an output layer. The choice of layer types, filter sizes, and activation functions was based on common practices for image classification tasks. The model was compiled with appropriate loss functions.

Training the model involved fitting it to the training data for a specified number of epochs. The model's performance was evaluated using validation data and tested on a separate test set. The achieved accuracy on the test set was around 62%, indicating room for improvement.

Several challenges were encountered throughout the project, including the unbalanced nature of the dataset, which resulted in uneven distributions of training examples for each digit. This issue can impact the model's performance, and potential solutions such as data augmentation and resampling were discussed to address the class imbalance.

In the further development of the project, it would be beneficial to explore these techniques and other methods for improving the model's performance. Experimenting with different architectures, hyperparameters, and optimization strategies could also yield better results. Additionally, expanding the dataset with more diverse and balanced examples would be valuable.

Despite the lower accuracy achieved, this project provided valuable insights into the process of building and training a CNN for multiple-digit recognition. It highlighted the importance of data pre-processing, architecture design, and careful evaluation. The project also emphasized the challenges of working with unbalanced datasets and the potential impact on model performance.

Overall, this project lays the foundation for future enhancements and improvements in multiple-digit recognition. With further iterations, fine-tuning, and exploration of advanced techniques, it is possible to achieve higher accuracy and develop a more robust model for practical applications.

Resources

<https://www.sciencedirect.com/science/article/abs/pii/S0925231217319112>

<https://keras.io/api/datasets/mnist/>