Macau University of Science and Technology

School of Computer Science and Engineering
Faculty of Innovation Engineering

SE482 Data Science Final Project

Handwritten Digits Recognization

Liu Zhankeng	2009853A-I011-0105
Mao Yifan	2009853E-I011-0084
Wang Yiye	2009853L-I011-0091

May 4, 2024

Abstract

This project focuses on developing a highly accurate model for recognizing handwritten digits using the well-known MINIST dataset. The dataset comprises 2,400 unique images for each digit (0 to 9) and serves as a fundamental resource in the realms of computer vision and machine learning. The proposed solution introduces a Convolutional Neural Network (CNN) model, known for its success in image recognition by mimicking human visual system functionality to identify patterns and features.

To optimize performance, the project adopts a strategy of fine-tuning the CNN's architecture. This involves adjusting the parameters of convolutional layers, pooling layers, and fully connected layers to enhance the model's ability to extract features and improve recognition results. Additionally, the project employs data augmentation techniques during training, incorporating rotations and translations to diversify the dataset and improve the model's generalization on new, unseen data. This not only increases sample diversity but also mitigates overfitting.

Key metrics such as accuracy and loss functions play a crucial role during training. Accuracy reflects the model's proficiency in recognizing digits, while the loss function measures the disparity between predicted and actual values. Continuous optimization of these metrics throughout training leads to gradual improvements in the model's performance.

Contents

1 Introduction

MINIST is a widely recognized and utilized classic dataset specifically designed for research and applications in handwritten digit recognition. In this dataset, there are 2,400 unique images for each digit (0 to 9). MINIST is commonly used as an introductory dataset in the fields of computer vision and machine learning.

The core objective of this project is to develop a model capable of accurately recognizing digits within images. To achieve this goal, the project introduces a structurally simple yet efficient Convolutional Neural Network (CNN) model. Convolutional Neural Networks are a highly successful learning architecture in the field of image recognition, as they mimic the functioning of the human visual system to identify patterns and features within images.

To optimize the final performance, the project considers the first strategy of optimizing the CNN's architecture. CNNs typically consist of convolutional layers, pooling layers, and fully connected layers. Convolutional layers are responsible for extracting local features from the images, pooling layers reduce the spatial dimensions of features to decrease computation and prevent overfitting to some extent, and fully connected layers transform these features into the final classification results. By adjusting the number and parameters of these layers, better recognition results can be achieved.

Furthermore, to enhance the model's generalization ability, i.e., its performance on unseen new data, the project employs data augmentation techniques during training. By applying rotations and translations to the existing training images, the model can learn the representations of digits in different positions and orientations, thus improving its ability to recognize handwritten digits in new data. Data augmentation not only increases the diversity of samples but also helps mitigate overfitting, where the model performs well on the training data but poorly on new data.

During training, accuracy and loss functions are two key metrics. Accuracy reflects the model's ability to recognize digits, while the loss function measures the discrepancy between the model's predictions and the actual values. By continuously optimizing these two metrics, the model's performance can be gradually improved.

In summary, using the classic MINIST dataset, this project constructs a Convolutional Neural Network and employs strategies such as architectural adjustments and data augmentation to establish an efficient and reliable handwritten digit recognition system. Furthermore, when compared to the initial model, based on accuracy and loss functions, the current model has shown significant improvement after parameter tuning and structural simplification. [1]

2 Methodology

2.1 Dataset

MINIST is a dataset for handwritten digit recognition. This dataset comprises images of handwritten digits ranging from 0 to 9. To establish an effective model, these images are divided into two parts: 2000 images are used as the training set, utilized for model learning and parameter tuning, whereas the remaining 400 images constitute the test set, which is employed for evaluating the model's performance and accuracy.

Figure 1: Example of Dataset

[1]You can see the whole code from: https://github.com/IvanMao714/DS_FinalProject/

2.2 Structure of CNN

To accurately recognize these handwritten digits, this project utilizes a relatively simple and efficient CNN model. The architecture of this model features a clear hierarchical structure designed to extract crucial features from images and subsequently perform digit classification.

The model consists of the following key components:

- **Two Convolutional Layers:** Firstly, the model systematically analyzes input images through two convolutional layers to capture local features within the images. Convolution operations are capable of identifying edges, textures, shapes, and other information in the images. The first convolutional layer has 1 input channel, 16 output channels, a convolutional kernel size of 5x5, a stride of 1, and padding of 2. The second convolutional layer shares a similar structure with the first but has 16 input channels and 32 output channels.
- **ReLU Activation Function:** Following each convolutional layer, there is a ReLU activation function. Its purpose is to introduce non-linearity, enabling the model to learn more complex features and patterns. ReLU activation function activates only in the positive interval and outputs 0 for negative input values. ReLU is computationally efficient and aids in mitigating the vanishing gradient problem, which is why this model adopts it as the activation function.

$$f(x) = \max(0, x) \tag{1}$$

- **Two Max Pooling Layers:** Immediately after the ReLU activation function, max-pooling layers are applied to reduce the dimensions of feature maps while retaining critical information. Both max-pooling layers have a pooling kernel size of 2x2. This helps reduce computational burden and prevent overfitting.
- **Fully Connected Layer:** The final part of the model comprises a fully connected layer, which transforms the feature maps extracted from the convolutional layers into the ultimate classification results. The fully connected layer associates feature maps with each of the ten-digit categories through a weight matrix. The number of output features in this layer is 10.
- **Softmax Layer:** Lastly, there is a Softmax layer to map the model's output into a probability distribution, enabling the determination of the likelihood that the input image belongs to each digit category.

This simple yet effective CNN architecture 8 has been designed to accurately identify handwritten digits. It accomplishes this by progressively processing images, extracting low-level features, and gradually building abstract representations, ultimately achieving the goal of digit classification.

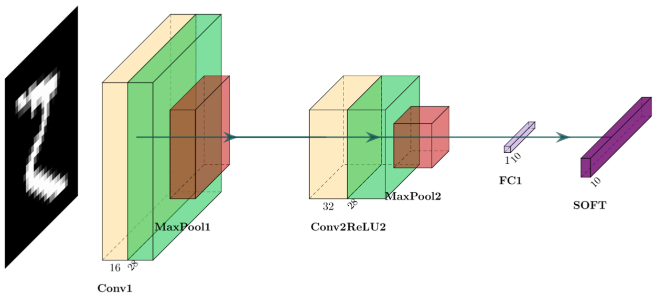


Figure 2: CNN Architecture.

2.3 Structure of DRNN

In order to get better performance, We try to design a more complex model(Digital Recognition Neutral Network) to solve this task. The model consists of the following key components:

- **Convolutional Layers:**
  - conv1: Convolutional layer with 1 input channel, 16 output channels, a kernel size of (3, 3), stride of (1, 1), and padding of (1, 1).
  - conv2: Convolutional layer with 16 input channels, 64 output channels, a kernel size of (3, 3), stride of (1, 1), and padding of (1, 1).
  - conv3: Convolutional layer with 64 input channels, 128 output channels, a kernel size of (3, 3), stride of (1, 1), and padding of (1, 1).
- **Batch Normalization Layers:** bn1, bn2, bn3: Batch normalization layers applied after each convolutional layer.
- **Max Pooling Layers:** pool: Max pooling layer with a kernel size of (2, 2), stride of 2, and padding of 1.
- **Dropout Layers**
  - dropout1: Dropout layer with a dropout probability of 0.75.
  - dropout2: Dropout layer with a dropout probability of 0.625.
- **Fully Connected Layer:**
  - fc1: Fully connected layer with 3200 input features and 64 output features.
  - fc2: Fully connected layer with 64 input features and 10 output features.
- **Softmax Layer:** Lastly, there is a Softmax layer to map the model's output into a probability distribution, enabling the determination of the likelihood that the input image belongs to each digit category.

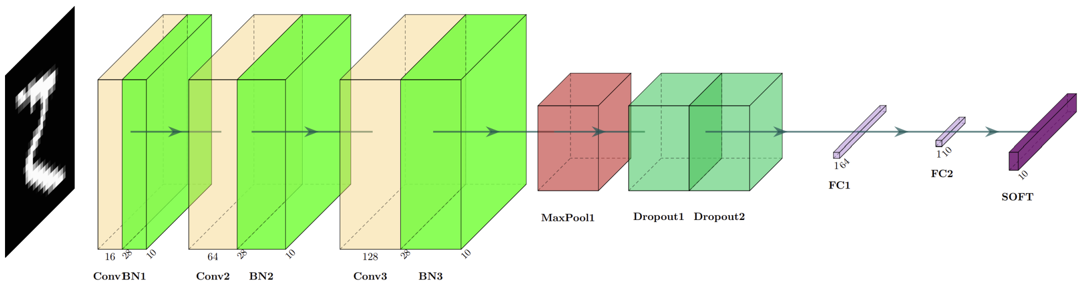
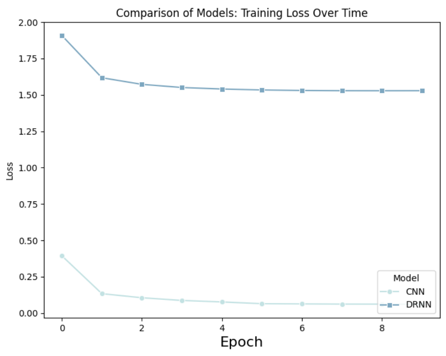


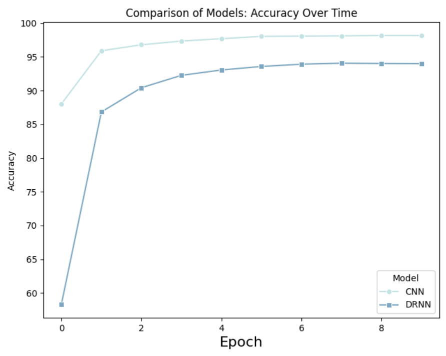
Figure 3: DRNN Architecture.

2.4 Loss Function

The loss function used in this model is *CrossEntropyLoss*. It is a key component in measuring the performance of the model. By calculating the difference between the probability distribution of the model's predicted outputs and the actual labels, it provides a quantitative indicator of the model's accuracy for the team. Specifically, it reveals the model's predictive ability for different categories, helping the team identify in which areas the model performs well and in which areas it needs improvement. For example, if the loss for certain categories is particularly high, this



(a) Training Loss Comparison



(b) Training Accuracy Comparison

Figure 4: Comparison between CNN and DRNN

may indicate a lack of understanding by the model in these specific areas, necessitating further adjustments or training. The core advantage of this type of loss function lies in its intuitiveness and effectiveness, especially when dealing with multi-class classification problems.

A key point is that *CrossEntropyLoss* does not just focus on the model's performance on training data; it also helps enhance the model's generalization ability on unseen data. By effectively minimizing the loss on training data, the model can learn broader and deeper patterns and associations, thus making accurate predictions when faced with new, unseen data. Therefore, *CrossEntropyLoss* is an essential tool, not only aiding the team in assessing and optimizing the model but also ensuring the model's reliability and effectiveness in practical applications.

2.5 Attempt

We design both structures with free training of both models to test which model is more suitable for this task. In our conception, the DRNN with a complex structure is able to capture more complex features and should theoretically perform better when dealing with complex image recognition tasks. The comparison results are shown in Figure 4 and the experimental results are not the same as what we expected. It is clear to find that the complex structure of DRNN does not provide better results for the task and the accuracy is lower than that of CNN. We speculate that perhaps for a dataset of the size of MINIST, the more complex the structure is, the more likely it is to be underfitted. Therefore, we have made CNN as our first choice, and we are no longer exploring the strategy of adding complex architectures to achieve higher accuracy. Also, we turned our attention to image transformation. Comparison is shown in Figure 4.

We found that the complex architecture of DRNN didn't bring a better effect on the task and the accuracy is less than CNN. Therefore, we think CNN is our first choice and we didn't explore adding complex architecture to acquire more accuracy anymore. We turned our attention to image transformation.

2.6 Image Transformation:

To further enhance accuracy, this project has employed multiple strategies to continuously optimize model performance. Not only have adjustments been made to the model's architecture and parameters, but image augmentation techniques have also been introduced during the training phase. These techniques include image rotation and translation operations. The purpose of this approach is to increase the model's robustness, enabling it to better adapt to input images at various angles, scales, and orientations, thus improving overall prediction accuracy. As shown in the bar chart below, the number of augmented images increases to between approximately 5,500 and 7,000.

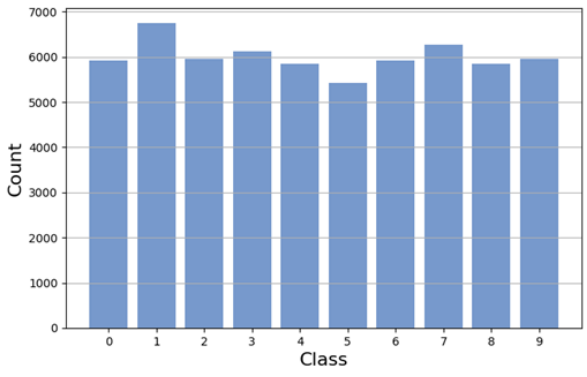


Figure 5: The Number of Augmented Images.

2.6.1 Data Augment:

This project defines a class whose primary function is to implement random rotation operations on an input image.

This class generates a parameter called ”degrees” and sets it to 20. Subsequently, based on this parameter, it generates a binary tuple containing two positive elements. This binary tuple effectively delineates a specified range. Then, a random angle is selected from this specified range, and it is used as the random rotation angle for the image, which is then returned.

This process allows for random rotation of input images, increasing data diversity and contributing to improved model robustness and generalization.

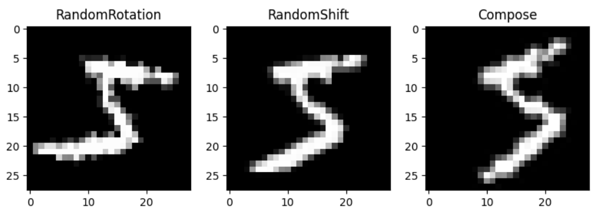


Figure 6: Example of Augment

```
class RandomRotation(object):

    def __init__(self, degrees, resample=False, expand=False, center=None):
        if isinstance(degrees, numbers.Number):
            if degrees < 0:
                raise ValueError("If degrees is a single number, it must be positive.")
            self.degrees = (-degrees, degrees)
        else:
            if len(degrees) != 2:
```

```python
                raise ValueError("If degrees is a sequence, it must be of len 2.")
            self.degrees = degrees

        self.resample = resample
        self.expand = expand
        self.center = center

    @staticmethod
    def get_params(degrees):
        angle = np.random.uniform(degrees[0], degrees[1])

        return angle

    def __call__(self, img):

        def rotate(img, angle, resample=False, expand=False, center=None):

            return img.rotate(angle, resample, expand, center)

        angle = self.get_params(self.degrees)

        return rotate(img, angle, self.resample, self.expand, self.center)

```

Algorithm 1: RandomRotation Class

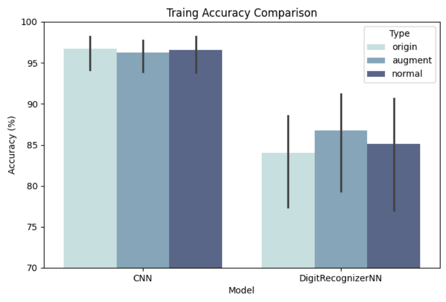
In this project, another class has been defined. The primary function of this class is to implement random image translation operations. The maximum range of translation is specified by the ”shift” parameter, which is set to 3. Horizontal and vertical translation values are randomly generated within this range. Finally, the class returns a matrix that is used to move the image horizontally and vertically based on this matrix. This simulates images from different perspectives, thereby enhancing the model’s robustness.

```python
class RandomShift(object):
    def __init__(self, shift):
        self.shift = shift

    @staticmethod
    def get_params(shift):
        hshift, vshift = np.random.uniform(-shift, shift, size=2)

        return hshift, vshift

    def __call__(self, img):
        hshift, vshift = self.get_params(self.shift)

        return img.transform(img.size, Image.AFFINE, (1, 0, hshift, 0, 1, vshift), resample=Image.BICUBIC, fill=1)

```

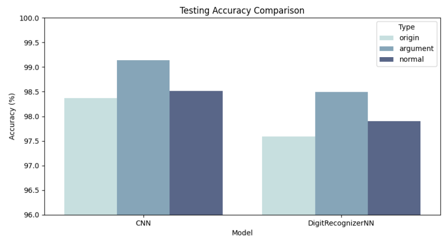
Algorithm 2: RandomRotation Class

2.6.2 Normalization

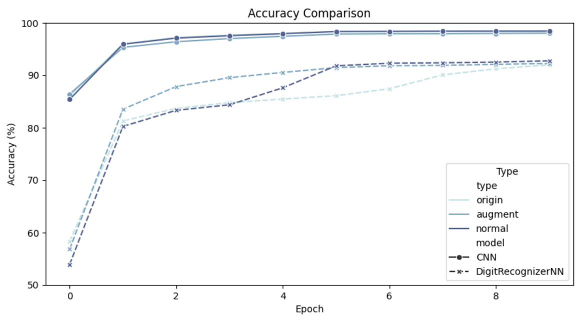
During training, this project also applies a normalization process to the images. To be more precise, both the mean and standard deviation are set to 0.5, ensuring that the results fall within the range of [-1, 1]. This normalization serves multiple purposes: firstly, it accelerates the convergence of the neural network, thus speeding up the training process. Secondly, it helps to maintain



(a) Training Accuracy Comparison



(b) Testing Accuracy Comparison



(c) Testing Accuracy Comparison

Figure 7: Accuracy Result

gradients within a smaller range, mitigating gradient-related issues to some extent. Most importantly, it enables the model to generalize better to unseen data, rather than merely memorizing the training data.

3 Performance

3.1 Accuracy

When evaluating the test set, it became clear that the initial model, the DRNN, was outperformed in overall effectiveness by the final adopted model, which is the CNN. This comparative result indicates that the adjustments to the model structure and parameters were effective and in the correct direction. Specifically, the model's recognition accuracy improved significantly from about 97.5% to nearly 98.5%. This improvement not only reflects the superiority of CNNs in handling such problems but also mirrors the correct decisions made during the model optimization process.

More importantly, after data augmentation on the training data, the recognition accuracy of the CNN model was further enhanced, exceeding 99%. This achievement proves that adjusting the training dataset, especially increasing data diversity through data augmentation strategies, is effective in enhancing the model's generalization capability and recognition precision. Data augmentation provided the model with a more abundant and diverse set of learning samples, enabling it to better learn and understand data features under different scenarios, thereby exhibiting higher accuracy and robustness in practical applications.

In summary, by comparing the performance of DRNN and CNN on the test set, we not only validated the effectiveness of model structure and parameter adjustments but also demonstrated the significant role of data augmentation in enhancing model performance. All of detail data shown in Figure 7.

3.2 Loss Value

As the number of training epochs increases, it becomes evident that the loss value is gradually decreasing. Moreover, the CNN model with adjusted architecture and parameters exhibits significantly lower loss values compared to the initial DRNN model. This implies that the final CNN demonstrates superior performance in recognizing handwritten digits, and its predictions are much closer to the actual answers.

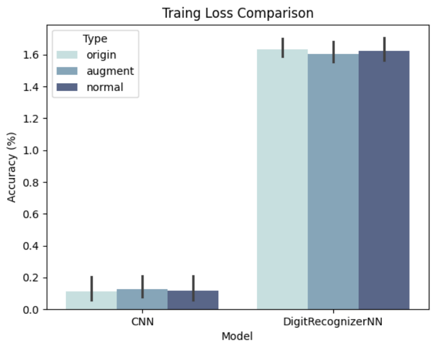


Figure 8: Training Loss Comparison

4 Contribution

a) Mao Yifan:

- Design DRNN and CNN models (90 lines of the code)
- Data Augment Class(60 lines of the code)
- Data Processing (80 lines of the code)
- Code of the training model (100 lines of the code)

b) Wang Yiye:

- Data loader (81 lines of the code)
- Parameter tuning
- Code of the testing model (35 lines of the code)
- Report written 30%

c) Liu Zhankeng:

- Generation of mode structure diagrams (60 lines of the code)
- Generation of charts for reporting (90 lines of the code)
- Report written 70%
- Presentation