INFOSYS SPRINGBOARD INTERNSHIP

Project:

Handwritten Digital Recognition Model

Members

Rudraksh Gupta

Sneha Kushwaha

Yash Nilesh Shah

Mentor

Narendra Kumar

Disclaimer - All the analysis done in this file is based on the programmed code published on the dedicated GitHub repository. For more details, please visit this website

Detailed Analysis Report on Handwritten Recognition Model

Table of Contents

Model 1: LeNet-5

1. Introduction

The LeNet-5 model, designed by Yann LeCun in 1998, is one of the pioneering convolutional neural network (CNN) architectures. It was primarily developed for handwritten digit recognition and has shown robust performance on the MNIST dataset. This report details the architecture, training process, and performance of the LeNet-5 model.

2. Dataset Overview

The MNIST dataset comprises 70,000 grayscale images of handwritten digits (0-9), each of size 28x28 pixels. It is divided into:

- **Training Set**: 60,000 images
- **Test Set**: 10,000 images

The dataset is a standard benchmark for evaluating image classification models.

3. Data Preprocessing

Preprocessing steps include:

- **Normalization**: Scaling pixel values to the range [0, 1].

```
x_train_full = x_train_full / 255.0
x_test = x_test / 255.0
```

- **Splitting**: Dividing the dataset into training, validation, and test sets.

```
x_valid, x_train = x_train_full[:5000], x_train_full[5000:]
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
```

- **Reshaping**: Adding a channel dimension.

```
x_train = x_train[..., np.newaxis]
x_valid = x_valid[..., np.newaxis]
x_test = x_test[..., np.newaxis]
```

- **One-Hot Encoding**: Converting labels to one-hot encoded vectors.

```
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_valid = tf.keras.utils.to_categorical(y_valid, 10)
```

```
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

4. Model Architecture

The LeNet-5 architecture:

- **Input Layer**: (28, 28, 1)
- **Convolutional Layer 1**: 6 filters, kernel size (5, 5), activation: relu

```
model = models.Sequential([
    layers.Conv2D(6, (5, 5), activation='relu', input_shape=(28, 28, 1)),
])
```

- **Average Pooling Layer 1**

```
layers.AveragePooling2D(),
```

- **Convolutional Layer 2**: 16 filters, kernel size (5, 5), activation: relu

```
layers.Conv2D(16, (5, 5), activation='relu'),
```

- **Average Pooling Layer 2**

```
layers.AveragePooling2D(),
```

- **Flatten Layer**

```
layers.Flatten(),
```

- **Dense Layer 1**: 120 units, activation: relu

```
layers.Dense(120, activation='relu'),
```

- **Dense Layer 2**: 84 units, activation: relu

```
layers.Dense(84, activation='relu'),
```

- **Output Layer**: 10 units, activation: softmax

```
layers.Dense(10, activation='softmax')
```

5. Training Process

Training details:

- **Optimizer**: Adam

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

- **Loss Function**: categorical_crossentropy
- **Metrics**: Accuracy
- **Epochs**: 10

```
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_valid, y_valid))
```

- **Batch Size**: 128

6. Evaluation Metrics

Performance metrics:

- **Validation Accuracy**: Monitored during training
- **Test Accuracy**: Final accuracy on the test set

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('\nTest accuracy:', test_acc)
```

7. Inference Example

An inference example shows the model's capability:

- Predicting a test image:

```
plt.imshow(x_test[0].reshape(28, 28), cmap="gray")
plt.show()

result = model.predict([x_test[0].reshape(1, 28, 28, 1)])
print(np.argmax(result))
```

8. Conclusion and Recommendations

The LeNet-5 model demonstrates robust performance for handwritten digit recognition. Future work could explore deeper architectures, hyperparameter tuning, and data augmentation techniques for improved performance.

Model 2: Logistic Regression

1. Introduction

Logistic regression is a fundamental machine learning algorithm used for binary and multi-class classification. This report analyzes the implementation and performance of logistic regression on the MNIST dataset.

2. Dataset Overview

The MNIST dataset includes 70,000 grayscale images of handwritten digits (0-9), each of size 28x28 pixels, divided into:

- **Training Set**: 60,000 images
- **Test Set**: 10,000 images

3. Data Preprocessing

Preprocessing steps:

- **Normalization**: Scaling pixel values to [0, 1].

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

- **Flattening**: Converting 2D images to 1D vectors.

```
x_train = x_train.reshape(x_train.shape[0], -1)
x_test = x_test.reshape(x_test.shape[0], -1)
```

4. Model Architecture

Logistic regression model:

- **Input Layer**: 784 units (28x28)
- **Output Layer**: 10 units (softmax activation for multi-class classification)

```
model = Sequential([
    Dense(10, input_shape=(784,), activation='softmax')
])
```

5. Training Process

Training details:

- **Optimizer**: SGD (Stochastic Gradient Descent)

```
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

- **Loss Function**: sparse_categorical_crossentropy
- **Metrics**: Accuracy
- **Epochs**: 10

```
model.fit(x_train, y_train, epochs=10, batch_size=32)
```

6. Evaluation Metrics

Performance metrics:

- **Test Accuracy**: Evaluated on the test set

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('\nTest accuracy:', test_acc)
```

7. Inference Example

Inference example:

- Predicting a test image:

```
plt.imshow(x_test[0].reshape(28, 28), cmap="gray")
plt.show()

result = model.predict([x_test[0].reshape(1, 784)])
print(np.argmax(result))
```

8. Conclusion and Recommendations

Logistic regression provides a simple yet effective approach for digit recognition, achieving decent accuracy. Future improvements could involve exploring more sophisticated models, regularization techniques, and feature engineering.

Model 3: Multi-Layer Perceptron (MLP)

1. Introduction

A Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural networks, consisting of multiple layers of neurons. This report details the architecture, training, and performance of an MLP model for handwritten digit recognition.

2. Dataset Overview

The MNIST dataset comprises 70,000 grayscale images of handwritten digits (0-9), each of size 28x28 pixels, divided into:

- **Training Set**: 60,000 images
- **Test Set**: 10,000 images

3. Data Preprocessing

Preprocessing steps:

- **Normalization**: Scaling pixel values to [0, 1].

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

- **Flattening**: Converting 2D images to 1D vectors.

```
x_train = x_train.reshape(x_train.shape[0], -1)
x_test = x_test.reshape(x_test.shape[0], -1)
```

4. Model Architecture

MLP architecture:

- **Input Layer**: 784 units (28x28)
- **Hidden Layer 1**: 128 units, activation: relu

```
model = Sequential([
   Dense(128, input_shape=(784,), activation='relu'),
])
```

- **Hidden Layer 2**: 64 units, activation: relu

```
python
Copy code
model.add(Dense(64, activation='relu'))
```

- **Output Layer**: 10 units, activation: softmax

```
python
Copy code
model.add(Dense(10, activation='softmax'))
```

5. Training Process

Training details:

- **Optimizer**: Adam

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

- **Loss Function**: sparse_categorical_crossentropy
- **Metrics**: Accuracy
- **Epochs**: 10

```
model.fit(x_train, y_train, epochs=10, batch_size=32)
```

6. Evaluation Metrics

Performance metrics:

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('\nTest accuracy:', test_acc)
```

7. Inference Example

Inference example:

- Predicting a test image:

```
plt.imshow(x_test[0].reshape(28, 28), cmap="gray")
plt.show()

result = model.predict([x_test[0].reshape(1, 784)])
print(np.argmax(result))
```

8. Conclusion and Recommendations

The MLP model offers a balance between simplicity and performance, providing accurate digit classification. Further enhancements could include experimenting with deeper networks, dropout for regularization, and hyperparameter tuning.

Comparison between the models

ASPECT	LeNet-5	Logistic Regression	Multi-Layer Perceptron (MLP)
ARCHITECTURE	Convolutional Neural Network (CNN)	Single-layer linear classifier	Feedforward Neural Network with hidden layers
NUMBER OF LAYERS	7 layers (including pooling and dense layers)	1 layer (output layer)	3 layers (2 hidden layers + output layer)
INPUT SHAPE	(28, 28, 1)	(784,)	(784,)
HIDDEN LAYERS	Convolutional and pooling layers	None	2 hidden layers (128 and 64 units)
ACTIVATION FUNCTIONS	ReLU (hidden layers), Softmax (output layer)	Softmax (output layer)	ReLU (hidden layers), Softmax (output layer)
OUTPUT SHAPE	10 units (one for each digit)	10 units (one for each digit)	10 units (one for each digit)
OPTIMIZER	Adam	SGD (Stochastic Gradient Descent)	Adam
LOSS FUNCTION	Categorical Crossentropy	Sparse Categorical Crossentropy	Sparse Categorical Crossentropy
TRAINING EPOCHS	30	30	30
BATCH SIZE	128	32	32
NORMALIZATION	Yes	Yes	Yes

FLATTENING	No (uses convolutional layers)	Yes (flattened to 1D vector)	Yes (flattened to 1D vector)
DATA PREPROCESSING	Normalization, reshaping, one-hot encoding	Normalization, flattening	Normalization, flattening
VALIDATION ACCURACY	Monitored during training	Not explicitly mentioned	Monitored during training
TEST ACCURACY	Evaluated on test set	Evaluated on test set	Evaluated on test set
INFERENCE EXAMPLE	Predicts using model.predict method	Predicts using model.predict method	Predicts using model.predict method
SUITABILITY	Best for image data, especially where spatial hierarchies exist	Simple and effective for baseline performance on structured data	Good balance between simplicity and performance, suitable for structured data with complex patterns

Summary of the Three Models

LeNet-5

LeNet-5 is a convolutional neural network (CNN) designed specifically for handwritten digit recognition. Developed by Yann LeCun, it is notable for its pioneering role in the development of deep learning. The architecture of LeNet-5 includes seven layers, comprising convolutional layers for feature extraction, average pooling layers for dimensionality reduction, and fully connected layers for classification. The network utilizes ReLU activation functions in the hidden layers and a softmax activation function in the output layer, making it well-suited for image data where capturing spatial hierarchies is crucial. LeNet-5 is optimized using the Adam optimizer and employs the categorical crossentropy loss function. It demonstrates high accuracy on the MNIST dataset, leveraging its ability to capture intricate patterns in image data through its convolutional layers.

Logistic Regression

Logistic regression is a simple yet effective machine learning algorithm used for classification tasks. Unlike neural networks, it consists of a single-layer linear classifier with no hidden layers. The model takes a flattened 1D vector as input, corresponding to the pixel values of the images, and outputs probabilities for each class using the softmax activation function. Due to its simplicity, logistic regression is computationally efficient and easy to interpret. It is optimized using Stochastic Gradient Descent (SGD) and utilizes the sparse categorical crossentropy loss function. While logistic regression provides a decent baseline performance on structured data, it may not capture complex patterns as effectively as neural networks. It is particularly suitable for scenarios where interpretability and computational efficiency are prioritized over capturing intricate data patterns.

Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron (MLP) is a type of feedforward artificial neural network that consists of multiple layers of neurons, including hidden layers. In this specific implementation, the MLP model comprises an input layer that takes a flattened 1D vector, two hidden layers with 128 and 64 units respectively, and an output layer with 10 units, each

representing a digit class. The hidden layers use ReLU activation functions to introduce non-linearity, while the output layer employs a softmax activation function for classification. The MLP model is optimized using the Adam optimizer and the sparse categorical crossentropy loss function. It strikes a balance between complexity and performance, making it suitable for structured data with more intricate patterns. The MLP model achieves good accuracy on the MNIST dataset, providing a robust alternative to simpler models like logistic regression, while being less computationally intensive than more complex CNNs like LeNet-5.

Comparative Summary

LeNet-5, logistic regression, and MLP each have distinct characteristics that make them suitable for different types of tasks. LeNet-5 excels in image classification tasks that require capturing spatial hierarchies, thanks to its convolutional layers. Logistic regression, with its simplicity and interpretability, serves as an efficient baseline for structured data but is limited in capturing complex patterns. The MLP model offers a middle ground, capable of modelling more complex patterns than logistic regression, while being computationally less demanding than LeNet-5. Each model's architecture, training process, and performance metrics highlight their respective strengths and suitability for different scenarios, providing a comprehensive toolkit for tackling a variety of machine learning tasks.