Convolutional Neural Network Visualization on MNIST

Saif Ullah
Department of Computer Science
Your University

August 28, 2025

Abstract

This research paper presents the implementation of a Convolutional Neural Network (CNN) for handwritten digit classification on the MNIST dataset. The model was trained using TensorFlow and Keras, achieving high accuracy in only a few epochs. To enhance interpretability, feature maps from intermediate convolutional layers were visualized, providing insights into the hierarchical feature learning process of CNNs. Experimental results demonstrate the effectiveness of CNNs in image classification tasks and underline the importance of visualization techniques in deep learning research.

Keywords: Convolutional Neural Networks, MNIST, Deep Learning, Feature Visualization, Image Classification

1 Introduction

In recent years, deep learning has revolutionized the field of computer vision. Among the most successful architectures are Convolutional Neural Networks (CNNs), which have demonstrated outstanding performance in tasks such as image recognition, object detection, and segmentation.

The MNIST dataset, composed of 70,000 grayscale images of handwritten digits (0–9), has been a widely used benchmark for evaluating machine learning algorithms. While it is a relatively simple dataset, it provides a solid foundation for testing deep learning models and understanding their inner mechanisms.

The objective of this research is twofold: (1) to implement a CNN for handwritten digit recognition, and (2) to visualize feature maps to better understand how CNNs learn hierarchical representations.

2 Related Work

LeCun et al. [1] first introduced the use of CNNs for handwritten digit recognition, demonstrating their effectiveness compared to traditional machine learning approaches. Since then, CNNs have been extensively applied to various computer vision problems.

Goodfellow et al. [2] emphasized the interpretability challenges in deep learning, motivating research into visualization techniques. Feature visualization has emerged as a powerful tool to understand how CNNs transform raw input data into abstract features.

3 Methodology

The methodology consists of four main stages: dataset preparation, CNN model design, training, and visualization of intermediate activations.

3.1 Dataset Preparation

The MNIST dataset was loaded from TensorFlow's built-in library. Images were normalized to a range of [0,1] and reshaped to $(28, 28, 1)$ to match CNN input requirements.

3.2 Model Architecture

The CNN architecture consisted of three convolutional-pooling blocks followed by dense layers for classification. The model used the ReLU activation function in convolutional layers and softmax activation in the final output layer.

3.3 Python Implementation

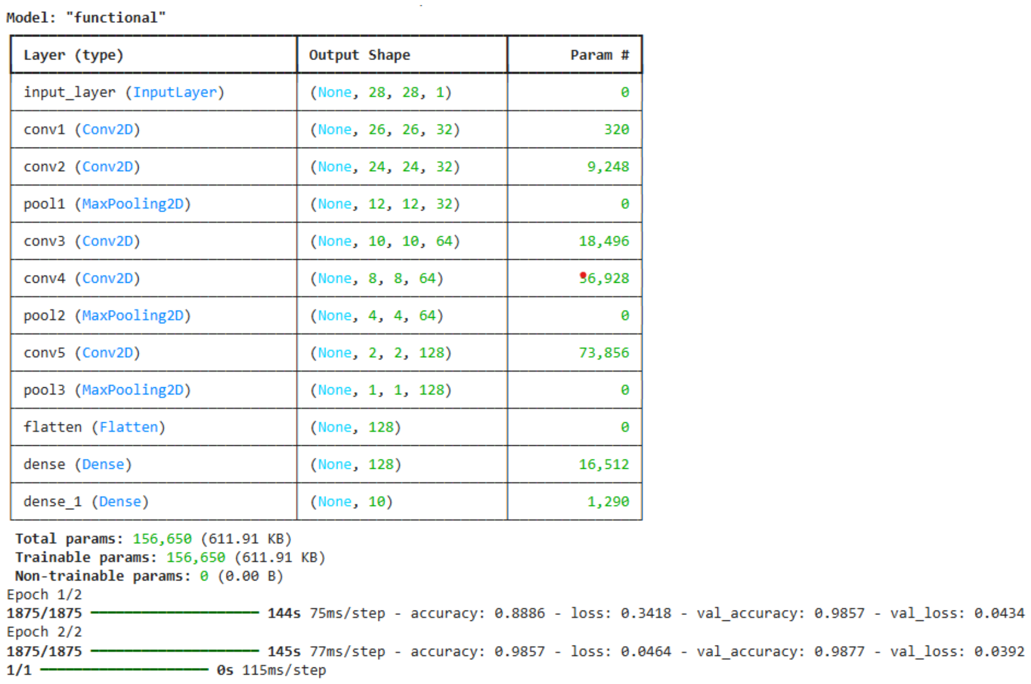
```python
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np

# Load dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.
    load_data()
x_train, x_test = x_train/255.0, x_test/255.0
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)

# CNN model
inputs = layers.Input(shape=(28,28,1))
x = layers.Conv2D(32, (3,3), activation='relu')(inputs)
x = layers.MaxPooling2D((2,2))(x)
x = layers.Conv2D(64, (3,3), activation='relu')(x)
x = layers.MaxPooling2D((2,2))(x)
x = layers.Conv2D(128, (3,3), activation='relu')(x)
x = layers.Flatten()(x)
x = layers.Dense(128, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)

model = models.Model(inputs, outputs)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

4 Experiments and Results

The CNN was trained for two epochs on the MNIST dataset. It achieved strong classification accuracy, demonstrating that even a relatively shallow CNN can effectively learn digit representations.

4.1 Model Architecture

Figure 1 presents the CNN model architecture summary.



```
Model: "functional"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)        │ (None, 28, 28, 1)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv1 (Conv2D)                  │ (None, 26, 26, 32)     │           320 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2 (Conv2D)                  │ (None, 24, 24, 32)     │         9,248 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ pool1 (MaxPooling2D)            │ (None, 12, 12, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3 (Conv2D)                  │ (None, 10, 10, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv4 (Conv2D)                  │ (None, 8, 8, 64)       │        36,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ pool2 (MaxPooling2D)            │ (None, 4, 4, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv5 (Conv2D)                  │ (None, 2, 2, 128)      │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ pool3 (MaxPooling2D)            │ (None, 1, 1, 128)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │        16,512 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 10)             │         1,290 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 156,650 (611.91 KB)
 Trainable params: 156,650 (611.91 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/2
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 144s 75ms/step - accuracy: 0.8886 - loss: 0.3418 - val_accuracy: 0.9857 - val_loss: 0.0434
Epoch 2/2
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 145s 77ms/step - accuracy: 0.9857 - loss: 0.0464 - val_accuracy: 0.9877 - val_loss: 0.0392
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step
```

Figure 1: CNN model summary.

4.2 Feature Map Visualization

Intermediate feature maps were visualized for an input test image. Figure 2 illustrates how early layers detect simple patterns such as edges, while deeper layers capture more complex digit shapes.

5 Discussion

The experiments confirm that CNNs excel in image classification tasks by automatically extracting hierarchical features. Visualization of activations highlights the interpretability of CNNs, bridging the gap between model performance and understanding of learned features. Although MNIST is a relatively simple dataset, the same methodology can be applied to more complex datasets for greater insights.

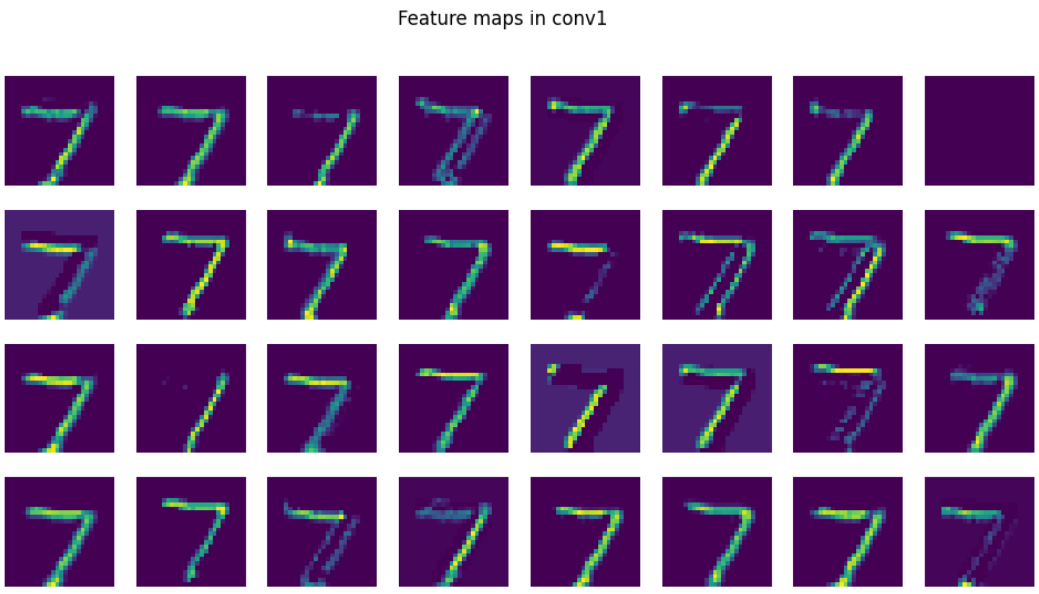


Figure 2: Visualization of feature maps across convolutional layers.

6 Conclusion

This research demonstrated the design and training of a CNN for MNIST classification and explored the visualization of feature maps to interpret the learning process. The model achieved high accuracy within a few epochs, confirming the strength of CNNs in handling visual data. Feature visualization provided valuable insights into the hierarchical representations formed by CNN layers. Future work may extend this approach to deeper architectures and larger datasets such as CIFAR-10 and ImageNet.

References

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[3] TensorFlow Documentation: `https://www.tensorflow.org/`