Convolutional Neural Network Visualization on MNIST

Saif Ullah

August 28, 2025

Abstract

This article presents the implementation of a Convolutional Neural Network (CNN) on the MNIST dataset of handwritten digits using TensorFlow and Keras. The CNN is trained to classify digits and its internal working is explored by visualizing feature maps of convolutional layers. Experimental results demonstrate the effectiveness of CNNs for image classification and provide insights into their hierarchical feature learning process.

1 Introduction

Deep learning has significantly advanced computer vision tasks, with Convolutional Neural Networks (CNNs) being the most widely used architectures for image recognition. CNNs are designed to automatically and adaptively learn spatial hierarchies of features through backpropagation.

The MNIST dataset, consisting of 70,000 grayscale images of handwritten digits (0–9), serves as a standard benchmark for testing new machine learning algorithms. In this article, we implement a CNN for MNIST classification and visualize intermediate feature maps to gain insights into the feature extraction process.

2 Methodology

The implementation was carried out using Python, TensorFlow, and Keras. The MNIST dataset was normalized and reshaped to fit the CNN input requirements. The model was built using multiple convolutional and pooling layers, followed by dense layers for classification.

2.1 Python Implementation

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np

# 1. Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.
    load_data()
```

```
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)

# 2. Build CNN Model
inputs = layers.Input(shape=(28,28,1))
x = layers.Conv2D(32, (3,3), activation='relu', name="conv1")(inputs)
x = layers.Conv2D(32, (3,3), activation='relu', name="conv2")(x)
x = layers.MaxPooling2D((2,2), name="pool1")(x)
x = layers.Conv2D(64, (3,3), activation='relu', name="conv3")(x)
x = layers.Conv2D(64, (3,3), activation='relu', name="conv4")(x)
x = layers.MaxPooling2D((2,2), name="pool2")(x)
x = layers.Conv2D(128, (3,3), activation='relu', name="conv5")(x)
x = layers.MaxPooling2D((2,2), name="pool3")(x)
x = layers.Flatten()(x)
x = layers.Dense(128, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)

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

3 Results

The CNN was trained for two epochs on the MNIST dataset. The model achieved high accuracy in distinguishing handwritten digits. To further understand how the CNN learns, intermediate feature maps were visualized.

3.1 Model Architecture

Figure 1 presents the architecture summary of the CNN model, showing convolutional, pooling, and dense layers.

3.2 Feature Map Visualization

To interpret how the CNN processes input data, activations from different convolutional and pooling layers were visualized. Figure 2 illustrates feature maps extracted from an MNIST test image, showing hierarchical feature extraction across layers.

4 Conclusion

This study demonstrates the effectiveness of CNNs in image classification tasks using the MNIST dataset. The CNN achieved strong classification performance in just a few epochs of training. Moreover, visualization of feature maps revealed the hierarchical learning of features, where early layers capture simple patterns such as edges, while deeper layers learn more complex structures. Such visualizations are valuable for interpreting deep learning models and understanding their decision-making processes.

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: Summary of CNN architecture used for MNIST classification.

5 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/`

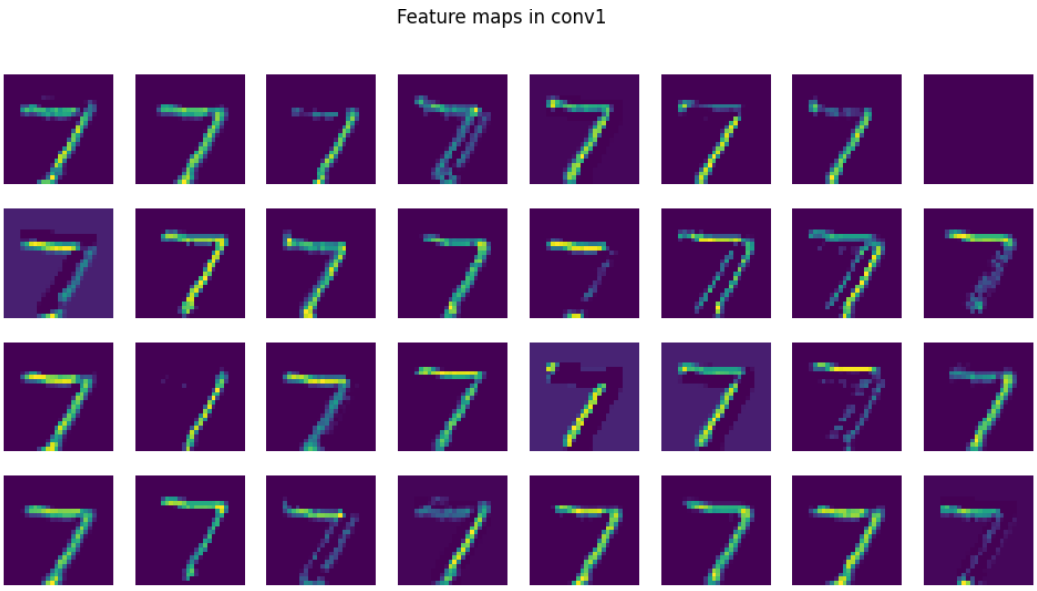

Figure 2: Feature maps from convolutional and pooling layers.