Understanding CNN Feature Extraction: Visualising Filters and Feature Maps on MNIST

1. Introduction

Convolutional Neural Networks (CNNs) are a foundational tool in modern machine learning for image recognition tasks. Unlike traditional fully connected networks, CNNs leverage the spatial structure of images by applying convolutional filters to extract features at multiple levels. These hierarchical features allow CNNs to achieve high accuracy in visual tasks with fewer parameters than fully connected networks.

This tutorial focuses on visualising how CNNs extract features, using the MNIST dataset of handwritten digits. By inspecting learned filters and feature maps, we aim to understand the inner workings of CNNs, providing both educational insight and practical guidance for using CNNs in image-based tasks.

2. The MNIST Dataset

MNIST consists of 70,000 grayscale images of handwritten digits (0-9) sized 28x28 pixels. It is split into 60,000 training images and 10,000 test images. MNIST is a standard dataset for teaching CNNs because its simplicity allows for fast training and clear visualisation of features.

In this tutorial, we normalised the images using the dataset's mean (0.1307) and standard deviation (0.3081) to improve model convergence.

3. CNN Architecture

The CNN implemented in PyTorch consists of:

- Conv1: 1 input channel -> 8 output channels, 3x3 kernels, ReLU activation
- Conv2: 8 -> 16 channels, 3x3 kernels, ReLU activation
- Max Pooling: Reduces spatial dimensions by a factor of 2
- Fully Connected Layers: Flattened output -> 128 neurons -> 10 output classes

The architecture balances simplicity and interpretability. The first convolutional layer is particularly important for visualising basic feature extraction (edges and simple patterns), while deeper layers combine these into more complex shapes.

4. Training the Model

We trained the CNN using the Adam optimizer with a learning rate of 0.001 for 3 epochs. Cross-entropy loss was used as the objective function. Training was performed on CPU; the final model achieved 98.42% test accuracy, demonstrating that even a small CNN can effectively capture the structure of MNIST digits.

Training Curves:

- Loss Curve: Shows smooth convergence to a low training and validation loss.
- Accuracy Curve: Training and validation accuracy increase rapidly, stabilising around 98%.

[Insert images/training_loss.png and images/training_accuracy.png here]

5. Visualising Learned Filters

Filters in the first convolutional layer can be interpreted as edge and pattern detectors. Visualising

the 8 filters reveals the types of low-level features the network learns automatically during training.
[Insert images/conv1_filters.png here]

6. Feature Maps

Feature maps are the outputs of convolutional layers after applying filters and activation functions (ReLU). They represent how the network responds to different patterns in the input.
- Conv1 Feature Maps: Highlight edges and basic shapes of the input digit.
- Conv2 Feature Maps: Combine these simple features into more complex patterns, showing corners, intersections, and parts of the digit.

[Insert images/featuremaps_conv1.png and images/featuremaps_conv2.png here]

7. Combined Visualization

A side-by-side comparison of the original digit and several Conv1 feature maps illustrates the immediate effect of convolution:
- The network highlights different aspects of the input, focusing on distinct edges or textures in each feature map.

[Insert images/orig_and_conv1_grid.png here]

8. Implementation Notes

- Hooks in PyTorch were used to capture activations for visualisation.
- Normalisation improves training stability.
- The architecture and number of filters can be scaled for more complex datasets.
- Reproducibility ensured by setting random seeds.

9. Ethical and Educational Considerations

- Understanding CNN internals improves trust and interpretability, critical in real-world applications such as medical imaging or autonomous vehicles.
- Visualisation helps detect biases or unexpected behaviours in models.
- Using small datasets and models reduces energy consumption, aligning with ethical AI principles.

10. Conclusion

This tutorial demonstrated how CNNs extract hierarchical features from images. By visualising learned filters and feature maps, we gained insight into how CNNs process raw pixel data into meaningful representations. Such understanding is crucial for both practical implementation and teaching others about deep learning.

Key takeaways:
1. First-layer filters capture low-level patterns (edges, blobs).
2. Deeper layers combine features into more abstract representations.
3. Visualisation enhances model interpretability and understanding.

11. References (Harvard Style)

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016) Deep Learning. MIT Press.
3. Chollet, F. (2017) Deep Learning with Python. Manning Publications.

4. PyTorch Documentation (2025) torch.nn. Available at: https://pytorch.org/docs/stable/nn.html
5. MNIST Database (2025), Yann LeCun. Available at: http://yann.lecun.com/exdb/mnist/