Replication of Visualizing CNN

Tongle Yao
Northeastern University
yao.to@northeastern.edu

December 3, 2024

Abstract

Understanding the internal mechanisms of convolutional neural networks (CNNs) has often been likened to probing a "black box." Inspired by Zeiler and Fergus's work, which introduced a deconvolutional network-based method to visualize feature maps [1] , we reproduced their results focusing on the hierarchical representations of CNNs.

1 Introduction

Motivated by a curiosity to bridge the gap between abstract mathematical representations and interpretable visualizations, we focused on reproducing the feature visualization experiments presented in Zeiler and Fergus's paper[1]. To achieve this, we implemented the deconvolution process, carefully replicating the model architecture, training parameters, and dataset (ImageNet).

Our reproduction process involved: (1) re-implementing the model architecture and training it on the ImageNet dataset, (2) visualizing feature maps across different layers using deconvolutional techniques, and (3) comparing our results to the original visualizations.

2 Model Training

To replicate the CNNs architecture described in Zeiler's paper[1], we referred to the architecture diagram provided in the original paper (Figure 1(b) 1). However, upon calculation, we found that the original paper does not explicitly specify the padding strategy for certain layers, which leads to misalignment in the output dimensions of the convolution operations.

The original text mentions initializing weights with a value of 10^{-2} and setting the initial learning rate to 10^{-2}. However, during training, this approach caused gradient explosion and vanishing gradient issues in the initial epochs. To address this problem: We switched to the Xavier Normal initialization method for weights and adjusted the initial learning rate to 0.0001.

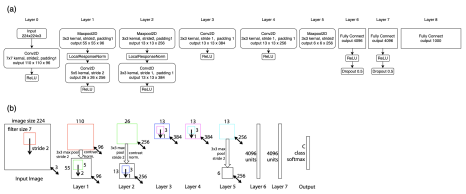

Figure 1: (a) Our stucture in replication (b) structure given in original paper[1].

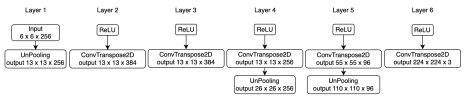

Figure 2: Structure of Deconvolutional Network

Regarding data preprocessing, the authors mentioned in the paper "using 10 different sub-crops of size 224x224"[1] but did not specify whether these images were preprocessed beforehand or if the transformations during training. After reviewing Krizhevsky's[2] training methodology, which was referenced in the zeiler's paper, I found that they only used TenCrop during testing, while Random Crop was applied to the input data during training. Therefore, I adopted the Random Crop method for my training.

3 Visualization

3.1 Structure of DeConv Network

We implemented an independent deconvolutional network to progressively reconstruct the feature maps from each layer of the CNN.(Figure 2)

The structure of the deconvolutional neural network is the reverse of the original CNN. Each convolutional layer is replaced by a deconvolutional layer, each MaxPooling layer is replaced by an Unpooling layer, and the ReLU activation function remains unchanged. The details are as follows:

Deconvolutional Layer Implemented by transposing the convolutional kernel weights, it is used to reconstruct the input features. We directly utilized the ConvTranspose2d method from the PyTorch framework. It is important to note that the weights, stride, kernel size, and padding of the convolutional layer must be copied to the corresponding deconvolutional layer.

Unpooling Layer The unpooling operation reverses the MaxPooling process by remapping the pooled features to their correct positions using the recorded indices of the maximum values (Switch).We added a Hook function to each pooling layer of the original CNN to record the Switch information. This

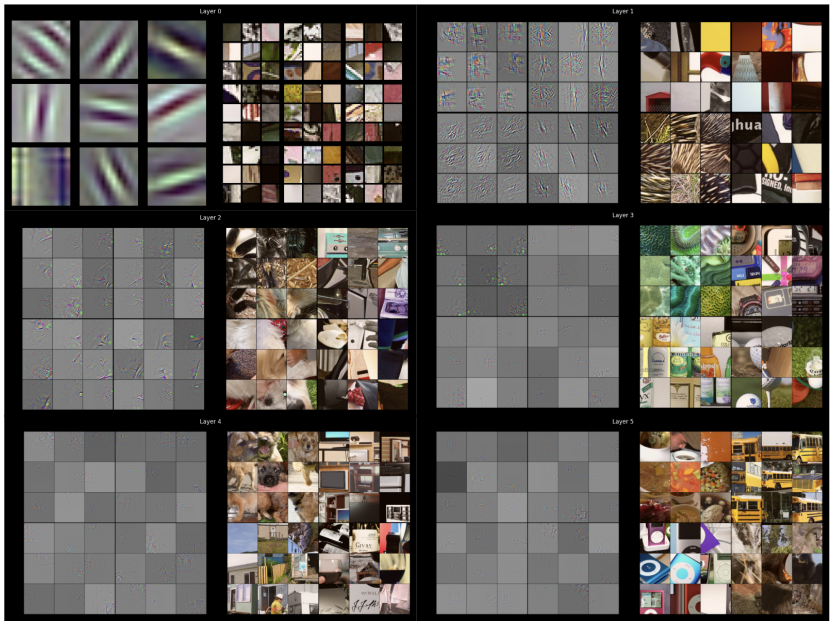

Figure 3: For Layer 0, the left side shows the convolutional kernels corresponding to the highest activation values, while for the remaining layers, the left side displays the deconvolution results, and the right side shows the original images.

information is then passed to the Unpooling layer of the deconvolutional network for use during the forward pass.

3.2 Process of Deconvolute

First, we obtain the feature maps of a specified layer through forward propagation. We chose the channel has highest activation. For each unselected channel, we set all its activation values to zero to ensure that only the target channels' information is retained.

Next, for the selected channels, we calculate the maximum activation value of each channel for every image in the batch and select the top nine images with the highest activation values. For these selected images, we further set all activation values other than the maximum to zero within the respective channel. This step is crucial as it directly determines the clarity of the final visualization while the original method did not explicitly mention this process.

Subsequently, we use a deconvolutional neural network to iteratively restore the feature maps to the input space. The final output is a three-channel colored feature activation map that intuitively displays the network's feature extraction and analysis at different layers.

3.3 Visualization and Analysis of Results

By using a deconvolutional network to visualize the outputs of different layers in a CNN, we can intuitively observe the feature extraction process from lower to higher layers. (Figure 3)

Low-Level Features (Layer 0)

For Layer 0, we directly visualized the convolutional kernels. The visualization of convolutional kernels revealed that Layer 0 primarily focuses on edges, color variations, and contrast, consistent with the findings in the original paper. These features serve as the foundation for the CNN to construct higher-level abstract features.

Mid-Level Features (Layers 1 to 3)

As the depth increases, the feature maps show attention to more complex patterns. From Layer 1, we can observe that the convolutional kernels combine features from Layer 0 to form corners, grid structures, and connections between edges and colors. The features in Layers 2 and 3 become increasingly complex, representing combinations of textures and local geometric patterns.

High-Level Features (Layers 4 to 5)

In Layers 4 and 5, the spatial scope of the feature maps expands significantly, indicating that the high-level features respond to larger regions of the image. This suggests that the network progressively shifts its focus from local details to the overall objects. We observed that these high-level features exhibit significant invariance, demonstrating stability against transformations such as rotation, scaling, and translation of the input images. For instance, in Layer 5, the bottom-left 9 images all focus on the control wheel of iPods, aligning perfectly with the characteristics described in the original paper.

3.4 Open Source Release

The related code and images from the paper will be publicly available later at `https://github.com/TL-Yao/Visualizating-CNN-Replication`

References

[1] MD Zeiler. Visualizing and understanding convolutional networks. In *European conference on computer vision/arXiv*, volume 1311, 2014.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.