Visualization of Kernel Function of Convolutional Neural Network

Chengfan Li¹,², Zirong Hu¹ and Xuehai Ding¹,*

¹School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
²Shanghai Institute of Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China

*Corresponding author: dinghai@shu.edu.cn

Abstract. In recent years, deep learning algorithms have been applied in various fields, and with the continuous development of neural networks, the newly emerging network structures have become more and more complex. However, neural networks are often similar to a black box. The evolution of parameters and the changes of neurons are unknowable. How to understand neural networks is an important research topic. Therefore, for beginners and researchers, a correct way to understand and explain neural networks is needed to improve neural networks. Visualization methods have always been used in the interpretability research of knowledge, which is very helpful to show complex algorithms and networks in the form of specific images, so that users and researchers have significant help in studying the structure of the algorithm and optimizing the performance of the algorithm. The visualization method has been applied to many mining fields. The maximization activation function was used to realize the visualization of the kernel function in the convolutional neural network (CNN) in this work, and then the network structure and other content are visualized.

Keywords: Convolutional Neural Network (CNN), Resnet50, Visualization, Kernel Function.

1. Introduction
In recent years, the field of artificial intelligence has developed rapidly, especially in the field of deep learning. Deep learning is to combine low-level features to form a more abstract high-level representation attribute category or feature to discover the distributed feature representation of the data [1]. Although neural network plays an important role in the development of artificial intelligence, it is still a black-box function approximator with limited interpretability. In particular, many models that have emerged are encapsulated, so that they do not understand the specific structure. For example, some deep learning models such as convolutional neural networks and recurrent neural networks are particularly prominent in many aspects, but the network structure is too complex and the model parameters are too many, so that the final output is explained from a mathematical perspective how the final output relates to the parameter band of the model There is no small difficulty [2, 3].
In recent years, the field of visualization is also booming, and visualization is also beginning to be closely related to deep learning. Although the field of deep learning has developed rapidly, deep learning models still have the characteristic of “black box” [4]. Therefore, researchers at home and abroad have explored this problem and put forward a visualization method to explain the deep learning model, through the visualization to conduct interpretability research on the model algorithm, and display the data content and information in the deep learning process to achieve the depth of understanding. The purpose of learning algorithms [5, 6]. Visualization is not only very helpful for model research, but also can greatly promote the work of setting parameters and network structure for users of deep learning models [7].

2. Related work
With the rapid development and application of deep learning networks, visualization has been extended to explain the overall working mechanism of deep learning networks [8]. These works mainly explain it from the perspective of network structure, algorithm implementation and semantic concepts. Several representative methods are:

In 2018, Zhou Bolei and others proposed visualization based on network segmentation. By citing the heterogeneous image data set-Borden, the network segmentation can effectively segment the input image into multiple parts with various semantic definitions. Since the semantics directly represent the meaning of the features, the interpretability of neurons can be significantly improved [9, 10]. There are many types of existing deep learning visualization tools to analyze models and predictions [11], such as DGMTracker [12] and DeepEyes [13] can help developers understand the training process of CNN and GAN.

Moreover, before this, researchers were also studying how to visualize algorithms [14], and they focused on visualizing the data structure of algorithms and the interaction between algorithms. Nowadays, there are two main development directions for deep learning visualization. First, there will be more and deeper learning visualization tools but the functions are the same, and the second is to focus on the research of deep learning visualization. Today's deep learning visualization tools such as tensorboard mainly focus on the feature map and the feature visualization of learning. The future deep learning visualization will run through the model, visualizing the feature changes during the entire model training process, and during use, the visualization of the kernel function in the convolutional network is not deep enough.

3. Visualization of Convolutional Neural Networks
This article focuses on the visualization of convolutional neural networks. How to visualize the output of the convolution kernel and the features learned by the convolution kernel in a convolutional network is a big challenge. This paper builds visualization tools such as kernel function visualization based on the current deep learning visualization tools. The visualization of convolutional neural networks is composed of three parts: network structure visualization, neuron feature map visualization, and convolution kernel visualization during model training.

3.1. Visualization of network structure
This article uses Resnet50 residual neural network to conduct experiments. The structure diagram is shown in the figure.

![Figure 1. Resnet50 network structure diagram.](image)

The first is input, input1 is extracted by a 7*7 convolution kernel, and the convolution kernel step size is 2, and a Max pooling layer is connected. The subsequent stage1 to stage4 are all composed of a...
down-sampling layer and two residual layers. In feature map visualization, the output of the sample after each layer of calculation forms a feature map. The corresponding high-dimensional tensor is obtained through the output of each layer, and the feature map visualization is realized by displaying the feature maps of different channels.

3.2. Visualization of Kernel Function

The visualization of the kernel function is achieved by the Activation Maximization algorithm to realize the visualization of the convolution kernel.

Activation Maximization is an algorithm used to visualize the input preferences of each convolutional layer. In other words, what kind of input can maximize the activation of specific neurons in a specific layer. Each neuron is responsible for extracting specific features, so given an input with that feature; it should be able to get a larger activation value at that neuron. The AM algorithm maximizes the activation function of the neuron by inputting x, and its expression is:

$$x^* = \arg \max_x a_{i,j}(\theta, x)$$

(1)

Where $a_{i,j}(\theta, x)$ represents the activation of the jth convolution in the i-th layer, and is a function of the input x, where $\theta$ represents the weight and bias value $bias$.

The flow of the algorithm is as follows:

Step 1: input an image to obtain the activation $a_{i,l}$ of a specific convolution of a certain layer;

Step 2: When the parameters of the convolutional neural network are fixed, calculate the gradient of activation and the input image $\frac{\delta a_{i,l}}{\delta x}$;

Step 3: Iteratively change the pixels of the input image to maximize activation. The gradient ascent algorithm formula used is:

$$x = x + \eta \cdot \frac{\delta a_{i,l}}{\delta x}$$

(2)

Step 4: This process terminates in a specific mode image $x^*$. This mode is regarded as the preferred input of the neuron.

However, because the AM algorithm is iterative based on random initial input, there is no restriction to ensure that the input that meets the real image is found, so the visual input obtained by the solution is not highly recognizable. In order to obtain a solution close to the real value, it needs to be introduced regular item.

In addition, since the input image when training the network is obtained by subtracting the mean value of the samples in the entire data set from the original image, the direct input x of the network can be considered as a zero mean input. Define the optimization problem as:

$$x^* = \arg \max_x [a_i(x) - R_{\phi}(x)]$$

(3)

In practice, the regularization operation $R_{\phi}(x)$ is used to map x to a more appropriate form, and $R_{\phi}(x)$ does not have to be the derivative of $R_{\phi}(x)$, so the iterative formula of x is:

$$x^* = R_{\phi} \left( x + \eta \cdot \frac{\delta a_{i,l}(x)}{\delta x} \right)$$

(4)

Among them, there are four expressions of $R_{\phi}$: L2 decay formula, Gaussian blur, setting the pixel with a small norm value to 0, and setting the pixel with a small contribution to 0.
4. Experiment

4.1. Introduction to the experimental environment and data set
This experiment was implemented on a Linux server with GPU. The required software environment is: Python3.6+Pytorch1.2+torchvision0.4+TensorBoard2.0. In this experiment, the public data set in the field of image recognition, the cifar10 data set, was selected. The data set has a total of 60,000 color images, and the image size is fixed at 32*32. Every 6000 images are images of one category, there are 10 categories in total. The training set has 50,000 images in the data set; the test set has 10,000 images, and the ratio is 5:1. In the data of the test set, 1000 sheets of each category are randomly selected. The rest is randomly arranged to form the training set.

4.2. Experimental process
Experimental steps:
Step 1: preprocess the data set. Load the cifar10 data set and the corresponding annotations for data preprocessing.
Step 2: Build a ResNet50 network. Call the feature map visualization and kernel function visualization functions in the visualization module. Write the content that needs to be visualized to the data file.
Step 3: Store the data file in the designated directory, open the visualization tool, and access it with a browser.

4.3. Experimental results
This experiment shows a total of three large pages, namely network structure visualization, kernel function visualization, and scalar data visualization.
1. As shown in Figure 2 below, on the left is the network structure visualization page, click on the right to view the output characteristics of each layer. In this experiment, the backbone of the ResNet50 network is the input, convolutional layer and four residual layers.
First of all, in the convolution layer, they are the convolution process, batch normalization, linear rectification function, and maximum pooling layer.

![Figure 2. Visualization of network structure.](image-url)
Figure 3. Visualization of feature maps between layers.

The right side of Figure 2 is the output feature display of Conv. In Figure 3, the upper left corner is the output feature of the Conv layer, the upper right corner is the output feature of the second residual layer, the lower left corner is the output feature of the third residual layer, and the lower right corner is the fourth. The output characteristics of the residual layer can be clearly seen compared with the Conv layer. The deeper the layer, the more obvious the captured features, and the changes between layers can be clearly reflected.

2. In the kernel visualization function, we can see from Figure 4 that in the network structure, the calculation layer that contains the convolution process is placed in the auxiliary layer, and the input can be viewed by clicking the corresponding layer Preferences, the input preferences shown on the right in Figure 4.

Figure 4. ResidualBlock4 network structure.
There is a calculation part of convolution in each residual layer. Therefore, in both the convolutional layer and the residual layer, you can see the input preferences, so as to understand what features the model has learned in ResNet50 and the process of model learning.

Because the input picture is three channels, we choose the output result of one channel to display. The upper left corner is the output preference of the Conv layer, the upper right corner is the output preference of the second residual layer, and the lower left corner is the output preference of the third residual layer. The lower corner is the output preference of the fourth residual layer, which corresponds to the output characteristics of Figure 3 one-to-one. In the image in the upper right corner of Figure 5, it can be seen that the neural network pays more attention to color features in the shallow layer. As the number of layers increases, it is obvious that the neural network pays more attention to texture features in the deep layer. According to the results of the visualization, it is possible to understand the extent to which the features are learned, and the model's effect on feature extraction. According to these bases, the parameters or other details of the model can be well adjusted to achieve a better feature learning effect.

5. Conclusions
This article mainly uses the maximum activation function method to realize the visualization of the kernel function in the convolutional neural network, which can clearly visualize the output characteristics of each layer and the characteristics learned by the neural network after training. Through the visualization of these details, you can it allows researchers to get a close understanding of the various changes and connections between each layer, and can improve their own work through visualization. Future work will show the formulas, parameters and outputs of each step simultaneously, so that it will be more convenient to understand the changes in each layer of the network and the relationship between them.
Acknowledgments
This work is supported by the Science and Technology Commission Shanghai Municipality (STCSM) (No. 1951121000).

References
[1] Yang Y, Chen T. Analysis and Visualization Implementation of Medical Big Data Resource Sharing Mechanism Based on Deep Learning[J]. IEEE Access, 2019, 7(99):156077-156088.
[2] A. W. Harley. An Interactive Node-Link Visualization of Convolutional Neural Networks. In Advances in Visual Computing, vol. 9474, pp. 867–877. Springer International Publishing, 2015.
[3] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Chau. CNN 101: Interactive visual learning for convolutional neural networks. In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, CHI ’20. ACM, Honolulu, HI, USA, 2020.
[4] Harley A W. An Interactive Node-Link Visualization of Convolutional Neural Networks[J]. 2015.
[5] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. IEEE Transactions on Visualization and Computer Graphics, 24(1):667–676, Jan. 2018.
[6] M. Kahng and D. H. Chau. How Does Visualization Help People Learn Deep Learning? Evaluation of GAN Lab. In IEEE VIS 2019 Workshop on EValuation of Interactive VisuAl Machine Learning Systems, Oct. 2019.
[7] Cho S, Choi J. Visualization of Convolutional Neural Networks for Time Series Input Data[J]. 2020.
[8] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding Neural Networks Through Deep Visualization. In ICML Deep Learning Workshop, 2015.
[9] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]//European conference on computer vision. Springer, Cham, 2014: 818-833.
[10] Zhou Z, Cai H, Rong S, et al. Activation maximization generative adversarial nets[J]. arXiv preprint arXiv:1703.02000, 2017.
[11] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. IEEE Transactions on Visualization and Computer Graphics, 24(1):88–97, Jan. 2018.
[12] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu. Analyzing the Training Processes of Deep Generative Models. IEEE Transactions on Visualization and Computer Graphics, 24(1):77–87, Jan. 2018.
[13] N. Pezzotti, T. Hollt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks. IEEE Transactions on Visualization and Computer Graphics, 24(1):98–108, Jan. 2018.
[14] C.A. Shaffer, M.L. Cooper, A.J.D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards. Algorithm Visualization: The State of the Field. ACM Transactions on Computing Education, 10(3):1–22, Aug. 2010.