Improving Network Accuracy via Network Manipulation

Taehyung Kim
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92037
`tak008@ucsd.edu`

Gauri Samith
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92037
`gsamith@ucsd.edu`

Kent Utama
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92037
`kutama@ucsd.edu`

Abstract

Deep learning has been growing rapidly over the past couple decades due to its ability in solving extremely complex problems. However, this machine learning method is often considered as a "black box" since it is unclear how the neurons of a deep learning model work together to arrive at the final output. A recently found method called Network Dissection has solved this interpretability issue by coming up with a visual that shows what each neuron looks for and why. Results show that groups of neurons detect different human-interpretable concepts. Inspired by Network Dissection, we will be investigating methods to improve a convolutional neural network's prediction power based on the knowledge of the role of each neuron. One such method is FocusedDropout: keeping neurons that focus on human-readable concepts while discarding everything else. Additionally, the effect of regularization of input gradients on model robustness and interpretability is also tested.

1 Introduction

Deep Neural Networks consist of a large number of hidden layers and units, the functionality of which we are not fully aware. While we may understand feature engineering at the input level and the final output classification, the intermediate hidden features are somewhat of a "black box". It is not clear how the network is solving a task of high complexity. However, it has been observed that many hidden units are actually associated with concepts that are human-interpretable. Thus, it becomes important to really understand the functionality and contribution that each of these hidden units has to the final output of the network. In order to understand this, network dissection is used - a method to correlate semantic concepts identified within a complex convolutional neural network (CNN) [1]. It aims to identify, visualize and quantify the contributions of each individual unit within a deep CNN.

Findings from network dissection show us that different groups of neurons correspond to different human-interpretable concepts such as trees, snow, etc. Taking this one step further, the goal of this paper will be to find out if we can improve a network's prediction accuracy given this information regarding each neuron's roles. Until recently, most methods that aims to improve neural networks do not include specific knowledge about these networks, such as regularization and fine-tuning. A method that we will implement in this paper is FocusedDropout: a dropout method that retains neurons that contain target-related features while discarding other neurons [2]. Additionally, we are also incorporating input gradient regularization, which involves minimizing the rate of change of loss with respect to the input features [3]. This is said to improve the robustness as well as the interpretability of the network.

2 Network Dissection

A strength of network dissection is that the key idea relates to analyzing the visual concept associated with units within the network, which is a concept that image-based neural networks employ. Thus, network dissection can be applied to a wide range of models rather than specific model architectures for which the algorithm is particularly designed. While this does restrict network dissection to image-based tasks, the versatility of network dissection streamlines the understanding of this type of model significantly. In addition, network dissection can be applied to multiple tasks as long as the units learn relations to visual concepts. This is exemplified within this paper as network dissection is applied to image classification. The variety of tasks and model architectures that network dissection applies to without the need for heavy customization makes it a powerful tool to analyze the role of individual hidden units and find insights about the black box natures of neural networks.

The first step in training a network on a scene classification task is to identify units that are capable of detecting objects. In this paper, we use the Places365 dataset from the Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory Scene Recognition Database [4] to train a convolutional neural network (CNN) using the VGG-16 architecture to classify images into 365 scene categories. Since the last convolution layer has the most object-detecting units [1], we mainly focus on conv5_3 layer.

The units compute activation functions $a_u(x, p)$ that output a signal at every position p of the image given a test image x. Bilinear upsampling facilitates the visualization and analysis of filters with low-resolution outputs. Denote by t_u the top 1% quantile level for a_u: That is, writing $\mathbb{P}_{x_p}[\cdot]$ to indicate the probability that an event is true when sampled over all positions and images, we define the threshold $t_u \equiv \mathbb{P}_{x_p}[a_u(x,p) > t] \geq 0.01$. Activation regions above the threshold are highlighted in visualizations by $\{p|a_u(x,p) > t_u\}$. Fig.2 shows how this region corresponds to semantics, for example, the heads of everyone in the picture. We use a computer vision segmentation model [5] that predicts the presence of the visual concept c within image x at position p based on the agreement between each filter and the visual concept c. $s_c : (x,p) \rightarrow \{0,1\}$ is trained to identify filters that match semantic concepts. Using the intersection over union (IoU) ratio, we quantify the agreement between concept c and unit u:

$$IoU_{u,c} = \frac{\mathbb{P}_{x,p}[s_c(x,p) \wedge (a_u(x,p) > t_u)]}{\mathbb{P}_{x,p}[s_c(x,p) \vee (a_u(x,p) > t_u)]}$$

This IoU ratio is computed on the set of held-out validation set images. Within this validation set, each unit is scored against 1,825 segmented concepts c, including object classes, parts of objects, materials, and colors. Then, each unit is labeled with the highest-scoring matching concept. Fig.3 shows several labeled concept detector units and the five images with the highest unit activations.

Within layer conv 3, we find 512 units matching 51 object classes, 22 parts, 12 materials, and eight colors. In some cases, more than one unit matches the same visual concept. Fig.1 shows the frequency of units matching segmented concepts with unit types in layer conv5_3, excluding units with IoU ratios of 4 or 4%. A total of 28 object classes, 25 parts, nine materials, and eight colors can be detected by units at the last convolutional layer, while the total number of object parts peaks two layers earlier, at layer conv5_1, where units match 28 object classes, 25 parts, and nine materials.

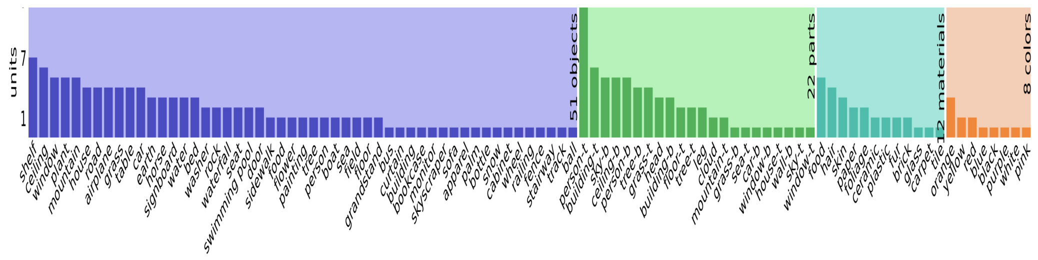

Figure 1: Segment Concepts in Conv5_3

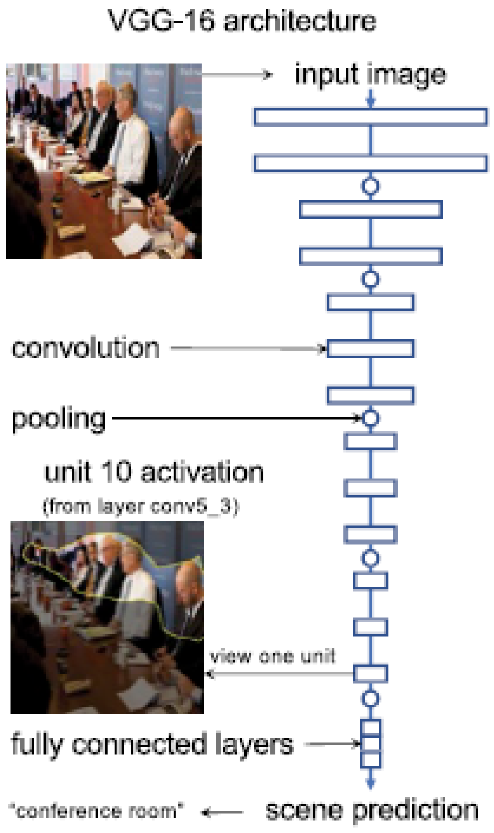

Figure 2: Semantic Regions

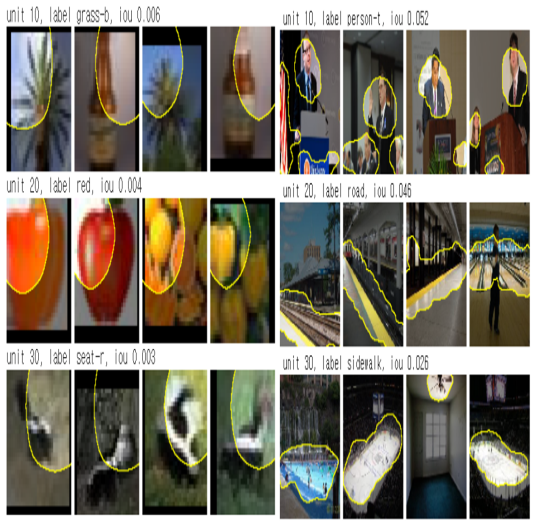
unit 10, label grass-b, iou 0.006
unit 10, label person-t, iou 0.052
unit 20, label red, iou 0.004
unit 20, label road, iou 0.046
unit 30, label seat-r, iou 0.003
unit 30, label sidewalk, iou 0.026

Figure 3: Labeled Concept Detector Units on Cifar-100(Left) and Places365(Right)

3 Methods

3.1 Method 1: Network Dissection Intervention

Network dissection intervention is the process of modifying internal representations to improve the performance of the network on a specific task. For example, if a network is trained to recognize images of cars, but it is found that it has trouble distinguishing between different models of cars, network dissection intervention can be used to identify the units that are responsible for recognizing different car models and modify them to improve performance on this task. As we use Places365 data, we calculate the accuracy for every 365 classes by removing one unit at once and determine which unit assists the network by increasing accuracy. After dividing units into good and bad, we reduce the weights or output of bad units.

3.2 Method 2: FocusedDropout

FocusedDropout is a highly targeted approach that makes the network focus on the most important features (based on the idea of Network Dissection) while dropping other features. The methodology for FocusedDropout is shown in Figure 4. To get a visual understanding of FocusedDropout, Figure 5 shows the process of creating the binary mask that will be applied to every channel based on the highest activation channel. For this paper, we will be using VGG-16 for the CNN and CIFAR-100 for the dataset.

```
Algorithm 1: FocusedDropout
Input: whole channels of the previous layer
    $C = [c_1, c_2, c_3, \ldots, c_n]$), mode,
Output: $C^*$
if mode = Inference, then
    Return $C^* = C$
end
Calculate the average activation value for each $c_i$ as
  $w_i$, Get the $c_k$ with the max $w_i$ ;
Get the unit having the highest activation value as
  $c_k(\overline{x}, \overline{y})$ ;
Make the all-zero mask $m$ having the same size with
  $c_i$;
foreach $m(i, j)$ in $m$ do
    if $c_k(i, j) > random(0.6, 0.9) \cdot c_k(\overline{x}, \overline{y})$ then
        $m(i, j) = 1$;
    end
end
Return $C^* = C \odot m$
```

Figure 4: FocusedDropout Algorithm

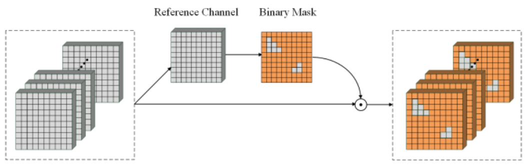

Figure 5: FocusedDropout Binary Mask

3.3 Method 3: Input Gradient Regularization

Input Gradient Regularization was initially introduced as "double back-propagation", and involved training neural networks by minimizing quadratic loss of the network as well as the rate of change of loss with respect to the input features. In this particular implementation, cross-entropy loss is used instead. The goal of this approach is to ensure that even if any input changes slightly, the KL divergence between predictions and labels will not change significantly. In turn, it serves as an interesting means to prevent adversarial attacks. In our paper, we aim to implement this method to check for general improvement in network accuracy, and combine it with network dissection to check for improved accuracy and interpretability. We formulate our overall loss function for this method as:

$$\theta^* = \arg\min_{\theta} \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log f_\theta(X_n)_k + \lambda \sum_{d=1}^{D} \sum_{n=1}^{N} \left(\frac{\partial}{\partial x_d} \sum_{k=1}^{K} -y_{nk} \log f_\theta(X_n)_k \right)^2,$$

The object function is presented more concisely as:

$$\arg\min_{\theta} H(y, \hat{y}) + \lambda ||\nabla_x H(y, \hat{y})||_2^2,$$

4 Experiment & Results

4.1 Network Dissection Intervention

For Network Dissection Intervention, we trained VGG-16 on Places365 and experimented with conv5_3 layer as the layer had units that captured most objects. To test the hypothesis, we performed the following steps:

1. Unit Deletion Analysis: We calculated the accuracy of the VGG16 model with and without each unit in layer conv5_3 to identify the best and worst performing units
2. Unit Activation Analysis: We modified the output of layer conv5_3 for the worst and best-performing units and reevaluated the accuracy of the model
3. Weight Modification Analysis: We modified the weights of layer conv5_3 for the worst and best-performing units and evaluated the accuracy of the model

For both modifying unit output and weights of the layer, we experimented with dividing and multiplying constant numbers and stopped the experiment when the accuracy got below the baseline. The results are shown in Table 1.

Table 1: Network Dissect Intervention Accuracy

Unit Type & Target	Baseline	Divide/Multiply 2	Divide/Multiply 3
worst unit output division	76.58%	76.52%	75.91%
worst unit weight division	76.58%	**76.6%**	76.02%
best unit weight division	76.58%	**76.35%**	75.56%

4.2 Input Gradient Regularization & FocusedDropout

For FocusedDropout and Input Gradient Regularization, we trained our VGG-16 on CIFAR-100 with the following training settings:

- Batch size of 128
- Stochastic Gradient Descent Optimizer (Learning rate = 1e-2, momentum = 9e-1, weight decay = 5e-4)
- CosineAnnealingLR Scheduler (T_max = 200)
- 150 Epochs

For input gradient regularization specifically, we made use of the L2 norm for regularization with a penalty of $\lambda = 0.1$.

The classification results are shown in Table 2.

Table 2: Accuracy of VGG-16 on CIFAR-100 (150 epochs)

Regularization Method	Training Accuracy	Test Accuracy
None (Baseline)	99.97%	72.32%
Dropout rate 0.2	98.04%	68.16%
FocusedDropout par_rate 0.1	90.71%	**72.90%**
L2 (lambda 0.1) Input Gradient Regularization	99.92%	**71.67%**

4.3 Network Dissection on reported and experiment models

After training VGG-16 on CIFAR-100, we performed Network Dissection on it and compared the results with that in the original Network Dissection paper (VGG-16 on Places365).

The results are shown side-by-side in Figure 6.

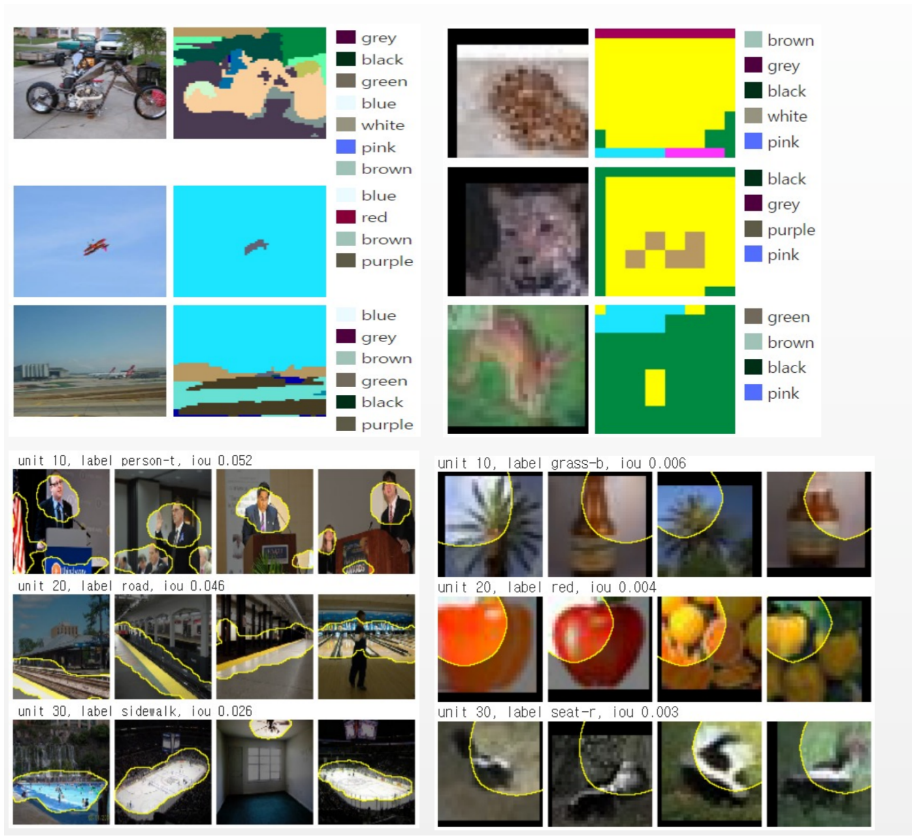
grey
black
green
blue
white
pink
brown
blue
red
brown
purple
blue
grey
brown
green
black
purple
brown
grey
black
white
pink
black
grey
purple
pink
green
brown
black
pink
unit 10, label person-t, iou 0.052
unit 20, label road, iou 0.046
unit 30, label sidewalk, iou 0.026
unit 10, label grass-b, iou 0.006
unit 20, label red, iou 0.004
unit 30, label seat-r, iou 0.003

Figure 6: VGG16 Masks for Places365 (left) and CIFAR100 (right)

5 Discussion

Based on the images from Network Dissection, we see that VGG16 on CIFAR100 does not detect scenes that well while the reported result (VGG16 on Places365) shows clear classifications of high-level concepts. This is highly suspected due to the fact that CIFAR100 is a dataset that revolves around object detection while Places365 is a dataset that revolves around scene detection. This difference in the datasets might prevent our model from learning high-level concepts. Additionally, CIFAR100 is a subset of a much bigger dataset and we feel that using a larger dataset will also allow for better generalization and conceptual understanding.

A next step for this project would be to train our VGG16 models on Places365. This was not done in this project due to limited resources: it was estimated to take around 50 days for our model to train on Places365 for 100 epochs with our GPU. As a result, we stuck with CIFAR100 for this project.

Through this project, we see that a network's information can be utilized to manipulate the network to achieve better performance.

6 References

[1] D. Bau, J. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, A. Torralba, Understanding the role of individual units in a deep neural network, 7 July 2020

[2] M. Liu, T. Xie, X. Cheng, J. Deng, M. Yang, X. Wang, M. Liu, FocusedDropout for Convolutional Neural Network, 30 July 2022

[3] A. Ross, F. Doshi-Velez, Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients, 26 November 2017

[4] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, “Learning deep features for scene recognition using places database” in Advances in Neural Information Processing Systems (Curran Associates, Red Hook, NY, 2014), pp. 487–495.

[5] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, J. Sun, “Unified perceptual parsing for scene understanding” in Proceedings of the European Conference on Computer Vision (Springer, Berlin, Germany, 2018), pp. 418–434.