Mathematical Foundations of Computing
©American Institute of Mathematical Sciences
Volume **1**, Number **2**, May **2018**
doi:10.3934/mfc.2018008
pp. **X–XX**

HOW CONVOLUTIONAL NEURAL NETWORKS SEE THE WORLD — A SURVEY OF CONVOLUTIONAL NEURAL NETWORK VISUALIZATION METHODS

Zhuwei Qin†

†George Mason University
4400 University Dr, Fairfax, VA 22030, USA

Fuxun Yu†, Chenchen Liu‡ and Xiang Chen†*

‡Clarkson University
8 Clarkson Ave, Potsdam, NY 13699, USA

(Communicated by Zhipeng Cai)

Abstract. Nowadays, the Convolutional Neural Networks (CNNs) have achieved impressive performance on many computer vision related tasks, such as object detection, image recognition, image retrieval, *etc.* These achievements benefit from the CNNs' outstanding capability to learn the input features with deep layers of neuron structures and iterative training process. However, these learned features are hard to identify and interpret from a human vision perspective, causing a lack of understanding of the CNNs' internal working mechanism. To improve the CNN interpretability, the CNN visualization is well utilized as a qualitative analysis method, which translates the internal features into visually perceptible patterns. And many CNN visualization works have been proposed in the literature to interpret the CNN in perspectives of network structure, operation, and semantic concept.

In this paper, we expect to provide a comprehensive survey of several representative CNN visualization methods, including *Activation Maximization*, *Network Inversion*, *Deconvolutional Neural Networks* (*DeconvNet*), and *Network Dissection* based visualization. These methods are presented in terms of motivations, algorithms, and experiment results. Based on these visualization methods, we also discuss their practical applications to demonstrate the significance of the CNN interpretability in areas of network design, optimization, security enhancement, *etc.*

1. **Introduction.** The Convolutional Neural Networks (CNNs) have been widely investigated as one of the most promising solutions for various computer vision related tasks, such as object detection [20, 58], image recognition [36, 63, 66, 27], image retrieval [22, 25], *etc.*

Inspired by the hierarchical organization of the human visual cortex [34], the CNN is constructed with many intricately interconnected layers of neuron structures. These neurons act as the basic units to learn and extract certain features from the input. With the network complexity increment caused by the neuron layer

2010 *Mathematics Subject Classification.* Primary: 58F15, 58F17; Secondary: 53C35.

Key words and phrases. Deep learning, convolutional neural network, CNN feature, CNN visualization, network interpretability.

The authors are supported by NSF Grant CNS-1717775.

* Corresponding Author: Xiang Chen.

depth, the performance of the input feature extraction is also enhanced. For example, *AlexNet* – one of the most representative CNNs, has 650K neurons and 60M related parameters [36]. Also, sophisticated algorithms are proposed to support the training and testing of such a complex network, and the backpropagation method is widely applied to train the CNN parameters through multiple layers [39, 44]. Furthermore, to fine-tune the network to specific functions, large pools of labeled data are required for iteratively training the massive neurons and connection weights. So far, many high-performance CNN designs have been proposed, such as *AlexNet* [36], *VGG* [63], *GoogleNet* [66], *ResNet* [27], *etc.* Some of the designs can even achieve beyond human-level accuracy on object recognition tasks [61].

Although the CNNs can achieve competitive classification accuracy, the CNNs still suffer from high computational cost, slow training speed, and security vulnerability [67, 38, 4]. One major reason causing these shortcomings is the lack of network interpretability, especially the limited understanding of the internal features learned by each convolutional layer: Mathematically, the convolutional layer neurons (namely the convolution filters) convolve with the input image or the outputs of the previous layer, the results are considered as learned features and recorded in the feature maps. With deeper layers, the neurons are expected to extract higher level features, and eventually converge to the final classification. However, as the CNN training is considered as a black-box process and the neurons are designed in the format of simple matrices, the formation of those neuron values are unpredictable and the neuron meanings are impossible to directly explained. Hence, the poor network interpretability significantly hinders the robustness evaluation of each network layer, the further optimization on the network structure, as well as the network adaptability and transferability to different applications [53, 60].

A qualitative way to improve the network interpretability is the network visualization, which translates the internal features into visually perceptible image patterns. This visualization process is referred from the human visual cortex system analysis: In a human brain, the human visual cortex is embedded in multiple vision neuron areas [57]. In each vision neuron area, numerous neurons selectively respond to different features, such as colors, edges, and shapes [35, 54]. To explore the relationship between the neurons and features, researchers usually find the preferred stimulus to identify individual kind of the response and illustrate the response to certain visual patterns. The CNN visualization also follows such an analytical approach to realize the CNN interpretability.

Up to now, many effective network visualization works have been proposed in the literature, and several representative methods are widely adopted: 1) Erhan *et al.* proposed the *Activation Maximization* to interpret traditional shallow networks [14, 28, 68]. Later, this method was further improved by Simonyan *et al.*, which synthesized an input image pattern with the maximum activation of a single CNN neuron for visualization [62]. This fundamental method was also extended by many other works with different regularizers for interpretability improvement of the synthesized image patterns [71, 52, 50]. 2) Besides the visualization of a single neuron, Mahendran *et al.* revealed the CNN internal features in the layer level [46, 47]. The *Network Inversion* was proposed to reconstruct an input image based on multiple neurons' activation to illustrate a comprehensive feature map learned by each single CNN layer. 3) Rather than reconstructing an input image for feature visualization, Zeiler *et al.* proposed the *Deconvolutional Neural Network based Visualization* (*DeconvNet*) [72], which utilized the *DeconvNet* framework to

project the feature map to an image dimension directly. With the direct projecting, *DeconvNet* can highlight what patterns in the input image activate the specific neurons and hence link the neurons and the meaning of input data directly. 4) Recently, Zhou *et al.* [3] proposed the *Network Dissection based Visualization*, which interpreted the CNN in the semantic level. By referencing a heterogeneous image dataset – *Borden*, the *Network Dissection* can effectively partition the input image into multiple sections with various semantic definitions. As the semantics directly represent the feature meanings, the neuron interpretability can be significantly enhanced.

This survey paper is expected to provide a comprehensive review of these representative CNN visualization methods, in terms of motivations, algorithms, and experiment results. We also discuss the practical applications of the CNN visualization, demonstrating the significance of the network interpretability in areas of network design, optimization, security enhancement, *etc.*

The rest of the paper is organized as follows: In Section 2, we introduce the background knowledge of the CNN and visualization. In Sections 3∼6, we describe the four aforementioned representative visualization methods, namely the *Activation Maximization*, *DeconvNet*, *Network Inversion*, and *Network Dissection* based visualization. In Section 7, we present several CNN visualization applications. In Section 8, the CNN visualization research potentials are discussed with the conclusion.

2. **Background.** In this section, we introduce the background knowledge of the CNN structure, algorithm, and CNN visualization.

2.1. **CNN structure.** In machine learning, the CNN is a type of deep neural networks (DNNs), which has been widely used for computer vision related tasks. Fig. 1 shows a representative CNN structure – *CaffeNet* [33], which is a replication of *AlexNet* with 5 convolutional layers (CLs) and 2 max-pooling layers (PLs) followed by 3 fully-connected layers (FLs):

Convolutional Layer: Fig. 1 demonstrates the CNN structure, and the yellow blocks represent the convolutional filters – neurons in the convolutional layers for feature extraction. These filters perform the convolution process to transform the input images or previous layer feature maps into the output feature maps, which are denoted as the blue blocks in Fig. 1.

Fig. 2 (a) shows the detailed convolutional process of the first CL. The convolutional filters in the first layer have three channels corresponding to the three RGB

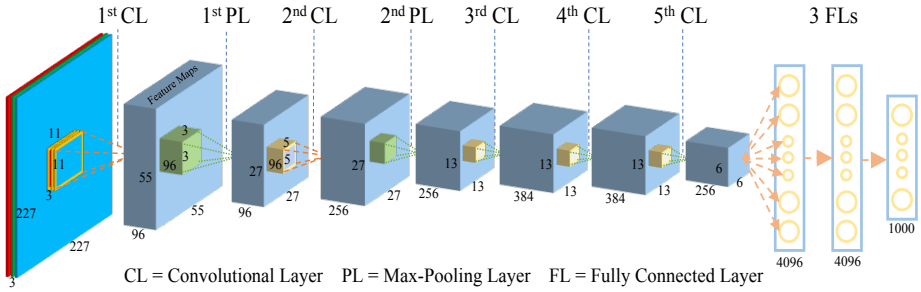

FIGURE 1. *CaffeNet* architecture

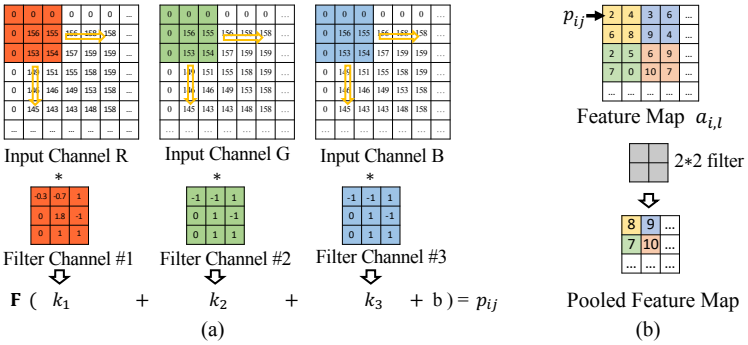

FIGURE 2. Convolutional and max-pooling process

color dimensions of the raw input images. Each filter channel performs dot production in a small region of the input data to compose a color specific feature channel. This process is usually followed by an activation function F, usually *relu* [5], which gives a summation of dot productions when positive or 0 otherwise. Hence, we can get a color comprehensive element p_{ij} of the final rectified feature map. Based on the small region convolution, each filter is replicated across the entire input image. Multiple p_{ij}s would produce the final rectified feature map (or activation map) $a_{i,l}$.

The rectified feature maps represent the extracted features and would act as the inputs for the next CL. Mathematically, for the filter i in layer l, This process can be viewed as:

$$a_{i,l+1} = F(\sum w_{i,l} a_{i,l} + b_{i,l}), \tag{1}$$

where w, b represents the weights and bias parameters respectively.

With such a process, the filters act as feature extractors from the original input image to learn the useful features for classification.

Pooling Layer: As shown in Fig. 1, after each CL, it's optimal to apply a PL on the output feature maps. As denoted by the green blocks, the pooling filters perform the down-sampling operation to reduce the data dimension of the input feature maps ($a_{i,l}$). Fig. 2 (b) shows the max-pooling process, which is a widely adopted pooling method. The max-pooling is achieved by applying a 2×2 pooling window to select the maximal element of a 2×2 region of the input feature map with a stride of 2. This process aggressively reduces the spatial size of the feature maps and condense the extracted feature information.

Hence, the pooling layers contribute the CNN with fewer data redundancy and therefore less data processing workload.

Fully-connected Layer: In Fig. 1, each yellow circle represent one neuron in the FLs, which is connected to all the previous input feature maps. The FLs perform a comprehensive feature evaluation based on the features extracted by the CLs, and generate an N-dimensional probability vector, where N is the number of the classification targets. For example, in the digit classification task, N would be 10 for the digits of 0∼9 [40].

After the final layer of FLs, a *SoftMax* function is used to generate the final classification probability as defined in the Eq. 2:

$$P_i = \frac{e^{a_i}}{\sum_{N=1}^{10} e^{a_n}}, \tag{2}$$

where a_i is i^{th} neuron output in the final FL layer. This function normalizes the final FL layer output to a vector of values between zero and one, which gives a probability over all 10 classes.

By applying the above-mentioned hierarchical structured layers, CNNs transform the input image layer by layer from the original pixel values to the final probability vector P, in which the largest P_i indicates the most predicted class.

2.2. **CNN algorithm.** The CNNs not only benefit from the deep hierarchical structure, but also the delicate learning algorithm [39]. The learning algorithm aims to minimize the training error between the predicted values and actual labels by updating the network parameters, which are quantified by the loss function. The training error can be viewed as:

$$C(w,b) = \frac{1}{n}\sum_{n}^{i=1} L(w,b,x_i,y_i), \tag{3}$$

where $L(\cdot)$ represents the loss function, and $(x_1,y_1),...(x_n,y_n)$ represent the training examples. During the learning process, a square loss is usually applied, then the loss function will be:

$$L(w,b,x_i,y_i) = (y_i - f(w,b,x_i))^2, \tag{4}$$

where $f(\cdot)$ indicates the predicted values calculated by the whole CNN:

$$f(w,b,x) = F(\sum w_{i,l} F(w_{i,l-1} F(...F(\sum w_{i,1} x_{i,0} + b_{i,0})...) + b_{i,l-1}) + b_{i,l}). \tag{5}$$

In order to minimize the $C(w,b)$, a partial derivative $\partial C/\partial(w,b)$ with respect to each weight w and bias b is calculated by backpropagating through all the layers in the CNN. The gradient descent method is utilized to iteratively update all parameter values. The update procedure for w from iteration j to $j+1$ can be viewed as:

$$w_{j+1} = w_j - \eta \cdot \frac{\partial C(w;x,y)}{\partial w}, \tag{6}$$

where η is the learning rate. Before the learning process, the parameters are usually randomly initialized [21]. With the learning process, the convolutional filters become well configured to extract certain features. The features captured by convolutional filters can be demonstrated by visualization.

A lot of works have been proposed to optimize the structure and algorithm of the CNNs. For example, much deeper network structures have been investigated, such as *VGG*, *GoogleNet*, and *ResNet*. At the same time, some regularization and optimization techniques have been applied, such as dropout [65], batch normalization [31], momentum [56], and adagrad [13]. As a result, CNNs have been well optimized and widely used in computer vision related tasks. However, the CNNs still suffer from high computational cost, slow training speed, and large training dataset requirement, which highly compromise the applicability and performance efficiency [26]. Hence, these weakness require more understanding about the CNN working mechanism to further optimize the CNN.

2.3. **CNN visualization mechanism.** CNN visualization is a well utilized qualitative method to analysis the CNN working mechanism regarding the network interpretability. The interpretability is related to the ability of the human to understand the CNNs, which can be improved by demonstrating the internal features learned by CNNs. Visualization greatly helps to interpret the CNN's internal features, since it utilizes the human visual cortex system as a reference.

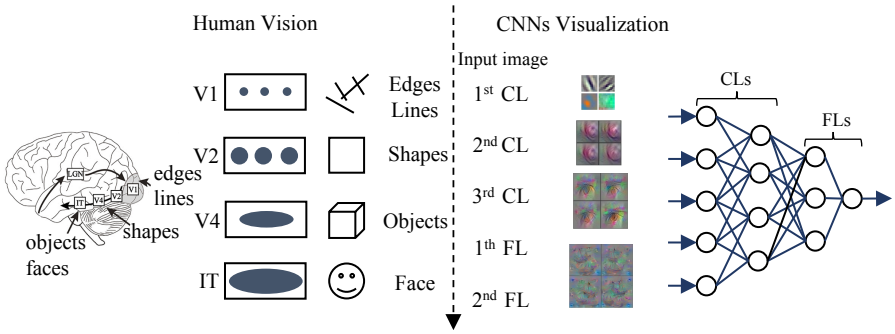

FIGURE 3. Human vision and CNNs visualization

Fig. 3 shows how human visual cortex system process the visual information and how CNNs extract the features. As shown in the left part Fig. 3, the human visual system processes the object features in a feed-forward and hierarchical approach through multiple visual neuron areas. When humans recognize a face, the visual neurons with small receptive fields in the lower visual neuron area (*e.g.* V1), are sensitive to basic visual features [48, 30, 29], such as edges and lines. In the higher visual neuron areas (*e.g.* V2 and V4), the visual neurons have larger receptive fields, and are sensitive to complex features, such as shapes and objects. In the visual neuron area of IT, the visual neurons have the largest and most comprehensive receptive fields, therefore they are sensitive to the entire face.

For CNN interpretability study, researchers found the similar feature representation through the CNNs visualization as shown in the right part of Fig. 3. Typically, the CNN feature extraction starts with small features such as edges and colored blobs in the first convolutional layer. Then the feature extraction progresses into general shapes and partial objects with deeper layers, and ends in a final classification with the fully-connected layers. By comparing the functionalities of brain visual neurons' receptive fields to the CNN's neurons [37], visualization illustrates the functionalities of each component in the CNNs.

2.4. **CNNs visualization methods.** The similarity between how the human vision system and CNN recognizes image inspired research to work on interpreting CNNs, and lots of CNN visualization works of the learned features have been widely discussed. In the early research stage, the visualization mainly focused on the low-level features [55, 42, 43]. With the rapid developments and implementations of

Method	Interpretation Perspective	Focused Layer	Applied Network	Representative Study
Activation Maximization	Individual Neuron with visualized pattern	CLs FLs	Auto-Encoder, DBN, AlexNet	[26]
Deconvolutional Neural Networks	Neuron activation in input image	CLs	AlexNet	[55]
Network Inversion	One layer	CLs FLs	HOG, SIFT, LBD, Bag of words, CaffeNet	[29][64]
Network Dissection	Individual Neuron with semantic concept	CLs	AlexNet, VGG, GoogLeNet, ResNet	[32][70]

TABLE 1. Visualization methods

CNNs, the visualization has been extended to interpret the overall working mechanism of CNNs. In Table 1, we give a brief review of four representative visualization methods, namely *Activation Maximization*, *Deconvolutional Networks* (*DeconNet*), *Network Inversion*, and *Network Dissection*:

- In *Activation Maximization*, a visualized input image pattern is synthesized to illustrate a specific neuron's max stimulus in each layer;
- *DeconNet* utilizes an inversed CNN structure, which is composed deconvolutional and unpooling layers, to find the image pattern in the original input image for a specific neuron activation;
- *Network Inversion* reconstructs an input image based on the original image from a specific layer's feature maps, which reveals what image information is preserved in that layer;
- *Network Dissection* describes neurons as visual semantic detectors, which can match six kinds of semantic concepts (*e.g.* scene, object, part, material, texture, and color).

To compare these methods directly, we summarize the overview, algorithms, and visualization results of these methods: 1) The overview summarizes the history and represent works in this line of work. 2) The algorithms explain how this method works for the CNNs visualization. 3) The visualization results provide a comprehensive understanding how CNNs extract features.

3. **Visualization by activation maximization.**

Synthesize an input pattern image that can maximize a specific neuron's activation in arbitrary layers.

3.1. **The overview.**

Activation Maximization (AM) is proposed to visualize the preferred inputs of neurons in each layer. The preferred input can indicate what features of a neuron has learned. The learned feature is represented by a synthesized input pattern that can cause maximal activation of a neuron. In order to synthesize such an input pattern, each pixel of the CNN's input is iteratively changed to maximize the activation of the neuron.

The idea behind the AM is intuitive, and the fundamental algorithm was proposed by Erhan *et al.* in 2009 [14]. They visualized the preferred input patterns for the hidden neurons in the Deep Belief Net [28] and the Stacked Denoising Auto-Encoder [68] learned from the MNIST digit dataset [40]. Later, Simonyan *et al.* utilized this method to maximize the activation of neurons in the last layer of CNNs [62]. Google also has synthesized similar visualized patterns for their inception network [49]. Yosinksi *et al.* further applied the AM in a large scale, which visualized the arbitrary neurons in all layers of a CNN [71]. Recently, a lot of optimization works have followed this idea to improve the interpretability and diversity of the visualized patterns [52, 50]. With all these works, the AM has demonstrated great capability to interpret the interests of neurons and identify the hierarchical features learned by CNNs.

3.2. **The algorithm.**

In this section, the fundamental algorithm of the AM is presented. Then, another optimized AM algorithm is discussed, which dramatically improves the interpretability of visualized patterns by utilizing a deep generator network [50].

3.2.1. *Activation maximization.* The fundamental algorithm of the AM can be viewed as synthesizing a pattern image x^*, which maximizes the activation of a target neuron:

$$x^* = \underset{x}{\operatorname{argmax}}\, a_{i,l}(\theta, x), \tag{7}$$

where θ denotes the network parameter sets (weight and bias).

This process can be divided into four steps:

(1) An image $x = x_0$ with random pixel values is set to be the input to the activation computation.

(2) The gradients with respect to the noise image $\frac{\partial a_{i,l}}{\partial x}$ are computed by using backpropagation, while the parameters of this CNN are fixed.

(3) Each pixel of the noise image is changed iteratively to maximize the activation of the neuron, which is guided by the direction of the gradient $\frac{\partial a_{i,l}}{\partial x}$. Every single iteration in this process applies the update:

$$x \leftarrow x + \eta \cdot \frac{\partial a_{i,l}(\theta, x)}{\partial x}, \tag{8}$$

where η denotes the gradient ascent step size.

(4) This process terminates at a specific pattern image x^*, when the image without any noise. This pattern is seen as preferred input for this neuron [71].

Typically, we are supposed to use the unnormalized activation $a_i(\theta, x)$ of class c in the final CNN layer of this visualization network , rather than the probability returned by the *SoftMax* in Eq. 2. Because the *SoftMax* normalize the final layer output to a vector of values between zero and one, the maximization of the class probability can be achieved by minimizing the probability of other classes. This method can be applied to any kinds of CNNs as long as we can compute the aforementioned gradients of the image pattern.

3.2.2. *Activation maximization with regulation.* However, the AM method has a considerable shortcoming: as the CNN becoming deeper, the visualized patterns in higher layers are usually tend to be unrealistic and uninterpretable. In order to find the human-interpretable patterns, many regularization methods have been experimentally shown to improve the interpretability of the patterns.

A regularization parameter of $\lambda(x)$ is usually introduced to bias the visualized pattern image:

$$x^* = \underset{x}{\operatorname{argmax}}(a_{i,l}(\theta, x) - \lambda(x)). \tag{9}$$

Different methods are adopted to implemented the $\lambda(x)$, such as ℓ_2 decay, Gaussian blur, mean image initialization, and clipping pixels with very small absolute value [62, 71, 70, 52]. For example, the ℓ_2 decay tends to prevent a small number of extreme pixel values from dominating the visualized patterns. The Gaussian blur penalize high frequency information in the visualized patterns, and the contribution of a pixel is measured as how much the activation increases or decreases when the pixel is set to zero. Each of these regularization methods can be applied to the AM individually or cooperatively. In the Section 3.3.2, we shows the bias patterns by applying these regularization methods to improve the interpretability.

3.2.3. *Activation maximization with generator networks.* Instead of utilizing regularizer $\lambda(x)$ to bias the visualized, Nguyen *et al.* [50] utilized a image generator network [10, 24] to to replace the iterative random pixel tuning, which maximize the activation of the neurons in the final CNN layer. The synthesized pattern image

by the generator is more close to the realistic image, which greatly improves the interpretability of the visualized patterns.

Recently, most of the generator networks related works are based on Generative Adversarial Networks (GAN) [24]. GANs can learn to mimic any distribution of data and generate realistic data samples, such as image, music, and speech, which is featured with a complementary composition of two neural networks: One generative network takes noise as input and aim to generate realistic data samples. Another discriminator network receives the generated data samples from the output of the generative network and the real data samples from the training data sets, which aim to distinguishes between the two sources. The goal of generative network is to generate passable data samples, to lie without being distinguished by the discriminator network. The goal of the discriminator is to identify images coming from the generative network as fake. After fine training of both the networks, the GAN eventually achieves a balance, where the discriminator can hardly distinguish generated data samples from real data samples. In such a case, we can claim that the generative network has achieved an optimal capability in generating realistic samples. So far GANs have particularly produced excellent results in image data, and primarily been used to generate samples of realistic images [9, 41, 2].

Benefit from the success of GANs, the generative network is utilized to overcome the aforementioned shortcoming of AM that the visualized patterns in higher layers are usually tend to be unrealistic and uninterpretable. The generative network is utilized to generate or synthesize the pattern image that maximize the activation of the selected neuron $a_{i,l}$ in the final layer This method is called *Deep Generative Network Activation Maximization* (DGN-AM).

The DGN-AM implemention can be view as:

$$x^* = \underset{x}{\operatorname{argmax}} \left(a_{i,l}(\theta, G(x)) - \lambda(x)\right), \tag{10}$$

where G indicates the generative network that takes the noise image as input. It can synthesize the pattern image that causes high activation of the target neuron $a_{i,l}$. In [50], the author found that the ℓ_2 regularization with small degree helps to generate more human-interpretable patterns. In the Section 3.3.3, we compare the pattern image synthesized by the AM and DGN-AM.

3.3. **Experiments with activation maximization.** In this section, the experiments of AM on *CaffeNet* trained with *ImageNet* dataset are demonstrated, which shows what features have been learned by each neuron. The first layer neurons can be visualized by directly mapping each filter matrix to an RGB pattern image, hence, the first layer visualization by AM and the direct mapping method are evaluated to show the effectiveness of AM. Then the hidden layer visualization shows the abundant and hierarchical features learned by different layer neurons. Finally the final layer visualization by AM and DGN-AM are compared to emphasize the improvement of interpretability by the DGN-AM.

3.3.1. *First layer visualization.* As aforementioned, the AM has more distinguishable performance on the early network layers, hence, we first evaluate the first layer visualization with AM.

Fig. 4 (a) shows several visualization results of synthesizing the patterns for four different neurons. This process uses the gradient $\frac{\partial a_{i,l}}{\partial x}$ to tweak the input noise iteratively, which increases the activation of neuron i. After 400 iterations, we

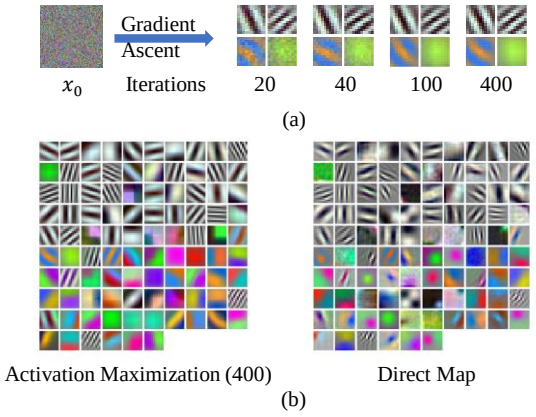

FIGURE 4. First layer of *CaffeNet* visualized by *Activation Maximization*

can get relatively smooth visualized patterns, which indicates a nicely converged network is trained.

Fig. 4 (b) shows the visualized patterns synthesized by the AM and direct mapping method. As we can see, most of the visualized patterns synthesized by the AM are almost the same as the corresponding direct mapped patterns. The visualized patterns are clustered into two groups: 1) the colorful patterns indicate the corresponding neurons greatly sensitive to color components in the under-test images; 2) the black-and-white patterns indicate the corresponding neurons greatly sensitive to shape information. In addition, through comparison with the direct map method, the AM can reveal the preferred inputs of each neuron accurately.

This interesting finding reveals that the CNNs attempt to imitate the human visual cortex system, which the neurons in the lower visual area are sensitive to basic patterns, such as colors, edges, and lines.

3.3.2. *Hidden layers visualization.* Beyond the first layer, the neurons in the following layers gradually learn to extract feature hierarchically.

Fig. 5 shows the visualization of 6 hidden layers from the second convolutional layer (CL 2) to the second fully connected layer (FL 2) in each row. Several neurons in each layer are randomly selected as our AM test targets. We observed that: 1) Some important patterns are visualized, such as edges (CL 2-4), faces (CL 4-1), wheels (CL 4-2), bottles (CL 5-1), eyes (CL 5-2), *etc.*, which demonstrate the abundant features learned by the neurons. 2) Meanwhile, not all the visualized patterns are interpretable even with multiple regularization methods are applied. 3) The complexity and variation of the visualized patterns are increasing from lower layers to higher layers, which indicates that increasingly invariant features are learned by the neurons. 4) From the CL 5 to FLs, we can find there is a large pattern variation increment, which could indicate the FLs provide a more comprehensive feature evaluation.

3.3.3. *Final layer visualization.* The final layer of the *CaffeNet* contains 1000 neurons, which corresponding to the 1000 classes in the *ImageNet* dataset. Five different neurons are selected to show the visualized pattern image.

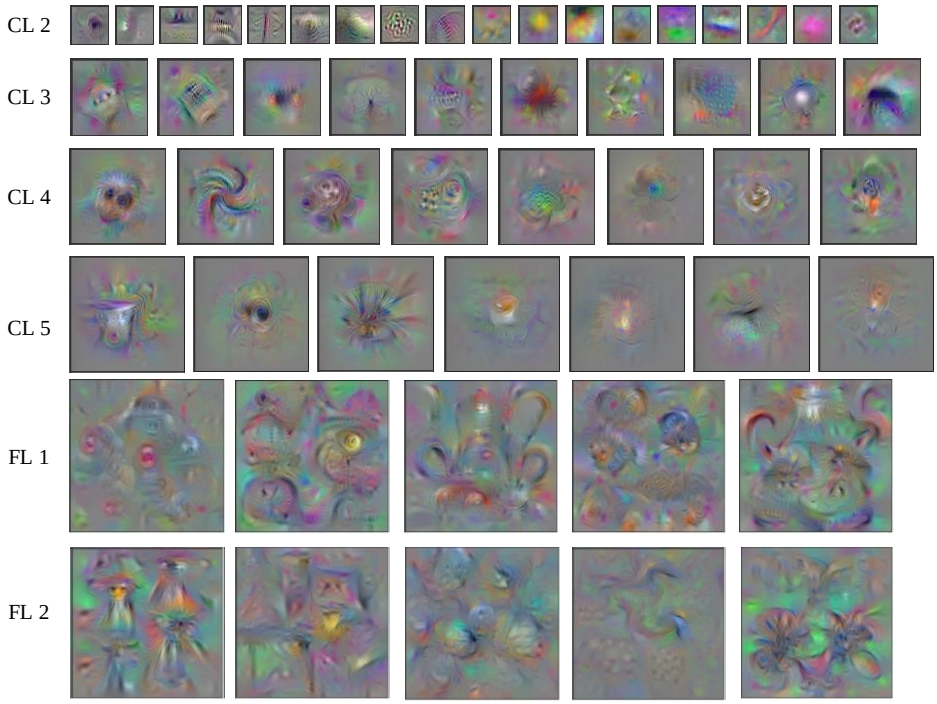

FIGURE 5. Hidden layers of *CaffeNet* visualization by *Activation Maximization*. Adapted from "Understanding Neural Networks Through Deep Visualization," by J. Yosinski, 2015.

Fig. 6 compares the visualized patterns synthesized by the AM and the DGN-AM for five different classes in FL 3. For the AM shown in the first row of Fig. 6, although we can guess which class the visualized patterns represents, there are multiple duplicate and vague objects in the each visualized pattern, such as three

FIGURE 6. Output layer of *CaffeNet* visualized by *Activation Maximization*.

lipsticks in the third column (AM-3) and the images are far from being photo-realistic. For the DGN-AM shown in the second row of Fig. 6, by utilizing the generator networks, the DGN-AM greatly improves the images quality in terms of color and texture. Because the fully connected layer contain information from all areas of the image and the generator network provides a strong biases toward realistic visualizations.

Through the final layer visualization, we can clearly see what objects combinations could affect the CNN classification decision. For example, if the CNN classifies an image of a cell phone held in a human hand as a cell phone, it is unclear if the classification decision is affected by the human hand. Through the visualization, we can see there is a cell phone and a human hand in the cell phone class, which are shown in the second row and third column. In this case, the visualization shows the CNN has learned to detect both object information in one image.

3.4. **The summary.** As the most intuitive visualization method, the AM reveals that CNNs learn to detect the important features such as faces, wheels, and bottles without our specification. At the same time, CNNs attempt to mimic the hierarchical organization of the visual cortex, and then successfully build up the hierarchical feature extraction. In addition, this visualization method suggests that the individual neurons extract features in a more local manner rather than distributed, which each neuron correspond to a specific pattern.

4. Visualization by deconvolutional network.

Find the selective pattern from a given input image that activate a specific neuron in the convolutional layers.

4.1. **The overview.** While the *Activation Maximization* interprets the CNNs from the perspective of the neurons, the *Deconvolutional Network* (*DeconvNet*) based CNN visualization explains the CNNs from the perspective of the input image. It finds the selective patterns from the input image that activate a specific neuron in the convolutional layers. The patterns are reconstructed by projecting the low-dimension neurons' feature maps back to the image dimension. This projection process is implemented by a *DeconvNet* structure, which contains deconvolutional layers and unpooling layers, performing the inversed computation of the convolutional and pooling layers. Rather than purely analyzing the neurons' interests, the *DeconvNet* based visualization demonstrates a straightforward feature analysis in an image level.

The research related to the *DeconvNet* structure is mainly led by Zeiler *et al.* In [73], they first proposed the *DeconvNet* structure aiming to capture certain general features for reconstructing the natural image by projecting a highly diverse set of low-dimension feature maps to high dimension. Later in [74], they utilized the *DeconvNet* structure to decompose an image hierarchically, which could capture the image information at all scales, from low-level edges to high-level object parts. Eventually, they applied the *DeconvNet* structure for CNN visualization by interpreting CNN hidden features [72], which made it become an effective method to visualize the CNNs.

4.2. **The algorithm.** The *DeconvNet* is an effective method to visualize the CNNs, we will explain the *DeconvNet* based visualization in terms of *DeconvNet* structure and the visualization process in this section.

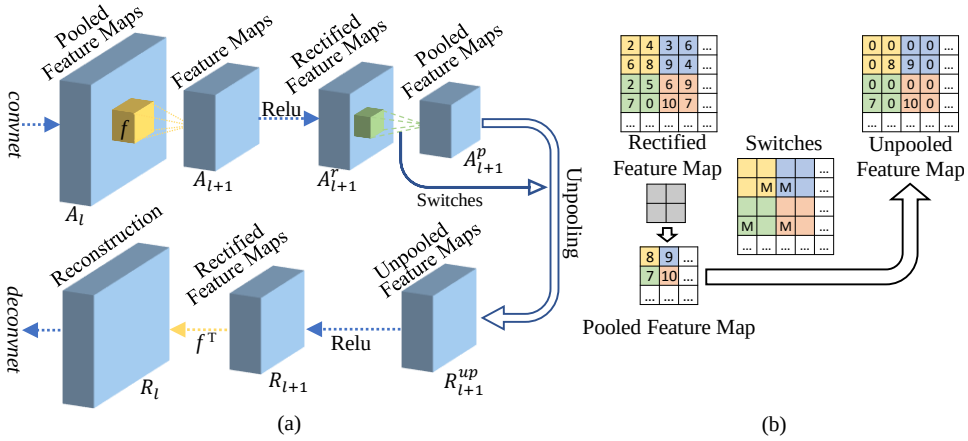

FIGURE 7. The structure of the *Deconvolutional Network*

4.2.1. *DeconvNet structure.* The *DeconvNet* provides a continuous path, which projects the low-dimension pooled feature map back to the image dimension. Typically, there are reversed convolutional layers (namely the deconvolutional layer), reversed rectification layers, and reversed max-pooling layers (namely unpooling layer) in the *DeconvNet* structure. A typical *DeconvNet* structure is shown in Fig. 7. In Fig. 7(a), the *DeconvNet* serves as a reversed process of CNN, which is composed of the reversed layers corresponding to the layers in CNNs.

Each layer of the *DeconvNet* is defined as follows:

Reversed Convolution/Deconvolutional Layer: To explain the deconvolutional layer, we first take a look at the convolutional layers as shown in the top Fig. 7 (a). The convolutional layer transforms the input feature maps into the output feature maps described in Eq. 1: $a_{i,l+1} = F(\sum w_{i,l}a_{i,l} + b_{i,l})$, where the w, b is the filter parameters and F is the *relu* activation function. We combine the convolutional and summing operations of layer l into a single filter matrix f_l and convert the multiple feature maps $a_{i,l}$ into a single feature vector A_l:

$$A_{l+1} = A_l * f_l. \tag{11}$$

After appling the *relu* function, the rectified feature maps A_l^r is produced.

While in the deconvolutional operation, the reversed convolutional layer, namely the deconvolutional layer, uses the transposed versions of the same convolutional filters to perform the convolutional operations. The deconvolution process can be viewed as:

$$R_l = R_{l+1} * f_l^T. \tag{12}$$

The f_l^T is the transposed versions of the convolutional filters, which is flipped from the filters f horizontally and vertically. The R_l indicates the rectified feature maps in the *DeconvNet*, which is convolved with the f_l^T.

Reversed Rectification Layer: The CNNs usually use the *relu* activation function, which rectifies the feature maps thus ensuring the feature maps are always positive. The feature maps of deconvolutional layer are also ensured to be positive in reconstruction by passing the unpooled feature maps R^{up} through a *relu* function.

Reversed Max-pooling/Unpooling Layer: The reversed max-pooling process in a *DeconvNet* is implemented by the unpooling layer. Fig. 7 (b) shows the unpooling

process in detail: In order to implement the reversed operation of max-pooling, which performs the downsampling operation on the rectified feature maps A^r_{l+1}, the unpooling layer transform the pooled feature maps to the unpooled feature maps.

During the max-pooling operation, the positions of maximal values within each pooling window are recorded in switch variables. The switches first specify the position of which elements in the rectified feature map are copied into the pooled feature map, then mark them as M in the switches. These switches variables are used in the unpooling operation to place each maximal value back to its original pooled location. Due to the dimension gap, certain amount of locations are inevitable constructed without certain information, therefore these locations are usually filled by zero for compensation.

4.2.2. *Visualization process.* Based on the reversed structure formed by those layers, the *DeconvNet* can be well utilized to visualize the CNNs. The visualization process can be described as follows:

(1) All neurons' feature maps can be captured when a specific input image is processed through the CNN.

(2) The feature map of the target neuron for visualization is selected while all other neurons' feature maps are set to zeros.

(3) In order to obtain the visualized pattern, the target neuron's feature map is projected back to the image dimension through the *DeconvNet*.

(4) To visualize all the neurons, this process is applied to all neurons repeatedly and obtain a set of corresponding pattern images for CNN visualization.

These visualized patterns indicate which pixels or features in the input image contribute to the activation of the neuron, and it also can be used to examine the CNN design shortcomings. In the next section, these visualized patterns will be demonstrated with practical experiements.

4.3. **Experiments with *DeconvNet* based visualization.** In this section, the experiments of *DeconvNet* based visualization on *CaffeNet* trained with *ImageNet* dataset are demonstrated, to show what features have been learned by each neuron. In addition, this method can be used as an efficient tools for network analysis, optimization and training monitoring.

4.3.1. *Convolutional layer visualization.* As aforementioned, the *DeconvNet* structure provides a continuous path back to the image dimension, which can highlight specific pattern in the input image that excites a neuron.

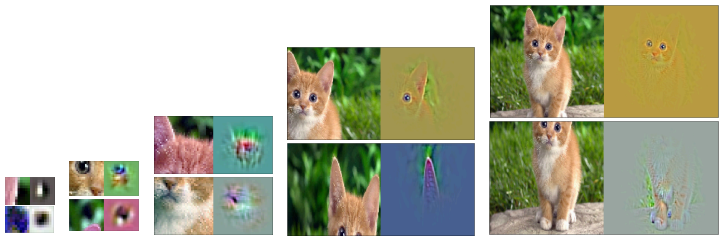

FIGURE 8. *CaffeNet* visualized by *DeconvNet*

Fig. 8 shows the *DeconvNet* based visualization examples with 5 convolutional layers of the *CaffeNet* from CL1 to CL5. In each layer, we randomly select two neurons' visualized patterns comparing to the corresponding local sections in the original image. From these examples, we can tell that: Each individual neuron extracts feature in a more local manner, where different neurons in one layer are responsible for different patterns, such as mouth, eyes, and ears.

Lower layers (CL1, CL 2) capture the small edges, corners, and parts. CL3 has more complex invariance, capturing similar textures such as mesh patterns. Higher layers (CL4, CL5) are more class-specific, which show the almost entire objects. Compared to the *Activation Maximization*, the *DeconvNet* based visualization can provide much more explicit and straightforward patterns.

4.3.2. *DeconvNet based visualization for network analysis and optimization.* Beside the convolutional layer visualization for interpretation analysis, *DeconvNet* can be also used to examine the CNN design for further optimization.

Fig. 9 (a) and (c) show the visualization of the first and second layers from *AlexNet*. We can find that: 1) There are some "dead" neurons without any specific patterns (indicated in pure gray color) in the first layer, which means they have no activation for the inputs. This could be a symptom of high learning rates or not good weights initialization. 2) The second layer visualization shows aliasing artifacts, highlighting by the red rectangles. This could be caused by the large stride used in the first-layer convolutions.

These findings from the visualization can be well applied to the CNN optimization. Hence, Zeiler *et al.* proposed *ZFNet*, which reduced the first layer filter size and shrink the convolutional stride of *AlexNet* to retain much more features in the first two convolutional layers.

The improvement introduced by *ZFNet* is demonstrated in Fig. 9 (b) and (d), which shows the visualizations of the first and second layers of *ZFNet*. We can see that the patterns in the first layer become more distinctive, and the patterns in the second layer have no aliasing artifacts. Hence, the visualization can be effectively applied in CNN analysis and further optimization.

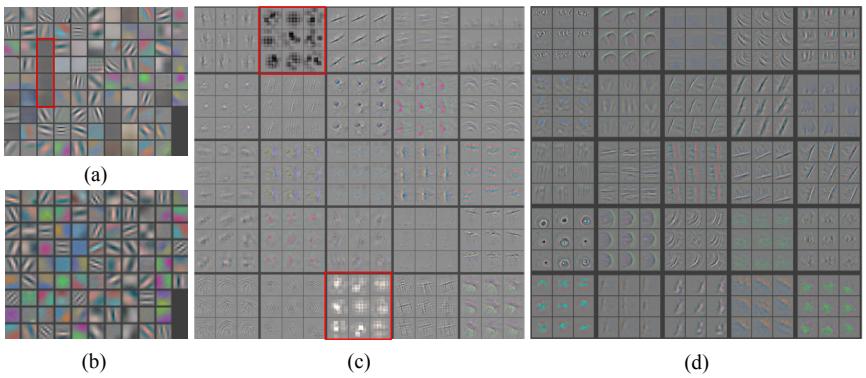

FIGURE 9. First and second layer visualization of *AlexNet* and *ZFNet*. Adapted from "Visualizing and Understanding Convolutional Networks," by M.D. Zeiler, 2014.

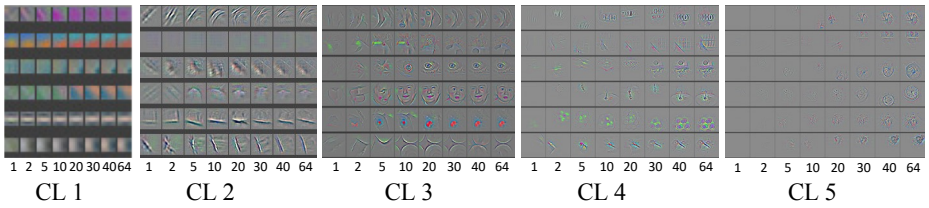

FIGURE 10. Feature evolution during training *ZFNet* . Adapted from "Visualizing and Understanding Convolutional Networks," by M.D. Zeiler, 2014.

4.3.3. *DeconvNet based visualization for training monitoring.* Besides the CNN network optimization, the interpretability analysis can also help to monitor the CNN training process for better training efficiency.

Fig. 10 shows the visualized pattens during the training the *ZFNet*. Each row indicates different neurons in the convolutional layers. A randomly chosen subset of visualized patterns at different training epoch are shown in each column. We can find that: 1) In each row, the color contrast is artificially enhanced as the training process. 2) The lower layers (CL1, CL2) converge quickly, since distinguished patterns appear within a few epochs. 3) However, the distinguished patterns appear after a considerable number of epochs in the upper layers (CL4, CL5), which means these layers need to be trained until fully converged.

Additionally, if noisy patterns are observed in the training process, that could indicate that the network hasn't been trained long enough, or low regularization strength that may result in overfitting. By visualizing features at several time points during training, we can find the design shortcomings and adjust the network parameters in time. In general, the visualization of training process is an effective way to monitor and evaluate the training statues.

4.4. **The summary.** The *DeconvNet* highlights which selected patterns in the input image contribute to the activation of a neuron in a more interpretable manner. Additionally, this method can be used to examine the problems with the CNNs for optimization. And the training monitoring could provide the CNN research with a better criteria when adjusting the training configuration and stopping training.

However, both methods of AM and *DeconvNet* visualize the CNN in the neuron level, lacking a comprehensive perspective from higher structure, such as layer and whole network. In the following sections, we will further discuss high-level CNN visualization methods, which interpret each individual layer and visualize the information captured by the set of neurons in a layer as a whole.

5. **Visualization by network inversion.**

Reconstruct an image from all the neurons' feature maps in an arbitrary layer to highlight the comprehensive CNN layer-level feature for a given input image.

5.1. **The overview.** Different from the activation from a single network neuron, the layer-level activation will reveal a comprehensive feature representation, which is composed of the all neuron activation patterns inside a layer. Hence, different form the aforementioned visualization methods, which visualize the CNN from a

single neuron's activation, the *Network Inversion* based visualization can be used to analysis the activation information from a layer level perspective.

Before the *Network Inversion* is applied to visualize the CNNs, the fundamental idea of *Network Inversion* was proposed to study the traditional computer vision representation, such as the Histogram of Oriented Gradients (HOG) [7, 15, 19], the Scale Invariant Feature Transform (SIFT) [45], the Local Binary Descriptors (LBD) [8], and the Bag of Visual Words Descriptors [6, 64]. Later, two variants of the *Network Inversion* were proposed for CNN visualization [46, 47, 11]:

(1) Regularizer based *Network Inversion*: It is proposed by Mahendran *et al.*, which reconstructs the image from each layer by using gradient descent approach and a regularization term [46, 47].

(2) *UpconvNet* based *Network Inversion*: It is proposed by Dosovitskiy *et al.* [11, 12], which reconstructs the image by training a dedicated *Up-convolutional Neural Network* (*UpconvNet*).

Overall, the main goal of both algorithms is to reconstruct the original input image from one whole layer's feature maps' specific activation. The Regularizer based *Network Inversion* is easier to be implemented, since it does not require to train an extra dedicated network. While the *UpconvNet* based *Network Inversion* can visualize more existent information in higher layers with an extra dedicated network and significantly more computational cost.

5.2. **The algorithm.** In this section, we compared the aforementioned two *Network Inversion based Visualization* methods regarding the network structure and the learning algorithm.

Fig. 11 shows the network implementation for the two *Network Inversion based Visualization* methods comparing with the original CNN: the Regularizer based *Network Inversion* is shown in the upper as denoted in green, and the *UpconvNet* based *Network Inversion* is shown in the bottom as denoted in orange, while the original CNN is shown in the middle as denoted in blue.

The Regularizer based *Network Inversion* has the same architecture and parameters as the original CNN before the visualization target layer. In this case, each pixel of the to be reconstructed image x_0 is adjusted to minimize the objective loss function error between the target feature map $A(x_0)$ of x_0 and the feature map $A(x)$ of the original input image x.

For the *UpconvNet* based *Network Inversion*, the *UpconvNet* provides a inverse path for the feature map back to the image dimension. The parameters of the *UpconvNet* are adjusted to minimize the objective loss function error between the reconstructed image x_0 and the original input image x.

In the following sections, we will give detailed explanations of the two methods mathematically.

5.2.1. *Regularizer based network inversion.* The fundamental algorithm of Regularizer based *Network Inversion* can be viewed as reconstructing an image x^* which minimizes the objective function as following:

$$x^* = \underset{x}{\operatorname{argmin}}(C \cdot \mathcal{L}(A(x), A(x_0)) - \lambda(x)), \tag{13}$$

where the loss function $\mathcal{L}$ computes the difference between the two aforementioned feature maps $A(x_0)$ and $A(x)$. The constant C trades off the loss and the regularizer, and the regularizer $\lambda(x)$ restricts the reconstructed image to a natural image. The

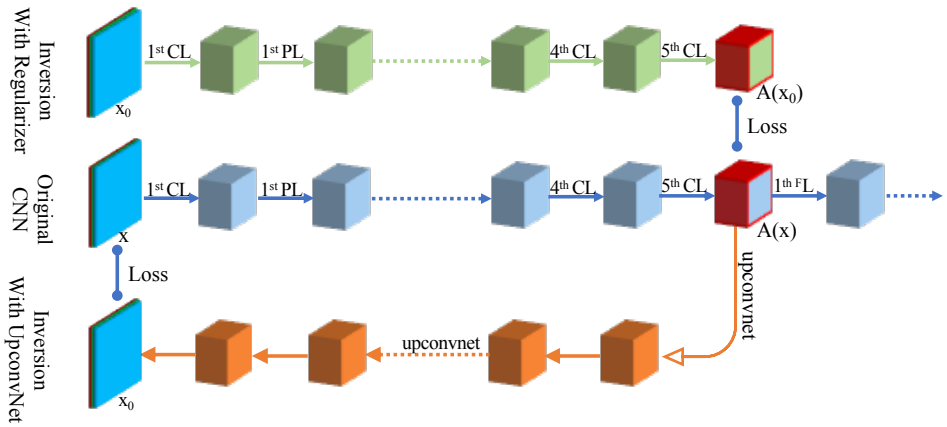

FIGURE 11. The data flow of the two *Network Inversion* algorithms

loss function is usually defined as a Euclidean distance:

$$\mathcal{L}(A(x), A(x_0)) = \|A(x) - A(x_0)\|^2, \tag{14}$$

which is the most commonly used measurement to evaluate the similarity between different images [69].

In order to make the reconstructed images look closer to the nature images, multiple regularization approaches have been experimentally studied to improve the reconstruction quality, such as α-norm, total variation norm (TV) , jittering, and texture or style regularizers [59, 49, 18]. As an example, for a discrete image data $x \subset R^{HXW}$, the TV norm is given by:

$$\lambda(x) = \sum_{i,j}((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)^{\frac{\beta}{2}}, \tag{15}$$

where the regularizer β =1 stands for the standard TV norm that is mostly used in image denoising. In this case, the TV norm penalizes the reconstructed images to encourage the spatial smoothness.

Based on such a *Network Inversion* framework, the visualization process can be divided into five steps:

(1) The visualization target layer's feature maps $A(x)$ of the original input image x and the feature maps $A(x_0)$ of the to be reconstructed x_0 (initialized with noise) are firstly computed.

(2) The error between the two feature map sets – $\mathcal{L}(A(x), A(x_0))$ is then computed by the objective loss function.

(3) Guided by the direction of the gradient $\frac{\mathcal{L}(A(x),A(x_0))}{\partial x}$, each pixel of the noise image is changed iteratively to minimize the objective loss function error.

(4) This process terminates at a specific reconstructed image x^*, which is used to demonstrate what information is preserve in the visualization target layer.

The Regularizer based *Network Inversion* iteratively tweaks the input noise towards the direction that minimizes the difference between the two feature map sets, while the *UpconvNet* based *Network Inversion* minimizes the image reconstruction error. In the next section, we will then discuss the *UpconvNet* based *Network Inversion* in detail.

5.2.2. *UpconvNet based network inversion.* Although the Regularizer based *Network Inversion* can reconstruct a image for CNN layer visualization, it still suffer from relatively slow computation speed due to gradient computation. To overcome this shortcoming, Dosovitskiy *et al.* proposed another *Network Inversion* approach, which trained an extra dedicated *Up-convolutional Neural Network* (*UpconvNet*) to reconstruct the image with better image quality and computation efficiency [11].

The *UpconvNet* can project the low-dimension feature maps back to the image dimension with similar reversed layers as in *DeconvNet*. As shown in the bottom part of Fig. 11, the *UpconvNet* takes the feature maps $A(x)$ as the input, and yields the reconstructed image as the output.

Each layer of the *UpconvNet* are described as follows:

Reversed Convolutional Layer: The filters are re-trained in the *UpconvNet* whereas the *DeconvNet* uses the transposed versions of the same convolutional filters. Given a training set of images and their feature maps $(x_i, A(x_i))$, the training procedure can be viewed as:

$$W^* = \underset{w}{\operatorname{argmin}} \sum_i ||x_i - D(A(x_i), W)||^2, \tag{16}$$

where the weight W of *UpconvNet* is optimized to minimize the squared Euclidean distance between the input image x_i and the output of *UpconvNet* – $D(A(x_i), W)$.

Reversed Rectification Layer: The feature maps of *UpconvNet* are also ensured to be positive, with the leaky *relu* nonlinearity of slope 0.2 is applied:

$$A(x) = \begin{cases} x & x \geqslant 0 \\ 0.2 & x < 0 \end{cases} \tag{17}$$

Reversed Max-pooling Layer: The unpooling layers in *UpconvNet* are quite simplified. The feature maps are upsampled by a factor of 2, which replaces each value by a 2×2 block with the original value in the top left corner and all other entries equal to zero.

After the training process, we can utilize this *UpconvNet* to reconstruct any input image without computing the gradients. Therefore, it dramatically decreases the computational cost and can be applied to various kinds of deep networks. In the next section, we will evaluate the visualization results based on these two approaches.

5.3. **Experiments with the network inversion based visualization.** In this section, the experiments of *Network Inversion based Visualization* is demonstrated based on *AlexNet* trained with *ImageNet* dataset. The experiments demonstrate that *Network Inversion based Visualization* can not only achieve optimal visualization performance, but can be also utilized enhance the CNN design.

5.3.1. *Layer-level visualization analysis.* Visualization from layer level can reveal what features are preserved by each layer. Fig. 12, shows Regularizer and *UpconvNet* based visualization from various layers of *AlexNet*.

From Fig. 12, we can find that: 1) The visualization from the CLs look similar to the original image, although with increasing fuzziness. This indicates that the lower layers preserve much more detailed information, such as colors and locations of objects. 2) The visualization quality has an obvious drop from the CLs to FLs. However, the visualization from higher CLs and even FLs preserve color (*UpconvNet*) and the approximate object location information. 3) The *UpconvNet* based visualization quality is better than the Regularizer based visualization, especially

for the FLs. 4) The unrelated information is gradually filtered from low layers to high layers.

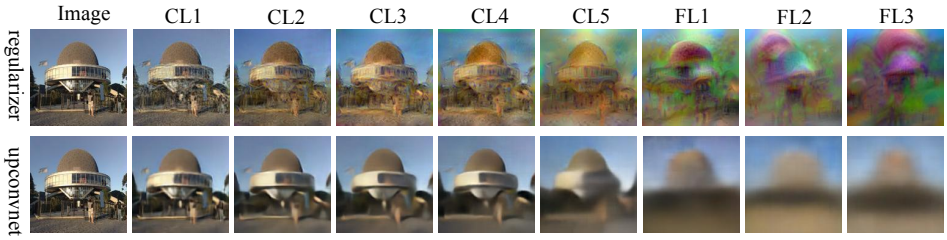

FIGURE 12. *AlexNet* reconstruction by *Network Inversion* with regularizer and *UpconvNet*. Adapted from "Inverting Visual Representations with Convolutional Networks," by A. Dosovitskiy, 2016.

5.3.2. *Layer level feature map analysis.* Based on the layer level analysis, here we further utilize the *Network Inversion based Visualization* to analysis the feature map characteristics. In [11], Dosovitskiy *et al.* perturbed part of the feature maps in one layer and visualized the reconstructed image from these perturbed features maps. The perturbation was performed in two ways:

(1) Binarization: the signs of all feature maps' values are kept, and their absolute values are set to be fixed numbers. The Euclidean norm of the values are remained unchanged.

(2) Dropout: 50% of the feature maps' values are set to be zeros and then normalized to keep their Euclidean norm unchanged.

Fig. 13 shows the reconstructed images under the two perturbation approaches in different layers. From Fig. 13 we can see that: 1) In FL1, the binarization hardly changes the reconstruction quality, which means almost all information about the input image is contained in the pattern of non-zero feature maps. 2) The Dropout changes the reconstructed images a lot. However, Dosovitskiy *et al.* also

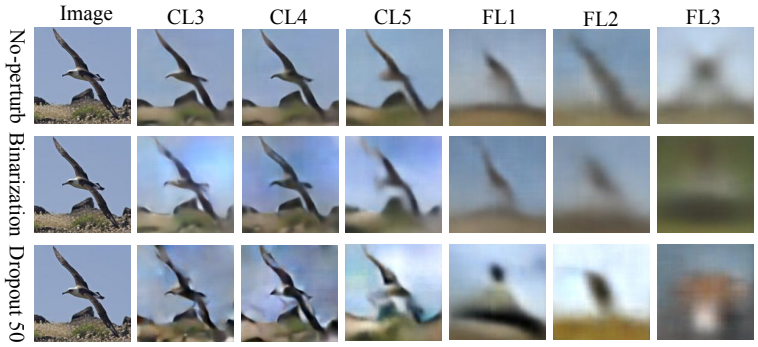

FIGURE 13. *AlexNet* reconstruction by perturbing the feature maps. Adapted from "Inverting Visual Representations with Convolutional Networks," by A. Dosovitskiy, 2016.

experimentally showed that by Dropouting the 50% least important feature maps could significantly reduce the reconstruction error, which is even better than not applying any Dropout for most layers.

These observations could be a proof that various CNN compression techniques could achieve optimal performance, such as quantization and filter pruning, due to the considerable amount of redundant information in each layer. Hence the *Network Inversion* based Visualization can be used to evaluate the importance of feature maps, and pruning the least important feature maps for network compression.

5.4. **The summary.** The *Network Inversion based Visualization* projects a specific layer's feature maps back to the image dimension, which provides insights into what features a specific layer would preserve. Additionally, by perturbing some feature maps for visualization, we can verify the CNN preserved a lot redundant information in each layer, and therefore further optimize the CNN design.

6. Visualization by network dissection.

Evaluate the correlation between each convolutional neuron or multiple neurons with a specific semantic concept.

6.1. **The overview.** In previous sections, multiple visualization methods were demonstrated to reveal the visual perceptible patterns that a single neuron or layer could capture. However, there is still a missing link between the visual perceptible patterns and the clear interpretable semantic concepts.

Hence, Bau *et al.* proposed the *Network Dissection*, which directly associates each convolutional neuron with a specific semantic concept, such as color, textures, materials, parts, objects, and scenes. The correlation between the neuron and the semantic concept is measured by seeking for the neuron that strongly responses to particular image content with specific semantic concepts. A heterogeneous image dataset — *Borden*, provides the images with specific semantic concepts labeled corresponding to local content. A set of *Broden* examples are shown in Fig. 14, in which the semantic concepts are divided into six categories highlighted with red boxes. Each semantic category may cover various classes, for example, the object category contains plant, train, *etc.* At the lower right corner of each example in Fig. 14, the semantic corresponding neuron is also identified. We can also see that black masks are introduced to cover the image content that is not related to the assigned semantics. Here, it is the proposed *Network Dissection* that generates these black masks.

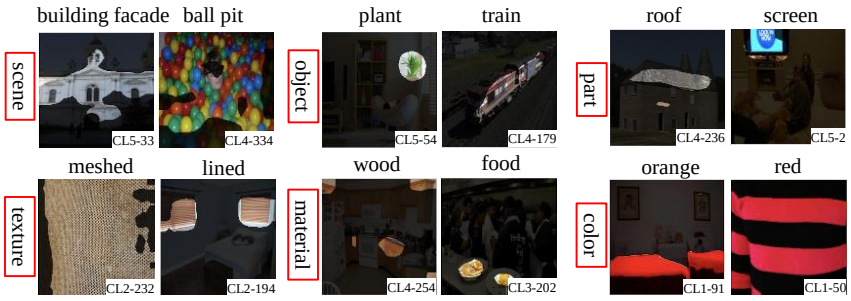

FIGURE 14. The *Broden* images that activate certain neurons in *AlexNet*

The development of the *Network Inversion* progressively connects the semantic concepts to different component levels in a CNN. The fundamental algorithm of *Network Inversion* illustrated the correlation between one semantic concept and one individual neurons. Such a correlation was based on an assumption that each semantic concept can be assigned to a single neuron [23]. Later, further *Network Inversion* works revealed that the feature representation can be distributed, which indicated that one semantic concept could be represented by multiple neurons' combination [1, 75]. Hence, following [3]'s paradigm, Fong *et al.* proposed another *Network Inversion* approach, namely *Net2Vec*, which visualized the semantic concepts based on neuron combinations [16].

Both methods provide comprehensive visualization results on interpreting CNN hidden neurons.

6.2. **The algorithm.** In this section, we introduce two *Network Dissection* methods, one method assigns the semantic concept to each individual neuron, while the other builds the correlation between the neuron combinations and the semantic concepts.

6.2.1. *Network dissection for the individual neuron.* The algorithm of *Network Dissection* for the individual neuron evaluates the correlation between each single neuron and the semantic concept. Specifically, every individual neuron is evaluated as a segmentation task to every semantic concept.

The evaluating process is shown in Fig. 15. The input image fetched from the *Broden* dataset contains pixel-wise annotations for the semantic concepts, which provides the ground truth segmentation masks. The target neuron's feature map is upsampled to the resolution of the ground truth segmentation masks. Then the *Network Dissection* works by measuring the alignment between the upsampled neuron activation map and the ground truth segmentation masks. If the measurement result is larger than a threshold, the neuron can be viewed as a visual detector for specific semantic concept.

This process can be described as follows:

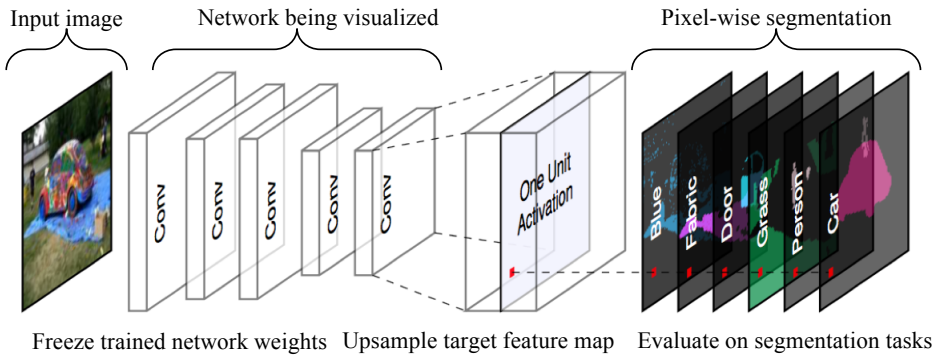

FIGURE 15. Illustration of network dissection for measuring semantic alignment of neuron in a given CNN. Adapted from "Network Dissection: Quantifying Interpretability of Deep Visual Representations," by D. Bau, 2017.

(1) The feature map $A_f(x)$ of every neuron f is computed by feeding in every input image x from *Broden*. So, the distributions of the activation scores $p(a_k)$ of neuron activation over all images in *Broden* are computed.

(2) The top activation maps from all feature maps are selected as valid map regions corresponding to neuron's semantics by setting an activation threshold that $P(a_f > T_f) = 0.005$.

(3) To match the low-resolution valid map to the ground truth segmentation mask L_c for some semantic concept c, the valid map is upsampled:

$$M_f(x) = S(A_f(x) > T_f), \tag{18}$$

where S denotes a bilinear interpolation function.

(4) The accuracy of neuron f in detecting semantic concept c is determined by the intersection-over-union (IoU) score:

$$IoU_{f,c} = \frac{\sum |M_f(x) \bigcap L_c(x)|}{\sum |M_f(x) \bigcup L_c(x)|}, \tag{19}$$

where $L_c(x)$ denotes the ground-truth mask of the semantic concept c on the image x. If $IoU_{f,c}$ is larger than a threshold (0.04), we consider the neuron f as a visual detector for concept c. The IoU score indicates the accuracy of neuron f in detecting concept c. Finally, every neuron's corresponding semantic concept can be determined by calculating its IoU score.

Hence, the *Network Dissection* for the individual neuron can automatically assign a semantic concept to each convolutional neuron.

6.2.2. *Network dissection for the neuron combinations.* Instead of interpreting the individual neuron, Fong *et al.* [16] proposed *Net2Vec* to evaluate the correlation between the neuron combinations and the semantic concepts. They implemented this approach as a segmentation task by using convolutional neuron combinations. Specifically, a learnable concept weight w is used to linearly combine the threshold based activation. And, it is passed through the sigmoid function $\sigma(x) = 1/(1 + exp(-x))$ to predict a segmentation mask $M(x; w)$:

$$M(x; w) = \sigma(\sum_k w_k \cdot \mathbb{I}(A_f(x) > T_f)), \tag{20}$$

where k is the number of neurons in a layer, and $\mathbb{I}(\cdot)$ is the indicator function. This function selects a subset of neurons in one layer whose activation is larger than the threshold. Hence, this subset of neurons can be used to generate the activation mask for specific semantic.

Fong *et al* experimentally found that, for the segmentation task, materials and parts reached near optimal performance around $k = 8$, which was much more quickly than that of objects $k = 16$. For each concept c, the weights w were learned using stochastic gradient descent with momentum to minimize per-pixel binary cross entropy loss.

Similar to the single neurons case, the IoU_{com} score for neuron combinations is computed as well:

$$IoU_{com} = \frac{\sum |M_{f,w}(x) \bigcap L_c(x)|}{\sum |M_{f,w}(x) \bigcup L_c(x)|}. \tag{21}$$

If the IoU_{com} score is larger than a threshold, we consider this neuron combinations as a visual detector for concept c.

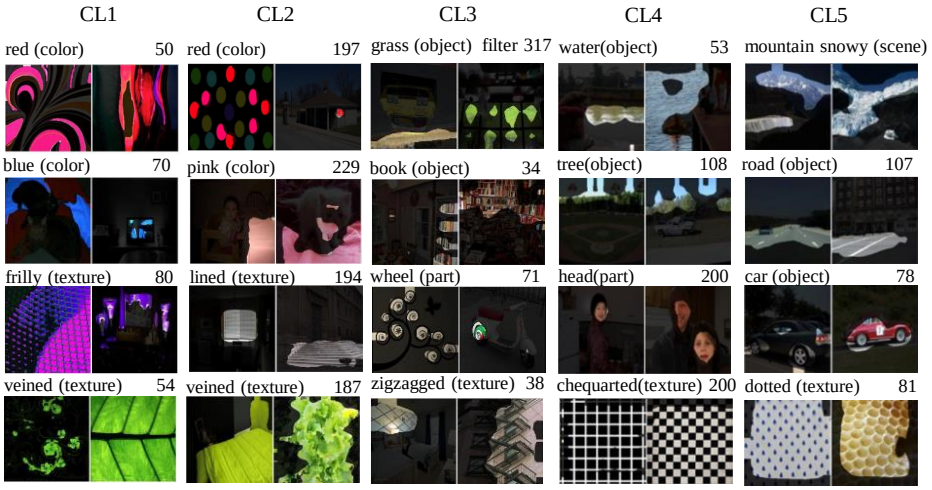

FIGURE 16. *AlexNet* visualization by *Network Dissection*

In fact, using the learned weights to combine neurons outperforms using a single neuron on the segmentation tasks. As a result, the generated segmentation masks demonstrate more complete and obvious objects in the image. In the next section, we demonstrate the visualization results by these two approaches.

6.3. **Experiments with the network dissection based visualization.** The *Network Dissection* can be applied to any CNN using a forward pass without the need for training or computing the gradients. In this section, we demonstrate the *Network Dissection based Visualization* results based on *AlexNet* trained with *ImageNet*.

6.3.1. *Network dissection for the individual neuron.* The visualization results of the individual neuron is demonstrated in the Fig. 16. In each column, four individual neurons along with two *Broden* images are shown in each CL. For each neuron, the top left shows the predicted semantic concepts, and the top right shows the neuron number. As mentioned, if the IoU_{com} score is larger than a threshold, we consider this neuron as a visual detector for concept c. Each layer's visual detector number is summarized in the left part of Fig. 17, which counts the number of unique concepts matched with neurons.

From the figures, we can find that: 1) Every image highlights the regions that cause the high neural activation from a real image. 2) The predicted labels match the highlighted regions pretty well. 3) From the number of the detector summary, the color concept dominates at lower layers (CL 1 and CL 2), while more object and texture detectors emerge in CL 5.

Compared with previous visualization methods, we conclude that the CNNs could detect the basic information, such as color and texture by all layer neurons rather than lower layer neurons. And the color information can be preserved even in higher layers, since many color detectors are also found in these layers.

6.3.2. *Interpretability under different training conditions.* The training conditions, such as the number of training iterations could also affect the representation learning of the CNNs. In [3], Bau *et al.* evaluated the effects on the interpretability of various state-of-the-art CNNs by using different training conditions, such as Dropout, batch

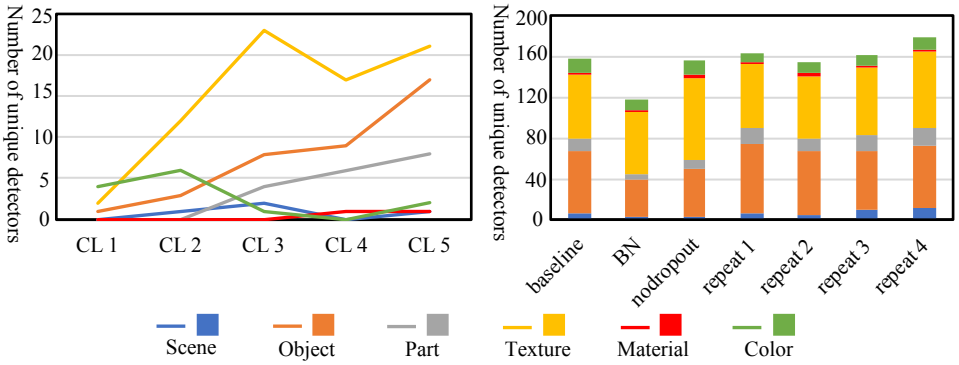

FIGURE 17. Semantic concept emerging in each layers and under different training conditions

normalization [32] and random initialization. As shown in the right part of Fig. 17, the NoDropout indicates the Dropout in the FC layers of the baseline model – *AlexNet* is removed. The BN indicates the batch normalization are applied at each CL. While, repeat1, repeat2 and repeat3 indicate randomly initialize the weights with the number of training iterations.

From Fig. 17, we can observe that: 1) The network shows similar interpretability under different initialization configurations. 2) For the network without Dropout applied, more texture detectors emerge, but fewer object detectors. 3) The batch normalization seems to decrease interpretability significantly. Overall, the Dropout

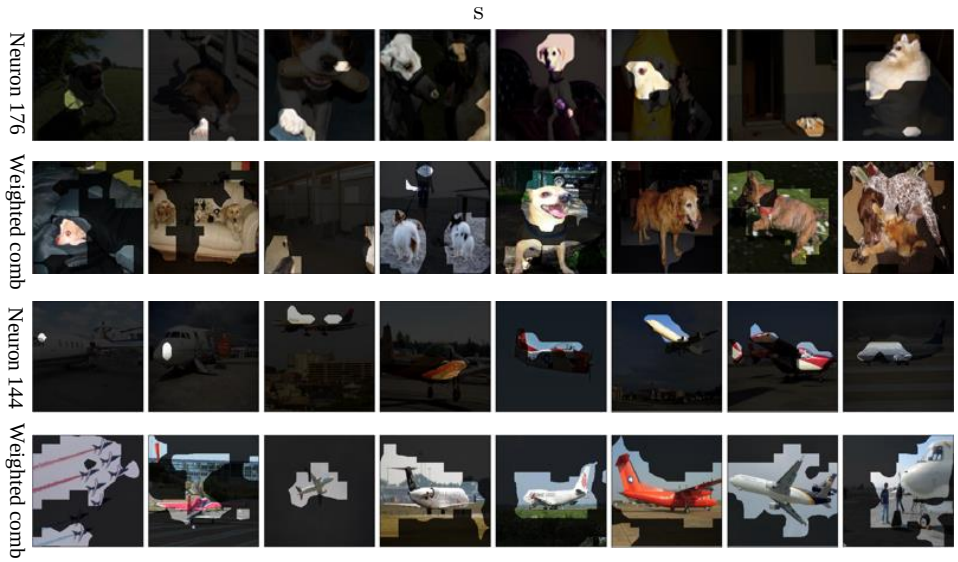

FIGURE 18. *Network Dissection* with single neuron and neuron combinations. Adapted from "Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks," by R. Fong, 2018.

and batch normalization can improve the classification accuracy. From the visualization perspective, the network tend to capture basic information without dropout. And, the batch normalization potentially decrease the feature diversity.

With such an evaluation, we can find that the *Network Dissection based Visualization* could effectively applied into evaluating different CNN optimization methods with a perspective of network interpretability.

6.3.3. *Network dissection for the neuron combinations.* The visualization results by using combined neurons are shown in Fig. 18. The first and third rows are the segmentation results by the individual neuron, while the second and fourth rows are segmented by neuron combinations. As we can see, for semantic visualization of "dog" and "airplane" using the weighted combination method, the predicted masks are informative and salient for most of the examples. This suggests that, although neurons that are specific to a concept can be found, these do not optimally represented or fully cover with the concept.

6.4. **The summary.** The *Network Dissection* is a distinguished visualization method to interpret the CNNs, which can automatically assign semantic concepts to internal neurons. By measuring the alignment between the unsampled neuron activation and the ground truth images with semantic labels, *Network Dissection* can visualize the types of semantic concepts represented by each convolutional neuron. The *Net2Vec* also verifies that the CNNs feature representation is distributed. Additionally, the *Network Dissection* can be utilized to evaluate various training conditions, which shows the training conditions can have a significant effect on the interpretability of the representation learned by hidden neurons. Hence, it is another representative example for CNN visualization and CNN optimization.

7. **CNN visualization application.** In this section, we review some practical applications of the CNN visualization. In fact, due to its ability to interpret the CNNs, CNN visualization has became an effective tools to reveal the differences between the way CNNs and humans recognize objects. We also applied the *Network Inversion* into an art generation algorithm called style transfer.

7.1. **Visualization analysis for CNN adversarial noises.** The CNNs has achieved impressive performance on a variety of computer vision related tasks. The CNNs are able to classify objects in images with even beyond human-level accuracy. However, we can still produce images with adversarial noises to attack the CNN for classification result manipulation, while the noises are completely unperceivable by the human vision recognition. Such adversarial noises could manipulate the results

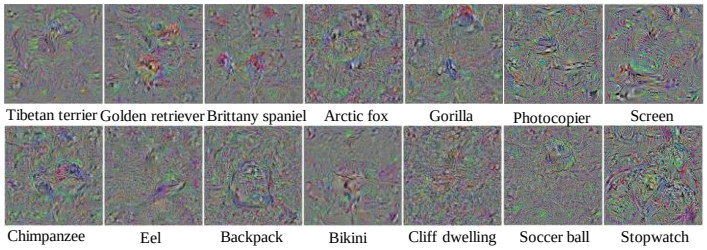

FIGURE 19. Adversarial noises that manipulate the CNN classification

of the state-of-the-art CNNs with a successful rate of 99.99% [51]. Hence, questions naturally arise as what differences remain between the CNNs and the human vision.

The *Activation Maximization* can be well utilized to examine those adversarial noises. As shown in the Fig. 19, the adversarial noises are generated by directly maximizing the final layer output for classes via gradient ascent. And continues until the CNNs confidence for the target class reaches 99.99%. Adding regularization makes images more recognizable, still far away from human interpretable images, but results have slightly lower confidence scores.

CNNs recognize these adversarial noises as near-perfect examples of recognizable images, which indicates that the differences between the way CNNs and humans recognize objects. Although CNNs are now being used for a variety of machine learning tasks. It is required to understand the generalization capabilities of CNNs, and find potential ways to make them robust. And the visualization is an optimal way to directly interpret those adversarial threat potentials.

7.2. **Visualization based adversarial examples.** In this section, we present another analysis approaching to demonstrate how the CNNs and the human vision differ. Recent studies show that the adversarial noises can be well embedded into images forming adversarial examples, which can manipulate the CNN classification results without noticing. Some adversarial examples are shown in Fig. 20.

To improve the robustness of the CNNs, the traditional techniques such as batch normalization and Dropout, generally do not provide a practical defense against adversarial examples. Some other strategies such as adversarial training and defensive distillation has been proposed to defense against the adversarial examples, which achieve state-of-art results. Yet even these specialized algorithms can easily be broken by giving more delicate adversarial examples.

The visualization can provide a solution to further uncover the mystery of adversarial examples. The activation maps of four convolutional neurons are shown in the Fig. 20. We can observe that the visualized feature maps have been changed a lot by the adversarial noises, especially for the CL2-97 and CL5-87. Hence, even the human vision can hard perceive the adversarial examples, the visualization could provide a significantly effective detection approach. The visualization analysis of the adversarial examples also reveal another major difference between CNN and the human vision: the imperceptible patterns can be captured by the CNNs and greatly affect the classification results.

7.3. **Visualization analysis for style transfer.** As we discussed in the Section 5, we can visualize the information each layer preserved from the input image, by reconstructing the image from feature maps in one layer. And we know the higher

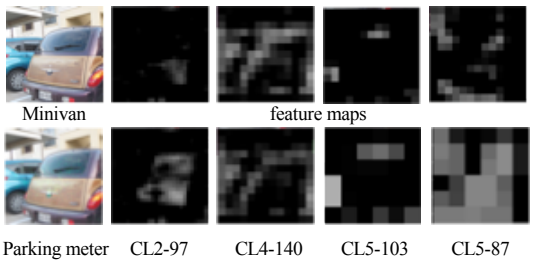

FIGURE 20. Adversarial example visualization

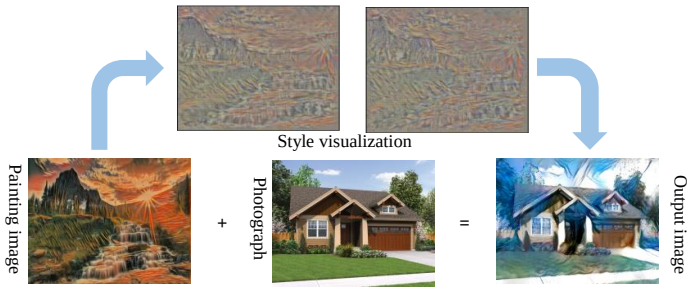

FIGURE 21. Style transfer example

layers capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the input image. These feature maps in higher layers can be referred as the *style* of an image [17].

As shown in Fig. 21, the style transfer generates the output image that combines the *style* of a painting image a with the content of a photograph p. We can utilize the *Network Inversion* visualize the style of the painting image. The process can be described viewed as jointly minimizing:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content}(A(p), A(x)) + \beta \mathcal{L}_{style}(A(a), A(x)), \tag{22}$$

where the $A(p)$ is the feature maps of the photograph in one layer and $A(a)$ is the feature maps painting image in multiple layers. The ratio $\frac{\alpha}{\beta}$ of content loss and style loss adjust the emphasis on matching the content of the photograph or the style of the painting image.

This process renders the photograph in the style of the painting image, which the appearance of the output image resembles the style of painting image, and the output image shows the same content as the photograph.

7.4. **Summary.** In this section, we briefed several applications of the CNN visualization beyond the scope of CNN interpretability enhancement and optimization. While more visualization applications still remained undiscussed. We do believe that the visualization could contribute to the CNN analysis in more and more perspectives.

8. **Conclusion.** In this paper, we have reviewed the latest developments of the CNN visualization methods. Four representative visualization methods are delicately presented, in terms of structure, algorithm, operation, and experiment, to cover the state-of-the-art research results of CNN interpretation.

Trough the study of the representative visualization methods, we can tell that: The CNNs do have hierarchical feature representation mechanism that imitates the hierarchical organization of the human visual cortex. Also, to reveal the CNN interpretation, the visualization works need to take various perspectives regarding different CNN components. Moreover, the better interpretability of the CNN introduced by visualization could practically contribute to the CNN optimization.

Hence, as the CNNs continually dominate the computer vision related tasks, the CNN visualization would play a more and more important role for better understanding and utilizing the CNNs.

Acknowledgments. This work was supported in part by NSF CNS-1717775.

REFERENCES

[1] P. Agrawal, R. Girshick and J. Malik, Analyzing the performance of multilayer neural networks for object recognition, in *Proceedings of the European Conference on Computer Vision*, 2014, 329–344.

[2] M. Arjovsky, S. Chintala and L. Bottou, Wasserstein gan, arXiv preprint, arXiv:1701.07875.

[3] D. Bau, B. Zhou, A. Khosla, A. Oliva and A. Torralba, Network dissection: Quantifying interpretability of deep visual representations, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, 3319–3327.

[4] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella and J. Schmidhuber, Flexible, High performance convolutional neural networks for image classification, in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 22, 2011, p1237.

[5] R. Collobert, K. Kavukcuoglu and C. Farabet, Torch7: A matlab-like environment for machine learning, in *Workshop on BigLearn, NIPS*, 2011.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski and C. Bray, Visual categorization with bags of keypoints, in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, 2004, 1–2.

[7] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, 886–893.

[8] E. d'Angelo, A. Alahi and P. Vandergheynst, Beyond bits: Reconstructing images from local binary descriptors, in *Proceedings of the IEEE Conference on Pattern Recognition,* 2012, 935–938.

[9] E. L. Denton, S. Chintala, R. Fergus et al., Deep generative image models using a Laplacian pyramid of adversarial networks, in *Proceedings of the Advances in Neural Information Processing Systems*, 2015, 1486–1494.

[10] A. Dosovitskiy and T. Brox, Generating images with perceptual similarity metrics based on deep networks, in *Proceedings of the Advances in Neural Information Processing Systems*, 2016, 658–666.

[11] A. Dosovitskiy and T. Brox, Inverting visual representations with convolutional networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 4829–4837.

[12] A. Dosovitskiy, J. Tobias Springenberg and T. Brox, Learning to generate chairs with convolutional neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, 1538–1546.

[13] J. Duchi, E. Hazan and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, **12** (2011), 2121–2159.

[14] D. Erhan, Y. Bengio, A. Courville and P. Vincent, Visualizing higher-layer features of a deep network, *Technical report, University of Montreal*, **1341** (2009), p3.

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, Object detection with discriminatively trained part-based models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32** (2010), 1627–1645.

[16] R. Fong and A. Vedaldi, Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks, arXiv preprint, arXiv:1801.03454.

[17] L. A. Gatys, A. S. Ecker and M. Bethge, A neural algorithm of artistic style, *Journal of Vision,* **16** (2016), p326, arXiv:1508.06576.

[18] L. A. Gatys, A. S. Ecker and M. Bethge, Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks, arXiv preprint, arXiv:1505.07376, **12**.

[19] R. B. Girshick, P. F. Felzenszwalb and D. McAllester, Discriminatively trained deformable part models, release 5, http://people.cs.uchicago.edu/~rbg/latent-release5/.

[20] R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, 580–587.

[21] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010, 249–256.

[22] Y. Gong, L. Wang, R. Guo and S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in *Proceedings of the European Conference on Computer Vision*, 2014, 392–407.

[23] A. Gonzalez-Garcia, D. Modolo and V. Ferrari, Do semantic parts emerge in convolutional neural networks?, *International Journal of Computer Vision*, **126** (2018), 476–494.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Generative adversarial nets, in *Proceedings of the Advances in Neural Information Processing Systems*, 2014, 2672–2680.
[25] A. Gordo, J. Almazán, J. Revaud and D. Larlus, Deep image retrieval: Learning global representations for image search, in *Proceedings of the European Conference on Computer Vision*, Springer, 2016, 241–257.
[26] S. Han, H. Mao and W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint, arXiv:1510.00149.
[27] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, 770–778.
[28] G. E. Hinton, S. Osindero and Y.-W. Teh, A fast learning algorithm for deep belief nets, *Neural computation*, **18** (2006), 1527–1554.
[29] D. H. Hubel and T. N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, *The Journal of Physiology*, **195** (1968), 215–243, URL http://dx.doi.org/10.1113/jphysiol.1968.sp008455.
[30] D. H. Hubel and T. N. Wiesel, Receptive fields of single neurones in the cat's striate cortex, *The Journal of physiology*, **148** (1959), 574–591.
[31] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the International Conference on Machine Learning*, 2015, 448–456.
[32] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the International Conference on Machine Learning*, 2015, 448–456.
[33] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in *Proceedings of the International Conference on Multimedia*, 2014, 675–678.
[34] G.-S. Kalanit and M. Rafael, The human visual cortex, *Annual Review of Neuroscience*, **27** (2004), 649–677.
[35] K. N. Kay, T. Naselaris, R. J. Prenger and J. L. Gallant, Identifying natural images from human brain activity, *Nature*, **452** (2008), p352.
[36] A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Proceedings of the Advances in Neural Information Processing Systems*, 2012, 1097–1150.
[37] N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez and L. Wiskott, Deep hierarchies in the primate visual cortex: What can we learn for computer vision?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35** (2013), 1847–1871.
[38] A. Kurakin, I. Goodfellow and S. Bengio, Adversarial examples in the physical world, arXiv preprint, arXiv:1607.02533.
[39] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86** (1998), 2278–2324.
[40] Y. LeCun, C. Cortes and C. J. Burges, The mnist database of handwritten digits, 1998.
[41] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., Photo-realistic single image super-resolution using a generative adversarial network, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.
[42] H. Lee, C. Ekanadham and A. Y. Ng, Sparse deep belief net model for visual area v2, in *Proceedings of the Advances in Neural Information Processing Systems*, 2008, 873–880.
[43] H. Lee, R. Grosse, R. Ranganath and A. Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in *Proceedings of the International Conference on Machine Learning*, 2009, 609–616.
[44] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. Zitnick, Microsoft coco: common objects in context. corr abs/1405.0312 (2014).
[45] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International journal of computer vision*, **60** (2004), 91–110.
[46] A. Mahendran and A. Vedaldi, Understanding deep image representations by inverting them, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, 5188–5196.

[47] A. Mahendran and A. Vedaldi, Visualizing deep convolutional neural networks using natural pre-images, *International Journal of Computer Vision*, **120** (2016), 233–255.
[48] M. Manassi, B. Sayim and M. H. Herzog, When crowding of crowding leads to uncrowding, *Journal of Vision*, **13** (2013), 10–10.
[49] A. Mordvintsev, C. Olah and M. Tyka, Inceptionism: Going deeper into neural networks, *Google Research Blog. Retrieved June*, **20** (2015), 14pp.
[50] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox and J. Clune, Synthesizing the preferred inputs for neurons in neural networks via deep generator networks, in *Proceedings of the Advances in Neural Information Processing Systems*, 2016, 3387–3395.
[51] A. Nguyen, J. Yosinski and J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, 427–436.
[52] A. Nguyen, J. Yosinski and J. Clune, Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks, arXiv preprint, arXiv:1602.03616.
[53] S. J. Pan and Q. Yang, A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, **22** (2010), 1345–1359.
[54] M. I. Posner and S. E. Petersen, The attention system of the human brain, *Annual review of neuroscience*, **13** (1990), 25–42.
[55] C. Poultney, S. Chopra, Y. L. Cun et al., Efficient learning of sparse representations with an energy-based model, in *Proceedings of the Advances in Neural Information Processing Systems*, 2007, 1137–1144.
[56] N. Qian, On the momentum term in gradient descent learning algorithms, *Neural networks*, **12** (1999), 145–151.
[57] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch and I. Fried., Invariant visual representation by single neurons in the human brain, *Nature*, **435** (2005), 1102–1107, URL http://dx.doi.org/10.1038/nature03687.
[58] S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39** (2017), 1137–1149.
[59] L. I. Rudin, S. Osher and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D: nonlinear phenomena*, **60** (1992), 259–268.
[60] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura and R. M. Summers, Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning, *IEEE Transactions on Medical Imaging*, **35** (2016), 1285–1298.
[61] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., Mastering the game of go with deep neural networks and tree search, *Nature*, **529** (2016), 484–489.
[62] K. Simonyan, A. Vedaldi and A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, arXiv preprint, arXiv:1312.6034.
[63] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556.
[64] J. Sivic and A. Zisserman, Video google: A text retrieval approach to object matching in videos, in *Proceeding of Ninth IEEE International Conference on Computer Vision*, 2003, 1470.
[65] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *Journal of machine learning research*, **15** (2014), 1929–1958.
[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, 1–9.
[67] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, Intriguing properties of neural networks, arXiv preprint, arXiv:1312.6199.
[68] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, *Journal of Machine Learning Research*, **11** (2010), 3371–3408.
[69] L. Wang, Y. Zhang and J. Feng, On the euclidean distance of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27** (2005), 1334–1339.

[70] D. Wei, B. Zhou, A. Torrabla and W. Freeman, Understanding intra-class knowledge inside cnn, arXiv preprint, arXiv:1507.02379.
[71] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs and H. Lipson, Understanding neural networks through deep visualization, arXiv preprint, arXiv:1506.06579.
[72] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, in *Proceedings of the European Conference on Computer Vision*, 2014, 818–833.
[73] M. D. Zeiler, D. Krishnan, G. W. Taylor and R. Fergus, Deconvolutional networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, 2528–2535.
[74] M. D. Zeiler, G. W. Taylor and R. Fergus, Adaptive deconvolutional networks for mid and high level feature learning, in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, 2018–2025.
[75] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva and A. Torralba, Object detectors emerge in deep scene CNNs, arXiv preprint, arXiv:1412.6856.

Received October 2017; revised December 2017.

E-mail address: zqin@gmu.edu
E-mail address: fyu2@gmu.edu
E-mail address: chliu@clarkson.edu
E-mail address: xchen26@gmu.edu