Visualizing and Understanding Convolutional Networks

Preprint · October 2022

DOI: 10.13140/RG.2.2.12182.22080

2 authors, including:

Md Hafizur Rahman Masum
Frankfurt University of Applied Sciences
6 PUBLICATIONS **2** CITATIONS

SEE PROFILE

Visualizing and Understanding Convolutional Networks

Md Hafizur Rahman Masum
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
md.masum@stud.fra-uas.de
Matriculation Number: 1354904

***Abstract*—Convolutional Neural Networks (CNNs) are capable of performing impressively working on computer vision tasks of all kinds, including object identification, picture recognition, image retrieval, etc. These successes are a result of CNN's exceptional capacity to learn input information using several deep layers of neuronal structures and iterative training methods. On the other hand, from a human visual standpoint, these learnt characteristics are challenging to recognize and interpret, making it difficult to comprehend the CNN's internal workings.[1] In order to enhance CNN interpretability, the CNN visualization is effectively used as a qualitative analysis method that converts internal information into visually observable patterns. A number of visualization techniques are described in this paper, including Deconvolutional Networks (DeconNet), Activation Maximization, Network Dissection and Network Inversion.[2] The performance contribution of different model layers are also investigated through an ablation study.**

***Index Terms*—Convolutional Neural Network, Training Image, DeconNet, Activation Maximization, Network Dissection, Network Inversion**

I. Introduction

There has been excellent performance demonstrated by convolutional networks in tasks such as Classification of handwritten digits and detection of faces from the early 1990s when they were introduced by LeCun et al. [3]. A number of studies have shown that they are also capable of performing well on more demanding tasks requiring visual classification.On the ImageNet 2012 classification benchmark, (Krizhevsky et al., 2012) demonstrate record-breaking accuracy with their convnet model.[4] The 2nd place model had error rates of 26.1%, compared to 16.4% for their model. There are several factors contributing to this renewed interest in convnet models, including the size of the training set will grow as more examples are labelled, the GPU implementations will become more efficient, and the regularization strategies will improve, such as Dropout.[4].

Although these advancements are encouraging, little is known about how these complex models work or why they perform so well. A visualization technique is introduced by (Zeiler et al.) that reveals the input stimulus that excites individual feature maps across layers. As well as observing how the model evolves during training, it allows us to diagnose potential errors. In order to return the feature activations to the input pixel space, the visualization method Zeiler et al. suggest employs a multi-layered Deconvolutional Network (deconvnet), as presented by Zeiler et al. (2011). By obscuring some of the input picture, it can also be possible to do a sensitivity analysis on the classifier output and determine which aspects of the scene are crucial for categorization.[1]

Zeiler et al. examine many designs using these tools, starting with the architecture of (Krizhevsky et al., 2012), and they eventually find ones that surpass their ImageNet findings. After that, they simply retrain the softmax classifier on top of the model to see whether it can be applied to different datasets. In contrast to the unsupervised pre-training techniques made famous by some researchers, this is a type of supervised pre-training. In related work , the generalizability of convnet characteristics is also investigated[5] [6] [7] [1]

II. Related Work

It is standard practice to visualize features to obtain insight about the network, however this is often restricted to the first layer when projections to pixel space are allowed. This is not the case at deeper levels, and there are few ways to interpret activity.[8]Gradient descent in picture space is used to optimize the activation of each unit by locating the best stimulus.[1] This entails careful setup and provides no knowledge of the unit's invariances. Le et al. (2010), building on Berkes & Wiskott's concept, demonstrate how the Hessian of a given unit may be determined numerically around the optimal response, which Is motivated by the latter's shortcoming, providing some understanding of invariances.[9] [10] The issue is that a straightforward quadratic approximation does a poor job of capturing the invariances for higher layers due to their enormous complexity. Contrarily,(Zeiler et al.) method offers a way to reveal which patterns from the training data activate the feature map in a non-parametric approach of invariance. (Donahue et al., 2013) Activation patches at higher model layers are visualized as dataset patches. (Zeiler et al.) visualizations are distinct because they aren't simply cropping of the input pictures; instead, they're top-down projections that show structures inside each patch that cause a user to think in new ways about a specific feature map. [11] [1]

III. Visualization Techniques

A visualization of CNNs has been developed in order to interpret their overall functioning mechanism. A few representative visualization methods are described here, including **Deconvolutional Networks (DeconNet)**, **Activation Maximization**, **Network Dissection**, and **Network Inversion**.[2]

A. Visualization by Activation Maximization

An activation maximization method (AM) is proposed to determine what neurons prefer to input into each layer. Neurons can learn features based on their preferred input. As a result of this learning process, an input pattern is synthesized that can cause the maximal activation of a neuron. This input pattern is synthesized by iterative changing the pixels of the CNN's input to maximize the neuron's activation.

Erhan et al. proposed the fundamental algorithm behind the AM in 2009 [8]. From the MNIST digit dataset [12], The Deep Belief Nets[4] and Stacked Denoising AutoEncoders were used to visualize the preferred input patterns for the hidden neurons [13]. Using this method, Simonyan et al. maximized the activation of neurons in the last layer of CNNs [14]. Similar visualized patterns have also been synthesized by Google for its inception network [15]. In a large-scale application of the AM, Yosinksi et al. visualized all layers of a CNN with arbitrary neurons. There have been a number of optimization works conducted recently to improve the interpretability and diversity of the visualized patterns. All of these works have demonstrated that the AM is capable of interpreting neural interests and identifying hierarchical features learned by CNNs.

This method of visualization reveals that CNNs are able to identify faces, wheels, and bottles without our guidance. Similarly, CNNs offer a hierarchical feature extraction method that mimics the hierarchical organization of the visual cortex. Additionally, this visualization method suggests that neurons extract features in a localized manner rather than a distributed manner, where neurons match patterns based on their localization.[1]

B. Visualization by Network Inversion

An arbitrary layer can be used to reconstruct an image that illustrates the CNN layer-level feature for a given input image. When layer-level activation is examined instead of individual neuron activation, a comprehensive feature representation of all activation patterns on a layer is revealed. Accordingly, network inversion-based visualization allows analyzing activation information at the layer level rather than from the perspective of a single neuron as in the aforementioned visualization methods. Before CNNs were used to visualize, Network Inversion was originally proposed to study traditional computer vision representations. Histograms of Oriented Gradients (HOGs) are one type of representation. [16] [17] [18], the Scale Invariant Feature Transform (SIFT) [19], Local Binary Descriptors (LBD)[20], and the Bag of Visual Word Descriptors [21] [22]. The Network Inversion was later used to visualize CNN [23] [24] in two variants: (1) Regularizer based Network Inversion: It is proposed by Mahendran et al., which uses gradient descent with a regularization term to reconstruct the image from each layer. (2)Dosovitskiy et al. [24] proposed the UpconvNet-based Network Inversion: A convolutional neural network (UpconvNet) is trained to reconstruct the image. Based on the specific activation of one entire layer's feature maps, both algorithms primarily strive to reconstruct the original input image. Since there is no need to train a second dedicated network, Regularizer-based Network Inversion is simpler to implement. A Network Inversion based on UpconvNet, on the other hand, is able to visualize more actual data at a higher layer of abstraction.

C. Visualization by Network Dissection

Convolutional neural networks or multiple neurons can be correlated with specific semantic concepts by evaluating their relationship. The previous section demonstrated how to visualize the visual patterns capturing a single neuron or layer by using various visualization techniques. There is still a gap between visually perceptible patterns and semantic concepts that are comprehensible.

In order to understand color, textures, materials, parts, objects, and scenes, Bau et al. proposed the Network Dissection, in which each convolutional neuron is directly associated with a particular semantic concept. By seeking neuron responses that are strongly correlated with semantic concepts, we can measure the correlation between neurons and semantic concepts [25]. With the Borden dataset, heterogeneous image data is labeled with local semantic concepts. In "Figure. 1" We see several Broden examples. There are six categories of semantic concepts highlighted with red boxes in "Figure. 1". There are many semantic categories. For example, the object category includes plants, trains, etc. Each example in "Figure. 1". has a lower right corner. It is also possible to identify the semantic corresponding neuron in "Figure. 1". Furthermore, black masks cover image content not related to the assigned semantics. Network Dissection is responsible for generating these black masks.[25]

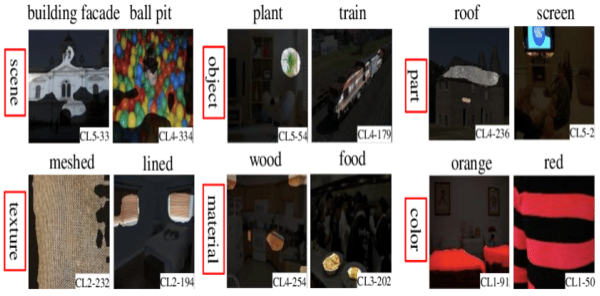

Fig. 1. AlexNet neurons are activated by Broden images.[20]

The Network Inversion works to gradually establish connections between semantic concepts and the many different

component levels of CNN. One semantic concept and one individual neuron were shown to be correlated by the Network Inversion's basic algorithm. A single neuron can be attributed to each semantic concept, which was the foundation for this correlation [26]. Later, more Network Inversion research showed that the feature representation could be distributed, suggesting that a semantic concept might be represented by a combination of different neurons [27]. Accordingly, Fong et al. proposed another Network Inversion method, called Net2Vec, which uses neuron combinations to visualize semantic concepts [28], following the paradigm of[25]. The Net2Vec Network Inversion method proposed by Fong et al. makes it possible to see semantic concepts based on combinations of neurons [25]. Both approaches offer thorough visualization outcomes for understanding CNN hidden neurons.

D. Visualization by Deconvolutional Network

Decide which convolutional layer is activated by a specific pattern from a given input image. CNN visualization based on deconvolutional networks (DeconvNet) describes CNNs from the perspective of the input image, as opposed to activation maximization. The convolutional layers are activated by selective patterns in the input image. An image dimension is projected onto low-dimensional neuronal feature maps to reconstruct patterns. Convolutions and pooling layers are reversed by an unpooling layer in a DeconvNet structure to accomplish this projection process. DeconvNet-based visualization not only analyzes neuronal interests but also demonstrates simple image-level feature analysis.

In most cases, Zeiler et al. lead research related to DeconvNet structures. DeconvNet was first proposed in [1] with the objective of reconstructing the natural image through the projection of a highly diverse set of low-dimension feature maps into high dimensions. Using the DeconvNet structure, they decomposed an image hierarchically in [1], capturing information at both low-level edges and high-level object parts as well as in between. Eventually, they applied the The method of interpreting CNN hidden features is an effective way to visualize CNNs using DeconvNet structure.

In this paper I described Visualization by Deconvolutional Network.

IV. Approach

Throughout the research, we employ typical fully supervised convnet models according to (LeCun et al., 1989) and (Krizhevsky et al., 2012) [12] [7]. These models convert a color 2D input picture (xi) into a probability vector (yi) over the C distinct classes by means of a number of layers. As an example, the first layer consists of convolution of the output of the previous layer with learned filters (relu(x) = max(x; 0)); (ii) passing the responses through a rectified linear function (max pooling over local neighborhoods is optional); (iii) normalizing the responses across feature maps by using a local contrast operation (optional). See (Krizhevsky et al., 2012) and for further information on these operations (Jarrett et al., 2009) [3]. Traditional fully-connected networks make up the top several layers of the network, and a softmax classifier is the topmost layer [1]. The model utilized in several of our tests is seen in "Figure. 2".

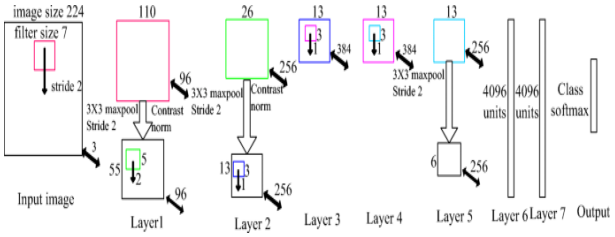

Fig. 2. The architecture of our 8 layer convnet model. The input is a 224 by 224 crop of a picture (with three color planes). Using a stride of 2 in both x and y, this is convolved with 96 separate first layer filters, each measuring 7 by 7, each in the color red. Following that, the obtained feature maps are subjected to a rectified linear function (not shown), (ii) pooled (max within 3x3 zones, using stride 2), and (iii) contrast adjusted across feature maps to produce 96 distinct 55 by 55 element feature maps. Layers 2,3,4,5 repeat similar procedures. The last two layers are completely linked and receive vectorized features from the top convolutional layer (6 6 256 = 9216 dimensions) as input. A C-way softmax function, where C is the number of classes, makes up the final layer. The shape of all filters and feature maps is square [1].

Zeiler et al. use a sizable set of N labeled pictures fx; yg to train these models, where the real class is identified by the discrete variable label yi. In order to compare yi with xi, they use a cross-entropy loss function that is suitable for classifying images. The network's parameters, such as the filters in the convolutional layers, weight matrices in the fully connected layers, and biases, are trained by backpropagating the derivative of the loss with respect to the parameters present throughout the network and updating the parameters using stochastic gradient descent.In section VI provides comprehensive training information[1].

V. Visualization with a Deconvnet

Interpreting the feature activity in intermediary levels is necessary to comprehend how a convnet works. By demonstrating what input pattern initially generated a certain activation in the feature maps, we(Zeiler et. al.) provide a unique method to map these activities back to the input pixel space. Using a deconvolutional network (deconvnet), they carry out this mapping (Zeiler et al., 2011). A deconvnet can be compared to a convnet model that employs the same building blocks (filtering, pooling) but operates in the other direction, mapping pixels to features instead. Deconvnets were suggested as a method of doing unsupervised learning in (Zeiler et al., 2011). Here, they serve just as a test for a previously trained convnet and are not employed in any learning capacities[1].

As shown in "Figure. 3" (Top), a deconvnet is coupled to each layer of a convnet to provide a continuous path back to the picture pixels. An input picture is first shown to the convnet, and features are calculated across all of the layers. Setting all other activations in the layer to zero and feeding the feature maps into the associated deconvnet layer allows

us to study a specific convnet activation. In order to rebuild the activity in the layer underneath that gave birth to the selected activation, we then unpool, (ii) correct, and (iii) filter the data in turn. Then, until input pixel space is reached, this is repeated[1].

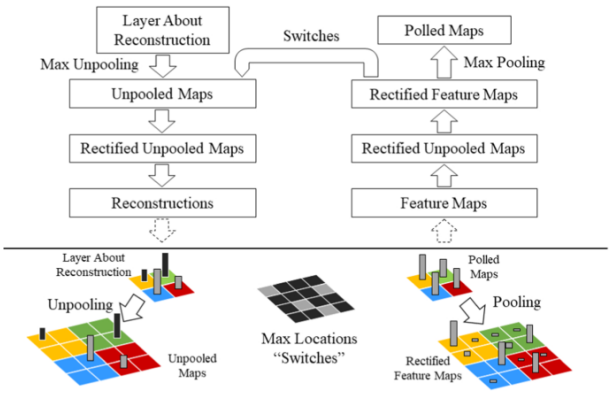

Fig. 3. Top: deconvnet and convnet layers (right). The deconvnet reconstructs the convnet layer underneath. Bottom: Unpooling in deconvnet utilizing switches that record the local max in each pooling region (colored zones) during convnet pooling. [1]

Unpooling: Max pooling is not invertible in the convnet, but by storing the maxima in each pooling zone in switch variables, we may approximate its inverse. Switches are used in deconvnet during unpooling to place reconstructions from the layer above appropriately while maintaining stimulus structure. "Figure. 3" (Bottom) shows the technique.

Rectification: Relu non-linearities are used by the convnet to correct the feature maps, making sure they are always positive. We run the reconstructed signal via a relu non-linearity in order to get accurate feature reconstructions at each layer (which also need to be positive).

Filtering: To convolve the feature maps from the preceding layer, the convnet employs learnt filters. The deconvnet inverts this by applying transposed copies of the same filters on the corrected maps rather than the output of the layer below. In actuality, this entails flipping each filter both horizontally and vertically.

A switch setting is used when projecting down from a higher level based on the maximum pooling in the convnet. The reconstruction that was obtained from a single activation consequently resembled a very small percentage of the original input image. Based on the switch settings used in the input image, the structures were weighted based on their contribution to feature activation. As a result of the model's discriminative training, it is implicitly obvious which elements of the input picture are discriminative. Because there is no generating process involved, it should be noted that these projections are not samples from the model[1].

VI. Training Details

In Section 4, a big convnet model will be displayed. This section describes such model. The architecture utilized by (Krizhevsky et al., 2012) for ImageNet classification is depicted in "Figure. 2". One distinction is that our (Zeiler et al.) model uses dense connections instead of the sparse connections utilized in Krizhevsky's layers 3, 4, which were chosen since the model was spread across two GPUs. Following examination of the visualizations in "Figure. 7", more significant distinctions pertaining to layers 1 and 2 were made, as detailed in Section VII-A.

The model was trained using the 1.3 million photos from the ImageNet 2012 training set, which were distributed across 1000 distinct classes. To prepare each RGB image, the lowest dimension was increased to 256, the center 256x256 region was cropped. After subtracting the per-pixel mean (across all pictures), ten distinct sub-crops of size 224 by 224 were employed, with or without horizontal flips. These sub-crops included the corners and the center of the image. Stochastic gradient descent with a mini-batch size of 128, a momentum term of 0.9, and an initial learning rate of 10^{-2} , was used to update the parameters. 10^{-2} When the validation error reaches a plateau, we manually anneal the learning rate during training. The completely linked layers (6 and 7) employ dropout (Hinton et al., 2012) at a rate of 0.5. Biases are set to 0 and all weights are initialized at 10^{-2}.[4]

A couple of the first layer filters become more prominent when they are seen during training, as illustrated in "Figure. 8 (a)". To counteract this, each convolutional layer's filter whose RMS value exceeds a predetermined radius of 101 has its RMS value renormalized to this fixed radius. This is essential, especially in the model's first layer because the input photos approximately fall inside the [-128,128] range. We construct numerous distinct crops and flips of each training sample, as in (Krizhevsky et al., 2012), to increase the size of the training set. Using an implementation based on (Krizhevsky et al., 2012), we ended training after 70 epochs, which on a single GTX580 GPU took about 12 days [7] [1].

VII. Convnet Visualization

In this section, the deconvnet model is utilized to display the feature activations on the ImageNet validation set by using the model discussed in Section VI of this article.

Feature Visualization: After training is finished, the model's feature visualizations are shown in "Figure. 6". However, it provide the top 9 activations rather than just the strongest activation for a particular feature map. The multiple structures that excite a particular feature map are shown by projecting each independently into pixel space, demonstrating the feature map's invariance to input deformations. They also display the relevant picture patches with these visualizations. These differ more from visualizations, which just highlight the discriminant structure inside each patch. For example,

in layer 5, row 1, column 2, the patches don't appear to have much in common; yet, the visualizations reveal that this feature map focuses on the grass in the background rather than the things in the front. [1].

The projections from each layer demonstrate how the network's characteristics are organized hierarchically. Corners and other edge/color conjunctions are handled by layer 2. Layer 3 features more intricate invariances that capture comparable textures, such as text (R2,C4) and mesh patterns (Row 1, Col 1). Significant variety is seen in Layer 4, however it is more class-specific: dog faces (R1,C1); bird's legs (R4,C2). Layer 5 displays whole objects with noticeable posture change, such as dogs (R1,C11) and keyboards (R1,C11) (R4)[1].

Feature Evolution during Training: In "Figure. 4" depicts the progression of the most significant activation (across all training instances) inside of a particular feature map when the map is projected back to pixel space. A shift in the picture that causes the strongest activation causes abrupt jumps in appearance. Within a few epochs, the model's lowest layers may be observed to converge. However, the development of the top layers takes place after a sizable number of epochs (40–50), highlighting the necessity of letting the models train until they have fully converged.

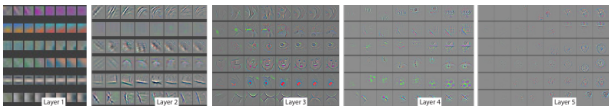

Fig. 4. Training-driven evolution of a subset of model characteristics. Different blocks display each layer's features. It displays a random selection of characteristics at epochs [1,2,5,10,20,30,40,64]. The graphic depicts the strongest activation (across all training samples) for a specific feature map in pixel space. The figure's color contrast is boosted, and it's best viewed electronically.[1]

Feature invariance: In "Figure. 5", five sample images with varied degrees of translation, rotation, and scaling are shown. As a comparison to the untransformed features, the top and bottom feature vectors of the model are examined. Small changes have a significant influence in the model's top feature layer, where translation and scaling are quasilinear and have a less dramatic effect. The network output is resilient to scaling and translation. With the exception of objects with rotational symmetry, the output is often not rotation-invariant (e.g., the entertainment center).

A. Architecture Choosing

Visualization can help you understand how a trained model works, and it can also help you choose designs that will work. When looking at Krizhevsky et al.'s first and second levels "Figure. 5 (b) & (d)", a number of issues become clear. Since the first layer filters don't cover enough of the midrange frequencies, they combine information at very high and very low frequencies. In addition, the aliasing artifacts brought on

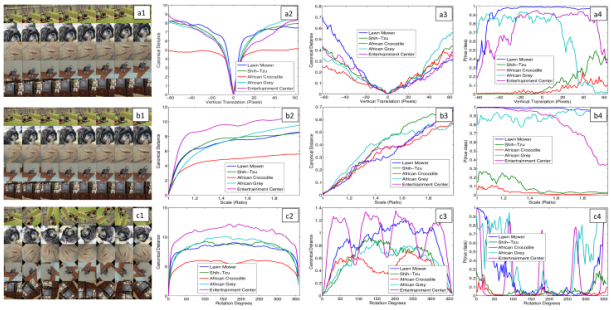

Fig. 5. Examines the model's vertical translation, scaling, and rotation invariance (rows a-c respectively). Col. 1: 5 illustrations of transformations in action. Cols. 2 and 3 show the Euclidean distance between the feature vectors in layers 1 and 7, respectively, from the original and modified pictures. Col. 4: As a picture is altered, the likelihood that each label is accurate.[1]

Fig. 6. Fully trained model characteristics For layers 2–5, it presents the top 9 activation's in a random subset of validation feature maps projected to pixel space. The reconstructions aren't model samples; they're validation set patterns that activate a feature map. It shows picture patches for each feature map. Notice how each feature map is tightly clustered, how there is more variation at higher layers, and how distinguishing parts of the picture, like dog eyes and noses (layer 4, row 1, cols 1), are made bigger. The electronic format is preferred.[1]

by the first layer convolutions' use of a long stride 4 are visible in the second layer visualization. In order to address these issues, we decreased the size of the first layer filter from 11x11 to 7x7 and (ii) made the convolution stride 2 rather than 4. According to "Figure. 5 (c) & (d)", this new design preserves significantly more information in the first and second layer features "Figure. 5 (e)". More significantly, it enhances classification performance, as seen in Section ImageNet 2012.

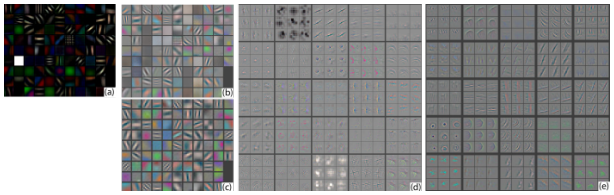

Fig. 7. (a) Features of the first layer without clipping A single feature dominates.(b) The first layer from Krizhevsky et al. (2012). first layer characteristics The reduced stride (2 versus 4) and filter size (7x7 vs. 11x11) provide more distinctive and less "dead" features. (c) Krizhevsky et al.'s second layer characteristics (e) Visuals on the second layer. (d)No aliasing artifacts are seen in these. [1]

B. Occlusion Sensitivity

Image classification techniques make you wonder if the model really knows where an object is or if it just uses context. "Figure. 8" answers this question by systematically obscuring the input picture with grey squares and monitoring the classifier's output. The examples indicate the model localizes objects in the scene, since the likelihood of the proper class lowers when The item is obscured. "Figure. 8" shows how the top convolution layer's strongest feature map and its activity (across all locations) change depending on where the occluder is. When the occluder covers a visualization zone, feature map activity drops dramatically. This demonstrates that the visualization matches the picture structure that drives the feature map, verifying "Figure. 4" & "Figure. 7"[1].

C. Correspondence Analysis

Deep models are different from many earlier recognition systems in that there is no explicit way to build a link between different parts of objects in different photos. Deep Models may implicitly compute them, though. We take five randomly selected frontal dog photos and mask out the identical section of the face in each (e.g. all the left eyes, see Figure 9). A smaller Hamming distance value means that the same parts of the same object are closer together in more than one picture and that the change caused by the masking operation is more consistent. For example, closing the left eye always changes how the feature is shown[1].

VIII. Experiments

A. ImageNet 2012

This dataset has 1.3 million training examples, 50 thousand validation examples, and 100 thousand test examples, spread across 1,000 categories. Zeiler et al. attempt to replicate Krizhevsky et al. (2012)'s findings using the precise

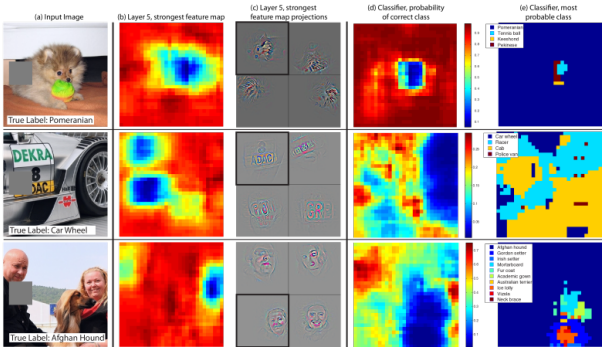

Fig. 8. This test example shows how the top five feature maps ((b) & (c)) and classifier output ((d) & (e)) change as gray squares are systematically drawn over different parts of the scene (1st column). To determine the total activation for each gray scale position, we record it in one feature map in layer 5 (the one with the strongest response in the unoccluded image). Feature map projections for the input image (black square) and projections of the feature map from other images are shown in (c). A dog's face is the most prominent feature in the first row example. In (b) the blue area in the feature map indicates decreased activity when this area is covered-up. It shows the probability of a correct class as a function of the location of the gray square (d). The probability of pomeranian" drops significantly when the dog's face is obscured. E.g. in the first row, the likelihood drops significantly. (f): the most probable label based on occluder position. It predicts pomeranian" in most locations, but tennis ball" if the dog's face, but not the ball, is obscured. The wheel is most sensitive to the classifier in the second example since the text on the car is the strongest feature in layer 5. There are multiple objects in the third example. Since layer 5 uses multiple feature maps, it is sensitive to the dog (blue region in (d)) due to the strongest feature in layer 5 picking out the faces. [1]

Fig. 9. Images used in tests on correspondence Col. 1: The first image Cols. 2, 3, and 4: The nose, right eye, and left eye, respectively, are blocked. Other columns show instances of arbitrary occlusions.[1]

architecture they recommended on the validation set. On the ImageNet 2012 validation set, they get an error rate that is within 1% of what is reported.

The performance of our model after the structural modifications described in Section 4.1 (77 filters in layer 1 and stride 2 convolutions in layers 1 & 2) is next examined. By exceeding their single model result by 1:7%, the model in "Figure. 2" greatly beats the architecture of Krizhevsky et al. (2012). (test top-5). It acquires a test error of 14:8% when it combines several models, which is the best documented performance on this dataset1 (despite only using the 2012 training set). We(Zeiler et al.) see that this error is nearly half

that of the best non-convnet submission for the ImageNet 2012 classification challenge, which had a 26:2% error rate (Gunji et al., 2012)[29].

B. Varying ImageNet Model Sizes

By changing the thickness of the layers or deleting them altogether, Zeiler et al. investigate the architecture of (Krizhevsky et al., 2012) in the beginning. Each time, the model is retrained using the updated architecture. Remove the completely linked layers (6,7), and the inaccuracy is only slightly increased. Given that they include the bulk of model parameters, this is unexpected. Two of the intermediate convolutional layers being removed also changes the error rate just little. However, a model with only 4 layers and significantly inferior performance is produced by deleting both the intermediate convolution layers and the fully linked layers. This would imply that achieving high performance depends on the model's entire depth. In "Figure. 2". Performance is not significantly affected by changing the amount of completely linked layers (similar to the model from Krizhevsky et al., 2012). However, enlarging the middle convolution layers can result in a useful speed improvement. However, expanding the completely linked layers while concurrently increasing these leads to over-fitting[7][1].

C. Feature Generalization

The studies above demonstrate the significance of the convolutional component of the Zeiler et al. ImageNet model in achieving cutting-edge performance. The visualizations in "Figure. 6", which display the intricate invariances learnt in the convolutional layers, support this. The capacity of these feature extraction layers to generalize to additional datasets, including Caltech-101 , Caltech-256 and PASCAL VOC 2012, is presently being investigated [30] [31]. To do this, they train a new softmax classifier (with the necessary number of classes) on top of their ImageNet-trained model, keeping layers 1 through 7 fixed. They accomplish this using the training pictures from the newly acquired dataset. The softmax can be trained fast from a limited number of instances, as is the case for some datasets, because it has a small number of parameters.

Before retraining the Imagenet models, they(Zeiler et al.) identified these few overlap images with normalized correlation and removed them from the Imagenet training set to prevent train/test contamination.[1].

IX. Conclusion

In various methods, researchers investigated large convolutional neural network models that have been trained for image categorization. First, researchers offered a fresh method for visualizing the model's activities. This demonstrates that the traits are not random, puzzling patterns. Instead, as we move up the layers, these exhibit numerous qualities that are intuitively appealing, including compositionality, rising invariance, and class discrimination. It also demonstrated how these visualizations may be used to diagnose model issues and enhance outcomes, such as the outstanding ImageNet 2012 result[7]. Researchers later showed by testing the model with occlusions that the model is not only sensitive to local structures in an image but also to broad scene context. Having a minimal depth to the network, as opposed to any specific region, is crucial to the model's performance, according to an ablation research on it[1].

References

[1] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[2] Z. Qin, F. Yu, C. Liu, and X. Chen, "How convolutional neural network see the world-a survey of convolutional neural network visualization methods," *arXiv preprint arXiv:1804.11191*, 2018.

[3] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" In *2009 IEEE 12th international conference on computer vision*, IEEE, 2009, pp. 2146–2153.

[4] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[9] P. Berkes and L. Wiskott, "On the analysis and interpretation of inhomogeneous quadratic forms as receptive fields," *Neural computation*, vol. 18, no. 8, pp. 1868–1895, 2006.

[10] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[11] J. Donahue, Y. Jia, O. Vinyals, *et al.*, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, PMLR, 2014, pp. 647–655.

[12] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.," *Journal of machine learning research*, vol. 11, no. 12, 2010.

[14] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[15] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Ieee, vol. 1, 2005, pp. 886–893.

[17] 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*, vol. 32, no. 9, pp. 1627–1645, 2010.

[18] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester, "Discriminatively trained deformable part models, release 5," 2012.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[20] E. d'Angelo, A. Alahi, and P. Vandergheynst, "Beyond bits: Reconstructing images from local binary descriptors," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, IEEE, 2012, pp. 935–938.

[21] 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*, Prague, vol. 1, 2004, pp. 1–2.

[22] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, IEEE International Conference on*, IEEE Computer Society, vol. 3, 2003, pp. 1470–1470.

[23] 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, pp. 5188–5196.

[24] N. Mayer, E. Ilg, P. Hausser, *et al.*, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[25] 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, pp. 6541–6549.

[26] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari, "Do semantic parts emerge in convolutional neural networks?" *International Journal of Computer Vision*, vol. 126, no. 5, pp. 476–494, 2018.

[27] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *European conference on computer vision*, Springer, 2014, pp. 329–344.

[28] R. Fong and A. Vedaldi, "Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8730–8738.

[29] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," *arXiv preprint arXiv:1312.5402*, 2013.

[30] G Griffin, A Holub, and P Perona, "The caltech 256 (technical report)," *Caltech Computation and Neural Systems*, 2006.

[31] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.