Understanding the Role of Individual Units in a Deep Neural Network

David Bau[a,1], Jun-Yan Zhu[a,f], Hendrik Strobelt[b], Agata Lapedriza[a,c], Bolei Zhou[d], and Antonio Torralba[a]

This preprint was compiled on September 11, 2020. The peer-reviewed version can be found at www.pnas.org/cgi/doi/10.1073/pnas.1907375117.

Deep neural networks excel at finding hierarchical representations that solve complex tasks over large data sets. How can we humans understand these learned representations? In this work, we present network dissection, an analytic framework to systematically identify the semantics of individual hidden units within image classification and image generation networks. First, we analyze a convolutional neural network (CNN) trained on scene classification and discover units that match a diverse set of object concepts. We find evidence that the network has learned many object classes that play crucial roles in classifying scene classes. Second, we use a similar analytic method to analyze a generative adversarial network (GAN) model trained to generate scenes. By analyzing changes made when small sets of units are activated or deactivated, we find that objects can be added and removed from the output scenes while adapting to the context. Finally, we apply our analytic framework to understanding adversarial attacks and to semantic image editing.

Machine Learning | Deep Networks | Computer Vision

Can the individual hidden units of a deep network teach us how the network solves a complex task? Intriguingly, within state-of-the-art deep networks, it has been observed that many single units match human-interpretable concepts that were not explicitly taught to the network: units have been found to detect objects, parts, textures, tense, gender, context, and sentiment (1–7). Finding such meaningful abstractions is one of the main goals of deep learning (8), but the emergence and role of such concept-specific units is not well-understood. Thus we ask: how can we quantify the emergence of concept units across the layers of a network? What types of concepts are matched; and what function do they serve? When a network contains a unit that activates on trees, we wish to understand if it is a spurious correlation, or if the unit has a causal role that reveals how the network models its higher-level notions about trees.

To investigate these questions, we introduce *network dissection* (9, 10), our method for systematically mapping the semantic concepts found within a deep convolutional neural network. The basic unit of computation within such a network is a learned convolutional filter; this architecture is the state-of-the-art for solving a wide variety of discriminative and generative tasks in computer vision (11–19). Network dissection identifies, visualizes, and quantifies the role of individual units in a network by comparing the activity of each unit to a range of human-interpretable pattern-matching tasks such as the detection of object classes.

Previous approaches for understanding a deep network include the use of salience maps (20–27): those methods ask *where* a network looks when it makes a decision. The goal of our current inquiry is different: we ask *what* a network is looking for, and why. Another approach is to create simplified surrogate models to mimic and summarize a complex network's behavior (28–30); and another technique is to train explanation networks that generate human-readable explanations of a network (31). In contrast to those methods, network dissection aims to directly interpret the internal computation of the network itself, rather than training an auxiliary model.

We dissect the units of networks trained on two different types of tasks: image classification and image generation. In both settings, we find that a trained network contains units that correspond to high-level visual concepts that were not explicitly labeled in the training data. For example, when trained to classify or generate natural scene images, both types of networks learn individual units that match the visual concept of a 'tree' even though we have never taught the network the tree concept during training.

Focusing our analysis on the units of a network allows us to test the causal structure of network behavior by activating and deactivating the units during processing. In a classifier, we use these interventions to ask whether the classification performance of a specific class can be explained by a small number of units that identify visual concepts in the scene class. For example, we ask how the ability of the network to classify an image as a ski resort is affected when removing a few units that detect snow, mountains, trees, and houses. Within a scene generation network, we ask how the rendering of objects in a scene is affected by object-specific units. How does the removal of tree units affect the appearance of trees and other objects in the output image?

Finally, we demonstrate the usefulness of our approach with two applications. We show how adversarial attacks on a classifier can be understood as attacks on the important units for a class. And we apply unit intervention on a generator to enable a human user to modify semantic concepts such as trees and doors in an image by directly manipulating units.

[a]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139; [b]Massachusetts Institute of Technology–International Business Machines (IBM) Watson Artificial Intelligence Laboratory, Cambridge, MA 02142; [c]Media Lab, Massachusetts Institute of Technology, Cambridge, MA 02139; [d]Estudis d'Informàtica, Multimèdia i Telecomunicació, Universitat Oberta de Catalunya, 08018 Barcelona, Spain; [e]Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China; [f]Adobe Research, Adobe Inc., San Jose, CA 95110
[1]Corresponding author e-mail: davidbau@csail.mit.edu

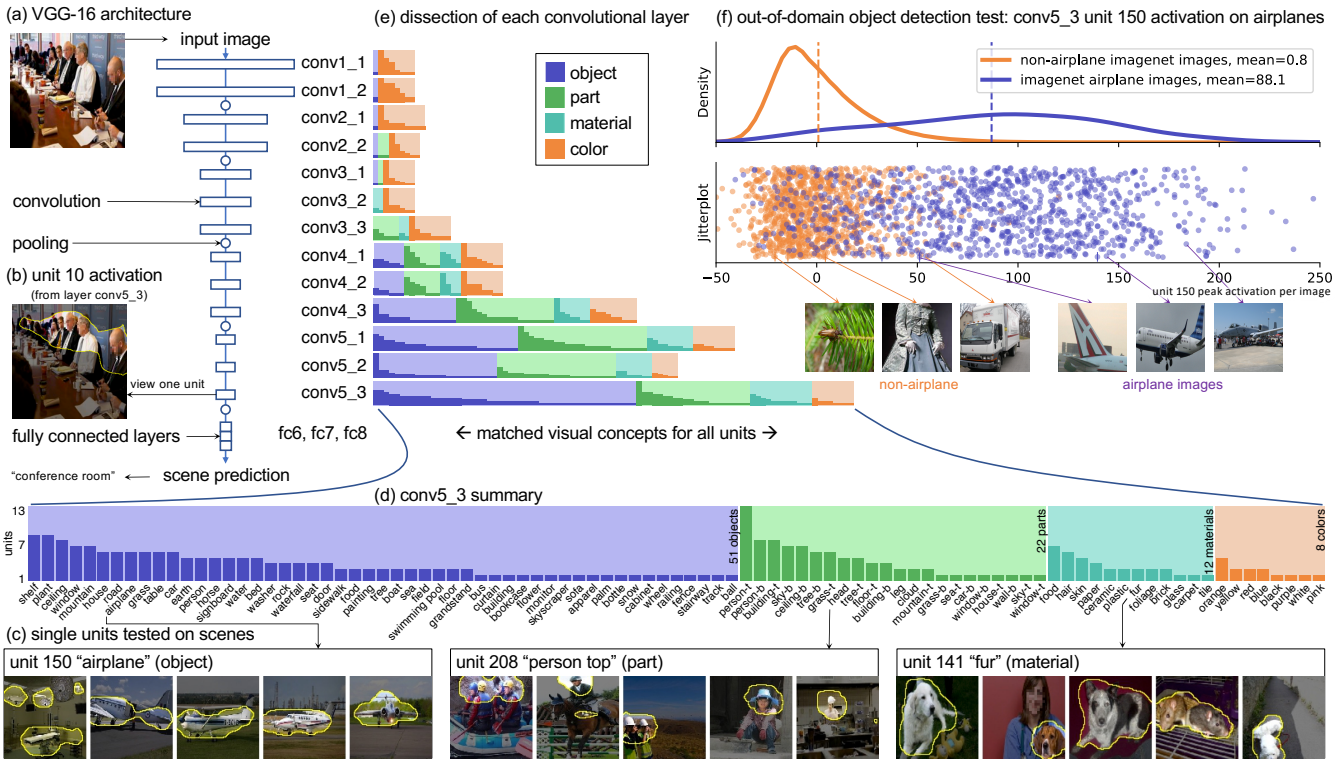

Fig. 1. The emergence of single-unit object detectors within a VGG-16 scene classifier. (a) VGG-16 consists of 13 convolutional layers, `conv1_1` through `conv5_3`, followed by three fully connected layers, `fc6,7,8`. (b) The activation of a single filter on an input image can be visualized as the region where the filter activates beyond its top 1% quantile level. (c) Single units are scored by matching high-activating regions against a set of human-interpretable visual concepts; each unit is labeled with its best-matching concept and visualized with maximally-activating images. (d) Concepts that match units in the final convolutional layer are summarized, showing a broad diversity of detectors for objects, object parts, materials, and colors. Many concepts are associated with multiple units. (e) Comparing all the layers of the network reveals that most object detectors emerge at the last convolutional layers. (f) Although the training set contains no object labels, unit 150 emerges as an 'airplane' object detector that activates much more strongly on airplane objects than non-airplane objects, as tested against a dataset of labeled object images not previously seen by the network. The jitter plot shows peak activations for the unit on randomly sampled 1,000 airplane and 1,000 non-airplane Imagenet images, and the curves show the kernel density estimates of these activations.

Results

Emergence of Object Detectors in a Scene Classifier. We first identify individual units that emerge as object detectors when training a network on a scene classification task. The network we analyze is a VGG-16 CNN (13) trained to classify images into 365 scene categories using the places365 data set (32). We analyze all units within the 13 convolutional layers of the network (Figure 1a). Please refer to materials and methods for further details on networks and datasets.

Each unit u computes an activation function $a_u(x, p)$ that outputs a signal at every image position p given a test image x. Filters with low-resolution outputs are visualized and analyzed at high-resolution positions p using bilinear upsampling. Denote by t_u the top 1% quantile level for a_u, that is, writing $\mathbb{P}_{x,p}[\cdot]$ to indicate the probability that an event is true when sampled over all positions and images, we define the threshold $t_u \equiv \max_t \mathbb{P}_{x,p}[a_u(x, p) > t] \geq 0.01$. In visualizations we highlight the activation region $\{p \mid a_u(x, p) > t_u\}$ above the threshold. As seen in Figure 1b, this region can correspond to semantics such as the heads of all the people in the image. To identify filters that match semantic concepts, we measure the agreement between each filter and a visual concept c using a computer vision segmentation model (33) $s_c : (x, p) \to \{0, 1\}$ that is trained to predict the presence of the visual concept c within image x at position p. We quantify the agreement between concept c and unit u using the intersection over union (IoU) ratio:

$$\text{IoU}_{u,c} = \frac{\mathbb{P}_{x,p}[s_c(x, p) \wedge (a_u(x, p) > t_u)]}{\mathbb{P}_{x,p}[s_c(x, p) \vee (a_u(x, p) > t_u)]} \quad [1]$$

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

When examining all 512 units in the last convolutional layer, we find many detected object classes and relatively fewer detected object parts and materials: `conv5_3` units match 51 object classes, 22 parts, 12 materials, and 8 colors. Several visual concepts such as 'airplane' and 'head' are matched by more than one unit. Figure 1d lists every segmented concept matching units in layer `conv5_3` excluding any units with IoU ratio $< 4\%$, showing the frequency of units matching each concept. Across different layers, the last convolutional layer has the largest number of object classes detected by units, while the number of object parts peaks two layers earlier, at `conv5_1`, which has units matching 28 object classes, 25 parts, 9 materials, and 8 colors (Figure 1e). A complete visualization of all the units of `conv5_3` is provided in SI, as well as more detailed comparisons between layers of VGG-16, comparisons to layers of AlexNet (12) and ResNet (16), and an

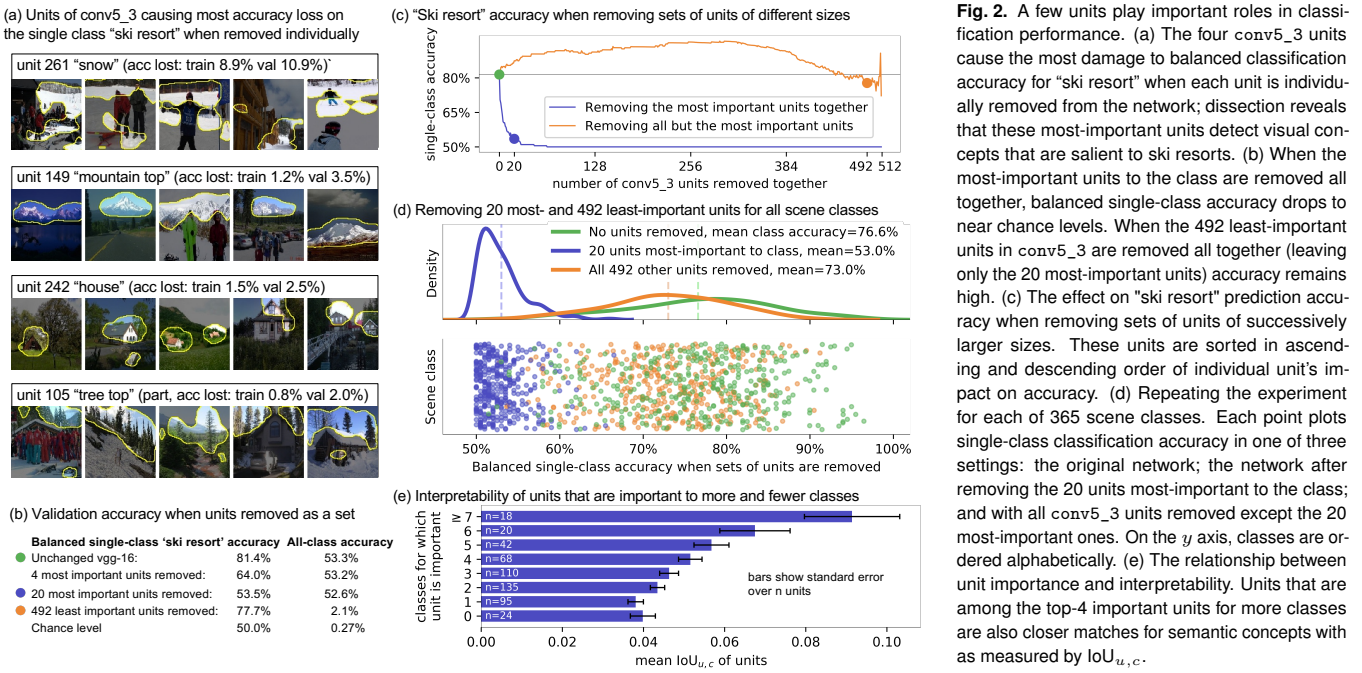

Fig. 2. A few units play important roles in classification performance. (a) The four `conv5_3` units cause the most damage to balanced classification accuracy for "ski resort" when each unit is individually removed from the network; dissection reveals that these most-important units detect visual concepts that are salient to ski resorts. (b) When the most-important units to the class are removed all together, balanced single-class accuracy drops to near chance levels. When the 492 least-important units in `conv5_3` are removed all together (leaving only the 20 most-important units) accuracy remains high. (c) The effect on "ski resort" prediction accuracy when removing sets of units of successively larger sizes. These units are sorted in ascending and descending order of individual unit's impact on accuracy. (d) Repeating the experiment for each of 365 scene classes. Each point plots single-class classification accuracy in one of three settings: the original network; the network after removing the 20 units most-important to the class; and with all `conv5_3` units removed except the 20 most-important ones. On the y axis, classes are ordered alphabetically. (e) The relationship between unit importance and interpretability. Units that are among the top-4 important units for more classes are also closer matches for semantic concepts with as measured by IoU$_{u,c}$.

analysis of the texture versus shape sensitivity of units using a stylization method based on (34).

Interestingly, object detectors emerge despite the absence of object labels in the training task. For example, the aviation-related scene classes in the training set are 'airfield', 'airport terminal', 'hangar', 'landing deck', and 'runway.' Scenes in these classes do not always contain airplanes, and there is no explicit 'airplane' object label in the training set. Yet unit 150 emerges as a detector that locates airplanes, scoring IoU $= 9.0\%$ agreement with our reference airplane segmentations in scene images. The accuracy of the unit as an airplane classifier can be further verified on Imagenet (35), a dataset that contains 1,000 object classes; its images and classes are disjoint from the Places365 training set. Imagenet contains two airplane class labels: 'airliner' and 'warplane', and a simple threshold on unit 150 (peak activation > 23.4) achieves 85.6% balanced classification accuracy on the task of distinguishing these airplane classes from the other object classes. Figure 1f shows the distribution of activations of this unit on a sample of airplane and non-airplane Imagenet images.

Role of Units in a Scene Classifier. How does the network use the above object detector units? Studies of network compression have shown that many units can be eliminated from a network while recovering overall classification accuracy by retraining (36, 37). One way to estimate the importance of an individual unit is to examine the impact of the removal of the unit on mean network accuracy (38, 39).

To obtain a more fine-grained understanding of the causal role of each unit within a network, we measure the impact of removing each unit on the network's ability of classifying each individual scene class. Units are removed by forcing the specified unit to output zero and leaving the rest of the network intact. No retraining is done. Single-class accuracy is tested on the balanced two-way classification problem of discriminating the specified class from all the other classes.

The relationships between objects and scenes learned by the network can be revealed by identifying the most important units for each class. For example, the four most important `conv5_3` units for the class 'ski resort' are shown in Figure 2a: these units damage ski resort accuracy most when removed. The units detect snow, mountains, houses, and trees, all of which seem salient to ski resort scenes.

To test whether the ability of the network to classify ski resorts can be attributed to just the most important units, we remove selected sets of units. Figure 2b shows that removing just these 4 (out of 512) units reduces the network's accuracy at discriminating 'ski resort' scenes from 81.4% to 64.0%, and removing the 20 most important units in `conv5_3` reduces class accuracy further to 53.5%, near chance levels (chance is 50%), even though classification accuracy over all scene classes is hardly affected (changing from 53.3% to 52.6%, where chance is 0.27%). In contrast, removing the 492 least-important units, leaving only the 20 most important units in `conv5_3`, has only a small impact on accuracy for the specific class, reducing ski resort accuracy by only 3.7%, to 77.7%. Of course, removing so many units damages the ability of the network to classify other scene classes: removing the 492 least-important units reduces all-class accuracy to 2.1% (chance is 0.27%).

The effect of removing varying numbers of most-important and least-important units upon 'ski resort' accuracy is shown in Figure 2c. To avoid overfitting to the evaluation data, we rank the importance of units according to their individual impact on single-class ski resort accuracy on the training set, and the plotted impact of removing sets of units is evaluated on the held-out validation set. The network can be seen to derive most of its performance for ski resort classification from just the most important units. Single-class accuracy can even be improved by removing the least important units; this effect is further explored in SI.

This internal organization, in which the network relies on a small number of important units for most of its accuracy with respect to a single output class, is seen across all classes. Figure 2d repeats the same experiment for each of the 365 scene classes. Removing the 20 most important `conv5_3` units for each class reduces single-class accuracy to 53.0% on average, near chance

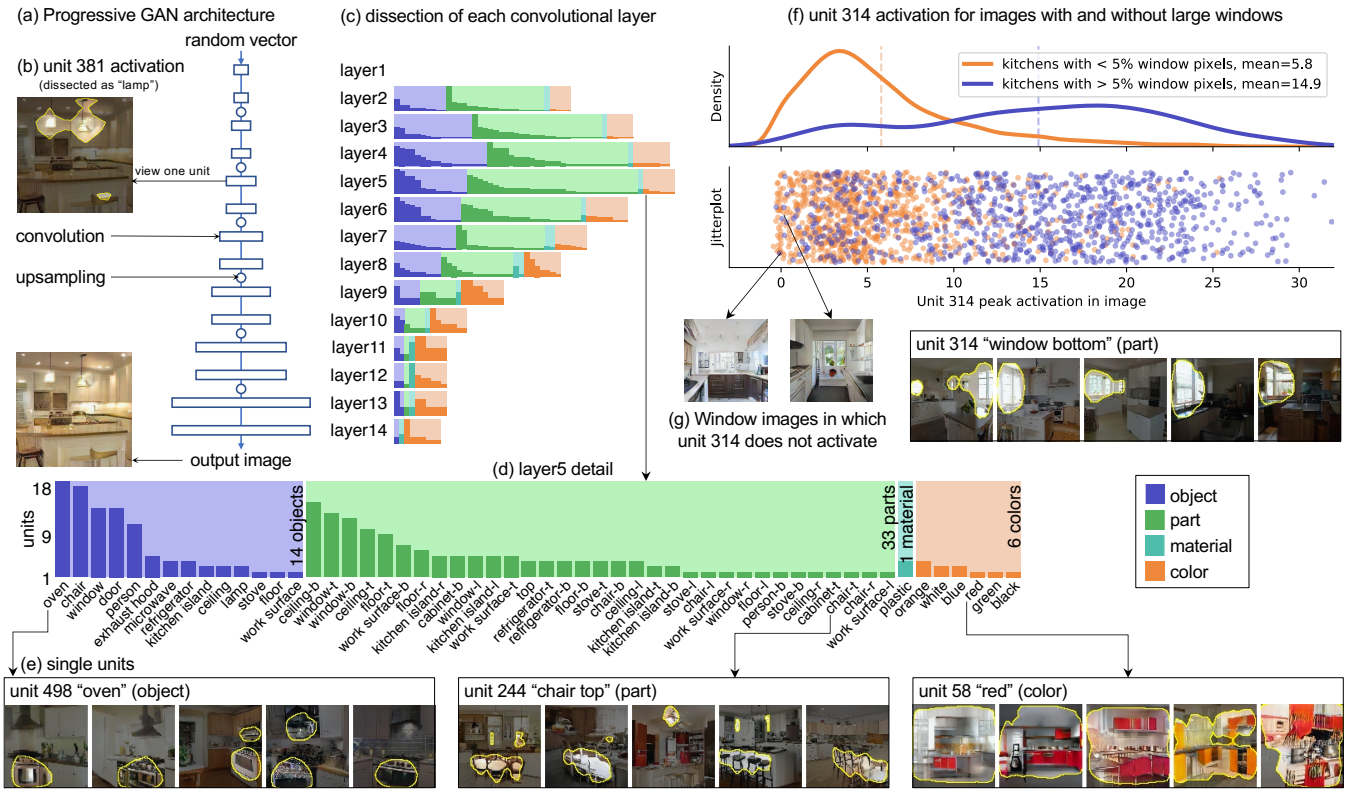

Fig. 3. The emergence of object- and part-specific units within a Progressive GAN generator'(19). (a) The analyzed Progressive GAN consists of 15 convolutional layers that transform a random input vector into a synthesized image of a kitchen. (b) A single filter is visualized as the region of the output image where the filter activates beyond its top 1% quantile level; note that the filters are all precursors to the output. (c) Dissecting all the layers of the network shows a peak in object-specific units at `layer5` of the network. (d) A detailed examination of `layer5` shows more part-specific units than objects, and many visual concepts corresponding to multiple units. (e) Units do not correspond to exact pixel patterns: a wide range of visual appearances for ovens and chairs are generated when an oven or chair part unit are activated. (f) When a unit specific to window parts is tested as a classifier, on average the unit activates more strongly on generated images that contain large windows than images that do not. The jitter plot shows the peak activation of unit 314 on 800 generated images that have windows larger than 5% of the image area as estimated by a segmentation algorithm, and 800 generated images that do not. (g) Some counterexamples: images for which unit 314 does not activate but where windows are synthesized nevertheless.

levels. In contrast, removing the 492 least important units only reduces single-class accuracy by an average of 3.6%, just a slight reduction. We conclude that the emergent object detection done by units of `conv5_3` is not spurious: each unit is important to a specific set of classes, and the object detectors can be interpreted as decomposing the network's classification of individual scene classes into simpler sub-problems.

Why do some units match interpretable concepts so well while other units do not? The data in Figure 2e show that the most interpretable units are those that are important to many different output classes. Units that are important to only one class (or none) are less interpretable, measured by IoU. We further find that important units are predominantly positively correlated with their associated classes, and different combinations of units provide support for each class. Measurements of unit-class correlations and examples of overlapping combinations of important units are detailed in SI.

Does the emergence of interpretable units such as airplane, snow, and tree detectors depend on having training set labels that divide the visual world into hundreds of scene classes? Perhaps the taxonomy of scenes encodes distinctions that are necessary to learn about objects. Or is it possible for a network to infer such concepts from the visual data itself? To investigate this question, we next conduct a similar set of experiments on networks trained to solve unsupervised tasks.

Emergence of Object Detectors in a GAN. A generative adversarial network (GAN) learns to synthesize random realistic images that mimic the distribution of real images in a training set (14). Architecturally, a trained GAN generator is the reverse of a classifier, producing a realistic image from a random input latent vector. Unlike classification, it is an unsupervised setting: no human annotations are provided to a GAN, so the network must learn the structure of the images by itself.

Remarkably, GANs have been observed to learn global semantics of an image: for example, interpolating between latent vectors can smoothly transform the layout of a room (40) or change the texture of an object (41). We wish to understand whether the GAN also learns hierarchical structure, for example, if it learns to decompose the generation of a scene into meaningful parts.

We test a Progressive GAN architecture (19) trained to imitate LSUN kitchen images (42). This network architecture consists of 15 convolutional layers, as shown in Figure 3a. Given a 512-dimensional vector sampled from a multivariate Gaussian distribution, the network produces a 256×256 realistic image after processing the data through the 15 layers. As with a classifier network, each unit is visualized by showing the regions where the filter activates above its top 1% quantile level, as shown in Figure 3b. Importantly, causality in a generator flows in the opposite direction as a classifier: when unit 381 activates on lamp shades in an image, it is not detecting objects in the image, because the

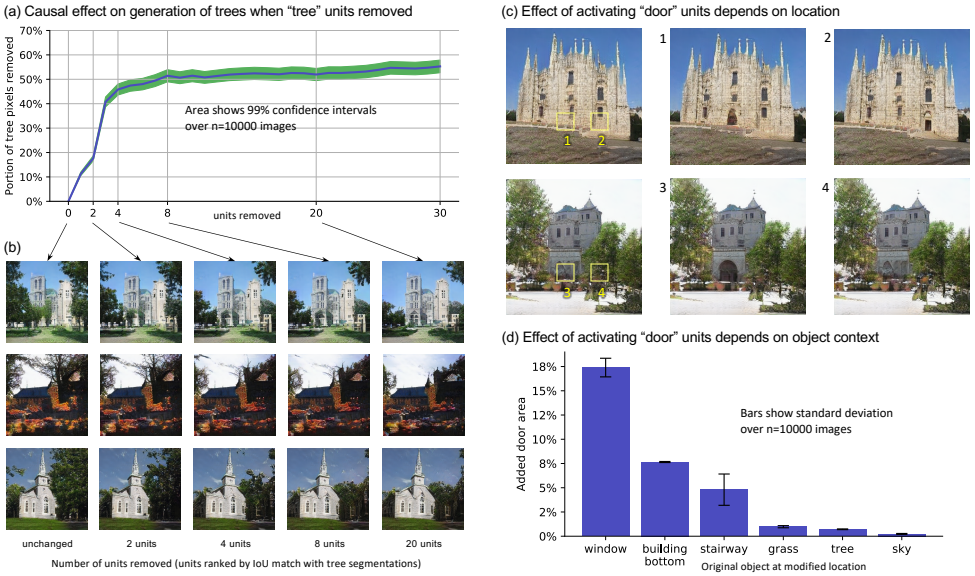

Fig. 4. The causal effect of altering units within a GAN generator. (a) When successively larger sets of units are removed from a GAN trained to generate outdoor church scenes, the tree area of the generated images is reduced. Removing 20 tree units removes more than half the generated tree pixels from the output. (b) Qualitative results: removing tree units affects trees while leaving other objects intact. Building parts that were previously occluded by trees are rendered as if revealing the objects that were behind the trees. (c) Doors can be added to buildings by activating 20 door units. The location, shape, size, and style of the rendered door depends on the location of the activated units. The same activation levels produce different doors, or no door at all (case 4) depending on locations. (d) Similar context dependence can be seen quantitatively: doors can be added in reasonable locations such as at the location of a window, but not in abnormal locations such as on a tree or in the sky.

filter activation occurs *before* the image is generated. Instead, the unit is part of the computation that ultimately renders the objects.

To identify the location of units in the network that are associated with object classes, we apply network dissection to the units of every layer of the network. In this experiment, the reference segmentation models and thresholds used are the same as those used to analyze the VGG-16 classifier. However, instead of analyzing agreement with objects that appear in the input data, we analyze agreement with segmented objects found in the generated output images. As shown in Figure 3c, the largest number of emergent concept units do not appear at the edge of the network as we saw in the classifier, but in the middle: the `layer5` has units that match the largest number of distinct object and part classes.

Figure 3d shows each object, part, material, and color that matches a unit in `layer5` with IoU $> 4\%$. This layer contains 19 object-specific units, 41 units that match object parts, one material, and six color units. As seen in the classification network, visual concepts such as 'oven' and 'chair' match many units. Different from the classifier, more object parts are matched than whole objects.

In Figure 3d, individual units show a wide range of visual diversity: the units do not appear to rigidly match a specific pixel pattern, but rather different appearances for a particular class, for example, various styles of ovens, or different colors and shapes of kitchen stools.

In Figure 3f, we apply the window-specific unit 314 as an image classifier. We find a strong gap between the activation of the unit when a large window is generated and when no large window is generated. Furthermore, a simple threshold (peak activation > 8.03) can achieve a 78.2% accuracy in predicting whether the generated image will have a large window or not. Nevertheless, the distribution density curve reveals that images that contain large windows can be often generated without activating unit 314. Two such samples are shown in Figure 3g. These examples suggest that other units could potentially synthesize windows.

Role of Units in a GAN. The correlations between units and generated object classes are suggestive, but they do not prove that the units that correlate with an object class actually cause the generator to render instances of the object class. To understand the causal role of a unit in a GAN generator, we test the output of the generator when sets of units are directly removed or activated.

We first remove successively larger sets of tree units from a Progressive GAN (19) trained on LSUN church scenes (42). We rank units in `layer4` according to $\text{IoU}_{u,\text{tree}}$ to identify the most tree-specific units. When successively larger sets of these tree units are removed from the network, the GAN generates images with fewer and smaller trees (Figure 4a). Removing the 20 most tree-specific units reduces the number of tree pixels in the generated output by 53.3%, as measured over 10,000 randomly generated images.

When tree-specific units are removed, the generated images continue to look similarly realistic. Although fewer and smaller trees are generated, other objects such as buildings are unchanged. Remarkably, parts of buildings that were occluded by trees are hallucinated, as if removing the trees reveals the walls and windows behind them (Figure 4b). The generator appears to have computed more details than are necessary to render the final output; the details of a building that are hidden behind a tree can only be revealed by suppressing the generation of the tree. The appearance of such hidden details strongly suggests that the GAN is learning a structured statistical model of the scene that extends beyond a flat summarization of visible pixel patterns.

Units can also be forced on to insert new objects into a generated scene. We use $\text{IoU}_{u,\text{door}}$ to find the 20 most door-specific units identified in `layer4` of the same outdoor church GAN. At tested locations, the activations for this set of 20 units are all forced to their high t_u value. Figure 4c shows the effect of applying this procedure to activate 20 door units at two different locations in two generated images. Although the same intervention is applied to all four cases, the doors obtained in each situation is different: In cases 1-3, the newly synthesized door has a size, style, and location that is appropriate to the scene context. In case 4, where door units are activated on a tree, no new door is added to the image.

Figure 4d quantifies the context-sensitivity of activating door units in different locations. In 10,000 randomly generated images, the same 20-door-unit activation is tested at every featuremap location, and the number of newly synthesized door pixels is evaluated using a segmentation algorithm. Doors can be easily added in some locations, such as in buildings and especially on top of an

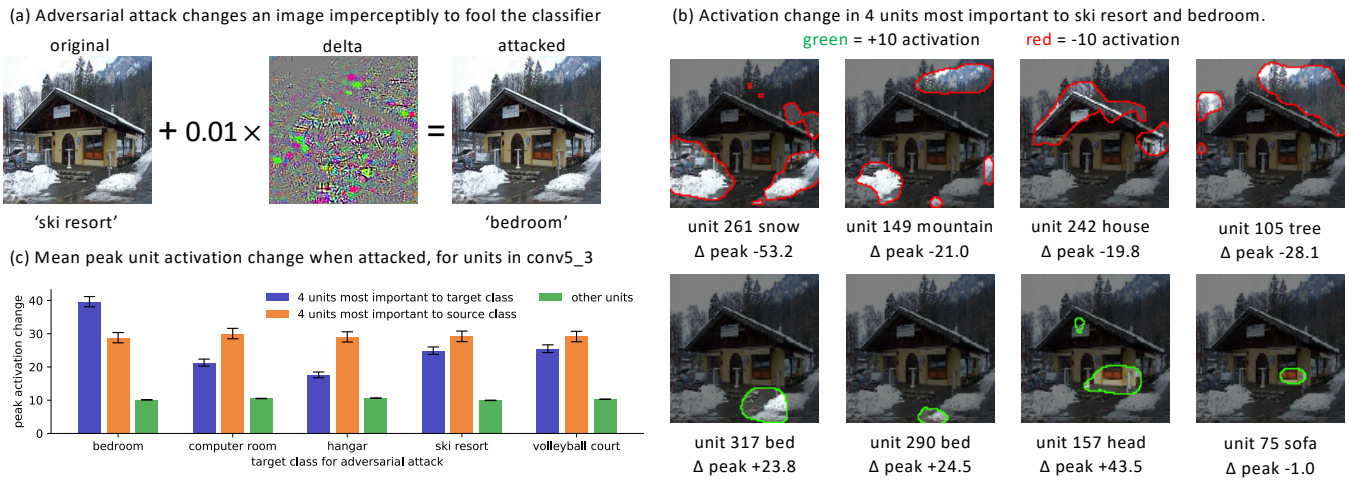

Fig. 5. Application: visualizing an adversarial attack. (a) the test image is correctly labeled as a ski resort, but when an adversarial perturbation is added, the visually indistinguishable result is classified as a bedroom. (b) Visualization of the attack on the four most important units to the ski resort class and the four units most important to the bedroom class. Areas of maximum increase and decrease are shown; Δpeak indicates the change in the peak activation level for the unit. (c) Over 1000 images attacked to misclassify images to various incorrect target classes, the units that are changed most are those that dissection has identified as most important to the source and target classes. Mean absolute value change in peak unit activation is graphed, with 99% confidence intervals shown.

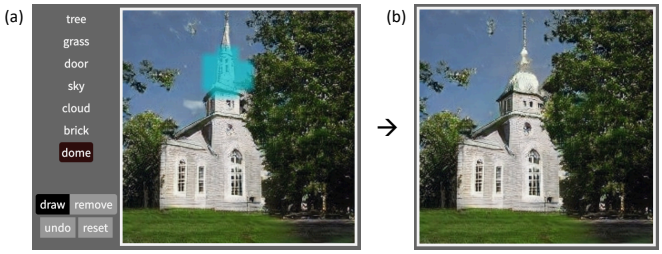

Fig. 6. Application: painting by manipulating GAN neurons. (a) An interactive interface allows a user to choose several high-level semantic visual concepts and paint them on to an image. Each concept corresponds to 20 units in the GAN. (b) After the user adds a dome in the specified location, the result is a modified image in which a dome has been added in place of the original steeple. Once the user's high-level intent has been expressed by changing 20 dome units, the generator automatically handles the pixel-level details of how to fit together objects to keep the output scene realistic.

existing window, but it is nearly impossible to add a door into trees or in the sky.

By learning to solve the unsupervised image generation problem, a GAN has learned units for emergent objects such as doors and trees. It has also learned a computational structure over those units that prevents it from rendering nonsensical output such as a door in the sky, or a door in a tree.

Applications

We now turn to two applications enabled by our understanding of the role of units: understanding attacks on a classifier, and interactively editing a photo by activating units of a GAN.

Analyzing Adversarial Attack of a Classifier. The sensitivity of image classifiers to adversarial attacks is an active research area (43–46). To visualize and understand how an attack works, we can examine the effects on important object detector units. In Figure 5a, a correctly classified 'ski resort' image is attacked to the target 'bedroom' by the C & W optimization method (45, 47). The adversarial algorithm computes a small perturbation which, when added to the original, results in a misclassified image that is visually indistinguishable from the original image. To understand how the attack works, we examine the four most important units to the ski resort class and the four most important units to the bedroom class. Figure 5b visualizes changes in the activations for these units between the original image and the adversarial image. This reveals that the attack has fooled the network by reducing detection of snow, mountain, house, and tree objects, and by increasing activations of detectors for beds, person heads, and sofas in locations where those objects do not actually exist in the image. Figure 5c shows that, across many images and classes, the units that are most changed by an attack are the few units that are important to a class.

Semantic Paint using a GAN. Understanding the roles of units within a network allows us to create a human interface for controlling the network via direct manipulation of its units. We apply this method to a GAN to create an interactive painting application. Instead of painting with a palette of colors, the application allows painting with a palette of high-level object concepts. Each concept is associated with 20 units that maximize $\text{IoU}_{u,c}$ for the concept u. Figure 6a shows our interactive interface. When a user adds brush strokes with a concept, the units for the concept are activated (if the user is drawing) or zeroed (if the user is erasing). Figure 6b shows typical results after the user adds an object to the image. The GAN deals with the pixel-level details of how to add objects while keeping the scene reasonable and realistic. Multiple changes in a scene can be composed for creative effects; movies of image editing demos are included in SI; online demos are also available at the website http://gandissect.csail.mit.edu.

Discussion

Simple measures of performance, such as classification accuracy, do not reveal *how* a network solves its task: good performance can be achieved by networks that have differing sensitivities to shapes, textures, or perturbations (34, 48).

To develop an improved understanding of how a network works, we have presented a way to analyze the roles of individual network units. In a classifier, the units reveal how the network decomposes the recognition of specific scene classes into particular visual concepts that are important to each scene class. And within a

generator, the behavior of the units reveals contextual relationships that the model enforces between classes of objects in a scene.

Network dissection relies on the emergence of disentangled, human-interpretable units during training. We have seen that many such interpretable units appear in state-of-the-art models, both supervised and unsupervised. How to train better disentangled models is an open problem that is the subject of ongoing efforts (49–52).

We conclude that a systematic analysis of individual units can yield insights about the black box internals of deep networks. By observing and manipulating units of a deep network, it is possible to understand the structure of the knowledge that the network has learned, and to build systems that help humans interact with these powerful models.

Materials and Methods

Data sets. Places365 (53) consists of 1.80 million photographic images, each labeled with one of 365 scene classes. The dataset also includes 36,500 labeled validation images (100 per class) that are not used for training. Imagenet (35) consists of 1.28 million photographic images, each focused on a single main object and labeled with one of 1,000 object classes. LSUN is a dataset with a large number of 256×256 images in a few classes (42). LSUN kitchens consists of 2.21 million indoor kitchen photographs, and LSUN outdoor churches consists of 1.26 million photographs of church building exteriors. Recognizable people in dataset images have been anonymized by pixelating faces in visualizations.

Tested Networks. We analyze the VGG-16 classifier (13) trained by the Places365 authors (32) to classify Places365 images. The network achieves classification accuracy of 53.3% on the held-out validation set (chance is 0.27%). The 13 convolutional layers of VGG-16 are divided into 5 groups. The layers in the first group contain 32 units that process image data at the full 224×224 resolution; at each successive group the feature depth is doubled and the feature maps are pooled to halve the resolution, so that at the final stage that includes `conv5_1` and `conv5_3` the layers contain 512 units at 14×14 resolution. The GAN models we analyze are trained by the Progressive GAN authors (19). The models are configured to generate 256×256 output images using 15 convolutional layers divided into 8 groups, starting with 512 units in each layer at 4×4 resolution, and doubling resolution at each successive group, so that `layer4` has 8×8 resolution and 512 units and `layer5` has 16×16 resolution and 512 units. Unit depth is halved in each group after `layer6`, so that the 14th layer has 32 units and 256×256 resolution. The 15th layer (not pictured in Figure 3a) produces a 3-channel RGB image.

Reference Segmentation. To locate human-interpretable visual concepts within large-scale data sets of images, we use the Unified Perceptual Parsing image segmentation network (33) trained on the ADE20K scene dataset (53) and an assignment of rgb values to color names (54). The segmentation algorithm achieves mean IoU of 23.4% on objects, 28.8% on parts, and 54.2% on materials. To further identify units that specialize in object parts, we expand each object class into four additional object part classes which denote the top, bottom, left, or right half of the bounding box of a connected component. Our reference segmentation algorithm can detect 335 object classes, 1452 object parts, 25 materials, and 11 colors.

Data Availability. The code, trained model weights and data sets needed to reproduce the results in this paper are public and available for download at GitHub at https://github.com/davidbau/dissect and at the project website https://dissect.csail.mit.edu/.

Acknowledgments

We are indebted to Aditya Khosla, Aude Oliva, William Peebles, Jonas Wulff, Joshua B. Tenenbaum, and William T. Freeman for their advice and collaboration. And we are grateful for the support of the MIT-IBM Watson AI Lab, the DARPA XAI program FA8750-18-C0004, NSF 1524817 on Advancing Visual Recognition with Feature Visualizations, NSF BIGDATA 1447476, Grant RTI2018-095232-B-C22 from the Spanish Ministry of Science, Innovation and Universities to AL, Early Career Scheme (ECS) of Hong Kong (No.24206219) to BZ, and a hardware donation from NVIDIA.

1 Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A (2015) Object detectors emerge in deep scene cnns in *ICLR*.
2 Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks in *ECCV*.
3 Mahendran A, Vedaldi A (2015) Understanding deep image representations by inverting them in *CVPR*.
4 Olah C, et al. (2018) The building blocks of interpretability. *Distill* 3(3):e10.
5 Bau A, et al. (2018) Identifying and controlling important neurons in neural machine translation in *NeurIPS*.
6 Karpathy A, Johnson J, Fei-Fei L (2016) Visualizing and understanding recurrent networks in *ICLR*.
7 Radford A, Jozefowicz R, Sutskever I (2017) Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.
8 Bengio Y, Courville A, Vincent P (2013) Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.
9 Zhou B, Bau D, Oliva A, Torralba A (2018) Interpreting deep visual representations via network dissection. *PAMI*.
10 Bau D, et al. (2019) Gan dissection: Visualizing and understanding generative adversarial networks in *ICLR*.
11 LeCun Y, Bengio Y, , et al. (1995) Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.
12 Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks in *NeurIPS*. pp. 1097–1105.
13 Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition in *ICLR*.
14 Goodfellow I, et al. (2014) Generative adversarial nets in *NeurIPS*.
15 Vinyals O, Toshev A, Bengio S, Erhan D (2015) Show and tell: A neural image caption generator in *CVPR*. pp. 3156–3164.
16 He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition in *CVPR*.
17 Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks in *CVPR*.
18 Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks in *ICCV*.
19 Karras T, Aila T, Laine S, Lehtinen J (2018) Progressive growing of gans for improved quality, stability, and variation in *ICLR*.
20 Bach S, et al. (2015) On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10(7).
21 Zhou T, Krahenbuhl P, Aubry M, Huang Q, Efros AA (2016) Learning dense correspondence via 3d-guided cycle consistency in *CVPR*.
22 Fong RC, Vedaldi A (2017) Interpretable explanations of black boxes by meaningful perturbation in *ICCV*. pp. 3429–3437.
23 Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions in *NeurIPS*.
24 Selvaraju RR, et al. (2017) Grad-cam: Visual explanations from deep networks via gradient-based localization. in *ICCV*.
25 Sundararajan M, Taly A, Yan Q (2017) Axiomatic attribution for deep networks in *ICML*.
26 Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M (2017) Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.
27 Petsiuk V, Das A, Saenko K (2018) Rise: Randomized input sampling for explanation of black-box models in *BMVC*.
28 Ribeiro MT, Singh S, Guestrin C (2016) Why should i trust you?: Explaining the predictions of any classifier in *SIGKDD*. (ACM), pp. 1135–1144.
29 Kim B, Gilmer J, Viegas F, Erlingsson U, Wattenberg M (2017) Tcav: Relative concept importance testing with linear concept activation vectors. *arXiv preprint arXiv:1711.11279*.
30 Koul A, Fern A, Greydanus S (2019) Learning finite state representations of recurrent policy networks in *ICLR*.
31 Hendricks LA, et al. (2016) Generating visual explanations in *ECCV*. (Springer), pp. 3–19.
32 Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A (2014) Learning deep features for scene recognition using places database in *NeurIPS*.
33 Xiao T, Liu Y, Zhou B, Jiang Y, Sun J (2018) Unified perceptual parsing for scene understanding in *ECCV*.
34 Geirhos R, et al. (2018) Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.
35 Deng J, et al. (2009) Imagenet: A large-scale hierarchical image database in *CVPR*.

36 Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks in *NeurIPS*. pp. 2074–2082.

37 Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2017) Pruning filters for efficient convnets in *ICLR*.

38 Morcos AS, Barrett DG, Rabinowitz NC, Botvinick M (2018) On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*.

39 Zhou B, Sun Y, Bau D, Torralba A (2018) Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*.

40 Radford A, Metz L, Chintala S (2016) Unsupervised representation learning with deep convolutional generative adversarial networks in *ICLR*.

41 Zhu JY, Krähenbühl P, Shechtman E, Efros AA (2016) Generative visual manipulation on the natural image manifold in *ECCV*.

42 Yu F, et al. (2015) Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

43 Szegedy C, et al. (2014) Intriguing properties of neural networks in *ICLR*.

44 Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples in *ICLR*.

45 Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks in *2017 IEEE Symposium on Security and Privacy (SP)*. (IEEE), pp. 39–57.

46 Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks in *ICLR*.

47 Rauber J, Brendel W, Bethge M (2017) Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*.

48 Ilyas A, et al. (2019) Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*.

49 Chen X, et al. (2016) Infogan: Interpretable representation learning by information maximizing generative adversarial nets in *NeurIPS*. pp. 2172–2180.

50 Higgins I, et al. (2017) beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR* 2(5):6.

51 Zhang Q, Nian Wu Y, Zhu SC (2018) Interpretable convolutional neural networks in *CVPR*.

52 Achille A, Soatto S (2018) Emergence of invariance and disentanglement in deep representations. *JMLR* 19(1):1947–1980.

53 Zhou B, et al. (2017) Scene parsing through ade20k dataset in *CVPR*.

54 Van De Weijer J, Schmid C, Verbeek J, Larlus D (2009) Learning color names for real-world applications. *IEEE Transactions on Image Processing* 18(7):1512–1523.