4
Understanding Neural Networks via Feature Visualization: A Survey

Anh Nguyen[1(✉)], Jason Yosinski[2], and Jeff Clune[2,3]

[1] Auburn University, Auburn, AL, USA
anhnguyen@auburn.edu
[2] Uber AI Labs, San Francisco, CA, USA
yosinski@uber.com
[3] University of Wyoming, Laramie, WY, USA
jeffclune@uwyo.edu

Abstract. A neuroscience method to understanding the brain is to find and study the *preferred stimuli* that highly activate an individual cell or groups of cells. Recent advances in machine learning enable a family of methods to synthesize preferred stimuli that cause a neuron in an artificial or biological brain to fire strongly. Those methods are known as Activation Maximization (AM) [10] or Feature Visualization via Optimization. In this chapter, we (1) review existing AM techniques in the literature; (2) discuss a probabilistic interpretation for AM; and (3) review the applications of AM in debugging and explaining networks.

Keywords: Neural networks · Feature visualization · Activation Maximization · Generator network · Generative models · Optimization

4.1 Introduction

Understanding the human brain has been a long-standing quest in human history. One path to understanding the brain is to study what each neuron[1] codes for [17], or what information its firing represents. In the classic 1950's experiment, Hubel and Wiesel studied a cat's brain by showing the subject different images on a screen while recording the neural firings in the cat's primary visual cortex (Fig. 4.1). Among a variety of test images, the researchers found *oriented edges* to cause high responses in one specific cell [14]. That cell is referred to as an *edge detector* and such images are called its *preferred stimuli*. The same technique later enabled scientists to discover fundamental findings of how neurons along the visual pathway detect increasingly complex patterns: from circles, edges to faces and high-level concepts such as one's grandmother [3] or specific celebrities like the actress Halle Berry [37].

[1] In this chapter, "neuron", "cell", "unit", and "feature" are used interchangeably.

W. Samek et al. (Eds.): Explainable AI, LNAI 11700, pp. 55–76, 2019.
https://doi.org/10.1007/978-3-030-28954-6_4

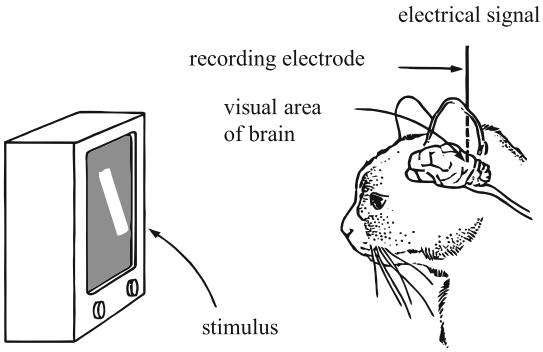


Fig. 4.1. In the classic neuroscience experiment, Hubel and Wiesel discovered a cat's visual cortex neuron (right) that fires strongly and selectively for a light bar (left) when it is in certain positions and orientations [14].

Similarly, in machine learning (ML), visually inspecting the preferred stimuli of a unit can shed more light into what the neuron is doing [48,49]. An intuitive approach is to find such preferred inputs from an existing, large image collection e.g. the training or test set [49]. However, that method may have undesired properties. First, it requires testing each neuron on a large image set. Second, in such a dataset, many informative images that would activate the unit may not exist because the image space is vast and neural behaviors can be complex [28]. Third, it is often ambiguous which visual features in an image are causing the neuron to fire e.g. if a unit is activated by a picture of a bird on a tree branch, it is unclear if the unit "cares about" the bird or the branch (Fig. 4.13b). Fourth, it is not trivial how to extract a holistic description of what a neuron is for from the typically large set of stimuli preferred by a neuron.

A common practice is to study the top 9 highest activating images for a unit [48,49]; however, the top-9 set may reflect only one among many types of features that are preferred by a unit [29].

Instead of finding real images from an existing dataset, one can synthesize the visual stimuli from scratch [10,25,27,29,32,42,46]. The synthesis approach offers multiple advantages: (1) given a strong image prior, one may synthesize (i.e. reconstruct) stimuli without the need to access the target model's training set, which may not be available in practice (see Sect. 4.5); (2) more control over the types and contents of images to synthesize, which helps shed light on more controlled research experiments.

Activation Maximization. Let θ be the parameters of a classifier that maps an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ (that has C color channels, each of which is W pixels wide and H pixels high) onto a probability distribution over the output classes. Finding an image $\mathbf{x}$ that maximizes the activation $a_i^l(\theta, \mathbf{x})$ of a neuron indexed i in a given layer l of the classifier network can be formulated as an optimization problem:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}}(a_i^l(\theta, \mathbf{x})) \tag{4.1}$$

This problem was introduced as *activation maximization*[2] (AM) by Erhan, Bengio and others [10]. Here, $a_i^l(.)$ returns the activation value of a *single* unit as in many previous works [27–29]; however, it can be extended to return any neural response $a(.)$ that we wish to study e.g. activating a group of neurons [24,26,33]. The remarkable DeepDream visualizations [24] were created by running AM to activate all the units across a given layer simultaneously. In this chapter, we will write $a(.)$ instead of $a_i^l(.)$ when the exact indices l, i can be omitted for generality.

AM is a non-convex optimization problem for which one can attempt to find a local minimum via gradient-based [44] or non-gradient methods [30]. In *post-hoc* interpretability [23], we often assume access to the parameters and architecture of the network being studied. In this case, a simple approach is to perform gradient ascent [10,27,31,48] with an update rule such as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon_1 \frac{\partial a(\theta, \mathbf{x}_t)}{\partial \mathbf{x}_t} \tag{4.2}$$

That is, starting from a random initialization $\mathbf{x}_0$ (here, a random image), we iteratively take steps in the input space following the gradient of $a(\theta, \mathbf{x})$ to find an input $\mathbf{x}$ that highly activates a given unit. ϵ_1 is the step size and is chosen empirically.

Note that this gradient ascent process is similar to the gradient descent process used to train neural networks via backpropagation [39], except that here we are optimizing the network input instead of the network parameters θ, which are frozen.[3] We may stop the optimization when the neural activation has reached a desired threshold or a certain number of steps has passed.

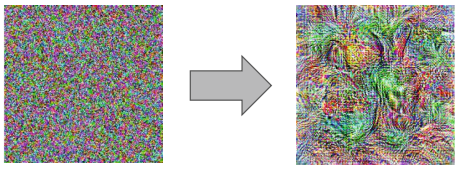
In practice, synthesizing an image from scratch to maximize the activation alone (i.e. an unconstrained optimization problem) often yields uninterpretable images [28]. In a high-dimensional image space, we often find *rubbish* examples (also known as *fooling* examples [28]) e.g. patterns of high-frequency noise that look like nothing but that highly activate a given unit (Fig. 4.2).

In a related way, if starting AM optimization from a real image (instead of a random one), we may easily encounter *adversarial* examples [44] e.g. an image that is slightly different from the starting image (e.g. of a school bus), but that a network would give an entirely different label e.g. "ostrich" [44]. Those early AM visualizations [28,44] revealed huge security and reliability concerns with machine learning applications and informed a plethora of follow-up adversarial attack and defense research [1,16].

Networks that We Visualize. Unless otherwise noted, throughout the chapter, we demonstrate AM on CaffeNet, a specific pre-trained model of the well-known AlexNet convnets [18] to perform single-label image classification on the ILSVRC 2012 ImageNet dataset [7,40].

[2] Also sometimes referred to as *feature visualization* [29,32,48]. In this chapter, the phrase "visualize a unit" means "synthesize preferred images for a single neuron".

[3] Therefore, hereafter, we will write $a(x)$ instead of $a(\theta, x)$, omitting θ, for simplicity.



(a) Random initialization

(b) Synthesized rubbish example

Fig. 4.2. Example of activation maximization without image priors. Starting from a random image (a), we iteratively take steps following the gradient to maximize the activation of a given unit, here the "bell pepper" output in CaffeNet [18]. Despite highly activating the unit and being classified as "bell pepper", the image (b) has high frequencies and is not human-recognizable.

4.2 Activation Maximization via Hand-Designed Priors

Examples like those in Fig. 4.2b are not human-recognizable. While the fact that the network responds strongly to such images is intriguing and has strong implications for security, if we cannot interpret the images, it limits our ability to understand what the unit's purpose is. Therefore, we want to constrain the search to be within a distribution of images that we can interpret e.g. photo-realistic images or images that look like those in the training set. That can be accomplished by incorporating *natural image priors* into the objective function, which was found to substantially improve the recognizability of AM images [21, 27,29,32,48]. For example, an image prior may encourage smoothness [21] or penalize pixels of extreme intensity [42]. Such constraints are often incorporated into the AM formulation as a *regularization* term $R(\mathbf{x})$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} (a(\mathbf{x}) - R(\mathbf{x})) \tag{4.3}$$

For example, to encourage the smoothness in AM images, $R : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}$ may compute the total variation (TV) across an image [21]. That is, in each update, we follow the gradients to (1) maximize the neural activation; and (2) minimize the total variation loss:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon_1 \frac{\partial a(\mathbf{x}_t)}{\partial \mathbf{x}_t} - \epsilon_2 \frac{\partial R(\mathbf{x}_t)}{\partial \mathbf{x}_t} \tag{4.4}$$

However, in practice, we do not always compute the analytical gradient $\partial R(\mathbf{x}_t)/\partial \mathbf{x}_t$. Instead, we may define a regularization operator $r : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H \times W \times C}$ (e.g. a Gaussian blur kernel), and map $\mathbf{x}$ to a more regularized (e.g. slightly blurrier as in [48]) version of itself in every step. In this case, the update step becomes:

$$\mathbf{x}_{t+1} = r(\mathbf{x}_t) + \epsilon_1 \frac{\partial a(\mathbf{x}_t)}{\partial \mathbf{x}_t} \tag{4.5}$$

Note that this update form in Eq. 4.5 is strictly more expressive [48], and allows the use of non-differentiable regularizers $r(.)$.

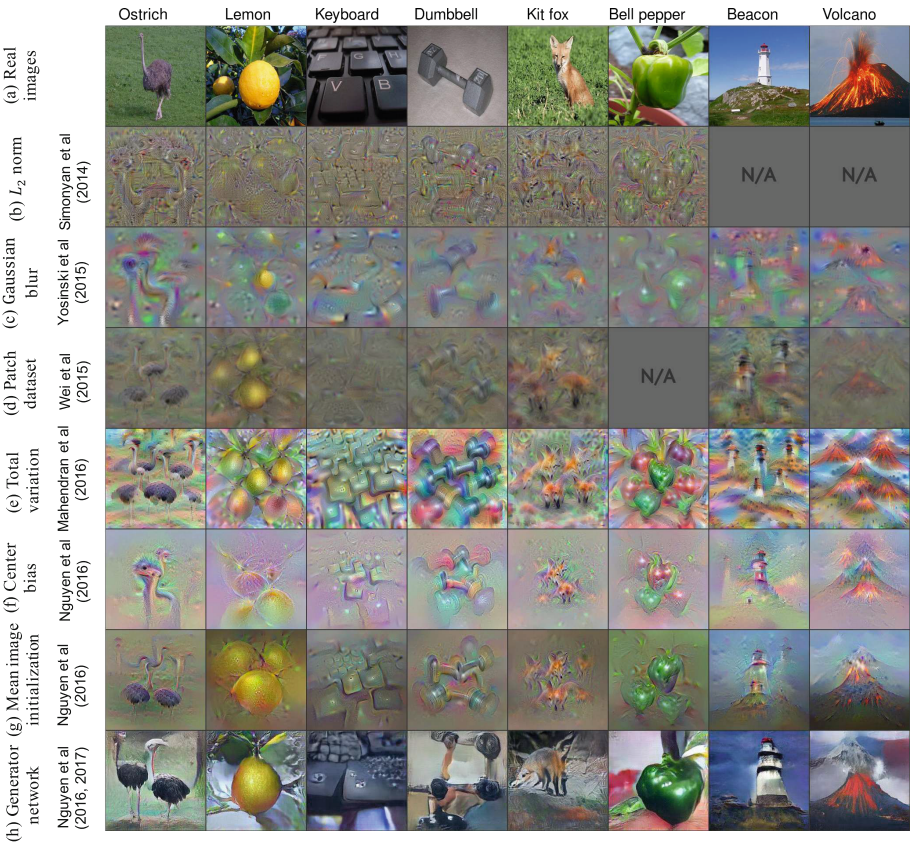


Fig. 4.3. Activation maximization results of seven methods in the literature (b–h), each employing a different image prior (e.g. L_2 norm, Gaussian blur, etc.). Images are synthesized to maximize the output neurons (each corresponding to a class) of the CaffeNet image classifier [18] trained on ImageNet. The categories were not cherry-picked, but instead were selected based on the images available in previous papers [21, 29, 42, 46, 48]. Overall, while it is a subjective judgement, Activation Maximization via Deep Generator Networks method (h) [27] produces images with more natural colors and realistic global structures. Image modified from [27].

Local Statistics. AM images without priors often appear to have high-frequency patterns and unnatural colors (Fig. 4.2b). Many regularizers have been designed in the literature to ameliorate these problems including:

- Penalize extreme-intensity pixels via α-norm [42, 46, 48] (Fig. 4.3b).
- Penalize high-frequency noise (i.e. smoothing) via total variation [21, 29] (Fig. 4.3e), Gaussian blurring [48, 54] (Fig. 4.3c) or a bilateral filter [45].

- Randomly jitter, rotate, or scale the image before each update step to synthesize stimuli that are robust to transformations, which has been shown to make images clearer and more interpretable [24,32].
- Penalize the high frequencies in the *gradient* image $\frac{\partial a(\mathbf{x}_t)}{\partial \mathbf{x}_t}$ (instead of the visualization $\mathbf{x}_t$) via Gaussian blurring [32,54].
- Encourage patch-level color statistics to be more realistic by (1) matching those of real images from a dataset [46] (Fig. 4.3d) or (2) learning a Gaussian mixture model of real patches [24].

While substantially improving the interpretability of images (compared to high-frequency rubbish examples), these methods only effectively attempt to match the *local* statistics of natural images.

Global Structures. Many AM images still lack *global* coherence; for example, an image synthesized to highly activate the "bell pepper" output neuron (Fig. 4.3b–e) may exhibit multiple bell-pepper segments scattered around the same image rather than a single bell pepper. Such stimuli suggest that the network has learned some *local* discriminative features e.g. the shiny, green skin of bell peppers, which are useful for the classification task. However, it raises an interesting question: Did the network ever learn the global structures (e.g. the whole pepper) or only the local discriminative parts? The high-frequency patterns as in Fig. 4.3b–e might also be a consequence of optimization in the image space. That is, when making pixel-wise changes, it is non-trivial to ensure global coherence across the entire image. Instead, it is easy to increase neural activations by simply creating more local discriminative features in the stimulus.

Previous attempts to improve the global coherence include:

- Gradually paint the image by scaling it and alternatively following the gradients from multiple output layers of the network [54].
- Bias the image changes to be near the image center [29] (Fig. 4.3g).
- Initialize optimization from an average image (computed from real training set images) instead of a random one [29] (Fig. 4.3h).

While these methods somewhat improved the global coherence of images (Fig. 4.3g–h), they rely on a variety of heuristics and introduce extra hyperparameters [29,54]. In addition, there is still a large realism gap between the real images and these visualizations (Fig. 4.3a vs. h).

Diversity. A neuron can be multifaceted in that it responds strongly to multiple distinct types of stimuli, i.e. *facets* [29]. That is, higher-level features are more invariant to changes in the input [19,49]. For example, a face-detecting unit in CaffeNet [18] was found to respond to both human and lion faces [48]. Therefore, we wish to uncover different facets via AM in order to have a fuller understanding of a unit.

However, AM optimization starting from different random images often converge to similar results [10,29]—a phenomenon also observed when training neural networks with different initializations [20]. Researchers have proposed different techniques to improve image diversity such as:

- Drop out certain neural paths in the network when performing backpropagation to produce different facets [46].
- Cluster the training set images into groups, and initialize from an average image computed from each group's images [29].
- Maximize the distance (e.g. cosine similarity in the pixel space) between a reference image and the one being synthesized [32].
- Activate two neurons at the same time e.g. activating (bird + apron) and (bird + candles) units would produce two distinct images of *birds* that activate the same *bird* unit [27] (Fig. 4.10).
- Add noise to the image in every update to increase image diversity [26].

While obtaining limited success, these methods also introduce extra hyperparameters and require further investigation. For example, if we enforce two stimuli to be different, exactly how far should they be and in which similarity metric should the difference be measured?

4.3 Activation Maximization via Deep Generator Networks

Much previous AM research were optimizing the preferred stimuli directly in the high-dimensional image space where pixel-wise changes are often slow and uncorrelated, yielding high-frequency visualizations (Fig. 4.3b–e). Instead, Nguyen et al. [27] propose to optimize in the low-dimensional latent space of a deep generator network, which they call Deep Generator Network Activation Maximization (DGN-AM). They train an image *generator* network to take in a highly compressed code and output a synthetic image that looks as close to real images from the ImageNet dataset [40] as possible. To produce an AM image for a given neuron, the authors optimize in the input latent space of the generator so that it outputs an image that activates the unit of interest (Fig. 4.4). Intuitively, DGN-AM restricts the search to only the set of images that can be drawn by the prior and encourages the image updates to be more coherent and correlated compared to pixel-wise changes (where each pixel is modified independently).

Generator Networks. We denote the sub-network of CaffeNet [18] that maps images onto 4096-D fc6 features as an encoder $E : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{4096}$. We train a generator network $G : \mathbb{R}^{4096} \rightarrow \mathbb{R}^{H \times W \times C}$ to invert E i.e. $G(E(\mathbf{x})) \approx \mathbf{x}$. In addition to the reconstruction losses, the generator was trained using the Generative Adversarial Network (GAN) loss [13] to improve the image realism. More training details are in [9,27]. Intuitively, G can be viewed as an artificial *general* "painter" that is capable of painting a variety of different types of images,

given an arbitrary input description (i.e. a latent code or a condition vector). The idea is that G would be able to faithfully portray what a target network has learned, which may be recognizable or unrecognizable patterns to humans.

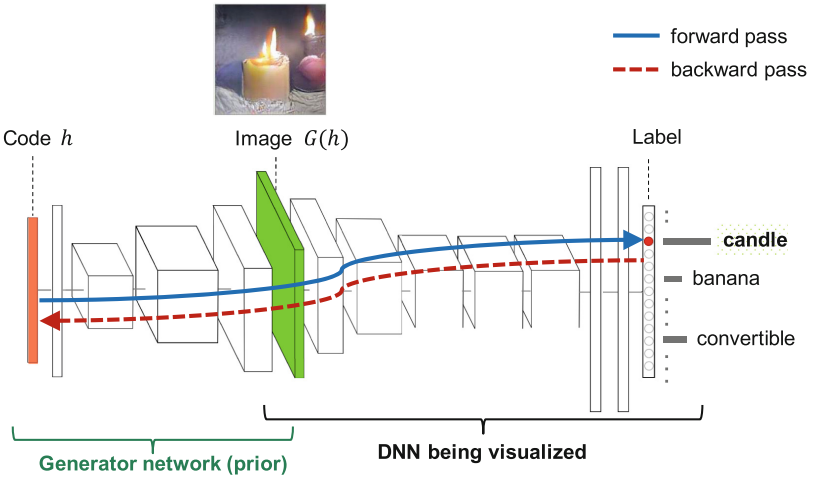


Fig. 4.4. We search for an input code (red bar) of a deep generator network (left) that produces an image (middle) that strongly activates a target neuron (e.g. the "candle" output unit) in a given pre-trained network (right). The iterative optimization procedure involves multiple forward and backward passes through both the generator and the target network being visualized.

Optimizing in the Latent Space. Intuitively, we search in the input code space of the generator G to find a code $\mathbf{h} \in \mathbb{R}^{4096}$ such that the image $G(\mathbf{h})$ maximizes the neural activation $a(G(\mathbf{h}))$ (see Fig. 4.4). The AM problem in Eq. 4.3 now becomes:

$$\mathbf{h}^* = \arg\max_{\mathbf{h}} (a(G(\mathbf{h})) - R(\mathbf{h})) \tag{4.6}$$

That is, we take steps in the latent space following the below update rule:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \epsilon_1 \frac{\partial a(G(\mathbf{h}_t))}{\partial \mathbf{h}_t} - \epsilon_2 \frac{\partial R(\mathbf{h}_t)}{\partial \mathbf{h}_t} \tag{4.7}$$

Note that, here, the regularization term $R(.)$ is on the latent code $\mathbf{h}$ instead of the image $\mathbf{x}$. Nguyen et al. [27] implemented a small amount of L_2 regularization and also clipped the code. These hand-designed regularizers can be replaced by a strong, learned prior for the code [26].

Optimizing in the latent space of a deep generator network showed a great improvement in image quality compared to previous methods that optimize in the pixel space (Fig. 4.5; and Fig. 4.3b–h vs. Fig. 4.3i). However, images synthesized by DGN-AM have limited diversity—they are qualitatively similar to the real top-9 validation images that highest activate a given unit (Fig. 4.6).

Fig. 4.5. Images synthesized from scratch via DGN-AM method [27] to highly activate output neurons in the CaffeNet deep neural network [15], which has learned to classify 1000 categories of ImageNet images. Image from [27].

To improve the image diversity, Nguyen et al. [26] harnessed a learned realism prior for $\mathbf{h}$ via a denoising autoencoder (DAE), and added a small amount of Gaussian noise in every update step to improve image diversity [26]. In addition to an improvement in image diversity, this AM procedure also has a theoretical probabilistic justification, which is discussed in Sect. 4.4.

4.4 Probabilistic Interpretation for Activation Maximization

In this section, we first make a note about the AM objective, and discuss a probabilistically interpretable formulation for AM, which is first proposed in Plug and Play Generative Networks (PPGNs) [26], and then interpret other AM methods under this framework. Intuitively, the AM process can be viewed as sampling from a generative model, which is composed of (1) an image prior and (2) a recognition network that we want to visualize.

4.4.1 Synthesizing Selective Stimuli

We start with a discussion on AM objectives. In the original AM formulation (Eq. 4.1), we only explicitly maximize the activation a_i^l of a unit indexed i in layer l; however, in practice, this objective may surprisingly also increase the activations $a_{j \neq i}^l$ of some other units j in the same layer and even higher than a_i^l [27]. For example, maximizing the output activation for the "hartebeest" class is likely to yield an image that also strongly activates the "impala" unit because these two animals are visually similar [27]. As the result, there is no guarantee that the target unit will be the highest activated across a layer. In that case, the resultant visualization may not portray what is unique about the target unit (l, i).

Instead, we are interested in *selective* stimuli that highly activate only a_i^l, but not $a_{j\neq i}^l$. That is, we wish to maximize a_i^l such that it is the highest single activation across the same layer l. To enforce that selectivity, we can either maximize the softmax or log of softmax of the raw activations across a layer [26,42] where the softmax transformation for unit i across layer l is given as $s_i^l = \exp(a_i^l)/\sum_j \exp(a_j^l)$. Such selective stimuli (1) are more interpretable and preferred in neuroscience [3] because they contain only visual features exclusively for one unit of interest but not others; (2) naturally fit in our probabilistic interpretation discussed below.

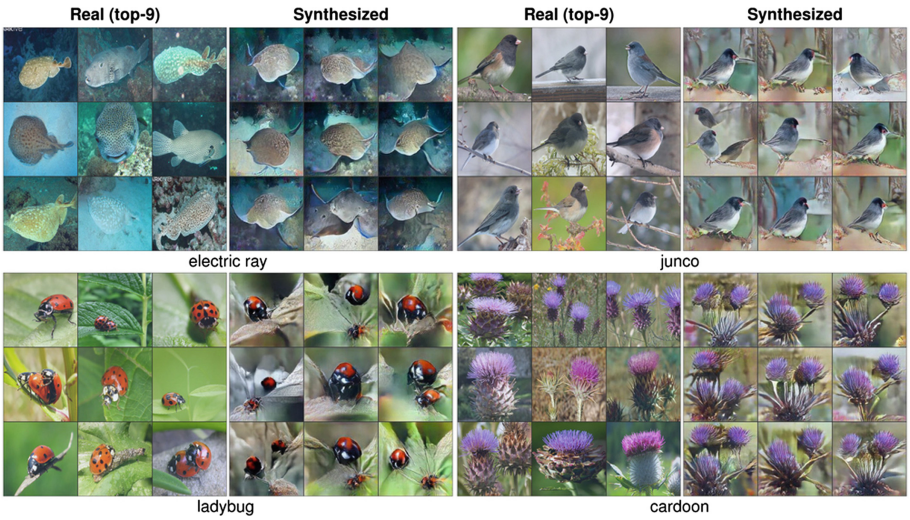


Fig. 4.6. Side-by-side comparison between real and synthetic stimuli synthesized via DGN-AM [27]. For each unit, we show the top 9 validation set images that highest activate a given neuron (left) and 9 synthetic images (right). Note that these synthetic images are of size 227×227 i.e. the input size of CaffeNet [18]. Image from [27].

4.4.2 Probabilistic Framework

Let us assume a joint probability distribution $p(\mathbf{x}, y)$ where $\mathbf{x}$ denotes images, and y is a categorical variable for a given neuron indexed i in layer l. This model can be decomposed into an image density model and an image classifier model:

$$p(\mathbf{x}, y) = p(\mathbf{x})p(y|\mathbf{x}) \tag{4.8}$$

Note that, when l is the output layer of an ImageNet 1000-way classifier [18], y also represents the image category (e.g. "volcano"), and $p(y|\mathbf{x})$ is the classification probability distribution (often modeled via softmax).

We can construct a Metropolis-adjusted Langevin [38] (MALA) sampler for our $p(\mathbf{x}, y)$ model [26]. This variant of MALA [26] does not have the accept/reject step, and uses the following transition operator:[4]

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon_{12} \nabla \log p(\mathbf{x}_t, y) + N(0, \epsilon_3^2) \tag{4.9}$$

Since y is a categorical variable, and chosen to be a fixed neuron y_c outside the sampler, the above update rule can be re-written as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon_{12} \nabla \log p(y = y_c | \mathbf{x}_t) + \epsilon_{12} \nabla \log p(\mathbf{x}_t) + N(0, \epsilon_3^2) \tag{4.10}$$

Decoupling ϵ_{12} into explicit ϵ_1 and ϵ_2 multipliers, and expanding the ∇ into explicit partial derivatives, we arrive at the following update rule:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon_1 \frac{\partial \log p(y = y_c | \mathbf{x}_t)}{\partial \mathbf{x}_t} + \epsilon_2 \frac{\partial \log p(\mathbf{x}_t)}{\partial \mathbf{x}_t} + N(0, \epsilon_3^2) \tag{4.11}$$

An intuitive interpretation of the roles of these three terms is illustrated in Fig. 4.7 and described as follows:

- ϵ_1 term: take a step toward an image that causes the neuron y_c to be the *highest activated* across a layer (Fig. 4.7; red arrow)
- ϵ_2 term: take a step toward a generic, realistic-looking image (Fig. 4.7; blue arrow).
- ϵ_3 term: add a small amount of noise to jump around the search space to encourage image diversity (Fig. 4.7; green arrow).

Maximizing Raw Activations vs. Softmax. Note that the ϵ_1 term in Eq. 4.11 is not the same as the *gradient of raw activation* term in Eq. 4.2. We summarize in Table 4.1 three variants of computing this ϵ_1 gradient term: (1) derivative of logits; (2) derivative of softmax; and (3) derivative of log of softmax. Several previous works empirically reported that maximizing raw, pre-softmax activations a_i^l produces better visualizations than directly maximizing the softmax values s_i^l (Table 4.1a vs. b); however, this observation had not been fully justified [42]. Nguyen et al. [26] found the log of softmax gradient term (1) working well empirically; and (2) theoretically justifiable under the probabilistic framework in Sect. 4.4.2.

We refer readers to [26] for a more complete derivation and discussion of the above MALA sampler. Using the update rule in Eq. 4.11, we will next interpret other AM algorithms in the literature.

[4] We abuse notation slightly in the interest of space and denote as $N(0, \epsilon_3^2)$ a sample from that distribution. The first step size is given as ϵ_{12} in anticipation of later splitting into separate ϵ_1 and ϵ_2 terms.

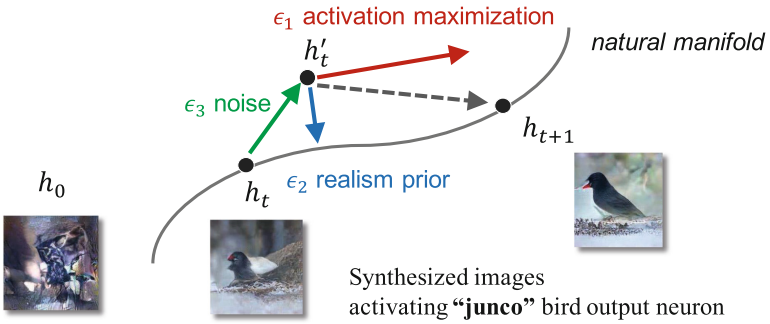


Fig. 4.7. AM can be considered as a sampler, traversing in the natural image manifold. We start from a random initialization h_0. In every step t, we first add a small amount of noise (green arrow), which pushes the sample off the natural-image manifold (h'_t). The gradients toward maximizing activation (red arrow) and more realistic images (blue arrow) pull the noisy h'_t back to the manifold at a new sample h_{t+1}.

Table 4.1. A comparison of derivatives for use in activation maximization methods. (a) has most commonly been used, (b) has worked in the past but with some difficulty, but (c) is correct under the sampler framework in Sect. 4.4.2 and [26].

a. Derivative of raw activations. Worked well in practice [10,27] but may produce *non-selective* stimuli and is not quite the right term under the probabilistic framework in Sect. 4.4.2	$\frac{\partial a_i^l}{\partial x}$
b. Derivative of softmax. Previously avoided due to poor performance [42,48], but poor performance may have been due to ill-conditioned optimization rather than the inclusion of logits from other classes	$\frac{\partial s_i^l}{\partial x} = s_i^l \left(\frac{\partial a_i^l}{\partial x} - \sum_j s_j^l \frac{\partial a_j^l}{\partial x} \right)$
c. Derivative of log of softmax. Correct term under the sampler framework in Sect. 4.4.2. Well-behaved under optimization, perhaps due to the $\frac{\partial a_i^l}{\partial x}$ term untouched by the s_i^l multiplier	$\frac{\partial \log s_i^l}{\partial x} = \frac{\partial \log p(y=y_i \mid x_t)}{\partial x} = \frac{\partial a_i^l}{\partial x} - \frac{\partial}{\partial x} \log \sum_j \exp(a_j^l)$

4.4.3 Interpretation of Previous Algorithms

Here, we consider four representative approaches in light of the probabilistic framework:

1. AM with no priors [10,28,44] (discussed in Sect. 4.1)
2. AM with a Gaussian prior [42,46,48] (discussed in Sect. 4.2)
3. AM with hand-designed priors [21,29,31,42,46,48] (discussed in Sect. 4.2)
4. AM in the latent space of generator networks [26,27] (discussed in Sect. 4.3)

Activation Maximization with No Priors. From Eq. 4.11, if we set $(\epsilon_1, \epsilon_2, \epsilon_3) = (1, 0, 0)$, we obtain a sampler that follows the neuron gradient directly without contributions from a $p(\mathbf{x})$ term or the addition of noise. In a high-dimensional space, this results in adversarial or rubbish images [28,44] (as discussed in Sect. 4.2). We can also interpret the optimization procedure in [28,44] as a sampler with a non-zero ϵ_1 but with a $p(\mathbf{x})$ such that $\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}} = 0$ i.e. a uniform $p(\mathbf{x})$ where all images are equally likely.

Activation Maximization with a Gaussian Prior. To avoid producing high-frequency images [28] that are uninterpretable, several works have used L_2 decay, which can be thought of as a simple zero-mean Gaussian prior over images [42,46,48]. From Eq. 4.11, if we define a Gaussian $p(\mathbf{x})$ centered at the origin (assume the mean image has been subtracted) and set $(\epsilon_1, \epsilon_2, \epsilon_3) = (1, \lambda, 0)$, pulling Gaussian constants into λ, we obtain the following noiseless update rule:

$$\mathbf{x}_{t+1} = (1 - \lambda)\mathbf{x}_t + \frac{\partial \log p(y = y_c|\mathbf{x}_t)}{\partial \mathbf{x}_t} \tag{4.12}$$

The first term decays the current image slightly toward the origin, as appropriate under a Gaussian image prior, and the second term pulls the image toward higher probability regions for the chosen neuron. Here, the second term is computed as the derivative of the log of a softmax transformation of all activations across a layer (see Table 4.1).

Activation Maximization with Hand-Designed Priors. In an effort to outdo the simple Gaussian prior, many works have proposed more creative, hand-designed image priors such as Gaussian blur [48], total variation [21], jitter, rotate, scale [24], and data-driven patch priors [46]. These priors effectively serve as a simple $p(\mathbf{x})$ component in Eq. 4.11. Note that all previous methods considered under this category are noiseless ($\epsilon_3 = 0$).

Activation Maximization in the Latent Space of Generator Networks. To ameliorate the problem of poor mixing in the high-dimensional pixel space [5], several works instead performed optimization in a semantically meaningful, low-dimensional feature space of a generator network [6,26,27,47,53].

That approach can be viewed as re-parameterizing $p(\mathbf{x})$ as $\int_{\mathbf{h}} p(\mathbf{x}|\mathbf{h})p(\mathbf{h})$, and sampling from the joint probability distribution $p(\mathbf{h}, y)$ instead of $p(\mathbf{x}, y)$, treating $\mathbf{x}$ as a deterministic variable. That is, the update rule in Eq. 4.11 is now changed into the below:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \epsilon_1 \frac{\partial \log p(y = y_c|\mathbf{h}_t)}{\partial \mathbf{h}_t} + \epsilon_2 \frac{\partial \log p(\mathbf{h}_t)}{\partial \mathbf{h}_t} + N(0, \epsilon_3^2) \tag{4.13}$$

In this category, DGN-AM [27] follows the above rule with (ϵ_1, ϵ_2, ϵ_3) = (1,1,0).[5] Specifically, we hand-designed a $p(\mathbf{h})$ via clipping and L_2 regularization

[5] $\epsilon_3 = 0$ because noise was not used in DGN-AM [27].

(i.e. a Gaussian prior) to keep the code **h** within a "realistic" range. PPGNs follows exactly the update rule in Eq. 4.13 with a better $p(\mathbf{h})$ prior learned via a denoising autoencoder [26]. PPGNs produce images with better diversity than DGN-AM [26].

4.5 Applications of Activation Maximization

In this section, we review how one may use activation maximization to understand and explain a pre-trained neural network. The results below are specifically generated by DGN-AM [27] and PPGNs [26] where the authors harnessed a general image generator network to synthesize AM images.

Visualize Output Units for New Tasks. We can harness a general learned ImageNet prior to synthesize images for networks trained on a different dataset e.g. MIT Places dataset [50] or UCF-101 activity videos [27] (Figs. 4.5 and 4.8).

Fig. 4.8. Preferred stimuli generated via DGN-AM [27] for output units of a network trained to classify images on the MIT Places dataset [51] (left) and a network trained to classify videos from the UCF-101 dataset (right). The results suggested that the learned ImageNet prior generalizes well to synthesizing images for other datasets.

Visualize Hidden Units. Instead of synthesizing preferred inputs for output neurons (Fig. 4.5), one may apply AM to the hidden units. In a comparison with visualizing *real* image regions that highly activate a unit [50], we found AM images may provide similar but sometimes also complementary evidence suggesting what a unit is for [27] (Fig. 4.9). For example, via DGN-AM, we found that a unit that detects "TV screens" also detects people on TV (Fig. 4.9, unit 106).

Synthesize Preferred Images Activating Multiple Neurons. First, one may synthesize images activating a group of units at the same time to study the interaction between them [27,32]. For example, it might be useful to study how a network distinguishes two related and visually similar concepts such as "impala"

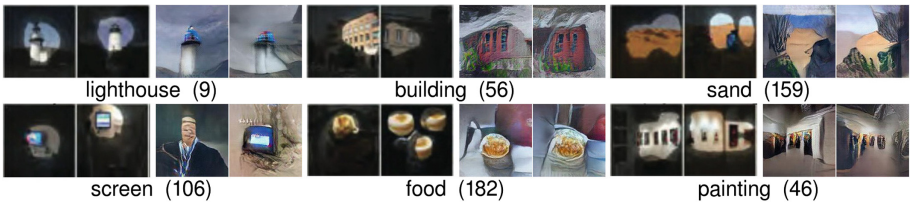


Fig. 4.9. AM images for example hidden units at layer 5 of an CaffeNet [18] trained to classify images of scenes [50]. For each unit: the left two images are masked-out real images, each highlighting a region that highly activates the unit via methods in [50], and humans provide text labels (e.g. "lighthouse") describing the common theme in the highlighted regions. The right two images are AM images, which enable the same conclusion regarding what feature a hidden unit has learned. Figure from [27].

and "hartebeest" animals in ImageNet [7]. One way to do this is to synthesize images that maximize the "impala" neuron's activation but also *minimize* the "hartebeest" neuron's activation. Second, one may reveal different facets of a neuron [29] by activating different pairs of units. That is, activating two units at the same time e.g. (castle + candle); and (piano + candle) would produce two distinct images of candles that activate the same "candle" unit [27] (Fig. 4.10). In addition, this method sometimes also produces interesting, creative art [12,27].

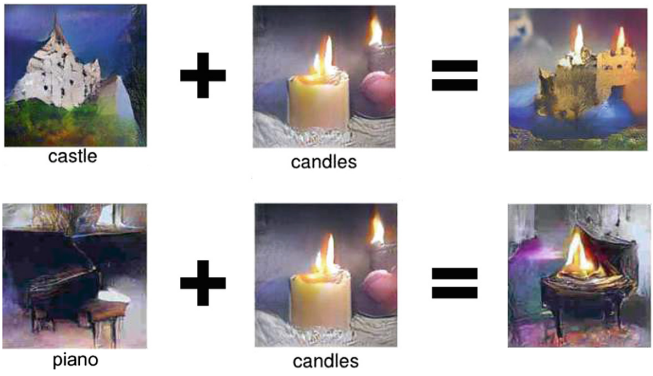


Fig. 4.10. Synthesizing images via DGN-AM [27] to activate both the "castle" and "candles" units of CaffeNet [18] produces an image that resembles a castle on fire (top right). Similarly, "piano" + "candles" produces a candle on a piano (bottom right). Both rightmost images highly activate the "candles" output neuron.

Watch Feature Evolution During Training. We can watch how the features evolved during the training of a target classifier network [27]. Example videos of AM visualizations for sample output and hidden neurons during the training of CaffeNet [15] are at: https://www.youtube.com/watch?v=q4yIwiYH6FQ and https://www.youtube.com/watch?v=G8AtatM1Sts. One may find that features at lower layers tend to converge faster vs. those at higher layers.

Synthesizing Videos. To gain insights into the inner functions of an activity recognition network [43], one can synthesize a single frame (Fig. 4.8; right) or an entire *preferred video*. By synthesizing videos, Nguyen et al. [27] found that a video recognition network (LRCN [8]) classifies videos without paying attention to temporal correlation across video frames. That is, the AM videos[6] appear to be a set of uncorrelated frames of activity e.g. a basketball game. Further tests confirmed that the network produces similar top-1 predicted labels regardless of whether the frames of the *original* UCF-101 videos [43] are randomly shuffled.

Activation Maximization as a Debugging Tool. We discuss here a case study where AM can be used as a debugging tool. Suppose there is a bug in your neural network image classifier implementation that internally and unexpectedly converts all input RGB images (Fig. 4.11a) into BRG images (Fig. 4.11b) before feeding them to the neural network. This bug might be hard to notice by only examining accuracy scores or attribution heatmaps [23]. Instead, AM visualizations could reflect the color space of the images that were fed to the neural network and reveal this bug (Fig. 4.11c).

Synthesize Preferred Images Conditioned on a Sentence. Instead of synthesizing images preferred by output units in an image classifier, we can also synthesize images that cause an image *captioning* network to output a desired sentence (examples in Fig. 4.12).

This reverse-engineering process may uncover interesting insights into the system's behaviors. For example, we discovered an interesting failure of a state-of-the-art image captioner [8] when it declares birds even when there is no bird in an image (Fig. 4.13).

Synthesize Preferred Images Conditioned on a Semantic Segmentation Map. We can extend AM methods to synthesize images with more fine-grained controls of where objects are placed by matching a semantic map output of a segmentation network (Fig. 4.14) or a target spatial feature map of a convolutional layer.

Synthesize Preferred Stimuli for Real, Biological Brains. While this survey aims at visualizing artificial networks, it is also possible to harness our AM techniques to study biological brains. Two teams of Neuroscientists [22,36] have recently been able to reconstruct stimuli for neurons in alive macaques' brains using either the ImageNet PPGN (as discussed in Sect. 4.4) [22] or the DGN-AM (as discussed in Sect. 4.3) [36]. The synthesized images surprisingly resemble monkeys and human nurses that the subject macaque meets frequently [36] or show eyes in neurons previously shown to be tuned for detecting faces [22]. Similar AM frameworks have also been interestingly applied to reconstruct stimuli from EEG or MRI signals of human brains [34,41].

[6] https://www.youtube.com/watch?v=IOYnIK6N5Bg.

(a) Regular ImageNet training images

(b) ImageNet training images converted into the BRG color space

(c) Visualizations of the units that are trained on BRG ImageNet images above (b)

Fig. 4.11. The original ImageNet training set images are in RGB color space (a). We train CaffeNet [18] on their BRG versions (b). The activation maximization images synthesized by DGN-AM [27], faithfully portray the color space of the images, here BRG, where the network was trained on.

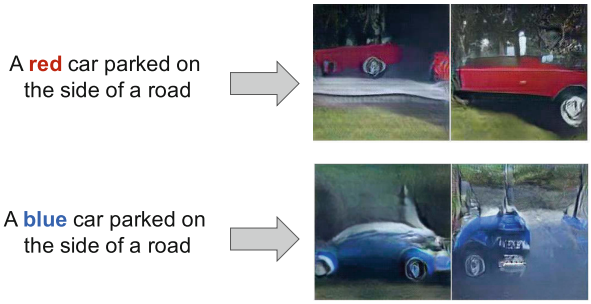


Fig. 4.12. We synthesize input images (right) such that a pre-trained image captioning network (LRCN [8]) outputs the target caption description (left sentences). Each image on the right was produced by starting optimization from a different random initialization.

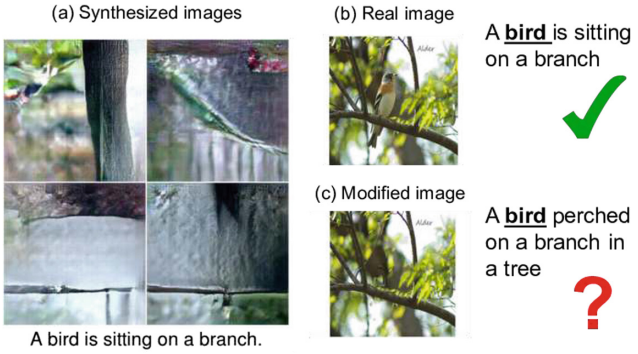


Fig. 4.13. While synthesizing images to cause an image captioning model [8] to output *"A bird is sitting on a branch"* via DGN-AM method [27], we only obtained images of branches or trees that surprisingly has no birds at all (a). Further tests on real MS COCO images revealed that the model [8] outputs correct captions for a test image that has a bird (b), but still insists on the existence of the bird, even when it is manually removed via Adobe Photoshop (c). This suggests the image captioner learned a strong correlation between birds and tree branches—a bias that might exist in the language or image model.

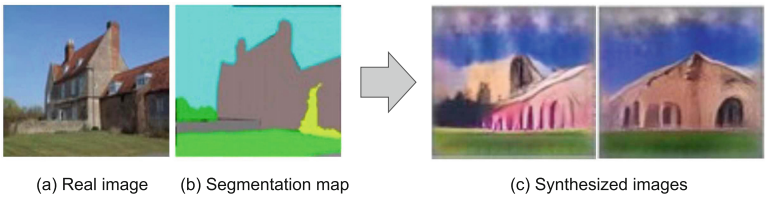


Fig. 4.14. A segmentation network from [52] is capable of producing a semantic segmentation map (b) given an input real image (a). The authors extend the DGN-AM method [27] to synthesize images (c) to match the target segmentation map (b), which specifies a scene with a building on green grass and under a blue sky background. Figure modified from [52].

4.6 Discussion and Conclusion

While activation maximization has proven a useful tool for understanding neural networks, there are still open challenges and opportunities such as:

– One might wish to harness AM to compare and contrast the features learned by different models. That would require a robust, principled AM approach that produces faithful and interpretable visualizations of the learned features for networks trained on different datasets or of different architectures. This is challenging due to two problems: (1) the image prior may not be general enough and may have a bias toward a target network or one dataset over the others; (2) AM optimization on different network architectures, especially of

different depths, often requires different hyper-parameter settings to obtain the best performance.
- It is important for the community to propose rigorous approaches for evaluating AM methods. A powerful image prior may incur a higher risk of producing misleading visualizations—it is unclear whether a synthesized visual feature comes from the image prior or the target network being studied or both. Note that we have investigated that and surprisingly found the DGN-AM prior to be able to generate a wide diversity of images including the non-realistic ones (e.g. blurry, cut-up, and BRG images [27]).
- Concepts in modern deep networks can be highly distributed [4,11,44]; therefore, it might be promising to apply AM to study networks at a different, larger scale than individual neurons, e.g. looking at groups of neurons [33].
- It might be a fruitful direction to combine AM with other tools such as attribution heatmapping [33] or integrate AM into the testbeds for AI applications [35] as we move towards safe, transparent, and fair AI.
- One may also perform AM in the parameter space of a 3D renderer (e.g. modifying the lighting, object geometry or appearances in a 3D scene) that renders a 2D image that strongly activates a unit [2]. AM in a 3D space allows us to synthesize stimuli by varying a controlled factor (e.g. lighting) and thus might offer deeper insights into a model's inner-workings.

Activation maximization techniques enable us to shine light into the black-box neural networks. As this survey shows, improving activation maximization techniques improves our ability to understand deep neural networks. We are excited for what the future holds regarding improved techniques that make neural networks more interpretable and less opaque so we can better understand how deep neural networks do the amazing things that they do.

Acknowledgements. Anh Nguyen is supported by the National Science Foundation under Grant No. 1850117, Amazon Research Credits, Auburn University, and donations from Adobe Systems Inc. and Nvidia.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**, 14410–14430 (2018)
2. Alcorn, M.A., et al.: Strike (with) a pose: neural networks are easily fooled by strange poses of familiar objects. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4845–4854. IEEE (2019)
3. Baer, M., Connors, B.W., Paradiso, M.A.: Neuroscience: Exploring the brain (2007)
4. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: quantifying interpretability of deep visual representations. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), pp. 3319–3327. IEEE (2017)
5. Bengio, Y., Mesnil, G., Dauphin, Y., Rifai, S.: Better mixing via deep representations. In: International Conference on Machine Learning, pp. 552–560 (2013)
6. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Neural photo editing with introspective adversarial networks. arXiv preprint arXiv:1609.07093 (2016)

7. Deng, J., et al.: Imagenet: a large-scale hierarchical image database. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), pp. 248–255 (2009)
8. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., et al.: Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), pp. 2625–2634 (2015)
9. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. In: Advances in Neural Information Processing Systems (NIPS), pp. 658–666 (2016)
10. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Dept. IRO, Université de Montréal, Technical report 4323 (2009)
11. Fong, R., Vedaldi, A.: Net2vec: quantifying and explaining how concepts are encoded by filters in deep neural networks. arXiv preprint arXiv:1801.03454 (2018)
12. Goh, G.: Image synthesis from Yahoo Open NSFW (2016). https://opennsfw.gitlab.io
13. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS), pp. 2672–2680 (2014)
14. Hubel, D.H., Wiesel, T.N.: Receptive fields of single neurones in the cat's striate cortex. J. Physiol. **148**(3), 574–591 (1959)
15. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
16. Kabilan, V.M., Morris, B., Nguyen, A.: Vectordefense: vectorization as a defense to adversarial examples. arXiv preprint arXiv:1804.08529 (2018)
17. Kandel, E.R., Schwartz, J.H., Jessell, T.M., Siegelbaum, S.A., Hudspeth, A.J., et al.: Principles of Neural Science, vol. 4. McGraw-Hill, New York (2000)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS), pp. 1097–1105 (2012)
19. Le, Q.V.: Building high-level features using large scale unsupervised learning. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8595–8598. IEEE (2013)
20. Li, Y., Yosinski, J., Clune, J., Lipson, H., Hopcroft, J.: Convergent learning: do different neural networks learn the same representations? In: Feature Extraction: Modern Questions and Challenges, pp. 196–212 (2015)
21. Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), pp. 233–255 (2016)
22. Malakhova, K.: Visualization of information encoded by neurons in the higher-level areas of the visual system. J. Opt. Technol. **85**(8), 494–498 (2018)
23. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digit. Signal Proc. **73**, 1–15 (2017)
24. Mordvintsev, A., Olah, C., Tyka, M.: Inceptionism: going deeper into neural networks. Google Research Blog (2015). Accessed 20 June
25. Nguyen, A., University of Wyoming. Computer Science Department, U.: AI Neuroscience: Visualizing and Understanding Deep Neural Networks. University of Wyoming (2017). https://books.google.com/books?id=QCexswEACAAJ
26. Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., Yosinski, J.: Plug & play generative networks: conditional iterative generation of images in latent space. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), pp. 3510–3520. IEEE (2017)

27. Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Advances in Neural Information Processing Systems, pp. 3387–3395 (2016)
28. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), pp. 427–436 (2015)
29. Nguyen, A., Yosinski, J., Clune, J.: Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks. In: Visualization for Deep Learning Workshop, ICML Conference (2016)
30. Nguyen, A., Yosinski, J., Clune, J.: Understanding innovation engines: automated creativity and improved stochastic optimization via deep learning. Evol. Comput. **24**(3), 545–572 (2016)
31. Nguyen, A.M., Yosinski, J., Clune, J.: Innovation engines: automated creativity and improved stochastic optimization via deep learning. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 959–966. ACM (2015)
32. Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill **2**(11), e7 (2017)
33. Olah, C., et al.: The building blocks of interpretability. Distill **3**(3), e10 (2018)
34. Palazzo, S., Spampinato, C., Kavasidis, I., Giordano, D., Shah, M.: Decoding brain representations by multimodal learning of neural activity and visual features. arXiv preprint arXiv:1810.10974 (2018)
35. Pei, K., Cao, Y., Yang, J., Jana, S.: DeepXplore: automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 1–18. ACM (2017)
36. Ponce, C.R., Xiao, W., Schade, P., Hartmann, T.S., Kreiman, G., Livingstone, M.S.: Evolving super stimuli for real neurons using deep generative networks. bioRxiv, p. 516484 (2019)
37. Quiroga, R.Q., Reddy, L., Kreiman, G., Koch, C., Fried, I.: Invariant visual representation by single neurons in the human brain. Nature **435**(7045), 1102–1107 (2005)
38. Roberts, G.O., Rosenthal, J.S.: Optimal scaling of discrete approximations to langevin diffusions. J. Roy. Stat. Soc. Ser. B (Stat. Methodol.) **60**(1), 255–268 (1998)
39. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533 (1986)
40. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
41. Shen, G., Horikawa, T., Majima, K., Kamitani, Y.: Deep image reconstruction from human brain activity. PLoS Comput. Biol. **15**(1), e1006633 (2019)
42. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. In: ICLR Workshop (2014)
43. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: a dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
44. Szegedy, C., et al.: Intriguing properties of neural networks. CoRR abs/1312.6199 (2013)
45. Tyka, M.: Class visualization with bilateral filters. https://mtyka.github.io/deepdream/2016/02/05/bilateral-class-vis.html. Accessed 26 June 2018
46. Wei, D., Zhou, B., Torrabla, A., Freeman, W.: Understanding intra-class knowledge inside CNN. arXiv preprint arXiv:1507.02379 (2015)

47. Yeh, R., Chen, C., Lim, T.Y., Hasegawa-Johnson, M., Do, M.N.: Semantic image inpainting with perceptual and contextual losses. arxiv preprint. arXiv preprint arXiv:1607.07539 (2016)
48. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. In: Deep Learning Workshop, ICML Conference (2015)
49. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
50. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Object detectors emerge in deep scene CNNs. In: International Conference on Learning Representations (ICLR) (2015)
51. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems, pp. 487–495 (2014)
52. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 633–641. IEEE (2017)
53. Zhu, J.-Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 597–613. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_36
54. Øygard, A.M.: Visualizing GoogLeNet classes — audun m øygard. https://www.auduno.com/2015/07/29/visualizing-googlenet-classes/. Accessed 26 June 2018