

# Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space

Anh Nguyen

University of Wyoming<sup>†</sup>

anh.ng8@gmail.com

Jeff Clune

Uber AI Labs<sup>†</sup>, University of Wyoming

jeffclune@uwyo.edu

Yoshua Bengio

Montreal Institute for Learning Algorithms

yoshua.umontreal@gmail.com

Alexey Dosovitskiy

University of Freiburg

dosovits@cs.uni-freiburg.de

Jason Yosinski

Uber AI Labs<sup>†</sup>

yosinski@uber.com

## Abstract

*Generating high-resolution, photo-realistic images has been a long-standing goal in machine learning. Recently, Nguyen et al. [37] showed one interesting way to synthesize novel images by performing gradient ascent in the latent space of a generator network to maximize the activations of one or multiple neurons in a separate classifier network. In this paper we extend this method by introducing an additional prior on the latent code, improving both sample quality and sample diversity, leading to a state-of-the-art generative model that produces high quality images at higher resolutions ( $227 \times 227$ ) than previous generative models, and does so for all 1000 ImageNet categories. In addition, we provide a unified probabilistic interpretation of related activation maximization methods and call the general class of models “Plug and Play Generative Networks.” PPGNs are composed of 1) a generator network  $G$  that is capable of drawing a wide range of image types and 2) a replaceable “condition” network  $C$  that tells the generator what to draw. We demonstrate the generation of images conditioned on a class (when  $C$  is an ImageNet or MIT Places classification network) and also conditioned on a caption (when  $C$  is an image captioning network). Our method also improves the state of the art of Multifaceted Feature Visualization [40], which generates the set of synthetic inputs that activate a neuron in order to better understand how deep neural networks operate. Finally, we show that our model performs reasonably well at the task of image inpainting. While image models are used in this paper, the approach is modality-agnostic and can be applied to many types of data.*

<sup>†</sup>This work was mostly performed at Geometric Intelligence, which Uber acquired to create Uber AI Labs.

Figure 1: Images synthetically generated by Plug and Play Generative Networks at high-resolution ( $227 \times 227$ ) for four ImageNet classes. Not only are many images nearly photo-realistic, but samples within a class are diverse.

## 1. Introduction

Recent years have seen generative models that are increasingly capable of synthesizing diverse, realistic images that capture both the fine-grained details and global coherence of natural images [54, 27, 9, 15, 43, 24]. However, many important open challenges remain, including (1) producing photo-realistic images at high resolutions [30], (2) training generators that can produce a wide variety of im-

(a) Real: top 9

(b) DGN-AM [37]

(c) Real: random 9

(d) PPGN (this)

Figure 2: For the “cartoon” class neuron in a pre-trained ImageNet classifier, we show: a) the 9 real training set images that most highly activate that neuron; b) images synthesized by DGN-AM [37], which are of similar type and diversity to the real top-9 images; c) random real training set images in the cartoon class; and d) images synthesized by PPGN, which better represent the diversity of random images from the class. Fig. S10 shows the same four groups for other classes.

ages (e.g. all 1000 ImageNet classes) instead of only one or a few types (e.g. faces or bedrooms [43]), and (3) producing a diversity of samples that match the diversity in the dataset instead of modeling only a subset of the data distribution [14, 53]. Current image generative models often work well at low resolutions (e.g.  $32 \times 32$ ), but struggle to generate high-resolution (e.g.  $128 \times 128$  or higher), globally coherent images (especially for datasets such as ImageNet [7] that have a large variability [41, 47, 14]) due to many challenges including difficulty in training [47, 41] and computationally expensive sampling procedures [54, 55].

Nguyen et al. [37] recently introduced a technique that produces high quality images at a high resolution. Their Deep Generator Network-based Activation Maximization<sup>1</sup> (DGN-AM) involves training a generator  $G$  to create realistic images from compressed features extracted from a pre-trained classifier network  $E$  (Fig. 3f). To generate images conditioned on a class, an optimization process is launched to find a hidden code  $h$  that  $G$  maps to an image that highly activates a neuron in another classifier  $C$  (not necessarily the same as  $E$ ). Not only does DGN-AM produce realistic images at a high resolution (Figs. 2b & S10b), but, without having to re-train  $G$ , it can also produce interesting new types of images that  $G$  never saw during training. For example, a  $G$  trained on ImageNet can produce ballrooms, jail cells, and picnic areas if  $C$  is trained on the MIT Places dataset (Fig. S17, top).

A major limitation with DGN-AM, however, is the lack of diversity in the generated samples. While samples may vary slightly (e.g. “cartoons” with two or three flowers viewed from slightly different angles; see Fig. 2b), the whole image tends to have the same composition (e.g. a closeup of a single cartoon plant with a green background). It is noteworthy that the images produced by DGN-AM

closely match the images from that class that most highly activate the class neuron (Fig. 2a). Optimization often converges to the same mode even with different random initializations, a phenomenon common with activation maximization [11, 40, 59]. In contrast, real images within a class tend to show more diversity (Fig. 2c). In this paper, we improve the diversity and quality of samples produced via DGN-AM by adding a prior on the latent code that keeps optimization along the manifold of realistic-looking images (Fig. 2d).

We do this by providing a probabilistic framework in which to unify and interpret activation maximization approaches [48, 64, 40, 37] as a type of *energy-based* model [4, 29] where the energy function is a sum of multiple constraint terms: (a) priors (e.g. biasing images to look realistic) and (b) conditions, typically given as a category of a separately trained classification model (e.g. encouraging images to look like “pianos” or both “pianos” and “candles”). We then show how to sample iteratively from such models using an approximate Metropolis-adjusted Langevin sampling algorithm.

We call this general class of models Plug and Play Generative Networks (PPGN). The name reflects an important, attractive property of the method: one is free to design an energy function, and “plug and play” with different priors and conditions to form a new generative model. This property has recently been shown to be useful in multiple image generation projects that use the DGN-AM generator network prior and swap in different condition networks [66, 13]. In addition to generating images conditioned on a class, PPGNs can generate images conditioned on text, forming a text-to-image generative model that allows one to describe an image with words and have it synthesized. We accomplish this by attaching a recurrent, image-captioning network (instead of an image classification network) to the output of the generator, and performing similar iterative sampling. Note that, while this paper discusses only the image generation domain, the approach should generalize to

<sup>1</sup> Activation maximization is a technique of searching via optimization for the synthetic image that maximally activates a target neuron in order to understand which features that neuron has learned to detect [11].

many other data types. We publish our code and the trained networks at <http://EvolvingAI.org/ppgn>.

## 2. Probabilistic interpretation of iterative image generation methods

Beginning with the Metropolis-adjusted Langevin algorithm [46, 45] (MALA), it is possible to define a Markov chain Monte Carlo (MCMC) sampler whose stationary distribution approximates a given distribution  $p(x)$ . We refer to our variant of MALA as *MALA-approx*, which uses the following transition operator:<sup>2</sup>

$$x_{t+1} = x_t + \frac{1}{\sqrt{2}} \log p(x_t) + N(0, \frac{2}{3}) \quad (1)$$

A full derivation and discussion is given in Sec. S6. Using this sampler we first derive a probabilistically interpretable formulation for activation maximization methods (Sec. 2.1) and then interpret other activation maximization algorithms in this framework (Sec. 2.2, Sec. S7).

### 2.1. Probabilistic framework for Activation Maximization

Assume we wish to sample from a joint model  $p(x, y)$ , which can be decomposed into an image model and a classification model:

$$p(x, y) = p(x)p(y|x) \quad (2)$$

This equation can be interpreted as a “product of experts” [19] in which each expert determines whether a soft constraint is satisfied. First, a  $p(y|x)$  expert determines a condition for image generation (e.g. images have to be classified as “cardoon”). Also, in a high-dimensional image space, a good  $p(x)$  expert is needed to ensure the search stays in the manifold of image distribution that we try to model (e.g. images of faces [6, 63], shoes [67] or natural images [37]), otherwise we might encounter “fooling” examples that are unrecognizable but have high  $p(y|x)$  [38, 51]. Thus,  $p(x)$  and  $p(y|x)$  together impose a complicated high-dimensional constraint on image generation.

We could write a sampler for the full joint  $p(x, y)$ , but because  $y$  variables are categorical, suppose for now that we fix  $y$  to be a particular chosen class  $y_c$ , with  $y_c$  either sampled or chosen outside the inner sampling loop.<sup>3</sup> This leaves us with the conditional  $p(x|y)$ :

<sup>2</sup> We abuse notation slightly in the interest of space and denote as  $N(0, \frac{2}{3})$  a sample from that distribution. The first step size is given as  $\frac{1}{\sqrt{2}}$  in anticipation of later splitting into separate  $\alpha_1$  and  $\alpha_2$  terms.

<sup>3</sup> One could resample  $y$  in the loop as well, but resampling  $y$  via the Langevin family under consideration is not a natural fit: because  $y$  values from the data set are one-hot – and from the model hopefully nearly so – there will be a wide small- or zero-likelihood region between  $(x, y)$  pairs coming from different classes. Thus making local jumps will not be a good sampling scheme for the  $y$  components.

$$\begin{aligned} p(x|y = y_c) &= p(x)p(y = y_c|x)/p(y = y_c) \\ &= p(x)p(y = y_c|x) \end{aligned} \quad (3)$$

We can construct a MALA-approx sampler for this model, which produces the following update step:

$$\begin{aligned} x_{t+1} &= x_t + \frac{1}{\sqrt{2}} \log p(x_t|y = y_c) + N(0, \frac{2}{3}) \\ &= x_t + \frac{1}{\sqrt{2}} \log p(x_t) + \frac{1}{\sqrt{2}} \log p(y = y_c|x_t) + N(0, \frac{2}{3}) \end{aligned} \quad (4)$$

Expanding the  $\log p(y = y_c|x_t)$  into explicit partial derivatives and decoupling  $\frac{1}{\sqrt{2}}$  into explicit  $\alpha_1$  and  $\alpha_2$  multipliers, we arrive at the following form of the update rule:

$$x_{t+1} = x_t + \alpha_1 \frac{\log p(x_t)}{\sqrt{2}} + \alpha_2 \frac{\log p(y = y_c|x_t)}{\sqrt{2}} + N(0, \frac{2}{3}) \quad (5)$$

We empirically found that decoupling the  $\alpha_1$  and  $\alpha_2$  multipliers works better. An intuitive interpretation of the actions of these three terms is as follows:

- $\alpha_1$  term: take a step from the current image  $x_t$  toward one that looks more like a generic image (an image from any class).
- $\alpha_2$  term: take a step from the current image  $x_t$  toward an image that causes the classifier to output higher confidence in the chosen class. The  $p(y = y_c|x_t)$  term is typically modeled by the softmax output units of a modern convnet, e.g. AlexNet [26] or VGG [49].
- $\alpha_3$  term: add a small amount of noise to jump around the search space to encourage a diversity of images.

### 2.2. Interpretation of previous models

Aside from the errors introduced by not including a reject step, the stationary distribution of the sampler in Eq. 5 will converge to the appropriate distribution if the  $\alpha$  terms are chosen appropriately [61]. Thus, we can use this framework to interpret previously proposed iterative methods for generating samples, evaluating whether each method faithfully computes and employs each term.

There are many previous approaches that iteratively sample from a trained model to generate images [48, 64, 40, 37, 60, 2, 11, 63, 67, 6, 39, 38, 34], with methods designed for different purposes such as activation maximization [48, 64, 40, 37, 60, 11, 38, 34] or generating realistic-looking images by sampling in the latent space of a generator network [63, 37, 67, 6, 2, 17]. However, most of them are gradient-based, and can be interpreted as a variant of MCMC sampling from a graphical model [25].

While an analysis of the full spectrum of approaches is outside this paper’s scope, we do examine a few representative approaches under this framework in Sec. S7. In particular, we interpret the models that lack a  $p(x)$  image

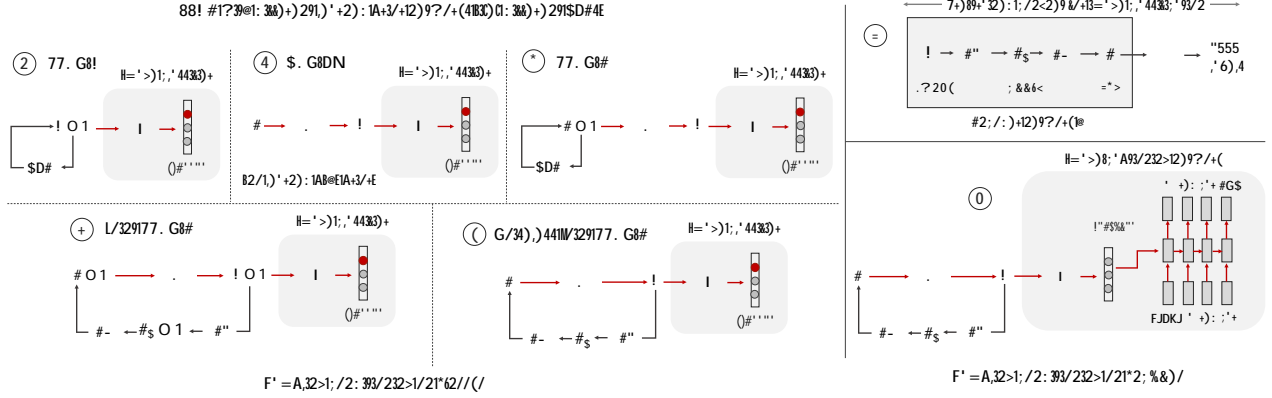


Figure 3: Different variants of PPGN models we tested. The Noiseless Joint PPGN-h (e), which we found empirically produces the best images, generated the results shown in Figs. 1 & 2 & Sections 3.5 & 4. In all variants, we perform iterative sampling following the gradients of two terms: the condition (red arrows) and the prior (black arrows). (a) PPGN-x (Sec. 3.1): To avoid fooling examples [38] when sampling in the high-dimensional image space, we incorporate a  $p(x)$  prior modeled via a denoising autoencoder (DAE) for images, and sample images conditioned on the output classes of a condition network C (or, to visualize hidden neurons, conditioned upon the activation of a hidden neuron in C). (b) DGN-AM (Sec. 3.2): Instead of sampling in the image space (i.e. in the space of individual pixels), Nguyen et al. [37] sample in the abstract, high-level feature space  $h$  of a generator  $G$  trained to reconstruct images  $x$  from compressed features  $h$  extracted from a pre-trained encoder  $E$  (f). Because the generator network was trained to produce realistic images, it serves as a prior on  $p(x)$  since it ideally can only generate real images. However, this model has no learned prior on  $p(h)$  (save for a simple Gaussian assumption). (c) PPGN-h (Sec. 3.3): We attempt to improve the mixing speed and image quality by incorporating a learned  $p(h)$  prior modeled via a multi-layer perceptron DAE for  $h$ . (d) Joint PPGN-h (Sec. 3.4): To improve upon the poor data modeling of the DAE in PPGN-h, we experiment with treating  $G + E_1 + E_2$  as a DAE that models  $h$  via  $x$ . In addition, to possibly improve the robustness of  $G$ , we also add a small amount of noise to  $h_1$  and  $x$  during training and sampling, treating the entire system as being composed of 4 interleaved models that share parameters: a GAN and 3 interleaved DAEs for  $x$ ,  $h_1$  and  $h$ , respectively. This model mixes substantially faster and produces better image quality than DGN-AM and PPGN-h (Fig. S14). (e) Noiseless Joint PPGN-h (Sec. 3.5): We perform an ablation study on the Joint PPGN-h, sweeping across noise levels or loss combinations, and found a Noiseless Joint PPGN-h variant trained with one less loss (Sec. S9.4) to produce the best image quality. (f) A pre-trained image classification network (here, AlexNet trained on ImageNet) serves as the encoder network  $E$  component of our model by mapping an image  $x$  to a useful, abstract, high-level feature space  $h$  (here, AlexNet’s fc6 layer). (g) Instead of conditioning on classes, we can generate images conditioned on a caption by attaching a recurrent, image-captioning network to the output layer of  $G$ , and performing similar iterative sampling.

prior, yielding adversarial or fooling examples [51, 38] as setting  $(\epsilon_1, \epsilon_2, \epsilon_3) = (0, 1, 0)$ ; and methods that use  $L_2$  decay during sampling as using a Gaussian  $p(x)$  prior with  $(\epsilon_1, \epsilon_2, \epsilon_3) = (\epsilon, 1, 0)$ . Both lack a noise term and thus sacrifice sample diversity.

### 3. Plug and Play Generative Networks

Previous models are often limited in that they use hand-engineered priors when sampling in either image space or the latent space of a generator network (see Sec. S7). In this paper, we experiment with 4 different explicitly learned priors modeled by a denoising autoencoder (DAE) [57].

We choose a DAE because, although it does not allow evaluation of  $p(x)$  directly, it *does* allow approximation of the gradient of the log probability when trained with Gaussian noise with variance  $\sigma^2$  [1]; with sufficient capacity and

training time, the approximation is perfect in the limit as  $\sigma \rightarrow 0$ :

$$\frac{\log p(x)}{x} \approx \frac{R_x(x) - x}{2} \quad (6)$$

where  $R_x$  is the reconstruction function in  $x$ -space representing the DAE, i.e.  $R_x(x)$  is a “denoised” output of the autoencoder  $R_x$  (an encoder followed by a decoder) when the encoder is fed input  $x$ . This term approximates exactly the  $\epsilon_1$  term required by our sampler, so we can use it to define the steps of a sampler for an image  $x$  from class  $c$ . Pulling the  $\epsilon_2$  term into  $\epsilon_1$ , the update is:

$$x_{t+1} = x_t + \epsilon_1 \frac{R_x(x_t) - x_t}{2} + \epsilon_2 \frac{\log p(y = y_c | x_t)}{x_t} + N(0, \sigma^2) \quad (7)$$



### 3.1. PPGN-x: DAE model of $p(x)$

First, we tested using a DAE to model  $p(x)$  directly (Fig. 3a) and sampling from the entire model via Eq. 7. However, we found that PPGN-x exhibits two expected problems: (1) it models the data distribution poorly; and (2) the chain mixes slowly. More details are in Sec. S11.

### 3.2. DGN-AM: sampling without a learned prior

Poor mixing in the high-dimensional pixel space of PPGN-x is consistent with previous observations that mixing on higher layers can result in faster exploration of the space [5, 33]. Thus, to ameliorate the problem of slow mixing, we may reparameterize  $p(x)$  as  $\int p(h)p(x|h)dh$  for some latent  $h$ , and perform sampling in this lower-dimensional  $h$ -space.

While several recent works had success with this approach [37, 6, 63], they often hand-design the  $p(h)$  prior. Among these, the DGN-AM method [37] searches in the latent space of a generator network  $G$  to find a code  $h$  such that the image  $G(h)$  highly activates a given neuron in a target DNN. We start by reproducing their results for comparison.  $G$  is trained following the methodology in Dosovitskiy & Brox [9] with an  $L_2$  image reconstruction loss, a Generative Adversarial Networks (GAN) loss [14], and an  $L_2$  loss in a feature space  $h_1$  of an encoder  $E$  (Fig. 3f). The last loss encourages generated images to match the real images in a high-level feature space and is referred to as “feature matching” [47] in this paper, but is also known as “perceptual similarity” [28, 9] or a form of “moment matching” [31]. Note that in the GAN training for  $G$ , we simultaneously train a discriminator  $D$  to tell apart real images  $x$  vs. generated images  $G(h)$ . More training details are in Sec. S9.4.

The directed graphical model interpretation of DGN-AM is  $h \rightarrow x \rightarrow y$  (see Fig. 3b) and the joint  $p(h, x, y)$  can be decomposed into:

$$p(h, x, y) = p(h)p(x|h)p(y|x) \quad (8)$$

where  $h$  in this case represents features extracted from the first fully connected layer (called  $fc6$ ) of a pre-trained AlexNet [26] 1000-class ImageNet [7] classification network (see Fig. 3f).  $p(x|h)$  is modeled by  $G$ , an upconvolutional (also “deconvolutional”) network [10] with 9 upconvolutional and 3 fully connected layers.  $p(y|x)$  is modeled by  $C$ , which in this case is also the AlexNet classifier. The model for  $p(h)$  was an implicit unimodal Gaussian implemented via L2 decay in  $h$ -space [37].

Since  $x$  is a deterministic variable, the model simplifies to:

$$p(h, y) = p(h)p(y|h) \quad (9)$$

From Eq. 5, if we define a Gaussian  $p(h)$  centered at 0 and set  $(\alpha_1, \alpha_2, \alpha_3) = (\alpha, 1, 0)$ , pulling Gaussian con-

stants into  $\alpha$ , we obtain the following noiseless update rule in Nguyen et al. [37] to sample  $h$  from class  $y_c$ :

$$\begin{aligned} h_{t+1} &= (1 - \alpha_2)h_t + \alpha_2 \frac{\log p(y = y_c|h_t)}{h_t} \\ &= (1 - \alpha_2)h_t + \alpha_2 \frac{\log C_c(G(h_t))}{G(h_t)} \frac{G(h_t)}{h_t} \end{aligned} \quad (10)$$

where  $C_c(\cdot)$  represents the output unit associated with class  $y_c$ . As before, all terms are computable in a single forward-backward pass. More concretely, to compute the  $\alpha_2$  term, we push a code  $h$  through the generator  $G$  and condition network  $C$  to the output class  $c$  that we want to condition on (Fig. 3b, red arrows), and back-propagate the gradient via the same path to  $h$ . The final  $h$  is pushed through  $G$  to produce an image sample.

Under this newly proposed framework, we have successfully reproduced the original DGN-AM results and their issue of converging to the same mode when starting from different random initializations (Fig. 2b). We also found that DGN-AM mixes somewhat poorly, yielding the same image after many sampling steps (Figs. S13b & S14b).

### 3.3. PPGN-h: Generator and DAE model of $p(h)$

We attempt to address the poor mixing speed of DGN-AM by incorporating a proper  $p(h)$  prior learned via a DAE into the sampling procedure described in Sec. 3.2. Specifically, we train  $R_h$ , a 7-layer, fully-connected DAE on  $h$  (as before,  $h$  is a  $fc6$  feature vector). The size of the hidden layers are respectively: 4096 – 2048 – 1024 – 500 – 1024 – 2048 – 4096. Full training details are provided in S9.3.

The update rule to sample  $h$  from this model is similar to Eq. 10 except for the inclusion of all three terms:

$$\begin{aligned} h_{t+1} &= h_t + \alpha_1(R_h(h_t) - h_t) + \alpha_2 \frac{\log C_c(G(h_t))}{G(h_t)} \frac{G(h_t)}{h_t} \\ &\quad + N(0, \frac{\alpha_3}{3}) \end{aligned} \quad (11)$$

Concretely, to compute  $R_h(h_t)$  we push  $h_t$  through the learned DAE, encoding and decoding it (Fig. 3c, black arrows). The  $\alpha_2$  term is computed via a forward and backward pass through both  $G$  and  $C$  networks as before (Fig. 3c, red arrows). Lastly, we add the same amount of noise  $N(0, \frac{\alpha_3}{3})$  used during DAE training to  $h$ . Equivalently, noise can also be added before the encode-decode step.

We sample<sup>4</sup> using  $(\alpha_1, \alpha_2, \alpha_3) = (10^{-5}, 1, 10^{-5})$  and show results in Figs. S13c & S14c. As expected, the chain mixes faster than PPGN-x, with subsequent samples appearing more qualitatively different from their predecessors. However, the samples for PPGN-h are qualitatively similar to those from DGN-AM (Figs. S13b & S14b). Samples still lack quality and diversity, which we hypothesize is due to the poor  $p(h)$  model learned by the DAE.

<sup>4</sup> If faster mixing or more stable samples are desired, then  $\alpha_1$  and  $\alpha_3$  can be scaled up or down together. Here we scale both down to  $10^{-5}$ .

### 3.4 Joint PPGN-h: joint Generator and DAE

The previous result suggests that the simple multi-layer perceptron DAE poorly modeled the distribution of  $fc6$  features. This could occur because the DAE faces the generally difficult unconstrained density estimation problem. To combat this issue, we experiment with modeling  $h$  via  $x$  with a DAE:  $h \rightarrow x \rightarrow h$ . Intuitively, to help the DAE better model  $h$ , we force it to generate realistic-looking images  $x$  from features  $h$  and then decode them back to  $h$ . One can train this DAE from scratch separately from  $G$  (as done for PPGN-h). However, in the DGN-AM formulation,  $G$  models the  $h \rightarrow x$  (Fig. 3b) and  $E$  models the  $x \rightarrow h$  (Fig. 3f). Thus, the composition  $G(E(\cdot))$  can be considered an AE  $h \rightarrow x \rightarrow h$  (Fig. 3d).

Note that  $G(E(\cdot))$  is theoretically not a formal  $h$ -DAE because its two components were trained with neither noise added to  $h$  nor an  $L_2$  reconstruction loss for  $h$  [37] (more details in Sec. S9.4) as is required for regular DAE training [57]. To make  $G(E(\cdot))$  a more theoretically justifiable  $h$ -DAE, we add noise to  $h$  and train  $G$  with an additional reconstruction loss for  $h$  (Fig. S9c). We do the same for  $x$  and  $h_1$  (pool5 features), hypothesizing that a little noise added to  $x$  and  $h_1$  might encourage  $G$  to be more robust [57]. In other words, with the same existing network structures from DGN-AM [37], we train  $G$  differently by treating the entire model as being composed of 3 interleaved DAEs that share parameters: one each for  $h$ ,  $h_1$ , and  $x$  (see Fig. S9c). Note that  $E$  remains frozen, and  $G$  is trained with 4 losses in total i.e. three  $L_2$  reconstruction losses for  $x$ ,  $h$ , and  $h_1$  and a GAN loss for  $x$ . See Sec. S9.5 for full training details. We call this the *Joint PPGN-h* model.

We sample from this model following the update rule in Eq. 11 with  $(\epsilon_1, \epsilon_2) = (10^{-5}, 1)$ , and with noise added to all three variables:  $h$ ,  $h_1$  and  $x$  instead of only to  $h$  (Fig. 3d vs e). The noise amounts added at each layer are the same as those used during training. As hypothesized, we observe that the sampling chain from this model mixes substantially faster and produces samples with better quality than all previous PPGN treatments (Figs. S13d & S14d) including PPGN-h, which has a multi-layer perceptron  $h$ -DAE.

### 3.5 Ablation study with Noiseless Joint PPGN-h

While the Joint PPGN-h outperforms all previous treatments in sample quality and diversity (as the chain mixes faster), the model is trained with a combination of four losses and noise added to all variables. This complex training process can be difficult to understand, making further improvements non-intuitive. To shed more light into how the Joint PPGN-h works, we perform ablation experiments which later reveal a better-performing variant.

**Noise sweeps.** To understand the effects of adding noise to each variable, we train variants of the Joint PPGN-h (1) with different noise levels, (2) using noise on only a single

variable, and (3) using noise on multiple variables simultaneously. We did not find these variants to produce qualitatively better reconstruction results than the Joint PPGN-h. Interestingly, in a PPGN variant trained with no noise at all, the  $h$ -autoencoder given by  $G(E(\cdot))$  still appears to be contractive, i.e. robust to a large amount of noise (Fig. S16). This is beneficial during sampling; if “unrealistic” codes appear,  $G$  could map them back to realistic-looking images. We believe this property might emerge for multiple reasons: (1)  $G$  and  $E$  are not trained jointly; (2)  $h$  features encode global, high-level rather than local, low-level information; (3) the presence of the adversarial cost when training  $G$  could make the  $h \rightarrow x$  mapping more “many-to-one” by pushing  $x$  towards modes of the image distribution.

**Combinations of losses.** To understand the effects of each loss component, we repeat the Joint PPGN-h training (Sec. 3.4), but without noise added to the variables. Specifically, we test different combinations of losses and compare the quality of images  $G(h)$  produced by pushing the codes  $h$  of real images through  $G$  (without MCMC sampling).

First, we found that removing the adversarial loss from the 4-loss combination yields blurrier images (Fig. S8c). Second, we compare 3 different feature matching losses:  $fc6$ ,  $pool5$ , and both  $fc6$  and  $pool5$  combined, and found that  $pool5$  feature matching loss leads to the best image quality (Sec. S8). Our result is consistent with Dosovitskiy & Brox [9]. Thus, the model that we found empirically to produce the best image quality is trained without noise and with three losses: a  $pool5$  feature matching loss, an adversarial loss, and an image reconstruction loss. We call this variant “Noiseless Joint PPGN-h”: it produced the results in Figs. 1 & 2 and Sections 3.5 & 4.

**Noiseless Joint PPGN-h.** We sample from this model with  $(\epsilon_1, \epsilon_2, \epsilon_3) = (10^{-5}, 1, 10^{-17})$  following the same update rule in Eq. 11 (we need noise to make it a proper sampling procedure, but found that infinitesimally small noise produces better and more diverse images, which is to be expected given that the DAE in this variant was trained without noise). Interestingly, the chain mixes substantially faster than DGN-AM (Figs. S13e & S13b) although the only difference between two treatments is the existence of the learned  $p(h)$  prior. Overall, the Noiseless Joint PPGN-h produces a large amount of sample diversity (Fig. 2). Compared to the Joint PPGN-h, the Noiseless Joint PPGN-h produces better image quality, but mixes slightly slower (Figs. S13 & S14). Sweeping across the noise levels during sampling, we noted that larger noise amounts often results in worse image quality, but not necessarily faster mixing speed (Fig. S15). Also, as expected, a small  $\epsilon_1$  multiplier makes the chain mix faster, and a large one pulls the samples towards being generic instead of class-specific (Fig. S23).

**Evaluations.** Evaluating image generative models is

challenging, and there is not yet a commonly accepted quantitative performance measure [53]. We qualitatively evaluate sample diversity of the Noiseless Joint PPGN-h variant by running 10 sampling chains, each for 200 steps, to produce 2000 samples, and filtering out samples with class probability of less than 0.97. From the remaining, we randomly pick 400 samples and plot them in a grid t-SNE [56] (Figs. S12 & S11). More examples for the reader’s evaluation of sample quality and diversity are provided in Figs. S21, S22 & S25. To better observe the mixing speed, we show videos of sampling chains (with one sample per frame; no samples filtered out) from within classes and between 10 different classes at <https://goo.gl/36S0Dy>. In addition, Table S3 provides quantitative comparisons between PPGN, auxiliary classifier GAN [41] and real ImageNet images in image quality (via Inception score [47] & Inception accuracy [41]) and diversity (via MS-SSIM metric [41]).

While future work is required to fully understand why the Noiseless Joint PPGN-h produces high-quality images at a high resolution for 1000-class ImageNet more successfully than other existing latent variable models [41, 47, 43], we discuss possible explanations in Sec. S12.

## 4. Additional results

In this section, we take the Noiseless Joint PPGN-h model and show its capabilities on several different tasks.

### 4.1. Generating images with different condition networks

A compelling property that makes PPGN different from other existing generative models is that one can “plug and play” with different prior and condition components (as shown in Eq. 2) and ask the model to perform new tasks, including challenging the generator to produce images that it has never seen before. Here, we demonstrate this feature by replacing the  $p(y|x)$  component with different networks.

#### Generating images conditioned on classes

Above we showed that PPGN could generate a diversity of high quality samples for ImageNet classes (Figs. 1 & 2 & Sec. 3.5). Here, we test whether the generator  $G$  within the PPGN can generalize to new types of images that it has never seen before. Specifically, we sample with a different  $p(y|x)$  model: an AlexNet DNN [26] trained to classify 205 categories of scene images from the MIT Places dataset [65]. Similar to DGN-AM [37], the PPGN generates realistic-looking images for classes that the generator was never trained on, such as “alley” or “hotel room” (Fig. 4). A side-by-side comparison between DGN-AM and PPGN are in Fig. S17.

#### Generating images conditioned on captions

Figure 4: Images synthesized conditioned on MIT Places [65] classes instead of ImageNet classes.

Instead of conditioning on classes, we can also condition the image generation on a caption (Fig. 3g). Here, we swap in an image-captioning recurrent network (called LRCN) from [8] that was trained on the MS COCO dataset [32] to predict a caption  $y$  given an image  $x$ . Specifically, LRCN is a two-layer LSTM network that generates captions conditioned on features extracted from the output softmax layer of AlexNet [26].

Figure 5: Images synthesized to match a text description. A PPGN containing the image captioning model from [8] can generate reasonable images that differ based on user-provided captions (e.g. *red* car vs. *blue* car, oranges vs. *a pile* of oranges). For each caption, we show 3 images synthesized starting from random codes (more in Fig. S18).

We found that PPGN can generate reasonable images in many cases (Figs. 5 & S18), although the image quality is lower than when conditioning on classes. In other cases, it also fails to generate high-quality images for certain types of images such as “people” or “giraffe”, which are not categories in the generator’s training set (Fig. S18). We also observe “fooling” images [38]—those that look unrecognizable to humans, but produce high-scoring captions. More results are in Fig. S18. The challenges for this task could be: (1) the sampling is conditioned on many (10 – 15) words at the same time, and the gradients backpropagated from dif-

ferent words could conflict with each other; (2) the LRCN captioning model itself is easily fooled, thus additional priors on the conversion from image features to natural language could improve the result further; (3) the depth of the entire model (AlexNet and LRCN) impairs gradient propagation during sampling. In the future, it would be interesting to experiment with other state-of-the-art image captioning models [12, 58]. Overall, we have demonstrated that PPGNs can be flexibly turned into a text-to-image model by combining the prior with an image captioning network, and this process *does not* even require additional training.

### Generating images conditioned on hidden neurons

PPGNs can perform a more challenging form of activation maximization called Multifaceted Feature Visualization (MFV) [40], which involves generating the *set* of inputs that activate a given neuron. Instead of conditioning on a class output neuron, here we condition on a *hidden* neuron, revealing many facets that a neuron has learned to detect (Fig. 6).

Figure 6: Images synthesized to activate a hidden neuron (number 196) previously identified as a “face detector neuron” [64] in the fifth convolutional layer of the AlexNet DNN trained on ImageNet. The PPGN uncovers a large diversity of types of inputs that activate this neuron, thus performing Multifaceted Feature Visualization [40], which sheds light into what the neuron has learned to detect. The different facets include different types of human faces (top row), dog faces (bottom row), and objects that only barely resemble faces (e.g. the windows of a house, or something resembling green hair above a flesh-colored patch). More examples and details are shown in Figs. S19 & S20.

## 4.2. Inpainting

Because PPGNs can be interpreted probabilistically, we can also sample from them conditioned on part of an image (in addition to the class condition) to perform inpainting—filling in missing pixels given the observed context regions [42, 3, 63, 54]. The model must understand the entire image to be able to reasonably fill in a large masked out region that is positioned randomly. Overall, we found that PPGNs are able to perform inpainting suggesting that the models do “understand” the semantics of concepts such as junco or bell pepper (Fig. 7) rather than merely memorizing the images. More details and results are in Sec. S10.

Figure 7: We perform class-conditional image sampling to fill in missing pixels (see Sec. 4.2). In addition to conditioning on a specific class (PPGN), PPGN-context also constrains the code  $h$  to produce an image that matches the context region. PPGN-context (c) matches the pixels surrounding the masked region better than PPGN (b), and semantically fills in better than the Context-Aware Fill feature in Photoshop (d) in many cases. The result shows that the class-conditional PPGN does understand the semantics of images. More PPGN-context results are in Fig. S24.

## 5. Conclusion

The most useful property of PPGN is the capability of “plug and play”—allowing one to drop in a replaceable condition network and generate images according to a condition specified at test time. Beyond the applications we demonstrated here, one could use PPGNs to synthesize images for videos or create arts with one or even multiple condition networks at the same time [13]. Note that DGN-AM [37]—the predecessor of PPGNs—has previously enabled both scientists and amateurs without substantial resources to take a pre-trained condition network and generate art [13] and scientific visualizations [66]. An explanation for why this is possible is that the  $fc6$  features that the generator was trained to invert are relatively general and cover the set of natural images. Thus, there is great value in producing flexible, powerful generators that can be combined with pre-trained condition networks in a plug and play fashion.

### Acknowledgments

We thank Theo Karaletsos and Noah Goodman for helpful discussions, and Jeff Donahue for providing a trained image captioning model [8] for our experiments. We also thank Joost Huizinga, Christopher Stanton, Rosanne Liu, Tyler Jaskowiak, Richard Yang, and Jon Berliner for invaluable suggestions on preliminary drafts.



## References

- [1] G. Alain and Y. Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014. **4, 17, 19**
- [2] K. Arulkumaran, A. Creswell, and A. A. Bharath. Improving sampling from generative autoencoders with markov chains. *arXiv preprint arXiv:1610.09296*, 2016. **3, 13**
- [3] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009. **8, 17**
- [4] I. G. Y. Bengio and A. Courville. Deep learning. Book in preparation for MIT Press, 2016. **2, 12**
- [5] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 552–560, 2013. **5**
- [6] A. Brock, T. Lim, J. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. **3, 5, 13**
- [7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. **2, 5, 14, 16, 29**
- [8] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition*, 2015. **7, 8, 29**
- [9] A. Dosovitskiy and T. Brox. Generating Images with Perceptual Similarity Metrics based on Deep Networks. In *Advances in Neural Information Processing Systems*, 2016. **1, 5, 6, 14, 15, 16, 18, 27**
- [10] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015. **5, 16**
- [11] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, Technical report, University of Montreal, 2009. **2, 3, 13, 14**
- [12] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc., 2013. **8**
- [13] G. Goh. Image synthesis from yahoo open nsfw. <https://opennsfw.gtlab.io>, 2016. **2, 8**
- [14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. **2, 5, 16, 18, 19, 27**
- [15] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015. **1**
- [16] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2006. **15**
- [17] T. Han, Y. Lu, S.-C. Zhu, and Y. N. Wu. Alternating back-propagation for generator network. In *AAAI*, 2017. **3, 13**
- [18] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. **12**
- [19] G. E. Hinton. Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 1–6. IET, 1999. **3**
- [20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015. **16**
- [21] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. <http://caffe.berkeleyvision.org/>, 2013. **16**
- [22] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXiv:1603.08155*, 2016. **14**
- [23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. **16**
- [24] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. Dec. 2014. **1, 19**
- [25] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. **3, 12**
- [26] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012. **3, 5, 7, 14, 16, 18, 27**
- [27] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. *Journal of Machine Learning Research*, 15:29–37, 2011. **1**
- [28] A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015. **5, 14**
- [29] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006. **2**
- [30] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. **1**
- [31] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015. **5**
- [32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. **7**

- [33] H. Luo, P. L. Carrier, A. C. Courville, and Y. Bengio. Texture modeling with convolutional spike-and-slab rbms and deep extensions. In *AISTATS*, pages 415–423, 2013. **5**
- [34] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, pages 1–23, 2016. **3, 13, 14**
- [35] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. **12**
- [36] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20, 2015. **14**
- [37] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, 2016. **1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 16, 17, 21, 24, 25, 28, 30**
- [38] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. **3, 4, 7, 13**
- [39] A. Nguyen, J. Yosinski, and J. Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2015. **3, 13**
- [40] A. Nguyen, J. Yosinski, and J. Clune. 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. **1, 2, 3, 8, 13, 14, 30**
- [41] A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *ArXiv e-prints*, Oct. 2016. **2, 7, 18, 20**
- [42] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *arXiv preprint arXiv:1604.07379*, 2016. **8, 17**
- [43] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Nov. 2015. **1, 2, 7, 16, 18**
- [44] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011. **27**
- [45] G. O. Roberts and J. S. Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998. **3, 12**
- [46] G. O. Roberts and R. L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996. **3, 12**
- [47] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. **2, 5, 7, 16, 18, 19, 20**
- [48] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034, presented at ICLR Workshop 2014*, 2013. **2, 3, 13, 14**
- [49] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. **3**
- [50] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. **20**
- [51] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. **3, 4, 13**
- [52] Y. W. Teh, A. H. Thiery, and S. J. Vollmer. Consistency and fluctuations for stochastic gradient langevin dynamics. 2014. **12**
- [53] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. Nov 2016. International Conference on Learning Representations. **2, 7, 19**
- [54] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. *ArXiv e-prints*, Jan. 2016. **1, 2, 8**
- [55] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016. **2**
- [56] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008. **7, 22, 23, 30**
- [57] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008. **4, 6, 17, 19, 27**
- [58] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014. **8**
- [59] D. Wei, B. Zhou, A. Torralba, and W. Freeman. Understanding intra-class knowledge inside cnn. *arXiv preprint arXiv:1507.02379*, 2015. **2, 14**
- [60] D. Wei, B. Zhou, A. Torralba, and W. Freeman. Understanding intra-class knowledge inside cnn. *arXiv preprint arXiv:1507.02379*, 2015. **3, 13**
- [61] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011. **3, 12**
- [62] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *arXiv preprint arXiv:1609.09408*, 2016. **17**
- [63] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016. **3, 5, 8, 13**
- [64] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on*

- Machine Learning (ICML)*, 2015. 2, 3, 8, 13, 14, 17, 30, 31
- [65] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations (ICLR)*, volume abs/1412.6856, 2014. 7, 28
- [66] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016. 2, 8
- [67] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. 3, 13