Synthesizing the preferred inputs for neurons in neural networks via deep generator networks

Akilesh, Phaneendra

Objective of paper

- Activation Maximization (AM) is a method to understand how a neural network functions internally by studying what each of its neurons has learned to detect.
- Use AM technique to understand and synthesize an image that highly activates a neuron.
- What's unique? - Use a powerful, learned prior: A deep generator network (DGN) to improve the AM.
- DGN generates state-of-the-art synthetic images that look almost real.
- It reveals the features learned by each neuron in an interpretable way.
- Generalizes well to new datasets and somewhat well to different network architectures without requiring the prior to be relearned.

Related work

- AM starts from random image and iteratively calculates via backpropagation how the color of each pixel in the image should be changed to increase the activation of a neuron.
- However, doing so without biasing the images produced creates unrealistic, uninterpretable images.
- By incorporating natural image priors (Gaussian-blur, alpha-norm etc.) into the objective function has been shown to substantially improve the recognizability of the images generated.
- In this work, generator networks are used as priors to produce synthetic preferred images.

Method

- The image generator DNN that they use as a prior is trained to take in a code (e.g. vector of scalars) and output a synthetic image that looks as close to real images from the ImageNet dataset as possible.
- To produce a preferred input for a neuron in a given DNN that they want to visualize, they optimize in the input code space of the image generator DNN so that it outputs an image that activates the neuron of interest.
- Their method (DGN-AM) restricts the search to only the set of images that can be drawn by the prior, which provides a strong biases toward realistic visualizations.

Architecture

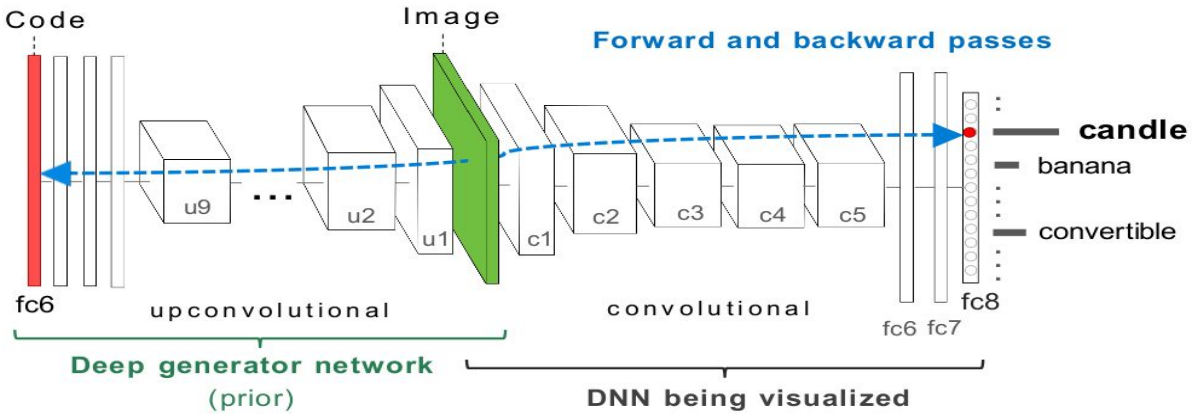


Figure 2: To synthesize a preferred input for a target neuron h (e.g. the "candle" class output neuron), we optimize the hidden code input (red bar) of a deep image generator network (DGN) to produce an image that highly activates h. In the example shown, the DGN is a network trained to invert the feature representations of layer fc6 of CaffeNet. The target DNN being visualized can be a different network (with a different architecture and or trained on different data). The gradient information (blue-dashed line) flows from the layer containing h in the target DNN (here, layer fc8) all the way through the image back to the input code layer of the DGN. Note that both the DGN and target DNN being visualized have fixed parameters, and optimization only changes the DGN input code (red).

Training procedure

The training procedure for GANs to reconstruct images from hidden-layer feature representations (Described in Generating Images with Perceptual Similarity Metrics based on Deep Networks, NIPS 2016).

The process involves four convolutional networks:

- A fixed encoder network *E* to be inverted.
- A generator network *G*.
- A fixed comparator network *C*.
- A discriminator *D*.

Training procedure contd.

G is trained to invert a feature representation extracted by the network *E*, and has to satisfy three objectives:

- For feature vector $y_i = E(x_i)$, the synthesized image $G(y_i)$ has to be close to the original image x_i.
- The features of the output image $C(G(y_i))$ have to be close to those of the real image $C(x_i)$.
- D should be unable to distinguish $G(y_i)$ from real images x_i

The objective for D is to discriminate between synthetic images $G(y_i)$ and real images x_i

In this paper, the encoder *E* is CaffeNet truncated at different layers. We denote CaffeNet truncated at layer *l* by E_l , and the network trained to invert E_l by G_l . The “comparator” ***C*** is CaffeNet up to layer pool5. ***D*** is a convolutional network with 5 convolutional and 2 fully connected layers. ***G*** is an up-convolutional architecture with 9 up-convolutional and 3 fully connected layers.

Objective function

Synthesizing the preferred images for a neuron. Intuitively, we search in the input code space of the image generator model G to find a code $\mathbf{y}$ such that $G(\mathbf{y})$ is an image that produces high activation of the target neuron h in the DNN Φ that we want to visualize (i.e. optimization maximizes $\Phi_h(G(\mathbf{y}))$). Recall that G_l is a generator network trained to reconstruct images from the l-th layer features of CaffeNet. Formally, and including a regularization term, we may pose the activation maximization problem as finding a code $\widehat{\mathbf{y}^l}$ such that:

$$\widehat{\mathbf{y}^l} = \arg\max_{\mathbf{y}^l} (\Phi_h(G_l(\mathbf{y}^l)) - \lambda \|\mathbf{y}^l\|) \quad (1)$$

Discussion

- Not only does DGN-AM produce realistic images at a high resolution, 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.
- However, the prior used in this paper does not generalize equally well to DNNs of different architectures.
- This motivates research into how to train such a general prior.
- Their method for synthesizing preferred images could naturally be applied to synthesize preferred videos for an activity recognition DNN to better understand how it works.

Plug and Play Generative Networks: Conditional Iterative Generation of Images in Latent Space.

Introduction and challenges

- They extend the previous 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 × 227).
- Provide a unified probabilistic interpretation of related activation maximization methods and call general class of models “PPGNs”.
- PPGNs are composed of: a generator network ***G*** that is capable of drawing a wide range of image types, a replaceable “condition” network ***C*** that tells the generator what to draw.

Challenges in image-generation

- Produce photo-realistic images at high resolutions.
- Training generators to produce a wide variety of images instead of only one or a few types.
- Produce a diversity of samples that match the diversity in the dataset instead of modeling only a subset of data distribution.

Probabilistic interpretation of iterative image generation methods

- Introduce a probabilistic framework that interprets AM as a energy-based model whose energy function is a sum of priors (biasing images to look realistic) and conditions (category or class).
- Metropolis-adjusted Langevin algorithm (MALA)-approx sampler:

$$x_{t+1} = x_t + \epsilon_{12} \nabla \log p(x_t) + N(0, \epsilon_3^2) \qquad (1)$$

- Suppose 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)***
- Suppose we fix ***y*** to be a particular chosen class y_c

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

Prob. interpretation contd.

- Now construct ***MALA-approx*** sampler for this model:

$$\begin{aligned} x_{t+1} &= x_t + \epsilon_{12} \nabla \log p(x_t|y = y_c) + N(0, \epsilon_3^2) \\ &= x_t + \epsilon_{12} \nabla \log p(x_t) + \epsilon_{12} \nabla \log p(y = y_c|x_t) + N(0, \epsilon_3^2) \end{aligned} \tag{4}$$

- Decoupling epsilon, we get:

$$x_{t+1} = x_t + \epsilon_1 \frac{\partial \log p(x_t)}{\partial x_t} + \epsilon_2 \frac{\partial \log p(y = y_c|x_t)}{\partial x_t} + N(0, \epsilon_3^2) \tag{5}$$

Intuitive interpretation of three terms

- ***e1*** term: take a step from the current image x_t toward one that looks more like a generic image (an image from any class).
- ***e2*** 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 $p(y = y_c \mid x_t)$ term is typically modeled by the softmax output units of a modern convnet.
- ***e3*** term: add a small amount of noise to jump around the search space to encourage a diversity of images.

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.
- In this paper, they experiment with 4 different explicitly learned priors modeled by a denoising autoencoder (DAE).
- Why DAE? - It allows approximation of the gradient of the log probability when trained with Gaussian noise.

$$\frac{\partial \log p(x)}{\partial x} \approx \frac{R_x(x) - x}{\sigma^2}$$

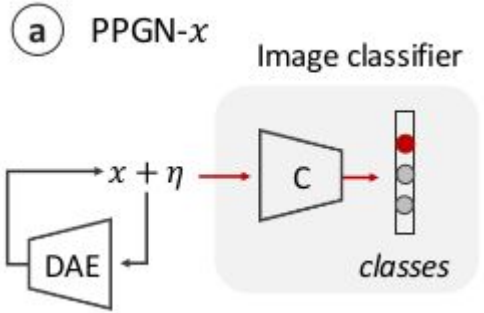
where R_x is the reconstruction function in x-space representing the DAE, i.e. $R_x(x)$ is a "denoised" output of the autoencoder R_x (an encoder followed by a decoder) when the encoder is fed input x.

- Update now becomes:

$$x_{t+1} = x_t + \epsilon_1 \left(R_x(x_t) - x_t\right) + \epsilon_2 \frac{\partial \log p(y = y_c | x_t)}{\partial x_t} + N(0, \epsilon_3^2) \quad (7)$$

Types of models

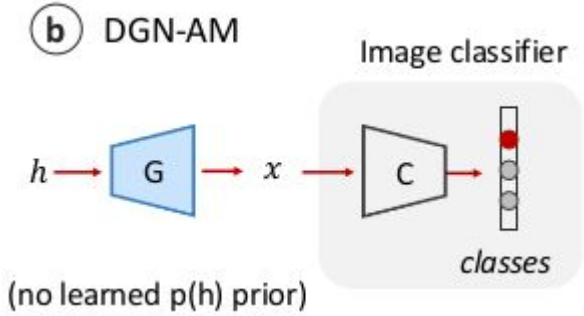
1. **PPGN-*x*: DAE model of *p(x)*:**



To avoid fooling examples when sampling in the high-dimensional image space, they 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***).It models data distribution poorly and chain mixes slowly.

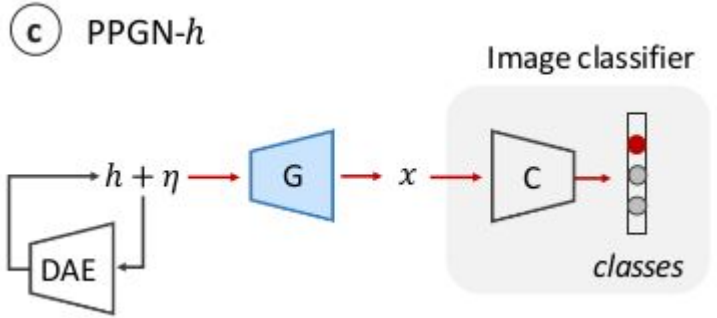
Note: In all variants, they perform iterative sampling following the gradients of two terms: the condition (red arrows) and the prior (black arrows).

2. **DGN-AM: sampling without a learned prior:**



Instead of sampling in the image space (i.e. in the space of individual pixels), they 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).

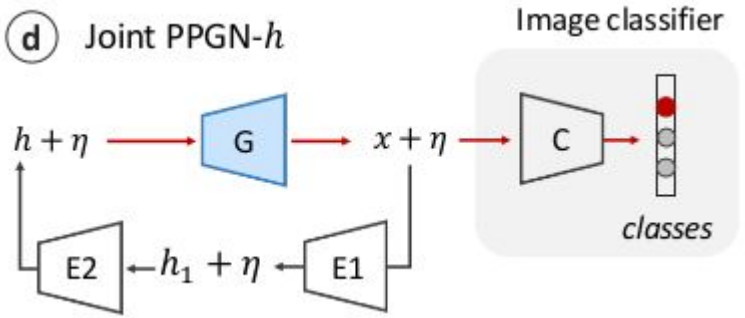
3. **PPGN-*h*: Generator and DAE model of *p(h)*:**



They attempt to improve the mixing speed and image quality of DGN-AM by incorporating a learned ***p(h)*** prior modeled via a multi-layer perceptron DAE for ***h***. *(e_1, e_2, e_3) = (10^{-5}, 1, 10^{-5})*

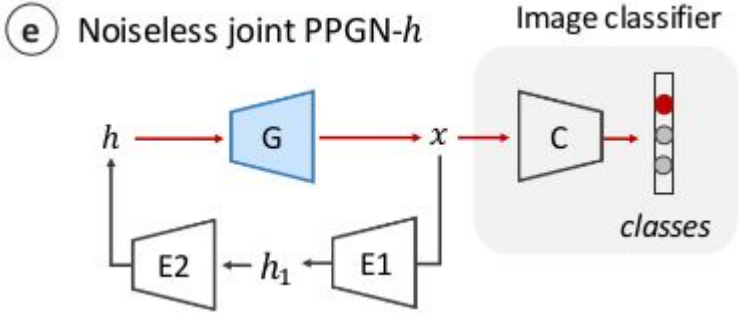
$$h_{t+1} = h_t + \epsilon_1 (R_h(h_t) - h_t) + \epsilon_2 \frac{\partial \log C_c(G(h_t))}{\partial G(h_t)} \frac{\partial G(h_t)}{\partial h_t} + N(0, \epsilon_3^2) \quad (11)$$

4. **Joint PPGN-*h*: joint Generator and DAE:**



To improve upon the poor data modeling of the DAE in PPGN-h and to help the DAE better model ***h***, they force it to generate realistic-looking images ***x*** from features ***h*** and then decode them back to ***h***. They 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***, they also add a small amount of noise to h_1 and ***x*** during training and sampling. In other words, Generator ***G*** from DGN-AM is trained differently by treating the entire model as being composed of 3 interleaved DAE's that share parameters: one each for ***h, h1, x***. This model mixes substantially faster and produces better image quality than **DGN-AM** and **PPGN-h**

5. **Noiseless Joint PPGN-*h***



They 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 to produce the best image quality.