Sampling the "Inverse Set" of a Neuron

Suryabhan Singh Hada and **Miguel Á. Carreira-Perpiñán**
EECS, University of California, Merced

Motivation With the recent success of deep neural networks in computer vision, it is important to understand the internal working of these networks. What does a given neuron represent? The concepts captured by a neuron may be hard to understand or express in simple terms. The approach we propose in this paper is to characterize the region of input space that excites a given neuron to a certain level; we call this the *inverse set*. This inverse set is a complicated high dimensional object that we explore by an optimization-based sampling approach. Inspection of samples of this set by a human can reveal regularities that help to understand the neuron. This goes beyond approaches which were limited to finding an image which maximally activates the neuron [5] or using Markov chain Monte Carlo to sample images [3], but this is very slow, generates samples with little diversity and lacks control over the activation value of the generated samples. Our approach also allows us to explore the intersection of inverse sets of several neurons and other variations.

The inverse set of a neuron, and how to sample it We say an input $\mathbf{x}$ is in the inverse set of a given neuron having a real-valued activation function f if it satisfies the following two properties:

$$z_1 \leq f(\mathbf{x}) \leq z_2 \qquad \mathbf{x} \text{ is a valid input} \tag{1}$$

where $z_1, z_2 \in \mathbb{R}$ are activation values of the neuron. $\mathbf{x}$ being a valid input means the image features are in the valid range (say, pixel values in [0,1]) and it is a realistic image (we explain this later).

For a simple model, the inverse set can be calculated analytically. For example, consider a linear model with logistic activation function $\sigma(\mathbf{w}^T\mathbf{x}+c)$ and all valid inputs to have pixel values between [0,1]. For $z_2 = 1$ (maximum activation value) and $0 < z_1 < z_2$, the inverse set will be the intersection of the half space $\mathbf{w}^T\mathbf{x} + c \geq \sigma^{-1}(z_1)$ and the [0,1] hypercube.

In general for deep neural networks we approximate the inverse set with a sample that covers it in a representative way. A simple way to do this is to select all the images in the training set that satisfy eq. (1), but this may rule out all images. A neuron may "like" certain aspects of a training image without being sufficiently activated by it, or, in other words, the images that activate a given neuron need not look like any specific training image. Therefore, we need an efficient algorithm to sample the inverse set.

Sampling the inverse set of a neuron: an optimization approach To create a sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that covers the inverse set, we transform eq. (1) into a constrained optimization problem (we do not explicitly write the "valid input" constraint to avoid clutter):

$$\underset{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n}{\arg\max} \sum_{i,j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \qquad \text{s.t.} \qquad z_1 \leq f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n) \leq z_2. \tag{2}$$

The objective function makes sure that the samples are different from each other but also satisfy eq. (1). However, this generates noisy-looking samples. To make them realistic we use an image generator network $\mathbf{G}$, which has been empirically shown to produce realistic images [1] when a feature vector $\mathbf{c}$ is passed as an input. Then we get:

$$\underset{\mathbf{c}_1,\mathbf{c}_2,\cdots,\mathbf{c}_n}{\arg\max} \sum_{i,j=1}^{n} \|\mathbf{G}(\mathbf{c}_i) - \mathbf{G}(\mathbf{c}_j)\|_2^2 \quad \text{s.t.} \quad z_1 \leq f(\mathbf{G}(\mathbf{c}_1)), \ldots, f(\mathbf{G}(\mathbf{c}_n)) \leq z_2 \tag{3}$$

We observe that using Euclidean distances directly on the generated images is very sensitive to small changes in their pixels. Instead, we compute distances on a low-dimensional encoding $\mathbf{E}(\mathbf{G}(\mathbf{c}))$ of the generated images, where $\mathbf{E}$ is obtained from the first layers of a deep neural network trained for classification. Then we have our final formulation of the optimization problem over the n samples $\mathbf{G}(\mathbf{c}_1), \ldots, \mathbf{G}(\mathbf{c}_n)$:

$$\underset{\mathbf{c}_1,\mathbf{c}_2,\cdots,\mathbf{c}_n}{\arg\max} \sum_{i,j=1}^{n} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_j))\|_2^2 \quad \text{s.t.} \quad z_1 \leq f(\mathbf{G}(\mathbf{c}_1)), \ldots, f(\mathbf{G}(\mathbf{c}_n)) \leq z_2 \tag{4}$$

Computational solution of the optimization problem Eq. (4) is a constrained optimization problem. We optimize it using the augmented Lagrangian method [4]. Because of the quadratic complexity of the objective function over the number of samples n, it is computationally expensive to generate many samples. We apply two approximations to speed up the sampling process.

Firstly. we solve the problem in an inexact but good enough way. The sum-of-all-pairs objective is not a strict necessity, it is really a mechanism to ensure diversity of the samples and coverage of the inverse set. We observe that this is already achieved by stopping the optimization algorithm once the samples enter the feasible set, by which time they already are sufficiently separated.

Second, we create the samples incrementally, K samples at a time (with $K \ll n$). For the first K samples (which we call *seeds*) we optimize eq. (4), initializing the code vectors $\mathbf{c}$ with random values and stopping as soon as all K samples are in the feasible region. These samples are then fixed. The next K samples use as objective that in eq. (4) plus their distances to the seeds. We initialize them to the previous K samples and take a single gradient step in the augmented Lagrangian optimization, so that the new samples move further away from both the seeds and each other while staying feasible. These gives K new samples which we fix, and the process is repeated until we generate the desired n samples.

Experiments Below we show some results of our sampling approach to create the inverse set for different neurons from CaffeNet [2]. Compared to the samples from [3], our samples are much more diverse as shown in fig: 1

In all cases, $\mathbf{G}$ is a pretrained generative network from [3] and $\mathbf{E}$ is pre-trained CaffeNet [2] that has been shortened to the fc6 layer which is the first fully connected layer.

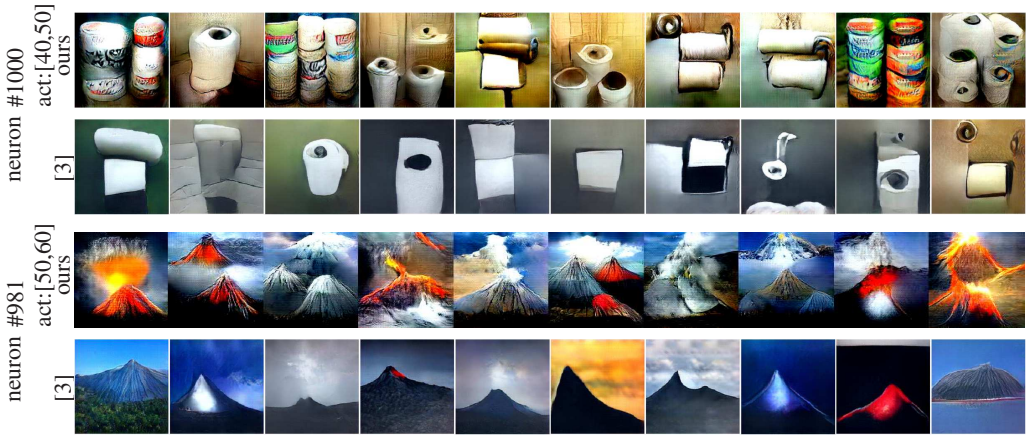

Figure 1: First and third row contains 10 samples picked from 500 samples generated by our sampling approach to cover the inverse set for the neuron number 1000 (represents toilet paper class) and 981 (represents volcano class) respectively. Both neurons are from layer fc8 of CaffeNet [2]. For first row (toilet paper class) the activation range is [40,50] and for third row (volcano class) the range is [50,60]. The second and fourth row shows samples generated for the same neurons by sampling approach from [3]. Activation range for the samples from [3] are not guaranteed to be in any fixed range like ours.

Figure 2: Sampling the intersection of two inverse sets. The sample images from left to right are from the inverse set of neuron 664 (monastery class), of neuron 862 (toilet seat class) and of their intersection, all in the activation range [40,50]. Both neurons are from layer fc8 of CaffeNet [2].

References

[1] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 658–666. MIT Press, Cambridge, MA, 2016.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv:1408.5093 [cs.CV], June 20 2014.

[3] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proc. of the 2017 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'17)*, Honolulu, HI, July 21–26 2017.

[4] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.

[5] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. of the 2nd Int. Conf. Learning Representations (ICLR 2014)*, Banff, Canada, Apr. 14–16 2014.