SAMPLING THE "INVERSE SET" OF A NEURON

Suryabhan Singh Hada *Miguel Á. Carreira-Perpiñán*

Dept. of Computer Science and Engineering, University of California, Merced
{shada, mcarreira-perpinan}@ucmerced.edu

ABSTRACT

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 [1] or using Markov chain Monte Carlo to sample images [2], 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.

Index Terms— Interpretability, deep neural networks, GANs.

1. INTRODUCTION

In the last few years, deep neural networks have shown great success in solving problems in the field of computer science. This leads to a surge in the usage of deep neural networks in real-life applications, making it very important to understand the working of deep neural networks. Some machine learning models are readily interpretable, i.e., by looking at the parameters and structure of the model, we can understand the prediction for a given input. However, this is not the case with deep networks, where the parameters of the model are interleaved in a very complex way. It is very difficult to make the prediction just by looking at the model's parameters.

In a broader sense, we can divide the deep networks' interpretability research into two categories: 1) produce an explanation in the input space that explains network prediction for a given input, and 2) understand what information is retained by the network.

In the first category, we want to understand the network decision for a given instance; it is called *local explanation* [3]. Methods like [4, 5, 6] approximate the deep network locally with a much simple model like decision trees or linear model; and using these simple models, they provide an explanation in terms of input segments (like superpixels in an image). Other works provide similar explanations via sensitivity analysis [7, 8]. One can also find the training instances that most affect the prediction for a given test image [9, 10].

Our work belongs to the second category, where we want to understand what information is retained by the network. This provides a global explanation, i.e., the explanations are independent of the given input. Currently available techniques that are used to understand the parameters of a trained deep neural network involve two major approaches: feature inversion and activation maximization. Feature inversion attempts to project the output of a layer to the input space, to understand what each layer of the network has learned. This is done by backpropagation, gradient descent or training a network [11, 12, 13, 14]. On the other hand, activation maximization involves maximizing the response of a neuron in the network for an input, as done by [15, 1, 16, 17, 18], and more. A quite different approach is based on building a global mimic via a decision tree [19].

Although these approaches work well, they have two major drawbacks. First, they do not consider that the real-life images do not usually have high activations [17]. Thus, producing images that have a very high activation value does not solve the problem entirely to understand what real-life images are preferred by a neuron. Second, they do not account for the fact that multiple images can maximally activate the same neuron. Authors in [18] tried to address this problem by initializing the input image with the mean of a cluster of training images for activation maximization. However, the generated images are noisy and not much diverse. Later, [2] used their plug and play model to generate comparatively diverse and more realistic images using the Markov chain Monte Carlo based sampling approach. Although authors in [2] generate more diverse samples for a single neuron compared to all the previous work, they do not address the issue that real-life images do not usually have high activations. There is no control over the activation value of the generated samples. Besides their Markov chain Monte Carlo based sampling approach is very slow.

In this work, we propose an approach that can improve our ability to understand the behavior of neurons by addressing the drawbacks mentioned above. We achieve this by characterizing the region of input space that excites a given neuron to a certain level; we call this the *inverse set*. In the next section, we describe our approach to address the issues mentioned above.

2. THE INVERSE SET OF A NEURON

2.1. The inverse set of a neuron: definition

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 weight vector ($\mathbf{w}$), bias (b), logistic activation function $\sigma(\mathbf{w}^T\mathbf{x}+b)$ 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

Fig. 1. Samples generated using our sampling approach for the neuron number 981 in the fc8 layer of the CaffeNet [20], which represents volcano class. Each row contains 10 samples picked from 500 samples generated with activation range mentioned on the left side. Generated samples are not just photo-realistic but also very diverse in nature, characterizing input space around certain activation very well. Unlike previous approaches [18, 2]; generated samples not just only contains volcanoes but they also contain lava flowing through water and on land, and the ash cloud after the volcanic eruption. These kind of samples never have been seen before. The first row that contains samples of high activation and the last row which have samples with low activation, both have volcanoes but the presence of lava and smoke create a huge difference in activation value of the neuron. Giving us a clear understanding how the amount of the lava and smoke impact the activation of the neuron, higher the amount of lava and smoke more the activation.

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.

2.2. Sampling the inverse set: an optimization approach

To generate the maximum activation image, the problem can be mathematically formulated as:

$$\arg\max_{\mathbf{x}} f(\mathbf{x}) + \mathcal{R}(\mathbf{x}). \tag{2}$$

Here, real-valued function f gives the activation value of the given neuron for an input image $\mathbf{x}$. $\mathcal{R}$ is a regularizer that makes sure that the generated image $\mathbf{x}$ looks like a real image. This is the same objective function used in [1, 16, 18], and others, where, $\mathcal{R}$ is replaced by their hand-crafted regularizers. As mentioned in [18], it mostly produces the same images for a given neuron. However, to construct the sample $S = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ that covers the inverse set, generated images should be different from each other. So, we propose the following formulation to construct S of size n as a constrained optimization problem:

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

This makes sure that the samples are different from each other and 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 [21] when a feature vector $\mathbf{c}$ is passed as an input. Then we get:

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

We observe that using Euclidean distances directly on the generated images $\mathbf{G}(\mathbf{c})$ 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 $\mathbf{G}(\mathbf{c}_1), \ldots, \mathbf{G}(\mathbf{c}_n)$:

$$\arg\max_{\mathbf{c}_1,\mathbf{c}_2,\cdots,\mathbf{c}_n} \sum_{i,j=1}^{n} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_j))\|_2^2$$
$$\text{s.t.} \quad z_1 \leq f(\mathbf{G}(\mathbf{c}_1)), \ldots, f(\mathbf{G}(\mathbf{c}_n)) \leq z_2. \tag{5}$$

Now to generate the samples, initialize $\mathbf{c}$ with random values and then optimize eq. (5) using the augmented Lagrangian [22].

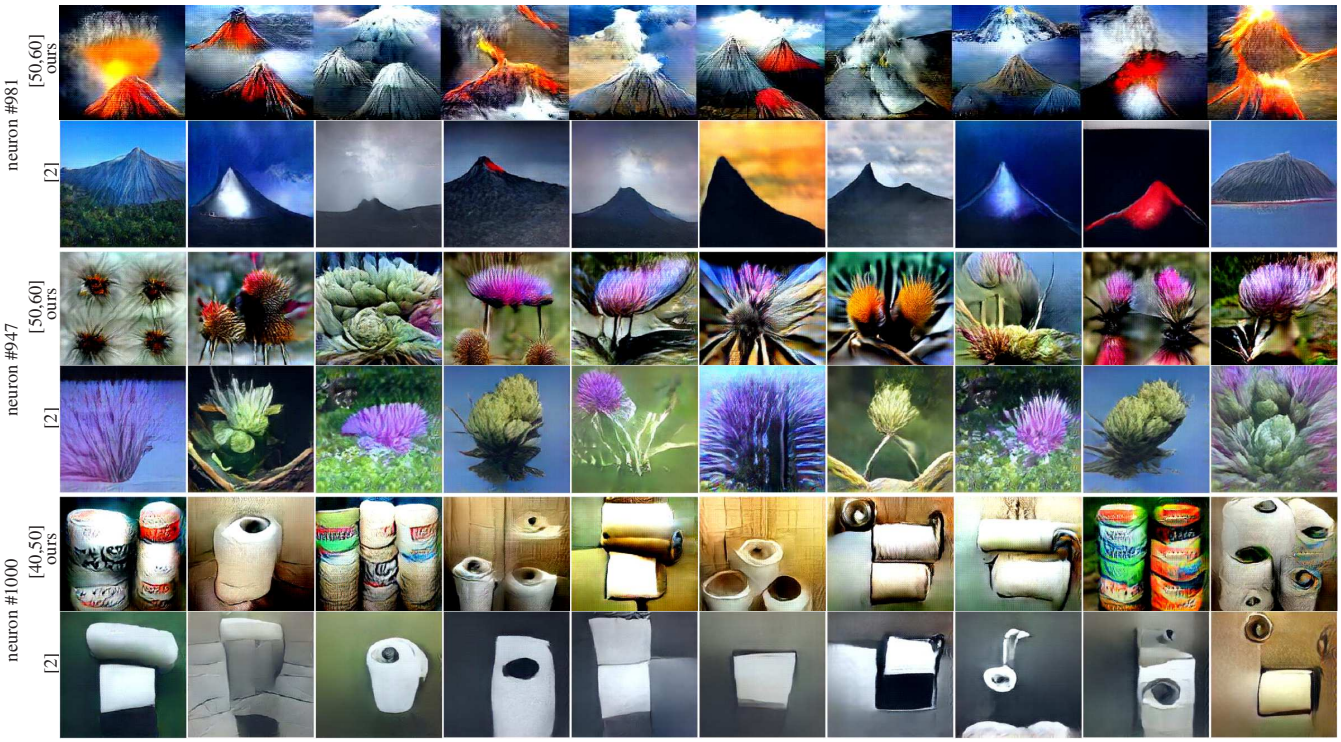

Fig. 2. The first, third and fifth row contains 10 samples picked from 500 samples generated by our sampling approach to cover the inverse set for the neuron number 981 (represents volcano class), 947 (represents cardoon class) and 1000 (represents toilet paper class) respectively. All three neurons are from layer fc8 of CaffeNet [20]. For the first row (volcano class) the activation range is [50,60], for the third (cardoon class) and the fifth row (toilet paper class) the range is [40,50]. The second, fourth and sixth row shows samples generated for the same neurons by sampling approach from [2]. Activation range for the samples from [2] is not guaranteed to be in any fixed range like ours.

Neuron in fc8	Ours	PPGN [2]
981(volcano class)	5.58	4.82
947(cardoon class)	6.00	4.99
1000(toilet paper class)	5.83	4.79

Table 1. Comparison between the proposed method and PPGN; in terms of mean pairwise Euclidean distance between the codes ($\mathbf{E}(\mathbf{G}(\mathbf{c}_i))$) of the generated samples (all 500 samples in fig. 2). Higher values mean generated samples are more diverse.

In theory eq. (5) is enough to generate the sample S. But in practice, as the n gets larger, which is required to correctly sample the inverse set, eq. (5) poses two issues. First, because of the quadratic complexity of the objective function over the number of samples n, it is computationally expensive to generate many samples. Second, since it involves optimizing all code vectors ($\mathbf{c}$) together, for larger n it is not possible to fit all in the GPU memory. In the next section, we describe a much faster and less computationally demanding approach to create the inverse set.

2.3. Sampling in feasible region

To sample the set S (eq. (5)) efficiently, we apply two approximations. First, 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 the 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* ($\mathbf{C}_0$)) we optimize eq. (5), 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 are generated by the following equation:

$$\begin{aligned}\underset{\mathbf{c}_1,\mathbf{c}_2,\cdots,\mathbf{c}_K}{\arg\max} & \sum_{i,j=1}^{K} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_j))\|_2^2 + \\ & \sum_{i=1}^{K}\sum_{y=1}^{|C_0|} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_y))\|_2^2 \\ \text{s.t.} \quad & z_1 \leq f(\mathbf{G}(\mathbf{c}_1)), \ldots, f(\mathbf{G}(\mathbf{c}_K)) \leq z_2 \text{ and } \mathbf{c}_y \in \mathbf{C}_0. \end{aligned} \quad (6)$$

The first part of the equation $\sum_{i,j=1}^{K} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_j))\|_2^2$ is similar to that of eq. (5), forces the samples to be apart from each other. On the other hand, the second part of the equation $\sum_{i=1}^{K}\sum_{y=1}^{|C_0|} \|\mathbf{E}(\mathbf{G}(\mathbf{c}_i)) - \mathbf{E}(\mathbf{G}(\mathbf{c}_y))\|_2^2$ makes sure that the generated samples are far apart from the previous ones. The constraints keeps the generated samples in the feasible region.

Now to pick next K samples, we initialize them to the previous K samples ($\mathbf{C}_0$) and take a single gradient step in the augmented Lagrangian optimization of eq. (6). This gives K new samples ($\mathbf{G}(\mathbf{c}_i)$) which we fix, and the process is repeated until we generate the desired n samples. Note that, here we are not trying to optimize anything. We are using eq. (6) to take steps inside the feasible region. It is like taking a random walk, but here we are taking steps inside the feasible region in a way that the next step gives the samples which are different from each other as well as from the ones we already have. Another good part of this approach is that unlike the previous sampling approaches, eq. (6) allows us to pick samples in parallel

Fig. 3. 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.

which further increases the speed of sampling [1].

3. EXPERIMENTS

All previous works to understand the neural networks took CaffeNet [20] a minor variant of AlexNet [23] as the main subject. So, we also used a pre-trained CaffeNet [20] for our experiments, which had been trained on ImageNet dataset [24]. We use Matcovnet [25] for all our experiments. Using previous papers' [17, 2], naming convention we will also call last three fully connected layers in CaffeNet [20] as fc6, fc7, and fc8. fc8 is the last layer before softmax. In all our experiments $\mathbf{E}$ is pre-trained CaffeNet [20]. However, the network has been shortened to the fc6 layer which is the first fully connected layer. The output of the $\mathbf{E}$ is a vector of size 4096. $\mathbf{G}$ is a pre-trained generative network from [2] [2]. $\mathbf{G}$ [3] has been trained to generate images from a feature vector of size 4096 more specifically output of fc6 layer of CaffeNet [20].

To do a fair comparison with [2], we pick the neuron 981 which represents "Volcano" class. We run our algorithm for six times to generate 500 samples to cover inverse set for this neuron corresponding to different activation levels. Fig. 1 shows results of this experiment. We pick 10 samples from generated 500 samples for each $[z_1, z_2]$. Unlike the previous visualization approaches [2, 17, 18] (second row of fig. 2), the generated samples are far more diverse and rich in information. Our approach allows us to look at the samples which excite the neuron at different activation levels; this was not possible before. In doing so, it uncovers samples which have never been seen before. For instance, images which do not even contain volcano, instead contain lava flowing through water, but still excite the neuron.

Fig. 2 shows some of the samples generated by our algorithm for a certain activation range with some other neurons and their comparison with [2]. To do a fair comparison with [2], we generate samples as it is described in the supplementary section of the paper [2]. We run 10 sampling chains conditioned on various classes each for 200 steps, to produce 2000 samples for each class. We picked 1 sample from each 200 steps to provide as much diversity as possible in the samples. Second row shows the generated samples for neuron number 947 which represents "Cardoon" class for $z_2 = 50$ and $z_1 = 40$. The generated samples not only have more diverse color distribution compared to the [2] (fig. 2 fourth row) but also show samples of different shapes and sizes, that can only be seen in real-life images.

To further test whether our sampling truly generates diverse samples or not, we pick a rather difficult class: "Toilet paper", neuron number 1000. The reason for the difficulty is as the realistic looking samples cannot just differ by having a different color like in the case of "Cardoon" class; they all should have the white color. Sampling method should pick the samples which are of different shape or quantity. The fifth row in fig. 2 shows our results. The generated samples are very diverse; they not only show just a toilet roll or toilet rolls hanged with a hanger but also show packed toilet rolls. In fact, samples contain packages with different shapes, labels, and even quantity. The first and ninth sample in the fifth row of fig. 2 shows rolls packed in two packages but have altogether different packaging labels, while the third sample shows toilet rolls packed in 3 packages which also have different packaging label than other two. Our sampling method was even able to pick samples such that the number of toilet rolls is different, as shown in the second, fourth and tenth sample. These type of samples can only be seen in real-life images thus showing how perfectly our sampling method covers the inverse set. On the other hand, samples generated by using PPGN [2] (sixth row of fig. 2) mostly contain rolls which are standalone or attached to a holder, clearly not much diverse. This diversity also translates numerically, as shown in table 1. Our approach also allows us to visualize the intersection of multiple inverse sets [26]. We show one such example in fig. 3.

4. DISCUSSION

Admittedly, the goal of understanding what a neuron in a deep neural network may be representing is not a well defined problem. It may well be that a neuron does represent a specific concept, but one which is very difficult to grasp for a human; or that one should look at what a group of neurons may be representing. That said, for some neurons their preferred response does correlate well with intuitive concepts or classes, such as the volcano or monastery examples we give. Our approach is to characterize a neuron's preference by a diverse set of examples it likes, which is something that people sometimes do in order to explain a subjective concept to each other. Also, it may be possible to extract specific concepts from this set of examples using data analysis techniques.

5. CONCLUSION

In conclusion, we propose a very simple, yet effective generalized approach to understand the neurons in a deep neural network: an approach that does not involve activation maximization or feature inversion, but instead characterizes the region of input space around different activation values of the neuron of interest. Thus overcoming the shortcomings of current visualization approaches and providing a great deal of understanding what parts of an image are responsible for impacting the activation of a neuron. This eventually helps us to have a much better understanding about the nature of a neuron in deep neural networks. We also provide a simpler, faster, and hyper-parameter free sampling method which generates far more diverse samples than previously proposed methods. Our sampling method also has more general applicability; just by modifying the constraints, it can also be used for high dimensional sampling in other domains.

[1] In most of our experiments $K = 10$ and $n = 500$. It took 85 gradient step of eq. (6) to generate rest of the 490 $(n - K)$ samples with $K = 10$ seeds. These are just a few extra gradient steps than the minimum required 49 gradient steps.

[2] `https://github.com/Evolving-AI-Lab/ppgn`

[3] The only reason we use same network as [2] is to do fair comparison of diversity in the generated images. Proposed method is independent of $\mathbf{G}$, so it can be replaced with any state-of-the-art generative network.

6. REFERENCES

[1] Karen Simonyan, Andrea Vedaldi, and Andrew 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.

[2] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason 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, pp. 3510–3520.

[3] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, pp. 93, May 2018.

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proc. of the 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2016)*, San Francisco, CA, Aug. 13–17 2016, pp. 1135–1144.

[5] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, May 2016.

[6] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini, "Factual and counterfactual explanations for black box decision making," *IEEE Access*, vol. 34, no. 6, pp. 14–23, Nov. – Dec. 2019.

[7] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. 16th Int. Conf. Computer Vision (ICCV'17)*, Venice, Italy, Dec. 11–18 2017, pp. 618–626.

[8] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje, "Learning important features through propagating activation differences," in *Proc. of the 34th Int. Conf. Machine Learning (ICML 2017)*, Doina Precup and Yee Whye Teh, Eds., Sydney, Australia, Aug. 6–11 2017, pp. 3145–3153.

[9] Pang Wei Koh and Percy Liang, "Understanding black-box predictions via influence functions," in *Proc. of the 34th Int. Conf. Machine Learning (ICML 2017)*, Doina Precup and Yee Whye Teh, Eds., Sydney, Australia, Aug. 6–11 2017, pp. 1885–1894.

[10] Chih-Kuan Yeh, Joon Sik Kim, Ian E. H. Yen, and Pradeep Ravikumar, "Representer point selection for explaining deep neural networks," in *Advances in Neural Information Processing Systems (NEURIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. 2018, vol. 31, MIT Press, Cambridge, MA.

[11] Aravindh Mahendran and Andrea Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *Int. J. Computer Vision*, vol. 120, no. 3, pp. 233–255, Dec. 2016.

[12] Donglai Wei, Bolei Zhou, Antonio Torralba, and William Freeman, "Understanding intra-class knowledge inside CNN," arXiv:1507.02379, July 15 2015.

[13] Matthew D. Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th European Conf. Computer Vision (ECCV'14)*, Zürich, Switzerland, Sept. 6–12 2014, pp. 818–833.

[14] Alexey Dosovitskiy and Thomas Brox, "Inverting visual representations with convolutional networks," in *Proc. of the 2016 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, NV, June 26 – July 1 2016.

[15] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Visualizing higher-layer features of a deep network," Tech. Rep. 1341, Université de Montréal, June 2009.

[16] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, "Understanding neural networks through deep visualization," in *ICML Workshop on Deep Learning*, 2015.

[17] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems (NIPS)*, D. D. Lee, M. Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and R. Garnett, Eds. 2016, vol. 29, pp. 3387–3395, MIT Press, Cambridge, MA.

[18] Anh Nguyen, Jason Yosinski, and Jeff Clune, "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks," in *ICML Workshop on Visualization for Deep Learning*, 2016.

[19] Suryabhan Singh Hada, Miguel Á. Carreira-Perpiñán, and Arman Zharmagambetov, "Sparse oblique decision trees: A tool to understand and manipulate neural net features," arXiv:2104.02922, Apr. 7 2021.

[20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," arXiv:1408.5093 [cs.CV], June 20 2014.

[21] Alexey Dosovitskiy and Thomas Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, D. D. Lee, M. Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and R. Garnett, Eds. 2016, vol. 29, pp. 658–666, MIT Press, Cambridge, MA.

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

[23] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. 2012, vol. 25, pp. 1106–1114, MIT Press, Cambridge, MA.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of the 2009 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'09)*, Miami, FL, June 20–26 2009, pp. 248–255.

[25] Andrea Vedaldi and Karel Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conference on Multimedia*, 2015, pp. 689–692.

[26] Suryabhan Singh Hada and Miguel Á. Carreira-Perpiñán, "Sampling the "inverse set" of a neuron: An approach to understanding neural nets," arXiv:1910.04857, Sept. 27 2019.