

# An Efficient Explorative Sampling Considering the Generative Boundaries of Deep Generative Neural Networks

**Giyoung Jeon,\* Haedong Jeong\***

Ulsan National Institute of Science and Technology  
50, UNIST-gil, Ulsan 44919, Republic of Korea  
{giyoung, haedong}@unist.ac.kr

**Jaesik Choi†**

Korea Advanced Institute of Science and Technology  
291 Daehak-ro, Daejeon 34141, Republic of Korea  
jaesik.choi@kaist.ac.kr

## Abstract

Deep generative neural networks (DGNNs) have achieved realistic and high-quality data generation. In particular, the adversarial training scheme has been applied to many DGNNs and has exhibited powerful performance. Despite of recent advances in generative networks, identifying the image generation mechanism still remains challenging. In this paper, we present an explorative sampling algorithm to analyze generation mechanism of DGNNs. Our method efficiently obtains samples with identical attributes from a query image in a perspective of the trained model. We define generative boundaries which determine the activation of nodes in the internal layer and probe inside the model with this information. To handle a large number of boundaries, we obtain the essential set of boundaries using optimization. By gathering samples within the region surrounded by generative boundaries, we can empirically reveal the characteristics of the internal layers of DGNNs. We also demonstrate that our algorithm can find more homogeneous, the model specific samples compared to the variations of  $\epsilon$ -based sampling method.

## 1 Introduction

The primary objective of a generative model is to generate realistic data. Recently proposed adversarial training schemes, such as generative adversarial networks (GANs), have exhibited remarkable performance not only in terms of the quality of each instance but also the diversity of the generated data. Despite those improvements, the generation mechanism inside the generative models is not well-studied.

In general, a generative model maps a point in the latent space to a sample in the data space. In other words, data instances are embedded as latent vectors in a perspective of the trained generative model. A latent space is divided by boundaries derived from the structure of the model, where the vectors in the space represent the generation information according to which side of boundaries they are placed. We utilize these characteristics to examine the generation mechanism of the model.

\*Equal Contribution

†Corresponding Author

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

When we select an internal layer and a latent vector in the DGNNs, there exists the corresponding region which is established by a set of boundaries. Samples in this region have the same activation pattern and deliver similar generation information to the next layer. The details of the delivered information can be identified indirectly by comparing the generated outputs from these samples. Given a DGNN trained to generate human faces, for example, if we identify the region in which samples share a certain hair color but vary in others characteristics (eye, mouth, etc.), such a region would be related to the generation of the same hair color.

However, it is non-trivial to obtain samples from the region with desired properties of DGNNs because (1) thousands of generative boundaries are involved in the generation mechanism and (2) a linear modification in the input dimension may cause a highly non-linear change in the internal units and the output. Visiting the previous example again, there may exist regions with different hair colors, distinct attributes, or their combinations. Furthermore, a small linear modification of the vector in the latent space may change the entire output (Szegedy et al. 2013). To overcome this difficulty, an efficient algorithm to identify the appropriate region and explore the space are necessary.

In this paper, we propose an efficient, explorative sampling algorithm to reveal the characteristics of the internal layer of DGNNs. Our algorithm consists of two steps: (1) to handle a large number of boundaries in DGNNs, our algorithm approximates the set of critical boundaries of the query which is the given latent vector using Bernoulli dropout approach (Chang et al. 2018); (2) then our algorithm efficiently obtains samples which share same attributions as the query in a perspective of the trained DGNNs by expanding the tree-like exploring structure (LaValle 1998) until it reaches the boundaries of the region.

The advantages of our algorithm are twofold: (1) it can guarantee sample acceptance in high dimensional space where the rejection sampling based on the Monte Carlo method easily fails when the region area is unknown; (2) it can handle sampling strategy in a perspective of the model where the commonly used  $\epsilon$ -based sampling (Erhan, Courville, and Bengio 2010) is not precise to obtain samples considering complex non-spherical generative boundaries

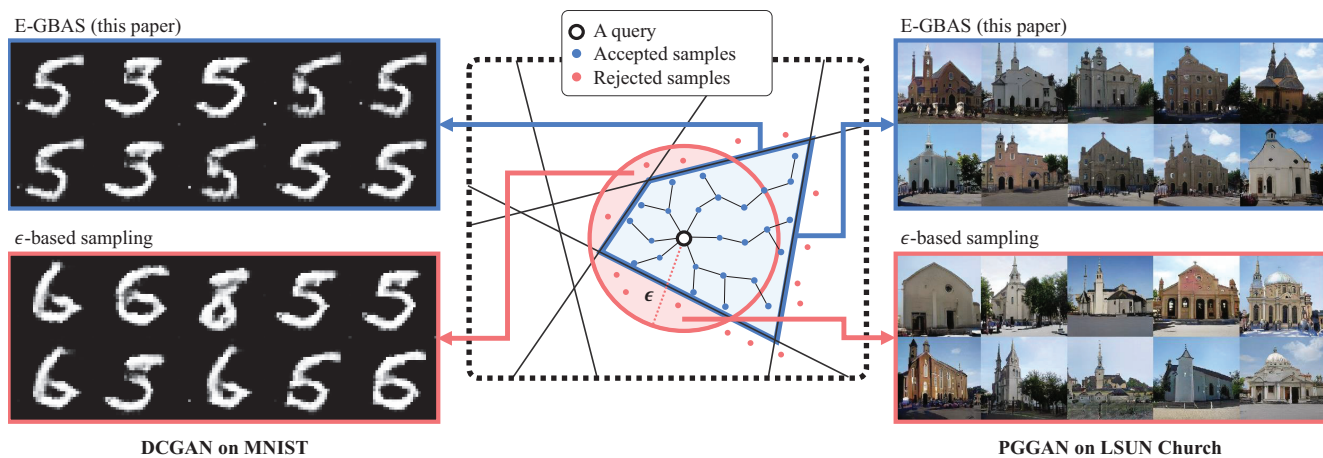


Figure 1: Illustrative examples of our explorative generative boundary aware sampling (E-GBAS) and  $\epsilon_{L_2}$ -based sampling.

(Laugel et al. 2019). We experimentally verify that our algorithm obtains more consistent samples compared to  $\epsilon$ -based sampling methods on deep convolutional GANs (DCGAN) (Radford, Metz, and Chintala 2015) and progressive growing of GANs (PGGAN) (Karras, Aila, and Laine 2015).

## 2 Related Work

**Generative Adversarial Networks** The adversarial training between a generator and a discriminator has highly improved the quality and diversity of samples generated by DGNNs (Goodfellow et al. 2014). Many generative models have been proposed to generate room images (Radford, Metz, and Chintala 2015) and realistic human face images (Karras, Aila, and Laine 2015; Karras, Laine, and Aila 2019). Despite those improvements, the generation mechanisms of the GANs are not clearly analyzed yet. Recent results revealed that the relationship between the input latent space and the output data space in a trained GAN by showing a manipulation in the latent vectors changes attributes in the generated data (Radford, Metz, and Chintala 2015; Zhu et al. 2016). Generation roles of some neural nodes in a trained GAN are identified with the intervention technique (Bau et al. 2019).

**Explaining deep neural networks** One can explain an output of neural networks by the sensitivity analysis, which aims to figure out which portion of an input contributes to the output. The sensitivity can be calculated by class activation probabilities (Zhou et al. 2016), relevance scores (Montavon et al. 2017) or gradients (Selvaraju et al. 2017). DeconvNet (Zeiler and Fergus 2014), LIME (Ribeiro, Singh, and Guestrin 2016) and SincNet (Ravanelli and Bengio 2018) trains a new model to explain the trained model. Geometric analysis could also reveal the internal structure indirectly (Montufar et al. 2014; Lei et al. 2018; Fawzi et al. 2018). The activation maximization (Erhan, Courville, and Bengio 2010), or GANs (Nguyen et al. 2016) have been used to explain the neural network by using examples. Our method is an example-based explanation which brings a new geometric perspective to analyze DGNNs.

## Geometric analysis on the inside of deep neural networks

Geometric analysis attempts to analyze the internal working process by relating the geometric properties, such as boundaries dividing the input space or manifolds along the boundaries, to the output of the model. The depth of a network with nonlinear activations was shown to contribute to the formation of boundary shape (Montufar et al. 2014). This property makes complex, non-convex regions surrounded by boundaries derived by internal layers. Although such regions are complicated, each region for a single classification in DNN classifiers is shown to be topologically connected (Fawzi et al. 2018). It has also been shown that the manifolds learned by DNNs and distributions over them are highly related to the representation capability of a network (Lei et al. 2018).

## Example-based explanation of the decision of the model

Activation maximization is one of example-based methods to visualize the preferred inputs of neurons in a layer and according patterns in hidden layers (Erhan, Courville, and Bengio 2010). The learned deep neural representation can be denoted by preferred inputs because it is related to the activation of specific neurons (Nguyen et al. 2016). The reliability of examples for explanation also has been argued considering the connectivity among the justified samples (Laugel et al. 2019).

## 3 Generative Boundary Aware Sampling in Deep Generative Neural Networks

This section presents our main contribution, the explorative generative boundary aware sampling (E-GBAS) algorithm, which can obtain samples sharing the identical attributes from the perspective of the DGNNs. Initially, we define the terms used in our algorithm. Then we explain E-GBAS which comprises of (1) an approximate representation of generative boundaries and (2) an efficient stochastic exploration to obtain samples in the complex, non-convex generative region.

### 3.1 Deep Generative Neural Networks

Although there are various architecture of DGNNs, we represent the DGNNs in a unified form. Given DGNNs with

$L$  layers, the function of DGNNs model  $G$  is decomposed into  $G(z) = f_L(f_{L-1}(\dots(f_1(z)))) = f_{L:1}(z)$ , where  $z$  is a vector in the latent space  $\mathcal{Z} \subset \mathbb{R}^{D_z}$ .  $f_{i:1}^i(\cdot)$  denotes the value of  $i$ -th element and  $f_{i:1}(z) \in \mathbb{R}^{D_i}$ . In general, the operation  $f_i(\cdot)$  includes linear transformations and a non-linear activation function.

### 3.2 Generative Boundary and Region

The latent space of the DGNNs is divided by hypersurfaces learned during the training. The networks make the final generation based on these boundaries. We refer these boundaries as the generative boundaries.

**Definition 1 (Generative Boundary (GB)).** *The  $i$ -th generative boundary at the  $l$ -th layer is defined as*

$$B_l^i = \{z | f_{i:1}^i(z) = 0, z \in \mathcal{Z}\}.$$

In general, there are numerous boundaries in the  $l$ -th layer of the network and the configuration of the boundaries comprises the region. Because we are mainly interested in the region created by a set of boundaries, we denote the definition of halfspace which is a basic component of the region.

**Definition 2 (Halfspace).** *Let a halfspace indicator  $\mathbf{V}_l \in \{-1, 0, +1\}^{D_l}$  for the  $l$ -th layer. Each element  $V_l^i$  indicates either or both of two sides of the halfspace divided by the  $i$ -th GB. We define the halfspace as*

$$\mathcal{H}_l^i = \begin{cases} \mathcal{Z} & \text{if } V_l^i = 0 \\ \{z | V_l^i f_{i:1}^i(z) \geq 0\} & \text{if } V_l^i \in \{-1, +1\}. \end{cases}$$

The region can be represented by the intersection of each halfspace in the  $l$ -th layer. For the case where  $V_l^i = 0$ , the halfspace is defined as the entire latent space, so  $i$ -th GB does not contribute to comprise the region.

**Definition 3 (Generative Region (GR)).** *Given a halfspace indicator  $\mathbf{V}_l$  in the  $l$ -th layer, let the set of according halfspaces  $\mathbf{H} = \{\mathcal{H}_l^1, \dots, \mathcal{H}_l^{D_l}\}$ . Then the generative region  $GR_{V_l}$  is defined as*

$$GR_{V_l} = \cap_{\mathcal{H} \in \mathbf{H}} \mathcal{H}.$$

For a network with a single layer ( $l=1$ ), the generative boundaries are linear hyperplanes. The generative region is constructed by those boundaries and appears as a convex polytope. However, if the layers are stacked with nonlinear activation functions ( $l>1$ ), the generative boundaries are bent, so the generative region will have a complicated non-convex shape (Montufar et al. 2014; Raghu et al. 2017).

### 3.3 Smallest Supporting Generative Boundary Set

Decision boundaries have an important role in classification task, as samples in the same decision region have the same class label. In the same context, we manipulate the generative boundaries and regions of the DGNNs.

Specifically, we want to collect samples that are placed in the same generative region and have identical attributes in a perspective of the DGNNs. To define this property formally, we first define the condition under which the samples share the neural representation.

### Definition 4 (Neural Representation Sharing (NRS)).

*Given a pair of latent vectors  $z_i, z_j \in \mathcal{Z}$  satisfies the neural representation sharing condition in  $l$ -th layer if*

$$\text{sign}(f_{i:1}^k(z_i)) = \text{sign}(f_{i:1}^k(z_j)), \quad \forall k \in \{1, 2, \dots, D_l\}.$$

It is practically challenging to find samples that satisfy the above condition, because a large number of generative boundaries exist in the latent space, as shown in Figure 2(a). Various information represented by thousands of generative boundaries makes it difficult to identify which boundary is in charge of each piece of information. We relax the condition of the neural representation sharing by considering a set of the significant boundaries.

**Definition 5 (Relaxed NRS).** *Given a subset  $S$  and a pair of latent vectors  $z_i, z_j \in \mathcal{Z}$  satisfies the relaxed neural representation sharing condition if*

$$\text{sign}(f_{i:1}^k(z_i)) = \text{sign}(f_{i:1}^k(z_j)), \quad \forall k \in S \subset \{1, 2, \dots, D_l\}.$$

Then, we must select important boundaries for the relaxed NRS in the  $l$ -th layer. We believe that not all nodes deliver important information for the final output of the model as some nodes could have low relevance of information (Morcos et al. 2018). Furthermore, it has been shown that a subset of features mainly contributes to the construction of outputs in GAN (Bau et al. 2019). We define the *smallest supporting generative boundary set (SSGBS)*, which minimizes the influences of minor (non-critical) generative boundaries.

### Definition 6 (Smallest Supporting Generative Boundary Set).

*Given the generator  $G$  and a query  $z_0 \in \mathcal{Z}$ , for  $l$ -th layer and any real value  $\delta > 0$ , if there exists an indicator  $\mathbf{V}_l^*$  such that*

$$\|G(z) - G(z_0)\| \leq \delta, \quad z \in \{z | f_{l-1:1}(z) \in GR_{\mathbf{V}_l^*}\}$$

*and there is no  $\mathbf{V}_l'$  where  $\|\mathbf{V}_l'\|_1 < \|\mathbf{V}_l^*\|_1$  such that*

$$\|G(z') - G(z_0)\| \leq \delta, \quad z' \in \{z' | f_{l-1:1}(z') \in GR_{\mathbf{V}_l'}\}$$

*then we denote a set  $\mathcal{B}_{V_l^*} = \{B_l^i | V_l^{*i} \neq 0, i \in \{1, 2, \dots, D_l\}\}$  as the smallest supporting generative boundary set (SSGBS).*

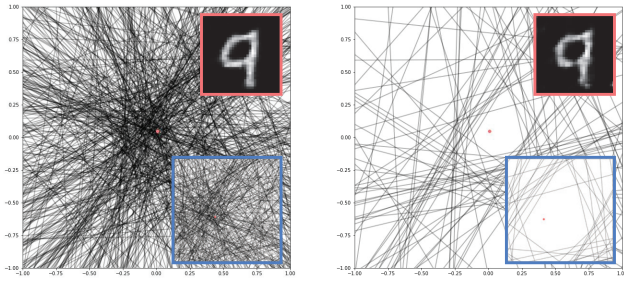
*In the same context, we denote the generative region  $GR_{\mathbf{V}_l^*}$  that corresponds to the SSGBS as the smallest supporting generative region (SSGR).*

It is impractical to explore all the combinations of boundaries to determine the optimal SSGBS, owing to the exponential combinatoric search space.<sup>1</sup> To avoid this problem, we used the Bernoulli dropout approach (Chang et al. 2018) to obtain the SSGBS. We define this dropout function as  $\phi(h, \theta) = h \odot m$ ,  $m \sim \text{Ber}(\theta)$ , where  $\odot$  is an element-wise multiplication. We optimize  $\theta$  to minimize the loss function  $L$ , which quantifies the degradation of generated image with the sparsity of Bernoulli mask.

$$\begin{aligned} \theta^* &= \arg \min_{\theta} L(z_0, l, \theta) \\ &= \arg \min_{\theta} \|f_{L:l+1}(\phi(f_{l:1}(z_0), \theta)) - G(z_0)\| + \lambda \|\theta\|_1 \end{aligned} \quad (1)$$

<sup>1</sup>For example, a simple fully connected layer with  $N$  outputs generates up to  $2^N$  generative boundary sets.





(a) Original generative boundaries and generated digit image of the query (b) Generative boundaries in the SSGBS and generated digit image of the query

Figure 2: Results of optimization of the Bernoulli parameter  $\theta$  for the given arbitrary query  $z$  in the trained DCGAN on MNIST with 2-D latent space (Radford, Metz, and Chintala 2015; LeCun, Cortes, and Burges 2010). The red box denotes the generated digit image and the blue box denotes the magnified area nearby the query. (a) shows all generative boundaries in the first hidden layer ( $l=1$ ). (b) shows SSGBS after optimization with constraint  $p > 0.5$ .

We iteratively update the parameter using gradient descent to minimize the loss function in Equation (4). Then we obtain the SSGBS  $\mathcal{B}_{V_l^*}$  from the optimized Bernoulli parameter  $\theta$  with a proper threshold in the  $l$ -th layer. For each iteration, we apply the element-wise multiplication between  $f_{l:1}(z_0)$  and sampled mask  $m \sim \text{Ber}(\theta)$  to obtain masked feature value and feed it to obtain the modified output.

**Algorithm 1** Bernoulli Mask Optimization (BerOpt)

Input:  $z_0$ : a query,  $G(\cdot) = f_{L:1}(\cdot)$ : a DGNN model,  $l$ : a target layer  
Output:  $\theta$ : the optimized Bernoulli parameter for SSGBS

- 1: Initialize  $\theta \in [0, 1]^{D_l}$
- 2:  $h_0 = f_{l:1}(z_0)$
- 3: **while** not converge **do**
- 4:   Sample  $m \sim \text{Ber}(\theta)$
- 5:    $h_m = h_0 \odot m$
- 6:    $x_0 = f_{L:l+1}(h_0)$ ,  $x_m = f_{L:l+1}(h_m)$
- 7:   Compute loss  $L(z_0, l, \theta)$
- 8:   Update  $\theta$  with  $\nabla_{\theta} L$
- 9: **return**  $\theta$

After obtaining the optimal Bernoulli parameter  $\theta^*$ , we first define an optimal halfspace indicator  $V_l^*$  with the proper probability threshold (e.g.,  $p = 0.5$ ). We set the value of elements in  $V_l^*$  to zero for removing GBs which have minor contributions to the generation mechanism. That is,

$$V_l^{*i} = \mathbb{I}(\theta^* > p) \cdot \text{sign}(f_{l:1}^i(z_0))$$

where  $\mathbb{I}$  is indicator function. Representing SSGBS  $\mathcal{B}_{V_l^*}$  and SSGR  $GR_{V_l^*}$  is straightforward from the Definition 6 with  $V_l^*$ . Figure 2 shows the generative boundaries and the generated digit of the SSGBS without and with the optimized Bernoulli parameter  $\theta^*$  with  $p > 0.5$ . The generated digits

indicate that the effect of the removal of minor generative boundaries on the output is not significant.

**3.4 Explorative Generative Boundary Aware Sampling**

After obtaining SSGR  $GR_{V_l^*}$ , we gather samples in the region and compare the generated outputs of them. Because the  $GR_{V_l^*}$  possesses a complicated shape, simple neighborhood sampling methods such as  $\epsilon$ -based sampling cannot guarantee exploration inside the  $GR_{V_l^*}$ . To guarantee the relaxed NRS, we apply the GB constrained exploration algorithm inspired by the rapidly-exploring random tree (RRT) algorithm (LaValle 1998), which is invented for the robot path planning in complex configuration spaces. We refer to the modified exploration algorithm as generative boundary constrained RRT (GB-RRT). Figure 3(a) depicts the explorative trajectories of GB-RRT.

**Algorithm 2** Generative boundary constrained rapidly-exploring random tree (GB-RRT)

Input:  $z_0$ : a query,  $GR_{V_l^*}$ : SSGR  
Parameters:  $I$ : a sampling interval,  
 $N$ : a maximum number of iterations,  $\delta$ : a step size  
Output:  $Q$ : samples in the SSGR

- 1: Initialize queue  $Q_0 = \{z_0\}$
- 2: **for**  $i = 1 \dots N$  **do**
- 3:   Sample  $z_i \sim U(z_0 - I, z_0 + I)$
- 4:    $q_i = \text{nearest}(Q_{i-1}, z_i)$
- 5:    $z'_i = (z_i - q_i) / \|z_i - q_i\| * \delta + q_i$
- 6:   **if**  $z' \in GR_{V_l^*}$  **and**  $\|z' - \text{nearest}(Q_{i-1}, z')\| > \delta$
- 7:    **then**  $Q_i = Q_{i-1} \cup \{z'\}$
- 8: **return**  $Q_N$

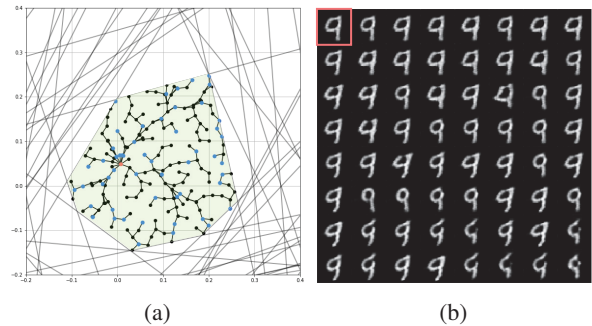


Figure 3: (a) Visualization of explorative trajectories of GB-RRT for a given query (red dot) in the first hidden layer ( $l=1$ ) of DCGAN-MNIST and (b) generated outputs from uniform randomly chosen samples (blue dot in (a)). The red box denotes the generated output of the query.

We name the entire sample exploration process for DGNNs which is comprised of finding SSGBS in arbitrary layer and efficiently sampling in SSGR as explorative generative boundary aware sampling (E-GBAS).

---

**Algorithm 3** Explorative generative boundary aware sampling (E-GBAS)
 

---

Input:  $z_0$ : a query,  $G(\cdot) = f_{L:1}(\cdot)$ : DGNN model,  
 $l$ : a target layer,  $p$ : threshold for SSGBS selection  
 Output:  $Z$ : a set of samples in the same SSGR of  $z_0$

- 1: Optimize  $\theta^* = \mathbf{BerOpt}(z_0, G, l)$
- 2: Compute  $\mathbf{V}_l^* = [V_l^{*1}, \dots, V_l^{*D_l}]^T$   
 where  $V_l^{*i} = \mathbb{I}(\theta^* > p) \cdot \text{sign}(f_{l:1}(z_0))$
- 3: Derive  $GR_{\mathbf{V}_l^*}$
- 4: Sample a set  $Z = \mathbf{GB-RRT}(z_0, GR_{\mathbf{V}_l^*})$
- 5: **return**  $Z$

---

## 4 Experimental Evaluations

This section presents analytical results of our algorithm and empirical comparisons with variants of  $\epsilon$ -based sampling method. We select three different DGNNs; (1) DCGAN (Radford, Metz, and Chintala 2015) with the wasserstein distance (Arjovsky, Chintala, and Bottou 2017) trained on MNIST, (2) PGGAN (Karras, Aila, and Laine 2015) trained on the church dataset of LSUN (Yu et al. 2015) and (3) the celebA dataset (Liu et al. 2015).

The  $\epsilon$ -based sampling collects samples based on  $L_p$  distance metric. We choose  $L_2$  and  $L_\infty$  distance as baseline, and sample in each  $\epsilon$ -ball centered at the query. In practice, the value of  $\epsilon$  is manually selected. We use the set of accepted samples and rejected samples,  $Z_{accept}$  and  $Z_{reject}$ , obtained by the E-GBAS to set the  $\epsilon$  for fair comparisons. We set the average of accepted samples  $z_{avg}$  which can represent the middle point of the SSGR, then we calculate  $\epsilon_{L_2}$  with min/max distance between  $z_{avg}$  and  $Z_{reject}$  as,

$$\epsilon_{L_2} = \frac{1}{2} \left( \max_{z \in Z_{reject}} \|z_{avg} - z\| + \min_{z \in Z_{reject}} \|z_{avg} - z\| \right).$$

Figure 4 indicates the visualization of calculating  $\epsilon$  in the DCGAN-MNIST. After  $\epsilon_{L_2}$  is set,  $\epsilon_{L_\infty}$  are determined to have the same volume as the  $\epsilon_{L_2}$ -ball. Figure 5 shows the geometric comparisons of each sampling method in the first hidden layer ( $l=1$ ) of DCGAN-MNIST.

### 4.1 Qualitative Comparison of E-GBAS and $\epsilon$ -based Sampling

We first demonstrate how the generated samples vary if they are inside or outside of the obtained GR. As shown in Figure 1, we mainly compare the samples generated from E-GBAS (blue region) to the samples from the  $\epsilon_{L_2}$ -based sampling (red region). A given query and a target layer, E-GBAS explores the SSGR and obtains samples that satisfy the relaxed NRS. Figure 7 depicts the results of the generated images from E-GBAS and the  $\epsilon_{L_2}$ -based sampling. We observed that the images generated by E-GBAS share more consistent attributes (e.g., composition of view and hair color) which is expected property of NRS. For example, in the first row of celebA results, we can identify the sampled images share the hair color and angle of face with different characteristics such as hair style. In LSUN dataset, the second row of results share the composition of buildings (right aligned) and the weather (cloudy).

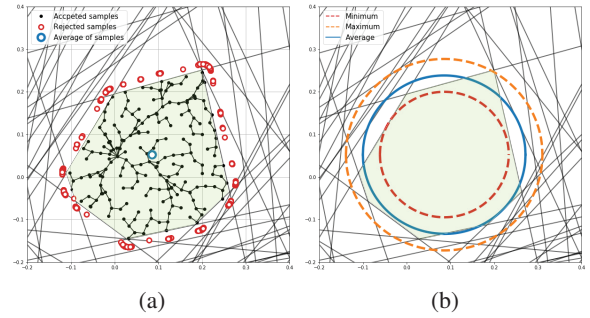


Figure 4: (a) The accepted samples (black dots), rejected samples (red dots) and average of the accepted samples (blue dot) by E-GBAS. (b) Visualizing the selection of  $\epsilon_{L_2}$  to make the area close to that of SSGR. The  $\epsilon_{L_2}$ -balls of each distance: min (red), max (orange) and average of min/max distance (blue).

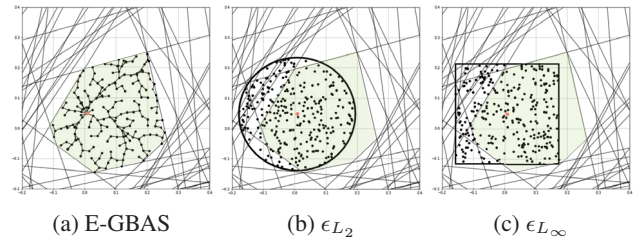


Figure 5: Geometric comparison of (a) E-GBAS, (b)  $\epsilon_{L_2}$  and (c)  $\epsilon_{L_\infty}$ -based sampling methods. Although the two  $\epsilon$ -balls cover some area of the GR, they cannot cover all of the GR and have a possibility to include infeasible area. Whereas, the E-GBAS includes the only feasible area of GR for sampling.

We try to analyze the generative mechanism of DGNNs along the depth of layer by changing the target layer. Figure 6 shows the examples and the standard deviations of the generated images by E-GBAS in each target layer. From the results, we discover that the variation of images is more localized as the target layer is set to be deeper. We argue that the GB in the lower layer attempts to maintain an abstract and generic information (e.g., angle of scene/entire shape of face), while those in the deeper layer tends to retain a concrete and localized information (e.g., edge of wall/mustache).

### 4.2 Quantitative Results

**The Similarity of Activation Values in Discriminator** A DGNN with the adversarial training has a discriminator to measure how realistic is the output created from a generator. During the training, the discriminator learns features to judge the quality of generated images. In this perspective, we expect that generated outputs from samples which satisfy NRS have similar feature values in the internal layers of the discriminator. We use cosine similarity between feature values of samples and the query. The relative evaluations of NRS for each sampling method are calculated by the average of similarities. When we denote a discriminator  $D(X) = d_{L:1}(X)$ , the query  $z_0$  and the obtained set of samples  $Z$ , the similarity

Layer #	DCGAN-MNIST				PGGAN-LSUN			PGGAN-celeBA		
	1	2	3	4	2	4	6	2	4	6
$\epsilon_{L_2}$ -based sampling	0.0819	0.0711	0.0718	0.0343	0.4951	0.4971	0.4735	0.5150	0.4994	0.4892
$\epsilon_{L_\infty}$ -based sampling	0.0834	0.0722	0.0720	0.0344	0.4641	0.4322	0.3365	0.4859	0.4799	0.3384
E-GBAS	<b>0.0781</b>	<b>0.0694</b>	<b>0.0675</b>	<b>0.0323</b>	<b>0.3116</b>	<b>0.2558</b>	<b>0.1748</b>	<b>0.2980</b>	<b>0.2789</b>	<b>0.1446</b>

Table 1: Standard deviations of generated images in each sampling methods. The number indicates the index of layer that GB constraint is applied in each DGNNs, where higher number is close to the output generation. The E-GBAS shows the lowest standard deviations compared to  $\epsilon$ -based sampling methods.

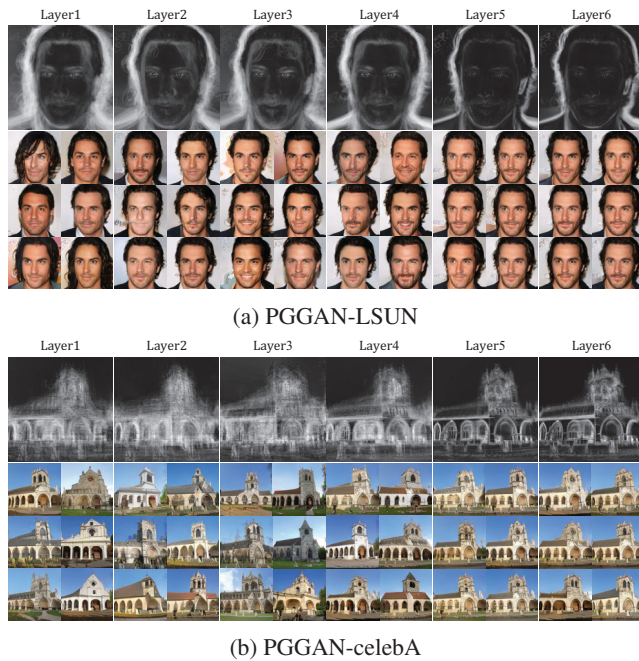


Figure 6: Examples of variations of generated images for each target layer. The first row shows the standard deviations of generated images for each target layer.

of feature values in the  $l$ -th layer is defined as the Equation (2). The operation  $d$  consists of linear transformations and a non-linear activation function.

$$S_{d_l} = E_{z \in Z} \left[ \frac{d_l(G(z))^T d_l(G(z_0))}{\|d_l(G(z))\| \|d_l(G(z_0))\|} \right], l \in \{1, 2, \dots, L\} \quad (2)$$

Table 2 shows the results of measuring the similarity for each internal layer in the discriminator.

**Variations of Generated Image** To quantify the consistency in attributes of the generated images, we calculate the standard deviation of generated images sampled by E-GBAS and variants of the  $\epsilon$ -based sampling. The standard deviation is calculated as Equation (3). The experimental results are shown in Table 1.

$$\sigma = \sqrt{E_{z \in GR_V} [G(z) - E_{z \in GR_V} [G(z)]]} \quad (3)$$

We randomly select 10 query samples and compute the average standard deviation of generated sets. Table 1 indicates

	Layer #	1	2	3	4
	MNIST	$\epsilon_{L_2}$ -based	0.722	0.819	0.864
$\epsilon_{L_\infty}$ -based		0.727	0.823	0.867	0.991
E-GBAS		<b>0.747</b>	<b>0.838</b>	<b>0.878</b>	<b>0.992</b>
LSUN	$\epsilon_{L_2}$ -based	0.578	0.602	0.957	0.920
	$\epsilon_{L_\infty}$ -based	0.551	0.613	0.960	0.946
	E-GBAS	<b>0.578</b>	<b>0.637</b>	<b>0.967</b>	<b>1.000</b>
celeBA	$\epsilon_{L_2}$ -based	0.678	0.718	0.785	0.963
	$\epsilon_{L_\infty}$ -based	0.684	0.720	0.789	0.965
	E-GBAS	<b>0.702</b>	<b>0.733</b>	<b>0.804</b>	<b>0.970</b>

Table 2: Comparisons of the average cosine similarity of feature values of the discriminator. The number indicates the index of layer in a discriminator.

that our E-GBAS has lower loss (i.e., consistent with the input query) compared to the  $\epsilon$ -based sampling in all three models and target layers.

## 5 Conclusion

In this study, we propose the explorative algorithm for analyzing the GR to identify generation mechanism in the DGNNs. Especially, we probe the internal layer in the trained DGNNs without additional training by introducing the GB of DGNNs. To gather samples which satisfy the NRS condition in the complicated and non-convex GR, we applied GB-RRT. We empirically show that the collected samples in the latent space with the NRS condition share the same generative properties. We also qualitatively indicate that the NRS in the distinct layers implies different generative attributes. Furthermore, the concept of the proposed algorithm is general and can also be used to probe the decision boundary in the classifier. So we believe that our method can be extended to different types of deep neural networks.

## Acknowledgement

This work was supported by the Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT), Korea (No. 2017-0-01779, XAI) and the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT), Korea (NRF-2017R1A1A1A05001456).





Figure 7: Generated samples from a query input (left),  $\epsilon$ -based sampling (middle) and E-GBAS sampling (right) of the three DGNNs (DCGAN-MNIST, PGGAN-LSUN and PGGAN-celebA.). We confirm that the generated images by E-GBAS have more consistent information compared to the  $\epsilon_{L_2}$ -based sampling.

## References

- Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 214–223.
- Bau, D.; Zhu, J.-Y.; Strobel, H.; Bolei, Z.; Tenenbaum, J. B.; Freeman, W. T.; and Torralba, A. 2019. Gan dissection: Visualizing and understanding generative adversarial networks. In *International Conference on Learning Representations*.
- Chang, C.-H.; Creager, E.; Goldenberg, A.; and Duvenaud, D. 2018. Explaining image classifiers by adaptive dropout and generative in-filling. *arXiv preprint arXiv:1807.08024*.
- Erhan, D.; Courville, A.; and Bengio, Y. 2010. Understanding representations learned in deep architectures. *Department d'Informatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep* 1355:1.
- Fawzi, A.; Moosavi-Dezfooli, S.-M.; Frossard, P.; and Soatto, S. 2018. Empirical study of the topology and geometry of deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Conference on Neural Information Processing Systems*, 2672–2680.
- Karras, T.; Aila, T.; and Laine. 2015. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.
- Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 4401–4410.
- Laugel, T.; Lesot, M.-J.; Marsala, C.; Renard, X.; and Detryniecki, M. 2019. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294*.
- LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Department, Iowa State University.
- LeCun, Y.; Cortes, C.; and Burges, C. 2010. Mnist handwritten digit database. *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2:18.
- Lei, N.; Luo, Z.; Yau, S.-T.; and Gu, D. X. 2018. Geometric understanding of deep learning. *arXiv preprint arXiv:1805.10451*.
- Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *International Conference on Computer Vision*.
- Montavon, G.; Lapuschkin, S.; Binder, A.; Samek, W.; and Müller, K.-R. 2017. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition* 65:211–222.
- Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. In *Conference on Neural Information Processing Systems*, 2924–2932.
- Morcos, A. S.; Barrett, D. G.; Rabinowitz, N. C.; and Botvinick, M. 2018. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*.
- Nguyen, A.; Dosovitskiy, A.; Yosinski, J.; Brox, T.; and Clune, J. 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Conference on Neural Information Processing Systems*, 3387–3395.
- Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Dickstein, J. S. 2017. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, 2847–2854.
- Ravaneli, M., and Bengio, Y. 2018. Interpretable convolutional filters with sinnet. *arXiv preprint arXiv:1811.09725*.
- Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.
- Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision*, 618–626.
- Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.
- Yu, F.; Seff, A.; Zhang, Y.; Song, S.; Funkhouser, T.; and Xiao, J. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.
- Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833.
- Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2016. Learning deep features for discriminative localization. In *IEEE conference on computer vision and pattern recognition*, 2921–2929.
- Zhu, J.-Y.; Krähenbühl, P.; Shechtman, E.; and Efros, A. A. 2016. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, 597–613.