STAG: Smart Tools and Applications in Graphics (2022)
D. Cabiddu, T. Schneider, G. Cherchi, and R. Scateni (Editors)

An Interactive Tuning Method for Generator Networks Trained by GAN

M. Zhou[1] and Y. Yamaguchi[1]

[1]The University of Tokyo, Japan

Abstract

The recent studies on GAN achieved impressive results in image synthesis. However, they are still not so perfect that output images may contain unnatural regions. We propose a tuning method for generator networks trained by GAN to improve their results by interactively removing unexpected objects and textures or changing the object colors. Our method could find and ablate those units in the generator networks that are highly related to the specific regions or their colors. Compared to the related studies, our proposed method can tune pre-trained generator networks without relying on any additional information like segmentation-based networks. We built the interactive system based on our method, capable of tuning the generator networks to make the resulting images as expected. The experiments show that our method could remove only unexpected objects and textures. It could change the selected area color as well. The method also gives us some hints to discuss the properties of generator networks which layers and units are associated with objects, textures, or colors.

CCS Concepts

• ***Computing methodologies*** → ***Image processing;*** *Neural networks;* • ***Human-centered computing*** → *Empirical studies in interaction design;*

1. Introduction

Generative Adversarial Networks (GANs) [GPM*14] have been able to synthesize data that are often indistinguishable from real-world data. They have been widely studied due to their outstanding data generation capability, and a large number of algorithms have been proposed. Some variants of GANs achieved impressive results for many real-world applications such as image synthesis and editing [WXH17], audio enhancement [TL19], music generation [YCY17], video super resolution [GK18], and so on.

Despite this tremendous success, it is still a difficult task to train GANs because of many crucial problems like mode collapse [GPM*14; AGL*17], vanishing gradient [GPM*14], and lack of proper evaluation metrics [ZIE*18; Bor19]. Therefore, many studies focus on changing the network architectures and training processes, such as mini-batch discrimination [SGZ*16], label smoothing [SVI*16], two time-scale update rule [HRU*17], noise to inputs [AB17], cycle-consistency loss [ZPIE17], adding self-attention layer [ZGMO19], progressive growing techniques [KALL18] and style-based generator [KLA19] to stabilize GANs training and improve GANs performance. Some of them also provide highly varied and large datasets, for instance, LSUN [YSZ*16], CelebA-HQ [KALL18], and Flickr-Faces-HQ (FFHQ) [KLA19].

These improved network architectures, training techniques, and high-quality datasets make the state-of-the-art GANs way better than before, but they are still not perfect. They sometimes produce images that are unnatural. For example, Fig. 1 shows some generated images of StyleGAN2 [KLA*20] trained on the largest LSUN dataset, and it still outputs images sometimes with visible artifacts or watermarks. It is of value tuning the generator networks to produce outputs without such unnatural regions. We would like to tackle this problem without changing the network architecture or retraining the network with higher quality datasets, which are both extremely expensive.

There are some studies about editing the generator networks of GANs [BZS*18; BSP*19]. It is possible to find out some groups of units that are closely related to objects by using a trained segmentation network. The outputs of generators could be changed in a meaningful way, such as adding or removing the objects through manipulating some units. But it requires a trained segmentation network as well as the generator network to figure out the precise relations between units and objects.

Although deep neural networks are hard to explain and usually considered as black-box models, there are some techniques [SCD*17; MLB*17; LL17; BZK*17] that could help us to find relations between output and network units. Since most generator networks of GANs are also deep neural networks, it gives us hints that we could use these techniques for locating the related units relevant to the specific region of generator output without additional information like a trained segmentation network. It would be possible to improve the generator outputs by manipulating those units.

DOI: 10.2312/stag.20221269

(a) (b) (c) (d)

Figure 1: *(a) Generated image of the bedroom by StyleGAN2 with visible artifacts; (c) Generated image of a cat by styleGAN2 with watermarks and captions. Using our tuning method, the proposed method can let the network output (b) and (d) without retraining the network or editing the images themselves.*

We propose a tuning method for generator networks to improve their results as expected. Without retraining the GANs, our method could find the units related to the specific objects or textures which allows us to remove them by ablating the corresponding units. Compared to previous studies, our proposed method focuses on tuning trained generator networks without retraining or relying on any additional information like a segmentation-based network. With our interactive system, it is easy to tune the generator networks for achieving the images that are as expected. Fig. 1(b) and Fig. 1(d) show the outputs after tuning the generator networks with our method.

The experiments show that our method could not only remove unexpected regions but also could change the selected area color. Our method could change unexpected regions while preserving other preferable regions as the user specified. Interestingly, the modification on a generator network for achieving an expected result with a latent vector may be effective for other results using different latent vectors. It would be possible to analyze the functionalities of units in a generator network which layers and units are related to certain objects, textures, or colors. The method is aimed at modifying the generator network which produces preferable results, rather than achieving a single better output image. Our method assumes that generator networks are convolutional neural networks, which means it can be applied to most variants of GANs. In this paper, we tested PGGAN and StyleGAN2, and our method succeeded in modifying both of the generator networks.

In Section 2, we first introduce some research about how to fine-tune the generator networks; then we note some studies about how to calculate the relevance between the network output and its units. Section 3 proposes our method of how to find those units that have the highest relevance with the specific objects. Section 4 explains the details of our models and the datasets for experiments and shows the experimental results of our method. We also discuss the properties of the tested generator networks, which layers are mainly associated with objects, textures, and colors. Finally, Section 5 concludes our research and the tasks for the future.

2. Related Work

2.1. Methods for Modifying Generator Networks

Many parts of GAN can be interpreted not only as signals related to the concept of objects but also as variables that are causally related to the composition of objects in the output. These interpretable effects can be used to compare, debug, modify, and infer GAN models. GAN inversion [XZY*21] is a technique to examine the relationship between a latent space and outputs, which is one of the most active research area in recent days. For instance, Endo proposed a method for editing layout of an image generated with StyleGAN [End22]. The method allows editing the layout of Sytle-GAN output images by manipulating the latent codes according to the user's inputs such as mouse dragging. It is based on latent space exploration by training a latent transformer with synthetic data and pseuod-user inputs. GAN inversion allows the user to explore the latent space and to obtain a preferable instance of image. But it is not aimed at modifying the network to produce natural results.

There have been some studies on editing the generator networks. GAN dissection [BZS*18] is an analysis framework used to visualize and understand GANs at different levels of abstraction from each neuron, to each object, to the contextual relationship between different objects. First, determining the set of interpretable units related to the object concept requires a pre-trained segmentation network. GAN dissection then participates in identifying the set of units that caused the specific object to disappear or appear. Finally, the contextual relationship between the object unit and its background will be obtained. This method is capable of determining an interpretable structure and provides a window into the internal mechanisms of GAN. Therefore, the knowledge of inserting the concept of an object into a specific location in a new image and interacting with other objects in the image is easy for humans to understand. However, it cannot be applied to usual generator networks because it completely depends on a segmentation network simultaneously trained with generator and discriminator networks.

The authors of GAN dissection have proposed a new scheme for editing an input image [BSP*19]. It first estimates a latent vector

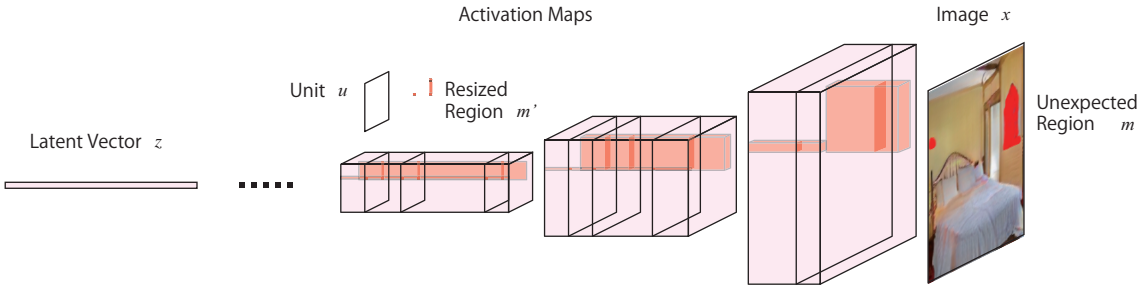

Figure 2: *Relevance values are calculated as the max/mean activation values within the user specified region. It exploits the location information only.*

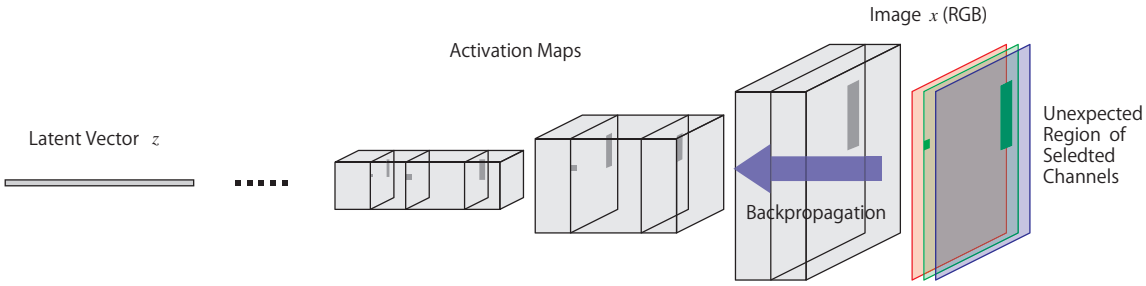

Figure 3: *Relevance values are obtained with backpropagation, e.g., DTD in our implementation. It takes account of output RGB channels as well as the location, which is crucial to handle color modification.*

that produces a similar result as the input image by analyzing the image. Then it allows the user to edit the output image of the generator network with the latent vector by applying GAN dissection. Finally, it completes the details of the output image by matching it with the original input image. Unfortunately, this method cannot be applied to usual generator networks, because it is also based on GAN dissection and requires the pre-trained segmentation network.

2.2. Network Explanation

Deep learning (DL) is a popular technology for processing large or complex data in recent years. DL's architecture is getting deeper and deeper, and machines can calculate and optimize millions of parameters, making it difficult for humans to track and recalculate [BMA19]. Especially for Deep Neural Networks (DNNs), the interpretability of the network is very poor comparing with the lately improved prediction accuracy. On the other hand, there exists an enormous need to understand the relationship between features and predictions to win the trust of users in real-world applications. Therefore, network explanation, namely, the techniques to explain the relationship between features and predictions, is an important issue in DNN. Some mechanisms must be provided for users to know traces of reasoning performed and to establish patient, trust, and fairness [Pre18].

The problem is that neural networks learn only from training data, but it is hard to prepare sufficient appropriate annotations on training data. Researchers have developed several so-called interpreters or explanators in an attempt to point out the attributions of the specific outputs. There have been proposed a lot of techniques in this area such as Deep Taylor Decomposition [MLB*17], Grad-CAM [SCD*17], Shapley values [Rot88], informative heatmaps [ZSBT18], grand tour [LZS20], saliency maps [SVZ14], and feature visualization [OMS17].

In this paper, we mainly used activation values that have been used in informative heatmaps and feature visualization. We also used Deep Taylor Decomposition to calculate the relevance between units and outputs. Our approach interactively explores the units of a generator network instead of the latent space which usually requires additional training.

3. Method

From the consideration of Section 2.2, we could suspect that it would be possible to improve generator networks by finding and ablating some units related to unexpected regions of an output image. The proposed approach allows the user to specify the unexpected region in a generated image; then, the method searches the most relevant units to the specified region. The user may obtain a modified image without the unexpected region by ablating the relevant units. The key issue is to design efficient algorithms to calculate the relevance of units from the user's input. In this paper, we propose two algorithms, respectively, based on activation values and Deep Taylor Decomposition.

To be clarified, for a generator network G, a latent vector z is firstly given. Once the latent vector z is given, a generated image x, as well as activation values of all units in G are fixed. The user

Figure 4: *Examples of removing objects. (a) Original outputs of StyleGAN2; (b) Specifying marks on unexpected regions; (c) Modified generator's outputs.*

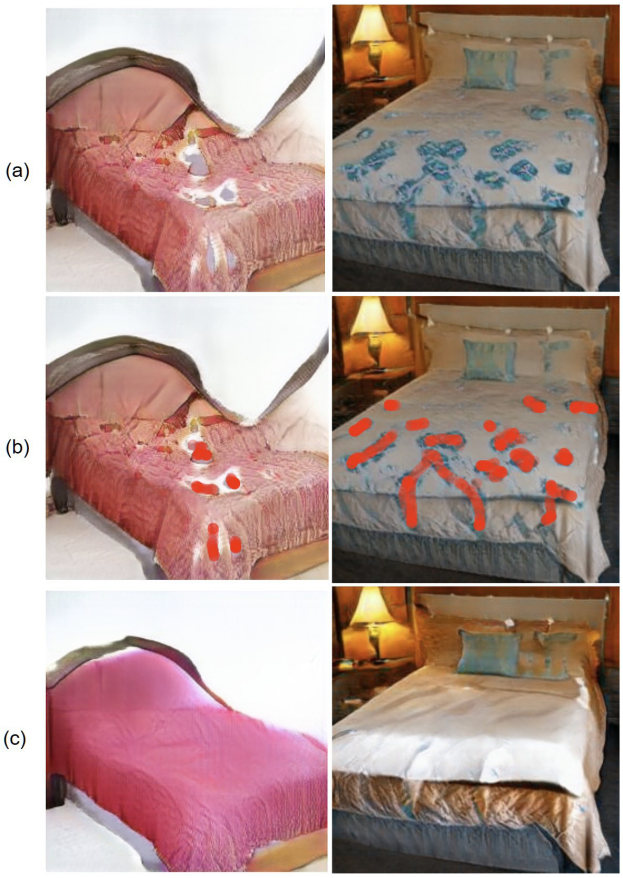

Figure 5: *Left: examples of removing unnatural stains from PGGAN output. (a) The original output; (b) Specifying marks on unexpected regions; (c) Modified generator's outputs. Right: examples of removing unpleasant patterns from StyleGAN2. (a) The original output; (b) Specifying marks on unexpected regions; (c) Modified generator's outputs.*

specifies the unexpected region m in x. Our algorithms take G and m as input, and outputs the relevance value of every unit u corresponding to m by analyzing the activation values in G. After that, our proposed method could sort the units by relevance values and the user ablates some of the highest ones. Then, a modified generator network G' generates a new image x' with the same latent vector z. If there are still unexpected regions in x', the user can repeat the above process until obtaining a satisfactory result.

3.1. Using Activation Values as Relevance

In a convolutional layer, a unit means a convolutional channel represented as a matrix spatially corresponding to the image. In the case of a classification DNN, its output is a label or probability of class that has no information about location in the image. Therefore, it is important to locate the regions in the relevant units closely related to the output. Informative heatmaps [ZSBT18] are aimed at illustrating the relevant regions.

On the other hand, our algorithms take the region information m as input. There is a simple and intuitive way to calculate the relevance values of units, which is to count activation values coincident with the specified region. For a given unit u, the proposed method first scales the unexpected region m to a matrix m', which has the same size as u. In matrix m', those elements that coincide with m will be set to one; others are set to zero. Then the Hadamard product $r = m' \circ u$ is calculated to obtain a relevance matrix r. As r is still a matrix, its max value r_{max} or mean value r_{mean} is regarded as the relevance of the unit. This process can be accelerated by using either adaptive max pool or adaptive mean pool. The procedure is summarized in Fig. 2.

3.2. Relevance by Deep Taylor Decomposition

Deep Taylor Decomposition (DTD) could be used to explain the generic DNNs. It is worth trying DTD on generator networks to calculate the relevance of units, which can distinguish channels of the output. The output image x has three channels if the output is an RGB image. It is possible to take account of RGB channel information to calculate relevance values by selecting channels as well as the region. For instance, if we want to change the red channel,

Figure 6: *Examples of generated images with other latent vectors with the same PGGAN generators as Fig. 5 left (a) and (c). (a) Images are generated with the original generator; (b) Images are generated with the modified generator.*

then we only assign ones to the unexpected region in the R channel and all zeros in the other two channels.

To use DTD on generator networks, the algorithm first takes an intersection between the region m and the selected channels of x. All the elements of the selected channels coincident with the selected region are set to one while others are set to zero. The DTD rules are applied to the activation maps as the backward computation. All the intermediate variables are saved as relevance values, while the backward computation is processed. For every unit, the relevance map is a matrix and, its max or mean value is used as the relevance value. Fig. 3 shows the process described above.

4. Experiments

4.1. Experiment Settings

We build a prototype system to try our method using PGGAN and StyleGAN2 that were trained on LSUN bedroom datasets. When feeling a generated image unsatisfied, the user first puts some marks on the unexpected regions of the image, and then the algorithms calculate the units' relevance values of the selected layers corresponding to the specified region. Here, the selected layers could be set to any layer or all layers. After sorting the units by their relevance values, a new network obtained by ablating some of the most relevant units generates a new image. If the new image is still unsatisfied, the user can select the different layers or relevance computing algorithm to get a new list of relevant units and can choose to ablate other units until the resulting image becomes as expected.

4.2. Removing Unexpected Regions

Removing Objects

Sometimes GANs may generate images with unnatural objects. For example, Fig. 4(a) shows some images generated by StyleGAN2 trained on the LSUN bedroom dataset that has strange small objects. As we described before, the user first puts marks on those unexpected regions as shown in Fig. 4(b), the user selects to ablate some units that our proposed method has judged most relevant to the unexpected regions, and modified networks generate new images as shown in Fig. 4(c).

Removing Stains and Textures

Sometimes the unexpected regions can be seen as textures or stains rather than objects. Especially for PGGAN, many generated images have those strange textures as shown in the left image of Fig. 5(a) and the three images of Fig. 6(a). The proposed method can remove the strange textures as shown in the left image of Fig. 5(c). When we tested StyleGAN2, we could also find some unpleasant textures, like the right image of Fig. 5(a). Our method succeeds in removing the patterns on the bedcovers. The right image of Fig. 5(c) is the result generated by the modified generator network. Unfortunately, the colors and wrinkles of bedcovers are also changed when the user specifies large areas to be removed as seen from the results. This is one of the limitations of our current system.

Once the user succeeds in removing unnatural textures from one image with a certain latent vector, similar textures in other images with different latent vectors are also removed. The examples are shown in Fig. 6 and Fig. 7. The images in Fig. 6 are generated with the original and modified generator of PGGAN with the same condition as Fig. 5 left. The images in Fig. 7 are generated with the original and modified generator of StyleGAN2 with the same condition as Fig. 5 right. These results suggest that the ablated units may have functions such as adding patterns to the bedsheets.

Removing Watermarks and Captions

The quality of the training datasets will certainly limit the model. Sometimes it is difficult to have clean and large datasets. For instance, the LSUN cat dataset is large enough. But the quality of images is hard to guarantee because they are collected from the Internet. Many images of the dataset have watermarks and captions, and our StyleGAN2 generator network trained on the LSUN cat dataset learned to generate the images with watermarks and captions.

Like the case of removing unnatural textures and stains, our

Figure 7: *Examples of generated images with other latent vectors with the same StyleGAN2 generators as Fig. 5 right (a) and (c). (a) Images are generated with the original generator; (b) Images are generated with the modified generator.*

Figure 8: *Examples of removing watermarks and captions. (a) Original outputs of StyleGAN2; (b) Final generator's outputs combined (d) and (f); (c) Specifying marks on watermarks; (d) Modified generator's outputs based on (c); (e) Specifying marks on captions; (f) Modified generator's outputs based on (e).*

method could easily solve this problem of the watermarks and captions by ablating the high-relevant units to the watermarks and captions of one image. Fig. 8 illustrates samples of removing watermarks and captions. Fig. 8(a) is an output image generated with the original generator network, which contains both a watermark and caption. The user specified the region of the watermark as in Fig. 8(c) and ablated two units in the network. The result of the modified network is Fig. 8(d), which contains no watermark any longer. The user also specified the region of the caption as in Fig. 8(e) and ablated a unit from the original network. The resulting image generated with the new network is Fig. 8(f), which has no more caption. After ablating all the units corresponding to the watermark and caption, the generator network outputs an image as shown in Fig. 8(b). Watermarks and captions also disappear in generated images with different latent vectors like the case of textures and stains. The results are illustrated in Fig. 9.

Interestingly, our method is able to detect the units related to some semantics such as textures, captions, and watermarks only from the low-level spatial information without using any segmentation-based network. The effect of multiple units ablation leads to the combination of the effects of each unit ablations. For instance, both watermarks and captions are eliminated from output images when all units corresponding to watermarks and captions are ablated.

4.3. Changing the Selected Area Colors

When using DTD, the user can select unexpected color channels as well as the spatial region in the image. It allows the user to control its colors. Our simple implementation is to mark ones at the specified area only in the selected channels and all zeros in the other channels. This implementation uses DTD to calculate the relevance between units and the specified area in the red, green, or blue chan-

Figure 9: *Examples of watermarks and captions with the same generator as in Fig. 8(a) and (b). (a) Original outputs of StyleGAN2 with watermarks and captions; (b) Modified generator's outputs.*

Figure 10: *Examples of changing selected area colors of PGGAN outputs. (a) Original output; (b) Adding marks and selecting the wanted channels (blue for the left one, green for the right one); (c) Modified generator's outputs.*

nel. Fig. 10 illustrates that our method can change the color of the specified area.

Now our method can remove objects and change colors. It is also possible to combine them in the tuning process. Fig. 11 shows how the user can tune a generator network to get a satisfactory result. Unfortunately the capability of changing colors is very limited, because the method can only ablate units relevant to user-selected channels but not increase or decrease effects of some channels. It is difficult to imagine the results by selecting channels and to control the amount of color change. We have to investigate the capability of changing colors much further.

4.4. Removing Unexpected Objects while Preserving Others

Sometimes, our method may find the units that not only affect the objects about which the user cares, but also other objects which the user wants to maintain. The examples are shown in Fig. 12(c). For solving this problem, our method first calculates the relevance r_1 of units corresponding to the unexpected objects (to be removed). Then it calculates the relevance r_2 of units corresponding to the expected objects (to be preserved). After that, the method uses $r = r_1 - \max(r_2, 0)$ as the new relevance value. The method can filter out the units related to the expected regions from those having high relevance to the unexpected regions. Fig. 12(e) shows that the new method can remove unexpected objects marked with red in Fig. 12(b) while preserving other objects marked with green in Fig. 12(d).

4.5. Discussion

Our implementation can immediately calculate the relevance values. The list of most relevant units, which is actually a set of toggle buttons for switching activation/deactivation of relevant units, can be updated in real time while the user is scribbling the unexpected regions. The image generation is also very quick so that the resulting output image is instantly generated after changing units selection with the toggle buttons. The modified image is displayed each time a button pressed. In practice, it usually takes less than 5 minutes to tune a generator network to achieve a satisfactory result.

Table 1: *Details of ablated units and the relevance computing algorithm. Relevance comp. means the relevance computation method, Lx : y means Unit y in Layer x, act. means activation values.*

Images	Ablated units	Relevance comp.	Generator
Fig. 4(c) left	L5:10,37,424; L6:476,486; L7:68,340,505	Max act.	StyleGAN2
Fig. 4(c) right	L3:3,222,248; L5:355; L7:409; L8:387	Max act.	StyleGAN2
Fig. 5(c) left / Fig. 6(b)	L2:191,349,436; L3:56,268,310; L4:281; L5:246,332; L7:104; L8:219	Max act. + Mean act.	PGGAN
Fig. 5(c) right / Fig. 7(b)	L5:94,375; L7:5,58,86,168,268,277,306,339,368,380, 465; L9:11,83,113,166,297,412; L11:159	Max act. + Mean act.	StyleGAN2
Fig. 8(b) / Fig. 9(b)	L2:390; L4:146; L5:88	Max act. + Mean act.	StyleGAN2
Fig. 8(d)	L4:146; L5:88	Max act. + Mean act.	StyleGAN2
Fig. 8(f)	L2:390	Max act. + Mean act.	StyleGAN2
Fig. 10 left	L11:23	DTD	PGGAN
Fig. 10 right	L12:9,55,2,30,49	DTD	PGGAN
Fig. 11(b)	L2:349	Max act.	PGGAN
Fig. 11(c)	L2:349,485; L4:281,222,462,363,256,348,146; L5:246,139,171,385,457,200	Max act. + Mean act.	PGGAN
Fig. 11(d)	L2:349,485; L4:281,222,462,363,256,348,146; L5:246,139,171,385,457,200; L6:336; L7:228; L12:55	Max act. + Mean act. + DTD	PGGAN
Fig. 12(c) left	L5:217,131,94,415,90,247,266,63	Max act.	StyleGAN2
Fig. 12(e) left	L5:439,333,131,90,63,183,113,224,247	Max act.	StyleGAN2
Fig. 12(c) right	L2:349,15	Max act.	PGGAN
Fig. 12(e) right	L3:451	Max act.	PGGAN

Figure 11: *Examples of removing objects and changing colors of a PGGAN output. (a) The original output; (b) Removing the stain on the left wall; (c) Removing the mirror on the right wall; (d) Changing the color of the bedcover.*

Table 1 shows the details of the ablated units and the used relevance computation algorithms together with the generator networks to obtain the images shown in Fig. 4 through Fig. 12. As seen from the column of relevance computation algorithm, relevance calculated with max activation values is usually effective if the region is small as an object in Fig. 4 and Fig. 12 right; Otherwise, like a case of textures and stains as in Fig. 5, relevance calculated with mean activation values is also useful to tune generator networks.

Units of the middle layers of networks are ablated to remove objects or textures. Sometimes, units from relatively early layers are also ablated in the case of PGGAN, as seen in Fig. 5 left, Fig. 11, and Fig. 12 right. In order to change the color of output images, DTD must be applied, and the very late layers, i.e., the very close layers to the output, should be ablated as seen in Fig. 10. From these empirical results, we can guess the properties of layers in the generator networks.

5. Conclusions

In conclusion, this paper proposed an interactive method that primarily changes a generator network to create an image with a specific latent vector as expected. The experiments show us that our method can remove unexpected objects, textures, and watermarks. Our improved method can remove unexpected objects while preserving the others. It can also change the color of the specific areas. However, the most important point is that the modified generator network outputs other images (with different latent vectors) without such patterns or watermarks, after the user removes patterns or watermarks from one image. It suggests that our method can find a group of units that causes understandable parts or features in output images only from the low-level spatial information without using any additional information like segmentation-based networks. In other word, our method succeeds to tune the generator network to produce output images without unnatural regions based only on the users' interaction. It would be one of the most useful applications to remove watermarks from output images. The generator

network must learn watermarks when the training dataset contains watermarks. Lots of efforts should have been made to provide good datasets without watermarks previously. The proposed method allows to train GAN using datasets with watermarks. Although the trained network learns watermarks, our method can change the network to produce output images without watermarks.

The proposed method has some limitations. Due to the nature of tuning, namely, ablation of selected units, it is merely capable of removing the unexpected region or changing the color to some extent, but not modifying image layout or adding some objects into the image. Especially the ability of changing color is very limited in our current implementation. Our method can find the units that are most relevant to the specific region and we have succeeded to remove anything from images in our experiments. But it is not guaranteed that a preferable result can be obtained by ablating the top (most relevant) unit. An acceptable result may be achieved by ablating a few top units when a PGGAN generator network is used and the unexpected region is small as in Fig. 11(b) and Fig. 12 right. However, several units must be removed to achieve satisfactory results if a StyleGAN2 generator network is used or the unexpected regions are rather large. The user must carefully select units from the top related units (not simply the top unit). We also found some side effects so that other region might be incidentally changed by the unit ablation. The ablation of units for a large area may affect other features like colors and wrinkles as in Fig. 5(c), which is also a limitation of our method. The generality of the network tuning is an open question. The similar kind of unexpected regions may be mostly removed from outputs generated with other latent vectors. But it is not guaranteed if the modified network can produce satisfactory outputs with any other latent vectors. It is also worth trying more models on different datasets for future work. We may find more facts about GANs by testing more models and datasets.

References

[AB17] ARJOVSKY, MARTIN and BOTTOU, LEON. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. arXiv: 1701.04862 [stat.ML] 1.

[AGL*17] ARORA, SANJEEV, GE, RONG, LIANG, YINGYU, et al. "Generalization and Equilibrium in Generative Adversarial Nets (GANs)". *Proceedings of the 34th International Conference on Machine Learning*. Ed. by PRECUP, DOINA and TEH, YEE WHYE. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, 224–232 1.

[BMA19] BUHRMESTER, VANESSA, MÜNCH, DAVID, and ARENS, MICHAEL. *Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey*. 2019. arXiv: 1911.12116 [cs.AI] 3.

[Bor19] BORJI, ALI. "Pros and cons of GAN evaluation measures". *Computer Vision and Image Understanding* 179 (2019), 41–65. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2018.10.009 1.

[BSP*19] BAU, DAVID, STROBELT, HENDRIK, PEEBLES, WILLIAM, et al. "Semantic Photo Manipulation with a Generative Image Prior". *ACM Trans. Graph.* 38.4 (July 2019), 1–11. ISSN: 0730-0301. DOI: 10.1145/3306346.3323023 1, 2.

[BZK*17] BAU, DAVID, ZHOU, BOLEI, KHOSLA, ADITYA, et al. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 3319–3327. DOI: 10.1109/CVPR.2017.354 1.

[BZS*18] BAU, DAVID, ZHU, JUN-YAN, STROBELT, HENDRIK, et al. *GAN Dissection: Visualizing and Understanding Generative Adversarial Networks*. 2018. arXiv: 1811.10597 [cs.CV] 1, 2.

[End22] ENDO, YUKI. "User-Controllable Latent Transformer for StyleGAN Image Layout Editing". *Computer Graphics Forum* 41.7 (2022). DOI: 10.1111/cgf.14686 2.

[GK18] GOPAN, KARTHIKA and KUMAR, G.S. "Video Super Resolution with Generative Adversarial Network". *2nd International Conference on Trends in Electronics and Informatics (ICOEI)*. 2018, 1489–1493. DOI: 10.1109/ICOEI.2018.8553719 1.

[GPM*14] GOODFELLOW, IAN J., POUGET-ABADIE, JEAN, MIRZA, MEHDI, et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML] 1.

[HRU*17] HEUSEL, MARTIN, RAMSAUER, HUBERT, UNTERTHINER, THOMAS, et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017, 6629–6640. ISBN: 9781510860964 1.

[KALL18] KARRAS, TERO, AILA, TIMO, LAINE, SAMULI, and LEHTINEN, JAAKKO. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. arXiv: 1710.10196 [cs.NE] 1.

[KLA*20] KARRAS, TERO, LAINE, SAMULI, AITTALA, MIIKA, et al. "Analyzing and Improving the Image Quality of StyleGAN". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, 8107–8116. DOI: 10.1109/CVPR42600.2020.00813 1.

[KLA19] KARRAS, TERO, LAINE, SAMULI, and AILA, TIMO. "A Style-Based Generator Architecture for Generative Adversarial Networks". *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, 4396–4405. DOI: 10.1109/CVPR.2019.00453 1.

[LL17] LUNDBERG, SCOTT M. and LEE, SU-IN. "A Unified Approach to Interpreting Model Predictions". *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017, 4768–4777. ISBN: 9781510860964 1.

[LZS20] LI, MINGWEI, ZHAO, ZHENGE, and SCHEIDEGGER, CARLOS. "Visualizing Neural Networks with the Grand Tour". *Distill* (2020). https://distill.pub/2020/grand-tour. DOI: 10.23915/distill.00025 3.

[MLB*17] MONTAVON, GRÉGOIRE, LAPUSCHKIN, SEBASTIAN, BINDER, ALEXANDER, et al. "Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition". *Pattern Recogn.* 65.C (May 2017), 211–222. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2016.11.008 1, 3.

[OMS17] OLAH, CHRIS, MORDVINTSEV, ALEXANDER, and SCHUBERT, LUDWIG. "Feature Visualization". *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007 3.

[Pre18] PREECE, ALUN. "Asking 'Why' in AI: Explainability of intelligent systems – perspectives and challenges". *Intelligent Systems in Accounting, Finance and Management* 25.2 (2018), 63–72. DOI: 10.1002/isaf.1422 3.

[Rot88] ROTH, A.E. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988. DOI: 10.1017/CBO9780511528446 3.

[SCD*17] SELVARAJU, RAMPRASAATH R., COGSWELL, MICHAEL, DAS, ABHISHEK, et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, 618–626. DOI: 10.1109/ICCV.2017.74 1, 3.

[SGZ*16] SALIMANS, TIM, GOODFELLOW, IAN, ZAREMBA, WOJCIECH, et al. "Improved Techniques for Training GANs". *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2016, 2234–2242. ISBN: 9781510838819 1.

[SVI*16] SZEGEDY, C., VANHOUCKE, V., IOFFE, S., et al. "Rethinking the Inception Architecture for Computer Vision". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2016, 2818–2826. DOI: 10.1109/CVPR.2016.308 1.

[SVZ14] SIMONYAN, KAREN, VEDALDI, ANDREA, and ZISSERMAN, ANDREW. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034 [cs.CV] 3.

[TL19] TORRES-REYES, NORBERTO and LATIFI, SHAHRAM. "Audio Enhancement and Synthesis using Generative Adversarial Networks: A Survey". *International Journal of Computer Applications* 182.35 (Jan. 2019), 27–31. ISSN: 0975-8887. DOI: 10.5120/ijca2019918334 1.

[WXH17] WU, XIAN, XU, KUN, and HALL, PETER. "A survey of image synthesis and editing with generative adversarial networks". *Tsinghua Science and Technology* 22.6 (2017), 660–674. DOI: 10.23919/TST.2017.8195348 1.

[XZY*21] XIA, WEIHAO, ZHANG, YULUN, YANG, YUJIU, et al. "GAN Inversion: A Survey". *CoRR* abs/2101.05278 (2021). arXiv: 2101.05278. URL: https://arxiv.org/abs/2101.05278 2.

[YCY17] YANG, LI-CHIA, CHOU, SZU-YU, and YANG, YI-HSUAN. *MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation*. 2017. arXiv: 1703.10847 [cs.SD] 1.

[YSZ*16] YU, FISHER, SEFF, ARI, ZHANG, YINDA, et al. *LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop*. 2016. arXiv: 1506.03365 [cs.CV] 1.

[ZGMO19] ZHANG, HAN, GOODFELLOW, IAN, METAXAS, DIMITRIS, and ODENA, AUGUSTUS. "Self-Attention Generative Adversarial Networks". *Proceedings of the 36th International Conference on Machine Learning*. Ed. by CHAUDHURI, KAMALIKA and SALAKHUTDINOV, RUSLAN. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, 7354–7363 1.

[ZIE*18] ZHANG, RICHARD, ISOLA, PHILLIP, EFROS, ALEXEI A., et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, 586–595. DOI: 10.1109/CVPR.2018.00068 1.

[ZPIE17] ZHU, JUN-YAN, PARK, TAESUNG, ISOLA, PHILLIP, and EFROS, ALEXEI A. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". *IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, 2242–2251. DOI: 10.1109/ICCV.2017.244 1.

[ZSBT18] ZHOU, BOLEI, SUN, YIYOU, BAU, DAVID, and TORRALBA, ANTONIO. "Interpretable Basis Decomposition for Visual Explanation". *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2018, 122–138. ISBN: 978-3-030-01237-3. DOI: 10.1007/978-3-030-01237-3_8 3, 4.

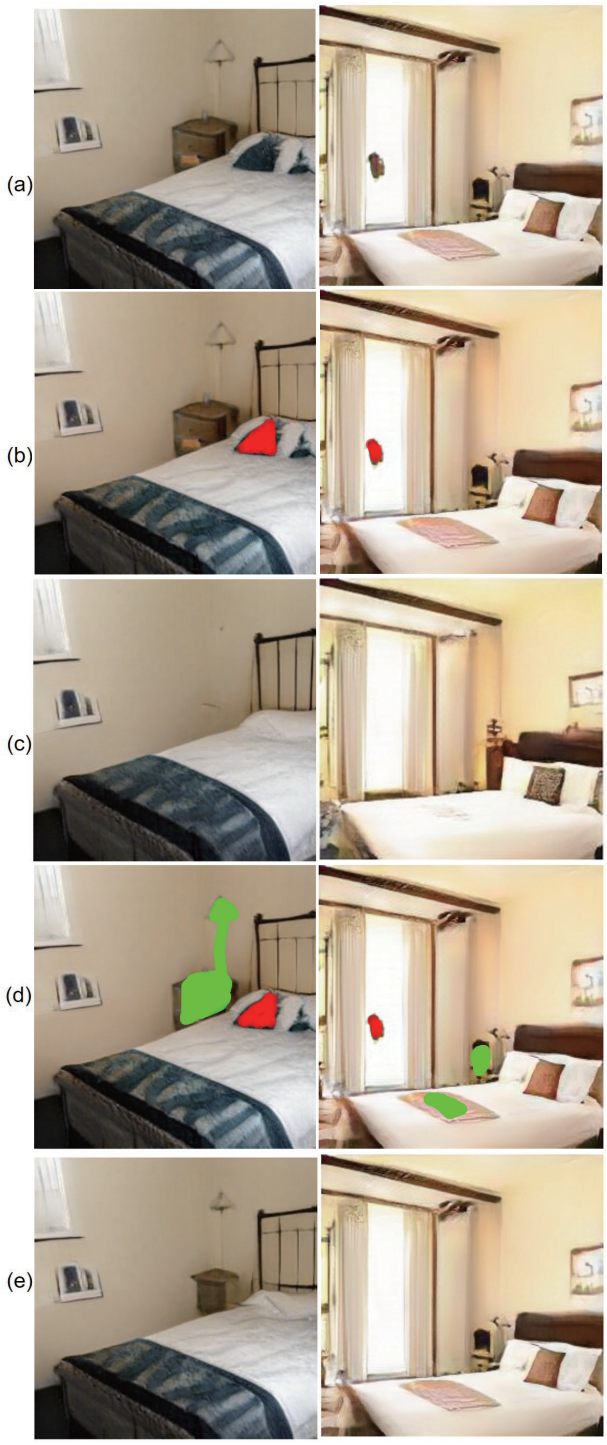

Figure 12: *Examples of removing unexpected objects while affecting other objects (Left: StyleGAN2; Right: PGGAN). (a) Original outputs; (b) Specifying marks on unexpected regions in red; (c) The modified generator's outputs that are lacking other objects as well (a bedside table and a torch in the left image, a bedside table with an object and a bed throw in the right image); (d) Specifying marks on regions to be preserved in green; (e) The final outputs.*