Predicting Gaussian noise injection for self-supervised Generative Adversarial Nets

Zhiyuan Wu[1], **Grigorios G Chrysos**[2], **Volkan Cevher**[2]
[1] Technical University of Munich, Germany
[2] EPFL, Switzerland

Abstract

Generative Adversarial Nets (GANs) have been a strong performing generative model, despite their reported training instability. Auxiliary tasks, e.g., through self-supervision, have been used to encourage the discriminator of the GANs to learn meaningful feature representations. Specifically, rotating an image and then predicting the rotation has been used as such a self-supervised task. However, the rotation prediction depends heavily on the dataset. In this work, we propose a new self-supervised pretext task that injects Gaussian noise in a block of the discriminator and then predicts in which block the noise was injected. We verify experimentally the proposed pretext task and demonstrate that it leads to improvements in the quality of the synthesized images in three datasets. In addition, we assess the representation learning performance of the pretext task and find that it performs on par with rotation-based pretext task. Interestingly, the representations learned with the proposed pretext task lead to a more balanced classification accuracy across classes, namely there are larger improvements for weaker classes.

1 Introduction

Since their advent, Generative Adversarial Nets (GANs) [6] have demonstrated strong performance in synthesizing photo-realistic images [13, 3]. The training of GANs includes a minmax game with two players, i.e., a generator network and a discriminator network. The generator maps the low dimensional input to a high-dimensional sample that resembles samples from the target distribution. At the same time, the discriminator learns to distinguish the synthesized samples from target distribution samples. However, the training of GANs, especially the performance control of the discriminator, is usually difficult and unstable [13, 17]. The spectral normalization [13] of the discriminator weights has proved a significant component in reducing the instability of training GANs. Despite such regularization schemes, the instability is not completely mitigated yet.

Self-supervised learning has been used to augment the adversarial minmax game in GANs [9] to stabilize the training [12, 15]. Similar to DiffAugment proposed in [19], the role of self-supervision is to regularize the discriminator and enhance the performance of GANs [15]. Chen et al. [2] are motivated by [1, 5] to add a pretext task classifier that shares weights with the discriminator. The pretext task rotates the images and then the classifier (an extra network sharing the core part of computation with the discriminator) predicts the rotation angle. In other words, simultaneously to the minmax game, the self-supervised classifier extracts continuously improved representations [2], thus leading to a discriminator with meaningful representations.

However, the performance of rotation-based pretext task depends heavily on the dataset. When the appearance of an image remains (approximately) invariant to (some) rotation angles, e.g. like in Figure 2, then a rotation-based pretext task, might not be optimal. Furthermore, we know from [16] that, after applying instance noise to samples, the discriminator is less likely to overfit because the instance noise can make the training distribution broader. Combining these facts, we propose a new pretext task, which applies Gaussian noise randomly in one of the blocks of the discriminator with the task to predict in which block the noise was inserted. Simultaneously, based on the developments on the representation learning, we find that replacing the convolutional discriminator with a polynomial neural network provides an improvement in both the quantitative and qualitative metrics, and we thus customize the pretext task for such a discriminator. The auxiliary loss (i.e., pretext task) predicts the identity of the polynomial that the noise was inserted. We assess the performance of the designed pretext task using both standard GANs metrics, and consider the representation learning in isolation. That is, after the training, we utilize the discriminator for a classification task. Our evaluation on three datasets demonstrates the efficacy of the proposed task and reveals that the proposed pretext task learns representations that are stronger for the weakest classes.

2 Self-Supervised Poly-GAN with Gaussian Noise

The preliminary experiments demonstrate that replacing the convolutional discriminator with a polynomial neural network from the recently proposed Π-net family [3] improves the performance of the GANs. We will also compare the performance in the experimental section, but we assume below that the discriminator is a polynomial neural network.

Different from augmenting the input samples of the discriminator [2, 19], the idea of Self-Supervised Poly-GAN (P-SS-GAN) with Gaussian Noise is to apply Gaussian noise randomly in one of the blocks of the discriminator to act on the intermediate representation. We compare the output of the discriminator with the pseudo labels [10] to see if the position of the inserted Gaussian noise can be correctly predicted. Figure 1 shows how the Gaussian noise-based pretext works in GANs.

Suppose there are N blocks in the discriminator with the output of the t^{th} block being $z_{t+1} = z_t + Cz_t + (Cz_t) * z_t$. The operator C represents a series of convolution operations, z_t is the output from the $(t-1)^{\text{th}}$ block and $(Cz_t) * z_t$ is the second order polynomial term, where $*$ is the Hadamard product. For each sample we choose one block out of N with a uniform probability and multiply the second order term by Gaussian noise as illustrated in Figure 3. In other words, if t is the block we introduce Gaussian noise, the output of this block will be: $z_{t+1} = z_t + Cz_t + (Cz_t) * z_t * \rho$. Here $\rho \sim \mathcal{N}\left(\mu, \delta^2\right)$ is samples from a Gaussian distribution parameterized by μ and σ^2 and has the same size as z_t. If t is not the block we apply Gaussian noise, the output of this block remains the same, i.e. $z_{t+1} = z_t + Cz_t + (Cz_t) * z_t$. More formally, we introduce an N-dimensional variable v as follows:

$$z_{t+1} = z_t + Cz_t + (Cz_t) * z_t * \mathcal{M} = z_t + Cz_t + (Cz_t) * z_t * \left[\rho * v[t] + (1 - v[t])\right], \quad (1)$$

where $\mathcal{M} = \rho * v[t] + (1 - v[t])$, with $\sum_{i=1}^{N} v[i] = 1$ and $v[i] \subseteq \{0, 1\}$, $l \subseteq L = \{1, 2 \ldots \mathcal{N}\}$ and l is randomly chosen with the same probability in each time. Given l, we have $v[l] = 1, v[i \neq l] = 0$.

A classifier Q, which shares all the weights except for the last layer with the discriminator D, evaluates the performance on the pretext task. Similar to the rotation-based pretext task, if $S = \{0, 1\}$ denotes fake or real images and $L = \{0, 1..., N\}$ denotes the position of Gaussian noise, the value function $V(G, D)$ as mentioned in [2] and loss function can be written as:

$$V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim P_{\text{data}}(\boldsymbol{x})}\left[\log P_D(S = 1 \mid \boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x} \sim P_G(\boldsymbol{z})}\left[\log\left(1 - P_D(S = 0 \mid \boldsymbol{x})\right)\right] \quad (2)$$

$$L_G = -V(G, D) - \alpha E_{x \sim P_G} E_{l \sim L}\left[\log Q_D(L = l \mid x, l)\right] \quad (3)$$

$$L_D = V(G, D) - \beta E_{x \sim P_{data}} E_{l \sim L}\left[\log Q_D(L = l \mid x, l)\right] \quad (4)$$

3 Experiments

Datasets Three widely used datasets are used for the experimental validation of the proposed pretext task. The CIFAR10 dataset [11] consists of 60,000 $32 \times 32 \times 3$ color images in 10 classes. The STL10 dataset [4] includes 100,000 images with resolution $96 \times 96 \times 3$, while STL10 includes 10 classes. The Columbia Object Image Library (COIL20) dataset [14] contains 20 objects, each of which has 72 images captured every 5 degrees along a viewing circle. We use CIFAR10 as a standard dataset in the task. STL10 is used since it contains images of higher-resolution but with

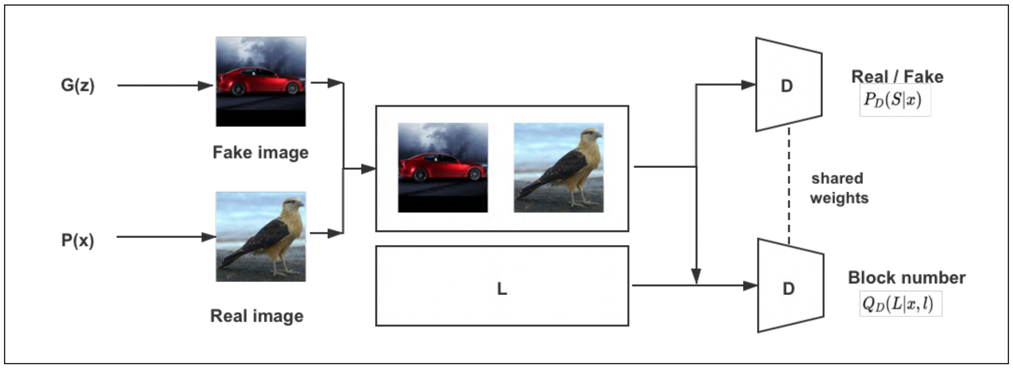

Figure 1: Diagram of our Self-Supervised Poly-GAN (P-SS-SNGAN(N)). $G(z)$ generates fake images that look like images in the real world. z is the latent variable. $P(x)$ is the real distribution. The generator and the discriminator play a two-player game to determine the image real or fake, and they collaborate to predict the position of Gaussian noise. L is a block to generate the random vector l. During the training, the images and the vector l are sent to the discriminator together.

similar semantic meaning to CIFAR10. COIL20 is used, since it contains more classes but fewer samples. For the STL10 dataset and the COIL20 dataset, images are firstly resized to 64×64, so the generator adds a block while we don't modify the discriminator.

Models For our experiment, we use Polynomial SNGAN(P-SNGAN) where we replace the convolutional discriminator of SNGAN with the Polynomial neural nets [3] and observe that they improve representation power. Figure 3 shows the basic block of P-SNGAN's discriminator. For Gaussian noise-based pretext task, we copy the images from in a batch and apply noise to the second half of the joined batch of images. First half of images(without noise) are used for true/fake classification task.

3.1 Results

Sample Quality For our quantitative evaluation we use the standard metric of Frechet Inception Distance (FID) [8] that performs up to second order matching of two distributions. Table 1 reports the FID scores on the three datasets. On the CIFAR10, within 50k iterations, SNGAN obtains the highest FID score. When the discriminator of SNGAN is replaced with the polynomial network, the FID score of P-SNGAN has been significantly improved by 4.418 compared to SNGAN. Both the rotation-based pretext task (P-SS-SNGAN(R)) and our pretext task (P-SS-SNGAN(N)) produce more extra improvement, get an FID score below 18. Indicative synthesized images from SNGAN, P-SNGAN, P-SS-SNGAN (R), and P-SS-SNGAN (N) are depicted in Fig. 4, 5, 6, and 7 respectively in the appendix. On the STL10 and COIL20, we add a block in the generator, and run 30k iterations. The performance of the proposed method is on par to the rotation-based pretext task.

Representation Quality Beyond the evaluation of the sample quality, we also assess the representation learning of the discriminator [18]. We replace the dense layers of the discriminator with a single dense layer trained for classification on the particular dataset, while we maintain the rest of the weights frozen. The accuracy of the classifier is considered as the evaluation metric indicating the extraction of high-quality representations. Table 2 reports the accuracy across different datasets. On CIFAR10, our method can exceed the baseline by 5.53%. As a supplement, we also delete the pretext loss term from P-SNGAN(N) to validate the effect of the regularization via Gaussian noise only. The accuracy is 61.88%. Our method also presents a similar performance to the rotation-based method on STL10 and COIL20. Moreover, Table 3 shows the accuracy of each class of images. On CIFAR10, the correctness on classes 3 and 6 exceeds the rotation-based pretext task by 5.8% and 8.2%, respectively. Interestingly, our method can achieve accuracy on nine out of ten categories exceed 50%, and the lowest accuracy is 49.4%. On STL10, our method's accuracy on classes 4 and 6 exceeds the rotation-based pretext task by 14.2% and 9.8%. What is more, The variance of the accuracy on ten categories has been reduced by 40%. The result on COIL20 is similar: The variance of the accuracy across categories is reduced by 34%.

Table 1: FID score. On CIFAR10, we run all the settings: baseline SNGAN, SNGAN with a polynomial discriminator(P-SNGAN), SNGAN with rotation-based pretext task(SNGAN(R)), and P-SNGAN with two different pretext tasks. The polynomial discriminator can alone contribute a decrease in the FID score of 4.418, i.e., from 23.680 to 19.292, while the SNGAN(R) decreases the FID score from 23.680 to 21.668 . With a polynomial discriminator, two pretext tasks can achieve lower FID scores: 17.166 and 17.886. As for STL10 and COIL20, we observe similar result.

Method	CIFAR10	STL10	COIL20
SNGAN	23.680	62.381	94.491
P-SNGAN	19.262	61.667	92.421
SNGAN(R)	21.668	61.142	92.257
P-SS-SNGAN (R)	**17.166**	**59.664**	**91.321**
P-SS-SNGAN (N)	17.886	60.163	91.782

Table 2: Accuracy. On CIFAR10, the polynomial discriminator can alone contribute an increase in the accuracy of 1.48%, i.e., from 58.28% to 59.76%, while the SNGAN(R) increases the accuracy from 58.28% to 60.17%. With a polynomial discriminator, two pretext tasks can achieve better performance: 64.17% and 63.81%. As for STL10 and COIL20, the accuracy of two pretext tasks is quite close.

Method	CIFAR10	STL10	COIL20
SNGAN	58.28%	56.84%	90.00%
P-SNGAN	59.76%	57.64%	92.67%
SNGAN(R)	60.17%	57.29%	94.00%
P-SS-SNGAN (R)	**64.17**%	**58.12**%	**96.67**%
P-SS-SNGAN (N)	63.81%	57.85%	95.67%

The detailed results in Tables [7, 8, 9, 10, 11, 12] indicate that the proposed pretext task results in a more balanced classification accuracy across different classes. That means that classes with weaker accuracy are learn richer representations.

4 Conclusion

We proposed a new pretext task for self-supervised GAN. Similar to present pretext tasks, our Gaussian noise-based pretext task can significantly improve the quality of the synthesized images and the capacity of the discriminator to learn representations. Experiments on different datasets verify that the Gaussian noise-based pretext task leads to a more balanced classification accuracy among different categories. Our pretext task exhibits a stable performance on different images and is especially suitable for images that are (approximately) rotation-invariant, such as flowers and medical CT images.

Table 3: Classification accuracy per class. We rerun P-SS-SNGAN(R) five times and report the classes with the lowest accuracy. Then we compare them with our method. On CIFAR10, the accuracy of class 3 (bird) and class 6 (dog) by P-SS-SNGAN(R) is significantly lower than other classes, while P-SS-SNGAN(N) improves the accuracy of class 3 and 6 significantly. On STL10, we observe similar result in class 4 (cat) and class 6 (dog). Also, on STL10 and COIL20, our method can decrease the variance of accuracy on classes. For detailed results, please find them in the appendix.

Method	CIFAR10			STL10			COIL20	
	3	6	σ^2	4	6	σ^2	2	σ^2
P-SS-SNGAN(R)	0.446	0.422	0.011	0.348	0.184	0.027	0.333	0.029
P-SS-SNGAN(N)	**0.504**	**0.504**	0.011	**0.490**	**0.282**	**0.015**	**0.920**	**0.019**

Acknowledgments and Disclosure of Funding

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-19-1-0404. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 725594 - time-data).

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. cite arxiv:1206.5538.

[2] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12146–12155, 2019.

[3] Grigorios G. Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos P Zafeiriou. Deep polynomial neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[5] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[9] Rui Huang, Wenju Xu, Teng-Yok Lee, Anoop Cherian, Ye Wang, and Tim K. Marks. Fx-gan: Self-supervised gan learning via feature exchange. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3183–3191, 2020.

[10] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:4037–4058, 2021.

[11] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[12] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In *International Conference on Machine Learning (ICML)*, 2019.

[13] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[14] Nayar and H. Murase. Columbia object image library: Coil-100. Technical Report CUCS-006-96, Department of Computer Science, Columbia University, February 1996.

[15] Parth Patel, Nupur Kumari, Mayank Kumar Singh, and Balaji Krishnamurthy. Lt-gan: Self-supervised gan with latent transformation detection. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3188–3197, 2021.

[16] C. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations*, 2017.

[17] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Linxiao Yang, and Ngai-Man Cheung. Self-supervised gan: Analysis and improvement with multi-class minimax game. In *NeurIPS*, 2019.

[18] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *ECCV*, 2016.

[19] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Table 4: Generator for the CIFAR10. We use similar architectures to the ones used in [13, 7]. As for STL10 and COIL20, there's an additional ResBlock.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense, $4 \times 4 \times 256$
ResBlock upsample
ResBlock upsample
ResBlock upsample
BN, ReLU, 3×3 conv, 3 Tanh

A Appendix

On the CIFAR10 dataset, we run the baseline SNGAN with the ResNet architecture [7, 13], SNGAN with auxiliary rotation loss, P-SNGAN, P-SNGAN with auxiliary rotation loss, and P-SNGAN with auxiliary Gaussian-noise loss. We used Adam for the optimization and used the hyper parameter used in [7]: The initial learning rate of D and G we set 0.0002, beta1 is 0.0, and beta2: 0.9. In each iteration, we run five times discriminator and one-time generator. The loss type is the Hinge loss, and the loss type for pretext task is cross entropy loss. We train the GAN for 50k iterations. During the training of P-SS-SNGAN (rotation), the hyper-parameter α and β we set is 1 and 1 according to [2]. During the training of P-SS-SNGAN (noise), α and β we set is 1 and 0.2. The mean μ and variance σ^2 of Gaussian noise we set is 0.8 and 0.25, which can make the pretext task challenge the discriminator enough. On the COIL20 dataset, we manually divide part of images into the training set and others into the test set for each class. As for Table 3, compared to Table 2, we run fewer but the same epochs five times for each dataset.

In our experiment, we calculate FID using 50,000 synthesised images and save 100 synthesised images for each 1,000 iterations. Experiments are mainly run on Colab and Google Cloud Plattform. The GPU we use is Tesla P100, Tesla V100 and Tesla A100.

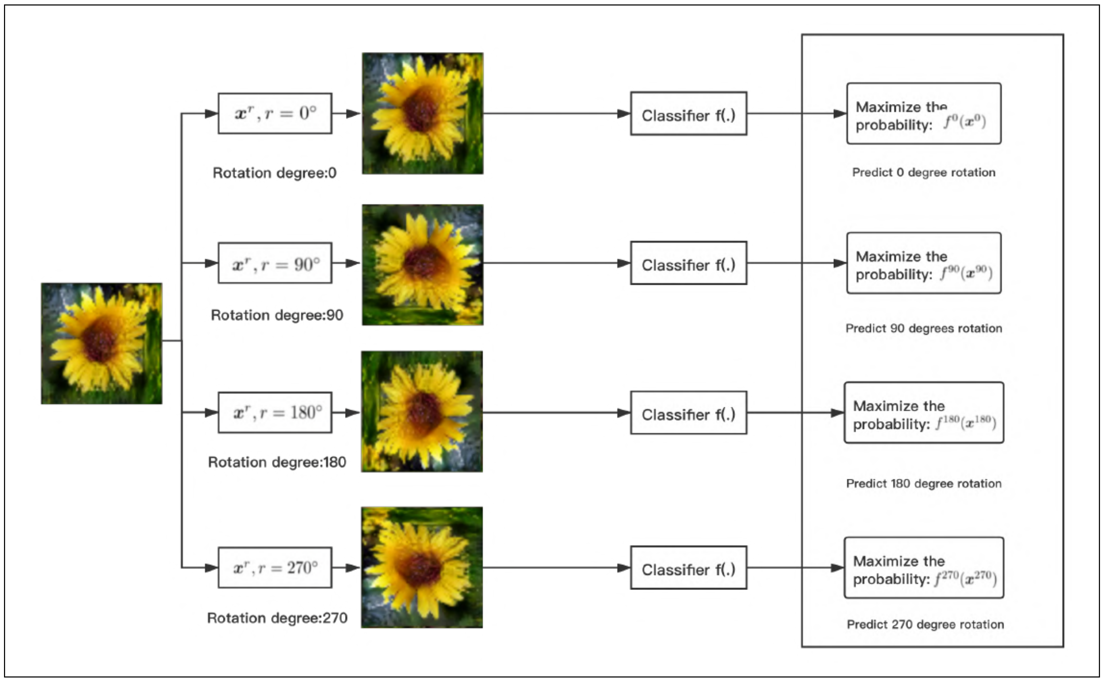
$x^r, r = 0°$
Rotation degree:0
$x^r, r = 90°$
Rotation degree:90
$x^r, r = 180°$
Rotation degree:180
$x^r, r = 270°$
Rotation degree:270
Classifier f(.)
Classifier f(.)
Classifier f(.)
Classifier f(.)
Maximize the probability: $f^0(x^0)$
Predict 0 degree rotation
Maximize the probability: $f^{90}(x^{90})$
Predict 90 degrees rotation
Maximize the probability: $f^{180}(x^{180})$
Predict 180 degree rotation
Maximize the probability: $f^{270}(x^{270})$
Predict 270 degree rotation

Figure 2: Rotation: The flowers look unchanged after rotation.

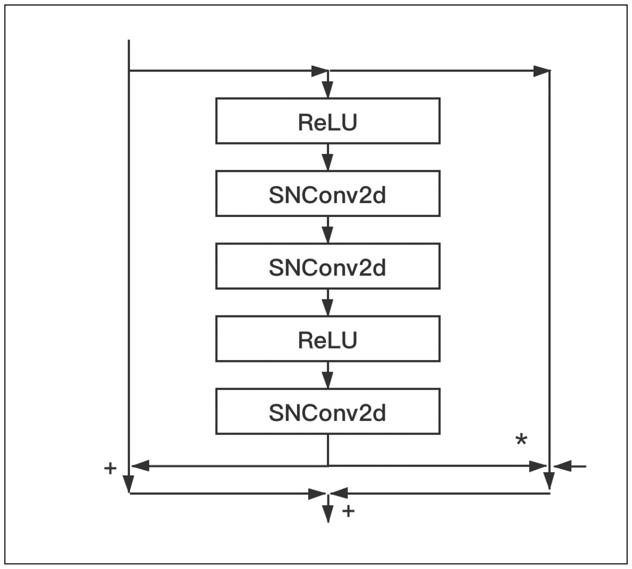

Figure 3: P-SNGAN's basic block architecture in the discriminator. The input from the upper-left is output from the last block, and the input from downright is 1 or Gaussian noise. This Gaussian noise or 1 will do a Hadamard product with the input from the upper-left and current block's output. At last there's a summation of three terms. As for the generator, the second order polynomial term is removed and there's a BN layer before the first ReLU layer and another BN layer between the second SNConv2d layer and the second ReLU layer.

Figure 4: CIFAR10: Synthetic images from SNGAN.

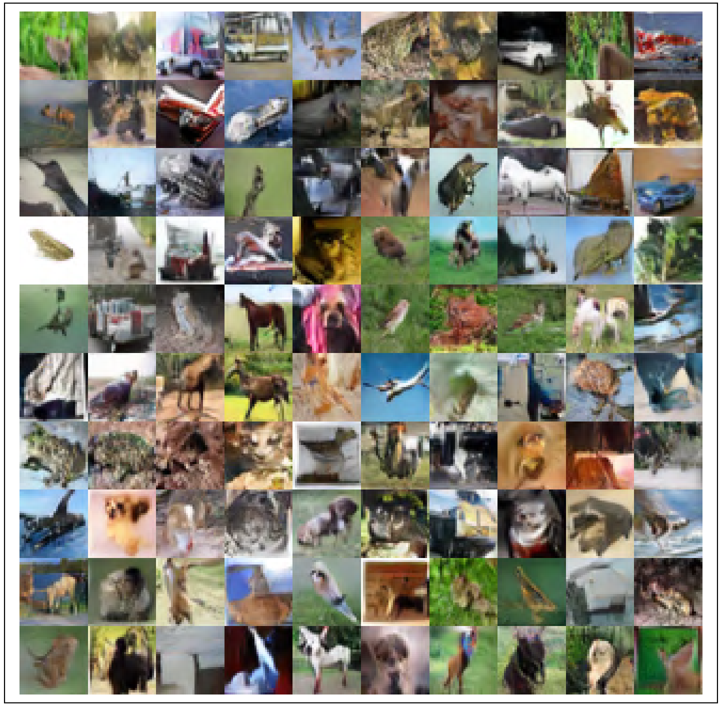
Figure 5: CIFAR10: Synthetic images from P-SNGAN.

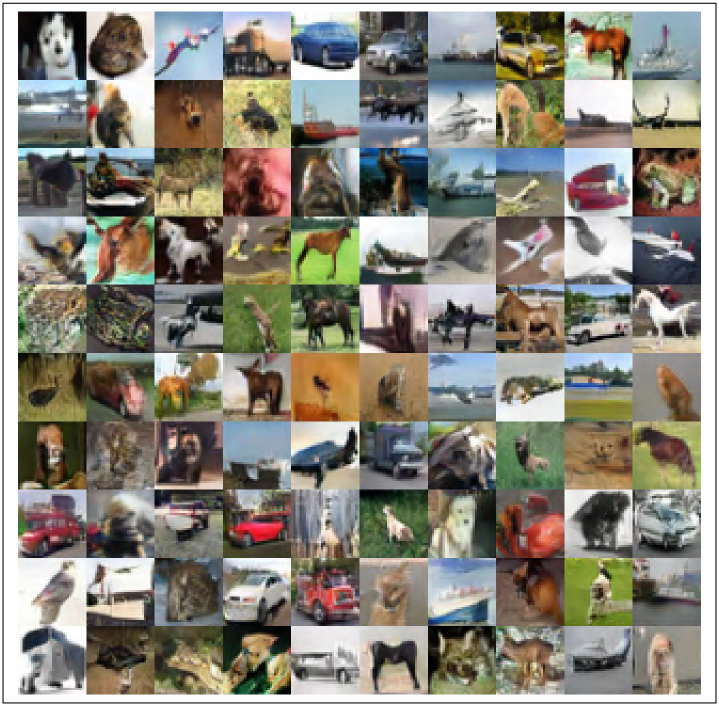
Figure 6: CIFAR10:Synthetic images from P-SS-SNGAN-R.

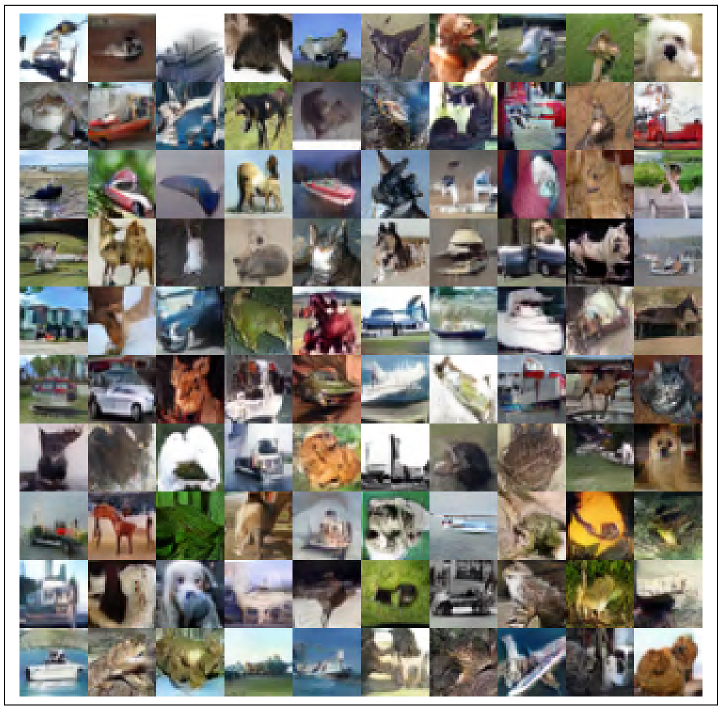
Figure 7: CIFAR10: Synthetic images from P-SS-SNGAN-N.

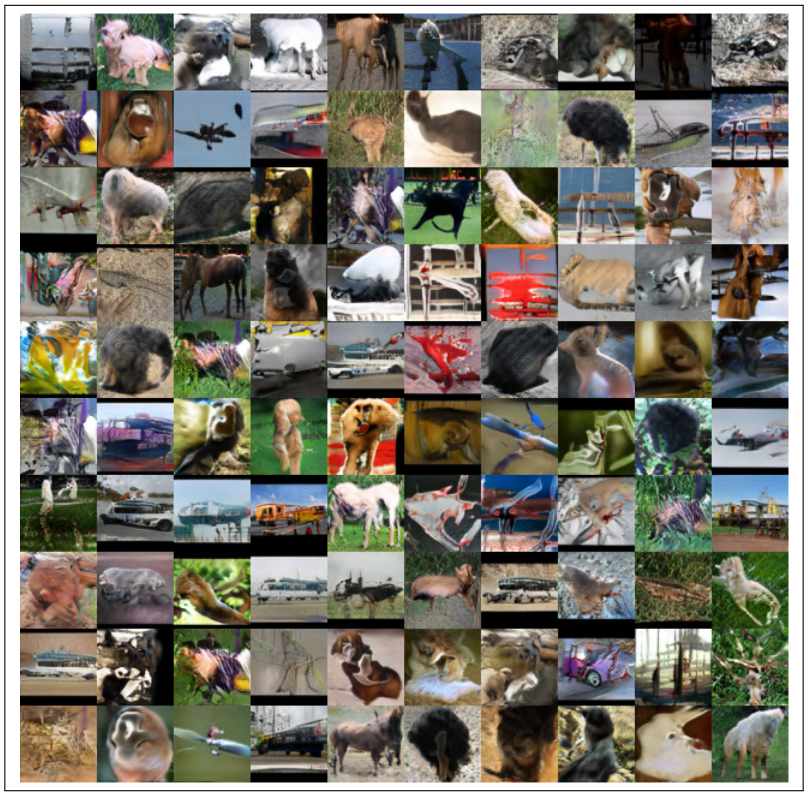
Figure 8: STL10: Synthetic images from P-SS-SNGAN-R.

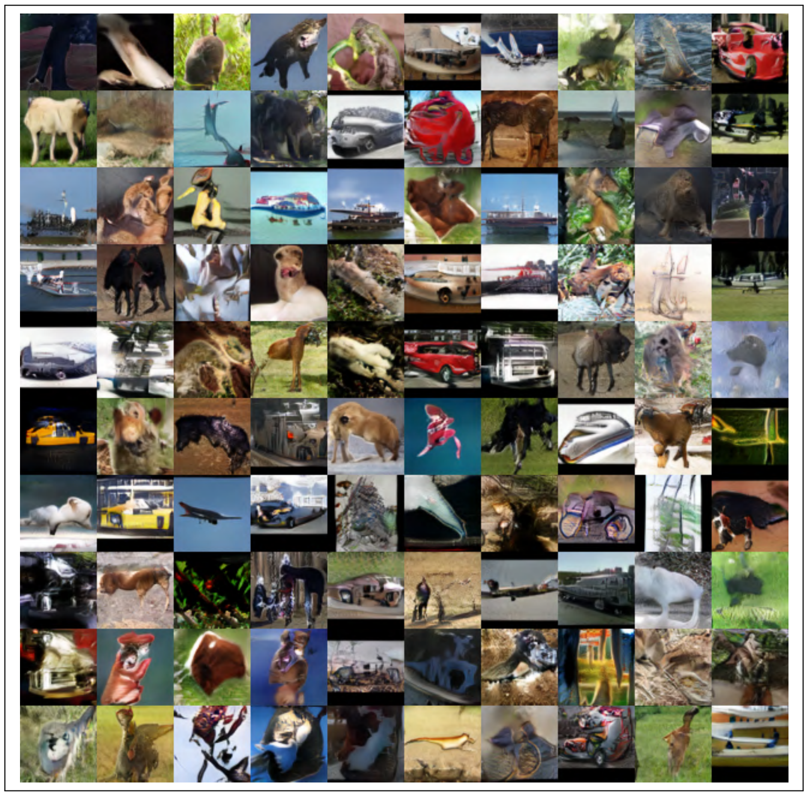

Figure 9: STL10: Synthetic images from P-SS-SNGAN-N.

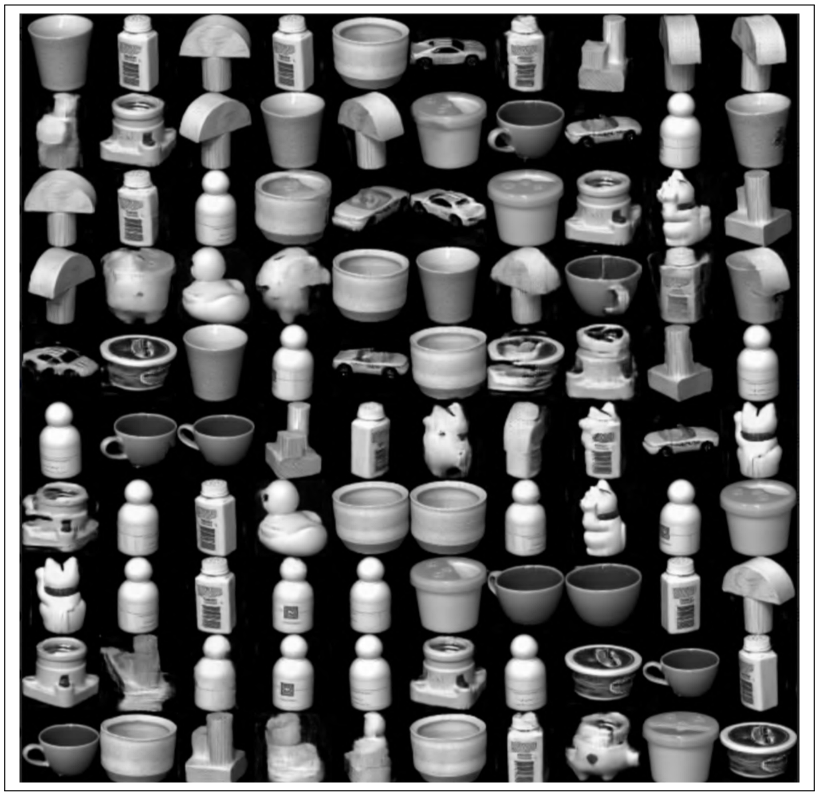

Figure 10: COIL20: Synthetic images from P-SS-SNGAN-R.

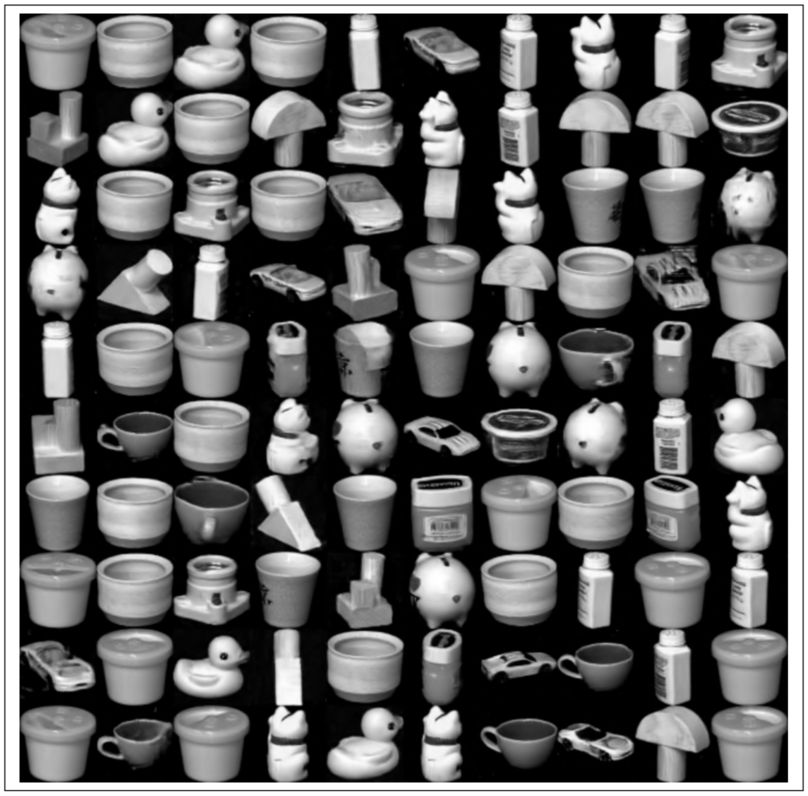

Figure 11: COIL:20 Synthetic images from P-SS-SNGAN-N.

Table 5: Discriminator for the CIFAR10. We use similar architectures to the ones used in [13, 7]. There're in total four basic blocks. Then there's a activation function and a global sum pooling layer. At last are two dense layers.

Image $x \in \mathbb{R}^{32\times32\times3}$
ResBlock downsample
ResBlock downsample
ResBlock
ResBlock
ReLU
Global sum pooling
dense $\rightarrow$ 1, dense $\rightarrow$ 5

Table 6: Discriminator for the STL10. The only difference from Table 5 is: The third basic block will also downsample the images from previous block.

Image $x \in \mathbb{R}^{64\times64\times3}$
ResBlock downsample
ResBlock downsample
ResBlock downsample
ResBlock
ReLU
Global sum pooling
dense $\rightarrow$ 1, dense $\rightarrow$ 5

Table 7: Accuracy score on CIFAR10 via rotation loss. The raw refers to 10 classes of images and the column refers to five times experiment. At last, we average on them.

	1	2	3	4	5	6	7	8	9	10
1	0.68	0.79	0.40	0.77	0.46	0.38	0.67	0.64	0.75	0.67
2	0.70	0.78	0.46	0.54	0.59	0.53	0.69	0.66	0.75	0.52
3	0.58	0.79	0.49	0.52	0.56	0.21	0.76	0.73	0.80	0.57
4	0.75	0.66	0.39	0.61	0.57	0.48	0.72	0.68	0.59	0.76
5	0.70	0.70	0.49	0.50	0.67	0.51	0.71	0.55	0.69	0.76
Ave	**0.682**	**0.744**	**0.446**	**0.588**	**0.570**	**0.422**	**0.710**	**0.652**	**0.716**	**0.656**

Table 8: Accuracy score on CIFAR10 via Gaussian noise loss. The raw refers to 10 classes of images and the column refers to five times experiment. At last, we average on them.

	1	2	3	4	5	6	7	8	9	10
1	0.59	0.67	0.60	0.60	0.54	0.52	0.64	0.57	0.69	0.83
2	0.59	0.82	0.46	0.54	0.50	0.46	0.75	0.59	0.65	0.74
3	0.61	0.57	0.46	0.45	0.45	0.51	0.77	0.70	0.80	0.80
4	0.71	0.77	0.42	0.40	0.50	0.53	0.74	0.71	0.62	0.78
5	0.70	0.70	0.49	0.50	0.67	0.51	0.71	0.55	0.69	0.76
Ave	**0.578**	**0.730**	**0.504**	**0.494**	**0.526**	**0.504**	**0.722**	**0.626**	**0.698**	**0.776**

Table 9: Accuracy score on STL10 via rotation loss. The raw refers to 10 classes of images and the column refers to five times experiment. At last, we average on them. In this experiment the images are resized to 64 $\times$ 64 at first.

	1	2	3	4	5	6	7	8	9	10
1	0.76	0.48	0.78	0.34	0.72	0.12	0.55	0.66	0.49	0.72
2	0.70	0.49	0.63	0.33	0.43	0.22	0.63	0.71	0.56	0.73
3	0.73	0.54	0.64	0.33	0.64	0.27	0.75	0.29	0.51	0.76
4	0.76	0.50	0.62	0.36	0.36	0.17	0.67	0.74	0.63	0.65
5	0.70	0.56	0.78	0.38	0.44	0.14	0.85	0.52	0.48	0.64
Ave	**0.730**	**0.514**	**0.690**	**0.348**	**0.518**	**0.184**	**0.690**	**0.584**	**0.534**	**0.700**

Table 10: Accuracy score on STL10 via Gaussian noise loss. The raw refers to 10 classes of images and the column refers to five times experiment. At last, we average on them.

	1	2	3	4	5	6	7	8	9	10
1	0.69	0.52	0.78	0.48	0.55	0.31	0.52	0.39	0.59	0.51
2	0.71	0.48	0.74	0.40	0.56	0.35	0.39	0.59	0.55	0.57
3	0.80	0.49	0.65	0.47	0.59	0.26	0.49	0.52	0.75	0.26
4	0.77	0.45	0.83	0.42	0.46	0.15	0.73	0.54	0.54	0.40
5	0.67	0.46	0.52	0.68	0.41	0.34	0.62	0.21	0.54	0.71
Ave	**0.728**	**0.480**	**0.704**	**0.490**	**0.514**	**0.282**	**0.550**	**0.450**	**0.594**	**0.490**

Table 11: Accuracy score on COIL20 via rotation loss. The raw refers to 20 classes of images and the column refers to five times experiment. At last, we average on them.

	1	2	3	4	5	6	7	8	9	10
1	1.0	0.53	1.0	1.0	0.53	1.0	1.0	1.0	1.0	1.0
2	1.0	0.07	0.27	1.0	0.93	1.0	1.0	1.0	0.93	1.0
3	0.87	0.27	1.0	1.0	0.40	1.0	1.0	1.0	1.0	1.0
4	0.60	0.53	1.0	1.0	0.87	0.87	1.0	0.87	0.53	1.0
5	0.70	0.70	0.49	0.50	0.67	0.51	0.71	0.55	0.69	0.76
Ave	**0.879**	**0.333**	**0.707**	**1.000**	**0.587**	**0.973**	**1.000**	**0.973**	**0.893**	**1.000**
	11	12	13	14	15	16	17	18	19	20
1	1.0	1.0	0.80	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	0.87	1.0	1.0	1.0	1.0	1.0	0.67	1.0
4	0.87	1.0	0.73	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	0.70	0.70	0.49	0.50	0.67	0.51	0.71	0.55	0.69	0.76
Ave	**0.973**	**1.000**	**0.880**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.973**	**1.000**

Table 12: Accuracy score on COIL20 via Gaussian noise loss. The raw refers to 20 classes of images and the column refers to five times experiment. At last, we average on them.

	1	2	3	4	5	6	7	8	9	10
1	1.0	0.93	1.0	1.0	1.0	0.13	1.0	1.0	0.87	0.60
2	1.0	1.0	0.53	1.0	0.67	1.0	1.0	1.0	0.53	1.0
3	1.0	1.0	0.87	1.0	0.73	0.87	1.0	1.0	0.87	1.0
4	1.0	0.80	0.67	1.0	0.20	0.93	1.0	1.0	0.73	1.0
5	1.0	0.87	0.53	1.0	0.27	0.40	1.01	1.0	0.73	1.0
Ave	**1.000**	**0.920**	**0.720**	**1.000**	**0.573**	**0.667**	**1.000**	**1.000**	**0.747**	**0.920**
	11	12	13	14	15	16	17	18	19	20
1	1.0	1.0	0.27	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	0.60	1.0	0.53	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	0.33	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.87	1.0
5	0.93	1.0	0.93	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Ave	**0.773**	**1.000**	**0.747**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.973**	**1.000**