Random Generative Adversarial Networks

No Author Given

No Institute Given

Abstract. Generative Adversarial Networks surprisingly show great ability in synthesizing high-fidelity. Nonetheless, the problem of so-called *mode collapse* remains a challenge. Borrowing ideas from Random Forest, we propose a novel training method, namely RandomGANs, which utilizes multiple discriminators, each of them will be separately trained on a dataset that is sampled without replacement from the original dataset. By this way, we encourage the discriminators to discover a wide range of different modes of the real data. To save memory and share knowledge, we further require those discriminators to share some first layers. Our training method can be easily applied to a wide range of existing GANs. Indeed, we evaluate the effectiveness and flexibility of our method by using different GAN losses and network architectures. We empirically demonstrate (1) the quality of generated images, (2) the diversity of knowledge in each discriminator to overcome the mode collapse problem, and (3) stabilizing the training process.

Keywords: Generative Adversarial Networks · Random Forest · Deep Generative Model.

1 Introduction

Goodfellow et al. [1] introduced the Generative Adversarial Networks (GANs), a framework for training deep generative models using a minimax game. The goal is to learn a generator distribution $P_G(x)$ that matches the real data distribution $P_{data}(x)$. Instead of trying to explicitly assign a probability to every x in the data distribution, GANs learn a generator network G that generates samples from the generator distribution P_G by transforming a noise variable $z \sim P_{noise}(z)$ into a sample $G(z)$. This generator is trained by playing against an adversarial discriminator network D that aims to distinguish between samples from the true data distribution P_{data} and the generator's distribution P_G. So for a given generator, the optimal discriminator is $D^*(x) = P_{data}(\mathrm{x})/(P_{data}(x) + P_G(x))$. At convergence, the generator learns to produce real-looking images. A few successful applications of GANs are video generation [2], image inpainting [3], image manipulation [4], 3D object generation [5], image to image translation [6], image super-resolution [7], and conditional GANs [8]. More formally, the minimax game is given by the following expression:

$$V(D, G) = E_{x \sim P_{data}}[\log(D(x)] + E_{z \sim noise}[\log(1 - D(G(z)))]$$

Despite the remarkable success of GANs, it suffers from the major problem of *mode collapse*. Though, theoretically, convergence guarantees the generator learning the true data distribution. However, practically, reaching the true equilibrium is difficult and not guaranteed, which potentially leads to the aforementioned problem of mode collapse. Generally, there are two schools of thought to address the issue: (1) improving the learning of GANs to reach better optimal point [9, 10]; and (2) explicitly enforcing GANs to capture diverse modes [11, 12]. Here we focus on the latter.

Borrowing from the multi-agent algorithm [13], we propose a method to ensure knowledge diversity on each discriminator to overcome the mode collapse problem while maintaining equilibrium and stability. This idea is based on associating with Random Forest [14] for generative images in GANs using multiple discriminators, where each discriminator is considered a tree in the forest trained on a random sample from the original data using the bagging technique. In this way, each discriminator becomes an expert to the point of overfitting a specialized data area, thereby encouraging the generator to explore modes of diversity to mislead the discriminators. Our experiments also show that using different layer masks on each discriminator while sharing weights on lower layers, achieves diversity and generality as well as impacts the quality of general images.

In addition, experimental results in the **MNIST** [15] and **CIFAR10** [16] datasets show that our method improves the stability in processing training, along with producing higher quality image samples than other research methods. Specifically, we evaluate the influence of Random Forest in GANs. Each discriminator illustrates different predictions on an evaluation dataset. Hence, the knowledge gained by each discriminator is separated, similarly in Random Forest.

2 Related Work

There exists many papers that utilize the idea of integrating the multi-agent algorithm into different existing GANs architectures.

Ghosh et al. [17] has already used a setup of multiple generators while there is only one discriminator in their model with the output now is not only a probability output whether fake or real the input is but a softmax layer with $K + 1$ outputs for K generators and one more output for real image. This enforces the generator to generate images in different modes, helping model to improve the mode collapse problem.

Hoang et al. [18] has the same idea with Ghosh et al. [17] by using numerous generators but differs from the aforementioned paper, it splits the recognition parts into 2 distinct tasks: a discriminator that recognizes whether input is real or fake and a classifier that classifies which generator was used to generate that image.

Tran et al. [19] has nearly the same direction as us (the same idea in utilizing numerous discriminators) that enables their model to generate diverse and high-quality images by using each data augmentation transformation in each separated discriminator for both real data and synthesized data. It's different from our perspective that because we follow the bagging technique, at each discriminator there will be a different batch of images instead of the same dataset for all discriminators like them.

MD-GAN: Hardy et al. [20] proposes a method to distribute GANs over a set of worker machines. In order to provide an answer to the computational load challenge on workers, they remove half of their burden by having a single generator in the system, hosted by the parameter server. This is made possible by a peer-to-peer like communication pattern between the discriminators spread on the workers.

Other directions are given such as replacing KL distance in original GANs to Earth Mover's Distance [21] in [22] or using more discriminators in which each discriminator only observes one specific sample by a fixed projection. It reduces the effect of d then resulting in stability in training in [23]. Another way that uses more discriminators is [24] which suggests using hypervolume-maximization [25]. A new multi-player objective named Discriminator Discrepancy Loss (DDL) [26] for diversifying the multi-discriminators introduced layer-sharing architecture for the multi-discriminators, which encourages harmonious collaboration between different discriminators, thus further enhancing the diversity.

However, as mentioned above, previous methods using multiple d allow it to train on the same dataset with a similar training process. As a result, although there are more d, acknowledgment gained is relatively the same, and gradients throughout training are not general to the generator.

Starting from above problems, we suggest using the idea of Random Forest in GANs. This method depends on producing overfitting trees on different datasets, and the predicted result is general of each tree to avoid instability of GANs. Specifically, when training D so d is trained in turn on each own dataset whereas when G is trained, information of gradients is obtained from the predicted result of d.

3 Proposed Method

In this section, we introduce Random Forest to GANs framework, enabling each discriminator in our discriminator forest to learn different knowledge, as a result helps generator to generate images with higher diversity and resolve the mode-collapse problem. We call the proposed method Random Generative Adversarial Networks (RandomGANs) because of its Random Forest module. This module is organized into two main parts as two main contributions of our method: The full design of our discriminator's system and its training process is illustrated in Section 3.1. Our final model can be observed in Fig. 1. Section 3.2 demonstrates Masking Layer and Layer-Sharing mechanisms that play an important role in making sure the diversity of knowledge each discriminator gains.

Below are the explanation of variables that be used from now on and the rest of this paper:

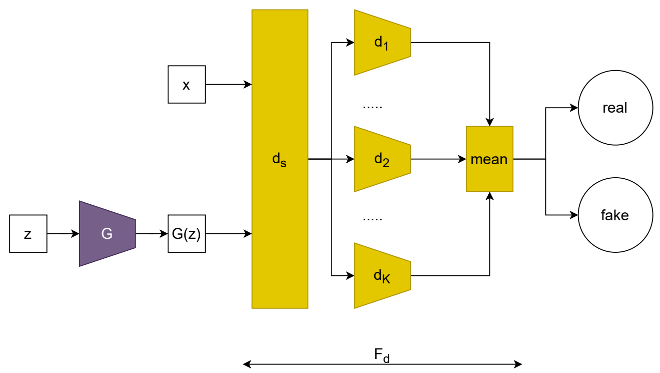


Fig. 1: Overview architecture of RandomGANs

- G: Generator in GANs model.
- D: Discriminator in GANs model (for model that uses only one Discriminator).
- K: number of discriminators used in multi-discriminator model.
- d_s: layer-sharing between numerous discriminators in multi-discriminator model.
- $d_i,\ i \in \{1, 2, ..., K\}$: each discriminator with its index in multi-discriminator model.
- $D = \{d_s, \{d_1, d_2, ..., d_K\}\}$: a full architecture of Discriminator in multi-discriminator model which includes layer-sharing d_s and numerous seperated layers $\{d_1, d_2, ..., d_K\}$.
- n: number of real instances.
- $D(.)$: a function that receives an input data x, returns $D(x)$ which is the probability of that data instance to be real or fake.

$$D(x) = \frac{1}{K} \sum_{k=1}^{K} d_k(d_s(x)) \tag{1}$$

- $S(k),\ k \in \{1, 2, ..., K\}$: sample of images that used in discriminator d_k during training process.

3.1 Random Generative Adversarial Networks (RandomGANs)

The use of multiple d has helped to stabilize the training process, but with previous methods, the output of each d was relatively similar, resulting in the knowledge learned by them was not too much different; thus the derivative information synthesized from d when training G was not highly general. Stemming from that, we propose the method of applying the idea of Random Forest in GANs, as mentioned in the previous sections. Besides, recent research has also shown that the sharing of parameters between discriminators helps to increase their generalization ability, so this paper uses multiple d architecture in which these d share parameters in some first layers. We notate that the model includes G and $D = d_s, d_1, d_2, ..., d_K$, where d_s

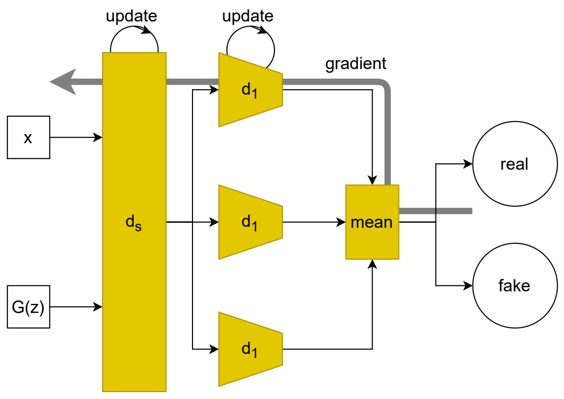


Fig. 2: Design of Discriminator's system in a RandomGANs model and updating process for each head during training process.

is the shared part of $d_1, d_2, ...d_K$ and K is the number of d, as can be seen in Fig. 2. The loss function of G is defined:

$$L_G = \mathbb{E}_{z\sim p_z} \log\left(1 - \frac{1}{K}\sum_{k=1}^{K} d_k(d_s(G(z)))\right) \tag{2}$$

Unlike the previous methods, while the loss function of G is the total loss of each d, the proposed method has only a single loss function. However, the output of D is the general synthesization of each d. The loss function for D is defined:

$$D(x) = \frac{1}{K}\sum_{k=1}^{K} d_k(d_s(x)) \tag{3}$$

$$L_D = -\mathbb{E}_{x\sim p_x} \log(D(x)) - \mathbb{E}_{z\sim p_z} \log(1 - D(G(z))) \tag{4}$$

The training process of D is similar to the previous methods of training each d. For each d computing the corresponding error function L_D on the dataset of dummy images generated by G and real images that d-tree is allowed to use for learning, and then update among the d-tree and the shared class. The process continues until all d in D have been learned. Although the d are trained in turn, it can be seen from formula 3 that the derivative information depends on the common prediction of all d in the forest when updating any d. Hence there is a greater amount of knowledge that d can learn than when just training each loss function for each d independently. Moreover, the fact that each d is only trained with a part of the original data also helps the amount of knowledge of each d be different and makes the aggregated result of the whole forest more general and stable. The training process of Random-

GANs is illustrated in the pseudocode below:

Algorithm 1: Learning with RandomGANs

```
1 for iter=1 to #iterations do
2     Sample $m$ noise $\{z_1, z_2, ..., z_m\}$ from distribution $p_z$
3     for k=1 to $K$ do
4         Sample $m_1$ observations $\{x_1, x_2, ..., x_{m_1}\}$ from $p_{data}$ satisfy
           $x_i \in S(k),\ \forall i \in \{1, 2, ..., m_1\}$
5         Update $\{\theta_{d_s}, \theta_{d_k}\}$ by minimizing $L_D$ as provided in (3) and (4)
6     end for
7     Sample $m$ noise $\{z_1, z_2, ..., z_m\}$ from distribution $p_z$
8     Update $\theta_g$ by minimizing $L_G$ in (2)
9 end for
```

3.2 Masking Layer combined with Layer-Sharing Architecture

The low-level network layer describes the general features of the data. This has little impact on Discriminator but is significant for image generation. A Layer-Sharing network architecture for multiple discriminators has been proposed in [26], which allows them to learn low-level shared features. We suggest the Masking layer to ensure that learned knowledge diverges in the mid-level network layer by having connections of each different architecture. Using more discriminators enhances performance but leads to over-parameterization and reduced training stability. Layer-sharing reduces the number of parameters when sharing a low-level layer between discriminators. However, deeper network architectures of discriminators will quickly lead to a large number of parameters. Masking layer combined with Layer-sharing solves both.

4 Experiments

We describe our experimental setups which consist of datasets, evaluation protocol and network architectures as well as parameters in Sections 4.1, 4.2 and 4.3 followed by results presented in mode collapse problem, how different knowledge each d learns compared to each others and effectiveness of number of heads used in our model in the last section. We also analyze the impact of our method on stabilizing the training process between generator and discriminator and the quality of synthesizing images.

4.1 Data

The experiments have been conducted using **MNIST** [15] and **CIFAR10** [16] datasets. Input image to Discriminator will be resized to 28×28 for the former and the latter is 32×32. Value in each pixel of image is scaled to interval $(0,1)$ and standardized to have $mean = 0.5$ and $std = 0.5$. For input of Generator, noise z will be sampled from a normal distribution with 100 dimensions. During evaluation process, 5000 random noise will be sampled at each evaluation metric.

4.2 Evaluation Metric

Fréchet Inception Distance (FID) We use Fréchet Inception Distance (FID) which is proposed by [27] as a measurement of similarity between two given image datasets. The lower the value of FID, the better in quality and diversity of the synthesized image dataset and vice versus. FID is the Fréchet distance [28] between two image representation datasets from the original two image datasets after the process of feed-forwarding through Inception Network [29] that is pre-trained.

Width range of each d prediction in *D* This evaluation metric is used to estimate that the amount of knowledge that each discriminator gained after the training stage is different from each others by measuring how distinct the prediction output at each head (each discriminator) is with input images from the test dataset. Specifically, let $\overline{x}$ be input of D on behalf of real image and fake image which is synthesized by G, $\{\overline{x}_1, \overline{x}_2, ..., \overline{x}_m\}$ be input dataset of D, m is number of samples, usually 5000. Then, width range of the prediction W_D on the dataset $\{\overline{x}_i\}_{i=1}^{m}$ is defined by:

$$S_i = \{d_1(d_s(\overline{x}_i)), d_2(d_s(\overline{x}_i)), ..., d_K(d_s(\overline{x}_i))\} \tag{5}$$

$$W_{D(\overline{x})} = \frac{1}{m}\sum_{i=1}^{m}[\max(S_i) - \min(S_i)] \tag{6}$$

4.3 Network Architecture and Implementation Details

Enriklindernoren Architecture The architecture of G and D is designed following the architecture that is implemented in [30], which applies Spectral Normalization [31] to weights of all Linear layers in D. Each d in D is carefully designed with a unique masking layer in order to guarantee the basic difference between each others regarding architecture. Empirically, this can be considered as the most natural way to make sure that knowledge learnt by each d during training process is different from each others. All multiple discriminators models in evaluation stage are designed with the same architecture in G and D, for models that only have 1 discriminator which includes (d_s, d_1), in this case, we do not utilize mask in the Linear layer of d_1. In the LeakyReLU, the slope of the leak was set to 0.2 in all models.

SAGAN Architecture [32] For Generator, the same architecture as [32] has been applied and for Discriminator part D which includes $\{d_1, d_2, .., d_K\}$ as list of discrimi-

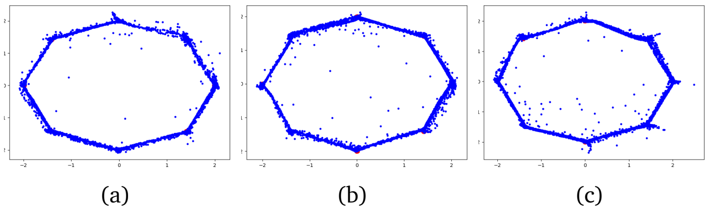


Fig. 3: a-c) Examples generated on our 2D-Ring synthetic dataset via different methods: (a) Vanilla GANs (b) HGAN (c) RandomGANs.

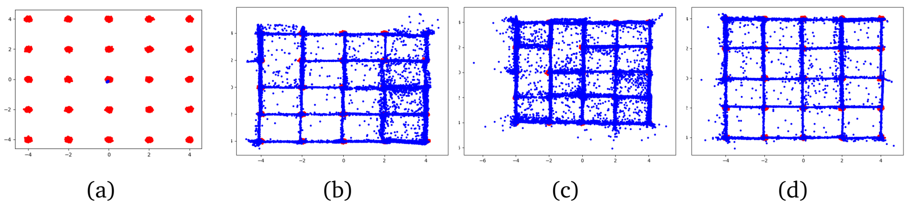


Fig. 4: a-d) Examples generated on our 2D-Grid synthetic dataset via different methods: (a) real data (b) Vanilla GANs (c) HGAN (d) RandomGANs.

nators, each d_i has the same architecture as [32]. At this moment, there is no shared layer between all d_i in D.

Experimental parameters We trained 450 epochs for Eriklindernoren architecture and 120 epochs for SAGAN Architecture. Adam Optimizer using $\beta_1 = 0.5, \beta_2 = 0.999$ with learning rate is 0.0002 and batch size 64. Noise z has 100 dimensions sampled from normal distribution $\mathcal{N}(0, \mathrm{I})$. All evaluation results are recorded every 4 epochs. The number of samples used to evaluate each metric is 5000.

4.4 Result

Evaluation on mode-collapse problem In this experiment, we conduct on 2D-ring dataset, which is a mixture of 8 2D-Gaussians and 2D-grid dataset, which is a mixture of 25 2D-Gaussians, each with means $(2i-4, 2j-4)$ and variance 0.0025, for $i, j \in \{0, ..., 4\}$.

We follow the metrics used in prior work [33,34]. A generated point is deemed high-quality if it is within three standard deviations from some mean [34]. The number of modes covered by a generator is the number of means that have at least one corresponding high-quality point. To compare the reverse KL divergence between the generated distribution and the real distribution, we assign each point to its closest

Table 1: Number of modes recovered, percent high-quality samples, and reverse KL divergence metrics for 2D-Grid and 2D-Ring experiments.

Method	2D-Ring			2D-Grid		
	Modes ↑	% ↑	Reverse KL ↓	Modes ↑	% ↑	Reverse KL ↓
GAN	8	92.2	$5.05e-05$	25	78.8	0.0014
HGAN	8	80.9	0.00022	25	53.7	0.00426
RandomGANs	8	80.07	0.00017	**25**	**79.2**	**0.00074**

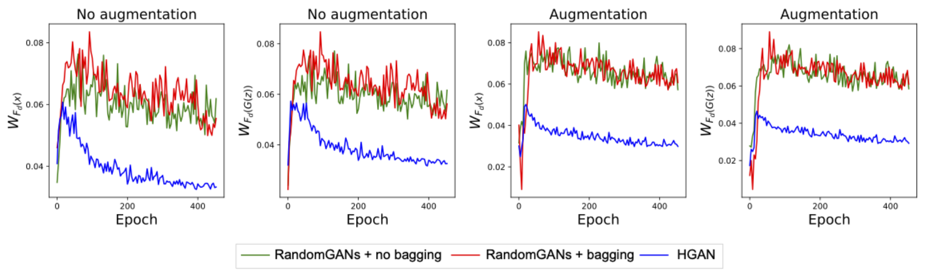


Fig. 5: Result on how different output prediction of each d in D using Eriklindernoren architecture with ($K = 10$).

mean in Euclidean distance, and compare the empirical distributions [33]. Results of these experiments are illustrated in Table 1 as well as in Fig. 3 and 4.

Evaluation on how different knowledge of each d in D compared to each others
We first start with number of heads (number of discriminators d) $K = 10$ for Eriklindernoren Architecture and $K = 5$ for SAGAN Architecture. This experiment is conducted on HGAN, RandomGANs without Bagging, and RandomGANs with Bagging with 4 experimental setup cases: (1) Eriklindernoren architecture without data augmentation; (2) Eriklindernoren architecture with data augmentation; (3) SAGAN without data augmentation; (4) SAGAN with data augmentation. In each experimental setup, the measurement of difference between prediction output of each d is calculated on real image dataset and image dataset that produced by generator.

As in Fig. 5, knowledge that each d in the proposed RandomGANs model obtained is much more different from each others, compared to HGAN, where this distance is just a little number. All of these observations are explicitly shown throughout all 4 figures. Considering the proposed RandomGANs method, in case we do not utilize data augmentation, the difference between prediction output on each d when using bagging technique is larger than when this technique is not applied. This sounds completely reasonable for the fact that when bagging technique is performed, each head of discriminator is able to perceive only a small part of the original dataset, thus knowledge that each d gained after training stage is distinct from each others, results in a significant difference in prediction output of each d on evaluated dataset.

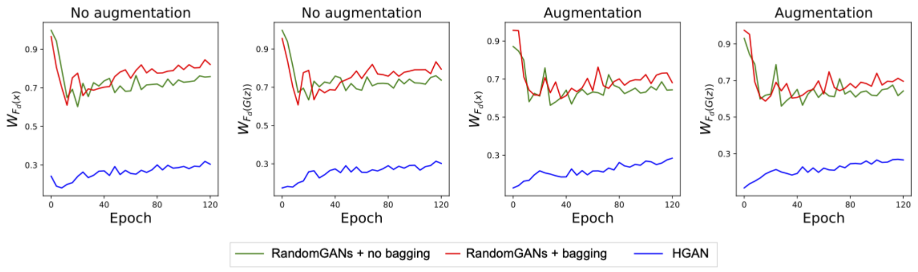


Fig. 6: Result on how different output prediction of each d in D using SAGAN architecture with ($K = 5$).

The experimental results shown in Fig. 6 indicate similar consequences when comparing the effectiveness of applying RandomGANs method on SAGAN and Eriklindernoren architecture. Moreover, in the case of SAGAN architecture, even after our original dataset is fed through data augmentation, the difference between prediction output on each d when utilizing bagging technique is greater than the case in that we exclude this technique. These results also demonstrate that when comparing the difference between knowledge learnt by each head in list of discriminators in RandomGANs method using two architectures: SAGAN and Eriklindernoren, the former outperforms the latter. This can be explained by the reason that in Eriklindernoren architecture, there is a shared layer between all d in D, resulting in sharing information between all of these heads, thus making the difference in prediction output of each d compared to each others become small.

Evaluate the effectiveness on number of heads used in RandomGANs method All experiments in this section do not utilize data augmentation and are only conducted following RandomGANs method in 2 cases: bagging and without bagging technique. In terms of Eriklindernoren architecture, the number of heads (number of d in D) ranges in $\{3, 5, 10\}$ while in SAGAN architecture, the number of heads that we estimate in this experiment is 3, 5 and 7.

Table 2 shows that when increasing number of heads (number of d), the difference between prediction output on each d in D also increases, and the quality and diversity of synthesized image dataset are better but when number of heads is really huge, the quality of synthesized image deteriorates or the improvement is not much. Besides, including bagging technique usage in RandomGANs method makes the difference in prediction output between each d becomes higher as discussed in the previous section and results in table 2 explicitly demonstrate this conclusion. In case $K = 7$ with SAGAN architecture, using bagging technique does not improve the difference because when bagging technique is not utilized, there exists some heads where prediction output on real and fake images achieves very high accuracy, results in the width range of prediction output on each d approximately reaches 1. So there exists a question that why does when the difference between prediction output

Table 2: Best FID recorded after finishing the training stage with different numbers of discriminators in Discriminator's forest.

	Eriklindernoren		SAGAN	
	Without Bagging (1)	Bagging (2)	Without Bagging (3)	Bagging (4)
K=3	13.84	14.44	6.87	7.19
K=5	**13.05**	**14.10**	**6.34**	6.83
K=7	-	-	6.51	**6.80**
K=10	14.51	14.9	-	-

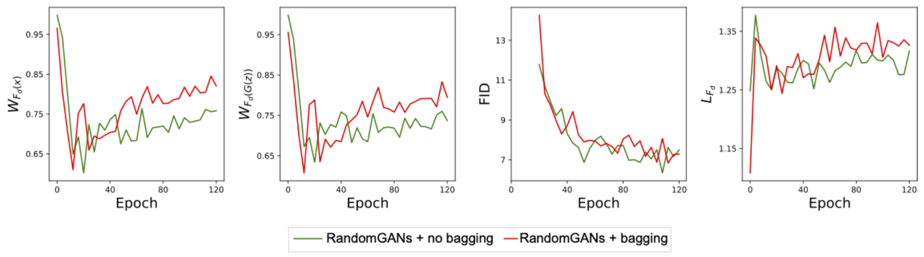


Fig. 7: Result of RandomGANs using SAGAN architecture without data augmentation for ($K = 5$).

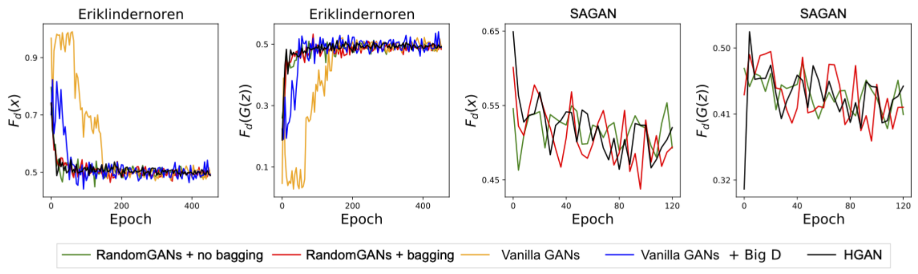


Fig. 8: Prediction output of D for real and fake images input on different types of model.

on each d in D increases to a fixed amount, the quality of synthesized image deteriorate? To solve this issue, let consider the case when using SAGAN architecture, $K = 5$, with and without bagging.

Considering experimental results in Fig. 7, when using bagging technique, the difference between prediction output of each d in D is much larger, moreover, each of heads only sees small part of the original dataset, thus making them become weaker. Moreover, observing from Fig. 8, applying our RandomGANs method to both Eriklindernoren and SAGAN architecture makes their training process to have a more

Table 3: Best FID throughout the training stage recorded and calculated on batch of 5000 instances of image.

Architecture	Eriklindernoren		SAGAN	
	Without augmentation	With augmentation	Without augmentation	With augmentation
Vanilla GANs	20.35	11.59	-	-
Vanilla GANs + Big D	16.94	12.86	-	-
HGAN	14.75	**10.53**	6.51	5.90
RandomGANs + No Bagging	**14.51**	10.62	**6.34**	**5.72**
RandomGANs + Bagging	14.99	11.41	6.83	5.96

stabilizing tendency compared to baseline GANs model as the prediction of both real and fake images gradually converged to optimal threshold.

Evaluate the stability of model This experiment is conducted with 2 architectures: Eriklindernoren and SAGAN, both of them do not utilize data augmentation. For Eriklindernoren, we execute 3 different following architectures: GANs model [1] with two versions which mentioned in Proposed Method section; RandomGANs model without bagging and $K = 10$; RandomGANs model with bagging and $K = 10$; HGAN model with $K = 10$. For SAGAN architecture, 3 following models are evaluated: RandomGANs model without bagging and $K = 5$; RandomGANs model with bagging and $K = 5$; HGAN model with $K = 5$. There are a significant number of evaluation metrics to evaluate the stability when training a GANs model, one of them is to carefully record the value of prediction output of discriminator with real and fake image as input. In theory, as already proved in original GANs model [1], value of prediction output of discriminator for both real and fake input will approach to 0.5.

Experimental results show that proposed RandomGANs model outperforms both two versions of the original GANs model in case of stability. Comparing to HGAN, RandomGANs show the similarity of stability. In addition, Eriklindernoren architecture tends to show better performance than SAGAN architecture in terms of stability.

Evaluate quality of synthesized image All experiments in this section are conducted on both Eriklindernoren and SAGAN architecture also with two experimental setups: before and after augmentation is applied. In the case of Eriklindernoren architecture, each experiment is executed on 5 different models, namely: Original GANs model [1]: Execute big and small architecture that mentioned in the Proposed Method section; RandomGANs with bagging and $K = 10$; RandomGANs without bagging and $K = 10$; HGAN with $K = 10$. While in SAGAN architecture, all experiments are executed on only 3 different kinds of model namely: RandomGANs with bagging and $K = 5$; RandomGANs without bagging and $K = 5$; HGAN with $K = 5$.

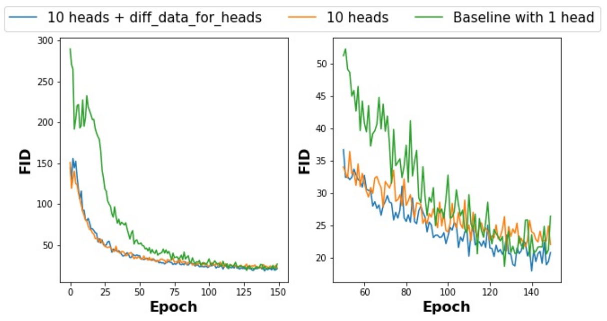


Fig. 9: Result when applying our method with LSGAN compared to LSGAN baseline.

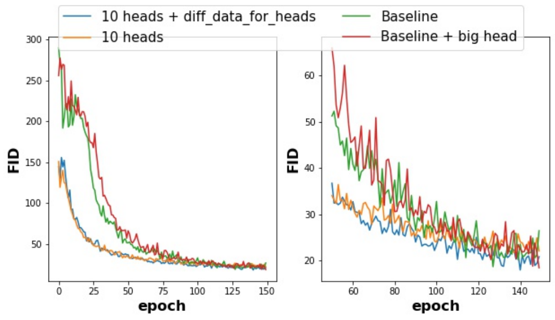


Fig. 10: Result when applying our method with WGAN compared to WGAN baseline.

Fig. 9 and 10 show FID value during the training process when applying our RandomGANs method to different GANs' loss function namely: least-square loss and Wasserstein loss. In this experiment, we compare our model in 4 cases: baseline, RandomGANs with and without bagging, baseline with big head (we increase the number of parameters in this model so that it is equal to RandomGANs model). All of these models are constructed under the Eriklindernoren architecture. As can be easily seen in two GANs' loss function versions, the method RandomGANs with 10 heads and bagging achieves the greatest result throughout the training phase, outperforms the method Baseline and Baseline with big head. This result helps us to conclude that our method can obtain good result by the diversity of the knowledge that each discriminator in the discriminator's forest observe but by making model become complex. Table 3 illustrates in detail the best FID value recorded of each model.

When testing on the RGB images (here we choose the dataset to be CIFAR10), our method with 10 heads and bagging still remains the best result compare to each others during the training process as shown in Fig. 11.

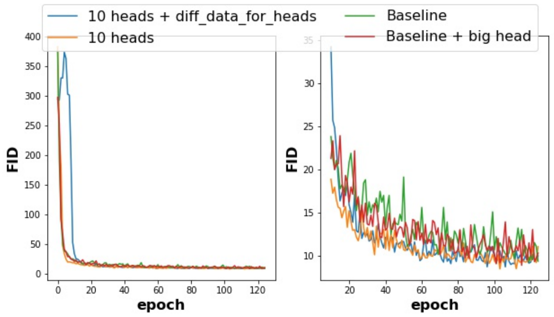
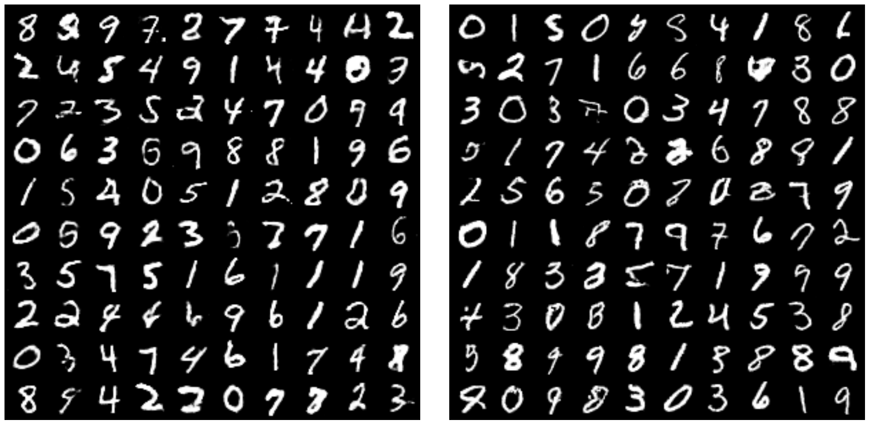


Fig. 11: FID value during the training process when applying our method to SAGAN on CIFAR10 dataset.



FID=10.62 FID=5.72

Fig. 12: Some handwritten images that generated by the proposed RandomGANs following Eriklindernoren (left) and SAGAN (right) architecture.

5 Conclusion

In this paper, we proposed Random Generative Adversarial Networks (RandomGANs), which incorporate Random Forest algorithm into the GANs framework. This type of framework is effective in not only encouraging GANs model to discover new modes in the input data but also stabilizing generator and discriminator during training process. Various experiments that we've already conducted show that FID value of images produced by our model is better than other architectures and we also show that each tree of the discriminator forest learns different knowledge compared to each others by illustrating how different prediction output of an arbitrary head compared to prediction output of other heads.

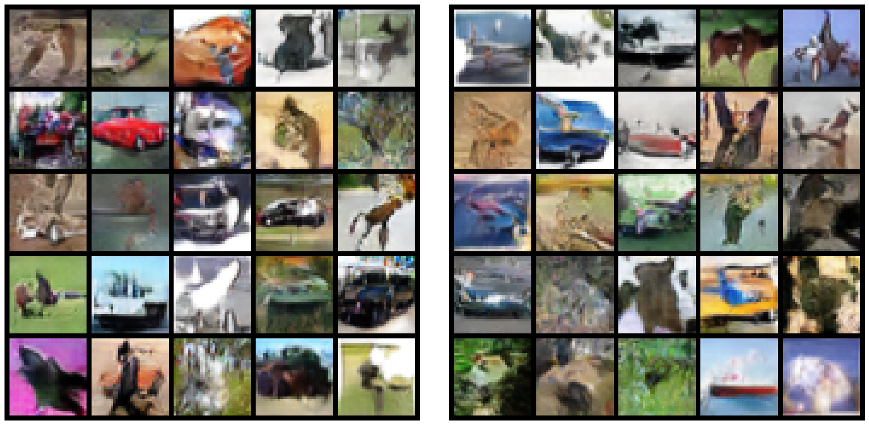


Fig. 13: Some RGB images that generated by the proposed RandomGANs following SAGAN architecture.

References

1. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil O., Aaron C., and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.
2. Sangeek Hyun, Jihwan Kim, and Jae-Pil Heo. Self-supervised video gans: Learning for appearance consistency and motion coherency. In *CVPR*, 2021.
3. Ryo Fujii, Ryo Hachiuma, and Hideo Saito. Rgb-d image inpainting using generative adversarial network with a late fusion approach, 2021.
4. Shoya Matsumori, Yuki Abe, Kosuke Shingyouchi, Komei Sugiura, and Michita Imai. Lattegan: Visually guided language attention for multi-turn text-conditioned image manipulation. *IEEE Access*, 2021.
5. Sebastian Lunz, Yingzhen Li, Andrew W. Fitzgibbon, and Nate Kushman. Inverse graphics gan: Learning to generate 3d shapes from unstructured 2d data. *CVPR*, 2020.
6. Zhenliang He, Meina Kan, Jichao Zhang, and S. Shan. Pa-gan: Progressive attention generative adversarial network for facial attribute editing. *CVPR*, 2020.
7. Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. *ICCV, 2019*.
8. Grigorios G. Chrysos, Jean Kossaifi, and Stefanos Zafeiriou. Rocgan: Robust conditional gan. *International Journal of Computer Vision*, 2020.
9. Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR 2017*.
10. Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.
11. Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *ICLR, 2017*.
12. Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. NIPS, 2016.

13. Martín Abadi and David G. Andersen. Learning to protect communications with adversarial neural cryptography. *ICLR*, 2017.
14. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
15. Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.
16. Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5), 2014.
17. Arna Ghosh, Viveka Kulharia, Vinay P. Namboodiri, Philip H. S. Torr, and Puneet Kumar Dokania. Multi-agent diverse generative adversarial networks. *CVPR*, 2018.
18. Quang Minh Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Q. Phung. Mgan: Training generative adversarial nets with multiple generators. In *ICLR*, 2018.
19. Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021.
20. Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2019.
21. Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *ICCV*, 1998.
22. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.
23. Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. Stabilizing gan training with multiple random projections. *arXiv preprint arXiv:1705.07831*, 2017.
24. Isabela Albuquerque, João Monteiro, Thang Doan, Breandan Considine, Tiago Falk, and Ioannis Mitliagkas. Multi-objective training of generative adversarial networks with multiple discriminators. In *ICML*, 2019.
25. Mark Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *EMO*, 2003.
26. Ying Jin, Yunbo Wang, Mingsheng Long, Jianmin Wang, Philip S. Yu, and Jiaguang Sun. A multi-player minimax game for generative adversarial networks. In *ICME*, 2020.
27. 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. *arXiv preprint arXiv:1706.08500*, 2017.
28. Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22:1–72.
29. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
30. https://github.com/eriklindernoren/pytorch-gan.
31. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.
32. Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019.
33. Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *JSAIT*, 2020.
34. Akash Srivastava, Lazar Valkov, Chris R., Michael U G., and Charles S. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NIPS*, 2017.