Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy

ZHENGWEI WANG, School of Computer Science and Statistics, Trinity Collge Dublin, Ireland
QI SHE, ByteDance AI Lab, Beijing, China
TOMÁS E. WARD, Insight Centre for Data Analytics, School of Computing, Dublin City University, Ireland

Generative adversarial networks (GANs) have been extensively studied in the past few years. Arguably their most significant impact has been in the area of computer vision where great advances have been made in challenges such as plausible image generation, image-to-image translation, facial attribute manipulation, and similar domains. Despite the significant successes achieved to date, applying GANs to real-world problems still poses significant challenges, three of which we focus on here. These are as follows: (1) the generation of high quality images, (2) diversity of image generation, and (3) stabilizing training. Focusing on the degree to which popular GAN technologies have made progress against these challenges, we provide a detailed review of the state-of-the-art in GAN-related research in the published scientific literature. We further structure this review through a convenient taxonomy we have adopted based on variations in GAN architectures and loss functions. While several reviews for GANs have been presented to date, none have considered the status of this field based on their progress toward addressing practical challenges relevant to computer vision. Accordingly, we review and critically discuss the most popular architecture-variant, and loss-variant GANs, for tackling these challenges. Our objective is to provide an overview as well as a critical analysis of the status of GAN research in terms of relevant progress toward critical computer vision application requirements. As we do this we also discuss the most compelling applications in computer vision in which GANs have demonstrated considerable success along with some suggestions for future research directions. Codes related to the GAN-variants studied in this work is summarized on https://github.com/sheqi/GAN_Review.

CCS Concepts: • **Computing methodologies → Computer vision**; *Reconstruction*; *Unsupervised learning*; *Neural networks*;

Additional Key Words and Phrases: Generative adversarial networks, computer vision, architecture-variants, loss-variants, stabilizing training

ACM Reference format:
Zhengwei Wang, Qi She, and Tomás E. Ward. 2021. Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy. *ACM Comput. Surv.* 54, 2, Article 37 (February 2021), 38 pages.
https://doi.org/10.1145/3439723

This work is funded as part of the Insight Centre for Data Analytics which is supported by Science Foundation Ireland under Grant Number SFI/12/RC/2289_P2.
Authors' addresses: Z. Wang, V-SENSE, School of Computer Science and Statistics, Stack B, Custom House Quay, IFSC, Dublin 1, Ireland, Postcode: D01 Y6C3; email: zhengwei.wang@tcd.ie; Q. She (corresponding author), Xi Zhi Men Wai Nan Lu Building 1, Xicheng District, Beijing, China, Postcode: 100044; email: sheqi1991@gmail.com; T. E. Ward, Insight Centre for Data Analytics, Dublin City University, Glasnevin, Dublin 9, Ireland, Postode: D09 W6Y4; email: tomas.ward@dcu.ie.

0360-0300/2021/02-ART37 $15.00
https://doi.org/10.1145/3439723

1 INTRODUCTION

Generative adversarial networks (GANs) are attracting growing interest in the deep learning community [45, 83, 107, 117, 128, 136]. GANs have been applied to various domains such as computer vision [38, 72, 79, 89, 106, 130, 139, 152], natural language processing [28, 40, 59, 141], time-series synthesis [15, 34, 39, 48, 75], semantic segmentation [37, 86, 112, 122, 155] and so on. GANs belong to the family of generative models in machine learning. Compared to other generative models, e.g., variational autoencoders, GANs offer advantages such as the ability to handle sharp estimated density functions, the ability to efficiently generate desired samples, the elimination of deterministic bias, and good compatibility with the internal neural architecture [44]. These properties have allowed GANs to enjoy great success especially in the field of computer vision, e.g., plausible image generation [22, 65, 110, 132, 150], image-to-image translation [23, 58, 82, 83, 90, 125, 153, 154], image super-resolution [37, 74, 91, 92, 133], and image completion [21, 33, 77, 142, 146].

However, GANs are not without problems. The two most significant are that they are hard to train and that they are difficult to evaluate. In terms of being difficult to train, it is non-trivial for the discriminator and generator to achieve Nash equilibrium during training and it is common for the generator to fail to learn well the full distribution of the datasets. This is the well-known mode collapse issue. Lots of work has been carried out in this area [27, 68, 69, 78]. In terms of evaluation, the primary issue is how best to measure the dissimilarity between the real distribution of the target p_r and the generated distribution p_g. Unfortunately accurate estimation of p_r is not possible. Thus, it is challenging to produce good estimations of the correspondence between p_r and p_g. Previous work has proposed various evaluation metrics for GANs [10, 12, 46, 47, 49, 124, 134, 135, 138] and it is an active area of research. However, it is the first set of problems, those associated with training, and in particular those concerning image quality, image diversity, and stability that we are concerned with here. In this work, we are going to study existing GAN-variants that handle this aspect in the area of computer vision. Those readers interested in the evaluation challenge may consult [12, 124].

Much of current GAN research can be considered in terms of the following two objectives: (1) the improvement of training and (2) the deployment of GANs for real-world applications. The former seeks to improve GANs performance and is therefore a foundation for the latter, i.e., applications. Considering the many published results that deal with GAN training improvement, we present a succinct review on the most important GAN-variants that focus on this aspect in this article. The improvement of the training process provides benefits in terms of GANs performance as follows: (1) improvements in the generated image diversity (also known as mode diversity), (2) increases in generated image quality, and (3) stabilizing training such as remedying the vanishing gradient issue for the generator. To improve the performance as mentioned above, modifications for GANs can be done from either the architectural side or the loss perspective. We will study these GAN-variants according to each perspective in terms of how they improve performance.

The rest of the article is organized as follows: (1) We introduce related review work for GANs and illustrate the difference between those reviews and this work, (2) we give a brief introduction to GANs, (3) we review the architecture-variant GANs in the literature, (4) we review the loss-variant GANs in the literature, (5) we introduce some GAN-based applications mainly in the area of computer vision, (6) we summarize the GAN-variants in this study and illustrate their differences and relationships and also discuss several avenues for future research regarding GANs, and (7) we conclude this review and preview likely future research work in the area.

Many GAN-variants have been proposed in the literature to improve performance. These can be divided into two types: (1) architecture-variants. The first proposed GAN used fully connected neural networks [45] so specific types of architecture may be beneficial for specific applications, e.g., convolutional neural networks (CNNs) for images and recurrent neural networks (RNNs) for

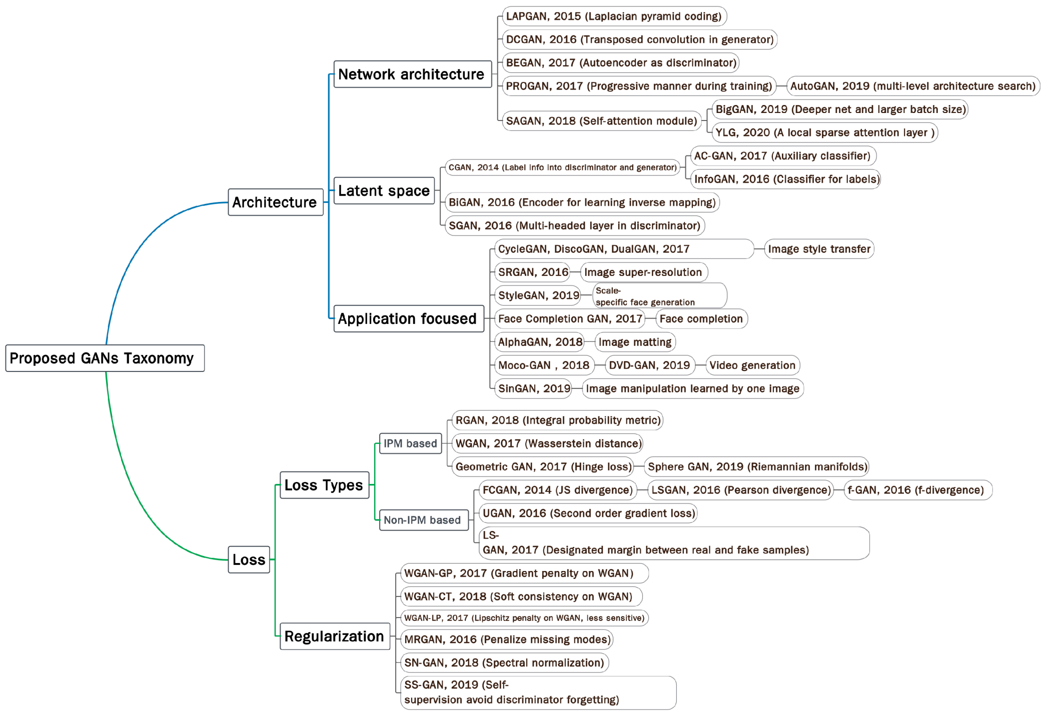

Proposed GANs Taxonomy
Architecture
Network architecture
LAPGAN, 2015 (Laplacian pyramid coding)
DCGAN, 2016 (Transposed convolution in generator)
BEGAN, 2017 (Autoencoder as discriminator)
PROGAN, 2017 (Progressive manner during training)
AutoGAN, 2019 (multi-level architecture search)
SAGAN, 2018 (Self-attention module)
BigGAN, 2019 (Deeper net and larger batch size)
YLG, 2020 (A local sparse attention layer)
Latent space
CGAN, 2014 (Label info into discriminator and generator)
AC-GAN, 2017 (Auxiliary classifier)
InfoGAN, 2016 (Classifier for labels)
BiGAN, 2016 (Encoder for learning inverse mapping)
SGAN, 2016 (Multi-headed layer in discriminator)
Application focused
CycleGAN, DiscoGAN, DualGAN, 2017
Image style transfer
SRGAN, 2016
Image super-resolution
StyleGAN, 2019
Scale-specific face generation
Face Completion GAN, 2017
Face completion
AlphaGAN, 2018
Image matting
Moco-GAN , 2018
DVD-GAN, 2019
Video generation
SinGAN, 2019
Image manipulation learned by one image
Loss
Loss Types
IPM based
RGAN, 2018 (Integral probability metric)
WGAN, 2017 (Wasserstein distance)
Geometric GAN, 2017 (Hinge loss)
Sphere GAN, 2019 (Riemannian manifolds)
Non-IPM based
FCGAN, 2014 (JS divergence)
LSGAN, 2016 (Pearson divergence)
f-GAN, 2016 (f-divergence)
UGAN, 2016 (Second order gradient loss)
LS-GAN, 2017 (Designated margin between real and fake samples)
Regularization
WGAN-GP, 2017 (Gradient penalty on WGAN)
WGAN-CT, 2018 (Soft consistency on WGAN)
WGAN-LP, 2017 (Lipschitz penalty on WGAN, less sensitive)
MRGAN, 2016 (Penalize missing modes)
SN-GAN, 2018 (Spectral normalization)
SS-GAN, 2019 (Self-supervision avoid discriminator forgetting)

Fig. 1. The proposed taxonomy of the recent GANs.

time-series data; and (2) loss-variants. Here different variations of the loss function (Equation (1)) are explored to enable more stable learning of G.

Figure 1 illustrates our proposed taxonomy for the representative GANs presented in the literature from 2014 to 2020. We divide current GANs into two main variants, i.e., the architecture-variant and the loss-variant. In the architecture-variant, we summarize three categories, which are network architecture, latent space, and application-focus. The network architecture category refers to improvement or modification made on the overall GAN architecture, e.g., the progressive mechanism deployed in Progressive GAN (PROGAN) [64]. The latent space category indicates that the architecture modification is made based on different representations of the latent space, e.g., Conditional GAN (CGAN) [99] involves providing label information to both the generator and the discriminator. The last category, application-focused, refers to modifications made according to different applications, e.g., CycleGAN [153] has a specific architecture that deals with image style transfer. In terms of the loss-variants, we divide this into two categories, loss types and regularization. Loss types refers to different loss functions that can be optimized for GANs and regularization refers to additional penalization designed into the loss function or any type of normalization operation made to the network. More specifically, we divide the loss function into *integral probability metric* (IPM) [101] based and non-IPM based. In IPM-based GANs, the **discriminator** is constrained to a specific class of function [60], e.g., the discriminator in Wasserstein GAN (WGAN) is constrained to *1-Lipschitz*. The discriminator in non-IPM based GANs does not have such constraints.

2 RELATED REVIEWS

There have been previous GANs review papers, for example, in terms of reviewing GANs performance [71]. That work focuses on experimental validation across different types of GANs benchmarking on the Large-scale Scene Understanding (LSUN)-BEDROOM [145], CELEBA-HQ-128 [84], and the CIFAR10 [70] image datasets. The results suggest that the original GAN [45] with spectral normalization [144] is a good starting choice when applying GANs to a new dataset. A limitation of that review is that the benchmark datasets do not consider diversity in a significant way. Thus the benchmark results tend to focus more on evaluation of the image quality, which may ignore GANs efficacy in producing diverse images. Other work [51] surveys different GANs architectures and their evaluation metrics. A further comparison on different architecture-variants' performance, applications, complexity and so on needs to be explored. Other papers [26, 52, 131] focus on investigation of the newest development treads and the applications of GANs. They compare GAN-variants through the lens of different application targets.

Comparing our review to other existing review articles we emphasize an introduction to GAN-variants based on their performance including their ability to produce high quality and diverse images, stability of training, and their ability to handle the vanishing gradient problem. We approach this exposition through taking a perspective based on architecture and loss function considerations. This perspective is important, because it covers fundamental challenges for GANs and it will help researchers on how best to choose an architecture and loss function for their GAN requirements and specific application. It also gives a snapshot of how researchers to date have dealt with those problems and will thus provide new researchers with a starting point for their own study. Our literature search strategy and the results of this search are presented in Supplementary Materials. A detail of searched papers are listed at this link: https://github.com/sheqi/GAN_Review/blob/master/GAN_CV.csv.

In summary, the contributions of this review are threefold:

- We focus on GANs by addressing three important problems: (1) high-quality image generation; (2) diverse image generation; and (3) stabilizing training.
- We propose a useful GAN taxonomy and contextualize recent GANs through variations in (1) architecture of generators and discriminators, e.g., network architecture, latent space, and application driven design, and (2) the objective function for training, e.g., loss design in IPM based and non-IPM based methods, regularization approaches. Compared to other reviews on GANs, this review provides a unique view of different GAN variants.
- We also provide a comparison and analysis in terms of pros and cons across the GAN-variants presented in this article.

3 GENERATIVE ADVERSARIAL NETWORKS

A typical GAN comprises two components, one of which is a discriminator (D) distinguishing between real images and generated images while the other one is a generator (G) creating images to fool the discriminator. Given a distribution $\mathbf{z} \sim p_{\mathbf{z}}$, G defines a probability distribution p_g as the distribution of the samples $G(\mathbf{z})$. The objective of a GAN is to learn the generator's distribution p_g that approximates the real data distribution p_r. Optimization of a GAN is performed with respect to a joint loss function for D and G

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right]. \tag{1}$$

GANs, as a member of the deep generative model (DGM) family, have attracted exponentially growing interest in the deep learning community because of some advantages comparing to the traditional DGMs: (1) GANs are able to produce better output than other DGMs. Compared to the most well-known DGMs, variational autoencoder (VAE), GANs are able to produce any type of probability density while VAE is unable to generate sharp images [44]. (2) The GAN framework can train any type of generator network. Other DGMs may have pre-requirements for the generator,

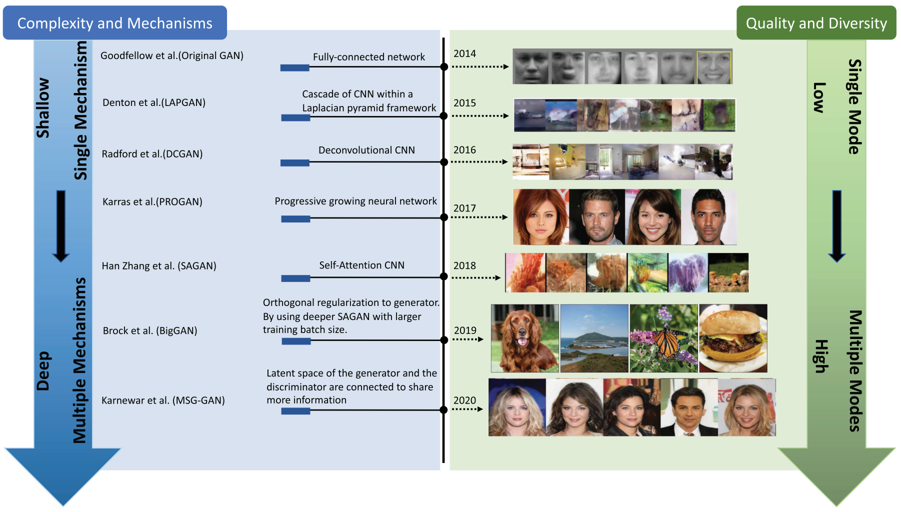

Complexity and Mechanisms
Quality and Diversity
Shallow
Single Mechanism
Deep
Multiple Mechanisms
Single Mode
Low
Multiple Modes
High
Goodfellow et al.(Original GAN)
Fully-connected network
2014
Denton et al.(LAPGAN)
Cascade of CNN within a Laplacian pyramid framework
2015
Radford et al.(DCGAN)
Deconvolutional CNN
2016
Karras et al.(PROGAN)
Progressive growing neural network
2017
Han Zhang et al. (SAGAN)
Self-Attention CNN
2018
Brock et al. (BigGAN)
Orthogonal regularization to generator. By using deeper SAGAN with larger training batch size.
2019
Karnewar et al. (MSG-GAN)
Latent space of the generator and the discriminator are connected to share more information
2020

Fig. 2. Timeline of architecture-variant GANs presented in this article. Complexity in blue stream refers to size of the architecture and computational cost such as batch size. Mechanisms refer to the number of types of operations (e.g., convolution, deconvolution, self-attention) used in the architecture (e.g., FCGAN uses fully connected layers for both discriminator and generator. In this case, the value for mechanisms is 1).

e.g., the output layer of generator is Gaussian [32, 44, 67]. (3) There is no restriction on the size of the latent variable. These advantages have led GANs to achieve the state-of-the-art performance on producing synthetic data especially for image data.

4 ARCHITECTURE-VARIANT GANS

There are many types of architecture-variants proposed in the literature (see Figure 2) [11, 64, 113, 150, 153]. Architecture-variant GANs are mainly proposed for the purpose of different applications, e.g., image-to-image transfer [153], image super resolution [74], image completion [55], and text-to-image generation [114]. In this section, we provide a review on architecture-variants that helps improve the performance for GANs under the three aspects mentioned before, namely that of improving image diversity, improving image quality and stabilizing training. A review of architecture-variants in terms of different applications can be found here [26, 51].

4.1 Fully connected GAN

The original Energy-based GAN (EBGAN) paper [45] uses fully connected neural networks for both generator and discriminator. This architecture-variant was applied for some simple image datasets, i.e., MNIST [73], CIFAR-10 [70], and the Toronto face dataset. The authors suggest k steps for optimizing D and one step for optimizing G due to overfitting of the discriminator if the completion of optimizing D is done in the inner loop of training. In practice, Equation (1) may induce the vanishing gradient issue for optimizing the generator and the authors instead maximize $\log D(G(z))$ for training G. This modification equivalently optimizes the reverse Kullback-Leibler (KL) divergence between p_g and p_r for G, which also causes the asymmetry issue. We will revisit this in detail in Section 5. For the architecture setting, maxout [41] was deployed for the discriminator while a mixture of ReLU and sigmoid activations were used for the generator. It does not demonstrate good generalization performance for more complex image types.

4.2 Semi-supervised GAN

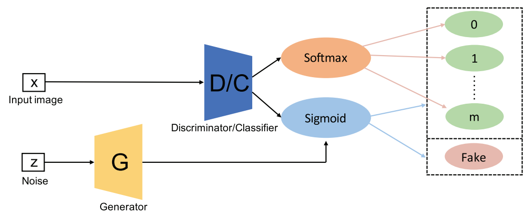
Semi-supervised GAN (SGAN) is proposed in the context of semi-supervised learning [105]. Semi-supervised learning is a promising research field between supervised learning and unsupervised learning. Unlike supervised learning, in which we need a label for every sample, and unsupervised learning, in which no labels are provided, semi-supervised learning has labels for a small subset of examples. Compared to fully connected GAN (FCGAN), SGAN's discriminator is multi-headed, i.e., it has softmax and sigmoid for classifying the real data and distinguishing between real and fake samples, respectively. The authors trained SGAN on the MNIST dataset. Results show that both the discriminator and the generator in SGAN are improved compared to the original GAN. We think the architecture of the multi-headed discriminator is relatively simple that limits the diversity of the model, i.e., the experiment is only carried out on the the MNIST dataset. A more complex architecture for the discriminator may improve the performance of the model.

4.3 Bidirectional GAN

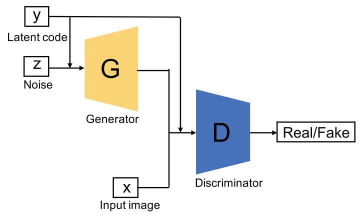
Traditional GANs have no means of learning the inverse mapping, i.e., projecting data back into the latent space. Bidirectional GAN (BiGAN) is designed for this purpose [35]. As seen in Figure 3(c), the overall architecture consists of an encoder (E), a generator (G), and a discriminator (D). E encodes real sample data into $E(\mathbf{x})$ while G decodes $\mathbf{z}$ into $G(\mathbf{z})$. As a result, D aims to evaluate the difference between each pair of $(E(\mathbf{x}), \mathbf{x})$ and $(G(\mathbf{z}), \mathbf{z})$. As E and G do not communicate directly, i.e., E never sees $G(\mathbf{z})$ and G never sees $E(\mathbf{x})$. The authors prove that the encoder and decoder must learn to invert one another to fool the discriminator in the original paper. It would be interesting to see if such a model is able to deal with adversarial examples in future work. BiGAN was trained on the MNIST and the ImageNet datasets. Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ is used for optimization. The batch size is 128 and the weight decay as 2.5×10^{-5} is applied for all parameters. Batch normalization is also deployed.

4.4 Conditional GAN

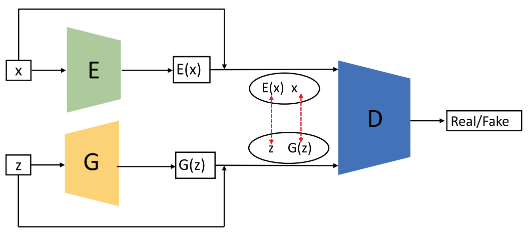
The CGAN has the innovation of conditioning on both the discriminator and the generator by feeding each with class labels [99]. As seen in Figure 3(b), CGAN feeds the extra information $\mathbf{y}$ ($\mathbf{y}$ that can be class label or other modal data) to both discriminator and generator. It should be noted that $\mathbf{y}$ is normally encoded inside the generator and discriminator before being concatenated with the encoded $\mathbf{z}$ and encoded $\mathbf{x}$. For example, in the MNIST experiment in the original work, both $\mathbf{z}$ and $\mathbf{y}$ are mapped to hidden layers with layer sizes 200 and 1,000, respectively, before being combined with each other (the combined layer dimensionality is $200 + 1000 = 1200$) in the generator. By doing this, CGAN enhances the discriminative ability of the discriminator. The loss function of CGAN is slightly different from the FCGAN as seen in Equation (2), in which $\mathbf{x}$ and $\mathbf{y}$ are conditioned by $\mathbf{z}$. Benefiting from the extra encoded y information, CGAN is not only able to handle unimodal image datasets but also multimodal datasets such as Flickr that contains labeled image data with their associated user-generated metadata, i.e., in particular user-tags, which brings GANs over to the area of multimodal data generation. The authors experimented with the CGAN on the MNIST and Yahoo Flickr Creative Common 100M (YFCC 100M). For the MNIST dataset, the model was trained using stochastic gradient descent (SGD) with mini-batch size of 100 and an initial learning rate of 0.1 that was exponentially decreased down to 1×10^{-6} with the decay factor set as 1.00004. Dropout was utilized with probability of 0.5 to both generator and discriminator. Momentum was used with an initial value of 0.5 and finally was increased up to 0.7. Class labels were encoded as one-hot vectors and fed to both G and D. In terms of the YFCC 100M experiment, training hyperparameters are the same as the set-up in the MNIST experiment. Even though CGAN



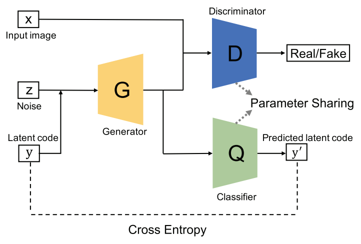
(a) SGAN architecture. The D is multi-headed, which is able to classify the real data and distinguish between real and fake samples.



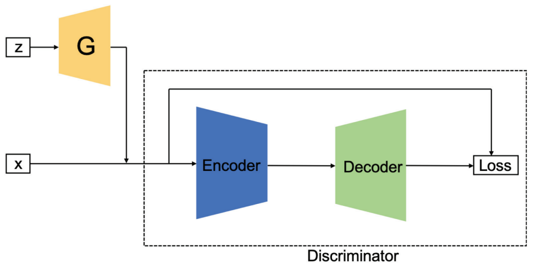
(b) CGAN architecture. Extra information (y can be class labels or other modal data) is encoded to both D and G.



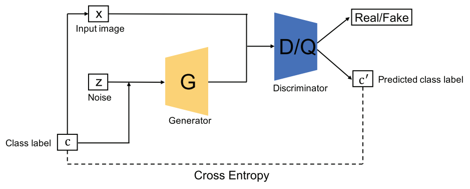
(c) BiGAN architecture. Inverse mapping is utilized for projecting data back into the latent space.



(d) InfoGAN architecture. Beyond the CGAN, another classifier Q is introduced to predict class labels.



(e) BEGAN architecture. $\mathbf{z}$ is the latent variable for G and $\mathbf{x}$ is input image. BEGAN deploys an autoencoder architecture for the discriminator. Loss is calculated using L_1 or L_2 norm at pixel level.



(f) AC-GAN architecture. $\mathbf{x}$ is the real data from class c, $\mathbf{z}$ is the input noise and $c^{'}$ is the output of auxiliary classifier.

Fig. 3. The architectures of SGAN, CGAN, BiGAN, InfoGAN, BEGAN, and AC-GAN studied in this work.

enhances the discriminative ability of the discriminator by introducing encoded labels, some of the encoded labels still loose connection with images,

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_z} \log\left[1 - D(G(\mathbf{z}|\mathbf{y}))\right]. \tag{2}$$

4.5 InfoGAN

InfoGAN is proposed as a step beyond the CGAN [20], which learns the interpretable representations in an unsupervised manner by maximizing the mutual information between conditional variables and the generative data. To achieve this, InfoGAN introduces another classifier Q (see Figure 3(d)) to predict the $\mathbf{y}$ given by $G(\mathbf{z}|\mathbf{y})$. The combination of G and Q here can be understood as an autoencoder, in which we aim to find the embedding $(G(\mathbf{z}|\mathbf{y}))$ minimizing the cross entropy between $\mathbf{y}$ and $\mathbf{y}'$. However, D performs the same job as with the FCGAN, which distinguishes samples generated from G or from real data. To reduuce computational cost, Q and D share all convolutional layers except the last fully connected layer, which enables the discriminator to have the capability of distinguishing real and fake samples and recover the information $\mathbf{y}$. This can improve the discriminative ability for InfoGAN compared to the original GAN architecture. The loss used in InfoGAN is a regularization of CGAN's loss

$$\min_{G} \max_{D} V(D, G) - \lambda I(G, Q), \quad \lambda > 0, \tag{3}$$

where $V(D, G)$ is the objective of CGAN except that the discriminator does not take $\mathbf{y}$ as input and $I(\cdot)$ is the mutual information. The authors experimented with InfoGAN using the MNIST, three-dimensional (3D) face images [109], 3D chair images [6], SVHN and CelebA. All datasets shared the same training setting, in which the Adam optimizer was used and batch normalization was applied. Leaky ReLU with a 0.1 leaky rate was applied to the discriminator while ReLU was used for the generator. A learning rate of 2×10^{-4} was set for D while 1×10^{-3} is set for G. λ was set as 1. Here we think the diversity of the model is very limited due to the fact that the parameters for D and Q are shared with each other except the last layer. A more complex set-up for Q could be usefully investigated.

4.6 Auxiliary Classifier GAN

Auxiliary Classifier GAN (AC-GAN) [106] is very similar to CGAN and InfoGAN. It contains an auxiliary classifier in the architecture as seen in Figure 3(f). In AC-GAN, each generated sample has a corresponding class label c in addition to $\mathbf{z}$. It should be noted that the difference between AC-GAN and the previous two architecture-variants (CGAN and InfoGAN) is the additional information here, which only refers to the class label while the previous two can be other domain data. Thus we use c and c' in AC-GAN to explicitly flag this difference from the previous two variants. The discriminator in AC-GAN consists of a discriminator D (distinguishes real and fake samples) and a classifier Q (classifies real and fake samples). Similarly to InfoGAN, the discriminator and classifier share all weights except the last layer. The loss function of AC-GAN can be constructed by considering the discriminator and classifier, which can be stated as

$$L_S = \mathbb{E}_{\mathbf{x} \sim p_r} \log[D(\mathbf{x}|\mathbf{c})] + \mathbb{E}_{\mathbf{z} \sim p_z} \log\left[1 - D(G(\mathbf{z}|\mathbf{c}))\right], L_C = \mathbb{E}_{\mathbf{x} \sim p_r} \log[Q(\mathbf{x}|\mathbf{c})] + \mathbb{E}_{\mathbf{z} \sim p_z} \log\left[Q(G(\mathbf{z}|\mathbf{c}))\right], \tag{4}$$

where D is trained by maximizing $L_S + L_C$ and G is trained on maximizing $L_C - L_S$. The authors trained AC-GAN on the CIFAR-10 and ImageNet datasets for all 1,000 classes. For both CIFAR-10 and ImageNet, the model was trained by using Adam with $\alpha = 2 \times 10^{-4}$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$ for D, G, and Q. The mini-batch size was set to 100. Details of model performance and relate experiments can be found in the original paper [106]. AC-GAN has improved visual quality for the generated images and has high model diversity. However, these improvements depend on large-scale labeled datasets, which may pose challenges in some real-world applications. A combination of AC-GAN and unsupervised or self-supervised approaches should be further investigated. We have also introduced a type of GAN, label-noise Robust GANs (rGANs) in Section 4.13, which deals with the noisy label issue.

4.7 Laplacian Pyramid of Adversarial Networks

Laplacian Pyramid of Adversarial Networks (LAPGAN) is proposed for the production of higher-resolution images from lower resolution input GAN [31]. The Laplacian pyramid [16] is an image coding approach, which uses local operators of many scales but identical shape as the basic functions. LAPGAN utilizes a cascade of CNNs within a Laplacian pyramid framework [16] to produce high quality images, which are demonstrated in Figure 4 (from right to left). Rather than using a deconvolutional process (i.e., used in Deep Convolutional GAN (DCGAN)) to up-sample the kernel output of the previous layer, LAPGAN uses Laplacian pyramids to up-sample the image. First,

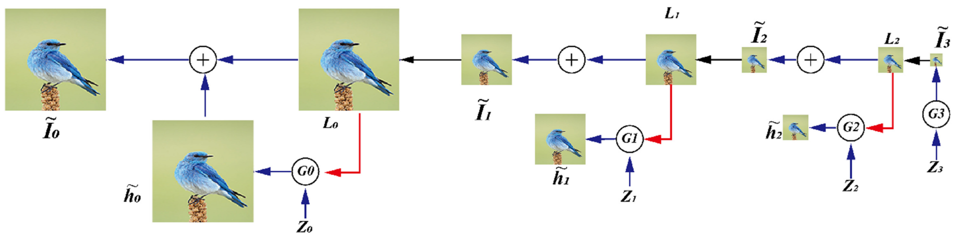


Fig. 4. Up-sampling process of generator in LAPGAN (from right to left). The up-sampling process is marked using green arrow and a conditioning process via a conditional GAN (CGAN) [99] is marked using the orange arrow. The process initially uses G_3 to generate image $\widetilde{I}_3$ and then up-samples the image $\widetilde{I}_3$ to l_2. Together with another noise z_2, G_2 generates a difference image $\widetilde{h}_2$ and adds $\widetilde{h}_2$ to l_2, which produces the generated image $\widetilde{I}_2$. The rest can be done in the same manner. LAPGAN contains 3 generators in this work to up-sample the image. Figure is regenerated from Ref. [31].

LAPGAN use the first generator to produce a very small image, which can alleviate the instability issue for the generator, and this image is then up-sampled through use of a Laplacian pyramid. Then the up-sampled image is fed to the next generator for producing the image difference and the summation of the image difference. The input image will be the generated image. It can be seen that only the G_3 in Figure 4 is used for generating images but the dimension is very small, which encourages stabilizing training. For larger images, the generator is used to generate the image difference, which is much less complex than the same sized raw images. This structure facilitates stabilizing training and high resolution modeling. CIFAR10 (28×28 pixel), STL (96×96 pixel), and LSUN (64×64 pixel) were used for generation. The Laplacian pyramid up-sampling processes for each dataset are $8 \rightarrow 14 \rightarrow 28$ (CIFAR10), $8 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 96$ (STL) and $4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 64$ (LSUN). The discriminator used three hidden layers and a sigmoid output, while the generator used a five-layer CNN with ReLU and batch normalization. A linear ouput layer was utilized. SGD with an initial learning rate of 0.02, decreased by a factor of (4×10^{-5}) at every epoch, was deployed in the experiment. Momentum started at 0.5, increasing by 0.0008 at each epoch up to a maximum of 0.8. The current structure includes multiple generators for generating images and the connections between these generators have not been established. In Section 4.10, we introduce a more advanced strategy, which trains the model in a progressive fashion, i.e., PROGAN.

4.8 Deep Convolutional GAN

DCGAN is the first work that applied a deconvolutional neural network architecture for G [113]. Deconvolution is proposed to visualize the features for a CNN and has shown good performance for CNNs visualization [148]. DCGAN deploys the spatial up-sampling ability of the deconvolution operation for G, which enables the generation of higher resolution images using GANs. There are some critical modifications in the architecture of DCGAN compared to original FCGAN, which benefits high-resolution modeling and stabilizing training. First, DCGAN replaces any pooling layers with strided convolutions for discriminator and fractional-strided convolutions for generator. Second, batch normalization is used for both the discriminator and the generator, which helps locate the generated samples and the real samples centering on zero, i.e., *similar statistics for generated samples and real samples.* Third, the ReLU activation is used in the generator for all layers except the output, which uses Tanh, while Leaky ReLU activation is used in the discriminator for all layers. In this case, the Leaky ReLU activation will prevent the network stagnating in a "dying state" situation (e.g., inputs smaller than 0 in the ReLU layers) as the generator receives gradients from the discriminator. DCGANs are trained on LSUN [145], ImageNet [30] and the customized-assembled face datasets. All models were trained using SGD with a mini-batch size of 128. All weights were initialized from a zero-centered Normal distribution with standard deviation of 0.02. An Adam optimizer was utilized with a learning rate of 0.0002 and momentum term of 0.5. The slope of Leaky ReLU was set to 0.2 for all models. Models were trained by using 64×64 pixel image. DCGAN is a very important milestone in the history of GANs and the deconvolution idea becomes a mainstay for the main architecture used in GAN generators. Due to the limit of the model capacity and the optimization used in DCGAN, it is only successful on low-resolution and less diverse images.

4.9 Boundary Equilibrium GAN

Boundary Equilibrium GAN (BEGAN) uses an autoencoder architecture for the discriminator that was first proposed in EBGAN [151]. As seen in Figure 3(e), the autoencoder loss can be generated for G and D, respectively. When training the autoencoder (D), the objective is to maximize the reconstruction loss of real images and maximizes the reconstruction loss for generated images, i.e., minimize $\mathbb{E}\left[\mathcal{L}(x)\right] - \mathbb{E}\left[\mathcal{L}(G(z))\right]$. When training the G, the objective is to minimize $\mathbb{E}\left[\mathcal{L}(G(z))\right]$. By introducing the autoencoder, the authors have proved the optimization of the reconstruction loss above is equivalent to the Wasserstein distance. The authors also propose the use of a hyperparameter $\gamma = \frac{\mathbb{E}[\mathcal{L}(G(z))]}{\mathbb{E}[\mathcal{L}(x)]}, \gamma \in [0,1]$ to control the balance between the generator and discriminant losses, which allows a balancing of the effort allocated to the generator and the discriminator, i.e., control a variety of generated faces. The experiment in the original paper shows that smaller γ makes G generate faces that look overly uniform. The variety of faces increases with a larger value of γ but this also introduces artifacts. The overall loss function is summarized in Equation (5),

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t \mathcal{L}(G(z_D)), & \text{for updating } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G_{z_G}), & \text{for updating } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G)), & \text{for each training iteration t} \end{cases}, \tag{5}$$

where $\mathcal{L}(\cdot)$ represents the autoencoder reconstruction loss (L_2), $k_t \in [0,1]$ is a variable that controls how much emphasis of $\mathcal{L}(G(z))$ is penalized for the loss. k is initialized as 0 and is controlled by λ_k (λ_k can be interpreted as a learning rate for k, which is set as 1×10^{-3} in the original paper).

Compared to traditional optimization, the BEGAN matches the autoencoder loss distributions using a loss derived from the Wasserstein distance instead of matching data distributions directly. This modification helps G to generate easy-to-reconstruct data for the autoencoder at the beginning, because the generated data are close to 0 and the real data distribution has not been learned accurately yet, which prevents D easily winning G at the early training stage. For encoder and decoder, exponential linear units were applied at their outputs. The model was trained on CelebA dataset using 128×128 pixel images. Separate Adam optimizers with initial learning rates of 1×10^{-4}, decaying by a factor of 2 when the measure of convergence stalls, were used for D and G. The batch size was set as 16 in the original work.

4.10 PROGAN

PROGAN involves progressive steps toward the expansion of the network architecture [64]. This architecture uses the idea of progressive neural networks first proposed in Reference [116]. This technology does not suffer from forgetting and is able to deploy prior knowledge via lateral connections to previously learned features. Consequently it is widely applied for learning complex task sequences. Figure 5 demonstrates the training process for PROGAN. Training starts with low resolution 4×4 pixels image. Both G and D start to grow with the training progressing.

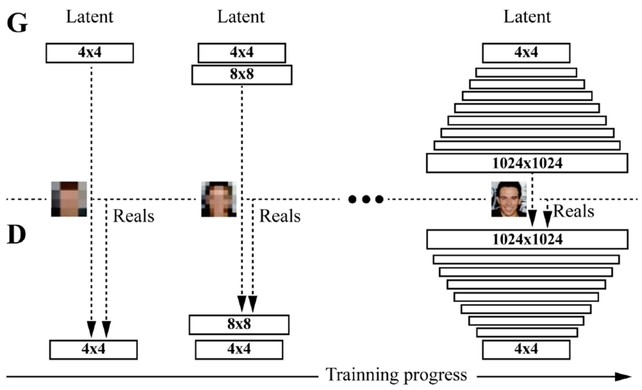


Fig. 5. Progressive growing step for PROGAN during the training process. Training starts with 4×4 pixels image resolution. With the training step growing, layers are incrementally added to G and D, which increases the resolution for the generated images. All existing layers are trainable throughout the training stage. Figure is regenerated from Reference [64].

Importantly, all variables remain trainable throughout this growing process. This progressive training strategy enables substantially more stable learning for both networks. By increasing the resolution little by little, the networks are continuously asked a much simpler question compared to the end goal of discovering a mapping from latent vectors. All current state-of-the-art GANs employ this type of training strategy and it has resulted in impressive, plausible images [13, 64, 65]. The authors start training the PROGAN with 4×4 pixel images and incrementally add the doubled-sized layers to G and D as seen in Figure 5, in which the new layers are faded smoothly. The multi-scaled training images are produced by using Laplacian pyramid representations [16], i.e., similar to LAPGAN. Models were trained on CIFAR10 (32×32 pixel images), LSUN (256×256 pixel images), and CelebA-HQ ($1{,}024 \times 1{,}024$ pixel images). Leaky ReLU with leakness 0.2 were used for all layers of both D and G except for the last layer (used linear activation). Only pixelwise normalization of the feature vectors after each Conv 3×3 layer in the generator was deployed, i.e., no batch normalization, layer normalization, or weight normalization in either network. The Adam optimizer with $\alpha = 1 \times 10^{-3}$, $\beta_1 = 0$, $\beta_2 = 0.99$, and $\epsilon = 1 \times 10^{-8}$ was utilized for training D and G. The mini-batch size was gradually decreased with increasing image pixel for saving on memory, i.e., batch size 16 for 4×4 to 8×8, $256 \times 256 \rightarrow 14$, $512 \times 512 \rightarrow 6$, and $1{,}024 \times 1{,}024 \rightarrow 3$. The WGAN-GP [136] loss was used for optimizing both D and G.

4.11 Self-attention GAN

Traditional CNNs can only capture local spatial information and the receptive field may not cover enough structure, which causes CNN-based GANs to have difficulty in learning multi-class image datasets (e.g., ImageNet) and the key components in generated images may shift, e.g., the nose in a face-generated image may not appear in the correct position. Self-attention mechanism have been proposed to ensure a large receptive field without sacrificing computational efficiency for CNNs [129]. Self-attention GAN (SAGAN) deploys a self-attention mechanism in the design of the discriminator and generator architectures for GANs [149] (see Figure 6). Benefiting from the self-attention mechanism, SAGAN is able to learn global, long-range dependencies for generating images. It has achieved great performance on multi-class image generation based on the ImageNet datasets. The authors trained SAGAN on the ImageNet dataset (128×128 pixel images). Spectral normalization [100] was applied for both D and G. Conditional batch normalization was used

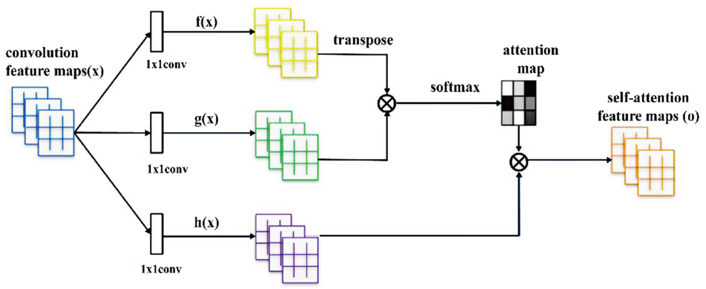


Fig. 6. Self-attention mechanism architecture proposed in the article. f, g, and h correspond with query, key and value in the self-attention mechanism. The attention map indicates the long-range spatial dependencies. The $\otimes$ denotes matrix multiplication. Figure is regenerated from Reference [149].

in the generator while batch projection was used in the discriminator. An Adam optimizer with $\beta_1 = 0$ and $\beta_2 = 0.9$ was used and the learning rate for the discriminator was 4×10^{-4} and for the generator was 1×10^{-4}. The authors also demonstrate that the deployment of a self-attention mechanism for both the discriminator and the generator at large feature maps is more effective, i.e., deployment of self-attention mechanism with a feature map with 32×32 size achieves the best performance using FID score and deployment of a self-attention mechanism with a feature map with 64×64 size achieves the best performance using Inception score, which indicates the self-attention mechanism is complementary to convolution for large feature maps. Thus the self-attention mechanism, we suggest, should be applied for large feature maps to improve the diversity for GANs.

4.12 BigGAN

BigGAN [13] has also achieved state-of-the-art performance on the ImageNet datasets. Its design is based on SAGAN and it has demonstrated that the performance can yield a scaling up of GAN training, i.e., an increase in the number of channels for each layer and an increase in the batch size. The authors train the model on ImageNet with 128×128, 256×256, and 512×512 resolutions. The training setting in this work follows the SAGAN, in which the learning rate was halved and train two D steps per G step. Different selections of latent variables z are explored and the authors state that Bernoulli $\{0, 1\}$ and Censored Normal $\max(\mathcal{N}(0, I), 0)$ work best without truncation. The truncation trick involves using a different distribution for the generator as latent space during training than during inference or image synthesis. In BigGAN, a Gaussian distribution is used during training, and a truncated Gaussian is used during inference. This truncation trick provides a tradeoff between image quality or fidelity and image variety. A more narrow sampling range results in better quality, whereas a larger sampling range results in more variety in sampled images. We summarize the following operations on BigGAN that make BigGAN scale-up the architecture. (1) *Self-attention module* and *Hinge loss*: The BigGAN uses a model architecture with attention modules from SAGAN and is trained via hinge loss, in which self-attention contributes to the model diversity and hinge loss enables stability of training. (2) *Class conditional information*: The class information is provided to the generator model via class-conditional batch normalization. (3) *Update discriminator more than generator*: The BigGAN slightly modifies this and updates the discriminator model twice before updating the generator model in each training iteration. (4) *Moving average of model weights*: Before images are generated for evaluation, the model weights are averaged across prior training iterations using a moving average. (5) Some operations on the network: *orthogonal weight initialization*, *larger batch size*, *skip-z connections* (skip connections from

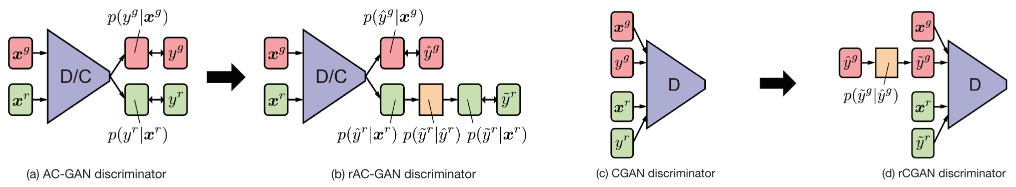


Fig. 7. Discriminators of (a) AC-GAN, (b) rAC-GAN, (c) CGAN, and (d) rCGAN. A noise transition model is denoted as an orange rectangle. Generators in rAC-GAN and rCGAN remain the same as in AC-GAN and CGAN. The figure is regenerated and reorganized from Reference [62].

the latent to multiple layers), and *shared embeddings*, i.e., the authors show these operations are all able to help improve the performance. The authors also characterize the analysis of instability specific to such large scale. More details can be found in the original paper.

4.13 Label-noise rGANs

We have discussed CGAN in Section 4.4 and AC-GAN in Section 4.6, respectively, in which both have the ability to learn the disentangled representation and improve the discriminative ability of GANs. However, large-scale labeled datasets are required for training models, which poses some challenges in real-world scenarios. Kaneko et al. [62] propose a family of GANs named label-noise rGANs, which incorporates a noise transition model that is able to learn a clean label conditional generative distribution even when provided training labels that are noisy. Two variants are discussed, which are an extension for AC-GAN (rAC-GAN) and an extension for CGAN (rCGAN) as seen in Figure 7. The core part of rGANs is a noise transition module $p(\tilde{y}|\hat{y})$ ($\tilde{y}$ is the noisy label and $\hat{y}$ is the clean label) introduced to the discriminator, in which $p(\tilde{y} = j|\hat{y} = i) = T_{(i,j)}$ as T is a noise transition matrix $T_{(i,j)} \in [0,1]^{(c\times c)}$ ($\sum_i T_{i,j} = 1$, c is the number of classes). The authors trained rGANs on CIFAR-10 and CIFAR-100. The authors demonstrate that rAC-GAN and rCGAN perform better than the original architectures on CIFAR-10 and also exhibit robustness to label noise. However, in CIFAR-100, when high noise is introduced to labels, their performance drops. We think such a framework is still somewhat limited when encountering more complicated datasets, e.g., ImageNet and it needs to be investigated further in the future.

4.14 Your Local GAN

This work [29] introduces a new local sparse attention layer that preserves the two-dimensional geometry and locality. To show the applicability of the idea, they replace the dense attention layer of SAGAN [129] with a new construction. The key innovations are (1) the attention patterns are well supported by the information theoretic framework of Information Flow Graphs; (2) Your Local GAN (YLG)-SAGAN is introduced and achieves superior performance with reducing the training time by approximately 40%; (3) they have made the natural inversion process of performing gradient descent on the loss work for bigger models rather than previous work on small GANs. One specific trick the author utilizes is called Enumerate, Shift, Apply (ESA). They modify one dimensional sparsifications to become aware of two-dimensional locality via *enumerating* pixels of the image based on their Manhattan distance from the pixel at location (0, 0) (breaking ties using row priority), *shifting* the indices of any given one-dimensional sparsification to match the Manhattan distance enumeration instead of the reshape enumeration, and *applying* this new one dimensional sparsification pattern, that respects two-dimensional locality, to the one-dimensional reshaped version of the image.

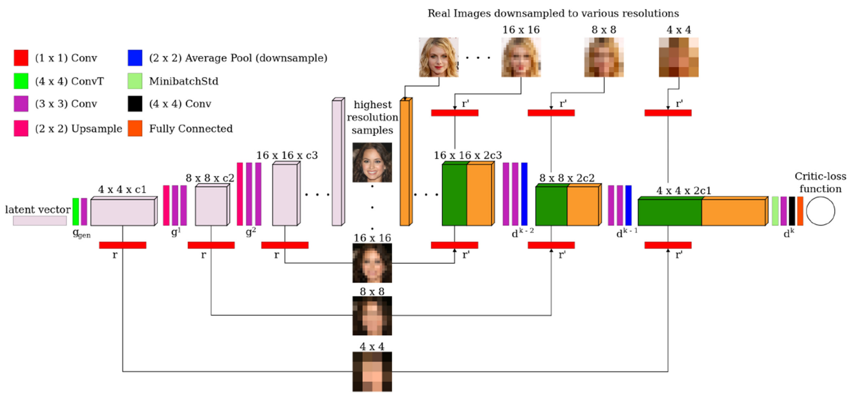


Fig. 8. Architecture of MSG-GAN. Progressive training similar to PROGAN is deployed here. MSG-GAN includes connections from the intermediate layers of the generator to the intermediate layers of the discriminator. Multi-scale images are sent to the discriminator, which are concatenated with the corresponding main path, i.e., green in the discriminator represents images produced by the generator while orange represents the original features in the main path of the discriminator. Figure is regenerated and reorganized from Reference [63].

However, we think two conflicting objectives exist in this work. On the one hand, this method intends to make the networks as sparse as possible for computational and statistical efficiency, on the other hand they still need to support good and full information flow.

4.15 AutoGAN

AutoGAN [43] introduces the neural architecture search (NAS) algorithm to generative adversarial networks (GANs). The search space of the *generator* architectural variations in are guided via an RNN together with parameter sharing and dynamic-resetting to accelerate the process. They use the Inception score as the reward, and introduce a multi-level search strategy to perform NAS in a progressive way. The authors use hinge loss for training the shared GAN, following the training setting of spectral normalization GAN (SN-GAN).

The whole pipeline is insightful but also poses novel challenges w.r.t. the marriage of NAS and GANs. NAS remains to be optimized for standard classification problems, let alone the unstable training problems brought by GANs. Although in the article AutoGAN shows promising results with NAS for GAN architecture, which is quite unique compared to the manual design GAN architectures introduced above. We think it still has two critical issues yet to be solved:

- The search space for the generator is limited and the search strategy for the discriminator is not discussed.
- AutoGAN has not yet been tested on high-resolution image generation datasets. Thus, we do not have an intuitive estimation of the applicability of this methods. The current image generation task is preliminary.

4.16 MSG-GAN

It is well known that GANs are very difficult to adapt to different datasets. Karnewar et al. argue that one of the reasons for this is gradients passing from the discriminator to the generator become uninformative when there is not enough overlap in the supports of the real and fake distributions. They propose MSG-GAN [63] as a means to handle such problems. As seen in Figure 8, latent space of the generator and the discriminator are connected so as to share more information between the

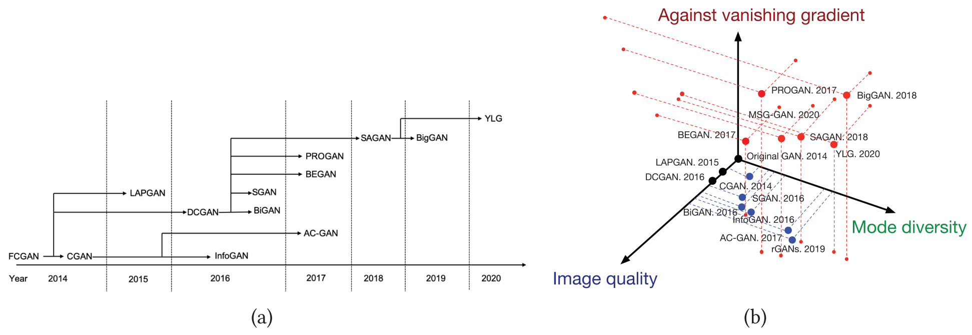


Fig. 9. (a) An overview of the footprint for architecture-variant GANs discussed in this section. (b) Summary of recent architecture-variant GANs for solving the three challenges highlighted in this article (by our estimation and quantitative results can be referred to Table 2 in Section 7). The challenges are represented by the three orthogonal axes. A larger value for each axis indicates better performance. Red points indicate GAN-variants that cover all three challenges, blue points cover two, and black points cover only one challenge. Quantitative results can be found in Table 2 in Section 7.

generator and the discriminator. More specifically, activations in each transpose convolutional step (three steps in Figure 8) in G are mapped to an image at different scales by operation r, i.e., 1×1 convolution in this case. Similarly, the mapped image is then encoded by r' to activations, which is concatenated with activations encoded by a real image. This connection enables more information sharing between D and G and experimental results demonstrate the benefits. The authors trained MSG-GAN on multiple datasets, i.e., CIFAR10, Oxford flowers, LSUN, Indian Celebs, CelebA-HQ ($1{,}024 \times 1{,}024$), and FFHQ ($1{,}024 \times 1{,}024$). The hyperparameter settings are almost identical for all datasets. Specifically, $\mathbf{z} \in \mathbb{R}^{512\times 1}$ is drawn from a standard normal distribution. The RMSprop with a learning rate of 0.003 was used for both D and G. WGAN-GP loss was used for training the network. Although MSG-GAN has achieved very good results on several image datasets, the ability of MSG-GAN to generate diverse images has not been tested yet and we found the results on CIFAR10 are not as good as for the other datasets. We guess this might be caused by the connection between G and D, which may constrain the diversity on G as activations from G and D. The diversity on images might cause inconsistent matched activations, which have a negative impact on the training. More development may solve this issue such as through the addition of a self-attention module.

4.17 Summary

We have provided an overview of architecture-variant GANs centered on how they can potentially improve performance with respect to the three key challenges of image quality, mode diversity and the unstable training problem. Figure 9(a) illustrates a footprint for architecture-variant GANs from 2014 to 2020 that discussed in this section. It can be seen that there are lots of interconnections in different GAN variants. Figure 9(b) illustrates relative performance with respect to the three challenges. We suggest that interested readers should consult the original articles to get deeper insights on the theory and the performance of each GAN variant. Here we give a quick recap on how architecture-variant GANs remedy are identified challenges (quantitative results are summarized in Table 2 in Section 7).

Image Quality One of the basic objectives of GANs is to produce realistic images with high image quality. The original GAN (FCGAN) was only applied to the MNIST, Toronto face and CIFAR-10 datasets because of the limited capacity of the architecture. DCGAN and LAPGAN introduced the deconvolutional and the up-sampling processes to the design. Both enable the model to produce higher resolution images. The remaining architecture variants (i.e., BEGAN, PROGAN, SAGAN, and BigGAN) all have some modifications on the loss function that we will address in a later section of this article and these are also beneficial in producing better image quality. Regarding the architecture component only, BEGAN uses an autoencoder architecture for the discriminator, which compares generated images and real images at the pixel level. This helps the generator produce easy-to-reconstruct data. PROGAN utilizes a deeper architecture and the model expands as the training progresses. This progressive training strategy improves the learning stability for discriminator and generator thus it is easier for the model to learn how to produce high resolution images. SAGAN mainly benefits from the spectral normalization, which we address in the next section. BigGAN demonstrates that high resolution image generation can benefit from a deeper model with larger batch sizes.

Vanishing Gradient Changing the loss function is the only way to remedy such a problem currently. Some architecture variants here avoid the vanishing gradient issue but only because they use different loss functions. We will explore this in the next section.

Mode Diversity This is the most challenging problem for GANs. It is very difficult for GANs to produce realistic diverse images such as natural images. In terms of architecture-variant GANs, only SAGAN and BigGAN address this issue explicitly. Benefiting from the self-attention mechanism, CNNs in SAGAN and BigGAN can process a large receptive field that overcomes the shifting components problem in generated images. This enables such types of GAN to produce diverse images.

5 LOSS-VARIANT GANS

Another design decision in GANs that significantly impacts performance is the choice of loss function in Equation (1). While the original GAN work [45] has already demonstrated global optimality and the convergence of GANs training, it still highlights the instability problem that can arise when training a GAN. The problem is caused by the global optimality criterion as stated in Reference [45]. Global optimality is achieved when an optimal D is reached for any G. So the optimal D is achieved when the derivative of D for the loss in Equation (1) equals 0. So we have

$$-\frac{p_r(\mathbf{x})}{D(\mathbf{x})} + \frac{p_g(\mathbf{x})}{1 - D(\mathbf{x})} = 0 \rightarrow D^*(\mathbf{x}) = \frac{p_r(\mathbf{x})}{p_r(\mathbf{x}) + p_g(\mathbf{x})}, \tag{6}$$

where $\mathbf{x}$ represents the real data and generated data, $D^*(\mathbf{x})$ is the optimal discriminator, $p_r(\mathbf{x})$ is the real data distribution and $p_g(\mathbf{x})$ is the generated data distribution. We have the optimal discriminator D so far and when we have the optimal D, the loss for G can be visualized by substituting $D^*(\mathbf{x})$ into Equation (1),

$$\mathcal{L}_G = \mathbb{E}_{\mathbf{x}\sim p_r} \log \frac{p_r(\mathbf{x})}{\frac{1}{2}\left[p_r(\mathbf{x}) + p_g(\mathbf{x})\right]} + \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g(\mathbf{x})}{\frac{1}{2}\left[p_r(\mathbf{x}) + p_g(\mathbf{x})\right]} - 2 \cdot \log 2. \tag{7}$$

Equation (7) demonstrates the loss function for a GAN when the discriminator is optimized and it is related to two important probability measurement metrics. One is KL divergence, which is defined as

$$KL(p_1 \| p_2) = \mathbb{E}_{\mathbf{x}\sim p_1} \log \frac{p_1}{p_2}, \tag{8}$$

and the other is Jensen-Shannon (JS) divergence, which is stated as

$$JS(p_1 \| p_2) = \frac{1}{2} KL\left(p_1 \| \frac{p_1 + p_2}{2}\right) + \frac{1}{2} KL\left(p_2 \| \frac{p_1 + p_2}{2}\right). \tag{9}$$

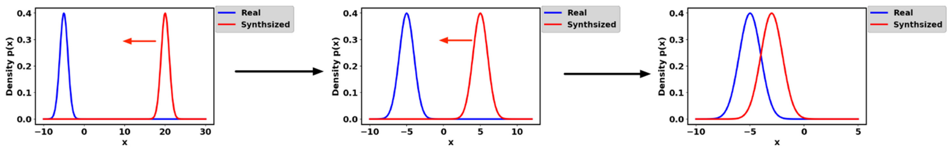


Fig. 10. Illustration of training progression for a GAN. Two normal distributions are used here for visualization. Given an optimal D, the objective of the GAN is to update G to move the generated distribution p_g (red) toward the real distribution p_r (blue) (G is updated from left to right in this figure. Left: initial state, middle: during training, right: training convergence). However, JS divergence for the left two figures are both 0.693 and the figure on the right is 0.336, indicating that JS divergence does not provide sufficient gradient at the initial state.

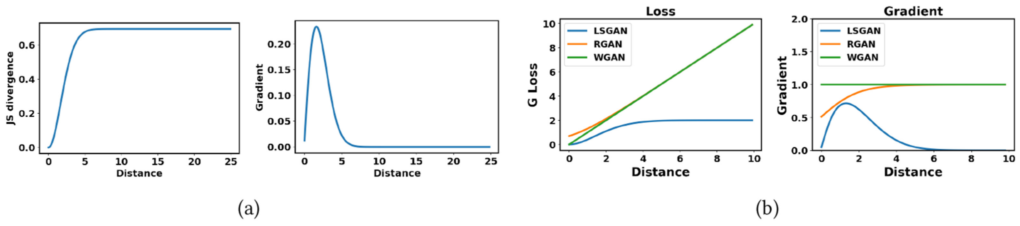


Fig. 11. JS divergence and gradient change with the distance between p_r and p_g. (a) Left: JS divergence changes with distance. Right: Gradient JS divergence changes with distance. The distance is the difference between the two distribution means. (b) Loss and gradient for generator of different loss-variant GANs.

Thus the loss for G regarding the optimal D in Equation (7) can be reformulated as

$$\mathcal{L}_G = 2 \cdot JS(p_r \| p_g) - 2 \cdot \log 2, \tag{10}$$

which indicates that the loss for G now equally becomes the minimization of the JS divergence between p_r and p_g. With training D step by step, the optimization of G will be closer to the minimization of JS divergence between p_r and p_g. We can now start to describe the instability problem in training, where D often easily wins over G. This unstable training problem is actually caused by the JS divergence in Equation (9). Given an optimal D, the objective of optimization for Equation (10) is to move p_g toward p_r (see Figure 10). JS divergence for the three plots from left to right are 0.693, 0.693, and 0.336, which indicates that JS divergence remains constant ($\log 2 = 0.693$) if there is no overlap between p_r and p_g. Figure 11(a) demonstrates the change of JS divergence and its gradient corresponding to the distance between p_r and p_g. It can be seen that JS divergence is constant and its gradient is almost 0 when the distance is greater than 5, which indicates that training process does not have any effect on G. The gradient of JS divergence for training the G is non-zero only when p_g and p_r have substantial overlap, i.e., the vanishing gradient will arise for G when D is close to optimal. In practice, the possibility that p_r and p_g do not overlap or have negligible overlap is very high [3].

The original GANs work [45] also highlights the minimization of $-\mathbb{E}_{\mathbf{x}\sim p_g} \log[D(\mathbf{x})]$ for training G to avoid a vanishing gradient. However, this training strategy will lead to another problem called mode dropping. First, let us examine $KL(p_g \| p_r) = \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g}{p_r}$. With an optimal discriminator D^*, $KL(p_g \| p_r)$ can be reformulated as

$$\begin{aligned} KL(p_g \| p_r) &= \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g(\mathbf{x})/(p_r(\mathbf{x}) + p_g(\mathbf{x}))}{p_r(\mathbf{x})/(p_r(\mathbf{x}) + p_g(\mathbf{x}))} = \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{1 - D^*(\mathbf{x})}{D^*(\mathbf{x})} \\ &= \mathbb{E}_{\mathbf{x}\sim p_g} \log[1 - D^*(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g} \log[\mathrm{D}^*(\mathbf{x})]. \end{aligned} \tag{11}$$

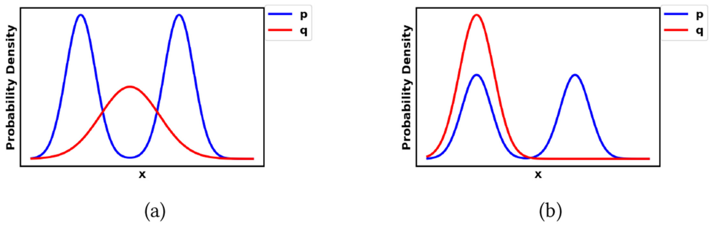


Fig. 12. Optimized q (in red) when minimizing KL divergence $KL(p\|q)$ (left) and reverse KL divergence $KL(q\|p)$ (right).

The alternative loss form for G can now can be stated by switching the order of the two sides in Equation (11)

$$\begin{aligned} -\mathbb{E}_{\mathbf{x}\sim p_g}\log[D^*(\mathbf{x})] &= KL(p_g\|p_r) - \mathbb{E}_{\mathbf{x}\sim p_g}\log[1-D^*(\mathbf{x})] \\ &= KL(p_g\|p_r) - 2\cdot JS(p_r\|p_g) + 2\cdot\log 2 + \mathbb{E}_{\mathbf{x}\sim p_x}\log[D^*(\mathbf{x})], \end{aligned} \tag{12}$$

where the alternative loss for G in Equation (12) is only affected by the first two terms (the last two terms are constant), however, this loss function is dominated by $KL(p_g\|p_r)$, since $JS(p_r\|p_g)$ is bounded in $[0, \log 2]$ as illustrated in Figure 11(a) on the left. It can be seen that the first term in Equation (12) is reverse KL divergence, in which the p_g optimized by the reverse is totally different from the p_g optimized by KL divergence. Figure 12 illustrates this difference by using a mixture of two Gaussians for p and a single Gaussian for q. When p has multiple modes, q tries to blur all modes together to put a high-probability mass on all as seen in Figure 12(a). However, Figure 12(b) shows that q chooses to recover a single Gaussian to avoid putting probability mass in the low-probability areas at the centre of the two Gaussians. The optimization on reverse KL divergence therefore will cause mode collapse during the training of GANs. This is highlighted below,

- When $p_g(\mathbf{x}) \to 0$, $p_r(\mathbf{x}) \to 1$, $KL(p_g\|p_r) \to 0$.
- When $p_g(\mathbf{x}) \to 1$, $p_r(\mathbf{x}) \to 0$, $KL(p_g\|p_r) \to +\infty$.

The penalization for two instances of poor performance made by G are totally different. The first instance of poor performance is that G is not producing a reasonable range of samples and yet incurs a very small penalization. The second instance of poor performance concerns G producing implausible samples but has very large penalization. The first example concerns the fact that the generated samples lack diversity while the second concerns that fact that the generated samples are not accurate. Considering this first case, G generates repeated but "safe" samples instead of taking risk to generate diverse but "unsafe" samples, which leads to the mode collapse problem. In summary, using the original loss in Equation (1) will result in the vanishing gradient for training G and using the alternative loss in Equation (12) will incur the mode collapse problem. These kinds of problems cannot be solved by simply changing architectures. Therefore, it could be argued that the ultimate problem for GANs stems from the design of the loss function and that innovative ideas for this redesign of the loss function may solve the problem. Loss-variant GANs have been researched extensively to improve the stability of training GANs and we consider these next.

5.1 Wasserstein GAN

WGAN [4] has successfully solved the two problems for the original GAN by using the Earth mover (EM) or Wasserstein-1 [115] distance as the loss measure for optimization. The EM distance is defined as

$$W(p_r, p_g) = \inf_{\gamma\in\prod(p_r,p_g)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}\|\mathbf{x}-\mathbf{y}\|, \tag{13}$$

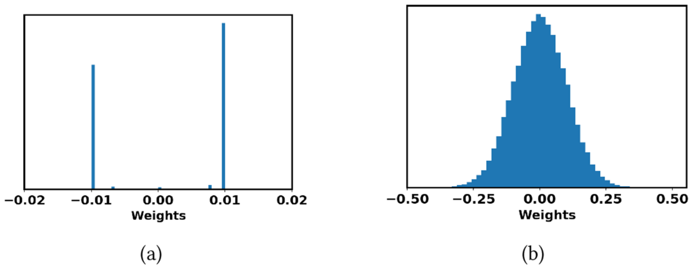


Fig. 13. Comparison of the parameter distribution between WGAN (left) and WGAN-GP (right). Figure from Reference [47].

where $\prod(p_r, p_g)$ denotes the set of all joint distributions and $\gamma(\mathbf{x}, \mathbf{y})$ whose marginals are p_r and p_g. Compared with KL and JS divergence, EM is able to reflect distance even when p_r and p_g do not overlap. It is also continuous and thus able to provide a meaningful gradient for training the generator. Figure 11(b) illustrates the WGAN gradient compared to that for the original GAN. It is noticeable that WGAN has a smooth gradient for training the generator spanning the complete space. However, the infimum in Equation (13) is intractable but the creators demonstrate that the Wasserstein distance can be alternatively estimated as

$$\max_{w \sim \mathcal{W}} \mathbb{E}_{\mathbf{x}_{p_r}}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_z}[f_w(G(\mathbf{z}))], \tag{14}$$

where f_w can be realized by D but has some constraints (for details the interested reader can refer to the original work [4]) and $\mathbf{z}$ is the input noise for G. So w here is the parameters in D where D aims to maximize equation (14) to make the optimization distance equivalent to the Wasserstein distance. When D is optimized, Equation (13) will become the Wasserstein distance and G aims to minimize it. So the loss for G is

$$-\min_{G} \mathbb{E}_{\mathbf{z} \sim p_z}[f_w(G(\mathbf{z}))]. \tag{15}$$

An important difference between WGAN and the original GAN is the function of D. The D in the original work is used as a binary classifier but D as used in the WGAN has the purpose of fitting the Wasserstein distance, which is a regression task. Thus, the sigmoid in the last layer of D is removed in the WGAN. The authors trained WGAN on the LSUN dataset with 64×64 resolution. *Importantly, training of the WGAN will be unstable if a momentum based optimizer such as Adam ($\beta_1 > 0$ is used).* Therefore, RMSProp is utilized for training the WGAN.

5.2 WGAN-GP

Even though WGAN has been shown to be successful in improving the stability of GAN training, it is not well generalized for a deeper model. Experimentally it has been determined that most WGAN parameters are localized at -0.01 and 0.01 because of parameter clipping (see Figure 13). This will dramatically reduce the modeling capacity of D. WGAN-GP has been proposed using a gradient penalty for restricting $\|f\|_L \leq K$ for the discriminator [47] and the modified loss for the discriminator now becomes

$$\mathcal{L}_D = \mathbb{E}_{\mathbf{x}_g \sim p_g}[D(\mathbf{x}_g)] - \mathbb{E}_{\mathbf{x}_r \sim p_r}[D(\mathbf{x}_r)] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2], \tag{16}$$

where $\mathbf{x}_r$ is sample data drawn from the real data distribution p_r, $\mathbf{x}_g$ is sample data drawn from the generated data distribution p_g and $p_{\hat{\mathbf{x}}}$ is sampled uniformly along the straight lines between those pairs of points, which are sampled from the real data distribution p_r and the generated data distribution p_g. The first two terms are the original loss in WGAN and the last term is the gradient penalty. WGAN-GP demonstrates a better distribution of trained parameters compared to WGAN

(Figure 13) and better stability performance during training of GANs. Before WGAN-GP, successful training of GANs only took place on those models consisting of a few layers both in the discriminator and generator, i.e., DCGAN uses four convolutional layers in D and 4 deconvolutional layers in G. WGAN-GP successfully demonstrates stabilizing training by using the ResNet-101 architecture as the backbone. This has had significant impact on research into GANs for large-scale image generation, i.e., PROGAN, BigGAN. As mentioned in the previous section, WGAN has stability issues when using momentum based optimizers such as Adam. WGAN-GP in contrast exhibits stabilizing training by using the Adam optimizer and even faster convergence using the same training settings. WGAN-GP was experimentally explored using the ImageNet dataset with 32×32 image resolution, the LSUN dataset with 128×128 image resolution and CIFAR-10 with 32×32 image resolution. Adam optimizer with $\alpha = 1 \times 10^{-4}$, $\beta_1 = 0$, $\beta_2 = 0.9$ was utilized in the experiment. The learning rate was 2×10^{-4} and a batch size of 64. The authors find piecewise linear activation functions, e.g., ReLU, Leaky ReLu and smooth activation functions, e.g., Tanh both can train the WGAN-GP in a stable way.

5.3 Least Square GAN (LSGAN)

The Least Square GAN (LSGAN) is a new approach proposed in Reference [94] to remedy the vanishing gradient problem for G from the perspective of the decision boundary determined by the discriminator. This work argues that the decision boundary for D of the original GAN penalizes as very small error in updates of G for those generated samples that are far away from the decision boundary. The creators propose using a least square loss for D instead of sigmoid cross entropy loss as stated in the original GAN paper [45]. The proposed loss function is defined as

$$\min_D \mathcal{L}_D = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_r}[(D(\mathbf{x}) - b)^2] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z}[(D(G(\mathbf{z})) - a)^2], \min_G \mathcal{L}_G = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z}[(D(G(\mathbf{z})) - c)^2], \tag{17}$$

where a is the label for the generated samples, b is the label for the real samples and c is the hyperparameter that G wants D to recognize the generated samples as the real samples by mistake. This modification has two benefits: (1) The new decision boundary generated by D penalizes large errors arising from those generated samples that are far away from the decision boundary. This pushes those "bad" generated samples toward the decision boundary. This is beneficial in terms of generating improved image quality. (2) Penalizing the generated samples that are far away from the decision boundary provides sufficient gradient when updating the G, which remedies the vanishing gradient problems for training G. Figure 14 demonstrates a comparison of decision boundaries for LSGAN and the original GAN. The work [94] has proven that the optimization of LSGAN is equivalent to minimizing the Pearson χ^2 divergence between $p_r + p_g$ and $2p_g$ when a, b and c subject to $b - c = 1$ and $b - a = 2$. Similarly to WGAN, D here involves regression and the sigmoid is also removed. LSGAN was evaluated on LSUN and HWDB1.0 [81] with 112×112 image resolution. The Adam optimizer with $\beta_1 = 0.5$ was used and the learning rate was 1×10^{-3} and 2×10^{-4} for LSUN and HWDB1.0, respectively. Similarly to DCGAN, ReLU activations and Leaky ReLU activations were used for the generator and discriminator respectively.

5.4 f-GAN

f-GAN works on the basis that GANs can be trained by using f-divergence [104]. f-divergence $D_f(p_r \| p_g)$ measures the difference between two probability distributions (p_r and p_g regarding GANs), e.g., KL divergence, JS divergence, and Pearson χ^2 as mentioned before, which can be summarized as

$$D_f(p_r \| p_g) = \int_{\mathcal{X}} p_g(x) f\left(\frac{p_r(x)}{p_g(x)}\right) dx, \tag{18}$$

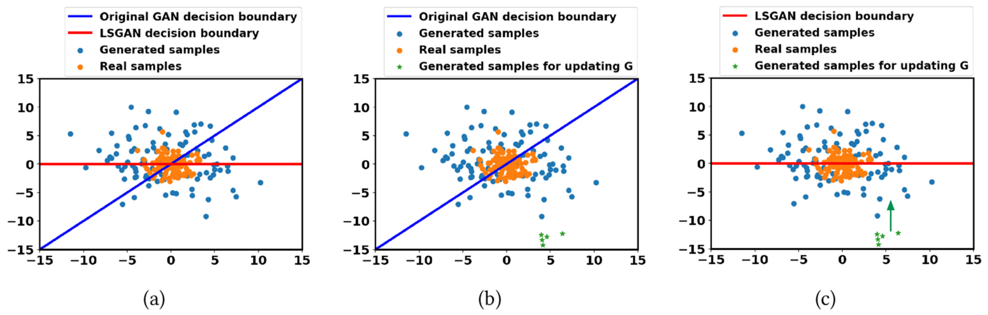


Fig. 14. Decision boundary illustration of original GAN and LSGAN. (a). Decision boundaries for D of original GAN and LSGAN. (b). Decision boundary of D for the original GAN. It has small errors for the generated samples, which are far away from the decision boundary (in green), for updating G. (c). Decision boundary for D of LSGAN. It penalizes as large error any generated sample that is far away from the boundary (in green). Thus it pushes generated samples (in green) toward the boundary [94].

Table 1. Examples of $D_f(p_r \Vert p_g)$ with f-divergence Function Based on References [103, 104]

Divergence	$D_f(p_g \Vert p_q)$	f-divergence function
KL divergence	$\int p_r(x) \log \frac{p_r(x)}{p_g(x)} dx$	$t \log t$
Reverse KL	$\int p_g(x) \log \frac{p_g(x)}{p_r(x)} dx$	$-\log t$
FCGAN $(2 \cdot JS - 2 \cdot \log 2)$ [45]	$\int p_r(x) \log \frac{2p_r(x)}{p_r(x)+p_g(x)} + p_g(x) \log \frac{2p_g(x)}{p_r(x)+p_g(x)} dx - 2 \cdot \log 2$	$t \log t - (t+1) \log(t+1)$
LSGAN (Pearson $\mathcal{X}^2$) [94]	$\int \frac{(p_g(x)-p_r(x))^2}{p_r(x)} dx$	$(t-1)^2$
EBGAN [151]	$\int \lvert p_r(x) - p_g(x) \rvert dx$	$\lvert t-1 \rvert$

where f is a convex function and $f(1) = 0$. It should be noted that f is termed a generator function in the original paper [104], which is totally different from the concept of a generator G in GANs. Thus we use f or f-divergence function in this section instead of generator function in the original paper to avoid confusion with the generator G used in this article. f-GAN generalizes the loss function of GANs according to an f-divergence function presented in Equation (18). A list of f-divergence with f-divergence functions are shown in Table 1. However, Equation (18) is intractable thus it requires estimation in a computable way such as through the use of an expectation form. By using the convex conjugate (Fenchel conjugate) $f(u) = \sup_{t \in \mathrm{dom}_{f^*}} \{tu - f^*(t)\}$ [50], f-divergence can be represented as a lower bound on the divergence

$$\begin{aligned} D_f(p_r \| p_g) &= \int_{\mathcal{X}} p_g(x) \sup_{t \in \mathrm{dom}_{f^*}} \left(t \frac{p_r(x)}{p_g(x)} - f^*(t) \right) dx \geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} T(x) p_r(x) - f^*(T(x)) p_g(x) \right) dx \\ &= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x \in p_r}[T(x)] - \mathbb{E}_{x \in p_g}[f^*(T(x))] \right), \end{aligned} \tag{19}$$

where $\mathcal{T}$ is an arbitrary function class of T that satisfies $\mathcal{X} \to \mathbb{R}$ (e.g., a parameterized discriminator with a specific activation function such as a sigmoid). The derivation above yields a lower bound for $D_f(p_r \| p_g)$ that is tractable, thus this can be directly calculated. The optimization for f-GAN firstly is characterized by maximizing the lower bound (last line in Equation (19)) with respect to the discriminator, which aims to make the lower bound the estimation of f-divergence, and then minimizes the f-divergence regarding the generator to bring p_g close to p_r. This optimization is known as variational divergence minimization (VDM). The authors trained generative

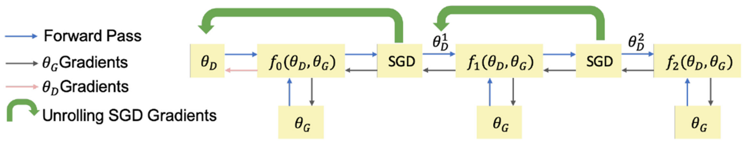


Fig. 15. An example of computation for an unrolled GAN with three unrolling steps. G and D update using Equation (22). Each step k uses the gradients of f_k regarding θ_D^k stated in the Equation (20).

neural samplers based on VDM on MNIST (28 × 28 pixel images) and LSUN (96 × 96 pixel images). The model architecture and training settings are the same as proposed in DCGAN.

5.5 Unrolled GAN

Unrolled GAN (UGAN) is a design proposed to solve the problem of mode collapse for GANs during training [98]. The core design innovation of UGAN is the addition of a gradient term for updating G, which has an ability of capturing responses of the discriminator to a change in the generator. The optimal parameter for D can be expressed as an iterative optimization procedure as follows:

$$\theta_D^0 = \theta_D, \quad \theta_D^{k+1} = \theta_D^k + \eta^k \frac{\mathrm{d}f(\theta_G \theta_D^k)}{\mathrm{d}\theta_D^k}, \quad \theta_D^*(\theta_G) = \lim_{k\to\infty} \theta_D^k, \tag{20}$$

where η^k is the learning rate, θ_D represents parameters for D and θ_G represents parameters for G. The surrogate loss by unrolling for K steps can be expressed as

$$f_K(\theta_G, \theta_D) = f(\theta_G, \theta_D^K(\theta_G, \theta_D)). \tag{21}$$

This surrogate loss is then used for updating parameters for D and G,

$$\theta_G \leftarrow \theta_G - \eta \frac{\mathrm{d}f_K(\theta_G \theta_D)}{\mathrm{d}\theta_G}, \quad \theta_D \leftarrow \theta_D + \eta \frac{\mathrm{d}f(\theta_G \theta_D)}{\mathrm{d}\theta_D}. \tag{22}$$

Figure 15 illustrates the computational diagram for an unrolled GAN with three unrolling steps. Equation (23) illustrates the gradient for updating G,

$$\frac{\mathrm{d}f_K(\theta_G, \theta_D)}{\mathrm{d}\theta_G} = \frac{\partial f(\theta_G, \theta_D^K(\theta_G, \theta_D))}{\theta_G} + \frac{\partial f(\theta_G, \theta_D^K(\theta_G, \theta_D))}{\partial \theta_D^K(\theta_G, \theta_D)} \frac{\mathrm{d}\theta_D^K(\theta_G, \theta_D)}{\mathrm{d}\theta_G}. \tag{23}$$

It should be noted that the first term in Equation (23) is the gradient for the original GAN. The second term here reflects how D reacts to changes in G. If G tends to collapse to one mode, then D will increase the loss for G. Thus, this unrolled approach is able to prevent the mode collapse problem for GANs. The authors trained UGAN on MNIST and CIFAR10 datasets. All convolutions have kernel size of 3 × 3 with batch normalization. The discriminator used Leaky ReLU with a 0.3 leakiness and the generator used ReLUs. The generator consisted of five layers, with one fully connected layer, three deconvolutional layers, and one convolutional layer. The discriminator had four layers, three of which are convolutional layers and one fully connected layer. The Adam optimizer with a generator learning rate of 1×10^{-4} and a discriminator learning rate as 2×10^{-4} was utilized in the experiment.

5.6 Loss Sensitive GAN (LS-GAN)

LS-GAN is introduced to train the generator to produce realistic samples by minimizing the designated margins between real and generated samples [111]. This work argues that the problems such as the vanishing gradient and mode collapse as appearing in the original GAN is caused by a non-parametric hypothesis that the discriminator is able to distinguish any type of probability

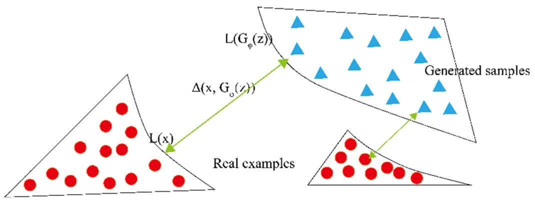


Fig. 16. Demonstration of the loss in Equation (25). $\Delta(\mathbf{x}, G(\mathbf{z}))$ is used to separate real samples and generated samples. If some generated samples are close enough to the real samples, then LS-GAN will focus on other generated samples that are far away from the real samples. This optimization loss puts a restriction on D to prevent it from separating generated and real samples too well. Thus, it solves the vanishing gradient problem that arises in the original GAN design. ($G_\phi(\mathbf{z})$ here is equivalent to $G(\mathbf{z})$ where ϕ represents the parameters for generator). Figure is regenerated from Reference [111].

distribution between real samples and generated samples. As mentioned before, it is very normal for the overlap between the real samples distribution and the generated samples distribution to be negligible. Moreover, D is also able to separate real samples and generated samples. The JS divergence will become a constant under this situation, where the vanishing gradient arises for G. In LS-GAN, the classification ability of D is restricted and is learned by a loss function $L_\theta(\mathbf{x})$ parameterized with θ, which assumes that a real sample ought to have smaller loss than a generated sample. The loss function can be incorporated as the following constraint:

$$L_\theta(\mathbf{x}) \leq L_\theta(G(\mathbf{z})) - \Delta(\mathbf{x}, G(\mathbf{z})), \tag{24}$$

where $\Delta(\mathbf{x}, G(\mathbf{z}))$ is the margin measuring the difference between real samples and generated samples. This constraint indicates that a real sample is separated from a generated sample by at least a margin of $\Delta(\mathbf{x}, G(\mathbf{z}))$. The optimization for the LS-GAN is then stated as

$$\min_D \mathcal{L}_D = \mathbb{E}_{\mathbf{x}\sim p_r} L_\theta(\mathbf{x}) + \lambda \mathbb{E}_{\substack{\mathbf{x}\sim p_r \\ \mathbf{z}\sim p_z}} (\Delta(\mathbf{x}, G(\mathbf{z})) + L_\theta(\mathbf{x}) - L_\theta(G(\mathbf{z})))_+, \ \min_G \mathcal{L}_G = \mathbb{E}_{\mathbf{z}\sim p_z} L_\theta(G(\mathbf{z})), \tag{25}$$

where λ is a positive balancing parameter, $(a)_+ = \max(a, 0)$ and $\boldsymbol{\theta}$ are the parameters in D. From the second term in $\mathcal{L}_D$ in the Equation (25), $\Delta(\mathbf{x}, G(\mathbf{z}))$ is added as a regularization term for optimizing D to prevent D from overfitting the real samples and the generated samples. Figure 16 demonstrates the efficacy of Equation (25). The loss for D puts a restriction on the capability of D, i.e., it challenges the ability of D to separate well-generated samples from real samples, which is the original cause for the vanishing gradient. More formally, LS-GAN assumes that p_r lies in a set of Lipschitz densities with a compact support.

The models were trained with CIFAR-10, SVHN [102], and CelebA with a mini-batch of 64 images. All weights were initialized from a zero-mean Gaussian distribution with a standard deviation of 0.02. The Adam optimizer was used to train the model with an initial learning rate set as 1×10^{-3} and β_1 as 0.5 while the learning rate was decreased every 25 epochs by a factor of 0.8.

5.7 Mode Regularized GAN

Mode Regularized GAN proposes a metric for regularization to penalize missing modes [18], which is then used to solve the mode collapse problem. The key idea behind this work is the use of an encoder $E(\mathbf{x})\colon \mathbf{x} \to \mathbf{z}$ to produce the latent variable $\mathbf{z}$ for G instead of using noise. This procedure has two benefits: (1) The encoder reconstruction can add more information to G so that is not that easy for D to distinguish between generated samples and real samples; and (2) the encoder ensures a correspondence between $\mathbf{x}$ and $\mathbf{z}$ ($E(\mathbf{x})$), which means G can cover different modes in the $\mathbf{x}$ space.

So it prevents the mode collapse problem. The loss function for this mode regularized GAN is

$$\begin{aligned}\mathcal{L}_G &= -\mathbb{E}_{\mathbf{z}}[\log[D(G(\mathbf{z}))]] + \mathbb{E}_{\mathbf{x}\sim p_r}[\lambda_1 d(\mathbf{x}, G \circ E(\mathbf{x})) + \lambda_2 \log[D(G(\mathbf{x}))]] \\ \mathcal{L}_E &= \mathbb{E}_{\mathbf{x}\sim p_r}[\lambda_1 d(\mathbf{x}, G \circ E(\mathbf{x})) + \lambda_2 \log[D(G(\mathbf{x}))]],\end{aligned} \tag{26}$$

where d is a geometric measurement that can be chosen from many options, e.g., pixelwise L^2 and distance of extracted features. The authors evaluate the performance of including the mode regularization on the MNIST and CelebA (64×64) datasets.

5.8 Geometric GAN

The loss function for geometric GANs [80] can be derived via an alternative means by minimizing the hinge loss as

$$\begin{aligned}\mathcal{L}_D &= -\mathbb{E}_{(x,y)\sim p_r}[min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_r}[min(0, -1 - D(G(z), y))] \\ \mathcal{L}_G &= -\mathbb{E}_{z\sim p_z, y\sim p_r} D(G(z), y).\end{aligned} \tag{27}$$

This hinge loss fashion is also deployed in the SAGAN mentioned in Section 4.11 and BigGAN mentioned in Section 4.12. Compared to the other loss functions, the authors demonstrate the efficacy of hinge loss for dealing with the high-dimension low-sample size problem [2, 17, 96], which is a classification problem caused when the mini-batch size is much smaller than the dimension of the feature space. In this article, the geometric GAN is designed based on a soft-margin SVM linear classifier rather than a hard-margin SVM linear classifier. The networks were trained on MNIST (64×64 resolution), CelebA (64×64 resolution), and LSUN (64×64 resolution) datasets. The DCGAN architecture trained using the RMSprop optimizer with learning rate 2×10^{-4} and mini-batch size 64 was deployed in this work. The authors demonstrate that geometric GAN is more stable for training and less prone to mode collapse.

5.9 Relativistic GAN

Relativistic GAN (RGAN) [60] is proposed as a general approach to devising new cost functions from the existing one, i.e., it can be generalized for all IPM [101, 123] GANs. The discriminator in the original GAN measures *the probability for a given real sample or a generated sample*. The authors argue that key relative discriminant information between real data and generated data is missing in the original GAN. The discriminator in RGAN takes into account *how a given real sample is more realistic compared to a given random generated sample*. The loss function of RGAN applied to the original GAN design is stated as

$$\min_D \mathbb{E}_{\substack{\mathbf{x}_r\sim p_r \\ \mathbf{x}_g\sim p_g}}[\log(\text{sigmoid}(C(\mathbf{x}_r) - C(\mathbf{x}_g)))], \quad \min_G \mathbb{E}_{\substack{\mathbf{x}_r\sim p_r \\ \mathbf{x}_g\sim p_g}}[\log(\text{sigmoid}(C(\mathbf{x}_g) - C(\mathbf{x}_r)))], \tag{28}$$

where $C(\mathbf{x})$ is the non-transformed layer. Figure 17 demonstrates the effect on D of using the RGAN approach compared to the original GAN. In terms of the original GAN, the optimization aims to push the $D(\mathbf{x})$ to 1 (right one). For RGAN, the optimization aims to push $D(\mathbf{x})$ to 0.5 (left one), which is more stable compared to the original GAN. The author also claims that RGAN can be generalized to other types of loss-variant GANs if those loss functions belong to IPMs. The generalization loss is stated as

$$\begin{aligned}\mathcal{L}_D &= \mathbb{E}_{\substack{\mathbf{x}_r\sim p_r \\ \mathbf{x}_g\sim p_g}}[f_1(C(x_r) - C(x_g))] + \mathbb{E}_{\substack{\mathbf{x}_r\sim p_r \\ \mathbf{x}_g\sim p_g}}[f_2(C(x_g) - C(x_r))] \\ \mathcal{L}_G &= \mathbb{E}_{\substack{\mathbf{x}_r\sim p_r \\ \mathbf{x}_g\sim p_g}}[g_1(C(x_r) - C(x_g))] + \mathbb{E}_{\substack{\mathbf{x}_r\sim p_r \\ \mathbf{x}_g\sim p_g}}[g_2(C(x_g) - C(x_r))],\end{aligned} \tag{29}$$

where $f_1(y) = g_2(y) = -y$ and $f_2(y) = g_1(y) = y$. Details of loss generalization for other GANs can be found in the original paper [60].

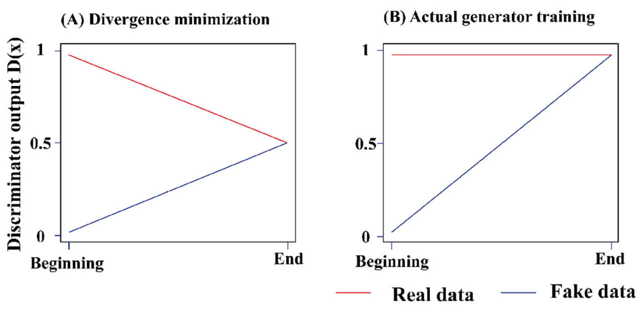


Fig. 17. *D* output comparison between RGAN and the original GAN. (a) *D* output in RGAN; (b) *D* output in the original GAN when training the *G*. Figure is regenerated from Reference [60].

The authors trained the networks on the CIFAR-10 and the CAT datasets with various image sizes, i.e., 64×64, 128×128, and 256×256. The DCGAN architecture with the Adam optimizer was used. Various training settings have been explored and more details can be found in the original paper [60]. The authors successfully demonstrate that the relativistic discriminator offers a way to improve on the standard GAN approach and is able to achieve better performance with other tricks, e.g., spectral normalization and gradient penalty. More importantly, the authors demonstrate the generability of this approach, in which any type of GAN can be trained through an RGAN fashion.

5.10 SN-GAN

SN-GAN [100] proposes the use of weight normalization to train the discriminator in a more stable way. This technique is computationally light and easily applied to existing GANs. Previous work for stabilizing the training of GANs [4, 47, 111] emphasizes the importance that *D* should be from the set of *K-Lipshitz continuous functions. Lipschitz continuity* [5, 36, 42] can be considered a more strict form of continuity, one which requires that the function does not change rapidly. This smooth *D* is of benefit in stabilizing the training of GANs. The work mentioned previously focused on the control of the *Lipschitz constant* of the discriminator function. This work demonstrates an alternative, simpler way to control the *Lipschitz constant* through spectral normalization of each layer for *D*. Spectral normalization is performed as

$$\bar{\mathbf{W}}_{SN}(\mathbf{W}) = \frac{\mathbf{W}}{\sigma(\mathbf{W})}, \tag{30}$$

where $\mathbf{W}$ represents weights on each layer for D and $\sigma(\mathbf{W})$ is the L_2 matrix norm of $\mathbf{W}$. The article proves this will make $\|f\| \leq 1$. The fast approximation for the $\sigma(\mathbf{W})$ is also demonstrated in the original paper.

The authors evaluated the performance of SN-GAN on the CIFAR-10 (32×32 resolution), the STL-10 (48×48 resolution) [25] and the ImageNet (128×128 resolution) by comparing to the existing regularization/normalization techniques including weight clipping [4], gradient penalty [136], batch normalization [57], weight normalization [118], layer normalization [7], and orthonormal regularization [14]. Several training settings have been carried out for a comprehensive comparison. The authors demonstrate the efficacy of spectral normalization on the diversity and the quality of generated images compared to previously proposed approaches.

5.11 RealnessGAN

In the original GAN design, the discriminator only outputs 0 and 1, i.e., real and fake instead of a continuous distribution as the measure of realness. Xiangli et al. [137] propose the RealnessGAN to tackle this new perspective, which treats realness as a random variable that can be estimated from multiple angles. Traditional GANs adopt a single scalar (discriminator output) as the measure of realness. The authors argue that the realness is more complicated and covers multiple factors such as texture and overall configuration in the case of images. Following this observation, the discriminator is re-designed to learn a realness distribution instead of a single scalar. To achieve this, RealnessGAN replaces the single scalar by a distribution p_{realness} so that $D(\mathbf{x}) = \{p_{\text{realness}}(\mathbf{x}, u); u \in \Omega\}$ given by an input sample, where Ω is the set of outcomes of p_{realness} and each outcome u can be viewed as a potential realness measure by a chosen realness measuring criteria. In the original paper, the discriminator returns N probabilities on these N outcomes $\Omega = \{u_0, u_1, \ldots, u_{N-1}\}$ as

$$p_{\text{realness}(\mathbf{x}, u_i)} = \frac{e^{\theta_i(\mathbf{x})}}{\sum_j e^{\theta_j(\mathbf{x})}}, \tag{31}$$

where $\theta = \{\theta_0, \theta_1, \ldots, \theta_{N-1}\}$ are the parameters of D. Apart from the outcomes Ω, two distributions $\mathcal{A}_1$ for real and $\mathcal{A}_0$ for fake are also defined on Ω. In practical implementation, given a mini-batch $\{x_0, x_1, \ldots, x_{m-1}\}$, i.e., logits computed by the discriminator on the ith outcome, a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i)$ is fitted on $\{\theta_i(\mathbf{x}_0), \theta_i(\mathbf{x}_1), \ldots, \theta_i(\mathbf{x}_{m-1})\}$ and new logits is re-computed as $\{\theta'_i(\mathbf{x}_0), \theta'_i(\mathbf{x}_1), \ldots, \theta'_i(\mathbf{x}_{m-1}); \theta'_i \sim \mathcal{N}(\mu_i, \sigma_i)\}$. Increasing the number of outcomes will make D more rigorous and put more constraints on G. In other words, a larger number of outcomes is suggested for a more complicated dataset. The minmax loss can finally be represented as

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \text{KL}(\mathcal{A}_1 \| D(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_z} \text{KL}(\mathcal{A}_0 \| D(G(\mathbf{z}))), \quad \text{KL refers to KL divergence.} \tag{32}$$

The authors trained RealnessGAN on CIFAR10 and CelebA by using the Adam optimizer. The network architecture of RealnessGAN is identical to the DCGAN architecture with $\mathbf{z} \sim \mathcal{N}(0, I)$. Batch normalization was deployed for G and spectral normalization was applied for D. The number of outcomes were set to 51 for CelebA and 3 for CIFAR10 datasets, respectively.

5.12 Sphere GAN

Sphere GAN [108] is a novel IPM-based GAN, which uses the hypersphere to bound IPMs in the objective function, thereby it can facilitate stability of training. By exploiting the information of higher-order statistics of the data using geometric moment matching, the GAN model can provide more accurate results. The objective function of sphere GAN is defined as

$$\min_G \max_D \sum_r E_{\mathbf{x}} \left[d_s^r(\mathbf{N}), D(\mathbf{x})\right] - \sum_r E_{\mathbf{z}} \left[d_s^r(\mathbf{N}, D(G(\mathbf{z})))\right]. \tag{33}$$

For $r = 1, \ldots, R$ where the function d_s^r measures the rth moment distance between each sample and the north pole of the hypersphere, $\mathbf{N}$. Note that the subscript s indicates that d_s^r is defined on $\mathbb{S}^n$. Different from conventional discriminators based on the Wasserstein distance that require *Lipschitz constraints* (which forces the discriminators to be *1-Lipschitz functions*) the sphere GAN relaxes this condition by defining IPMs on the hypersphere. Figure 18 shows the pipeline of sphere GAN.

Unlike with conventional approaches such as WGAN-GP, WGAN-CT, and WGANL, sphere GAN does not need any additional constraints that forces discriminators to lie in a desired function space. By using geometric transformations, sphere GAN ensures that distance functions lie in a desired function space with no additional constraint term.

5.13 Self-supervised GAN (SS-GAN)

Although the conditional GAN has achieved great success in natural image synthesis. The main drawback of conditional GANs is the necessity for labeled data. Self-Supervised GANs [19] exploit

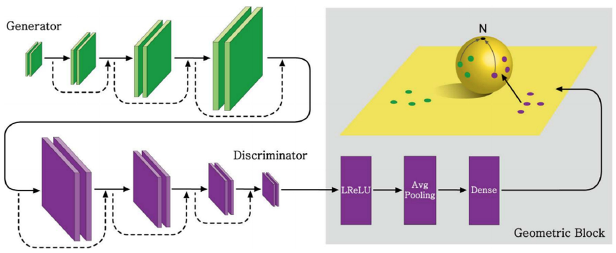


Fig. 18. Pipeline of sphere GAN. The generator outputs the real and fake data (generated via noisy inputs) into the discriminator. The final output is an n-dimensional Euclidean feature space (yellow plane). The green and purple circles are the feature points of the fake and real samples. *The key idea behind the sphere GAN is a remapping of the feature points on to the n-dimensional hypersphere* (i.e., yellow sphere). After the geometric transformation, the mapped points can be used for calculating the geometric moments centered at the north pole of the hypersphere. At the same time, the discriminator tries to maximize the moment differences of real and fake samples, while the generator tries to interfere with the discriminator by minimizing the moment differences. Figure is adopted from Reference [108].

adversarial training and *self-supervision* for bridging the gap between conditional and unconditional GANs.

This work imbues the discriminator with a mechanism to learn useful representations, independently of the quality of the current generator. In a self-supervised manner, they train a model on predicting rotation angle for extracting representations from the resulting networks, and then propose to add a self-supervised task (a rotation-based loss) to the discriminator, as

$$L_G = -V(G,D) - \alpha \mathbb{E}_{\mathbf{x}\sim P_G} \mathbb{E}_{r\sim\mathcal{R}} \left[\log Q_D \left(R = r|\mathbf{x}^r\right)\right], \tag{34}$$

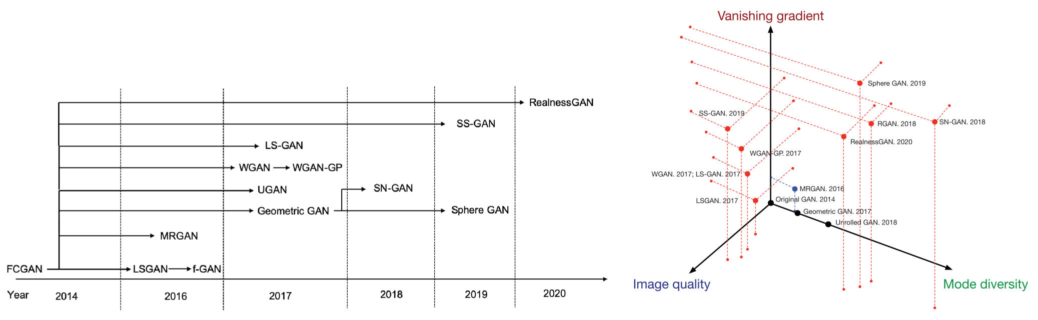
$$L_D = V(G,D) - \beta \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}} \mathbb{E}_{r\sim\mathcal{R}} \left[\log Q_D \left(R = r|\mathbf{x}^r\right)\right], \tag{35}$$

where $V(G,D)$ is the original value function [45], $r \in \mathcal{R}$ is a rotation selected from a set of possible rotations ($\mathcal{R} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$). Image $\mathbf{x}$ rotated by r degrees is denoted as $\mathbf{x}^r$, and $Q\,(R|\mathbf{x}^r)$ is the discriminator's predictive distribution over the angles of rotation of the sample. The implementation trick is they use output of the second last layer of discriminator added with a linear layer to predict the rotation type. This work tries to enforce the discriminator to learn good representation via learning the rotation information.

5.14 Summary

We explain the training problems (mode collapse and vanishing gradient for G) present in the original GAN design and we have introduced loss-variant GANs from the literature, which are proposed primarily for the purposes of improving the GANs performance in terms of three key aspects. Figure 19(a) illustrates the footprint of loss-variants that are discussed in this section. Figure 19(b) summarizes the efficacy of loss-variant GANs for these challenges. More details of quantitative results are provided in Section 7. Losses of LSGAN, RGAN, and WGAN are very similar to the original GAN loss. We use a toy example (i.e., two distributions used in Figure 10) to demonstrate the G loss regarding the distance between real data distribution and generated data distribution in Figure 11(b).

It can be seen that RGAN and WGAN are able to inherently solve the vanishing gradient problems for the generator when the discriminator is optimized. LSGAN in contrast still suffers from a vanishing gradient for the generator, however, it is able to provide a better gradient compared to the original GAN in Figure 11(a) when the distance between the real data distribution and the



(a) (b)

Fig. 19. (a) An overview of the footprint for loss-variant GANs discussed in this section. (b) Current loss-variants for solving the main identified challenges. Challenges are categorized in terms of three independent axes. Red points indicate the GAN-variant covers all three challenges, blue points cover two, and black points cover only one challenge. A larger value for each axis indicates better performance.

generated data distribution is relatively small. This is demonstrated in the original paper [94] where LSGAN is shown to more easily push the generated samples to the boundary established by the discriminator.

Loss-variant GANs can be applied to architecture-variants. However, SN-GAN and RGAN show stronger generalization abilities compared to other loss-variants, where these two loss-variants can be deployed by other types of loss-variants. Spectral normalization can be applied to any GAN-variant [100] while the RGAN concept can be applied to any IPM-based GAN [60]. We strongly recommend the use of spectral normalization for all GANs applications as described here. There are a number of loss-variant GANs mentioned in this article that are able to solve the mode collapse and training stability problems.

6 APPLICATIONS

6.1 Image Synthesis

Image synthesis is still a main focus area for GANs currently, and as a result has produced many GAN variants. In this section, we classify all applications related to images under the image synthesis category such as image super-resolution, image-to-image translation, and image matting.

Image Super-Resolution Image super-resolution enables a high-resolution image to be generated from a low-resolution image by upsampling. SRGAN [74] is a representative framework for image super-resolution by using GANs. Apart from general adversarial loss in GANs, SRGAN extends the loss by adding content loss (e.g., a pixelwise MSE loss) in the area of super-resolution, which lead to a perceptual loss presented as follows:

$$\mathcal{L}^{\mathrm{SR}} = \mathcal{L}^{\mathrm{SR}}_{\mathrm{X}} + 10^{-3}\mathcal{L}^{\mathrm{SR}}_{\mathrm{GAN}}, \tag{36}$$

where $\mathcal{L}^{\mathrm{SR}}_{X}$ is the content loss and $\mathcal{L}^{\mathrm{SR}}_{\mathrm{GAN}}$ is the GAN loss. In practice, the content loss $\mathcal{L}^{\mathrm{SR}}_{X}$ is chosen depending on applications. SRGAN presents three content losses (1) standard pixelwise MSE loss $\mathcal{L}^{\mathrm{SR}}_{MSE}$, (2) a loss defined on feature maps representing lower-level features $\mathcal{L}^{\mathrm{SR}}_{VGG22}$, and (3) a loss defined on feature maps representing higher-level features $\mathcal{L}^{\mathrm{SR}}_{VGG54}$. The authors show that different content losses perform differently according to different evaluation metrics. The generator in SRGAN is conditioned by low-resolution images, which are inferred with 4× upscaling factors. The authors show the superior perceptual performance of SRGAN compared to traditional approaches.

Image Completion/Repair Image completion/repair is a common image editing operation, which aims to fill the missing or masked regions in images with synthesized content. Most efficient traditional completion algorithms [9, 54] depend on low-level cues, which are used to search for patches from known regions of the same image, and synthesize the contents that locally appear similarly to the matched patches. These approaches perform well for background completion as patterns from the background are similar to each other. The assumption of similar patterns for a missing part with other parts in an image can be violated in some situations, e.g., filling missing parts for a face image, in which many objects have unique patterns. Li et al. [77] propose the use of an autoencoder incorporating GANs. Two discriminators D_G and D_L are deployed (one for global image content and the other for local image content), which produces two adversarial losses that are used for optimization. The overall loss function is represented as

$$\mathcal{L} = \mathcal{L}_r + \lambda_1 \mathcal{L}_{D_G} + \lambda_2 \mathcal{L}_{D_L} + \lambda_3 \mathcal{L}_p, \tag{37}$$

where $\mathcal{L}_r$ is the L_2 distance between the autoencoder output and the original image, $\mathcal{L}_{D_G}$ and $\mathcal{L}_{D_L}$ are adversarial losses of D_G and D_L, and $\mathcal{L}_p$ is the pixelwise softmax for the parsing network [85, 140]. λ_1, λ_2, and λ_3 are hyperparameters used to control the effects caused by different losses.

Image Matting Natural image matting is defined as the operation of accurately estimating the opacity of a foreground object in an image or video stream [88]. This field is attracting growing interest as it has wide applications such as for image editing and film post-production. Image matting approaches formally require an input image with a foreground object and the image background, which can be mathematically expressed as

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \quad \alpha_i \in [0, 1], \tag{38}$$

where α_i is a scalar value that defines the foreground opacity at pixel i, F_i is a scalar value of foreground object at pixel i and B_i is a scalar value of the background at pixel i. Lutz et al. introduces GANs to this field and proposed AlphaGAN that can produce visually appealing compositions. In their work, they train the discriminator on images composited with the ground-truth alpha and the predicted alpha and the generator is used to generate the compositions.

Image-to-image Translation Image-to-image translation is a class of graphic problems, in which the objective is to learn the mapping between an output image and an input image using a training set of aligned image pairs. Isola et al. [58] proposed the use of CGAN for image-to-image translation when paired training data is available. CycleGAN [153] is proposed as a soluution when unpaired training data is not available for image-to-image translation. CycleGAN achieves this by introducing a cycle consistency loss to enforce the requirement that the mapping from one domain X to the other domain Y is roughly the same in each direction. Other work such as DiscoGAN [66] and DualGAN [143] are also proposed to solve the missing paired training data issue in the area of image-to-image translation. More details can be found in the original papers.

6.2 Video Generation

GANs have already achieved striking results for natural images generation. Some recent work has attempted to extend this success to the area of video generation [24, 61, 126]. Convincing video generation using GANs remains a significant challenge as it exacerbates all the issues associated with image generation using GANs. Also it has the problems of increased memory and computation costs due to the nature of video and the requirement for temporal modeling. In addition, due to the limitation of memory and training stability, the generation becomes increasingly challenging with the increase of the resolution/duration of videos. The current foci of video generation research via GANs we believe are (1) the production of high-resolution videos, e.g., up to/more than

256×256, (2) increasing the lengths of the generated video beyond 48 frames, and (3) the generation of more realistic video so that much of the generated video content is not blurred, indistinct, or even surreal. Video-based GANs do not only have to consider the spatial modeling problem but also require temporal modeling, i.e., the motion between each adjacent frames in a video sequence. MoCoGAN [126] is proposed to learn motion and content in an unsupervised fashion and divides the image latent space into content and motion spaces. DVD-GAN [24] is able to generate longer and higher resolution videos based on the BigGAN architecture while introducing scalable, video-specific generator and discriminator architectures. DVD-GAN comprises two discriminators D_S (spatial discriminators) and D_T (temporal discriminators), in which D_S scores input frames from the spatial perspective and D_T scores input frames from the temporal perspective (motion).

6.3 Feature Generation

"Can machines think?" This is the question raised in Alan Turing's seminal paper entitled "Computing Machinery and Intelligence" [127] in 1950. The ultimate goal of machines is to be as intelligent as human beings. Current AI depends on large datasets rather than being generalized from small datasets and has a catastrophic forgetting problem [97] when learning tasks sequentially. Few-shot learning and continual learning attract significant attention from the scientific community in recent years and aims to fill the gaps between AI and humans. Both of these fields suffer from the problem of a lack of data, i.e., few-shot learning uses few samples (e.g., normally 1 sample or 5 samples) for each category and continual learning comes across unseen data in sequential tasks. Mandal et al. [93] proposes the use of conditional Wasserstein GAN with two additional terms, i.e., cosine embedding and cycle-consistency losses to synthesize unseen action features for zero-shot action recognition. Shin et al. [119] proposes a deep generative replay framework for continual learning, in which training data for previous tasks are sampled by the generator and interleaved with those for a new task. These show that GANs have the potential ability to provide solutions for other machine learning problems.

7 DISCUSSION

We have introduced the most significant problems present in the original GAN design namely mode collapse and vanishing gradients when updating G. We have surveyed significant GAN-variants that remedy these problems through two design considerations: (1) architecture-variants. This aspect focuses on architectural options for GANs. This approach enables GANs to be successfully applied to different applications, however, it is unable to fully solve the problems mentioned above; (2) loss-variants. We have provided a detailed explanation as to why these problems arise in the original GAN. These problems are essentially caused by the loss function in the original GAN. Consequently, modifying this loss function can solve this problem. It should be noted however that the loss function may change for some architecture-variants and it is, in many cases, an architecture-specific loss. Therefore, it is unable to generalize to other architectures. Table 2 demonstrates our summary for the performance of the presented GANs in this work by using Inception Score and FID as presented in the literature. Four image datasets are considered in the summary, i.e., CIFAR10, ImageNet, LSUN, and CelebA, which are the most widely used benchmarking datasets.

Interconnections between Architecture and Loss In this article, we highlight the problems inherent in the original GAN design. In highlighting how subsequent researchers have remedied those problems, we explored architecture-variants and loss-variants in GAN designs separately. However, it should be noted that there are interconnections between these two types of GAN-variants. As mentioned before, loss functions are easily integrated to different architectures. Benefiting from improved convergence and stabilization through a redesigned loss function,

Table 2. Performance Summary across Different Types of GANs Discussed in This Paper on Different Datasets in the Literature, i.e., CIFAR10, ImageNet, LSUN, and CelebA

Model	CIFAR10 (IS/FID)	ImageNet (IS/FID)	LSUN (FID)	CelebA (FID)
FCGAN	6.41/42.6	—/—	—	—
BEGAN	5.62/—	—/—	—	83.3
PROGAN	8.80/—	—/—	8.3	**7.3**
LSGAN	6.76/29.5	—/—	21.6	—
DCGAN	6.69/42.5	—/74.2	160.1	63.1
WGAN-GP	8.21/21.5	11.6/62.1	22.8	—
SN-GAN	8.43/18.8	36.8/27.6	—	—
Geometric GAN	—/27.1	—/—	—	—
RGAN	—/15.9	—/—	—	—
AC-GAN	8.25/—	—/—	—	—
BigGAN	**9.22/14.7**	**166.5/7.4**	—	—
RealnessGAN	—/34.6	—/—	—	23.5
MSG-GAN	—/—	—/—	**5.2**	8.0
SS-GAN	—/15.7	—/43.9	13.3	24.36
YLG	—/—	57.2/15.9	—	—
Sphere GAN	—/—	—/—	16.9	—

"—" refers to experiments have not been done in the literature. Studies from References [47, 56, 60, 63, 63, 64, 94, 95, 100, 106, 113, 120, 120, 121, 134, 137, 149].

architecture-variants are able to achieve better performance and accomplish solutions to more difficult problems. For examples, BEGAN and PROGAN use Wasserstein distance instead of JS divergence. SAGAN and BigGAN deploy spectral normalization, where they achieve good performance based on multi-class image generation. These two variant types equally contribute to GANs progress.

Future Directions GANs were originally proposed to produce plausible synthetic images and have achieved exciting performance in the computer vision area. GANs have been applied to some other fields (e.g., time-series generation [15, 39, 48, 87] and natural language processing [8, 40, 76, 147]) with some success. Compared to computer vision, GANs research in other areas is still somewhat limited. The limitation is caused by the different properties inherent in image versus non-image data. For instance, GANs work to produce continuous value data but natural language is based on discrete values like words, characters and bytes, so it is difficult to apply GANs successfully for such data. As this is also a very promising area, success in this area will lead to many applications such as generating subtitles and comments to live streaming, and so on. Research areas such as health have sensitivity around privacy issues in particular and successful data augmentation will have significant impact in these areas. However, generation for other data modalities relevant to such application areas such as time-series data has only been explored in a limited way. Notable challenges here include the lack of efficient metrics for evaluating the performance of GANs in such areas. More research is required in these areas.

Since the first GAN was proposed in 2014, their development has brought many benefits both in fomenting new research ideas and through impact in the world outside research and academia. However, improper use of GANs can bring challenges and should be of concern to society, e.g., GANs can be used to generate fake video of specific people and fabricate evidence of events that may have never happened, creating media sources that are counterfactual to a particular person's opinions and actions, detrimental to their reputation, and may even affect their personal safety

[1, 53]. We should therefore focus too, on developing forgery detection and processes to efficiently and effectively detect AI-genereated images (including those developed using GANs).

8 CONCLUSION

In this article, we reviewed GAN-variants based on performance improvement offered in terms of higher image quality, more diverse images and stability of training. We reviewed the current state of GAN-related research from both an architecture and a loss basis. More complex architectures with larger batch size are associated with an increase in both image quality and image diversity, i.e., BigGAN. However, limited GPU memory could be an issue for processing large batches of images. As an alternative the progressive training strategy used in PROGAN can increase image quality without requiring very large batches of images. In terms of image diversity, adding self-attention mechanisms to both the generator and the discriminator has achieved exciting results with the SAGAN demonstrating compelling performance using the ImageNet dataset. In terms of training stability, the loss function plays an important role here and different types of loss functions have been proposed to deal with this challenge. Having reviewed different types of loss functions, we find that spectral normalization has good generalization qualities, i.e., it can be applied to every GAN, is easy to implement and has very low computational cost. Current state-of-the-art GANs models such as BigGAN and PROGAN are able to produce high quality images and diverse images in the computer vision field. However, research that has applied GANs to the more challenging scenario of video is limited. Moreover, GAN-related research in other areas such as time-series generation and natural language processing lags that for computer vision in terms of performance and capability. We conclude that there are clearly opportunities for future research and application in these fields in particular.

ACKNOWLEDGMENT

The authors appreciate informative and insightful comments provided by anonymous reviewers, which significantly improve the quality of this survey.

REFERENCES

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. 2018. MesoNet: A compact facial video forgery detection network. In *Proceedings of the 2018 IEEE International Workshop on Information Forensics and Security (WIFS'18).* IEEE, 1–7.

[2] Jeongyoun Ahn and J. S. Marron. 2010. The maximal data piling direction for discrimination. *Biometrika* 97, 1 (2010), 254–259.

[3] Martin Arjovsky and Léon Bottou. 2017. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862.* Retrieved from https://arxiv.org/abs/1701.04862.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *arXiv:1701.07875.* Retrieved from https://arxiv.org/abs/1701.07875.

[5] Larry Armijo. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. *Pac. J. Math.* 16, 1 (1966), 1–3.

[6] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. 2014. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 3762–3769.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv:1607.06450.* Retrieved from https://arxiv.org/abs/1607.06450.

[8] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An Actor-critic Algorithm for Sequence Prediction. *arXiv:1607.07086.* Retrieved from https://arxiv.org/abs/1607.07086.

[9] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* 28, 3 (2009), 24.

[10] Shane Barratt and Rishi Sharma. 2018. A Note on the Inception Score. *arXiv:1801.01973.* Retrieved from https://arxiv.org/abs/1801.01973.

[11] David Berthelot, Thomas Schumm, and Luke Metz. 2017. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *arXiv:1703.10717*. Retrieved from https://arxiv.org/abs/1703.10717.

[12] Ali Borji. 2019. Pros and cons of GAN evaluation measures. *Comput. Vis. Image Understand.* 179 (2019), 41–65.

[13] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv:1809.11096*. Retrieved from https://arxiv.org/abs/1809.11096.

[14] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. 2016. Neural Photo Editing with Introspective Adversarial Networks. *arXiv:1609.07093*. Retrieved from https://arxiv.org/abs/1609.07093.

[15] Eoin Brophy, Zhengwei Wang, and Tomas E. Ward. 2019. Quick and Easy Time Series Generation with Established Image-based GANs. *arXiv:1902.05624*. Retrieved from https://arxiv.org/abs/1902.05624.

[16] Peter Burt and Edward Adelson. 1983. The laplacian pyramid as a compact image code. *IEEE Trans. Commun.* 31, 4 (1983), 532–540.

[17] Iain Carmichael and J. S. Marron. 2017. Geometric Insights into Support Vector Machine Behavior Using the KKT Conditions. *arXiv:1704.00767*. Retrieved from https://arxiv.org/abs/1704.00767.

[18] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 2016. Mode Regularized Generative Adversarial Networks. *arXiv:1612.02136*. Retrieved from https://arxiv.org/abs/1612.02136.

[19] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. 2019. Self-supervised GANs via auxiliary rotation loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*.

[20] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2172–2180.

[21] Zeyuan Chen, Shaoliang Nie, Tianfu Wu, and Christopher G. Healey. 2018. High Resolution Face Completion with Multiple Controllable Attributes via Fully End-to-end Progressive Generative Adversarial Networks. *arXiv:1801.07632*. Retrieved from https://arxiv.org/abs/1801.07632.

[22] Junsuk Choe, Song Park, Kyungmin Kim, Joo Hyun Park, Dongseob Kim, and Hyunjung Shim. 2017. Face generation for low-shot learning using generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 1940–1948.

[23] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8789–8797.

[24] A. Clark, J. Donahue, and K. Simonyan. 2019. Adversarial Video Generation on Complex Datasets. *arXiv:1907.06571*. Retrieved from https://arxiv.org/abs/1907.0657.

[25] Adam Coates, Andrew Ng, and Honglak Lee. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 215–223.

[26] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative adversarial networks: An overview. *IEEE Sign. Process. Mag.* 35, 1 (2018), 53–65.

[27] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial network embedding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[28] Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, and Ruslan R. Salakhutdinov. 2017. Good semi-supervised learning that requires a bad GAN. In *Advances in Neural Information Processing Systems*. 6510–6520.

[29] Giannis Daras, Augustus Odena, Han Zhang, and Alexandros G. Dimakis. 2020. Your local GAN: Designing two dimensional local attention mechanisms for generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*.

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 248–255.

[31] Emily L. Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*. 1486–1494.

[32] Carl Doersch. 2016. Tutorial on Variational Autoencoders. *arXiv:1606.05908*. Retrieved from https://arxiv.org/abs/1606.05908.

[33] Brian Dolhansky and Cristian Canton Ferrer. 2018. Eye in-painting with exemplar generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7902–7911.

[34] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Synthesizing Audio with Generative Adversarial Networks. *arXiv:1802.04208*. Retrieved from https://arxiv.orb/abs/1802.04208.

[35] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial Feature Learning. *arXiv:1605.09782*. Retrieved from https://arxiv.org/abs/1605.0978.

[36] Tzanko Donchev and Elza Farkhi. 1998. Stability and euler approximation of one-sided lipschitz differential inclusions. *SIAM J. Contr. Optim.* 36, 2 (1998), 780–796.

[37] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. 2017. Semantic image synthesis via adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 5706–5714.
[38] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. 2015. Training Generative Neural Networks via Maximum Mean Discrepancy Optimization. *arXiv:1505.03906*. Retrieved from https://arxiv.org/abs/1505.03906.
[39] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. 2017. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv:1706.02633*. Retrieved from https://arxiv.org/abs/1706.02633.
[40] William Fedus, Ian Goodfellow, and Andrew M. Dai. 2018. MaskGAN: Better Text Generation via Filling in the _. *arXiv:1801.07736*. Retrieved from https://arxiv.org/abs/1801.07736.
[41] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 315–323.
[42] A. A. Goldstein. 1977. Optimization of Lipschitz continuous functions. *Math. Program.* 13, 1 (1977), 14–22.
[43] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. 2019. AutoGAN: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'19)*.
[44] Ian Goodfellow. 2016. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160*. Retrieved from https://arxiv.orb/abs/1701.00160.
[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.
[46] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *J. Mach. Learn. Res.* 13, 1 (Mar. 2012), 723–773.
[47] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*. 5767–5777.
[48] Kay Gregor Hartmann, Robin Tibor Schirrmeister, and Tonio Ball. 2018. EEG-GAN: Generative Adversarial Networks for Electroencephalographic Brain Signals. *arXiv:1806.01875*. Retrieved from https://arxiv.org/abs/1806.01875.
[49] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*. 6626–6637.
[50] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. 2012. *Fundamentals of Convex Analysis*. Springer Science & Business Media.
[51] Saifuddin Hitawala. 2018. Comparative Study on Generative Adversarial Networks. *arXiv:1801.04271* (2). Retrieved from https://arxiv.org/abs/1801.04271.
[52] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. 2019. How generative adversarial networks and their variants work: An overview. *ACM Comput. Surv.* 52, 1 (2019), 10.
[53] Chih-Chung Hsu, Chia-Yen Lee, and Yi-Xiu Zhuang. 2018. Learning to detect fake face images in the wild. In *Proceedings of the 2018 International Symposium on Computer, Consumer and Control (IS3C'18)*. IEEE, 388–391.
[54] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. 2014. Image completion using planar structure guidance. *ACM Trans. Graph.* 33, 4 (2014), 1–10.
[55] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. *ACM Trans. Graph.* 36, 4 (2017), 107.
[56] Daniel Jiwoong Im, He Ma, Graham Taylor, and Kristin Branson. 2018. Quantitatively Evaluating GANs with Divergences Proposed for Training. *arXiv:1803.01045*. Retrieved from https://arxiv.org/abs/1803.01045.
[57] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*. Retrieved from https://arxiv.org/abs/1502.03167.
[58] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1125–1134.
[59] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. 2016. Texture Synthesis with Spatial Generative Adversarial Networks. *arXiv:1611.08207*. Retrieved from https://arxiv.org/abs/1611.08207.
[60] Alexia Jolicoeur-Martineau. 2018. The Relativistic Discriminator: A Key Element Missing from Standard GAN. *arXiv:1807.00734*. Retrieved from https://arxiv.org/abs/1807.00734.
[61] Emmanuel Kahembwe and Subramanian Ramamoorthy. 2019. Lower Dimensional Kernels for Video Discriminators. *arXiv:1912.08860*. Retrieved from https://arxiv.org/abs/1912.08860.
[62] Takuhiro Kaneko, Yoshitaka Ushiku, and Tatsuya Harada. 2019. Label-noise robust generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2467–2476.
[63] Animesh Karnewar and Oliver Wang. 2020. MSG-GAN: Multi-scale gradients for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7799–7808.
[64] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv:1710.10196*. Retrieved from https://arxiv.org/abs/1710.10196.

[65] Tero Karras, Samuli Laine, and Timo Aila. 2018. A Style-based Generator Architecture for Generative Adversarial Networks. *arXiv:1812.04948*. Retrieved from https://arxiv.org/abs/1812.04948.

[66] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, Volume 70. 1857–1865.

[67] Diederik P. Kingma and Max Welling. 2013. Auto-encoding Variational Bayes. *arXiv:1312.6114*. Retrieved from https://arxiv.org/abs/1312.6114.

[68] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. 2017. On Convergence and Stability of GANs. *arXiv:1705.07215*. Retrieved from https://arxiv.org/abs/1705.07215.

[69] Jean Kossaifi, Linh Tran, Yannis Panagakis, and Maja Pantic. 2018. GANGAN: Geometry-aware generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 878–887.

[70] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. Citeseer.

[71] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. 2018. The GAN Landscape: Losses, Architectures, Regularization, and Normalization. *arXiv:1807.04720*. Retrieved from https://arxiv.org/abs/1807.04720.

[72] Christoph Lassner, Gerard Pons-Moll, and Peter V. Gehler. 2017. A generative model of people in clothing. In *Proceedings of the IEEE International Conference on Computer Vision*. 853–862.

[73] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[74] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 105–114.

[75] Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. *arXiv:1901.04997*. Retrieved from https://arxiv.org/abs/1901.04997.

[76] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep Reinforcement Learning for Dialogue Generation. *arXiv:1606.01541*. Retrieved from https://arxiv.org/abs/1606.01541.

[77] Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. 2017. Generative face completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3911–3919.

[78] Yujia Li, Alexander Schwing, Kuan-Chieh Wang, and Richard Zemel. 2017. Dualing GANs. In *Advances in Neural Information Processing Systems*. 5606–5616.

[79] Yujia Li, Kevin Swersky, and Rich Zemel. 2015. Generative moment matching networks. In *Proceedings of the International Conference on Machine Learning*. 1718–1727.

[80] Jae Hyun Lim and Jong Chul Ye. 2017. Geometric GAN. *arXiv:1705.02894*. Retrieved from https://arxiv.org/abs/1705.02894.

[81] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. 2013. Online and offline handwritten chinese character recognition: Benchmarking on new databases. *Pattern Recogn.* 46, 1 (2013), 155–162.

[82] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*. 700–708.

[83] Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*. 469–477.

[84] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2018. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV'15)*.

[85] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[86] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. 2016. Semantic Segmentation Using Adversarial Networks. *arXiv:1611.08408*. Retrieved from https://arxiv.org/abs/1611.08408.

[87] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. 2018. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*. 1596–1607.

[88] Sebastian Lutz, Konstantinos Amplianitis, and Aljosa Smolic. 2018. AlphaGAN: Generative Adversarial Networks for Natural Image Matting. *arXiv:1807.10088*. Retrieved from https://arxiv.org/abs/1807.10088.

[89] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. 2017. Pose guided person image generation. In *Advances in Neural Information Processing Systems*. 406–416.

[90] Shuang Ma, Jianlong Fu, Chang Wen Chen, and Tao Mei. 2018. DA-GAN: Instance-level image translation by deep attention generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5657–5666.

[91] Dwarikanath Mahapatra, Behzad Bozorgtabar, and Rahil Garnavi. 2019. Image super-resolution using progressive generative adversarial networks for medical image analysis. *Comput. Med. Imag. Graph.* 71 (2019), 30–39.

[92] Dwarikanath Mahapatra, Behzad Bozorgtabar, Sajini Hewavitharanage, and Rahil Garnavi. 2017. Image super resolution using generative adversarial networks and local saliency maps for retinal image analysis. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 382–390.

[93] Devraj Mandal, Sanath Narayan, Sai Kumar Dwivedi, Vikram Gupta, Shuaib Ahmed, Fahad Shahbaz Khan, and Ling Shao. 2019. Out-of-distribution detection for generalized zero-shot action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 9985–9993.

[94] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *Proceedings of the 2017 IEEE International Conference on Computer Vision.* IEEE, 2813–2821.

[95] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. 2018. On the effectiveness of least squares generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 12 (2018), 2947–2960.

[96] James Stephen Marron, Michael J. Todd, and Jeongyoun Ahn. 2007. Distance-weighted discrimination. *J. Am. Stat. Assoc.* 102, 480 (2007), 1267–1271.

[97] Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation.* Vol. 24. Elsevier, 109–165.

[98] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled Generative Adversarial Networks. *arXiv:1611.02163.* Retrieved from http://arxiv.org/abs/1611.02163.

[99] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *arXiv:1411.1784.* Retrieved from https://arxiv.org/abs/1411.1784.

[100] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. *arXiv:1802.05957.* Retrieved from https://arxiv.org/abs/1802.05957.

[101] Alfred Müller. 1997. Integral probability metrics and their generating classes of functions. *Adv. Appl. Probab.* 29, 2 (1997), 429–443.

[102] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning.*

[103] Frank Nielsen and Richard Nock. 2013. On the chi square and higher-order chi distances for approximating f-divergences. *IEEE Sign. Process. Lett.* 21, 1 (2013), 10–13.

[104] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems.* 271–279.

[105] Augustus Odena. 2016. Semi-supervised Learning with Generative Adversarial Networks. *arXiv:1606.01583.* Retrieved from https://arxiv.org/abs/1606.01583.

[106] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 2642–2651.

[107] Junting Pan, Cristian Canton Ferrer, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro-i Nieto. 2017. SalGAN: Visual Saliency Prediction with Generative Adversarial Networks. *arXiv:1701.01081.* Retrieved from https://arxiv.org/abs/1701.01081.

[108] Sung Woo Park and Junseok Kwon. 2019. Sphere generative adversarial network based on geometric moment matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19).*

[109] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. 2009. A 3D face model for pose and illumination invariant face recognition. In *Proceedings of the 2009 6th IEEE International Conference on Advanced Video and Signal Based Surveillance.* IEEE, 296–301.

[110] Ben Poole, Alexander A. Alemi, Jascha Sohl-Dickstein, and Anelia Angelova. 2016. Improved Generator Objectives for GANs. *arXiv:1612.02780.* Retrieved from https://arxiv.org/abs/1612.02780.

[111] Guo-Jun Qi. 2017. Loss-sensitive Generative Adversarial Networks on Lipschitz Densities. *arXiv:1701.06264.* Retrieved from https://arxiv.org/abs/1701.06264.

[112] Zhaofan Qiu, Yingwei Pan, Ting Yao, and Tao Mei. 2017. Deep semantic hashing with generative adversarial networks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 225–234.

[113] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434.* Retrieved from https://arxiv.org/abs/1511.06434.

[114] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative Adversarial Text to Image Synthesis. *arXiv:1605.05396.* Retrieved from https://arxiv.org/abs/1605.05396.

[115] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.* 40, 2 (2000), 99–121.

[116] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *arXiv:1606.04671*. Retrieved from https://arxiv.org/abs/1606.04671.

[117] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*. 2234–2242.

[118] Tim Salimans and Durk P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*. 901–909.

[119] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*. 2990–2999.

[120] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. 2018. How good is my GAN? In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 213–229.

[121] Jiaming Song and Stefano Ermon. 2019. Bridging the Gap between f-GANs and Wasserstein GANs. *arXiv:1910.09779*. Retrieved from https://arxiv.org/abs/1910.09779.

[122] Nasim Souly, Concetto Spampinato, and Mubarak Shah. 2017. Semi supervised semantic segmentation using generative adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision*. 5688–5696.

[123] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. 2009. On Integral Probability Metrics,ϕ-divergences and Binary Classification. *arXiv:0901.2698*. Retrieved from https://arxiv.org/abs/0901.2698.

[124] Lucas Theis, Aäron van den Oord, and Matthias Bethge. 2015. A Note on the Evaluation of Generative Models. *arXiv:1511.01844*. Retrieved from https://arxiv.org/abs/1511.01844.

[125] Matteo Tomei, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2018. Art2Real: Unfolding the Reality of Artworks via Semantically-aware Image-to-image Translation. *arXiv:1811.10666*. Retrieved from https://arxiv.org/abs/1811.10666.

[126] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. 2018. MoCoGAN: Decomposing motion and content for video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1526–1535.

[127] Alan M. Turing. 2009. Computing machinery and intelligence. In *Parsing the Turing Test*. Springer, 23–65.

[128] Mehmet Ozgur Turkoglu, Luuk Spreeuwers, William Thong, and Berkay Kicanaoglu. 2019. A layer-based sequential framework for scene generation with GANs. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

[129] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[130] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems*. 613–621.

[131] Kunfeng Wang, Chao Gou, Yanjie Duan, Yilun Lin, Xinhu Zheng, and Fei-Yue Wang. 2017. Generative adversarial networks: Introduction and outlook. *IEEE/CAA J. Autom. Sin.* 4, 4 (2017), 588–598.

[132] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8798–8807.

[133] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. 2018. ESRGAN: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision Workshop*.

[134] Zhengwei Wang, Graham Healy, Alan F. Smeaton, and Tomas E. Ward. 2020. Use of neural signals to evaluate the quality of generative adversarial network performance in facial image generation. *Cogn. Comput.* 12, 1 (2020), 13–24.

[135] Zhengwei Wang, Qi She, Alan F. Smeaton, Tomas E. Ward, and Graham Healy. 2020. Synthetic-neuroscore: Using a neuro-ai interface for evaluating generative adversarial networks. *Neurocomputing* 405 (2020), 26–36.

[136] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. 2017. GP-GAN: Towards Realistic High-resolution Image Blending. *arXiv:1703.07195*. Retrieved from https://arxiv.org/abs/1703.07195.

[137] Yuanbo Xiangli, Yubin Deng, Bo Dai, Chen Change Loy, and Dahua Lin. 2020. Real or Not Real, That Is the Question. *arXiv:2002.05512*. Retrieved from https://arxiv.org/abs/2002.05512.

[138] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. 2018. An Empirical Study on Evaluation Metrics of Generative Adversarial Networks. *arXiv:1806.07755*. Retrieved from https://arxiv.org/abs/1806.07755.

[139] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. 2017. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6721–6729.

[140] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. 2016. Object contour detection with a fully convolutional encoder-decoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 193–202.

[141] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1040–1050.

[142] Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, and Minh N. Do. 2017. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5485–5493.

[143] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE International Conference On Computer Vision*. 2849–2857.

[144] Yuichi Yoshida and Takeru Miyato. 2017. Spectral Norm Regularization for Improving the Generalizability of Deep Learning. *arXiv:1705.10941*. Retrieved from https://arxiv.org/abs/1705.10941.

[145] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. Lsun: Construction of a Large-scale Image Dataset Using Deep Learning with Humans in the Loop. *arXiv:1506.03365*. Retrieved from https://arxiv.org/abs/1506.03365.

[146] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. 2018. Generative Image Inpainting with Contextual Attention. *arXiv:1801.07892*. Retrieved from https://arxiv.org/abs/1801.07892.

[147] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

[148] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*. Springer, 818–833.

[149] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention Generative Adversarial Networks. *arXiv:1805.08318*. Retrieved from https://arxiv.org/abs/1805.08318.

[150] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 5907–5915.

[151] Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based Generative Adversarial Network. *arXiv:1609.03126*. Retrieved from https://arxiv.org/abs/1609.03126.

[152] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative visual manipulation on the natural image manifold. In *Proceedings of the European Conference on Computer Vision*. Springer, 597–613.

[153] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-image Translation Using Cycle-consistent Adversarial Networks. *arXiv:1703.10593v6*. Retrieved from https://arxiv.org/abs/1703.10593v6.

[154] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. 2017. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*. 465–476.

[155] Wentao Zhu, Xiang Xiang, Trac D. Tran, and Xiaohui Xie. 2016. Adversarial Deep Structural Networks for Mammographic Mass Segmentation. *arXiv:1612.05970*. Retrieved from https://arxiv.org/abs/1612.05970.

Received November 2019; revised September 2020; accepted November 2020