Image Generation with Convolutional Neural Networks

Hjalte M. R. Mann
Section for Biodiversity and Conservation
Aarhus University
Aarhus, Denmark
mann@bios.au.dk

***Abstract*—The field of image generation has developed fast since the first introduction of Generative Adversarial Networks (GANs) six years ago. Solving the intrinsic drawbacks in the GAN architecture has received huge attention and many improvements have been suggested to the initial design. At the same time, new network types, such as the Convolutional Adversarial Autoencoder (CVA) have been suggested. These networks solve some of the fundamental issues with the GANs, but come with their own trade-offs. Collectively the field has seen significant improvement in recent years and the most sophisticated image generating models today can produce images of photorealistic quality. Herein, I review the fundamental structure of the GAN, demonstrate the use of a GAN and a CVA for image generation, and lastly present some of the the state-of-the-art image generating networks.**

I. Introduction

The production of realistic images using generative modeling has received a lot of attention in recent years and the performance of such models have improved immensely since the intriguing first presentation of the Generative Adversarial Network (GAN) framework by Ian Goodfellow in 2014 [1]. Later the use of deep convolutional GANs (DCGANs) was described [2], which has served as a basis for most subsequent GANs [3]. The GAN is without doubt the most commonly used framework for generative modeling of images. The basic premise is simple: Two convolutional neural networks, a generator and a discriminator, are set up in a competing fashion in order to train the generator to produce realistic samples. The framework has been modified and adjusted for countless fields and applications. However, despite the ingenious architecture there are some intrinsic issues to the design of the GAN, which cause them to suffer from instable training and mode collapse and lack of variation in the produced output. Other model types have been put forward that solve some of these issues, for example the Convolutional Variational Autoencoder (CVA), but each method comes with trade-offs and limitations on its own. The great amount of research focused on developing and advancing the state of image generative models means that the models of today can produce impressive photorealistic images indistinguishable from real images for the human eye (fig. 1). In this paper I review the fundamental architecture and technicalities of the GAN along with the alternative CVA. I present the functionality of these two types of networks

Fig. 1. Photorealistic images generated by a conditional GAN (from [4]).

Fig. 2. Examples of images of hand-written digits from the MNIST database (from [5]).

by demonstrating the production of realistic renders of hand-written digits based on the MNIST dataset. Finally, I review the state-of-the-art in image generation, including an alternative to the GAN and VA frameworks.

II. The MNIST Dataset

The MNIST dataset is a collection of handwritten digits commonly used as a baseline for developing and evaluating pattern recognition and generation methods [5] (available at: http://yann.lecun.com/exdb/mnist/). The dataset is split into 60.000 training images and 10.000 testing images.

The dataset is very commonly used for developing and benchmarking image generative models and will function as

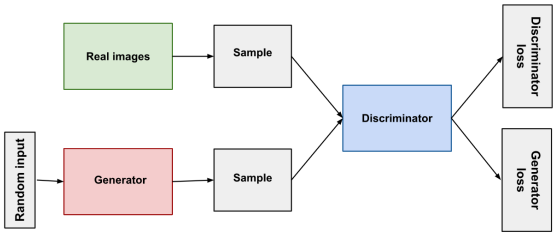

Fig. 3. Schematized architecture of a Generative Adversarial Network (from the Google Machine Learning Course, available at "https://developers.google.com/machine-learning/gan/gan_structure").

the basis for the demonstration of generative models in this review. However, as discussed later, there exist much more complicated datasets and for state-of-the-art generative models it is common to test on a combination of the MNIST dataset and one or several of the more complex datasets.

III. Generative Adversarial Networks

A generative model is one that can either estimate the probability distribution of a set of samples or that can generate new samples from the distribution. In the field of machine learning in particular, the generative model learns the features of a training dataset to the extent where it can generate new data instances that plausibly could have been sampled from the distribution of training data. The GAN is a model framework designed to optimize the generative model. It is essentially a two-player Minimax game between a generator and a discriminator, and the two models are trained simultaneously. While the generator produces a new data instance from a random noise input, the discriminator estimates the probability that the sample has been drawn from the training data distribution versus the generator and determines it as real or fake. The generator attempts to optimize the probability that the discriminator makes a mistake and thereby learns to produce samples that could plausibly come from the distribution of training data. Simultaneously, the discriminator learns to distinguish between samples from the data distribution and generator distribution as it is fed with positive samples from the training data and negative samples from the generator. The competition drives both models to optimize until the generator can fool the discriminator. The model has reached convergence when at the equilibrium where the discriminator can no longer distinguish between training data and generated data and therefore has 50% accuracy). At this point the GAN has learnt an estimate of the probability distribution of the training data and can now produce samples that are not part of the original training data but correspond well to it. As apparent, the basic structure of the network is quite simple, but it has been modified and adjusted for a very wide range of specific use cases (Fig. 3.

A. Training a GAN

In training of a GAN, the components of the network, the generator and the discriminator, are trained simultaneously. As the generator becomes better at producing realistic samples, the discriminator becomes better at distinguishing between real and fake images. An important artifact of the way GANs are designed is that as the generator learns to produce plausible samples the classification by the discriminator becomes more random and less useful for updating the generator. This means that a GAN may move away from convergence as it is trained beyond the point of random output from the discriminator (accuracy of 0.5). In order to avoid overfitting and to reduce the risk of mode collapse (discussed later), training must be balanced between the two networks.

1) The discriminator: The discriminator is a binary classifier trained by supervised learning, with training samples classified as either fake or real, depending on if the source is the generator or the training data. The discriminator is connected to both the discriminator loss function and the generator loss function, but the generator loss is not used during discriminator training. The discriminator is penalized from the discriminator loss function when it classifies a sample incorrectly. The discriminator weights are updated by backpropagation.

2) The generator: The generator is penalized from the generator loss when it produces a sample that is classified as fake by the discriminator. The gradients propagate through the discriminator to the generator to update the generator weights, but the discriminator weights are kept constant while the generator is training. In practice this means that training is alternating between the generator and the discriminator. To prevent overfitting, training is usually set to optimize the discriminator for k steps for each one step of optimizing the generator, where k is a hyperparameter set by the user. This procedure keeps the discriminator close to its optimal solution as long as the generator is not changing too fast.

B. Applications of GANs

GANs have been applied to a very wide range of problems, recent ones including removing rain from images [6], adding haze to images [7], and production of graphical layout proposals [8]. Naturally, many variations to the original framework have been suggested, optimizing the method for specific problems. One such variation is the CycleGAN [9]. This network can learn to perform image-to-image translation based on training on two sets of images without any labels or predefined pairing of images from the two sets, e.g. translating an image of a summer-landscape into a winter-landscape, an image of horse into a zebra, or an image into a rendition of a famous painter. Maziarka et al. [10] introduced the Mol-CycleGAN, a modified version of the CycleGAN adapted to generate molecular compounds optimized for specific parameters, a tool that could be deployed in drug development. Another modification of the CycleGAN is the Identity-Aware CycleGAN, optimized for translating photos of faces to sketches and vice versa with high accuracy by adding a perceptual loss function to the generator training [11].

Conditional GANs are trained on labelled images, thereby allowing the user to specify a label for the output of the gen-

erator [12]. An example of a conditional GAN is the House-GAN, a recently presented tool for generating floorplan layouts for houses based on a GAN constrained by bubble diagrams containing high-level architectural designs [13]. The Stack-GAN is another impressive example of a conditional GAN which can produce realistic images from text descriptions [14]. Production of high resolution images using the basic GAN architecture is difficult because the high dimensional pixel space is hard to capture, a problem that worsens with increasing image resolution. This means that the generated samples lack the diversity of training data distribution. The StackGAN method address this issue by dividing the task into two stages. The first stage generates a primitive, low resolution render of an object based on a text description as input. The second stage takes the output of the first stage along with the text description as input and generates a highly detailed, high-resolution final image.

As proposed in the original presentation of the GAN framework [1], GANs have been found to be very useful in the setting of semi-supervised learning as they can alleviate the need for large amounts of labeled training data [15]. The premise for semi-supervised learning is that the model is trained on a large amount of training data, but of which only a small subset is labelled. By including unlabeled data in the training, performance of the model is increased compared to a model trained only on the labeled dataset. In a standard unsupervised use of a GAN, the discriminator is used for teaching the generator the distribution of the training data. Once the model is trained, the generator can be used to produce new data instances without the discriminator. In semi-supervised training of GAN, a multiclass discriminator can be trained with the use of the generator. Once the model is trained, the discriminator can function as a classifier without the generator. In practice, the discriminator is trained to k+1 number of classes, where k is the number of real classes and the added class is for classifying the fake images from the generator. For a given input, the discriminator outputs a k+1-dimensional vector of logits, from which the probability of the input being fake or the probability of the input being real and belonging to each of the k classes can be derived. Using a semi-supervised learning approach a GAN could produce digits indistinguishable from the real MNIST digits for humans with only a small fraction of the training data being labeled [16]. Examples of the results produced by this network is shown in figure 4).

GANs also have relevant use in supervised learning and have been proposed as an alternative to traditional augmentation of training data when training a neural network. The GAN can be trained on available training data and can then be used to produce additional data, increasing the total size of the training dataset for e.g. an image classification model. Such an approach was demonstrated for the task of insect pest identification where an improved model was obtained by the use of GAN produced training data compared to classic data augmentation [17].

Naturally, many applications of GAN frameworks have focused on some form of image production. However, GANs have also been put to use in problems not involving images. Examples from the audio domain include the GAN-TTS that can generate high-fidelity natural sounding speech from text and the GANSynth, a modified Progressive GAN that is capable of synthesizing high-fidelity audio [18], [19].

Fig. 4. Ooutput of a GAN trained by semi-supervised training (from [15]).

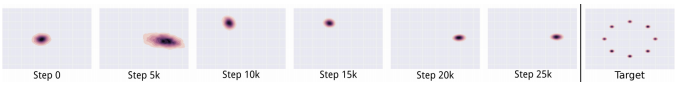

Fig. 5. Demonstration of GAN mode collapse. The standard GAN was trained on a circular arragned dataset consisting of 8 Gaussian mixtures. The network focuses on a single mode of the data and never learns the full distribution (modified from [20]).

C. Advantages and disadvantages of the GAN framework

The GAN framework has some computational advantaged compared to other generative modelling approached such as Deep Directed Graphical models and Generative Autoencoders [1]. However, there are also drawbacks to the GAN framework. One such disadvantages is the risk of mode collapse, sometimes referred to as the Helvetica scenario [1], either complete mode collapse or more commonly partial mode collapse. Mode collapse is a scenario where the generator fails to produce a wide range of outputs, but instead focuses on learning only a subset of the full data distribution. As the generator attempts to find the best output for fooling the discriminator, the production of a very good sample may lead the generator to focus on only this mode of the data. The generator ends up producing just this output as it becomes trapped at a local minimum. This issue was demonstrated by training a standard GAN on a two dimensional dataset consisting of 8 Gaussians mixtures arranged in a circular fashion [20]. Figure 5 shows the networks inability to capture the full target distribution, as it rotates through each single mode of the data.

The problems of mode collapse entails some limitations to the practical applicability of GANs. Several approaches for alleviating the issue have been proposed. One such approach is the use of unrolled optimization of the discriminator [20], which implements a modified loss function that prevents the generator in optimizing for one discriminator. Evolutionary GANs have recently been proposed as a way to alleviate the problems of mode collapse by selecting well-performing individuals from a competing population of generators [21]. Another approach is the already mentioned StackGANs that have been shown to produce very wide outputs [14]. New approaches to solving the intrinsic problems with the GANs are continually being put forward and the performance of the GAN networks have been developing fast, manifested by the impressive results achieved with state-of-the-art networks [22], [23], [24].

IV. Generating hand written digits with a DCGAN

In order to demonstrate the functionality of a DCGAN I used a Google Colab implementation of a DCGAN built with tensorflow (available at: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb).
The code for the implementation can be found in the supplementary material (sup. 1). The dataset for the model is the MNIST dataset presented above. The generator in this implementation uses a dense layer to upscale the random noise input to the correct output image size and transposed convolutional layers to generate the output. A tanh activation function is used for the final layer and Rectified Linear Unit (ReLU) activation function is used for the other layers. The discriminator does the image classification based on a simple CNN composed of two transposed convolutional layers, each with a ReLU activation function and a dropout function, a flattening layer, and a dense layer. The discriminator loss is quantified with a cross-entropy loss function. Each prediction as a real image is compared to 1, giving the loss for real image prediction, and conversely each prediction as a fake image is compared to 0, giving the loss for fake image prediction. The models are optimized using an Adam optimizer, although an optimizer is defined for each network separately in order to train them in an alternating fashion. I trained the model for 200 epochs with a buffer size of 60.000 and a batch size of 256. Results from the network at specific time points during training can be seen in figure 6. After about 50 epochs the digits generated by the GAN are comparable to those of the MNIST dataset. Training beyond 100 epochs do not seem to add obvious quality to the results. Despite high similarity, I would argue that for a human evaluator the outputs from the network are generally distinguishable from the real MNIST digits.

V. Convolutional Variational Autoencoders

Recently the likelihood-based Convolutional Variational Autoencoders (CVAs) have been proposed as an alternative to GANs that avoid some of the common problems with the GANs, mode collapse and lack of diversity in the output, as

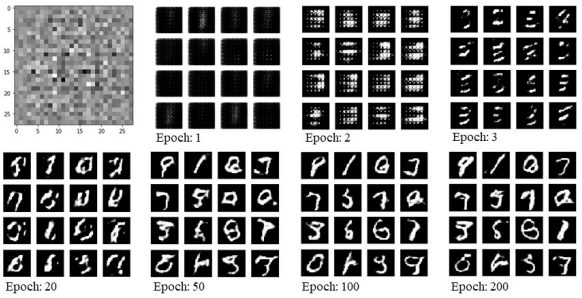

Fig. 6. Randomly sampled outputs from the DCGAN at specific time-points during training. The top left image shows an example of a random noise input for the generator.

discussed above. An autoencoder consists of two connected networks, an encoder that can accept an input and convert it to a smaller, compact version, and a decoder that can convert the downscaled version back to the original input. By defining the encoder-decoder pair as neural networks, the framework allows for iterative optimization of the encoder-decoder scheme using training by gradient descent. A Variational Autoencoder is a regularized form of autoencoder. The regularization of the training suppresses overfitting and optimizes the latent space for the generative process. The CVA is trained by feeding it an input that is encoded a distribution in the latent space. A random sample is drawn from this distribution and is reconstructed by the decoder. The reconstruction error is then backpropagated through the network. The training is based on the optimization of the negative log-likelihood of the training data, in practice by maximizing the evidence lower bound (ELBO) using gradient descent. In contrast to GANs, this gives some benefits in that the models do not suffer from mode collapse and lack of output diversity. However, the CVAs suffer from problems of their own, and the choice of the best image generating model depends on the specific task [24].

VI. Generating hand written digits with a CVA

As with the demonstration of the DCGAN, I have used an implementation of a CVA to generate hand-written digits based on the MNIST dataset (available at: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/cvae.ipynb). The code for the implementation can be found in the supplementary material (sup. 2). Both the generative network and the inference network are based small convolutional neural networks. The generative network is three-layered, consisting of two convolutional layers, using a ReLu activation function and a fully-connected layer. The input for the generative network is a latent encoding and the output generated is a set of parameters for a conditional distribution of the observation. The inference network consists of four layers, a fully-connected layer and three convolution transpose layers, the first two of which uses a ReLu activation function while the last has no activation function. The input for the inference network is an observation and the output is a set of parameters for the conditional distribution of the latent representation. I

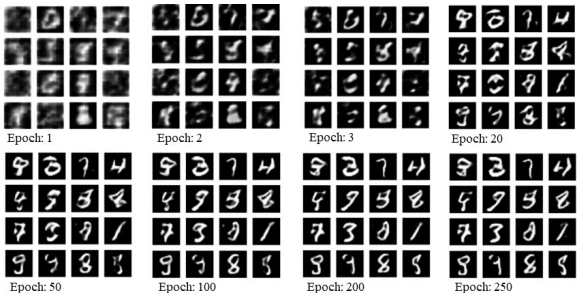

Fig. 7. Randomly sampled outputs from the CVA at specific time-points during training.

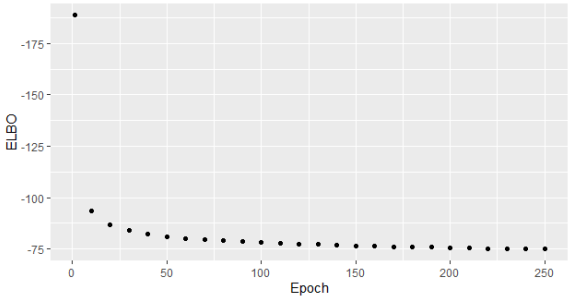

Fig. 8. Evidence lower bound values during training of the CVA. The metric reaches a plateau early in the training and little improvement is made to the quality of the generated samples after this.

trained the model for 250 epochs with a buffer size of 60.000 and a batch size of 100. Results from the network at specific time points during training can be seen in figure 7. Within the first 20 epochs the model learns to produce the sharp black edges of the images.

Similar to the results from the DCGAN, the generated images starts to resemble the hand-written digits at around 50 epochs of training. At 200 epochs a few of the samples are very similar to the real digits, for example the “3” in the first row, second column and the “8” in the fourth row, third column are difficult to distinguish from real hand-written digits. Other digits, e.g. the sample in the first row, fourth column seem to stand out as obvious fakes. There is little improvement in the quality of the generated samples from around 100 epochs and forward. This corresponds well with the plateauing ELBO value as shown in figure 8.

VII. State of the Art in Image Generation

Although the MNIST dataset is still commonly used for testing and benchmarking generative models, it is considered a simple dataset, and state-of-the art generative models work on much more complicated dataset, such as the CIFAR-10 and CIFAR-100 (consisting of 60.000 color images belonging to 10 or 100 classes, respectively) [25], and the FFHQ dataset containing 70.000 high resolution images of human faces [26]. The intrinsic flaws of the basic architecture of a GAN has received huge attention and many methods for improving the network have been suggested. For example, progressive GANs have shown stabilized training, reduced the risk of

Fig. 9. Examples of samples generated by the StyleGAN2 showing high image quality and high variation in the output (from [22]).

mode collapse and improved quality of the outputs [27]. At the time of introduction, this network achieved record performance on the CIFAR-10 dataset. The later introduced StyleGAN [26], also a progressive GAN, along with a recently presented improvement to the model [22], the StyleGAN2 has set the bar even higher. The model architecture of the StyleGAN2 is characterized by a generator including a special normalization layer, the adaptive instance normalization and an alternative to progressive growing that benefits from increasing resolution in training images during training without the change in network topology that is necessary in ordinary progressive GANs. The network is capable of generating highly varied outputs of very high image quality, as visualised in figure 9.

An interesting approach to generating high-quality images is taken in the design of the COCO-GAN [28]. Inspired by the way humans construct a view of their surrounding environment from partial perceptions, the model processes only pathces of full images. The coordinate system of the full image is included in the training and as the generator learns to produce high-quality image parts, the discriminator evaluates whether adjacent patches fit well together without visible seams. Apart from producing state-of-the-art results at inference, even though full images are never produced during training, some interesting applications for image generating models are suggested. These include production of panorama images by using a cylindrical coordinate system, and extrapolation of images by generation of images patches outside the original border of the image. The extrapolation is done on the learned coordinates of the original image and the additional image contents are generated. Examples of the extrapolation technique on images of bedrooms are shown in figure 10.

Fig. 10. Examples of image extrapolation generated by the StyleGAN2. The red borders show the limits of the original image. (from [28]).

VIII. Conclusion

The field of GANs and CVAs is still quite young, but with intense focus from the scientific community, it has seen immense development in only a short period of time. A lot of work is being put into optimizing the performance and applicability of both GANs and VAs and new network designs with significant improvements are continually being put forward. Recently it was even proposed to use the two network types in conjunction. Using the weights from a trained VA as a starting point for the generator in a GAN, thereby effectively pretraining the generator, has some beneficial effects, such as stabilizing the training of the GAN, reducing mode collapse and faster convergence [29].

At the same time, alternative network architectures are also being put forward that may compete with both GANs and CVAs. A new type of generative model design has been proposed that generate samples based on the Stein score, an estimate of the derivative of the log density function of the data distribution [30]. This model does not require sampling during training or use adversarial methods for model optimization. This design has some advantages compared, including flexible network architecture and that the underlying training objective is suitable for comparing different models using the same dataset. The model produces results on the MNIST and CelebA dataset comparable to GANs and even state-of-the-art results on the CIFAR-10 dataset.

An active research field on its own is the challenge of evaluating the outputs of different image generative models. In some cases the performance of a model is evaluated by quantifying the ability of human evaluators to distinguish between real and fake image [31]. However, this method has been found to be sensitive to the group of people that is scoring the samples [15]. A more objective score that has been found to correlate well with how humans annotate an image is the Inception Score [15]. The score is a metric for both output variation and "objectness" of an image, meaning how much the image resembles a real object. However, as the score is based on the Inception image classifier, there is an intrinsic limit to which objects it can evaluate. Also, there may be cases where the classifier cannot detect or distinguish features that seem relevant for a subjective concept of image quality. A newer objective metric for generated image quality is the Fréchet Inception Distance that scores generated images by comparing their statistics to those of real image [32]. As the performance of image generating models increase so will the need for sophisticated metrics for scoring and comparing models. The image generative methods are being applied in an impressively wide range of fields and as the models get better, the possible applications for the technology widens, as illustrated by the image extrapolation method discussed above. There is little doubt that the field of image generation will continue to produce impressive results.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 3, no. January, pp. 2672–2680, 2014.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–16, 2016.

[3] I. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," 2017.

[4] A. Brock, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," pp. 1–35, 2019.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, no. November, pp. 1–46, 1998.

[6] H. Zhang, V. Sindagi, and V. M. Patel, "Image De-raining Using a Conditional Generative Adversarial Network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8215, no. c, pp. 1–1, 2019.

[7] J. Xiao, J. Zhou, J. Lei, C. Xu, and H. Sui, "Image Hazing Algorithm Based on Generative Adversarial Networks," *IEEE Access*, vol. 8, pp. 22 884–22 892, 2020.

[8] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "LayoutGAN: Synthesizing Graphic Layouts with Vector-Wireframe Adversarial Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8828, no. c, pp. 1–1, 2020.

[9] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *Proceedings of the IEEE international conference on computer vision*, pp. 183–202, 2017. [Online]. Available: http://link.springer.com/10.1007/978-1-60327-005-2{_}13

[10] Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel, and M. Warchoł, "Mol-CycleGAN: A generative model for molecular optimization," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–19, 2020. [Online]. Available: https://doi.org/10.1186/s13321-019-0404-1

[11] Y. Fang, W. Deng, J. Du, and J. Hu, "Identity-aware CycleGAN for face photo-sketch synthesis and recognition," *Pattern Recognition*, vol. 102, 2020.

[12] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," pp. 1–7, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[13] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation," 2020. [Online]. Available: http://arxiv.org/abs/2003.06988

[14] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Realistic Image Synthesis with Stacked Generative Adversarial Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1947–1962, 2019.

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[16] A. Odena, A. Odena, and G. Com, "Semi-Supervised Learning with Generative Adversarial Networks," pp. 1–3, 2016.

[17] C. Y. Lu, D. J. Arcega Rustia, and T. T. Lin, "Generative Adversarial Network Based Image Augmentation for Insect Pest Classification Enhancement," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 1–5, 2019. [Online]. Available: https://doi.org/10.1016/j.ifacol.2019.12.406

[18] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," *ICLR*, pp. 1–14, 2019. [Online]. Available: https://magenta.tensorflow.org/datasets/nsynth

[19] J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High Fidelity Speech Synthesis with Adversarial Networks," pp. 1–15, 2019.

[20] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, pp. 469–477, 2016.

[21] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 921–934, 2019.

[22] T. Karras and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN."

[23] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," 2017.

[24] A. Razavi, "Generating Diverse High-Fidelity Images with VQ-VAE-2."

[25] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[26] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," Tech. Rep., 2019.

[27] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–26, 2018.

[28] C. H. Lin, C. C. Chang, Y. S. Chen, D. C. Juan, W. Wei, and H. T. Chen, "COCO-GAN: Generation by parts via conditional coordinating," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 4511–4520, 2019.

[29] H. Ham, T. J. Jun, and D. Kim, "Unbalanced GANs: Pre-training the Generator of Generative Adversarial Network using Variational Autoencoder," 2020. [Online]. Available: http://arxiv.org/abs/2002.02112

[30] S. Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution," no. NeurIPS, 2019.

[31] E. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 1486–1494, 2015.

[32] M. Heusel, L. G. Jan, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," no. Nips, 2017.