GANs for Simulatiom Representation and Inference

becominghuman.ai • Ari Heljakka

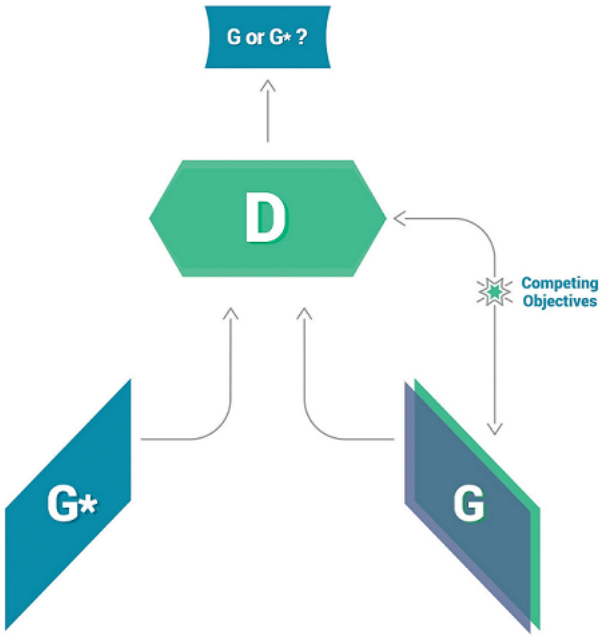
(Update 2018/10/15: Note that this review covers the models only up to 06/2017. On the whole, the analysis is still valid, but of course, there have been further improvements.)

Over 100 variants of GANs (Generative Adversarial Networks) were introduced in 2017 alone. To understand this development, we select a subset of these to observe some of the major axes of the variation, and examine GANs from the perspectives of simulation, representation and inference. The reader is expected to have general understanding of neural networks and at least superficial familiarity of GANs.

Image interpolation with Boundary Equilibrium GAN. The real training samples at far left and far right. [1]

Generative neural network models have recently aroused much interest, primarily for two reasons. First, trained on a certain dataset, they can learn to generate synthetic sample data that exhibits the salient features of the original data. Second, there is a hypothesis that, in comparison to other learning tasks, the generation task requires (and perhaps guides) the network to learn a relatively powerful internal representation of the data. To this end, the network must compactly capture the underlying factors that explain most of the variation in the data. Correspondingly, the actual task to generate (or transform) content can be considered either an end in itself, or a *measure* of the power of the internal representation. This distinction is often not made explicit, but it is a fundamental one.

Generative Adversarial Network (GAN) variants are networks that learn complex data distributions with an internal representation that allows generating novel realistic samples that resemble the training data. So far, they have been found most applicable for various image generation and transformation tasks, with unsupervised and semi-supervised learning.



Here, we define the term "**GAN pattern**" to stand for the idea that underlies all GAN variants. GAN pattern is similar to what is often called "adversarial training", but it also involves a specific architecture constraint. The architecture requires two trainable networks G and D and a non-trainable data provider G*, where D takes input from both G and G*. The learning goal of D includes developing and maintaining sensitivity to the factors that distinguish between the samples from the data sources G and G*. The goal of G includes the opposite of this – hence the term 'adversarial'. G is called *generator* and D *discriminator* or *critic*. If balanced carefully, D provides sufficient gradient to guide G towards creating samples that are increasingly G*-like, until the output of G becomes indistinguishable from G*.

Further, we invoke a simple concept that has intuitive meaning across many disciplines – **simulator**. A simulator imitates another system by creating sample events, states or structures based on some computable rules we come up with – e.g. approximations to the laws of nature, your theory of human relationships, or the latest crazy mesh of deconvolutional networks. In the GAN pattern, we can say that G learns to *simulate* G*. GAN pattern is, then, a specific way to *learn* a simulator, in the form of its *generator*.

GAN variants (including the original GAN [2]) implement the GAN pattern in one or more ways. Often, the data from G* is static training data such as images, but it can be any distribution of random variables. Note that G is intended to mimic the *data generating process* G*, not to reproduce the exact same data that G* produces during training. We can then utilize the synthetic data samples from G directly, or use the internal *representation* of the network for unsupervised and semi-supervised learning. Now, importantly, GAN pattern alone does not do any *inference* from input variables. However, we will see how to extend G to take such input.

Major recent GAN applications include e.g. generating images with increasingly high resolution, image-to-image translation from one domain to another without labeled image pairs, image classification with only a small number of labeled samples [3], and transforming existing imagery e.g. by way of super-resolution [4] or by refining a synthetic "toy image" into a more realistic one [5]. There is an increasingly wide variation of applications that falls outside the scope of this article, as well as those intended for supervised or semi-supervised (class-conditioned) scenarios (e.g. StackGAN which in some respects has been state-of-the-art for image generation), domain adaptation models, and models with discrete variables. GANs are also connected to reinforcement learning (RL). In e.g. inverse RL, the learning task involves recovering the reward function itself, much like learning the discriminator in GAN.

Major recent theoretical development around GANs include the analysis of the GAN convergence problem ([6], [4]), generalizing GANs to the wider class of f-divergences [7], and connections to implicit models ([8], [9]). Other important theoretical work includes the analysis of evaluation methods ([10], [11]), extensions towards two-sample testing ([12], [13]), and making GANs amenable for likelihood analysis [14].

Major limitations of GANs have, until recently, been usually related to the stability of training and the lack of diversity of generated samples. Several improvements were introduced by [3], followed by e.g. various modifications to the loss function, supporting the training with an autoencoder, and adding noise to the distributions in particular ways. At present, perhaps the most pressing problem of GAN research is the **lack of sufficient standard performance benchmarks**. Even though GANs are customarily treated as probabilistic models, the likelihood of a sample under the learned model, in particular, appears to be unsuitable as a GAN metric. We will look at various alternatives.

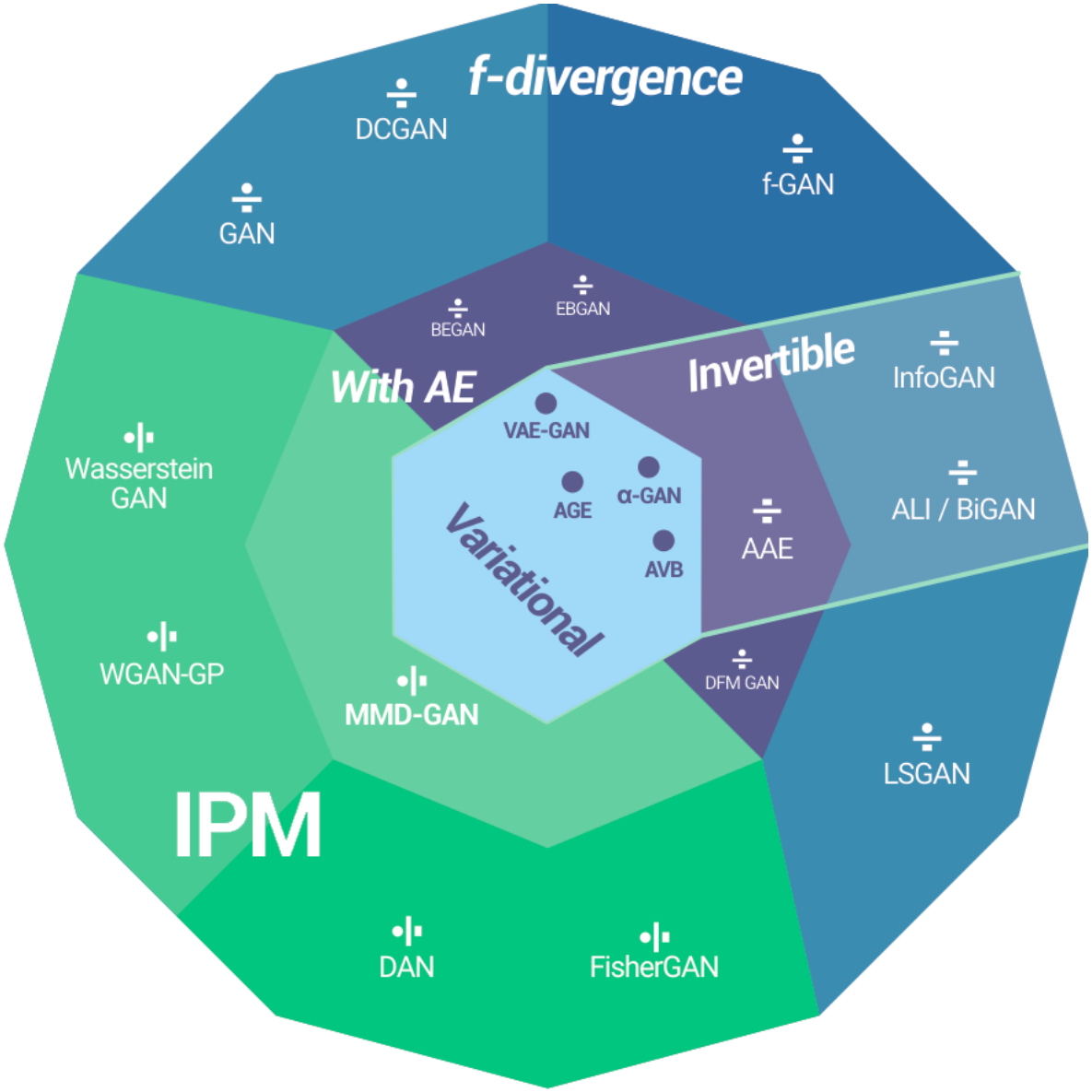
In the following, we will first establish the major axes of variation between GAN variants, and then proceed to look at them from the perspective of simulation, representation and inference. **Simulator learning** is identified as the general goal of the GAN pattern, and we look at it separately in terms of loss functions and architecture. If you only need to generate random samples for another downstream purpose, this may be enough. If your goal is **representation learning** for e.g. downstream semi-supervised learning, you might expect a good simulation to imply a good underlying representation. However, we need to cover a range of measurement approaches to understand what that means. Finally, if you need to guide or control[1] the generation process, or condition it on a

partial or full sample as an input to the network, we need **inference**. This enables applications that process, transform or complete an input sample in some way.

Navigating the GAN Variant Space

We start with a map of **some**[2] of the most important generic GAN variants published by the end of H1/2017. Over 20 variants are included, primarily from the perspective of unsupervised learning.

Confusingly, the contemporary *GAN nomenclature mixes together GAN variants that only change the loss function, ones that also change the architecture, ones that support some kind of an inference, and ones with application-specific extensions or changes in the training algorithm. Even ignoring the applications, this calls for a map that allows for at least three levels.



LEGEND

GAN	Generative Adversarial Network (Goodfellow, 2014), (Salimans, 2016)
DCGAN	Deep Convolutional GAN (Radford, 2015)
f-GAN	f-GAN (Nowozin, 2016)
Wasserstein GAN	Wasserstein GAN (Arjovsky, 2017)
WGAN-GP	Improved Wasserstein GAN (Gulrajani, 2017)
MMD-GAN	MMD-GAN (Li Chun-Liang, 2017)
FisherGAN	FisherGAN (Mroueh, 2017)
InfoGAN	Information Maximizing GAN (Chen, 2016)
BiGAN	Bidirectional GAN (Donahue, 2017)
ALI	Adversarially Learned Inference (Dumoulin, 2017)
EBGAN	Energy-Based GAN (Zhao, 2017)
BEGAN	Boundary Equilibrium GAN (Berthelot, 2017)
AAE	Adversarial Autoencoder (Makhzani, 2015)
AVB	Adversarial Variational Bayes (Mescheder, 2017)
AGE	Adversarial Generator-Encoder Network (Ulyanov, 2017)
VAE/GAN	VAE/GAN (Larsen, 2015)
α-GAN	α-GAN (Rosca, 2017)
DFM-GAN	Denoising Feature Matching GAN (Warde-Farley, 2017)
LSGAN	Least Squares GAN (Mao, 2016)
DAN	Distributional Adversarial Network (Li Chengtao, 2017)

The top-level distinction (the green and blue halves of the figure) is between the two prominent **loss behavior families** (f-divergences vs. Integral Probability Metrics). Loosely, these GAN variants fall on one side or the other. The second-level distinction (the outer and inner rings) is based on whether the architecture of the GAN variant has been extended with an **autoencoder**. Next, please bear in mind that the GAN pattern as such can only learn a mapping from latent space to generated samples. The third-level distinction, then, is made based on whether the GAN variant has been extended to allow for **inverting** the sample generation process with an inference network that maps from samples back to latent space. Note that this can be done with or without an autoencoder. Finally, we separate the variants that invert the generator with **variational** inference.

Crucially, this map provides a *bottom-up view*, in the sense that we have started from the GAN variants that are out there, and crafted a map to accommodate them. A more insightful approach, however, is to take a *top-down view* of what the GAN variants are expected to do, and justify the distinctive virtues of each GAN variant on this basis. This view brings us to simulation, representation, and inference.

Simulator Learning — The Overarching Goal

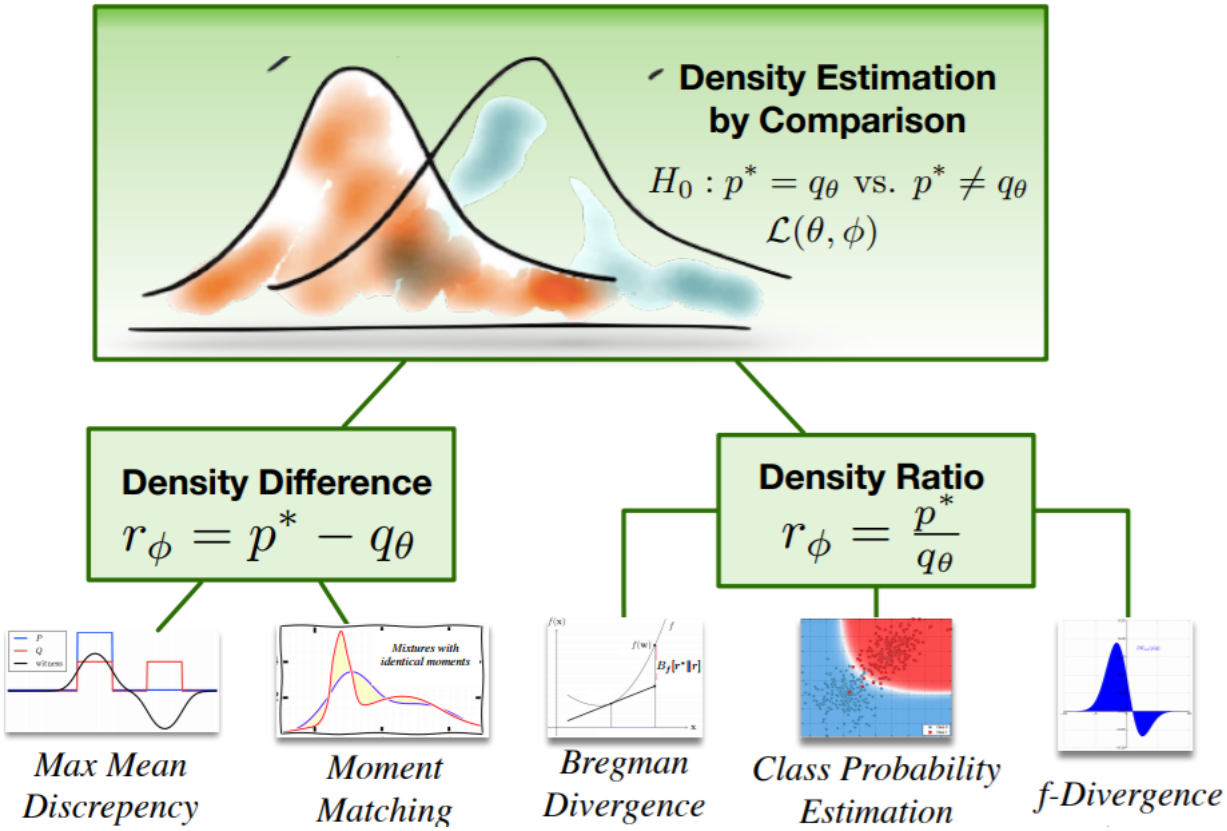
We mentioned that the GAN pattern is a specific way to learn a simulator. Now, we will show how simulators can be connected to probability density in rather surprising ways. We do this to explain two separate things: why the GAN loss functions have recently split into two directions, and how we can use the GAN pattern for probabilistic inference.

Simulated faces from CelebA by Alpha-GAN. [15]

As the GAN generator begins to learn, it becomes a simulator that favors the stochastic generation of certain kinds of samples over the others. Thus, **it behaves as if it used a specific probability distribution for the generated samples**, but implicitly (unless of course the simulator was defined according to explicit probability rules).

In a neural network model, we may have several distributions to consider, such as the prior distribution of the latent variable and the likelihood of the observed variable under the model. Under certain conditions, a GAN pattern can be used to turn any or all those distributions implicit![3]

Such a distribution cannot be used directly for calculating probabilities. Luckily, it turns out that often we need just a *ratio* of two distributions, and in that case, one or both can be implicit. We will just have to resort to the GAN discriminator, built for comparing two distributions. So, instead of dealing with the distributions *as distributions* in a closed form, we compare the population of generated samples of each distribution as they come. We can use this to estimate either *density ratio* or *density difference* of these populations. Correspondingly, there are two popular approaches for GAN loss functions to implement this **density estimation-by-comparison** ([7], [9], [16]). (For succinctness, our explanation is extremely simplified.)



A more precise breakdown of Density Difference and Density Ratio estimation methods [9].

(1) f-divergence measures the ratio of probability densities between the two distributions. f-divergences include Jensen-Shannon divergence, Kullback-Leibler divergence, and others. For two distributions p(x) and q(x), an f-divergence is, effectively, an expectation of their density ratio:

$$\int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

With a suitable *choice* of f, this equation can directly reproduce the said divergences, and more. In GANs, the distributions p and q are typically the real and the generated one. Now, **GAN loss function can either *converge* into f-divergence behavior via *class probability estimation*, or use it explicitly**. For instance, the original GAN loss function has no mention of f-divergences, but under certain conditions, it was famously shown to converge to Jensen-Shannon divergence. Similarly, e.g. LSGAN [43] has been shown to converge to Pearson chi-square divergence. The conditions under which these training algorithms actually *do* converge are a topic of intense research. Alternatively, we can make any f-divergence our *explicit* loss measure by ***divergence minimization***, as demonstrated in f-GAN [7].

(2) Integral probability metrics compare expectations of the distributions under a suitable *learnable* smooth transformation function f (called *witness*) that aims to find the biggest differences between the distributions:

$$\sup_{f \in \mathsf{F}} \left| \int_M f \, d\mathbb{P} - \int_M f \, d\mathbb{Q} \right|$$

Varying the F, this form produces e.g. *1-Wasserstein distance* (aka Earth Mover distance), *Maximum Mean Discrepancy (MMD) distance*, as well as the *feature matching loss* of [3]. *Moment matching* looks at the higher moments of the distributions of this form, either explicitly or via use of embedding kernels. Next, we will see examples of all of these!

Simulator Learning — Loss Function Perspective

The loss function formulation of the original GAN has been found challenging to work with. It was supported later by the set of Improved GAN Techniques [3] that provided generic tools for GAN training while also showing good results in semi-supervised scenarios. Still, problems remained.

At least one major cause appears to be **dimensional misspecification** ([6], [4], [17]) that troubles most GANs of the f-divergence family. To learn the generator properly, the GAN discriminator needs to provide it with a sufficiently stable gradient. But in GANs, the generated data and the real data behave as if they were two distinct relatively low-dimensional manifolds, with very little initial overlap between them. It can be shown that in this setup, gradients based on e.g. JS divergence or KL divergence will easily become overtly sharp or vanishing (due to the log probability ratios), and cannot point the gradient descent process to the right direction. As soon as the discriminator becomes good enough, it can use the gap between the distributions for perfect discrimination, preventing the generator from improving.

As a solution, [6] offered the 1-Wasserstein distance, leading us to **Wasserstein GAN (WGAN)** [18]. WGAN was a major a milestone due to its convergence and stability properties that were unparalleled at the time of its introduction. It makes a simple but major shift in the objective function. **Instead of making the discriminator compare something like contrastive log probabilities, it now compares something like quality scores.** Technically, the scores give rise to an approximation of 1-Wasserstein distance, using the fairly involved theory of Optimal Transport. The theory looks for the smallest amount of changes required to move enough mass from one distribution to another, to make them equal. Unlike previous GANs, WGAN showed stable training convergence that clearly correlated with increasing quality of generated samples. Most of us can skip the complex theory of WGANs, and just keep in mind that the WGAN removes the logarithms of the loss function and aggressively clips the weights of the discriminator to keep the network on a stable learning path. This clipping reduces the learning capacity of the network, but the problem was fixed in a follow-up variant **WGAN-GP** [19].

WGAN led us down the IPM track, soon accompanied with methods based on **Maximum Mean Discrepancy (MMD)** distance, descending from Generative Moment Matching Networks [20]. By operating on populations of samples rather than individual ones, they tend to offer robustness at the cost of performance. Generative Matching Networks utilized fixed kernels for measuring distances between distributions. **MMD-GAN** [21] and **Distributional Adversarial Networks** [22] improve upon this by making those kernels learnable with adversarial setup. **FisherGAN** [23] uses a variance control scheme based on Mahalanobis distance, to the purported effect of achieving the stability of IPM-based training with performance superior to WGAN and WGAN-GP.

To overcome the dimensional misspecification, [4] proposed "instance noise" as a solution to increase the overlap between the distributions, while remaining on the f-divergence track. To the same effect, [17] offers a cleaner method of adding the noise analytically – with a **penalty on the weighted gradient norm**. This regularization term can be added to almost any GAN network, potentially with major stability gains without increasing variance.

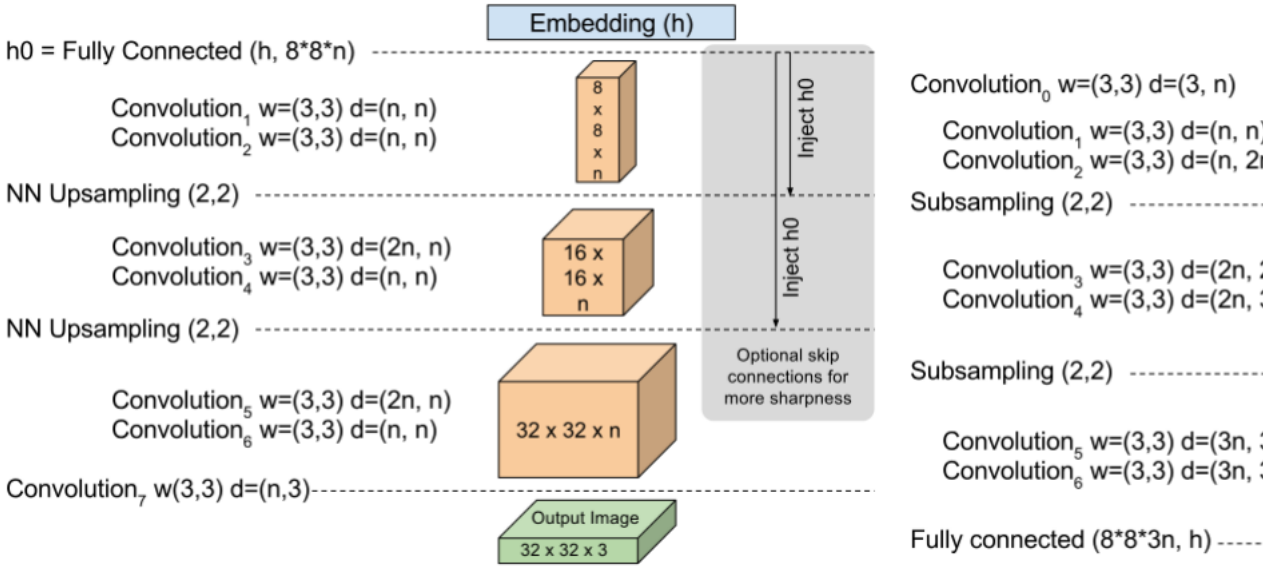
The **distinction between IPM and f-divergence** objectives should be taken with a grain of salt, however. First, the f-divergence behavior of the original GAN objective is only achieved under the assumption of a perfect discriminator. Almost all GAN variants change the objective in some way, and might no longer share this convergence property, and even when they do, the practical relevance of this is not clear. Second, different IPM variations behave in very different ways. For instance, the use of the second moment in MMD methods makes the scaling quadratic with respect to the training batch sizes, unlike the first moment used in Wasserstein. Empirically, the practical effect might be tolerable ([12], [21]) but more research is clearly needed. Third, we can have a setup like FisherGAN which utilizes an IPM-based loss function, but at the non-parametric limit is shown to converge to chi-squared divergence (which is no longer IPM).

The **major drawback of WGAN and WGAN-GP is the inferior performance** in comparison to f-

divergence models. Nonetheless, they have already powered a number of applications, and their performance can be improved. An early example is the "fast-learning" Wasserstein critic [14]. At the moment, an alternative approach may be to use the generally less stable but faster f-divergence GANs by stabilizing them with new techniques like the penalty on the weighted gradient norm.

Simulator Learning — Architecture Perspective

Since the introduction of the original GAN architecture, **DCGAN** [24] is by far the most often cited implementation for image generation, and remains a reference architecture or baseline for the recent models. DCGAN successfully combined a convolutional architecture with GAN and was able to produce images with compelling quality.



Architecture of Boundary Equilibrium GAN [1], extending the original DCGAN.

Autoencoders have been introduced to stabilize the training. To see why this is, bear in mind that an autoencoder takes a sample, encodes it to latent format, and then tries to decode it back into the original sample, like a compression/decompression utility such as jpeg. Now, for the autoencoder to succeed, it needs to make the decoded sample as similar to the input sample as possible, i.e. we minimize the difference between the two, a measure known as *reconstruction loss*. This enables the network to learn even in the absence of adversarially generated samples.

Autoencoders can, of course, be used in very different ways. For instance, Boundary Equilibrium GAN (BEGAN) [1] uses a particular equilibrium enforcing method to balance the training and control the trade-off between sharpness and diversity of generated images. DFM-GAN [25] improves stability by training a denoising autoencoder to denoise the feature representations of real data, and matches the features of the real and generated data only after denoising both, achieving high Inception score (more on which later). Energy-Based GAN [26] uses a regular autoencoder to a similar effect.

The critical part of any DCGAN-style architecture for high-resolution image generation is, of course, the CNN. Therein lies a question. As such, the CNN part could of course be used with other generative models. **What is, then, the specific contribution of the GAN pattern for the results?** How do we know that GANs are not just one successful generative method for piggybacking on the massive opportunities created by CNNs? Surprisingly little attention has been given to this fundamental question, so [27] attempted to find out exactly this. With a generator network devoid of adversarial training component, they produce results with quality high enough to at least warrant further research. If this approach turns out to produce increasingly good results with no adversarial training at all, this could imply that, in the context of image generation, "GAN" as such is not the most interesting level of abstraction to focus on!

Representation Learning

By now, we have covered some candidates for architecture and loss functions that may provide a quick and stable way for training our GAN variant. Next, we can proceed to measure the model's ability to *represent* the data. How much can the model actually learn, and exactly what can it actually generate?

We can start from likelihood estimation, but we will not get far. In a GAN pattern, the generator is presumed to learn a rich representation of the original data distribution. Traditionally, such models have been measured via the likelihood of some test data under the learned model. Now, the GAN pattern does not immediately lend itself for likelihood evaluation, but methods have been developed to that end.[4] Examples include Annealed Importance Sampling [11] and analyses based on Real NVPs, such as [14]. The results, however, are confusing. When e.g. the latter method was used to train an increasingly powerful generator with WGAN, the measured log-probability density actually *decreased*. Other approaches ([11], [10]) have also showed that likelihood of validation data under the model is not necessarily correlated with the quality of the generated samples. The papers [28] and [29] show that the generator of GAN can "win" the discriminator even though it actually converges to a distribution that, at best, takes into account only a part of the training distribution.

From this, various conclusions are possible. On possibility is that **we have simply confirmed the difference between representation and probability estimation**. Producing a "more representative" sample is simply a different thing than producing a "more probable" sample. A network that can produce a variety of different sharp pink elephants must have a fairly good representation of elephants, yet the probability of its creations remains zero. In general, should it learn the features that can be combined to reproduce any sample in the training distribution, or the joint probability density of the actual combinations of those features? We clearly need to step up the game and try something else.

Fortunately, instead of likelihood estimates, we can **use the generative ability of the system as a proxy for its learning ability**. That is, if you can generate something, you prove that you have learned something about it. Hence, we can focus on measuring the *quality* and *diversity* of samples that the network can generate, *within and across classes*, with respect to the *sampling speed*.

Less fortunately, **manual comparison by humans** is the most popular method *actually* used to evaluate the generated images for quality and realism. This is obviously insufficient due to various inaccuracies. It is especially hard for humans to evaluate the *diversity* of generated images.

Inception score is the most established quantitative measure of GAN performance [3]. It uses a pre-trained Inception model to measure two things. First, for each generated sample to have an unambiguous class, we want *low entropy* for **p(y|x)**. Second, for classes to be evenly represented, we want *high entropy* for

$$\int p(y|\boldsymbol{x} = G(z))dz$$

These two conditions combine into

$$\exp(\mathbb{E}_{\boldsymbol{x}}\mathrm{KL}(p(y|\boldsymbol{x})||p(y)))$$

However, Inception score does not account for diversity *within* a class. E.g. if the most specific matched class is 'beagle', then your score will not improve by actually making more than one kind of a beagle image, resulting in mode collapse. In order to measure *diversity within each class*, **multiscale structural similarity (MS-SSIM) score** [30] has been used by [15] and [31], but does not directly work across classes in unsupervised setup.

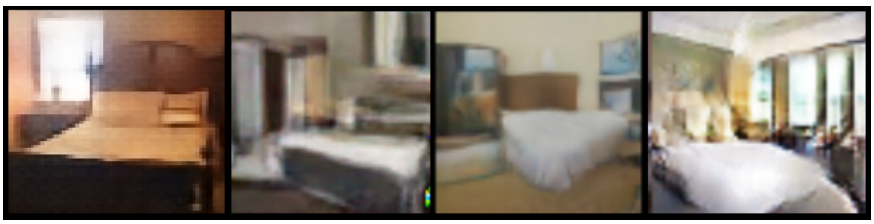
Training a separate evaluation network with IPM-based methods has recently been introduced as a powerful generic evaluation measure. Wasserstein distance could provide a fair measure of sample quality as well as detection of overfitting and mode collapse. [14] trains a **separate Wasserstein critic** with validation data, uses it *for the evaluation* of the trained GAN, and could further use it to compare different GAN architectures. In

addition, **Classifier 2-sample testing (C2ST)** [13] and **a separate MMD estimator** [12] can serve as fairly advanced sample quality evaluation approaches that also promise human-interpretability. Note that [12] extends the idea to GAN *training*, too.

Note that, up to now, we have stayed in the context of representation learning *before* any downstream task, which is known to be a limiting setup for evaluating any unsupervised method, since ultimately different tasks would benefit from learning different kinds of things. By adding any auxiliary task with its own performance measure, models are rendered comparable. It is, then, the pragmatic measure of task-specific performance, not a universal objective evaluation method, that gives us a fixed reference point. **Semi-supervised learning** can provide one such measure for generative models, e.g. in the form of classification performance, used effectively by [3].

Finally, while GAN is a general-purpose network model, most of the GAN work so far has focused on **image generation**, to the unfortunate effect that we do not yet know how well the analyses done in the image domain will hold across domains.

Still, at least within the image domain, we can see several paths that may lead us to more established standards and benchmarks. Fortunately so, since staring at two generated bedrooms and trying to convince yourself of the existence of a meaningful discernible difference between them is a particular purgatory which nobody would equate with good science or engineering.



Bedrooms created with different architectures. Can you spot the one produced with no adversarial training at all?

Inference Learning

Moving beyond mere simulation and representation learning, there are several ways to use GAN pattern for *likelihood-free inference*, i.e. to invert the generator.

Strictly speaking, there are indirect ways to **invert any GAN generator and approximately recover the latent code of a sample**. A simple way is to use regression on a fully-trained GAN to find the latent variables that produce the sample that is closest to our target. Recovering the codes allows linear arithmetic in the latent space, and has been used for arithmetic of the kind shown below.

Arithmetic on images in latent space.[24]

Should we require more control, it may suffice to have a subset of latent dimensions be interpretable. This is the approach of InfoGAN, which separates a few information-maximizing latent codes from the others that are treated as incompressible noise.

A more advanced approach is to use the GAN pattern with some kind of a probabilistic autoencoder. In order to get there, we can start from the regular autoencoder, and extend it with two things. First, we would like to sample easily from the latent space of the autoencoder, and in order to do this, we should have our latent space values distributed in a nice manner; for instance, as a normal distribution. This would allow us to generate all of the interesting variation of the samples by using latent values only within a focused narrow range (say, walking from -5 to +5 for each dimension, instead of random guessing or some kind of MCMC approach). Second, we would like to have some notion of "degree of precision" in the input. We need this to express the idea that a certain latent code value range, say the value between 1.161–1.163, corresponds to the same pixel value in the image space. In other

words, we would like to say things like "differences smaller than 0.002 in the latent variable values should not make any difference in the output images".

Variational inference provides us with these tools, and can be combined with autoencoders as **Variational Autoencoder (VAE)** ([32], [33]). Alternatively, there is a way to extend the GAN approach to learn the input and latent variables in a joint manner. This can be done deterministically (BiGAN [34]) or stochastically (ALI [35]).

For VAE, here we can only provide a simplified explanation. VAE loss function maximizes so-called Evidence Lower Bound (ELBO) of the marginal likelihood of the data under the model. ELBO contains three components:

- **Prior latent distribution p(z)** that reflects our chosen target for the distribution of the latent variables,
- **Approximated posterior q(z|x)** that measures the encoding success, and
- **Likelihood p(x|z)** that measures the decoding success.

Specifically, VAE minimizes the sum of KL divergence

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

and cross-entropy

$$-\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\Big[\log p(\mathbf{x}|\mathbf{z})\Big]$$

Hence, we often focus on the **KL divergence term** rather than on q(z|x) and p(z) separately. VAE in itself can generate fairly nice samples for many use-cases.[5]

Car coloring based on a grayscale input image by AGE network. [36]

Now, in order to use GAN pattern with variational inference, recall that we can leverage density estimation-by-comparison to approximate any ratio of two distributions. This *"density ratio trick"* [15] relieves us from the burden of having to figure out an analytical form for both distributions, which typically need approximations that may be detrimental to model performance. Because each of the three ELBO terms is a distribution, we can use the density ratio trick separately to replace any or all of them (with a separate GAN pattern for each). In principle, we will then be protected against any limitations that *explicit approximations* of those distributions would have entailed ([8], [15]).

For GAN-VAE variants, we will cover the Adversarial Generator Encoder Networks (AGE) [36], VAE-GAN [37], Adversarial Autoencoder (AAE) [38], Adversarial Variational Bayes (AVB) [39] and Alpha-GAN [15]. They all implement an autoencoder, and most of them define an extra discriminator.

- AGE replaces the **likelihood** term with an L1 loss and the **KL divergence** term with an empirical approximation, and connects the loss functions of the encoder and decoder (generator) in an adversarial setup.
- VAE-GAN extends VAE with an **extra discriminator** for the data space.
- AVB and AAE (and also AffGAN [4]) modify VAE by replacing the **KL divergence** with a discriminator for the latent space.
- Alpha-GAN further modifies AAE by also replacing the **likelihood term** with a combination of a separate discriminator and L1 reconstruction loss.

Going deeper, the picture gets rather complicated, with various differences between the models. First, while e.g. VAE-GAN, AVB and AGE are strictly variational, AAE is not[6]. Second, ALI and BiGAN are closely related to these

models. While also not strictly variational, they have the benefit of symmetrically learning the mapping not only from x to z but vice versa [8], with the downside that they do not really use the inference model for inference. Empirically, these models are often not able to reconstruct the input image well, but it is too early to say what that really means.

Alpha-GAN has some superior results over the other models, and it uses the density ratio trick in both the likelihood and divergence terms. However, it still needs the L1 reconstruction loss for the explicit purpose of avoiding mode collapse. Also, as a replacement for the divergence term in Alpha-GAN, the AGE-style empirical approximation actually outperformed the discriminator that was based on the ratio density trick, when evaluated on Inception score metric [15]. In terms of architecture, AGE remains the simplest one of these, avoiding separate discriminators altogether.

Conclusion

We have presented one possible map for categorizing recent GAN variants. We illustrated their differences with respect to their ability to learn to simulate, represent and infer.

For simulator learning to succeed in the first place, we observed at least three promising approaches: Complementing **f-divergence GANs** with the **penalty on the weighted gradient norm** [17], **WGAN-GP** with performance improvements from e.g. the **fast learning critic** [14], and MMD-based methods such as **FisherGAN** [23].

For representation learning, the key question is how to measure the representational power. Currently, we can use either a combination of partial metrics, train an auxiliary network to do the evaluation, or measure with an auxiliary task such as classification in semi-supervised learning context. More research is needed to address open questions about the ability of GANs to learn the distribution in the first place, as opposed to learning the representation in a different sense.

For invertible GANs, we have identified several GAN-VAE hybrids that show reasonable results. They mainly differ in how extensively they use the GAN pattern, but it is not yet clear under which conditions "more is better".

Fortunately, there are already several plausible answers to most of our questions. We conclude with an example of how to stabilize training in the case of unsupervised adversarial domain adaptation. The same domain adaptation model was used by CycleGAN [44], DiscoGAN [45], and DualGAN [46]. They all work well, despite the fact that each one uses a different loss function form (CycleGAN uses LSGAN, DualGAN uses WGAN, and DiscoGAN uses the original GAN loss). It may turn out that many different solutions work. But first, we need to standardize the evaluation metrics to find out.

Notes

[1] For semi-supervised learning, we can of course utilize labels to make the whole GAN effectively class-conditional, which has shown good results already, but here we will keep our focus on the unsupervised setup. [Back]

[2] Why only these GAN variants have been included, and not MyFavouriteGAN? We attempted to include most of the GAN variants that are important from the *unsupervised* perspective, published prior to June 30th, 2017.

The variants have been selected as follows. The original GAN, "improved" GAN [3] and DC-GAN[24] are considered reference architectures due to being universally cited by nearly all the GAN papers. The division between f-divergences and IPM variants is here considered a primary one, as defined by [7] and [9]. Due to the particularly many recent papers on MMD methods, the MMD class of papers is included here since the appearance of these papers is a strong signal for the MMD direction (FisherGAN, MMD-GAN, DAN). McGAN is considered less relevant after the introduction of FisherGAN (by the same authors).

Therefore, we include the GAN variants created by the authors of the primary theoretical papers, the GAN variants that those papers quantitatively compare to, and the MMD family. AffGAN is a specialized model, but its stochastic version was shown in the paper to be related to variational inference, and the paper contains theoretical results that overlap with [6]. Generative Moment Matching Network is explicitly not a GAN model. Class-conditional models (StackGAN, PPGN) and discrete models (e.g. Boundary Seeking GAN) were not included. The criteria for considering AAE, ALI and BiGAN as non-variational is based on [8]. f-GAN stands as the canonical example of generalizing GANs from Jensen-Shannon divergence to all f-divergences [7].

You could argue that the subset of models chosen here for examination is too narrow, especially insofar as one assumes that the *applications* of GANs are what matters in the end. Unless stated otherwise, the omission of this or that GAN variant from this article only implies that it did not meet the specific selection criteria at the time of writing. Consequently, outside the sources we cite, the mileage of any and all conclusions may vary. Or, you could argue that our approach is too "wide and thin", since we cover e.g. the GANs with inference but not e.g. the GAN variants for domain adaptation. Finally, one could claim that the level of abstraction is wrong, since treating these models specifically as GANs (as opposed to e.g. the wider class of implicit models) will hinder us from seeing the connections to the other disciplines that tackle very similar problems.

Is it appropriate to call these models "GAN variants"? In the general context, this simplification should be used with care since some of these are hybrid generative models that incorporate several ideas (to the point that e.g. some GAN-VAE hybrids might better be considered "VAE variants"). With the recent flood of network models using *GAN naming scheme, the distinction is not trivial. Even several near-duplicate models have been introduced under different names, e.g. ALI/BiGAN, VAE-GAN/MDGAN, DiscoGAN/CycleGAN/DualGAN. [Back]

[3] This is why GANs, simulators and simulation-based methods are often also called *implicit models* or *implicit generative models*. However, the terminology easily gets messy. Since the network may contain multiple distributions, it would be more appropriate to talk about specific *implicit distributions within the model*, rather than addressing the whole (neural network) model as implicit or not. Just like the network can have two or more sub-networks that separately implement the GAN pattern, there can be two or more distributions that are being (separately) implicitly modelled. Some more terminology: First, a model is sometimes called *prescribed* or *explicit* if it has explicit parameters for observation likelihood (i.e. the likelihood of the data under the model that we try to learn). An implicit model does not have one. Second, note that some authors use the term 'generative model' to mean 'implicit model', which gets confusing since we can have prescribed models that generate samples. Third, note that the whole GAN pattern, with both a generator *and a discriminator*, is usually called 'generative model'. Finally, it is important to bear in mind that other implicit models exist, e.g. the Classifier Approximate Bayesian Computation (ABC) [40]. Classifier ABC has similarities to GANs, although ABC is built for inference, not sample generation. [Back]

[4] Kernel Density Estimates, popular in many domains of statistics, do not work here [10]. "Birthday testing" (estimating entropy by counting coincidences) was used by [29] to find the size of the support that the model distribution has, but it also requires manual human participation. Mode Score from [41] was proposed to improve on Inception score, but it has not been widely adopted. [Back]

[5] Why should we involve GAN pattern at all, then? The typical narrative on this mentions that in image generation tasks, the VAE-generated images appear blurry. (By extension, it is maybe implied but not proven that this would be a problem with other application domains as well.) There is a wide range of hypotheses for why this would be so. Possibly the most convincing one is the following. VAE is derived with KL divergence which is unsymmetrical by definition. Therefore, even though putting probability mass on the parts of q(x,z) that are not shared by p(x,z) is quickly corrected, the reverse does not hold. This may leave enough "ELBO room" for the encoder to assign probability to places where there should be none! Still, there is much unexplored space around VAEs to tackle this problem without GANs, too, and progress has been made. [Back]

[6] Unlike AVB, AAE's discriminator only depends on z (and not on x) and hence violates the requirements of

variational inference. While this approximation might be dangerous, it was recently connected to the minimization of 2-Wasserstein distance between the generative model and data, unlike the AVB approach [42]. [Back]

References

[1] BEGAN: Boundary Equilibrium Generative Adversarial Networks [PDF]
Berthelot, D., Schumm, T. and Metz, L. CoRR, abs/1703.10717, 2017.

[2] Generative Adversarial Networks [PDF]
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. In NIPS, 2014.

[3] Improved Techniques for Training GANs [PDF]
Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. and Chen, X. In NIPS, 2016.

[4] Amortised MAP Inference for Image Super-resolution [PDF]
Kaae Sonderby, C., Caballero, J., Theis, L., Shi, W. and Huszar, F. In ICLR, 2017.

[5] Learning from Simulated and Unsupervised Images through Adversarial Training [PDF]
Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W. and Webb, R. In CVPR, 2017.

[6] Towards Principled Methods for Training Generative Adversarial Networks [PDF]
Arjovsky, M. and Bottou, L. In ICLR, 2017.

[7] f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization [PDF]
Nowozin, S., Cseke, B. and Tomioka, R. In NIPS, 2016.

[8] Variational Inference using Implicit Distributions [PDF]
Huszar, F. arXiv:1702.08235, 2017.

[9] Learning in Implicit Generative Models [PDF]
Mohamed, S. and Lakshminarayanan, B. In ICLR, 2016.

[10] A note on the evaluation of generative models [PDF]
Theis, L., van den Oord, A. and Bethge, M. arXiv:1511.01844, 2015.

[11] On the Quantitative Analysis of Decoder-Based Generative Models [PDF]
Wu, Y., Burda, Y., Salakhutdinov, R. and Grosse, R. In ICLR, 2017.

[12] Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy [PDF]
Sutherland, D., Tung, H., Strathmann, H., De, S., Ramdas, A., Smola, A. and Gretton, A. In ICML, 2016.

[13] Revisiting Classifier Two-Sample Tests [PDF]
Lopez-Paz, D. and Oquab, M. arXiv:1610.06545, 2016.

[14] Comparison of Maximum Likelihood and GAN-based training of Real NVPs [PDF]
Danihelka, I., Lakshminarayanan, B., Uria, B., Wierstra, D. and Dayan, P. arXiv:1705.05263, 2017.

[15] Variational Approaches for Auto-Encoding Generative Adversarial Networks [PDF]
Rosca, M., Lakshminarayanan, B., Warde-Farley, D. and Mohamed, S. arXiv:1706.04987, 2017.

[16] Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation [PDF]
Sugiyama, M. and Kanamori, T. Annals of the Institute of Statistical Mathematics, 2012.

[17] Stabilizing Training of Generative Adversarial Networks through Regularization [PDF]
Roth, K., Lucchi, A., Nowozin, S. and Hofmann, T. CoRR abs/1705.09367, 2017.

[18] Wasserstein GAN [PDF]
Arjovsky, M., Chintala, S. and Bottou, L. CoRR, abs/1701.07875, 2017.

[19] Improved Training of Wasserstein GANs [PDF]
Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A. CoRR, abs/1704.00028, 2017.

[20] Generative Moment Matching Networks [PDF]
Li, Y., Swersky, K. and Zemel, R., 2015. arXiv:1502.02761, 2015.

[21] MMD GAN: Towards Deeper Understanding of Moment Matching Network [PDF]
Li, C., Chang, W., Cheng, Y., Yang, Y. and Poczos, B. arXiv:1705.08584, 2017.

[22] Distributional Adversarial Networks [PDF]
Li, C., Alvarez-Melis, D., Xu, K., Jegelka, S. and Sra, S. arXiv:1706.09549, 2017.

[23] Fisher GAN [PDF]
Mroueh, Y. and Sercu, T. arXiv:1705.09675, 2017.

[24] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [PDF]
Radford, A., Metz, L. and Chintala, S. In ICLR, 2016.

[25] Improving generative adversarial networks with denoising feature matching [link]
Warde-Farley, D. In ICLR, 2017.

[26] Energy-based Generative Adversarial Network [PDF]
Zhao, J., Mathieu, M. and LeCun, Y. In ICLR, 2017.

[27] Optimizing the Latent Space of Generative Networks [PDF]
Bojanowski, P., Joulin, A., Lopez-Paz, D. and Szlam, A. arXiv:1707.05776, 2017.

[28] Generalization and Equilibrium in Generative Adversarial Nets (GANs) [PDF]
Arora, S., Ge, R., Liang, Y., Ma, T. and Zhang, Y. In ICML, 2017.

[29] Do GANs actually learn the distribution? An empirical study [PDF]
Arora, S. and Zhang, Y. In ICML, 2017.

[30] Image quality assessment: from error visibility to structural similarity
Wang, Z. and Simoncelli, E.P. IEEE transactions on image processing, 13(4):600–612, 2004.

[31] Conditional Image Synthesis With Auxiliary Classifier GANs [PDF]
Odena, A., Olah, C. and Shlens, J. In ICML, 2017.

[32] Auto-Encoding Variational Bayes [PDF]
Kingma, D. and Welling, M. arXiv:1312.6114, 2013.

[33] Stochastic Backpropagation and Approximate Inference in Deep Generative Models [PDF]
Jimenez Rezende, D., Mohamed, S. and Wierstra, D. In ICML, pages 1278–1286, 2014.

[34] Adversarial Feature Learning [PDF]
Donahue, J., Krahenbuhl, P. and Darrell, T. arXiv:1605.09782, 2016.

[35] Adversarially Learned Inference [PDF]
Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M. and Courville, A., 2016. CoRR, abs/1606.00704.

[36] It Takes (Only) Two: Adversarial Generator-Encoder Networks [PDF]

Ulyanov, D., Vedaldi, A. and Lempitsky, V. CoRR, abs/1704.02304, 2017.

[37] Autoencoding beyond pixels using a learned similarity metric [PDF]
Boesen Lindbo Larsen, A., Kaae Sonderby, S., Larochelle, H. and Winther, O. In ICML, pages 1558–1566, 2016.

[38] Adversarial Autoencoders [PDF]
Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B. arXiv:1511.05644, 2015.

[39] Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks [PDF]
Mescheder, L., Nowozin, S. and Geiger, A. In ICML, 2017.

[40] Likelihood-free inference via classification [PDF]
Gutmann, M., Dutta, R., Kaski, S. and Corander, J. In Statistics and Computing, 2017.

[41] Mode Regularized Generative Adversarial Networks [PDF]
Che, T., Li, Y., Jacob, A., Bengio, Y. and Li, W. ICLR, 2017.

[42] From optimal transport to generative modeling: the VEGAN cookbook [PDF]
Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C. and Schoelkopf, B., arXiv:1705.07642, 2017.

[43] Least Squares Generative Adversarial Networks [PDF]
Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., Smolley, S. arXiv:1611.04076, 2017.

[44] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks [PDF]
Zhu, J., Park, T., Isola, P., Efros, A. arXiv:1703.10593, 2017.

[45] Learning to Discover Cross-Domain Relations with Generative Adversarial Networks [PDF]
Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J. arXiv:1703.05192, 2017.

[46] DualGAN: Unsupervised Dual Learning for Image-to-Image Translation [PDF]
Yi, Z., Zhang, H., Tan, P., Gong, M. arXiv:1704.02510, 2017.