Generative Adversarial Networks

Jacob Smith

December 5, 2017

Abstract

Deep generative networks have often fallen short, especial when compared to discriminative models. Since their conception in 2014, generative adversarial networks have seen exponential growth in research. These networks offer promising results for both unsupervised learning and generative models. This research paper presents an overview of generative adversarial networks. Specifically, we will examine the motivation behind GANs, their theoretical justification, the current state of GANs and the research frontiers that have yet to be explored.

1 Introduction

Deep learning have made astounding progress within the past decade. Discriminative models have recently surpassed the abilities of human within the domain of pattern recognition [15]. These successes can attributed to vast, high dimension datasets in conjunction with large neural networks using linear activation functions, dropout regularization techniques and backpropagation to update the parameters [3]. However, deep learning possesses many more ambitious goals. Until recently, success has mostly been seen with supervised classifiers; however, deep generative models now competitively rival their discriminative counterparts.

Generative models learn the joint probability distribution $p(x, y)$ of an input x and label y whereas discriminative models directly learn the conditional probability $p(y|x)$. Generative models may be used as classifiers using Bayes rules to calculate the conditional probability $p(x, y)$ which can then be used to make predictions [12]. Depending on the model, knowledge of the probability distribution can be created both explicitly and implicitly [4].

Figure 1: The process density estimation of one-dimensional data and a Gaussian distribution [2]. Generative models take a dataset D, sourced from a distribution p_{data}, a create an estimate of that distribution p_{model}.

Those which do not directly model a probability distribution offer mechanisms which require implicit knowledge of the underlying distribution, such as creating a sample from that distribution [4]. As humans, we are able to understand the world around us with tremendous precision. It is easy to underestimate the complexity of the data we process to accomplish this feat. Although progress within the machine learning field is rapidly advancing, computers still have limited understanding of the data the process. Generative models, especially deep generative models, offer promising results towards this goal [8].

The process of training a generative model is very similar to a discriminative model. Using large amounts of data collected from a specific domain, we train the model to generate data from that domain. As these models have fewer parameters than that the number of data samples, they are forced to internalize some representation of the data. Although, unlike supervised training, there exists no explicit desired output. The problem becomes defining a cost function which forces the generative model to produce data more like that of the domain.

Generative adversarial networks (GANs), first introduced in 2014, offer a new framework for estimating generative models with use of an adversarial process [3]. These models offer a clever approach to solving the aforementioned problem of explicitly defining a cost function. Rather than training a single model, a discriminator is introduced. These two networks are pitted against each other in a minimax game. The generative model G attempts to produce data that resembles that of the training set while the discriminative model D attempts to classify whether or not a sample was real of generated. To train this network, G attempts to maximize the error rate of D whereas D attempts to minimize it. The goal of the GAN algorithm is to reduce the distance between the data and generated probability distributions. Backpropagation is used to update the parameters and to train each model. In

the ideal case, D is unable to distinguish the generated samples from the real samples and produces an error rate of $\frac{1}{2}$ [3].

2 Motivations

There exists several compelling reasons for studying generative modeling [2]. These models are extremely useful when your goal is understand the underlying distribution parameters. For example, joint probability distributions of high dimensional data is relevant to both applied math and engineering [2]. Generative models are also applicable to reinforcement learning, particularly model-based learning algorithms. A time series generative model, one which predicts future states of an environment given the current state of the environment and an agents actions, are of particular interest [2]. An agent may learn the best action to take in a given scenario by querying this generative model and choosing the one which outputs the best state of the world. Furthermore, generative models, especially GANs, have recently excelled at semi-supervised learning.

Proposed in the original GAN paper [3], semi-supervised techniques used in conjunction with GANs have recently proven themselves as a reputable technique. The premise involves training the discriminator to classify images into $n+1$ classes where n is the number of class labels and the additional 1 is the class of a fake image. This algorithm is able to obtain great performance on relatively small labeled datasets when compared to labeled datasets typically state of the art supervised algorithms [2]. Currently, the best performing semi-supervised GAN are feature matching GANs, where the new objective of the generative model is to minimize the distance between a layer l of the discriminator [14].

Many applications require multi-modal outputs. For example, predicting the next action of a self-driving car. Within this context, it is important the model not be trained by traditional methods, such as using mean squared error to minimize the distance between the expected and predicted actions. These models cannot be used in situations where there is more than one appropriate prediction. Generative modeling, especially GAN, allow machine learning to function the scenarios where one input corresponds to multiple acceptable outputs [2].

Tasks which involve image generation, modification or translation are all highly applicable to generative models. Although GANs may be used to

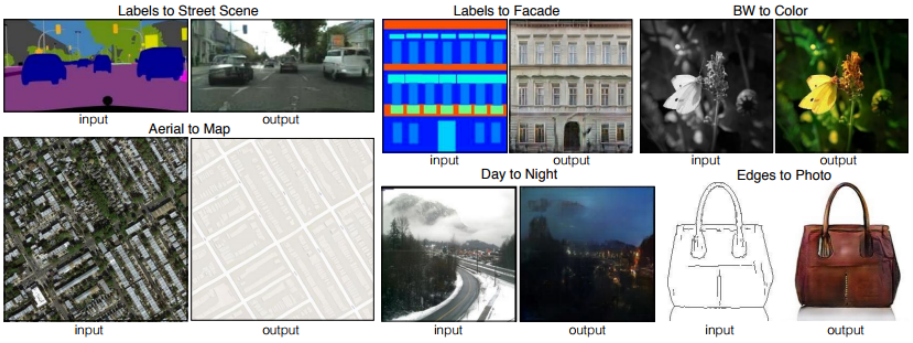

Figure 2: Examples of the image-to-image technique developed by Isola *et al.* [7]. This technique makes use of conditional generative adversarial networks to map across image domains.

generate any type of data, images remain the most commonly used source of training data. Research conducted within the past three years demonstrates the capabilities of GANs to perform these tasks [2, 16, 11]. For example, GANs have recently been shown to be capable of translating aerial photographs into maps and sketches into realistic images [2].

When compared to other generative models, especially deep generative models, GANs possess several advantages... quicker? no Markov chains?

3 Background

Generative adversarial networks may use any differentiable functions for the models; however, the optimal results are typically achieved when neural networks are used [3]. We first define a vector z such that $z\ p_z(z)$. This vector becomes the argument of the generator $G(z;\theta_g)$ where G is a differentiable function and θ_g is its parameters. Another differentiable function $D(x;\theta_d)$ for the discriminator. This function takes as input a vector x, either from the output of G or a sample from the training set, and is parametrized by θ_d. The output of the discriminator is a scalar number representation the probability that the arguments were derived from the training data. In this situation, there exists a minimax game where D attempts to maximize the probability of assigning a correct label and G attempts to minimize $\log(1 - D(G(z)))$. The networks G and D play a minimax game with the value function $V(D, G)$:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}\Big[\log(D(x))\Big] + E_{z\sim p_z(z)}\Big[\log(1-D(G(z)))\Big]$$

For the training step, any variant of gradient descent may be used with respect to the individual cost functions [2]. The first updates θ_D while attempting to minimize the discriminator cost function J_D while the other updates θ_G while attempting to minimize the generator cost function J_G. There exist several variations of the cost function that may be used for training purposes; however, the discriminator cost has remained the same for each.

$$J^{(D)}(\theta_D,\theta) = -\frac{1}{2}E_{x\sim p_{data}(x)}\Big[\log(D(x))\Big] - \frac{1}{2}E_{z\sim p_z(z)}\Big[\log(1-D(G(z)))\Big]$$

This is a standard cross entropy cost function that is trained on two mini-batches, one from the dataset and one from the generator. For this to become a zero-sum game, the cost function of the generator is simply the negation of the discriminator.

$$J^{(G)}(\theta_D,\theta) = -J_d(\theta_D,\theta)$$

This version of the generator loss function is particularly useful for theoretical analysis as shown in the first GAN paper [3]. Goodfellow demonstrated that the minimax game is the same as minimizing the Jensen-Shannon divergence and that each player will eventually converge if both players are able to directly update the function space as oppose to the parameters within the network [2]. However, in this minimax version, the loss function of the generator suffers from a vanishing gradient when the discriminator confidently rejects the generated data. The solution proposed by Goodfellow is to train the generator to maximize $log(D(G(z)))$ rather than minimize $log(1-D(G(z)))$ [3]. This function is heuristically motivated as to provide strong gradients when either the generator or discriminator is not functioning optimally. In this version, the generator attempts to maximize the log probability of a misclassification whereas the minimax version attempts to minimize the log probability of the discriminator being correct.

$$J^{(G)}(\theta_D,\theta) = -\frac{1}{2}E_{z\sim p_z(z)}\Big[\log(D(G(z)))\Big]$$

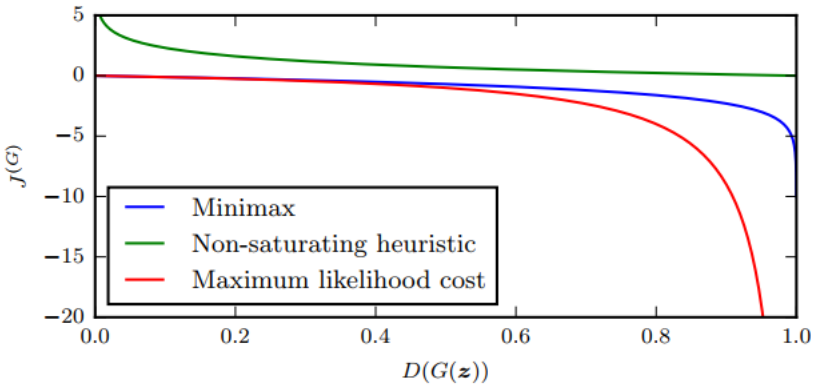

Figure 3: A comparison of the generator cost function for generative adversarial networks. When $D(G(z))$ is close to 0, the gradient is saturated for both the minimax and maximum likelihood cost functions. This situation occurs when the discriminator D confidently rejects the generated value $G(z)$ as fake. Therefore, the non-saturated heuristic cost function is the only one which works well in practice. Furthermore, the maximum likelihood cost function may produce high variance due to the rapid decrease as $D(G(z))$ nears 1. Image retrieved from Goodfellow NIPs tutorial [2].

A maximum likelihood cost is another option for training GANs [2]. It can be shown that that this cost function minimizes the Kullback-Leibler divergence between p_{data} and p_{model} instead of the Jensen-Shannon divergence. This cost estimator is useful as it allows for comparison with other generative models; however, it behaves similarly to the minimax cost function as seen in 3.

$$J^{(G)}(\theta_D, \theta) = -\frac{1}{2} E_{z \sim p_z(z)} \Big[\exp(\sigma^{-1}(D(G(z)))) \Big]$$

This following algorithm presents initial training algorithm used by Goodfellow *et al.* when GANs were introduced in 2014 [3]. Many modifications have been made by various sources. The first step in the algorithm is to optimize the discriminator D. Initially, many thought it would be optimal to train D to completion; however, as previously mentioned, this is still a controversial topic among researchers [2]. For the initial GAN experiments, this number was set to 1 to reduce computation time.

n: The number of training iterations
k: The number of discriminator iterations
for *training iteration in* n **do**
  for *discriminator iteration in* k **do**
    sample m vectors $\{z^{(1)}, ..., z^{(m)}\}$ from distribution $p_z(z)$;
    sample m vectors $\{x^{(1)}, ..., x^{(m)}\}$ from the training samples;
    update the discriminator using its stochastic gradient

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log(D(x^{(i)})) + \log(1 - D(G(z^{(i)})))]$$

  end
  sample m vectors $\{z^{(1)}, ..., z^{(m)}\}$ from distribution $p_z(z)$;
  update the generator using its stochastic gradient;

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z^{(i)})))]$$

end

Algorithm 1: The initial generative adversarial network algorithm developed by Goodfellow *et al.* in 2014 [3]. Multiple variations of this algorithm have since been produced. One of the benefits of adversarial networks is that they may be updated using back propogation.

4 Issues

There exist several issues that characterize generative adversarial networks. Some are related design of GANs or their underlying network implementations. These issues lay at the forefront of GAN research.

4.1 Instability

Generative adversarial networks are often characterized by their instability [2]. The training process makes use of gradient descent which does not guarantee convergence to the Nash equilibrium, the game theory equilibrium that involves two competing players. This is different than network optimization which is common in discriminative models. When each network trains, even if the specified network improves, the other may regress.

This sometimes leads to the scenario where each network repeatedly undo's the other's progress. This problem is not specific to GANs, but to any game situations [2]. It was shown in the first paper that convergence was guaranteed if updates could be made to the function space, but no theoretical justification was been found that guarantees convergence within the parameter space [3]. However, there is no theoretical proof that that these games should not converge [2]. Typically, adversarial networks often oscillate between different kinds of samples without converging to an equilibrium in a phenomenon called mode collapse.

4.2 Mode Collapse

Mode collapse occurs when the generator collapses several different latent space vectors z map to the same outputs. Full mode collapse is rare; however, partial mode collapse always occurs and varies by intensity. For example, the generator may create two images that only differ by colour but were generated with very different latent vectors. This occurs when the capacity of the generator is superior to that of the discriminator. Backpropagation does not necessarily follow the minimax game mentioned earlier and may behave like the maximin version.

$$G^* = \max_D \min_G V(F, D)$$

When this occurs, the generator attempts to map every latent vector to the one x variable the discriminator is most likely to guess is real. When training, one can only hope the behavior resembles the minimax version. It is often speculated that the root cause of the issue the particular divergence measure used. Specifically, it is often asserted that mode collapse is caused by the use of Jensen-Shannon divergence; however, the use of Kullback-Leibler divergence has be shown to worsen the problem [2]. As mode collapse is currently an unavoidable reality for GANs, their applications are limited to situations where it is acceptable to generate only a small number of distant samples [2]. A potential solution to this problem is discussed in the following section.

Figure 4: An example of mode collapse as demonstrated by Metz *et al.* [10]. The generated samples cluster around only one of the modes and do not spread out as they should. Throughout the training process, this cycle continues does not reach an equilibrium.

4.3 Image Quality

When generating images, problems often arise due to the generators difficulties with perspective, counting and structure [2]. For example, an generated animal may not have a 3D structure, have misplaced limbs or have too many of a specific body part. These issues arise due to the underlying model architecture and are not an inherent problem the adversarial network. Furthermore, generated image resolution has remained relatively low; however, recent improvements outlined in the following section show improvements with regard to resolution [9].

4.4 Evaluation

The evaluation procedure for generative adversarial networks remains an open debate as there exist no clear quantitative measurement to evaluate the models [2]. For example, models with high likelihood may produce poor samples and models with low likelihood may produce realistic samples. Several variations of the GANs examine the tradeoffs of divergence measures other than the original Kullback-Leibler divergence [17]. It can be shown that different metrics favor different models. Therefore, it is important to asses one's model within the context of the whole application [17].

4.5 Discrete Data

GANs assume a model implementation that is differentiable due to it's use of back propagation. Therefore, this restricts their ability to generate discrete

data. Goodfellow proposes several solutions such as algorithm modifications, discretization and the use of concrete distributions or Gumbel-softmax [2].

5 Training Improvements

A substantial amount of research has been conducted within the past three years with regards to improving GAN training procedures. A paper released by Goodfellow approximately one year after the debut of GANs outlines several techniques that can be applied to improve the training process [14]. As previously mentioned, the optimal solution for GANs remains convergence of the Nash equilibrium of the minimax game; however, gradient descent attempts to find a local minima. The following techniques introduced by Goodfellow *et al.* address the instability of GANs and attempt to increase the probability of convergence.

5.1 Feature Matching

Feature matching involves the definition of a new cost function for GANs such that they do not overtrain on the current discriminator [14]. Instead of optimizing on the output of the discriminator, the new objective requires the generator to minimize the difference between the expected value on some intermediate layer. The objective function can be described as $\left|\left|E_{x\sim p_{data}(x)}\Big[f(x)\Big] - E_{z\sim p_z(z)}\Big[f(G(z))\Big]\right|\right|_2^2$ where $f(x)$ is the output of the activation function of some layer. Experimental evidence has been gathered that demonstrates the usefulness of this technique, especially when the training process becomes instable [14]. Furthermore, feature matching is useful for semi-supervised GAN applications as will later be described in section 7.3.

5.2 Minibatch Discrimination

Minibatch discrimination attempts to minimize the issues caused by mode collapse. This problem occurs as the discriminator processes each value x independently of each other value. The gradient of each will point in the same direction thus leading to mode collapse. This technique involves the use of a minibatch in place of a single value to allow to discriminator to examine multiple data examples at one time. The discriminator is then able

to examine the distance between each sample in conjunction with the samples themselves and look for abnormal minibatch distributions. Successful application of this technique allows for visually appealing images to be generated much quicker than by previous training techniques [14]. Preliminary results also show great reduction of mode collapse issues [2].

5.3 Historical Averaging

Historical averaging introduces an extra term for each player's cost function such that they are penalized when choosing new parameters that differ from the historical mean value [14]. This new term $||\boldsymbol{\theta} - \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\theta}[i]||^2$ where t is the total number of iterations performed and $\boldsymbol{\theta}[i]$ is the value of the parameters at iteration i. This technique has been demonstrated to help low dimensionality games reach an equilibrium where normal gradient descent has failed [14].

5.4 Label Smoothing

Label smoothing was a technique first introduced in the 1980s which involves replacing the 0 and 1 targets with *smoother* values such as 0.1 and 0.9 respectively [14]. This technique is used to improve the networks resistance to adversarial examples. Specifically, it is recommendended to use one-sided label smoothing by using 0.9 label instead of 1 for the real data. By altering the label for fake data, this may lead to problems when $p_{data}(x)$ nears 0 and $p_{model}(x)$ becomes large for some sample x [14].

5.5 Virtual Batch Normalization

Batch normalization is a technique which involves applying normalization for each training minibatch. This allows for much higher learning rates, more stable training and reduces problems associated with parameter initialization [6]. This normalization technique was shown to successful for Deep Convolutional GANs; however, it causes a dependence relationship between the samples of the minibatch [14]. When implementation virtual batch normalization, a reference batch of samples is randomly chosen at the start of training and used throughout to normalize the current sample. However, this technique is computationally expensive as it requires forwarding two minibatches.

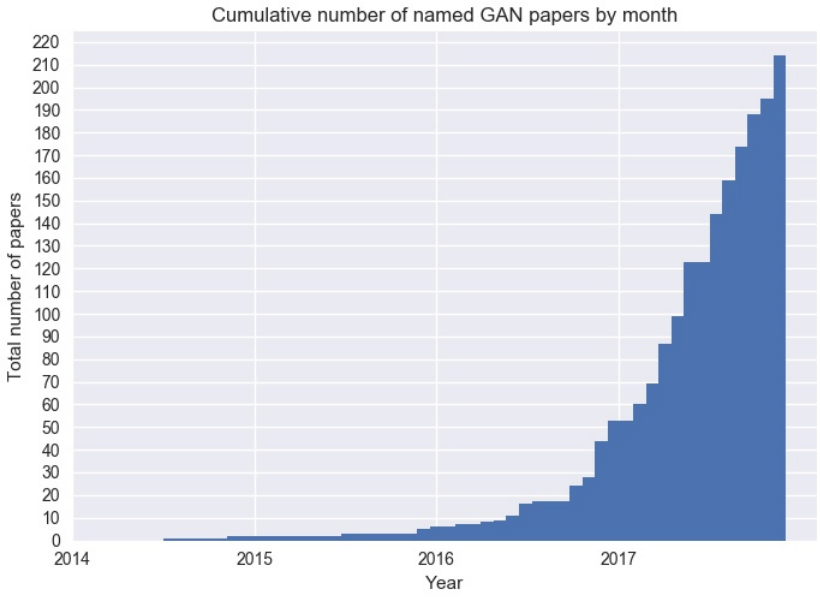

Figure 5: The cumulative amount of papers written about generative adversarial networks. The graph shows the exponential growth of GANs within the past three years. Image maintained by Avinash Hindipur [5].

6 Architecture Variations

Research within the field of deep generative modeling has increased exponentially since the debut of generative adversarial networks in 2014. Significant improvements are being made with respect to image quality and training stability. Hundreds of papers have been released since 2014 as seen in figure 5 which propose numerous architecture variations.

6.1 Fully Connected GANs

The first generative adversarial network paper by Goodfellow *et al.* made use of relatively simple fully connected neural networks for the generator and discriminator models. These are networks where each neuron in one layer is connected to every neuron in the following layer. This design is able to generate simple images and useful for proof of concept; however, several

other architectures have overtaken this implementation [1].

6.2 Convolutional GANs

Although convolutional neural networks (CNNs) seem well suited for the image generation tasks associated with GANs, difficulties associated with training initially constrained the use of convolutional models for both the discriminator and generator [1]. When implementing standard CNNs, one is often is unable to generate the same level of capacity and representational power. One solution proposed by Denton *et al.* made use of Laplacian pyramids (LAPGAN), a type of image representation which repeatedly smooths and downsamples an image forming a sort of pyramid of images. At each layer of the pyramid, a conditional convolutional GAN is trained given generator output of the above layer. Another solution to this problem was proposed by Radford *et al.* simply made alterations to the existing deep convolutional network architecture [13]. Max pooling is replaced with strided layers within the discriminator and fractionally-strided layers within the generator. Furthermore, LeakyReLUs are used for all layers of the discriminator in place of the standard ReLU non-linearity function. The fractionally strided convolutions replicate within the generator replicate the benefits of the progressive upscaling within the Laplacian pyramid. The strided layers allow both the generator and discriminator to learn both the down-sampling and up-sampling operators during training [1]. Their architecture, called DCGAN, has become the base model for most GANs today [2].

6.3 Conditional GANs

Initially proposed by Goodfellow as a straightforward extension of the first GAN architecture [3], Mirza *et al.* successfully implemented the conditional generative model in late 2014 [11]. Conditional generative models are constructed by feeding the additional parameter $\boldsymbol{c}$ to both to generator $G(\boldsymbol{z}|\boldsymbol{c})$ and discriminator $D(\boldsymbol{x}|\boldsymbol{c})$ to *condition* the GAN. These architectures are better suited for multi-modal output [1]. By conditioning the GAN, it is possible to control the data generation process. These additional input may be class labels or other items such as text or an image. In section 7.7, we will see how conditional GANs may be applied to image translation tasks. The new objective function of a conditional GAN becomes:

$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x)} \Big[\log(D(\boldsymbol{x}|\boldsymbol{c})) \Big] + E_{z \sim p_z(z)} \Big[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{c}))) \Big]$$

6.4 Inference GANs

The first GAN architectures lacked methods to convert

6.5 Progressive GANs

7 Applications

GANs are an exciting idea that are currently leading the frontier of deep generative models.

7.1 Divergence Measures

7.2 Image Quality

Hinton's Capsule Network Progressive GANs

7.3 Semi-Supervised Learning

7.4 Feature Learning

7.5 Semi-Supervised Learning

7.6 Reinforcement Learning

7.7 Image to Image Translation

8 Research Frontiers

Evaluation, instability, mode collapse

Reinforcement learning

9 Conclusion

Generative models are currently being applied to many problems [8]; however, they offer many more possibilities due to their ability to understand the data they are given.

References

[1] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A Bharath. Generative Adversarial Networks: An Overview. *ArXiv e-prints*, October 2017.

[2] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *ArXiv e-prints*, December 2017.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[5] Avinash Hindupur. The GAN Zoo, 2017. [Online; accessed November 27, 2017].

[6] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv e-prints*, February 2015.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *ArXiv e-prints*, November 2016.

[8] Andrej Karpathy, Pieter Abbeel, Greg Brockman, Peter Chen, Vicki Cheung, Rocky Duan, Ian Goodfellow, Durk Kingma, Jonathan Ho, Rein Houthooft, Tim Salimans, John Schulman, Ilya Sutskever, , and Wojciech Zaremba. Generative Models, June 2016.

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *ArXiv e-prints*, October 2017.

[10] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled Generative Adversarial Networks. *ArXiv e-prints*, November 2016.

[11] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *ArXiv e-prints*, November 2014.

[12] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press, 2002.

[13] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv e-prints*, November 2015.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. *ArXiv e-prints*, June 2016.

[15] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *ArXiv e-prints*, April 2014.

[16] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised Cross-Domain Image Generation. *ArXiv e-prints*, November 2016.

[17] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *ArXiv e-prints*, November 2015.