Generative Adversarial Networks

Jacob Smith

December 5, 2017

Abstract

Within the past decade, deep generative networks have often fallen short, especially when compared to deep discriminative models. However, generative adversarial networks (GANs), first introduced in 2014 by Goodfellow *et al.* [8], have proved themselves as a promising research avenue for deep generative models. GANs present a new architecture which allows for the unsupervised training of a generative model through backpropogation. Due to the current research effort, state-of-the-art results are now being achieved on a regular basis [14, 25]; however, GANs still encounter instability during training and do not yet poccess the ability to produce completely realistic images. Nevertheless, these networks offer encouraging results for unsupervised learning, semi-supervised, reinforcement learning and several image generation tasks. This research paper will examine the motivations and challenges associated with GANs, their training process, architecture variations and some applications that are currently being explored.

1 Introduction

Deep learning has made astounding progress within the past decade. Discriminative models have already surpassed the abilities of humans to recognize patterns within certain domains [23]. 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 their parameters [8]. However, deep learning possesses much more ambitious goals within the realm of unsupervised learning. 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 pro-

Figure 1: The process density estimation of one-dimensional data and a Gaussian distribution [7]. Generative models take a dataset D, sourced from a distribution p_{data} and create an estimate of the distribution p_{model}.

cess to accomplish this feat. Although progress within the machine learning field is rapidly advancing, computers maintain a limited understanding of the data they process. Generative models, especially deep generative models such as GANs, offer promising results towards this formidable challenge [13].

1.1 Generative Modeling

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 $p(y|x)$. An example of this process is illustrated in figure 1. Generative models may be used as classifiers by applying Bayes rules to calculate the conditional probability $p(y|x)$ [19]. However, generative models offer several other applications due to their knowledge of the underlying data distribution. Depending on the model, knowledge of the probability distribution can be created both explicitly and implicitly [9]. Those which do not directly model a probability distribution offer mechanisms which require some implicit knowledge of the underlying distribution, such as generating samples from that distribution [9].

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 generative model to generate data from that domain. As these models typically have fewer parameters than that the number of data samples, they are forced to internalize some representation of the data [13]. Unlike supervised training, there exists no single desired output. Therefore, the problem becomes defining a cost function which forces the generative model to produce data more like that of the target domain [13].

1.2 Generative Adversarial Networks

Generative adversarial networks (GANs), first introduced in 2014 by Goodfellow *et al.*, offer a new framework for estimating generative models with use of an adversarial process [8]. 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, zero-sum game. The generative model G attempts to produce data that resembles that of the training set while the discriminative model D attempts to discriminate between a real and fake (generated) sample. 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 real data and generated data probability distributions. Backpropagation is used to update the parameters of both the discriminator and generator. In the ideal case, D is unable to distinguish generated samples from real samples and produces an error rate of $\frac{1}{2}$ [8].

The adversarial process is often analogized to a counterfeiter trying to fool a detective [8]. The detective will receive both real and counterfeited money and will attempt to distinguish between the two. This leads to a competitive scenario where both parties gradually improve their abilities until the counterfeits are indistinguishable from the true currency.

1.3 Motivations

There exists several compelling reasons for studying generative modeling, especially GANs [7]. Generative models are extremely useful when your goal is understand the underlying data generating distribution. The ability to learn the joint probability distribution of high dimensional data is relevant to both applied math and engineering, in addition to many other fields [7]. Furthermore, GANs promise numerous real world applications which act as key motivators for their research. Several of these applications are expanded upon in section 7.

1.4 Related Work

Until the advent of GANs, most deep generative models provided a parametric specification of an probability distribution function [8]. These models

could be trained using maximum likelihood estimation which often required gradient estimation. The most successful of these models was the deep Boltzmann machine [8]. The difficulties associated with gradient estimation motivated the development of *generative machines*. These new models implicitly represent the joint probability distribution by generating samples from that distribution rather than explicitly rendering a probability distribution.

Deep generative stochastic networks are an example of a generative machine. This model, developed by Bengio *et al.* in 2013 [1], was able to use back-propagation during training; however, still made use Markov chains which introduced instability problems. GANs extend the ideas presented within generative stochastic networks by eliminating the use of Markov chains [8].

2 Background

The goal of the GAN training process is to determine the discriminator and generator parameters such that the classifier error is minimized and maximized respectively [2]. GANs may use any differentiable functions for their underlying implementation; however, optimal results are typically achieved when neural networks are used [8]. We first define a vector z which is sampled from a prior noise distribution $z\ p_z(z)$. This vector becomes the argument of the generator $G(z;\theta_g)$ where G is a differentiable function and θ_g are its parameters. Another differentiable function $D(x;\theta_d)$ is defined for the discriminator. This function takes as input a vector x from either the output of G or the training set and is parametrized by θ_d. The output of the discriminator is a scalar number which represents the probability that the sample x was derived from the training data. In this situation, there exists a minimax game, a scenario derived from game theory, where D attempts to maximize the probability of assigning a correct label and G attempts to minimize that same probability. The value function $V(D,G)$ becomes:

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

2.1 Cost Functions

For training, any variant of gradient descent may be used [7]. The first part of the training step involves updating θ_D while attempting to minimize the discriminator cost function J_D. The next part involves updating 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:

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

This is a standard cross entropy cost function of a neural network that is trained on two minibatches, one from the training dataset and the other 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 original GAN paper [8]. Goodfellow demonstrated that learning within this minimax game is identical to minimizing the Jensen-Shannon divergence and that the game will eventually converge if both players are able to directly update the function space as oppose to the parameters within the network [7]. 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 $log(1-D(G(z)))$ [8]. 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:

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

A maximum likelihood cost is another option for training GANs [7]. It can be shown that that this cost function minimizes the Kullback-Leibler divergence

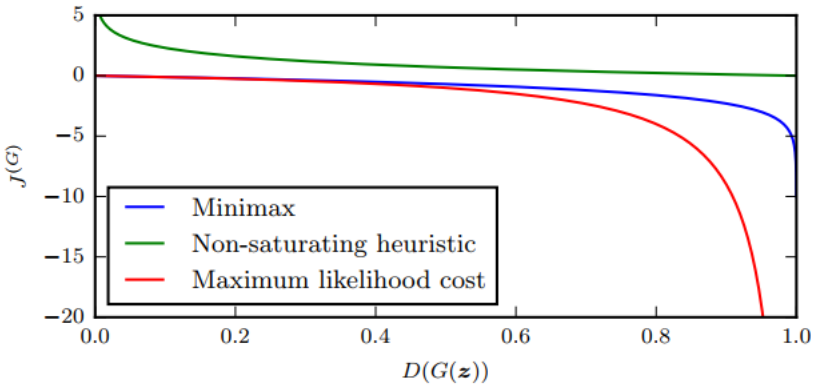

Figure 2: A comparison of the generator cost function for generative adversarial networks. When $D(G(z))$ is close to 0, the gradient is saturated for both the minimax and maximum likelihood cost functions. This situation occurs when the discriminator D confidently rejects the generated sample $G(z)$. 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 of $J^{(G)}$ as $D(G(z))$ nears 1. Image retrieved from Goodfellow's NIPs tutorial [7].

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 figure 2. The cost function for this version becomes:

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

2.2 Algorithm

The following algorithm presents the initial training algorithm as introduced by Goodfellow *et al.* in 2014 [8]. The first step in the algorithm is to optimize the discriminator D. Many researchers initially held the belief that it would be optimal to train D for several iterations before training the generator; however, this is still a controversial topic among researchers [7].

It is Goodfellow's opinion that each model should be trained simultaneously [7].

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 [8]. Multiple variations of this algorithm have since been produced. One of the benefits of adversarial networks is that they are able to make use of back propagation for training.

3 Issues

There exist several issues that characterize GANs. Some are due to their architecture while others to implementations of the generator and discriminator. These issues lay at the forefront of GAN research.

3.1 Instability

GANs are often characterized by their instability [7]. 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 training process is different than network optimization which is common in discriminative models. When either the generator or discriminator trains, even if that specific 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 zero-sum game situations [7].

Goodfellow has demonstrated that convergence is guaranteed if updates are made directly to the function space; however, this is not what occurs in practice. During training, we make updates to the parameter space rather than the function space [8]. As of yet, no theoretical justifications have been found that guarantee convergence or non-convergence when updates are made to the parameters [7]. In practice, adversarial networks often oscillate between different kinds of samples without converging to an equilibrium in a phenomenon called mode collapse.

3.2 Mode Collapse

Mode collapse occurs when the generator collapses several different latent space vectors z to the same outputs. This problem can be visualized in figure 3. Full mode collapse is rare; however, partial mode collapse often occurs and varies by intensity [7]. For example, the generator may create several images that contain the same object at different perspectives, but generated with dissimilar 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 is section 2 and may behave like the maximin version:

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

When this occurs, the generator attempts to map every z vector to the sample x that which the discriminator is most likely to guess is real. When training, one can only hope that the behavior of gradient descent resembles that of the minimax game [7]. It is often speculated that the root cause of the issue is the particular divergence measure used [7]. 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 been shown

Figure 3: An example of mode collapse as demonstrated by Metz *et al.* [17]. The generated samples cluster around only one of the modes and do not spread out as would be expected. Throughout the training process, this cycle continues and does not reach an equilibrium.

to worsen the problem [7]. As partial mode collapse is currently an unavoidable reality, the applications of GANs are limited to situations where it is acceptable to generate only a small number of distinct outputs [7]. However, recent progress has been made to reduce this problem. A potential solution is discussed in section 4.2.

3.3 Image Quality

When generating images, problems often arise due to the generators difficulties with perspective, counting and structure [7]. For example, an generated animal may have a 2D structure, misplaced limbs or too many of a specific body part. These issues arise due to the underlying generator architecture, typically deconvolutional neural networks, and are not an inherent problem of the adversarial network [7]. Furthermore, generated image resolution has remained relatively low; however, recent improvements outlined in section 5.5 show drastic improvements with regard to resolution.

3.4 Evaluation

The evaluation procedure for GANs remains an open debate as there exist no clear quantitative measurement to evaluate generative models [7]. For example, models with a high likelihood may produce unrealistic samples whereas models with a low likelihood may produce realistic samples [7]. It has been shown that different metrics lead to different tradeoffs and different evaluations favor different models [24]. Therefore, it is important to asses one's generative model within the context of the whole application [24].

3.5 Discrete Data

GANs assume a model implementation that is differentiable due to their use of back propagation. Therefore, this restricts their ability to generate discrete data. Goodfellow proposes several solutions such as algorithm modifications, discretization of outputs or the use of concrete distributions or Gumbel-softmax [7].

4 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 Salimans *et al.* while at OpenAI outlines several techniques that can be applied to improve the GAN training process [22]. As mentioned in section 3.1, the optimal solution for GANs is the convergence to the Nash equilibrium of the minimax game; however, gradient descent attempts to minimize the cost function and will not necessarily find the Nash equilibrium. The following techniques introduced in the paper address the instability of GANs and attempt to increase the probability of convergence.

4.1 Feature Matching

Feature matching involves the definition of a new cost function for GANs to reduce the probability that the generator will not overtrain on the current discriminator [22]. Instead of optimizing on the output of the discriminator, the new objective requires the generator to minimize the difference between the expected value of some intermediate layer. The new 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 of the discriminator. Experimental evidence gathered has demonstrated the usefulness of this technique, especially when the training process becomes instable [22]. Furthermore, feature matching is useful for semi-supervised GAN applications as will later be described in section 7.2.

4.2 Minibatch Discrimination

Minibatch discrimination attempts to minimize the issues caused by mode collapse. This issue occurs as the discriminator processes each sample x independently of each other sample. Therefore, the gradients of discriminator will all point in the same direction, thus leading to mode collapse. This technique involves the use of a minibatch in place of a single sample 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 applications of this technique allow for visually appealing images to be generated much quicker than previous training techniques [22]. Preliminary results also show a great reduction of mode collapse [7].

4.3 Historical Averaging

Historical averaging introduces an extra term to each player's cost function such that they are penalized when choosing new parameters that differ from the historical mean value [22]. The new term added is $||\theta - \frac{1}{t}\sum_{i=1}^{t}\theta[i]||^2$ where t is the total number of iterations performed and $\theta[i]$ is the value of the parameters at iteration i. This technique has been demonstrated to help low dimensionality games reach the Nash equilibrium where normal gradient descent would have failed [22].

4.4 Label Smoothing

Label smoothing is a technique that was first introduced in the 1980s which involves replacing the 0 and 1 targets with *smoother* values such as 0.1 and 0.9 respectively [22]. This technique is used to improve the networks resistance to adversarial examples. Specifically, it is recommended to use only one-sided label smoothing by replacing the 0.9 label instead of 1 for the real data label. If one alters the label for fake data, problems may arise when $p_{data}(x)$ nears 0 and $p_{model}(x)$ becomes large for some sample x [22].

4.5 Virtual Batch Normalization

Batch normalization (BN), a technique that was first introduced in 2014, involves normalizing each training minibatch [11]. It allows for much higher learning rates, a more stable training process and a reduction of problems associated with parameter initialization [11]. BN has proved useful for GAN training; however, it often causes a dependence relationship between the samples of the minibatch [22]. A new technique called virtual batch normalization (VBN) resembles BN; however, a reference batch of samples is randomly chosen at the start of the training process and used throughout to normalize each training sample. This technique is computationally expensive as it requires forwarding two minibatches and it is recommended that one only use VBN for the generator [22].

5 Architecture & Training Variations

Research within the field of deep generative modeling has increased exponentially since the debut of GANs in 2014. Significant improvements have been made with respect to image quality and training stability. Hundreds of papers have been released on the subject of GANs (see figure 4) which propose numerous architectural and training variations.

5.1 Fully Connected GANs

The first GAN paper by Goodfellow *et al.* made use of relatively simple fully connected neural networks for the generator and discriminator models [2]. 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 is useful for proof of concept; however, several other architectures regularly outperform this implementation [2].

5.2 Convolutional GANs

Although convolutional neural networks (CNNs) seem well suited for the image generation tasks often associated with GANs, difficulties with training initially constrained the use of convolutional models for both the discriminator and generator [2]. When implementing CNNs within a GAN, one is

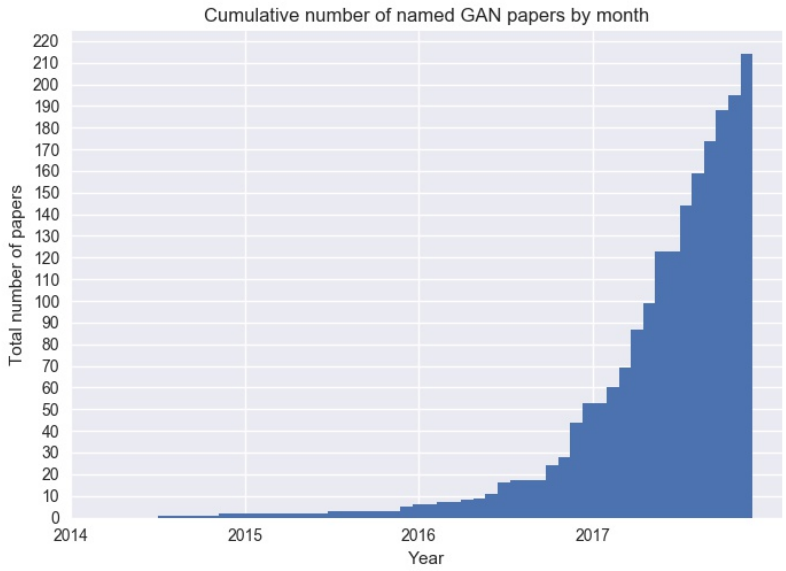

Figure 4: The cumulative amount of papers written about generative adversarial networks (GANs) by month. The graph shows the exponential growth of GAN research that has occurred within the past three and a half years. Image maintained by Avinash Hindipur [10].

unable to train the discriminator and generator to the same level of capacity and representational power as supervised CNN models [2]. One solution proposed by Denton *et al.* made use of Laplacian pyramids to address this problem [3]. A Laplacian pyramid is a type of multi-scale signal representation which repeatedly smooths and downsamples an image to form a sort of pyramid of images. At each layer of the pyramid, a conditional convolutional GAN (see section 5.3) is trained given the generator output of the above layer.

Another solution to this problem proposed by Radford *et al.* made alterations to the conventional deep convolutional network architecture [20]. Max pooling was replaced with strided layers within the discriminator and fractionally-strided layers within the generator. Furthermore, LeakyReLUs were used for all layers of the discriminator in place of the standard ReLU non-linearity function. The fractionally strided layers within the generator replicate the benefits of the progressive upscaling within the Laplacian pyramid [2]. The strided layers allow the GAN to learn both the down-sampling and up-sampling operators during training [2]. This architecture, called DCGAN, has become the base model for most GANs today [7].

5.3 Conditional GANs

Initially proposed by Goodfellow as a straightforward extension of the first GAN architecture [8], Mirza *et al.* successfully implemented a conditional GAN (CGAN) in late 2014 [18]. CGANs differ from conventional models due to the additional parameter c that is given to both the generator $G(z|c)$ and discriminator $D(x|c)$ to condition the GAN. The new objective function of a CGAN becomes:

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

These architectures are better suited for multi-modal output (see section 7.1) as they are able to provide better data representations [2]. By conditioning the GAN, it is possible to control the data generation process. This additional input may be class label or other items such as text or an image. For example, given that a CGAN was conditioned on sentences, one may be able to describe a scenario and have that specific scenario synthesized as

an image [21]. In section 7.1, we will see how CGANs are applied to image translation tasks.

5.4 Inference GANs

The first GAN architectures lacked the ability to map an sample x to its associated latent vector z. The ability to do this is often referred to as having an *inference mechanism* [2]. In 2016, two new architectures were independently proposed which both introduced this mechanism by adding an inference network [4, 5]. This new architecture involves two generative models, a decoder and encoder, which work together to fool the discriminator. Instead of only receiving a sample x, the discriminator is also trained on a latent space vector z and has to decide if the pair is a real sample and its encoding or generated sample and its associated latent space vector. This architecture allows the encoder to learn the mapping of x to z. Furthermore, the encoder generates a set of continuous variables. That is, if the vector z is of length n, then n means and n variances will be generated. This allows for a probability distribution for each value of z. However, the reconstructed images still remain of poor quality [2].

5.5 Progressive GANs

Progressive GANs do not offer an architecture variation but a new training methodology. Introduced in October 2017 by Karras *et al.* [14], these networks incrementally increase the number of convolutions within the generator and discriminator. A new convolution is added after the previous generator converges until a limit is reached. This permits the generator to progressively discover the image distribution. This technique also increases training speed and improves training stability [14]. Combined with new normalization techniques, progressive GANs allow for much higher resolution images to be generated [14]. The experimental results received by Karras offer promising new directions for GAN research and applications.

6 Latent Space

The generator model of a GAN must internalize a representation of the data that it is trained on. This quality is shared with variatinal autoencoders and

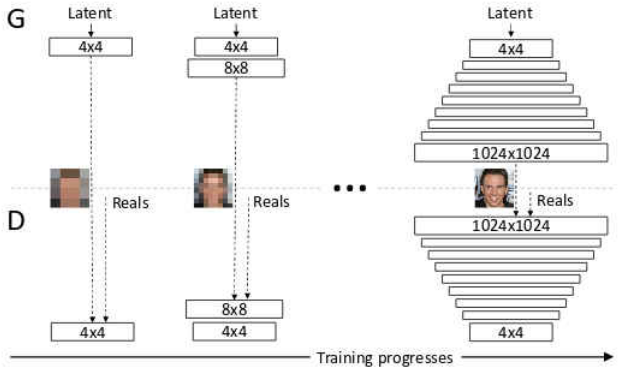

Figure 5: The training process of progressively growing generative adversarial networks recently introduced by Karras *et al.* [14]. The generator and discriminator begin at $4x4$ resolution and progressively grow to generate and discriminate $1024x1024$ resolution images.

word linguistic models such as *word2vec* [2]. One interesting characteristic of the latent space is that it is typically of lower dimension than the training data [2]. This forces the model to efficiently represent the training data by extracting only useful features. As long as the effects of mode collapse are minimal, the latent space is often highly structured [2]. We may observe semantic transformations when interpolations are conducted. For example, movement of the latent vector in certain directions may alter rotation, produce colour changes or see the addition of attributes such as eyeglasses [2].

As mentioned in section 5.4, it is possible to train an encoder to map a sample x to a latent vector z. This is an extremely useful feature for exploring the latent space [2]. For example, one may discover *concept vectors* such as "wearing glasses" or "wearing a hat" and apply them to a vector associated with a human. An example of this process is illustrated in figure 6.

7 Applications

Although GANs are a relatively new in the field of machine learning, many different applications have already been discovered. Several of the reoccurring themes are elaborated within this section.

Figure 6: The addition of the "smile" concept vector (far right) to a vector associated with a woman (far left). The images in-between were created through an interpolation of the two vectors. Example retrieved from Dumoulin *et al.* [5].

7.1 Image Generation

Image generation is extremely useful as many tasks intrinsically require the ability to generate realistic samples [7]. Moreover, image generator often requires the ability to model multi-modal outputs (see figure 7) [7]. GANs are well suited for both of these tasks. Most research conducted with respect the GANs attempts to improve the quality of synthesized images [2]. As such, several useful and fascinating image applications have already been found.

7.1.1 Image Super-Resolution

Image super-resolution is the process of generating a high resolution image from a lower resolution image by inferring details. For each lower resolution sample, there exists multiple possible correct outputs. Due to this characteristic, GANs are optimally suited for this task. The Super-Resolution GAN (SRGAN) proposed by Ledig *et al.* uses the conditional GAN architecture. The discriminator is trained to distinguish real images from super-resolution images [15]. SRGANs infer photo characteristics with respect to tje domain of training set [2]. Therefore, it is essential to train different models for each domain of images.

7.1.2 Image-to-Image Translation

Image-to-image translation has made astounding progress recently and offers many yet undiscovered applications [7]. This process includes the translation

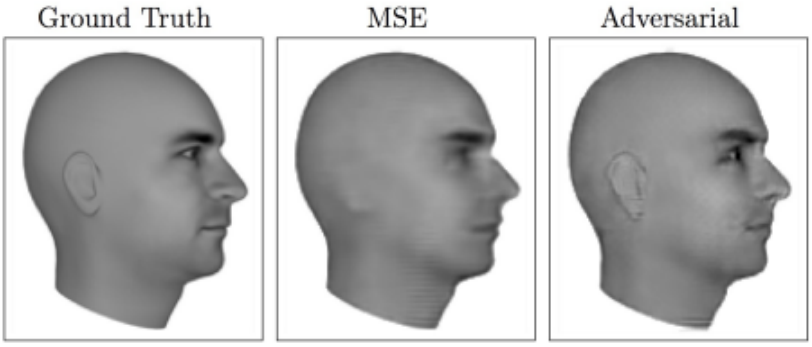

Figure 7: An example of scenario where the ability to model multi-modal data is incredibly important. This figure demonstrates the process of predicting the next video frame given the previous one. A generator trained with mean squared error (middle) produces an image which averages each possible future. When using an adversarial process (right), the image produced corresponds with the most likely scenario and is not blurry unlike the image generated using MSE. Example retrieved form Lotter *et al.* [16].

of images from a source domain X to the target domain Y. The translated images are then compared against images from the target domain and the generator parameters are updated as to reduce the distance between the translated and real images. An example of this operation can be visualized in figure 8. Conditional networks are well suited for this task and there exists several different GAN architectures which take advantage of their characteristics to preform these translations [12, 25, 26]. Recent work by Zhu *et al.* demonstrate the feasibility of unpaired image-to-image translation by introducing a cycle consistency loss to enforce $F(G(X)) \approx X$ where G is the mapper and F is the inverse mapper [26]. Furthermore, research released in November 2017 by Wang *et al.* further improve the general quality of synthesized images [25]. Their semantic map translations are now almost indistinguishable from real images.

7.2 Semi-Supervised Learning

There exist several techniques for training a classifier using a GAN. These methods all take advantage of the unsupervised aspect of GANs to implement semi-supervised learning. The ability to use GAN for semi-supervised

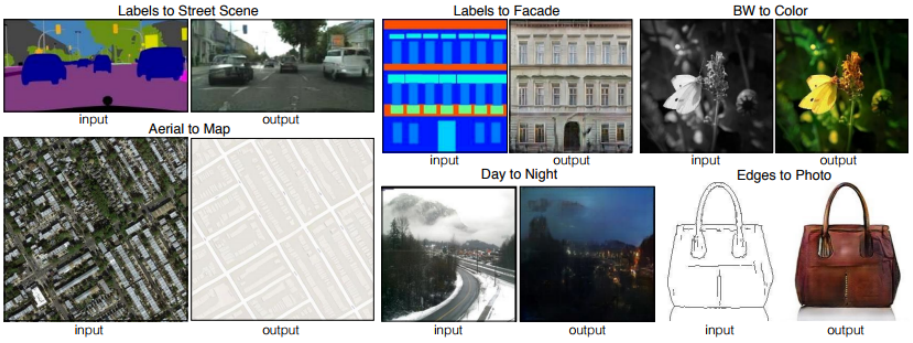

Figure 8: Examples of the image-to-image technique developed by Isola *et al.* [12]. This method makes use of conditional generative adversarial networks to map across image domains.

learning also offers a quantitative method to rank different GAN models.

One method involves the use of transfer learning. After the GAN training process has completed, the discriminator may be used within classification tasks as a feature extractor. A new machine learning model, such as a support vector machine, is trained on the extracted features of some layer l of the discriminator. Another transfer learning approach makes use of inference GANs. A new machine learning model is trained on the last three hidden layers of the of the encoder. These methods have both received state-of-the-art classification results [2].

Another general approach to semi-supervised learning involves the transformation of the discriminator into a classifier. Given that there are k different labels for some classification task, the new classifier will have $k + 1$ classes where the $k + 1$ class is associated with a fake sample. When a real sample is given to the discriminator, it attempts to classify the sample as one of the $k + 1$ classes; however, when a generated or unlabeled sample is given, the probabilities of the k classes are summed together to represent the probability of the sample being real. In this situation, the classifier reverts back to its discriminative behavior. Typical supervised models need more than $50,000$ samples whereas state-of-the-art performance can be achieved on labeled datasets ranging from 20 to $8,000$ samples using this technique [7].

7.3 Reinforcement Learning

Modeling environments offers another promising application of GANs. This capability could potentially be used for reinforcement learning and motion planning. One possible method includes the use of generative models to predict possible futures [7]. The GAN would learn a conditional distribution over future states given the current state and a hypothetical action. An agent could make multiple queries to the generator and subsequently choose the action associated with the best possible future. This technique has already been applied by Finn and Levine in 2016 [6]. Generative models may also be used to maintain a record of past actions and visited locations to aid in future decisions [7].

8 Conclusion

GANs are generative models which allow us to parametrize complicated probability distribution functions. They can be used to generate, modify and manipulate images. Furthermore, they allow for the use novel techniques within the context of reinforcement learning and semi-supervised learning. Today, they are already being applied to many problems [13]; however, the majority of their applications have yet to be discovered. As long as the research effort continues, GANs promise exciting new research avenues within the field of machine learning.

References

[1] Y. Bengio, É. Thibodeau-Laufer, G. Alain, and J. Yosinski. Deep Generative Stochastic Networks Trainable by Backprop. *ArXiv e-prints*, June 2013.

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

[3] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *ArXiv e-prints*, June 2015.

[4] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial Feature Learning. *ArXiv e-prints*, May 2016.

[5] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially Learned Inference. *ArXiv e-prints*, June 2016.

[6] C. Finn and S. Levine. Deep Visual Foresight for Planning Robot Motion. *ArXiv e-prints*, October 2016.

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

[8] 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.

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

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

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

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

[13] 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.

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

[15] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *ArXiv e-prints*, September 2016.

[16] W. Lotter, G. Kreiman, and D. Cox. Unsupervised Learning of Visual Structure using Predictive Generative Networks. *ArXiv e-prints*, November 2015.

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

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

[19] 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.

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

[21] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative Adversarial Text to Image Synthesis. *ArXiv e-prints*, May 2016.

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

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

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

[25] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *ArXiv e-prints*, November 2017.

[26] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *ArXiv e-prints*, March 2017.