LogicGAN: Logic-guided Generative Adversarial Networks

Laura Graves†, Vineel Nagisetty†, Joseph Scott, and Vijay Ganesh
University of Waterloo
{laura.graves, vineel.nagisetty, joseph.scott, vganesh}@uwaterloo.ca

Abstract
Generative Adversarial Networks (GANs) are a revolutionary class of Deep Neural Networks (DNNs) that have been successfully used to generate realistic images, music, text, and other data. However, it is well known that GAN training can be notoriously resource-intensive and presents many challenges. Further, a potential weakness in GANs is that discriminator DNNs typically provide only one value (loss) of corrective feedback to generator DNNs (namely, the discriminator’s assessment of the generated example). By contrast, we propose a new class of GAN we refer to as LogicGAN, that leverages recent advances in (logic-based) explainable AI (xAI) systems to provide a “richer” form of corrective feedback from discriminators to generators. Specifically, we modify the gradient descent process using xAI systems that specify the reason as to why the discriminator made the classification it did, thus providing the richer corrective feedback that helps the generator to better fool the discriminator. Using our approach, we show that LogicGANs learn much faster on MNIST data, achieving an improvement in data efficiency of 45% in single and 12.73% in multi-class setting over standard GANs while maintaining the same quality as measured by Fréchet Inception Distance. Further, we argue that LogicGAN enables users greater control over how models learn than standard GAN systems.

1 Introduction
Generative Adversarial Networks (GANs), introduced only a few short years ago, already have had a revolutionary impact on generating data of varied kinds such as images, text, music, and videos [Goodfellow et al., 2014]. The critical insight behind a GAN is the idea of corrective feedback loop from a deep neural network (DNN) called the discriminator back to a generator. However, a weakness of the standard GAN architecture is that the discriminator DNNs provide only one real-numbered value of corrective feedback to the generator DNN. That is, the discriminator simply informs the generator whether or not the data generated by it was acceptable, in the form of a single loss value. The downside of this approach is that the generator has to somehow figure out why the discriminator rejected its generated data, and then make corrections as it learns how to fool the discriminator. Hence, the research questions we address in this paper are the following: is it possible to provide “richer” corrective feedback from the discriminator back to the generator? If so, does it enable GANs to learn faster and with greater accuracy?

1.1 An Overview of LogicGANs
To answer the above-mentioned research questions, we propose a new class of GANs we refer to as LogicGAN1, wherein it is possible to provide rich corrective feedback (more than a single value) during training from discriminator to generators via Explainable AI (xAI) systems such as SHAP [Lundberg and Lee, 2017]. We refer to our approach as logic guided and the entire GAN as LogicGAN because explainable AI (xAI) systems are often based on some form of logic or symbolic reasoning method.

A high-level architectural overview of our LogicGAN system is given in Figure 1. Consider the problem of training a GAN with the aim of producing images of digits. Initially, the untrained generator $G$ is given a noise sample $z$ from a prior noise distribution $p_z$ and produces an example $G(z)$ that is then given to discriminator $D$. The loss is calculated, and then the generated image $G(z)$, the discriminator $D$, and the output of the discriminator $D(G(z))$ are fed into an xAI system, such as SHAP, that produces an explanation as to why

---

1Joint First Authors

1An implementation of LogicGAN, which can be used to view and replicate our results, can be found at http://tiny.cc/LogicGAN.
the image resulted in that loss value. This explanation is then used to guide the training of the generator (refer Section 4 for more details).

A common analogy for GAN training is that of a counterfeiter (the generator) and a detective (the discriminator) playing an adversarial game where the counterfeiter makes a fake and the detective tries to tell if it’s real or not. Over the training process, the detective and counterfeiter both get better at their jobs, with the end goal being that the counterfeiter is so proficient that their fakes can pass for the real thing. To extend this analogy to the LogicGAN setting, our method works by using an expert in the field (the xAI system) to help improve the counterfeiter. When the detective recognizes a fake, the expert tells the counterfeiter what parts of the fake tipped off the detective. The counterfeiter is thus able to learn better why the detective detected a fake, and make better decisions to avoid pitfalls over future learning iterations.

SHAP xAI Explanations: The explanation produced by SHAP [Lundberg and Lee, 2017] takes the form of an “explanation matrix” $M$, wherein, for every feature in an example (e.g., an input image), a value in the range [-1,1] is assigned to the corresponding cell of the matrix $M$. If a feature (more precisely, the corresponding cell in $M$) is assigned value 0 (or close to 0), then it means that pixel had no impact on the classification decision made by the discriminator $D$. If a feature is assigned a value of 1 (or close to 1), then that means that feature is very important and “determinative” in the classification made by $D$. Finally, if a feature is assigned a value of -1 (or close to -1), then that means feature “hurt” the classification made by $D$ (more precisely, if the feature were to be changed, then the confidence of $D$ in its classification would improve).

xAI-guided Backpropagation: The matrix $M$ generated by the xAI system SHAP is then used in a modified backpropagation algorithm (see Algorithm 1) to update the weights of the generator as follows: traditionally, in a GAN the weights of the generator are modified by first computing the gradient of generator’s output with respect to the loss and then applying the chain rule. We modify this algorithm by first computing the explanation matrix $M$ (via the xAI system) and then calculating the product (specifically, a Hadamard product [Horn, 1990]) between $M$ and the gradient of the generator output with respect to the loss $\Delta G(z)$. More precisely, the explanation matrix $M$ is used to mask the gradient function, and consequently the “importance of the pixels” that went into the discriminator’s classification are taken into account in the modification of the weights of the generator during the application of the backpropagation algorithm. Using our approach, one can foresee that users might be able to augment such explanation matrices with specifications that spell out relationships (using logical formulas) between the update methods for the various weights of a generator.

We would like to emphasize that the standard gradient descent method simply moves toward the greatest decrease in loss over an $n$-dimensional space, while by contrast, xAI-guided gradient descent algorithms can give users greater control over the learning process.

1.2 Contributions

1. Our key contribution is a logic or xAI-guided gradient descent method (and a resultant GAN we refer to as LogicGAN) that utilizes xAI systems to focus the gradient descent algorithm on weights that are determined to be most influential by the xAI system (refer Section 4.2). In Section 6.1, we discuss how x-AI guided gradient descent methods can give those training models greater control over the learning process.

2. We performed experiments to evaluate the efficiency and quality (as measured by Frechet Inception Distance, abbreviated as FID [Heusel et al., 2017]) of LogicGAN relative to standard GANs. We show that on the MNIST single digit classification problem, LogicGAN achieves a 45% improvement in data efficiency compared to standard GANs (refer Section 5.3), while maintaining the same level of quality as measured by FID score.

3. We extend our experiments to the multi-class digit classification problem. LogicGANs outperform GANs with a 12.73% improvement in data efficiency while maintaining the same level of quality as measured by FID score in this setting as well (refer Section 5.4).

2 Prior Work

Goodfellow et al. were the first to introduce GANs in [2014]. Since then, GANs have continued to be a popular research topic with many versions of GANs developed [Pan et al., 2019]. GANs can be broadly classified based on their architecture [Radford et al., 2015; Mirza and Osindero, 2014; Chen et al., 2016; Makhzani et al., 2015] and the type of objective function used [Metz et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Mao et al., 2017; Dziugaite et al., 2015; Zhao et al., 2016]. To the best of our knowledge, there is no GAN that uses xAI feedback for training, thus making LogicGAN the first of its kind. We note that the approach exemplified in LogicGANs is independent of architecture or type of objective function used, and therefore can be applied to make any GAN xAI-guided.

ADAGRAD [Duchi et al., 2011] is an optimization algorithm that maintains separate learning-rates for each parameter of a DNN based on how frequently the parameter is updated. On the other hand, LogicGAN uses xAI feedback to determine how generator parameters are updated (refer Section 4.2).

Explainable AI (xAI) Systems: As AI models become more complex, there is an increasing demand for interpretability or explainability of these models from decision makers, stakeholders, and lay users. In addition, one can make a strong case for a scientific need for explainable AI. Consequently, there has been considerable interest in xAI systems aimed at creating interpretable AI models that enable human understanding of AI systems [Biran and Cotton, 2017].

One way to define xAI systems is as follows: they are algorithms that, given a model and a prediction, assigns values to each feature of an input that measures how important that feature is to the prediction. There have been several different xAI systems applied to DNNs with the goal of improving our
understanding of how these systems learn. These systems can do so in a variety of ways, and approaches have been developed such as ones using formal logic [Ignatiev et al., 2019], game-theoretic approaches [Lundberg and Lee, 2017], or gradient descent measures [Shrikumar et al., 2017]. These systems output explanations in a variety of forms such as ranked lists of features, select subsets of the feature sets, and values weighting the importance of features of input data used to train machine learning models.

3 The DeepSHAP xAI System

In our work, we use the DeepSHAP xAI system, which is one of the most widely used xAI systems [Lundberg and Lee, 2017]. DeepSHAP is a combination of the DeepLIFT platform and Shapley value explanations. Introduced in 2017, the platform is well-suited for neural network applications and is freely available. DeepSHAP is an efficient Shapley value estimation algorithm. It uses linear composition rules and backpropagation to calculate a compositional approximation of feature importance values.

Shapley Value Estimation: Classic Shapley regression values are intended for linear models, where the values represent feature importance. Values are calculated by retraining models on every subset of features $S \subseteq F$ and valuing each feature based on the prediction values on models with that feature and without. Unfortunately, this method not only requires significant retraining but also requires at least $2^{|F|}$ separate models to cover all combinations of included features. Methods to approximate the Shapley values by iterating only over local feature regions, approximating importance using samples from the training dataset, and other approaches have been proposed to reduce computational effort.

The DeepLIFT xAI system: DeepLIFT uses a set of reference inputs and the consequent model outputs to identify the importance of features [Shrikumar et al., 2017]. The difference between an output and a reference output, denoted by $\Delta y$, is explained in terms of the differences between the corresponding input and a reference input, given by $\Delta x_i$. The reference input is chosen by the user based on domain knowledge to represent a typical uninformed state. For example, in the MNIST state, the reference value is an image from the dataset. Each feature $x_i$ is given a value $C_{\Delta x_i, \Delta y}$ which measures the effect of the model output on that feature being the reference value instead of its original value. The system uses a summation property where the sum of each feature’s changes sum up to the change in the model output $\Delta y$ of the original in comparison to the reference model: $\sum_{i=1}^{n} C_{\Delta x_i, \Delta y} = \Delta y$.

Our xAI-guided gradient descent uses the DeepSHAP algorithm to generate feature importance values that are in the range $[-1, 1]$. We then take the absolute value of the explanation matrix $M$ and normalize them to the range $[0, 1]$ to use as our mask. Taking the absolute value helps us focus the learning process on the most influential features, regardless of whether those features were beneficial or harmful to the classification. We found that this approach was significantly more effective than only normalizing the xAI results to $[0,1]$.

### Algorithm 1: Generator Training Algorithm

Note that the highlighted part only applies to xAI-guided generator training.

```plaintext
input : generator G
discriminator D
use_xAI : boolean Flag
output: trained generator G

1 foreach noise sample $z$ do
2 Loss $L = Loss(1 - D(G(z)))$
3 compute Discriminator Gradient $\Delta D$ from $L$
4 compute Generated Example Gradient $\Delta G(z)$ from $\Delta D$
5 if use_xAI is True then
6 compute Explanation Matrix $M$ using xAI
7 compute Modified Gradient $\Delta' G(z) = \Delta G(z) \times M$
8 compute Generator Gradient $\Delta G$ from $\Delta' G(z)$
9 else
10 compute Generator Gradient $\Delta G$ from $\Delta G(z)$
11 end
12 update Generator parameters $\theta_G$ using $\Delta G$
```

4 Detailed Overview of LogicGAN Systems

In this section, we provide a detailed overview of our LogicGAN system and contrast it with standard GAN architectures and the way they are trained. The intuition behind the xAI-guided generator training process is that the xAI system acts as a guide, shaping the gradient descent in a way that focuses generator training on the features that the discriminator recognizes as most important. In our implementation we use the DeepSHAP xAI system, though we note that LogicGAN is xAI agnostic and any system that gives a measure for feature importance can be used based on individual needs.

4.1 Generator Training in Standard GANs

Briefly, standard GAN architectures consist of a system of paired DNNs, namely, a discriminator $D$ and a generator $G$. The standard training method involves alternate cycles of discriminator and generator training. Initially, the discriminator is trained on a mini batch of examples drawn from both training data from the target distribution, as well as data generated by the untrained generator (which initially is expected to be just noise). These examples are correctly labeled as “real” (if they were from the training set) or “generated” (if they are from the generator).

Subsequently, the generator is trained as follows (please refer to the non-highlighted part in Algorithm 1): a selection of noise samples are drawn from the noise prior and passed through the generator to get a batch of generated examples (line 1). This batch is labeled as “real” and given to the discriminator, where the loss is found (line 2), and then used to update the generator parameters (the corrective feedback step). More precisely, the discriminator’s gradient $\Delta D$ is computed using the parameters of the discriminator and its loss (line 3), which is used to find the gradient of the generated example $\Delta G(z)$ (line 4). Further, the gradients of all lay-
ers in the generator $\Delta_G$ are then computed using $\Delta_G(z)$ (line 10). Finally, the parameters $\Theta_G$ of the generator are updated using $\Delta_G$ (line 12) - completing one iteration of training.

In subsequent iterations, the discriminator receives mini batches of real and generated examples from the generator trained in the previous iterations. The ideal termination condition for this process is when both the generated examples are high-quality and the discriminator is unable to distinguish between “real” and “generated” examples.

4.2 xAI-guided Generator Training in LogicGANs

We start our description of xAI-guided training by first observing that in the standard GAN setting the discriminator only gives a single value of corrective feedback to the generator per generated image. The entire point of xAI-guided training is to augment this corrective feedback with the “reason” for the discriminator’s decision, as determined by an xAI system such as DeepSHAP.

During our xAI-guided gradient descent generator training process, the backpropagation algorithm is modified to focus generator training on the most meaningful features for the discriminator’s prediction (please refer to the highlighted part of Algorithm 1). Following with propagating the loss through the discriminator to find $\Delta_G(z)$, we use an xAI system $E$ to find $M = E(G(z))$ (line 6). $M$ is a set of real values $\in [0,1]$, where greater values represent features that are more important to the discriminator’s prediction. The Hadamard (piecewise) product of $\Delta_G(z)$ and $M$ is calculated to get the modified gradient $\Delta'_G(z)$ (line 7). In an intuitive sense, the explanation $M$ acts as a mask for $\Delta_G(z)$, focusing the gradient on the most important features and limiting the gradient on the less important ones. From there, the gradients of the generator $\Delta_G$ are calculated from $\Delta'_G(z)$ (line 8) and the parameters are then updated (line 12).

4.3 LogicGAN Implementation Details

We implemented LogicGAN using Pytorch 1.3 [Paszke et al., 2019], an open source machine learning framework popular in deep learning research. Our xAI system is SHAP 0.31.0 which implements DeepSHAP [Lundberg and Lee, 2017]. In order to provide the reference outputs to DeepSHAP, we created a background selector system that provides a sample of relevant digits for both single and multi-class settings. We process the explanation matrix $M$ generated by SHAP by taking the absolute value and normalizing the matrix to create a mask vector with values in range $[0,1]$. Pytorch notably has the autograd [Paszke et al., 2017] package which handles automatic differentiation of all tensors. In order to provide xAI-guided feedback to the generator, we overrode the register_backward_hook function normally used to inspect gradients. We modified the gradients of the output layer of the generator using the resultant vector computed by the Hadamard (piecewise) product with the computed mask. This modified gradient is backpropagated through the generator via the autograd. This allows us to easily add our xAI-guided gradient descent method to a GAN with few lines of code.

5 Experimental Results

We performed extensive experimental evaluation of our LogicGAN implementation, comparing against standard GANs.

5.1 Experimental Setup

All our experiments were performed over the MNIST dataset [LeCun and Cortes, 2010], a collection of 60,000 28x28 grayscale images of handwritten digits. We performed experiments in both single-class and multi-class settings. We used the same architecture for both the standard GAN and LogicGAN models. The generators are fully connected networks with 3 hidden layers, taking 100 values sampled from a normal distribution as input and returning a 28x28 grayscale image. Each hidden layer uses Leaky ReLU, and the output layer uses tanh activation functions. The discriminators are fully connected networks with 3 hidden layers, taking a 28x28 grayscale image as input and returning a prediction score. Each hidden layer uses Leaky ReLU, and the output layer uses the sigmoid activation function. A dropout rate of 0.3 is used during training. The batch size is selected to be 100. The Adam optimizer [Kingma and Ba, 2014] is used for both generator and discriminator training. We varied learning rate between low (0.00002) and high (0.0002). We ran experiments using Amazon’s SageMaker platform run on an ml.p2.xlarge instance which uses Nvidia’s 1xK80 GPU with 12GB RAM.

5.2 Evaluation Criteria

Evaluating GANs remains an open problem as there are no single set of metrics identified as universally applicable for GANs [Pan et al., 2019]. However, there are several domain-dependent metrics widely used in GAN literature currently such as Inception Score (IS), Mode Score (MS) and Fréchet Inception Distance (FID). An overview of these metrics as well as their pros and cons can be found at [Borji, 2019]. Based on a thorough literature survey of metrics for the image domain, we developed the following criteria in order to perform a fair comparison of LogicGANs vs. standard GANs:

1. **Quality of Generated Images**: This involves measuring the similarity of the generated images w.r.t the training images as well as measuring diversity to determine mode collapse. We opted to use Fréchet Inception Distance (FID) to measure quality since it has been shown to be consistent with human evaluation of quality [Heusel et al., 2017]. FID was introduced by Heusel et al., to address the shortcomings of Inception Score (IS) such as the latter’s inability to detect intra-class mode dropping and vulnerability to noise [2017]. FID was used as the primary metric by Lucic and the team at Google Brain to compare various GANs [Lucic et al., 2018]. Moreover, the authors argue that GANs can be objectively and fairly compared by measuring FID scores for fixed computational budgets. We go a step further in our evaluation by comparing the FID scores of LogicGAN and standard GANs for each training epoch to get a sense of how the quality of their images change during training. This would help us better understand how our xAI-guided gradient descent affects GAN training.
order to apply FID to MNIST, we use a LeNet like convolutional classifier, consistent with [Bińkowski et al., 2018].

2. Data Efficiency: We quantify data efficiency in terms of the training cycles required for convergence. Kernel Inception Distance (KID) is shown to be an improved measure of GAN convergence in [Dziugaite et al., 2015]. At a high level, KID computes the square of the Maximum Mean Discrepancy (MMD) between the Inception scores of generated and training samples. KID has been shown to address over-fitting and over-training of GANs and so is a good metric to determine convergence [Pan et al., 2019]. Here, we denote a GAN as converged if the mean KID score does not vary by more than 1 for 10 continuous epochs.

3. Training Time: We also measure the time required for convergence to identify the overhead that xAI system adds to LogicGAN. We further separate the time required for the xAI system to better understand its effect on LogicGAN. Here we use the same definition of convergence as above.

5.3 MNIST Single Class Results

We performed testing on the MNIST dataset, first limiting the setting to a single class (selecting the digit 7). We observed during preliminary testing that for low learning rates, LogicGAN had similar FID scores as standard GAN and required the same amount of time for convergence, although it took more time to run due to the overhead of the xAI system. However, in the case of high learning rates, the standard GAN frequently resulted in mode collapse while LogicGAN was observed to converge and learn faster without suffering from the same. Thus, we picked the best version of each, i.e. LogicGAN with a high learning rate and standard GAN with low learning rate and compared the two.

The results of data efficiency metrics and time required for convergence for LogicGAN and standard GAN are given in Tables 1 and 2 respectively. To achieve convergence, the standard GAN required 20 epochs while LogicGAN only needed 11 epochs - resulting in an 45% improvement in data efficiency. However, we note that standard GAN needed 243.2 seconds while LogicGAN took 994.4 seconds of run time - out of which the xAI system required 642.3 seconds. We believe improvements in efficiency of xAI systems will help reduce this difference. LogicGAN was observed to produce good quality images faster than standard GAN. Figures 2 and 4 show images generated by LogicGAN and standard GAN at epoch 1 in batch 15/63. We can clearly see identifiable “7s” generated by LogicGAN here. Figures 3 and 5 show images generated by both GANs at convergence.

The FID scores for LogicGAN were observed to be very low (around 0.2) from epoch 1 while the scores in standard GAN required around 6 epochs to get to that value (see Figure 7). Further, both remained within a range of 0.1 till convergence. This reaffirms the fact that LogicGANs learn quicker and create better quality images sooner compared to standard GANs. Overall, LogicGAN resulted in an improvement in data efficiency compared to standard GANs while maintaining the same quality as measured by the FID score.

5.4 MNIST Multi Class Results

We next extended LogicGAN to multi-class setting by training a separate classifier to provide relevant background images to the xAI system. We used the entire MNIST dataset as training data for our experiment. We used the same parameters as in single-class setting.

Again, Tables 1 and 2 respectively outline the results of data efficiency metrics and time required for convergence for LogicGAN and standard GAN. We found that standard GAN took 55 epochs to converge while LogicGAN took 48 epochs - resulting in a 12.73% improvement in data efficiency. However, standard GAN required 27,735.27 seconds while LogicGAN needed 88,294.15 seconds (out of which xAI system required 56,885.49 seconds). Again, we note that this gap can be shortened as xAI systems get more efficient. Figures 6 and 8 show the images generated by LogicGAN and the standard GAN respectively at convergence. A visual inspection of the pictures yielded no significant differences between them.

As for quality measures, standard GAN was observed to start with a lower FID score, but after epoch 2 both GANs result in similar scores (see Figure 9). We viewed the images generated by standard GAN in those two epochs and they were close to random noise. Overall, LogicGAN is observed to be more data efficient as compared to standard GAN while maintaining the same quality scores here as well.
Figure 6: Multi-class LogicGAN samples at convergence

Figure 7: FID Values for single-class setting.

Table 1: Number of epochs required for LogicGAN and Standard GAN to converge for both single- and multi-class settings.

| Class Setting | Standard GAN | LogicGAN |
|---------------|--------------|----------|
| Single        | 20 epochs    | 11 epochs|
| Multi         | 55 epochs    | 48 epochs|

5.5 Discussion of Experimental Results

We performed extensive experiments in both single-class and multi-class settings using the MNIST dataset and showed that LogicGAN is significantly more data efficient compared to standard GANs while maintaining the same quality score. A crucial finding of our experiments is that LogicGAN is able to learn effectively even at higher learning rates, while standard GAN suffers from mode collapse with the same learning rates. There may be settings where a higher learning rate can be crucial, making this a suitable avenue for future research.

6 Future Work

Our results suggest LogicGAN can be leveraged in settings where data efficiency is important - such as where training data is limited or in privacy conscious settings where training with less data allows for increased privacy guarantees.

6.1 Controlling How Models Learn

While standard GANs only use one value of corrective feedback (loss) to the generator, there are many ways this feedback is used. For instance, several GANs vary the type of loss function [Metz et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Mao et al., 2017; Dziugaite et al., 2015; Zhao et al., 2016] and the selection of the optimizer (such as Adam, Stochastic Gradient Descent, etc.) to control how the model learns.

Similarly, we believe that the feedback provided by xAI system using LogicGAN - which is richer compared to only using the loss value - can allow for greater control over this learning process. This control can be applied in various ways, such as in selecting the type of xAI system to use, varying the parameters of the chosen xAI system, offsetting the mask $M$ to give more weight to xAI feedback, alternating between xAI-guided and standard generator training, and selecting methods to combine xAI feedback with loss. We argue that LogicGANs are a powerful way for users to gain greater control over the training process of GAN models, and that there are many avenues, applications, and extensions of this idea worth exploring in the future.

7 Conclusion

In this paper, we introduce LogicGANs, a class of generative adversarial network (GAN), that use an explainable AI (xAI) system to provide richer feedback from the discriminator to the generator to enable more guided training and greater control. We next overview xAI systems and standard GAN training and then introduce our xAI-guided generator training algorithm, contrasting it's difference with standard generator training. We perform experiments using MNIST dataset and show that LogicGAN has an improvement in data efficiency of 45% in single-class and 12.73% in multi-class setting as compared to standard GANs while maintaining the same quality score as measured by the Fréchet Inception Distance. To the best of our knowledge, LogicGAN is the first GAN to utilize xAI feedback for training. We further provide insights in section 6.1 on various ways to integrate this feedback into LogicGAN. Currently, xAI systems are developed and viewed to be “explainable to humans”. While this has its uses, we argue that there is a lot of potential in viewing and applying these systems to “explain to AI”. Ultimately, LogicGAN may enable more control over the GAN learning process - allowing for better performance as well as a better understanding of GAN learning.
References

[Arjovsky et al., 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

[Bińkowski et al., 2015] Mikolaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. arXiv preprint arXiv:1801.01401, 2018.

[Biran and Cotton, 2017] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In IJCAI-17 workshop on explainable AI (XAI), volume 8, page 1, 2017.

[Borji, 2019] Ali Borji. Pros and cons of gan evaluation measures. Computer Vision and Image Understanding, 179:41–65, 2019.

[Chen et al., 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Info-gan: Interpretable representation learning by information maximizing generative adversarial nets. In Advances in neural information processing systems, pages 2172–2180, 2016.

[Duchi et al., 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul):2121–2159, 2011.

[Dziugaite et al., 2015] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. arXiv preprint arXiv:1505.03906, 2015.

[Goodfellow et al., 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc., 2014.

[Gulrajani et al., 2017] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In Advances in neural information processing systems, pages 5767–5777, 2017.

[Heusel et al., 2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Advances in Neural Information Processing Systems, pages 6626–6637, 2017.

[Horn, 1990] Roger A Horn. The hadamard product. In Proc. Symp. Appl. Math, volume 40, pages 87–169, 1990.

[Ignatiev et al., 2019] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 1511–1519, 2019.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[LeCun and Cortes, 2010] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[Lucic et al., 2018] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In Advances in neural information processing systems, pages 700–709, 2018.

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems, pages 4765–4774, 2017.

[Makhzani et al., 2015] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. arXiv preprint arXiv:1511.05644, 2015.

[Mao et al., 2017] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 2794–2802, 2017.

[Metz et al., 2016] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2016.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

[Pan et al., 2019] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. IEEE Access, 7:36322–36333, 2019.

[Paszke et al., 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017.

[Paszke et al., 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, pages 8024–8035, 2019.

[Radford et al., 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[Shrikumar et al., 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 3145–3153. JMLR. org, 2017.

[Zhao et al., 2016] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126, 2016.