Report: Marginal GAN and GAN Range Space

Kellin Pelrine
Department of Economics
Yale University
New Haven, CT 06520
Kellin.Pelrine@Yale.edu

Abstract

This project has developed in two directions. First, there is the original project of marginal GAN: training GANs on individual classes instead of every class at once. Marginal GAN output looks reasonable, and quantitative results, particularly the class-aware fréchet distance [12], suggest it may be superior to normal GAN output. Second, observational data generated using a modification of the Defense-GAN algorithm [17] indicates that GAN range spaces can be much larger than one might expect. For instance, a GAN which seems to only output ones can output other numbers or shapes (e.g. a smiley face). This has implications for failure modes: just because a GAN seems to perform well on average does not mean it cannot generate extreme outliers.

1 Marginal GAN

I first discuss marginal GAN. Although the observations on GAN range spaces were motivated by unexpected marginal GAN output, the topics are not that closely related. Therefore, I split this report in two halves.

1.1 Introduction

Breaking problems down into more manageable chunks is a pervasive strategy. Humans do this for diverse problems such as learning to write (letters, words, sentences...), organization in business (different employees and teams with different tasks), and many more. In contrast, GANs are typically used in a much more unified manner, with one GAN trained to produce the entire target distribution. The idea of Marginal GAN is to take a simple approach to breaking down the GAN learning problem, by training GANs on individual classes or subsets of classes separately and combining their output as needed for a full GAN.

Beyond the general goal of improving GAN output, there are two more reasons this could be useful. First, this automatically avoids mode collapse where a GAN omits one or more classes. Second, depending on the downstream application of the GAN, output may need a very clear class. For example, an original motivation for this project was how Defense-GAN [17] may convert a noisy image into one from the wrong class, thus causing problems for classification.

I tested Marginal GAN on MNIST and FMNIST. The results (see below) suggest it performs better than a normal GAN. These datasets are not the most complex, so further testing is warranted, but so far it appears this method is worth considering, particularly in cases where one does not mind training multiple GANs if it can improve the quality of the output.

Preprint. Work in progress.

1.2 Literature

The general idea of somehow combining multiple GANs is not new, but surprisingly the direct approach of training different GANs on different classes has not been tested. AdaGAN [18] explicitly rejects a related approach (excluding classes where one is doing well from further training) in favor of a boosting method that works on unlabeled data. Although this is indeed more general, in many situations we have accurate classifiers. Further, [5, 7, 8] suggest that using labels improves performance, "almost always" and regardless of the way they are used [7]. Therefore, ideas that could yield improvements in the labeled data context are worth studying.

CGAN [14] takes a step in the direction of training GANs per-class. They pass labels to both the generator and discriminator. In the generator this label augments the random noise input. However, aside from the input, the network is fully combined.

After CGAN, there have been many GAN designs that use classes in some form. For example, AC-GAN [15] includes a classifier in the discriminator, and trains both discriminator and generator to maximize the chance of the correct class. Splitting GAN [8] creates classes through clustering.

Besides AdaGAN, papers using multiple GANs include: MIX + GAN [1], which corresponds to allowing the generator to play mixed strategies by mixing different generators. The generators are not class-specific. MGAN [11], which uses multiple generators with shared parameters. MAD-GAN [6], which trains the discriminator to determine which generator an output came from, which is intended to push the generators to different modes. MIX + GAN and MAD-GAN both are designed to avoid mode collapse, but they don't seem to consider the simpler strategy of training GANs on data that fewer modes to begin with and combining afterwards.

1.3 Experimental Results

This project originally was connected to Defense-GAN [17]. Although due to the range space observations I abandoned the plan of using their framework to test marginal GAN effectiveness (see additional details in the range space section), I still use their basic code and network, which is available at `https://github.com/kabkabm/defensegan`.

I trained a normal GAN on all classes, and then trained marginal GANs on each class by only loading the data corresponding to that class. All GANs used the original Defense-GAN hyperparameters. They were trained for 24 hours with 20 CPU cores, per GAN, using the Grace cluster.

Some sample images are shown in figures 1 and 2. Qualitatively, the marginal GAN output appears reasonable, with no clear winner between it and the original GAN.

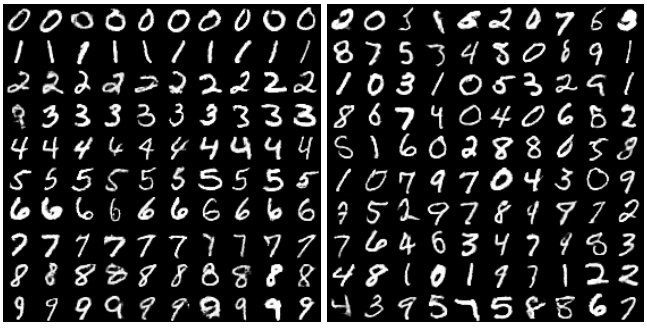

Figure 1: MNIST. Left: marginal GAN. Right: original WGAN

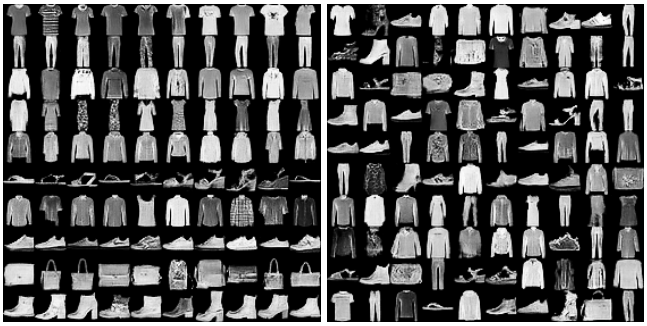

Figure 2: FMNIST. Left: marginal GAN. Right: original WGAN

Quantitative evaluation is tricky for GANs because there is no single definitive metric for evaluating GAN performance. The Inception Score (IS) [16] is "perhaps the most widely adopted" metric [4]. It weights classifiability of the output against its diversity (intuitively, one would like output that reflects the characteristics of the classes but is still original). However, multiple papers [4, 3, 10] note issues or limitations with this measure.

Another common method is Fréchet Inception Distance (FID) [10]. FID compares the distance between real and generated data by embedding samples in an estimated feature space. This is done by representing them as activations of the coding layer of Inception net. FID then fits Gaussians to the respective vectors (one Gaussian to the real ones, one to the generated ones) and compares their Fréchet distance. It is "consistent with human judgments and is more robust to noise than IS" [4]. A recent large-scale empirical study of (unconditional) GAN performance [13] used FID, and provides evidence FID "is a reasonable metric." However, the single Gaussian model is restrictive, and it is not clear why ImageNet-based Inception is the right choice for other datasets.

Class-Aware Fréchet Distance [12] refines FID by using a domain-specific classifier instead of Inception, and modeling the distributions of the real and generated data as mixtures of Gaussians corresponding to the different classes. It increases robustness compared with FID [12]. With the focus on class accuracy of marginal GAN, this metric seems the most relevant.

Results are shown in tables 1 and 2. The CAFD is calculated with 10000 images using code from /urlhttps://github.com/B1ueber2y/CAFD. The FID and IS are substantially slower and are thus calculated using only 1000 images (but multiple trials indicate this is sufficient, with results varying in a small range). Note that classes in MNIST are not exactly uniformly distributed. This doesn't affect the CAFD (because it separates classes anyways) but is corrected for in the FID by taking 1000 images randomly according to the class distribution [9] (though this correction does not seem to have a substantial impact). Note that lower CAFD and FID is better, and higher IS.

Table 1: Quantitative Results: MNIST

Name	CAFD	FID	IS
Original GAN	107.4	40.1	2.23
Marginal GAN	90.7	39.7	2.14

Table 2: Quantitative Results: FMNIST

Name	CAFD	FID	IS
Original GAN	42.9	107.8	4.16
Marginal GAN	35.8	110.8	4.25

The FID and IS are similar between the two GANs, with no clear winner. On the other hand, the CAFD is significantly better for marginal GAN. As argued above, this metric seems like the best way to evaluate the output here. Therefore, although these results are not enough to be conclusive, they

indicate that marginal GAN may yield improvements compared to training GANs in the standard way.

1.4 Conclusion

The idea of marginal GAN is quite simple and fairly intuitive, so it is surprising that it has not been tested. Perhaps the main reason is that it may be computationally expensive. However, it is embarrassingly parallel, since training the individual GANs is completely independent. The results indicate it may be superior to training a GAN normally. Therefore, in applications where one computing power available and needs the best GAN output possible this method is worth considering.

A clear path to extend this work is to experiment with more datasets. I also only tested one GAN per class, so it would be interesting to see if something in between this and normal GAN, training GANs on particular subsets of classes, could also yield improvements. Finally, since computational cost is relevant here, it would be interesting to test what happens as the training time is reduced. If it turned out that one can train each marginal GAN faster than a normal GAN, then this method could also be useful in situations where there is a significant cost to real-world as well as computational time.

2 GAN Range Space

2.1 Introduction

Originally, I intended to test marginal GAN in the framework of Defense-GAN, which tries to defend a classifier from adversarial noise by replacing the noisy image with the closest one from a GAN's range space. However, while testing a GAN that had only been trained on ones in this algorithm, I noticed it appeared to produce images from all classes.

Further testing indicates that what GANs produce on average is not indicative of what they can produce rarely, and the Defense-GAN algorithm can find these rare examples. This suggests that Defense-GAN may be gradient masking [2], providing an unreliable defense, because it can reproduce too much beyond the intended output. Moreover, it suggests that good performance according to standard qualitative and quantitative measures, like those described in the marginal GAN experiment section above, will not necessarily detect extreme outliers. If a GAN is to be applied in a situation where a rare outlier can have a very high cost, either new metrics that can detect extreme outliers, substantial testing, or downstream robustness is needed.

2.2 Defense-GAN Range Space Search Algorithm

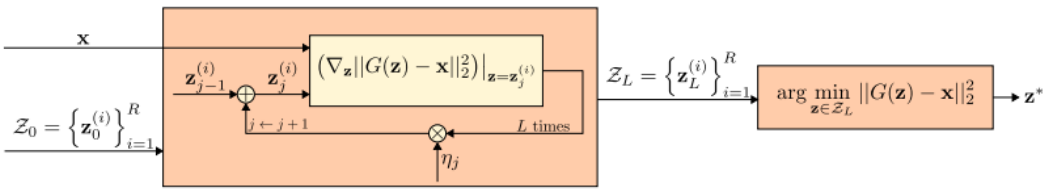

Figure 3: Defense-GAN algorithm. Figure 2 in the Defense-GAN paper [17]

The part of Defense-GAN which searches the range-space works according to the figure above. It searches for an image that is closest to an input image x by performing L steps of gradient descent. It does this R times starting from different random noise samples, then takes the best. Below I use L = 200 and R = 10, which are the default parameters for Defense-GAN.

2.3 Experimental Observations

I used the selection of input images in Figure 4 to examine the range space. Some are from MNIST (one with noise), some from FMNIST, and the rest simple images constructed for this test.

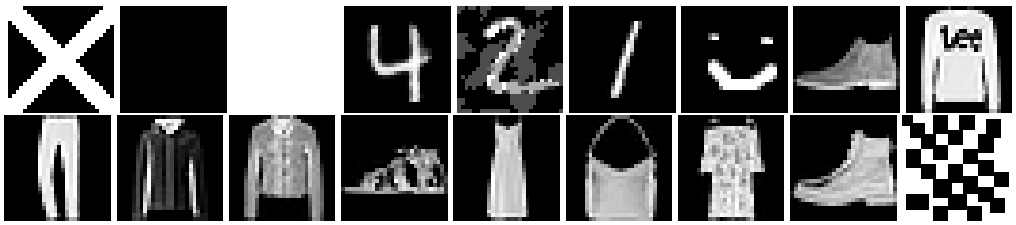

Figure 4: input images

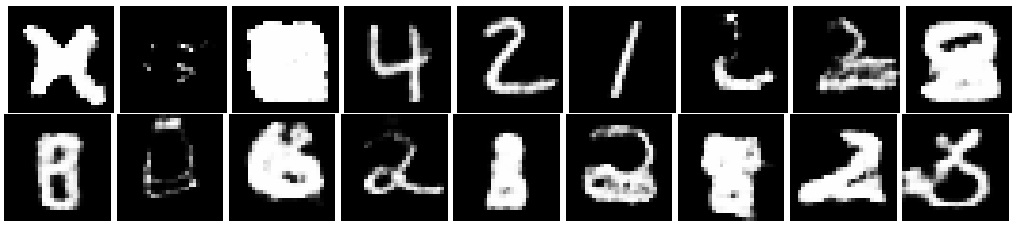

Figure 5: output from original MNIST GAN

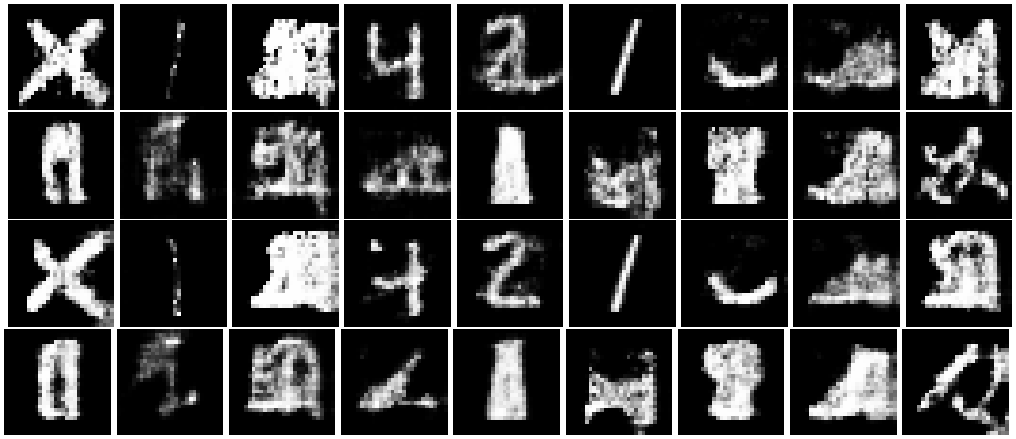

Figure 6: output from ones marginal GAN. Above: full network. Below: reduced parameters.

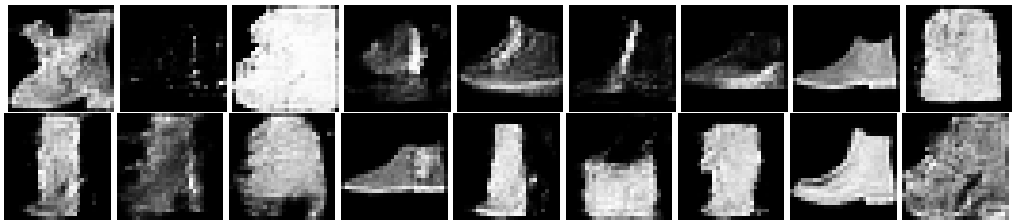

Figure 7: output from original FMNIST GAN

In the output in figures 5 and 6, one can see multiple images that do not look like numbers. Likewise in figure 7, there are images that don't look like cloths. The first 3 images in the top row provide a particularly clear illustration. The first two rows of Figure 6 are from a marginal GAN trained only on ones with the usual number of parameters. The second two rows are the same training set but with the number of parameters in the two layers (the largest two) divided by 4 and 2. Reducing the parameters, at least this amount, does not seem to have much effect on the range space, though further testing is needed.

Note that the last image appears to be the hardest to replicate. For the MNIST GANs, many images have a clear or number-like pattern. This means that the GAN's range is approximating a space with only number images, just not that well (there is a lot of other stuff too). A similar observation holds for the FMNIST GAN. There is also substantial difference between MNIST and FMNIST GANs. Although the range is bigger than might be desirable, it is clear that it is not just any image at all. These factors suggest that with sufficient adjustments one may be able to reduce the range space to contain fewer bad images, though further testing is needed.

2.4 Conclusion

It appears that the range space search from Defense-GAN is effective for exploring the limits of the space. The critical result here is not just that the range is large but that it can be large even when the output usually appears perfectly reasonable (e.g. figure 1). If one is concerned a GAN might producing outlying images, checking a few samples or one of the common quantitative metrics is insufficient.

From these observations, it is not completely clear how to mitigate this issue. One possibility is to further reduce the parameters in the network. Another is that modifying the noise, e.g. by reducing the variance or bounding the tails, could reduce the chance of outliers or avoid them entirely. Finally, one could also consider metrics that severely penalizes outliers (with some suitable mathematical representation) for checking robustness. A standard metric such as FID or CAFD scaled to penalize large deviations more might be effective.

3 Some notes on replication

The files sent with this report, along with the original Defense-GAN code, include the code and data to produce these results. They are not in a particularly user-friendly format, but should be everything needed to replicate these results.

The files the produce the metrics are cafd.py and process_fid_is.py. They need the path to the desired saved images, which are in the folders "output for CAFD - xx." These will run without Defense-GAN.

The file that reproduces the range space images is reconstruct_test.py. It's based on Defense-GAN's blackbox.py (and therefore includes some unnecessary flags). It requires Defense-GAN. To use it, first one needs to specify the number of images in "batch_size." They should be named ximg (though this can be changed without much difficulty), where the brackets are replaced by a number starting from 1 up to batch_size, and should be .png files stored in the same directory as reconstruct_test itself. There may be bugs if they are not 28x28. The file should be called with flags –cfg path -debug, where path specifies a folder with a GAN save. These are included in "mnist gan save" and "fmnist gan save" folders.

Note that running the original Defense-GAN will require old versions of Tensorflow and Cleverhans. 1.7.0 and 2.1.0 respectively seem to work (with some warning messages). There is a typo in their "requirements.txt" file; tqdm=4.28.1 should be tqdm==4.28.1. There is an error in "download_dataset.py" which prevents it from downloading FMNIST; I've included a corrected version. There is a typo in "gan.py" which causes "train.py –cfg path -test_generator" - which saves images from the GAN - to fail for FMNIST. I have included a fixed version, which also includes code to save images separately rather than in one file for the purposes of processing the evaluation metrics above. It is the lines under MnistDefenseGAN.generate_image that are currently commented out; to use them, uncomment those and comment the tflib.save_images.save_images lines directly below (similarly for FMNIST). As is this will save 128 images; this can be changed by replacing the "128" in self.fixed_noise, under _build, with the desired number. Note that changing the number may crash tflib.save_images.save_images, but will work when saving images individually.

Cafd.py also produces a KL divergence measure. This measures the difference in the distribution of class labels. It will appear worse for MNIST marginal GAN because the distribution of classes loaded in that file follows a uniform distribution of classes (which does not match the actual data, but also does not affect the CAFD negatively). It's therefore not reported above because it's irrelevant. The IS also produces a standard error. I don't report it because I don't have standard errors for the other metrics, but it is in accordance with the result stated above that the IS does not give reason to prefer either original or marginal GANs.

On my computer, running cafd with 10000 images completes within a couple minutes.FID and IS with 1000 images take roughly 1500 seconds each.

References

[1] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and Equilibrium in Generative Adversarial Nets (GANs). 2017. URL https://arxiv.org/pdf/1703.00573.pdf.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. 2018. URL https://arxiv.org/pdf/1802.00420.pdf.

[3] S. Barratt and R. Sharma. A Note on the Inception Score. 2018. URL https://arxiv.org/pdf/1801.01973.pdf.

[4] A. Borji. Pros and Cons of GAN Evaluation Measures. 2018. URL https://arxiv.org/pdf/1802.03446.pdf.

[5] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. 2015. URL https://arxiv.org/pdf/1506.05751.pdf.

[6] A. Ghosh, V. Kulharia, V. Namboodiri, P. H. Torr, and P. K. Dokania. Multi-Agent Diverse Generative Adversarial Networks. 2018. URL https://arxiv.org/pdf/1704.02906.pdf.

[7] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. 2017. URL https://arxiv.org/pdf/1701.00160.pdf.

[8] G. L. Grinblat, L. C. Uzal, and P. M. Granitto. Class-Splitting Generative Adversarial Networks. 2018. URL https://arxiv.org/pdf/1709.07359.pdf.

[9] M. Hamidi and A. Borji. Invariance analysis of modified c2 features: case study - handwritten digit recognition. 2007. URL file:///E:/Downloads/Invariance_analysis_of_modified_C2_features_Case_s.pdf.

[10] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. 2018. URL https://arxiv.org/pdf/1706.08500.pdf.

[11] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung. MGAN: training generative adversarial nets with multiple generators. 2017. URL https://arxiv.org/pdf/1708.02556.pdf.

[12] S. Liu, Y. Wei, J. Lu, and J. Zhou. An Improved Evaluation Framework for Generative Adversarial Networks. 2018. URL https://arxiv.org/pdf/1803.07474.pdf.

[13] M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly. Are GANs Created Equal? A Large-Scale Study. 2018. URL https://arxiv.org/pdf/1711.10337.pdf.

[14] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. 2014. URL https://arxiv.org/pdf/1411.1784.pdf.

[15] A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. 2017. URL https://arxiv.org/pdf/1610.09585.pdf.

[16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. 2016. URL https://arxiv.org/pdf/1606.03498.pdf.

[17] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: protecting classifiers against adversarial attacks using generative models. 2018. URL https://arxiv.org/pdf/1805.06605.pdf.

[18] I. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Scholköpf. AdaGAN: Boosting Generative Models. 2017. URL https://arxiv.org/pdf/1701.02386.pdf.