C6.5 mini project: addressing mode collapse in GANs

Simon Martina-Perez
martinaperez@maths.ox.ac.uk

Abstract

Generative Adversarial Nets (GANs) are popular for their capacity to create output that seems indistinguishable from real-world data for humans. Some GANs are known to exhibit *mode collapse*, which occurs when the distribution of GAN outputs fail to capture the distribution of the real-world test data. In this mini project, we review strategies employed to combat mode collapse and metrics to assess GAN performance. In the final section, we consider the performance of a GAN a multimodal toy dataset and find that the dimension of the latent variables is related to observing mode collapse in this problem. Code and animations can be found at `https://github.com/simonmape/modecollapse`.

1 Introduction

Generative Adversarial Nets (GANs) were proposed by Goodfellow *et al.* [4] as a framework for estimating generative models. In recent years, GANs have gained great popularity for their capacity to create output that is seemingly indistinguishable from real-world data to a human observer, leading to many applications, ranging from creating realistic images of human faces [8] to high-fidelity weak lensing convergence maps in cosmology [13].

The main idea behind GANs is that they consist of two parts: a *generator*, G, and a *discriminator*, D. The inputs of G are *noise variables* or *latent variables* $\mathbf{z}$, whose distribution is specified by a prior $p_{\mathbf{z}}$ and its outputs are samples in the data space. The *discriminator*, D, estimates the probability that a sample $\mathbf{x}$ comes from the training data instead of being generated by G. As noted by Goodfellow, "[t]he training procedure for G is to maximize the probability of D making a mistake" until D cannot distinguish between the data and the outputs of G, with other words, $D(x) = \frac{1}{2}$ for all outputs $\mathbf{x}$. This is done by optimizing the objective function $V(D, G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log(D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\log(1 - D(G(\mathbf{z})))\right]. \tag{1}$$

In most applications, G and D are deep nets [18], and the whole system can be trained using backpropagation [4]. In [3], Goodfellow notes that the aim is to take a training set of samples drawn from a distribution p_{data} and approximate that distribution. The resulting estimate is a probability distribution denoted by p_{model}. While in some cases, the goal of generative models is to estimate p_{data} directly, this is often not possible or feasible since the data are sampled from a high-dimensional or infinite-dimensional manifold, and the model is limited to generating new samples. GANs focus primarily on sample generation [3], though "it is possible to design GANs that can do both".

This mini project will focus on the theoretical issue of *mode collapse* in GANs. Mode collapse refers to an important shortcoming of GANs – and generative models more generally – which is that they often "fail to model the entire data distribution" [18]. This could mean specifically that the GAN outputs are merely perturbations around some point or points in the data manifold, but in more general terms it is when p_{model} fails to approximate p_{data} [3]. This becomes problematic when the GAN's specific application needs that p_{model} is somehow close to p_{data}. This is why Goodfellow notes that "[b]ecause of the mode collapse problem, applications of GANs are often limited to problems where it is acceptable for the model to produce a small number of distinct outputs" [3]. The issue of

mode collapse has invited many contributions to the field, which I will showcase in two *roughly* separate categories in this project. The first involves methods to address mode collapse in GANs through choice of network architecture, metrics and other choices in the training of the GAN. The second concerns how to evaluate the performance of the GAN in approximating p_{data}, where we are particularly interested in how to quantify the "diversity of the generated samples, regardless of the data distribution" [18].

2 Proposed solutions for the mode collapse problem

Some authors propose changing the metric used to arrive at the objective function in Equation (1). To quantify the divergence between p_{model} and p_{data}, there are alternatives to the Jensen-Shannon distance – which is minimized through Equation (1) – such as the Kullback-Leibler divergence, the total variation distance and the Wasserstein distance. Minimizing these divergences leads to different objective functions. In addition, theoretical considerations suggest some choices of divergence are preferable – see [1]. Wasserstein GANs [1] are motivated by optimal transport theory and minimize the Earth Mover or Wasserstein distance. Informally, the idea is to quantify how much *mass* must be transported in order to transform p_{model} into p_{data} [1]. The objective function is the "cost" of optimal transport, which – omitting some details – due to the Kantorovich-Rubinstein duality gives the optimization problem

$$\min_G \max_D W^{(D)}(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[D(\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[D(G(\mathbf{z}))\right]. \tag{2}$$

Another option is f-GANs, proposed by Nowozin *et al.* [14]. They propose a variational divergence minimization problem with divergence $D_f(P\|Q) = \int q(x) f\left(p(x)/q(x)\right) dx$, where $f : \mathbb{R}^+ \to \mathbb{R}$ is a convex, lower-semicontinuous function satisfying $f(1) = 0$. Different choices of f recover different known divergences. Here, again, the variational divergence translates into a different objective function to be used. Energy Based GANs (EBGANs) [22], "[view] the discriminator as an energy function that attributes low energies to the regions near the data manifold and higher energies to other regions" [22]. Interestingly, EBGAN's discriminator is an autoencoder, as is common in energy based models [22]. Other implementations similar in vein are implementing a least squares metric to obtain LSGANs [11] or absolute deviation with margin [22].

Goodfellow is critical of the idea that mode collapse is driven by the choice of divergence [3]. On the question if the choice of divergence is a distinguishing feature of GANs, he notes new evidence suggests that GANs often "prefer to choose to generate from very few modes: fewer than the limitation imposed by the model capacity". "The reverse KL prefers to generate from as many modes of the data distribution as the model is able to and does not prefer fewer modes in general". [3] This suggests that the mode collapse is driven by a factor other than the choice of divergence.

Radford *et al.* [17] introduce Deep Convolutional GANs (DCGANs) and propose a convolutional architecture to train G and D. DCGANs allow building image representations by training GANs and partially reusing G and D as feature extractors for supervised tasks. In short, the proposal in [17] amounts to replacing any pooling layers with strided convolutions for D and fractional-strided convolutions for G, using batch normalization, removing fully connected hidden layers, using ReLU activation in G for all layers except for the output, which uses Tanh, and using LeakyReLU activation in D for all layers [17]. Radford *et al.* show that the networks exhibit little memorization of the test data. Furthermore, the network allows the authors to visualize that the space learnt has smooth transitions and that every image generated looks like a bedroom. They are also able to visualize the different filters learnt by GANs and show that specific filters have learned to draw specific objects.

Srivastava *et al.* [20] introduce a Variational Encoder Enhancement to Generative Adversarial Networks (VEEGAN). The main idea is to train an additional network, $F_{\tilde{\theta}}$, called the *reconstructor*, to map p_{data} to a distribution over the latent variables and to approximately invert the generator network. The intuition for choosing an inverter is that if $F_{\tilde{\theta}}$ maps both the true data and the generated outputs to $p_{\mathbf{z}}$ – in [20] and practically everywhere else a Gaussian distribution – then the generated data is likely to coincide with true data [20]. Measuring if $F_{\tilde{\theta}}$ approximately inverts the generator can be done naturally using an antoencoder loss. To assess whether $F_{\tilde{\theta}}$ maps p_{data} to $p_{\mathbf{z}}$, Srivastava *et al.* [20] use the cross entropy $H(\mathbf{z}, F_{\tilde{\theta}}(\mathbf{x}))$. Hence, learning θ, the parameter for the

generator and $\tilde{\theta}$ for the inverter happens by minimising

$$\mathcal{O}_{\text{entropy}}(\theta, \tilde{\theta}) = \mathbb{E}\left[\left\|\mathbf{z} - F_{\tilde{\theta}}\right\|_2^2\right] + H\left(\mathbf{z}, F_{\tilde{\theta}}(\mathbf{x})\right). \tag{3}$$

Equation (3) cannot be easily computed and minimised, so a computable version with theoretical guarantees is used [20]

$$\mathcal{O}(\theta, \tilde{\theta}) = \text{KL}\left[q_\theta(\mathbf{x}|\mathbf{z})p_0(\mathbf{z})\|p_{\tilde{\theta}}(\mathbf{z}|\mathbf{x})p(\mathbf{x})\right] - \mathbb{E}\left[\log p_0(\mathbf{z})\right] + \mathbb{E}\left[d(\mathbf{z}, F_{\tilde{\theta}}(\mathbf{x}))\right], \tag{4}$$

where q_θ is a variational distribution, which can be approximated using a neural network, p_0 is a standard normal distribution, $p = p_{\text{data}}$, and $p_{\tilde{\theta}}$ gives the distribution of the outputs of the reconstructor network when applied to fixed $\mathbf{x}$ [20]. It is shown that VEEGAN, like unrolled GAN – to be discussed next – and unlike other GAN implementations, is successful in preventing mode collapse when training data consist of mixtures of Gaussians arranged in a ring or a grid.

Unrolled GANs were proposed by Metz *et al.* to address the mode collapse problem [12]. The unrolling works as follows. As noted by Metz *et al.* [12], a local optimum of discriminator parameters θ_D^* can be expressed as the fixed point of the iterative optimization procedure

$$\theta_D^0 = \theta_D$$
$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{df\left(\theta_G, \theta_D^k\right)}{d\theta_D^k}$$

and $\theta_D^* = \lim_{k\to\infty} \theta_D^k$, where η is the learning rate. Of course the iterative optimization need not be full-batch. Unrolling for K steps amounts to creating a *surrogate objective* for the update of G:

$$f_K(\theta_G, \theta_D) = f\left(\theta_G, \theta_D^K(\theta_G, \theta_D\right). \tag{5}$$

K = 0 retrieves the GAN objective, but when $K \to \infty$, the solution corresponds to the true generator objective function. By K, then, one can interpolate between standard GAN training and more costly gradient descent on the true generator loss [12]. The intuition for why this works, according to Metz *et al.*, is that "G's actions take into account how D will respond. In particular, G will try to make steps that D will have a hard time responding to. This extra information helps G spread its mass to make the next D step less effective instead of collapsing to a point." [12]. For details, we refer to [12]. In the implementation of [12], Adam is unrolled and it is shown that unrolled GAN has excellent performance in retrieving the multimodal Gaussian mixture discussed in the case of VEEGAN.

Lastly, we discuss MMD GAN, proposed by Li *et al.* [9]. The idea is to use a kernel to estimate the distance between two observed distributions. Distinguishing two distributions by finite samples is known as Two-Sample Testing and one way to conduct such a two-sample test is via kernel maximum mean discrepancy (MMD) [9]. Given a kernel k, the square MMD distance is defined as

$$M_k(\mathbb{P}, \mathbb{Q}) = \mathbb{E}_{\mathbb{P}}\left[k(x, x')\right] + 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}\left[k(x, y)\right] + \mathbb{E}_{\mathbb{Q}}\left[k(x, x')\right] \tag{6}$$

This loss function is appealing due its very simple implementation and computability and can be directly substituded into the minimax problem. Li *et al.* show that when trained on the MNIST data set, MMD GAN "generate[s] diversified digits, avoiding the mode collapse problems appeared in the literature of training GANs." [9]. As for performance on the CelebA data set, Li *et al.* "observe varied poses, expressions, genders, skin colors and light exposure" [9].

3 Evaluation metrics for GANs

The literature offers several distinct metrics to evaluate the performance of GANs. The first commonly used metric to do so is the *Inception Score* (IS) [19]. IS uses a third-party neural network trained on a supervised classification task to evaluate performance by computing the expectation of divergence between the distribution of class predictions for samples from the GAN compared to the distribution of class labels used to train the third-party network:

$$\exp\left(\mathbb{E}_{\mathbf{x}\sim Q_\theta}\left[\text{KL}(p(y|\mathbf{x})\|p(y))\right]\right), \tag{7}$$

where the class prediction given a sample $\mathbf{x}$ is computed using the third-party neural network [7]. In [19], Google's Inception Network trained on ImageNet was the third-party neural network [7].

While IS is a very popular metric, implementation can be difficult as it requires an additional neural network trained separately [7]. In addition, Im *et al.* show that IS can fail [7]. A direct extension of IS is the *Fréchet Inception Distance* (FID) [5], which "instead of using the final classification outputs from the third-party network as representations of samples, uses a representation computed from a late layer of the third-party network. It compares the mean m_Q and covariance C_Q of the Inception-based representation of samples generated by the GAN to the mean m_P and covariance C_P of the same representation for training samples" [7].

Other metrics include the *Generative Adversarial Metric* [6], which "measures the relative performance of two GANs by measuring the likelihood ratio of the two models" [7]. Important to note is the caveat that "the two discriminators must have an approximately similar performance on a calibration dataset, which can be difficult to satisfy in practice" [7]. *Classifier Two-Sample Tests* (C2ST) [10] depend on a classifier to distinguish real samples from p_{data} and generated samples from p_{model} and use the error rate of the classifier as a metric for GAN performance. In [10], Lopez *et al.* use single-layer and k-nearest neighbor classifiers. Finally, Im *et al.* propose to use new metrics, based on the formulations for divergence of p_{model} and p_{data} seen in Section 2, including Wasserstein distance, KL divergence and least squares [7]. In addition, Im *et al.* compare all of the above metrics by "training and comparing multiple types of GANs with multiple architectures on multiple data sets" and found that "the different proposed metrics still agreed on the best GAN framework for each dataset" [7].

It is worth noting two special metrics to quantify the diversity of the generated samples. The first concerns the *Birthday paradox test* proposed by Arora *et al.* [2]. This test is based on the birthday paradox: a sample of size $\sim \sqrt{N}$ from a discrete distribution with support of size $\sim N$ will have duplicates with high probability. Hence, Arora *et al.* propose: "(a) Pick a sample of size s from the generated distribution. (b) Use an automated measure of image similarity to flag the 20 (say) most similar pairs in the sample. (c) Visually inspect the flagged pairs and check for duplicates. (d) Repeat." [2]. Thus, if the test reveals that samples of size N very likely have duplicates, then the distribution it is sampling from has support size about N^2, which gives an indication of the sample diversity. Arora *et al.* perform this test on GANs known to exhibit mode collapse and tentatively suggest that "the size of the distribution's support scales near-linearly with discriminator capacity" [2]. Another popular metric, used in the evaluation of GANs whose outputs are images, is the *Multiscale Structural Similarity* metric (MS-SIM) proposed by Wang *et al.* [21]. The idea is to take two images as input and iteratively apply a filter and subsequently down-sample the filtered image by a factor of 2. At each scale, a contrast and structure comparison are calculated and the measurements at different scales are combined to give a single score [21].

4 Numerical simulations

For a simple demonstration of mode collapse in a GAN, we employ the *Moons* toy data set in Python's scikit-learn module [16]. This is a simple data distribution, where the data is concentrated around two *modes*: as seen in Figures 1 and 2, the data is distributed around two half-circles in the plane. We will show with a simple numerical experiment that changing the network architecture to allow for enough expressivity can prevent mode collapse when training a GAN on the moon data set.

We perform the numerical simulations using PyTorch [15]. We choose cross-entropy loss for the objective function and train two different models. In the first, we let the dimension of the latent variables be equal to 2, and let the latent variables be independent standard normally distributed. We build the generator as a fully connected network with 2 hidden layers, with width 8 and 4 respectively, all with ReLU activation. We build the discriminator as a fully connected network with 2 hidden layers, with width 64 and 8 respectively with Tanh activation and sigmoid nonlinearity after the final layer. Weights and biases are initialized with standard normal initialization. We choose a batch size of 32 and learning rate of 0.0005. Furthermore, we use stochastic gradient descent in PyTorch for the training of both generator and discriminator. We train for 150000 epochs to ensure convergence. We note here that the simulations here could have been more efficient. For instance, the network architecture could have been optimized, and this may have an effect on the mode collapse observed. Equally, the choice for the nonlinear activations could also have had an effect. The optimal training rate was not determined through cross-validation and this could have improved performance too.

For visualization, we generate a random draw of size 500 of the latent variables every 250 epochs and plot the predicted values together with the half-moons. Figure 1 shows the simulations at different epochs. We can see that the network fails to approximate the true shape of the distribution, as the distribution is fully connected and does not distinguish between the two clear modes. Note that the half moons were generated without any noise, by selecting noise equal to 0 in the scikit moon function. We see that there is mode collapse, as the distribution is concentrated around just one mode.

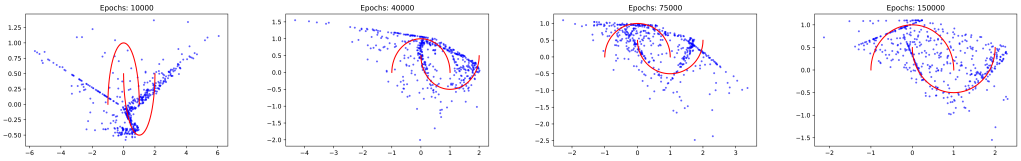

Figure 1: Network trained using latent dimension 2, the network fails to learn the appropriate shape of the distribution. From left to right: after 10000 epochs, after 40000 epochs, after 75000 epochs and after 150000 epochs.

We repeat the experiment above, now with latent dimension equal to 4, leaving all other training parameters identical. The density at different epochs is given in Figure 2. We can see that the network is now better able to distinguish between the different modes of the distribution, even though the separation is not very good.

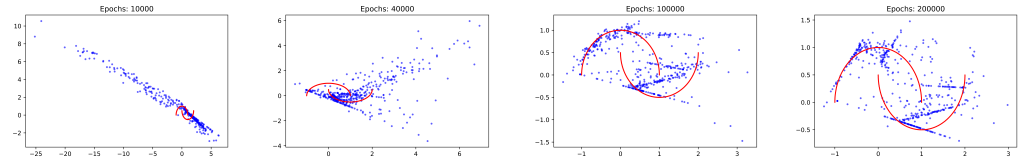

Figure 2: Network trained using latent dimension 4, the network is better able to capture the appropriate shape of the distribution. From left to right: after 10000 epochs, after 40000 epochs, after 100000 epochs and after 200000 epochs.

A Jupyter notebook with the code used for the above plots can be found at `https://github.com/simonmape/modecollapse`, where the output animates the distribution as it changes in time.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875.

[2] Sanjeev Arora and Yi Zhang. "Do GANs actually learn the distribution? An empirical study". In: *CoRR* abs/1706.08224 (2017). arXiv: 1706.08224. URL: https://dblp.org/rec/journals/corr/AroraZ17.bib.

[3] Ian J. Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *CoRR* abs/1701.00160 (2017). arXiv: 1701.00160.

[4] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2672–2680.

[5] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium". In: *CoRR* abs/1706.08500 (2017). arXiv: 1706.08500.

[6] Daniel Jiwoong Im et al. "Generating images with recurrent adversarial networks". In: *CoRR* abs/1602.05110 (2016). arXiv: 1602.05110. URL: http://arxiv.org/abs/1602.05110.

[7] Daniel Jiwoong Im et al. "Quantitatively Evaluating GANs With Divergences Proposed for Training". In: *CoRR* abs/1803.01045 (2018). arXiv: 1803.01045. URL: http://arxiv.org/abs/1803.01045.

[8] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948. URL: http://arxiv.org/abs/1812.04948.

[9] Chun-Liang Li et al. "MMD GAN: Towards Deeper Understanding of Moment Matching Network". In: *CoRR* abs/1705.08584 (2017). arXiv: 1705.08584. URL: http://arxiv.org/abs/1705.08584.

[10] David Lopez-Paz and Maxime Oquab. *Revisiting Classifier Two-Sample Tests*. 2018. arXiv: 1610.06545 [stat.ML].

[11] Xudong Mao et al. "Multi-class Generative Adversarial Networks with the L2 Loss Function". In: *CoRR* abs/1611.04076 (2016). arXiv: 1611.04076.

[12] Luke Metz et al. "Unrolled Generative Adversarial Networks". In: *CoRR* abs/1611.02163 (2016). arXiv: 1611.02163. URL: http://arxiv.org/abs/1611.02163.

[13] Mustafa Mustafa et al. "CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks". In: *Computational Astrophysics and Cosmology* 6.1 (May 2019). ISSN: 2197-7909. DOI: 10.1186/s40668-019-0029-9. URL: http://dx.doi.org/10.1186/s40668-019-0029-9.

[14] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. *f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization*. 2016. arXiv: 1606.00709 [stat.ML].

[15] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[16] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[17] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[18] Eitan Richardson and Yair Weiss. "On GANs and GMMs". In: *CoRR* abs/1805.12462 (2018). arXiv: 1805.12462. URL: http://arxiv.org/abs/1805.12462.

[19] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[20] Akash Srivastava et al. *VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning*. 2017. arXiv: 1705.07761 [stat.ML].

[21] Z. Wang, E. P. Simoncelli, and A. C. Bovik. "Multiscale structural similarity for image quality assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*. Vol. 2. 2003, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.

[22] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. "Energy-based Generative Adversarial Network". In: *CoRR* abs/1609.03126 (2016). arXiv: 1609.03126.