VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning

Code

Akash Srivastava akash.srivastava@ed.ac.uk

Lazar Valkov L.Valkov@sms.ed.ac.uk

Chris Russell crussell@turing.ac.uk

Michael U. Gutmann Michael.Gutmann@ed.ac.uk

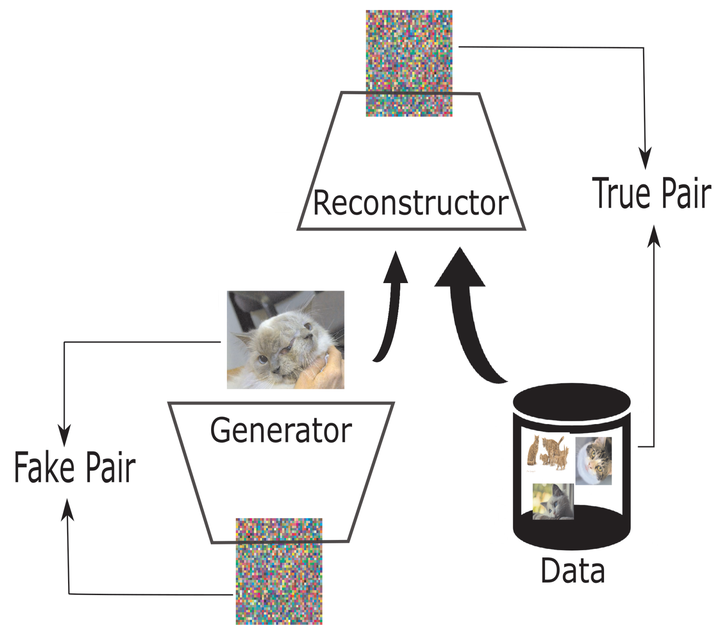
Charles Sutton csutton@ed.ac.uk

SETUP

- $\{\boldsymbol{x}_i\}_{i=1}^N$ represents training data where each $\boldsymbol{x}_i \in \mathbb{R}^D$ is drawn from an unknown distribution $p(\boldsymbol{x})$.
- A GAN is a neural network $G\gamma$ that maps representation vectors $z \in \mathbb{R}^K$, typically drawn from a standard normal distribution, to data items $x \in \mathbb{R}^D$.
- Because this mapping defines an implicit probability distribution, training is accomplished by introducing a second neural network D_ω, called a discriminator, whose goal is to distinguish samples from the generator to those from the data.
- The parameters of these networks are estimated by solving the minimax problem

$$\max_{\omega} \min_{\gamma} O_{\text{gan}}(\omega, \gamma) := \mathbb{E}_z[\log \sigma\left(D_\omega(G_\gamma(z))\right)] + \mathbb{E}_x[\log\left(1 - \sigma\left(D_\omega(x)\right)\right)]$$

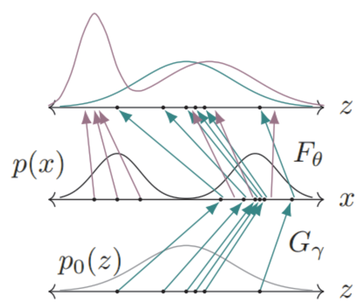
- At the optimum, in the limit of infinite data and arbitrarily powerful networks, we will have $D_\omega = \log q_\gamma(x)/p(x)$, where q_γ is the density that is induced by running the network G_γ on normally distributed input, and hence that $q_\gamma = p$.



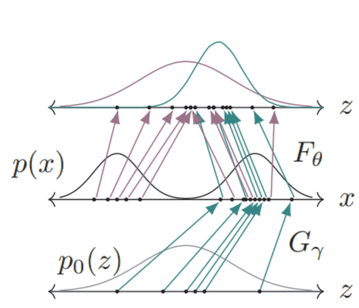
MODE COLLAPSING

- Mode collapsing happens when samples from $q_\gamma(x)$ capture only a few of the modes of $p(x)$.
- An intuition behind why mode collapse occurs is that the only information that the objective function provides about γ is mediated by the discriminator network D_ω.
- For example, if D_ω is a constant, then O_{gan} is constant with respect to γ, and so learning the generator is impossible. When this situation occurs in a localized region of input space, for example, when there is a specific type of image that the generator cannot replicate, this can cause mode collapse.

RECONSTRUCTION CAN HELP BUT...



(a) Suppose F_θ is trained to approximately invert G_γ. Then applying F_θ to true data is likely to produce a non-Gaussian distribution, allowing us to detect mode collapse.



(b) When F_θ is trained to map the data to a Gaussian distribution, then treating $F_\theta \circ G_\gamma$ as an autoencoder provides learning signal to correct $G\gamma$.

Figure 1: Illustration of how a reconstructor network F_θ can help to detect mode collapse in a deep generative network G_γ. The data distribution is $p(x)$ and the Gaussian is $p_0(z)$. See text for details.

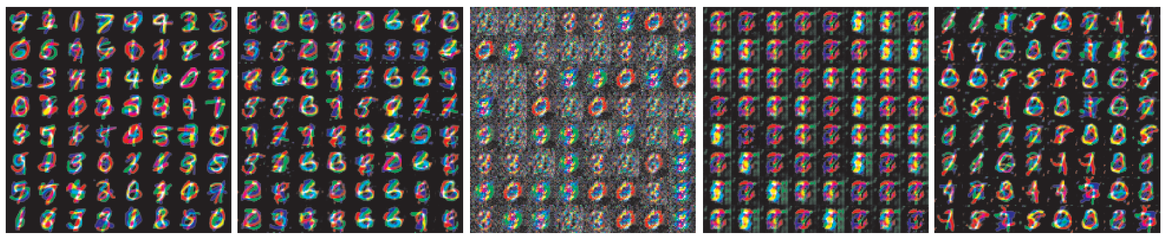
RESULTS

Table 1: Sample quality and degree of mode collapse on mixtures of Gaussians. VEEGAN consistently captures the highest number of modes and produces better samples.

	2D Ring		2D Grid		1200D Synthetic	
	Modes (Max 8)	**% High Quality Samples**	**Modes (Max 25)**	**% High Quality Samples**	**Modes (Max 10)**	**% High Quality Samples**
GAN	1	99.3	3.3	0.5	1.6	2.0
ALI	2.8	0.13	15.8	1.6	3	5.4
Unrolled GAN	7.6	35.6	23.6	16	0	0.0
VEEGAN	**8**	**52.9**	**24.6**	**40**	**5.5**	**28.29**

	Stacked-MNIST		CIFAR-10
	Modes (Max 1000)	KL	IvOM
DCGAN	99	3.4	0.00844 ± 0.002
ALI	16	5.4	**0.0067** ± 0.004
Unrolled GAN	48.7	4.32	0.013 ± 0.0009
VEEGAN	**150**	**2.95**	**0.0068** ± 0.0001

Table 2: Degree of mode collapse, measured by modes captured and the inference via optimization measure (IvOM), and sample quality (as measured by KL) on Stacked-MNIST and CIFAR. VEEGAN captures the most modes and also achieves the highest quality.



(a) True Data (b) DCGAN (c) ALI (d) Unrolled (e) VEEGAN

SOLUTION: VEEGAN TRAINING

The main idea of VEEGAN is to introduce a second network F_θ that we call the **Reconstructor**, which is **trained on the true data distribution** and **learns the reverse feature mapping** from data items x to representations z.

- We start by establishing that, $-\int p_0(z) \log p_\theta(z) \leq O(\gamma, \theta)$ where,

$$O(\gamma, \theta) = KL[q_\gamma(x|z)p_0(z)||p_\theta(z|x)p(x)] - \mathbb{E}[\log p_0(z)] + \mathbb{E}[d(z, F_\theta(x))].$$

  Here all $\mathbb{E}$ are taken with respect to $p_0(z)q_\gamma(x \mid z)$.
- The function d denotes a loss function in representation space $\mathbb{R}^K$, such as l_2 loss. The third term is then an autoencoder in representation space.
- In this case, we cannot optimize O directly, because the KL divergence depends on a density ratio which is unknown. We estimate this ratio using a discriminator network,

$$D_\omega(z, x) = \log \frac{q_\gamma(x \mid z)p_0(z)}{p_\theta(z \mid x)p(x)}.$$

- This allows us to estimate O as,

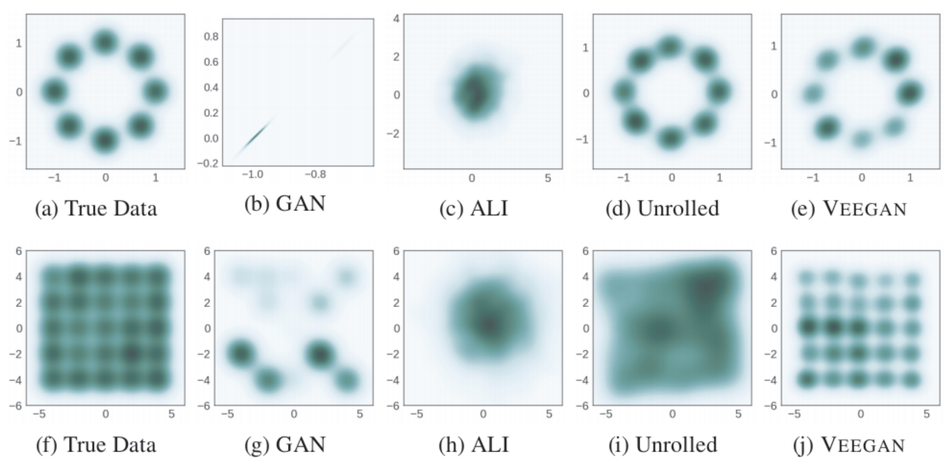
$$\hat{O}(\omega, \gamma, \theta) = \frac{1}{N}\sum_{i=1}^N D_\omega(z^i, x_g^i) + \frac{1}{N}\sum_{i=1}^N d(z^i, x_g^i).$$

- We train the discriminator network using,

$$O_{\text{LR}}(\omega, \gamma, \theta) = -\mathbb{E}_\gamma[\log\left(\sigma\left(D_\omega(z, x)\right)\right)] - \mathbb{E}_\theta[\log\left(1 - \sigma\left(D_\omega(z, x)\right)\right)].$$

- Training the **Reconstructor** on the true data within the KL term and then forcing it to reconstruct the generator output with the penalty function d, **makes the generator and reconstructor approximate inverses of each other**. Hence resolving mode collapse.

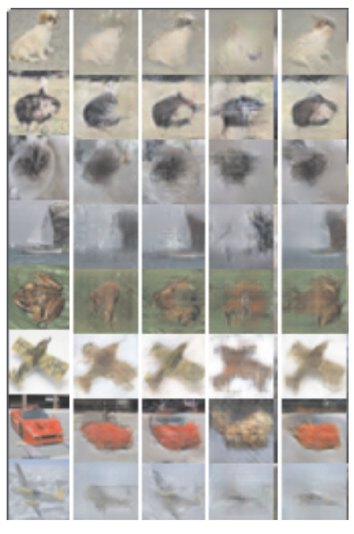
Figure 2: Density plots of the true data and generator distributions from different GAN methods trained on mixtures of Gaussians arranged in a ring (top) or a grid (bottom).



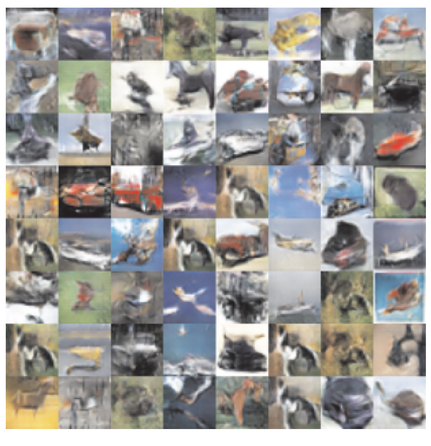
(a) True Data (b) GAN (c) ALI (d) Unrolled (e) VEEGAN

(f) True Data (g) GAN (h) ALI (i) Unrolled (j) VEEGAN

Figure 3: Sample images from GANs trained on CIFAR-10. Best viewed magnified on screen.



(a) Generated samples nearest to real images from CIFAR-10. In each of the two panels, the first column are real images, followed by the nearest images from DCGAN, ALI, Unrolled GAN, and VEEGAN respectively.



(b) Random samples from generator of VEEGAN trained on CIFAR-10.