VEEGAN

Xiaojing Hu
Department of Computer Science
University at Buffalo
xhu7@buffalo.edu

Yujiang Zhao
Department of Computer Science
University at Buffalo
yujiangz@buffalo.edu

Haolan Wang
Department of Computer Science
University at Buffalo
haolanwa@buffalo.edu

Abstract

This report implemented and studied the newly proposed VEEGAN, which is an enhanced GAN network resolving mode collapse. We introduce a reconstructor in VEEGAN and require it to both be an inverse of the generator and map the true data to the normal distribution. Thus GAN can learn from those signals and reduce mode collapse.

1. Introduction

As we all know, mode collapse happens during the training of GAN, which means the generative network can only generate several or one modes of the true data instead of all. This means that generative networks do not learn the overall distribution of the true data. In VEEGAN, a reconstructor is introduced to provide extra learning signal to the generative network to make its output have closer distribution to the true data. Though the reconstructor has been proposed before in other paper, in VEEGAN, it is required to both be and inverse of the generator and map the true data back to normal distribution. In this way, the generative network will gradually learn more modes. And our experiment also showed that VEEGAN is doing well in practice compared to other variants of GAN.

2. Background

2.1 GAN

Before explaining VEEGAN, we briefly introduce generative adversarial networks(GANs) first. Figure 1[1] shows the adversarial principle of GANs. GANs are made of two parts, the generative network and the discriminative network. The generative network generates new data which fits the distribution of the training data, while the discriminative network discriminates between the true data and the generative data. The goal of the generative network is to generate data as true as possible to 'fool' the discriminative network. As a result, GANs could generate high quality samples.

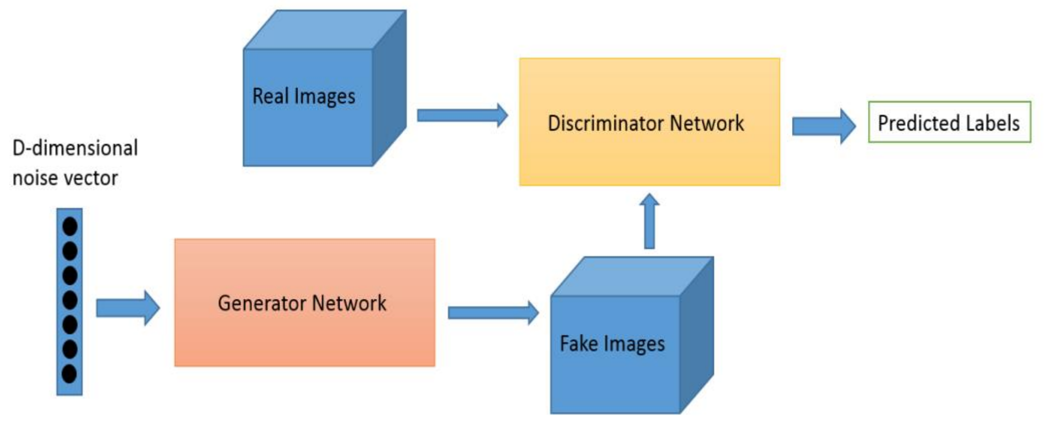


Figure 1: The adversarial principle of GANs

However, GANs fail to capture all models and it limits the quality of output data. For example, as shown in figure 2, we train the GAN with a training set consisting of cat and dog images. But the trained GAN can only generate artificial images of cats. VEEGAN is designed to solve the problem of mode collapse.

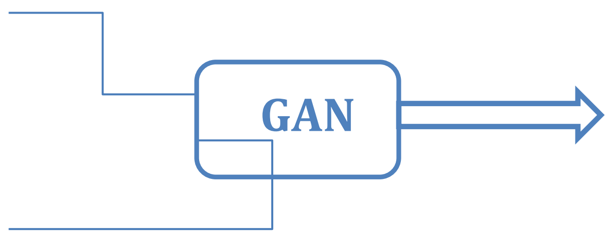


Figure 2: Mode Collapse

2.2 Mode collapse

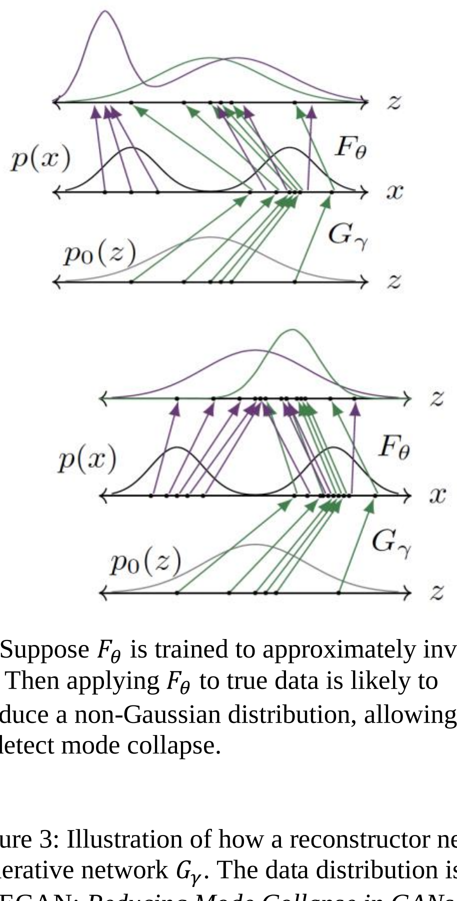
First, we will explain mode collapse. Implicit probability distributions are sampling procedure without a tractable density, so it is hard to estimate. However, recent progresses make it possible to enable implicit model estimation. Here is GAN's object function:

$$max_{\omega}\ min_{\gamma}\ O_{GAN}(\omega,\gamma) = E_z[log\sigma(D_{\omega}(G_{\gamma}(z)))]\ \ + E_x[log\ (1-\sigma(D_{\omega}(x)))\ ]\#(1)$$

where E_z indicates an expectation over the standard normal z, E_x indicates an expectation over the data distribution $p(x)$, and σ denotes the sigmoid function. At the optimum, in the limit of infinite data and arbitrary powerful networks, we will have $D_{\omega} = log\ \frac{q_{\gamma}(x)}{p(x)}$, (Note that D_{ω} means we label the data as generated, which is different from normal discriminators). Model collapse happens when samples from q_γ (x) capture only a few of the modes of p(x). Because the object function cannot provide direct information about γ. For example, if $D_\omega$is a constant function, the object function will never change when the γ changes and mode collapse happens.

3. Basic method for VEEGAN

To solve this problem, VEEGAN introduces a new network, a reconstructor network to reduce the influence of mode collapse. The generative network maps Gaussian random noise to true data distribution, while the reconstructor network is an inverse of the generative network, mapping true data distribution to Gaussian random noise. Figure 3[2] (a) illustrates the method to detect model collapse and (b) presents the basic idea of VEEGAN. In figure 3a, $p(x)$ is the probability density function(pdf) of true data, a mixture of 2 Gaussians. G_γ is a generative network. F_θ is the inverse of network G_γ. $p_0(z)$ is the pdf of input z, a standard normal distribution. Generator labels G_γ, which are shown as arrows. The purple arrows labelled F_θ show the action of the reconstructor on the true data, whereas the green arrows show the action of the reconstructor on data from the generator. G_γ transfers x into z space, and F_θ reconstructs generated x' into z space. If mode collapse happens, we will find that there are 2 peaks in the reconstructed z space instead of one. For figure 3b, F_θ is learned both, to map the true data distribution $p(x)$ to a Gaussian and to approximately invert the generator network. If mode collapse occurs, then F_θ will not map all G_γ (z) back to the original z and the resulting penalty provides us with a strong learning signal for both γ and θ.



(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   is trained to map the data to a Gaussian distribution, then treating ◦ as an autoencoder provides learning signal to correct .

Figure 3: 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). Image source: VEEGAN: *Reducing Mode Collapse in GANs using Implicit Variational Learning.*

As a result, we need to train the reconstructor maps both true data and generated data to Gaussian distribution. An autoencoder is used to measure if F_θ works. We need to minimize a loss function, like l_2 loss between $z \sim p_0$ and $F_\theta(G_\gamma(z))$. We also add a cross entropy to judge whether F_θ maps the true data distribution to a Gaussian. Combining these two parts, we get the equation:

$$O_{entropy}(\gamma,\theta) = E[||z - F_\theta(G_\gamma(z))||_2^2] + H(Z, F_\theta(X)) \#(2)$$

However, it is difficult to minimize this equation and we need to transform it further. Because the cross-entropy function contains an intractable integration, which cannot be computed. We thus introduce a variational distribution $q_\gamma(x|z)$ and by Jensen's inequality, we get equation 3. We here represent the distribution $q_\gamma(z)$ implicitly as a deep generative model, instead of the parametric assumptions are mainly built on q_γ. The variational distribution $q_\gamma(z)$ plays exactly the same role as the generator in a GAN, and for that reason, we will parameterize $q_\gamma(x|z)$ as the output of a stochastic neural network $G_\gamma(z)$.

$$-log\ p_\theta(z) = -log \int p_\theta(x)p(x)\frac{q_\gamma(z)}{q_\gamma(z)}dx \leq -\int q_\gamma(z) log\ \frac{p_\theta(x)p(x)}{q_\gamma(z)}dx \ \ \#(3)$$

It is too easy for a discriminator to distinguish between $p(x)$ and $q_\gamma(x|z)$ because the latter one is much more peaked as a conditional distribution. Instead, we write it as;

$$-\int p_0(z) log\ p_\theta(z) \ \leq\ KL[q_\gamma(z)p_0(z)||p_\theta(x)p(x)] - E_{z\sim p_0(z)}[log\ p_0(z)] \qquad (4)$$

After those steps, we get the final objective function:

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

Instead of minimizing the intractable$O_{entropy}(\gamma,\theta)$, we aim to minimize the upper bound O with respect to γ and θ. Indeed, if the networks F_θ and G_γ are sufficiently powerful, then if we succeed in globally minimizing O, we can guarantee that q_γ recovers the true data distribution.

2.3 Learning with Implicit Probability Distributions

Then we demonstrate how to approximate O when we have implicit representations for q_γ and p_θ rather than explicit densities. Because KL divergence depends on an unknown ratio, we are unable to compute O directly. We estimate this ratio using a discriminator network $D_\omega(z,x)$ which we will train to encourage:

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

This will allow us to estimate O as:

$$\hat{O}(\omega,\gamma,\theta) = \frac{1}{N}\sum_{i=1}^{N} D_{\omega}(z^{i}, x_{g}^{i}) + \frac{1}{N}\sum_{i=1}^{N} d(z^{i}, x_{g}^{i})\#(5)$$

where $(z^{i}, x_{g}^{i}) \sim p_{0}(z)q_{\gamma}(x|z)$.

We will train D_{ω} to distinguish samples from the joint distribution $q_{\gamma}(z)p_{0}(z)$ from $q_{\theta}(x)p(x)$. The objective function for this is:

$$O_{LR}(\omega,\gamma,\theta) = -E_{\gamma}[log\ (\sigma(D_{\omega}(z,x)))\ ] - E_{\theta}[log\ (1-\sigma(D_{\omega}(z,x)))\ ]\#(6)$$

Our goal is to minimize the object function and we want to label data coming from G_{γ} as true and data coming from F_{θ} as false.

3 Experiments

After the implementation of the VEEGAN model, we did some experiments on it to verify the improvement brought by the new model.

And in the experiments, we are going to use three different datasets to verify the results. First one is synthetic datasets, and the second one is stacked MNIST dataset, and the Third one is Cifar10 dataset. And besides VEEGAN model, we will also apply these datasets onto two other models, which are DCGAN and ALI_DCGAN model for comparing.

But before the experiments, we need to notice that due to that quantitative evaluation of GANs is problematic because implicit distributions do not have a tractable likelihood term to quantify generative accuracy. And also, quantifying mode collapsing is also not straightforward, except in the case of synthetic data with known modes. Hence, we will apply KL divergency and IvOM metrics to evaluate the improvement.

3.1 Synthetic Dataset

3.1.1 Dataset

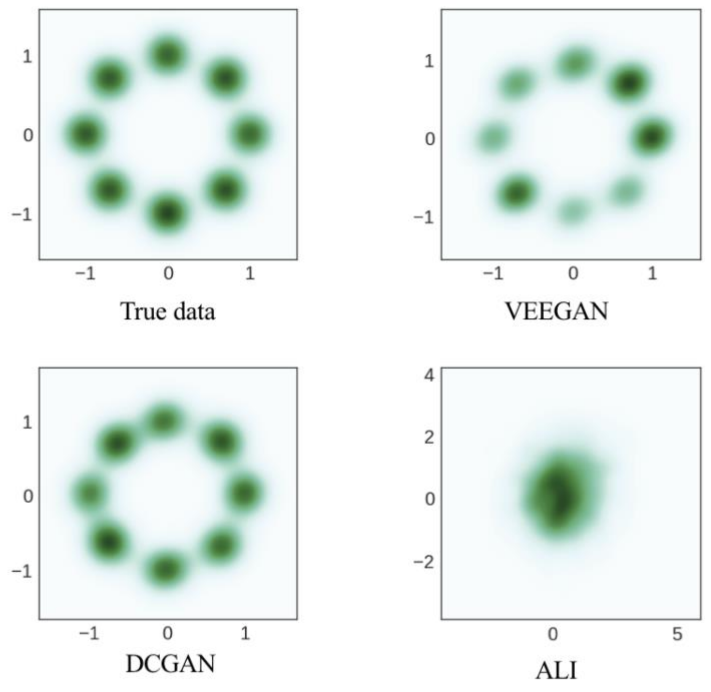
In this part, we used 2Dring For Synthetic Dataset. 2D ring data are made up of 8 2D gaussian distributions arranged in a ring. In this dataset, mode collapse can be accurately measured.

3.1.2 Method

In order to measure the quality of modes captured, we sample points and count a sample as high-quality sample if it is within three standard deviation of the nearest mode.

3.1.3 Result

	2D Ring	
	Modes (Max 8)	%High Quality Samples
DCGAN	8	37.5
ALI_DCGAN	3	30
VEEGAN	8	50



We could notice that, comparing the three methods, both DCGAN and VEEGAN could capture all the 8 modes which form the ring in 2D ring datasets, while ALI_DCGAN could only capture 3 out of 8. And while both VEEGAN and DCGAN capturing the same number of modes, VEEGAN still overcome DCGAN with higher quality of modes captured. Image source: *VEEGAN: Reducing Mode Collapse in GANs using implicit Variational Learning.*

3.2 Stacked MNIST

3.2.1 Dataset

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

And the stacked mnist dataset is synthesized by stacking three randomly sampled MNIST digits along the color channel resulting in a 28x28x3 image, which is designed to increase the number of discrete modes into 1000.

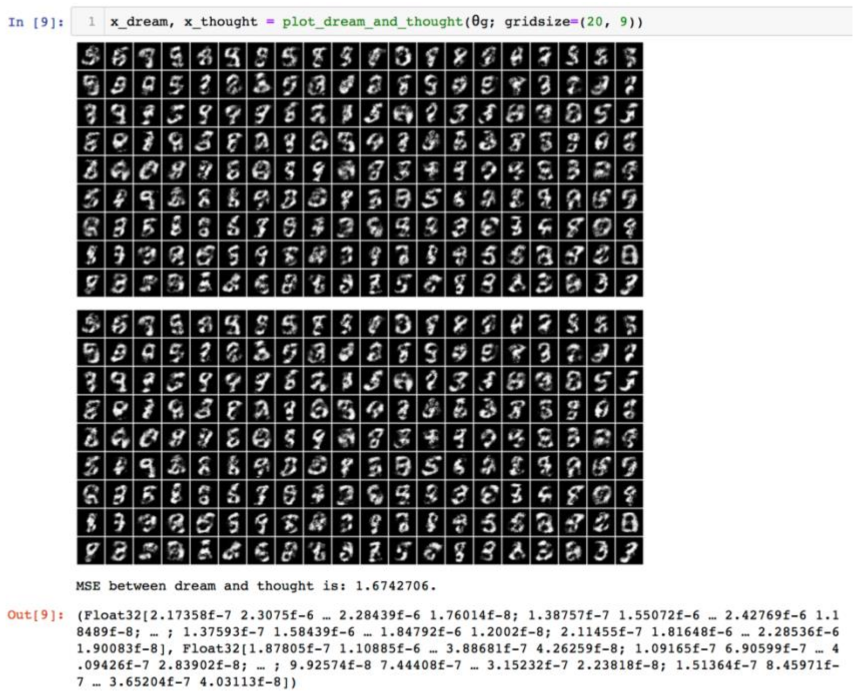
3.2.2 Method

As the true location of modes in this dataset are unknown, the number of modes is estimated using a trained classifier, which is a third-party discriminator between real data and generated data from model. The output of this discriminator can be viewed as estimator for the quantity of modes.

Besides this, we use KL divergency between generator distribution and data distribution to evaluate the quality of modes, models with lower KL divergency can be thought to be with better quality on the captured modes.

3.2.3 Results

Stacked_MNIST		
	Modes (Max 1000)	KL
ALI_DCGAN	20	4.97
DCGAN	108	3.58
VEEGAN	150	2.95



```
In [9]:  x_dream, x_thought = plot_dream_and_thought(θg; gridsize=(20, 9))
```

```
MSE between dream and thought is: 1.6742706.
```

```
Out[9]: (Float32[2.17358f-7 2.3075f-6 … 2.28439f-6 1.76014f-8; 1.38757f-7 1.55072f-6 … 2.42769f-6 1.1
        8489f-8; … ; 1.37593f-7 1.58439f-6 … 1.84792f-6 1.2002f-8; 2.11455f-7 1.81648f-6 … 2.28536f-6
        1.90083f-8], Float32[1.87805f-7 1.10885f-6 … 3.88681f-7 4.26259f-8; 1.09165f-7 6.90599f-7 … 4
        .09426f-7 2.83902f-8; … ; 9.92574f-8 7.44408f-7 … 3.15232f-7 2.23818f-8; 1.51364f-7 8.45971f-
        7 … 3.65204f-7 4.03113f-8])
```

According to the result, we could see that the number of captured modes for both DCGAN and VEEGAN are over 100, which is way more than that of ALI_DCGAN. But VEEGAN still performs somehow better than DCGAN with lower KL divergency.

We could notice that the result of our implementation above is not that good to tell the improvement brought by the model VEEGAN. The images below shows what the author of [5] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann V EEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning achieve with VEEGAN, and compare with other three GAN models.

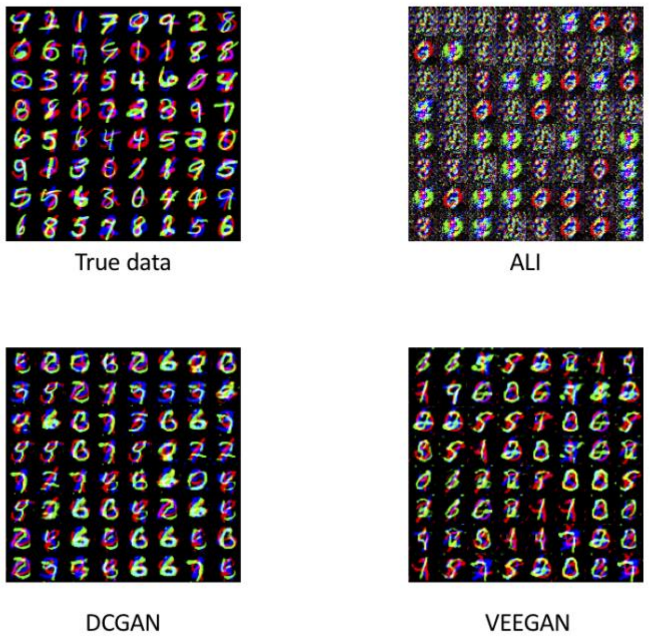


Image referred from [5] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann V EEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning.

3.3 CIFAR10

3.3.1 Dataset

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

3.3.2 Method

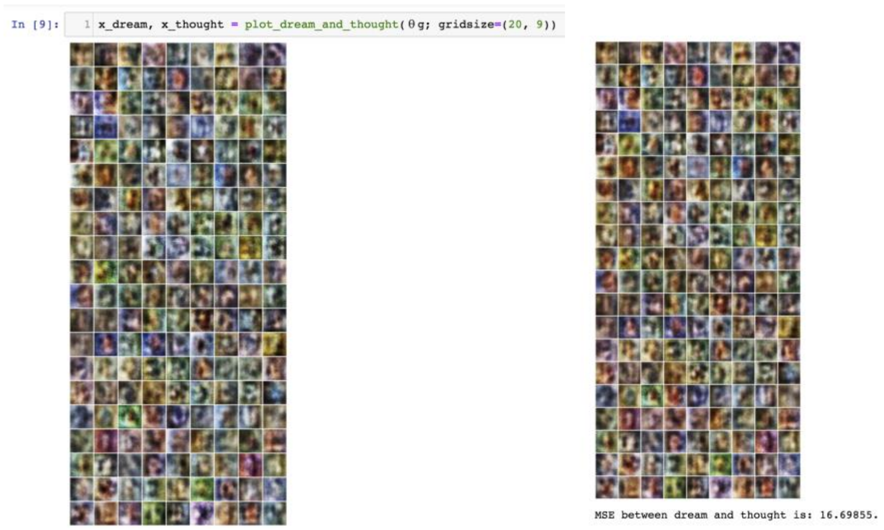
Owing to that Cifar10's much greater diversity, compared with mnist. It is inappropriate for us to assume that each labelled class is corresponding to a single mode of the data distribution.

And what we apply a metrics introduced by [3], which we will call it the inference via optimization metric(IvOM). The idea behind this metric is to Compare real images from the test set to the nearest generated image; if the generator suffers from mode collapse, then there will be some images for which this distance is large. For example, we sample a real image x from the test set, and find the closest image that the GAN is capable of generating, i.e. optimizing the l_2 loss between x and generated image G γ (z) with respect to z. If a method consistently attains low MSE, then it can be assumed to be capturing more modes than the ones which attain a higher MSE.

3.3.3 Result

CIFAR-10 (IvOM)	
ALI_DCGAN	0.04546
DCGAN	0.04805
VEEGAN	0.04613

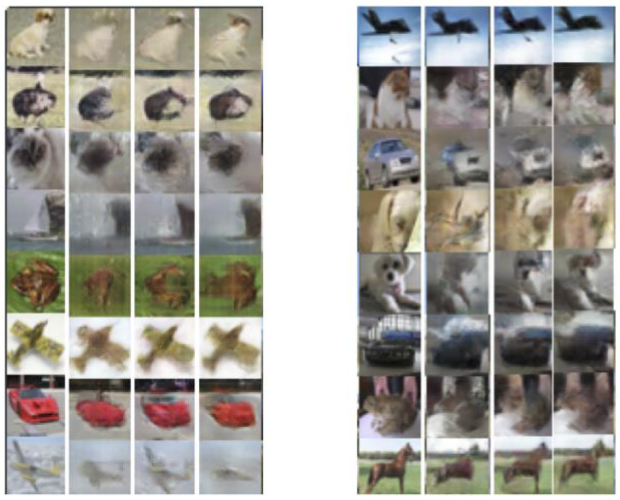
In this part of the experiment, we apply IvOM metric as the measurement for the result of the three models in the comparing groups. But it is not that obvious as the first two experiments, the differences among the output of IvOM are not that large. We could just say that by observing the result of IvOM, VEEGAN model and ALI_DCGAN model generate slightly better result than the DCGAN model.



```
In [9]:  x_dream, x_thought = plot_dream_and_thought(θg; gridsize=(20, 9))
```

```
MSE between dream and thought is: 16.69855.
```

The images above are the result of our implementation of VEEGAN applied on cifar-10 dataset.



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 and VEEGAN respectively (Image referred from [5] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann V EEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning.)

4. Our own study on VEEGAN

In VEEGAN, the objective function is

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

where $(z^{i},\ x_{g}^{i}) \sim p_{0}(z)q_{\gamma}(x|z)$.

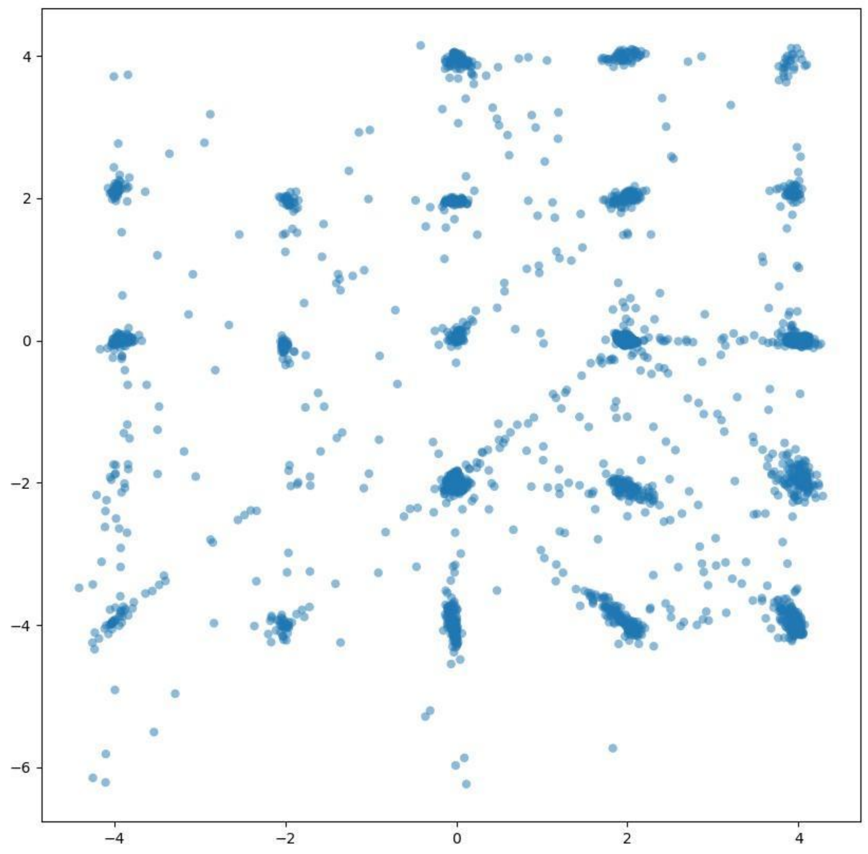
We can see that the first term and second term have the same weight of one. The author didn't mention the influence of the weight and theoretically weights seem do not matter since we can always converge to the optimum of the both terms with infinite training data. But in fact, when we are calculating the KL divergence and the l2 loss, we find they are not in the same magnitude.

So, we are wondering what will happen using different weight.

For simplicity and intuition, we use synthetic dataset of mixture Gaussians.

So here is the original experiment the author provided in the paper. The true data are a mixture of 25 2-d gaussians arranged as a grid. And the network is trained 300 times. And instead of l2 loss, the author chose reconstruction log-likelihood to measure the autoencoder loss.

Here is the sampled data from the trained VEEGAN:

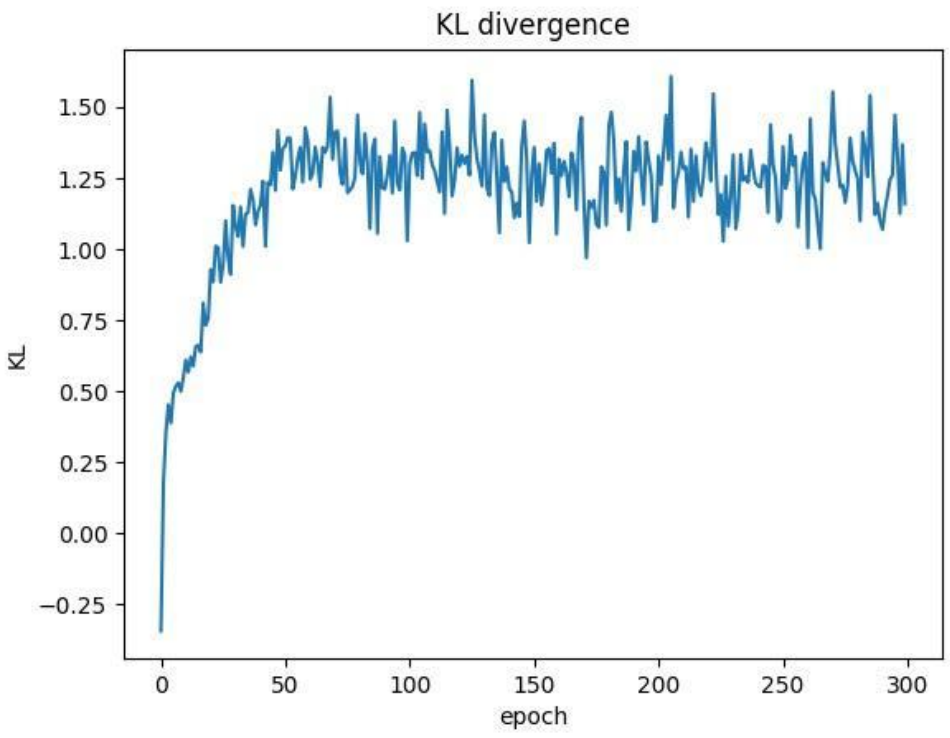


And the result is:

Number of Modes Captured: 23

Number of Points Falling Within 3 std. of the Nearest Mode dict_values([89, 55, 47, 74, 58, 340, 17, 37, 39, 180, 137, 66, 69, 127, 7, 56, 59, 30, 62, 13, 13, 31, 5])

The KL divergence per epoch is:

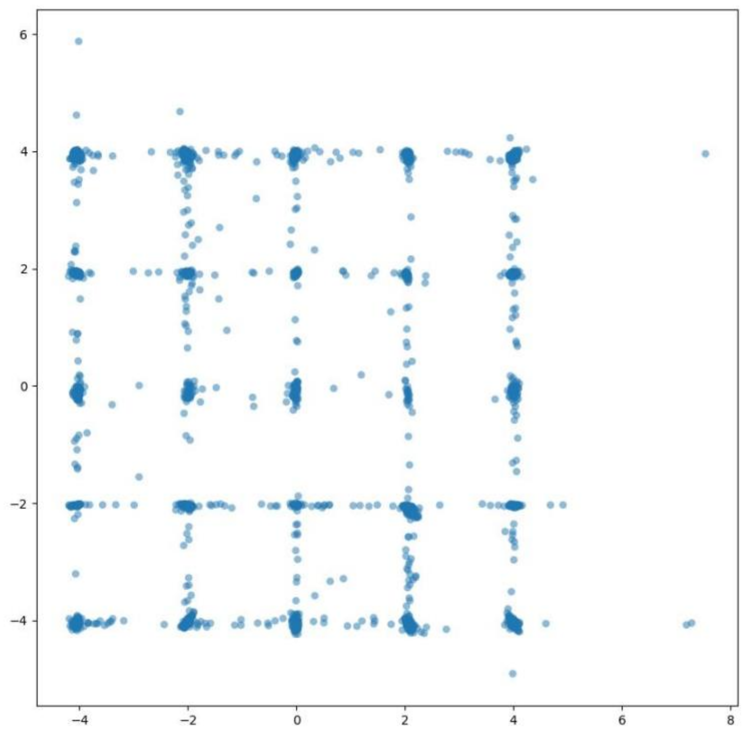


The reconstruction likelihood is rising per epoch.

We can see that the KL divergence does not drop per epoch. But this is not enough to say that the first term does not work because we have another variable ω, which may cause the KL divergence to rise.

The magnitude of the second term is largely influenced by the dimension input z, so we divide the second term by the dimension of z, making the magnitude of the first term and the second term almost the same.

And the data sampled is:



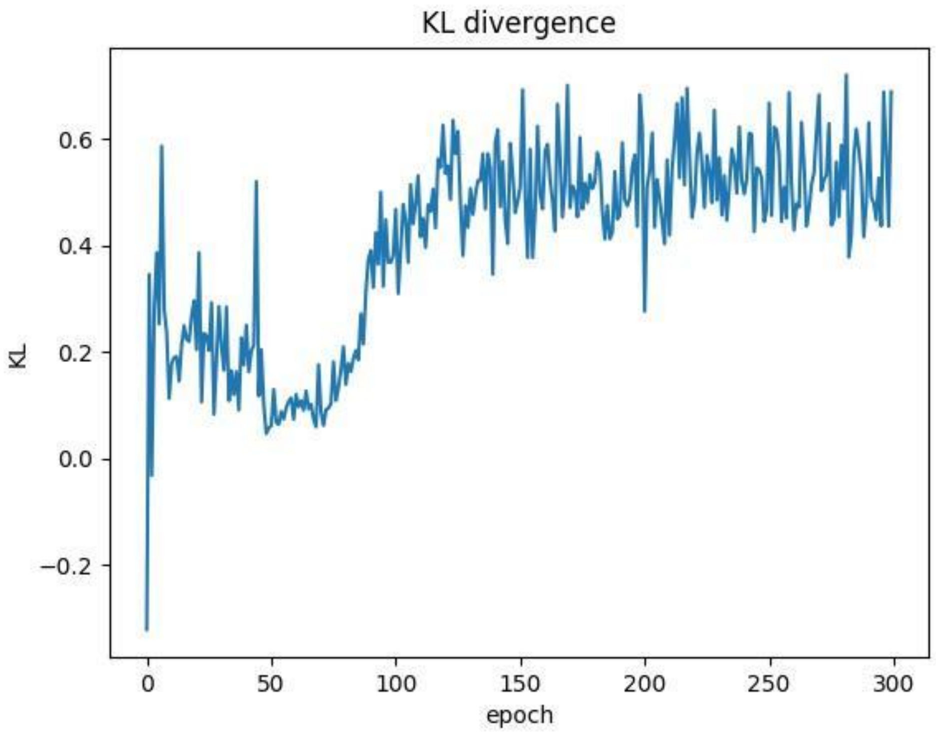
The result is:

Number of Modes Captured: 25

Number of Points Falling Within 3 std. of the Nearest Mode dict_values([112, 78, 95, 142, 137, 70, 111, 76, 125, 103, 55, 104, 78, 46, 26, 111, 116, 122, 68, 56, 34, 50, 75, 74, 9])

The KL divergence is:

Which

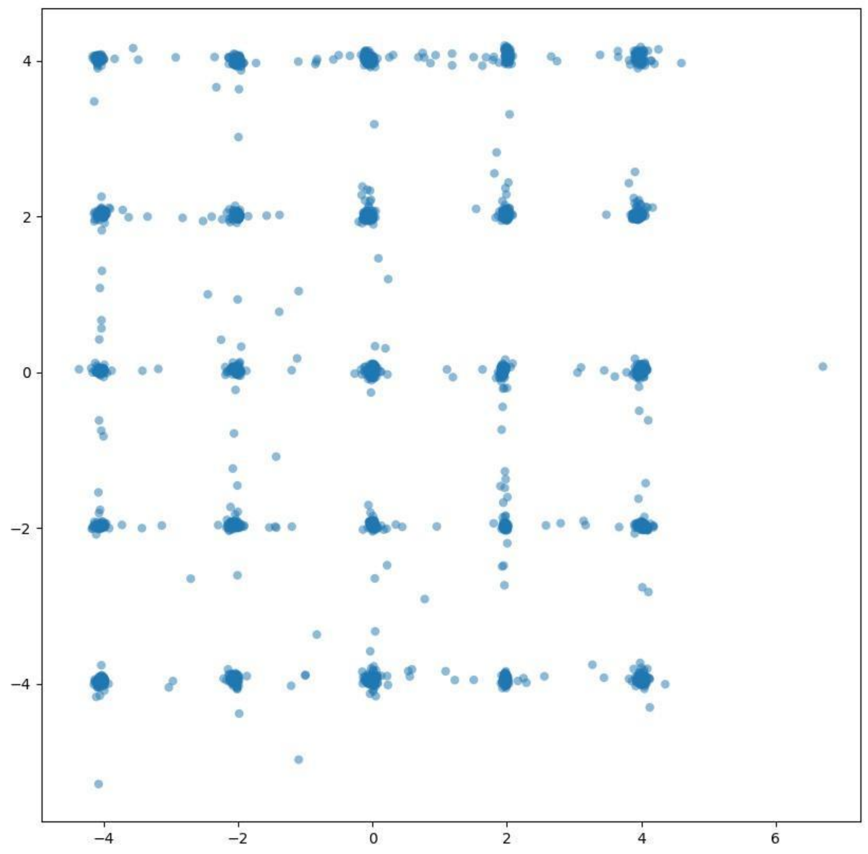


which is much smaller than the model without weights.

We can see that the result is much better since we captured more mode and for each mode we catch almost equivalent number of points. Since the gaussians we mixed are equally distributed, we think the latter result is more likely to be the real distribution of the true data.

And we can even get better result when we drop the likelihood dimension to 40 and normalize it (adding a weight).

So below is the sample data from a network with input of 40 dimension and is trained 100 epochs.

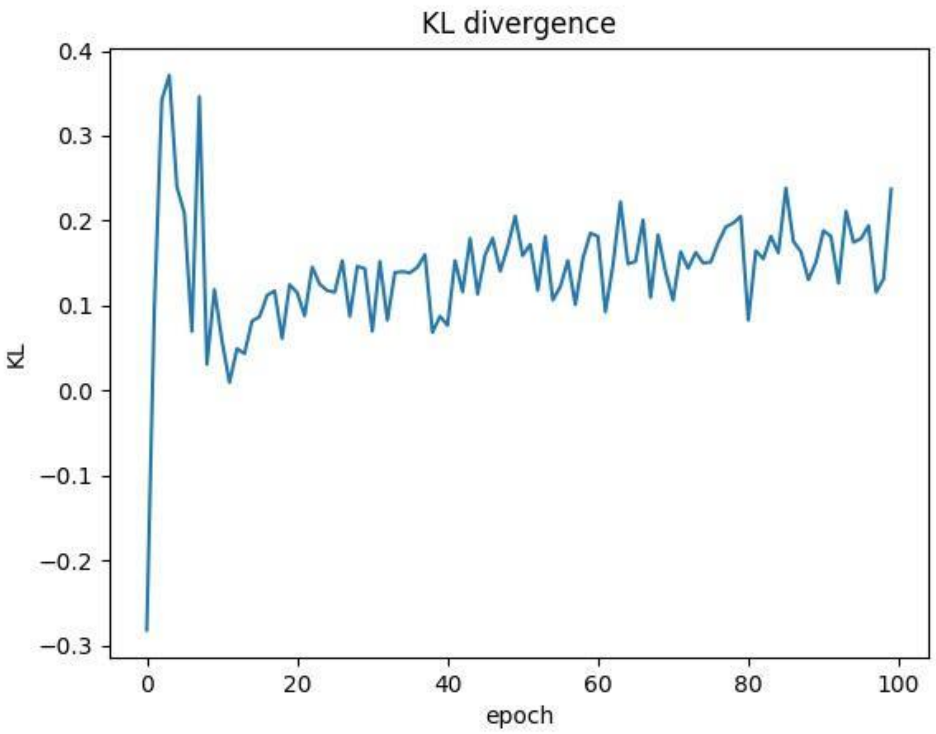


The result is:

Number of Modes Captured: 25

Number of Points Falling Within 3 std. of the Nearest Mode dict_values([89, 60, 81, 84, 99, 115, 95, 80, 128, 129, 100, 84, 103, 65, 66, 94, 107, 80, 83, 75, 82, 80, 94, 68, 50])

And the KL divergence is:

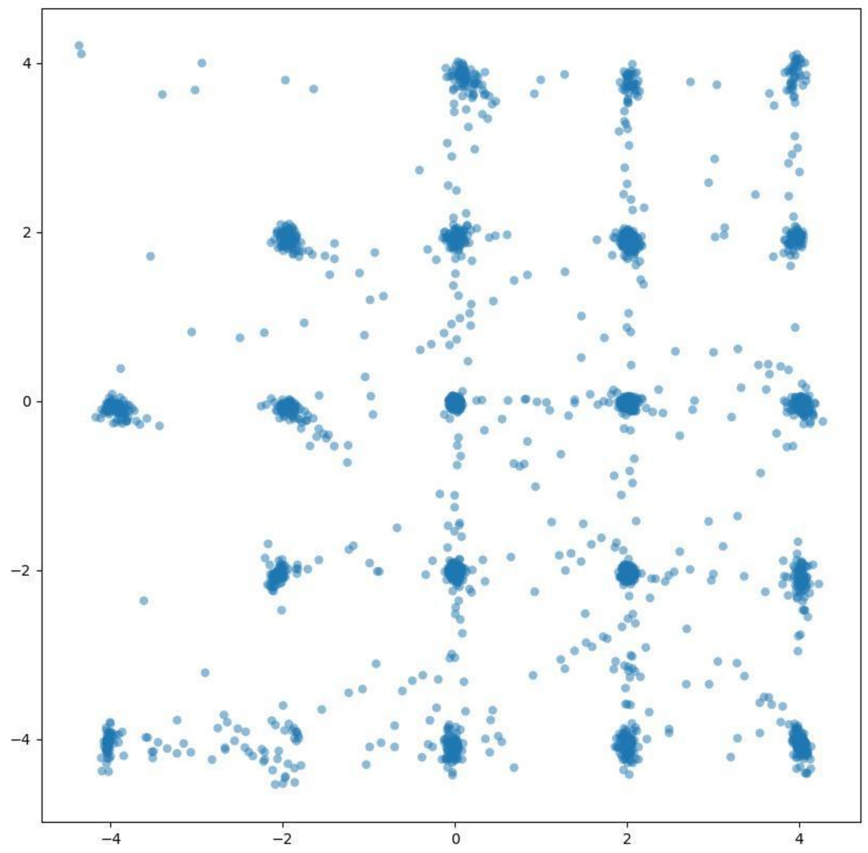


We can see that the KL divergence is dropping and then vibrating, and is near 0 at last.

To make a comparison, we delete the weight in the third experiment(40 dimensions of input, 100 epochs of training and no weight).

So, the data sampled is:

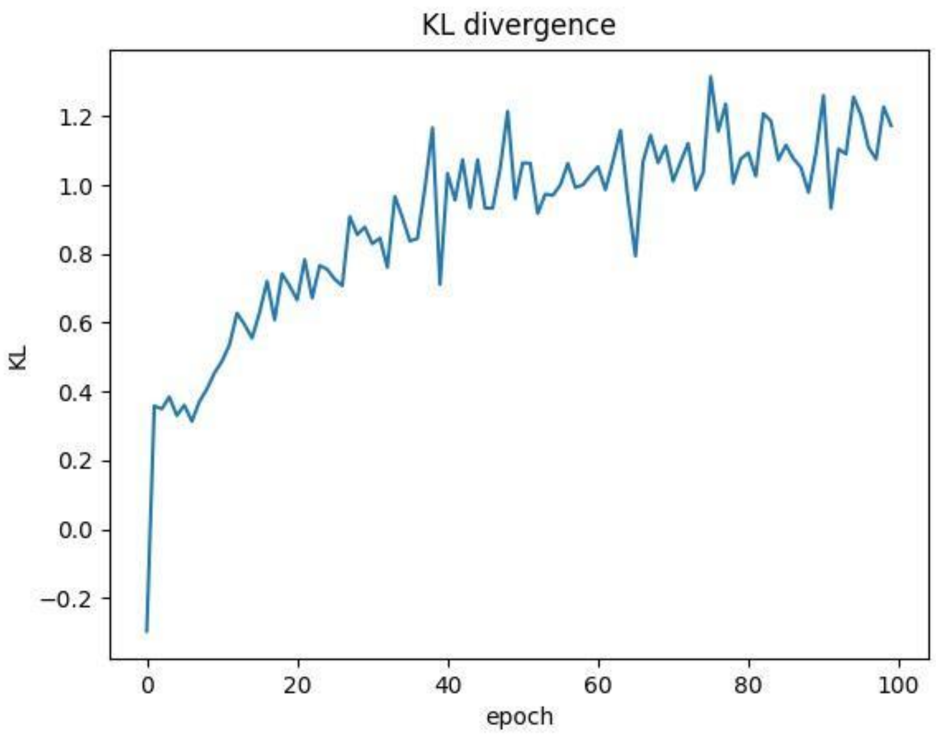
ththe



the result is: Number of Modes Captured: 22

Number of Points Falling Within 3 std. of the Nearest Mode dict_values([96, 104, 118, 79, 219, 186, 69, 77, 40, 227, 77, 75, 58, 88, 146, 45, 84, 14, 38, 6, 63, 3])

And the KL divergence is:



We have an assumption that if we do not add the weight, the second term may dominate the entire objective function. Though VEEGAN works well comparing to ALI, DCGAN and Unrolled GAN, it doesn't mean that it is because it makes the reconstructed z closer to normal distribution. Since VEEGAN has 3 networks with more parameter, it may be just more good at memorizing the true data.

From our experiment, we find that adding a weight really helps the KL term to converge, making mode collapse less likely to happen.

References

[1]: A beginner' guide to GANs. Retrieved from https://deeplearning4j.org/generative-adversarial-network

[2]: Akash Srivastava, Lazar Valkov, Chris Russel, Michael U. Gutmann and Charles Sutton. VEEGAN: Reducing mode collapse in GANs using implicit variational learing. arXiv preprint arXiv:1705,07761, 2017.

[3]: Metz, Luke, Poole, Ben, Pfau, David, and Sohl-Dickstein, Jascha. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163, 2016.

[4]: https://media.nips.cc/nipsbooks/nipspapers/paper_files/nips30/reviews/1879.html

[5] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann V EEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning.