Supplementary Material For
GDPP: Learning Diverse Generations using Determinantal Point Process

A. Experimental Details

To ensure a fair comparison, we use the same architecture and experimental setting for all competing methods.

A.1. Architecture

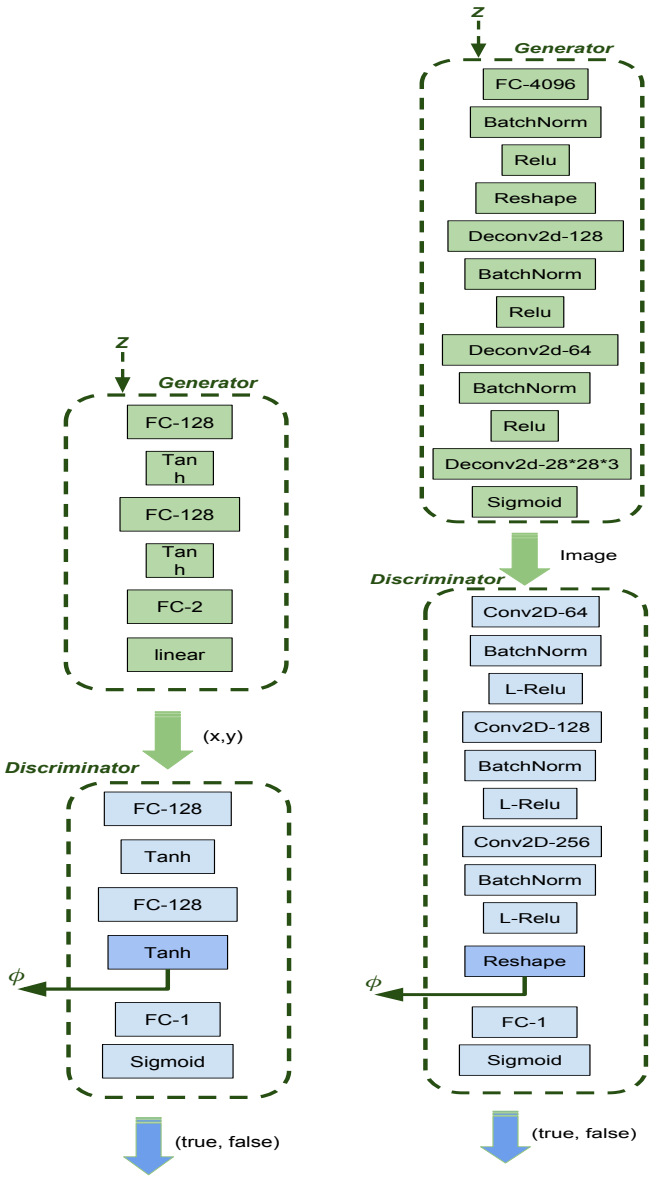


Figure 1. Architecture employed in (a) Synthetic experiments. (b) Stacked-MNIST and CIFAR-10 experiments.

A.2. Hyperparameters

In all of our experiments, we use Adam Optimizer with $\beta_1 = 0.5$ and $\epsilon = 1 \times 10^{-8}$. For the synthetic data experiments, we follow the configurations used by (Srivastava et al., 2017) and (Metz et al., 2017). We use 1×10^{-4} for the discriminator learning rate, and 1×10^{-3} for the generator learning rate. For synthetic data, we use a batch size of 512. For Stacked-MNIST and CIFAR-10 we use a batch size of 64. For CelebA, we use a batch size of 16.

For the Stacked MNIST, CIFAR-10 and CelebA datasets, we use 2×10^{-4} as the learning rate for both of the generator and the discriminator. To relatively stabilize the training of DCGAN, we follow the protocol in (Gulrajani et al., 2017) to train it by applying a learning rate scheduler. The decay is to happen with a ratio of $1/(\#max - iters)$ at every iteration.

In our experiments, we used the official implementations of: WGAN (Arjovsky et al., 2017)[1], WGAN-GP (Gulrajani et al., 2017)[2], DCGAN (Radford et al., 2015)[3], ALI (Dumoulin et al., 2017)[4], VEEGAN (Srivastava et al., 2017)[5] and DeLiGAN (Gurumurthy et al., 2017)[6].

B. Synthetic Data Collections

The first data collection is introduced in (Metz et al., 2017) as a mixture of eight 2D Gaussian distributions arranged in a ring. This distribution is the easiest to mimic since it only requires the generated data to have an equal repulsion from the center of the distribution, even if it is not targeted to the modes. The second and third collections were introduced by (Srivastava et al., 2017). In the second collection, there is a mixture of twenty-five 2D Gaussian distributions arranged in a grid. Unlike the first collection, this one requires a more structured knowledge of the true data modes' locations. The last collection is a mixture of ten 700 dimensional Gaussian distributions embedded in a 1200 dimensional space. This mixture arrangement mimics the higher dimensional manifolds of natural images and demonstrates the effectiveness of each method on manipulating sparse patterns.

C. Additional Experiments

C.1. Invariance to Poor Initialization

Since the weights of the generator are being initialized using a random number generator $\mathcal{N}(0, 1)$, the result of a generative model may be affected by poor initializations. In Figure 2 we show qualitative examples on 2D Grid data, where we use high standard deviation for the random number generator $(i.e., \sigma > 100)$ as an example of poor initializations. Evidently, GDPP-GAN attains the true-data structure manifold even with poor initializations. On the other extreme, WGAN-GP tends to map the input noise to a disperse distribution covering all modes but with low-quality generations.

C.2. (Srivastava et al., 2017) Experimental Setting on Real Data

To show the robustness of our approach to the experimental setting, we further examine it under another more challenging setting. The setting described in (Srivastava et al., 2017) entails an architecture and hyperparameters that produce relatively poor results as compared with the setting of Table 3. For example, In (Srivastava et al., 2017) setting, DCGAN produces 99 modes, while in our experimental setting, DCGAN produces 427 modes on Stacked MNIST dataset. We note that our main results in Table 3 are computed using the same experimental setting suggested by (Gulrajani et al., 2017) and (Metz et al., 2017) on a more realistic architecture. Our method remains to have a clear advantage when compared to the rest of the baselines for both CIFAR-10 and Stacked-MNIST (e.g., covering 90.6% more modes on Stacked-MNIST from 150 to 286 and at a higher quality). We obtain the first four rows from (Srivastava et al., 2017).

[1] https://github.com/martinarjovsky/WassersteinGAN
[2] https://github.com/igul222/improved_wgan_training
[3] https://github.com/carpedm20/DCGAN-tensorflow
[4] https://github.com/IshmaelBelghazi/ALI
[5] https://github.com/akashgit/VEEGAN
[6] https://github.com/val-iisc/deligan

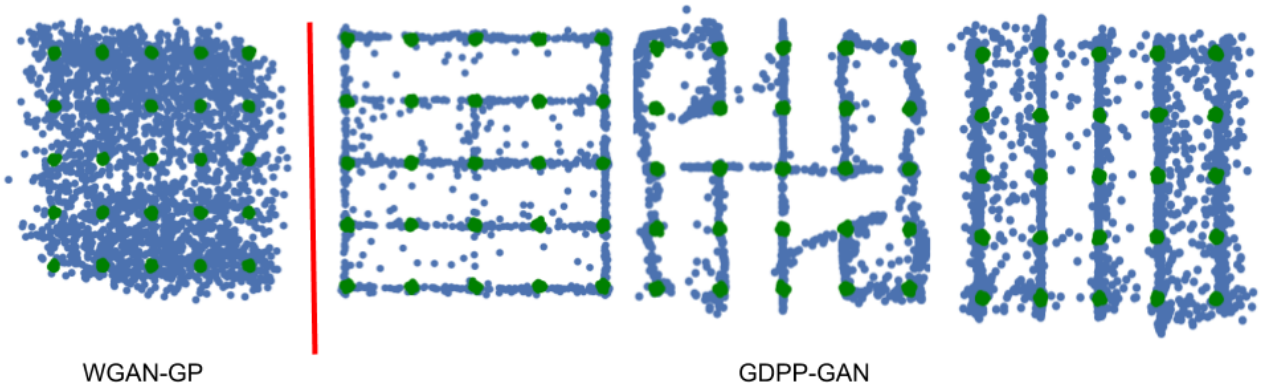


Figure 2. The effect of poor initialization on generations: GDPP-GAN models true manifold structure even with poor initializations, while WGAN-GP maps noise to disperse distribution covering the modes with low quality samples.

	Stacked-MNIST		CIFAR-10
	#Modes (Max 1000)	KL div.	IvO
DCGAN (Radford et al., 2016)	99	3.4	0.00844
ALI (Dumoulin et al., 2017)	16	5.4	0.0067
Unrolled-GAN (Metz et al., 2017)	48.7	4.32	0.013
VEEGAN (Srivastava et al., 2017)	150	2.95	0.0068
GDPP-GAN (Ours)	**286**	**2.12**	**0.0051**

Table 1. Performance on real datasets using the challenging experimental setting of (Srivastava et al., 2017). GDPP-GAN continues to outperform all baselines on both Stacked-MNIST and CIFAR-10 for all metrics.

C.3. Eigendecomposition Running time

Eigendecomposition of an $n \times n$ matrix requires $O(n^3 + n^2 \log^2 n \log b)$ runtime within a relative error bound of 2^{-b} as shown in (Pan & Chen, 1999). In our GDPP loss, we perform two eigendecompositions every training iteration: L_{S_B}, L_{D_B} corresponding to the fake and true DPP kernels respectively. Therefore, the run-time analysis of our loss is $O(n^3)$ at every iteration, where n is the batch size.

Normally the batch size does not exceed 1024 for most training paradigms due to memory constraints. In our experiments, we used 512 for synthetic data and 64 or 16 for real data. Hence, the eigendecomposition does not account for a significant delay in the method.

To further verify this claim, we measured the relative time that eigendecompositions take of each iteration time. We obtained 11.61% for Synthetic data, 9.36% for Stacked-MNIST data and 8.27% for CIFAR-10. Additionally, Table 4 in the original text shows that our method obtains the closest running time to the standard DCGAN, and is faster than the rest of baselines by a large margin (e.g., 5.8$\times$ faster than WGAN-GP). The large margin in running time between GDPP and WGAN-GP (Gulrajani et al., 2017) is attributed to two factors. First, WGAN-GP trains the discriminator several times for every one iteration of the generator, which significantly increase the running time. Second, WGAN-GP calculates the gradients of the discriminator at every iteration, which is a very computationally expensive operation. On the other hand, GDPP does not alter the adversarial learning paradigm and only adds an insignificant computation overhead as discussed.

C.4. Number of statistically-Different bins (NDB)

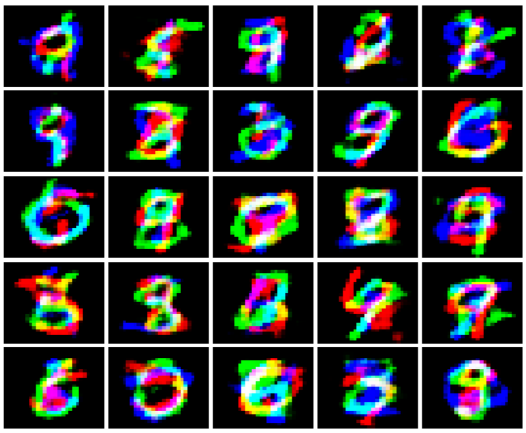
(Richardson & Weiss, 2018) proposed to use a new evaluation metric to assess the severity mode collapse severity in a generative model. They based their metric on a simple observation: In two sets of samples that represent the same distribution, the number of samples that fall into a given bin should be the same up to a sampling noise. In other words, if we clustered the true-data distribution and fake-data distribution to the same number of clusters/bins, then the number of samples from each distribution in every bin should be similar.

We follow (Richardson & Weiss, 2018) to compute this metric on MNIST (LeCun, 1998) dataset, and compare our method with their results in Table 2. We note that we used their open-source implementation of the metric, and we obtained the first three rows from their paper. We use 20,000 samples from our model and the MNIST training data to compute the NDB/K.

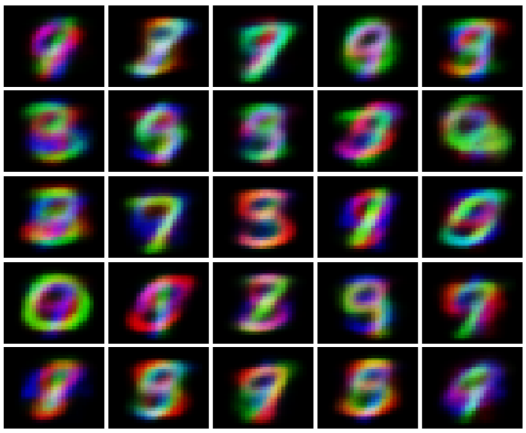
Model	K =100	K =200	K =300
Train	0.06	0.04	0.05
MFA (Richardson & Weiss, 2018)	0.14	**0.13**	0.14
DCGAN (Radford et al., 2016)	0.41	0.38	0.46
WGAN (Arjovsky et al., 2017)	0.16	0.20	0.21
GDPP-GAN	**0.11**	0.15	**0.12**

Table 2. NDB/K - numbers of statistically different bins, with significance level of 0.05, divided by the number of bins K.

D. Additional Qualitative Results



(a) GDPP-GAN after 15K iterations.



(b) GDPP-VAE after 45K iterations.

Figure 3. Random samples generated on Stacked-MNIST. GDPP-GAN converges faster than GDPP-VAE and generates sharper samples.

(a) Generations by GDPP-GAN after **100K** iterations.

(b) Generations by WGAN-GP after **200K** iterations.

Figure 4. Random samples generated by GDPP-GAN and WGAN-GP respectively in an unsupervised setting. The generations are qualitatively similar while GDPP-GAN outperforms WGAN-GP quantitatively even though it was trained for half the number of iterations.

Figure 5. Fixed noise qualitative progression for different models. GDPP-GAN starts synthesizing realistic generations the first with more diverse patterns than both DCGAN and WGAN-GP.

(a) random samples using WGAN-GP

(b) random samples when adding GDPP loss

Figure 6. Comparing (Karras et al., 2018) without (a) and with our loss (b) after 200,000 training iterations.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *stat*, 1050:26, 2017.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. Adversarially learned inference. *ICLR*, 2017.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. In *NIPS*. 2017.

Gurumurthy, S., Sarvadevabhatla, R. K., and Babu, R. V. Deligan: Generative adversarial networks for diverse and limited data. In *CVPR*, pp. 4941–4949, 2017.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. *ICLR*, 2018.

LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *ICLR*, 2017.

Pan, V. Y. and Chen, Z. Q. The complexity of the matrix eigenproblem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 507–516. ACM, 1999.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016.

Richardson, E. and Weiss, Y. On gans and gmms. *arXiv preprint arXiv:1805.12462*, 2018.

Srivastava, A., Valkoz, L., Russell, C., Gutmann, M. U., and Sutton, C. Veegan: Reducing mode collapse in GANs using implicit variational learning. In *NIPS*, 2017.