Supplementary Material: The Bures Metric for Generative Adversarial Networks

No Author Given

No Institute Given

1 Organization

In Section 2, some details about our theoretical results are given. Next, additional experiments are described in Section 3. Importantly, the latter section includes

- another training algorithm (Alt-BuresGAN), which is parameter-free and achieves comptetitive results with BuresGAN,
- an ablation study comparing the effect of the Bures metric with the Frobenius norm,
- timings per iteration for all the generatives methods listed in this paper,
- supplementary tables for the experiments described in the paper,
- a more extensive study on Stacked MNIST of the influence of the batch size for a simpler DCGAN architecture with 2 convolutional layers in the discriminator.

Next, illustrative figures including generated images are listed in Section 4. Finally, Section 5 gives details about the architectures used in this paper.

2 Details of the theoretical results

Let A and B be symmetric and positive semi-definite matrices. Let $A^{1/2} = U\,\mathrm{diag}(\sqrt{\lambda})U^\top$ where U and λ are obtained thanks to the eigenvalue decomposition $A = U\,\mathrm{diag}(\lambda)U^\top$. We show here that the Bures distance between A and B is

$$\mathcal{B}\left(A,B\right)^2 = \min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A + B - 2(A^{\frac{1}{2}}BA^{\frac{1}{2}})^{\frac{1}{2}}), \qquad (1)$$

where $O(\ell)$ is the set of $\ell \times \ell$ orthonormal matrices. We can simplify the above expression as follows

$$\min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A) + \mathrm{Tr}(B) - 2\max_{U\in O(\ell)} \mathrm{Tr}(A^{1/2}B^{1/2}U) \qquad (2)$$

since $\mathrm{Tr}(U^\top B^{1/2}A^{1/2}) = \mathrm{Tr}(A^{1/2}B^{1/2}U)$. Let the characteristic function of the set of orthonormal matrices be $f(U) = \chi_{O(\ell)}(U)$ that is, $f(U) = 0$ if $U \in O(\ell)$ and $+\infty$ otherwise.

Lemma 2. *The Fenchel conjugate of* $f(U) = \chi_{O(\ell)}(U)$ *is* $f^\star(M) = \|M\|_\star$*, where the nuclear norm is* $\|M\|_\star = \mathrm{Tr}(\sqrt{M^\top M})$ *and* $U, M \in \mathbb{R}^{\ell \times \ell}$.

Proof. The definition of the Fenchel conjugate with respect to the Frobenius inner product gives

$$f^\star(M) = \sup_{U \in \mathbb{R}^{\ell \times \ell}} \mathrm{Tr}(U^\top M) - f(U) = \max_{U \in O(\ell)} \mathrm{Tr}(U^\top M).$$

Next we decompose M as follows: $M = W \Sigma V^\top$, where $W, V \in O(\ell)$ are orthogonal matrices and Σ is a $\ell \times \ell$ diagonal matrix with non negative diagonal entries, such that $MM^\top = W\Sigma^2 W^\top$ and $M^\top M = V \Sigma^2 V^\top$. Notice that the non zero diagonal entries of Σ are the singular values of M. Then,

$$\max_{U \in O(\ell)} \mathrm{Tr}(U^\top M) = \max_{U \in O(\ell)} \mathrm{Tr}(W \Sigma V^\top U) = \max_{U' \in O(\ell)} \mathrm{Tr}(\Sigma U'),$$

where we renamed $U' = V^\top U W$. Next, we remark that $\mathrm{Tr}(\Sigma U') = \mathrm{Tr}(\Sigma\, \mathrm{diag}(U'))$. Since by construction, Σ is diagonal with non negative entries the maximum is attained at $U' = \mathbb{I}$. Then, the optimal objective is $\mathrm{Tr}(\Sigma) = \mathrm{Tr}(\sqrt{M^\top M})$.

By taking $M = A^{1/2} B^{1/2}$ we obtain equation 1. Notice that the role of A and B can be exchanged in equation 1 since U is orthogonal.

3 Additional Experiments

3.1 Alternating training

In the sequel, we also provide the results obtained thanks to Alt-BuresGAN, described in Algorithm 2, which is parameter-free and yield competitive results with BuresGAN. Nevertheless, the training time of Alt-BuresGAN is larger compared to BuresGAN.

Algorithm 2 Alt-BuresGAN

Sample a real and fake batch
Update G by minimizing $\mathcal{B}(\hat{\boldsymbol{C}}_r, \hat{\boldsymbol{C}}_g)^2$
Update G by minimizing V_G
Update D by maximizing $-V_D$;

3.2 Another matrix norm

In this section, we also compare BuresGAN with a similar construction using another matrix norm. Precisely, we also use the squared Frobenius norm (Frobenius^{2}GAN), i.e., with the following generator loss $V_G + \lambda \|C_d - C_g\|_F^2$, with $\lambda = 1$. We observe that

– Frobenius^{2}GAN also performs well on simple image data sets such as Stacked MNIST,
– Frobenius^{2}GAN has a worse performance with respect to BuresGAN on CIFAR and STL-10 data sets (see Table 5).

This observation emphasizes that the use of the Bures metric is requisite in order to obtain high quality samples for higher dimensional image data sets such as CIFAR and STL-10 and that the simpler variant based on the Frobenius norm cannot achieve similar result.

3.3 Timings

The timings per iteration for the experiments presented in the paper are listed in Table 1. Times are given for all the methods considered, although some methods do not always generate meaningful images for all datasets. The methods are timed for 50 iterations after the first 5 iterations, which is used as a warmup period, following which the average number of iterations per second is computed. The fastest method is the vanilla GAN. BuresGAN has a similar computation cost as GDPP. We observe that (alt-)BuresGAN is significantly faster compared to WGAN-GP. In order to obtain reliable timings, these results were obtained on the same GPU Nvidia Quadro P4000, although, for convenience, the experiments on these image datasets were executed on a machines equipped with different GPUs.

Table 1. Average time per iteration in seconds for the convolutional architecture. Average over 5 runs, with std in parenthesis. The batch size is 256. For Stacked MNIST, we use a discriminator architecture with 3 convolutional layers (see Table 11). For readability, the computation times for the Grid and Ring dataset are multiplied by 10^{-3}.

	Grid	Ring	stacked MNIST	CIFAR-10	CIFAR-100	STL-10
GAN	**1.28**(0.008)	**1.28**(0.016)	**0.54**(0.0005)	**0.65**(0.02)	**0.64**(0.0008)	**6.00**(0.01)
WGAN-GP	9.0(0.04)	8.96(0.04)	2.99(0.004)	3.41(0.009)	3.41(0.006)	36.5(0.03)
UnrolledGAN	3.92(0.008)	3.92(0.028)	1.90(0.002)	2.17(0.003)	2.18(0.004)	21.99(0.06)
MDGAN-v2	3.24(0.008)	3.24(0.032)	1.66(0.002)	1.98(0.002)	1.98(0.002)	18(0.03)
VEEGAN	1.96(0.012)	1.96(0.02)	0.56(0.006)	0.66 (0.006)	0.65(0.004)	6.10(0.03)
GDPP	15.68(1.0)	16.16(0.2)	0.69(0.02)	0.80(0.02)	0.80(0.02)	7.46(0.03)
PacGAN2	1.72(0.016)	1.72(0.024)	0.77(0.006)	0.91(0.007)	0.91(0.007)	8.02(0.008)
Frobenius^{2}GAN	2.32(0.024)	2.28(0.028)	0.98(0.004)	1.09(0.004)	1.09(0.003)	13.6(0.01)
BuresGAN	11.16(0.016)	11.92(0.04)	0.72(0.02)	0.82(0.001)	0.82(0.0008)	7.6(0.03)
Alt-BuresGAN	11.92(0.024)	11.12(0.04)	0.98(0.008)	1.15(0.007)	1.15(0.007)	10.10(0.03)

3.4 Best inception scores achieved with DCGAN architecture

The inception scores for the best trained models are listed in Table 2. For the CIFAR datasets, the largest inception score is significantly better than the mean for UnrolledGAN and VEEGAN. This is the same for GAN and GDPP on the STL-10 dataset, where the methods often converge to bad results. Only the proposed methods are capable of consistently generating high quality images over all datasets.

Table 2. Inception Score for the best trained models on CIFAR-10, CIFAR-100 and STL-10, with a DCGAN architecture (higher is better).

	CIFAR-10	CIFAR-100	STL-10
GAN	5.92	6.33	6.13
WGAN-GP	2.54	2.56	/
UnrolledGAN	4.06	4.14	/
VEEGAN	3.51	3.85	/
GDPP	6.21	6.32	6.05
BuresGAN	6.69	6.67	7.94
Alt-BuresGAN	6.40	6.48	7.88

3.5 Influence of the number of convolutional layers for DCGAN architecture

Results for an architecture with 3 and 4 convolutional layers described in Table 11 and Table 6 are listed in the paper. An overview of similar experiments in other papers is given in Table 9 hereafter. A more complete study is performed in this section. We provide here results with a DCGAN architecture for a simpler architecture with 2 convolutional layers for the discriminator (see, Table 4 for the results and Table 11 for the architecture).

Discussion. Interestingly, for most models, an improvement is observed in the quality of the images – KL divergence – and in terms of mode collapse – number of modes attained – as the size of the batch increases. For the same batch size, architecture and iterations, the image quality is improved by BuresGAN, which is robust with respect to batch size and architecture choice. The other methods show a higher variability over the different experiments. WGAN-GP has the best single run performance with a discriminator with 3 convolutional layers and has on average a superior performance when using a discriminator with 2 convolutional layers (see Table 4) but sometimes fails to converge when increasing the number of discriminator layers by 1 along with increasing the batch size.

MDGANv2, VEEGAN, GDPP and WGAN-GP often have an excellent single run performance. However, when increasing the number of discriminator layers, the training of these models has a tendency to collapse more often as indicated by the large standard deviation. Vanilla GAN is one of the best performing models in the variant with 3 layers. This indicates that, for certain datasets, careful architecture tuning can be more important than complicated training schemes. A lesson from Table 4 is that BuresGAN's mode coverage does not vary much if the batch size increases, although the KL divergence seems to be slightly improved.

Ablation study. Results obtained with the Frobenius norm rather than the Bures metric are also included in the tables of this section.

Table 3. Comparison for Frobenius norm. Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations.

	Grid (25 modes)		Ring (8 modes)	
	Nb modes	% in 3σ	Nb modes	% in 3σ
Frobenius^{2}GAN	25(0)	73(1)	8(0)	59(16)
BuresGAN	25(0)	82(1)	8(0)	82(4)
Alt-BuresGAN	**25**(0)	**84**(1)	**8**(0)	**84**(6)

Table 4. KL-divergence between the generated distribution and true distribution (Quality, lower is better). The number of counted modes indicates the amount of mode collapse (higher is better). 25k iterations and average and std over 10 runs. Same architecture as in Table 2 in the paper with a discriminator with 2 convolutional layers. The architecture details are in Table 11.

		Nb modes(↑)			KL div.(↓)		
	Batch size	64	128	256	64	128	256
2 conv. layers	GAN	970.5(5.8)	972.7(6.4)	979(3.5)	0.47(0.04)	0.44(0.02)	0.41(0.03)
	WGAN-GP	**996.7**(1.6)	**997.5**(0.9)	**998.1**(1.5)	**0.25**(0.01)	**0.22**(0.01)	**0.21**(0.05)
	MDGAN-v1	115.9(197)	260.9(267)	134.3(157)	5.5(1.4)	4.9(1.7)	5.8(0.9)
	MDGAN-v2	698.1(456)	898.4(299)	599.2(488)	2.2(3.0)	0.86(1.9)	2.8(3.2)
	UnrolledGAN	953.5(11)	971.4(4.8)	966.2(17.3)	0.71(0.06)	0.60(0.04)	0.58(0.10)
	VEEGAN	876.7(290)	688.5(443)	775.9(386)	0.92(1.6)	1.9(2.4)	1.54(2.2)
	GDPP	974.4(3.3)	978.2(7.6)	980.5(6.0)	0.45(0.02)	0.43(0.03)	0.41(0.03)
	PacGAN2	969.8(6.9)	971.1(3.6)	977.9(4.3)	0.54(0.04)	0.51(0.01)	0.48(0.02)
	Frobenius^{2}GAN	975.8(8.0)	981.0(4.1)	981.8(3.2)	0.34(0.03)	0.28(0.02)	0.25(0.01)
	BuresGAN	973.2(1.3)	979.9(4.0)	981.1(4.9)	0.36(0.02)	0.30(0.02)	0.25(0.01)
	Alt-BuresGAN	975.4(6.8)	978.2(5.4)	980.2(3.0)	0.37(0.02)	0.30(0.01)	0.28(0.01)

Table 5. Ablation study on real image data sets for a convolutional architecture. Frobenius norm is also considered in the place of the Bures metric. Generation quality on CIFAR-10, CIFAR-100 and STL-10 with a DCGAN architecture. Average(std) over 10 runs. 100k iterations for each. For improving readability, SWD score was multiplied by 100.

	CIFAR-10			CIFAR-100			STL-10		
	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)
Frobenius^{2}GAN	5.40(0.16)	57.7(4.8)	**1.3**(0.3)	5.52(0.16)	60.1(4.4)	**1.6**(0.5)	6.26(0.15)	132.18(3.6)	**1.8**(0.4)
BuresGAN	**6.34**(0.17)	**43.7**(0.9)	2.1(0.6)	**6.5**(0.1)	**47.2**(1.2)	2.1(1.0)	**7.6**(0.3)	**109**(7)	2.3(0.3)
Alt-BuresGAN	6.23(0.07)	45.4(2.8)	1.7(0.9)	6.4(0.1)	49.4(3.4)	1.8(0.6)	7.5(0.3)	110(4)	2.8(0.4)

Table 6. Ablation study for Stacked MNIST experiment for an architecture with 4 convolutional layers (see Tbale 10). All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2e−4 for Adam and a normal latent distribution. The evaluation is over 10k samples and we report an average (std) over 10 runs.

	Nb modes(↑)	KL div.(↓)
Frobenius^{2}GAN	984.7(6.4)	0.38(0.04)
BuresGAN	989.9(4.7)	0.38(0.06)
Alt-BuresGAN	990.0(4.9)	0.33(0.04)

4 Additional Figures

4.1 Training visualization at different steps for the artificial data sets

In Figure 1 and Figure 2, we provide snapshots of generated samples for the methods studied in the case of Ring and Grid. In both cases, BuresGAN seems to generate samples in such a way that the symmetry of the training data is preserved during training. This is also true for WGAN and MDGAN, while some other GANs seem to recover the symmetry of the problem at the end of the training phase.

4.2 Generated images of the trained GANs

Next, in Figure 3, Figure 4, Figure 5 and Figure 6, we list samples of generated images for the models trained with a DCGAN architecture for Stacked MNIST, CIFAR-10, CIFAR-100 and STL-10 respectively.

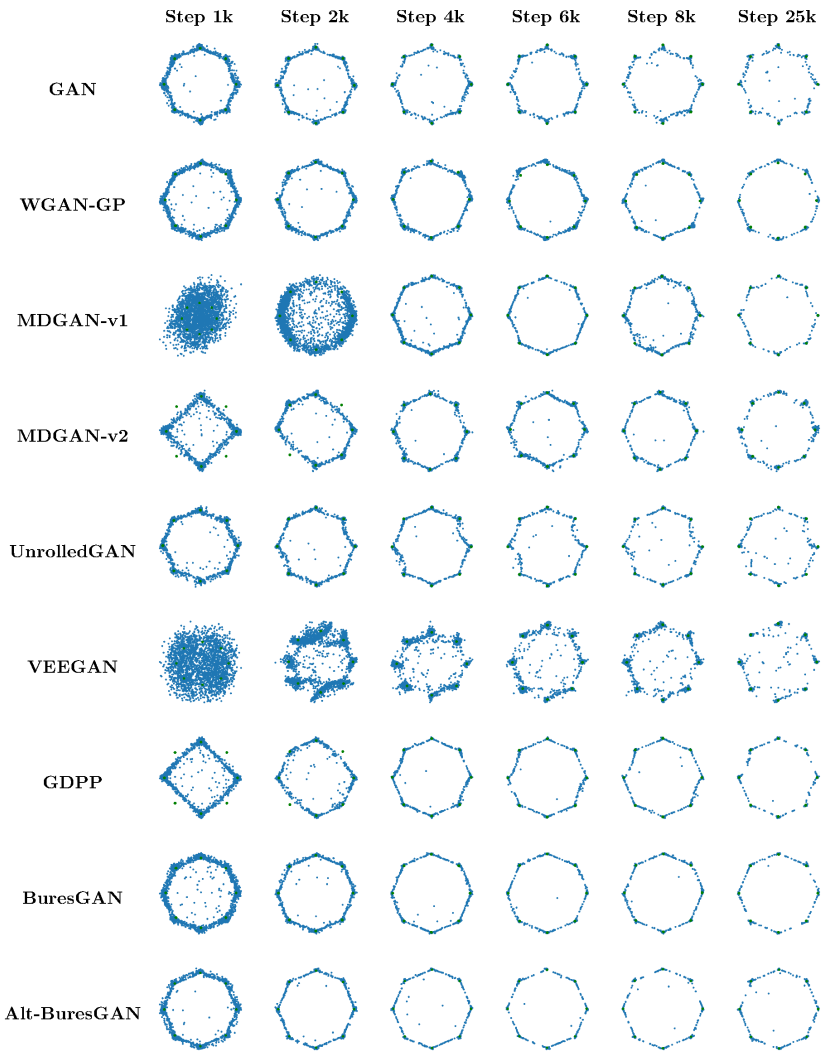

Fig. 1. The progress of different GANs on the synthetic ring example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

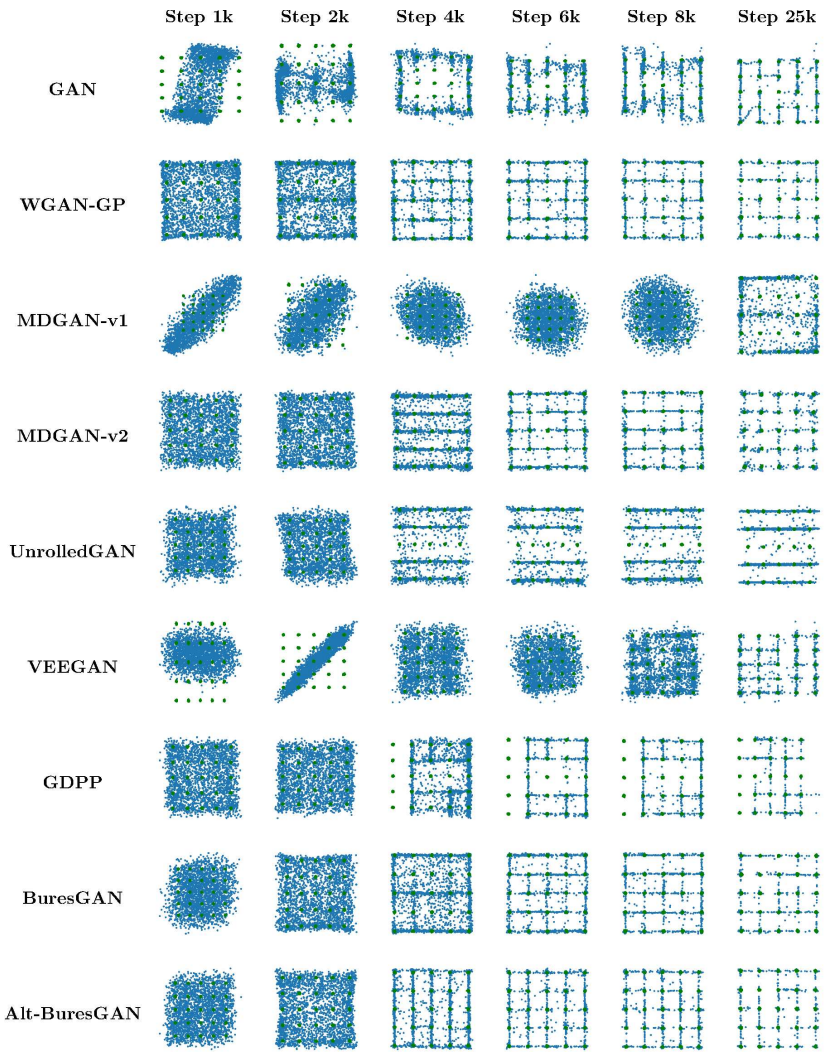

Fig. 2. The progress of different GANs on the synthetic grid example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

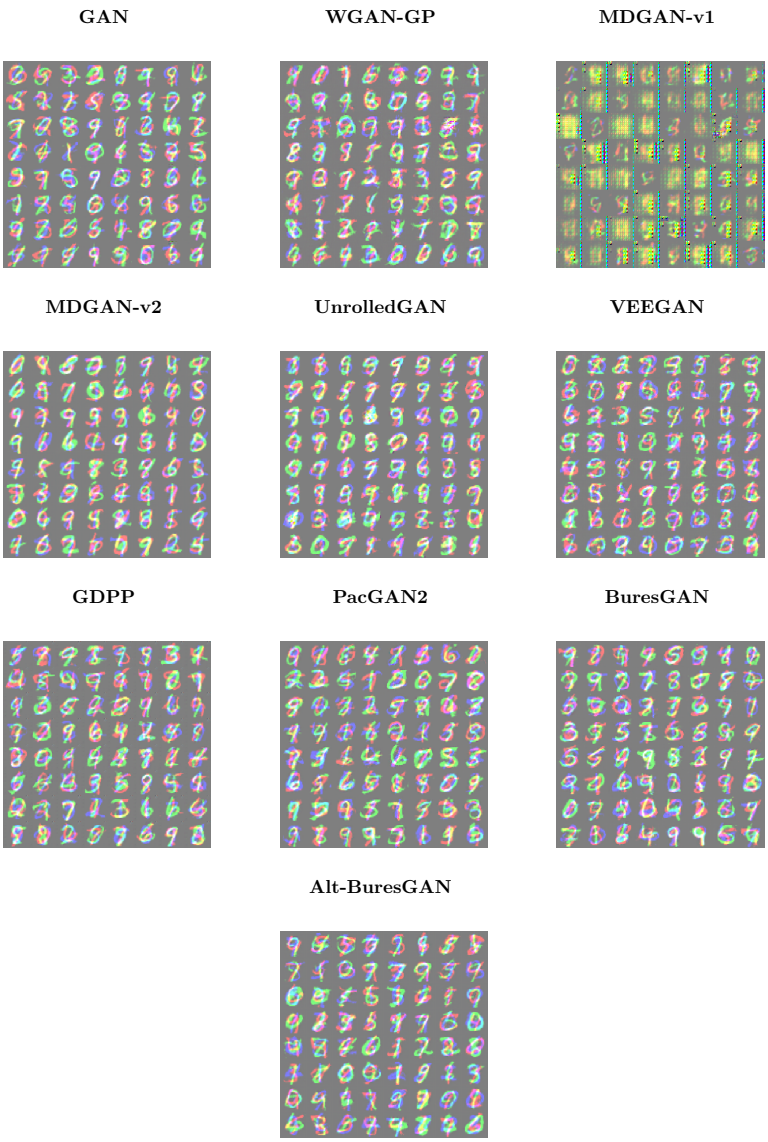

Fig. 3. Generated images for the Stacked MNIST dataset. Each model is trained with 3 layers and mini-batch size 256. Each square shows 64 samples from the trained generator.

Fig. 4. Generated images for CIFAR-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

Fig. 5. Generated images for CIFAR-100 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

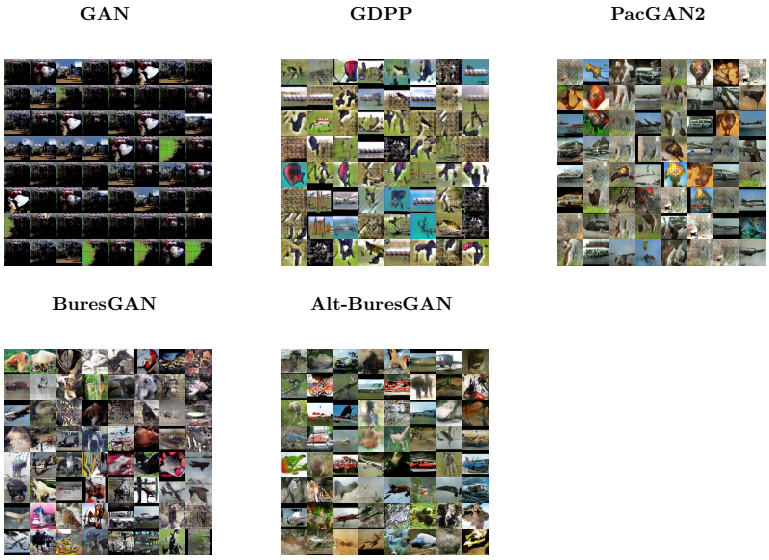

Fig. 6. Generated images for STL-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

5 Architectures

5.1 Synthetic Architectures

Following the recommendation in the original work of [2], the same fully-connected architecture is used for the VEEGAN reconstructor in all experiments.

Table 7. The generator and discriminator architectures for the synthetic examples.

Layer	Output	Activation
Input	256	-
Dense	128	tanh
Dense	128	tanh
Dense	2	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	1	-

Table 8. Respectively the MDGAN encoder model and VEEGAN stochastic inverse generator architectures for the synthetic examples. The output of the VEEGAN models are samples drawn from a normal distribution with scale 1 and where the location is learned.

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	256	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	256	tanh
Normal	256	-

5.2 Stacked MNIST Architectures

For clarity, we report in Table 9 the settings of different papers for the Stacked MNIST experiment. This helps to clarify the difference between our simulations and other papers' simulations. On the one hand, in Table 10, the architecture of the experiment reported in the paper is described. On the other hand, the architecture of the supplementary experiments on Stacked MNIST is given in Table 11. Details of the architecture of MDGAN are also reported in Table 13.

Table 9. The hyperparameters and architectures of the stacked MNIST experiment performed by the different methods are given here. This represents our best efforts at providing a comparison between the different parameters used. The asterisks ** indicate that the parameters where obtained by the Github repository. Notice that GDPP paper also used 30000 iterations for training DCGAN and unrolled GAN (indicated by *). BuresGAN's column refers to the settings of experiment of Table 2 in the paper and Table 6 in SM respectively, for which the different values are separated by the symbol &.

Parameters	BuresGAN	PacGAN	VEEGAN	GDPP
learning rates	1×10^{-3} & 2×10^{-4}	2×10^{-4}	2×10^{-4}	1×10^{-4} **
learning rate decay	no	no	no	no**
Adam β_1	0.5	0.5	0.5	0.5
Adam β_2	0.999	0.999	0.999	0.9
iterations	25000	20000	?	15000*
disc. conv. layers	3 & 4	4	4	3
gen. conv. layers	3 & 4	4	4	3
batch size	64, 128, 256 & 64	64	64	64
evaluation samples	10000	26000	26000	26000
ℓ (i.e., z dimension)	100	100	100	128
z distribution	$\mathcal{N}(0, \mathbb{I}_\ell)$	unif$[-1,1]^\ell$	unif$[-1,1]^\ell$ **	unif$[-1,1]^\ell$

Table 10. The generator and discriminator architectures for the Stacked MNIST experiments with 4 layers for both generator and discrimintor. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	2048	ReLU	Yes
Reshape	2, 2, 512	-	-
Conv'	4, 4, 256	ReLU	Yes
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Conv	2, 2, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table 11. The generator and discriminator architectures for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not. For the experiments with 2 convolution layers in Table 4, the final convolution layer is removed in the discriminator.

Layer	Output	Activation	BN
Input	100	-	-
Dense	12544	ReLU	Yes
Reshape	7, 7, 256	-	-
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table 12. The CNN architecture of the classifier used during the evaluation of the stackedMNIST experiments. Dropout with a rate of 0.5 is used before the final dense layer.

Layer	Output	Activation
Input	28, 28,1	-
Conv	24, 24, 32	ReLU
MaxPool	12, 12, 32	-
Conv	8, 8, 64	ReLU
MaxPool	4, 4, 64	-
Flatten	-	-
Dense	1024	ReLU
Dense	10	-

Table 13. The MDGAN encoder model architecture for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 3	ReLU	Yes
Conv	7, 7, 64	ReLU	Yes
Conv	7, 7, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

5.3 CIFAR-10 and 100 DCGAN Architectures

Convolutional architectures for CIFAR data set are reported in Table 14 and Table 15 below.

Table 14. The generator and discriminator architectures for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	16384	ReLU	Yes
Reshape	8, 8, 256	-	-
Conv'	8, 8, 128	ReLU	Yes
Conv'	16, 16, 64	ReLU	Yes
Conv'	32, 32, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	32, 32, 3	-	-
Conv	16, 16, 64	Leaky ReLU	No
Conv	8, 8, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table 15. The MDGAN encoder model architecture for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	32, 32, 3	-	-
Conv	16, 16, 3	ReLU	Yes
Conv	8, 8, 64	ReLU	Yes
Conv	8, 8, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

5.4 STL-10 DCGAN Architectures

In Table 16, the convolutional architecture is described for the $96 \times 96 \times 3$ STL-10 data set, while MDGAN details are reported in Table 17.

Table 16. The generator and discriminator architectures for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	36864	ReLU	Yes
Reshape	12, 12, 256	-	-
Conv'	12, 12, 256	ReLU	Yes
Conv'	24, 24, 128	ReLU	Yes
Conv'	48, 48, 64	ReLU	Yes
Conv'	96, 96, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	96, 96, 3	-	-
Conv	48, 48, 64	Leaky ReLU	No
Conv	24, 24, 128	Leaky ReLU	Yes
Conv	12, 12, 256	Leaky ReLU	Yes
Conv	6, 6, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table 17. The MDGAN encoder model architecture for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	96, 96, 3	-	-
Conv	48, 48, 3	ReLU	Yes
Conv	24, 24, 64	ReLU	Yes
Conv	12, 12, 128	ReLU	Yes
Conv	12, 12, 256	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

5.5 ResNet Architectures

For CIFAR-10, we used the ResNet architecture from the appendix of [1] with minor changes as given in Table 18. We used an initial learning rate of 5e−4 for CIFAR-10 and STL-10. For both datasets, the models are run for 200k iterations. For STL-10, we used a similar architecture that is given in Table 19.

Table 18. The generator (top) and discriminator (bottom) ResNet architectures for the CIFAR-10 experiments.

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$200 \cdot 4 \cdot 4$
Reshape	-	-	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 8 \times 8$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 16 \times 16$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 32 \times 32$
Conv, tanh	3×3	-	$3 \times 32 \times 32$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 16 \times 16$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 8 \times 8$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 4 \times 4$
ReLu, meanpool	-	-	200
Dense	-	-	1

Table 19. The generator (top) and discriminator (bottom) ResNet architectures for the STL-10 experiments. For the experiment with full-sized 96x96 images, an extra upsampling block was added to the generator.

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$200 \cdot 6 \cdot 6$
Reshape	-	-	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 48 \times 48$
Conv, tanh	3×3	-	$3 \times 48 \times 48$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 6 \times 6$
ReLu, meanpool	-	-	200
Dense	-	-	1

References

1. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: Advances in neural information processing systems 31 (2017)
2. Srivastava, A., Valkov, L., Russell, C., Gutmann, M.U., Sutton, C.: Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning. In: Advances in Neural Information Processing Systems 30 (2017)