THE BURES METRIC FOR TAMING MODE COLLAPSE IN GENERATIVE ADVERSARIAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Adversarial Networks (GANs) are performant generative methods yielding high-quality samples. However, under certain circumstances, the training of GANs can lead to mode collapse or mode dropping, i.e. the generative models not being able to sample from the entire probability distribution. To address this problem, we use the last layer of the discriminator as a feature map to study the distribution of the real and the fake data. During training, we propose to match the real batch diversity to the fake batch diversity by using the Bures distance between covariance matrices in feature space. The computation of the Bures distance can be conveniently done in either feature space or kernel space in terms of the covariance and kernel matrix respectively. We observe that diversity matching reduces mode collapse substantially and has a positive effect on the sample quality. On the practical side, a very simple training procedure, that does not require additional hyperparameter tuning, is proposed and assessed on several datasets.

1 INTRODUCTION

In several applications, datasets are assumed to be obtained by sampling an implicit probability distribution. The estimation of this empirical implicit distribution is often intractable, especially in high dimensions. Hence, the so-called generative models usually provide an algorithmic procedure in order to yield samples from the unknown distribution. Popular approaches are Variational Auto-Encoders (VAEs) (Kingma & Welling, 2014), Generating Flow models (Rezende & Mohamed, 2015) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). The latter are particularly successful approaches to produce high quality samples, especially in the case of natural images, although, the training of a GAN is notoriously difficult. GANs consist of two networks: a generator and a discriminator, where a generator network maps random noise, usually drawn from a multivariate normal, to fake data points that simulate the probability distribution of real data; and a discriminator network that estimates the likelihood ratio of the generator network to the data distribution. A specific issue – the 'mode collapse' – appears when, after training, a GAN is able to sample only from a few of the many modes of the distribution.

Contribution. We propose a generative adversarial network called BuresGAN, which has the objective function of a vanilla GAN complemented by a additional term, which promotes a matching of fake and real data in a feature space $\mathbb{R}^f$. Specifically, the squared Bures distance is used to match the covariance matrix of real and fake batches in feature space, so that the empirical effect of this term is to reduce mode collapse. Conveniently, the Bures distance also admits both a feature space and kernel based expression. Contrary to other related approaches (Che et al., 2017; Srivastava et al., 2017), the architecture of the GAN is unchanged, only the objective is modified. Importantly, a variant called alt-BuresGAN, which is trained with alternating minimization, achieved competitive performance with a simple training procedure that does not require hyperparameter tuning or additional regularization such as a gradient penalty. We empirically show that the proposed methods are robust when it comes to the choice of architecture and do not require an additional, fine, architecture search. Finally, we show competitive or better results compared with the state of the art on CIFAR-10 and STL-10 using a ResNet architecture.

Related works. The Bures distance is closely related to the Fréchet distance (Dowson & Landau, 1982) which is a 2-Wasserstein distance between multivariate normal distributions. Namely,

the Fréchet distance between multivariate normals of equal means is the Bures distance between their covariance matrices. The Bures distance is also equivalent to the exact expression for the 2-Wasserstein distance between two elliptically contoured distributions with the same mean (Gelbrich, 1990; Peyré et al., 2019). Noticeably, the Fréchet Inception Distance score (FID) is a popular manner to assess the quality of generative models. This score uses the Fréchet distance between real and generated samples in the feature space of a pre-trained inception network (Salimans et al., 2016; Heusel et al., 2017).

There exist numerous variants of GANs devoted to improving the GAN training. For the mode collapse evaluation, we compare BuresGAN to the most closely related works. Methods like GDPP-GAN (Elfeki et al., 2019) and VEEGAN (Srivastava et al., 2017), similarly to our proposed method, try to enforce diversity in 'latent' space. GDPP-GAN matches the eigenvectors and eigenvalues of the real and fake diversity kernel. In VEEGAN, an additional reconstructor network that maps the true data distribution to Gaussian random noise is introduced. The Unrolled-GAN (Metz et al., 2017) updates the generator with respect to the unrolled optimization of the discriminator. This allows the training to be adjusted between using the optimal discriminator in the generator's objective, which is ideal but infeasible in practice. Wasserstein GANs (Arjovsky et al., 2017; Gulrajani et al., 2017) leverage the 1-Wasserstein distance to produce a value function which has better theoretical properties than the original one. In MDGAN (Li et al., 2017), a regularization is added to the objective function, where the generator takes advantage of other similarity metrics with more predictable behavior. The idea is combined with an approach that penalizes the missing modes. More recent approaches to reducing mode collapse are variations of WGAN (Wu et al., 2018), or consider entropy regularization PresGAN (Dieng et al., 2020), while some theoretical studies can be found in (Lin et al., 2018; Xiao et al., 2018). Also, for image quality evaluation of Section 4, we compare BuresGAN to recent works using ResNets (see Table 5).

2 METHOD

A GAN consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks, and parametrized by real vectors. The value $D(\boldsymbol{x})$ gives the probability that $\boldsymbol{x}$ comes from the empirical distribution, while the generator G maps a point $\boldsymbol{z}$ in the latent space $\mathbb{R}^\ell$ to a point in the data space $\mathbb{R}^d$. The training of a GAN consists in solving

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))], \tag{1}$$

by alternating two phases of training. In equation 1, the first expectation is over the empirical data distribution p_d and the second is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. It is common to define the discriminator loss by

$$V_D = -\mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))], \tag{2}$$

which has to be minimized. In practice, it is proposed in (Goodfellow et al., 2014) to minimize generator loss

$$V_G = -\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\mathbb{I}_\ell)}[\log D(G(\boldsymbol{z}))], \tag{3}$$

rather than the second term of equation 1, for an improved training efficiency.

Matching real and fake data covariance. To prevent mode collapse, we encourage the generator to sample fake data of similar diversity to the real data. This is achieved by matching the sample covariance matrices of the real and fake data respectively, also referred to as 'covariance matching'. Similar ideas were explored for GANs in (Mroueh et al., 2017; Elfeki et al., 2019). In order to compare covariance matrices, we propose to use the squared Bures distance between positive semi-definite $\ell \times \ell$ matrices (Bhatia et al., 2019), i.e.,

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

Being a Riemannian metric on the manifold of positive semi-definite matrices (Massart & Absil, 2020), the Bures metric is adequate to compare covariance matrices – it is also related to an optimal transport distance as explained hereafter. The covariances are defined in a feature space associated to the discriminator. More precisely, the last layer of the discriminator, denoted by $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^f$,

defines a feature map, namely $D(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}))$, where $\boldsymbol{w}$ is the weight vector of the last dense layer and σ is the sigmoid function. We use the normalization $\bar{\boldsymbol{\phi}}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})/\|\boldsymbol{\phi}(\boldsymbol{x})\|_2$, after the centering of $\boldsymbol{\phi}(\boldsymbol{x})$. Then, we define a covariance matrix as follows: $C(p) = \mathbb{E}_{\boldsymbol{x}\sim p}[\bar{\boldsymbol{\phi}}(\boldsymbol{x})\bar{\boldsymbol{\phi}}(\boldsymbol{x})^\top]$. Denote $C_d = C(p_d)$ and $C_g = C(p_g)$ the real data and generated data covariance matrices. Our proposal is to replace the generator loss by $V_G + \lambda \mathcal{B}(C_d, C_g)^2$. The value $\lambda = 1$ was found to yield good results in the studied datasets. Two specific training algorithms are proposed. Algorithm 1 deals with the squared Bures distance as an additive term to the generator loss, while an alternating training is used in Algorithm 2 and does not introduce an extra parameter.

Algorithm 1: BuresGAN

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

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 $-V_D$;

The training described in Algorithm 1 is analogous to the training of GDPP GAN (Elfeki et al., 2019), although the additional generator loss is rather different. The computational advantage of the Bures distance is that it admits two expressions which can be evaluated numerically in a stable way. Namely, there is no need to calculate a gradient update through an eigendecomposition.

Feature space expression. In the training procedure, real $\boldsymbol{x}_i^{(d)}$ and fake data $\boldsymbol{x}_i^{(g)}$ with $i = 1, \dots, b$ are sampled respectively from the empirical distribution and the mapping of the normal distribution $\mathcal{N}(\mathbf{0}, \mathbb{I}_\ell)$ by the generator. Consider the case where the batch size b is larger than the feature space dimension. Let the embedding of the batches in feature space be $\Phi_\alpha = [\boldsymbol{\phi}(\boldsymbol{x}_1^{(\alpha)}), \dots, \boldsymbol{\phi}(\boldsymbol{x}_b^{(\alpha)})]^\top \in \mathbb{R}^{b\times f}$ with $\alpha = d, g$. The covariance matrix of one batch in feature space[1] is $\hat{C} = \bar{\Phi}^\top \bar{\Phi}$, where $\bar{\Phi}$ is the ℓ_2-normalized centered feature map of the batch. Numerical instabilities can be avoided by regularizing the covariance matrices by adding a small number, e.g. 10^{-14}, to its diagonal elements, so that, in practice, we only deal with strictly positive definite matrices. From the computational perspective, an interesting alternative expression for the Bures distance is given by

$$\mathcal{B}\left(C_d, C_g\right)^2 = \mathrm{Tr}\big(C_d + C_g - 2(C_g C_d)^{\frac{1}{2}}\big), \tag{4}$$

which requires only one matrix square root. This identity can be obtained from Lemma 1. Note that an analogous result is proved in (Oh et al., 2020).

Lemma 1. *Let A and B be $f \times f$ symmetric positive semidefinite matrices. Then, we have: (i) AB is diagonalizable with nonnegative eigenvalues, and (ii) Let $B = Y^\top Y$, then* $\mathrm{Tr}((AB)^{\frac{1}{2}}) = \mathrm{Tr}((YAY^\top)^{\frac{1}{2}})$.

Kernel based expression. Alternatively, if the feature space dimension f is larger than the batch size b, it is more efficient to compute $\mathcal{B}(\hat{C}_d, \hat{C}_g)$ thanks to $b \times b$ kernel matrices: $K_d = \bar{\Phi}_d \bar{\Phi}_d^\top$, $K_g = \bar{\Phi}_g \bar{\Phi}_g^\top$ and $K_{dg} = \bar{\Phi}_d \bar{\Phi}_g^\top$. Then, we have the kernel based expression

$$\mathcal{B}(\hat{C}_d, \hat{C}_g)^2 = \mathrm{Tr}\left(K_d + K_g - 2\left(K_{dg}K_{dg}^\top\right)^{\frac{1}{2}}\right), \tag{5}$$

which allows to calculate the Bures distance between covariance matrices by computing a matrix square root of a $b \times b$ matrix. This is a consequence of Lemma 2.

Lemma 2. *The matrices $X^\top X Y^\top Y$ and $Y X^\top X Y^\top$ are diagonalizable with nonnegative eigenvalues and share the same non-zero eigenvalues.*

Connection with Wasserstein GAN and integral probability metrics. The Bures distance is proportional to the 2-Wasserstein distance $\mathcal{W}_2$ between two ellipically contoured distributions, with the same mean (Gelbrich, 1990). For instance, in the case of multivariate normal distributions, we have

$$\mathcal{B}\left(A, B\right)^2 = \min_\pi \mathbb{E}_{(X,Y)\sim\pi}\|X - Y\|_2^2 \text{ s.t. } X \sim \mathcal{N}(0, A) \text{ and } Y \sim \mathcal{N}(0, B),$$

where the minimization is over the joint distributions π. More precisely, in this paper, we make the approximation that the implicit distribution of the real and generated data in the feature space $\mathbb{R}^f$

[1]For simplicity, we omit the normalization by $\frac{1}{b-1}$ in front of the covariance matrix.

(associated to $\boldsymbol{\phi}(\boldsymbol{x})$) are elliptically contoured with the same mean. Under different assumptions, the Generative Moment Matching Networks (Ren et al., 2016; Li et al., 2017) work in the same spirit, but use a different approach to match covariance matrices. On the contrary, WGAN uses the Kantorovich dual formula for the 1-Wasserstein distance: $\mathcal{W}_1(\alpha, \beta) = \sup_{f \in \mathrm{Lip}} \int f \, \mathrm{d}(\alpha - \beta)$, where α, β are signed measures. Generalizations of such integral formulae are called integral probability metrics (see for instance (Binkowski et al., 2018)). Here, f is the discriminator, so that the maximization over Lipschitz f functions plays the role of the maximization over discriminator parameters in the min-max game of equation 1. Then, in the training procedure, this maximization alternates with a minimization over the generator parameters.

We can now discuss the connection with Wasserstein GAN. Coming back to the definition of Bures-GAN, we can now explain that the 2-Wasserstein distance provides an upper bound on an integral probability metric. Then, if we assume that the densities are elliptically contoured distributions in feature space, the use of the Bures distance to calculate $\mathcal{W}_2$ allows to spare the maximization over the discriminator parameters – and this motivates why the optimization of $\mathcal{B}$ only influences updates of the generator in Algorithm 1 and Algorithm 2. Going more into detail, the 2-Wasserstein distance between two probability densities (w.r.t. the same measure) is equivalent to a Sobolev dual norm, which can be interpreted as an integral probability metric. Indeed, let the Sobolev seminorm $\|f\|_{H^1} = (\int \|\nabla f(x)\|^2 \, \mathrm{d}x)^{1/2}$. Then, its dual norm over signed measures is defined as $\|\nu\|_{H^{-1}} = \sup_{\|f\|_{H^1} \leq 1} \int f \, \mathrm{d}\nu$. It is then shown in (Peyre, 2018; Peyré et al., 2019) that there exist two positive constants c_1 and c_2 such that

$$c_1 \|\alpha - \beta\|_{H^{-1}} \leq \mathcal{W}_2(\alpha, \beta) \leq c_2 \|\alpha - \beta\|_{H^{-1}}.$$

Hence, the 2-Wasserstein distance gives an upper bound on an integral probability metric.

Algorithmic details. The matrix square root in equation 4 and equation 5 is obtained thanks to the Newton-Schultz algorithm which is inversion free and can be efficiently calculated on GPUs since it involves only matrix products. In practice, we found 15 iterations of the algorithm to be sufficient. A small regularization term $1\mathrm{e}{-14}$ is added for stability. The latent prior distribution is $\mathcal{N}(0, \mathbb{I}_\ell)$ with $\ell = 100$ and the parameter in Algorithm 1 is always set to $\lambda = 1$. In the tables hereafter, we indicate the largest scores in bold if they differ from lower scores by at least one std.

3 Empirical Evaluation of Mode Collapse

The BuresGAN and alt-BuresGAN performances are compared with several other methods on synthetic data, artificial and real images. Namely we compare with the standard DCGAN (Salimans et al., 2016), WGAN-GP (Gulrajani et al., 2017), MDGAN (Che et al., 2017), UnrolledGAN (Metz et al., 2017), VEEGAN (Srivastava et al., 2017) and GDPP (Elfeki et al., 2019). A low dimensional feature space ($f = 128$) is used for the synthetic data so that the feature space formula equation 4 is used, while the dual formula equation 5 is used for the image datasets (Stacked MNIST, CIFAR-10, CIFAR-100 and STL-10) for which the feature space is larger than the batch size. The architectures used for the image datasets are based on the DCGAN (Radford et al., 2016), while results using ResNets are given in Section 4. All images are scaled in between -1 and 1 before running the algorithms. Additional information on the architectures and datasets is given in the Appendix. The hyperparameters of other methods are typically chosen as suggested in the authors' reference implementation. The number of unrolling steps in Unrolled GAN (Metz et al., 2017) is chosen to be 5. For MDGAN (Che et al., 2017), both versions are implemented. The first version, which corresponds to the mode regularizer, has hyperparameters $\lambda_1 = 0.2$ and $\lambda_2 = 0.4$, for the second version, which corresponds to manifold diffusion training for regularized GANs, has $\lambda = 10^{-2}$. WGAN-GP (Gulrajani et al., 2017) uses $\lambda = 10.0$ and $n_{\mathrm{critic}} = 5$. All models are trained using Adam (Kingma & Ba, 2015) with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate 10^{-3} for both the generator and discriminator. Unless stated otherwise, the batch size is taken to 256. Examples of random generations of all the GANs are given in Appendix. Notice that in this section we report the results achieved only at the end of the training.

3.1 Artificial Data

Synthetic. The ring is a mixture of eight two-dimensional spherical Gaussians in the plane with means $2.5 \times (\cos((2\pi/8)i), \sin((2\pi/8)i))$ and std 0.01 for $i \in \{1, \ldots, 8\}$. The 2D-grid is a mixture

of 25 two-dimensional isotropic normals in the plane with means separated by 2 and with standard deviation 0.05. All models are trained for 25k iterations. The evaluation is done by sampling 3k points from the generator network. A sample is counted as high quality if it is within 3 standard deviations of the nearest mode. The experiments are repeated 10 times for all models and their performance is compared in Table 1. All methods have the same architecture, with $\ell = 256$, following (Elfeki et al., 2019). The proposed (Alt-)BuresGAN consistently captures all the modes and produces the highest quality samples. The training progress of the Alt-BuresGAN is shown on Figure 1, where we observe that all the modes early on in the training procedure, afterwards improving the quality. The training progress of the other GAN models listed in Table 1 is given in Appendix.

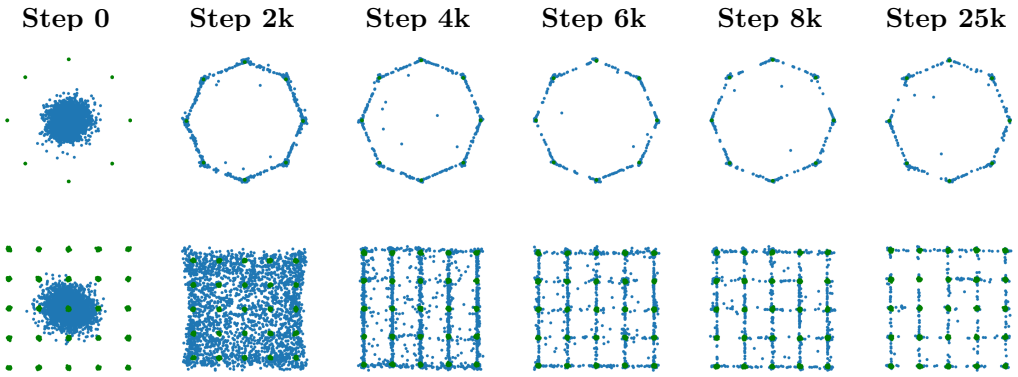

Figure 1: Figure accompanying Table 1, the progress of Alt-BuresGAN on the synthetic examples. Each column shows 3k samples from the training of the generator in blue and 3k samples from the true distribution in green.

	Grid with 25 modes			Ring with 8 modes		
	Nb modes	% in 3σ	time (s)	Nb modes	% in 3σ	time (s)
GAN	**22.9**(4)	76(13)	32(0.2)	7.4(2)	76(25)	32(0.4)
WGAN-GP	**24.9**(0.3)	77(10)	225(1)	7.1(1)	9(5)	224(1)
MDGAN-v1	21(3)	49(18)	65(0.2)	6.5(2)	65(26)	65(0.5)
MDGAN-v2	**25**(0)	68(11)	81(0.2)	5(3)	20(15)	81(0.8)
UnrolledGAN	19.7(1)	78(19)	98(0.2)	**8**(0)	77(18)	98(0.7)
VEEGAN	**25**(0)	67(3)	49(0.3)	**8**(0)	29(5)	49(0.5)
GDPP	20.5(5)	79(23)	392(25)	7.5(0.8)	73(25)	404(5)
Alt-BuresGAN	**25**(0)	**84**(1)	298(0.6)	**8**(0)	**84**(6)	298(1)
BuresGAN	**25**(0)	82(1)	279(0.4)	**8**(0)	82(4)	278(1)

Table 1: Experiments on the synthetic datasets with a GPU Nvidia Quadro P4000. Average(std) over 10 runs. All the models are trained for 25k iterations. The largest averaged values are in bold.

Stacked MNIST. The Stacked MNIST dataset is specifically constructed to contain 1000 known modes. This is done by stacking three digits, sampled uniformly at random from the original MNIST dataset, each in a different channel. The proposed GANs are compared to the other models for different batch sizes in Tables 2. All models are trained for 25k iterations. For the evaluation of the performance we follow (Metz et al., 2017) and use the following metrics: the number of captured modes measures mode collapse, with the KL divergence also measures sample quality. The mode of each generated image is identified by using a standard MNIST classifier which is trained up to 98.43% accuracy on the validation set (see Appendix), that classifies each channel of the fake sample. The quality of samples is evaluated by computing the KL-divergence between generated label distribution and training labels distribution. The same classifier is used to count the number of captured modes. The metrics are calculated based on 10k generated images for all the models. 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, 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. WGAN-GP on average shows superior performance when using a discriminator with 2 convolutional layers (see Appendix) 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, on average, the training of these models has a tendency to collapse more often, indicated by the large std. 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. Generated samples from the Alt-BuresGAN are given in Figure 2.

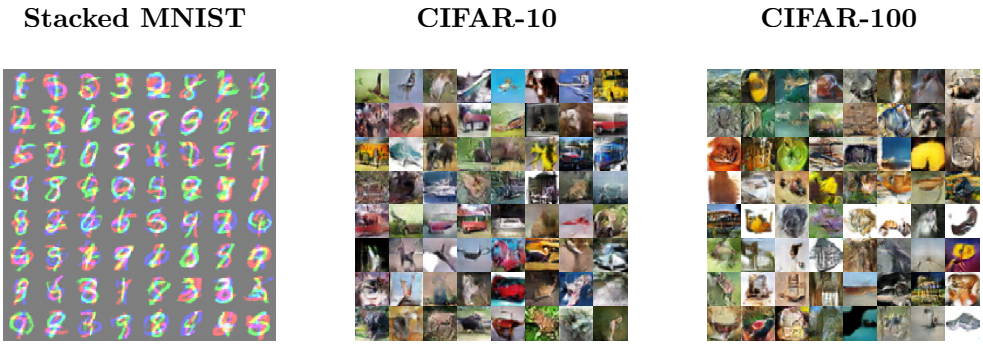

Figure 2: Generated samples from a trained Alt-BuresGAN, with a DCGAN architecture.

		Nb modes(↑)			KL div.(↓)		
	Batch size	64	128	256	64	128	256
3 conv. layers	GAN	**993.3**(3.1)	**995.4**(1.7)	**998.3**(1.2)	0.28(0.02)	**0.24**(0.02)	**0.21**(0.02)
	WGAN-GP	980.2(57)	838.3(219)	785.1(389)	**0.26**(0.34)	1.05(1)	1.6(2.4)
	MDGAN-v1	233.8(250)	204.0(202)	215.5(213)	5.0(1.6)	4.9(1.3)	5.0(1.2)
	MDGAN-v2	299.9(457)	300.4(457)	200.0(398)	4.8(3.0)	4.7(3.0)	5.5(2.6)
	UnrolledGAN	934.7(107)	874.1(290)	884.9(290)	0.72(0.51)	0.98(1.46)	0.90(1.4)
	VEEGAN	974.2(10.3)	687.9(447)	395.6(466)	0.33(0.05)	2.04(2.61)	3.52(2.64)
	GDPP	894.2(298)	897.1(299)	**997.5**(1.4)	0.92(1.92)	0.88(1.93)	**0.20**(0.02)
	BuresGAN	**993.5**(2.7)	**996.3**(1.6)	**997.1**(2.4)	0.29(0.02)	**0.25**(0.02)	**0.23**(0.01)
	Alt-BuresGAN	**995.0**(2.4)	**995.6**(3.6)	**995.5**(2.4)	0.28(0.03)	**0.27**(0.02)	**0.24**(0.02)

Table 2: KL-divergence between the generated distribution and true distribution. The number of counted modes indicates the amount of mode collapse. 25k iterations and average(std) over 10 runs.

3.2 Real Images

Metrics. The image quality is assessed thanks to the Inception Score (IS), Fréchet Inception Distance (FID), Inference via Optimization (IvO) and Sliced Wasserstein Distance (SWD). The latter was also used in (Elfeki et al., 2019; Karras et al., 2017) to evaluate the quality of images as well as the severity of mode-collapse. The multiscale statistical similarity between distributions of local image patches drawn from Laplacian pyramids are evaluated using the SWD. A small Wasserstein distance indicates that the distribution of the patches is similar, thus real and fake images appear similar in both appearance and variation at this spatial resolution. We always show the average SWD to evaluate performance. IvO (Metz et al., 2017) measures mode collapse by comparing real images with the nearest generated image. IvO optimizes within the latent space to find the closest generated image to a given test image, and returns the distance which can be large in the case of mode collapse. The metrics are calculated based on 10k generated images for all the models.

CIFAR datasets. In Table 3, we evaluate the performance on the $32 \times 32 \times 3$ CIFAR datasets. While all models are trained for 100k iterations, the best performance is observed for BuresGAN and Alt-BuresGAN in terms of image quality, measured by FID and Inception Score, and in terms of mode collapse, measured by SWD and IvO. We also notice that UnrolledGAN, VEEGAN and WGAN-GP have difficulty converging to a satisfactory result for this architecture. This in contrast to the 'simpler' synthetic data and the Stacked MNIST dataset, where the models get comparable performance to BuresGAN and Alt-BuresGAN. In the paper (Arjovsky et al., 2017), WGAN-GP achieves a very good performance on CIFAR-10 with a ResNet architecture which is considerably more complicated than the DCGAN used here. Also, for this architecture and number of training iterations, MDGAN-v1 and MDGAN-v2 did not converge to a meaningful result in our simulations. CIFAR-100 dataset consists of 100 different classes and is therefore more diverse. Compared to the

	CIFAR-10				CIFAR-100			
	IvO(↓)	IS(↑)	FID(↓)	SWD(↓)	IvO(↓)	IS(↑)	FID(↓)	SWD(↓)
GAN	0.30(0.06)	5.67(0.22)	59(8.5)	3.7(0.9)	0.37(0.10)	5.2(1.1)	91.7(66)	7.8(4.9)
WGAN-GP	0.49(0.24)	2.01(0.47)	291(87)	8.3(1.9)	0.54(0.27)	1.2(0.5)	283(113)	9.7(2.5)
UnrolledGAN	0.36(0.08)	3.1(0.6)	148(42)	9.0(5)	0.44(0.10)	3.2(0.7)	172.9(40)	13.1(9.2)
VEEGAN	0.36(0.08)	2.5(0.6)	198(33.5)	12.0(3)	0.35(0.08)	2.8(0.7)	177.2(27)	12.8(3.9)
GDPP	0.29(0.05)	5.76(0.27)	62.1(5.5)	4.1(1.1)	0.37(0.05)	5.9(0.2)	65.0(8)	4.4(1.9)
BuresGAN	0.30(0.03)	**6.34**(0.17)	**43.7**(0.9)	**2.1**(0.6)	0.36(0.08)	**6.5**(0.1)	**47.2**(1.2)	**2.1**(1.0)
Alt-BuresGAN	0.29(0.03)	**6.23**(0.07)	**45.4**(2.8)	**1.7**(0.9)	0.33(0.05)	**6.4**(0.1)	**49.4**(3.4)	**1.8**(0.6)

Table 3: Generation quality on CIFAR-10 and CIFAR-100 with DCGAN architecture. Average(std) over 10 runs. 100k iterations for each. SWD score was multiplied by 100.

original CIFAR-10 dataset, the performance of the proposed GANs remains mostly the same, with a small increase in IvO. Especially vanilla GAN shows a higher presence of mode collapse (measured by IvO and SWD).

STL-10. The STL-10 dataset includes higher resolution images of size $96 \times 96 \times 3$. The best performing models from previous experiments are trained for 150k iterations. Samples of generated images from a trained Alt-BuresGAN are given on Figure 3. The metrics are calculated based on 5k generated images for all the models. Compared to the previous datasets, GDPP and vanilla GAN are rarely able to generate high quality images on the higher resolution STL-10 dataset. Only BuresGANs are capable of consistently generating high quality images as well as preventing mode collapse, for the same architecture.

Figure 3: Samples CIFAR-10 (left) and STL-10 (right) images from BuresGAN with a ResNet architecture.

	IvO(↓)	IS(↑)	FID(↓)	SWD(↓)
GAN	0.50(0.15)	2.9(1.8)	237(54)	12.3(4.1)
GDPP	**0.46**(0.09)	3.3(2.2)	232(84)	8.2(4.0)
BuresGAN	**0.44**(0.05)	**7.6**(0.3)	**109**(7)	**2.3**(0.3)
Alt-BuresGAN	**0.45**(0.04)	**7.5**(0.3)	**110**(4)	**2.8**(0.4)

Table 4: Generation quality on STL-10 with DCGAN architecture. Average(std) over 5 runs. 150k iterations for each. For readability, SWD score was multiplied by 100.

Timings. The timings for these datasets are in the Appendix. For the same number of iterations, (alt-)BuresGAN training time is comparable to WGAN-GP training for the simple data in Table 1. For more complicated architectures, the (alt-)BuresGAN scales better and the training time was observed to be significantly shorter with respect to WGAN-GP and several other methods.

4 High Quality Generation using a ResNet Architecture

As noted by (Lucic et al., 2018), a fair comparison should involve GANs with the same architecture, and this is why we restricted in our paper to a classical DCGAN architecture. It is natural to question the performance of BuresGAN with a ResNet architecture. Hence, we trained the BuresGANs on the CIFAR-10 and STL-10 dataset using the ResNet architecture taken from (Gulrajani et al., 2017). The results are displayed in Table 5, where BuresGAN is compared with the state of the art of unconditional GAN models that use a ResNet architecture. In contrast with Section 3, we report here the best performance achieved at any time during the training, averaged over several runs. To the best of our knowledge, our method achieves a new state of the art inception score on STL-10 and is within a standard deviation of state of the art on CIFAR-10 using a ResNet architecture. A visual inspection (see Figure 3) shows that the high inception score is warranted, the samples are clear, diverse and often recognizable.

	CIFAR-10 (↑)	STL-10 (↑)
WGAN-GP ResNet (Gulrajani et al., 2017)*	7.86(0.07)	/
InfoMax-GAN Lee et al. (2019)*	8.08(0.08)	8.54(0.12)
SN-GAN ResNet (Miyato et al., 2018)*	8.22(0.05)	9.10(0.04)
ProgressiveGAN (Karras et al., 2017)*	**8.80**(0.05)	/
CR-GAN (Zhang et al., 2020)*	8.4	/
NCSN (Song & Ermon, 2019)*	**8.87**(0.12)	/
Improving MMD GAN (Wang et al., 2019)*	8.29	9.34
BuresGAN ResNet	**8.81**(0.08)	**10.31**(0.03)

Table 5: Best achieved IS (↑), using a ResNet architecture. Results with an asterisk are quoted from their respective papers, std in parenthesis. BuresGAN results were averaged over 3 runs.

5 Conclusion

In this work, we discussed an additional term based on the Bures distance which, when added to the GAN objective, promotes similarity of the distribution of the generated and real data in feature space $\mathbb{R}^f$. The Bures distance admits both a feature space and kernel based expression, which makes the proposed model time and data efficient when compared to state of the art models. Two training procedures are proposed: Algorithm 1 deals with the squared Bures distance as an additive term to the generator loss, while an alternating training is used in Algorithm 2 which does not introduce an extra parameter. Our experiments show that the proposed methods are capable of reducing mode collapse and, on the real datasets, achieve a clear improvement of sample quality without parameter tuning and without the need for regularization such as a gradient penalty. Moreover, the proposed GANs show a stable performance over different architectures, datasets and hyperparmeters.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165 – 191, 2019.

Mikolaj Binkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode Regularized Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Adji B. Dieng, Francisco J. R. Ruiz, David M. Blei, and Michalis K. Titsias. Prescribed generative adversarial networks. *arxiv:1910.04302*, 2020.

D.C Dowson and B.V Landau. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450 – 455, 1982.

Mohamed Elfeki, Camille Couprie, Morgane Riviere, and Mohamed Elhoseiny. GDPP: Learning Diverse Generations using Determinantal Point Processes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Matthias Gelbrich. On a Formula for the L2 Wasserstein Metric between Measures on Euclidean and Hilbert spaces. *Mathematische Nachrichten*, 147(1):185–203, 1990.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in neural information processing systems 31*, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30*, 2017.

Yoopyo Hong and Roger A. Horn. The Jordan Canonical Form of a Product of a Hermitian and a Positive Semidefinite Matrix. *Linear Algebra and its Applications*, 147:373 – 386, 1991.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Kwot Sin Lee, Ngoc-Trung Tran, and Ngai-Man Cheung. Infomax-gan: Mutual information maximization for improved adversarial image generation. In *NeurIPS 2019 Workshop on Information Theory and Machine Learning*, 2019.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabas Poczos. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *Advances in Neural Information Processing Systems 30*, 2017.

Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems 31*. 2018.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in neural information processing systems*, pp. 700–709, 2018.

Estelle Massart and P.-A. Absil. Quotient Geometry with Simple Geodesics for the Manifold of Fixed-Rank Positive-Semidefinite Matrices. *SIAM Journal on Matrix Analysis and Applications*, 41(1):171–198, 2020.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Youssef Mroueh, Tom Sercu, and Vaibhava Goel. McGan: Mean and Covariance Feature Matching GAN. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Jung Hun Oh, Maryam Pouryahya, Aditi Iyer, Aditya P. Apte, Joseph O. Deasy, and Allen Tannenbaum. A novel kernel Wasserstein distance on Gaussian measures: An application of identifying dental artifacts in head and neck computed tomography. *Computers in Biology and Medicine*, 120:103731, 2020.

Gabriel Peyré, Marco Cuturi, et al. Computational Optimal Transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Rémi Peyre. Comparison between W_2 distance and $\dot{H}^{-1}$ norm, and Localization of Wasserstein distance. *ESAIM: COCV*, 24(4):1489–1501, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Yong Ren, Jun Zhu, Jialian Li, and Yucen Luo. Conditional Generative Moment-Matching Networks. In *Advances in Neural Information Processing Systems 29*, 2016.

Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *Proceedings of the 32th International Conference on Machine Learning (ICML)*, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *Advances in neural information processing systems 29*, 2016.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930, 2019.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems 30*, 2017.

Wei Wang, Yuan Sun, and Saman Halgamuge. Improving MMD-GAN training with repulsive loss function. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. Wasserstein Divergence for GANs. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018.

Chang Xiao, Peilin Zhong, and Changxi Zheng. BourGAN: Generative Networks with Metric Embeddings. In *Advances in Neural Information Processing Systems 32*, 2018.

Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

A PROOFS

Proof of Lemma 1. (i) is a consequence of Corollary 2.3 in (Hong & Horn, 1991). (ii) We now follow (Oh et al., 2020). Thanks to (i), we have $AB = PDP^{-1}$ where D is a nonnegative diagonal and the columns of P contain the right eigenvectors of AB. Therefore, $\mathrm{Tr}((AB)^{1/2}) = \mathrm{Tr}(D^{1/2})$. Then, $YAY^\top$ is clearly diagonalizable. Let us show that it shares its nonzero eigenvalues with AB. a) We have $ABP = PD$, so that, by multiplying of the left by Y, it holds that $(YAY^\top)YP = YPD$. b) Similarly, suppose that we have the eigenvalue decomposition $YAY^\top Q = Q\Lambda$. Then, we have $BAY^\top Q = Y^\top Q\Lambda$ with $B = Y^\top Y$. This means that the non-zero eigenvalues of $YAY^\top$ are also eigenvalues of BA. Since A and B are symmetric, this completes the proof. □

Proof of Lemma 2. The result follows from Lemma 1 and its proof, where $A = X^\top X$ and $B = Y^\top Y$. □

B 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

$$\mathcal{B}(A,B)^2 = \min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A + B - 2(A^{\frac{1}{2}}BA^{\frac{1}{2}})^{\frac{1}{2}}), \tag{6}$$

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

$$\min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A) + \mathrm{Tr}(B) - 2\max_{U\in O(\ell)} \mathrm{Tr}(A^{1/2}B^{1/2}U) \tag{7}$$

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 3. *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 $MM^\top = W\Sigma^2W^\top$ and $M^\top M = V\Sigma^2V^\top$. 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 $U' = V^\top UW$. 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 6. Notice that the role of A and B can be exchanged in equation 6 since U is orthogonal.

C TRAINING DETAILS

C.1 SYNTHETIC ARCHITECTURES

Following the recommendation in the original work (Srivastava et al., 2017), the same fully-connected architecture is used for the VEEGAN reconstructor in all experiments.

C.2 STACKED MNIST ARCHITECTURES

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 6: The generator and discriminator architectures for the synthetic examples.

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	-

Table 7: 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	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
Flatten	-	-	-
Dense	1	-	-

Table 8: The generator and discriminator architectures 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	-	-

Table 9: The MDGAN encoder model architecture for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not.

C.3 CIFAR-10 AND 100 DCGAN ARCHITECTURES

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 10: 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	32, 32, 3	-	-
Conv	16, 16, 3	ReLU	Yes
Conv	8, 8, 64	ReLU	Yes
Conv	8, 8, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

Table 11: 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.

C.4 STL-10 DCGAN ARCHITECTURES

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 12: 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	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	-	-

Table 13: The MDGAN encoder model architecture for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

C.5 ResNet Architectures

For CIFAR-10, we used the ResNet architecture from the appendix of Gulrajani et al. (2017) with minor changes as given in Table 14. We used an initial learning rate of $5\mathrm{e}{-4}$ for CIFAR-10 and $1\mathrm{e}{-3}$ for STL-10. For both data sets, the models are run for 200k iterations. For STL-10, we used a similar architecture that is given in Table 15.

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

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$128 \cdot 6 \cdot 6$
Reshape	-	-	$128 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	up	$128 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	up	$128 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	up	$128 \times 48 \times 48$
ResBlock	$[3 \times 3] \times 2$	up	$128 \times 96 \times 96$
Conv, tanh	3×3	-	$3 \times 96 \times 96$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$128 \times 48 \times 48$
ResBlock	$[3 \times 3] \times 2$	Down	$128 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	Down	$128 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	-	$128 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	-	$128 \times 12 \times 12$
ReLu, meanpool	-	-	128
Dense	-	-	1

Table 15: The generator (top) and discriminator (bottom) ResNet architectures for the STL-10 experiments.

D ADDITIONAL EXPERIMENTS

D.1 TIMINGS

The timings per iteration for the experiments presented in the paper are listed in Table 16. Times are given for all the methods considered, although some method do not always generate meaningful images for all datasets. They are measured for 50 iterations after the first 5 iterations, and 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 machine equipped with different GPUs.

	stacked MNIST	CIFAR-10	CIFAR-100	STL-10
GAN	$\mathbf{0.54}(0.0005)$	$\mathbf{0.65}(0.02)$	$\mathbf{0.64}(0.0008)$	$\mathbf{6.00}(0.01)$
WGAN-GP	$2.99(0.004)$	$3.41(0.009)$	$3.41(0.006)$	$36.5(0.03)$
UnrolledGAN	$1.90(0.002)$	$2.17(0.003)$	$2.18(0.004)$	$21.99(0.06)$
MDGAN-v1	$1.24(0.002)$	$1.47(0.001)$	$1.47(0.002)$	$13.35(0.03)$
MDGAN-v2	$1.66(0.002)$	$1.98(0.002)$	$1.98(0.002)$	$18(0.03)$
VEEGAN	$0.56(0.006)$	0.66 (0.006)	$0.65(0.004)$	$6.10(0.03)$
GDPP	$0.69(0.02)$	$0.80(0.02)$	$0.80(0.02)$	$7.46(0.03)$
BuresGAN	$0.72(0.02)$	$0.82(0.001)$	$0.82(0.0008)$	$7.6(0.03)$
Alt-BuresGAN	$0.98(0.008)$	$1.15(0.007)$	$1.15(0.007)$	$10.10(0.03)$

Table 16: Average time per iteration in seconds. Averaged over 5 runs, with std in parenthesis. The batch size is 256. For Stacked MNIST, we use a discriminator architecture with 3 conv. layers.

D.2 BEST INCEPTION SCORES ACHIEVED WITH DCGAN ARCHITECTURE

The inception scores for the best trained models are listed in Table 17. 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.

	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

Table 17: Inception Score for the best trained models on CIFAR-10, CIFAR-100 and STL-10, with a DCGAN architecture (higher is better).

D.3 Influence of the number of convolutional layers for DCGAN architecture

Also, we provide in Figure 18 results with a DCGAN architecture including only 2 conv. layers for the discriminator in contrast to Table 2 which uses 3 conv. layers.

		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)
	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 18: 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 with a discriminator with 2 convolutional layers.

E Additional Figures

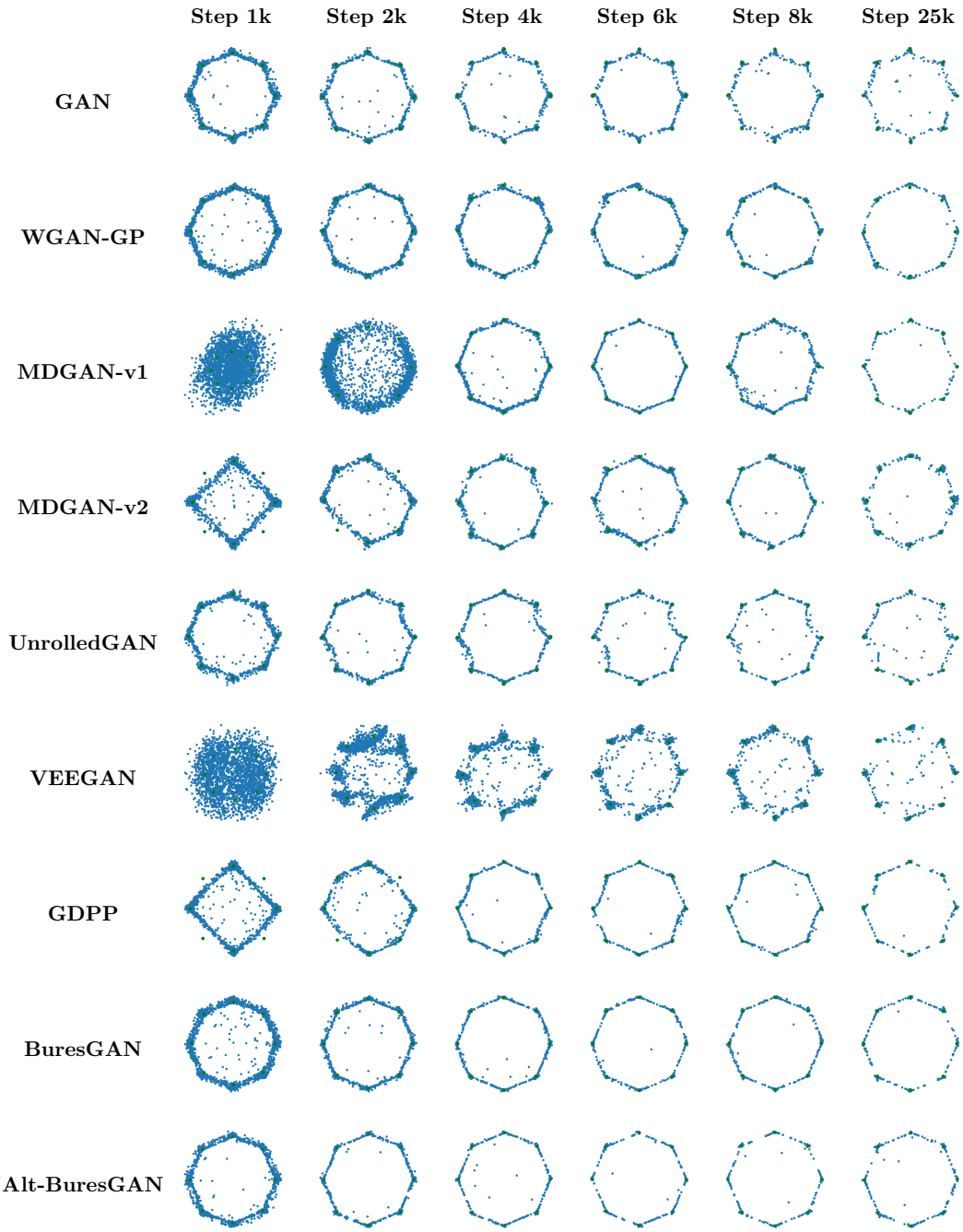

Figure 4: The progress of different GANs on the synthetic ring example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

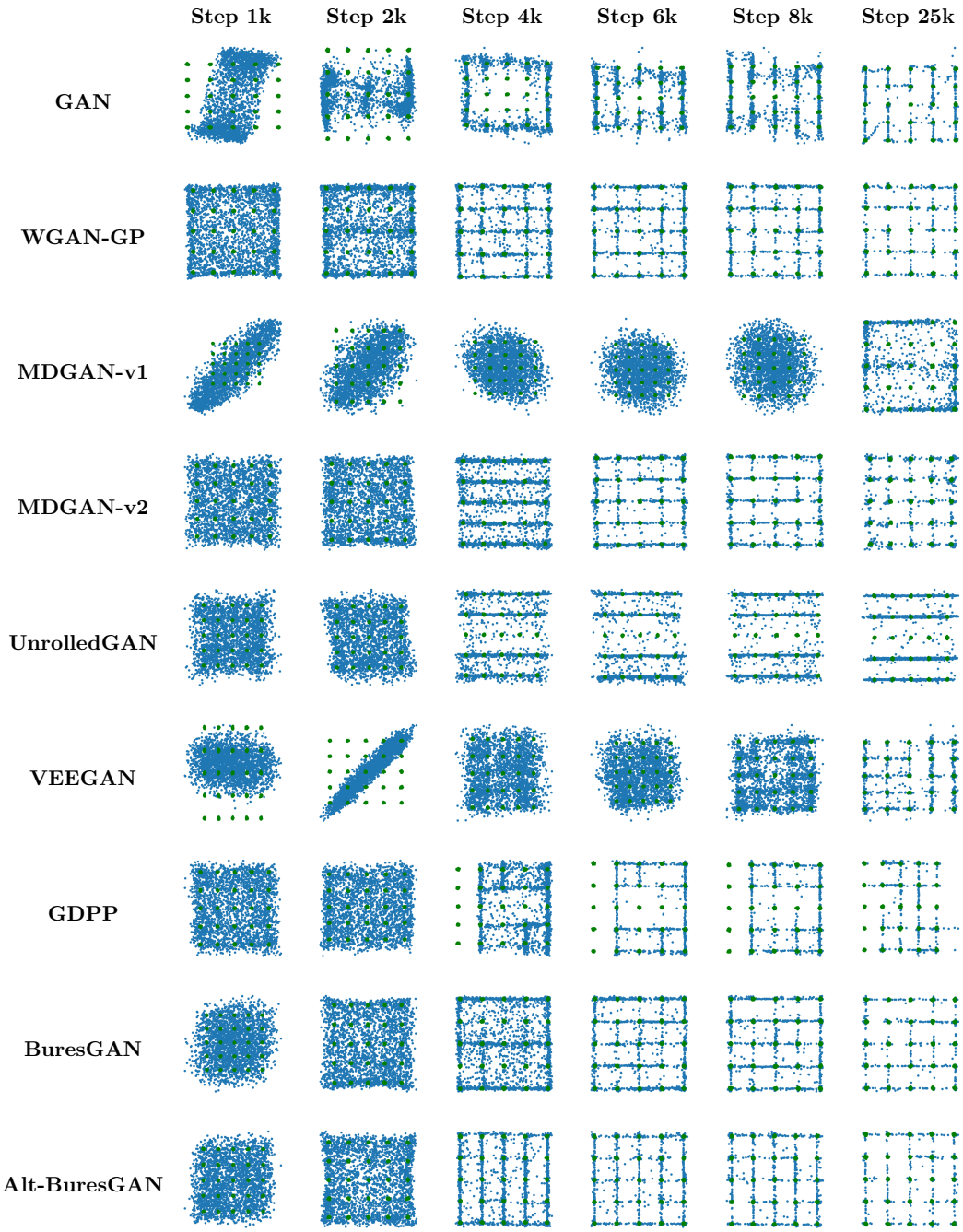

Figure 5: The progress of different GANs on the synthetic grid example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

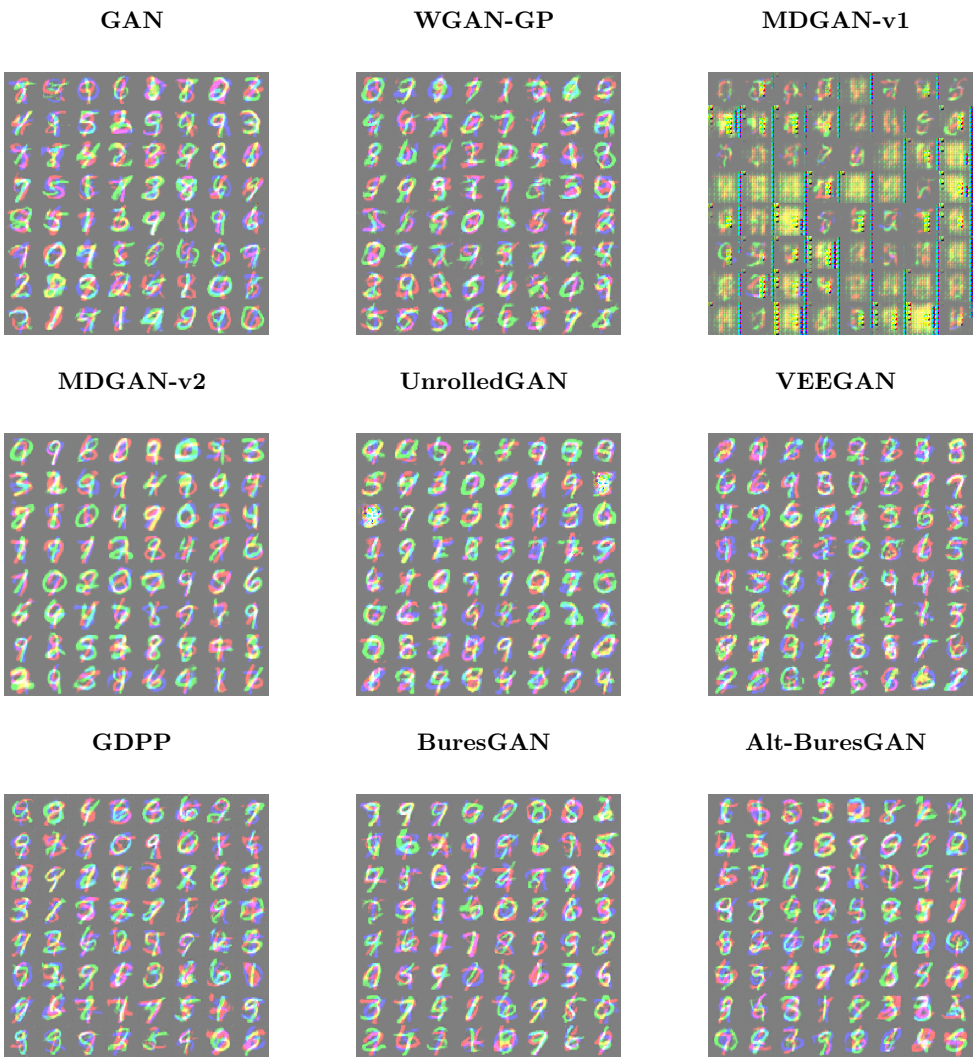

Figure 6: Generated images for the Stacked MNIST dataset. Each model is trained with 3 layers and mini-batch size 256 Each column shows 64 samples from the trained generator.

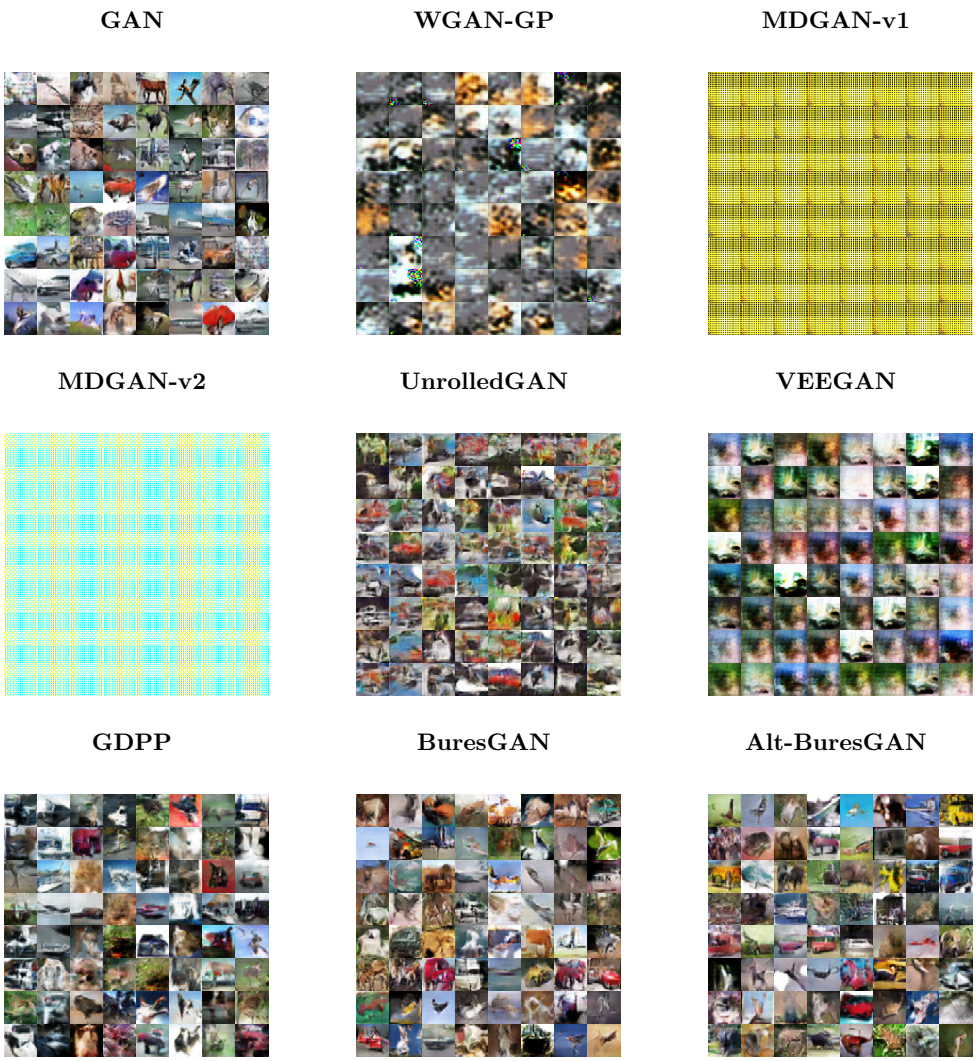

Figure 7: Generated images for CIFAR-10 using a DCGAN architecture. Each column shows 64 samples from the trained generator.

Figure 8: Generated images for CIFAR-100 using a DCGAN architecture. Each column shows 64 samples from the the trained generator.

Figure 9: Generated images for STL-10 using a DCGAN architecture. Each column shows 64 samples from the trained generator.