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) [1], Generating Flow models [2] and Generative Adversarial Networks (GANs) [3]. 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 to add an additional term to the vanilla GAN objective which promotes feature maps such that the distribution of fake and real data in a feature space \( \mathbb{R}^f \) are similar. We call this proposed model the BuresGAN. Namely, we use the Bures distance to match the covariance matrix of real and fake batches in feature space. The empirical effect of this term is to reduce mode collapse. The Bures distance admits both a feature space and kernel based expression. Contrary to other approaches \([4, 5]\), the architecture of the GAN is unchanged, only the objective is modified. Importantly, alt-BuresGAN, 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. Finally, 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.

### 1.1 Related works

The Bures distance is closely related to the Fréchet distance \([6]\) 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. It is important to note that the Bures distance is equivalent to the exact expression for the 2-Wasserstein distance between two elliptically contoured distributions with the same mean \([7, 8]\). 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 \([9, 10]\).

There exist numerous variants of GANs devoted to improving the GAN training. Here, we compare BuresGAN to the most closely related work. Methods like GDPP-GAN \([11]\) and VEEGAN \([5]\), 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. This reconstructor learns both to map all of the true data to the noise distribution and is an approximate inverse of the generator network, this change encourages the generator network to map from the noise distribution to the entirety of the true data distribution, thus diminishing mode collapse. Other methods adapt the objective function itself. The Unrolled-GAN \([12]\) 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. The latter improves the generator training process which helps to reduce mode collapse. Wasserstein GANs \([13]\) leverage the 1-Wasserstein distance to produce a value function which has better theoretical properties than the original one. In Wasserstein GAN-GP \([14]\), an alternative procedure to weight clipping is proposed which penalizes the norm of gradient of the critic with respect to its input, which gives more stable training. In MDGAN \([15]\), 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.

### 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 \( D(x) \) gives the probability that \( x \) comes from the empirical distribution, while the generator \( G \) maps a point \( z \) 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

\[
\min_{G} \max_{D} \mathbb{E}_{x \sim p_{d}}[\log D(x)] + \mathbb{E}_{z \sim p_{g}}[\log(1 - D(G(z)))],
\]

by alternating two phases of training. In \([1]\), 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 \( \mathcal{N}(0, I_{\ell}) \). It is common to define the discriminator loss by

\[
V_D = -\mathbb{E}_{x \sim p_{d}}[\log D(x)] - \mathbb{E}_{z \sim p_{g}}[\log(1 - D(G(z)))],
\]

which has to be minimized. In practice, it is proposed in \([3]\) to minimize generator loss

\[
V_G = -\mathbb{E}_{z \sim \mathcal{N}(0, I_{\ell})}[\log D(G(z))],
\]

rather than the second term of \([1]\), for an improved training efficiency.
2.1 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 [16, 11]. In order to compare covariance matrices, we propose to use the squared Bures distance between positive semi-definite $\ell \times \ell$ matrices [17], i.e.,

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

Being a Riemannian metric on the manifold of positive semi-definite matrices [13], the Bures metric is adequate to compare covariance matrices – it is also related to the 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 $\phi(x) \in \mathbb{R}^f$, defines a feature map, namely $D(x) = \sigma(w^T \phi(x))$, where $w$ is the weight vector of the last dense layer and $\sigma$ is the sigmoid function. We use the normalisation $\phi(x) = \phi(x)/\|\phi(x)\|_2$, after centering of $\phi(x)$. Then, we define a covariance matrix as follows: $C(p) = E_{x \sim \mathcal{P}}(\phi(x)\phi(x)^T)$. 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 $V_G + \lambda B(C_d, C_g)^2$. 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                                      | Algorithm 2: Alt-BuresGAN |
|-----------------------------------------------------------|--------------------------|
| Sample a real and fake batch ;                            | Sample a real and fake batch ; |
| Update $G$ by minimizing $V_G + \lambda B(\hat{C}_r, \hat{C}_g)^2$; | Update $G$ by minimizing $B(\hat{C}_r, \hat{C}_g)^2$; |
| Update $D$ by maximizing $-V_D$;                          | 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 [11], 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 $x_i^{(d)}$ and fake data $x_i^{(g)}$ with $i = 1, \ldots, b$ are sampled respectively from the empirical distribution and the mapping of the normal distribution $\mathcal{N}(0, I_f)$ 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 $\Phi_\alpha = [\phi(x_1^{(\alpha)}), \ldots, \phi(x_b^{(\alpha)})]^T \in \mathbb{R}^{b \times f}$ with $\alpha = d, g$. The covariance matrix of one batch in feature space is $\hat{C} = \tilde{\Phi}^T \tilde{\Phi}$, where $\tilde{\Phi}$ is the $\ell_2$-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

$$B(C_d, C_g)^2 = \text{Tr}(C_d + C_g - 2(C_g C_d)^{1/2}),$$

(4) which requires only one matrix square root. This identity can be obtained from Lemma[1] Note that an analogous result is proved in [19].

**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 $B = Y^T Y$, then $\text{Tr}((AB)^{1/2}) = \text{Tr}((YA Y^T)^{1/2})$.

**Proof.** (i) is a consequence of Corollay 2.3 in [20]. (ii) We now follow [19]. 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, $\text{Tr}((AB)^{1/2}) = \text{Tr}(P^{1/2})$. Then, $YA Y^T$ 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 $YA Y^TYP = YPD$. b) Similarly, suppose that we have the eigenvalue decomposition $YA Y^TQ = QA$. Then, we have $BAY^TQ = Y^T QA$ with $B = Y^T Y$. This means that the non-zero eigenvalues of $YA Y^T$ are also eigenvalues of $BA$. Since $A$ and $B$ are symmetric, this completes the proof.

\[1\] For simplicity, we omit the normalization by $\frac{1}{b-1}$ in front of the covariance matrix.
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: $K_d = \Phi_d \Phi_d^\top$, $K_g = \Phi_g \Phi_g^\top$, and $K_{d|g} = \Phi_d \Phi_g^\top$. Then, we have the kernel based expression

$$\mathcal{B}(\hat{C}_d, \hat{C}_g)^2 = \text{Tr} \left( K_d + K_g - 2 \left( K_{d|g} K_{d|g}^\top \right)^{1/2} \right),$$

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 $Y^\top Y X^\top X$ are diagonalizable with nonnegative eigenvalues and share the same non-zero eigenvalues.

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

Connection with Wasserstein GAN and integral probability metrics. The Bures distance is proportional to the 2-Wasserstein distance $\mathcal{W}_2$ between two elliptically contoured distributions, with the same mean \[7\]. For instance, in the case of multivariate normal distributions, we have

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

where the minimization is over the joint distributions $\pi$. More precisely, in this paper, we make the approximation that the implicit distribution of the real and generated data in the feature space $\mathbb{R}^f$ (associated to $\phi(x)$) are elliptically contoured with the same mean. Under different assumptions, the Generative Moment Matching Networks \[21\], [15\] 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:

$$\mathcal{W}_1(\alpha, \beta) = \sup_{f \in \text{Lip}} \int f(\alpha - \beta),$$

where $\alpha, \beta$ are signed measures. Generalizations of such integral formulae are called integral probability metrics (see for instance \[22\]). Here, $f$ is the discriminator, so that the maximization over Lipschitz functions plays the role of the maximization over discriminator parameters in the min-max game (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 BuresGAN, 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 (wrt the same measure) is equivalent to a Sobolev dual norm, which can be interpreted as an integral probability metric. Indeed, let the Sobolev semi-norm $\|f\|_{H^1} = \left( \int \|\nabla f(x)\|_2^2 \, dx \right)^{1/2}$. Then, its dual norm over signed measures is defined as $\|\nu\|_{H^{-1}} = \sup_{\|f\|_{H^1} \leq 1} \int f \, d\nu$. It is then shown in \[23\], \[8\] 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 (4) and (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 $1e-14$ is added for stability. The latent prior distribution is $N(0, \mathbb{I}_\ell)$ with $\ell = 100$ and the parameter in Algorithm 1 is always set to $\lambda = 1$.

3 Experiments

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 \[9\], WGAN-GP \[13\], MDGAN \[4\], UnrolledGAN \[12\], VEEGAN \[5\] and GDPP \[11\]. A low dimensional feature space ($f = 128$) is used for the synthetic data so that the feature space formula (4) is used, while the dual formula (5) is used for the image datasets (Stacked MNIST, CIFAR-10, CIFAR-100 and STL-100) for which the feature space is larger than the batch size. The architectures used for the image datasets are based on the DCGAN \[24\] model. All images are scaled in between -1 and 1 before running the algorithms. Additional information on the architectures and datasets is given in the Supplementary Material. The hyperparameters of other methods are typically chosen as suggested in the authors’ reference implementation. The number of unrolling steps in Unrolled GAN \[12\] is chosen to be 5. For the MDGAN \[4\], 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 \cite{14} uses $\lambda = 10.0$ and $n_{\text{critic}} = 5$. All models are trained using Adam \cite{25} 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 the Supplementary Material. All methods are implemented in Tensorflow 2 \cite{26} and the code will be available soon.

**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 \ref{tab:tab1}. All methods have the same architecture, with $\ell = 256$, following \cite{11}. 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 \ref{fig:fig1}. The proposed GAN captures all the modes early on in the training procedure, afterwards improving the quality. The training progress of the other GAN models listed in Table \ref{tab:tab1} is given in Supplementary Material.

![Figure 1](image.png)

**Table 1:** Experiments on the synthetic datasets. Average over 10 runs with std in parenthesis. All the models are trained for 25k iterations. The largest averaged values are in bold.

| GAN       | Nb modes | % in 3$\sigma$ | time (s) | WGAN-GP | Nb modes | % in 3$\sigma$ | time (s) |
|-----------|----------|----------------|----------|---------|----------|----------------|----------|
| GAN       | 22.9(4)  | 76(13)         | 32(0.2)  | 24.9(0.3) | 77(10)  | 225(1)       | 7.4(2)   |
| 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)      |          |
| GDPF      | 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)       |          |

**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 and two different architecture in Tables \ref{tab:tab2}.
All models are trained for 25k iterations. For the evaluation of the performance we follow [12] and use the following metrics: KL divergence for the generation quality and the number of captured modes to measure mode collapse. 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 (the exact architecture is given in the Supplementary Material), 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 both architectures, WGAN-GP has the best single run performance. WGAN-GP on average shows superior performance when using a discriminator with 2 convolutional layers 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: The generated samples from a trained Alt-BuresGAN. The proposed model is capable of generating high-quality images, at the same time preserving the diversity of the dataset.

CIFAR-10. Next, we evaluate the performance on the CIFAR-10 dataset. All models are trained for 100k iterations. In Table 3, the image quality is assessed thanks to the Inception Score, Fréchet Inception Distance (FID), Inference via Optimization (IvO) and Sliced Wasserstein Distance (SWD). The latter was also used in [11, 27] 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 [12] 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. 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 [13], 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. A richer variant of CIFAR-10 is the CIFAR-100 dataset. The latter consists of 100 different classes and is therefore more diverse. All models are trained for 100k iterations. The models are evaluated using the Inception Score, FID, IvO and SWD in Table 3. The metrics are calculated based on 10k generated images for all models.
100 with GPU Nvidia Quadro P4000. The timings for the other datasets are in the Supplementary Material. For the higher resolution STL-10 dataset. Only the proposed methods are capable of consistently generating high quality images as well as preventing mode collapse, for the same architecture.

Timings. The training times are indicated for the synthetic experiments which were performed on a single machine with GPU Nvidia Quadro P4000. The timings for the other datasets are in the Supplementary Material. For the

| Batch size | Nb modes | KL div. |
|------------|----------|---------|
|            | 64       | 128     | 256     |
| GAN        | 970.5(5.8)| 972.7(6.4)| 979.3(3.5)| 0.47(0.04) | 0.44(0.02) | 0.41(0.03) |
| WGAN-GP    | **996.7**(1.6)| **975.7**(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   | 689.1(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)| 685.4(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 2: 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 with a discriminator with 2 and 3 convolutional layers (on top and bottom respectively).

| Batch size | Nb modes | KL div. |
|------------|----------|---------|
|            | 64       | 128     | 256     |
| 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) | 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)| 891.7(299)| **975.5**(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 3: Experiments on CIFAR-10 and CIFAR-100. Average over 10 runs, std in parenthesis. 100k iterations for each. Inference via Optimization (IvO, lower is better), Inception Score (IS, higher is better), Fréchet Inception Distance (FID, lower is better) and Sliced Wasserstein Distance (SWD, lower is better). For readability, SWD score was multiplied by 100.

Compared to the 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). This is expected since the number of classes is multiplied by a factor 10 compared to CIFAR-10.

**STL-10.** The STL-10 dataset is inspired by the CIFAR-10 dataset, but it includes higher resolution images of size 96 × 96 × 3. This makes it a challenging benchmark for generating high quality samples as well as for the evaluation of mode collapse. The best performing models from previous experiments are trained for 150k iterations. Samples of 96 × 96 STL-10 images from a trained Alt-BuresGAN are given on Figure 3. The models are evaluated using the Inception Score, FID, IvO and SWD in Table 4. 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 the proposed methods are capable of consistently generating high quality images as well as preventing mode collapse, for the same architecture.

**Timings.** The training times are indicated for the synthetic experiments which were performed on a single machine with GPU Nvidia Quadro P4000. The timings for the other datasets are in the Supplementary Material. For the
Figure 3: Samples of $96 \times 96$ STL-10 images from a trained Alt-BuresGAN for 150k iterations.

| Method      | IvO    | Inception Score | 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: Experiments on STL-10. Average over 5 runs, std in parenthesis. 150k iterations for each. Inference via Optimization (IvO, lower is better), Inception Score (higher is better), Fréchet Inception Distance (FID, lower is better) and Sliced Wasserstein Distance (SWD, lower is better). For readability, SWD score was multiplied by 100.

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 (cfr. Supplementary Material).

4 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 hyperparameters.
Acknowledgements

EU: The research leading to these results has received funding from the European Research Council under the European Union’s Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960); (grant agreement No 885682); (EU H2020-SC1-2016-2017 Grant Agreement No.727721: MIDAS). This paper reflects only the authors’ views and the Union is not liable for any use that may be made of the contained information. Research Council KUL: Optimization frameworks for deep kernel machines C14/18/068 ; Research Fund (projects C16/15/059, C32/16/013, C24/18/022), Industrial Research Fund (Fellowship 13-0260) and several Leuven Research and Development bilateral industrial projects, Flemish Government Agencies: FWO (EOS Project no 30468160 (SeLMA), SBO project I013218N, PhD Grants (SB/ISA1319N, SB/151622)). Flemish Government: FWO project GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), EOS Project 30468160 (SeLMA), SBO project I013218N, PhD/Postdoc grant (SB/ISA1319N, SB/151622)), Impulsfonds AI: VR 2019 2203 DOC.0318/1QUATER Kenniscentrum Data en Maatschappij. This research received funding from the Flemish Government (AI Research Program), VLAIO (City of Things (COT.2018.018), PhD grants: Baekeland (HBC.20192204) and Innovation mandate (HBC.2019.2209), Industrial Projects (HBC.2018.0405). Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms). The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government – department EWI.

References

[1] Diederik P Kingma and Max Welling. Auto-encoding Variational Bayes. In Proceedings of the International Conference on Learning Representations (ICLR), 2014.
[2] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. Proceedings of the 32th International Conference on Machine Learning (ICML), 2015.
[3] 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.
[4] 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.
[5] 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.
[6] D.C Dowson and B.V Landau. The Fréchet distance between multivariate normal distributions. Journal of Multivariate Analysis, 12(3):450 – 455, 1982.
[7] Matthias Gelbrich. On a Formula for the L2 Wasserstein Metric between Measures on Euclidean and Hilbert spaces. Mathematische Nachrichten, 147(1):185–203, 1990.
[8] Gabriel Peyré, Marco Cuturi, et al. Computational Optimal Transport. Foundations and Trends® in Machine Learning, 11(5-6):355–607, 2019.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.
[14] 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.
[15] 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.
[16] 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.
[17] Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the Bures–Wasserstein distance between positive definite matrices. Expositiones Mathematicae, 37(2):165 – 191, 2019.
[18] 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.
[19] Jung Hun Oh, Maryam Pouryahya, Aditi Iyer, Aditya P. Apte, Allen Tannenbaum, and Joseph O. Deasy. Kernel Wasserstein Distance, arxiv preprint arxiv:1905.09314, 2019.

[20] 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.

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

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

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

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations (ICLR)*, 2016.

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

[26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, and Craig Citro et al. TensorFlow: Large-scale machine learning on heterogeneous systems , arxiv preprint arxiv:1603.04467, 2015.

[27] 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.
A Details of the theoretical results

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

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

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

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

since $\text{Tr}(U^T B^{1/2} A^{1/2}) = \text{Tr}(A^{1/2} B^{1/2} U)$. 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 $f^*(\Sigma) = \|\Sigma\|_*$, where the nuclear norm is $\|\Sigma\|_* = \text{Tr}(\sqrt{\Sigma^T \Sigma})$ and $U, M \in \mathbb{R}^{\ell \times \ell}$.

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

$$f^*(\Sigma) = \sup_{U \in \mathbb{R}^{\ell \times \ell}} \text{Tr}(U^T \Sigma) - f(U) = \max_{U \in O(\ell)} \text{Tr}(U^T \Sigma).$$

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

$$\max_{U \in O(\ell)} \text{Tr}(U^T \Sigma) = \max_{U \in O(\ell)} \text{Tr}(W \Sigma V^T U) = \max_{U' \in O(\ell)} \text{Tr}(\Sigma U'),$$

where we renamed $U' = V^T U W$. Next, we remark that $\text{Tr}(\Sigma U') = \text{Tr}(\Sigma \text{diag}(U'))$. Since by construction, $\Sigma$ is diagonal with non negative entries the maximum is attained at $U' = 1$. Then, the optimal objective is $\text{Tr}(\Sigma) = \text{Tr}(\sqrt{\Sigma^T \Sigma})$. \hfill \Box

By taking $M = A^{1/2} B^{1/2}$ we obtain (6). Notice that the role of $A$ and $B$ can be exchanged in (6) since $U$ is orthogonal.

B Training Details

B.1 Synthetic Architectures

Following the recommendation in the original work [5], the same fully-connected architecture is used for the VEEGAN reconstructor in all experiments.

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

Table 5: The generator and discriminator architectures for the synthetic examples.

B.2 Stacked MNIST Architectures
Table 6: 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.

Table 7: The generator and discriminator architectures for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not.

Table 8: The MDGAN encoder model architecture for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not.
### B.3 CIFAR-10 and 100 Architectures

| Layer   | Output | Activation | BN | Layer   | Output | Activation | BN |
|---------|--------|------------|----|---------|--------|------------|----|
| Input   | 100    | -          | -  | Input   | 32, 32, 3| -          | -  |
| Dense   | 16384  | ReLU       | Yes| Conv    | 16, 16, 64| Leaky ReLU | No |
| Reshape | 8, 8, 256| -        | -  | Conv    | 8, 8, 128| Leaky ReLU | Yes|
| Conv'   | 8, 8, 128| ReLU     | Yes| Conv    | 4, 4, 256| Leaky ReLU | Yes|
| Conv'   | 16, 16, 64| ReLU     | Yes| Flatten | -        | -          | -  |
| Conv'   | 32, 32, 3| ReLU      | Yes| Dense   | 1        | -          | -  |

Table 9: 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 | Layer   | Output | Activation | BN |
|---------|--------|------------|----|---------|--------|------------|----|
| Input   | 32, 32, 3| -          | -  | Conv    | 16, 16, 3| ReLU       | Yes|
| Conv    | 16, 16, 3| ReLU       | Yes| Conv    | 8, 8, 64 | ReLU       | Yes|
| Conv    | 8, 8, 128| ReLU       | Yes| Conv    | 8, 8, 128| ReLU       | Yes|
| Flatten | -      | -          | -  | Dense   | 100     | -          | -  |

Table 10: 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.

### B.4 STL-10 Architectures

| Layer   | Output | Activation | BN | Layer   | Output | Activation | BN |
|---------|--------|------------|----|---------|--------|------------|----|
| Input   | 100    | -          | -  | Input   | 96, 96, 3| -          | -  |
| Dense   | 36864  | ReLU       | Yes| Conv    | 48, 48, 64| Leaky ReLU | No |
| Reshape | 12, 12, 256| -       | -  | Conv    | 24, 24, 128| Leaky ReLU | Yes|
| Conv'   | 12, 12, 256| ReLU     | Yes| Conv    | 12, 12, 256| Leaky ReLU | Yes|
| Conv'   | 24, 24, 128| ReLU      | Yes| Conv    | 6, 6, 512 | Leaky ReLU | Yes|
| Conv'   | 48, 48, 64| ReLU      | Yes| Flatten | -        | -          | -  |
| Conv'   | 96, 96, 3| ReLU      | Yes| Dense   | 1        | -          | -  |

Table 11: The generator and discriminator architectures for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.
Table 12: 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 Additional Experiments

The timings per iteration for the experiments presented in the paper are listed in Table 13. Times are given for all the methods considered, although some methods 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.

Table 13: Average time per iteration in seconds. Averaged over 5 runs, with std in parenthesis. The batch size is 256.

| Method      | stacked MNIST | CIFAR-10 | CIFAR-100 | STL-10 |
|-------------|---------------|----------|-----------|--------|
| GAN         | 0.54(0.0005)  | 0.65(0.02)| 0.64(0.0008)| 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)| 18.03(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 14: Inception Score for best trained models on CIFAR-10, CIFAR-100 and STL-10.

| Method      | 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   |

The inception scores for the best trained models are listed in Table 14. 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.

D Additional Figures
| GAN      | Step 1k | Step 2k | Step 4k | Step 6k | Step 8k | Step 25k |
|----------|---------|---------|---------|---------|---------|----------|
| GAN      | ![image](GAN_step1k.png) | ![image](GAN_step2k.png) | ![image](GAN_step4k.png) | ![image](GAN_step6k.png) | ![image](GAN_step8k.png) | ![image](GAN_step25k.png) |
| WGAN-GP  | ![image](WGAN-GP_step1k.png) | ![image](WGAN-GP_step2k.png) | ![image](WGAN-GP_step4k.png) | ![image](WGAN-GP_step6k.png) | ![image](WGAN-GP_step8k.png) | ![image](WGAN-GP_step25k.png) |
| MDGAN-v1 | ![image](MDGAN-v1_step1k.png) | ![image](MDGAN-v1_step2k.png) | ![image](MDGAN-v1_step4k.png) | ![image](MDGAN-v1_step6k.png) | ![image](MDGAN-v1_step8k.png) | ![image](MDGAN-v1_step25k.png) |
| MDGAN-v2 | ![image](MDGAN-v2_step1k.png) | ![image](MDGAN-v2_step2k.png) | ![image](MDGAN-v2_step4k.png) | ![image](MDGAN-v2_step6k.png) | ![image](MDGAN-v2_step8k.png) | ![image](MDGAN-v2_step25k.png) |
| UnrolledGAN | ![image](UnrolledGAN_step1k.png) | ![image](UnrolledGAN_step2k.png) | ![image](UnrolledGAN_step4k.png) | ![image](UnrolledGAN_step6k.png) | ![image](UnrolledGAN_step8k.png) | ![image](UnrolledGAN_step25k.png) |
| VEEGAN   | ![image](VEEGAN_step1k.png) | ![image](VEEGAN_step2k.png) | ![image](VEEGAN_step4k.png) | ![image](VEEGAN_step6k.png) | ![image](VEEGAN_step8k.png) | ![image](VEEGAN_step25k.png) |
| GDPP     | ![image](GDPP_step1k.png) | ![image](GDPP_step2k.png) | ![image](GDPP_step4k.png) | ![image](GDPP_step6k.png) | ![image](GDPP_step8k.png) | ![image](GDPP_step25k.png) |
| BuresGAN | ![image](BuresGAN_step1k.png) | ![image](BuresGAN_step2k.png) | ![image](BuresGAN_step4k.png) | ![image](BuresGAN_step6k.png) | ![image](BuresGAN_step8k.png) | ![image](BuresGAN_step25k.png) |
| Alt-BuresGAN | ![image](Alt-BuresGAN_step1k.png) | ![image](Alt-BuresGAN_step2k.png) | ![image](Alt-BuresGAN_step4k.png) | ![image](Alt-BuresGAN_step6k.png) | ![image](Alt-BuresGAN_step8k.png) | ![image](Alt-BuresGAN_step25k.png) |

Figure 4: The progress of different GANs on the synthetic ring example. Each column shows 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.
|              | GAN                                      | WGAN-GP                                 | MDGAN-v1                                  |
|--------------|------------------------------------------|-----------------------------------------|-------------------------------------------|
| MDGAN-v2     |                                          |                                        |                                           |
| UnrolledGAN  |                                          |                                        |                                           |
| VEEGAN       |                                          |                                        |                                           |
| GDPP         |                                          |                                        |                                           |
| BuresGAN     |                                          |                                        |                                           |
| Alt-BuresGAN |                                          |                                        |                                           |

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. Each column shows 64 samples from the trained generator.
Figure 8: Generated images for CIFAR-100. Each column shows 64 samples from the trained generator.
Figure 9: Generated images for STL-10. Each column shows 64 samples from the trained generator.