Latent reweighting, an almost free improvement for GANs

Thibaut Issenhuth[1,2], Ugo Tanielian[1], David Picard [2], Jérémie Mary[1]
[1] Criteo AI Lab, Paris, France.
[2] LIGM, Ecole des Ponts, Marne-la-vallée, France.
{thibaut.issenhuth,ugo.tanielian,jeremie.mary}@criteo.com,david.picard@enpc.fr

Abstract

Standard formulations of GANs, where a continuous function deforms a connected latent space, have been shown to be misspecified when fitting different classes of images. In particular, the generator will necessarily sample some low-quality images in between the classes. Rather than modifying the architecture, a line of works aims at improving the sampling quality from pre-trained generators at the expense of increased computational cost. Building on this, we introduce an additional network to predict latent importance weights and two associated sampling methods to avoid the poorest samples. This idea has several advantages: 1) it provides a way to inject disconnectedness into any GAN architecture, 2) since the rejection happens in the latent space, it avoids going through both the generator and the discriminator, saving computation time, 3) this importance weights formulation provides a principled way to reduce the Wasserstein's distance to the target distribution. We demonstrate the effectiveness of our method on several datasets, both synthetic and high-dimensional.

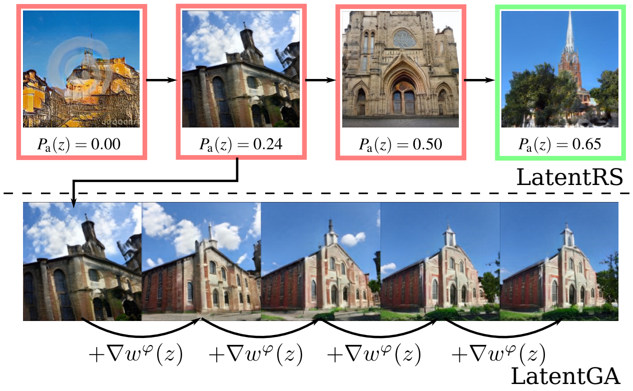


Figure 1: Overview of the proposed method. GANs tend to produce poor images for unlucky draws of the latent variable (top row, left). We introduce importance weights $w^{\varphi}(z)$ in the latent space that allow us to use rejection sampling and accept a given latent variable z with probability $P_a(z) \propto w^{\varphi}(z)$ (LatentRS, top row), or to perform a simple gradient ascent over the importance weight (LatentGA, bottom row), leading to better images. Both strategies can be combined for improved image quality. Images generated with StyleGAN2 trained on LSUN Church.

1. Introduction

GANs [10] are an effective way to learn complex and high-dimensional distributions, leading to state-of-the-art models for image synthesis in both unconditional [18] and conditional settings [6]. However, it is well-known that a single generator with an unimodal latent variable cannot recover a distribution composed of disconnected sub-manifolds [20]. This leads to a common problem for practitioners: the existence of very low-quality samples when covering different modes. This is formalized by [33] which refers to this area as the no GAN's land and provides impossibility theorems on the learning of disconnected manifolds with standard formulations of GANs. Fitting a disconnected target distribution requires an additional mechanism inserting disconnectedness in the modeled distribution. A first solution is to add some expressivity to the model: [20] propose to train a mixture of generators, while [13] make use of a multi-modal latent distribution.

A second line of research relies heavily on a variety of Monte-Carlo algorithms, such as Rejection Sampling [3] or Metropolis-Hastings [34]. Monte-Carlo methods aim at sampling from a target distribution, while having only access to samples generated from a proposal distribution. Using the previously learned generative distribution as a proposal distribution, this idea was successfully applied to GANs. However, one of the main drawbacks is that Monte-Carlo algorithms only guarantee to sample from the target distribution under strong assumptions. First, we need access to the density ratios between the proposal and target distributions or equivalently to a perfect discriminator [3]. Second, these methods are efficient only if the support of the proposal distribution fully covers the one of the target distribution. This is unlikely to be the case when dealing with high-dimensional datasets [1].

To tackle this issue, we propose a novel method aiming at reducing the Wasserstein distance between the previously trained generative model and the target distribution. This is done via the adversarial training of a third network that learns importance weights in the latent space. Note that this network does not aim at increasing the support of the proposal distribution but at re-weighting the latent distribution, under a Wasserstein criterion. Thus, these importance weights define a new distribution in the latent space, from which we propose to sample with two complementary methods: latent rejection sampling (latentRS) and latent gradient ascent (latentGA). To better understand our approach, we illustrate its efficiency with simple examples. On the top of the Figure 1, we show samples coming from a pre-trained StyleGAN2 [18] and their respective acceptance probability (latentRS). At the bottom, we exhibit a sequence of generated images while following a gradient ascent on the learned importance weights (latentGA).

Our contributions are the following:

- We propose a novel approach that trains a neural network to directly modify the latent space of a GAN. This provides a principled way to reduce the Wasserstein distance to the target distribution.
- We show how to sample from this new generative model with different methods: latent Rejection Sampling (latentRS), latent Gradient Ascent (latentGA), and latentRS+GA, a method that leverages the complementarity between the two previous solutions.
- We run a large empirical comparison between our proposed methods and previous approaches on a variety of datasets and distributions. We empirically show that all of our proposed solutions significantly reduce the computational cost of inference. More interestingly, our solutions propose a wide span of performances ranging from latentRS, optimizing speed, that matches state-of-the-art almost for free (computational cost divided by 15) and latentRS+GA (computational cost divided by 3) that outperforms previous approaches.

Notation. Before moving to the related work section, we shortly present the notation needed in the paper. The goal of the generator is to generate data points that are "similar" to samples collected from some target probability measure $\mu_\star$. The measure $\mu_\star$ is defined on a potentially high-dimensional space $\mathbb{R}^D$, equipped with the euclidean norm $\|\cdot\|$. We call μ_n the empirical measure. To approach $\mu_\star$, we use a parametric family of generative distribution, where each distribution is the push-forward measure of a latent distribution Z and a continuous function modeled by a neural network. In most applications, the random variable Z defined on a low-dimensional space $\mathbb{R}^d$ is either a multivariate Gaussian distribution or uniform distribution. The generator is a parameterized class of functions from $\mathbb{R}^d$ to $\mathbb{R}^D$, say $\mathscr{G} = \{G_\theta : \theta \in \Theta\}$, where $\Theta \subseteq \mathbb{R}^p$ is the set of parameters describing the model. Each function G_θ takes input from Z and outputs "fake" observations with distribution $\mu_\theta = G_\theta \sharp Z$. On the other hand, the discriminator is described by a family of functions from $\mathbb{R}^D$ to $\mathbb{R}$, say $\mathscr{D} = \{D_\alpha : \alpha \in \Lambda\}$, $\Lambda \subseteq \mathbb{R}^Q$. Finally, for any given distribution μ, we note S_μ its support.

2. Related Work

[10] already stated that when training vanilla GANs, the generator could ignore modes of the target distribution: this is called mode collapse. A significant step towards understanding this phenomenon was made by [1] who explained that the standard formulation of GANs leads to vanishing or unstable gradients. The authors proposed the Wasserstein GANs (WGANs) architecture [2] where, in particular, discriminative functions are restricted to the class of 1-Lipschitz functions. WGANs aim at solving the following:

$$\sup_{\alpha \in A} \inf_{\theta \in \Theta} \mathbb{E}_{x \sim \mu_\star} D_\alpha(x) - \mathbb{E}_{z \sim \gamma} D_\alpha(G_\theta(z)) \tag{1}$$

2.1. Learning disconnected manifolds with GANs: training and evaluation

The broader drawback of standard GANs is that, since any modeled distribution is the push-forward of a unimodal distribution by a continuous transformation, it has a connected support. This means that when the generator covers multiple disconnected modes of the target distribution, it necessarily generates samples out of the real data manifold [20]. Consequently, any thorough evaluation of GANs should assess simultaneously both the quality and the variety of the generated samples. To solve this issue, [29] and [21] propose a Precision/Recall metric that aims at measuring both the *mode dropping* and the *mode inventing*. The precision refers to the portion of generated points that belongs to the target manifold, while the recall measures how much of the target distribution can be reconstructed by the model distribution.

Building on this metric, [33] highlighted the trade-off property of GANs deriving upper-bounds on the precision of standard GANs. To solve this problem, a common direction of research consists in over-parameterizing the generative model. [20] enforces diversity by using a mixture of generators, while [13] suggests that a mixture of Gaussians in the latent space is efficient to learn diverse and limited data. Similarly, [4] propose importance weights that aim at robustifying the training of GANs and make it less sensitive to the target distribution's outliers.

2.2. Improving the quality of GANs post-training

Another line of research consists in improving the sampling quality of pre-trained GANs. [33] proposed a heuristic to insert disconnectedness and remove the samples mapped out of the true manifold. [32] designed Discriminator Optimal Transport (DOT), a gradient ascent driven by a Wasserstein discriminator to improve every single sample. Similarly, [7] follow a discriminator-driven Langevin dynamic.

Another well-studied possibility would be to use Monte-Carlo (MC) methods [27]. Following this path, [3] were the first to use a rejection sampling method to improve the quality of the proposal distribution μ_θ. The authors use the fact that the optimal vanilla discriminator trained with binary cross-entropy is equal to $\mu_\star/(\mu_\star+\mu_\theta)$. Thus, a parametric discriminator $D_\alpha : \mathbb{R}^D \to [0,1]$ can be used to approximate the density ratios r_α as follows:

$$r_\alpha(x) := \frac{\mu_\star(x)}{\mu_\theta(x)} = \frac{D_\alpha(x)}{1-D_\alpha(x)}. \tag{2}$$

This density ratio can then be plugged in the Rejection Sampling (RS) algorithm. Doing so, it can be shown that sampling from μ_θ and accepting samples probabilistically is equivalent to sample from the target distribution $\mu_\star$. The acceptance probability of a given sample x is $\mathbb{P}_a(x) = \frac{r_\alpha(x)}{k}$. This is valid as long as there is a constant $k \in \mathbb{R}^+$ such that $\mu_\star(x) \leq k\mu_\theta(x)$ for all x.

[34] use similar density ratios and derive MH-GAN, by using the independent Metropolis-Hasting algorithm [15]. Finally, [11] use these density ratios r_α as importance weights and perform discrete sampling relying on the Sampling-Importance-Resampling (SIR) algorithm [28]. Given $X_1,\dots,X_n \sim \mu_\theta^n$, we have:

$$\mu_{\theta,\alpha}^{\text{SIR}}(X_i) = \frac{r_\alpha(X_i)}{\sum\limits_{j=1}^{n} r_\alpha(X_j)}. \tag{3}$$

Note that these algorithms all rely on similar density ratios and differ by the acceptance-rejection scheme chosen. Interestingly, in RS, the acceptance rate is not controlled, but we are guaranteed to sample from $\mu_\star$. Conversely, with SIR and MH, the acceptance rate is a chosen parameter, but we are sampling from an approximation of the target distribution.

2.3. Drawbacks of density-ratio-based methods

Even though these methods have the advantage of being straightforward, they suffer from one main drawback. In practice, because both the target and the proposal manifold do not have full dimension in $\mathbb{R}^D$ [9], [1, Lemma 3] show that it is highly likely that $\mu_\theta(S_{\mu_\theta} \bigcap S_{\mu_\star}) = 0$ and $\mu_\star(S_{\mu_\theta} \bigcap S_{\mu_\star}) = 0$. Consequently, when dealing with high-dimensional datasets, the proposal distribution μ_θ and the target distribution $\mu_\star$ might intersect on a null set. Thus, one would have $r_\alpha(x) = 0$ almost everywhere on S_{μ_θ}. In this setting, the assumptions of MC methods are broken, and these algorithms will not allow sampling from $\mu_\star$.

In order to correct this drawback, our method proposes to avoid the computation of density ratios from a classifier and to directly learn how to re-weight the proposal distribution. Our proposed scheme aims at minimizing the Wasserstein distance to the empirical measure while controlling the range of these importance weights.

3. Adversarial Learning of Latent Importance weights

Similar to previous works, our method aims at improving the performance of a generative model, post-training. We assume the existence of a WGAN model (G_θ, D_α) pre-trained using equation 1. The pushforward generative distribution μ_θ is assumed to be an imperfect approximation of the target distribution. The goal is now to learn how to redistribute the mass of the modeled distribution so that it best fits the target distribution.

3.1. Definition of the method

To improve the sampling quality of our pre-trained GANs, we propose to learn an importance weight function that directly learns how to avoid low-quality images and focus on very realistic ones. More formally, we over-parameterize the class of generative distributions and define a parametric class $\Omega = \{w^\varphi, \varphi \in \Phi\}$ of importance weight functions. Each function w^φ associates importance weights to latent space variables and is defined from $\mathbb{R}^d$ to $\mathbb{R}^+$. For a given latent space distribution γ and a network w^φ, a new measure γ^φ is defined on $\mathbb{R}^d$:

$$\text{for all } z \in \mathbb{R}^d, \, \mathrm{d}\gamma^\varphi(\mathrm{z}) = \mathrm{w}^\varphi(\mathrm{z})\mathrm{d}\gamma(\mathrm{z}) \tag{4}$$

Using this formulation, we can prove the following lemma:

Lemma 1 *Assume that $\mathbb{E}_\gamma \, w^\varphi = 1$, then the measure γ^φ is a probability distribution defined on $\mathbb{R}^d$.*

Consequently, we now propose a new modeled generative distribution μ_θ^φ, the pushforward distribution $\mu_\theta^\varphi = G_\theta \sharp \gamma^\varphi$. The objective is to find the optimal importance weights w^φ that minimizes the Wasserstein distance between the true distribution $\mu_\star$ and the new class of generative distributions. The proposed method can thus be seen as minimizing the Wasserstein distance to the target distribution, over an increased class of generative distributions. Denoting by Lip_1 the set of 1-Lipschitz real-valued functions on $\mathbb{R}^D$, i.e.,

$$\text{Lip}_1 = \left\{ f : \mathbb{R}^D \to \mathbb{R} : \frac{|f(x)-f(y)|}{\|x-y\|} \leqslant 1, \ (x \neq y) \in (\mathbb{R}^D)^2 \right\},$$

we want, given a pre-trained model μ_θ, to solve:

$$\begin{aligned}\arg\min_{\varphi\in\Phi} W(\mu_\star,\mu_\theta^\varphi) &= \arg\min_{w^\varphi\in\Omega}\ \sup_{D\in\text{Lip}_1}\ \mathbb{E}_{\mu_\star}D - \mathbb{E}_{\mu_\theta^\varphi}D \\ &= \arg\min_{w^\varphi\in\Omega}\ \sup_{D\in\text{Lip}_1}\ \mathbb{E}_{\mu_\star}D - \mathbb{E}_{\mu_\theta}w^\varphi D\end{aligned}$$

The network w^φ, parameterized using a feed-forward neural network, thus learns how to redistribute the mass of μ_θ such that μ_θ^φ is closer to $\mu_\star$ in terms of Wasserstein distance. Similarly to the WGANs training, the discriminator D_α approximates the Wasserstein distance. D_α and w^φ are trained adversarially, whilst keeping the weights of G_θ frozen, using the following optimization scheme:

$$\inf_{\varphi\in\Phi}\ \sup_{\alpha\in\Lambda}\ \mathbb{E}_{x\sim\mu_\star}D_\alpha(x) - \mathbb{E}_{z\sim Z}\ w^\varphi(z)\times D_\alpha(G_\theta(z)) \qquad (5)$$

Note that our formulation can also be plugged on top of any objective function used for GANs.

3.2. Optimization procedure

However, as in the field of counterfactual estimation, a naive optimization of importance weights by gradient descent can lead to trivial solutions.

1. First, if for example, the Wasserstein critic D_α outputs negative values for any generated sample, the network w^φ could simply learn to avoid the dataset and output 0 everywhere [30].

2. Second, another problem comes from the fact that equation 5 can be minimized not only by putting large importance weights $w^\varphi(z)$ on the examples with high likelihoods $D_\alpha(G(z))$ but also by maximizing the sum of the weights: this is the propensity overfitting [31].

3. For the objective defined in equation 5 to be a valid Wasserstein distance minimization scheme, the measure μ_θ^φ must be a probability distribution, *i.e.* $\mathbb{E}_\gamma w^\varphi = 1$.

To tackle this, we first add a penalty term in the loss to enforce the expectation of the importance weights to be close to 1. This is similar to the self-normalization proposed by [31]. However, one still has to cope with the setting where the distribution γ^φ collapses to discrete data points:

Theorem 1 *Given a pre-trained generative distribution μ_θ absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}^D$. Let Φ be the non-parametric class of continuous functions satisfying $\mathbb{E}_\gamma w^\varphi = 1$. We have that:*

$$W(\mu_n, \frac{1}{n}\sum_{i=1}^{n}\delta(\tilde{X}_i)) \leqslant \inf_{\varphi\in\Phi} W(\mu_n,\mu_\theta^\varphi)$$

where δ refers to the Dirac probability distribution and $\tilde{X}_i = \arg\min_{x\in S_{\mu_\theta}} \|x - X_i\|$.

For clarity, the proof is delayed in Appendix. Intuitively, this theorem shows that the best way to approximate the empirical measure μ_n would be by considering a mixture of Diracs with each mode being the projection of a training data point on the support of the learned manifold S_{μ_θ}. The network w^φ could thus be tempted to approximate this mixture of Diracs defined in Theorem 1 and collapse on some specific latent data points. This could lead to an increased time complexity at inference (see [3, Section 3]). More importantly, this would mean *a mode collapse* and a lack of diversity in the generated samples.

To avoid such cases where small areas of z have really high $w^\varphi(z)$ values (mode collapse), we enforce a soft-clipping on the weights [5, 11]. Note that this constraint on $w^\varphi(z)$ could also be implemented with a bounded activation function on the final layer, such as a re-scaled sigmoid or tanh activation. Finally, we get the following objective function for the network w^φ:

$$\begin{aligned}\sup_{\varphi\in\Phi}\ \mathbb{E}_{z\sim Z}\ &\underbrace{w^\varphi(z)\big(D_\alpha(G_\theta(z)) - \Delta\big)}_{\text{discriminator reward}} - \lambda_1\underbrace{\big(\mathbb{E}_{z\sim Z}w^\varphi(z) - 1\big)^2}_{\text{self-normalization}} \\ &- \lambda_2\,\underbrace{\mathbb{E}_{z\sim Z}\max\big(0,(w^\varphi(z) - m)\big)^2}_{\text{soft-clipping}},\end{aligned} \qquad (6)$$

where $\Delta = \min_{z\sim Z}\ D_\alpha(G(z))$. λ_1, λ_2, and m are hyper-parameters (values displayed in Appendix). For more details, we refer the reader to Algorithm 1.

Algorithm 1: Adversarial learning of w^φ

```
Require: Data $\mu_n$, Prior $Z$, Gen. $G_\theta$, Disc. $D_\alpha$,
  number of $D_\alpha$ updates $n_d$, soft-clipping param. $m$,
  regularization weights $\lambda_1$ and $\lambda_2$, batch size $b$;
while $\varphi$ has not converged do
    for $i = 0, ..., n_d$ do
        Sample real data $\{x_i\}_{i=1}^{b} \sim \mu_n$;
        Sample latent vectors $\{z_i\}_{i=1}^{b} \sim Z$ ;
        EMD $\leftarrow \frac{1}{b}\sum_{i=1}^{b} D_\alpha(x_i) - w^\varphi(z_i)D_\alpha(G_\theta(z_i))$;
        GP $\leftarrow$ Gradient-Penalty$(D_\alpha, x, G_\theta(z))$;
        grad$_\alpha \leftarrow \nabla_\alpha(-\text{EMD} + \text{GP})$ ;
        Update $\alpha$ with grad$_\alpha$;
    end
    Sample $\{z_i\}_{i=1}^{b} \sim Z$ ;
    $\Delta \leftarrow \min_i[D_\alpha(G_\theta(z_i))]$ ;
    EMD $\leftarrow \frac{1}{b}\sum_{i=1}^{b} w(z_i)[D_\alpha(G_\theta(z_i)) - \Delta]$;
    $R_{norm} \leftarrow ([\frac{1}{b}\sum_{i=1}^{b} w(z_i)] - 1)^2$ ;
    $R_{clip} \leftarrow \frac{1}{b}\sum_{i=1}^{b}\max(0, w^\varphi(z_i) - m)^2$ ;
    grad$_\varphi \leftarrow \nabla_\varphi(\text{EMD} + \lambda_1 R_{norm} + \lambda_2 R_{clip})$ ;
    Update $\varphi$ with grad$_\varphi$;
end
```

3.3. Sampling from the latent importance weights

Given a pre-trained generator G_θ and an importance network w^φ, we now present the three proposed sampling algorithms associated with our model:

1) Latent Rejection Sampling (latentRS, Algorithm 2). The first proposed method aims at sampling from the newly learned latent distribution γ^φ defined in equation 4. Since the learned importance weights are capped by m defined in equation 6, this setting fits in the Rejection Sampling (RS) algorithm [27]. Any sample $z \sim \gamma$ is now accepted with probability $\mathbb{P}_\mathrm{a}(z) = w^\varphi(z)/m$. Interestingly, by actively capping the importance weights as it is done in counterfactual estimation [5, 8], one controls the acceptance rates $\mathbb{P}_a(z)$ of the rejection sampling algorithm:

$$\mathbb{E}_\gamma\, \mathbb{P}_\mathrm{a}(z) = \int_{\mathbb{R}^d} \frac{w^\varphi(z)}{m} \mathrm{d}\gamma(\mathrm{z}) = \frac{1}{\mathrm{m}}.$$

```
Algorithm 2: LatentRS
  Requires: Prior Z, Gen. $G_\theta$, Importance weight
    network $w^\varphi$, maximum importance weight $m$;
  while True do
      Sample $z \sim Z$ ;
      Sample $\alpha \sim$ Uniform$[0,1]$ ;
      if $\frac{w^\varphi(z)}{m} \geq \alpha$ then
          break;
      end
  end
  $x \leftarrow G_\theta(z)$;
  Result: Selected point x
```

2) Latent Gradient Ascent (latentGA). Inspired from [32, Algorithm 2], we propose a second method, latentGA, where we perform gradient ascent in the latent space (see the algorithm in Appendix). For any given sample in the latent space, we follow the path maximizing the learned importance weights. This method is denoted latentGA. Note that the learning rate and the number of updates used for this method are hyper-parameters that need to be tuned.

3) Combining latentRS with Gradient Ascent (latent RS+GA, see Appendix). Finally, we propose to combine sequentially both methods. In a first step, we avoid low-quality samples with latentRS. Then, we use latentGA to further improve the remaining generated samples. See algorithm in Appendix.

3.4. Advantages of the proposed approach

We now discuss two advantages of our method compared to previous density-ratio-based Monte-Carlo methods.

Computational cost. By using sampling algorithms in the latent space, we avoid going through both the generator and the discriminator, leading to a significant computational speed-up. This is of particular interest when dealing with high-dimensional spaces, since we do not need to pass through deep CNNs generator and discriminator [18]. In the next experimental section, we observe a computational cost decreased by a factor of 10.

Monte-Carlo methods do not properly work when the support S_{μ_θ} does not fully cover the support $S_{\mu_\star}$. To better illustrate this claim, we consider a simple 2D motivational example where the real data lies on four disconnected manifolds. We start with a proposal distribution (in blue) that does not fully recover the target distribution (Figure 2a). In this setting, we see in Figure 2b that the discriminator's density-ratio-based methods [3] avoids half of the proposal distribution, while our proposed method learns a very different re-weighting (see Figure 2c).

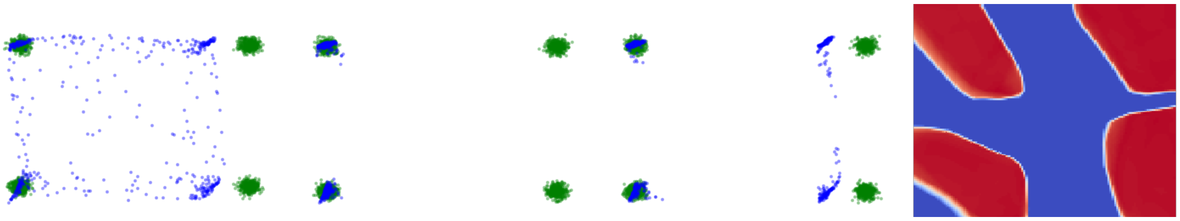
This illustration is important since [1, Theorem 2.2] have shown that in high-dimension the intersection $S_{\mu_\star} \bigcap S_{\mu_\theta}$ is likely to be a negligible set under μ_θ. Knowing that S_{μ_θ} does not fully recover $S_{\mu_\star}$, there is thus no theoretical guarantee that using a sampling algorithm will improve the estimation of $\mu_\star$. On the opposite, our method looks for the optimal re-weighting of μ_θ under a well-defined criterion: the Wasserstein distance. This results in a better fit of the real data distribution (see next section).

4. Experiments

In this section, we illustrate the efficiency of the proposed methods, latentRS, latentGA, and latentRS+GA on both synthetic and natural image datasets. On image generation tasks, we empirically stress that latentRS slightly surpasses density-ratio-based methods with respect to the Earth Mover's distance while reducing the time complexity by a factor of around 10. The use of latentGA also gives interesting experimental visualizations and improves image quality. More importantly, when combined, we show that latenRS+GA surpasses the concurrent methods, while still being less computationally intensive. Finally, we show results with different models such as Progressive GAN [17] and StyleGAN2 [19].

4.1. Evaluation metrics

To measure the performances of GANs when dealing with low-dimensional applications, we equip our space with the standard Euclidean distance. However, for the case of image generation, we follow [6, 21] and consider the euclidean distance between embeddings of a pre-trained network, that convey more semantic information. Thus, for a pair of images (a,b), we define the distance $d(a,b)$ as



(a) Synthetic WGAN: real samples in green and fake ones in blue.

(b) MC method optimizing for a precision criterion [3].

(c) Optimizing for Wasserstein criterion with latentRS (ours $\star$).

(d) Heatmap of the w^{φ} in the latent space (in the blue areas, w^{φ}=0).

Figure 2: Synthetic experiment mimicking the setting of GANs in high-dimension, where data and generated manifolds are close but do not perfectly intersect. While DRS only selects the intersection of manifolds and ignores the rest, the latent importance weights define a rejection mechanism that minimizes the Wasserstein distance. For conciseness, WGAN stands for WGAN-GP.

	EMD Swiss Roll	EMD 25 Gaussians
WGAN	0.030±0.002	0.044±0.001
WGAN: DRS	0.036±0.004	0.038±0.002
WGAN: SIR	0.037±0.003	0.041±0.001
WGAN: DOT	0.029±0.003	**0.035±0.002**
WGAN: latentRS ($\star$)	**0.025**±0.002	0.036±0.001

Table 1: Comparison of latentRS with concurrent methods on two synthetic datasets in the same setting as DOT [32]. Our method enables a consistent gain in EMD, surpassing other methods on Swiss Roll and slightly behind DOT on Mixture of 25 Gaussians. For conciseness, WGAN stands for WGAN-GP.

$d(a,b) = \|\phi(a) - \phi(b)\|_2$ where ϕ is a pre-softmax layer of a supervised classifier. On MNIST and F-MNIST, the classifier is pre-trained on the given dataset. On CelebA and LSUN Church, we use VGG-16 pre-trained on ImageNet.

To begin with, we report the FID [16]. We also compare the performance of the different methods with the Precision/Recall (PR) metric [21]. It is a more robust version of the Precision/Recall metric, which was first applied in the context of GANs by [29]. Finally, we approximate the Wasserstein distance using the Earth Mover's Distance (EMD) between generated and real data points. This measure is particularly suited to the study of WGANs, since it is linked to their objective function. Letting $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_n\}$ be two collections of n data points and $\mathscr{S}$ be the set of permutations of $[1,n]$, the Earth Mover's distance between X and Y is defined by:

$$\mathrm{EMD}(X,Y) = \min_{\sigma \in \mathscr{S}} \sum_{i=1}^{n} \|x_i - y_{\sigma_i}\|$$

4.2. Synthetic datasets

To begin the experimental study, we test our method on 2D synthetic datasets in the same setting as [32]. Table 1 compares the latentRS method with previous approaches on the Swiss roll dataset and on a mixture of 25 Gaussians. We see that the network w^{φ} efficiently redistributes the pre-trained distribution μ_θ since $\mathrm{EMD}(\mu_n, \mu_\theta^{\varphi})$ is significantly smaller than $\mathrm{EMD}(\mu_n, \mu_\theta)$.

4.3. Image datasets

Implementation of baselines. We now compare latentRS, latentGA, and latentRS+GA with previous works leveraging discriminator's information on high-dimensional data. In particular, we implemented a wide set of post-processing methods for GANs: DRS [3], MH-GAN [34], SIR [11] and DOT [32]. DRS, MH-GAN and SIR use the same density ratios, and we did not see significant differences between those three methods in our experiments. Consequently, for the following experiments, we compare our algorithms to SIR and DOT. For SIR, we take the discriminator at the end of the adversarial training, fine-tune it with the binary cross-entropy loss and select the best model in terms of EMD. Overall, we explicitly follow the framework used by [3, 11]: we keep the gradient penalty [12], spectral normalization [25] during fine-tuning and do not include an explicit mechanism to calibrate the classifier.

4.3.1 Description of datasets and neural architectures

We first consider two well-known image datasets that are **MNIST** [22] and **FashionMNIST** (F-MNIST). We follow [20] and use a standard CNN architecture composed of a sequence of blocks made of 3x3 convolution layer and ReLU activations with nearest neighbor upsampling. For these datasets, the discriminator is trained using the hinge loss [23] with gradient penalty (Hinge-GP). Finally, the architecture used for the network w^{φ} is very simple: an MLP with 4 fully-connected layers and ReLU activation (with a width $= 4 \times d$).

CelebA [24] is a large-scale dataset of faces covering a variety of poses. We use a pre-trained model of Progressive GAN [17] at 128x128 resolution. The discriminator is

CelebA 128x128	Prec. (↑)	Rec. (↑)	EMD (↓)	FID (↓)	Inference (ms)
ProGAN	74.2±0.9	60.7±1.4	25.4±0.1	11.30±0.02	3.6
ProGAN: SIR	79.5±0.4	**57.3±1.0**	24.9±0.2	12.01±0.04	49.0
ProGAN: DOT	81.3±1.0	52.9±1.4	25.0±0.1	11.01±0.03	67.6
ProGAN: latentRS (⋆)	80.4±0.9	55.7±1.0	24.7±0.1	10.77±0.04	**4.5**
ProGAN: latentRS+GA (⋆)	**83.3±1.0**	52.7±0.9	**24.5±0.1**	**10.75±0.04**	20.5
LSUN Church 256x256					
StyleGAN2	55.6±1.2	62.4±1.1	23.6±0.1	6.91±0.02	11.7
StyleGAN2: SIR	60.5±1.4	**58.1±1.3**	23.4±0.1	7.36±0.01	130.0
StyleGAN2: DOT	67.4±1.4	48.3±1.0	23.1±0.1	6.85±0.02	196.7
StyleGAN2: latentRS (⋆)	63.3±0.7	57.7±1.0	23.1±0.1	6.31±0.02	**16.2**
StyleGAN2: latentRS+GA (⋆)	**72.6±1.1**	43.2±1.3	**22.6±0.1**	**6.27±0.03**	43.2

Table 2: latentRS+GA is the best performer, and latentRS matches SOTA with a significantly reduced inference cost (by an order of at least 10). ± is 97% confidence interval. Inference refers to the time in milliseconds needed to compute one image on a NVIDIA V100 GPU.

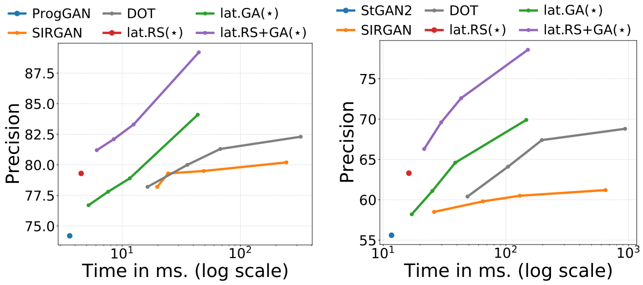


Figure 3: Visualization of the trade-off between the time spent to generate an image and its average precision. Interestingly, latentRS+GA has the best Pareto front. Left: ProGAN trained on CelebA. Right: StyleGan2 trained on LSUN Church.

trained using a Wasserstein loss with gradient-penalty. Also, the architecture used for the network w^{φ} is really standard: a 5 hidden-layer MLP with a width of the same size than the latent space dimension.

LSUN Church [35] is a dataset of church images with a lot of variety. We use a pre-trained model of StyleGAN2 [19] at 256x256 resolution. Similarly to the CelebA dataset, the discriminator is trained using a Wasserstein loss with gradient-penalty. Also, the architecture used for the network w^{φ} is a 3 hidden-layer MLP with width equal to the latent space dimension. Note that the StyleGAN architecture already contains an 8-layer MLP network $M_{\theta} : \mathbb{R}^d \to \mathbb{R}^d$ that transforms a latent space variable to an intermediate latent variable [18]. We consequently leverage this pre-trained M_{θ} and train the network w^{φ} on top of it.

4.3.2 Results

The main results of this comparison are shown in Table 2 and Figure 3. On all studied datasets, our latentRS+GA outperforms every other method on the EMD with lower computational cost. Interestingly, latentRS achieves good performance on FID while being more than 15 times faster. Figure 3 is particularly interesting since it gives a good visualization of the trade-off between computational cost and quality of the generated samples. On this experiment ran on CelebA and LSUN, we observe that latentRS+GA can achieve a significantly better precision than both SIR and DOT while being much faster. Interestingly, even though these datasets are high-dimensional, contain only one-class, and w^{φ} has a low capacity, our proposed methods still produce interesting results.

To visualize the efficiency of the proposed method, Figure 4 shows generated samples along with their acceptance probabilities. As expected, we observe that higher acceptance probabilities correlate with higher quality images. Figure 5 stresses how generated images improve when performing latent gradient ascent on the importance weights. Finally, we provide more qualitative results and details on the experiments in supplementary material.

5. Conclusion

This paper deals with improving the quality of pre-trained GANs. Conversely, to concurrent methods which leverage the discriminator at inference time, we propose to train adversarially a neural network which learns importance weights in the latent space of GANs. These latent importance weights are then used with two complementary sampling methods: latentRS and latentGA. We experimentally show that this latent reweighting consistently enhances the quality of the pre-trained model. When these two methods are combined in latentRS+GA, it surpasses concurrent post-training methods while being less computationally intensive.

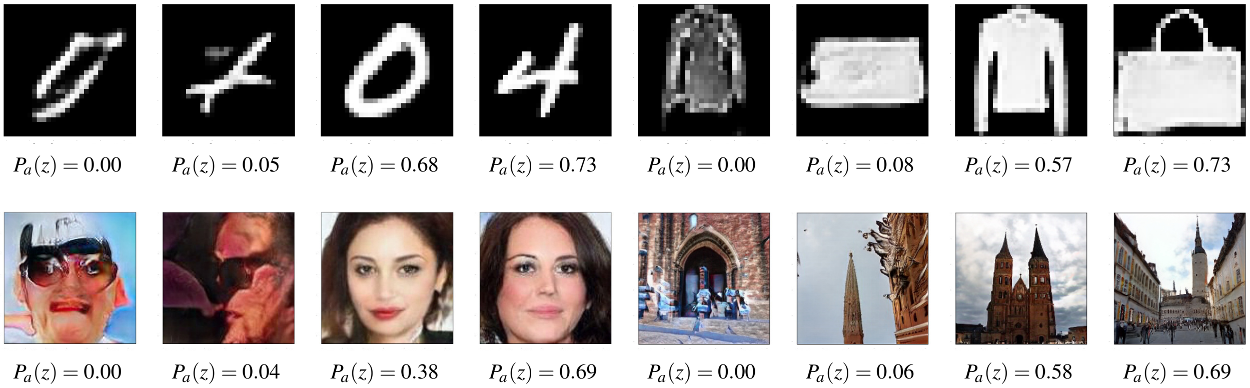


Figure 4: Images drawn from the generative model and their acceptance probabilities with the latentRS algorithm, given by the network w^{φ}. As expected, the quality of images correlates with higher acceptance rates on all datasets: MNIST, F-MNIST, CelebA, and LSUN.

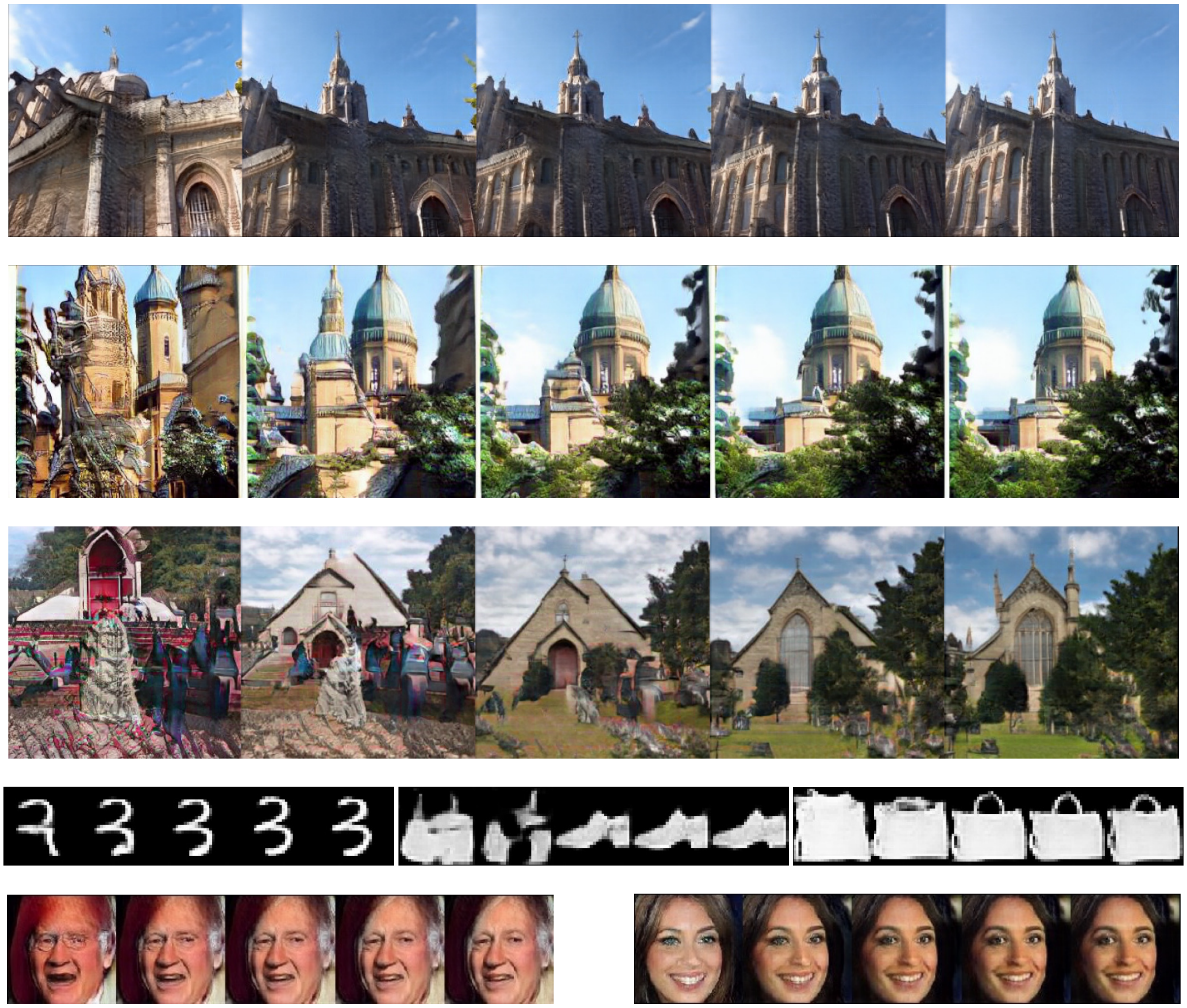


Figure 5: Gradient ascent on latent importance weights (latentGA): the quality is gradually improved as we move to larger importance weights. Each image is generated only for visualization, and one can run this gradient ascent directly in the latent space using w^{φ}. Interestingly, this gradient ascent only involves a simple MLP network which is computationally cheap.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017. 1, 2, 3, 5

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR, 2017. 2

[3] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *International Conference on Learning Representations*, 2019. 1, 3, 4, 5, 6

[4] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Robust optimal transport with applications in generative modeling and domain adaptation. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[5] Léon Bottou, Jonas Peters, Joaquin Quinonero Candela, Denis Xavier Charles, Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Y Simard, and Ed Snelson. Counterfactual Reasoning and Learning Systems: the example of Computational Advertising. *Journal of Machine Learning Research*, 14(1):3207–3260, 2013. 4, 5

[6] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 1, 5

[7] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems*, volume 33, pages 12275–12287. Curran Associates, Inc., 2020. 3

[8] L. Faury, U. Tanielian, E. Dohmatob, E. Smirnova, and F. Vasile. Distributionally robust counterfactual risk minimization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3850–3857. AAAI Press, 2020. 5

[9] C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29:983–1049, 2016. 3

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and J. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 1, 2

[11] Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In *Advances in Neural Information Processing Systems*, pages 11056–11068, 2019. 3, 4, 6, 12

[12] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A.C. Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017. 6

[13] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 166–174, 2017. 1, 2

[14] Valentin Hartmann and Dominic Schuhmacher. Semi-discrete optimal transport: a solution procedure for the unsquared euclidean distance case. *Mathematical Methods of Operations Research*, pages 1–31, 2020. 11

[15] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970. 3

[16] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. 6

[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. 5, 6

[18] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 1, 2, 5, 7

[19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 5, 7

[20] Mahyar Khayatkhoei, Maneesh K Singh, and Ahmed Elgammal. Disconnected manifold learning for generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 7343–7353, 2018. 1, 2, 6, 11

[21] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural*

Information Processing Systems 32, pages 3927–3936. Curran Associates, Inc., 2019. 2, 5, 6

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 6

[23] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 6

[24] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 6

[25] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 6

[26] Aldo Pratelli. On the equality between monge's infimum and kantorovich's minimum in optimal mass transportation. In *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, volume 43, pages 1–13. Elsevier, 2007. 11

[27] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013. 3, 5

[28] Donald B Rubin. Using the sir algorithm to simulate posterior distributions. *Bayesian statistics*, 3:395–402, 1988. 3

[29] M.S.M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems 31*, pages 5228–5237. Curran Associates, Inc., 2018. 2, 6

[30] Adith Swaminathan and Thorsten Joachims. Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization. *Journal of Machine Learning Research*, 16(1):1731–1755, 2015. 4

[31] A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems 28*, pages 3231–3239. Curran Associates, Inc., 2015. 4

[32] Akinori Tanaka. Discriminator optimal transport. In *Advances in Neural Information Processing Systems*, pages 6813–6823, 2019. 3, 5, 6, 12

[33] U. Tanielian, T. Issenhuth, E. Dohmatob, and J. Mary. Learning disconnected manifolds: a no gan's land. In *International Conference on Machine Learning*, 2020. 1, 2, 3

[34] Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-hastings generative adversarial networks. In *International Conference on Machine Learning*, pages 6345–6353, 2019. 1, 3, 6

[35] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 7

A. Proof of Lemma 1

Let's prove that $\mathbb{E}_{\gamma^{\varphi}} = 1$. We have that:

$$\int_{\mathbb{R}^d} 1\gamma^{\varphi}(\mathrm{dz}) = \int_{\mathbb{R}^{\mathrm{d}}} \mathrm{w}^{\varphi}(\mathrm{z})\gamma(\mathrm{dz}) = 1,$$

by assumption. Consequently, the measure γ^{φ} is a well-defined probability distribution on $\mathbb{R}^d$.

B. Proof of Theorem 1

It is clear that the network w^{φ} is a density function with respect to the distribution γ defined on $\mathbb{R}^d$. Consequently, the measure μ_θ^φ is absolutely continuous with respect to μ_θ and thus with respect to the Lebesgue measure.

We start the proof by stating that for any absolutely continuous distribution μ_θ^φ, there exists an optimal transport T_θ^φ, [26, Theorem B], such that $W(\mu_n, \mu_\theta^\varphi) = \int_{\mathbb{R}^D} \|x - T_\theta^\varphi(x)\| \mathrm{d}\mu_\theta^\varphi$ [14]. Recall that for any $x \in \mathbb{R}^D$, there exists $X_i \in [1,n]$ such that $T_\theta^\varphi(x) = X_i$. Since μ_θ^φ is absolutely continuous, there exists a ball $B(z,r)$ centered in $z \in \mathbb{R}^d$ with radius $r > 0$ such that $\mu_\theta^\varphi(B(z,r)) > 0$ and, we have:

1. there exists $i \in [1,n]$ such that for all $x \in B(z,r)$, $T_\theta^\varphi(x) = X_i$,
2. for all $x \in B(z,r)$, $\|x - X_i\| > \|X_i - \tilde{X}_i\|$, recall that $\tilde{X}_i = \underset{z \in \mathbb{R}^d}{\arg\min} \|X_i - G_\theta(z)\|$.

Consequently, we have:

$$\begin{aligned}
W(\mu_n, \mu_\theta^\varphi) = \int_{\mathbb{R}^D} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) &= \int_{\mathbb{R}^D \setminus B(z,r)} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) + \int_{B(z,r)} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) \\
&> \int_{\mathbb{R}^D \setminus B(z,r)} \|x - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) + \int_{B(z,r)} \|X_i - \tilde{X}_i\| \mu_\theta^\varphi(\mathrm{d}x) \\
&\geqslant \int_{\mathbb{R}^D \setminus B(z,r)} \|\tilde{T}_\theta^\varphi(x) - T_\theta^\varphi(x)\| \mu_\theta^\varphi(\mathrm{d}x) + \int_{B(z,r)} \|X_i - \tilde{X}_i\| \mu_\theta^\varphi(\mathrm{d}x) \\
&\text{(where } \tilde{T}_\theta^\varphi(x) = \min_{z \in \mathbb{R}^d} \|z - T_\theta^\varphi(x)\|) \\
&= \frac{1}{n} \sum_{i=1}^{n} \|X_i - \tilde{X}_i\| \\
&= W(\mu_n, \frac{1}{n} \sum_{i=1}^{n} \delta(\tilde{X}_i)),
\end{aligned}$$

where δ refers to the Dirac probability distribution.

Finally, when taking the infinimum over all continuous functions φ, we have that:

$$W(\mu_n, \frac{1}{n} \sum_{i=1}^{n} \delta(\tilde{X}_i)) \leqslant \inf_{\varphi \in \Phi} W(\mu_n, \mu_\theta^\varphi)$$

C. Evaluation details

Precision recall metric. For the precision-recall metric, we use the algorithm from [20]. Namely, when comparing the set of real data points $(x_1, ..., x_n)$ with the set of fake data points $(y_1, ..., y_n)$:

A point x_i has a recall $r(x_i) = 1$ if there exists y_j, such that $\|x_i - y_j\| \leq \|y_j - y_j(k)\|$, where $y_j(k)$ is the k-nearest neighbor of n. Finally, the recall is the average of individual recall: $\frac{1}{n} \sum_i r(x_i)$.

A point y_i has a precision $p(y_i) = 1$ if there exists x_j, such that $\|y_i - x_j\| \leq \|x_j - x_j(k)\|$, where $x_j(k)$ is the k-nearest neighbor of n. Finally, the precision is the average of individual precision: $\frac{1}{n} \sum_i p(x_i)$.

Parameters. For all datasets, we use $k = 3$ (3rd nearest neighbor). For MNIST and F-MNIST, we use a set of $n = 2048$ points. For CelebA and LSUN Church, we use a set of $n = 1024$ points. This is also valid for the EMD. For FID, we use the standard protocol with $n = 50000$ points and Inception Net. We run 10 evaluations of each metric (each evaluation is done with a different set of random points), report the average and the 97% confidence interval by considering that we have 10 i.i.d. samples from a normal distribution.

D. Sampling algorithms: latentRS, latentGA, and latentRS+GA

We present here the three sampling algorithms associated with our importance weight function w^{φ}. See Section E for details on the hyper-parameters used in latentGA and latentRS+GA.

```
Algorithm 3: LatentRS
Requires: Prior $Z$, Gen. $G_\theta$, Importance weight network $w^{\varphi}$, maximum importance weight $m$;
while True do
    Sample $z \sim Z$ ;
    Sample $\alpha \sim$ Uniform$[0,1]$ ;
    if $\frac{w^{\varphi}(z)}{m} \geq \alpha$ then
        break;
    end
end
$x \leftarrow G_\theta(z)$;
Result: Selected point x
```

```
Algorithm 4: Latent Gradient Ascent (latentGA)
Requires: Prior $Z$, number of dimensions of the prior $d$, Gen. $G_\theta$, Importance weight network $w^{\varphi}$, number of steps $N$,
  step size $\varepsilon$;
Sample $z \sim Z$ ;
for $n = 1 : N$ do
    $\text{grad}_z \leftarrow \nabla_z w^{\varphi}(z)$ ;
    if $Z == \mathcal{N}(0, I)$ and $d >> 1$ then
        ## Projection step for high-dimensional gaussians ## ;
        $\text{grad}_z \leftarrow \text{grad}_z - (\text{grad}_z \cdot z)z/\sqrt{d}$ ;
    end
    $z \leftarrow z + \varepsilon * \text{grad}_z$ ;
end
$x \leftarrow G_\theta(z)$;
Result: Selected point x
```

E. Hyper-parameters.

SIR [11]: Model selection: we fine-tune with a binary cross-entropy loss the discriminator from the end of the adversarial training and select the best model in terms of EMD. We tested with/without regularizing the discriminator during the fine-tuning (with gradient penalty or spectral normalization). Without regularization, the performance drops fast. Best results are obtained by regularizing the discriminator, thus we report these results.

We use then use Sampling-Importance-Resampling algorithm. In SIR, we sample N points from the generator, compute their importance weights according density ratios, and accept one of them (each point is accepted with a probability proportional to its importance weight). The hyper-parameter of SIR algorithm is N. Results for grid search on N are shown below in Table 4 and Table 5. In Table **??**, results are shown with $N = 10$.

DOT [32]: Model selection: we fine-tune with the WGAN-GP loss the discriminator from the end of the adversarial training and select the best model in terms of EMD, when running DOT. We perform a projected gradient descent as described in [32] with SGD. Hyper-parameters are the number of steps N_{steps} and the step size ε. We made the following grid search: $N_{steps} = [2, 5, 10, 50]$ and $\varepsilon = [0.01, 0.05, 0.1]$. Results for grid search on N_{steps} are shown below in Table 4 and Table 5. In Table **??**, results are shown with $N_{steps} = 10$ and $\varepsilon = 0.05$ or $\varepsilon = 0.01$ depending on the dataset (we select the best one).

Training of w^{φ}: For MNIST and F-MNIST, we use the same hyper-parameters: $\lambda_1 = 10$, $\lambda_2 = 3$ and $m = 3$. w^{φ} is a standard MLP with 4 hidden layers, each having 400 nodes (4x dimension of latent space), and relu activation. The output

Algorithm 5: Latent RS+GA

```
Requires: Prior $Z$, Number of dimensions of the prior $d$, Gen. $G_\theta$, Importance weight network $w^\varphi$, maximum
  importance weight $m$, number of steps $N$, step size $\varepsilon$;
while True do
    Sample $z \sim Z$ ;
    Sample $\alpha \sim$ Uniform$[0;1]$ ;
    if $\frac{w^\varphi(z)}{m} \geq \alpha$ then
        break;
    end
end
for $n = 1 : N$ do
    $\mathrm{grad}_z \leftarrow \nabla_z w^\varphi(z)$ ;
    if $Z = \mathcal{N}(0, I)$ and $d >> 1$ then
        ## Projection step for high-dimensional gaussians ## ;
        $\mathrm{grad}_z \leftarrow \mathrm{grad}_z - (\mathrm{grad}_z \cdot z)z/\sqrt{d}$ ;
    end
    $z \leftarrow z + \varepsilon * \mathrm{grad}_z$ ;
end
Compute $x = G_\theta(z)$;
Result: Selected point x
```

layer is 1-dimensional and with a relu activation. The learning rate of the discriminator is $4 * 10^{-4}$, the learning rate of w^φ is 10^{-4}. The two networks are optimized with Adam algorithm, where we set $\beta = (0.5, 0.5)$. We use 1 step of importance weight optimization for 1 step of discriminator optimization.

For Progressive GAN on CelebA (128x128), we use: $\lambda_1 = 20$, $\lambda_2 = 5$ and $m = 3$. w^φ is a standard MLP with 4 hidden layers, each having 512 nodes (1x dimension of latent space), and leaky-relu activation (0.2 of negative slope). The output layer is 1-dimensional and with relu activation. Since we do not have the pre-trained discriminator, we first train a WGAN-GP discriminator between ProGAN and CelebA images for 500 steps, and then start the adversarial training of w^φ. The learning rate of the discriminator is 10^{-4}, the learning rate of w^φ is 10^{-5}. The two networks are optimized with Adam algorithm, where we set $\beta = (0., 0.999)$. During optimization, we perform iteratively 3 w^φ updates and 1 discriminator's updates.

For StyleGAN2 on LSUN Church (256x256), we use: $\lambda_1 = 30$, $\lambda_2 = 5$ and $m = 2$. w^φ is a standard MLP with 3 hidden layers, each having 512 nodes (1x dimension of latent space), and leaky-relu activation (0.2 of negative slope). The output layer is 1-dimensional and with a relu activation. Since we do not have the pre-trained discriminator, we first train a WGAN-GP discriminator between StyleGAN2 and LSUN Church images for 500 steps, and then start the adversarial training of w^φ. The learning rate of the discriminator is 10^{-4}, the learning rate of w^φ is 10^{-5}. The two networks are optimized with Adam algorithm, where we set $\beta = (0., 0.999)$. During optimization, we perform 3 w^φ updates for 1 discriminator's updates.

LatentRS: Once the network w^φ is trained (see above), there is no hyper-parameter for latentRS algorithm.

LatentGA and latentRS+GA: We use the same neural network than in LatentRS. The hyper-parameters for this method are similar to DOT: number of steps of gradient ascent N_{steps} and step size ε. With the model selected on LRS, we make the following grid search: $N_{steps} = [2, 5, 10, 50]$ and $\varepsilon = [0.01, 0.05, 0.1]$. Best results were obtained with $\varepsilon = 0.05$ on all datasets. Results for grid search on N_{steps} are shown below in Table 4 and Table 5. In Table 2, results are shown with $N_{steps} = 10$ and $\varepsilon = 0.05$.

F. Comparisons with concurrent methods on synthetic and real-world datasets

In this section, we provide more quantitative results: a comparison of SIR, DOT, SIR, LatentRS and latentRS+GA on MNIST and F-MNIST in Table 3; an ablation study on the impact of number of points (respectively gradient ascent steps) in SIR (respectively DOT, latentGA and latentRS+GA), on ProGAN trained on CelebA in Table 4 and StyleGAN2 trained on Lsun Church in Table 5.

MNIST	Prec. (↑)	Rec. (↑)	EMD (↓)	FID (↓)	Inference (ms)
Hinge-GP	87.4±0.9	94.6±0.4	24.9±0.3	53.6±7.2	0.7
HGP: SIR	88.8±1.0	94.3±0.5	24.2±0.2	38.7±3.1	10.0
HGP: DOT	89.5±0.6	94.0±0.3	24.8±0.2	43.3±3.4	15.7
HGP: latentRS (⋆)	89.0±0.4	**94.7±0.7**	24.1±0.3	**36.3±3.2**	**1.6**
HGP: latentRS+GA (⋆)	**91.8±1.0**	92.8±0.4	**23.4±0.2**	38.2±3.8	8.6
F-MNIST					
Hinge-GP	86.4±0.6	86.8±0.6	68.6±0.4	598.9±55.5	0.7
HGP: SIR	86.6±1.1	**88.0±0.5**	68.0±0.5	499.6±31.1	10.0
HGP: DOT	**88.7±0.6**	86.6±0.7	67.7±0.5	508.3±45.7	15.7
HGP: latentRS (⋆)	86.8±0.8	87.5±0.9	67.6±0.6	**438.3±50.2**	**1.6**
HGP: latentRS+GA (⋆)	88.4±0.7	86.8±0.7	**67.0±0.9**	475.5±58.5	8.6

Table 3: latentRS+GA is the best performer and latentRS matches SOTA with a significantly reduced inference cost (by an order of at least 10). FID was computed using the same dataset-specific classifier used for the Precision/Recall metric. ± is 97% confidence interval. Inference refers to the time in milliseconds needed to compute one image on a NVIDIA V100 GPU.

CelebA 128x128	Precision	Recall	EMD	Inference Time
ProGAN	74.2±0.9	60.7±1.4	25.4±0.1	3.6
ProGAN: SIR (n=2)	78.2±1.0	58.4±1.3	25.0±0.1	9.8
ProGAN: SIR (n=5)	79.3±0.6	57.6±1.3	24.9±0.1	24.5
ProGAN: SIR (n=10)	79.5±0.4	57.3±1.0	24.9±0.2	49.0
ProGAN: SIR (n=50)	80.2±1.0	57.4±1.4	25.0±0.1	245.0
ProGAN: DOT (n=2)	78.2±1.1	58.6±1.1	24.9±0.1	16.4
ProGAN: DOT (n=5)	80.0±1.0	56.0±1.1	24.8±0.1	35.6
ProGAN: DOT (n=10)	81.3±1.0	52.9±1.4	25.0±0.1	67.6
ProGAN: DOT (n=50)	82.3±0.7	52.1±1.3	25.0±0.1	323.6
ProGAN: latentGA (n=2) (⋆)	76.7±1.2	**59.4±0.9**	25.2±0.1	5.2
ProGAN: latentGA (n=5) (⋆)	77.8±1.2	58.4±0.7	25.1±0.1	7.6
ProGAN: latentGA (n=10) (⋆)	78.9±1.2	57.4±0.7	25.0±0.1	11.6
ProGAN: latentGA (n=50) (⋆)	84.1±1.2	49.0±1.3	24.8±0.1	43.6
ProGAN: latentRS+GA (n=2) (⋆)	81.2±0.8	55.3±1.5	24.7±0.1	6.1
ProGAN: latentRS+GA (n=5) (⋆)	82.1±0.7	54.3±1.2	24.6±0.2	8.5
ProGAN: latentRS+GA (n=10) (⋆)	83.3±1.0	52.7±0.9	**24.5±0.1**	12.5
ProGAN: latentRS+GA (n=50) (⋆)	**89.2±0.8**	36.1±0.7	25.0±0.1	44.5
ProGAN: latentRS (⋆)	79.3±1.0	56.5±1.2	24.8±0.2	**4.5**

Table 4: Comparison of the proposed methods (latentRS, latentGA, and latentRS+GA) with concurrent methods on ProgressiveGan (CelebA 128x128). For this specific study, we explore different computational budgets for SIR, DOT, latentGA, and latentRS+GA. latentRS+GA enables a consistent gain in both EMD and precision for a reasonable computational overhead.

LSUN Church (256x256)	Precision	Recall	EMD	Inference Time
StyleGAN2	55.6±1.2	62.4±1.1	23.6±0.1	11.7
StyleGAN2: SIR (n=2)	58.5±0.7	60.7±1.3	23.5±0.1	26.0
StyleGAN2: SIR (n=5)	59.8±1.1	59.0±1.2	23.5±0.1	65.0
StyleGAN2: SIR (n=10)	60.5±1.4	58.1±1.3	23.4±0.1	130.0
StyleGAN2: SIR (n=50)	61.2±1.2	57.8±0.9	23.4±0.1	650.0
StyleGAN2: DOT (n=2)	60.4±1.4	57.0±1.1	23.4±0.1	48.7
StyleGAN2: DOT (n=5)	64.1±0.9	52.2±1.0	23.2±0.1	104.2
StyleGAN2: DOT (n=10)	67.4±1.4	48.3±1.0	23.1±0.1	196.7
StyleGAN2: DOT (n=50)	68.8±0.9	37.0±1.1	23.6±0.1	937.7
StyleGAN2: latentGA (n=2) (⋆)	58.2±1.0	**61.4±1.2**	23.4±0.1	17.1
StyleGAN2: latentGA (n=5) (⋆)	61.1±0.9	58.5±1.1	23.2±0.1	25.2
StyleGAN2: latentGA (n=10) (⋆)	64.6±0.9	55.9±1.5	23.0±0.1	38.7
StyleGAN2: latentGA (n=50) (⋆)	69.9±1.1	47.2±1.4	22.8±0.1	146.7
StyleGAN2: latentRS+GA (n=2) (⋆)	66.3±1.2	54.8±1.3	23.0±0.1	21.6
StyleGAN2: latentRS+GA (n=5) (⋆)	69.6±1.0	50.6±0.9	22.8±0.2	29.7
StyleGAN2: latentRS+GA (n=10) (⋆)	72.6±1.1	43.2±1.3	**22.6±0.1**	43.2
StyleGAN2: latentRS+GA (n=50) (⋆)	**78.6±1.2**	34.1±0.9	**22.6±0.1**	151.2
StyleGAN2: latentRS (⋆)	63.3±0.7	57.7±1.0	23.1±0.2	**16.2**

Table 5: Comparison of the proposed methods (latentRS, latentGA, and latentRS+GA) with concurrent methods on StyleGAN2 (LSUN Church 256x256). For this specific study, we explore different computational budgets for SIR, DOT, latentGA, and latentRS+GA. latentRS+GA enables a consistent gain in both EMD and precision with a reasonable computational overhead.

MNIST/F-MNIST 28x28	Inference Time
MNIST Generator	0.7
MNIST Generator + Discriminator	1.0
MNIST $\nabla_z D(G(z))$ (gradient for latent DOT)	1.5
MNIST Network w^{φ} (⋆)	0.3
MNIST: $\nabla_z W(z)$ (gradient for latent GA on IW) (⋆)	0.7

CelebA 128x128	Inference Time
ProGAN Generator	3.6
ProGAN Generator + Discriminator	4.9
ProGAN $\nabla_z D(G(z))$ (gradient for latent DOT)	6.4
ProGAN Network w^{φ} (⋆)	0.3
ProGAN: $\nabla_z W(z)$ (gradient for latent GA on IW) (⋆)	0.8

LSUN Church 256x256	Inference Time
StyleGAN2 Generator	11.7
StyleGAN2 Generator + Discriminator	13.0
StyleGAN2 $\nabla_z D(G(z))$ (gradient for latent DOT)	18.5
StyleGAN2 Network w^{φ} (⋆)	1.5
StyleGAN2: $\nabla_z W(z)$ (gradient for latent GA on IW) (⋆)	2.7

Table 6: Inference time for one pass of different computational graphs. With the acceptance rate of the different methods, it allows to compute the runtime of these methods.

G. Qualitative results of latentGA.

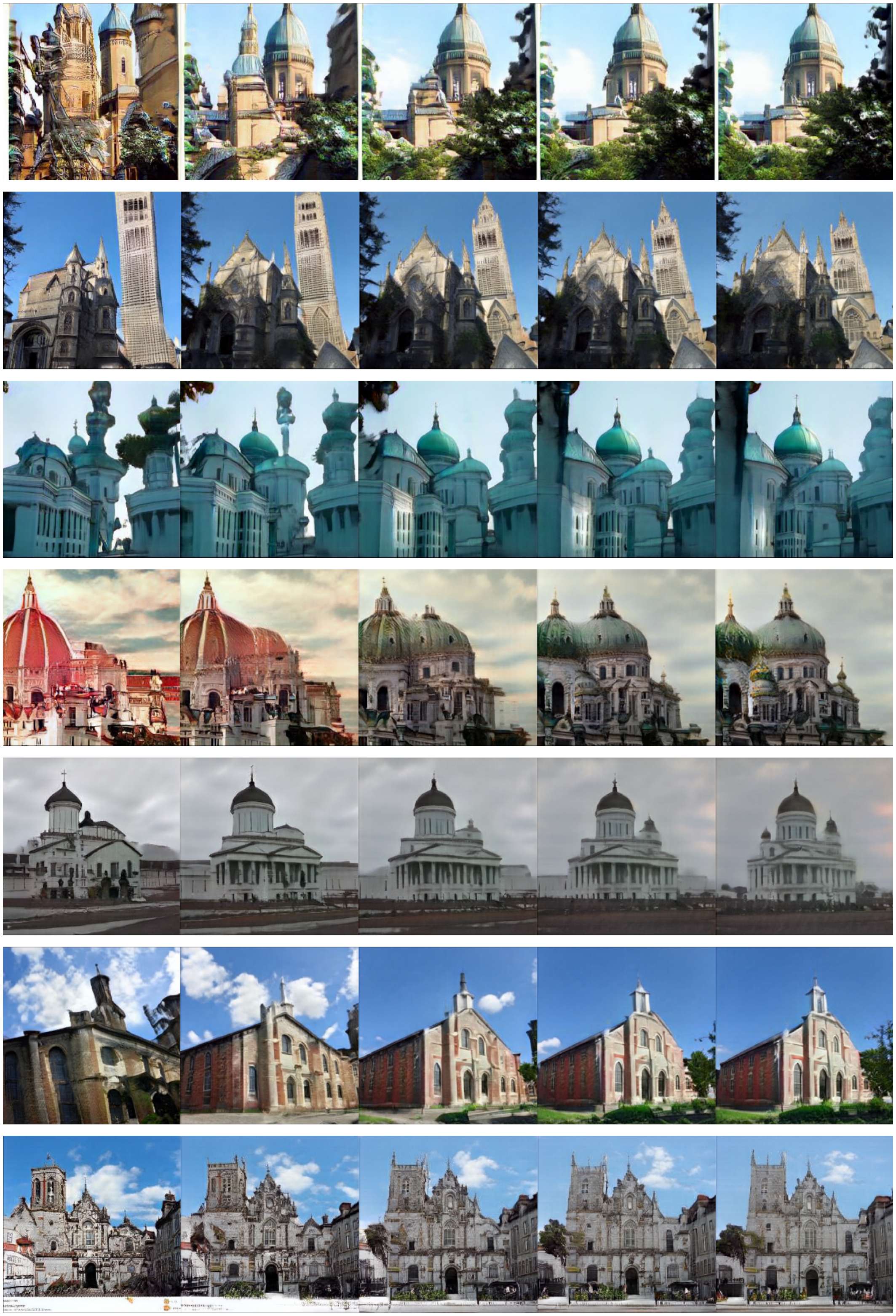


Figure 6: Gradient ascent on latent importance weights (latentGA), on StyleGAN2 trained on LSUN Church.

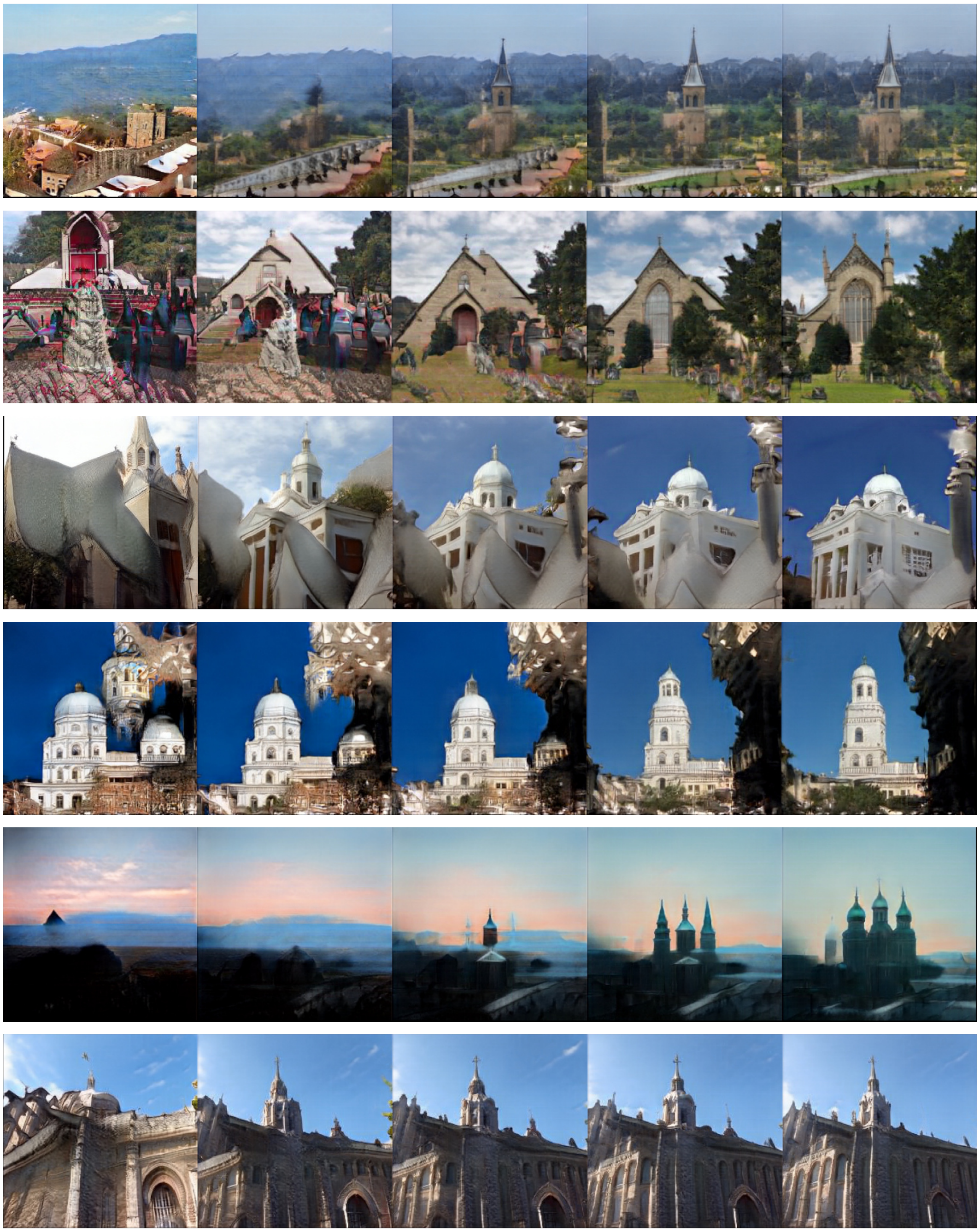
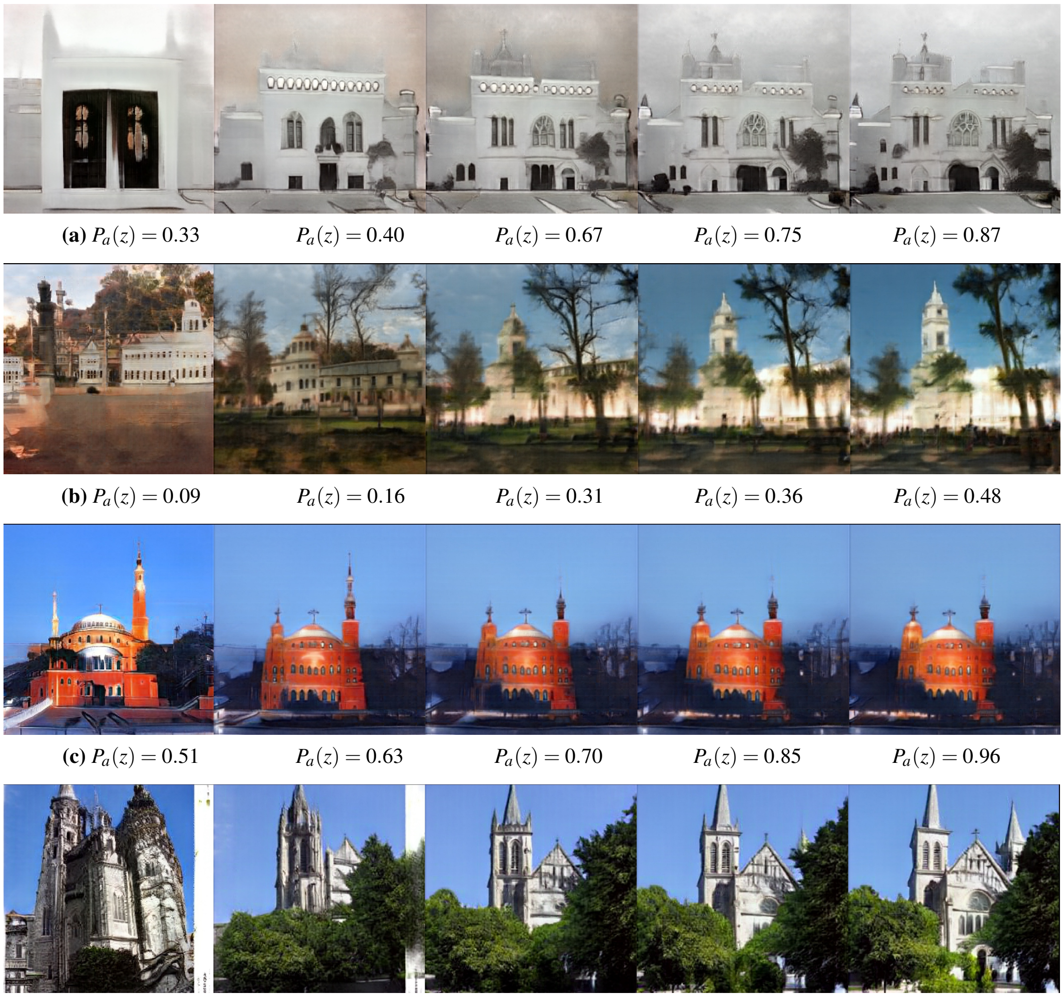


Figure 7: Gradient ascent on latent importance weights (latentGA), on StyleGAN2 trained on LSUN Church.



(a) $P_a(z) = 0.33$ $P_a(z) = 0.40$ $P_a(z) = 0.67$ $P_a(z) = 0.75$ $P_a(z) = 0.87$

(b) $P_a(z) = 0.09$ $P_a(z) = 0.16$ $P_a(z) = 0.31$ $P_a(z) = 0.36$ $P_a(z) = 0.48$

(c) $P_a(z) = 0.51$ $P_a(z) = 0.63$ $P_a(z) = 0.70$ $P_a(z) = 0.85$ $P_a(z) = 0.96$

(d) $P_a(z) = 0.41$ $P_a(z) = 0.56$ $P_a(z) = 0.59$ $P_a(z) = 0.64$ $P_a(z) = 0.68$

Figure 8: Gradient ascent on latent importance weights (latentGA) on StyleGAN2 trained on LSUN Church. We visualize the evolution of the probability $P_a(z) = \frac{w^{\varphi}(z)}{m}$ associated to each sample.

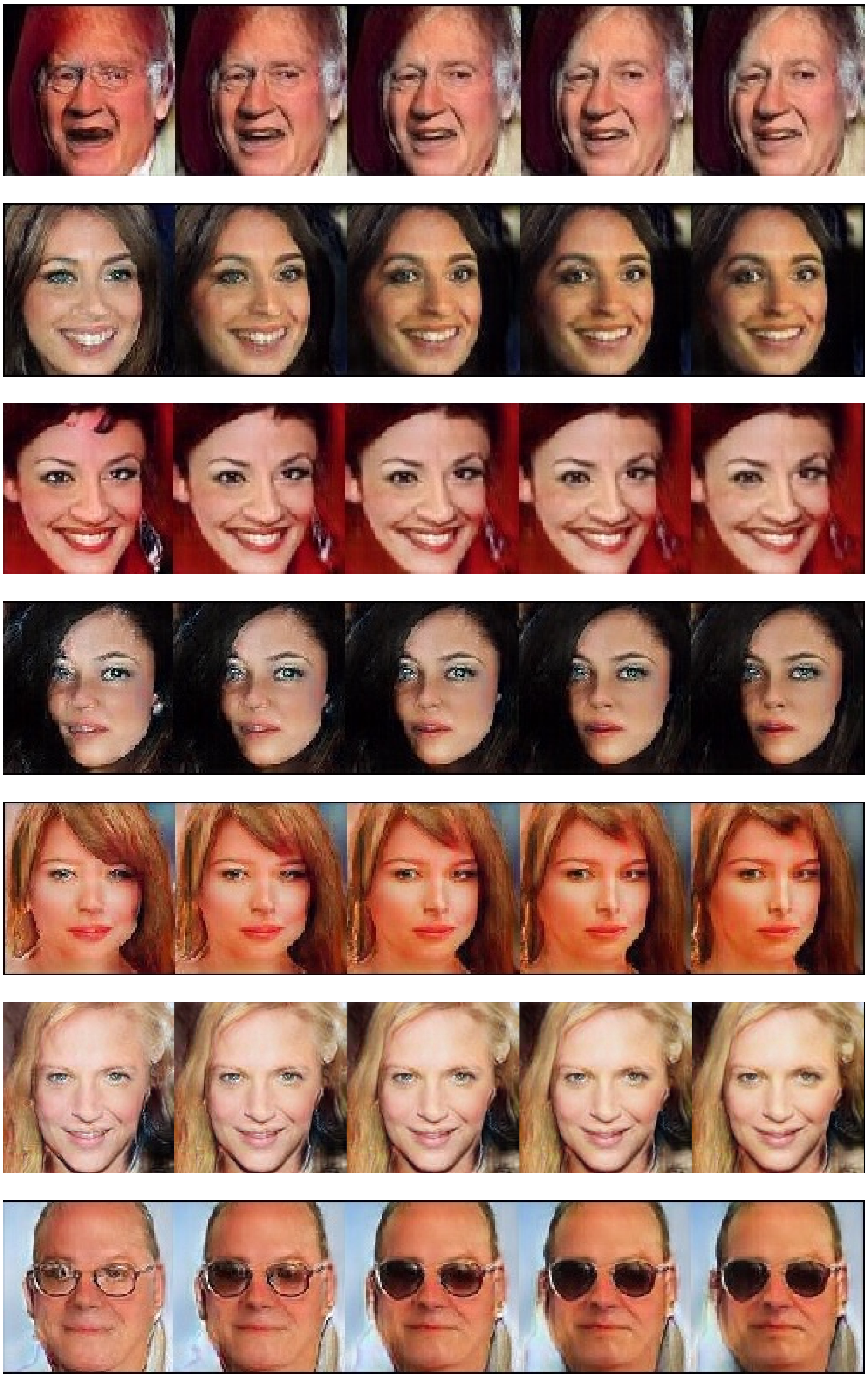


Figure 9: Gradient ascent on latent importance weights (latentGA), on Progressive GAN trained on CelebA.