A. Dimension drop in GANs

Here we validate Lemma 1 and Lemma 2 empirically. Specifically, we want to make the following two questions clear.

1. Does the Jacobian rank decrease as the network gets deeper?
2. Does the intrinsic dimension of feature space decrease as the network gets deeper?

However, those two things are not easy to validate. Estimating the Jacobian rank of complicated function couplings and estimating the dimension of complicated data manifolds are open questions in data science. For that reason, we are only able to conduct the estimation to those simple structures, such as linear functions and manifolds produced by them.

In what follows, we conduct the estimation to the first eight dense layers of StyleGAN2 on FFHQ. Those layers have sufficiently simple structures for PCA, but also play vital role in the network as discussed in (Karras et al., 2019b; a).

Each of the eight dense layers, denoted as Dense0, Dense1,...,Dense7, is composed of a linear transformation
\[ l_i(x) = W_i x + b_i \]
and a LeakyRelu activation
\[ \text{act}(x) = \begin{cases} x & x > 0 \\ 0.2x & x \leq 0 \end{cases} \] (S1)

Thus the function coupling that maps input to the output of the \( k \)-th layer is
\[ \text{Dense}_k(x) = \text{act}(l_k) \circ \cdots \circ \text{act}(l_0). \] (S2)

We conduct PCA to the Jacobian of each dense layer. The results are reported in Fig. S1. We compute the number of components that have strength larger than 1% of that of the maximum component. The results are reported in Tab. S1. We can find that as layer gets deeper, the component strengths gather towards the left components, which means the number of valid components gets smaller and the rank of corresponding Jacobian gets smaller. This means the rank drop indicated by Lemma 2 does happen in practice.

In fact, we find that each weight matrix \( W_i \) has a low rank structure. We conduct PCA to the weight matrix \( W_0, \ldots, W_7 \). The results are reported in Fig. S2. We compute the number of components that have strength larger than 1% of that of the maximum component. The results are reported in Tab. S2. We can find that the valid components of each weight matrix are around 450, which means each of the weight matrix will drop around 60 dimensions of the inputs.

We then look into the intrinsic dimension of the feature spaces produced by those dense layers. For each dense layer, we sample 51200 random inputs \( z \) from \( \mathcal{N}(0,1) \) and feed them to the layer to produce 51200 points in the corresponding feature space. We then conduct PCA to those inputs.
For each $x \in \mathcal{N}$, there exists a smooth chart $(V, \psi)$ around $x$ and a smooth chart $(U, \phi)$ around $f(z)$ such that $f(U) \subset V$, and

$$\psi \circ f \circ \phi^{-1}(a_1, \ldots, a_n) = (a_1, \ldots, a_r, 0, \ldots, 0).$$

Let $r = \max_{z \in \mathcal{N}} J_z f$, and $\mathcal{R} = \{z \in \mathcal{N} : rank_z f = r\}$. Then Lemma D says that $f(\mathcal{R})$ and $\mathcal{N}$ have intrinsic dimension $r$, and $\mathcal{N} \setminus \mathcal{R}$ belongs to the set of critical points. By Lemma B, $f(\mathcal{N} \setminus \mathcal{R})$ is a zero measure set. Thus for almost every point $x \in \mathcal{M}$, its preimage has rank $r$. \hfill \Box

**Table S3.** Estimated intrinsic dimension of intermediate feature space.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Intrinsic dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense0</td>
<td>512</td>
</tr>
<tr>
<td>Dense1</td>
<td>508</td>
</tr>
<tr>
<td>Dense2</td>
<td>422</td>
</tr>
<tr>
<td>Dense3</td>
<td>289</td>
</tr>
<tr>
<td>Dense4</td>
<td>207</td>
</tr>
<tr>
<td>Dense5</td>
<td>148</td>
</tr>
<tr>
<td>Dense6</td>
<td>105</td>
</tr>
<tr>
<td>Dense7</td>
<td>91</td>
</tr>
</tbody>
</table>

By the chain rule of differential (Rudin et al., 1964), we have

$$rank(J(f^1 \circ f^2)) = rank(Jf^1 Jf^2).$$

(S5)

Recall that for any two matrices $A$ and $B$

$$rank(AB) \leq \min\{rankA, rankB\}.$$  

(S6)

We then have Lemma 2.

**D.1. Lemma 2**

By the chain rule of differential (Rudin et al., 1964), we have

$$rank(J(f^1 \circ f^2)) = rank(Jf^1 Jf^2).$$

(S5)

Recall that for any two matrices $A$ and $B$

$$rank(AB) \leq \min\{rankA, rankB\}.$$  

(S6)

We then have Lemma 2.

**D.2. Theorem 1**

**Proof.** Denote the dimensions of $G(Z)$ and $\mathcal{X}$ as $d_g$ and $d_x$, respectively. There are two possible cases for $G$: $d_g$ is lower than $d_x$, or $d_g$ is higher than or equal to $d_x$.

For the first case, a direct consequence is that, for almost all points in $\mathcal{X}$, there are no pre-images under $G$. This means that for an arbitrary point $x \in \mathcal{X}$, the possibility of $G^{-1}(x) = \emptyset$ is 1, as $\{x \in \mathcal{X} : G^{-1}(x) \neq \emptyset\} \subset G(Z) \cap \mathcal{X}$, which is a zero measure set in $\mathcal{X}$. This also implies that the generator is unable to perform inversion. Another consequence is that, the generated distribution $P_g$ can never get aligned with real data distribution $P_r$. Namely, the distance between $P_r$ and $P_g$ cannot be zero for arbitrary distance metrics. For the KL divergence, the distance will even approach infinity.

Specifically, let $p_r$ and $p_g$ be the densities of $P_r$ and $P_g$ respectively. For the Jensen-Shannon divergence, we have

$$D_{JS}(P_r, P_g) = \frac{1}{2} \int \log \left( \frac{2p_r}{p_r + p_g} \right) p_r + \frac{1}{2} \int \log \left( \frac{2p_g}{p_r + p_g} \right) p_g.$$ 

(S7)

As the support of $p_g$ is a zero measure set of the support of $p_r$, we have

$$\int \log \left( \frac{2p_r}{p_r + p_g} \right) dP_r = \int_{p_g = 0} \log \left( \frac{2p_r}{p_r + p_g} \right) dP_r$$

$$= \int_{p_g = 0} \log \left( \frac{2p_r}{p_g} \right) dP_r = \log 2 \int p_g = \log 2,$$

(S8)

and

$$\int \log \left( \frac{2p_g}{p_r + p_g} \right) dP_g \geq 0.$$  

(S9)

Thus $D_{JS} \geq \frac{\log 2}{2}$.

For the second case, $d_g \geq d_x > d_z$. We simply show that a Lipschitz-continuous function cannot map zero measure set into positive measure set. Specifically, the image of low dimensional space of a Lipschitz-continuous function has...
measure zero. Thus if \( d_g \geq d_e \), \( G \) cannot be Lipschitz. As Lipschitz constant is the supremum of gradient norm, we then prove our theorem.

Now we prove our claim.

Suppose that \( f : \mathbb{R}^n \to \mathbb{R}^m, n < m \), and \( f \) is Lipschitz with Lipschitz constant \( L \). We show that \( f(\mathbb{R}^n) \) has measure zero in \( \mathbb{R}^m \). As \( \mathbb{R}^n \) is a zero measure set in \( \mathbb{R}^m \), by the Kirszbraun theorem (Deimling, 2010), \( f \) has an extension to a Lipschitz function of the same Lipschitz constant on \( \mathbb{R}^m \).

For convenience, we still denote the extension as \( f \). Then the problem reduces to proving that \( f \) maps zero measure set to zero measure set. For every \( \epsilon > 0 \), we can find countable union of balls \( \{B_k\}_k \) of radius \( r_k \) such that \( \mathbb{R}^n \subset \bigcup_k B_k \) and \( \sum_k m(B_k) < \epsilon \) in \( \mathbb{R}^m \), where \( m(\cdot) \) is the Lebesgue measure in \( \mathbb{R}^m \). But \( f(B_k) \) is contained in a ball with radius \( L r_k \). Thus we have \( m(f(\mathbb{R}^n)) \leq L^m \sum_k m(B_k) < L^m \epsilon \), which means that it is a zero measure set in \( \mathbb{R}^m \). For the mapping between manifolds, using the chart system can turn it into the case we analyze above, which completes our proof.

We want to remind the readers that, even if the generator suits one of the cases in Theorem 1, the other case can still occur. For example, \( G \) could succeed in capturing the distribution of certain parts of the real data, while it may fail in the other parts. Then for the pre-image of those successfully captured data, the generator will not have finite Lipschitz constant.

**D.3. Theorems 2 & 3**

*Proof.* Theorems 2 & 3 are classical conclusions in Riemannian manifold. We refer readers to section 5.5 of the book written by Petersen et al. (2006) for detailed proofs and illustration.

**D.4. Theorem 4**

*Proof.* Theorem 4 is a natural extension of the Heine-Borel theorem (Rudin et al., 1964).

**Lemma E** (Heine-Borel Theorem). For any compact set \( \mathcal{M} \), if \( \{U_i\}_{i \in I} \) is an open cover of \( \mathcal{M} \), (that is, for each \( i \in I \), \( U_i \) is an open set, and \( \mathcal{M} \subset \bigcup_{i \in I} U_i \)), then there exist finite many elements \( U_{i_1}, \ldots, U_{i_k} \) of \( \{U_i\}_{i \in I} \), such that \( \mathcal{M} \subset \bigcup_{1 \leq j \leq k} U_{i_j} \).

Let the skeleton set be all points of \( \mathcal{M} \). Then the representative pairs in Theorems 2 & 3 define an open cover of \( \mathcal{M} \). By Lemma E, we can pick finite many points of skeleton set \( \mu_1, \ldots, \mu_k \), such that their representative pairs also define an open cover of \( \mathcal{M} \).

For each local neighborhood of representative pairs, it is easy to see that the error is \( o(r) \) by Taylor expansion of Theorem 3.

**E.1. Theorem 5**

*Proof.*

\[
E[||g(x) - g(y)||_2] \leq \|\mu(x) - \mu(y)\|_2 \\
+ E[||\sigma(x)e - \sigma(y)d||_2] \\
\leq L_\mu ||x - y||_2 + 2C||\sigma||_\infty \\
\leq L_\mu ||x - y||_2 + o(1),
\]

where \( C \) is a constant related to the dimension of the image space of \( \sigma \) and \( L_\mu \) is Lipschitz constant of \( \mu \).

**F. Why Gaussian distribution?**

We first introduce the notion of fuzzy equivalence relations (Zhang & Zhang, 2005; Murali, 1989).

**Definition 1.** A t-norm is a function \( T : [0,1] \times [0,1] \to [0,1] \) which satisfies the following properties:

1. Commutativity: \( T(a,b) = T(b,a) \).
2. Monotonicity: \( T(a, b) \leq T(c, d) \), if \( a \leq c \) and \( b \leq d \).
3. Associativity: \( T(a, T(b, c)) = T(T(a, b), c) \).
4. The number 1 acts as identity element: \( T(a, 1) = a \).

**Definition 2.** Given a t-norm \( T \), a \( T \)-equivalence relation on a set \( X \) is a fuzzy relation \( E \) on \( X \) and satisfies the following conditions:

1. \( E(x, x) = 1, \forall x \in X \) (Reflexivity).
2. \( E(x, y) = E(y, x), \forall x, y \in X \) (Symmetry).
3. \( T(E(x, y), E(y, z)) \leq E(x, z) \forall x, y, z \in X \) (\( T \)-transitivity).

Then it is easy to check that \( T(x, y) = xy \) is a t-norm, and \( E(x, y) = e^{-d(x,y)} \) is a \( T \)-equivalence for any distance metric \( d \) on \( X \), as

\[
T(E(x, y), E(y, z)) = e^{-(d(x,y)+d(y,z))} \leq e^{-d(x,z)} = E(x, z).
\]
We illustrate the generator architectures of StyleGAN2. All experiments are carried out by TensorFlow 1.14 and which consists of 126 thousand church images of various works (GANs). The dataset consists of 70,000 high-quality images, and then quickly falls into mode collapse after training on 963 thousand images, and then quickly falls into mode collapse after that.

G. Datasets

**FFHQ**  Flickr-Faces-HQ (FFHQ) (Karras et al., 2019a) is a high-quality image dataset of human faces, originally created as a benchmark data for generative adversarial networks (GANs). The dataset consists of 70,000 high-quality PNG images and contains considerable variations in terms of age, pose, expression, hair style, ethnicity and image backgrounds. It also covers diverse accessories such as eyeglasses, sunglasses, hats, etc.

**LSUN-Church and Cat-Selected**  LSUN-Church is the church outdoor category of LSUN dataset (Yu et al., 2015), which consists of 126 thousand church images of various styles. Cat-Selected contains 100 thousand cat images selected by ranking algorithm (Zhou et al., 2004) from the LSUN cat category. The plausibility of using PageRank to rank data was analyzed in (Zhou et al., 2004). We also used the algorithm presented in (Zhao & Tang, 2009) to construct the graph from the cat data.

**CIFAR-10**  The CIFAR-10 dataset (Krizhevsky et al., 2009) consists of 60,000 images of size 32x32. There are all 10 classes and 6000 images per class. There are 50,000 training images and 10,000 test images.

H. Implementation details

H.1. Models

We illustrate the generator architectures of StyleGAN2 based methods in Figure S4. For all those models, the discriminators share the same architecture as the original StyleGAN2. The generator architecture of DCGAN based methods are illustrated in Figure S5. For all those models, the discriminators share the same architecture as the original DCGAN.

I. Experiment environment

All experiments are carried out by TensorFlow 1.14 and Python 3.6 with CUDA Version 10.2 and NVIDIA-SMI 440.64.00. We basically build our code upon the framework of NVIDIA official StyleGAN2 code, which is available at https://github.com/NVlabs/stylegan2. We use a variety of servers to run the experiments as reported in Table S4.

J. Image encoding and GAN inversion

From a mathematical perspective, a well behaved generator should be easily invertible. In the last section, we have shown that our method is well conditioned, which implies that it could be easily invertible. We adopt the methods in Image2StyleGAN (Abdal et al., 2019) to perform GAN inversion and compare the mean square error and perceptual loss on a manually collected dataset of 20 images. The source code of inversion is from Luxemburg (2020). The images are shown in Figure S6 and the quantitative results are provided in Table S5. For our RNI methods, we further optimize the $\alpha$ parameter in Eq. 7 in section 4.3, which fine-tunes the local geometries of the network to suit the new images that might not be settled in the model. Considering that $\alpha$ is limited to $[0, 1]$, we use $\alpha_t = \frac{\alpha^*}{(\alpha^*)^t + (1 - \alpha^*)^t}$ to replace the original $\alpha$ and optimize $t$. The initial value of $t$ is set to 1.0 and $\alpha^*$ is constant with the same value as $\alpha$ in the converged RNI models.

During the experiments, we find that the StyleGAN2 model is prone to work well for full-face, non-blocking human face images. For this type of images, we observe comparable performance for all the GAN architectures. We think that this is because those images are closed to the ‘mean’ face of FFHQ dataset (Karras et al., 2019a), thus easy to learn. We adopt the methods shown that our method is well conditioned, which implies that it could be easily invertible. In the last section, we have shown that our method is well conditioned, which implies that it could be easily invertible. We adopt the methods in Image2StyleGAN (Abdal et al., 2019) to perform GAN inversion and compare the mean square error and perceptual loss on a manually collected dataset of 20 images. The source code of inversion is from Luxemburg (2020). The images are shown in Figure S6 and the quantitative results are provided in Table S5. For our RNI methods, we further optimize the $\alpha$ parameter in Eq. 7 in section 4.3, which fine-tunes the local geometries of the network to suit the new images that might not be settled in the model. Considering that $\alpha$ is limited to $[0, 1]$, we use $\alpha_t = \frac{\alpha^*}{(\alpha^*)^t + (1 - \alpha^*)^t}$ to replace the original $\alpha$ and optimize $t$. The initial value of $t$ is set to 1.0 and $\alpha^*$ is constant with the same value as $\alpha$ in the converged RNI models.

K. Ablation study of RNI

In Tab. S6, we perform the ablation study of the proposed RNI method on the FFHQ dataset. We test 5 different choices of RNI implementation and compare their FID and PPL scores after convergence.

1. No normalization: in this setting we remove the normalization of $\tilde{\mu}$ in Eq. (14), and use the unnormalized $\tilde{\mu}$ to replace $s$ in the following equations. The network comes to a minimum FID of 23.77 after training on 1323 thousand images, and then quickly falls into mode collapse after that.

2. No stabilization: in this setting we remove the stabilization technique in Eq. (16). The network comes to a minimum FID of 50.27 after training on 963 thousand images, and then quickly falls to mode collapse after
Table S4. GPU environments for all experiments in this work.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleGAN2 based GAN model training</td>
<td>8 NVIDIA Tesla V100-SXM2-16GB GPUs (DGX-1 station)</td>
</tr>
<tr>
<td>DCGAN based GAN model training</td>
<td>4 TITAN Xp GPUs</td>
</tr>
<tr>
<td>Metrics measurement</td>
<td>8 GeForce GTX 1080Ti GPUs</td>
</tr>
<tr>
<td>GAN inversion</td>
<td>1 TITAN Xp GPU</td>
</tr>
</tbody>
</table>

Table S5. Image inversion metrics for different StyleGAN2 based models. The perceptual loss is the mean square distance of VGG16 features between the original and projected images as in Abdal et al. (2019)

<table>
<thead>
<tr>
<th>GAN arch</th>
<th>Overall</th>
<th>Hard Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE (↓)</td>
<td>Perceptual Loss (↓)</td>
</tr>
<tr>
<td>Bald StyleGAN2</td>
<td>1.34</td>
<td>5.42</td>
</tr>
<tr>
<td>StyleGAN2 + ENI</td>
<td>1.24</td>
<td>4.86</td>
</tr>
<tr>
<td>StyleGAN2-NoPathReg + RNI</td>
<td>1.24</td>
<td>5.11</td>
</tr>
<tr>
<td>StyleGAN2 + RNI</td>
<td>1.13</td>
<td>4.52</td>
</tr>
</tbody>
</table>

3. No decomposition: in this setting we remove the decomposition in Eq. (15). The network successfully converges, but admits a large PPL score.

4. CNN: in this setting we use a convolutional neural network to replace the procedure that we get $\sigma$ in section 4.3. Namely, we take $\sigma = \text{CNN}(\mu)$. The network successfully converges, but admits a very large FID score.

The zero PPL scores in ‘No normalization’ and ‘No stabilization’ suggest that the generator output is invariant to small perturbations, which means mode collapse. We can find that the stabilization and normalization in the RNI implementation in section 4.3 is necessary for the network to avoid numerical instability and mode collapse. The implementation of RNI method reaches the best performance in PPL score and comparable performance against the ‘no decomposition’ method in FID score. As analyzed in StyleGAN (Karras et al., 2019a) and StyleGAN2 (Karras et al., 2019b), for high fidelity images, PPL is more convincing than the FID score in measuring the synthesis quality. Therefore, the RNI implementation is the best among these methods.

Table S6. Ablation study of different noise injection methods on FFHQ. The zero values of PPL scores in the first two methods suggest mode collapse.

<table>
<thead>
<tr>
<th>Method</th>
<th>FID</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>No normalization</td>
<td>628.94</td>
<td>0</td>
</tr>
<tr>
<td>No stabilization</td>
<td>184.30</td>
<td>0</td>
</tr>
<tr>
<td>No decomposition</td>
<td>6.48</td>
<td>18.78</td>
</tr>
<tr>
<td>CNN</td>
<td>22.54</td>
<td>14.53</td>
</tr>
<tr>
<td>RNI</td>
<td>7.31</td>
<td>13.05</td>
</tr>
</tbody>
</table>

References


Figure S1. Strengths of the principal components of the Jacobian matrix of each dense layer. The sum of strengths of all components is normalized to 1. We can find that as layer gets deeper, the component strengths gather towards the left components, which means the number of valid components gets smaller and the rank of corresponding Jacobian gets smaller.
Figure S2. Strengths of the principal components of the weight matrices. The sum of strengths of all components is normalized to 1. We can find that the valid components of each weight matrix is around 450, which means each of the weight matrices will drop around 60 dimensions of the inputs.
Figure S3. Strengths of the principal components of the output space of each dense layer. The sum of strengths of all components is normalized to 1. We can find that as layer gets deeper, the component strengths gather towards the left components, which means the number of valid components gets smaller and the intrinsic dimension of corresponding feature space gets smaller.
Figure S4. Generator architectures of StyleGAN2 based models. (a) The generator of bald StyleGAN2. (b) The generator of StyleGAN2. (c) The generator of StyleGAN2 + RNI and StyleGAN2-NoPathReg + RNI. ‘Mod’ and ‘Demod’ denote the weight demodulation method proposed in section 2.2 of StyleGAN2 (Karras et al., 2019b). A denotes a learned affine transformation from the intermediate latent space $W$. 
Figure S5. Generator architecture of DCGAN based models. (a) The generator of DCGAN. (b) The generator of DCGAN + Additive Noise. (c) The generator of DCGAN + RNI.
Figure S6. Manually collected 20 images for GAN inversion.