

# Spider GAN: Leveraging Friendly Neighbors to Accelerate GAN Training

Siddarth Asokan\*

Robert Bosch Center for Cyber-Physical Systems  
Indian Institute of Science  
Bengaluru - 50012, India  
siddartha@iisc.ac.in

Chandra Sekhar Seelamantula

Department of Electrical Engineering  
Indian Institute of Science  
Bengaluru - 50012, India  
css@iisc.ac.in

## Abstract

Training Generative adversarial networks (GANs) stably is a challenging task. The generator in GANs transform noise vectors, typically Gaussian distributed, into realistic data such as images. In this paper, we propose a novel approach for training GANs with images as inputs, but without enforcing any pairwise constraints. The intuition is that images are more structured than noise, which the generator can leverage to learn a more robust transformation. The process can be made efficient by identifying closely related datasets, or a “friendly neighborhood” of the target distribution, inspiring the moniker, Spider GAN. To define friendly neighborhoods leveraging proximity between datasets, we propose a new measure called the signed inception distance (SID), inspired by the polyharmonic kernel. We show that the Spider GAN formulation results in faster convergence, as the generator can discover correspondence even between seemingly unrelated datasets, for instance, between Tiny-ImageNet and CelebA faces. Further, we demonstrate cascading Spider GAN, where the output distribution from a pre-trained GAN generator is used as the input to the subsequent network. Effectively, transporting one distribution to another in a cascaded fashion until the target is learnt – a new flavor of transfer learning. We demonstrate the efficacy of the Spider approach on DCGAN, conditional GAN, PG-GAN, StyleGAN2 and StyleGAN3. The proposed approach achieves state-of-the-art Fréchet inception distance (FID) values, with one-fifth of the training iterations, in comparison to their baseline counterparts on high-resolution small datasets such as MetFaces, Ukiyo-E Faces and AFHQ-Cats.

## 1. Introduction

Generative adversarial networks (GANs) [1] are designed to model the underlying distribution of a target dataset (with

underlying distribution  $p_d$ ) through a *min-max* optimization between the generator  $G$  and the discriminator  $D$  networks. The generator transforms an input  $z \sim p_z$ , typically Gaussian or uniform distributed, into a generated sample  $G(z) \sim p_g$ . The discriminator is trained to classify samples drawn from  $p_g$  or  $p_d$  as real or fake. The optimal generator is the one that outputs images that confuse the discriminator.

**Inputs to the GAN generator:** The input distribution plays a definitive role in the quality of GAN output. Low-dimensional latent vectors have been shown to help disentangle the representations and control features of the target being learnt [2, 3]. Prior work on optimizing the latent distribution in GANs has been motivated by the need to improve the quality of interpolated images. Several works have considered replacing the Gaussian prior with Gaussian mixtures, Gamma, non-parametric distributions, etc [4–9]. Alternatively, the GAN generator can be trained with the latent-space distribution of the target dataset, as learnt by variational autoencoders [10, 11]. However, such approaches are not in conformity with the low-dimensional manifold structure of real data. Khayatkhoei *et al.* [12] attributed the poor quality of the interpolates to the disjoint structure of data distribution in high-dimensions, which motivates the need for an informed choice of the input distribution.

**GANs and image-to-image translation:** GANs that accept images as input fall under the umbrella of *image translation*. Here, the task is to modify particular features of an image, either within domain (style transfer) or across domains (domain adaptation). Examples for in-domain translation include changing aspects of face images, such as the expression, gender, accessories, etc. [13–15], or modifying the illumination or seasonal characteristics of natural scenes [16]. On the other hand, *domain adaptation tasks* aim at transforming the image from one style to another. Common applications include simulation to real-world translation [17–20], or translating images across styles of artwork [21–23]. While the supervised Pix2Pix framework [22] originally proposed training GANs with pairs of images drawn from the source and target domains, semi-supervised and unsupervised ex-

\*Siddarth Asokan is funded by the Qualcomm Innovation Fellowship, and the Robert Bosch Center for Cyber-Physical Systems Ph.D. Fellowship.

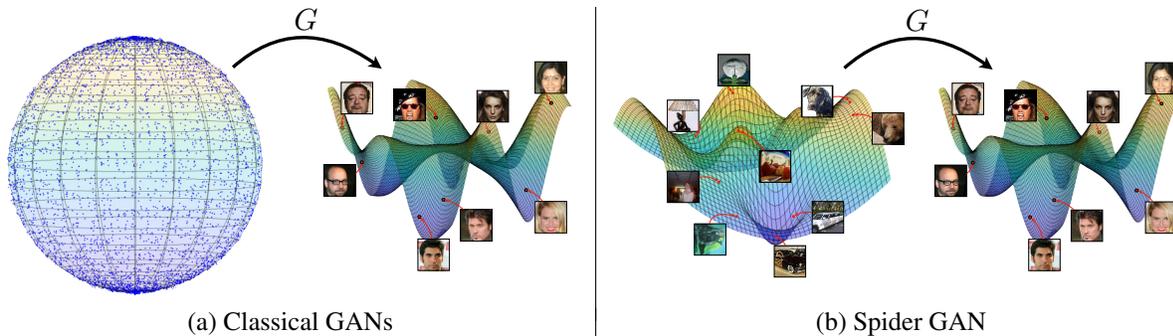


Figure 1. (Color online) A comparison of design philosophies of the standard GANs and Spider GAN. (a) A prototypical GAN transforms high-dimensional Gaussian data, which is concentrated at the surface of hyperspheres in  $n$ -D, into an image distribution comprising a union of low-dimensional manifolds embedded in a higher-dimensional space. (b) The Spider GAN generator aims to learn a simpler transformation between two closely related data manifolds in an unconstrained manner, thereby accelerating convergence.

tensions [23–28] tackle the problem in an unpaired setting, and introduce modifications such as cycle-consistency or the addition of regularization functionals to the GAN loss to maintain a measure of consistency between images. Existing domain-adaptation GANs [29, 30] enforce cross-domain consistency to retain visual similarity. Ultimately, these approaches rely on enforcing some form of coupling between the source and the target via feature-space mapping.

## 2. The Proposed Approach: Spider GAN

We propose the Spider GAN formulation motivated by the low-dimensional disconnected manifold structure of data [12, 31–33]. Spider GANs lie at the cross-roads between classical GANs and image-translation GANs. As opposed to optimizing the latent parametric prior, we hypothesize that providing the generator with closely related image source datasets, (dubbed the *friendly neighborhood*, leading to the moniker *Spider GAN*) will result in superior convergence of the GAN. Unlike image translation tasks, the Spider GAN generator is agnostic to individual input-image features, and is allowed to *discover* implicit structure in the mapping from the source distribution to the target. Figure 1 depicts the design philosophy of Spider GAN juxtaposed with the classical GAN training approach.

The choice of the *input dataset* affects the generator’s ability to learn a stable and accurate mapping. Intuitively, if the GAN has to be trained to learn the distribution of *street view house numbers* (SVHN) [34], the MNIST [35] dataset proves to be a better initialization of the input space than standard densities such as the uniform or Gaussian. It is a well known result that, for a given mean and variance, the Gaussian has maximum entropy, while for a given support (say,  $[-1, 1]$  when training with re-normalized images), the uniform distribution has maximum entropy [36]. However, image datasets are highly structured, and possess lower entropy [37]. Therefore, one could interpret the generative modeling of images using GANs as effectively one of entropy minimization [13].

We argue that choosing a low entropy input distribution that is structurally closer to the target would lead to a more efficient generator transformation, thereby accelerating the training process. Existing image-translation approaches aim to maintain semantic information, for example, translating a specific instance of the digit ‘2’ in the MNIST dataset to the SVHN style. However, the Spider GAN formulation neither enforces nor requires such constraints. Rather, it allows for an implicit structure in the source dataset to be used to learn the target efficiently. It is entirely possible for the *Trouser* class in Fashion-MNIST [38] to map to the digit ‘1’ in MNIST due to structural similarity. Thus, the scope of Spider GAN is much wider than image translation.

### 2.1. Our Contributions

In Section 3, we discuss the central focus in Spider GANs: defining what constitutes a *friendly neighborhood*. Preliminary experiments suggest that, while the well known Fréchet inception distance (FID) [39] and kernel inception distance (KID) [40] are able to capture visual similarity, they are unable to quantify the diversity of samples in the underlying manifold. We therefore propose a novel distance measure to evaluate the input to GANs, one that is motivated by electrostatic potential fields and charge neutralization between the (positively charged) target data samples and (negatively charged) generator samples [41, 42], named *signed inception distance* (SID) (Section 3.1). An implementation of SID atop the Clean-FID [43] backbone is available at <https://github.com/DarthSid95/clean-sid>. We identify *friendly neighborhoods* for multiple classes of standard image datasets such as MNIST, Fashion MNIST, SVHN, CIFAR-10 [44], Tiny-ImageNet [45], LSUN-Churches [46], CelebA [47], and Ukiyo-E Faces [48]. We present experimental validation on training the *Spider* variant of DC-GAN [49] (Section 4) and show that it results in up to 30% improvement in terms of FID, KID and cumulative SID of the converged models. The *Spider* framework is

lightweight and can be extended to any GAN architecture, which we demonstrate via class-conditional learning with the *Spider* variant of auxiliary classifier GANs (ACGANs) [50] (Section 4). The source code for Spider GANs built atop the DCGAN architecture are available at <https://github.com/DarthSid95/SpiderDCGAN>. We also present a novel approach to transfer learning using Spider GANs by feeding the output distribution of a pre-trained generator to the input of the subsequent stage (Section 5). Considering progressively growing GAN (PGGAN) [51] and StyleGAN [52–54] architectures, we show that the corresponding *Spider* variants achieve competitive FID scores in one-fifth of the training iterations on FFHQ [14] and AFHQ-Cats [30], while achieving state-of-the-art FID on high-resolution small-sized datasets such as Ukiyo-E Faces and MetFaces [53] (Section 5.1). The source code for implementing Spider StyleGANs is available at <https://github.com/DarthSid95/SpiderStyleGAN>.

## 2.2. Related Works

The choice of the input distribution in GANs determines the quality of images generated by feeding the generator interpolated points, which in turn is determined by the probability of the interpolated points lying on the manifold. High-dimensional Gaussian random vectors are concentrated on the surface of a hypersphere (*Gaussian annulus theorem* [55]), akin to a *soap bubble*, resulting in interpolated points that are less likely to lie on the manifold. Alternatives such as the Gamma [6] or Cauchy [7] prior result in superior performance over interpolated points, while Singh *et al.* [9] derive a non-parametric prior that minimized the divergence between the input and the midpoint distributions.

A well known result in high-dimensional data analysis is that structured datasets are embedded in a low-dimensional manifold with an *intrinsic dimensionality* ( $n_{\mathcal{D}}$ ) significantly lower than the ambient dimensionality  $n$  [37]. For instance, in MNIST,  $n = 784$ , while  $n_{\mathcal{D}} \approx 12$  [56]. Feng *et al.* [57] showed that the mismatch between  $n_{\mathcal{D}}$  of the generator input and its output adversely affects performance. Although in practice, estimating  $n_{\mathcal{D}}$  may not always be possible [12, 56, 58], these results justify picking input distributions that are structurally similar to the target. In instance-conditioned GANs [59], the target data is modeled as clusters on the data manifold to improve learning.

The philosophy of cascading Spider GAN generators runs in parallel to input optimization in transfer learning with GANs, such as Mine GAN [60] where *mining* networks are implemented that transform the input distribution of the GAN nonlinearly to learn the target samples better. Keras *et al.* [53] showed that transfer learning improves the performance of GANs on small datasets, and observed empirically that transferring weights from models trained on visually diverse data lead to better performance of the target model.

## 3. Where is the Friendly Neighborhood?

We now consider various distance measures between datasets that can be used to identify the friendly neighborhood/source dataset in Spider GANs. While the most direct approach is to compare the intrinsic dimensions of the manifolds, such approaches are either computationally intensive [61], or do not scale with sample size [56, 58]. We observed that the friendly neighbors detected by such approach did not correlate well experimentally, and therefore, defer discussions on such methods to Appendix A.

Based on the approach advocated by Wang *et al.* [62] to identify pre-trained GAN networks for transfer learning, we initially considered FID and KID to identify *friendly neighbors*. We use the FID to measure the distance between the source (generator input) and the target data distributions. A source that has a lower FID is closer to the target and will serve as a better input to the generator. The first four columns of Table 1 present FID scores between the standard datasets we consider in this paper. The **first**, **second** and **third** *friendly neighbors* (color coded) of a target dataset are the source datasets with the lowest three FIDs. As observed from Table 1, a limitation is that the FID of a dataset with itself is not always zero, which is counterintuitive for a distance measure. In cases such as CIFAR-10 or Tiny-ImageNet, this is indicative of the variability in the dataset, and in Ukiyo-E Faces, this is due to limited availability of data samples, which has been shown to negatively affect FID estimation [40, 63]. FID satisfies *reciprocity*, *i.e.*, it identifies datasets as being mutually close to each other, such as CIFAR-10 and Tiny-ImageNet. However, preliminary experiments on training Spider GAN using FID to identify friendly neighbors showed that the relative diversity between datasets is not captured. Given a source, learning a less diverse target distribution is easier (cf. Section 4 and Appendix D.2). These issues are similar to the observations made by Keras *et al.* [53] in the context of weight transfer. This can be understood via an example — fitting a multimodal target Gaussian having 10 modes would be easier with a 20-component source distribution than a 5-component one.

### 3.1. The Signed Inception Distance (SID)

Given the limitations of FID discussed above, we propose a novel signed distance for measuring the proximity between two distributions. The distance is “signed” in the sense that it can also take negative values. Further, it is not symmetric. The distance is also practical to compute because it is expressed in terms of the samples drawn from the distributions. The proposed distance draws inspiration from the improved precision-recall scores of GANs [64] and the potential-field interpretation in Coulomb GANs [41] and Poly-LSGAN [42]. Consider batches of samples drawn from distributions  $\mu_p$  and  $\mu_q$ , given by  $\mathcal{D}_p = \{\tilde{c}_i\}_{i=1}^{N_p}$  and

Table 1. A comparison of FID and CSID<sub>m</sub> between popular training datasets for  $m = \lfloor \frac{n}{2} \rfloor$ . The rows represent the source and the columns correspond to the target. The **first**, **second** and **third friendly neighbors** of the target are the sources with the three lowest FID, or lowest positive CSID values, respectively. CSID is superior to FID, as it assigns negative values to sources that are less diverse than the target. MNIST and Fashion-MNIST are shown in gray to denote scenarios where grayscale images are not valid sources for the color-image targets.

Source \ Target	FID (Source, Target)				CSID <sub>m</sub> (Source    Target)			
	MNIST	CIFAR-10	TinyImageNet	Ukiyo-E	MNIST	CIFAR-10	TinyImageNet	Ukiyo-E
MNIST	1.2491	258.246	264.250	398.280	0.1863	29.298	9.436	201.550
F-MNIST	<b>176.813</b>	188.367	197.057	387.049	<b>162.962</b>	19.051	-2.5571	191.010
SVHN	<b>236.707</b>	<b>168.615</b>	<b>189.133</b>	372.444	212.473	<b>34.534</b>	<b>21.668</b>	214.507
CIFAR-10	<b>259.045</b>	5.0724	<b>64.3941</b>	303.694	221.337	-0.1487	-7.109	198.991
TinyImageNet	264.309	<b>64.0312</b>	6.4854	<b>257.078</b>	230.916	<b>12.892</b>	0.6743	<b>197.447</b>
CelebA	360.773	303.490	<b>250.735</b>	<b>301.108</b>	<b>204.794</b>	<b>23.685</b>	<b>8.829</b>	<b>184.170</b>
Ukiyo-E	396.791	300.511	254.102	5.9137	250.226	39.793	<b>18.727</b>	0.5494
Church	350.708	<b>294.982</b>	254.991	<b>267.638</b>	<b>212.452</b>	-4.655	-23.115	<b>198.750</b>

$\mathcal{D}_q = \{c_j\}_{j=1}^{N_q}$ , respectively. Given a test vector  $\mathbf{x} \in \mathbb{R}^n$ , consider the Coulomb GAN discriminator [41]:

$$f(\mathbf{x}) = \frac{1}{N_p} \sum_{\substack{i=1 \\ \tilde{c}_i \sim \mu_p}}^{N_p} \Phi(\mathbf{x}, \tilde{c}_i) - \frac{1}{N_q} \sum_{\substack{j=1 \\ c_j \sim \mu_q}}^{N_q} \Phi(\mathbf{x}, c_j), \quad (1)$$

where  $\Phi$  is the polyharmonic kernel [42, 65]:

$$\Phi(\mathbf{x}, \mathbf{y}) = \kappa_{m,n} \begin{cases} \|\mathbf{x} - \mathbf{y}\|^{2m-n}, & \text{if } 2m-n < 0 \\ & \text{or } n \text{ is odd,} \\ \|\mathbf{x} - \mathbf{y}\|^{2m-n} \ln(\|\mathbf{x} - \mathbf{y}\|), & \text{if } 2m-n \geq 0 \\ & \text{and } n \text{ is even,} \end{cases}$$

and  $\kappa_{m,n}$  is a positive constant, given the order  $m$  and dimensionality  $n$ . The higher-order generalization gives us more flexibility and numerical stability in computation. We use  $m \approx \lfloor \frac{n}{2} \rfloor$  as a stable choice, while ablation studies on choosing  $m$  are given in Appendix B.4

From the perspective of electrostatics, for  $\mu_p = p_d$  and  $\mu_q = p_d$ ,  $f(\mathbf{x})$  in Equation (1) treats the target data as negative charges, and generator samples as positive charges. The quality of  $\mu_p$  in approximating/matching  $\mu_q$  is measurable by computing the effect of the net charge present in any chosen volume around the target  $\mu_q$  on a test charge  $\mathbf{x}$ . Consider a hypercube  $\mathcal{C}_{q,r}$  of side length  $r$ , centered around  $\mu_q$  with test charges  $\{\mathbf{x}_\ell\}_{\ell=1}^{M_x}$ ,  $\mathbf{x}_\ell \in \mathcal{C}_{q,r}$ . To analyze the average behavior of target and generated samples in  $\mathcal{C}_{q,r}$ , we draw  $\mathbf{x}_\ell$  uniformly within  $\mathcal{C}_{q,r}$ . We consider  $N_p = N_q = N$  for simplicity. We now define the *signed distance* of  $\mu_p$  from  $\mu_q$  as the negative of  $f(\mathbf{x})$ , summed over a uniform sampling of points over  $\mathcal{C}_{q,r}$ , i.e.  $\text{SD}_{m,r}(\mu_p || \mu_q)$  is given by:

$$\frac{1}{NM_x} \sum_{\substack{\ell=1 \\ \tilde{\mathbf{x}}_\ell \in \mathcal{C}_{q,r}}}^{M_x} \left( \sum_{\substack{j=1 \\ c_j \sim \mu_q}}^N \Phi(\mathbf{x}_\ell, c_j) - \sum_{\substack{i=1 \\ \tilde{c}_i \sim \mu_p}}^N \Phi(\mathbf{x}_\ell, \tilde{c}_i) \right). \quad (2)$$

Similar to the improved precision and recall (IPR) metrics,  $\text{SD}_{m,r}(\mu_p || \mu_q)$  is asymmetrical, i.e.,  $\text{SD}_{m,r}(\mu_p || \mu_q) \neq \text{SD}_{m,r}(\mu_q || \mu_p)$ . When  $\text{SD}_{m,r}(\mu_p || \mu_q) < 0$ , on the average,

samples from  $\mu_q$  are relative more spread out than those drawn from  $\mu_p$  with respect to  $\mathcal{C}_{q,r}$ , and vice versa. When  $\mu_p = \mu_q$ , we have  $\text{SD}_{m,r}(\mu_p || \mu_q) \approx 0$ . Illustrations of these three scenarios are provided in Appendix B.3.

In practice, similar to the standard GAN metrics, the computation of SD can be made practical and efficient on higher-resolution images by evaluating the measure on the feature-space of the images learnt by the pre-trained InceptionV3 [66] network mapping  $\psi(c)$ . This results in the *signed inception distance*  $\text{SID}_{m,r}(\mu_p || \mu_q) =$  given by:

$$\frac{1}{NM_x} \sum_{\substack{\ell=1 \\ \mathbf{x}_\ell \in \mathcal{C}'_{q,r}}}^{M_x} \left( \sum_{\substack{j=1 \\ c_j \sim \mu_q}}^N \Phi(\mathbf{x}_\ell, \psi(c_j)) - \sum_{\substack{i=1 \\ \tilde{c}_i \sim \mu_p}}^N \Phi(\mathbf{x}_\ell, \psi(\tilde{c}_i)) \right), \quad (3)$$

where  $\mathcal{C}'_{q,r}$  denotes the hypercube of side  $r$  centered on the transformed distribution  $\psi(\mu_q)$ . To begin with, we find  $\sigma_q = \max\{\text{diag}(\Sigma_q)\}$ , where in turn,  $\Sigma_q$  is the covariance matrix of the samples in  $\mathcal{D}_q$ . We define the hypercube  $\mathcal{C}'_{q,r}$  as having side  $r = \sigma_q$  along each dimension and centered around the mean of  $\mu_q$ . To compare two datasets, we plot  $\text{SID}_{m,r}(\mu_p || \mu_q)$  as a function of  $r \in [\sigma_q, 100\sigma_q]$  varying  $r$  in steps of 0.5. SID comparison figures for a few representative target datasets are given in Figure 2. We observe that, when two datasets are closely related, SID is close to zero even for small  $r$ . Datasets with lower diversity than the target have a negative SID, and vice versa. In order to quantify SID as a single number (akin to FID and KID) we consider SID, accumulated over all radii  $r$  (the cumulative SID or CSID, for short) given by:  $\text{CSID}_m = \sum_r \text{SID}_{m,r}$ . The last four columns of Table 1 presents CSID for  $m = \lfloor \frac{n}{2} \rfloor$  for the various datasets considered. We observe that CSID is highly correlated with FID when the source is more diverse than the target, while it is able to single out sources that lack diversity, which FID cannot. These results quantitatively verify the empirical *closeness* observed when transfer-learning across datasets [53]. Additional experiments and ablation studies on SID are given in Appendices A and B.

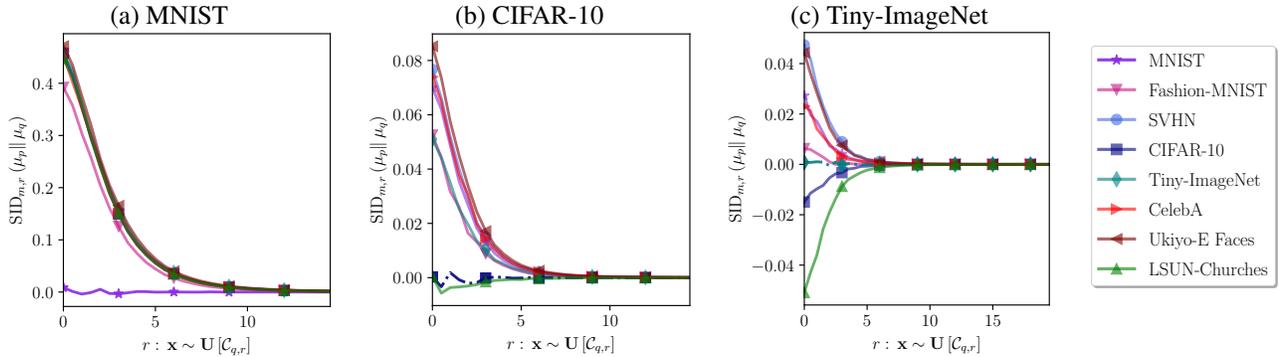


Figure 2. (Color online)  $SID_{m,r}$  as a function of the hyper-cube length  $r$ . We observe that Fashion-MNIST is the closest to MNIST, while Tiny-ImageNet and SVHN are closest to CIFAR-10. Fashion-MNIST and CelebA are friendly neighbors of Tiny-ImageNet.

**Picking the Friendliest Neighbor:** While the various approaches to compare datasets generally suggest different *friendly neighbors*, we observe that the overall trend is consistent across the measures. For example, Tiny-ImageNet and CelebA are consistently friendly neighbors to multiple datasets. We show in Sections 4 and 5 that choosing these datasets as the input indeed improves the GAN training algorithm. Both the proposed SID, and baseline FID/KID measures are relative in that they can only measure closeness between provided candidate datasets. Incorporating domain-awareness aids in the selection of appropriate input datasets between which SID can be compared. For example, all metrics identify Fashion-MNIST as a friendly neighbor when compared against color-image targets, although, as expected, the performance is sub-par in practice (cf. Section 4). One would therefore discard MNIST and Fashion-MNIST when identifying friendly neighbors of color-image datasets. Although SID is superior to FID and KID in identifying less diverse source datasets, no single approach can always find the best dataset yet in all real-world scenarios. A pragmatic strategy is to compute various similarity measures between the target and visually/structurally similar datasets, and identify the closest one by voting.

## 4. Experimental Validation

To demonstrate the Spider GAN philosophy, we train *Spider DCGAN* on MNIST, CIFAR-10, and  $256 \times 256$  Ukiyo-E Faces datasets using the input datasets mentioned in Section 3. While encoder-decoder architectures akin to image-to-image translation GANs could also be employed, their performance does not scale with image dimensionality. Detailed ablation experiments are provided in Appendix D.1. The second aspect is the limited stochasticity of the input dataset, when its cardinality is lower than that of the target. In these scenarios, the generator would attempt to learn one-to-many mappings between images, thereby not modeling the target entirely. For Spider DCGAN variants, the source data is resized to  $16 \times 16$ , vectorized, and provided as input. Based on preliminary experimentation (cf. Ap-

pendix D.2.1), to improve the input dataset diversity, we consider a Gaussian mixture centered around the samples of the source dataset formed by adding zero-mean Gaussian noise with variance  $\sigma \approx 0.25$  to each source image. An alternative solution, based on pre-trained generators is presented in Section 5. We consider the Wasserstein GAN [67] loss with a one-sided gradient penalty [68]. The training parameters are described in Appendix C. In addition to FID and KID, we compare the GAN variants in terms of the cumulative SID ( $CSID_m$ ) for  $m = \lfloor \frac{n}{2} \rfloor$  to demonstrate the viability of evaluating GANs with the proposed SID metric.

**Results:** We demonstrate the ability of Spider GAN to leverage the structure present in the source dataset. From the input-output pairs given in Figure 3, we observe that, although trained in an unconstrained manner, the generator learns structurally motivated mappings. In the case when learning MNIST images with Fashion-MNIST as input, the generator has learnt to cluster similar classes, such as *Trousers* and the *1* class, or the *Shoes* class and digit 2, which serendipitously are also visually similar. Even in scenarios where such pairwise similarity is not present, as in the case of generating Ukiyo-E Faces from CelebA or CIFAR-10, Spider GAN leverages implicit/latent structure to accelerate the generator convergence. Figure 4 presents FID as a function of iterations for each learning task for a few select target datasets. Spider GAN variants with *friendly neighborhood* inputs outperform the baseline models with parametric noise inputs, while also converging faster (up to an order in the case of MNIST). Table 2 presents the FID of the best-case models. In choosing a *friendly neighbor*, a poorly related dataset results in worse performance than the baselines, while a closely related input results in FID improvements of about 30%. The poor performance of Fashion MNIST as a friendly neighbor to CIFAR-10 and Ukiyo-E faces datasets corroborate the observations made in Section 3. We observe that  $CSID_m$  is generally in agreement with the performance indicated by FID/KID, making it a viable alternative in evaluating GANs. Experiments on remaining source-target combinations are provided in Appendix D.2.

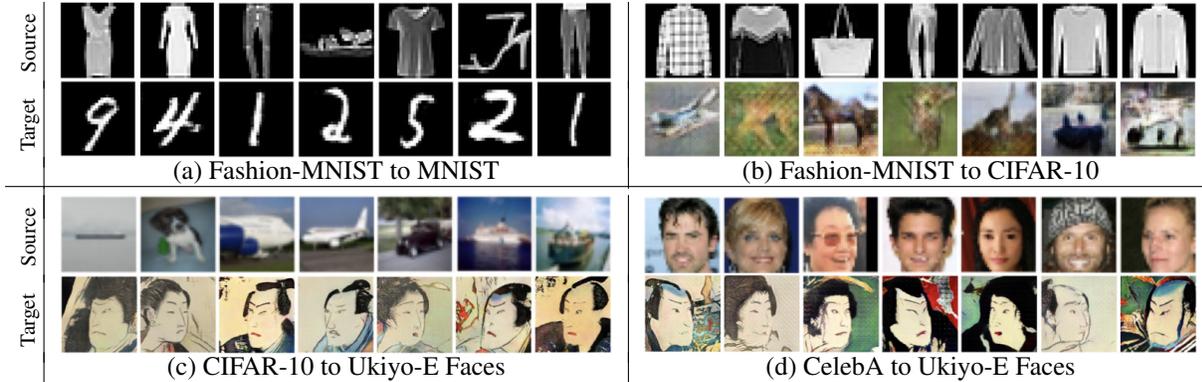


Figure 3. (Color online) Figures depicting the implicit structure learnt by Spider GAN when transforming the source to the target. The network learns both visual, and implicit correspondences across datasets. For example, the *Trouser* class in Fashion-MNIST maps to the digit *1* in MNIST, while the implicit structure is leveraged by the generator in transforming either CIFAR-10 or CelebA to Ukiyo-E Faces. A poor choice of the input distribution, for instance selecting Fashion-MNIST as the friendly neighbor of CIFAR-10, results in suboptimal learning.

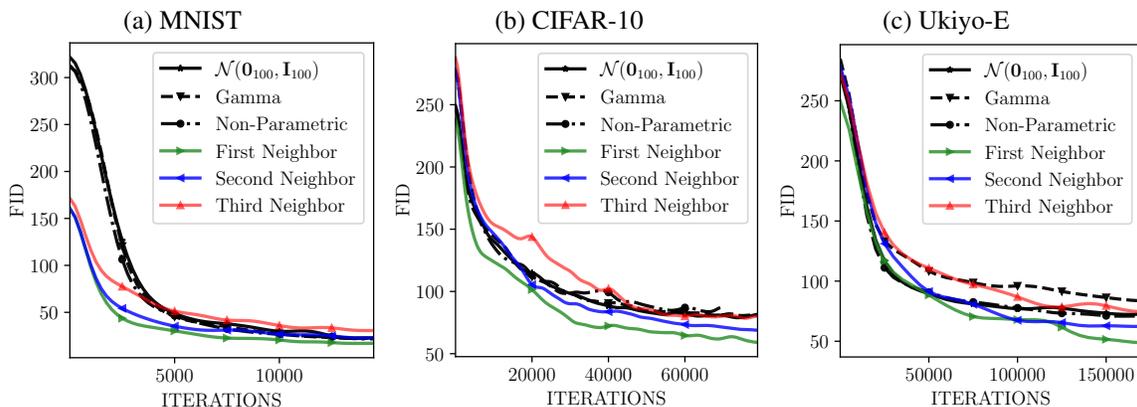


Figure 4. (Color online) FID versus iterations for training baseline and Spider GAN with the **first**, **second** and **third** friendly neighbors (color coded) identified by CSID (cf. Table 1). Using the friendliest neighbor results in the best (lowest) FID scores. On MNIST, Spider GAN variants saturate to a lower FID in an order of iterations faster than the baselines.

Table 2. Comparison of FID, KID and the proposed  $CSID_m$  (with  $m = \lfloor \frac{n}{2} \rfloor$ ) for the Spider DCGAN and baseline variants on MNIST, CIFAR-10, and Ukiyo-E Faces datasets. The first ( $\dagger$ ), second ( $\ddagger$ ) and third ( $\star$ ) friendly neighbors (cf. CSID; Table 1) of the target are marked for cross-referencing against the **first**, **second** and **third** best FID/KID/ $CSID_m$  scores. Spider DCGAN, with friendly neighborhood input datasets outperform the baseline parametric and non-parametric priors, while a bad choice for the input results in a poorer performance.

Input Distribution		MNIST			CIFAR10			Ukiyo-E Faces		
		FID	KID	$CSID_m$	FID	KID	$CSID_m$	FID	KID	$CSID_m$
Baselines	Gaussian [49] ( $\mathbb{R}^{100}$ )	21.49	0.0139	21.31	71.84	0.0619	19.90	62.26	0.0535	23.10
	Gamma [6] ( $\mathbb{R}^{100}$ )	21.15	0.0133	19.44	72.66	0.0483	19.87	70.02	0.0495	30.59
	Non-Parametric [9] ( $\mathbb{R}^{100}$ )	20.94	0.0137	20.78	74.90	0.0530	19.45	65.36	<u>0.0421</u>	25.40
	Gaussian ( $\mathbb{R}^{H \times W \times C}$ )	42.44	0.0354	32.20	73.00	0.0504	21.99	70.96	0.0501	35.30
Spider DCGAN	MNIST	–	–	–	71.70	0.0535	21.83	68.87	0.0438	33.13
	Fashion MNIST	$\dagger$ 16.80	$\dagger$ 0.0103	$\dagger$ 12.44	77.86	0.0550	28.85	72.431	0.0455	36.21
	SVHN	27.17	0.0205	17.23	$\star$ 64.30	$\star$ 0.0451	$\star$ 18.44	70.13	0.0482	25.06
	CIFAR-10	29.22	0.0220	24.96	–	–	–	70.55	0.0530	24.12
	TinyImageNet	32.66	0.0244	36.90	$\dagger$ 58.82	$\dagger$ 0.0305	$\dagger$ 14.02	$\ddagger$ 61.91	$\ddagger$ 0.0463	$\ddagger$ 21.07
	CelebA	$\ddagger$ 20.55	$\ddagger$ 0.0144	$\ddagger$ 15.74	$\ddagger$ 60.09	$\ddagger$ 0.0434	$\ddagger$ 17.68	$\dagger$ 54.09	$\dagger$ 0.0408	$\dagger$ 20.12
	Ukiyo-E	<u>18.72</u>	<u>0.0122</u>	19.35	67.80	0.0463	19.90	–	–	–
	LSUN-Churches	$\star$ 30.67	$\star$ 0.0228	$\star$ 30.61	61.46	<u>0.0365</u>	19.82	$\star$ 66.26	$\star$ 0.0496	$\star$ 25.21

**Extension to Class-conditional Learning:** As a proof of concept, we developed the *Spider* counterpart to the auxiliary classifier GAN (ACGAN) [50], entitled Spider ACGAN. Here, the discriminator predicts the class label of the input in addition to the *real* versus *fake* classification. We consider two variants of the generator, one without class information, and the other with the class label provided as a fully-connected embedding to the input layer. While Spider ACGAN without generator embeddings is superior to the baseline Spider GAN in learning class-level consistency, mixing between the classes is not eliminated entirely. However, with the inclusion of class embeddings in the generator, the disentanglement of classes can be achieved in Spider ACGAN. Additional details are provided in Appendix D.3. Extensions of Spider GAN to larger class-conditional GAN models such as BigGAN [69], and scenarios involving mismatch between the number of classes in the input and output datasets, are promising directions for future research.

## 5. Cascading Spider GANs

The DCGAN architecture employed in Section 4 does not scale well for generating high-resolution images. While training with image datasets has proven to improve the generated image quality, the improvement is accompanied by an additional memory requirement. While inference with standard GANs requires inputs drawn purely from random number generators, Spider DCGAN would require storing an additional dataset as input. To overcome this limitation, we propose a novel cascading approach, where the output distribution of a publicly available pre-trained generator is used as the input distribution to subsequent Spider GAN stages. The benefits are four-fold: First, the memory requirement is significantly lower (by an order or two), as only the weights of an input-stage generator network are required to be stored. Second, the issue of limited stochasticity in the input distribution is overcome, as infinitely many unique input samples can be drawn. Third, the network can be cascaded across architectures and styles, *i.e.*, one could employ a BigGAN input stage (trained on CIFAR-10, for example) to train a Spider StyleGAN network on ImageNet, or vice versa. *Essentially, no pre-trained GAN gets left behind.* Lastly, the cascaded Spider GANs can be coupled with existing transfer learning approaches to further improve the generator performance on small datasets [53].

### 5.1. Spider Variants of PGGAN and StyleGAN

We consider training the *Spider* variants of StyleGAN2 [52] and progressively growing GAN (PGGAN) [51] on small datasets, specifically the 1024-MetFaces and 1024-Ukiyo-E Faces datasets, and high-resolution FFHQ. We consider input from pre-trained GAN generators trained on the following two distributions (a) Tiny-ImageNet, based on CSID<sub>m</sub>, that suggest that it is a *friendly neighbor* to the

targets; and (b) AFHQ-Dogs, which possesses structural similarity to the face datasets. The experimental setup is provided in Appendix D.4, while evaluation metrics are described in Appendix C.2. To maintain consistency with the reported scores for state-of-the-art baselines models, we report only FID/KID here, and defer comparisons on CSID<sub>m</sub> to Appendix D.5. To isolate and assess the performance improvements introduced by the Spider GAN framework, we do not incorporate any augmentation or weight transfer [53]. Table 3 shows the FID values obtained by the baselines and their *Spider* variants. Spider PGGAN performs on par with the baseline StyleGAN2 in terms of FID. Spider StyleGAN2 achieves state-of-the-art FID on both Ukiyo-E and MetFaces.

To incorporate transfer learning techniques, we consider (a) learning FFHQ considering StyleGAN with adaptive discriminator augmentation (ADA) [53]; and (b) learning AFHQ-Cats considering both ADA and weight transfer [53]. Spider StyleGAN2-ADA achieves FID scores on par with the state of the art, outperforming improved sampling techniques such as Polarity-StyleGAN2 [71] and MaGNET-StyleGAN2 [72]. While StyleGAN-XL achieves marginally superior FID, it does so at the cost of a three-fold increase in network complexity [70]. The FID and KID scores, and training configurations are described in Tables 4-5. Spider StyleGAN2-ADA and Spider StyleGAN3 achieve competitive FID scores with a mere one-fifth of the training iterations. The Spider StyleGAN3 model with weight transfer achieves a state-of-the-art FID of 3.07 on AFHQ-Cats, in a fourth of the training iterations as StyleGAN3 with weight transfer. Additional results are provided in Appendix D.5.

### 5.2. Understanding the Spider GAN Generator

The idea of learning an optimal transformation between a pair of distributions has been explored in the context of optimal transport in *Schrödinger bridge* diffusion models [73–76]. The *closer* the two distributions are, the easier it is to learn a transport map between them. Spider GANs leverage underlying similarity, not necessarily visual, between datasets to improve generator learning. Similar discrepancies between visual features and those learnt by networks have been observed in ImageNet [77] object classification [78]. To shed more light on this intuition, consider a scenario where both the input and target datasets in Spider DCGAN are the same, with or without random noise perturbation. As expected, the generator learns an identity mapping, reproducing the input image at the output (cf. Appendix D.2.5).

**Input Dataset Bias:** Owing to the unpaired nature of training, Spider GANs do not enforce image-level structure to learn pairwise transformations. Therefore, the diversity of the source dataset (such as racial or gender diversity) does not affect the diversity in the learnt distribution. Experiments on Spider DCGAN with varying levels of class-imbalance in the input dataset validate this claim (cf. Appendix D.2.3).

Table 3. A comparison of the FID and KID values achieved by the PGGAN and StyleGAN2 baselines and their *Spider* variants, when trained on small datasets. A \* indicates scores computed on publicly available pre-trained models using the Clean-FID library [43]. Spider StyleGAN2 achieves state-of-the-art FID and KID scores, while Spider PGGAN achieves performance comparable with the baseline StyleGAN methods.

Architecture	Input	Ukiyo-E Faces		MetFaces	
		FID	KID	FID	KID
PGGAN [51]	Gaussian	69.03	0.0762	85.74	0.0123
Spider PGGAN (Ours)	TinyImageNet	57.63	0.0161	45.32	0.0063
StyleGAN2* [52]	Gaussian	56.74	0.0159	65.74	0.0350
StyleGAN2-ADA* [53]	Gaussian	26.74	0.0109	18.75	<b>0.0023</b>
Spider StyleGAN2 (Ours)	TinyImageNet	<b>20.44</b>	<b>0.0059</b>	<b>15.60</b>	0.0026
Spider StyleGAN2 (Ours)	AFHQ-Dogs	32.59	0.0269	29.82	0.0019

Table 4. A comparison of StyleGAN2-ADA and StyleGAN3 variants in terms of FID, on learning FFHQ. A † indicates a reported score. Spider StyleGAN2-ADA performs on par with the state-of-the-art StyleGAN-XL (three fold higher network complexity) [70], and outperforms variants with customized sampling techniques [71, 72].

Architecture	Input	FID
StyleGAN-XL [70]	Gaussian	<b>2.02</b> †
Polarity-StyleGAN2 [71]	Gaussian	2.57†
MaGNET-StyleGAN2 [72]	Gaussian	2.66†
StyleGAN2-ADA [53]	Gaussian	2.70†
Spider StyleGAN2-ADA (Ours)	TinyImageNet	<b>2.45</b>
Spider StyleGAN2-ADA (Ours)	AFHQ-Dogs	3.07
StyleGAN3-T [54]	Gaussian	2.79†
Spider StyleGAN3-T (Ours)	TinyImageNet	2.86

Table 5. A comparison of the FID and KID values achieved by the StyleGAN baselines and their *Spider* variants, when trained on the the AFHQ-Cats dataset, considering various training configurations. A \* indicates a score reported in the Clean-FID library [43]. † Karras *et al.* only report FID on the combined AFHQv2 dataset consisting of images from the *Dogs*, *Cats*, and *Wild-Animals* classes. Spider StyleGAN2-ADA and Spider StyleGAN3 achieve FID and KID scores competitive with the baselines in a mere one-fifth of the training iterations, while Spider StyleGAN3 with weight transfer achieves state-of-the-art FID on AFHQ in one-fourth of the training iterations.

Architecture	Weight Transfer	Input Distribution	Training steps	FID	KID ( $\times 10^{-3}$ )
StyleGAN2-ADA [53]	–	Gaussian	25000	5.13*	1.54*
StyleGAN3-T [54]	–	Gaussian	25000	4.04†	–
Spider StyleGAN3-T (Ours)	–	AFHQ-Dogs	5000	6.29	1.64
StyleGAN2-ADA [53]	FFHQ	Gaussian	5000	3.55	0.35
Spider StyleGAN2-ADA (Ours)	FFHQ	Tiny-ImageNet	1000	3.91	1.23
StyleGAN2-ADA [53]	AFHQ-Dogs	Gaussian	5000	3.47*	0.37*
Spider StyleGAN2-ADA (Ours)	AFHQ-Dogs	Tiny-ImageNet	1500	<b>3.07</b>	<b>0.29</b>
Spider StyleGAN3-T (Ours)	AFHQ-Dogs	Tiny-ImageNet	1000	3.86	1.01

**Input-space Interpolation:** Lastly, to understand the representations learnt by Spider GANs, we consider input-space interpolation. Unlike classical GANs, where the input noise vectors are the only source of control, in cascaded Spider GANs, interpolation can be carried out at two levels. Interpolating linearly between the noise inputs to the pre-trained GAN result in a set of interpolations of the intermediate image. Transforming these images through the Spider StyleGAN generator results in greater diversity in the output images, with sharper transitions between images. This is expected as interpolating on the Gaussian manifold is known to result in discontinuities in the generated images [6, 7]. Alternatively, for fine-grained tuning, linear interpolations of the intermediate input images can be carried out, resulting in smoother transitions in the output images. Images demonstrating this behavior are provided in Appendix D.5.1. Qualitative experiments on input-space interpolation in Spider DC-GAN and additional images are provided in Appendix D.2.2. These results indicate that stacking multiple Spider GAN stages yields varying levels of fineness in controlling features in the generated images.

## 6. Conclusions

We introduced the Spider GAN formulation, where we provide the GAN generator with an input dataset of samples from a closely related neighborhood of the target. Unlike image-translation GANs, there are no pairwise or cycle-consistency requirements in Spider GAN, and the trained generator learns a transformation from the underlying latent data distribution to the target data. While the *best* input dataset is a problem-specific design choice, we proposed approaches to identify promising friendly neighbors. We proposed a novel signed inception distance, which measures the relative diversity between two datasets. Experimental validation showed that Spider GANs, trained with closely related datasets, outperform baseline GANs with parametric input distributions, achieving state-of-the-art FID on Ukiyo-E Faces, MetFaces, FFHQ and AFHQ-Cats.

While we focused on adaptive augmentation and weight transfer, incorporating other transfer learning approaches [29, 60, 79] is a promising direction for future research. One could also explore extensions to vector quantized GANs [80, 81] or high-resolution class-conditional GANs [69, 82].

## References

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014.
- [2] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning GAN for pose-invariant face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1283–1292, 2017.
- [3] E. Agustsson, A. Sage, R. Timofte, and L. V. Gool, "Optimal transport maps for distribution preserving operations on latent spaces of generative models," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.
- [4] S. Gurumurthy, R. K. Sarvadevabhatla, and R. V. Babu, "DeLiGAN: Generative adversarial networks for diverse and limited data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, July 2017.
- [5] T. White, "Sampling generative networks," *arXiv preprints, arXiv:1609.04468*, 2016.
- [6] Y. Kilcher, A. Lucchi, and T. Hofmann, "Semantic interpolation in implicit models," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [7] D. Leśniak, I. Sieradzki, and I. Podolak, "Distribution-interpolation trade off in generative models," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.
- [8] M. Kuznetsov, D. Polykovskiy, D. P. Vetrov, and A. Zhebrak, "A prior of a googol gaussians: a tensor ring induced prior for generative models," in *Advances in Neural Information Processing Systems 32*, 2019.
- [9] R. Singh, P. Turaga, S. Jayasuriya, R. Garg, and M. Braun, "Non-parametric priors for generative adversarial networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 5838–5847, June 2019.
- [10] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, Jun 2016.
- [11] G. Parmar, D. Li, K. Lee, and Z. Tu, "Dual contradistinctive generative autoencoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.
- [12] M. Khayatkhoei, M. K. Singh, and A. Elgammal, "Disconnected manifold learning for generative adversarial networks," in *Advances in Neural Information Processing Systems 31*, pp. 7343–7353, 2018.
- [13] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems 29*, pp. 2180–2188, 2016.
- [14] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.
- [15] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of GANs for semantic face editing," in *Proceeding of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9240–9249, 2020.
- [16] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in Neural Information Processing Systems 30*, pp. 465–476, 2017.
- [17] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, July 2017.
- [18] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2962–2971, 2017.
- [19] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4500–4509, 2018.
- [20] B. Khurana, S. R. Dash, A. Bhatia, A. Mahapatra, H. Singh, and K. Kulkarni, "SemIE: Semantically-aware image extrapolation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14900–14909, October 2021.
- [21] S. Hicsönmez, N. Samet, E. Akbas, and P. Duygulu, "GANILLA: Generative adversarial networks for image to illustration translation," *Image and Vision Computing*, vol. 95, p. 103886, Feb, 2020.
- [22] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprints, arXiv:1611.07004*, 2018.
- [23] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of International Conference on Computer Vision*, 2017.
- [24] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems 30*, pp. 700–708, 2017.
- [25] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *Proceedings of the International Conference on Computer Vision*, Oct. 2017.
- [26] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 1989–1998, July 2018.
- [27] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European Conference on Computer Vision*, 2018.

- [28] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision*, Sep. 2018.
- [29] U. Ojha, Y. Li, C. Lu, A. A. Efros, Y. J. Lee, E. Shechtman, and R. Zhang, "Few-shot image generation via cross-domain correspondence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [30] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [31] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," *Journal of the American Mathematical Society*, vol. 29, pp. 983–1049, 2016.
- [32] J. Liang, J. Yang, H.-Y. Lee, K. Wang, and M.-H. Yang, "SubGAN: An unsupervised generative model via subspaces," in *Proceedings of the European Conference on Computer Vision*, September 2018.
- [33] U. Tanielian, T. Issenhuth, E. Dohmatob, and J. Mary, "Learning disconnected manifolds: a no GANs land," *arXiv preprints, arXiv:2006.04596*, 2020.
- [34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- [35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [36] J. A. Thomas and T. M. Cover, *Elements of Information Theory*. John Wiley and Sons, Ltd, 2005.
- [37] J. L. Kelley, *General Topology*. Courier Dover Publications, Inc., 2017.
- [38] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint, arXiv:1708.07747*, Aug. 2017.
- [39] 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," *arXiv preprints, arXiv:1706.08500*, 2018.
- [40] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [41] T. Unterthiner, B. Nessler, C. Seward, G. Klambauer, M. Heusel, H. Ramsauer, and S. Hochreiter, "Coulomb GANs: Provably optimal Nash equilibria via potential fields," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [42] S. Asokan and C. S. Seelamantula, "LSGANs with gradient regularizers are smooth high-dimensional interpolators," in *Proceedings of the "First Workshop on Interpolation and Beyond" at NeurIPS*, 2022.
- [43] G. Parmar, R. Zhang, and J.-Y. Zhu, "On buggy resizing libraries and surprising subtleties in FID calculation," *arXiv preprint, arXiv:2104.11222*, vol. abs/2104.11222, April 2021.
- [44] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Master's thesis, University of Toronto*, 2009.
- [45] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," 2015.
- [46] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprints, arXiv:1506.03365*, 2016.
- [47] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision*, 2015.
- [48] J. N. M. Pinkney and D. Adler, "Resolution dependent GAN interpolation for controllable image synthesis between domains," *arXiv preprint, arXiv:2010.05334*, Oct. 2020.
- [49] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proceedings of the 4th International Conference on Learning Representations*, 2016.
- [50] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
- [51] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [52] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [53] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Advances in Neural Information Processing Systems* 33, 2020.
- [54] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Advances in Neural Information Processing Systems*, June 2021.
- [55] A. Blum, J. Hopcroft, and R. Kannan, *Foundations of Data Science*. Cambridge University Press, 2013.
- [56] M. Hein and J.-Y. Audibert, "Intrinsic dimensionality estimation of submanifolds in  $R^d$ ," in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 289–296, 2005.
- [57] R. Feng, Z. Lin, J. Zhu, D. Zhao, J. Zhou, and Z.-J. Zha, "Uncertainty principles of encoding GANs," in *Proceedings of the 38th International Conference on Machine Learning*, pp. 3240–3251, Jul 2021.
- [58] E. Facco, M. d'Errico, A. Rodriguez, and A. Laio, "Estimating the intrinsic dimension of datasets by a minimal neighborhood information," *Scientific Reports*, vol. 7, Sep. 2017.

- [59] A. Casanova, M. Careil, J. Verbeek, M. Drozdal, and A. R. Soriano, "Instance-conditioned GAN," in *Advances in Neural Information Processing Systems 34*, pp. 27517–27529, 2021.
- [60] Y. Wang, A. Gonzalez-Garcia, D. Berga, L. Herranz, F. S. Khan, and J. Weijer, "MineGAN: Effective knowledge transfer from GANs to target domains with few images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.
- [61] F. Camastra and A. Staiano, "Intrinsic dimension estimation: Advances and open problems," *Information Sciences*, vol. 328, pp. 26–41, 2016.
- [62] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring GANs: Generating images from limited data," in *Proceedings of the European Conference on Computer Vision*, pp. 220–236, 2018.
- [63] 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*, pp. 5228–5237, 2018.
- [64] 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*, 2019.
- [65] N. Aronszajn, T. Creese, and L. Lipkin, *Polyharmonic Functions*. Oxford: Clarendon, 1983.
- [66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint, arXiv:1512.00567*, Dec. 2015.
- [67] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 214–223, 2017.
- [68] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 3481–3490, 2018.
- [69] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprints, arXiv:1809.11096*, Sep. 2018.
- [70] A. Sauer, K. Schwarz, and A. Geiger, "StyleGAN-XL: Scaling StyleGAN to large diverse datasets," *arXiv.org*, vol. abs/2201.00273, 2022.
- [71] A. I. Humayun, R. Balestrieri, and R. Baraniuk, "Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.
- [72] A. I. Humayun, R. Balestrieri, and R. Baraniuk, "MaGNET: Uniform sampling from deep generative network manifolds without retraining," in *International Conference on Learning Representations (ICLR)*, 2022.
- [73] J. Ho, A. Jain, and P. Abbeel, "Denosing diffusion probabilistic models," *preprint arxiv:2006.11239*, 2020.
- [74] F. Vargas, P. Thodoroff, A. Lamacraft, and N. Lawrence, "Solving Schrödinger bridges via maximum likelihood," *Entropy*, vol. 23, 2021.
- [75] V. D. Bortoli, J. Thornton, J. Heng, and A. Doucet, "Diffusion Schrödinger bridge with applications to score-based generative modeling," in *Advances in Neural Information Processing Systems*, 2021.
- [76] T. Chen, G.-H. Liu, and E. Theodorou, "Likelihood training of Schrödinger bridge using forward-backward SDEs theory," in *International Conference on Learning Representations*, 2022.
- [77] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.
- [78] T. Fel, I. Felipe, D. Linsley, and T. Serre, "Harmonizing the object recognition strategies of deep neural networks with humans," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [79] Y. Li, R. Zhang, J. Lu, and E. Shechtman, "Few-shot image generation with elastic weight consolidation," in *Advances in Neural Information Processing Systems*, 2020.
- [80] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.
- [81] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu, "Vector-quantized image modeling with improved VQGAN," in *Proceedings of the 10th International Conference on Learning Representations*, 2022.
- [82] M. Kang, W. Shim, M. Cho, and J. Park, "Rebooting ACGAN: Auxiliary classifier GANs with stable training," in *Advances in Neural Information Processing Systems 34*, 2021.
- [83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [84] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint, arXiv:1603.04467*, Mar. 2016.
- [85] T. Liang, "How well generative adversarial networks learn distributions," *Journal of Machine Learning Research*, vol. 22, no. 228, pp. 1–41, 2021.
- [86] N. Schreuder, V.-E. Brunel, and A. Dalalyan, "Statistical guarantees for generative models without domination," in *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, vol. 132, pp. 1051–1071, Mar 2021.

# Spider GAN: Leveraging Friendly Neighbors to Accelerate GAN Training

## Supporting Document

Siddarth Asokan  
Robert Bosch Center for Cyber-Physical Systems  
Indian Institute of Science  
Bengaluru - 50012, India  
siddartha@iisc.ac.in

Chandra Sekhar Seelamantula  
Department of Electrical Engineering  
Indian Institute of Science  
Bengaluru - 50012, India  
css@iisc.ac.in

## Part I

# Appendix

## Table of Contents

---

<b>A Baselines for Identifying the Friendly Neighborhood</b>	<b>2</b>
A.1. The Davis-Kahan Theorem	2
A.2. Comparison of Approaches for Identifying the Friendly Neighborhood	3
<b>B The Signed Inception Distance (SID)</b>	<b>6</b>
B.1. Asymptotic Behavior of the Signed Distance	6
B.2. SID Computation	6
B.3. Experiments on Gaussian Data	7
B.4. Evaluating GANs with SID	8
<b>C Implementation Details</b>	<b>14</b>
C.1. Experimental Setup	14
C.2. Evaluation Metrics	14
C.3. Computational Resources	15
<b>D Additional Experimentation on Spider GAN</b>	<b>16</b>
D.1. Exploring Generator Architectures	16
D.2. Additional Experiments on Spider DCGAN	17
D.3. Class-conditional Spider GAN	34
D.4. Additional Experiments on Spider PGGAN	34
D.5. Additional Experimental on Spider StyleGAN	34
<b>E GitHub Repository and Code Release</b>	<b>61</b>

---

## Overview of the Supplementary Material

The Supplementary Material comprises this *Supporting Document* of appendices, the source codes of this project, consisting of the implementations of various Spider GAN variants and SID metric, and animations corresponding to (a) Evaluating the signed distance on Gaussain data; and (ii) Interpolation in the input, and intermediate stages of Spider StyleGAN. The appendices contains the additional discussions on identifying the friendly neighborhood in Spider GANs, ablation studies on SID, implementation details, and additional experiments on the Spider GAN variants considered in the *Main Manuscript*. The citations to prior art in the *Supporting Document* are with respect to the references listed herein.

### A. Baselines for Identifying the Friendly Neighborhood

Approaches that compute the intrinsic dimensionality  $n_{\mathcal{D}}$  of a dataset are either computationally intensive [1] or do not scale with sample size [2, 3]. Campadelli *et al.* [4] presented a survey of various nearest-neighbor and maximum-likelihood estimators of  $n_{\mathcal{D}}$  for low-dimensional datasets. A well-known approach for computing  $n_{\mathcal{D}}$  is provided in the Davis-Kahan  $\sin \Theta$  theorem [5], which provides an upper bound on the distance between two subspaces in terms of the eigen-gap between them. A practically implementable version [6] is based on the sample covariance matrix of the two datasets and their eigenvalues. Along a parallel vertical, multiple works have derived convergence guarantees on the GAN training algorithms, given  $n_{\mathcal{D}}$  [7, 8, 9]. We now discuss the Davis-Kahan  $\sin \Theta$  theorem, and compare its performance against the FID, KID and  $\text{CSID}_m$  approaches in terms of the friendliest neighbors picked by them.

#### A.1. The Davis-Kahan Theorem

The Davis-Kahan  $\sin \Theta$  Theorem [5] upper-bounds the distance between subspaces in terms of the eigen-gap between them. Let  $\Sigma_p, \Sigma_q \in \mathbb{R}^{n \times n}$  denote the sample covariance matrices of two datasets  $\mathcal{D}_p$  and  $\mathcal{D}_q$ , respectively, with  $\lambda_{1_p} \geq \lambda_{2_p} \geq \dots \geq \lambda_{n_p}$  and  $\lambda_{1_q} \geq \lambda_{2_q} \geq \dots \geq \lambda_{n_q}$  denoting their respective eigenvalues in order. Consider  $1 \leq r \leq s \leq n$ , and define  $d := s - r + 1$ ,  $\lambda_0 := \infty$  and  $\lambda_{n+1} := -\infty$ . Consider the subspaces  $\mathcal{V}_p = \text{span} \{v_r^p, v_{r+1}^p, \dots, v_s^p\}$  and  $\mathcal{V}_q = \text{span} \{v_r^q, v_{r+1}^q, \dots, v_s^q\}$  that are spanned by the eigenvectors of  $\Sigma_p$  and  $\Sigma_q$ , respectively. The Davis-Kahan  $\sin \Theta$  theorem bounds the distance between the two subspaces  $\mathcal{V}_p$  and  $\mathcal{V}_q$  as follows:

$$\|\sin \Theta(\mathcal{V}_p, \mathcal{V}_q)\|_F \leq \frac{\|\Sigma_p - \Sigma_q\|_F}{\delta}, \quad (1)$$

$$\text{where } \delta = \inf \left\{ |\hat{\lambda} - \lambda| : \lambda \in [\lambda_s^q, \lambda_r^q], \hat{\lambda} \in \left( -\infty, \hat{\lambda}_{s-1}^p \right] \cup \left[ \hat{\lambda}_{r+1}^p, \infty \right) \right\}.$$

As noted by Yu *et al.* [6], evaluating the infimum among all pairs of eigenvalues requires a huge computational overhead, particularly on high-dimensional data. They derived a loose, but computationally efficient upper bound:

$$\|\sin \Theta(\mathcal{V}_p, \mathcal{V}_q)\|_F \leq \frac{2 \min \left\{ d^{\frac{1}{2}} \|\Sigma_p - \Sigma_q\|_{op}, \|\Sigma_p - \Sigma_q\|_F \right\}}{\min \left\{ \lambda_{r-1}^q - \lambda_r^q, \lambda_s^q - \lambda_{s+1}^q \right\}}, \quad (2)$$

where  $\|\cdot\|_{op}$  and  $\|\cdot\|_F$  denote the operator and Frobenius norms, respectively. For large  $n$ , the operator norm can be approximated by the  $\ell_\infty$  norm of the difference between the eigenvalues of  $\Sigma_p$  and  $\Sigma_q$  [6]. The form of the  $\sin \Theta$  distance in Equation (2) replaces the infimum amongst all pairs with the minimum between only two pairs of eigenvalues, which requires less computation.

We now discuss a variant of the  $\sin \Theta$  distance between the subspaces spanned by two datasets. Since the intrinsic dimensionality of the data is not known priori, we compute the  $\sin \Theta$  distance for various choices of  $r$  and  $s$ , and pick the *best* amongst them, which we call the *min sin  $\Theta$  distance*.

**The min sin  $\Theta$  Distance:** Consider the space spanned by the (vectorized) images in the datasets. Since the pixel resolution of the images across datasets is not the same, it is appropriate to first rescale them to the same dimension, for instance, using bilinear interpolation. Depending on whether the rescaled image dimension is greater or smaller than the image dimension, there is a trade-off between the image quality (superior at higher resolution) and computational efficiency (superior at lower resolution). We found out experimentally that resizing all images to  $32 \times 32 \times 3$  is a viable compromise. We consider  $r = 1$  and compute the  $\sin \Theta(\mathcal{V}_p, \mathcal{V}_q; s)$ , for  $s = 3, 4, \dots, \lceil n/10 \rceil$ , where  $n = 3072 = 32 \times 32 \times 3$ . The friendly neighborhood as indicated by the min sin  $\Theta$  distance is  $\min_s \{\sin \Theta(\mathcal{V}_p, \mathcal{V}_q; s)\}$ . In other words, the closest source dataset given all  $s$  is deemed the friendliest neighbor of the target.

## A.2. Comparison of Approaches for Identifying the Friendly Neighborhood

We compare the  $\min \sin \Theta$ , FID, KID and  $\text{CSID}_m$  distances in terms of the friendliest neighbor predicted by these methods. FID, KID and  $\text{CSID}_m$  distances have been defined in Section 3 of the *Main Manuscript*. Table 1 shows the  $\min \sin \Theta$  distance for the various datasets considered in Section 3 of the Main Manuscript. We also present KID between the various datasets in Table 3 of this document. Tables 2 and 4 present the remaining combinations between datasets left out from the *Main Manuscript*. The **first**, **second** and **third** *friendly neighbors* are color-coded for quick and easy identification. We observe across all datasets that, FID and KID are highly correlated in terms of the friendly neighbors they identify for a given target.  $\text{CSID}_m$  is also in agreement with the observations when the target is more diverse, but in scenarios such as TinyImageNet or CIFAR-10, it is able to indicate the less diverse sources as a poor input choice. The experiments on learning Tiny-ImageNet within the Spider GAN framework in Appendix D.2 are more in agreement with the friendly neighbors identified by  $\text{CSID}_m$ .

Across all distances, we observe that the results obtained on MNIST or Fashion-MNIST as the source do not correlate well with the experimental results (cf. Appendix D.2). This is attributed to the limitation of the Inception-Net embedding in handling grayscale images. Inception-Net operates on color images and offers limited performance on grayscale images.

Table 1 shows that the  $\min \sin \Theta$  distance is unable to identify the friendliest neighbor accurately and consistently. For instance, the ordering of the top three neighbors on MNIST, CelebA or LSUN-Churches identified by using the  $\min \sin \Theta$  distance is not consistent with the ordering suggested by  $\text{CSID}_m$  and that verified experimentally. However, on the other datasets,  $\min \sin \Theta$  is worse than the InceptionNet approaches for identifying the friendliest neighborhood.

Table 1. The best-case  $\min \sin \Theta(\cdot)$  distance between the spaces spanned by the eigenvectors of the source and target datasets. The rows represent the sources and the columns correspond to the target datasets. The **first**, **second** and **third** *friendly neighbors* (color coded) of the target is the source with the three lowest  $\min \sin \Theta(\cdot)$  values is that column. We observe that the *friendliest neighbor* identified by the  $\min \sin \Theta$  distance are generally not in agreement with those identified by FID, KID or  $\text{CSID}_m$ .

Src \ Tar	MNIST	F-MNIST	SVHN	CIFAR-10	T-ImgNet	CelebA	Ukiyo-E	Church
MNIST	0	60.74	63.25	85.73	43.19	27.43	23.79	35.35
F-MNIST	96.68	0	69.01	110.7	53.77	36.69	45.02	48.29
SVHN	79.91	<b>54.77</b>	0	57.99	23.62	19.86	25.95	29.55
CIFAR-10	72.16	58.56	<b>35.97</b>	0	<b>7.521</b>	<b>14.63</b>	21.16	<b>15.89</b>
T-ImgNet	<b>70.86</b>	<b>55.43</b>	<b>30.67</b>	<b>14.67</b>	0	<b>13.97</b>	<b>20.05</b>	<b>15.52</b>
CelebA	72.13	60.62	<b>41.35</b>	<b>45.74</b>	<b>22.39</b>	0	<b>19.16</b>	23.48
Ukiyo-E	<b>54.09</b>	59.30	43.08	52.75	25.65	<b>15.29</b>	0	<b>22.50</b>
Church	<b>66.54</b>	<b>57.11</b>	44.02	<b>35.55</b>	<b>17.81</b>	16.80	<b>20.19</b>	0

Table 2. A comparison of FID between popular training datasets. The rows correspond to the source (Src) and the columns correspond to the target (Tar). The **first**, **second** and **third** *friendly neighbors* (color coded) of the target are the sources with the three lowest FID values. FID fails to detect scenarios where the source possesses lower sample diverse that the target, as in the case of CIFAR-10 and LSUN-Church sources in comparison to the Tiny-ImageNet target.

Src \ Tar	MNIST	F-MNIST	SVHN	CIFAR-10	T-ImgNet	CelebA	Ukiyo-E	Church
MNIST	1.2491	<b>175.739</b>	234.850	258.246	264.250	360.622	398.280	357.428
F-MNIST	<b>176.813</b>	2.4936	212.619	188.367	197.057	365.222	387.049	345.011
SVHN	<b>236.707</b>	214.262	3.4766	<b>168.615</b>	<b>189.133</b>	357.193	372.444	356.148
CIFAR-10	<b>259.045</b>	<b>188.710</b>	<b>168.113</b>	5.0724	<b>64.3941</b>	305.528	303.694	<b>256.207</b>
T-ImgNet	264.309	<b>197.918</b>	<b>188.823</b>	<b>64.0312</b>	6.4845	<b>251.198</b>	<b>257.078</b>	<b>203.899</b>
CelebA	360.773	364.586	357.383	303.490	250.735	2.5846	<b>301.108</b>	<b>265.954</b>
Ukiyo-E	396.791	387.088	372.557	300.511	254.102	<b>300.259</b>	5.9137	267.624
Church	350.708	343.781	<b>354.885</b>	<b>254.991</b>	<b>204.162</b>	<b>266.508</b>	<b>267.638</b>	2.5085

Table 3. KID between popular training datasets. The **first**, **second** and **third** *friendly neighbors* (color coded) of the target (column) are the sources (rows) with the three lowest KID values. We observe that, akin to FID, the KID measure is also unable to compare the leave diversity between the source and target datasets, as is the case between Tiny-ImageNet and CIFAR-10.

Src \ Tar	MNIST	F-MNIST	SVHN	CIFAR-10	T-ImgNet	CelebA	Ukiyo-E	Church
MNIST	$2 \times 10^{-6}$	<b>0.1587</b>	0.2428	0.2380	0.2393	0.4284	0.5082	0.4376
F-MNIST	<b>0.1606</b>	$1 \times 10^{-6}$	0.1922	0.1353	0.1578	0.4291	0.4751	0.3963
SVHN	0.2458	0.1943	$2 \times 10^{-7}$	<b>0.1377</b>	<b>0.1674</b>	0.4059	0.4393	0.3962
CIFAR-10	<b>0.2404</b>	<b>0.1357</b>	<b>0.1377</b>	$6 \times 10^{-6}$	<b>0.0334</b>	<b>0.3205</b>	<b>0.3229</b>	<b>0.2453</b>
T-ImgNet	<b>0.2397</b>	<b>0.1579</b>	<b>0.1667</b>	<b>0.0321</b>	$8 \times 10^{-6}$	<b>0.2403</b>	<b>0.2595</b>	<b>0.1692</b>
CelebA	0.4388	0.4265	0.4054	0.3165	0.2406	$7 \times 10^{-6}$	0.3620	<b>0.2856</b>
Ukiyo-E	0.5064	0.4746	0.4408	0.3183	0.2568	0.3610	$2 \times 10^{-5}$	0.3022
Church	0.4379	0.3916	<b>0.3932</b>	<b>0.2408</b>	<b>0.1695</b>	<b>0.2857</b>	<b>0.3019</b>	$3 \times 10^{-5}$

Table 4. A comparison of  $CSID_m$  between popular training datasets for  $m = \lfloor \frac{n}{2} \rfloor$ . The rows represent the source (Src) and the columns represent to the target (Tar). The **first**, **second** and **third** *friendly neighbors* (color coded) of the target are the sources with the three lowest positive  $CSID_m$  values, respectively.  $CSID_m$  is superior to FID or KID, as it assigns negative values to source datasets that are less diverse than the target.

Src \ Tar	MNIST	F-MNIST	SVHN	CIFAR-10	T-ImgNet	CelebA	Ukiyo-E	Church
MNIST	0.1865	<b>21.886</b>	37.227	29.298	9.436	198.714	201.550	205.322
F-MNIST	<b>162.962</b>	0.1097	46.938	19.051	-0.5571	167.840	191.010	181.458
SVHN	<b>212.473</b>	77.357	-0.0566	<b>34.534</b>	<b>21.668</b>	195.631	214.507	219.790
CIFAR-10	221.337	<b>65.426</b>	<b>52.051</b>	-0.1478	-7.109	<b>180.491</b>	198.991	<b>173.655</b>
T-ImgNet	230.916	75.737	<b>67.902</b>	<b>12.892</b>	0.6743	<b>157.520</b>	<b>197.447</b>	<b>184.977</b>
CelebA	<b>204.794</b>	68.828	65.299	<b>23.685</b>	<b>8.829</b>	0.6241	<b>184.170</b>	191.927
Ukiyo-E	250.226	92.741	82.157	39.792	<b>18.727</b>	191.930	0.5494	<b>180.697</b>
Church	212.452	<b>48.676</b>	<b>56.136</b>	-4.655	-23.115	<b>185.740</b>	<b>198.750</b>	-0.5258

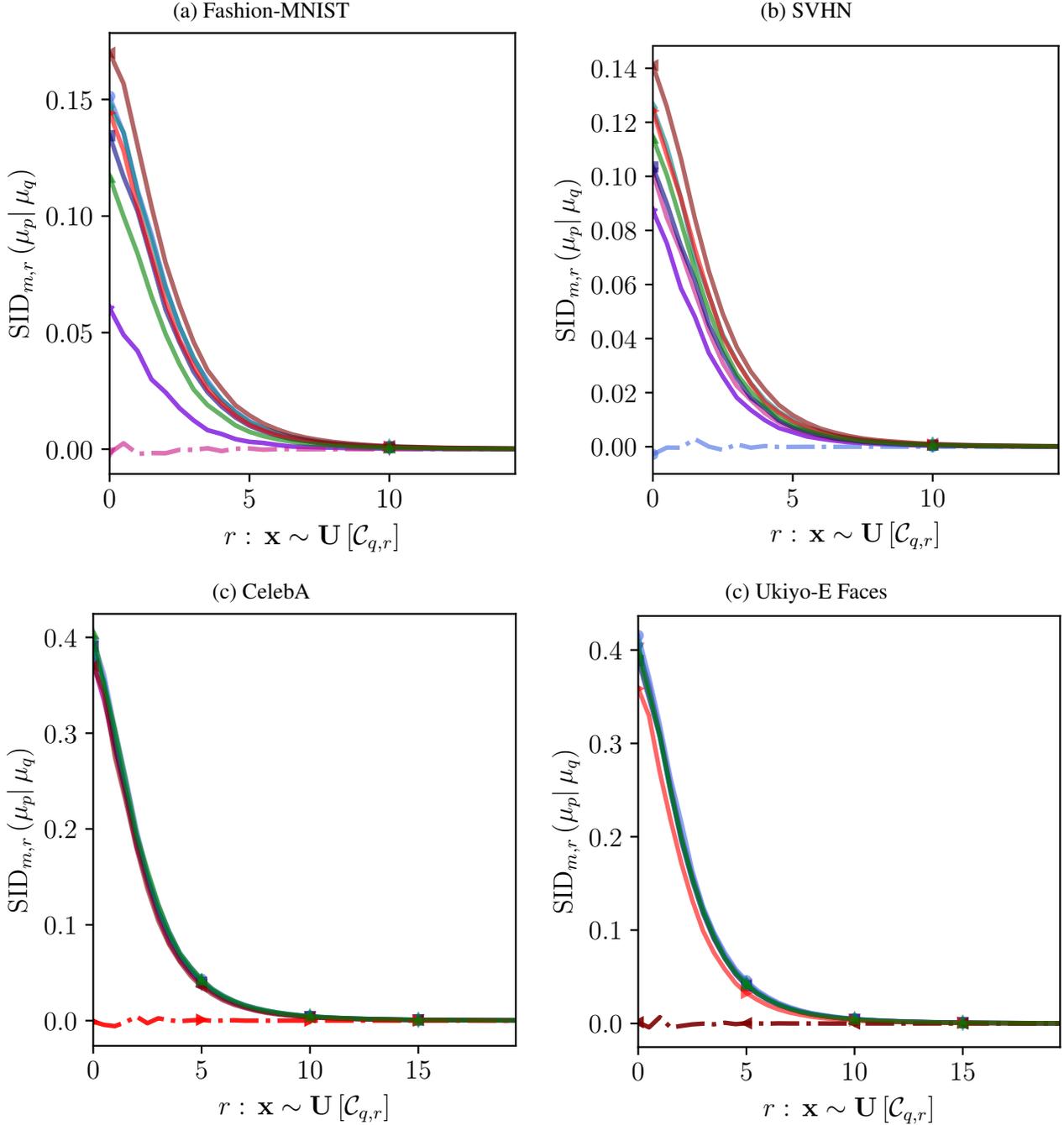


Figure 1.  $SID_{m,r}$  as a function of the hyper-cube length  $r$ . We observe that MNIST is the closest neighbor to both Fashion-MNIST and SVHN, while CelebA is marginally closer to Ukiyo-E than the other baselines considered. In scenarios such as case when the target is CelebA or Ukiyo-E Faces, the SID curve alone cannot be used to conclude the *friendliest neighbor* of a target dataset, and the area under the curve,  $CSID_m$  is more informative (cf. Table 4).

## B. The Signed Inception Distance (SID)

In this appendix, we derive a favorable theoretical guarantee of the SID metric, discuss the algorithm for the computation of SID with relevant ablation experiments on synthetic Gaussian and image datasets.

### B.1. Asymptotic Behavior of the Signed Distance

Without loss of generality, consider the *signed distance* presented in Equation (2) of the *Main Manuscript*:

$$\text{SD}_{m,r}(\mu_p||\mu_q) = \frac{1}{M_{\mathbf{x}}} \sum_{\substack{\ell=1 \\ \tilde{\mathbf{x}}_{\ell} \in \mathcal{C}_{q,r}}}^{M_{\mathbf{x}}} \left\{ \frac{1}{N_q} \sum_{\substack{j=1 \\ \mathbf{c}_j \sim \mu_q}}^{N_q} \Phi(\mathbf{x}_{\ell}, \mathbf{c}_j) - \frac{1}{N_p} \sum_{\substack{i=1 \\ \tilde{\mathbf{c}}_i \sim \mu_p}}^{N_p} \Phi(\mathbf{x}_{\ell}, \tilde{\mathbf{c}}_i) \right\}.$$

Asymptotically, when infinite samples are drawn from the test space,  $\mathcal{C}_{q,r}$ , we get

$$\begin{aligned} \text{SD}_{m,r}(\mu_p||\mu_q) &= \lim_{M_{\mathbf{x}} \rightarrow \infty} \left\{ \frac{1}{M_{\mathbf{x}}} \sum_{\substack{\ell=1 \\ \tilde{\mathbf{x}}_{\ell} \in \mathcal{C}_{q,r}}}^{M_{\mathbf{x}}} \left\{ \frac{1}{N_q} \sum_{\substack{j=1 \\ \mathbf{c}_j \sim \mu_q}}^{N_q} \Phi(\mathbf{x}_{\ell}, \mathbf{c}_j) - \frac{1}{N_p} \sum_{\substack{i=1 \\ \tilde{\mathbf{c}}_i \sim \mu_p}}^{N_p} \Phi(\mathbf{x}_{\ell}, \tilde{\mathbf{c}}_i) \right\} \right\}. \\ &= \kappa \int_{\mathbf{x} \in \mathcal{C}_{q,r}} \left\{ \frac{1}{N_q} \sum_{\substack{j=1 \\ \mathbf{c}_j \sim \mu_q}}^{N_q} \Phi(\mathbf{x}, \mathbf{c}_j) - \frac{1}{N_p} \sum_{\substack{i=1 \\ \tilde{\mathbf{c}}_i \sim \mu_p}}^{N_p} \Phi(\mathbf{x}, \tilde{\mathbf{c}}_i) \right\} d\mathbf{x}, \end{aligned}$$

for some positive constant  $\kappa$ . Similarly, when the number of centers drawn from  $\mu_p$  and  $\mu_q$  tends to infinity, the inner summations can be replaced with their corresponding expectations, resulting in

$$\begin{aligned} \text{SD}_{m,r}(\mu_p||\mu_q) &= \kappa \int_{\mathbf{x} \in \mathcal{C}_{q,r}} \lim_{N_q \rightarrow \infty} \left\{ \frac{1}{N_q} \sum_{\substack{j=1 \\ \mathbf{c}_j \sim \mu_q}}^{N_q} \Phi(\mathbf{x}, \mathbf{c}_j) \right\} - \lim_{N_p \rightarrow \infty} \left\{ \frac{1}{N_p} \sum_{\substack{i=1 \\ \tilde{\mathbf{c}}_i \sim \mu_p}}^{N_p} \Phi(\mathbf{x}, \tilde{\mathbf{c}}_i) \right\} d\mathbf{x}. \\ &= \kappa' \int_{\mathbf{x} \in \mathcal{C}_{q,r}} \left( \int_{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y}) \mu_q(\mathbf{y}) d\mathbf{y} - \int_{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y}) \mu_p(\mathbf{y}) d\mathbf{y} \right) d\mathbf{x}. \\ &= \kappa' \int_{\mathbf{x} \in \mathcal{C}_{q,r}} \left( \mathbb{E}_{\mathbf{y} \sim \mu_q} [\Phi(\mathbf{x}, \mathbf{y})] - \mathbb{E}_{\mathbf{y} \sim \mu_p} [\Phi(\mathbf{x}, \mathbf{y})] \right) d\mathbf{x}. \end{aligned}$$

Recall that the samples  $\mathbf{x}_{\ell}$  are drawn uniformly at random from  $\mathcal{C}_{q,r}$  (cf. Section 3.1 of the *Main Manuscript*). This allows us to replace the outer integral with another expectation, resulting in

$$\text{SD}_{m,r}(\mu_p||\mu_q) = \mathbb{E}_{\mathbf{x} \sim \mathcal{C}_{q,r}, \mathbf{y} \sim \mu_q} [\Phi(\mathbf{x}, \mathbf{y})] - \mathbb{E}_{\mathbf{x} \sim \mathcal{C}_{q,r}, \mathbf{y} \sim \mu_p} [\Phi(\mathbf{x}, \mathbf{y})].$$

The above result links the SD to kernel statistics and provides the asymptotic guarantee that when the two distributions  $\mu_p$  and  $\mu_q$  coincide, *i.e.*,  $\mu_p = \mu_q$ , and therefore,  $\text{SD}_{m,r}(\mu_p||\mu_p) := 0$ .

### B.2. SID Computation

The procedure to compute the signed distance between the samples drawn from two distributions is given in Algorithm 1. While the algorithm is easily implementable for low-dimensional data, an extension to practical settings with images necessitates computing Inception embeddings over batches of samples. The signed distance (SD) computed over Inception embeddings is called SID. To extend the SID computation algorithm for evaluating GANs, we consider  $\mathcal{D}_q$ , the target dataset, and  $\mathcal{D}_p$ , samples drawn from the generator. We set  $|\mathcal{D}_q| = |\mathcal{D}_p| = 5000$ . For each  $r$ , we average  $\text{SID}_{m,r}$  over batches of size  $N_B = 100$ . This allows for efficient computation of the Inception features for high-resolution images. Algorithm 2 presents this modified approach for evaluating GANs with SID. We implement the SID computation atop the publicly available Clean-FID [10] library. Similar to the Clean-FID framework, SID can be computed between any two image folders using the Clean-FID backend. As a result, the InceptionV3 mapping and resizing functions are consistent with the existing Clean-FID approach. Details regarding the public release of the Python + TensorFlow/PyTorch library for SID computation are discussed in Appendix E of this document.

---

**Algorithm 1: Signed distance (SD) between two distributions**

---

**Input:** Source data  $\mathcal{D}_p = \{\tilde{c}_i \mid i = 1, 2, 3, \dots, N_p; \tilde{c}_i \sim \mu_p\}$ , kernel order  $m$ , dimensionality  $n$ ,  
Target data  $\mathcal{D}_q = \{c_j \mid j = 1, 2, 3, \dots, N_q; c_j \sim \mu_q\}$ , max radius  $r_{\max}$ , step size  $\eta$ ,  
batch size  $M_x$

**Compute:**  $\mu_q = \text{mean}(c_j \sim \mathcal{D}_q)$ ;  $\Sigma_q = \text{covariance}(c_j \sim \mathcal{D}_q)$

**for**  $k = 0, 1, 2, \dots, r_{\max}$  **do**

**Compute:** Hypercube length  $r = \eta \times k \times \max(\text{diag}(\Sigma_q))$

**Define:** Hypercube  $\mathcal{C}_{q,r}$ : Center =  $\mu_q$

**Sample:**  $x_\ell \sim \text{Uniform}[\mathcal{C}_{q,r}]$ ;  $\ell = 1, 2, 3, \dots, M_x$

**Compute:**  $\text{SD}_{m,r}(\mu_p \parallel \mu_q)$  based on Equation (2) of the *Main Manuscript*

**Output:** Plot of  $\text{SD}_{m,r}$  versus  $r$

---

---

**Algorithm 2: Signed inception distance (SID) between the generator output and target data**

---

**Input:** Target data  $\mathcal{D}_q = \{c_j \mid j = 1, 2, \dots, N_q; c_j \sim \mu_q\}$ , kernel order  $m$ , dimensionality  $n$ ,  
max radius  $r_{\max}$ , step size  $\eta$ , hypercube sample batch size  $M_x$ , Generator  $G$ ,  
Generator batch size  $N_B$ , Inception model  $\psi$ .

**Compute:**  $\mu_q = \text{mean}(c_j \sim \mathcal{D}_q)$ ;  $\Sigma_q = \text{covariance}(c_j \sim \mathcal{D}_q)$

**for**  $k = 0, 1, 2, \dots, r_{\max}$  **do**

**for** Batches  $i = 1, 2, \dots, \frac{N_q}{N_B}$  **do**

**Sample:**  $z_\ell \sim p_Z(z)$ ;  $\ell = 1, 2, 3, \dots, N_B$  – Generator inputs

**Sample:**  $\tilde{c}_\ell \sim G(z)$ ;  $\ell = 1, 2, 3, \dots, N_B$  – Generator outputs

**Sample:**  $c_j \sim \mathcal{D}_q$ ;  $\ell = 1, 2, 3, \dots, N_B$  – Target data samples

**Compute:**  $\psi(\tilde{c}_\ell)$  and  $\psi(c_j)$  – Inception embeddings of generator output and target data.

**Compute:** Hypercube length  $r = \eta \times k \times \max(\text{diag}(\Sigma_q))$

**Define:** Hypercube  $\mathcal{C}_{q,r}$ : Center =  $\mu_q$

**Sample:**  $x_\ell \sim \text{Uniform}[\mathcal{C}_{q,r}]$ ;  $\ell = 1, 2, 3, \dots, M_x$

**Compute:**  $\text{SID}_{m,r}$  between  $\psi(\tilde{c}_\ell)$  and  $\psi(c_j)$  based on Equation (3) of the *Main Manuscript*

**Compute:**  $\text{CSID}_m = \sum_r \text{SID}_{m,r}(\mu_p \parallel \mu_q)$

**Output:** Plot of  $\text{SID}_{m,r}$  versus  $r$ ; Measure  $\text{CSID}_m$

---

### B.3. Experiments on Gaussian Data

To begin with, we present results on computing the signed distance (SD) for various representative Gaussian and Gaussian mixture source and target distributions.

Figures 2(a)-(c) present the visualization of SD versus  $r$  for a Gaussian target distribution with  $\mu_q = \mathcal{N}(5.5\mathbf{1}_2, 0.75\mathbb{I}_2)$ , where  $\mathbf{1}_2$  denotes a 2-D vector with all entries equal to 1. Consider the scenario where the source and target Gaussians possess the same variance, but different means. Consider three different sources  $\mu_p = \mathcal{N}(\mathbf{m}_p, \Sigma_p)$ , given by: (a)  $\mathbf{m}_p = \mathbf{0}_2$  and  $\Sigma_p = 0.75\mathbb{I}_2$ ; (b)  $\mathbf{m}_p = 2.5\mathbf{1}_2$  and  $\Sigma_p = 0.75\mathbb{I}_2$ ; and (c)  $\mathbf{m}_p = 5.5\mathbf{1}_2$  and  $\Sigma_p = 0.75\mathbb{I}_2$ . We observe that, when the source is far away from the target, SD is positive-valued and gradually approaches zero. When the two distributions are identical, SD is zero for all  $r$ . In the context of identifying a friendly neighbor, a closer source dataset is expected to converge faster to zero than one that is far away.

Figures 3(a)-(c) present the results for the other scenario where the mean is fixed, but the variances are different. Consider the same target as before, but with the following source distributions: (a)  $\mathbf{m}_p = 5\mathbf{1}_2$  and  $\Sigma_p = 0.1\mathbb{I}_2$ ; (b)  $\mathbf{m}_p = 5\mathbf{1}_2$  and  $\Sigma_p = 0.25\mathbb{I}_2$ ; and (c)  $\mathbf{m}_p = \mathbf{1}_2$  and  $\Sigma_p = \mathbb{I}_2$ . We observe that when the spread of the source is smaller than the target, SD initially goes negative, and subsequently converges to zero once the hypercube  $\mathcal{C}_{q,r}$  encompasses the source. On the other hand, when the spread of the source is greater than that of the target (as desired for identifying friendly neighbors), SD is always positive, and converges to zero faster if the relative spread between the source and target is smaller.

To evaluate SD on a Gaussian mixture target, consider an 8-component Gaussian mixture model (GMM) with means drawn from  $[0, 1] \times [0, 1]$  and identical covariance matrices  $0.02\mathbb{I}_2$ . Consider three source distributions: (a) A Gaussian with first and second moments matching that of the target; (b) An 8-component GMM distinct from the target; and (c) A 4-component

GMM that has mode-collapsed on to some of the modes of the target. Figures 4(a)-(c) present these three scenarios and the associated SD versus  $r$ . For Scenario (a), although the mean and covariance of both the source and target are identical, we observe that SD is negative, as the two distributions do not have a large overlap, preventing the positive and negative charges from cancelling each other. In Scenario (b), SD is able to capture the change in concentration between  $\mu_p$  and  $\mu_q$ , indicated by the sudden sign change in SD. When  $\mu_p$  converges to a few modes of the target  $\mu_q$ , SD is not zero for all  $r$ , which indicates that the two distributions are not identical. In this scenario, however, FID between the two distribution would be close to zero as they have approximately the same first and second moments.

Animations pertaining to these experiments are available at <https://github.com/DarthSid95/clean-sid>.

#### B.4. Evaluating GANs with SID

We consider evaluating pre-trained models with the SID measure to compare the performance with FID and KID. As a demonstration, we consider StyleGAN2 [11] and StyleGAN3 [12] models with weights trained on  $512 \times 512$  high-quality Animal Faces (AFHQ) dataset [11]. As a reference/benchmark, we also consider SID of the AFHQ dataset with itself. We consider orders in the range  $m = \lfloor \frac{n}{2} \rfloor - 3, \lfloor \frac{n}{2} \rfloor - 2, \dots, \lfloor \frac{n}{2} \rfloor + 2$ . Figure 5 shows SID for select orders, comparing StyleGAN2 and StyleGAN3. For positive orders, we flip the sign of SID to maintain consistency with the interpretations developed for the negative order. Across all test cases, we observe that StyleGAN3 outperforms StyleGAN2, as suggested by the FID and KID values [10]. As the order  $m$  reduces, GAN models with lower FID/KID/CSID $_m$  approach zero more rapidly, which can be used to quantify the relative performance of converged GAN models. For  $m < \lfloor \frac{n}{2} \rfloor - 3$  numerical instability causes SID to approach zero and for  $m > \lfloor \frac{n}{2} \rfloor + 2$  numerical instability blows up SID computation. While these experiments serve to demonstrate the feasibility in evaluating pre-trained GAN models with CSID $_m$ , comparisons between Spider DCGAN and the corresponding baselines are provided in Section 4 of the *Main Manuscript* and Appendix D.2 of this *Supporting Document*, while comparisons of Spider StyleGANs and baseline StyleGANs on FFHQ and MetFaces is provided in Appendix D.5.

SID can also be used to compare the relative performance of GAN generators. Consider three GANs trained on the MNIST dataset where one generator has learnt the distribution accurately, while the other two have *mode-collapsed* on to a subset of the classes (specifically, digits 0,8,6 and 9) or a single class (digit 4) of the target dataset. Figure 6 presents samples output by these generators and the SID versus  $r$  plot for the corresponding pair of generators. We observe that, when the reference generator has learnt the target accurately, the SID of a test generator’s output with respect to the reference will always be negative, as the test generator has less diversity. However, the SID between the output of two generators that have mode-collapsed would be positive if there is no overlap between the classes they have collapsed to. This could be used to evaluate GANs with ensemble-generators [13], where each network is trained to learn a different mode/class.

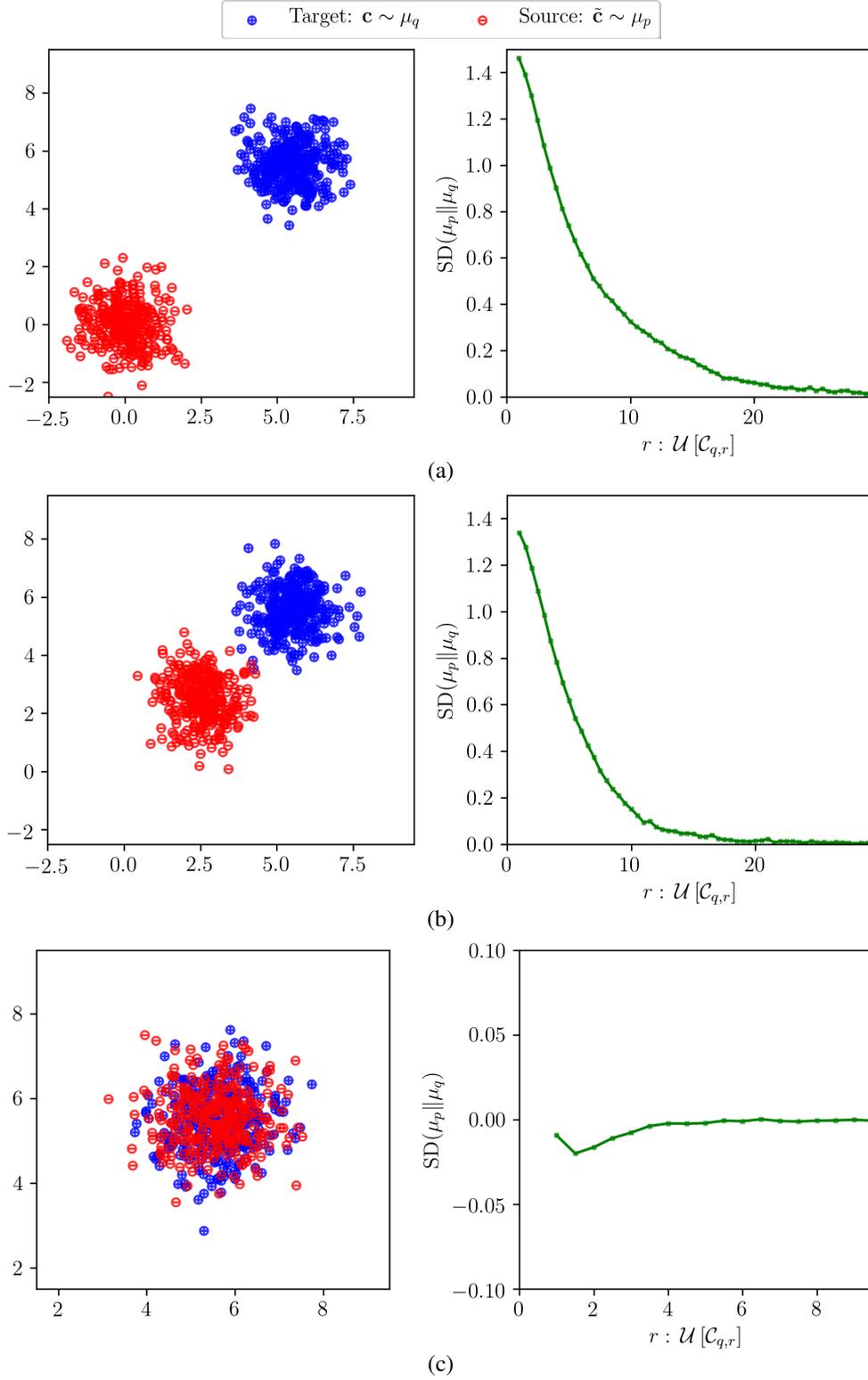


Figure 2. Plots of the signed distance  $SD_{m,r}$  between a source Gaussian  $\mu_p = \mathcal{N}(\mathbf{m}, 0.75\mathbb{I}_2)$  from a target Gaussian  $\mu_q = \mathcal{N}(5.5\mathbf{1}_2, 0.75\mathbb{I}_2)$  for (a)  $\mathbf{m}_p = \mathbf{0}_2$ ; (b)  $\mathbf{m}_p = 2.5\mathbf{1}_2$ ; and (c)  $\mathbf{m}_p = 5.5\mathbf{1}_2$ . The closer the source Gaussian is to the target, the faster  $SD_{m,r}(\mu_p || \mu_q)$  approaches zero. When the two distributions coincide,  $SD_{m,r}(\mu_p || \mu_q)$  is zero for all  $r$ .

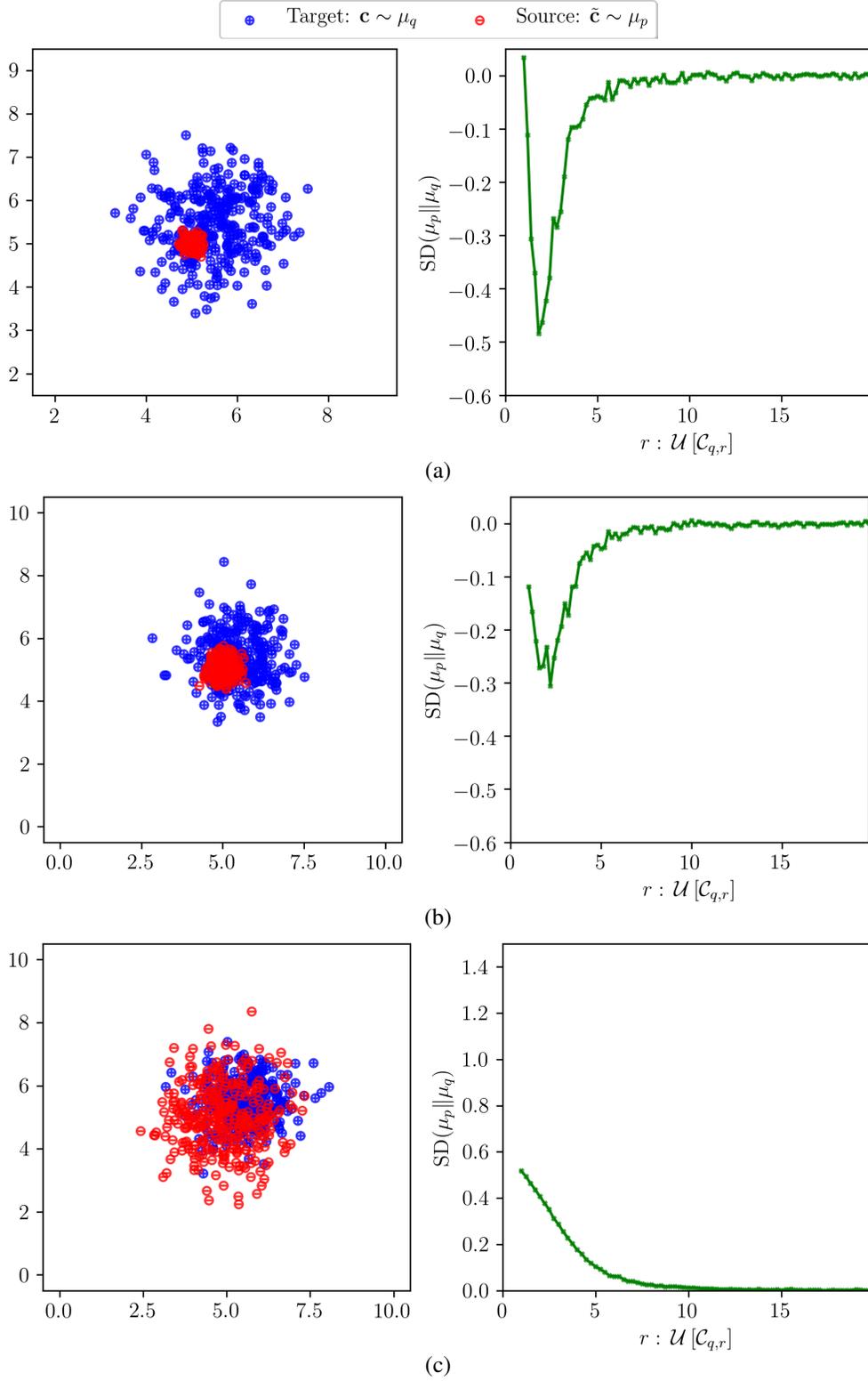


Figure 3. Plots comparing the signed distance  $SD_{m,r}$  between a source Gaussian  $\mu_p = \mathcal{N}(5.5\mathbf{1}_2, 0.75\mathbb{I}_2)$  and a target Gaussian  $\mu_q = \mathcal{N}(5.5\mathbf{1}_2, \Sigma_p)$  for (a)  $\Sigma_p = 0.1\mathbb{I}_2$ ; (b)  $\Sigma_p = 0.25\mathbb{I}_2$ ; and (c)  $\Sigma_p = \mathbb{I}_2$ . When the source Gaussian overlaps with the target but with a smaller variance,  $SD_{m,r}$  is negative. However, if the source has a larger variance than the target,  $SD_{m,r}$  is positive.

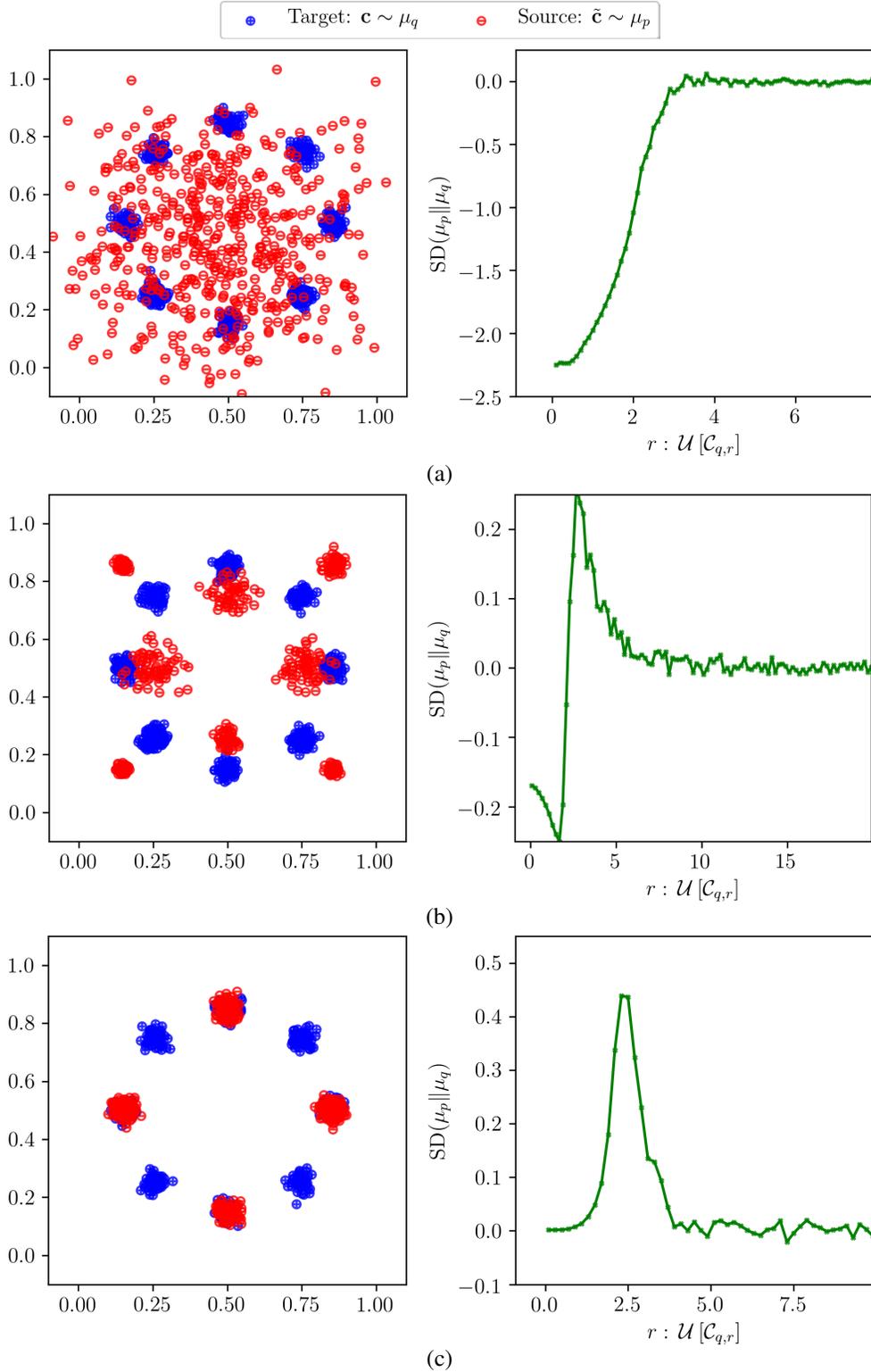


Figure 4. Plots comparing the signed distance  $SD_{m,r}$  when the target is a Gaussian mixture density. (a) Unimodal Gaussian source with identical first and second moments as the target;  $SD_{m,r}$  is negative as the source has lower diversity than the target. (b) A Gaussian mixture distinct from the target;  $SD_{m,r}$  flips sign based on the relative concentrations of the source and target samples. (c) A mode-collapsed source results in a non-zero  $SD_{m,r}$ , although FID and KID between these distributions would be zero.

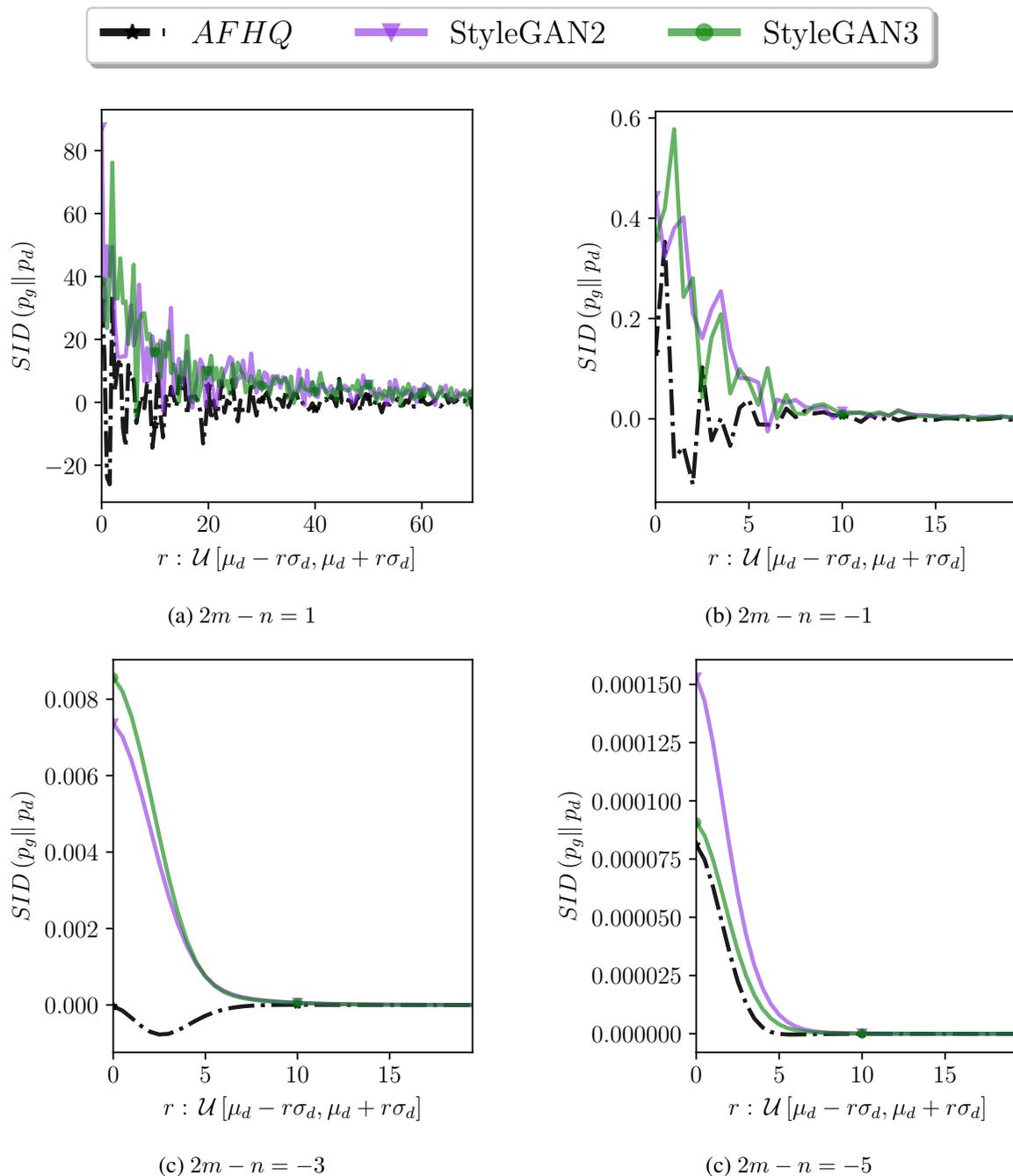


Figure 5. SID versus  $r$  for multiple choices of  $2m - n$  for the case when the target dataset is  $512 \times 512$  Animal Faces HQ. Images generated by StyleGAN2 and StyleGAN3 are compared with the AFHQ dataset as the target. We observe that StyleGAN3 has a performance comparable to StyleGAN2 for higher orders  $m$ . Convergence for lower orders is indicative of superior performance, as the penalty for mismatch between the source and target distributions increases with decrease in the order. The SID for StyleGAN3 closely matches the SID of the target data with itself for  $2m - n = -5$ , indicating superior performance to StyleGAN2. This finding is in agreement with the comparison between StyleGAN2 and StyleGAN3 in terms of FID/KID reported in [10].

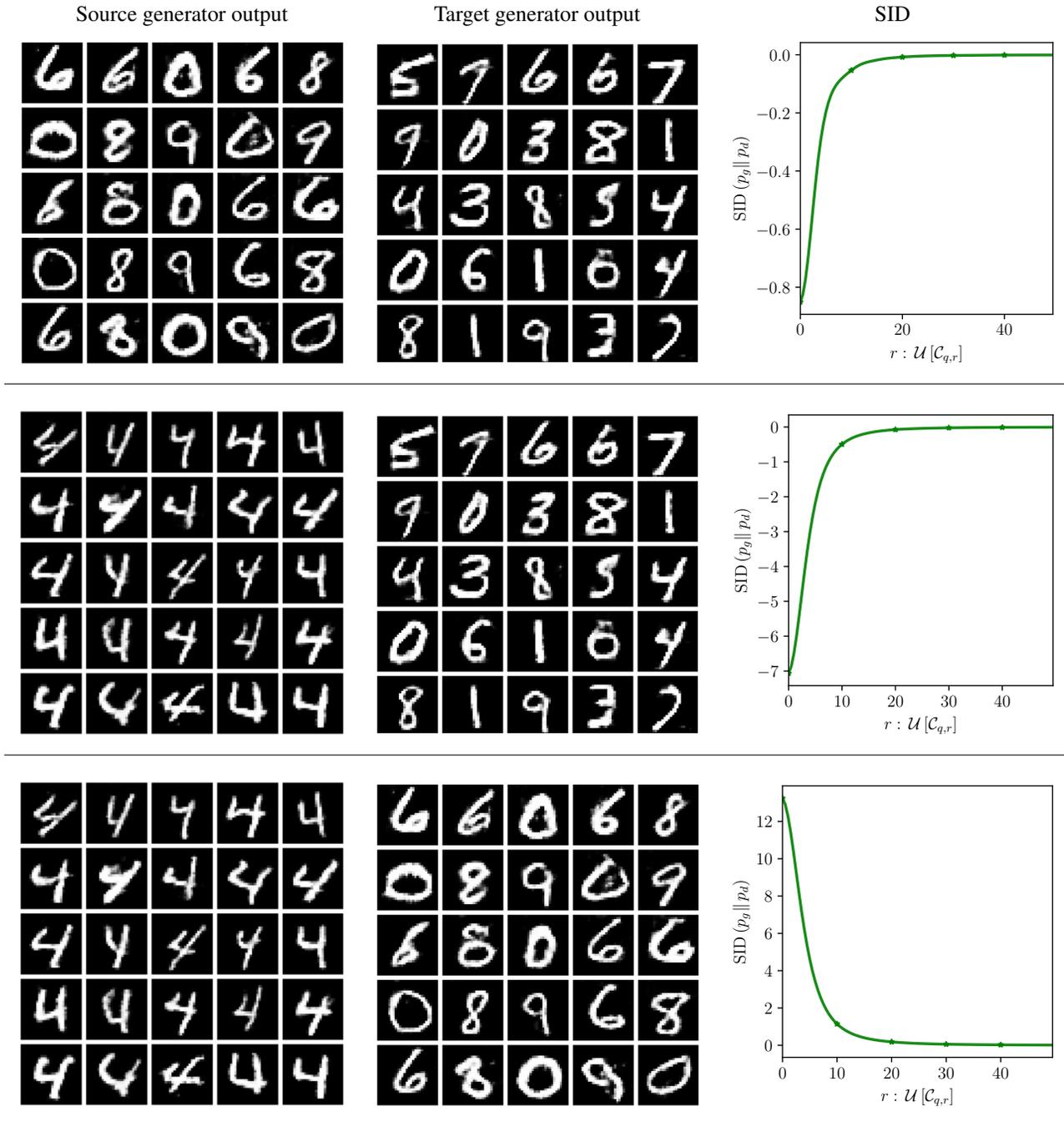


Figure 6. SID versus  $r$  when the source and target samples are drawn from GAN generators trained on various subsets of MNIST. When the source generator has mode-collapsed, either to a single digit or a subset of digits, the corresponding SID is negative. When comparing two mode-collapsed generators, the SID will be positive as the distributions of the Inception embeddings are less likely to overlap.

## C. Implementation Details

We provide details on the experimental setup, evaluation metrics and computational resources employed in the various experiments reported in the *Main Manuscript* and this *Supporting Document*.

### C.1. Experimental Setup

**Spider DCGAN:** The experiments presented in Section 4 consider the DCGAN [14] architecture for the generator and discriminator. For the baseline GANs, the parametric input is drawn from  $\mathbb{R}^{100}$ . We consider the Gaussian, Gamma [15] and non-parametric [16] input distributions drawn from  $\mathbb{R}^{100}$  as baselines. In the case of Spider GAN, we conducted experiments by resizing the input data to  $16 \times 16 \times 3$ . To bridge the gap between the two noise variants, we also consider Gaussian noise drawn from  $\mathbb{R}^{16 \times 16 \times 3}$  provided as input in a similar fashion to the datasets. We did not observe improvement in performance with higher-resolution images for the input dataset. The images are vectorized and provided as input to the generator. Both Spider DCGAN and the baselines are trained on the Wasserstein GAN [17] loss with a stable version of the gradient penalty [18] enforced only on samples drawn from  $p_d$ . The choice was motivated by its successful usage in baseline StyleGAN2 and StyleGAN3 variants.

The networks are trained on batches of 100 samples. The Adam [19] optimizer is used with a learning rate  $\eta = 2 \times 10^{-4}$ , and the exponential decay parameters for the first and second moments are  $\beta_1 = 0.5$  and  $\beta_2 = 0.999$ , respectively. The implementation was carried out using TensorFlow 2.0 [20]. The networks are trained for  $15 \times 10^3$  iterations on MNIST and Fashion-MNIST,  $10^4$  iterations on SVHN and CIFAR-10, and  $3 \times 10^4$  iterations on Celeb-A, Ukiyo-E and Tiny-ImageNet learning tasks.

**Spider PGGAN:** The publicly available PGGAN GitHub repository (URL: [https://github.com/tkarras/progressive\\_growing\\_of\\_gans](https://github.com/tkarras/progressive_growing_of_gans)) was extended to incorporate the *Spider* framework. The implementation was carried out using TensorFlow 2.0 [20]. The input distributions are drawn from PGGAN models, trained on Tiny-ImageNet images of resolution  $16 \times 16 \times 3$ . The input PGGAN was trained for  $12 \times 10^3$  iterations. Samples drawn from the input PGGAN are resized to  $14 \times 14 \times 3$ , vectorized, and provided as input to the cascaded Spider PGGAN layer.

**Spider StyleGAN:** The publicly available, PyTorch 1.10 [21] based StyleGAN3 GitHub repository (URL: <https://github.com/NVlabs/stylegan3>) was extended to incorporate the *Spider* framework, allowing for the implementation of both StyleGAN2, StyleGAN2-ADA and StyleGAN3 variants. The input distributions are drawn from StyleGAN2-ADA models, trained on (a) Tiny-ImageNet images of resolution  $16 \times 16 \times 3$ ; and (b) Images from the AFHQ-Dogs dataset, resized to  $16 \times 16 \times 3$ . The input StyleGAN was trained for  $25 \times 10^3$  iterations in both cases. We considered the following two input transformations to obtain 512-dimensional input vectors: (i) Samples drawn from the input StyleGAN are averaged across the color channels, resized to  $16 \times 32 \times 1$ , vectorized, and provided as input to the cascaded layer; and (ii) Samples drawn from the input StyleGAN are averaged across the color channels, resized to  $23 \times 23 \times 1$  and vectorized. The vectors are truncated to 512 entries, and provided as input to the cascaded stage. We did not observe a significant difference in performance when considering either of the two configurations. As in classical StyleGANs, the cascaded StyleGAN network transforms the input dataset to the latent  $\mathcal{W}$ -space, and subsequently learn the target. Spider StyleGANs are trained with transformation-(i) on FFHQ and AFHQ-Cats data, while transformation-(ii) is used to train the Spider StyleGAN variants on Ukiyo-E faces and MetFaces.

### C.2. Evaluation Metrics

To draw a fair comparison with the baseline approaches, we evaluate various Spider GAN and baseline models in terms of their FID, KID and CSID<sub>m</sub>. We also compare the interpolation quality of the networks based on the sharpness of the interpolated images.

**Fréchet Inception Distance (FID):** Proposed by Heusel *et al.* [22], FID can be used to quantify how *real* samples generated by GANs are. FID is computed as the Wasserstein distance between Gaussian distributed embeddings of the generated and target images. To compute the image embedding, we consider the InceptionV3 [23] model without the topmost layer, loaded with weights for the ImageNet [24] classification task. Images are resized to  $299 \times 299 \times 3$  and given as input to these networks. Grayscale images are replicated across the color channels. FID is computed by assuming a Gaussian prior on the embeddings of real and fake images. The means and covariances are estimated using 10,000 samples. The publicly available TensorFlow based *Clean-FID* library [10] is used to compute FID. As noted by Parmar *et al.* [10], the Clean-FID is generally found to be a few points higher than those computed through base PyTorch and TensorFlow implementations. Our implementation of the DCGAN baselines [15, 16] also exhibit similar offsets between the reported FID and those computed by Clean-FID. However, in our experiments, we were able to reproduce the scores reported in [10] for PGGAN and StyleGAN architectures fairly accurately.

**Kernel Inception Distance (KID):** The kernel inception distance [25] is an unbiased alternative to FID. The KID computes the squared maximum-mean discrepancy (MMD) between the InceptionV3 embeddings of data in  $\mathbb{R}^n$ . The embeddings are computed as in the FID case. The third-order polynomial kernel  $\mathcal{K}(x, y) = (\frac{1}{n}x^T y + 1)^3$  is used to compute the MMD over a batch of 5000 samples. As in the case of FID, to maintain consistency, we use the *Clean-FID* [10] library implementation of KID.

**Image Interpolation and Sharpness:** In order to compare the performance of GAN for generating unseen images, we evaluate the output of the generator when the interpolated points between two input distribution samples are provided to the generator. We use the sharpness metric introduced by Tolstikhin *et al.* [26] in the context of Wasserstein autoencoders. The edge-map of an image is obtained using the Laplacian operator. The average sharpness of the images is then defined as the variance in pixel intensities on the edge-map, averaged over batches of 50,000 images. In the case of baseline GAN, the inputs are interpolated points between random samples drawn from the parametric noise distribution, while in the case of Spider GAN, the interpolation between two images from the input dataset are fed to the generator.

### C.3. Computational Resources

All experiments on low-resolution ( $\leq 32 \times 32 \times 3$ ) images with the DCGAN architecture were conducted on workstations with one of two configurations: (a)  $4 \times$  NVIDIA 2080Ti GPUs with 11 GB visual RAM (VRAM) each, and 256 GB system RAM; and (b)  $2 \times$  NVIDIA 3090 GPUs with 24 GB VRAM and 256 GB system RAM. The high-resolution experimentation involving PGGAN or StyleGAN was carried out on workstations with one of the two configurations: (i) NVIDIA DGX with  $8 \times$  Tesla V100 GPUs with 32 GB VRAM each, and 512 GB system RAM; and (ii)  $8 \times$  NVIDIA A6000 GPUs with 48 GB VRAM each, and 512 GB system RAM. The memory requirements and training times for StyleGAN and PGGAN variants are on par with training times reported for the baselines [27, 12].

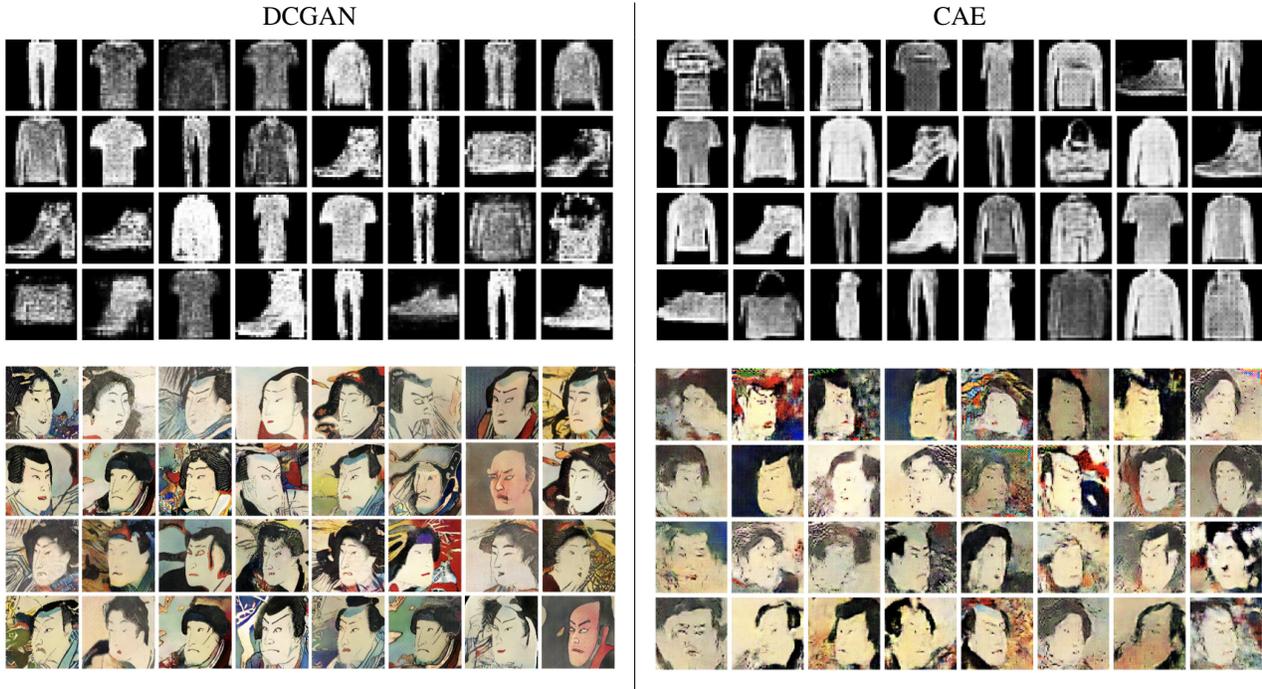


Figure 7. Images generated by Spider GAN on Fashion-MNIST and Ukiyo-E Faces, given the friendliest neighbor input as identified by SID. Both CAE and DCGAN result in images of comparable visual quality on Fashion-MNIST. However, for high-resolution image generation on 256-dimensional Ukiyo-E Faces, the fully convolutional structure of the CAE generator result in images of poorer visual quality than those generated by DCGAN.

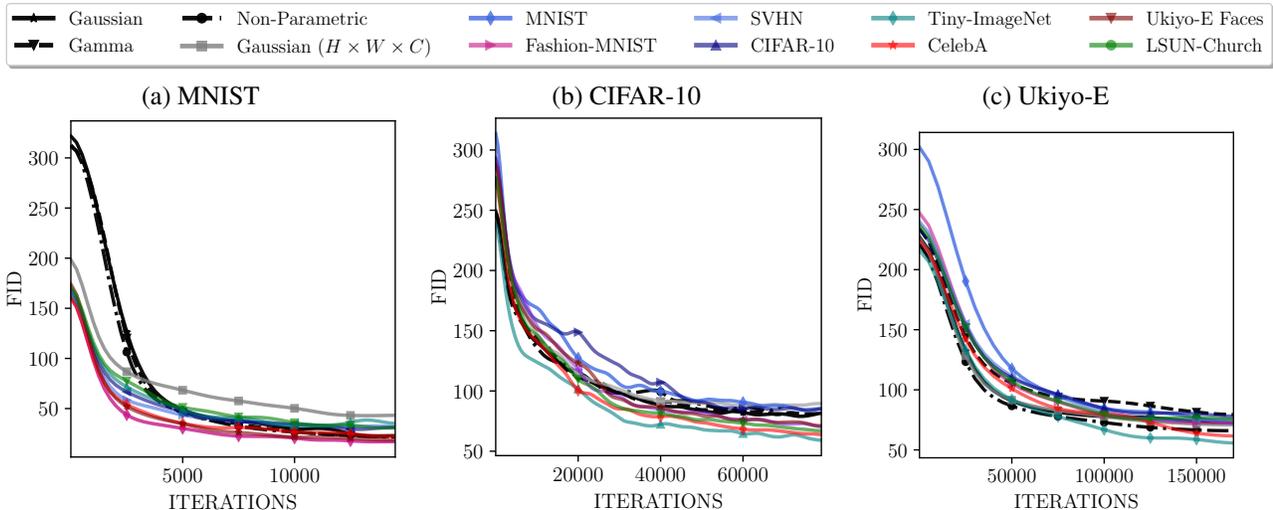


Figure 8. FID versus iterations for training baseline and Spider GAN variants. Spider GAN trained with the friendliest neighbor identified in Section 3 (of the *Main Manuscript*) result in the best (lowest) FID scores. On MNIST, Spider DCGAN approaches converge an order faster than the baseline counterparts.

Table 5. Comparison of FID, KID and  $CSID_m$  for the Spider DCGAN and baseline variants on Fashion-MNIST, SVHN, Tiny-ImageNet, and CelebA datasets. Spider DCGANs with *friendly neighborhood* inputs outperform the baselines with parametric and non-parametric priors. The performance of Spider DCGAN with MNIST or Fashion-MNIST as the input is sub par when the target is a color-image dataset.

Input Distribution		Fashion-MNIST			SVHN			Tiny-ImageNet			CelebA		
		FID	KID	$CSID_m$	FID	KID	$CSID_m$	FID	KID	$CSID_m$	FID	KID	$CSID_m$
Baselines	Gaussian ( $\mathbb{R}^{100}$ )	76.60	0.0557	22.24	135.4	0.1245	30.02	89.94	0.0657	18.06	50.32	0.0554	24.31
	Gamma ( $\mathbb{R}^{100}$ )	65.36	0.0513	<u>19.72</u>	130.8	0.1181	27.13	83.33	0.0536	14.63	40.69	0.0544	20.98
	Non-Parametric ( $\mathbb{R}^{100}$ )	<u>62.42</u>	<u>0.0426</u>	21.96	107.2	0.1053	33.52	<u>82.37</u>	<u>0.0579</u>	13.25	40.41	0.0543	72.18
	Gaussian ( $\mathbb{R}^{H \times W \times C}$ )	119.2	0.0905	28.96	113.7	0.1121	31.45	103.0	0.0844	15.62	83.61	0.0912	113.4
Spider GAN	MNIST	<b>56.59</b>	<b>0.0387</b>	<b>18.50</b>	<b>95.71</b>	<b>0.0817</b>	<b>20.62</b>	96.91	0.0669	14.95	40.78	0.0595	32.70
	Fashion MNIST	–	–	–	115.0	0.1096	32.57	108.8	0.0667	<u>13.06</u>	<u>35.18</u>	0.0574	23.98
	SVHN	79.14	0.0526	24.67	–	–	–	98.11	0.0655	15.62	40.27	0.0575	<u>20.64</u>
	CIFAR-10	92.60	0.0658	30.21	<u>101.8</u>	<u>0.0998</u>	32.40	98.22	0.0642	17.90	36.16	<u>0.0508</u>	22.16
	TinyImageNet	130.5	0.0883	22.26	111.7	0.1082	31.77	–	–	–	<b>29.47</b>	<b>0.0468</b>	<b>18.16</b>
	CelebA	81.38	0.0604	24.73	108.9	0.1029	<u>22.77</u>	<b>75.68</b>	<b>0.0511</b>	<b>12.42</b>	–	–	–
	Ukiyo-E	66.90	0.0475	23.29	114.8	0.1145	38.28	88.51	0.0612	16.01	39.41	0.0630	28.23
	LSUN-Churches	102.9	0.0774	33.87	106.8	0.1020	26.52	92.86	0.0697	15.98	53.01	0.0636	25.72

## D. Additional Experimentation on Spider GAN

We now discuss additional experimental results and ablation studies on various *Spider GAN* flavors presented in the *Main Manuscript*. One could also extend the *Spider* philosophy to VQGAN [28, 29] or diverse class-conditional models [30, 31]

### D.1. Exploring Generator Architectures

We now discuss the choice of the generator architecture in Spider GAN (cf. Section 4 of the *Main Manuscript*). We consider two network architectures:

- **DCGAN**: We consider standard DCGAN where the images from the friendly neighborhood are resized, vectorized, and provided as input to generator as described in Appendix C.1.
- **Convolutional autoencoder (CAE)**: In this setup, the images are resized to  $16 \times 16 \times 3$  and provided as input to convolutional layers to learn a low-dimensional latent representation. The output image is generated by deconvolution layers.

Table 6. A comparison of FID and KID in Spider GAN for various noise perturbations considered when the input dataset is Ukiyo-E Faces. Gaussian perturbations such as  $\mathcal{N}(\mathbf{0}, 0.25\mathbb{I})$  and  $\mathcal{N}(\mathbf{0}, 0.1\mathbb{I})$  that are concentrated about their mean result in the best performance improvements over the baseline Spider GAN.

Input Distribution	Fashion-MNIST		CIFAR-10	
	FID ↓	KID ↓	FID ↓	KID ↓
Ukiyo-E Faces	55.1200	0.0376	74.7085	0.0518
Ukiyo-E + $\mathcal{N}(\mathbf{0}, 0.1\mathbb{I})$	<b>47.2873</b>	<b>0.0285</b>	<u>70.101</u>	0.0488
Ukiyo-E + $\mathcal{N}(\mathbf{0}, 0.25\mathbb{I})$	<u>50.2150</u>	0.0345	<b>68.7473</b>	<b>0.0473</b>
Ukiyo-E + $\mathcal{N}(\mathbf{0}, \mathbb{I})$	79.8415	0.0690	71.9181	0.0531
Ukiyo-E + Gamma noise	51.8201	0.0343	70.362	<u>0.0476</u>
Ukiyo-E + Non-parametric noise	50.8536	<u>0.0329</u>	72.9138	0.0495

The number of trainable parameters are fewer for the CAE architecture than the DCGAN approach in both cases. Figure 7 shows the output images generated by these approaches considering the friendliest neighbor (as suggested by Tables 1-4) provided as input when learning the Fashion-MNIST and Ukiyo-E Faces datasets. We observe that the CAE based Spider GAN outperforms the DCGAN approach on Fashion-MNIST. However, on higher resolution images, multiple visual artifacts were found as a consequence of the fully convolutional architecture. We observed similar degradation in image quality when training Spider GAN with CAE on other high-resolution datasets such as CelebA. We therefore consider the DCGAN approach in the experiments presented in Section 4 of the *Main Manuscript* and Appendix D.2.

## D.2. Additional Experiments on Spider DCGAN

We now present results on additional experimental validation run on the Spider DCGAN architecture. The experimental setup is the same as the one described in Appendix C.1. First, we consider training Spider GAN on Fashion-MNIST, SVHN, Tiny-ImageNet and 64-dimensional CelebA datasets. The FID and KID of the converged models are presented in Table 5. On the Fashion-MNIST, SVHN, and CelebA datasets, we observe that the Spider GAN approach with the friendliest neighbor (as identified by FID, KID and  $CSID_m$ ), results in improved learning over the baselines. On the Tiny-ImageNet learning task, we observe that a source dataset with less diversity (such as CIFAR-10, as suggested by FID and KID) performs poorly, while a more diverse source dataset, such as CelebA, improves the best-case FID over the baselines. These results validate the *friendly neighborhood* of Tiny-ImageNet identified using  $CSID_m$  in Section 3 of the *Main Manuscript*, where CIFAR-10 and LSUN-Churches are less diverse, having a negative  $CSID_m$ . Figures 10-16 present the images generated by Spider GAN and the baseline variants on various datasets considered. Figure 8 presents the convergence of FID as a function of iterations for the remaining source dataset combinations of Spider GAN models considered in Section 4 and Figure 4 of the *Main Manuscript*.

### D.2.1 Noise Perturbations on the Input Dataset

The SpiderGAN framework relies on the variability present in the chosen input dataset to learn the target better. As discussed in Section 4 of the *Main Manuscript*, we considered addition of noise to the dataset input to the Spider GAN generator when the cardinality of the input is small. We observed that CelebA or Tiny-ImageNet are more diverse and perform better than small datasets such as Ukiyo-E Faces. To overcome the lack of diversity in small datasets, we consider additive-noise perturbations to augment the data. While Gaussians are a popular choice, we also consider the Gamma density and non-parametric densities to generate noise, which are known to improve the performance of the GANs on latent-space interpolation. We consider three Gaussian examples: the standard normal  $\mathcal{N}(\mathbf{0}, \mathbb{I})$ ,  $\mathcal{N}(\mathbf{0}, 0.25\mathbb{I})$ , and  $\mathcal{N}(\mathbf{0}, 0.1\mathbb{I})$ . Three variances are considered to highlight the trade-off between generating noisy images (Gaussians with high variance) and low diversity in the input dataset (Gaussians with low variance). We present results on learning MNIST and CIFAR-10 datasets with Ukiyo-E Faces dataset as input.

**Results:** Figures 17(a)-(f) show the images generated by Spider GAN with various noise perturbations applied to Ukiyo-E Faces. Adding Gaussian noise with a small variance, or Gamma distributed noise results in diverse images and better visual quality of generated images. On the other hand, models trained with the standard normal or non-parametric densities resulted in poor learning, with several out-of-distribution images. The performance of the converged models in presented in Table 6.

Perturbations that are Gaussian, and concentrated about the mean, such as  $\mathcal{N}(\mathbf{0}, 0.1\mathbb{I})$  or  $\mathcal{N}(\mathbf{0}, 0.25\mathbb{I})$  resulted in the lowest FID and KID. Therefore, Gaussian perturbations with a small variance result in better performance when the input datasets have small cardinality.

Table 7. Comparison of sharpness metric evaluated on interpolated images in MNIST, CIFAR-10 and Ukiyo-E learning tasks. The benchmark sharpness is computed on target data samples. Spider GAN variants outperform the baselines on CIFAR-10 and MNIST, while being on par with the non-parametric prior on the Ukiyo-E Faces. The values shown in **bold** are closest to the benchmark sharpness.

Input Distribution		Sharpness of the Interpolated Image		
		MNIST	CIFAR-10	Ukiyo-E
Baselines	Gaussian	0.0868	0.587	1.730
	Gamma	0.0536	1.217	1.981
	Non-parametric	0.2522	0.785	<b>2.538</b>
SpiderGAN	MNIST	–	0.467	2.008
	Fashion MNIST	<b>0.1408</b>	0.377	1.353
	SVHN	0.0898	1.214	1.480
	CIFAR-10	0.0859	–	2.533
	TinyImageNet	0.0623	<b>0.906</b>	1.274
	CelebA	0.1735	0.449	2.104
	Benchmark	0.1396	0.993	2.748

Table 8. Comparison of *Interpolation FID* and *Interpolation KID* for the Spider GAN and baseline variants on MNIST, CIFAR-10, and Ukiyo-E Faces datasets. The input provided to the generator  $z_{in} = \frac{z_1+z_2}{2}$ ;  $z_1, z_2 \sim p_Z$  is the mid-point between two samples drawn from the input distribution  $p_Z$ , either of parametric form in the case of the baselines, or the *friendly neighborhood* datasets, in the case of Spider GAN. The values in the parentheses indicate the relative increase in the FID/KID scores, in comparison to those reported in Table 2 of the *Main Manuscript*. Spider GANs with *friendliest neighborhood* input datasets achieve FID and KID scores on par with the best-case baseline.

Input Distribution		MNIST		CIFAR10		Ukiyo-E Faces	
		FID	KID	FID	KID	FID	KID
Baselines	Gaussian ( $\mathbb{R}^{100}$ )	25.111 (+16.8%)	0.0181 (+30.2%)	121.198 (+68.7%)	0.0848 (+36.9%)	74.241 (+3.1%)	0.0612 (+14.4%)
	Gamma ( $\mathbb{R}^{100}$ )	23.564 (+11.3%)	0.0149 (+12.1%)	77.113 (+6.1%)	0.0492 (+1.8%)	70.302 (+0.4%)	0.0558 (+18.7%)
	Non-Parametric ( $\mathbb{R}^{100}$ )	22.301 (+6.4%)	0.0142 (+3.6%)	87.478 (+16.7%)	0.0568 (+7.1%)	<b>66.022</b> (+1.0%)	<b>0.0434</b> (+3.0%)
Spider GAN	MNIST	–	–	122.084 (+71.2%)	0.0790 (+47.5%)	103.80 (+51.4%)	0.0732 (+67.1%)
	Fashion MNIST	<b>20.644</b> (+22.8%)	<b>0.0147</b> (+42.7%)	113.109 (+46.7%)	0.0731 (+32.9%)	89.901 (+23.6%)	0.0654 (+43.7%)
	SVHN	27.630 (+1.8%)	0.0208 (+1.5%)	89.161 (+39.1%)	0.0558 (+23.7%)	77.302 (+10.0%)	0.0542 (+12.4%)
	CIFAR-10	30.214 (+3.4%)	0.0305 (+38.6%)	–	–	87.981 (+24.2%)	0.0621 (+17.1%)
	TinyImageNet	46.233 (+41.6%)	0.0397 (+50.4%)	86.708 (+47.3%)	<b>0.0520</b> (+70.4%)	79.848 (+28.9%)	0.0565 (+29.5%)
	CelebA	21.517 (+4.6%)	0.0152 (+5.5%)	<b>86.475</b> (+43.9%)	0.0534 (+23.0%)	68.849 (+27.2%)	0.0449 (+10.1%)
	Ukiyo-E	38.950 (+26.9%)	0.0318 (+39.4%)	98.045 (+60.6%)	0.0671 (+83.8%)	–	–

## D.2.2 Input-space Interpolation with Spider DCGAN

Gamma and non-parametric priors were introduced to the GAN landscape to improve the quality of interpolated images in GANs [15, 16]. We compare the image interpolation quality of Spider GAN with respect to the gamma and non-parametric baselines. The experimental setup is similar to that in Appendix C.1. We compare the visual quality of images generated by interpolated inputs to the generator. In the baseline GANs, we provide the generator with eight linearly interpolated points between two random samples drawn from the prior densities. In the case of Spider GAN, we draw two random samples from the input dataset, and generate eight linearly interpolated images that are input to the Spider GAN generator. The quality of the interpolation is evaluated in terms of the sharpness metric. We present results on MNIST, CIFAR-10, and Ukiyo-E Faces.

Figures 18-20 present the images generated by the interpolated input vectors by the three baseline GAN variants and Spider GAN with the three friendliest neighbors as the input datasets. We observe that, Spider GAN, although not trained for the task, is able to generate realistic interpolated images. The visual quality is on par with the non-parametric interpolation scheme in the case of MNIST, and superior to the baselines on the Ukiyo-E Faces learning task. All variants fail to generate realistic images on CIFAR-10. Table 7 shows the sharpness metric computed on the interpolated images. We observe that Spider GAN variants attain values closer to the benchmark in comparison with the baselines. As discussed in the Main Manuscript, the best performance of Spider GAN is achieved when the input dataset is the *friendliest neighbor* of all the target datasets under consideration. Table 8 presents the FID and KID scores of the Spider GAN and baseline variants, when computed on a batch of  $10^4$  samples obtained by providing the mid-point sample  $z_{in} = \frac{z_1+z_2}{2}$ ;  $z_1, z_2 \sim p_Z$  as input to the generator. The inputs  $z_1$  and  $z_2$  are samples drawn from parametric distributions as in the case of the baselines, or images from the *friendly neighborhood* input dataset as in the case of Spider GAN. Table 8 also shows the relative increase in FID and KID compared to those obtained when unaltered samples drawn from  $p_Z$  are provided as input to the generator (cf. Table 2 of the *Main Manuscript*). Across all the datasets considered, we observe that Spider GAN variants with the *friendliest neighbor* input result in a performance comparable with the best-case baselines in terms of FID and KID. However, the baselines GAN with the non-parametric or gamma-distributed priors, which are designed to minimize the interpolation error [15, 16], and consequently, result in lower relative change in the scores. The results suggest that, while Spider GAN is superior to Gaussian latent spaces, a trade-off exists between the interpolated image quality offered by non-parametric or gamma priors, and the overall superior performance offered by Spider GAN. A detailed discussion on the input-space control over the generated images is discussed in the context of Spider StyleGAN2-ADA in Appendix D.5.1

## D.2.3 Impact of Diversity and Dataset Bias on Spider GANs

The *friendly neighbourhood* of a target in SpiderGAN is chosen based on the SID metric, which compares the distance between data manifolds. SpiderGAN does not enforce image-level structure to learn pairwise transformations. We therefore expect that the diversity of the source dataset (such as racial or gender bias) should not affect the diversity in the learnt distribution. To demonstrate this, consider the task of learning Ukiyo-E faces dataset with CelebA dataset as input. We consider three variants of CelebA – (i) The entire dataset of  $2 \times 10^5$  images, comprising an even split of the *male* and *females* classes; (ii) Only the *female* class comprising  $10^5$  images; and (iii) A simulated imbalance, created by including the entire *male* class and 200 images from the *female* class. The input resolution is  $64 \times 64$ , while the output resolution is set to  $128 \times 128$ . The models are trained using the DCGAN architecture with hyperparameters as described in Appendix C.1. All the models are trained for  $10^5$  generator iterations.

The images output by the Spider GAN model in each case are presented in Figure 21 (a.1-a.3). We did not observe bias in the images generated by the three models. To demonstrate this further, we compared the Spider GAN outputs for the same 20 samples of the *female* class images provided as input (cf. Figure 21 (a)). The results indicate that, while correspondence between images is not learnt, the bias in the source dataset of the generator in SpiderGAN does not affect the target diversity. The bias in these datasets is neither leveraged, nor exemplified by Spider GAN.

## D.2.4 Mode Coverage in Spider GANs

In order to evaluate the mode coverage in Spider GAN learning, consider the *partial MNIST* experiment proposed by Zhong *et al.* [32] involving the 11-class augmented Fashion-MNIST dataset consisting of an additional 100 images drawn from from the *digit 1* class of MNIST. We train SpiderGAN on the Fashion-MNIST dataset with the CAE architecture (cf. Appendix D.1). We consider two input datasets: CIFAR-10 and Tiny-ImageNet.

In order to evaluate mode coverage, the trained GAN generators are compared on the ability to faithfully generate samples from the underrepresented *digit 1* class. For evaluation, a 11-class fully-connected classifier is trained on the augmented

dataset consisting of all 10 classes from Fashion-MNIST and the entire *digit 1* class from MNSIT. Following the approach presented in [32], the GANs are evaluated by sampling a batch of  $9 \times 10^5$  images, and computing the number of instances of *digit 1* generated, as indicated by the output of the classifier. We compare against the DCGAN, AdaGAN [33] and the GAN with mixture of generators (MixGAN) [32]. The images from *digit class 1* generated by the Spider GAN variants are presented in Figure 9, while Table 9 summarizes the performance of the baseline and Spider GAN models. The results highlight the need for class diversity in the input dataset. When SpiderGAN is trained with CIFAR-10, consisting of fewer classes than the target, the minority *digit 1* class is poorly represented. On the other hand, for Spider GAN with Tiny-ImageNet or CelebA as the input, the minority class is generated faithfully.

### D.2.5 Learning the Identity Mapping

Based on the intuition that GAN generators perform entropy minimization [34], we expect the generator to learn an identity mapping when the same dataset is provided as both input and output. To validate this, we consider the Fashion-MNIST learning task with the DCGAN architecture. We considered all four combinations of adding noise to the input or target datasets. The learnt input-output pairs are presented in Figure 22. In all four scenarios, although pairwise consistency is not explicitly enforced, it was discovered by Spider GAN, resulting in a GAN generator that approximates an identity function. When the input and output datasets are both noisy, the generator attempts to retain the noise in the generated images. However, when the input dataset is clean but the target dataset incorporates noise, artifacts are introduced in the generated images as the models attempts to create noise (which has a higher entropy than the dataset).

Table 9. Mode coverage of Spider GAN in comparison to baseline GANs on the *Fashion-MNIST and partial MNIST* experiment. The \* indicates values reported by Zhong *et al.* [32]. The measure *#1s* indicates the number of the samples from the *digit class 1* predicted in a batch of  $9 \times 10^5$  samples drawn the generator. *Avg. Prob.* denotes the average classification probability of *digit class 1* samples output by a pre-trained classifier. Spider GAN trained with an input dataset that posses higher diversity than the target, such as Tiny-ImageNet, outperforms the baselines.

Measure (↑)	DCGAN*	AdaGAN*	MixGAN*	Spider GAN (CIFAR-10 Source)	Spider GAN (Tiny-ImageNet Source)
<i>#1s</i>	13	60	289	201	<b>345</b>
<i>Avg. Prob.</i>	0.49	0.45	0.69	0.81	<b>0.89</b>

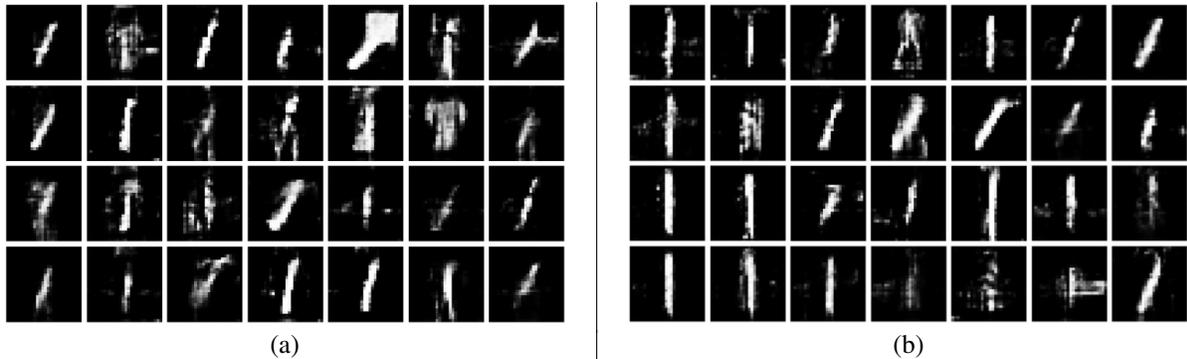


Figure 9. Images from the *digit class 1* generated by Spider GAN with input images drawn from (a) CIFAR-10, and (b) Tiny-ImageNet datasets. The samples were identified based on the output of a pre-trained 11-class classifier network. Spider GAN with an input class diversity lower than the target (CIFAR-10 dataset) generated images of inferior quality in comparison to the Spider GAN trained on a more diverse input dataset such as Tiny-ImageNet.

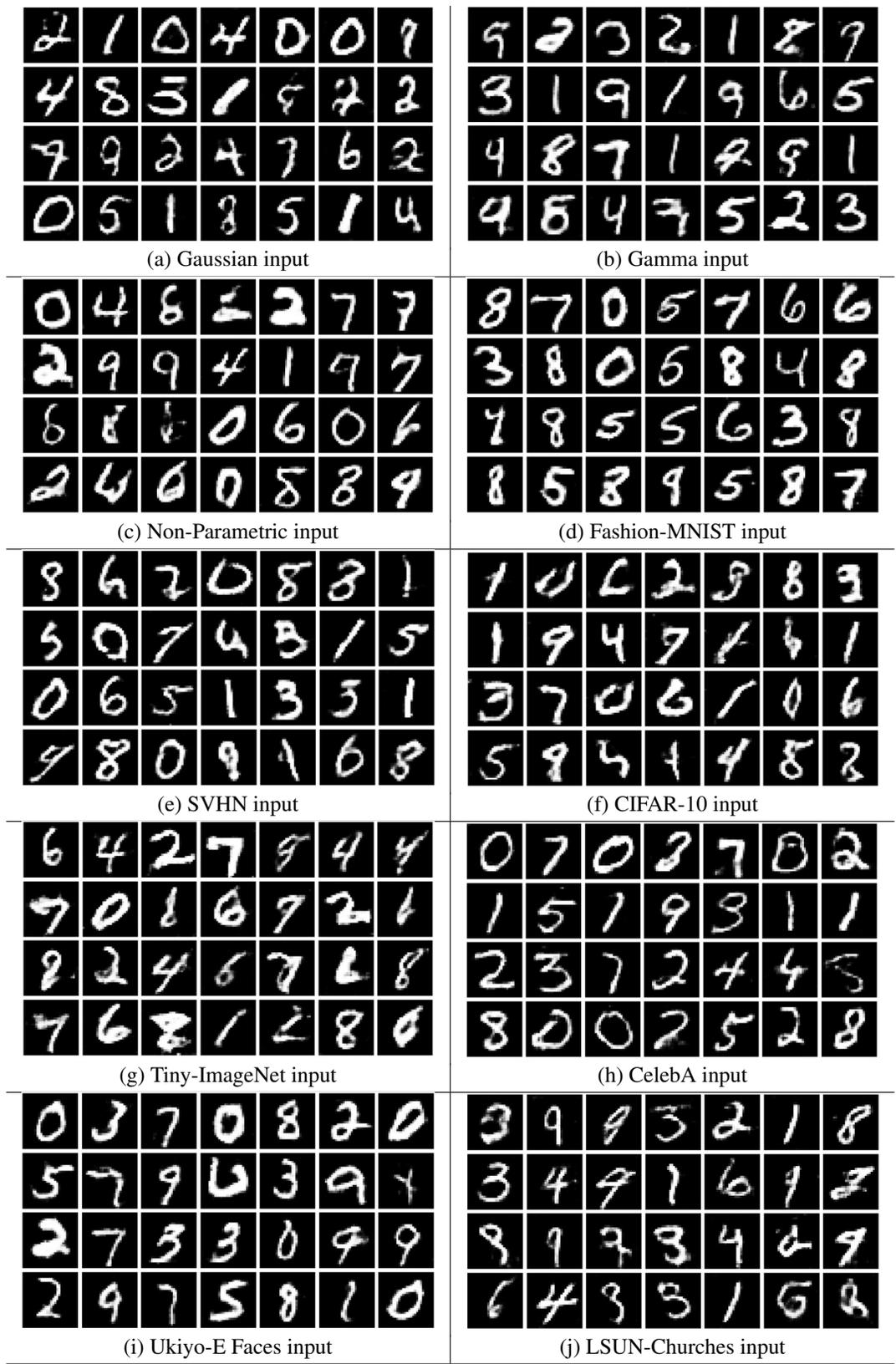


Figure 10. Images generated by the baseline GAN and Spider GAN for various input distributions, with MNIST being the target. Spider GAN trained with Fashion-MNIST input (the friendliest neighbor of MNIST as identified by SID) generates sharper output images.

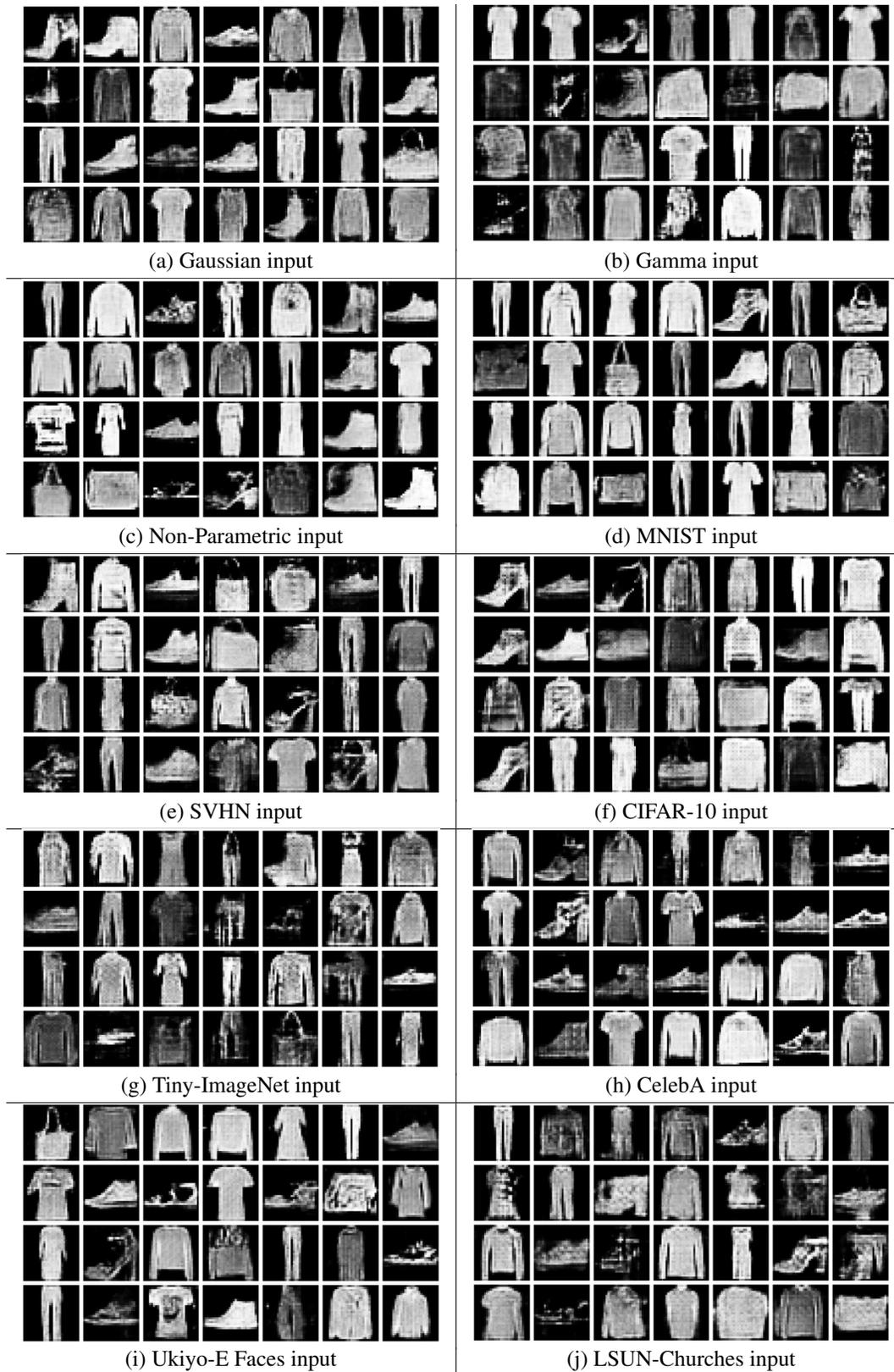


Figure 11. Images generated by the baseline GAN and Spider GAN for various input distributions, with Fashion-MNIST chosen as the target. A poor choice of the input distribution results in a suboptimal generator that outputs low-quality images. For instance, the output generated for inputs coming from CelebA or a non-parametric distribution.



Figure 12. Images generated by the baseline GAN and Spider GAN for various input distributions, when trained with SVHN as the target. A poor choice of the input distribution results in low-quality images output by the generator.

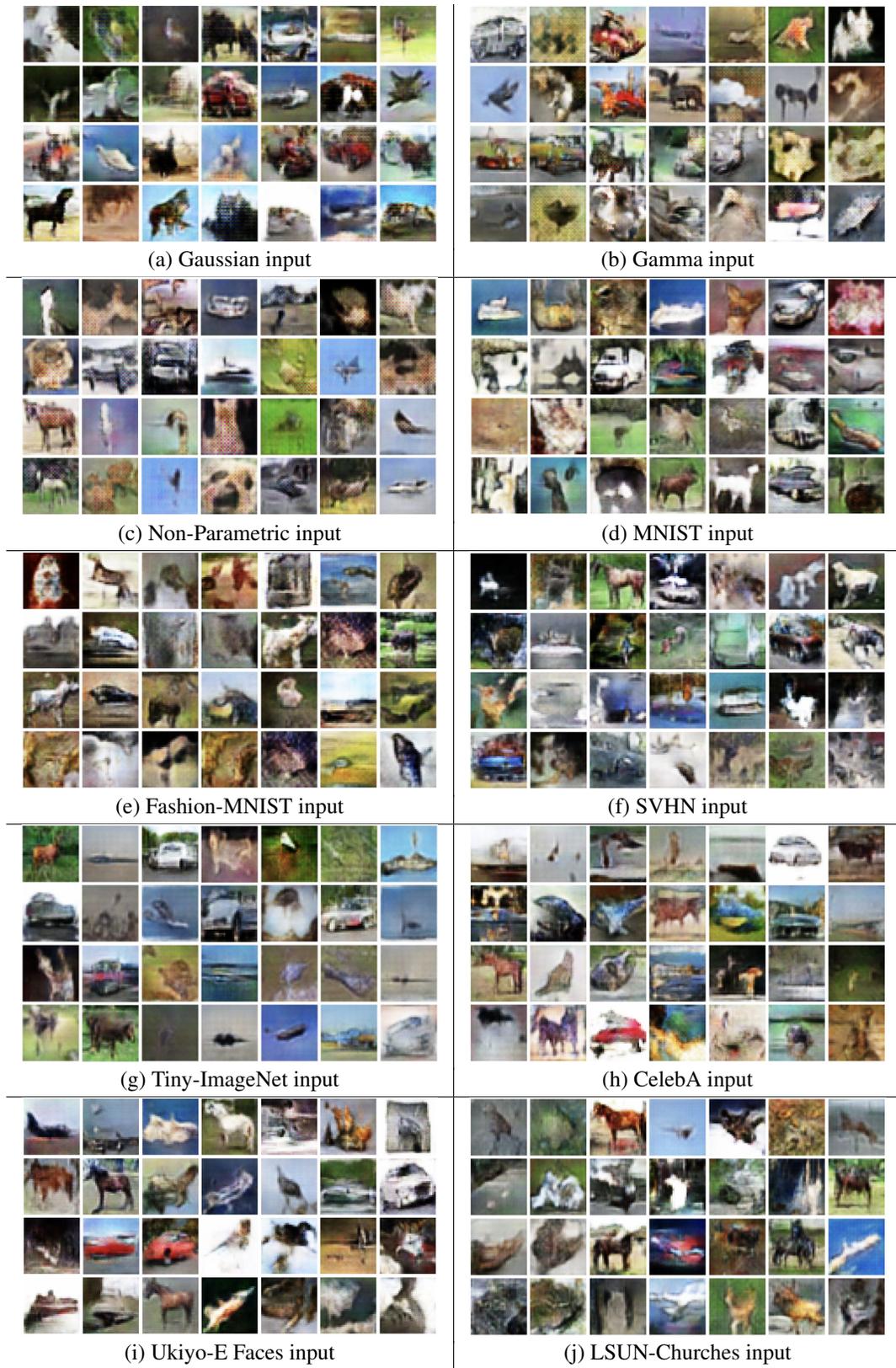


Figure 13. Images generated by the baseline GAN and Spider GAN with CIFAR-10 as the target, for various input distributions. While some classes, such as the *horse*, *car* or *boat* are well generated by all GAN, neither the baseline GANs nor the Spider GANs are able to reliably learn all the classes in CIFAR-10.

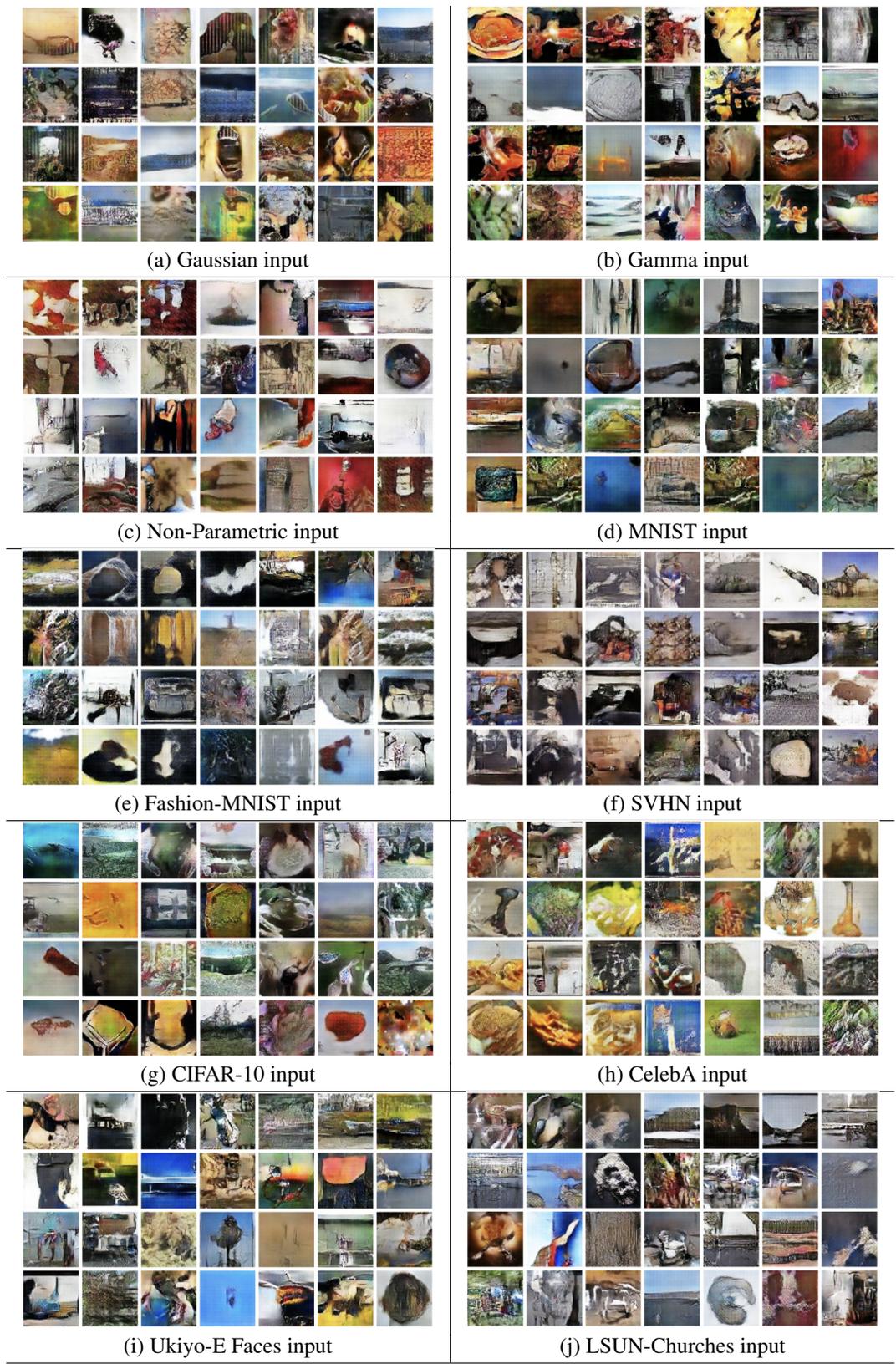


Figure 14. Images generated by the baseline GAN and Spider GAN on Tiny-ImageNet as the target, for various input distributions as indicated. While Spider GAN approaches achieve a lower FID than the baselines on this task, none of the GAN variants generate realistic output images.

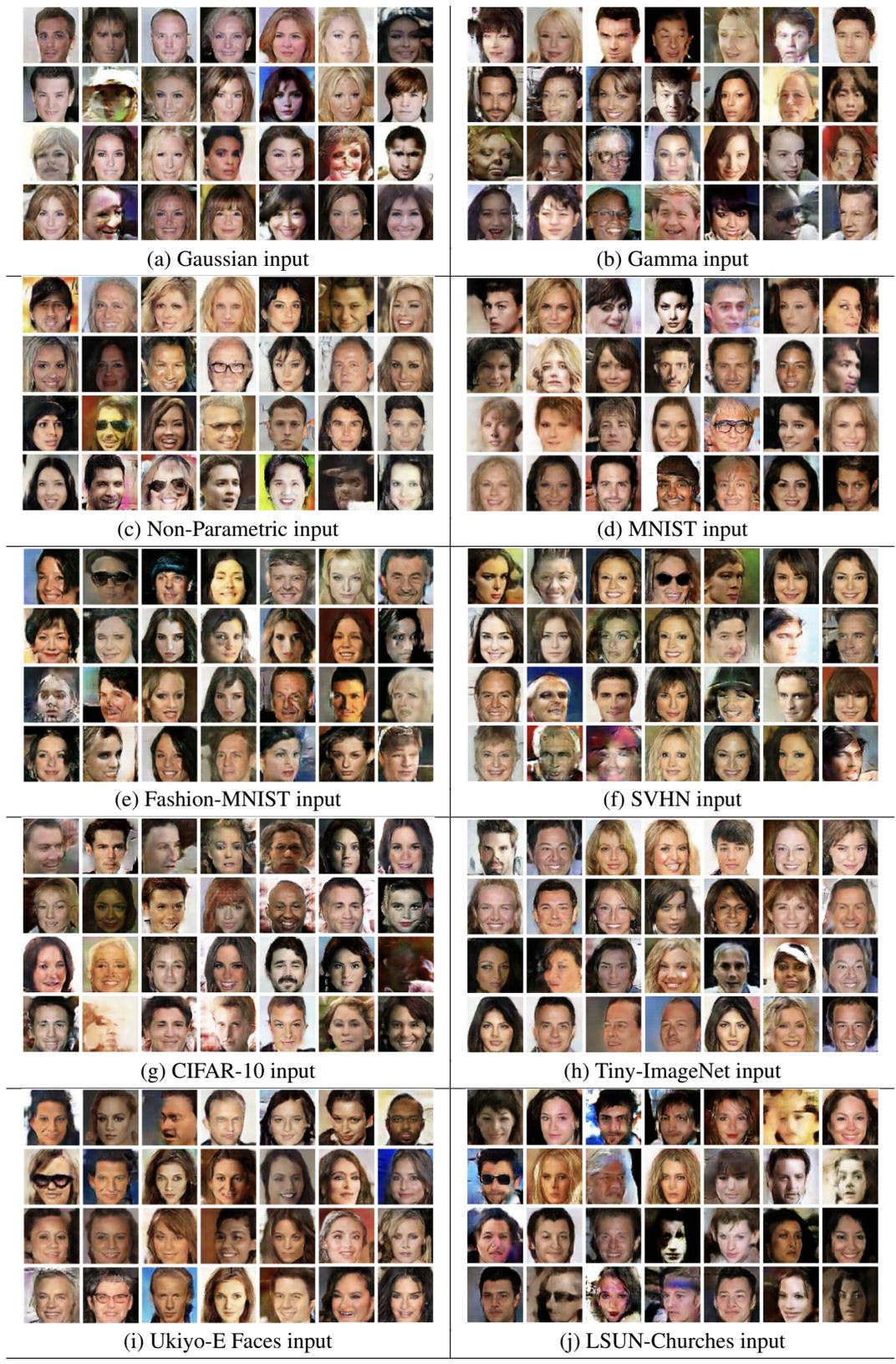


Figure 15. Images generated by the baseline GAN and Spider GAN on the low resolution CelebA ( $64 \times 64$ ), given various input distributions. Images generated by Spider GAN trained with Tiny-ImageNet and Ukiyo-E Faces as the input outperform other GAN flavors.



Figure 16. Images generated by the baseline GAN and Spider GAN variants on the Ukiyo-E Faces for different inputs to the generator. Images generated by Spider GAN with Tiny-ImageNet or CelebA images as input results in sharper images in comparison to the baselines.



Figure 17. Images generated by Spider GAN on Fashion-MNIST and CIFAR-10, when trained with the Ukiyo-E Faces as input. Ukiyo-E Faces are perturbed mildly with various parametric noise sources to enhance the input diversity. Gaussian perturbations result in superior image quality compared to the rest.

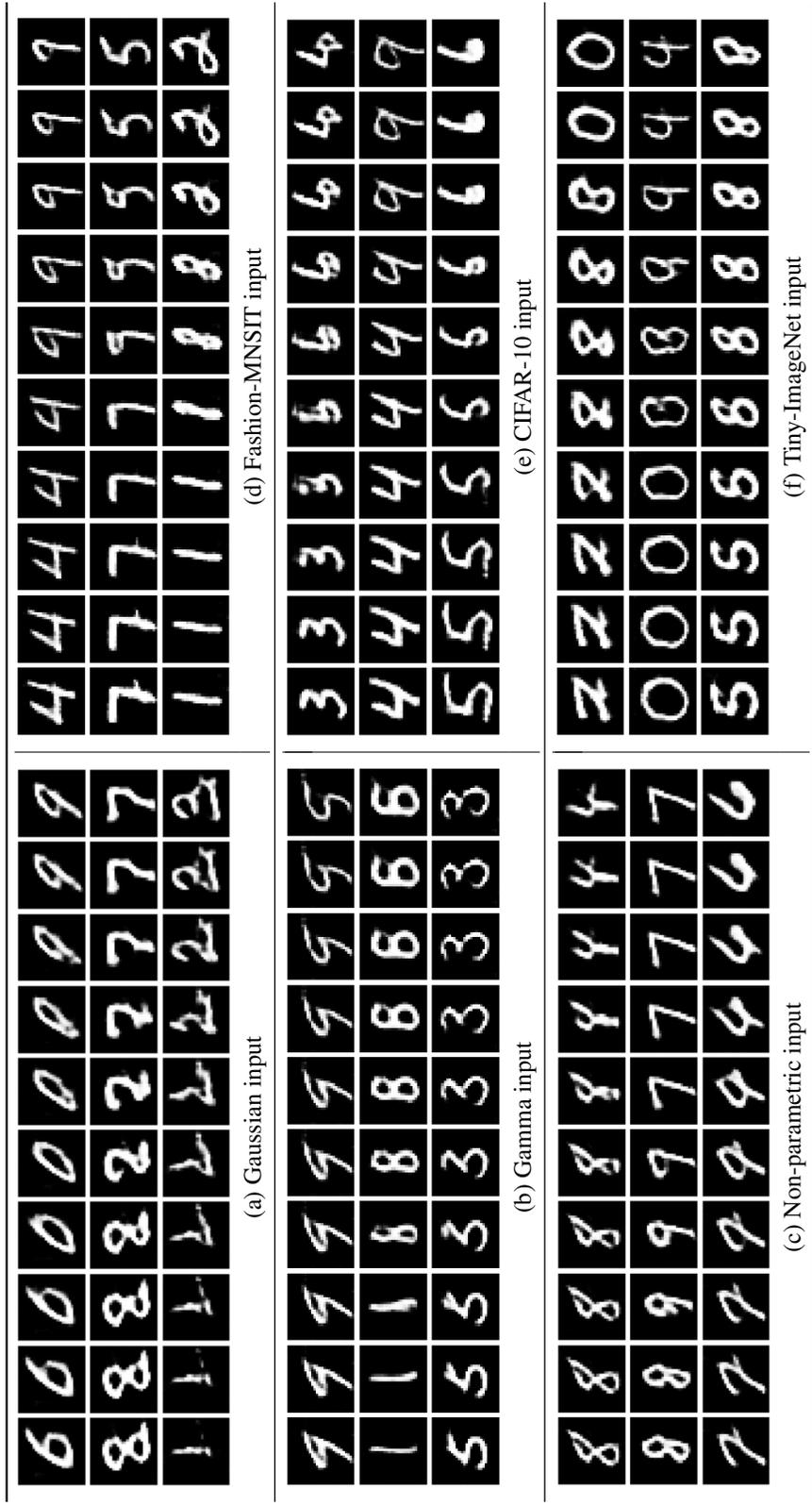


Figure 18. Input-space interpolation on the baseline and Spider GAN variants trained on the MNIST dataset. Interpolation with Fashion-MNIST input for Spider GAN results in output images that transition smoothly, compared to the baselines.

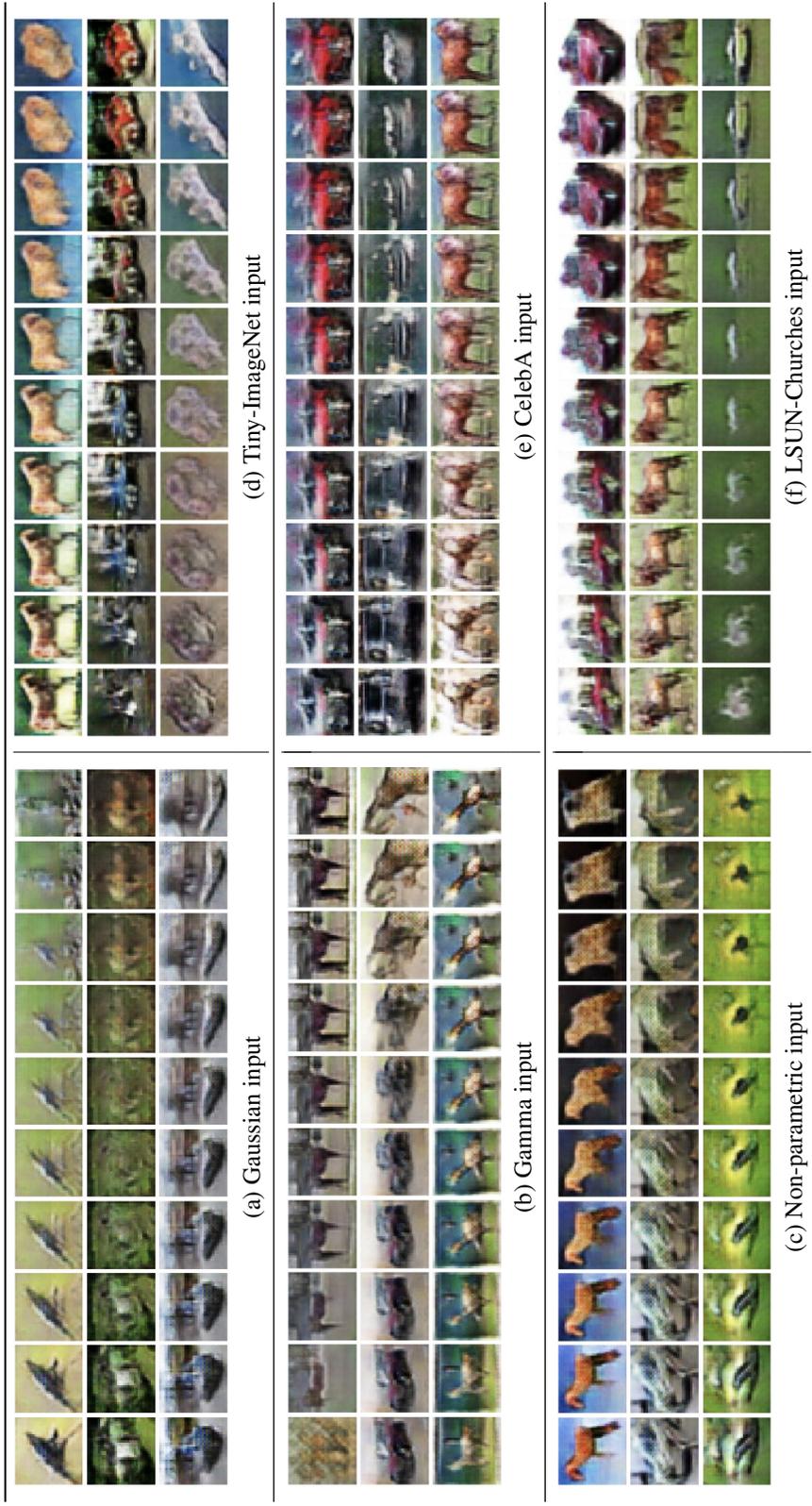


Figure 19. (Color online) Input-space interpolation on the baseline GANs and Spider GAN trained on the CIFAR-10 dataset. Interpolation with Tiny-ImageNet input for Spider GAN result in output images of superior quality. However, all variants fail to create realistic images when provided with interpolated input samples.

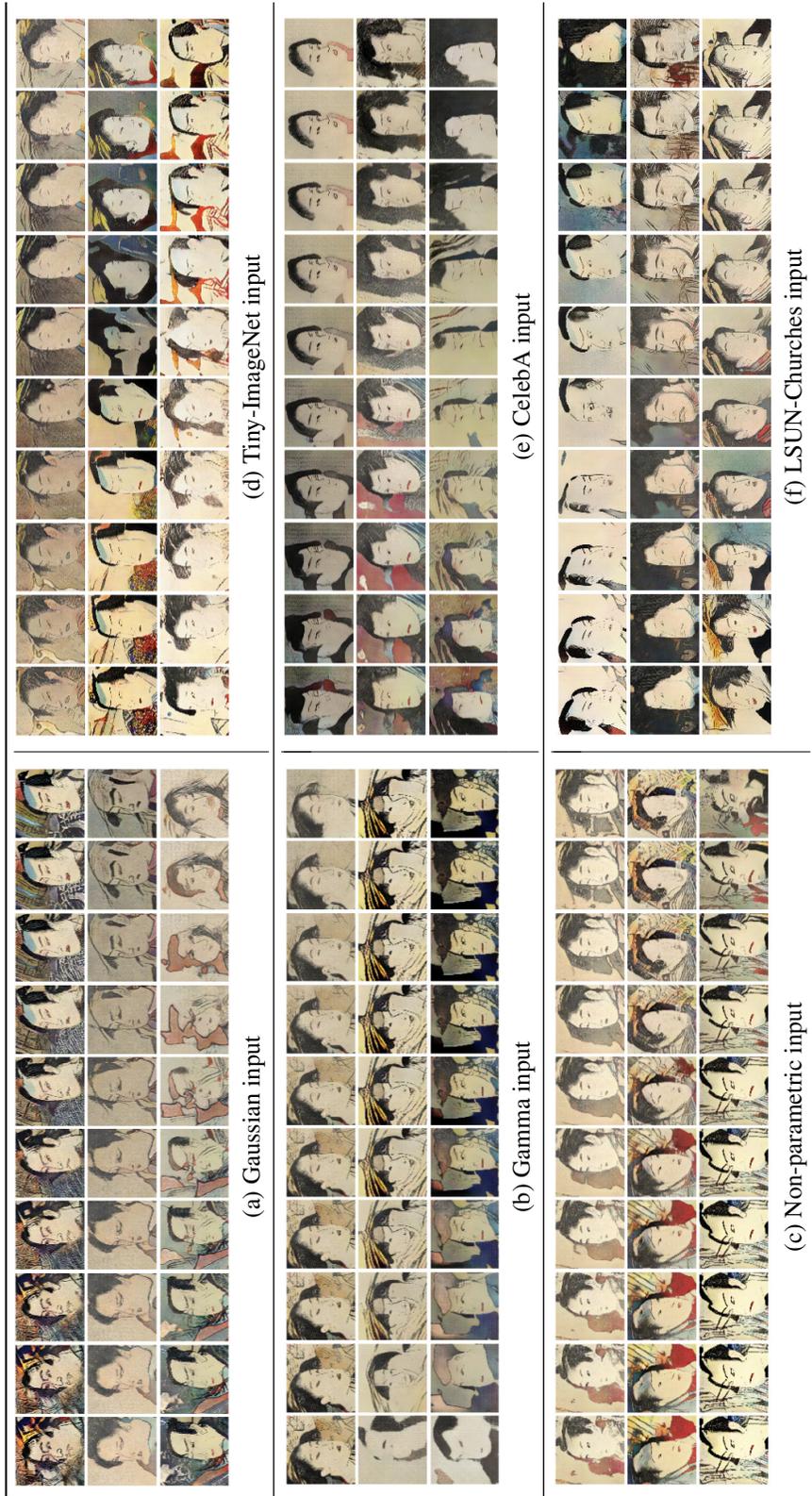


Figure 20. (Color online) Input-space interpolation on the baseline and Spider GAN variants trained on the Ukiyo-E Faces. Interpolations with CelebA input for Spider GAN are the smoothest. Baseline variants result in sharp fluctuations in the orientation of the faces, which is indicative of a non-smooth generator in the output space.



(a) Input Samples from the *Female Class* of the source CelebA dataset.



(a.1) Corresponding outputs for balanced source data.



(a.2) Corresponding outputs for source data bias: 100% *Males* class + 0.2% *Female Class*.

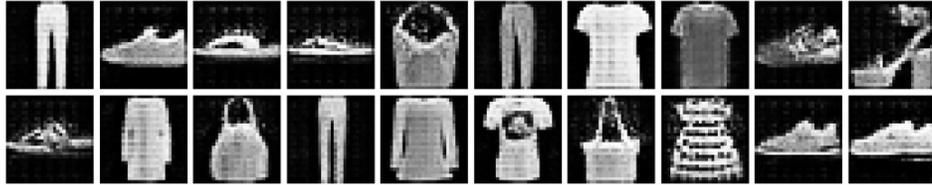


(a.3) Corresponding outputs for source data bias: 0% *Males* class + 100% *Female Class*.

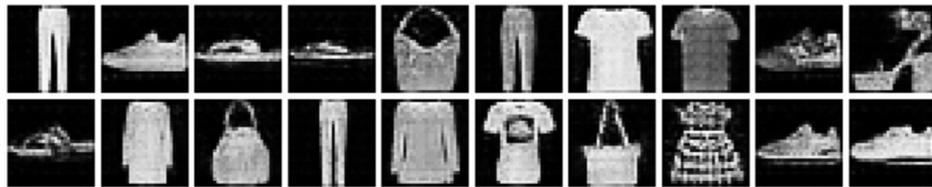
Figure 21. Images generated by Spider GAN when trained on the Ukiyo-E Faces as the target dataset, with varying levels of bias simulated in the source CelebA dataset. The output images (a.1-a.3) correspond to the generator input with the same *Females class* CelebA images depicted. The bias in the input dataset does not carry over to the generator outputs in Spider GAN formulation. Irrespective of the class imbalance in the source CelebA images, the generated Ukiyo-E Faces possess sufficient class diversity.



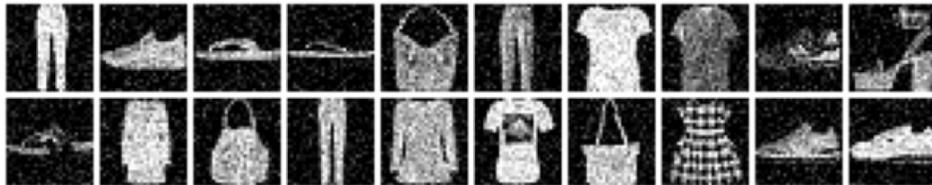
(a) Input Samples drawn from Fashion-MNIST.



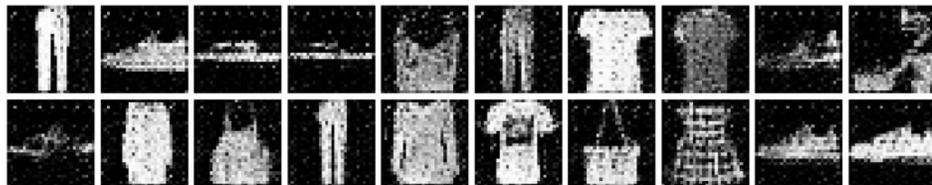
(a.1) Spider GAN output when trained on noisy Fashion-MNIST as target.



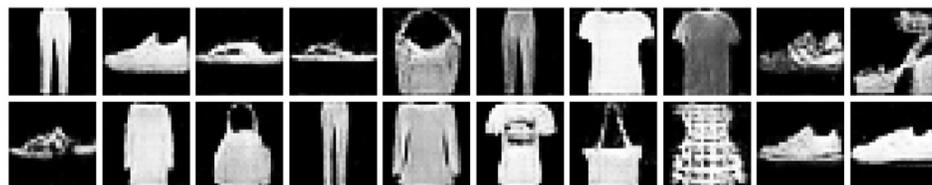
(a.2) Spider GAN output when trained on Fashion-MNIST as target.



(b) Input Samples drawn from noisy Fashion-MNIST.



(b.1) Spider GAN output when trained on noisy Fashion-MNIST as target.



(b.2) Spider GAN output when trained on Fashion-MNIST as target.

---

Figure 22. Images generated by Spider GAN when trained on various combinations of noisy and clean Fashion-MNIST images provided as the input and output to the GAN. In all scenarios, although pairwise consistency was not explicitly enforced, it was discovered by Spider GAN network. When the input and output datasets are (a.2) both clean, or (b.1) both noisy, the generator attempts to learn an identity mapping. When the input dataset is clean but the target dataset incorporates noise (a.1), we observe artifacts in the generated images. SpiderGAN with a noisy input dataset and clean target samples learns a denoising network.

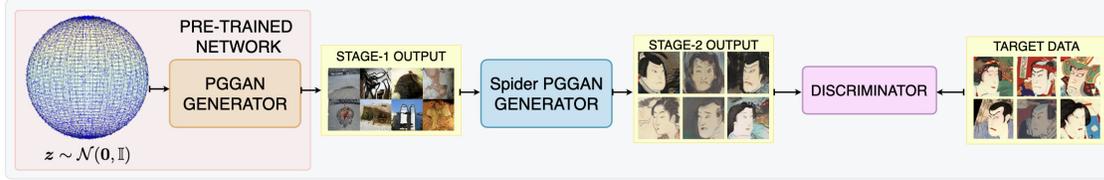


Figure 23. Spider GAN based progressively growing GAN (PGGAN) architecture. The output distribution of PGGAN trained on Tiny-ImageNet data is provided as input to the second Spider PGGAN stage that is trained to learn a high-resolution, small-sized dataset such as Ukiyo-E Faces.

### D.3. Class-conditional Spider GAN

We present a *Spider* counterpart to the auxiliary classifier GAN (ACGAN [35]) formulation, entitled Spider ACGAN. In Spider ACGAN, the discriminator not only provides a *real versus fake* classification of its input, but also provides a prediction of the class from which the sample is drawn. The discriminator is trained to minimize both the WGAN loss with the  $R_d$  penalty [18], and the classification cross-entropy loss. We consider two variants of the generator, one without class information, and the other with the class label provided as a fully-connected embedding to the input layer. The Spider ACGAN variants are compared with the un-conditioned Spider GAN baseline. We present experiments on learning Fashion-MNIST dataset with MNIST as the input. The pairwise correspondences between the input and output images are presented in Figures 24-26. While Spider ACGAN without generator embeddings is superior to the baseline Spider GAN in learning class-level consistency, mixing between the classes is not eliminated. However, with the inclusion of class embeddings in the generator, the disentanglement of classes can be achieved in Spider ACGAN.

While this experiment demonstrates the feasibility of employing Spider GAN in class-conditional settings, scenarios involving mismatch between the number of classes in the input and output datasets, is a promising direction for future research.

### D.4. Additional Experiments on Spider PGGAN

We now present additional experiments conducted with the Spider PGGAN architecture, and present the images generated by the Spider PGGAN variants. Figure 23 depicts the philosophy employed in a two-stage cascaded Spider PGGAN model considered in Section 5.1 of the *Main Manuscript*, where the input-stage PGGAN generated Tiny-ImageNet images, while the second Spider PGGAN stage transforms Tiny-ImageNet into Ukiyo-E Faces. Consider two extensions of the Spider PGGAN training algorithms: (a) The Spider PGGAN is trained on  $32 \times 32 \times 3$  CIFAR-10 data with the input images drawn from the output of a PGGAN pre-trained on Tiny-ImageNet. Additionally, weights from PGGAN pre-trained on Tiny-ImageNet are transferred to Spider PGGAN for all layers but the first because the dimensionality in the first layer does not match. The trained model achieves an FID of 9.56, which is an improvement over the base Spider GAN trained on CIFAR-10 without the weight transfer. Images generated by Spider PGGAN with weight transfer are shown in Figure 28. This suggests that other network modifications and augmentations can be used in combination with the Spider GAN framework to improve the performance of PGGAN. (b) We train the Spider PGGAN with multiple cascade layers. The output of a Stage-I PGGAN pre-trained on Tiny-ImageNet is used to train a Spider PGGAN (Stage-II) to generate CIFAR-10 images. The output of the converged second stage model is used to generate high-resolution Ukiyo-E and MetFaces images (Stage-III). The final model achieves an FID of 45.32 on MetFaces (a 12% improvement over a single-stage Spider PGGAN), and 57.63 on Ukiyo-E Faces (a 10% improvement over single stage). The MetFaces images generated by the cascade network, juxtaposed the images generated by the baseline methods are provided in Figure 32. These results suggest that having multiple stages of pre-trained networks in the Spider PGGAN, and training incrementally results in superior performance than a single-stage Spider PGGAN.

### D.5. Additional Experimental on Spider StyleGAN

The philosophy behind StyleGAN [11, 12] architectures run parallel to our proposed philosophy, where a *mapping* network is used to learn editable intermediate representations of the input noise distribution. A *synthesis* network subsequently transforms this representation into an image. The Spider GAN approach can be incorporated readily into any StyleGAN network, by replacing the input noise distribution to the mapping network with samples from the input dataset, drawn from a pre-trained GAN.

We trained the *Spider* variants of StyleGAN2, StyleGAN2-ADA [11] and StyleGAN3 [12] on the Ukiyo-E Faces, MetFaces, FFHQ, animal faces HQ Cats (AFHQ-Cats) dataset using the various combinations that included adaptive regularization and weight transfer. Across all experiments, two pre-trained networks were employed to generated the input dataset distribution

– (i) A StyleGAN2-ADA network trained on the AFHQ-Dogs dataset of resolution  $32 \times 32$ ; and (ii) A StyleGAN2-ADA network trained on the Tiny-ImageNet dataset of resolution  $32 \times 32$ . The outputs are transformed based on the approach described in Appendix C.1. To generate higher-quality samples, we adopted the popular *truncation trick* [30] in sampling from the input-stage generator – The input-stage baseline generator is trained to transform samples drawn from the standard normal distribution to those coming from Tiny-ImageNet, or AFHQ-Dogs datasets. When generating the inputs to the cascaded SpiderGAN stage, samples are drawn from a truncated Gaussian, where a sample is re-drawn if it lies outside the  $[-2, 2]^n$  hypercube (a  $2\sigma$  interval). This was shown to improve the generator output quality at a small cost of marginally reduced sample diversity [30]. On the experiments on learning Ukiyo-E Faces, MetFaces, and FFHQ with cascaded Spider StyleGAN2-ADA, the truncation trick resulted in a 10% improvement in FID on the average. Figure 29 presents the images generated by these models considering baseline sampling and the truncation trick.

The comparison of FID and  $CSID_m$  of the StyleGAN variants trained on FFHQ are provided in Table 10. Spider StyleGAN2-ADA with the Tiny-ImageNet input achieved an FID score on par with StyleGAN-XL, a model with three-fold higher network complexity. However, in terms of  $CSID_m$ , Spider StyleGAN2-ADA achieves state-of-the-art performance, which suggests that the diversity of images generated by Spider StyleGAN2-ADA is superior to that of StyleGAN-XL. The Spider StyleGAN3 model with weight transfer achieves a state-of-the-art FID of 3.07 on AFHQ-Cats, with one-fourth of the training iterations as the baselines. The accelerated convergence can be attributed to the superior *initialization* in the Spider GAN framework, as opposed to initializing with high-dimensional Gaussian inputs. Figures 30- 41 show the images generated by the various models and side-by-side comparison of the images generated by Spider StyleGAN and baseline variants.

### D.5.1 Interpolating with Spider StyleGAN3

In order to better understand the control over representations that the *Spider* framework provides, we consider interpolation experiments on cascaded Spider StyleGAN2-ADA. A pre-trained SpiderStyleGAN2 with Gaussian distributed input and AFHQ-Dogs as output forms the input-stage network. The outputs of this network serve as the input to Spider StyleGAN2-ADA. As discussed in Section 5.2, we consider the following two interpolation schemes:

- Scheme-1, where interpolation is carried out between the AFHQ-Dogs images generated by the input-stage GAN, and subsequently fed to cascaded Spider GAN stage. Figures 42, 44 and 46 present the outputs of the first- and second-stage GANs, when trained on Ukiyo-E Faces, MetFaces and FFHQ images, respectively.
- Scheme-2, where linear interpolation is performed in the Gaussian space. The corresponding samples are used to generate AFHQ-Dogs images, which are fed as input to the Spider GAN stage. Figures 43, 45 and 47 show the intermediate AFHQ-Dogs and Spider GAN outputs for this configuration, when trained on Ukiyo-E Faces, MetFaces and FFHQ images, respectively.

Across all datasets, we observe that Scheme-1 results in superior control over the features, with gradual, fine-grained transitions between the images. On the other hand, images generated by Scheme-2 are affected by the known caveats of Gaussian-space interpolation [15, 16]. Interpolations of Gaussian-distributed points have a very low probability of lying on the Gaussian manifold. Consequently, the generated AFHQ-Dogs images, and the the subsequent target-dataset images possess unnatural discontinuities, appearing unrealistic. In the case of generating FFHQ and Ukiyo-E Faces, this results in the generation of noisy images at intermediate locations.

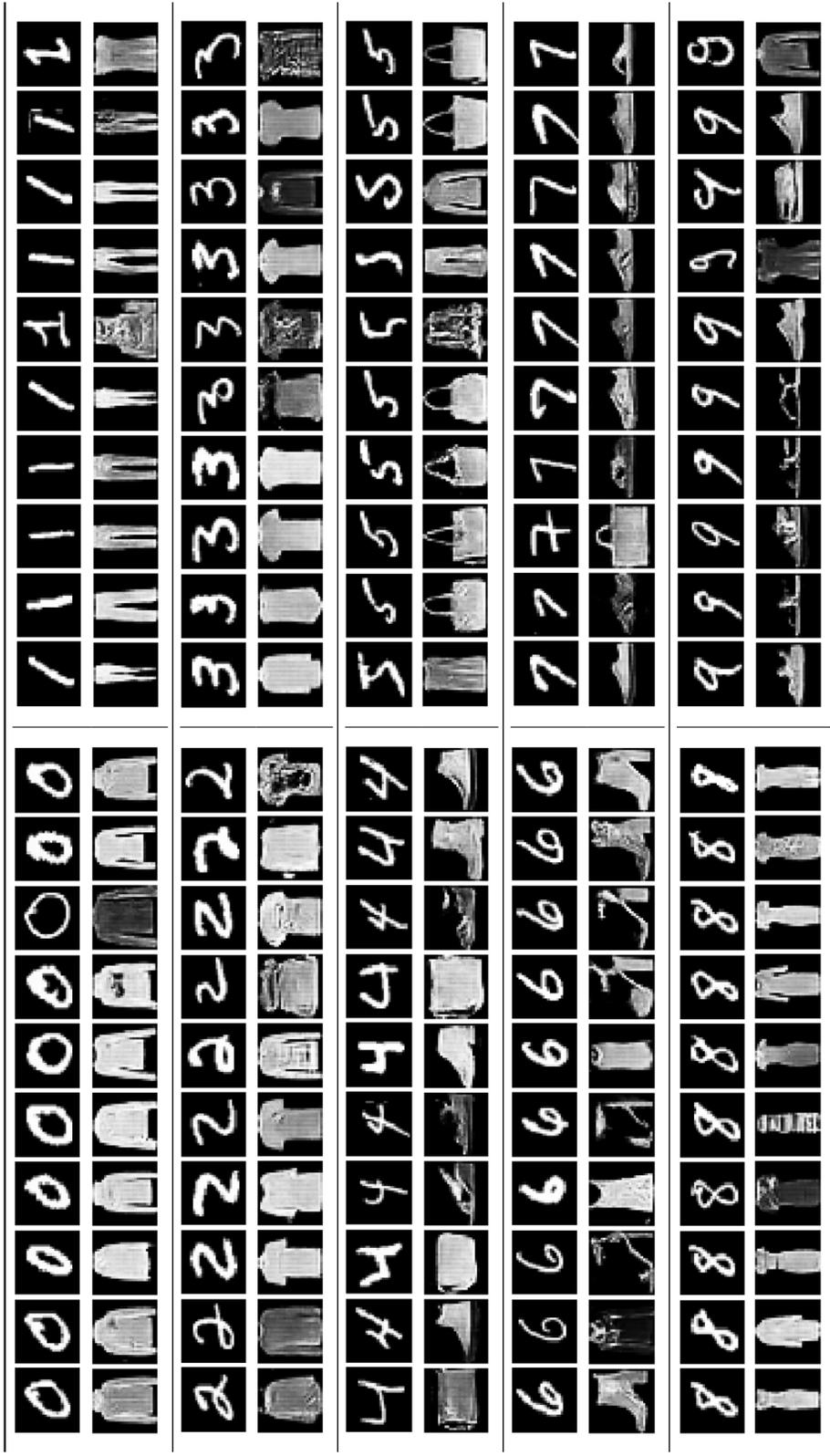


Figure 24. Images representing the class correspondence in a baseline Spider GAN where class embedding are not provided to the networks. While Style GAN learns to map images based on image-level structure, there is overlap between classes.

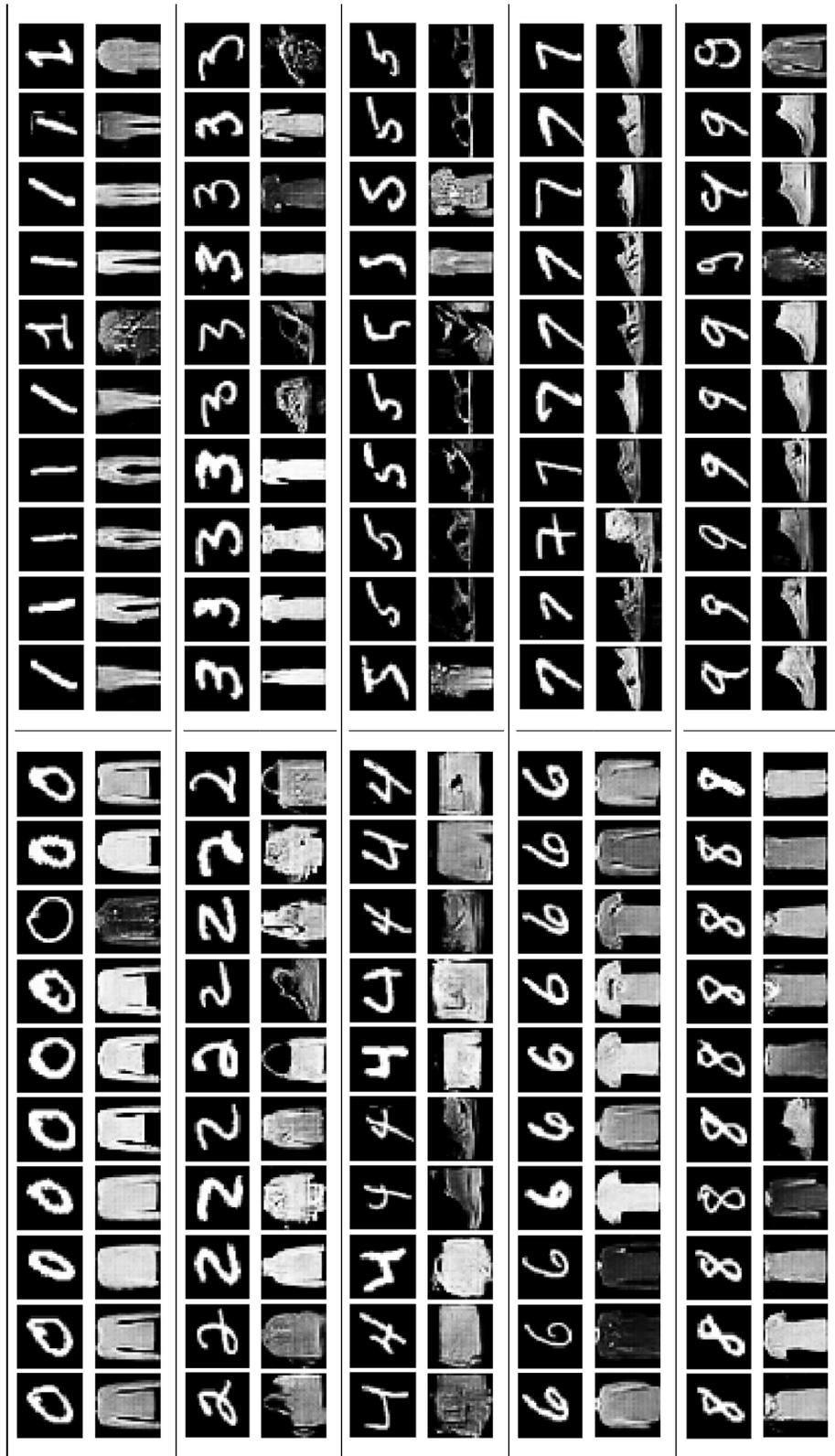


Figure 25. Images representing the class correspondence between the source and target data in a class-conditional Spider GAN where discriminator is modified to output the class label of the generated images. The Spider GAN generator is not provided class information, but is trained to minimize the classification accuracy. While the model's class-correspondence is superior to that of the baseline Spider GAN, class overlap is not eliminated.

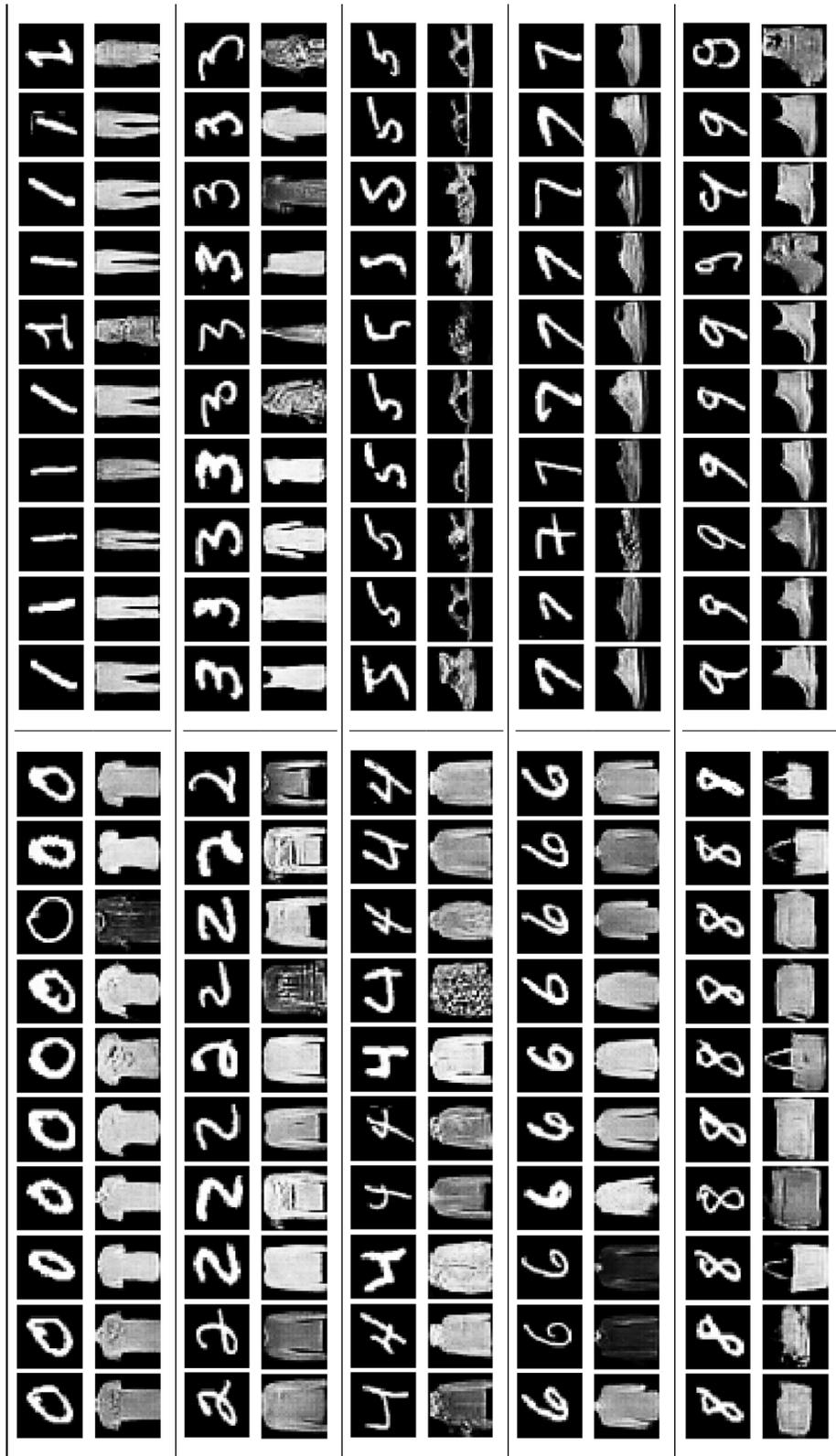


Figure 26. The class correspondence of the images generated by Spider ACGAN. The discriminator is modified to output the class label of the generated images, while the source class information is provided as an embedding layer to the generator. The Spider ACGAN model achieves superior class disentanglement in comparison to the baselines.



Figure 27. Images generated by Spider PGGAN with CIFAR-10 as the target, when trained with Tiny-ImageNet as input, which in turn, is the target for a PGGAN trained with Gaussian noise as the input.

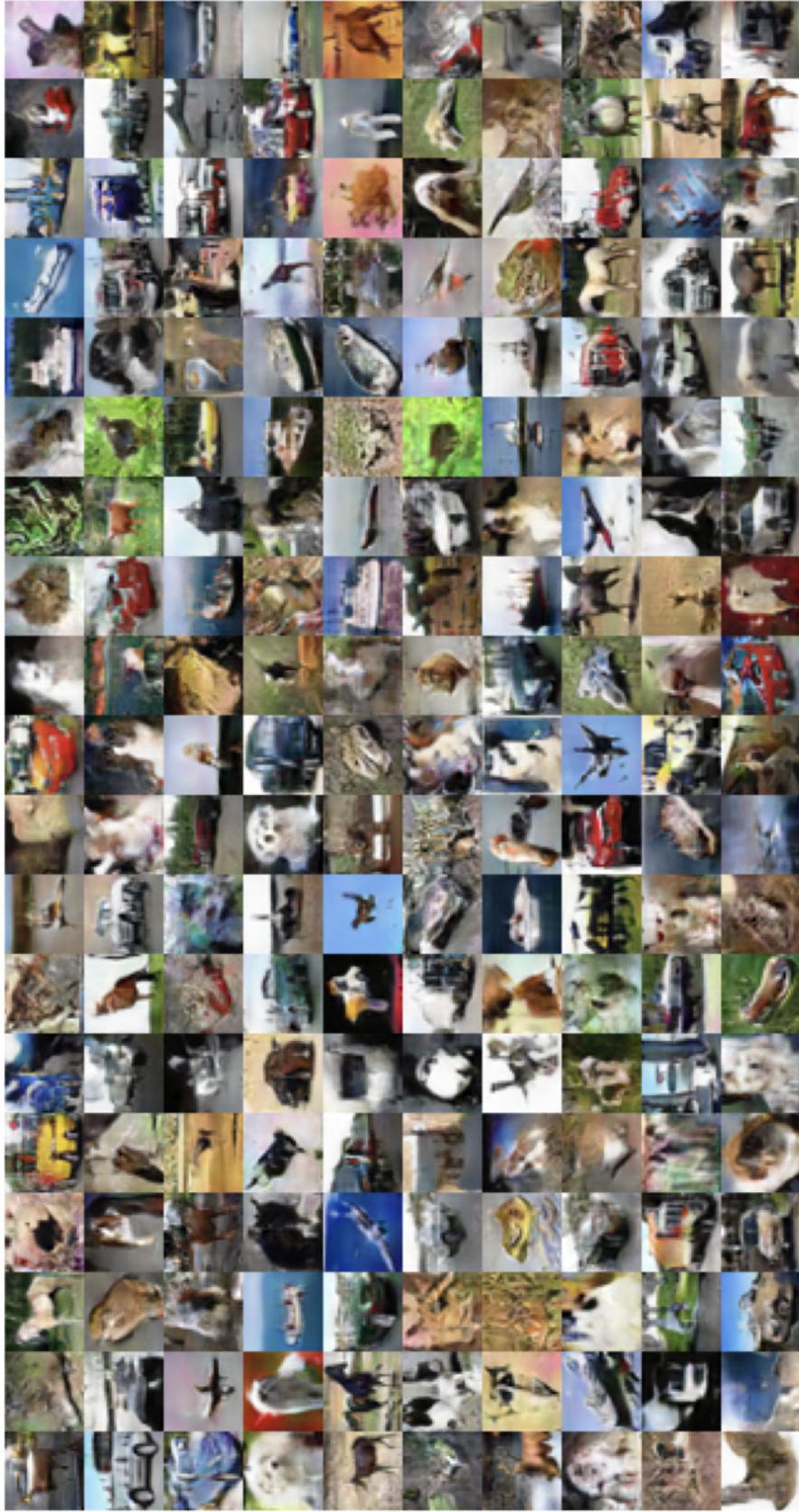


Figure 28. Images generated by Spider PGGAN with CIFAR-10 as the target, when trained with Tiny-ImageNet as the input, which in turn is drawn from a PGGAN trained with a Gaussian input. The weights of the Spider PGGAN are also initialized with the weights of the input PGGAN, resulting in faster training and superior output image quality.

Table 10. A comparison of StyleGAN2-ADA and StyleGAN3 variants, in terms of FID, KID and  $CSID_m$ , on learning FFHQ. A † indicates a reported score. Spider StyleGAN2-ADA performs on par with the state-of-the-art StyleGAN-XL (three fold higher network complexity) [36] in terms of FID and KID. However, Spider StyleGAN2-ADA variants achieved the best (lowest)  $CSID_m$  scores, which suggests that the *Spider* variants learnt more diverse representations of the target dataset when compared against the baselines.

Architecture	Input	Clean-FID [10]	Clean-KID [10]	$CSID_m$
StyleGAN2-ADA [11]	Gaussian	2.70†	$0.906 \times 10^{-3}$	2.65
StyleGAN3-T [12]	Gaussian	2.79†	$1.031 \times 10^{-3}$	2.95
StyleGAN-XL [36]	Gaussian	<b>2.02†</b>	<b><math>0.287 \times 10^{-3}</math></b>	3.94
Spider StyleGAN2-ADA ( <b>Ours</b> )	TinyImageNet	<u>2.45</u>	$0.915 \times 10^{-3}$	<b>1.99</b>
Spider StyleGAN2-ADA ( <b>Ours</b> )	AFHQ-Dogs	3.07	<u><math>0.795 \times 10^{-3}</math></u>	<u>2.55</u>
Spider StyleGAN3-T ( <b>Ours</b> )	TinyImageNet	2.86	$1.162 \times 10^{-3}$	3.25

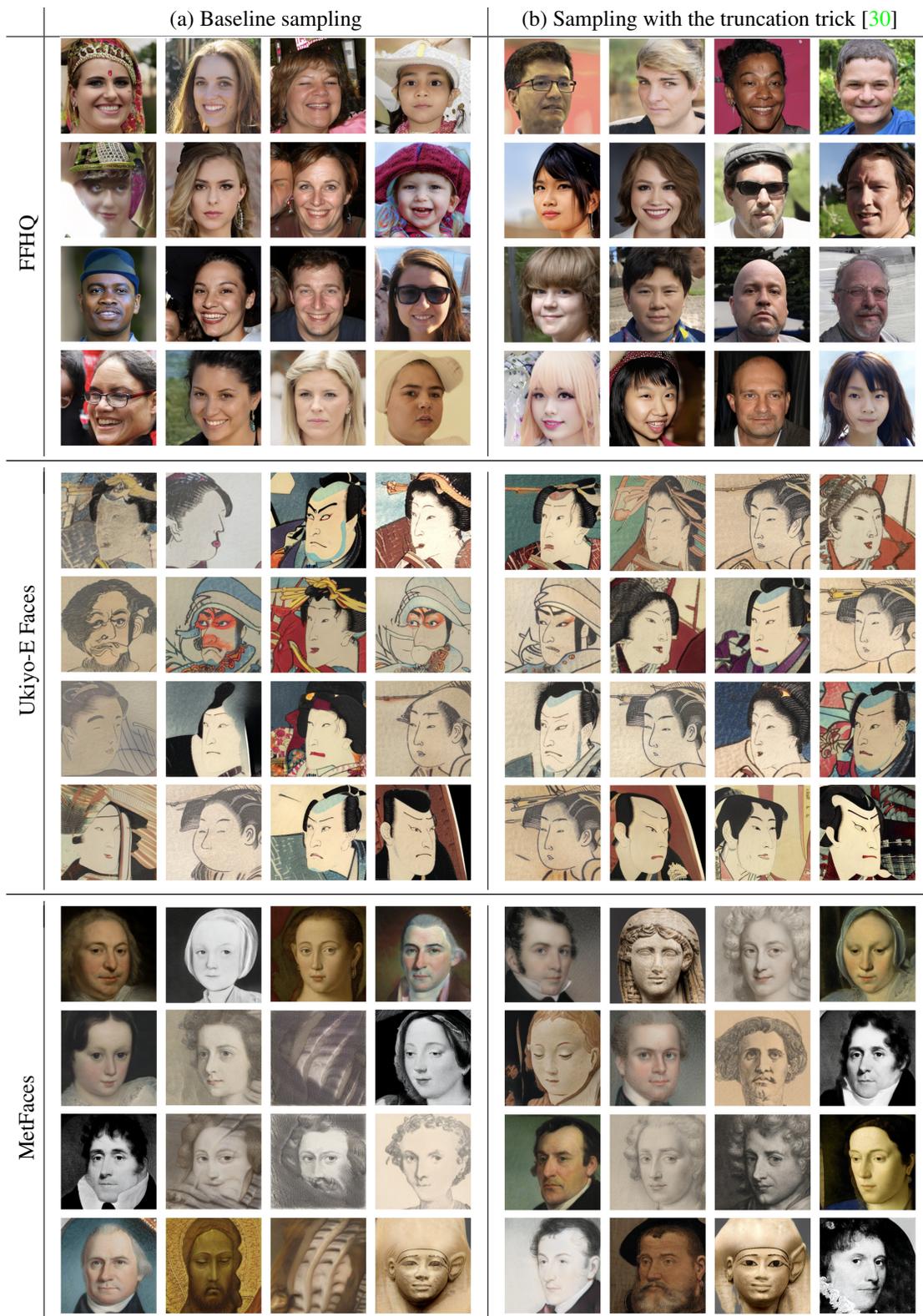


Figure 29. Images generated by cascaded Spider GAN variants when the Gaussian samples provided to the input-stage are (a) retained as-is; and (b) resampled when lying outside of the  $2\sigma$  interval  $[-2, 2]$  (*the truncation trick* [30]). Images generated using truncated input samples are of a superior visual quality. Baseline sampling results in distorted faces in the case of FFHQ and Ukiyo-E faces datasets, while on MetFaces, poor quality samples resulted in *alien* patterns.



Figure 30. Ukiyo-E images generated by the *Spider* variant of StyleGAN2, trained on AFHQ-Dogs input. Since the AGFQ-Dogs dataset has relatively lower diversity than the target, the generated Ukiyo-E samples are visually sup-par compared to the performance of the baseline StyleGAN2-ADA.



Figure 31. Ukiyo-E face images generated by the *Spider* variant of StyleGAN2, trained on Tiny-ImageNet input. The *Spider* variant achieves state-of-the-art FID of 20.44, compared to 26.74 of the baseline StyleGAN2-ADA (lower FID is better).



Figure 32. A comparison of MetFaces images generated by the baseline and Spider GAN variants. Spider StyleGAN2 with the Tiny-ImageNet (TIN) input data outperforms all other variants, generating sharper and more diverse images, achieving a state-of-the-art FID of 15.60 as opposed to an FID of 18.75 achieved by StyleGAN2-ADA.

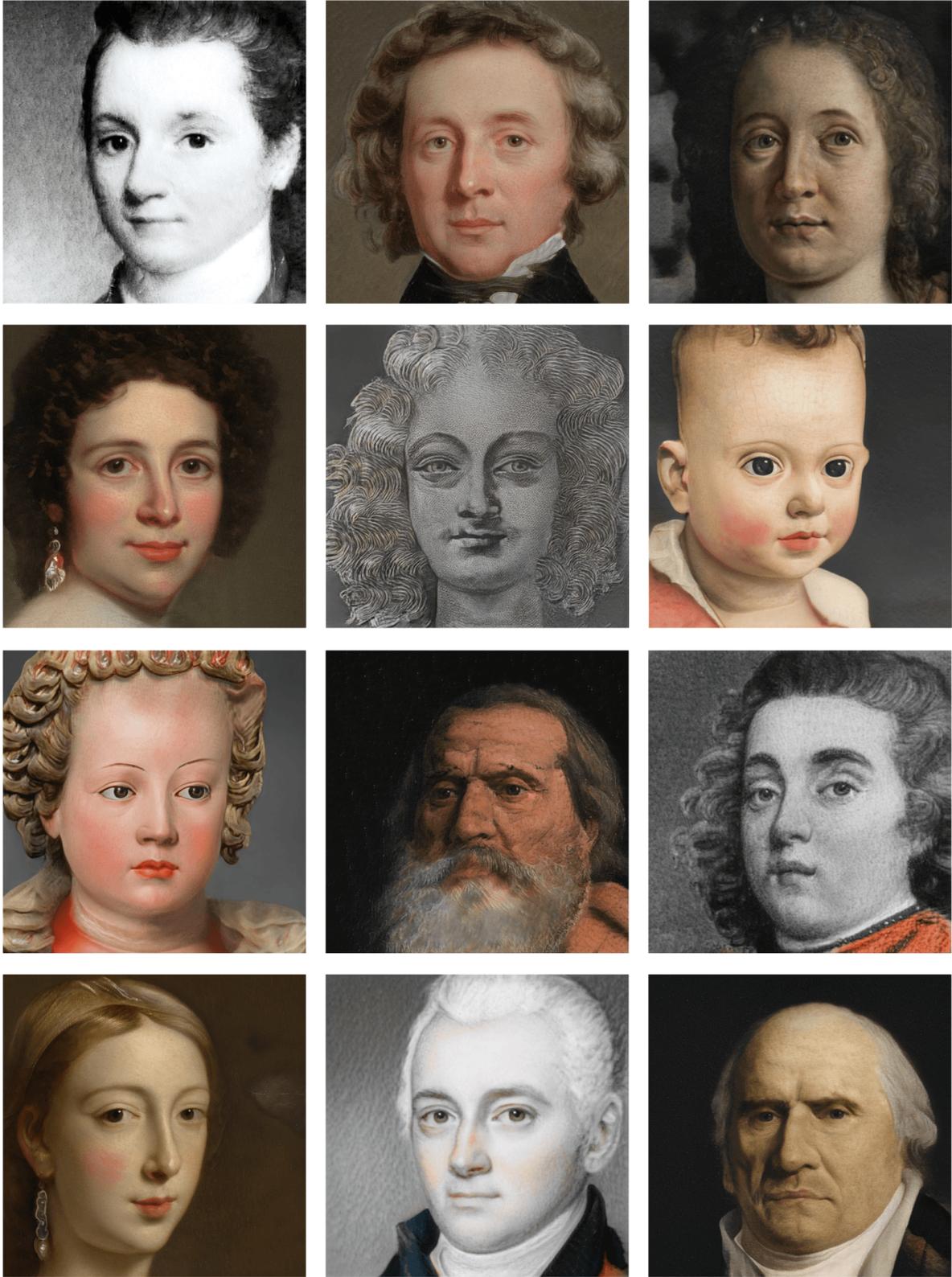


Figure 33. Representative MetFaces images generated by the *Spider* variant of StyleGAN2, trained on AFHQ-Dogs input. The model achieved an FID score of 29.82, which is lower than the FID of the StyleGAN2-ADA baseline (18.75). This is expected, as the AFHQ-Dogs is not a *friendly neighbor* of the target dataset.

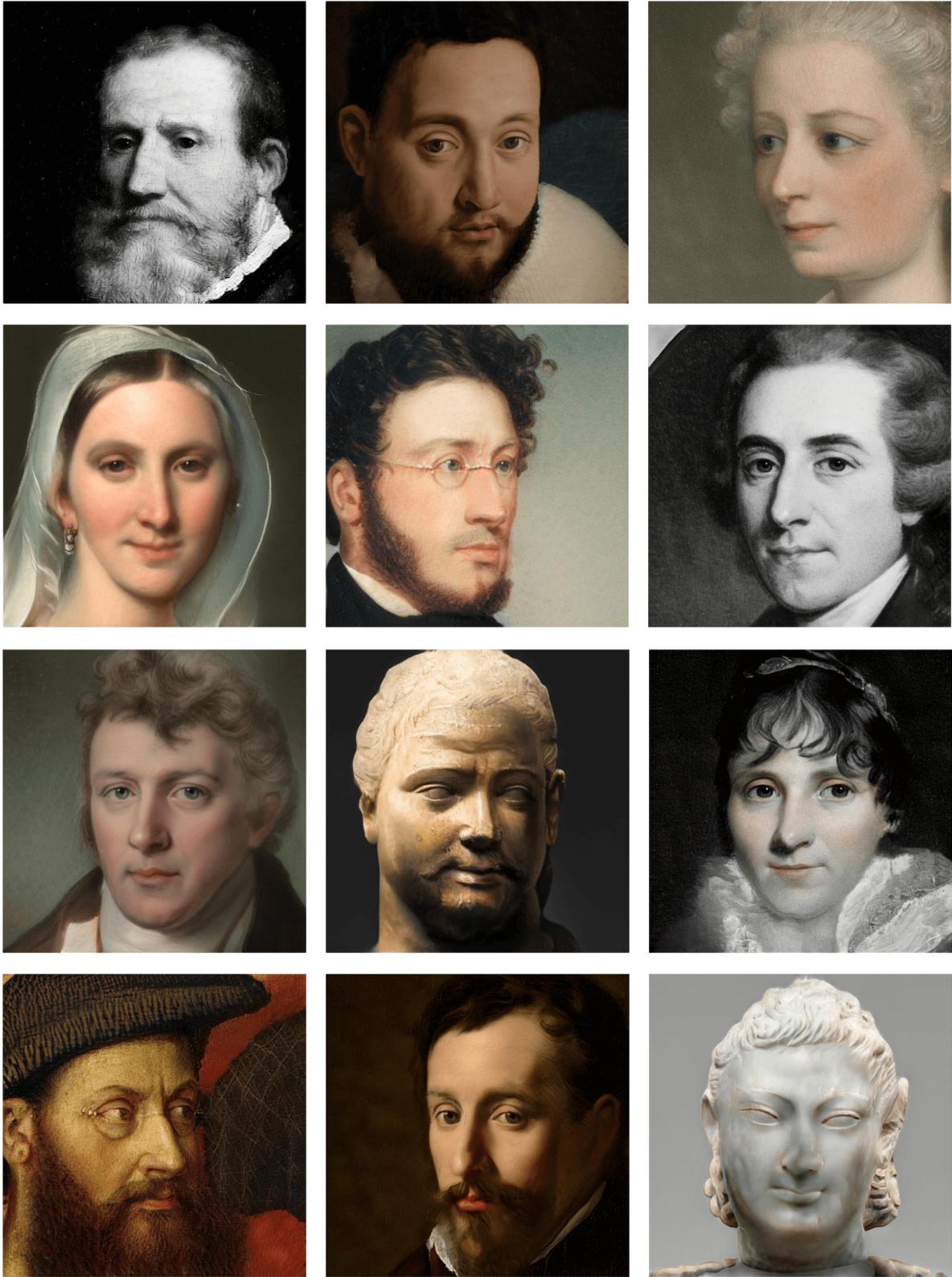


Figure 34. Sample images generated by the *Spider* variant of StyleGAN2, trained on Tiny-ImageNet input and MetFaces as output. The Spider StyleGAN variant achieves state-of-the-art FID of 15.60, against an FID of 18.75 achieved by the StyleGAN2-ADA baseline.



Figure 35. A comparison of FFHQ ages generated by the baseline and Spider GAN variants trained with AFHQ-Dogs and Tiny-ImageNet (TIN) inputs. Spider StyleGAN2-ADA with the Tiny-ImageNet input performs on par with the StyleGAN-XL baseline (FID of 2.45 for the proposed approach versus FID of 2.07 for the baseline), with a mere one-third of the network complexity.

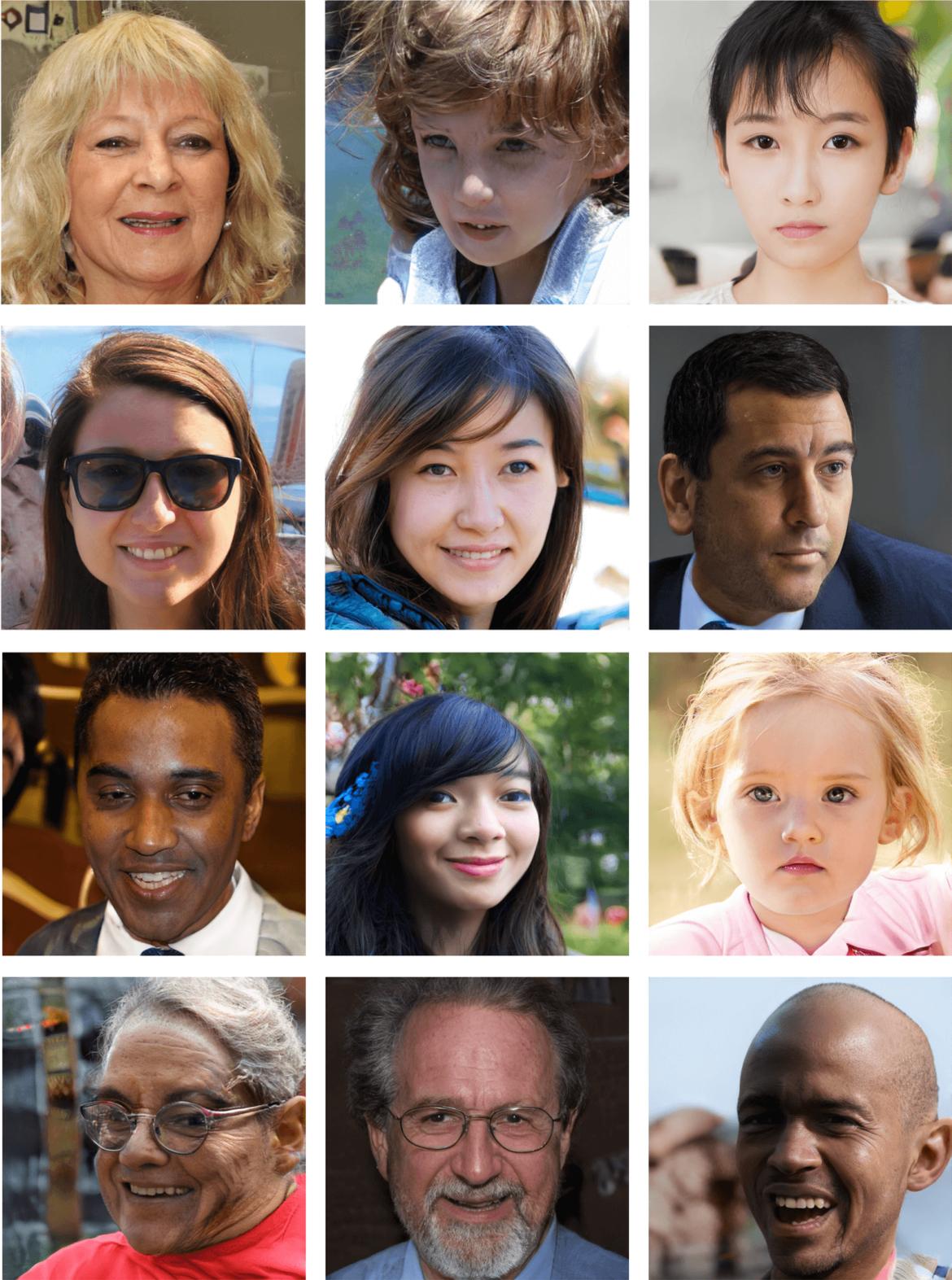


Figure 36. FFHQ images generated by the *Spider* variant of StyleGAN2-ADA, trained on a model incorporating weight transfer from the StyleGAN2-ADA model trained on AFHQ-Dogs. The input samples are drawn from a StyleGAN2-ADA model pre-trained on  $32 \times 32$  Tiny-ImageNet images. The converged model achieved an FID of 3.07 as opposed to 2.70 of the baseline model. The lower FID can be attributed to the choice of a *poor neighbor* of the target dataset.

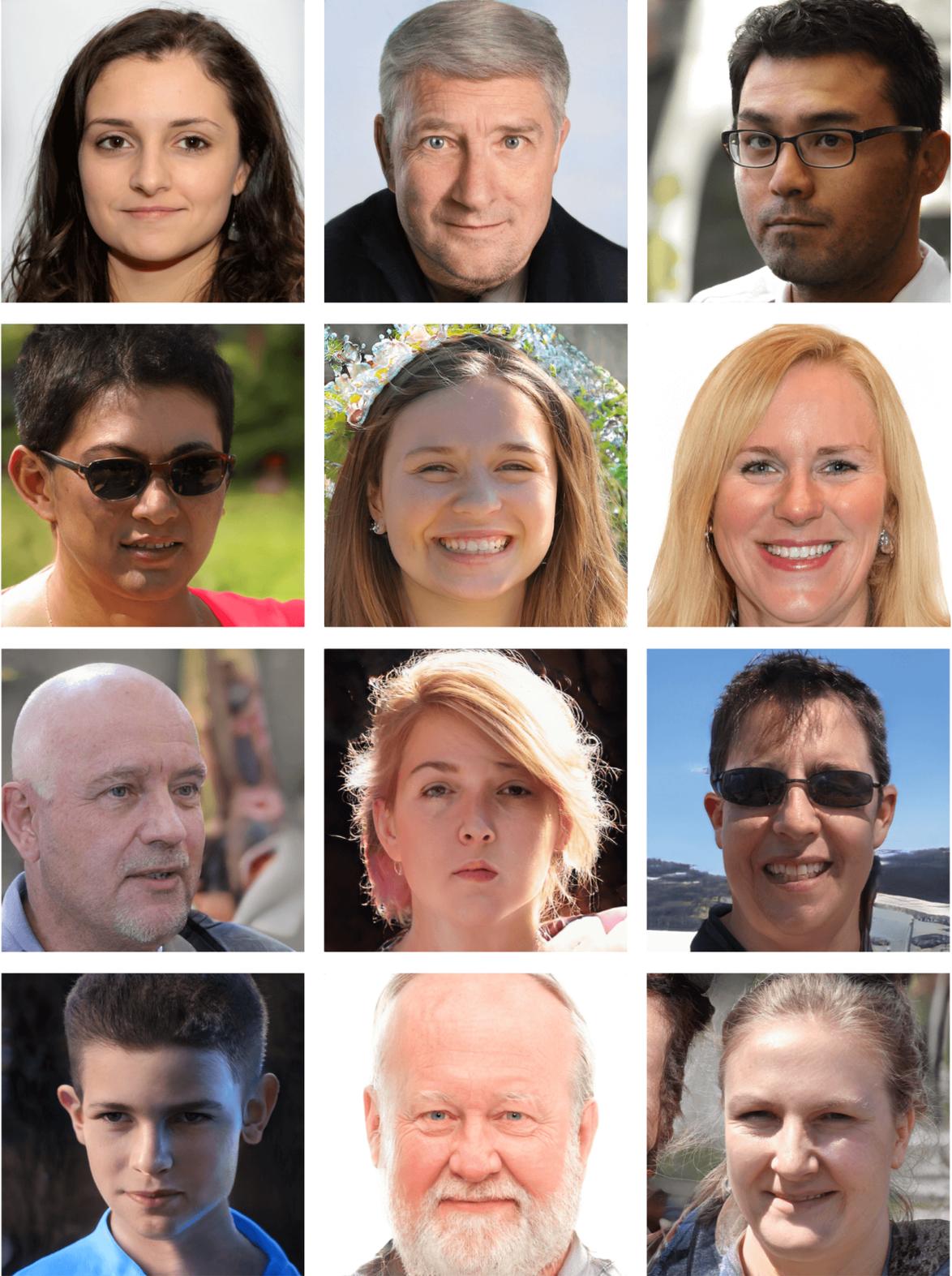


Figure 37. FFHQ images generated by the *Spider* variant of StyleGAN2-ADA, trained on a model incorporating weight transfer from the StyleGAN2-ADA model trained on AFHQ-Dogs images. The model achieves an FID of 2.45, superior to the baseline StyleGAN2-ADA, Polarity-StyleGAN2 and MaGNET-StyleGAN2 (with FID scores of 2.70, 2.57 and 2.66, respectively).

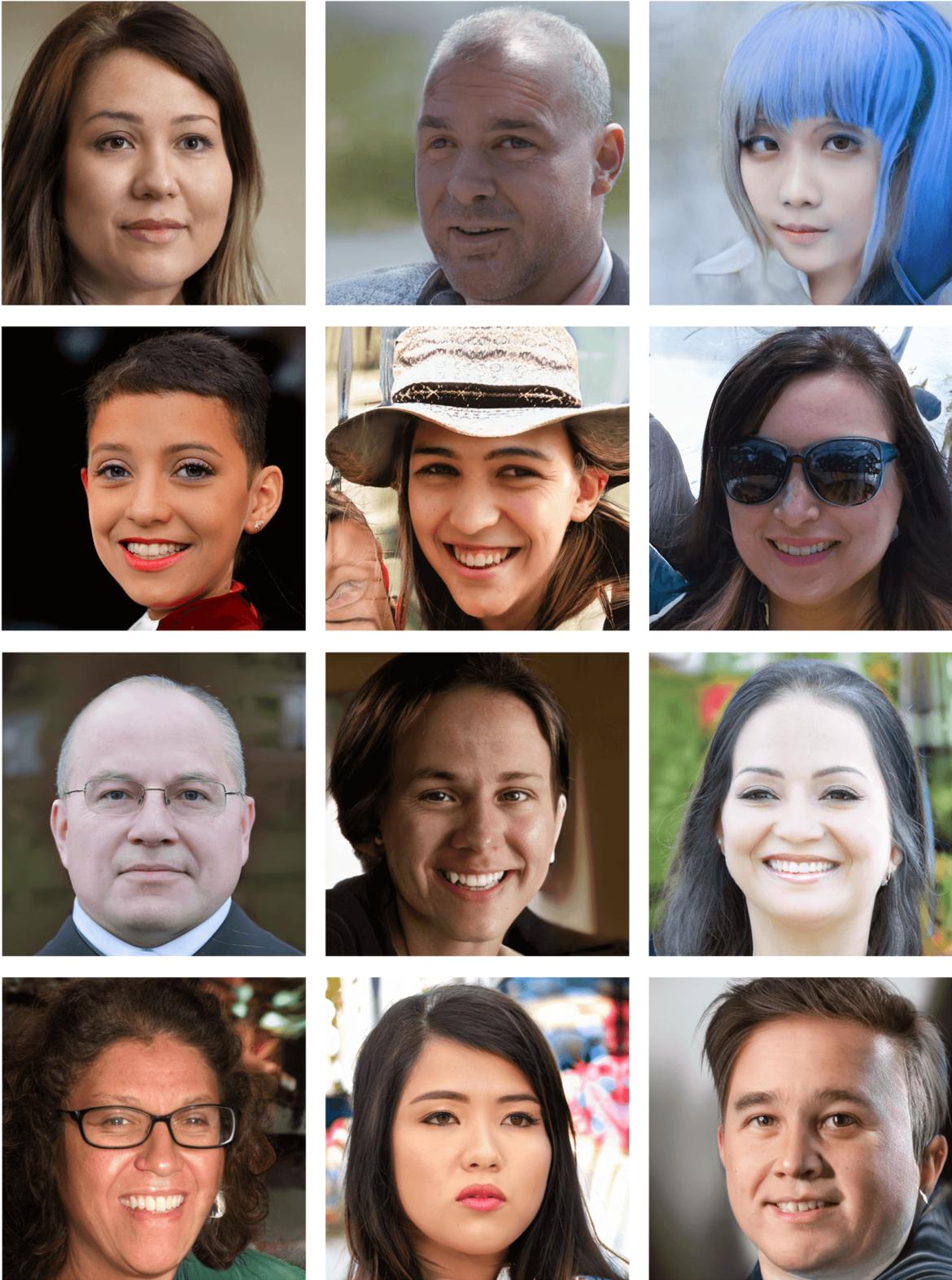


Figure 38. FFHQ images generated by the *Spider* variant of StyleGAN3-T, trained on Tiny-ImageNet dataset. The model achieved an FID of 2.86.



Figure 39. AFHQ-Cat images generated by the *Spider* variant of StyleGAN2-ADA, trained on a model incorporating weight transfer from the StyleGAN2-ADA model trained on AFHQ-Dogs. The input samples are drawn from a StyleGAN2-ADA model pre-trained on  $32 \times 32$  Tiny-ImageNet images. The converged model achieves an FID of 3.86 in one-fifth of the training iterations required by the baseline.



Figure 40. AFHQ-Cat images generated by the *Spider* variant of StyleGAN3-T from scratch. The input samples are drawn from a StyleGAN3-T model model pre-trained on  $32 \times 32$  AFHQ-Dog images. The converged model achieves an FID of 6.29, which is on par with the baselines, in a mere one-fifth of the suggested [12] training iterations.



Figure 41. AFHQ-Cat images generated by the *Spider* variant of StyleGAN3-T, trained on a model incorporating weight transfer from the StyleGAN3-T model trained on AFHQv2-Dog. The input samples are drawn from a StyleGAN2-ADA model model pre-trained on  $32 \times 32$  Tiny-ImageNet images. The converged model achieves a state-of-the-art FID of 3.07 and KID of  $0.23 \times 10^{-3}$  in one-fourth of the training iterations of baseline StyleGAN3 [12].



Figure 42. A grid of interpolated Ukiyo-E images generated by Spider StyleGAN2-ADA, trained on AFHQ-Dogs. Images are generated by transforming Gaussian noise to AFHQ-Dogs images via an input-stage model, whose subsequent outputs serve as the input to Spider StyleGAN2-ADA. The interpolation is performed in the AFHQ-Dogs space, and provided as input to Spider StyleGAN2-ADA. We observe smooth transitions between the interpolated images, which allows for fine-grained control of the features.



Figure 43. A grid of interpolated Ukiyo-E images generated by the Spider StyleGAN2-ADA, trained on AFHQ-Dogs. Images are generated by transforming Gaussian noise to AFHQ-Dogs images via an input-stage model, whose subsequent outputs serve as the input Spider StyleGAN2-ADA. In this case, the interpolation is performed in the Gaussian space and fed to the input-stage pre-trained StyleGAN. The corresponding AFHQ-Dogs images generated are provided as input to the Spider StyleGAN2-ADA. We observe abrupt and unnatural transitions between images. Some images also appear to be unrealistic, which is not surprising, as the interpolation of points drawn from a Gaussian manifold have an extremely low probability of lying on the manifold.

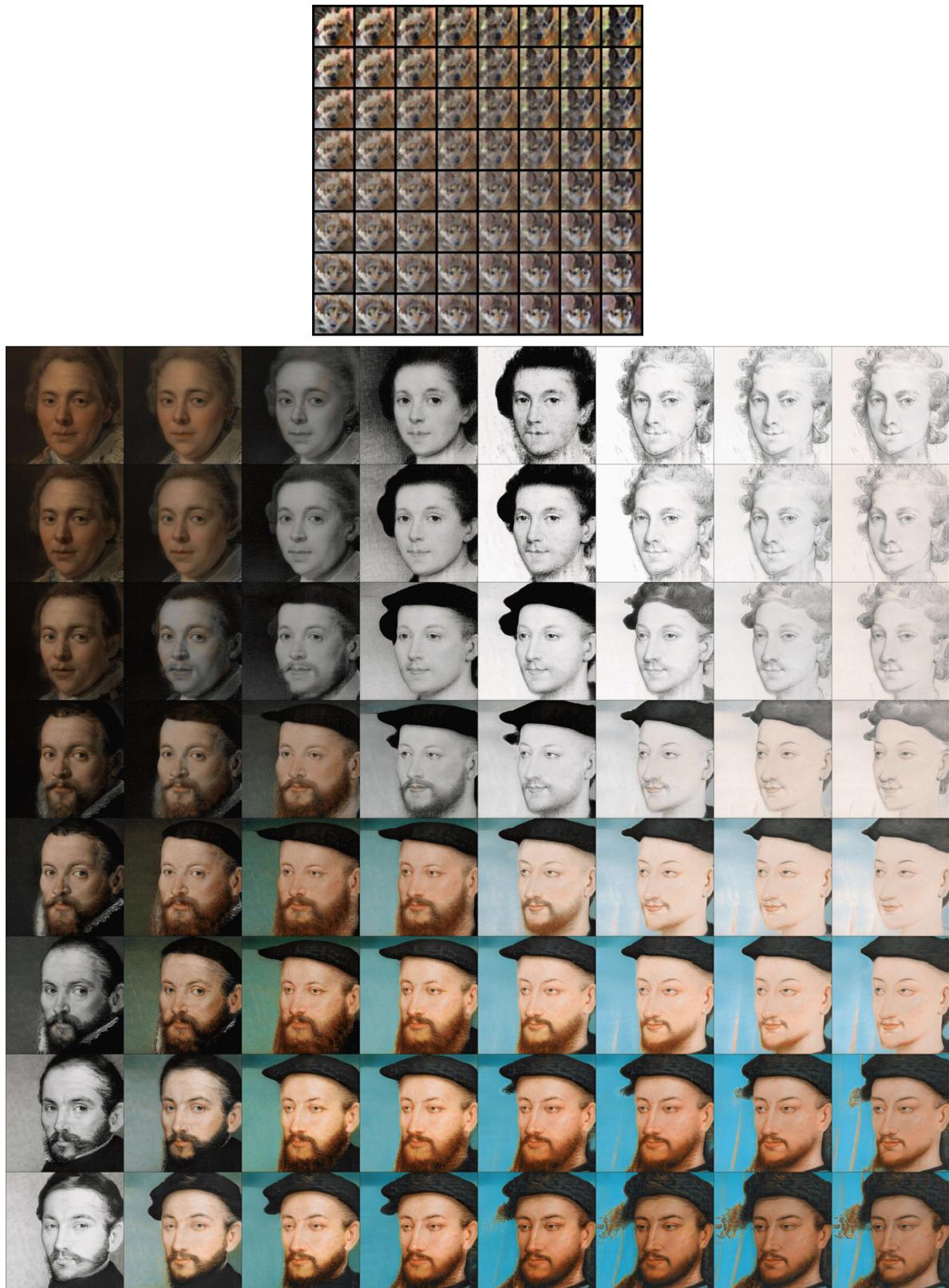


Figure 44. Interpolations on the MetFaces images generated by the Spider StyleGAN2-ADA, trained on AFHQ-Dogs. The inputs to the Spider StyleGAN are linearly interpolated AFHQ-Dogs images. We observe smooth and gradual transitions between the color- and sketch-based images generated by Spider StyleGAN, which is highly desirable for feature manipulation.

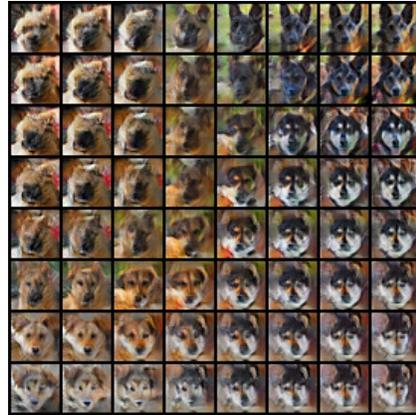


Figure 45. Interpolated MetFaces images generated by the Spider StyleGAN2-ADA, trained on AFHQ-Dogs. The interpolation is carried out in the Gaussian fed to the pre-trained input-stage StyleGAN. The corresponding AFHQ-Dogs images generated are given as input to Spider StyleGAN2-ADA. We observe unnatural and discontinuous transitions between the color and sketch images which can be attributed to the disconnected manifold structure of the dataset.

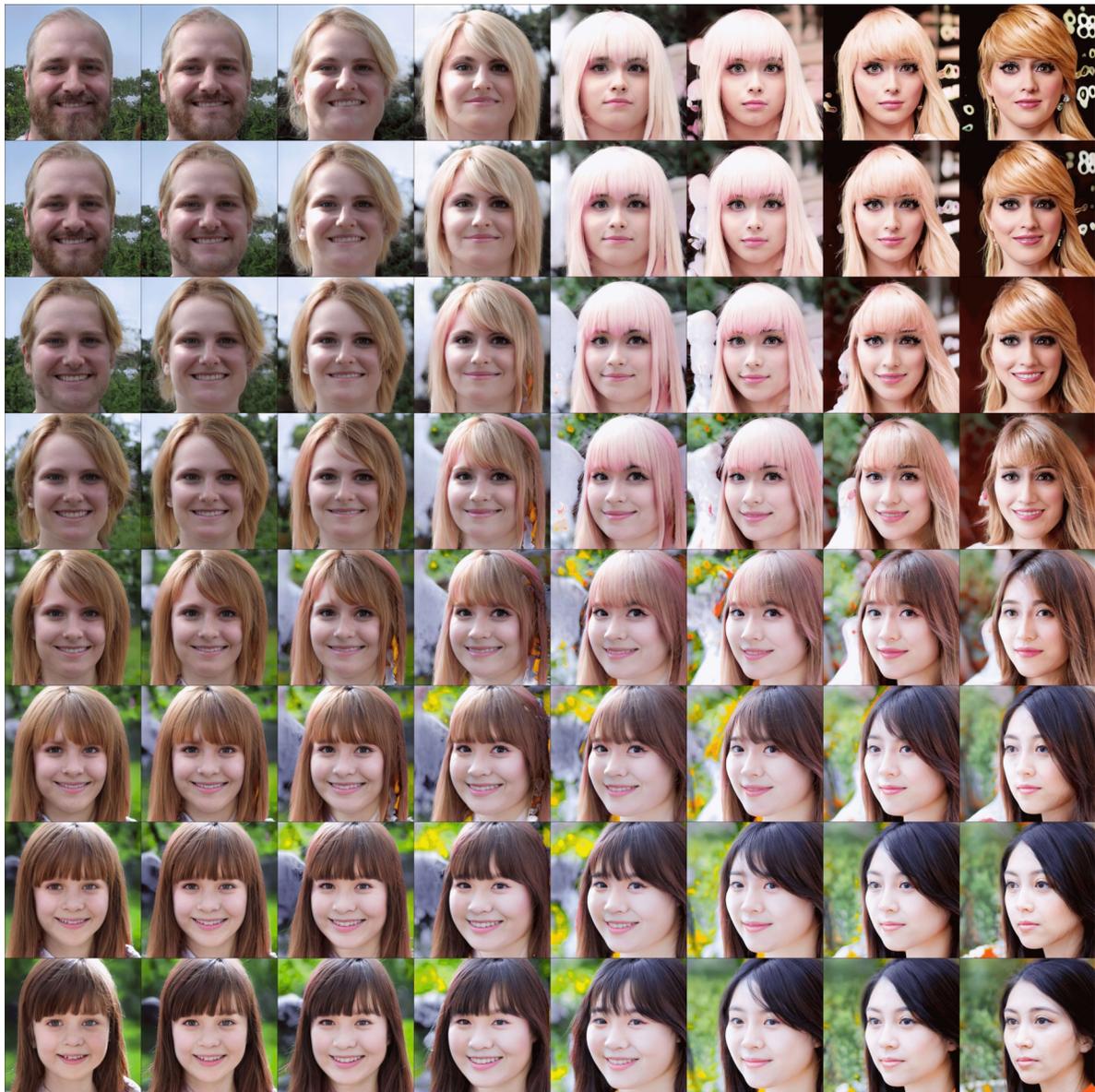


Figure 46. Interpolations on the FFHQ images generated by the Spider StyleGAN2-ADA, trained on AFHQ-Dogs. The inputs to the Spider StyleGAN are linear interpolates computed on the AFHQ-Dogs images. We observe that the proposed Spider variant generates smooth and gradual transitions with fine-grained facial features allowing for superior control of the image generation.

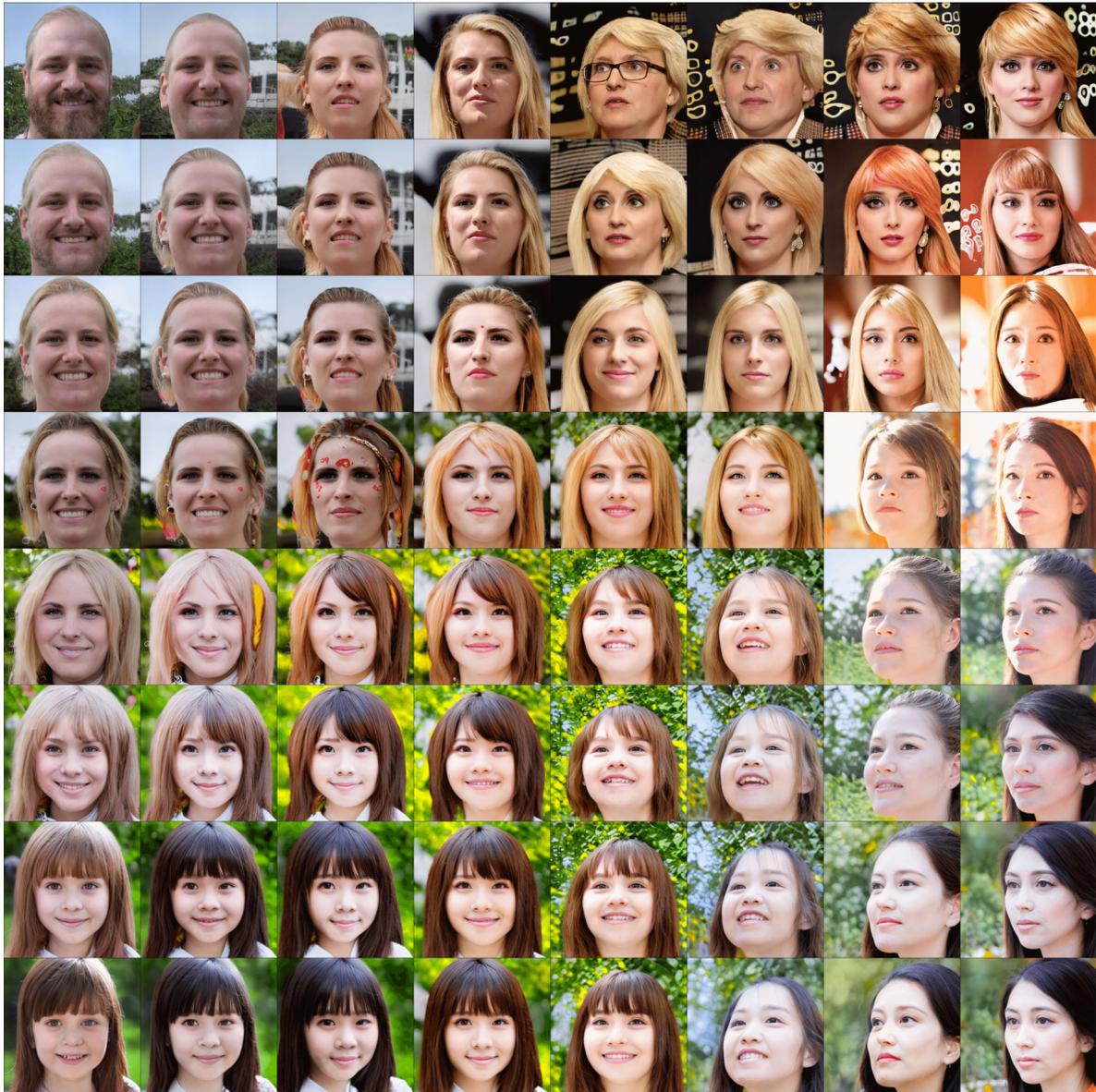


Figure 47. Interpolated FFHQ images generated by the Spider StyleGAN2-ADA, trained on AFHQ-Dogs. Interpolation is performed in the Gaussian space of the input-stage StyleGAN, which generate a set of AFHQ-Dogs images, which in turn serve as the input to the Spider StyleGAN2-ADA. We observe discontinuous transitions in the hair, color, and other features of the generated images. Some images are also noisy, as they correspond to inputs drawn from outside of the training manifold.

## E. GitHub Repository and Code Release

The codebase for implementing Spider GAN, Spider PGGAN, Spider StyleGAN and SID has been included as part of the Supplementary. The baseline non-parametric prior [16] was implemented using the publicly released `.mat` file. PGGAN [27] and StyleGAN2 and StyleGAN3 were implemented using publicly available GitHub repositories, with modification included to implement their respective *Spider* variants.

The implementation for CSID<sub>m</sub> and SID are based on the Inception features provided by the Clean-FID [10] library. In order to maintain uniformity, SID can also be computed by providing the path to existing source and target image folders, akin to FID and KID.

An implementation of SID atop the Clean-FID [10] backbone, with associated animations of the experiments presented in this manuscript are available at <https://github.com/DarthSid95/clean-sid>. The TensorFlow-based source code for Spider GANs built atop the DCGAN architecture, and associated pre-trained models are available at <https://github.com/DarthSid95/SpiderDCGAN>. The PyTorch-based source code for implementing the *Spider* variants of PGGAN, StyleGAN2, StyleGAN2-ADA and StyleGAN3, with the corresponding pre-trained models are available at <https://github.com/DarthSid95/SpiderStyleGAN>. Images in the *Main Manuscript* and *Supplementary Document* have been compressed to meet the file-size limits of the venue. The GitHub repositories also include the full-resolution versions of the images provided in the *Main Manuscript* and *Supporting Document*.

## References

- [1] F. Camastra and A. Staiano, “Intrinsic dimension estimation: Advances and open problems,” *Information Sciences*, vol. 328, pp. 26–41, 2016.
- [2] M. Hein and J.-Y. Audibert, “Intrinsic dimensionality estimation of submanifolds in  $R^d$ ,” in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 289–296, 2005.
- [3] E. Facco, M. d’Errico, A. Rodriguez, and A. Laio, “Estimating the intrinsic dimension of datasets by a minimal neighborhood information,” *Scientific Reports*, vol. 7, Sep. 2017.
- [4] P. Campadelli, E. Casiraghi, C. Ceruti, and A. Rozza, “Intrinsic dimension estimation: Relevant techniques and a benchmark framework,” *Mathematical Problems in Engineering*, vol. 2015, pp. 1–21, 10 2015.
- [5] C. Davis and W. M. Kahan, “The rotation of eigenvectors by a perturbation. iii,” *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970.
- [6] Y. Yu, T. Wang, and R. J. Samworth, “A useful variant of the Davis—Kahan theorem for statisticians,” *Biometrika*, vol. 102, no. 2, pp. 315–323, 2015.
- [7] T. Liang, “How well generative adversarial networks learn distributions,” *Journal of Machine Learning Research*, vol. 22, no. 228, pp. 1–41, 2021.
- [8] N. Schreuder, V.-E. Brunel, and A. Dalalyan, “Statistical guarantees for generative models without domination,” in *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, vol. 132, pp. 1051–1071, Mar 2021.
- [9] A. Block, Z. Jia, Y. Polyanskiy, and A. Rakhlin, “Intrinsic dimension estimation,” 2021.
- [10] G. Parmar, R. Zhang, and J.-Y. Zhu, “On buggy resizing libraries and surprising subtleties in FID calculation,” *arXiv preprint, arXiv:2104.11222*, vol. abs/2104.11222, April 2021.
- [11] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training generative adversarial networks with limited data,” in *Advances in Neural Information Processing Systems 33*, 2020.
- [12] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, “Alias-free generative adversarial networks,” in *Advances in Neural Information Processing Systems*, June 2021.
- [13] M. Khayatkhoei, M. K. Singh, and A. Elgammal, “Disconnected manifold learning for generative adversarial networks,” in *Advances in Neural Information Processing Systems 31*, pp. 7343–7353, 2018.
- [14] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *Proceedings of the 4th International Conference on Learning Representations*, 2016.
- [15] Y. Kilcher, A. Lucchi, and T. Hofmann, “Semantic interpolation in implicit models,” in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [16] R. Singh, P. Turaga, S. Jayasuriya, R. Garg, and M. Braun, “Non-parametric priors for generative adversarial networks,” in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 5838–5847, June 2019.

- [17] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *Proceedings of the 34th International Conference on Machine Learning*, pp. 214–223, 2017.
- [18] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for GANs do actually converge?,” in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 3481–3490, 2018.
- [19] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [20] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint, arXiv:1603.04467*, Mar. 2016.
- [21] A. Paszke *et al.*, “PyTorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, vol. 32, 2019.
- [22] 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,” *arXiv preprints, arXiv:1706.08500*, 2018.
- [23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *arXiv preprint, arXiv:1512.00567*, Dec. 2015.
- [24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.
- [25] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, “Demystifying MMD GANs,” in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [26] I. O. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, “Wasserstein auto-encoders,” in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [27] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” in *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [28] P. Esser, R. Rombach, and B. Ommer, “Taming transformers for high-resolution image synthesis,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.
- [29] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu, “Vector-quantized image modeling with improved VQGAN,” in *Proceedings of the 10th International Conference on Learning Representations*, 2022.
- [30] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” *arXiv preprints, arXiv:1809.11096*, Sep. 2018.
- [31] M. Kang, W. Shim, M. Cho, and J. Park, “Rebooting ACGAN: Auxiliary classifier GANs with stable training,” in *Advances in Neural Information Processing Systems 34*, 2021.
- [32] P. Zhong, Y. Mo, C. Xiao, P. Chen, and C. Zheng, “Rethinking generative mode coverage: A pointwise guaranteed approach,” in *Advances in Neural Information Processing Systems*, 2019.
- [33] I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf, “AdaGAN: boosting generative models,” in *Advances in Neural Information Processing Systems*, 2017.
- [34] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *Advances in Neural Information Processing Systems 29*, pp. 2180–2188, 2016.
- [35] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier GANs,” in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
- [36] A. Sauer, K. Schwarz, and A. Geiger, “StyleGAN-XL: Scaling StyleGAN to large diverse datasets,” *arXiv.org*, vol. abs/2201.00273, 2022.