Max-Affine Spline Insights into Deep Generative Networks

Randall Balestriero [1] Sebastien Paris [2] Richard G. Baraniuk [1]

Abstract

We connect a large class of Generative Deep Networks (GDNs) with spline operators in order to derive their properties, limitations, and new opportunities. By characterizing the latent space partition, dimension and angularity of the generated manifold, we relate the manifold dimension and approximation error to the sample size. The manifold-per-region affine subspace defines a local coordinate basis; we provide necessary and sufficient conditions relating those basis vectors with disentanglement. We also derive the output probability density mapped onto the generated manifold in terms of the latent space density, which enables the computation of key statistics such as its Shannon entropy. This finding also enables the computation of the GDN likelihood, which provides a new mechanism for model comparison as well as providing a quality measure for (generated) samples under the learned distribution. We demonstrate how low entropy and/or multimodal distributions are not naturally modeled by DGNs and are a cause of training instabilities.

1. Introduction

Deep Generative Networks (DGNs), which map a low-dimensional latent variable $\boldsymbol{z}$ to a higher-dimensional generated sample $\boldsymbol{x}$, have made enormous leaps in capabilities in recent years. Popular DGNs include Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and their variants (Dziugaite et al., 2015; Zhao et al., 2016; Durugkar et al., 2016; Arjovsky et al., 2017; Mao et al., 2017; Yang et al., 2019); Variational Autoencoders (Kingma & Welling, 2013) and their variants (Fabius & van Amersfoort, 2014; van den Oord et al., 2017; Higgins et al., 2017; Tomczak & Welling, 2017; Davidson et al., 2018); and flow based models such as NICE (Dinh et al., 2014), Normalizing Flow (Rezende & Mohamed, 2015), and their variants (Dinh et al., 2016; Grathwohl et al., 2018; Kingma & Dhariwal, 2018).

While DGNs are easy to describe and analyze locally in terms of simple affine operators and scalar nonlinearities, a general framework for their *global structure* has remained elusive. In this paper, we take a step in the direction of a better theoretical understanding of DGNs constructed using continuous, piecewise affine nonlinearities by leveraging recent progress on *max-affine spline operators* (MASOs) (Balestriero & Baraniuk, 2018b). Our main contributions are as follows;

[C1] We characterize the *piecewise-affine manifold structure* of the generated samples, including its *intrinsic dimension* (Section 3), which sheds new light on the impact of techniques like dropout (Section 3.3) and provides practionioners with sensible design principles for constructing a desired manifold.

[C2] We characterize the *local coordinates of the generated manifold* and the inverse mapping from data points $\boldsymbol{x}$ back to latent variables $\boldsymbol{z}$ (Section 4.1), which provides new necessary and sufficient conditions for *disentanglement*, *interpretability* and new links between DGNs and *adaptive basis methods* (Section 4.2). By characterizing the angles between adjacent local affine regions, we demonstrate how weight sharing in a DGN heavily constrains the *curvature* of the generated manifold despite the fact that the DGN might be tremendously overparameterized (Section 4.3).

[C3] We provide a DGN input-output formula that enables us to derive the analytical *probability density on the generated manifold* that is induced by the latent space (Section 5.1). We use this result to derive *Normalizing Flows* (NMFs) form first principles and highlight how the DGN design, $\boldsymbol{x} \mapsto \boldsymbol{z}$ (most NMFs) versus $\boldsymbol{z} \mapsto \boldsymbol{x}$ (DGNs) allow for either fast likelihood computation and slow sampling or vice-versa (Section 5.2). Finally, the Shannon entropy of the output density provide a new lens through which to study the difficulty of generating multidimensional, low-entropy distributions using DGNs (Section 5.3).

Reproducible code for the various experiments and figures is be provided on Github[1].

[1]ECE Department, Rice University, TX, USA [2]LSIS, UTLN, La Garde, France. Correspondence to: Randall Balestriero <randallbalestriero@gmail.com>.

[1]https://github.com/RandallBalestriero/GAN.git

2. Background

Deep Networks. A deep (neural) network (DN) is an operator f_Θ with parameters Θ that maps the input $\boldsymbol{z} \in \mathbb{R}^S$ to the output $\boldsymbol{x} \in \mathbb{R}^D$ by composing L intermediate *layer* mappings f_ℓ, $\ell = 1, \dots, L$, that combine affine and simple nonlinear operators such as the *fully connected operator* (simply the affine transformation defined by the weight matrix $\boldsymbol{W}_\ell$ and bias vector $\boldsymbol{b}_\ell$), *convolution operator* (with circulent $\boldsymbol{W}_\ell$), *activation operator* (applying a scalar nonlinearity such as the ubiquitous ReLU), or *pooling operator*. Precise definitions of these operators can be found in (Goodfellow et al., 2016). We will omit Θ for conciseness unless it is needed.

We precisely define a layer f_ℓ as comprising a single nonlinear operator composed with any (if any) preceding linear operators that lie between it and the preceding nonlinear operator. Each layer f_ℓ transforms its input *feature map* $\boldsymbol{v}_{\ell-1} \in \mathbb{R}^{D_{\ell-1}}$ into an output feature map $\boldsymbol{v}_\ell \in \mathbb{R}^{D_\ell}$ with the initializations $\boldsymbol{v}_0 := \boldsymbol{z}, D_0 = S$, and $\boldsymbol{v}_L := \boldsymbol{x}, D_L = D$. In this paper, we focus on DGNs, where $S < D$, $\boldsymbol{z}$ is interpreted as a latent representation, and $\boldsymbol{x}$ is the generated data, e.g, a time-serie or image. The feature maps $\boldsymbol{v}_\ell$ can be viewed equivalently as signals, flattened column vectors, or tensors depending on the context.

Max-Affine Spline Deep Networks. A K-dimensional *max-affine spline operator* (MASO) concatenates K independent *max-affine spline* (MAS) functions, with each MAS formed from the point-wise maximum of R affine mappings (Magnani & Boyd, 2009; Hannah & Dunson, 2013). Given an input vector $\boldsymbol{u}$, the output of a MASO is given by

$$\text{MASO}(\boldsymbol{u}; \{\boldsymbol{A}_r, \boldsymbol{b}_r\}_{r=1}^R) = \max_{r=1,\dots,R} \boldsymbol{A}_r \boldsymbol{u} + \boldsymbol{b}_r, \qquad (1)$$

where $\boldsymbol{A}_r \in \mathbb{R}^{D_\ell \times D_{\ell-1}}$ are the slopes and $\boldsymbol{b}_r \in \mathbb{R}^{D_\ell}$ are the offset/bias parameters $\forall r$ and the maximum is taken coordinate-wise. Note that a MASO is a *continuous piecewise-affine* (CPA) operator (**?**).

The key background result for this paper is that *the layers of DNs (DGNs) constructed from piecewise affine operators (e.g., convolution, ReLU, and max-pooling) are MASOs* Balestriero & Baraniuk (2018b;a); hence a DN (DGN) is a composition of MASOs. For example, a layer comprising a fully connected operator with weights $\boldsymbol{W}_\ell$ and biases $\boldsymbol{b}_\ell$ followed by a ReLU activation operator has parameters $R = 2, \boldsymbol{A}_1 = \boldsymbol{W}_\ell, \boldsymbol{A}_2 = \boldsymbol{0}, \boldsymbol{b}_1 = \boldsymbol{b}_\ell, \boldsymbol{b}_2 = \boldsymbol{0}$.

The piecewise-affine spline interpretation provides a powerful global geometric interpretation of a DN (DGN) in which it partitions its input space $\mathbb{R}^L$ into polyhedral regions (the set Ω) and then assigns a different, fixed affine transformation to each region. The partition regions are built up over the layers via a *subdivision* process and are closely related to Voronoi and power diagrams (Balestriero et al., 2019).

3. The Generated Manifold of a DGN

In this section we study the properties of the mapping $\boldsymbol{G}_\Theta : \mathbb{R}^S \to \mathbb{R}^D$ of a deep generative network (DGN) comprising L piecewise-affine MASO layers.

3.1. Input Space Partition and Region Mapping

While our approach holds for arbitrary piecewise affine layers, for concreteness of exposition, we will focus on nonlinearities with $R = 2$ (e.g., ReLU, leaky ReLU, absolute value). In all such cases, the state of the nonlinearity can be encoded as a value from $\{\alpha, 1\}$ with $\alpha = 0$ for ReLU, $\alpha = -1$ for absolute value and $\alpha > 0$ for leaky-ReLU. At layer ℓ, observing an input $\boldsymbol{v}_{\ell-1}$ defines the state of the layer nonlinearities and in turn defines the "piece" of the layer MASO used to produce the output $\boldsymbol{v}_\ell$. We call the nonlinearity's state its *code* $\boldsymbol{q}_\ell(\boldsymbol{v}_{\ell-1}) \in \{\alpha, 1\}^{D_\ell}, \boldsymbol{v}_{\ell-1} \in \mathbb{R}^{D_{\ell-1}}$ with

$$[\boldsymbol{q}_\ell(\boldsymbol{v}_{\ell-1})]_i = \begin{cases} \alpha, & [\boldsymbol{W}_\ell \boldsymbol{v}_{\ell-1} + \boldsymbol{b}_\ell]_i \leq 0 \\ 1, & [\boldsymbol{W}_\ell \boldsymbol{v}_{\ell-1} + \boldsymbol{b}_\ell]_i > 0 \end{cases} \qquad (2)$$

leading to the simple forward formula for the layer

$$f_\ell(\boldsymbol{v}_{\ell-1}) = \text{diag}(\boldsymbol{q}_\ell(\boldsymbol{v}))(\boldsymbol{W}_\ell \boldsymbol{v} + \boldsymbol{b}_\ell). \qquad (3)$$

We concatenate the per-layer codes into $\boldsymbol{q}(\boldsymbol{z}) = [\boldsymbol{q}_1(\boldsymbol{z})^T, \dots, \boldsymbol{q}_L(\boldsymbol{z})^T]^T \in \{\alpha, 1\}^{\prod_{l=1}^L D_l}$ with $\boldsymbol{z} \in \mathbb{R}^S$ the DGN input and $\boldsymbol{q}_\ell(\boldsymbol{v}_{\ell-1})$ abbreviated as $\boldsymbol{q}_\ell(\boldsymbol{z})$.

Definition 1. *A* **partition region** $\omega_{\mathbf{k}}$ *of the DGN input space partition* Ω *is defined as the input space region for which the MASO states* $\boldsymbol{q}(\cdot)$ *are identical*

$$\omega_{\mathbf{k}} = \left\{\boldsymbol{z} \in \mathbb{R}^S : \boldsymbol{q}(\boldsymbol{z}) = \mathbf{k}\right\}, \forall \mathbf{k} \in \{\alpha, 1\}^{\prod_{\ell=1}^L D_\ell}, \qquad (4)$$

$$\Omega = \left\{\omega_{\mathbf{k}}, \mathbf{k} \in \{\alpha, 1\}^{\prod_{\ell=1}^L D_\ell}\right\} \setminus \emptyset. \qquad (5)$$

Note that $\cup_{\omega \in \Omega}\, \omega = \mathbb{R}^S$ and $\forall (\omega, \omega') \in \Omega^2, \omega \neq \omega', \omega^\circ \cap \omega'^\circ = \emptyset$, with $(\cdot)^\circ$ the interior operator (Munkres, 2014).

Since $\boldsymbol{G}$ is formed from a composition of MASOs, it is itself a CPA with a fixed affine mapping over each region $\omega \in \Omega$.

As a result, the generator $\boldsymbol{G}$ maps each S-dimensional convex latent space region $\omega \in \Omega$ to the convex affine subspace $\boldsymbol{G}(\omega) \subset \mathbb{R}^D$ as

$$\forall \omega \in \Omega, \;\; \boldsymbol{G}(\omega) = \{\boldsymbol{A}_\omega \boldsymbol{z} + \boldsymbol{b}_\omega, \boldsymbol{z} \in \omega\} \subseteq \mathbb{R}^D, \qquad (6)$$

with $\boldsymbol{A}_\omega$ and $\boldsymbol{b}_\omega$ obtained by composing (3) and distributing the terms; we will call this the *generated manifold*.

Proposition 1. *A DGN* $\boldsymbol{G}$ *comprised of MASO layers is a CPA operator with input space partition* Ω *given by (5). Each region* $\omega \in \Omega$ *is a convex polytope in* $\mathbb{R}^S$ *that is affinely mapped to the affine subspace* $\boldsymbol{G}(\omega)$ *(recall (6), a convex polytope in* $\mathbb{R}^D$ *given by (6). (Proof in Appendix C.1.)*

Analytically, the form of (6) can be obtained directly by composing the per layer mappings from (3), distributing and rearranging the terms into the slope and bias terms of the affine mapping. Computationally, the affine parameters $\boldsymbol{A}_\omega, \boldsymbol{b}_\omega$ can be obtained efficiently in one of the two following ways. On the one hand, if one possesses an input $\boldsymbol{z}$ belonging to the desired region ω, then we simply perform

$$\boldsymbol{A}_\omega = \nabla_{\boldsymbol{z}} \boldsymbol{G}(\boldsymbol{z}), \;\; \boldsymbol{b}_\omega = \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{A}_\omega \boldsymbol{z}. \tag{7}$$

On the other hand, in the case where one has access to the code $\boldsymbol{q}(\omega)$ of the region (as opposed to a point in the region), one can directly impose the nonlinearity states (defined by $\boldsymbol{q}(\omega)$) on the DGN mapping. Once the nonlinearities are fixed, one can feed an arbitrary input $\boldsymbol{z} \in \mathbb{R}^S$ and compute the affine parameters as in (7) on the fixed DGN.

We can extend (6) to the entire domain of the generator via

$$\boldsymbol{G}(\text{supp}(\boldsymbol{p_z})) = \bigcup_{\omega \in \Omega} \boldsymbol{G}(\omega \cap \text{supp}(\boldsymbol{p_z})), \tag{8}$$

with $\boldsymbol{p_z}$ the probability distribution on the latent space that generates $\boldsymbol{z}$, and where $\boldsymbol{G}(\text{supp}(p_{\boldsymbol{z}}))$ denotes the image of $\boldsymbol{G}$ by considering all inputs $\boldsymbol{z} \in \text{supp}(p_{\boldsymbol{z}})$ with nonzero probability, e.g., $\mathbb{R}^S$ if the latent distribution is a Gaussian, and the S-dimensional hypercube if it is a standard Uniform distribution. Thus, the generated manifold (8) combines the per-region affine transformations of the input space partition per (6). With this formulation, we now characterize the intrinsic dimension of the per-region and overall manifold mapping of $\boldsymbol{G}$.

3.2. Generated Manifold Intrinsic Dimension

We now turn into the intrinsic dimension of the per-region affine subspaces $\boldsymbol{G}(\omega))$ that comprise the generated manifold. In fact, as per (8), its dimension depends not only on the latent dimension S but also on the per layer parameters.

Lemma 1. *The intrinsic dimension of the affine subspace $\boldsymbol{G}(\omega))$ (recall (8)) has the following upper bound*

$$\dim(\boldsymbol{G}(\omega)) \leq \min\Big(S, \min_\ell \big(\text{rank}\left(\text{diag}(\boldsymbol{q}_\ell(\omega))\boldsymbol{W}_\ell\right)\big)\Big).$$

(Proof in Appendix C.2.)

We make three observations. First, we see that the choice of the nonlinearity (i.e., the choice of α) and/or the choice of the per-layer dimensions (i.e., the "width" of the DGN) are the key elements controlling the upper bound of $\dim(\boldsymbol{G})$. For example, in the case of ReLU ($\alpha = 0$) then $\dim(\boldsymbol{G}(\omega))$ is directly impacted by the number of 0s in the codes $\boldsymbol{q}_\ell(\omega)$ of each layer in addition of the rank of $\boldsymbol{W}_\ell$; this sensitivity does not occur when using other nonlinearities ($\alpha \neq 0$). Second, "bottleneck layers" impact directly the dimension of the subspace and thus should also be carefully picked based on the a priori knowledge of the target manifold intrinsic dimension. Third, we obtain the following condition relating the per-region dimension to the bijectivity and surjectivity of the mapping. The latter should be avoided in DGNs, since it implies that multiple different latent vectors will generate the same output sample.

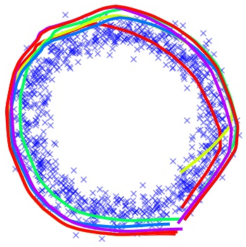


Figure 1. Demonstration of a GAN DGN trained on a circle dataset. Each line is the learned piecewise linear manifold generated by the DGN; each color corresponds to a different realization of dropout noise. Dropout turns a DGN into a finite ensemble of DGNs with the same or lower intrinsic dimension.

Proposition 2. *A DGN is bijective on ω iff $\dim(\boldsymbol{G}(\omega)) = S, \forall \omega \in \Omega$ and surjective iff $\exists \omega \in \Omega$ s.t. $\dim(\boldsymbol{G}(\omega)) < S$. A DGN is bijective on supp($\boldsymbol{p_z}$) iff it is bijective for each region $\omega \in \Omega$ and $\boldsymbol{G}(\omega) \cap \boldsymbol{G}(\omega') = \emptyset, \forall \omega \neq \omega'$. (Proof in Appendix C.3.)*

3.3. Application: Effect of Dropout/Dropconnect

Noise techniques, such as dropout (Wager et al., 2013) and dropconnect (Wan et al., 2013), alter the per-region affine mapping in a very particular way that we now characterize.

Dropout and dropconnect techniques apply a multiplicative binary noise onto the feature maps and/or the weights; the multiplicative noise $\epsilon_\ell \in \{0,1\}^{D_\ell}$ is typically an iid Bernoulli random variable. (Isola et al., 2017). To characterize how this noise impacts the DGN mapping, denote the generator $\boldsymbol{G}$ equipped with random dropout/dropconnect by $\widetilde{\boldsymbol{G}}$, and the generator in which the noise realization is fixed by $\boldsymbol{G}(\boldsymbol{z}|\boldsymbol{\epsilon}) = \boldsymbol{A}_\omega(\boldsymbol{\epsilon})\boldsymbol{z} + \boldsymbol{b}_\omega(\boldsymbol{\epsilon})$ where $\boldsymbol{\epsilon}$ concatenates the random variables of each layer. Given the mapping form (recall (3)) with per layer parameters $\boldsymbol{W}_\ell, \boldsymbol{b}_\ell$, the presence of dropout noise leads to the following noisy input-output mapping on a region ω (recall (6)) as

$$\boldsymbol{G}(\boldsymbol{z}|\boldsymbol{\epsilon}) = \left(\prod_{\ell=L}^{1} \text{diag}(\boldsymbol{q}_\ell(\omega) \odot \epsilon_\ell)\boldsymbol{W}_\ell\right)\boldsymbol{z} + \sum_{\ell=1}^{L}\left(\prod_{\ell'=L}^{\ell+1} \text{diag}(\boldsymbol{q}_{\ell'}(\omega) \odot \epsilon_{\ell'})\boldsymbol{W}_{\ell'}\right)\boldsymbol{b}_\ell, \tag{9}$$

where $\odot$ is the Hadamard product and $\boldsymbol{z} \in \omega$. (See Appendix G for the dropconnect formula.) Thus, the noisy generator actually combines all the above mappings for each noise realisation via

$$\widetilde{\boldsymbol{G}}(\text{supp}(p_{\boldsymbol{z}})) = \bigcup_{\boldsymbol{\epsilon} \in \text{supp}(p_{\boldsymbol{\epsilon}})} \boldsymbol{G}(\text{supp}(p_{\boldsymbol{z}})|\boldsymbol{\epsilon}). \tag{10}$$

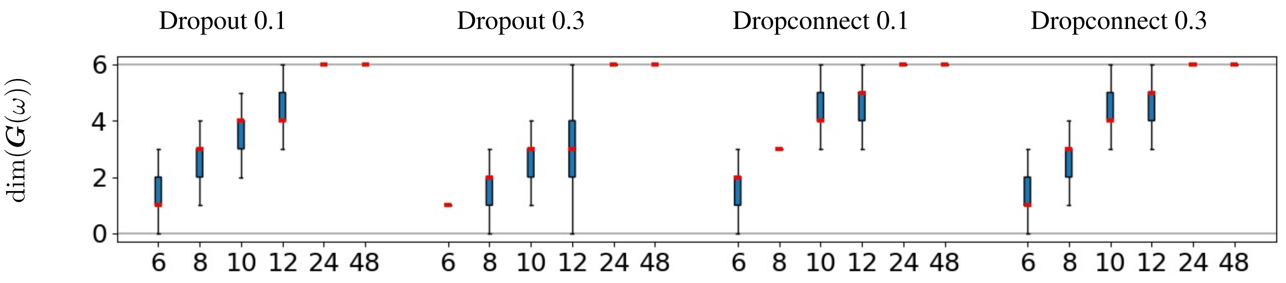


Figure 2. Impact of dropout and dropconnect on the dimension of the noisy generator affine subspaces $\boldsymbol{G}(\omega|\epsilon), \forall\omega$ (recall (10)). We depict two "drop" probabilities 0.1 and 0.3 for a generator $\boldsymbol{G}$ with $S = 6$, $D = 10$, $L = 3$ and varying width $D_1 = D_2$ ranging in $\{6, 8, 10, 12, 24, 48\}$ (x-axis); note that the architecture limits the dimension to $S = 6$. The boxplot represents the distribution of the per-region affine subspace dimensions for 2000 sampled regions over 2000 different noise realizations ϵ. We make two observations. First, dropconnect tends to preserve the latent dimension S even when the width D_1, D_2 is close to S. Second, the dropout-induced collection of generators tends to have degenerate dimension (much smaller than S) until the width is twice the latent space dimension ($D_\ell \geq 2S$). As a result, while dropout turns a generator into a collection of generators, those generators will have degenerate dimension unless $\boldsymbol{G}$ is much wider that S.

Proposition 3. *Multiplicative binary dropout/dropconnect transforms the generator $\boldsymbol{G}$ into the union of generators given by (10), each with per-region dimension between* 0 *and* $\max_\omega$ *dim*$(\boldsymbol{G}(\omega))$. *(Proof in Appendix C.4.)*

From the above, we see that the multiplicative binary noise does not make the generator $\widetilde{\boldsymbol{G}}$ dense in its output space, but rather turns $\boldsymbol{G}$ into a collection of piecewise linear generators, each corresponding to a different realization of $\boldsymbol{\epsilon}$ as depicted in Fig. 1. Furthermore, each noise realization produces a generator with a possibly different per-region dimension being upper bounded by the dimension of the original generator $\boldsymbol{G}$. Also, each induced generator has a possibly different input space partition based on the noise realisation. On the one hand, this highlights a potential limitation of those techniques for narrow models ($D_\ell \approx S$) for which the noisy generators will tend to be degenerate (per-region dimension smaller than S), implying surjectivity (recall Prop. 2). On the other hand, when used with wide DGNs ($D_\ell \gg S$) much more noisy generators will maintain the same affine subspace dimension that the original generator. The latter is crucial when S is picked a priori to match exactly the true intrinsic dimension. We illustrate the above in Fig. 2.

3.4. Application: Optimal Dimension and Training Error Increase

We now emphasize how the DGN dimension $\tilde{S} \triangleq \max_\omega \dim(\boldsymbol{G}(\omega))$ impacts the training error loss and training sample recovery. We answer the following question: *Can a generator generate N training samples from a continuous S^* dimensional manifold if $\tilde{S} < S^*$?* Denote the empirical error measuring the ability to generate the data samples by $E^*_N = \min_\Theta \frac{1}{N}\sum_{n=1}^{N} \min_{\boldsymbol{z}} \|\boldsymbol{G}_\Theta(\boldsymbol{z}) - \boldsymbol{x}_n\|$. We now demonstrate and empirically validate that if $\tilde{S} < S^*$ then E^*_N increases with N for any data manifold.

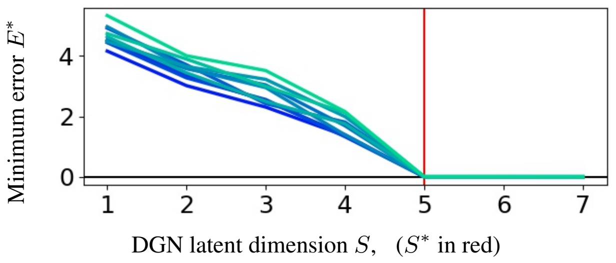


Figure 3. Error E^* (y-axis) for a linear manifold with $S^* = 5$, for increasing dataset size $N \in \{100, 120, 140, 160, 180, 200, 300, 400, 500, 1000\}$ (blue to green) for different latent space dimension $S \in \{1, 2, 3, 4, 5, 6, 7\}$ (x-axis) which forces $\tilde{S} < S$, the $E^* = 0$ line is depicted in black. This demonstrates, as per Thm. 1, that whenever $\tilde{S} < S^*$, the training error E^* increases with the dataset size N and is 0 otherwise whenever $\tilde{S} \geq S^*$.

Theorem 1. *Given the true intrinsic dimension of an arbitrary manifold S^*, any finite DGN with $\tilde{S} < S^*$ will have increasing error E^* with the dataset size N as in $\exists \ell \in \{0, \dots, L\} : D_\ell < S^* \implies \forall N > 0, \exists N' > N : E^*_{N'} > E^*_N$. (Proof in Appendix C.11.)*

In general, it is clear that, for smooth manifolds, E^* increases with N since the DGN is piecewise linear. However, the above result extends this to any manifold, even when the data manifold is as simple as a linear manifold (translated subspace). We empirically validate this phenomenon in Fig. 3 for the simple case of a linear manifold. (The experimental details are given in Appendix F.)

It is clear that a direct implication of Thm. 1 is that there does not exist a finite architecture with $D_\ell < D$ for some ℓ and parameter Θ such that a MASO DGN would be bijective with $\mathbb{R}^D$. The above results are key to understand the challenges and importance of the design of DGN starting with the width of the hidden layers and latent dimensions in conjunction with the choice of nonlinearities and constraints on $\boldsymbol{W}_\ell$ of all layers.

4. Manifold Local Coordinates and Curvature

We now turn to the study of the local coordinates of the affine mappings comprising a DGN's generated manifold. We then study the coupling between the affine mappings of adjacent regionsto characterize the curvature/angularity of the generated manifold.

4.1. Local Coordinate Systems and Inverse Mapping

Recall from (8) that a DGN is a CPA operator. Inside region $\omega \in \Omega$, points are mapped to the output space affine subspace which is itself governed by a coordinate system or basis. For the remaining of the section we assume that $\dim(\boldsymbol{G}(\omega)) = S, \forall \omega \in \Omega$ and thus the columns of $\boldsymbol{A}_\omega$ are linearly independent. For cases where $\dim(\boldsymbol{G}(\omega)) < S$ then the following analysis also applies by considering a lower dimensional latent space $S' < S$ and the corresponding sub-network that only depend on the kept latent space dimensions.

Lemma 2. *A basis for the affine subspace $\boldsymbol{G}(\omega)$ (recall (6)) is given by the columns of $\boldsymbol{A}_\omega$.*

In other words, the columns of $\boldsymbol{A}_\omega$ form the local coordinate space, and each latent space dimension moves a point in this region by adding to it the corresponding slope column. Prior leveraging this result for latent space characterization, we derive an inverse of the generator $\boldsymbol{G}$ that maps any point from the generated manifold to the latent space. This inverse is well-defined as long as the generator is injective, preventing that $\exists \boldsymbol{z}_1 \neq \boldsymbol{z}_2$ s.t. $\boldsymbol{G}(\boldsymbol{z}_1) = \boldsymbol{G}(\boldsymbol{z}_2)$. Assuming injectivity, the inverse of $\boldsymbol{G}$ on a region $\boldsymbol{G}(\omega)$ in the output space is obtained by

$$\boldsymbol{G}_\omega^{-1}(\boldsymbol{x}) = \left(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega\right)^{-1} \boldsymbol{A}_\omega^T (\boldsymbol{x} - \boldsymbol{b}_\omega), \forall \boldsymbol{x} \in \boldsymbol{G}(\omega), \quad (11)$$

leading to $\boldsymbol{G}_\omega^{-1}(\boldsymbol{G}(\omega)) = \omega, \forall \omega \in \Omega$. Note that the inverse $\left(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega\right)^{-1}$ is well defined as $\boldsymbol{A}_\omega$ is full column rank since we only consider a generator with $\tilde{S} = S$. We can then simply combine the region-conditioned inverses to obtain the overall generator inverse.

Lemma 3. *The inverse mapping of an injective DGN is the CPA operator mapping $\boldsymbol{G}(supp(\boldsymbol{p_z})) \mapsto supp(\boldsymbol{p_z})$ given by $\boldsymbol{G}^{-1}(\boldsymbol{x}) = \sum_{\omega \in \Omega} \boldsymbol{G}_\omega^{-1}(\boldsymbol{x}) \mathbb{1}_{\{\boldsymbol{x} \in \boldsymbol{G}(\omega)\}}$. (Proof in Appendix C.5.)*

4.2. Application: Adaptive Basis and Disentenglement

As mentioned in the above section, $\boldsymbol{A}_\omega$ forms a basis of the affine subspace $\boldsymbol{G}(\omega)$. The latent vector $\boldsymbol{v}$ combines them to obtain the subspace which is then shifted by the bias $\boldsymbol{b}_\omega$.

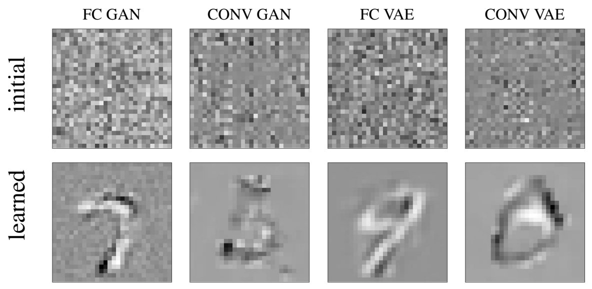


Figure 4. Visualization of a single basis bector $[\boldsymbol{A}_\omega]_{.,k}$ with ω at initialization and after learning obtained from a region ω containing the digits $7, 5, 9,$ and 0 respectively, and this for GAN and VAE models made of fully connected or convolutional layer (see Appendix B for details). By visual inspection, one can observe that the depicted basis vector encodes right rotation, cedilla extension, left rotation, and upward translation respectively. In addition, we observe how the basis vectors are smoother for VAE-based models which why they tend to generate blurred samples.

This process is performed locally for each region ω, in a manner similar to an "adaptive basis" (Donoho et al., 1994).

In this context, we aim to characterize the subspace basis in term of *disentanglement*, i.e., the alignment of the basis vectors with respect to each other. While there is not a unique definition, a disentangled basis should provide a "compact" and interpretable latent representation $\boldsymbol{z}$ for the associated $\boldsymbol{x} = \boldsymbol{G}(\boldsymbol{z})$. In particular, it should ensure that a small perturbation of dimension d of $\boldsymbol{z}$ implies a transformation independent from a small perturbation of dimension $d' \neq d$ (Schmidhuber, 1992; Bengio et al., 2013). That is, $\langle \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{G}(\boldsymbol{z} + \epsilon\delta_d), \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{G}(\boldsymbol{z} + \epsilon\delta_{d'}) \rangle \approx 0$ with δ_d a one-hot vector at position d and length Z (Kim & Mnih, 2018). A disentangled representation is thus considered to be most informative as each latent dimension imply a transformation that leaves the others unchanged (Bryant & Yarnold, 1995). For example, rotating an object should not alter its vertical or horizontal translation and vice-versa

Proposition 4. *A necessary condition for disentanglement is to have "near orthogonal" columns as $\langle [\boldsymbol{A}_\omega]_{.,i}, [\boldsymbol{A}_\omega]_{.,j} \rangle \approx 0, \forall i, \neq j, \forall \omega \in \Omega$. (Proof in Appendix C.6.)*

Figure 4 visualizes one of the basis vectors of four different DGNs trained on the MNIST dataset with $S = 10$. Interpretability of the transformation encoded by the dimension of the basis vector can be done as well as model comparison such as blurriness of VAE samples that is empirically observed across datasets (**??**). We also provide in Table 1 the value of $\|Q_\omega - I\|_2$ with $Q_\omega \in [0, 1]^{S \times S}$ the matrix of cosine angles between basis vector of $\boldsymbol{A}_\omega$ for 10,000 regions sampled randomly and where we report the average over the regions and the maximum. Finally, this process is performed over 8 runs, the mean and standard deviation are reported in the table. We observe that there does not seem

Table 1. Depiction of the cosine similarity summed over pairwise different columns of $\boldsymbol{A}_\omega$. Measure of 0 means that the basis vectors are orthogonal, improving disentanglement (recall Prop. 4). The first line represent the maximum of this quantity over 10000 sampled regions ω, the second line represents the average; the std of those quantities is given for 8 runs. We see that training increases disentanglement, and fully connected models offer increased disentanglement as compared to convolutional models.

	FC GAN	CONV GAN	FC VAE	CONV VAE
init.	8.84 ± 0.07	3.2 ± 0.33	5.23 ± 0.29	3.5 ± 0.27
	4.41 ± 0.26	1.84 ± 0.08	2.25 ± 0.08	1.74 ± 0.06
learn	$1.36 \pm .08$	1.72 ± 0.07	1.32 ± 0.07	1.77 ± 0.11
	0.9 ± 0.03	1.12 ± 0.03	0.89 ± 0.03	1.15 ± 0.03

to be a difference in the degree of disentanglement different GAN and VAE; however, the topology, fully connected vs. convolution, plays an important part, favoring the former. To visually control the quality of the DGN, randomly generated digits are given in Fig. 13 in the Appendix; we also provide more background on disentanglement in Appendix E.

4.3. Generated Manifold Curvature

We now study the curvature or angularity of the generated mapping. That is, whenever $\widetilde{S} < D$, the per-region affine subspace of adjacent region are continuous, and joint at the region boundaries with a certain angle that we now characterize.

Definition 2. *Two regions ω, ω' are adjacent whenever they share part of their boundary as in $\overline{\omega} \cap \overline{\omega'} \neq \emptyset$.*

The angle between adjacent affine subspace is characterized by means of the greatest principal angle (Afriat, 1957; Bjorck & Golub, 1973) and denote θ. Denote the per-region projection matrix of the DGN by

$$P(\boldsymbol{A}_\omega) = \boldsymbol{A}_\omega(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)^{-1}\boldsymbol{A}_\omega^T \tag{12}$$

where $\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega \in \mathbb{R}^{S \times S}$ and $P(\boldsymbol{A}_\omega) \in \mathbb{R}^{D \times D}$. We now assume that $\dim(\boldsymbol{G}) = Z$ ensuring that $\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega$ is invertible.[2]

Theorem 2. *The angle between adjacent (recall Def. 2) region mappings $\theta(\boldsymbol{G}(\omega), \boldsymbol{G}(\omega'))$ is given by*

$$\sin\left(\theta(\boldsymbol{G}(\omega), \boldsymbol{G}(\omega'))\right) = \|P(\boldsymbol{A}_\omega) - P(\boldsymbol{A}_{\omega'})\|_2, \tag{13}$$

$\forall \omega \in \Omega, \omega' \in adj(\omega)$. (Proof in Appendix C.7.)

Notice that two special cases of the above theorem emerge. When $S = 1$, the angle is given by the cosine similarity between the vectors $\boldsymbol{A}_\omega$ and $\boldsymbol{A}_{\omega'}$ of adjacent regions. When $S = D - 1$ the angle is given by the cosine similarity between the normal vectors of the $D - 1$ subspace spanned by $\boldsymbol{A}_\omega$ and $\boldsymbol{A}_{\omega'}$ respectively.

[2]The derivation also applies if $\dim(\boldsymbol{G}) < Z$ by replacing $\boldsymbol{A}_\omega$ with $\boldsymbol{A}'_\omega$ (recall Lemma 2).

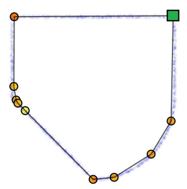


Figure 5. Piecewise linear continuous 1-D manifold learned by a GAN DGN (in black) from a collection of data points (in blue). The breakpoints between adjacent regions are depicted by dots of color proportional to the angle. The manifold starts at the green box and proceeds clockwise. Figure 11 in the Appendix contains additional examples.

We illustrate the angles in a simple case $D = 2$ and $Z = 1$ in Fig. 5. It can be seen how a DGN with few parameters produces angles mainly at the points of curvature of the manifold. We also provide many additional figures with different training settings in Fig. 10 in the Appendix as well as repetitions of the same experiment with different random seeds.

4.4. Application: Angle Distribution of a DGN with Random Weights

We can use the above result to study the distribution of angles of different DGNs with random weights and study the impact of depth, width, as well Z and D, the latent and output dimensions respectively. Figure 6 summarizes the distribution of angles for several different settings.

Two key observations emerge. First, the codes of adjacent regions $\boldsymbol{q}(\omega), \boldsymbol{q}(\omega')$ share a large number of their values (see Appendix D for details) implying that most of the DGN parameters are shared in their composition to produce $\boldsymbol{A}_\omega$ and $\boldsymbol{A}_{\omega'}$. In turn, this *weight sharing* correlates adjacent hyperplanes, such that their angles are much smaller than if randomly picked form one another. The random case (in blue in Fig. 6) favors aggressively large angles as opposed to the ones of DGNs. Second, the distribution moments depend on the ratio S/D rather that those values taken independently. In particular, as this ratio gets smaller, as the angle distribution becomes bi-modal with an emergence of high angles. That is, the manifold is "flat" overall except in some parts of the space where high angularity is present. This effect is strengthened with wider DGNs. Notice that this large ratio S/D is the one encountered in practice where it is common to have $S \approx 100$ and $D > 800$.

The above experiment demonstrates the impact of width and latent space dimension into the angularity of the DGN output manifold and how to pick its architecture based on a priori knowledge of the target manifold. Under the often-made assumptions that the weights of overparametrized DGN do not move far from their initialization during training (Li & Liang, 2018), these results also hint at the distribution of angles after training.

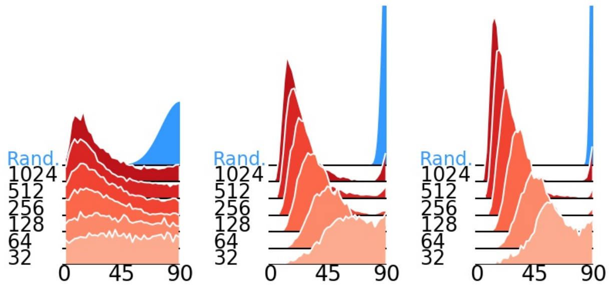


Figure 6. Histograms of the largest principal angles for DGNs with two hidden layers, $S = 16$ and $D = 17, D = 32, D = 64$ respectively and varying width D_ℓ on the y-axis. Three trends to observe: When the width becomes large, the distribution becomes more bimodal and greatly favors near 0 angles; when the output space dimension becomes large, there is an increase in the number of angles near orthogonal; the amount of *weight sharing* between the parameters $\boldsymbol{A}_\omega$ and $\boldsymbol{A}_{\omega'}$ of adjacent regions ω and ω' allow to greatly constrain the angles to be small, as opposed to the distribution of angles between random subspaces (Absil et al., 2006) depicted in blue. Hence despite the large amount of regions, most will be aligned with each other leading to an overall well behave manifold. Additional figures are available in Fig. 10 in the Appendix.

5. Density on the Generated Manifold

The study of DGNs would not be complete without considering that the latent space is equipped with a density distribution $\boldsymbol{p}_{\boldsymbol{z}}$ from which $\boldsymbol{z}$ are sampled in turn leading to sampling of $\boldsymbol{G}(\boldsymbol{z})$. Thus, we now study how this density is spread over the output space, covering the generated manifold and highlighting some key properties such as density conentration, entropy computation and training instabilities.

5.1. Analytical Output Density

Given a distribution $\boldsymbol{p}_{\boldsymbol{z}}$ over the latent space, we can explicitly compute the output distribution after the application of $\boldsymbol{G}$, which lead to an intuitive result exploiting the piecewise affine property of the generator.

Lemma 4. *Denote by $\sigma_i(\boldsymbol{A}_\omega)$ the i^{th} singular value of $\boldsymbol{A}_\omega$. Then, the volume of a region $\omega \in \Omega$ denoted by $\mu(\omega)$ is related to the volume of $\boldsymbol{G}(\omega)$ by*

$$\mu(\boldsymbol{G}(\omega)) = \sqrt{\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)}\mu(\omega) = \prod_{i:\sigma_i(\boldsymbol{A}_\omega)>0} \sigma_i(\boldsymbol{A}_\omega)\mu(\omega).$$

(Proof in Appendix C.8.)

Theorem 3. *The generator probability density $p_{\boldsymbol{G}}(\boldsymbol{x})$ given $\boldsymbol{p}_{\boldsymbol{z}}$ and a injective generator $\boldsymbol{G}$ with per-region inverse $\boldsymbol{G}_\omega^{-1}$ from (11) is given by*

$$p_{\boldsymbol{G}}(\boldsymbol{x}) = \sum_{\omega\in\Omega} \frac{\boldsymbol{p}_{\boldsymbol{z}}\left(\boldsymbol{G}_\omega^{-1}(\boldsymbol{x})\right)}{\sqrt{\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)}} \mathbb{1}_{\{\boldsymbol{x}\in\boldsymbol{G}(\omega)\}}. \tag{14}$$

(Proof in Appendix C.9.)

That is, the distribution obtained in the output space naturally corresponds to a piecewise affine transformation of the original latent space distribution, weighted by the change in volume of the per-region mappings.

From the analytical derivation of the generator density distribution, we obtain its differential entropy, i.e., the Shannon entropy for continuous distributions.

Corollary 1. *The differential entropy of the output distribution $\boldsymbol{p}_{\boldsymbol{G}}$ of the DGN is given by*

$$E(\boldsymbol{p}_{\boldsymbol{G}}) = E(\boldsymbol{p}_{\boldsymbol{z}}) + \sum_{\omega\in\Omega} P(\boldsymbol{z}\in\omega)\log(\sqrt{\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)}).$$

(Proof in Appendix C.10.)

As the result, the differential entropy of the output distribution $\boldsymbol{p}_{\boldsymbol{G}}$ corresponds to the differential entropy of the latent distribution $\boldsymbol{p}_{\boldsymbol{z}}$ plus a convex combination of the per-region volume change. Two results emerge directly. First, it is possible to optimize the latent distribution $\boldsymbol{p}_{\boldsymbol{z}}$ to better fit the target distribution entropy as been done for example in (Ben-Yosef & Weinshall, 2018). Second, whenever this distribution is fixed, any gap between the latent and output distribution entropy imply the need for high change in volumes between ω and $\boldsymbol{G}(\omega)$.

Gaussian Case. We now demonstrate the use of the above derivation by considering practical examples for which we are able to gain insights into the DGN data modeling and generation. First, consider that the latent distribution is set as $\boldsymbol{z} \sim \mathcal{N}(0, 1)$ we obtain the following result directly from Thm. 3.

Corollary 2. *The generator density distribution $p_{\boldsymbol{G}}(\boldsymbol{x})$ given $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I)$ is*

$$p_{\boldsymbol{G}}(\boldsymbol{x}) = \sum_{\omega\in\Omega} \frac{e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{b}_\omega)^T(\boldsymbol{A}_\omega^+)^T\boldsymbol{A}_\omega^+(\boldsymbol{x}-\boldsymbol{b}_\omega)}}{\sqrt{(2\pi)^S\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)}} \mathbb{1}_{\{\boldsymbol{x}\in\boldsymbol{G}(\omega)\}}.$$

(Proof in Appendix C.12.)

The above formula is reminiscent of Kernel Density Estimation (KDE) (Rosenblatt, 1956) and in particular adaptive KDE (Breiman et al., 1977), where a partitioning of the data manifold is performed on each cell (ω in our case) different kernel parameters are used.

Uniform Case. We now turn into the uniform latent distribution case. Consider the following question: *Suppose we start from a uniform distribution $\boldsymbol{z} \sim \mathcal{U}(0, 1)$ on the hypercube in $\mathbb{R}^S$, will the samples be uniformly distributed on the manifold of $\boldsymbol{G}$?*

Proposition 5. *Given a uniform latent distribution $\boldsymbol{v} \sim \mathcal{U}(0, 1)$, the sampling of the manifold $\boldsymbol{G}(supp(\boldsymbol{p}_{\boldsymbol{z}}))$ will be uniform iff $\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega) = c, \forall\omega : \omega \cap supp(\boldsymbol{p}_{\boldsymbol{z}}) \neq \emptyset, c > 0$.*

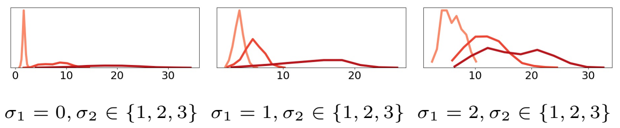


Figure 7. Distribution of $\log(\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)})$ for 2000 regions ω with a DGN with $L = 3, S = 6, D = 10$ and weights initialized with Xavier; then, half of the weights' coefficients (picked randomly) are rescaled by σ_1 and the other half by σ_2. We observe that greater variance of the weights increase the spread of the log-determinants and increase the mean of the distribution.

5.2. Generative Deep Network Likelihood and Normalizing Flows

Note from Thm. 3 that we obtain an explicit density distribution. One possibility for learning thus corresponds to minimizing the negative log-likelihood (NLL) between the generator output distribution and the data. Recall from Thm. 3 that $\sqrt{\det((\boldsymbol{A}_\omega^+)^T \boldsymbol{A}_\omega^+)} = (\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)})^{-1}$; thus we can write the log density from (14) over a sample $\boldsymbol{x}_n$ as $\mathcal{L}(\boldsymbol{x}_n) = \log(\boldsymbol{p}_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}_n))) + \log(\sqrt{\det(J(\boldsymbol{x}_n))})$, where $J(\boldsymbol{x}_n) = J_{\boldsymbol{G}^{-1}}(\boldsymbol{x}_n)^T J_{\boldsymbol{G}^{-1}}(\boldsymbol{x}_n)$, J the Jacobian operator. Learning the weights of the DGN by minimization of the NLL given by $-\sum_{n=1}^{N} \mathcal{L}(\boldsymbol{x}_n)$, corresponds to the normalizing flow model. The practical difference between this formulation and most NF models comes from having either a mapping from $\boldsymbol{x} \mapsto \boldsymbol{z}$ (NF) or from $\boldsymbol{z} \mapsto \boldsymbol{x}$ (DGN case). This change only impacts the speed to either sample points or to compute the probability of observations. In fact, the forward pass of a DGN is easily obtained as opposed to its inverse requiring a search over the codes $\boldsymbol{q}$ itself requiring some optimization. Thus, the DGN formulation will have inefficient training (slow to compute the likelihood) but fast sampling while NMFs will have efficient training but inefficient sampling.

5.3. On the Difficulty of Generating Low entropy/Multimodal Distributions

We conclude this section with the study of the instabilities encountered when training DGNs on multimodal densities or other atypical cases.

We demonstrated in Thm. 3 and Cor. 1 that the product of the nonzero singular values of $\boldsymbol{A}_\omega$ plays the central role to concentrate or disperse the density on $\boldsymbol{G}(\omega)$. Let now consider a simple case of mixture of Gaussians. It becomes clear that the standard deviation of the modes and the inter-mode distances will put constraints on the singular values of the slope matrix $\boldsymbol{A}_\omega$. However, imposing a large variance in the singular values of $\boldsymbol{A}_\omega$ for different regions ω directly stress the parameters $\boldsymbol{W}_\ell$ as they compose the slope matrix. This is highlighted in Fig. 7 where we depict the distribution of the log-determinants for different bimodal

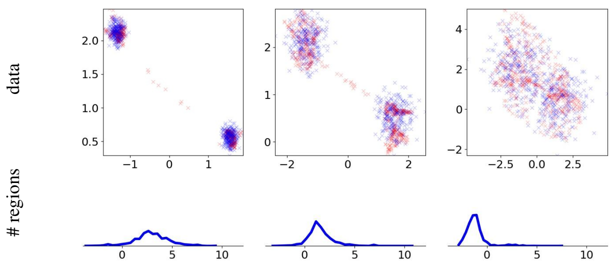


Figure 8. Distribution of the log-determinant of the per-region mappings for different true distributions: two Gaussian with increasing standard deviation (left to right). We observe how the concentration and inter-mode distance impacts greatly the distribution of the log-determinant to allow the generator to fit the distribution, this in turns increases the variance of the weights $\boldsymbol{W}_\ell$.

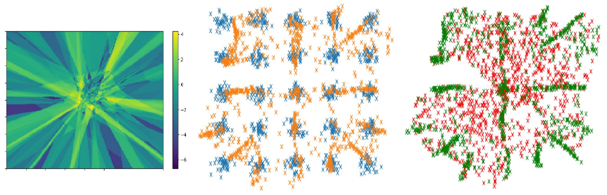


Figure 9. Example of a GAN DGN trained on a mixture of 25 Gaussians. On the **left** is depicted the latent space per-region log-determinant (color coded) with values ranging from -6 to 4 in log-scale (0.002 to 55 in linear scale). On the **middle** are depicted the true sample (blue) and generated ones (oranges), and on the **right** are depicted points sampled from the generator from regions with low determinant (green) and large determinant (red). We can observe that this failure case (poor approximation of the true distribution) is due to the continuous properties of MASO DGNs, which makes the generator move continuously between modes while not being able to reduce enough the sampling probability $\boldsymbol{p}_{\boldsymbol{G}}$ in between the modes. Additional examples are contained in Fig. 14

distribution for the weights $\boldsymbol{W}_\ell$ showing the correlation between the variance of those parameters and the variance of log-determinant over different regions.

We also illustrate the variation of the learned generator log-determinant distribution across regions in Fig. 8, where we trained a GAN DGN on two Gaussians for different scalings. This further highlights the importance of the distribution of the determinants that is reachable by a DGN which depends on the architecture and parameter space. In conclusion, for multimodal and low entropy distribution, the required log-determinant for the DGN to approximate the true distribution goes against some standard regularization techniques such as Tikhonov, which (recall Fig. 7) pushes the generator output density $\boldsymbol{p}_{\boldsymbol{G}}$ to be more uniform with higher entropy.

SUPPLEMENTARY MATERIAL

A. Extra Figures

All the below pictures have been compressed to be uploaded on Arxiv.

A.1. Angles Histogram

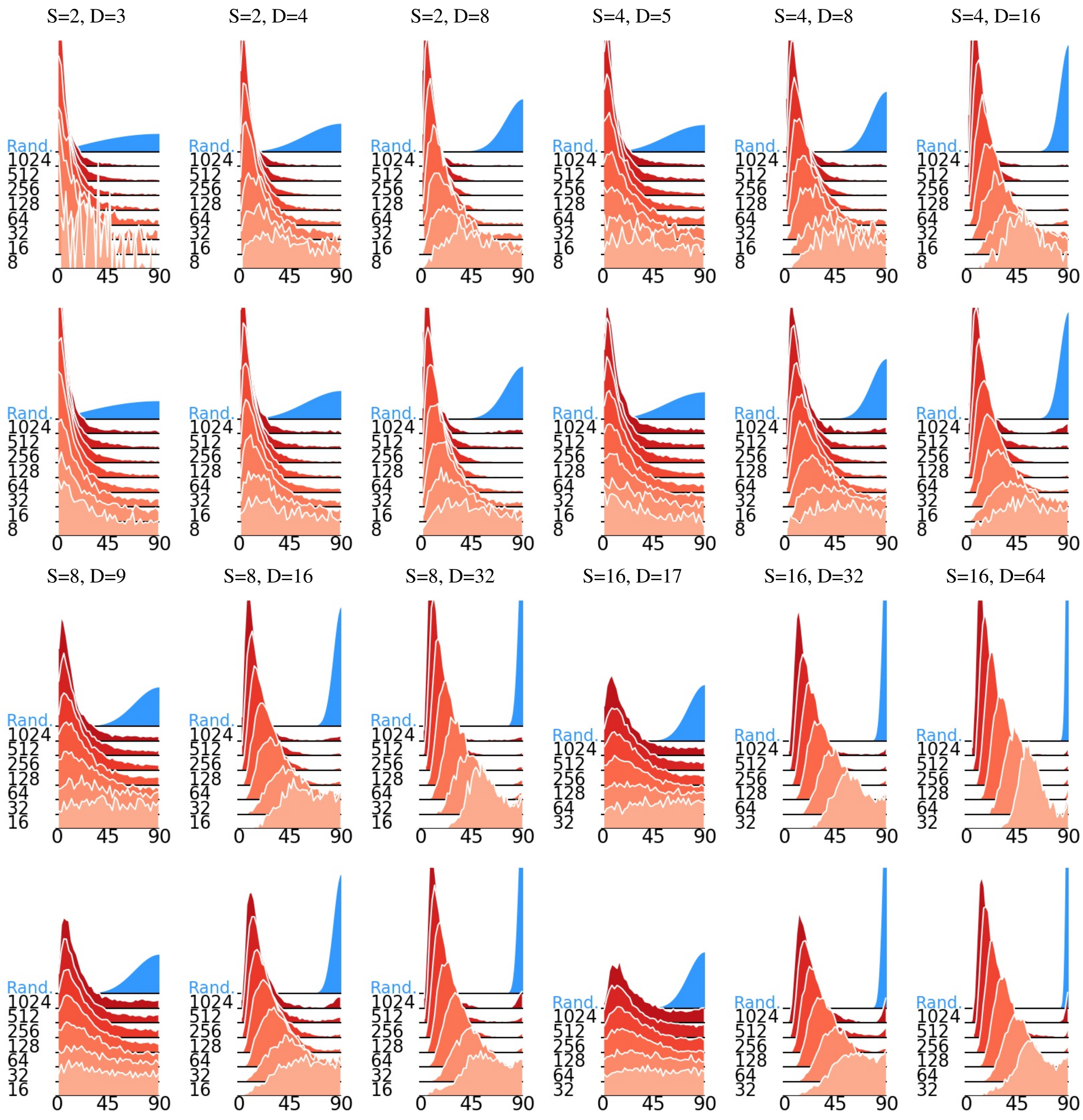


Figure 10. Reproduction of Fig. 6. Histograms of the largest principal angles for DGNs with one hidden layer (first two rows) and two hidden layers (last two rows). In each case the latent space dimension and width of the hidden layers is in the top of the column. The observations reinforce the claims on the role of width and S versus D dimensions.

A.2. Angles Manifold

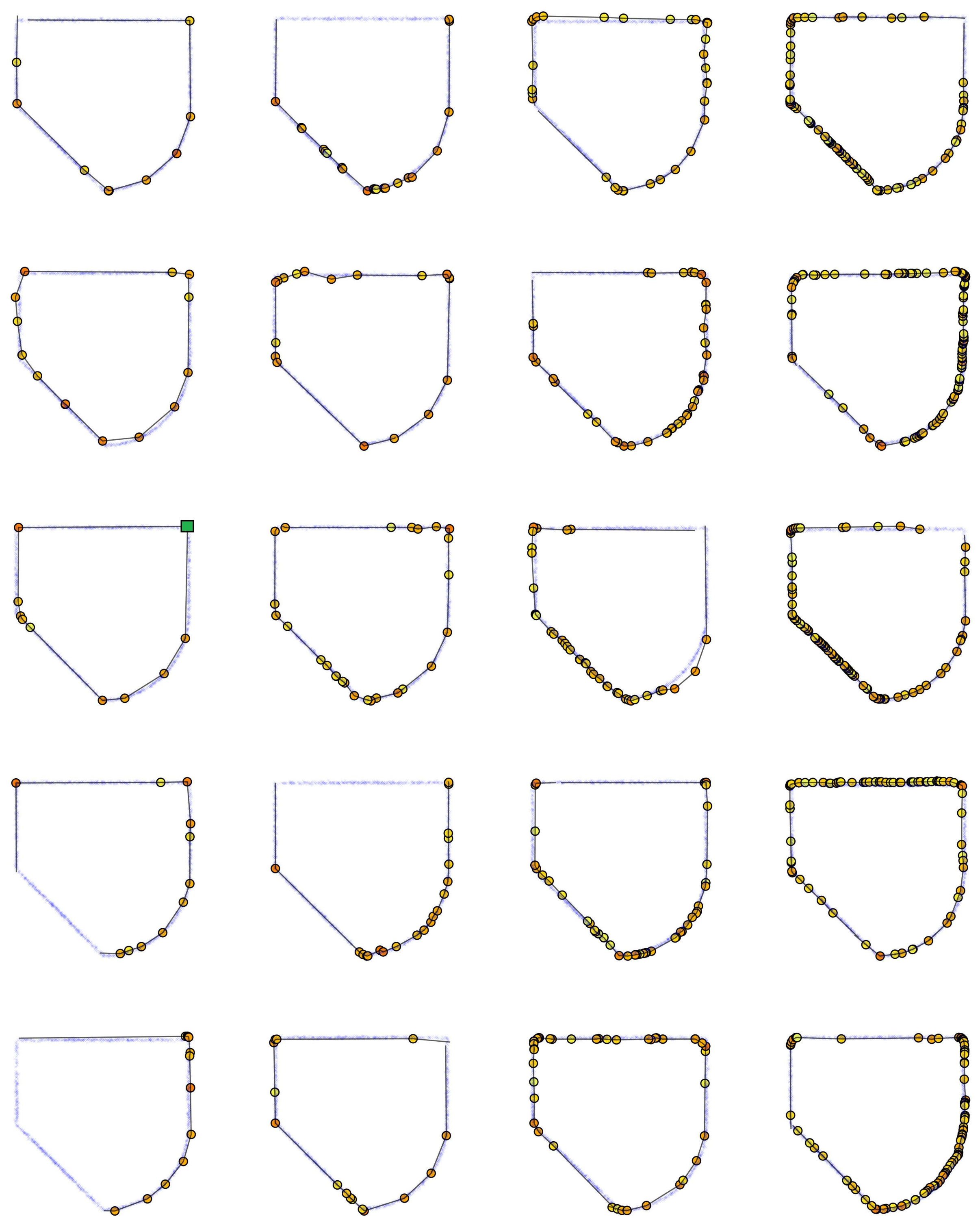


Figure 11. Reproduction of Fig. 5 for various GDNs topologies. The columns represent different widths $D_\ell \in \{6, 8, 16, 32\}$ and the rows correspond to repetition of the learning for different random initializations of the GDNs for consecutive seeds.

A.3. More on MNIST Disentanglement

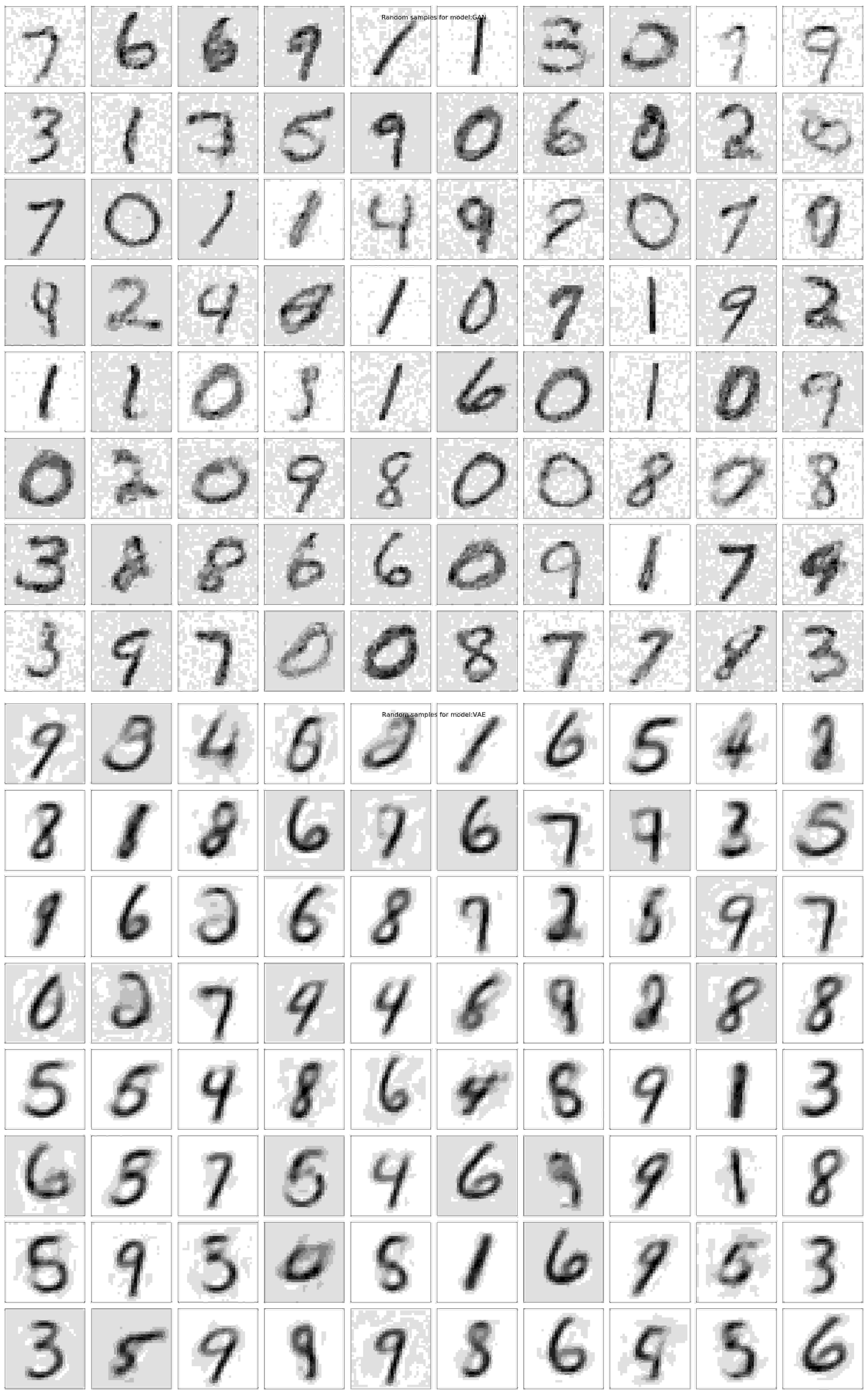
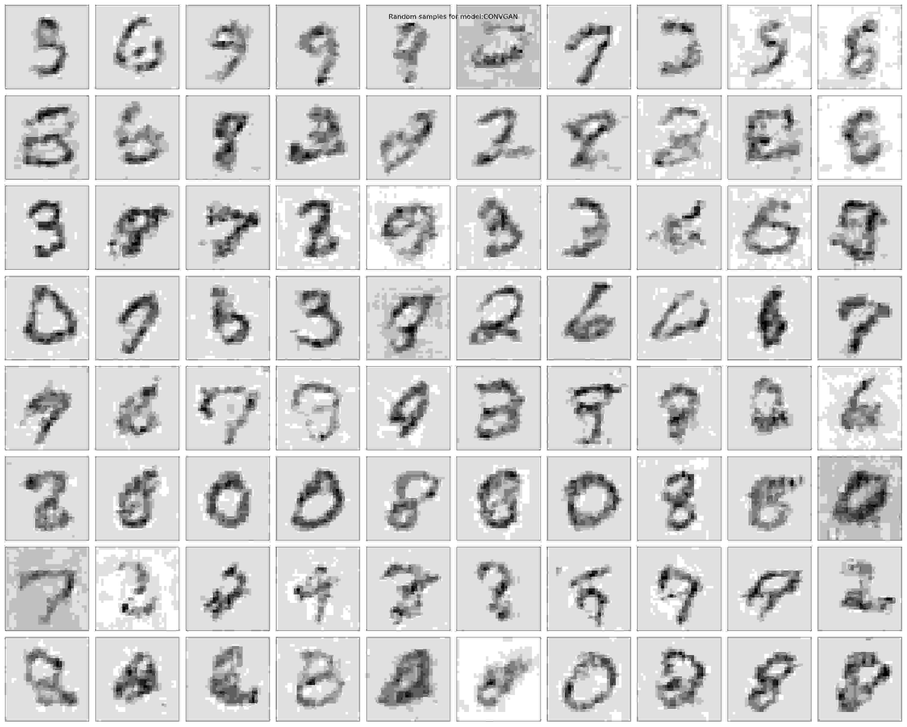


Figure 12. Randomly generated digits from the trained GAN (top) and trained VAE(bottom) models for the experiment from Fig. 4. Each row represents a model that was training on a different random initialization (8 runs in total) which produced the result in Table 1.



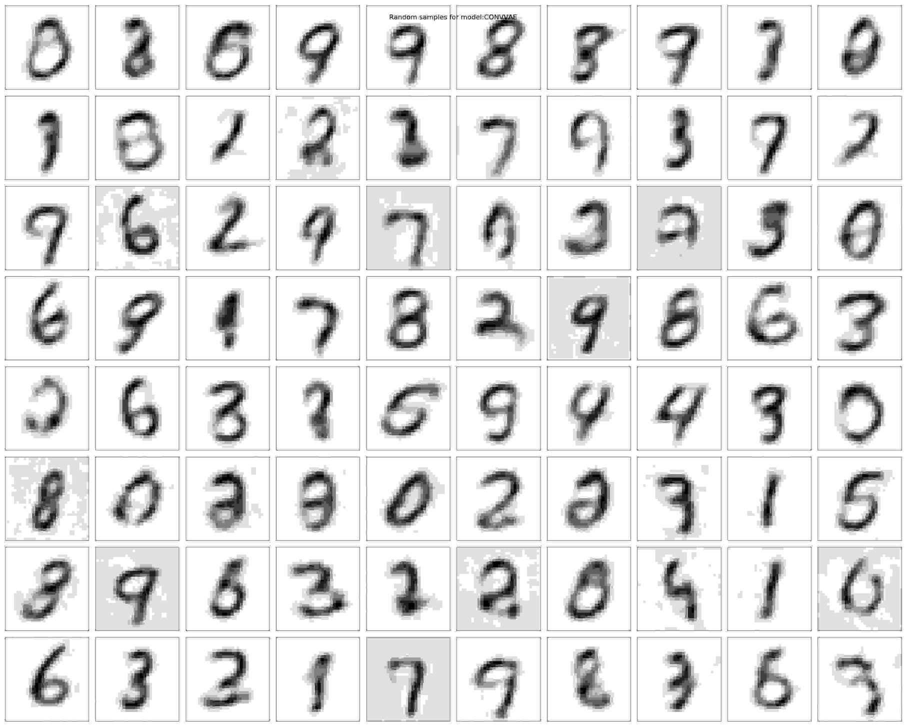


Figure 13. Randomly generated digits from the trained CONV GAN (top) and trained CONV VAE(bottom) models for the experiment from Fig. 4. Each row represents a model that was training on a different random initialization (8 runs in total) which produced the result in Table 1.

A.4. More on Determinant Figures

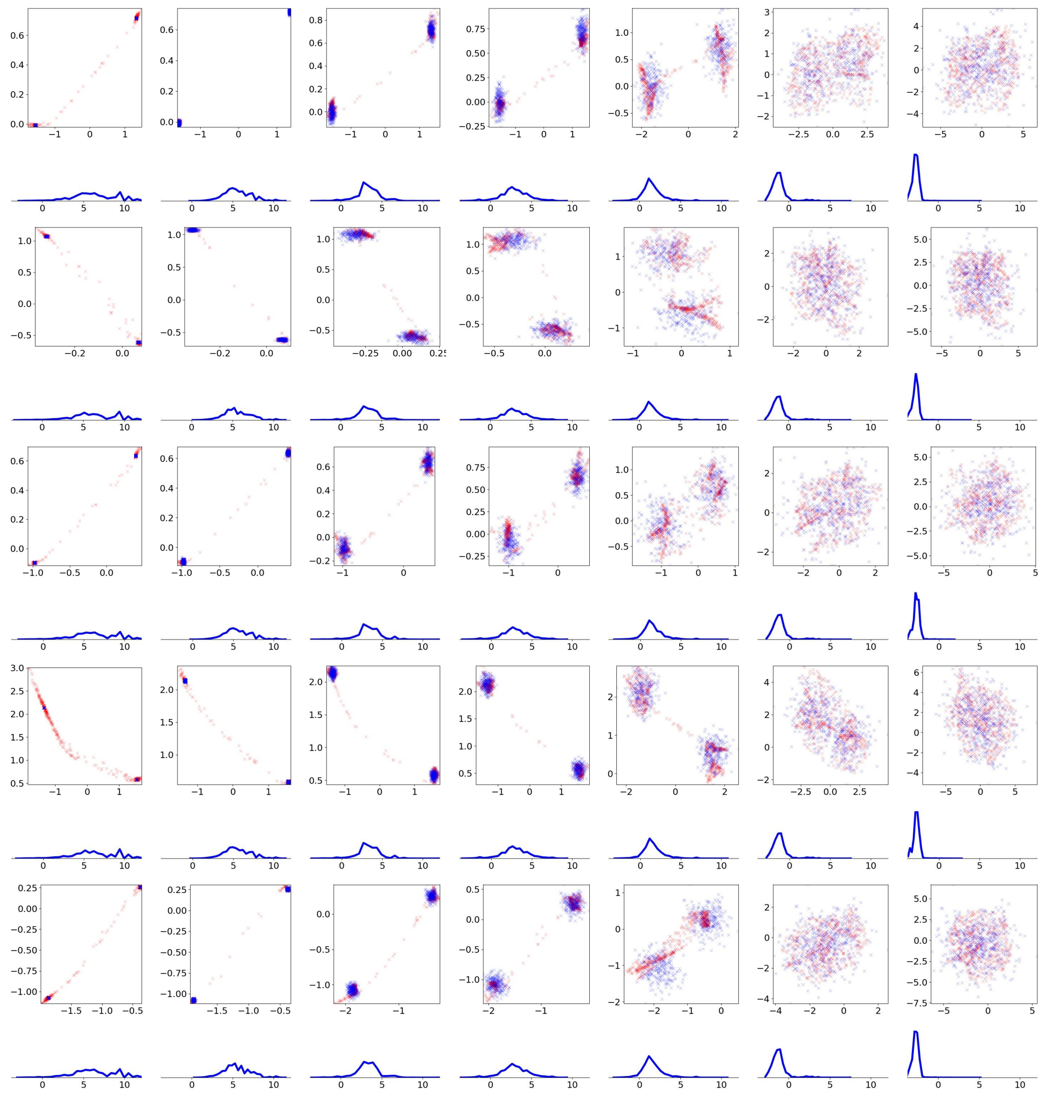


Figure 14. Reproduction of Fig. 8 for multiple standard deviations and multiple random seeds. Each column represent a different standard deviation of the two Gaussians $\sigma \in \{0.002, 0.01, 0.05, 0.1, 0.3, 1, 2\}$ and each row is a run with a different seed. As can be seen in all cases (except when lack of convergence) the distribution of the determinants support the claim and relate with the Entropy of the true distribution (blue points).

B. Architecture Details

We describe the used models below. The Dense(T) represents a fully connected layer with T units (activation function not included). The Conv2D(I, J, K) represent I filters of spatial shape (J, K) and the input dilation and padding follow the standard definition. For the VAE models the encoder is given below and for the GAN models the discriminator is given below as well. FC GAN model means that the FC generator is used in conjonction with the discriminator, the CONV GAN means that the CONV generator is used in conjonction with the discriminator and similarly for the VAE case.

FC generator	CONV generator	Encoder	Discriminator
Dense(256)	Dense(256)	Dense(512)	Dense(1024)
leaky ReLU	leaky ReLU	Dropout(0.3)	Dropout(0.3)
Dense(512)	Dense(8 * 6 * 6)	leaky ReLU	leaky ReLU
leaky ReLU	leaky ReLU	Dense(256)	Dense(512)
Dense(1024)	Reshape(8, 6, 6)	leaky ReLU	Dropout(0.3)
leaky ReLU	Conv2D(8, 3, 3, inputdilation=2, pad=same)	Dense(2*S)	leaky ReLU
Dense(28*28)	leaky ReLU		Dense(256)
	Conv2D(8, 4, 4, inputdilation=3, pad=valid)		Dropout(0.3)
	Reshape(28*28)		leaky ReLU
			Dense(2)

all the training procedures employ the Adam optimizer with a learning of 0.0001 which stays constant until training completion. In all cases trainig is done on 300 epochs, an epoch consisting of viewing the entire image training set once.

C. Proofs

C.1. Proof of Proposition 1

Proof. The result is a direct application of Corollary 3 in (Balestriero et al., 2019) adapted to GDNs (and not classification based DNs). The input regions are proven to be convex polytopes. Then by linearity of the per region mapping, conexity is preserved and with form given by (6). □

C.2. Proof of Lemma 1

Proof. First recall the standard result that

$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)),$$

for any matrix $A \in \mathbb{R}^{N \times K}$ and $B \in \mathbb{R}^{K \times D}$ (see for example (**?**) chapter 5). Now, noticing that $\min(\min(a, b), \min(c, d)) = \min(a, b, c, d)$ leads to the desired result by unrolling the product of matrices that make up the $\boldsymbol{A}_\omega$ matrix to obtain the desired result. □

C.3. Proof of Proposition 2

Proof. First notice that there can only be two major cases. First for the dimension of the affinely mapped region $\boldsymbol{G}(\omega)$ to be S or to be smaller than S. Let first consider the bijective case. For the GDN to be bijective on the region we need a one-to-one mapping from ω to $\boldsymbol{G}(\omega)$. If the dimension of the subsapce $\boldsymbol{G}(\omega)$ is S, then it means that the matrix $\boldsymbol{A}_\omega$ is full-rank, with rank S. In turn, this means that the columns of the matrix are linearly independent. This implies bijectivity on the region as each point in ω is mapped to an unique point in $\boldsymbol{G}(\omega)$ and vice-versa. The surjectivity is direct as if the dimension is smaller than S, then the matrix $\boldsymbol{A}_\omega$ is not full-rank and all the points in the region ω that leave in the kernel of the matrix (lifted with the bias $\boldsymbol{b}_\omega$) will be mapped to the same output points. This means that there exists different points in ω s.t. they are mapped to the same point in $\boldsymbol{G}(\omega)$ which gives surjectivity.

For global bijectivity, we need an additional condition. In fact, to ensure that the entire GDN preserves a one-to-one mapping, we need per region bijectivity coupled with the fact that the mapping for different region do not intersect. In fact, we know look at bijectivity between supp$(\boldsymbol{p}_{\boldsymbol{z}})$ and $\boldsymbol{G}(\text{supp}(\boldsymbol{p}_{\boldsymbol{z}}))$. Thus if the regions do not intersection after affine projections then there does not exist different latent vectors $\boldsymbol{z}$ and $\boldsymbol{z}'$ that would be mapped to the same output point. Yet because we have bijectivity on between ω and $\boldsymbol{G}(\omega), \forall \omega$ it means that each point in supp$(\boldsymbol{p}_{\boldsymbol{z}})$ is mapped to an unique point in $\boldsymbol{G}(\text{supp}(\boldsymbol{p}_{\boldsymbol{z}}))$ which gives global bijectivity.

□

C.4. Proof of Proposition 3

Proof. The first bound is obtained by taking the realization of the noise where $\boldsymbol{r} = \mathbf{0}$, in that case the input space partition is the entire space as any input is mapped to the same VQ code. As such, the mapping associated to this trivial partition has $\mathbf{0}$ slope (matrix filled with zeros) and a possibly nonzeros bias; as such the mapping is zero-dimensional (any points is mapped to the same point). This gives the lower bound stating that in the mixture of GDNs, one will have dimension 0. For the other case, simply take the trivial case of $\boldsymbol{r} = \mathbf{1}$ which gives the result. □

C.5. Proof of Lemma 3

Proof. First, as we impose injectivity, we can not have multiple regions of the input space, say ω and ω' such that $\boldsymbol{G}(\omega) \cap \boldsymbol{G}(\omega') \neq \emptyset$. Second, a region of the input space is mapped to another region in the output space by means of the affine transformation, thus even though the ambiant space D might be of greater dimension that $\dim(\boldsymbol{G})$, the injectivity implies that points in ω are mapped to at most one point in $\boldsymbol{G}(\omega)$. They are affinely mapped meaning that the inverse is given by removing the bias and inverting the linear mapping which is given by the pseudo inverse. Recalling the above result on surjectivity, we see that for the GDN to be injective the per region dimension msut be S showing existence of the pseudo inverse. □

C.6. Proof of Proposition 4

Proof. The proof is straightforward from the used definition of disententglement. In fact, recall that we aim to have $\langle \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{G}(\boldsymbol{z} + \epsilon\delta_d), \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{G}(\boldsymbol{z} + \epsilon\delta_{d'})\rangle \approx 0$. In our case, consider only small transformation such that $\boldsymbol{z} + \epsilon\delta_d$ and $\boldsymbol{z} + \epsilon\delta_{d'}$ remain the in the same region ω in which was $\boldsymbol{z}$. Then it is clear that for any positive constant ϵ fulfilling this condition, the above disentangelement definition translates into

$$\langle \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{G}(\boldsymbol{z} + \epsilon\delta_d), \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{G}(\boldsymbol{z} + \epsilon\delta_{d'})\rangle \approx 0 \iff \langle [\boldsymbol{A}_\omega]_{.,d}, [\boldsymbol{A}_\omega]_{.,d'}\rangle \approx 0,$$

This gives a necessary condition which is not sufficient as this alone does not guarantee that each dimension of the latent space only impacts a single transformation of the output. But a disentangled representation must have near orthogonal columns for the slope matrices $\boldsymbol{A}_\omega$. □

C.7. Proof of Theorem 2

Proof. First, notice that $P(\boldsymbol{A}_\omega) = \boldsymbol{A}_\omega(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-1}\boldsymbol{A}_\omega^T$ defines a projection matrix. In fact, we have that

$$\begin{aligned} P(\boldsymbol{A}_\omega)^2 &= \boldsymbol{A}_\omega(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-1}\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-1}\boldsymbol{A}_\omega^T \\ &= \boldsymbol{A}_\omega(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-1}\boldsymbol{A}_\omega^T \\ &= P(\boldsymbol{A}_\omega) \end{aligned}$$

and we have that $(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-1}$ is well defined as we assume injectivity (rank$(\boldsymbol{A}_\omega) = S$) making the $S \times S$ matrix $\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega$ full rank. Now it is clear that this projection matrix maps an arbitrary point $\boldsymbol{x} \in \mathbb{R}^D$ to the affine subspace $\boldsymbol{G}(\omega)$ up to the bias shift. As we are interested in the angle between two adjacent subspaces $\boldsymbol{G}(\omega)$ and $\boldsymbol{G}(\omega')$ it is also clear that the biases (which do not change the angle) can be omited. Hence the task simplifies to finding the angle between $P(\boldsymbol{A}_\omega)$ and $P(\boldsymbol{A}_{\omega'})$. This can be done by means of the greatest principal angle (proof can be found in Stewart (1973)) with the result being $\sin\big(\theta(\boldsymbol{G}(\omega), \boldsymbol{G}(\omega'))\big) = \|P(\boldsymbol{A}_\omega) - P(\boldsymbol{A}_{\omega'})\|_2$ as desired. □

C.8. Proof of Lemma 4

Proof. In the special case of an affine transform of the coordinate given by the matrix $A \in \mathbb{R}^{D \times D}$ the well known result from demonstrates that the change of volume is given by $|\det(A)|$ (see Theorem 7.26 in (?)). However in our case the mapping is a rectangular matrix as we span an affine subspace in the ambiant space $\mathbb{R}^D$ making $|\det(A)|$ not defined. However by applying Sard's theorem (?) we obtain that the change of volume from the region ω to the affine subspace $\boldsymbol{G}(\omega)$ is given by $\sqrt{\det(A^TA)}$ which can also be written as follows with USV^T the svd-decomposition of the matrix A:

$$\sqrt{\det(A^TA)} = \sqrt{\det((USV^T)^T(USV^T))} = \sqrt{\det((VS^TU^T)(USV^T))}$$

$$
\begin{aligned}
&= \sqrt{\det(VS^TSV^T)} \\
&= \sqrt{\det(S^TS)} \\
&= \prod_{i:\sigma_i \neq 0} \sigma_i(A)
\end{aligned}
$$

□

C.9. Proof of Theorem 3

Proof. We will be doing the change of variables $\boldsymbol{z} = (\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)^{-1} \boldsymbol{A}_\omega^T (\boldsymbol{x} - \boldsymbol{b}_\omega) = \boldsymbol{A}_\omega^+ (\boldsymbol{x} - \boldsymbol{b}_\omega)$, also notice that $J_{\boldsymbol{G}^{-1}}(\boldsymbol{x}) = A^+$. First, we know that $P_{\boldsymbol{G}(\boldsymbol{z})}(\boldsymbol{x} \in w) = P_{\boldsymbol{z}}(\boldsymbol{z} \in \boldsymbol{G}^{-1}(w)) = \int_{\boldsymbol{G}^{-1}(w)} p_{\boldsymbol{z}}(\boldsymbol{z}) d\boldsymbol{z}$ which is well defined based on our full rank assumptions. We then proceed by

$$
\begin{aligned}
P_{\boldsymbol{G}}(\boldsymbol{x} \in w) &= \sum_{\omega \in \Omega} \int_{\omega \cap w} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x})) \sqrt{\det(J_{\boldsymbol{G}^{-1}}(\boldsymbol{x})^T J_{\boldsymbol{G}^{-1}}(\boldsymbol{x}))} d\boldsymbol{x} \\
&= \sum_{\omega \in \Omega} \int_{\omega \cap w} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x})) \sqrt{\det((\boldsymbol{A}_\omega^+)^T \boldsymbol{A}_\omega^+)} d\boldsymbol{x} \\
&= \sum_{\omega \in \Omega} \int_{\omega \cap w} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x})) (\prod_{i:\sigma_i(\boldsymbol{A}_\omega^+) > 0} \sigma_i(\boldsymbol{A}_\omega^+)) d\boldsymbol{x} \\
&= \sum_{\omega \in \Omega} \int_{\omega \cap w} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x})) (\prod_{i:\sigma_i(\boldsymbol{A}_\omega) > 0} \sigma_i(\boldsymbol{A}_\omega))^{-1} d\boldsymbol{x} \quad \text{Etape 1} \\
&= \sum_{\omega \in \Omega} \int_{\omega \cap w} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x})) \frac{1}{\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)}} d\boldsymbol{x}
\end{aligned}
$$

Let now prove the Etape 1 step by proving that $\sigma_i(A^+) = (\sigma_i(A))^{-1}$ where we lighten notations as $A := \boldsymbol{A}_\omega$ and USV^T is the svd-decomposition of A:

$$
\begin{aligned}
A^+ = (A^TA)^{-1}A^T &= ((USV^T)^T(USV^T))^{-1}(USV^T)^T \\
&= (VS^TU^TUSV^T)^{-1}(USV^T)^T \\
&= (VS^TSV^T)^{-1}VS^TU^T \\
&= V(S^TS)^{-1}S^TU^T \\
&\implies \sigma_i(A^+) = (\sigma_i(A))^{-1}
\end{aligned}
$$

with the above it is direct to see that $\sqrt{\det((\boldsymbol{A}_\omega^+)^T \boldsymbol{A}_\omega^+)} = \frac{1}{\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)}}$ as follows

$$
\begin{aligned}
\sqrt{\det((\boldsymbol{A}_\omega^+)^T \boldsymbol{A}_\omega^+)} = \prod_{i:\sigma_i \neq 0} \sigma_i(\boldsymbol{A}_\omega^+) = \prod_{i:\sigma_i \neq 0} \sigma_i(\boldsymbol{A}_\omega)^{-1} \\
= \left(\prod_{i:\sigma_i \neq 0} \sigma_i(\boldsymbol{A}_\omega) \right)^{-1} \\
= \frac{1}{\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)}}
\end{aligned}
$$

which gives the desired result. □

C.10. Proof of Cor. 1

Proof. The derivation of the Entropy will consist in rewritting the Entropy w.r.t the distribution in the output space of the GDN and performing the change of coordinates leveragign the abvoe result to finally obtain the desired result as follows:

$$\begin{aligned}
E(\boldsymbol{p_G}) =& -\sum_{\omega\in\Omega}\int_{\boldsymbol{G}(\omega)} \boldsymbol{p_G}(\boldsymbol{x})\log(\boldsymbol{p_G}(\boldsymbol{x}))d\boldsymbol{x}\\
=& -\sum_{\omega\in\Omega}\int_{\boldsymbol{G}(\omega)} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\log\left(p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\right)\\
=& -\sum_{\omega\in\Omega}\int_{\boldsymbol{G}(\omega)} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\left(\log\left(p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\right)+\log\left(\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\right)\right)\\
=& -\sum_{\omega\in\Omega}\int_{\boldsymbol{G}(\omega)} \det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\log\left(p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\right)\\
&-\sum_{\omega\in\Omega}\int_{\boldsymbol{G}(\omega)} \det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\log\left(\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\right)\\
=&E(\boldsymbol{p_z}) \qquad \text{(apply the change of coordinate } \boldsymbol{z}=G(\boldsymbol{x}))\\
&-\sum_{\omega\in\Omega}\int_{\boldsymbol{G}(\omega)} p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\log\left(\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}\right)\\
=&E(\boldsymbol{p_z})+\frac{1}{2}\sum_{\omega\in\Omega}P(\boldsymbol{z}\in\omega)\log\left(\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)\right) \qquad \text{(apply the change of coordinate } \boldsymbol{z}=G(\boldsymbol{x}))
\end{aligned}$$

which gives the desired result. For a complete review of integrals on manifold please see (Cover & Thomas, 2012). □

C.11. Proof of Theorem 1

Proof. For very small N it is clear than in general, even if $S < S^*$, the memorization capacity of the generator will be s.t. it can fit through those points. Just imagine a couple of points sampled from a $2D$ linear manifold, even though $S = 1$, the GDN can go through those two points and thus have $E^* = 0$ for $N = 2$. We now consider the case where N is large enough. Two cases have to be studied.

- Case $S < S^*$: if $S < S^*$, the generated manifold can never be dense in the true linear manifold. This means that the newly introduced point will almost surely not lie in the span of the current generated manifold. Thus, $E^*(N + 1) > E^*(N)$.
- Case $S \geq S^*$: in that case, it is clear the there always exist a setting of the parameters Θ s.t. the DGN spans the linear manifold. For example if using ReLU, consider any weights for the first $L - 1$ layers s.t. the ReLU ae always “on” and use the last layer affine mapping to rotate and translate the affine subspace to the true one. That is, $E^*(N) = 0, \forall N > 0$.

The above demonstrates how for the simplest target manifold (linear) an too narrow DN leading to $S < S^*$ will haev a training error E^* increasing with N or 0 if $S \geq S^*$ for any N. □

C.12. Proof of Corollary 2

Proof. First, by applying the above results on the general density formula and setting $\boldsymbol{p_z}$ a standard Normal distribution we obtain that

$$\begin{aligned}
\boldsymbol{p_G}(\boldsymbol{x}\in w) =& \sum_{\omega\in\Omega}\int_{\omega\cap w}\mathbb{1}_{\boldsymbol{x}\in\boldsymbol{G}(\omega)}p_{\boldsymbol{z}}(\boldsymbol{G}^{-1}(\boldsymbol{x}))\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-\frac{1}{2}}d\boldsymbol{x}\\
=& \sum_{\omega\in\Omega}\int_{\omega\cap w}\mathbb{1}_{\boldsymbol{x}\in\boldsymbol{G}(\omega)}\frac{1}{(2\pi)^{S/2}\sqrt{\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)}}e^{-\frac{1}{2}\|\boldsymbol{G}^{-1}(\boldsymbol{x})\|_2^2}d\boldsymbol{x}\\
=& \sum_{\omega\in\Omega}\int_{\omega\cap w}\mathbb{1}_{\boldsymbol{x}\in\boldsymbol{G}(\omega)}\frac{1}{(2\pi)^{S/2}\sqrt{\det(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)}}e^{-\frac{1}{2}((\boldsymbol{A}_\omega^+(\boldsymbol{x}-\boldsymbol{b}_\omega))^T((\boldsymbol{A}^+(\boldsymbol{x}-\boldsymbol{b}_\omega))}d\boldsymbol{x}
\end{aligned}$$

$$= \sum_{\omega \in \Omega} \int_{\omega \cap w} \mathbb{1}_{\boldsymbol{x} \in \boldsymbol{G}(\omega)} \frac{1}{(2\pi)^{S/2}\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{b}_\omega)^T (\boldsymbol{A}_\omega^+)^T \boldsymbol{A}_\omega^+ (\boldsymbol{x}-\boldsymbol{b}_\omega)} d\boldsymbol{x}$$

giving the desired result. □

D. Codes of neighbour regions

Each code is equivalent to a system of inequalities that define the regions. In fact, a code depends on the signs of the feature map pre activation (recall (3)). This defines a polytope in the input space and also in the output space. Now, when traveling from a point $\boldsymbol{z}$ to another point $\boldsymbol{z}'$ of a neighbouring region (recall Def. 2), we ask the question on how many indices of the code will change. That is, what is the Hamming distance between $\boldsymbol{q}(\boldsymbol{z})$ and $\boldsymbol{q}(\boldsymbol{z}')$. As a neighbouring region is defined as a region which shares some of its boundary with another (their interior is disjoint) we can see that the degree of the face that is shared between the two regions define the amount of changes in their corresponding codes. If two regions share a $S-1$ dimensional face, then only 1 value of the code changes. If they share in general a $S-r$ dimensional face, then the code will change by r values. As most adjacent regions will share a high dimensional face, we see that r tends to be small and thus codes are similar. For details and analytical study of the above please see (**?**).

E. More on Disentangled Latent Representations

It has been coined that providing interpretable and practical generators lies in the ability to learn a *disentangled representation* of an input $\boldsymbol{x} = \boldsymbol{G}(\boldsymbol{z})$ (Schmidhuber, 1992; Bengio et al., 2013). The code $\boldsymbol{z}$ should contain all the information present in $\boldsymbol{x}$ in a compact and interpretable structure where distinct, informative factors of variations are encoded by different dimensions. Such motivations orginitated from the (non-)linear independent component analysis focusing on recovering independent factors from observed data (Comon, 1994; Hyvarinen & Morioka, 2016). In fact, even in recent GAN/VAE based models, disentengled representations are associated to independent transformations of the input such as pose, hair color, eye color and so on which should behave independently form each other (**??**). For a more in-depth review of learning disentangled representation, see (Locatello et al., 2018).

F. Details on Training Procedure

The experiment aims at depicting the training error being E^* on the training set for varying latent dimensions S in the simple case of a linear true data manifold approximation. In order to prevent any optimization unlucky degeneracy we repeat the training procedure 30 times and compute for each poch the error E^* and report the minimum over the 30 trials and training epochs. We also set a very large number of epochs: 2000. Due to the large number of trials and epochs the reported results are not due to some random initialization settings and convey the point of the result which is that even for such a simple data model (linear manifold) if $S < S^*$ then the training error E^* will increase with N. Finally, the minimization over z is replacer by an autoencoder with a very wide encoder s.t. it has the capacity for each training point to memorize the optimum z that minimizes E. That is, when minimizing

$$\min_{\Theta} \min_{\Theta'} \|\boldsymbol{G}_\Theta(E_{\Theta'}(\boldsymbol{x})) - \boldsymbol{x}\| \approx \min_{\Theta} \min_{\boldsymbol{z}} \|\boldsymbol{G}_\Theta(\boldsymbol{z}) - \boldsymbol{x}\|,$$

got a large enough encoder network E. In our case given that we used $S^* = 6$ we used an encoder with $D_\ell = 256$ units and $L = 3$.

G. More on Effect of Dropout/Dropconnect

As opposed to the Dropout case which applies the binary noise ont the feature maps $\boldsymbol{v}_\ell$, Dropconnect (Wan et al., 2013) applies this binary noise onto the slope matrices $\boldsymbol{W}_\ell$ making the mapping noise become

$$\boldsymbol{G}(\boldsymbol{z}) = \left(\prod_{\ell=L}^{1} \text{diag}(\boldsymbol{q}_\ell)(\boldsymbol{W}_\ell \odot \boldsymbol{R}_\ell)\right)\boldsymbol{z} + \sum_{\ell=1}^{L}\left(\prod_{\ell'=L}^{\ell+1} \text{diag}(\boldsymbol{q}_{\ell'})(\boldsymbol{W}_{\ell'} \odot \boldsymbol{R}_{\ell'})\right)\boldsymbol{b}_\ell,$$

where the binary noise matrices are denoted by $\boldsymbol{R}_\ell$. Despite this change of nosei application, the exact same result applies and Prop. 3 also holds. That is the dropconnect equipped GDN becomes a mixture of GDNs with varying dimensions and parameters. Notice however that dropconnect will be less likely to reduce the ablated generator dimension as opposed to dropout due to its application on each entry of the weight matrix as opposed to an entire row at a time as depicted in Fig. 2.

H. Acknowledgments

This work was supported by NSF grants CCF-1911094, IIS-1838177, and IIS-1730574; ONR grants N00014-18-12571 and N00014-17-1-2551; AFOSR grant FA9550-18-1-0478; DARPA grant G001534-7500; and a Vannevar Bush Faculty Fellowship, ONR grant N00014-18-1-2047, the Ken Kennedy Institute 2019/20 BP Graduate Fellowship.

References

Absil, P.-A., Edelman, A., and Koev, P. On the largest principal angle between random subspaces. *Linear Algebra and its applications*, 414(1):288–294, 2006.

Afriat, S. N. Orthogonal and oblique projectors and the characteristics of pairs of vector spaces. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 53, pp. 800–816. Cambridge University Press, 1957.

Andrs-Terr, H. and Li, P. Perturbation theory approach to study the latent space degeneracy of variational autoencoders, 2019.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. arxiv, 2017. *arXiv preprint arXiv:1701.04862*.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Bakır, G. H., Weston, J., and Schölkopf, B. Learning to find pre-images. *Advances in neural information processing systems*, 16:449–456, 2004.

Balestriero, R. and Baraniuk, R. Mad max: Affine spline insights into deep learning. *arXiv preprint arXiv:1805.06576*, 2018a.

Balestriero, R. and Baraniuk, R. G. A spline theory of deep networks. In *Proc. Int. Conf. Mach. Learn.*, volume 80, pp. 374–383, Jul. 2018b.

Balestriero, R., Cosentino, R., Aazhang, B., and Baraniuk, R. The geometry of deep networks: Power diagram subdivision. In *Advances in Neural Information Processing Systems 32*, pp. 15806–15815. 2019.

Ben-Yosef, M. and Weinshall, D. Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. *arXiv preprint arXiv:1808.10356*, 2018.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Biau, G., Cadre, B., Sangnier, M., and Tanielian, U. Some theoretical properties of gans. *arXiv preprint arXiv:1803.07819*, 2018.

Bjorck, A. and Golub, G. H. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973.

Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

Breiman, L., Meisel, W., and Purcell, E. Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144, 1977.

Bryant, F. B. and Yarnold, P. R. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

Chongxuan, L., Xu, T., Zhu, J., and Zhang, B. Triple generative adversarial nets. In *Advances in neural information processing systems*, pp. 4088–4098, 2017.

Comon, P. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

Cover, T. M. and Thomas, J. A. *Elements of information theory*. John Wiley & Sons, 2012.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

Dieng, A. B., Ruiz, F. J. R., Blei, D. M., and Titsias, M. K. Prescribed generative adversarial networks, 2019.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Dinh, L., Sohl-Dickstein, J., Pascanu, R., and Larochelle, H. A rad approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

Donoho, D. L., Johnstone, I. M., et al. Ideal denoising in an orthonormal basis chosen from a library of bases. *Comptes rendus de l'Académie des sciences. Série I, Mathématique*, 319(12):1317–1322, 1994.

Durugkar, I., Gemp, I., and Mahadevan, S. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

Fabius, O. and van Amersfoort, J. R. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Gan, Z., Chen, L., Wang, W., Pu, Y., Zhang, Y., Liu, H., Li, C., and Carin, L. Triangle generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 5247–5256, 2017.

Ghosh, A., Kulharia, V., Namboodiri, V. P., Torr, P. H., and Dokania, P. K. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8513–8521, 2018.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*, volume 1. MIT Press, 2016. http://www.deeplearningbook.org.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pp. 2672–2680. MIT Press, 2014.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Hannah, L. A. and Dunson, D. B. Multivariate convex regression with adaptive partitioning. *J. Mach. Learn. Res.*, 14(1): 3261–3294, 2013.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

Hyvarinen, A. and Morioka, H. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, pp. 3765–3773, 2016.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Kim, H. and Mnih, A. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Li, J., Madry, A., Peebles, J., and Schmidt, L. Towards understanding the dynamics of generative adversarial networks. *arXiv preprint arXiv:1706.09884*, 2017.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8157–8166, 2018.

Liu, S., Bousquet, O., and Chaudhuri, K. Approximation and convergence properties of generative adversarial learning. In *Advances in Neural Information Processing Systems*, pp. 5545–5553, 2017.

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359*, 2018.

Magnani, A. and Boyd, S. P. Convex piecewise-linear fitting. *Optim. Eng.*, 10(1):1–17, 2009.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Munkres, J. *Topology*. Pearson Education, 2014.

Qi, L. and Sun, D. Nonsmooth equations and smoothing newton methods. *Applied Mathematics Report AMR*, 98(10), 1998.

Qi, L. and Sun, J. A nonsmooth version of newton's method. *Mathematical programming*, 58(1-3):353–367, 1993.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Rosenblatt, M. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pp. 832–837, 1956.

Roth, K., Lucchi, A., Nowozin, S., and Hofmann, T. Stabilizing training of generative adversarial networks through regularization. In *Advances in neural information processing systems*, pp. 2018–2028, 2017.

Roy, A., Vaswani, A., Neelakantan, A., and Parmar, N. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*, 2018.

Schmidhuber, J. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

Stewart, G. W. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM review*, 15 (4):727–764, 1973.

Streubel, T., Griewank, A., Radons, M., and Bernt, J.-U. Representation and analysis of piecewise linear functions in abs-normal form. In *IFIP Conference on System Modeling and Optimization*, pp. 327–336. Springer, 2013.

Tomczak, J. M. and Welling, M. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

Tomczak, J. M. and Welling, M. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pp. 351–359, 2013.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066, 2013.

Yang, D., Hong, S., Jang, Y., Zhao, T., and Lee, H. Diversity-sensitive conditional generative adversarial networks, 2019.

Zhang, P., Liu, Q., Zhou, D., Xu, T., and He, X. On the discrimination-generalization tradeoff in gans. *arXiv preprint arXiv:1711.02771*, 2017.

Zhao, J., Mathieu, M., and LeCun, Y. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.