MAX-AFFINE SPLINE INSIGHTS INTO DEEP GENERATIVE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We connect a large class of Deep Generative Networks (DGNs) with spline operators in order to analyze their properties and limitations and suggest new opportunities. By characterizing the latent space partition and per-region affine mappings of a DGN, we relate the manifold dimension and approximation error to the sample size, provide a theoretically motivated "elbow method" to infer the intrinsic dimensionality of the manifold being approximated, study the (not always) beneficial impact of dropout/dropconnect, and demonstrate how a DGN's piecewise affine generated manifold can be "smoothed" to facilitate latent space optimization as in inverse problems. The per-region affine subspace defines a local coordinate basis; we provide necessary and sufficient conditions relating those basis vectors with disentanglement and demonstrate how to interpret the DGN's learned parameters. We also derive the output probability density that is 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 enables us to highlight the source of training instabilities when the target density is multimodal: as the modes move further apart and/or are more concentrated, the singular values of the approximant DGN per-region mappings increase, leading to overly large layer weights and causing 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 a DGN's generated samples (Sec. 3.1), including its *intrinsic dimension*, which sheds new light on the impact of techniques like dropout (Sec. 3.2) and provides practitioners with sensible design principles for constructing a desired manifold (Sec. 3.3).

[C2] We characterize the *local coordinates of a DGN's generated manifold* and the inverse mapping from the generated data points $\boldsymbol{x}$ back to the latent variables $\boldsymbol{z}$. This sheds new insights on the DGN architectures and suggests a novel discrete optimization problem for DGN inversion and inverse problem applications (Sec. 4.1). Armed with this result, we derive necessary conditions on the DGN per-region affine mappings for *disentanglement*, to *interpret* the disentangled mappings, and to draw new links between DGNs and *adaptive basis methods* (Sec. 4.2). We also demonstrate how to *smooth* a DGN's generated continuous piecewise affine manifold in a principled way, which enables new approaches for (smooth) manifold approximation (Sec. 4.3).

[C3] We provide an input-output formula for a DGN 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 the analytical form of this density for several interesting settings and demonstrate that there exists a crucial link between the DGN partition, the singular values of the per-region affine mappings, and the Shannon entropy of the output density. This provides a new lens through which to study the difficulty of generating multidimensional, low-entropy distributions using DGNs (Section 5.2).

Reproducible code for the various experiments and figures will be provided on Github upon completion of the review process. Proofs of our results are provided in Appendix F.

2 Background

Deep Generative Networks. A deep generative network (DGN) is an operator $\boldsymbol{G}_\Theta$ with parameters Θ mapping a *latent* input $\boldsymbol{z} \in \mathbb{R}^S$ to an *observation* $\boldsymbol{x} \in \mathbb{R}^D$ by composing L intermediate *layer* mappings g_ℓ, $\ell = 1, \ldots, L$. We precisely define a layer g^ℓ as comprising a single nonlinear operator composed with any (if any) preceding linear operators that lie between it and the preceding nonlinear operator. Examples of linear operators are the *fully connected operator* (affine transformation with weight matrix $\boldsymbol{W}^\ell$ and bias vector $\boldsymbol{b}^\ell$), or the *convolution operator* (circulent $\boldsymbol{W}^\ell$ matrix); examples of nonlinear operators are the *activation operator* (applying a scalar nonlinearity such as the ubiquitous ReLU), or the *pooling operator* (dimensionality reduction given a maximum operator). Precise definitions of these operators can be found in Goodfellow et al. (2016). We will omit Θ from the f_Θ operator for conciseness unless needed.

Each layer g^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 in particular $\boldsymbol{v}^0 := \boldsymbol{z}$, $D^0 = S$, and $\boldsymbol{v}_L := \boldsymbol{x}, D_L = D$. In this paper, we focus on DGNs, where in general $S \leq D$ and in some cases $S \ll D$. In such framework $\boldsymbol{z}$ is interpreted as a latent representation, and $\boldsymbol{x}$ is the generated/observed 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 *max-affine spline operator* (MASO) concatenates 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). For our purpose each MASO will express a DN layer and is thus an operator producing a D^ℓ dimensional vector from a $D^{\ell-1}$ dimensional vector and is formally given by

$$\text{MASO}(\boldsymbol{v}; \{\boldsymbol{A}_r, \boldsymbol{b}_r\}_{r=1}^R) = \max_{r=1,\ldots,R} \boldsymbol{A}_r \boldsymbol{v} + \boldsymbol{b}_r, \tag{1}$$

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 and the maximum is taken coordinate-wise. 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 corresponds to a (single) MASO with $R = 2, \boldsymbol{A}_1 = \boldsymbol{W}^\ell, \boldsymbol{A}_2 = \boldsymbol{0}, \boldsymbol{b}_1 = \boldsymbol{b}^\ell, \boldsymbol{b}_2 = \boldsymbol{0}$. Note that a MASO is a *continuous piecewise-affine* (CPA) operator Wang & Sun (2005).

The key background result for this paper is that *the layers of DNs constructed from piecewise affine operators (e.g., convolution, ReLU, and max-pooling) are MASOs* Balestriero & Baraniuk (2018b;a):

$$\exists R \in \mathbb{N}^*, \exists \{\boldsymbol{A}_r, \boldsymbol{b}_r\}_{r=1}^R \text{ s.t. } \text{MASO}(\boldsymbol{v}; \{\boldsymbol{A}_r, \boldsymbol{b}_r\}_{r=1}^R) = f_\ell(\boldsymbol{v}), \forall \boldsymbol{v} \in \mathbb{R}^{D^{\ell-1}}, \tag{2}$$

making the entire DGN a composition of MASOs.

The CPA spline interpretation enabled from a MASO formulation of DGNs provides a powerful global geometric interpretation of the network mapping f based on a partition of its input space $\mathbb{R}^S$ into polyhedral regions and a per-region affine transformation producing the network output. 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). We now propose to greatly extend such CPA spline driven insights to carefully characterize and understand DGNs as well as providing theoretical justifications to various observed behaviors such as mode collapse.

3 Importance of the Deep Generative Network Dimension

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

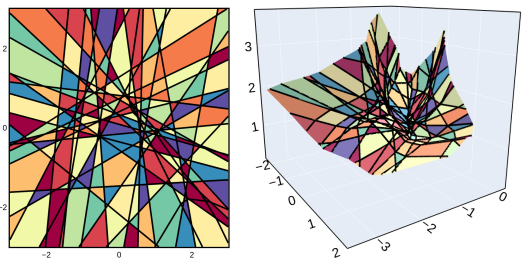


Figure 1: Visual depiction of Thm. 1 with a (random) generator $\boldsymbol{G} : \mathbb{R}^2 \mapsto \mathbb{R}^3$. **Left:** generator input space partition Ω made of polytopal regions. **Right:** generator image $Im(\boldsymbol{G})$ which is a continuous piecewise affine surface composed of the polytopes obtained by affinely transforming the polytopes from the input space partition (left) the colors are per-region and correspond between left and right plots. *This input-space-partition / generator-image / per-region-affine-mapping relation holds for any architecture employing piecewise affine activation functions. Understanding each of the three brings insights into the others, as we demonstrate in this paper.*

3.1 Input-Output Space Partition and Per-Region Mapping

As was hint in the previous section, the MASO formulation of a DGN allows to express the (entire) DGN mapping $\boldsymbol{G}$ (a composition of L MASOs) as a per-region affine mapping

$$\boldsymbol{G}(\boldsymbol{z}) = \sum_{\omega \in \Omega} \left(\boldsymbol{A}_\omega \boldsymbol{z} + \boldsymbol{b}_\omega\right) \mathbb{1}_{\boldsymbol{z} \in \omega}, \qquad \boldsymbol{z} \in \mathbb{R}^S, \tag{3}$$

with $\cup_{\omega \in \Omega} \omega = \mathbb{R}^S$ and $\forall (\omega, \omega') \in \Omega^2, \omega \neq \omega', \omega^\circ \cap \omega'^\circ = \emptyset$, with $(\cdot)^\circ$ the interior operator Munkres (2014), and $\boldsymbol{A}_\omega \in \mathbb{R}^{D \times S}, \boldsymbol{b}_\omega \in \mathbb{R}^D$ the per-region affine parameters. The above comes naturally from the composition of linear operators interleaved with continuous piecewise affine which are current activation functions (ReLU Glorot et al. (2011), leaky-ReLU Maas et al. (2013), absolute value Bruna & Mallat (2013), parametric ReLU He et al. (2015) and the likes) and/or max-pooling. For a dedicated study of the partition Ω we refer the reader to Zhang et al. (2018); Balestriero et al. (2019).

In order to study and characterize the DGN mapping (3), we make explicit the formation of the per-region slope and bias parameters. The affine parameters $\boldsymbol{A}_\omega, \boldsymbol{b}_\omega$ can be obtained efficiently by using an input $\boldsymbol{z} \in \omega$ and the Jacobian (J) operator, leading to

$$\boldsymbol{A}_\omega = (J_{\boldsymbol{z}} \boldsymbol{G})(\boldsymbol{z}) = \left(\prod_{i=0}^{L-1} \text{diag}\left(\dot{\sigma}^{L-i}(\omega)\right) \boldsymbol{W}^{L-i} \right) = \text{diag}\left(\dot{\sigma}^L(\boldsymbol{z})\right) \boldsymbol{W}^L \ldots \text{diag}\left(\dot{\sigma}^1(\boldsymbol{z})\right) \boldsymbol{W}^1, \tag{4}$$

where $\dot{\sigma}^\ell(\boldsymbol{z})$ is the pointwise derivative of the activation function of layer ℓ based on its input $\boldsymbol{W}^\ell \boldsymbol{z}^{\ell-1} + \boldsymbol{b}^\ell$, which we note as a function of $\boldsymbol{z}$ directly. The diag operator simply puts the given vector into a diagonal square matrix. For convolutional layers (or else) one can simply replace the corresponding $\boldsymbol{W}^\ell$ with the correct slope matrix parametrization as discussed in Sec. 2. Notice that since the employed activation functions $\sigma^\ell, \forall \ell \in \{1, \ldots, L\}$ are piecewise affine, their derivative is piecewise constant, in particular with values $[\dot{\sigma}^\ell(\boldsymbol{z})]_k \in \{\alpha, 1\}$ with $\alpha = 0$ for ReLU, $\alpha = -1$ for absolute value, and in general with $\alpha > 0$ for Leaky-ReLU for $k \in \{1, \ldots, D^\ell\}$. We denote the collection of all the per-layer activation derivatives $[\dot{\sigma}^1(\boldsymbol{z}), \ldots, \dot{\sigma}^L(\boldsymbol{z})]$ as the *activation pattern* of the generator. Based on the above, if one already knows the associated activation pattern of a region ω, then the matrix $\boldsymbol{A}_\omega$ can be formed directly by using them without having access to an input $\boldsymbol{z} \in \omega$. In this case, we will slightly abuse notation and denote those known activation patterns as $\dot{\sigma}^\ell(\omega) \triangleq \dot{\sigma}^\ell(\boldsymbol{z}), \boldsymbol{z} \in \omega$ with ω being the considered region. In a similar way, the bias vector is obtained as

$$\boldsymbol{b}_\omega = \boldsymbol{G}(\boldsymbol{z}) - \boldsymbol{A}_\omega \boldsymbol{z} = \boldsymbol{b}^L + \sum_{\ell=1}^{L-1} \left[\left(\prod_{i=0}^{L-\ell-1} \text{diag}\left(\dot{\sigma}^{L-i}(\omega)\right) \boldsymbol{W}^{L-i} \right) \text{diag}\left(\dot{\sigma}^\ell(\omega)\right) \boldsymbol{b}^\ell \right].$$

The $\boldsymbol{b}_\omega$ vector can be obtained either from a sample $\boldsymbol{z} \in \omega$ using the left part of the above equation or based on the known region activation pattern using the right hand side of the equation.

Theorem 1. *The image of a generator $\boldsymbol{G}$ employing MASO layers is made of connected polytopes corresponding to the polytopes of the input space partition Ω, each being affinely transformed by the per-region affine parameters as in*

$$Im(\boldsymbol{G}) \triangleq \{\boldsymbol{G}(\boldsymbol{z}) : \boldsymbol{z} \in \mathbb{R}^S\} = \bigcup_{\omega \in \Omega} \mathit{Aff}(\omega; \boldsymbol{A}_\omega, \boldsymbol{b}_\omega) \tag{5}$$

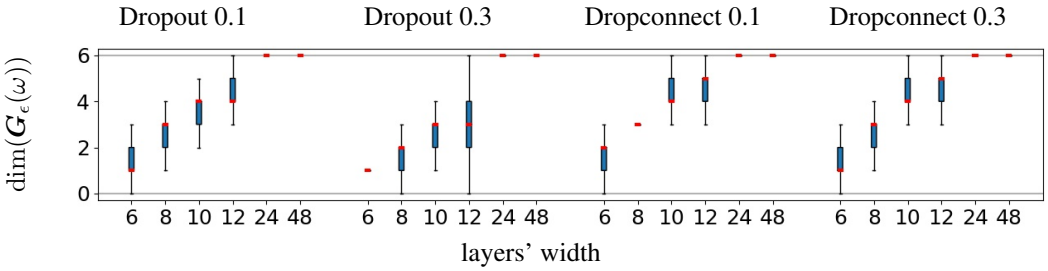


Figure 2: Impact of dropout and dropconnect on the intrinsic dimension of the noise induced generators for two "drop" probabilities 0.1 and 0.3 and for a generator $\boldsymbol{G}$ with $S = 6$, $D = 10$, $L = 3$ with varying width $D^1 = D^2$ ranging from 6 to 48 (x-axis). The boxplot represents the distribution of the per-region intrinsic dimensions over 2000 sampled regions and 2000 different noise realizations. Recall that the intrinsic dimension is upper bounded by $S = 6$ in this case. Two key observations: first dropconnect tends to producing DGN with intrisic dimension preserving the latent dimension ($S = 6$) even for narrow models ($D_1, D_2 \approx S$), as opposed to dropout which tends to produce DGNs with much smaller intrinsic dimension than S. As a result, if the DGN is much wider than S, both techniques can be used, while in narrow models, either none or dropconnect should be preferred.

with Aff$(\omega; \boldsymbol{A}, \boldsymbol{b}) = \{\boldsymbol{A}\boldsymbol{z} + \boldsymbol{b} : \boldsymbol{z} \in \omega\}$*; we will denote for conciseness* $\boldsymbol{G}(\omega) \triangleq$ *Aff*$(\omega; \boldsymbol{A}_\omega, \boldsymbol{b}_\omega)$.
The above result is pivotal in bridging the understanding of the input space partition Ω, the per-region affine mappings $\boldsymbol{A}_\omega, \boldsymbol{b}_\omega$, and the generator mapping. We visualize Thm. 1 in Fig. 1 to make it clear that characterizing $\boldsymbol{A}_\omega$ alone already provides tremendous information about the generator, as we demonstrate in the next section.

3.2 Generated Manifold Intrinsic Dimension

We now turn into the intrinsic dimension of the per-region affine subspaces $\boldsymbol{G}(\omega)$ that are the pieces forming the generated manifold. In fact, as per (5), its dimension depends not only on the latent dimension S but also on the per-layer parameters.

Proposition 1. *The intrinsic dimension of the affine subspace* $\boldsymbol{G}(\omega))$ *(recall (5)) has the following upper bound:* $\dim(\boldsymbol{G}(\omega)) \leq \min\left(S, \min_{\ell=1,\dots,L}\left(\operatorname{rank}\left(\operatorname{diag}(\dot{\sigma}^\ell(\omega))\boldsymbol{W}^\ell\right)\right)\right)$.

We make two 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 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 $\dot{\sigma}^\ell(\omega)$ 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" (layers with width D^ℓ smaller than other layers) impact directly the dimension of the subspace and thus should be carefully employed based on the a priori knowledge of the target manifold intrinsic dimension. In particular, if $\alpha \neq 0$ and the weights are random (such as at initialization) then one has almost surely $\dim(\boldsymbol{G}(\omega)) \leq \min(S, \min_{\ell=1,\dots,L} D^\ell)$.

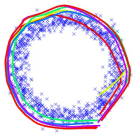


Figure 3: DGN with dropout trained (GAN) on a circle dataset (blue dots); dropout turns a DGN into an ensemble of DGNs (each dropout realization is drawn in a different color).

Application: Effect of Dropout/Dropconnect. Noise techniques, such as dropout Wager et al. (2013) and dropconnect Wan et al. (2013); Isola et al. (2017), alter the per-region affine mapping in a very particular way impact the per-region affine mapping. Those techniques perform an Hadamard product of iid Bernoulli random variables against the feature maps (dropout) or against the layer weights (dropconnect); we denote a DGN equipped with such technique and given a noise realization by $\boldsymbol{G}_\epsilon$ where ϵ includes the noise realization of all layers; note that $\boldsymbol{G}_\epsilon$ now has its own input space partition Ω_ϵ. We provide explicit formula of the piecewise affine deep generative network in Appendix J. In a classification setting, those techniques have been thoroughly studied and it was shown that adding dropout/dropconnect to a DN classifier turned the network into an ensemble of classifiers Warde-Farley et al. (2013); Baldi & Sadowski (2013); Bachman et al. (2014); Hara et al. (2016).

Proposition 2. *Adding dropout/dropconnect to a deep generative network* $\boldsymbol{G}$ *produces a (finite) ensemble of generators* $\boldsymbol{G}_\epsilon$*, each with per-region intrinsic dimension* $0 \leq \dim(\boldsymbol{G}_\epsilon(\omega_\epsilon)) \leq \max_{\omega\in\Omega} \dim(\boldsymbol{G}(\omega))$ *for* $\omega_\epsilon \in \Omega_\epsilon$*, those bounds are tight.*
We visualize this result in Fig. 2, where we observe how different noise realizations produce slightly different generators possibly with different per-region dimension. By leveraging the above and

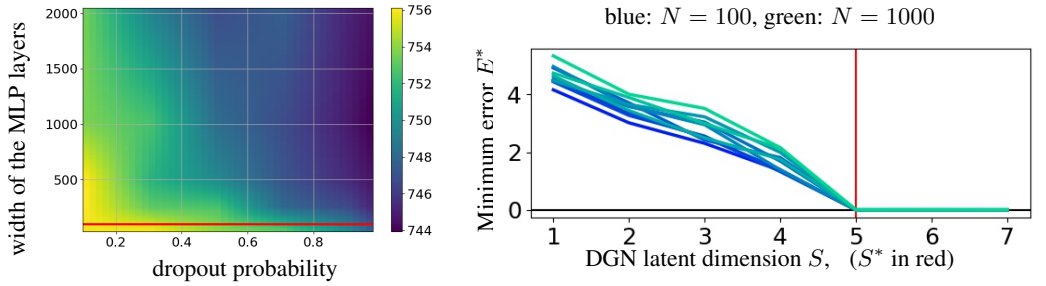


Figure 4: **Left:** final training negative ELBO (smaller is better) for a MLP DGN ($S = 100$ in red) on MNIST with varying width (y-axis) and dropout rates (x-axis) of the DGN (decoder); we observe that the impact of dropout on performances depends on the layer widths (CIFAR10 results can be seen in Fig. 9). **Right:**Minimal approximation error (E^*) for a target linear manifold ($S^* = 5$, in red), with increasing dataset size from 100 to 1000 (blue to green) and different latent space dimensions S (x-axis), the $E^* = 0$ line is depicted in black. This demonstrates (per Thm. 2) that whenever $\tilde{S} < S^*$, the training error E^* increases with the dataset size N.

Thm. 1 we can highlight a potential limitation of those techniques for narrow models ($D^\ell \approx S$) for which the noise induced generators will tend to be have a per-region intrinsic dimension smaller than S hurting the modeling capacity of those generators. On the other hand, when used with wide DGNs ($D^\ell \gg S$), noise induced generators will maintain an intrinsic dimension closer to the one of the original generator. We empirically demonstrate the impact of dropout into the dimension of the DGN surface in Fig. 3; and reinforce the possibly detrimental impact of employing large dropout rates in narrow DNs in Fig. 4. Those observations should open the door to novel technique theoretically establishing the relation between dropout rate, layer widths, and dimensionality of the desired (target) surface to produce.

From the above analysis we see that one should carefully consider the use of dropout/dropconnect based on the type of generator architecture that is used and the desired generator intrinsic dimension.

3.3 Intrinsic Dimension and The Elbow Method for Deep Generative Networks

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 $E_N^* = \min_\Theta \sum_{n=1}^N \min_{\boldsymbol{z}} \|\boldsymbol{G}_\Theta(\boldsymbol{z}) - \boldsymbol{x}_n\|$, a standard measure used for example in Bojanowski et al. (2017). We now demonstrate that if $\tilde{S} < S^*$ then E_N^* increases with N for any data manifold.

Theorem 2. *Given the intrinsic dimension of a target manifold S^*, then for any finite DGN with smaller intrinsic dimension ($\tilde{S} < S^*$), there exists a $N' \in \mathbb{N}^+$ such that for any dataset size $N_2 > N_1 \geq N' \geq N$ of random samples from the target manifold one has $E_N^* = 0$ and $E_{N_2}^* > E_{N_1}^*$.*

From the above result, we see that, even when fitting a linear manifold, whenever there is a bottleneck layer (very narrow width) collapsing the DGN intrinsic dimension below the target manifold dimension S^*, then as the dataset size exceeds a threshold, as the training error will increase. For illustration, consider a dataset formed by a few samples from a S^* dimensional affine manifold. If the number of samples is small, even a DGN with intrinsic dimension of 1 (thus producing a $1D$ continuous piecewise affine curve in $\mathbb{R}^D$) can fit the dataset. However, as the dataset size grows, as it will become impossible for such a low dimension (even though nonlinear) manifold to go through all the samples. We empirically observe this result in Fig. 4 (the experimental details are given in Appendix I). This should open the door to developing new techniques a la "Elbow method" Thorndike (1953); Ketchen & Shook (1996) to estimate intrinsic dimensions of manifolds based on studying the approximation error of DGNs.

The above results are key to understanding 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 PER-REGION AFFINE MAPPING INTERPRETABILITY, MANIFOLD SMOOTHING, AND INVERSE PROBLEMS

We now turn to the study of the local coordinates of the affine mappings comprising a DGN's generated manifold.

4.1 DEEP GENERATIVE NETWORK INVERSION

We now consider the task of *inverting* the DGN mapping to retrieve $\boldsymbol{z}_0$ from $\boldsymbol{x} = \boldsymbol{G}(\boldsymbol{z}_0)$. Note that we use the term "inversion" only in this way, since, in general, $Im(\boldsymbol{G})$ is a continuous piecewise affine surface in $\mathbb{R}^D$, and thus not all $\boldsymbol{x} \in \mathbb{R}^D$ can be mapped to a latent vector that would generate $\boldsymbol{x}$ exactly.

First, we obtain the following condition relating the per-region dimension to the bijectivity of the mapping. Bijectivity is an important property that is often needed in generators to allow uniqueness of the generated sample given a latent vector as well as ensuring that a generated sample has a unique latent representation vector. That is, following the above example, we would prefer that only a unique $\boldsymbol{z}_0$ produces $\boldsymbol{x}$. Such uniqueness condition is crucial in inverse problem applications Lucas et al. (2018); Puthawala et al. (2020).

Proposition 3. *A DGN employing leaky-ReLU activation functions is bijective between its domain* $\mathbb{R}^S$ *and its image* $Im(\boldsymbol{G})$ *iff* $\dim(\mathit{Aff}(\omega, \boldsymbol{A}_\omega, \boldsymbol{b}_\omega)) = S, \forall \omega \in \Omega$*, or equivalently, iff rank*$(\boldsymbol{A}_\omega) = S, \forall \omega \in \Omega$.

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 $\boldsymbol{G}_\omega^{-1}(\boldsymbol{x}) = \left(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega\right)^{-1} \boldsymbol{A}_\omega^T (\boldsymbol{x} - \boldsymbol{b}_\omega), \forall \boldsymbol{x} \in \boldsymbol{G}(\omega)$ 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; which is also a CPA operator and is given by $\boldsymbol{G}^{-1}(\boldsymbol{x}) = \sum_{\omega \in \Omega} \boldsymbol{G}_\omega^{-1}(\boldsymbol{x}) \mathbb{1}_{\{\boldsymbol{x} \in \boldsymbol{G}(\omega)\}}$. In particular, we obtain the following result offer new perspective of latent space optimization and inversion of DGNs.

Proposition 4. *Given a sample* $\boldsymbol{x} = \boldsymbol{G}(\boldsymbol{z}_0)$*, inverting* $\boldsymbol{x}$ *can be done via* $\min_{\boldsymbol{z}} \|\boldsymbol{x} - \boldsymbol{G}(\boldsymbol{z})\|_2^2$ *(standard technique, see), but can also be done via* $\min_{\omega \in \Omega} \|\boldsymbol{x} - \boldsymbol{G}(\boldsymbol{G}_\omega^{-1}(\boldsymbol{x}))\|_2^2$*, offering new solutions from discrete combinatorial optimization to invert DGNs.*

Recalling that there is a relation between the per-layer region activation r from Eq. 2 and the induced input space partition, we can see how the search through $\omega \in \Omega$ as given in the above result can further be cast as an integer programming problem Schrijver (1998). For additional details on this integer programming formulation of DN regions, see Tjeng et al. (2017); Balestriero et al. (2019) and Appendix G.

We leveraged $\boldsymbol{A}_\omega$ as the basis of $\text{Aff}(\omega; \boldsymbol{A}_\omega, \boldsymbol{b}_\omega)$ in order to derive the inverse DGN mapping, we now extend this analysis by linking $\boldsymbol{A}_\omega$ to local coordinates and adaptive basis to gain further insights and interpretability.

4.2 PER-REGION MAPPING AS LOCAL COORDINATE SYSTEM AND DISENTANGLEMENT

Recall from (5) 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 $(\boldsymbol{A}_\omega)$ which we assume to be full rank for any $\omega \in \Omega$ through this section.

The affine mapping 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 no unique definition for disentanglement, a general consensus is that 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 the d^{th} dimension $(d = 1, \dots, S)$ of $\boldsymbol{z}$ implies a transformation independent from a small perturbation of $d' \neq d$ 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 S 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).

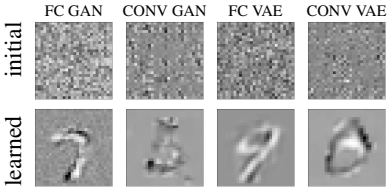


Figure 5: Visualization of a single basis vectors $[\boldsymbol{A}_\omega]_{.,k}$ before and after learning obtained from a region ω containing the digits $7, 5, 9$, and 0 respectively, for GAN and VAE models made of fully connected or convolutional layer (see Appendix E for details). We observe how those basis vectors encodes: right rotation, cedilla extension, left rotation, and upward translation respectively allowing interpretability into the learn DGN affine parameters.

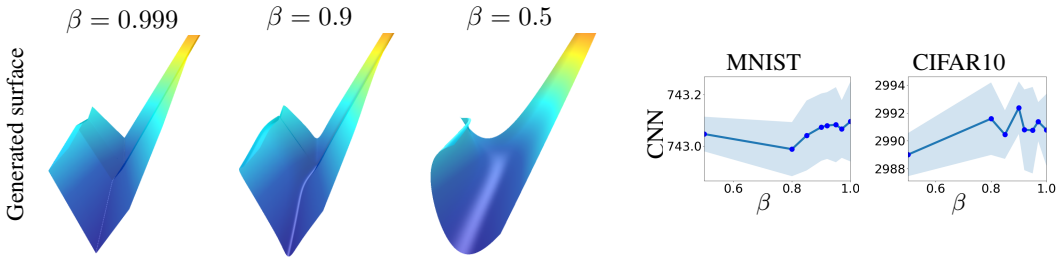


Figure 6: **Left:** Depiction of the DGN image $Im(\boldsymbol{G})$ for different values of smoothing β; clearly the smoothing preserves the overall form of the generated surface while smoothing the (originally) non-differentiable parts. No other change is applied on $\boldsymbol{G}$. **Right:** Negative ELBO (lower is better, avg. and std. over 10 runs) for different DGNs (VAE trained) with CNN architecture with varying smoothing parameters β. Regardless of the dataset, the trend seems to be that smoother/less-smooth models produce better performance in CNNs.

Proposition 5. *A necessary condition for disentanglement is to have "near orthogonal" columns, i.e.,* $\langle [\boldsymbol{A}_\omega]_{.,i}, [\boldsymbol{A}_\omega]_{.,j} \rangle \approx 0, \forall i, \neq j, \forall \omega \in \Omega$.

We provide in Table 1 in Appendix H 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 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. Additionally, Fig. 5 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 Zhao et al. (2017); Huang et al. (2018). To visually control the quality of the DGN, randomly generated digits are given in Fig. 14 in the Appendix; we also provide more background on disentanglement in Appendix H.

4.3 Manifold Smoothing

We conclude this section by demonstrating how to smooth, in a principled way, the generated CPA manifold from $\boldsymbol{G}$. Manifold smoothing has great advantages for example for denoising Park et al. (2004); Yin et al. (2008) and has been extensively study in univariate tasks with smoothing splines De Boor et al. (1978); Green & Silverman (1993); Wang (2011); Gu (2013) where a (smooth) spline is fit to (noisy) data while having the norm of its second (or higher order) derivative penalized. Recently, Balestriero & Baraniuk (2018c) demonstrated that one could smooth, in a principled way, the MASO layers through a probabilistic region assignment of the layer input. For example, a ReLU $\max(u, 0)$ is turned into sigmoid$(u\beta/(1-\beta))u$ with $\beta \in [0, 1]$ controlling the amount of smoothness. In the limit $\beta = 0$ makes the entire DGN a globally affine mapping, while $\beta \to 1$ brings back the CPA surface. We propose a simple visualization of the impact of β onto the generated manifold as well as VAE training experiments in Fig. 6 demonstrating that introducing some smoothness indeed provides better manifold fitting performances. While we only highlighted here the ability to smooth a DGN manifold and highlighted the potential improvement in performances, we believe that such control in the DGN smoothing opens the door to many avenues such as better denoising, or simpler optimization in inverse problem settings Wakin et al. (2005).

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 $\boldsymbol{p_z}$ from which $\boldsymbol{z}$ are sampled in turn leading to sampling of $\boldsymbol{G}(\boldsymbol{z})$; we now study this density and its properties.

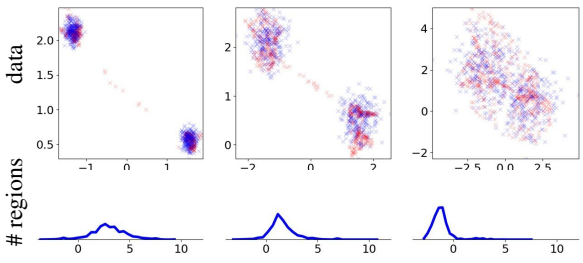


Figure 7: Distribution of the per-region log-determinants (bottom row) for DGN trained on a bimodal distribution with varying per mode variance (first row), demonstrating how the data multimodality and concentration pushes the per-region determinant of $\boldsymbol{A}_\omega$ to greatly increase in turn leading to large amplitude layer weights and impacting the stability of the DGN learning (also recall Lemma 1). Additional experiments are given in Fig. 15 in Appendix D.6.

5.1 Analytical Output Density

Given a distribution $\boldsymbol{p_z}$ 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; denote by $\sigma_i(\boldsymbol{A}_\omega)$ the i^{th} singular value of $\boldsymbol{A}_\omega$.

Lemma 1. *Then, the volume of a region $\omega \in \Omega$ denoted by $\mu(\omega)$ is related to the volume of $\boldsymbol{G}(\omega)$ by $\mu(\boldsymbol{G}(\omega)) = \sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)}\mu(\omega) = \prod_{i:\sigma_i(\boldsymbol{A}_\omega)>0} \sigma_i(\boldsymbol{A}_\omega)\mu(\omega)$.*

Theorem 3. *The generator probability density $p_{\boldsymbol{G}}(\boldsymbol{x})$ given $\boldsymbol{p_z}$ and a injective generator $\boldsymbol{G}$ with per-region inverse $\boldsymbol{G}_\omega^{-1}$ from Thm. 4 is given by $p_{\boldsymbol{G}}(\boldsymbol{x}) = \sum_{\omega\in\Omega} \frac{\boldsymbol{p_z}\left(\boldsymbol{G}_\omega^{-1}(\boldsymbol{x})\right)}{\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)}} \mathbb{1}_{\{\boldsymbol{x}\in\boldsymbol{G}(\omega)\}}$.*

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. We derive the analytical form for the case of Gaussian and Uniform latent distribution in Appendix B. From the analytical derivation of the generator density distribution, we obtain its differential entropy.

Corollary 1. *The differential Shannon entropy of the output distribution $\boldsymbol{p_G}$ of the DGN is given by $E(\boldsymbol{p_G}) = E(\boldsymbol{p_z}) + \sum_{\omega\in\Omega} P(\boldsymbol{z}\in\omega)\log(\sqrt{\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)})$.*

As the result, the differential entropy of the output distribution $\boldsymbol{p_G}$ corresponds to the differential entropy of the latent distribution $\boldsymbol{p_z}$ plus a convex combination of the per-region volume change. It is thus possible to optimize the latent distribution $\boldsymbol{p_z}$ to better fit the target distribution entropy as in Ben-Yosef & Weinshall (2018) and whenever the prior distribution is fixed, any gap between the latent and output distribution entropy imply the need for high change in volumes between ω and $\boldsymbol{G}(\omega)$. The use of the above results to obtain the analytical form of the density covering the generative manifold in the Gaussian and Uniform prior cases is done in Appendix B, where we relate DGN to Kernel Density Estimation (KDE) Rosenblatt (1956) and in particular adaptive KDE Breiman et al. (1977). We also carefully link DGNs and normalizing flows in Appendix C.

5.2 On the Difficulty of Generating Low Entropy/Multimodal Distributions

We conclude this study by hinting at the possible main cause of 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 $\boldsymbol{G}(\omega)$. Even when considering the a simple mixture of Gaussians case, it is clear that the standard deviation of the modes and the inter-mode distances will put constraints on the singular values of the slope matrix $\boldsymbol{A}_\omega$, in turn stressing the parameters $\boldsymbol{W}^\ell$ as they compose the slope matrix (see Fig. 10 in the Appendix for details on this relationship). This problem emerges from the continuous property of DGNs which have to somehow connect in the output space the different modes. We highlight this in Fig. 7, where we trained a GAN DGN on two Gaussians for different scaling

In conclusion, the continuity property is a source of the training instabilities while playing a key role into the DGN interpolation and generalization capacity. Understanding the relationships between the per-layer parameters, standard regularization techniques, such as Tikhonov, and the multimodality of the target distribution will thus enable practitioners to find the best compromise to stabilize training of a continuous DGN that provides sufficient data approximation.

REFERENCES

Sidney N Afriat. 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.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in neural information processing systems*, pp. 3365–3373, 2014.

Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in neural information processing systems*, pp. 2814–2822, 2013.

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

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

Randall Balestriero and Richard G Baraniuk. From hard to soft: Understanding deep network nonlinearities via vector quantization and statistical inference. *arXiv preprint arXiv:1810.09274*, 2018c.

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

Sudipto Banerjee and Anindya Roy. *Linear Algebra and Matrix Analysis for Statistics*. Chapman and Hall/CRC, 2014.

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

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

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

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

Leo Breiman, William Meisel, and Edward Purcell. Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144, 1977.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Fred B Bryant and Paul R Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

Pierre Comon. Independent Component Analysis, a new Concept? *Signal Processing*, 36(3):287–314, 1994.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

David L Donoho, Iain M Johnstone, 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.

Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

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

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, volume 1. MIT Press, 2016.

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

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Peter J Green and Bernard W Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.

Chong Gu. *Smoothing spline ANOVA models*, volume 297. Springer Science & Business Media, 2013.

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

Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*, pp. 72–79. Springer, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing systems*, pp. 52–63, 2018.

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

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, 2017.

David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

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

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

Claudia Lautensack and Sergei Zuyev. Random laguerre tessellations. *Advances in Applied Probability*, 40(3):630–650, 2008. doi: 10.1239/aap/1222868179.

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

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359*, 2018.

Alice Lucas, Michael Iliadis, Rafael Molina, and Aggelos K Katsaggelos. Using deep neural networks for inverse problems in imaging: beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

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

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.

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

JinHyeong Park, Zhenyue Zhang, Hongyuan Zha, and Rangachar Kasturi. Local smoothing for manifold learning. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pp. II–II. IEEE, 2004.

Michael Puthawala, Konik Kothari, Matti Lassas, Ivan Dokmanić, and Maarten de Hoop. Globally injective relu networks. *arXiv preprint arXiv:2006.08464*, 2020.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

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

Walter Rudin. *Real and Complex Analysis*. Tata McGraw-hill education, 2006.

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

Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

Michael Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press, 2018.

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

Robert L Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

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

Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1415–1424, 2017.

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

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing systems*, pp. 351–359, 2013.

M. B. Wakin, D. L. Donoho, H. Choi, and R. G. Baraniuk. The multiscale structure of non-differentiable image manifolds. In *Proc. Int. Soc. Optical Eng.*, pp. 59141B1–59141B17, July 2005.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pp. 1058–1066, 2013.

Shuning Wang and Xusheng Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

Yuedong Wang. *Smoothing splines: methods and applications*. CRC Press, 2011.

David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197*, 2013.

Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks, 2019.

Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 676–684, 2015.

Junsong Yin, Dewen Hu, and Zongtan Zhou. Noisy manifold learning using neighborhood smoothing embedding. *Pattern Recognition Letters*, 29(11):1613–1620, 2008.

Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. *arXiv preprint arXiv:1805.07091*, 2018.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017.

SUPPLEMENTARY MATERIAL

The following appendices support the main paper and are organized as follows: Appendix A provides additional insights that can be gained in the generated surface $\boldsymbol{G}$ by studying its angularity, that is, how adjacent affine subspaces are aligned with each other. This brings new ways to capture and probe the "curvature" of the generated surface. Appendix B provides additional details on the probability distribution leaving on the generated surface while Appendix C provides a careful bridge between DGNs and Normalizing Flows leveraging the proposed CPA formulation. Lastly, Appendices D and E provide additional figures and training details for all the experiments that were studied in the main text, and Appendix F provides the proofs of all the theoretical results.

A GENERATED MANIFOLD ANGULARITY

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 1. *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 $P(\boldsymbol{A}_\omega) = \boldsymbol{A}_\omega(\boldsymbol{A}_\omega^T\boldsymbol{A}_\omega)^{-1}\boldsymbol{A}_\omega^T$ 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.[1]

Theorem 4. *The angle between adjacent (recall Def. 1) region mappings $\theta(\boldsymbol{G}(\omega), \boldsymbol{G}(\omega'))$ is given by* $\sin\left(\theta(\boldsymbol{G}(\omega), \boldsymbol{G}(\omega'))\right) = \|P(\boldsymbol{A}_\omega) - P(\boldsymbol{A}_{\omega'})\|_2, \forall \omega \in \Omega, \omega' \in adj(\omega)$.

Notice that in the special case of $S = 1$, the angle is given by the cosine similarity between the vectors $\boldsymbol{A}_\omega$ and $\boldsymbol{A}_{\omega'}$ of adjacent regions, and 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. We illustrate the angles in a simple case $D = 2$ and $Z = 1$ in Fig. 3. 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. 11 in the Appendix as well as repetitions of the same experiment with different random seeds.

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 8 summarizes the distribution of angles for several different settings from which we observe two key trends. First, as codes of adjacent regions $\boldsymbol{q}(\omega), \boldsymbol{q}(\omega')$ share a large number of their values (see Appendix G for details) the affine parameters $\boldsymbol{A}_\omega$ and $\boldsymbol{A}_{\omega'}$ of adjacent regions also benefit from this *weight sharing* constraining the angles between those regions to be much smaller than if they were random and independent which favors aggressively large angles. 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 making the manifold "flatter" overall except in some parts of the space where high angularity is present.

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.

B DISTRIBUTIONS

Gaussian Case. We now demonstrate the use of the above derivation by considering practical examples for which we are able to gain ingights into the DGN data modeling and generation. First,

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

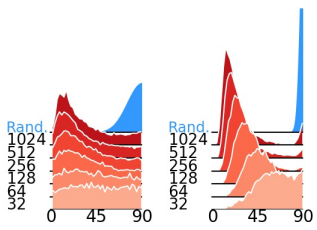


Figure 8: Histograms of the DGN adjacent region angles for DGNs with two hidden layers, $S = 16$ and $D = 17, D = 32$ respectively and varying width D_ℓ on the y-axis. Three trends to observe: increasing the width increases the bimodality of the distribution while favoring near 0 angles; increasing the output space dimension increases 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' makes those distributions much more concentrated near 0 than with no weight sharing (depicted in blue), leading to an overall well behave manifold. Additional figures are available in Fig. 11.

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

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

giving the desired result. □

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 on a bounded domain U in the DGN input space. By employing the given formula one can directly obtain that the output density is given by

$$p_{\boldsymbol{G}}(\boldsymbol{x}) = \frac{\sum_{\omega \in \Omega} \mathbb{1}_{\boldsymbol{x} \in \omega} \det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega)^{-\frac{1}{2}}}{Vol(U)} \tag{6}$$

We can also 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 6. *Given a uniform latent distribution $\boldsymbol{v} \sim \mathcal{U}(0, 1)$, the sampling of the manifold $\boldsymbol{G}(supp(\boldsymbol{p_z}))$ will be uniform iff $\det(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega) = c, \forall \omega : \omega \cap supp(\boldsymbol{p_z}) \neq \emptyset, c > 0$.*

C Normalizing Flows and DGNs

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 $\sqrt{\det\left((\boldsymbol{A}_\omega^+)^T \boldsymbol{A}_\omega^+\right)} = (\sqrt{\det\left(\boldsymbol{A}_\omega^T \boldsymbol{A}_\omega\right)})^{-1}$; thus we can write the log density from over a sample $\boldsymbol{x}_n$ as $\mathcal{L}(\boldsymbol{x}_n) = \log(\boldsymbol{p_z}(\boldsymbol{G}^{-1}(\boldsymbol{x}_n))) + \log(\sqrt{\det(J(\boldsymbol{x}_n))})$, 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.

D Extra Figures

D.1 Dropout and DGNs

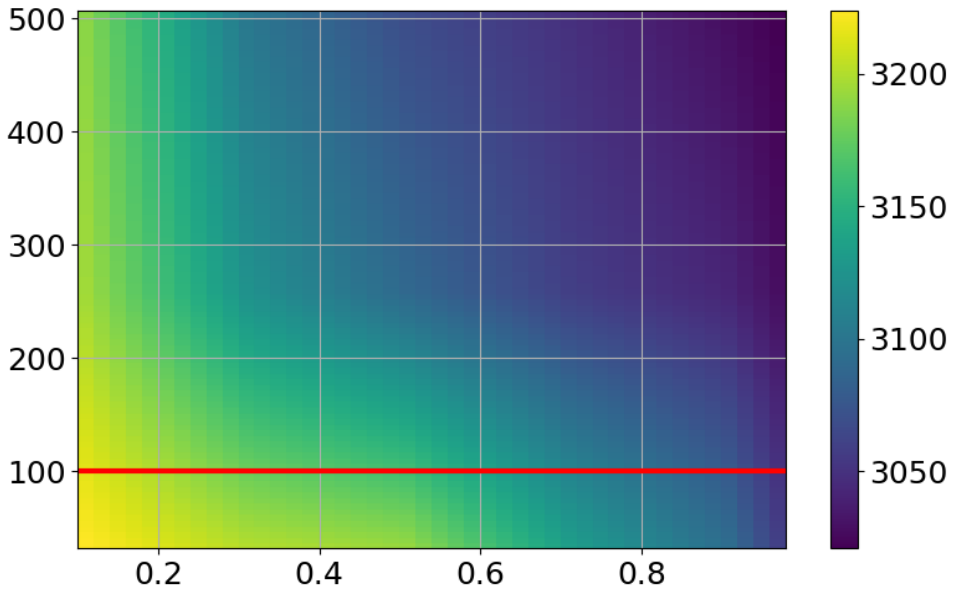
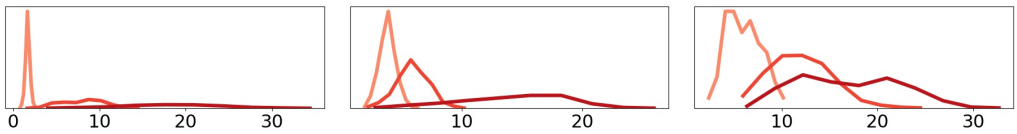


Figure 9: Reprise of Fig. 4

D.2 Example of Determinants



$\sigma_1 = 0, \sigma_2 \in \{1, 2, 3\}$ $\sigma_1 = 1, \sigma_2 \in \{1, 2, 3\}$ $\sigma_1 = 2, \sigma_2 \in \{1, 2, 3\}$

Figure 10: 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.

D.3 ANGLES HISTOGRAM

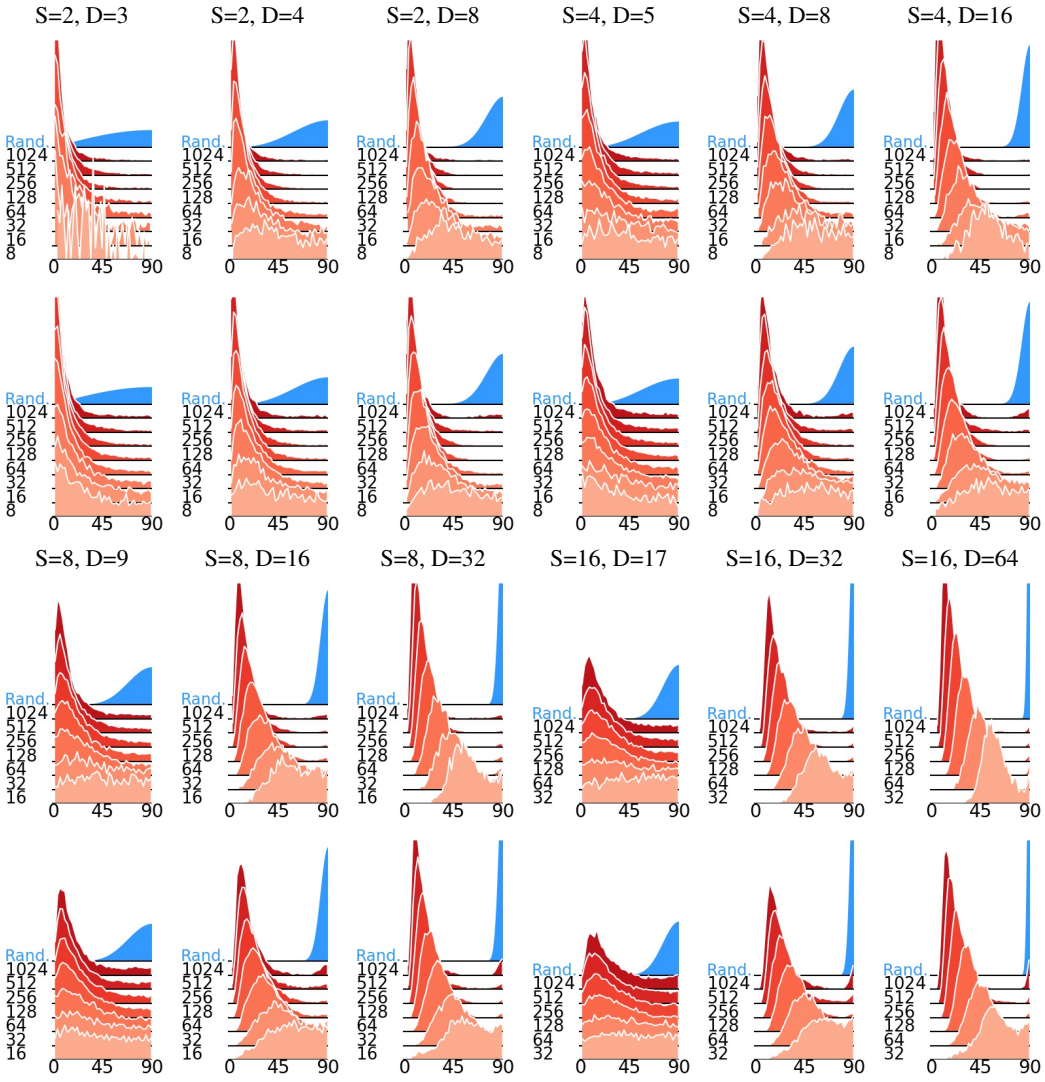


Figure 11: Reproduction of Fig. 8. Histograms of the largest principal angles for DGNs with one hidden layer (first two rows) and two hidden layers (last two rows). In each case the latent space dimension and width of the hidden layers is in the top of the column. The observations reinforce the claims on the role of width and S versus D dimensions.

D.4 ANGLES MANIFOLD

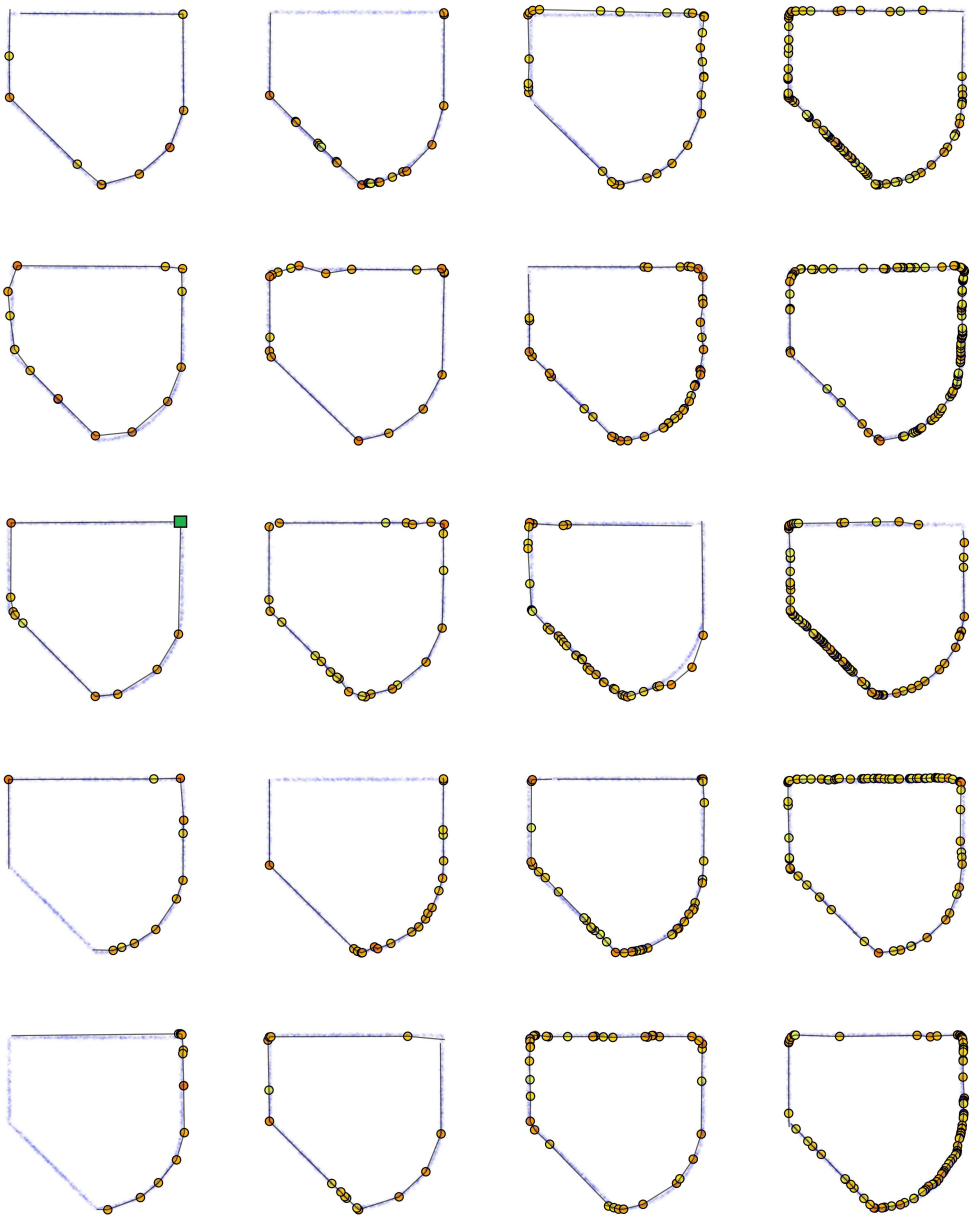


Figure 12: The columns represent different widths $D_\ell \in \{6, 8, 16, 32\}$ and the rows correspond to repetition of the learning for different random initializations of the GDNs for consecutive seeds.

D.5 More on MNIST Disentanglement

Figure 13: Randomly generated digits from the trained GAN (top) and trained VAE(bottom) models for the experiment from Fig. 5. 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 14: Randomly generated digits from the trained CONV GAN (top) and trained CONV VAE(bottom) models for the experiment from Fig. 5. Each row represents a model that was training on a different random initialization (8 runs in total) which produced the result in Table 1.

D.6 MORE ON DETERMINANT FIGURES

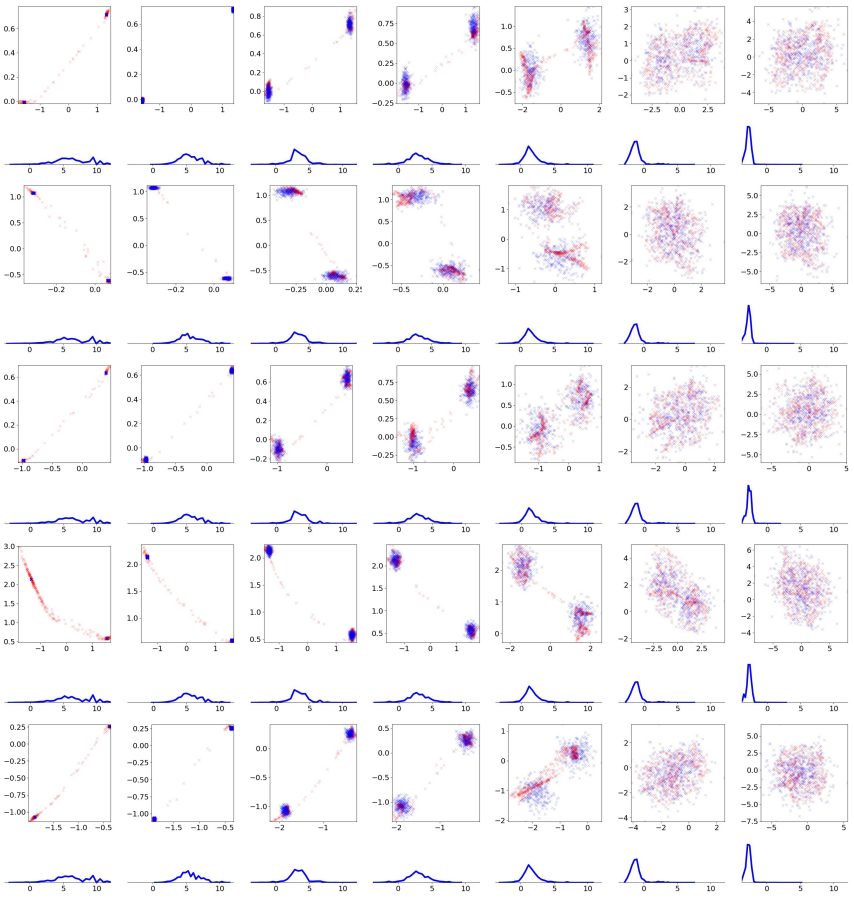


Figure 15: Reproduction of Fig. 7 for multiple standard deviations and multiple random seeds. Each column represent a different standard deviation of the two Gaussians $\sigma \in \{0.002, 0.01, 0.05, 0.1, 0.3, 1, 2\}$ and each row is a run with a different seed. As can be seen in all cases (except when lack of convergence) the distribution of the determinants support the claim and relate with the Entropy of the true distribution (blue points).

E 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.

F PROOFS

F.1 PROOF OF THM 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 (5). □

F.2 PROOF OF PROPOSITION 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 Banerjee & Roy (2014) 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. □

F.3 PROOF OF PROPOSITION 2

Proof. The first bound is obtained by taking the realization of the noise where $\boldsymbol{r} = \boldsymbol{0}$, 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 $\boldsymbol{0}$ 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 $\boldsymbol{r} = \boldsymbol{1}$ which gives the result. □

F.4 PROOF OF PROPOSITION 3

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 $\text{supp}(\boldsymbol{p_z})$ and $\boldsymbol{G}(\text{supp}(\boldsymbol{p_z}))$. 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 $\text{supp}(\boldsymbol{p_z})$ is mapped to an unique point in $\boldsymbol{G}(\text{supp}(\boldsymbol{p_z}))$ which gives global bijectivity.

□

F.5 Proof of Lemma 4

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. □

F.6 Proof of Proposition 5

Proof. The proof is straightforward from the used definition of disentenglement. 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 $\boldsymbol{A}_\omega$. □

F.7 Proof of Theorem 4

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 ($\text{rank}(\boldsymbol{A}_\omega) = S$) 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 $\sin\left(\theta(\boldsymbol{G}(\omega), \boldsymbol{G}(\omega'))\right) = \|P(\boldsymbol{A}_\omega) - P(\boldsymbol{A}_{\omega'})\|_2$ as desired. □

F.8 Proof of Lemma 1

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 Rudin (2006)). 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 Spivak (2018) 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:

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

□

F.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

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

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. □

F.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:

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

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

F.11 PROOF OF THEOREM 2

Proof. Suppose the target manifold is of intrinsic dimension S^*. Samples from it are obtain by random sampling (it does not need to be uniform however the support of the distribution need to cover the entire manifold which simply means that each part of the target manifold must have a nonzero probability to produce a sample). We have:

1. A DGN with intrinsic dimension $\tilde{S} < S^*$ can only perfectly approximate $\tilde{S}+1$ samples on each region of its piecewise affine generated manifold: almost surely no more than $\tilde{S}+1$ target manifold samples lie on the same affine subspace of dimension $\tilde{S}$.
2. Each region has a unique set of affine parameters: almost surely, one can not group N target manifold random samples into groups of $\tilde{S}+1$ samples such that affine subpsace span by each group of samples has same parameters than any other group.

Combining 1. and 2. it is clear that one can directly obtain a tight bound on the number of regions R^* in the approximant partition needed to fit the samples. Given N samples from the target manifold, any continuous piecewise affine approximant of intrinsic dimension $\tilde{S}$ must have at least $\frac{N}{\tilde{S}+1}$ pieces (and thus regions) to perfectly fit the target manifold.

Small dataset size case. Now given a DGN it is clear that if the number of samples lead to $\frac{N}{\tilde{S}+1}$ being smaller than the number of regions in the DGN, the approximation error E_N^* will be 0 and this with N increasing from 1 to the bound limit. This proves that $E_N^* = 0$ for $N \leq R(\tilde{S}+1)$ with R the number of regions of the considered CPA approximant.

Increasing dataset size case. When N further increases it implies that more than $\tilde{S}+1$ will be approximated by the same local affine subspace of the approximant. Let first assume that we do not enforce continuity and that we simply add a single point of the given dataset. That is, only the piecewise affine approximation of one region (which is now with an additional point) needs to be adapted. From OLS (which is now done on this region specifically) we see that adding a new sample will strictly increase the training set approximation error (the marginal increase decays as

Table 1: Depiction of the cosine similarity summed over pairwise different columns of $\boldsymbol{A}_\omega$ (0 means orthogonal basis vectors, improving disentanglement from Prop. 5)), the maximum (first row) and average (second column) are presented, computed over 10000 sampled regions ω. 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

the number of samples already present in a region grows, in the limit going to 0). Now if continuity is enforced, it only implies that the increase in the training error will further increase as (i) the region OLS is now constrained by the continuity constraints and (ii) the adjacent regions must also move from their previous optimum to fulfill that constraint with the adapted region; so not only the considered region error will further increase than in the discontinuous but also the adjacent regions approximation will increase. This proves that $E^*_{N_2} > E^*_{N_1}$ for any $N_2 > N_1$ as long as $N_1 > N'$ with N' given by $\frac{N}{S+1}$. □

G 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. 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. 1), 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 Lautensack & Zuyev (2008).

H 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 Yim et al. (2015); Tran et al. (2017). For a more in-depth review of learning disentangled representation, see Locatello et al. (2018).

I 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 $L = 3$.

J More on Effect of Dropout/Dropconnect

Thus, the noisy generator actually combines all the above mappings for each noise realization via $\widetilde{\boldsymbol{G}}(\text{supp}(p_{\boldsymbol{z}})) = \bigcup_{\boldsymbol{\epsilon} \in \text{supp}(p_{\boldsymbol{\epsilon}})} \boldsymbol{G}(\text{supp}(p_{\boldsymbol{z}})|\boldsymbol{\epsilon})$. Denote by $\widetilde{\boldsymbol{G}}$ the generator $\boldsymbol{G}$ equipped with dropout/-dropconnect and by $\widetilde{\boldsymbol{G}}(\boldsymbol{z}|\boldsymbol{\epsilon})$ the case where the noise realization is given a priori. For dropout and by leveraging Eq. 1 we have

$$\boldsymbol{G}(\boldsymbol{z}|\boldsymbol{\epsilon}) = \left(\prod_{\ell=L}^{1} \text{diag}(\boldsymbol{q}_\ell(\omega) \odot \epsilon_\ell) \boldsymbol{W}_\ell \right) \boldsymbol{z} + \sum_{\ell=1}^{L} \left(\prod_{\ell'=L}^{\ell+1} \text{diag}(\boldsymbol{q}_{\ell'}(\omega) \odot \epsilon_{\ell'}) \boldsymbol{W}_{\ell'} \right) \boldsymbol{b}_\ell, \tag{7}$$

where $\odot$ is the Hadamard product and $\boldsymbol{z} \in \omega$. 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. 2 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. You may include other additional sections here.