THE UNION OF MANIFOLDS HYPOTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning has had tremendous success at learning low-dimensional representations of high-dimensional data. This success would be impossible if there was no hidden low-dimensional structure in data of interest; this existence is posited by the *manifold hypothesis*, which states that the data lies on an unknown manifold of low intrinsic dimension. In this paper, we argue that this hypothesis does not properly capture the low-dimensional structure typically present in data. Assuming that data lies on a single manifold implies intrinsic dimension is identical across the entire data space, and does not allow for subregions of this space to have a different number of factors of variation. To address this deficiency, we put forth the *union of manifolds hypothesis*, which states that data lies on a disjoint union of manifolds of varying intrinsic dimensions. We empirically verify this hypothesis on commonly-used image datasets, finding that indeed, observed data lies on a disconnected set and that intrinsic dimension is not constant. We also provide insights into the implications the union of manifolds hypothesis has for deep learning, both supervised and unsupervised, showing that designing models with an inductive bias for this structure improves performance across classification and generative modelling tasks.

1 INTRODUCTION

The manifold hypothesis (Bengio et al., 2013) states that high-dimensional data of interest often lives in an unknown lower-dimensional manifold embedded in ambient space, and there is strong evidence supporting this hypothesis. From a theoretical perspective, it is known that both manifold learning and density estimation scale exponentially with the (low) *intrinsic* dimension when such structure exists (Ozakin & Gray, 2009; Narayanan & Mitter, 2010), while scaling exponentially with the (high) *ambient* dimension otherwise (Cacoullos, 1966). Thus, the most plausible explanation for the success of machine learning methods on high-dimensional data is the existence of far lower intrinsic dimension, which facilitates learning on datasets of fairly reasonable size. This is verified empirically by Pope et al. (2021), in which a comprehensive study estimating the intrinsic dimension of commonly-used datasets is performed, clearly finding low-dimensional structure.

However, thinking of observed data as lying on a single unknown low-dimensional manifold is quite limiting, as this immediately implies that the intrinsic dimension throughout the dataset is constant. If we consider the intrinsic dimensionality to be the number of factors of variation generating the data, we can see that this formulation prevents distinct regions of the data's support from having differing quantities of factors of variation. Yet this seems to be unrealistic: for example, we should not expect the number of factors needed to describe 8s and 1s in the MNIST dataset (LeCun et al., 1998) to be equal.

To accommodate this intuition, in this paper we propose the *union of manifolds* hypothesis. We posit that high-dimensional data often lies not on a single manifold, but on a disjoint union of manifolds of *different intrinsic dimensions*.[1] While this hypothesis has been mentioned in the clustering literature (Elhamifar & Vidal, 2011; 2012; 2013; Abdolali & Gillis, 2021), to the best of our knowledge it has never been empirically explored analogously to the way that Pope et al. (2021) probe the manifold hypothesis. In this work we carry out this verification on commonly-used image datasets, first by confirming their supports are disconnected, and then by estimating the intrinsic dimension on each

[1]The disjoint union of d-dimensional manifolds is itself a d-dimensional manifold: the possibility of having different intrinsic dimensions separates the union of manifolds hypothesis from the manifold hypothesis.

component, finding that there is indeed variation in these estimates. In order to verify that the support of the data is disconnected, we leverage *pushforward deep generative models* (Salmona et al., 2022; Ross et al., 2022), which generate samples by transforming noise through a neural network G. We first prove that these deep generative models (DGMs) are incapable of modelling disconnected supports. We then argue that the class labels provided in our considered datasets identify connected components (i.e. different classes are disconnected from each other), and show that training a pushforward model on each class outperforms training a single such model on the entire dataset, even when using the same computational budget: this improvement is a firm indicator that the support is truly disconnected, thus strongly supporting our thesis.

After empirically verifying the union of manifolds hypothesis, we turn our attention to some of its implications in deep learning. We establish that classes with higher intrinsic dimension are harder to classify, and guided by this insight, we also show that classification accuracy can be improved by more heavily weighting the terms corresponding to classes of higher intrinsic dimension in the cross entropy loss. We also show that the same DGMs we used to confirm the disconnectedness of the data support, which we call clustered DGMs, provide a performant class of models which are competitive with their non-clustered baselines across a wide range of datasets and models, and thus provide a promising direction towards improving generative models.

2 Background and related work

Throughout our paper, we consider the setup where we have access to a dataset $\mathcal{D} = \{x_i\}_{i=1}^n$, generated i.i.d. from some distribution $\mathbb{P}^*$ in a high dimensional ambient space $\mathcal{X} = \mathbb{R}^D$.

Pushforward DGMs As mentioned in the introduction, we leverage DGMs – in particular pushforward DGMs – in order to verify the disconnectedness of the support of $\mathbb{P}^*$. We call a *pushforward model* any DGM whose samples X are given by:

$$Z \sim \mathbb{P}_Z \quad \text{and} \quad X = G(Z), \tag{1}$$

where $\mathbb{P}_Z$ is a (potentially trainable) base distribution in some latent space $\mathcal{Z}$, and $G : \mathcal{Z} \to \mathcal{X}$ is a neural network. We highlight many popular DGMs fall into this category, including (Gaussian) variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014), normalizing flows (NFs) (Dinh et al., 2017; Kingma & Dhariwal, 2018; Behrmann et al., 2019; Chen et al., 2019; Durkan et al., 2019), generative adversarial networks (GANs) (Goodfellow et al., 2014), and Wasserstein autoencoders (WAEs) (Tolstikhin et al., 2018).

Intrinsic dimension estimation If we assume that $\mathbb{P}^*$ is supported on a d-dimensional manifold embedded in $\mathcal{X}$ for some unknown $d < D$, a natural question is how to estimate d from $\mathcal{D}$. Pope et al. (2021, Section 3) provide an excellent overview of the topic and cover the main estimator of intrinsic dimension that we will use in our paper: the Levina & Bickel (2004) estimator with the MacKay & Ghahramani (2005) extension, given by:

$$\hat{d}_k \coloneqq \left(\frac{1}{n(k-1)} \sum_{i=1}^{n} \sum_{j=1}^{k-1} \log \frac{T_k(x_i)}{T_j(x_i)} \right)^{-1}, \tag{2}$$

where $T_j(x)$ is the Euclidean distance from x to its jth-nearest neighbour in $\mathcal{D} \setminus \{x\}$, and k is a hyperparameter specifying the maximum number of nearest neighbours to consider. While other estimators have been recently proposed (Block et al., 2021; Lim et al., 2021; Tempczyk et al., 2022), we stick with (2) throughout this work as it is well-established in the literature. As we will see, popular image datasets exhibit different intrinsic dimensions in separate regions of data space.

On the relevance of understanding low-dimensional structure As mentioned in the introduction, the manifold hypothesis provides intuition as to why deep learning has been so successful. One might naïvely believe its usefulness is merely conceptual, yet uncovering and exploiting the structure underlying data of interest is a highly relevant problem (Bronstein et al., 2021) which has recently started receiving increased attention. Pope et al. (2021) use (2) to estimate the intrinsic dimension of image datasets, not only finding strong evidence of low-dimensional structure, but also uncovering

that datasets of higher intrinsic dimension are harder to classify. Similarly, Ansuini et al. (2019) link the intrinsic dimension of the representations of deep classifiers to their classification accuracy. Furthermore, the low-dimensional structure of data not only affects supervised methods: for example, training a high-dimensional density to model data lying on a low-dimensional manifold through maximum-likelihood can result in manifold overfitting (Dai & Wipf, 2019; Loaiza-Ganem et al., 2022), a phenomenon resulting in $\mathbb{P}^*$ not being learned, even if an infinite amount of data was observed. Indeed, there is a rapidly growing body of work aiming to account for the manifold hypothesis within DGMs (Gemici et al., 2016; Rezende et al., 2020; Brehmer & Cranmer, 2020; Mathieu & Nickel, 2020; Arbel et al., 2021; Kothari et al., 2021; Caterini et al., 2021; Ross & Cresswell, 2021; Cunningham et al., 2022; De Bortoli et al., 2022; Ross et al., 2022). All these works highlight the relevance of properly understanding and accounting for the low-dimensional structure of data, an understanding we improve upon in this work.

3 THE UNION OF MANIFOLDS HYPOTHESIS

We will assume that the support of $\mathbb{P}^*$, $supp(\mathbb{P}^*)$, can be written as:

$$supp(\mathbb{P}^*) = \bigsqcup_{\ell=1}^{L} \mathcal{M}_\ell \subset \mathcal{X}, \tag{3}$$

where $\sqcup$ denotes disjoint union, and each $\mathcal{M}_\ell$ is a connected manifold of dimension $d^{(\ell)}$. We call this assumption the *union of manifolds hypothesis*. Note that we make the assumption that each $\mathcal{M}_\ell$ is connected merely for notational simplicity, as we could collapse each union of manifolds of matching dimension into a single disconnected manifold, but this notation allows us to reason about different $\mathcal{M}_\ell$ as connected components of interest. Note also that the standard manifold hypothesis is equivalent to adding the assumption that all $d^{(\ell)}$s are identical. In the rest of this section we verify our hypothesis in two parts: first by establishing the disconnectedness of $supp(\mathbb{P}^*)$, and then by estimating the intrinsic dimension on each of its connected components.

3.1 VERIFYING THE HYPOTHESIS: DISCONNECTEDNESS

Before empirically confirming that the support of commonly-used image datasets is disconnected, let us develop intuition as to why this might be the case. For example, on MNIST we can likely continuously transform any 2 into any other 2 without leaving the manifold of 2s, whereas attempting to similarly transform a 2 into an 8 would likely result in intermediate images that are neither 2s nor 8s. We conjecture that class labels, which are often available for our considered datasets, provide a sensible proxy for the connected components $\mathcal{M}_\ell$ of $supp(\mathbb{P}^*)$, precisely because one can conceive of continuously transforming any one member of any class into any other member of the same class without ever leaving the class.

In order to verify this conjecture, we leverage a shortcoming of pushforward models, namely their inability to model disconnected supports, which we formalize in the following proposition. We use measure-theoretic notation, where the model from (1) is given by the pushforward measure $G_\# \mathbb{P}_Z$.

Proposition 1: Let $\mathcal{Z}$ and $\mathcal{X}$ be topological spaces and $G : \mathcal{Z} \to \mathcal{X}$ be continuous. Considering $\mathcal{Z}$ and $\mathcal{X}$ as measurable spaces with their respective Borel σ-algebras, let $\mathbb{P}_Z$ be a probability measure on $\mathcal{Z}$ such that $supp(\mathbb{P}_Z)$ is connected and $\mathbb{P}_Z(supp(\mathbb{P}_Z)) = 1$.[2] Then $supp(G_\# \mathbb{P}_Z)$ is connected.

Proof sketch: We begin by pointing out that from the formal definition of support, $supp(G_\# \mathbb{P}_Z)$ need not be equal to $G(supp(\mathbb{P}_Z))$. In other words, the proof does not immediately follow from the fact that continuous functions map connected sets to connected sets. In our proof we first show that $supp(G_\# \mathbb{P}_Z) = cl(G(supp(\mathbb{P}_Z)))$, where $cl(\cdot)$ denotes closure in $\mathcal{X}$, and the result follows from the closure of connected sets being connected. The formal proof along with the definition of support is included in Appendix A.

The implication of Proposition 1 is simple: if the support of the data is not connected (i.e. has multiple connected components), then pushforward models are insufficient to properly model the data, since in practice $supp(\mathbb{P}_Z)$ is always connected.

[2]The condition $\mathbb{P}_Z(supp(\mathbb{P}_Z)) = 1$ is highly technical and holds in all settings of practical interest, but can be violated in contrived counterexamples. See Appendix A for a discussion.

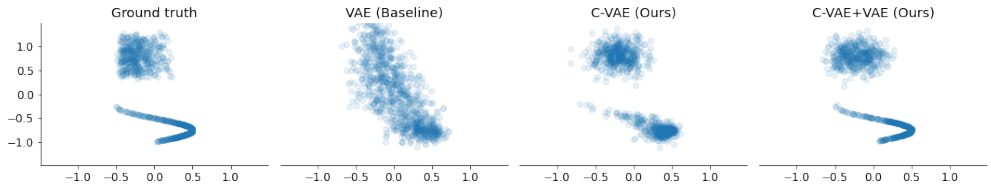

Figure 2: Samples from various VAE models on a synthetic dataset.

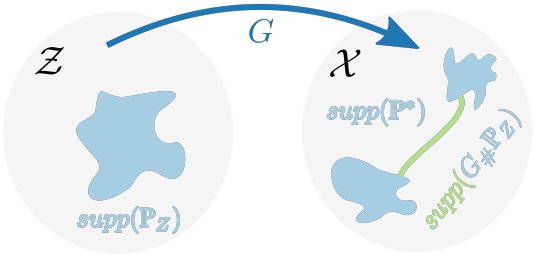

Figure 1: Depiction of the inability of pushforward models to represent multiple connected components. If $supp(\mathbb{P}^*)$ is given by the two blue connected components on $\mathcal{X}$, then by continuity, G cannot map $supp(\mathbb{P}_\mathcal{Z})$ to this set, and so its image must include the green region connecting them. Proposition 1 shows that the model $G_\#\mathbb{P}_Z$ cannot learn to assign 0 probability to this green region, as the support of a distribution can be intuitively understood as the "smallest" set to which it assigns probability 1.

Intuitively, this result rules out the possibility of $G_\#\mathbb{P}_Z$ assigning 0 probability to regions of $G(supp(\mathbb{P}_Z))$ which connect different disconnected components of the data support $supp(\mathbb{P}^*)$, as illustrated in Figure 1. We highlight that topological limitations of pushforward models have been studied before (Cornish et al., 2020; Salmona et al., 2022; Ross et al., 2022), although to the best of our knowledge not to the generality of Proposition 1. Indeed, there are many methods aiming to better model disconnected supports with pushforward models (Arora et al., 2017; Banijamali et al., 2017; Locatello et al., 2018; Khayatkhoei et al., 2018; Dinh et al., 2019; Tanielian et al., 2020; Luzi et al., 2020; Pires & Figueiredo, 2020; Duan, 2021; Ye & Bors, 2021). Figure 2, which we will use as a running example throughout this section, further visualizes this result, with the first panel showing a synthetic dataset obeying the union of manifolds hypothesis (two connected components of respective intrinsic dimensions 1 and 2), along with a Gaussian VAE trained on this dataset on the second panel. The VAE fails to recover a disconnected support, further confirming this limitation of pushforward models. We include all experimental details pertaining to this section in Appendix D.1.

As a means to empirically verify that $supp(\mathbb{P}^*)$ is disconnected, we introduce *clustered DGMs*: rather than train a single pushforward model $(\mathbb{P}_Z, G)$ on $\mathcal{D}$ as in standard DGMs, we first partition the data as $\mathcal{D} = \sqcup_{\ell=1}^{L} \mathcal{D}_\ell$, where $\mathcal{D}_\ell$ contains only the datapoints belonging to the ℓth class, and we then train a pushforward DGM $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ on each cluster $\mathcal{D}_\ell$. In order to obtain a sample X from a trained clustered DGM, we simply sample ℓ with probability proportional to the number of datapoints in $\mathcal{D}_\ell$, $|\mathcal{D}_\ell|$, and then sample from the corresponding DGM:

$$\ell \sim p(\ell) = \frac{|\mathcal{D}_\ell|}{\sum_{\ell'=1}^{L} |\mathcal{D}_{\ell'}|}, \qquad Z \mid \ell \sim \mathbb{P}_Z^{(\ell)}, \quad \text{and} \quad X = G_\ell(Z), \tag{4}$$

which can be performed in a memory-efficient way (i.e. without having to load L models in memory) as described in Appendix B.2.

We see clustered DGMs as the most straightforward modification that can be done to pushforward models so as to enable them to model disconnected supports. Clustered DGMs also allow to straightforwardly compare against standard pushforward models in a FLOP-equivalent manner that uses the same architectures (see Appendix B.1 for details). If we observe any improvement of clustered DGMs over their standard counterparts, the **simplest explanation for this behaviour would be that $supp(\mathbb{P}^*)$ is disconnected**. The third panel of Figure 2 shows a clustered VAE (denoted C-VAE), which correctly recovers two connected components (albeit not their intrinsic dimensions, which we will address later with the fourth panel), and was given the same computational budget as the VAE on the second panel: demonstrating that clustered DGMs show improvements over their standard counterparts when data is disconnected, thus providing more evidence for the validity of our test of disconnectedness.

Table 1: FID scores. We show means and standard errors across 3 runs.

Model	MNIST	FMNIST	SVHN	CIFAR-10	CIFAR-100
VAE	22.4 ± 0.0	18.5 ± 0.1	10.1 ± 0.3	16.1 ± 1.7	15.0 ± 0.1
C-VAE (random)	22.0 ± 0.7	17.4 ± 0.0	9.4 ± 0.3	14.8 ± 0.0	16.2 ± 0.1
C-VAE (classes)	$\mathbf{19.3 \pm 0.0}$	$\mathbf{14.4 \pm 0.0}$	$\mathbf{9.0 \pm 0.1}$	14.5 ± 0.2	15.1 ± 0.1
VAE (GMM)	22.5 ± 0.0	18.7 ± 0.1	$\mathbf{8.8 \pm 0.2}$	15.3 ± 0.2	14.9 ± 0.1
Conditional VAE (classes)	21.0 ± 0.1	17.7 ± 0.1	9.5 ± 0.2	$\mathbf{14.1 \pm 0.1}$	$\mathbf{13.5 \pm 0.0}$
Conditional VAE (clusters)	21.4 ± 0.2	18.0 ± 0.1	10.0 ± 0.5	$\mathbf{14.2 \pm 0.1}$	14.4 ± 0.6
C-VAE (clusters)	19.6 ± 0.0	15.5 ± 0.1	9.7 ± 0.0	14.6 ± 0.1	14.0 ± 0.0

The top half of Table 1 shows analogous results in the more realistic setting of image datasets. We use the FID score (Heusel et al., 2017) (lower is better), a commonly-used sample quality metric, to measure performance on the MNIST, FMNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), CIFAR-10, and CIFAR-100 (Krizhevsky et al., 2009) datasets. We can see that clustered VAEs (indicated as "C-VAE (classes)") consistently outperform VAEs, strongly supporting our hypothesis that $supp(\mathbb{P}^*)$ is indeed not connected in these datasets. We highlight once again that clustered VAEs were given the exact same computational budget as VAEs (both in terms of floating point operations and memory usage) to ensure fair comparisons. Nonetheless, clustered VAEs do have more parameters. In order to ensure that the added parameters are not the reason that clustered VAEs perform better, we perform an ablation, where we train a clustered VAE but use random labels instead of class labels (indicated as "C-VAE (random)"). We can see that clustered VAEs using labels are still the best performing option, once again providing strong empirical evidence to the claim that one can indeed think of classes of these datasets as different connected components $\mathcal{M}_\ell$ of $supp(\mathbb{P}^*)$. Finally, we point out that these results are not exclusive to VAEs: we carried out the same comparisons using other pushforward models, and found the conclusions to be the same, further backing our claims. Due to space constraints, we present these results in Tables 3, 4, and 5 for GANs, NFs, and WAEs, respectively, in Appendix C.1.

3.2 Verifying the hypothesis: Varying intrinsic dimensions

So far we have empirically confirmed that commonly-used image datasets have disconnected supports and that class labels provide a good proxy for the corresponding connected components. We now confirm that these disconnected components have varying intrinsic dimensions. We achieve this by applying (2) to each $\mathcal{D}_\ell$ (the subset of the data corresponding to the ℓth class) separately: observing similar estimates across different values of ℓ would support the manifold hypothesis, whereas observing different values would instead support the union of manifolds hypothesis.

Figure 3 shows, for different values of the hyperparameter k, boxplots of the values $\hat{d}_k^{(\ell)}$ for class labels $\ell = 1, \ldots, L$ in our considered datasets: MNIST, FMNIST, SVHN, CIFAR-10, CIFAR-100, and ImageNet (Russakovsky et al., 2015). Two relevant patterns emerge. First, within each dataset, results are mostly consistent across different choices of k, so that any conclusions we draw are not caused by a specific choice of this hyperparameter. Second, for all datasets except SVHN, we can observe a wide range of estimated intrinsic dimensions. These results empirically verify that the union of manifolds hypothesis is a more appropriate way to think about images than the manifold hypothesis. We highlight that (2) is known to underestimate the true intrinsic dimension when the dataset is not large enough, which might in principle affect our estimates on CIFAR-100 and ImageNet, since these datasets have fewer datapoints per class. Strong consistency across k on CIFAR-100 however suggests this is not a concern for this dataset; and even if the estimates we obtained on ImageNet do exhibit slightly more variation across k, the overall range of estimated intrinsic dimensions remains consistent across k. In other words, these results still strongly support the union of manifolds hypothesis. We take a more granular look into these estimates in Appendix C.2, where we show further choices of k as well as the intrinsic dimensions of individual classes.

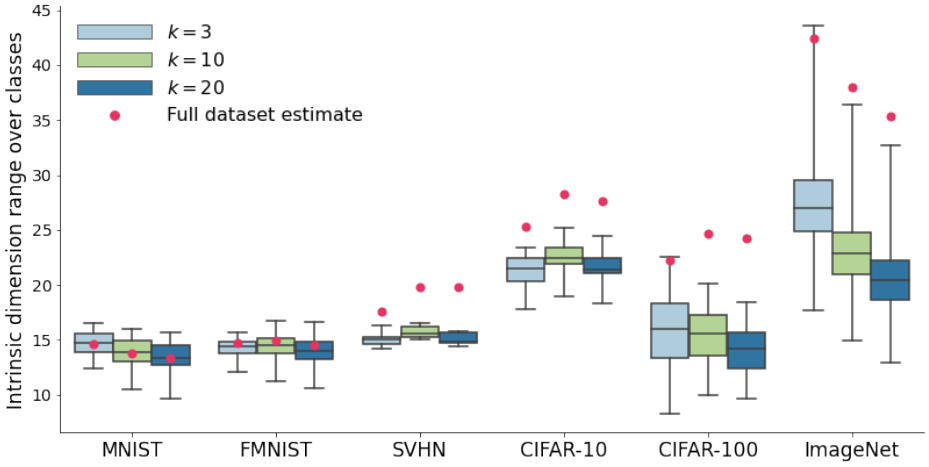

Figure 3: Boxplots showing variability of intrinsic dimension estimates across classes.

4 Exploiting the union of manifolds hypothesis

Now that we have verified the union of manifolds hypothesis – both the disconnectedness of the support of the data and its varying intrinsic dimensions – we turn our attention to the benefits brought to deep learning by being aware of the union of manifolds structure present in observed data. We see the results in this section as a sensible and promising first step towards realizing these benefits, and hope to encourage the community to further explore them.

4.1 Supervised learning: Classification accuracy and intrinsic dimension

Pope et al. (2021) showed that *datasets* of high intrinsic dimension are harder to classify. Here we provide a more granular look, by showing that *classes* of higher intrinsic dimension are harder to classify. We train 3 classifiers on CIFAR-100 (see Appendix D.2 for details): a VGG-19 (Simonyan & Zisserman, 2015), a ResNet-18, and a ResNet-34 (He et al., 2016). We focused on CIFAR-100 here as the other datasets considered in this work were either too simple to classify (MNIST, FMNIST, SVHN, CIFAR-10), or produced less-reliable intrinsic dimension estimates (ImageNet). For each class, we compute its classification accuracy and plot this against its estimated intrinsic dimension in Figure 4. We can see that, consistently across classifiers, there is an inverse relationship between estimated intrinsic dimension and classification accuracy. We also quantitatively compare intrinsic dimension and classification accuracy by computing their correlation, along with a p-value for independence with a t-test: we find that the negative correlation is significant (see figure caption). In other words, the higher the intrinsic dimension of a class, the harder it is to classify. While clearly intrinsic dimension does not fully explain accuracy, this correlation suggests the following intuition: learning useful representations for classes with higher intrinsic dimension requires learning more factors of variation and is thus more challenging.

Table 2: Means and standard errors of ResNet-18 accuracy on CIFAR-100 across 5 runs.

Weights	Test accuracy
Standard	$61.38\% \pm 0.17\%$
Proportional to intrinsic dimension	$\mathbf{61.77\% \pm 0.20\%}$

In order to check if this insight can help improve classifiers, we train the ResNet-18 in two different ways (see Appendix D.2 for experimental setup): in the first, we use the standard cross entropy loss, and in the second, we weight the terms corresponding to each class in the cross entropy loss in a manner proportional to their intrinsic dimension. The idea behind this reweighting of the loss is simply to focus more on classes of higher intrinsic dimension, as we have just shown these are harder to classify. Results are shown in Table 2, where we can see that this very simple change to the cross entropy loss marginally (but significantly, as error bars do not overlap) improves the accuracy of the network, providing an essentially "free" improvement, given the low computational overhead of

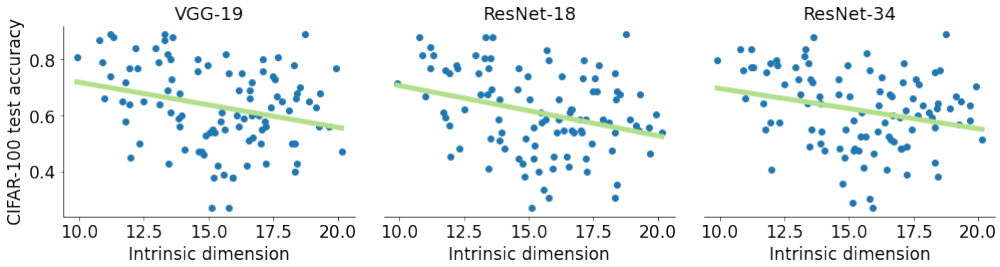

Figure 4: Class intrinsic dimension versus test accuracy on CIFAR-100, along with a least-squares regression line. Correlation coefficients for each model are -0.243 ± 0.015, -0.269 ± 0.012, and -0.274 ± 0.007, respectively (means and standard errors over 5 runs). Corresponding p-values, 0.008 ± 0.002, 0.019 ± 0.008, and 0.006 ± 0.001, show the relationship is significant.

estimating intrinsic dimension. Note that we did not perform any data augmentation scheme so as to not affect the intrinsic dimension of the data.

4.2 Unsupervised learning: Clustered deep generative models

In subsection 3.1 we used clustered DGMs to show that the support of image datasets is disconnected. Here we show that despite their simplicity, clustered DGMs are performant models which provide improvements over competing alternatives. We highlight that the aim of these experiments is not to achieve state-of-the-art performance, but rather to show that clustered DGMs outperform comparable non-clustered models, emphasizing the relevance of properly accounting for the union of manifolds structure present in data. To this end, we introduce a fully-unsupervised version of clustered DGMs, where we run a clustering algorithm to partition the dataset instead of using class labels. The procedure is summarized in Algorithm 1. The idea behind this modification is simply that if class labels are unavailable, a clustering algorithm might manage to recover connected components of the data support as clusters, at least approximately. In practice, we use agglomerative clustering (Rokach & Maimon, 2005) with Ward's criterion (Ward Jr, 1963) (see Appendix B.1 for details on this clustering method). We note that the clustered models we used in subsection 3.1 are an instance of Algorithm 1 where `clustering_algorithm` partitions the dataset based on known class labels.

Algorithm 1: Training of clustered DGMs

Input: `clustering_algorithm`$(\cdot)$, $\mathcal{D}$
Output: $\{(\mathbb{P}_Z^{(\ell)}, G_\ell)\}_{\ell=1}^L$

$\mathcal{D}_1, \ldots, \mathcal{D}_L \leftarrow$ `clustering_algorithm`$(\mathcal{D})$
for $\ell = 1$ **to** L **do**
  Initialize $\mathbb{P}_Z^{(\ell)}$ and G_ℓ
  Train G_ℓ and potentially $\mathbb{P}_Z^{(\ell)}$ on $\mathcal{D}_\ell$
end

We highlight once again that we use the same computational budget for training clustered models as for their standard counterparts. The only computational overhead is running the clustering algorithm itself – and more disk space is required to store the model – but the number of floating point operations and memory usage throughout training is identical (see Appendix B.1). All experimental details relevant to this section are included in Appendix D.1.

4.2.1 Better modelling of disconnected supports

As we already showed in subsection 3.1, C-VAEs can outperform standard VAEs. The comparisons we have discussed so far were however meant to confirm the union of manifolds hypothesis, and remain unfair as comparisons of generative performance since the C-VAE (classes) model has access to class labels. The bottom half of Table 1 includes additional comparisons to show that C-VAEs are not just useful as a way of confirming the disconnectedness of the data support, but that they are also an effective way of accounting for this structure. We include comparisons against a conditional VAE (Sohn et al., 2015), which also has access to class labels (indicated as "Conditional VAE (classes)"), and we can see that our C-VAE (classes) performs better on most cases. We also show results for the fully-unsupervised version of the C-VAE (indicated as "C-VAE (clusters)"), and we can see that it

not only outperforms the standard VAE, but that it also outperforms the same conditional VAE, now conditioned on cluster membership rather than class labels for a fully-unsupervised baseline (indicated as "Conditional VAE (clusters)"). Finally, we also include a comparison against a VAE whose prior $\mathbb{P}_Z$ is given by a mixture of L Gaussians (indicated as "VAE (GMM)") instead of a standard Gaussian as in the other models. Having a multimodal $\mathbb{P}_Z$ is a popular strategy in pushforward models to better model multimodal target distributions (Nalisnick et al., 2016; Dilokthanakul et al., 2017; Jiang et al., 2017; Ben-Yosef & Weinshall, 2018; Izmailov et al., 2020; Cao et al., 2020; Śmieja et al., 2020; Potapczynski et al., 2020). We can see that our C-VAE models outperform VAE (GMM) in most cases: this is not too surprising in light of Proposition 1, since the support of $\mathbb{P}_Z$ remains connected when using mixtures, even if it is now multimodal. Once again, we omit comparisons with other pushforward models here due to space constraints, but highlight that Appendix C.1 includes analogous comparisons with GANs, NFs, and WAEs, in Tables 3, 4, and 5, respectively, where we see the exact same behaviour of clustered DGMs outperforming or remaining competitive with naïve conditioning and using mixtures as base distributions.

4.2.2 Better modelling of varying intrinsic dimensions

While so far we have only focused on the ability of clustered DGMs to model disconnected supports, our method also easily allows modelling varying intrinsic dimensions, thus fully accounting for the union of manifolds hypothesis. In order to do this, we simply set different dimensions for the latent spaces $\mathcal{Z}_\ell$ of each of the L DGMs. In particular, we use (2) to obtain an estimate $\hat{d}_k^{(\ell)}$ of the intrinsic dimension of $\mathcal{M}_\ell$ using $\mathcal{D}_\ell$, and then set $\mathcal{Z}_\ell = \mathbb{R}^{\hat{d}_k^{(\ell)}}$. Any observed improvement of clustered models with varying $\hat{d}_k^{(\ell)}$s over clustered models with fixed $\hat{d}_k^{(\ell)}$s can thus be attributed to having accounted for varying intrinsic dimensions. For each of the L DGMs $(\mathbb{P}_Z^{(\ell)}, G_\ell)$, we use two-step models (Dai & Wipf, 2019; Loaiza-Ganem et al., 2022), which were specifically designed to properly model manifold-supported data by avoiding manifold overfitting: we first train an autoencoder-type model as a first step, whose decoder is G_ℓ; and then train a DGM $\mathbb{P}_Z^{(\ell)}$ on the low-dimensional representations obtained by running $\mathcal{D}_\ell$ through the encoder. In other words, we use a clustered version of two-step models as a way to account for varying intrinsic dimensions. The fourth panel of Figure 2 illustrates the benefits of this approach by training a clustered two-step VAE (indicated as "C-VAE+VAE"). This model is trained by clustering the data to obtain its connected components, estimating the respective intrinsic dimensions as 2 and 1, and then training a VAE+VAE model on each of these clusters. In the first cluster (of intrinsic dimension 2), the first VAE obtains 2-dimensional representations, and the second VAE learns the distribution of these representations. The same is done for the second cluster, except the first VAE obtains 1-dimensional representations.

In Table 6 in Appendix C.1, we carry out an analogous comparison on image datasets using VAE+NF models. Surprisingly, we found that while both versions of clustered models (i.e. with fixed or varying intrinsic dimensions) outperformed non-clustered two-step models – once again confirming the conclusions about better modelling of disconnected supports – setting intrinsic dimensions differently on every cluster did not improve performance over keeping intrinsic dimension constant across clusters. At a first glance this might appear to imply that there is no benefit to accounting for varying intrinsic dimensions, seemingly contradicting the theoretical result of Loaiza-Ganem et al. (2022, Theorem 1), which states that manifold overfitting will occur whenever intrinsic dimension is overspecified. However, upon closer inspection of the proof of Theorem 1 in Loaiza-Ganem et al. (2022), we can see that the rate at which manifold overfitting happens depends on the difference between ambient and intrinsic dimension.[3] This observation explains both the good empirical results of Loaiza-Ganem et al. (2022) and our aforementioned results: the former are driven by the difference between ambient and intrinsic dimension (on the order of hundreds/thousands), while the latter by the error between the true and estimated intrinsic dimension (on the order of ones/tens). While a partially negative result, we provide further insight into the connection between the low-dimensional structure of the data and DGMs.

In order to verify that properly setting varying intrinsic dimensions in DGMs is advantageous, provided the true intrinsic dimensions are different enough, we generate a synthetic dataset using a pretrained BigGAN (Brock et al., 2019). We use this model, which has $\mathcal{Z} = \mathbb{R}^{120}$, to generate 64,000 samples from the golden retriever class.

[3]For positive $\sigma \to 0$, Loaiza-Ganem et al. (2022) show that the likelihood of a D-dimensional model can go to infinity at a rate of σ^{d-D}, while not recovering $\mathbb{P}^*$, if the data is supported on a d-dimensional manifold.

These samples have true intrinsic dimension at most 120, and their estimated intrinsic dimension through (2) is 22 (we rounded the real-valued estimate up). We also generate another 64,000 samples from this model, except we zero out all but $m = 1$ latent coordinates before passing them through the generator, resulting in samples of true intrinsic dimension of at most m, and estimated intrinsic dimension 3. We then combine these two types of samples into a single dataset of size 128,000, with each type of sample corresponding to a class. This dataset has the property that its classes have a large difference in intrinsic dimension, and estimating the intrinsic dimension of the entire dataset while ignoring classes gives a value of 5. We then train two models on this dataset: a clustered two-step model, where the first models are WAEs with fixed intrinsic dimension set to 5 (i.e. the single estimate of intrinsic dimension over the entire dataset) and the second models NFs (indicated as "C-WAE+NF (constant $\hat{d}$)"); and a clustered version of the same model except clusters have their latent dimensions set to 22 and 3 (i.e. class intrinsic dimension estimates), respectively (indicated as "C-WAE+NF"). We then repeat this entire process for $m = 3$ and $m = 5$, and show the resulting FID scores in Figure 5 for 3 runs. We can see that setting the latent dimension of each cluster to its estimated intrinsic dimension results in a large performance improvement versus setting it without accounting for varying intrinsic dimensions. We can also see that this gap in performance tightens as the difference in true intrinsic dimensions decreases. These results show that considering varying intrinsic dimensions is practically relevant in DGMs, particularly when the difference in true intrinsic dimensions is large.

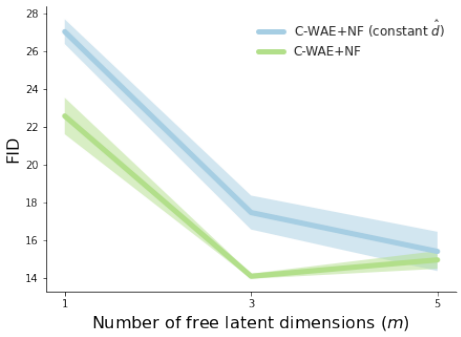

Figure 5: Mean FID scores for 3 runs on golden retriever data, the shaded areas show standard errors.

5 Conclusions, Limitations, and Future Work

Conclusions In this work, we have proposed the *union of manifolds hypothesis*, which extends the standard manifold hypothesis (Bengio et al., 2013) to accommodate data of varying intrinsic dimensions. We first confirm this hypothesis by showing both that datasets have disconnected supports, and that the intrinsic dimension on the respective connected components can vary widely. We also establish uses of the union of manifolds hypothesis, showing that classes of higher intrinsic dimension are harder to classify and how this insight can be used to improve classification accuracy. Finally, we also propose clustered DGMs, which in addition to being a useful tool in the verification of our main hypothesis, are also a promising direction for DGM research. We anticipate that the broader deep learning community will further unlock the potential of the union of manifolds hypothesis for both understanding natural data and discovering empirical improvements.

Limitations and future work Although we confirmed the union of manifolds hypothesis on commonly-used datasets, we have not tested on anything besides image data; verification of the union of manifolds hypothesis and the efficacy of clustered DGMs on other types of data remain to be proven. We have reason to believe our improvements will generalize though, as has often been the case with other DGM innovations developed on image-like data.

Another point to note is that, while we did obtain classification accuracy improvements through the union of manifolds hypothesis, these were marginal. Also, clustered DGMs do not share parameters nor representations, and are thus somewhat limited in how they extract information from the entire dataset. Although the simplicity of clustered DGMs makes them ideal to verify the union of manifolds hypothesis, they can likely be significantly further improved upon as DGMs. Finally, we treated the number of clusters in clustered DGMs as known *a priori*, by setting this value to the number of classes. For a fully unsupervised approach, the number of clusters could be learned (Salvador & Chan, 2004; Kulis & Jordan, 2012; Villecroze et al., 2022).

REFERENCES

Maryam Abdolali and Nicolas Gillis. Beyond linear subspace clustering: A comparative study of nonlinear manifold clustering algorithms. *Computer Science Review*, 42:100435, nov 2021. doi: 10.1016/j.cosrev.2021.100435. URL https://doi.org/10.1016%2Fj.cosrev.2021.100435.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *ICLR*, 2021.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning*, pp. 224–232. PMLR, 2017.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2007.

Ershad Banijamali, Ali Ghodsi, and Pascal Popuart. Generative mixture of networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3753–3760. IEEE, 2017.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2019.

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.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Adam Block, Zeyu Jia, Yury Polyanskiy, and Alexander Rakhlin. Intrinsic dimension estimation. *arXiv preprint arXiv:2106.04018*, 2021.

Vladimir Igorevich Bogachev. *Measure theory*, volume 2. Springer, 2007.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33:442–453, 2020.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *ICLR*, 2019.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Theophilos Cacoullos. Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics*, 18(1):179–189, 1966.

Lele Cao, Sahar Asadi, Wenfei Zhu, Christian Schmidli, and Michael Sjöberg. Simple, scalable, and stable variational deep clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 108–124. Springer, 2020.

Anthony L Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P Cunningham. Rectangular flows for manifold learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Ricky T. Q. Chen, Jens Behrmann, David K Duvenaud, and Joern-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International conference on machine learning*, pp. 2133–2143. PMLR, 2020.

Edmond Cunningham, Adam Cobb, and Susmit Jha. Principal manifold flows. *arXiv preprint arXiv:2202.07037*, 2022.

Bin Dai and David Wipf. Diagnosing and enhancing VAE models. *ICLR*, 2019.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modeling. *arXiv preprint arXiv:2202.02763*, 2022.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *ICLR*, 2017.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *ICLR*, 2017.

Laurent Dinh, Jascha Sohl-Dickstein, Hugo Larochelle, and Razvan Pascanu. A RAD approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

Leo L Duan. Transport Monte Carlo: High-accuracy posterior approximation via random transport. *Journal of the American Statistical Association*, (just-accepted):1–30, 2021.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural Spline Flows. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Ehsan Elhamifar and René Vidal. Sparse manifold clustering and embedding. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper/2011/file/fc490ca45c00b1249bbe3554a4fdf6fb-Paper.pdf`.

Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications, 2012. URL `https://arxiv.org/abs/1203.1005`.

Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Pavel Izmailov, Polina Kirichenko, Marc Finzi, and Andrew Gordon Wilson. Semi-supervised learning with normalizing flows. In *International Conference on Machine Learning*, pp. 4615–4630. PMLR, 2020.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1965–1972, 2017.

Mahyar Khayatkhoei, Maneesh K Singh, and Ahmed Elgammal. Disconnected manifold learning for generative adversarial networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible $1\times$ 1 Convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *ICLR*, 2014.

Konik Kothari, AmirEhsan Khorashadizadeh, Maarten de Hoop, and Ivan Dokmanić. Trumpets: Injective flows for inference and inverse problems. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161, pp. 1269–1278, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Brian Kulis and Michael I Jordan. Revisiting k-means: new algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1131–1138, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004.

Uzu Lim, Vidit Nanda, and Harald Oberhauser. Tangent space and dimension estimation with the wasserstein distance. *arXiv preprint arXiv:2110.06357*, 2021.

Gabriel Loaiza-Ganem, Brendan Leigh Ross, Jesse C Cresswell, and Anthony L. Caterini. Diagnosing and fixing manifold overfitting in deep generative models. *Transactions on Machine Learning Research*, 2022. URL `https://openreview.net/forum?id=0nEZCVshxS`.

Francesco Locatello, Damien Vincent, Ilya Tolstikhin, Gunnar Rätsch, Sylvain Gelly, and Bernhard Schölkopf. Competitive training of mixtures of independent deep generative models. *arXiv preprint arXiv:1804.11130*, 2018.

Lorenzo Luzi, Randall Balestriero, and Richard G Baraniuk. Ensembles of generative adversarial networks for disconnected data. *arXiv preprint arXiv:2006.14600*, 2020.

David JC MacKay and Zoubin Ghahramani. Comments on'maximum likelihood estimation of intrinsic dimension'by e. levina and p. bickel (2004). *The Inference Group Website, Cavendish Laboratory, Cambridge University*, 2005.

Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, pp. 131, 2016.

Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. *Advances in neural information processing systems*, 23, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Arkadas Ozakin and Alexander Gray. Submanifold density estimation. *Advances in Neural Information Processing Systems*, 22, 2009.

Guilherme GP Freitas Pires and Mário AT Figueiredo. Variational mixture of normalizing flows. In *ESANN*, 2020.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021.

Andres Potapczynski, Gabriel Loaiza-Ganem, and John P Cunningham. Invertible gaussian reparameterization: Revisiting the gumbel-softmax. *Advances in Neural Information Processing Systems*, 33:12311–12321, 2020.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Danilo Jimenez Rezende, George Papamakarios, Sébastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pp. 8083–8092. PMLR, 2020.

Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pp. 321–352. Springer, 2005.

Brendan Leigh Ross and Jesse C Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Brendan Leigh Ross, Gabriel Loaiza-Ganem, Anthony L Caterini, and Jesse C Cresswell. Neural implicit manifold learning for topology-aware generative modelling. *arXiv preprint arXiv:2206.11267*, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Antoine Salmona, Valentin de Bortoli, Julie Delon, and Agnès Desolneux. Can push-forward generative models fit multimodal distributions? *arXiv preprint arXiv:2206.14476*, 2022.

Stan Salvador and Philip Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *16th IEEE international conference on tools with artificial intelligence*, pp. 576–584. IEEE, 2004.

René L Schilling and Franziska Kühn. *Counterexamples in Measure and Integration*. Cambridge University Press, 2021.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

Marek Śmieja, Maciej Wołczyk, Jacek Tabor, and Bernhard C Geiger. Segma: Semi-supervised gaussian mixture autoencoder. *IEEE transactions on neural networks and learning systems*, 32(9): 3930–3941, 2020.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Ugo Tanielian, Thibaut Issenhuth, Elvis Dohmatob, and Jeremie Mary. Learning disconnected manifolds: a no gan's land. In *International Conference on Machine Learning*, pp. 9418–9427. PMLR, 2020.

Piotr Tempczyk, Rafał Michaluk, Lukasz Garncarek, Przemysław Spurek, Jacek Tabor, and Adam Golinski. Lidl: Local intrinsic dimension estimation using approximate likelihood. In *International Conference on Machine Learning*, pp. 21205–21231. PMLR, 2022.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *ICLR*, 2018.

Valentin Villecroze, Harry J Braviner, Panteha Naderian, Chris J Maddison, and Gabriel Loaiza-Ganem. Bayesian nonparametrics for offline skill discovery. *arXiv preprint arXiv:2202.04675*, 2022.

Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Fei Ye and Adrian G Bors. Lifelong mixture of variational autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

A Proof of Proposition 1

Before giving the standard definition of support, we motivate it. Intuitively, the support of a distribution $\mathbb{P}$ is "the smallest set" to which $\mathbb{P}$ assigns probability 1. The idea of "the smallest set" satisfying a property is often formalized by taking the intersection of all sets satisfying the property. Naïve application of this principle however results in a "definition" of support with undesirable properties. Consider for example the case where $\mathbb{P}$ is a standard Gaussian distribution on $\mathbb{R}$. Clearly any definition of support that we use should obey that the support of $\mathbb{P}$ is $\mathbb{R}$. However, if we simply follow the "definition" of support as the intersection of all sets of probability 1, we would not obtain this: for any $x \in \mathbb{R}$, clearly $\mathbb{P}(\mathbb{R} \setminus \{x\}) = 1$, yet $\bigcap_{x \in \mathbb{R}} \mathbb{R} \setminus \{x\} = \emptyset$, where $\emptyset$ denotes the empty set. In other words, arbitrary intersections of sets of probability 1 need not have probability 1, and in fact can have probability 0. More abstractly, sets of probability 1 are not closed under arbitrary intersections, which is desirable when defining "the smallest set" satisfying a given property as the intersection of all such sets. In order to solve this problem, the support is defined as the intersection of all closed sets having probability 1 (closed sets are indeed closed under intersections).

Definition 1 (Support of probability distributions): Let $\mathcal{Z}$ be a topological space. Equipping $\mathcal{Z}$ with its Borel σ-algebra to make it a measurable space, let $\mathbb{P}$ be a probability distribution on $\mathcal{Z}$. Let $\mathcal{C}(\mathbb{P})$ be the collection of all closed sets C in $\mathcal{Z}$ such that $\mathbb{P}(C) = 1$. The support of $\mathbb{P}$, denoted as $supp(\mathbb{P})$, is defined as:

$$supp(\mathbb{P}) = \bigcap_{C \in \mathcal{C}(\mathbb{P})} C. \tag{5}$$

Before continuing, we point out that there exist counterexamples showing that the support of a distribution need not have probability 1 (Schilling & Kühn, 2021, Examples 6.2 and 6.3). In other words, even though closed sets are closed under intersections, closed sets of probability 1 need not share this property. These examples are however much more contrived than the one above and of no practical interest (if $\mathcal{Z}$ is a separable metric space, as is the case of $\mathbb{R}^d$, then any Borel measure $\mathbb{P}$ on $\mathcal{Z}$ satisfies $\mathbb{P}(supp(\mathbb{P})) = 1$, see Bogachev (2007, Proposition 7.2.9)), and thus support is commonly defined as above. Note that the assumption that $\mathbb{P}_Z(supp(\mathbb{P}_Z)) = 1$ could be replaced by the requirement that $\mathcal{Z}$ be a separable metric space and $\mathbb{P}_Z$ a Borel measure on it (which covers all settings of practical interest), although this would yield a slightly less general result. We restate Proposition 1 below for convenience:

Proposition 1: Let $\mathcal{Z}$ and $\mathcal{X}$ be topological spaces and $G : \mathcal{Z} \to \mathcal{X}$ be continuous. Considering $\mathcal{Z}$ and $\mathcal{X}$ as measurable spaces with their respective Borel σ-algebras, let $\mathbb{P}_Z$ be a probability measure on $\mathcal{Z}$ such that $supp(\mathbb{P}_Z)$ is connected and $\mathbb{P}_Z(supp(\mathbb{P}_Z)) = 1$. Then $supp(G_\#\mathbb{P}_Z)$ is connected.

Proof: As mentioned in the proof sketch in the main manuscript, showing that $supp(G_\#\mathbb{P}_Z) = cl(G(supp(\mathbb{P}_Z)))$, where $cl(\cdot)$ denotes closure in $\mathcal{X}$, is enough, since by continuity of G, $G(supp(\mathbb{P}_Z))$ is connected, and the closure of connected sets is itself connected.

We begin by verifying that $cl(G(supp(\mathbb{P}_Z))) \subseteq supp(G_\#\mathbb{P}_Z)$. Let $C \in \mathcal{C}(G_\#\mathbb{P}_Z)$. By definition of $\mathcal{C}(G_\#\mathbb{P}_Z)$, C is closed in $\mathcal{X}$ and is such that $G_\#\mathbb{P}_Z(C) = 1$. By definition of pushforward measure, it follows that:

$$\mathbb{P}_Z(G^{-1}(C)) = G_\#\mathbb{P}_Z(C) = 1, \tag{6}$$

and by continuity of G, it also follows that $G^{-1}(C)$ is closed in $\mathcal{Z}$. Then, $G^{-1}(C) \in \mathcal{C}(\mathbb{P}_Z)$, and by definition of support, $supp(\mathbb{P}_Z) \subseteq G^{-1}(C)$. It then follows that $G(supp(\mathbb{P}_Z)) \subseteq G(G^{-1}(C)) \subseteq C$. Since this holds for every $C \in \mathcal{C}(G_\#\mathbb{P}_Z)$, we have that:

$$G(supp(\mathbb{P}_Z)) \subseteq \bigcap_{C \in \mathcal{C}(G_\#\mathbb{P}_Z)} C = supp(G_\#\mathbb{P}_Z). \tag{7}$$

Applying $cl(\cdot)$ on both sides, and using the fact that $supp(G_{\#}\mathbb{P}_Z)$ is closed in $\mathcal{X}$ (since it is an intersection of closed sets), yields $cl(G(supp(\mathbb{P}_Z))) \subseteq supp(G_{\#}\mathbb{P}_Z)$.

It now only remains to show that $supp(G_{\#}\mathbb{P}_Z) \subseteq cl(G(supp(\mathbb{P}_Z)))$. By definition of pushforward measure, we have that:

$$G_{\#}\mathbb{P}_Z(G(supp(\mathbb{P}_Z))) = \mathbb{P}_Z(G^{-1}(G(supp(\mathbb{P}_Z)))) \geq \mathbb{P}_Z(supp(\mathbb{P}_Z)) = 1, \tag{8}$$

where the inequality follows from $supp(\mathbb{P}_Z) \subseteq G^{-1}(G(supp(\mathbb{P}_Z)))$. Since $G(supp(\mathbb{P}_Z)) \subseteq cl(G(supp(\mathbb{P}_Z)))$, then $G_{\#}\mathbb{P}_Z(cl(G(supp(\mathbb{P}_Z)))) = 1$. It follows that $cl(G(supp(\mathbb{P}_Z))) \in \mathcal{C}(G_{\#}\mathbb{P}_Z)$, and thus $supp(G_{\#}\mathbb{P}_Z) \subseteq cl(G(supp(\mathbb{P}_Z)))$ by definition of support.

□

B Clustered DGMs: Details

B.1 Training clustered DGMs

Training cost As mentioned in the main manuscript, we use the exact same computational budget to train a clustered DGM $\{(\mathbb{P}_Z^{(\ell)}, G_\ell)\}_{\ell=1}^L$ as we do to train an equivalent non-clustered baseline $(\mathbb{P}_Z, G)$. We begin with the observation that due to the serial nature of Algorithm 1, all L models need not be in memory when training a clustered DGM: once the ℓth model has been trained, it can be moved to disk before loading the $(\ell+1)^{th}$ model to memory. In other words, even though clustered models do have more parameters than their non-clustered counterparts and thus require more disk space to be stored, it is not needed at any point during training to load more than a single model in memory. We also make the key observation that, if using the same architecture throughout (i.e. G and every G_ℓ have the same architecture, as do $\mathbb{P}_Z$ and $\mathbb{P}_Z^{(\ell)}$ if they are trainable), then training the non-clustered model $(\mathbb{P}_Z, G)$ for N epochs on $\mathcal{D}$ requires the same amount of compute as training each $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ for N epochs, since every $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ is trained on a smaller dataset $\mathcal{D}_\ell$. To see this in more detail, assume training $(\mathbb{P}_Z, G)$ for a single epoch requires T gradient steps, so the total training cost is TN gradient steps. In order to train a single $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ model for a single epoch, since $\mathcal{D}_\ell$ is $|\mathcal{D}|/|\mathcal{D}_\ell|$ times smaller than $\mathcal{D}$, only $|\mathcal{D}_\ell|/|\mathcal{D}|T$ gradient steps are required. Thus, training a single $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ model for N epochs requires $|\mathcal{D}_\ell|/|\mathcal{D}|TN$ gradient steps, and it follows that training all the $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ for $\ell = 1, \ldots, L$ requires

$$\sum_{\ell=1}^{L} \frac{|\mathcal{D}_\ell|}{|\mathcal{D}|} TN = \frac{TN}{|\mathcal{D}|} \sum_{\ell=1}^{L} |\mathcal{D}_\ell| = \frac{TN}{|\mathcal{D}|} |\mathcal{D}| = TN \tag{9}$$

gradient steps. Finally, since exactly the same number of steps are required to train clustered and non-clustered models, and these models share architectures, it follows that the FLOP and memory costs of training them are indeed equivalent under our setup.

Clustering algorithm In order to generate the dataset clusters to train each component in our clustered models (with the goal of approximating the connected components of the full dataset) without labels, we perform agglomerative clustering on the dataset. At the start of the algorithm, each datapoint is in its own cluster. Then, at every step, the two clusters with the smallest linkage distance are merged, decreasing the number of clusters by 1. This occurs until a pre-specified number of clusters, L, is reached. For all of our datasets, we set $L = 10$ which is the number of classes for the MNIST, FMNIST, SVHN, and CIFAR-10 datasets. We experimented with ranges of L between 7 and 15 and did not find a large variation in performance. Automatically inferring L from a dataset is the subject of future work and would make our clustering models completely unsupervised. For all of our experiments involving agglomerative clustering, we used Ward's linkage criterion (Ward Jr, 1963). The distance between two clusters with this criterion is the variance of the Euclidean distance between all datapoints in the clusters being merged. Therefore, at each step, the two clusters with the smallest variance will be combined. We experimented with using single linkage, which defines the distance between two clusters as the minimum distance between a datapoint in both clusters, but found that this led to very unbalanced cluster sizes and had very poor performance when training our DGMs. We also tried a custom linkage metric that merges the two clusters at each step to maximize the inter-cluster intrinsic dimension estimate variance, with the intuition that building clusters with a large variance in intrinsic dimension would better model the submanifolds of the dataset. In practice,

we found this method very sensitive to initialization and saw the same unbalanced cluster failure mode as the single linkage criterion.

For all of our experiments, we also tried using the `k-means++` (Arthur & Vassilvitskii, 2007) clustering algorithm. This also worked well but gave slightly worse results than agglomerative clustering with Ward's criterion across the board.

B.2 Sampling from clustered DGMs

Algorithm 2: Sampling of clustered DGMs

Input: m, trained clustered DGM $\{(\mathbb{P}_Z^{(\ell)}, G_\ell)\}_{\ell=1}^L$, and corresponding cluster sizes $|\mathcal{D}_1|, \dots, |\mathcal{D}_L|$.
Output: $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$
$(m_1, \dots, m_L) \sim \text{Multinomial}\left(m, \left(\frac{|\mathcal{D}_1|}{\sum_{\ell'=1}^L |\mathcal{D}_{\ell'}|}, \dots, \frac{|\mathcal{D}_L|}{\sum_{\ell'=1}^L |\mathcal{D}_{\ell'}|}\right)\right)
for $\ell = 1$ **to** L **do**
  for $t = 1$ **to** m_ℓ **do**
    $Z \sim \mathbb{P}_Z^{(\ell)}$
    $X = G_\ell(Z)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{X\}$
  end
end

As mentioned in the main manuscript, the sampling process described in (4) can be made memory-efficient. We now describe how to achieve this. Assume we want to generate m samples from a trained clustered DGM model $\{(\mathbb{P}_Z^{(\ell)}, G_\ell)\}_{\ell=1}^L$. The idea is to make sure that each of the L DGMs are only loaded into memory once and one at a time, which requires first knowing exactly how many samples will be required from each model. Note that, if we denote the number of samples coming from the ℓth model as m_ℓ, then clearly $(m_1, \dots, m_L)$ follows a multinomial distribution with parameters m and $(|\mathcal{D}_1|, \dots, |\mathcal{D}_L|)/\sum_{\ell'=1}^L |\mathcal{D}_{\ell'}|$. We can thus first sample from this multinomial, and then sample the appropriate number of times from each model, as described in Algorithm 2. Note that, similarly to Algorithm 1, the outer `for` loop in Algorithm 2 can be trivially made memory-efficient by loading the required model $(\mathbb{P}_Z^{(\ell)}, G_\ell)$ into memory and then removing it from memory once it has been sampled from m_ℓ times (and the inner `for` loop can be trivially parallelized).

C Additional results

C.1 Clustered DGMs

Tables 3, 4, and 5 show analogous results to those of Table 1 for GANs, NFs, and WAEs, respectively. We can see that, modulo a few exceptions, the same results observed for VAEs in the main manuscript also hold for these pushforward models, further confirming the disconnectedness of the support of the data, and that clustered DGMs are a performant class of DGMs.

Table 6 shows results of running a VAE+NF two-step model mentioned in subsubsection 4.2.2. We can see that the best performing clustered versions still mostly outperform the non-clustered version. This is consistent with our previous results and highlights once again the relevance of accounting for multiple connected components. As mentioned in the main manuscript though, there is no significant improvement between fixing the latent dimension of every cluster to its estimated intrinsic dimension, and simply fixing the intrinsic dimension of every cluster to the same value (the estimated intrinsic dimension of the entire dataset).

C.2 Intrinsic dimension estimates

We show a more granular breakdown of Figure 3, with intrinsic dimension estimate values for each class in Figure 6 for MNIST, Figure 7 for FMNIST, Figure 8 for SVHN, and Figure 9 for CIFAR-10.

Table 3: FID scores. We show means and standard errors across 3 runs.

Model	MNIST	FMNIST	SVHN	CIFAR-10	CIFAR-100
GAN	$\mathbf{14.4 \pm 3.0}$	$\mathbf{12.8 \pm 0.4}$	9.4 ± 0.2	11.2 ± 0.1	10.6 ± 0.0
C-GAN (random)	26.5 ± 1.2	20.2 ± 0.3	7.5 ± 0.0	12.1 ± 0.0	15.2 ± 0.1
C-GAN (classes)	29.2 ± 1.6	27.3 ± 0.8	$\mathbf{7.0 \pm 0.0}$	$\mathbf{10.6 \pm 0.0}$	13.5 ± 0.0
GAN (GMM)	$\mathbf{16.2 \pm 2.6}$	15.9 ± 2.7	8.8 ± 0.8	13.5 ± 0.3	12.8 ± 0.3
Conditional GAN (classes)	18.9 ± 0.4	$\mathbf{14.4 \pm 2.7}$	9.3 ± 0.1	$\mathbf{10.7 \pm 0.3}$	$\mathbf{10.0 \pm 0.3}$
Conditional GAN (clusters)	$\mathbf{16.9 \pm 2.5}$	$\mathbf{12.0 \pm 0.6}$	9.4 ± 0.0	11.4 ± 0.1	$\mathbf{10.0 \pm 0.1}$
C-GAN (clusters)	30.8 ± 0.7	28.9 ± 1.3	$\mathbf{7.0 \pm 0.0}$	$\mathbf{10.5 \pm 0.1}$	16.6 ± 0.2

Table 4: FID scores. We show means and standard errors across 3 runs.

Model	MNIST	FMNIST	SVHN	CIFAR-10	CIFAR-100
NF	10.3 ± 0.1	11.1 ± 0.1	5.0 ± 0.0	11.3 ± 0.0	11.2 ± 0.0
C-NF (random)	20.2 ± 0.0	18.0 ± 0.0	13.7 ± 0.1	18.2 ± 0.1	45.7 ± 19.2
C-NF (classes)	$\mathbf{8.1 \pm 0.0}$	$\mathbf{9.1 \pm 0.0}$	8.2 ± 0.1	$\mathbf{10.8 \pm 0.0}$	52.3 ± 2.5
NF (GMM)	10.0 ± 0.1	10.9 ± 0.1	5.0 ± 0.0	11.3 ± 0.0	11.1 ± 0.0
Conditional NF (classes)	13.2 ± 0.1	11.9 ± 0.1	$\mathbf{4.5 \pm 0.1}$	10.5 ± 0.0	$\mathbf{10.2 \pm 0.0}$
Conditional NF (clusters)	13.1 ± 0.1	12.2 ± 0.1	$\mathbf{4.5 \pm 0.0}$	10.6 ± 0.0	10.7 ± 0.1
C-NF (clusters)	9.2 ± 0.0	9.7 ± 0.0	29.4 ± 12.1	12.8 ± 0.1	12.7 ± 0.1

Table 5: FID scores. We show means and standard errors across 3 runs.

Model	MNIST	FMNIST	SVHN	CIFAR-10	CIFAR-100
WAE	5.3 ± 0.2	8.9 ± 0.4	6.9 ± 0.2	10.6 ± 0.0	9.5 ± 0.1
C-WAE (random)	5.2 ± 1.3	9.2 ± 0.1	7.2 ± 1.0	10.7 ± 0.0	12.1 ± 0.1
C-WAE (classes)	6.3 ± 1.5	$\mathbf{7.5 \pm 0.2}$	$\mathbf{6.4 \pm 0.6}$	$\mathbf{9.9 \pm 0.6}$	10.3 ± 0.1
WAE (GMM)	8.5 ± 0.4	9.1 ± 0.1	$\mathbf{6.1 \pm 0.1}$	13.1 ± 0.2	11.7 ± 0.0
Conditional WAE (classes)	$\mathbf{2.9 \pm 0.1}$	$\mathbf{7.4 \pm 0.1}$	$\mathbf{5.9 \pm 0.3}$	10.6 ± 0.0	11.6 ± 1.6
Conditional WAE (clusters)	$\mathbf{18.5 \pm 15.6}$	$\mathbf{7.6 \pm 0.1}$	$\mathbf{5.8 \pm 0.3}$	10.9 ± 0.1	21.1 ± 0.4
C-WAE (clusters)	6.1 ± 2.4	$\mathbf{7.3 \pm 0.2}$	$\mathbf{6.9 \pm 1.6}$	$\mathbf{10.6 \pm 0.8}$	$\mathbf{8.9 \pm 0.0}$

Table 6: FID scores. We show means and standard errors across 3 runs.

Model	MNIST	FMNIST	SVHN	CIFAR-10	CIFAR-100
VAE+NF	10.5 ± 0.1	12.4 ± 0.2	$\mathbf{5.5 \pm 0.0}$	11.2 ± 0.1	10.9 ± 0.1
C-VAE+NF (constant $\hat{d}$, classes)	$\mathbf{9.4 \pm 0.0}$	$\mathbf{8.7 \pm 0.0}$	$\mathbf{5.5 \pm 0.1}$	$\mathbf{10.2 \pm 0.0}$	11.1 ± 0.0
C-VAE+NF (constant $\hat{d}$, clusters)	$\mathbf{9.4 \pm 0.1}$	$\mathbf{8.8 \pm 0.1}$	9.0 ± 3.3	11.5 ± 0.9	$\mathbf{10.0 \pm 0.1}$
C-VAE+NF (classes)	9.5 ± 0.0	$\mathbf{8.8 \pm 0.1}$	$\mathbf{5.6 \pm 0.1}$	10.5 ± 0.1	11.4 ± 0.1
C-VAE+NF (clusters)	$\mathbf{9.4 \pm 0.0}$	$\mathbf{8.7 \pm 0.1}$	5.7 ± 0.1	13.2 ± 2.4	$\mathbf{9.9 \pm 0.0}$

Since CIFAR-100 and ImageNet have too many classes to show, we only include the top 5 highest and lowest intrinsic dimension classes in Figure 10 for CIFAR-100, and Figure 11 for ImageNet.

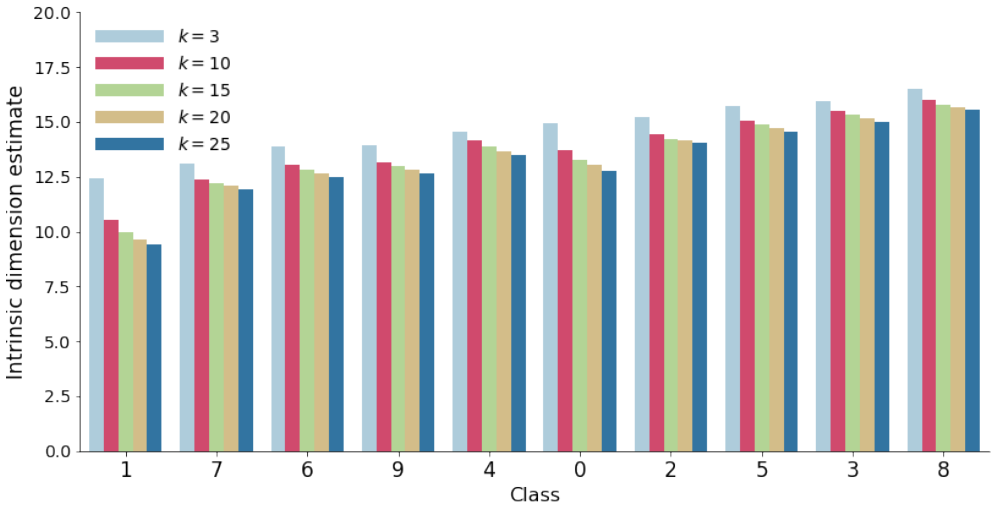

Figure 6: Intrinsic dimension estimates for classes in the MNIST dataset.

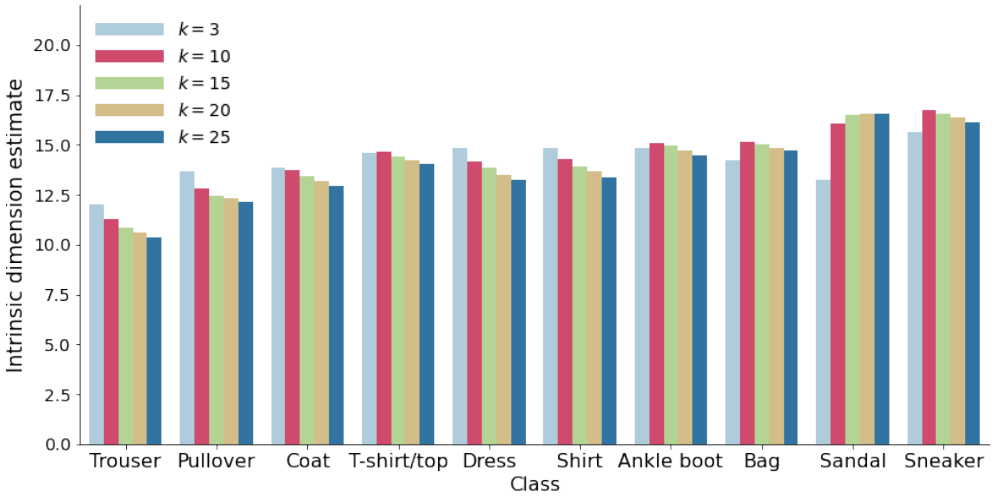

Figure 7: Intrinsic dimension estimates for classes in the FMINST dataset.

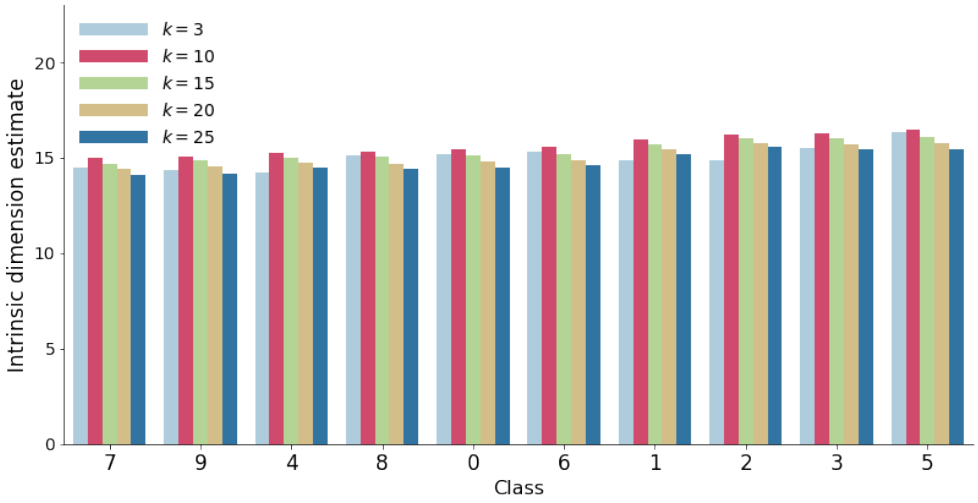

Figure 8: Intrinsic dimension estimates for classes in the SVHN dataset.

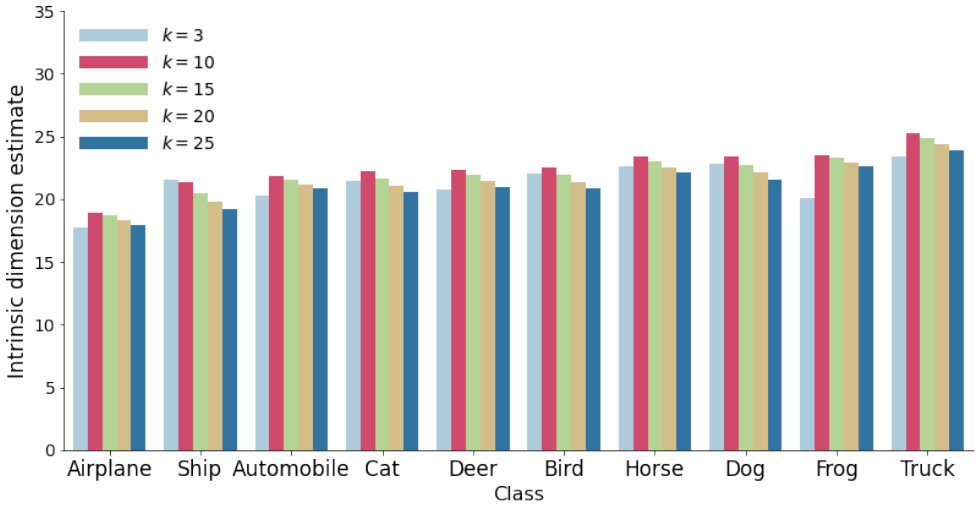

Figure 9: Intrinsic dimension estimates for classes in the CIFAR-10 dataset.

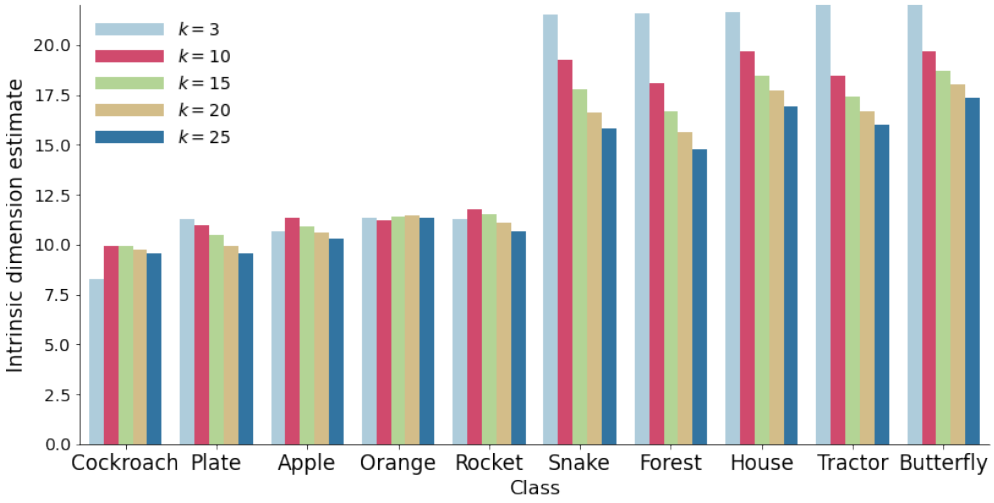

Figure 10: Intrinsic dimension estimates for classes in the CIFAR-100 dataset.

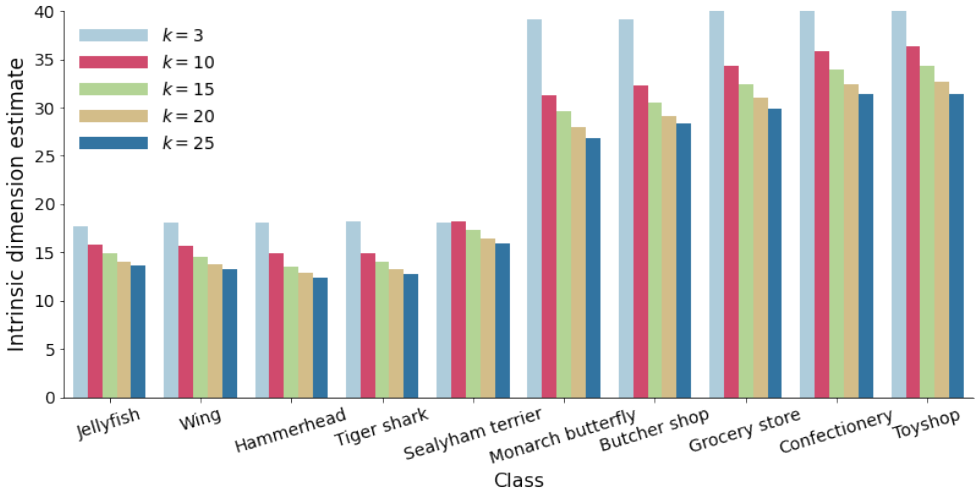

Figure 11: Intrinsic dimension estimates for classes in the ImageNet dataset.

D EXPERIMENTAL DETAILS

D.1 CLUSTERED DGMS

For all models, we randomly select 10% of the training dataset to be used for validation and train on the remaining 90%. We use a separate test dataset to report all metrics for our models. A batch size of 128 is used for all datasets. Unless otherwise noted, at the beginning of training, we scale all the data to between 0 and 1. For all experiments, we use the Adam optimizer, typically with learning rate 0.001 and cosine annealing for a maximum of 100 epochs. We also use gradient norm clipping with a value of 10.

Simulated Data To generate the ground truth data, we create a uniform mixture of two components. The first is a two dimensional square of width 1 centered at $(0, 0.8)$. The distribution along the x and y axis are independent beta distributions with (α, β) set to $(2, 2)$ for the y axis and $(2, 5)$ for the x axis. The second component is a one dimensional sinusoid (with y as the input and x as the output) that is one unit in length centred at 0.5 with an amplitude of 0.5 and a period of 1. The distribution along this line is a beta distribution with (α, β) set to $(2, 5)$. We generate $10{,}000$ samples for training, $10{,}000$ for validation and $5{,}000$ for testing. We train three models on this simulated data. For the baseline model we use a VAE with MLPs for the encoder and decoder, with ReLU activations. The encoder and decoder each have a single hidden layer with 25 units. The latent dimension of the model is set to 2. The learning rate is set to 0.0001. We do not use early stopping and train for 200 epochs. For the second model, we train a clustering VAE with two components trained using the same setup as the baseline model. For the third model we train a clustered VAE+VAE where each component is a two step VAE+VAE model. Both VAEs have the same architecture and are each trained the same way as the baseline model. The latent dimension of the GAE is obtained by running the MLE intrinsic dimension estimator 2 on each component.

VAEs For MNIST and FMNIST, we use MLPs for the encoder and decoder, with ReLU activations. The encoder and decoder have each a single hidden layer with 512 units. For SVHN, CIFAR-100 and CIFAR-10, we use convolutional networks. The encoder and decoder have 4 convolutional layers with $(32, 32, 16, 16)$ and $(16, 16, 32, 32)$ channels, respectively, followed by a flattening operation and a fully-connected layer. The convolutional networks also use ReLU activations, and have kernel size 3 and stride 1. The latent dimension of the model is set to 20. We do not use early stopping and always train for 100 epochs. For the GMM, we set the prior to have 10 modes with learnable mixture weights, means and standard deviations. The mixture components of the prior were initialized with fixed means spaced 3 units apart centred at 0, standard deviations of 1 and uniform mixture weights.

GANs For all datasets we use a 3-layer fully convolutional generator and discriminator. Each layer is composed of a convolution (transpose convolution for the discriminator), instance normalization and a leaky relu activation with slope set to 0.01. For the generator, the number of filters was set to 256, 128, 64 with kernel sizes of 4, 4, 4 and strides 2, 2, 2. The discriminator followed the same hyperparameters in reverse order. The generator had a final convolution to reduce the number of channels to that of the data followed by a Tanh activation. The discriminator had a final transpose convolution to reduce the number of channels to one. The latent dimension was set to 100. We trained the generator for one step every five discriminator steps. We trained using the Adam optimizer with betas set to (0.5,0.999) with a constant learning rate of 0.001. We train used the WGAN loss and gradient penalty with lambda set to 10. For the SVHN, CIFAR-10 and CIFAR-100 datasets, we use a whitening transform at the start of training to make the data zero-mean and marginally unit-variance; note that this affine transformation does not affect the manifold structure of the data.

NFs We use a rational quadratic spline flow (Durkan et al., 2019) with 64 hidden units, 4 layers, and 3 blocks per layer. For the SVHN, CIFAR-10 and CIFAR-100 datasets, we use a whitening transform at the start of training to make the data zero-mean and marginally unit-variance; note that this affine transformation does not affect the manifold structure of the data. Otherwise we scale the data followed by a logit transform with epsilon set to 0.001. We configure the mixture base distribution in the same way as the VAE prior. We set the initial learning rate to 0.0005.

WAEs For all datasets, we use an MLP with 2 hidden layers of size 256 each for the discriminator. The encoder and decoder for all datasets are the same convolutional networks used for VAEs except we increase the channels to $(64, 64, 32, 32)$ and $(32, 32, 64, 64)$ respectively. The latent dimension of the model is also set to 20. We train every model for a total of 300 epochs and perform early stopping on the reconstruction error with a patience of 30 epochs only for the GMM baselines. We note that when training for the full 300 epochs (without early stopping) the GMM baselines perform significantly worse and the other models perform better. We use a learning rate of 0.0005 for the discriminator and do not use any learning rate schedule for the model. We weight the discriminator loss with a lambda value of 10. For the GMM, we set the prior to have 10 modes with fixed means spaced 3 units apart centred at 0, fixed standard deviations of 1 and uniform mixture weights. We attempted to learn these parameters similar to the VAE but observed worse results.

VAE+NF For the VAE, we followed the same setup as the single VAE except we keep the learning rate fixed to 0.001 throughout training. The NF density estimator is also identical to the single NF with the following differences: (1) for all datasets, we standardize the data (by subtracting the mean and dividing by the standard deviation) and do not scale the data or apply a logit transform, and (2) we set the learning rate to 0.001.

WAE+NF For the WAE, we followed the same setup as the single WAE except we use residual networks to handle the larger and more complex golden retriever datasets. The encoder first applies a convolution (increasing the channel dimension to 16), batch normalization, and ReLU activation to the input before a max pool operation with stride 2 and kernel size of 3. The main body of the encoder is a set of 4 residual layers with channel dimensions $(16, 32, 64, 128)$. Each layer consists of two basic blocks which are two consecutive convolution, BatchNorm and ReLU layers where the input of the block is added to the activation before the final ReLU. The stride of the first convolution of each layer is set to 2 to halve the spatial resolution. The skip connection is downsampled by a convolution with a stride of 2. The activations are then average-pooled across each channel and projected to the latent space dimension by a linear layer. The decoder is identical to the encoder main body except there are no downsampling operations, the spatial resolution of the activation is doubled at each layer by bilinear upsampling and the channel dimensions are $(128, 64, 32, 16)$. We perform early stopping on validation loss with a patience of 30 epochs, for a maximum of 300 epochs. The NF density estimator is also identical to the single NF with the following differences: (1) for all datasets, we standardize the data (by subtracting the mean and dividing by the standard deviation) and do not scale the data or apply a logit transform, (2) we set the learning rate to 0.001, and (3) we perform early stopping on validation loss with a patience of 30 epochs, for a maximum of 100 epochs.

D.2 Exploiting the union of manifolds hypothesis: Supervised learning

In order to investigate if there is a correlation between intrinsic dimension and the difficulty of classification for a given class, we train three different classifiers on CIFAR-100 and measure the accuracy for each class. We use the same hyperparameters and training strategy for each of the three models we consider (VGG-19 (Simonyan & Zisserman, 2015), ResNet-18, and ResNet-34 (He et al., 2016)). We do not modify the model architecture except for changing the dimension of the last linear layer to project into a 100-dimensional vector to match the number of classes in CIFAR-100. We use the CIFAR-100 dataset test split ($10{,}000$ images) then randomly select 10% of the training dataset as our validation set ($5{,}000$ images) and use the remaining 90% as our train split. Note that this validation split is constant across all runs. The classification accuracy is reported as the accuracy on the test set on the same epoch where the best validation accuracy is achieved. We do not add any data augmentations as doing so could modify intrinsic dimension estimates and confound our results. Studying the relationship between intrinsic dimension and data augmentations is an interesting direction for future work. The combination of (1) no data augmentations, and (2) the presence of a validation set causes our results to be lower than the state-of-the-art for the models, although the results remain self-consistent here. We train each model using the cross entropy loss for a total of 200 epochs starting with a learning rate of 0.1 and decrease this to $10^{-2}, 4 \times 10^{-3}$, and 8×10^{-4} after $60, 120$, and 160 epochs respectively. We use the standard SGD optimizer with momentum set to 0.9 and train with a weight decay of 5×10^{-4}. For all datasets, we normalize by the channel-wise mean and standard deviation of all training images. We use a batch size of 128 for training, and a batch size of 100 for both validation and test.

We now detail the exact loss we used for our experiments with the re-weighted cross entropy loss. A weighted version of the categorical cross entropy loss is given by:

$$-\sum_{i=1}^{n}\sum_{\ell=1}^{L}\omega_\ell \cdot y_{i,\ell} \cdot \log f_\theta(x_i)_\ell, \tag{10}$$

where y_i is a one-hot vector of length L corresponding to the label of x_i, $f_\theta(x_i)$ is the L-dimensional output of the classifier containing assigned class probabilities, and ω_ℓ is the scalar weight given to the ℓth class. The standard categorical cross entropy uses the weights $\omega_\ell = 1$ for $\ell = 1, \ldots, L$. Our modified weights are given by:

$$\omega_\ell = L \cdot \frac{\hat{d}_k^{(\ell)}}{\sum_{\ell'=1}^{L} \hat{d}_k^{(\ell')}}. \tag{11}$$

Note that when the intrinsic dimension estimates of all classes match, our proposed weights recover the standard ones, but more heavily weight classes of higher intrinsic dimension otherwise.