LIDL: LOCAL INTRINSIC DIMENSION ESTIMATION USING LIKELIHOOD

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the problem of local intrinsic dimension (LID) estimation. LID of the data is the minimal number of coordinates which are necessary to describe the data point and its neighborhood without the significant information loss. Existing methods for LID estimation do not scale well to high dimensional data because they rely on estimating the LID based on nearest neighbors structure, which may cause problems due to the curse of dimensionality. We propose a new method for Local Intrinsic Dimension estimation using Likelihood (**LIDL**), which yields more accurate LID estimates thanks to the recent progress in likelihood estimation in high dimensions, such as normalizing flows (NF). We show our method yields more accurate estimates than previous state-of-the-art algorithms for LID estimation on standard benchmarks for this problem, and that unlike other methods, it scales well to problems with thousands of dimensions. We anticipate this new approach to open a way to accurate LID estimation for real-world, high dimensional datasets and expect it to improve further with advances in the NF literature.

1 INTRODUCTION

One of the important problems in representation learning is estimation of the intrinsic dimensionality (ID) of the data (Ansuini et al., 2019; Li et al., 2018; Rubenstein et al., 2018), which we will refer to as IDE. It is a well-studied problem in the context of dimensionality reduction, clustering and classification problems (Camastra & Staiano, 2016; Kleindessner & Luxburg, 2015; Vapnik, 2013). E.g. using more dimensions than necessary can lead to several problems, such as an increase in the space required to store data, and a decrease in the algorithm speed, since it generally depends on data dimensionality. Besides, building reliable classifiers becomes harder when the ID grows (curse of dimensionality) (Bellman, 2015). ID is also relevant for some prototype-based clustering algorithms (Claussen & Villmann, 2005; Struski et al., 2018). IDE becomes even more important in the context of representation learning: Rubenstein et al. (2018) show how the mismatch between the latent space dimensionality and the intrinsic data dimensionality may hurt the performance of auto-encoder based generative models like VAE (Kingma & Welling, 2014), WAE (Tolstikhin et al., 2017) or CWAE (Knop et al., 2020).

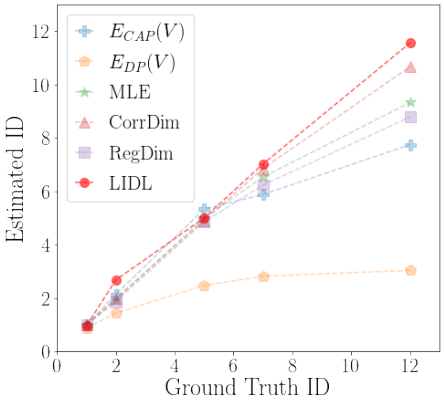

Figure 1: Comparison of LIDL and methods from Kleindessner & Luxburg (2015). Based on data from Table 1.

IDE methods can be divided into two broad categories: global and local (Camastra & Staiano, 2016). The former aims to give a single estimate of the dimensionality of the entire dataset. However, describing the ID structure of a dataset using just a single number might discard the nuanced information available such as when the data lies on a union of manifolds with different numbers of dimensions. In contrast, local methods (Carter et al., 2009) aim to estimate the local dimensionality of the data manifold in the neighborhood of an arbitrary point on the manifold. This approach gives more insight into the nature of the dataset and provides more options to summarize the dimensionality of the manifold from the global perspective, similarly to how in general the estimate of

the density of a random variable provides richer information than just the estimates of its summary statistics. Most existing methods (Kleindessner & Luxburg, 2015; Levina & Bickel, 2004; Hino et al., 2017; Camastra & Staiano, 2016) tend to analyze the local structure of the manifold by investigating the sample's neighborhood, which unfortunately does not scale well to higher dimensions due to the curse of dimensionality affecting such approaches.

To address this problem, we propose a new method for Local Intrinsic Dimension estimation using Likelihood (LIDL), which yields more accurate LID estimates thanks to the recent progress in likelihood estimation in high dimensions. Our method makes use of the observation that LID can be estimated using likelihood estimates under data distributions perturbed with isotropic Gaussian noise of varying magnitude. Those likelihoods can be estimated sufficiently accurately even in high dimensional spaces thanks to the recent progress in neural density estimators; in this work, we make use of the novel model class of NF (Dinh et al., 2014; Rezende & Mohamed, 2015; Kingma & Dhariwal, 2018).

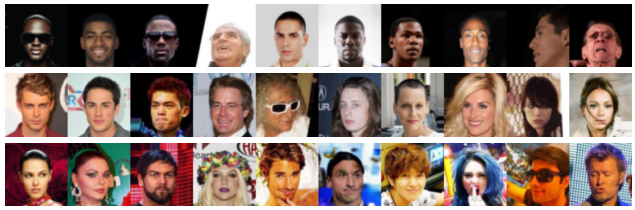

Figure 2: Images from Celeb-A dataset with low (1st row), medium (2nd row), high (3rd row) LID estimates calculated using our algorithm.

Our contributions We introduce an algorithm for LID estimation based on NF models, we show that the algorithm outperforms other methods for low dimensional standard benchmarks for this problem, and that it scales well to high-dimensional datasets.

2 METHOD

The manifold hypothesis (Fefferman et al., 2016), which we assume, says that locally every data point is sampled from a distribution defined on a K–dimensional manifold embedded in a higher dimensional space of dimension M. A K–dimensional manifold, is a topological space with the property that each point has a neighborhood that is homeomorphic to the Euclidean space of dimension K. A dataset $D \subseteq \mathbb{R}^M$ is said to have ID equal to K if its elements lie entirely, without information loss, within a K-dimensional manifold of $\mathbb{R}^M$ (Camastra & Staiano, 2016).

Firstly, we present the insight that lies at the core of our method. Let's analyze how the probability density around point x_j from the data manifold behaves, when the points around it are perturbed with a small isotropic Gaussian noise $\epsilon \sim \mathcal{N}(0, \delta I), \epsilon \in \mathbb{R}^M$. This distribution is parametrized by a single scalar parameter δ, which controls the noise amplitude. The important thing to notice is, that we perturb x_j by adding noise of dimensionality M, so after this operation the perturbed point $\hat{x}_j$ no longer lies on the manifold if $K < M$, and we can estimate its probability density in $\mathbb{R}^M$. We can rotate whole space in a way, that locally around x_j the manifold is aligned with first K dimensions of the data space. This operation does not change probability density of $\hat{x}_j$ but simplifies our inference. If we do this, we can divide our space into dimensions that lie entirely on the data manifold in the neighborhood of x_j, and for those that are orthogonal to it. We can now calculate density of this perturbed point in $\mathbb{R}^M$ using this separation and fact, that the noise is independent from the data distribution.

Adding the noise to the first K dimensions means adding the noise in the directions that lie on the manifold. When we assume that the noise amplitude is small and the distribution is approximately constant near x_j, this operation just moves data points around the manifold without changing their probability density (except a small fraction of points near the edges). We are going to show how it works on the example of the uniform distribution on the $[a, b]$ interval in $\mathbb{R}^1$:

$$f_{D+\delta}(x) = \int_{-\infty}^{\infty} f_D(x) f_\delta(x-g)dg = C \int_a^b f_\delta(x-g)dg = \frac{C}{2}\left[\text{erf}\left(\frac{b-x}{\sqrt{2\delta^2}}\right) - \text{erf}\left(\frac{a-x}{\sqrt{2\delta^2}}\right)\right]$$

where erf is an error function; $f_D(x) = \frac{1}{b-a}$, where $a < x < b$, is a data distribution; $f_\delta(x) = \frac{1}{\sqrt{2\pi\delta^2}} e^{-x^2/2\delta^2}$, where $-\infty < x < \infty$, is a noise distribution; $f_{D+\delta}$ is the data distribution f_D

perturbed with noise f_δ and $C = \frac{1}{b-a}$. For points x that lie between a and b and in a distance at least a few δ's away from the edges of the data distribution the probability density is in practice equal $\frac{1}{b-a}$. If we take δ that is small enough, probability density for majority of the distribution will not be affected by adding the noise.

If we add the noise in the directions that are orthogonal to the data manifold we lower the logarithm of probability density proportionally to the number of dimensions orthogonal to the data manifold. The noise f_δ does not change on-manifold density of the data (as we shown in previous paragraph) and is independent from f_D in the directions orthogonal to the manifold, so the joint distribution $f_{D+\delta}$ is the product of f_D and f_O – the probability density of the noise for the dimensions orthogonal to the manifold. If the original point from the distribution has a density $f_D(x)$, the probability density of perturbed point measured at x_j has a probability density equal $f_{D+\delta}(x) = f_D(x) * f_O(0) = f_D(x) * (1/\sqrt{2\pi\delta^2})^L$ where $L = M - K$. If we take logarithm of both sides of the equation and we obtain:

$$\log f_{D+\delta}(x_j) = \log f_D(x_j) + L \log \frac{1}{\delta} + L \log \frac{1}{\sqrt{2\pi}} = -L \log \delta + const. \tag{1}$$

If we use a few NF models (described in Appendix B) to obtain estimate $q_i(x_j)$ of the perturbed distributions $f_{D+\delta_i}(x_j)$'s for a few values of δ_i, we can fit linear regression algorithm for those points and obtain estimate of L_j at the point x_j. This estimate can then be used to calculate K_j, i.e. the manifold dimensionality at x_j. Interesting thing is, that after we estimate L using regression we can substitute it into Eq. 1 and calculate $\log f_D(x)$ – the density on the original manifold. Complementary derivation of this approach for Gaussian distribution can be found in Appendix C. Now we are ready to introduce **LIDL** algorithm.

LIDL Algorithm To estimate LID at a set of N points $S = x_1, ..., x_j, ..., x_N$ in the dataset D we have to fit $d > 1$ NF models $\mathcal{F}_i$ ($i = 1, ..., d$) to d datasets D_i. Each D_i is perturbed version of D with different noise $\mathcal{N}(0, \delta_i I)$ added to each point. The probability density for the same coordinates is different when estimated by different $\mathcal{F}_i$ and it goes down monotonically as the value of δ_i increases. If we then calculate values of $\log p_i(x_j)$ for a point x_j obtained from different models $\mathcal{F}_i$ and plot them against $\log \delta_i$ used to perturb the dataset D_i they line up around the straight line. Slope of this line is equal $L = M - K$, i.e. the difference between the data space dimensionality and the manifold dimensionality at the point x_i. Algorithm is written out in details in Appendix A.

3 Experiments

Behavior for different ranges of δ's For many real-world datasets the manifold thickness varies in different on-manifold directions in space, and in our method we make an assumption, that δ is much smaller than manifold thickness in any on-manifold direction. But what happens when our assumption is not true? We analyzed this behavior on some artificial datasets. We prepared a rectangular dataset with one side of the rectangle equals 0.01 and other one equals 1. When we used LIDL with δ's around 0.0001 to estimate ID, we get the number of manifold dimensions $k \approx 2$. When we used δ's around 0.01 on the same dataset, we get $1 < k < 2$, when we used δ's around 0.1 we get $k \approx 1$ and for bigger deltas k was smaller than 1. We also analyzed this behavior on artificial datasets with the whole spectrum of dimension thicknesses, and we observed, that we roughly count number of dimensions thicker than z when we use δ's around z.

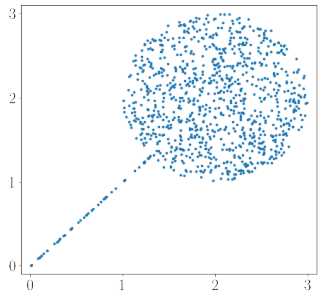

Figure 3: 2D lollipop dataset used in our experiments.

Lollipop dataset For the first experiment we used an artificial 1D/2D dataset shown in Fig. 3. We trained MAF (Papamakarios et al., 2017) model to fit to 8 different datasets D_i perturbed with $0.02 \leq \delta_i \leq 0.04$. Average of LID estimated for points in the the head of the lollipop is 1.96 and for points from the stick part it was 1.0.

Comparison with other algorithms on artificial datasets We collated LIDL with other LID estimation algorithms (Levina & Bickel, 2004; Kleindessner & Luxburg, 2015) for intrinsic dimension estimation by comparing it with estimates in Table 1 from Kleindessner & Luxburg (2015). Each algorithm from this table was tested on a dataset of size 1000, so we used the data set of the same size for LIDL training (we used 750 examples from this set for training and 250 for validation). We used RQ-NSF(Durkan et al., 2019) and MAF models on this datasets. We trained for 20000 batches of size 750 with early stopping after 1000 batches without improvement on the validation dataset. We present results of this comparison (each one with standard deviation calculated using 10 experiments) in Table 1 and show the results from it in Fig. 1. We can clearly see on this figure, that LIDL gives results closer to the original dimensionality, than any other LID estimation algorithms, especially for higher dimensions.

Table 1: LIDL comparison with algorithms from Table 1 in Kleindessner & Luxburg (2015).

Distribution	ID	$E_{CAP}(V)$	$E_{DP}(V)$	MLE	CorrDim	RegDim	**LIDL**
uniform on a helix in $\mathbb{R}^3$	1	$1.00_{\pm.05}$	$0.88_{\pm.01}$	$1.00_{\pm.01}$	$1.00_{\pm.11}$	$0.99_{\pm.01}$	$0.97_{\pm.15}$
Swiss roll in $\mathbb{R}^3$	2	$2.14_{\pm.05}$	$1.44_{\pm.01}$	$1.94_{\pm.02}$	$1.99_{\pm.23}$	$1.87_{\pm.04}$	$2.68_{\pm.35}$
$\mathcal{N}_5(0, I) \subseteq \mathbb{R}^5$	5	$5.33_{\pm.07}$	$2.47_{\pm.01}$	$5.00_{\pm.04}$	$4.91_{\pm.56}$	$4.86_{\pm.05}$	$5.00_{\pm.02}$
uniform on sphere $S^7 \subseteq \mathbb{R}^8$	7	$5.88_{\pm.06}$	$2.82_{\pm.01}$	$6.53_{\pm.07}$	$6.85_{\pm.66}$	$6.23_{\pm.09}$	$7.02_{\pm.18}$
uniform on $[0, 1]^{12}$ in $\mathbb{R}^{12}$	12	$7.74_{\pm.08}$	$3.04_{\pm.01}$	$9.32_{\pm.10}$	$10.66_{\pm1.18}$	$8.78_{\pm.10}$	$11.55_{\pm.33}$

High-dimensional artificial datasets We trained LIDL on four datasets of size 10K with Gaussian distribution embedded in higher dimensional space with ID equals 1, 10, 100 and 1K dimensions. Results are presented in Table 2. We obtain highest error (relative and absolute) for the 2K dimensional dataset (with manifold with ID equals 1K), which may be due the small sample size compared to dimensionality of the data space. We can see from those experiments, that LIDL has no problem with scaling to higher dimensions.

Table 2: LIDL estimated ID in higher dimensions.

Data distribution	$\mathcal{N}_1(0, I) \subseteq \mathbb{R}^2$	$\mathcal{N}_{10}(0, I) \subseteq \mathbb{R}^{20}$	$\mathcal{N}_{100}(0, I) \subseteq \mathbb{R}^{200}$	$\mathcal{N}_{1000}(0, I) \subseteq \mathbb{R}^{2000}$
LIDL ID Estimate	$1.02_{\pm.04}$	$10.14_{\pm.08}$	$100.92_{\pm.62}$	$1048.42_{\pm21.52}$

Image datasets We ran LIDL on MNIST (image size 32x32x1) (LeCun & Cortes, 2010), FMNIST (32x32x1) (Xiao et al., 2017) and Celeb-A (64x64x3) (Liu et al., 2015) datasets using Glow (Kingma & Dhariwal, 2018) as a density model. Estimated dimensionalities for MNIST images span roughly from 50 to 250, for FMNIST those numbers are 100 and 500 and for Celeb-A we estimated dimensionalities between 3500 and 8000. We sorted the dataset according to the dimensionality and observed, that more complicated examples have higher dimensionalities. Some small, medium and high dimensional images are shown in Fig. 2 and 4.

More results and details of our experiments can be found in Appendix D.

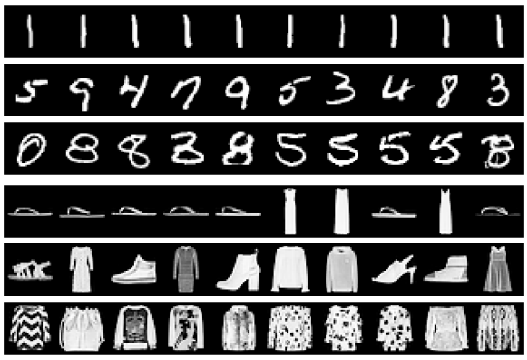

Figure 4: At the first three rows we show MNIST images with low, medium and high estimates from LIDL. Three bottom rows show FMNIST images with low, medium and high estimates from LIDL.

4 Conclusions

We introduced an algorithm for LID estimation based on NF as density estimators, provided a theoretical justification for it and showed that it can scale to datasets of thousands of dimensions. Our approach however is limited by the ability of NF models to scale to even higher dimensions. For now we are not able to cope with datasets of images consisting millions of pixels. We hope that current intensive research on NF models will make them able to scale to those datasets eventually and automatically make LIDL to be able to estimate LID for them as well.

REFERENCES

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 6111–6122, 2019.

Richard E Bellman. *Adaptive control processes: a guided tour*. Princeton university press, 2015.

Francesco Camastra and Antonino Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

Kevin M Carter, Raviv Raich, and Alfred O Hero III. On local intrinsic dimension estimation and its applications. *IEEE Transactions on Signal Processing*, 58(2):650–663, 2009.

Gabriel B Cavallari, Leonardo SF Ribeiro, and Moacir A Ponti. Unsupervised representation learning using convolutional and stacked auto-encoders: a domain and cross-domain feature space analysis. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 440–446. IEEE, 2018.

Jens Christian Claussen and Thomas Villmann. Magnification control in winner relaxing neural gas. *Neurocomputing*, 63:125–137, 2005.

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

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *arXiv preprint arXiv:1906.04032*, 2019.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. nflows: normalizing flows in pytorch, November 2020. URL `https://doi.org/10.5281/zenodo.4296287`.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Hideitsu Hino, Jun Fujiki, Shotaro Akaho, and Noboru Murata. Local intrinsic dimension estimation by generalized linear modeling. *Neural Computation*, 29(7):1838–1878, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `http://arxiv.org/abs/1412.6980`. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

D.P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv:1312.6114*, 2014.

Matthäus Kleindessner and Ulrike Luxburg. Dimensionality estimation without distances. In *Artificial Intelligence and Statistics*, pp. 471–479, 2015.

Szymon Knop, Przemysław Spurek, Jacek Tabor, Igor Podolak, Marcin Mazur, and Stanisław Jastrzebski. Cramer-wold auto-encoder. *Journal of Machine Learning Research*, 21, 2020.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17:777–784, 2004.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, 2015.

Paul K Rubenstein, Bernhard Schoelkopf, and Ilya Tolstikhin. On the latent space of wasserstein auto-encoders. *arXiv preprint arXiv:1802.03761*, 2018.

Łukasz Struski, Jacek Tabor, and Przemysław Spurek. Lossy compression approach to subspace clustering. *Information Sciences*, 435:161–183, 2018.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

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

Appendices

A LIDL ALGORITHM

Algorithm 1: LIDL – algorithm for local intrinsic dimension estimation using NF.

Input: Dataset D;
d – number of models to estimate;
M – dimensionality of the data space;
$\Delta = (\delta_1, ..., \delta_d)$ – list of δ's;
Set S of N points from $\mathbb{R}^M$, at which we want to estimate LID;
Result: List of K_j's – LID estimates for points in S;
for δ_i *in* Δ **do**
  initialize model $\mathcal{F}_i$;
  while $\mathcal{F}_i$ *not converged* **do**
    Sample batch b from D;
    Sample noise n_i from $\mathcal{N}(0, \delta_i)$;
    $b_i \leftarrow b + n_i$;
    Make training step on $\mathcal{F}_i$ using b_i;
  end
  for x_j *in* S **do**
    for $\mathcal{F}_i$ *in* $(\mathcal{F}_1, ..., \mathcal{F}_d)$ **do**
      Estimate likelihood q_{ij} at point x_j using model $\mathcal{F}_i$;
    end
    Calculate regression coefficient L_j for a list of d pairs:
    $((\log q_{1j}, \log \delta_1), ..., (\log q_{dj}, \log \delta_d))$;
    $K_j \leftarrow M - L_j$;
  end
end

We observed that when perturbing the original dataset with different noise sample in each batch we obtain more accurate results when estimating LID, than when we perturb the dataset once at the beginning of the training. It also stabilizes training, especially for larger values of δ.

B NORMALIZING FLOWS

NF are very flexible tools for approximating probability distributions. They use parametrized non-linear invertible transformation f_θ and change of variable formula to transform a simple density $\pi(z)$ into a more complicated one. NF are trained using gradient-based methods (e.g. SGD) to maximize log-likelihood of the data

$$\max_\theta \sum_i \log q(x_i)$$

where

$$q(x) = \pi(f_\theta(x)) \left| \det \frac{\partial f_\theta(x)}{\partial x} \right|.$$

We used MAF (Papamakarios et al., 2017), RQ-NSF (Durkan et al., 2019) and Glow (Kingma & Dhariwal, 2018) models in our experiments. Model selection criterion was: we took the simplest model able to fit the data.

C ALTERNATIVE DERIVATION FOR GAUSSIAN DISTRIBUTION

Normal distribution with a diagonal covariance matrix We have a M-dimensional i.i.d. random variable $X \in \mathbb{R}^M$ with a normal density on a K-dimensional subspace of $\mathbb{R}^M$, $K < M$:

$$X \sim \mathcal{N}(\mu, \Sigma), \quad \mu = (\underbrace{0, \dots, 0}_{\text{M}}), \quad \Sigma = \text{diag}(\underbrace{\sigma_1^2, \dots, \sigma_K^2}_{K}, \underbrace{0, \dots, 0}_{L}),$$

where $L = M - K$.

Assume we add an isotropic normal noise $\epsilon \sim \mathcal{N}(0, \delta), \epsilon \in \mathbb{R}^M$ to X and thus we obtain a new random variable $X' = X + \epsilon$ which is also normally distributed with $\mu' = \mu$ and covariance matrix Σ' taking the form

$$\Sigma' = \Sigma + \delta^2 I = \text{diag}(\underbrace{\sigma_1^2 + \delta^2, \dots, \sigma_K^2 + \delta^2}_{K}, \underbrace{\delta^2, \dots, \delta^2}_{L}). \tag{2}$$

Then the differential entropy of X' takes the form

$$\begin{aligned}
h_{X'} &= -\int p_{X'}(x) \log p_{X'}(x)\, dx && (3)\\
&= -\int \mathcal{N}(x|\mu, \Sigma') \log \mathcal{N}(x|\mu, \Sigma')\, dx && (4)\\
&= \frac{1}{2} \ln \det (2\pi \mathrm{e} \Sigma') && (5)\\
&= \frac{1}{2} \ln \det \Sigma' + \text{const} && (6)\\
&= \frac{1}{2} \sum_{i=1}^{K} \ln(\sigma_i^2 + \delta^2) + \frac{1}{2} \sum_{i=K+1}^{M} \ln(\delta^2) + \text{const} && (7)\\
&= \frac{1}{2} \sum_{i=1}^{K} \ln(\sigma_i^2 + \delta^2) + L \ln(\delta) + \text{const.} && (8)
\end{aligned}$$

We are interested in the differential entropy as a function of the magnitude of the added noise δ, i.e., $h_{X'}(\delta)$. Assuming that $\forall_i, \delta \ll \sigma_i$ we can approximate the first term of equation 8 as constant wrt δ since $\frac{d \ln \delta^2}{d\delta} \gg \frac{d \ln(\sigma_i^2 + \delta^2)}{d\delta}$, yielding

$$h_{X'}(\delta) = L \ln(\delta) + \text{const.} \tag{9}$$

We can approximate the differential entropy $h_{X'}(\delta)$ through the negative of expected log-likelihood of the trained density estimator $p_{X'}$ under the empirical data distribution $\hat{p}_{X'}$ which is simply the Maximum Likelihood training loss for a density estimator with parameters θ,

$$\mathcal{L}(\theta, \delta) = -\mathbb{E}_{x \sim \hat{p}_{X'}(x|\delta)} \left[p_{X'}(x; \delta, \theta) \right] \approx h_{X'}(\delta) \tag{10}$$

where $\hat{p}_{X'}(x|\delta)$ is the empirical data distribution over X' (the dataset with added normal noise of magnitude δ), and $p_{X'}(x;\delta,\theta)$ is the density under the generative model trained on data with added noise of magnitude δ.

Let's assume we've trained multiple density estimators for different values δ (or a single density estimator conditioned on δ) and evaluated $\mathcal{L}_f(\delta) = \mathcal{L}(\theta_{f,\delta}, \delta)$ for multiple values of δ where $\theta_{f,\delta}$ are the final parameters of the training of the density estimator on data with noise of magnitude δ). From equation 9 we see we are able to form it into a linear regression problem which allows us to estimate the value of l, and hence the dimensionality of the subspace generating distribution $k = K = M - L$, as per

$$\mathcal{L}_f(\delta) = L\ln(\delta) + \text{const.} \tag{11}$$

Normal distribution with an arbitrary covariance matrix In case of a normal distribution with an arbitrary covariance matrix Σ, thanks to Eigendecomposition Σ can always be expressed as $\Sigma = Q\Lambda Q^{-1}$ where Λ is a diagonal matrix whose diagonal elements are the Eigenvalues of Σ and Q is the square orthonormal $m \times m$ matrix whose columns are Eigenvectors of Σ. Transforming the frame of reference according to Q the covariance matrix diagonalizes as following

$$Q^{-1}\Sigma' Q = Q^{-1}(\Sigma + \delta^2 I)Q = Q^{-1}\Sigma Q + \delta^2 Q^{-1} I Q = \Lambda + \delta^2 Q^{-1} Q = \Lambda + \delta^2 I.$$

Given that our procedure doesn't depend on the frame of reference (since the noise ϵ is isotropic), the method from previous paragraph applies in this setting.

D Experimental details and other results

δ sampling In each experimented we used δ's equally distanced in logarithmic scale. This means that when we write, that we used n δ's between a and b, this means that we used $\Delta = (a, \exp(\log a + \gamma), \exp(\log a + 2\gamma), ..., \exp(\log a + (n-2)\gamma), b)$ in LIDL, where $\gamma = (\log b - \log a)/(n-1)$.

Lollipop dataset we used MAF implementation from nflows library (Durkan et al., 2020). Each model was trained for 10K steps on batches of size 1K sampled from the lollipop distribution using Adam optimizer (Kingma & Ba, 2014) with learinig rate equal 0.002. We trained 8 models using δ's between 0.02 and 0.04.

Comparison with other algorithms on artificial datasets we used MAF and RQ-NSF implementations from nflows library. RQ-NSF was used for helix and Swiss roll datasets, MAF was used for rest of the datasets. Each model was trained on training set containing 750 examples for maximum of 20K steps on batches of size 750. Model was optimized using Adam method (Kingma & Ba, 2014) with learning rate equal 0.00002. We used early stopping after 1K steps without improvement on validation set containing 250 examples. We trained 8 models using δ's between 0.08 and 0.12.

High-dimensional artificial datasets we used MAF implementation from nflows library. Each model was trained on training set containing 7500 examples for maximum of 20K steps on batches of size 750. Model was optimized using Adam method (Kingma & Ba, 2014) with learning rate equal 0.00002. We used early stopping after 1K steps without improvement on validation set containing 2500 examples. We trained 8 models using δ's between 0.08 and 0.12.

Image datasets we used a pytorch Glow implementation from `https://github.com/chaiyujin/glow-pytorch`. Models for MNIST and FMNIST datasets were trained on 8 δ's between 0.01 and 0.1 for 200 epochs with batch size equal 64. Optimization was done using Adam with learning rate equals 0.00005. Models on Celeb-A were trained on the same δ's, but for 40 epochs, batch size equal 16 and learning rate equal 0.00002.

Dimensionality distributions for MNIST and FMNIST Empirical CDF's for both datasets are presented in Fig. 5 and 6. For MNIST dataset we obtained dimensionalities between 40 and 250 dimensions. Maximum dimensionality reported in Table 1 in Kleindessner & Luxburg (2015) is 15. From the other hand the best classification results using auto-encoder representation as an input is

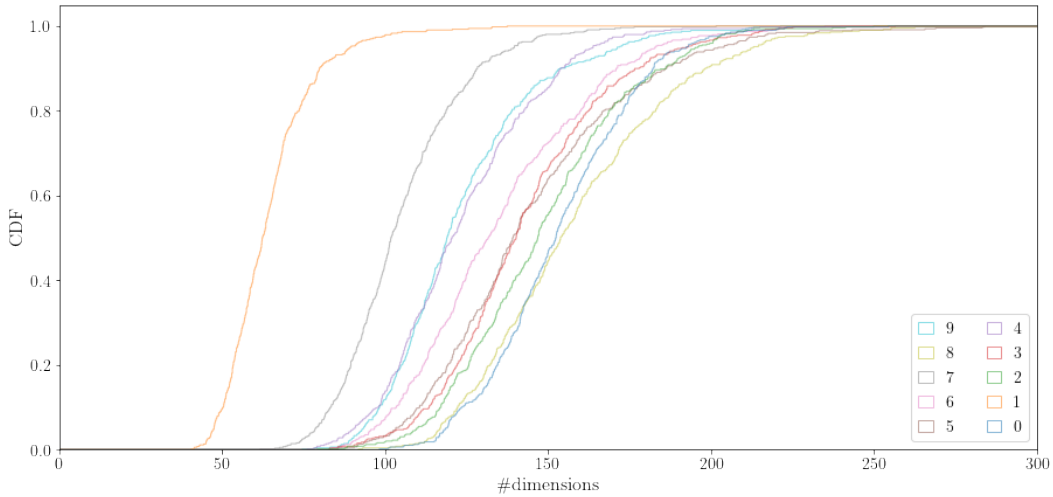

Figure 5: Empirical cumulative distribution function (CDF) of 5000 examples from MNIST dataset. Each line represents CDF for separate class in dataset. Class number (which also is a represented digit in this case) can be found in legend.

reported in Cavallari et al. (2018) and Wang et al. (2016) for bottleneck size in auto-encoder greater than 100, which is in accordance with our result. This is the reason we think that dimensionalities estimated by LIDL seems more reasonable than those reported in Kleindessner & Luxburg (2015)

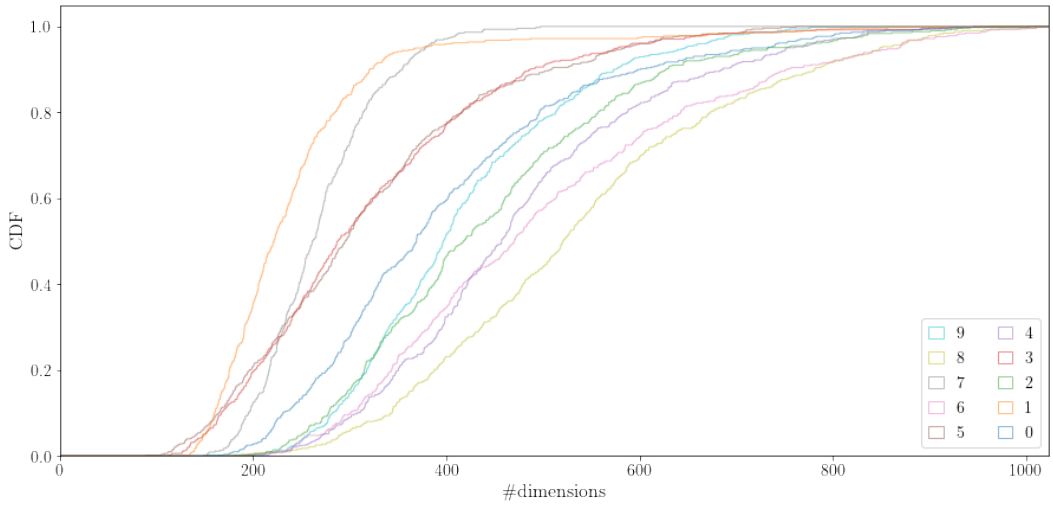

Figure 6: Empirical cumulative distribution function (CDF) of 5000 examples from FMNIST dataset. Each line represents CDF for separate class in dataset. Class number can be found in legend.

Training time and its connection with dimensionality We observed, that the longer the training was, the lower the dimensionalities were. The effect was stronger for real-world datasets. We also observed that validation and test dataset dimensionality was often slightly higher than the train dataset dimensionality. We think that it may be connected to the process of training NF: we observed that they often wrap roughly around the major shapes of the original manifold and during the process of training they are fitting to the smaller and smaller details of the manifold. For artificial datasets we used validation loss increase as a criterion for stopping the training. For image datasets we trained the models for an amount of epochs sufficient to obtain a good quality samples from NF model.