Differentiable Local Intrinsic Dimension Estimation with Diffusion Models

Hamidreza Kamkari [1] **Brendan Leigh Ross** [1] **Rasa Hosseinzadeh** [1] **Jesse C. Cresswell** [1] **Gabriel Loaiza-Ganem** [1]

Abstract

High-dimensional data commonly lies on low-dimensional submanifolds, and estimating the *local intrinsic dimension* (LID) of a datum is a long-standing problem. LID can be understood as the number of local factors of variation: the more factors of variation a datum has, the more complex it tends to be. Estimating this quantity has proven useful in contexts ranging from generalization in neural networks to detection of out-of-distribution data, adversarial examples, and AI-generated text. While many model-free and several model-based estimation techniques exist, none are differentiable by design. In this work, we show that the Fokker-Planck equation associated with a diffusion model can provide the first LID estimator that is differentiable and scales to high dimensional data while outperforming existing baselines on LID estimation benchmarks.

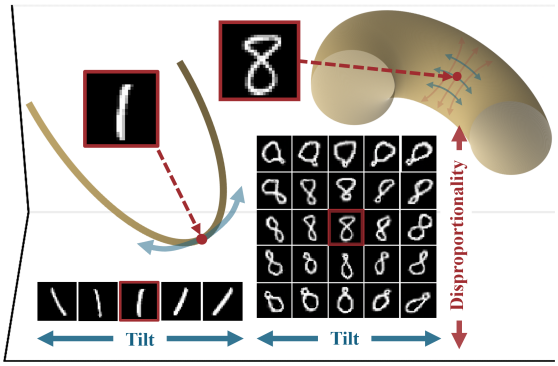


Figure 1. An illustration showing that LID is a natural measure of relative complexity. We depict two manifolds of MNIST digits, corresponding to 1s and 8s, as 1d and 2d submanifolds of $\mathbb{R}^3$, respectively. The relatively simpler manifold of 1s exhibits a single factor of variation ("tilt"), whereas 8s have an additional factor of variation ("disproportionality").

1. Introduction

The manifold hypothesis (Bengio et al., 2013) states that high-dimensional data in $\mathbb{R}^D$ often lies on low-dimensional submanifolds. For a given datum $x \in \mathbb{R}^D$, this hypothesis motivates using its *local intrinsic dimension* (LID), denoted $\mathrm{LID}(x)$, as a natural measure of its complexity. $\mathrm{LID}(x)$ can be intuitively understood as the minimal number of variables needed to describe x, as illustrated in Figure 1.

LID has been used to detect outliers (Houle et al., 2018; Kamkari et al., 2024b), AI-generated text (Tulchinskii et al., 2023), and adversarial examples (Ma et al., 2018), among other applications. Traditional *model-free* estimators of intrinsic dimension (Fukunaga & Olsen, 1971; Levina & Bickel, 2004; Johnsson et al., 2014) typically rely on pairwise distances and nearest neighbours, rendering them expensive and non-differentiable in practice. Recent work uses deep generative models that implicitly learn the data submanifolds, suggesting they can be used to construct LID estimators. However, existing *model-based* estimators suffer from drawbacks including being inaccurate and computationally expensive (Stanczuk et al., 2022), not leveraging the best existing generative models (Zheng et al., 2022), requiring training several models (Tempczyk et al., 2022) or altering the training procedure rather than relying on a pre-trained model (Horvat & Pfister, 2024). In all these cases the LID estimators are non-differentiable in practice.

We address all these issues by showing how to estimate LID efficiently using only a single pre-trained diffusion model (DM). We leverage the Fokker-Planck equation to propose FLIPD[1], the first LID estimator to scale to high-resolution images ($\sim 10^6$ dimensions), and the first that is tractably differentiable. While differentiating through local intrinsic dimension estimation is not a common sub-task today, this may simply be because no existing methods enable it. We present these ideas with the hope that future work will find even more applications given these new capabilities.

[1]Layer 6 AI, Toronto, Canada. Correspondence to: Hamidreza Kamkari <hamid@layer6.ai>.

Published at the 2nd Differentiable Almost Everything Workshop at the 41st International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

[1]Pronounced as "flipped", the acronym is a rearrangement of "FP" from Fokker-Planck and "LID".

2. Background and Related Work

Diffusion Models Score-based diffusion models (Song et al., 2021b) convert data to noise through the forward (Itô) stochastic differential equation (SDE),

$$\mathrm{d}X_t = f(X_t, t)\mathrm{d}t + g(t)\mathrm{d}W_t, \quad X_0 \sim p(\cdot, 0), \tag{1}$$

where W_t denotes a D-dimensional Brownian motion. We write the distribution of X_t as $p(\cdot, t)$. The backward process $Y_t := X_{1-t}$ obeys a backward SDE involving the score function $s(x, t) := \nabla \log p(x, t)$ which is modeled by a neural network $\hat{s}(x, t)$ via denoising score matching (Vincent, 2011). DMs admit density evaluation, placing them in the same category as other deep generative models, such as normalizing flows. As we will see, generative models with this capability can be utilized to produce LID estimates.

LID The *local intrinsic dimension* of a point x is the dimension of the manifold it belongs to, hence, LID is not an intrinsic property of the point x, but rather a property of x with respect to the manifold that contains it. Tempczyk et al. (2022) proposed LIDL, a method for LID estimation relying on normalizing flows. LIDL works thanks to a surprising result linking Gaussian convolutions and LID (Loaiza-Ganem et al., 2022; Tempczyk et al., 2022; Zheng et al., 2022). We will denote the convolution of $p(\cdot, 0)$ and Gaussian noise with log standard deviation δ as $\varrho(\cdot, \delta)$, i.e.

$$\varrho(x, \delta) := \int p(x_0, 0)\mathcal{N}(x - x_0; 0, e^{2\delta} I_D)\mathrm{d}x_0. \tag{2}$$

The aforementioned result suggests that, for negative enough values of δ (i.e. small enough standard deviations):

$$\log \varrho(x, \delta) \approx \delta(\mathrm{LID}(x) - D) + c \tag{3}$$

for some constant c. If we could evaluate $\log \varrho(x, \delta)$ for various values of δ, this would provide an avenue for estimating $\mathrm{LID}(x)$: set some values $\delta_1, \ldots, \delta_m$, fit a linear regression using $\{(\delta_i, \log \varrho(x, \delta_i))\}_{i=1}^m$ (with δ as the covariate and $\log \varrho(x, \delta)$ as the response), and let $\hat{\beta}_x$ be the corresponding slope. It follows that $\hat{\beta}_x$ estimates $\mathrm{LID}(x) - D$, so that $\mathrm{LID}(x) \approx D + \hat{\beta}_x$ is a sensible estimator of LID.

Since $\varrho(x, \delta)$ is unknown, LIDL requires training m normalizing flows. More specifically, for each δ_i, a normalizing flow is trained on data to which $\mathcal{N}(0, e^{2\delta_i} I_D)$ noise is added. In LIDL, the log densities of the trained models are then used instead of the unknown true log densities $\log \varrho(x, \delta_i)$ when fitting the regression as described above.

3. Method

Although Tempczyk et al. (2022) used normalizing flows in LIDL, they did point out that these models could be swapped for any other generative model admitting density evaluation. Indeed, one could trivially train m DMs and replace the flows with them. As opposed to the normalizing flows used in LIDL which are individually trained on datasets with different levels of noise added, a single diffusion model already works by convolving data with various noise levels and allows density evaluation of the resulting noisy distributions (Song et al., 2021b). Hence, we show that LIDL can be used with a *single* DM. Throughout this section, we assume access to a pre-trained DM such that $f(x, t) = b(t)x$. This choice implies that the transition kernel $p_{t|0}$ associated with Equation 1 is Gaussian (Särkkä & Solin, 2019):

$$p_{t|0}(x_t \mid x_0) = \mathcal{N}(x_t; \psi(t)x_0, \sigma^2(t)I_D). \tag{4}$$

We assume that b and g are such that ψ and σ are differentiable and $\lambda(t) := \sigma(t)/\psi(t)$ is injective. This encompasses all DMs commonly used in practice, including variance-exploding, variance-preserving (of which DDPM (Ho et al., 2020) is a discretized instance), and sub-variance-preserving (Song et al., 2021b). In Appendix A we include explicit formulas for $\psi(t)$, $\sigma^2(t)$, and $\lambda(t)$ for these particular DMs.

3.1. FLIPD: A Fokker-Planck-Based LID Estimator

LIDL is based on Equation 3, which justifies using a regression. Differentiating this equation yields that $\partial \log \varrho(x, \delta_0)/\partial\delta \approx \mathrm{LID}(x) - D$ for negative enough δ_0. By leveraging the Fokker-Planck equation associated with Equation 1, we show in Appendix C that, for DMs with transition kernel as in Equation 4,

$$\begin{aligned}\frac{\partial}{\partial\delta} \log \varrho(x, \delta) = &\sigma^2\big(t(\delta)\big) \Big(\mathrm{tr}\Big(\nabla s\big(\psi(t(\delta))x, t(\delta)\big)\Big) \\ &+ \big\|s\big(\psi(t(\delta))x, t(\delta)\big)\big\|_2^2\Big) =: \nu\big(t(\delta); s, x\big).\end{aligned} \tag{5}$$

Equation 5 directly provides the rate of change that the regression in LIDL aims to estimate, from which we get:

$$\mathrm{LID}(x) \approx D + \nu\big(t(\delta_0); \hat{s}, x\big) =: \mathrm{FLIPD}(x, t_0), \tag{6}$$

where $t(\delta) := \lambda^{-1}(e^\delta)$ and $t_0 := t(\delta_0)$. Computing FLIPD requires only a single hyperparameter, whereas regression-based estimators require m. Since $\nu(t(\delta); \hat{s}, x)$ depends on δ only through $t(\delta)$, we can directly set t_0 as the hyperparameter rather than δ_0, which avoids the potentially cumbersome computation of $t(\delta_0) = \lambda^{-1}(e^{\delta_0})$: instead of setting a suitably negative δ_0, we set $t_0 > 0$ sufficiently close to 0. Moreover, the expression for $\nu\big(t(\delta_0); \hat{s}, x\big)$ in Equation 5 contains only standard operations which are amenable to automatic differentiation. In Appendix A we include explicit formulas for $\mathrm{FLIPD}(x, t_0)$ for common DMs.

4. Experiments

Throughout our experiments, we use variance-preserving DMs, the most popular variant of DMs, and compatible with DDPMs; see Appendix D.1 for hyperparameters.

Table 1. MAE (lower is better) | concordance indices (higher is better with 1.0 being the gold standard). Rows show synthetic manifolds and columns represent LID estimation methods. Columns are grouped based on whether they use a generative model, with the best results for each metric within each group being bolded.

	Model-based						Model-free			
Synthetic Manifold	FLIPD		NB		LIDL		ESS		LPCA	
String within doughnut $\subseteq \mathbb{R}^3$	**0.06**	**1.00**	1.48	0.48	1.10	0.99	0.02	**1.00**	**0.00**	**1.00**
$\mathcal{L}_5 \subseteq \mathbb{R}^{10}$	0.17	-	1.00	-	**0.10**	-	0.07	-	**0.00**	-
$\mathcal{N}_{90} \subseteq \mathbb{R}^{100}$	0.49	-	**0.18**	-	0.33	-	**1.67**	-	21.9	-
$\mathcal{U}_{10} + \mathcal{U}_{30} + \mathcal{U}_{90} \subseteq \mathbb{R}^{100}$	**1.30**	**1.00**	61.6	0.34	8.46	0.74	21.9	0.74	**20.1**	**0.86**
$\mathcal{N}_{10} + \mathcal{N}_{25} + \mathcal{N}_{50} \subseteq \mathbb{R}^{100}$	**1.81**	**1.00**	74.2	0.34	8.87	0.74	7.71	0.88	**5.72**	**0.91**
$\mathcal{F}_{10} + \mathcal{F}_{25} + \mathcal{F}_{50} \subseteq \mathbb{R}^{100}$	**3.93**	**1.00**	74.2	0.34	18.6	0.70	9.20	0.90	**6.77**	**1.00**
$\mathcal{U}_{10} + \mathcal{U}_{80} + \mathcal{U}_{200} \subseteq \mathbb{R}^{800}$	**14.3**	**1.00**	715	0.34	120	0.70	**2.35**	**1.00**	33.7	**1.00**
$\mathcal{U}_{900} \subseteq \mathbb{R}^{1000}$	**12.8**	-	100	-	24.9	-	**75.2**	-	801	-

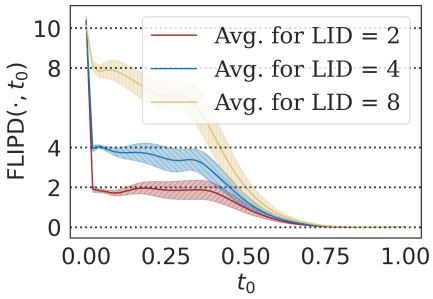


Figure 2. FLIPD curves with knees at the true LID.

The effect of t_0 FLIPD requires setting t_0 close to 0. It is important to note that DMs fitted to low-dimensional manifolds are known to exhibit numerically unstable scores $s(\cdot, t_0)$ as $t_0 \searrow 0$ (Vahdat et al., 2021; Lu et al., 2023; Loaiza-Ganem et al., 2024). Our first set of experiments examines the effect of t_0 on FLIPD(x, t_0) by varying t_0 within the range $(0, 1)$.

In Figure 2, we train a DM on a mixture of three isotropic Gaussians with dimensions 2, 4, and 8, embedded in $\mathbb{R}^{10}$ (each embedding is carried out by multiplication against a random matrix with orthonormal columns plus a random translation). While FLIPD(x, t_0) is inaccurate at $t_0 = 0$ due to the aforementioned instabilities, it quickly stabilizes around the true LID for all datapoints. We refer to this pattern as a *knee* in the FLIPD curve. We persistently see knees in FLIPD curves (in Appendix D.2, we show similar curves for more complex data manifolds). Not only is this in line with the observations of LIDL on normalizing flows (see Figure 5 of Tempczyk et al. (2022)), but it also gives us a fully automated approach to setting t_0. We leverage `kneedle` (Satopaa et al., 2011), a knee detection algorithm which aims to find points of maximum curvature. Rather than fixing t_0, we evaluate Equation 6 for 50 values of t_0 and pass the results to `kneedle` to automatically detect the t_0 where a knee occurs.

Synthetic experiments We create a benchmark for LID evaluation on complex unions of manifolds where true LID is known. We sample from simple distributions on low-dimensional spaces, and then embed the samples into $\mathbb{R}^D$. We denote uniform, Gaussian, and Laplace distributions as $\mathcal{U}, \mathcal{N}$, and $\mathcal{L}$, respectively, with sub-indices indicating LID, and a plus sign denoting mixtures. To embed samples into higher dimensions, we apply a random matrix with orthonormal columns and then apply a random translation. For example, $\mathcal{N}_{10} + \mathcal{L}_{20} \subseteq \mathbb{R}^{100}$ indicates a 10-dimensional Gaussian and a 20-dimensional Laplace, each of which undergoes a random affine transformation mapping to $\mathbb{R}^{100}$ (one transformation per component). We also generate non-linear manifolds, denoted with $\mathcal{F}$, by applying a randomly initialized D-dimensional neural spline flow (Durkan et al., 2019) after the affine transformation (when using flows, the input noise is always uniform); since the flow is a diffeomorphism, it preserves LID.

Here, we summarize our synthetic experiments in Table 1 using two metrics of performance: the mean absolute error (MAE) between the predicted and true LID for individual datapoints; and (when the dataset has variability in its ground truth LIDs) the concordance index, which measures similarity in the rankings between then true LIDs and the estimated ones. We compare against the LIDL estimator described in Section 2, as well another DM-based method (NB, (Stanczuk et al., 2022)) and two of the most performant model-free baselines (LPCA, (Fukunaga & Olsen, 1971); ESS (Johnsson et al., 2014)). For the NB baseline, we use the exact same DM backbone as for FLIPD, and for LIDL we use 8 neural spline flows. In terms of MAE, FLIPD tends to be the best model-based estimator particularly as dimension increases. Although model-free baselines perform well in simplistic scenarios, they produce unreliable results as LID increases or more non-linearity is introduced in the data manifold. In terms of concordance index, FLIPD achieves *perfect* scores in all scenarios, meaning that even when its estimates are off, it always provides correct LID rankings. In Appendix D we include ablations for FLIPD and comparisons on more high-dimensional datasets, and with other model-free baselines.

Image experiments We first focus on the simple image datasets MNIST and FMNIST. We flatten the images and use

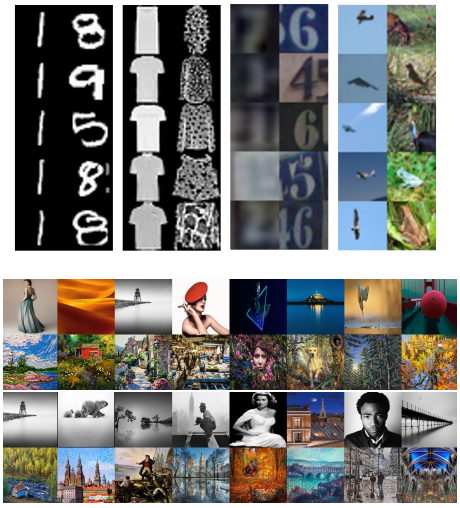


Figure 3. **Top:** Images with small and large FLIPD estimates from FMNIST, MNIST, SVHN, and CIFAR10. **Bottom:** LAION images with small and large FLIPD estimates using Stable Diffusion (top, $t_0 = 0.5$) and PNG compression sizes (bottom).

the same MLP architecture as in our synthetic experiments. Despite using an MLP, our DMs can generate reasonable samples (Appendix E.1). The average LID estimates are 130 and 170, respectively. Although our LID estimates are higher than given in prior work (Pope et al., 2021; Brown et al., 2023), our high-dimensional experiments (Table 7 of Appendix D.4) and findings by Tempczyk et al. (2022) and Stanczuk et al. (2022) show that model-free baselines underestimate LID of high-dimensional data, especially images. On more complex image datasets like SVHN and CIFAR10, the simplistic MLP score network backbone fails to generate high-quality samples. Therefore, we replaced it with state-of-the-art UNets (Ronneberger et al., 2015; von Platen et al., 2022) (see Appendix E.2). We took a random subset of 4096 images from FMNIST, MNIST, SVHN, and CIFAR10, and sorted them according to their FLIPD estimates. We show the top and bottom 5 images for each dataset in Figure 3, and include more samples in Appendix E.3. Our visualization shows that higher FLIPD estimates indeed correspond to images with more detail and texture, while lower estimates correspond to less complex ones. FLIPD not only visually corresponds to image complexity, but also required only 50 Jacobian-vector-products, resulting in a significant speedup over all prior work.

Further, we quantitatively assess our estimates by computing Spearman's rank correlation coefficient between different LID estimators and PNG compression size, used as a proxy for complexity in the absence of ground truth. As shown in Table 2, FLIPD has a high correlation with PNG, whereas model-free estimators do not. We find that the NB estimator correlates slightly more with PNG on MNIST and CIFAR10, but significantly less in FMNIST and SVHN.

Table 2. Spearman's correlation between LID estimates and PNG compression size.

Method	MNIST	FMNIST	CIFAR10	SVHN
FLIPD	0.837	**0.883**	0.819	**0.876**
NB	**0.864**	0.480	**0.894**	0.573
ESS	0.444	0.063	0.326	0.019
LPCA	0.413	0.01	0.302	−0.008

Finally, we consider high-resolution images from LAION-Aesthetics (Schuhmann et al., 2022) and, for the first time, estimate LID for extremely high-dimensional images with $D = 3 \times 512 \times 512 = 786{,}432$. To achieve this, we use Stable Diffusion (Rombach et al., 2022), a latent DM pretrained on LAION-5B (Schuhmann et al., 2022). This includes an encoder and a decoder trained to preserve relevant characteristics of the data manifold in latent representations. Since the encoder and decoder are continuous and effectively invert each other, we argue that the Stable Diffusion encoder can, for practical purposes, be considered a topological embedding of the LAION-5B dataset into its latent space of dimension $4 \times 64 \times 64 = 16{,}384$. Therefore, the dimension of the LAION-5B submanifold in latent space should be unchanged. This approximation allows us to estimate image LIDs by carrying out FLIPD in the latent space of Stable Diffusion. Here, we set the Hutchinson sample count to 1, meaning we only require a *single* Jacobian-vector-product. When we order a random subset of 1600 samples according to their FLIPD at $t_0 = 0.3$, the more complex images are clustered at the end, while the least complex are clustered at the beginning: see Figure 3 for the lowest- and highest-LID images from this ordering, and Figure 24 in Appendix E.6 to view the entire subset and other values of t_0. In comparison to orderings according to PNG compression size (Figure 3), FLIPD estimates prioritize semantic complexity over low-level details like colouration.

5. Conclusions, Limitations, and Future Work

In this work, we have shown that the Fokker-Planck equation can be utilized for efficient LID estimation with any pre-trained DM. We have provided strong theoretical foundations and extensive benchmarks showing that FLIPD estimates accurately reflect data complexity. Although FLIPD produces excellent LID estimates on synthetic benchmarks, the lack of knees in FLIPD curves on image data when using state-of-the-art architectures is surprising, and results in unstable LID estimates which strongly depend on t_0. We see this behaviour as a limitation, even if FLIPD provides a meaningful measure of complexity in these cases. Given that FLIPD is tractable, differentiable, and compatible with any DM, we hope that it will find uses in applications where LID estimates have already proven helpful, including OOD detection, AI-generated data analysis, and adversarial example detection.

References

Albergante, L., Bac, J., and Zinovyev, A. Estimating the effective dimension of large biological datasets using fisher separability analysis. In *2019 International Joint Conference on Neural Networks*, pp. 1–8. IEEE, 2019.

Bac, J., Mirkes, E. M., Gorban, A. N., Tyukin, I., and Zinovyev, A. Scikit-Dimension: A Python Package for Intrinsic Dimension Estimation. *Entropy*, 23(10):1368, 2021.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013.

Brown, B. C., Caterini, A. L., Ross, B. L., Cresswell, J. C., and Loaiza-Ganem, G. Verifying the union of manifolds hypothesis for image data. In *International Conference on Learning Representations*, 2023.

Cangelosi, R. and Goriely, A. Component retention in principal component analysis with application to cdna microarray data. *Biology Direct*, 2:1–21, 2007.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. In *Advances in Neural Information Processing Systems*, 2019.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. nflows: normalizing flows in PyTorch, November 2020. URL https://doi.org/10.5281/zenodo.4296287.

Fukunaga, K. and Olsen, D. R. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 100(2):176–183, 1971.

Harrell Jr, F. E., Lee, K. L., and Mark, D. B. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, 15(4):361–387, 1996.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Horvat, C. and Pfister, J.-P. On gauge freedom, conservativity and intrinsic dimensionality estimation in diffusion models. In *International Conference on Learning Representations*, 2024.

Houle, M. E., Schubert, E., and Zimek, A. On the correlation between local intrinsic dimensionality and outlierness. In *Similarity Search and Applications: 11th International Conference, SISAP 2018*, pp. 177–191. Springer, 2018.

Johnsson, K., Soneson, C., and Fontes, M. Low bias local intrinsic dimension estimation from expected simplex skewness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):196–202, 2014.

Kamkari, H., Balazadeh, V., Zehtab, V., and Krishnan, R. G. Order-based structure learning with normalizing flows. *arXiv:2308.07480*, 2024a.

Kamkari, H., Ross, B. L., Cresswell, J. C., Caterini, A. L., Krishnan, R. G., and Loaiza-Ganem, G. A geometric explanation of the likelihood OOD detection paradox. *arXiv:2403.18910*, 2024b.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Why normalizing flows fail to detect out-of-distribution data. In *Advances in Neural Information Processing Systems*, 2020.

Levina, E. and Bickel, P. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems*, 2004.

Loaiza-Ganem, G., Ross, B. L., Cresswell, J. C., and Caterini, A. L. Diagnosing and fixing manifold overfitting in deep generative models. *Transactions on Machine Learning Research*, 2022.

Loaiza-Ganem, G., Ross, B. L., Hosseinzadeh, R., Caterini, A. L., and Cresswell, J. C. Deep generative models through the lens of the manifold hypothesis: A survey and new connections. *arXiv:2404.02954*, 2024.

Lu, Y., Wang, Z., and Bal, G. Mathematical analysis of singularities in the diffusion model under the submanifold assumption. *arXiv:2301.07882*, 2023.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018.

MacKay, D. J. and Ghahramani, Z. Comments on "Maximum likelihood estimation of intrinsic dimension' by E. Levina and P. Bickel (2004). *The Inference Group Website, Cavendish Laboratory, Cambridge University*, 2005.

Mayr, P. and Petras, V. Cross-concordances: terminology mapping and its effectiveness for information retrieval. *arXiv:0806.3765*, 2008.

Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*. Cambridge University Press, 2019.

Satopaa, V., Albrecht, J., Irwin, D., and Raghavan, B. Finding a "kneedle" in a Haystack: Detecting Knee Points in System Behavior. In *31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171. IEEE, 2011.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. In *Advances in Neural Information Processing Systems*, 2021a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Stanczuk, J., Batzolis, G., Deveney, T., and Schönlieb, C.-B. Your diffusion model secretly knows the dimension of the data manifold. *arXiv:2212.12611*, 2022.

Steck, H., Krishnapuram, B., Dehing-Oberije, C., Lambin, P., and Raykar, V. C. On ranking in survival analysis: Bounds on the concordance index. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Teles, J. Concordance coefficients to measure the agreement among several sets of ranks. *Journal of Applied Statistics*, 39(8):1749–1764, 2012.

Tempczyk, P., Michaluk, R., Garncarek, L., Spurek, P., Tabor, J., and Golinski, A. LIDL: Local intrinsic dimension estimation using approximate likelihood. In *International Conference on Machine Learning*, pp. 21205–21231, 2022.

Tulchinskii, E., Kuznetsov, K., Kushnareva, L., Cherniavskii, D., Nikolenko, S., Burnaev, E., Barannikov, S., and Piontkovskaya, I. Intrinsic dimension estimation for robust detection of ai-generated texts. In *Advances in Neural Information Processing Systems*, 2023.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems*, 2021.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M., and Wolf, T. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

Zheng, Y., He, T., Qiu, Y., and Wipf, D. P. Learning manifold dimensions with conditional variational autoencoders. In *Advances in Neural Information Processing Systems*, 2022.

A. Explicit Formulas

A.1. Variance-Exploding Diffusion Models

Variance-exploding DMs are such that $f(x,t) = 0$ with g being non-zero. In this case (Song et al., 2021b):

$$\psi(t) = 1, \quad \text{and} \quad \sigma^2(t) = \int_0^t g^2(u)\mathrm{d}u. \tag{7}$$

Since g is non-zero, g^2 is positive, so that σ^2 is increasing, and thus injective. It follows that σ is also injective, so that $\lambda = \sigma/\psi = \sigma$ is injective. Equation 6 then implies that

$$\text{FLIPD}(x, t_0) = D + \sigma^2(t_0)\left[\text{tr}\left(\nabla s(x, t_0)\right) + \|s(x, t_0)\|_2^2\right]. \tag{8}$$

A.2. Variance-Preserving Diffusion Models (DDPMs)

Variance-preserving DMs are such that

$$f(x,t) = -\frac{1}{2}\beta(t)x, \quad \text{and} \quad g(t) = \sqrt{\beta(t)}, \tag{9}$$

where β is a positive scalar function. In this case (Song et al., 2021b):

$$\psi(t) = e^{-\frac{1}{2}B(t)}, \quad \text{and} \quad \sigma^2(t) = 1 - e^{-B(t)}, \quad \text{where} \quad B(t) \coloneqq \int_0^t \beta(u)\mathrm{d}u. \tag{10}$$

We then have that

$$\lambda(t) = \sqrt{\frac{\sigma^2(t)}{\psi^2(t)}} = \sqrt{e^{B(t)} - 1}. \tag{11}$$

Since β is positive, B is increasing and thus injective, from which it follows that λ is injective as well. Plugging everything into Equation 6, we obtain:

$$\text{FLIPD}(x, t_0) = D + \left(1 - e^{-B(t_0)}\right)\left(\text{tr}\left(\nabla s\left(e^{-\frac{1}{2}B(t_0)}x, t_0\right)\right) + \left\|s\left(e^{-\frac{1}{2}B(t_0)}x, t_0\right)\right\|_2^2\right). \tag{12}$$

A.3. Sub-Variance-Preserving Diffusion Models

Sub-variance-preserving DMs are such that

$$f(x,t) = -\frac{1}{2}\beta(t)x, \quad \text{and} \quad g(t) = \sqrt{\beta(t)\left(1 - e^{-2B(t)}\right)}, \quad \text{where} \quad B(t) \coloneqq \int_0^t \beta(u)\mathrm{d}u, \tag{13}$$

and where β is a positive scalar function. In this case (Song et al., 2021b):

$$\psi(t) = e^{-\frac{1}{2}B(t)}, \quad \text{and} \quad \sigma^2(t) = \left(1 - e^{-B(t)}\right)^2. \tag{14}$$

We then have that

$$\lambda(t) = \frac{\sigma(t)}{\psi(t)} = e^{\frac{1}{2}B(t)} - e^{-\frac{1}{2}B(t)} = 2\sinh\left(\frac{1}{2}B(t)\right). \tag{15}$$

Since β is positive, B is increasing and thus injective, from which it follows that λ is injective as well due to the injectivity of $\sinh$. Plugging everything into Equation 6, we obtain:

$$\text{FLIPD}(x, t_0) = D + \left(1 - e^{-B(t_0)}\right)^2\left(\text{tr}\left(\nabla s\left(e^{-\frac{1}{2}B(t_0)}x, t_0\right)\right) + \left\|s\left(e^{-\frac{1}{2}B(t_0)}x, t_0\right)\right\|_2^2\right). \tag{16}$$

Table 3. Comparing the discretized DDPM notation with score-based DM side-by-side.

Term	DDPM (Ho et al., 2020)	Score-based DM (Song et al., 2021b)
Timestep	$t \in \{0, 1, \dots, T\}$	$t/T = t \in [0, 1]$
(Noised out) datapoint	x_t	$x_{t/T} = x_t$
Diffusion process hyperparameter	β_t	$\beta(t/T) = \beta(t)$
Mean of transition kernel	$\sqrt{\bar{\alpha}_t}$	$\psi(t/T) = \psi(t)$
Std of transition kernel	$\sqrt{1 - \bar{\alpha}_t}$	$\sigma(t/T) = \sigma(t)$
Network parameterization	$-\epsilon(x, t)/\sqrt{1 - \bar{\alpha}_t}$	$\hat{s}(x, t/T) = \hat{s}(x, t)$

B. Adapting FLIPD for DDPMs

Here, we adapt FLIPD for state-of-the-art DDPM architectures and follow the discretized notation from Ho et al. (2020) where instead of using a continuous time index t from 0 to 1, a timestep t belongs instead to the sequence $\{0, \dots, T\}$ with T being the largest timescale. We use the colour gold to indicate the notation used by Ho et al. (2020). We highlight that the content of this section is a summary of the equivalence between DDPMs and the score-based formulation established by Song et al. (2021b).

As a reminder, DDPMs can be viewed as discretizations of the *forward* SDE process of a DM, where the process turns into a Markov *noising* process:

$$p(x_t \mid x_{t-1}) \coloneqq \mathcal{N}(x_t; \sqrt{1 - \beta_t} \cdot x_{t-1}, \beta_t I_D). \tag{17}$$

We also use sub-indices t instead of functions evaluated at t to keep consistent with Ho et al. (2020)'s notation. This in turn implies the following transition kernel:

$$p(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I_D) \tag{18}$$

where $\alpha_t \coloneqq 1 - \beta_t$ and $\bar{\alpha}_t \coloneqq \prod_{s=1}^{t} \alpha_t$.

DDPMs model the *backward* diffusion process (or *denoising* process) by modelling a network $\epsilon : \mathbb{R}^D \times \{1, \dots, T\} \to \mathbb{R}^D$ that takes in a noised-out point and outputs a residual that can be used to denoise. In particular, for every t:

$$x_t + \epsilon(x_t, t) \sim p(x_{t-1} | x_t). \tag{19}$$

Song et al. (2021b) show that one can draw an equivalence between the network $\epsilon(\cdot, t)$ and the score network (see their Appendix B). Here, we rephrase the connections in a more explicit manner where we note that:

$$-\epsilon(x, t)/\sqrt{1 - \bar{\alpha}_t} = s(x, t/T). \tag{20}$$

Consequently, plugging into Equation 12, we get the following formula adapted for DDPMs:

$$\text{FLIPD}(x, t_0) = D - \sqrt{1 - \bar{\alpha}_{t_0}} \operatorname{tr}\left(\nabla \epsilon(\sqrt{\bar{\alpha}_{t_0}} x, t_0)\right) + \|\epsilon(\sqrt{\bar{\alpha}_{t_0}} x, t_0)\|_2^2, \tag{21}$$

where $t_0 = t_0 \times T$ (best viewed in colour). We include Table 3, which summarizes all of the equivalent notation when moving from DDPMs to score-based DMs and vice-versa.

C. Derivations

C.1. Diffusion Model Densities are Convolutions

In this section we give a concrete relation between the density of a DM and the convolution of data with Gaussian noise defined in Equation 2. We show that for any arbitrary DM with transition kernel as in Equation 4, it holds that

$$\log \varrho(x, \delta) = D \log \gamma(\delta) + \log p\big(\gamma(\delta) x, t(\delta)\big), \tag{22}$$

where $t(\delta) \coloneqq \lambda^{-1}(e^{\delta})$ and $\gamma(\delta) \coloneqq \psi(t(\delta))$. We have that:

$$p\big(\gamma(\delta)x, t(\delta)\big) = \int p_{t(\delta)|0}\big(\gamma(\delta)x \mid x_0\big)p(x_0, 0)\mathrm{d}x_0 \tag{23}$$

$$= \int \mathcal{N}\Big(\gamma(\delta)x; \psi\big(t(\delta)\big)x_0, \sigma^2\big(t(\delta)\big)I_D\Big)p(x_0, 0)\mathrm{d}x_0 \tag{24}$$

$$= \int \mathcal{N}\Big(\gamma(\delta)x; \gamma(\delta)x_0, \sigma^2\big(t(\delta)\big)I_D\Big)p(x_0, 0)\mathrm{d}x_0 \tag{25}$$

$$= \frac{1}{\gamma^D(\delta)} \int \mathcal{N}\left(x; x_0, \frac{\sigma^2\big(t(\delta)\big)}{\psi^2\big(t(\delta)\big)} I_D\right) p(x_0, 0)\mathrm{d}x_0 \tag{26}$$

$$= \frac{1}{\gamma^D(\delta)} \int \mathcal{N}\Big(x; x_0, \lambda^2\big(t(\delta)\big)I_D\Big)p(x_0, 0)\mathrm{d}x_0 \tag{27}$$

$$= \frac{1}{\gamma^D(\delta)} \int \mathcal{N}\big(x; x_0, e^{2\delta}I_D\big)p(x_0, 0)\mathrm{d}x_0 = \frac{1}{\gamma^D(\delta)}\varrho(x, \delta), \tag{28}$$

where we used that $\mathcal{N}(ax; ax_0, \sigma^2 I_D) = \frac{1}{a^D}\mathcal{N}(x; x_0, (\sigma^2/a^2)I_D)$. Taking logarithms yields Equation 22.

C.2. Derivation of Equation 5

First, we recall the Fokker-Planck equation associated with the SDE in Equation 1, which states that:

$$\frac{\partial}{\partial t}p(x, t) = -p(x, t)\left[\nabla \cdot f(x, t)\right] - \langle f(x, t), \nabla p(x, t)\rangle + \frac{1}{2}g^2(t)\,\mathrm{tr}\left(\nabla^2 p(x, t)\right). \tag{29}$$

We begin by using this equation to derive $\partial/\partial t\ \log p(x, t)$. Noting that $\nabla p(x, t) = p(x, t)s(x, t)$, we have that:

$$\mathrm{tr}\left(\nabla^2 p(x, t)\right) = \mathrm{tr}\left(\nabla\left[p(x, t)s(x, t)\right]\right) = p(x, t)\,\mathrm{tr}\left(\nabla s(x, t)\right) + \mathrm{tr}\left(s(x, t)\nabla p(x, t)^\top\right) \tag{30}$$

$$= p(x, t)\left[\mathrm{tr}\left(\nabla s(x, t)\right) + \|s(x, t)\|_2^2\right]. \tag{31}$$

Because

$$\frac{\partial}{\partial t}p(x, t) = p(x, t)\frac{\partial}{\partial t}\log p(x, t), \tag{32}$$

it then follows that

$$\frac{\partial}{\partial t}\log p(x, t) = -\left[\nabla \cdot f(x, t)\right] - \langle f(x, t), s(x, t)\rangle + \frac{1}{2}g^2(t)\left[\mathrm{tr}\left(\nabla s(x, t)\right) + \|s(x, t)\|_2^2\right]. \tag{33}$$

Then, from Equation 22 and the chain rule, we get:

$$\frac{\partial}{\partial \delta}\log \varrho(x, \delta) = \frac{\mathrm{d}}{\mathrm{d}\delta}\left[D\log\psi\big(t(\delta)\big) + \log p\Big(\psi\big(t(\delta)\big)x, t(\delta)\Big)\right] \tag{34}$$

$$= D\left[\frac{\mathrm{d}}{\mathrm{d}\delta}\log\psi\big(t(\delta)\big)\right] + \begin{pmatrix} \nabla \log p\Big(\psi\big(t(\delta)\big)x, t(\delta)\Big) \\ \frac{\partial}{\partial t}\log p\Big(\psi\big(t(\delta)\big)x, t(\delta)\Big) \end{pmatrix}^\top \begin{pmatrix} \frac{\partial}{\partial t}\psi\big(t(\delta)\big)x \\ 1 \end{pmatrix} \frac{\partial}{\partial \delta}t(\delta) \tag{35}$$

$$= \left[\frac{\partial}{\partial \delta}t(\delta)\right]\left[D\frac{\frac{\partial}{\partial t}\psi\big(t(\delta)\big)}{\gamma(\delta)} + \begin{pmatrix} s\big(\gamma(\delta)x, t(\delta)\big) \\ \frac{\partial}{\partial t}\log p\big(\gamma(\delta)x, t(\delta)\big) \end{pmatrix}^\top \begin{pmatrix} \frac{\partial}{\partial t}\psi\big(t(\delta)\big)x \\ 1 \end{pmatrix}\right] \tag{36}$$

$$= \left[\frac{\partial}{\partial \delta}t(\delta)\right]\left[\left(\frac{\partial}{\partial t}\psi\big(t(\delta)\big)\right)\left(\frac{D}{\gamma(\delta)} + \Big\langle x, s\big(\gamma(\delta)x, t(\delta)\big)\Big\rangle\right) + \frac{\partial}{\partial t}\log p\big(\gamma(\delta)x, t(\delta)\big)\right]. \tag{37}$$

Substituting Equation 33 into Equation 37 yields:

$$\frac{\partial}{\partial\delta}\log\varrho(x,\delta) = \left[\frac{\partial}{\partial\delta}t(\delta)\right]\Bigg[\left(\frac{\partial}{\partial t}\psi\big(t(\delta)\big)\right)\left(\frac{D}{\gamma(\delta)} + \Big\langle x, s\big(\gamma(\delta)x, t(\delta)\big)\Big\rangle\right) - \big[\nabla\cdot f\big(\gamma(\delta)x, t(\delta)\big)\big] - \Big\langle f\big(\gamma(\delta)x, t(\delta)\big), s\big(\gamma(\delta)x, t(\delta)\big)\Big\rangle + \frac{1}{2}g^2\big(t(\delta)\big)\left(\operatorname{tr}\left(\nabla s\big(\gamma(\delta)x, t(\delta)\big)\right) + \big\|s\big(\gamma(\delta)x, t(\delta)\big)\big\|_2^2\right)\Bigg]. \tag{38}$$

From now on, to simplify notation, when dealing with a scalar function h, we will denote its derivative as h'. Since $t(\delta) = \lambda^{-1}(e^\delta)$, the chain rule gives:

$$\frac{\partial}{\partial\delta}t(\delta) = \frac{e^\delta}{\lambda'\big(\lambda^{-1}(e^\delta)\big)} = \frac{\lambda\big(t(\delta)\big)}{\lambda'\big(t(\delta)\big)}. \tag{39}$$

So far, we have not used that $f(x,t) = b(t)x$, which implies that $\nabla\cdot f(x,t) = Db(t)$ and that $\langle f(x,t), s(x,t)\rangle = b(t)\langle x, s(x,t)\rangle$. Using these observations and Equation 39, Equation 38 becomes:

$$\frac{\partial}{\partial\delta}\varrho(x,\delta) = \frac{\lambda\big(t(\delta)\big)}{\lambda'\big(t(\delta)\big)}\Bigg[\left(\frac{\psi'\big(t(\delta)\big)}{\psi\big(t(\delta)\big)} - b\big(t(\delta)\big)\right)D + \Big\langle\Big(\psi'\big(t(\delta)\big) - b\big(t(\delta)\big)\psi\big(t(\delta)\big)\Big)x, s\big(\gamma(\delta)x, t(\delta)\big)\Big\rangle + \frac{1}{2}g^2\big(t(\delta)\big)\left(\operatorname{tr}\left(\nabla s\big(\gamma(\delta)x, t(\delta)\big)\right) + \big\|s\big(\gamma(\delta)x, t(\delta)\big)\big\|_2^2\right)\Bigg]. \tag{40}$$

If we showed that

$$\psi'(t) - b(t)\psi(t) = 0, \quad \text{and that} \quad \frac{\lambda(t)}{2\lambda'(t)}g^2(t) = \sigma^2(t), \tag{41}$$

then Equation 40 would simplify to Equation 5. From equation 5.50 in (Särkkä & Solin, 2019), we have that

$$\psi'(t) = b(t)\psi(t), \tag{42}$$

which shows that indeed $\psi'(t) - b(t)\psi(t) = 0$. Then, from equation 5.51 in (Särkkä & Solin, 2019), we also have that

$$\left(\sigma^2\right)'(t) = 2b(t)\sigma^2(t) + g^2(t), \tag{43}$$

and from the chain rule this gives that

$$\sigma'(t) = \frac{2b(t)\sigma^2(t) + g^2(t)}{2\sigma(t)}. \tag{44}$$

We now finish verifying Equation 41. Since $\lambda(t) = \sigma(t)/\psi(t)$, the chain rule implies that

$$\frac{\lambda(t)}{2\lambda'(t)}g^2(t) = \frac{\dfrac{\sigma(t)}{\psi(t)}}{\dfrac{\sigma'(t)\psi(t) - \sigma(t)\psi'(t)}{\psi^2(t)}}\frac{g^2(t)}{2} = \frac{\sigma(t)\psi(t)}{\sigma'(t)\psi(t) - \sigma(t)b(t)\psi(t)}\frac{g^2(t)}{2} \tag{45}$$

$$= \frac{\sigma(t)}{\sigma'(t) - b(t)\sigma(t)}\frac{g^2(t)}{2} = \frac{\sigma(t)}{\dfrac{2b(t)\sigma^2(t) + g^2(t)}{2\sigma(t)} - b(t)\sigma(t)}\frac{g^2(t)}{2} \tag{46}$$

$$= \frac{2\sigma^2(t)}{2b(t)\sigma^2(t) + g^2(t) - 2b(t)\sigma^2(t)}\frac{g^2(t)}{2} = \sigma^2(t), \tag{47}$$

which, as previously mentioned, shows that Equation 40 simplifies to Equation 5.

Table 4. Essential hyperparameter settings for the Diffusion models with MLP backbone.

Property	Model Configuration
Learning rate	10^{-4}
Optimizer	`AdamW`
Scheduler	Cosine scheduling with 500 warmup steps (von Platen et al., 2022)
Epochs	200, 400, 800, or 1000 based on the ambient dimension
Score-matching loss	Likelihood weighting (Song et al., 2021a)
SDE drift	$f(x,t) := -\frac{1}{2}\beta(t)x$
SDE diffusion	$g(t) := \sqrt{\beta(t)}$
$\beta(t)$	Linear interpolation: $\beta(t) := 0.1 + 20t$
Score network	MLP
MLP hidden sizes	$\langle 4096, 2048, 2\times 1024, 3\times 512, 2\times 1024, 2048, 4096\rangle$
Time embedding size	128

D. Experimental Details

Throughout all our experiments, we used an NVIDIA A100 GPU with 40GB of memory.

D.1. DM Hyperparameter Setup

Throughout our experiments, we use an MLP architecture with a bottleneck as our score network: $2\times L+1$ fully connected layers with dimensions $\langle h_1, h_2, \ldots, h_L, \ldots h_{2L+1}\rangle$ forming a bottleneck, i.e., h_L has the smallest size. Notably, for $1 \leq i \leq L$, the ith transform connects layer $i-1$ (or the input) to layer i with a linear transform of dimensions "$h_{i-1}\times h_i$", and the $(L+i)$th layer (or the output) not only contains input from the $(L+i-1)$th layer but also, contains skip connections from layer $(L-i)$ (or the input), thus forming a linear transform of dimension "$(h_{L+i-1}+h_{L-i})\times h_{L+i}$". For image experiments, we scale and shift the pixel intensities to be zero-centered with a standard deviation of 1. In addition, we embed times $t \in (0,1)$ using the scheme in (von Platen et al., 2022) and concatenate with the input before passing to the score network. All hyperparameters are summarized in Table 4.

D.2. FLIPD Estimates and Curves for Synthetic Distributions

Figure 4 shows pointwise LID estimates for a lollipop distribution taken from (Tempczyk et al., 2022). It is a uniform distribution over three submanifolds: (i) a 2d candy, (ii) a 1d stick, and (iii) an isolated point of zero dimensions. Note that the FLIPD estimates at $t_0 = 0.05$ for all three submanifolds are coherent.

Figure 5 shows the FLIPD curve as training progresses on the lollipop example and the Gaussian mixture that was already discussed in Section 4. We see that gradually, knee patterns emerge at the correct LID, indicating that the DM is learning the data manifold. Notably, data with higher LID values get assigned higher estimates *even after a few epochs*, demonstrating that FLIPD effectively ranks data based on LID, even when the DM is underfitted.

Finally, Figure 6 presents a summary of complex manifolds obtained from neural spline flows and high-dimensional mixtures, showing knees around the true LID.

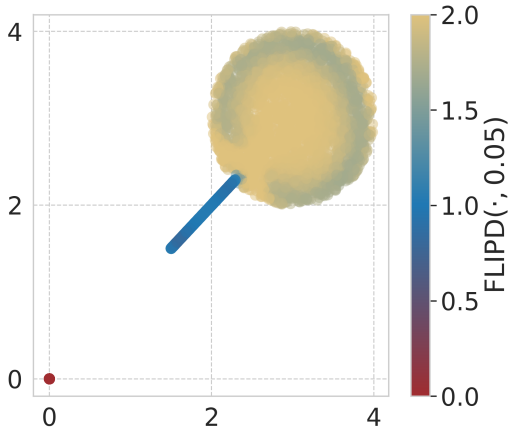
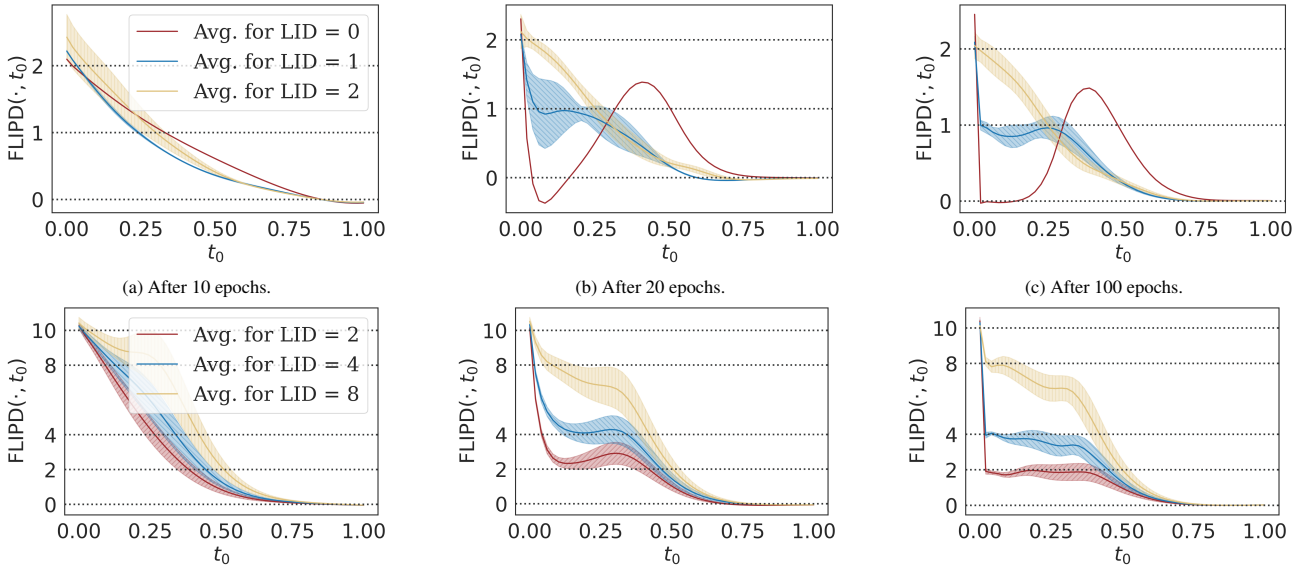


Figure 4. The FLIPD estimates on a Lollipop dataset from (Tempczyk et al., 2022).

D.3. A Simple Multiscale Experiment

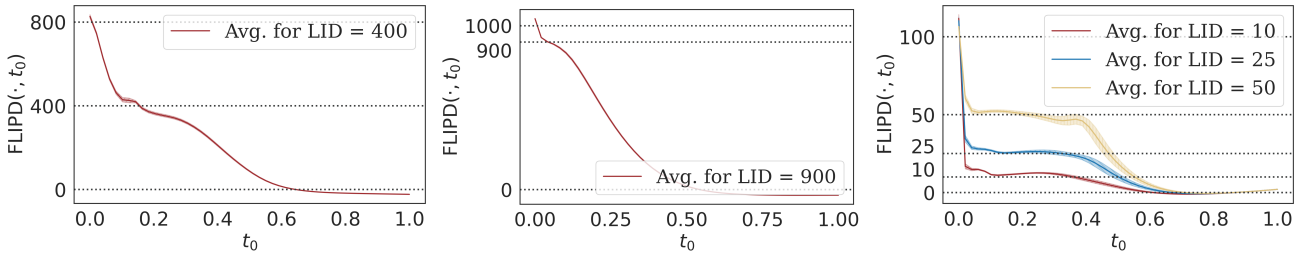
Tempczyk et al. (2022) argue that when setting δ, all the directions of data variation that have a log standard deviation below δ are ignored. Here, we make this connection more explicit.



(a) After 10 epochs. (b) After 20 epochs. (c) After 100 epochs.

(d) After 10 epochs. (e) After 20 epochs. (f) After 100 epochs.

Figure 5. Evolution of the FLIPD curve while training the DM to fit a Lollipop (top) and a manifold mixture $\mathcal{N}_2 + \mathcal{N}_4 + \mathcal{N}_8 \subseteq \mathbb{R}^{10}$ (bottom).



(a) $\mathcal{F}_{400} \subseteq \mathbb{R}^{800}$. (b) $\mathcal{U}_{900} \subseteq \mathbb{R}^{1000}$. (c) $\mathcal{N}_{10} + \mathcal{N}_{25} + \mathcal{N}_{50} \subseteq \mathbb{R}^{100}$.

Figure 6. The FLIPD curve for complex and high-dimensional manifolds.

We define a multivariate Gaussian distribution with a prespecified eigenspectrum for its covariance matrix: having three eigenvalues of 10^{-4}, three eigenvalues of 1, and four eigenvalues of 10^3. This ensures that the distribution is numerically 7d and that the directions and amount of data variation are controlled using the eigenvectors and eigenvalues of the covariance matrix.

For this multivariate Gaussian, the score function in Equation 5 can be written in closed form; thus, we evaluate FLIPD both with and without training a DM. We see in Figure 7 the estimates obtained in both scenarios match closely, with some deviations due to imperfect model fit, which we found matches perfectly when training for longer.

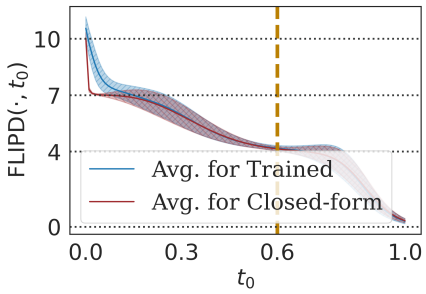


Figure 7. FLIPD curve for a multivariate Gaussian with controlled covariance eigenspectrum.

Apart from the initial knee at 7, which is expected, we find another at $t_0 = 0.6$ (corresponding to $e^{\delta} \approx 6.178$ with our hyperparameter setup in Table 4) where the value of FLIPD is 4. This indeed confirms that the estimator focuses solely on the 4d space characterized by the eigenvectors having eigenvalues of 10^3, and by ignoring the eigenvalues 10^{-4} and 1 which are both smaller than 6.178.

D.4. In-depth Analysis of the Synthetic Benchmark

Generating Manifold Mixtures To create synthetic data, we generate each component of our manifold mixture separately and then join them to form a distribution. All mixture components are sampled with equal probability in the final distribution. For a component with the intrinsic dimension of d, we first sample from a base distribution in d dimensions. This base distribution is isotropic Gaussian and Laplace for $\mathcal{N}_d$ and $\mathcal{L}_d$ and uniform for the case of $\mathcal{U}_d$ and $\mathcal{F}_d$. We then zero-pad these samples to match the ambient dimension D and perform a random $D \times D$ rotation on $\mathbb{R}^D$ (each component has one such transformation). For $\mathcal{F}_d$, we have an additional step to make the submanifold complex. We first initialize a neural spline flow (using the `nflows` library (Durkan et al., 2020)) with 5 coupling transforms, 32 hidden layers, 32 hidden blocks, and tail bounds of 10. The data is then passed through the flow, resulting in a complex manifold embedded in $\mathbb{R}^D$ with an LID of d. Finally, we standardize each mixture component individually and set their modes such that the pairwise Euclidean distance between them is at least 20. We then translate each component so its barycenter matches the corresponding mode, ensuring that data from different components rarely mixes, thus maintaining distinct LIDs.

LIDL Baseline Following the hyperparameter setup in (Tempczyk et al., 2022), we train 8 models with different noised-out versions of the dataset with standard deviations $e^{\delta_i} \in \{0.01, 0.014, 0.019, 0.027, 0.037, 0.052, 0.072, 0.1\}$. The normalizing flow backbone is taken from (Durkan et al., 2020), using 10 piecewise rational quadratic transforms with 32 hidden dimensions, 32 blocks, and a tail bound of 10. While (Tempczyk et al., 2022) uses an autoregressive architecture, we use coupling transforms for increased training efficiency and to match the training time of a single DM.

Setup We use model-free estimators from the `skdim` library (Bac et al., 2021) and across our experiments, we sample 10^6 points from the synthetic distributions for either fitting generative models or fitting the model-free estimators. We then evaluate LID estimates on a uniformly subsampled set of 2^{12} points from the original set of 10^6 points. Some methods are relatively slow, and this allows us to have a fair, yet feasible comparison. We do not see a significant difference even when we double the size of the subsampled set. We focus on four different model-free baselines for our evaluation: ESS (Johnsson et al., 2014), LPCA (Fukunaga & Olsen, 1971; Cangelosi & Goriely, 2007), MLE (Levina & Bickel, 2004; MacKay & Ghahramani, 2005), and FIS (Albergante et al., 2019), all with default settings. Note that FIS does not scale beyond 100 dimensions. Computing pairwise distances on high dimensions ($D \geq 800$) on all 10^6 samples takes over 24 hours even with 40 CPU cores. Therefore, for $D \geq 800$, we use the same 2^{12} subsamples we use for evaluation.

Evaluation We have three tables to summarize our analysis: (i) Table 6 shows the MAE of the LID estimates, comparing each datapoint's estimate to the ground truth at a fine-grained level; (ii) Table 7 shows the average LID estimate for synthetic manifolds with only one component, this average is typically used to estimate *global* intrinsic dimensionality in baselines; finally, (iii) looks at the concordance index (Harrell Jr et al., 1996) of estimates for cases with multiple submanifolds of different dimensionalities. Concordance indices for a sequence of LID estimates $\{\widehat{\text{LID}}\}_{n=1}^N$ are formally defined as follows:

$$\mathcal{C}\left(\{\text{LID}_n\}_{n=1}^N, \{\widehat{\text{LID}}_n\}_{n=1}^N\right) = \sum_{\substack{1 \leq n_1 \neq n_2 \leq N \\ \text{LID}_{n_1} \leq \text{LID}_{n_2}}} \mathbb{I}(\widehat{\text{LID}}_{n_1} \leq \widehat{\text{LID}}_{n_2}) / \binom{N}{2} \tag{48}$$

where $\mathbb{I}$ is the indicator function; a perfect estimator will have a $\mathcal{C}$ of 1. Instead of emphasizing the actual values of the LID estimates, this metric assesses how well the ranks of an estimator align with those of ground truth (Steck et al., 2007; Mayr & Petras, 2008; Teles, 2012; Kamkari et al., 2024a), thus evaluating LID as a "relative" measure of complexity.

Model-free Analysis Among model-free methods, LPCA and ESS show good performance in low dimensions (with LPCA being exceptionally good) but falter as dimensions increase. As we see in Table 6, while model-free methods outperform other estimators when $D < 100$, as D increases the paradigm shifts with model-based estimators outperforming the model-free ones. In addition, as shown in Table 7, all model-free baselines underestimate intrinsic dimensionality to some degree, with LPCA and MLE being particularly drastic for $D \geq 800$. We note that ESS performs relatively well, even beating model-based methods in some 800-dimensional scenarios. However, we note that the $\mathcal{C}$ values in Table 8 suggest it cannot rank data by LID as effectively as even the least performant MLE estimator.

Model-based Analysis Moving to model-based methods, we see in Table 8 that all perform poorly in ranking data based on LID except our FLIPD estimator. Remarkably, FLIPD achieves perfect $\mathcal{C}$ values among *all* datasets; further justifying it as a relative measure of complexity. We also note that while LIDL and NB provide better global estimates in Table 7 for

Table 5. MAE (lower is better). Rows show synthetic manifolds and columns represent different variations of our Fokker-Planck-based estimators.

Synthetic Manifold	FLIPD $t_0 = .05$	FLIPD `kneedle`	FPRegress `kneedle`	FPRegress $\delta_1 = -1$
Lollipop in $\mathbb{R}^2$	**0.142**	0.419	0.572	0.162
String within doughnut $\mathbb{R}^3$	**0.052**	0.055	0.398	0.377
Swiss Roll in $\mathbb{R}^3$	**0.053**	0.055	0.087	0.161
$\mathcal{L}_5 \subseteq \mathbb{R}^{10}$	**0.100**	0.169	1.168	0.455
$\mathcal{N}_{90} \subseteq \mathbb{R}^{100}$	0.501	**0.492**	3.142	0.998
$\mathcal{U}_{10} + \mathcal{U}_{30} + \mathcal{U}_{90} \subseteq \mathbb{R}^{100}$	3.140	**1.298**	5.608	10.617
$\mathcal{F}_{10} + \mathcal{F}_{25} + \mathcal{F}_{50} \subseteq \mathbb{R}^{100}$	14.37	**3.925**	16.32	21.01
$\mathcal{U}_{10} + \mathcal{U}_{80} + \mathcal{U}_{200} \subseteq \mathbb{R}^{800}$	39.54	**14.30**	30.06	29.99

high dimensions, they have worse MAE performance in Table 6. This once again suggests that at a local level, our estimator is superior compared to others, beating all baselines in 2 out of 3 groups of synthetic manifolds with $D \geq 100$ in Table 6. Finally, we see a curious flipped behaviour with NB estimators where they perform better when increasing dimensionality.

D.5. Ablations

We begin by evaluating the impact of using `kneedle`. Our findings, summarized in Table 5, indicate that while setting a small fixed t_0 is effective in low dimensions, the advantage of `kneedle` becomes particularly evident as the number of dimensions increases.

We introduce "FPRegress" which uses a single DM to perform the regression required by LIDL. To obtain LIDL estimates, one must solve an ODE m times for a set of log standard deviations $\delta_1 < \ldots < \delta_m$. For a given origin δ_1, we set $\delta_i = i \times \delta_1$ and use Euler's method with $m - 1$ steps to get $\log \hat{\varrho}(x, \delta_i)$ for $1 \leq i \leq 8$. Finally, a regression similar to LIDL is used to extract the LID estimate. This estimator is not only slower than FLIPD, but as we see in Table 5, it has worse MAEs.

We also tried combining it with `kneedle` by sweeping over the origin δ_1 and arguing that the estimates obtained from this method also exhibit knees. Despite some improvement in high-dimensional settings, Table 5 shows that even coupling it with `kneedle` does not help.

D.6. Improving the NB Estimators with `kneedle`

We recall that the NB estimator requires computing $\operatorname{rank} S(x)$, where $S(x)$ is a $K \times D$ matrix formed by stacking the scores $\hat{s}(\cdot, t_0)$. We set $t_0 = 0.01$ as it provides the most reasonable estimates. To compute $\operatorname{rank} S(x)$ numerically, Stanczuk et al. (2022) perform a singular value decomposition on $S(x)$ and use a cutoff threshold τ below which singular values are considered zero. Finding the best τ is challenging, so Stanczuk et al. (2022) propose finding the two consecutive singular values with the maximum gap. Furthermore, we see that sometimes the top few singular values are disproportionately higher than the rest, resulting in severe overestimations of the LID. Thus, we introduce an alternative algorithm to determine the optimal τ. For each τ, we estimate LID by thresholding the singular values. Sweeping 100 different τ values from 0 to 1000 at a geometric scale (to further emphasize smaller thresholds) produces estimates ranging from D (keeping all singular values) to 0 (ignoring all). As τ varies, we see that the estimates plateau over a certain range of τ. We use `kneedle` to detect this plateau because the starting point of a plateau is indeed a knee in the curve. This significantly improves the baseline, especially in high dimensions: see the third column of Tables 6, 7, and 8 compared to the second column.

Table 6. MAE (lower is better). Each row represents a synthetic dataset and each column represents an LID estimation method. Rows are split into groups based on the ambient dimension: the first group of rows shows toy examples; the second shows low-dimensional data with $D = 10$; the third shows moderate-dimensional data with $D = 100$; and the last two show high-dimensional data with $D = 800$ and $D = 1000$, respectively.

Synthetic Manifold	FLIPD `kneedle`	NB Vanilla	NB `kneedle`	LIDL	ESS	LPCA	MLE	FIS
Lollipop in $\mathbb{R}^2$	0.419	0.577	0.855	0.052	0.009	**0.000**	0.142	0.094
Swiss Roll in $\mathbb{R}^3$	0.055	0.998	0.016	0.532	0.017	**0.000**	0.165	0.018
Doughnut Mixture in $\mathbb{R}^3$	0.055	1.475	0.414	1.104	0.017	**0.000**	0.128	0.041
Summary (Toy Manifolds)	0.176	1.017	0.428	0.563	0.014	**0.000**	0.145	0.051
$\mathcal{N}_5 \subseteq \mathbb{R}^{10}$	0.084	5.000	0.005	0.071	0.061	**0.000**	0.441	0.206
$\mathcal{L}_5 \subseteq \mathbb{R}^{10}$	0.169	1.000	0.146	0.101	0.068	**0.000**	0.462	0.203
$\mathcal{U}_5 \subseteq \mathbb{R}^{10}$	0.324	4.994	0.933	0.123	0.153	**0.000**	0.451	0.186
$\mathcal{F}_5 \subseteq \mathbb{R}^{10}$	0.666	1.216	0.765	0.487	0.168	**0.000**	0.497	0.176
$\mathcal{N}_2 + \mathcal{N}_4 + \mathcal{N}_8 \subseteq \mathbb{R}^{10}$	0.287	5.706	1.078	0.308	0.156	**0.000**	0.406	0.206
$\mathcal{L}_2 + \mathcal{L}_4 + \mathcal{L}_8 \subseteq \mathbb{R}^{10}$	0.253	5.708	0.772	0.515	0.193	**0.001**	0.437	0.018
$\mathcal{U}_2 + \mathcal{U}_4 + \mathcal{U}_8 \subseteq \mathbb{R}^{10}$	0.586	5.677	3.685	0.363	0.331	**0.115**	0.540	0.222
$\mathcal{F}_2 + \mathcal{F}_4 + \mathcal{F}_8 \subseteq \mathbb{R}^{10}$	0.622	5.709	2.187	1.013	0.428	**0.115**	0.642	0.269
Summary (10-dimensional)	0.066	0.381	0.161	0.211	0.005	**0.000**	0.054	0.019
$\mathcal{U}_{10} \subseteq \mathbb{R}^{100}$	0.910	30.115	0.000	1.370	0.644	**0.000**	1.263	–
$\mathcal{U}_{30} \subseteq \mathbb{R}^{100}$	0.505	50.521	0.000	0.542	1.465	**0.002**	7.622	–
$\mathcal{U}_{90} \subseteq \mathbb{R}^{100}$	0.640	1.157	1.327	**0.332**	2.034	21.90	39.65	–
$\mathcal{N}_{30} \subseteq \mathbb{R}^{100}$	0.887	52.43	**0.000**	0.892	0.534	**0.000**	5.703	–
$\mathcal{N}_{90} \subseteq \mathbb{R}^{100}$	0.492	**0.184**	2.693	0.329	1.673	21.88	39.45	–
$\mathcal{F}_{80} \subseteq \mathbb{R}^{100}$	**1.869**	20.00	3.441	1.871	3.660	16.78	34.41	–
$\mathcal{U}_{10} + \mathcal{U}_{25} + \mathcal{U}_{50} \subseteq \mathbb{R}^{100}$	**0.868**	57.96	0.890	5.869	4.988	6.749	16.12	–
$\mathcal{U}_{10} + \mathcal{U}_{30} + \mathcal{U}_{90} \subseteq \mathbb{R}^{100}$	**1.298**	61.58	1.482	8.460	21.89	20.06	41.05	–
$\mathcal{N}_{10} + \mathcal{N}_{25} + \mathcal{N}_{50} \subseteq \mathbb{R}^{100}$	1.813	74.20	**0.555**	8.873	7.712	5.716	14.37	–
$\mathcal{F}_{10} + \mathcal{F}_{25} + \mathcal{F}_{50} \subseteq \mathbb{R}^{100}$	**3.925**	74.20	6.205	18.61	9.200	6.769	16.78	–
Summary (100-dimensional)	**1.321**	42.24	1.659	4.715	5.380	9.986	21.64	–
$\mathcal{U}_{200} \subseteq \mathbb{R}^{800}$	11.54	600.0	7.205	55.98	**5.116**	104.7	139.5	–
$\mathcal{F}_{400} \subseteq \mathbb{R}^{800}$	20.46	400.0	**10.15**	207.2	26.18	301.0	312.5	–
$\mathcal{U}_{10} + \mathcal{U}_{80} + \mathcal{U}_{200} \subseteq \mathbb{R}^{800}$	14.30	715.3	18.82	120.7	**2.349**	33.69	57.19	–
Summary (800-dimensional)	15.43	571.8	12.06	128.0	**11.22**	146.5	169.7	-
$\mathcal{N}_{900} \subseteq \mathbb{R}^{1000}$	**3.913**	100.0	24.38	10.45	76.40	801.0	774.4	–
$\mathcal{U}_{100} \subseteq \mathbb{R}^{1000}$	12.81	900.0	62.68	12.65	**2.055**	28.09	59.62	–
$\mathcal{U}_{900} \subseteq \mathbb{R}^{1000}$	12.81	100.0	**0.104**	24.90	75.17	801.0	762.2	–
$\mathcal{F}_{500} \subseteq \mathbb{R}^{1000}$	**21.77**	500.0	52.19	341.3	38.63	401.0	401.6	–
Summary (1000-dimensional)	**12.83**	400.0	34.84	97.33	48.06	507.8	499.5	–

Table 7. Average LID for manifolds with a single global intrinsic dimension. Rows represent uni-dimensional datasets and each column represents an LID estimation method.

Synthetic Manifold	FLIPD `kneedle`	NB Vanilla	NB `kneedle`	LIDL	ESS	LPCA	MLE	FIS
Swiss Roll in $\mathbb{R}^3$	2.012	2.998	1.984	2.527	2.008	**2.000**	2.013	2.003
$\mathcal{N}_5 \subseteq \mathbb{R}^{10}$	5.017	10.00	4.995	5.067	5.004	**5.000**	5.108	5.187
$\mathcal{L}_5 \subseteq \mathbb{R}^{10}$	4.968	6.000	4.854	5.089	4.985	**5.000**	5.123	5.182
$\mathcal{U}_5 \subseteq \mathbb{R}^{10}$	4.796	9.994	4.067	5.107	4.880	**5.000**	4.833	5.152
$\mathcal{F}_5 \subseteq \mathbb{R}^{10}$	4.722	6.216	4.461	5.482	4.890	**5.000**	4.829	5.131
$\mathcal{U}_{10} \subseteq \mathbb{R}^{100}$	11.68	40.12	10.00	11.29	9.361	**10.00**	8.905	–
$\mathcal{U}_{30} \subseteq \mathbb{R}^{100}$	31.01	80.52	30.00	30.08	28.54	**29.99**	22.38	–
$\mathcal{U}_{90} \subseteq \mathbb{R}^{100}$	89.54	91.16	91.32	**90.24**	88.27	68.10	50.35	–
$\mathcal{N}_{30} \subseteq \mathbb{R}^{100}$	30.79	82.43	30.00	30.85	29.67	**30.00**	24.38	–
$\mathcal{N}_{90} \subseteq \mathbb{R}^{100}$	89.88	90.18	92.62	**90.21**	88.89	68.12	50.55	–
$\mathcal{F}_{80} \subseteq \mathbb{R}^{100}$	77.97	100.0	83.23	**81.46**	76.35	63.22	45.59	–
$\mathcal{U}_{200} \subseteq \mathbb{R}^{800}$	211.5	800.0	**207.2**	256.0	195.3	95.30	60.51	–
$\mathcal{F}_{400} \subseteq \mathbb{R}^{800}$	454.7	800.0	**410.1**	607.2	373.8	99.00	87.53	–
$\mathcal{N}_{900} \subseteq \mathbb{R}^{1000}$	**890.3**	1000.	924.4	924.4	823.6	99.00	125.6	–
$\mathcal{U}_{100} \subseteq \mathbb{R}^{1000}$	135.76	1000.	162.7	112.6	**98.41**	71.91	40.38	–
$\mathcal{U}_{900} \subseteq \mathbb{R}^{1000}$	864.9	1000.	**900.1**	911.85	824.8	99.00	137.8	–
$\mathcal{F}_{500} \subseteq \mathbb{R}^{1000}$	582.7	1000.	**552.2**	841.3	461.4	99.00	98.38	–

Table 8. Concordance index (higher is better with 1.000 being the gold standard). Each row represents a mixture of multi-dimensional manifolds, and each column represents an LID estimation method. This table evaluates how accurately different estimators rank datapoints based on their LID.

Synthetic Manifold	FLIPD `kneedle`	NB Vanilla	NB `kneedle`	LIDL	ESS	LPCA	MLE	FIS
Lollipop in $\mathbb{R}^2$	**1.000**	0.426	0.394	0.999	**1.000**	**1.000**	**1.000**	**1.000**
Doughnut Mixture in $\mathbb{R}^3$	**1.000**	0.483	0.486	0.565	**1.000**	**1.000**	**1.000**	**1.000**
$\mathcal{N}_2 + \mathcal{N}_4 + \mathcal{N}_8 \subseteq \mathbb{R}^{10}$	**1.000**	0.341	0.725	0.943	**1.000**	**1.000**	**1.000**	**1.000**
$\mathcal{L}_2 + \mathcal{L}_4 + \mathcal{L}_8 \subseteq \mathbb{R}^{10}$	**1.000**	0.342	0.752	0.884	**1.000**	**1.000**	**1.000**	**1.000**
$\mathcal{U}_2 + \mathcal{U}_4 + \mathcal{U}_8 \subseteq \mathbb{R}^{10}$	**1.000**	0.334	0.462	0.903	**1.000**	**1.000**	**1.000**	**1.000**
$\mathcal{F}_2 + \mathcal{F}_4 + \mathcal{F}_8 \subseteq \mathbb{R}^{10}$	**1.000**	0.342	0.578	0.867	**1.000**	**1.000**	0.999	**1.000**
$\mathcal{U}_{10} + \mathcal{U}_{25} + \mathcal{U}_{50} \subseteq \mathbb{R}^{100}$	**1.000**	0.467	0.879	0.759	0.855	0.897	**1.000**	–
$\mathcal{U}_{10} + \mathcal{U}_{30} + \mathcal{U}_{90} \subseteq \mathbb{R}^{100}$	**1.000**	0.342	0.826	0.742	0.742	0.855	**1.000**	–
$\mathcal{N}_{10} + \mathcal{N}_{25} + \mathcal{N}_{50} \subseteq \mathbb{R}^{100}$	**1.000**	0.342	0.866	0.736	0.878	0.917	**1.000**	–
$\mathcal{F}_{10} + \mathcal{F}_{25} + \mathcal{F}_{50} \subseteq \mathbb{R}^{100}$	**1.000**	0.342	0.731	0.695	0.847	0.897	**1.000**	–
$\mathcal{U}_{10} + \mathcal{U}_{80} + \mathcal{U}_{200} \subseteq \mathbb{R}^{800}$	**1.000**	0.342	0.841	0.697	**1.000**	**1.000**	**1.000**	–

E. Image Experiments

E.1. FLIPD Curves and DM Samples

Figures 8 and 9 show DM-generated samples and the associated LID curves for 4096 samples from the datasets. In Figure 9, the FLIPD curves with MLPs, which have clearly discernible knees as predicted by the theory, are more reasonable than those generated with UNets. The fact that FLIPD estimates are worse for UNets is surprising given that MLPs produce worse-looking samples, as shown in Figure 8. However, comparing the first and second rows of Figure 8, it also becomes clear that MLP-generated images still match some characteristics of the true dataset, suggesting that they may still be capturing the image manifold in useful ways. To explain the surprisingly poor FLIPD estimates of UNets, we hypothesize that the convolutional layers in the UNet provide some inductive biases which, while helpful to produce visually pleasing images, might also encourage the network to over-fixate on high-frequency features which are not visually perceptible. For example, Kirichenko et al. (2020) showed how normalizing flows over-fixate on these features, and how this can cause them to assign large likelihoods to out-of-distribution data, even when the model produces visually convincing samples. We hypothesize that a similar underlying phenomenon might be at play here, and that DMs with UNets might be over-emphasizing "high-frequency" directions of variation in their LID estimates, even if they learn the semantic ones and thus produce pleasing images. However, an exploration of this hypothesis is outside the scope of our work, and we highlight once again that FLIPD remains a useful measure of complexity when using UNets.

E.2. UNet Architecture

We utilize UNet architectures from the `diffusers` library (von Platen et al., 2022). The DM setup mirrors that in Table 4 except for the score backbone. For greyscale images, we employ a convolutional block followed by two attention-based downsampling blocks. The channel sizes are 128, 256, and 256, respectively. For colour images, we use two convolutional downsampling blocks (each with 128 channels), followed by two attention downsampling blocks (each with 256 channels). In both cases, these blocks are inverted using their mirrored upsampling counterparts.

E.3. Images Sorted by FLIPD

Figures 10, 11, 12, and 13 show 4096 samples of CIFAR10, SVHN, MNIST, and FMNIST sorted according to their FLIPD estimate, showing a gradient transition from the least complex datapoints (e.g., the digit 1 in MNIST) to the most complex ones (e.g., the digit 8 in MNIST). We use MLPs for greyscales and UNets for colour images but see similar trends when switching between backbones.

E.4. How Many Hutchinson Samples are Needed?

Figure 14 compares the Spearman's rank correlation coefficient between FLIPD estimates while we use $k \in \{1, 50\}$ Hutchinson samples vs. computing the trace deterministically with D Jacobian-vector-products. We see that: (i) Hutchinson sampling is particularly well-suited for UNet backbones, having generally higher correlations compared to their MLP counterparts; (ii) as t_0 increases, the correlation becomes smaller, suggesting that the Hutchinson sample complexity increases at larger timescales; (iii) for small t_0, even one Hutchinson sample is enough to estimate LID; (iv) for the UNet backbone, 50 Hutchinson samples are enough and have a high correlation (larger than 0.8) even for t_0 as large as 0.5.

E.5. Multiscale Analysis on Images

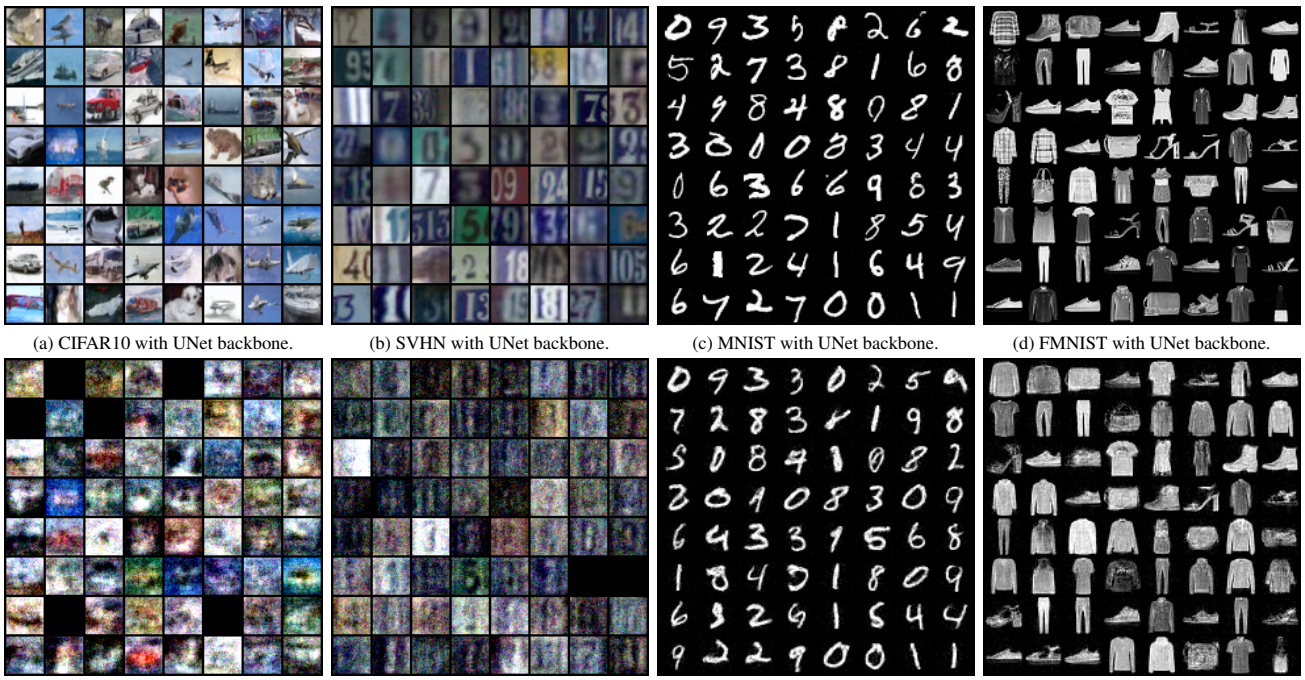
Figure 15 shows the correlation of FLIPD estimates with PNG compression for $t_0 \in (0, 1)$, indicating a consistently high correlation at small t_0, and a general decrease while increasing t_0. In addition, we see that UNet backbones correlate better with PNG.

Figures 16, 18, 20, and 22 show images with smallest and largest FLIPD estimates at different values of t_0 for the UNet backbone and Figures 17, 19, 21, and 23 show the same for the MLP backbone: (i) we see a clear difference in the complexity of top and bottom FLIPD estimates, especially for smaller t_0; this difference becomes less distinct as t_0 increases; (ii) interestingly, even for larger t_0 values with smaller PNG correlations, we qualitatively observe a clustering of the most complex datapoints at the end; however, the characteristic of this clustering changes. For example, see Figure 16 at $t_0 = 0.3$ or Figure 21 and Figure 23 at $t_0 = 0.8$, suggesting that FLIPD focuses on more coarse-grained measures of complexity at these scales; and finally (iii) while MLP backbones underperform in sample generation, their orderings are more meaningful, even showing coherent visual clustering up to $t = 0.8$ in all Figures 17, 19, 21, and 23.

E.6. Stable Diffusion

To test FLIPD with Stable Diffusion v1.5 (Rombach et al., 2022), which is known to have been finetuned on a subset of LAION-Aesthetics, we sampled 1600 images from LAION-Aesthetics-650k and computed FLIPD scores for each.

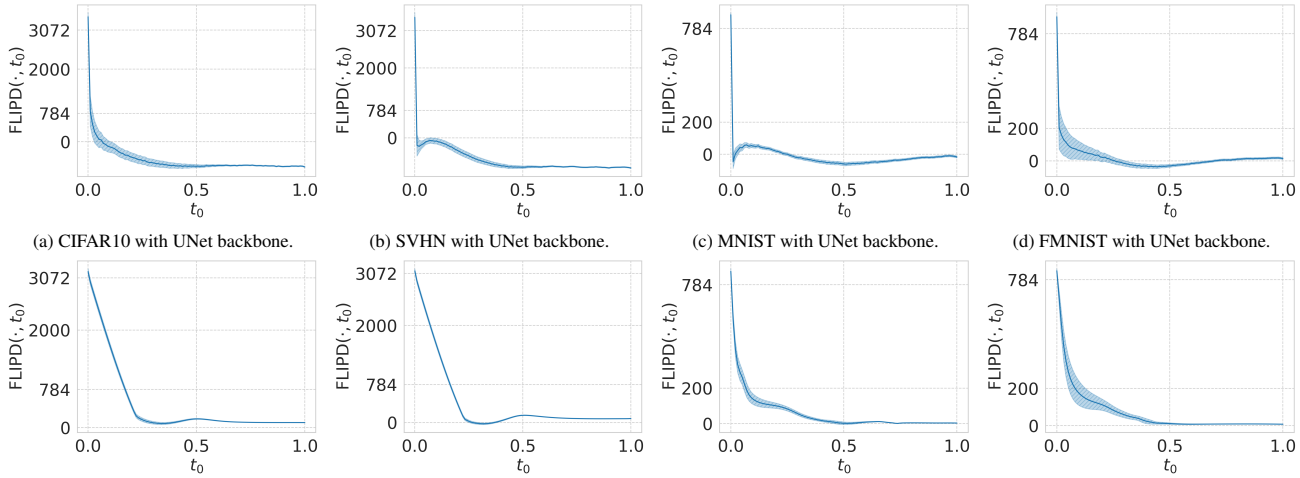
We ran FLIPD with $t_0 \in \{0.01, 0.1, 0.3, 0.8\}$ and a single Hutchinson trace sample. In all cases, FLIPD ranking clearly corresponded to complexity, though we decided upon $t_0 = 0.3$ as best capturing the "semantic complexity" of image contents. All 1600 images for $t = 0.3$ are depicted in Figure 24. For all timesteps, we show previews of the 8 lowest- and highest-LID images in Figure 25. Note that LAION is essentially a collection of URLs, and some are outdated. For the comparisons in Figure 25, we remove placeholder icons or blank images, which likely correspond to images that, at the time of writing this paper, have been removed from their respective URLs and which are generally given among the lowest LIDs.



(a) CIFAR10 with UNet backbone. (b) SVHN with UNet backbone. (c) MNIST with UNet backbone. (d) FMNIST with UNet backbone.

(e) CIFAR10 with MLP backbone. (f) SVHN with MLP backbone. (g) MNIST with MLP backbone. (h) FMNIST with MLP backbone.

Figure 8. Samples from DMs with different score network backbones, using the same seed for control. Despite the variation in backbones, images of the same cell in the grid (comparing top and bottom rows) show rough similarities, especially on CIFAR10 and SVHN.



(a) CIFAR10 with UNet backbone. (b) SVHN with UNet backbone. (c) MNIST with UNet backbone. (d) FMNIST with UNet backbone.

(e) CIFAR10 with MLP backbone. (f) SVHN with MLP backbone. (g) MNIST with MLP backbone. (h) FMNIST with MLP backbone.

Figure 9. FLIPD curves from all the different DMs with different score network backbones.

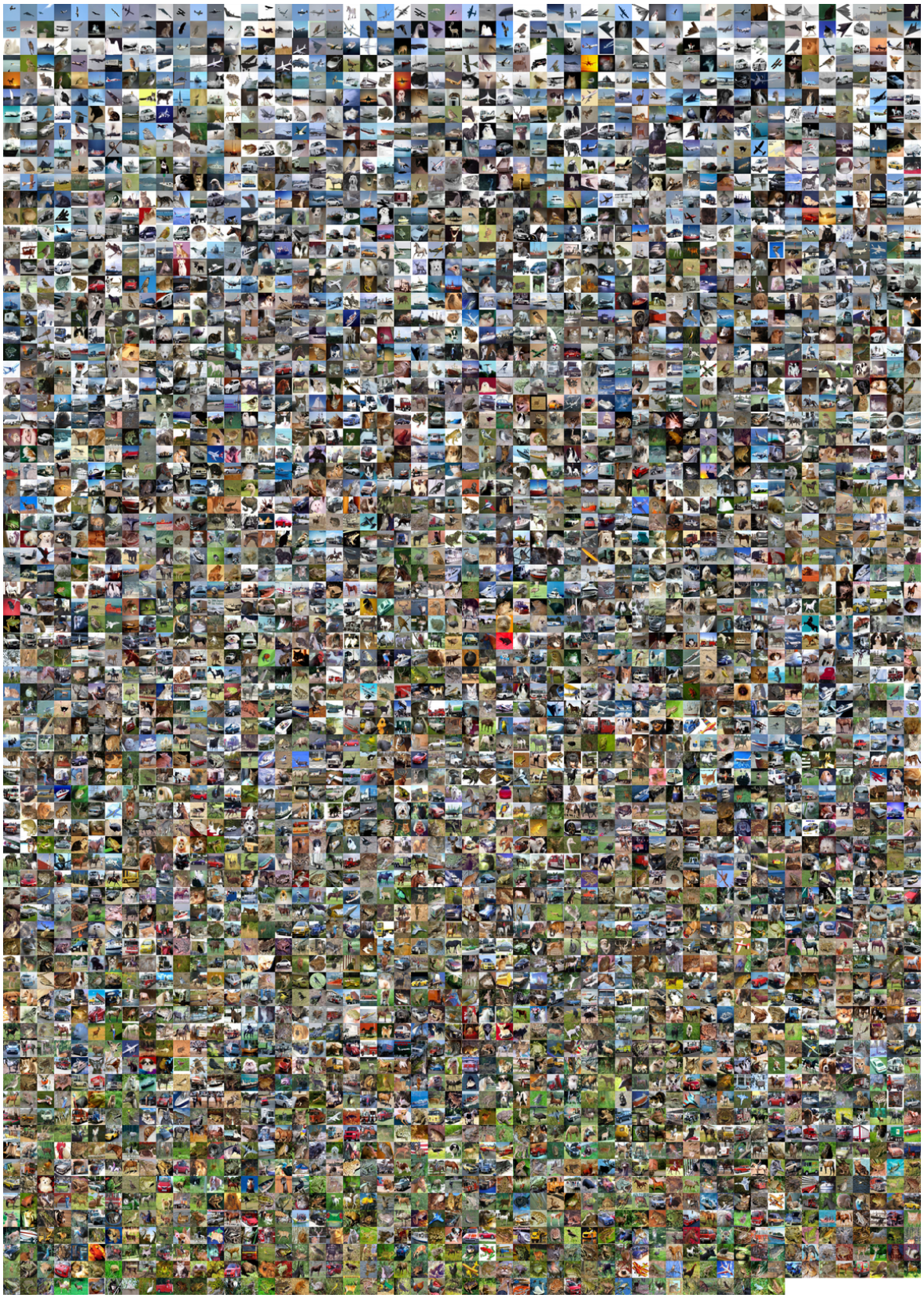


Figure 10. CIFAR10 sorted (left to right and top to bottom) by FLIPD estimate (UNet) at $t_0 = 0.01$.

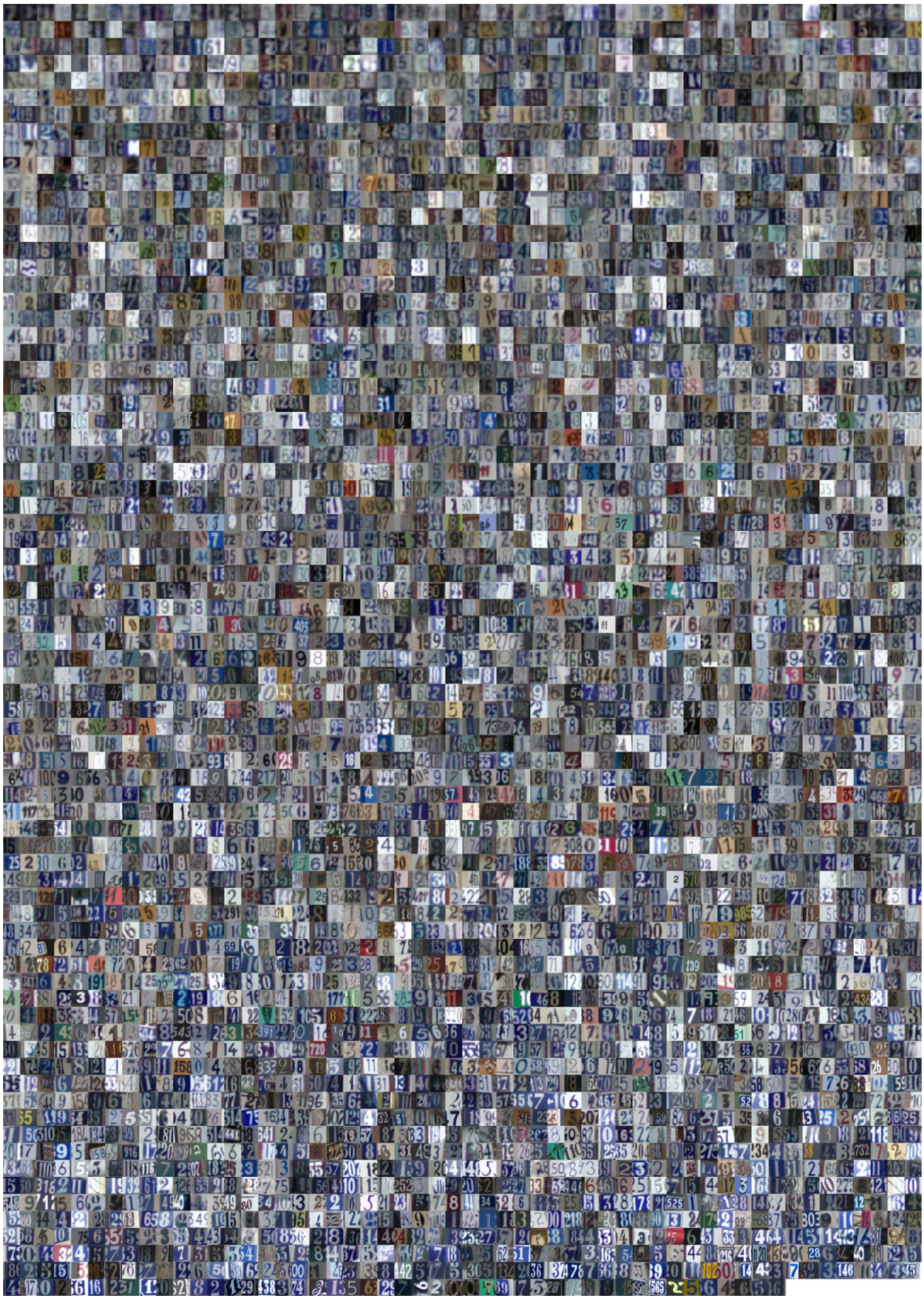


Figure 11. SVHN sorted (left to right and top to bottom) by FLIPD estimate (UNet) at $t_0 = 0.01$.

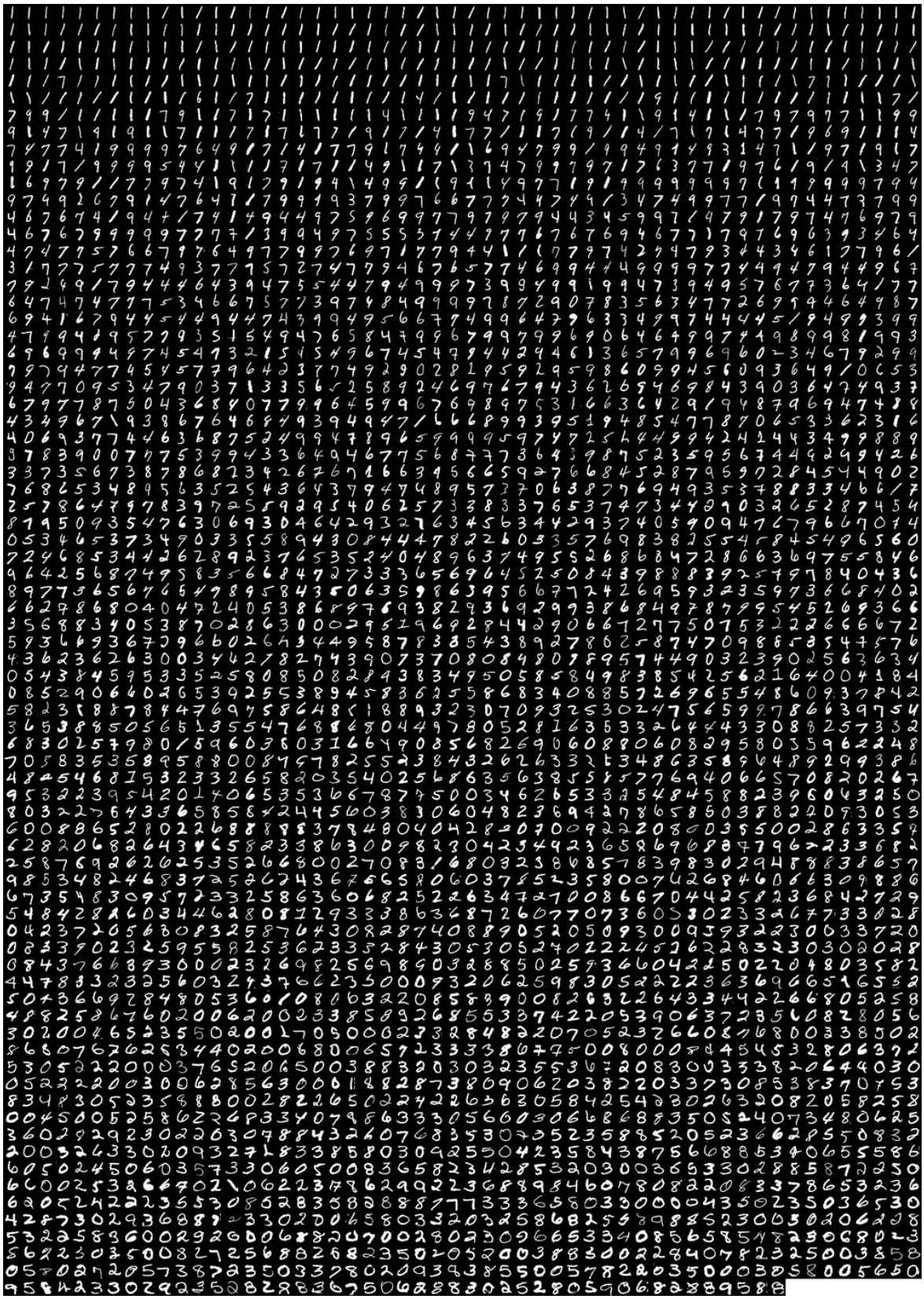


Figure 12. MNIST sorted (left to right and top to bottom) by FLIPD estimate (MLP) at $t_0 = 0.1$.

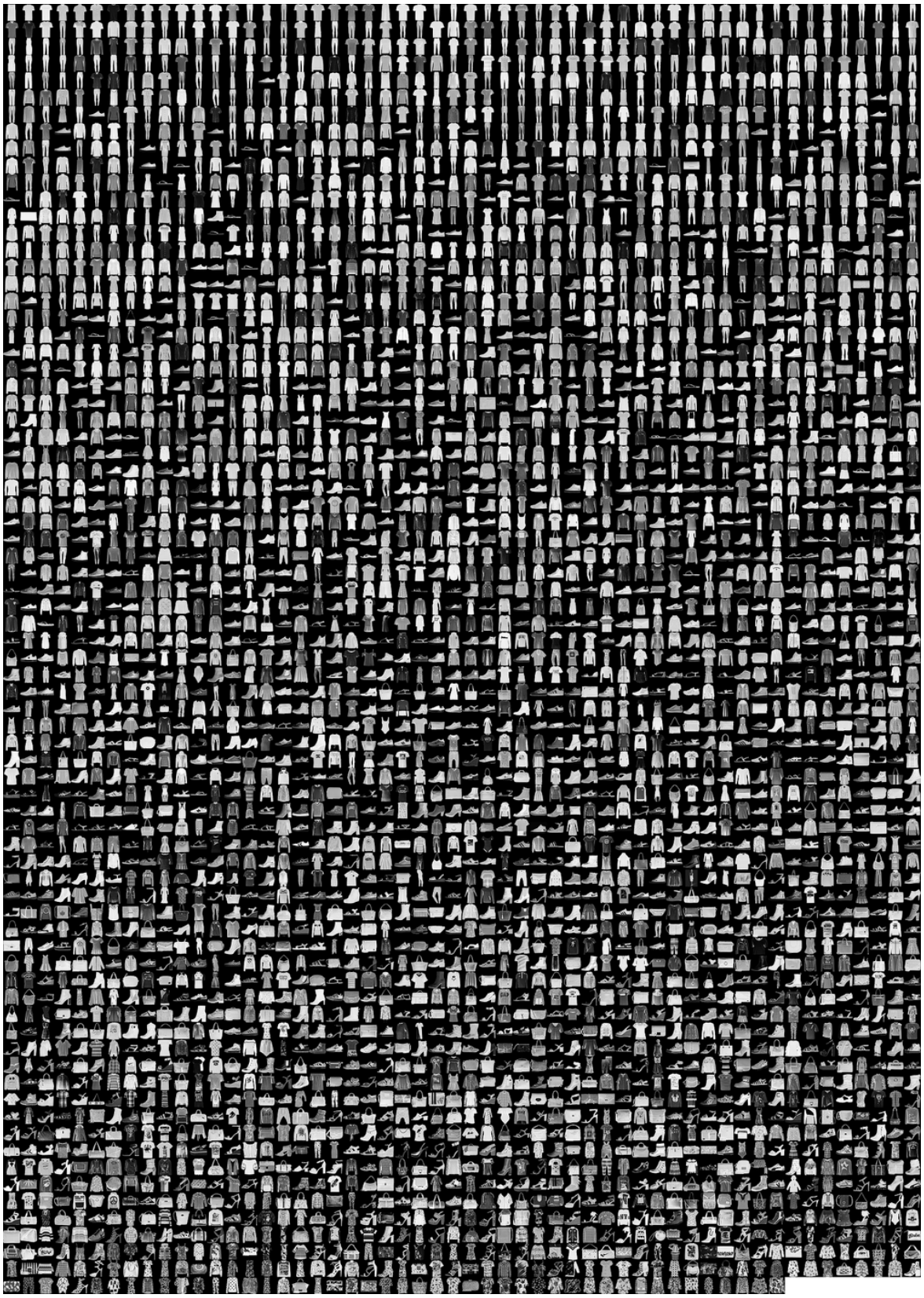
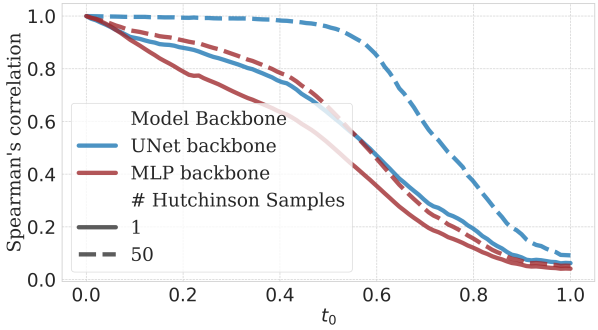
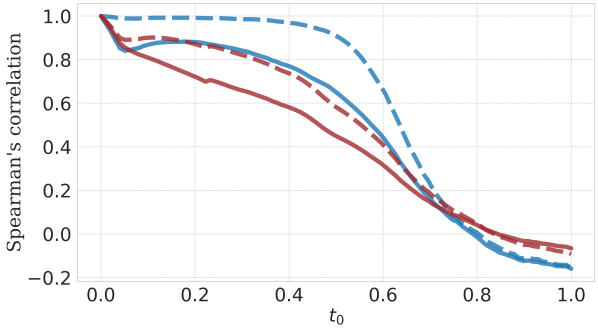


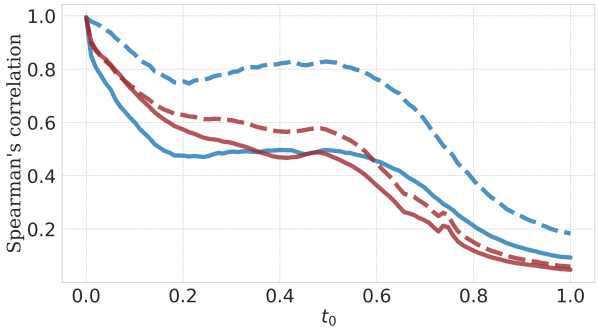
Figure 13. FMNIST sorted (left to right and top to bottom) by FLIPD estimate (MLP) at $t_0 = 0.1$.



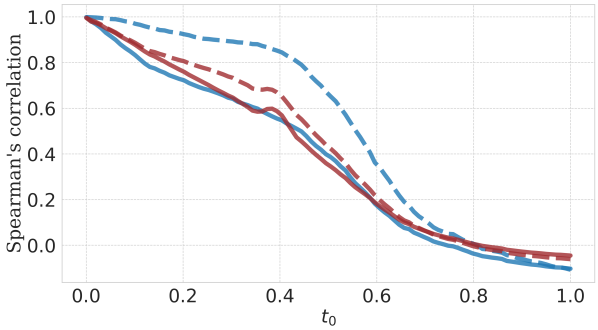
(a) Multiscale Spearman's correlation on CIFAR10.



(b) Multiscale Spearman's correlation on SVHN.

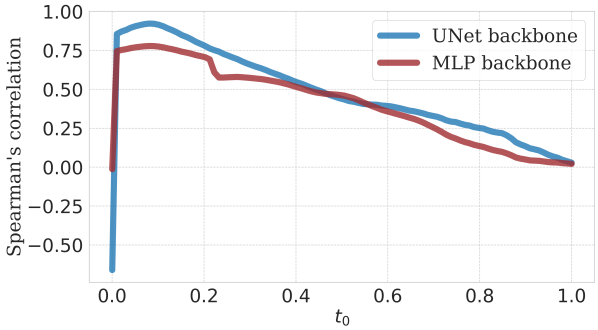


(c) Multiscale Spearman's correlation on MNIST.

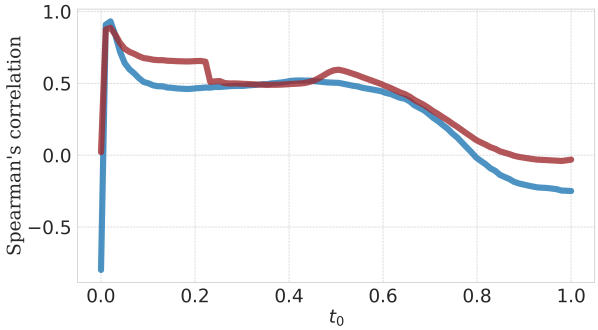


(d) Multiscale Spearman's correlation on FMNIST.

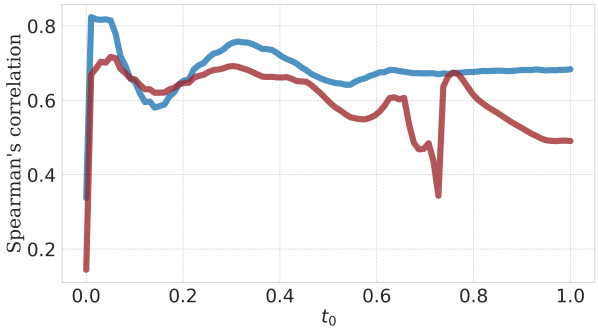
Figure 14. Spearman's correlation of FLIPD estimates while using different numbers of Hutchinson samples compared to computing the trace term of FLIPD deterministically with D Jacobian vector product calls. These estimates are evaluated at different values of $t_0 \in (0, 1)$ on four datasets using the UNet and MLP backbones.



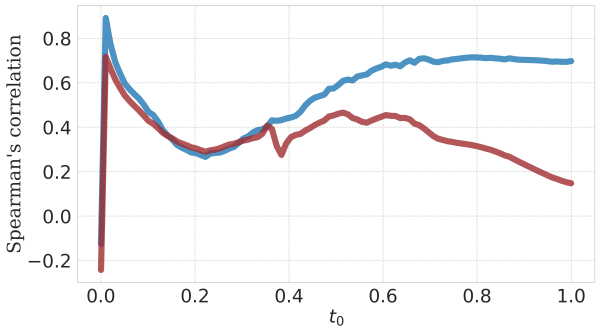
(a) Multiscale correlation with PNG on CIFAR10.



(b) Multiscale correlation with PNG on SVHN.



(c) Multiscale correlation with PNG on MNIST.



(d) Multiscale correlation with PNG on FMNIST.

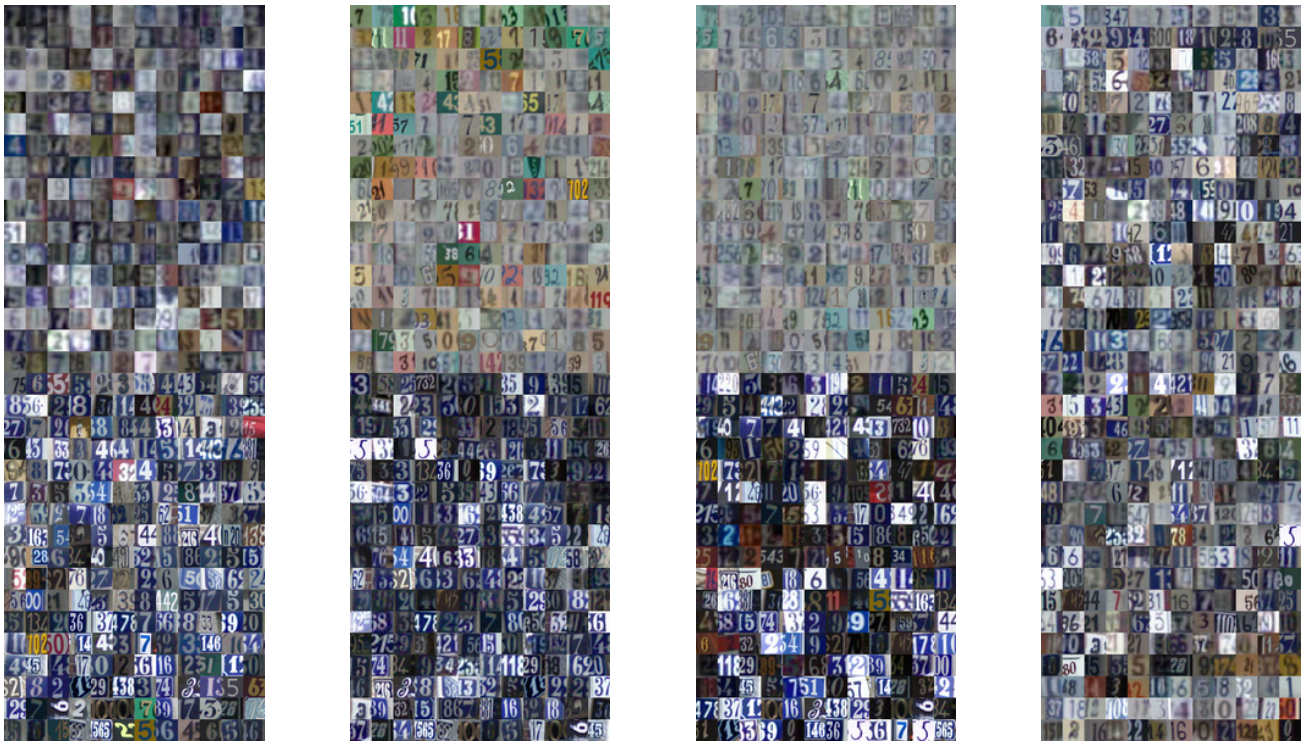
Figure 15. Spearman's correlation of FLIPD estimates with PNG for $t_0 \in (0, 1)$ on different backbones and different image datasets.

(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 16. The 204 smallest (top) and 204 largest (bottom) CIFAR10 FLIPD estimates with UNet evaluated at different t_0.

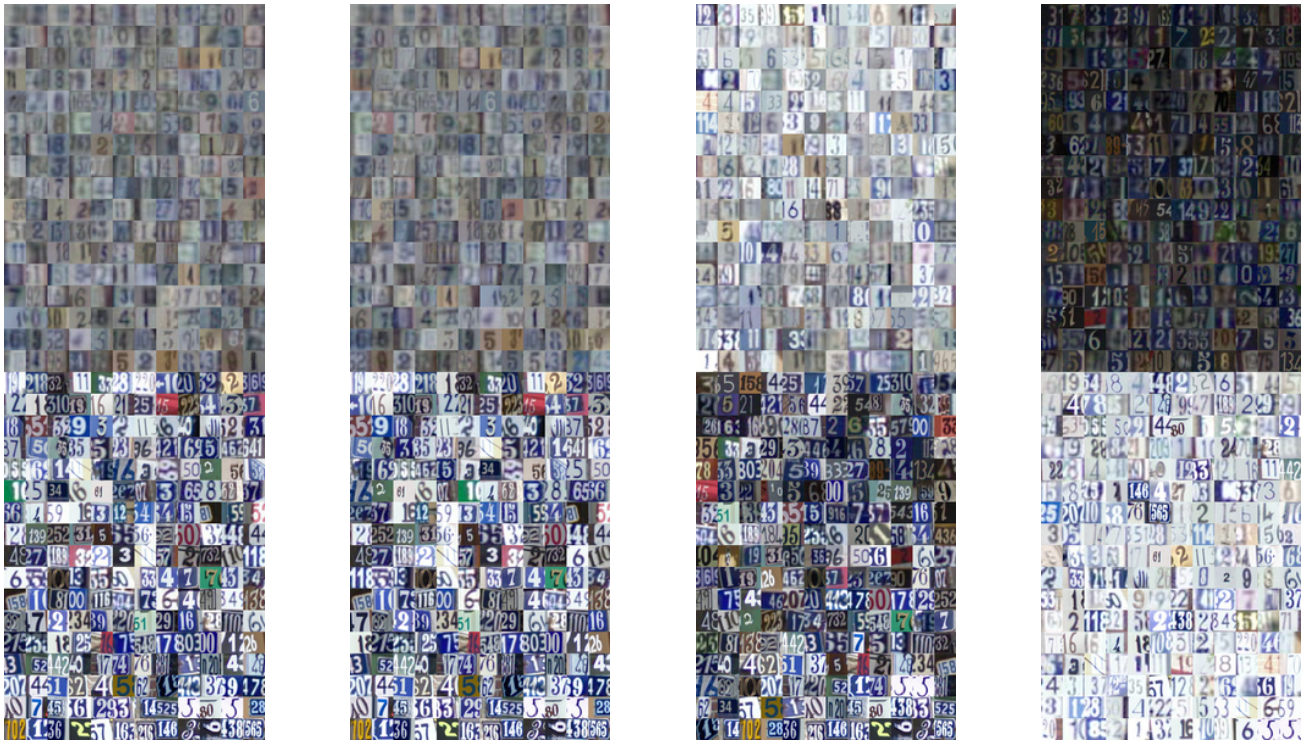
(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 17. The 204 smallest (top) and 204 largest (bottom) CIFAR10 FLIPD estimates with MLP evaluated at different t_0.



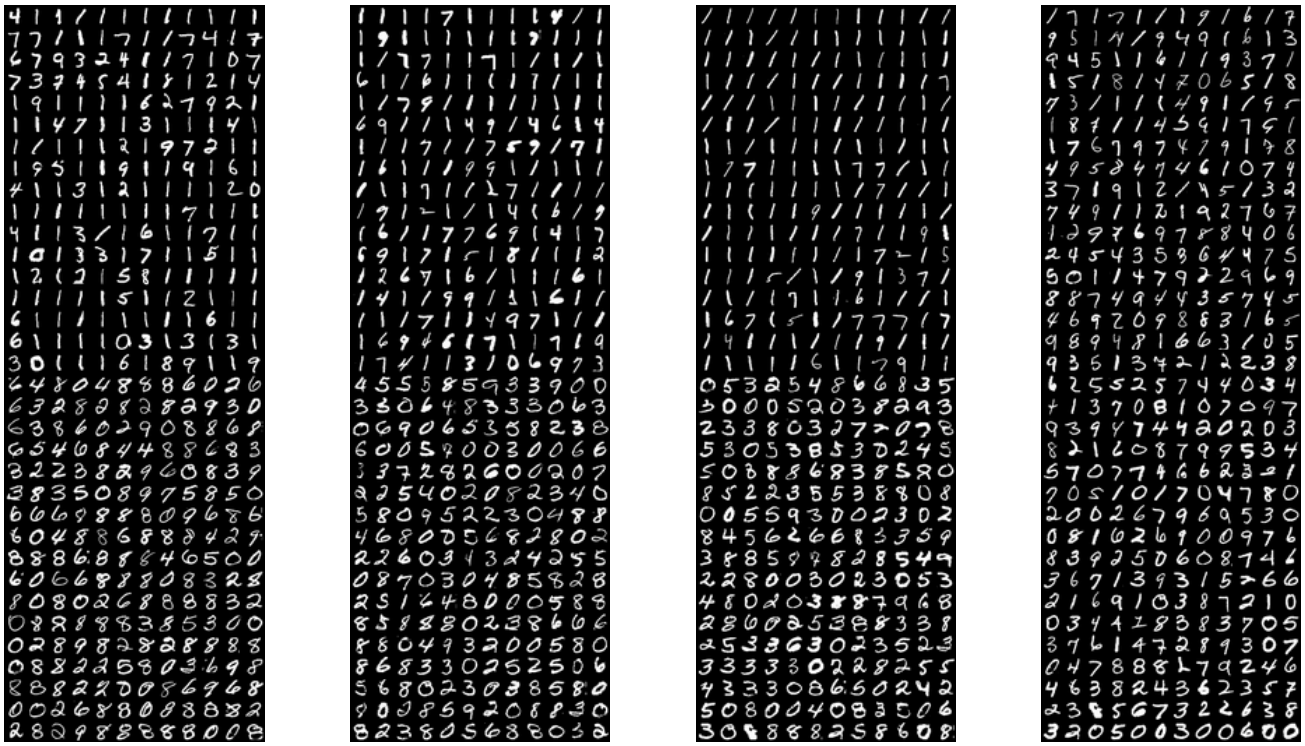
(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 18. The 204 smallest (top) and 204 largest (bottom) SVHN FLIPD with UNet estimates evaluated at different t_0.



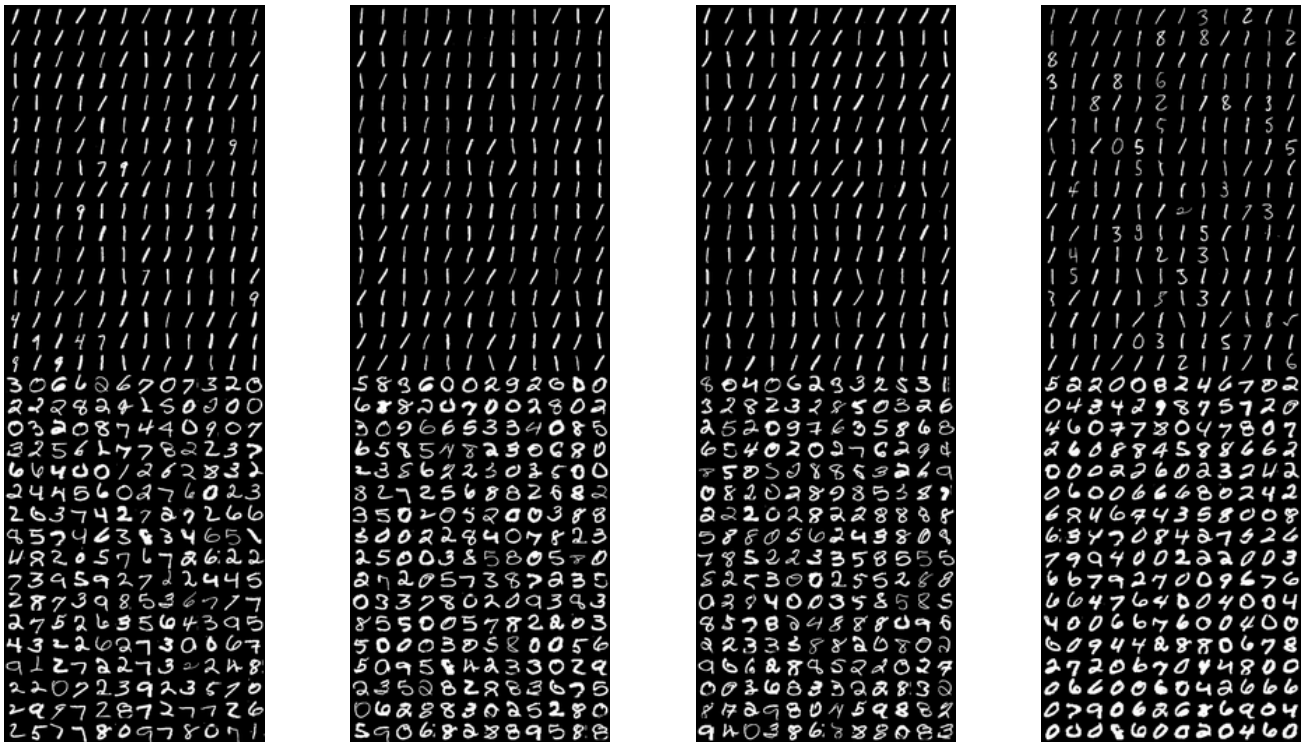
(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 19. The 204 smallest (top) and 204 largest (bottom) SVHN FLIPD with MLP estimates evaluated at different t_0.



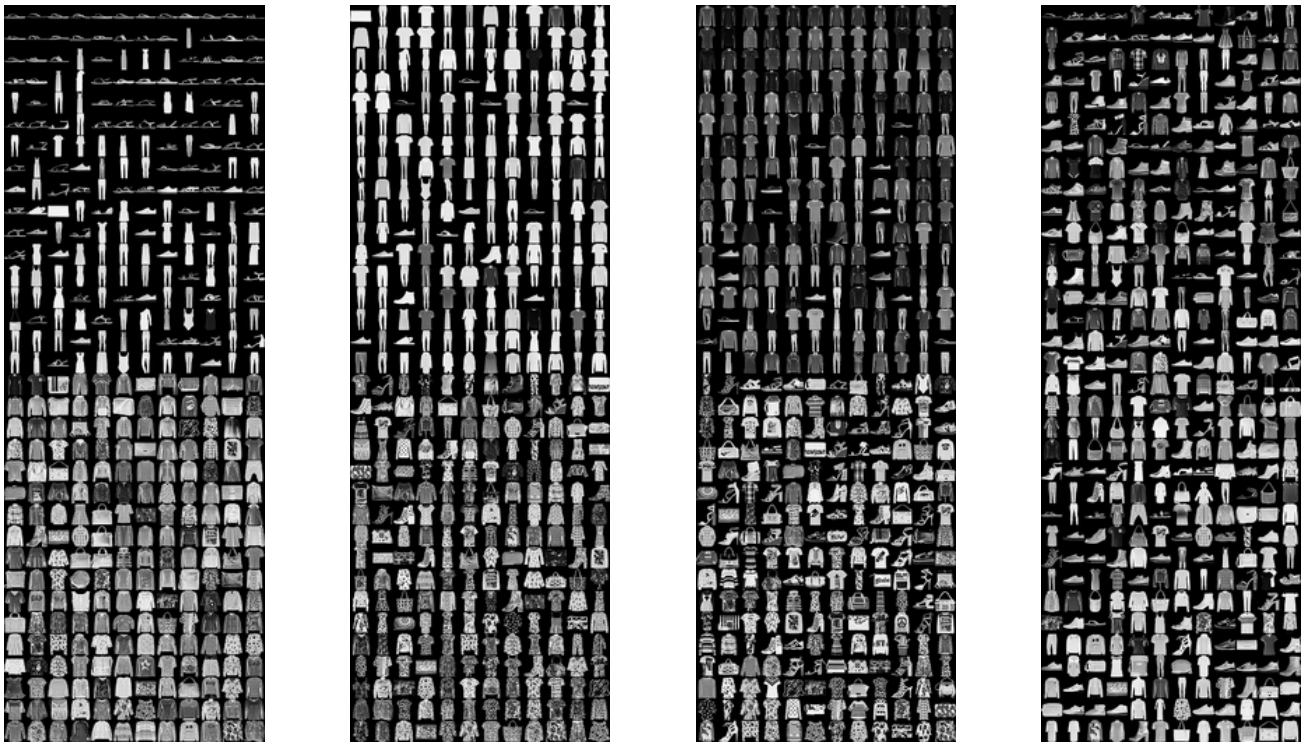
(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 20. The 204 smallest (top) and 204 largest (bottom) MNIST FLIPD with UNet estimates evaluated at different t_0.



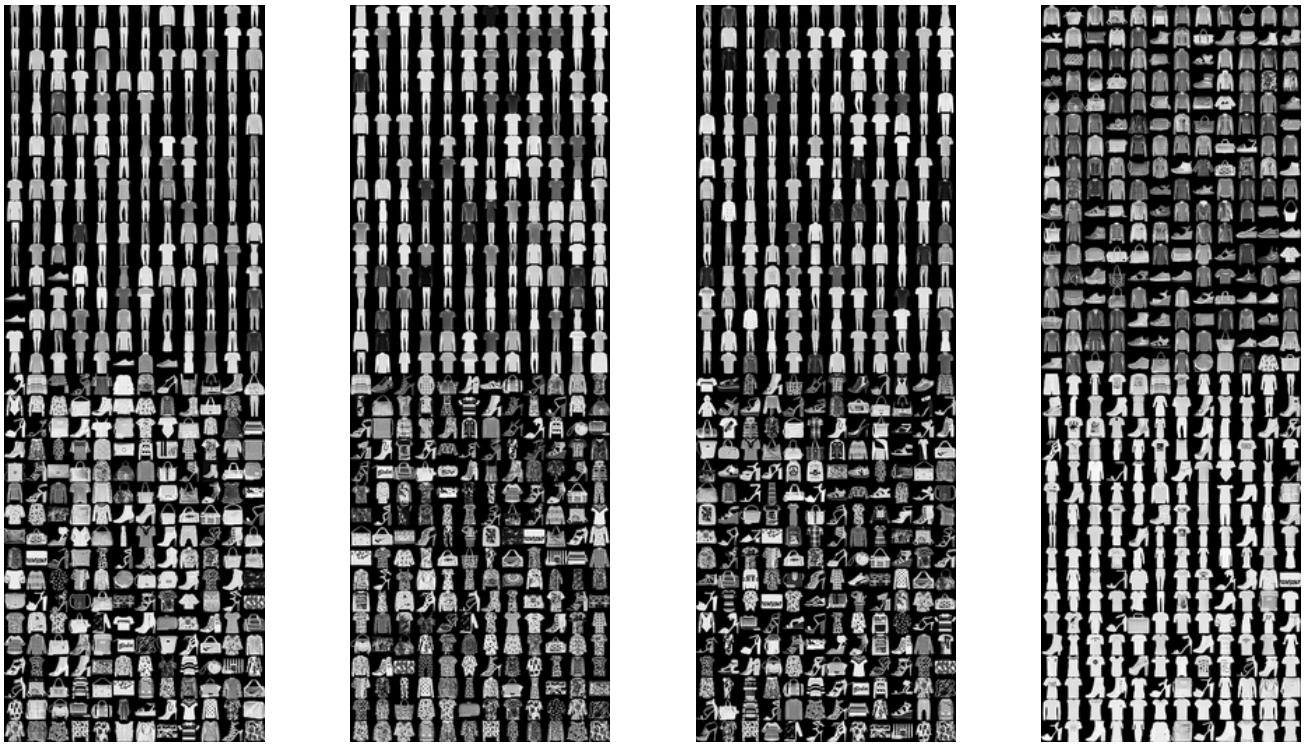
(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 21. The 204 smallest (top) and 204 largest (bottom) MNIST FLIPD with MLP estimates evaluated at different t_0.



(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 22. The 204 smallest (top) and 204 largest (bottom) FMNIST FLIPD estimates with UNet evaluated at different t_0.



(a) $t_0 = 0.01$. (b) $t_0 = 0.1$. (c) $t_0 = 0.3$. (d) $t_0 = 0.8$.

Figure 23. The 204 smallest (top) and 204 largest (bottom) FMNIST FLIPD estimates with MLP evaluated at different t_0.

Figure 24. 1600 images from LAION-Aesthetics-625K, sorted by FLIPD estimates with $t_0 = 0.3$.

(a) $t_0 = 0.01$.

(b) $t_0 = 0.1$.

(c) $t_0 = 0.8$.

Figure 25. The 8 lowest- and highest-FLIPD values out of 1600 from LAION-Aesthetics-625k evaluated at different values of t_0. Placeholder icons or blank images have been excluded from this comparison.