NOT A GAN: Flows for the data manifold

Johann Brehmer [1] **Kyle Cranmer** [1]

Abstract

We introduce ***no****rmalizing flows for simul****t****aneous* ***ma****nifold learnin****g*** ***a****nd density estimatio****n*** (NOT A GAN or even shorter $\mathcal{M}$-flows). This new class of generative models simultaneously learns the data manifold as well as a tractable probability density on that manifold. Combining aspects of normalizing flows, GANs, autoencoders, and energy-based models, $\mathcal{M}$-flows have the potential to represent datasets with a manifold structure more faithfully.

1. Introduction

Inferring a probability distribution from example data is a common problem that is increasingly tackled with deep generative models. Both generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma & Welling, 2014) are based on a lower-dimensional latent space and a learnable mapping to the data space, in essence describing a lower-dimensional data manifold embedded in the feature space. While they allow for efficient sampling, their probability density (or likelihood) is intractable, leading to a challenge for training and limiting their usefulness for inference tasks. On the other hand, normalizing flows (Dinh et al., 2015; 2019; Papamakarios et al., 2019; Rezende & Mohamed, 2015) are based on a latent space with the same dimensionality as the data space and a diffeomorphism; their tractable density permeates through the full data space and is not restricted to a lower-dimensional surface.

The flow approach may be unsuited to data that do not populate the full ambient data space they natively reside in, but are restricted to a lower-dimensional manifold (Feinman & Parthasarathy, 2019). Standard normalizing flows by construction assign a non-zero probability density to every point in the ambient space, they are thus not able to represent such a structure exactly and instead learn a smeared-out version with support off the data manifold. We illustrate this in the left panel of Fig. 1. While flows have been generalized from Euclidean spaces to Riemannian manifolds (Gemici et al., 2016), this approach has so far been limited to the case where the chart for the manifold is prescribed.

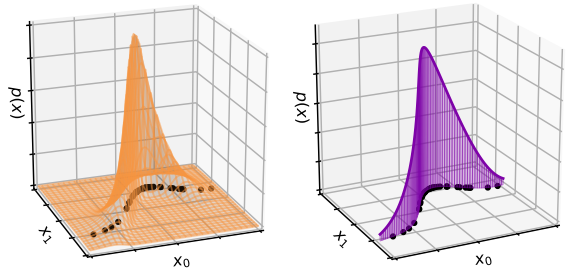

Figure 1. Sketch of how a standard flow in the ambient data space (left) and an $\mathcal{M}$-flow (right) model data.

We introduce *manifold-learning flows* ($\mathcal{M}$-flows): normalizing flows based on an injective, invertible map from a lower-dimensional latent space to the data space. $\mathcal{M}$-flows simultaneously learn the shape of the data manifold, provide a tractable bijective chart, and learn a probability density over the manifold, as sketched in the right panel of Fig. 1.

We discuss how this approach marries aspects of normalizing flows, GANs, autoencoders, and energy-based models (Arbel et al., 2020; Che et al., 2020; LeCun et al., 2006). Compared to standard flow-based generative models, $\mathcal{M}$-flows may more accurately approximate the true data distribution, avoiding probability mass off the data manifold. In addition, the model architecture naturally allows one to model a conditional density that lives on a fixed manifold. This should improve data efficiency in such situations as it is ingrained in the architecture and does not need to be learned. The lower-dimensional latent space may also reduce the complexity of the model, and the ability to project onto the data manifold provides dimensionality reduction, denoising, and out-of-distribution detection capabilities.

In this paper we summarize the main conceptual ideas and findings. For in-depth discussions, additional results, and implementation details, see a substantially extended version of this paper at Ref. (Brehmer & Cranmer, 2020).

[1]New York University. Correspondence to: Johann Brehmer <johann.brehmer@nyu.edu>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

2. Manifold-learning flows

Consider a true data-generating process that draws samples $x \in \mathcal{M}^* \subset X = \mathbb{R}^d$ according to $x \sim p^*(x)$, where $\mathcal{M}^*$ is a n-dimensional Riemannian manifold embedded in the d-dimensional data space X and $n < d$. We consider the two problems of estimating the density $p^*(x)$ as well as the manifold $\mathcal{M}^*$ given some training samples $\{x_i\} \sim p^*(x)$. For simplicity, we treat the manifold as topologically equivalent to $\mathbb{R}^n$ and assume that its dimensionality n is known; we discuss how these simplifications may be relaxed in the extended online version of this paper (Brehmer & Cranmer, 2020).

Manifold-learning flow ($\mathcal{M}$-flow). The $\mathcal{M}$-flow model is based on a latent space $U = \mathbb{R}^n$, which corresponds to the manifold coordinates, another latent space $V = \mathbb{R}^{d-n}$, which describes the off-the-manifold directions, and a diffeomorphism $f : U \times V \mapsto X$. We define the model manifold $\mathcal{M}$ through the level set

$$g : U \mapsto \mathcal{M} \subset X\,, \quad u \to g(u) = f(u, 0)\,. \tag{1}$$

In practice, we implement this transformation as a zero padding followed by a series of invertible transformations, $g = f_k \circ \cdots \circ f_1 \circ \mathrm{Pad}$, where Pad denotes padding a n-dimensional vector with $d - n$ zeros.

The base density $p_u(u)$ is modeled with an n-dimensional Euclidean normalizing flow h, which maps u to another latent variable $\tilde{u}$ with an associated tractable base density $p_{\tilde{u}}(\tilde{u})$. The induced probability density on the manifold is then given by

$$p_{\mathcal{M}}(x) = p_{\tilde{u}}(h^{-1}(g^{-1}(x)))\ \left|\det J_h(h^{-1}(g^{-1}(x)))\right|^{-1} \cdot \left|\det[J_g^T(g^{-1}(x))J_g(g^{-1}(x))]\right|^{-\frac{1}{2}}\,. \tag{2}$$

Sampling from an $\mathcal{M}$-flow is straightforward: one draws $\tilde{u} \sim p_{\tilde{u}}(\tilde{u})$ and pushes the latent variable forward to the data space as $u = h(\tilde{u})$ followed by $x = g(u) = f(u, 0)$, leading to data points on the manifold that consistently follow $x \sim p_{\mathcal{M}}(x)$.

As a final ingredient to the $\mathcal{M}$-flow approach, we add a prescription for evaluating arbitrary points $x \in X$, which may be off the manifold. g maps from a low-dimensional latent space to the data space and is therefore a decoder. We define a matching encoder g^{-1} as f^{-1} followed by a projection to the u component: $g^{-1} : X \mapsto U, x \to g^{-1}(x) = \mathrm{Proj}(f^{-1}(x))$ with $\mathrm{Proj}(u, v) = u$. This extends the inverse of g (which is so far only defined for $x \in \mathcal{M}$) to the whole data space X. Similar to an autoencoder, combining g and g^{-1} allows us to calculate a reconstruction error $\|x - x'\| = \|x - g(g^{-1}(x))\|$, which is zero if and only if $x \in \mathcal{M}$. Unlike for standard autoencoders, the encoder and decoder are exact inverses of each other as long as points on the manifold are studied.

For an arbitrary $x \in X$, an $\mathcal{M}$-flow thus lets us compute three quantities: the projection onto the manifold $x' = g(g^{-1}(x))$, which may be used as a denoised version of the input; the reconstruction error $\|x - x'\|$, which will be important for training, but may also be useful for anomaly detection or out-of-distribution detection; and the likelihood on the manifold after the projection, $p_{\mathcal{M}}(x')$. In this way, $\mathcal{M}$-flows separate the distance from the data manifold and the density on the manifold—two concepts that easily get conflated in an ambient flow. $\mathcal{M}$-flows embrace ideas of energy-based models for dealing with off-the-manifold issues, but still have a tractable, exact likelihood on the learned data manifold.

Manifold-learning flows with separate encoder ($\mathcal{M}_e$-flow). We also introduce a variant of the $\mathcal{M}$-flow model where instead of using the inverse f^{-1} followed by a projection as an encoder, we encode the data with a separate function $e : X \mapsto U$, $x \to e(x)$. This encoder is not restricted to be invertible or to have a tractable Jacobian, potentially increasing the expressiveness of the network. Just as in the $\mathcal{M}$-flow approach, for a given data point x an $\mathcal{M}_e$-flow model returns a projected point onto the learned manifold, a reconstruction error, and the likelihood on the manifold evaluated after the projection. The added expressivity of this encoder comes at the price of potential inconsistencies between encoder and decoder, which the training procedure will have to try to penalize, exactly as for standard autoencoders.

3. Related work

Our work is closely related to a number of different probabilistic and generative models. This includes normalizing flows in the ambient data space, which we will label as ambient flows (AF) (Dinh et al., 2015; 2019; Papamakarios et al., 2019; Rezende & Mohamed, 2015), as well as flows on prescribed manifolds (FOMs) (Bose et al., 2020; Gemici et al., 2016; Rezende et al., 2020), GANs (Goodfellow et al., 2014), and VAEs (Kingma & Welling, 2014). Here we want to highlight two particularly closely related works:

Pseudo-invertible encoder (PIE). The PIE model (Beitler et al., 2019) splits the latent variables of an ambient flow into two types u and v with different base densities and relies on the training to align one class of latent variables with the manifold coordinates and the other with the off-the-manifold directions. While the density is very similar to that of an AF, the authors of Ref. (Beitler et al., 2019) propose sampling from this

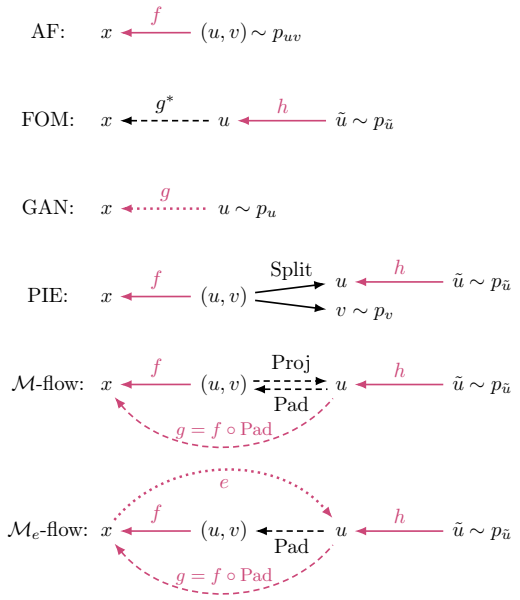

Figure 2. Schematic relation between data x and various latent variables $u, v, \tilde{u}$ in different generative models. Red arrows show learnable transformations, black arrows prescribed ones. Solid lines denote invertible bijections, dashed lines invertible injections, dotted lines unrestricted transformations.

model by $u \sim p_u(u)$ and applying $x = f(u, 0)$, i. e. fixing the off-the-manifold latents to $v = 0$. Note, however, that the density defined by this sampling procedure is *not* the same as the tractable density $p_x(x)$. We use PIE as a baseline our experiments.

Injective flows. Relaxed injective probability flows (Kumar et al., 2020) are similar to our $\mathcal{M}_e$-flows. Instead of using invertible flow transformations, they enforce the invertibility of the decoder g by bounding the norm of the Jacobian of an otherwise unrestricted transformation. While this makes the transformation locally invertible, it does not eliminate the possibility of multiple points in latent space pointing to the same point in data space. Furthermore, the inverse of g and the likelihood are not tractable for unseen data points, limiting the usefulness for inference tasks. The approaches also differ in the treatment of points off the learned manifold and the training.

In Fig. 2 and Tbl. 1 we compare our new $\mathcal{M}$-flows and $\mathcal{M}_e$-flows to various other generative models.

4. Efficient training

Maximum likelihood is not enough. Since the $\mathcal{M}$-flow density is tractable, maximum likelihood is an obvious candidate for a training objective. However, the situation is more subtle as the $\mathcal{M}$-flow model describes the density *after projecting onto the learned manifold*. The definition

Model	Manifold	Chart	Tractable density	Restr. to $\mathcal{M}$
AF	no manifold	×	✓	×
FOM	prescribed	✓	✓	✓
GAN	learned	×	×	✓
VAE	learned	×	only ELBO	(×)
PIE	learned	✓	✓	(×)
$\mathcal{M}$-flow	learned	✓	✓	✓
$\mathcal{M}_e$-flow	learned	✓	✓	✓

Table 1. Comparison of generative models. The last column classifies models by whether their density is restricted to the manifold; parantheses indicate an alternative sampling procedure that can generate data restricted to the manifold, but which does not correspond to the model density.

of the data variable in the likelihood hence depends on the weights ϕ_f of the manifold-defining transformation f, and a comparison of naive likelihood values between different configurations of ϕ_f is meaningless. Instead of thinking of a likelihood function $p(x|\phi_f, \phi_h)$, where ϕ_h are the weights defining h, it is instructive to think of a family of likelihood functions $p_{\phi_f}(x|\phi_h)$ parameterized by the different ϕ_f.

Training $\mathcal{M}$-flows by simply maximizing the naive likelihood $p(x|\phi_f, \phi_h)$ therefore does not incentivize the network to learn the correct manifold. As an extreme example, consider a model manifold that is perpendicular to the true data manifold. Since this configuration allows the $\mathcal{M}$-flow to project all points to a small region of high density on the model manifold, this pathological configuration may lead to a high naive likelihood value.

A second challenge is the computational efficiency of evaluating the $\mathcal{M}$-flow density in Eq. (2). While this quantity is in principle tractable, it cannot be computed as efficiently as the likelihood of an ambient flow. The underlying reason is that since the Jacobian J_g is not square, it is not obvious how the determinant $\det J_g^T J_g$ can be decomposed further, at least when we compose an $\mathcal{M}$-flow out of the typical elements of ambient flows like coupling layers. Evaluating the $\mathcal{M}$-flow density then requires the computation of all entries of the Jacobians of the individual transformation. While this cost can be reasonable for the evaluation of a limited number of test samples, it can be prohibitively expensive during training. Since the computational cost grows with increasing data dimensionality d, training by maximizing $\log p_{\mathcal{M}}$ does not scale to high-dimensional problems.

Separate manifold and density training (M/D). We can solve both problems at once by separating the training into two phases. In the *manifold phase*, we update only the parameters of f, which through a level set also define the manifold $\mathcal{M}$ and the chart g. Similarly to autoencoders, we minimize the reconstruction error from the projection onto the manifold, $\|x - g(g^{-1}(x))\|$. For the $\mathcal{M}_e$-flow model, the parameters of the encoder e are also updated during this

Model	Polynomial surface			Particle physics			Images			
	Distance	RE	MMD	Closure	RE	$\log p(\theta^*)$	$n=2$ FID	$n=64$ FID	$n=64$ $\log p(\theta^*)$	CelebA FID
AF	0.005	–	0.071	**0.0019**	–	−3.94	58.3	24.0	0.17	**33.6**
PIE	0.035	1.278	0.131	0.0023	2.054	−4.68	139.5	32.2	−6.40	75.7
$\mathcal{M}$-flow	**0.002**	**0.003**	0.020	0.0045	**0.012**	−1.71	43.9	**20.8**	**2.67**	37.4
$\mathcal{M}_e$-flow	**0.002**	**0.002**	**0.007**	0.0046	0.029	**−1.44**	**43.5**	23.7	1.81	35.8

Table 2. Selected experimental results. Error bars are available in the online version of this paper (Brehmer & Cranmer, 2020).

phase. In the *density phase*, we update only the parameters of h by maximum likelihood. h only affects the density p_u, we are thus keeping the manifold fixed during this phase.

Such a training procedure is not prone to the gradient flow aligning the manifold with pathological configurations. Moreover, training only h by maximum likelihood does not require computing the expensive terms in the model likelihood. The loss in the density phase is given by the log of Eq. (2); only the last term in that equation is expensive to evaluate, but it does not depend on the parameters of h and does not contribute to the gradient updates in this phase! We can therefore train the parameters of h by minimizing only the first two terms, which can be evaluated efficiently.

Likelihood evaluation. For high-dimensional data, evaluating the likelihood in Eq. (2) can become so expensive that even the likelihood evaluations at test time must be limited. Several approximate inference techniques may reduce this computational cost. The issue can be side-stepped entirely in the common case where the density (but not the manifold) is conditional on some model parameters θ and the downstream goal is inferring these model parameters θ. In this case, the $\mathcal{M}$-flow setup enables the fast and exact computation of likelihood *ratios* and MCMC acceptance probabilities.

5. Experiments

Datasets. We consider a variety of synethetic and real-world datasets:

1. A two-dimensional manifold described by a polynomial surface equation embedded in $\mathbb{R}^3$ with a Gaussian mixture model.
2. The invariant probability measure of the Lorenz system (Lorenz, 1963). This system of ordinary differential equations is known for its chaotic behaviour, with many solutions of the system tending to a set that has a Hausdorff dimension of approximately 2.06 (Guckenheimer & Sparrow, 1984; Viswanath, 2004).
3. Simulation-based (likelihood-free) inference (Cranmer et al., 2020) for a real-world particle physics problem, where observations populate a 14-dimensional manifold embedded in $\mathbb{R}^{40}$.
4. Synthetic images on an n-dimensional manifold. We generate them from a StyleGAN2 model (Karras et al., 2019) trained on the FFHQ dataset (Karras et al., 2018), keeping all but n of the GAN latent variables fixed. We study one dataset with $n = 2$ and one with $n = 64$, in both cases downsampling the images to a resolution of 64×64.
5. The real-world CelebA-HQ (Karras et al., 2018) dataset, downsampled to a resolution of 64×64.

Architectures. We study $\mathcal{M}$-flow and $\mathcal{M}_e$-flow models as well as AF and PIE baselines. All models are based on rational-quadratic neural spline flows (Durkan et al., 2019).

Metrics. A common metric for flow-based models is the likelihood evaluated on a test set, but such a comparison is not meaningful in our context: since the $\mathcal{M}$-flow variants evaluate the likelihood after projecting to a learned manifold, the data variable in the likelihood is different for every model and the likelihoods of different models may not even have the same units. Instead, we consider a range of metrics that gauges the qualities of samples generated from the different flows, including the distance to the true data manifold in the polynomial surface dataset, a domain-specific closure test for the particle physics problem, and the Fréchet Inception Distance (FID) (Heusel et al., 2017; Lucic et al., 2017) for the image datasets. The quality of the manifolds learned by $\mathcal{M}$-flow, $\mathcal{M}_e$-flow, and PIE models is measured through the reconstruction error (RE) of projecting test samples to the learned manifolds. Finally, in datasets where the density depends on some parameter θ, we study inference on this parameter with MCMC based on the flow likelihoods. We evaluate the models either based on the maximum mean discrepancy (MMD) (Gretton et al., 2012) between the approximate and true posterior, or through the log posterior evaluated at the true parameter point used to generate synthetic observations ($\log p(\theta^*)$).

Results. Table 2 shows a selection of metrics across the four datasets. We find that $\mathcal{M}$-flow and $\mathcal{M}_e$-flow models learn manifolds of a higher quality than PIE, beat AF and PIE baselines on some of the generative metrics, and outperform the baselines on all inference tasks. In Fig. 3 we illustrate some of the learned models.

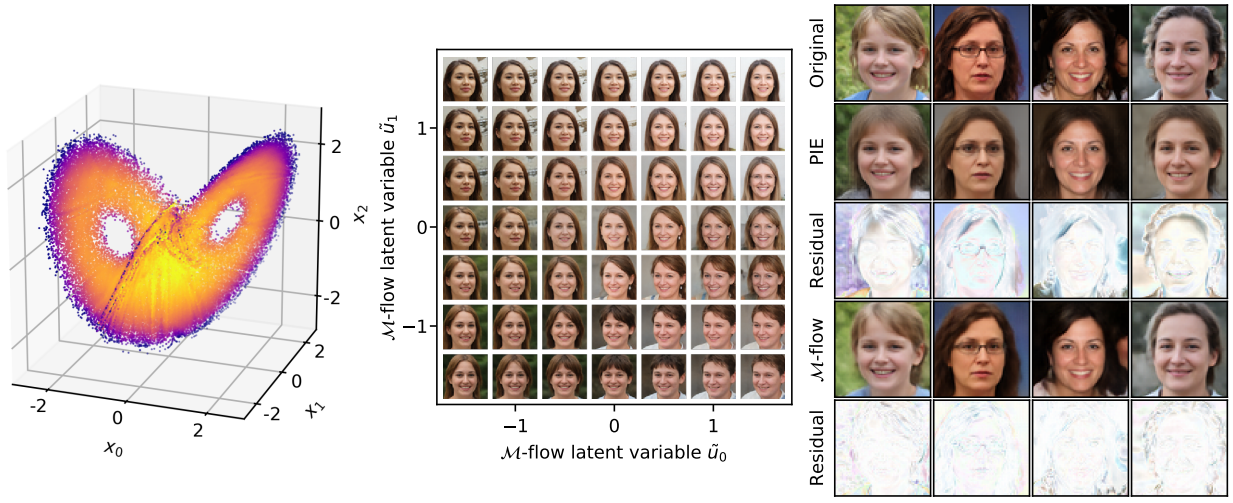

Figure 3. Selected results. **Left**: Manifold and density for the Lorenz attractor as learned by an $\mathcal{M}$-flow model. **Middle**: 2-D image manifold learned by an $\mathcal{M}$-flow model. **Right**: 64-D manifold test samples, their projection to the learned manifold, and residuals.

6. Conclusions

We introduced manifold-learning flows ($\mathcal{M}$-flows), a new type of generative model that combines aspects of normalizing flows, autoencoders, and GANs. $\mathcal{M}$-flows describe data with a tractable probability density over a lower-dimensional manifold embedded in data space. Both the manifold and the density are learned from data. We discussed how these models should be trained (hint: maximum likelihood alone is not enough) and demonstrated their potential in a first range of experiments.

Problems in which data populate a lower-dimensional manifold embedded in a high-dimensional feature space are almost everywhere. The manifold structure may be particularly explicit in some scientific problems, while the success of GANs on numerous datasets is testament to the presence of low-dimensional data manifolds in other domains. Manifold-learning flows may help us unify generative and inference tasks in a way that is well-suited to the structure of the data.

Acknowledgements

We would like to thank Jens Behrmann, Kyunghyun Cho, Jack Collins, Jean Feydy, Siavash Golkar, Michael Kagan, Gilles Louppe, George Papamakarios, Merle Reinhart, Frank Rösler, John Tamanas, Antoine Wehenkel, and Andrew Wilson for useful discussions. We are grateful to Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios for publishing their excellent neural spline flow codebase (Durkan et al., 2019), which we used extensively in our analysis. Similarly, we want to thank George Papamakarios, David Sterratt, and Iain Murray for publishing their Sequential Neural Likelihood code (Papamakarios et al., 2018), parts of which were used in the evaluation steps in our experiments.

We are grateful to the authors and maintainers of DELPHES 3 (Demin & Selvaggi, 2014), GEOMLOSS (Feydy et al., 2019), JUPYTER (Kluyver et al., 2016), MADGRAPH5_AMC (Alwall et al., 2014), MADMINER (Brehmer et al., 2020), MATPLOTLIB (Hunter, 2007), NUMPY (van der Walt et al., 2011), PYTHIA8 (Sjöstrand et al., 2008), PYTORCH (Paszke et al., 2017), PYTORCH-FID (Seitzer, 2020), SCIKIT-LEARN (Pedregosa et al., 2011), and SCIPY (Jones et al., 2001). We are grateful for the support of the National Science Foundation under the awards ACI-1450310, OAC-1836650, and OAC-1841471, as well as by the Moore-Sloan data science environment at NYU. This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise; through the NYU Courant Institute of Mathematical Sciences; and by the Scientific Data and Computing Center at Brookhaven National Laboratory.

References

Alwall, J., Frederix, R., Frixione, S., Hirschi, V., Maltoni, F., Mattelaer, O., Shao, H. S., Stelzer, T., Torrielli, P., and Zaro, M. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *Journal of High Energy Physics*, 2014(7):79, 2014. ISSN 10298479. doi: 10.1007/JHEP07(2014)079.

Arbel, M., Zhou, L., and Gretton, A. KALE: When Energy-Based Learning Meets Adversarial Training. *arXiv:2003.05033*, 2020.

Beitler, J. J., Sosnovik, I., and Smeulders, A. {PIE}:

Pseudo-Invertible Encoder, 2019. URL https://openreview.net/forum?id=SkgiX2Aqtm.

Bose, A. J., Smofsky, A., Liao, R., Panangaden, P., and Hamilton, W. L. Latent Variable Modelling with Hyperbolic Normalizing Flows. *arXiv:2002.06336*, 2020.

Brehmer, J. and Cranmer, K. Flows for simultaneous manifold learning and density estimation. *arXiv:2003.13913*, 2020.

Brehmer, J., Kling, F., Espejo, I., and Cranmer, K. MadMiner: Machine Learning-Based Inference for Particle Physics. *Computing and Software for Big Science*, 4(1), 2020. ISSN 2510-2036. doi: 10.1007/s41781-020-0035-2.

Che, T., Zhang, R., Sohl-Dickstein, J., Larochelle, H., Paull, L., Cao, Y., and Bengio, Y. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling. *arXiv:2003.06060*, 2020.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. In *Proceedings of the National Academy of Sciences*. National Academy of Sciences, 2020. doi: 10.1073/pnas.1912789117.

Demin, P. and Selvaggi, M. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:57, 2014. ISSN 1029-8479. doi: 10.1007/JHEP02(2014)057.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural Spline Flows. *Advances in Neural Information Processing Systems*, pp. 7509–7520, 2019.

Feinman, R. and Parthasarathy, N. A Linear Systems Theory of Normalizing Flows. *arXiv:1907.06496*, jul 2019.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690, 2019.

Gemici, M. C., Rezende, D., and Mohamed, S. Normalizing Flows on Riemannian Manifolds. *arXiv:1611.02304*, 2016.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Guckenheimer, J. and Sparrow, C. The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors. *The American Mathematical Monthly*, 1984. ISSN 00029890. doi: 10.2307/2322694.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv:1706.08500*, 2017.

Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

Jones, E., Oliphant, T., Peterson, P., and Others. {SciPy}: Open source scientific tools for {Python}, 2001. URL http://www.scipy.org/.

Karras, T., Laine, S., and Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv:1812.04948*, 2018.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and Improving the Image Quality of StyleGAN. *arXiv:1912.04958*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and al., E. Jupyter Notebooks - a publishing format for reproducible computational workflows. In *ELPUB*, 2016.

Kumar, A., Poole, B., and Murphy, K. Regularized Autoencoders via Relaxed Injective Probability Flow. *arXiv:2002.08927*, 2020.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Lorenz, E. N. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. Are GANs Created Equal? A Large-Scale Study. *arXiv:1711.10337*, 2017.

Papamakarios, G., Sterratt, D. C., and Murray, I. Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows. 2018.

Papamakarios, G., Nalisnick, E., Jimenez Rezende, D., Mohamed, S., and Lakshminarayanan, B. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv:1912.02762*, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *32nd International Conference on Machine Learning, ICML 2015*, 2015. ISBN 9781510810587.

Rezende, D. J., Papamakarios, G., Racanière, S., Albergo, M. S., Kanwar, G., Shanahan, P. E., and Cranmer, K. Normalizing Flows on Tori and Spheres. *arXiv:2002.02428*, 2020.

Seitzer, M. Fréchet Inception Distance (FID score) in PyTorch, 2020. URL https://github.com/mseitzer/pytorch-fid.

Sjöstrand, T., Mrenna, S., and Skands, P. A brief introduction to PYTHIA 8.1. *Computer Physics Communications*, 178(11):852–867, 2008. ISSN 00104655. doi: 10.1016/j.cpc.2008.01.036.

van der Walt, S., Colbert, S. C., and Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering*, 13(2):22–30, 2011. doi: 10.1109/MCSE.2011.37.

Viswanath, D. The fractal property of the Lorenz attractor. *Physica D: Nonlinear Phenomena*, 190:115–128, 2004. doi: 10.1016/j.physd.2003.10.006.