The Journal of Open Source Software

normflows: A PyTorch Package for Normalizing Flows

Vincent Stimper [1,2], **David Liu**[1], **Andrew Campbell**[1], **Vincent Berenz**[2], **Lukas Ryll**[1], **Bernhard Schölkopf** [2], **and José Miguel Hernández-Lobato**[1]

1 University of Cambridge, Cambridge, United Kingdom **2** Max Planck Institute for Intelligent Systems, Tübingen, Germany

DOI: 10.21105/joss.05361

Software

- Review
- Repository
- Archive

Editor: Marcel Stimberg

Reviewers:

- @matejgrcic
- @kazewong

Submitted: 18 February 2023
Published: 24 June 2023

Summary

Normalizing flows model probability distributions through an expressive tractable density (D. Rezende & Mohamed, 2015; Esteban G. Tabak & Turner, 2013; Esteban G. Tabak & Vanden-Eijnden, 2010). They transform a simple base distribution, such as a Gaussian, through a sequence of invertible functions, which are referred to as layers. These layers typically use neural networks to become very expressive. Flows are ubiquitous in machine learning and have been applied to image generation (Grcić et al., 2021; Kingma & Dhariwal, 2018), text modeling (Wang & Wang, 2019), variational inference (D. Rezende & Mohamed, 2015), approximating Boltzmann distributions (Noé et al., 2019), and many other problems (Kobyzev et al., 2021; Papamakarios et al., 2021).

Here, we present `normflows`, a Python package for normalizing flows. It allows to build normalizing flow models from a suite of base distributions, flow layers, and neural networks. The package is implemented in the popular deep learning framework PyTorch (Paszke et al., 2019), which simplifies the integration of flows in larger machine learning models or pipelines. It supports most of the common normalizing flow architectures, such as Real NVP (Dinh et al., 2017), Glow (Kingma & Dhariwal, 2018), Masked Autoregressive Flows (Papamakarios et al., 2017), Neural Spline Flows (Durkan et al., 2019; Müller et al., 2019), Residual Flows (Chen et al., 2019), and many more. The package can be easily installed via `pip` and the code is publicly available on GitHub.

Statement of need

`normflows` focuses on flows that are composed of discrete transformations, as opposed to continuous normalizing flows (Chen et al., 2018; Papamakarios et al., 2021). There are several other packages implementing discrete normalizing flows, such as TensorFlow Probability (Dillon et al., 2017) for TensorFlow, `distrax` (Babuschkin et al., 2020) for JAX, and `nflows` (Durkan et al., 2020) and `FrEIA` (Ardizzone et al., 2018-2022) for PyTorch. However, none of them support the two popular flow architectures, residual and autoregressive flows, within a single package, while we do so.

Moreover, `normflows` stands out by providing tools that are often used when approximating Boltzmann distributions. First, sampling layers needed for Stochastic Normalizing Flows (Nielsen et al., 2020; Wu et al., 2020) are included. Second, Neural Spline Flows on circular coordinates are supported (D. J. Rezende et al., 2020), which can be combined with standard coordinates on bounded or unbounded intervals. They are needed when modeling the internal coordinates of molecules consisting of angles and lengths (Midgley et al., 2023). Furthermore, there is an extension for `normflows` that adds Boltzmann distributions as targets as well as flow layers converting between Cartesian and internal coordinates (Stimper et al., 2023).

Our package has already been used in several scientific projects and publications (Campbell et

al., 2021; Midgley et al., 2023; Stimper et al., 2022). Due to its modular nature, `normflows` can be easily extended to house new flow layers, base distributions, or other tools. For instance, (Stimper et al., 2022) extends the package by adding resampled base distributions, which overcome an architectural weakness of normalizing flows and make them more expressive.

Examples

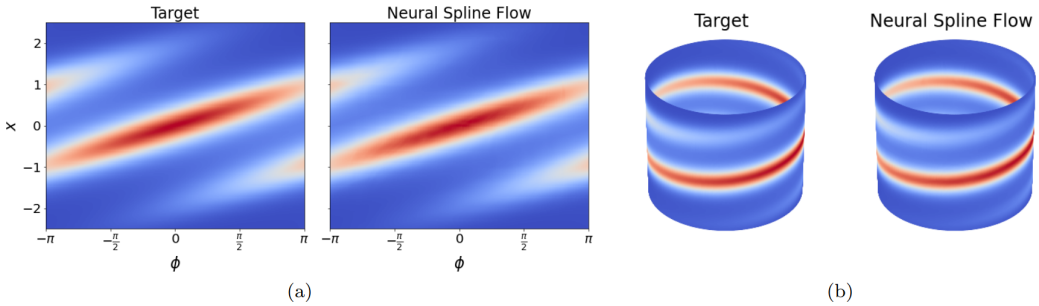

Figure 1: Target density defined on a cylinder surface, having an unbounded coordinate x and a circular coordinate ϕ. A Neural Spline Flow is fit to it, being almost indistinguishable from the target. (a) shows the densities in 2D and (b) is a visualization on the cylinder surface.

In the GitHub repository of our package, we provide various examples illustrating how to use it. We show how to build a flow model from a base distribution, a list of flow layers, and, optionally, a target distribution. They can be trained by computing a loss through the respective methods provided and minimizing it with the standard PyTorch optimizers. We show how to approximate simple 2D distributions, but, moreover, apply flows to images through the multiscale architecture (Dinh et al., 2017), which `normflows` provides as well. Furthermore, there is an example of how to build a variational autoencoder with normalizing flows as well.

Here, we want to illustrate a strength of `normflows`, i.e. that it can deal with combinations of standard and circular coordinates. Therefore, we consider a distribution of two random variables, x and ϕ. x follows a Gaussian distribution with density $p(x) = \mathcal{N}(x|0, 1)$ and ϕ has a circular von Mises distribution such that $p(\phi|x) = \mathcal{M}(\phi|\mu(x), 1)$ with $\mu(x) = 3x$. We train a Neural Spline Flow with an unbound and a circular coordinate to approximate the target distribution $p(x, \phi) = p(x)p(\phi|x)$ by minimizing the reverse Kullback-Leibler divergence. As shown in Figure 1, the density of the flow is almost indistinguishable from the target.

Acknowledgements

We thank Laurence Midgley and Timothy Gebhard for their valuable contributions to the package. Moreover, we thank everyone who contacted us via mail or on GitHub for the valuable feedback and spotting bugs.

José Miguel Hernández-Lobato acknowledges support from a Turing AI Fellowship under grant EP/V023756/1. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B; and by the Machine Learning Cluster of Excellence, EXC number 2064/1 - Project number 390727645.

References

Ardizzone, L., Bungert, T., Draxler, F., Köthe, U., Kruse, J., Schmier, R., & Sorrenson, P. (2018-2022). *Framework for Easily Invertible Architectures (FrEIA)*. https://github.com/vislearn/FrEIA

Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., … Viola, F. (2020). *The DeepMind JAX Ecosystem*.

Campbell, A., Chen, W., Stimper, V., Hernandez-Lobato, J. M., & Zhang, Y. (2021). A gradient based strategy for Hamiltonian Monte Carlo hyperparameter optimization. *Proceedings of the 38th International Conference on Machine Learning*, 1238–1248.

Chen, R. T. Q., Behrmann, J., Duvenaud, D. K., & Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, *32*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*, *31*.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., & Saurous, R. A. (2017). TensorFlow Distributions. *arXiv Preprint arXiv:1711.10604*. https://doi.org/10.48550/arXiv.1711.10604

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2017). Density estimation using Real NVP. *International Conference on Learning Representations*.

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2019). Neural spline flows. *Advances in Neural Information Processing Systems*, *32*, 7511–7522.

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2020). nflows: Normalizing flows in PyTorch. *Zenodo*. https://doi.org/10.5281/zenodo.4296287

Grcić, M., Grubišić, I., & Šegvić, S. (2021). Densely connected normalizing flows. *Advances in Neural Information Processing Systems*, *34*.

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, *31*.

Kobyzev, I., Prince, S. J. D., & Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3964–3979. https://doi.org/10.1109/TPAMI.2020.2992934

Midgley, L. I., Stimper, V., Simm, G. N. C., Schölkopf, B., & Hernández-Lobato, J. M. (2023). Flow Annealed Importance Sampling Bootstrap. *International Conference on Learning Representations*.

Müller, T., McWilliams, B., Rousselle, F., Gross, M., & Novák, J. (2019). Neural importance sampling. *ACM Transactions on Graphics (TOG)*, *38*(5), 1–19.

Nielsen, D., Jaini, P., Hoogeboom, E., Winther, O., & Welling, M. (2020). SurVAE flows: Surjections to bridge the gap between VAEs and flows. *Advances in Neural Information Processing Systems 33*.

Noé, F., Olsson, S., Köhler, J., & Wu, H. (2019). Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, *365*(6457). https://doi.org/10.1126/science.aaw1147

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, *22*(57), 1–64.

Papamakarios, G., Pavlakou, T., & Murray, I. (2017). Masked autoregressive flow for density estimation. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2335–2344.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M.,

Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035).

Rezende, D. J., Papamakarios, G., Racanière, S., Albergo, M. S., Kanwar, G., Shanahan, P. E., & Cranmer, K. (2020). Normalizing flows on tori and spheres. *Proceedings of the 37th International Conference on Machine Learning*, *119*, 8083–8092.

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on Machine Learning*, 1530–1538.

Stimper, V., Campbell, A., & Hernández-Lobato, J. M. (2023). Implementing Boltzmann generators with normflows. *Zenodo*. https://doi.org/10.5281/zenodo.7565800

Stimper, V., Schölkopf, B., & Hernández-Lobato, J. M. (2022). Resampling Base Distributions of Normalizing Flows. *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, *151*, 4915–4936.

Tabak, Esteban G., & Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, *66*(2), 145–164. https://doi.org/10.1002/cpa.21423

Tabak, Esteban G., & Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, *8*(1), 217–233. https://doi.org/10.4310/CMS.2010.v8.n1.a11

Wang, P. Z., & Wang, W. Y. (2019). Riemannian normalizing flow on variational Wasserstein autoencoder for text modeling. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Wu, H., Köhler, J., & Noe, F. (2020). Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, *33*, 5933–5944.