normflows: A PyTorch Package for Normalizing Flows

Vincent Stimper, David Liu, Andrew Campbell, Vincent Berenz, Lukas Ryll, Bernhard Schölkopf, and José Miguel Hernández-Lobato

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

DOI: 10.21105/joss.05361

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, Neur al 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 $\phi$. 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.](image)

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 $\phi$. $x$ follows a Gaussian distribution with density $p(x) = \mathcal{N}(x|0, 1)$ and $\phi$ 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


